UNIVERSIDAD DEL VALLE DE GUATEMALA

Facultad de Ciencias y Humanidades

DISEÑO DE UNA COMPUTADORA CON MULTIPROCESADORES

DAVID NICOLAS ALVAREZ ZECEÑA

Guatemala

1982

DISEÑO DE UNA COMPUTADORA CON MULTIPROCESADORES

UNIVERSIDAD DEL VALLE DE GUATEMALA

Facultad de Ciencias y Humanidades

DISEÑO DE UNA COMPUTADORA CON MULTIPROCESADORES

DAVID NICOLAS ALVAREZ ZECEÑA

Trabajo de Investigación Presentado para Optar
al Grado Académico de

LICENCIATURA EN CIENCIAS DE LA COMPUTACION

Guatemala

1982

PREFACIO

Desde los inicios de la computación electrónica por la década 1940-1950, una de las más grandes aplicaciones de la computadora digital se encontró en los cálculos numéricos. La velocidad y exactitud de estas máquinas las hacen una herramienta muy poderosa. Sistemas de computación como la Cray-1 de Cray Research son capaces de ejecutar más de 100 MFLOPS, resolviendo en segundos problemas que antes requerían meses de trabajo.

La finalidad de este proyecto es la de presentar el diseño de una computadora capaz de ejecutar operaciones en vectores y matrices a alta velocidad. Por alta velocidad me refiero a que otra computadora basada en la misma tecnología electrónica, el mismo procesador (INTEL 8085) y usando un diseño de uniprocesador requiere al menos un 75% más de tiempo para ejecutar el mismo número de operaciones aritméticas sobre una matriz o vector.

Se ha de mencionar que la computadora que se presenta no se halla limitada a trabajos sobre matrices. Se escogió esta aplicación porque es una de las más usadas. Se puede utilizar también para la ejecución de programas que puedan ser divididos en secciones paralelas para procesarlas concurrentemente y aumentar así la velocidad del programa en conjunto.

Sobre la parte de software se presentan como apéndice los programas que sirven de prueba de la funcionalidad del diseño. Se incluye también una posible implementación de las instrucciones "join" y "fork" de uso general en computadoras con multiprocesadores.

Las descripciones del diseño se hacen de dos tipos: para la parte de arquitectura se utilizan diagramas de bloques y PMS; para la descripción electrónica, planos de las interconexiones entre los diferentes componentes y diagramas lógicos que las representan. También se incluye como apéndice una descripción de cada uno de los componentes del sistema. Estas descripciones son las proporcionadas por la compañía fabricante.

CONTENIDO

LISTA DE FIGURAS

I. INTRODUCCION

Los diversos métodos para aumentar la velocidad de las computadoras se pueden dividir en dos ramas distintas: los métodos electrónicos y los estructurales. Generalmente, se usa una combinación de ambos.

A. LOS METODOS ELECTRONICOS

La forma más sencilla de aumentar la velocidad de una computadora es la de utilizar piezas electrónicas más rápidas. Estas han aumentado su velocidad desde la fabricación del primer transistor.

Las técnicas de fabricación actuales han llevado las piezas hasta tal punto que la velocidad de éstos ya no puede ser aumentada significativamente. Esto se debe a que no se puede transmitir información más rápido de lo que viaja la electricidad por los alambres de conexión, generalmente un 45% de la velocidad de la luz con alambres comunes y un 90% si se utilizan alambres coaxiales.

Se ha estado investigando también métodos criogénicos para aumentar la velocidad, enfriando los componentes a temperaturas cercanas al cero absoluto para disminuir la resistencia de los alambres y transmitir la información a la velocidad de la luz. Actualmente, se han obtenido velocidades de hasta 1,000 MFLOPS con máquinas experimentales.

B. LOS METODOS ESTRUCTURALES

Como se ve, los métodos electrónicos tienen su límite para aumentar la velocidad, además de ser costosos. Los diseñadores de computadoras buscan, como alternativa, hacer la arquitectura de la computadora más funcional y con posibilidad de ejecutar varias operaciones al mismo tiempo. Actualmente existen cuatro arquitecturas básicas.

1. Sistema de uniprocesador. Este sistema es el más usado en la actualidad; es prácticamente el mismo diseño que hizo Von Neumann en 1940. Su ventaja es su simplicidad. Consta de una unidad de control que maneja las relaciones entre los demás componentes, un procesador que ejecuta todas las instrucciones aritméticas y lógicas, un sistema de entrada/salida para comunicarse al exterior y la unidad de memoria para almacenar información. Generalmente el procesador y el control se hallan unidos en un solo bloque llamado procesador central. La desventaja principal estriba en que la única forma de aumentar la velocidad es utilizando componentes electrónicos más rápidos. La arquitectura de este sistema se presenta en la Fig. 1.

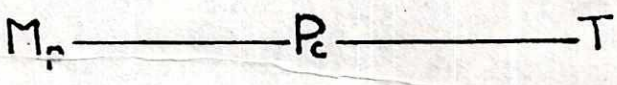


Fig. 1 PMS del Sistema Uniprocesador

2. Sistema multiprocesador. A diferencia del uniproceador este sistema tiene varios procesadores. Cada procesador tiene su propia unidad de control. Pero está conectado con los otros con fines de sincronización. La memoria está dividida en bloques que los procesadores comparten al igual que las unidades de entrada/salida. Su ventaja principal es la capacidad de ejecutar tantos procesos como procesadores existan, pudiendo ser éstos partes concurrentes de un mismo trabajo.

El problema más grande de este tipo de arquitectura es toda la lógica necesaria para sincronizar los procesadores, los accesos a memoria y el uso de los dispositivos de entrada/salida.

Se puede mencionar entre las máquinas que utilizan esta arquitectura a la ILLIAC IV, de 64 procesadores que se utilizó hasta 1981 cuando fue substituida por una máquina Cray-1 (con arquitectura de pipeline). La arquitectura del multiprocesador se aprecia en la Fig. 2.

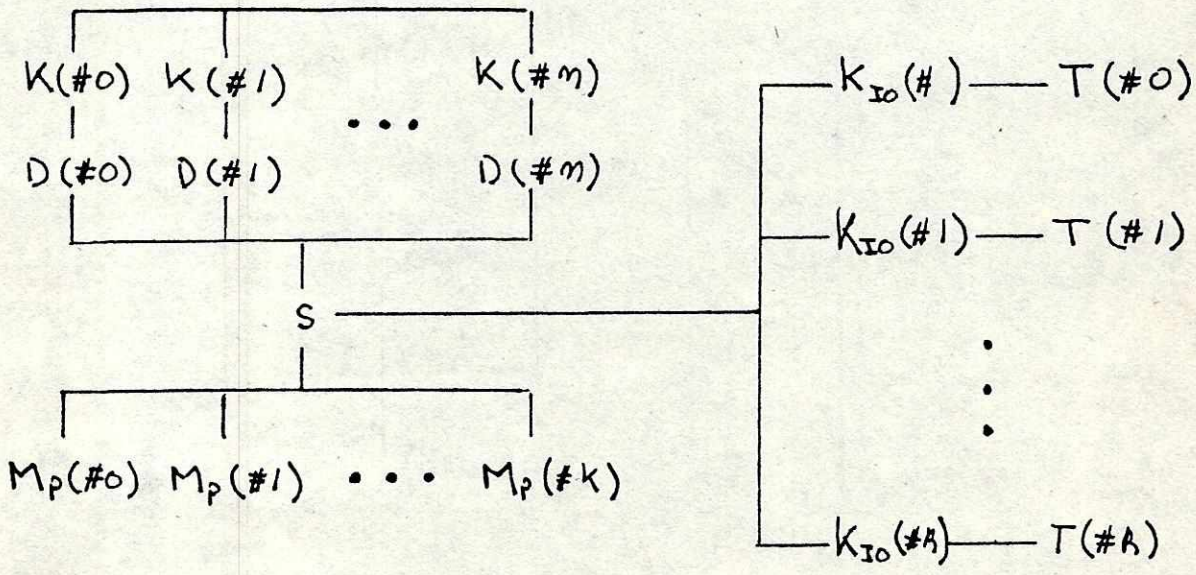


Fig. 2 PMS del Sistema Multiprocesador

3. Sistema de procesadores en paralelo. Esta arquitectura presenta cierta similitud con la de multiprocesadores, pues ambas tienen más de un procesador. La diferencia radica en que el procesador en paralelo sólo posee una unidad de control y en consecuencia, todos los procesadores deben trabajar al mismo tiempo con las mismas operaciones, sobre distintos operandos. Es muy práctico para vectores pues en ellos se ejecuta la misma operación sobre una serie de elementos.

La desventaja de este método es la poca flexibilidad que tiene. Puede llegarse a desperdiciar más del 50% de la máquina, si se trabaja sólo con escalares y no vectores. La sincronización de todos los procesadores no permite la ejecución de otras operaciones escalares mientras se está procesando ya una.

La arquitectura de estas máquinas se halla en la Fig. 3.

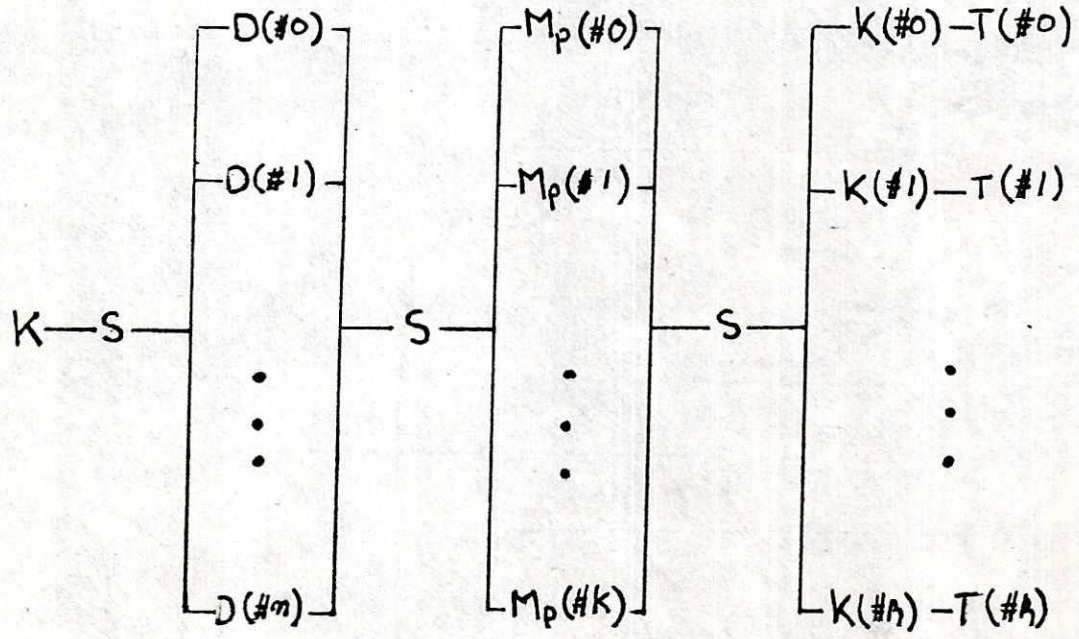


Fig. 3
PMS de la Arquitectura de Procesadores en Paralelo

4. Sistema de procesador de múltiple estación (Pipeline).

Esta arquitectura nos muestra otro concepto de cómo aumentar la velocidad. En ella sólo se encuentra una unidad de control, una serie de registros internos (o memoria local) y una serie de procesadores de función específica (sumar, restar, etc.). La conexión de los procesadores forma la múltiple estación. Estas se unen para ejecutar ciertas operaciones distribuidas en etapas.

La ventaja es que mientras una operación está en la etapa n, otras operaciones se encuentran en la etapa n-1, n-2, n-3, ...1, con el resultado de tener n operaciones ejecutándose al mismo tiempo.

Normalmente las computadores construidas de esta manera tienen varias estaciones múltiples para aumentar la concurrencia de las operaciones. La arquitectura se describe en la Fig. 4.

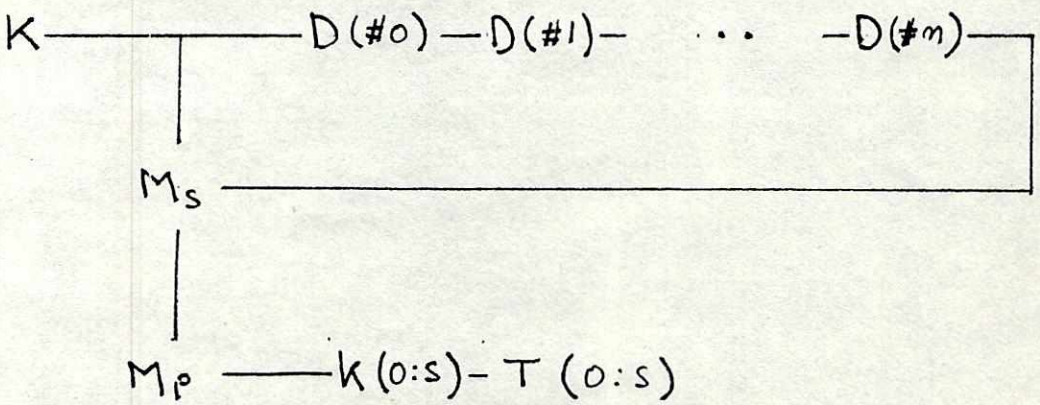


Fig. 4 Arquitectura de Múltiple Estación

II. OBJETIVOS

1. Diseñar una computadora basada en la arquitectura de multiprocesador, que sea funcional.
2. Demostrar que el diseño es más rápido para ejecutar operaciones sobre matrices y vectores que otro basado en un sistema de uniprocesador.
3. Demostrar la utilidad en otros campos como procesamiento concurrente de dos distintos algoritmos a una mayor velocidad.
4. Construir una cantidad mínima de software para operar el sistema, parte de ellos será la implementación de las instrucciones "join" y "fork".
5. Dejar todo el material necesario para que a partir de él se pueda construir otro modelo igual y expanderlo o modificarlo.

III. METODO

A. ARQUITECTURA GENERAL DEL SISTEMA

El sistema se basa en la arquitectura de multiprocesador. Se utilizan dos procesadores centrales con un sistema de sincronización, construidos alrededor de dos microprocesadores INTEL 8085 (versión avanzada del INTEL 8080) de ocho bits.

Cada procesador posee su propia memoria dividida en dos bloques de 1K Byte. Un bloque es memoria RAM y otro es ROM. Uno de los procesadores controla los medios de comunicación al exterior. El único dispositivo externo implementado es una microcomputadora Z90 de Zenith desde la cual se podrán introducir programas y/o datos.

Existe un bloque de memoria común a los procesadores. El bloque es de 4K Byte RAM y su organización es de tipo entrelazada (Interleaved) para permitir el acceso concurrente de los dos procesadores.

El diagrama PMS del sistema se encuentra en la Fig. 5 y el diagrama de bloques en la Fig. 6. Los bloques que lo componen son: Lógica de 'inicialización' del sistema, dos unidades centrales de procesamiento, cada una con su memoria y su lógica de acceso; un sistema de entrada/salida, un sistema de sincronización una lógica de 'ready' para cada procesador, un arbitro y la memoria común. Cada uno de ellos se describe en detalle en las siguientes secciones.

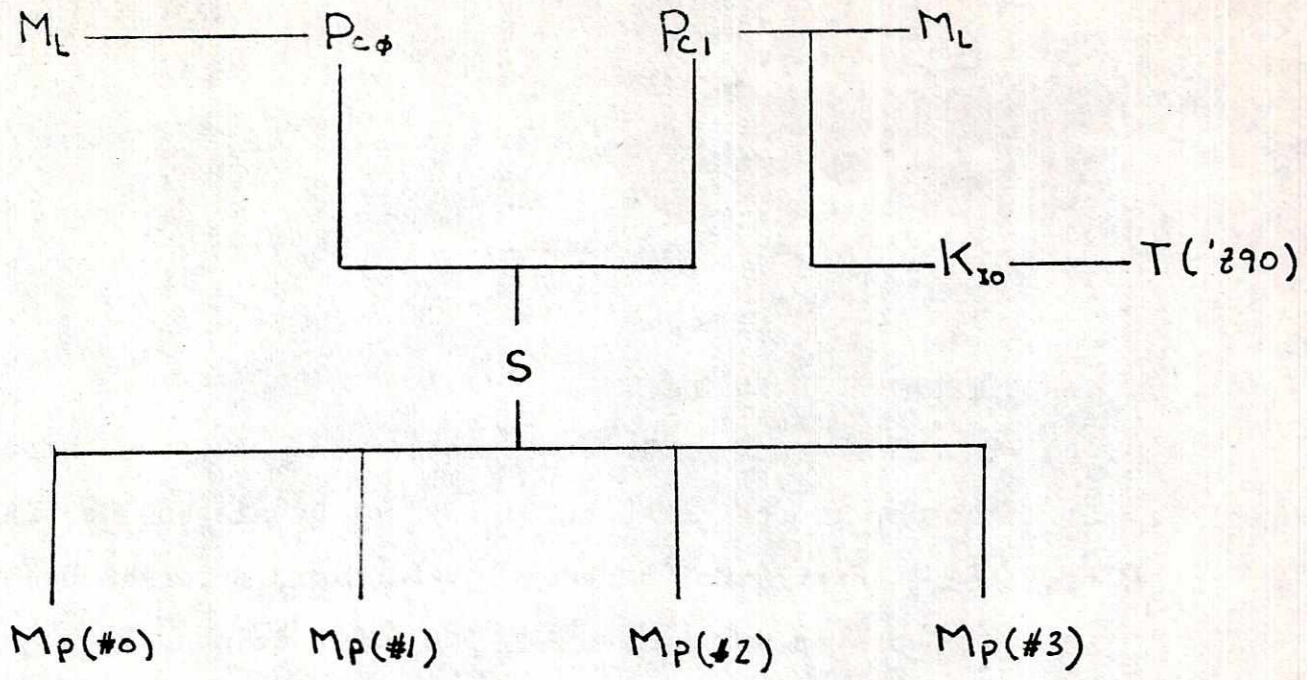


Fig. 5
Arquitectura General del Sistema (PMS)

B. LOGICA DE 'RESET' DEL SISTEMA

El propósito de esta lógica es llevar a los microprocesadores 8085 a un estado inicial conocido. Esto se debe a que al poner a funcionar el microprocesador no existe modo alguno de determinar el estado en que se encuentra.

La forma de inicializar el procesador es la de aplicar un cero lógico en el Pin 36 ($\overline{\text{Reset In}}$). Esto provoca que el registro PC de 8085 sea borrado y se inicie la ejecución de instrucciones a partir de la dirección cero, además las interrupciones son deshabilitadas y se genera una señal de salida (Reset Out) en el Pin 3, que se utiliza para dar 'reset' a las demás piezas del sistema.

