UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

Facultad de Ingeniería
División de Estudios Superiores

MULTIPROCESAMIENTO BASADO EN UNA MEMORIA COMPARTIDA POR n PROCESADORES

T E S I S

Que para obtener el título de:

MAESTRO EN INGENIERIA - ELECTRONICA

P R E S E N T A:

LUIS HUGO PEÑARRIETA ECHENIQUE

México D. F. 1978

Con cariño a mi abuelita, papas y hermana

Deseo expresar mi agradecimiento a Juan Luis Neuman, director de la tésis, Carlos E. Ramirez y Luis Lyons por sus orientaciones y - - amistad, asi como al Instituto de Ingeniería UNAM por las facilidades en el desarrollo de la parte practica, al grupo AHR por invitarme a participar en el proyecto de la Máquina de LISP, a José Sánchez y Raúl Cors por sus colaboraciones en este trabajo y a todos los que de una u otra manera contribuyeron en mi formación.

CONTENIDO

APENDICES

1

SISTEMAS COMPUTACIONALES MULTIPLES Y PARALELOS

1.1 INTRODUCCION

El desarrollo de las modernas computadoras digitales que se conocen en la actualidad se inició en la década de los 50; a fines de esta ya se conocían algunos sistemas compuestos por varias computadoras, como es el caso de SAGE (ref 1, pág 11) que apareció en 1958 y que estaba compuesto por dos computadoras que ejecutaban en forma independiente el mismo programa sobre la misma entrada de datos e iban comparando resultados intermedios hasta llegar al final, con objeto de incrementar la confiabilidad del sistema.

Sistemas redundantes de este tipo tuvieron que ser utilizados debido a la baja confianza que ofrecían las computadoras de aquella época, pero que sin embargo, contribuyeron con datos y experiencias importantes para el posterior desarrollo de sistemas de procesamiento paralelo y multiprocesamiento.

Los sistemas de procesamiento paralelo surgieron - de la necesidad de cambiar el procesador -usualmente grande y veloz-, de una computadora dedicada a manejar grandes cantidades de información por procesadores más pequeños que, - aunque más lentos, podían procesar la misma información - en forma paralela más rápidamente.

Los sistemas de multiprocesamiento se originaron - para satisfacer necesidades militares; el objetivo era disponer de un sistema con múltiples unidades de procesamien idénticas que en cualquier momento permitiera utilizar procesadores desocupados para satisfacer exigencias nuevas o - urgentes. Sin embargo, fue hasta después de varios años -- que los diseñadores de procesadores de carácter general sepercataron que los sistemas de multiprocesamiento podían -- también aumentar su rendimiento.

Muchas veces se confunden los términos procesamiento paralelo y multiprocesamiento, en otras se toman como sinónimos; sin embargo, conviene aclarar la gran diferencia - que existe: En procesamiento paralelo varios procesadores- realizan la misma operación simultáneamente* sobre datos - diferentes; en multiprocesamiento, los procesadores realizan concurrentemente* operaciones diferentes sobre datos -- iguales o diferentes.

Desafortunadamente, la aceptación de sistemas de - multiprocesamiento no fue tan grande como se esperaba debido a que la mayoría de las firmas comerciales se dedicaron- al desarrollo de la nueva generación de computadoras que se conoce actualmente; además, la experiencia adquirida con los pocos proyectos de multiproceso que se llevaron a cabo en - las universidades demostró que el problema principal de es

---

* Simultáneo: ocurrencia de dos o más eventos en el mismo- instante. Concurrente: ocurrencia de dos o más eventos- dentro de un mismo intervalo de tiempo.

te tipo de sistemas es la programación del mismo, que aun - en la actualidad no se ha llegado a resolver en forma óptima.

La gran disponibilidad actual de microprocesadores de bajo costo (MOS LSI μPs) ha creado un renovado interés - en los sistemas de multiprocesamiento; su bajo costo permite el uso de varios μPs en un sistema. Al presente se cuenta con una variedad amplia de μPs que pueden satisfacer distintas necesidades (Apéndice A).

1.2 ARQUITECTURA DE LOS SISTEMAS COMPUTACIONALES

Flynn (ref 2) ha clasificado los sistemas computacionales, según la secuencia de ejecución de instrucciones y adquisición de datos, en cuatro categorías:

SISD: Secuencia de instrucciones única, secuencia de datos unica.
SIMD: Secuencia de instrucciones única, secuencia de datos múltiple.
MISD: Secuencia de instrucciones múltiple, secuencia de datos única.
MIMD: Secuencia de instrucciones múltiple, secuencia de datos múltiple.

Debido a que MISD es prácticamente una clasificación vacía, no se considera en este estudio.

1.2.1 Secuencia de instrucciones única, secuencia de datos única (SISD)

A esta categoría pertenecen las computadoras de un solo procesador que ejecutan los programas en forma secuencial.

1.2.2. Secuencia de instrucciones única, secuencia de datos múltiple (SIMD)

A esta categoría pertenecen los llamados sistemas de procesamiento paralelo; la unidad de control central toma y decodifica las instrucciones que son ejecutadas por la misma unidad de control (saltos condicionales e incondicionales) o trasmitidas para su ejecución a otros elementos de procesamiento. De acuerdo con la complejidad de las unidades de control, de los modos de direccionamiento y de la interacción entre los elementos de procesamiento, SIMD se divide en las subcategorías que se detallan a continuación.

1.2.2.1 Procesadores ordenados

Sistema computacional con múltiples unidades aritméticas y lógicas, algunas veces referidas como elementos de procesamiento; su organización es tal que las operaciones se ejecutan en forma simultánea sobre un arreglo de datos. Estos procesadores son también llamados vectoriales porque ejecutan dichas operaciones sobre datos arreglados en forma de vectores. La capacidad de la unidad de control es limitada y solo ejecuta saltos condicionales e incondicionales; las operaciones aritméticas y lógicas las transfiere a los elementos de procesamiento para que sean ejecutadas por estos (ref 3).

Para conseguir el paralelismo, estos sistemas deben reunir tres características:

a) Las computaciones deben ser descritas por una secuencia de instrucciones que afectará simultáneamente a todos los elementos (datos) de un arreglo vectorial.

b) Presentar una velocidad alta en la transferencia de datos entre procesadores debido a que estas transferencias se llevan a cabo entre la ejecución de 2 instrucciones.

c) Todos los operandos contenidos en el arreglo vectorial deben tomarse simultáneamente.

De no cumplir estas condiciones, existe el peligro de pasar de un proceso paralelo de alta velocidad de salida a uno en serie.

Un ejemplo típico de estos sistemas es el Illiac - IV (ref 1, pp 45-47, 79, 103, y ref 4), que por las restricciones de operación síncrona y limitada capacidad de la unidad de control tiene un sistema operativo que reside en una computadora satélite. La aplicación de los procesadores ordenados se enfoca hacia análisis numérico, control de tráfico aéreo y procesamiento de señales de radar.

1.2.2.2 Procesadores asociativos

Los procesadores de este tipo accesan y operan sobre datos más por su contenido que por su dirección, (ref - 5). Un ejemplo es STARAN (ref 1, Apéndice F); que puede realizar simultáneamente operaciones aritméticas, lógicas o de búsqueda sobre todas o un bloque de palabras de su memoria.

1.2.2.3 Procesamiento por ensamble

Estos sistemas utilizan una combinación de procesadores ordenados y asociativos para obtener una velocidad alta de salida. Tienen una computadora de carácter general - que gobierna todos los elementos de procesamiento cuyo núme

ro varia y puede ser expandido o reducido según las necesidades de aplicación. Un ejemplo típico es PEPE (ref 1, Apéndice A).

Las computadoras Pipeline son difíciles de clasificarlas, porque según su funcionamiento pueden pertenecer a diferentes categorías (ref 4, pág 1272). Se consideran SIMD cuando su unidad aritmética de varias etapas ejecuta una misma operación sobre datos diferentes. El número de etapas es igual al de operaciones simultáneas que pueden realizar. Ejemplos de sistemas Pipeline son CDC 7600, IBM-S/360-91, CDC STAR-100, Texas Instruments ASC, etc (ref 1: pp 41-44).

1.2.3. Secuencia de instrucciones múltiple, secuencia de datos múltiple (MIMD)

A diferencia de los sistemas de procesamiento paralelo (SIMD), en este caso el paralelismo resulta de ejecutar concurrentemente trabajos independientes sobre conjuntos de datos separados. A esta categoría pertenecen los llamados sistemas de multiprocesamiento.

Existen dos características de interés para diferenciar los diseños de estos sistemas:

a) Acoplamiento o conmutación de procesadores, memorias y dispositivos de entrada/salida (I/O).

b) Homogeneidad de las unidades de procesamiento.

La sincronización y distribución de trabajos son indispensables para llevar a cabo un proceso. Esto contrasta con las organizaciones SIMD, donde la sincronización es automática al ejecutar las instrucciones y no hay problema-

de distribución de trabajos porque todos los procesadores - realizan la misma tarea (sobre datos diferentes).

MIMD se divide en 2 subcategorías:

1.2.3.1 Sistemas distribuidos o asimétricos

Consisten en múltiples procesadores donde cada uno ejecuta, exclusivamente, su tarea asignada. Esta distribución estática de trabajos es posible si se conoce con anterioridad el tipo, frecuencia de ocurrencia e importancia relativa de los mismos, para que las funciones del sistema -- queden repartidas entre los procesadores. Esto implica segmentar el programa en subrutinas que se asignan a cada procesador. Usualmente, la comunicación entre procesadores se restringe a transferir mensajes o bloques de datos a través de periféricos o memoria compartidos. Estos sistemas tienen - tres desventajas.

a) Raramente se conocen en forma explícita las características de la carga de un sistema.

b) La expansión del sistema (más trabajos o procesadores)- puede requerir una reconfiguración total del mismo.

c) La falla de cualquier módulo de procesamiento puede reducir gravemente el rendimiento del sistema.

1.2.3.2. Sistemas simétricos

Estos emplean un sistema operativo que distribuye- los trabajos dinámicamente a medida que se reciben. El sistema operativo puede ser distribuido y estar integrado en - cada módulo de procesamiento o concentrado en un procesador maestro, el cual distribuirá trabajos a procesadores escla-

vos (ref 6).

Los procesadores deben ser funcionalmente equivalentes y capaces de ejecutar cualquier trabajo que requiera el sistema para que un procesador inactivo se pueda utilizar en determinado momento, en cambio, en los sistemas asimétricos, hay que esperar o buscar un procesador entre los inactivos que sea capaz de ejecutar cierta tarea pendiente. La falla de un procesador ocaciona poca degradación en el rendimiento del sistema.

Los problemas de los sistemas simétricos son:

a) Se requiere que cada procesador tenga capacidad para ejecutar cualquier trabajo que le asigne el sistema operativo (incrementa el costo de los implementos electromecánicos).

b) Debe buscarse la manera óptima de reconfigurar el sistema.

c) Es necesario sincronizar los trabajos de los procesadores, ya que aun cuando cada uno ejecuta su propia tarea, en conjunto todos están destinados a resolver un problema común.

Los incisos b y c son bastante complicados y han requerido estudios detallados; sin embargo, se han propuesto algunas soluciones a estos problemas (ref 7-10).

1.3 ACOPLAMIENTO EN SISTEMAS DE MULTIPROCESAMIENTO

Los sistemas de multiprocesamiento (MPS) se clasifican según los grados de interacción entre procesadores:

1.3.1 Acoplamiento rígido (directo)

Opera bajo la supervisión de un esquema de control estricto. Por ejemplo, los sistemas CDC 6600, 6700, IBM -- S/360-65, IBM S/370-158MP, S/370-168MP, CDC-STAR-100, Te-- xas Instr. ASC, UNIVAC 1108, 1110, (ref 1, pp 17, 32, 36,- 40, 55, 80, 85 y Apéndices I, J, M, N, O).

Esta tendencia evidente solo en super computado-- ras es cuestionable, ya que se pueden tener sistemas con -- acoplamiento rígido basados en μPs. En la ref 11 se propo ne un procesador de Múltiples Funciones Unitarias con blo-- ques LSI (que bien podrían ser microprocesadores).

El acoplamiento rígido en un MPS se obtiene al - - efectuar las comunicaciones entre procesadores a través de una memoria compartida dividida en diferentes partes, en la que cada parte corresponde a una comunicación exclusiva en- tre dos procesadores (ref 12).

1.3.2 Acoplamiento flexible (indirecto)

No hay interacción entre procesadores acoplados -- flexiblemente aunque puedan compartir una memoria para comu nicarse. Cuando el sistema consiste en numerosos módulos - pequeños de carácter general es conveniente emplear conexio nes de tipo flexible; en este caso los procesadores deben - ser capaces de:

a) Accesar la memoria con mínimo de conflictos.

b) Comunicarse entre sí lo menos posible a través de recur sos compartidos.

c) Reconfigurar el sistema dinámicamente en caso de que falle algún procesador.

El acoplamiento flexible se utiliza en sistemas -- construidos con μPs para aplicaciones de carácter general - que requieren velocidad alta de procesamiento. Weissberger (ref 13) predice que estos sistemas eventualmente remplazarán a muchas minicomputadoras dedicadas al manejo de bases- de datos y al control en tiempo real; esta predicción se -- justifica por el siguiente razonamiento:

a) La flexibilidad inherente de los sistemas simétricos -- permiten pequeñas expansiones del sistema con un mínimo de diseño.

b) La habilidad de distribuir trabajos dinámicamente, para balancear la carga del procesamiento, mejora la velocidad de salida y la respuesta en tiempo real.

c) La reconfiguración permite al sistema dar preferencia a los trabajos más importantes y/o aceptar fallas tolerantes durante un proceso; por ejemplo, la falla de algún- procesador.

Las aplicaciones de un MPS basado en microprocesadores (μMPS) son: el manejo de bases de datos para sistemas interactivos (reservaciones aéreas, autorizaciones de - crédito, control de inventarios, historias clínicas, etc) o control en tiempo real (control industrial, comunicación de datos, reconocimiento de patrones y de voz, supervisión de- pacientes hospitalizados, etc).

Cualquier sistema factible de ser dividido en pequeñas labores individuales es candidato para ser realizado por medio de un μMPS.

1.4 ESTRUCTURAS DE INTERCONEXION

Es posible pensar en tres tipos principales de organización para la conmutación entre procesadores, memorias y dispositivos de I/O (ref 1, pp 28-39).

1.4.1 Sistemas de Barras cruzadas

Cualquier procesador o dispositivo de I/O se puede conectar a cualquier memoria durante el tiempo que dure la transferencia de datos a través de una matriz de barras cruzadas (fig 1.1).

Para p procesadores, n dispositivos de I/O y m memorias se requieren (p + n) m interruptores de barras cruzadas. Son posibles conexiones simultáneas si los elementos a conectar son mutuamente exclusivos, con lo que se puede producir muy alta velocidad de procesamiento del sistema.

En esta estructura la matriz de conmutación es necesariamente compleja y físicamente grande, por lo que representa el factor dominante en el costo del sistema. (Nótese que una línea en la matriz representa varios conductores de direcciones, datos y señales de control.)

Wulf y Bell (ref 14) proponen una matriz para conectar 16 PDP-11 ligeramente modificadas a 16 módulos de memoria.

1.4.2 Sistemas de memorias multipuerto

En esta estructura la conmutación se concentra en los módulos de memoria y cada procesador accesa las memorias a través de su propio canal (fig 1.2).

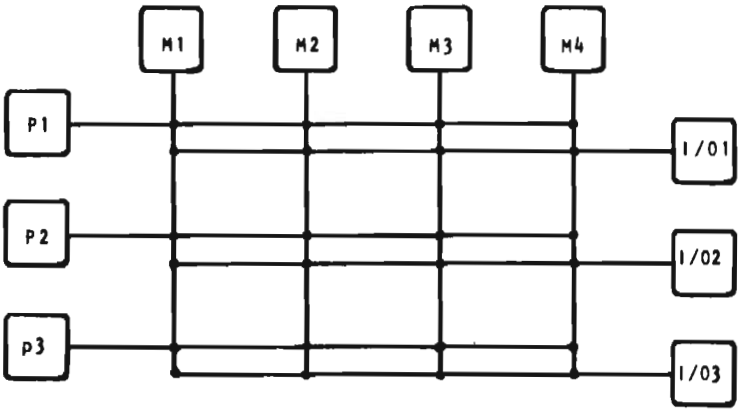
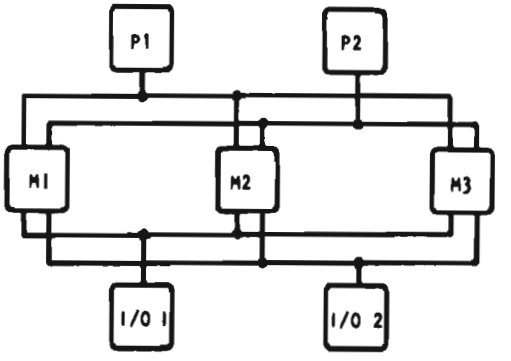


Fig 1.1 Estructura de barras cruzadas



P procesador
M memoria
I/O periférico

Fig 1.2 Estructura multipuerto

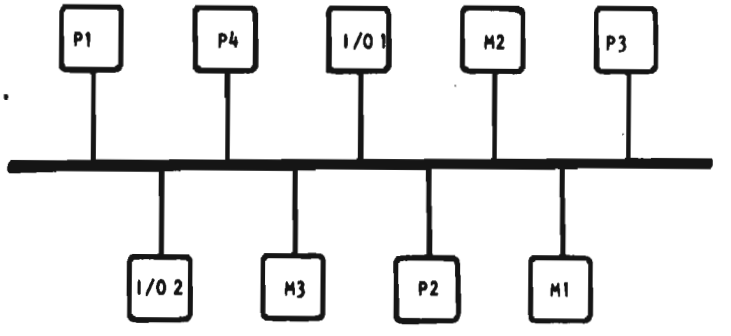


Fig 1.3 Estructura de canal único.

Debido a que los conectores y el cableado de interconexión resultan costosos, no es posible tener un elevado número de puertos en los módulos de memoria; por esto, el número restringido de puertos representa la mayor desventaja ya que equivale al número máximo de procesadores que se puede conectar en el sistema. Antes de la construcción debe definirse claramente el número máximo de procesadores a que se aspira.

1.4.3 Sistemas de canal único de tiempo compartido

Es la estructura más económica y sencilla de realizar así como la más versátil para reconfigurar el sistema. Consiste en un canal que comparten todos los procesadores, memorias y dispositivos de I/O (fig 1.3).

En este sistema no hay paralelismo en las comunicaciones ya que solo se puede efectuar una comunicación a la vez. El sistema, trabajando en forma concurrente, es muy efectivo y la circuitería requerida para añadir procesadores es de mucho menor costo que en las dos estructuras anteriores.

La velocidad de procesamiento del sistema está limitada por la capacidad de transferencia del canal. Sería catastrófico para este si llega a fallar el canal o si no lo suelta algún procesador.

Russo (ref 6) menciona el diseño de un comunicador entre procesadores COSMAC para un sistema simétrico (maestro-esclavos) utilizando un canal único de tiempo compartido. Reyling (ref 15) hace un análisis de rendimiento, control y costo de un μMPS asimétrico de canal único.

Se pueden obtener otras estructuras de intercone-

xión como resultado de combinar las tres anteriores.

1.5 ARBITRAJE EN UNA MEMORIA COMPARTIDA

Cuando se tiene un módulo de memoria compartida -- por varios procesadores, independientemente de la estructura en que se encuentren interconectados, es necesario disponer de un árbitro que regule el acceso a esa memoria. Los árbitros son dispositivos que sincronizan las solicitudes - de acceso a memoria hechas asíncronamente por los procesadores, por tanto, no permiten que dos o más procesadores accesen simultáneamente la memoria. Pueden ser básicamente dedos tipos.

1.5.1 Arbitro centralizado

Es un dispositivo que está totalmente integrado en un módulo desde donde regula el acceso de los procesadores- a memoria; las solicitudes de acceso deberan hacerse a este dispositivo. La capacidad del árbitro (número máximo de procesadores que puede manejar) queda claramente definida - al terminar el diseño del módulo, lo cual es una desventaja porque al sobrepasarla se presentan problemas costosos.

1.5.2 Arbitro decentralizado

En este tipo de árbitro la lógica que regula al -- acceso a memoria está distribuida en todos los elementos de procesamiento, asimismo, facilita construir el sistema en - forma modular y su capacidad es indefinida ya que crece enforma modular de acuerdo con el incremento del número de -- procesadores.

1.5.3 Implantacion del arbitro

Son tres las formas importantes para implantar un árbitro; enseguida se analizan estas.

1.5.3.1 Asignación de prioridades

Consiste en asignar a los procesadores y dispositivos de I/O prioridades de acceso a la memoria; es decir, si dos o más procesadores solicitan acceso simultáneamente, se concederá acceso a memoria al procesador que tenga mayor prioridad.

En este caso el mayor problema es determinar esa jerarquía, ya que los procesadores de mayor prioridad deben ser los que hacen menos solicitudes de acceso a memoria. Lo contrario ocasionaría que a procesadores de baja prioridad pocas veces se les conceda acceso a memoria.

1.5.3.2 Por orden de solicitud

Se lleva una lista de espera en la que se apuntan todos los procesadores que solicitan acceso, de esta manera el orden de acceso a memoria lo otorga dicha lista. Todo procesador que solicite acceso se apunta al final de la lista y se concede acceso a memoria al primero de la misma; terminado ese acceso se quita de la lista y ahora le toca (accesar) al siguiente procesador.

En este caso se evita el problema anterior de asignar un orden de prioridades a los procesadores; consecuentemente, todos tendrán la misma prioridad de acceso al canal. Sin embargo, dependerá del tamaño de la lista el tiempo que transcurre desde que se solicita el acceso hasta que se concede. El inconveniente en este caso es la realización y --

costo de esa lista de espera.

1.5.3.3 Por orden estático

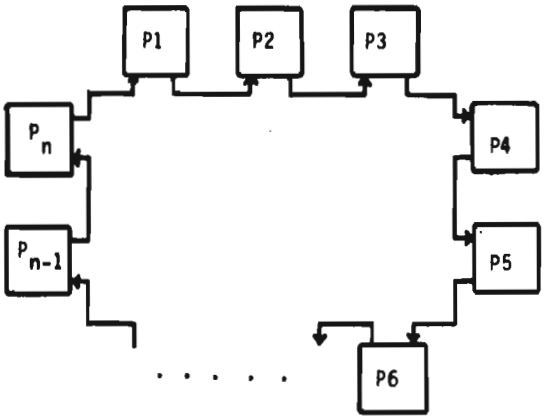
Consiste en ordenar el acceso de procesadores a memoria de acuerdo con la posición que ocupan, el árbitro en este caso es una señal que recorre todos los procesadores por orden estático y concede acceso a los que solicitaron y esperan hacerlo. Los procesadores no tienen que apuntarse en ninguna lista ni tener prioridad de acceso; es el método más sencillo.

El inconveniente en un arbitraje de este tipo es el tiempo que se pierde cuando existen muchos procesadores que accesan pocas veces, ya que el árbitro tarda cierto tiempo en decidir si un procesador solicitó o no acceso; si no hubo solicitud de acceso el árbitro pasa inmediatamente al siguiente procesador. Ese tiempo de decisión se pierde porque no se ocupa la memoria; si algún procesador solicita acceso y el árbitro tiene que recorrer varios procesadores que no lo hicieron, ese tiempo de espera es perdido para el sistema porque el procesador debe esperar su turno de todas maneras, aunque nadie ocupe la memoria.

El arbitraje por orden estático es fácilmente realizado en forma modular, por lo que generalmente es del tipo distribuido.

Lo más común es que el árbitro recorra los procesadores en forma circular, que consiste en organizar una trayectoria con las posiciones sucesivas que ocupan los procesadores y cerrar la trayectoria uniendo la última con la primera posición; o sea para que a un procesador le vuelva a tocar el turno (retorne el árbitro a esa posición) deberá esperar a que el árbitro recorra el resto de los procesado-

res (fig 1.4).



P procesador

Fig 1.4 Trayectoria circular del árbitro

1.6 REFERENCIAS

1. Enslow, P H, "Multiprocessors and parallel processing", Wiley-Interscience (1974).

2. Flynn, M J, "Very high speed computing system", Procs, IEEE, Vol 54 (dic 1966) pp 1901-1909.

3. Lorin, H, "Parallelism in hardware and software: Real and apparent concurrency", cap 21, Prentice-Hall, Engle-Wood Cliffs, NJ (1972).

4. Baer, J L, "Multiprocessing systems", IEEE Transac- - tions on Computers, (dic 1976), p. 1272.

5. Thurber, K J y Wald, L D, "Associative and parallel - processors", Comput. Surveys, Vol 7 (dic 1975), pp -- 215-255.

6. Russo, P M, "Interprocessor communication for multi-- microcomputer systems", Computer (abril 1977), pp 67-76.

7. Dijkstra, E W, "Solution of a problem in concurrent - programming control", Commun. A C M, Vol 8 (sep 1965), p. 569.

8. Lamport, L, "A new solution of Dijkstra's concurrent-programming problem", Commun. A C M, Vol 17 (ago - - 1976), pp 453-454.

9. Guzmán, A y Segovia, R, "A parallel configurable LISP machine", Comunicaciones Técnicas IIMAS, Vol 7, Nº 133, Serie naranja, México, D.F. (1976).

10. Steele, "Multiprocessing compactifying garbage collection", Commun A C M (sep 1975) pp 495-508.

11. Bowra, J L y Torng, H C, "The modelling and design of multiple function-unit processors", IEEE Transactions on Computers (mar 1976), pp 210-221.

12. Searle, B C y Freberg, D E, "Microprocessor aplications in multiple processor systems", Computer (oct 1975), p. 28.

13. Weissberger, A J, "Analysis of multiple-microprocessor system architectures", Computer Design, Vol 16, No 6 (jun 1977), p. 152.

