UNIVERSIDAD POLITECNICA DE CANARIAS
ESCUELA UNIVERSITARIA DE TELECOMUNICACIONES

Proyecto:

SIMULACION, ANALISIS DINAMICO Y PRUEBAS DE COMPORTAMIENTO DEL PROCESADOR CPA.

AUTOR: JOSE CARLOS PULIDO ACOSTA.
TUTOR: DR. ANTONIO NUÑEZ ORDOÑEZ.
CO-TUTOR: JAVIER LOPEZ CURBELO.

LAS PALMAS DE G.C., 19 DE MAYO DE 1989.

INDICE

SIMULACION, ANALISIS DINAMICO Y PRUEBAS DE COMPORTAMIENTO DEL PROCESADOR CPA.

I. OBJETO DEL PROYECTO

El objetivo del proyecto ha sido, siguiendo la línea de trabajo de diseño microelectrónico del Departamento de Electrónica y Telecomunicación de la UPC, emular el microprocesador i8085, utilizando una tecnología de alta velocidad. Como único requisito se establece el respetar el aspecto funcional, es decir que nuestro procesador ha de tener el mismo número de pines, proporcionar las mismas señales de salida e interpretar el conjunto de instrucciones del i8085.

Se ha hecho una simulación a nivel de registros utilizando el paquete de software MICRO, con el que se puede obtener un análisis dinámico y proporcionar así una comparación real en cuanto a velocidad de proceso y en consecuencia efectividad, entre CPA e i8085.

Este trabajo se ha desarrollado conjuntamente con el Proyecto Fin de Carrera de F. Caballero, en el que se hace un estudio profundo a nivel eléctrico del hardware de CPA, por tanto aquí sólo se comentan los aspectos más relevantes de la arquitectura a modo general.

El diseño se ha realizado utilizando bloques bit-slice de la familia Am2900 en tecnología ECL, con el propósito de sustituir los dispositivos por sus equivalentes en versión en tecnología AsGa, trabajo que se desarrolla actualmente así como una implementación de la arquitectura diseñada en células estándar CMOS.

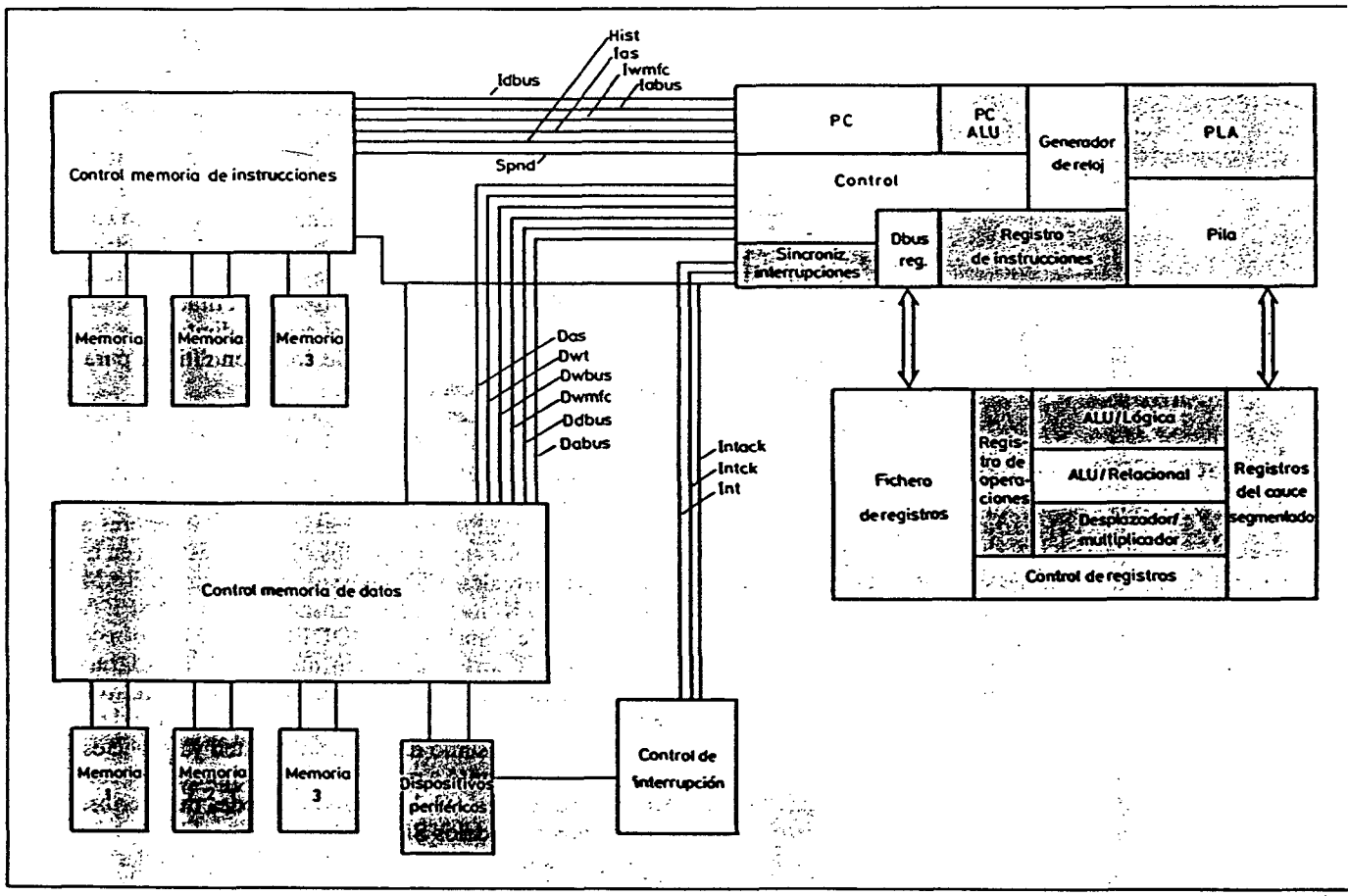
II. ANTECEDENTES

2.1. OTROS PROCESADORES DEL DET.

2.1.1 MVM.

Microprocesador de 16 bits, de arquitectura de migración vertical modificada, adaptada a la tecnología de AsGa y a aplicaciones de rápida respuesta y muy alta productividad de instrucciones. Implantación con tecnología E/D-MESFET DCFL, proceso de 1 µm, con un reloj de 400 MHz.

Microprocesador realizado por A. Nuñez en la Escuela de Ingeniería Electrica de la Universidad de Purdue en Indiana, USA.



El núcleo originario de este trabajo ha sido la arquitectura GVM (Generalized Vertical Migration). que migra verticalmente, en sentido descendente, la interpretación de construcciones de alto nivel, desde el compilador al lenguaje de máquina, mediante el sistema de compilación directa a microcódigo, de forma que la distancia semántica entre el lenguaje y la máquina es mínima, y el código objeto generado por el compilador corresponde ya al nivel de microprograma, interpretado directamente por el hardware.

MVM (Modified Vertical Migration), se ha concebido partiendo de la versión reducida para integración en AsGa de GVM, denominada RVM (Reduced Vertical Migration), potenciando considerablemente la arquitectura con: un tercer elemento de proceso de operación en paralelo, una ampliación del fichero de registro, segmentación del cauce de instrucción. segmentación del cauce de ejecución, mejor mecanismo de control de interrupciones, ejecución de instrucciones por ciclo polifásico variable, y buses de datos e instrucciones asíncronos.

El procesador MVM soporta construcciones de lenguajes de alto nivel. Las instrucciones del nivel de máquina convencional pertenecen en realidad al nivel de

microinstrucciones y su interpretación se hace directamente en un solo nivel.

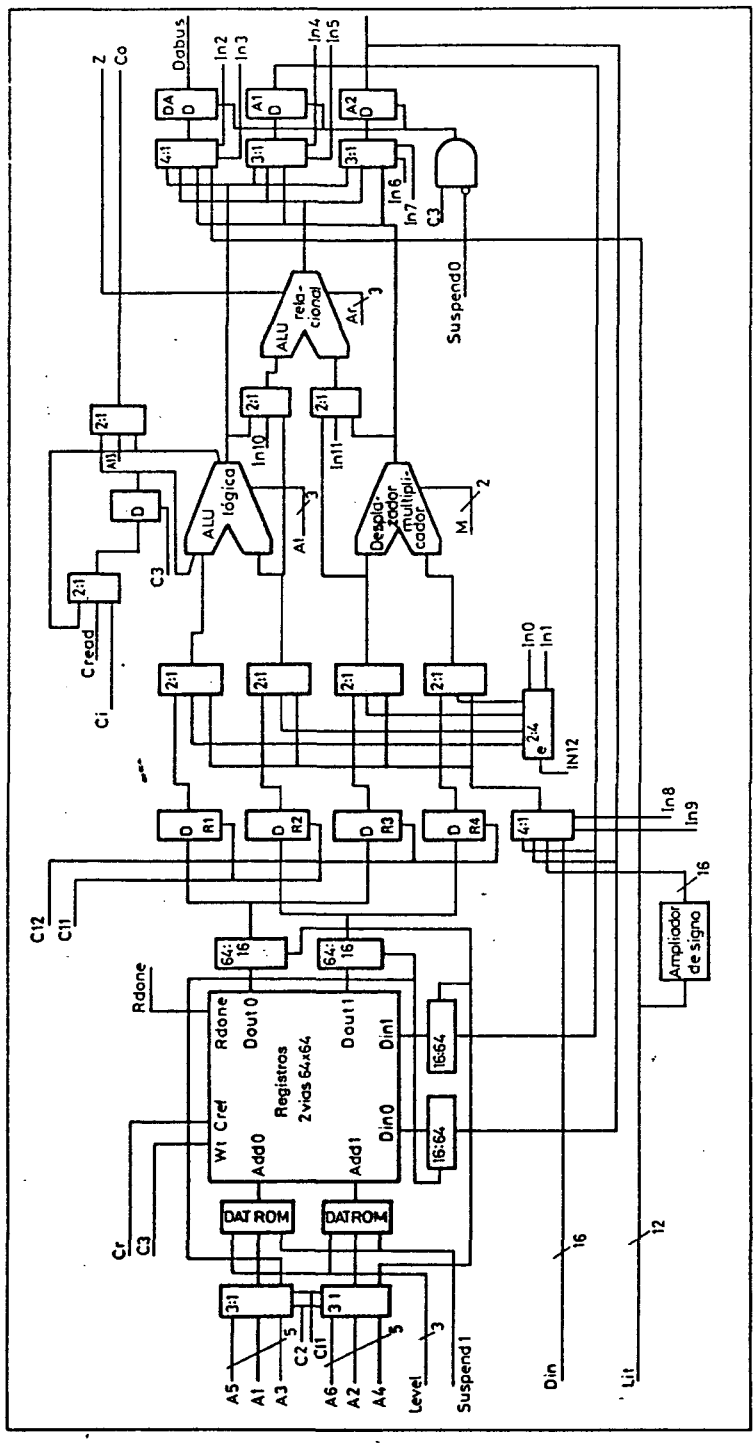


Figura Unidad de ejecución (EU).

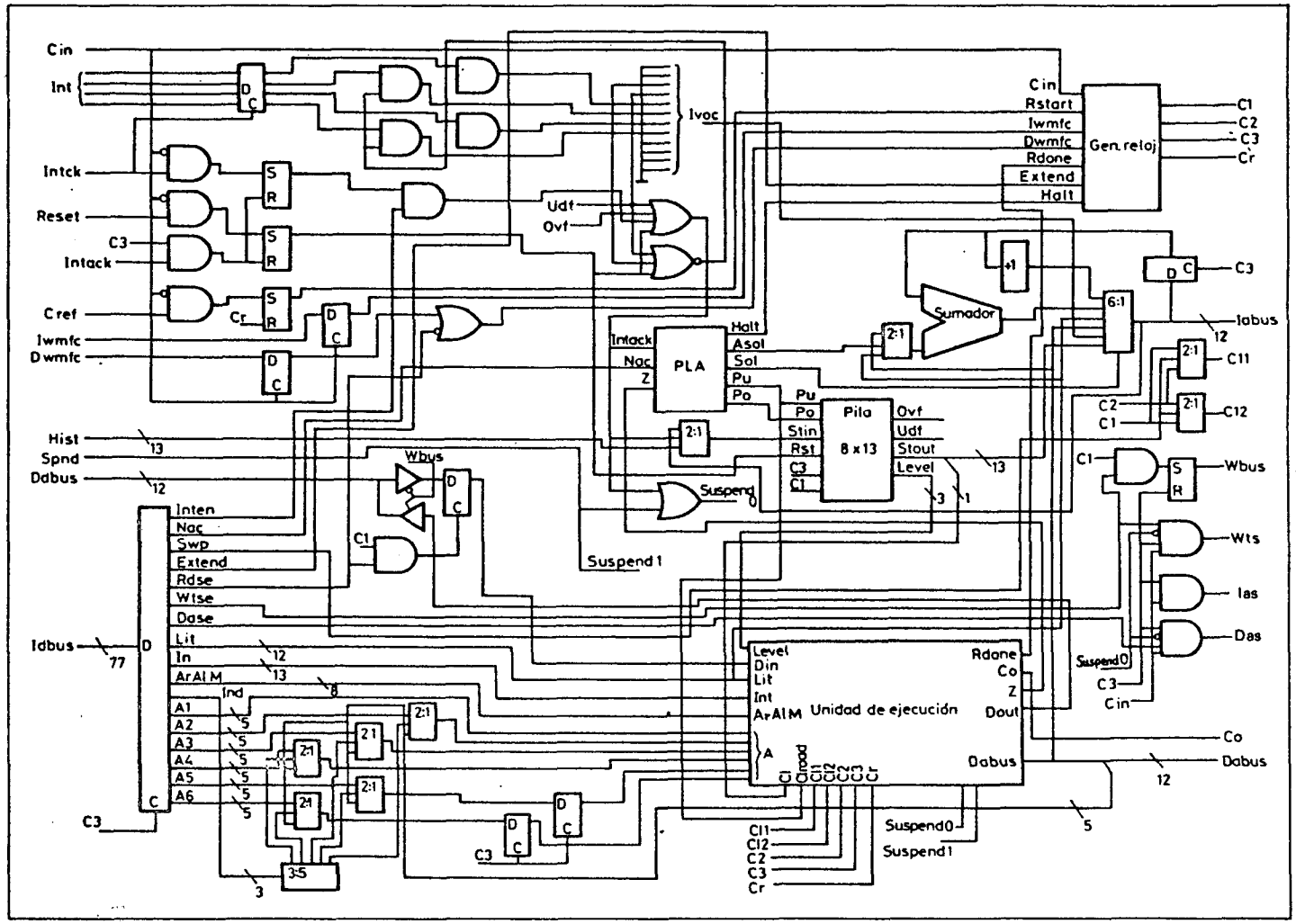


Figura Unidad de control (CU).

2.1.3 MC_AC.

Este microprocesador es el resultado de una experiencia de diseño full-custom. Se trata de un microcontrolador para un sistema de acceso por clave, en el que mediante la introducción de una clave de cuatro dígitos, se activa un sistema de apertura.

El microprocesador posee una estructura pipeline de dos etapas. En su arquitectura podemos destacar una ROM, de 464 palabras de 12 bits, donde reside el microprograma; una RAM, de 22 palabras de cuatro bits, direccionada mediante dos puertos, en la que se almacenan las variables del programa de control y la clave del usuario; una ALU de cuatro bits y una RAM para mapear la salida de datos al display.

Tanto en la estructura del chip, como en la superficie utilizada, este microprocesador es de poca complejidad. Ha sido diseñado con tecnología NMOS, utilizando técnicas de diseño y fabricación asistido por ordenador (CAD/CAM), que permitían desde el trazado de transistores y pistas de conexión sobre el silicio, mediante programas de diseño gráfico interactivo.

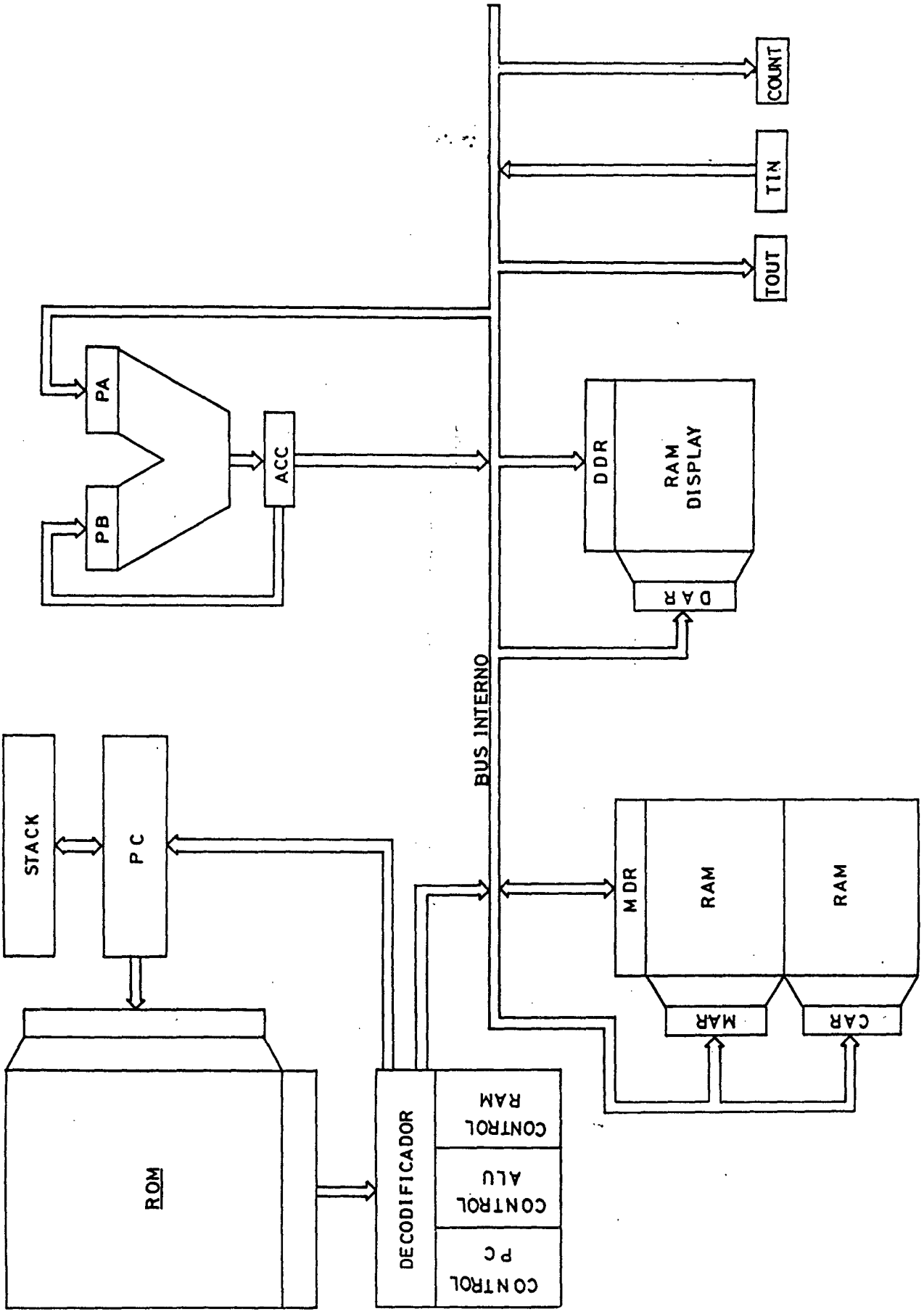


Figura. Diagrama de bloques de MC_AC.

2.1.4 PL/O

Procesador de 16 bits, de arquitectura a stack, orientado a lenguaje de alto nivel e ideado para ejecutar programas en PL/O. La implementación en bloques LSI de este procesador fue realizada en la Escuela Politécnica de Laussane, Suiza, por A. Nuñez.

PL/O es un PASCAL reducido y como tal un lenguaje recursivo, estructurado en bloques, fue introducido para mostrar las diferentes etapas en el desarrollo de un compilador de lenguaje.

La unidad de tratamiento está formada por la memoria de programa de palabras de 16 bits, la memoria de datos en forma de stack de 16 bits, la ALU y un banco de registros.

La unidad de control está formada por una memoria de 8 bits de mapeado de las diferentes instrucciones de lenguaje máquina en el microprograma, una memoria de microprograma, un registro direccionador y un secuenciador, de la memoria de microprograma.

La memoria de programa contiene el código objeto generado por el compilador PL/O. La memoria de datos está organizada como una pila, en la cual las dos palabras de la parte superior de la misma son los operandos de los operadores aritméticos, siendo

restituidas por el resultado en cada operación.

El procesador se diseñó a stack para facilitar la creación y activación de los bloques de datos que son invocados de manera recursiva, así como toda estructura de bloques de un programa en general.

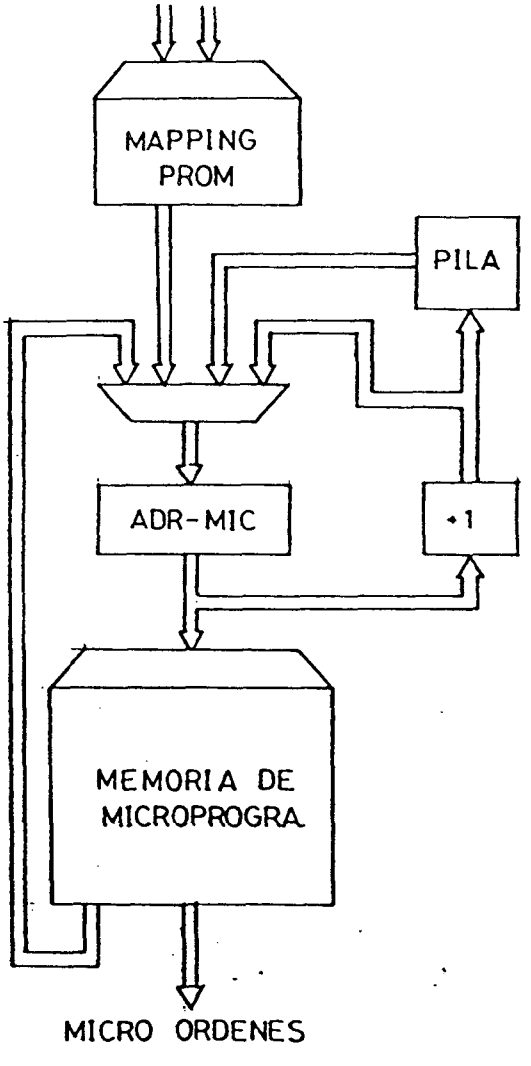


Figura. Unidad de control de PL/O.

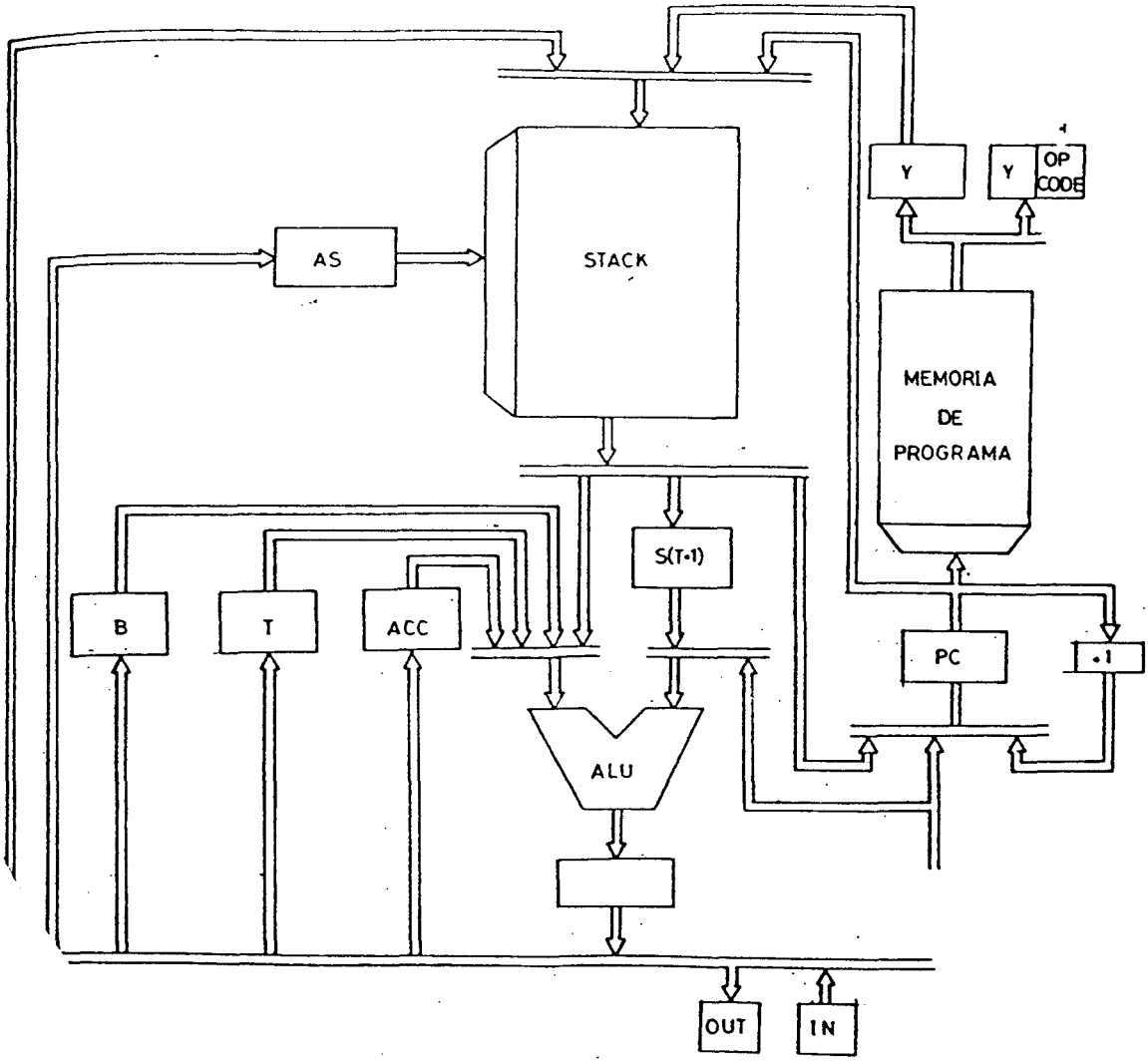


Figura. Unidad de ejecución de PL/0.

2.2. HERRAMIENTAS CAD EN DISEÑO DE PROCESADORES.

Herramientas que tiene reunidas actualmente el departamento para su actividad de Diseño Microelectrónico de Sistemas Digitales:

*Como ayuda a concepción

-MICRO y SILOSS.

-KARL III y SCIL.

-MICRO86.

-Utilidades LEX y YACC en UNIX.

*Como ayuda al diseño y lenguajes de diseño hardware (HLDL)

-Captura de esquemas orCAD/SDT-III en PC's.

-Captura de esquemas SOLO 2000 en SUN.

-Soporte ASIC, librerias, descripción circuital y simulación lógica

PPDS (Philips Personal Design Station),

TDU (Transportable Design Utility) Texas Instrument.

-Soporte PLD (Programable Logic Devices)

AMAZE de Philips.

-HILO III y SOLO 2000.

*Simulación electrica

SPICE en SUN y VAX,

PSPICE en PC's y SUN.

*Editores de trazado

Colocación e interconexión de SOLO 2000,

Colocación e interconexión de Philips.

*Editores layout full-custom

LUCIE-C

KIC-2

MAGIC

SOLO 2000

*Extractores circuitales SOLO 2000 y MAGIC.

2.2.1 MICRO.

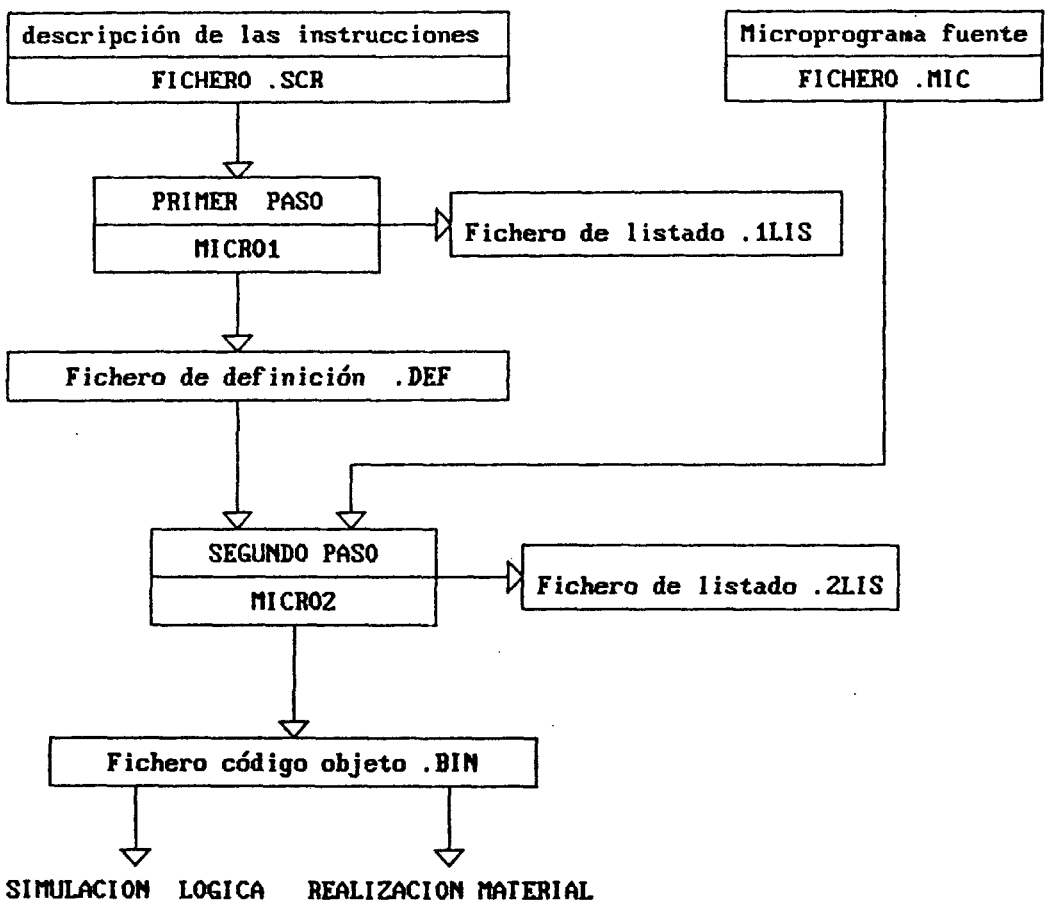
Este paquete de software (Micro y Siloss) fue desarrollado en el **Politécnico Federal de Laussane** (**EPFL**, Suiza 1981). Se aplica al diseño **LSI** con bloques **funcionales** o con bloques **bit-slice**, y al diseño **VLSI** **custom** o **semicustom**. Adecuado para la realización de circuitos integrados de aplicación específica (**ASICs**) de cierto nivel de complejidad, y para procesadores de propósito general.

MICRO es un **microensamblador** de **definición** **variable**. Permite la elaboración en **lenguaje simbólico**, definido por el usuario, de un **microprograma** y realiza la transformación del mismo a **código objeto**. Entre sus ventajas más notorias está: una escritura simple y comprensiva del microprograma, independencia frente a la estructura material del sistema y facilidad de corrección. Es capaz de adaptarse a cualquier tipo de máquina y de **microinstrucción**.

En un primer paso reconoce la definición de los formatos de microinstrucción (longitud, campos, valores por defecto,...) y la interpretación de los diferentes **mnemónicos** de las microoperaciones y parámetros a utilizar posteriormente. En el segundo paso reconoce el microprograma escrito en lenguaje definido

en el primer paso y realiza la traducción a código objeto.

Estructura general de MICRO

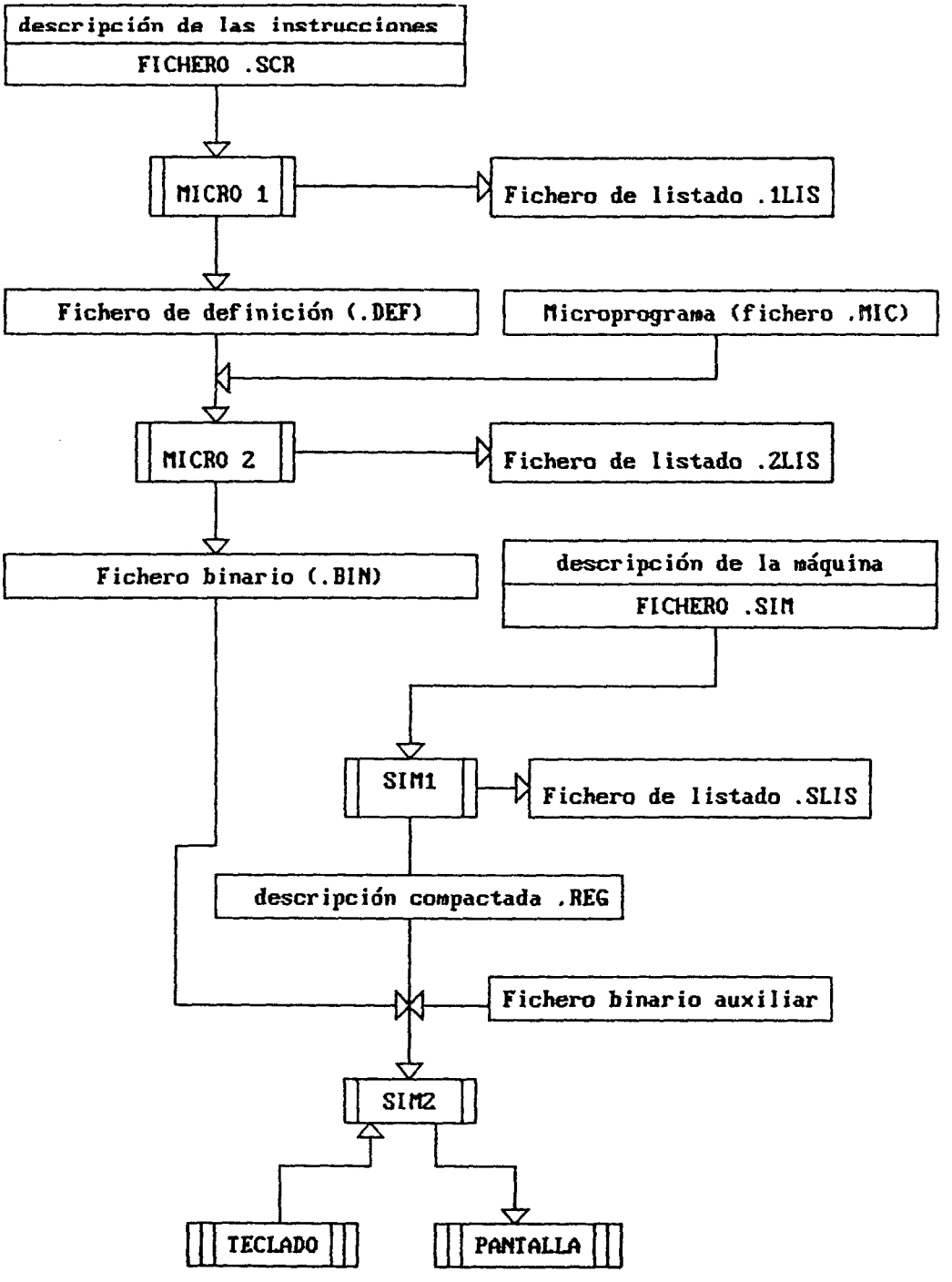


SILOSS es un simulador lógico de sistemas secuenciales a nivel de transferencia de registros. Permite el testeo del microprograma, utilizando el fichero de salida generado por el microensamblador.

Una descripción de los elementos e interconexiones de la estructura interna del sistema mediante un lenguaje no procedural, (el orden de escritura de las microoperaciones no influye en el orden de ejecución) nos describe posteriormente en la fase de simulación el comportamiento del sistema en un período de reloj. Todas las transferencias serán leidas en paralelo en dicho período.

La estructura de control no secuencia la ejecución como en el caso de los lenguajes de programación sino que elige, en función de las condiciones predefinidas, las microoperaciones que serán ejecutadas. La secuencia de microoperaciones se da implícitamente por el microprograma, que es el que modifica las condiciones de ejecución.

El simulador permite también la comprobación y depurado del microprograma ya que, la ejecución se puede realizar paso a paso o con puntos de parada definidos por el usuario.



Figura

2.2.2 MICRO86.

Microensamblador elaborado en el **Centro de Proceso de Datos de la Universidad de Valladolid** (1986), sobre VAX-11/780. También, la misión de este **traductor** es obtener el **microcódigo** de una CPU cualquiera a partir de un fichero fuente escrito con mnemónicos fijados por el programador.

La parte más importante del programa microensamblador se ha escrito con el **compilador** VAX-11 PASCAL, sin embargo un conjunto de rutinas que se ejecutan muchas veces en el proceso de traducción se han escrito con el **macroensamblador** VAX-11 MACRO. Esto proporciona mayor rapidez en el tiempo de microensamblado.

2.2.3 KARL-III.

Es un lenguaje de transferencia de registros no procedural que incluye primitivas para descripción de hardware a nivel de transferencia de registro, nivel de puertas, nivel de circuitos y mezcla de estos niveles. Ha sido elaborado en la Universidad de Kaiserlautern, R.F.A (1986).

A nivel de circuito KARL-III usa una notación independiente de la tecnología utilizada para emular toda clase de transistores, etc. Incluye también un modelo para circuiteria dinámica.

Las primitivas funcionales de KARL-III incluyen operadores lógicos, aritméticos, relacionales, varias funciones standard, elementos de hardware tales como registros, ROM, RAM, clocks, switches, etc.

Implementado en PASCAL se ejecuta sobre VMS en VAX-11. No es sólo una herramienta de diseño y especificación, incluye también el simulador SCIL, un editor de diagramas de bloques gráficos (ABLED), el preprocesador hyperKARL y el sistema de verificación VERENA.

2.2.4 Utilidades YACC y LEX.

YACC.- Es un compilador de compiladores, escrito en lenguaje C. Proporciona una herramienta general para describir el lenguaje de entrada de un computador; este puede ser tan complejo como un lenguaje de programación o tan simple como una secuencia de números.

El usuario de YACC especifica la estructura de su entrada, junto con el código que será invocado cuando se reconozca dicha estructura. YACC da paso a una subrutina que maneja el proceso de entrada haciendo una llamada al analizador léxico.

LEX.- Es un generador de analizadores léxicos soportado sobre UNIX y diseñado para simplificar su conexión con YACC. Acepta un problema orientado a la especificación de cadenas de caracteres del mismo tipo, y produce un programa en un lenguaje de propósito general que reconoce expresiones regulares.

III. PROCESADOR CPA Y MICROPROGRAMA

INTRODUCCION.

La versión en silicio de CPA se ha diseñado utilizando la familia bit-slice de AMD, Am2900. Esta familia, formada por bloques bipolar LSI, se usa en aplicaciones que requieren gran velocidad de proceso, velocidad que no proporcionan las tecnologías MOS.

AMD es uno de los fabricantes más importantes de chips destinados a la construcción de microproocesadores bipolares. En su esfuerzo por lograr mejores prestaciones ha llegado a optimizar las características de los circuitos integrados bipolares, apoyándose en dos aspectos básicos:

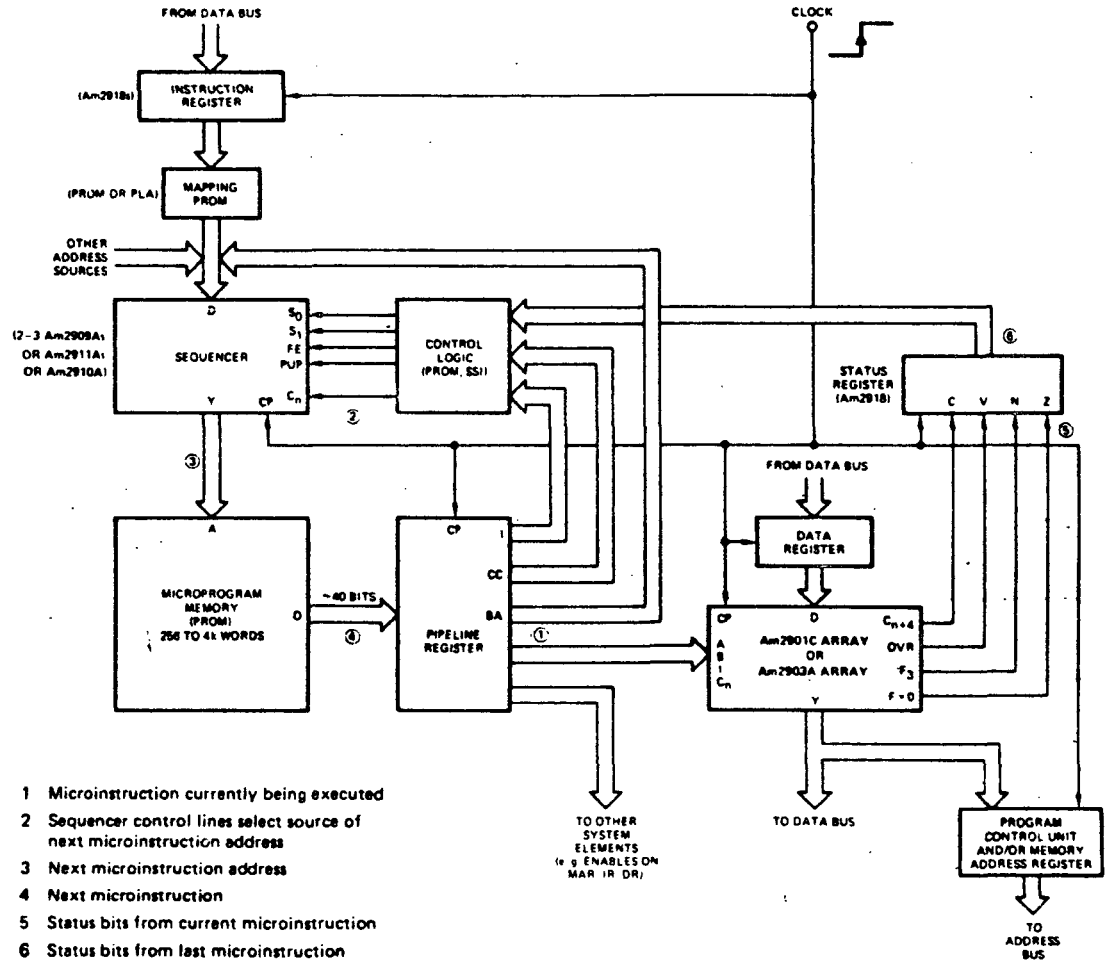
-La utilización de la tecnología ECL interna y TTL externa, para lograr los niveles lógicos adecuados y compatibles con el resto de componentes electrónicos, hace posible una velocidad de proceso muy grande. La tecnología ECL que utiliza AMD para estos productos es de menor consumo que las clásicas ECL 10K y 100K. La utilización de TTL para conseguir los niveles lógicos precisos, sólo empeora la rapidez del sistema en un 7-15% .

-El proceso de fabricación IMOX (Ion-IMplantacion OXide-isolation), que empezó a utilizarse en el año 1980, con el que se consiguió

disminuir la superficie de silicio necesaria y las capacidades parásitas. En 1983 se mejoró este proceso dando lugar al IMOX-S2, con el cual se alcanzó mayores densidades de integración, a la vez que se logró un aumento de la velocidad del orden del 30% .

Cada bloque es lo suficientemente flexible y expandible como para emular a la mayoría de los sistemas microprogramados existentes. En la figura siguiente se ilustra la arquitectura de un sistema típico. En ella se pueden diferenciar las dos partes constituyentes: la unidad de control y la unidad de ejecución.

Figure



1 Microinstruction currently being executed
2 Sequencer control lines select source of next microinstruction address
3 Next microinstruction address
4 Next microinstruction
5 Status bits from current microinstruction
6 Status bits from last microinstruction

El direccionamiento de las microinstrucciones es generado por el secuenciador, comenzando desde un flanco de reloj. La dirección va desde el secuenciador a la ROM. A la salida de la memoria de microprograma , el registro pipeline contiene la microinstrucción en curso que va a ser ejecutada (1) . Los bits para el control de la unidad de ejecución se situan en los elementos del sistema, a la vez que una porción de la microinstrucción retorna al secuenciador (2) para determinar la dirección de la próxima microinstrucción que va a ser ejecutada. La dirección se envía a la ROM y la siguiente microinstrucción (4) se situa a la entrada del registro pipeline. Mientras los 2901Cs están ejecutando una microinstrucción, la siguiente microinstrucción está siendo fecheada en la ROM. Nótese que no hay una secuencia lógica en el secuenciador, entre la salida y la línea seleccionada. Esto es importante ya que el lazo (1) a (2) a (3) a (4) debe ocurrir durante un simple ciclo de reloj. Durante el mismo tiempo se producirá el lazo de (1) a (5) .

3.1 ARQUITECTURA DEL PROCESADOR.

La unidad de control de CPA corresponde a la típica de un sistema microprogramado horizontalmente.

El código de operación de una instrucción que viene desde la memoria de programa externa, se almacena en el registro de instrucción; a continuación es mapeado en la PROM. Esta memoria está formada por 256 palabras de 10 bits. Cada palabra, cuya dirección en la PROM corresponde al opcode del conjunto de instrucciones del i8085, es una dirección de la memoria de microprograma. A partir de esta dirección de la micromemoria, se encuentra el conjunto de microinstrucciones que genera las microoperaciones que ejecutan cada instrucción.

Un total de 536 microinstrucciones de 64 bits cada una, distribuidos en 30 campos, conforman la memoria de microprograma.

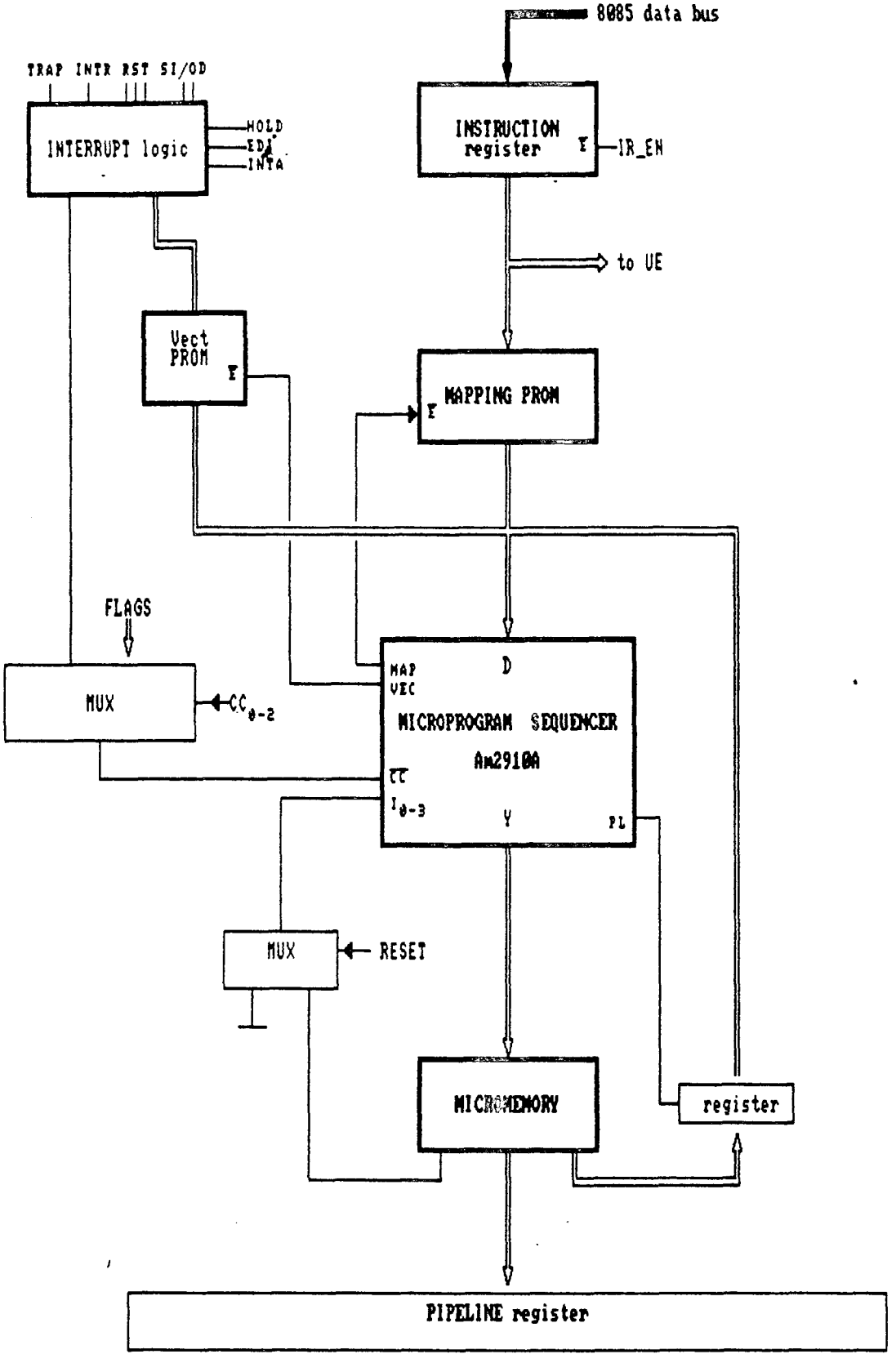


Figura. Unidad de control de CPA.

El controlador de microprograma Am2910A es un secuenciador de direcciones propuesto para controlar la secuencia de ejecución del conjunto de microinstrucciones almacenado en la memoria de microprograma.

Durante cada microinstrucción, el secuenciador proporciona una dirección de 12 bits de una de las fuentes siguientes: del registro de dirección de microprograma (μPC), de una entrada directa (D), o de una pila LIFO de cinco palabras.

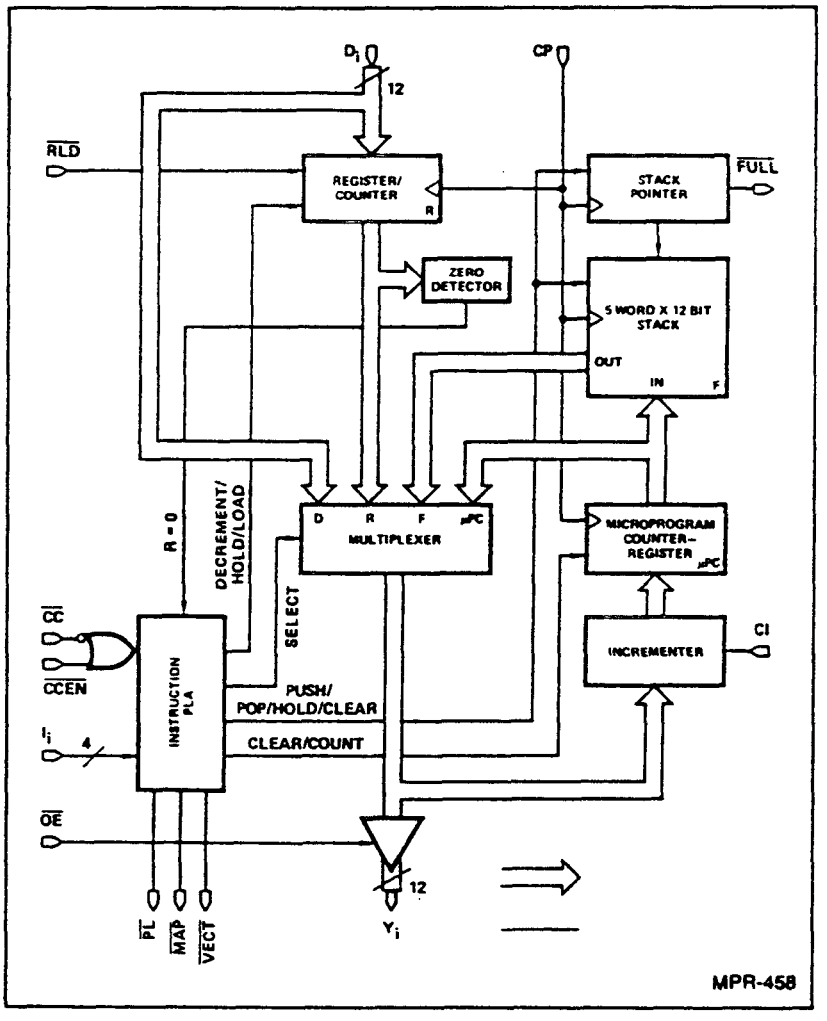


Figure . Am2910 Block Diagram.

A su vez el secuenciador controla mediante las señales $\overline{PL}$, $\overline{MAP}$, y $\overline{VECT}$, si la entrada D proviene de una porción del registro pipeline, de la mapping PROM o de una dirección de interrupción, respectivamente.

El bit de salida del multiplexor de condiciones junto con el campo secuencia de la próxima microinstrucción (I0-3), determinan la opción a tomar.

HEX I_{3-I_0}	MNEMONIC	NAME	REG/ CNTR CON-TENTS	FAIL $\overline{CCEN}$ = LOW and $\overline{CC}$ = HIGH		PASS $\overline{CCEN}$ = HIGH or $\overline{CC}$ = LOW		REG/ CNTR	ENABLE
				Y	STACK	Y	STACK		
0	JZ	JUMP ZERO	X	0	CLEAR	0	CLEAR	HOLD	PL
1	CJS	COND JSB PL	X	PC	HOLD	D	PUSH	HOLD	PL
2	JMAP	JUMP MAP	X	D	HOLD	D	HOLD	HOLD	MAP
3	CJP	COND JUMP PL	X	PC	HOLD	D	HOLD	HOLD	PL
4	PUSH	PUSH/COND LD CNTR	X	PC	PUSH	PC	PUSH	Note 1	PL
5	JSRP	COND JSB R/PL	X	R	PUSH	D	PUSH	HOLD	PL
6	CJV	COND JUMP VECTOR	X	PC	HOLD	D	HOLD	HOLD	VECT
7	JRP	COND JUMP R/PL	X	R	HOLD	D	HOLD	HOLD	PL
8	RFCT	REPEAT LOOP, CNTR ≠ 0	≠0	F	HOLD	F	HOLD	DEC	PL
			=0	PC	POP	PC	POP	HOLD	PL
9	RPCT	REPEAT PL, CNTR ≠ 0	≠0	D	HOLD	D	HOLD	DEC	PL
			=0	PC	HOLD	PC	HOLD	HOLD	PL
A	CRTN	COND RTN	X	PC	HOLD	F	POP	HOLD	PL
B	CJPP	COND JUMP PL & POP	X	PC	HOLD	D	POP	HOLD	PL
C	LDCT	LD CNTR & CONTINUE	X	PC	HOLD	PC	HOLD	LOAD	PL
D	LOOP	TEST END LOOP	X	F	HOLD	PC	POP	HOLD	PL
E	CONT	CONTINUE	X	PC	HOLD	PC	HOLD	HOLD	PL
F	TWB	THREE-WAY BRANCH	≠0	F	HOLD	PC	POP	DEC	PL
			=0	D	POP	PC	POP	HOLD	PL

Note: If $\overline{CCEN}$ = LOW and $\overline{CC}$ = HIGH, hold; else load. X = Don't Care.

Figura. Instrucciones del secuenciador.

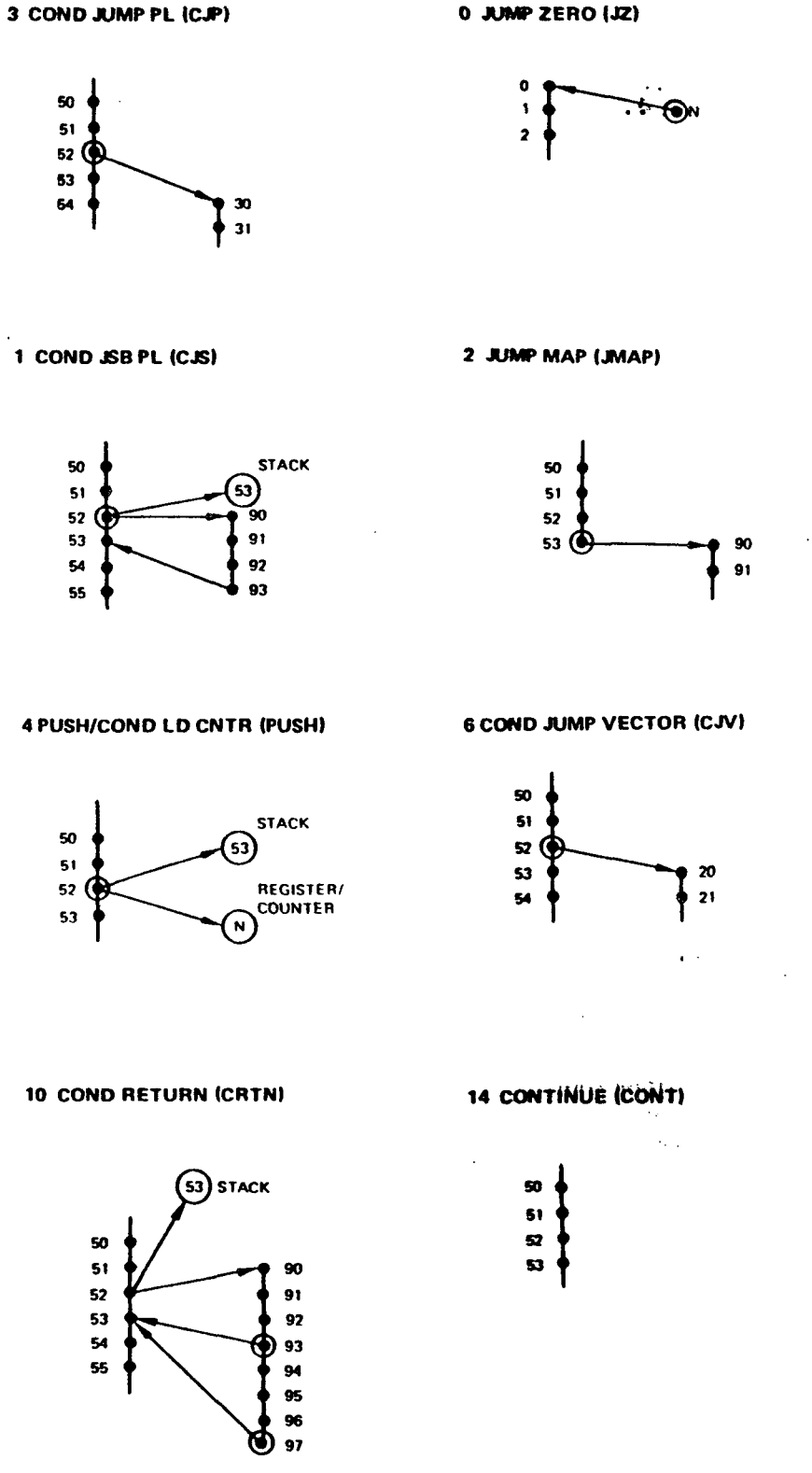


Figura. Diagrama del conjunto de instrucciones del Am2910A que se han utilizado en CPA.

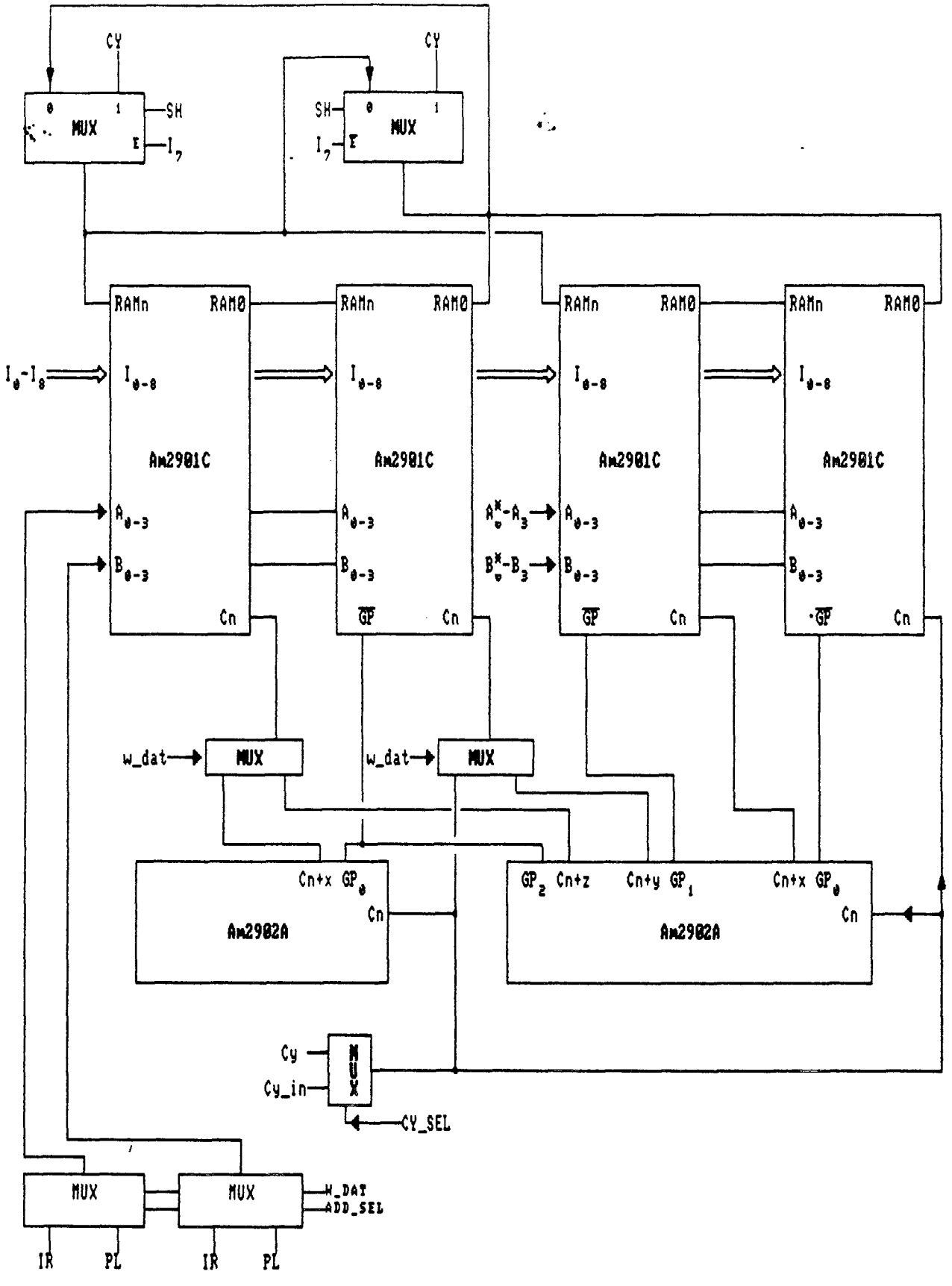


Figura. Esquema 1 de la UE de CPA.

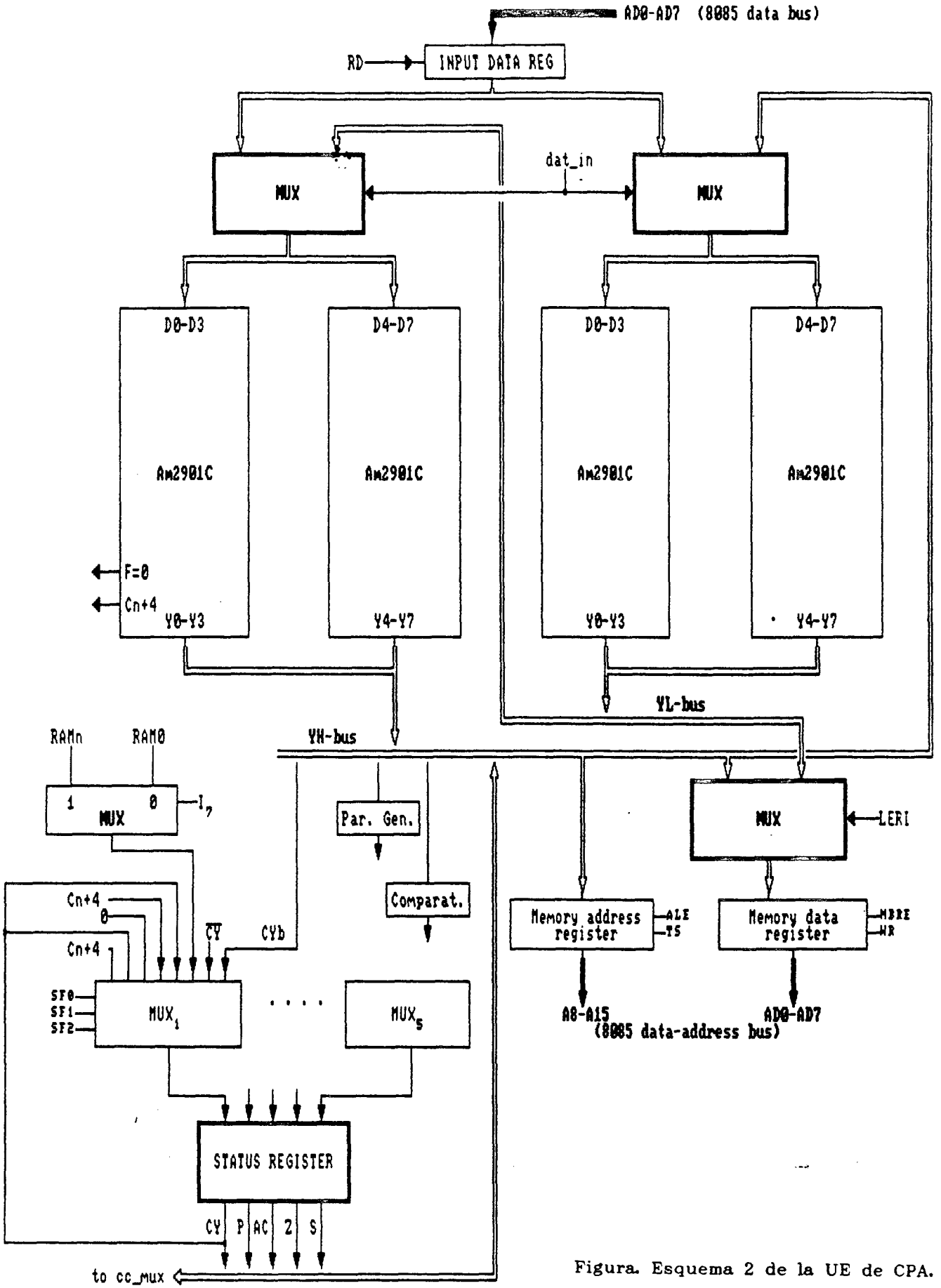


Figura. Esquema 2 de la UE de CPA.

La unidad de ejecución está formada por cuatro Am2901C, interconectados en paralelo a través de dos generadores de acarreo paralelo rápido (carry look-ahead), multiplexados, que nos permiten realizar operaciones de 8/16 bits.

Los elementos claves de cada Am2901C son la RAM de dos puertos, de 16 palabras de 4 bits, y la ALU de alta velocidad. La memoria RAM interna la utilizamos para disponer los registros de propósito general.

Cualquiera de las 16 palabras de la RAM pueden ser leidas a través de los puertos A y B, direccionando cada uno por medio de los campos de cuatro bits A y B indistintamente.

La entrada de datos a la RAM pasa por multiplexores de tres entradas. Esta configuración se usa para realizar desplazamientos sobre el dato de salida de la ALU, (F); un bit hacia arriba, un bit hacia abajo o no desplazar.

La salida de datos del puerto A de la RAM y del puerto B se separan del resto mediante latches de cuatro bits. Estos latches cierran la salida de la RAM mientras el reloj este a nivel bajo. Así, se eliminan posibles realimentaciones indeseadas, cuando se está escribiendo un dato en la RAM. En la escritura sólo se utiliza la entrada de direcciones B.

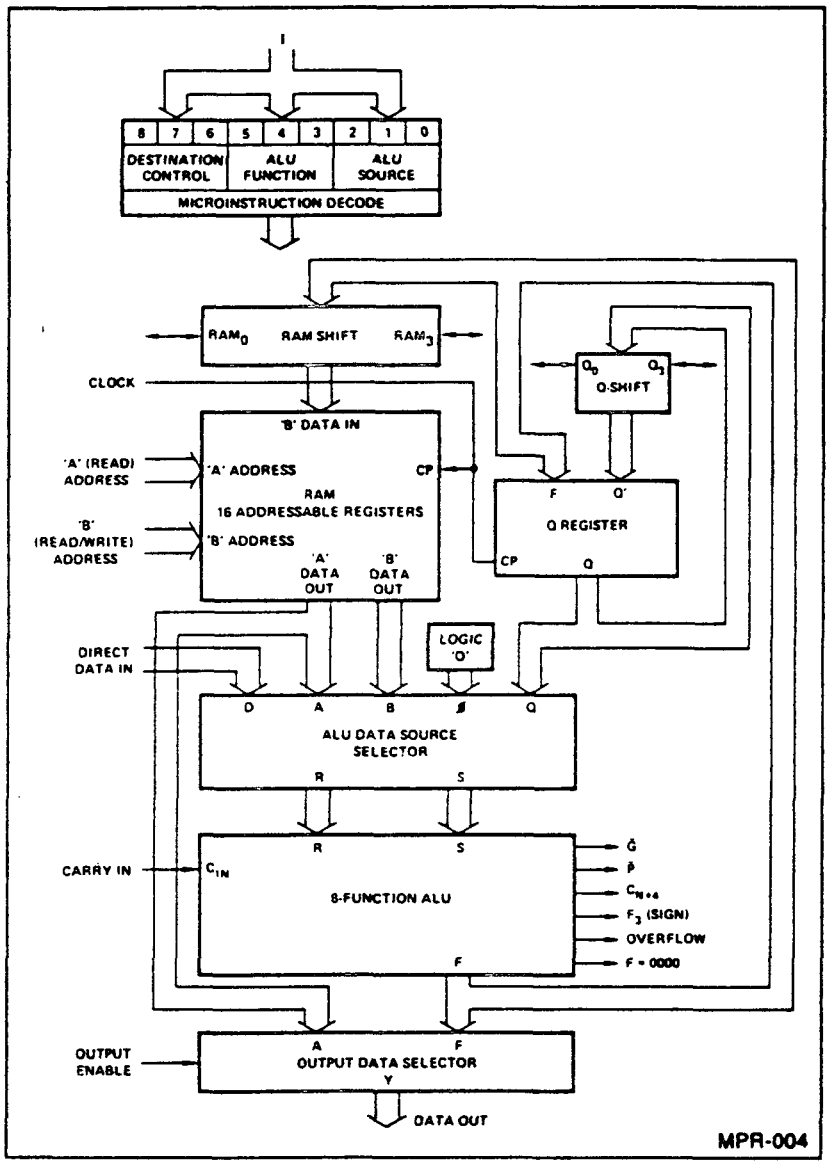


Figure ''. Am2901A Block Diagram.

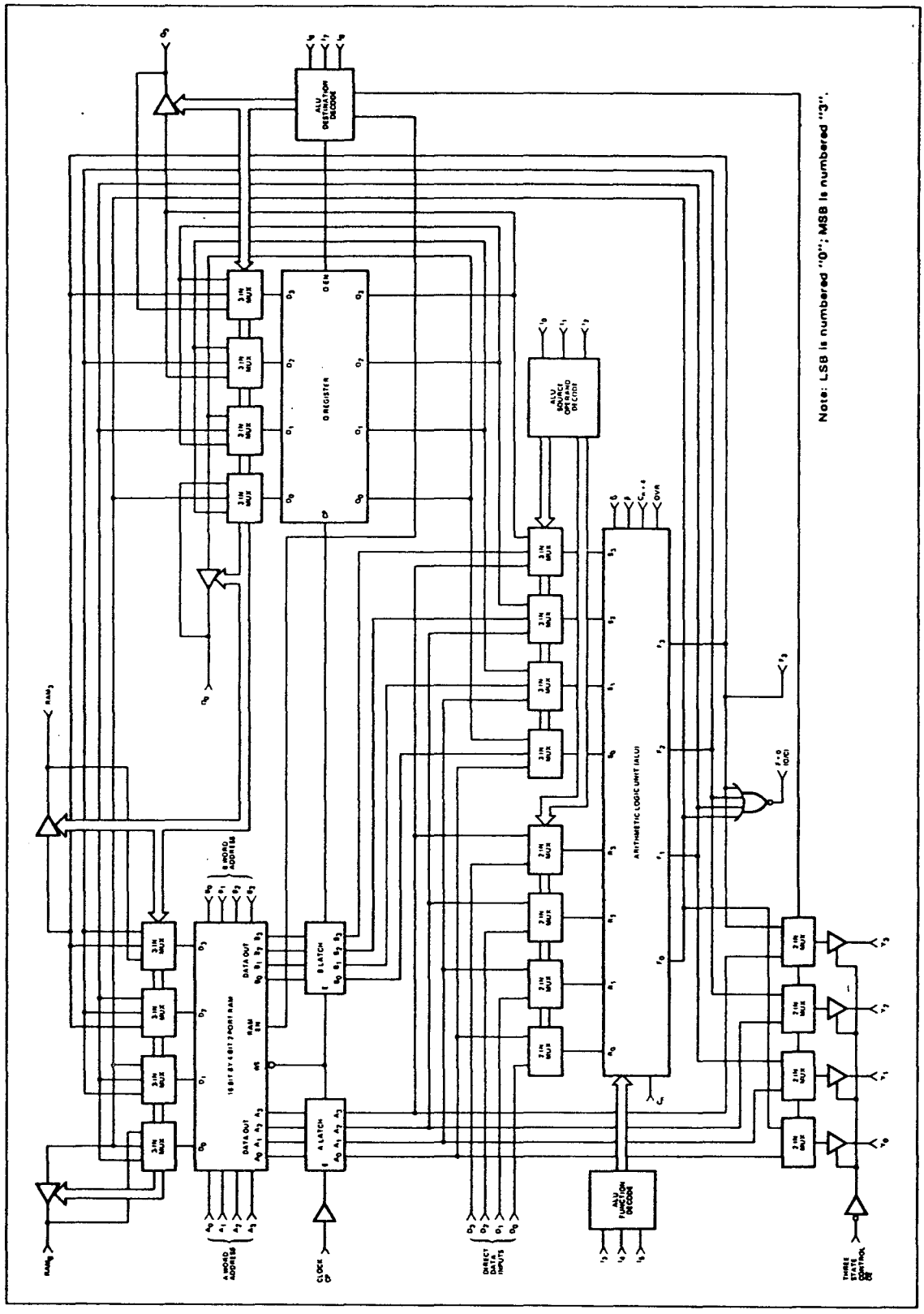


Figura. Diagrama de bloques detallado del Am2901C.

La ALU realiza tres operaciones aritméticas y cinco lógicas, sobre datos de cuatro bits, entre las entradas R y S. Estas entradas están multiplexadas posibilitando varias fuentes de datos para cada operación. Las entradas I0, I1, e I2, seleccionan los operandos fuentes de la ALU.

La función de la ALU se determina mediante I3, I4, e I5. Las salidas carry generado, $\overline{G}$, y carry propagado , $\overline{P}$, conectan cada dispositivo con el generador de acarreo paralelo (Am2902A).

La salida Cn+4 proporciona el flag de acarreo correspondiente para el registro de estado. Cn es el carry de entrada, activo a nivel alto. F3 es el bit más signifivativo de la ALU, determina el signo del resultado. F=0 detecta un resultado nulo.

EL resultado de cada operación puede ser ruteado a ocho posibles destinos según sean I6, I7 e I8.

	MICRO CODE				ALU SOURCE OPERANDS	
Mnemonic	I_2	I_1	I_0	Octal Code	R	S
AQ	L	L	L	0	A	Q
AB	L	L	H	1	A	B
ZQ	L	H	L	2	0	Q
ZB	L	H	H	3	0	B
ZA	H	L	L	4	0	A
DA	H	L	H	5	D	A
DQ	H	H	L	6	D	Q
DZ	H	H	H	7	D	0

Figure '. ALU Source Operand Control.

