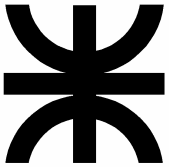


UNIVERSIDAD TECNOLOGICA NACIONAL

FACULTAD REGIONAL BUENOS AIRES

Departamento de Electrónica

Cátedra: Técnicas Digitales II - Plan 1995 - Versión 2006

CAPITULO II: Un Microprocesadores Comercial de 8 bits

Autor: Ing. Marcelo E. Romeo

meromeo@frba.utn.edu.ar

Tabla de Contenidos

1. Introducción a los microprocesadores de la familia Intel 8088/8086

En 1973 Intel, una empresa controlada por ex integrantes del grupo de profesionales que produjo en Fairchild el primer circuito integrado digital, fabricó el primer microprocesador comercial de 8 bits, el 8080.

Poco después, en 1975, se introdujo una segunda versión de microprocesador comercial denominada 8085, que solucionó dos de los principales inconvenientes que presentaba el 8080:

- El 8085 necesita una sola tensión de alimentación de +5 V (el 8080 requería +5, -5 y +12 Volt).
- El 8085 es un microprocesador en un solo circuito, mientras que el 8080, por limitaciones tecnológicas (cantidad de patas y cantidad de componentes por integrado), requería de dos circuitos integrados adicionales para poder operar.

Probablemente el concepto más importante que introdujo Intel fue el de ***compatibilidad hacia arriba (upward compatible),*** que trataría de mantener con todas sus familias de microprocesadores. Ésto quiere decir que el programa desarrollado para un microprocesador previo, se ejecuta similarmente (seguramente en menos tiempo) en un microprocesador posterior, vale decir que no será necesario rescribir el programa cada vez que se pasa a un nuevo escalón tecnológico.

En junio de 1978 Intel lanzó al mercado el primer microprocesador comercial de 16 bits en el bus de datos: el 8086.

En aquel entonces, las memorias eran costosas, por lo que un sistema mínimo de 16 bits requería dos memorias de programa (una para los 8 bits más significativos y otra para los menos significativos del bus de datos) y dos memorias de lectura / escritura.

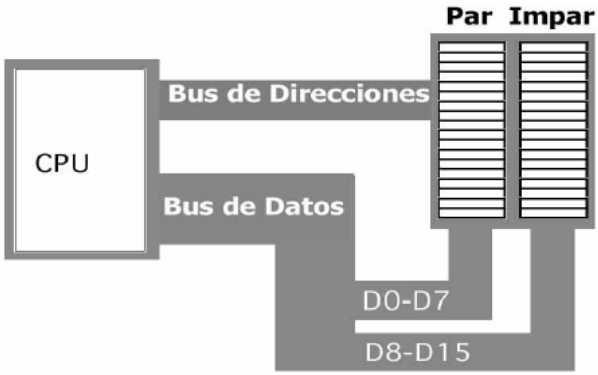


Fig. 1. Conexión de un microprocesador de 16 bits a las memorias.

Como se observa en la Fig. 1, una memoria de 8 bits de ancho, estaba conectada sobre las líneas más significativas del bus de datos (IMPAR), mientras que otra memoria se conectaba sobre los bits menos significativas del bus de datos.[1]

Por tal motivo, en junio de 1979 apareció el 8088, internamente igual que el 8086 y totalmente compatible a nivel de repertorio de instrucciones con su predecesor, pero con bus externo de datos de 8 bits, con el que un sistema mínimo necesitaba solamente una memoria de programa y otra de datos. El 8088 era, a nivel de conexionado con el mundo exterior, similar al 8085, por lo que la actualización circuital no presentó grandes inconvenientes y fue rápidamente aceptada por el mercado.

---

[1] Como la estructura de memoria de todos los microprocesadores de Intel se hizo bajo la organización de bytes, una palabra de 16 bits que se escriba en la dirección 0, ocupará los bytes 0 (PAR) y 1 (IMPAR).

En 1980 aparecieron dos componentes que completaron la familia e introdujeron el concepto de *coprocesador*[2]. Los coprocesadores 8087[3] (matemático) y 8089 (de entrada y salida).

2. Descripción de las patas de conexión del 8088

El 8086 es un microprocesador de 16 bits, tanto en lo que se refiere a su estructura como en sus conexiones externas, mientras que el 8088 es un procesador de 16 bits en el bus interno de datos y de 8 bits en el bus externo de datos. Internamente es casi idéntico al 8086. Las diferencias entre ambos son

- El tamaño del bus de datos externo.
- El tamaño de la cola de almacenamiento de instrucciones.
- La señal IO/*M o M/*IO según el caso.

El 8088 tiene un bus de datos externo reducido de 8 bits. La razón para ello era prever la continuidad entre el 8086 y los antiguos procesadores de 8 bits, como el 8080 y el 8085. Teniendo el mismo tamaño del bus (así como similares requerimientos para las señales de control y temporización), el 8088, que es internamente un procesador de 16 bits, pudo reemplazar con moderado esfuerzo a los microprocesadores ya nombrados en un sistema ya existente.

El 8088 tiene muchas señales en común con el 8085, particularmente las asociadas con la forma en que los datos y las direcciones están multiplexadas, aunque el 8088 no produce sus propias señales de reloj como lo hace el 8085 (necesita un circuito integrado de soporte llamado 8284).

El 8088 y el 8085 comparten la misma política de multiplexar en el tiempo las líneas del bus de datos con la parte baja del bus de direcciones, indicando por medio de una señal auxiliar ALE (Address Latch Enable) en que momento sobre ese bus se hallan presentes las direcciones a fin de retenerlas por medio de un latch externo. Por este método se ahorran 8 terminales para otras funciones del microprocesador. El 8086 comparte los 16 bits del bus de datos con los 16 más bajos del bus de direcciones.

El 8085 y el 8088 pueden, de hecho, dirigir directamente los mismos integrados controladores de periféricos. Los desarrollos de hardware para sistemas basados en el 8080 o el 8085 son, en su mayoría, aplicables al 8088.

Todo ésto colaboró para el éxito del 8088.

El 8086/8088 puede conectarse al circuito de dos formas distintas: el modo máximo y el modo mínimo. El modo queda determinado al poner un determinado terminal (llamado MN/MX*) a "0" o a la tensión de alimentación. El 8086/8088 debe estar en modo máximo si se desea trabajar en colaboración con el Procesador de Datos Numérico 8087 y/o el Procesador de Entrada/Salida 8089 (de aquí se desprende que en la IBM PC el 8088 está en modo máximo). En este modo el 8086/8088 depende de otros chips adicionales como el Controlador de Bus 8288 para generar el conjunto completo de señales del bus de control. El modo mínimo permite al 8086/8088 trabajar de una forma más autónoma (para circuitos más sencillos) en una manera casi idéntica al microprocesador 8085.

---

2 Coprocesador es un circuito integrado que realiza funciones complementarias al procesador principal (por ejemplo actividades matemáticas), pero que necesita imprescindiblemente del microprocesador, pues no tiene posibilidades de operar autónomamente, ya que, por ejemplo no tiene bus de direcciones, sino que necesita que el procesador principal direccione una posición de memoria o de entrada/salida

3 Este coprocesador se necesitó para cierto tipo de aplicaciones científicas en las que era inaceptable el tiempo que demoraba la realización de operaciones trigonométricas o trascendentes por programa. No se justificaba incorporar esas prestaciones al procesador principal, pues el incremento de costo no se justificaba para las aplicaciones comunes. Recién con la aparición del 80486, debido a la disminución de costos del silicio integrado, se incorporó el coprocesador al procesador principal.

2.1 Descripción de las patas en modo mínimo

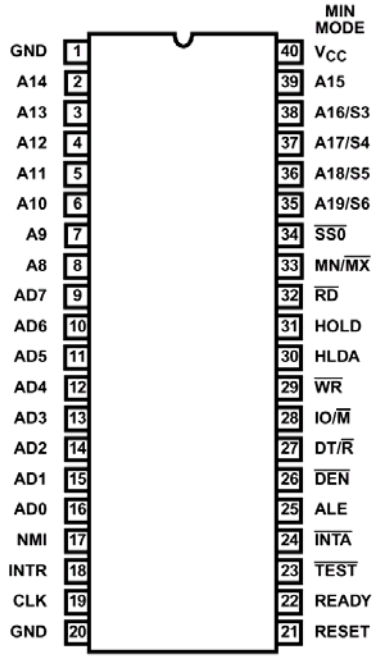


Fig. 2. **Esquema del 8088 en modo mínimo.**

Las 40 patas del 8088 se pueden dividir funcionalmente en cuatro grupos:

- Líneas de direcciones.
- Líneas de datos.
- Líneas de control y estado.
- Líneas de alimentación y temporización.

Las 40 patas de conexión del 8088 se sintetizan en la **Tabla 1.** Descripción de las patas del 8088 en modo mínimo:

Pata	Ent/Sal	Nombre	Descripción
1	Aliment.	GND	Referencia de la alimentación.
2	Salida	A14	Línea 14 del bus de direcciones (Unidireccional y estable en el tiempo).
3	Salida	A13	Línea 13 del bus de direcciones (Unidireccional y estable en el tiempo).
4	Salida	A12	Línea 12 del bus de direcciones (Unidireccional y estable en el tiempo).
5	Salida	A11	Línea 11 del bus de direcciones (Unidireccional y estable en el tiempo).
6	Salida	A10	Línea 10 del bus de direcciones (Unidireccional y estable en el tiempo).
7	Salida	A9	Línea 9 del bus de direcciones (Unidireccional y estable en el tiempo).
8	Salida	A8	Línea 8 del bus de direcciones (Unidireccional y estable en el tiempo).
9	Bidirec.	AD7	Línea de direcciones 7 multiplexada con la línea de datos 7. En el primer ciclo de reloj de un ciclo de máquina de lectura o escritura, se presenta A7. Durante el resto de los ciclos de reloj, aparecerá D7 o bien estará en alta impedancia, según el requerimiento del ciclo de máquina.
10	Bidirec.	AD6	Línea de direcciones 6 multiplexada con la línea de datos 6
11	Bidirec.	AD5	Línea de direcciones 5 multiplexada con la línea de datos 5

12	Bidirec.	AD4	Línea de direcciones 4 multiplexada con la línea de datos 4
13	Bidirec.	AD3	Línea de direcciones 3 multiplexada con la línea de datos 3
14	Bidirec.	AD2	Línea de direcciones 2 multiplexada con la línea de datos 2
15	Bidirec.	AD1	Línea de direcciones 1 multiplexada con la línea de datos 1
16	Bidirec.	AD0	Línea de direcciones 0 multiplexada con la línea de datos 0
17	Entrada	INTR	Entrada de interrupción vectorizada y enmascarable (a ser estudiada en el capítulo 3).
18	Entrada	NMI	Entrada de interrupción vectorizada y ***NO*** enmascarable (ver capítulo 3).
19	Entrada	CLK	Entrada de reloj generada por el 8284. Su frecuencia es la tercera parte de la frecuencia de resonancia del cristal piezoeléctrico que se conecta sobre dicho integrado y su ciclo de actividad deberá ser de 0,333. En las primeras versiones, la frecuencia máxima era de 5 MHz y posteriormente aparecieron versiones *turbo* de 8 y 12 MHz. Existe una frecuencia mínima que es la que permite el refresco de los registros dinámicos (ver capítulo 2).
20	Aliment.	GND	Referencia de la alimentación
21	Entrada	RESET	Señal de reset del sistema. Es una entrada proveniente del 8284 y debe estar en alto por lo menos por 4 ciclos de reloj (salvo cuando se enciende el sistema, que para dar tiempo a que se polarice el sustrato del integrado, deberá estar en alto por lo menos por 50 μs. El 8284 recibe una señal (activa baja) y la sincroniza y retransmite para el 8088. Cuando esta señal vuelve a "0", se producen los siguientes hechos: • Registro de flags ⇒ 0000H • Registros DS, SS, ES e IP ⇒ 0000H (Se analizarán en este mismo capítulo) • Registro CS ⇒ FFFFH (Se analizará posteriormente).
22	Entrada	READY	Se emplea para sincronizar al microprocesador con memorias o dispositivos de E/S lentos. Ya que no puede acelerárselos, se frena al microprocesador hasta que dichos dispositivos se hallen en condiciones de culminar la transacción de información. Para la operación a máxima velocidad, la señal deberá estar en "1". Si se desea acceder a una memoria lenta, se deberá activar un circuito (diseñado por el usuario) que producirá una señal "0". Si la señal esta en dicho valor 8 ns antes de finalizar el segundo ciclo de reloj, del ciclo de máquina de lectura o escritura, se prolongará el ciclo de máquina agregando tiempos de espera (wait states, medidos en ciclos de reloj), hasta que la señal READY pase a valer "1". Esta señal es provista por el 8284, que dispone de dos entradas /RDY1 y /RDY2 que sincroniza para producir la entrada al 8088.
23	Entrada	/TEST	Es una señal cuyo estado es tenido en cuenta sólo por la instrucción ***WAIT***. Cuando esta instrucción se ejecuta, el microprocesador se detendrá hasta que la señal /TEST valga "0". Se emplea para que el microprocesador sepa que, por ejemplo el coprocesador aritmético culminó una operación cuyo resultado es necesitado por el 8088.
24	Salida	/INTA	Reconocimiento de interrupción (**In**terrup**t** **A**cknowledge). Es una señal que le informa al mundo exterior que el microprocesador aceptó un pedido de interrupción solicitado por algún dispositivo de E/S. Su actuación y temporización se verán en el capítulo 3.
25	Salida	ALE	Habilitación del latch de direcciones (**A**ddress **L**atch **E**nable). Cuando se halla en "1", indica que el contenido del bus multiplexado de datos y direcciones contiene direcciones. Cuando vale "0", dicho bus contendrá datos o bien estará en el estado de alta impedancia.
26	Salida	/DEN	Habilitación para datos (**D**ata **En**able). Es similar a ALE pero para datos. Cuando esta activa (en "0"), indica que sobre el bus multiplexado aparecerán datos, provistos por el microprocesador o por la memoria o dispositivo de E/S[4]. Se

[4] Nótese que cuando ALE = 0 no hay direcciones y cuando /DEN = 1 no hay datos sobre el bus multiplexado. Si se dan ambas circunstancias simultáneamente, es bus estará en alta impedancia.

			emplea para el caso de que sea necesario emplear trasceptores (buffers) bidireccionales (tipo 74HC245) sobre el bus de datos. Esta señal se utiliza para sacar al buffer del estado de alta impedancia.
27	Salida	DT/R*	Transmisión o recepción de datos (Data Transmit or Receive). Los transceptores anteriormente mencionados tienen una pata de sentido (desde o hacia el microprocesador). Esta señal comanda dicho sentido.
28	Salida	IO/M*	El 8088 puede direccionar 1 M byte de memoria o hasta 64 K puertas (o dispositivos) de entrada o salida. Esta señal es la que suele ingresar en la decodificación y habilitar dispositivos de entrada salida. Al realizar operaciones con posiciones de memoria, toma el valor "0" y solo valdrá "1" cuando se ejecuten dos instrucciones de entrada /salida, llamada IN y OUT.[5]
29	Salida	/WR	Write o escritura. Indica escritura hacia memoria o E/S.
30	Salida	HLDA	Respuesta afirmativa, aceptando el pedido de HOLD.
31	Entrada	HOLD	Es un pedido de algún dispositivo (controlador de acceso directo a memoria o DMA, como se verá en el capítulo 3) que quiere adueñarse del control de los buses, pasando los mismos al estado de alta impedancia.
32	Salida	/RD	Read o lectura. Indica lectura de memoria o E/S.
33	Entrada	MN/MX*	Mínimo o máximo. Cuando esta entrada está en estado alto, el 8088 está en modo mínimo (similar a un 8085 más potente), en caso contrario está en modo máximo)
34	Salida	/SSO	Bit de estado 0. Junto con IO/M y DT/R esta salida sirve para determinar el estado (es decir que esta haciendo) el 8088.
35	Salida	A19/S6	Línea de direcciones 19, multiplexada con el bit de estado 6.
36	Salida	A18/S5	Línea de direcciones 18, multiplexada con el bit de estado 5.
37	Salida	A17/S4	Línea de direcciones 17, multiplexada con el bit de estado 4.
38	Salida	A16/S3	Línea de direcciones 16, multiplexada con el bit de estado 3.
39	Salida	A15	Línea 15 del bus de direcciones (Unidireccional y estable en el tiempo).
40	Aliment.	Vcc	Alimentación del microprocesador (+5V).

Tabla 1. Descripción de las patas del 8088 en modo mínimo

2.2 Una Unidad Central de proceso básica en modo mínimo

[5] Una de las diferencias entre el 8088 y el 8086 (además del tamaño del bus externo de datos) es que en el 8086, la señal equivalente se llama M/IO* y vale "1" cuando se realizan operaciones con memoria. La forma que tomó esta señal en el 8088 fue para mantener compatibilidad con los microprocesadores de 8 bits antecesores.

Como Resumen, en la Fig. 3 se observa la parte central de una microcomputadora basada en un 8088 operando en modo mínimo. Se presenta parte de la interconexión del microprocesador con el circuito de reloj 8284 y el circuito RC que se encarga del reset al encendido (Power on reset) del mismo.

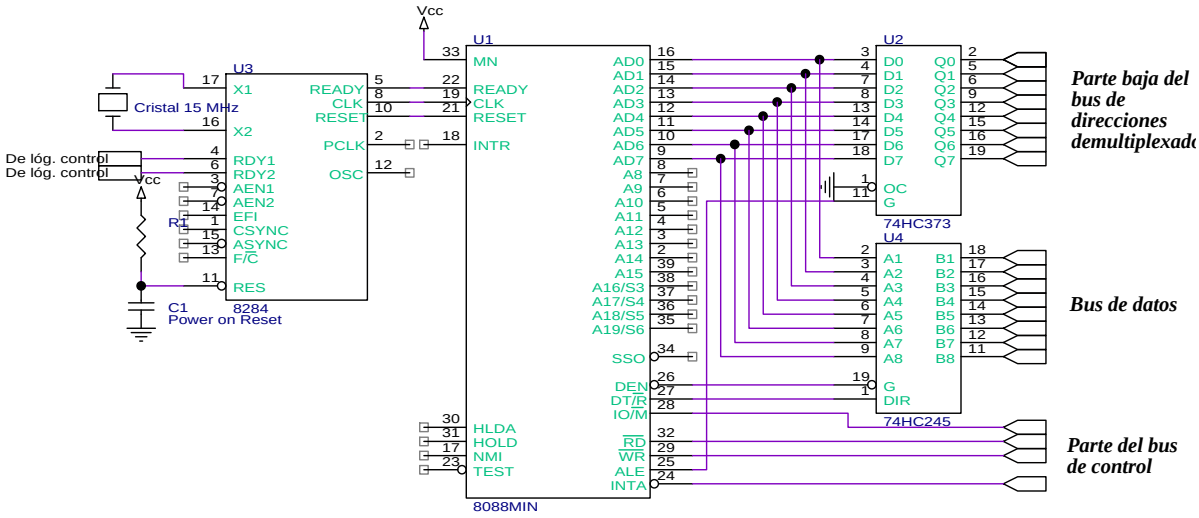


Fig. 3. Circuito central de una CPU con un 8088 en modo mínimo.

Se deberá instalar un latch activo por nivel alto 74HC373 (ó 74ALS373) para demultiplexar la parte baja del bus de direcciones $AD_{0\text{-}7}$. Mientras el microprocesador coloca direcciones sobre el bus multiplexado, mantiene la señal ALE activa. Esa señal maneja el control del latch, de forma que retenga las direcciones cuando ALE se desactiva y pasa a presentar datos.

Algo similar puede realizarse para demultiplexar la parte más alta $A_{16\text{-}19}$ empleando otro 74HC373, controlado también con la señal ALE.

En algún momento existieron componentes (8155 y 8755) que recibían directamente el bus multiplexado e internamente efectuaban la separación entre datos y parte baja de direcciones. Estos dispositivos permitían ahorrar un componente (latch) pero debido a que no tuvieron aceptación comercial, su costo no se redujo y se hicieron prohitivos.

Puede darse el caso que sea necesario colocar buffers sobre las líneas estables de direcciones $A_{8\text{-}15}$, en tal caso se suelen emplear 74HC244, habilitados en forma permanente.

2.3 Descripción de las patas en modo máximo

El 8088 descrito en el punto anterior, no es más que un 8085 mejorado, con mayor capacidad de direccionamiento, mayor repertorio de instrucciones y más rápido. Es un microprocesador pensado para trabajar en forma autónoma y con la única compañía del generador de reloj 8284 y memorias y dispositivos de E/S estándar.