14. Wulf, WAy Bell, C G, "C. mmp-A multimini processor", in 1972 Fall Joint Computer Conf., AFIPS Proc , Vol 41, Montvale, NJ, AFIPS Press, pp 765-777.

15. Reyling, G, "Performance and control of multiple microprocessor systems", Computer Design (mar 1974), pp 81-87.

2

SISTEMA DE MULTIPROCESAMIENTO

2.1 INTRODUCCION

En el capitulo 1 se clasificaron los sistemas computacionales de acuerdo con la secuencia de ejecución de -- instrucciones y adquisición de datos, con lo que se destaca la diferencia entre los sistemas de procesamiento paralelo- y de multiprocesamiento.

Los sistemas de procesamiento paralelo han sido explotados desde hace tiempo, prueba de ello es la variedad - de sistemas de este tipo, como Illiac IV, PEPE, etc (ref 1).

En sistemas de control basados en computadora, se- hace cada vez más necesario ejecutar en forma concurrente - procesos* destintos. La manera más sencilla y económica de realizarlos ha sido ejecutando concurrentemente, sobre un -

---

* Proceso: Un programa en ejecución.

solo procesador, varios de estos procesos (multiprograma- - ción).

En otros casos, la multiprogramación ha surgido para explotar al máximo el empleo de la unidad de procesamiento central (CPU). La ejecución de los procesos en forma secuencial ocasiona una utilización deficiente del CPU debido a la necesidad de tenerlo en espera de la ocurrencia de algún evento (tiempo desperdiciado). La ventaja de la multiprogramación es la disminución del tiempo desperdiciado por el CPU al intercalar la ejecución de dos o más programas en un solo procesador (ref 2).

Muchas veces es más importante terminar cierto número de trabajos en un periodo de tiempo determinado que la intensa utilización de los componentes (CPU, memoria y dispositivos de I/O) de una computadora (ref 2). Tal es el - caso de los sistemas de control por computadora, donde es - preciso añadir procesadores para poder ejecutar varios programas en forma simultánea y aumentar la velocidad de procesamiento (multiprocesamiento).

En multiprocesamiento, sin embargo, ciertos problemas de interacción entre procesos llegan a ser tan críticos que requieren de soluciones complejas. Como se mencionó en 1.2.3.1, la sincronización de trabajos es un problema serio debido a la concurrencia de procesos que se presenta con -- más intensidad que en multiprogramación (ref 3).

Un factor que había limitado hasta hace algunos -- años el desarrollo de los sistemas de multiprocesamiento -- era el costo de los procesadores. La aparición de minicomputadoras, primero, y de microprocesadores, después, han -- eliminado ese problema, y ocasionó que la investigación, en esta área, se enfocara hacia el desarrollo de sistemas ope-

rativos eficientes en el manejo de la interacción de los -- procesos.

Los sistemas de tiempo real que utilizan la memo-- ria dividida en múltiples partes, varios conjuntos de regis tros y un solo procesador, muestran decadencia a medida que el costo de los procesadores decrece. Sin lugar a dudas,- los sistemas de tiempo real de multiples procesadores serán utilizados ampliamente en un futuro no muy lejano (ref 4).

2.2 SISTEMAS OPERATIVOS PARA MULTIPROCESAMIENTO

Existen tres organizaciones básicas en el sistema- operativo para multiprocesamiento (ref 5), los cuales se - describen enseguida.

2.2.1 Maestro-Esclavo

La ejecución del sistema operativo está siempre - a cargo de un mismo procesador (llamado maestro). Los procesadores esclavos tienen a su cargo las rutinas de ejecu-- ción del programa común. No se presentan conflictos en el- acceso a datos comunes, ya que el encargado de proporcionar los a los procesadores esclavos es el maestro (una falla en el procesador maestro representaría una catástrofe para el- sistema). Los tiempos muertos de los procesadores esclavos (procesador con su ejecución detenida) se deben a que el -- procesador maestro no es suficientemente rápido para atender con eficacia todos los procesadores esclavos. Esta organización es efectiva para sistemas asimétricos.

2.2.2 Distribuido

En este caso la ejecución del sistema operativo es tá distribuida entre todos los procesadores del sistema; ca

da procesador tiene su propio conjunto de tablas y da servicio a sus necesidades. La falla de algún procesador no implica una catástrofe, sin embargo, puede ser difícil restaurar un procesador durante la operación del sistema. La reconfiguración de trabajos de los procesadores puede hacerse dinámicamente, por lo que resulta apropiado para sistemas - simétricos.

2.2.3 Flotante

En esta configuración el sistema operativo es flotante, es decir, cambia de uno a otro procesador, lo que -- permite balancear mejor la carga. Los conflictos en los requerimientos de servicios, usualmente, son resueltos por -- prioridades. Esta organización es efectiva para sistemas - simétricos.

2.3 MAQUINA DE ARQUITECTURA HETERARQUICA RECONFIGURABLE -- (AHR)

De los Sistemas de Multiprocesamiento más conocidos son los que se desarrollaron en la Universidad de Carnegie-Mellon, EUA: C. mmp y C.m*. En la Universidad de México - UNAM, se pretende desarrollar una máquina que ejecute concurrentemente programas en LISP, que sea heterarquica** y reconfigrable dinámicamente. Se utiliza el lenguaje LISP porque el orden (tiempo) de la evaluación no interesa, o está especificado por el usuario de manera inconsciente (ref-6, pp 4-5), además, facilita la interacción de los procesos y la utilización de un sistema operativo distribuido. Cada procesador ejecuta un programa y satisface sus propias necesidades. La reconfiguración ayuda a equilibrar la carga, -

---

** Unidades de procesamiento libres de jerarquía

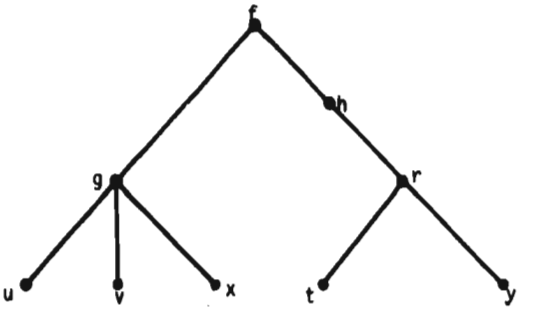
es decir, evita sobrecargar algunas unidades de procesamiento habiendo otras libres o menos ocupadas.

En la siguiente función descrita en LISP

f(g(u,v,x),h(r(t,y)))

las variables u, v, x, t, y pueden evaluarse en cualquier orden o simultáneamente, sin embargo, t y y deben ser evaluadas antes que r. Asimismo, h tiene que evaluarse después de r y, finalmente, f después de g y h. La regla es simple: antes de evaluar la función, se deben evaluar sus argumentos (ref 6, pp 4-5).

La función f puede representarse mediante el árbol de nodos siguiente



Los nodos (puntos) representan las funciones por evaluar; las ramas (líneas) conectan los nodos para formar el árbol. Se denomina copa al nodo que se encuentra en el extremo superior del árbol, y base a los nodos que se encuentran ligados a una sola rama.

Entre dos nodos conectados por una rama, uno ubica

do en un nivel más arriba que el otro, se denomina padre al nodo del nivel superior e hijo al del nivel inferior.

El orden de evaluación de la función f, mencionado anteriormente, queda expresado por el árbol en forma gráfica, en el que se evalúan primero los nodos de la base, luego los padres de estos y así sucesivamente siguiendo hacia-arriba hasta llegar a la copa (f). Se observa que entre -- más grande es la base del árbol se obtiene mayor número de-procesos en ejecución simultánea. La construcción del ár-- bol (creación de nodos) se realiza de arriba hacia abajo, - empezando por la copa y finalizando en la base.

De esto se concluye que expresar las funciones de-acuerdo con el lenguaje LISP evita que el programador tenga que especificar lo que se debe computar antes, simultánea-- mente o después (ref 6, pp 4-5).

2.3.1 Descripción

El modelo propuesto consiste en los elementos si-- guientes:

1. Unidades de procesamiento (UP). Cada unidad ejecuta - una operación primitiva del lenguaje LISP.

2. Memoria pasiva. Contiene programas y datos de gran tamaño, como datos de doble precisión, de punto flotante, etc.

3. Parrilla (memoria de ejecución). En la que se encuentran los programas que están en ejecución (pueden ser-uno o varios).

4. Pizarrones. Son listas de las funciones que deben ser evaluadas. Cada pizarrón corresponde a una función.

5. Memoria de variables. En la que se localiza el valor de las variables evaluadas.

La representación de un nodo se muestra en la fig 2.1 (ref 6, pág 10), en la que se tiene el nombre de la función, el número de argumentos no evaluados (NANE), los argumentos y apuntadores al padre, código (programa al que corresponde el nodo) y medio ambiente (intervalo de posiciones en la memoria de variables donde buscar el valor de estas).

2.3.1.1 Creación de los nodos

La compilación de los programas, es decir, la transformación de las expresiones en árboles de nodos se efectúa en la memoria pasiva. De esta memoria, algunos procesadores que crean y también ejecutan nodos, copian en la parrilla árboles o partes de ellos que al procesarse son destruidos. La copia en parrilla concluye cuando se llega a la base del árbol, donde los nodos tienen NANE = 0, lo que implica apuntarlos en el pizarrón correspondiente (fig 2.2). El original del árbol permanece en la memoria pasiva hasta que es evaluado totalmente.

2.3.1.2 Evaluación de los nodos

Las unidades de procesamiento buscan trabajo en los pizarrones constantemente (cada una en su respectivo pizarrón); cuando alguna lo encuentra, transfiere el nodo que le indica el pizarrón a su memoria privada y ejecuta la operación con los argumentos del mismo. Una vez evaluado el nodo, deja el resultado en el padre y decrementa en uno el NANE de este, que puede resultar:

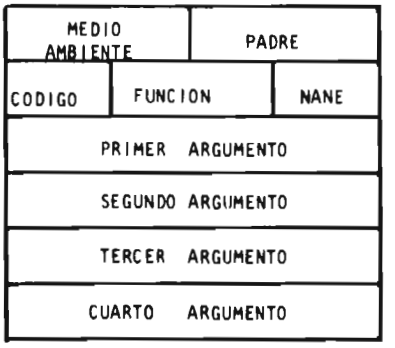


Fig 2.1 Representación de un nodo de cuatro argumentos

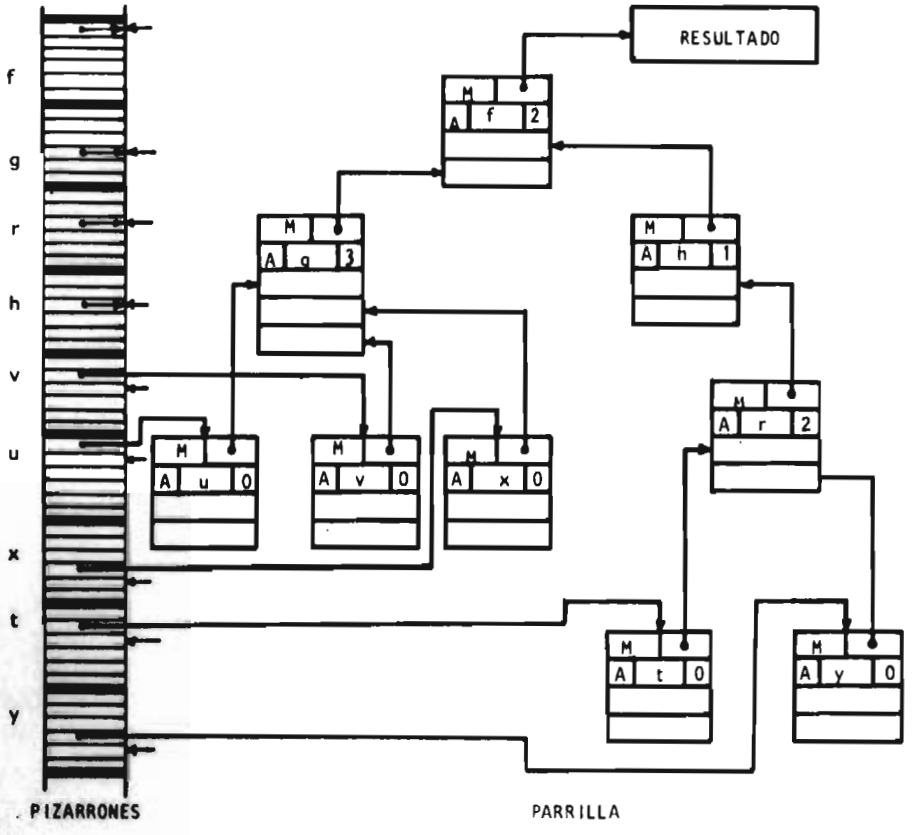


Fig 2.2 Representación de la función f en la parrillada y los Pizarrones.

NANE ≠ 0 Existe uno o varios argumentos no evaluados en el padre, por lo que la UP termina su función y se - dirige a buscar un nuevo trabajo.

NANE = 0 Todos los argumentos del padre fueron evaluados.- En este caso, la UP identifica la función del padre y lo apunta en el pizarrón correspondiente. - Termina, ahora sí, su función y se dirige a bus-- car un nuevo trabajo.

2.3.1.3 Reconfiguración

Existe una unidad de procesamiento cuya función -- única es reconfigurar dinámicamente la máquina (reconfigurador), la cual consiste en analizar los pizarrones y determinar el número de trabajos pendientes que hay en cada uno. - Dicho análisis le permite al reconfigurador saber qué funciones son de mayor o menor demanda. La función del reconfigurador es compensar la carga, es decir, procurar que el- número de trabajos pendientes sea igual en todos los piza-- rrones. Esto se logra cambiando la función de un procesa-- dor de menor demanda por una de mayor demanda.

El reconfigurador hace estos análisis cada ciertotiempo. La frecuencia de reconfiguración se determina se-- gún el tiempo que emplean las UP en ejecutar las funciones- y el número de cambios de función que pueda hacer por evento.

2.3.1.4 Entrada-Salida

La edición de un programa se puede hacer con ayuda de un procesador externo. Sin embargo, una alternativa es- que varias UP internas tengan editor y compilador, lo que - da grandes ventajas a la máquina, ya que podría editar, com

pilar y ejecutar diferentes programas simultáneamente, con-lo cual la máquina obtendría gran velocidad de procesamiento.

La salida de resultados e impresiones se puede - - efectuar a través de un procesador externo o utilizando alguna de las UP internas.

2.4 OTRAS PROPIEDADES DE LA MAQUINA

Como se mencionó al inicio de la descripción de la Máquina AHR, el sistema operativo es distribuido ya que está implícito en la ejecución de las funciones de cada procesador.

Las siguientes son algunas de las ventajas y características de la máquina (ref 6, pp 23-25).

a) Puede procesar cualquier lenguaje expresado en notación de cálculo lambda.

b) No requiere de un sistema operativo complejo.

c) Facilita a los usuarios la programación ya que no re- - quiere de etiquetas ni de la especificación de secuencias o simultaneidad de operaciones.

d) Nunca detiene su funcionamiento. Cuando un programa ha sido evaluado, las UP continúan buscando trabajo.

e) No hay control central, contador de programa, ni ciclo de máquina. La estructura es totalmente asíncrona.

f) No hay prioridades para la ejecución de los procesos.

g) No hay interrupciones a mitad de la ejecución de un programa.

h) Las UP no necesariamente son procesadores, pueden ser lógica discreta en combinación con otros dispositivos, -- por ejemplo, multiplicadores rápidos, etc. En estos casos esas UP no son reconfigurables.

i) Puede editar, compilar y ejecutar programas diferentes en forma simultánea.

2.5 REFERENCIAS

1. Hobbs,L C Associates, Inc, "Parallel processor systems, technologies and applications", Spartan Books, Nueva - - York (1970).

2. Donovan, J J,"Systems programming", McGraw-Hill Comp. - Science Series, Nueva York (1972), pp 366-400.

3. Lorin, H, "Parallelism in Hardware and software: Real- and apparent concurrency", Prentice-Hall, Englewood -- Cliffs, N J (1972), pp 224-25.

4. Korn,G A, "Microprocessors & small digital computer -- systems for engineers & scientists", McGraw-Hill, Inc,- Nueva York (1977), pág. 230

5. Enslow, P H, "Multiprocessors and parallel processing" Wiley Interscience, Nueva York (1974), pág. 81

6. Guzmán, A y Segovia, R, "A parallel configurable LISP - machine", Comunicaciones Técnicas IIMAS, Vol 7, Serie - Naranja, No 133, México, D.F (1976)

3

ACOPLAMIENTO DE VARIOS PROCESADORES A UNA MEMORIA

3.1 INTRODUCCION

De acuerdo con la descripción de la Máquina AHR -- (cap 2), esta se compone de varias unidades de procesamiento y cuatro memorias: memoria pasiva, memoria de variables, parrilla y pizarrones. Se presenta de inmediato el problema de acoplar las unidades de procesamiento a dichas memo-- rias, ya que todos los procesadores deben tener acceso a estas en forma concurrente.

Al respecto, el subcap 1.4, se presentan tres es-- quemas para conmutar memorias y dispositivos de I/O entre - procesadores. Las estructuras de barras cruzadas y canal - único de tiempo compartido se presentan como dos alternativas para la Máquina AHR. El tercer esquema que es el de memorias multipuerto se descarta debido al costo del mismo y- a la limitación de procesadores al número de puertos de la- memoria.

De las dos primeras, la estructura más económica - y sencilla es la de canal único de tiempo compartido, sin - embargo, es la más lenta debido a que no es posible efec- - tuar accesos simultáneos.

En este capítulo se trata el problema de conectar- varios procesadores a una memoria convencional* de una sola vía de acceso (un solo puerto) utilizando una estructura de- canal único; se discute un subsistema de protección de memo ria que permite a cada procesador proteger una posición - - cualquiera de memoria y se presenta un análisis del conges- tionamiento (varios procesadores tratando de utilizar memo- ria) del canal.

En el cap 5 se discute la manera de ampliar esta - estructura de canal único a una de varios canales. Cabe -- aclarar que la Maquina AHR puede utilizar cualquiera de es-- tos dos esquemas y la decisión de utilizar uno u otro depen de del compromiso costo vs velocidad de procesamiento.

3.2 ESTRUCTURA DE CANAL UNICO DE TIEMPO COMPARTIDO

Al conectar varios procesadores a una memoria de - una sola vía de acceso utilizando un canal único de tiempo- compartido, surge la necesidad de diseñar un dispositivo -- (acoplador) que permita que varios procesadores se comuni-- quen con la memoria.

Se propone el modelo que se muestra en la fig 3.1, el cual consiste en una memoria compartida por varios proce sadores. La conexión de los procesadores a la memoria com-

---

* Las cuatro memorias que requiere la Máquina AHR se pue-- den englobar en una sola.

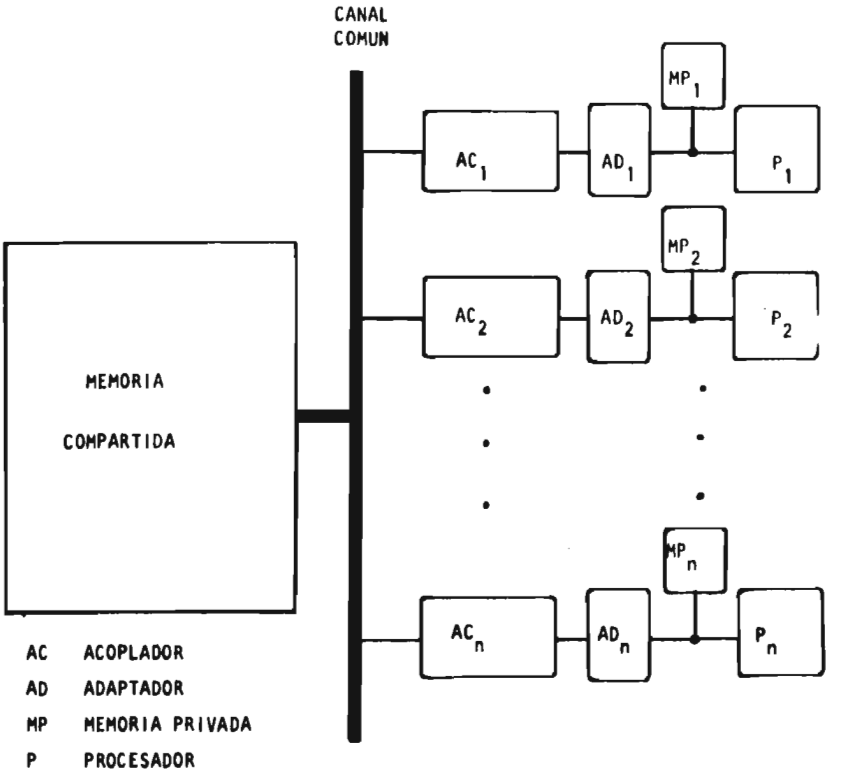


Fig 3.1 Estructura de canal único para un sistema de multiprocesamiento

partida se hace mediante un acoplador y un adaptador por -- procesador. Cada procesador puede tener, además, una memoria privada.

3.2.1 Acoplador

Es la parte central para las comunicaciones entre-procesador y memoria compartida; su diseño está destinado a cumplir con los siguientes aspectos:

1. Evitar restricciones de tipo lógico para conectar tantos procesadores como se requiera.
2. Poder utilizar el mayor número de procesadores diferentes.
3. Reducir al máximo los tiempos muertos entre accesos -- (tiempos en que debido a las funciones del árbitro no-es posible ocupar la memoria).
4. Obtener un diseño modular flexible del que fácilmente-se pueda pasar a un acoplador para una estructura de - barras cruzadas.
5. Construcción de bajo costo.
6. Su lógica no debe afectar la programación de los procesadores.
7. El diseño modular debe permitir que se añadan o eliminen procesadores con facilidad.
8. Todos los procesadores deben tener la misma prioridad-de acceso a memoria compartida.
9. Proteger una posición cualquiera de memoria.

3.2.2 Arbitraje

Se requiera un árbitro que solucione los conflic--

tos de acceso a memoria compartida. De acuerdo con lo que- se menciona en el subcap 1.5, el tipo de árbitro que se necesita es descentralizado, porque facilita la modularidad - del sistema.

Se utiliza, entonces, un arbitraje por orden estático debido a que no es deseable hacerlo por asignación de- prioridades. Un arbitraje por orden de solicitud resulta-- ría de costo alto y difícil de implantarlo modularmente. -- Las ventajas del arbitraje por orden estático son: costo - bajo y relativa facilidad de construcción. Una señal (pulso de rastreo) recorre secuencialmente los procesadores, -- que están organizados como una cadena circular, y permite - acceso a memoria compartida solo a aquellos que lo solici-- tan.

3.2.3 Adaptador

El diseño del acoplador no está sujeto a las características de acceso a memoria de ningún procesador. Por - este motivo es necesario un dispositivo que adapte (lógica- y eléctricamente) las señales del procesador con las del -- acoplador. El diseño del adaptador depende, entonces, de - cada procesador en particular.

3.2.4 Memorias compartida y privada

Con objeto de aumentar la velocidad de procesamiento del sistema, es conveniente utilizar la memoria compartida como memoria de datos comunes a los procesadores. Los - programas y algunos datos particulares se almacenan en la - memoria privada de cada procesador (ref 1, pág 84).

Las posiciones que ocupan las memorias privada y -

compartida deben pertenecer a la capacidad de direccionamiento del procesador. Un ejemplo de distribución de memorias, considerando cinco procesadores, es el que se muestra en la fig 3.2. Esta distribución indica que las direcciones de la memoria compartida son comunes para todos los procesadores, el límite inferior de la misma es 4000 (notación hexadecimal) y el superior 9FFF. La distribución de posiciones de memoria, para cualquier procesador, resulta

$$MTOT = MCOM \cup MPRIV \cup MDISP$$

MTOT Memoria Total (capacidad de direccionamiento).
MPRIV Memoria Privada.
MCOM Memoria Compartida
MDISP Memoria Disponible

donde:

$MCOM \neq \emptyset$
$MPRIV \neq \emptyset$
MDISP puede o no ser $\emptyset$
$\emptyset$ = conjunto vacío

3.2.5 Procesadores

Desde el punto de vista del acoplador, un procesador puede ser cualquier dispositivo que requiera de la memoria y con capacidad para esperar su turno de usarla. En el Apéndice B se encuentran clasificados algunos microprocesadores de acuerdo con esta propiedad.