	MICRO CODE				ALU Function	SYMBOL
Mnemonic	I_5	I_4	I_3	Octal Code		
ADD	L	L	L	0	R Plus S	R + S
SUBR	L	L	H	1	S Minus R	S - R
SUBS	L	H	L	2	R Minus S	R - S
OR	L	H	H	3	R OR S	R $\vee$ S
AND	H	L	L	4	R AND S	R $\wedge$ S
NOTRS	H	L	H	5	$\overline{R}$ AND S	$\overline{R} \wedge S$
EXOR	H	H	L	6	R EX-OR S	R $\forall$ S
EXNOR	H	H	H	7	R EX-NOR S	$\overline{R \forall S}$

Figure '. ALU Function Control.

Mnemonic	MICRO CODE				RAM FUNCTION		Q-REG. FUNCTION		Y OUTPUT	RAM SHIFTER		Q SHIFTER	
	I_8	I_7	I_6	Octal Code	Shift	Load	Shift	Load		RAM_0	RAM_3	Q_0	Q_3
QREG	L	L	L	0	X	NONE	NONE	F → Q	F	X	X	X	X
NOP	L	L	H	1	X	NONE	X	NONE	F	X	X	X	X
RAMA	L	H	L	2	NONE	F → B	X	NONE	A	X	X	X	X
RAMF	L	H	H	3	NONE	F → B	X	NONE	F	X	X	X	X
RAMQD	H	L	L	4	DOWN	F/2 → B	DOWN	Q/2 → Q	F	F_0	IN_3	Q_0	IN_3
RAMD	H	L	H	5	DOWN	F/2 → B	X	NONE	F	F_0	IN_3	Q_0	X
RAMQU	H	H	L	6	UP	2F → B	UP	2Q → Q	F	IN_0	F_3	IN_0	Q_3
RAMU	H	H	H	7	UP	2F → B	X	NONE	F	IN_0	F_3	X	Q_3

X = Don't care. Electrically, the shift pin is a TTL input internally connected to a three-state output which is in the high-impedance state
B = Register Addressed by B inputs.
UP is toward MSB, DOWN is toward LSB.

Figure ALU Destination Control.

OCTAL I_{543}	I_{210} OCTAL / ALU Function	0	1	2	3	4	5	6	7
	ALU Source	A, Q	A, B	0, Q	0, B	0, A	D, A	D, Q	D, 0
0	C_n = L R Plus S	A+Q	A+B	Q	B	A	D+A	D+Q	D
	C_n = H	A+Q+1	A+B+1	Q+1	B+1	A+1	D+A+1	D+Q+1	D+1
1	C_n = L S Minus R	Q−A−1	B−A−1	Q−1	B−1	A−1	A−D−1	Q−D−1	−D−1
	C_n = H	Q−A	B−A	Q	B	A	A−D	Q−D	−D
2	C_n = L R Minus S	A−Q−1	A−B−1	−Q−1	−B−1	−A−1	D−A−1	D−Q−1	D−1
	C_n = H	A−Q	A−B	−Q	−B	−A	D−A	D−Q	D
3	R OR S	A∨Q	A∨B	Q	B	A	D∨A	D∨Q	D
4	R AND S	A∧Q	A∧B	0	0	0	D∧A	D∧Q	0
5	$\bar{R}$ AND S	$\bar{A}$∧Q	$\bar{A}$∧B	Q	B	A	$\bar{D}$∧A	$\bar{D}$∧Q	0
6	R EX-OR S	A⊻Q	A⊻B	Q	B	A	D⊻A	D⊻Q	D
7	R EX-NOR S	$\overline{A \veebar Q}$	$\overline{A \veebar B}$	$\bar{Q}$	$\bar{B}$	$\bar{A}$	$\overline{D \veebar A}$	$\overline{D \veebar Q}$	$\bar{D}$

+ = Plus, − = Minus; ∨ = OR; ∧ = AND; ⊻ = EX-OR

Figure '. Source Operand and ALU Function Matrix.

La unidad de ejecución (UE) se ha desarrollado con una estructura interna de 16 bits, para obtener mayor velocidad. De esta forma las operaciones de 8 y 16 bits se pueden realizar en una sola microinstrucción. Tal como se refleja en los diagramas de bloques, la UE se divide en dos subprocesadores de 8 bits (ALU izquierda y ALU derecha), formados cada uno de ellos por dos Am2901C. En operaciones de 8 bits las dos ALUs desarrollan la misma operación simultáneamente, y en operaciones de 16 bits ambos se unen en paralelo.

La disposición de los registros se ha diseñado siguiendo la estructura propuesta por Shavit. De esta forma es posible aprovechar los campos del código de operaciones, donde se definen los registros, para aplicarlo a los puertos de direcciones A y B de los Am2901Cs, con una mínima lógica de decodificación y una estructura entrelazada de las salidas y entradas de ambas.

Dirección	ALU izda	ALU dcha
0000	B	C
0001	C	B
0010	D	E
0011	E	D
0100	H	L
0101	L	H
0110	no usado	A
0111	A	no usado
1000	no usado	no usado
1001	SP0-7	SP8-15
1010	Z	W
1011	W	Z
1100	0	6
1101	6	0
1110	no usado	no usado
1111	PC0-7	PC8-15

Figura. Disposición de registros en CPA.

La estructura desarrollada para seleccionar los indicadores (flags), permite mantener la compatibilidad con el i8085. Se incluye además un generador de paridad. Dos comparadores de cuatro bits ayudan a migrar al firmware la ejecución del ajuste decimal del acumulador.

La ruta crítica que da la frecuencia del reloj se encuentra en la realización de operaciones aritméticas de 8 bits, debido a la necesidad de incluir un generador de paridad, tal como se apuntaba.

Dispositivo	Ruta	Poner Cy	Poner P
2918	CP to OUT	13,00	13,00
74F04	IN to OUT	5,00	5,00
74F253	IN to OUT	8,00	8,00
2901L	AB to CN+4	40,00	——
2901R	AB to GP	——	37,00
2902A	GP to Cn+z	——	7,00
74F157	IN to OUT	6,50	——
2901L	CN to Cn+4	20,00	——
2901R	CN to Y	——	22,00
74F280	IN to OUT	——	13,00
75F151	IN to OUT	6,50	6,50
2918	Set-up	3,00	3,00
	Total ns	102,00	114,50

Mnemonic	Flags	CY	AC	S	Z	P	Comentario
arit	0 0 0	C_{N+4}	$(C_{N+4})'$	F_3	F=0	P	Operaciones aritmeticas 8 bits
indc	0 0 1	—	$(C_{N+4})'$	F_3	F=0	P	Incrementar y decrementar
lop	0 1 0	0	0	F_3	F=0	P	Operaciones logicas
fdad	0 1 1	C_{N+4}	—	—	—	—	DAD
inva	1 0 0	—	—	—	—	—	No se modifican ninguno
rota	1 0 1	ramx	—	—	—	—	Rotaciones
cmcy	1 1 0	$\overline{CY}$	—	—	—	—	CMC
fbus	1 1 1	CY_B	AC_B	S_B	Z_B	P_B	POP PSW

Figura. Tabla de los indicadores que se actualizan en el registro de estado según el tipo de instrucción.

Mnemonic	Add-sel	W-dat	B_0^*	B_0	B_1	B_2	B_3	A_0^*	A_0	A_1	A_2	A_3
IRIR	0 0	0	$\overline{D}_3$	D_3	D_4	D_5	0	$\overline{D}_0$	D_0	D_1	D_2	0
IRPL	0 1	0	D_3	D_3	D_4	D_5	0	$\overline{A}_{PL0}$	A_{PL0}	A_{PL1}	A_{PL2}	A_{PL3}
PLIR	1 0	0	$\overline{B}_{PL0}$	B_{PL0}	B_{PL1}	B_{PL2}	B_{PL3}	$\overline{D}_0$	D_0	D_1	D_2	0
PLPL	1 1	0	$\overline{B}_{PL0}$	B_{PL0}	B_{PL1}	B_{PL2}	B_{PL3}	$\overline{A}_{PL0}$	A_{PL0}	A_{PL1}	A_{PL2}	A_{PL3}
IRIR	0 0	1	0	0	D_4	D_5	0	0	0	D_4	D_5	0
IRPL	0 1	1	0	0	D_4	D_5	0	A_{PL0}	A_{PL0}	A_{PL1}	A_{PL2}	A_{PL3}
PLIR	1 0	1	B_{PL0}	B_{PL0}	B_{PL1}	B_{PL2}	B_{PL3}	0	0	D_4	D_5	0
PLPL	1 1	1	B_{PL0}	B_{PL0}	B_{PL1}	B_{PL2}	B_{PL3}	A_{PL0}	A_{PL0}	A_{PL1}	A_{PL2}	A_{PL3}

Figura. Tabla de la lógica de decodificación que selecciona los registros.

3.2 MICROPROGRAMA

```
TITLE                    CPA;
MACHINE 64;

[

   ********************************************************
   Descripción de los campos de las microinstrucciones
   ********************************************************

    0-3    Dirección de la ruta B
    4-7    Dirección de la ruta A
   8-10   Selección del operando  fuente
     11    Carry inicial
  12-14    Función de la alu
     15    Selección de carry de entrada
  16-18    Destino de la operación
     19    Selección del tipo de rotación
  20-22    Selección de la fuente de los flags
     23    Selección de la ALU left/right
  24-25    Selecciona A B del PL or IR
     26    Selecciona un dato interno de 8/16 bits
     27    Dato de entrada a la alu interno o externo
  28-29    Status (S0, S1)
     30    Señal Read
     31    Input-output/memory (IO/M)
     32    Señal Write
     33    Triestado para rd, wr, io/m
     34    Enable del MBR
     35    Enable del MAR
     36    Señal ALE
  37-38    Control del bus interno
     39    Reconocimiento de hold
     40    Reconocimiento de interrupción (INTA)
     41    Enable/disable de interrupciones
     42    Enable del registro de puesta de máscara
     43    Enable del registro de instrucción
  44-47    Multiplexor de condición
  48-51    Secuencia de la próxima microinstrucción
  52-63    Dirección de la próxima microinstrucción
]

[.....registros
]

bc =0;
cb =1;
de =2;
ed =3;
hl =4;
lh =5;
xa =6;
ax =7;
r8 =8;
sp =9;
wz =10;
zw =11;
r12 =12;
r13 =13;
r14 =14;
pc =15;
```

```
[........ruta B de entrada ALU
]

b_alu  FIELD length=4, place=0, default=ax, format=1;


[........ruta A de entrada ALU
]

a_alu  FIELD length=4, place=4, default=ax, format=1;


[........selección del operando fuente
]


ab =1;      [ fuente de la alu A y B ]
zb =3;      [ fuente de la alu cero y B ]
za =4;      [ fuente de la alu cero y A ]
da =5;      [ fuente de la alu D y A ]
dz =7;      [ fuente de la alu D y cero ]

source  FIELD length=3, place=8, default=ab, format=1;


[........carry inicial
]

cy_in FIELD length=1, place=11,  default=0,  format=1;


[........operaciones de la ALU
]

add_rs  =0;  [ suma  R y S ]
sub_r   =1; [ resta S menos R  ]
sub_s   =2; [ resta R menos S  ]
or_rs   =3;  [ OR  de R y S ]
and_rs  =4;  [ AND de R y S ]
and_nor =5; [ AND de R negado con S  ]
xor_rs  =6;  [ OR EXCLUSIVO de R y S ]
noxor_rs=7; [ OR EXCLUSIVO negado de R y S ]

alu  FIELD length=3, place=12, default=or_rs, format=1;


[........selecciona el carry de entrada
]

cy_sel  FIELD length=1, place=15, default=0,  format=1;
```

```
[.........selecciona el destino de la operación
]

noper =1;      [ F --> ninguno;  F --> Y ]
ram_a =2;      [ F --> B; A --> Y ]
ram_f =3;      [ F --> B; F --> Y ]
ram_d =5;      [ F/2 --> B; F --> Y ]
ram_u =7;      [ 2F --> B; F--> Y ]

destino  FIELD  length=3; place=16; default=ram_f; format=1;

[.........selecciona el tipo de rotación
]

ramx   =0;   [ introduce ram0 o ramn  ]
carry =1;     [ introduce el carry ]

shifter     FIELD length=1; place=19; default=carry; format  =1;


[.........selección de los flags
]

arit =0; [ para operaciones aritméticas ]
indc =1; [ para operaciones de incremento y decremento ]
lop =2; [ para operaciones lógicas  ]
fdad =3; [ para la instrucción DAD ]
inva =4; [ los flags permanecen invariables ]
rota =5; [ para las instrucciones de rotación ]
cmcy =6; [ para la instrucción CMC ]
fbus =7; [ para la instrucción POP PSW ]

flags  FIELD length=3; place=20; default=inva;  format=1;


[.........alu left o alu right
]

left =0;
right =1;

leri  FIELD length=1; place=23; default=left; format=1;


[.........selecciona la fuente de las direcciones A y B
]

irir =0; [ B y A del IR ]
irpl =1; [ B del IR y A del PL ]
plir =2; [ B del PL y A de IR ]
plpl =3; [ B y A de PL ]

add_sel  FIELD   length=2; place=24; default=irir; format=1;
```

```
[.........ancho del dato
]

byte =0; [ dato de 8 bits ]
word =1; [ dato de 16 bits ]

w_dat  FIELD length=1, place=26, default=byte,  format=1;


[.........dato de entrada a la alu
]

in =0; [ dato interno ]
ex =1; [ dato externo ]

dat_in  FIELD length=1, place=27, default=in,  format=1;

[.........estado del microprocesador
]

hlt =0; [ estado HALT ]
mw =1;  [ escritura en memoria ]
mr =2;  [ lectura en memoria ]
op =3;  [ búsqueda del código de operación ]

status  FIELD length=2, place=28, default=hlt,  format=1;


rd  FIELD length=1, place=30, default=1, format=1;


[.........elección de memoria o entrada/salida
]

mem =0;
io =1;

io_m  FIELD length=1, place=31, default=mem, format=1;


wr  FIELD length=1, place=32, default=1, format=1;


ts      FIELD length=1, place=33, default=0,  format=1;


[.........enable del registro de memoria
]

mbre  FIELD length=1, place=34, default=1, format=1;
```

```
[........enable del MAR
]

mare  FIELD length=1, place=35, default=1, format=1;


ale  FIELD   length=1, place=36, default=0, format=1;


[........control de la salida al bus interno
]

bus_y =0;  [ salida de la alu ]
foe =1;    [ salida del registro de estado ]
rd_mask =2;[ salida del registro de máscara ]
udat =3;   [ salida del registro de datos de la micromemoria ]

b_ctrl  FIELD length=2, place=37, default=bus_y, format=1;


hlda  FIELD length=1, place=39, default=0, format=1;


inta  FIELD length=1, place=40, default=1, format=1;


[........habilita o deshabilita interrupciones
]

ds_int =0;
en_int =1;

edi  FIELD   length=1, place=41, default=ds_int, format=1;


[........entrada al registro de puesta de máscara
]

st_mask  FIELD length=1, place=42, default=1, format=1;


[........operaciones del opcode
]

ld_op =0; [ cargar el código de operación ]
no_op =1; [ no cargar el código operación ]

ir_en  FIELD length=1, place=43, default=no_op, format=1;
```

```
[........multiplexor de condición
]

neg  =0;
pos =1;
zero =2;
nozero  =3;
par =4;
impar =5;
cy =6;
nocy =7;
daal =8;
daah =10;
intr =12;
nopass =14;
pass =15;

cc_mux  FIELD   length=4, place=44, default=nopass, format=1;


[........control de la próxima dirección
]

jz =0;   [ salta a la posición de memoria cero ]
cjs =1; [ salto condicional a subrutina  ]
jmap =2; [ salta a la dirección dada por MAP ]
cjp =3;  [ salto condiconal a la dirección dada por PL ]
push =4; [ salvar μPC en la μPILA ]
cjv =6;  [ salto condicional a VECT ]
crtn =10;[ retorno  condicional desde subrutina ]
cont =14;

secuence FIELD length=4, place=48, default=cont, format=1;


v0= 6;
v1= 96;
v2= 1;
v3= 7;
v4= 8;
v5= 16;
v6= 24;
v7= 32;
v8= 40;
v9= 48;
v10= 56;
v11= 36;
v12= 60;
v13= 52;
v14= 44;

next_addr FIELD length=12, place=52, default=0,  format=1;
[
**************************************************************************************
]
```

```
[
 ************************************
 Definición de las microoperaciones
 ************************************
]


[........operaciones de lectura y escritura
]

read  INSTR io_m, rd=0, status=mr;


write  INSTR io_m, wr=0, status=mw;



[........microoperación de fetch
]

fetch1  INSTR  status, ale=1,  ts=0;


fetch2  INSTR status=op, rd=0, ir_en=ld_op;


[........seleccionar registros de trabajo
]

select INSTR b_alu,  a_alu,  add_sel;


[........tipo de dato
]

dat  INSTR w_dat,  dat_in;


[........poner a cero un registro
]

clear  INSTR destino=ram_f, source=za, alu=and_rs;
[
  0 and S
]


[........operaciones lógicas
]

l_and  INSTR   destino, source, flags=lop, alu=and_rs;
[
   R and S
]
```

```
l_or  INSTR   destino, source, flags=lop, alu=or_rs;
[
  R or S
]

l_xor  INSTR   destino, source, flags=lop, alu=xor_rs;
[
R xor S
]


nop  INSTR destino=noper, source=zb, alu=or_rs;
[
  no reliza operación alguna
]


cma  INSTR destino, source, flags=inva, alu=noxor_rs;
[
  complementar registros
]


cmc  INSTR flags=cmcy;
[
 complementar carry
]


[........operaciones aritméticas
]

add  INSTR   destino, source, flags=arit, alu=add_rs;
[
  R + S
]


adc  INSTR destino, source, flags=arit, cy_sel=1, alu=add_rs;
[
  R + S + cy
]

subr  INSTR destino, source, flags=arit, cy_in=1, alu=sub_r;
[
  S - R
]


subs  INSTR destino, source, flags=arit, cy_in=1, alu=sub_s;
[
  R - S
]
```

```
sbbr  INSTR destino, source, flags=arit, cy_sel=1, alu=sub_r;
[
  R - S - cy
]


sbbs  INSTR destino, source, flags=arit, cy_sel=1, alu=sub_s;
[
  S - R - cy
]


add_rp  INSTR destino, source, flags=fdad, alu=add_rs;



[.........operaciones de incremento y decremento
]

inr INSTR destino, source, flags=indc,  cy_in=1,  alu=add_rs;
[
  R <---- R + 1
]


inx  INSTR destino, source, cy_in=1, alu=add_rs;
[
  Rp <---- Rp + 1
]


dcr  INSTR destino, source, flags=indc, alu=sub_r;
[
  R <---- R - 1
]


dcrm  INSTR destino=ram_f, source=dz, flags=indc, alu=sub_s;
[
 M <---- M - 1
]


dcx  INSTR destino, source, alu=sub_r;
[
  Rp <---- Rp - 1
]


[.........operaciones de rotación
]

rlc  INSTR destino=ram_u,source=zb,alu=or_rs,shifter=ramx,flags=rota;
[
  rotación a la izquierda
```

```
rrc  INSTR destino=ram_d,source=zb,alu=or_rs,shifter=ramx,flags=rota;
[
  rotación a la derecha
]


ral  INSTR destino=ram_u,source=zb,alu=or_rs,shifter=carry,flags=rota;
[
  rotación a la izquierda incluyendo el carry
]


rar  INSTR destino=ram_d,source=zb,alu=or_rs,shifter=carry,flags=rota;
[
  rotación a la derecha incluyendo el carry
]


[.........transferencia entre registros
]

mov  INSTR destino, source, flags=inva, alu=or_rs;

[   Rj <----- Ri
]


stm  INSTR st_mask=0;
[
  habilita la entrada al registro de máscara
]


rdm  INSTR b_ctrl=rd_mask;
[
  habilita la salida del registro de máscara
]


[.........operaciones de control
]

jumap  INSTR   secuence=jmap;
[
  salta a la dirección dada por map
]


jcond  INSTR   next_addr,  cc_mux,  secuence=cjp;
[
  salto condicional a la dirección dada por pl
]
```

```
jump INSTR next_addr, cc_mux=pass, secuence=cjp;
[
  salto incondicional con la dirección del pl
]


call  INSTR next_addr, cc_mux=pass, secuence=cjs;
[
  llamada a subrutina
]


ccond  INSTR next_addr, cc_mux, secuence=cjs;
[
  llamada condicional a subrutina
]


ret  INSTR cc_mux=pass, secuence=crtn;
[
  retorno de subrutina
]


rcond  INSTR cc_mux, secuence=crtn;
[
   retorno condicional desde subrutina
]


vect  INSTR cc_mux=intr, secuence=cjv;
[
  saltar al vector de  interrupciones
]


halt  INSTR ts=1, status=hlt;
[
  estado de parada de microprocesador
]


[........control de las interrupciones
]

ei  INSTR edi=en_int;
[
  habilita las interrupciones
]


di  INSTR   edi=ds_int;
[
  deshabilita las interrupciones
]

end;
```

```
 1          TITLE                              CPA;
 2          MACHINE 64;
 3
 4          [
 5
 6                    ********************************************************
 7                    Descripción de los campos de las microinstrucciones
 8                    ********************************************************
 9
10                     0- 3   Dirección de la ruta B
11                     4- 7   Dirección de la ruta A
12                     8-10   Selección del operando fuente
13                       11   Carry inicial
14                    12-14   Función de la alu
15                       15   Selección de carry de entrada
16                    16-18   Destino de la operación
17                       19   Selección del tipo de rotación
18                    20-22   Selección de la fuente de los flags
19                       23   Selección de la ALU left/right
20                    24-25   Selecciona A B del PL or IR
21                       26   Selecciona un dato interno de 8/16 bits
22                       27   Dato de entrada a la alu interno o externo
23                    28-29   Status (S0, S1)
24                       30   Señal Read
25                       31   Input-output/memory (IO/M)
26                       32   Señal Write
27                       33   Triestado para rd, wr, io/m
28                       34   Enable del MBR
29                       35   Enable del MAR
30                       36   Señal ALE
31                    37-38   Control del bus interno
32                       39   Reconocimiento de hold
33                       40   Reconocimiento de interrupción (INTA)
34                       41   Enable/disable de interrupciones
35                       42   Enable del registro de puesta de máscara
36                       43   Enable del registro de instrucción
37                    44-47   Multiplexor de condición
38                  48-51 Secuencia de la próxima microinstrucción
39                    52-63   Dirección de la próxima microinstrucción
40          ]
41
42
43          [........registros
44          ]
45
46  0000    bc        =0;
47  0001    cb        =1;
48  0002    de        =2;
49  0003    ed        =3;
50  0004    hl        =4;
51  0005    lh        =5;
52  0006    xa        =6;
53  0007    ax        =7;
54  0008    r8        =8;
55  0009    sp        =9;
```

```
 56  000A  wz       =10;
 57  000B  zw       =11;
 58  000C  r12      =12;
 59  000D  r13      =13;
 60  000E  r14      =14;
 61  000F  pc       =15;
 62
 63        [........ruta B de entrada ALU
 64        ]
 65
 66        b_alu            FIELD   length=4, place=0, default=ax, format=1;
 67
 68
 69        [........ruta A de entrada ALU
 70        ]
 71
 72        a_alu            FIELD   length=4, place=4, default=ax, format=1;
 73
 74
 75        [........selección del operando fuente
 76        ]
 77
 78
 79  0001  ab       =1;      [ fuente de la alu A y B ]
 80  0003  zb       =3;      [ fuente de la alu cero y B ]
 81  0004  za       =4;      [ fuente de la alu cero y A ]
 82  0005  da       =5;      [ fuente de la alu D y A ]
 83  0007  dz       =7;      [ fuente de la alu D y cero ]
 84
 85        source           FIELD   length=3, place=8, default=ab, format=1;
 86
 87
 88        [........carry inicial
 89        ]
 90
 91        cy_in            FIELD   length=1, place=11, default=0, format=1;
 92
 93
 94        [........operaciones de la ALU
 95        ]
 96
 97  0000  add_rs   =0;      [ suma  R y S ]
 98  0001  sub_r    =1;      [ resta S menos R ]
 99  0002  sub_s    =2;      [ resta R menos S ]
100  0003  or_rs    =3;      [ OR  de R y S ]
101  0004  and_rs   =4;      [ AND de R y S ]
102  0005  and_nor  =5;      [ AND de R negado con S ]
103  0006  xor_rs   =6;      [ OR EXCLUSIVO de R y S ]
104  0007  noxor_rs =7;      [ OR EXCLUSIVO negado de R y S ]
105
106       alu              FIELD   length=3, place=12, default=or_rs, format=1
107
108
109       [........selecciona el carry de entrada
110       ]
```

```
111
112         cy_sel          FIELD   length=1, place=15, default=0, format=1;
113
114
115         [........selecciona el destino de la operación
116         ]
117
118   0001  noper    =1;     [ F --> ninguno,  F --> Y ]
119   0002  ram_a    =2;     [ F --> B, A --> Y ]
120   0003  ram_f    =3;     [ F --> B, F --> Y ]
121   0005  ram_d    =5;     [ F/2 --> B, F --> Y ]
122   0007  ram_u    =7;     [ 2F --> B, F--> Y ]
123
124         destino         FIELD    length=3, place=16, default=ram_f, format=
125
126
127         [........selecciona el tipo de rotación
128         ]
129
130   0000  ramx     =0;     [ introduce ram0 o ramn ]
131   0001  carry    =1;     [ introduce el carry ]
132
133         shifter         FIELD   length=1, place=19, default=carry, format =
134
135
136         [........selección de los flags
137         ]
138
139   0000  arit     =0;     [ para operaciones aritméticas ]
140   0001  indc     =1;     [ para operaciones de incremento y decremento ]
141   0002  lop      =2;     [ para operaciones lógicas ]
142   0003  fdad     =3;     [ para la instrucción DAD ]
143   0004  inva     =4;     [ los flags permanecen invariables ]
144   0005  rota     =5;     [ para las instrucciones de rotación ]
145   0006  cmcy     =6;     [ para la instrucción CMC ]
146   0007  fbus     =7;     [ para la instrucción POP PSW ]
147
148         flags           FIELD   length=3, place=20, default=inva, format=1;
149
150
151         [........alu left o alu right
152         ]
153
154   0000  left     =0;
155   0001  right    =1;
156
157         leri            FIELD   length=1, place=23, default=left, format=1;
158
159
160         [........selecciona la fuente de las direcciones A y B
161         ]
162
163   0000  irir     =0;     [ B y A del IR ]
164   0001  irpl     =1;     [ B del IR y A del PL ]
165   0002  plir     =2;     [ B del PL y A de IR ]
```

```
166  0003  plpl     =3;      [ B y A de PL ]
167
168        add_sel           FIELD   length=2, place=24, default=irir, format=1;
169
170
171        [........ancho del dato
172        ]
173
174  0000  byte     =0;      [ dato de 8 bits ]
175  0001  word     =1;      [ dato de 16 bits ]
176
177        w_dat             FIELD   length=1, place=26, default=byte, format=1;
178
179
180        [........dato de entrada a la alu
181        ]
182
183  0000  in       =0;      [ dato interno ]
184  0001  ex       =1;      [ dato externo ]
185
186        dat_in            FIELD   length=1, place=27, default=in, format=1;
187
188
189        [........estado del microprocesador
190        ]
191
192  0000  hlt      =0;      [ estado HALT ]
193  0001  mw       =1;      [ escritura en memoria ]
194  0002  mr       =2;      [ lectura en memoria ]
195  0003  op       =3;      [ búsqueda de código de operación ]
196
197        status            FIELD   length=2, place=28, default=hlt, format=1;
198
199
200        rd                FIELD   length=1, place=30, default=1, format=1;
201
202
203        [........elección de memoria o entrada/salida
204        ]
205
206  0000  mem      =0;
207  0001  io       =1;
208
209        io_m              FIELD   length=1, place=31, default=mem, format=1;
210
211
212        wr                FIELD   length=1, place=32, default=1, format=1;
213
214
215        ts                FIELD   length=1, place=33, default=0, format=1;
216
217
218        [........enable del registro de memoria
219        ]
220
```

```
221         mbre            FIELD   length=1, place=34, default=1, format=1;
222
223
224         [........enable del MAR
225         ]
226
227         mare            FIELD   length=1, place=35, default=1, format=1;
228
229
230         ale             FIELD   length=1, place=36, default=0, format=1;
231
232
233
234         [........control de la salida al bus interno
235         ]
236
237  0000   bus_y   =0;     [ salida de la alu ]
238  0001   foe     =1;     [ salida del registro de estado ]
239  0002   rd_mask =2;     [ salida del registro de máscara ]
240  0003   udat    =3;     [ salida del registro de datos de la micromemoria ]
241
242         b_ctrl          FIELD   length=2, place=37, default=bus_y, format=1
243
244
245         hlda            FIELD   length=1, place=39, default=0, format=1;
246
247
248         inta            FIELD   length=1, place=40, default=1, format=1;
249
250
251         [........habilita o deshabilita interrupciones
252         ]
253
254  0000   ds_int  =0;
255  0001   en_int  =1;
256
257         edi             FIELD   length=1, place=41, default=ds_int, format=
258
259
260         [........entrada al registro de puesta de máscara
261         ]
262
263         st_mask         FIELD   length=1, place=42, default=1, format=1;
264
265
266         [........operaciones del opcode
267         ]
268
269  0000   ld_op   =0;     [ cargar el código de operación ]
270  0001   no_op   =1;     [ no cargar el código operación ]
271
272         ir_en           FIELD   length=1, place=43, default=no_op, format=1
273
274
275         [........multiplexor de condición
```

```
276         ]
277
278  0000   neg      =0;
279  0001   pos      =1;
280  0002   zero     =2;
281  0003   nozero   =3;
282  0004   par      =4;
283  0005   impar    =5;
284  0006   cy       =6;
285  0007   nocy     =7;
286  0008   daal     =8;
287  000A   daah     =10;
288  000C   intr     =12;
289  000E   nopass   =14;
290  000F   pass     =15;
291
292         cc_mux           FIELD   length=4, place=44, default=nopass, format=
293
294
295         [........control de la próxima dirección
296         ]
297
298  0000   jz       =0;      [ salta a la posición de memoria cero ]
299  0001   cjs      =1;      [ salto condicional a subrutina ]
300  0002   jmap     =2;      [ salta a la dirección dada por MAP ]
301  0003   cjp      =3;      [ salto condicional a la direccion dada por PL ]
302  0004   push     =4;      [ salvar μPC en la μPILA ]
303  0006   cjv      =6;      [ salto condicional a VECT ]
304  000A   crtn     =10;     [ retorno  condicional desde subrutina ]
305  000E   cont     =14;
306
307         secuence         FIELD   length=4, place=48, default=cont, format=1;
308
309
310  0006   v0=      6;
311  0060   v1=      96;
312  0001   v2=      1;
313  0007   v3=      7;
314  0008   v4=      8;
315  0010   v5=      16;
316  0018   v6=      24;
317  0020   v7=      32;
318  0028   v8=      40;
319  0030   v9=      48;
320  0038   v10=     56;
321  0024   v11=     36;
322  003C   v12=     60;
323  0034   v13=     52;
324  002C   v14=     44;
325
326         next_addr        FIELD   length=12, place=52, default=0, format=1;
327
328
329
330
```

```
331         [
332         ********************************************************************************
333         ]
334
335
336
337
338         [
339                 ************************************
340                 Definición de las microoperaciones
341                 ************************************
342         ]
343
344
345         [........operaciones de lectura y escritura
346         ]
347
348         read            INSTR   io_m, rd=0, status=mr;
349
350
351         write           INSTR   io_m, wr=0, status=mw;
352
353
354
355         [........microoperación de fetch
356         ]
357
358         fetch1          INSTR   status, ale=1, ts=0;
359
360
361         fetch2          INSTR   status=op, rd=0, ir_en=ld_op;
362
363
364         [........seleccionar registros de trabajo
365         ]
366
367         select          INSTR   b_alu, a_alu, add_sel;
368
369
370         [........tipo de dato
371         ]
372
373         dat             INSTR   w_dat, dat_in;
374
375
376         [........poner a cero un registro
377         ]
378
379         clear           INSTR   destino=ram_f, source=za, alu=and_rs;
380         [
381                         0 and S
382         ]
383
384
385         [........operaciones lógicas
```

```
386         ]
387
388         l_and           INSTR   destino, source, flags=lop, alu=and_rs;
389         [
390                         R and S
391         ]
392
393
394         l_or            INSTR   destino, source, flags=lop, alu=or_rs;
395         [
396                         R or S
397         ]
398
399
400         l_xor           INSTR   destino, source, flags=lop, alu=xor_rs;
401         [
402                         R xor S
403         ]
404
405
406         nop             INSTR   destino=noper, source=zb, alu=or_rs;
407         [
408                         no reliza operación alguna
409         ]
410
411
412         cma             INSTR   destino, source, flags=inva, alu=noxor_rs;
413         [
414                         complementar registros
415         ]
416
417
418         cmc             INSTR   flags=cmcy;
419
420
421
422         [.....operaciones aritméticas
423         ]
424
425         add             INSTR   destino, source, flags=arit, alu=add_rs;
426         [
427                         R + S
428         ]
429
430
431         adc             INSTR   destino, source, flags=arit, cy_sel=1, alu=
432         [
433                         R + S + cy
434         ]
435
436
437         subr            INSTR   destino, source, flags=arit, cy_in=1, alu=s
438         [
439                         S - R
440         ]
```

```
441
442
443         subs            INSTR   destino, source, flags=arit, cy_in=1, alu=s
444         [
445                         R - S
446         ]
447
448
449         sbbr            INSTR   destino, source, flags=arit, cy_sel=1, alu=
450         [
451                         R - S - cy
452         ]
453
454
455         sbbs            INSTR   destino, source, flags=arit, cy_sel=1, alu=
456         [
457                         S - R - cy
458         ]
459
460
461         add_rp          INSTR   destino, source, flags=fdad, alu=add_rs;
462
463
464
465         [.........operaciones de incremento y decremento
466         ]
467
468         inr             INSTR   destino, source, flags=indc, cy_in=1, alu=a
469         [
470                         R <---- R + 1
471         ]
472
473
474         inx             INSTR   destino, source, cy_in=1, alu=add_rs;
475         [
476                         Rp <---- Rp + 1
477         ]
478
479
480         dcr             INSTR   destino, source, flags=indc, alu=sub_r;
481         [
482                         R <---- R - 1
483         ]
484
485
486         dcrm            INSTR   destino=ram_f, source=dz, flags=indc, alu=s
487         [               M <---- M - 1
488         ]
489
490
491         dcx             INSTR   destino, source, alu=sub_r;
492         [
493                         Rp <---- Rp - 1
494         ]
495
```

```
496
497
498         [.........operaciones de rotación
499         ]
500
501         rlc             INSTR destino=ram_u,source=zb,alu=or_rs,shifter=ram
502         rota;
503         [
504                         rotación a la izquierda
505         ]
506
507
508         rrc             INSTR destino=ram_d,source=zb,alu=or_rs,shifter=ram
509         rota;
510         [
511                         rotación a la derecha
512         ]
513
514         ral             INSTR destino=ram_u,source=zb,alu=or_rs,shifter=car
515         rota;
516         [
517                         rotación a la izquierda incluyendo el carry
518         ]
519
520
521         rar             INSTR destino=ram_d,source=zb,alu=or_rs,shifter=car
522         rota;
523         [
524                         rotación a la derecha incluyendo el carry
525         ]
526
527
528         [.........transferencia entre registros
529         ]
530
531         mov             INSTR   destino, source, flags=inva, alu=or_rs;
532
533         [                       Rj <----- Ri
534         ]
535
536
537         stm             INSTR   st_mask=0;
538         [
539                         habilita la entrada al registro de máscara
540         ]
541
542
543         rdm             INSTR   b_ctrl=rd_mask;
544         [
545                         habilita la salida del registro de máscara
546         ]
547
548
549         [.........operaciones de control
550         ]
```

```
551
552         jumap           INSTR   secuence=jmap;
553         [
554                         salta a la dirección dada por map
555         ]
556
557
558         jcond           INSTR   next_addr, cc_mux, secuence=cjp;
559         [
560                         salto condicional a la dirección dada por pl
561         ]
562
563
564         jump            INSTR   next_addr, cc_mux=pass, secuence=cjp;
565         [
566                         salto incondicional con la dirección del pl
567         ]
568
569
570         call            INSTR   next_addr, cc_mux=pass, secuence=cjs;
571         [
572                         llamada a subrutina
573         ]
574
575
576         ccond           INSTR   next_addr, cc_mux, secuence=cjs;
577         [
578                         llamada condicional a subrutina
579         ]
580
581
582         ret             INSTR   cc_mux=pass, secuence=crtn;
583         [
584                         retorno de subrutina
585         ]
586
587
588         rcond           INSTR   cc_mux, secuence=crtn;
589         [
590                         retorno condicional desde subrutina
591         ]
592
593
594         vect            INSTR   cc_mux=intr, secuence=cjv;
595         [
596                         saltar al vector de  interrupciones
597         ]
598
599
600         halt            INSTR   ts=1, status=hlt;
601         [
602                         estado de parada de microprocesador
603         ]
604
605
```

```
606         [.........control de las interrupciones
607         ]
608
609         ei                  INSTR   edi=en_int;
610         [
611                             habilita las interrupciones
612         ]
613
614
615         di                  INSTR   edi=ds_int;
616         [
617                             deshabilita las interrupciones
618         ]
619
620         end;
```

MICRO EPFL Version 1.2 CPA 23-MAR-1989
Tabla de simbolos

Este fichero contiene 104 definiciones de constantes

AB	0001	ADD_RS	0000	AND_NOR	0005	AND_RS	
AX	0007	BC	0000	BUS_Y	0000	BYTE	
CB	0001	CJP	0003	CJS	0001	CJV	
CONT	000E	CRTN	000A	CY	0006	DA	
DAAL	0008	DE	0002	DS_INT	0000	DZ	
EN_INT	0001	EX	0001	FBUS	0007	FDAD	
HL	0004	HLT	0000	IMPAR	0005	IN	
INTR	000C	INVA	0004	IO	0001	IRIR	
JMAP	0002	JZ	0000	LD_OP	0000	LEFT	
LOP	0002	MEM	0000	MR	0002	MW	
NOCY	0007	NOPASS	000E	NOPER	0001	NOXOR_RS	
NO_OP	0001	OP	0003	OR_RS	0003	PAR	
PC	000F	PLIR	0002	PLPL	0003	POS	
R12	000C	R13	000D	R14	000E	R8	
RAM_A	0002	RAM_D	0005	RAM_F	0003	RAM_U	
RIGHT	0001	ROTA	0005	SP	0009	SUB_R	
UDAT	0003	V0	0006	V1	0060	V10	
V12	003C	V13	0034	V14	002C	V2	
V4	0008	V5	0010	V6	0018	V7	
V9	0030	WORD	0001	WZ	000A	XA	
ZA	0004	ZB	0003	ZERO	0002	ZW	

30 definiciones de campos

ADD_SEL	ALE	ALU	A_ALU	B_ALU
CC_MUX	CY_IN	CY_SEL	DAT_IN	DESTINO
FLAGS	HLDA	INTA	IO_M	IR_EN
MARE	MBRE	NEXT_ADDR	RD	SECUENCE
SOURCE	STATUS	ST_MASK	TS	WR

y 43 definiciones de instrucciones

ADC	ADD	ADD_RP	CALL	CCOND
CMA	CMC	DAT	DCR	DCRM
DI	EI	FETCH1	FETCH2	HALT
INX	JCOND	JUMAP	JUMP	L_AND
L_XOR	MOV	NOP	RAL	RAR
RDM	READ	RET	RLC	RRC
SBBS	SELECT	STM	SUBR	SUBS
WRITE				