Al conectar la pata MN/MX a "0" lógico, el 8088 se transforma en el corazón de un sistema más complejo.

En efecto, varias de las patas descriptas recientemente, cambian su función y pasan a entregar información codificada que será interpretada por un conjunto de dispositivos conexos y se formará en su derredor un verdadero sistema de micro cómputo de aplicaciones profesionales.

Antes de analizar este modo de trabajo, tratemos de ubicarnos en la realidad tecnológica de los finales de la década del '70. Sólo se podían encapsular circuitos de hasta 40 patas (tanto el Z8000 de Zilog -con 48 pines-, como el 68000 de Motorola -con 64 pines-, aparecieron en 1980), por tal motivo, Intel decidió generar el modo máximo codificando la información para que los circuitos de extensión se encargaran de producir más señales que las que permitirían las 40 patas.

Los circuitos integrados adicionales son el 8288, llamado Controlador del Bus y que es imprescindible para cualquier aplicación que requiera modo máximo (podría ser reemplazado por lógica discreta, pero su tamaño, complicación y costo hacen que dicho reemplazo no sea recomendable) y el 8289, llamado Arbitrador de Bus y que es un componente opcional que se requiere cuando se trabaja en la modalidad multiprocesador (varias microcomputadoras compartiendo recursos y trabajando en paralelo).

Las patas que cambian de función son las siguientes:

Pata	Ent/Sal	Nombre en Modo Mínimo	Nombre en Modo Máximo	Descripción
26	Salida	/DEN	/S0	Conjuntamente con /S1 y /S2, provee información codificada de estado o status (lo que se halla haciendo el microprocesador en este instante), como se indica en la Tabla 3.
27	Salida	DT/R*	/S1	Provee información codificada de status.
28	Salida	IO/M*	/S2	Provee información codificada de status.
29	Salida	/WR	/LOCK	Es una señal que es hecha valer "0" cuando el microprocesador ejecuta la instrucción LOCK (en realidad, veremos que se trata de un prefijo). Esa instrucción se coloca en un programa para evitar que un tramo del mismo sea cortado en su ejecución por un eventual pedido de control del bus por parte de los demás dispositivos inteligentes que comparten los buses. Cuando vale cero indica a otros controladores del bus (otros microprocesadores o un dispositivo de acceso directo a memoria DMA) que no deben ganar el control de los buses.
25	Salida	ALE	QS0	Conjuntamente con QS1 indican el estado de la cola de instrucciones del microprocesador. El significado de la combinación esta dado por la Tabla 6
24	Salida	/INTA	QS1	Estado de la cola de instrucciones
31	Bidireccional	HOLD	/RQ/GT0*	El significado de esta línea es: Pedido (Request) y Confirmación (Grant). La idea es la de reemplazar la secuencia HOLD⇒HOLDA del modo mínimo por un mecanismo más efectivo. El periférico que desea utilizar los buses del microprocesador, produce un pulso activo bajo en esta línea *(Request).* Cuando el microprocesador esta dispuesto, entra al estado de Hold y genera por esta misma línea otro pulso activo bajo hacia el periférico otorgándole es uso de los buses *(Grant).* Al finalizar su actuación, el periférico produce otro pulso activo bajo y el microprocesador recupera el control de los buses.
30	Bidireccional	HLDA	/RQ/GT1*	Igual al /RQ/GT0*. Por este mecanismo de Pedido⇒Confirmación se puede manejar hasta

				dos dispositivos que soliciten el control de los buses.

Tabla 2. Descripción de las patas del 8088 en modo máximo

Las tres líneas de status indican que estado de máquina se está ejecutando. Con esa información el 8288 generará las señales "naturales" para las memorias y dispositivos (/RD, /WR, IO/M*, etc.), como las que se vieron para el núcleo básico de la microcomputadora en modo mínimo.

/S2	/S1	/S0	Ciclo de máquina
0	0	0	Reconocimiento de interrupción (Ver Cap. 3)
0	0	1	Lectura de E/S
0	1	0	Escritura de E/S
0	1	1	Halt
1	0	0	Búsqueda de código de operación
1	0	1	Lectura de Memoria
1	1	0	Escritura de Memoria
1	1	1	Inactivo

Tabla 3. Ciclos de máquina indicados por la codificación de las líneas de status.

Hemos dicho que las líneas QS0 y QS1 marcan el estado de la memoria interna denominada cola. Esa información, debe interpretarse según se indica en la **Tabla 6.** Significado del estado de la cola, indicado por QS0 y QS1

QS0	QS1	Descripción
0	0	Ninguna Operación
0	1	Se esta ejecutando el primer byte de una instrucción.
1	0	La cola se está vaciando
1	1	Se esta tomando un byte subsiguiente de la instrucción.

Tabla 4. Significado del estado de la cola, indicado por QS0 y QS1

2.4 Unidad Central de Proceso básica trabajando en modo máximo.

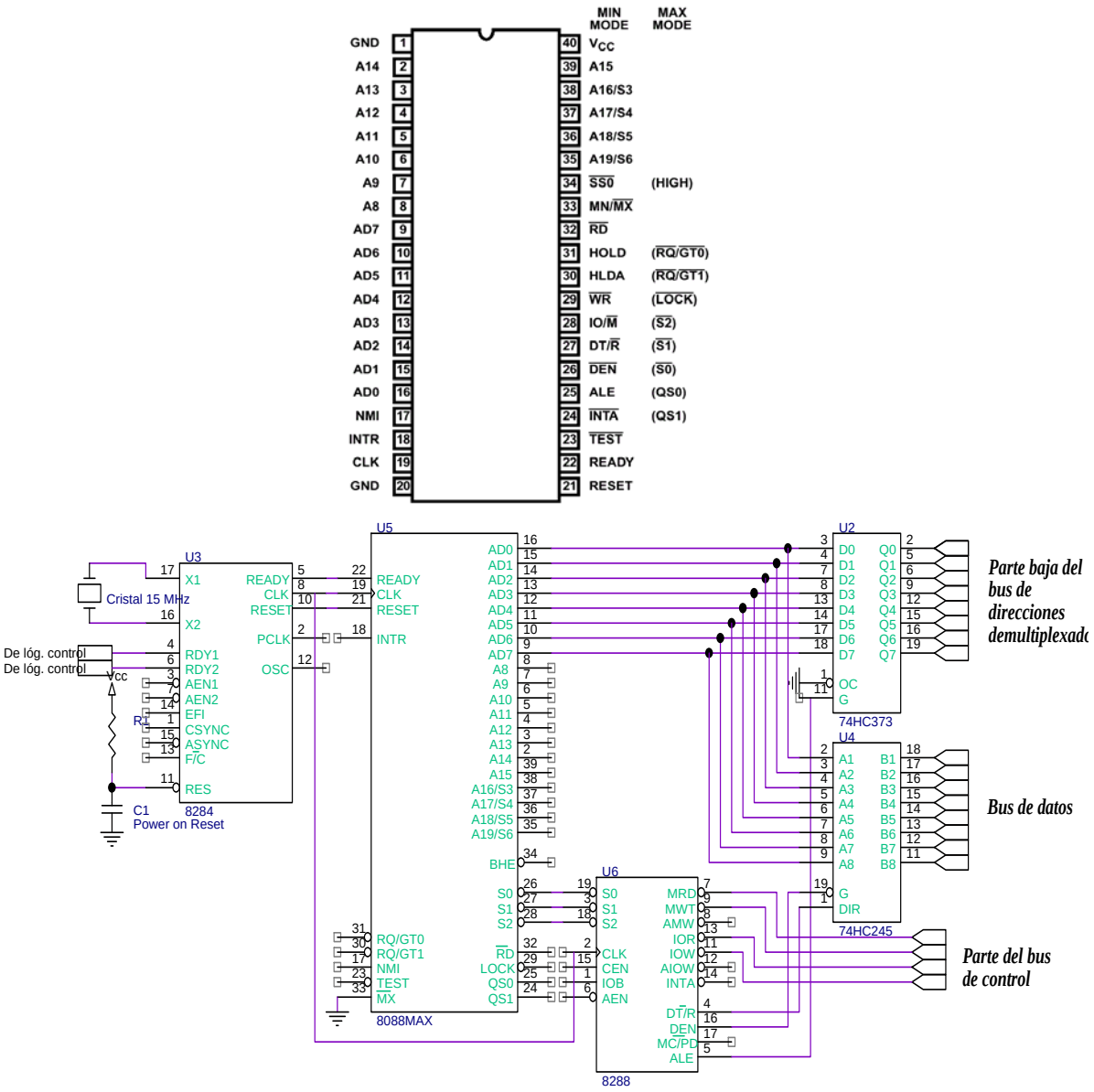


Fig. 4. a) Esquema del 8088 en modo máximo y b) Circuito básico del 8088 en modo máximo.

En el circuito de la Fig. 4, se observa el núcleo de una microcomputadora operando en modo máximo. Se puede notar la aparición del circuito integrado ***controlador del bus*** 8288, que recibe las señales de estado /S0, /S1 y /S2 y se encarga de generar, entre otras, las señales de control ALE, DEN y DT/R* que serán empleadas por la lógica de decodificación y los demás circuitos integrados adicionales.

2.5 Circuito Integrado en formato PLCC.

Con el advenimiento de la necesidad de miniaturizar los circuitos impresos y aprovechando las mejoras tecnológicas en los encapsulados, paulatinamente se fue reemplazando el tradicional formato dual – in – line (DIP) en los que los contactos del integrado se hallan en ambos lados del mismo, por otros en los que se aprovecha mucho mejor el área.

Uno de esos formatos es el PLCC (Plastic leaded chip carrier). En el mismo, los contactos se hallan en los cuatro lados del integrado, constituido por pequeños flejes metálicos elásticos. El integrado va montado (NO se suelda) en un zócalo que tiene unos flejes similares que hacen contacto a presión con los del integrado asegurando una excelente conexión aún en circuitos sujetos a vibraciones y movimientos.

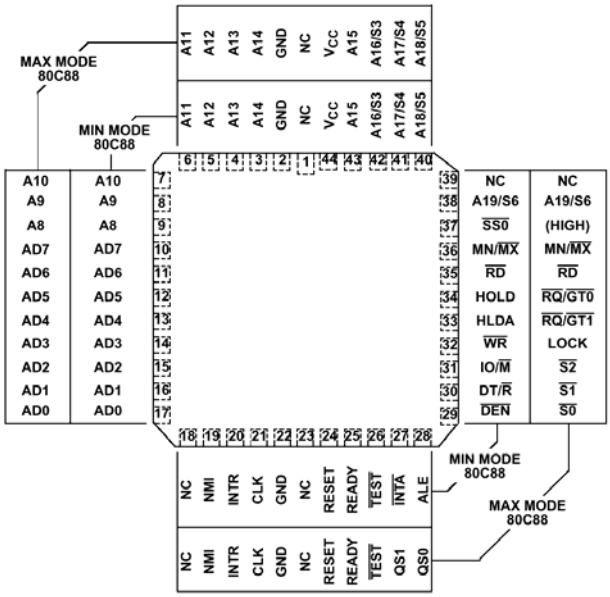


Fig. 5. 8088 en formato PLCC con las distintas señales para modo mínimo y máximo

2.6 Bus Local y Multimaster.

Según lo recientemente descrito, la operación en modo máximo de una microcomputadora, busca generar información (que por restricciones en la cantidad de patas, debió ser codificada), para los coprocesadores, que son dispositivos inteligentes periféricos que no pueden tener existencia autónoma, sino que deben tener un procesador principal que produzca señales fundamentales como las de reloj y las de lectura / escritura.

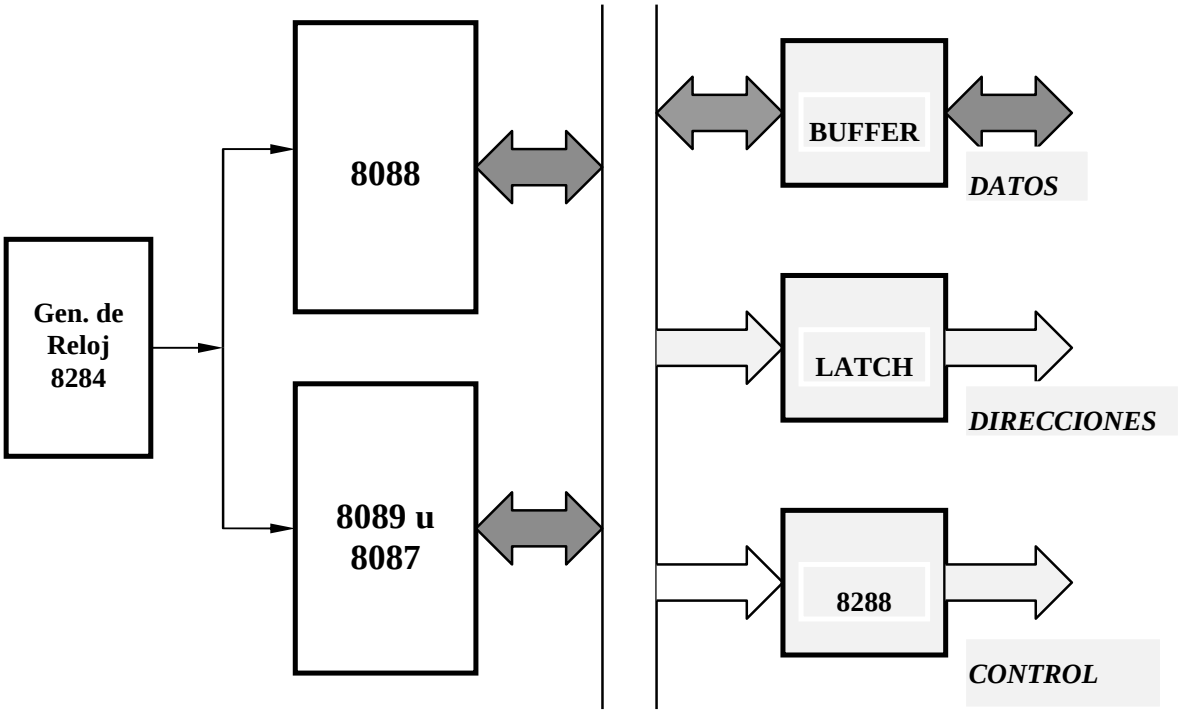


Fig. 6. Diagrama en bloques de una microcomputadora con coprocesador.

En la Fig. 6 se ve una configuración en la que un 8088 operando en modo máximo comparte un bus local con un coprocesador. Se pone de manifiesto la necesidad de disponer de un controlador de bus para generar las señales de control que requerirán los demás dispositivos. Esta configuración es la base de una PC, con la disponibilidad de un coprocesador aritmético 8087.

2.7 Multiprocesamiento

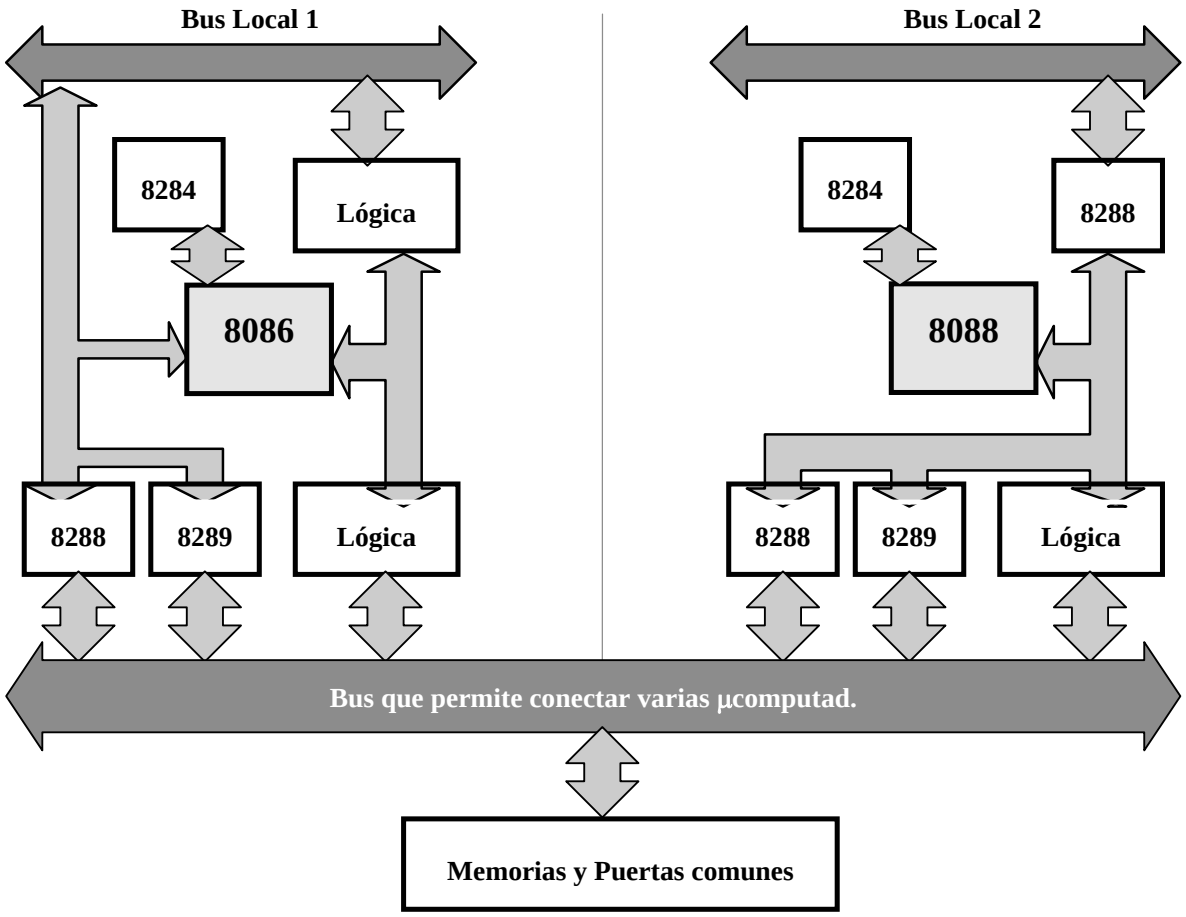


Fig. 7. Diagrama en bloques de una computadora multiprocesador.

Otro de los componentes adicionales que se produjeron conjuntamente con el 8088, fue el llamado ***Arbitrador del bus*** 8289. Dicho dispositivo fue planeado para el caso en que hubiera varias microcomputadoras con 8088/8086 en un único sistema y que compartieran recursos.

En el esquema de la Fig. 7 se observa un esquema de una computadora compuesta por varias microcomputadoras, cada una de ellas con sus propios buses, su propia memoria y dispositivos de entrada / salida. Esta configuración puede emplearse cuando para aumentar la velocidad, se coloquen varias computadoras, cada una de ellas encargada de una actividad específica. Por ejemplo la microcomputadora ***2*** es la encargada de controlar, recibir y transmitir información a un conjunto de líneas serie, mientras que la microcomputadora ***1*** será la encargada de procesar o generar la información que luego comunicará la otra microcomputadora.

La finalidad de esta configuración es la de descargar a la microcomputadora ***1*** de la tarea de gerenciar la entrada / salida de información y dedicarse exclusivamente de procesarla.

Queda como problema sin resolver aún la forma en la que se transfiere la información entre ambas microcomputadoras (el método es extensible a más de dos), ya que es imposible que se sincronicen para transferirse información directamente a través de los buses. Por tal motivo en la parte inferior de la Fig. 7, se observa un bloque de ***"Memoria y puertas comunes"*** que son accesibles para ambas microcomputadoras y se empleará para transferir información entre una y otra.

En esa zona común, por ejemplo, la microcomputadora ***2***, de entrada / salida colocará los datos que reciba por la línea serie en una zona de memoria y cambiará el contenido de otra posición de memoria que será leída periódicamente por la microcomputadora ***1,*** indicando que hay información disponible (dependiendo del protocolo en que deberán haber sido diseñadas ambas computadoras) y quizás la cantidad de bytes recibidos. Cuando la microcomputadora ***1*** tome los datos, limpiará el contenido de esa posición de memoria de control. Esas posiciones de memoria que se utilizan para el control de la información (no las que contiene la información), suelen ser llamadas ***semáforos*** y son escritas por una de las microcomputadoras y limpiadas por la otra cuando tomó la información.

El procedimiento es totalmente análogo para el caso de que la microcomputadora ***1*** haya generado información y desea que la ***2*** la transmita en forma serie.

El acceso de cada una de las microcomputadoras intervinientes de la operación a los recursos comunes, deberá hacerse en forma prolija y ordenada a fin de evitar contención del bus y conflictos entre los buses de las distintas microcomputadoras.