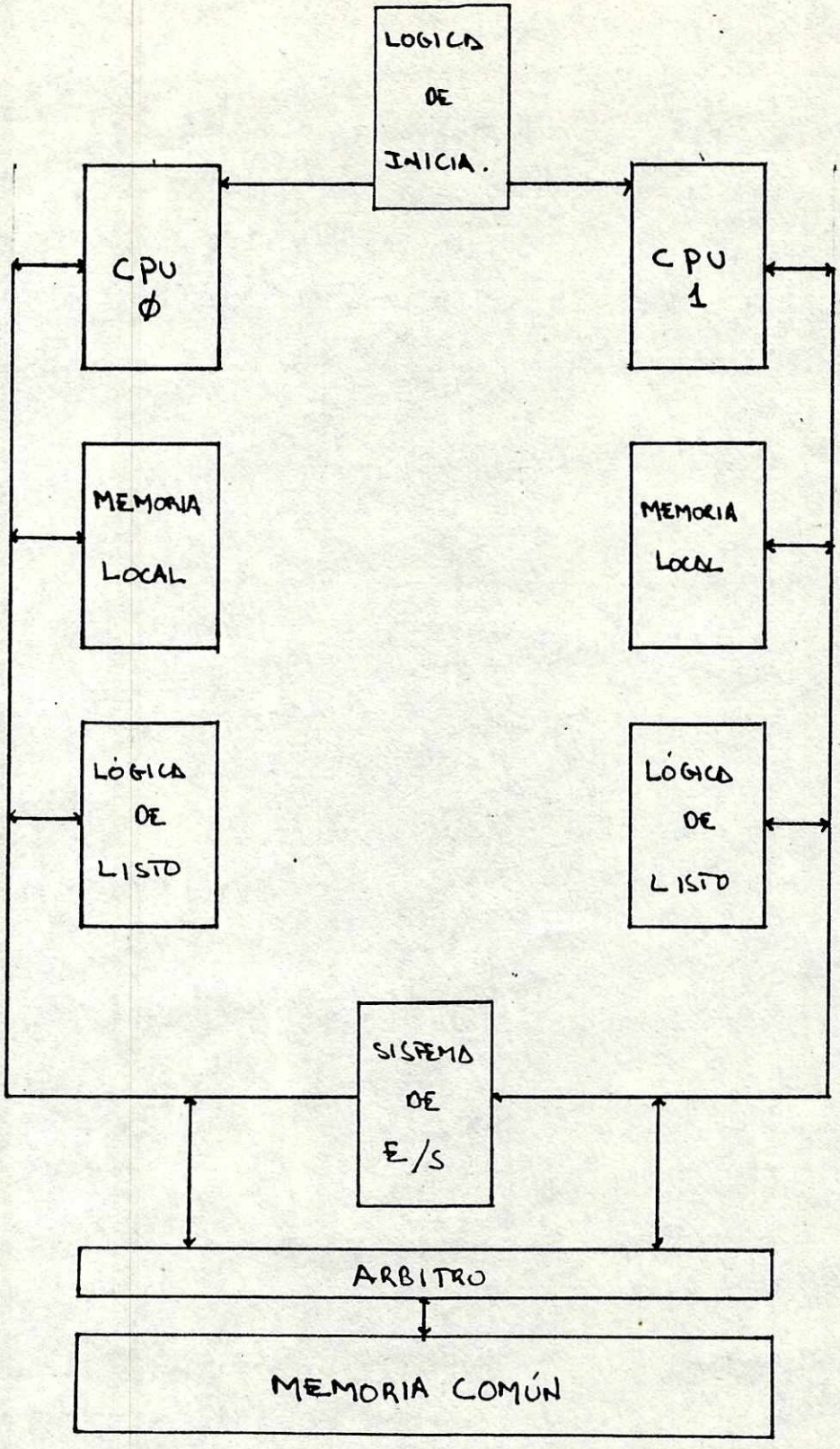


Fig. 6 Diagrama de bloques

Para generar el cero lógico en el Pin 36, se utiliza un interruptor que normalmente se halla en '1' y al ser accionado envía un '0'. Es interruptor se describe en la Fig. 7a. Se utiliza el mismo interruptor para dar 'reset' a los dos microprocesadores para que sea simultáneo. Fig. 7b.

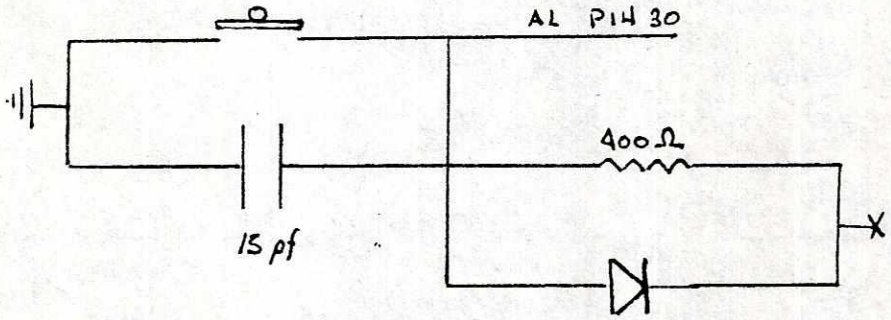


Fig. 7a. Interruptor

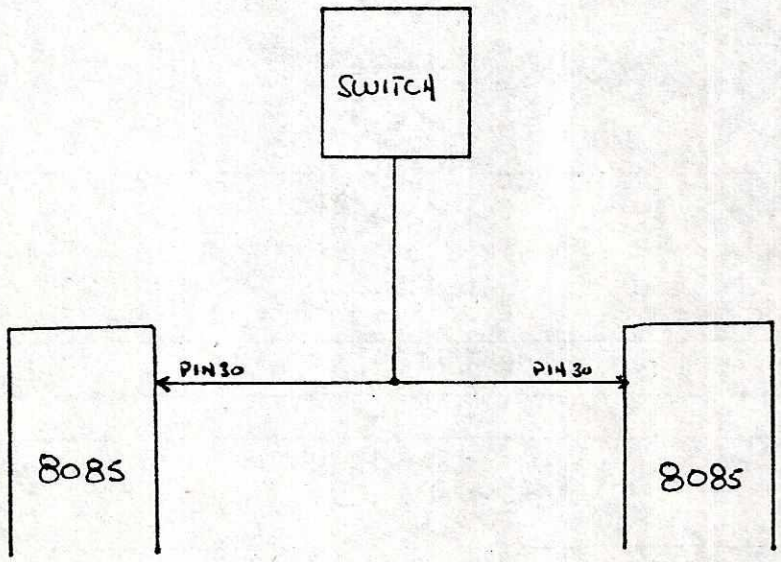


Fig. 7b Inicialización simultánea

C. ESTRUCTURA DE LA UNIDAD CENTRAL DE PROCESAMIENTO (CPU)

La descripción del CPU (Diagrama de conexión eléctrica se aprecia en la Fig. 8. Se construye alrededor del procesador INTEL 8085. Este procesador de ocho bits puede hacer referencia hasta 64K Bytes de memoria y 256 puertos de entrada/salida. Posee una lógica avanzada para el manejo de interrupciones y accesos directos a memoria (DMA).

Para construir el CPU se necesita, además del microprocesador, una lógica de reloj, un medio de 'demultiplexar' las líneas de dirección de las de información, unos 'drivers' para generar el 'bus' del sistema, una lógica de 'ready' para sincronizar las lecturas del microprocesador a memoria (esta, por ser sumamente especializada, se explicará en una sección aparte), por último, una fuente de enregía de +5 voltios.

1. Lógica de Reloj. El 8085 contiene el 'driver' para generar su reloj básico. La única pieza adicional que se necesita es un cristal para activarla. El cristal debe ser de una frecuencia máxima de 6.25 MHz y el que se usa es de 6.144MHz. Debido a que el 'driver' interno del 8085 divide entre dos la frecuencia del cristal para generar su ciclo básico, éste es de aproximadamente 325 nanosegundos.

Para instalar el cristal, se conecta directamente a los Pines 1 y 2 del 8085, Fig. 9.

2. Sistema de Demultiplexación. El 8085 multiplexa las ocho líneas de información con las ocho líneas menos significativas de dirección (AD0 - AD7, Pines 12 - 19).

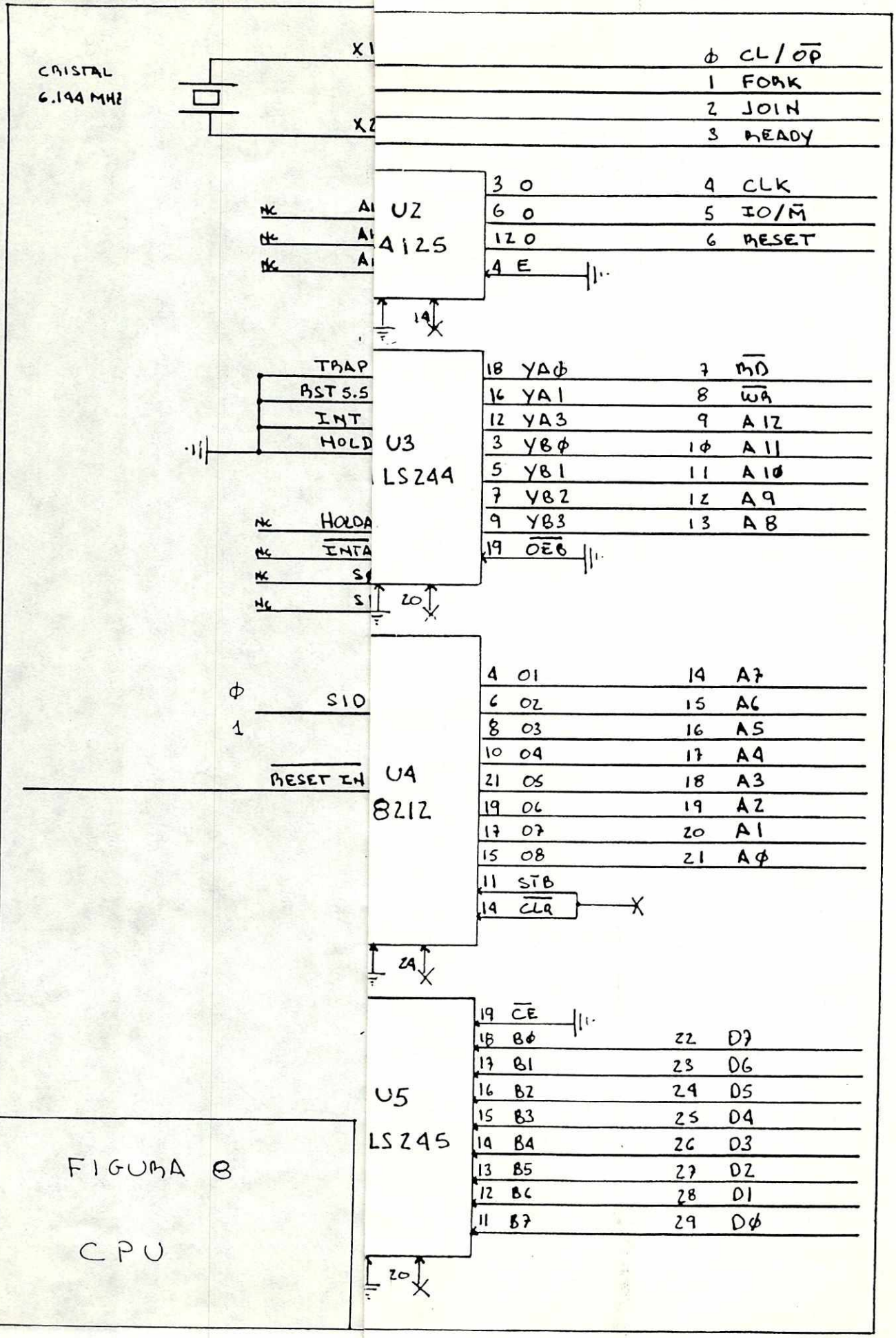

CRISTAL
6.144 MHz
X1
X2
U2
8125
3 O
6 O
12 O
4 E
14
Φ CL/OP
1 FORK
2 JOIN
3 READY
4 CLK
5 IO/M
6 RESET
TRAP
RST 5.5
INT
HOLD
U3
LS244
18 YAΦ
16 YA1
12 YA3
3 YBΦ
5 YB1
7 YB2
9 YB3
19 OEB
7 MD
8 WR
9 A12
10 A11
11 A10
12 A9
13 A8
20
Φ
1
S10
RESET IN
U4
8212
4 O1
6 O2
8 O3
10 O4
21 O5
19 O6
17 O7
15 O8
11 STB
14 CLR
14 A7
15 A6
16 A5
17 A4
18 A3
19 A2
20 A1
21 AΦ
24
U5
LS245
19 CE
18 BΦ
17 B1
16 B2
15 B3
14 B4
13 B5
12 B6
11 B7
22 D7
23 D6
24 D5
25 D4
26 D3
27 D2
28 D1
29 DΦ
20

FIGURA 8

CPU

Para distinguir cuando hay datos y cuando direcciones existe una línea adicional llamada ALE (Pin 30). Si ALE se encuentra en "1", indica que en los Pines 12 al 19 está la parte baja de la dirección; en caso de estar en "0" se asume que debe haber datos.

El método utilizado es el de guardar en una pieza adicional la parte baja de la dirección cuando el ALE se encuentre activo. La pieza a utilizar es un 8212 (U4 en Fig. 8). Esta pieza tiene ocho líneas por las cuales recibe información y ocho líneas por las que saca información guardada en su interior, o sea el mismo número de líneas que hay multiplexadas en el 8085.

Cuando el Pin 13 del 8212 está activo, la información que se encuentra en las líneas de entrada pasa a un registro interior y activando las señales MD, $\overline{CLR}$, STB y desactivando DS1, pasa el registro a las líneas de salida, demultiplexando así la dirección (ver Fig. 10). Cuando el Pin 13 del 8212 está bajo (ALE bajo también) se ignoran las líneas de entrada.

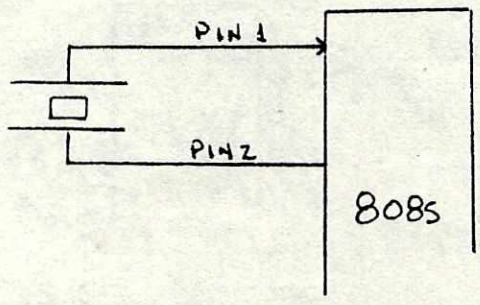


Fig. 9 Conexión de Cristal

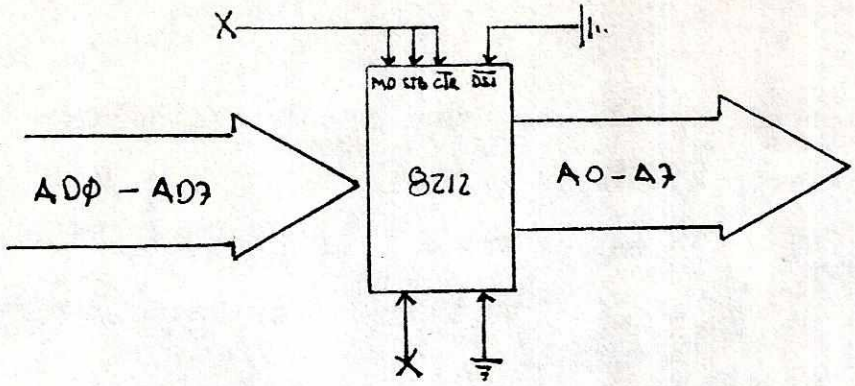


Fig. 10 Conexión del 8212

3. Conexión de los Drivers. Debido a que el número de circuitos integrados que utiliza el modelo es muy grande se necesita instalar unos drivers para las líneas más usadas. Esto evitará problemas de sobrecarga en el procesador (dispersión)

Los Drivers usados son cuatro. El 8212 (U3 en Fig. 8) usado para demultiplexar es un driver de por sí. Para las líneas AD0-AD7 que quedan 'libres' como bus de datos se utiliza un circuito integrado 74LS245 (U5 en Fig. 8). La forma en que se conectó se aprecia en la Fig. 11.

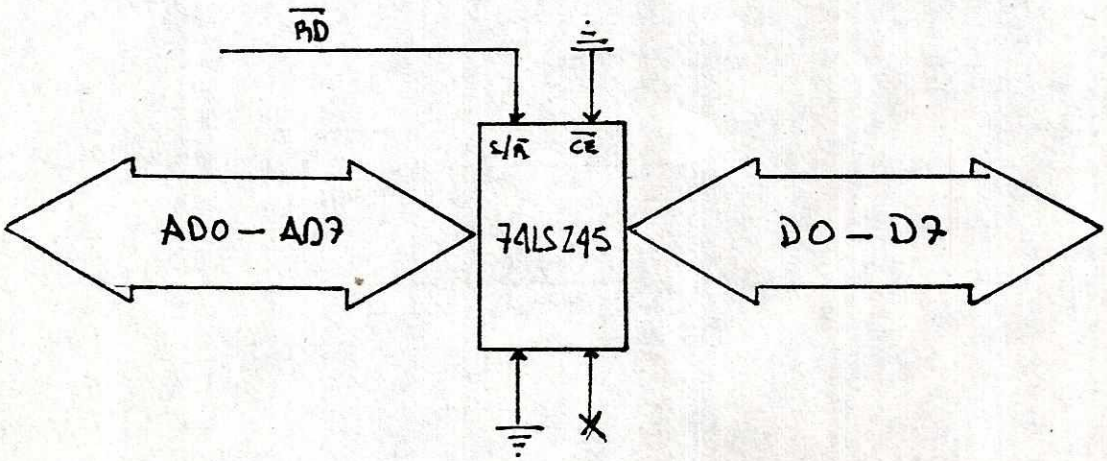


Fig. 11 Conexión del 74LS245

Para las líneas A8-A12 (Pines 21-25), $\overline{WR}$ (Pin 31), $\overline{RD}$ (Pin 32) se utiliza un 74LS244 pues no son líneas bidireccionales. En la Fig. 8, es la U3 y su esquema de conexión se encuentra en la Fig. 12.

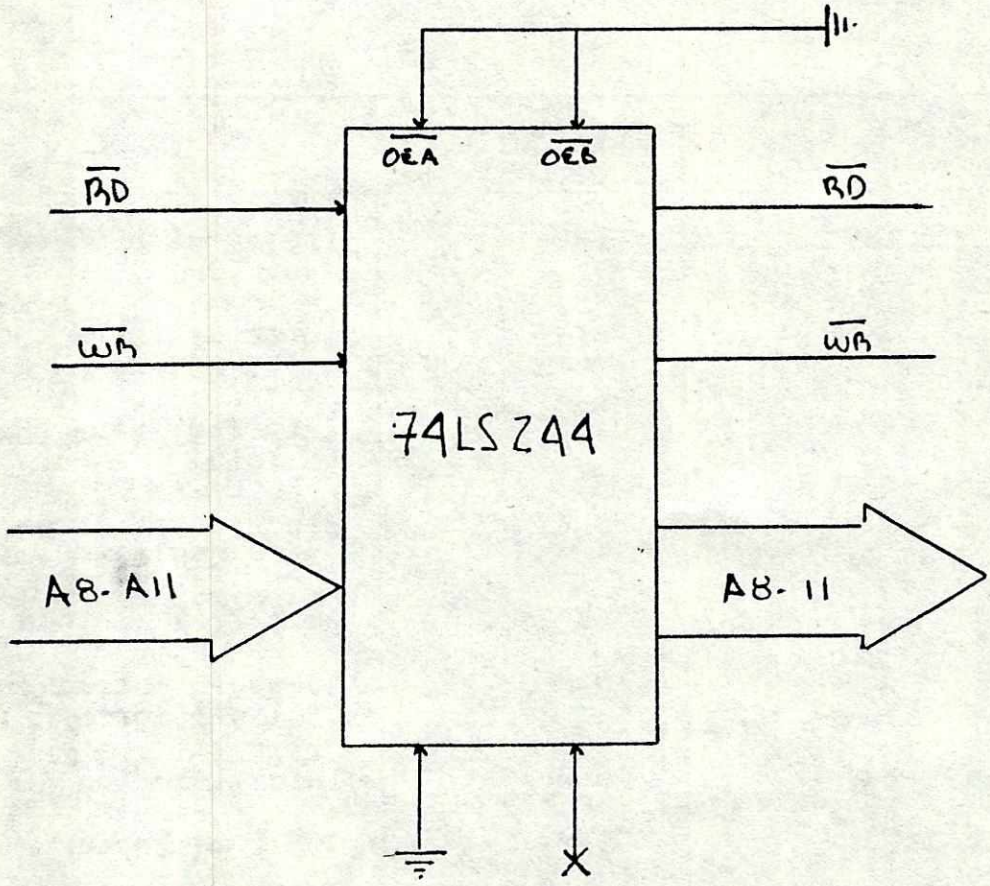


Fig. 12 Conexión del 74LS244

4. Señales Generadas por el CPU. El microprocesador 8085 genera un cierto número de señales para comunicarse con los demás componentes. Estos generan a su vez otras señales. El conjunto de todas ellas forman el bus del sistema. El número de señales generadas por el CPU son 30. El significado de las señales se detallan en la tabla 1.

Número de Señal	Mnemónico	USO
0	CL/$\overline{OP}$	Instrucción Close (CL) y Open ($\overline{OP}$)
1	Fork	Interrupt para ejecutar un Fork
2	Join	Interrupt para ejecutar un Join
3	Ready	Sincronización de memoria y CPU
4	CLK	Reloj del sistema
5	IO/$\overline{M}$	Acceso a entrada/salida (IO) o memoria ($\overline{M}$)
6	Reset Out	Inicialización
7	$\overline{RD}$	Señal de Lectura
8	$\overline{WR}$	Señal de Escritura
9-21	A12-A0	Bus de Direcciones
22-29	D7-D0	Bus de Datos

Tabla 1 Señales Generadas por el CPU

Para diferenciar las señales de los procesadores, las provenientes del 1 serán primadas, las del 0 no.