Tiempo de ensamblaje : 103000 ms

```
TITLE   CPA MICROPROGRAM;
NOBINARY;
LISTING;
[                        **************
                         MICROPROGRAMA
                         **************
]

init:  nop, next_addr=v0, b_ctrl=udat;
       select(r13,r13,plpl), dat(,), mov(,dz), b_ctrl=udat;
       nop, next_addr=v1, b_ctrl=udat;
       select(r12,r12,plpl), dat(,), mov(,dz), b_ctrl=udat;
       stm, nop, next_addr=v3, b_ctrl=udat;
       select(pc,pc,plpl), dat(word,), clear, mare=0;
fet_1: fetch1(op), nop;
fet_2: fetch2, nop, vect;
decod: select(pc,pc,plpl), dat(word,), inx(,za), mare=0, jumap;
inesc1:select(hl,hl,plpl),dat(word,),mov(ram_a,za),mare=0;
       fetch1(mw),select(pc,pc,plpl),dat(word,),mov(,za),mare=0;
       write(mem), nop, jump(fet_1);
movrr: select(,,), dat(,), mov(,za), fetch1(op), jump(fet_2);
movrm: select(hl,hl,plpl),dat(word,),mov(ram_a,za),mare=0;
       fetch1(mr),select(pc,pc,plpl),dat(word,),mov(,za),mare=0;
       read(mem), nop;
       select(,,irpl), dat(,ex), mov(,dz), fetch1(op), jump(fet_2);
movmr: select(,,plir), dat(,), mov(noper,za), mbre=0, jump(inesc1);
 sphl: select(sp,hl,plpl), dat(word,), mov(,za), fetch1(op), jump(fet_2);
mvird: fetch1(mr), select(pc,pc,plpl), dat(word,), inx(,za),mare=0;
       read(mem), nop;
       select(,,irpl), dat(,ex), mov(,dz), fetch1(op),jump(fet_2);
mvimd: fetch1(mr), select(pc,pc,plpl), dat(word,), inx(,za),mare=0;
       read(mem), nop;
       select(,,), dat(,ex), mov(noper,dz), mbre=0, jump(inesc1);
lxibc: fetch1(mr), select(pc,pc,plpl), dat(word,), inx(,za), mare=0;
       read(mem), nop;
       select(cb,,plpl), dat(,ex), mov(,dz);
       fetch1(mr), select(pc,pc,plpl), dat(word,), inx(,za), mare=0;
       read(mem), nop;
       select(bc,,plpl), dat(,ex), mov(,dz), fetch1(op), jump(fet_2);
lxide: fetch1(mr), select(pc,pc,plpl), dat(word,), inx(,za), mare=0;
       read(mem), nop;
       select(ed,,plpl), dat(,ex), mov(,dz);
       fetch1(mr), select(pc,pc,plpl), dat(word,), inx(,za), mare=0;
       read(mem), nop;
       select(de,,plpl), dat(,ex), mov(,dz), fetch1(op), jump(fet_2);
lxihl: fetch1(mr), select(pc,pc,plpl), dat(word,), inx(,za), mare=0;
       read(mem), nop;
       select(lh,,plpl), dat(,ex), mov(,dz);
       fetch1(mr), select(pc,pc,plpl), dat(word,), inx(,za), mare=0;
       read(mem), nop;
       select(hl,,plpl), dat(,ex), mov(,dz), fetch1(op), jump(fet_2);
lxisp: fetch1(mr), select(pc,pc,plpl), dat(word,), inx(,za), mare=0;
       read(mem), nop;
       select(r8,,plpl), dat(,ex), mov(,dz);
       fetch1(mr), select(pc,pc,plpl), dat(word,), inx(,za), mare=0;
       read(mem), nop;
       select(sp,,plpl), dat(,ex), mov(,dz), fetch1(op), jump(fet_2);
```

```
lda:    fetch1(mr); select(pc,pc,plpl); dat(word,); inx(,za),mare=0;
        read(mem); nop;
        select(zw,,plpl); dat(,ex); mov(,dz);
        fetch1(mr); select(pc,pc,plpl); dat(word,); inx(,za),mare=0;
        read(mem); nop;
        select(wz,,plpl); dat(,ex); mov(,dz);
        select(wz,wz,plpl); dat(word,); mov(ram_a,za),mare=0;
        fetch1(mr); select(pc,pc,plpl); dat(word,); mov(,za); mare=0;
        read(mem); nop;
        select(ax,,plpl); dat(,ex); mov(,dz),fetch1(op),jump(fet_2);
sta:    fetch1(mr); select(pc,pc,plpl); dat(word,); inx(,za),mare=0;
        read(mem); nop;
        select(zw,,plpl); dat(,ex); mov(,dz);
        fetch1(mr); select(pc,pc,plpl); dat(word,); inx(,za),mare=0;
        read(mem); nop;
        select(wz,,plpl); dat(,ex); mov(,dz);
        select(,ax,plpl); dat(,); mov(,za); mbre=0;
        select(wz,wz,plpl); dat(word,); mov(ram_a,za),mare=0;
        fetch1(mw); select(pc,pc,plpl); dat(word,); mov(,za); mare=0;
        write(mem); nop; jump(fet_1);
lhld:   fetch1(mr); select(pc,pc,plpl); dat(word,); inx(,za),mare=0;
        read(mem); nop;
        select(zw,,plpl); dat(,ex); mov(,dz);
        fetch1(mr); select(pc,pc,plpl); dat(word,); inx(,za),mare=0;
        read(mem); nop;
        select(wz,,plpl); dat(,ex); mov(,dz);
        select(wz,wz,plpl); dat(word,); inx(ram_a,za); mare=0;
        fetch1(mr); select(wz,wz,plpl); dat(word,); mov(,za),mare=0;
        read(mem); nop;
        select(lh,lh,plpl); dat(,ex); mov(,dz);
        fetch1(mr),select(pc,pc,plpl); dat(word,); mov(,za); mare=0;
        read(mem); nop;
        select(hl,,plpl); dat(,ex); mov(,dz); jump(fet_1);
shld:   fetch1(mr); select(pc,pc,plpl); dat(word,); inx(,za),mare=0;
        read(mem); nop;
        select(zw,,plpl); dat(,ex); mov(,dz);
        fetch1(mr); select(pc,pc,plpl); dat(word,); inx(,za),mare=0;
        read(mem); nop;
        select(wz,,plpl); dat(,ex); mov(,dz);
        select(wz,wz,plpl); dat(word,); inx(ram_a,za); mare=0;
        fetch1(mw); select(lh,lh,plpl); dat(,); mov(,za); mbre=0;
        write(mem); select(wz,wz,plpl); dat(word,); mov(,za),mare=0;
        fetch1(mw); select(hl,hl,plpl); dat(,); mov(,za),mbre=0;
        write(mem),select(pc,pc,plpl),dat(word,),mov(,za),mare=0,jump(fet_1);
ldaxr:  select(,,irir); dat(word,); mov(,za); mare=0;
        fetch1(mr); select(pc,pc,plpl); dat(word,); mov(,za); mare=0;
        read(mem); nop;
        select(ax,,plir); dat(,ex); mov(,dz); fetch1(op); jump(fet_2);
staxr:  select(,,irir); dat(word,); mov(,za); mare=0;
        fetch1(mw); select(ax,ax,plpl); dat(,); mov(,za); mbre=0;
        write(mem),select(pc,pc,plpl),dat(word,),mov(,za),mare=0,jump(fet_1);
```

```
xchg:  select(wz,hl,plpl); dat(word,); mov(,za);
       select(hl,de,plpl); dat(word,); mov(,za);
       select(lh,de,plpl); dat(word,); mov(ram_a,dz);
       select(de,wz,plpl); dat(word,); mov(,za);
       select(ed,de,plpl); dat(word,); mov(ram_a,dz); fetch1(op); jump(fet_2)
add_r: select(ax,,plir); dat(,); add(,ab); fetch1(op); jump(fet_2);
add_m: select(hl,hl,plpl);dat(word,);mov(ram_a,za);mare=0;
       fetch1(mr);select(pc,pc,plpl);dat(word,);mov(,za);mare=0;
       read(mem); nop;
       select(ax,ax,plpl); dat(,ex); add(,da); fetch1(op); jump(fet_2);
adi:   fetch1(mr); select(pc,pc,plpl); dat(word,); inx(,za);mare=0;
       read(mem); nop;
       select(ax,ax,plpl); dat(,ex); add(,da); fetch1(op); jump(fet_2);
adc_r: select(ax,,plir); dat(,); adc(,ab); fetch1(op); jump(fet_2);
adc_m: select(hl,hl,plpl);dat(word,);mov(ram_a,za);mare=0;
       fetch1(mr);select(pc,pc,plpl);dat(word,);mov(,za);mare=0;
       read(mem); nop;
       select(ax,ax,plpl); dat(,ex); adc(,da); fetch1(op); jump(fet_2);
aci:   fetch1(mr); select(pc,pc,plpl); dat(word,); inx(,za);mare=0;
       read(mem); nop;
       select(ax,ax,plpl); dat(,ex); adc(,da); fetch1(op); jump(fet_2);
sub_r: select(ax,,plir); dat(,); subr(,ab); fetch1(op); jump(fet_2);
sub_m: select(hl,hl,plpl);dat(word,);mov(ram_a,za);mare=0;
       fetch1(mr);select(pc,pc,plpl);dat(word,);mov(,za);mare=0;
       read(mem); nop;
       select(ax,ax,plpl); dat(,ex); subr(,da); fetch1(op); jump(fet_2);
sui:   fetch1(mr); select(pc,pc,plpl); dat(word,); inx(,za);mare=0;
       read(mem); nop;
       select(ax,ax,plpl); dat(,ex); subr(,da); fetch1(op); jump(fet_2);
sbb_r: select(ax,,plir); dat(,); sbbr(,ab); fetch1(op); jump(fet_2);
sbb_m: select(hl,hl,plpl);dat(word,);mov(ram_a,za);mare=0;
       fetch1(mr);select(pc,pc,plpl);dat(word,);mov(,za);mare=0;
       read(mem); nop;
       select(ax,ax,plpl); dat(,ex); sbbr(,da); fetch1(op); jump(fet_2);
sbi:   fetch1(mr); select(pc,pc,plpl); dat(word,); inx(,za);mare=0;
       read(mem); nop;
       select(ax,ax,plpl); dat(,ex); sbbr(,da); fetch1(op); jump(fet_2);
inr_r: select(,,irpl); dat(,); inr(,zb); fetch1(op); jump(fet_2);
inr_m: select(hl,hl,plpl);dat(word,);mov(ram_a,za);mare=0;
       fetch1(mr);select(pc,pc,plpl);dat(word,);mov(,za);mare=0;
       read(mem); nop;
       select(,,plpl); dat(,ex); inr(,dz); mbre=0; jump(inesc1);
dcr_r: select(,,irpl); dat(,); dcr(,zb); fetch1(op); jump(fet_2);
dcr_m: select(hl,hl,plpl);dat(word,);mov(ram_a,za);mare=0;
       fetch1(mr);select(pc,pc,plpl);dat(word,);mov(,za);mare=0;
       read(mem); nop;
       select(,,plpl); dat(,ex); dcrm; mbre=0; jump(inesc1);
inx_b: select(bc,bc,plpl); dat(word,); inx(,za);
       select(cb,bc,plpl); dat(word,); mov(ram_a,dz); fetch1(op);jump(fet_2);
inx_d: select(de,de,plpl); dat(word,); inx(,za);
       select(ed,de,plpl); dat(word,); mov(ram_a,dz); fetch1(op);jump(fet_2);
inx_h: select(hl,hl,plpl); dat(word,); inx(,za);
       select(lh,hl,plpl); dat(word,); mov(ram_a,dz); fetch1(op);jump(fet_2);
```

```
inxsp: select(sp,sp,plpl), dat(word,), inx(,za);
       select(r8,sp,plpl), dat(word,), mov(ram_a,dz), fetch1(op),jump(fet_2);
dcx_b: select(bc,bc,plpl), dat(word,), dcx(,za);
       select(cb,bc,plpl), dat(word,), mov(ram_a,dz), fetch1(op),jump(fet_2);
dcx_d: select(de,de,plpl), dat(word,), dcx(,za);
       select(ed,de,plpl), dat(word,), mov(ram_a,dz), fetch1(op),jump(fet_2);
dcx_h: select(hl,hl,plpl), dat(word,), dcx(,za);
       select(lh,hl,plpl), dat(word,), mov(ram_a,dz), fetch1(op),jump(fet_2);
dcxsp: select(sp,sp,plpl), dat(word,), dcx(,za);
       select(r8,sp,plpl), dat(word,), mov(ram_a,dz), fetch1(op),jump(fet_2);
dad_b: select(hl,bc,plpl), dat(word,), add_rp(,ab);
       select(lh,hl,plpl), dat(word,), mov(ram_a,dz), fetch1(op),jump(fet_2);
dad_d: select(hl,de,plpl), dat(word,), add_rp(,ab);
       select(lh,hl,plpl), dat(word,), mov(ram_a,dz), fetch1(op),jump(fet_2);
dad_h: select(hl,hl,plpl), dat(word,), add_rp(,ab);
       select(lh,hl,plpl), dat(word,), mov(ram_a,dz), fetch1(op),jump(fet_2);
dadsp: elect(hl,sp,plpl), dat(word,), add_rp(,ab);
       select(lh,hl,plpl), dat(word,), mov(ram_a,dz), fetch1(op), jump(fet_2);
daa_a: select(ax,ax,plpl), dat(,), mov(ram_a,za);
       select(ax,ax,plpl), dat(,), mov(ram_a,za), jcond(e1,daal);
e3:    nop, jcond(e2,daah);
       fetch1(op), nop, jump(fet_2);
e1:    select(ax,r13,plpl),dat(,),add(,ab), jump(e3);
e2:    select(ax,r12,plpl),dat(,),add(,ab), fetch1(op), jump(fet_2);
ana_r: select(ax,,plir), dat(,), l_and(,ab), fetch1(op), jump(fet_2);
ana_m: select(hl,hl,plpl),dat(word,),mov(ram_a,za),mare=0;
       fetch1(mr),select(pc,pc,plpl),dat(word,),mov(,za),mare=0;
       read(mem), nop;
       select(ax,ax,plpl), dat(,ex), l_and(,da), fetch1(op), jump(fet_2);
ani:   fetch1(mr), select(pc,pc,plpl), dat(word,), inx(,za),mare=0;
       read(mem), nop;
       select(ax,ax,plpl), dat(,ex), l_and(,da), fetch1(op), jump(fet_2);
xra_r: select(ax,,plir), dat(,), l_xor(,ab), fetch1(op), jump(fet_2);
xra_m: select(hl,hl,plpl),dat(word,),mov(ram_a,za),mare=0;
       fetch1(mr),select(pc,pc,plpl),dat(word,),mov(,za),mare=0;
       read(mem), nop;
       select(ax,ax,plpl), dat(,ex), l_xor(,da), fetch1(op), jump(fet_2);
xri:   fetch1(mr), select(pc,pc,plpl), dat(word,), inx(,za),mare=0;
       read(mem), nop;
       select(ax,ax,plpl), dat(,ex), l_xor(,da), fetch1(op), jump(fet_2);
ora_r: select(ax,,plir), dat(,), l_or(,ab), fetch1(op), jump(fet_2);
ora_m: select(hl,hl,plpl),dat(word,),mov(ram_a,za),mare=0;
       fetch1(mr),select(pc,pc,plpl),dat(word,),mov(,za),mare=0;
       read(mem), nop;
       select(ax,ax,plpl), dat(,ex), l_or(,da), fetch1(op), jump(fet_2);
ori:   fetch1(mr), select(pc,pc,plpl), dat(word,), inx(,za),mare=0;
       read(mem), nop;
       select(ax,ax,plpl), dat(,ex), l_or(,da), fetch1(op), jump(fet_2);
cmp_r: select(ax,,plir), dat(,), subr(noper,ab), fetch1(op), jump(fet_2);
cmp_m: select(hl,hl,plpl),dat(word,),mov(ram_a,za),mare=0;
       fetch1(mr),select(pc,pc,plpl),dat(word,),mov(,za),mare=0;
       read(mem), nop;
       select(,ax,plpl), dat(,ex), subr(noper,da), fetch1(op), jump(fet_2);
```

```
cpi:   fetch1(mr), select(pc,pc,plpl), dat(word,), inx(,za),mare=0;
       read(mem), nop;
       select(ax,ax,plpl), dat(,ex), subr(noper,da),fetch1(op), jump(fet_2);
rlc_a: select(ax,,plpl), dat(,), rlc, fetch1(op), jump(fet_2);
rrc_a: select(ax,,plpl), dat(,), rrc, fetch1(op), jump(fet_2);
ral_a: select(ax,,plpl), dat(,), ral, fetch1(op), jump(fet_2);
rar_a: select(ax,,plpl), dat(,), rar, fetch1(op), jump(fet_2);
cma_a: select(ax,,plpl), dat(,), cma(,zb), fetch1(op), jump(fet_2);
cmc_f: cmc, nop, fetch1(op), jump(fet_2);
stc_f: nop, next_addr=v2, b_ctrl=udat;
       select(wz,wz,plpl), dat(,), mov(,dz), b_ctrl=udat;
       select(wz,wz,plpl), dat(,), rar, fetch1(op), jump(fet_2);
jmp_a: fetch1(mr), select(pc,pc,plpl), dat(word,), inx(,za),mare=0;
       read(mem), nop;
       select(zw,,plpl), dat(,ex), mov(,dz);
       fetch1(mr), select(pc,pc,plpl), dat(word,), inx(,za),mare=0;
       read(mem), nop;
       select(wz,,plpl), dat(,ex), mov(,dz);
       select(pc,wz,plpl),dat(word,),mov(,za),mare=0,jump(fet_1);
jnz:  nop,  jcond(jmp_a,nozero);
       fetch1(mr), select(pc,pc,plpl), dat(word,), inx(,za),mare=0;
       read(mem), nop;
       select(zw,,plpl), dat(,ex), mov(,dz);
       fetch1(mr), select(pc,pc,plpl), dat(word,), inx(,za),mare=0;
       read(mem), nop;
       select(wz,,plpl), dat(,ex), mov(,dz);
       fetch1(op), nop, jump(fet_2);
jz:    jcond(jmp_a,zero);
       fetch1(mr), select(pc,pc,plpl), dat(word,), inx(,za),mare=0;
       read(mem), nop;
       select(zw,,plpl), dat(,ex), mov(,dz);
       fetch1(mr), select(pc,pc,plpl), dat(word,), inx(,za),mare=0;
       read(mem), nop;
       select(wz,,plpl), dat(,ex), mov(,dz);
       fetch1(op), nop, jump(fet_2);
jnc:   nop, jcond(jmp_a,nocy);
       fetch1(mr), select(pc,pc,plpl), dat(word,), inx(,za),mare=0;
       read(mem), nop;
       select(zw,,plpl), dat(,ex), mov(,dz);
       fetch1(mr), select(pc,pc,plpl), dat(word,), inx(,za),mare=0;
       read(mem), nop;
       select(wz,,plpl), dat(,ex), mov(,dz);
       fetch1(op), nop, jump(fet_2);
jc:   nop,  jcond(jmp_a,cy);
       fetch1(mr), select(pc,pc,plpl), dat(word,), inx(,za),mare=0;
       read(mem), nop;
       select(zw,,plpl), dat(,ex), mov(,dz);
       fetch1(mr), select(pc,pc,plpl), dat(word,), inx(,za),mare=0;
       read(mem), nop;
       select(wz,,plpl), dat(,ex), mov(,dz);
       fetch1(op), nop, jump(fet_2);
```

```
jpo:    nop, jcond(jmp_a,impar);
        fetch1(mr), select(pc,pc,plpl), dat(word,), inx(,za),mare=0;
        read(mem), nop;
        select(zw,,plpl), dat(,ex), mov(,dz);
        fetch1(mr), select(pc,pc,plpl), dat(word,), inx(,za),mare=0;
        read(mem), nop;
        select(wz,,plpl), dat(,ex), mov(,dz);
        fetch1(op), nop, jump(fet_2);
jpe:    nop, jcond(jmp_a,par);
        fetch1(mr), select(pc,pc,plpl), dat(word,), inx(,za),mare=0;
        read(mem), nop;
        select(zw,,plpl), dat(,ex), mov(,dz);
        fetch1(mr), select(pc,pc,plpl), dat(word,), inx(,za),mare=0;
        read(mem), nop;
        select(wz,,plpl), dat(,ex), mov(,dz);
        fetch1(op), nop, jump(fet_2);
jp:     nop, jcond(jmp_a,pos);
        fetch1(mr), select(pc,pc,plpl), dat(word,), inx(,za),mare=0;
        read(mem), nop;
        select(zw,,plpl), dat(,ex), mov(,dz);
        fetch1(mr), select(pc,pc,plpl), dat(word,), inx(,za),mare=0;
        read(mem), nop;
        select(wz,,plpl), dat(,ex), mov(,dz);
        fetch1(op), nop, jump(fet_2);
jm:     nop, jcond(jmp_a,neg);
        fetch1(mr), select(pc,pc,plpl), dat(word,), inx(,za),mare=0;
        read(mem), nop;
        select(zw,,plpl), dat(,ex), mov(,dz);
        fetch1(mr), select(pc,pc,plpl), dat(word,), inx(,za),mare=0;
        read(mem), nop;
        select(wz,,plpl), dat(,ex), mov(,dz);
        fetch1(op), nop, jump(fet_2);
calla:  select(sp,sp,plpl), dat(word,), dcx(,za);
        fetch1(mr), select(pc,pc,plpl), dat(word,), inx(,za),mare=0;
        read(mem), nop;
        select(zw,,plpl), dat(,ex), mov(,dz);
        fetch1(mr), select(pc,pc,plpl), dat(word,), inx(,za),mare=0;
        read(mem), nop;
        select(wz,,plpl), dat(,ex), mov(,dz);
        select(sp,sp,plpl), dat(word,), dcx(ram_a,za), mare=0;
        fetch1(mw), select(pc,pc,plpl), dat(,), mbre=0, mov(,za);
        write(mem),select(sp,sp,plpl),dat(word,),mov(,za),mare=0;
        fetch1(mw),select(r14,r14,plpl),dat(,),mov(,za),leri=right,mbre=0;
        write(mem),select(pc,wz,plpl),dat(word,),mov(,za),mare=0,jump(fet_1);
cnz:    nop, jcond(calla,nozero);
        fetch1(mr), select(pc,pc,plpl), dat(word,), inx(,za),mare=0;
        read(mem), nop;
        select(zw,,plpl), dat(,ex), mov(,dz);
        fetch1(mr), select(pc,pc,plpl), dat(word,), inx(,za),mare=0;
        read(mem), nop;
        select(wz,,plpl), dat(,ex), mov(,dz);
        fetch1(op), nop, jump(fet_2);
```

```
cz:     nop, jcond(calla,zero);
        fetch1(mr), select(pc,pc,plpl), dat(word,), inx(,za),mare=0;
        read(mem), nop;
        select(zw,,plpl), dat(,ex), mov(,dz);
        fetch1(mr), select(pc,pc,plpl), dat(word,), inx(,za),mare=0;
        read(mem), nop;
        select(wz,,plpl), dat(,ex), mov(,dz);
        fetch1(op), nop, jump(fet_2);
cnc:    nop, jcond(calla,nocy);
        fetch1(mr), select(pc,pc,plpl), dat(word,), inx(,za),mare=0;
        read(mem), nop;
        select(zw,,plpl), dat(,ex), mov(,dz);
        fetch1(mr), select(pc,pc,plpl), dat(word,), inx(,za),mare=0;
        read(mem), nop;
        select(wz,,plpl), dat(,ex), mov(,dz);
        fetch1(op), nop, jump(fet_2);
cc:     nop, jcond(calla,cy);
        fetch1(mr), select(pc,pc,plpl), dat(word,), inx(,za),mare=0;
        read(mem), nop;
        select(zw,,plpl), dat(,ex), mov(,dz);
        fetch1(mr), select(pc,pc,plpl), dat(word,), inx(,za),mare=0;
        read(mem), nop;
        select(wz,,plpl), dat(,ex), mov(,dz);
        fetch1(op), nop, jump(fet_2);
cpo:    nop, jcond(calla,impar);
        fetch1(mr), select(pc,pc,plpl), dat(word,), inx(,za),mare=0;
        read(mem), nop;
        select(zw,,plpl), dat(,ex), mov(,dz);
        fetch1(mr), select(pc,pc,plpl), dat(word,), inx(,za),mare=0;
        read(mem), nop;
        select(wz,,plpl), dat(,ex), mov(,dz);
        fetch1(op), nop, jump(fet_2);
cpe:    nop, jcond(calla,par);
        fetch1(mr), select(pc,pc,plpl), dat(word,), inx(,za),mare=0;
        read(mem), nop;
        select(zw,,plpl), dat(,ex), mov(,dz);
        fetch1(mr), select(pc,pc,plpl), dat(word,), inx(,za),mare=0;
        read(mem), nop;
        select(wz,,plpl), dat(,ex), mov(,dz);
        fetch1(op), nop, jump(fet_2);
cp:     nop, jcond(calla,pos);
        fetch1(mr), select(pc,pc,plpl), dat(word,), inx(,za),mare=0;
        read(mem), nop;
        select(zw,,plpl), dat(,ex), mov(,dz);
        fetch1(mr), select(pc,pc,plpl), dat(word,), inx(,za),mare=0;
        read(mem), nop;
        select(wz,,plpl), dat(,ex), mov(,dz);
        fetch1(op), nop, jump(fet_2);
cm:     nop, jcond(calla,neg);
        fetch1(mr), select(pc,pc,plpl), dat(word,), inx(,za),mare=0;
        read(mem), nop;
        select(zw,,plpl), dat(,ex), mov(,dz);
        fetch1(mr), select(pc,pc,plpl), dat(word,), inx(,za),mare=0;
        read(mem), nop;
        select(wz,,plpl), dat(,ex), mov(,dz);
        fetch1(op), nop, jump(fet_2);
```

```
retn:  select(sp,sp,plpl), dat(word,), inx(ram_a,za), mare=0;
       fetch1(mr),select(sp,sp,plpl),dat(word,),inx(ram_a,za),mare=0;
       nop, read(mem) ;
       fetch1(mr),select(r14,,plpl), dat(,ex), mov(,dz);
       nop, read(mem);
       select(pc,,plpl), dat(,ex), mov(,dz);
       select(pc,pc,plpl),dat(word,),mov(,za),mare=0,jump(fet_1);
rnz:   nop, jcond(retn,nozero);
       fetch1(op), nop, jump(fet_2);
rz:    nop, jcond(retn,zero);
       fetch1(op), nop, jump(fet_2);
rnc:   nop, jcond(retn,nocy);
       fetch1(op), nop, jump(fet_2);
rc:    nop, jcond(retn,cy);
       fetch1(op), nop, jump(fet_2);
rpo:   nop, jcond(retn,impar);
       fetch1(op), nop, jump(fet_2);
rpe:   nop, jcond(retn,par);
       fetch1(op), nop, jump(fet_2);
rp:    nop, jcond(retn,pos);
       fetch1(op), nop, jump(fet_2);
rm:    nop, jcond(retn,neg);
       fetch1(op), nop, jump(fet_2);
pchl:  select(pc,hl,plpl), dat(word,), mov(,za), mare=0, jump(fet_1);
pushb: select(sp,sp,plpl), dat(word,), dcx(,za), mare=0;
       select(sp,sp,plpl), dat(word,), dcx(,za), fetch1(mw), mare=0;
       select(bc,bc,plpl), dat(,), mov(ram_a,za), mbre=0;
       nop, write(mem);
       select(cb,cb,plpl), dat(,), mov(ram_a,za), mbre=0, fetch1(mw);
       select(pc,pc,plpl),dat(word,),mov(,za),mare=0,write(mem),jump(fet_1);
pushd: select(sp,sp,plpl), dat(word,), dcx(,za), mare=0;
       select(sp,sp,plpl), dat(word,), dcx(,za), fetch1(mw), mare=0;
       select(de,de,plpl), dat(,), mov(ram_a,za), mbre=0;
       nop, write(mem);
       select(ed,ed,plpl), dat(,), mov(ram_a,za), mbre=0, fetch1(mw);
       select(pc,pc,plpl),dat(word,),mov(,za),mare=0,write(mem), jump(fet_1);
pushh: select(sp,sp,plpl), dat(word,), dcx(,za), mare=0;
       select(sp,sp,plpl), dat(word,), dcx(,za), fetch1(mw), mare=0;
       select(hl,hl,plpl), dat(,), mov(ram_a,za), mbre=0;
       nop, write(mem);
       select(lh,lh,plpl), dat(,), mov(ram_a,za), mbre=0, fetch1(mw);
       select(pc,pc,plpl),dat(word,),mov(,za),mare=0,write(mem), jump(fet_1);
pushp: select(sp,sp,plpl), dat(word,), dcx(,za), mare=0;
       select(sp,sp,plpl), dat(word,), dcx(,za), fetch1(mw), mare=0;
       select(ax,ax,plpl), dat(,), mov(ram_a,za), mbre=0;
       nop, write(mem);
       nop, b_ctrl=foe, mbre=0, fetch1(mw);
       select(pc,pc,plpl),dat(word,),mov(,za),mare=0,write(mem), jump(fet_1);
popb:  select(sp,sp,plpl), dat(word,), inx(ram_a,za),  mare=0;
       select(sp,sp,plpl), dat(word,), mov(,za), mare=0, fetch1(mr);
       nop, read(mem);
       select(cb,cb,plpl), dat(,ex), mov(,dz);
       select(sp,sp,plpl), dat(word,), inx(,za), fetch1(mr);
       select(pc,pc,plpl),dat(word,),mov(,za),mare=0,read(mem);
       select(bc,bc,plpl), dat(,ex), mov(,dz), fetch1(op), jump(fet_2);
```

```
popd: select(sp,sp,plpl), dat(word,), inx(ram_a,za),  mare=0;
      select(sp,sp,plpl), dat(word,), mov(,za), mare=0, fetch1(mr);
      nop, read(mem);
      select(ed,ed,plpl), dat(,ex), mov(,dz);
      select(sp,sp,plpl), dat(word,), inx(,za), fetch1(mr);
      select(pc,pc,plpl),dat(word,),mov(,za),mare=0,read(mem);
      select(de,de,plpl), dat(,ex), mov(,dz), fetch1(op), jump(fet_2);
poph: select(sp,sp,plpl), dat(word,), inx(ram_a,za),  mare=0;
      select(sp,sp,plpl), dat(word,), mov(,za), mare=0, fetch1(mr);
      nop, read(mem);
      select(lh,lh,plpl), dat(,ex), mov(,dz);
      select(sp,sp,plpl), dat(word,), inx(,za), fetch1(mr);
      read(mem), select(pc,pc,plpl), dat(word,), mov(,za), mare=0;
      select(hl,hl,plpl), dat(,ex), mov(,dz), fetch1(op), jump(fet_2);
popp:  select(sp,sp,plpl), dat(word,), inx(ram_a,za), mare=0;
      select(sp,sp,plpl), dat(word,), mov(,za), mare=0, fetch1(mr);
      nop, read(mem);
      select(,,plpl), dat(,ex),alu=or_rs,destino=noper,source=dz,flags=fbus;
      select(sp,sp,plpl), dat(word,), inx(ram_a,za), fetch1(mr);
      read(mem), select(pc,pc,plpl), mov(,za), mare=0;
      select(ax,ax,plpl), dat(,ex), mov(,dz), fetch1(op), jump(fet_2);
xthl: select(sp,sp,plpl), dat(word,), inx(ram_a,za),  mare=0;
      fetch1(mr), select(sp,sp,plpl), dat(word,), mov(,za), mare=0;
      nop, read(mem);
      fetch1(mr), select(zw,zw,plpl), dat(,ex), mov(,dz);
      read(mem), select(hl,hl,plpl), dat(,), mov(,za), mbre=0;
      fetch1(mw), select(wz,wz,plpl), dat(,ex), mov(,dz);
      write(mem), select(sp,sp,plpl), dat(word,), dcx(,za),mare=0;
      fetch1(mw), select(lh,lh,plpl), dat(,), mov(,za), mbre=0;
      write(mem), select(hl,wz,plpl), dat(word,), mov(,za);
      select(lh,zw,plpl), dat(word,), mov(,za);
      select(pc,pc,plpl), dat(word,), mov(,za), mare=0, jump(fet_1);
inp:   fetch1(mr), select(pc,pc,plpl), dat(word,), inx(,za), mare=0;
      read(mem), select(wz,wz,plpl), dat(word,ex), mov(,dz), mare=0;
      select(zw,zw,plpl), dat(word,ex), mov(,dz), mare=0;
      fetch1(mr), select(pc,pc,plpl), dat(word,), mov(,za), mare=0;
      read(io), select(ax,ax,plpl), dat(,ex), mov(,dz), jump(fet_1);
outp:  fetch1(mr), select(pc,pc,plpl), dat(word,), inx(,za), mare=0;
      read(mem), select(wz,wz,plpl), dat(word,ex), mov(,dz), mare=0;
      select(zw,zw,plpl), dat(word,ex), mov(,dz), mare=0;
      fetch1(mw), select(ax,ax,plpl), dat(,), mov(,za), mbre=0;
      write(io), select(pc,pc,plpl), dat(word,), mov(,za),mare=0,jump(fet_1);
eint:  ei, fetch1(op), nop, jump(fet_2);
dint:  di, fetch1(op), nop, jump(fet_2);
hlt:   nop, fetch1(hlt);
e5:    nop, halt, vect;
 nop,  jump(e5);
nopr:  nop, fetch1(op), jump(fet_2);
rim:   nop,b_ctrl=rd_mask;
      select(ax,ax,plpl),dat(,),mov(,dz),fetch1(op),jump(fet_2);
sim: select(ax,ax,plpl),dat(,),mov(ram_a,za),st_mask=0,fetch1(op),jump(fet_2);
rst_0: select(wz,wz,plpl), dat(,), clear;
      select(zw,zw,plpl), dat(,), clear;
      select(sp,sp,plpl), dat(word,), dcx(,za), mare=0;
```

```
prox:  select(pc,pc,plpl); dat(,); mov(,za); mbre=0;
       fetch1(mw); select(sp,sp,plpl); dat(word,); dcx(,za); mare=0;
       nop; write(mem);
       fetch1(mw);select(r14,r14,plpl);dat(,);mov(,za);leri=right;mbre=0;
       write(mem);select(pc,wz,plpl);dat(word,);mov(,za);mare=0;jump(fet_1);
rst_1: select(wz,wz,plpl); dat(,); clear;
       nop; next_addr=v4; b_ctrl=udat;
       select(zw,zw,plpl); dat(,); mov(,dz); b_ctrl=udat;
       select(sp,sp,plpl); dat(word,); dcx(,za); mare=0; jump(prox);
rst_2: select(wz,wz,plpl); dat(,); clear;
       nop; next_addr=v5; b_ctrl=udat;
       select(zw,zw,plpl); dat(,); mov(,dz); b_ctrl=udat;
       select(sp,sp,plpl); dat(word,); dcx(,za); mare=0; jump(prox);
rst_3: select(wz,wz,plpl); dat(,); clear;
       nop; next_addr=v6; b_ctrl=udat;
       select(zw,zw,plpl); dat(,);mov(,dz),b_ctrl=udat;
       select(sp,sp,plpl); dat(word,); dcx(,za); mare=0; jump(prox);
rst_4: select(wz,wz,plpl); dat(,); clear;
       nop; next_addr=v7; b_ctrl=udat;
       select(zw,zw,plpl); dat(,); mov(,dz); b_ctrl=udat;
       select(sp,sp,plpl); dat(word,); dcx(,za); mare=0; jump(prox);
rst_5: select(wz,wz,plpl); dat(,); clear;
       nop; next_addr=v8; b_ctrl=udat;
       select(zw,zw,plpl); dat(,); mov(,dz); b_ctrl=udat;
       select(sp,sp,plpl); dat(word,); dcx(,za); mare=0; jump(prox);
rst_6: select(wz,wz,plpl); dat(,); clear;
       nop; next_addr=v9; b_ctrl=udat;
       select(zw,zw,plpl); dat(,); mov(,dz); b_ctrl=udat;
       select(sp,sp,plpl); dat(word,); dcx(,za); mare=0; jump(prox);
rst_7: select(wz,wz,plpl); dat(,); clear;
       nop; next_addr=v10; b_ctrl=udat;
       select(zw,zw,plpl); dat(,); mov(,dz); b_ctrl=udat;
       select(sp,sp,plpl); dat(word,); dcx(,za); mare=0; jump(prox);
trap:  select(wz,wz,plpl); dat(,); clear;
       nop; next_addr=v11;
       select(zw,zw,plpl); dat(,); mov(,dz); b_ctrl=udat;
       select(sp,sp,plpl); dat(word,); dcx(,za); mare=0; jump(prox);
rst75: select(wz,wz,plpl); dat(,); clear;
       nop; next_addr=v12; b_ctrl=udat;
       select(zw,zw,plpl); dat(,); mov(,dz); b_ctrl=udat;
       select(sp,sp,plpl); dat(word,); dcx(,za); mare=0; jump(prox);
rst65: select(wz,wz,plpl); dat(,); clear;
       nop; next_addr=v13; b_ctrl=udat;
       select(zw,zw,plpl); dat(,); mov(,dz); b_ctrl=udat;
       select(sp,sp,plpl); dat(word,); dcx(,za); mare=0; jump(prox);
rst55: select(wz,wz,plpl); dat(,); clear;
       nop; next_addr=v14; b_ctrl=udat;
       select(zw,zw,plpl); dat(,); mov(,dz); b_ctrl=udat;
       select(sp,sp,plpl); dat(word,); dcx(,za); mare=0; jump(prox);
sint:  nop; ir_en=ld_op; inta=0;
       jumap;
hold:  vect; nop; hlda=1; ts=1;
       nop; jump(hold);
end;
```

```
 1                              TITLE                    CPA MICROPROGRAM;
 2                              NOBINARY;
 3                              LISTING;
 4
 5                              [                        **************
 6                                                       MICROPROGRAMA
 7                                                       **************
 8                              ]
 9
10  0000  006EED6D40493377  init:   nop, next_addr=v0, b_ctrl=udat;
11  0001  000EED6D434B37DD          select(r13,r13,plpl), dat(,), mov(,dz), b
12  0002  060EED6D40493377          nop, next_addr=v1, b_ctrl=udat;
13  0003  000EED6D434B37CC          select(r12,r12,plpl), dat(,), mov(,dz), b
14  0004  007EE96D40493377          stm, nop, next_addr=v3, b_ctrl=udat;
15  0005  000EED05474B44FF          select(pc,pc,plpl), dat(word,), clear, ma
16  0006  000EED1D70493377  fet_1:  fetch1(op), nop;
17  0007  0006C50D30493377  fet_2:  fetch2, nop, vect;
18  0008  0002ED05474B0CFF  decod:  select(pc,pc,plpl), dat(word,), inx(,za),
19  0009  000EED05474A3444  inesc1: select(hl,hl,plpl),dat(word,),mov(ram_a,z
20  000A  000EED15574B34FF          fetch1(mw),select(pc,pc,plpl),dat(word,),
21  000B  0063FD0C50493377          write(mem), nop, jump(fet_1);
22  000C  0073FD1D704B3477  movrr:  select(,,), dat(,), mov(,za), fetch1(op),
23  000D  000EED05474A3444  movrm:  select(hl,hl,plpl),dat(word,),mov(ram_a,z
24  000E  000EED15674B34FF          fetch1(mr),select(pc,pc,plpl),dat(word,),
25  000F  000EED0D20493377          read(mem), nop;
26  0010  0073FD1D794B3777          select(,,irpl), dat(,ex), mov(,dz), fetch
27  0011  0093FD0942493477  movmr:  select(,,plir), dat(,), mov(noper,za), mb
28  0012  0073FD1D774B3449   sphl:  select(sp,hl,plpl), dat(word,), mov(,za),
29  0013  000EED15674B0CFF  mvird:  fetch1(mr), select(pc,pc,plpl), dat(word,
30  0014  000EED0D20493377          read(mem), nop;
31  0015  0073FD1D794B3777          select(,,irpl), dat(,ex), mov(,dz), fetch
32  0016  000EED15674B0CFF  mvimd:  fetch1(mr), select(pc,pc,plpl), dat(word,
33  0017  000EED0D20493377          read(mem), nop;
34  0018  0093FD0948493777          select(,,), dat(,ex), mov(noper,dz), mbre
35  0019  000EED15674B0CFF  lxibc:  fetch1(mr), select(pc,pc,plpl), dat(word,
36  001A  000EED0D20493377          read(mem), nop;
37  001B  000EED0D4B4B3771          select(cb,,plpl), dat(,ex), mov(,dz);
38  001C  000EED15674B0CFF          fetch1(mr), select(pc,pc,plpl), dat(word,
39  001D  000EED0D20493377          read(mem), nop;
40  001E  0073FD1D7B4B3770          select(bc,,plpl), dat(,ex), mov(,dz), fet
41  001F  000EED15674B0CFF  lxide:  fetch1(mr), select(pc,pc,plpl), dat(word,
42  0020  000EED0D20493377          read(mem), nop;
43  0021  000EED0D4B4B3773          select(ed,,plpl), dat(,ex), mov(,dz);
44  0022  000EED15674B0CFF          fetch1(mr), select(pc,pc,plpl), dat(word,
45  0023  000EED0D20493377          read(mem), nop;
46  0024  0073FD1D7B4B3772          select(de,,plpl), dat(,ex), mov(,dz), fet
47  0025  000EED15674B0CFF  lxihl:  fetch1(mr), select(pc,pc,plpl), dat(word,
48  0026  000EED0D20493377          read(mem), nop;
49  0027  000EED0D4B4B3775          select(lh,,plpl), dat(,ex), mov(,dz);
50  0028  000EED15674B0CFF          fetch1(mr), select(pc,pc,plpl), dat(word,
51  0029  000EED0D20493377          read(mem), nop;
52  002A  0073FD1D7B4B3774          select(hl,,plpl), dat(,ex), mov(,dz), fet
53  002B  000EED15674B0CFF  lxisp:  fetch1(mr), select(pc,pc,plpl), dat(word,
54  002C  000EED0D20493377          read(mem), nop;
55  002D  000EED0D4B4B3778          select(r8,,plpl), dat(,ex), mov(,dz);
```

```
 56  002E  000EED15674B0CFF          fetch1(mr), select(pc,pc,plpl), dat(word,
 57  002F  000EED0D20493377          read(mem), nop;
 58  0030  0073FD1D7B4B3779          select(sp,,plpl), dat(,ex), mov(,dz), fet
 59  0031  000EED15674B0CFF  lda:    fetch1(mr), select(pc,pc,plpl), dat(word,
 60  0032  000EED0D20493377          read(mem), nop;
 61  0033  000EED0D4B4B377B          select(zw,,plpl), dat(,ex), mov(,dz);
 62  0034  000EED15674B0CFF          fetch1(mr), select(pc,pc,plpl), dat(word,
 63  0035  000EED0D20493377          read(mem), nop;
 64  0036  000EED0D4B4B377A          select(wz,,plpl), dat(,ex), mov(,dz);
 65  0037  000EED05474A34AA          select(wz,wz,plpl), dat(word,), mov(ram_a
 66  0038  000EED15674B34FF          fetch1(mr), select(pc,pc,plpl), dat(word,
 67  0039  000EED0D20493377          read(mem), nop;
 68  003A  0073FD1D7B4B3777          select(ax,,plpl), dat(,ex), mov(,dz),fetc
 69  003B  000EED15674B0CFF  sta:    fetch1(mr), select(pc,pc,plpl), dat(word,
 70  003C  000EED0D20493377          read(mem), nop;
 71  003D  000EED0D4B4B377B          select(zw,,plpl), dat(,ex), mov(,dz);
 72  003E  000EED15674B0CFF          fetch1(mr), select(pc,pc,plpl), dat(word,
 73  003F  000EED0D20493377          read(mem), nop;
 74  0040  000EED0D4B4B377A          select(wz,,plpl), dat(,ex), mov(,dz);
 75  0041  000EED09434B3477          select(,ax,plpl), dat(,), mov(,za), mbre=
 76  0042  000EED05474A34AA          select(wz,wz,plpl), dat(word,), mov(ram_a
 77  0043  000EED15574B34FF          fetch1(mw), select(pc,pc,plpl), dat(word,
 78  0044  0063FD0C50493377          write(mem), nop, jump(fet_1);
 79  0045  000EED15674B0CFF  lhld:   fetch1(mr), select(pc,pc,plpl), dat(word,
 80  0046  000EED0D20493377          read(mem), nop;
 81  0047  000EED0D4B4B377B          select(zw,,plpl), dat(,ex), mov(,dz);
 82  0048  000EED15674B0CFF          fetch1(mr), select(pc,pc,plpl), dat(word,
 83  0049  000EED0D20493377          read(mem), nop;
 84  004A  000EED0D4B4B377A          select(wz,,plpl), dat(,ex), mov(,dz);
 85  004B  000EED05474A0CAA          select(wz,wz,plpl), dat(word,), inx(ram_a
 86  004C  000EED15674B34AA          fetch1(mr), select(wz,wz,plpl), dat(word,
 87  004D  000EED0D20493377          read(mem), nop;
 88  004E  000EED0D4B4B3755          select(lh,lh,plpl), dat(,ex), mov(,dz);
 89  004F  000EED15674B34FF          fetch1(mr),select(pc,pc,plpl), dat(word,)
 90  0050  000EED0D20493377          read(mem), nop;
 91  0051  0063FD0D4B4B3774          select(hl,,plpl), dat(,ex), mov(,dz), jum
 92  0052  000EED15674B0CFF  shld:   fetch1(mr), select(pc,pc,plpl), dat(word,
 93  0053  000EED0D20493377          read(mem), nop;
 94  0054  000EED0D4B4B377B          select(zw,,plpl), dat(,ex), mov(,dz);
 95  0055  000EED15674B0CFF          fetch1(mr), select(pc,pc,plpl), dat(word,
 96  0056  000EED0D20493377          read(mem), nop;
 97  0057  000EED0D4B4B377A          select(wz,,plpl), dat(,ex), mov(,dz);
 98  0058  000EED05474A0CAA          select(wz,wz,plpl), dat(word,), inx(ram_a
 99  0059  000EED19534B3455          fetch1(mw), select(lh,lh,plpl), dat(,), m
100  005A  000EED04574B34AA          write(mem), select(wz,wz,plpl), dat(word,
101  005B  000EED19534B3444          fetch1(mw), select(hl,hl,plpl), dat(,), m
102  005C  0063FD04574B34FF          write(mem),select(pc,pc,plpl),dat(word,),
103  005D  000EED05444B3477  ldaxr:  select(,,irir), dat(word,), mov(,za), mar
104  005E  000EED15674B34FF          fetch1(mr), select(pc,pc,plpl), dat(word,
105  005F  000EED0D20493377          read(mem), nop;
106  0060  0073FD1D7A4B3777          select(ax,,plir), dat(,ex), mov(,dz), fet
107  0061  000EED05444B3477  staxr:  select(,,irir), dat(word,), mov(,za), mar
108  0062  000EED19534B3477          fetch1(mw), select(ax,ax,plpl), dat(,), m
109  0063  0063FD04574B34FF          write(mem),select(pc,pc,plpl),dat(word,),
110  0064  000EED0D474B344A  xchg:   select(wz,hl,plpl), dat(word,), mov(,za);
```

```
111  0065  000EED0D474B3424          select(hl,de,plpl); dat(word,); mov(,za);
112  0066  000EED0D474A3725          select(lh,de,plpl); dat(word,); mov(ram_a
113  0067  000EED0D474B34A2          select(de,wz,plpl); dat(word,); mov(,za);
114  0068  0073FD1D774A3723          select(ed,de,plpl); dat(word,); mov(ram_a
115  0069  0073FD1D720B0177  add_r:  select(ax,,plir); dat(,); add(,ab); fetch
116  006A  000EED05474A3444  add_m:  select(hl,hl,plpl);dat(word,);mov(ram_a,z
117  006B  000EED15674B34FF          fetch1(mr),select(pc,pc,plpl),dat(word,),
118  006C  000EED0D20493377          read(mem); nop;
119  006D  0073FD1D7B0B0577          select(ax,ax,plpl); dat(,ex); add(,da); f
120  006E  000EED15674B0CFF  adi:    fetch1(mr); select(pc,pc,plpl); dat(word,
121  006F  000EED0D20493377          read(mem); nop;
122  0070  0073FD1D7B0B0577          select(ax,ax,plpl); dat(,ex); add(,da); f
123  0071  0073FD1D720B8177  adc_r:  select(ax,,plir); dat(,); adc(,ab); fetch
124  0072  000EED05474A3444  adc_m:  select(hl,hl,plpl);dat(word,);mov(ram_a,z
125  0073  000EED15674B34FF          fetch1(mr),select(pc,pc,plpl),dat(word,),
126  0074  000EED0D20493377          read(mem); nop;
127  0075  0073FD1D7B0B8577          select(ax,ax,plpl); dat(,ex); adc(,da); f
128  0076  000EED15674B0CFF  aci:    fetch1(mr); select(pc,pc,plpl); dat(word,
129  0077  000EED0D20493377          read(mem); nop;
130  0078  0073FD1D7B0B8577          select(ax,ax,plpl); dat(,ex); adc(,da); f
131  0079  0073FD1D720B1977  sub_r:  select(ax,,plir); dat(,); subr(,ab); fetc
132  007A  000EED05474A3444  sub_m:  select(hl,hl,plpl);dat(word,);mov(ram_a,z
133  007B  000EED15674B34FF          fetch1(mr),select(pc,pc,plpl),dat(word,),
134  007C  000EED0D20493377          read(mem); nop;
135  007D  0073FD1D7B0B1D77          select(ax,ax,plpl); dat(,ex); subr(,da);
136  007E  000EED15674B0CFF  sui:    fetch1(mr); select(pc,pc,plpl); dat(word,
137  007F  000EED0D20493377          read(mem); nop;
138  0080  0073FD1D7B0B1D77          select(ax,ax,plpl); dat(,ex); subr(,da);
139  0081  0073FD1D720B9177  sbb_r:  select(ax,,plir); dat(,); sbbr(,ab); fetc
140  0082  000EED05474A3444  sbb_m:  select(hl,hl,plpl);dat(word,);mov(ram_a,z
141  0083  000EED15674B34FF          fetch1(mr),select(pc,pc,plpl),dat(word,),
142  0084  000EED0D20493377          read(mem); nop;
143  0085  0073FD1D7B0B9577          select(ax,ax,plpl); dat(,ex); sbbr(,da);
144  0086  000EED15674B0CFF  sbi:    fetch1(mr); select(pc,pc,plpl); dat(word,
145  0087  000EED0D20493377          read(mem); nop;
146  0088  0073FD1D7B0B9577          select(ax,ax,plpl); dat(,ex); sbbr(,da);
147  0089  0073FD1D711B0B77  inr_r:  select(,,irpl); dat(,); inr(,zb); fetch1(
148  008A  000EED05474A3444  inr_m:  select(hl,hl,plpl);dat(word,);mov(ram_a,z
149  008B  000EED15674B34FF          fetch1(mr),select(pc,pc,plpl),dat(word,),
150  008C  000EED0D20493377          read(mem); nop;
151  008D  0093FD094B1B0F77          select(,,plpl); dat(,ex); inr(,dz); mbre=
152  008E  0073FD1D711B1377  dcr_r:  select(,,irpl); dat(,); dcr(,zb); fetch1(
153  008F  000EED05474A3444  dcr_m:  select(hl,hl,plpl);dat(word,);mov(ram_a,z
154  0090  000EED15674B34FF          fetch1(mr),select(pc,pc,plpl),dat(word,),
155  0091  000EED0D20493377          read(mem); nop;
156  0092  0093FD094B1B2777          select(,,plpl); dat(,ex); dcrm, mbre=0; j
157  0093  000EED0D474B0C00  inx_b:  select(bc,bc,plpl); dat(word,); inx(,za);
158  0094  0073FD1D774A3701          select(cb,bc,plpl); dat(word,); mov(ram_a
159  0095  000EED0D474B0C22  inx_d:  select(de,de,plpl); dat(word,); inx(,za);
160  0096  0073FD1D774A3723          select(ed,de,plpl); dat(word,); mov(ram_a
161  0097  000EED0D474B0C44  inx_h:  select(hl,hl,plpl); dat(word,); inx(,za);
162  0098  0073FD1D774A3745          select(lh,hl,plpl); dat(word,); mov(ram_a
163  0099  000EED0D474B0C99  inxsp:  select(sp,sp,plpl); dat(word,); inx(,za);
164  009A  0073FD1D774A3798          select(r8,sp,plpl); dat(word,); mov(ram_a
165  009B  000EED0D474B1400  dcx_b:  select(bc,bc,plpl); dat(word,); dcx(,za);
```

```
166  009C  0073FD1D774A3701          select(cb,bc,plpl); dat(word,); mov(ram_a
167  009D  000EED0D474B1422  dcx_d:  select(de,de,plpl); dat(word,); dcx(,za);
168  009E  0073FD1D774A3723          select(ed,de,plpl); dat(word,); mov(ram_a
169  009F  000EED0D474B1444  dcx_h:  select(hl,hl,plpl); dat(word,); dcx(,za);
170  00A0  0073FD1D774A3745          select(lh,hl,plpl); dat(word,); mov(ram_a
171  00A1  000EED0D474B1499  dcxsp:  select(sp,sp,plpl); dat(word,); dcx(,za);
172  00A2  0073FD1D774A3798          select(r8,sp,plpl); dat(word,); mov(ram_a
173  00A3  000EED0D473B0104  dad_b:  select(hl,bc,plpl); dat(word,); add_rp(,a
174  00A4  0073FD1D774A3745          select(lh,hl,plpl); dat(word,); mov(ram_a
175  00A5  000EED0D473B0124  dad_d:  select(hl,de,plpl); dat(word,); add_rp(,a
176  00A6  0073FD1D774A3745          select(lh,hl,plpl); dat(word,); mov(ram_a
177  00A7  000EED0D473B0144  dad_h:  select(hl,hl,plpl); dat(word,); add_rp(,a
178  00A8  0073FD1D774A3745          select(lh,hl,plpl); dat(word,); mov(ram_a
179  00A9  000EED0D473B0194  dadsp:  select(hl,sp,plpl); dat(word,); add_rp(,a
180  00AA  0073FD1D774A3745          select(lh,hl,plpl); dat(word,); mov(ram_a
181  00AB  000EED0D434A3477  daa_a:  select(ax,ax,plpl); dat(,); mov(ram_a,za)
182  00AC  0AF38D0D434A3477          select(ax,ax,plpl); dat(,); mov(ram_a,za)
183  00AD  0B03AD0D40493377  e3:     nop; jcond(e2,daah);
184  00AE  0073FD1D70493377          fetch1(op); nop; jump(fet_2);
185  00AF  0AD3FD0D430B01D7  e1:     select(ax,r13,plpl);dat(,);add(,ab); jump
186  00B0  0073FD1D730B01C7  e2:     select(ax,r12,plpl);dat(,);add(,ab); fetc
187  00B1  0073FD1D722B4177  ana_r:  select(ax,,plir); dat(,); l_and(,ab); fet
188  00B2  000EED05474A3444  ana_m:  select(hl,hl,plpl);dat(word,);mov(ram_a,z
189  00B3  000EED15674B34FF          fetch1(mr);select(pc,pc,plpl);dat(word,);
190  00B4  000EED0D20493377          read(mem); nop;
191  00B5  0073FD1D7B2B4577          select(ax,ax,plpl); dat(,ex); l_and(,da);
192  00B6  000EED15674B0CFF  ani:    fetch1(mr); select(pc,pc,plpl); dat(word,
193  00B7  000EED0D20493377          read(mem); nop;
194  00B8  0073FD1D7B2B4577          select(ax,ax,plpl); dat(,ex); l_and(,da);
195  00B9  0073FD1D722B6177  xra_r:  select(ax,,plir); dat(,); l_xor(,ab); fet
196  00BA  000EED05474A3444  xra_m:  select(hl,hl,plpl);dat(word,);mov(ram_a,z
197  00BB  000EED15674B34FF          fetch1(mr);select(pc,pc,plpl);dat(word,);
198  00BC  000EED0D20493377          read(mem); nop;
199  00BD  0073FD1D7B2B6577          select(ax,ax,plpl); dat(,ex); l_xor(,da);
200  00BE  000EED15674B0CFF  xri:    fetch1(mr); select(pc,pc,plpl); dat(word,
201  00BF  000EED0D20493377          read(mem); nop;
202  00C0  0073FD1D7B2B6577          select(ax,ax,plpl); dat(,ex); l_xor(,da);
203  00C1  0073FD1D722B3177  ora_r:  select(ax,,plir); dat(,); l_or(,ab); fetc
204  00C2  000EED05474A3444  ora_m:  select(hl,hl,plpl);dat(word,);mov(ram_a,z
205  00C3  000EED15674B34FF          fetch1(mr);select(pc,pc,plpl);dat(word,);
206  00C4  000EED0D20493377          read(mem); nop;
207  00C5  0073FD1D7B2B3577          select(ax,ax,plpl); dat(,ex); l_or(,da);
208  00C6  000EED15674B0CFF  ori:    fetch1(mr); select(pc,pc,plpl); dat(word,
209  00C7  000EED0D20493377          read(mem); nop;
210  00C8  0073FD1D7B2B3577          select(ax,ax,plpl); dat(,ex); l_or(,da);
211  00C9  0073FD1D72091977  cmp_r:  select(ax,,plir); dat(,); subr(noper,ab);
212  00CA  000EED05474A3444  cmp_m:  select(hl,hl,plpl);dat(word,);mov(ram_a,z
213  00CB  000EED15674B34FF          fetch1(mr);select(pc,pc,plpl);dat(word,);
214  00CC  000EED0D20493377          read(mem); nop;
215  00CD  0073FD1D7B091D77          select(,ax,plpl); dat(,ex); subr(noper,da
216  00CE  000EED15674B0CFF  cpi:    fetch1(mr); select(pc,pc,plpl); dat(word,
217  00CF  000EED0D20493377          read(mem); nop;
218  00D0  0073FD1D7B091D77          select(ax,ax,plpl); dat(,ex); subr(noper,
219  00D1  0073FD1D73573377  rlc_a:  select(ax,,plpl); dat(,); rlc, fetch1(op)
220  00D2  0073FD1D73553377  rrc_a:  select(ax,,plpl); dat(,); rrc, fetch1(op)
```

```
221  00D3  0073FD1D735F3377  ral_a:  select(ax,,plpl), dat(,), ral, fetch1(op)
222  00D4  0073FD1D735D3377  rar_a:  select(ax,,plpl), dat(,), rar, fetch1(op)
223  00D5  0073FD1D734B7377  cma_a:  select(ax,,plpl), dat(,), cma(,zb), fetch
224  00D6  0073FD1D70693377  cmc_f:  cmc, nop, fetch1(op), jump(fet_2);
225  00D7  001EED6D40493377  stc_f:  nop, next_addr=v2, b_ctrl=udat;
226  00D8  000EED6D434B37AA          select(wz,wz,plpl), dat(,), mov(,dz), b_c
227  00D9  0073FD1D735D33AA          select(wz,wz,plpl), dat(,), rar, fetch1(o
228  00DA  000EED15674B0CFF  jmp_a:  fetch1(mr), select(pc,pc,plpl), dat(word,
229  00DB  000EED0D20493377          read(mem), nop;
230  00DC  000EED0D4B4B377B          select(zw,,plpl), dat(,ex), mov(,dz);
231  00DD  000EED15674B0CFF          fetch1(mr), select(pc,pc,plpl), dat(word,
232  00DE  000EED0D20493377          read(mem), nop;
233  00DF  000EED0D4B4B377A          select(wz,,plpl), dat(,ex), mov(,dz);
234  00E0  0063FD05474B34AF          select(pc,wz,plpl),dat(word,),mov(,za),ma
235  00E1  0DA33D0D40493377  jnz:    nop, jcond(jmp_a,nozero);
236  00E2  000EED15674B0CFF          fetch1(mr), select(pc,pc,plpl), dat(word,
237  00E3  000EED0D20493377          read(mem), nop;
238  00E4  000EED0D4B4B377B          select(zw,,plpl), dat(,ex), mov(,dz);
239  00E5  000EED15674B0CFF          fetch1(mr), select(pc,pc,plpl), dat(word,
240  00E6  000EED0D20493377          read(mem), nop;
241  00E7  000EED0D4B4B377A          select(wz,,plpl), dat(,ex), mov(,dz);
242  00E8  0073FD1D70493377          fetch1(op), nop, jump(fet_2);
243  00E9  0DA32D0D404B3177  jz:     jcond(jmp_a,zero);
244  00EA  000EED15674B0CFF          fetch1(mr), select(pc,pc,plpl), dat(word,
245  00EB  000EED0D20493377          read(mem), nop;
246  00EC  000EED0D4B4B377B          select(zw,,plpl), dat(,ex), mov(,dz);
247  00ED  000EED15674B0CFF          fetch1(mr), select(pc,pc,plpl), dat(word,
248  00EE  000EED0D20493377          read(mem), nop;
249  00EF  000EED0D4B4B377A          select(wz,,plpl), dat(,ex), mov(,dz);
250  00F0  0073FD1D70493377          fetch1(op), nop, jump(fet_2);
251  00F1  0DA37D0D40493377  jnc:    nop, jcond(jmp_a,nocy);
252  00F2  000EED15674B0CFF          fetch1(mr), select(pc,pc,plpl), dat(word,
253  00F3  000EED0D20493377          read(mem), nop;
254  00F4  000EED0D4B4B377B          select(zw,,plpl), dat(,ex), mov(,dz);
255  00F5  000EED15674B0CFF          fetch1(mr), select(pc,pc,plpl), dat(word,
256  00F6  000EED0D20493377          read(mem), nop;
257  00F7  000EED0D4B4B377A          select(wz,,plpl), dat(,ex), mov(,dz);
258  00F8  0073FD1D70493377          fetch1(op), nop, jump(fet_2);
259  00F9  0DA36D0D40493377  jc:     nop, jcond(jmp_a,cy);
260  00FA  000EED15674B0CFF          fetch1(mr), select(pc,pc,plpl), dat(word,
261  00FB  000EED0D20493377          read(mem), nop;
262  00FC  000EED0D4B4B377B          select(zw,,plpl), dat(,ex), mov(,dz);
263  00FD  000EED15674B0CFF          fetch1(mr), select(pc,pc,plpl), dat(word,
264  00FE  000EED0D20493377          read(mem), nop;
265  00FF  000EED0D4B4B377A          select(wz,,plpl), dat(,ex), mov(,dz);
266  0100  0073FD1D70493377          fetch1(op), nop, jump(fet_2);
267  0101  0DA35D0D40493377  jpo:    nop, jcond(jmp_a,impar);
268  0102  000EED15674B0CFF          fetch1(mr), select(pc,pc,plpl), dat(word,
269  0103  000EED0D20493377          read(mem), nop;
270  0104  000EED0D4B4B377B          select(zw,,plpl), dat(,ex), mov(,dz);
271  0105  000EED15674B0CFF          fetch1(mr), select(pc,pc,plpl), dat(word,
272  0106  000EED0D20493377          read(mem), nop;
273  0107  000EED0D4B4B377A          select(wz,,plpl), dat(,ex), mov(,dz);
274  0108  0073FD1D70493377          fetch1(op), nop, jump(fet_2);
275  0109  0DA34D0D40493377  jpe:    nop, jcond(jmp_a,par);
```

```
276  010A  000EED15674B0CFF          fetch1(mr); select(pc;pc;plpl); dat(word;
277  010B  000EED0D20493377          read(mem); nop;
278  010C  000EED0D4B4B377B          select(zw;;plpl); dat(;ex); mov(;dz);
279  010D  000EED15674B0CFF          fetch1(mr); select(pc;pc;plpl); dat(word;
280  010E  000EED0D20493377          read(mem); nop;
281  010F  000EED0D4B4B377A          select(wz;;plpl); dat(;ex); mov(;dz);
282  0110  0073FD1D70493377          fetch1(op); nop; jump(fet_2);
283  0111  0DA31D0D40493377  jp:     nop; jcond(jmp_a;pos);
284  0112  000EED15674B0CFF          fetch1(mr); select(pc;pc;plpl); dat(word;
285  0113  000EED0D20493377          read(mem); nop;
286  0114  000EED0D4B4B377B          select(zw;;plpl); dat(;ex); mov(;dz);
287  0115  000EED15674B0CFF          fetch1(mr); select(pc;pc;plpl); dat(word;
288  0116  000EED0D20493377          read(mem); nop;
289  0117  000EED0D4B4B377A          select(wz;;plpl); dat(;ex); mov(;dz);
290  0118  0073FD1D70493377          fetch1(op); nop; jump(fet_2);
291  0119  0DA30D0D40493377  jm:     nop; jcond(jmp_a;neg);
292  011A  000EED15674B0CFF          fetch1(mr); select(pc;pc;plpl); dat(word;
293  011B  000EED0D20493377          read(mem); nop;
294  011C  000EED0D4B4B377B          select(zw;;plpl); dat(;ex); mov(;dz);
295  011D  000EED15674B0CFF          fetch1(mr); select(pc;pc;plpl); dat(word;
296  011E  000EED0D20493377          read(mem); nop;
297  011F  000EED0D4B4B377A          select(wz;;plpl); dat(;ex); mov(;dz);
298  0120  0073FD1D70493377          fetch1(op); nop; jump(fet_2);
299  0121  000EED0D474B1499  calla:  select(sp;sp;plpl); dat(word;); dcx(;za);
300  0122  000EED15674B0CFF          fetch1(mr); select(pc;pc;plpl); dat(word;
301  0123  000EED0D20493377          read(mem); nop;
302  0124  000EED0D4B4B377B          select(zw;;plpl); dat(;ex); mov(;dz);
303  0125  000EED15674B0CFF          fetch1(mr); select(pc;pc;plpl); dat(word;
304  0126  000EED0D20493377          read(mem); nop;
305  0127  000EED0D4B4B377A          select(wz;;plpl); dat(;ex); mov(;dz);
306  0128  000EED05474A1499          select(sp;sp;plpl); dat(word;); dcx(ram_a
307  0129  000EED19534B34FF          fetch1(mw); select(pc;pc;plpl); dat(;); m
308  012A  000EED04574B3499          write(mem);select(sp;sp;plpl);dat(word;);
309  012B  000EED1953CB34EE          fetch1(mw);select(r14;r14;plpl);dat(;);mo
310  012C  0063FD04574B34AF          write(mem);select(pc;wz;plpl);dat(word;);
311  012D  12133D0D40493377  cnz:    nop; jcond(calla;nozero);
312  012E  000EED15674B0CFF          fetch1(mr); select(pc;pc;plpl); dat(word;
313  012F  000EED0D20493377          read(mem); nop;
314  0130  000EED0D4B4B377B          select(zw;;plpl); dat(;ex); mov(;dz);
315  0131  000EED15674B0CFF          fetch1(mr); select(pc;pc;plpl); dat(word;
316  0132  000EED0D20493377          read(mem); nop;
317  0133  000EED0D4B4B377A          select(wz;;plpl); dat(;ex); mov(;dz);
318  0134  0073FD1D70493377          fetch1(op); nop; jump(fet_2);
319  0135  12132D0D40493377  cz:     nop; jcond(calla;zero);
320  0136  000EED15674B0CFF          fetch1(mr); select(pc;pc;plpl); dat(word;
321  0137  000EED0D20493377          read(mem); nop;
322  0138  000EED0D4B4B377B          select(zw;;plpl); dat(;ex); mov(;dz);
323  0139  000EED15674B0CFF          fetch1(mr); select(pc;pc;plpl); dat(word;
324  013A  000EED0D20493377          read(mem); nop;
325  013B  000EED0D4B4B377A          select(wz;;plpl); dat(;ex); mov(;dz);
326  013C  0073FD1D70493377          fetch1(op); nop; jump(fet_2);
327  013D  12137D0D40493377  cnc:    nop; jcond(calla;nocy);
328  013E  000EED15674B0CFF          fetch1(mr); select(pc;pc;plpl); dat(word;
329  013F  000EED0D20493377          read(mem); nop;
330  0140  000EED0D4B4B377B          select(zw;;plpl); dat(;ex); mov(;dz);
```

```
331  0141  000EED15674B0CFF          fetch1(mr); select(pc;pc;plpl); dat(word;
332  0142  000EED0D20493377          read(mem); nop;
333  0143  000EED0D4B4B377A          select(wz;;plpl); dat(;ex); mov(;dz);
334  0144  0073FD1D70493377          fetch1(op); nop; jump(fet_2);
335  0145  12136D0D40493377  cc:     nop; jcond(calla;cy);
336  0146  000EED15674B0CFF          fetch1(mr); select(pc;pc;plpl); dat(word;
337  0147  000EED0D20493377          read(mem); nop;
338  0148  000EED0D4B4B377B          select(zw;;plpl); dat(;ex); mov(;dz);
339  0149  000EED15674B0CFF          fetch1(mr); select(pc;pc;plpl); dat(word;
340  014A  000EED0D20493377          read(mem); nop;
341  014B  000EED0D4B4B377A          select(wz;;plpl); dat(;ex); mov(;dz);
342  014C  0073FD1D70493377          fetch1(op); nop; jump(fet_2);
343  014D  12135D0D40493377  cpo:    nop; jcond(calla;impar);
344  014E  000EED15674B0CFF          fetch1(mr); select(pc;pc;plpl); dat(word;
345  014F  000EED0D20493377          read(mem); nop;
346  0150  000EED0D4B4B377B          select(zw;;plpl); dat(;ex); mov(;dz);
347  0151  000EED15674B0CFF          fetch1(mr); select(pc;pc;plpl); dat(word;
348  0152  000EED0D20493377          read(mem); nop;
349  0153  000EED0D4B4B377A          select(wz;;plpl); dat(;ex); mov(;dz);
350  0154  0073FD1D70493377          fetch1(op); nop; jump(fet_2);
351  0155  12134D0D40493377  cpe:    nop; jcond(calla;par);
352  0156  000EED15674B0CFF          fetch1(mr); select(pc;pc;plpl); dat(word;
353  0157  000EED0D20493377          read(mem); nop;
354  0158  000EED0D4B4B377B          select(zw;;plpl); dat(;ex); mov(;dz);
355  0159  000EED15674B0CFF          fetch1(mr); select(pc;pc;plpl); dat(word;
356  015A  000EED0D20493377          read(mem); nop;
357  015B  000EED0D4B4B377A          select(wz;;plpl); dat(;ex); mov(;dz);
358  015C  0073FD1D70493377          fetch1(op); nop; jump(fet_2);
359  015D  12131D0D40493377  cp:     nop; jcond(calla;pos);
360  015E  000EED15674B0CFF          fetch1(mr); select(pc;pc;plpl); dat(word;
361  015F  000EED0D20493377          read(mem); nop;
362  0160  000EED0D4B4B377B          select(zw;;plpl); dat(;ex); mov(;dz);
363  0161  000EED15674B0CFF          fetch1(mr); select(pc;pc;plpl); dat(word;
364  0162  000EED0D20493377          read(mem); nop;
365  0163  000EED0D4B4B377A          select(wz;;plpl); dat(;ex); mov(;dz);
366  0164  0073FD1D70493377          fetch1(op); nop; jump(fet_2);
367  0165  12130D0D40493377  cm:     nop; jcond(calla;neg);
368  0166  000EED15674B0CFF          fetch1(mr); select(pc;pc;plpl); dat(word;
369  0167  000EED0D20493377          read(mem); nop;
370  0168  000EED0D4B4B377B          select(zw;;plpl); dat(;ex); mov(;dz);
371  0169  000EED15674B0CFF          fetch1(mr); select(pc;pc;plpl); dat(word;
372  016A  000EED0D20493377          read(mem); nop;
373  016B  000EED0D4B4B377A          select(wz;;plpl); dat(;ex); mov(;dz);
374  016C  0073FD1D70493377          fetch1(op); nop; jump(fet_2);
375  016D  000EED05474A0C99  retn:   select(sp;sp;plpl); dat(word;); inx(ram_a
376  016E  000EED15674A0C99          fetch1(mr);select(sp;sp;plpl);dat(word;);
377  016F  000EED0D20493377          nop; read(mem) ;
378  0170  000EED1D6B4B377E          fetch1(mr);select(r14;;plpl); dat(;ex); m
379  0171  000EED0D20493377          nop; read(mem);
380  0172  000EED0D4B4B377F          select(pc;;plpl); dat(;ex); mov(;dz);
381  0173  0063FD05474B34FF          select(pc;pc;plpl);dat(word;);mov(;za);ma
382  0174  16D33D0D40493377  rnz:    nop; jcond(retn;nozero);
383  0175  0073FD1D70493377          fetch1(op); nop; jump(fet_2);
384  0176  16D32D0D40493377  rz:     nop; jcond(retn;zero);
385  0177  0073FD1D70493377          fetch1(op); nop; jump(fet_2);
```

```
386  0178  16D37D0D40493377  rnc:    nop; jcond(retn;nocy);
387  0179  0073FD1D70493377          fetch1(op); nop; jump(fet_2);
388  017A  16D36D0D40493377  rc:     nop; jcond(retn;cy);
389  017B  0073FD1D70493377          fetch1(op); nop; jump(fet_2);
390  017C  16D35D0D40493377  rpo:    nop; jcond(retn;impar);
391  017D  0073FD1D70493377          fetch1(op); nop; jump(fet_2);
392  017E  16D34D0D40493377  rpe:    nop; jcond(retn;par);
393  017F  0073FD1D70493377          fetch1(op); nop; jump(fet_2);
394  0180  16D31D0D40493377  rp:     nop; jcond(retn;pos);
395  0181  0073FD1D70493377          fetch1(op); nop; jump(fet_2);
396  0182  16D30D0D40493377  rm:     nop; jcond(retn;neg);
397  0183  0073FD1D70493377          fetch1(op); nop; jump(fet_2);
398  0184  0063FD05474B344F  pchl:   select(pc;hl;plpl); dat(word;); mov(;za);
399  0185  000EED05474B1499  pushb:  select(sp;sp;plpl); dat(word;); dcx(;za);
400  0186  000EED15574B1499          select(sp;sp;plpl); dat(word;); dcx(;za);
401  0187  000EED09434A3400          select(bc;bc;plpl); dat(;); mov(ram_a;za)
402  0188  000EED0C50493377          nop; write(mem);
403  0189  000EED19534A3411          select(cb;cb;plpl); dat(;); mov(ram_a;za)
404  018A  0063FD04574B34FF          select(pc;pc;plpl);dat(word;);mov(;za);ma
405  018B  000EED05474B1499  pushd:  select(sp;sp;plpl); dat(word;); dcx(;za);
406  018C  000EED15574B1499          select(sp;sp;plpl); dat(word;); dcx(;za);
407  018D  000EED09434A3422          select(de;de;plpl); dat(;); mov(ram_a;za)
408  018E  000EED0C50493377          nop; write(mem);
409  018F  000EED19534A3433          select(ed;ed;plpl); dat(;); mov(ram_a;za)
410  0190  0063FD04574B34FF          select(pc;pc;plpl);dat(word;);mov(;za);ma
411  0191  000EED05474B1499  pushh:  select(sp;sp;plpl); dat(word;); dcx(;za);
412  0192  000EED15574B1499          select(sp;sp;plpl); dat(word;); dcx(;za);
413  0193  000EED09434A3444          select(hl;hl;plpl); dat(;); mov(ram_a;za)
414  0194  000EED0C50493377          nop; write(mem);
415  0195  000EED19534A3455          select(lh;lh;plpl); dat(;); mov(ram_a;za)
416  0196  0063FD04574B34FF          select(pc;pc;plpl);dat(word;);mov(;za);ma
417  0197  000EED05474B1499  pushp:  select(sp;sp;plpl); dat(word;); dcx(;za);
418  0198  000EED15574B1499          select(sp;sp;plpl); dat(word;); dcx(;za);
419  0199  000EED09434A3477          select(ax;ax;plpl); dat(;); mov(ram_a;za)
420  019A  000EED0C50493377          nop; write(mem);
421  019B  000EED3950493377          nop; b_ctrl=foe; mbre=0; fetch1(mw);
422  019C  0063FD04574B34FF          select(pc;pc;plpl);dat(word;);mov(;za);ma
423  019D  000EED05474A0C99  popb:   select(sp;sp;plpl); dat(word;); inx(ram_a
424  019E  000EED15674B3499          select(sp;sp;plpl); dat(word;); mov(;za);
425  019F  000EED0D20493377          nop; read(mem);
426  01A0  000EED0D4B4B3711          select(cb;cb;plpl); dat(;ex); mov(;dz);
427  01A1  000EED1D674B0C99          select(sp;sp;plpl); dat(word;); inx(;za);
428  01A2  000EED05274B34FF          select(pc;pc;plpl);dat(word;);mov(;za);ma
429  01A3  0073FD1D7B4B3700          select(bc;bc;plpl); dat(;ex); mov(;dz); f
430  01A4  000EED05474A0C99  popd:   select(sp;sp;plpl); dat(word;); inx(ram_a
431  01A5  000EED15674B3499          select(sp;sp;plpl); dat(word;); mov(;za);
432  01A6  000EED0D20493377          nop; read(mem);
433  01A7  000EED0D4B4B3733          select(ed;ed;plpl); dat(;ex); mov(;dz);
434  01A8  000EED1D674B0C99          select(sp;sp;plpl); dat(word;); inx(;za);
435  01A9  000EED05274B34FF          select(pc;pc;plpl);dat(word;);mov(;za);ma
436  01AA  0073FD1D7B4B3722          select(de;de;plpl); dat(;ex); mov(;dz); f
437  01AB  000EED05474A0C99  poph:   select(sp;sp;plpl); dat(word;); inx(ram_a
438  01AC  000EED15674B3499          select(sp;sp;plpl); dat(word;); mov(;za);
439  01AD  000EED0D20493377          nop; read(mem);
440  01AE  000EED0D4B4B3755          select(lh;lh;plpl); dat(;ex); mov(;dz);
```

```
441  01AF  000EED1D674B0C99          select(sp,sp,plpl); dat(word,); inx(,za),
442  01B0  000EED05274B34FF          read(mem); select(pc,pc,plpl); dat(word,)
443  01B1  0073FD1D7B4B3744          select(hl,hl,plpl); dat(,ex); mov(,dz); f
444  01B2  000EED05474A0C99  popp:   select(sp,sp,plpl); dat(word,); inx(ram_a
445  01B3  000EED15674B3499          select(sp,sp,plpl); dat(word,); mov(,za),
446  01B4  000EED0D20493377          nop; read(mem);
447  01B5  000EED0D4B793777          select(,,plpl); dat(,ex),alu=or_rs,destin
448  01B6  000EED1D674A0C99          select(sp,sp,plpl); dat(word,); inx(ram_a
449  01B7  000EED05234B34FF          read(mem); select(pc,pc,plpl); mov(,za),
450  01B8  0073FD1D7B4B3777          select(ax,ax,plpl); dat(,ex); mov(,dz); f
451  01B9  000EED05474A0C99  xthl:   select(sp,sp,plpl); dat(word,); inx(ram_a
452  01BA  000EED15674B3499          fetch1(mr); select(sp,sp,plpl); dat(word,
453  01BB  000EED0D20493377          nop; read(mem);
454  01BC  000EED1D6B4B37BB          fetch1(mr); select(zw,zw,plpl); dat(,ex),
455  01BD  000EED09234B3444          read(mem); select(hl,hl,plpl); dat(,); mo
456  01BE  000EED1D5B4B37AA          fetch1(mw); select(wz,wz,plpl); dat(,ex),
457  01BF  000EED04574B1499          write(mem); select(sp,sp,plpl); dat(word,
458  01C0  000EED19534B3455          fetch1(mw); select(lh,lh,plpl); dat(,); m
459  01C1  000EED0C574B34A4          write(mem); select(hl,wz,plpl); dat(word,
460  01C2  000EED0D474B34B5          select(lh,zw,plpl); dat(word,); mov(,za);
461  01C3  0063FD05474B34FF          select(pc,pc,plpl); dat(word,); mov(,za),
462  01C4  000EED15674B0CFF  inp:    fetch1(mr); select(pc,pc,plpl); dat(word,
463  01C5  000EED052F4B37AA          read(mem); select(wz,wz,plpl); dat(word,e
464  01C6  000EED054F4B37BB          select(zw,zw,plpl); dat(word,ex); mov(,dz
465  01C7  000EED15674B34FF          fetch1(mr); select(pc,pc,plpl); dat(word,
466  01C8  0063FD0DAB4B3777          read(io); select(ax,ax,plpl); dat(,ex); m
467  01C9  000EED15674B0CFF  outp:   fetch1(mr); select(pc,pc,plpl); dat(word,
468  01CA  000EED052F4B37AA          read(mem); select(wz,wz,plpl); dat(word,e
469  01CB  000EED054F4B37BB          select(zw,zw,plpl); dat(word,ex); mov(,dz
470  01CC  000EED19534B3477          fetch1(mw); select(ax,ax,plpl); dat(,); m
471  01CD  0063FD04D74B34FF          write(io); select(pc,pc,plpl); dat(word,)
472  01CE  0073FF1D70493377  eint:   ei; fetch1(op); nop; jump(fet_2);
473  01CF  0073FD1D70493377  dint:   di; fetch1(op); nop; jump(fet_2);
474  01D0  000EED1D40493377  hlt:    nop; fetch1(hlt);
475  01D1  0006CD0F40493377  e5:     nop; halt; vect;
476  01D2  1D13FD0D40493377          nop; jump(e5);
477  01D3  0073FD1D70493377  nopr:   nop; fetch1(op); jump(fet_2);
478  01D4  000EED4D40493377  rim:    nop,b_ctrl=rd_mask;
479  01D5  0073FD1D734B3777          select(ax,ax,plpl),dat(,),mov(,dz),fetch1
480  01D6  0073F91D734A3477  sim:    select(ax,ax,plpl),dat(,),mov(ram_a,za),s
481  01D7  000EED0D434B44AA  rst_0:  select(wz,wz,plpl); dat(,); clear;
482  01D8  000EED0D434B44BB          select(zw,zw,plpl); dat(,); clear;
483  01D9  000EED05474B1499          select(sp,sp,plpl); dat(word,); dcx(,za),
484  01DA  000EED09434B34FF  prox:   select(pc,pc,plpl); dat(,); mov(,za); mbr
485  01DB  000EED15574B1499          fetch1(mw); select(sp,sp,plpl); dat(word,
486  01DC  000EED0C50493377          nop; write(mem);
487  01DD  000EED1953CB34EE          fetch1(mw),select(r14,r14,plpl),dat(,),mo
488  01DE  0063FD04574B34AF          write(mem),select(pc,wz,plpl),dat(word,),
489  01DF  000EED0D434B44AA  rst_1:  select(wz,wz,plpl); dat(,); clear;
490  01E0  008EED6D40493377          nop; next_addr=v4; b_ctrl=udat;
491  01E1  000EED6D434B37BB          select(zw,zw,plpl); dat(,); mov(,dz); b_c
492  01E2  1DA3FD05474B1499          select(sp,sp,plpl); dat(word,); dcx(,za),
493  01E3  000EED0D434B44AA  rst_2:  select(wz,wz,plpl); dat(,); clear;
494  01E4  010EED6D40493377          nop; next_addr=v5; b_ctrl=udat;
495  01E5  000EED6D434B37BB          select(zw,zw,plpl); dat(,); mov(,dz); b_c
```

```
496  01E6  1DA3FD05474B1499          select(sp,sp,plpl), dat(word,), dcx(,za),
497  01E7  000EED0D434B44AA  rst_3: select(wz,wz,plpl), dat(,), clear;
498  01E8  018EED6D40493377          nop, next_addr=v6, b_ctrl=udat;
499  01E9  000EED6D434B37BB          select(zw,zw,plpl), dat(,),mov(,dz),b_ctr
500  01EA  1DA3FD05474B1499          select(sp,sp,plpl), dat(word,), dcx(,za),
501  01EB  000EED0D434B44AA  rst_4: select(wz,wz,plpl), dat(,), clear;
502  01EC  020EED6D40493377          nop, next_addr=v7, b_ctrl=udat;
503  01ED  000EED6D434B37BB          select(zw,zw,plpl), dat(,), mov(,dz), b_c
504  01EE  1DA3FD05474B1499          select(sp,sp,plpl), dat(word,), dcx(,za),
505  01EF  000EED0D434B44AA  rst_5: select(wz,wz,plpl), dat(,), clear;
506  01F0  028EED6D40493377          nop, next_addr=v8, b_ctrl=udat;
507  01F1  000EED6D434B37BB          select(zw,zw,plpl), dat(,), mov(,dz), b_c
508  01F2  1DA3FD05474B1499          select(sp,sp,plpl), dat(word,), dcx(,za),
509  01F3  000EED0D434B44AA  rst_6: select(wz,wz,plpl), dat(,), clear;
510  01F4  030EED6D40493377          nop, next_addr=v9, b_ctrl=udat;
511  01F5  000EED6D434B37BB          select(zw,zw,plpl), dat(,), mov(,dz), b_c
512  01F6  1DA3FD05474B1499          select(sp,sp,plpl), dat(word,), dcx(,za),
513  01F7  000EED0D434B44AA  rst_7: select(wz,wz,plpl), dat(,), clear;
514  01F8  038EED6D40493377          nop, next_addr=v10, b_ctrl=udat;
515  01F9  000EED6D434B37BB          select(zw,zw,plpl), dat(,), mov(,dz), b_c
516  01FA  1DA3FD05474B1499          select(sp,sp,plpl), dat(word,), dcx(,za),
517  01FB  000EED0D434B44AA  trap:  select(wz,wz,plpl), dat(,), clear;
518  01FC  024EED0D40493377          nop, next_addr=v11;
519  01FD  000EED6D434B37BB          select(zw,zw,plpl), dat(,), mov(,dz), b_c
520  01FE  1DA3FD05474B1499          select(sp,sp,plpl), dat(word,), dcx(,za),
521  01FF  000EED0D434B44AA  rst75: select(wz,wz,plpl), dat(,), clear;
522  0200  03CEED6D40493377          nop, next_addr=v12, b_ctrl=udat;
523  0201  000EED6D434B37BB          select(zw,zw,plpl), dat(,), mov(,dz), b_c
524  0202  1DA3FD05474B1499          select(sp,sp,plpl), dat(word,), dcx(,za),
525  0203  000EED0D434B44AA  rst65: select(wz,wz,plpl), dat(,), clear;
526  0204  034EED6D40493377          nop, next_addr=v13, b_ctrl=udat;
527  0205  000EED6D434B37BB          select(zw,zw,plpl), dat(,), mov(,dz), b_c
528  0206  1DA3FD05474B1499          select(sp,sp,plpl), dat(word,), dcx(,za),
529  0207  000EED0D434B44AA  rst55: select(wz,wz,plpl), dat(,), clear;
530  0208  02CEED6D40493377          nop, next_addr=v14, b_ctrl=udat;
531  0209  000EED6D434B37BB          select(zw,zw,plpl), dat(,), mov(,dz), b_c
532  020A  1DA3FD05474B1499          select(sp,sp,plpl), dat(word,), dcx(,za),
533  020B  000EE40D40493377  sint:  nop, ir_en=ld_op, inta=0;
534  020C  0002ED0D404B3177          jumap;
535  020D  0006CD8F40493377  hold:  vect, nop, hlda=1, ts=1;
536  020E  20D3FD0D40493377          nop, jump(hold);
537                                  end;
```

Este fichero contiene 235 definiciones de constantes

AB	0001	ACI	0076	ADC_M	0072	ADC_R
ADD_R	0069	ADD_RS	0000	ADI	006E	ANA_M
AND_NOR	0005	AND_RS	0004	ANI	00B6	ARIT
BC	0000	BUS_Y	0000	BYTE	0000	CALLA
CB	0001	CC	0145	CJP	0003	CJS
CM	0165	CMA_A	00D5	CMCY	0006	CMC_F
CMP_R	00C9	CNC	013D	CNZ	012D	CONT
CPE	0155	CPI	00CE	CPO	014D	CRTN
CZ	0135	DA	0005	DAAH	000A	DAAL
DADSP	00A9	DAD_B	00A3	DAD_D	00A5	DAD_H
DCR_R	008E	DCXSP	00A1	DCX_B	009B	DCX_D
DE	0002	DECOD	0008	DINT	01CF	DS_INT
E1	00AF	E2	00B0	E3	00AD	E5
EINT	01CE	EN_INT	0001	EX	0001	FBUS
FET_1	0006	FET_2	0007	FOE	0001	HL
HOLD	020D	IMPAR	0005	IN	0000	INDC
INIT	0000	INP	01C4	INR_M	008A	INR_R
INVA	0004	INXSP	0099	INX_B	0093	INX_D
IO	0001	IRIR	0000	IRPL	0001	JC
JMAP	0002	JMP_A	00DA	JNC	00F1	JNZ
JPE	0109	JPO	0101	JZ	0000	LDA
LD_OP	0000	LEFT	0000	LH	0005	LHLD
LXIBC	0019	LXIDE	001F	LXIHL	0025	LXISP
MOVMR	0011	MOVRM	000D	MOVRR	000C	MR
MVIRD	0013	MW	0001	NEG	0000	NOCY
NOPER	0001	NOPR	01D3	NOXOR_RS	0007	NOZERO
OP	0003	ORA_M	00C2	ORA_R	00C1	ORI
OUTP	01C9	PAR	0004	PASS	000F	PC
PLIR	0002	PLPL	0003	POPB	019D	POPD
POPP	01B2	POS	0001	PROX	01DA	PUSH
PUSHD	018B	PUSHH	0191	PUSHP	0197	R12
R14	000E	RB	0008	RAL_A	00D3	RAMX
RAM_D	0005	RAM_F	0003	RAM_U	0007	RAR_A
RD_MASK	0002	RETN	016D	RIGHT	0001	RIM
RM	0182	RNC	0178	RNZ	0174	ROTA
RPE	017E	RPO	017C	RRC_A	00D2	RST55
RST75	01FF	RST_0	01D7	RST_1	01DF	RST_2
RST_4	01EB	RST_5	01EF	RST_6	01F3	RST_7
SBB_M	0082	SBB_R	0081	SBI	0086	SHLD
SINT	020B	SP	0009	SPHL	0012	STA
STC_F	00D7	SUB_M	007A	SUB_R	0001	SUB_S
TRAP	01FB	UDAT	0003	V0	0006	V1
V11	0024	V12	003C	V13	0034	V14
V3	0007	V4	0008	V5	0010	V6
V8	0028	V9	0030	WORD	0001	WZ
XCHG	0064	XOR_RS	0006	XRA_M	00BA	XRA_R
XTHL	01B9	ZA	0004	ZB	0003	ZERO

Tiempo de ensamblaje 46240 ms

```
0000 006EED6D40493377
0001 000EED6D434B37DD
0002 060EED6D40493377
0003 000EED6D434B37CC
0004 007EE96D40493377
0005 000EED05474B44FF
0006 000EED1D70493377
0007 0006C50D30493377
0008 0002ED05474B0CFF
0009 000EED05474A3444
000A 000EED15574B34FF
000B 0063FD0C50493377
000C 0073FD1D704B3477
000D 000EED05474A3444
000E 000EED15674B34FF
000F 000EED0D20493377
0010 0073FD1D794B3777
0011 0093FD0942493477
0012 0073FD1D774B3449
0013 000EED15674B0CFF
0014 000EED0D20493377
0015 0073FD1D794B3777
0016 000EED15674B0CFF
0017 000EED0D20493377
0018 0093FD0948493777
0019 000EED15674B0CFF
001A 000EED0D20493377
001B 000EED0D4B4B3771
001C 000EED15674B0CFF
001D 000EED0D20493377
001E 0073FD1D7B4B3770
001F 000EED15674B0CFF
0020 000EED0D20493377
0021 000EED0D4B4B3773
0022 000EED15674B0CFF
0023 000EED0D20493377
0024 0073FD1D7B4B3772
0025 000EED15674B0CFF
0026 000EED0D20493377
0027 000EED0D4B4B3775
0028 000EED15674B0CFF
0029 000EED0D20493377
002A 0073FD1D7B4B3774
002B 000EED15674B0CFF
002C 000EED0D20493377
002D 000EED0D4B4B3778
002E 000EED15674B0CFF
002F 000EED0D20493377
0030 0073FD1D7B4B3779
0031 000EED15674B0CFF
0032 000EED0D20493377
0033 000EED0D4B4B377B
0034 000EED15674B0CFF
0035 000EED0D20493377
```

```
0036 000EED0D4B4B377A
0037 000EED05474A34AA
0038 000EED15674B34FF
0039 000EED0D20493377
003A 0073FD1D7B4B3777
003B 000EED15674B0CFF
003C 000EED0D20493377
003D 000EED0D4B4B377B
003E 000EED15674B0CFF
003F 000EED0D20493377
0040 000EED0D4B4B377A
0041 000EED09434B3477
0042 000EED05474A34AA
0043 000EED15574B34FF
0044 0063FD0C50493377
0045 000EED15674B0CFF
0046 000EED0D20493377
0047 000EED0D4B4B377B
0048 000EED15674B0CFF
0049 000EED0D20493377
004A 000EED0D4B4B377A
004B 000EED05474A0CAA
004C 000EED15674B34AA
004D 000EED0D20493377
004E 000EED0D4B4B3755
004F 000EED15674B34FF
0050 000EED0D20493377
0051 0063FD0D4B4B3774
0052 000EED15674B0CFF
0053 000EED0D20493377
0054 000EED0D4B4B377B
0055 000EED15674B0CFF
0056 000EED0D20493377
0057 000EED0D4B4B377A
0058 000EED05474A0CAA
0059 000EED19534B3455
005A 000EED04574B34AA
005B 000EED19534B3444
005C 0063FD04574B34FF
005D 000EED05444B3477
005E 000EED15674B34FF
005F 000EED0D20493377
0060 0073FD1D7A4B3777
0061 000EED05444B3477
0062 000EED19534B3477
0063 0063FD04574B34FF
0064 000EED0D474B344A
0065 000EED0D474B3424
0066 000EED0D474A3725
0067 000EED0D474B34A2
0068 0073FD1D774A3723
0069 0073FD1D720B0177
006A 000EED05474A3444
006B 000EED15674B34FF
```

006C 000EED0D20493377
006D 0073FD1D7B0B0577
006E 000EED15674B0CFF
006F 000EED0D20493377
0070 0073FD1D7B0B0577
0071 0073FD1D720B8177
0072 000EED05474A3444
0073 000EED15674B34FF
0074 000EED0D20493377
0075 0073FD1D7B0B8577
0076 000EED15674B0CFF
0077 000EED0D20493377
0078 0073FD1D7B0B8577
0079 0073FD1D720B1977
007A 000EED05474A3444
007B 000EED15674B34FF
007C 000EED0D20493377
007D 0073FD1D7B0B1D77
007E 000EED15674B0CFF
007F 000EED0D20493377
0080 0073FD1D7B0B1D77
0081 0073FD1D720B9177
0082 000EED05474A3444
0083 000EED15674B34FF
0084 000EED0D20493377
0085 0073FD1D7B0B9577
0086 000EED15674B0CFF
0087 000EED0D20493377
0088 0073FD1D7B0B9577
0089 0073FD1D711B0B77
008A 000EED05474A3444
008B 000EED15674B34FF
008C 000EED0D20493377
008D 0093FD094B1B0F77
008E 0073FD1D711B1377
008F 000EED05474A3444
0090 000EED15674B34FF
0091 000EED0D20493377
0092 0093FD094B1B2777
0093 000EED0D474B0C00
0094 0073FD1D774A3701
0095 000EED0D474B0C22
0096 0073FD1D774A3723
0097 000EED0D474B0C44
0098 0073FD1D774A3745
0099 000EED0D474B0C99
009A 0073FD1D774A3798
009B 000EED0D474B1400
009C 0073FD1D774A3701
009D 000EED0D474B1422
009E 0073FD1D774A3723
009F 000EED0D474B1444
00A0 0073FD1D774A3745
00A1 000EED0D474B1499

00A2 0073FD1D774A3798
00A3 000EED0D473B0104
00A4 0073FD1D774A3745
00A5 000EED0D473B0124
00A6 0073FD1D774A3745
00A7 000EED0D473B0144
00A8 0073FD1D774A3745
00A9 000EED0D473B0194
00AA 0073FD1D774A3745
00AB 000EED0D434A3477
00AC 0AF38D0D434A3477
00AD 0B03AD0D40493377
00AE 0073FD1D70493377
00AF 0AD3FD0D430B01D7
00B0 0073FD1D730B01C7
00B1 0073FD1D722B4177
00B2 000EED05474A3444
00B3 000EED15674B34FF
00B4 000EED0D20493377
00B5 0073FD1D7B2B4577
00B6 000EED15674B0CFF
00B7 000EED0D20493377
00B8 0073FD1D7B2B4577
00B9 0073FD1D722B6177
00BA 000EED05474A3444
00BB 000EED15674B34FF
00BC 000EED0D20493377
00BD 0073FD1D7B2B6577
00BE 000EED15674B0CFF
00BF 000EED0D20493377
00C0 0073FD1D7B2B6577
00C1 0073FD1D722B3177
00C2 000EED05474A3444
00C3 000EED15674B34FF
00C4 000EED0D20493377
00C5 0073FD1D7B2B3577
00C6 000EED15674B0CFF
00C7 000EED0D20493377
00C8 0073FD1D7B2B3577
00C9 0073FD1D72091977
00CA 000EED05474A3444
00CB 000EED15674B34FF
00CC 000EED0D20493377
00CD 0073FD1D7B091D77
00CE 000EED15674B0CFF
00CF 000EED0D20493377
00D0 0073FD1D7B091D77
00D1 0073FD1D73573377
00D2 0073FD1D73553377
00D3 0073FD1D735F3377
00D4 0073FD1D735D3377
00D5 0073FD1D734B7377
00D6 0073FD1D70693377
00D7 001EED6D40493377

00D8 000EED6D434B37AA
00D9 0073FD1D735D33AA
00DA 000EED15674B0CFF
00DB 000EED0D20493377
00DC 000EED0D4B4B377B
00DD 000EED15674B0CFF
00DE 000EED0D20493377
00DF 000EED0D4B4B377A
00E0 0063FD05474B34AF
00E1 0DA33D0D40493377
00E2 000EED15674B0CFF
00E3 000EED0D20493377
00E4 000EED0D4B4B377B
00E5 000EED15674B0CFF
00E6 000EED0D20493377
00E7 000EED0D4B4B377A
00E8 0073FD1D70493377
00E9 0DA32D0D404B3177
00EA 000EED15674B0CFF
00EB 000EED0D20493377
00EC 000EED0D4B4B377B
00ED 000EED15674B0CFF
00EE 000EED0D20493377
00EF 000EED0D4B4B377A
00F0 0073FD1D70493377
00F1 0DA37D0D40493377
00F2 000EED15674B0CFF
00F3 000EED0D20493377
00F4 000EED0D4B4B377B
00F5 000EED15674B0CFF
00F6 000EED0D20493377
00F7 000EED0D4B4B377A
00F8 0073FD1D70493377
00F9 0DA36D0D40493377
00FA 000EED15674B0CFF
00FB 000EED0D20493377
00FC 000EED0D4B4B377B
00FD 000EED15674B0CFF
00FE 000EED0D20493377
00FF 000EED0D4B4B377A
0100 0073FD1D70493377
0101 0DA35D0D40493377
0102 000EED15674B0CFF
0103 000EED0D20493377
0104 000EED0D4B4B377B
0105 000EED15674B0CFF
0106 000EED0D20493377
0107 000EED0D4B4B377A
0108 0073FD1D70493377
0109 0DA34D0D40493377
010A 000EED15674B0CFF
010B 000EED0D20493377
010C 000EED0D4B4B377B
010D 000EED15674B0CFF

```
010E 000EED0D20493377
010F 000EED0D4B4B377A
0110 0073FD1D70493377
0111 0DA31D0D40493377
0112 000EED15674B0CFF
0113 000EED0D20493377
0114 000EED0D4B4B377B
0115 000EED15674B0CFF
0116 000EED0D20493377
0117 000EED0D4B4B377A
0118 0073FD1D70493377
0119 0DA30D0D40493377
011A 000EED15674B0CFF
011B 000EED0D20493377
011C 000EED0D4B4B377B
011D 000EED15674B0CFF
011E 000EED0D20493377
011F 000EED0D4B4B377A
0120 0073FD1D70493377
0121 000EED0D474B1499
0122 000EED15674B0CFF
0123 000EED0D20493377
0124 000EED0D4B4B377B
0125 000EED15674B0CFF
0126 000EED0D20493377
0127 000EED0D4B4B377A
0128 000EED05474A1499
0129 000EED19534B34FF
012A 000EED04574B3499
012B 000EED1953CB34EE
012C 0063FD04574B34AF
012D 12133D0D40493377
012E 000EED15674B0CFF
012F 000EED0D20493377
0130 000EED0D4B4B377B
0131 000EED15674B0CFF
0132 000EED0D20493377
0133 000EED0D4B4B377A
0134 0073FD1D70493377
0135 12132D0D40493377
0136 000EED15674B0CFF
0137 000EED0D20493377
0138 000EED0D4B4B377B
0139 000EED15674B0CFF
013A 000EED0D20493377
013B 000EED0D4B4B377A
013C 0073FD1D70493377
013D 12137D0D40493377
013E 000EED15674B0CFF
013F 000EED0D20493377
0140 000EED0D4B4B377B
0141 000EED15674B0CFF
0142 000EED0D20493377
0143 000EED0D4B4B377A
```

```
0144 0073FD1D70493377
0145 12136D0D40493377
0146 000EED15674B0CFF
0147 000EED0D20493377
0148 000EED0D4B4B377B
0149 000EED15674B0CFF
014A 000EED0D20493377
014B 000EED0D4B4B377A
014C 0073FD1D70493377
014D 12135D0D40493377
014E 000EED15674B0CFF
014F 000EED0D20493377
0150 000EED0D4B4B377B
0151 000EED15674B0CFF
0152 000EED0D20493377
0153 000EED0D4B4B377A
0154 0073FD1D70493377
0155 12134D0D40493377
0156 000EED15674B0CFF
0157 000EED0D20493377
0158 000EED0D4B4B377B
0159 000EED15674B0CFF
015A 000EED0D20493377
015B 000EED0D4B4B377A
015C 0073FD1D70493377
015D 12131D0D40493377
015E 000EED15674B0CFF
015F 000EED0D20493377
0160 000EED0D4B4B377B
0161 000EED15674B0CFF
0162 000EED0D20493377
0163 000EED0D4B4B377A
0164 0073FD1D70493377
0165 12130D0D40493377
0166 000EED15674B0CFF
0167 000EED0D20493377
0168 000EED0D4B4B377B
0169 000EED15674B0CFF
016A 000EED0D20493377
016B 000EED0D4B4B377A
016C 0073FD1D70493377
016D 000EED05474A0C99
016E 000EED15674A0C99
016F 000EED0D20493377
0170 000EED1D6B4B377E
0171 000EED0D20493377
0172 000EED0D4B4B377F
0173 0063FD05474B34FF
0174 16D33D0D40493377
0175 0073FD1D70493377
0176 16D32D0D40493377
0177 0073FD1D70493377
0178 16D37D0D40493377
0179 0073FD1D70493377
```

017A 16D36D0D40493377
017B 0073FD1D70493377
017C 16D35D0D40493377
017D 0073FD1D70493377
017E 16D34D0D40493377
017F 0073FD1D70493377
0180 16D31D0D40493377
0181 0073FD1D70493377
0182 16D30D0D40493377
0183 0073FD1D70493377
0184 0063FD05474B344F
0185 000EED05474B1499
0186 000EED15574B1499
0187 000EED09434A3400
0188 000EED0C50493377
0189 000EED19534A3411
018A 0063FD04574B34FF
018B 000EED05474B1499
018C 000EED15574B1499
018D 000EED09434A3422
018E 000EED0C50493377
018F 000EED19534A3433
0190 0063FD04574B34FF
0191 000EED05474B1499
0192 000EED15574B1499
0193 000EED09434A3444
0194 000EED0C50493377
0195 000EED19534A3455
0196 0063FD04574B34FF
0197 000EED05474B1499
0198 000EED15574B1499
0199 000EED09434A3477
019A 000EED0C50493377
019B 000EED3950493377
019C 0063FD04574B34FF
019D 000EED05474A0C99
019E 000EED15674B3499
019F 000EED0D20493377
01A0 000EED0D4B4B3711
01A1 000EED1D674B0C99
01A2 000EED05274B34FF
01A3 0073FD1D7B4B3700
01A4 000EED05474A0C99
01A5 000EED15674B3499
01A6 000EED0D20493377
01A7 000EED0D4B4B3733
01A8 000EED1D674B0C99
01A9 000EED05274B34FF
01AA 0073FD1D784B3722
01AB 000EED05474A0C99
01AC 000EED15674B3499
01AD 000EED0D20493377
01AE 000EED0D4B4B3755
01AF 000EED1D674B0C99

01B0 000EED05274B34FF
01B1 0073FD1D7B4B3744
01B2 000EED05474A0C99
01B3 000EED15674B3499
01B4 000EED0D20493377
01B5 000EED0D4B793777
01B6 000EED1D674A0C99
01B7 000EED05234B34FF
01B8 0073FD1D7B4B3777
01B9 000EED05474A0C99
01BA 000EED15674B3499
01BB 000EED0D20493377
01BC 000EED1D6B4B37BB
01BD 000EED09234B3444
01BE 000EED1D5B4B37AA
01BF 000EED04574B1499
01C0 000EED19534B3455
01C1 000EED0C574B34A4
01C2 000EED0D474B34B5
01C3 0063FD05474B34FF
01C4 000EED15674B0CFF
01C5 000EED052F4B37AA
01C6 000EED054F4B37BB
01C7 000EED15674B34FF
01C8 0063FD0DAB4B3777
01C9 000EED15674B0CFF
01CA 000EED052F4B37AA
01CB 000EED054F4B37BB
01CC 000EED19534B3477
01CD 0063FD04D74B34FF
01CE 0073FF1D70493377
01CF 0073FD1D70493377
01D0 000EED1D40493377
01D1 0006CD0F40493377
01D2 1D13FD0D40493377
01D3 0073FD1D70493377
01D4 000EED4D40493377
01D5 0073FD1D734B3777
01D6 0073F91D734A3477
01D7 000EED0D434B44AA
01D8 000EED0D434B44BB
01D9 000EED05474B1499
01DA 000EED09434B34FF
01DB 000EED15574B1499
01DC 000EED0C50493377
01DD 000EED1953CB34EE
01DE 0063FD04574B34AF
01DF 000EED0D434B44AA
01E0 008EED6D40493377
01E1 000EED6D434B37BB
01E2 1DA3FD05474B1499
01E3 000EED0D434B44AA
01E4 010EED6D40493377
01E5 000EED6D434B37BB

```
01E6 1DA3FD05474B1499
01E7 000EED0D434B44AA
01E8 018EED6D40493377
01E9 000EED6D434B37BB
01EA 1DA3FD05474B1499
01EB 000EED0D434B44AA
01EC 020EED6D40493377
01ED 000EED6D434B37BB
01EE 1DA3FD05474B1499
01EF 000EED0D434B44AA
01F0 028EED6D40493377
01F1 000EED6D434B37BB
01F2 1DA3FD05474B1499
01F3 000EED0D434B44AA
01F4 030EED6D40493377
01F5 000EED6D434B37BB
01F6 1DA3FD05474B1499
01F7 000EED0D434B44AA
01F8 038EED6D40493377
01F9 000EED6D434B37BB
01FA 1DA3FD05474B1499
01FB 000EED0D434B44AA
01FC 024EED0D40493377
01FD 000EED6D434B37BB
01FE 1DA3FD05474B1499
01FF 000EED0D434B44AA
0200 03CEED6D40493377
0201 000EED6D434B37BB
0202 1DA3FD05474B1499
0203 000EED0D434B44AA
0204 034EED6D40493377
0205 000EED6D434B37BB
0206 1DA3FD05474B1499
0207 000EED0D434B44AA
0208 02CEED6D40493377
0209 000EED6D434B37BB
020A 1DA3FD05474B1499
020B 000EE40D40493377
020C 0002ED0D404B3177
020D 0006CD8F40493377
020E 20D3FD0D40493377
```