Todo ello se logra por medio de un protocolo de hardware que de acuerdo con cada fabricante se puede llamar Multibus II (Intel) o VMA (Motorola). Cada uno de los microprocesadores intervinientes en este sistema multiprocesador deberá disponer de un arbitrador del bus que, analizando las señales de control del bus común que los vincula con la memoria compartida, determine en que momento puede, cada uno, acceder a los recursos comunes.

3. Diagramas temporales

Las señales descriptas precedentemente, aparecerán en una determinada secuencia temporal, con tiempos indicados por el fabricante.

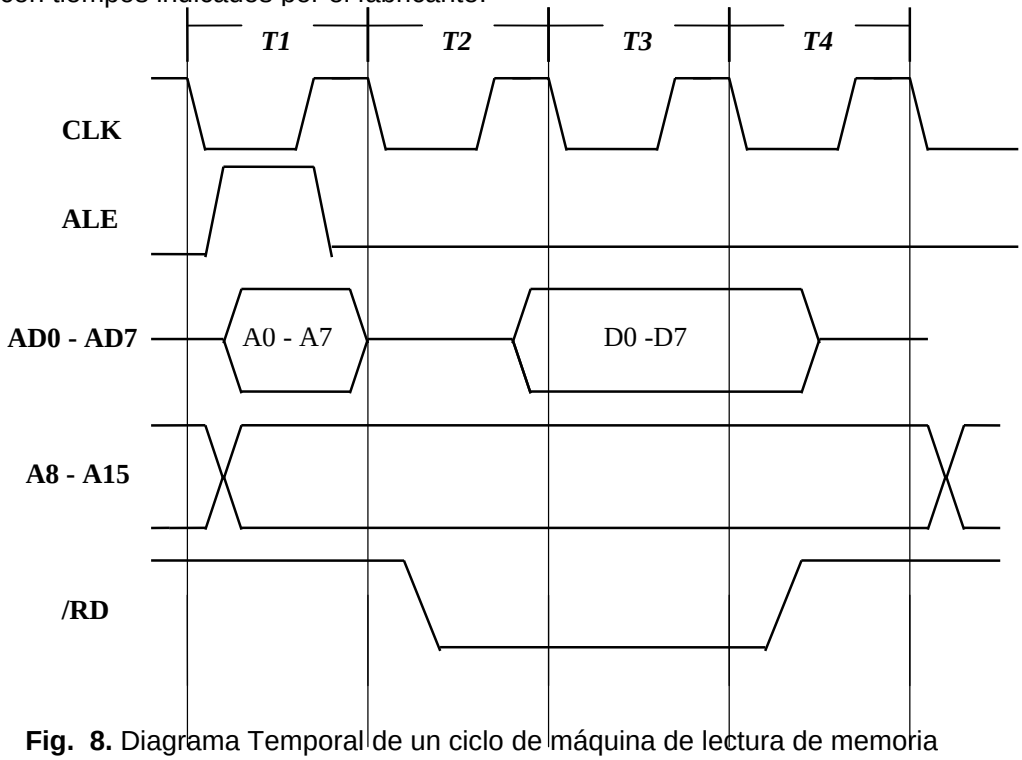


Fig. 8. Diagrama Temporal de un ciclo de máquina de lectura de memoria

Como en todo sistema secuencial sincrónico, existirá un reloj que es quien comanda todas las operaciones. En nuestro caso, el reloj (CLK), será una señal con una frecuencia un tercio de la que corresponde al cristal conectado sobre el generador de reloj 8284 y con un ciclo de actividad[6] de 1/3.

Según lo visto en la definición de ciclo de máquina en el capítulo anterior, en el 8088 un ciclo de máquina de lectura de memoria estará típicamente compuesto por 4 ciclos de reloj (o estados).

1. En el primero de esos estados, el microprocesador coloca sobre el bus de direcciones los 20 bits (ver Generación de la dirección física de 20 bits. en la página 21). En el bus multiplexado de direcciones y datos aparecerán A0 - A7 y ALE estará activa.
2. En el segundo estado, la señal /RD se activa. La memoria reconoce que se trata de un ciclo de lectura de memoria y se prepara para colocar el contenido del byte direccionado sobre el bus de datos.
3. En el tercer ciclo de reloj, el microprocesador configura su bus de datos como entrada. No se realiza ninguna otra operación. Este estado se provee para dar tiempo a la memoria para buscar el dato dentro de su estructura.
4. Finalmente, el microprocesador supone que la memoria ha colocado el contenido esperado sobre el bus de datos y retiene (latchea) la información que ella ha colocado sobre el bus. Libera la señal de control /RD y así se marca el fin del ciclo de máquina.

Para el caso de tratarse de un ciclo de máquina de escritura, el diagrama temporal aparece en la Fig. 9 y la descripción de las actividades realizadas en el ciclo de máquina se describen a continuación:

---

[6] Ciclo de actividad es la relación entre el tiempo en que la señal se halla en el estado lógico "1" respecto del período de la misma.

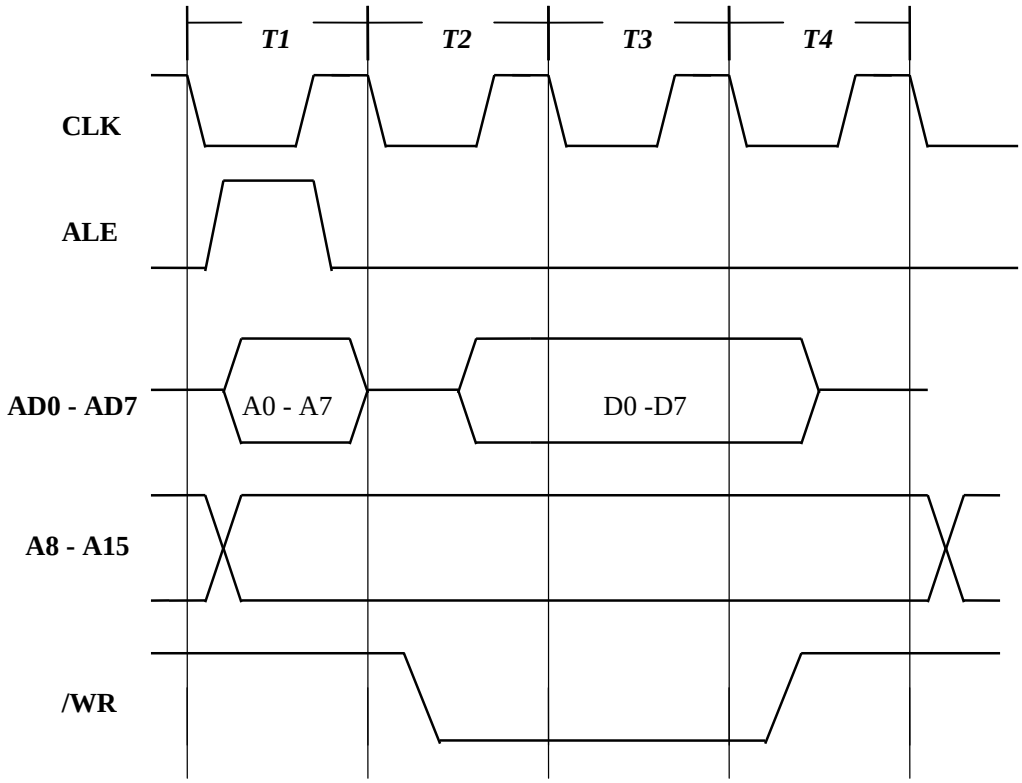


Fig. 9. Diagrama Temporal de un ciclo de máquina de escritura de memoria

1. En el primero de esos estados, el microprocesador coloca sobre el bus de direcciones los 20 bits. En el bus multiplexado de direcciones y datos aparecerán A0 - A7 y ALE estará activa.
2. En el segundo estado, la señal /WR se activa. El microprocesador coloca sobre el bus multiplexado el byte que se desea escribir y la memoria reconoce que se trata de un ciclo de escritura y se prepara para guardar en su interior el contenido del bus de datos.
3. En el tercer ciclo de reloj, la memoria configura su bus de datos como entrada. Esto implica realizar una decodificación interna y habilitar los buffers de entrada / salida hacia la memoria.. No se realiza ninguna otra operación. Este estado se provee para dar tiempo a la memoria para guardar el dato dentro de su estructura.
4. Finalmente, el microprocesador supone que la memoria ha tomado el contenido del bus de datos. Libera la señal de control /RD y coloca en alta impedancia el bus de datos y así se marca el fin del ciclo de máquina.

4. Arquitectura interna del 8088

Con esta familia de microprocesadores se comenzó a trabajar en el concepto de procesamiento interno paralelo. Esto quiere decir que se dividió al microprocesador en subunidades de actuación específica y que trabajarán simultáneamente (paralelismo) buscando mejorar la velocidad de todo el conjunto.

El concepto del paralelismo es similar a la operatoria que realizan un gerente con su secretaria. El gerente necesita una comunicación telefónica (vinculación con el mundo exterior) y se la solicita a la secretaria. Mientras esta realiza el contacto, el gerente continúa con su actividad normal. Cuando la secretaria consiguió la comunicación le avisa al gerente y este toma contacto con su interlocutor. Ambos han trabajado en paralelo y sincronizadamente para realizar una transacción.

Cada procesador 8086 y 8088 se halla dividido en dos sub-procesadores llamados unidad de ejecución **(EU: Execution Unit)** y unidad interfaz del bus **(BIU: Bus Interface Unit)**.

La ***unidad de ejecución*** es la encargada de realizar todas las operaciones propias de ejecutar cada instrucción, mientras que **la *unidad de interfaz con el bus*** es la encargada de las transacciones de datos e instrucciones con el mundo exterior. Las unidades de ejecución son idénticas en el 8088 y en el 8086, pero las unidades de interfaz del bus son diferentes en varios aspectos, como se desprende del diagrama en bloques de la Fig. 10.

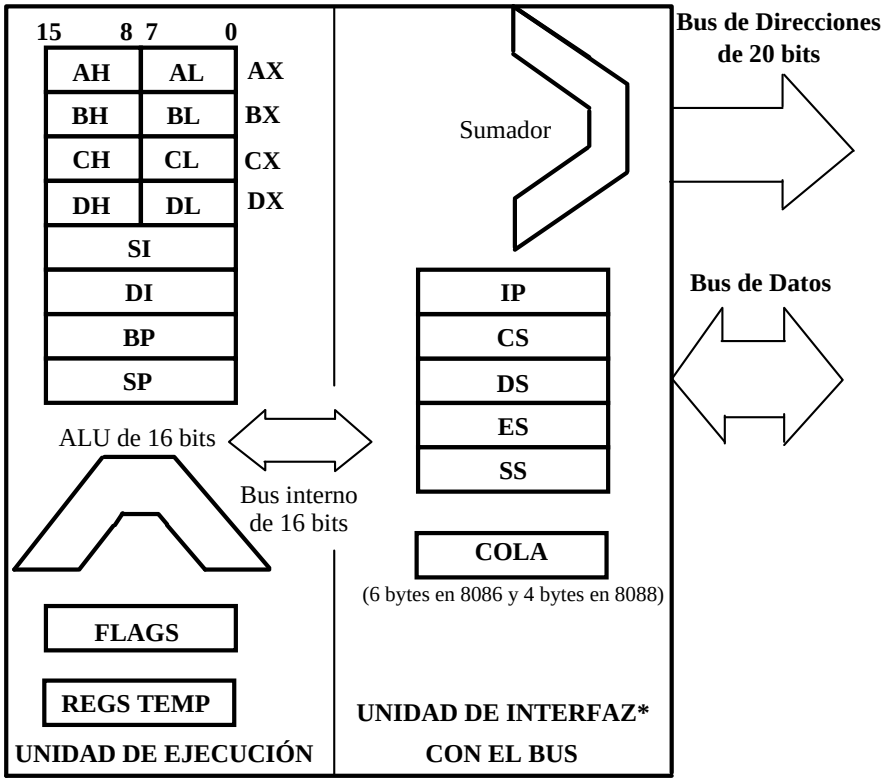


Fig. 10. Arquitectura interna del 8086/8088

La ventaja de esta división fue el ahorro de esfuerzo necesario para producir el 8088. Sólo una mitad del 8086 (el BIU) tuvo que rediseñarse ligeramente para producir el 8088.

A continuación analizaremos cada una de los bloques constitutivos del 8088.

4.1 Registros de uso general del 8086/8088.

Tienen 16 bits cada uno y son ocho:

AX	Registro acumulador, puede dividirse en AH y AL (8 bits cada uno).
BX	Registro base, puede dividirse en BH y BL.
CX	Registro contador, puede dividirse en CH y CL.
DX	Registro de datos, puede dividirse en DH y DL.
SP	Puntero de pila (no se puede subdividir).
BP	Puntero base (no se puede subdividir).
SI	Puntero índice fuente (no se puede subdividir).
DI	Puntero índice destino (no se puede subdividir).

Tabla 5. Registros internos del 8086/8088.

Si bien cualquiera de estos registros se puede emplear como fuente o destino en operaciones aritmético-lógicas, los microprocesadores de 8086 y por ende 8088, no pueden ser catalogados como ortogonales[7]. ya que cada registro tiene usos específicos:

AX: Todas las operaciones de entrada salida deberán emplear obligatoriamente AX (para 16 bits) o AL (para 8 bits)[8]. El registro AL es el equivalente al acumulador de los procesadores 8080 y 8085. Además hay instrucciones como DAA; DAS; AAA; AAS; AAM; AAD; LAHF; SAHF; CBW; IN y OUT que trabajan con AX o con la mitad inferior (AL). También se utiliza este registro (junto con DX a veces) en multiplicaciones y divisiones.

BX: Es el registro base de propósito similar (se usa para direccionamiento indirecto mediante registro) y es una versión más potente del par de registros HL de los procesadores anteriores.

CX: Se utiliza como contador en lazos (instrucción LOOP), en operaciones con cadenas (usando el prefijo REP) y en desplazamientos y rotaciones (usando el registro CL en los dos últimos casos).

DX: Se utiliza junto con el registro AX en multiplicaciones y divisiones, en la instrucción CWD y en IN y OUT para direccionamiento indirecto de puertos (el registro DX indica el número de puerto de entrada/salida).

SP: Aunque es un registro de uso general, debe utilizarse sólo como puntero de pila, la cual sirve para almacenar las direcciones de retorno de subrutinas y los datos temporarios (mediante las instrucciones PUSH y POP). Al introducir (push) un valor en la pila a este registro se le resta dos, mientras que al extraer (pop) un valor de la pila este a registro se le suma dos. Este registro debe inicializarse al comienzo del programa principal, como se describirá posteriormente.

---

7 Se dice que un microprocesador tiene una arquitectura simétrica u ortogonal cuando todos los registros pueden realizar todas las operaciones, es decir que no existen registros "especializados".

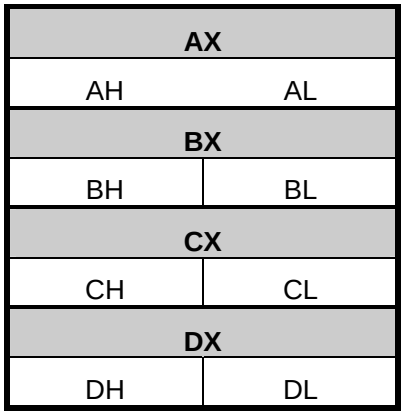
8 Las instrucciones que empleen AX o AL, en general producirán instrucciones cuya extensión ocupará un byte menos que si se emplearan otros registros de propósito general.

BP: Generalmente se utiliza para realizar direccionamiento indirecto dentro de la pila. Se lo suele llamar puntero a la pila del usuario, pues permite operar con la pila sin afectar el valor de SP. Permite retirar las variables locales pasadas a una función.

SI: Sirve para realizar direccionamiento indirecto mediante registros. También sirve como puntero fuente para las operaciones con cadenas.

DI: Sirve para realizar direccionamiento indirecto mediante registros. También sirve como puntero destino para las operaciones con cadenas.

4.2 Resumen de los usos y aplicaciones de los registros



Registro	Operación
AX	Multiplicación y división de palabras. E/S de 16 bits
AL	Multiplicación y división de bytes. E/S de 8 bits. XLAT. Aritmética decimal.
AH	Multiplicación y división de bytes
BX	Puntero. Base de XLAT.
CX	Operaciones de cadenas (strings). Lazos.
CL	Desplazamiento de variables. Rotaciones
DX	Multiplicación y división de palabras. Direccionamiento indirecto de E/S.

Fig. 11. Registros de propósito general y de uso específico.

4.3 Unidad aritmética y lógica:

Es la encargada de realizar las operaciones aritméticas (suma, suma con "arrastre", resta, resta con "préstamo" y comparaciones) y lógicas (AND, OR, XOR y TEST). Las operaciones pueden ser de 16 bits o de 8 bits.

4.4 Indicadores (flags):

Hay nueve indicadores de un bit en este registro de 16 bits. Los cuatro bits más significativos están indefinidos, mientras que hay tres bits con valores determinados: los bits 5 y 3 siempre valen cero y el bit 1 siempre vale uno (esto también ocurría en los procesadores anteriores).

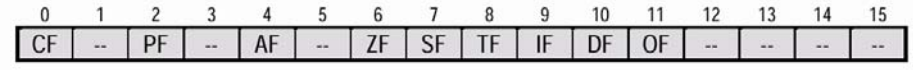


Fig. 12. Palabra de flags.

CF (Carry Flag, bit 0): Si vale 1, indica que hubo "arrastre" (en caso de suma) hacia, o "préstamo" (en caso de resta) desde el bit de orden más significativo del resultado. Este indicador es usado por instrucciones que suman o restan números que ocupan varios bytes. Las instrucciones de rotación pueden aislar un bit de la memoria o de un registro poniéndolo en el CF.

PF (Parity Flag, bit 2): Si vale uno, el resultado tiene paridad par, es decir, un número par de bits a 1. Este indicador se puede utilizar para detectar errores en transmisiones.

AF (Auxiliary carry Flag, bit 4): Si vale 1, indica que hubo "arrastre" o "préstamo" del nibble (cuatro bits) menos significativo al nibble más significativo. Este indicador se usa con las instrucciones de ajuste decimal.

ZF (Zero Flag, bit 6): Si este indicador vale 1, el resultado de la operación es cero.

SF (Sign Flag, bit 7): Refleja el bit más significativo del resultado. Como los números negativos se representan en la notación de complemento a dos, este bit representa el signo: 0 si es positivo, 1 si es negativo.

TF (Trap Flag, bit 8): Si vale 1, el procesador está en modo paso a paso. En este modo, la CPU automáticamente genera una interrupción interna después de cada instrucción, permitiendo inspeccionar los resultados del programa a medida que se ejecuta instrucción por instrucción.

IF (Interrupt Flag, bit 9): Si vale 1, la CPU reconoce pedidos de interrupción externas enmascarables (por el pin INTR). Si vale 0, no se reconocen tales interrupciones. Las interrupciones no enmascarables y las internas siempre se reconocen independientemente del valor de IF.

DF (Direction Flag, bit 10): Si vale 1, las instrucciones con cadenas sufrirán "auto-decremento", esto es, se procesarán las cadenas desde las direcciones más altas de memoria hacia las más bajas. Si vale 0, habrá "auto-incremento", lo que quiere decir que las cadenas se procesarán de "izquierda a derecha".

OF (Overflow flag, bit 11): Si vale 1, hubo un desborde en una operación aritmética con signo, esto es, un dígito significativo se perdió debido a que tamaño del resultado es mayor que el tamaño del destino.

4.5 Sistema de control de la unidad de ejecución:

Es el encargado de decodificar las instrucciones que le envía la cola y enviarle las órdenes a la unidad aritmética y lógica según una tabla que tiene almacenada en ROM llamada CROM (Control Read Only Memory).

4.6 Cola de instrucciones:

Es una memoria interna de 4 (para el 8088) ó 6 (para el 8086) bytes de extensión. Almacena las instrucciones para ser ejecutadas. La cola se carga cuando el bus está desocupado, de esta manera se logra una mayor eficiencia del mismo.

En efecto, aprovechando los lapsos en los cuales el microprocesador se halla ocupado en otras actividades y no emplea los buses, el microprocesador copia parte de la memoria de programa a una memoria interna llamada cola. De esta forma, cuando necesite un operando o un código de operación, no necesitará buscarlo en el exterior, sino que ganará tiempo (al no necesitar activar los buffers de salida y esperar los ciclos de máquina de lectura) disponiéndolo inmediatamente en la cola.