5. Notas sobre las Señales del 8085. Debido a que los medios de entrada/salida son muy limitados, no se usa la lógica de interrupciones, dejando las señales INT, $\overline{\text{INTA}}$, RST 5.5 y TRAP sin uso. Las líneas RST 7.5 y RST 6.5 tienen un uso especial para comunicación entre procesadores.

Las líneas SO y S1 se dejan 'libres'. No tienen mayor uso, pues sólo revelan el estado del 8085.

Como no existe la necesidad de accesos directos a memoria (DMA) no se utilizan las señales de Hold y Holda.

Como el sistema tiene como máximo 4K Bytes y existen dos tipos diferentes de memoria, se requieren sólo 13 bits de dirección (A0-A12) A13, A14 y A15 se dejan 'libres'.

Por último, SID se utiliza para identificar el número del procesador; si SID está inactivo, las señales son del procesador 1; si no son del 2.

D. MEMORIA LOCAL

La memoria local de cada procesador, es de 2K Bytes, divididos en 1K ROM y 1K RAM. El método para distinguir cuando se está usando la memoria local se halla en la Fig 14.

Si IO/$\overline{\text{M}}$ y A12 están en "0", se está haciendo uso de la memoria local, y con $\overline{\text{RD}}$ y $\overline{\text{WR}}$ aseguramos una activación válida.

Para distinguir entre ROM y RAM, se sigue la lógica de la Fig. 15.

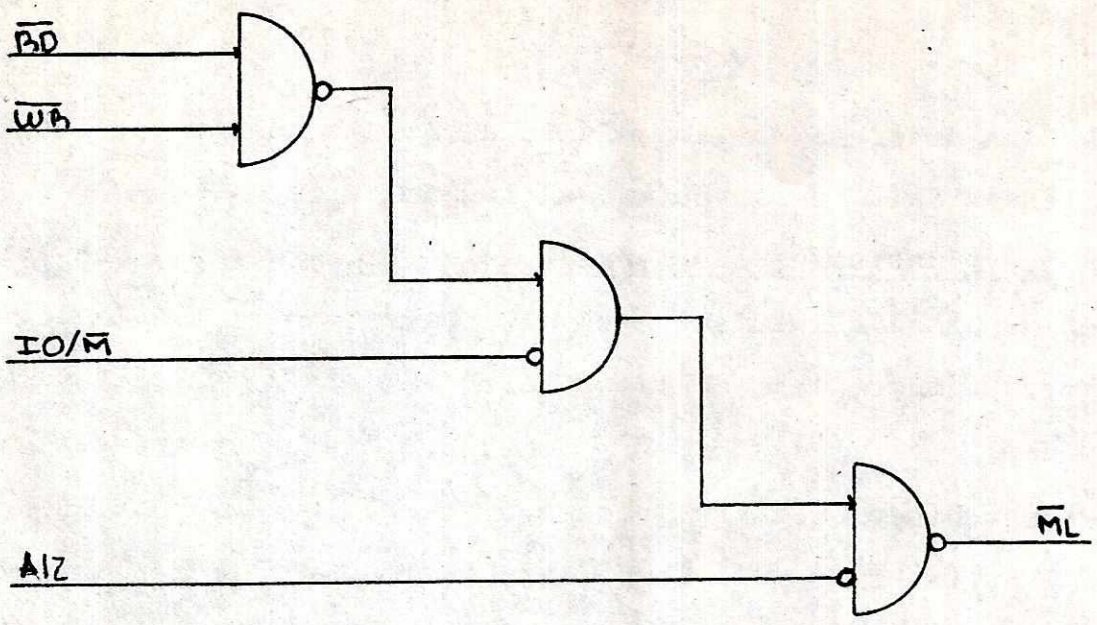


Fig. 14 Selección de Memoria Local

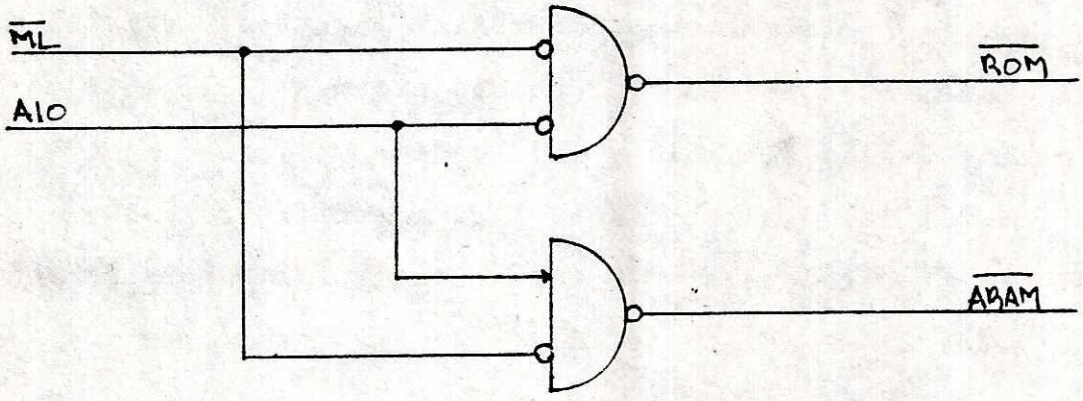


Fig. 15 Distinción entre ROM y RAM

Si se está usando la memoria local y A10 está en "0", se activa el ROM. Si A10 está en "1", se activa el RAM.

Con el ROM existe el problema de que sólo debe ser leído. Para solucionar este problema se usa la lógica de la Fig. 17.

Así, sólo cuando se esté leyendo ($\overline{RD}$ = 0 y $\overline{WR}$ = 1) se podrá activar el ROM.

El bloque completo de memoria local se describe en la Fig. 16.

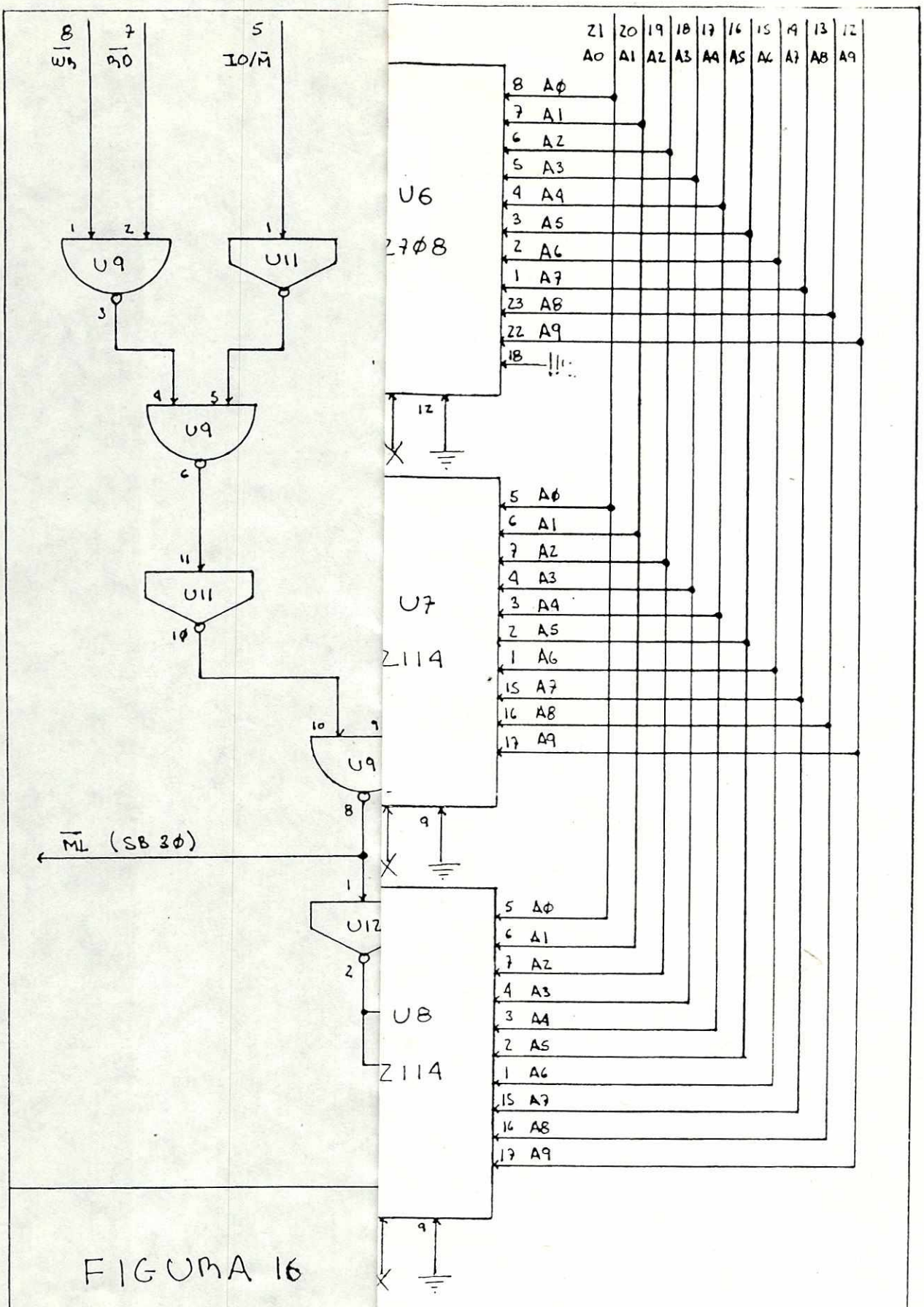


FIGURA 16

MEMORIA LOCAL

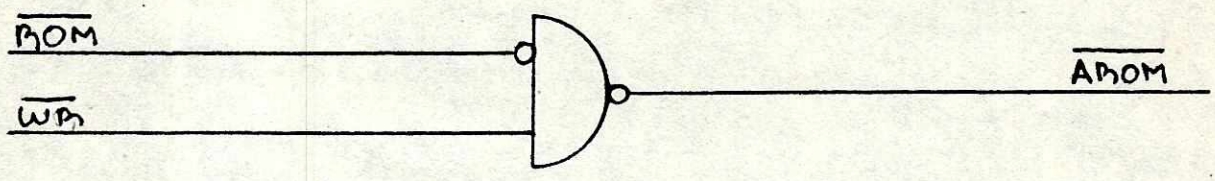


Fig. 17 Acceso a ROM

Es necesario aclarar que toda la lógica se implementa usando nada más que compuertas NAND (circuitos integrados 7400), y negadores (Tipo 7404).

1. Implementación del ROM. Para hacer 1K Byte de ROM se usa el circuito integrado 2708 (U6 Fig. 17), cuya organización es de 1K x 8 Bits. Este es del tipo EPROM y requiere tres fuentes distintas de voltaje +5, +12 y -5 voltios.

La forma en que se conecta al bus del sistema se presenta en la Fig. 18.

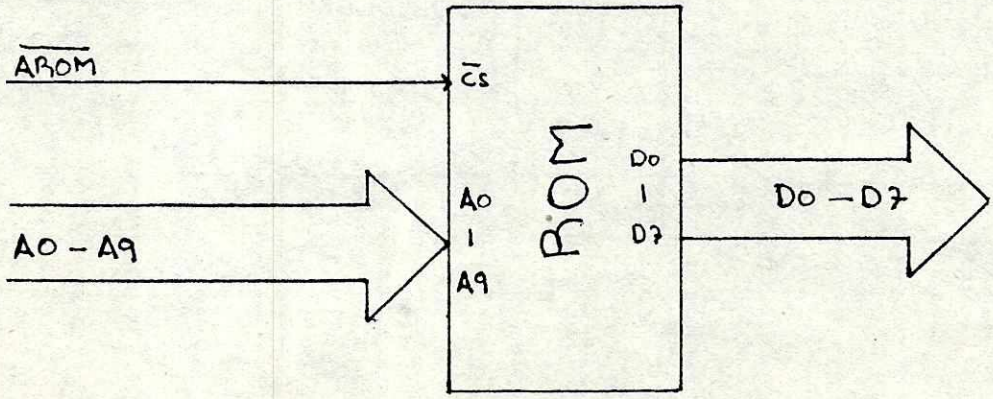


Fig. 18 Conexión del ROM

Las diez señales de dirección del 2708 se conectan directamente a las líneas del bus del sistema AO - A9. La línea $\overline{CS}$ de activación del ROM se conecta a $\overline{AROM}$ proveniente de la lógica de acceso a ROM (Fig. 16). Por último, las ocho salidas de datos se dirigen a DO - D7 del bus del sistema.

2. Implementación de RAM. Para crear 1K Byte de RAM, se utilizan dos circuitos integrados 2114. La organización de éstos es de 1K x 4 Bits, razón por la cual se requieren dos.

En la Fig. 17 se puede identificar los circuitos como U7 y U8. U7 es el Nibble alto y U8, el bajo. El diagrama de bloques de la conexión se describe en la Fig. 19.

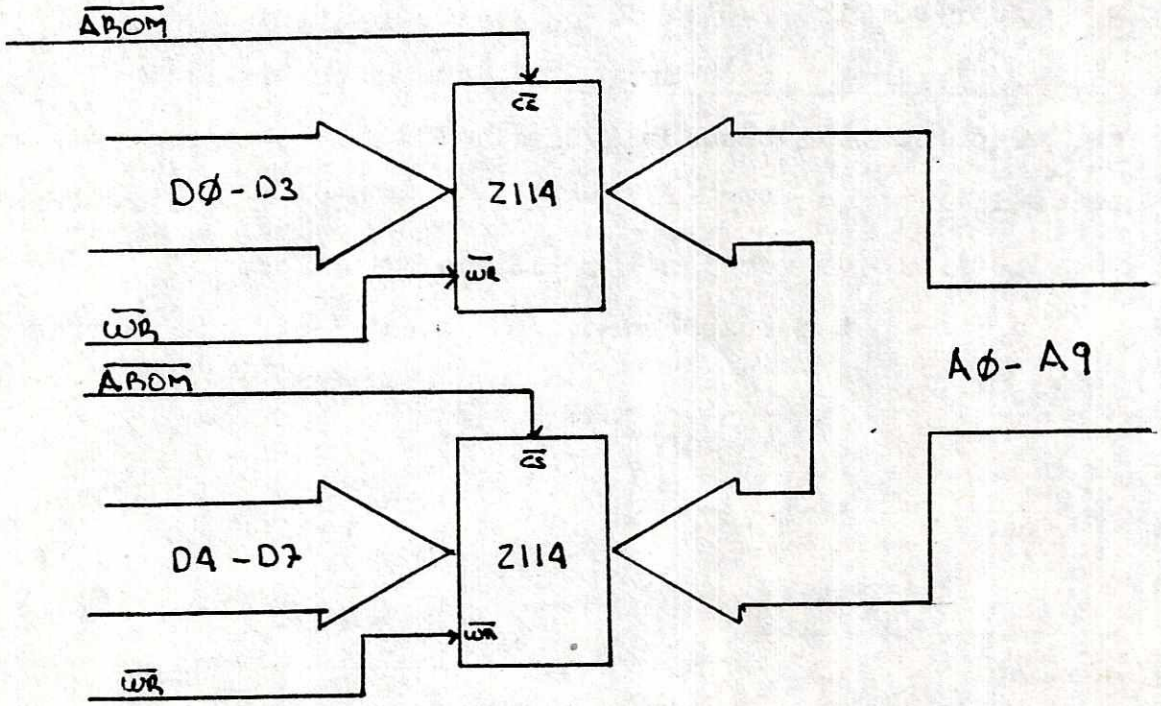


Fig. 19 Conexión de RAM

Cada 2114 requiere un solo voltaje (+5 voltios). Tiene por entrada A0 - A9 del bus del sistema y como respuesta 4 Bits de datos U7 conecta éstos Bits a D7 - D4 y U8 a D3- D0 formando así un Byte completo. La línea de selección viene de $\overline{ARAM}$ de la lógica de distinción de ROM y RAM (Fig. 15). Por último, como la memoria es RAM, necesitamos saber si se escribe o lee y para esto se utiliza la línea $\overline{WR}$ del bus del sistema, que se conecta a la línea $\overline{WR}$ del 2114.

E. SISTEMA DE ENTRADA/SALIDA (E/S)

Dentro del diseño se pueden distinguir dos tipos de unidades de E/S las locales y las exteriores.

Las unidades locales son los dos microprocesadores; cada uno se comunica con el otro como una unidad de E/S.

Las exteriores son por ahora 1. Se comunican al exterior por medio de un interface RS-232C a una computadora Z90 de Zenith.

El diagrama completo del sistema de E/S se detalla en la Fig. 21.

1. Acceso al Sistema de E/S. La lógica de habilitación de entrada y salida se describe en la Fig. 20.

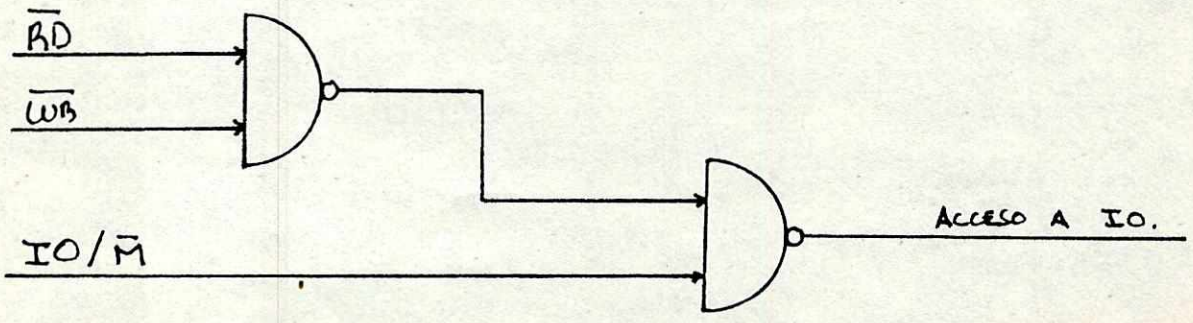


Fig. 20 Habilitación de E/S

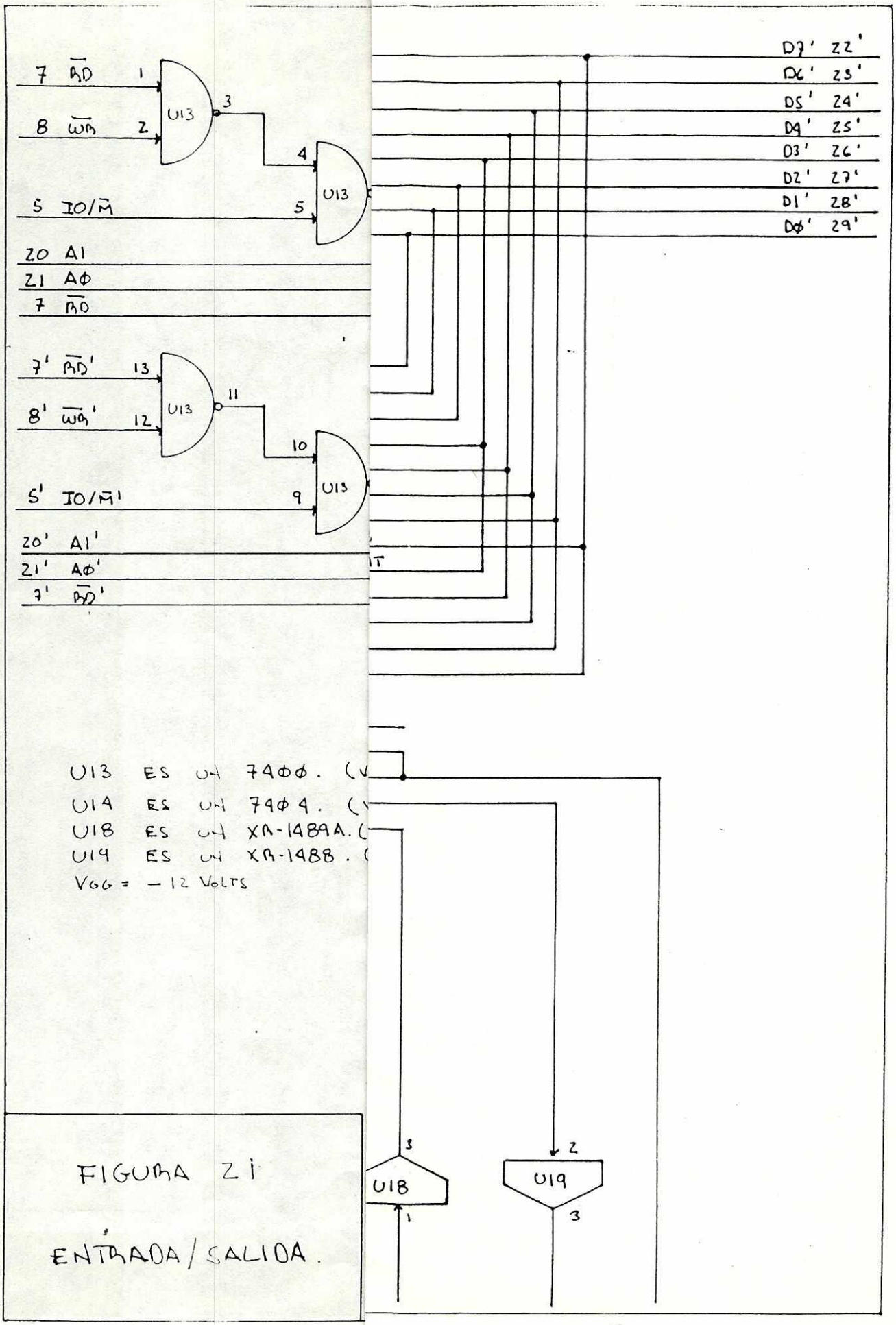


FIGURA 21

ENTRADA / SALIDA.