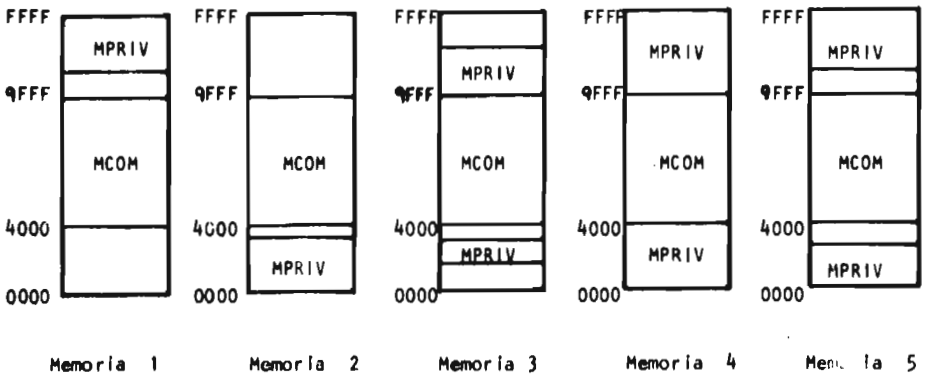


Fig 3.2 Distribución de memorias

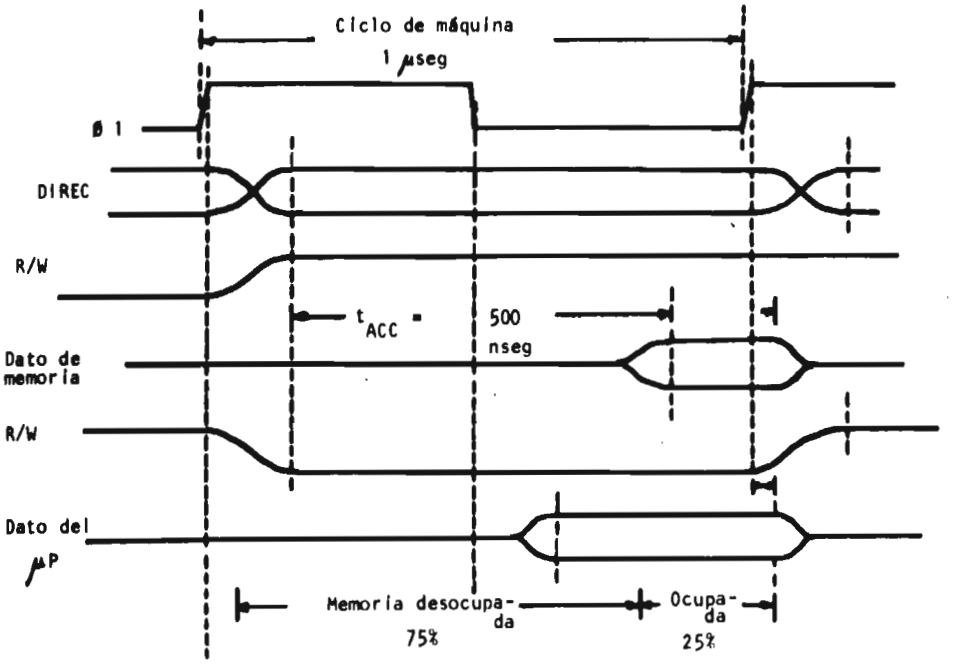


Fig 3.3 Ciclo de memoria del µP 6502

3.2.6 Tiempo de acceso a memoria compartida

Cada μP* ejecuta un programa que tiene almacenado en MPRIV, por este motivo los accesos a MCOM se efectúan solo en instrucciones que se refieren a datos de memoria -- (Apéndice C). Las instrucciones que no se relacionan con - datos de memoria no ocasionan accesos a MCOM, por ejemplo,- transferencias entre registros o con el acumulador, etc. -- Las instrucciones de saltos condicionales, incondicionales, a subrutinas y de subrutinas, tampoco utilizan MCOM.

La mayoría de las instrucciones que se relacionan- con datos de memoria se refieren una sola vez a MCOM, sin - embargo, hay otras que lo hacen dos (Apéndice D). Toda vez que un μP se refiere a MCOM lo hace en un ciclo de máquina. De acuerdo con el Apéndice D, en toda instrucción la rela-- ción: Referencias a MCOM/Ciclo de Máquina es siempre menor de uno, lo cual permite que varios μP utilicen MCOM durante el tiempo que dura la ejecución de una instrucción. Cada ciclo de máquina que realiza un μP con su MPRIV puede - ser utilizado por otro para llevar a cabo un acceso a MCOM.

Supóngase que el ciclo de máquina de un μP es de- 1 μseg y que el ciclo de la memoria compartida es de 0.25- μ seg (memoria de semiconductores). En este caso, se -- pierde 75 porciento del tiempo de utilización de la memoria. Como ejemplo obsérvense (fig 3.3) los tiempos que se deben- cumplir en un ciclo de máquina del μP MCS6502 (ref 2).

La ventaja en la rapidez de acceso que ofrece la - memoria de semiconductores y el bajo aprovechamiento de la- misma por los μPs permite pensar en un sistema de acceso -

---

* A partir de este momento, y sin perder generalidad, se - considera a las unidades de procesamiento como microprocesadores (μPs).

rápido a MCOM en el que, el μP permanezca en el canal exclusivamente el tiempo requerido para leer o escribir un dato en MCOM. Esto implica el uso de registros temporales que almacenan dirección, comando de lectura o escritura, datos de escritura y de lectura, para liberar a la memoria de la lentitud de los procesadores.

3.2.7 Subsistema de protección

En un sistema en el que varios procesadores comparten datos contenidos en una memoria, se define como proteger una posición de esta al hecho de obtener la exclusividad de su contenido por tiempo indefinido. Es indispensable que en este tipo de sistemas exista alguna forma de proteger posiciones de memoria, porque la ejecución concurrente de programas diferentes crea la posibilidad de que un procesador destruya datos de importancia para otro, o bienque el contenido de la posición sea erróneo debido a que un procesador toma alguna decisión con este.

Este último caso se presenta en la Máquina AHR, principalmente al decrementar NANE de un nodo (Sec. 2.3.1.2). Es decir, si dos μP leen en forma contigua el mismos NANEde un nodo, al decrementarlo, ambos almacenarán el mismo dato, y el NANE resultante disminuirá solo en uno y no en dos.

Ese problema se resuelve mediante un subsistema de protección, que permite al NANE ser decrementado por el primer μP que lo ocupe, evitando el acceso a esta posición de otro μP mientras el dato correcto (decrementado) no esté almacenado. El segundo μP puede decrementar NANE una vezalmacenado el resultado del primero. De esta manera, el NANE resultante es decrementado en dos.

Se consultaron varias referencias sobre la forma - de proteger una posición, pero los resultados obtenidos fueron de poca significación para los fines que se perseguían. Sin embargo, se encontraron algunas formas de protección a- través del sistema operativo, pero ninguna que se pudiera - implantar por un circuito de protección. Por ejemplo, C.m* utiliza un método sofisticado en que a un procesador le toma mucho tiempo proteger un bloque de memoria (ref 3). Enslow (ref 4) dice que la protección es un problema estrictamente de programación, aunque admite el desarrollo de algunos circuitos para solucionarlo.

Se puede decir, entonces, que lo más novedoso del- acoplador es la inclusión de un bloque protector que le permite al procesador indicar protección, por programa, de una posición cualquiera de MCOM. El contenido de esa posición- se protege sin necesidad de recurrir al sistema operativo.- Tampoco se requiere seleccionar un área para protecciones,- ya que cualquier procesador puede proteger cualquier posición de MCOM. El tiempo que se protege una posición es indefinido, así como el número de veces que se protege la misma u otras posiciones.

3.3 FUNCIONAMIENTO DEL ACOPLADOR

De acuerdo con lo planteado hasta aquí, el acoplador que permite conectar varios procesadores a una memoria- de un solo puerto tiene básicamente tres bloques (fig 3.4): Uno para comunicarse con el procesador (DETECTOR), otro para comunicarse con MCOM (ACCESO), y un tercero para proteger posiciones de memoria compartida (PROTECTOR).

El bloque DETECTOR identifica las direcciones que- se refieren a MCOM (conoce los límites superior e inferior-

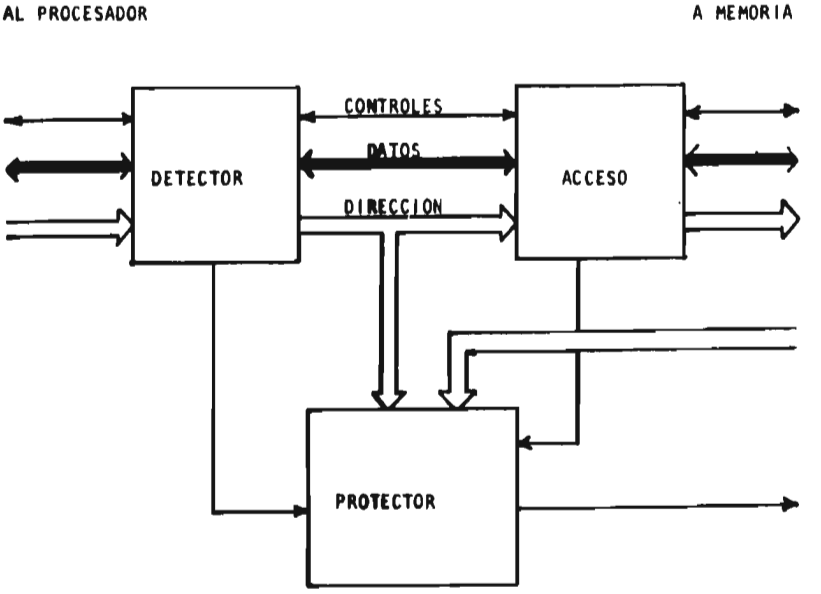


Fig 3.4 Diagrama de Bloques del Acoplador

de la misma) y tiene registros temporales para la dirección y dato de escritura. Sus funciones son: generar, cuando sea necesario, una señal (espera) que detenga el funcionamiento del μP; indicar al bloque ACCESO si hay solicitudes de acceso pendientes, y comunicar al bloque PROTECTOR la posición de MCOM a proteger.

El bloque ACCESO habilita la señal de acceso pendiente cada vez que el bloque DETECTOR se lo indica. Esta señal es la que detiene el pulso de rastreo para ejecutar el acceso requerido a MCOM. Tiene un registro temporal para lecturas que se hagan de MCOM y evita el acceso a posiciones protegidas.

El bloque PROTECTOR protege una posición de MCOM cuando el DETECTOR le ordena e indica la posición a proteger. Desprotege esa posición por orden del bloque ACCESO. Genera una señal, llamada protesta, cuando otro μP trata de accesar la posición que protege.

En seguida se describe el comportamiento del acoplador en lectura y escritura de MCOM y protección.

3.3.1 Escritura en memoria compartida

El bloque DETECTOR al identificar una dirección que se refiere a MCOM y el comando de escritura; los almacena juntamente con el dato a escribir en registros temporales y comunica al bloque ACCESO de una escritura para que este habite la señal de acceso pendiente. La escritura se realiza cuando el pulso de rastreo se lo permite (cuando llega a la posición que ocupa el acoplador). Terminada la escritura se deshabilita la señal de acceso pendiente y se transfiere el pulso de rastreo al siguiente acoplador. Debido a que el μP no ejecuta ninguna acción posterior a la

escritura, no es necesario detener su funcionamiento para - realizarla. La fig 3.5 (a) muestra estas secuencias representadas en un diagrama de tiempos.

3.3.2 Lectura de memoria compartida

Para obtener un lectura de MCOM es preciso detener el funcionamiento del μP, ya que este lee el dato después de que llega el pulso de rastreo. El bloque DTECTOR al - - identificar la dirección que se refiere a MCOM y el comando de lectura; los almacena, genera la señal que detiene el - funcionamiento del μP(espera) e indica al bloque ACCESO de una lectura pendiente. El μP se mantiene en estado de espera hasta que el pulso de rastreo le permita realizar la lectura.

El dato que entrega MCOM, resultante de la lectura, se almacena en el registro temporal de ACCESO. Terminada - la lectura se deshabilitan las señales acceso pendiente y - espera, así mismo, se transfiere el pulso de rastreo al siguiente acoplador. Deshabilitada la señal de espera el μP continua el transcurso de su programa y lee el dato que contiene el bloque ACCESO. En la fig 3.5 (b) se representan - las secuencias para lectura.

3.3.3 Escritura sobre escritura

El hecho de que un μP, al efectuar una escritura- en MCOM, deje dirección y dato en los registros temporales- del bloque DETECTOR y continúe con su programa, crea la posibilidad de que antes de llegar el pulso de rastreo a atender esa escritura el μP pretenda escribir nuevamente en -- MCOM. En este caso el bloque DETECTOR evita que el μP almacene dirección y dato de la nueva escritura generando la-

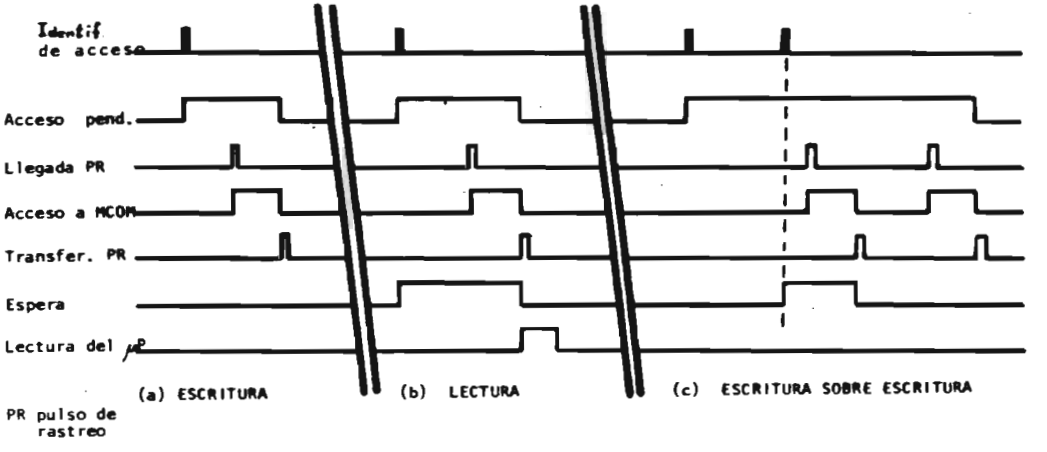


Fig 3.5 Secuencias del funcionamiento del acoplador

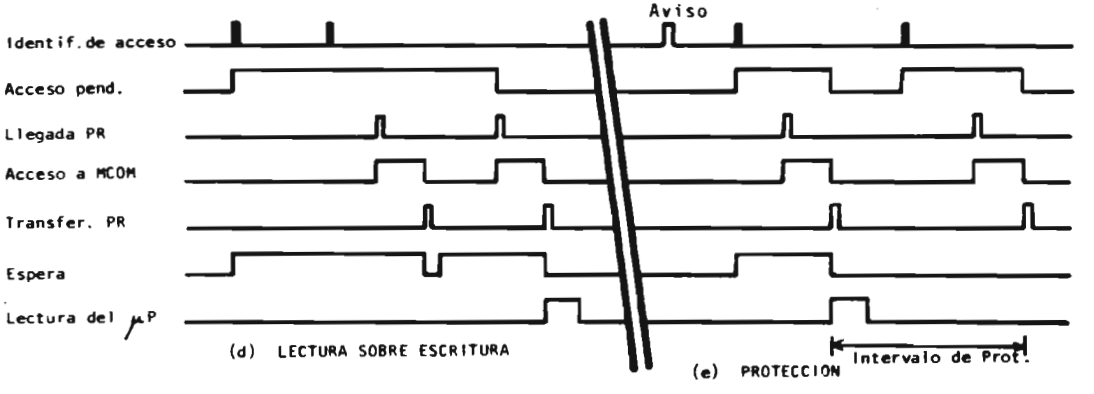


Fig 3.6 Secuencias de Lectura sobre escritura y protección

señal de espera que lo detiene hasta finalizar la primera.

Una vez hecha la escritura pendiente se quita la - señal de espera del μP para que deje dirección y dato de la nueva escritura en los registros de DETECTOR y pueda continuar con su programa. Esta escritura se llevará a efecto cuando el pulso de rastreo arribe nuevamente al acoplador.- En la fig 3.5 (c) están representadas estas secuencias.

3.3.4 Lectura sobre escritura

Lo mismo que en el caso anterior, el pulso de rastreo tarda en llegar a un acoplador en el que hay una escritura pendiente. Cuando el μP trata de hacer una lectura de MCOM, el bloque DETECTOR lo detiene hasta que se realiza la escritura pendiente. Una vez hecha, se quita la señal de - espera para que el μP deje la dirección de la lectura en -- los registros de DETECTOR. Este bloque nuevamente detiene- el funcionamiento del μP hasta el siguiente arribo del pulso de rastreo, con el cual se realiza la lectura. En la -- fig 3.6 (d) están representadas estas secuencias.

3.3.5 Proteccion de una posición de memoria compartida

Para proteger una posición de MCOM el μP debe seguir una secuencia que consiste en:

1o Aviso de protección
2o Primer acceso a MCOM, después del aviso:
(Acceso de Protección)
3o Segundo acceso a MCOM, después del aviso:
(Acceso de Desprotección)

La fig 3.7 representa estos tres pasos en un eje - de tiempo.

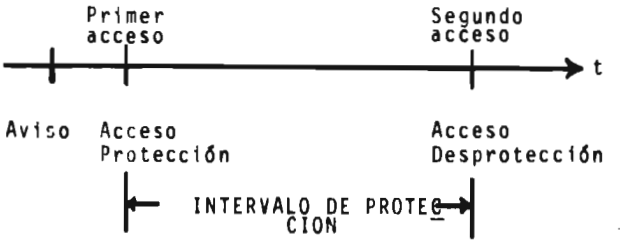


Fig. 3.7 Secuencia de protección

En la fig 3.6 (e) se muestran estos pasos para un- acceso de lectura y uno de escritura.

3.3.5.1 Cómo proteger

Existe una posición de memoria llamada "posición - de reservaciones" que no pertenece a MPRIV ni a MCOM. Tiene su propio decodificador en el acoplador y cada vez que - se direcciona equivale a indicar un aviso de protección. El tipo de acceso a la posición de reservaciones no interesa,- ya que puede ser de lectura o escritura. Este acceso no -- trasciende al canal compartido por lo que no se espera el- pulso de rastreo para realizarlo.

Una vez hecho el aviso de protección se activa el- bloque PROTECTOR el cual queda en espera del acceso de protección. Al realizarse este, la dirección se almacena en - los registros del bloque DETECTOR y se compara con las di-- recciones que se presentan en el canal hasta que ocurra el- acceso de desprotección, con el que se libera en forma auto

mática la posición protegida. El intervalo de tiempo entre los dos accesos a MCOM subsecuentes al aviso de protección- se denomina intervalo de protección. El bloque PROTECTOR - genera una señal (protesta) durante el intervalo de protección cada vez que otro μP trata de accesar la posición almacenada en el bloque DETECTOR.

3.3.5.2 Cómo evitar el acceso a una posición protegida

Cuando un μP(A) trata de accesar una posición que protege el μP(B), este último protesta de inmediato, señal que llega a todos los μPs pero es reconocida únicamente -- por el μP(A) que le obliga a quitar la dirección del canal- compartido. El μP(A) no logra realizar el acceso porque el tiempo que la dirección se mantiene en el canal es pequeño- en relación con el tiempo de acceso.

Ese acceso queda pendiente y posteriormente el μP (A) sigue insistiendo en realizarlo cada vez que le llega - el pulso de rastreo hasta lograrlo (cuando el μP(B) quita- la protección).

3.4 ANALISIS DE LAS CARACTERISTICAS DE ACCESO A MCOM

En un sistema de multiprocesamiento de canal único de tiempo compartido el acceso a MCOM está restringido por factores como: frecuencia de acceso de los μPs, tiempo que- se tarda MCOM en atender los accesos, etc; por tanto, es posible encontrar a MCOM libre (no ocupada por ningún μP) o - congestionada (muchos μPs esperando). En el primer caso, - la utilización de la MCOM es baja y su costo puede ser elevado respecto al provecho que se obtenga de ella. En el segundo, la velocidad de procesamiento se reduce debido al -- tiempo que espera cada μP antes de efectuar el acceso a -- MCOM.

Al presente, se ha realizado una serie de estudios a fin de optimar el uso de dispositivos compartidos en un sistema de multiprocesamiento, como son las memorias (ref-1, 5 y 6), llegándose a expresiones para determinar: número óptimo de procesadores, número de instrucciones ejecutadas por unidad de tiempo, su relación con el costo del sistema y modelos enfocados a determinar el congestionamiento de MCOM.

El objeto de este análisis es determinar una relación que mantenga MCOM a su máxima utilización, sin que esto implique un congestionamiento en el canal.

En un intervalo de tiempo t se tiene que:

$$t = tEJ + tW \qquad (1)$$

$$t = tOC + tLI \qquad (2)$$

donde

- t = intervalo de tiempo cualquiera
- tEJ = tiempo de ejecución de un μP, durante t
- tW = tiempo en espera de un μP, durante t
- tOC = tiempo que se ocupó MCOM, durante t
- tLI = tiempo que no se ocupó MCOM, durante t

Se define:

$$RMCOM = \text{Rendimiento de MCOM} = \frac{tOC}{t} \times 100\% \qquad (3)$$

$$RSIS = \text{Rendimiento del sistema} = \frac{\sum_{n=1}^{N} tEJ_n}{N \times t} \times 100\% \qquad (4)$$

donde

- N número de procesadores

RMCOM porcentaje del tiempo que está ocupada MCOM

RSIS porcentaje promedio de ejecución de los μPs

tEJ y tOC dependen de N, de la frecuencia de solicitudes de acceso que hagan los procesadores (f_p) y de la velocidad -- con que se atiendan esos accesos (frecuencia máxima de acceso de MCOM = $f_M$máx).

Si N y f_p son grandes y $f_M$máx es relativamente -- chica, entonces RSIS es bajo y RMCOM alto. Por lo contrario, si N y f_p son chicos y $f_M$máx es relativamente grande, entonces, RSIS es alto y RMCOM bajo. Se pretende que RSIS y RMCOM sean lo más alto posibles, esto equivale a decir -- que cada acceso se tarda lo menos posible en ejecutar y que MCOM se mantiene ocupada el mayor tiempo posible.

Dados un f_p y $f_M$máx hay un número η de μPs que logra que RSIS y RMCOM sean máximos, a este número η se le denomina número óptimo de μPs, para esas f_p y $f_M$máx.

Para determinar η se harán las siguientes hipótesis:

1. f_{Pn}máx es igual para n = 1,2,...,N
2. $f_M$máx $\gg f_p$máx
3. f_{Pn} es igual para n = 1,2,...,N
4. $f_M \gg f_p$ (en un caso cualquiera)
5. Cualquier solicitud de acceso se atiende de inmediato si no hay μPs esperando accesar o en-acceso.

como:

$$f_M\text{máx} = \frac{1}{t_{ACC}\text{mín}} \qquad (5)$$

donde

t_{ACC}mín = tiempo mínimo de acceso de MCOM

Para que RMCOM y RSIS sean máximos se debe solicitar acceso a MCOM a la frecuencia máxima que puede atenderlos, o sea:

$$f_M máx = \sum_{n=1}^{N} f_{Pn} \qquad (6)$$

pero como f_{Pn} es la misma para n = 1,2,...,N, se tiene

$$f_M máx = Nf_p \qquad (7)$$

donde

$$\eta = \frac{f_M máx}{f_p}$$

Al despejar N de la ecuación 7, se obtiene η . -- Existe un número óptimo η que es el mínimo obtenible y corresponde a $f_p máx$, llamado η^* (óptimo mínimo), o sea

$$\eta^* = \frac{f_M máx}{f_p máx} \qquad (8)$$

3.4.1 Comportamiento de RMCOM y RSIS

Es posible obtener una curva que muestre el comportamiento de f_M respecto a n y a una f_p dada. La velocidad de acceso a MCOM no debe sobrepasar $f_M máx$, por lo que la -- curva resulta:

Para $0 \leq n < \eta$ $\qquad f_M = nf_p$

Para $\eta \leq n < \infty$ $\qquad f_M = f_M máx$

Haciendo que $f_M máx$ corresponda al 100 porciento de RMCOM se obtiene la curva del comportamiento de RMCOM (fig-3.8). Para obtener la curva del comportamiento de RSIS se sigue el criterio siguiente:

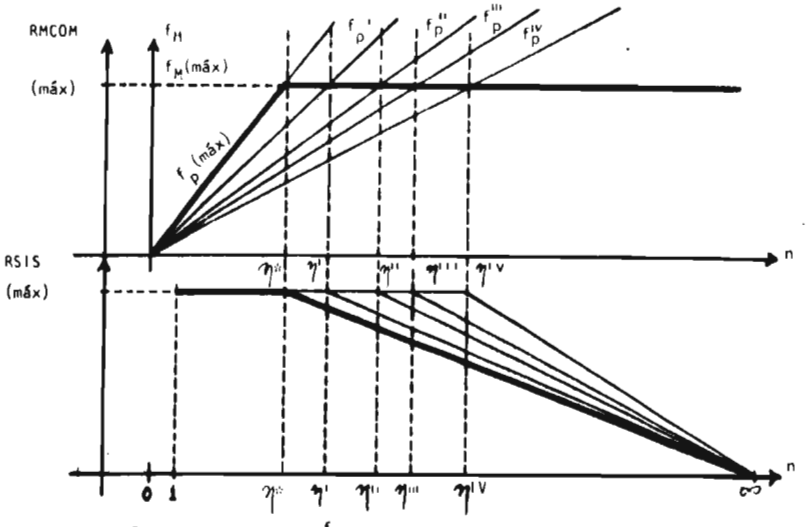


Fig 3.8 Comportamiento de f_M, RMCOM y RSIS respecto a n y f_p

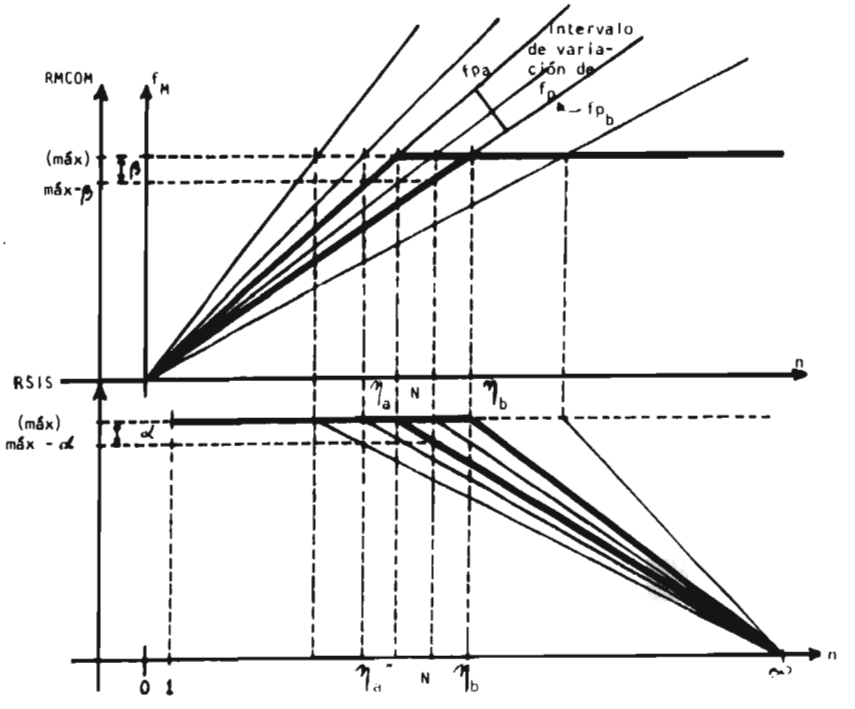


Fig 3.9 N apropiada para un intervalo de variación de f_p

Para $0 \leq n \leq \eta$ RSIS = 100%

Para $\eta < n < \infty$ 100% > RSIS > 0%

En el primer caso se cumple la hipótesis 5, en el segundo RSIS se decrementa a medida que n es cada vez mayor que η. El comportamiento de RSIS se muestra en la fig 3.8.