La cola del 8086 tiene 6 bytes y se carga de a dos bytes por vez (debido al tamaño del bus de datos), mientras que el del 8088 tiene cuatro bytes. Esta estructura tiene rendimiento óptimo cuando no se realizan saltos, ya que en este caso habría que vaciar la cola (porque no se van a ejecutar las instrucciones que van después del salto) y volverla a cargar con instrucciones que se encuentran a partir de la dirección a donde se salta. Debido a esto las instrucciones de salto son (después de multiplicaciones y divisiones) las más lentas de este microprocesador.

Repetimos a continuación la**Tabla 4.** Significado del estado de la cola, indicado por QS0 y QS1, en la que se observa el significado de la información codificada provista por las señales de modo máximo QS0 y QS1.

QS0	QS1	Descripción
0	0	Ninguna Operación
0	1	Se esta ejecutando el primer byte de una instrucción.
1	0	La cola se está vaciando
1	1	Se esta tomando un byte subsiguiente de la instrucción.

Tabla 6. Significado del estado de la cola, indicado por QS0 y QS1

El microprocesador copia en la cola, zonas de la memoria de programa sin ninguna inteligencia ni decisión. Los procesadores posteriores, a partir del 80386 copian en una cola de mucho mayor tamaño (hasta 1 Mbyte) denominada memoria cache, tramos de la memoria de programa pero con un cierto criterio.

A partir de los procesadores Pentium, al encontrarse con una bifurcación de programa, el microprocesador (o mejor dicho, el controlador de memoria cache) copia ambos tramos del programa en la memoria cache y hasta hace un análisis probabilístico de cual camino ejecutará, para darle prioridad en la carga.

Toda esta acción se realiza en forma transparente para el usuario final. Será el sistema operativo quien se encargará de programar en el controlador específico la política de manejo de la memoria cache.

4.7 Registros de Segmento.

El 8088 dispone de cuatro registros que se emplean para generar los 20 bits de direccionamiento. Cada uno de ellos permite apuntar a un espacio distinto de memoria y tienen aplicaciones específicas.

CS: (Code Segment) Registro de segmento de código, Normalmente se lo emplea para apuntar a la memoria de programa y permite direccionar códigos de operación y operandos.

DS: (Data Segment) Registro de segmento de datos. Normalmente se emplea para apuntar a la zona de memoria de lectura y escritura en la que se hallan datos y variables.

ES: (Extra Segment) Registro de segmento extra. Normalmente se emplea como alternativa del Data Segment.

SS: (Stack Segment) Registro de segmento de pila. Se utiliza para apuntar a la pila.

Estos registros de segmento desde el 80386, se llamarán selectores y apuntarán a pequeñas tablas (descriptores) que se encargarán de caracterizar (tamaño, protecciones, acceso) cada segmento de memoria.

4.8 Generación de la dirección física de 20 bits.

Cada uno de estos registros de segmento se apareará con otro registro (o dirección) llamado desplazamiento (offset) para generar la dirección física de 20 bits, según se presenta en la figura Fig. 13. En dicha figura se observa que se adicionará bit a bit el registro de segmento con el desplazamiento, éste desplazado en cuatro lugares (bits) hacia la derecha.

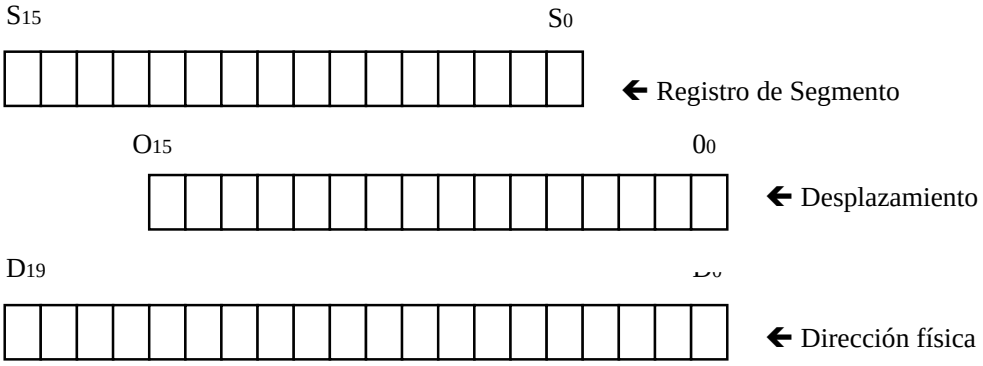


Fig. 13. Generación de la dirección física empleando un registro de segmento y el desplazamiento

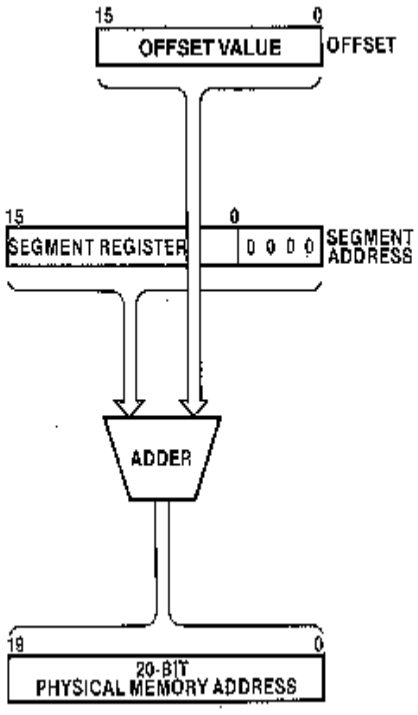


Fig. 14. Sumador para la generación de la dirección física de 20 bits.

Supongamos que el acceso a una variable en la zona de datos, se realiza empleando DS como registro de segmento y el registro índice puntero a memoria SI del procesador. Admitamos que DS vale FFFF en hexadecimal SI vale 000FH. La dirección física en la que hallará esa variable se calcula como sigue:

DS	➔	FFFF
	+	
SI	➔	000F
Dirección Física	➔	FFFFF

Es decir que se apuntará a la memoria cuya dirección es FFFFFH.

Debe destacarse que no existe acarreo, de forma que, supongamos que en ejemplo anterior SI vale 10H:

DS	➔	FFFF
	+	
SI	➔	0010
Dirección Física	➔	00000

Es decir que la posición de memoria posterior a la FFFFFH, será la 00000H.

4.9 Segmentación

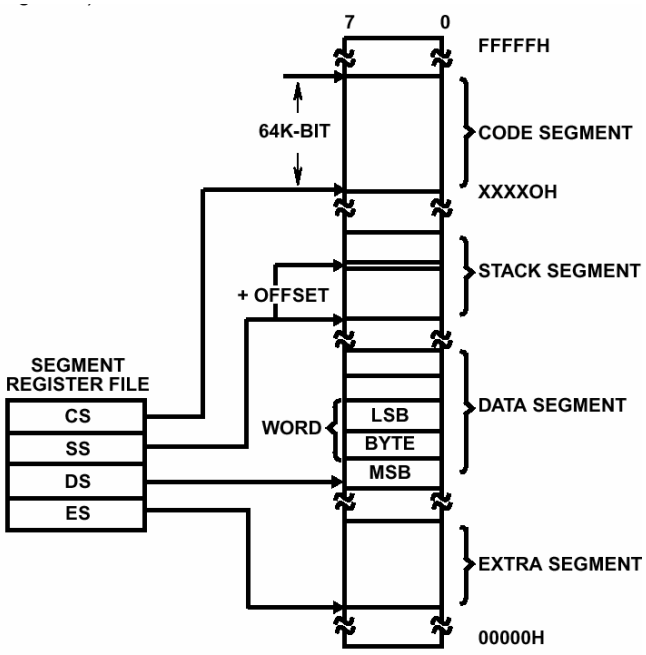


Fig. 15. Uso de los segmentos para apuntar al mapa de memoria.

Manteniendo un registro de segmento fijo, al variar el desplazamiento podemos recorrer 64K (desde 0000 a FFFFH). Para poder emplear más de 64 Kbytes de memoria, podemos hacer que cada registro de segmento apunte a una zona distinta del mapa, como vemos en la Fig. 15. Allí tendremos una zona de código o programa apuntada por CS, otra de Pila o Stack apuntada por SS y otra de datos apuntada por DS. Habitualmente ES se emplea para apuntar también a datos.

5. Reset y mapeo de memoria.

Luego del Reset (ya sea en el encendido o posteriormente), todos los registros del 8088 valen 0000H, salvo el CS que vale FFFFH.

Al producirse un reset, el 8088 pasa a ejecutar el programa que se halla en la dirección CS:IP, o sea FFFF:0000 o bien según lo visto anteriormente, FFFF0H.

Esto quiere decir que en esa dirección se deberá encontrar la memoria de programa para que inmediatamente después del reset se comience a ejecutar el programa de inicialización del sistema.

En la Fig. 16, vemos una parte del circuito de una microcomputadora con el 8088. Allí podemos ver la decodificación necesaria para que el procesador acceda a la memoria de programa inmediatamente después del reset.

Es habitual (aunque no es obligatorio que lo sea) que en la posición de memoria 0000:0000 se halle memoria de datos[9].

[9] Allí se ubicarán los vectores de interrupción.

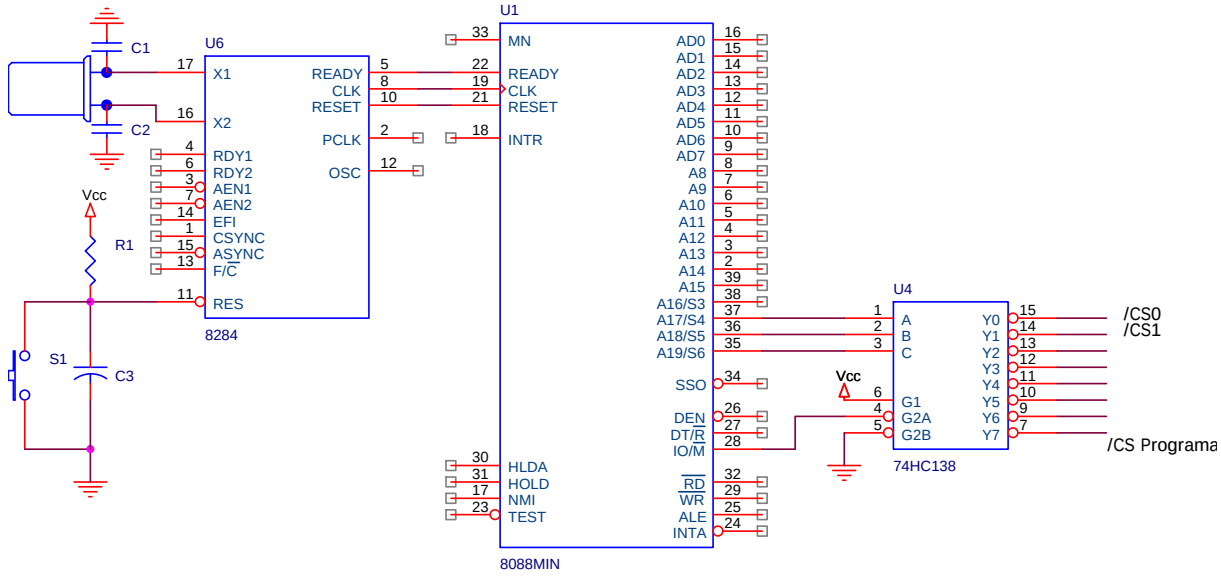


Fig. 16. 8088 con el circuito de reset en el encendido y la decodificación de la memoria de Programa.

6. Manejo de Entradas y Salidas.

Los microprocesadores de la familia Intel se caracterizan por tener espacios separados para memoria y para entradas y salidas[10]. Los dispositivos de E/S tienen 64 K de espacio para su selección.

La línea IO/*M actúa como una línea adicional de direccionamiento y permite separar el mapa de memoria del de las interfaces de entrada / salida.

Para el 8088 existen dos instrucciones que hace que IO/*M valgan 1 (es decir actuando sobre el mapa de E/S). Son IN y OUT que se corresponden biunívocamente con inport y outport del C.

[10] Otras familias de microprocesadores no disponen de estas líneas y deben mapear sus interfaces de entrada / salida como memoria, lo cual, si bien no es nada grave, implica desperdiciar posiciones de memoria en ellos. En el caso de Intel las interfaces de entrada / salida se pueden ubicar tanto en el mapa de IO como en el de memoria.

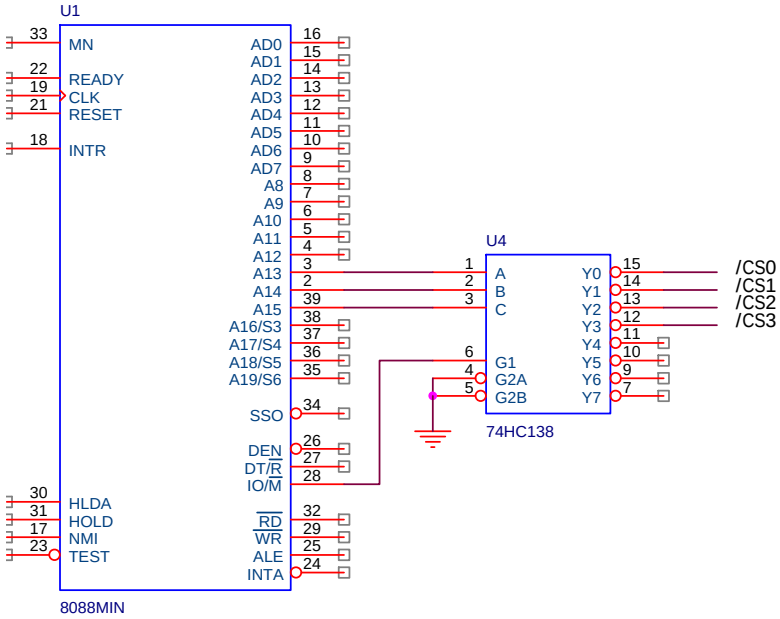


Fig. 17. Generación de habilitaciones para dispositivos de E/S

En la Fig. 17, vemos la utilización de ***otro*** decodificador adicional al empleado para seleccionar las memorias, y que habilitará los distintos dispositivos de E/S.

Vemos que las líneas de selección no son las más altas, sino que son A_{13} – A_{15}, ya que la capacidad de direccionamiento para E/S es de 64 K. Las líneas A_{16} – A_{19} durante las operaciones de E/S se hallan en “0”.

Debe destacarse que en el ciclo de instrucción de entrada o salida (IN – OUT), los primeros ciclos de máquina (búsqueda de código de operación y operandos) se realiza como habitualmente se hizo sobre memoria y solamente el último ciclo de máquina lleva IO/*M a “1” para acceder a los dispositivos de E/S.

7. Modos de direccionamiento del 8086/8088:

7.1 Planteo.

Recordemos que los modos de direccionamiemto permiten que el microprocesador obtenga alguno de los operandos necesarios para ejecutar una instrucción.

Tanto el 8088 como el 8086 tienen 27 modos de direccionamiento. Los tres básicos, son comunes a todos los microprocesadores fueron analizados en el capítulo 1: a) Direccionamiento directo (se provee la dirección del segundo operando), b) direccionamiento inmediato (se entrega el operando como parte de la misma instrucción) y c) direccionamiento inherente (el operando está implícito en la instrucción), por ejemplo, en la multiplicación uno de los operandos siempre es el acumulador.

Para esta familia de microprocesadores se agregan múltiples modos de direccionamiento entre los que pueden destacarse como los más importantes: d) Direccionamiento a registro (el operando es un registro del microprocesador) y e) Direccionamiento indirecto por registro (el operando es apuntado por un registro del microprocesador). Para este último modo de direccionamiento, debe destacarse que no todos los registros pueden emplearse como punteros a memoria.

El resto de los modos sirve para localizar un operando en memoria

El caso más genérico, consiste en acceder a ese operando obteniendo la dirección efectiva del mismo por la suma de cuatro cantidades:

a) Dirección de segmento (multiplicada por 16 o desplazada en cuatro bits a la izquierda). La dirección de segmento se almacena en el registro de segmentación (DS, ES, SS o CS). El registro de segmentación siempre se usa para referenciar a memoria. Se empleará, por ejemplo, para apuntar a una zona definida de memoria en la que se hallarán los datos del personal de un supermercado.

b) Dirección base. Se almacena en el registro base (BX o BP). Por ejemplo para apuntar al inicio de una tabla, como podría ser la de los datos del personal de la Sección "Cajeros".

c) Una cantidad índice. Se emplearán los registros índice SI o DI. Por ejemplo para apuntar a los datos personales del cajero Nº15.

d) Un desplazamiento de 16 bits, 8 bits o 0 bits (sin desplazamiento). Esta es una cifra constante. Se fija al tiempo de ensamblado (paso de código fuente a código de máquina) y no puede cambiarse durante la ejecución del programa (a menos que el programa se escriba sobre sí mismo, lo que constituye una práctica no aconsejada). El desplazamiento se empleará, por ejemplo, para apuntar a la edad del cajero Nº 15 del supermercado.

Seguramente nos encontraremos con direccionamientos a memoria en los que falten algunos de estos cuatro parámetros.

El índice o la base, la suma de las dos o ninguna, pueden utilizarse para calcular la dirección efectiva, pero no pueden sumarse dos bases o dos índices.

En base a lo anterior, presentaremos los siguientes modos de direccionamiento:

a) Registro indirecto:

1.- [BX]

2.- [DI].

3.- [SI]. Esto debe leerse como que BX, DI o SI apuntan a memoria.

b) Basado:

4.- desp8[BX] ; BX apunta a memoria con un desplazamiento de 8 bits

5.- desp8[BP] ; BP apunta a memoria con un desplazamiento de 8 bits

6.- desp16[BX] ; Idem 16 bits

7.- desp16[BP]

c) Indexado:

8.- desp8[SI]

9.- desp8[DI]

10.- desp16[SI]

11.- desp16[DI].

d) Basado-indexado:

12.- [BX+SI] ; Dirección apuntada por la suma de BX + SI (sin afectar a ninguno de los dos).

13.- [BX+DI]

14.- [BP+SI]

15.- [BP+DI].

e) Basado-indexado con desplazamiento:

16.- desp8[BX+SI] ; Dirección de memoria apuntada por la suma de BX + SI con un desplazamiento de 8 bits.

17.- desp8[BX+DI]

18.- desp8[BP+SI]

19.- desp8[BP+DI]

20.- desp16[BX+SI] ; Idem con desplazamiento de 16 bits

21.- desp16[BX+DI]

22.- desp16[BP+SI]

23.- desp16[BP+DI].

f) Directo:

24.- [desp16].

Aquí desp8 indica desplazamiento de 8 bits y desp16 indica desplazamiento de 16 bits. Otras combinaciones no están implementadas en la CPU y generarán error al querer ensamblar, por ejemplo, ADD CL,[DX+SI].[11]

El ensamblador genera el tipo de desplazamiento más apropiado (0, 8 ó 16 bits) dependiendo del valor que tenga la constante: si vale cero se utiliza el primer caso, si vale entre -128 y 127 se utiliza el segundo, y en cualquier otro caso se utiliza el tercero. Nótese que no existe el direccionamiento que emplea [BP] sin desplazamiento.

Al ensamblar una instrucción como, por ejemplo, MOV AL,[BP], se generará un desplazamiento de 8 bits con valor cero. Esta instrucción ocupa tres bytes, mientras que MOV AL,[SI] ocupa dos, porque no necesita el desplazamiento.

Estos modos de direccionamiento producen algunos inconvenientes en el 8086/8088. La CPU gasta tiempo calculando una dirección compuesta de varias cantidades. Principalmente esto se debe al hecho de que el cálculo de direcciones está programado en microcódigo (dentro de la CROM del sistema de control de la unidad de ejecución). En las siguientes versiones (a partir del 80186/80188) estos cálculos están cableados en la máquina y, por lo tanto, se emplea mucho menos tiempo en realizarlos.

Veamos un ejemplo: MOV AL, ES:[BX+SI+6]. En este caso el operando de la izquierda tiene direccionamiento a registro mientras que el de la derecha indica una posición de memoria. Poniendo valores numéricos, supongamos que los valores actuales de los registros sean:

ES = 3200h

BX = 200h

SI = 38h.

Como se indicó precedentemente, la dirección física de de memoria será:

ES * 10h + BX + SI + 6 = 3200h * 10h + 200h + 38h + 6 = 3223Eh

Hay dos registros de segmento que tienen usos especiales: el microprocesador utiliza el registro CS (con el offset almacenado en el puntero de instrucción IP) cada vez que se debe acceder a un byte de instrucción de programa, mientras que las instrucciones que utilizan la pila (llamados a subrutinas, retornos, etc. y las instrucciones PUSH y POP) siempre utilizan el registro de segmento SS (con el offset almacenado en el registro puntero de pila SP). De ahí los nombres que toman: CS es el segmento de código mientras que SS es el registro segmento de pila.