Se requieren las mismas condiciones que para habilitar la memoria, sólo que la señal IO/$\overline{M}$ debe estar en "1".

2. Acceso a Unidades Locales. Cada microprocesador se refiere al otro como la unidad E/S cero. Al haber un IN Ø acciona la línea de 'join' (RST 6.5), creando una interrupción al otro microprocesador. Una instrucción OUT Ø activa la señal Fork (RST 7.5) que genera otra interrupción.

Para hacer esto se utiliza un decodificador 8-1 tipo 74LS138 (U15 y U16 en Fig. 21), el cual también se utilizará en el procesador 1 para activar la unidad externa (el procesador 1 tiene el control sobre ella).

La lógica para utilizar el 74LS138 se presenta en la Fig. 22.

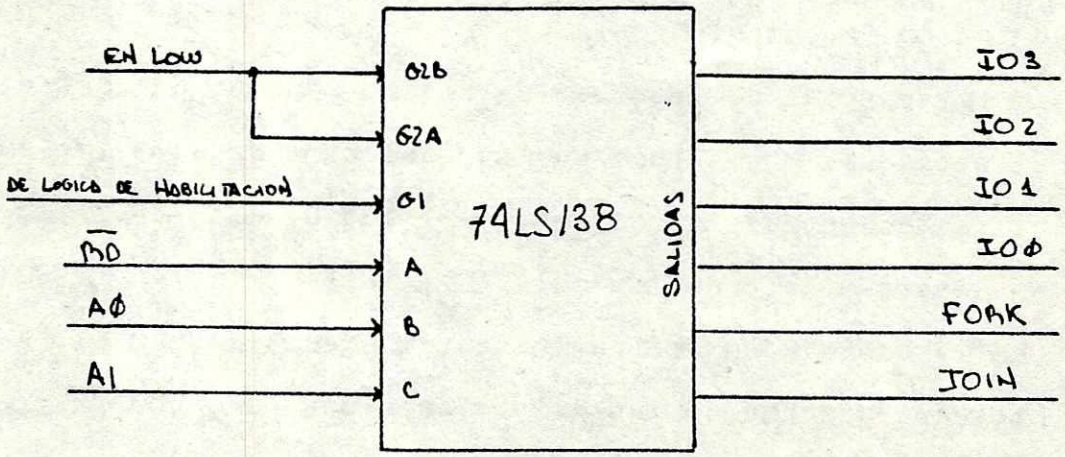


Fig. 22 Uso del 74LS138

Como se aprecia en la Fig. 22, se utiliza la señal de $\overline{RD}$ para decodificar las direcciones como de E/S haciendo la lógica más sencilla.

3. Acceso a la Unidad Exterior. El CPU 1 utiliza el 74LS138 para activar un UART de interface a la microcomputadora Z90.

El UART (U17 en Fig. 21) ocupa cuatro direcciones de E/S: dos de lectura y dos de escritura. Debido a esto se utilizan las direcciones 1 y 2 en conjunto con la señal $\overline{RD}$ para activarlo.

El UART utilizado para comunicación al exterior se puede considerar como un microprocesador de E/S. El UART recibe del exterior una línea para sincronizar las velocidades de recepción y transmisión (es un reloj con frecuencia 16 x Baund utilizado). Otra de sus características, es la arquitectura interior que permite la selección de diferentes modalidades de transmisión. Como se dijo antes, se tiene cuatro diferentes direcciones: una para lectura, otra para escritura, una de control y otra de estado UART. Tiene la ventaja de poder escribir y leer al mismo tiempo (2 unidades independientes).

Nota especial de la conexión de las señales de control y del estado es que éstas, por ser del tipo de alta impedancia, se hallan conectadas directamente al bus de datos. En la Fig. 23 se describe la conexión del UART.

En la tabla 2 se define la acción ejecutada por el UART al activar cada una de las direcciones con $\overline{RD}$ o con $\overline{WR}$ y en la Fig. 24 está el significado de los Bytes de control y estado.

Ya que el UART no es compatible en la E/S de señales con el interface RS-232 C, se necesitan dos 'drivers' (U18 y U19 Fig. 21).

para modificarlas. El XR-1489 se usa para entrada y el XR-1488 se usa para salida. El diagrama de éstos se encuentra en la Fig. 25.

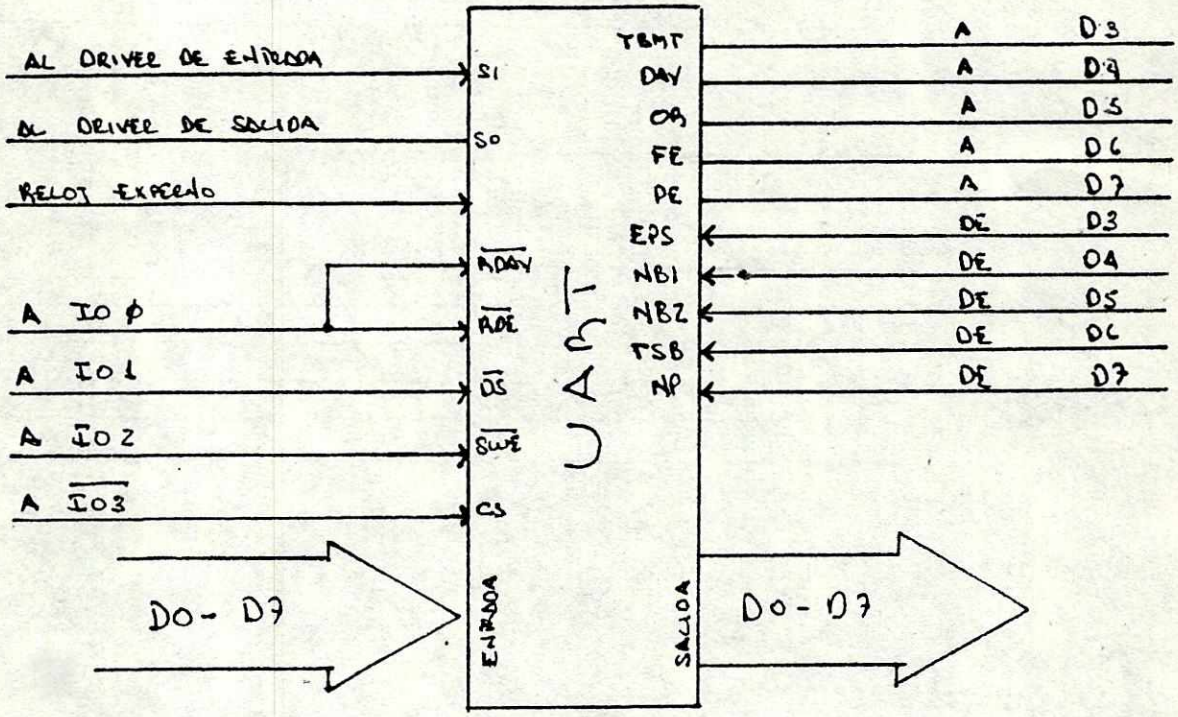


Fig. 23 Conexión del UART

Dirección	E/S	Acción
1	$\overline{RD}$	Leer Datos
1	$\overline{WR}$	Escribir Datos
2	$\overline{RD}$	Leer Estado
2	$\overline{WR}$	Escribir Control

Tabla 2 Direcciones de E/S

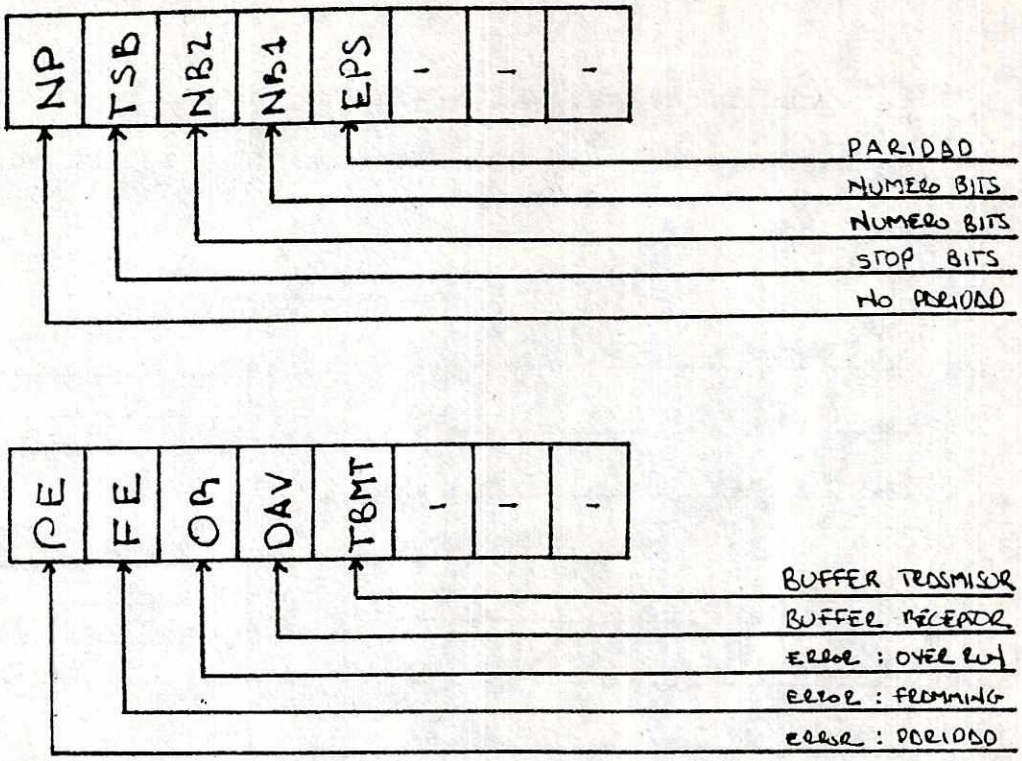


Fig. 24 Bytes de Control y Estado

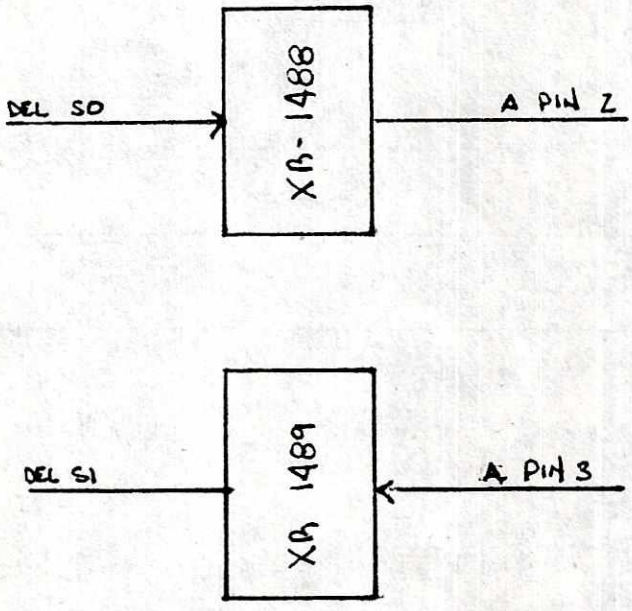


Fig. 25 Interface RS-232C

4. Programación de E/S. Las instrucciones para dar 'reset' al UART de comunicación son éstas:

```
MVI  A, Control
Out  2
```

Donde control es un Byte con el formato de la Fig. 23. El significado de cada Bit se detalla en la tabla 3.

BIT	MNEMONICO	SIGNIFICADO
0	-	Ninguno
1	-	Ninguno
2	-	Ninguno
3	EPS	Activa el bit de paridad
4	NB1	Número de bits a transmitir
5	NB2	Número de bits a transmitir
6	TSB	Número de bits de "stop"
7	NP	Desactiva la paridad

Tabla 3 Byte de control

El significado de cada Bit en el Byte de estados se detalla en la tabla 4.

BIT	MNEMONICO	SIGNIFICADO
0	-	Ninguno
1	-	Ninguno
2	-	Ninguno
3	TBMT	Transmisor vacío
4	DAV	Receptor lleno
5	OR	Error de sobreposición
6	FE	Error de bit de inicio
7	PE	Error de paridad

Tabla 4 Byte de Estado

Viendo el significado de los Bits de estado, es fácil construir las rutinas de E/S. La de lectura de un caracter es como sigue:

```
Read:  IN  2        ;
       ANI 20Q      ;el sufijo Q representa octal
       JPZ Read     ;verificar si hay caracter
       IN  1        ;entra carácter
       RET
```

Y la de escritura es esta:

```
Write: IN  2
       ANI 10Q
```

```
JPZ Write
MOV A, M
OUT 1
```

La verificación de los distintos tipos de error es similar a estas rutinas.

F. ARBITRO DE MEMORIA

La ventaja de la memoria entrelazada es la facilidad con que se puede construir un medio para permitir accesos concurrentes a sus distintos bancos.

El árbitro de memoria no es más que un sistema para resolver los conflictos ocasionados por la competencia de varios procesadores por un mismo banco.

La construcción del árbitro es conceptualmente sencilla. Lo único que tiene que hacer es verificar si los dos procesadores quieren usar el mismo banco; si no es así, se le da acceso a los dos procesadores a la memoria. En otro caso, sólo uno de los dos recibe la orden de acceso. Para esto se utiliza la lógica de la Fig. 26 que en realidad no es más que un XOR aplicado a las líneas AO, Al, AO' y Al'. La función del Flip-Flop es la de cerrar el sistema para el procesador de menos prioridad (procesador 1), evitando que un procesador pase sobre otro.

Existe otra lógica adicional en este bloque. La línea de CL/$\overline{OP}$ de cada procesador es analizada para inhibir los accesos de uno u otro procesador a la memoria. Esto permite la creación de regiones críticas y se puede considerar como la imple-