IV. SIMULACION DE CPA

4.1 PROGRAMA DE SIMULACION

```
MACHINE CPA;
CONSTANT

(...selección del operando fuente...)
     ab =1;
     zb =3;
     za =4;
     da =5;
     dz =7;

(...operaciones de la alu...)
     add       =0;
     sub_r     =1;
     sub_s     =2;
     or_rs     =3;
     and_rs    =4;
     and_nors  =5;
     exor_rs   =6;
     exnor_rs  =7;

(...selección de destino...)
     noper =1;
     ram_a =2;
     ram_f =3;
     ram_d =5;
     ram_u =7;

(...control de la próxima dirección...)
     jz    =0;
     cjs   =1;
     jmap  =2;
     cjp   =3;
     push_ =4;
     cjv   =6;
     crtn  =10;
     cont  =14;

(...tamaño datos...)
     byte  =0;
     word  =1;

(...control de salida al bus interno...)
     bus_y =0;
     foe   =1;
     udat =3;
     rd_mask =2;
```

```
(...multiplexor de condición...)
     neg_    =0;
     pos     =1;
     zero_   =2;
     nozero  =3;
     par     =4;
     impar   =5;
     cy      =6;
     nocy    =7;
     daal    =8;
     daah    =10;
     intr    =12;
     nopass  =14;
     pass    =15;

(...dato de entrada a la alu...)
     in      =0;
     ex      =1;

(...selección de los flags...)
     arit    =0;
     indc    =1;
     lop     =2;
     fdad    =3;
     inva    =4;
     rota    =5;
     cmcy    =6;
     fbus    =7;

(...fuente para direccionar los registros...)
     irir    =0;
     irpl    =1;
     plir    =2;
     plpl    =3;

(...selección de la alu...)
     left    =0;
     right   =1;
```

```
(*************************************************************************)
(       DECLARACION DE LOS ELEMENTOS DE LA ARQUITECTURA                  )
(*************************************************************************)

(. . . . . declaración de los registros . . . . . .)

REGISTER
     reg_inst<8>:=0;
     micro_pc<12>:=0;
     adr_mem<16>:=0;
     adr_mapping<8>:=0;
     status_reg<8>:=0;
     input_data<8>;
     mask_reg<8>:=0;
     data_reg<8>:=0;
     adress_reg<16>:=0;
     uno<1>:=1;

LATCH
     parity_reg<1>;
     d_latch<16>;
     lat_d<8>;
     lat_i<8>;


(. . . . . declaración de buses . . . . . .)

BUS
     salida_alu<17>;
     salida_alui<9>;
     salida_alud<9>;
     a_latch<16>;
     b_latch<16>;
     di_latch<8>;
     dd_latch<8>;
     alu_r<16>;
     alu_s<16>;
     y_bus<17>;
     alu_ir<8>;
     alu_is<8>;
     alu_dr<8>;
     alu_ds<8>;
     bus_i<9>;
     bus_d<9>;
```

```
      ader<4>;
      bder<4>;
      aiz<4>;
      biz<4>;
      par_test<4>;
      temp<17>;
      tempi<9>;
      tempd<9>;
      temp1<1>;
      temp2<1>;
      temp3<1>;
      cn1<1>;
      cn2<1>;
      cn<1>;
      cc<1>;
      ac<5>;

(. . . . . memorias . . . . . .)

MEMORY
      micromem[580]<64>:micro_pc;

      mapping[256]<12>:adr_mapping;0:
      1D3H,019H,061H,093H,089H,08EH,013H,0D1H,0000,0A3H,
      05DH,09BH,089H,08EH,013H,0D2H,0000,01FH,061H,095H,
      089H,08EH,013H,0D3H,0000,0A5H,05DH,09DH,089H,08EH,
      013H,0D4H,1D4H,025H,052H,097H,089H,08EH,013H,0ABH,
      0000,0A7H,045H,09FH,089H,08EH,013H,0D5H,1D6H,02BH,
      03BH,099H,08AH,08FH,016H,0D7H,0000,0A9H,031H,0A1H,
      089H,08EH,013H,0D6H,00CH,00CH,00CH,00CH,00CH,00CH,
      00DH,00CH,00CH,00CH,00CH,00CH,00CH,00CH,00DH,00CH,
      00CH,00CH,00CH,00CH,00CH,00CH,00DH,00CH,00CH,00CH,
      00CH,00CH,00CH,00CH,00DH,00CH,00CH,00CH,00CH,00CH,
      00CH,00CH,00DH,00CH,00CH,00CH,00CH,00CH,00CH,00CH,
      00DH,00CH,011H,011H,011H,011H,011H,011H,1D0H,011H,
      00CH,00CH,00CH,00CH,00CH,00CH,00DH,00CH,069H,069H,
      069H,069H,069H,069H,06AH,069H,071H,071H,071H,071H,
      071H,071H,072H,071H,079H,079H,079H,079H,079H,079H,
      07AH,079H,081H,081H,081H,081H,081H,081H,082H,081H,
      081H,081H,081H,081H,081H,081H,082H,081H,089H,089H,
      089H,089H,089H,089H,08AH,089H,0C1H,0C1H,0C1H,0C1H,
      0C1H,0C1H,0C2H,0C1H,0C9H,0C9H,0C9H,0C9H,0C9H,0C9H,
      0CAH,0C9H,174H,19DH,0E1H,0DAH,12DH,185H,06EH,1D7H,
      176H,16DH,0E9H,0000,135H,121H,076H,1DFH,178H,1A4H,
      0F1H,1C9H,13DH,18BH,07EH,1E3H,17AH,0000,0F9H,1C4H,
      145H,0000,086H,1E7H,17CH,1ABH,101H,1B9H,14DH,191H,
      0B6H,1EBH,17EH,184H,109H,064H,155H,0000,0BEH,1EFH,
      180H,1B2H,111H,1CFH,15DH,197H,0C6H,1F3H,182H,012H,
      119H,1CEH,165H,0000,0CEH,1F7H;
```

```
      mem_prog[65535]<8>:adr_mem;

      aimem[017]<8>:aiz;
      admem[017]<8>:ader;
      bimem[017]<8>:biz;
      bdmem[017]<8>:bder;


STACK
      microstack[20H]<12>;


(. . . . . definición de campos . . . . . .)

FIELD
      b_alu     =<0..3>;
      a_alu     =<4..7>;
      source    =<8..10>;
      cy_in     =<11>;
      alu       =<12..14>;
      cy_sel    =<15>;
      destino   =<16..18>;
      shifter   =<19>;
      flags     =<20..22>;
      leri      =<23>;
      add_sel   =<24..25>;
      w_dat     =<26>;
      dat_in    =<27>;
      status    =<28..29>;
      rd        =<30>;
      io_m      =<31>;
      wr        =<32>;
      ts        =<33>;
      mbre      =<34>;
      mare      =<35>;
      ale       =<36>;
      b_ctrl    =<37..38>;
      hlda      =<39>;
      inta      =<40>;
      edi       =<41>;
      st_mask   =<42>;
      ir_en     =<43>;
      cc_mux    =<44..47>;
      secuence  =<48..51>;
      next_addr=<52..63>;
```

```
(. . . . . microoperaciones . . . . . .)

MICRO_OP


(.........tratamiento para los datos de 16 bits.........)

IF w_dat IS
   word: BEGIN
          IF add_sel IS
                     irir: BEGIN
                               aiz<0>:= ZERO;
                               aiz<3>:= ZERO;
                               aiz<1..2>:= reg_inst<4..5>;
                               ader:= aiz;
                               biz:= aiz;
                               bder:= aiz;
                           END;
                     irpl: BEGIN
                               aiz:= a_alu;
                               ader:= a_alu;
                               biz<0>:= ZERO;
                               biz<3>:= ZERO;
                               biz<1..2>:= reg_inst<4..5>;
                               bder:= biz;
                           END;
                     plir: BEGIN
                               aiz<0>:= ZERO;
                               aiz<3>:= ZERO;
                               aiz<1..2>:= reg_inst<4..5>;
                               ader:= aiz;
                               biz:= b_alu;
                               bder:= b_alu;
                           END;
                     plpl: BEGIN
                               aiz:= a_alu;
                               ader:= a_alu;
                               biz:= b_alu;
                               bder:= b_alu;
                           END;
          END_IF;

          a_latch<0..7>:= admem;
          a_latch<8..15>:= aimem;
          b_latch<0..7>:= bdmem;
          b_latch<8..15>:= bimem;
```

```
(....selección de los operandos de la alu....)
          IF source IS
                  ab: BEGIN
                          alu_r:= a_latch;
                          alu_s:= b_latch;
                      END;
                  zb: BEGIN
                          alu_r:= ZERO;
                          alu_s:= b_latch;
                      END;
                  za: BEGIN
                          alu_r:= ZERO;
                          alu_s:= a_latch;
                      END;
                  da: BEGIN
                          alu_r:= d_latch;
                          alu_s:= a_latch;
                      END;
                  dz: BEGIN
                          alu_r:= d_latch;
                          alu_s:= ZERO;
                      END;
          END_IF;

(....selección del carry....)
          IF cy_sel IS
                     1: cn:= status_reg<0>;
                     0: cn:= cy_in;
          END_IF;

(....función de la alu....)
          IF alu IS
                  add: BEGIN
                           temp:= alu_r + alu_s;
                           IF cn IS
                                  0: salida_alu:= temp;
                                  1: salida_alu:= INC temp;
                           END_IF;
                       END;

                  sub_r: BEGIN
                             temp:= alu_s - alu_r;
                             IF cn IS
                                    0: salida_alu:= DEC temp;
                                    1: salida_alu:= temp;
                             END_IF;
                         END;
```

```
                sub_s: BEGIN
                           temp:= alu_r - alu_s;
                           IF cn IS
                                  0: salida_alu:= DEC temp;
                                  1: salida_alu:= temp;
                           END_IF;
                       END;

                or_rs: salida_alu:= alu_r OR alu_s;

                and_rs: salida_alu:= alu_r AND alu_s;

                and_nors: salida_alu:= alu_s AND NOT alu_r;

                exor_rs: salida_alu:= alu_r XOR alu_s;

                exnor_rs: BEGIN
                              temp:= alu_r XOR alu_s;
                              salida_alu:= NOT temp;
                          END;
          END_IF;

(....control del bus interno y destino de la operación....)
          IF b_ctrl IS
               bus_y: BEGIN
                        IF destino IS
                            noper: y_bus:= salida_alu;
                            ram_a: BEGIN
                                       ader:= bder;
                                       aiz:= biz;
                                       bdmem:= salida_alu<0..7>;
                                       admem:= salida_alu<0..7>;
                                       bimem:= salida_alu<8..15>;
                                       aimem:= salida_alu<8..15>;
                                       y_bus<0..15>:= a_latch;
                                   END;
                             ram_f: BEGIN
                                       ader:= bder;
                                       aiz:= biz;
                                       bdmem:= salida_alu<0..7>;
                                       admem:= salida_alu<0..7>;
                                       bimem:= salida_alu<8..15>;
                                       aimem:= salida_alu<8..15>;
                                       y_bus:= salida_alu;
                                   END;
                          END_IF;
                      END;
               foe: y_bus<0..7>:= status_reg;
          END_IF;
```

```
        IF st_mask=0: mask_reg:= y_bus<0..7>;

        IF dat_in IS
              in: BEGIN
                     d_latch<0..7>:= y_bus<8..15>;
                     d_latch<8..15>:= y_bus<0..7>;
                  END;
        END_IF;

        IF mare=0: adress_reg:= y_bus<0..15>;

   END;

(.......tratamiento para los datos de 8 bits........)

   byte: BEGIN
        IF add_sel IS
           irir: BEGIN
                     aiz<0..2>:= reg_inst<0..2>;
                     aiz<3>:= ZERO;
                     ader<0>:= NOT aiz<0>;
                     ader<1..3>:= aiz<1..3>;
                     biz<0..2>:= reg_inst<3..5>;
                     biz<3>:= ZERO;
                     bder<0>:= NOT biz<0>;
                     bder<1..3>:= biz<1..3>;
                 END;
           irpl: BEGIN
                     aiz:= a_alu;
                     ader<0>:= NOT aiz<0>;
                     ader<1..3>:= aiz<1..3>;
                     biz<0..2>:= reg_inst<3..5>;
                     biz<3>:= ZERO;
                     bder<0>:= NOT biz<0>;
                     bder<1..3>:= biz<1..3>;
                 END;
           plir: BEGIN
                     aiz<0..2>:= reg_inst<0..2>;
                     aiz<3>:= ZERO;
                     ader<0>:= NOT aiz<0>;
                     ader<1..3>:= aiz<1..3>;
                     biz:= b_alu;
                     bder<0>:= NOT biz<0>;
                     bder<1..3>:= biz<1..3>;
                 END;
```

```
            plpl: BEGIN
                      aiz:= a_alu;
                      ader<0>:= NOT aiz<0>;
                      ader<1..3>:= aiz<1..3>;
                      biz:= b_alu;
                      bder<0>:= NOT biz<0>;
                      bder<1..3>:= biz<1..3>;
                  END;
        END_IF;

        IF b_ctrl IS
                udat: BEGIN
                          bus_i:= next_addr;
                          bus_d:= next_addr;
                      END;
                rd_mask: BEGIN
                             bus_i:= mask_reg;
                             bus_d:= mask_reg;
                         END;
        END_IF;
        IF dat_in IS
                ex: BEGIN
                        di_latch:= input_data;
                        dd_latch:= input_data;
                    END;
                in: BEGIN
                        di_latch:= lat_i;
                        dd_latch:= lat_d;
                    END;
        END_IF;

(....fuente de datos de la alu....)
          IF source IS
              ab: BEGIN
                      alu_ir:= aimem;
                      alu_is:= bimem;
                      alu_dr:= admem;
                      alu_ds:= bdmem;
                  END;
              zb: BEGIN
                      alu_ir:= ZERO;
                      alu_is:= bimem;
                      alu_dr:= ZERO;
                      alu_ds:= bdmem;
                  END;
```

```
            za: BEGIN
                    alu_ir:= ZERO;
                    alu_is:= aimem;
                    alu_dr:= ZERO;
                    alu_ds:= admem;
                END;
            da: BEGIN
                    alu_ir:= di_latch;
                    alu_is:= aimem;
                    alu_dr:= dd_latch;
                    alu_ds:= admem;
                END;
            dz: BEGIN
                    alu_ir:= di_latch;
                    alu_is:= ZERO;
                    alu_dr:= dd_latch;
                    alu_ds:= ZERO;
                END;
        END_IF;

(....selección del carry....)
        IF cy_sel IS
                1: BEGIN
                       cn1:= status_reg<0>;
                       cn2:= status_reg<0>;
                   END;
                0: BEGIN
                         cn1:= cy_in;
                         cn2:= cy_in;
                   END;
        END_IF;

(....operación de la alu....)
        IF alu IS
            add: BEGIN
                    tempi:= alu_ir + alu_is;
                    tempd:= alu_dr + alu_ds;
                    ac:= alu_ir<0..3> + alu_is<0..3>;
                    IF cn1 IS
                            0: salida_alui:= tempi;
                            1: BEGIN
                                salida_alui:= INC tempi;
                                ac:= INC ac;
                               END;
                    END_IF;
```

```
          IF cn2 IS
                  0: salida_alud:= tempd;
                  1: salida_alud:= INC tempd;
          END_IF;
      END;
sub_r: BEGIN
            tempi:= alu_is - alu_ir;
            tempd:= alu_ds - alu_dr;
            ac:= alu_is<0..3> - alu_ir<0..3>;
          IF cn1 IS
                  0: BEGIN
                      salida_alui:= DEC tempi;
                      ac:= DEC ac;
                     END;
                  1: salida_alui:= tempi;
          END_IF;
          IF cn2 IS
                  0: salida_alud:= DEC tempd;
                  1: salida_alud:= tempd;
          END_IF;
        END;
sub_s: BEGIN
            tempi:= alu_ir - alu_is;
            tempd:= alu_dr - alu_ds;
            ac:= alu_ir<0..3> - alu_is<0..3>;
          IF cn1 IS
                  0: BEGIN
                      salida_alui:= DEC tempi;
                      ac:= DEC ac;
                     END;
                  1: salida_alui:= tempi;
          END_IF;
          IF cn2 IS
                  0: salida_alud:= DEC tempd;
                  1: salida_alud:= tempd;
          END_IF;
        END;
or_rs: BEGIN
            salida_alui:= alu_ir OR alu_is;
            salida_alud:= alu_dr OR alu_ds;
        END;
and_rs: BEGIN
             salida_alui:= alu_ir AND alu_is;
             salida_alud:= alu_dr AND alu_ds;
         END;
```

```
            and_nors: BEGIN
                          salida_alui:= alu_ir AND NOT alu_is;
                          salida_alud:= alu_dr AND NOT alu_ds;
                      END;
            exor_rs: BEGIN
                         salida_alui:= alu_ir XOR alu_is;
                         salida_alud:= alu_dr XOR alu_ds;
                      END;
            exnor_rs: BEGIN
                          tempi:= alu_ir XOR alu_is;
                          tempd:= alu_dr XOR alu_ds;
                          salida_alui:= NOT tempi;
                          salida_alud:= NOT tempd;
                      END;
          END_IF;

(....control del bus y destino de la operación....)
          IF b_ctrl IS
             bus_y: IF destino IS
                        noper: BEGIN
                                   bus_i:= salida_alui;
                                   bus_d:= salida_alud;
                               END;
                        ram_a: BEGIN
                                   ader:= bder;
                                   aiz:= biz;
                                   bus_i<0..7>:= aimem;
                                   bimem:= salida_alui<0..7>;
                                   aimem:= salida_alui<0..7>;
                                   bus_d<0..7>:= admem;
                                   bdmem:= salida_alud<0..7>;
                                   admem:= salida_alud<0..7>;
                               END;
                        ram_f: BEGIN
                                   ader:= bder;
                                   aiz:= biz;
                                   bimem:= salida_alui<0..7>;
                                   aimem:= salida_alui<0..7>;
                                   bdmem:= salida_alud<0..7>;
                                   admem:= salida_alud<0..7>;
                                   bus_i:= salida_alui;
                                   bus_d:= salida_alud;
                               END;
```

```
ram_d: BEGIN
           bus_i:= salida_alui;
           bus_d:= salida_alud;
           ader:= bder;
           aiz:= biz;
        IF shifter IS
            0: BEGIN
                salida_alui:= RR salida_alui;
                salida_alui<7>:=salida_alui<8>;
                bimem:=salida_alui<0..7>;
                aimem:=salida_alui<0..7>;
                salida_alud:= RR salida_alud;
                salida_alud<7>:=salida_alud<8>;
                bdmem:=salida_alud<0..7>;
                admem:=salida_alud<0..7>;
               END;
            1: BEGIN
                salida_alui:= RR salida_alui;
                salida_alui<7>:= status_reg<0>;
                bimem:=salida_alui<0..7>;
                aimem:=salida_alui<0..7>;
                salida_alud:= RR salida_alud;
                salida_alud<7>:= status_reg<0>;
                bdmem:=salida_alud<0..7>;
                admem:=salida_alud<0..7>;
               END;
        END_IF;
       END;
ram_u: BEGIN
           bus_i:= salida_alui;
           bus_d:= salida_alud;
           ader:= bder;
           aiz:= biz;
        IF shifter IS
            0: BEGIN
                salida_alui:= LSL salida_alui;
                salida_alui<0>:=salida_alui<8>;
                bimem:=salida_alui<0..7>;
                aimem:=salida_alui<0..7>;
                salida_alud:= LSL salida_alud;
                salida_alud<0>:= salida_alud<8>;
                bdmem:=salida_alud<0..7>;
                admem:=salida_alud<0..7>;
               END;
```

```
                                1: BEGIN
                                    salida_alui:= LSL salida_alui;
                                    salida_alui<0>:= status_reg<0>;
                                    bimem:=salida_alui<0..7>;
                                    aimem:=salida_alui<0..7>;
                                    salida_alud:= LSL salida_alud;
                                    salida_alud<0>:= status_reg<0>;
                                    bdmem:=salida_alud<0..7>;
                                    admem:=salida_alud<0..7>;
                                   END;
                               END_IF;
                           END;
        END_IF;
      foe: BEGIN
              bus_i<0..7>:= status_reg;
              bus_d<0..7>:= status_reg;
           END;
      udat: BEGIN
               ader:= bder;
               aiz:= biz;
               bimem:= salida_alui<0..7>;
               aimem:= salida_alui<0..7>;
               bdmem:= salida_alud<0..7>;
               admem:= salida_alud<0..7>;
            END;

      rd_mask: BEGIN
                  ader:= bder;
                  aiz:= biz;
                   bimem:= salida_alui<0..7>;
                  aimem:= salida_alui<0..7>;
                  bdmem:= salida_alud<0..7>;
                  admem:= salida_alud<0..7>;
               END;
END_IF;

IF st_mask=0: mask_reg:= bus_i;

IF mbre=0: IF leri IS
                left:  data_reg:=bus_i<0..7>;
                right: data_reg:=bus_d<0..7>;
           END_IF;
lat_i:= bus_d;
lat_d:= bus_i;
```

```
(....cálculo de la paridad....)
          par_test:= ZERO;

          IF salida_alui<0>=1: par_test:= INC par_test;
          IF salida_alui<1>=1: par_test:= INC par_test;
          IF salida_alui<2>=1: par_test:= INC par_test;
          IF salida_alui<3>=1: par_test:= INC par_test;
          IF salida_alui<4>=1: par_test:= INC par_test;
          IF salida_alui<5>=1: par_test:= INC par_test;
          IF salida_alui<6>=1: par_test:= INC par_test;
          IF salida_alui<7>=1: par_test:= INC par_test;

          IF par_test<0>=0: parity_reg:= uno;
          IF par_test<0>=1: parity_reg:= ZERO;
     END;
  END_IF;

(....carga del estado de las operaciones....)

  IF flags IS
      arit: BEGIN
                status_reg<0>:= salida_alui<8>;
                status_reg<2>:= parity_reg;
                status_reg<4>:= ac<4>;
                IF salida_alui=0: status_reg<6>:=uno;
                status_reg<7>:= salida_alui<7>;
            END;
      indc: BEGIN
                status_reg<2>:= parity_reg;
                status_reg<4>:= ac<4>;
                IF salida_alui=0: status_reg<6>:=uno;
                status_reg<7>:= salida_alui<7>;
            END;
      lop: BEGIN
                status_reg<0>:= ZERO;
                status_reg<2>:= parity_reg;
                status_reg<4>:= ZERO;
                IF salida_alui=0: status_reg<6>:=uno;
                status_reg<7>:= salida_alui<7>;
            END;
      fdad: status_reg<0>:= salida_alu<16>;
      cmcy: status_reg<0>:= NOT status_reg<0>;
      fbus: status_reg:= bus_i<0..7>;
      rota: status_reg<0>:= salida_alui<8>;
  END_IF;
```

```
(........búsqueda del opcode y dirección del operando.........)

  IF ir_en=0: BEGIN
                  reg_inst:= mem_prog;
                  adr_mapping:= mem_prog;
              END;

  IF wr=0: mem_prog:= data_reg;

  IF rd=0: input_data:= mem_prog;

  IF ale=1: adr_mem:= adress_reg;

(.........estado del multiplexor de condición.........)

  IF cc_mux IS
          neg_:  cc:= NOT status_reg<7>;
          pos:  cc:= status_reg<7>;
          zero_:  cc:= NOT status_reg<6>;
          nozero:  cc:= status_reg<6>;
          par:  cc:= NOT status_reg<2>;
          impar:  cc:= status_reg<2>;
          cy:  cc:= NOT status_reg<0>;
          nocy:  cc:= status_reg<0>;
          pass:  cc:= ZERO;
          nopass:  cc:= uno;
          daal: BEGIN
                   temp1:= salida_alui<2> OR salida_alui<1>;
                   temp2:= salida_alui<3> AND temp1;
                   temp3:= status_reg<4> OR temp2;
                   IF temp3 IS
                          1: cc:=ZERO;
                          0: cc:= uno;
                   END_IF;
                END;
          daah: BEGIN
                   temp1:= salida_alui<6> OR salida_alui<5>;
                   temp2:= salida_alui<7> AND temp1;
                   temp3:= status_reg<0> OR temp2;
                   IF temp3 IS
                          1: cc:=ZERO;
                          0: cc:= uno;
                   END_IF;
                END;
  END_IF;
```

```
(.....control de la secuencia del microprograma.....)

  IF secuence IS
          cont: micro_pc:= INC micro_pc;
          cjs: BEGIN
                    IF cc=0: BEGIN
                                  PUSH microstack:= INC micro_pc;
                                  micro_pc:= next_addr;
                             END;
                    IF cc=1: micro_pc:= INC micro_pc;
               END;
          cjp: BEGIN
                    IF cc=0: micro_pc:= next_addr;
                    IF cc=1: micro_pc:= INC micro_pc;
               END;
          crtn: BEGIN
                     IF cc=0: micro_pc:= POP microstack;
                     IF cc=1: micro_pc:= INC micro_pc;
                END;
          jz: micro_pc:= ZERO;
          jmap: micro_pc:= mapping;
          cjv: micro_pc:= INC micro_pc;
  END_IF;

END_MICRO



Tiempo de ejecucion : 22360 ms      (2165 1 neas/minuto)
```

```
    1  MACHINE CPA;
    2  CONSTANT
    3
    4
    5  (...selección del operando fuente...)
    6          ab =1;
    7          zb =3;
    8          za =4;
    9          da =5;
   10          dz =7;
   11
   12  (...operaciones de la alu...)
   13          add       =0;
   14          sub_r     =1;
   15          sub_s     =2;
   16          or_rs     =3;
   17          and_rs    =4;
   18          and_nors  =5;
   19          exor_rs   =6;
   20          exnor_rs  =7;
   21
   22  (...selección de destino...)
   23          noper =1;
   24          ram_a =2;
   25          ram_f =3;
   26          ram_d =5;
   27          ram_u =7;
   28
   29  (...control de la próxima dirección...)
   30          jz    =0;
   31          cjs   =1;
   32          jmap  =2;
   33          cjp   =3;
   34          push_ =4;
   35          cjv   =6;
   36          crtn  =10;
   37          cont  =14;
   38
   39  (...tamaño datos...)
   40          byte  =0;
   41          word  =1;
   42
   43  (...control de salida al bus interno...)
   44          bus_y =0;
   45          foe   =1;
   46          udat =3;
   47          rd_mask =2;
   48
   49  (...multiplexor de condición...)
   50          neg_   =0;
   51          pos    =1;
   52          zero_  =2;
   53          nozero =3;
   54          par    =4;
   55          impar  =5;
```

```
 56          cy      =6;
 57          nocy    =7;
 58          daal    =8;
 59          daah    =10;
 60          intr    =12;
 61          nopass  =14;
 62          pass    =15;
 63
 64  (...dato de entrada a la alu...)
 65          in      =0;
 66          ex      =1;
 67
 68  (...selección de los flags...)
 69          arit    =0;
 70          indc    =1;
 71          lop     =2;
 72          fdad    =3;
 73          inva    =4;
 74          rota    =5;
 75          cmcy    =6;
 76          fbus    =7;
 77
 78  (...fuente para direccionar los registros...)
 79          irir    =0;
 80          irpl    =1;
 81          plir    =2;
 82          plpl    =3;
 83
 84  (...selección de la alu...)
 85          left    =0;
 86          right   =1;
 87
 88
 89  (*************************************************************************)
 90  (       DECLARACION DE LOS ELEMENTOS DE LA ARQUITECTURA                  )
 91  (*************************************************************************)
 92
 93  (. . . . . declaración de los registros . . . . . .)
 94
 95  REGISTER
 96          reg_inst<8>:=0;
 97          micro_pc<12>:=0;
 98          adr_mem<16>:=0;
 99          adr_mapping<8>:=0;
100          status_reg<8>:=0;
101          input_data<8>;
102          mask_reg<8>:=0;
103          data_reg<8>:=0;
104          adress_reg<16>:=0;
105          uno<1>:=1;
106
107  LATCH
108          parity_reg<1>;
109          d_latch<16>;
110          lat_d<8>;
111          lat_i<8>;
```

```
112
113  (. . . . . declaración de buses . . . . . .)
114
115  BUS
116          salida_alu<17>;
117          salida_alui<9>;
118          salida_alud<9>;
119          a_latch<16>;
120          b_latch<16>;
121          di_latch<8>;
122          dd_latch<8>;
123          alu_r<16>;
124          alu_s<16>;
125          y_bus<17>;
126          alu_ir<8>;
127          alu_is<8>;
128          alu_dr<8>;
129          alu_ds<8>;
130          bus_i<9>;
131          bus_d<9>;
132          ader<4>;
133          bder<4>;
134          aiz<4>;
135          biz<4>;
136          par_test<4>;
137          temp<17>;
138          tempi<9>;
139          tempd<9>;
140          temp1<1>;
141          temp2<1>;
142          temp3<1>;
143          cn1<1>;
144          cn2<1>;
145          cn<1>;
146          cc<1>;
147          ac<5>;
148
149  (. . . . . memorias . . . . . .)
150
151  MEMORY
152          micromem[580]<64>:micro_pc;
153
154          mapping[256]<12>:adr_mapping,0:
155          1D3H,019H,061H,093H,089H,08EH,013H,0D1H,0000,0A3H,
156          05DH,09BH,089H,08EH,013H,0D2H,0000,01FH,061H,095H,
157          089H,08EH,013H,0D3H,0000,0A5H,05DH,09DH,089H,08EH,
158          013H,0D4H,1D4H,025H,052H,097H,089H,08EH,013H,0ABH,
159          0000,0A7H,045H,09FH,089H,08EH,013H,0D5H,1D6H,02BH,
160          03BH,099H,08AH,08FH,016H,0D7H,0000,0A9H,031H,0A1H,
161          089H,08EH,013H,0D6H,00CH,00CH,00CH,00CH,00CH,00CH,
162          00DH,00CH,00CH,00CH,00CH,00CH,00CH,00CH,00DH,00CH,
163          00CH,00CH,00CH,00CH,00CH,00CH,00DH,00CH,00CH,00CH,
164          00CH,00CH,00CH,00CH,00DH,00CH,00CH,00CH,00CH,00CH,
165          00CH,00CH,00DH,00CH,00CH,00CH,00CH,00CH,00CH,00CH,
166          00DH,00CH,011H,011H,011H,011H,011H,011H,1D0H,011H,
167          00CH,00CH,00CH,00CH,00CH,00CH,00DH,00CH,069H,069H,
```

```
168           069H,069H,069H,069H,06AH,069H,071H,071H,071H,071H,
169           071H,071H,072H,071H,079H,079H,079H,079H,079H,079H,
170           07AH,079H,081H,081H,081H,081H,081H,081H,082H,081H,
171           0B1H,0B1H,0B1H,0B1H,0B1H,0B1H,0B2H,0B1H,0B9H,0B9H,
172           0B9H,0B9H,0B9H,0B9H,0BAH,0B9H,0C1H,0C1H,0C1H,0C1H,
173           0C1H,0C1H,0C2H,0C1H,0C9H,0C9H,0C9H,0C9H,0C9H,0C9H,
174           0CAH,0C9H,174H,19DH,0E1H,0DAH,12DH,185H,06EH,1D7H,
175           176H,16DH,0E9H,0000,135H,121H,076H,1DFH,178H,1A4H,
176           0F1H,1C9H,13DH,18BH,07EH,1E3H,17AH,0000,0F9H,1C4H,
177           145H,0000,086H,1E7H,17CH,1ABH,101H,1B9H,14DH,191H,
178           0B6H,1EBH,17EH,184H,109H,064H,155H,0000,0BEH,1EFH,
179           180H,1B2H,111H,1CFH,15DH,197H,0C6H,1F3H,182H,012H,
180           119H,1CEH,165H,0000,0CEH,1F7H;
181
182           mem_prog[65535]<8>:adr_mem;
183
184           aimem[017]<8>:aiz;
185           admem[017]<8>:ader;
186           bimem[017]<8>:biz;
187           bdmem[017]<8>:bder;
188
189
190   STACK
191           microstack[20H]<12>;
192
193
194   (. . . . . definición de campos . . . . . .)
195
196   FIELD
197           b_alu     =<0..3>;
198           a_alu     =<4..7>;
199           source    =<8..10>;
200           cy_in     =<11>;
201           alu       =<12..14>;
202           cy_sel    =<15>;
203           destino   =<16..18>;
204           shifter   =<19>;
205           flags     =<20..22>;
206           leri      =<23>;
207           add_sel   =<24..25>;
208           w_dat     =<26>;
209           dat_in    =<27>;
210           status    =<28..29>;
211           rd        =<30>;
212           io_m      =<31>;
213           wr        =<32>;
214           ts        =<33>;
215           mbre      =<34>;
216           mare      =<35>;
217           ale       =<36>;
218           b_ctrl    =<37..38>;
219           hlda      =<39>;
220           inta      =<40>;
221           edi       =<41>;
222           st_mask   =<42>;
223           ir_en     =<43>;
```

```
224          cc_mux    =<44..47>; .
225          secuence =<48..51>;
226          next_addr=<52..63>;
227
228
229  (. . . . . microoperaciones . . . . . .)
230
231  MICRO_OP
232
233
234  (........tratamiento para los datos de 16 bits........)
235
236  IF w_dat IS
237   word: BEGIN
238          IF add_sel IS
239                     irir: BEGIN
240                               aiz<0>:= ZERO;
241                               aiz<3>:= ZERO;
242                               aiz<1..2>:= reg_inst<4..5>;
243                               ader:= aiz;
244                               biz:= aiz;
245                               bder:= aiz;
246                           END;
247                     irpl: BEGIN
248                               aiz:= a_alu;
249                               ader:= a_alu;
250                               biz<0>:= ZERO;
251                               biz<3>:= ZERO;
252                               biz<1..2>:= reg_inst<4..5>;
253                               bder:= biz;
254                           END;
255                     plir: BEGIN
256                               aiz<0>:= ZERO;
257                               aiz<3>:= ZERO;
258                               aiz<1..2>:= reg_inst<4..5>;
259                               ader:= aiz;
260                               biz:= b_alu;
261                               bder:= b_alu;
262                           END;
263                     plpl: BEGIN
264                               aiz:= a_alu;
265                               ader:= a_alu;
266                               biz:= b_alu;
267                               bder:= b_alu;
268                           END;
269          END_IF;
270
271          a_latch<0..7>:= admem;
272          a_latch<8..15>:= aimem;
273          b_latch<0..7>:= bdmem;
274          b_latch<8..15>:= bimem;
275
276
277  (....selección de los operandos de la alu....)
278          IF source IS
279                  ab: BEGIN
```

```
280                     alu_r:= a_latch;
281                     alu_s:= b_latch;
282                 END;
283             zb: BEGIN
284                     alu_r:= ZERO;
285                     alu_s:= b_latch;
286                 END;
287             za: BEGIN
288                     alu_r:= ZERO;
289                     alu_s:= a_latch;
290                 END;
291             da: BEGIN
292                     alu_r:= d_latch;
293                     alu_s:= a_latch;
294                 END;
295             dz: BEGIN
296                     alu_r:= d_latch;
297                     alu_s:= ZERO;
298                 END;
299     END_IF;
300
301 (....selección del carry....)
302     IF cy_sel IS
303             1: cn:= status_reg<0>;
304             0: cn:= cy_in;
305     END_IF;
306
307 (....función de la alu....)
308     IF alu IS
309             add: BEGIN
310                     temp:= alu_r + alu_s;
311                     IF cn IS
312                             0: salida_alu:= temp;
313                             1: salida_alu:= INC temp;
314                     END_IF;
315                 END;
316
317             sub_r: BEGIN
318                         temp:= alu_s - alu_r;
319                         IF cn IS
320                                 0: salida_alu:= DEC temp;
321                                 1: salida_alu:= temp;
322                         END_IF;
323                    END;
324
325             sub_s: BEGIN
326                         temp:= alu_r - alu_s;
327                         IF cn IS
328                                 0: salida_alu:= DEC temp;
329                                 1: salida_alu:= temp;
330                         END_IF;
331                    END;
332
333             or_rs: salida_alu:= alu_r OR alu_s;
334
335             and_rs: salida_alu:= alu_r AND alu_s;
```

```
336
337                 and_nors: salida_alu:= alu_s AND NOT alu_r;
338
339                 exor_rs: salida_alu:= alu_r XOR alu_s;
340
341                 exnor_rs: BEGIN
342                             temp:= alu_r XOR alu_s;
343                             salida_alu:= NOT temp;
344                           END;
345         END_IF;
346
347 (....control del bus interno y destino de la operación....)
348         IF b_ctrl IS
349             bus_y: BEGIN
350                      IF destino IS
351                          noper: y_bus:= salida_alu;
352                          ram_a: BEGIN
353                                   ader:= bder;
354                                   aiz:= biz;
355                                   bdmem:= salida_alu<0..7>;
356                                   admem:= salida_alu<0..7>;
357                                   bimem:= salida_alu<8..15>;
358                                   aimem:= salida_alu<8..15>;
359                                   y_bus<0..15>:= a_latch;
360                                 END;
361                           ram_f: BEGIN
362                                   ader:= bder;
363                                   aiz:= biz;
364                                   bdmem:= salida_alu<0..7>;
365                                   admem:= salida_alu<0..7>;
366                                   bimem:= salida_alu<8..15>;
367                                   aimem:= salida_alu<8..15>;
368                                   y_bus:= salida_alu;
369                                 END;
370                      END_IF;
371                    END;
372             foe: y_bus<0..7>:= status_reg;
373         END_IF;
374
375         IF st_mask=0: mask_reg:= y_bus<0..7>;
376
377         IF dat_in IS
378                 in: BEGIN
379                         d_latch<0..7>:= y_bus<8..15>;
380                         d_latch<8..15>:= y_bus<0..7>;
381                     END;
382         END_IF;
383
384         IF mare=0: adress_reg:= y_bus<0..15>;
385
386  END;
387
388 (........tratamiento para los datos de 8 bits........)
389
390  byte: BEGIN
391         IF add_sel IS
```

```
392             irir: BEGIN
393                       aiz<0..2>:= reg_inst<0..2>;
394                       aiz<3>:= ZERO;
395                       ader<0>:= NOT aiz<0>;
396                       ader<1..3>:= aiz<1..3>;
397                       biz<0..2>:= reg_inst<3..5>;
398                       biz<3>:= ZERO;
399                       bder<0>:= NOT biz<0>;
400                       bder<1..3>:= biz<1..3>;
401                   END;
402             irpl: BEGIN
403                       aiz:= a_alu;
404                       ader<0>:= NOT aiz<0>;
405                       ader<1..3>:= aiz<1..3>;
406                       biz<0..2>:= reg_inst<3..5>;
407                       biz<3>:= ZERO;
408                       bder<0>:= NOT biz<0>;
409                       bder<1..3>:= biz<1..3>;
410                   END;
411             plir: BEGIN
412                       aiz<0..2>:= reg_inst<0..2>;
413                       aiz<3>:= ZERO;
414                       ader<0>:= NOT aiz<0>;
415                       ader<1..3>:= aiz<1..3>;
416                       biz:= b_alu;
417                       bder<0>:= NOT biz<0>;
418                       bder<1..3>:= biz<1..3>;
419                   END;
420             plpl: BEGIN
421                       aiz:= a_alu;
422                       ader<0>:= NOT aiz<0>;
423                       ader<1..3>:= aiz<1..3>;
424                       biz:= b_alu;
425                       bder<0>:= NOT biz<0>;
426                       bder<1..3>:= biz<1..3>;
427                   END;
428         END_IF;
429
430         IF b_ctrl IS
431                 udat: BEGIN
432                           bus_i:= next_addr;
433                           bus_d:= next_addr;
434                       END;
435                 rd_mask: BEGIN
436                             bus_i:= mask_reg;
437                             bus_d:= mask_reg;
438                           END;
439         END_IF;
440         IF dat_in IS
441                 ex: BEGIN
442                         di_latch:= input_data;
443                         dd_latch:= input_data;
444                     END;
445                 in: BEGIN
446                         di_latch:= lat_i;
447                         dd_latch:= lat_d;
```

```
448                     END;
449         END_IF;
450
451  (....fuente de datos de la alu....)
452         IF source IS
453             ab: BEGIN
454                     alu_ir:= aimem;
455                     alu_is:= bimem;
456                     alu_dr:= admem;
457                     alu_ds:= bdmem;
458                 END;
459             zb: BEGIN
460                     alu_ir:= ZERO;
461                     alu_is:= bimem;
462                     alu_dr:= ZERO;
463                     alu_ds:= bdmem;
464                 END;
465             za: BEGIN
466                     alu_ir:= ZERO;
467                     alu_is:= aimem;
468                     alu_dr:= ZERO;
469                     alu_ds:= admem;
470                 END;
471             da: BEGIN
472                     alu_ir:= di_latch;
473                     alu_is:= aimem;
474                     alu_dr:= dd_latch;
475                     alu_ds:= admem;
476                 END;
477             dz: BEGIN
478                     alu_ir:= di_latch;
479                     alu_is:= ZERO;
480                     alu_dr:= dd_latch;
481                     alu_ds:= ZERO;
482                 END;
483         END_IF;
484
485  (....selección del carry....)
486         IF cy_sel IS
487                 1: BEGIN
488                        cn1:= status_reg<0>;
489                        cn2:= status_reg<0>;
490                    END;
491                 0: BEGIN
492                           cn1:= cy_in;
493                           cn2:= cy_in;
494                    END;
495         END_IF;
496
497  (....operación de la alu....)
498         IF alu IS
499             add: BEGIN
500                     tempi:= alu_ir + alu_is;
501                     tempd:= alu_dr + alu_ds;
502                     ac:= alu_ir<0..3> + alu_is<0..3>;
503                     IF cn1 IS
```

```
504                              0: salida_alui:= tempi;
505                              1: BEGIN
506                                  salida_alui:= INC tempi;
507                                  ac:= INC ac;
508                                 END;
509                      END_IF;
510                      IF cn2 IS
511                              0: salida_alud:= tempd;
512                              1: salida_alud:= INC tempd;
513                      END_IF;
514                   END;
515              sub_r: BEGIN
516                        tempi:= alu_is - alu_ir;
517                        tempd:= alu_ds - alu_dr;
518                        ac:= alu_is<0..3> - alu_ir<0..3>;
519                      IF cn1 IS
520                              0: BEGIN
521                                  salida_alui:= DEC tempi;
522                                  ac:= DEC ac;
523                                 END;
524                              1: salida_alui:= tempi;
525                      END_IF;
526                      IF cn2 IS
527                              0: salida_alud:= DEC tempd;
528                              1: salida_alud:= tempd;
529                      END_IF;
530                    END;
531              sub_s: BEGIN
532                        tempi:= alu_ir - alu_is;
533                        tempd:= alu_dr - alu_ds;
534                        ac:= alu_ir<0..3> - alu_is<0..3>;
535                      IF cn1 IS
536                              0: BEGIN
537                                  salida_alui:= DEC tempi;
538                                  ac:= DEC ac;
539                                 END;
540                              1: salida_alui:= tempi;
541                      END_IF;
542                      IF cn2 IS
543                              0: salida_alud:= DEC tempd;
544                              1: salida_alud:= tempd;
545                      END_IF;
546                    END;
547              or_rs: BEGIN
548                        salida_alui:= alu_ir OR alu_is;
549                        salida_alud:= alu_dr OR alu_ds;
550                    END;
551             and_rs: BEGIN
552                         salida_alui:= alu_ir AND alu_is;
553                         salida_alud:= alu_dr AND alu_ds;
554                     END;
555            and_nors: BEGIN
556                         salida_alui:= alu_ir AND NOT alu_is;
557                         salida_alud:= alu_dr AND NOT alu_ds;
558                      END;
559            exor_rs: BEGIN
```

```
560                          salida_alui:= alu_ir XOR alu_is;
561                          salida_alud:= alu_dr XOR alu_ds;
562                       END;
563              exnor_rs: BEGIN
564                          tempi:= alu_ir XOR alu_is;
565                          tempd:= alu_dr XOR alu_ds;
566                          salida_alui:= NOT tempi;
567                          salida_alud:= NOT tempd;
568                       END;
569          END_IF;
570
571  (....control del bus y destino de la operación....)
572          IF b_ctrl IS
573               bus_y: IF destino IS
574                         noper: BEGIN
575                                   bus_i:= salida_alui;
576                                   bus_d:= salida_alud;
577                                END;
578                         ram_a: BEGIN
579                                   ader:= bder;
580                                   aiz:= biz;
581                                   bus_i<0..7>:= aimem;
582                                   bimem:= salida_alui<0..7>;
583                                   aimem:= salida_alui<0..7>;
584                                   bus_d<0..7>:= admem;
585                                   bdmem:= salida_alud<0..7>;
586                                   admem:= salida_alud<0..7>;
587                                END;
588                         ram_f: BEGIN
589                                   ader:= bder;
590                                   aiz:= biz;
591                                   bimem:= salida_alui<0..7>;
592                                   aimem:= salida_alui<0..7>;
593                                   bdmem:= salida_alud<0..7>;
594                                   admem:= salida_alud<0..7>;
595                                   bus_i:= salida_alui;
596                                   bus_d:= salida_alud;
597                                END;
598                         ram_d: BEGIN
599                                   bus_i:= salida_alui;
600                                   bus_d:= salida_alud;
601                                   ader:= bder;
602                                   aiz:= biz;
603                                  IF shifter IS
604                                    0: BEGIN
605                                         salida_alui:= RR salida_alui;
606                                         salida_alui<7>:=salida_alui<8>;
607                                         bimem:=salida_alui<0..7>;
608                                         aimem:=salida_alui<0..7>;
609                                         salida_alud:= RR salida_alud;
610                                         salida_alud<7>:=salida_alud<8>;
611                                         bdmem:=salida_alud<0..7>;
612                                         admem:=salida_alud<0..7>;
613                                       END;
614                                    1: BEGIN
615                                         salida_alui:= RR salida_alui;
```

```
616                                    ..      salida_alui<7>:= status_reg<0>;
617                                            bimem:=salida_alui<0..7>;
618                                            aimem:=salida_alui<0..7>;
619                                            salida_alud:= RR salida_alud;
620                                            salida_alud<7>:= status_reg<0>;
621                                            bdmem:=salida_alud<0..7>;
622                                            admem:=salida_alud<0..7>;
623                                          END;
624                                    END_IF;
625                                  END;
626                          ram_u: BEGIN
627                                     bus_i:= salida_alui;
628                                     bus_d:= salida_alud;
629                                     ader:= bder;
630                                     aiz:= biz;
631                                   IF shifter IS
632                                      0: BEGIN
633                                          salida_alui:= LSL salida_alui;
634                                          salida_alui<0>:=salida_alui<8>;
635                                          bimem:=salida_alui<0..7>;
636                                          aimem:=salida_alui<0..7>;
637                                          salida_alud:= LSL salida_alud;
638                                          salida_alud<0>:= salida_alud<8>;
639                                          bdmem:=salida_alud<0..7>;
640                                          admem:=salida_alud<0..7>;
641                                        END;
642                                      1: BEGIN
643                                          salida_alui:= LSL salida_alui;
644                                          salida_alui<0>:= status_reg<0>;
645                                          bimem:=salida_alui<0..7>;
646                                          aimem:=salida_alui<0..7>;
647                                          salida_alud:= LSL salida_alud;
648                                          salida_alud<0>:= status_reg<0>;
649                                          bdmem:=salida_alud<0..7>;
650                                          admem:=salida_alud<0..7>;
651                                        END;
652                                     END_IF;
653                                  END;
654                     END_IF;
655                  foe: BEGIN
656                           bus_i<0..7>:= status_reg;
657                           bus_d<0..7>:= status_reg;
658                        END;
659                  udat: BEGIN
660                            ader:= bder;
661                            aiz:= biz;
662                            bimem:= salida_alui<0..7>;
663                            aimem:= salida_alui<0..7>;
664                            bdmem:= salida_alud<0..7>;
665                            admem:= salida_alud<0..7>;
666                        END;
667
668                  rd_mask: BEGIN
669                               ader:= bder;
670                               aiz:= biz;
671                               bimem:= salida_alui<0..7>;
```

```
672                         aimem:= salida_alui<0..7>;
673                         bdmem:= salida_alud<0..7>;
674                         admem:= salida_alud<0..7>;
675                       END;
676         END_IF;
677
678         IF st_mask=0: mask_reg:= bus_i;
679
680         IF mbre=0: IF leri IS
681                       left:  data_reg:=bus_i<0..7>;
682                       right: data_reg:=bus_d<0..7>;
683                    END_IF;
684         lat_i:= bus_d;
685         lat_d:= bus_i;
686
687  (....cálculo de la paridad....)
688         par_test:= ZERO;
689
690         IF salida_alui<0>=1: par_test:= INC par_test;
691         IF salida_alui<1>=1: par_test:= INC par_test;
692         IF salida_alui<2>=1: par_test:= INC par_test;
693         IF salida_alui<3>=1: par_test:= INC par_test;
694         IF salida_alui<4>=1: par_test:= INC par_test;
695         IF salida_alui<5>=1: par_test:= INC par_test;
696         IF salida_alui<6>=1: par_test:= INC par_test;
697         IF salida_alui<7>=1: par_test:= INC par_test;
698
699         IF par_test<0>=0: parity_reg:= uno;
700         IF par_test<0>=1: parity_reg:= ZERO;
701     END;
702  END_IF;
703
704  (....carga del estado de las operaciones....)
705
706  IF flags IS
707      arit: BEGIN
708                status_reg<0>:= salida_alui<8>;
709                status_reg<2>:= parity_reg;
710                status_reg<4>:= ac<4>;
711                IF salida_alui=0: status_reg<6>:=uno;
712                status_reg<7>:= salida_alui<7>;
713            END;
714      indc: BEGIN
715                status_reg<2>:= parity_reg;
716                status_reg<4>:= ac<4>;
717                IF salida_alui=0: status_reg<6>:=uno;
718                status_reg<7>:= salida_alui<7>;
719            END;
720      lop: BEGIN
721                status_reg<0>:= ZERO;
722                status_reg<2>:= parity_reg;
723                status_reg<4>:= ZERO;
724                IF salida_alui=0: status_reg<6>:=uno;
725                status_reg<7>:= salida_alui<7>;
726            END;
727      fdad: status_reg<0>:= salida_alu<16>;
```

```
728     cmcy: status_reg<0>:= NOT status_reg<0>;
729     fbus: status_reg:=.bus_i<0..7>;
730     rota: status_reg<0>:= salida_alui<8>;
731 END_IF;
732
733 (.........búsqueda del opcode y dirección del operando.........)
734
735 IF ir_en=0: BEGIN
736                 reg_inst:= mem_prog;
737                 adr_mapping:= mem_prog;
738             END;
739
740 IF wr=0: mem_prog:= data_reg;
741
742 IF rd=0: input_data:= mem_prog;
743
744 IF ale=1: adr_mem:= adress_reg;
745
746 (.........estado del multiplexor de condición.........)
747
748 IF cc_mux IS
749         neg_:  cc:= NOT status_reg<7>;
750         pos:  cc:= status_reg<7>;
751         zero_:  cc:= NOT status_reg<6>;
752         nozero:  cc:= status_reg<6>;
753         par:  cc:= NOT status_reg<2>;
754         impar:  cc:= status_reg<2>;
755         cy:  cc:= NOT status_reg<0>;
756         nocy:  cc:= status_reg<0>;
757         pass:  cc:= ZERO;
758         nopass:  cc:= uno;
759         daal: BEGIN
760                 temp1:= salida_alui<2> OR salida_alui<1>;
761                 temp2:= salida_alui<3> AND temp1;
762                 temp3:= status_reg<4> OR temp2;
763                 IF temp3 IS
764                         1: cc:=ZERO;
765                         0: cc:= uno;
766                 END_IF;
767             END;
768         daah: BEGIN
769                 temp1:= salida_alui<6> OR salida_alui<5>;
770                 temp2:= salida_alui<7> AND temp1;
771                 temp3:= status_reg<0> OR temp2;
772                 IF temp3 IS
773                         1: cc:=ZERO;
774                         0: cc:= uno;
775                 END_IF;
776             END;
777 END_IF;
778
```

```
779  (.....control de la secuencia del microprograma.....)
780
781  IF secuence IS
782          cont: micro_pc:= INC micro_pc;
783          cjs: BEGIN
784                   IF cc=0: BEGIN
785                                PUSH microstack:= INC micro_pc;
786                                micro_pc:= next_addr;
787                            END;
788                   IF cc=1: micro_pc:= INC micro_pc;
789               END;
790          cjp: BEGIN
791                   IF cc=0: micro_pc:= next_addr;
792                   IF cc=1: micro_pc:= INC micro_pc;
793               END;
794          crtn: BEGIN
795                    IF cc=0: micro_pc:= POP microstack;
796                    IF cc=1: micro_pc:= INC micro_pc;
797                END;
798          jz: micro_pc:= ZERO;
799          jmap: micro_pc:= mapping;
800          cjv: micro_pc:= INC micro_pc;
801  END_IF;
802
803  END_MICRO
804
805
806
```

Tiempo de ejecucion : 22360 ms (2165 l neas/minuto)

4.2 DEPURACION DE PROGRAMAS

```
                    SIMULADOR   RTL  SILOSS
                        EPFL - DET(UPC)

run sim2 /definición=CPA.REG /binario=CPA.BIN


SIMULADOR SILOSS. Versión 2.4 DPTO. ELECTRONICA. MARZO 1.989
============================================================

*****EJECUCION:

Para visualizar el menú pulsar "?"


Lectura de definiciones
Lectura del micro-programa de la memoria MICROMEM
SIM> R
ADRESS_REG            = 00000000
ADR_MAPPING           = 00000000
ADR_MEM               = 00000000
DATA_REG              = 00000000
INPUT_DATA            = 00000000
MASK_REG              = 00000000
MICRO_PC              = 00000000
REG_INST              = 00000000
STATUS_REG            = 00000000
UNO                   = 00000001
SIM> ED
paso 1.

  236 test : MICROMEM              26 26 == 00000000
  391 test : MICROMEM              24 25 == 00000000
  393 asig.: AIZ                    0  2 := 00000000
  394 asig.: AIZ                    3  3 := 00000000
  395 asig.: ADER                   0  0 := 00000001
  396 asig.: ADER                   1  3 := 00000000
  397 asig.: BIZ                    0  2 := 00000000
  398 asig.: BIZ                    3  3 := 00000000
  399 asig.: BDER                   0  0 := 00000001
  400 asig.: BDER                   1  3 := 00000000
  430 test : MICROMEM              37 38 == 00000003
  432 asig.: BUS_I                  0  8 := 00000006
  433 asig.: BUS_D                  0  8 := 00000006
  440 test : MICROMEM              27 27 == 00000000
  446 asig.: DI_LATCH               0  7 := 00000000
  447 asig.: DD_LATCH               0  7 := 00000000
  452 test : MICROMEM               8 10 == 00000003
  460 asig.: ALU_IR                 0  7 := 00000000
  461 asig.: ALU_IS                 0  7 := 00000000
  462 asig.: ALU_DR                 0  7 := 00000000
  463 asig.: ALU_DS                 0  7 := 00000000
  486 test : MICROMEM              15 15 == 00000000
  492 asig.: CN1                    0  0 := 00000000
  493 asig.: CN2                    0  0 := 00000000
  498 test : MICROMEM              12 14 == 00000003
  548 asig.: SALIDA_ALUI            0  8 := 00000000
  549 asig.: SALIDA_ALUD            0  8 := 00000000
```

```
  572 test : MICROMEM              37 38 == 00000003
  660 asig.: ADER                   0  3 := 00000001
  661 asig.: AIZ                    0  3 := 00000000
  662 asig.: BIMEM                  0  7 := 00000000
  663 asig.: AIMEM                  0  7 := 00000000
  664 asig.: BDMEM                  0  7 := 00000000
  665 asig.: ADMEM                  0  7 := 00000000
  678 test : MICROMEM              42 42 == 00000001
  680 test : MICROMEM              34 34 == 00000001
  684 asig.: LAT_I                  0  7 := 00000006
  685 asig.: LAT_D                  0  7 := 00000006
  688 asig.: PAR_TEST               0  3 := 00000000
  690 test : SALIDA_ALUI            0  0 == 00000000
  691 test : SALIDA_ALUI            1  1 == 00000000
  692 test : SALIDA_ALUI            2  2 == 00000000
  693 test : SALIDA_ALUI            3  3 == 00000000
  694 test : SALIDA_ALUI            4  4 == 00000000
  695 test : SALIDA_ALUI            5  5 == 00000000
  696 test : SALIDA_ALUI            6  6 == 00000000
  697 test : SALIDA_ALUI            7  7 == 00000000
  699 test : PAR_TEST               0  0 == 00000000
  699 asig.: PARITY_REG             0  0 := 00000001
  700 test : PAR_TEST               0  0 == 00000000
  706 test : MICROMEM              20 22 == 00000004
  735 test : MICROMEM              43 43 == 00000001
  740 test : MICROMEM              32 32 == 00000001
  742 test : MICROMEM              30 30 == 00000001
  744 test : MICROMEM              36 36 == 00000000
  748 test : MICROMEM              44 47 == 0000000E
  758 asig.: CC                     0  0 := 00000001
  781 test : MICROMEM              48 51 == 0000000E
  782 asig.: MICRO_PC               0 11 := 00000001

1. paso : fin normal
SIM> ED
paso 1.

  236 test : MICROMEM              26 26 == 00000000
  391 test : MICROMEM              24 25 == 00000003
  421 asig.: AIZ                    0  3 := 0000000D
  422 asig.: ADER                   0  0 := 00000000
  423 asig.: ADER                   1  3 := 00000006
  424 asig.: BIZ                    0  3 := 0000000D
  425 asig.: BDER                   0  0 := 00000000
  426 asig.: BDER                   1  3 := 00000006
  430 test : MICROMEM              37 38 == 00000003
  432 asig.: BUS_I                  0  8 := 00000000
  433 asig.: BUS_D                  0  8 := 00000000
  440 test : MICROMEM              27 27 == 00000000
  446 asig.: DI_LATCH               0  7 := 00000006
  447 asig.: DD_LATCH               0  7 := 00000006
  452 test : MICROMEM               8 10 == 00000007
  478 asig.: ALU_IR                 0  7 := 00000006
  479 asig.: ALU_IS                 0  7 := 00000000
```

```
  480 asig.: ALU_DR              0  7 := 00000006
  481 asig.: ALU_DS              0  7 := 00000000
  486 test : MICROMEM           15 15 == 00000000
  492 asig.: CN1                 0  0 := 00000000
  493 asig.: CN2                 0  0 := 00000000
  498 test : MICROMEM           12 14 == 00000003
  548 asig.: SALIDA_ALUI         0  8 := 00000006
  549 asig.: SALIDA_ALUD         0  8 := 00000006
  572 test : MICROMEM           37 38 == 00000003
  660 asig.: ADER                0  3 := 0000000C
  661 asig.: AIZ                 0  3 := 0000000D
  662 asig.: BIMEM               0  7 := 00000006
  663 asig.: AIMEM               0  7 := 00000006
  664 asig.: BDMEM               0  7 := 00000006
  665 asig.: ADMEM               0  7 := 00000006
  678 test : MICROMEM           42 42 == 00000001
  680 test : MICROMEM           34 34 == 00000001
  684 asig.: LAT_I               0  7 := 00000000
  685 asig.: LAT_D               0  7 := 00000000
  688 asig.: PAR_TEST            0  3 := 00000000
  690 test : SALIDA_ALUI         0  0 == 00000000
  691 test : SALIDA_ALUI         1  1 == 00000001
  691 asig.: PAR_TEST            0  3 := 00000001
  692 test : SALIDA_ALUI         2  2 == 00000001
  692 asig.: PAR_TEST            0  3 := 00000002
  693 test : SALIDA_ALUI         3  3 == 00000000
  694 test : SALIDA_ALUI         4  4 == 00000000
  695 test : SALIDA_ALUI         5  5 == 00000000
  696 test : SALIDA_ALUI         6  6 == 00000000
  697 test : SALIDA_ALUI         7  7 == 00000000
  699 test : PAR_TEST            0  0 == 00000000
  699 asig.: PARITY_REG          0  0 := 00000001
  700 test : PAR_TEST            0  0 == 00000000
  706 test : MICROMEM           20 22 == 00000004
  735 test : MICROMEM           43 43 == 00000001
  740 test : MICROMEM           32 32 == 00000001
  742 test : MICROMEM           30 30 == 00000001
  744 test : MICROMEM           36 36 == 00000000
  748 test : MICROMEM           44 47 == 0000000E
  758 asig.: CC                  0  0 := 00000001
  781 test : MICROMEM           48 51 == 0000000E
  782 asig.: MICRO_PC            0 11 := 00000002

1. paso : fin normal
SIM> E2
2. paso : fin normal
SIM> ED1
paso 1.

  236 test : MICROMEM           26 26 == 00000000
  391 test : MICROMEM           24 25 == 00000000
  393 asig.: AIZ                 0  2 := 00000000
  394 asig.: AIZ                 3  3 := 00000000
  395 asig.: ADER                0  0 := 00000001
```

```
396 asig.: ADER                1  3 := 00000000
397 asig.: BIZ                 0  2 := 00000000
398 asig.: BIZ                 3  3 := 00000000
399 asig.: BDER                0  0 := 00000001
400 asig.: BDER                1  3 := 00000000
430 test : MICROMEM           37 38 == 00000003
432 asig.: BUS_I               0  8 := 00000007
433 asig.: BUS_D               0  8 := 00000007
440 test : MICROMEM           27 27 == 00000000
446 asig.: DI_LATCH            0  7 := 00000000
447 asig.: DD_LATCH            0  7 := 00000000
452 test : MICROMEM            8 10 == 00000003
460 asig.: ALU_IR              0  7 := 00000000
461 asig.: ALU_IS              0  7 := 00000000
462 asig.: ALU_DR              0  7 := 00000000
463 asig.: ALU_DS              0  7 := 00000000
486 test : MICROMEM           15 15 == 00000000
492 asig.: CN1                 0  0 := 00000000
493 asig.: CN2                 0  0 := 00000000
498 test : MICROMEM           12 14 == 00000003
548 asig.: SALIDA_ALUI         0  8 := 00000000
549 asig.: SALIDA_ALUD         0  8 := 00000000
572 test : MICROMEM           37 38 == 00000003
660 asig.: ADER                0  3 := 00000001
661 asig.: AIZ                 0  3 := 00000000
662 asig.: BIMEM               0  7 := 00000000
663 asig.: AIMEM               0  7 := 00000000
664 asig.: BDMEM               0  7 := 00000000
665 asig.: ADMEM               0  7 := 00000000
678 test : MICROMEM           42 42 == 00000000
678 asig.: MASK_REG            0  7 := 00000007
680 test : MICROMEM           34 34 == 00000001
684 asig.: LAT_I               0  7 := 00000007
685 asig.: LAT_D               0  7 := 00000007
688 asig.: PAR_TEST            0  3 := 00000000
690 test : SALIDA_ALUI         0  0 == 00000000
691 test : SALIDA_ALUI         1  1 == 00000000
692 test : SALIDA_ALUI         2  2 == 00000000
693 test : SALIDA_ALUI         3  3 == 00000000
694 test : SALIDA_ALUI         4  4 == 00000000
695 test : SALIDA_ALUI         5  5 == 00000000
696 test : SALIDA_ALUI         6  6 == 00000000
697 test : SALIDA_ALUI         7  7 == 00000000
699 test : PAR_TEST            0  0 == 00000000
699 asig.: PARITY_REG          0  0 := 00000001
700 test : PAR_TEST            0  0 == 00000000
706 test : MICROMEM           20 22 == 00000004
735 test : MICROMEM           43 43 == 00000001
740 test : MICROMEM           32 32 == 00000001
742 test : MICROMEM           30 30 == 00000001
744 test : MICROMEM           36 36 == 00000000
748 test : MICROMEM           44 47 == 0000000E
758 asig.: CC                  0  0 := 00000001
781 test : MICROMEM           48 51 == 0000000E
782 asig.: MICRO_PC            0 11 := 00000005
```

```
1. paso : fin normal
SIM> ED
paso 1.

  236 test : MICROMEM              26 26 == 00000001
  238 test : MICROMEM              24 25 == 00000003
  264 asig.: AIZ                    0  3 := 0000000F
  265 asig.: ADER                   0  3 := 0000000F
  266 asig.: BIZ                    0  3 := 0000000F
  267 asig.: BDER                   0  3 := 0000000F
  271 asig.: A_LATCH                0  7 := 00000000
  272 asig.: A_LATCH                8 15 := 00000000
  273 asig.: B_LATCH                0  7 := 00000000
  274 asig.: B_LATCH                8 15 := 00000000
  278 test : MICROMEM               8 10 == 00000004
  288 asig.: ALU_R                  0 15 := 00000000
  289 asig.: ALU_S                  0 15 := 00000000
  302 test : MICROMEM              15 15 == 00000000
  304 asig.: CN                     0  0 := 00000000
  308 test : MICROMEM              12 14 == 00000004
  335 asig.: SALIDA_ALU             0 16 := 00000000
  348 test : MICROMEM              37 38 == 00000000
  350 test : MICROMEM              16 18 == 00000003
  362 asig.: ADER                   0  3 := 0000000F
  363 asig.: AIZ                    0  3 := 0000000F
  364 asig.: BDMEM                  0  7 := 00000000
  365 asig.: ADMEM                  0  7 := 00000000
  366 asig.: BIMEM                  0  7 := 00000000
  367 asig.: AIMEM                  0  7 := 00000000
  368 asig.: Y_BUS                  0 16 := 00000000
  375 test : MICROMEM              42 42 == 00000001
  377 test : MICROMEM              27 27 == 00000000
  379 asig.: D_LATCH                0  7 := 00000000
  380 asig.: D_LATCH                8 15 := 00000000
  384 test : MICROMEM              35 35 == 00000000
  384 asig.: ADRESS_REG             0 15 := 00000000
  706 test : MICROMEM              20 22 == 00000004
  735 test : MICROMEM              43 43 == 00000001
  740 test : MICROMEM              32 32 == 00000001
  742 test : MICROMEM              30 30 == 00000001
  744 test : MICROMEM              36 36 == 00000000
  748 test : MICROMEM              44 47 == 0000000E
  758 asig.: CC                     0  0 := 00000001
  781 test : MICROMEM              48 51 == 0000000E
  782 asig.: MICRO_PC               0 11 := 00000006

1. paso : fin normal
SIM> R
ADRESS_REG          = 00000000
ADR_MAPPING         = 00000000
ADR_MEM             = 00000000
DATA_REG            = 00000000
INPUT_DATA          = 00000000
MASK_REG            = 00000007
MICRO_PC            = 00000006
```

```
REG_INST              = 00000000
STATUS_REG            = 00000000
UNO                   = 00000001
SIM> AM AIMEM 0 10
AIMEM                 : AIZ
[00000000]   00
[00000001]   00
[00000002]   00
[00000003]   00
[00000004]   00
[00000005]   00
[00000006]   00
[00000007]   00
[00000008]   00
[00000009]   00
[0000000A]   00
[0000000B]   00
[0000000C]   60
[0000000D]   06
[0000000E]   00
[0000000F]   00
SIM> AM ADMEM 0 10
ADMEM                 : ADER
[00000000]   00
[00000001]   00
[00000002]   00
[00000003]   00
[00000004]   00
[00000005]   00
[00000006]   00
[00000007]   00
[00000008]   00
[00000009]   00
[0000000A]   00
[0000000B]   00
[0000000C]   06
[0000000D]   60
[0000000E]   00
[0000000F]   00
SIM> AM BIMEM 0 10
BIMEM                 : BIZ
[00000000]   00
[00000001]   00
[00000002]   00
[00000003]   00
[00000004]   00
[00000005]   00
[00000006]   00
[00000007]   00
[00000008]   00
[00000009]   00
[0000000A]   00
[0000000B]   00
[0000000C]   60
```

```
[0000000D]   06
[0000000E]   00
[0000000F]   00
SIM> AM BDMEM 0 10
BDMEM                  : BDER
[00000000]   00
[00000001]   00
[00000002]   00
[00000003]   00
[00000004]   00
[00000005]   00
[00000006]   00
[00000007]   00
[00000008]   00
[00000009]   00
[0000000A]   00
[0000000B]   00
[0000000C]   06
[0000000D]   60
[0000000E]   00
[0000000F]   00
SIM> ED3
paso 1.

  236 test : MICROMEM          26 26 == 00000000
  391 test : MICROMEM          24 25 == 00000000
  393 asig.: AIZ                0  2 := 00000000
  394 asig.: AIZ                3  3 := 00000000
  395 asig.: ADER               0  0 := 00000001
  396 asig.: ADER               1  3 := 00000000
  397 asig.: BIZ                0  2 := 00000000
  398 asig.: BIZ                3  3 := 00000000
  399 asig.: BDER               0  0 := 00000001
  400 asig.: BDER               1  3 := 00000000
  430 test : MICROMEM          37 38 == 00000000
  440 test : MICROMEM          27 27 == 00000000
  446 asig.: DI_LATCH           0  7 := 00000007
  447 asig.: DD_LATCH           0  7 := 00000007
  452 test : MICROMEM           8 10 == 00000003
  460 asig.: ALU_IR             0  7 := 00000000
  461 asig.: ALU_IS             0  7 := 00000000
  462 asig.: ALU_DR             0  7 := 00000000
  463 asig.: ALU_DS             0  7 := 00000000
  486 test : MICROMEM          15 15 == 00000000
  492 asig.: CN1                0  0 := 00000000
  493 asig.: CN2                0  0 := 00000000
  498 test : MICROMEM          12 14 == 00000003
  548 asig.: SALIDA_ALUI        0  8 := 00000000
  549 asig.: SALIDA_ALUD        0  8 := 00000000
  572 test : MICROMEM          37 38 == 00000000
  573 test : MICROMEM          16 18 == 00000001
  575 asig.: BUS_I              0  8 := 00000000
  576 asig.: BUS_D              0  8 := 00000000
  678 test : MICROMEM          42 42 == 00000001
```

```
680 test : MICROMEM          34 34 == 00000001
684 asig.: LAT_I              0  7 := 00000000
685 asig.: LAT_D              0  7 := 00000000
688 asig.: PAR_TEST           0  3 := 00000000
690 test : SALIDA_ALUI        0  0 == 00000000
691 test : SALIDA_ALUI        1  1 == 00000000
692 test : SALIDA_ALUI        2  2 == 00000000
693 test : SALIDA_ALUI        3  3 == 00000000
694 test : SALIDA_ALUI        4  4 == 00000000
695 test : SALIDA_ALUI        5  5 == 00000000
696 test : SALIDA_ALUI        6  6 == 00000000
697 test : SALIDA_ALUI        7  7 == 00000000
699 test : PAR_TEST           0  0 == 00000000
699 asig.: PARITY_REG         0  0 := 00000001
700 test : PAR_TEST           0  0 == 00000000
706 test : MICROMEM          20 22 == 00000004
735 test : MICROMEM          43 43 == 00000001
740 test : MICROMEM          32 32 == 00000001
742 test : MICROMEM          30 30 == 00000001
744 test : MICROMEM          36 36 == 00000001
744 asig.: ADR_MEM            0 15 := 00000000
748 test : MICROMEM          44 47 == 0000000E
758 asig.: CC                 0  0 := 00000001
781 test : MICROMEM          48 51 == 0000000E
782 asig.: MICRO_PC           0 11 := 00000007
```

paso 2.

```
236 test : MICROMEM          26 26 == 00000000
391 test : MICROMEM          24 25 == (0000000
393 asig.: AIZ                0  2 := 00000000
394 asig.: AIZ                3  3 := 00000000
395 asig.: ADER               0  0 := 00000001
396 asig.: ADER               1  3 := 00000000
397 asig.: BIZ                0  2 := 00000000
398 asig.: BIZ                3  3 := 00000000
399 asig.: BDER               0  0 := 00000001
400 asig.: BDER               1  3 := 00000000
430 test : MICROMEM          37 38 == 00000000
440 test : MICROMEM          27 27 == 00000000
446 asig.: DI_LATCH           0  7 := 00000000
447 asig.: DD_LATCH           0  7 := 00000000
452 test : MICROMEM           8 10 == 00000003
460 asig.: ALU_IR             0  7 := 00000000
461 asig.: ALU_IS             0  7 := 00000000
462 asig.: ALU_DR             0  7 := 00000000
463 asig.: ALU_DS             0  7 := 00000000
486 test : MICROMEM          15 15 == 00000000
492 asig.: CN1                0  0 := 00000000
493 asig.: CN2                0  0 := 00000000
498 test : MICROMEM          12 14 == 00000003
548 asig.: SALIDA_ALUI        0  8 := 00000000
549 asig.: SALIDA_ALUD        0  8 := 00000000
572 test : MICROMEM          37 38 == 00000000
```

```
573 test : MICROMEM          16 18 == 00000001
575 asig.: BUS_I              0  8 := 00000000
576 asig.: BUS_D              0  8 := 00000000
678 test : MICROMEM          42 42 == 00000001
680 test : MICROMEM          34 34 == 00000001
684 asig.: LAT_I              0  7 := 00000000
685 asig.: LAT_D              0  7 := 00000000
688 asig.: PAR_TEST           0  3 := 00000000
690 test : SALIDA_ALUI        0  0 == 00000000
691 test : SALIDA_ALUI        1  1 == 00000000
692 test : SALIDA_ALUI        2  2 == 00000000
693 test : SALIDA_ALUI        3  3 == 00000000
694 test : SALIDA_ALUI        4  4 == 00000000
695 test : SALIDA_ALUI        5  5 == 00000000
696 test : SALIDA_ALUI        6  6 == 00000000
697 test : SALIDA_ALUI        7  7 == 00000000
699 test : PAR_TEST           0  0 == 00000000
699 asig.: PARITY_REG         0  0 := 00000001
700 test : PAR_TEST           0  0 == 00000000
706 test : MICROMEM          20 22 == 00000004
735 test : MICROMEM          43 43 == 00000000
736 asig.: REG_INST           0  7 := 00000000
737 asig.: ADR_MAPPING        0  7 := 00000000
740 test : MICROMEM          32 32 == 00000001
742 test : MICROMEM          30 30 == 00000000
742 asig.: INPUT_DATA         0  7 := 00000000
744 test : MICROMEM          36 36 == 00000000
748 test : MICROMEM          44 47 == 0000000C
781 test : MICROMEM          48 51 == 00000006
800 asig.: MICRO_PC           0 11 := 00000008
```