Para acceder a datos en la memoria se puede utilizar cualquiera de los cuatro registros de segmento, pero uno de ellos provoca que la instrucción ocupe un byte menos de memoria: es el llamado segmento por defecto, por lo que en lo posible hay que tratar de usar dicho segmento para direccionar datos.

---

[11] Se recomienda al alumno que practique con una PC el programa Debug (incluído dentro del DOS) o bien con el Turbo Debugger, la sintaxis permitida de cada modo de direccionamiento.

Este segmento es el DS (registro de segmento de datos) para todos los casos excepto cuando se utiliza el registro base BP. En este caso el segmento por defecto es SS.

Si se utiliza otro registro de segmento, el ensamblador genera un byte de prefijo[12] correspondiente al segmento forzado y que se inserta antes de la instrucción.

El uso de estos diferentes segmentos significa que hay áreas de trabajo separadas para el programa, pila y los datos. Cada área tiene un tamaño máximo de 64 Kbytes. Dado que hay cuatro registros de segmentación, uno de programa (CS), uno de pila (SS) y dos de datos (segmento de datos DS y segmento extra ES) el área de trabajo puede llegar a 4 * 64 KB = 256 KB en un momento dado suponiendo que las áreas no se superponen.

Si el programa y los datos ocupan menos de 64 KB, lo que se hace es fijar los registros de segmentación al principio del programa y luego se utilizan diferentes offsets para acceder a distintas posiciones de memoria. En caso contrario necesariamente deberán cambiarse los registros de segmento en la parte del programa que lo requiera. Los registros de segmento DS, ES y SS se cargan mediante las instrucciones MOV y POP, mientras que CS se carga mediante transferencias de control (saltos, llamadas, retornos, interrupciones) ínter segmento.

7.2 Resumen de direccionamiento.

MODO	OPERADOR	REGISTRO BASE	EJEMPLO
Registro	Registro	--	`MOV AX, BX`
Valor	Valor inmediato	--	`MOV AX, 2604`
Variable	Offset inmediato	DS	`MOV AX, [1432]`
Indirecto mediante registro	[BX]	DS	`MOV DX,DS:[BX]`
	[BP]	SS	`MOV DX,SS:[BP]`
	[DI]	DS	`MOV DX,DS:[DI]`
	[SI]	DS	`MOV DX,DS:[SI]`
Relativo a base	[BX] + desp	DS	`MOV CX,DS:[BX+40000]`
	[BP] + desp	SS	`MOV CX,SS:[BP+40000]`
Directo indexado	[DI] + desp	DS	`MOV CX,DS:[DI+40000]`
	[SI] + desp	DS	`MOV CX,DS:[SI+40000]`
Indexado a base	[BX] + [SI] + desp	DS	`MOV AX,DS:[BX][SI]+300`
	[BX] + [DI] + desp	DS	`MOV AX,DS:[BX][DI]+300`
	[BP] + [SI] + desp	SS	`MOV AX,SS:[BP][SI]+300`
	[BP] + [DI] + desp	SS	`MOV AX,SS:[BP][DI]+300`

8. Repertorio de instrucciones.

8.1 Planteo introductorio.

Hasta aquí hemos estudiado la arquitectura interna de la familia 8088/86. El alumno deberá disponer de una herramienta de depuración sobre PC como el DEBUG de Microsoft, el Turbo debugger de Borland u otro programa similar, a fin de poder comprobar y ejercitarse en el manejo del repertorio de instrucciones que se planteará a continuación.

La descripción de la totalidad de las instrucciones del 8088 puede ocupar centenas de páginas[13] y ser sumamente tedioso. Nuestro planteo será realizar una clasificación general del repertorio de instrucciones y dejar al alumno la tarea de ejercitarlas con el depurador e intentar su uso en diversos modos de direccionamiento.

En esta circunstancia, el mecanismo de prueba y error permite un aprendizaje mucho más efectivo que una descripción en papel.

---

[12] Prefijos son códigos de operación que sólo afectan sólo a la instrucción siguiente. Por ejemplo:

MOV CS: [BX],AX copiará el contenido del registro AX en la dirección apuntada por BX, pero empleando como registro de segmento a CS en lugar de DS que es el registro habitual (por omisión) para apuntar a memoria de datos. La siguiente instrucción volverá a emplear el registro de segmento habitual.

[13] Ver The 8086 book de Russell Rector y George Alexy. Ed. Osborne Mc. Graw Hill. 1980.

8.2 Instrucciones de movimiento de datos.

El 8088 dispone de un conjunto de instrucciones que permiten el movimiento de datos entre registros y entre registros y memoria externa. Este grupo incluye:

8.2.1 Las instrucciones de movimiento MOV de byte (8 bits) o de Word (palabra, 16 bits).

En la Fig. 18 se presenta un breve resumen de las características de la instrucción de movimiento de datos. Allí vemos el nemónico de la instrucción -MOV-, el significado -movimiento de datos-, el formato -MOV D(estino),F(uente)-, lo que implica que el movimiento de datos del registro AX al registro CX se abreviará como MOV CX,AX y la expresión simbólica de la operación -(F) $\rightarrow$ (D)-. Debe destacarse como una característica distintiva muy importante para los microprocesadores de Intel que los movimiento de datos **NO** afectan a los flags o indicadores de estado.

Nemónico	Significado	Formato	Operación	Flags Afect.
MOV	Movimiento	MOV D,F	(F) $\rightarrow$ (D)	Ninguno

Fig. 18. Resumen de las instrucciones MOV

En la Fig. 19 se pone de manifiesto la fuente y el destino de los datos objeto del movimiento. Se observa la no-existencia de movimiento de datos desde memoria hacia memoria en forma directa. Ello requeriría que el microprocesador actuase como una máquina de dos direcciones, mientras que esta diseñado para operar como máquina de una dirección.

En este caso, se deberá utilizar un registro del microprocesador como intermediario del movimiento.

Destino	Fuente
Memoria	Acumulador
Acumulador	Memoria
Registro	Registro
Registro	Memoria
Memoria	Registro
Registro	Inmediato
Memoria	Inmediato
Registro Segm.	Reg. 16 bits
Registro Segm.	Mem. 16 bits
Reg 16 bits	Registro Segm.
Mem 16 bits	Registro Segm.

Fig. 19. Operandos permitidos para la instrucción MOV.

El movimiento de datos entre registros y memoria puede ser a nivel de bytes o palabras de 16 bits. El programa ensamblador colocará uno u otro código de operación dependiendo de que tipo de registro (AX o AL, por ejemplo) se halla especificado en la instrucción.

Un caso especial es el movimiento inmediato. Supongamos tener la instrucción[14]:

```
MOV    [1234H], 56H
```

Se pretende hacer un movimiento de datos que lleve en direccionamiento inmediato el valor 56H a una posición de memoria. La duda aparece respecto de si la posición de memoria es un byte o una palabra, ya que ello no se deduce del nemónico empleado. Normalmente (depende del ensamblador empleado) aparecerá un mensaje de error del tipo "Definición ambigua", no terminándose la compilación en forma satisfactoria.

Esta situación ambigua se resuelve colocando una indicación de que tipo de variable se desea direccionar, byte o Word, como se observa a continuación:

```
MOV    byte ptr [1234H],56H    ; guardará 56H en la posición de memoria DS:1234H

MOV    Word ptr [1234H],56H    ; guardará 56H en la posición de memoria DS:1234H y 00H
                               ; la siguiente, ya que se interpretó que se quiere almacenar
                               ; en una palabra 0056H en memoria. Recuérdese que en
                               ; formato Intel primero se almacena la parte baja y luego la
                               ; parte alta de la variable
```

Si bien no es obligatorio, es recomendable emplear estos prefijos *byte ptr* y *Word ptr* cada vez que se acceda a una variable en memoria, pues facilitan la comprensión del programador respecto de que tipo de variable se esta empleando.

Alguno ensambladores generan un código de advertencia (warning) cuando se pretende acceder a una variable definida como byte con el prefijo *word ptr* y viceversa. Ello es para advertirle al programador la violación de una definición previa. Puede ser que el programador experimentado este utilizando adrede esta violación para los fines de su programa.

Como se deduce de la Fig. 19, no esta permitida la carga inmediata de los registros de segmento. Supongamos que por la ubicación de la memoria de datos de un determinado circuito se necesita que DS apunte al valor 2000H. Para inicializarlo al comienzo de programa, deberá apelarse a un recurso como el siguiente:

```
MOV    AX,2000H        ; Se carga el registro intermedio AX con el valor que se pretende dar a DS
MOV    DS,AX           ; Se carga DS desde AX con el valor 2000H
```

8.2.2 Las instrucciones de Entrada / Salida

Como dijimos anteriormente, el 8088 permite operar sobre un mapa de E/S independiente del de memoria. Ello quiere decir que habrá instrucciones particulares que permiten traer información del mundo exterior o llevarla hasta el mismo.

Esas instrucciones son IN y OUT y son las únicas que llevan la línea IO/*M a 1 en el último ciclo de máquina.

De acuerdo con lo dicho precedentemente al describir las particularidades de los registros, todas las operaciones de E/S requieren al Acumulador con operando. AL para el caso de operaciones de 8 bits o AX para las de 16 bits. En este último caso se realizarán dos operaciones sobre direcciones de E/S consecutivas en forma automática. Es decir que el programador deberá dar la dirección de la posición más baja y automáticamente el microprocesador incrementará ese valor (temporariamente y en forma transparente para el usuario) para acceder a la dirección siguiente de donde obtendrá el byte más significativo de la magnitud de 16 bits.

Para las operaciones de E/S dispondremos de dos modos de direccionamiento:

- Directo. Válido entre las posiciones de E/S 0 y 0FFH. A continuación del Registro (AL o AX) se indicará la puerta de E/S.

  ```
  IN     AL,60H    ; Se lleva al registro AL el contenido de la puerta 60H
  ```

[14] Recordemos que los corchetes indican "la memoria apuntada por" y la falta de corchetes indica direccionamiento inmediato.

```
IN      AX,60H    ; Se lleva a AL el contenido de la puerta 60H y a AH
                  ; el de la puerta 61H.
OUT     60H,AL    ; Se enviará a la puerta 60H el contenido de AL.
OUT     60H,AX    ; Se sacará por la puerta 60H el contenido de AL y  por 61H el de AH.
```

- Registro. En todo el entorno[15] (0000 a 0FFFFH) se podrá acceder a puertas de E/S empleando a DX como puntero. Para las operaciones de 16 bits, DX apuntará a la dirección más baja de la puerta y automáticamente se incrementará (en la unidad generadora de direcciones) para apuntar a la dirección más alta. El valor de DX no se altera en toda la operación.

```
MOV     DX,789AH    ; Se inicializa DX para apuntar a 789AH
IN      AL,DX  ; Se lleva al registro AL el contenido de la puerta 789AH
IN      AX,DX  ; Se lleva a AL el contenido de la puerta 789AH y a AH el de la puerta
               ; 789BH. DX no se altera.
OUT     60H,AL ; Se enviará a la puerta 60H el contenido de AL.
OUT     60H,AX ; Se sacará por la puerta 60H el contenido de AL y  por la 61H el de AH.
```

Nemónico	Significado	Formato	Operación	Flags Afect.
IN	Entrada	IN An,F	(F) $\rightarrow$ (An)	Ninguno
OUT	Salida	OUT D,An	(An) $\rightarrow$ (D)	Ninguno

Donde:

An = Acumulador, AL o AX.

F = Fuente. Puede ser un valor (direccionamiento directo) o DX.

D = Destino. Puede ser un valor (direccionamiento directo) o DX.

8.2.3 Las instrucciones de intercambio (XCHG o exchange).

La instrucción MOV cambia el contenido del registro o posición de memoria destino, pero a afecta a la fuente. La instrucción XCHG es el intercambio de contenidos entre fuente y destino. Por ejemplo la instrucción

```
XCHG    AX,DX
```

permuta los contenidos de loa registros AX con DX. En forma simbólica:

(AX original) $\rightarrow$ (DX)

(DX original) $\rightarrow$ (AX)

Nemónico	Significado	Formato	Operación	Flags Afect.
XCHG	Intercambio	XCHG D,F	(F) $\leftrightarrow$ (D)	Ninguno

Fig. 20. Resumen de las instrucciones XCHG

En la Fig. 20 se presentan resumidas las características de la instrucción XCHG.

[15] Incluyendo también al rango 00 – 0FFH accesible por direccionamiento directo.

Destino	Fuente
Acumulador	Reg. 16
Memoria	Registro
Registro	Registro
Registro	Memoria

Fig. 21. Operandos permitidos para la instrucción XCHG.

8.2.4 La traducción de byte (XLAT).

La instrucción de traducción (translate) XLAT suele ser empleada para la búsqueda en tablas. Por ejemplo, supongamos tener que traducir símbolos recibidos en el código EBDIC al código ASCII. Para ello crearemos una tabla de conversión como la de la Fig. 22, en la que se escribirán los equivalentes ASCII al EBCDIC 00, en el byte siguiente de la tabla el equivalente ASCII al EBCDIC 01 y así sucesivamente hasta el equivalente ASCII del EBCDIC FFH.

Byte de la tabla	Contenido de la tabla
00	Equivalente ASCII del EBCDIC 00
01	Idem del EBCDIC 01
02	Idem del EBCDIC 02
....	
....	
FE	Idem del EBCDIC FE
FF	Idem del EBCDIC FF

Fig. 22. Tabla de traducción entre EBDIC y ASCII

El valor EBCDIC a traducir se cargará en AL (empleado como offset, lo cual permite acceder a tablas de 256 bytes como máximo), en BX se cargará la dirección de inicio de la tabla. El registro de segmento DS tendrá el valor adecuado para poder acceder a la dirección física en la que se hallará la tabla de conversión.

En estas condiciones, al ejecutar la instrucción XLAT se reemplazará en valor anterior del registro AL por el contenido de la tabla apuntada por BX + AL.

Otro ejemplo típico es el uso de XLAT para obtener la temperatura medida por una termocupla (alineal) y cuya ecuación característica es muy difícil de implementar. Se genera una tabla de conversión de mV (en formato binario) a ºC.

En resumen, XLAT se utiliza para acceder a tablas de conversión.

Nemónico	Significado	Formato	Operación	Flags Afect.
XLAT	Traducción	XLAT	(DS:(AL)+(BX)) $\rightarrow$ (AL)	Ninguno

8.2.5 La carga de dirección efectiva (LEA), del segmento de datos (LDS) y del segmento Extra (LES).

Estas instrucciones se emplean para cargar directamente los registros de segmento y otro registro de propósito general, en una sola operación y desde memoria. Así por ejemplo:

```
LEA     SI, [DI + BX + 5]          ; Supongamos que DI = 100H, BX = 0020H.
                                   ; Se cargará en el registro SI el valor 100H + 0020H + 5 →125H.
```

LDS y LES son similares a LEA, salvo que se agregará en el nemónico un registro de propósito general. Se cargarán en forma simultánea el registro de segmento y un registro desde memoria.

8.3 Instrucciones aritméticas

El 8088 contiene un variado conjunto de instrucciones aritméticas. Desde las básicas de suma, resta, multiplicación y división, con las variantes dadas por el modo de direccionamiento incluyendo operaciones con bytes o palabras signadas o no signadas, BCD empaquetado y desempaquetado o valores ASCII.

En los flags quedará una imagen resumida de la última operación aritmética (o lógica).Los indicadores (Ver 4.4) afectados por las operaciones aritméticas son:

a) El flag de acarreo CF.

b) El acarreo auxiliar AF entre el nibble (paquete de 4 bits) menos significativo y el más significativo.

c) El flag de signo S.

d) El flag de cero Z.

e) El flag de paridad PF.

f) El flag de desborde OF.

En la Fig. 23 se presentan las instrucciones aritméticas divididas en subgrupos funcionales.

Suma	
ADD	Suma de byte o palabra
ADC	Suma de byte o palabra con el estado del acarreo anterior
INC	Incrementar en 1 byte o palabra
AAA	Ajuste ASCII de la suma.
DAA	Ajuste decimal de la suma
Sustracción	
SUB	Resta de byte o palabra.
SBB	Resta de byte o palabra con el préstamo de la operación anterior.
DEC	Decrementar en 1 un byte o palabra.
NEG	Complementar byte o palabra.
AAS	Ajuste ASCII de la resta.
DAS	Ajuste decimal de la resta.
Multiplicación	
MUL	Producto de bytes o palabras no signadas.
IMUL	Producto de bytes o palabras enteras.
AAM	Ajuste ASCII del producto.
División	
DIV	Cociente de bytes o palabras no signadas.
IDIV	Cociente de bytes o palabras enteras.
AAD	Ajuste ASCII del cociente.
CBW	Convierte byte a palabra.
CWD	Convierte palabra a doble palabra.

Fig. 23. Resumen de las instrucciones aritméticas

8.3.1 Instrucciones de suma: ADD, ADC, INC, AAA y DAA.

En las Fig. 24,Fig. 25 y Fig. 26 se resumen las principales características de las instrucciones aritméticas. En la Fig. 24 se presenta el Resumen de las instrucciones. Allí se emplea la misma nomenclatura definida anteriormente. D es el registro o posición de memoria ***D***estino y F corresponde al ***F***uente. Los flags mantienen el significado y la nomenclatura definida en *4.4 Indicadores (flags):*

En la Fig. 25 y en la Fig. 26 se observan los operandos permitidos para las instrucciones de suma e incremento.

Debe destacarse que para las instrucciones de ajuste decimal y ASCII de la suma y resta, uno de los operandos deberá ser AL, que también será el destino.

Nemónico	Significado	Formato	Operación	Flags afectados
ADD	Suma	ADD D,F	(F)+ (D) → (D) Carry → (CF)	AF, CF, OF, SF, ZF, PF
ADD	Suma	ADD D,F	(F)+(D)+ (CF) → (D) Carry → (CF)	AF, CF, OF, SF, ZF, PF
INC	Incrementar en 1	INC D	(D) + 1 → (D)	OF, SF, ZF, AF, PF
AAA	Ajuste ASCII de la suma	AAA		AF, CF, OF, SF, ZF, PF
DAA	Ajuste decimal de la suma	DAA		AF, CF, OF, SF, ZF, PF

Fig. 24. Instrucciones de suma aritméticas

Nótese que INC (utilizada habitualmente para manejar punteros) no afecta al flan de acarreo, pues no se desea que el movimiento de punteros interfiera con operaciones aritméticas.

Destino	Fuente
Registro	Registro
Registro	Memoria
Memoria	Registro
Registro	Inmediato
Memoria	Inmediato
Acumulador	Inmediato

Fig. 25. Operandos permitidos para las instrucciones ADD y ADC

Destino
Reg 16
Reg 8
Memoria

Fig. 26. Operandos permitidos para la instrucción INC.

8.3.1.1 Ejemplos:

a) Supongamos que AX = 1100H, BX 0ABH. ¿Cuál es el contenido de los registros anteriores al ejecutar la instrucción ADD AX,BX?.

- Según lo expresado anteriormente, el orden de los operandos indica cual será el destino. En este caso el resultado quedará en AX (el que se halla a la izquierda). Por lo tanto:

(BX) + (AX) = 0ABH + 1100H = 11ABH → (AX) y BX no se ve afectado.

a) Repetir el ejemplo anterior pero para la instrucción ADC AX,BX. Suponer que como resultado de una instrucción aritmética anterior CF = 1.

(BX) + (AX) + (CF) = 0ABH + 1100H + 1 = 11ACH → (AX)

b) Suponer que AX = 1234H, BL = 0ABH y el contenido de una variable de tipo palabra en memoria llamada SUM vale 00CDH y el valor inicial del CF = 0. Analice el contenido de los registros al ejecutar el programa:

```
ADC   AX, word ptr SUM    ; (AX) + (SUM) + (CF) = 1234H + 00CDH + 0 = 1201H → (AX)
                          ; (CF) = 0
ADC   BL, 05H             ; (BL) + 5 + (CF) = 0ABH + 5 + 0 = 0B0H → (BL)
INC   word ptr SUM        ; (SUM) + 1 = 00CDH + 1 = 00CEH → (SUM)
```

d) Supongamos que AL = 32H, BL = 38H. Analicemos el resultado de ejecutar el programa:

```
ADD   AL,BL               ; (AL) + (BL) = 32H + 38H = 6AH → (AL)
DAA                       ; Analiza si alguno de los nibbles de AL es mayor que 9 o si hubo
                          ; acarreo o acarreo auxiliar. En caso de darse alguna de esas
                          ; condiciones se suma 6 al nibble. En este caso:
                          ; 6AH + 6 = 70H → AL
```

e) Supongamos que AL = 32H (en ASCII es el número **2**), BL = 34H (en ASCII es el **4**) y AH = 0. Analicemos el resultado de ejecutar el programa:

```
ADD   AL,BL               ; (AL) + (BL) = 32H + 34H = 66H → (AL)
AAA                       ; Realiza AH + AL y hace el ajuste del resultado a ASCII.
                          ; AH = 0, AL = 6 (ASCII), CF = AF = 0.
```

8.3.2 Instrucciones de resta.

Su explicación es muy similar a la de la suma, con las excepciones de que en lugar de la suma con acarreo se tendrá la resta con préstamo (o borrow) y en lugar del incremento en 1, se tendrá el decremento en 1.