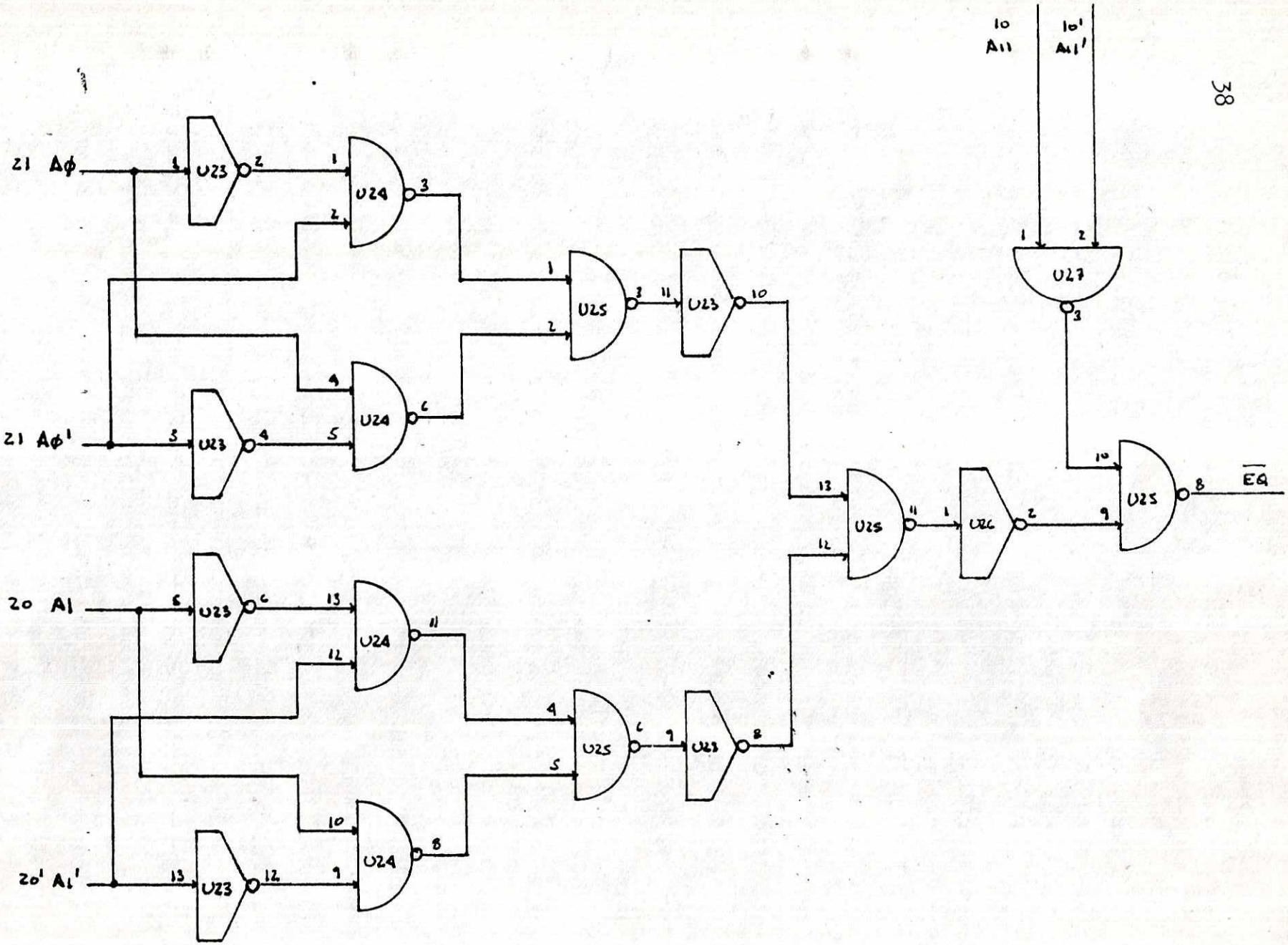


Fig. 26a. Detector de Igualdad

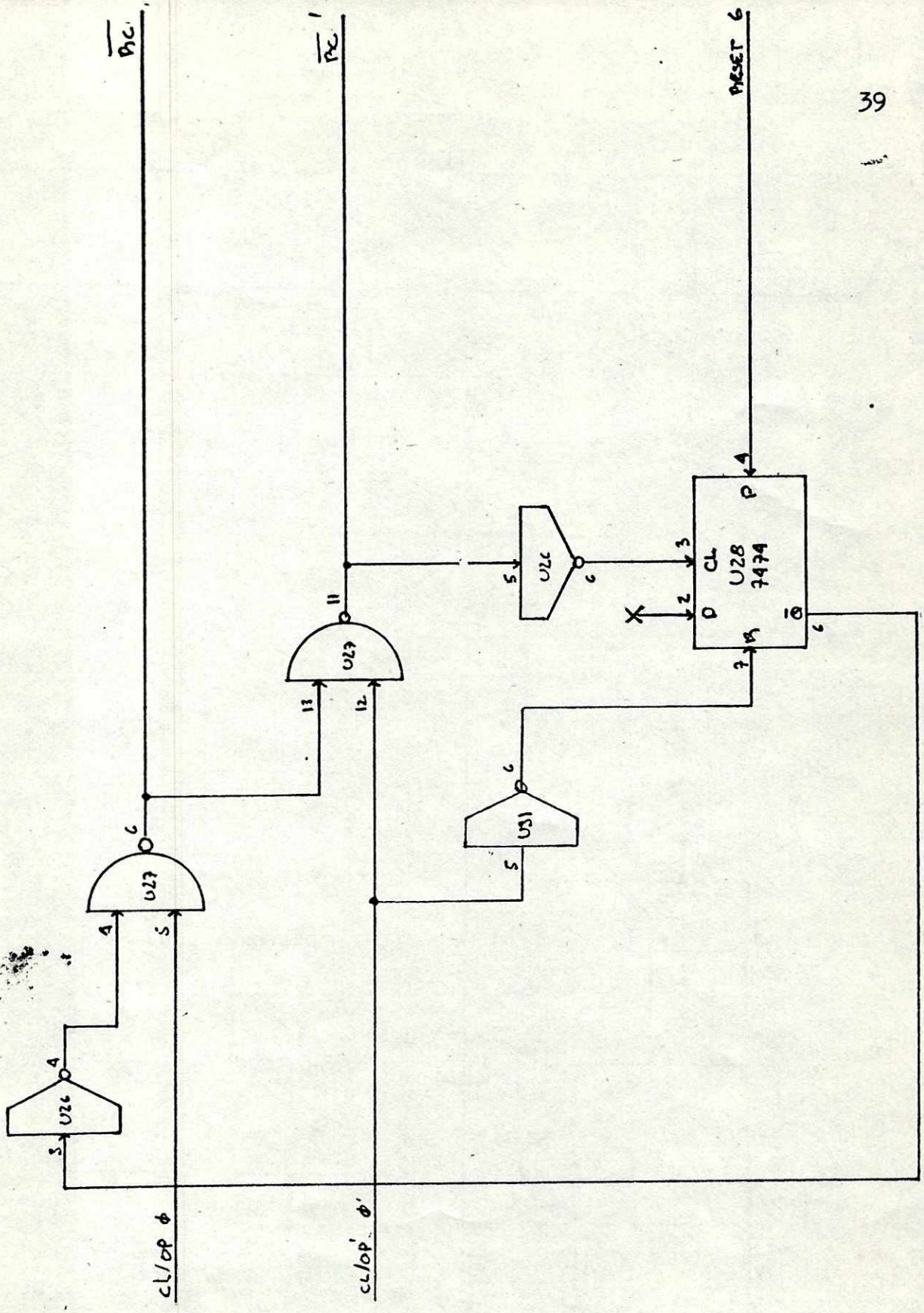


Fig. 26b. Control de Región Crítica

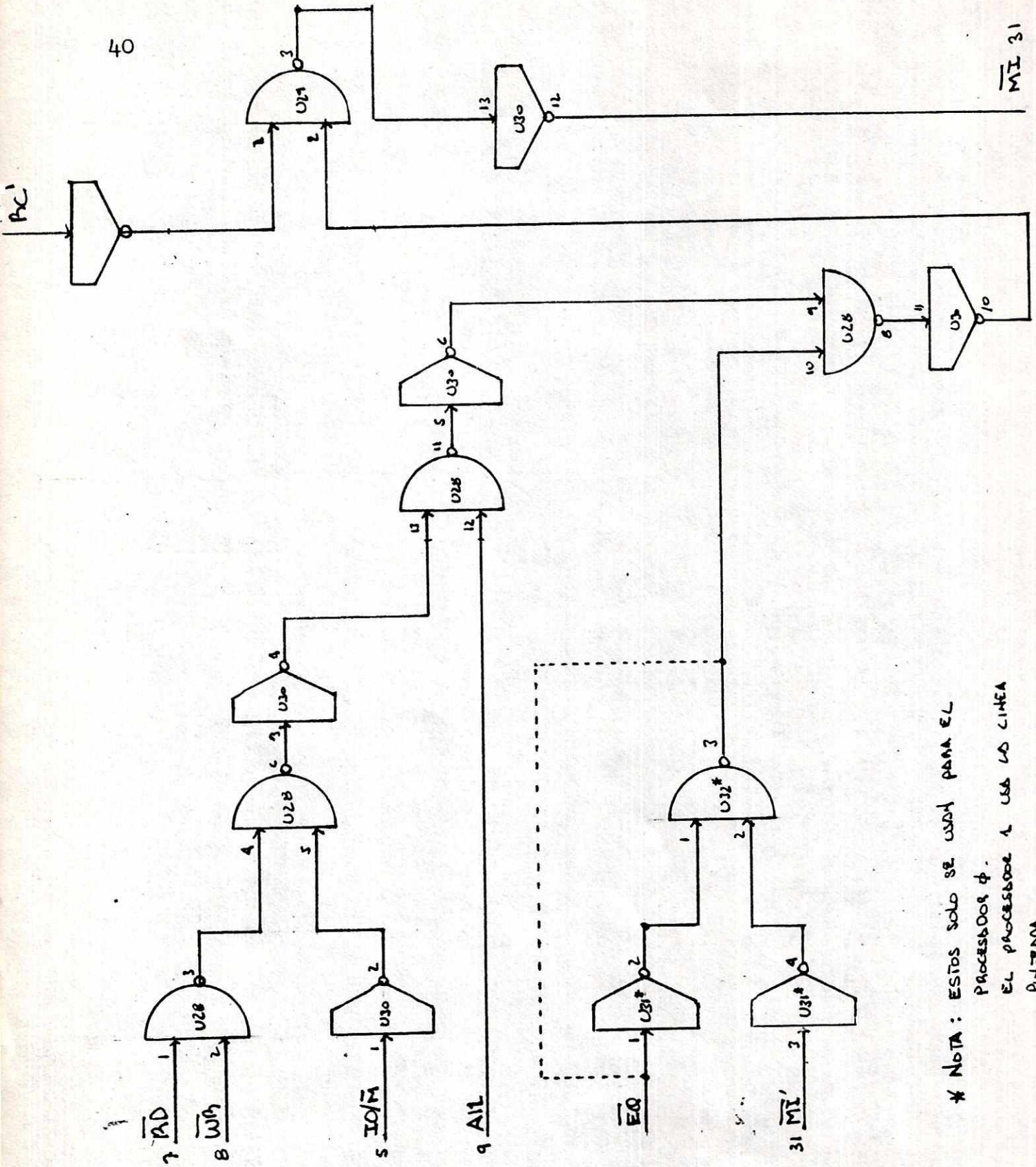


Fig. 26c. Controlador de Acceso a Memoria Común

mentación en Hardware de las primitivas P (Mutex) y V (Mutex).

Es necesario recordar que toda la lógica se construye utilizando NANDS y negadores, por lo que aparentemente el árbitro se ve complejo. Si se utilizaran compuertas XOR y OR además de las otras, el número de piezas disminuiría y las conexiones serían más sencillas.

El árbitro de memoria contribuye con dos líneas al bus del sistema: $\overline{MI}$ y $\overline{MI}$' que indican cuando un procesador tiene acceso a la memoria entrelazada.

G. MEMORIA ENTRELAZADA

La memoria está construida de tal manera que permite a los dos procesadores usarla al mismo tiempo. Está dividida en dos secciones, el sistema de activación y la memoria en sí. El detalle de la memoria se encuentra en la Fig. 27.

1. Sistema de Activación. Esta sección es la encargada de activar uno de los cuatro bancos para cada procesador. Para ello, se utiliza la mitad de un decodificador 74LS139 para cada uno. El decodificador es de 4-1 dual y su activación proviene del árbitro de memoria a través de las líneas $\overline{MI}$ y $\overline{MI}$' (línea del bus del sistema 31). Su diagrama de conexión es la Fig. 28.

Cada línea de salida del decodificador va a un banco de memoria.

2. Memoria. La organización de la memoria es sencilla, se utilizan circuitos integrados tipo 2114 (1K x 4 Bits) al igual que en la memoria local. En consecuencia la conexión es similar.

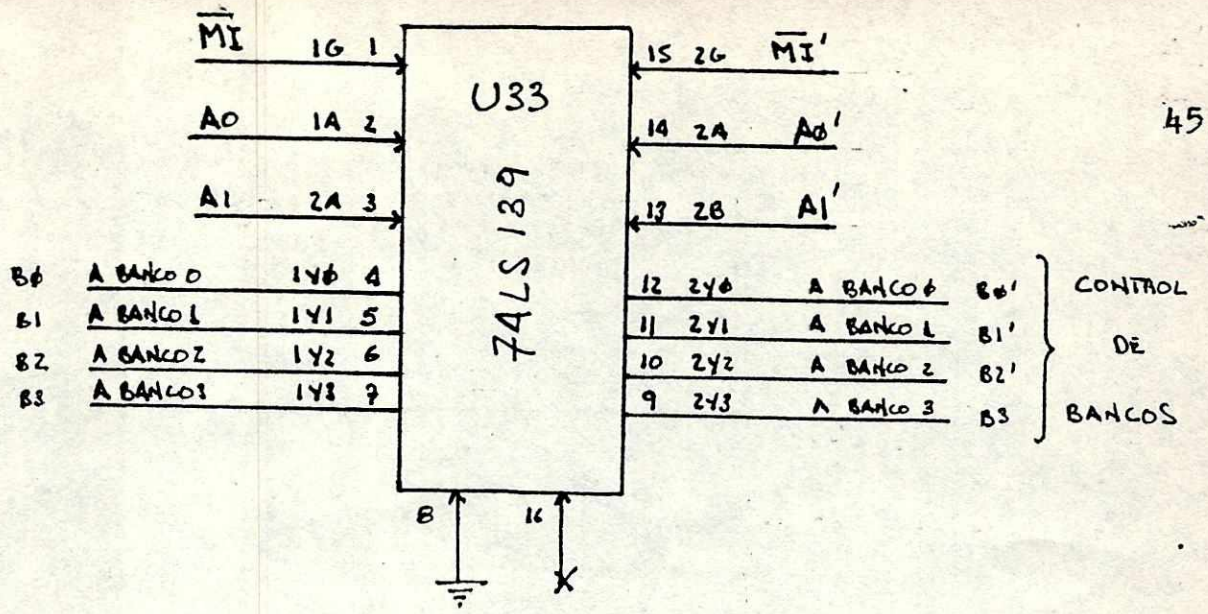


Fig. 28 Conexión del 74LS139

Lo que hace distinta la conexión de la memoria entrelazada de la local es que la entrelazada recibe y da su información a dos grupos de drivers. Cada grupo se encuentra conectado al bus del sistema de un procesador, transmitiendo la información al bus del driver que se encuentra activado en ese momento, creando así dos vías de transmisión para cada banco. En ningún momento se abren las dos vías, pues el sistema de árbitro de memoria no lo permite.

El diagrama de conexión de los bancos es la Fig. 29. La única diferencia es la línea de activación que llega a cada uno.

El total de memoria que se tiene es de 4K Bytes (1K por banco) quedando en un banco las direcciones 0, 4, 8..., en otro 1, 5, 9..., en el siguiente 2, 6, 10... y en el último 3, 7, 11... para completar todas las direcciones posibles.

H. LOGICA DE 'READY'

La lógica de listo se divide en dos partes, la primera es la

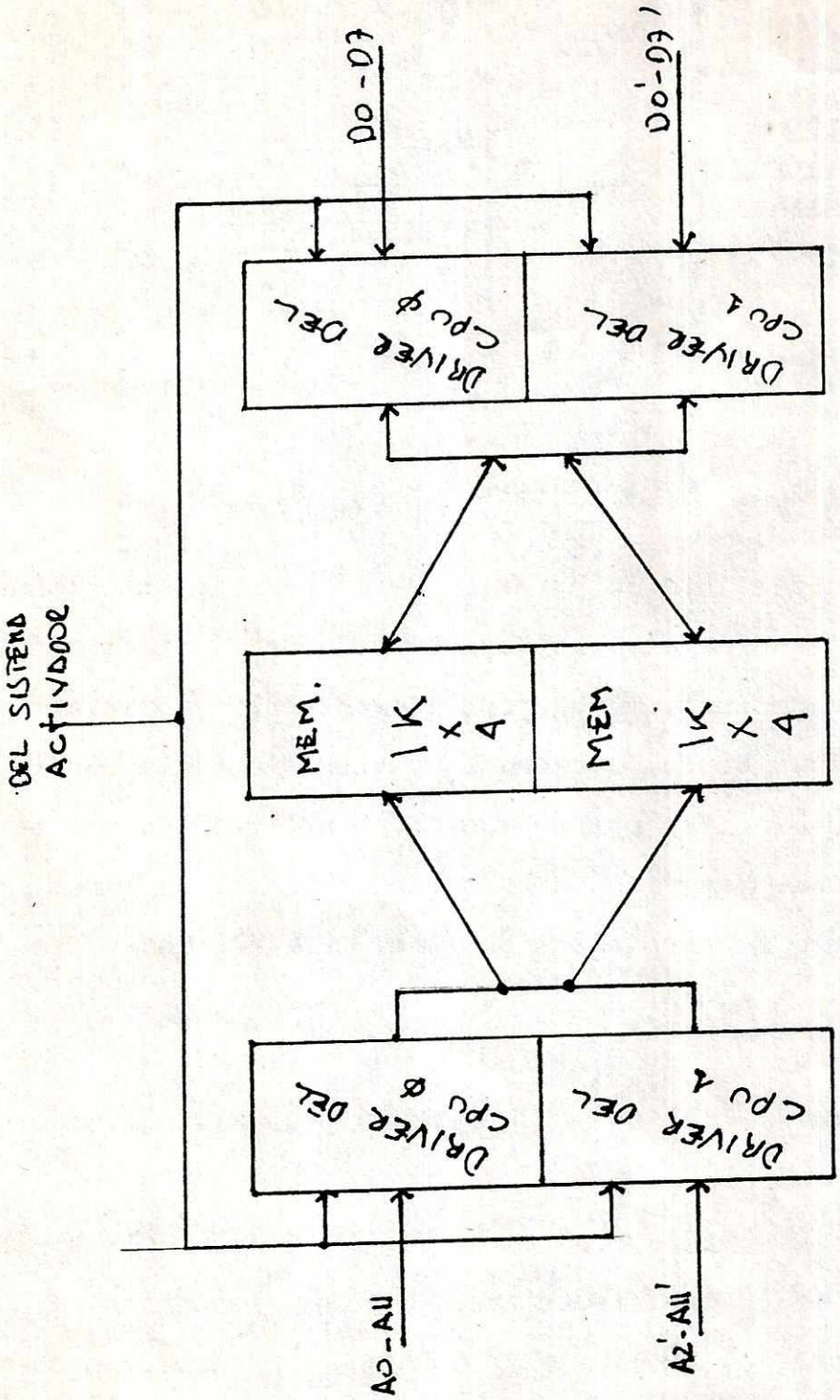


Fig. 29 Conexión de los Bancos

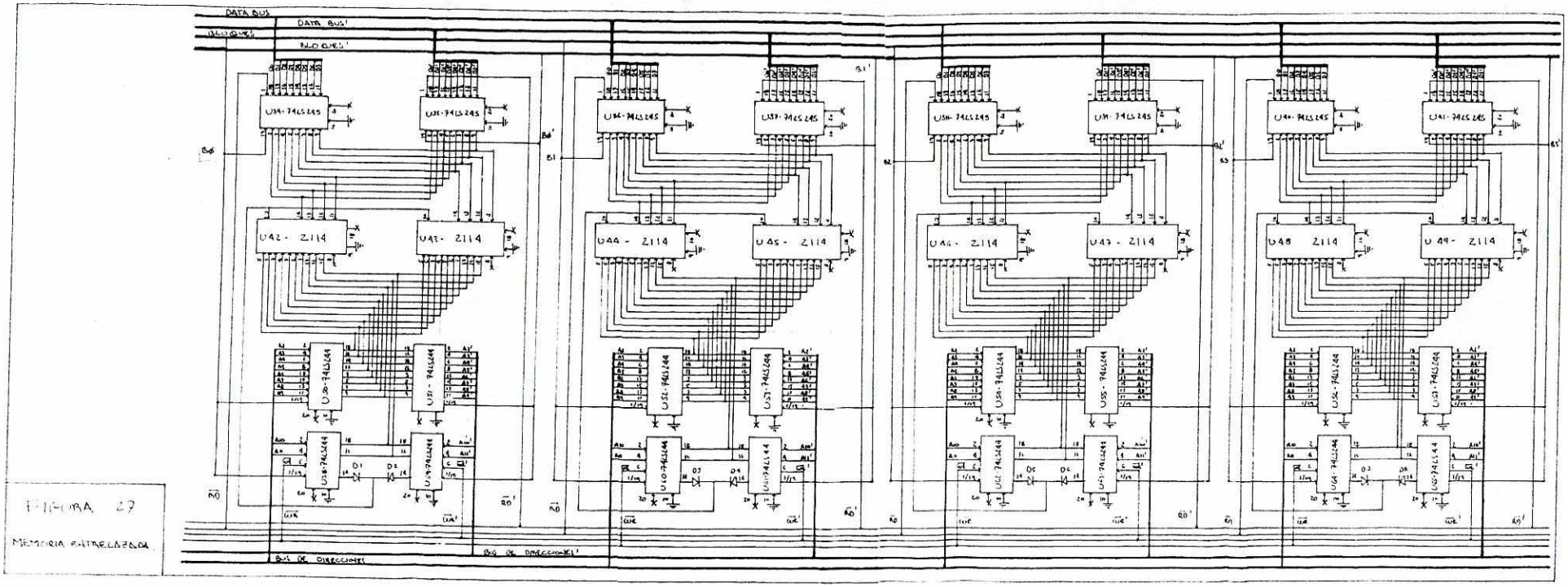
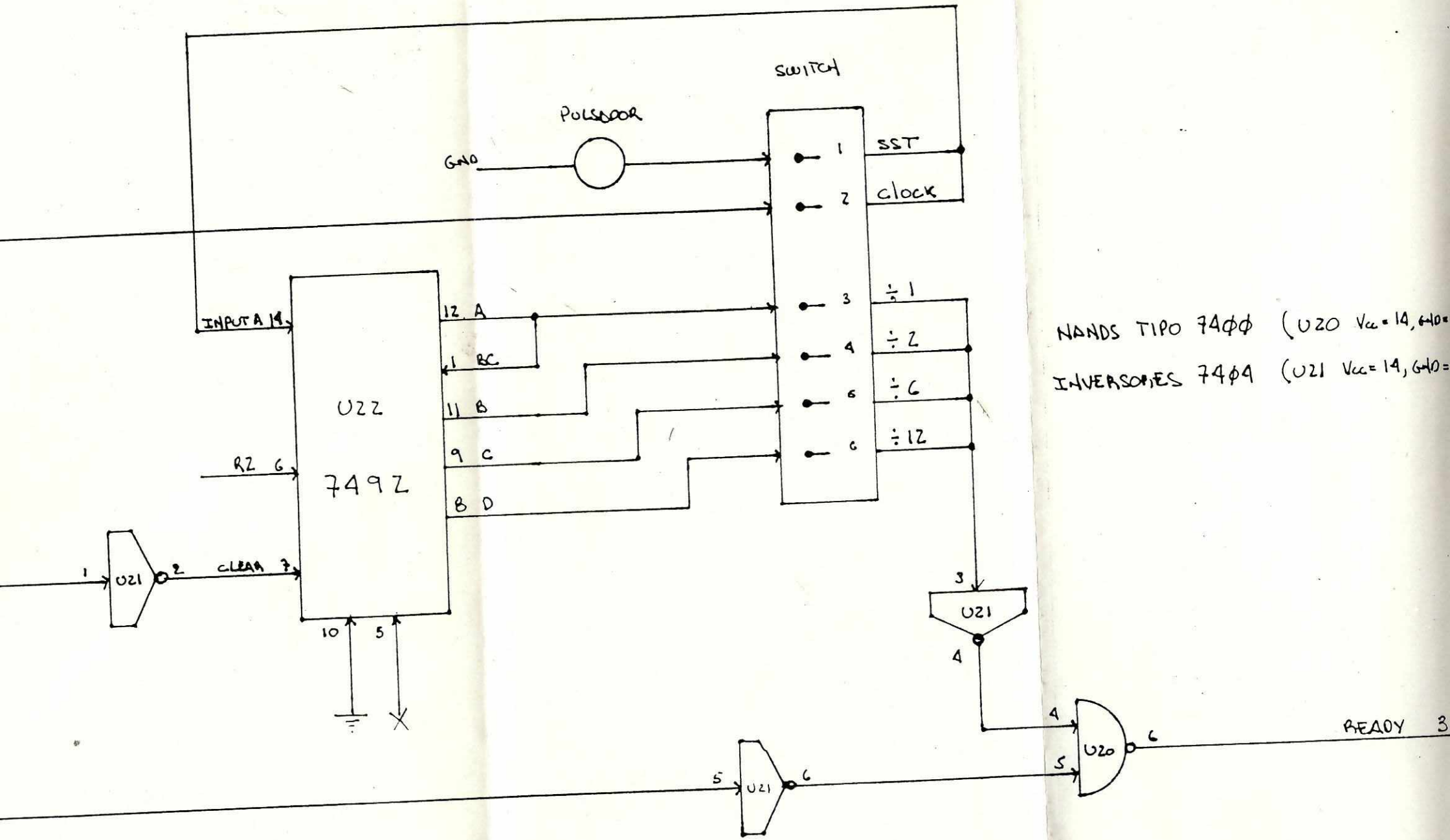


FIGURA 27

MEMORIA ENTRELAZADA



NANDS TIPO 74φφ (U20 Vcc = 14, GND =

INVERSORES 74φ4 (U21 Vcc = 14, GND =

de sincronización de la memoria con el procesador en los accesos y la segunda es una herramienta de trabajo. Es una lógica que permite detener indefinidamente el procesador cada vez que quiere usar la memoria. Esto sirve para que en ese lapso se pueda verificar el estado de cada línea del sistema. Toda la lógica se aprecia en la Fig. 30.