La fig 3.8 muestra una familia de curvas, una por cada f_p incluyendo $f_p$máx. El punto η en el eje horizontal es el único que logra que RSIS y RMCOM sean máximos para una f_p dada.

Obsérvese que el comportamiento de estas curvas es lineal, dado que sus ecuaciones son de primer grado; con este enfoque se obtiene una primera aproximación del comportamiento real de RSIS y RMCOM. El propósito del análisis no es obtener curvas precisas, sino mostrar la importancia de estas para futuros estudios de mayor detalle.

La importancia de las curvas radica en que es posible conocer el número óptimo de procesadores (N) conociendo $f_p$máx y el intervalo de variación de f_p, como se muestra en la fig 3.9, para asegurar variaciones de RSIS y RMCOM de α y β, respectivamente, abajo de los máximos permitidos.- También es posible conocer RSIS y RMCOM de un sistema cualquiera y las acciones que se deben tomar para optimarlo, o bien determinar la velocidad de acceso a memoria requerida ($f_M$máx) conociendo f_p y el número de procesadores.

3.5 REFERENCIAS

1. Reyling, G, "Performance and Control of multiple microprocessor systems", Computer Design (mar 1974), pp 81--87.

2. "MCS 6500 microcomputer family, hardware manual", MOS - Technology, Inc, Publication No 6500-10A, Norristown, - Filadelfia (ene 1976), pp 17-19.

3. Swan, R J, Fuller, S H y Siewiorek, D P, "The structure and architecture of Cm*: a modular multi-microprocessor", Computer Science Research Review 1975-1976 Carnegie-Mellon University, Pittsburgh, Pensilvania (sept 1976), - pp 25-47.

4. Enslow, P H, "Multiprocessors and parallel processing", Wiley-Interscience, Nueva York (1974), pp 98-99.

5. Bowra, J L, y Torng, H C, "The modelling and design of multiple function-Unit processors", IEEE Transactions - on Computers (mar 1976), pp 210-221

6. Hoogendoorn, C H, "A general model for memory interfe--rence in multiprocessors", IEEE Transactions on Compu--ters, Vol C-26, No 10 (oct 1977), pp 998-1005.

4

DESCRIPCION DETALLADA DEL ACOPLADOR

4.1 INTRODUCCION

El acoplador se ha representado en la fig 3.7 por-tres bloques generales: DETECTOR, ACCESO y PROTECTOR. El-objeto de este capítulo es presentar el diseño lógico -- del acoplador, el cual se realizó tomando como base esos -- tres bloques principales; para ello se dividieron las funciones de los mismos en subfunciones más sencillas y fáci-- les de implantarlas lógicamente.

Primero se definen las señales que relacionan acoplador con MCOM y con procesador; posteriormente se dividen en subloques los tres principales definiendo las funciones-de cada uno de estos. Finalmente se lleva a cabo el diseño lógico de estos bloques.

4.2 ESPECIFICACION DE LAS LINEAS DE DIRECCION, DATOS Y CONTROL

El acoplador es un intercambiador de información -

entre el procesador y MCOM, por tanto, tiene dos vías bidireccionales de comunicación (fig 4.1), formadas cada una -- por un grupo de líneas. Además, existe un tercer grupo de líneas que conecta el acoplador con acopladores vecinos para formar la cadena circular que determina el orden en que los procesadores deben tener acceso a MCOM.

La fig 4.1 muestra un diagrama general del acoplador, donde aparecen los bloques DETECTOR y ACCESO formado por bloques más pequeños para identificar los dos grupos de líneas pertenecientes a las vías (o puertos) y al grupo de líneas que conecta los acopladores entre sí.

4.2.1 Líneas de interconexión con el adaptador

Las líneas a la izquierda del diagrama de la fig - 4.1 corresponden al puerto que interconecta acoplador y - - adaptador; entre ellas, se distinguen líneas de dirección,- datos y control.

El número de líneas de dirección depende de la capacidad de direccionamiento del procesador. Muchos micro-- procesadores (μPs) y minicomputadoras direccionan con 16- líneas hasta 64K posiciones de memoria (1K = 1024), motivo- por el que se adoptó este criterio para el acoplador.

El número de líneas de datos depende del tamaño de la palabra de información que maneje el procesador. Actualmente, la mayoría de μPs tiene 8 líneas de datos bidireccionales, y 16 las minicomputadoras. Debido al bajo costo- y alta confiabilidad de los μPs, se pretende utilizarlos - como unidades de procesamiento en la máquina AHR, motivo - por el que se optó por 8 líneas; sin embargo, en el siguiente capítulo se estudia la manera de ampliarlas a 16 para po

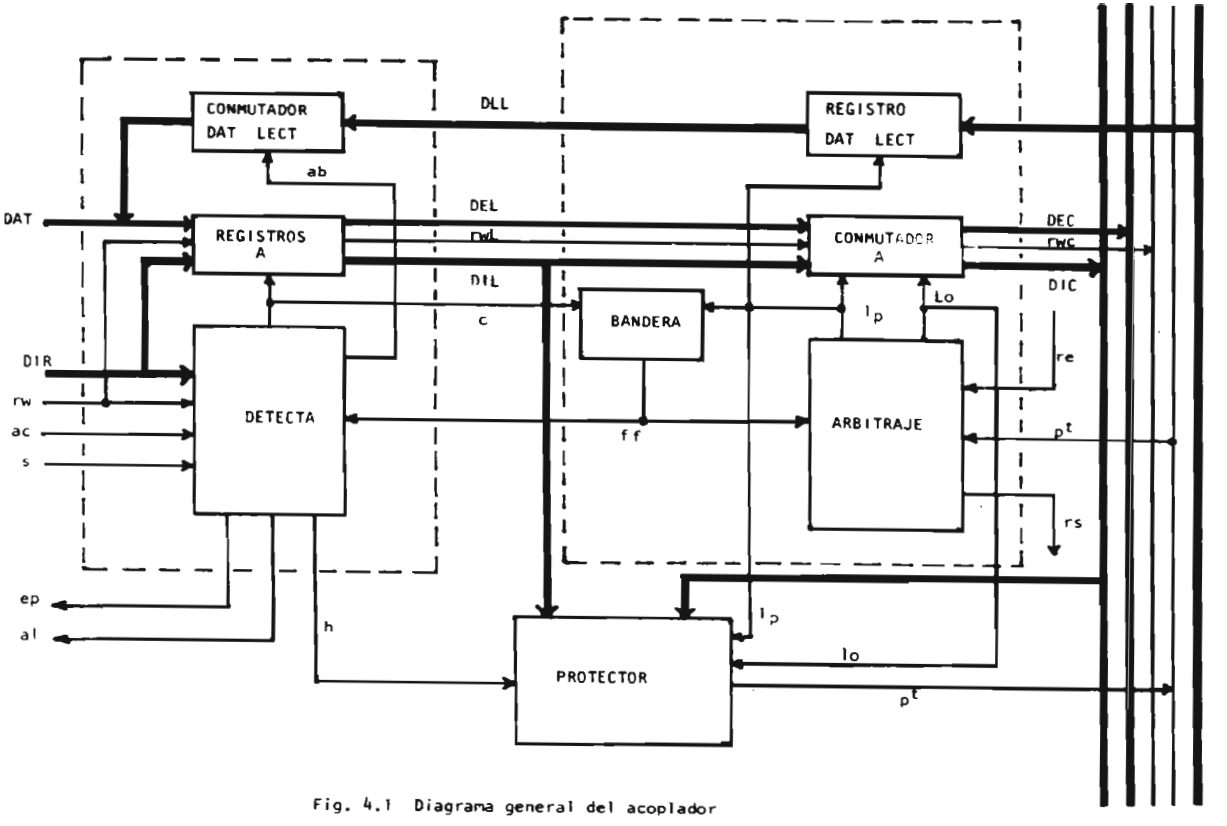


Fig. 4.1 Diagrama general del acoplador

der emplear micro o minicomputadoras que utilizan 16 líneas de datos trasmitidos en dos grupos de 8. Las líneas de control son:

rw R/W, especifica la ejecución de una lectura o escritura en memoria por parte del procesador

ac acceso, señal que debe ser generada por el adaptador y sirve para indicar el momento exacto en que la dirección R/W y dato (en caso de escritura) de un acceso a MCOM se encuentran estables en el canal del procesador

s señal que indica los intervalos de tiempo permitidos para deshabilitar la señal que detiene al procesador

ep señal de espera, detiene la ejecución del programa que sigue el procesador

al señal de alarma, indica al procesador y/o al usuario que hay un retraso demasiado largo en la ejecución de un acceso a MCOM

4.2.2 Líneas de interconexión con MCOM

Corresponden al puerto que interconecta acoplador y MCOM, se forman, también, por líneas de dirección, datos y control (fig 4.1, derecha).

En algunas memorias convencionales las líneas de datos de entrada y las de salida están separadas, mientras que en otras son bidireccionales. El acoplador se diseñó para memorias del primer tipo.

Las líneas de control son:

rwc R/W, señal que determina accesos a MCOM de escritura o lectura

pt protesta, esta señal evita que un procesador tenga - - acceso a una posición protegida

4.2.3 Líneas de interconexión entre acopladores

Los acopladores forman una trayectoria circular para el pulso de rastreo, cada uno tiene dos líneas que lo conectan con sus acopladores vecinos:

re línea por donde llega el pulso de rastreo del acoplador anterior

rs Línea por donde se transfiere ese pulso al acoplador - siguiente

4.3 DESCRIPCION DEL ACOPLADOR A NIVEL DE BLOQUES

En esta sección se describen las funciones de las-cajas de la fig 4.1 mediante bloques de funciones más sim-- ples.

4.3.1 Detector

Sirve para comunicarse con el adaptador del procesador; sus tres cajas DETECTA, REGISTROS A y CONMUTADOR DATOS LECTURA (fig 4.1) se dividen en bloques más pequeños cuya distribución y relación se muestra en la fig 4.2. A continuación se describe la función de cada uno de estos bloques.

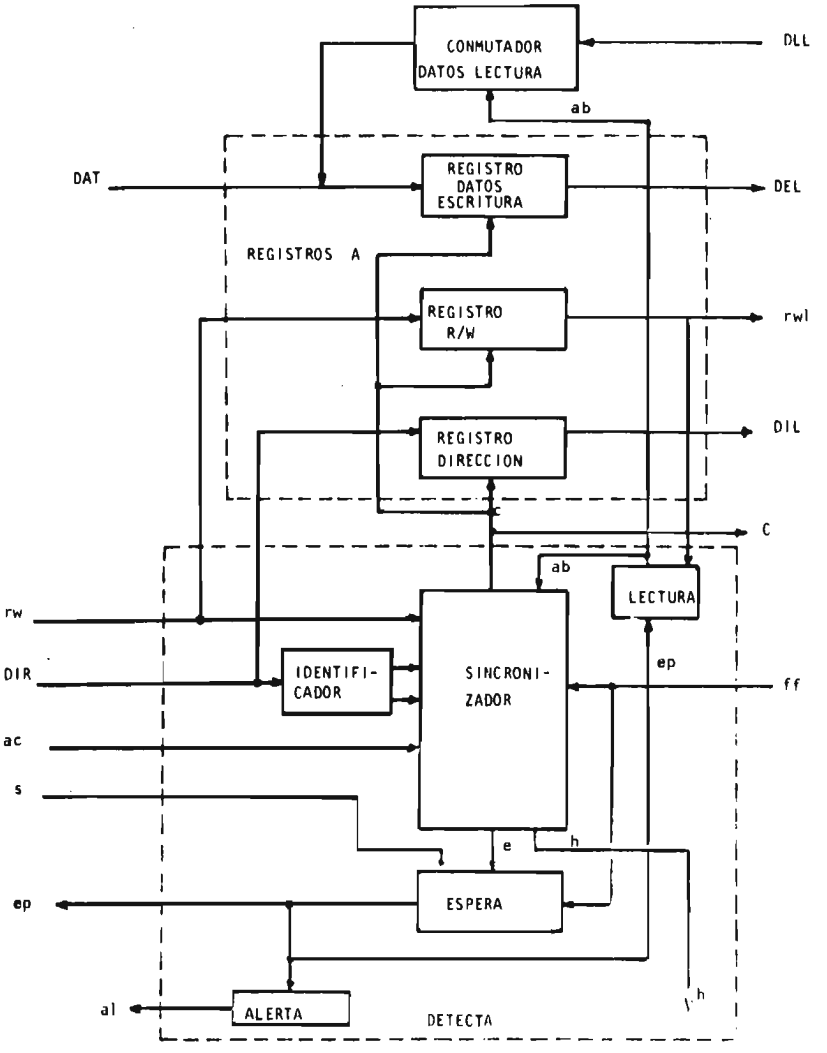


Fig 4.2 Diagrama de bloques de DETECTOR

4.3.1.1 Identificador

Entrada: vector DIR (dirección generada por el procesador); salidas: ad y rv. La función de ad es identificar las direcciones que se refieren a MCOM comparando DIR - con el límite inferior (dirección menos significante = LI)- y con el límite superior (dirección más significante = LS)- de la misma, y la de rv es identificar la dirección de la - posición de reservaciones (PR) comparándola con DIR.

4.3.1.2 Sincronizador

Entradas: ac, rw, ad, rv, ff (señal de acceso pendiente) y ab (señal que le permite al procesador leer datos del acoplador); salidas: c, h y e. La función de c es indicar un acceso pendiente y comandar los registros para almacenar dirección, R/W y dato (en caso de escritura). La - función de h es indicar un aviso de protección, que se produce cuando se direcciona PR, y la de e es excitar el blo-- que ESPERA para que detenga el funcionamiento del procesa-- dor.

4.3.1.3 Espera

Entradas: e, s y ff; salida: ep. La función de ep es detener la ejecución del procesador.

4.3.1.4 Alerta

Entrada: ep; salida: al. La función de al es prevenir de cualquier falla eléctrica al procesador y/o al - - usuario si ep permanece habilitada un tiempo arbitrariamente grande para el procesador.

4.3.1.5 Lectura

Entradas: ep y rwl (señal que indica si el anterior acceso a MCOM fue de lectura o escritura); salida: ab. La función de ab es habilitar el bloque CONMUTADOR DATOS LECTURA para que el procesador lea los datos obtenidos de las lecturas de MCOM.

4.3.1.6 Registro de dirección

Entradas: vector DIR y c; salida: vector DIL. La función del bloque es almacenar las direcciones generadas por el procesador que se refieren a MCOM.

4.3.1.7 Registro de R/W

Entradas: rw y c; salida: rwl. La función del bloque es almacenar el comando de lectura o escritura para todo acceso a MCOM.

4.3.1.8 Registro de datos de escritura

Entradas: vector DAT (dato de escritura generado por el procesador) y c; salida: vector DEL. La función del bloque es almacenar los datos que pretende escribir el procesador en MCOM.

4.3.1.9 Conmutador datos lectura

Entradas: vector DLL (dato leído de MCOM) y ab; salida: vector DAT. La función del bloque es mantener, el tiempo que indica ab, el contenido de DLL en DAT para que el procesador lea los datos obtenidos de MCOM.

4.3.2 Acceso

Sirve para comunicarse con MCOM, sus cuatro cajas: BANDERA, ARBITRAJE, CONMUTADOR A y REGISTRO DATOS LECTURA - (fig 4.1) se dividen en bloques aún más pequeños cuya dis-- tribución y relación se muestra en la fig 4.3. A continuación se describen las funciones de estos bloques.

4.3.2.1 Bandera

Entradas: c y lp (señal que permite el acceso a -- MCOM); salida: ff. La función de ff es indicar la existencia de un acceso pendiente para detener el pulso de rastreo cuando llegue a este acoplador.

4.3.2.2 Rastreador

Entradas: ff, re y no (señal que indica acceso evitado a causa de una protesta); salidas: y y rs. La función - de y es iniciar la secuencia para permitir el acceso a MCOM cada vez que llegue un pulso de rastreo (pulso en re) cuando hay un acceso pendiente (ff habilitada), y la de rs es- transferir el pulso de rastreo al siguiente acoplador cuando: a) llega un pulso de rastreo y no hay acceso pendiente, b) después de un acceso a MCOM (cuando se deshabilita ff)- y c) se evita el acceso a causa de una protesta (pulso en - no).

4.3.2.3 Tiempo

Entrada: y: salida: ta. La función del bloque es producir un pulso en ta de duración $t = t_{ACC} + t_{PROT}$ cada - vez que lo indique y.

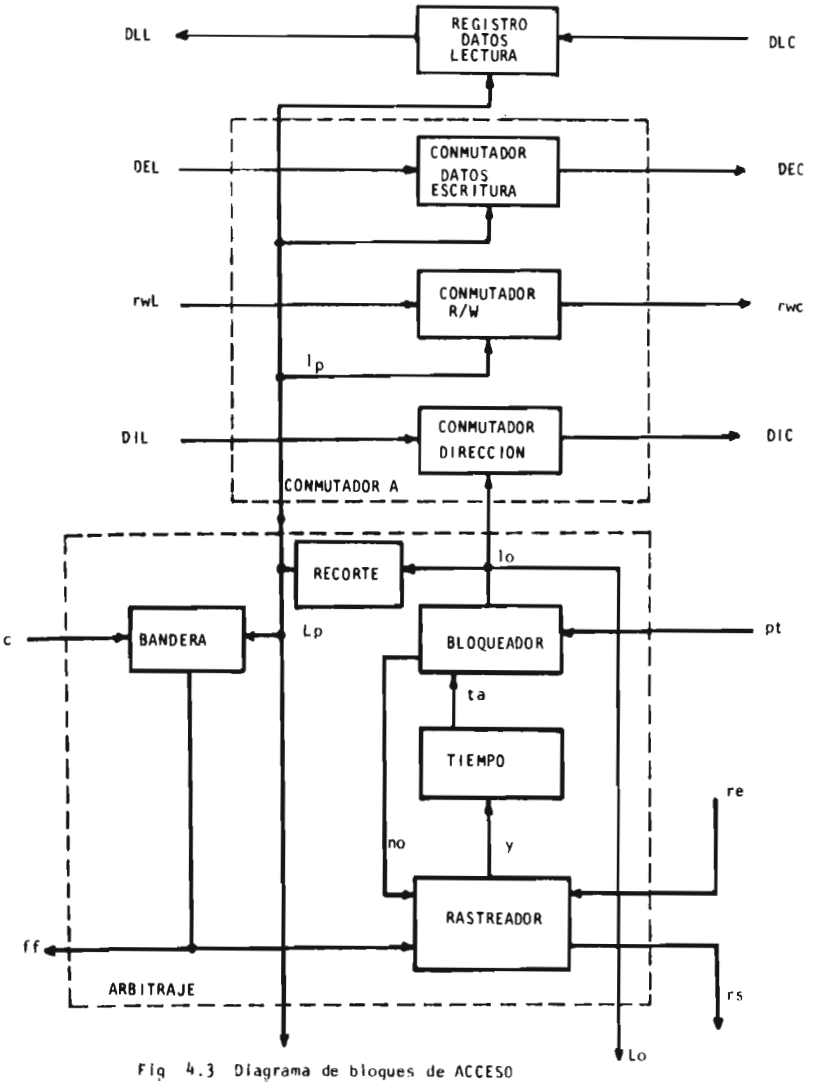


Fig 4.3 Diagrama de bloques de ACCESO

t_{ACC} tiempo del ciclo de MCOM

t_{PROT} tiempo de comparación (necesario para que -- otro acoplador proteste por la dirección que se trata de tener acceso)

4.3.2.4 Bloqueador

Entradas: ta y pt; salidas: lo y no. la función lo es permitir colocar la dirección del acceso en el canal común el tiempo ta si no hay protecta por esta dirección - (no hay pulso en pt). En caso de haber protesta se deshabilita lo y con ello se bloquea el acceso, indicándolo me-- diante un pulso en no.

4.3.2.5 Recorte

Entrada: lo; salida: lp. La función de lp es permitir el acceso a MCOM el tiempo t_{ACC} cuando no haya protesta por la dirección que se trata de ocupar (en caso de protesta esta se presenta durante el tiempo t_{PROT} no dejando- que se habilite lp).

4.3.2.6 Conmutador de direcciones

Entradas: vector DIL y lo; salida: vector DIC. La función del bloque es mantener, el tiempo que indica lo, el contenido de DIL en el canal común de direcciones (DIC).

4.3.2.7 Conmutador de R/W

Entradas: rwl y lp; salida: rwc. La función del -- bloque es mantener, el tiempo que indica lp, el estado de- rwl en el canal común de R/W (rwc).

4.3.2.8 Conmutador de datos de escritura

Entradas: vector DEL y lp; salida: vector DEC. La función del bloque es mantener, el tiempo que indica lp, el contenido de DEL en el canal común de datos de escritura a-MCOM (DEC).

4.3.2.9 Registro de datos de lectura

Entradas: vector DLC (contenido del canal común de datos de lectura de MCOM) y lp; salida: vector DLL. La función del bloque es almacenar los datos que se leen de MCOM.

4.3.3 Protector

Sirve para proteger una posición de MCOM. La secuencia de protección y el funcionamiento del mismo se detallaron en 3.3.5. Su configuración se muestra en la fig 4.4 dividida en cuatro bloques que se describen a continuación.

4.3.3.1 Aviso

Entradas: h y qi (señal para quitar el aviso de protección); salida: bt. La función del bloque es recibir y mantener pendiente el aviso de protección hasta que se realice el acceso del mismo nombre.

4.3.3.2 Habilita

Entradas: bt y lp; salidas: z y qi. La función del bloque es habilitar la señal de protección (z) durante el intervalo del mismo nombre y habilitar qi al inicio de la habilitación de z.

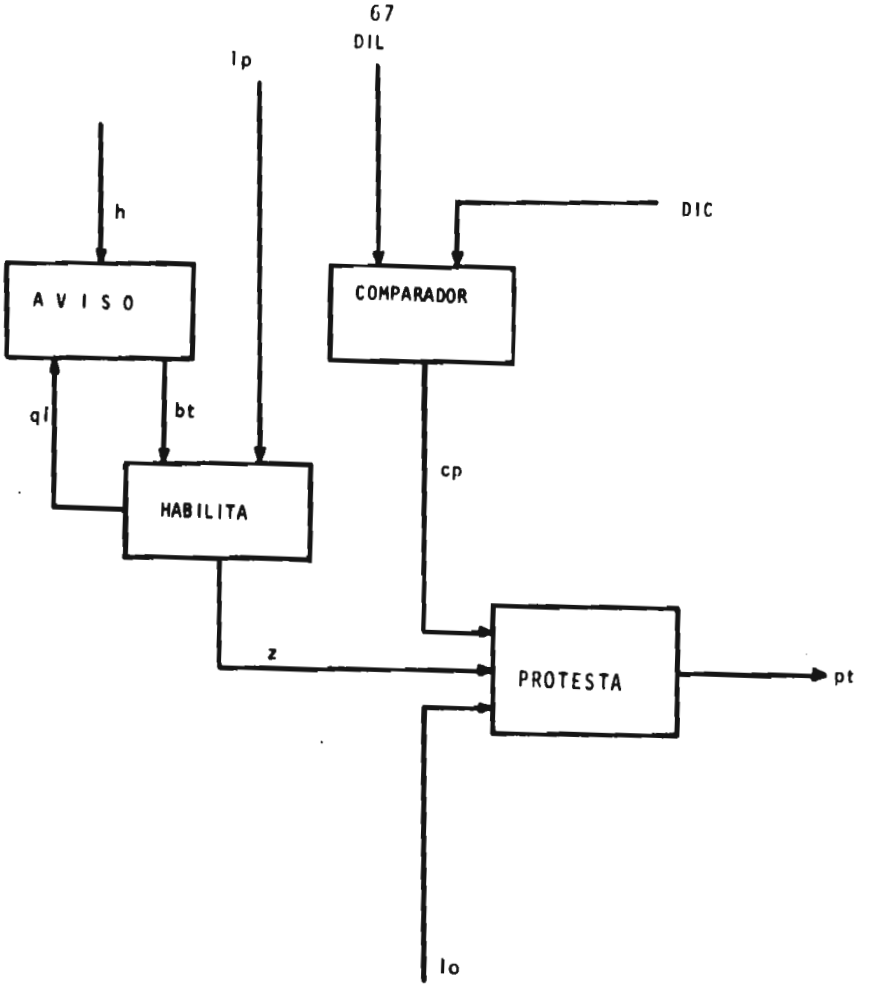


Fig. 4.4 Diagrama de bloques del PROTECTOR

4.3.3.3 Comparador

Entradas: los vectores DIL y DIC; salida: cp. La función del bloque es comparar los vectores DIL y DIC e indicar en cp cuando sean iguales.

4.3.3.4 Protesta

Entradas: cp, z y lo; salida pt. La función del bloque es protestar (generar un pulso en pt) cuando DIL = DIC (cp habilitada) durante el intervalo de protección (z habilitada), excepto que el acceso sea del mismo procesador (lo deshabilitada).

En el diagrama de bloques del acoplador (fig 4.5) se presentan integrados los tres diagramas antes estudiados.

4.4 DISEÑO LOGICO DEL ACOPLADOR

Esta sección se dedica al diseño lógico de los bloques descritos.

Se decidio utilizar la familia lógica TTL debido a su versatilidad de funciones integradas, confiabilidad y economía (ref 1, cap 2), además de que es la más popular en México.