Para el caso de la resta, se tiene una variante que en la instrucción CMP (compare) que es equivalente a la SUB, salvo que el resultado de la resta (posicionando los flags en el valor adecuado), no modifica a los registros intervinientes (F) - (D) **no** reemplaza a (D).

8.3.3 Instrucciones de multiplicación y división.

Históricamente, en los primeros microprocesadores, cuando se deseaba realizar un producto o cociente, se llamaba a una subrutina que por medio de un algoritmo más o menos ingenioso cumplimentaba con el pedido.

A partir del 8088, se implementaron esas operaciones por hardware, es decir circuitos que resuelven los productos y cocientes en un tiempo mucho menor que el que demora un programa.

El 8088 dispone de instrucciones de multiplicación y división de magnitudes binarias (signadas y no signadas) y BCD. En la Fig. 27, se presenta un Resumen de dichas instrucciones.

Debe notarse que para el caso del producto, se especifica sólo uno de los operandos (ya sea de 8 como de 16 bits). El otro operando, se supone que será AL para operaciones de 8 bits ó AX para operaciones de 16 bits. El destino será o bien AX para el producto de dos magnitudes de 8 bits o bien el par DX,AX para el caso de magnitudes de 16 bits.

Para el caso de la división, también se especifica solamente el denominador. Para el caso de que el mismo sea de 8 bits, el numerador será de 16 bits y estará formado por AX. Al efectuar el cociente, AL contendrá el resultado y AH el resto. Si el denominador es de 16 bits, el numerador será el conjunto DX,AX y al efectuar el cociente, el resultado quedará en AX y el resto en DX.

En la Fig. 28 se observan los modos en que se proveerán los operandos para estas dos instrucciones.

En la Fig. 27, se observa también la existencia de dos instrucciones de ajuste. Esto se debe a que las operaciones de producto y cociente, el microprocesador las realiza con operadores binarios. Para poder emplear cifras BCD se deberán hacer conversiones binario a BCD y viceversa.

La primera de ellas es AAM (Ajuste ASCII después de la multiplicación). Se supone que se han multiplicado dos cifras BCD desempaquetadas (los nibbles más significativos de los operandos deben ser cero). Luego de esa instrucción, al ejecutar AAM, quedará en AH el cociente del resultado/10 y en AL el módulo 10 del resultado.

La segunda instrucción de ajuste es AAD. Ésta realiza un ajuste anterior a un cociente. Supóngase tener dos cifras BCD desempaquetadas en AX. Para poder hacer algún cociente, será necesario convertirlas a binario empleando AAD. En AL quedará el producto de AH*10 + AL. En AH quedará cero.

Las últimas dos instrucciones expanden bytes a palabras (CBW), repitiendo el bit más significativo y palabras a dobles palabras (CWD).

Nemónico	Significado	Formato	Operación	Flags afectados
MUL	Producto (no signado)	MUL F	(AL)*(F8[16])→(AX)	AF, CF, OF, SF, ZF, PF
DIV	Cociente (no signado)	DIV F	***1.-*** ((AX)/(F8))→ (AL) *Si vale 0FFH, se generará una interrupción de división por 0. (Ver Cap. 3).* Resto → (AH). ***2.-*** ((DX,AX)/(F16[17]))→(AX) *Si vale 0FFFFH, se generará una interrupción de división por 0. Ver Cap. 3.* Resto → (DX)	AF, CF, OF, SF, ZF, PF
IMUL	Producto entero (signado)	IMUL F	(AL)*(F8[18])→(AX)	AF, CF, OF, SF, ZF, PF
IDIV	Cociente entero (signado)	IDIV F	***1.-*** ((AX)/(F8))→ (AL) Resto → (AH). ***2.-*** ((DX,AX)/(F16[19]))→(AX) *Si es positiva y **>** 7FFFH o si es negativo y < 8001H, se generará una interrupción de división por*	AF, CF, OF, SF, ZF, PF

[16] Fuente para la operación (registro, memoria) de 8 bits de extensión.

[17] Fuente para la operación (registro, memoria) de 16 bits de extensión.

[18] Fuente para la operación (registro, memoria) de 8 bits de extensión.

[19] Fuente para la operación (registro, memoria) de 16 bits de extensión.

			0. Resto → (DX)	
AAM	Ajuste de AL para el producto.	AAM	((AL)/10)) → (AH) Resto ((AL)/10) → (AL)	OF, SF, ZF, AF, PF
AAD	Ajuste de AX para el cociente.	AAD	(AH)*10) + (AL) → (AL) 00 → (AH)	OF, SF, ZF, AF, PF
CBW	Convertir byte a palabra	CBW	El bit más significativo de AL se repite en todo AH	Ninguno
CWD	Convertir palabra a doble palabra (32 bits)	CWD	El bit más significativo de AX se repite en todo DX	Ninguno

Fig. 27. Resumen de las instrucciones de producto y cociente

Fuente
Registro 8 bits
Registro 16 bits
Memoria 8 bits
Memoria 16 bits

Fig. 28. Operandos permitidos para el producto y cociente

8.4 Instrucciones lógicas

El 8088 dispone de un conjunto de operaciones lógicas como AND, OR, OR exclusiva y NOT. Todas ellas se realizan bit a bit y las tres primeras requieren un segundo operando.

Nemónico	**Significado**	**Formato**	**Operación**	**Flags afectados**
AND	AND lógica	AND D,F	(F).(D) → (D)	CF, OF, SF, ZF, PF AF indefinido
OR	OR inclusiva lógica	OR D,F	(F) + (D) → (D)	CF, OF, SF, ZF, PF AF indefinido
XOR	OR exclusiva lógica	XOR D,F	(F) ⊕ (D) → (D)	CF, OF, SF, ZF, PF AF indefinido
NOT	NOT lógica	NOT D	(/D) → (D)	Ninguno

Fig. 29. Instrucciones lógicas.

Destino	Fuente
Registro	Registro
Registro	Memoria
Memoria	Registro
Registro	Inmediato
Memoria	Inmediato
Acumulador	Inmediato

Fig. 30. Operandos permitidos para AND, OR y XOR

Destino
Registro
Memoria

Fig. 31. Operandos permitidos para la Instrucción NOT

En Fig. 29 se observa la sintaxis y las características generales de las operaciones lógicas. En las Fig. 30 y Fig. 31 se observan los operandos permitidos, de donde se pueden concluir los modos de direccionamiento lícitos.

La importancia de las operaciones lógicas radica en que permiten aislar o forzar bits de un byte o una palabra por medio de una *máscara*. Por ejemplo:

a) Forzar un bit a cero. En la instrucción: AND AL,11101111b, al realizar la operación AND lógico bit a bit entre AL y la máscara 11101111 en binario, se observa que los bits 7, 6, 5, 3, 2, 1 y 0 de AL no se ven afectados por la operación, mientras que el bit 4, cualquiera hubiera sido su valor anterior se convierte en cero.

b) Forzar un bit a uno. En la instrucción OR BX,0001000000000000b, se puede ver que cualquiera hubiera sido el valor anterior del bit 12 de BX, será forzado a valer 1.

c) Permutar un bit. En la instrucción XOR DH,00000100b y en base a la definición de la función OR exclusiva, el bit 2 de DH será obligado a permutarse. Si valía 1 pasará a valer 0 y viceversa.

d) Analizar el estado de un bit. Con la instrucción AND CL,00010000b, el resultado que quedará en CL, será 00H si el bit 4 de dicho registro era cero antes de la operación o será distinta de cero si dicho bit era 1. En base a ello el flag de cero tomará el valor de acuerdo al resultado de esta operación. La utilidad de esta operación consiste en que posteriormente se podrá emplear una instrucción de salto condicional en base al tetado de dicho flag (JZ o JNZ).

Esta última instrucción tiene una variante que es la instrucción TEST que es equivalente a la AND, salvo que el resultado de la AND (posicionando los flags en el valor adecuado), no modifica a los registros intervinientes (F) . (D) **no** reemplaza a (D).

Si bien por simplicidad se ha ejemplificado empleando registros, todas estas operaciones en base a bits son extensivas a los modos de direccionamiento permitidos.

8.5 Instrucciones de salto (Jump).

El propósito de las instrucciones de salto es alterar el orden consecutivo del programa. Esas disrupciones al orden podrán utilizarse para hacer lógica o tomar decisiones. A diferencia de lo planteado en el capítulo anterior al referirnos a las Subrutinas, no se deberá almacenar ninguna dirección de retorno, ya que el salto no regresa al punto de partida.

El 8088 dispone de dos tipos de saltos. El salto incondicional y el condicional. En el caso de los incondicionales, no se analiza el estado de los flags y se altera el orden de ejecución del programa ***siempre.*** En cambio para los saltos condicionales, se analizará el estado (tanto por cero como por 1) de uno o más flags para producir el salto.

8.5.1 Salto incondicional.

Nemónico	Significado	Formato	Operación	Flags afectados
JMP	Salto incondicional	JMP operando	Salto a la dirección indicada en Operando.	Ninguno

Fig. 32. Instrucción de salto incondicional

Fuente
Short
Near
Far
Puntero en memoria 16 bits
Puntero Registro 16 bits
Puntero en memoria 32 bits

Fig. 33. Operandos permitidos

En la Fig. 33, se observa la existencia de tres alternativas básicas para el salto. El salto corto (short) implica cambiar el puntero a instrucción IP con un byte de desplazamiento en complemento a 2 (rango -128 a + 127) del valor actual de IP.

El salto intra segmento (dentro del mismo segmento) o Near implica cambiar el IP por un nuevo valor dando los 16 bits del nuevo IP. Para los saltos lejanos o FAR se cambiará tanto el IP como el CS, por lo que deberán darse 4 bytes con la nueva dirección.

Además tenemos la posibilidad de proveer la dirección del salto por medio del contenido de una palabra en memoria o por medio de un registro de 16 bits, como por ejemplo:

```
JMP BX
```

implica saltar (sin cambiar CS) a la dirección apuntada por BX. En otras palabras, el contenido de BX es copiado a IP.

Análogamente puede emplearse otro nivel de indirección por medio de:

```
JMP  [BX]
```

en la que se emplea a BX como puntero a memoria en donde se sacarán 16 bits que se copiarán en IP.

Para el salto lejano, ínter segmento (entre dos segmentos distintos) o FAR se dará un medio de apuntar a memoria en donde se hallarán 4 bytes de donde se extraerán IP y CS de la nueva dirección a ejecutar. Por ejemplo:

```
JMP  DWORD PTR [DI]
```

donde se utilizará DS:DI para extraer 4 bytes de memoria en el orden IP_L, IP_H, CS_L y CS_H.

8.5.2 Saltos condicionales.

Son todos saltos cortos o shorts en los que se analizarán el estado de los flags para saltar o no en el rango -128 a +127 bytes en torno del valor actual del IP.

Nemónico	Significado	Formato	Operación	Flags afectados
Jcc	Salto condicional	Jcc operando	Salto a la dirección indicada en Operando si se verifica el estado de los flags del código de condición..	Ninguno

Fig. 34. Instrucción de salto condicional

Nemónico	Significado	Condición
JA	above (por encima) NO signado.	CF=0, ZF=0
JAE	above or equal (encima o igual)	CF=0
JB	below (por debajo)	CF=1
JBE	below or equal	CF=1 ó ZF=1
JC	acarreo	CF=1
JCXZ	registro CX=0	(CF ó ZF) = 0
JE	igual	ZF=1
JG	mayor o greater (signado)	ZF=0 y SF=OF (overflow)
JGE	mayor o igual	SF=OF
JL	menor o less (signado)	(SF ⊕ OF)=1
JLE	menor o igual	((SF ⊕ OF)=1)+ZF)=1
JNA	not above	CF=1 ó ZF=1
JNAE	not above or equal	CF=1
JNB	not below	CF=0
JNBE	not below nor equal	CF=0 y ZF=0
JNC	not carry	CF=0
JNE	not equal	ZF=0
JNG	not greater	((SF ⊕ OF)=1)+ZF)=1
JNGE	not greater or equal	(SF ⊕ OF)=1
JNL	not less	SF=OF
JNLE	not less or equal	ZF=0 y SF=OF
JNO	not overflow	OF=0
JNP	not parity	PF=0
JNS	not sign (no negativo)	SF=0
JNZ	not zero	ZF=0
JO	overflow	OF=1
JP	parity	P=1
JPE	parity even (par)	PF=1
JPO	parity odd (impar)	PF=0
JS	sign (positivo)	SF=1
JZ	zero	ZF=1

Fig. 35. Saltos condicionales

8.6 Otras.

Se propone al alumno que realice un análisis similar a los anteriores (consultando la bibliografía y las hojas de datos de los microprocesadores) para las instrucciones:

a) Shift.
b) Rotate.
c) Loop.
d) Llamado a subrutinas.
e) Push y Pop.
f) Operaciones de cadenas (strings).

9. Bibliografía recomendada.

a) The 8088 and 8086 Microprocessors (second edition). Triebel y Singh. Prentice Hall 1997.
b) The 8086/8088 family. Design, programming and Interfacing. Uffenbeck. Prentice Hall 1987.
c) The 8086 book. Rector - Alexy. Mc Graw Hill 1980.
d) Introducción al microprocesador 8086/8088. Morgan - Waite. Mc Graw Hill. 1982.

10. Cuestionario.

10.1 Un camino de evolución. el 8088.

10.1.1 Arquitectura Interna.

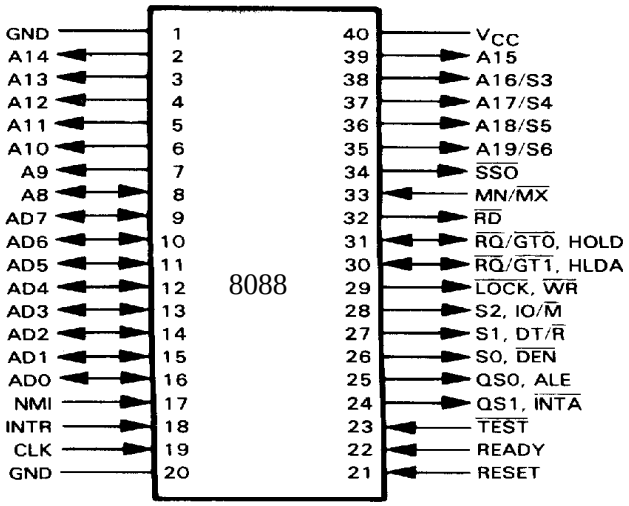
a) ¿Existen diferencias circuitales entre el 8086 y el 8088?. ¿Y a nivel del repertorio de instrucciones?.

b) ¿Qué entiende por coprocesador aritmético?. ¿Puede existir el coprocesador sin el procesador principal? ¿Y al revés?.

c) ¿Qué entiende por modo máximo y modo mínimo?. Para operar con un coprocesador, ¿el 8088 deberá estar en modo máximo o mínimo?.

d) Describa las operaciones que realiza el procesador principal que no puede realizar el coprocesador aritmético.

e) ¿Cuál es el significado de cada una de las patas del 8088?. ¿Es el mismo en modo máximo que en modo mínimo?.



f) Explique para que se usan las siguientes líneas del 8088, especifique si son de entrada o de salida:

- AD0-AD7 Address Data Bus
- A8 -A15 Address Bus
- A16-A19/S3-S6 Address Bus / Status
- ALE Address Latch Enable
- DEN/ Data Enable
- DT/*R Data Transmit / Receive/

g) ¿Qué significa que el 8088 tiene multiplexadas en el tiempo la parte baja de las direcciones con los datos?

h) ¿Qué función cumplen los latches que se conectan al 8088?

i) ¿Qué función cumple la señal ALE (Address Latch Enable)?

j) ¿Qué se indica cuando la señal ALE se encuentre en "1"?. ¿Y cuando esta en "0"?

k) Busque la hoja de datos del 74HC373. Analice la tabla de verdad que aparece en la misma. Conéctelo adecuadamente al 8088. A la salida del 373, ¿qué obtiene?.

l) Repita el punto anterior para los circuitos integrados 74HC244 y 74HC245.

m) ¿Cuál es el significado de la pata DT/*R y dónde se conecta?. ¿Para qué?.

n) ¿Cómo le indica el 8088 a una memoria que quiere leerla?. ¿Y Qué quiere escribirla?

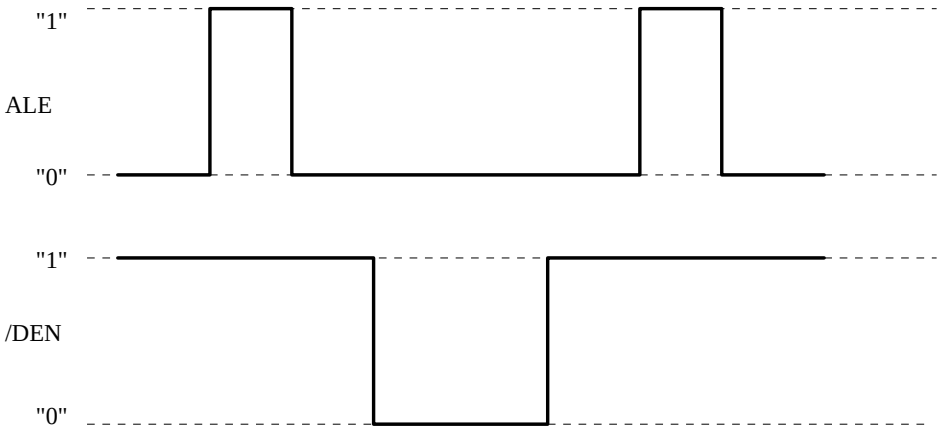
o) El 74HC245 puede estar en alta impedancia. ¿Que señal del 8088 usa para habilitarlo?. ¿En qué momento sucede ello?.

p) Los siguientes diagramas temporales corresponden a las señales ALE y DEN/ de salida del 8088. Marque en los mismos:

q) cuando hay datos válidos.

r) cuando hay direcciones válidas.

s) cuando el bus multiplexado se halla en estado de alta impedancia ("tri-state").



a) ¿Puede darse el caso de que /DEN y ALE estén simultáneamente activas?. ¿Y desactivas?.

b) ¿Cuál es la necesidad de la seña IO/M* (Input Output/ Memory) del 8088?. ¿Cuándo esta en 1 y cuándo en cero?. ¿Con que instrucciones se las hace valer "0" y "1"?.

c) ¿Pueden coexistir la posición de memoria 0 y la posición de entrada / salida 0?. ¿Por que?.

d) Si a la entrada del buffer 74HC245 se encuentra el bus multiplexado de direcciones y datos, ¿qué bus se encuentra a la salida?

e) ¿Qué bus se conecta a la entrada del latch 74HC373?. ¿Qué bus se obtiene a la salida cuando ésta está habilitada?

f) ¿Cuál es el objetivo de conectar un segundo 74HC373 a las líneas más altas del bus de direcciones del 8088?.

g) ¿Qué función cumple la señal de salida RESET del 8088?. ¿Cuánto tiempo debe estar activa como mínimo?.

h) ¿Qué significa "POWER_ON_RESET"?. Dibuje el circuito que se utiliza para generarlo.

i) ¿Por qué la señal de RESET que entra al 8284 (activa baja) tiene que permanecer en cero durante 5 ciclos de reloj?.

j) ¿Qué función cumple la señal READY de entrada al microprocesador?.

k) ¿Qué efecto tiene un "1" lógico en la entrada HOLD de entrada al 8088?. ¿Con qué señal contesta el microprocesador a un pedido de HOLD?

l) ¿Qué operación se realiza mientras los buses están inactivos (se esta ejecutando alguna instrucción que no los usa)? ¿Cuál es la ventaja de este procedimiento?

m) Dibuje la arquitectura interna del microprocesador 8088, especificando el contenido de cada una.

n) ¿Cuál es la función de la BIU ("Bus Interface Unit")?

o) ¿Cuál es la función de la EU ("Execution Unit")?. ¿Qué funciones realiza la unidad de control?

p) ¿Qué es un registro?. ¿Qué ventaja tiene usar un registro en lugar de una posición de memoria?

q) ¿Qué función cumple la cola?. Y en el 80386 y posteriores ¿la memoria cache?.