1. Sincronizador. El problema de sincronización surge de que la memoria es más lenta (450 Nanoseg.) que la velocidad de acceso del microprocesador (325 Nanoseg.). Los diseñadores del 8085, pensando en que eso podía suceder según del tipo de memoria que se usara, implementaron la señal 'ready' (pin 35) hasta que ésta se ponga en '1'. El 8085 asume que la información de piezas más lentas no está lista.

La lógica recibe dos señales de activación $\overline{ML}$ y $\overline{MI}$ ($\overline{ML}$' y $\overline{MI}$' en el otro procesador). Cuando una de las líneas está activa, se habilita un divisor de frecuencia que tiene como entrada el reloj del bus. Un interruptor escoge por cuanto se quiere dividir el reloj y después de ese tiempo se activa la señal de 'ready'. La función que tiene el interruptor se explica en la Tabla 5.

2. Lógica de Paso a Paso (SST). Para crear el SST lo único que varía de la lógica de sincronización es la entrada del reloj. Normalmente es el reloj del bus. Cuando el interruptor se pone en SST (ver Tabla 5), el reloj proviene de un procesador digital manual, con lo cual podemos detener indefinidamente el procesador.

No.	Acción
1	Activa lógica de paso a paso
2	Activa el reloj del sistema
3	Divide entre uno el reloj
4	Divide entre dos el reloj
5	Divide entre seis el reloj
6	Divide entre doce el reloj
7	Ninguno
8	Ninguno

Tabla 5 Interruptor

I. SOFTWARE DEL SISTEMA

El software implementado para pruebas se divide en dos partes: el software del prototipo en sí y el creado para la Z90, como unidad exterior. El del prototipo se encuentra grabado en ROM y el de la Z90, en diskette. Los listados de los programas se detallan en el apéndice "B".

1. Software del Prototipo. Este consiste en dos partes: software de E/S y un programa para hacer la prueba del sistema.

El software de E/S está unicamente en el procesador uno, que posee los medios de control a los periféricos exteriores. Son dos rutinas, una de lectura y otra de escritura de un caracter. La estructura es como se explicó en la sección de E/S.

La razón por la cual el programa se colocó en los dos procesadores es la facilidad de tenerlos ya grabados en ambos ROMS y no tener que leer el programa de una unidad externa. El programa es exactamente el mismo en ambos ROMS (sólo varían las direcciones donde se graba). Podría escribirse directamente a la memoria común para que ambos procesadores trabajaran sobre el mismo código (es código puro) aunque esto haría un poco menos eficiente el algoritmo.

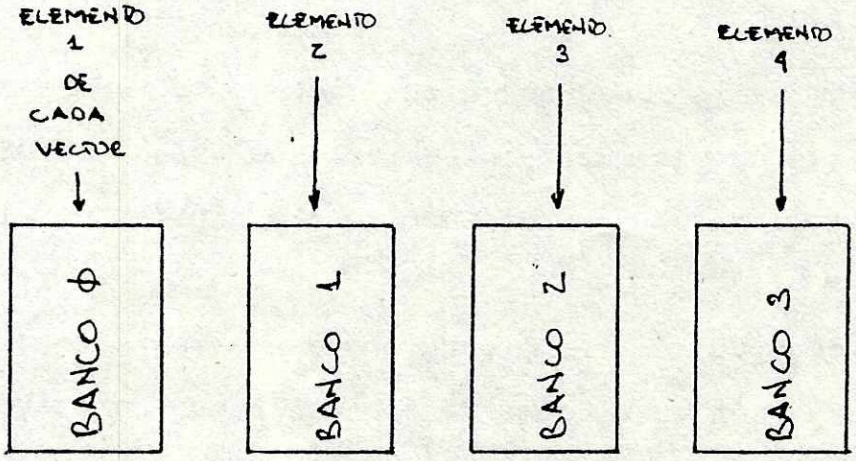
2. Software para el Z90. El software consiste en una rutina que permite utilizar la computadora Z90 como terminal. Es una rutina que lee y escribe en la pantalla lo que el modelo le pida.

Esta rutina es fácilmente expansible para permitir que la Z90 sirva como medio para guardar los programas y datos que el modelo necesite.

IV. CONCLUSIONES

Asumiendo que el diseño está correcto la demostración de que este computador es más rápido que uno basado en el uniprocesador se puede hacer así:

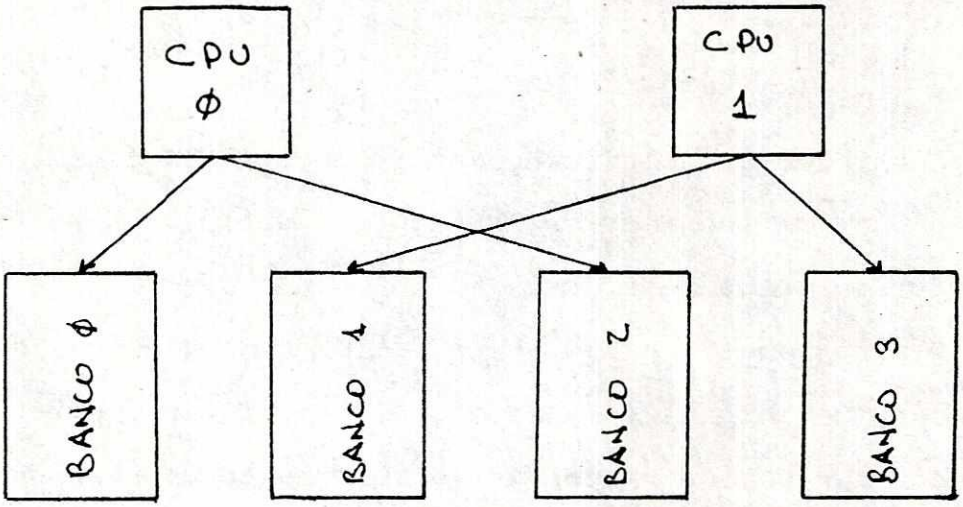
El programa de prueba (ver Apéndice C) que suma dos vectores de Bytes, organiza la información de la siguiente manera; en cada uno de los bancos de la memoria común pone un Byte de cada vector. (Incluyendo el de los resultados).



El programa principal (en el procesador 0) suma todos los elementos con dirección impar y el programa secundario (en el procesador 1) suma los elementos con dirección par.

Por la forma en que funciona la memoria entrelazada, nunca habrá conflicto por un dato entre los dos procesadores (nunca usarán el mismo banco) ejecutando cada uno la mitad de las

operaciones. O sea, tardando la mitad del tiempo.



En el caso de usar enteros en vez de bytes, sucede exactamente lo mismo.

Para reales puede haber conflicto, pero media vez exista uno, pasará más de una instrucción para que exista otro; por lo tanto, no es el doble de rápido, pero es más rápido que el de uniprocesador.

Como se ve, el método para obtener mayor eficacia es dividir la información en bloques de dirección que no provoquen conflictos, o reducir éstos al mínimo.

La concurrencia de dos algoritmos con código en la memoria común es otra forma de aumentar la velocidad. Para esto ya existe las primitivas Join y Fork (conjuntamente con CL/$\overline{OP}$) y el número de conflictos por un cierto banco se reduce a un 25 porciento (puesto que son cuatro bancos) obteniendo un 75 porciento más de velocidad.

APENDICE A

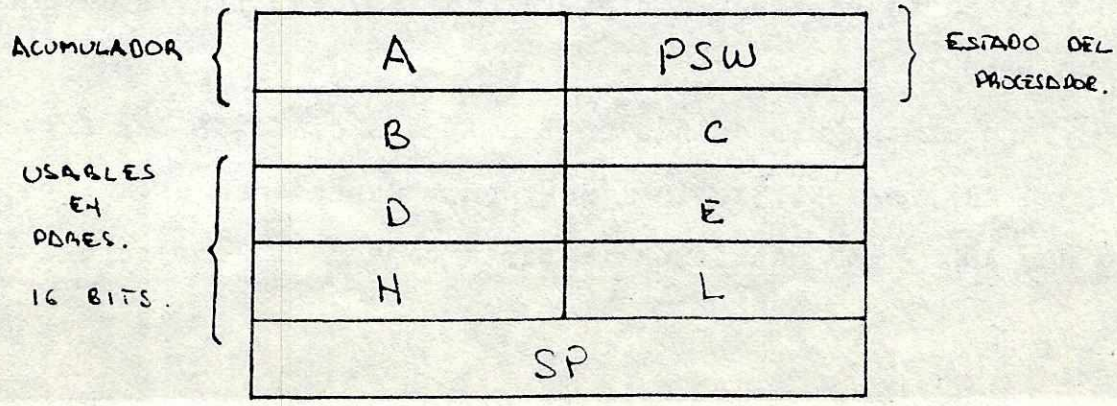
Descripción de Circuitos Integrados

1. INTEL 8085

El Intel 8085 es una versión avanzada del Intel 8080. Las principales diferencias son las siguientes:

- El 8085 sólo requiere +5 voltios.
- El 8085 usa una sóla señal de reloj.
- El 8085 posee una capacidad primitiva para hacer E/S serial.
- El 8085 tiene pines para interrupciones por vector.
- El 8085 trabaja a velocidades de hasta 350 Nanoseg, contra 500 Nanoseg del 8080.
- El 8085 multiplexa las líneas de datos con las líneas de dirección (AD0 - AD7).
- El 8085 usa las instrucciones SIM y RIM que no posee el 8080, por lo demás, el juego de instrucciones es el mismo.

1.1 Registros programables. Estos son idénticos a los del 8080. Se pueden ilustrar de la siguiente forma.

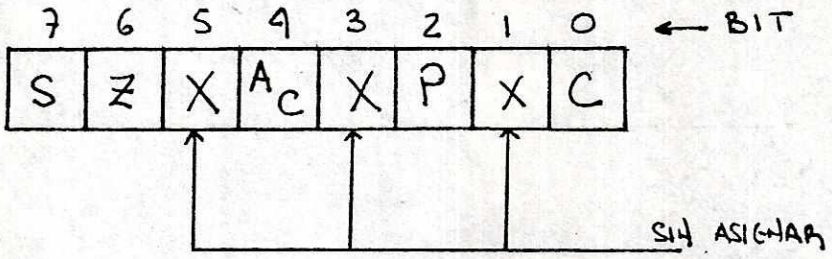


1.2 Formas de 'Referencia'. El 8085 usa referencia directa, implícita e inmediata al igual que el 8080.

1.3 Estado del 8085. La palabra de estado del programa (PSW) contiene las siguientes señales:

- Cero (Z)
- Signo (S)
- Paridad (P)
- Acarreo (C)
- Acarreo Auxiliar (AC)

La forma en que se asignan es así:



1.4 Pines y Señales. Las señales y sus pines correspondientes se ilustran en la Fig. 31 .

Las líneas de dirección y de datos se encuentran multiplexadas. La parte alta de la dirección la forman las líneas A15 - A8 y la parte baja se multiplexa con las ocho líneas de datos (AD0 - AD7).

Para distinguir cuando en las líneas de datos hay dirección (AD0 - AD7) se utiliza la línea ALE. Cuando ALE esta alto, indica que AD0 - AD7 contienen la parte baja de la dirección.

Se tienen cinco líneas de control para memoria y entrada/salida. $\overline{RD}$ indica una operación de lectura cuando está baja; $\overline{WR}$, una operación de escritura; IO/$\overline{M}$, acceso a la memoria si está baja, y acceso a entrada/salida si está alta.

El estado del bus del sistema se define por medio de las líneas SO y Sl. Su significado es el siguiente:

Sl	SO	Operación
0	0	Detenido (HALT)
0	1	Se está escribiendo
1	0	Se está leyendo
1	1	Se está leyendo una Instrucción

La lógica externa que es muy lenta para responderle al 8085 puede usar la señal de Ready para insertar períodos de espera en un ciclo de máquina. Para esto, se debe pulsar Ready a un estado de cero lógico.

Las líneas SID y SOD sirven para leer un bit (el más significativo) en el acumulador o para sacarlo al exterior. Representan una forma primitiva de entrada/salida.

Existen dos líneas para que la lógica externa pueda tomar control del bus del sistema. HOLD se usa para obligar al 8085 a poner el bus del sistema en estado Z, y HOLDA indica cuando el bus ya se encuentra en ese estado. Estas líneas se pueden usar para hacer accesos directos a memoria (DMA).

Hay seis señales asociadas con la lógica de interrupciones:

INTR, TRAP, RST 7.5, RST 6.5, RST 5.5. $\overline{\text{INTA}}$ se usa para avisar que un INT ha sido reconocido. INTR es una línea de interrupciones de tipo general, es igual a la del 8080. RST 7.5, RST 6.5 y RST 5.5 son implementaciones en hardware de interrupciones vectoriales y TRAP es una interrupción no enmascarable.

Hay dos líneas relacionadas con la lógica de 'inicialización': $\overline{\text{RESET IN}}$ que, cuando está baja, inicializa el 8085 y RESET OUT, que es una señal generada por el 8085 cuando es inicializado.

Las líneas X1 y X2 se usan para dar al 8085 la señal de reloj. Se puede conectar un cristal directamente o poner un reloj directo en X1. Se puede generar el reloj también por medio de un circuito RC.

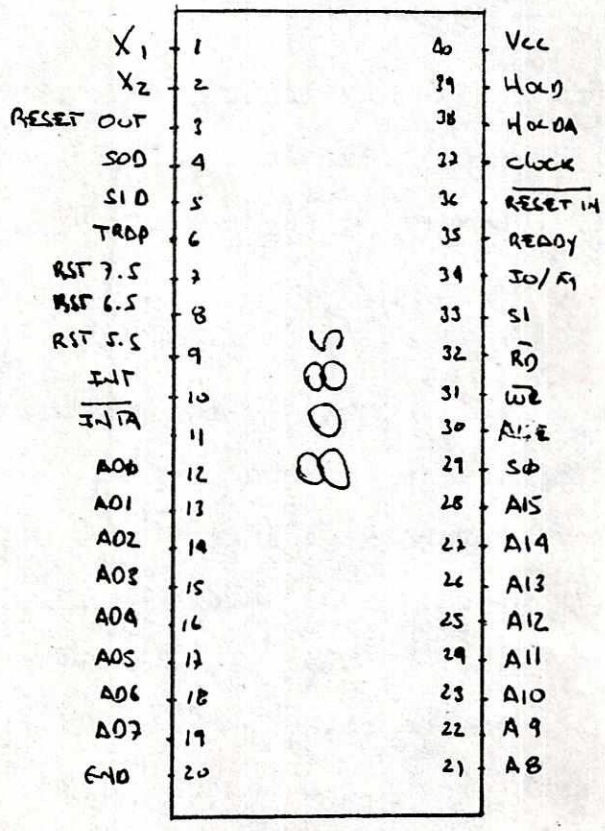


Fig 31. Pines del 8085

2. 7400 (QUADRUPLE COMPUERTA NAND DE DOS ENTRADAS)

2.1 Descripción General. Utiliza lógica TTL para obtener una alta velocidad con una disipación moderada de energía.

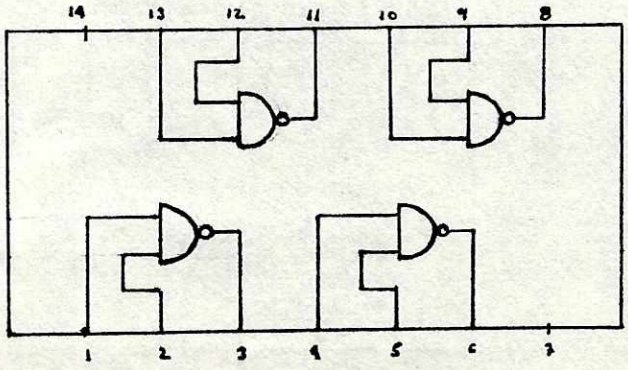
Estas compuertas ofrecen las funciones básicas en la implementación de circuitos digitales.

Son compatibles con las series 54/74.

2.2 Características

Parámetro	Magnitud	Unidad
- Inmunidad típica al ruido	1	Voltios
- Inmunidad al ruido garantizada	400	mVoltios
- Carga de salida	10	-
- Tiempo promedio de propagación	13	nS
- Disipación promedio de energía (por compuerta)	10	mWatts

2.3 Esquema.



3. 7404 (HEXA-INVERSOR)

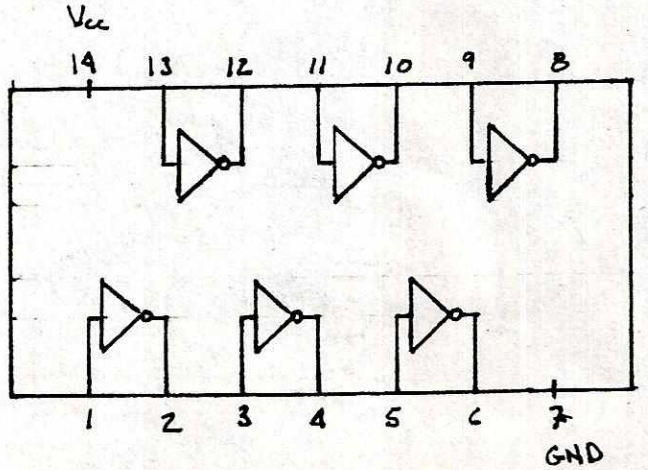
3.1 Descripción General. El 7404 es un Hexa-Inversor que utiliza tecnología TTL para obtener una velocidad a un consumo nominal de energía.

Es compatible con las series 74/54.

3.2 Características

Parámetro	Magnitud	Unidad
- Diodo de protección en la entrada		
- Inmunidad típica al ruido	1	Voltio
- Inmunidad garantizada al ruido	400	mVoltio
- Carga de salida	10	-
- Tiempo promedio de propagación	12	nS
- Disipación promedio de energía (por compuerta)	10	mWatts

3.3 Esquema.



4. 7474 (DE FLIP-FLOP D DUAL)

4.1 Descripción General. Están diseñadas para usarse donde la flexibilidad de dos entradas, como en Flip-Flop JK y RS, no es necesaria.

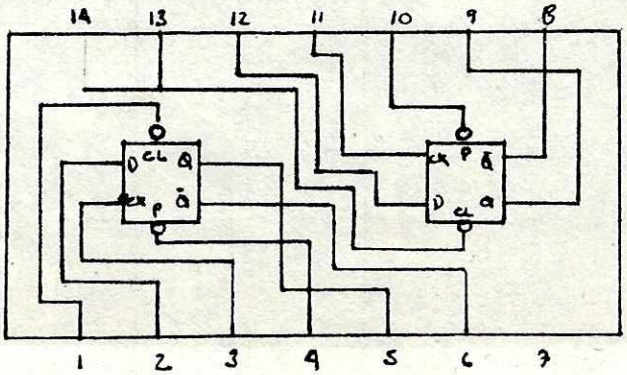
El 7474 sólo posee una entrada de datos que es transmitida a la señal Q cuando el reloj hace su transición a uno. La información debe estar lista unos nanosegundos antes que el reloj y debe permanecer unos cuantos más después de que el reloj se encuentra en uno.

Se ofrece la capacidad de preset y reset dentro del mismo paquete de 14 Pines.