paso 3.

```
236 test : MICROMEM          26 26 == 00000001
238 test : MICROMEM          24 25 == 00000003
264 asig.: AIZ                0  3 := 0000000F
265 asig.: ADER               0  3 := 0000000F
266 asig.: BIZ                0  3 := 0000000F
267 asig.: BDER               0  3 := 0000000F
271 asig.: A_LATCH            0  7 := 00000000
272 asig.: A_LATCH            8 15 := 00000000
273 asig.: B_LATCH            0  7 := 00000000
274 asig.: B_LATCH            8 15 := 00000000
278 test : MICROMEM           8 10 == 00000004
288 asig.: ALU_R              0 15 := 00000000
289 asig.: ALU_S              0 15 := 00000000
302 test : MICROMEM          15 15 == 00000000
304 asig.: CN                 0  0 := 00000001
308 test : MICROMEM          12 14 == 00000000
310 asig.: TEMP               0 16 := 00000000
311 test : CN                 0  0 == 00000001
313 asig.: SALIDA_ALU         0 16 := 00000001
348 test : MICROMEM          37 38 == 00000000
350 test : MICROMEM          16 18 == 00000003
```

```
  362 asig.: ADER                    0  3 := 0000000F
  363 asig.: AIZ                     0  3 := 0000000F
  364 asig.: BDMEM                   0  7 := 00000001
  365 asig.: ADMEM                   0  7 := 00000001
  366 asig.: BIMEM                   0  7 := 00000000
  367 asig.: AIMEM                   0  7 := 00000000
  368 asig.: Y_BUS                   0 16 := 00000001
  375 test : MICROMEM               42 42 == 00000001
  377 test : MICROMEM               27 27 == 00000000
  379 asig.: D_LATCH                 0  7 := 00000000
  380 asig.: D_LATCH                 8 15 := 00000001
  384 test : MICROMEM               35 35 == 00000000
  384 asig.: ADRESS_REG              0 15 := 00000001
  706 test : MICROMEM               20 22 == 00000004
  735 test : MICROMEM               43 43 == 00000001
  740 test : MICROMEM               32 32 == 00000001
  742 test : MICROMEM               30 30 == 00000001
  744 test : MICROMEM               36 36 == 00000000
  748 test : MICROMEM               44 47 == 0000000E
  758 asig.: CC                      0  0 := 00000001
  781 test : MICROMEM               48 51 == 00000002
  799 asig.: MICRO_PC                0 11 := 000001D3

3. paso : fin normal
SIM> F
FIN: está seguro ? (S/N)S
Fin de la simulación
Tiempo de procesado :      15.3 segundos
```

```
Ejecución paso a paso de la instrucción:

          LXI DE, 1D1E


SIMULADOR SILOSS. Versión 2.4 DPTO. ELECTRONICA. MARZO 1.989
============================================================

*****EJECUCION:

Para visualizar el menú pulsar "?"


Lectura de definiciones
Lectura del micro-programa de la memoria MICROMEM
SIM> IM MEM_PROG 0000 11,1E,1D
SIM> E6
1. paso : fin normal
SIM> R
ADRESS_REG            = 00000000
ADR_MAPPING           = 00000000
ADR_MEM               = 00000000
DATA_REG              = 00000000
INPUT_DATA            = 00000000
MASK_REG              = 00000007
MICRO_PC              = 00000006
REG_INST              = 00000000
STATUS_REG            = 00000000
UNO                   = 00000001
SIM> E2
2. paso : fin normal
SIM> R
ADRESS_REG            = 00000000
ADR_MAPPING           = 00000011
ADR_MEM               = 00000000
DATA_REG              = 00000000
INPUT_DATA            = 00000011
MASK_REG              = 00000007
MICRO_PC              = 00000008
REG_INST              = 00000011
STATUS_REG            = 00000000
UNO                   = 00000001
SIM> ED
paso 1.

  236 test : MICROMEM            26 26 == 00000001
  238 test : MICROMEM            24 25 == 00000003
  264 asig.: AIZ                  0  3 := 0000000F
  265 asig.: ADER                 0  3 := 0000000F
  266 asig.: BIZ                  0  3 := 0000000F
  267 asig.: BDER                 0  3 := 0000000F
  271 asig.: A_LATCH              0  7 := 00000000
  272 asig.: A_LATCH              8 15 := 00000000
  273 asig.: B_LATCH              0  7 := 00000000
  274 asig.: B_LATCH              8 15 := 00000000
  278 test : MICROMEM             8 10 == 00000004
  288 asig.: ALU_R                0 15 := 00000000
```

```
  289 asig.: ALU_S                0 15 := 00000000
  302 test : MICROMEM            15 15 == 00000000
  304 asig.: CN                   0  0 := 00000001
  308 test : MICROMEM            12 14 == 00000000
  310 asig.: TEMP                 0 16 := 00000000
  311 test : CN                   0  0 == 00000001
  313 asig.: SALIDA_ALU           0 16 := 00000001
  348 test : MICROMEM            37 38 == 00000000
  350 test : MICROMEM            16 18 == 00000003
  362 asig.: ADER                 0  3 := 0000000F
  363 asig.: AIZ                  0  3 := 0000000F
  364 asig.: BDMEM                0  7 := 00000001
  365 asig.: ADMEM                0  7 := 00000001
  366 asig.: BIMEM                0  7 := 00000000
  367 asig.: AIMEM                0  7 := 00000000
  368 asig.: Y_BUS                0 16 := 00000001
  375 test : MICROMEM            42 42 == 00000001
  377 test : MICROMEM            27 27 == 00000000
  379 asig.: D_LATCH              0  7 := 00000000
  380 asig.: D_LATCH              8 15 := 00000001
  384 test : MICROMEM            35 35 == 00000000
  384 asig.: ADRESS_REG           0 15 := 00000001
  706 test : MICROMEM            20 22 == 00000004
  735 test : MICROMEM            43 43 == 00000001
  740 test : MICROMEM            32 32 == 00000001
  742 test : MICROMEM            30 30 == 00000001
  744 test : MICROMEM            36 36 == 00000000
  748 test : MICROMEM            44 47 == 0000000E
  758 asig.: CC                   0  0 := 00000001
  781 test : MICROMEM            48 51 == 00000002
  799 asig.: MICRO_PC             0 11 := 0000001F

1. paso : fin normal
SIM> AM AIMEM 0 10
AIMEM                 : AIZ
[00000000]   00
[00000001]   00
[00000002]   00
[00000003]   00
[00000004]   00
[00000005]   00
[00000006]   00
[00000007]   00
[00000008]   00
[00000009]   00
[0000000A]   00
[0000000B]   00
[0000000C]   60
[0000000D]   06
[0000000E]   00
[0000000F]   00
```

```
SIM> AM BDMEM 0 10
BDMEM                  : BDER
[00000000]   00
[00000001]   00
[00000002]   00
[00000003]   00
[00000004]   00
[00000005]   00
[00000006]   00
[00000007]   00
[00000008]   00
[00000009]   00
[0000000A]   00
[0000000B]   00
[0000000C]   06
[0000000D]   60
[0000000E]   00
[0000000F]   01
SIM> R
ADRESS_REG             = 00000001
ADR_MAPPING            = 00000011
ADR_MEM                = 00000000
DATA_REG               = 00000000
INPUT_DATA             = 00000011
MASK_REG               = 00000007
MICRO_PC               = 0000001F
REG_INST               = 00000011
STATUS_REG             = 00000000
UNO                    = 00000001
SIM> ED
paso 1.

  236 test : MICROMEM              26 26 == 00000001
  238 test : MICROMEM              24 25 == 00000003
  264 asig.: AIZ                    0  3 := 0000000F
  265 asig.: ADER                   0  3 := 0000000F
  266 asig.: BIZ                    0  3 := 0000000F
  267 asig.: BDER                   0  3 := 0000000F
  271 asig.: A_LATCH                0  7 := 00000001
  272 asig.: A_LATCH                8 15 := 00000000
  273 asig.: B_LATCH                0  7 := 00000001
  274 asig.: B_LATCH                8 15 := 00000000
  278 test : MICROMEM               8 10 == 00000004
  288 asig.: ALU_R                  0 15 := 00000000
  289 asig.: ALU_S                  0 15 := 00000001
  302 test : MICROMEM              15 15 == 00000000
  304 asig.: CN                     0  0 := 00000001
  308 test : MICROMEM              12 14 == 00000000
  310 asig.: TEMP                   0 16 := 00000001
  311 test : CN                     0  0 == 00000001
  313 asig.: SALIDA_ALU             0 16 := 00000002
  348 test : MICROMEM              37 38 == 00000000
  350 test : MICROMEM              16 18 == 00000003
  362 asig.: ADER                   0  3 := 0000000F
```

```
  363 asig.: AIZ                  0  3 := 0000000F
  364 asig.: BDMEM                0  7 := 00000002
  365 asig.: ADMEM                0  7 := 00000002
  366 asig.: BIMEM                0  7 := 00000000
  367 asig.: AIMEM                0  7 := 00000000
  368 asig.: Y_BUS                0 16 := 00000002
  375 test : MICROMEM            42 42 == 00000001
  377 test : MICROMEM            27 27 == 00000000
  379 asig.: D_LATCH              0  7 := 00000000
  380 asig.: D_LATCH              8 15 := 00000002
  384 test : MICROMEM            35 35 == 00000000
  384 asig.: ADRESS_REG           0 15 := 00000002
  706 test : MICROMEM            20 22 == 00000004
  735 test : MICROMEM            43 43 == 00000001
  740 test : MICROMEM            32 32 == 00000001
  742 test : MICROMEM            30 30 == 00000001
  744 test : MICROMEM            36 36 == 00000001
  744 asig.: ADR_MEM              0 15 := 00000001
  748 test : MICROMEM            44 47 == 0000000E
  758 asig.: CC                   0  0 := 00000001
  781 test : MICROMEM            48 51 == 0000000E
  782 asig.: MICRO_PC             0 11 := 00000020

1. paso : fin normal
SIM> AM AIMEM F
AIMEM                 : AIZ
[0000000F]   00
SIM> AM ADMEM F
ADMEM                 : ADER
[0000000F]   02
SIM> ED
paso 1.

  236 test : MICROMEM            26 26 == 00000000
  391 test : MICROMEM            24 25 == 00000000
  393 asig.: AIZ                  0  2 := 00000001
  394 asig.: AIZ                  3  3 := 00000000
  395 asig.: ADER                 0  0 := 00000000
  396 asig.: ADER                 1  3 := 00000000
  397 asig.: BIZ                  0  2 := 00000002
  398 asig.: BIZ                  3  3 := 00000000
  399 asig.: BDER                 0  0 := 00000001
  400 asig.: BDER                 1  3 := 00000001
  430 test : MICROMEM            37 38 == 00000000
  440 test : MICROMEM            27 27 == 00000000
  446 asig.: DI_LATCH             0  7 := 00000000
  447 asig.: DD_LATCH             0  7 := 00000000
  452 test : MICROMEM             8 10 == 00000003
  460 asig.: ALU_IR               0  7 := 00000000
  461 asig.: ALU_IS               0  7 := 00000000
  462 asig.: ALU_DR               0  7 := 00000000
  463 asig.: ALU_DS               0  7 := 00000000
  486 test : MICROMEM            15 15 == 00000000
  492 asig.: CN1                  0  0 := 00000000
```

```
 493 asig.: CN2                    0  0 := 00000000
 498 test : MICROMEM              12 14 == 00000003
 548 asig.: SALIDA_ALUI            0  8 := 00000000
 549 asig.: SALIDA_ALUD            0  8 := 00000000
 572 test : MICROMEM              37 38 == 00000000
 573 test : MICROMEM              16 18 == 00000001
 575 asig.: BUS_I                  0  8 := 00000000
 576 asig.: BUS_D                  0  8 := 00000000
 678 test : MICROMEM              42 42 == 00000001
 680 test : MICROMEM              34 34 == 00000001
 684 asig.: LAT_I                  0  7 := 00000000
 685 asig.: LAT_D                  0  7 := 00000000
 688 asig.: PAR_TEST               0  3 := 00000000
 690 test : SALIDA_ALUI            0  0 == 00000000
 691 test : SALIDA_ALUI            1  1 == 00000000
 692 test : SALIDA_ALUI            2  2 == 00000000
 693 test : SALIDA_ALUI            3  3 == 00000000
 694 test : SALIDA_ALUI            4  4 == 00000000
 695 test : SALIDA_ALUI            5  5 == 00000000
 696 test : SALIDA_ALUI            6  6 == 00000000
 697 test : SALIDA_ALUI            7  7 == 00000000
 699 test : PAR_TEST               0  0 == 00000000
 699 asig.: PARITY_REG             0  0 := 00000001
 700 test : PAR_TEST               0  0 == 00000000
 706 test : MICROMEM              20 22 == 00000004
 735 test : MICROMEM              43 43 == 00000001
 740 test : MICROMEM              32 32 == 00000001
 742 test : MICROMEM              30 30 == 00000000
 742 asig.: INPUT_DATA             0  7 := 0000001E
 744 test : MICROMEM              36 36 == 00000000
 748 test : MICROMEM              44 47 == 0000000E
 758 asig.: CC                     0  0 := 00000001
 781 test : MICROMEM              48 51 == 0000000E
 782 asig.: MICRO_PC               0 11 := 00000021

1. paso : fin normal
SIM> ED
paso 1.

 236 test : MICROMEM              26 26 == 00000000
 391 test : MICROMEM              24 25 == 00000003
 421 asig.: AIZ                    0  3 := 00000007
 422 asig.: ADER                   0  0 := 00000000
 423 asig.: ADER                   1  3 := 00000003
 424 asig.: BIZ                    0  3 := 00000003
 425 asig.: BDER                   0  0 := 00000000
 426 asig.: BDER                   1  3 := 00000001
 430 test : MICROMEM              37 38 == 00000000
 440 test : MICROMEM              27 27 == 00000001
 442 asig.: DI_LATCH               0  7 := 0000001E
 443 asig.: DD_LATCH               0  7 := 0000001E
 452 test : MICROMEM               8 10 == 00000007
 478 asig.: ALU_IR                 0  7 := 0000001E
 479 asig.: ALU_IS                 0  7 := 00000000
```

```
480 asig.: ALU_DR                0  7 := 0000001E
481 asig.: ALU_DS                0  7 := 00000000
486 test : MICROMEM             15 15 == 00000000
492 asig.: CN1                   0  0 := 00000000
493 asig.: CN2                   0  0 := 00000000
498 test : MICROMEM             12 14 == 00000003
548 asig.: SALIDA_ALUI           0  8 := 0000001E
549 asig.: SALIDA_ALUD           0  8 := 0000001E
572 test : MICROMEM             37 38 == 00000000
573 test : MICROMEM             16 18 == 00000003
589 asig.: ADER                  0  3 := 00000002
590 asig.: AIZ                   0  3 := 00000003
591 asig.: BIMEM                 0  7 := 0000001E
592 asig.: AIMEM                 0  7 := 0000001E
593 asig.: BDMEM                 0  7 := 0000001E
594 asig.: ADMEM                 0  7 := 0000001E
595 asig.: BUS_I                 0  8 := 0000001E
596 asig.: BUS_D                 0  8 := 0000001E
678 test : MICROMEM             42 42 == 00000001
680 test : MICROMEM             34 34 == 00000001
684 asig.: LAT_I                 0  7 := 0000001E
685 asig.: LAT_D                 0  7 := 0000001E
688 asig.: PAR_TEST              0  3 := 00000000
690 test : SALIDA_ALUI           0  0 == 00000000
691 test : SALIDA_ALUI           1  1 == 00000001
691 asig.: PAR_TEST              0  3 := 00000001
692 test : SALIDA_ALUI           2  2 == 00000001
692 asig.: PAR_TEST              0  3 := 00000002
693 test : SALIDA_ALUI           3  3 == 00000001
693 asig.: PAR_TEST              0  3 := 00000003
694 test : SALIDA_ALUI           4  4 == 00000001
694 asig.: PAR_TEST              0  3 := 00000004
695 test : SALIDA_ALUI           5  5 == 00000000
696 test : SALIDA_ALUI           6  6 == 00000000
697 test : SALIDA_ALUI           7  7 == 00000000
699 test : PAR_TEST              0  0 == 00000000
699 asig.: PARITY_REG            0  0 := 00000001
700 test : PAR_TEST              0  0 == 00000000
706 test : MICROMEM             20 22 == 00000004
735 test : MICROMEM             43 43 == 00000001
740 test : MICROMEM             32 32 == 00000001
742 test : MICROMEM             30 30 == 00000001
744 test : MICROMEM             36 36 == 00000000
748 test : MICROMEM             44 47 == 0000000E
758 asig.: CC                    0  0 := 00000001
781 test : MICROMEM             48 51 == 0000000E
782 asig.: MICRO_PC              0 11 := 00000022

1. paso : fin normal
```

```
SIM> AM BIMEM 0 10
BIMEM                : BIZ
[00000000]   00
[00000001]   00
[00000002]   00
[00000003]   1E
[00000004]   00
[00000005]   00
[00000006]   00
[00000007]   00
[00000008]   00
[00000009]   00
[0000000A]   00
[0000000B]   00
[0000000C]   60
[0000000D]   06
[0000000E]   00
[0000000F]   00
SIM> AM ADMEM 0 10
ADMEM                : ADER
[00000000]   00
[00000001]   00
[00000002]   1E
[00000003]   00
[00000004]   00
[00000005]   00
[00000006]   00
[00000007]   00
[00000008]   00
[00000009]   00
[0000000A]   00
[0000000B]   00
[0000000C]   06
[0000000D]   60
[0000000E]   00
[0000000F]   02
SIM> R
ADRESS_REG           = 00000002
ADR_MAPPING          = 00000011
ADR_MEM              = 00000001
DATA_REG             = 00000000
INPUT_DATA           = 0000001E
MASK_REG             = 00000007
MICRO_PC             = 00000022
REG_INST             = 00000011
STATUS_REG           = 00000000
UNO                  = 00000001
SIM> E
1. paso : fin normal
SIM> R
ADRESS_REG           = 00000003
ADR_MAPPING          = 00000011
ADR_MEM              = 00000002
DATA_REG             = 00000000
```

```
INPUT_DATA              = 0000001E
MASK_REG                = 00000007
MICRO_PC                = 00000023
REG_INST                = 00000011
STATUS_REG              = 00000000
UNO                     = 00000001
SIM> ED
paso 1.

  236 test : MICROMEM              26 26 == 00000000
  391 test : MICROMEM              24 25 == 00000000
  393 asig.: AIZ                    0  2 := 00000001
  394 asig.: AIZ                    3  3 := 00000000
  395 asig.: ADER                   0  0 := 00000000
  396 asig.: ADER                   1  3 := 00000000
  397 asig.: BIZ                    0  2 := 00000002
  398 asig.: BIZ                    3  3 := 00000000
  399 asig.: BDER                   0  0 := 00000001
  400 asig.: BDER                   1  3 := 00000001
  430 test : MICROMEM              37 38 == 00000000
  440 test : MICROMEM              27 27 == 00000000
  446 asig.: DI_LATCH               0  7 := 0000001E
  447 asig.: DD_LATCH               0  7 := 0000001E
  452 test : MICROMEM               8 10 == 00000003
  460 asig.: ALU_IR                 0  7 := 00000000
  461 asig.: ALU_IS                 0  7 := 00000000
  462 asig.: ALU_DR                 0  7 := 00000000
  463 asig.: ALU_DS                 0  7 := 00000000
  486 test : MICROMEM              15 15 == 00000000
  492 asig.: CN1                    0  0 := 00000000
  493 asig.: CN2                    0  0 := 00000000
  498 test : MICROMEM              12 14 == 00000003
  548 asig.: SALIDA_ALUI            0  8 := 00000000
  549 asig.: SALIDA_ALUD            0  8 := 00000000
  572 test : MICROMEM              37 38 == 00000000
  573 test : MICROMEM              16 18 == 00000001
  575 asig.: BUS_I                  0  8 := 00000000
  576 asig.: BUS_D                  0  8 := 00000000
  678 test : MICROMEM              42 42 == 00000001
  680 test : MICROMEM              34 34 == 00000001
  684 asig.: LAT_I                  0  7 := 00000000
  685 asig.: LAT_D                  0  7 := 00000000
  688 asig.: PAR_TEST               0  3 := 00000000
  690 test : SALIDA_ALUI            0  0 == 00000000
  691 test : SALIDA_ALUI            1  1 == 00000000
  692 test : SALIDA_ALUI            2  2 == 00000000
  693 test : SALIDA_ALUI            3  3 == 00000000
  694 test : SALIDA_ALUI            4  4 == 00000000
  695 test : SALIDA_ALUI            5  5 == 00000000
  696 test : SALIDA_ALUI            6  6 == 00000000
  697 test : SALIDA_ALUI            7  7 == 00000000
  699 test : PAR_TEST               0  0 == 00000000
  699 asig.: PARITY_REG             0  0 := 00000001
  700 test : PAR_TEST               0  0 == 00000000
```

```
  706 test : MICROMEM              20 22 == 00000004
  735 test : MICROMEM              43 43 == 00000001
  740 test : MICROMEM              32 32 == 00000001
  742 test : MICROMEM              30 30 == 00000000
  742 asig.: INPUT_DATA             0  7 := 0000001D
  744 test : MICROMEM              36 36 == 00000000
  748 test : MICROMEM              44 47 == 0000000E
  758 asig.: CC                     0  0 := 00000001
  781 test : MICROMEM              48 51 == 0000000E
  782 asig.: MICRO_PC               0 11 := 00000024

1. paso : fin normal
SIM> R
ADRESS_REG            = 00000003
ADR_MAPPING           = 00000011
AIMEM                 = 00000002
DATA_REG              = 00000000
INPUT_DATA            = 0000001D
MASK_REG              = 00000007
MICRO_PC              = 00000024
REG_INST              = 00000011
STATUS_REG            = 00000000
UNO                   = 00000001
SIM> AM AIMEM 0 10
AIMEM                 : AIZ
[00000000]   00
[00000001]   00
[00000002]   00
[00000003]   1E
[00000004]   00
[00000005]   00
[00000006]   00
[00000007]   00
[00000008]   00
[00000009]   00
[0000000A]   00
[0000000B]   00
[0000000C]   60
[0000000D]   06
[0000000E]   00
[0000000F]   00
SIM> AM BDMEM 0 10
BDMEM                 : BDER
[00000000]   00
[00000001]   00
[00000002]   1E
[00000003]   00
[00000004]   00
[00000005]   00
[00000006]   00
[00000007]   00
[00000008]   00
[00000009]   00
[0000000A]   00
```

```
[0000000B]   00
[0000000C]   06
[0000000D]   60
[0000000E]   00
[0000000F]   03

SIM> FIN: está seguro ? (S/N) Fin de la simulación
Tiempo de procesado :     14.7 segundos

                  SIMULADOR   RTL  SILOSS
                      EPFL - DET(UPC)
```

Ejecución para el grupo de Stack, E/S y Control de
máquina del programa:

```
            LXI SP, 20C2
            CALL 1B07
            SPHL
      1B07: RIM
            RET
```

```
SIMULADOR SILOSS. Versión 2.4 DPTO. ELECTRONICA. MARZO 1.989
============================================================

*****EJECUCION:

Para visualizar el menú pulsar "?"


Lectura de definiciones
 ectura del micro-programa de la memoria MICROMEM
SIM> IM MEM_PROG 0000 31,C2,20,CD,07,1B,F9
SIM> IM MEM_PROG 1B07 20,C9
SIM> E9
9. paso : fin normal
SIM> A MICRO_PC
MICRO_PC              = 02B
SIM> E8
8. paso : fin normal
SIM> A MICRO_PC
MICRO_PC              = 121
SIM> ED2
paso 1.

  236 test : MICROMEM              26 26 == 00000001
  238 test : MICROMEM              24 25 == 00000003
  264 asig.: AIZ                    0  3 := 00000009
  265 asig.: ADER                   0  3 := 00000009
  266 asig.: BIZ                    0  3 := 00000009
  267 asig.: BDER                   0  3 := 00000009
  271 asig.: A_LATCH                0  7 := 000000C2
  272 asig.: A_LATCH                8 15 := 00000020
  273 asig.: B_LATCH                0  7 := 000000C2
  274 asig.: B_LATCH                8 15 := 00000020
  278 test : MICROMEM               8 10 == 00000004
  288 asig.: ALU_R                  0 15 := 00000000
  289 asig.: ALU_S                  0 15 := 000020C2
  302 test : MICROMEM              15 15 == 00000000
  304 asig.: CN                     0  0 := 00000000
  308 test : MICROMEM              12 14 == 00000001
  318 asig.: TEMP                   0 16 := 000020C2
  319 test : CN                     0  0 == 00000000
  320 asig.: SALIDA_ALU             0 16 := 000020C1
  348 test : MICROMEM              37 38 == 00000000
  350 test : MICROMEM              16 18 == 00000003
  362 asig.: ADER                   0  3 := 00000009
  363 asig.: AIZ                    0  3 := 00000009
  364 asig.: BDMEM                  0  7 := 000000C1
  365 asig.: ADMEM                  0  7 := 000000C1
  366 asig.: BIMEM                  0  7 := 00000020
  367 asig.: AIMEM                  0  7 := 00000020
```

```
368 asig.: Y_BUS                0 16 := 000020C1
375 test : MICROMEM            42 42 == 00000001
377 test : MICROMEM            27 27 == 00000000
379 asig.: D_LATCH              0  7 := 00000020
380 asig.: D_LATCH              8 15 := 000000C1
384 test : MICROMEM            35 35 == 00000001
706 test : MICROMEM            20 22 == 00000004
735 test : MICROMEM            43 43 == 00000001
740 test : MICROMEM            32 32 == 00000001
742 test : MICROMEM            30 30 == 00000001
744 test : MICROMEM            36 36 == 00000000
748 test : MICROMEM            44 47 == 0000000E
758 asig.: CC                   0  0 := 00000001
781 test : MICROMEM            48 51 == 0000000E
782 asig.: MICRO_PC             0 11 := 00000122
```

paso 2.

```
236 test : MICROMEM            26 26 == 00000001
238 test : MICROMEM            24 25 == 00000003
264 asig.: AIZ                  0  3 := 0000000F
265 asig.: ADER                 0  3 := 0000000F
266 asig.: BIZ                  0  3 := 0000000F
267 asig.: BDER                 0  3 := 0000000F
271 asig.: A_LATCH              0  7 := 00000004
272 asig.: A_LATCH              8 15 := 00000000
273 asig.: B_LATCH              0  7 := 00000004
274 asig.: B_LATCH              8 15 := 00000000
278 test : MICROMEM            8 10 == 00000004
288 asig.: ALU_R                0 15 := 00000000
289 asig.: ALU_S                0 15 := 00000004
302 test : MICROMEM           15 15 == 00000000
304 asig.: CN                   0  0 := 00000001
308 test : MICROMEM           12 14 == 00000000
310 asig.: TEMP                 0 16 := 00000004
311 test : CN                   0  0 == 00000001
313 asig.: SALIDA_ALU           0 16 := 00000005
348 test : MICROMEM           37 38 == 00000000
350 test : MICROMEM           16 18 == 00000003
362 asig.: ADER                 0  3 := 0000000F
363 asig.: AIZ                  0  3 := 0000000F
364 asig.: BDMEM                0  7 := 00000005
365 asig.: ADMEM                0  7 := 00000005
366 asig.: BIMEM                0  7 := 00000000
367 asig.: AIMEM                0  7 := 00000000
368 asig.: Y_BUS                0 16 := 00000005
375 test : MICROMEM           42 42 == 00000001
377 test : MICROMEM           27 27 == 00000000
379 asig.: D_LATCH              0  7 := 00000000
380 asig.: D_LATCH              8 15 := 00000005
384 test : MICROMEM           35 35 == 00000000
384 asig.: ADRESS_REG           0 15 := 00000005
706 test : MICROMEM           20 22 == 00000004
735 test : MICROMEM           43 43 == 00000001
740 test : MICROMEM           32 32 == 00000001
```

```
  742 test : MICROMEM              30 30 == 00000001
  744 test : MICROMEM              36 36 == 00000001
  744 asig.: ADR_MEM                0 15 := 00000004
  748 test : MICROMEM              44 47 == 0000000E
  758 asig.: CC                     0  0 := 00000001
  781 test : MICROMEM              48 51 == 0000000E
  782 asig.: MICRO_PC               0 11 := 00000123

2. paso : fin normal
SIM> E
1. paso : fin normal
SIM> ED
paso 1.

  236 test : MICROMEM              26 26 == 00000000
  391 test : MICROMEM              24 25 == 00000003
  421 asig.: AIZ                    0  3 := 00000007
  422 asig.: ADER                   0  0 := 00000000
  423 asig.: ADER                   1  3 := 00000003
  424 asig.: BIZ                    0  3 := 0000000B
  425 asig.: BDER                   0  0 := 00000000
  426 asig.: BDER                   1  3 := 00000005
  430 test : MICROMEM              37 38 == 00000000
  440 test : MICROMEM              27 27 == 00000001
  442 asig.: DI_LATCH               0  7 := 00000007
  443 asig.: DD_LATCH               0  7 := 00000007
  452 test : MICROMEM               8 10 == 00000007
  478 asig.: ALU_IR                 0  7 := 00000007
  479 asig.: ALU_IS                 0  7 := 00000000
  480 asig.: ALU_DR                 0  7 := 00000007
  481 asig.: ALU_DS                 0  7 := 00000000
  486 test : MICROMEM              15 15 == 00000000
  492 asig.: CN1                    0  0 := 00000000
  493 asig.: CN2                    0  0 := 00000000
  498 test : MICROMEM              12 14 == 00000003
  548 asig.: SALIDA_ALUI            0  8 := 00000007
  549 asig.: SALIDA_ALUD            0  8 := 00000007
  572 test : MICROMEM              37 38 == 00000000
  573 test : MICROMEM              16 18 == 00000003
  589 asig.: ADER                   0  3 := 0000000A
  590 asig.: AIZ                    0  3 := 0000000B
  591 asig.: BIMEM                  0  7 := 00000007
  592 asig.: AIMEM                  0  7 := 00000007
  593 asig.: BDMEM                  0  7 := 00000007
  594 asig.: ADMEM                  0  7 := 00000007
  595 asig.: BUS_I                  0  8 := 00000007
  596 asig.: BUS_D                  0  8 := 00000007
  678 test : MICROMEM              42 42 == 00000001
  680 test : MICROMEM              34 34 == 00000001
  684 asig.: LAT_I                  0  7 := 00000007
  685 asig.: LAT_D                  0  7 := 00000007
  688 asig.: PAR_TEST               0  3 := 00000000
  690 test : SALIDA_ALUI            0  0 == 00000001
  690 asig.: PAR_TEST               0  3 := 00000001
  691 test : SALIDA_ALUI            1  1 == 00000001
```

```
 691 asig.: PAR_TEST              0  3 := 00000002
 692 test : SALIDA_ALUI           2  2 == 00000001
 692 asig.: PAR_TEST              0  3 := 00000003
 693 test : SALIDA_ALUI           3  3 == 00000000
 694 test : SALIDA_ALUI           4  4 == 00000000
 695 test : SALIDA_ALUI           5  5 == 00000000
 696 test : SALIDA_ALUI           6  6 == 00000000
 697 test : SALIDA_ALUI           7  7 == 00000000
 699 test : PAR_TEST              0  0 == 00000001
 700 test : PAR_TEST              0  0 == 00000001
 700 asig.: PARITY_REG            0  0 := 00000000
 706 test : MICROMEM             20 22 == 00000004
 735 test : MICROMEM             43 43 == 00000001
 740 test : MICROMEM             32 32 == 00000001
 742 test : MICROMEM             30 30 == 00000001
 744 test : MICROMEM             36 36 == 00000000
 748 test : MICROMEM             44 47 == 0000000E
 758 asig.: CC                    0  0 := 00000001
 781 test : MICROMEM             48 51 == 0000000E
 782 asig.: MICRO_PC              0 11 := 00000125

1. paso : fin normal
SIM> E3
3. paso : fin normal
SIM> ED
paso 1.

 236 test : MICROMEM             26 26 == 00000001
 238 test : MICROMEM             24 25 == 00000003
 264 asig.: AIZ                   0  3 := 00000009
 265 asig.: ADER                  0  3 := 00000009
 266 asig.: BIZ                   0  3 := 00000009
 267 asig.: BDER                  0  3 := 00000009
 271 asig.: A_LATCH               0  7 := 000000C1
 272 asig.: A_LATCH               8 15 := 00000020
 273 asig.: B_LATCH               0  7 := 000000C1
 274 asig.: B_LATCH               8 15 := 00000020
 278 test : MICROMEM              8 10 == 00000004
 288 asig.: ALU_R                 0 15 := 00000000
 289 asig.: ALU_S                 0 15 := 000020C1
 302 test : MICROMEM             15 15 == 00000000
 304 asig.: CN                    0  0 := 00000000
 308 test : MICROMEM             12 14 == 00000001
 318 asig.: TEMP                  0 16 := 000020C1
 319 test : CN                    0  0 == 00000000
 320 asig.: SALIDA_ALU            0 16 := 000020C0
 348 test : MICROMEM             37 38 == 00000000
 350 test : MICROMEM             16 18 == 00000002
 353 asig.: ADER                  0  3 := 00000009
 354 asig.: AIZ                   0  3 := 00000009
 355 asig.: BDMEM                 0  7 := 000000C0
 356 asig.: ADMEM                 0  7 := 000000C0
 357 asig.: BIMEM                 0  7 := 00000020
 358 asig.: AIMEM                 0  7 := 00000020
 359 asig.: Y_BUS                 0 15 := 000020C1
```

```
  375 test : MICROMEM          42 42 == 00000001
  377 test : MICROMEM          27 27 == 00000000
  379 asig.: D_LATCH            0  7 := 00000020
  380 asig.: D_LATCH            8 15 := 000000C1
  384 test : MICROMEM          35 35 == 00000000
  384 asig.: ADRESS_REG         0 15 := 000020C1
  706 test : MICROMEM          20 22 == 00000004
  735 test : MICROMEM          43 43 == 00000001
  740 test : MICROMEM          32 32 == 00000001
  742 test : MICROMEM          30 30 == 00000001
  744 test : MICROMEM          36 36 == 00000000
  748 test : MICROMEM          44 47 == 0000000E
  758 asig.: CC                 0  0 := 00000001
  781 test : MICROMEM          48 51 == 0000000E
  782 asig.: MICRO_PC           0 11 := 00000129

1. paso : fin normal
SIM> E
1. paso : fin normal
SIM> ED
paso 1.

  236 test : MICROMEM          26 26 == 00000001
  238 test : MICROMEM          24 25 == 00000003
  264 asig.: AIZ                0  3 := 00000009
  265 asig.: ADER               0  3 := 00000009
  266 asig.: BIZ                0  3 := 00000009
  267 asig.: BDER               0  3 := 00000009
  271 asig.: A_LATCH            0  7 := 000000C0
  272 asig.: A_LATCH            8 15 := 00000020
  273 asig.: B_LATCH            0  7 := 000000C0
  274 asig.: B_LATCH            8 15 := 00000020
  278 test : MICROMEM           8 10 == 00000004
  288 asig.: ALU_R              0 15 := 00000000
  289 asig.: ALU_S              0 15 := 000020C0
  302 test : MICROMEM          15 15 == 00000000
  304 asig.: CN                 0  0 := 00000000
  308 test : MICROMEM          12 14 == 00000003
  333 asig.: SALIDA_ALU         0 16 := 000020C0
  348 test : MICROMEM          37 38 == 00000000
  350 test : MICROMEM          16 18 == 00000003
  362 asig.: ADER               0  3 := 00000009
  363 asig.: AIZ                0  3 := 00000009
  364 asig.: BDMEM              0  7 := 000000C0
  365 asig.: ADMEM              0  7 := 000000C0
  366 asig.: BIMEM              0  7 := 00000020
  367 asig.: AIMEM              0  7 := 00000020
  368 asig.: Y_BUS              0 16 := 000020C0
  375 test : MICROMEM          42 42 == 00000001
  377 test : MICROMEM          27 27 == 00000000
  379 asig.: D_LATCH            0  7 := 00000020
  380 asig.: D_LATCH            8 15 := 000000C0
  384 test : MICROMEM          35 35 == 00000000
  384 asig.: ADRESS_REG         0 15 := 000020C0
  706 test : MICROMEM          20 22 == 00000004
```

```
  735 test : MICROMEM              43 43 == 00000001
  740 test : MICROMEM              32 32 == 00000000
  740 asig.: MEM_PROG               0  7 := 00000000
  742 test : MICROMEM              30 30 == 00000001
  744 test : MICROMEM              36 36 == 00000000
  748 test : MICROMEM              44 47 == 0000000E
  758 asig.: CC                     0  0 := 00000001
  781 test : MICROMEM              48 51 == 0000000E
  782 asig.: MICRO_PC               0 11 := 0000012B

1. paso : fin normal
SIM> E2
2. paso : fin normal

SIM> A MICRO_PC
MICRO_PC              = 006
SIM> E3
3. paso : fin normal
SIM> A MICRO_PC
MICRO_PC              = 1D4
SIM> AM AIMEM 7
AIMEM                 : AIZ
[00000007]   00
SIM> ED2
paso 1.

  236 test : MICROMEM              26 26 == 00000000
  391 test : MICROMEM              24 25 == 00000000
  393 asig.: AIZ                    0  2 := 00000000
  394 asig.: AIZ                    3  3 := 00000000
  395 asig.: ADER                   0  0 := 00000001
  396 asig.: ADER                   1  3 := 00000000
  397 asig.: BIZ                    0  2 := 00000004
  398 asig.: BIZ                    3  3 := 00000000
  399 asig.: BDER                   0  0 := 00000001
  400 asig.: BDER                   1  3 := 00000002
  430 test : MICROMEM              37 38 == 00000002
  436 asig.: BUS_I                  0  8 := 00000007
  437 asig.: BUS_D                  0  8 := 00000007
  440 test : MICROMEM              27 27 == 00000000
  446 asig.: DI_LATCH               0  7 := 00000000
  447 asig.: DD_LATCH               0  7 := 00000000
  452 test : MICROMEM               8 10 == 00000003
  460 asig.: ALU_IR                 0  7 := 00000000
  461 asig.: ALU_IS                 0  7 := 00000000
  462 asig.: ALU_DR                 0  7 := 00000000
  463 asig.: ALU_DS                 0  7 := 00000000
  486 test : MICROMEM              15 15 == 00000000
  492 asig.: CN1                    0  0 := 00000000
  493 asig.: CN2                    0  0 := 00000000
  498 test : MICROMEM              12 14 == 00000003
  548 asig.: SALIDA_ALUI            0  8 := 00000000
  549 asig.: SALIDA_ALUD            0  8 := 00000000
  572 test : MICROMEM              37 38 == 00000002
  669 asig.: ADER                   0  3 := 00000005
```

```
670 asig.: AIZ                0  3 := 00000004
671 asig.: BIMEM              0  7 := 00000000
672 asig.: AIMEM              0  7 := 00000000
673 asig.: BDMEM              0  7 := 00000000
674 asig.: ADMEM              0  7 := 00000000
678 test : MICROMEM          42 42 == 00000001
680 test : MICROMEM          34 34 == 00000001
684 asig.: LAT_I              0  7 := 00000007
685 asig.: LAT_D              0  7 := 00000007
688 asig.: PAR_TEST           0  3 := 00000000
690 test : SALIDA_ALUI        0  0 == 00000000
691 test : SALIDA_ALUI        1  1 == 00000000
692 test : SALIDA_ALUI        2  2 == 00000000
693 test : SALIDA_ALUI        3  3 == 00000000
694 test : SALIDA_ALUI        4  4 == 00000000
695 test : SALIDA_ALUI        5  5 == 00000000
696 test : SALIDA_ALUI        6  6 == 00000000
697 test : SALIDA_ALUI        7  7 == 00000000
699 test : PAR_TEST           0  0 == 00000000
699 asig.: PARITY_REG         0  0 := 00000001
700 test : PAR_TEST           0  0 == 00000000
706 test : MICROMEM          20 22 == 00000004
735 test : MICROMEM          43 43 == 00000001
740 test : MICROMEM          32 32 == 00000001
742 test : MICROMEM          30 30 == 00000001
744 test : MICROMEM          36 36 == 00000000
748 test : MICROMEM          44 47 == 0000000E
758 asig.: CC                 0  0 := 00000001
781 test : MICROMEM          48 51 == 0000000E
782 asig.: MICRO_PC           0 11 := 000001D5
```

paso 2.

```
236 test : MICROMEM          26 26 == 00000000
391 test : MICROMEM          24 25 == 00000003
421 asig.: AIZ                0  3 := 00000007
422 asig.: ADER               0  0 := 00000000
423 asig.: ADER               1  3 := 00000003
424 asig.: BIZ                0  3 := 00000007
425 asig.: BDER               0  0 := 00000000
426 asig.: BDER               1  3 := 00000003
430 test : MICROMEM          37 38 == 00000000
440 test : MICROMEM          27 27 == 00000000
446 asig.: DI_LATCH           0  7 := 00000007
447 asig.: DD_LATCH           0  7 := 00000007
452 test : MICROMEM           8 10 == 00000007
478 asig.: ALU_IR             0  7 := 00000007
479 asig.: ALU_IS             0  7 := 00000000
480 asig.: ALU_DR             0  7 := 00000007
481 asig.: ALU_DS             0  7 := 00000000
486 test : MICROMEM          15 15 == 00000000
492 asig.: CN1                0  0 := 00000000
493 asig.: CN2                0  0 := 00000000
498 test : MICROMEM          12 14 == 00000003
548 asig.: SALIDA_ALUI        0  8 := 00000007
```

```
549 asig.: SALIDA_ALUD          0  8 := 00000007
572 test : MICROMEM            37 38 == 00000000
573 test : MICROMEM            16 18 == 00000003
589 asig.: ADER                 0  3 := 00000006
590 asig.: AIZ                  0  3 := 00000007
591 asig.: BIMEM                0  7 := 00000007
592 asig.: AIMEM                0  7 := 00000007
593 asig.: BDMEM                0  7 := 00000007
594 asig.: ADMEM                0  7 := 00000007
595 asig.: BUS_I                0  8 := 00000007
596 asig.: BUS_D                0  8 := 00000007
678 test : MICROMEM            42 42 == 00000001
680 test : MICROMEM            34 34 == 00000001
684 asig.: LAT_I                0  7 := 00000007
685 asig.: LAT_D                0  7 := 00000007
688 asig.: PAR_TEST             0  3 := 00000000
690 test : SALIDA_ALUI          0  0 == 00000001
690 asig.: PAR_TEST             0  3 := 00000001
691 test : SALIDA_ALUI          1  1 == 00000001
691 asig.: PAR_TEST             0  3 := 00000002
692 test : SALIDA_ALUI          2  2 == 00000001
692 asig.: PAR_TEST             0  3 := 00000003
693 test : SALIDA_ALUI          3  3 == 00000000
694 test : SALIDA_ALUI          4  4 == 00000000
695 test : SALIDA_ALUI          5  5 == 00000000
696 test : SALIDA_ALUI          6  6 == 00000000
697 test : SALIDA_ALUI          7  7 == 00000000
699 test : PAR_TEST             0  0 == 00000001
700 test : PAR_TEST             0  0 == 00000001
700 asig.: PARITY_REG           0  0 := 00000000
706 test : MICROMEM            20 22 == 00000004
735 test : MICROMEM            43 43 == 00000001
740 test : MICROMEM            32 32 == 00000001
742 test : MICROMEM            30 30 == 00000001
744 test : MICROMEM            36 36 == 00000001
744 asig.: ADR_MEM              0 15 := 00001B08
748 test : MICROMEM            44 47 == 0000000F
757 asig.: CC                   0  0 := 00000000
781 test : MICROMEM            48 51 == 00000003
791 test : CC                   0  0 == 00000000
791 asig.: MICRO_PC             0 11 := 00000007
792 test : CC                   0  0 == 00000000

2. paso : fin normal
SIM> AM  AIMEM 7
AIMEM                : AIZ
[00000007]   07
SIM> E2
2. paso : fin normal
SIM> A MICRO_PC
MICRO_PC             = 16D
SIM> E7
7. paso : fin normal
```

```
SIM> AM AIMEM 4 2
AIMEM                : AIZ
[00000004]   00
[00000005]   00
SIM> AM AIMEM 9
AIMEM                : AIZ
[00000009]   20
SIM> AM ADMEM 9
ADMEM                : ADER
[00000009]   C2
SIM> A MICRO_PC
MICRO_PC              = 006
SIM> E3
3. paso : fin normal
SIM> A MICRO_PC
MICRO_PC              = 012
SIM> E
1. paso : fin normal
SIM> AM AIMEM 9
AIMEM                : AIZ
[00000009]   00
SIM> AM ADMEM 9
ADMEM                : ADER
[00000009]   00

SIM> F
FIN: está seguro ? (S/N) Fin de la simulación
Tiempo de procesado :      18.5 segundos

                SIMULADOR   RTL  SILOSS
                    EPFL - DET(UPC)
```

Ejecución para el grupo lógico del programa:

```
          ADI 88
          RAR
          STC
          RLC
          XRI FF
```

```
SIMULADOR SILOSS. Versión 2.4 DPTO. ELECTRONICA. MARZO 1.987
============================================================

*****EJECUCION:

Para visualizar el men  pulsar "?"


Lectura de definiciones
Lectura del micro-programa de la memoria MICROMEM
SIM> IM MEM_PROG 0000 C6,8B,1F,37,07,EE,FF
SIM> E9
9. paso : fin normal
SIM> E2
2. paso : fin normal
SIM> ED
paso 1.

  236 test : MICROMEM              26 26 == 00000000
  391 test : MICROMEM              24 25 == 00000003
  421 asig.: AIZ                    0  3 := 00000007
  422 asig.: ADER                   0  0 := 00000000
  423 asig.: ADER                   1  3 := 00000003
  424 asig.: BIZ                    0  3 := 00000007
  425 asig.: BDER                   0  0 := 00000000
  426 asig.: BDER                   1  3 := 00000003
  430 test : MICROMEM              37 38 == 00000000
  440 test : MICROMEM              27 27 == 00000001
  442 asig.: DI_LATCH               0  7 := 00000088
  443 asig.: DD_LATCH               0  7 := 00000088
  452 test : MICROMEM               8 10 == 00000005
  472 asig.: ALU_IR                 0  7 := 00000088
  473 asig.: ALU_IS                 0  7 := 00000000
  474 asig.: ALU_DR                 0  7 := 00000088
  475 asig.: ALU_DS                 0  7 := 00000000
  486 test : MICROMEM              15 15 == 00000000
  492 asig.: CN1                    0  0 := 00000000
  493 asig.: CN2                    0  0 := 00000000
  498 test : MICROMEM              12 14 == 00000000
  500 asig.: TEMPI                  0  8 := 00000088
  501 asig.: TEMPD                  0  8 := 00000088
  502 asig.: AC                     0  4 := 00000008
  503 test : CN1                    0  0 == 00000000
  504 asig.: SALIDA_ALUI            0  8 := 00000088
  510 test : CN2                    0  0 == 00000000
  511 asig.: SALIDA_ALUD            0  8 := 00000088
  572 test : MICROMEM              37 38 == 00000000
  573 test : MICROMEM              16 18 == 00000003
  589 asig.: ADER                   0  3 := 00000006
  590 asig.: AIZ                    0  3 := 00000007
```

```
  591 asig.: BIMEM                0  7 := 00000088
  592 asig.: AIMEM                0  7 := 00000088
  593 asig.: BDMEM                0  7 := 00000088
  594 asig.: ADMEM                0  7 := 00000088
  595 asig.: BUS_I                0  8 := 00000088
  596 asig.: BUS_D                0  8 := 00000088
  678 test : MICROMEM            42 42 == 00000001
  680 test : MICROMEM            34 34 == 00000001
  684 asig.: LAT_I                0  7 := 00000088
  685 asig.: LAT_D                0  7 := 00000088
  688 asig.: PAR_TEST             0  3 := 00000000
  690 test : SALIDA_ALUI          0  0 == 00000000
  691 test : SALIDA_ALUI          1  1 == 00000000
  692 test : SALIDA_ALUI          2  2 == 00000000
  693 test : SALIDA_ALUI          3  3 == 00000001
  693 asig.: PAR_TEST             0  3 := 00000001
  694 test : SALIDA_ALUI          4  4 == 00000000
  695 test : SALIDA_ALUI          5  5 == 00000000
  696 test : SALIDA_ALUI          6  6 == 00000000
  697 test : SALIDA_ALUI          7  7 == 00000001
  697 asig.: PAR_TEST             0  3 := 00000002
  699 test : PAR_TEST             0  0 == 00000000
  699 asig.: PARITY_REG           0  0 := 00000001
  700 test : PAR_TEST             0  0 == 00000000
  706 test : MICROMEM            20 22 == 00000000
  708 asig.: STATUS_REG           0  0 := 00000000
  709 asig.: STATUS_REG           2  2 := 00000001
  710 asig.: STATUS_REG           4  4 := 00000000
  711 test : SALIDA_ALUI          0  8 == 00000088
  712 asig.: STATUS_REG           7  7 := 00000001
  735 test : MICROMEM            43 43 == 00000001
  740 test : MICROMEM            32 32 == 00000001
  742 test : MICROMEM            30 30 == 00000001
  744 test : MICROMEM            36 36 == 00000001
  744 asig.: ADR_MEM              0 15 := 00000002
  748 test : MICROMEM            44 47 == 0000000F
  757 asig.: CC                   0  0 := 00000000
  781 test : MICROMEM            48 51 == 00000003
  791 test : CC                   0  0 == 00000000
  791 asig.: MICRO_PC             0 11 := 00000007
  792 test : CC                   0  0 == 00000000

1. paso : fin normal
SIM> AM AIMEM 7
AIMEM                : AIZ
[00000007]   38
SIM> E2
2. paso : fin normal
SIM> ED
paso 1.

  236 test : MICROMEM            26 26 == 00000000
  391 test : MICROMEM            24 25 == 00000003
  421 asig.: AIZ                  0  3 := 00000007
```

```
422 asig.: ADER                0  0 := 00000000
423 asig.: ADER                1  3 := 00000003
424 asig.: BIZ                 0  3 := 00000007
425 asig.: BDER                0  0 := 00000000
426 asig.: BDER                1  3 := 00000003
430 test : MICROMEM           37 38 == 00000000
440 test : MICROMEM           27 27 == 00000000
446 asig.: DI_LATCH            0  7 := 00000000
447 asig.: DD_LATCH            0  7 := 00000000
452 test : MICROMEM            8 10 == 00000003
460 asig.: ALU_IR              0  7 := 00000000
461 asig.: ALU_IS              0  7 := 00000088
462 asig.: ALU_DR              0  7 := 00000000
463 asig.: ALU_DS              0  7 := 00000088
486 test : MICROMEM           15 15 == 00000000
492 asig.: CN1                 0  0 := 00000000
493 asig.: CN2                 0  0 := 00000000
498 test : MICROMEM           12 14 == 00000003
548 asig.: SALIDA_ALUI         0  8 := 00000088
549 asig.: SALIDA_ALUD         0  8 := 00000088
572 test : MICROMEM           37 38 == 00000000
573 test : MICROMEM           16 18 == 00000005
599 asig.: BUS_I               0  8 := 00000088
600 asig.: BUS_D               0  8 := 00000088
601 asig.: ADER                0  3 := 00000006
602 asig.: AIZ                 0  3 := 00000007
603 test : MICROMEM           19 19 == 00000001
615 asig.: SALIDA_ALUI         0  8 := 00000044
616 asig.: SALIDA_ALUI         7  7 := 00000000
617 asig.: BIMEM               0  7 := 00000044
618 asig.: AIMEM               0  7 := 00000044
619 asig.: SALIDA_ALUD         0  8 := 00000044
620 asig.: SALIDA_ALUD         7  7 := 00000000
621 asig.: BDMEM               0  7 := 00000044
622 asig.: ADMEM               0  7 := 00000044
678 test : MICROMEM           42 42 == 00000001
680 test : MICROMEM           34 34 == 00000001
684 asig.: LAT_I               0  7 := 00000088
685 asig.: LAT_D               0  7 := 00000088
688 asig.: PAR_TEST            0  3 := 00000000
690 test : SALIDA_ALUI         0  0 == 00000000
691 test : SALIDA_ALUI         1  1 == 00000000
692 test : SALIDA_ALUI         2  2 == 00000001
692 asig.: PAR_TEST            0  3 := 00000001
693 test : SALIDA_ALUI         3  3 == 00000000
694 test : SALIDA_ALUI         4  4 == 00000000
695 test : SALIDA_ALUI         5  5 == 00000000
696 test : SALIDA_ALUI         6  6 == 00000001
696 asig.: PAR_TEST            0  3 := 00000002
697 test : SALIDA_ALUI         7  7 == 00000000
699 test : PAR_TEST            0  0 == 00000000
699 asig.: PARITY_REG          0  0 := 00000001
700 test : PAR_TEST            0  0 == 00000000
706 test : MICROMEM           20 22 == 00000005
```

```
  730 asig.: STATUS_REG          0  0 := 00000000
  735 test : MICROMEM           43 43 == 00000001
  740 test : MICROMEM           32 32 == 00000001
  742 test : MICROMEM           30 30 == 00000001
  744 test : MICROMEM           36 36 == 00000001
  744 asig.: ADR_MEM             0 15 := 00000003
  748 test : MICROMEM           44 47 == 0000000F
  757 asig.: CC                  0  0 := 00000000
  781 test : MICROMEM           48 51 == 00000003
  791 test : CC                  0  0 == 00000000
  791 asig.: MICRO_PC            0 11 := 00000007
  792 test : CC                  0  0 == 00000000

1. paso : fin normal
SIM> AM ADMEM 6
ADMEM                 : ADER
[00000006]   44
SIM> E2
2. paso : fin normal
SIM> ED3
paso 1.

  236 test : MICROMEM           26 26 == 00000000
  391 test : MICROMEM           24 25 == 00000000
  393 asig.: AIZ                 0  2 := 00000007
  394 asig.: AIZ                 3  3 := 00000000
  395 asig.: ADER                0  0 := 00000000
  396 asig.: ADER                1  3 := 00000003
  397 asig.: BIZ                 0  2 := 00000006
  398 asig.: BIZ                 3  3 := 00000000
  399 asig.: BDER                0  0 := 00000001
  400 asig.: BDER                1  3 := 00000003
  430 test : MICROMEM           37 38 == 00000003
  432 asig.: BUS_I               0  8 := 00000001
  433 asig.: BUS_D               0  8 := 00000001
  440 test : MICROMEM           27 27 == 00000000
  446 asig.: DI_LATCH            0  7 := 00000000
  447 asig.: DD_LATCH            0  7 := 00000000
  452 test : MICROMEM            8 10 == 00000003
  460 asig.: ALU_IR              0  7 := 00000000
  461 asig.: ALU_IS              0  7 := 00000000
  462 asig.: ALU_DR              0  7 := 00000000
  463 asig.: ALU_DS              0  7 := 00000000
  486 test : MICROMEM           13 15 == 00000000
  492 asig.: CN1                 0  0 := 00000000
  493 asig.: CN2                 0  0 := 00000000
  498 test : MICROMEM           12 14 == 00000003
  548 asig.: SALIDA_ALUI         0  8 := 00000000
  549 asig.: SALIDA_ALUD         0  8 := 00000000
  572 test : MICROMEM           37 38 == 00000003
  660 asig.: ADER                0  3 := 00000007
  661 asig.: AIZ                 0  3 := 00000006
  662 asig.: BIMEM               0  7 := 00000000
  663 asig.: AIMEM               0  7 := 00000000
```

```
 664 asig.: BDMEM          0  7 := 00000000
 665 asig.: ADMEM          0  7 := 00000000
 678 test : MICROMEM      42 42 == 00000001
 680 test : MICROMEM      34 34 == 00000001
 684 asig.: LAT_I          0  7 := 00000001
 685 asig.: LAT_D          0  7 := 00000001
 688 asig.: PAR_TEST       0  3 := 00000000
 690 test : SALIDA_ALUI    0  0 == 00000000
 691 test : SALIDA_ALUI    1  1 == 00000000
 692 test : SALIDA_ALUI    2  2 == 00000000
 693 test : SALIDA_ALUI    3  3 == 00000000
 694 test : SALIDA_ALUI    4  4 == 00000000
 695 test : SALIDA_ALUI    5  5 == 00000000
 696 test : SALIDA_ALUI    6  6 == 00000000
 697 test : SALIDA_ALUI    7  7 == 00000000
 699 test : PAR_TEST       0  0 == 00000000
 699 asig.: PARITY_REG     0  0 := 00000001
 700 test : PAR_TEST       0  0 == 00000000
 706 test : MICROMEM      20 22 == 00000004
 735 test : MICROMEM      43 43 == 00000001
 740 test : MICROMEM      32 32 == 00000001
 742 test : MICROMEM      30 30 == 00000001
 744 test : MICROMEM      36 36 == 00000000
 748 test : MICROMEM      44 47 == 0000000E
 758 asig.: CC             0  0 := 00000001
 781 test : MICROMEM      48 51 == 0000000E
 782 asig.: MICRO_PC       0 11 := 000000D8
```

paso 2.

```
 236 test : MICROMEM      26 26 == 00000000
 391 test : MICROMEM      24 25 == 00000003
 421 asig.: AIZ            0  3 := 0000000A
 422 asig.: ADER           0  0 := 00000001
 423 asig.: ADER           1  3 := 00000005
 424 asig.: BIZ            0  3 := 0000000A
 425 asig.: BDER           0  0 := 00000001
 426 asig.: BDER           1  3 := 00000005
 430 test : MICROMEM      37 38 == 00000003
 432 asig.: BUS_I          0  8 := 00000000
 433 asig.: BUS_D          0  8 := 00000000
 440 test : MICROMEM      27 27 == 00000000
 446 asig.: DI_LATCH       0  7 := 00000001
 447 asig.: DD_LATCH       0  7 := 00000001
 452 test : MICROMEM       8 10 == 00000007
 478 asig.: ALU_IR         0  7 := 00000001
 479 asig.: ALU_IS         0  7 := 00000000
 480 asig.: ALU_DR         0  7 := 00000001
z481 asig.: ALU_DS         0  7 := 00000000
 486 test : MICROMEM      15 15 == 00000000
 492 asig.: CN1            0  0 := 00000000
 493 asig.: CN2            0  0 := 00000000
 498 test : MICROMEM      12 14 == 00000003
 548 asig.: SALIDA_ALUI    0  8 := 00000001
```

```
549 asig.: SALIDA_ALUD          0  8 := 00000001
572 test : MICROMEM            37 38 == 00000003
660 asig.: ADER                 0  3 := 0000000B
661 asig.: AIZ                  0  3 := 0000000A
662 asig.: BIMEM                0  7 := 00000001
663 asig.: AIMEM                0  7 := 00000001
664 asig.: BDMEM                0  7 := 00000001
665 asig.: ADMEM                0  7 := 00000001
678 test : MICROMEM            42 42 == 00000001
680 test : MICROMEM            34 34 == 00000001
684 asig.: LAT_I                0  7 := 00000000
685 asig.: LAT_D                0  7 := 00000000
688 asig.: PAR_TEST             0  3 := 00000000
690 test : SALIDA_ALUI          0  0 == 00000001
690 asig.: PAR_TEST             0  3 := 00000001
691 test : SALIDA_ALUI          1  1 == 00000000
692 test : SALIDA_ALUI          2  2 == 00000000
693 test : SALIDA_ALUI          3  3 == 00000000
694 test : SALIDA_ALUI          4  4 == 00000000
695 test : SALIDA_ALUI          5  5 == 00000000
696 test : SALIDA_ALUI          6  6 == 00000000
697 test : SALIDA_ALUI          7  7 == 00000000
699 test : PAR_TEST             0  0 == 00000001
700 test : PAR_TEST             0  0 == 00000001
700 asig.: PARITY_REG           0  0 := 00000000
706 test : MICROMEM            20 22 == 00000004
735 test : MICROMEM            43 43 == 00000001
740 test : MICROMEM            32 32 == 00000001
742 test : MICROMEM            30 30 == 00000001
744 test : MICROMEM            36 36 == 00000000
748 test : MICROMEM            44 47 == 0000000E
758 asig.: CC                   0  0 := 00000001
781 test : MICROMEM            48 51 == 0000000E
782 asig.: MICRO_PC             0 11 := 000000D9
```

paso 3.

```
236 test : MICROMEM            26 26 == 00000000
391 test : MICROMEM            24 25 == 00000003
421 asig.: AIZ                  0  3 := 0000000A
422 asig.: ADER                 0  0 := 00000001
423 asig.: ADER                 1  3 := 00000005
424 asig.: BIZ                  0  3 := 0000000A
425 asig.: BDER                 0  0 := 00000001
426 asig.: BDER                 1  3 := 00000005
430 test : MICROMEM            37 38 == 00000000
440 test : MICROMEM            27 27 == 00000000
446 asig.: DI_LATCH             0  7 := 00000000
447 asig.: DD_LATCH             0  7 := 00000000
452 test : MICROMEM             8 10 == 00000003
460 asig.: ALU_IR               0  7 := 00000000
461 asig.: ALU_IS               0  7 := 00000001
462 asig.: ALU_DR               0  7 := 00000000
463 asig.: ALU_DS               0  7 := 00000001
```

```
486 test : MICROMEM          15 15 == 00000000
492 asig.: CN1                0  0 := 00000000
493 asig.: CN2                0  0 := 00000000
498 test : MICROMEM          12 14 == 00000003
548 asig.: SALIDA_ALUI        0  8 := 00000001
549 asig.: SALIDA_ALUD        0  8 := 00000001
572 test : MICROMEM          37 38 == 00000000
573 test : MICROMEM          16 18 == 00000005
599 asig.: BUS_I              0  8 := 00000001
600 asig.: BUS_D              0  8 := 00000001
601 asig.: ADER               0  3 := 0000000B
602 asig.: AIZ                0  3 := 0000000A
603 test : MICROMEM          19 19 == 00000001
615 asig.: SALIDA_ALUI        0  8 := 00000100
616 asig.: SALIDA_ALUI        7  7 := 00000000
617 asig.: BIMEM              0  7 := 00000000
618 asig.: AIMEM              0  7 := 00000000
619 asig.: SALIDA_ALUD        0  8 := 00000100
620 asig.: SALIDA_ALUD        7  7 := 00000000
621 asig.: BDMEM              0  7 := 00000000
622 asig.: ADMEM              0  7 := 00000000
678 test : MICROMEM          42 42 == 00000001
680 test : MICROMEM          34 34 == 00000001
684 asig.: LAT_I              0  7 := 00000001
685 asig.: LAT_D              0  7 := 00000001
688 asig.: PAR_TEST           0  3 := 00000000
690 test : SALIDA_ALUI        0  0 == 00000000
691 test : SALIDA_ALUI        1  1 == 00000000
692 test : SALIDA_ALUI        2  2 == 00000000
693 test : SALIDA_ALUI        3  3 == 00000000
694 test : SALIDA_ALUI        4  4 == 00000000
695 test : SALIDA_ALUI        5  5 == 00000000
696 test : SALIDA_ALUI        6  6 == 00000000
697 test : SALIDA_ALUI        7  7 == 00000000
699 test : PAR_TEST           0  0 == 00000000
699 asig.: PARITY_REG         0  0 := 00000001
700 test : PAR_TEST           0  0 == 00000000
706 test : MICROMEM          20 22 == 00000005
730 asig.: STATUS_REG         0  0 := 00000001
735 test : MICROMEM          43 43 == 00000001
740 test : MICROMEM          32 32 == 00000001
742 test : MICROMEM          30 30 == 00000001
744 test : MICROMEM          36 36 == 00000001
744 asig.: ADR_MEM            0 15 := 00000004
748 test : MICROMEM          44 47 == 0000000F
757 asig.: CC                 0  0 := 00000000
781 test : MICROMEM          48 51 == 00000003
791 test : CC                 0  0 == 00000000
791 asig.: MICRO_PC           0 11 := 00000007
792 test : CC                 0  0 == 00000000

3. paso : fin normal
```

```
SIM> A STATUS_REG
STATUS_REG              = 85
SIM> E2
2. paso : fin normal
SIM> ED
paso 1.

  236 test : MICROMEM          26 26 == 00000000
  391 test : MICROMEM          24 25 == 00000003
  421 asig.: AIZ                0  3 := 00000007
  422 asig.: ADER               0  0 := 00000000
  423 asig.: ADER               1  3 := 00000003
  424 asig.: BIZ                0  3 := 00000007
  425 asig.: BDER               0  0 := 00000000
  426 asig.: BDER               1  3 := 00000003
  430 test : MICROMEM          37 38 == 00000000
  440 test : MICROMEM          27 27 == 00000000
  446 asig.: DI_LATCH           0  7 := 00000000
  447 asig.: DD_LATCH           0  7 := 00000000
  452 test : MICROMEM           8 10 == 00000003
  460 asig.: ALU_IR             0  7 := 00000000
  461 asig.: ALU_IS             0  7 := 00000044
  462 asig.: ALU_DR             0  7 := 00000000
  463 asig.: ALU_DS             0  7 := 00000044
  486 test : MICROMEM          15 15 == 00000000
  492 asig.: CN1                0  0 := 00000000
  493 asig.: CN2                0  0 := 00000000
  498 test : MICROMEM          12 14 == 00000003
  548 asig.: SALIDA_ALUI        0  8 := 00000044
  549 asig.: SALIDA_ALUD        0  8 := 00000044
  572 test : MICROMEM          37 38 == 00000000
  573 test : MICROMEM          16 18 == 00000007
  627 asig.: BUS_I              0  8 := 00000044
  628 asig.: BUS_D              0  8 := 00000044
  629 asig.: ADER               0  3 := 00000006
  630 asig.: AIZ                0  3 := 00000007
  631 test : MICROMEM          19 19 == 00000000
  633 asig.: SALIDA_ALUI        0  8 := 00000088
  634 asig.: SALIDA_ALUI        0  0 := 00000000
  635 asig.: BIMEM              0  7 := 00000088
  636 asig.: AIMEM              0  7 := 00000088
  637 asig.: SALIDA_ALUD        0  8 := 00000088
  638 asig.: SALIDA_ALUD        0  0 := 00000000
  639 asig.: BDMEM              0  7 := 00000088
  640 asig.: ADMEM              0  7 := 00000088
  678 test : MICROMEM          42 42 == 00000001
  680 test : MICROMEM          34 34 == 00000001
  684 asig.: LAT_I              0  7 := 00000044
  685 asig.: LAT_D              0  7 := 00000044
  688 asig.: PAR_TEST           0  3 := 00000000
  690 test : SALIDA_ALUI        0  0 == 00000000
  691 test : SALIDA_ALUI        1  1 == 00000000
  692 test : SALIDA_ALUI        2  2 == 00000000
  693 test : SALIDA_ALUI        3  3 == 00000001
```

```
  693 asig.: PAR_TEST              0  3 := 00000001
  694 test : SALIDA_ALUI           4  4 == 00000000
  695 test : SALIDA_ALUI           5  5 == 00000000
  696 test : SALIDA_ALUI           6  6 == 00000000
  697 test : SALIDA_ALUI           7  7 == 00000001
  697 asig.: PAR_TEST              0  3 := 00000002
  699 test : PAR_TEST              0  0 == 00000000
  699 asig.: PARITY_REG            0  0 := 00000001
  700 test : PAR_TEST              0  0 == 00000000
  706 test : MICROMEM             20 22 == 00000005
  730 asig.: STATUS_REG            0  0 := 00000000
  735 test : MICROMEM             43 43 == 00000001
  740 test : MICROMEM             32 32 == 00000001
  742 test : MICROMEM             30 30 == 00000001
  744 test : MICROMEM             36 36 == 00000001
  744 asig.: ADR_MEM               0 15 := 00000005
  748 test : MICROMEM             44 47 == 0000000F
  757 asig.: CC                    0  0 := 00000000
  781 test : MICROMEM             48 51 == 00000003
  791 test : CC                    0  0 == 00000000
  791 asig.: MICRO_PC              0 11 := 00000007
  792 test : CC                    0  0 == 00000000

1. paso : fin normal
SIM> AM BIMEM 7
BIMEM                : BIZ
[00000007]   88
SIM> E2
2. paso : fin normal
SIM> E2
2. paso : fin normal
SIM> A MICRO_PC
MICRO_PC             = 0C0
SIM> ED
paso 1.

  236 test : MICROMEM             26 26 == 00000000
  391 test : MICROMEM             24 25 == 00000003
  421 asig.: AIZ                   0  3 := 00000007
  422 asig.: ADER                  0  0 := 00000000
  423 asig.: ADER                  1  3 := 00000003
  424 asig.: BIZ                   0  3 := 00000007
  425 asig.: BDER                  0  0 := 00000000
  426 asig.: BDER                  1  3 := 00000003
  430 test : MICROMEM             37 38 == 00000000
  440 test : MICROMEM             27 27 == 00000001
  442 asig.: DI_LATCH              0  7 := 000000FF
  443 asig.: DD_LATCH              0  7 := 000000FF
  452 test : MICROMEM              8 10 == 00000005
  472 asig.: ALU_IR                0  7 := 000000FF
  473 asig.: ALU_IS                0  7 := 00000088
  474 asig.: ALU_DR                0  7 := 000000FF
  475 asig.: ALU_DS                0  7 := 00000088
  486 test : MICROMEM             15 15 == 00000000
```

```
492 asig.: CN1                    0  0 := 00000000
493 asig.: CN2                    0  0 := 00000000
498 test : MICROMEM              12 14 == 00000006
560 asig.: SALIDA_ALUI            0  8 := 00000077
561 asig.: SALIDA_ALUD            0  8 := 00000077
572 test : MICROMEM              37 38 == 00000000
573 test : MICROMEM              16 18 == 00000003
589 asig.: ADER                   0  3 := 00000006
590 asig.: AIZ                    0  3 := 00000007
591 asig.: BIMEM                  0  7 := 00000077
592 asig.: AIMEM                  0  7 := 00000077
593 asig.: BDMEM                  0  7 := 00000077
594 asig.: ADMEM                  0  7 := 00000077
595 asig.: BUS_I                  0  8 := 00000077
596 asig.: BUS_D                  0  8 := 00000077
678 test : MICROMEM              42 42 == 00000001
680 test : MICROMEM              34 34 == 00000001
684 asig.: LAT_I                  0  7 := 00000077
685 asig.: LAT_D                  0  7 := 00000077
688 asig.: PAR_TEST               0  3 := 00000000
690 test : SALIDA_ALUI            0  0 == 00000001
690 asig.: PAR_TEST               0  3 := 00000001
691 test : SALIDA_ALUI            1  1 == 00000001
691 asig.: PAR_TEST               0  3 := 00000002
692 test : SALIDA_ALUI            2  2 == 00000001
692 asig.: PAR_TEST               0  3 := 00000003
693 test : SALIDA_ALUI            3  3 == 00000000
694 test : SALIDA_ALUI            4  4 == 00000001
694 asig.: PAR_TEST               0  3 := 00000004
695 test : SALIDA_ALUI            5  5 == 00000001
695 asig.: PAR_TEST               0  3 := 00000005
696 test : SALIDA_ALUI            6  6 == 00000001
696 asig.: PAR_TEST               0  3 := 00000006
697 test : SALIDA_ALUI            7  7 == 00000000
699 test : PAR_TEST               0  0 == 00000000
699 asig.: PARITY_REG             0  0 := 00000001
700 test : PAR_TEST               0  0 == 00000000
706 test : MICROMEM              20 22 == 00000002
721 asig.: STATUS_REG             0  0 := 00000000
722 asig.: STATUS_REG             2  2 := 00000001
723 asig.: STATUS_REG             4  4 := 00000000
724 test : SALIDA_ALUI            0  8 == 00000077
725 asig.: STATUS_REG             7  7 := 00000000
735 test : MICROMEM              43 43 == 00000001
740 test : MICROMEM              32 32 == 00000001
742 test : MICROMEM              30 30 == 00000001
744 test : MICROMEM              36 36 == 00000001
744 asig.: ADR_MEM                0 15 := 00000007
748 test : MICROMEM              44 47 == 0000000F
757 asig.: CC                     0  0 := 00000000
781 test : MICROMEM              48 51 == 00000003
791 test : CC                     0  0 == 00000000
791 asig.: MICRO_PC               0 11 := 00000007
792 test : CC                     0  0 == 00000000
```

```
1. paso : fin normal
SIM> AM BDMEM 6
BDMEM                : BDER
[00000006]   77
SIM> F
FIN: está seguro ? (S/N) Fin de la simulación
Tiempo de procesado :      17.2 segundos

                  SIMULADOR   RTL  SILOSS
                      EPFL - DET(UPC)
```