Todos los bloques anteriormente mencionados serán diseñados con la serie normal 74 de circuitos integrados TTL (ref 1, pp 16, 17) la cual ofrece una combinación adecuada de velocidad (tiempo de respuesta) y consumo de energía para la mayoría de los bloques. En algunos casos es conveniente utilizar dispositivos de otra serie para obtener diferente velocidad o menor consumo de energía; se reco

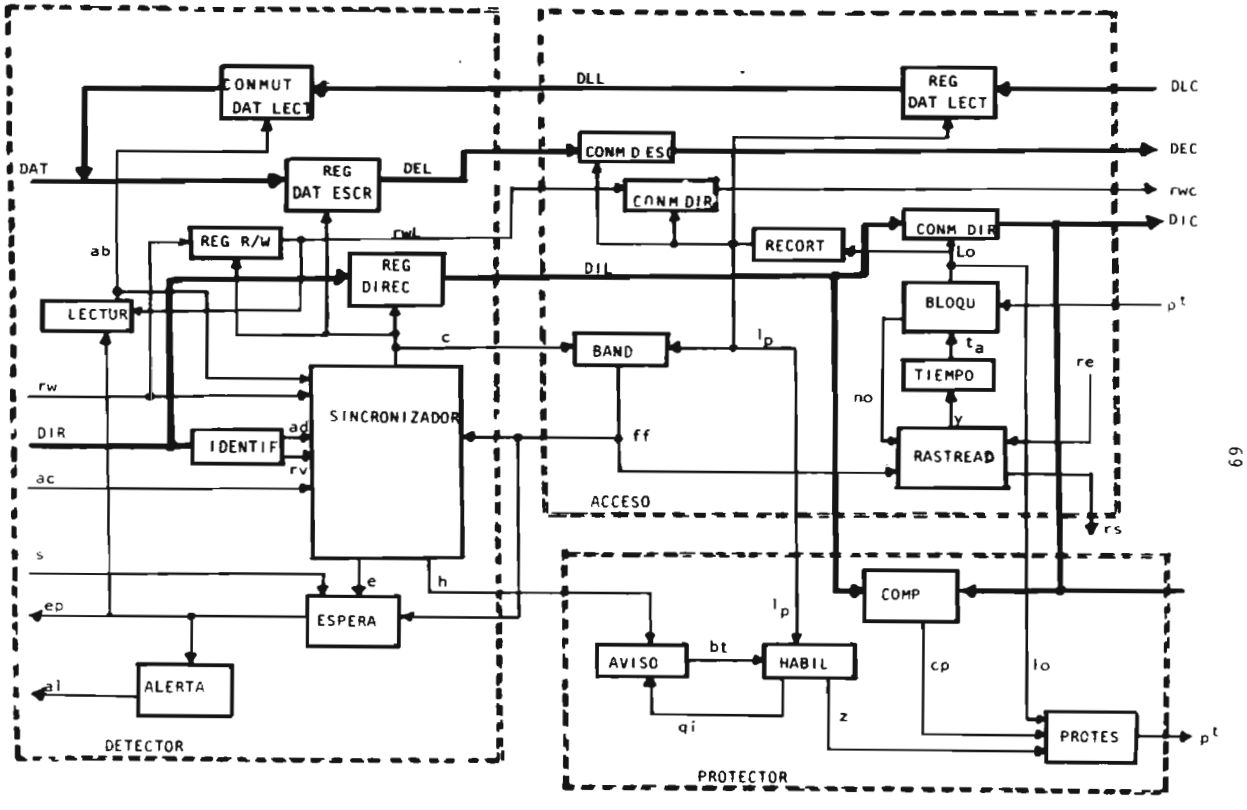


Fig 4.5 Diagrama detallado del acoplador

mendará la serie a utilizar cada vez que se presenten estos casos.

4.4.1 Detector

Los estados lógicos de las señales que relacionan el acoplador con el adaptador del procesador definen lo siguiente:

rw = 0 escritura en memoria
rw = 1 lectura de memoria
ac = 0 no hay solicitud de acceso
ac = 1 aviso de acceso
$\overline{ep}$ = 0 procesador detenido
$\overline{ep}$ = 1 procesador en ejecución
$\overline{al}$ = 0 alarma habilitada
$\overline{al}$ = 1 alarma deshabilitada
s = 0 no se permite quitar ep
s = 1 se puede quitar ep

La señal ac es la única que tiene características especiales que no corresponden directamente con ninguna señal generada por el procesador, debe ser un pulso producido por el adaptador que indica el momento exacto en que la dirección, comando de escritura o lectura y dato (en caso de escritura) se encuentran estables en la vía (canal) de acceso a memoria. Debe ser posible detener la ejecución del procesador después del pulso de ac, ya que en muchas ocasiones hace que se genere la señal de espera ep.

Debe tenerse cuidado con la carga que maneja el procesador en sus líneas de dirección, R/W y dato, puesto que el acoplador demanda corriente en estas equivalente a varias cargas TTL. Si la corriente que maneja el procesa-

dor en esas líneas no es suficiente, habrá que añadir amplificadores de corriente (buffers).

4.4.1.1 Identificador

Se requieren dos comparadores, uno que compare la dirección del μP(DIR) con LI y otro con LS. Por simplicidad y costo se optó por comparar solo los 4 bits más significantes de la dirección, lo que permite identificar bloques de memoria de 4 K posiciones (1K = 1024) o múltiplos de este valor. Por tanto, MCOM está definida por las relaciones

$$MCOM \geq LI \quad y \quad MCOM < LS$$

de donde se concluye que MCOM (mín) es de 4K posiciones y MCOM (máx) de 60 K. En la fig 4.6a se presenta un ejemplo donde LI = 5 y LS = A (hexadecimal). En la 4.6b se muestra una MCOM (mín) y en la c una MCOM (máx).

Por sencillez la posición de reservaciones (PR) pertenece al bloque de 4 K posiciones que indica LS, de esta manera no se requiere añadir otro comparador. La posición exacta de PR será la que tenga ceros en los 12 bits menos significantes; entonces

$$PR = \overset{4MSB}{LS\ 0000\ 0000\ 0000}$$

donde 4MSB se refiere a los 4 bits más significantes.

El comparador de 4 bits 7485 (ref 4, pp 202-208), proporciona señales de salida cuando las entradas A y B son: $A = B$, $A > B$ y $A < B$.

En la fig 4.7 se muestra el mapa de Karnaugh para la señal ad de acuerdo con

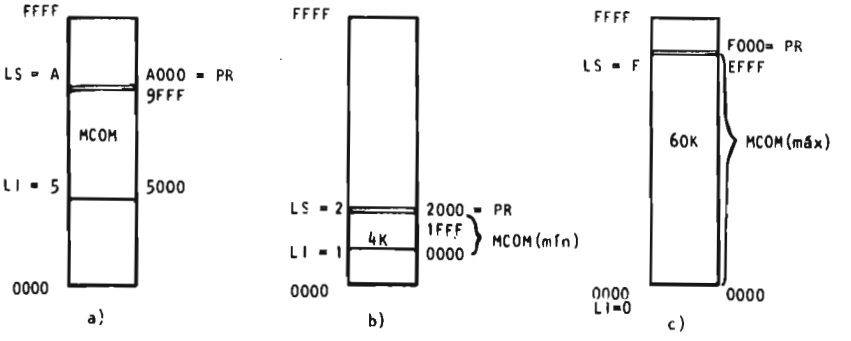


Fig 4.6 Zonas de memoria compartida (MCOM)

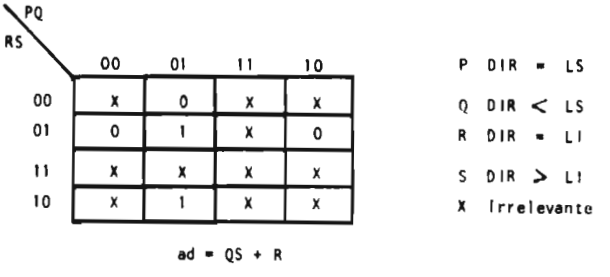


Fig 4.7 Mapa de Karnaugh para ad

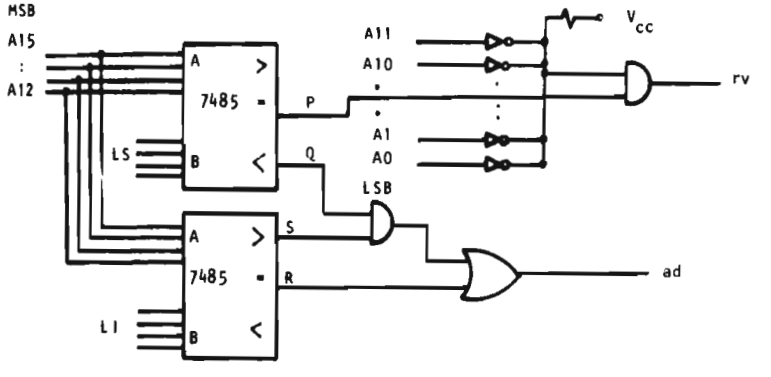


Fig 4.8 Bloque IDENTIFICADOR

P: DIR ■ LS
Q: DIR < LS
R: DIR = LI
S: DIR > LI

De esta tabla se obtiene la expresión lógica de la - fig 4.8, en la que ad = Q S + R; en esta figura se muestra, además, el diagrama lógico de rv. Para identificar que los- 12 bits menos significantes de la dirección sean ceros, `- utiliza un OR ALAMBRADO formado por doce inversores de co-- lector abierto. El AND de la señal P con la salida de este dispositivo da rv.

4.4.1.2. Sincronizador

Se define como

ff = 0 no hay acceso pendiente
ff = 1 acceso pendiente

Las figs 4.9a y b son los mapas de Karnaugh para - las señales c y h, respectivamente, que son

$$c = ac' \wedge \overline{ff} \wedge ad \qquad \wedge = \text{AND}$$
$$h = ac' \wedge \overline{ff} \wedge rv$$

donde

$$ac' = ac \wedge \overline{ab}$$

$\overline{c}'$ y $\overline{h}'$ se definen como

$$\overline{c}^{\top} = ac' \wedge ad$$
$$\overline{h}^{\top} = ac' \wedge rv$$

señales que permiten eliminar la variable ac en el mapa de - Karnaugh de e (fig 4.9c). La expresión de este mapa es

$$e = ff\ h' + c'(ff + rw)$$

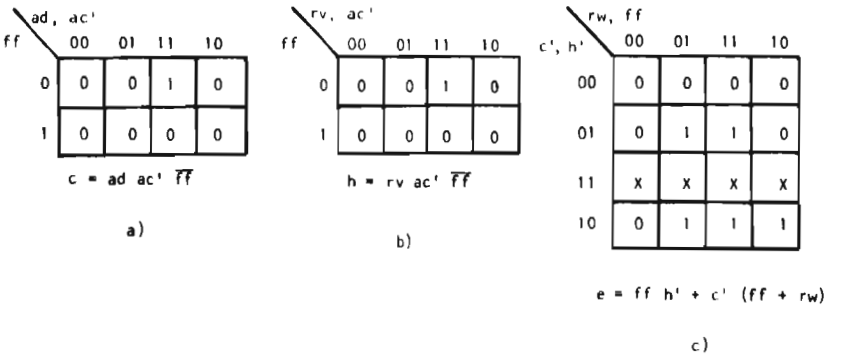


Fig 4.9 Mapas de Karnaugh para SINCRONIZADOR

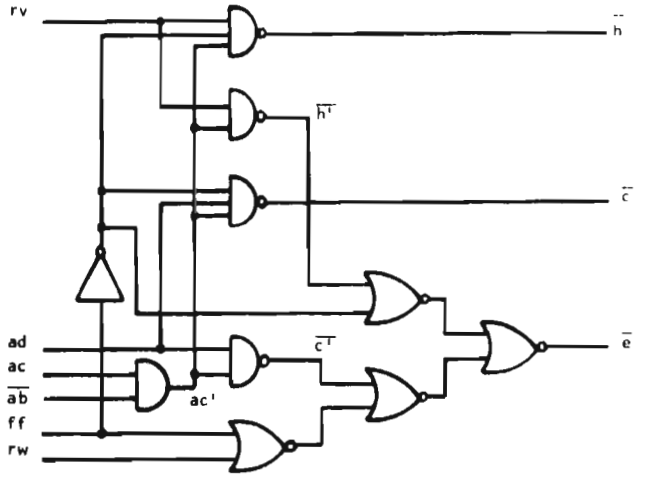


Fig 4.10 Bloque SINCRONIZADOR

El diagrama lógico del bloque aparece en la fig-4.10.

4.4.1.3 Espera

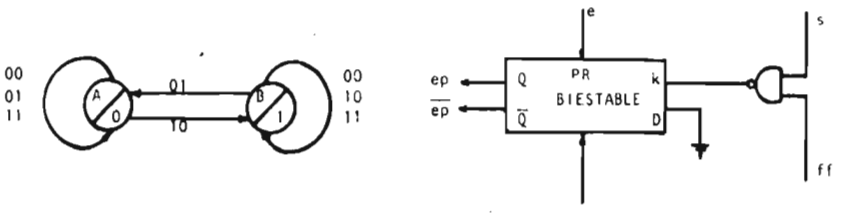
La señal ep se habilita con $\bar{e}$ y se deshabilita -- con ff . Su función se representa mediante un diagrama de-estados modo pulso (ref 2, cap 11) en la fig 4.11a. Algunos μP requieren que se deshabilite la señal de espera solo en determinados momentos, por este motivo se optó disponer de una señal s que permita deshabilitar ep adecuadamente.

En la fig 4.11b se muestra el diagrama lógico de-ese bloque, el cual utiliza un biestable tipo D (7474) que opera como habilitador - deshabilitardor.

4.4.1.4 Alerta

El diseño lógico del bloque se basa en un monoestable redisparable 74123 (ref 4, pág 138) que genera un pulso de duración Q (arbitrariamente grande para el μP) cada-vez que se habilita ep.

El diagrama lógico del bloque se muestra en la -- fig 4.12 y su operación se representa en el diagrama de ti-



a) Diagrama de estados b) Diagrama lógico

Fig 4.11 ESPERA

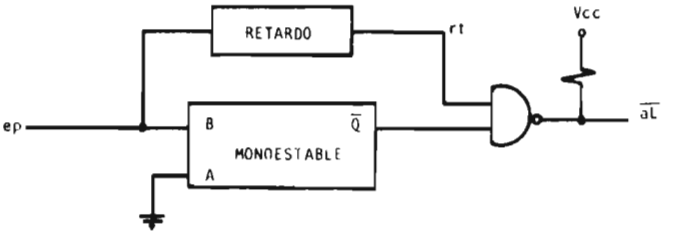


Fig 4.12 Diagrama lógico de ALERTA

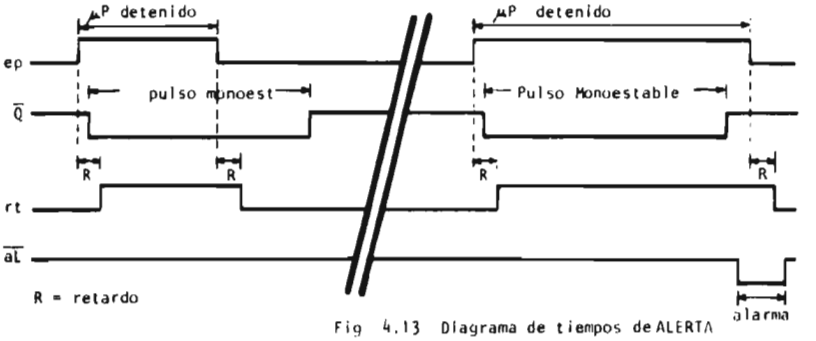


Fig 4.13 Diagrama de tiempos de ALERTA

empos de la fig 4.13. Obsérvese que:

$$t(ep) \leq t\ (Q) \qquad \overline{aT} = 1$$
$$t(ep) > t\ (Q) \qquad \overline{aT} = 0$$

El objeto de utilizar un monoestable redisparables para los casos en que se presenta una nueva habilitación de ep antes de que finalice el pulso Q. El bloque retardo es necesario debido a que el monoestable es más lento en responder que la compuerta NAND. Este bloque puede estar constituido por una o dos compuertas de la serie L (de consumo bajo de potencia) que son lentas en responder. El retraso debe ser mayor de 36 nseg (10^{-9} seg) tiempo máximo en que el monoestable responde (tP_{HL}(máx)) (ref 4, pág 140).

4.4.1.5 Lectura

La señal ab se habilita cuando $\overline{ep}$ cambia del estado "0" al "1", siempre y cuando el acceso sea de lectura. Se deshabilita después de un tiempo t(R) (necesario para que el μP lea un dato). En la fig 4.14 se muestra el diagrama lógico de este bloque; el monoestable (2) proporciona el tiempo t(R), y se dispara solo si rwl = 1 (acceso de lectura). El monoestable (1) evita que el (2) se dispare con los cambios de rwl. En la fig 4.15 se muestra el diagrama de tiempos de las señales de dicho bloque y se observan dos casos: en el a) ab se dispara después de ep (ciclo de lectura), y en el b) no se dispara debido a que es un ciclo de escritura.

4.4.1.6 Registros A

Los tres registros temporales que pertenecen a este bloque son de la siguiente magnitud:

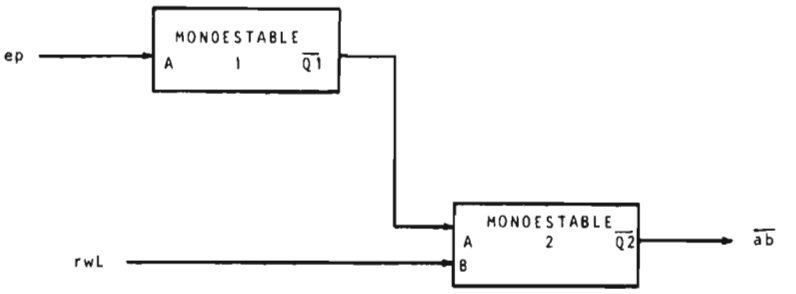
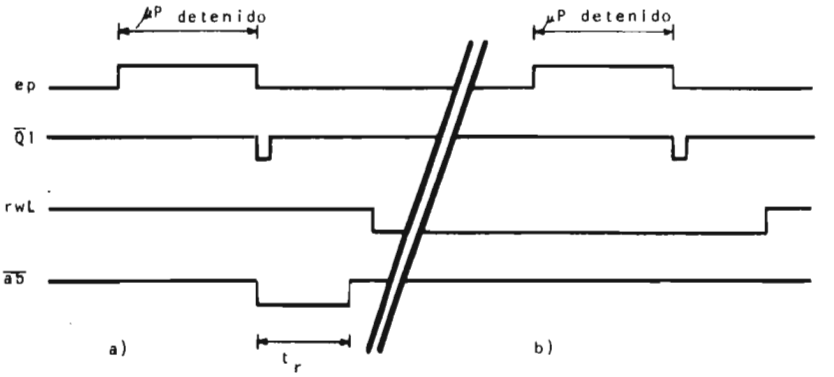


Fig 4.14 Diagrama Lógico de LECTURA



t_r tiempo de lectura del μP

Fig 4.15 Diagrama de tiempos de LECTURA

Direcciones	16 bits
Datos de escritura	8 bits
R/W	1 bit

Los registros se forman con pastillas 7475 (ref 4, pp 182-186), y se habilitan con la señal $\overline{c}$, producida por - SINCRONIZADOR.

4.4.1.7 Conmutador datos lectura

El bloque lo integran ocho compuertas de conmuta-- ción; la salida de cada una tiene tres estados: "0", "1" y- otro de alta impedancia. Las compuertas utilizadas son del tipo 74125 (ref 4, pág 83). La señal $\overline{ab}$ del bloque LECTURA maneja la salida de estas compuertas.

4.4.2 Acceso

Las señales que relacionan el acoplador con MCOM y con sus acopladores vecinos son: vector de direcciones DIC, vectores de datos DLC (lectura), DEC (escritura) y señales- de control, cuyos estados lógicos se definen así

$\overline{re}$ = 0 recepción del pulso de rastreo
$\overline{re}$ = 1 estado estacionario
$\overline{rs}$ = 0 transferencia del pulso de rastreo
$\overline{rs}$ = 1 estado estacionario
$\overline{pt}$ = 0 protesta habilitada
$\overline{pt}$ = 1 protesta deshabilitada
rwc= 0 escritura en MCOM
rwc= 1 lectura de MCOM

4.4.2.1 Bandera

Sincroniza la transferencia de información entre - μP y MCOM. Su circuito lógico lo constituye un biestable tipo D que opera asíncronamente como habilitador-deshabilitador de la señal ff, mediante $\overline{c}$ y $\overline{lp}$, respectivamente.

4.4.2.2 Rastreador

El diseño lógico de este bloque es de los más delicados, ya que el funcionamiento del árbitro depende de él. Para asegurar la confiabilidad se optó por diseñarlo en forma síncrona y consecuentemente es un circuito secuencial.

La trayectoria del pulso de rastreo (cadena circular) es una red compuesta por celdas secuenciales idénticas, en la que cada una se enlaza con sus celdas vecinas para -- formar la trayectoria del pulso de rastreo (fig 4.16). Esas celdas las constituyen los bloques RASTREADORES de cada - - acoplador.

El algoritmo de la celda se describe en el diagrama de estados de la fig 4.17, donde se utiliza el concepto de circuitos secuenciales modo pulso correspondiente a un - circuito Moore (ref 2, pp 309-311). Los estados A, B, C y D se definen como 00, 01, 10 y 11, respectivamente. El -- par de señales de entrada que relacionan los estados entre sí corresponden a x (salida de un dispositivo cuyas entradas son no y ff) y re, y las salidas en cada estado equivalen a rs y $\overline{y}$; de esta manera se construyen la tabla de transición (fig 4.18). Si se utilizan biestables tipo D, las - ecuaciones resultantes para D_0 y D_1 son

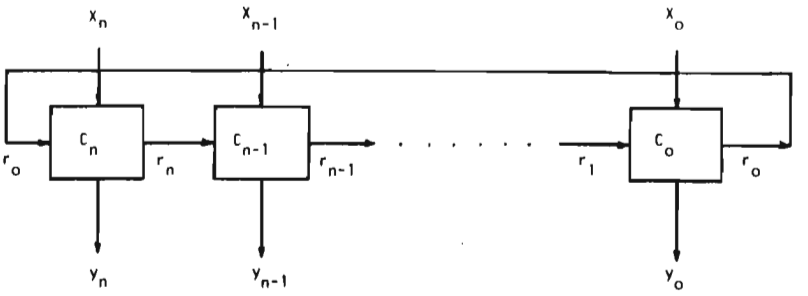


Fig 4.16 Red de celdas secuenciales (RASTREADORES)

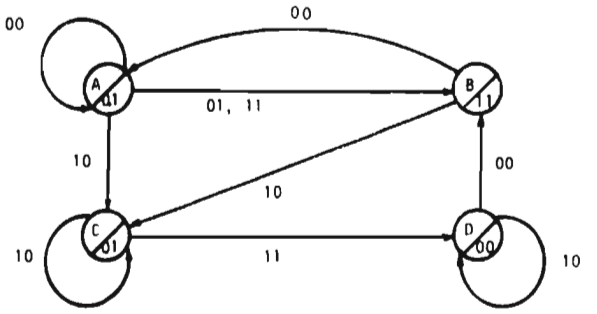


Fig 4.17 Diagrama de estados de la celda

x, re	00	01	11	10	SALIDAS rs, y
00	00	01	01	10	0, 1
01	00	X	X	10	1, 1
11	01	X	X	11	0, 0
10	X	X	11	10	0, 1

Fig 4.18 Tabla de transición de la celda

$$D_0 = q_0 q_1 + re$$
$$D_1 = x\ re + x\ q_1$$

y las correspondientes para rs y y

$$rs = \overline{q_1} q_0$$
$$y = \overline{q_0 q_1}$$

Con estas ecuaciones se obtiene el circuito lógico de la celda que se muestra en la fig 4.19, la entrada x corresponde a la salida de un circuito que conjunta las señales ff y $\overline{no}$. La operación de este circuito se muestra en la fig 4.20, en la que se aprecian dos casos; a) no hay -- protesta por la dirección que se trata de ocupar (no hay - pulso en $\overline{no}$) por lo que x = ff, y b) se presenta una pro-- testa (siempre se presenta cuando ff = 1) lo que ocasiona- que en x se produzca un pulso del ancho del periodo del reloj. Este pulso lo genera el monoestable y permite transferir el pulso de rastreo al siguiente acoplador sin haber- efectuado el acceso a MCOM. Este acceso queda pendiente y- se realiza cuando se retira la protección a esa posición.

La fig 4.21 muestra el funcionamiento de RASTREA-- DOR en un diagrama de tiempos. En el caso a), x se habilita en un periodo de reloj durante el cual no existe pulso- de rastreo, por lo que $\overline{y}$ se habilita al arribar este, mientras que en el caso b) x se habilita durante el mismo lapso en que llega el pulso de rastreo, por lo que $\overline{y}$ no se habilita de inmediato sino al siguiente arribo del pulso de- rastreo; esto es con objeto de evitar que x se habilite durante t_{SETUP} (mínimo) del biestable D1, lo que ocasionaría perder el pulso de rastreo.