4.2 Características.

Parámetro	Min.	Typ.	Max	Unidad
- Diodo protector en la entrada				
- Voltaje de 1 lógico	2.0			Voltios
- Voltaje de 0 lógico			0.80	Voltios
- Corriente por flip-flop		8.2		mAmp.
- Frecuencia máxima del reloj	15	25		MHZ

4.3 Esquema.



5. 74LS138 (DECODIFICADOR/DEMULTIPLEXADOR)

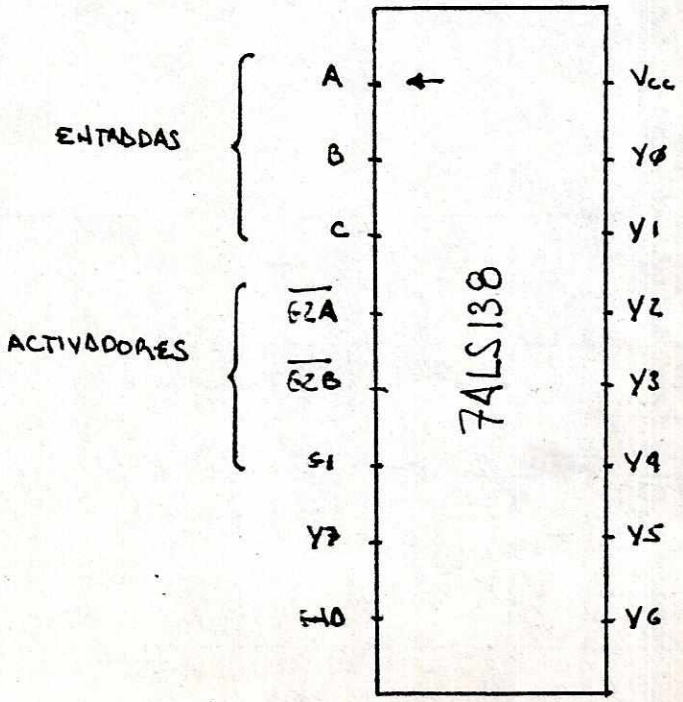
5.1 Descripción General. Estos circuitos MSl, TTL con diodo Schottky están diseñados para usarse en decodificación de memoria con suma eficiencia, donde es necesario pequeños tiempos de propagación.

El 74LS138 son decodificadores del tipo 3-8 con tres entradas de datos. Tiene también tres líneas de selección que reduce la lógica externa cuando se utilizan en cascada. En aplicaciones especiales se puede usar una de las líneas de selección como entrada de datos..

5.2 Características.

- 3 líneas de selección (G2A y G2B se usan en conjunto).
- Voltaje necesario . . . . . 5 Voltios
- Corriente en Nivel alto . . . . . 400 mAmp.
- Corriente en Nivel bajo . . . . . 8 mAmp.

5.3 Esquema



6. 74LS139 (DECODIFICADOR/DEMULTIPLEXAR)

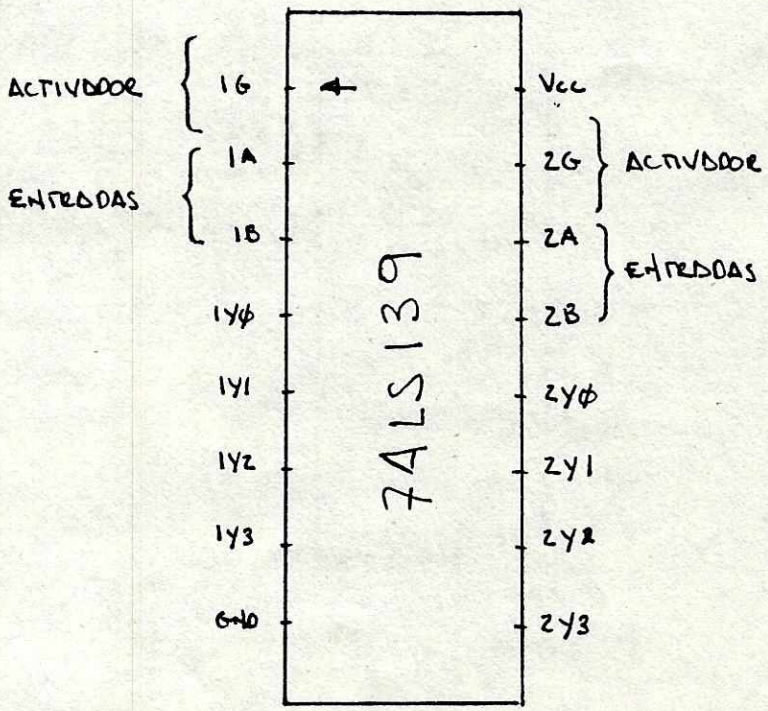
6.1 Descripción General. Este circuito TTL MSI con diodo Schottky están diseñados para usarse en decodificación de memoria con suma eficiencia.

El 74LS139 es un decodificador 1 a 4 dual en un simple paquete. El activador de estado bajo se puede usar como dato en aplicaciones de demultiplexación.

6.2 Características.

- 3 líneas de selección (G2A y G2B se usan en conjunto).
- Voltaje necesario . . . . . 5 Voltios
- Corriente en Nivel alto . . . . . 400 mAmp.
- Corriente en Nivel bajo . . . . . 8 mAmp.

6.3 Esquema.

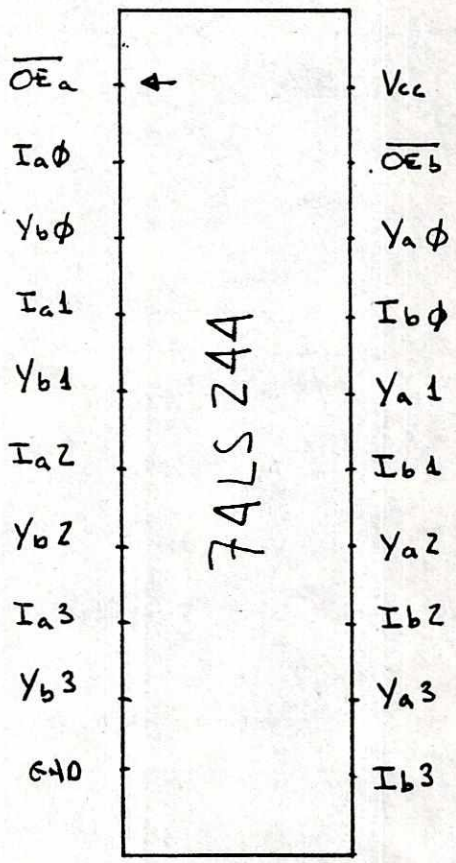


7. 74LS244 (BUFFER OCTAL DE ESTADO Z)

7.1 Características.

- Tiempo de Propagación . . . . . 18 nS
- Activación . . . . . 23 nS
- Desactivación . . . . . 28 nS
- Fuente de Voltaje . . . . . 5 Voltios

7.2 Esquema.

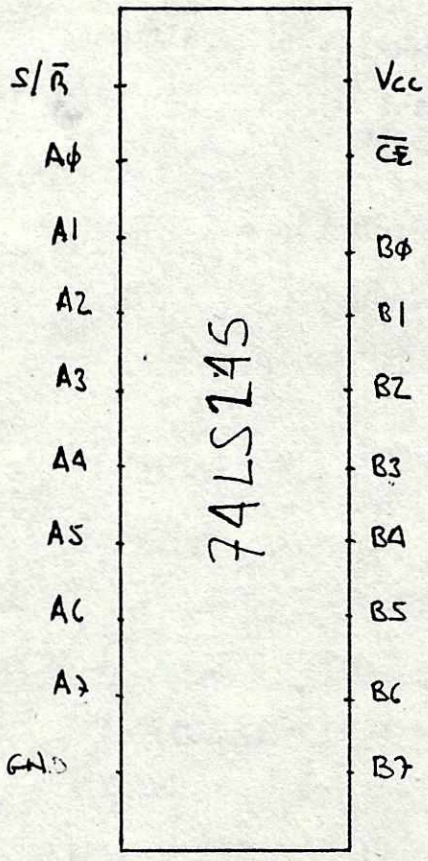


8. 74LS245 (BUFFER OCTAL BIDIRECCIONAL DE ESTADO Z)

8.1 Características.

- Tiempo de propagación . . . . . 12 nS
- Activación . . . . . 40 nS
- Desactivación . . . . . 40 nS
- Fuente de Voltaje . . . . . 5 Voltios

8.2 Esquema.



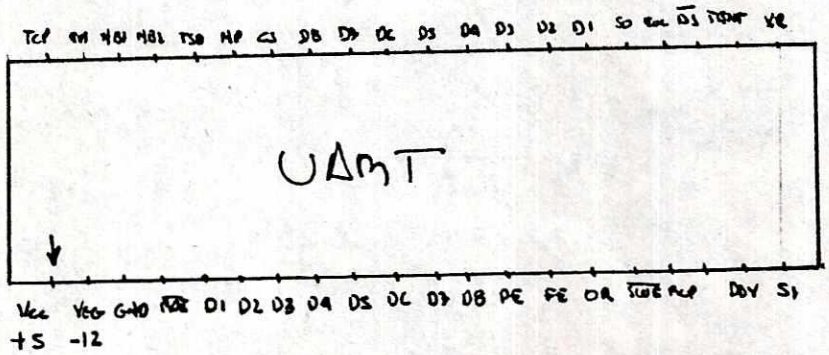
9. AY - 3 - 1013 (UART)

9.1 Descripción General. Este es un subsistema receptor/transmisor asincrono. Usa un start bit, con la opción de transmitir de cinco a ocho bits, con o sin paridad. Detecta errores de superposición, paridad o inicio fàlso. Es compatible con los niveles TTL.

9.2 Características.

- Compatible en DTL y TTL
- Transmisión y recepción simultáneo
- Bauds seleccionables (Full Duplex)
- Verificación de inicio
- Inmunidad de distorsión del 46%
- Capacidad de estado Z
- Bajo consumo de potencia
- Entrada protegida

9.3 Esquema.



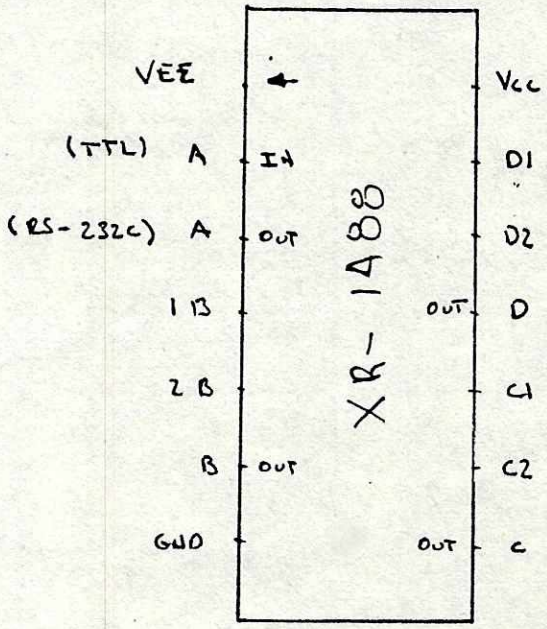
10. XR - 1488 (DRIVER DE CUATRO LINEAS RS-232C)

10.1 Descripción General. El XR-1488 es un driver de cuatro líneas monolítico, diseñado para conectar periféricos con las especificaciones EIA del standard RS-232C.

10.2 Características.

- Salida de corriente limitada a ±10 mA
- Impedancia mínima 300Ohms
- Compatible con todas las familias Motorola

10.3 Esquema.



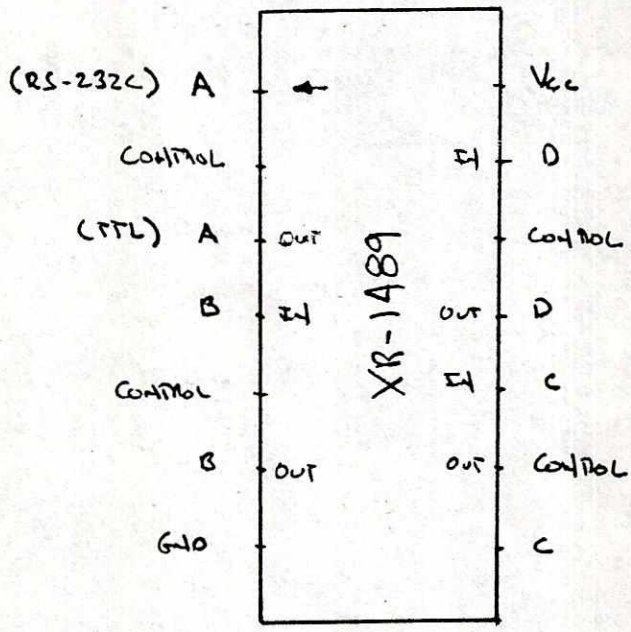
11. XR - 1489 (DRIVER DE CUATRO LINEAS RS-232C)

11.1 Descripción General. El XR - 1489 es un driver de cuatro líneas monolítico, diseñado para conectar periféricos con las especificaciones EIA del standard RS-232C.

11.2 Características.

- Resistencia de entrada de 3K a 7K
- Rango de la señal de entrada ± 30 Voltios

11.3 Esquema.



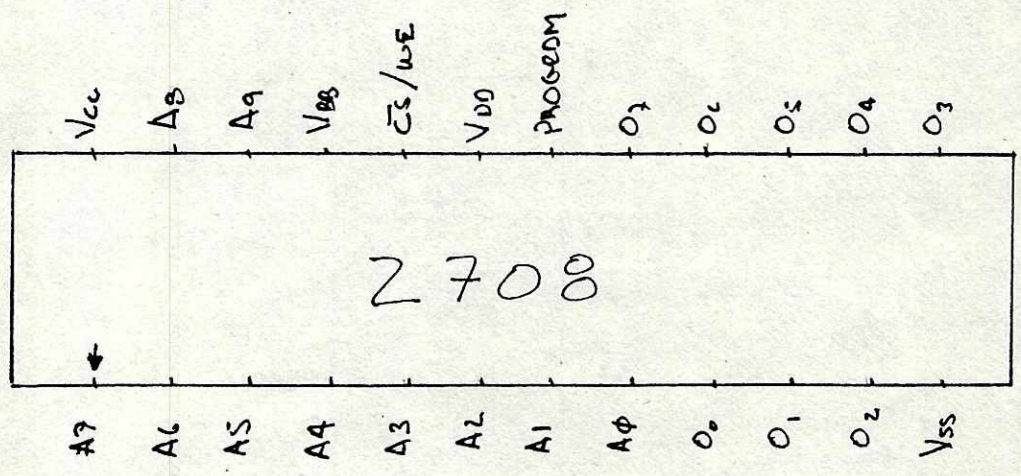
12. 2708 (EPROM)

12.1 Descripción General. El Intel 2708 es un EPROM de 8,192 Bits borrable por medio de luz ultravioleta y reprogramable electrónicamente. Tiene capacidad de estado Z para conectar directamente a los buses. Su uso es ideal para experimentación y en sistemas de alta velocidad.

12.2 Características.

- Tiempo de Acceso . . . . . 350 nS
- Compatible con los ROM de 8 y 16K
- Tiempo de programación . . . . . 100 Seg.
- Organización de 1K por 8 bits
- Compatible con los niveles TTL
- Salidas de estado Z

12.3 Esquema.



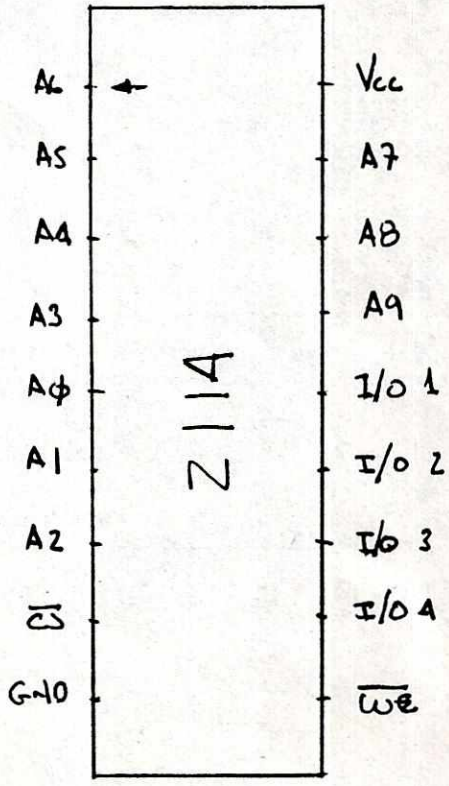
13. 2114 (RAM)

13.1 Descripción General. La familia 2114 de organización 1,024 por 4 bits es totalmente estática, no requiere renovaciones. Se utilizan los mismos pines de entrada que para salida y se conserva la polaridad ..

13.2 Características.

- Compatible con los niveles TTL
- Operación estática
- Bajo consumo 225 mWatts
- Alta velocidad 200 Ns
- Capacidad de estado Z
- Un solo voltaje

13.3 Esquema.



APENDICE B

Instrucciones Join y Fork
Implementación Completa

1. INTRODUCCION

La creación de arquitecturas de tipo multiprocesador que permite la concurrencia de dos o más procesos y la necesidad de implementar esta capacidad en lenguajes de alto nivel, ha obligado al desarrollo de nuevas instrucciones y medios de sincronización.

Se explica a continuación el posible diseño de las instrucciones Join y Fork, así como el uso de regiones críticas con fines de sincronización entre procesos.

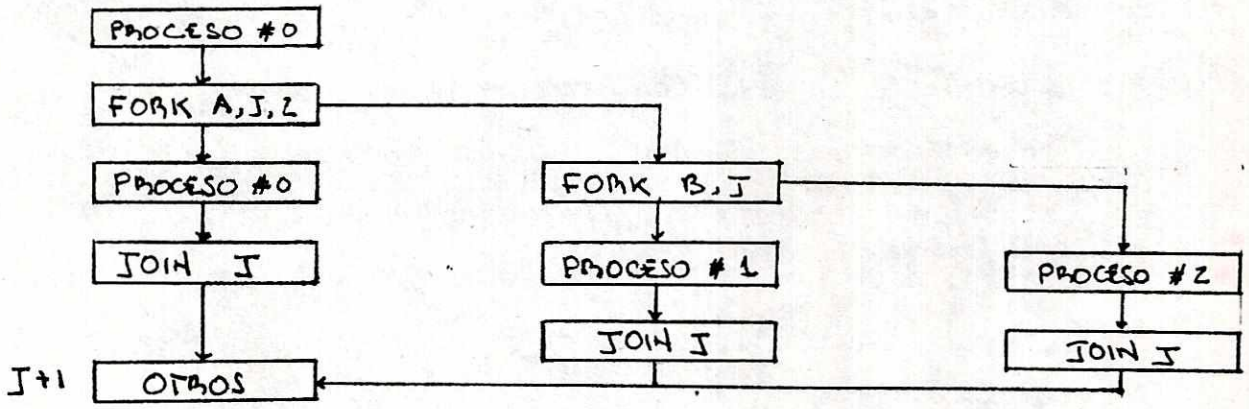
2. SEMANTICA DEL PAR JOIN-FORK

- Fork A: Inicia un proceso en la dirección A; el proceso activador continúa su operación normalmente.
- Fork A, J: Igual a Fork A. Además, el contador en la dirección J, es incrementado en 1.
- Fork A, J, N: Igual a Fork A. El contador en la dirección J toma el valor N.
- Join J: El contador en la dirección J se decrementa en uno. Si el contador es cero, se inicia un proceso en la dirección (J + 1), si no se libera el procesador que ejecuta el Join.
- Join J, B: Igual a Join J. Sólo que cuando el contador

es cero, el flujo de control se traslada a la dirección B.

3. USO DEL JOIN-FORK

El siguiente ejemplo ilustra el uso de las instrucciones para controlar tres procesos concurrentes.



4. DISEÑO DE LA INSTRUCCION FORK

Son dos programas. Uno en el procesador activante y otro en el activado.