10.2 Registros fundamentales.

a) Con un registro de 16 bits ¿cuantas posiciones de memoria se pueden direccionar?

b) El manual del 8088 dice que este es capaz de direccionar 1 Mbyte. ¿Cuántos bits se necesitan para direccionar 1 Mbyte? Explique como logra el microprocesador direccionar 1 Mbyte si los registros que se emplean son de 16 bits.. ¿Cómo se denominan los 20 bits generados?.

c) ¿Qué unidad del microprocesador es la encargada de generar la dirección física?.

d) Esquematice todos los registros principales del 8088, indicando su longitud y su uso dedicado (si lo tienen), especificando las partes constituyentes de aquellos que pueden ser divididos.

e) ¿Cómo se llaman los registros de segmento y que longitud en bits tienen?. ¿Cómo se genera una dirección física? ¿Qué registros se usan como base y como desplazamiento?. Ejemplifique aclarando la cantidad de bits que tiene cada registro y la dirección física por ellos generada. Explique claramente el procedimiento empleado para generarla.

f) En un programa se pueden diferenciar claramente 3 áreas. ¿Cuáles son?. ¿Cómo las diferencia?. ¿Qué funciones cumple cada una de ellas?.

g) La búsqueda de códigos de operación y operandos, ¿Qué par de registros indefectiblemente emplea?.

h) ¿Qué es una dirección efectiva?

i) ¿Qué registro de segmento se utiliza por omisión cuando se apunta a la zona de datos?

j) ¿Que es la pila y qué registro de segmento se utiliza por defecto cuando se la apunta?. ¿Qué tipo de memoria es la pila: FIFO ("First Input First Output") o LIFO ("Last Input First Output")?.

k) ¿Qué es el SP ("Stack Pointer")?. ¿Qué longitud tiene (en bits)?. ¿Qué es el BP ("Base Pointer")? ¿Qué longitud tiene?

l) ¿Para qué utiliza el ES ("Extra Segment")?

m) ¿En qué valor se inicializan los registros después de un RESET?

n) ¿Cuál es la dirección física donde se encuentra el primer código de operación a ejecutarse inmediatamente después de un RESET? ¿Cuál es el valor de CS:IP?. ¿Cuántos bytes quedan desde esa posición hasta el final del mapa de memoria del 8088?

o) ¿Qué es la cola del microprocesador?. Describa detalladamente cómo entiende que debería operar al ejecutar un programa en un 8088. ¿Qué tipo de arquitectura tiene la cola?

p) Indique las características de los registros AX, BX, CX y DX, indicando si pueden emplearse para direccionar a memoria.

q) ¿Qué son los indicadores o "flags"?. ¿Dónde se encuentran?. ¿Cómo se los altera?. ¿Cómo se toman decisiones en base al estado de los flags?.

r) Indique las características de los registros SI, DI y BP indicando si pueden emplearse para direccionar a memoria y en caso afirmativo que registro de segmento emplean por omisión (default).

s) ¿Para que casos no se pueden alterar los segmentos asociados a los registros SI y DI?.

t) ¿En qué se diferencian el BP ("Base Pointer") y el SP ("Stack Pointer") si ambos se usan como punteros a la pila?

u) Si IP = 100 y SP = 100. ¿Apuntan a la misma posición de memoria?

v) Suponga que SS = 7FF0H y SP = 100H. Indique como se halla la pila antes y después de guardar en ella la palabra 1234H.

w) ¿En qué sentido crece normalmente la memoria de datos, de las posiciones más bajas a las más altas o al revés?. ¿Y la pila?

x) ¿Cuál de estos valores le parece que es incorrecto para inicializar al SP cuando se ha definido una pila de 256 bytes?. ¿0 o 256?. ¿Por que?

y) Si antes del llamado a una subrutina el IP = 100H y el SP = 100H, ¿Qué valores tienen ambos después de ejecutar una instrucción CALL a la dirección 1000H?.

z) Repetir el punto anterior suponiendo que se realiza un llamado FAR a F800:1000.

aa) Explique claramente que registros se ven afectados y la procedencia de los datos al ejecutar las siguientes instrucciones.

- MOV AX, SI
- MOV AX, [SI]
- MOV DL, [SI]

10.3 Ciclos de instrucción, de maquina y de reloj del 8088.

a) Defina ciclo de instrucción y como se subdivide.

b) Si el 8088 tiene un cristal de 15 MHz ¿Cuál es el tiempo de instrucción de:

- AND reg, reg
- MUL reg? Usar las hojas de datos del manual del 8088.

c) ¿Qué es un ciclo de máquina?

d) Enumere los ciclos de maquina que existen.

e) Describa los ciclos de máquina que encuentra en la siguiente instrucción:

- INC AX

f) ¿Cuál es la diferencia entre el ciclo de máquina "lectura de memoria" y "búsqueda de código de operación"?

g) ¿Qué unidad es la encargada de decodificar los códigos de operación?

h) ¿Qué es un ciclo de reloj?. ¿Cuál es la función del reloj en el 8088?. ¿Cuánto vale el estado o ciclo de reloj del 8088 cuando su frecuencia de cristal es de 15 MHz?.

i) ¿Qué circuito integrado hace las veces de oscilador para conectar al 8088?. Describa claramente cada una de las señales en sus patas y si se hallan vinculadas con la frecuencia del cristal, su relación.

10.4 Ejemplo de movimiento de datos.

a) ¿Qué entiende por lenguaje de máquina u objeto?.

b) ¿Qué función cumple el lenguaje Assembler?

c) Indique cuál de las siguientes expresiones corresponde a una etiqueta válida o a un número hexadecimal válido, justificando la respuesta.

- 6F
- F6
- 0F6

d) ¿Cuál es la ventaja de usar etiquetas cuando se desea realizar una instrucción de salto en un programa Assembler?

e) ¿Qué es el hipotético IR ("Instruction Register")? ¿Qué longitud tiene (en bits)?. ¿Es accesible al usuario?.

f) El IP ("Instruction Pointer") una vez que se cargo un código de operacion en el IR ("Instruction Register") generalmente se auto incrementa. ¿Cuándo no lo hace?.

g) ¿Qué es el MAR ("Memory Address Register")? ¿Qué características tiene?. ¿Cómo se lo divide?. ¿Es accesible al usuario?

h) Sea la instrucción MOV AX, [2010]

i) ¿Qué sucede con el IP ("Instruction Pointer") una vez que se sacó por el bus de datos el contenido de la posición de memoria [2010]?. ¿Y qué sucede con el contenido del registro MAR ("Memory Address Register")?.

j) ¿Cuántos ciclos de máquina puede identificar en la siguiente instrucción?. ¿De qué tipo de ciclos de máquina se trata?.

```
MOV     AX, [2010]
```

k) Sea la instrucción ADD AX, 0F6H. ¿Cómo se llama el registro al cual va a parar el segundo operando? ¿Qué longitud en bits tiene?. ¿Cómo se lo subdivide?. ¿Qué características tiene?.

l) Indique el contenido del acumulador y los flags luego de ejecutar los siguientes programas:

```
MOV     AX,0F6H          MOV     AX,0F6H
SUB     AX,100H          CMP     AX,100H
```

m) ¿En cuántos bytes se incrementa el IP ("Instruction Pointer") cuando se realiza y cuando no se realiza el siguiente salto condicional?

```
        JC   LABEL_1
        NOT  AX
LABEL_1:  MOV  [ 2010 ], AX
```

NOTA: el código de operación de: NOT AX es F7D0

n) ¿Cuál es el alcance máximo en bytes de un salto condicional?

o) ¿Cuál es el alcance máximo de los saltos condicionales?. Si por alguna razón debe producir una salto condicional de mayor rango, ¿qué triquiñuela emplea?

p) ¿Cuantos tipos de saltos no condicionales conoce y cuáles son sus alcances en bytes?

q) Analice el siguiente programa:

```
        MOV     AX, [ 2010 ]
        ADD     AX, 0F600H
ACA:    JNC     F6
        NOT     AX
F6:
```

Suponga que el contenido de la posición de memoria 2010H era 1234H. ¿Cuál será la siguiente operación que realizará el microprocesador después de la marcada con el rótulo ACÁ?.

r) Repetir el punto anterior para el siguiente programa:

```
        MOV     AX, [ 2010 ]
        CMP     AX, 0F600H
ACA:    JNA     F6
        NOT     AX
F6:
```

s) Repetir el punto anterior para el siguiente programa:

```
        MOV   AX, [ 2010 ]
        CMP   AX, 0F600H
ACA:    JG    F6
        NOT   AX
F6:
```

Analizar las diferencias en la operación signada y no signada.

t) Suponga que el contenido del registro DI es 57FAH. ¿Qué interpreta el microprocesador al encontrarse con las siguientes instrucciones?. Analice el contenido de la posición apuntada por DI y su anterior y posterior. Comente las respuestas.

```
MOV   [DI],1234H
MOV   [DI],123
MOV   [DI],12H
```

u) ¿Es lícito el movimiento de datos de memoria a memoria? ¿Por que?.

v) ¿Cómo guarda en la pila una variable?. ¿Cómo la rescata?. ¿De qué tamaño (en bits) son las variables?.

w) ¿Cuál es la capacidad de direccionamiento de puertas del 8088?. ¿Cuántas pueden accederse en forma directa?. ¿Qué otro método puede emplear para acceder la totalidad del mapa de E/S?.

x) ¿Qué instrucciones emplea para acceder a las puertas de E/S?.

y) ¿Qué entiende por puertas mapeadas como memoria?. ¿Cómo las implementa en la decodificación?. ¿Qué ventajas y desventajas presenta frente al mapeo como E/S?.

z) ¿Cómo puede el microprocesador y su lógica periférica discriminar entre la posición de memoria 0 y la puerta de E/S 0?.

aa) ¿Cuál es el registro que obligatoriamente debe emplearse como fuente y destino de las operaciones de 16 bits de E/S?. ¿Y para 8 bits?.

bb) ¿Puede una puerta de entrada encontrarse en la misma dirección que una puerta de salida?. ¿Por que?.

10.5 MODOS DE DIRECCIONAMIENTO PROPIOS ADICIONALES.

a) Cite por lo menos un ejemplo del modo de direccionamiento:

- De movimiento entre registros.
- De direccionamiento indirecto.
- De direccionamiento basado sin desplazamiento. Exprese los dos tipos de sintaxis posibles.
- De direccionamiento basado con desplazamiento de 8 y 16 bits. Exprese los dos tipos de sintaxis posibles.

b) ¿Afectan los movimientos de datos a los indicadores o Flags ("flags")? ¿Por qué?

c) Si el CF ("Carry Flag") se encuentra en "1" antes de realizar la instrucción:

```
MOV   AX, 0
```

¿en qué estado se encontrará después de ejecutarse la misma?

d) ¿Cómo haría para poner a cero (resetear) los flags de cero **(Z)** y de **(CY)** sin alterar el contenido de ningún registro.

10.6 REPERTORIO DE INSTRUCCIONES.

10.6.1 OPERACIONES ARITMETICAS.

a) ¿Cuáles son las operaciones aritméticas básicas?. ¿Cómo las modifican los modos de direccionamiento?.

b) ¿Las operaciones aritméticas afectan los flags?

c) Explique que realizan estas dos instrucciones y coméntelas.

- ADD [BX], CX
- ADC [BX], CX

d) Si el contenido de las posiciones de memoria 2010 y 2011 es:

[2010]: FFH

[2011]: FFH

e) ¿Cuál es el contenido de ambas una vez que se han ejecutado las instrucciones:

- INC byte ptr [2010]
- INC word ptr [2010]

Analice en ambos casos que indicara el flag de acarreo (CF, "Carry Flag") y el de cero ("Zero Flag").

f) Explique que realizan estas tres instrucciones, compárelas y coméntelas.

- SUB CX, BX
- SBB CX, BX
- CMP CX,BX

g) Si se realiza una multiplicación empleando el registro BL como segundo operando, ¿cuál es el primer operando y en que registro quedará el resultado? ¿Cuántos bits tendrá el mismo?

h) Al realizar la instrucción:

MUL CX

¿Cuál es el primer operando?. ¿Dónde se encontrará el resultado? ¿Cuántos bits tendrá el mismo?.

i) ¿Qué diferencia existe entre las instrucciones: IMUL y MUL?

j) Si se realiza una división usando el registro CL como segundo operando, ¿Cuál es el primer operando y dónde quedará el resultado? ¿Cuántos bits tendrá?.

k) Cuando se realiza la siguiente instrucción:

DIV BX

¿Cuál es el primer operando?. ¿Cuántos bits debe tener?. ¿Dónde se encuentra el resultado?. ¿Cuántos bits tendrá?

l) Analizando los ejemplos anteriores, ¿qué precaución deberá tomar cuando se va a realizar una división?. En caso de no tomarla, ¿qué error se producirá?.

m) ¿Qué diferencia existe entre las instrucciones: IDIV y DIV?

n) ¿Cuál es la utilidad de la aritmética decimal? ¿Cuáles instrucciones conoce?. ¿Qué registro deberá emplear indefectiblemente?

10.6.2 USO DE LAS INSTRUCCIONES LOGICAS.

a) ¿Cuáles son las instrucciones lógicas básicas?. Especifique la cantidad de operandos que requiere cada una. ¿Cómo las afectan los modos de direccionamiento?.

b) ¿Cuál es el resultado de la siguiente operación lógica:

AND AL, CH

Si el contenido de los registros es (AL) = 0F6H y (CH): 31H

c) Escriba la instrucción lógica básica que permite conocer el estado del bit 3 de la posición de memoria [2345H].

d) Si en el ejemplo anterior no se quiere alterar el contenido de la posición de memoria, ¿qué otra operación lógica debe usar?

e) ¿Cuál es la diferencia entre las instrucciones AND y TEST?.

f) Escriba los programas para:

- Poner a cero el bit 3 de la posición de memoria 2345H. Repita poniéndolo en "1".
- Obtener el complemento a uno de un operando.
- Obtener el complemento a dos de un operando.
- Complementar los bits impares de AL.

g) ¿Qué instrucción lógica le permite limpiar un registro (poner todos sus bits en cero) y también el bit de acarreo?. ¿Qué indicarán los flags de paridad, PF; de cero, ZF; de signo, SF; de desborde, OF y el de dirección, DF?. ¿Qué decisiones puede tomar analizando el estado de los flags?

h) En que se diferencian las instrucciones:

- JL ("Jump on Less ") con JB ("Jump on Below ")
- JG ("Jump on Greater") con JA ("Jump on Above ")

i) Analice los siguientes programas, cuando el contenido de BL es 0H y el de AL es 80H:

- CMP BL, AL

  JG ES_MAYOR
- CMP BL, AL

  JA ES_MAYOR

ES_MAYOR es el nombre de una etiqueta que se encontrará en algún lugar del programa.

j) ¿Qué realiza el ensamblador cuando lee NOT en la siguiente instrucción?

MOV AX, NOT 24H ¿NOT es (en este caso) una instrucción?

k) Escriba las instrucciones en Assembler que permitan conocer el estado del bit 6 de una variable de 1 byte llamada DATO y saltar a la posición de memoria ES_1 si fue "1" y ES_0 en caso contrario. No se desea conservar el valor original de la variable DATO.

l) Modifique las instrucciones anteriores de manera de que no se destruya la información contenida en la variable DATO.

m) Escriba las instrucciones que permitan forzar a "1" el bit 7 de la variable byte SAL_DATO_1 y forzar a "0" el bit 4 de la variable Word SAL_DATO_2.

n) Escriba la instrucción que permite complementar los bits pares de la variable de 1 byte llamada ESTADO.

10.6.3 LLAMADO A SUBRUTINAS.

a) ¿Qué es una subrutina?.

b) ¿Qué instrucción se usa para llamar a una subrutina?.

c) ¿Qué puede resguardar en la pila? ¿De cuántos bits?. ¿Qué instrucción usa para ello?. ¿Qué instrucción le permite recuperar lo guardado? Escriba algunos ejemplos. ¿Cómo puede emplear el registro BP?.

d) ¿Adónde apunta el IP ("Instruction Pointer") después de llamar a una subrutina?.

e) ¿En qué sentido varía el SP ("Stack Pointer") cuando se guardan datos en la pila?. ¿De cuántos bytes son estas variaciones?.

f) ¿Cuál es la diferencia entre un llamado a subrutina NEAR y FAR?.

g) Suponga el siguiente programa en el cual el SP inicial vale 9876H:

```
MOV   SI,1234H
MOV   BP,5678H
PUSH  BP
PUSH  SI
CALL  NEAR 0023H
```

Indique como queda la pila y los registros involucrados luego de ejecutarlo.

h) Repita el punto anterior suponiendo que la última instrucción es CALL FAR 3E2C:459F

i) ¿Qué hace la instrucción RET?. ¿Con qué valor se carga el IP?. ¿Cómo cambia el SP?.

j) Si la pila ocupa desde la posición de memoria 00H hasta la posición 3FH, ¿con qué valor inicialaría el SP?.

k) Para el problema anterior, ¿Qué sucede si se almacenan (con pushes o llamados a subrutina) 70 palabras?.

l) ¿Para qué se suele emplear el registro BP?.

m) ¿Pueden utilizarse las instrucciones PUSH y POP para copiar un registro en otro?.

n) Describa el estado de los registros AX y CX después de ejecutar el siguiente programa:

```
PUSH   AX
PUSH   CX
--     --
--     --
POP    AX
POP    CX
```

o) Es frecuente que dentro de una subrutina se resguarden registros en la pila. ¿Qué precaución debe tomar el programador antes de retornar al programa llamante de la subrutina?.

p) ¿Cómo actúa la instrucción RET en un procedimiento NEAR y en uno FAR?. ¿Cómo hace el microprocesador para diferenciar un caso del otro?.

10.6.4 TRANSFERENCIA DE DATOS.

a) La instrucción MOVSB permite llevar un byte de la posición de memoria apuntada por DS:SI a la posición de memoria apuntada por ES:DI. ¿Qué sucede con los punteros SI ("Source Index") y DI ("Destination Index") cuando se ejecuta dicha instruccion?. ¿Cuál es el efecto del flag D?.

b) Repetir el análisis para la instrucción MOVSW.

c) Escriba el programa que le permita llevar 8 bytes que se encuentran a partir de la posición de memoria MEM1 (fuente) a la posición de memoria MEM2 (destino). Use el prefijo REP.

d) Analizar la operatoria del siguiente programa

```
PUSH   AX
PUSH   [1234]
-
-
```

POP [3456]

POP BX

e) Escriba un conjunto de instrucciones que permitan realizar la misma operación por medio de instrucciones tradicionales de movimiento de datos. Compare los tiempos de ejecución. Saque conclusiones.

f) ¿Cómo haría para llenar una zona de memoria amplia (p.ej. 30K ó más) con ceros en el menor tiempo posible?.

g) ¿Cómo haría para buscar un byte en una tabla de origen y tamaño conocido?. ¿Y una palabra?.

10.6.5 VARIOS.

a) Escriba una línea en Assembler para reservar una posición de memoria de un byte, sin inicializar.

b) Si realiza operaciones con variables y no desea usar el Data Segment sino el Stack Segment, el Code Segment o el Extra Segment como lo indica?. ¿Conoce más de una sintaxis aceptada por el Assembler?. De un ejemplo de su uso en el siguiente caso:

MOV AX, [SI]

c) Indique que realiza la instrucción modificada.

d) ¿Qué uso tiene la instrucción XLAT?. Explíquelo mediante un ejemplo.

e) ¿Cuándo usa la instrucción XLAT para manejar una tabla, que tamaño máximo puede tener la tabla?. ¿Por qué?