Para obtener la máxima frecuencia del reloj en un-

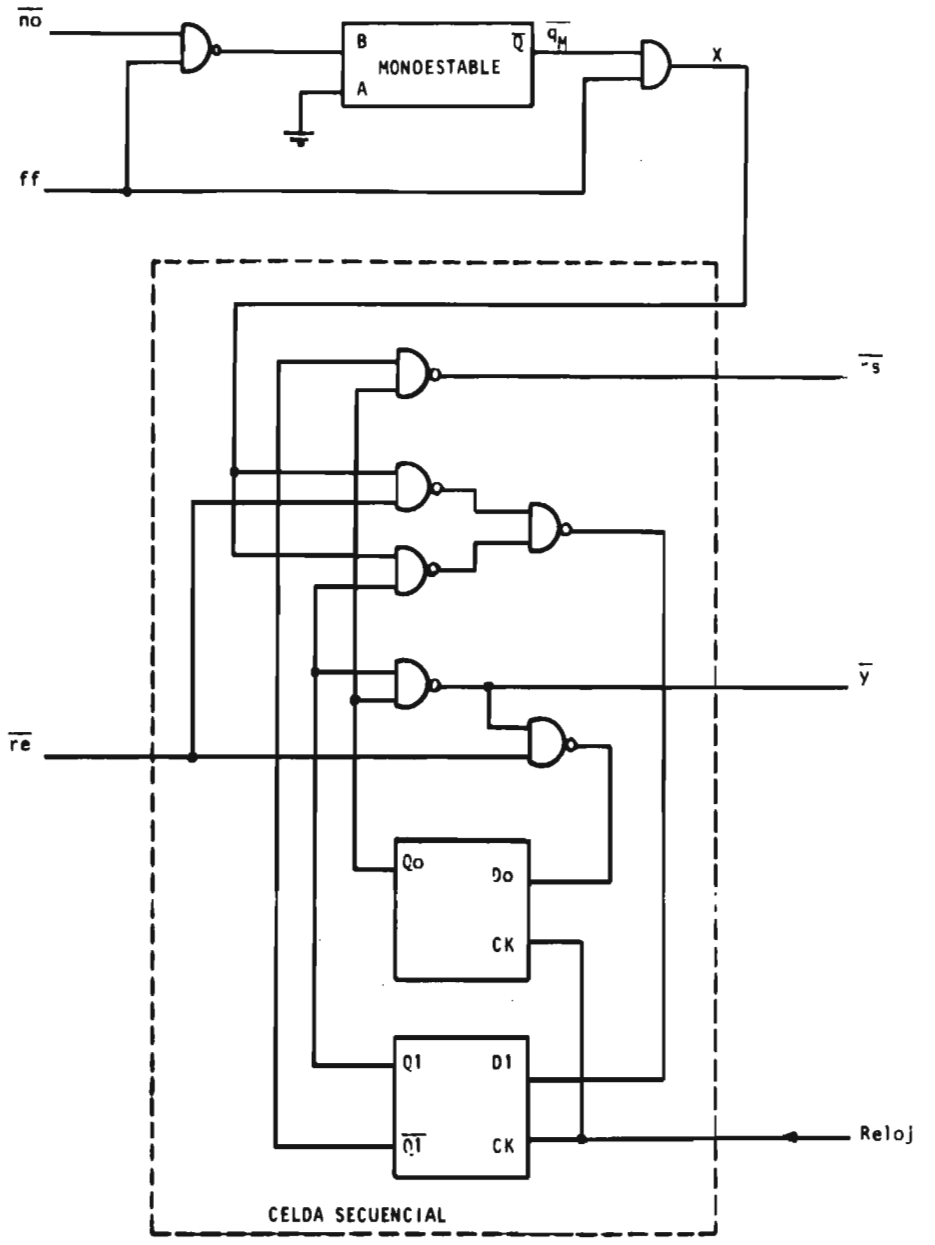


Fig 4.19 Bloque RASTREADOR

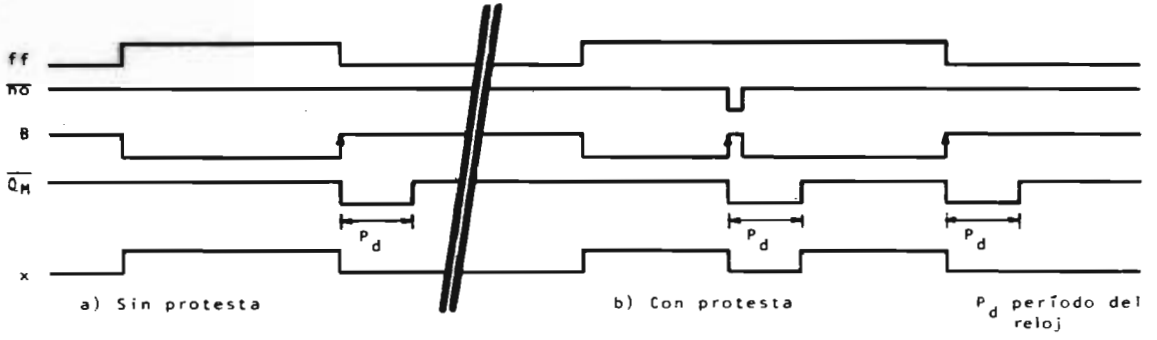


Fig 4.20 Diagrama de tiempo de la operación de x

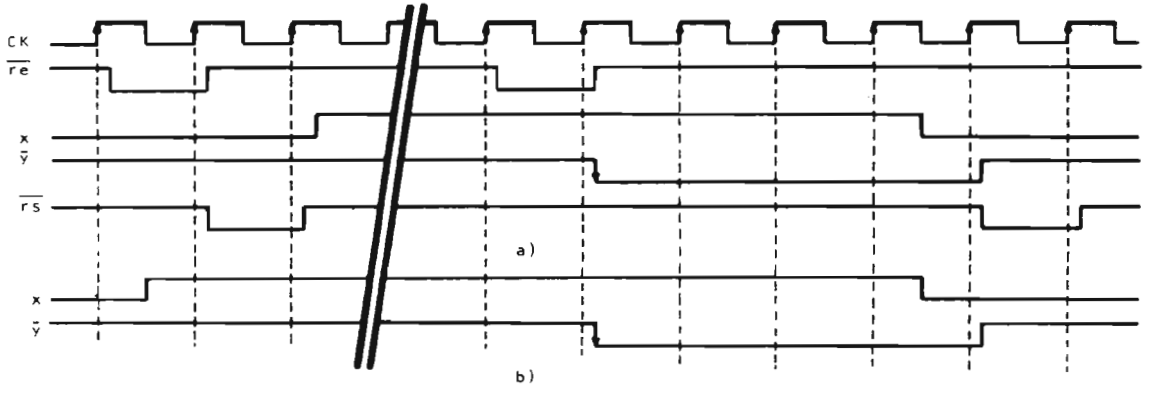


Fig 4.21 Funcionamiento de RASTREADOR

circuito síncrono (ref 3, pp 161-167) se supone el sistema de la fig 3.1, compuesto por dos procesadores y se obtiene, detallando los retardos de las compuertas y biestables, la operación del pulso de rastreo en la fig 4.22. La tabla 4.1 es con ejemplo de cómo obtener el t_p(mín) con base en los retardos especificados en la fig 4.22, de donde se obtienen las frecuencias máximas garantizadas del reloj al emplear circuitos integrados TTL de las series normal, alta velocidad (H) y schottky (S). La decisión de utilizar alguna de las series dependerá del compromiso costo, consumo de energía y disipación de calor vs velocidad de transferencia del pulso de rastreo.

4.4.2.3 Tiempo

El diagrama lógico de este bloque está constituido unicamente por un monoestable que se encarga de generar el pulso de duración $t = t_{ACC} + t_{PROT}$. Este monoestable se dispara con el flanco de "1" a "0" de $\overline{y}$.

4.4.2.4 Bloqueador

Los diagramas lógicos de este bloque y el de TIEMPO se muestran en la fig 4.23. La operación de ambos se caracteriza por los cuatro casos que se incluyen en la fig 4.24 y se describen a continuación.

a) Acceso normal. En este caso no hay protesta (no hay pulso en $\overline{pt}$), por lo que el pulso en $\overline{lo}$ es igual al pulso $\overline{ta}$ que genera el monoestable.

b) Protesta en acoplador inactivo. Inactivo es aquel acoplador que no tiene su dirección colocada en el canal común, mientras que el activo si. Cualquier protesta llega a todos los acopladores, de los cuales solo el activo

Tabla 4.1 OBTENCION DE LA FRECUENCIA MAXIMA DEL RELOJ PARA CADA SERIE

SERIE		e	f	d	c	a	b	g	h	t_p	f_{ck}		i	j
		7400	7400	7400	7400	7474	7474	7474	7474	MIN	MAX		7474	7474
		t (HL)	t (LH)	t (HL)	t (LH)	t_q (LH)	t_q (HL)	t STP	t STP	nseg	MHZ		t HOLD	t HOLD
NORMAL	TIP	7		7	11	14		58		97	10.3		31	
74 --	MAX	15		15	22	25		20		97	10.3		??	
NORMAL	TIP		11				20		66	97	10.3			39
74 --	MAX		22				40		35	97	10.3			77
ALTA VELOC.	TIP	6.2		6.2	6	8.5		33.1		60	16.6		19	
74H --	MAX	10		10	10	15		15		60	16.6		30	
ALTA VELOC.	TIP		6				13		41	60	16.6			26.9
74H --	MAX		10				20		30	60	16.6			45
SCHOTTKY	TIP	3		3	3	6		11.5		26.5	37.7		9	
74S --	MAX	5		5	4.5	9		3		26.5	37.7		13.5	
SCHOTTKY	TIP		3				6		17.5	26.5	37.7			15
74S --	MAX		4.5				9		13	26.5	37.7			23.5

STP = SETUP

t_p (mín) = e(máx) + d(máx) + c(máx) + a(máx) + g(máx)

donde

g(máx) = t_{SETUP}(mín) especificado por el fábricante

g(típ) = t_p(mín) - e(típ) - d(típ) - c(típ) - a(típ)

h(típ o máx) = t_p(mín) - f(mín) - f(típ o máx) - b(típ o máx)

j = a + e + c + d = t_p (mín) - g

i = b + f = t_p(mín) - h

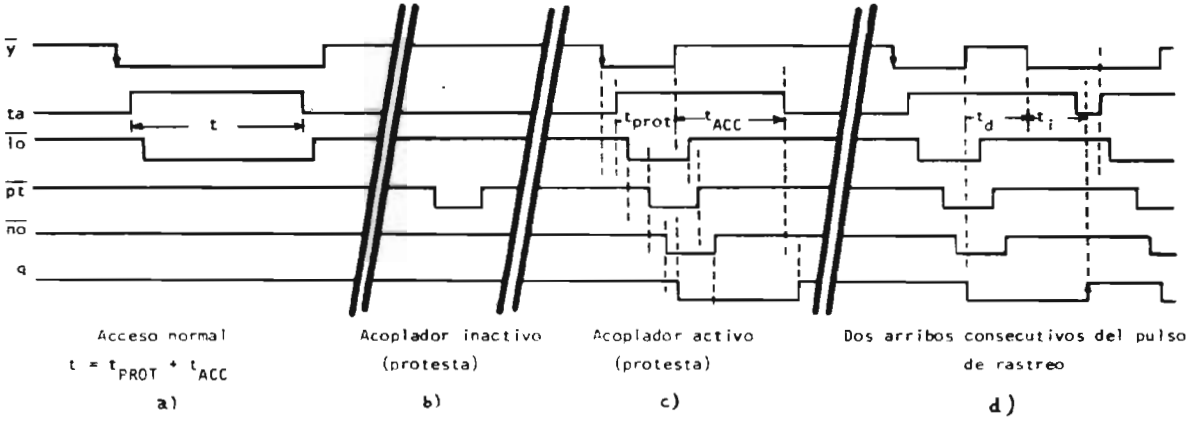


Fig 4.24 Operación de BLOQUEADOR

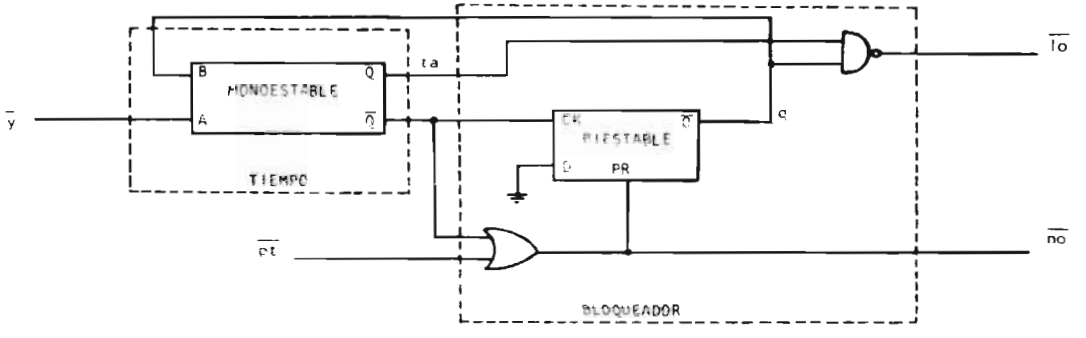


Fig 4.23 Diagrama lógico de TIEMPO y BLOQUEADOR

la toma en cuenta. En la fig 4.24b se muestra que una protesta en un procesador inactivo no lo produce ningún efecto.

c) Protesta en acoplador activo. Este quita de inmediato la dirección del canal común y habilita la señal $\overline{no}$, que indica acceso bloqueado. Se transfiere el pulso de rastreo al siguiente acoplador, sin embargo, queda pendiente el acceso.

d) Dos arribos consecutivos del pulso de rastreo. Cuando son pocos μPs, en ocasiones el pulso de rastreo transferido a causa de una protesta regresa antes de que termine el pulso t_{ACC}. En este caso se pierde un tiempo t_i durante el cual no se tiene acceso a MCOM ni se transfiere el pulso de rastreo. Terminado el pulso en ta se transfiere el pulso de rastreo, como indica la fig 4.24d.

4.4.2.5 Recorte

Se presentan dos casos típicos en el funcionamiento de este bloque: cuando el acceso es normal (no hay protesta) y cuando existe protesta. Su diagrama lógico se presenta en la fig 4.25 y en la fig 4.26 se muestra el funcionamiento del bloque en un diagrama de tiempos de acuerdo con esos dos casos.

a) Acceso normal. El inicio del pulso en $\overline{Tp}$ se retrasa cierto tiempo respecto al pulso $\overline{To}$, pero la terminación de ambos es simultánea (fig 4.26); el retraso es de duración t_{PROT}, tiempo durante el cual se compara la dirección del canal común con las direcciones almacenadas en todos los acopladores; si alguno protesta lo hace durante este intervalo, por lo contrario ya no hay protesta y se puede llevar a cabo el acceso a MCOM (se habilita el pulso en $\overline{Tp}$).

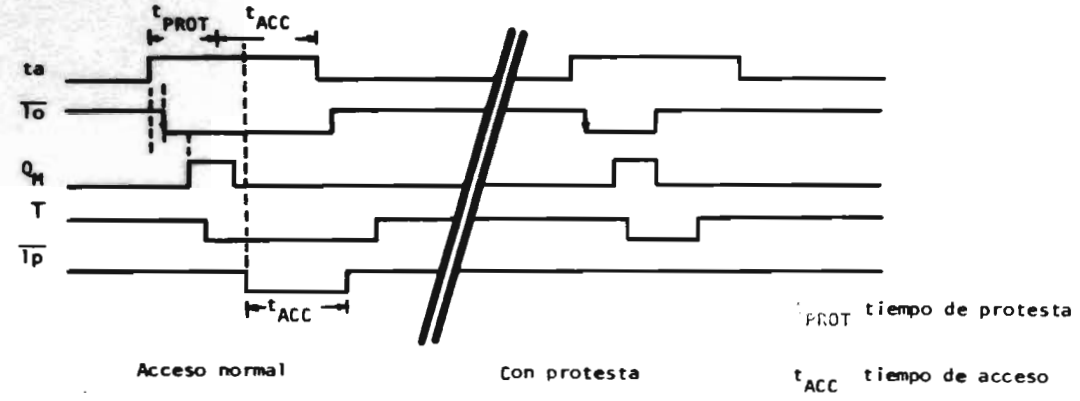


Fig 4.26 Tiempos de RECORTE

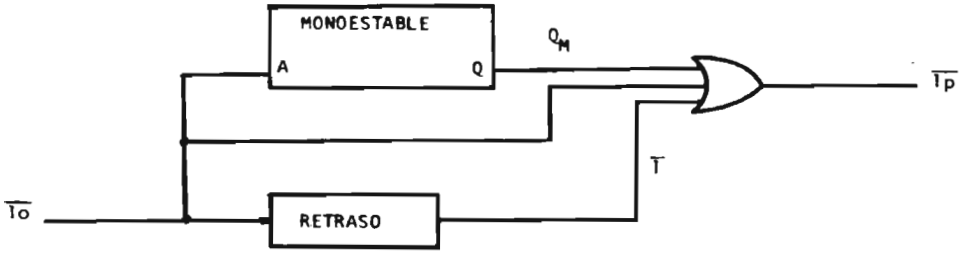


Fig 4.25 Bloque RECORTE

b) Existe protesta. Esto ocasiona que el acoplador que trata de tener acceso a MCOM quite de inmediato la dirección del canal común, deshabilitando $\overline{To}$, y no permitiendo que se genere el pulso en $\overline{Tp}$. Posteriormente, se transfiere el pulso de rastreo como se menciona en 4.4.2.4.

En la fig 4.25 el monoestable retrasa el inicio del pulso $\overline{Tp}$ respecto al de $\overline{To}$ un tiempo de duración t_{PROT}; el bloque RETRASO tiene la misma función que el de la fig 4.12.

4.4.2.6 Conmutador A

Está compuesto por tres bloques de conmpuertas tres-estados 74125. En el conmutador de dirección existen 16 compuertas, en el de R/W una y en el de datos de escritura ocho. El conmutador de dirección es comandado por $\overline{To}$ y los otros dos por $\overline{Tp}$.

4.4.2.7 Registro de datos de lectura

Este bloque recibe el dato que se obtiene de lecturas a MCOM y consiste en un registro de ocho bits que se comanda con la señal $\overline{Tp}$ del bloque RECORTE.

4.4.3 Protector.

Este bloque se comunica con DETECTOR, ACCESO y el canal común por medio de los vectores de direcciones DIL y DIC, y de las señales de control cuyos estados lógicos definen lo siguiente

$\overline{h} = 0$ aviso de protección
$\overline{h} = 1$ estado estable
$\overline{Tp} = 0$ acceso a MCOM

$\overline{Tp}$ = 1 estado estable
$\overline{To}$ = 0 direccionamiento a MCOM
$\overline{To}$ = 1 estado estable
$\overline{pt}$ = 0 protesta habilitada
$\overline{pt}$ = 1 protesta deshabilitada

4.4.3.1 Aviso

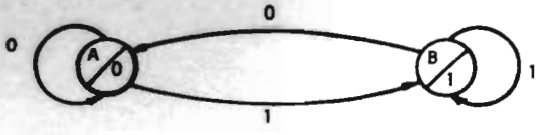
El diagrama lógico está constituido por un biestable tipo D que opera como habilitador-deshabilitador de la señal bt. La señal h habilita bt lo que indica que deberá protegerse el siguiente acceso a MCOM; una vez efectuado ese acceso se deshabilita bt por medio de qi. A partir de este momento el bloque queda en condiciones de aceptar un nuevo aviso de protección.

4.4.3.2 Habilita

Este bloque habilita z (señal de protección) durante el intervalo de protección. La señal $\overline{Tp}$ indica inicio y final del intervalo, por tanto se usa como reloj en un circuito secuencial cuya entrada es bt.

En la fig 4.27 se muestra el diagrama de estados y la tabla de transición de HABILITA, donde se obtiene la expresión para la salida z al utilizar un biestable tipo D, resulta: z = bt. Los diagramas lógicos de AVISO y HABILITA se muestran en la fig 4.28; la función del bloque retraso, es rezagar la señal $\overline{Tp}$ un tiempo mayor de 20 nseg para cambiar el estado de z antes que lo haga bt.

Si durante el intervalo de protección se realiza nuevamente un aviso dicho intervalo se prolonga porque el acceso de desprotección inicial se convierte en uno de protección. De esta manera se pueden realizar avisos durante



a) Diagrama de estados

Q \ bt	0	1	z
0	0	1	0
1	0	1	1

b) Tabla de transición

Fig 4.27 HABILITA

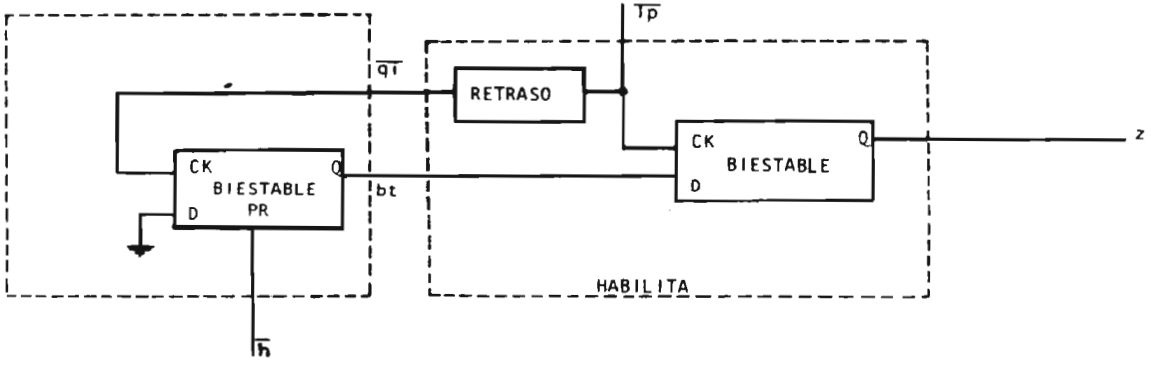


Fig 4.28 Bloques AVISO y HABILITA

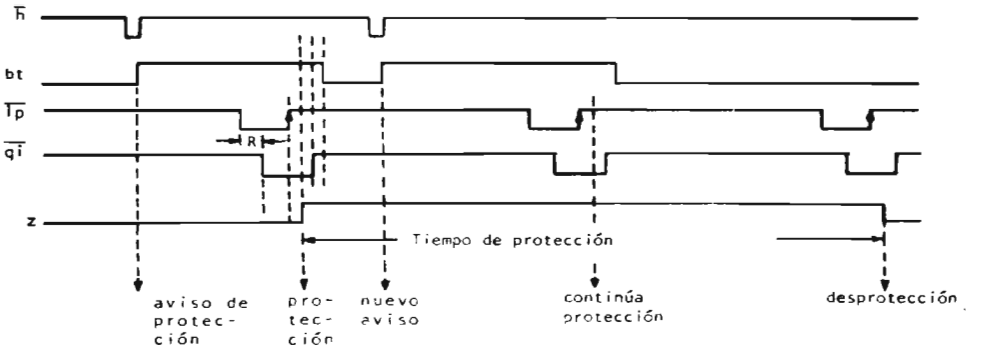


Fig 4.29 Diagrama de tiempos para AVISO y HABILITA

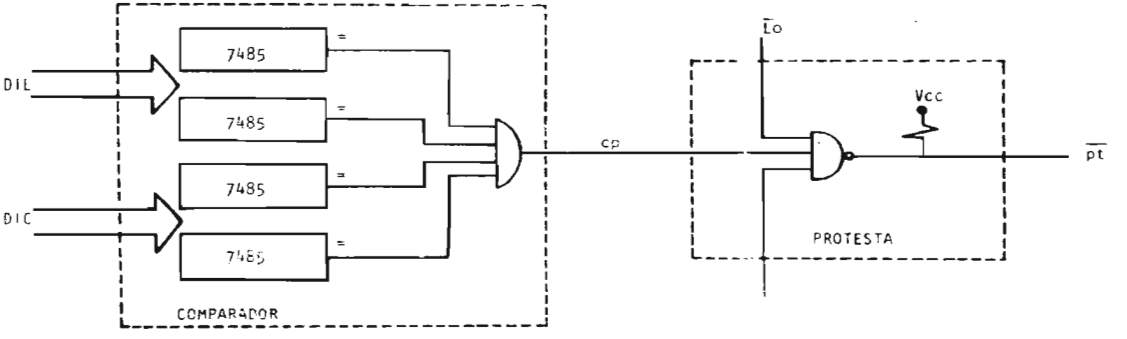


Fig 4.30 BLOQUES PROTESTA Y COMPARADOR

4.5 REFERENCIAS

1. Morris, R L y Miller, J R, "Designing with TTL integrated circuits", McGraw-Hill Kogakusha, LTD, Tokyo, Japón (1971)

2. Hill, F J y Peterson, G R, "Introduction to switching - theory & logical design", segunda edición, Wiley International edition, Nueva York (1974)

3. Peatman, J B, "The design of digital systems", McGraw - Hill book Co, Nueva York (1972)

4. - "The TTL data book for design engineers", Texas Ins-- truments Inc, primera edición, Dallas, Texas (1973)

5

FUNCIONAMIENTO DEL SISTEMA

5.1 INTRODUCCION

Para iniciar el funcionamiento del sistema se requieren, además de los acopladores, un generador de pulsos (reloj), un dispositivo que proporcione el pulso de arranque (arrancador) al pulso de rastreo, y los adaptadores respectivos a los procesadores. En la parte práctica del presente trabajo se construyó un acoplador con un adaptador adecuado para el μP 6502 y se simuló un segundo acoplador para probar la operación del procesador con memoria.

En este capítulo se discuten los diseños del reloj, arrancador, adaptador y simulador; se presentan resultados de algunas pruebas sobre la operación del acoplador y, finalmente, se proponen nuevas versiones del mismo para ampliar su campo de aplicación.

5.2 REQUERIMIENTOS PARA EL FUNCIONAMIENTO

Los elementos necesarios para iniciar el funcionamiento del sistema se describen en los siguientes párrafos.

5.2.1 Reloj

El generador de pulsos es necesario para transferir síncronamente el pulso de rastreo, que puede ser externo o interno; en el primer caso hay que respetar los niveles lógicos y la frecuencia máxima que acepta el bloque RASTREADOR (sec 4.3.2.2).

Dado que se incluyó el reloj en el acoplador, aquél fue diseñado para una frecuencia de 10 MHz ya que la serie que se utiliza en RASTREADOR es la normalizada, que acepta como máximo una frecuencia de 10.3 MHz. El diseño del reloj (fig. 5.1) aprovecha la característica de histéresis de las compuertas TTL "schmitt trigger" (ref 1, pág 98).