Se usan las tres primeras palabras de la memoria entrelazada para paso de parámetros. El significado de cada Byte es el siguiente:

Byte	Contenido	Significado
0	0	Indica Instrucción Fork A
0	1	Indica Instrucción Fork A, J
0	2	Indica Instrucción Fork A, J, N
1,2		Dirección A
3,4		Dirección J
5		Valor N

El programa en el procesador activante podría ser así: (usando un macro assembler para definirla).

```
FORK  MACRO  T, A, J, N
      CALL   SAVER          ;Guardar Registros
      LXI    H, PARAM       ;Dirección de los parámetros
      MVI    A, T
      MOV    M, A           ;Tipo de Fork
      INX    H
      LXI    D, A
      MOV    M, D
      INX    H
      MOV    M, E           ;Dirección A
      INX    H
      LXI    D, J
      MOV    M, D
      INX    H
      MOV    M, E           ;Dirección J
      INX    H
      MVI    A, N
      MOV    M, A           ;Valor N
      OUT    O              ;Activar Procesador
      CALL   RECUP          ;Recuperar Registros
      ENDM
```

El programa receptor del Fork en el otro procesador se encuentra en la dirección $3C_{16}$ y es así:

```
FORK:   LDA   PARAM
        CP    0
        JPZ   FORKA          ;Es un Fork A
        CP    1
        JPZ   FORKAJ         ;Es un Fork A, J
        LXI   H, PARAM + 3
        LDA   PARAM + 5
        MOV   M, A           ;Poner contador J en valor N
        LXI   H, PARAM + 1
        PCHL
FORKAJ: LXI   H, PARAM + 3
        MOV   A, M
        INR   A
        MOV   M, A           ;Incrementar Contador J
FORKA:  LXI   H, PARAM + 1   ;Bifurcación a Dirección A
        PCHL
```

5. DISEÑO DE LA INSTRUCCION JOIN

Usando el mismo formato que para Fork, obtenemos:

```
        MACRO T, J, B
        CALL  SAVER          ;Guardar Registros
        LXI   H, PARAM       ;Dirección de parámetros
        MVI   A, T           ;Tipo de Join
        MVO   M, A           ;0  Join J
        INX   H              ;1  Join J, B
        LXI   D, J
        MOV   M, D
```

```
        INX   H
        LXI   D, B
        MOV   M, D
        INX   H
        MOV   M, E              ;Dirección B
        IN    O                 ;Ejecute Join
        CALL  RECUP             ;Recupera Registros
        GO TO HALT
        ENDM
```

El programa receptor de Join es así: (se encuentra en la dirección 34_{16})

```
JOIN:   CALL  SAVER             ;Guardar Registros
        LDA   PARAM
        CP    O
        JPZ   JOINJ             ;Es un Join J
        LXI   H, PARAM + 1
        MOV   A, M
        DCR   A                 ;Decrementar Contador
        MOV   M, A
        CP    O
        JPZ   BIF               ;En caso de Cero
        CALL  RECUP             ;Recuperar registros
        RET
BIF:    LXI   H, PARAM + 3
        PCHL
```

```
JOINJ:  LXI   H, PARAM + 1      ;Es Join J
        MOV   A, M
        PCR   A
        MOV   M, A              ;Decrementar Contador
        INX   H
        CP    O
        JPZ   BIF1              ;Hacer la Bifurcación
        CALL  RECUP             ;Recuperar Registros
        RET
BIF1:   PCHL                    ;Hacer el Cambio
```

6. PROBLEMAS DE SINCRONIZACION

Para evitar la destrucción de información en el paso de parámetros o superposición de un proceso sobre otro, se deben crear regiones críticas. En hardware existe el sistema CL/$\overline{OP}$ que nos ayuda a crearlas.

La instrucción CLOSE (P (MUTEX)), es diseñada así:

```
CLOSE   MACRO
        DI                      ;Quitar interrupciones
        CALL  SAVER             ;Guardar Registros
        MVI   A, 300Q
        SIM                     ;Activación de CL/OP
        CALL  RECUP             ;Recuperar Registros
        ENDM
```

La región crítica es la memoria común (donde pueden ocurrir los conflictos) y la forma de abrirla es de la siguiente forma (V (MUTEX)).

```
OPEN  MACRO
      CALL  SAVER          ;Guardar Registros
      MVI   A, 100Q
      SIM                  ;Desactivación CL/OP
      CALL  RECUP          ;Recuperar Registros
      EI                   ;Poner Interrupciones
      ENDM
```

Estas rutinas para cerrar y abrir regiones críticas deben ser incluidas al principio y final de cada instrucción Join y Fork; ya sea de activación o recepción.

APENDICE C

Programas

Se presentan los programas que convierten la Z90 en terminal inteligente y el que sirve para probar el prototipo. A continuación se presenta el programa de prueba:

```
          ORG   0
INICIO:   LXI   H, VECA          ;programa en procesador 0
          SHLD  A
          LXI   H, VECB
          SHLD  B
          LXI   H, VECC
          SHLD  C
          MVI   A, 254           ;508 elementos
          STA   CONT
          MVI   A, 0
          STA   PARAM
          OUT   0                ;activar CPU1
LOOP:     LHLD  A
          MOV   A, M
          INX   H
          INX   H
          SHLD  A
          LHLD  B
```

```
        MOV   B, M
        INX   H
        INX   H
        SHLD  B
        LHLD  C
        ADD   B
        MOV   M, A
        INX   C
        INX   C
        SHLD  C
        LDA   CONT
        DCR   A
        DCR   A
        JNZ   LOOP
VECA    EQU   5000          ;Datos del programa
VECB    EQU   5508
VECC    EQU   6016
PARAM   EQU   4096
A       EQU   1024
B       EQU   1026
C       EQU   1028
        END   INICIO
```

El programa en el procesador uno sería el siguiente:

```
        ORG   0
INICIO: HLT
        ORG   60
```

```
        LDA   PARAM
        CMP   0
        JZ    100              ;Correr programa
        CMP   1
        JZ    200              ;Sacar caracter
        CMP   2
        JZ    250              ;Leer caracter
        JMP   300              ;Inicializar
        ORG   100
        LXI   H, VECA + 1      ;Elementos impares
        SHLD  A
        LXI   H, VECB + 1
        SHLD  B
        LXI   H, VECC + 1
        SHLD  C
        MVI   A, 254           ;508 Elementos
        STA.  CONT
        ;MVI  A, 0             ;Sólo en CPU0
        ;STA  PARAM
        ;OUT  0
LOOP:   LHLD  A                ;Vector A
        MOV   A, M
        INX   H
        INX   H
        SHLD  A
        LHLD  B                ;Vector B
```

```
        MOV   B, M
        INX   H
        INX   H
        SHLD  B
        LHLD  C
        ADD   B
        MOV   M,A
        INX   C
        INX   C
        SHLD  C
        LDA   CONT
        DCR   A
        DCR   A
        JNZ   LOOP
        JMP   INICIO
VECA    EQU   5000
VECB    EQU   5508
VECC    EQU   6016
PARAM   EQU   4096
A       EQU   1024
B       EQU   1026
C       EQU   1028
        END   INICIO
```

```
&TES0  T=00004 IS ON CR00015 USING 00012 BLKS R=0000

0001  ;*****************************************************************************
0002  ;*                                                                           *
0003  ;*          PROGRAMA DE COMUNICACION ENTRE EL PROTOTIPO Y LA Z90              *
0004  ;*          ---------------------------------------------                    *
0005  ;*                                                                           *
0006  ;* I-ABSTRACTO:                                                              *
0007  ;* ------------                                                              *
0008  ;*    Este programa sirve de comunicacion entre la Z90 y el prototipo. El    *
0009  ;*    metodo utilizado es el de IO programado. Convierte la Z90 en una ter-  *
0010  ;*    minal inteligente.                                                     *
0011  ;*                                                                           *
0012  ;* II-AUTOR:                                                                 *
0013  ;* ---------                                                                 *
0014  ;*    David Alvarez.                                                         *
0015  ;*                                                                           *
0016  ;*****************************************************************************
0017  ;
0018  ;
0019  ;
0020  INICIO:  MVI   A,0      ;INICIALIZAR LOS PUERTOS DE LA Z90
0021           OUT   320Q
0022           OUT   350Q
0023           MVI   A,200Q   ;PONER LOS BAUDS.
0024           OUT   323Q
0025           OUT   353Q
0026           MVI   A,14Q    ;PONEMOS A 9600
0027           OUT   320Q
0028           OUT   350Q
0029           MVI   A,0
0030           OUT   321Q
0031           OUT   351Q
0032           MVI   A,6      ;PONEMOS 7 BITS Y DOS STOP BITS, SIN PARIDAD
0033           OUT   323Q
0034           OUT   353Q
0035  LOOP0:   IN    325Q     ;VER SI TRASMITE ALGO EL PROTOTIPO
0036           ANI   1
0037           JZ    INPRO    ;SI ENTRO.
0038           IN    355Q     ;VER SI METIERON ALGO EN LA PANTALLA
0039           ANI   1
0040           JNZ   LOOP0
0041           IN    350Q
0042           OUT   320Q     ;SE TRANSMITE
0043           JMP   LOOP0
0044  INPRO:    IN   320Q
0045           OUT   350Q     ;TRANSMINSION
0046           JMP   LOOP0
0047           END   INICIO
```

APENDICE D

Detección de Paralelismo

Al crear lenguajes de alto nivel para máquinas con multiprocesadores, es necesario detectar cuando dos o más porciones de código pueden ser ejecutadas concurrentemente.

Para detectar el paralelismo en un programa (P_i) se deben clasificar primero las distintas variables en la siguiente forma:

W_i Representa Variables sólo de lectura

X_i Representa Variables sólo de escritura

Y_i Representa Variables primero leídas y luego modificadas

Z_i Representa Variables primero modificadas y luego leídas

Para poder ejecutar dos segmentos de un programa P_i y P_j en forma concurrente antes de llegar al segmento P_k, existen tres condiciones que debe ser satisfechas.

*Las variables leídas por P_i no deben ser modificadas por P_j.

$$(W_i \cup Y_i \cup Z_i) \cap (X_j \cup Y_j \cup Z_j) = \emptyset$$

*Las variables leídas por P_j no deben ser modificadas por P_i.

$$(W_j \cup Y_j \cup Z_j) \cap (X_i \cup Y_i \cup Z_j) = \emptyset$$

*El estado del cómputo al llegar al segmento P_k debe ser independiente de los cómputos entre P_i y P_j.

O sea:

$$(W_k \cup Y_k) \cap (X_i \cup Y_i \cup Z_i) \cap (X_J \cup Y_J \cup Z_J) = \phi$$

O lo que es lo mismo:

$$X_i \cap X_J \cap (W_k \cup Y_k) = \phi$$

Si los P_i son proposición de un lenguaje de alto nivel, entonces las tres condiciones se reducen A:

$$I_i \cap O_J = \phi$$
$$I_J \cap O_i = \phi$$
$$O_i \cap O_J = \phi$$

Donde I_i representa las entradas del segmento P_i y O_i las salidas.

Las fases para analizar el paralelismo en un lenguaje de alto nivel (ALGOL, FORTRAN) usan las tres reglas anteriores y generan código de línea recta.

El análisis se sumariza así:

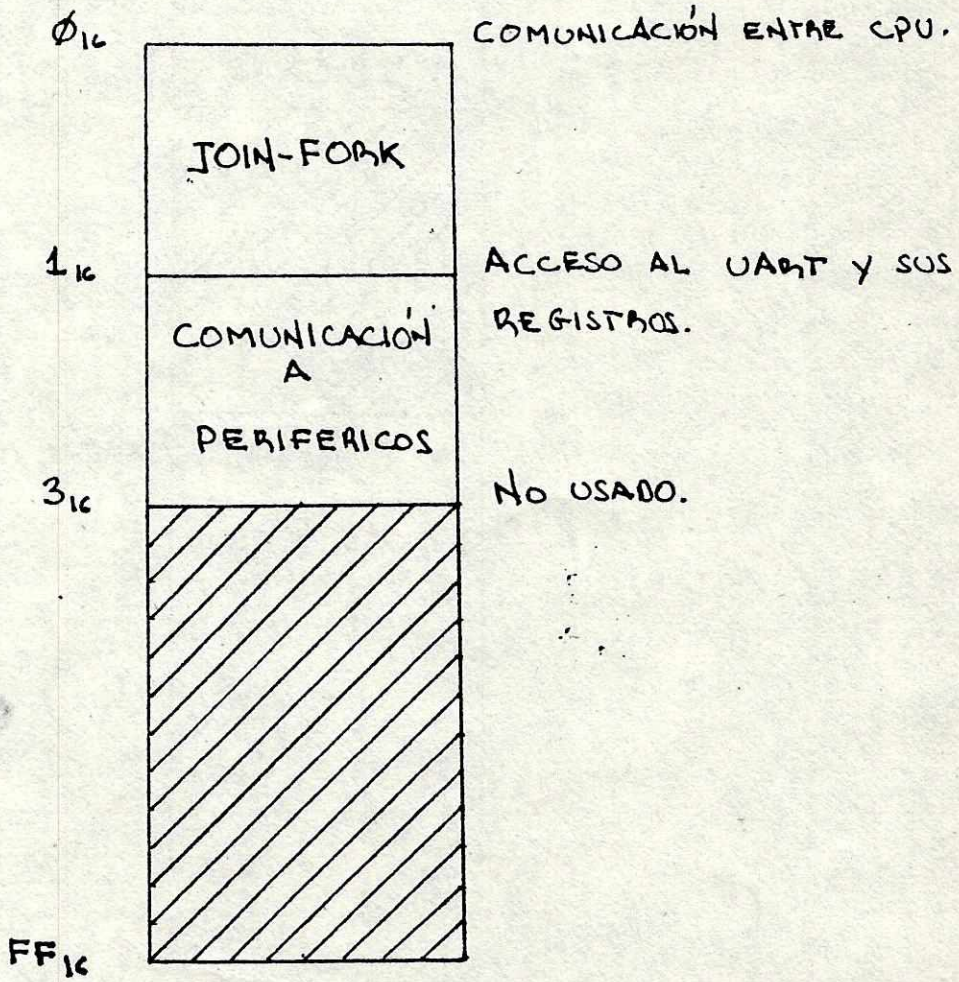
- Fase 1: Construir un grafo donde cada modo es una proposición y los conectores representan el flujo de control
- Fase 2: Detectar loops y ciclos, suprimir la retroalimentación convirtiendo el grafo en aciclico.
- Fase 3: Hacer una lista de los procedimientos dependientes (IF, Loops, etc).
- Fase 4: Crear los conjuntos I_i, O_i para cada una de las proposiciones.
- Fase 5: Detectar paralelismo (en el grafo aciclico) y modificar el grafo.

- Fase 6: Construir un nuevo grafo "paralelo" combinando el grafo aciclico de la fase 5 y los procedimientos dependientes. Este grafo es un grafo AND/OR. Con AND para los Join-Fork y OR para Loops y los Tests.

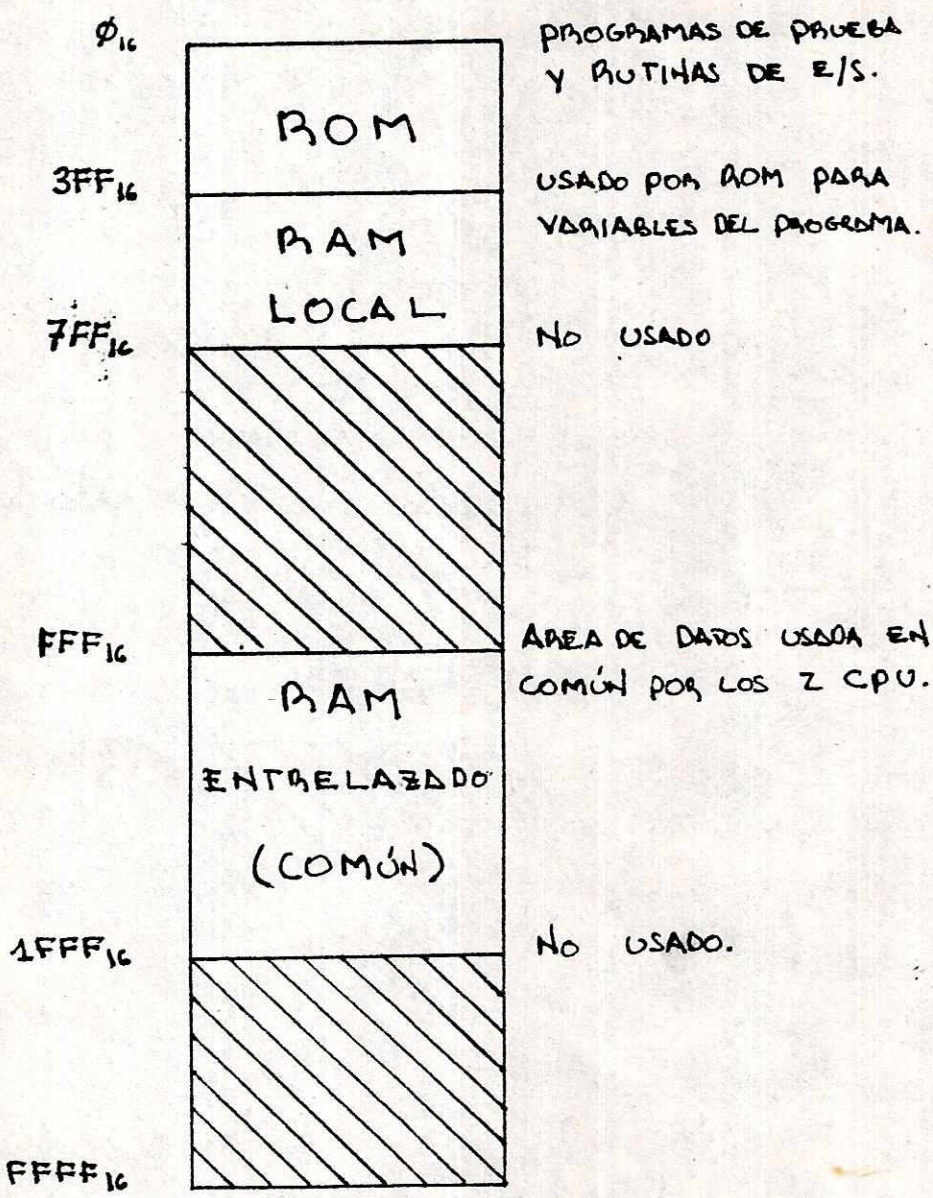
La detección del paralelismo también se puede extender a un análisis más riguroso dentro de los loops para aumentar la concurrencia.

APENDICE E

Mapeo de Entrada/Salida y Memoria



Mapeo de Entrada/Salida



Mapeo de Memoria

GLOSARIO DE TERMINOS

1. ALE: Address latch enable.
2. BAUD: Cantidad de bits a transmitir por segundo.
3. BIT: Un uno o cero binario.
4. BUS: Físicamente es un conjunto de alambres. Puede ser también un conjunto de señales.
5. BYTE: Ocho bits.
6. CPU: Central Processing Unit.
7. DRIVER: Regulador de las señales de entrada.
8. EPROM: Ereasable programmable read only memory.
9. INT: Interrupt.
10. LOOP: Ciclo o iteración dentro de un grafo o programa.
11. MFLOPS: Mega floating point operations per second.
12. PMS: Processor memory switch.
13. RAM: Random access memory.
14. ROM: Read only memory.
15. RST: Restart.
16. SID: Serial in data.
17. SOD: Serial out data.
18. TTL: Transistor transistor logic.
19. UART: Universal asyncronus receiver transmitter.

BIBLIOGRAFIA

Baer, Jean-Loup, Computer Systems Architecture. Rockville,
1980 Maryland, Computer Science Press, Inc. 625 pp.

Jim-Pak, 7400/74LS Data Book. Belmont, California. sf.

Jim-Pak, CMOS/Linear Data Book. Belmont, California. sf.

Jim Pak, Microprocessor/LED Data Book. Belmont, California.
sf.

Kuck, David, The Structure of Computer and Computations, Volumen
1978 1. New York, New York, John Wiley. 611 pp.

Myers, Glenford, Advances in Computer Architecture. New York,
1981 New York, John Wiley. 545 pp.

Digital System Design with LSI Bit-Slice Logic,
1980 New York, New York, John Wiley. 338 pp.

Osborne Adam, G. Kane, Osborne 4 & 8-Bit Microprocessor Handbook,
1981 San Francisco, California, Osborne/McGraw-Hill.

Tanembaum, Andrew, Structured Computer Organization, New Jersey,
1976 Prentice-Hall. 443 pp.

Wakely, John, Microcomputer Architecture and Programming, New York,
1981 New York, John Wiley. 692 pp.

Wegner, Peter, Programming with Ada, New York, New York,
1980 Prentice-Hall. 211 pp.