Ejecución para el grupo aritmético del programa:

```
          LXI  HL,2026
           INR A
           ADC M
           INX DE
          DAD  DE
           DAA
```

```
SIMULADOR SILOSS. Versión 2.4 DPTO. ELECTRONICA. MARZO 1.989
============================================================

*****EJECUCION:

Para visualizar el menú pulsar "?"


Lectura de definiciones
Lectura del micro-programa de la memoria MICROMEM
SIM> IM MEM_PROG 0000 21,26,20,3C,8E,13,19,27
SIM> IM MEM_PROG 2026 1A
SIM> E 0F
15. paso : fin normal
SIM> R
ADRESS_REG          = 00000003
ADR_MAPPING         = 00000021
ADR_MEM             = 00000003
DATA_REG            = 00000000
INPUT_DATA          = 00000020
MASK_REG            = 00000007
MICRO_PC            = 00000007
REG_INST            = 00000021
STATUS_REG          = 00000000
UNO                 = 00000001
SIM> E2
2. paso : fin normal
SIM> ED
paso 1.

  236 test : MICROMEM          26 26 == 00000000
  391 test : MICROMEM          24 25 == 00000001
  403 asig.: AIZ                0  3 := 00000007
  404 asig.: ADER               0  0 := 00000000
  405 asig.: ADER               1  3 := 00000003
  406 asig.: BIZ                0  2 := 00000007
  407 asig.: BIZ                3  3 := 00000000
  408 asig.: BDER               0  0 := 00000000
  409 asig.: BDER               1  3 := 00000003
  430 test : MICROMEM          37 38 == 00000000
  440 test : MICROMEM          27 27 == 00000000
  446 asig.: DI_LATCH           0  7 := 00000020
  447 asig.: DD_LATCH           0  7 := 00000020
  452 test : MICROMEM           8 10 == 00000003
  460 asig.: ALU_IR             0  7 := 00000000
  461 asig.: ALU_IS             0  7 := 00000000
  462 asig.: ALU_DR             0  7 := 00000000
  463 asig.: ALU_DS             0  7 := 00000000
  486 test : MICROMEM          15 15 == 00000000
```

```
492 asig.: CN1                0  0 := 00000001
493 asig.: CN2                0  0 := 00000001
498 test : MICROMEM          12 14 == 00000000
500 asig.: TEMPI              0  8 := 00000000
501 asig.: TEMPD              0  8 := 00000000
502 asig.: AC                 0  4 := 00000000
503 test : CN1                0  0 == 00000001
506 asig.: SALIDA_ALUI        0  8 := 00000001
507 asig.: AC                 0  4 := 00000001
510 test : CN2                0  0 == 00000001
512 asig.: SALIDA_ALUD        0  8 := 00000001
572 test : MICROMEM          37 38 == 00000000
573 test : MICROMEM          16 18 == 00000003
589 asig.: ADER               0  3 := 00000006
590 asig.: AII                0  3 := 00000007
591 asig.: BIMEM              0  7 := 00000001
592 asig.: AIMEM              0  7 := 00000001
593 asig.: BDMEM              0  7 := 00000001
594 asig.: ADMEM              0  7 := 00000001
595 asig.: BUS_I              0  8 := 00000001
596 asig.: BUS_D              0  8 := 00000001
678 test : MICROMEM          42 42 == 00000001
680 test : MICROMEM          34 34 == 00000001
684 asig.: LAT_I              0  7 := 00000001
685 asig.: LAT_D              0  7 := 00000001
688 asig.: PAR_TEST           0  3 := 00000000
690 test : SALIDA_ALUI        0  0 == 00000001
690 asig.: PAR_TEST           0  3 := 00000001
691 test : SALIDA_ALUI        1  1 == 00000000
692 test : SALIDA_ALUI        2  2 == 00000000
693 test : SALIDA_ALUI        3  3 == 00000000
694 test : SALIDA_ALUI        4  4 == 00000000
695 test : SALIDA_ALUI        5  5 == 00000000
696 test : SALIDA_ALUI        6  6 == 00000000
697 test : SALIDA_ALUI        7  7 == 00000000
699 test : PAR_TEST           0  0 == 00000001
700 test : PAR_TEST           0  0 == 00000001
700 asig.: PARITY_REG         0  0 := 00000000
706 test : MICROMEM          20 22 == 00000001
715 asig.: STATUS_REG         2  2 := 00000000
716 asig.: STATUS_REG         4  4 := 00000000
717 test : SALIDA_ALUI        0  8 == 00000001
718 asig.: STATUS_REG         7  7 := 00000000
735 test : MICROMEM          43 43 == 00000001
740 test : MICROMEM          32 32 == 00000001
742 test : MICROMEM          30 30 == 00000001
744 test : MICROMEM          36 36 == 00000001
744 asig.: ADR_MEM            0 15 := 00000004
748 test : MICROMEM          44 47 == 0000000F
757 asig.: CC                 0  0 := 00000000
781 test : MICROMEM          48 51 == 00000003
791 test : CC                 0  0 == 00000000
791 asig.: MICRO_PC           0 11 := 00000007
792 test : CC                 0  0 == 00000000
```

```
1. paso : fin normal
SIM> AM AIMEM 0 10
AIMEM                : AIZ
[00000000]   00
[00000001]   00
[00000002]   00
[00000003]   00
[00000004]   20
[00000005]   26
[00000006]   00
[00000007]   01
[00000008]   00
[00000009]   00
[0000000A]   00
[0000000B]   00
[0000000C]   60
[0000000D]   06
[0000000E]   00
[0000000F]   00
SIM> AM ADMEM 6
ADMEM                : ADER
[00000006]   01
SIM> I STATUS_REG 01
STATUS_REG           (anterior / nuevo valor)
00 / 01
SIM> E2
2. paso : fin normal
SIM> ED4
paso 1.

  236 test : MICROMEM            26 26 == 00000001
  238 test : MICROMEM            24 25 == 00000003
  264 asig.: AIZ                  0  3 := 00000004
  265 asig.: ADER                 0  3 := 00000004
  266 asig.: BIZ                  0  3 := 00000004
  267 asig.: BDER                 0  3 := 00000004
  271 asig.: A_LATCH              0  7 := 00000026
  272 asig.: A_LATCH              8 15 := 00000020
  273 asig.: B_LATCH              0  7 := 00000026
  274 asig.: B_LATCH              8 15 := 00000020
  278 test : MICROMEM             8 10 == 00000004
  288 asig.: ALU_R                0 15 := 00000000
  289 asig.: ALU_S                0 15 := 00002026
  302 test : MICROMEM            15 15 == 00000000
  304 asig.: CN                   0  0 := 00000000
  308 test : MICROMEM            12 14 == 00000003
  333 asig.: SALIDA_ALU           0 16 := 00002026
  348 test : MICROMEM            37 38 == 00000000
  350 test : MICROMEM            16 18 == 00000002
  353 asig.: ADER                 0  3 := 00000004
  354 asig.: AIZ                  0  3 := 00000004
  355 asig.: BDMEM                0  7 := 00000026
  356 asig.: ADMEM                0  7 := 00000026
  357 asig.: BIMEM                0  7 := 00000020
```

```
  358 asig.: AIMEM                0  7 := 00000020
  359 asig.: Y_BUS                0 15 := 00002026
  375 test : MICROMEM            42 42 == 00000001
  377 test : MICROMEM            27 27 == 00000000
  379 asig.: D_LATCH              0  7 := 00000020
  380 asig.: D_LATCH              8 15 := 00000026
  384 test : MICROMEM            35 35 == 00000000
  384 asig.: ADRESS_REG           0 15 := 00002026
  706 test : MICROMEM            20 22 == 00000004
  735 test : MICROMEM            43 43 == 00000001
  740 test : MICROMEM            32 32 == 00000001
  742 test : MICROMEM            30 30 == 00000001
  744 test : MICROMEM            36 36 == 00000000
  748 test : MICROMEM            44 47 == 0000000E
  758 asig.: CC                   0  0 := 00000001
  781 test : MICROMEM            48 51 == 0000000E
  782 asig.: MICRO_PC             0 11 := 00000073
```

paso 2.

```
  236 test : MICROMEM            26 26 == 00000001
  238 test : MICROMEM            24 25 == 00000003
  264 asig.: AIZ                  0  3 := 0000000F
  265 asig.: ADER                 0  3 := 0000000F
  266 asig.: BIZ                  0  3 := 0000000F
  267 asig.: BDER                 0  3 := 0000000F
  271 asig.: A_LATCH              0  7 := 00000005
  272 asig.: A_LATCH              8 15 := 00000000
  273 asig.: B_LATCH              0  7 := 00000005
  274 asig.: B_LATCH              8 15 := 00000000
  278 test : MICROMEM            8 10 == 00000004
  288 asig.: ALU_R                0 15 := 00000000
  289 asig.: ALU_S                0 15 := 00000005
  302 test : MICROMEM            15 15 == 00000000
  304 asig.: CN                   0  0 := 00000000
  308 test : MICROMEM            12 14 == 00000003
  333 asig.: SALIDA_ALU           0 16 := 00000005
  348 test : MICROMEM            37 38 == 00000000
  350 test : MICROMEM            16 18 == 00000003
  362 asig.: ADER                 0  3 := 0000000F
  363 asig.: AIZ                  0  3 := 0000000F
  364 asig.: BDMEM                0  7 := 00000005
  365 asig.: ADMEM                0  7 := 00000005
  366 asig.: BIMEM                0  7 := 00000000
  367 asig.: AIMEM                0  7 := 00000000
  368 asig.: Y_BUS                0 16 := 00000005
  375 test : MICROMEM            42 42 == 00000001
  377 test : MICROMEM            27 27 == 00000000
  379 asig.: D_LATCH              0  7 := 00000000
  380 asig.: D_LATCH              8 15 := 00000005
  384 test : MICROMEM            35 35 == 00000000
  384 asig.: ADRESS_REG           0 15 := 00000005
  706 test : MICROMEM            20 22 == 00000004
  735 test : MICROMEM            43 43 == 00000001
```

```
  740 test : MICROMEM          32 32 == 00000001
  742 test : MICROMEM          30 30 == 00000001
  744 test : MICROMEM          36 36 == 00000001
  744 asig.: ADR_MEM            0 15 := 00002026
  748 test : MICROMEM          44 47 == 0000000E
  758 asig.: CC                 0  0 := 00000001
  781 test : MICROMEM          48 51 == 0000000E
  782 asig.: MICRO_PC           0 11 := 00000074
```

paso 3.

```
  236 test : MICROMEM          26 26 == 00000000
  391 test : MICROMEM          24 25 == 00000000
  393 asig.: AIZ                0  2 := 00000006
  394 asig.: AIZ                3  3 := 00000000
  395 asig.: ADER               0  0 := 00000001
  396 asig.: ADER               1  3 := 00000003
  397 asig.: BIZ                0  2 := 00000001
  398 asig.: BIZ                3  3 := 00000000
  399 asig.: BDER               0  0 := 00000000
  400 asig.: BDER               1  3 := 00000000
  430 test : MICROMEM          37 38 == 00000000
  440 test : MICROMEM          27 27 == 00000000
  446 asig.: DI_LATCH           0  7 := 00000001
  447 asig.: DD_LATCH           0  7 := 00000001
  452 test : MICROMEM           8 10 == 00000003
  460 asig.: ALU_IR             0  7 := 00000000
  461 asig.: ALU_IS             0  7 := 00000000
  462 asig.: ALU_DR             0  7 := 00000000
  463 asig.: ALU_DS             0  7 := 00000000
  486 test : MICROMEM          15 15 == 00000000
  492 asig.: CN1                0  0 := 00000000
  493 asig.: CN2                0  0 := 00000000
  498 test : MICROMEM          12 14 == 00000003
  548 asig.: SALIDA_ALUI        0  8 := 00000000
  549 asig.: SALIDA_ALUD        0  8 := 00000000
  572 test : MICROMEM          37 38 == 00000000
  573 test : MICROMEM          16 18 == 00000001
  575 asig.: BUS_I              0  8 := 00000000
  576 asig.: BUS_D              0  8 := 00000000
  678 test : MICROMEM          42 42 == 00000001
  680 test : MICROMEM          34 34 == 00000001
  684 asig.: LAT_I              0  7 := 00000000
  685 asig.: LAT_D              0  7 := 00000000
  688 asig.: PAR_TEST           0  3 := 00000000
  690 test : SALIDA_ALUI        0  0 == 00000000
  691 test : SALIDA_ALUI        1  1 == 00000000
  692 test : SALIDA_ALUI        2  2 == 00000000
  693 test : SALIDA_ALUI        3  3 == 00000000
  694 test : SALIDA_ALUI        4  4 == 00000000
  695 test : SALIDA_ALUI        5  5 == 00000000
  696 test : SALIDA_ALUI        6  6 == 00000000
  697 test : SALIDA_ALUI        7  7 == 00000000
  699 test : PAR_TEST           0  0 == 00000000
```

```
699 asig.: PARITY_REG          0  0 := 00000001
700 test : PAR_TEST            0  0 == 00000000
706 test : MICROMEM           20 22 == 00000004
735 test : MICROMEM           43 43 == 00000001
740 test : MICROMEM           32 32 == 00000001
742 test : MICROMEM           30 30 == 00000000
742 asig.: INPUT_DATA          0  7 := 0000001E
744 test : MICROMEM           36 36 == 00000000
748 test : MICROMEM           44 47 == 0000000E
758 asig.: CC                  0  0 := 00000001
781 test : MICROMEM           48 51 == 0000000E
782 asig.: MICRO_PC            0 11 := 00000075
```

paso 4.

```
236 test : MICROMEM           26 26 == 00000000
391 test : MICROMEM           24 25 == 00000003
421 asig.: AIZ                 0  3 := 00000007
422 asig.: ADER                0  0 := 00000000
423 asig.: ADER                1  3 := 00000003
424 asig.: BIZ                 0  3 := 00000007
425 asig.: BDER                0  0 := 00000000
426 asig.: BDER                1  3 := 00000003
430 test : MICROMEM           37 38 == 00000000
440 test : MICROMEM           27 27 == 00000001
442 asig.: DI_LATCH            0  7 := 0000001E
443 asig.: DD_LATCH            0  7 := 0000001E
452 test : MICROMEM            8 10 == 00000005
472 asig.: ALU_IR              0  7 := 0000001E
473 asig.: ALU_IS              0  7 := 00000001
474 asig.: ALU_DR              0  7 := 0000001E
475 asig.: ALU_DS              0  7 := 00000001
486 test : MICROMEM           15 15 == 00000001
488 asig.: CN1                 0  0 := 00000001
489 asig.: CN2                 0  0 := 00000001
498 test : MICROMEM           12 14 == 00000000
500 asig.: TEMPI               0  8 := 0000001F
501 asig.: TEMPD               0  8 := 0000001F
502 asig.: AC                  0  4 := 0000000F
503 test : CN1                 0  0 == 00000001
506 asig.: SALIDA_ALUI         0  8 := 00000020
507 asig.: AC                  0  4 := 00000010
510 test : CN2                 0  0 == 00000001
512 asig.: SALIDA_ALUD         0  8 := 00000020
572 test : MICROMEM           37 38 == 00000000
573 test : MICROMEM           16 18 == 00000003
589 asig.: ADER                0  3 := 00000006
590 asig.: AIZ                 0  3 := 00000007
591 asig.: BIMEM               0  7 := 00000020
592 asig.: AIMEM               0  7 := 00000020
593 asig.: BDMEM               0  7 := 00000020
594 asig.: ADMEM               0  7 := 00000020
595 asig.: BUS_I               0  8 := 00000020
596 asig.: BUS_D               0  8 := 00000020
```

```
  678 test : MICROMEM          42 42 == 00000001
  680 test : MICROMEM          34 34 == 00000001
  684 asig.: LAT_I              0  7 := 00000020
  685 asig.: LAT_D              0  7 := 00000020
  688 asig.: PAR_TEST           0  3 := 00000000
  690 test : SALIDA_ALUI        0  0 == 00000000
  691 test : SALIDA_ALUI        1  1 == 00000000
  692 test : SALIDA_ALUI        2  2 == 00000000
  693 test : SALIDA_ALUI        3  3 == 00000000
  694 test : SALIDA_ALUI        4  4 == 00000000
  695 test : SALIDA_ALUI        5  5 == 00000001
  695 asig.: PAR_TEST           0  3 := 00000001
  696 test : SALIDA_ALUI        6  6 == 00000000
  697 test : SALIDA_ALUI        7  7 == 00000000
  699 test : PAR_TEST           0  0 == 00000001
  700 test : PAR_TEST           0  0 == 00000001
  700 asig.: PARITY_REG         0  0 := 00000000
  706 test : MICROMEM          20 22 == 00000000
  708 asig.: STATUS_REG         0  0 := 00000000
  709 asig.: STATUS_REG         2  2 := 00000000
  710 asig.: STATUS_REG         4  4 := 00000001
  711 test : SALIDA_ALUI        0  8 == 00000020
  712 asig.: STATUS_REG         7  7 := 00000000
  735 test : MICROMEM          43 43 == 00000001
  740 test : MICROMEM          32 32 == 00000001
  742 test : MICROMEM          30 30 == 00000001
  744 test : MICROMEM          36 36 == 00000001
  744 asig.: ADR_MEM            0 15 := 00000005
  748 test : MICROMEM          44 47 == 0000000F
  757 asig.: CC                 0  0 := 00000000
  781 test : MICROMEM          48 51 == 00000003
  791 test : CC                 0  0 == 00000000
  791 asig.: MICRO_PC           0 11 := 00000007
  792 test : CC                 0  0 == 00000000

4. paso : fin normal
SIM> AM AIMEM 0 10
AIMEM                 : AIZ
[00000000]   00
[00000001]   00
[00000002]   00
[00000003]   00
[00000004]   20
[00000005]   26
[00000006]   00
[00000007]   20
[00000008]   00
[00000009]   00
[0000000A]   00
[0000000B]   00
[0000000C]   60
[0000000D]   06
[0000000E]   00
[0000000F]   00
```

```
SIM> AM ADMEM 0 10
ADMEM                : ADER
[00000000]   00
[00000001]   00
[00000002]   00
[00000003]   00
[00000004]   26
[00000005]   20
[00000006]   20
[00000007]   00
[00000008]   00
[00000009]   00
[0000000A]   00
[0000000B]   00
[0000000C]   06
[0000000D]   60
[0000000E]   00
[0000000F]   05
SIM> E2
2. paso : fin normal
SIM> ED2
paso 1.

  236 test : MICROMEM          26 26 == 00000001
  238 test : MICROMEM          24 25 == 00000003
  264 asig.: AIZ                0  3 := 00000002
  265 asig.: ADER               0  3 := 00000002
  266 asig.: BIZ                0  3 := 00000002
  267 asig.: BDER               0  3 := 00000002
  271 asig.: A_LATCH            0  7 := 00000000
  272 asig.: A_LATCH            8 15 := 00000000
  273 asig.: B_LATCH            0  7 := 00000000
  274 asig.: B_LATCH            8 15 := 00000000
  278 test : MICROMEM           8 10 == 00000004
  288 asig.: ALU_R              0 15 := 00000000
  289 asig.: ALU_S              0 15 := 00000000
  302 test : MICROMEM          15 15 == 00000000
  304 asig.: CN                 0  0 := 00000001
  308 test : MICROMEM          12 14 == 00000000
  310 asig.: TEMP               0 16 := 00000000
  311 test : CN                 0  0 == 00000001
  313 asig.: SALIDA_ALU         0 16 := 00000001
  348 test : MICROMEM          37 38 == 00000000
  350 test : MICROMEM          16 18 == 00000003
  362 asig.: ADER               0  3 := 00000002
  363 asig.: AIZ                0  3 := 00000002
  364 asig.: BDMEM              0  7 := 00000001
  365 asig.: ADMEM              0  7 := 00000001
  366 asig.: BIMEM              0  7 := 00000000
  367 asig.: AIMEM              0  7 := 00000000
  368 asig.: Y_BUS              0 16 := 00000001
  375 test : MICROMEM          42 42 == 00000001
  377 test : MICROMEM          27 27 == 00000000
  379 asig.: D_LATCH            0  7 := 00000000
```

```
380 asig.: D_LATCH          8 15 := 00000001
384 test : MICROMEM        35 35 == 00000001
706 test : MICROMEM        20 22 == 00000004
735 test : MICROMEM        43 43 == 00000001
740 test : MICROMEM        32 32 == 00000001
742 test : MICROMEM        30 30 == 00000001
744 test : MICROMEM        36 36 == 00000000
748 test : MICROMEM        44 47 == 0000000E
758 asig.: CC               0  0 := 00000001
781 test : MICROMEM        48 51 == 0000000E
782 asig.: MICRO_PC         0 11 := 00000096
```

paso 2.

```
236 test : MICROMEM        26 26 == 00000001
238 test : MICROMEM        24 25 == 00000003
264 asig.: AIZ              0  3 := 00000002
265 asig.: ADER             0  3 := 00000002
266 asig.: BIZ              0  3 := 00000003
267 asig.: BDER             0  3 := 00000003
271 asig.: A_LATCH          0  7 := 00000001
272 asig.: A_LATCH          8 15 := 00000000
273 asig.: B_LATCH          0  7 := 00000000
274 asig.: B_LATCH          8 15 := 00000000
278 test : MICROMEM         8 10 == 00000007
296 asig.: ALU_R            0 15 := 00000100
297 asig.: ALU_S            0 15 := 00000000
302 test : MICROMEM        15 15 == 00000000
304 asig.: CN               0  0 := 00000000
308 test : MICROMEM        12 14 == 00000003
333 asig.: SALIDA_ALU       0 16 := 00000100
348 test : MICROMEM        37 38 == 00000000
350 test : MICROMEM        16 18 == 00000002
353 asig.: ADER             0  3 := 00000003
354 asig.: AIZ              0  3 := 00000003
355 asig.: BDMEM            0  7 := 00000000
356 asig.: ADMEM            0  7 := 00000000
357 asig.: BIMEM            0  7 := 00000001
358 asig.: AIMEM            0  7 := 00000001
359 asig.: Y_BUS            0 15 := 00000001
375 test : MICROMEM        42 42 == 00000001
377 test : MICROMEM        27 27 == 00000000
379 asig.: D_LATCH          0  7 := 00000000
380 asig.: D_LATCH          8 15 := 00000001
384 test : MICROMEM        35 35 == 00000001
706 test : MICROMEM        20 22 == 00000004
735 test : MICROMEM        43 43 == 00000001
740 test : MICROMEM        32 32 == 00000001
742 test : MICROMEM        30 30 == 00000001
744 test : MICROMEM        36 36 == 00000001
744 asig.: ADR_MEM          0 15 := 00000006
748 test : MICROMEM        44 47 == 0000000F
757 asig.: CC               0  0 := 00000000
781 test : MICROMEM        48 51 == 00000003
```

```
  791 test : CC                    0  0 == 00000000
  791 asig.: MICRO_PC              0 11 := 00000007
  792 test : CC                    0  0 == 00000000

2. paso : fin normal
SIM> AM AIMEM  0 4
AIMEM                : AIZ
[00000000]   00
[00000001]   00
[00000002]   00
[00000003]   01
SIM> AM ADMEM 0 4
ADMEM                : ADER
[00000000]   00
[00000001]   00
[00000002]   01
[00000003]   00
SIM> E2
2. paso : fin normal
SIM> ED2
paso 1.

  236 test : MICROMEM             26 26 == 00000001
  238 test : MICROMEM             24 25 == 00000003
  264 asig.: AIZ                   0  3 := 00000002
  265 asig.: ADER                  0  3 := 00000002
  266 asig.: BIZ                   0  3 := 00000004
  267 asig.: BDER                  0  3 := 00000004
  271 asig.: A_LATCH               0  7 := 00000001
  272 asig.: A_LATCH               8 15 := 00000000
  273 asig.: B_LATCH               0  7 := 00000026
  274 asig.: B_LATCH               8 15 := 00000020
  278 test : MICROMEM              8 10 == 00000001
  280 asig.: ALU_R                 0 15 := 00000001
  281 asig.: ALU_S                 0 15 := 00002026
  302 test : MICROMEM             15 15 == 00000000
  304 asig.: CN                    0  0 := 00000000
  308 test : MICROMEM             12 14 == 00000000
  310 asig.: TEMP                  0 16 := 00002027
  311 test : CN                    0  0 == 00000000
  312 asig.: SALIDA_ALU            0 16 := 00002027
  348 test : MICROMEM             37 38 == 00000000
  350 test : MICROMEM             16 18 == 00000003
  362 asig.: ADER                  0  3 := 00000004
  363 asig.: AIZ                   0  3 := 00000004
  364 asig.: BDMEM                 0  7 := 00000027
  365 asig.: ADMEM                 0  7 := 00000027
  366 asig.: BIMEM                 0  7 := 00000020
  367 asig.: AIMEM                 0  7 := 00000020
  368 asig.: Y_BUS                 0 16 := 00002027
  375 test : MICROMEM             42 42 == 00000001
  377 test : MICROMEM             27 27 == 00000000
  379 asig.: D_LATCH               0  7 := 00000020
  380 asig.: D_LATCH               8 15 := 00000027
```

```
 384 test : MICROMEM          35 35 == 00000001
 706 test : MICROMEM          20 22 == 00000003
 727 asig.: STATUS_REG         0  0 := 00000000
 735 test : MICROMEM          43 43 == 00000001
 740 test : MICROMEM          32 32 == 00000001
 742 test : MICROMEM          30 30 == 00000001
 744 test : MICROMEM          36 36 == 00000000
 748 test : MICROMEM          44 47 == 0000000E
 758 asig.: CC                 0  0 := 00000001
 781 test : MICROMEM          48 51 == 0000000E
 782 asig.: MICRO_PC           0 11 := 000000A6
```

paso 2.

```
 236 test : MICROMEM          26 26 == 00000001
 238 test : MICROMEM          24 25 == 00000003
 264 asig.: AIZ                0  3 := 00000004
 265 asig.: ADER               0  3 := 00000004
 266 asig.: BIZ                0  3 := 00000005
 267 asig.: BDER               0  3 := 00000005
 271 asig.: A_LATCH            0  7 := 00000027
 272 asig.: A_LATCH            8 15 := 00000020
 273 asig.: B_LATCH            0  7 := 00000020
 274 asig.: B_LATCH            8 15 := 00000026
 278 test : MICROMEM           8 10 == 00000007
 296 asig.: ALU_R              0 15 := 00002720
 297 asig.: ALU_S              0 15 := 00000000
 302 test : MICROMEM          15 15 == 00000000
 304 asig.: CN                 0  0 := 00000000
 308 test : MICROMEM          12 14 == 00000003
 333 asig.: SALIDA_ALU         0 16 := 00002720
 348 test : MICROMEM          37 38 == 00000000
 350 test : MICROMEM          16 18 == 00000002
 353 asig.: ADER               0  3 := 00000005
 354 asig.: AIZ                0  3 := 00000005
 355 asig.: BDMEM              0  7 := 00000020
 356 asig.: ADMEM              0  7 := 00000020
 357 asig.: BIMEM              0  7 := 00000027
 358 asig.: AIMEM              0  7 := 00000027
 359 asig.: Y_BUS              0 15 := 00002027
 375 test : MICROMEM          42 42 == 00000001
 377 test : MICROMEM          27 27 == 00000000
 379 asig.: D_LATCH            0  7 := 00000020
 380 asig.: D_LATCH            8 15 := 00000027
 384 test : MICROMEM          35 35 == 00000001
 706 test : MICROMEM          20 22 == 00000004
 735 test : MICROMEM          43 43 == 00000001
 740 test : MICROMEM          32 32 == 00000001
 742 test : MICROMEM          30 30 == 00000001
 744 test : MICROMEM          36 36 == 00000001
 744 asig.: ADR_MEM            0 15 := 00000007
 748 test : MICROMEM          44 47 == 0000000F
 757 asig.: CC                 0  0 := 00000000
 781 test : MICROMEM          48 51 == 00000003
```

```
  791 test : CC                    0  0 == 00000000
  791 asig.: MICRO_PC              0 11 := 00000007
  792 test : CC                    0  0 == 00000000

2. paso : fin normal
SIM> AM AIMEM 0 10
AIMEM                : AIZ
[00000000]   00
[00000001]   00
[00000002]   00
[00000003]   01
[00000004]   20
[00000005]   27
[00000006]   00
[00000007]   20
[00000008]   00
[00000009]   00
[0000000A]   00
[0000000B]   00
[0000000C]   60
[0000000D]   06
[0000000E]   00
[0000000F]   00
SIM> AM BDMEM 0 10
BDMEM                : BDER
[00000000]   00
[00000001]   00
[00000002]   01
[00000003]   00
[00000004]   27
[00000005]   20
[00000006]   20
[00000007]   00
[00000008]   00
[00000009]   00
[0000000A]   00
[0000000B]   00
[0000000C]   06
[0000000D]   60
[0000000E]   00
[0000000F]   07
SIM> R
ADRESS_REG           = 00000007
ADR_MAPPING          = 00000019
ADR_MEM              = 00000007
DATA_REG             = 00000000
INPUT_DATA           = 00000019
MASK_REG             = 00000007
MICRO_PC             = 00000007
REG_INST             = 00000019
STATUS_REG           = 00000010
UNO                  = 00000001
```

```
SIM> ED7
paso 1.

  236 test : MICROMEM          26 26 == 00000000
  391 test : MICROMEM          24 25 == 00000000
  393 asig.: AIZ                0  2 := 00000001
  394 asig.: AIZ                3  3 := 00000000
  395 asig.: ADER               0  0 := 00000000
  396 asig.: ADER               1  3 := 00000000
  397 asig.: BIZ                0  2 := 00000003
  398 asig.: BIZ                3  3 := 00000000
  399 asig.: BDER               0  0 := 00000000
  400 asig.: BDER               1  3 := 00000001
  430 test : MICROMEM          37 38 == 00000000
  440 test : MICROMEM          27 27 == 00000000
  446 asig.: DI_LATCH           0  7 := 00000000
  447 asig.: DD_LATCH           0  7 := 00000000
  452 test : MICROMEM           8 10 == 00000003
  460 asig.: ALU_IR             0  7 := 00000000
  461 asig.: ALU_IS             0  7 := 00000001
  462 asig.: ALU_DR             0  7 := 00000000
  463 asig.: ALU_DS             0  7 := 00000001
  486 test : MICROMEM          15 15 == 00000000
  492 asig.: CN1                0  0 := 00000000
  493 asig.: CN2                0  0 := 00000000
  498 test : MICROMEM          12 14 == 00000003
  548 asig.: SALIDA_ALUI        0  8 := 00000001
  549 asig.: SALIDA_ALUD        0  8 := 00000001
  572 test : MICROMEM          37 38 == 00000000
  573 test : MICROMEM          16 18 == 00000001
  575 asig.: BUS_I              0  8 := 00000001
  576 asig.: BUS_D              0  8 := 00000001
  678 test : MICROMEM          42 42 == 00000001
  680 test : MICROMEM          34 34 == 00000001
  684 asig.: LAT_I              0  7 := 00000001
  685 asig.: LAT_D              0  7 := 00000001
  688 asig.: PAR_TEST           0  3 := 00000000
  690 test : SALIDA_ALUI        0  0 == 00000001
  690 asig.: PAR_TEST           0  3 := 00000001
  691 test : SALIDA_ALUI        1  1 == 00000000
  692 test : SALIDA_ALUI        2  2 == 00000000
  693 test : SALIDA_ALUI        3  3 == 00000000
  694 test : SALIDA_ALUI        4  4 == 00000000
  695 test : SALIDA_ALUI        5  5 == 00000000
  696 test : SALIDA_ALUI        6  6 == 00000000
  697 test : SALIDA_ALUI        7  7 == 00000000
  699 test : PAR_TEST           0  0 == 00000001
  700 test : PAR_TEST           0  0 == 00000001
  700 asig.: PARITY_REG         0  0 := 00000000
  706 test : MICROMEM          20 22 == 00000004
  735 test : MICROMEM          43 43 == 00000000
  736 asig.: REG_INST           0  7 := 00000027
  737 asig.: ADR_MAPPING        0  7 := 00000027
  740 test : MICROMEM          32 32 == 00000001
```

```
  742 test : MICROMEM          30 30 == 00000000
  742 asig.: INPUT_DATA         0  7 := 00000027
  744 test : MICROMEM          36 36 == 00000000
  748 test : MICROMEM          44 47 == 0000000C
  781 test : MICROMEM          48 51 == 00000006
  800 asig.: MICRO_PC           0 11 := 00000008
```

paso 2.

```
  236 test : MICROMEM          26 26 == 00000001
  238 test : MICROMEM          24 25 == 00000003
  264 asig.: AIZ                0  3 := 0000000F
  265 asig.: ADER               0  3 := 0000000F
  266 asig.: BIZ                0  3 := 0000000F
  267 asig.: BDER               0  3 := 0000000F
  271 asig.: A_LATCH            0  7 := 00000007
  272 asig.: A_LATCH            8 15 := 00000000
  273 asig.: B_LATCH            0  7 := 00000007
  274 asig.: B_LATCH            8 15 := 00000000
  278 test : MICROMEM           8 10 == 00000004
  288 asig.: ALU_R              0 15 := 00000000
  289 asig.: ALU_S              0 15 := 00000007
  302 test : MICROMEM          15 15 == 00000000
  304 asig.: CN                 0  0 := 00000001
  308 test : MICROMEM          12 14 == 00000000
  310 asig.: TEMP               0 16 := 00000007
  311 test : CN                 0  0 == 00000001
  313 asig.: SALIDA_ALU         0 16 := 00000008
  348 test : MICROMEM          37 38 == 00000000
  350 test : MICROMEM          16 18 == 00000003
  362 asig.: ADER               0  3 := 0000000F
  363 asig.: AIZ                0  3 := 0000000F
  364 asig.: BDMEM              0  7 := 00000008
  365 asig.: ADMEM              0  7 := 00000008
  366 asig.: BIMEM              0  7 := 00000000
  367 asig.: AIMEM              0  7 := 00000000
  368 asig.: Y_BUS              0 16 := 00000008
  375 test : MICROMEM          42 42 == 00000001
  377 test : MICROMEM          27 27 == 00000000
  379 asig.: D_LATCH            0  7 := 00000000
  380 asig.: D_LATCH            8 15 := 00000008
  384 test : MICROMEM          35 35 == 00000000
  384 asig.: ADRESS_REG         0 15 := 00000008
  706 test : MICROMEM          20 22 == 00000004
  735 test : MICROMEM          43 43 == 00000001
  740 test : MICROMEM          32 32 == 00000001
  742 test : MICROMEM          30 30 == 00000001
  744 test : MICROMEM          36 36 == 00000000
  748 test : MICROMEM          44 47 == 0000000E
  758 asig.: CC                 0  0 := 00000001
  781 test : MICROMEM          48 51 == 00000002
  799 asig.: MICRO_PC           0 11 := 000000AB
```

paso 3.

```
236 test : MICROMEM          26 26 == 00000000
391 test : MICROMEM          24 25 == 00000003
421 asig.: AIZ                0  3 := 00000007
422 asig.: ADER               0  0 := 00000000
423 asig.: ADER               1  3 := 00000003
424 asig.: BIZ                0  3 := 00000007
425 asig.: BDER               0  0 := 00000000
426 asig.: BDER               1  3 := 00000003
430 test : MICROMEM          37 38 == 00000000
440 test : MICROMEM          27 27 == 00000000
446 asig.: DI_LATCH           0  7 := 00000001
447 asig.: DD_LATCH           0  7 := 00000001
452 test : MICROMEM           8 10 == 00000004
466 asig.: ALU_IR             0  7 := 00000000
467 asig.: ALU_IS             0  7 := 00000020
468 asig.: ALU_DR             0  7 := 00000000
469 asig.: ALU_DS             0  7 := 00000020
486 test : MICROMEM          15 15 == 00000000
492 asig.: CN1                0  0 := 00000000
493 asig.: CN2                0  0 := 00000000
498 test : MICROMEM          12 14 == 00000003
548 asig.: SALIDA_ALUI        0  8 := 00000020
549 asig.: SALIDA_ALUD        0  8 := 00000020
572 test : MICROMEM          37 38 == 00000000
573 test : MICROMEM          16 18 == 00000002
579 asig.: ADER               0  3 := 00000006
580 asig.: AIZ                0  3 := 00000007
581 asig.: BUS_I              0  7 := 00000020
582 asig.: BIMEM              0  7 := 00000020
583 asig.: AIMEM              0  7 := 00000020
584 asig.: BUS_D              0  7 := 00000020
585 asig.: BDMEM              0  7 := 00000020
586 asig.: ADMEM              0  7 := 00000020
678 test : MICROMEM          42 42 == 00000001
680 test : MICROMEM          34 34 == 00000001
684 asig.: LAT_I              0  7 := 00000020
685 asig.: LAT_D              0  7 := 00000020
688 asig.: PAR_TEST           0  3 := 00000000
690 test : SALIDA_ALUI        0  0 == 00000000
691 test : SALIDA_ALUI        1  1 == 00000000
692 test : SALIDA_ALUI        2  2 == 00000000
693 test : SALIDA_ALUI        3  3 == 00000000
694 test : SALIDA_ALUI        4  4 == 00000000
695 test : SALIDA_ALUI        5  5 == 00000001
695 asig.: PAR_TEST           0  3 := 00000001
696 test : SALIDA_ALUI        6  6 == 00000000
697 test : SALIDA_ALUI        7  7 == 00000000
699 test : PAR_TEST           0  0 == 00000001
700 test : PAR_TEST           0  0 == 00000001
700 asig.: PARITY_REG         0  0 := 00000000
706 test : MICROMEM          20 22 == 00000004
735 test : MICROMEM          43 43 == 00000001
740 test : MICROMEM          32 32 == 00000001
```

```
  742 test : MICROMEM          30 30 == 00000001
  744 test : MICROMEM          36 36 == 00000000
  748 test : MICROMEM          44 47 == 0000000E
  758 asig.: CC                 0  0 := 00000001
  781 test : MICROMEM          48 51 == 0000000E
  782 asig.: MICRO_PC           0 11 := 000000AC

paso 4.

  236 test : MICROMEM          26 26 == 00000000
  391 test : MICROMEM          24 25 == 00000003
  421 asig.: AIZ                0  3 := 00000007
  422 asig.: ADER               0  0 := 00000000
  423 asig.: ADER               1  3 := 00000003
  424 asig.: BIZ                0  3 := 00000007
  425 asig.: BDER               0  0 := 00000000
  426 asig.: BDER               1  3 := 00000003
  430 test : MICROMEM          37 38 == 00000000
  440 test : MICROMEM          27 27 == 00000000
  446 asig.: DI_LATCH           0  7 := 00000020
  447 asig.: DD_LATCH           0  7 := 00000020
  452 test : MICROMEM           8 10 == 00000004
  466 asig.: ALU_IR             0  7 := 00000000
  467 asig.: ALU_IS             0  7 := 00000020
  468 asig.: ALU_DR             0  7 := 00000000
  469 asig.: ALU_DS             0  7 := 00000020
  486 test : MICROMEM          15 15 == 00000000
  492 asig.: CN1                0  0 := 00000000
  493 asig.: CN2                0  0 := 00000000
  498 test : MICROMEM          12 14 == 00000003
  548 asig.: SALIDA_ALUI        0  8 := 00000020
  549 asig.: SALIDA_ALUD        0  8 := 00000020
  572 test : MICROMEM          37 38 == 00000000
  573 test : MICROMEM          16 18 == 00000002
  579 asig.: ADER               0  3 := 00000006
  580 asig.: AIZ                0  3 := 00000007
  581 asig.: BUS_I              0  7 := 00000020
  582 asig.: BIMEM              0  7 := 00000020
  583 asig.: AIMEM              0  7 := 00000020
  584 asig.: BUS_D              0  7 := 00000020
  585 asig.: BDMEM              0  7 := 00000020
  586 asig.: ADMEM              0  7 := 00000020
  678 test : MICROMEM          42 42 == 00000001
  680 test : MICROMEM          34 34 == 00000001
  684 asig.: LAT_I              0  7 := 00000020
  685 asig.: LAT_D              0  7 := 00000020
  688 asig.: PAR_TEST           0  3 := 00000000
  690 test : SALIDA_ALUI        0  0 == 00000000
  691 test : SALIDA_ALUI        1  1 == 00000000
  692 test : SALIDA_ALUI        2  2 == 00000000
  693 test : SALIDA_ALUI        3  3 == 00000000
  694 test : SALIDA_ALUI        4  4 == 00000000
  695 test : SALIDA_ALUI        5  5 == 00000001
  695 asig.: PAR_TEST           0  3 := 00000001
```

```
696 test : SALIDA_ALUI          6  6 == 00000000
697 test : SALIDA_ALUI          7  7 == 00000000
699 test : PAR_TEST             0  0 == 00000001
700 test : PAR_TEST             0  0 == 00000001
700 asig.: PARITY_REG           0  0 := 00000000
706 test : MICROMEM            20 22 == 00000004
735 test : MICROMEM            43 43 == 00000001
740 test : MICROMEM            32 32 == 00000001
742 test : MICROMEM            30 30 == 00000001
744 test : MICROMEM            36 36 == 00000000
748 test : MICROMEM            44 47 == 00000008
760 asig.: TEMP1                0  0 := 00000000
761 asig.: TEMP2                0  0 := 00000000
762 asig.: TEMP3                0  0 := 00000001
763 test : TEMP3                0  0 == 00000001
764 asig.: CC                   0  0 := 00000000
781 test : MICROMEM            48 51 == 00000003
791 test : CC                   0  0 == 00000000
791 asig.: MICRO_PC             0 11 := 000000AF
792 test : CC                   0  0 == 00000000
```

paso 5.

```
236 test : MICROMEM            26 26 == 00000000
391 test : MICROMEM            24 25 == 00000003
421 asig.: AIZ                  0  3 := 0000000D
422 asig.: ADER                 0  0 := 00000000
423 asig.: ADER                 1  3 := 00000006
424 asig.: BIZ                  0  3 := 00000007
425 asig.: BDER                 0  0 := 00000000
426 asig.: BDER                 1  3 := 00000003
430 test : MICROMEM            37 38 == 00000000
440 test : MICROMEM            27 27 == 00000000
446 asig.: DI_LATCH             0  7 := 00000020
447 asig.: DD_LATCH             0  7 := 00000020
452 test : MICROMEM             8 10 == 00000001
454 asig.: ALU_IR               0  7 := 00000006
455 asig.: ALU_IS               0  7 := 00000020
456 asig.: ALU_DR               0  7 := 00000006
457 asig.: ALU_DS               0  7 := 00000020
486 test : MICROMEM            15 15 == 00000000
492 asig.: CN1                  0  0 := 00000000
493 asig.: CN2                  0  0 := 00000000
498 test : MICROMEM            12 14 == 00000000
500 asig.: TEMPI                0  8 := 00000026
501 asig.: TEMPD                0  8 := 00000026
502 asig.: AC                   0  4 := 00000006
503 test : CN1                  0  0 == 00000000
504 asig.: SALIDA_ALUI          0  8 := 00000026
510 test : CN2                  0  0 == 00000000
511 asig.: SALIDA_ALUD          0  8 := 00000026
572 test : MICROMEM            37 38 == 00000000
573 test : MICROMEM            16 18 == 00000003
589 asig.: ADER                 0  3 := 00000006
```

```
590 asig.: AIZ                  0  3 := 00000007
591 asig.: BIMEM                0  7 := 00000026
592 asig.: AIMEM                0  7 := 00000026
593 asig.: BDMEM                0  7 := 00000026
594 asig.: ADMEM                0  7 := 00000026
595 asig.: BUS_I                0  8 := 00000026
596 asig.: BUS_D                0  8 := 00000026
678 test : MICROMEM            42 42 == 00000001
680 test : MICROMEM            34 34 == 00000001
684 asig.: LAT_I                0  7 := 00000026
685 asig.: LAT_D                0  7 := 00000026
688 asig.: PAR_TEST             0  3 := 00000000
690 test : SALIDA_ALUI          0  0 == 00000000
691 test : SALIDA_ALUI          1  1 == 00000001
691 asig.: PAR_TEST             0  3 := 00000001
692 test : SALIDA_ALUI          2  2 == 00000001
692 asig.: PAR_TEST             0  3 := 00000002
693 test : SALIDA_ALUI          3  3 == 00000000
694 test : SALIDA_ALUI          4  4 == 00000000
695 test : SALIDA_ALUI          5  5 == 00000001
695 asig.: PAR_TEST             0  3 := 00000003
696 test : SALIDA_ALUI          6  6 == 00000000
697 test : SALIDA_ALUI          7  7 == 00000000
699 test : PAR_TEST             0  0 == 00000001
700 test : PAR_TEST             0  0 == 00000001
700 asig.: PARITY_REG           0  0 := 00000000
706 test : MICROMEM            20 22 == 00000000
708 asig.: STATUS_REG           0  0 := 00000000
709 asig.: STATUS_REG           2  2 := 00000000
710 asig.: STATUS_REG           4  4 := 00000000
711 test : SALIDA_ALUI          0  8 == 00000026
712 asig.: STATUS_REG           7  7 := 00000000
735 test : MICROMEM            43 43 == 00000001
740 test : MICROMEM            32 32 == 00000001
742 test : MICROMEM            30 30 == 00000001
744 test : MICROMEM            36 36 == 00000000
748 test : MICROMEM            44 47 == 0000000F
757 asig.: CC                   0  0 := 00000000
781 test : MICROMEM            48 51 == 00000003
791 test : CC                   0  0 == 00000000
791 asig.: MICRO_PC             0 11 := 000000AD
792 test : CC                   0  0 == 00000000
```

paso 6.

```
236 test : MICROMEM            26 26 == 00000000
391 test : MICROMEM            24 25 == 00000000
393 asig.: AIZ                  0  2 := 00000007
394 asig.: AIZ                  3  3 := 00000000
395 asig.: ADER                 0  0 := 00000000
396 asig.: ADER                 1  3 := 00000003
397 asig.: BIZ                  0  2 := 00000004
398 asig.: BIZ                  3  3 := 00000000
399 asig.: BDER                 0  0 := 00000001
```

```
400 asig.: BDER                 1  3 := 00000002
430 test : MICROMEM            37 38 == 00000000
440 test : MICROMEM            27 27 == 00000000
446 asig.: DI_LATCH             0  7 := 00000026
447 asig.: DD_LATCH             0  7 := 00000026
452 test : MICROMEM             8 10 == 00000003
460 asig.: ALU_IR               0  7 := 00000000
461 asig.: ALU_IS               0  7 := 00000020
462 asig.: ALU_DR               0  7 := 00000000
463 asig.: ALU_DS               0  7 := 00000020
486 test : MICROMEM            15 15 == 00000000
492 asig.: CN1                  0  0 := 00000000
493 asig.: CN2                  0  0 := 00000000
498 test : MICROMEM            12 14 == 00000003
548 asig.: SALIDA_ALUI          0  8 := 00000020
549 asig.: SALIDA_ALUD          0  8 := 00000020
572 test : MICROMEM            37 38 == 00000000
573 test : MICROMEM            16 18 == 00000001
575 asig.: BUS_I                0  8 := 00000020
576 asig.: BUS_D                0  8 := 00000020
678 test : MICROMEM            42 42 == 00000001
680 test : MICROMEM            34 34 == 00000001
684 asig.: LAT_I                0  7 := 00000020
685 asig.: LAT_D                0  7 := 00000020
688 asig.: PAR_TEST             0  3 := 00000000
690 test : SALIDA_ALUI          0  0 == 00000000
691 test : SALIDA_ALUI          1  1 == 00000000
692 test : SALIDA_ALUI          2  2 == 00000000
693 test : SALIDA_ALUI          3  3 == 00000000
694 test : SALIDA_ALUI          4  4 == 00000000
695 test : SALIDA_ALUI          5  5 == 00000001
695 asig.: PAR_TEST             0  3 := 00000001
696 test : SALIDA_ALUI          6  6 == 00000000
697 test : SALIDA_ALUI          7  7 == 00000000
699 test : PAR_TEST             0  0 == 00000001
700 test : PAR_TEST             0  0 == 00000001
700 asig.: PARITY_REG           0  0 := 00000000
706 test : MICROMEM            20 22 == 00000004
735 test : MICROMEM            43 43 == 00000001
740 test : MICROMEM            32 32 == 00000001
742 test : MICROMEM            30 30 == 00000001
744 test : MICROMEM            36 36 == 00000000
748 test : MICROMEM            44 47 == 0000000A
769 asig.: TEMP1                0  0 := 00000001
770 asig.: TEMP2                0  0 := 00000000
771 asig.: TEMP3                0  0 := 00000000
772 test : TEMP3                0  0 == 00000000
774 asig.: CC                   0  0 := 00000001
781 test : MICROMEM            48 51 == 00000003
791 test : CC                   0  0 == 00000001
792 test : CC                   0  0 == 00000001
792 asig.: MICRO_PC             0 11 := 000000AE
```

paso 7.

```
  236 test : MICROMEM          26 26 == 00000000
  391 test : MICROMEM          24 25 == 00000000
  393 asig.: AIZ                0  2 := 00000007
  394 asig.: AIZ                3  3 := 00000000
  395 asig.: ADER               0  0 := 00000000
  396 asig.: ADER               1  3 := 00000003
  397 asig.: BIZ                0  2 := 00000004
  398 asig.: BIZ                3  3 := 00000000
  399 asig.: BDER               0  0 := 00000001
  400 asig.: BDER               1  3 := 00000002
  430 test : MICROMEM          37 38 == 00000000
  440 test : MICROMEM          27 27 == 00000000
  446 asig.: DI_LATCH           0  7 := 00000020
  447 asig.: DD_LATCH           0  7 := 00000020
  452 test : MICROMEM           8 10 == 00000003
  460 asig.: ALU_IR             0  7 := 00000000
  461 asig.: ALU_IS             0  7 := 00000020
  462 asig.: ALU_DR             0  7 := 00000000
  463 asig.: ALU_DS             0  7 := 00000020
  486 test : MICROMEM          15 15 == 00000000
  492 asig.: CN1                0  0 := 00000000
  493 asig.: CN2                0  0 := 00000000
  498 test : MICROMEM          12 14 == 00000003
  548 asig.: SALIDA_ALUI        0  8 := 00000020
  549 asig.: SALIDA_ALUD        0  8 := 00000020
  572 test : MICROMEM          37 38 == 00000000
  573 test : MICROMEM          16 18 == 00000001
  575 asig.: BUS_I              0  8 := 00000020
  576 asig.: BUS_D              0  8 := 00000020
  678 test : MICROMEM          42 42 == 00000001
  680 test : MICROMEM          34 34 == 00000001
  684 asig.: LAT_I              0  7 := 00000020
  685 asig.: LAT_D              0  7 := 00000020
  688 asig.: PAR_TEST           0  3 := 00000000
  690 test : SALIDA_ALUI        0  0 == 00000000
  691 test : SALIDA_ALUI        1  1 == 00000000
  692 test : SALIDA_ALUI        2  2 == 00000000
  693 test : SALIDA_ALUI        3  3 == 00000000
  694 test : SALIDA_ALUI        4  4 == 00000000
  695 test : SALIDA_ALUI        5  5 == 00000001
  695 asig.: PAR_TEST           0  3 := 00000001
  696 test : SALIDA_ALUI        6  6 == 00000000
  697 test : SALIDA_ALUI        7  7 == 00000000
  699 test : PAR_TEST           0  0 == 00000001
  700 test : PAR_TEST           0  0 == 00000001
  700 asig.: PARITY_REG         0  0 := 00000000
  706 test : MICROMEM          20 22 == 00000004
  735 test : MICROMEM          43 43 == 00000001
  740 test : MICROMEM          32 32 == 00000001
  742 test : MICROMEM          30 30 == 00000001
  744 test : MICROMEM          36 36 == 00000001
  744 asig.: ADR_MEM            0 15 := 00000008
```

```
  748 test : MICROMEM              44 47 == 0000000F
  757 asig.: CC                     0  0 := 00000000
  781 test : MICROMEM              48 51 == 00000003
  791 test : CC                     0  0 == 00000000
  791 asig.: MICRO_PC               0 11 := 00000007
  792 test : CC                     0  0 == 00000000

7. paso : fin normal
SIM> AM AIMEM 7
AIMEM                : AIZ
[00000007]   26
SIM> AM ADMEM 7
ADMEM                : ADER
[00000006]   26
SIM> F
FIN: está seguro ? (S/N) Fin de la simulación
Tiempo de procesado :     20.9 segundos

                  SIMULADOR   RTL  SILOSS
                      EPFL - DET(UPC)
```

Ejecución para el grupo de tranferencia de datos del
programa:

```
            LXI HL, 20C2
            LXI DE, 1D1E
            XCHG
            MVI A,0A
            STAX DE
            LHLD 20C2
```

```
SIMULADOR SILOSS. Versión 2.4 DPTO. ELECTRONICA. MARZO 1.989
============================================================

*****EJECUCION:

Para visualizar el menú pulsar "?"


Lectura de definiciones
Lectura del micro-programa de la memoria MICROMEM
SIM> IM MEM_PROG 0000 21,C2,20,11,1E,1D,EB,3E,0A,12,2A,C2,20
SIM> E9
9. paso : fin normal
SIM> R
ADRESS_REG          = 00000001
ADR_MAPPING         = 00000021
ADR_MEM             = 00000000
DATA_REG            = 00000000
INPUT_DATA          = 00000021
MASK_REG            = 00000007
MICRO_PC            = 00000025
REG_INST            = 00000021
STATUS_REG          = 00000000
UNO                 = 00000001
SIM> E8
8. paso : fin normal
SIM> R
ADRESS_REG          = 00000004
ADR_MAPPING         = 00000011
ADR_MEM             = 00000003
DATA_REG            = 00000000
INPUT_DATA          = 00000011
MASK_REG            = 00000007
MICRO_PC            = 0000001F
REG_INST            = 00000011
STATUS_REG          = 00000000
UNO                 = 00000001
SIM> E7
7. paso : fin normal
SIM> AM AIMEM 0 10
AIMEM               : AIZ
[00000000]   00
[00000001]   00
[00000002]   1D
[00000003]   1E
[00000004]   20
[00000005]   C2
[00000006]   00
```

```
[00000007]   00
[00000008]   00
[00000009]   00
[0000000A]   00
[0000000B]   00
[0000000C]   60
[0000000D]   06
[0000000E]   00
[0000000F]   00
SIM> AM ADMEM 0 10
ADMEM                : ADER
[00000000]   00
[00000001]   00
[00000002]   1E
[00000003]   1D
[00000004]   C2
[00000005]   20
[00000006]   00
[00000007]   00
[00000008]   00
[00000009]   00
[0000000A]   00
[0000000B]   00
[0000000C]   06
[0000000D]   60
[0000000E]   00
[0000000F]   06
SIM> A MICRO_PC
MICRO_PC             = 008
SIM> E
1. paso : fin normal
SIM> A MICRO_PC
MICRO_PC             = 064
SIM> ED5
paso 1.

  236 test : MICROMEM            26 26 == 00000001
  238 test : MICROMEM            24 25 == 00000003
  264 asig.: AIZ                  0  3 := 00000004
  265 asig.: ADER                 0  3 := 00000004
  266 asig.: BIZ                  0  3 := 0000000A
  267 asig.: BDER                 0  3 := 0000000A
  271 asig.: A_LATCH              0  7 := 000000C2
  272 asig.: A_LATCH              8 15 := 00000020
  273 asig.: B_LATCH              0  7 := 00000000
  274 asig.: B_LATCH              8 15 := 00000000
  278 test : MICROMEM             8 10 == 00000004
  288 asig.: ALU_R                0 15 := 00000000
  289 asig.: ALU_S                0 15 := 000020C2
  302 test : MICROMEM            15 15 == 00000000
  304 asig.: CN                   0  0 := 00000000
  308 test : MICROMEM            12 14 == 00000003
  333 asig.: SALIDA_ALU           0 16 := 000020C2
  348 test : MICROMEM            37 38 == 00000000
```

```
 350 test : MICROMEM          16 18 == 00000003
 362 asig.: ADER               0  3 := 0000000A
 363 asig.: AIZ                0  3 := 0000000A
 364 asig.: BDMEM              0  7 := 000000C2
 365 asig.: ADMEM              0  7 := 000000C2
 366 asig.: BIMEM              0  7 := 00000020
 367 asig.: AIMEM              0  7 := 00000020
 368 asig.: Y_BUS              0 16 := 000020C2
 375 test : MICROMEM          42 42 == 00000001
 377 test : MICROMEM          27 27 == 00000000
 379 asig.: D_LATCH            0  7 := 00000020
 380 asig.: D_LATCH            8 15 := 000000C2
 384 test : MICROMEM          35 35 == 00000001
 706 test : MICROMEM          20 22 == 00000004
 735 test : MICROMEM          43 43 == 00000001
 740 test : MICROMEM          32 32 == 00000001
 742 test : MICROMEM          30 30 == 00000001
 744 test : MICROMEM          36 36 == 00000000
 748 test : MICROMEM          44 47 == 0000000E
 758 asig.: CC                 0  0 := 00000001
 781 test : MICROMEM          48 51 == 0000000E
 782 asig.: MICRO_PC           0 11 := 00000065
```

paso 2.

```
 236 test : MICROMEM          26 26 == 00000001
 238 test : MICROMEM          24 25 == 00000003
 264 asig.: AIZ                0  3 := 00000002
 265 asig.: ADER               0  3 := 00000002
 266 asig.: BIZ                0  3 := 00000004
 267 asig.: BDER               0  3 := 00000004
 271 asig.: A_LATCH            0  7 := 0000001E
 272 asig.: A_LATCH            8 15 := 0000001D
 273 asig.: B_LATCH            0  7 := 000000C2
 274 asig.: B_LATCH            8 15 := 00000020
 278 test : MICROMEM           8 10 == 00000004
 288 asig.: ALU_R              0 15 := 00000000
 289 asig.: ALU_S              0 15 := 00001D1E
 302 test : MICROMEM          15 15 == 00000000
 304 asig.: CN                 0  0 := 00000000
 308 test : MICROMEM          12 14 == 00000003
 333 asig.: SALIDA_ALU         0 16 := 00001D1E
 348 test : MICROMEM          37 38 == 00000000
 350 test : MICROMEM          16 18 == 00000003
 362 asig.: ADER               0  3 := 00000004
 363 asig.: AIZ                0  3 := 00000004
 364 asig.: BDMEM              0  7 := 0000001E
 365 asig.: ADMEM              0  7 := 0000001E
 366 asig.: BIMEM              0  7 := 0000001D
 367 asig.: AIMEM              0  7 := 0000001D
 368 asig.: Y_BUS              0 16 := 00001D1E
 375 test : MICROMEM          42 42 == 00000001
 377 test : MICROMEM          27 27 == 00000000
 379 asig.: D_LATCH            0  7 := 0000001D
```

```
380 asig.: D_LATCH          8 15 := 0000001E
384 test : MICROMEM        35 35 == 00000001
706 test : MICROMEM        20 22 == 00000004
735 test : MICROMEM        43 43 == 00000001
740 test : MICROMEM        32 32 == 00000001
742 test : MICROMEM        30 30 == 00000001
744 test : MICROMEM        36 36 == 00000000
748 test : MICROMEM        44 47 == 0000000E
758 asig.: CC               0  0 := 00000001
781 test : MICROMEM        48 51 == 0000000E
782 asig.: MICRO_PC         0 11 := 00000066
```

paso 3.

```
236 test : MICROMEM        26 26 == 00000001
238 test : MICROMEM        24 25 == 00000003
264 asig.: AIZ              0  3 := 00000002
265 asig.: ADER             0  3 := 00000002
266 asig.: BIZ              0  3 := 00000005
267 asig.: BDER             0  3 := 00000005
271 asig.: A_LATCH          0  7 := 0000001E
272 asig.: A_LATCH          8 15 := 0000001D
273 asig.: B_LATCH          0  7 := 00000020
274 asig.: B_LATCH          8 15 := 000000C2
278 test : MICROMEM         8 10 == 00000007
296 asig.: ALU_R            0 15 := 00001E1D
297 asig.: ALU_S            0 15 := 00000000
302 test : MICROMEM        15 15 == 00000000
304 asig.: CN               0  0 := 00000000
308 test : MICROMEM        12 14 == 00000003
333 asig.: SALIDA_ALU       0 16 := 00001E1D
348 test : MICROMEM        37 38 == 00000000
350 test : MICROMEM        16 18 == 00000002
353 asig.: ADER             0  3 := 00000005
354 asig.: AIZ              0  3 := 00000005
355 asig.: BDMEM            0  7 := 0000001D
356 asig.: ADMEM            0  7 := 0000001D
357 asig.: BIMEM            0  7 := 0000001E
358 asig.: AIMEM            0  7 := 0000001E
359 asig.: Y_BUS            0 15 := 00001D1E
375 test : MICROMEM        42 42 == 00000001
377 test : MICROMEM        27 27 == 00000000
379 asig.: D_LATCH          0  7 := 0000001D
380 asig.: D_LATCH          8 15 := 0000001E
384 test : MICROMEM        35 35 == 00000001
706 test : MICROMEM        20 22 == 00000004
735 test : MICROMEM        43 43 == 00000001
740 test : MICROMEM        32 32 == 00000001
742 test : MICROMEM        30 30 == 00000001
744 test : MICROMEM        36 36 == 00000000
748 test : MICROMEM        44 47 == 0000000E
758 asig.: CC               0  0 := 00000001
781 test : MICROMEM        48 51 == 0000000E
782 asig.: MICRO_PC         0 11 := 00000067
```

paso 4.

```
  236 test : MICROMEM          26 26 == 00000001
  238 test : MICROMEM          24 25 == 00000003
  264 asig.: AIZ                0  3 := 0000000A
  265 asig.: ADER               0  3 := 0000000A
  266 asig.: BIZ                0  3 := 00000002
  267 asig.: BDER               0  3 := 00000002
  271 asig.: A_LATCH            0  7 := 000000C2
  272 asig.: A_LATCH            8 15 := 00000020
  273 asig.: B_LATCH            0  7 := 0000001E
  274 asig.: B_LATCH            8 15 := 0000001D
  278 test : MICROMEM           8 10 == 00000004
  288 asig.: ALU_R              0 15 := 00000000
  289 asig.: ALU_S              0 15 := 000020C2
  302 test : MICROMEM          15 15 == 00000000
  304 asig.: CN                 0  0 := 00000000
  308 test : MICROMEM          12 14 == 00000003
  333 asig.: SALIDA_ALU         0 16 := 000020C2
  348 test : MICROMEM          37 38 == 00000000
  350 test : MICROMEM          16 18 == 00000003
  362 asig.: ADER               0  3 := 00000002
  363 asig.: AIZ                0  3 := 00000002
  364 asig.: BDMEM              0  7 := 000000C2
  365 asig.: ADMEM              0  7 := 000000C2
  366 asig.: BIMEM              0  7 := 00000020
  367 asig.: AIMEM              0  7 := 00000020
  368 asig.: Y_BUS              0 16 := 000020C2
  375 test : MICROMEM          42 42 == 00000001
  377 test : MICROMEM          27 27 == 00000000
  379 asig.: D_LATCH            0  7 := 00000020
  380 asig.: D_LATCH            8 15 := 000000C2
  384 test : MICROMEM          35 35 == 00000001
  706 test : MICROMEM          20 22 == 00000004
  735 test : MICROMEM          43 43 == 00000001
  740 test : MICROMEM          32 32 == 00000001
  742 test : MICROMEM          30 30 == 00000001
  744 test : MICROMEM          36 36 == 00000000
  748 test : MICROMEM          44 47 == 0000000E
  758 asig.: CC                 0  0 := 00000001
  781 test : MICROMEM          48 51 == 0000000E
  782 asig.: MICRO_PC           0 11 := 00000068
```

paso 5.

```
  236 test : MICROMEM          26 26 == 00000001
  238 test : MICROMEM          24 25 == 00000003
  264 asig.: AIZ                0  3 := 00000002
  265 asig.: ADER               0  3 := 00000002
  266 asig.: BIZ                0  3 := 00000003
  267 asig.: BDER               0  3 := 00000003
  271 asig.: A_LATCH            0  7 := 000000C2
  272 asig.: A_LATCH            8 15 := 00000020
  273 asig.: B_LATCH            0  7 := 0000001D
```

```
  274 asig.: B_LATCH                8 15 := 0000001E
  278 test : MICROMEM               8 10 == 00000007
  296 asig.: ALU_R                  0 15 := 0000C220
  297 asig.: ALU_S                  0 15 := 00000000
  302 test : MICROMEM              15 15 == 00000000
  304 asig.: CN                     0  0 := 00000000
  308 test : MICROMEM              12 14 == 00000003
  333 asig.: SALIDA_ALU             0 16 := 0000C220
  348 test : MICROMEM              37 38 == 00000000
  350 test : MICROMEM              16 18 == 00000002
  353 asig.: ADER                   0  3 := 00000003
  354 asig.: AIZ                    0  3 := 00000003
  355 asig.: BDMEM                  0  7 := 00000020
  356 asig.: ADMEM                  0  7 := 00000020
  357 asig.: BIMEM                  0  7 := 000000C2
  358 asig.: AIMEM                  0  7 := 000000C2
  359 asig.: Y_BUS                  0 15 := 000020C2
  375 test : MICROMEM              42 42 == 00000001
  377 test : MICROMEM              27 27 == 00000000
  379 asig.: D_LATCH                0  7 := 00000020
  380 asig.: D_LATCH                8 15 := 000000C2
  384 test : MICROMEM              35 35 == 00000001
  706 test : MICROMEM              20 22 == 00000004
  735 test : MICROMEM              43 43 == 00000001
  740 test : MICROMEM              32 32 == 00000001
  742 test : MICROMEM              30 30 == 00000001
  744 test : MICROMEM              36 36 == 00000001
  744 asig.: ADR_MEM                0 15 := 00000007
  748 test : MICROMEM              44 47 == 0000000F
  757 asig.: CC                     0  0 := 00000000
  781 test : MICROMEM              48 51 == 00000003
  791 test : CC                     0  0 == 00000000
  791 asig.: MICRO_PC               0 11 := 00000007
  792 test : CC                     0  0 == 00000000

5. paso : fin normal
SIM> AM AIMEM 0 10
AIMEM                  : AIZ
[00000000]   00
[00000001]   00
[00000002]   20
[00000003]   C2
[00000004]   1D
[00000005]   1E
[00000006]   00
[00000007]   00
[00000008]   00
[00000009]   00
[0000000A]   20
[0000000B]   00
[0000000C]   60
[0000000D]   06
[0000000E]   00
[0000000F]   00
```

```
SIM> AM ADMEM 0 10
ADMEM                 : ADER
[00000000]    00
[00000001]    00
[00000002]    C2
[00000003]    20
[00000004]    1E
[00000005]    1D
[00000006]    00
[00000007]    00
[00000008]    00
[00000009]    00
[0000000A]    C2
[0000000B]    00
[0000000C]    06
[0000000D]    60
[0000000E]    00
[0000000F]    07
SIM> R
ADRESS_REG            = 00000007
ADR_MAPPING           = 000000EB
ADR_MEM               = 00000007
DATA_REG              = 00000000
INPUT_DATA            = 000000EB
MASK_REG              = 00000007
MICRO_PC              = 00000007
REG_INST              = 000000EB
STATUS_REG            = 00000000
UNO                   = 00000001
SIM> E2
2. paso : fin normal
SIM> ED3
paso 1.

  236 test : MICROMEM              26 26 == 00000001
  238 test : MICROMEM              24 25 == 00000003
  264 asig.: AIZ                    0  3 := 0000000F
  265 asig.: ADER                   0  3 := 0000000F
  266 asig.: BIZ                    0  3 := 0000000F
  267 asig.: BDER                   0  3 := 0000000F
  271 asig.: A_LATCH                0  7 := 00000008
  272 asig.: A_LATCH                8 15 := 00000000
  273 asig.: B_LATCH                0  7 := 00000008
  274 asig.: B_LATCH                8 15 := 00000000
  278 test : MICROMEM               8 10 == 00000004
  288 asig.: ALU_R                  0 15 := 00000000
  289 asig.: ALU_S                  0 15 := 00000008
  302 test : MICROMEM              15 15 == 00000000
  304 asig.: CN                     0  0 := 00000001
  308 test : MICROMEM              12 14 == 00000000
  310 asig.: TEMP                   0 16 := 00000008
  311 test : CN                     0  0 == 00000001
  313 asig.: SALIDA_ALU             0 16 := 00000009
  348 test : MICROMEM              37 38 == 00000000
```

```
 350 test : MICROMEM          16 18 == 00000003
 362 asig.: ADER               0  3 := 0000000F
 363 asig.: AIZ                0  3 := 0000000F
 364 asig.: BDMEM              0  7 := 00000009
 365 asig.: ADMEM              0  7 := 00000009
 366 asig.: BIMEM              0  7 := 00000000
 367 asig.: AIMEM              0  7 := 00000000
 368 asig.: Y_BUS              0 16 := 00000009
 375 test : MICROMEM          42 42 == 00000001
 377 test : MICROMEM          27 27 == 00000000
 379 asig.: D_LATCH            0  7 := 00000000
 380 asig.: D_LATCH            8 15 := 00000009
 384 test : MICROMEM          35 35 == 00000000
 384 asig.: ADRESS_REG         0 15 := 00000009
 706 test : MICROMEM          20 22 == 00000004
 735 test : MICROMEM          43 43 == 00000001
 740 test : MICROMEM          32 32 == 00000001
 742 test : MICROMEM          30 30 == 00000001
 744 test : MICROMEM          36 36 == 00000001
 744 asig.: ADR_MEM            0 15 := 00000008
 748 test : MICROMEM          44 47 == 0000000E
 758 asig.: CC                 0  0 := 00000001
 781 test : MICROMEM          48 51 == 0000000E
 782 asig.: MICRO_PC           0 11 := 00000014
```

paso 2.

```
 236 test : MICROMEM          26 26 == 00000000
 391 test : MICROMEM          24 25 == 00000000
 393 asig.: AIZ                0  2 := 00000006
 394 asig.: AIZ                3  3 := 00000000
 395 asig.: ADER               0  0 := 00000001
 396 asig.: ADER               1  3 := 00000003
 397 asig.: BIZ                0  2 := 00000007
 398 asig.: BIZ                3  3 := 00000000
 399 asig.: BDER               0  0 := 00000000
 400 asig.: BDER               1  3 := 00000003
 430 test : MICROMEM          37 38 == 00000000
 440 test : MICROMEM          27 27 == 00000000
 446 asig.: DI_LATCH           0  7 := 0000001E
 447 asig.: DD_LATCH           0  7 := 0000001E
 452 test : MICROMEM           8 10 == 00000003
 460 asig.: ALU_IR             0  7 := 00000000
 461 asig.: ALU_IS             0  7 := 00000000
 462 asig.: ALU_DR             0  7 := 00000000
 463 asig.: ALU_DS             0  7 := 00000000
 486 test : MICROMEM          15 15 == 00000000
 492 asig.: CN1                0  0 := 00000000
 493 asig.: CN2                0  0 := 00000000
 498 test : MICROMEM          12 14 == 00000003
 548 asig.: SALIDA_ALUI        0  8 := 00000000
 549 asig.: SALIDA_ALUD        0  8 := 00000000
 572 test : MICROMEM          37 38 == 00000000
 573 test : MICROMEM          16 18 == 00000001
```

```
575 asig.: BUS_I                0  8 := 00000000
576 asig.: BUS_D                0  8 := 00000000
678 test : MICROMEM            42 42 == 00000001
680 test : MICROMEM            34 34 == 00000001
684 asig.: LAT_I                0  7 := 00000000
685 asig.: LAT_D                0  7 := 00000000
688 asig.: PAR_TEST             0  3 := 00000000
690 test : SALIDA_ALUI          0  0 == 00000000
691 test : SALIDA_ALUI          1  1 == 00000000
692 test : SALIDA_ALUI          2  2 == 00000000
693 test : SALIDA_ALUI          3  3 == 00000000
694 test : SALIDA_ALUI          4  4 == 00000000
695 test : SALIDA_ALUI          5  5 == 00000000
696 test : SALIDA_ALUI          6  6 == 00000000
697 test : SALIDA_ALUI          7  7 == 00000000
699 test : PAR_TEST             0  0 == 00000000
699 asig.: PARITY_REG           0  0 := 00000001
700 test : PAR_TEST             0  0 == 00000000
706 test : MICROMEM            20 22 == 00000004
735 test : MICROMEM            43 43 == 00000001
740 test : MICROMEM            32 32 == 00000001
742 test : MICROMEM            30 30 == 00000000
742 asig.: INPUT_DATA           0  7 := 0000000A
744 test : MICROMEM            36 36 == 00000000
748 test : MICROMEM            44 47 == 0000000E
758 asig.: CC                   0  0 := 00000001
781 test : MICROMEM            48 51 == 0000000E
782 asig.: MICRO_PC             0 11 := 00000015
```

paso 3.

```
236 test : MICROMEM            26 26 == 00000000
391 test : MICROMEM            24 25 == 00000001
403 asig.: AIZ                  0  3 := 00000007
404 asig.: ADER                 0  0 := 00000000
405 asig.: ADER                 1  3 := 00000003
406 asig.: BIZ                  0  2 := 00000007
407 asig.: BIZ                  3  3 := 00000000
408 asig.: BDER                 0  0 := 00000000
409 asig.: BDER                 1  3 := 00000003
430 test : MICROMEM            37 38 == 00000000
440 test : MICROMEM            27 27 == 00000001
442 asig.: DI_LATCH             0  7 := 0000000A
443 asig.: DD_LATCH             0  7 := 0000000A
452 test : MICROMEM             8 10 == 00000007
478 asig.: ALU_IR               0  7 := 0000000A
479 asig.: ALU_IS               0  7 := 00000000
480 asig.: ALU_DR               0  7 := 0000000A
481 asig.: ALU_DS               0  7 := 00000000
486 test : MICROMEM            15 15 == 00000000
492 asig.: CN1                  0  0 := 00000000
493 asig.: CN2                  0  0 := 00000000
498 test : MICROMEM            12 14 == 00000007
548 asig.: SALIDA_ALUI          0  8 := 0000000A
```

```
549 asig.: SALIDA_ALUD          0  8 := 0000000A
572 test : MICROMEM            37 38 == 00000000
573 test : MICROMEM            16 18 == 00000003
589 asig.: ADER                 0  3 := 00000006
590 asig.: AIZ                  0  3 := 00000007
591 asig.: BIMEM                0  7 := 0000000A
592 asig.: AIMEM                0  7 := 0000000A
593 asig.: BDMEM                0  7 := 0000000A
594 asig.: ADMEM                0  7 := 0000000A
595 asig.: BUS_I                0  8 := 0000000A
596 asig.: BUS_D                0  8 := 0000000A
678 test : MICROMEM            42 42 == 00000001
680 test : MICROMEM            34 34 == 00000001
684 asig.: LAT_I                0  7 := 0000000A
685 asig.: LAT_D                0  7 := 0000000A
688 asig.: PAR_TEST             0  3 := 00000000
690 test : SALIDA_ALUI          0  0 == 00000000
691 test : SALIDA_ALUI          1  1 == 00000001
691 asig.: PAR_TEST             0  3 := 00000001
692 test : SALIDA_ALUI          2  2 == 00000000
693 test : SALIDA_ALUI          3  3 == 00000001
693 asig.: PAR_TEST             0  3 := 00000002
694 test : SALIDA_ALUI          4  4 == 00000000
695 test : SALIDA_ALUI          5  5 == 00000000
696 test : SALIDA_ALUI          6  6 == 00000000
697 test : SALIDA_ALUI          7  7 == 00000000
699 test : PAR_TEST             0  0 == 00000000
699 asig.: PARITY_REG           0  0 := 00000001
700 test : PAR_TEST             0  0 == 00000000
706 test : MICROMEM            20 22 == 00000004
735 test : MICROMEM            43 43 == 00000001
740 test : MICROMEM            32 32 == 00000001
742 test : MICROMEM            30 30 == 00000001
744 test : MICROMEM            36 36 == 00000001
744 asig.: ADR_MEM              0 15 := 00000009
748 test : MICROMEM            44 47 == 0000000F
757 asig.: CC                   0  0 := 00000000
781 test : MICROMEM            48 51 == 00000003
791 test : CC                   0  0 == 00000000
791 asig.: MICRO_PC             0 11 := 00000007
792 test : CC                   0  0 == 00000000

3. pase : fin normal
SIM> AM BIMEM 0 10
BIMEM                  : BIZ
[00000000]   00
[00000001]   00
[00000002]   20
[00000003]   C2
[00000004]   1D
[00000005]   1E
[00000006]   00
[00000007]   0A
[00000008]   00
```

```
[00000009]   00
[0000000A]   20
[0000000B]   00
[0000000C]   60
[0000000D]   06
[0000000E]   00
[0000000F]   00
SIM> AM BDMEM 0 10
BDMEM                : BDER
[00000000]   00
[00000001]   00
[00000002]   C2
[00000003]   20
[00000004]   1E
[00000005]   1D
[00000006]   0A
[00000007]   00
[00000008]   00
[00000009]   00
[0000000A]   C2
[0000000B]   00
[0000000C]   06
[0000000D]   60
[0000000E]   00
[0000000F]   09
SIM> E2
2. paso : fin normal
SIM> A MICRO_PC
MICRO_PC             = 061
SIM> ED3
paso 1.

  236 test : MICROMEM          26 26 == 00000001
  238 test : MICROMEM          24 25 == 00000000
  240 asig.: AIZ                0  0 := 00000000
  241 asig.: AIZ                3  3 := 00000000
  242 asig.: AIZ                1  2 := 00000001
  243 asig.: ADER               0  3 := 00000002
  244 asig.: BIZ                0  3 := 00000002
  245 asig.: BDER               0  3 := 00000002
  271 asig.: A_LATCH            0  7 := 000000C2
  272 asig.: A_LATCH            8 15 := 00000020
  273 asig.: B_LATCH            0  7 := 000000C2
  274 asig.: B_LATCH            8 15 := 00000020
  278 test : MICROMEM           8 10 == 00000004
  288 asig.: ALU_R              0 15 := 00000000
  289 asig.: ALU_S              0 15 := 000020C2
  302 test : MICROMEM          15 15 == 00000000
  304 asig.: CN                 0  0 := 00000000
  308 test : MICROMEM          12 14 == 00000003
  333 asig.: SALIDA_ALU         0 16 := 000020C2
  348 test : MICROMEM          37 38 == 00000000
  350 test : MICROMEM          16 18 == 00000003
  362 asig.: ADER               0  3 := 00000002
```

```
363 asig.: AIZ              0  3 := 00000002
364 asig.: BDMEM            0  7 := 000000C2
365 asig.: ADMEM            0  7 := 000000C2
366 asig.: BIMEM            0  7 := 00000020
367 asig.: AIMEM            0  7 := 00000020
368 asig.: Y_BUS            0 16 := 000020C2
375 test : MICROMEM        42 42 == 00000001
377 test : MICROMEM        27 27 == 00000000
379 asig.: D_LATCH          0  7 := 00000020
380 asig.: D_LATCH          8 15 := 000000C2
384 test : MICROMEM        35 35 == 00000000
384 asig.: ADRESS_REG       0 15 := 000020C2
706 test : MICROMEM        20 22 == 00000004
735 test : MICROMEM        43 43 == 00000001
740 test : MICROMEM        32 32 == 00000001
742 test : MICROMEM        30 30 == 00000001
744 test : MICROMEM        36 36 == 00000000
748 test : MICROMEM        44 47 == 0000000E
758 asig.: CC               0  0 := 00000001
781 test : MICROMEM        48 51 == 0000000E
782 asig.: MICRO_PC         0 11 := 00000062
```

paso 2.

```
236 test : MICROMEM        26 26 == 00000000
391 test : MICROMEM        24 25 == 00000003
421 asig.: AIZ              0  3 := 00000007
422 asig.: ADER             0  0 := 00000000
423 asig.: ADER             1  3 := 00000003
424 asig.: BIZ              0  3 := 00000007
425 asig.: BDER             0  0 := 00000000
426 asig.: BDER             1  3 := 00000003
430 test : MICROMEM        37 38 == 00000000
440 test : MICROMEM        27 27 == 00000000
446 asig.: DI_LATCH         0  7 := 0000000A
447 asig.: DD_LATCH         0  7 := 0000000A
452 test : MICROMEM         8 10 == 00000004
466 asig.: ALU_IR           0  7 := 00000000
467 asig.: ALU_IS           0  7 := 0000000A
468 asig.: ALU_DR           0  7 := 00000000
469 asig.: ALU_DS           0  7 := 0000000A
486 test : MICROMEM        15 15 == 00000000
492 asig.: CN1              0  0 := 00000000
493 asig.: CN2              0  0 := 00000000
498 test : MICROMEM        12 14 == 00000003
548 asig.: SALIDA_ALUI      0  8 := 0000000A
549 asig.: SALIDA_ALUD      0  8 := 0000000A
572 test : MICROMEM        37 38 == 00000000
573 test : MICROMEM        16 18 == 00000003
589 asig.: ADER             0  3 := 00000006
590 asig.: AIZ              0  3 := 00000007
591 asig.: BIMEM            0  7 := 0000000A
592 asig.: AIMEM            0  7 := 0000000A
593 asig.: BDMEM            0  7 := 0000000A
```

```
594 asig.: ADMEM          0  7 := 0000000A
595 asig.: BUS_I          0  8 := 0000000A
596 asig.: BUS_D          0  8 := 0000000A
678 test : MICROMEM      42 42 == 00000001
680 test : MICROMEM      34 34 == 00000000
680 test : MICROMEM      23 23 == 00000000
681 asig.: DATA_REG       0  7 := 0000000A
684 asig.: LAT_I          0  7 := 0000000A
685 asig.: LAT_D          0  7 := 0000000A
688 asig.: PAR_TEST       0  3 := 00000000
690 test : SALIDA_ALUI    0  0 == 00000000
691 test : SALIDA_ALUI    1  1 == 00000001
691 asig.: PAR_TEST       0  3 := 00000001
692 test : SALIDA_ALUI    2  2 == 00000000
693 test : SALIDA_ALUI    3  3 == 00000001
693 asig.: PAR_TEST       0  3 := 00000002
694 test : SALIDA_ALUI    4  4 == 00000000
695 test : SALIDA_ALUI    5  5 == 00000000
696 test : SALIDA_ALUI    6  6 == 00000000
697 test : SALIDA_ALUI    7  7 == 00000000
699 test : PAR_TEST       0  0 == 00000000
699 asig.: PARITY_REG     0  0 := 00000001
700 test : PAR_TEST       0  0 == 00000000
706 test : MICROMEM      20 22 == 00000004
735 test : MICROMEM      43 43 == 00000001
740 test : MICROMEM      32 32 == 00000001
742 test : MICROMEM      30 30 == 00000001
744 test : MICROMEM      36 36 == 00000001
744 asig.: ADR_MEM        0 15 := 000020C2
748 test : MICROMEM      44 47 == 0000000E
758 asig.: CC             0  0 := 00000001
781 test : MICROMEM      48 51 == 0000000E
782 asig.: MICRO_PC       0 11 := 00000063
```

paso 3.

```
236 test : MICROMEM      26 26 == 00000001
238 test : MICROMEM      24 25 == 00000003
264 asig.: AIZ            0  3 := 0000000F
265 asig.: ADER           0  3 := 0000000F
266 asig.: BIZ            0  3 := 0000000F
267 asig.: BDER           0  3 := 0000000F
271 asig.: A_LATCH        0  7 := 0000000A
272 asig.: A_LATCH        8 15 := 00000000
273 asig.: B_LATCH        0  7 := 0000000A
274 asig.: B_LATCH        8 15 := 00000000
278 test : MICROMEM       8 10 == 00000004
288 asig.: ALU_R          0 15 := 00000000
289 asig.: ALU_S          0 15 := 0000000A
302 test : MICROMEM      15 15 == 00000000
304 asig.: CN             0  0 := 00000000
308 test : MICROMEM      12 14 == 00000003
333 asig.: SALIDA_ALU     0 16 := 0000000A
348 test : MICROMEM      37 38 == 00000000
```

```
  350 test : MICROMEM              16 18 == 00000003
  362 asig.: ADER                   0  3 := 0000000F
  363 asig.: AIZ                    0  3 := 0000000F
  364 asig.: BDMEM                  0  7 := 0000000A
  365 asig.: ADMEM                  0  7 := 0000000A
  366 asig.: BIMEM                  0  7 := 00000000
  367 asig.: AIMEM                  0  7 := 00000000
  368 asig.: Y_BUS                  0 16 := 0000000A
  375 test : MICROMEM              42 42 == 00000001
  377 test : MICROMEM              27 27 == 00000000
  379 asig.: D_LATCH                0  7 := 00000000
  380 asig.: D_LATCH                8 15 := 0000000A
  384 test : MICROMEM              35 35 == 00000000
  384 asig.: ADRESS_REG             0 15 := 0000000A
  706 test : MICROMEM              20 22 == 00000004
  735 test : MICROMEM              43 43 == 00000001
  740 test : MICROMEM              32 32 == 00000000
  740 asig.: MEM_PROG               0  7 := 0000000A
  742 test : MICROMEM              30 30 == 00000001
  744 test : MICROMEM              36 36 == 00000000
  748 test : MICROMEM              44 47 == 0000000F
  757 asig.: CC                     0  0 := 00000000
  781 test : MICROMEM              48 51 == 00000003
  791 test : CC                     0  0 == 00000000
  791 asig.: MICRO_PC               0 11 := 00000006
  792 test : CC                     0  0 == 00000000

3. paso : fin normal
SIM> AM MEM_PROG 20C2
MEM_PROG              : ADR_MEM
[000020C2]   0A
SIM> E3
3. paso : fin normal
SIM> A MICRO_PC
MICRO_PC               = 045
SIM> E 0D
13. paso : fin normal
SIM> AM AIMEM 0 10
AIMEM                 : AIZ
[00000000]   00
[00000001]   00
[00000002]   20
[00000003]   C2
[00000004]   00
[00000005]   0A
[00000006]   00
[00000007]   0A
[00000008]   00
[00000009]   00
[0000000A]   20
[0000000B]   C2
[0000000C]   60
[0000000D]   06
[0000000E]   00
```

```
[0000000F]   00
SIM> AM ADMEM 0 10
ADMEM                : ADER
[00000000]   00
[00000001]   00
[00000002]   C2
[00000003]   20
[00000004]   0A
[00000005]   00
[00000006]   0A
[00000007]   00
[00000008]   00
[00000009]   00
[0000000A]   C3
[0000000B]   20
[0000000C]   06
[0000000D]   60
[0000000E]   00
[0000000F]   0D
SIM> F
FIN: está seguro ? (S/N) Fin de la simulación
Tiempo de procesado :     21.3 segundos
```

V. PROGRAMAS DE PRUEBA

5.1 CARACTERIZACION DE BENCHMARKS.-

Como programas de prueba para establecer una comparación real entre i8085 y CPA, se eligen algunos métodos de ordenación de *arrays*. Estos, denominados *métodos directos*, utilizan unos sencillos algoritmos de ordenación que requieren del orden de n^2 comparaciones. Dichas permutaciones se realizan utilizando el espacio ocupado por el *array* (ordenación *in situ*).