5.2.2 Arrancador del pulso de rastreo

Para iniciar su recorrido a través de todos los acopladores, el pulso de rastreo necesita un impulso que puede ser proporcionado por el arrancador en forma automática (al energizar el sistema) o manualmente (al oprimir un botón); además, el mismo arrancador permite limpiar (colocar en estados adecuados) los biestables que pudieran estar en estado indeseable para el funcionamiento del sistema.

Se optó por diseñar un arrancador manual (fig 5.2) dividido en tres partes: la primera es un filtro pasabajas que proporciona un sólo pulso cada vez que se oprime el botón (el cual sirve también para limpiar los biestables); la segunda parte es un monoestable (2) que produce

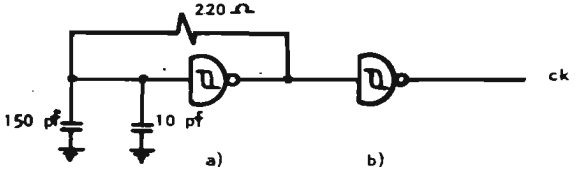


Fig 5.1 Reloj de frecuencia de 10 MHz

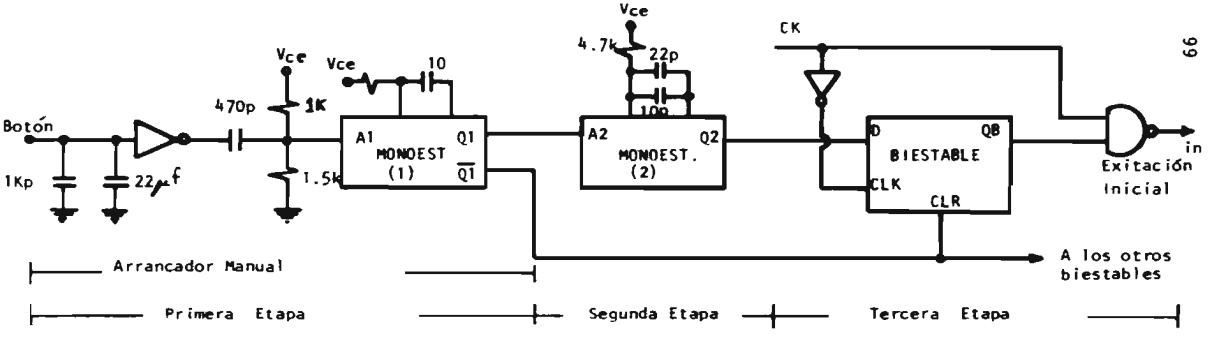


Fig 5.2 Arrancador del pulso de rastreo

un pulso de duración igual al periodo del reloj; y la tercera consiste en un biestable y dos compuertas que permiten habilitar adecuadamente el pulso de arranque (o sea, no permiten que se habiliten dos pulsos de rastreo).

5.2.3 Adaptador para el μP 6502

El procesador utilizado durante las pruebas fue el μP MCS6502 (ref 2) para el cual se diseñó un adaptador adecuado. La fig 5.3 muestra las señales que requieren ser adaptadas; las líneas de salida del μP pueden manejar corriente equivalente a una carga TTL, pero el acoplador demanda corriente equivalente a 5.33 cargas en las líneas dirección y 2 en las de datos (ref 3), por lo que son necesarios amplificadores de corriente para estas líneas.

El diagrama lógico del adaptador (fig 5.4) muestra las amplificadores de corriente para las líneas de dirección y datos, estos últimos tienen salida de tres estados para manejarlos en forma bidireccional. La señal ac es un pulso de 50 nseg que se produce, en cada ciclo de máquina, al asegurar la estabilidad de las señales de salida del μP (300 nseg después del inicio de la fase Ø1). Finalmente, las demás señales se adaptan de la siguiente manera: rw = R/W, RDY = $\overline{ep}$ y ac = Ø1.

5.3 PRUEBAS DEL ACOPLADOR

Para probar el acoplador se empló una microcomputadora KIM-1 (ref 4) construida con el μP 6502 y un dispositivo que simula la operación de otro acoplador (simulador). El KIM-1 utiliza las zonas de memoria 0000 a 03FF (hexadecimal) y 1C00 a 1FFF, como memorias privadas, el resto está disponible, por esta razón se escogió la zona que principia

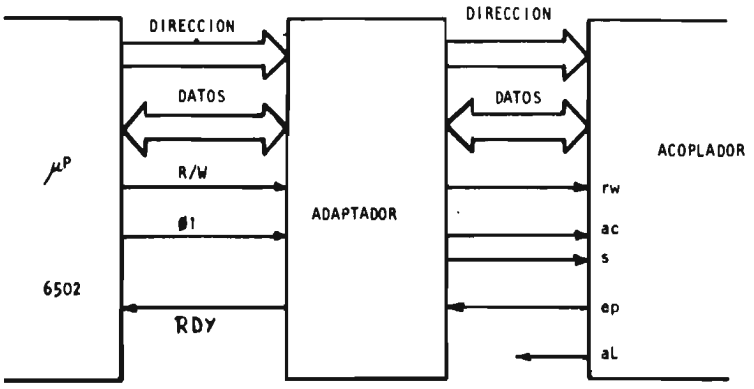


Fig 5.3 Conexión entre μP 6502 y Acoplador

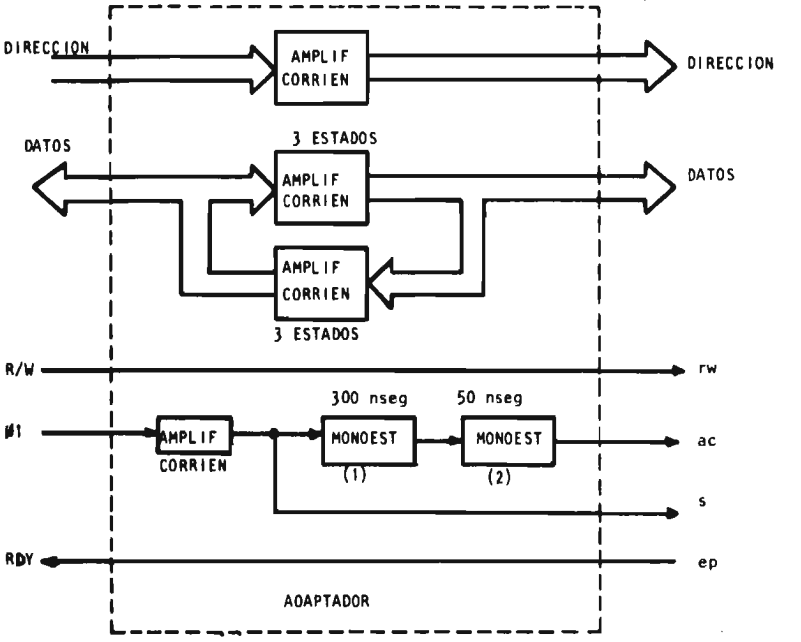


Fig. 5.4 Diagrama lógico del adaptador

en la dirección 5000 y termina en la 7FFF para utilizarla como memoria compartida. La posición de reservaciones resulta ser 8000.

De acuerdo con lo que se establece en el Apéndice B, el μP 6502 no detiene su ejecución en ciclos de escritura, por lo que no fue posible realizar escritura sobre escritura (sec 3.3.3) ni lectura sobre escritura (sec 3.3.4).

El simulador es un dispositivo que consiste en un generador de pulsos disparable y los bloques RASTREADOR, CONMUTADOR DE DIRECCIONES Y COMPARADOR del acoplador; sus funciones son: Pasiva, transferir el pulso de rastreo cada vez que lo recibe sin ejecutar ninguna acción, Activa, ejecutar un acceso a una posición fija al recibir el pulso de rastreo, y Protección, protestar por una posición de MCOM como si la hubiera protegido.

Al respecto, existen dos tipos de pruebas para mostrar el funcionamiento del acoplador:

5.3.1 Comunicación entre procesador y MCOM

Esta prueba consiste en escribir y leer datos de MCOM, para lo cual el simulador opera en forma pasiva. Los parámetros obtenidos son:

Retardo en atender una solicitud de acceso, t_{AT}, variable entre 100 y 200 nseg

Retención del pulso de rastreo en cada acceso, t_{RET} = 600 nseg

Tiempo desde que detecta una posición de MCOM hasta que habilita la señal acceso pendiente, t_{AP} = 160 nseg

Tiempo que se mantiene la dirección en el canal común, t_{DIR} = 370 nseg

Tiempo real de acceso, t_{ACC} = 280 nseg

Tiempo desde que detecta una posición de MCOM hasta que genera la señal que detiene el μP, t_{DET} = 180 nseg

Tiempo que se mantiene detenido el μP, t_D = 1 μseg

Tiempo para habilitar el dato adquirido de MCOM a fin de que sea leído por el μP, t_{LEC} = 800 nseg

De lo anterior se concluye:

Rendimiento máximo de MCOM = $\frac{t_{ACC}}{t_{RET}}$ X 100% = 46.8%

Porcentaje de tiempo perdido a causa de transferir el pulso de rastreo = $\frac{t_{TRA}*}{t_{RET}}$ X 100% = 16.6%

Porcentaje de tiempo perdido a causa del subsistema de protección = $\frac{t_{DIR} - t_{ACC}}{t_{RET}}$ = 15 %

Porcentaje de tiempo perdido a causa de retardos internos del acoplador = 100% - 46.8% - 16.6% - 15% = 21.6%

Una característica importante de mencionar es que algunas instrucciones, por ejemplo, las llamadas "lee-modifica-escribe" hacen referencia a MCOM dos veces (Apéndice D), sin embargo, la dirección se mantiene en el canal del μP 3 o 4 ciclos de máquina (depende de la instrucción), lo cual ocasiona que el acoplador realice igual número de accesos a

* t_{TRA} tiempo de transferencia = 100 nseg

MCOM. Esto, aunque reduce la velocidad de procesamiento, no influye negativamente en la operación del procesador ni en - la del sistema.

5.3.2 Subsistema de protección

A fin de probar el funcionamiento del subsistema - de protección, la prueba se dividió en dos partes: en la primera el simulador (operando en forma activa) intenta tener - acceso a una posición de MCOM protegida por el procesador. El parámetro importante en este caso es el tiempo de protesta, t_{PROT} = 60 nseg. En la segunda parte se cambiaron las funciones, o sea que el procesador tiene acceso a una posición de MCOM que está protegida por el simulador (opera en forma de protección). Como el objeto es examinar la reacción del acoplador ante una protesta, se obtuvieron los siguientes parámetros:

Tiempo en reconocer la protesta, t_{REP} = 62 nseg

Tiempo en retener el pulso de rastreo, t_{RET} = 300 nseg

Tiempo en mantener habilitada la dirección en el canal, t_{DIR} = 120 nseg

5.4 NUEVAS VERSIONES DEL ACOPLADOR

Debido al diseño modular del acoplador se puede - ampliar facilmente para satisfacer diferentes necesidades.- A continuación se proponen dos nuevas versiones del mismo.

5.4.1 Acoplador para estructura de barras cruzadas

En la sec 3.1 se discutió la facilidad de ampliarla estructura de canal único de tiempo compartido a una devarios canales, a fin de aumentar la velocidad de procesa---miento mediante accesos simultáneos a diferentes memorias.

El modelolo integrann procesadores y m módulos de memoria convencionales de una sola vía de acceso, interconectados por una matriz de conmutación. Cada vía de acceso a - memoria tiene un árbitro distribuido e implantado por ordenestático (similar al utilizado en la estructura de canal único). Por tanto,el acoplador que se requiere es una versión-modificada del original con capacidad para identificar la memoria que solicita el procesador, pedir acceso a la misma, y realizarlo cuando le llegue el pulso de rastreo correspondiente.

La fig 5.5 muestra el diagrama de bloques de estanueva versión, en la que el procesador le indica una sola -- vez al acoplador la memoria que desea emplear para que todos los accesos subsecuentes sehagan a esta; cada vez que cambie - de memoria deberá indicarlo al acoplador; por lo que este -- dispone de un registro donde almacena el número de memoria - (índice) que solicita el procesador conectado a un decodificador de 8 a m, donde el valor máximo de m es 256.

5.4.2 Acoplador para 16 bits de datos

Muchas minicomputadoras -y actualmente, algunos - microprocesadores - manejan datos de 16 bits, generados o - leídos por el CPU en dos grupos de 8; internamente manejandatos de 8 bits pero las transferencias externas son de 16, como en el caso del μP Z80 (ref 5). Para realizar opera--ciones con palabras de 16 bits en memorias de 8, se especi-

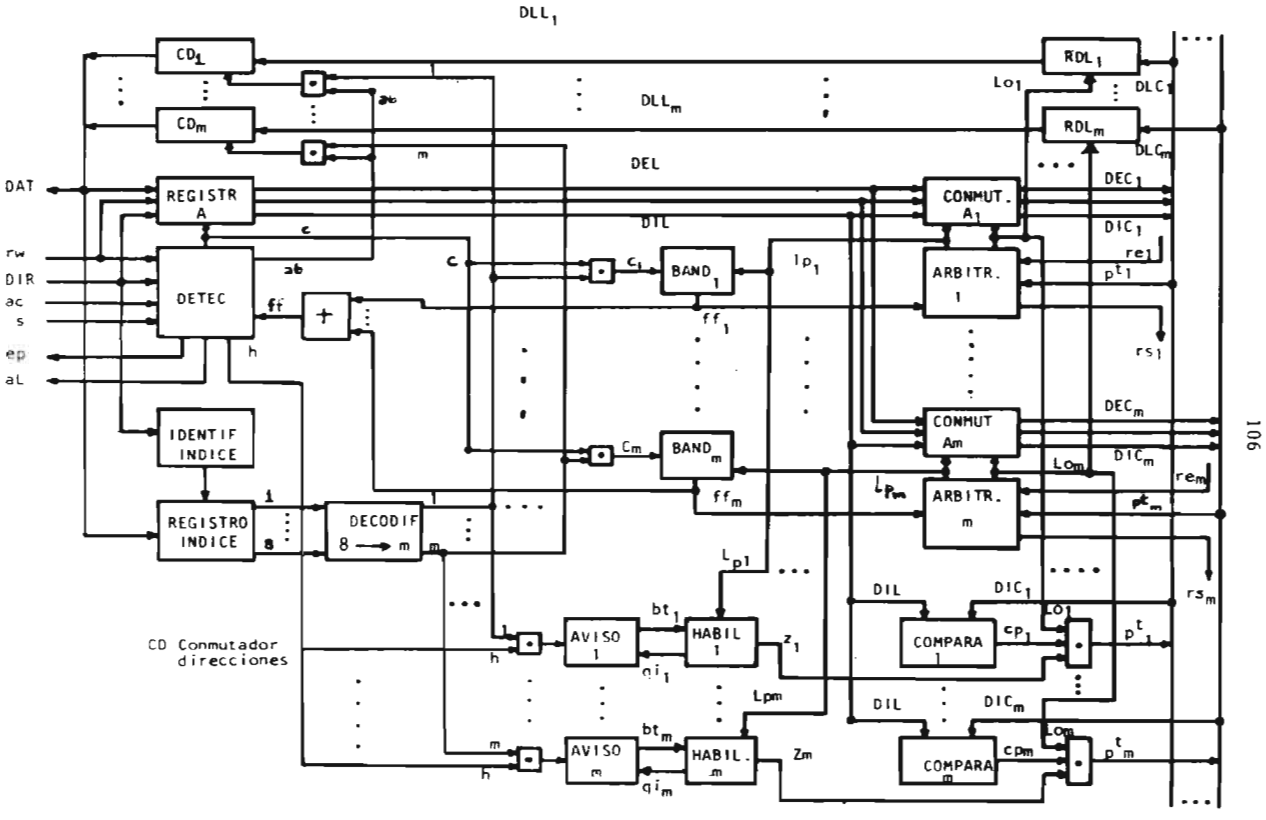


Fig 5.5 Acoplador para La estructura de barras cruzadas

fica solo la posición par de las dos consecutivas donde se - encuentra el dato y el procesador automáticamente maneja la otra. En el caso de memorias de 16 bits se acostumbra asignar direcciones pares a estas posiciones, de esta manera queda implícito, para acciones internas del procesador, que la- posición impar la compone la segunda mitad de la palabra.

La fig 5.6 muestra el diseño de un acoplador que maneja palabras de 16 bits con MCOM y 8 con el procesador; el bloque DETECTOR se divide en dos partes; una identifica las- posiciones impares de MCOM (byte más significante MSB del dato) con las que realiza las transferencias entre procesador- y acoplador únicamente, no requiriendo esperar el pulso de - rastreo para llevarlas a efecto; la otra identifica las posiciones pares (byte menos significante LSB del dato) con las- que realiza las transferencias de 16 bits con MCOM tomando - el byte más significante del REGISTRO DATO (MSB).

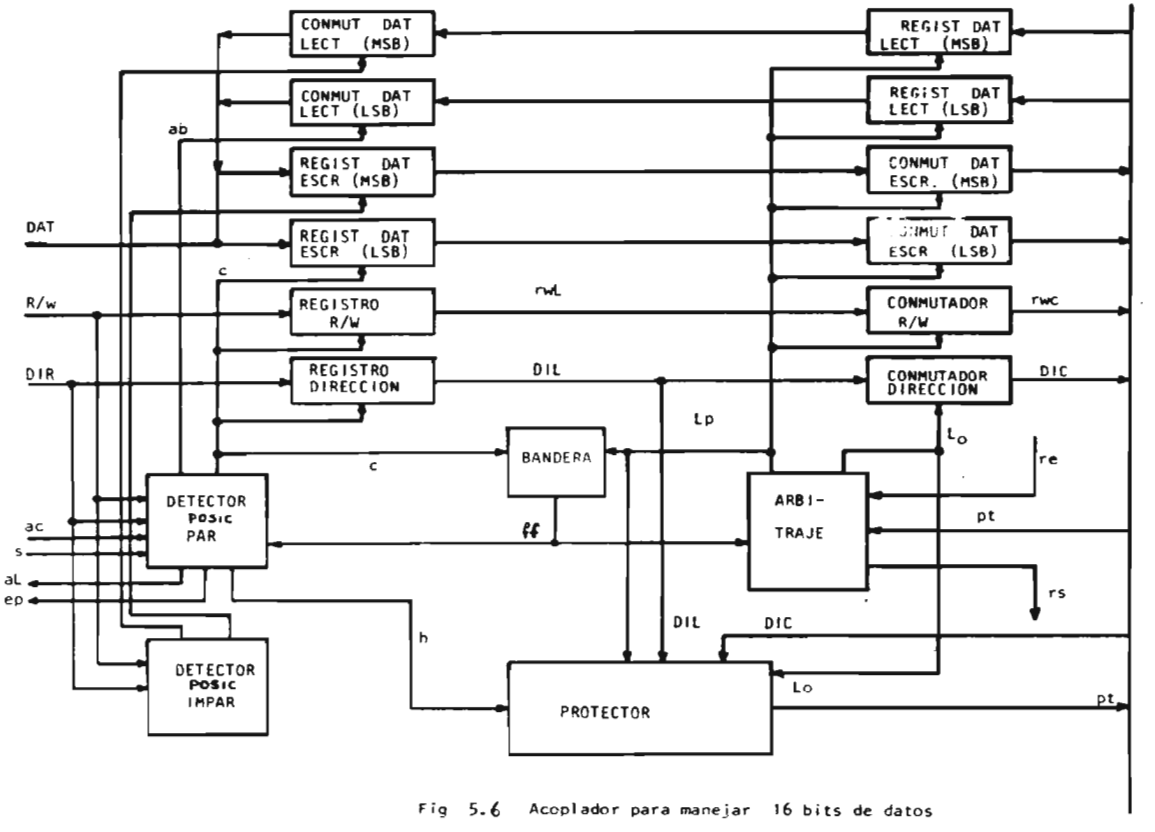


Fig 5.6 Acoplador para manejar 16 bits de datos

5.5 REFERENCIAS

1. - "The TTL data book for design engineers", Texas Instruments, Inc, Primera edición, Dallas Texas (1973)

2. - "MCS 6500 microcomputer family, hardware manual", MOS Technology, Inc, Publicación No 6500-10A, Norristown, - Filadelfia (ene 1976)

3. Morris, R L y Miller, J R, "Designing with TTL inte-- grated circuits", Mc Graw - Hill Kogakusha, LTD, Tokyo, Japón (1971), pp 57 - 65

4. - "KIM-1 user manual", MOS Technology, Inc, Norristown, Filadelfia (ene 1976)

5. - "Z80 - CPU, Z80A - CPU, technical manual", Zilog, --- Nueva York (1977)

6

SUBSISTEMA DE PROTECCION

6.1 INTRODUCCION

El subsistema de protección adquirió gran importancia debido a su característica modular congruente con el concepto del acoplador: facilita la sustitución, adición o sustracción de procesadores. Además, de los resultados del cap 5 se observa que los tiempos en protestar, cuando se trata de utilizar una posición protegida, y en reconocer las protestas, son muy cortos, por lo que resulta bajo el porcentaje de tiempo que se desaprovechala memoria compartida, a causa del subsistema de protección.

Gracias a esas ventajas se dedica este capítulo a discutir sus características, problemas que acarrea su funcionamiento, soluciones para esos problemas, sugerencias para operarlo adecuadamente y nuevas versiones del mismo.

6.2 SECUENCIA DE PROTECCION

Existen dos casos típicos en que es preciso proteger una posición de MCOM: a) para evitar que algún procesador lea el dato de una posición mientras se toma una decisión con este, y b) para evitar que el contenido de una posición se destruya o se altere durante determinado tiempo.

Estos casos tienen que sujetarse a la secuencia de protección descrita en 3.3.5, la cual se representa ahora en un eje del tiempo (fig 6.1) y en un diagrama de flujo (fig 6.2). Los intervalos de tiempo 2 → 3 y 4 → 5 (fig 6.1) se deben a que el acoplador espera el pulso de rastreo (su turno) para realizar el acceso. La duración de estos intervalos es aleatoria entre t_{TRA} y un tiempo máximo, que se produce cuando el pulso de rastreo recorre los demás acopladores concediendo acceso a todos; este tiempo máximo se puede expresar como:

$$t_{RET}(máx) = (N-1)\ (t_{ACC} + t_{PROT} + t_{TRA})$$

donde

t_{RET} (máx)	retardo máximo en atender un acceso
N	número de procesadores
t_{ACC}	tiempo de acceso a MCOM
t_{PROT}	tiempo de protesta
t_{TRA}	tiempo en transferir al siguiente acoplador el pulso de rastreo

6.3 FORMAS DE PROTECCION

Existen dos casos para proteger una posición de MCOM los cuales se diferencian en la manera de desproteger la posición.

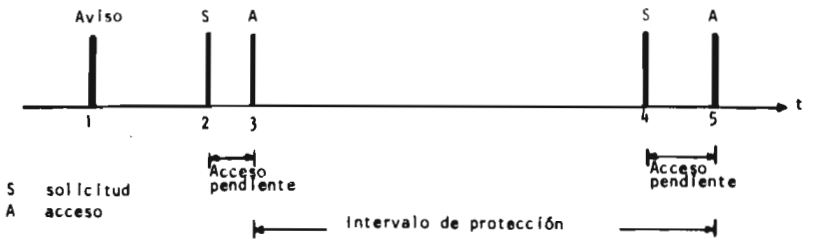


Fig 6.1 Secuencia de protección en un eje del tiempo

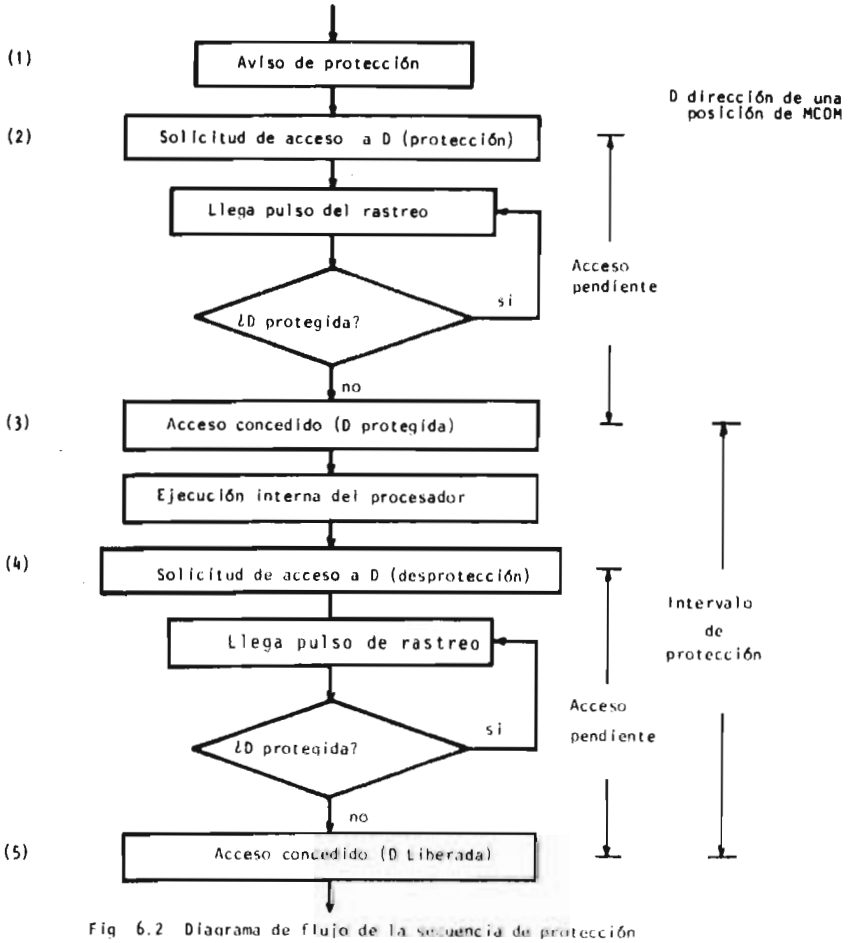


Fig 6.2 Diagrama de flujo de la secuencia de protección

6.3.1 Protección consecuente

Debe su nombre a que el acceso de desprotección es consecuente con el de protección. En el diagrama de flujo- (fig 6.2) ningún procesador protesta cuando alguno trata de hacer el acceso de desprotección, porque la posición es la que él mismo protegió.