10.6.6 DIAGRAMAS TEMPORALES.

a) ¿Cuál es la diferencia entre ciclo de reloj y estado?.

b) ¿Cuántos ciclos de reloj constituyen un ciclo del bus?

c) Especifique como procede la unidad de interfaz con el bus, BIU ("Bus Interface Unit") en cada ciclo del bus.

d) Explique como interactúan las unidades de interfaz con el bus, BIU, y la de ejecución, EU ("Execution Unit").

e) ¿Cuál de las dos unidades en que se divide el 8088 es la encargada de decodificar la instrucción?

f) ¿Qué unidad genera las direcciones?

g) ¿Qué actividad realiza la BIU cuando la EU no requiere leer o escribir información sobre los buses?.

h) ¿Qué información proporcionan las líneas de salida del microprocesador QS0 y QS1?.

i) ¿Qué flanco del ciclo de reloj se usa para sincronizar?

j) ¿Cómo procede el microprocesador durante el primer ciclo de reloj? ¿Qué señal genera?

k) Si ALE vale 0, en el bus multiplexado, ¿hay datos?. ¿Porqué?.

l) ¿Cuál es la función de las señales ALE, /DEN y DT/R?. ¿Existe alguna similitud entre ALE y /DEN?.

m) Durante cuantos ciclos de reloj permanecen estables la parte baja y la más alta del bus de direcciones?. Por este motivo ¿qué precaución se debe tomar al diseñar un circuito?

n) ¿Cuál es el ciclo de reloj para un microprocesador 8088 que opera con un cristal de 15 MHz?.

o) ¿Cuál es el tiempo típico de propagación de las señales a través de los buffers y latches (consulte las hojas de datos)?

p) ¿Qué relación temporal existe entre la bajada de la señal /WR ("Write") y datos estables en el

bus?

q) En un ciclo de máquina de lectura, ¿en qué ciclo de reloj baja la señal /RD?.

r) Defina tiempo de acceso de una memoria.

s) Para que el microprocesador 8088 tome información de la memoria, ¿cómo debe estar la señal /RD?.

t) Si se dispone de una memoria cuyo tiempo de acceso es mayor al tiempo de lectura previsto para el microprocesador y su cristal, ¿Cuál será el resultado de una lectura de la misma?.

u) Para frenar al microprocesador cuando accede a dispositivos lentos se emplea la línea READY del mismo. Explique como se la usa y como se generan los estados de "wait state".

v) Cuándo la señal READY esta activa, ¿en qué estado lógico se encuentra?

w) ¿Durante qué ciclo de reloj el microprocesador lee el estado de la señal READY?

x) Dibuje el diagrama en bloques del circuito que permita generar un wait state y analice su funcionamiento.

10.7 El Microprocesador 8088.

a) ¿Qué entiende por coprocesador?. ¿Es autónomo?

b) ¿Cómo funciona un coprocesador aritmético?

10.8 Modo mínimo, máximo, coprocesamiento y multiprocesamiento.

a) Caracterice al 8088 trabajando en modo mínimo.

b) ¿Cuándo es imprescindible que el 8088 trabaje en modo máximo?. ¿Cómo se encuentran las señales en este caso y que dispositivo se requiere?.

c) ¿Cómo selecciona un modo u otro de trabajo?.

d) ¿En que modo trabajan las PC?. ¿Por qué?.

e) Dibuje la configuración del 8088 trabajando en modo mínimo.

f) Dibuje la configuración del 8088 trabajando en modo máximo.

g) ¿Qué entiende por multiprocesamiento?. ¿Qué buses hay en este caso?.

10.9 Unidad de interfaz con el bus, unidad de ejecución y cola.

a) ¿Qué entiende por cola?. ¿Qué se busca optimizar con el empleo de la cola?.¿De qué tipo de memoria se trata?. ¿Qué sucede con la misma cuando se produce un salto?

b) ¿Qué es la BIU (Bus Interface Unit)?

c) ¿Qué es la EU (Execution Unit)?

10.10 Registros del microprocesador 8088.

a) ¿Cuáles son los registros de 16 bits de propósitos generales que se pueden utilizar como registros de 8 bits?. ¿Cómo se designan en ambos casos?. ¿Qué funciones cumplen?

b) ¿Qué registros de 16 bits de propósitos generales no divisibles en dos de 8 bits tiene el 8088?. ¿Cómo se designan? Para que se usan?

c) ¿Cuáles son los registros de segmentos?

d) ¿Cuántos bits tiene el registro de flags?. ¿Qué flags conoce?. ¿Para qué se los usa?.

e) ¿Qué par de registros se usa indefectiblemente para la búsqueda de un código de operación?

f) ¿Qué par de registros se usa indefectiblemente en toda operación con la pila?.

g) ¿Qué par de registros se usa indefectiblemente en toda operación en la que se apunte a una cadena (string) origen?.

h) ¿Qué par de registros se usan indefectiblemente en toda operación en la que se apunte a una cadena (string) destino?.

i) ¿Cuándo se usa el Segment Override? De un ejemplo usando cada una de las sintaxis posibles. ¿Cómo son los códigos de operación de ambas?.

j) ¿Qué es un prefijo?. ¿A cuántas instrucciones afecta?. ¿Modifica el código de operación?.

10.11 Modos de direccionamiento.

a) ¿Qué entiende por modo de direccionamiento directo?. De un ejemplo.

b) ¿Qué entiende por modo de direccionamiento inmediato? De un ejemplo.

c) ¿Qué entiende por modo de direccionamiento indirecto? De un ejemplo.

d) ¿Qué entiende por modo de direccionamiento registro? De un ejemplo.

e) ¿Qué entiende por modo de direccionamiento basado y con desplazamiento?. De un ejemplo.

f) Mediante un ejemplo explique cuando se usa el modo de direccionamiento basado e indexado con desplazamiento.

g) Explique como debe proceder para mover una cadena de bytes de un lugar a otro de la memoria. ¿Qué instrucciones permiten realizar la operación antes mencionada?. ¿Y de palabras?

10.12 Esquema de memoria.

a) ¿En que estado aparece el CS, ("Code Segment") cuando se resetea?. ¿Y el resto de los registros?. ¿Dónde se deberá hallar la primera instrucción que se ejecuta después del RESET?

b) ¿Qué tipo de memoria debe haber en la posición de memoria FFFFH?. ¿Por que?.

c) ¿Puede haber memoria EPROM en las direcciones bajas del microprocesador (vectores de interrupción)?. ¿Por qué?. ¿En que caso?. ¿Qué tipo de memoria se usa en la práctica en la parte baja?. ¿Por que?. ¿Cómo se deberán inicializar los vectores de interrupción?.

d) ¿Qué significa interceptar una interrupción?.

e) Describa qué es y que funciones cumple el circuito integrado controlador priorizado de interrupciones 8259 (PIC) en la PC.

f) ¿Por qué motivo el microprocesador tiene registros dinámicos?

g) ¿Qué característica tiene la parte baja y la mas alta del bus de direcciones del 8088?

h) ¿Para qué se usan los buffers 74HC244 y 74HC245?

i) ¿Qué funciones cumplen las líneas de control?.

10.13 Ensamblador y linker

a) ¿Cuántas letras permite el Assembler que tenga el nombre de variables y rótulos?

b) ¿Por qué convienen definir las directivas EQUs al principio del programa?

c) Indique las instrucciones en Assembler que permiten inicializar el registro de segmento de datos.

d) Indique la/s instruccion/es en Assembler que permiten leer la puerta 280H. ¿De qué tipo de direccionamiento se trata?.

e) Indique si son correctas las siguientes instrucciones en Assembler:

```
IN AL, 300H
IN AL, FFH      ¿Por qué?.
```

f) ¿Qué registro se usa como puntero a puertas?

g) Escriba la instrucción en Assembler que permite obtener el estado del bit más significativo de un byte que se encuentra en el registro AL sin destruir el contenido del mismo.

10.14 80188 y 80186.

a) ¿Qué ventajas tienen el 80188 y el 80186 con respecto al 8088? ¿Para qué ámbito fueron concebidos?. ¿Cuántos bits tiene el bus de datos de cada uno?.

b) El repertorio de instrucciones del 8088 ¿es compatible totalmente con el del 80186?. ¿Y al revés?.

c) Para igual velocidad de reloj ¿cuál microprocesador es más rápido, el 8088 ó el 80188?. ¿Porqué?.

d) ¿En que zona la memoria se hallará la memoria de programa?.

e) ¿Qué uso tienen los tres temporizadores del 80188?

f) ¿Qué función cumple el buffer 74LS244?. ¿Conque señal del 80188 se lo habilita?

g) ¿Qué característica tiene el buffer 74LS245?. ¿Sobre que bus se lo conecta?. ¿Con qué línea del 80188 se lo habilita?. ¿Para qué se usa la línea DT/*R del 80188?.

h) ¿Qué es el registro base?.

i) ¿Se puede programar la dirección de origen del primer "Chip Select" para puertas de entrada / salida?. ¿Qué ocurre con los demás "Chip Select"?.

j) ¿Qué es necesario programar para las memorias intermedias?.

k) ¿Qué se puede programar de los temporizadores accesibles para el usuario?.

l) ¿Cómo conecta memorias preparadas para trabajar con microprocesadores con un bus de datos de 8 bits cuando las quiere utilizar con microprocesadores que tienen un bus de datos de 16 bits?

m) ¿Qué significa que el 80186, el 8086 y el 80286 tengan la facilidad de trabajar con palabras alineadas y no alineadas?.

n) En el 8086 y en el 80186, ¿qué señal permite acceder a un byte que se encuentra en la parte alta del bus?.

o) ¿Qué diferencia hay entre acceder a una palabra alineada o a una no alineada?. ¿Cómo resulta para el programador?.

10.15 Repertorio de Instrucciones

10.15.1 Transferencia de datos.

Explique cual es el contenido de los registros y/o posiciones de memoria antes y después de ejecutarse cada una de las instrucciones siguientes, si es necesario explíquelas e indique cuales son erróneas o ambiguas:

- MOVAX, [2034]
- MOVAX, WORD PTR [2034]
- MOV[2034], 0
- MOV[SI], 2
- XCHG SI, DI
- XCHG WORD PTR [2034], AH
- XCHG WORD PTR [2034], AX
- PUSHA
- POPA

- IN AL, 23H
- MOVDX, 0123H
- OUT DX, AL
- OUT DX, AX
- IN AX, 23H
- MOVAL, ASCII
- XLAT TABLA

10.15.2 Transferencia de Direcciones.

Explique cual es el contenido de los registros y/o posiciones de memoria antes y después de ejecutarse cada una de las instrucciones siguientes, si es necesario explíquelas e indique cuales son erróneas o ambiguas:

- LDS BX, VAR1
- LES DI, VAR1
- SEGES (PARA EL TURBO ASSEMBLER)
- MOVAL, BYTE PTR [DI]
- LEA SI, [PEPE]

10.15.3 Instrucciones aritméticas.

Explique cual es el contenido de los registros y/o posiciones de memoria antes y después de ejecutarse cada una de las instrucciones siguientes, si es necesario explíquelas e indique cuales son erróneas o ambiguas:

10.15.3.1 Sustracción.

- SUB BX, WORD PTR [SI]
- SUB WORD PTR [SI], BX
- DEC BX
- DEC BYTE PTR [SI]
- CMP BX, WORD PTR [SI]
- NEG AL

10.15.3.2 Adición

- ADD AX, WORD PTR [BX]
- ADC AX, WORD PTR [BX]

10.15.3.3 Multiplicación y división.

- MUL BL ; Si los operandos son ambos de 8 bits, cuantos bits tiene
  ; el resultado de la multiplicación y en que registro queda?
- MUL DX ; Si los operandos son ambos de 16 bits, cuantos bits tiene
  ;el resultado de la multiplicación y en que registros queda?
- MUL EDX ; Si los operandos son ambos de 32 bits (80386), cuantos
  ; bits tienen el resultado de la multiplicación y en que registros queda?

Si el dividendo es de 32 bits, en que par de registros se lo carga en un 8088? En que registros quedan el cociente y el resto?

- DIV BL ; ¿Cuál registro es el dividendo y cual el divisor?

10.15.3.4 Aritmética decimal (BCD).

Explique la operación del siguiente programa:

```
MOV     AL, BYTE PTR DIG1
ADD     AL, BYTE PTR DIG2
DAA
MOV     BYTE PTR RESULT, AL
```

10.15.3.5 Instrucciones Lógicas.

```
AND     AX, DX
AND     AH, BYTE PTR MASCARA
TEST    BYTE PTR STATUS, 1
CMP     AX, WORD PTR TECLA
OR      AX, WORD PTR FUERZA_1
OR      DH, DL
NOT     BX
MASK_1  EQU  00010000B
AND     BX, NOT MASK_1
XOR     AL, AL
XOR     AX, MASK_1
XOR     BYTE PTR DATO, BL
```

10.15.3.6 Desplazamiento (shifts) y rotaciones.

```
SHL     AL, 4
SHR     AH, 2
SAR     AX, 1
SAL     DX, 4
ROR     WORD PTR VALOR, 2
ROL     CX, 7
RCR     BYTE PTR DATO, 3
RCL     AX, 3
```

10.15.3.7 Operaciones con cadenas (strings).

```
MOVSB
MOVSW
SCASB
CMPSB
INS     DX
OUTS    DX
```

10.15.3.8 Instrucciones de transferencia.

a) Explique la diferencia entre las siguientes instrucciones:

```
JMP     SHORT
JMP     NEAR
JMP     FAR
```

b) ¿Cuál es el tipo de salto por omisión?

c) Teniendo en cuenta los tipos de salto mencionados ¿Cómo clasificaría los saltos condicionales?

d) Modifique las instrucciones siguientes para el caso en que SIGUE se encuentre a una distancia mayor que un byte de la instrucción de salto:

```
JNC     SIGUE
---
SIGUE:
```

e) Explique las siguientes sentencias:

```
JMPDWORD [PEPE]
PEPE:   DW [OFFSET] SUBRU
        DW [SEGMENT] SUBRU
```

f) Defina un procedimiento NEAR llamado SUBRU1. ¿Cuál debe ser la última instrucción del mismo?. ¿Qué modificación debe hacer para que el procedimiento sea FAR?

g) ¿Qué instrucción permite llamar a un procedimiento NEAR? ¿Y a uno FAR?

h) ¿Cuál es el tipo por omisión? Escriba un ejemplo llamando a SUBRU1 que sea FAR.

i) Use las siguientes directivas e instrucciones del Assembler de manera que desde el programa principal se llame a un procedimiento NEAR, llamado NADA, que se encuentra en otro modulo y que este procedimiento a su vez salte a PEPE que es una dirección (rotulo) que se encuentra dentro de un procedimiento FAR correspondiente a otro modulo. Dibuje como bloques el programa principal y los módulos y escriba dentro de cada uno las instrucciones que correspondan:

- CALL
- EXTRN
- PUBLIC
- PROC
- JMP
- FAR
- NEAR

¿Qué precaución hay que tomar cuando se quiere definir variables accesibles por otros módulos?

10.16 Ejercicio Resumen

Se dispone de un microprocesador 8088 con una memoria de programa de 64 Kbytes y una de saltos de igual extensión.

Se desea implementar un sistema de control de temperatura. Para ello se conectará una termocupla con adecuado sistema de amplificación a un conversor A/D[20] de 8 bits de salida que se conectarán al bus de datos de un microprocesador.

Este conversor tiene una señal de entrada activa baja que ordena el inicio del proceso de conversión. La misma se conectará a una de las salidas de un decodificador 74HC138. En la misma dirección se podrá leer una palabra de 8 bits de estado en la que en el bit 2 se podrá observar cuando se culminó con el proceso de conversión.

[20] Dispositivo que convierte una magnitud analógica a una magnitud digital.

Una vez culminada la conversión se deberá leer una palabra de 8 bits en la que se encuentra en formato digital la tensión medida de la termocupla. Para ello se deberá producir la misma señal de activación del punto anterior pero con una señal de direccionamiento conectada a A0 del microprocesador en el estado 1.

La tensión medida se trasformará en temperatura a través de una tabla de conversión que se halla en la memoria de programa. La temperatura medida se comparará con una temperatura de referencia que se encontrará en la posición de memoria de datos TREF.

Si la temperatura medida es mayor que la de referencia se deberá excitar un ventilador cuya señal de activación será la línea 3 de una puerta de salida de 8 bits. Si la temperatura es menor que la de referencia, se deberá excitar un calefactor conectado (a través de la adecuada amplificación) a la pata 6 de la puerta de salida.

Se pide:

1. Bosquejar el circuito, demultiplexando los buses necesarios y utilizando buffers para el bus de datos y la parte no multiplexada dl bus de direcciones.
2. Desarrollar una decodificación empleando 74HC138 y admitiendo el empleo de la decodificación incompleta. Analizar la dirección de la memoria de programa y suponer que la memoria de datos comienza en la dirección 00000H.
3. Dibuje el diagrama de flujos que responda al sistema de adquisición de datos planteado.
4. Escriba el programa que responda al sistema de adquisición de datos planteado. Tenga en cuenta la necesidad de cambio del registro de segmento para apuntar a los distintos periféricos, a la memoria de datos, etc. Para la conversión de tensión a temperatura emplee la instrucción XLAT. Suponga que en caso de ser igual la temperatura medida a la de referencia, no se producirá ningún cambio en el estado del ventilador o el calefactor.
5. Suponga dos temperaturas de referencia contenida en dos posiciones de memoria de datos. La primera, TREF1 corresponde al límite superior por encima del cual actuará el ventilador y otra posición de memoria TREF2 que corresponde al límite inferior por debajo del cual actuará el calefactor.

Apéndice 1

Microprocesadores de más de 8 bits de palabra de datos

Al comienzo del capítulo se presentó la existencia de procesadores de más de 8 bits en el bus de datos.

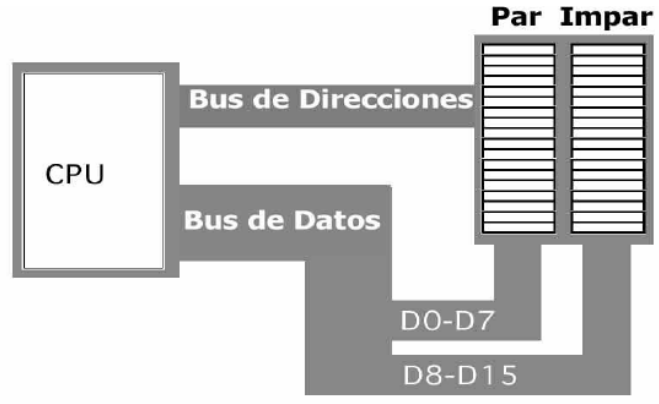


Fig. 36 Conexión de un microprocesador de 16 bits con memorias de 8 bits de ancho.

Como se observa en la Fig. 36, una memoria de 8 bits de ancho, está conectada sobre las líneas más significativas del bus de datos (IMPAR), mientras que otra memoria se conectaba sobre los bits menos significativas del bus de datos. La selección de la memoria se realizará con la línea A_0.

En el caso de que se desee realizar la lectura de una palabra de 16 bits sobre una dirección par, se accederá simultáneamente a los 16 bits y se realizará esa lectura en un único ciclo de máquina.

En cambio, si la palabra se hallaba en una dirección impar, se emplearán dos ciclos de máquina para la lectura, pues la palabra se encontrará en dos filas consecutivas.

Supongamos que se desea leer una palabra que se halla en la dirección 1H. El byte menos significativo (recordar la metodología de Intel de almacenar siempre primero el byte menos significativo) se hallará en la posición de memoria 1, mientras que el más significativo se encontrará en la posición 2. Ello significa que ha cambiado A_1 entre ambas posiciones y que ello obliga a cambiar la dirección entre una lectura y otra y por lo tanto no se pueden realizar en un único ciclo de máquina.

Si en cambio la palabra se encontrase “alineada” en una dirección par, A_{19} – A_1 se mantendrían constantes y por ello se podrían leer los 16 bits de daos en una sola vez.

Todo esto, pasa desapercibido para el programador pues es propio de la ejecución del programa. La única penalidad que se paga al tener variables de 16 bits 1ue comiencen en dirección impar es el mayor tiempo de ejecución de las instrucciones de lectura de 16 bits que se realicen.

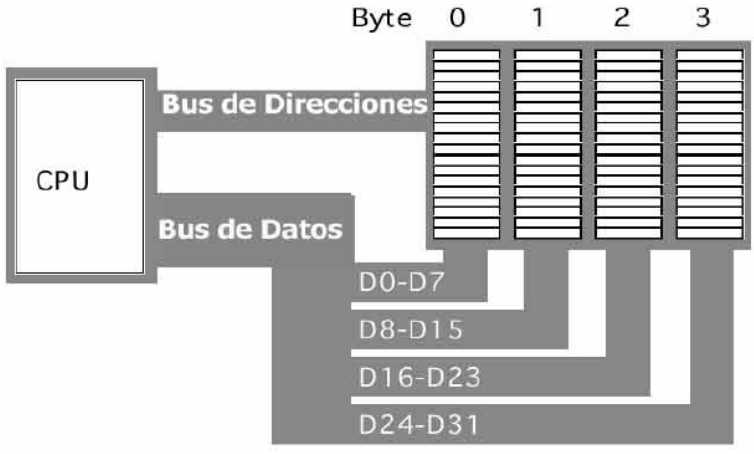


Fig. 37. Conexionado de una microprocesador de 32 bits con memorias de 8 bits.

Si el microprocesador dispone de 32 bits en el bus de datos (80386 y posteriores), podrá acceder a 32 bits en un solo ciclo de máquina siempre que la dirección de la variable sea MOD 4 (es decir que su cociente con cuatro, dé resto nulo). En caso contrario se necesitarán dos ciclos de máquina.

Aquí se emplearán las líneas A_0 y A_1 para seleccionar el byte de la palabra.