Los métodos directos se clasifican en tres categorias de acuerdo con el método base que utilizan:

1. Ordenación por inserción.
2. Ordenación por selección.
3. Ordenación por intercambio.

Estos tres principios básicos seran objeto de exámen en lo que sigue.

5.1.1 Ordenación por inserción directa.

En este método los elementos del *array* (ítems) se dividen conceptualmente en una secuencia destino $a_1,\ldots,$ a_{i-1} y una secuencia origen $a_i,\ldots,$ a_n. En cada paso, empezando con i=2, e incrementando i de uno en uno , se toma el elememnto i de la secuencia origen y se transfiere a la secuencia destino insertándolo en el sitio apropiado.

Claves iniciales	44	55	12	42	94	18	06	67
$i = 2$	44	55	12	42	94	18	06	67
$i = 3$	12	44	55	42	94	18	06	67
$i = 4$	12	42	44	55	94	18	06	67
$i = 5$	12	42	44	55	94	18	06	67
$i = 6$	12	18	42	44	55	94	06	67
$i = 7$	06	12	18	42	44	55	94	67
$i = 8$	06	12	18	42	44	55	67	94

Tabla . Proceso típico de ordenación por inserción directa.

5.1.2 Ordenación por selección directa.

En este método se selecciona el artículo con clave mínima y se intercambia con el primero, a_1.

A continuación se repiten estas operaciones con los ítems restantes $n-1, n-2, \ldots$, hasta que quede el artículo mayor.

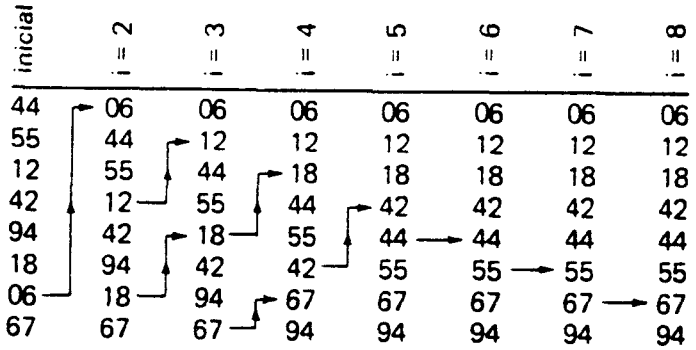
Claves iniciales	44	55	12	42	94	18	06	67
	06	55	12	42	94	18	44	67
	06	12	55	42	94	18	44	67
	06	12	18	42	94	55	44	67
	06	12	18	42	94	55	44	67
	06	12	18	42	44	55	94	67
	06	12	18	42	44	55	94	67
	06	12	18	42	44	55	67	94

Tabla . Ejemplo de proceso de ordenación por selección directa.

5.1.3 Ordenación por intercambio directo.

El algoritmo de intercambio directo se basa en el principio de comparar e **intercambiar** pares de ítems adyacentes hasta que estén todos ordenados.

Se hacen repetidas pasadas sobre el *array*, moviendo en cada una el elemento de clave mínima hasta el extremo izquierdo del *array*, como en los métodos anteriores aunque, en este caso, se mira el *array* como si estuviera en posición vertical en vez de horizontal. Este método se conoce generalmente como *método de la burbuja*.



inicial	i = 2	i = 3	i = 4	i = 5	i = 6	i = 7	i = 8
44	06	06	06	06	06	06	06
55	44	12	12	12	12	12	12
12	55	44	18	18	18	18	18
42	12	55	44	42	42	42	42
94	42	18	55	44	44	44	44
18	94	42	42	55	55	55	55
06	18	94	67	67	67	67	67
67	67	67	94	94	94	94	94

Tabla . Un ejemplo de ordenación por el método de la burbuja.

5.1.4 Análisis de los métodos descritos.

El número de comparaciones entre claves, C, y el número de movimientos M, particularizados para ocho elementos (n = 8) es el siguiente:

Inserción directa

$Cmin = n - 1 = 7$ $\quad$ $Mmin = 2(n - 1) = 14$

$Cmed = \frac{1}{4}(n^2 + n - 2) = 17.5$ $\quad$ $Mmed = \frac{1}{4}(n^2 + 9n - 10) = 31.5$

$Cmax = \frac{1}{2}(n^2 + n) - 1 = 35$ $\quad$ $Mmax = \frac{1}{2}(n^2 + 3n - 4) = 42$

Selección directa

$C = \frac{1}{2}(n^2 - n) = 28$ $\quad$ $Mmin = 3(n - 1) = 21$

Método de la burbuja

$C = \frac{1}{2}(n^2 - n) = 28$ $\quad$ $Mmed = \frac{3}{4}(n^2 - n) = 42$

$Mmax = \frac{3}{2}(n^2 - n) = 84$

Como conclusión, este análisis demuestra que, en general, el algoritmo de selección directa es preferible al de inserción directa, aunque en los casos en que las claves están inicialmente ordenadas o casi ordenadas, este último puede ser algo más rápido. La ordenación por intercambio es inferior a la ordenación por selección o por inserción.

5.2 OBTENCION DE PROGRAMAS Y EMULACION

Para obtener estos benchmarks, escritos en PASCAL, en lenguaje objeto para el 8085 se ha utilizado el comnpilador de PLM/80, sobre el sistema de desarrollo Intellec.

PLM/80 es un lenguaje modular elemental, con una estructura similar al PASCAL estándar, con lo cual resulta sencillo pasar programas no muy complejos escritos en PASCAL a este lenguaje, con objeto de obtener en el proceso de compilación un listado del programa en lenguaje ensamblador para el 8085, aplicable directamente al simulador.

```
$PAGELENGHT(50)
$CODE
SELECCIONDIRECTA: DO;
      DECLARE TOP BYTE INITIAL(8);
      DECLARE SORTLIST(8) BYTE INITIAL(44,55,12,42,94,18,06,67);
CO:   PROCEDURE (CHAR) EXTERNAL;
      DECLARE CHAR BYTE;
      END CO;
EXIT: PROCEDURE EXTERNAL;
      END EXIT;
      DECLARE (I,J,K,X,CHAR) BYTE;
      DO I=1 TO TOP-1;
      K=I;
      X=SORTLIST(I);
         DO J=I+1 TO TOP;
            IF SORTLIST(J)<X THEN
            DO;
              K=J;
              X=SORTLIST(J);
            END;
         END;
         SORTLIST(K)=SORTLIST(I);
         SORTLIST(I)=X;
         END;
      DO I=1 TO 8;
      CALL CO (CHAR:=SORTLIST(I));
      CALL CO (0AH);
      CALL CO (0DH);
      END;
      CALL EXIT;
END SELECCIONDIRECTA;
```

```
PL/M-80 COMPILER

ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE SELECCIONDIRECTA
OBJECT MODULE PLACED IN SELEC.OBJ
COMPILER INVOKED BY:  PLM80 SELEC.SRC WORKFILES(:F0:,:F0:)


            $PAGELENGTH(50)
            $CODE
  1         SELECCIONDIRECTA: DO;
  2    1          DECLARE TOP BYTE INITIAL(8);
  3    1          DECLARE SORTLIST(8) BYTE INITIAL(44,55,12,42,94,18,06,6
  4    1    CO:   PROCEDURE (CHAR) EXTERNAL;
  5    2          DECLARE CHAR BYTE;
  6    2          END CO;
  7    1    EXIT: PROCEDURE EXTERNAL;
  8    2          END EXIT;
  9    1          DECLARE (I,J,K,X,CHAR) BYTE;
 10    1          DO I=1 TO TOP-1;
                                                  ; STATEMENT # 10
          0000  310000          LXI     SP,@STACK$ORIGIN
          0003  210900          LXI     H,I
          0006  3601            MVI     M,1H
                          @2:
          0008  3A0000          LDA     TOP
          000B  3D              DCR     A
          000C  210900          LXI     H,I
          000F  BE              CMP     M
          0010  DA8000          JC      @3
 11    2             K=I;
                                                  ; STATEMENT # 11
          0013  3A0900          LDA     I
          0016  320B00          STA     K
 12    2             X=SORTLIST(I);
                                                  ; STATEMENT # 12
          0019  4F              MOV     C,A
          001A  0600            MVI     B,0
          001C  210100          LXI     H,SORTLIST
          001F  09              DAD     B
          0020  7E              MOV     A,M
          0021  320C00          STA     X
 13    2                DO J=I+1 TO TOP;
                                                  ; STATEMENT # 13
          0024  0C              INR     C
          0025  79              MOV     A,C
          0026  320A00          STA     J
                          @4:
          0029  3A0000          LDA     TOP
```

```
                002C  210A00          LXI     H,J
                002F  BE              CMP     M
                0030  DA5C00          JC      @5
14    3                     IF SORTLIST(J)<X THEN
                                                      ; STATEMENT # 14
                0033  2A0A00          LHLD    J
                0036  2600            MVI     H,0
                0038  010100          LXI     B,SORTLIST
                003B  09              DAD     B
                003C  7E              MOV     A,M
                003D  210C00          LXI     H,X
                0040  BE              CMP     M
                0041  D25500          JNC     @1
15    3                       DO;
16    4                   K=J;
                                                      ; STATEMENT # 16
                0044  3A0A00          LDA     J
                0047  320B00          STA     K
17    4                   X=SORTLIST(J);
                                                      ; STATEMENT # 17
                004A  4F              MOV     C,A
                004B  0600            MVI     B,0
                004D  210100          LXI     H,SORTLIST
                0050  09              DAD     B
                0051  7E              MOV     A,M
                0052  320C00          STA     X
18    4                 END;
                                                      ; STATEMENT # 18
                                @1:
19    3               END;
                                                      ; STATEMENT # 19
                0055  210A00          LXI     H,J
                0058  34              INR     M
                0059  C22900          JNZ     @4
                                @5:
20    2               SORTLIST(K)=SORTLIST(I);
                                                      ; STATEMENT # 20
                005C  2A0900          LHLD    I
                005F  2600            MVI     H,0
                0061  010100          LXI     B,SORTLIST
                0064  09              DAD     B
                0065  E5              PUSH    H         ; 1
                0066  2A0B00          LHLD    K
                0069  2600            MVI     H,0
                006B  09              DAD     B
                006C  D1              POP     D         ; 1
                006D  1A              LDAX    D
```

```
                006E  77              MOV     M,A
21    2             SORTLIST(I)=X;
                                                ; STATEMENT # 21
                006F  2A0900          LHLD    I
                0072  2600            MVI     H,0
                0074  09              DAD     B
                0075  3A0C00          LDA     X
                0078  77              MOV     M,A
22    2             END;
                                                ; STATEMENT # 22
                0079  210900          LXI     H,I
                007C  34              INR     M
                007D  C20800          JNZ     @2
                              @3:
23    1             DO I=1 TO 8;
                                                ; STATEMENT # 23
                0080  210900          LXI     H,I
                0083  3601            MVI     M,1H
                              @6:
                0085  3E08            MVI     A,8H
                0087  210900          LXI     H,I
                008A  BE              CMP     M
                008B  DAB000          JC      @7
24    2             CALL CO (CHAR:=SORTLIST(I));
                                                ; STATEMENT # 24
                008E  2A0900          LHLD    I
                0091  2600            MVI     H,0
                0093  010100          LXI     B,SORTLIST
                0096  09              DAD     B
                0097  7E              MOV     A,M
                0098  320D00          STA     CHAR
                009B  4F              MOV     C,A
                009C  CD0000          CALL    CO
25    2             CALL CO (0AH);
                                                ; STATEMENT # 25
                009F  0E0A            MVI     C,0AH
                00A1  CD0000          CALL    CO
26    2             CALL CO (0DH);
                                                ; STATEMENT # 26
                00A4  0E0D            MVI     C,0DH
                00A6  CD0000          CALL    CO
27    2             END;
                                                ; STATEMENT # 27
                00A9  210900          LXI     H,I
                00AC  34              INR     M
                00AD  C28500          JNZ     @6
                              @7:
```

```
PL/M-80 COMPILER


 28    1              CALL EXIT;
                                                    ; STATEMENT # 28
           0080  CD0000           CALL     EXIT
 29    1       END SELECCIONDIRECTA;
                                                    ; STATEMENT # 29
           0083  FB               EI
           0084  76               HLT


10DULE INFORMATION:

     CODE AREA SIZE      = 0085H     181D
     VARIABLE AREA SIZE  = 000EH      14D
     MAXIMUM STACK SIZE  = 0002H       2D
     31 LINES READ
     0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION
```

```
$CODE
INSERCIONDIRECTA: DO;
      DECLARE TOP BYTE INITIAL(8);
      DECLARE SORTLIST(8) BYTE INITIAL (44,55,12,42,94,18,06,67
CO:   PROCEDURE (CHAR) EXTERNAL;
      DECLARE CHAR BYTE;
      END CO;
EXIT: PROCEDURE EXTERNAL;
      END EXIT;
  DECLARE (I,J,K,CHAR) BYTE;
  DO I=2 TO TOP;
     K=SORTLIST(I);
     SORTLIST(0)=K;
     J=I-1;
     DO WHILE K<SORTLIST(J);
        SORTLIST(J+1)=SORTLIST(J);
 J=J-1;
     END;
     SORTLIST(J+1)=K;
  END;
  DO I=1 TO 8;
  CALL CO (CHAR:=SORTLIST(I));
  CALL CO (0AH);
  CALL CO (0DH);
  END;
  CALL EXIT;
  END INSERCIONDIRECTA;
```

```
ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE INSERCIONDIRECTA
OBJECT MODULE PLACED IN INSER.OBJ
COMPILER INVOKED BY:  PLM80 INSER.SRC WORKFILES (:F0:,:F0:)


                $CODE
   1            INSERCIONDIRECTA: DO;
   2    1             DECLARE TOP BYTE INITIAL(8);
   3    1             DECLARE SORTLIST(8) BYTE INITIAL (44,55,12,42,94,18,06,
   4    1       CO:   PROCEDURE (CHAR) EXTERNAL;
   5    2             DECLARE CHAR BYTE;
   6    2             END CO;
   7    1       EXIT: PROCEDURE EXTERNAL;
   8    2             END EXIT;
   9    1         DECLARE (I,J,K,CHAR) BYTE;
  10    1         DO I=2 TO TOP;
                                                        ; STATEMENT # 10
          0000  310000          LXI     SP,@STACK$ORIGIN
          0003  210900          LXI     H,I
          0006  3602            MVI     M,2H
                          @1:
          0008  3A0000          LDA     TOP
          000B  210900          LXI     H,I
          000E  BE              CMP     M
          000F  DA6800          JC      @2
  11    2           K=SORTLIST(I);
                                                        ; STATEMENT # 11
          0012  2A0900          LHLD    I
          0015  2600            MVI     H,0
          0017  010100          LXI     B,SORTLIST
          001A  09              DAD     B
          001B  7E              MOV     A,M
          001C  320B00          STA     K
  12    2           SORTLIST(0)=K;
                                                        ; STATEMENT # 12
          001F  60              MOV     H,B
          0020  69              MOV     L,C
          0021  77              MOV     M,A
  13    2           J=I-1;
                                                        ; STATEMENT # 13
          0022  3A0900          LDA     I
          0025  3D              DCR     A
          0026  320A00          STA     J
  14    2           DO WHILE K<SORTLIST(J);
                                                        ; STATEMENT # 14
                          @3:
          0029  2A0A00          LHLD    J
          002C  2600            MVI     H,0
          002E  010100          LXI     B,SORTLIST
          0031  09              DAD     B
          0032  3A0B00          LDA     K
          0035  BE              CMP     M
          0036  D25400          JNC     @4
  15    3             SORTLIST(J+1)=SORTLIST(J);
                                                        ; STATEMENT # 15
          0039  2A0A00          LHLD    J
```

```
                003C  2600            MVI     H,0
                003E  010100          LXI     B,SORTLIST
                0041  09              DAD     B
                0042  E5              PUSH    H         ; 1
                0043  2A0A00          LHLD    J
                0046  2600            MVI     H,0
                0048  03              INX     B
                0049  09              DAD     B
                004A  D1              POP     D         ; 1
                004B  1A              LDAX    D
                004C  77              MOV     M,A
16    3           J=J-1;
                                                        ; STATEMENT # 16
                004D  210A00          LXI     H,J
                0050  35              DCR     M
17    3           END;
                                                        ; STATEMENT # 17
                0051  C32900          JMP     @3
                                @4:
18    2           SORTLIST(J+1)=K;
                                                        ; STATEMENT # 18
                0054  2A0A00          LHLD    J
                0057  2600            MVI     H,0
                0059  010200          LXI     B,SORTLIST+1H
                005C  09              DAD     B
                005D  3A0B00          LDA     K
                0060  77              MOV     M,A
19    2         END;
                                                        ; STATEMENT # 19
                0061  210900          LXI     H,I
                0064  34              INR     M
                0065  C20800          JNZ     @1
                                @2:
20    1         DO I=1 TO 8;
                                                        ; STATEMENT # 20
                0068  210900          LXI     H,I
                006B  3601            MVI     M,1H
                                @5:
                006D  3E08            MVI     A,8H
                006F  210900          LXI     H,I
                0072  BE              CMP     M
                0073  DA9800          JC      @6
21    2         CALL CO (CHAR:=SORTLIST(I));
                                                        ; STATEMENT # 21
                0076  2A0900          LHLD    I
                0079  2600            MVI     H,0
                007B  010100          LXI     B,SORTLIST
                007E  09              DAD     B
                007F  7E              MOV     A,M
                0080  320C00          STA     CHAR
                0083  4F              MOV     C,A
                0084  CD0000          CALL    CO
22    2         CALL CO (0AH);
                                                        ; STATEMENT # 22
                0087  0E0A            MVI     C,0AH
                0089  CD0000          CALL    CO
23    2         CALL CO (0DH);
```

```
                                                        ; STATEMENT # 23
            008C  0E0D              MVI     C,0DH
            008E  CD0000            CALL    CO
 24    2          END;
                                                        ; STATEMENT # 24
            0091  210900            LXI     H,I
            0094  34                INR     M
            0095  C26D00            JNZ     @5
                        @6:
 25    1          CALL EXIT;
                                                        ; STATEMENT # 25
            0098  CD0000            CALL    EXIT
 26    1          END INSERCIONDIRECTA;
                                                        ; STATEMENT # 26
            009B  FB                EI
            009C  76                HLT
```

MODULE INFORMATION:

```
     CODE AREA SIZE      = 009DH     157D
     VARIABLE AREA SIZE  = 000DH      13D
     MAXIMUM STACK SIZE  = 0002H       2D
     27 LINES READ
     0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION
```

```
$CODE

BUBBLESORT: DO;
 DECLARE TOP BYTE INITIAL (8);
 DECLARE SORTLIST (8) BYTE INITIAL  (44,55,12,42,94,18,06,67);
CO:  PROCEDURE (CHAR) EXTERNAL;
 DECLARE CHAR BYTE;
 END CO;
EXIT: PROCEDURE EXTERNAL;
 END EXIT;
 DECLARE (I,J,CHAR) BYTE;
 DO WHILE TOP>1;
  I=1;
  DO WHILE I<TOP;
  IF SORTLIST(I)>SORTLIST(I+1) THEN
  DO;
   J=SORTLIST(I);
   SORTLIST(I)=SORTLIST(I+1);
   SORTLIST(I+1)=J;
  END;
  I=I+1;
  END;
  TOP=TOP-1;
 END;
 DO I=1 TO 8;
 CALL CO (CHAR:=SORTLIST(I));
 CALL CO (0AH);
 CALL CO (0DH);
 END;
 CALL EXIT;
END BUBBLESORT;
```

```
ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE BUBBLESORT
OBJECT MODULE PLACED IN BUBB.OBJ
COMPILER INVOKED BY:  PLM80 BUBB.SRC WORKFILES(:F0:,:F0:)


                $CODE

  1             BUBBLESORT: DO;
  2    1            DECLARE TOP BYTE INITIAL (8);
  3    1            DECLARE SORTLIST (8) BYTE INITIAL  (44,55,12,42,94,18,06,
  4    1        CO:      PROCEDURE (CHAR) EXTERNAL;
  5    2            DECLARE CHAR BYTE;
  6    2            END CO;
  7    1        EXIT:    PROCEDURE EXTERNAL;
  8    2            END EXIT;
  9    1            DECLARE (I,J,CHAR) BYTE;
 10    1            DO WHILE TOP>1;
                                                    ; STATEMENT # 10
       0000  310000             LXI      SP,@STACK$ORIGIN
                          @2:
       0003  3E01               MVI      A,1H
       0005  210000             LXI      H,TOP
       0008  BE                 CMP      M
       0009  D26A00             JNC      @3
 11    2                 I=1;
                                                    ; STATEMENT # 11
       000C  210900             LXI      H,I
       000F  3601               MVI      M,1H
 12    2                 DO WHILE I<TOP;
                                                    ; STATEMENT # 12
                          @4:
       0011  210000             LXI      H,TOP
       0014  3A0900             LDA      I
       0017  BE                 CMP      M
       0018  D26300             JNC      @5
 13    3                 IF SORTLIST(I)>SORTLIST(I+1) THEN
                                                    ; STATEMENT # 13
       001B  2A0900             LHLD     I
       001E  2600               MVI      H,0
       0020  010100             LXI      B,SORTLIST
       0023  09                 DAD      B
       0024  E5                 PUSH     H          ; 1
       0025  2A0900             LHLD     I
       0028  2600               MVI      H,0
       002A  03                 INX      B
       002B  09                 DAD      B
       002C  7E                 MOV      A,M
       002D  E1                 POP      H          ; 1
       002E  BE                 CMP      M
       002F  D25C00             JNC      @1
 14    3                   DO;
 15    4                     J=SORTLIST(I);
                                                    ; STATEMENT # 15
       0032  2A0900             LHLD     I
       0035  2600               MVI      H,0
       0037  010100             LXI      B,SORTLIST
```

```
                003A  09              DAD     B
                003B  7E              MOV     A,M
                003C  320A00          STA     J
16    4                     SORTLIST(I)=SORTLIST(I+1);
                                                    ; STATEMENT # 16
                003F  2A0900          LHLD    I
                0042  2600            MVI     H,0
                0044  03              INX     B
                0045  09              DAD     B
                0046  E5              PUSH    H       ; 1
                0047  2A0900          LHLD    I
                004A  2600            MVI     H,0
                004C  0B              DCX     B
                004D  09              DAD     B
                004E  D1              POP     D       ; 1
                004F  1A              LDAX    D
                0050  77              MOV     M,A
17    4                     SORTLIST(I+1)=J;
                                                    ; STATEMENT # 17
                0051  2A0900          LHLD    I
                0054  2600            MVI     H,0
                0056  03              INX     B
                0057  09              DAD     B
                0058  3A0A00          LDA     J
                005B  77              MOV     M,A
18    4               END;
                                                    ; STATEMENT # 18
                                @1:
19    3               I=I+1;
                                                    ; STATEMENT # 19
                005C  210900          LXI     H,I
                005F  34              INR     M
20    3               END;
                                                    ; STATEMENT # 20
                0060  C31100          JMP     @4
                                @5:
21    2               TOP=TOP-1;
                                                    ; STATEMENT # 21
                0063  210000          LXI     H,TOP
                0066  35              DCR     M
22    2            END;
                                                    ; STATEMENT # 22
                0067  C30300          JMP     @2
                                @3:
23    1            DO I=1 TO 8;
                                                    ; STATEMENT # 23
                006A  210900          LXI     H,I
                006D  3601            MVI     M,1H
                                @6:
                006F  3E08            MVI     A,8H
                0071  210900          LXI     H,I
                0074  BE              CMP     M
                0075  DA9A00          JC      @7
24    2            CALL CO (CHAR:=SORTLIST(I));
                                                    ; STATEMENT # 24
                0078  2A0900          LHLD    I
                007B  2600            MVI     H,0
```

```
                007D  010100          LXI     B,SORTLIST
                0080  09              DAD     B
                0081  7E              MOV     A,M
                0082  320B00          STA     CHAR
                0085  4F              MOV     C,A
                0086  CD0000          CALL    CO
  25    2              CALL CO (0AH);
                                                ; STATEMENT # 25
                0089  0E0A            MVI     C,0AH
                008B  CD0000          CALL    CO
  26    2              CALL CO (0DH);
                                                ; STATEMENT # 26
                008E  0E0D            MVI     C,0DH
                0090  CD0000          CALL    CO
  27    2              END;
                                                ; STATEMENT # 27
                0093  210900          LXI     H,I
                0096  34              INR     M
                0097  C26F00          JNZ     @6
                              @7:
  28    1              CALL EXIT;
                                                ; STATEMENT # 28
                009A  CD0000          CALL    EXIT
  29    1          END BUBBLESORT;
                                                ; STATEMENT # 29
                009D  FB              EI
                009E  76              HLT


MODULE INFORMATION:

     CODE AREA SIZE      = 009FH     159D
     VARIABLE AREA SIZE  = 000CH      12D
     MAXIMUM STACK SIZE  = 0002H       2D
     32 LINES READ
     0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION
```

Códigos de operación del conjunto de instrucciones que ejecutan el programa de ordenación por **INSERCION DIRECTA** para un array de ocho elementos.

```
2000 08
2001 00
2002 2C
2003 37
2004 0C
2005 2A
2006 5E
2007 12
2008 06
2009 43
200A 00
0000 31
0001 C2
0002 28
0003 21
0004 0B
0005 20
0006 36
0007 02
0008 3A
0009 00
000A 20
000B 21
000C 0B
000D 20
000E BE
000F DA
0010 68
0011 00
0012 2A
0013 0B
0014 20
0015 26
0016 00
0017 01
0018 01
0019 20
001A 09
001B 7E
001C 32
001D 0D
001E 20
001F 60
0020 69
0021 77
0022 3A
0023 0B
0024 20
0025 3D
0026 32
0027 0C
```

```
0028 20
0029 2A
002A 0C
002B 20
002C 26
002D 00
002E 01
002F 01
0030 20
0031 09
0032 3A
0033 0D
0034 20
0035 BE
0036 D2
0037 54
0038 00
0039 2A
003A 0C
003B 20
003C 26
003D 00
003E 01
003F 01
0040 20
0041 09
0042 E5
0043 2A
0044 0C
0045 20
0046 26
0047 00
0048 03
0049 09
004A D1
004B 1A
004C 77
004D 21
004E 0C
004F 20
0050 35
0051 C3
0052 29
0053 00
0054 2A
0055 0C
0056 20
0057 26
0058 00
0059 01
005A 02
005B 20
005C 09
005D 3A
005E 0D
```

```
005F 20
0060 77
0061 21
0062 0B
0063 20
0064 34
0065 C3
0066 08
0067 00
0068 00
0069 FB
```

Códigos de operación del conjunto de instrucciones que ejecutan el programa de ordenación por **SELECCION DIRECTA** para un array de ocho elementos.

```
2000 08
2001 00
2002 2C
2003 37
2004 0C
2005 2A
2006 5E
2007 12
2008 06
2009 43
200A 00
200B 00
200C 00
200D 00
0000 31
0001 C2
0002 28
0003 21
0004 0E
0005 20
0006 36
0007 01
0008 3A
0009 00
000A 20
000B 3D
000C 21
000D 0E
000E 20
000F BE
0010 DA
0011 80
0012 00
0013 3A
0014 0E
0015 20
0016 32
0017 0B
0018 20
0019 4F
001A 06
001B 00
001C 21
001D 01
001E 20
001F 09
0020 7E
0021 32
0022 0C
0023 20
0024 0C
```

```
0025 79
0026 32
0027 0A
0028 20
0029 3A
002A 00
002B 20
002C 21
002D 0A
002E 20
002F BE
0030 DA
0031 5C
0032 00
0033 2A
0034 0A
0035 20
0036 26
0037 00
0038 01
0039 01
003A 20
003B 09
003C 7E
003D 21
003E 0C
003F 20
0040 BE
0041 D2
0042 55
0043 00
0044 3A
0045 0A
0046 20
0047 32
0048 0B
0049 20
004A 4F
004B 06
004C 00
004D 21
004E 01
004F 20
0050 09
0051 7E
0052 32
0053 0C
0054 20
0055 21
0056 0A
0057 20
0058 34
0059 C3
005A 29
005B 00
```

```
005C 2A
005D 0E
005E 20
005F 26
0060 00
0061 01
0062 01
0063 20
0064 09
0065 E5
0066 2A
0067 0B
0068 20
0069 26
006A 00
006B 09
006C D1
006D 1A
006E 77
006F 2A
0070 0E
0071 20
0072 26
0073 00
0074 09
0075 3A
0076 0C
0077 20
0078 77
0079 21
007A 0E
007B 20
007C 34
007D C3
007E 08
007F 00
0080 00
0081 FB
```

Códigos de operación del conjunto de instrucciones que ejecutan el programa de ordenación por **INTERCAMBIO DIRECTO** para un array de ocho elementos.

```
2000 08
2001 00
2002 2C
2003 37
2004 0C
2005 2A
2006 5E
2007 12
2008 06
2009 43
200A 00
0000 31
0001 C2
0002 26
0003 3E
0004 01
0005 21
0006 00
0007 20
0008 BE
0009 D2
000A 6A
000B 00
000C 21
000D 19
000E 20
000F 36
0010 01
0011 21
0012 00
0013 20
0014 3A
0015 19
0016 20
0017 BE
0018 D2
0019 63
001A 00
001B 2A
001C 19
001D 20
001E 26
001F 00
0020 01
0021 01
0022 20
0023 09
0024 E5
0025 2A
0026 19
0027 20
0028 26
```

```
0029 00
002A 03
002B 09
002C 7E
002D E1
002E BE
002F D2
0030 5C
0031 00
0032 2A
0033 19
0034 20
0035 26
0036 00
0037 01
0038 01
0039 20
003A 09
003B 7E
003C 32
003D 1A
003E 20
003F 2A
0040 19
0041 20
0042 26
0043 00
0044 03
0045 09
0046 E5
0047 2A
0048 19
0049 20
004A 26
004B 00
004C 0B
004D 09
004E D1
004F 1A
0050 77
0051 2A
0052 19
0053 20
0054 26
0055 00
0056 03
0057 09
0058 3A
0059 1A
005A 20
005B 77
005C 21
005D 19
005E 20
005F 34
```

```
0060 C3
0061 11
0062 00
0063 21
0064 00
0065 20
0066 35
0067 C3
0068 03
0069 00
006A FB
```

Simulación del programa de ordenación de un array de ocho elementos, mediante el método de INSERCION DIRECTA.

```
SIMULADOR SILOSS. Versión 2.4 DPTO. ELECTRONICA. MARZO 1.989
============================================================

*****EJECUCION:

Para visualizar el menú pulsar "?"


Lectura de definiciones
Lectura del micro-programa de la memoria MICROMEM
SIM> L MEM_PROG INSER.PRG
SIM> AM MEM_PROG 2000 OE
MEM_PROG              : ADR_MEM
[00002000]   08
[00002001]   00
[00002002]   2C
[00002003]   37
[00002004]   0C
[00002005]   2A
[00002006]   5E
[00002007]   12
[00002008]   06
[00002009]   43
[0000200A]   00
[0000200B]   00
[0000200C]   00
[0000200D]   00
SIM> C REG_INST FB
SIM> CA
condiciones activas
SIM> EP REG_INST
 0       00
 7       31
 15      21
 23      36
 32      3A
 44      21
 52      BE
 58      DA
 68      2A
 84      26
 89      01
 97      09
 101     7E
 107     32
 120     60
 123     69
 126     77
 133     3A
 145     3D
 148     32
 161     2A
 177     26
 182     01
```

190	09
194	3A
206	BE
212	D2
223	2A
239	26
244	01
252	09
256	3A
268	77
275	21
283	34
293	C3
303	3A
315	21
323	BE
329	DA
339	2A
355	26
360	01
368	09
372	7E
378	32
391	60
394	69
397	77
404	3A
416	3D
419	32
432	2A
448	26
453	01
461	09
465	3A
477	BE
483	D2
493	2A
509	26
514	01
522	09
526	E5
535	2A
551	26
556	03
560	09
564	D1
573	1A
579	77
586	21
594	35
604	C3
614	2A
630	26
635	01
643	09

647	3A
659	BE
665	D2
675	2A
691	26
696	01
704	09
708	E5
717	2A
733	26
738	03
742	09
746	D1
755	1A
761	77
768	21
776	35
786	C3
796	2A
812	26
817	01
825	09
829	3A
841	BE
847	D2
858	2A
874	26
879	01
887	09
891	3A
903	77
910	21
918	34
928	C3
938	3A
950	21
958	BE
964	DA
974	2A
990	26
995	01
1003	09
1007	7E
1013	32
1026	60
1029	69
1032	77
1039	3A
1051	3D
1054	32
1067	2A
1083	26
1088	01
1096	09
1100	3A

1112	BE
1118	D2
1128	2A
1144	26
1149	01
1157	09
1161	E5
1170	2A
1186	26
1191	03
1195	09
1199	D1
1208	1A
1214	77
1221	21
1229	35
1239	C3
1249	2A
1265	26
1270	01
1278	09
1282	3A
1294	BE
1300	D2
1310	2A
1326	26
1331	01
1339	09
1343	E5
1352	2A
1368	26
1373	03
1377	09
1381	D1
1390	1A
1396	77
1403	21
1411	35
1421	C3
1431	2A
1447	26
1452	01
1460	09
1464	3A
1476	BE
1482	D2
1493	2A
1509	26
1514	01
1522	09
1526	3A
1538	77
1545	21
1553	34
1563	C3

1573	3A
1585	21
1593	BE
1599	DA
1609	2A
1625	26
1630	01
1638	09
1642	7E
1648	32
1661	60
1664	69
1667	77
1674	3A
1686	3D
1689	32
1702	2A
1718	26
1723	01
1731	09
1735	3A
1747	BE
1753	D2
1764	2A
1780	26
1785	01
1793	09
1797	3A
1809	77
1816	21
1824	34
1834	C3
1844	3A
1856	21
1864	BE
1870	DA
1880	2A
1896	26
1901	01
1909	09
1913	7E
1919	32
1932	60
1935	69
1938	77
1945	3A
1957	3D
1960	32
1973	2A
1989	26
1994	01
2002	09
2006	3A
2018	BE
2024	D2

2034	2A
2050	26
2055	01
2063	09
2067	E5
2076	2A
2092	26
2097	03
2101	09
2105	D1
2114	1A
2120	77
2127	21
2135	35
2145	C3
2155	2A
2171	26
2176	01
2184	09
2188	3A
2200	BE
2206	D2
2216	2A
2232	26
2237	01
2245	09
2249	E5
2258	2A
2274	26
2279	03
2283	09
2287	D1
2296	1A
2302	77
2309	21
2317	35
2327	C3
2337	2A
2353	26
2358	01
2366	09
2370	3A
2382	BE
2388	D2
2398	2A
2414	26
2419	01
2427	09
2431	E5
2440	2A
2456	26
2461	03
2465	09
2469	D1
2478	1A

2484	77
2491	21
2499	35
2509	C3
2519	2A
2535	26
2540	01
2548	09
2552	3A
2564	BE
2570	D2
2580	2A
2596	26
2601	01
2609	09
2613	E5
2622	2A
2638	26
2643	03
2647	09
2651	D1
2660	1A
2666	77
2673	21
2681	35
2691	C3
2701	2A
2717	26
2722	01
2730	09
2734	3A
2746	BE
2752	D2
2763	2A
2779	26
2784	01
2792	09
2796	3A
2808	77
2815	21
2823	34
2833	C3
2843	3A
2855	21
2863	BE
2869	DA
2879	2A
2895	26
2900	01
2908	09
2912	7E
2918	32
2931	60
2934	69
2937	77

2944	3A
2956	3D
2959	32
2972	2A
2988	26
2993	01
3001	09
3005	3A
3017	BE
3023	D2
3033	2A
3049	26
3054	01
3062	09
3066	E5
3075	2A
3091	26
3096	03
3100	09
3104	D1
3113	1A
3119	77
3126	21
3134	35
3144	C3
3154	2A
3170	26
3175	01
3183	09
3187	3A
3199	BE
3205	D2
3215	2A
3231	26
3236	01
3244	09
3248	E5
3257	2A
3273	26
3278	03
3282	09
3286	D1
3295	1A
3301	77
3308	21
3316	35
3326	C3
3336	2A
3352	26
3357	01
3365	09
3369	3A
3381	BE
3387	D2
3397	2A

```
3413    26
3418    01
3426    09
3430    E5
3439    2A
3455    26
3460    03
3464    09
3468    D1
3477    1A
3483    77
3490    21
3498    35
3508    C3
3518    2A
3534    26
3539    01
3547    09
3551    3A
3563    BE
3569    D2
3579    2A
3595    26
3600    01
3608    09
3612    E5
3621    2A
3637    26
3642    03
3646    09
3650    D1
3659    1A
3665    77
3672    21
3680    35
3690    C3
3700    2A
3716    26
3721    01
3729    09
3733    3A
3745    BE
3751    D2
3761    2A
3777    26
3782    01
3790    09
3794    E5
3803    2A
3819    26
3824    03
3828    09
3832    D1
3841    1A
3847    77
```

3854	21
3862	35
3872	C3
3882	2A
3898	26
3903	01
3911	09
3915	3A
3927	BE
3933	D2
3943	2A
3959	26
3964	01
3972	09
3976	E5
3985	2A
4001	26
4006	03
4010	09
4014	D1
4023	1A
4029	77
4036	21
4044	35
4054	C3
4064	2A
4080	26
4085	01
4093	09
4097	3A
4109	BE
4115	D2
4126	2A
4142	26
4147	01
4155	09
4159	3A
4171	77
4178	21
4186	34
4196	C3
4206	3A
4218	21
4226	BE
4232	DA
4242	2A
4258	26
4263	01
4271	09
4275	7E
4281	32
4294	60
4297	69
4300	77
4307	3A

```
4319    3D
4322    32
4335    2A
4351    26
4356    01
4364    09
4368    3A
4380    BE
4386    D2
4396    2A
4412    26
4417    01
4425    09
4429    E5
4438    2A
4454    26
4459    03
4463    09
4467    D1
4476    1A
4482    77
4489    21
4497    35
4507    C3
4517    2A
4533    26
4538    01
4546    09
4550    3A
4562    BE
4568    D2
4579    2A
4595    26
4600    01
4608    09
4612    3A
4624    77
4631    21
4639    34
4649    C3
4659    3A
4671    21
4679    BE
4685    DA
4696    00
4699    FB
```

```
SIM> CN
conditiones inactivas
SIM> AM MEM_PROG 2000 OE
MEM_PROG              : ADR_MEM
[00002000]    08
[00002001]    43
[00002002]    06
[00002003]    0C
[00002004]    12
[00002005]    2A
[00002006]    2C
[00002007]    37
[00002008]    43
[00002009]    5E
[0000200A]    00
[0000200B]    09
[0000200C]    06
[0000200D]    43
SIM> F
FIN: está seguro ? (S/N)S Fin de la simulación
Tiempo de procesado :     486.2 segundos
```

Simulación del programa de ordenación por SELECCION DIRECTA de un array de ocho elementos.

```
SIMULADOR SILOSS. Versión 2.4 DPTO. ELECTRONICA. MARZO 1.989
============================================================

*****EJECUCION:

Para visualizar el menú pulsar "?"


Lectura de definiciones
Lectura del micro-programa de la memoria MICROMEM
SIM> L MEM_PROG SELEC.PRG
SIM> AM MEM_PROG 2000 OF
MEM_PROG              : ADR_MEM
[00002000]   08
[00002001]   00
[00002002]   2C
[00002003]   37
[00002004]   0C
[00002005]   2A
[00002006]   5E
[00002007]   12
[00002008]   06
[00002009]   43
[0000200A]   00
[0000200B]   00
[0000200C]   00
[0000200D]   00
SIM> C REG_INST FB
SIM> CA
condiciones activas
SIM> EP REG_INST
 0       00
 7       31
 15      21
 23      36
 32      3A
 44      3D
 47      21
 55      BE
 61      DA
 71      3A
 83      32
 96      4F
 99      06
 104     21
 112     09
 116     7E
 122     32
 135     0C
 138     79
 141     32
 154     3A
 166     21
 174     BE
```

180	DA
190	2A
206	26
211	01
219	09
223	7E
229	21
237	BE
243	D2
254	21
262	34
272	C3
282	3A
294	21
302	BE
308	DA
318	2A
334	26
339	01
347	09
351	7E
357	21
365	BE
371	D2
381	3A
393	32
406	4F
409	06
414	21
422	09
426	7E
432	32
445	21
453	34
463	C3
473	3A
485	21
493	BE
499	DA
509	2A
525	26
530	01
538	09
542	7E
548	21
556	BE
562	D2
573	21
581	34
591	C3
601	3A
613	21
621	BE
627	DA

637	2A
653	26
658	01
666	09
670	7E
676	21
684	BE
690	D2
701	21
709	34
719	C3
729	3A
741	21
749	BE
755	DA
765	2A
781	26
786	01
794	09
798	7E
804	21
812	BE
818	D2
829	21
837	34
847	C3
857	3A
869	21
877	BE
883	DA
893	2A
909	26
914	01
922	09
926	7E
932	21
940	BE
946	D2
956	3A
968	32
981	4F
984	06
989	21
997	09
1001	7E
1007	32
1020	21
1028	34
1038	C3
1048	3A
1060	21
1068	BE
1074	DA
1084	2A

```
1100   26
1105   01
1113   09
1117   7E
1123   21
1131   BE
1137   D2
1148   21
1156   34
1166   C3
1176   3A
1188   21
1196   BE
1202   DA
1213   2A
1229   26
1234   01
1242   09
1246   E5
1255   2A
1271   26
1276   09
1280   D1
1289   1A
1295   77
1302   2A
1318   26
1323   09
1327   3A
1339   77
1346   21
1354   34
1364   C3
1374   3A
1386   3D
1389   21
1397   BE
1403   DA
1413   3A
1425   32
1438   4F
1441   06
1446   21
1454   09
1458   7E
1464   32
1477   0C
1480   79
1483   32
1496   3A
1508   21
1516   BE
1522   DA
1532   2A
```

1548	26
1553	01
1561	09
1565	7E
1571	21
1579	BE
1585	D2
1595	3A
1607	32
1620	4F
1623	06
1628	21
1636	09
1640	7E
1646	32
1659	21
1667	34
1677	C3
1687	3A
1699	21
1707	BE
1713	DA
1723	2A
1739	26
1744	01
1752	09
1756	7E
1762	21
1770	BE
1776	D2
1787	21
1795	34
1805	C3
1815	3A
1827	21
1835	BE
1841	DA
1851	2A
1867	26
1872	01
1880	09
1884	7E
1890	21
1898	BE
1904	D2
1915	21
1923	34
1933	C3
1943	3A
1955	21
1963	BE
1969	DA
1979	2A
1995	26

```
2000    01
2008    09
2012    7E
2018    21
2026    BE
2032    D2
2043    21
2051    34
2061    C3
2071    3A
2083    21
2091    BE
2097    DA
2107    2A
2123    26
2128    01
2136    09
2140    7E
2146    21
2154    BE
2160    D2
2171    21
2179    34
2189    C3
2199    3A
2211    21
2219    BE
2225    DA
2235    2A
2251    26
2256    01
2264    09
2268    7E
2274    21
2282    BE
2288    D2
2299    21
2307    34
2317    C3
2327    3A
2339    21
2347    BE
2353    DA
2364    2A
2380    26
2385    01
2393    09
2397    E5
2406    2A
2422    26
2427    09
2431    D1
2440    1A
2446    77
```

```
2453    2A
2469    26
2474    09
2478    3A
2490    77
2497    21
2505    34
2515    C3
2525    3A
2537    3D
2540    21
2548    BE
2554    DA
2564    3A
2576    32
2589    4F
2592    06
2597    21
2605    09
2609    7E
2615    32
2628    0C
2631    79
2634    32
2647    3A
2659    21
2667    BE
2673    DA
2683    2A
2699    26
2704    01
2712    09
2716    7E
2722    21
2730    BE
2736    D2
2746    3A
2758    32
2771    4F
2774    06
2779    21
2787    09
2791    7E
2797    32
2810    21
2818    34
2828    C3
2838    3A
2850    21
2858    BE
2864    DA
2874    2A
2890    26
2895    01
```

```
2903    09
2907    7E
2913    21
2921    BE
2927    D2
2938    21
2946    34
2956    C3
2966    3A
2978    21
2986    BE
2992    DA
3002    2A
3018    26
3023    01
3031    09
3035    7E
3041    21
3049    BE
3055    D2
3065    3A
3077    32
3090    4F
3093    06
3098    21
3106    09
3110    7E
3116    32
3129    21
3137    34
3147    C3
3157    3A
3169    21
3177    BE
3183    DA
3193    2A
3209    26
3214    01
3222    09
3226    7E
3232    21
3240    BE
3246    D2
3257    21
3265    34
3275    C3
3285    3A
3297    21
3305    BE
3311    DA
3321    2A
3337    26
3342    01
3350    09
```

```
3354    7E
3360    21
3368    BE
3374    D2
3385    21
3393    34
3403    C3
3413    3A
3425    21
3433    BE
3439    DA
3450    2A
3466    26
3471    01
3479    09
3483    E5
3492    2A
3508    26
3513    09
3517    D1
3526    1A
3532    77
3539    2A
3555    26
3560    09
3564    3A
3576    77
3583    21
3591    34
3601    C3
3611    3A
3623    3D
3626    21
3634    BE
3640    DA
3650    3A
3662    32
3675    4F
3678    06
3683    21
3691    09
3695    7E
3701    32
3714    0C
3717    79
3720    32
3733    3A
3745    21
3753    BE
3759    DA
3769    2A
3785    26
3790    01
3798    09
```

3802	7E
3808	21
3816	BE
3822	D2
3833	21
3841	34
3851	C3
3861	3A
3873	21
3881	BE
3887	DA
3897	2A
3913	26
3918	01
3926	09
3930	7E
3936	21
3944	BE
3950	D2
3961	21
3969	34
3979	C3
3989	3A
4001	21
4009	BE
4015	DA
4025	2A
4041	26
4046	01
4054	09
4058	7E
4064	21
4072	BE
4078	D2
4089	21
4097	34
4107	C3
4117	3A
4129	21
4137	BE
4143	DA
4153	2A
4169	26
4174	01
4182	09
4186	7E
4192	21
4200	BE
4206	D2
4217	21
4225	34
4235	C3
4245	3A
4257	21

```
4265    BE
4271    DA
4282    2A
4298    26
4303    01
4311    09
4315    E5
4324    2A
4340    26
4345    09
4349    D1
4358    1A
4364    77
4371    2A
4387    26
4392    09
4396    3A
4408    77
4415    21
4423    34
4433    C3
4443    3A
4455    3D
4458    21
4466    BE
4472    DA
4482    3A
4494    32
4507    4F
4510    06
4515    21
4523    09
4527    7E
4533    32
4546    0C
4549    79
4552    32
4565    3A
4577    21
4585    BE
4591    DA
4601    2A
4617    26
4622    01
4630    09
4634    7E
4640    21
4648    BE
4654    D2
4664    3A
4676    32
4689    4F
4692    06
4697    21
```

4705 09
4709 7E
4715 32
4728 21
4736 34
4746 C3
4756 3A
4768 21
4776 BE
4782 DA
4792 2A
4808 26
4813 01
4821 09
4825 7E
4831 21
4839 BE
4845 D2
4855 3A
4867 32
4880 4F
4883 06
4888 21
4896 09
4900 7E
4906 32
4919 21
4927 34
4937 C3
4947 3A
4959 21
4967 BE
4973 DA
4983 2A
4999 26
5004 01
5012 09
5016 7E
5022 21
5030 BE
5036 D2
5047 21
5055 34
5065 C3
5075 3A
5087 21
5095 BE
5101 DA
5112 2A
5128 26
5133 01
5141 09
5145 E5
5154 2A

```
5170    26
5175    09
5179    D1
5188    1A
5194    77
5201    2A
5217    26
5222    09
5226    3A
5238    77
5245    21
5253    34
5263    C3
5273    3A
5285    3D
5288    21
5296    BE
5302    DA
5312    3A
5324    32
5337    4F
5340    06
5345    21
5353    09
5357    7E
5363    32
5376    0C
5379    79
5382    32
5395    3A
5407    21
5415    BE
5421    DA
5431    2A
5447    26
5452    01
5460    09
5464    7E
5470    21
5478    BE
5484    D2
5495    21
5503    34
5513    C3
5523    3A
5535    21
5543    BE
5549    DA
5559    2A
5575    26
5580    01
5588    09
5592    7E
5598    21
```

```
5606    BE
5612    D2
5623    21
5631    34
5641    C3
5651    3A
5663    21
5671    BE
5677    DA
5688    2A
5704    26
5709    01
5717    09
5721    E5
5730    2A
5746    26
5751    09
5755    D1
5764    1A
5770    77
5777    2A
5793    26
5798    09
5802    3A
5814    77
5821    21
5829    34
5839    C3
5849    3A
5861    3D
5864    21
5872    BE
5878    DA
5888    3A
5900    32
5913    4F
5916    06
5921    21
5929    09
5933    7E
5939    32
5952    0C
5955    79
5958    32
5971    3A
5983    21
5991    BE
5997    DA
6007    2A
6023    26
6028    01
6036    09
6040    7E
6046    21
```

```
6054    BE
6060    D2
6070    3A
6082    32
6095    4F
6098    06
6103    21
6111    09
6115    7E
6121    32
6134    21
6142    34
6152    C3
6162    3A
6174    21
6182    BE
6188    DA
6199    2A
6215    26
6220    01
6228    09
6232    E5
6241    2A
6257    26
6262    09
6266    D1
6275    1A
6281    77
6288    2A
6304    26
6309    09
6313    3A
6325    77
6332    21
6340    34
6350    C3
6360    3A
6372    3D
6375    21
6383    BE
6389    DA
6400    00
6403    FB
```

```
SIM> CN
condiciones inactivas
SIM> AM MEM_PROG 2000 OF
MEM_PROG              : ADR MEM
[00002000]    08
[00002001]    00
[00002002]    06
[00002003]    0C
[00002004]    12
[00002005]    2A
[00002006]    2C
[00002007]    37
[00002008]    43
[00002009]    5E
[0000200A]    09
[0000200B]    08
[0000200C]    43
[0000200D]    00
SIM> F
FIN: está seguro ? (S/N) Fin de la simulación
Tiempo de procesado :     666.8 segundos
```

Simulación del programa de ordenación de un array de ocho elementos, mediante el método de INTERCAMBIO DIRECTO.

```
SIMULADOR SILOSS. Versión 2.4 DPTO. ELECTRONICA. MARZO 1.989
============================================================

*****EJECUCION:

Para visualizar el menú pulsar "?"


Lectura de definiciones
Lectura del micro-programa de la memoria MICROMEM
SIM> L MEM_PROG PROG10.DAT
SIM> AM MEM_PROG 2000 C
MEM_PROG                : ADR_MEM
[00002000]   08
[00002001]   00
[00002002]   2C
[00002003]   37
[00002004]   0C
[00002005]   2A
[00002006]   5E
[00002007]   12
[00002008]   06
[00002009]   43
[0000200A]   00
[0000200B]   00
SIM> C REG_INST FB
SIM> CA
condiciones activas
SIM> EP REG_INST
    0  00
    7  31
   15  3E
   20  21
   28  BE
   34  D2
   44  21
   52  36
   61  21
   69  3A
   81  BE
   87  D2
   97  2A
  113  26
  118  01
  126  09
  130  E5
  139  2A
  155  26
  160  03
  164  09
  168  7E
  174  E1
  183  BE
  189  D2
  200  21
  208  34
```

```
218  C3
228  21
236  3A
248  BE
254  D2
264  2A
280  26
285  01
293  09
297  E5
306  2A
322  26
327  03
331  09
335  7E
341  E1
350  BE
356  D2
366  2A
382  26
387  01
395  09
399  7E
405  32
418  2A
434  26
439  03
443  09
447  E5
456  2A
472  26
477  0B
481  09
485  D1
494  1A
500  77
507  2A
523  26
528  03
532  09
536  3A
548  77
555  21
563  34
573  C3
583  21
591  3A
603  BE
609  D2
619  2A
635  26
640  01
648  09
652  E5
661  2A
```

677 26
682 03
686 09
690 7E
696 E1
705 BE
711 D2
721 2A
737 26
742 01
750 09
754 7E
760 32
773 2A
789 26
794 03
798 09
802 E5
811 2A
827 26
832 0B
836 09
840 D1
849 1A
855 77
862 2A
878 26
883 03
887 09
891 3A
903 77
910 21
918 34
928 C3
938 21
946 3A
958 BE
964 D2
974 2A
990 26
995 01
1003 09
1007 E5
1016 2A
1032 26
1037 03
1041 09
1045 7E
1051 E1
1060 BE
1066 D2
1077 21
1085 34
1095 C3
1105 21

1113 3A
1125 BE
1131 D2
1141 2A
1157 26
1162 01
1170 09
1174 E5
1183 2A
1199 26
1204 03
1208 09
1212 7E
1218 E1
1227 BE
1233 D2
1243 2A
1259 26
1264 01
1272 09
1276 7E
1282 32
1295 2A
1311 26
1316 03
1320 09
1324 E5
1333 2A
1349 26
1354 0B
1358 09
1362 D1
1371 1A
1377 77
1384 2A
1400 26
1405 03
1409 09
1413 3A
1425 77
1432 21
1440 34
1450 C3
1460 21
1468 3A
1480 BE
1486 D2
1496 2A
1512 26
1517 01
1525 09
1529 E5
1538 2A
1554 26
1559 03

```
1563  09
1567  7E
1573  E1
1582  BE
1588  D2
1598  2A
1614  26
1619  01
1627  09
1631  7E
1637  32
1650  2A
1666  26
1671  03
1675  09
1679  E5
1688  2A
1704  26
1709  0B
1713  09
1717  D1
1726  1A
1732  77
1739  2A
1755  26
1760  03
1764  09
1768  3A
1780  77
1787  21
1795  34
1805  C3
1815  21
1823  3A
1835  BE
1841  D2
1851  2A
1867  26
1872  01
1880  09
1884  E5
1893  2A
1909  26
1914  03
1918  09
1922  7E
1928  E1
1937  BE
1943  D2
1953  2A
1969  26
1974  01
1982  09
1986  7E
1992  32
```

```
2005  2A
2021  26
2026  03
2030  09
2034  E5
2043  2A
2059  26
2064  0B
2068  09
2072  D1
2081  1A
2087  77
2094  2A
2110  26
2115  03
2119  09
2123  3A
2135  77
2142  21
2150  34
2160  C3
2170  21
2178  3A
2190  BE
2196  D2
2207  21
2215  35
2225  C3
2235  3E
2240  21
2248  BE
2254  D2
2264  21
2272  36
2281  21
2289  3A
2301  BE
2307  D2
2317  2A
2333  26
2338  01
2346  09
2350  E5
2359  2A
2375  26
2380  03
2384  09
2388  7E
2394  E1
2403  BE
2409  D2
2419  2A
2435  26
2440  01
2448  09
```

```
2452  7E
2458  32
2471  2A
2487  26
2492  03
2496  09
2500  E5
2509  2A
2525  26
2530  0B
2534  09
2538  D1
2547  1A
2553  77
2560  2A
2576  26
2581  03
2585  09
2589  3A
2601  77
2608  21
2616  34
2626  C3
2636  21
2644  3A
2656  BE
2662  D2
2672  2A
2688  26
2693  01
2701  09
2705  E5
2714  2A
2730  26
2735  03
2739  09
2743  7E
2749  E1
2758  BE
2764  D2
2774  2A
2790  26
2795  01
2803  09
2807  7E
2813  32
2826  2A
2842  26
2847  03
2851  09
2855  E5
2864  2A
2880  26
2885  0B
2889  09
```

2893 D1
2902 1A
2908 77
2915 2A
2931 26
2936 03
2940 09
2944 3A
2956 77
2963 21
2971 34
2981 C3
2991 21
2999 3A
3011 BE
3017 D2
3027 2A
3043 26
3048 01
3056 09
3060 E5
3069 2A
3085 26
3090 03
3094 09
3098 7E
3104 E1
3113 BE
3119 D2
3130 21
3138 34
3148 C3
3158 21
3166 3A
3178 BE
3184 D2
3194 2A
3210 26
3215 01
3223 09
3227 E5
3236 2A
3252 26
3257 03
3261 09
3265 7E
3271 E1
3280 BE
3286 D2
3296 2A
3312 26
3317 01
3325 09
3329 7E
3335 32

```
3348  2A
3364  26
3369  03
3373  09
3377  E5
3386  2A
3402  26
3407  0B
3411  09
3415  D1
3424  1A
3430  77
3437  2A
3453  26
3458  03
3462  09
3466  3A
3478  77
3485  21
3493  34
3503  C3
3513  21
3521  3A
3533  BE
3539  D2
3549  2A
3565  26
3570  01
3578  09
3582  E5
3591  2A
3607  26
3612  03
3616  09
3620  7E
3626  E1
3635  BE
3641  D2
3651  2A
3667  26
3672  01
3680  09
3684  7E
3690  32
3703  2A
3719  26
3724  03
3728  09
3732  E5
3741  2A
3757  26
3762  0B
3766  09
3770  D1
3779  1A
```

```
3785  77
3792  2A
3808  26
3813  03
3817  09
3821  3A
3833  77
3840  21
3848  34
3858  C3
3868  21
3876  3A
3888  BE
3894  D2
3904  2A
3920  26
3925  01
3933  09
3937  E5
3946  2A
3962  26
3967  03
3971  09
3975  7E
3981  E1
3990  BE
3996  D2
4007  21
4015  34
4025  C3
4035  21
4043  3A
4055  BE
4061  D2
4072  21
4080  35
4090  C3
4100  3E
4105  21
4113  BE
4119  D2
4129  21
4137  36
4146  21
4154  3A
4166  BE
4172  D2
4182  2A
4198  26
4203  01
4211  09
4215  E5
4224  2A
4240  26
4245  03
```

```
4249  09
4253  7E
4259  E1
4268  BE
4274  D2
4285  21
4293  34
4303  C3
4313  21
4321  3A
4333  BE
4339  D2
4349  2A
4365  26
4370  01
4378  09
4382  E5
4391  2A
4407  26
4412  03
4416  09
4420  7E
4426  E1
4435  BE
4441  D2
4452  21
4460  34
4470  C3
4480  21
4488  3A
4500  BE
4506  D2
4516  2A
4532  26
4537  01
4545  09
4549  E5
4558  2A
4574  26
4579  03
4583  09
4587  7E
4593  E1
4602  BE
4608  D2
4618  2A
4634  26
4639  01
4647  09
4651  7E
4657  32
4670  2A
4686  26
4691  03
4695  09
```

```
4699  E5
4708  2A
4724  26
4729  0B
4733  09
4737  D1
4746  1A
4752  77
4759  2A
4775  26
4780  03
4784  09
4788  3A
4800  77
4807  21
4815  34
4825  C3
4835  21
4843  3A
4855  BE
4861  D2
4871  2A
4887  26
4892  01
4900  09
4904  E5
4913  2A
4929  26
4934  03
4938  09
4942  7E
4948  E1
4957  BE
4963  D2
4973  2A
4989  26
4994  01
5002  09
5006  7E
5012  32
5025  2A
5041  26
5046  03
5050  09
5054  E5
5063  2A
5079  26
5084  0B
5088  09
5092  D1
5101  1A
5107  77
5114  2A
5130  26
5135  03
```

```
5139  09
5143  3A
5155  77
5162  21
5170  34
5180  C3
5190  21
5198  3A
5210  BE
5216  D2
5226  2A
5242  26
5247  01
5255  09
5259  E5
5268  2A
5284  26
5289  03
5293  09
5297  7E
5303  E1
5312  BE
5318  D2
5329  21
5337  34
5347  C3
5357  21
5365  3A
5377  BE
5383  D2
5394  21
5402  35
5412  C3
5422  3E
5427  21
5435  BE
5441  D2
5451  21
5459  36
5468  21
5476  3A
5488  BE
5494  D2
5504  2A
5520  26
5525  01
5533  09
5537  E5
5546  2A
5562  26
5567  03
5571  09
5575  7E
5581  E1
5590  BE
```

```
5596  D2
5607  21
5615  34
5625  C3
5635  21
5643  3A
5655  BE
5661  D2
5671  2A
5687  26
5692  01
5700  09
5704  E5
5713  2A
5729  26
5734  03
5738  09
5742  7E
5748  E1
5757  BE
5763  D2
5773  2A
5789  26
5794  01
5802  09
5806  7E
5812  32
5825  2A
5841  26
5846  03
5850  09
5854  E5
5863  2A
5879  26
5884  0B
5888  09
5892  D1
5901  1A
5907  77
5914  2A
5930  26
5935  03
5939  09
5943  3A
5955  77
5962  21
5970  34
5980  C3
5990  21
5998  3A
6010  BE
6016  D2
6026  2A
6042  26
6047  01
```

```
6055  09
6059  E5
6068  2A
6084  26
6089  03
6093  09
6097  7E
6103  E1
6112  BE
6118  D2
6128  2A
6144  26
6149  01
6157  09
6161  7E
6167  32
6180  2A
6196  26
6201  03
6205  09
6209  E5
6218  2A
6234  26
6239  0B
6243  09
6247  D1
6256  1A
6262  77
6269  2A
6285  26
6290  03
6294  09
6298  3A
6310  77
6317  21
6325  34
6335  C3
6345  21
6353  3A
6365  BE
6371  D2
6381  2A
6397  26
6402  01
6410  09
6414  E5
6423  2A
6439  26
6444  03
6448  09
6452  7E
6458  E1
6467  BE
6473  D2
6484  21
```

```
6492  34
6502  C3
6512  21
6520  3A
6532  BE
6538  D2
6549  21
6557  35
6567  C3
6577  3E
6582  21
6590  BE
6596  D2
6606  21
6614  36
6623  21
6631  3A
6643  BE
6649  D2
6659  2A
6675  26
6680  01
6688  09
6692  E5
6701  2A
6717  26
6722  03
6726  09
6730  7E
6736  E1
6745  BE
6751  D2
6762  21
6770  34
6780  C3
6790  21
6798  3A
6810  BE
6816  D2
6826  2A
6842  26
6847  01
6855  09
6859  E5
6868  2A
6884  26
6889  03
6893  09
6897  7E
6903  E1
6912  BE
6918  D2
6928  2A
6944  26
6949  01
```

```
6957  09
6961  7E
6967  32
6980  2A
6996  26
7001  03
7005  09
7009  E5
7018  2A
7034  26
7039  0B
7043  09
7047  D1
7056  1A
7062  77
7069  2A
7085  26
7090  03
7094  09
7098  3A
7110  77
7117  21
7125  34
7135  C3
7145  21
7153  3A
7165  BE
7171  D2
7181  2A
7197  26
7202  01
7210  09
7214  E5
7223  2A
7239  26
7244  03
7248  09
7252  7E
7258  E1
7267  BE
7273  D2
7284  21
7292  34
7302  C3
7312  21
7320  3A
7332  BE
7338  D2
7349  21
7357  35
7367  C3
7377  3E
7382  21
7390  BE
7396  D2
```

```
7406  21
7414  36
7423  21
7431  3A
7443  BE
7449  D2
7459  2A
7475  26
7480  01
7488  09
7492  E5
7501  2A
7517  26
7522  03
7526  09
7530  7E
7536  E1
7545  BE
7551  D2
7561  2A
7577  26
7582  01
7590  09
7594  7E
7600  32
7613  2A
7629  26
7634  03
7638  09
7642  E5
7651  2A
7667  26
7672  0B
7676  09
7680  D1
7689  1A
7695  77
7702  2A
7718  26
7723  03
7727  09
7731  3A
7743  77
7750  21
7758  34
7768  C3
7778  21
7786  3A
7798  BE
7804  D2
7814  2A
7830  26
7835  01
7843  09
7847  E5
```

7856 2A
7872 26
7877 03
7881 09
7885 7E
7891 E1
7900 BE
7906 D2
7917 21
7925 34
7935 C3
7945 21
7953 3A
7965 BE
7971 D2
7982 21
7990 35
8000 C3
8010 3E
8015 21
8023 BE
8029 D2
8039 21
8047 36
8056 21
8064 3A
8076 BE
8082 D2
8092 2A
8108 26
8113 01
8121 09
8125 E5
8134 2A
8150 26
8155 03
8159 09
8163 7E
8169 E1
8178 BE
8184 D2
8195 21
8203 34
8213 C3
8223 21
8231 3A
8243 BE
8249 D2
8260 21
8268 35
8278 C3
8288 3E
8293 21
8301 BE
8307 D2

```
 8318  FB
SIM> CN
condiciones inactivas
SIM> AM MEM_PROG 2000 C
MEM_PROG              : ADR MEM
[00002000]   01
[00002001]   00
[00002002]   06
[00002003]   0C
[00002004]   12
[00002005]   2A
[00002006]   2C
[00002007]   37
[00002008]   43
[00002009]   5E
[0000200A]   00
[0000200B]   00
SIM> F
FIN: está seguro ? (S/N) Fin de la simulación
Tiempo de procesado :    843.1 segundos
```

VI. EXPERIMENTACION

Comparación entre el número de estados de los ciclos de máquina correspondientes a cada instrucción.

Grupo de transferencia de datos.-

Mnemonic	8085	CPA
MOV r1,r2	4T	3T
MOV r,M	7T	6T
MOV M,r	7T	7T
MVI r,data	7T	5T
MVI M,data	10T	9T
LXI rp,data 16	10T	8T
LDA addr	13T	12T
STA addr	13T	13T
LHLD addr	16T	16T
SHLD addr	16T	14T
LDAX rp	7T	6T
STAX rp	7T	6T
XCHG	4T	7T

Grupo aritmético.-

Mnemonic	8085	CPA
ADD r	4T	3T
ADD M	7T	6T
ADI data	7T	5T
ADC r	4T	3T
ADC M	7T	6T
ACI data	7T	5T
SUB r	4T	3T
SUB M	7T	6T
SUI data	7T	5T
SBB r	4T	3T
SBB M	7T	6T
SBI data	7T	5T
INR r	4T	3T
INR M	10T	10T
DCR r	4T	3T
DCR M	10T	10T
INX rp	6T	4T
DCX rp	6T	4T
DAD rp	10T	4T
DAA	4T	6/7T

Grupo lógico.-

Mnemonic	8085	CPA
ANA r	4T	3T
ANA M	7T	6T
ANI data	7T	5T
XRA r	4T	3T
XRA M	7T	6T
XRI data	7T	5T
ORA r	4T	3T
ORA M	7T	6T
ORI data	7T	5T
CMP r	4T	3T
CMP M	7T	6T
CPI data	7T	5T
RLC	4T	3T
RRC	4T	3T
RAL	4T	3T
RAR	4T	3T
CMA	4T	3T
CMC	4T	3T
STC	4T	5T

Grupo de bifurcación, de stack, E/S y control de máquina.-

Mnemonic	8085	CPA
JMP adr	10T	10T
J cond adr	7/10T	10/11T
CALL adr	18T	15T
C cond adr	9/18T	10/16T
RET	10T	10T
R cond	6/12T	4/11T
RST n	12T	12T
PCHL	6T	4T
PUSH rp	12T	9T
PUSH PSW	12T	9T
POP rp	10T	9T
POP PSW	10T	9T
XTHL	16T	14T
SPHL	6T	3T
IN port	10T	8T
OUT port	10T	8T
EI	4T	3T
DI	4T	3T
HLT	5T	5T
NOP	4T	3T
RIM	4T	3T
SIM	4T	3T

Tabla comparativa entre los *estados* y tiempos de ejecución del conjunto de instrucciones en cada procesador.

μP	Estados	T	Ejecución
8085	de 4 a 18	320ns	de 1.28 a 5.76 μs
8085A-2	de 4 a 18	200ns	de 0.8 a 3.6 μs
8085AH-1	de 4 a 18	167ns	de 0.67 a 3 μs
8085AH-1[1]	de 4 a 18	67ns	de 0.27 a 1.2 μs
CPA (Si)	de 3 a 16	150ns	de 0.45 a 2.4 μs
CPA (GaAs)	de 3 a 16	25ns	de 0.075 a 0.4 μs

[1] Versión del AH a 15MHz.

Análisis dinámico para la ejecución del programa de ordenación de un *array* de 8 elementos mediante el método de *inserción directa*.

N° Inst.	Mnemonic	N° Estados para 8085
1	LXI SP, dble	10
29	LXI H, dble	290
1	MVI M, data	10
44	LDA adr	572
29	CMP M	203
1	JC adr (cond T)	10
7	JC adr (cond F)	49
66	LHLD adr	1056
66	MVI H, data	462
51	LXI B, dble	510
66	DAD B	660
7	MOV A,M	49
14	STA adr	182
7	MOV H,B	28
7	MOV L,C	28
29	MOV M,A	203
7	DCR A	28
7	JNC adr (cond T)	70
15	JNC adr (cond F)	105
15	PUSH H	180
15	INX B	240
15	POP D	150
15	LDAX D	105
15	DCR M	150
22	JMP	220
7	INR M	70
1	NOP	4

El número total de estados que se ejecutan para la resolución del programa de ordenación por *inserción directa* es de:

en i8085	en CPA
5644 Estados	**4699** Estados

Comparación del tiempo de ejecución del programa de ordenación por *inserción directa*.

µP	Tiempo de ejecución
8085A	1.81 mseg
8085A-2	1.13 mseg
8085AH-1(6MHz)	0.94 mseg
8085AH-1(15MHz)	0.38 mseg
CPA (Si)	0.71 mseg
CPA (GaAs)	0.12 mseg

Análisis dinámico para la ejecución del programa de ordenación por *selección directa* para un *array* de 8 elementos.

N° Inst.	Mnemonic	N° Estados para 8085
1	LXI SP, dble	10
122	LXI H, dble	1220
1	MVI M, data	10
65	LDA adr	845
7	DCR A	28
71	CMP M	497
8	JC adr (cond T)	80
35	JC adr (cond F)	245
37	STA adr	481
15	MOV C,A	60
15	MVI B,data	105
64	DAD B	640
43	MOV A,M	301
7	INR C	28
7	MOV A,C	28
49	LHLD adr	784
49	MVI H, data	343
35	LXI B, dble	350
20	JNC adr (cond T)	200
8	JNC adr (cond F)	56
7	PUSH H	84
7	POP D	70
7	LDAX D	49
14	MOV M,A	98
35	INR M	350
35	JMP	350
1	NOP	4

El número total de estados que se ejecutan para la resolución del programa de ordenación por *selección directa* es de:

en i8085	en CPA
7316 Estados	6403 Estados

Comparación del tiempo de ejecución del programa de ordenación por *selección directa*.

μP	Tiempo de ejecución
8085A	2.34 mseg
8085A-2	1.46 mseg
8085AH-1(6MHz)	1.22 mseg
8085AH-1(15MHz)	0.49 mseg
CPA (Si)	0.96 mseg
CPA (GaAs)	0.16 mseg

Análisis dinámico para la ejecución del programa de ordenación de un *array* de 8 elementos mediante el método de la burbuja (*Bubblesort*).

N° Inst.	Mnemonic	N° Estados para 8085
1	LXI SP, dble	10
8	MVI A, byte	56
85	LXI H, dble	850
71	CMP M	497
26	JNC adr (cond T)	260
45	JNC adr (cond F)	315
7	MVI M, byte	70
50	LDA adr	650
116	LHLD adr	1856
116	MVI H, byte	812
43	LXI B, dble	430
116	DAD B	1160
43	PUSH H	516
58	INX B	348
43	MOV A,M	301
28	POP H	280
15	STA adr	195
15	DCX B	90
15	POP D	150
30	MOV M,A	210
28	INR M	280
35	JMP adr	350
7	DCR M	70
15	LDAX D	105

El número total de estados que se ejecutan para la resolución del programa *Bubblesort* es de:

en i8085	en CPA
9861 Estados	8318 Estados

Comparación del tiempo de ejecución del programa *Bubblesort* para 8 elementos en cada procesador.

µP	Tiempo de ejecución
8085A	3.16 mseg
8085A-2	1.97 mseg
8085AH-1(6MHz)	1.65 mseg
8085AH-1(15MHz)	0.66 mseg
CPA (Si)	1.25 mseg
CPA (GaAs)	0.21 mseg

CONCLUSIONES.

Finalmente se puede concluir diciendo que, para un análisis estático, en el 83% de las instrucciones se mejora el número de estados o ciclos de reloj que se requieren para la ejecución de las mismas, con respecto al que emplea el i8085. En el 12% se emplea el mismo número, y sólo en el 5% se empeoran.

Con los datos obtenidos del análisis dinámico se establece que la versión en silicio de CPA, mejora en un 25% la velocidad de ejecución del 8085AH-1 (6MHz, tecnología HMOS de 1985). Si bien, hay que resaltar que los tiempos de propagación de los dispositivos, que se han manejado en el estudio de la ruta de datos crítica, para determinar el ciclo de reloj de CPA-ECL (150ns), son relativos a 1983 (Bipolar Microprocessor Logic and Interface, Am2900 Family Data Book). AMD mejoró, con el proceso de fabricación IMOX-S2, la velocidad de sus dispositivos en un 30%; por lo que se puede afirmar que el reloj del sistema de CPA-ECL trabaja perfectamente a 10MHz, optimizandose aún los datos mencionados.