6.3.2 Protección inconsecuente

En este caso se solicita el acceso de desprotección a una posición diferente de la que se protegió. Inicialmente el procesador protege la posición A, al solicitar el acceso de desprotección lo hace a la posición B (evento 4) -- ocasionando que se libere A y proteja B. En el intervalo - 3 → 4 (fig 6.2) se protege A, mientras que en el 4 → 5 se protege B.

Cuando se requiere proteger una posición por un -- tiempo definido, el intervalo de protección se puede controlar por programa, en forma precisa, protegiendo la posición solo el lapso 3 → 4, ya que no se tiene control sobre el -- 4 → 5 (protección inconsecuente). Sin embargo, la protec-- ción inconsecuente crea la posibilidad de llevar a punto -- muerto (detener para siempre la ejecución) dos procesado-- res, ya que podrían caer en una malla de la que no salen a- menos que se detenga el sistema. Un caso de este tipo se - muestra en el diagrama de flujo (fig 6.3) donde el procesador 1 realiza una protección inconsecuente que se traslapa- con la protección consecuente del procesador 2; cuando tratan de realizar el acceso de desprotección a la dirección 2 no pueden llevarlo a cabo porque ambos la protegen.

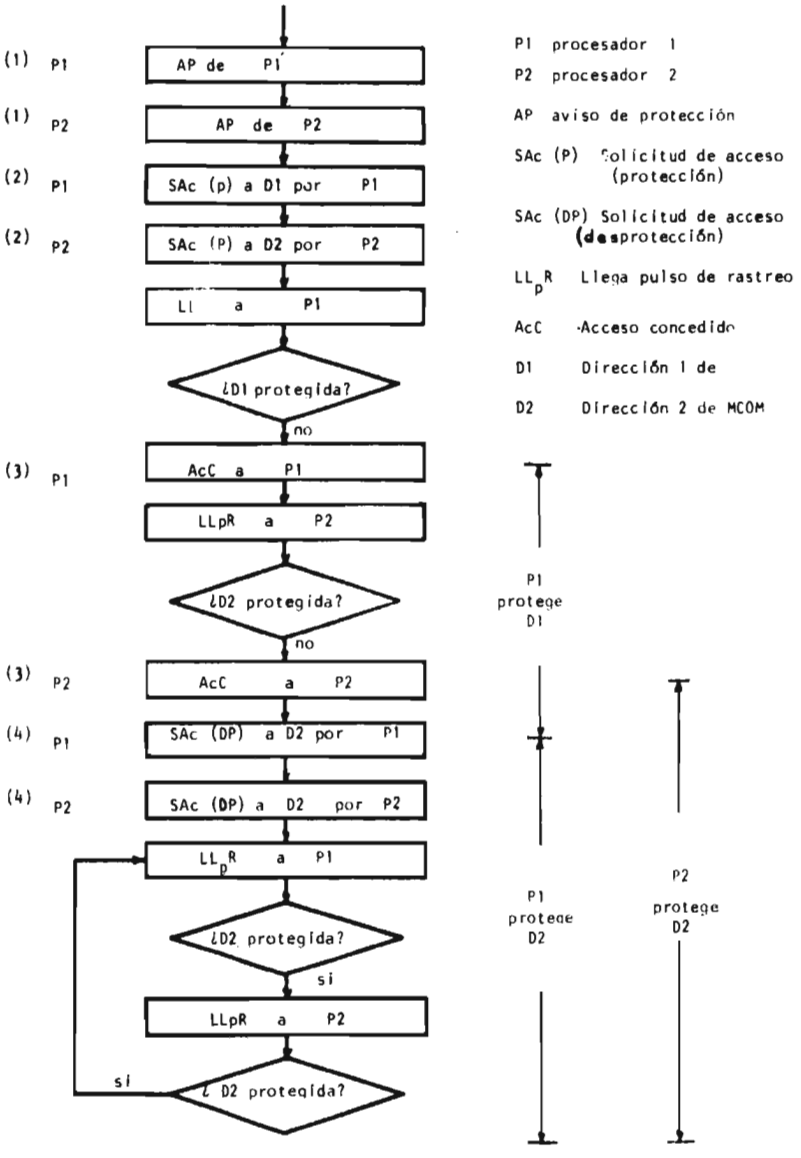


Fig 6.3 Posibilidad de llevar a punto muerto a P1 y P2

6.3.2.1 Solución a la protección inconsecuente

No debe permitirse que suceda un caso como el de la fig 6.3 porque el sistema pierde dos procesadores. El problema básico es evitar que se cambie el REGISTRO DE DIRECCION durante el intervalo de protección, para lo cual se debe utilizar un dispositivo que supervise, durante los intervalos de protección, cambios de estado en este registro; de esta manera, cuando suceda un cambio, automáticamente quita la protección. La solución ayuda a controlar, en forma precisa, el intervalo de protección cuando se desea proteger una posición por tiempo definido.

Para evitar esos problemas se sugiere que mientras no se realice esta modificación al acoplador se utilice el subsistema de protección únicamente en forma consecuente.

6.4 NUEVAS VERSIONES DEL SUBSISTEMA DE PROTECCION

Para dar al usuario la facilidad de utilizar el subsistema de protección por programa de la manera más provechosa posible, se proponen nuevas versiones del bloque PROTECTOR.

6.4.1 Protección de un bloque de magnitud fija

La idea del subsistema de protección, inicialmente concebida para proteger una posición de memoria, puede extenderse fácilmente para proteger bloques de memoria de magnitudes expresables como 2^x (x entero positivo); lo único que se debe hacer es no comparar los bits menos significantes de DIL y DIC (sec 4.4.3.3); por ejemplo, si se quiere proteger bloques de cuatro posiciones no deben compararse los dos bits menos significantes.

Cabe recordar que debe añadirse un dispositivo que supervise los accesos a memoria durante los intervalos de protección, ya que para evitar que dos procesadores lleguen a punto muerto, estos accesos no deben hacerse fuera del bloque que se protege.

6.4.2 Protección de un bloque de magnitud programable

En esta versión, a diferencia de la anterior, se tiene la facilidad de indicar por programa la magnitud del bloque que se desea proteger, que puede ser desde una hasta 256 posiciones. Esta nueva versión tan flexible del subsistema de protección requiere añadir un registro (con su decodificador) para almacenar el comando que proporciona la magnitud del bloque, que debe ser:

Comando	Bloque	Comando	Bloque
0000 0000	1	0000 1111	16
0000 0001	2	0001 1111	32
0000 0011	4	0011 1111	64
0000 0111	8	0111 1111	128
		1111 1111	256

Los "1" lógicos del comando determinan en el byte menos significante de DIL y DIC, los bits que no se comparan, por lo que se requiere un dispositivo programable que evite esas comparaciones. Los bits que se comparan determinan el orden del bloque que se protege; por ejemplo, si se protege un bloque de 8 posiciones y la dirección del acceso de protección es 45A3 (hexadecimal), entonces el bloque que se protege es: desde la dirección 45A0 hasta 45A7.

6.4.3 Intervalo de protección programable

Ya se mencionó que para alargar el intervalo de protección se debe realizar un aviso después de cada acceso, de esta manera, se prolonga dicho intervalo hasta que cesan los avisos. Sin embargo, esta actividad resulta lenta debido a la cantidad de avisos que hay que hacer, por lo que se propone una nueva versión para proteger un bloque (de magnitud fija) en la que se indique por programa, al momento de realizar el aviso, el número de accesos que se desea hacer durante el intervalo de protección. Para lograrlo se necesita cargar en un contador el número de accesos, el cual los decrementará a medida que se realicen y al momento de llegar a cero el contador deshabilita la protección. El número máximo de accesos que se puede lograr en un intervalo de protección con esta versión es 255.

6.4.4 Magnitud del bloque e intervalo de protección programables

Esta es una versión que conjunta las dos anteriores para hacer el subsistema de protección versátil y flexible, ya que se indica por programa la magnitud del bloque y el intervalo de protección.

7

CONCLUSIONES

Es indudable la importancia que tendrán en un futuro cercano los sistemas de multiprocesamiento. Enslow* predice que próximamente estaremos viviendo la "época del multiprocesamiento", en la que estos sistemas dominarán el mercado por completo dadas sus enormes ventajas en rendimiento, costo, confiabilidad y facilidad de integrarlos modularmente.

Como una pequeña aportación al advenimiento de los multiprocesadores modernos, el presente trabajo está enfocado a satisfacer uno de los requerimientos más comunes de estos: la transferencia de información entre procesadores (indispensable para que puedan interactuar procesos entre sí).

---

* Enslow, PH, "Multiprocessors and parallel processing", - Wiley-Interscience, Nueva York (1974)

De las experiencias teóricas y prácticas obtenidas durante el desarrollo del acoplador se llegaron a las siguientes conclusiones:

1. El diseño del acoplador, originalmente propuesto para la Máquina AHR, puede ser utilizado en sistemas de multiprocesamiento, procesamiento paralelo, distribuido y, en general, cualquier sistema donde se requiere que dos o más dispositivos compartan datos de una memoria digital.

2. La lógica del acoplador es totalmente transparente a los programas de los procesadores y su diseño modular facilita, el desarrollo de nuevas versiones para satisfacer diferentes necesidades sin alterar esa transparencia.

3. El subsistema de protección, congruente con el concepto modular del acoplador, ha sido implantado en forma distribuida a diferencia de algunos conocidos que son centralizados.

4. No hay restricciones lógicas para añadir o quitar procesadores y su facilidad para hacerlo es tal que solo se requiere abrir la trayectoria cerrada del pulso de rastreo, añadir o quitar los procesadores y volverla a cerrar, lo que se reduce a cambiar un cable que une dos terminales en el panel de conexiones.

5. El tiempo en investigar si una posición está protegida es muy corto 90×10^{-9} seg (máximo), que se puede reducir, utilizando lógica schottky a 40×10^{-9} seg (máximo).

6. La flexibilidad que se puede obtener con las nuevas versiones del subsistema de protección es muy útil ya que facilita al usuario utilizar el subsistema de la manera más provechosa posible.

7. Al utilizar lógica schottky, en comparación con los re-- sultados obtenidos, se observa que el rendimiento máximo de memoria se incrementa:

	Actual	Schottky
Rendimiento máximo de memoria	46.8%	67.8%
Porcentaje de tiempo perdido por transferencia del pulso de rastreo	16.6%	6.6%
Porcentaje de tiempo perdido por el subsistema de protección	15%	9.8%
Porcentaje de tiempo perdido por retardos internos del acoplador	21.6%	15.8%

8. El costo del material para el acoplador es muy bajo: --- $(MEX) 1,400.00, incluyendo tarjeta impresa, dos conectores y 43 circuitos integrados TTL (no requiere ninguno - especial).

9. El consumo de potencia es, también, bajo; se requiere -- únicamente una fuente de 5 VDC X 1 Amp = 5 Watts

10. Aun cuando el acoplador no tiene restricciones lógicas - para utilizar n procesadores, las limitaciones físicas - permiten el empleo de 88 procesadores (considerando unicamente lógica TTL).

A P E N D I C E A

CLASIFICACION DE PROCESADORES LSI

Debido a la variedad de microprocesadores (μPs) - o procesadores LSI que existe actualmente, se los puede clasificar de la siguiente manera:

A.1 PROCESADORES ORIENTADOS HACIA CALCULADORAS

National Semiconductor	5781, 82, 99, 140
Rockwell	MM76, MM77
Texas Instruments	TMS 1000, 1100, 1200
General Instrument	PIC 1650

Son controladores de bajo costo, en una sola pastilla incluyen ROM, RAM, puertos de I/O, reloj y la unidad de-procesamiento. El conjunto de instrucciones y los modos de direccionamiento son primitivos. Sus aplicaciones más importantes son: sintonizadores de radio, escalar valores, ins-- trumentos simples y cajas registradoras.

A.2 CONTROLADORES PARA FUNCIONES PARTICULARES

Intel	8048, 8041
Mostek	3870
Fairchild	F8/1,2
National Semiconductor	SC/MP I, II
Texas Instruments	9940

Tienen el control orientado hacia aplicaciones concretas y una capacidad de procesamiento limitada. En una sola pastilla incluyen el reloj, varios puertos de I/O, RAM, - y CPU. El conjunto de instrucciones incluye operaciones en

BCD, manejo de tablas y capacidad de saltos condicionales. - Usuados en equipo de control, bombeo de gas, marcadores telefonicos, terminales de venta y manejo de comunicaciones.

A.3 UNIDADES DE CARACTER GENERAL

Intel	8080
Motorola	6800
Signetics	2650
MOS Technology	6502
Rockwell	PPS-8
Texas Instruments	9980
RCA	1802
Intersil	6100
National Semiconductor	PACE
General Instruments	1600

Tienen una capacidad de procesamiento entre minicomputadora y controlador; tienden a tener un buen conjunto de instrucciones y modos de direccionamiento, las funciones de I/O las tienen integradas en la misma pastilla. Sus aplicaciones son muy variadas: instrumentos y terminales inteligentes, control de procesos, medición y pruebas, adquisición de datos, juegos, etc.

A.4 UNIDADES DE RENDIMIENTO ALTO

Zilog	Z80
Intel	8085
Texas Instruments	TMS, SBP9900

El conjunto de instrucciones, los modos de direccionamiento y el manejo de registros son similares a los de las minicomputadoras. El reloj y el control para multiples interrupciones se incluyen en una sola pastilla. Sus carac-

terísticas principales son: rendimiento bueno y velocidad alta de procesamiento.

A.5 ALGUNAS MICROCOMPUTADORAS

Digital Equipment	LSI-11
Data General	μNOVA
Intel	SBC 80/10, 20
Zilog	Z80 MCB
Mostek	SDB, OEM 80
Motorola	Monoboard

Incluyen en una sola tarjeta: reloj, CPU, memoria, puertos de I/O y el control para interrupciones. Tienen un rendimiento bueno, facilitan el diseño de sistemas digitales y la ampliación de memoria y puertos de I/O. Son utilizados en aplicaciones similares a las de minicomputadoras.

A.6 ELEMENTOS DE PROCESAMIENTO BIPOLAR LSI (BIT SLICE)

Intel	3002
American Micro Devices	2901
Motorola	10800
Fairchild	9405

Los "bit slice" (revanado por bit) son componentes de un ALU + grupo de registros, que manejan la lógica equivalente a un bit. Para implantar un CPU microprogramado se requieren varios "bit slice", registros rápidos (latches), amplificadores de corriente (buffers), transreceptores y lógica para manejar los acarreos. Son los dispositivos de mayor velocidad de procesamiento y flexibilidad, sin embargo, requieren gran cantidad de componentes adicionales. Son usados en la construcción de minicomputadoras, computadoras de caracter especial, controladores de disco, etc.

A P E N D I C E B

μPs QUE SE PUEDEN INCLUIR EN EL SISTEMA

Los elementos de procesamiento que se recomienda incluir en un sistema de multiprocesamiento construido con el acoplador, desarrollado en el presente trabajo, son microcomputadoras construidas con μPs de caracter general o de rendimiento alto. Es condición necesaria que estos tengan posibilidad de detener su funcionamiento (generar ciclos de espera) antes de escribir o leer un dato en memoria compartida.

La presente lista muestra algunos μPs que pueden ser utilizados, debido a que cumplen con esta característica:

Microprocesador	Se detiene con:	DMA con:	Se Recomienda
8080A	READY	HOLD	SI
Z80	WAIT	BUSRQ	SI
8085	READY	HOLD	SI
1802	CLEAR Y WAIT	CLEAR y WAIT	SI
	CLOCK	CLOCK	SI
6502	READY*		SI*
6800		HALT	NO
2650		PAUSE	NO

* El μP 6502 no espera en ciclos de escritura, por lo que se recomienda solo si se tiene la seguridad que no se darán los casos ESCRITURA SOBRE ESCRITURA (sec 2.3.3) y LECTURA SOBRE ESCRITURA (sec 2.3.4).

A P E N D I C E C

INSTRUCCIONES QUE SE PUEDEN EJECUTAR CON MEMORIA COMPARTIDA

Dado que la memoria compartida se utilizará exclusivamente para datos comunes a todos los μPs (los programas los almacena cada uno en su memoria privada), en la presente lista se muestra las instrucciones que pueden ser ejecutadas con esta memoria por los μPs 8080, 6502, Z80 y COSMAC 1802.

Se hace la aclaración que las abreviaturas que se utilizan en muchos casos, no corresponden al del fabricante, sin embargo, es facil relacionarlas conociendo la descripción de la instrucción.

Nomenclatura:

1	uno lógico	A	acumulador
0	cero lógico	c	bit de acarreo
00	cero aritmético	X	registro del indice X
$\wedge$	AND	Y	registro del indice Y
$\vee$	OR	M	contenido de una posición de memoria
$\veebar$	OR EXCLUSIVO		
N	dato de 8 bits	m_b	un bit del contenido de una posición de memoria.

R	registro de 8 bits de caracter general
H	registro de 8 bits del 8080 y Z80
L	registro de 8 bits del 8080 y Z80
R_H	registro de 8 bits B o D del Z80
R_L	registro de 8 bits C o E del Z80
M+1	contenido de la posición de memoria siguiente a la posición M.

(P) contenido de un puerto de I/O
(R') posición de memoria que direcciona el registro R'.

En Z80: R' = HL, IX o IY
En 1802: R' = R1, R2, ..., R16

m_7, ..., m_0 bits del contenido de la posición de memoria M
MSB LSB

MACRO Estas 2 instrucciones están compuestas de varias -- elementales*.

En Z80: LDI = LDI, LDD, LDIR, o LDDR
CPI = CPI, CPD, CPIR o CPDR
INP = INI, IND, INIR o INDR
OUT = OUTI, OTIR, OUTD, o OTDR

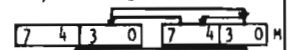
---

* Zilog, "Manual of Z80", 1977

TABLA C. 1 INSTRUCCIONES QUE SE PUEDEN EJECUTAR CON MEMORIA COMPARTIDA

TIPO	ABREVIAT	DESCRIPCION	8080	Z80	6502	COSMAC 1802
ARITMETICAS	ADD	$A \leftarrow A + M$	X	X		X
	ADC	$A \leftarrow A + M + C$	X	X	X	X
	SUB	$A \leftarrow A - M$	X	X		X
	SBC	$A \leftarrow A - M - \bar{C}$	X	X	X	X
	SUD	$A \leftarrow M - A$				X
	SDB	$A \leftarrow M - A - \bar{C}$				X
	CMP	$A - M$	X	X	X	
	CPX	$X - M$			X	
	CPY	$Y - M$			X	
	INC	$M \leftarrow M + 1$	X	X	X	
	DEC	$M \leftarrow M - 1$	X	X	X	
LOGICAS	AND	$A \leftarrow A \wedge M$	X	X	X	X
	ORA	$A \leftarrow A \vee M$	X	X	X	X
	XOR	$A \leftarrow A \veebar M$	X	X	X	X
	BIT	$A \wedge M$			X	
	BTN	$1 \wedge M_b$		X		
	SET	$M_b \leftarrow 1$		X		
	RES	$M_b \leftarrow 0$		X		
CARGA	LDA	$A \leftarrow M$	X	X	X	X
	LDX	$X \leftarrow M$			X	
	LDY	$Y \leftarrow M$			X	
	LDR	$R \leftarrow M$	X	X		
	LHL	$H \leftarrow M + 1;\quad L \leftarrow M$	X	X		
	LPA	$R_H \leftarrow M + 1;\quad R_L \leftarrow M$		X		

TABLA C.1 (Continuación)

TIPO	ABREVIAT	DESCRIPCION	8080	Z80	6502	COSMAC 1802
ALMACENAMIENTO	STA	$M \leftarrow A$	X	X	X	X
	STX	$M \leftarrow X$			X	
	STY	$M \leftarrow Y$			X	
	STR	$M \leftarrow R$	X	X		
	SHL	$M+1 \leftarrow H$; $M \leftarrow L$	X	X		
	SPA	$M+1 \leftarrow R_H$; $M \leftarrow R_L$		X		
	STN	$M \leftarrow N$		X		
CORRIMIENTO Y ROTACION	ASL	$c \leftarrow m_7, \ldots, m_0 \leftarrow 0$		X	X	
	ASR	$m_7 \rightarrow m_7, \ldots, m_0 \rightarrow c$		X		
	LSR	$0 \rightarrow m_7, \ldots, m_0 \rightarrow c$		X	X	
	ROL	$c \leftarrow m_7, \ldots, m_0 \leftarrow c$		X	X	
	ROR	$c \rightarrow m_7, \ldots, m_0 \rightarrow c$		X	X	
	RLC	$c \leftarrow m_7, \ldots, m_0 \leftarrow m_7$		X		
	RRC	$m_0 \rightarrow m_7, \ldots, m_0 \rightarrow c$		X		
	RLD			X		
	RRD			X		
MACRO	LDI	$(DE) \leftarrow (HL)$; $DE \leftarrow DE \pm 1$; $HL \leftarrow HL \pm 1$; $BC \leftarrow BC \pm 1$		X		
	CPI	$A \leftarrow (HL)$; $HL \leftarrow HL \pm 1$; $BC \leftarrow BC \pm 1$		X		
I/O	INP	$(R') \leftarrow (P)$		X		X
	OUT	$(P) \leftarrow (R')$		X		X

A P E N D I C E D

REFERENCIAS A MCOM RESPECTO A LOS CICLOS DE MAQUINA

El objeto del apéndice es mostrar la relación:

REFERENCIAS A MCOM / CICLOS DE MAQUINA

de las instrucciones del Apéndice C en sus diferentes modos de direccionamiento.

No hay uniformidad en los fabricantes para denominar los modos de direccionamiento, por lo que se trata de homogenizarlos de la siguiente manera:

DIRECTO:

DIRECTO	en 8080
ABSOLUTO	en 6502
EXTENDIDO	en Z80

POR REGISTRO:

La posición de memoria la direcciona el registro R

En Z80 y 8080	R = Registro par HL
En COSMAC 1802	R = R1, R2, ..., R15 o R16

OTROS REGISTROS:

Lo mismo que en el caso anterior pero R es diferente:

En 8080 y Z80	R = Registro par BC o DE

X o Y CON DESPLAZAMIENTO:

Un desplazamiento "d" es sumado al registro IX o IY de Z80 para obtener la dirección del acceso.

BLOQUES:

Muestra el número de ciclos de máquina para las instrucciones MACRO y de I/O cuando el registro par BC y el - son diferentes de cero.

TABLA D.1 RELACION: REFERENCIAS A MCOM/CICLOS DE MAQUINA

INSTRUCCION		DIRECTO			ABSOL X	ABSOL Y	INDIR X	INDIR Y	POR REGISTRO			OTROS REGISTROS		X o Y DESPL	BLO-- QUES
TIPO	ABREVIAT	8080	Z80	6502	6502	6502	6502	6502	8080	Z80	COSMAC 1802	8080	Z80	Z80	Z80
ARITMETICAS	ADD								1/2	1/2	1/2			1/5	
	ADC			1/4	1/4	1/4	1/6	1/5	1/2	1/2	1/2			1/5	
	SUB								1/2	1/2	1/2			1/5	
	SBC			1/4	1/4	1/4	1/6	1/5	1/2	1/2	1/2			1/5	
	SUD										1/2				
	SAB										1/2				
	CMP			1/4	1/4	1/4	1/6	1/5	1/2	1/2				1/5	
	CPX			1/4											
	CPY			1/4											
	INC			2/6	2/7				2/3	2/3				2/6	
	DEC			2/6	2/7				2/3	2/3				2/6	
LOGICAS	AND			1/4	1/4	1/4	1/6	1/5	1/2	1/2	1/2			1/5	
	ORA			1/4	1/4	1/4	1/6	1/5	1/2	1/2	1/2			1/5	
	XOR			1/4	1/4	1/4	1/6	1/5	1/2	1/2	1/2			1/5	
	BIT			1/4											
	BTN									1/3				1/5	
	SET									2/4				2/6	
	RES									2/4				2/6	
CARGA	LDA	1/4	1/4	1/4	1/4	1/4	1/6	1/5	1/2	1/2	1/2	1/2	1/2	1/5	
	LDX			1/4		1/4									
	LDY			1/4	1/4										

TABLA D.1 (Continuación)

INSTRUCCION		DIRECTO			ABSOL X	ABSOL Y	INDIR X	INDIR Y	POR REGISTRO			OTROS REGISTROS		X o Y DESPL	BLOQUES
TIPO	ABREVIAT	8080	Z80	6502	6502	6502	6502	6502	8080	Z80	COSMAC 1802	8080	Z80	Z80	Z80
CARGA	LDR								1/2	1/2				1/5	
	LHL	2/5	2/5												
	LPA		2/6												
ALMACENAMIENTO	STA	1/4	1/4	1/4	1/5	1/5	1/6	1/6	1/2	1/2	1/2	1/2	1/2	1/5	
	STX			1/4											
	STY			1/4											
	STR								1/2	1/2				1/5	
	SHL	2/5	2/5												
	SPA		2/6												
	STN									1/3				1/5	
CORRIMIENTO Y ROTACION	ASL			2/6	2/7					2/4				2/6	
	ASR									2/4				2/6	
	LSR			2/6	2/7					2/4				2/6	
	ROL			2/6	2/7					2/4				2/6	
	ROR			2/6	2/7					2/4				2/6	
	RLC									2/4				2/6	
	RRC									2/4				2/6	
	RLD									2/5					
	RRD									2/5					
MACRO	LDI									2/4					2/5
	CPI									1/4					1/5
I/O	INP									1/4	1/2				1/5
	OUT									1/4	1/2				1/5