Sustituyendo los elementos de la arquitectura por sus equivalentes en AsGa, CPA-AsGa mejora la velocidad de ejecución del 8085AH-1 (versión más reciente capaz de trabajar con un reloj de sistema de

15MHz) en un 70% , es decir que resulta ser 3 veces más rápido. En comparación con el 8085AH-1 (6MHz) sería 8 veces más rápido.

Se demuestra un incremento de velocidad de CPA-AsGa (40MHz) con respecto a la versión en silicio del orden de seis veces. Esto viene a corroborar una vez más las importantes ventajas que ofrece la tecnología del AsGa, aunque actualmente el campo de aplicaciones está restringido por su alto costo, diez veces superior en muchos casos el costo del correspondiente sistema en silicio de última tecnología.

Para finalizar, los bloques bit-slice stándar utilizados limitan la velocidad aproximadamente en un 60% con respecto a la lograda sin estos esquemas. Con el afán de optimizar el microprograma se podían haber añadido más elementos hardware, pero esto sería costoso. También se puede establecer una estructura modular del microprograma, haciendo llamadas a aquellas rutinas repetitivas (ciclo de máquina de lectura a memoria, etc.) pero incrementaría el número de estados de muchas instrucciones, desmejorando las ventajas obtenidas.

APENDICE A

3.- SILOSS. SIMULADOR LOGICO DE SISTEMAS SECUENCIALES.

3.1.- PRESENTACION DEL SIMULADOR.

Existen ya muchos simuladores; he aquí uno nuevo. La necesidad de este programa está de hecho justificada por la complejidad de los sistemas secuenciales programables. Su realización está contemplada en el desarrollo de herramientas destinadas a facilitar la el diseño de sistemas lógicos. La existencia de un microensamblador universal simplifica la implantación. La etapa siguiente es el testeo de los microprogramas mediante el uso de un simulador.

Entre los simuladores existentes podemos señalar el SPICE, que simula en el tiempo las características eléctricas de un circuito, LOGMOS orientado a las simulaciones lógicas para células MOS, etc,... . Los modelos utilizados son demasiado limitados para simular el aspecto funcional de una máquina secuencial, formada a base de memorias y registros entre otros elementos; ello conduciría a una descripción larga y compleja. SILOSS se conforma con una enumeración de los elementos que forman la máquina (registros, buses, memorias, pilas, entradas y salidas), sin necesidad de especificar

todas y cada una de sus conexiones.

El funcionamiento está descrito por las instrucciones de transferencia de la forma:

SI R(i)=C ENTONCES R(j):=f(R(k),R(1))

donde R(n) es un registro cualquiera o parte de un registro y C es una constante.

Este lenguaje de descripción es no procedural (no secuencial), esto es, que las instrucciones se ejecutan en un orden que no es forzosamente el de su declaración.

La simulación se desarrolla en dos fases:

- Un primer programa compila la descripción de la máquina y genera un fichero listado y un fichero de código.
- Este fichero y un fichero binario, generado por el microensamblador, son leidos por un segundo programa que permite realizar la ejecución del microprograma, en la máquina descrita, de manera interactiva.

Estos programas están escritos en PASCAL para VAX.

3.2.- PROGRAMA DE ANALISIS DE LA DESCRIPCION.

Se trata de hecho de un compilador que verifica la sintaxis y la semántica de la descripción y que, si la descripción es correcta, genera un fichero que contiene la descripción compactada, utilizable por el programa de

ejecución. La compilación se efectúa en un sólo paso. Esto implica que todos los identificadores que aparecen en la descripción deberán estar declarados antes de su primera utilización.

A continuación se describen las diferentes reglas lexicales, tanto las sintácticas como las semánticas. Los símbolos terminales de las reglas sintácticas aparecen en un círculo, mientras que los símbolos no terminales, definidos en otra parte, aparecen en un rectángulo.

3.2.1.- REGLAS LEXICALES.

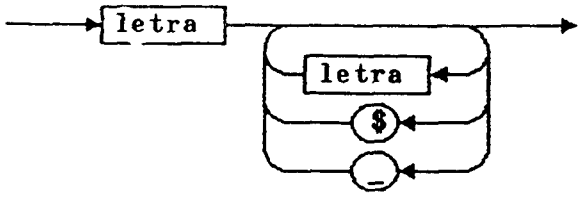
Como en algunos lenguajes de alto nivel, el formato de entrada es libre. El uso de palabras claves permite la realización de las ordenes deseadas. Las reglas lexicales son las siguientes:

- Las letras mayúsculas y minúsculas son equivalentes.
- Son admisibles los caracteres siguientes:
    * las letras (mayúsculas y minúsculas);
    * los digitos decimales;
    * los siguientes símbolos: "$", "*", "+", ",", "-", ".", "/", ":", "<", ">", "=", "(", ")", "_".

  En los comentarios se aceptan todos los tipos de caracteres.

- Los caracteres de paginación se reemplazan (lo mismo que los comentarios) por dos espacios: uno para el form-feed, line-feed o back-space; otro más para el tabulador.
- Los comentarios se escriben entre paréntesis y pueden estar imbricados de manera ilimitada.
- Un identificador debe comenzar por una letra y puede contener los caracteres "$" y "_" (Figura 3.1). La longitud de los identificadores puede ser cualquiera pero solamente son significativos los primeros 20 caracteres.
- Un número debe comenzar por una cifra. Pueden contener el caracter "_" y las letras de la "A" a la "F" (Figura 3.2). La base se indica por una letra al final del número: B = Binario, O = Octal, H = Hexadecimal. Por defecto los números están expresados en decimal.
- Un número cualquiera (diferente de 0) de blancos, tabuladores y finales de línea <CR> o comentarios se consideran como separador.
- Entre dos símbolos alfanuméricos (palabra clave, identificador o número) es obligatorio el uso de separadores, pero no es necesario entre un símbolo alfanumérico y un símbolo no alfanumérico (caracteres de puntuación o delimitación).

Identificador



Letra

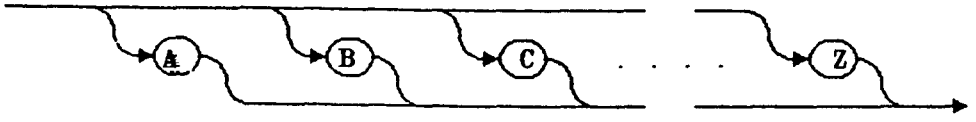
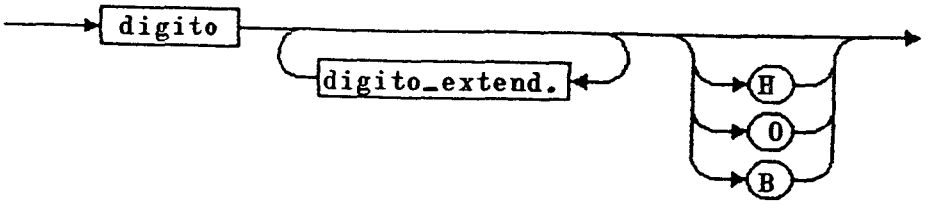
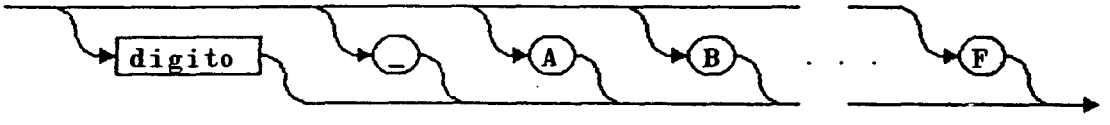


Figura 3.1

Número



Digito_extendido



Digito

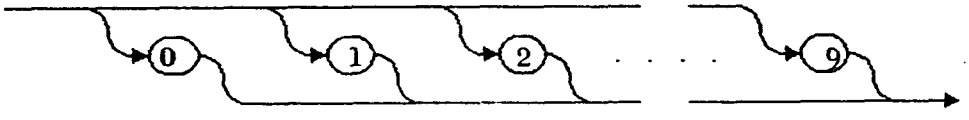


Figura 3.2

Tabla de identificadores reservados.

AND	MACHINE
BEGIN	MEMORY
BUS	MICRO_OP
FIELD	NEG
CONSTANT	NOT
ASR	OR
ASL	STACK
DEC	POP
LSR	PUSH
LSL	RR
INPUT	REGISTER
IS	RL
END	OUTPUT
END_MICRO	IF
END_IF	XOR
INC	ZERO
LATCH	

Tabla 3.1

3.2.2.- ENCABEZAMIENTO DE LA DESCRIPCION.

La descripción se compone de tres partes (figura 3.3):

* Un encabezamiento,
* Un bloque de declaraciones, y
* Un bloque principal.

El encabezamiento permite dar un nombre a la máquina descrita y de transmitir los parámetros al compilador (Figura 3.4). Los parámetros tienen los siguientes significados:

* TABLE da la orden de imprimir la tabla de los identificadores declarados en la descripción al final del listado (por defecto la tabla no

se imprime).

* NOT_CODE permite suprimir la generación de la descripción compactada (fichero .REG), con el fin de acelerar la compilación en la fase de depurado.

* PAGE = número, indica al compilador el número de líneas por página (por defecto, se toman 60 líneas por página).

Las palabras claves **TABLE**, **NOT_CODE** y **PAGE** no son reservadas y su empleo como identificadores es libre, contrariamente a las palabras claves de la Tabla 3.1 que no deben usarse como identificadores.

Simulador

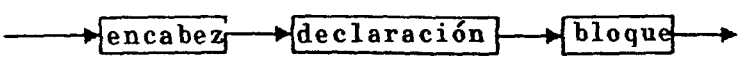
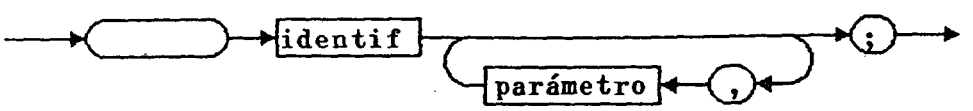


Figura 3.3

Encabezamiento



Parámetros

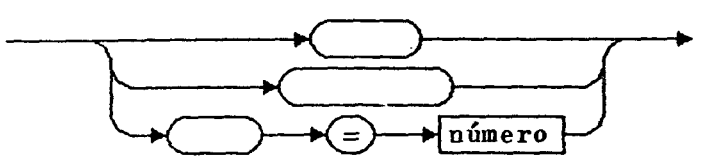


Figura 3.4

3.2.3.-PARTE DE DECLARACION.

Esta parte contiene la enumeración de los elementos que constituyen la máquina y los identificadores de constantes o de referencias de bits. El orden de declaración es fijo: **CONSTANT**, **REGISTER**, **LATCH**, **BUS**, **MEMORY**, **STACK**, **INPUT**, **OUTPUT**, y **FIELD** (Figura 3.5). Es obligatorio la declaración de al menos una memoria a fin de poder cargar el microprograma; las demás declaraciones son facultativas.

La estructura de las declaraciones es siempre la misma: en primer lugar la palabra clave indicando el tipo de la declaración; después la declaración del elemento, separados por puntos y comas ";"

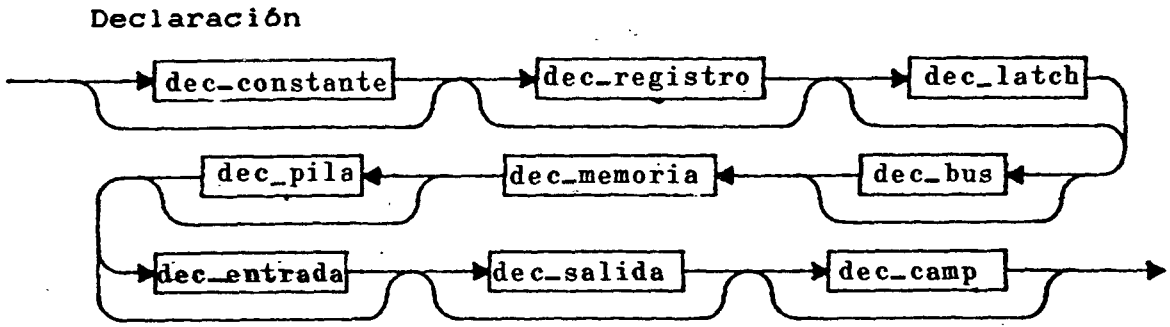


Figura 3.5

3.2.3.1.- Declaración de constantes.

La sintaxis de la declaración de las constantes se describe en la figura 3.6: nombre de la constante, el signo "=" seguida de una expresión numérica dando el

valor de la constante. Permiten las operaciones de multiplicación, división, suma y resta, y se realizan de la misma manera que el tipo INTEGER de PASCAL. Las operaciones de multiplicación y de división se efectúan antes que las de suma y resta. Los paréntesis no están permitidos, ya que se utilizan como delimitadores de los comentarios.

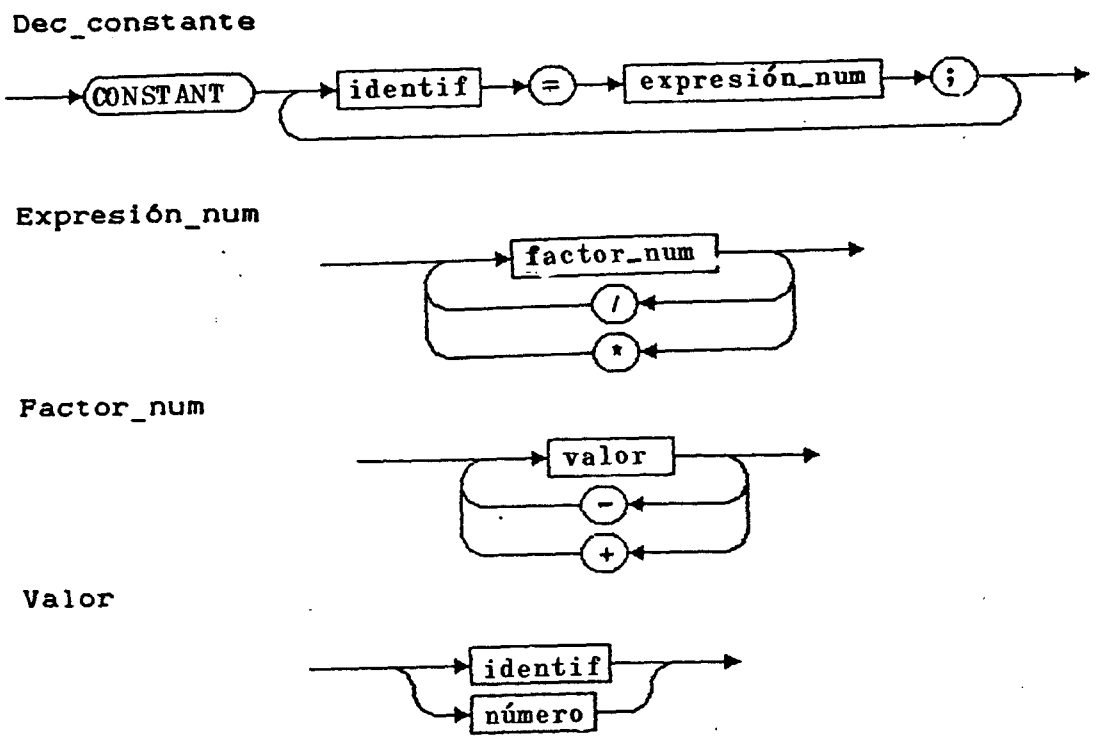


Figura 3.6

Los valores que aparecen en las expresiones pueden ser numéricos o identificadores de constantes. La definición recursiva de una constante es sintácticamente correcta pero no útil en la práctica ya que las constantes están inicializadas a cero antes del cálculo de la expresión. Por ejemplo, estas dos declaraciones son

correctas y semánticamente equivalentes:

```
RECORD=RECORD+1;
RECORD=0+1;
```

3.2.3.2.-Declaración de registros.

La sintaxis se da en la figura 3.7. Un registro se define por su longitud y por su valor inicial. La longitud de un registro es el número de bits que contiene, número comprendido entre 1 y 32 y colocados entre los símbolos "<" y ">". Por defecto, el valor inicial del registro es cero.

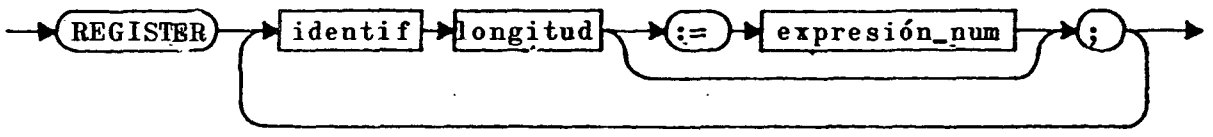
Existen dos tipos de registros:

* El REGISTRO, que es un elemento síncrono, ya que cambia de valor solamente sobre los flancos de la señal de reloj. Por tanto, el simulador atribuye dos valores a un registro: el valor presente o actual, que podemos leer y el valor futuro, que es el próximo valor que tomará el registro. Al final de un período de reloj el valor futuro pasa a ser valor presente.

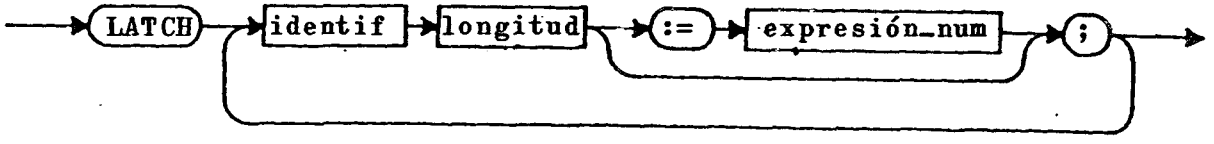
* El LATCH, que es un elemento asíncrono; el simulador no le atribuye más que un solo

valor que puede ser cargado y leido en un mismo ciclo de reloj.

Dec_registro



Dec_latch



Longitud

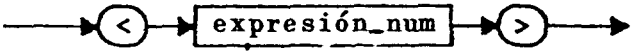


Figura 3.7

3.2.3.3.- Declaración del bus.

La sintaxis se da en la figura 3.8. Un bus va a estar identificado por su longitud de palabra. Este no es un elemento de memoria: antes de ser leido ,en el mismo período de reloj, debe de haber recibido un valor. Dicho valor no se conserva de un período a otro.

La utilidad de un bus, que de hecho no es más que un elemento de conexión, se pone de manifiesto cuando pretendemos llevar una información a varios elementos distintos; es más simple el calcular de una vez un valor, almacenarlo en el bus y transmitirlo a todos los

elementos de destino que tener que calcularlos para cada uno de los registros destinos.

Dec_bus

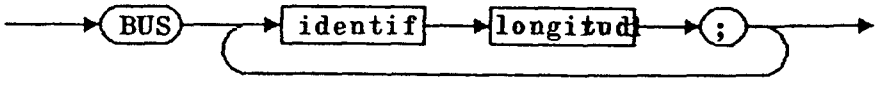


Figura 3.8

3.2.3.4.- Declaración de memorias

La sintaxis se da en la figura 3.9. Una memoria se caracteriza por su longitud, por su ancho, su direccionamiento y su contenido inicial. La longitud es el número de palabras de la memoria, número comprendido entre 1 y 65.535 (64K) y colocado entre corchetes ("[","]"). El ancho es el número de bits de una palabra. El direccionamiento es el nombre del elemento, registro bus o memoria, ya declarado, cuyo contenido representa la dirección de la memoria. Por tanto será el elemento direccionador de dicha memoria. El número de bits de este elemento no debe corresponder obligatoriamente a la dimensión de la memoria ya que se efectúa un control de la dirección antes de acceder a la memoria.

La inicialización permite dar un valor a ciertas

palabas de la memoria. Ello puede realizarse de tres modos diferentes:

- Un valor solo: este valor se coloca en la dirección actual. La dirección actual se inicializa a cero, incrementándose después de cada utilización.

- Dos valores separados por ":": el primer valor representa la dirección donde vamos a colocar el segundo valor dado. Esta dirección pasa a ser la nueva dirección en curso.

- Tres valores separados por ".." y ":": todas las direcciones entre la primera y la segunda se inicializan al valor de la tercera. La dirección final se convierte en la dirección en curso.

Para la inicialización de una memoria podemos combinar estas tres posibilidades. La inicialización es facultativa, pero las palabras no inicializadas pueden tomar un valor cualquiera al comienzo de la simulación.

La memoria es un elemento asíncrono: para una dirección dada, el valor leído es el último valor escrito. La lectura o la escritura se realiza en la dirección contenida en el elemento declarado como de direccionamiento de la memoria. Si este elemento no toma un valor definido (bus no afectado) la operación no

puede realizarse.

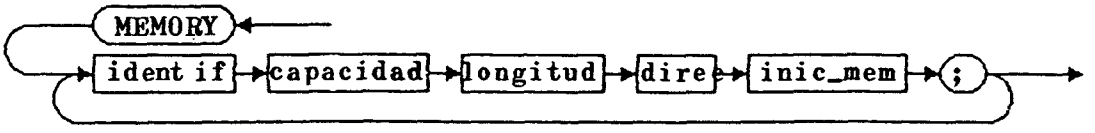
La primera memoria declarada tiene una misión muy particular: es ella la que va a recibir el microprograma; de aquí que la declaración de al menos una memoria sea obligatoria. Su longitud de palabra puede llegar hasta 256 bits, ya que una microinstrucción contiene generalmente más de 32 bits. Si la longitud es superior a 32 bits, tenemos las siguientes restricciones:

- La inicialización está prohibida. Esta memoria se inicializa al inicio de la ejecución del simulador o mediante la carga de dicha memoria a partir de un fichero binario fuente. Parece lógico ya que cuando las microinstrucciones son muy grandes son más difíciles de tratarlas y es por ello que se realiza con un microensamblador.

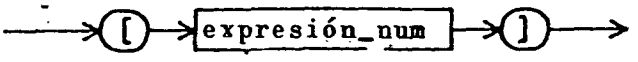
- La capacidad de la memoria se reduce, ya que una palabra de más de 32 bits está formada por varias palabras de 32 bits y el número máximo permitido de palabras de 32 bits es 65.536. Por ejemplo, 100 bits se almacenan en 4 palabras (128 bits), por tanto la longitud máxima es de 65.536/4=16.384.

Si la longitud de palabra es inferior a 32 bits, la primera memoria (donde guardamos el microprograma) no es diferente de las demás.

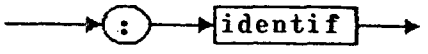
Dec memoria



Capacidad



Direcc.



Inic_mem

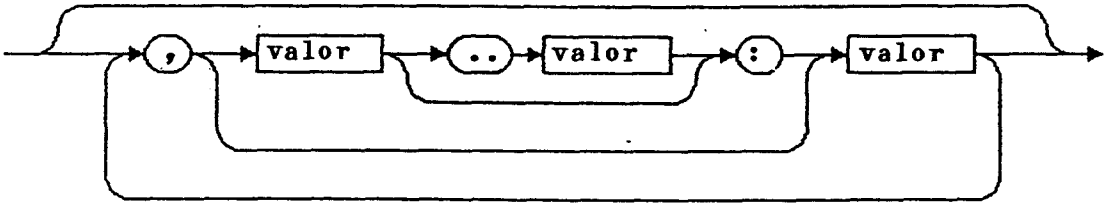


Figura 3.9

3.2.3.5.- Declaración de pilas.

La sintaxis se da en la figura 3.10. Una pila se caracteriza por su longitud y por su ancho. La pila se considera como un elemento asíncrono. Las dimensiones máximas son las mismas que para las memorias.

Las posibles operaciones que podemos realizar con una pila son: la lectura, la lectura con decrementación (PUSH), la escritura y la escritura con incrementación (POP). No es posible leer una pila vacía o escribir en

una pila llena.

Dec_Pila

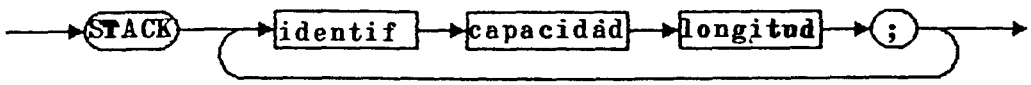


Figura 3.10

3.2.3.6.- Declaración de entradas.

La sintaxis se da en la figura 3.11. Una entrada se caracteriza por su ancho. La única operación posible es la lectura.

Dec_entrada

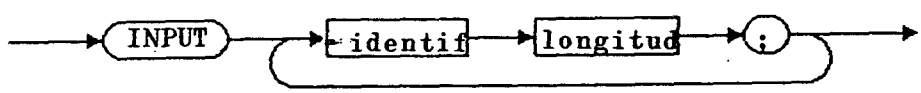


Figura 3.11

3.2.3.7.- Declaración de salidas.

La sintaxis se da en la figura 3.12. Una salida se caracteriza por su ancho. La única operación posible es la escritura.

Dec_salida

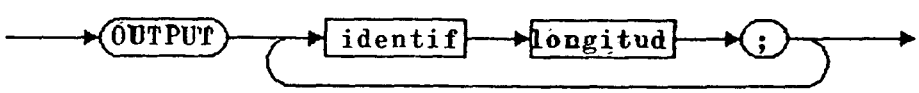


Figura 3.12

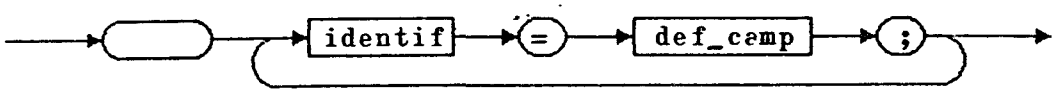
3.2.3.8.- Declaración de campos.

La declaración de un campo es una posibilidad de dar un mombre a un conjunto continuo de bits de un elemento. La sintaxis se da en la figura 3.13. La definición de un campo (Def_camp) puede hacerse de varias maneras:

- Un identificador: el campo está definido como todos los bits del elemento dado por su nombre.
- Un identificador con la especificación de los bits: el campo representa el o los bits especificados del elemento dado por el identificador. Los bits se dan por su número (el bit 0 es el de menor peso).
- Especificación de los bits: El campo representa el o los bits especificados de la memoria de microprograma (primera memoria declarada).

No es posible el definir un campo a partir de otro, ni definir un campo de más de 32 bits de ancho.

Dec_camp



Def_camp

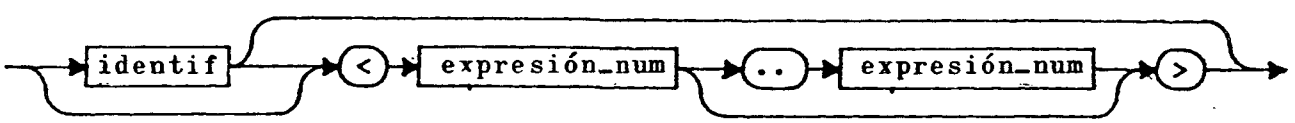


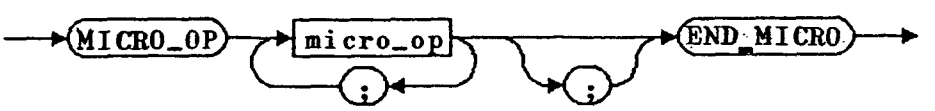
Figura 3.13

3.2.4.- MICROOPERACIONES.

El cuerpo o bloque principal de la descripción está formado por las microoperaciones (Figura 3.14). El conjunto de estas microoperaciones indica todas las transferencias a efectuar en un período de reloj.

Los apartados siguientes describen en detalle estas instrucciones.

Bloque



Micro_op

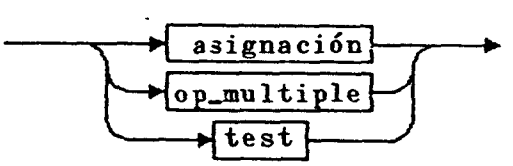


Figura 3.14

3.2.4.1.- Instrucciones de asignación.

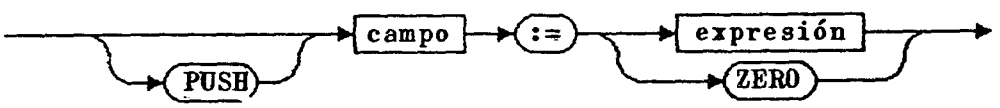
Una asignación, cuya sintaxis se da en la figura 3.15, define una transferencia de información. Esto es equivalente a una conexión unidireccional entre dos elementos.

El destino es un campo (Figura 3.16); es decir un

identificador de campo (declarado previamente) o una definición de campo. Si el campo se refiere a una pila, puede estar precedido de la palabra PUSH, añadiendose el valor a la pila. En caso contrario se sustituye el anterior valor del top de la pila por el nuevo. La fuente puede ser tanto una exprexión como la palabra clave ZERO, que equivale a borrar los bits del elemento o campo de destino.

Una expresión permite crear una función de dos o más elementos. Se compone de uno o dos operandos separados por un operador (OPER2, Figura 3.16). En cualquier caso, un operando puede modificarse por un operador de un solo operando (OPER1, Figura 3.16).

Asignación



Expresión

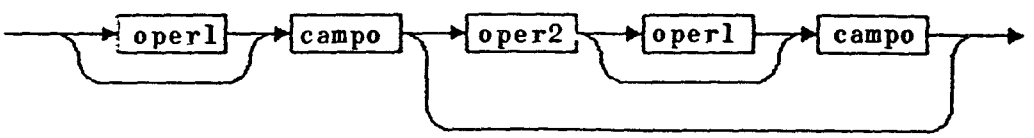


Figura 3.15

Los operadores que afectan a un solo operando son los siguientes:

- POP: Lectura con decrementación de la pila.
- NOT: Complemento lógico (cambia 0 por 1 y 1 por 0).

- NEG: Negación (O-operando, en complemento a 2).
- ASR, ASL: Decalage aritmético a derecha y a izquierda (Figura 3.17).
- LSR, LSL: Decalage lógico a la derecha y a la izquierda (Figura 3.18).
- RR, RL : Rotación a la derecha y a la izquierda (Figura 3.19).
- INC : Incrementación.
- DEC : Decrementación.

Para las operaciones INC y DEC, un rebosamiento añade un bit al resultado (no debe sobrepasarse los 32 bits). Para las otras, el resultado tiene el mismo número de bits que los datos.

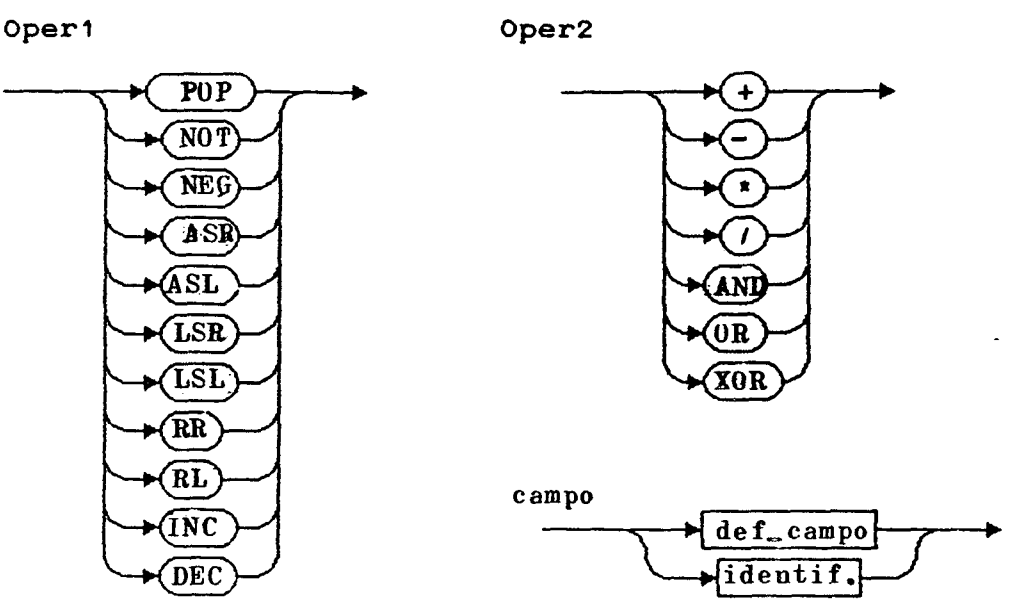


Figura 3.16

Los operadores que afectan a dos operandos son los siguientes:

- "+": Adición.
- "-": Sustracción.
- "*": Multiplicación.
- "/": División.
- "AND": AND lógico.
- "OR": OR lógico.
- "XOR": XOR lógico.

Los operadores lógicos actúan bit a bit; para los operadores aritméticos, sólo se retienen los operadores de menor peso.

ASR

ASL

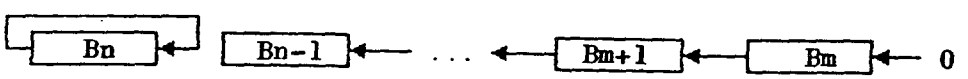
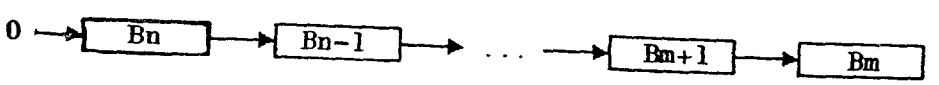


Figura 3.17

LSR



LSL

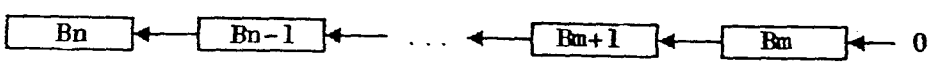


Figura 3.18

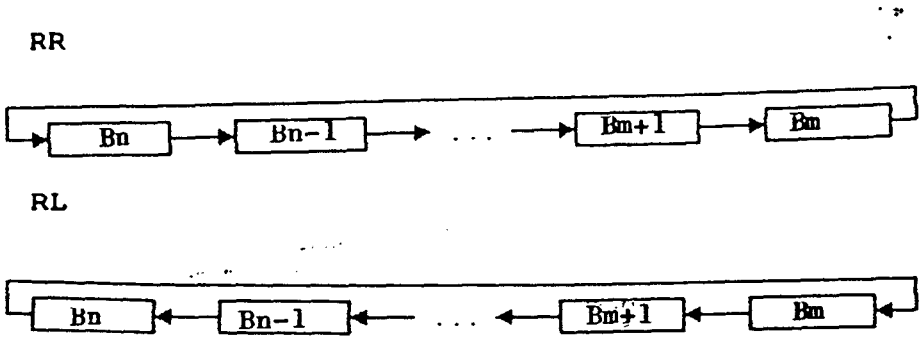


Figura 3.19

Si los operandos no poseen el mismo número de bits, el de menor número de bits se completa con ceros a la izquierda. Esto mismo ocurre para el resultado de la expresión, si es más pequeño que elcampo donde va a ir destinado. En el caso inverso solamente los bits de la derecha (menor peso) del resultado se transfiere al destino.

Estas operaciones básicas permiten calcular todas las funciones posibles, pero no podemos realizarlo de manera directa. Por ejemplo, si deseamos calcular

```
reg_C:=NOT (reg_A AND reg_B);
```

donde 'reg_A', 'reg_B' y 'reg_C' son los registros u otros elementos. Esta expresión no se ajusta a la sintaxis de la figura 3.15, debiendo proceder en varias etapas, de la siguiente manera:

```
temp := reg_A AND reg_B;
reg_C := NOT temp;
```

donde 'temp' es un bus, utilizado para almacenar temporalmente el resultado intermedio. Es necesario que

'temp' sea un bus y no un registro, ya que el bus es asíncrono y además se efectúa un test antes de la lectura del bus para saber si éste ha sido inicializado. Si 'temp' no ha sido utilizado en ninguna otra parte, existe otra posibilidad de escribir la expresión anterior ya que el lenguaje de descripción es no-procedural. Así podemos poner

```
reg_C := NOT temp;
temp := reg_A AND reg_B;
```

ya que las operaciones se ejecutan en un orden que a priori puede ser cualquiera. Ello conlleva a que la primera instrucción no se pueda ejecutar en tanto la segunda no haya terminado.

3.2.4.2.- Instrucciones de test

Las instrucciones de test (Figura 3.20) permiten elegir las instrucciones a ejecutar en función del valor presente de un campo. Existen dos posibilidades (Figura 3.21):

- El TEST SIMPLE: el valor se compara con un solo valor y las microoperaciones se ejecutan si ellos son iguales; en el caso contrario, no es posible su ejecución.

- El TEST MULTIPLE: El campo se compara con una serie de valores, ejecutándose la microoperación corres-

pondiente. En el caso de que no se presente ningún valor que corresponda a los del test no se ejecutará ninguna instrucción. Cada valor de la lista debe aparecer al menos una vez y debe estar comprendido entre 0 y 2^n-1, donde n = número de bits del campo.

Test

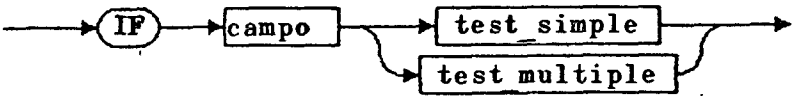
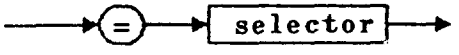
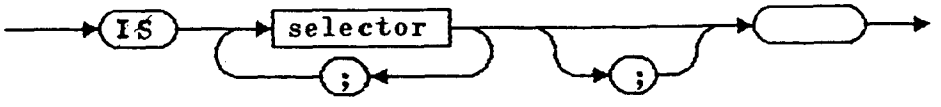


Figura 3.20

Test_simple



Test_multiple



Selector

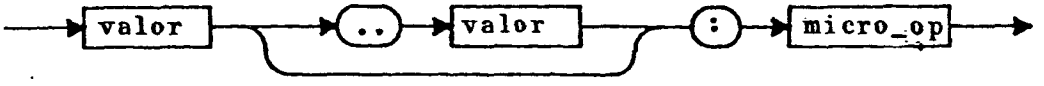


Figura 3.21

3.2.4.3.- Instrucción multiple.

Se trata simplemente de una microoperación formada por una o varias microoperaciones, delimitadas por dos palabras claves (Figura 3.22). Ello permite, por ejemplo en un test, el realizar varias microperaciones por un solo valor.

Op_multiple

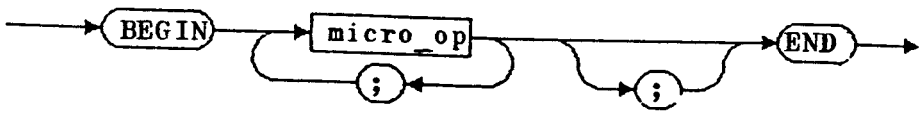


Figura 3.22

3.3.- PROGRAMA INTERACTIVO DE SIMULACION.

Este programa simula la ejecución de un programa sobre la máquina descrita, en diálogo con el usuario.

3.3.1.- ENTRADAS-SALIDAS.

El simulador requiere dos ficheros de entrada:

- El fichero de definición compactada generado por el compilador de la descripción. Contiene la estructura de los datos correspondientes a la arquitectura de la máquina y de las microoperaciones que permiten simular su funcionamiento.
- El fichero binario generado por el microensamblador. Contiene el código del microprograma que vamos a simular. Este fichero es cargado automáticamente en la primera memoria declarada. La longitud de esta memoria debe, por tanto, corresponder a la longitud declarada en el microensamblaje.

Es posible cargar un fichero binario en una memoria cualquiera, para inicializar, por ejemplo, una ROM de decodificación. Este fichero puede ser generado por el microemsamblador o editado manualmente. El formato de este fichero es el siguiente:

- Pasa de las líneas en blanco y toma las líneas de datos.

- Una línea contiene la dirección, un espacio y el dato.
- Los datos están formados por ((n+3) DIV 4) cifras hexadecimales, donde n es el número de bits de una palabra de memoria.
- Una cifra hexadecimal son los digitos 0..9, A..F, que representa los números decimales desde el 0 hasta el 15.

En el transcurso de la simulación, el programa debe acceder a elementos de entrada o salida. En la versión actual sobre VAX las salidas se vuelcan a la pantalla del terminal TRC y las entradas deben ser tecleadas cuando lo demande el programa. El usuario puede interrumpir la simulación pulsando dos veces <RETURN>.

Todos los comandos del simulador se entran desde el teclado y todas las salidas se vuelcan en pantalla. El simulador es independiente del terminal utilizado, ya que no llama a ninguna función especifica del mismo como paginado o desplazamiento del cursor.

3.3.2.- ESTRUCTURA DE DATOS

Todos los elementos se memorizan mediante variables dinámicas (punteros) en un árbol. A cada elemhto se le asocia su nombre y sus características. Además:

- Un registro contiene dos valores: el valor presente y el futuro. Después de un período de simulación (un ciclo de reloj) el valor futuro se transforma en valor presente.
- Un latch contiene el valor actual.
- Un bus contiene un valor y una variable que indica si el valor está definido. Esta variable se inicializa a "indefinida" al principio de cada ciclo.
- Una memoria contiene el nombre del elemento que la direcciona. Los valores se representan en una tabla. En realidad esta tabla es de la forma

  ARRAY [0..255] OF ^ARRAY [0..255] OF INTEGER

  Representa un buen compromiso entre el tamaño total y el espacio ocupado, pero no utilizado (si la longitud de la memoria no es un múltiplo de 256) más el espacio requerido para la dirección (la primera tabla de punteros).
- Una pila contiene el último valor disponible (-1 si la pila está vacía). Los valores se guardan de la misma manera que para el caso de las memorias.
- Las entradas y las salidas no contienen más que un número de orden, que se le asigna en su creación. No se memoriza ningún valor.

Las microoperaciones se registran también en variables dinámicas y ello permite , en teoría, simular un

sistema independientemente de su complejidad. En la práctica, el tamaño de la memoria y la velocidad del procesador pueden limitar la dimensión del sistema a simular.

3.3.3.- METODOS DE CALCULO

Es útil conocer los métodos de cálculo usados con el fin de comprender y de utilizar de manera eficiente el simulador.

3.3.3.1.- Cálculo de una asignación.

Todos los bits de menor peso del primer operando se extraen y se almacenan en los de menor peso de una variable, mientras que los bits de mayor peso permanecen inalterados. En este estado, los posibles errores que pueden producirse son:

- Lectura de una pila vacía
- direccionamiento de una memoria fuera de los límites declarados.

Este valor, ya colocado en la variable, es modificado por un operador si ello está indicado. Esto no producirá ningún errror. Si existe un segundo operando, se extrae y modifica de la misma manera y después se realiza la operación indicada. Solamente la división puede producir dos errores:

- División por cero

- División de cero por cero

Los bits de menor peso del resultado se almacenan en el elemnto de destino. Solamente se modifican los bits indicados por el campo. Los posibles errores que se presentan son los siguientes:

- Direccionamiento de una memoria fuera de los límites declarados
- Escritura de una pila vacía (solamente sin PUSH)
- Incrementación de una pila llena (solamente con PUSH).

3.3.3.2.- Evaluación de un test

El campo a testear se extrae de la misma manera que los operandos de una expresión, y por tanto podrán aparecer los mismos errores. Después este valor se compara con los valores de selección. Si alguno de ellos es idéntico, la ejecución se produce con las instrucciones correspondientes. Si el valor no coincide, la ejecución continúa con la instrucción siguiente al test.

3.3.3.3.- Algoritmo de simulación

Para simular un período de reloj, las microoperaciones son tratadas en el orden de su aparición en la descripción, teniendo en cuenta los tests: Si la ejecución de una microoperación no puede ser posible, esta se almacena en una lista de atención.

La ejecución es imposible si la instrucción demanda la lectura de un bus en el que todavía no se ha almacenado ningún valor o el acceso a una memoria cuya dirección no está disponible. Si se trata de un test, todas la microoperaciones cuya ejecución está controlada por dicho test se añaden a la lista.

Una vez que todas las microoperaciones han sido tratadas, las que permanecen en la lista se tratan de nuevo, y creando eventualmente una segunda lista de atención. El proceso continúa hasta que se presentan una de las dos situaciones siguientes:

- La última lista de atención creada está vacía; todas las microoperaciones han sido ejecutadas
- Cuando no se puede ejecutar alguna operación, quedando como última de la lista. Se trata de una secuencia imposible de calcular.

Una secuencia imposible puede provenir por ejemplo de la utilización de un bus que no es afectado en el ciclo, o también de la asignación recíproca de dos buses:

```
bus_1 := bus_2;
bus_2 := bus_1;
```

Las memorias cuya dirección viene dada por un bus pueden presentar estos tipos de problemas.

3.3.4.- DESCRIPCION DE LOS COMANDOS.

Cuando demandamos la ejecución del simulador interactivo mediante el comando

$ <@SIM2>

el programa nos pedirá los nombres de los ficheros descritos con anterioridad apareciendo en pantalla los siguientes mensajes:

```
Nombre del fichero de definición (default .REG):<    >
Nombre del fichero binario (default .BIN):<    >
run sim2 /definición=<    >/binario=<    >

SIMULADOR Versión 2.4 DPTO. ELECTRONICA UPC MARZO 1.987
=======================================================

*****EJECUCION:

Para visualizar el menú pulsar "?"

Lectura de definiciones
Lectura del micro-programa de la memoria
SIM>
```

Ahora podemos teclear un comando. Un comando se compone de una o dos letras mayúsculas o minúsculas (son indiferentes) y completado en algunos casos por uno o varios parámetros. La tecla <RETURN> desencadena la ejecución del comando o el display en pantalla del mensaje de error "comando desconocido". Cuando un comando requiere parámetros y no se especifican en él , el programa los demandará explícitamente, pudiendo

abandonar el comando, si se desea, pulsando de nuevo la tecla <RETURN>.

Se pueden dar dos o más comandos en una misma línea, separados al menos por un blanco. En caso de error en la ejecución de algunos de los comandos se ignora la ejecución del resto de ellos.

Es posible obtener un resumen de los comandos tecleando "?" después del prompt:

```
SIM> ?

    Menú principal de comandos :
    ============================

    A*       :  Visualiza un elemento
    I*       :  Inicializa un elemento
    R        :  Visualiza todos los Registros
    M        :  Visualiza todas las Memorias
    P        :  Visualiza todas las Pilas
    E*       :  Ejecución
    C*       :  Condición de parada
    B*       :  Cambio de la Base de los números
    L mem f  :  Carga el fichero "f" en la memoria "mem"
    F        :  Fin de la simulación
    ?        :  Muestra este texto

    Los comandos seguidos de '*'pueden tener varios
    parámetros, donde la descripción se obtiene
    tecleando '?' despues de la letra
```

Los comandos se describen en los apartados siguientes.

3.3.4.1.- Comandos de visualización.

Permiten tener conocimiento del contenido de todos

los elementos. Por contenido podemos entender:

- Para los registros, el valor actual;
- Para los latches, el valor en curso;
- Para las memorias, el contenido de la posición dada por la dirección, si la memoria no sobrepasa los 32 bits de largo, o de otra manera el mensaje "demasiado larga". Si la dirección no está disponible aparecerá el mensaje "indefinida".
- Para las pilas, la cima (top) de la pila ("indefinida" si la pila está vacía).
- Para los buses, el valor actual, si el bus ha sido utilizado en el último ciclo.

Tecleando "A?" después del prompt, aparecerá la lista de los comandos de visualización:

```
SIM> A?
     Menú  de comandos de Visualización :
     ===================================

     [opción: 0 o 1 ]  (repetición : n )

     A  nom       :  Visualiza un elemento dado  por  su
                     nombre
     AL           :  Visualiza todos los Latches
     AB           :  Visualiza todos los Buses
     AE           :  Visualiza todas las Entradas
     AT           :  Visualiza Todos los elementos
     AM nom adr [ n ]:
                     Visualiza "n" palabras de la memoria o
                     de la pila "nom" a partir de la
                     dirección "adr".Por defecto n=1.
     AP nom       :  Visualiza el puntero de la pila "nom"
     AS nom b1 b2 :Visualiza: nom <b1..b2>; (b2 - b1<32.)
     A?           :  Visualiza este texto
```

Veamos la descripción de estos comandos:

A nom muestra el contenido del elemento "nom". Si el elemento no existe aparecerá el mensaje de error "elemento inexistente". Es posible visualizar una memoria de más de 32 bits de largo.

AL muestra todos los latches por orden alfabético

AB muestra todos los buses por orden alfabético.

AE muestra todas las entradas, por orden alfabético. Sobre el VAX este comando no es muy útil ya que el programa demandará los valores de cada entrada.

AT muestra todos los elementos por orden alfabético.

AM nom adr [n] muestra el contenido de una memoria o de una pila. La dirección de la primera palabra es obligatoria; el número de palabras es por defecto 1. Si el nombre dado no corresponde ni al de una memoria ni al de una pila aparecerá un mensaje de error "ni memoria ni pila", lo mismo que si la dirección sobrepasa el tamaño de la memoria "direccionamiento fuera de límites".

AP nom muestra el puntero, es decir, el número del último elemento de la pila, o el mensaje "pila VACIA". Si el nombre no corresponde al de una pila, aparecerá el mensaje "pila esperada".

AS nom b1 b2 visualización especial: únicamente aparecerán en pantalla los bits b1 a b2 del elemento nombrado. b1 y b2 deben ser compatibles con la dimensión del elemento y b2-b1 debe ser inferior a 32, si no

aparecerá el mensaje de error "bits incorrectos".

3.3.4.2.- Comandos de inicialización.

Permiten inicializar cualquier elemento o parte de él. El listado de comandos de inicialización se obtiene tecleando "I?" después del prompt:

```
SIM> I?
     Menú  de comandos de inicialización :
     ====================================

     [opción: 0 o 1 ]  (repetición : n )

       I nom val   : "nom" := "val"
       IM nom adr val ( , val ) :
           Inicializa la memoria o la pila "nom" con los
           valores "val" después de la dirección "adr"
       IP nom val  : Inicializa el puntero de la pila
                     "nom" con el valor "val"
       IV nom      : Inicializa la pila "nom" al estado
                     VACIA
       IS nom b1 b2 val :
             nom <b1..b2> := val; (b2 - b1 < 32.)
       I?        : Visualiza este texto
```

Veamos la descripción de estos comandos:

I nom val inicializa el elemnto de monbre "nom" al valor "val". Solamente se modifican los bits correspondientes a la declaración del elemento. Si se trata de una memoria de más de 32 bits sólo se modifican los de menor peso. Cuando pretendemos realizar la inicialización de un elemento se visualizará en pantalla una confirmación del nuevo valor que ha tomado el elemento,visualizándose

el antigüo y el nuevo valor tomado por el elemento. En caso contrario aparecerá el mensaje "Lectura imposible" o "Escritura imposible".

IM nom adr val (, val) inicializa una memoria o una pila, a partir de la dirección "adr" con el valor "val" dado. La dirección se incrementa a cada valor. Si la línea se termina por una coma entonces se demandarán uno o varios valores. Las direcciones deberán estar en los límites declarados de las memorias o de las pilas. Si no aparecerá el mensaje " direccionamiento fuera de límites".

IP nom val inicializa el puntero de una pila; cambia el nivel de la pila sin modificar su contenido. "Nom" deberá ser por tanto una pila ("pila esperada") y "val" debe estar en los límites declarados de la pila ("direccionamiento fuera de límites"). El cambio se confirma mediante la visualización en pantalla del antigüo y del nuevo valor. Mediante este comando no podemos poner una pila a estado de vacío. Para ello se emplea el comando siguiente.

IV nom inicializa el puntero de una pila a -1, es decir, que la pila está vacía.

IS nom b1 b2 val inicialización especial: solamente se modifican los bits comprendidos entre b1y b2; esto permite inicializar, en varios pasos, el contenido de una palabra de memoria de más de 32 bits de largo. Igual que en los casos anteriores, se muestran en pantalla los

viejos y nuevos valores de los bits como confirmación de la modificación efectuada.

3.3.4.3.- Comandos de ejecución.

Independientemente del modo de ejecución, la simulación se para al final de un ciclo, o por la aparición de un error. Los posibles errores han sido señalados en los parrafos anteriores. Además la parada de la simulación se puede producir por:

- Secuencias imposibles en un ciclo de ejecución;
- Condición de saltos reencontradas;
- Paradas por el usuario.

El listado de los comandos de ejecución se obtienen tecleando "E?" después del prompt:

```
SIM> E?
    Menú de comandos de ejecución :
    ================================
    [opción: 0 o 1 ]  (repetición : n )

    E [nb]   : Ejecución de nb pasos (por defecto nb=1)
    EC       : Ejecución continua (hasta un error)
    ED [nb]  : Ejecución de nb pasos con Debugging
    ET nom ( , nom ) [ nb ] :
               Ejecución de nb pasos, con Trace de
               los elementos nombrados
    E?       : Visualiza este texto
```

Veamos la descripción de estos comandos:

E [nb] ejecución durante "nb" períodos de reloj. La simulación se parará después de los nb pasos o si

aparece un error de ejecución.

EC ejecución continua, hasta que aparezca un error o por una parada provocada por el usuario.

ED [nb] ejecución de "nb", con deebug, es decir con visualización de las microoperaciones tratadas. Para una microperación de asignación, la visualización toma la forma siguiente:

nnnn asg: elemento b1 b2 := resultado

donde "nnnn" es el número de la línea en el fichero de descripción; asg indicará que se trata de una asignación; "elemento b1 b2" es el nombre y los bits del elemento destinatario; "resultado" es el valor obtenido por el cálculo de la expresión si la microoperación está completamente ejecutada. Si no lo está, se reemplaza por "*****".

Para una microoperación de test, se visualizará

nnnn test : elemento b1 b2 == resultado

donde "nnnn" es el número de línea; "test" indica que se trata de un test; "elemento b1 b2" es el nombre y los bits del elemento a testear; "resultado" es el valor o "*****".

Cada pasada a una línea siguiente de las microoperaciones se marca por una línea en blanco.

ET nom (, nom) [nb] ejecución de "nb" pasos con trace, es decir, con la visualización de los elementos después de cada paso. Cada "nom" representa un elemento, del que queremos conocer su valor. El programa demandará más

nombres de elementos hasta que no aparezaca una coma.

3.3.4.4.- Comandos de tratamiento de condiciones de parada.

Una condición de parada está formada de un conjunto de bits de un elemento y de un valor. Después de cada ciclo, si el test de condiciones está activo, los bits del elemento se comparan al valor y la simulación se para si son iguales. La condición no puede considerarse como satisfecha si no es posible leer el elemento.

El listado de estos comandos se obtienen tecleando "C?":

```
SIM> C?
    Menú de los comandos de condición :
    ===================================
    [opción: 0 o 1 ]  (repetición : n )

    C cond  : Crea un condición aislada
    C cond (,cond)  :
      Crea una serie de condiciones ligadas (AND lógico)
    CL      : Visualiza todas las condiciones
    CD      : Visualiza las condiciones para su
              destrucción
    CA      : Condiciones Activas
    CN      : Condiciones No Activas
    C?      : Visualiza este texto

    Sintaxis  de "cond" : nom [<bit1 [..bit2]>] valor
              Una condición es un elemento (ev. con
              indicación de los bits concernientes) y un
              valor. La condición es verdadera hasta que
              el elemento es igual al valor.

    Las condiciones están actualmente (in)activas.
```

C cond crea una condición aislada, es decir, independiente de las demás. Si durante la simulación se satisface esta condición, la ejecución se interrumpe.

C cond { , cond} crea una serie de condiciones ligadas, es decir, que es necesario que se satisfagan todas las condiciones de la serie para que la ejecución se pare. En tanto que aparezcan comas el programa espera una condición y las liga a las condiciones precedentes de la serie.

Una condición puede tomar cualquiera de las tres formas siguientes:

- Nombre valor: el test se realiza con todos los bits del elemento, salvo que la memoria sea de más de 32 bits, ya que en este caso el test se realiza con los 32 bit de menor peso.
- Nombre <b1> valor: el test concierne solamente al bit b1.
- Nombre <b1..b2> valor. El test se efectúa sobre los bits b1 a b2, para aquellos que no representen más de 32 bits.

CL visualiza la lista de todas las condiciones, dándonos una condición por línea. Si la línea se termina por "AND" ello significa que esta condición está ligada a la siguiente.

CD visualiza la lista de las condiciones y pregunta para cada una si se conserva dicha condición o se destruye si el primer caracter de la respuesta es "O".

CA activa las condiciones.

CN vuelve las condiciones a inactivas.

3.3.4.5.- Comandos de cambio de base.

Todos los números leidos desde el teclado y la mayor parte de los visualizados en la pantalla se expresan en la base actual, sea 2, 8, 10, o 16. Al comienzo del programa, la base actual es hexadecimal. El listado de comandos y la base actual se visualizan tecleando "B?" despues del prompt:

```
SIM> B?
    Menú de comandos de cambio de base :
    ====================================

    BB      : Base binaria (2.)
    BO      : Base Octal (8.)
    BD      : Base Decimal (10.)
    BH      : Base Hexadecimal (16.)
    B?      : Visualiza este texto

    Todos los números leidos o visualizados son expresa-
    dos en la base actual (con la excepción de los
    números seguidos de un punto, que son escritos en
    decimal).
    La base actual es 16.
```

En base 2, 8, o 16, los números se visualizan como los números positivos de 32 bits. En base 10 se tratan como números con signo en complemento a 2, salvo el 2**31 que se visualiza como -0. Además, en base 10, es imposible de dar, por el teclado, un número de 32 bits; la conversión en enteros negativos no se realiza.

3.3.4.6.- Comandos diversos.

R muestra todos los registros, por orden alfabético.

M muestra todas las memorias por orden alfabético; el contenido de una memoria de más de 32 bits no es posible visualizarla.

P muestra las cimas (top) de todas las pilas, por orden alfabético.

L mem f carga el fichero binario "f" en la memoria "mem". Se ha de tener cuidado en que las dimensiones de la memoria y de los datos en el fichero sean compatibles (ver apartado 3.1.). Además se debe tomar bastante atención en escribir el nombre del fichero ya que un error provoca la parada del programa.

F fin de la simulación. Se demanda una confirmación.

3.4.- MENSAJES DE ERROR DADOS POR EL COMPILADOR SIM1.

Los mensajes de error dados por el simulador y que aparecen en la pantalla son los siguientes :

1: Caracter ilegal.

2: Número demasiado largo.

3: Entrada demasiado grande.

4: Número que contiene una cifra ilegal.

5: Símbolo ilegal.

10: Se espera ";".

11: Se espera "=".

12: Se espera "<".

13: Se espera ">".

14: Se espera "[".

15: Se espera "]".

16: Se espera ",".

17: Se espera ":".

18: Se espera ":=".

20: Se espera "identificador".

21: Se espera "valor".

22: Se espera "operador" (+, -, * o /).

23: Se espera "definición de campo".

24: Se espera "microoperación".

25: Se espera "IS" o "=".

26: Se espera "constante".

27: Se espera "referencia de campo".

28: Se espera "declaración de memoria".

29: Se espera "identificador de pila".

30: Se espera "MACHINE".

31: Se espera "CONST","REGISTER","BUS" o "MEMORY".

32: Se espera "REGISTER", "BUS" o "MEMORY".

33: Se espera "BUS" o "MEMORY".

34: Se espera "MEMORY".

35: Se espera "MICRO_OP".

36: Se espera "END_MICRO".

37: Se espera "END".

38: Se espera "END_IF".

39: Se espera "PUSH".

40: Identificador no declarado.

41: Identificador ya declarado.

42: Valor ya declarado.

43: Direccionamiento fuera de los límites de la memo
ria.

44: Lectura de una salida.

45: Escritura de una entrada.

46: Inicialización prohibida para esta memoria.

50: Rebosamiento de una adición.

51: Rebosamiento de una sustracción.

52: Rebosamiento de un producto.

53: División por 0.

54: Valor negativo no admitido.

55: Valor nulo o negativo no admisible.

56: Valor que se sale del campo.

57: Bit fuera del campo.

58: Valor superior < valor inferior.

59: Campo limitado por la implementación.

60: Memoria limitada por la implementación.

90: Fin; El resto del texto es ignorado.

99: Demasiados errores en esta línea.

APENDICE B

El programa fuente de SILOSS está escrito en un excelente PASCAL perfectamente estructurado, esto facilita la labor a la hora de añadir procedimientos para obtener información adicional, al respecto del proceso de simulación.

Para obtener un análisis dinámico, análisis en tiempo de ejecución, SILOSS originalmente no disponía de un procedimiento que proporcionara datos relativos a que instrucciones son las que se ejecutan y que número de veces, a lo largo de un programa.

Para solucionar esto se ha desarrollado un pequeño procedimiento que produce un listado en tiempo de ejecución o de simulación, del valor que toma en cada ciclo de reloj un elemento (a definir según interese) de la arquitectura. Eligiendo como aquí se ha hecho, el REGISTRO de INSTRUCCION, se puede observar que instrucción (opcode) de la memoria de programa externa se está ejecutando o simulando, en cada período de reloj. Como una instrucción se ejecuta en varios ciclos de reloj (de 3 a 16 en CPA), se tiene presente para que en el listado final que se genera, aparezca sólo una vez en el momento en el cual se produce.

Finalmente se obtiene todas y cada una de las instrucciones que se ejecutan, que no tiene porque corresponder exactamente al número de opcodes que forman un programa estático, ya que en éste hay saltos, bucles de repetición, etc. Se obtiene así una información con la cual se realiza un estudio del tiempo en que se ejecuta un programa de prueba.

```
PROCEDURE obtener_dato;
        (**************)

VAR
   n_element     : p_element;
   y,y1          : mot;
   x,b2          : bit_2;
   w             : integer;

BEGIN
   lire_nom (n_element, 'elemento a testear  ');
   IF n_element <> NIL THEN
   WHILE  statut = normal  DO
   BEGIN
      execute;
         WITH n_element^  DO
         BEGIN
            IF largeur <= nb_max_bit
            THEN IF lire_element (nom, -1, 0, y)
                 THEN BEGIN
                         IF  y <> y1 THEN
                            BEGIN
                               write(w:5,'  ');
                               imprime (y, largeur);
                               writeln;
                            END
                      END
                 ELSE writeln ('lectura imposible')
            ELSE
            BEGIN
               b2 := largeur - 1;
               x := (b2 DIV nb_max_bit + 1) * nb_max_bit;
               REPEAT
                  x := x - nb_max_bit;
                  IF lire_element (nom, x, b2, y)
                  THEN BEGIN
                          IF  y <> y1 THEN
                             BEGIN
                                write(w:5,'  ');
                                imprime (y, b2 - x + 1);
                                writeln;
                             END
                       END
                  ELSE BEGIN
                          writeln ('lectura imposible');
                          x := 0;
                       END;
                  IF x <> 0 THEN b2 := x - 1;
               UNTIL X = 0;
            END;
            y1 := y;
            w:= w+1;
         END;
```

```
   IF cond_actif AND (statut = normal) THEN
   tester_condition;
   END;
END;   (*obtener_dato*)
```

```
PROCEDURE exec_commandes;
        (*****************)

BEGIN  (*exec_commandes*)
   readln;
   writeln;
   writeln ('   Menú  de comandos de ejecución :');
   writeln ('   ================================');
   writeln ('   [opción: 0 o 1 ]  {repetición : n }');
   writeln;
   writeln ('   E [nb]   : Ejecución de nb pasos (por defecto nb=1)');
   writeln ('   EC       : Ejecución continua (hasta un error)');
   writeln ('   ED [nb]  : Ejecución de nb pasos con Debug');
   writeln ('   EP nom   : Ejecución continua visualizando el');
   writeln ('              valor del elemento que se indica');
   writeln ('   ET nom { , nom } [ nb ] :');
   writeln ('              Ejecución de nb pasos, con Trace de');
   writeln ('              los elementos nombrados');
   writeln ('   E?       : Visualiza este texto');
   writeln;
   write (prompt);
END;   (*exec_commandes*)
```

BIBLIOGRAFIA

Bibliografía.

-«Bit-slice Microprocessor Design , J. Mick & J.Brick McGraw-Hill, U.S.A 1980.

-«Bipolar Microprocessor Logic and Interface», Advanced Micro Devices Data Book, Sunnyvale, California 1983.

-«An emulation of the Am9080A», Shavit M., Advanced Micro Devices, Sunnyvale, California 1978.

-«A survey of GaAs Computer Designs», A. Nuñez, Proc. Euromicro 87, North Holland 1987 pp 662-670.

-«Diseño, simulación y experimentación de microprocesadores en "MICRO"», P. Carballo, P.F.C. ETSII UPC 1987.

-«Diseño de procesadores en AsGa con CI LSI», A. Nuñez, R. Sarmiento, P. Carballo, undo Electrónico, Boixareu Ed. Barcelona 1988 pp 73-79.

-«Diseño Microelectrónico en tecnología nMOS-VLSI de un controlador para un sistema de acceso por clave», A.Vega, P.F.C ETSII UPC 1987.

-«Microensamblador y desarrollo de una CPU microprogramada de alta velocidad basada en el Am29116A», A. Sanz, P.F.C. ETSII Valladolid 1987.

-«Outils d'aide a la microprogrammation et simulation des circuits integrés», P. Sommer, E. Sanchez, LSL-Ecole Polytechnique Fédérale de Laussane, 1983.

-«Introdution to VAX/VMS version 4.0», Digital Corp. Massachusetss 1984.

-«Guide to Text Processing on VAX/VMS», Digital Corp. Massachusetss 1984.

-«Algoritmos+Estructuras de Datos=Programas» Niklaus Wirth, IFT Suiza, 1976.

-«Organización de computadoras». Andrew S. Tanenbaum, Prentice Hall, México 1986.

-«Component Data Catalog» Intel Corp. 1985.

-«Fast TTL Logic Series», IC15, Philips Data handbook 1988.

AD[0..8]

CLK

PL43

U1 74F377

IR0 IR1 IR2 IR3 IR5 IR6

U4 63S141A

U3 63S141A

U2 63S141A

D[0..11]

U5 2918

U6 2918

U7 2918

U8 63S081A

U9 63S081A

U10

Y[0..9]

+5V

clk

U12 74F399

CLK RESET

U11

signo
cero
paridad
carry
daal
daah
intr

ub44 ub45 ub46 ub47

ub[0..63]

udat

U13 74F373

V[0..2]

2910A

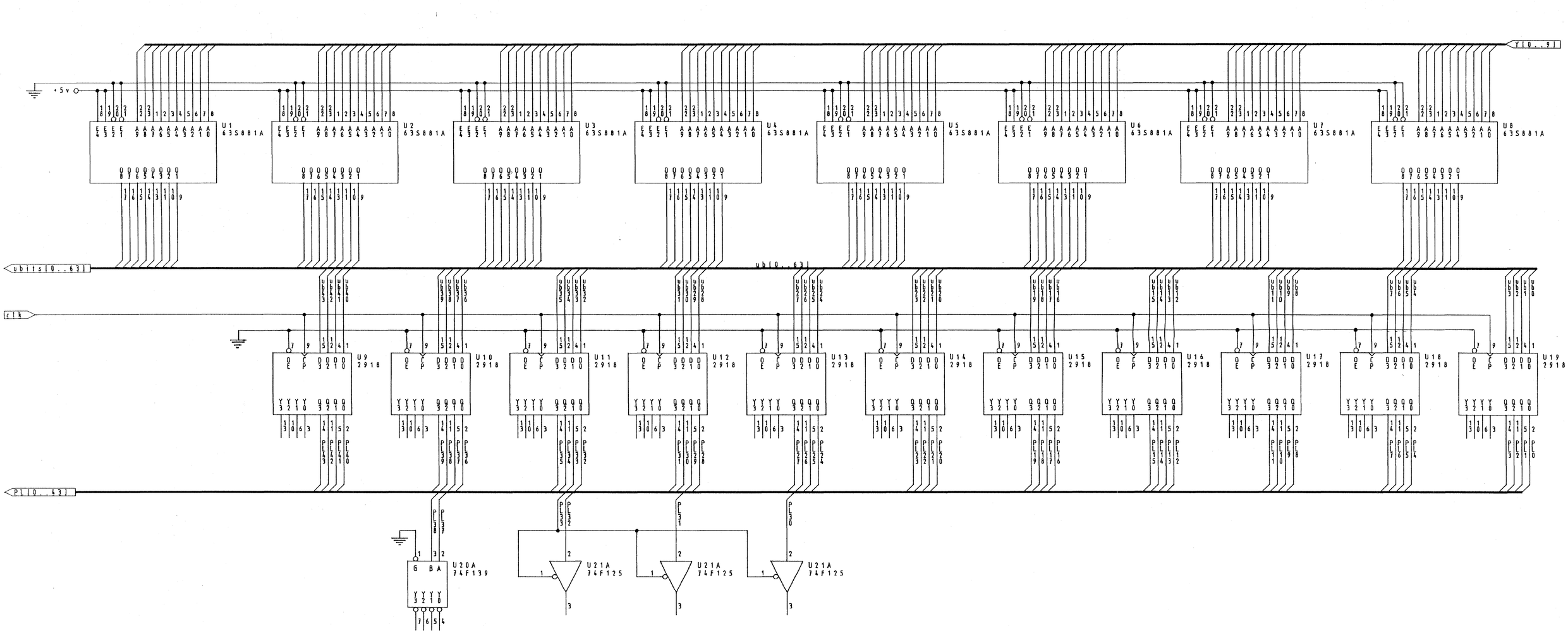
2922

SIMULACION DEL PROCESADOR CPA
ETSII DE LAS PALMAS (UPC)
Tutor: Dr. ANTONIO NUÑEZ ORDOÑEZ
Alumno: JOSE CARLOS PULIDO ACOSTA

Title: UNIDAD DE CONTROL DEL CPA

Size: C | Document Number: 1 | REV: 1

Date: April 13, 1989 | Sheet 1 of 1



SIMULACION DEL PROCESADOR CPA
ETSII DE LAS PALMAS (UPC)
Tutor: Dr. ANTONIO NUÑEZ ORDOÑEZ
Alumno: JOSE CARLOS PULIDO ACOSTA

Title
TARJETA DE MEMORIA DEL CPA

Size	Document Number	REV
C	2	
Date: April 13, 1989	Sheet 1 of 1	

SIMULACION DEL PROCESADOR CPA
ETSII DE LAS PALMAS (UPC)
Tutor: Dr. ANTONIO NUÑEZ DRODNEZ
Alumno: JOSE CARLOS PULIDO ACOSTA

Title
HADWARE DE LA UNIDAD DE EJECUCION DEL CPA

Size	Document Number	REV
D	3	1

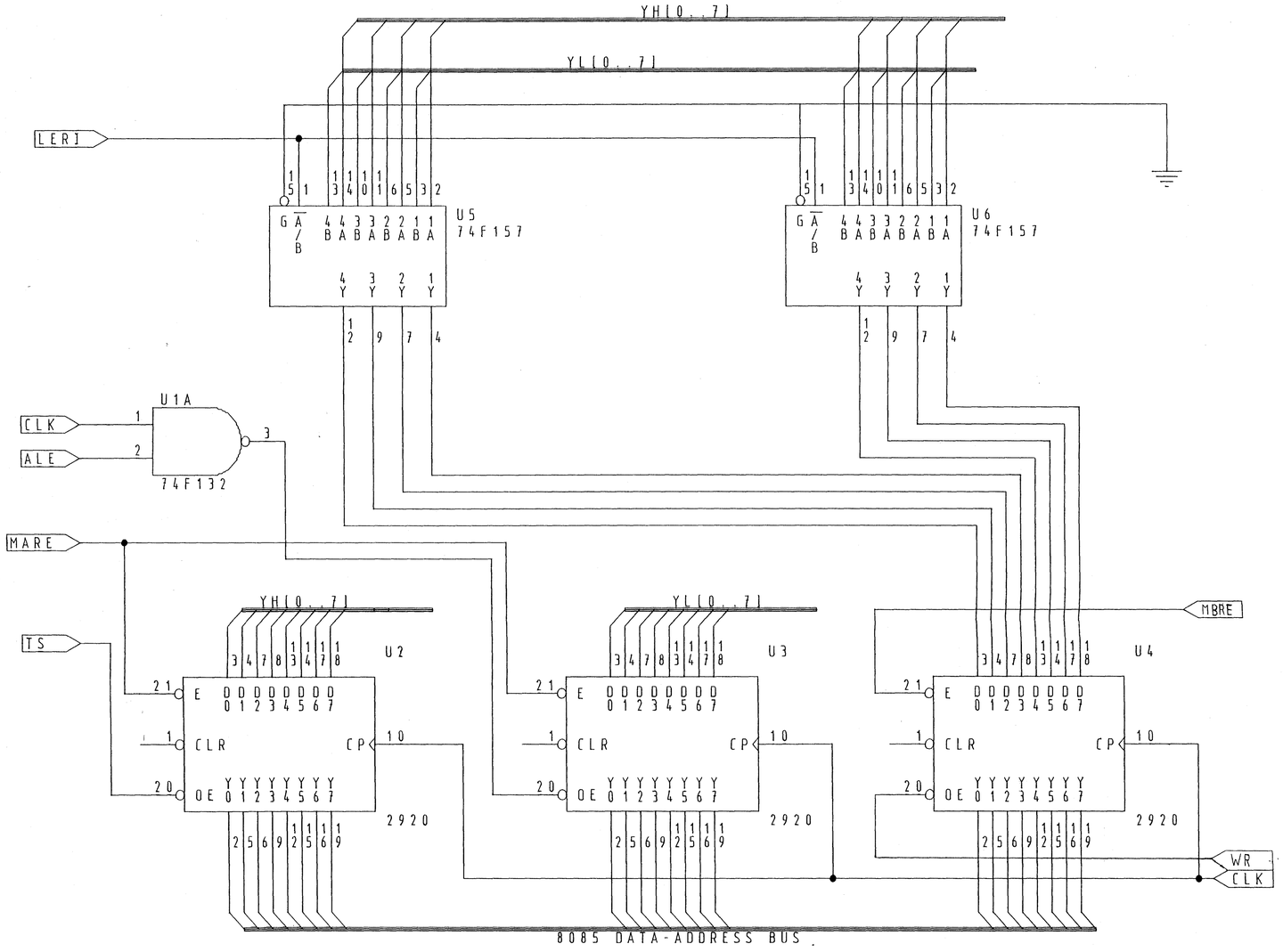
Date: April 13, 1989 | Sheet 1 of 1

HOLD
U16A
74F04
TRAP
+5V
U7A
74F74
U15A
74F804
RST 7.5
RESET 7.5
INTE
I7.5
RST 6.5
I6.5
RST 5.5
I5.5
INTR
U12A
74F08
U14A
74F08
U11A
74F00
M5.5
M6.5
M7.5
MSE
U17A
74F04
D D
2 3
U14A
74F08
D D D D D D
0 1 4 5 6 7
U9
74F30
U10A
74F109
EDI
U16A
74F32
to vect PROM *
U1
74F379
U2
2913
U3
74F374
U4
74F30
INTERRUPT DETECT
RD_MASK
U5
74F377
to internal DATA BUS
U6
74F377
from internal DATA BUS
clk
ST_MASK
U8A
74F74
U?A
74F126
SOD
SID

SIMULACION DEL PROCESADOR CPA
ETSII DE LAS PALMAS (UPC)
Tutor: Dr. ANTONIO NUÑEZ ORDOÑEZ
Alumno: JOSE CARLOS PULIDO ACOSTA

Title
LOGICA DE INTERRUPCIONES DEL CPA

Size	Document Number	REV
B	4	1
Date: April 13, 1989	Sheet 1 of 1	



SIMULACION DEL PROCESADOR CPA
ETSII DE LAS PALMAS (UPC)
Tutor: Dr. ANTONIO NUÑEZ ORDOÑEZ
Alumno: JOSE CARLOS PULIDO ACOSTA

Title
ESQUEMA DE INTERFACE DE CPA

Size	Document Number	REV
B	5	1
Date: April 13, 1989	Sheet 1 of 1	