

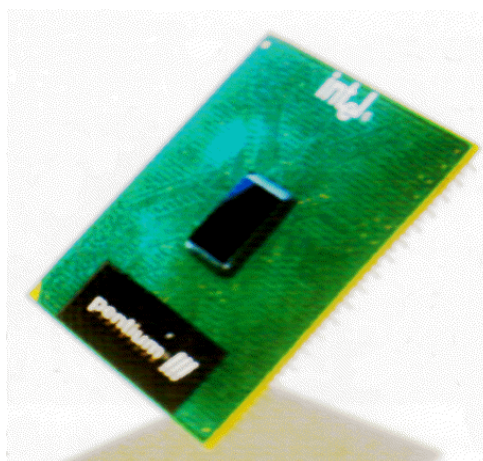
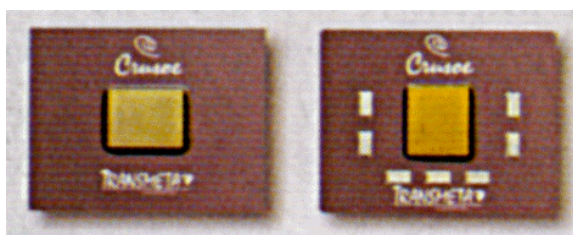


Universidad
de Huelva

TERCER CURSO. ELECTRÓNICA DIGITAL

Escuela Politécnica Superior
Universidad de Huelva

LA UNIDAD CENTRAL DE PROCESO



Manuel Sánchez Raya
Versión 1.0
8 de Marzo de 2000

INDICE GENERAL

| | |
|---|----|
| 1.- Entorno del sistema..... | 1 |
| 1.1.- Elementos de la UCP..... | 1 |
| 2.- Perfil del microprocesador..... | 2 |
| 2.1.- Señales externas..... | 2 |
| 2.2.- Funcionamiento del ?P..... | 5 |
| 3.- Arquitectura del microprocesador. | 8 |
| 3.1.- Registros internos. | 8 |
| 3.1.1.- Registros no accesibles | 9 |
| 3.1.2.- Registros accesibles | 10 |
| 3.2.- Unidad Lógica y Aritmética. | 13 |
| 3.3.- Unidad de control..... | 13 |
| 4.- Funcionamiento del microprocesador..... | 14 |
| 4.1.- La Pila. | 20 |
| 4.2.- Ciclos fundamentales..... | 20 |
| 4.2.1.- Funcionamiento interno..... | 20 |
| 4.2.2.- Funcionamiento externo. | 24 |
| 5.- Juego de Instrucciones..... | 26 |
| 5.1.- Estructura..... | 26 |
| 5.2.- Modos de direccionamiento..... | 27 |
| 5.3.- Clasificación de las instrucciones..... | 31 |
| 5.3.1.- Instrucciones de carga..... | 32 |
| 5.3.2.- Instrucciones lógicas..... | 33 |
| 5.3.3.- Instrucciones aritméticas..... | 35 |
| 5.3.4.- Instrucciones de salto..... | 36 |
| 5.3.5.- Instrucciones de control..... | 39 |
| 5.3.6.- Instrucciones de Entrada/Salida..... | 40 |
| 5.3.7.- Otras instrucciones..... | 40 |
| 6.- Interfaz con el bus del sistema..... | 41 |
| 7.- Interrupciones. | 43 |
| 7.1.- Tipos de interrupciones..... | 44 |
| 8.- El PC de IBM..... | 48 |
| 8.1.- La CPU 8086. | 48 |
| 8.2.- Generador de Reloj y control del bus. | 50 |
| 8.3.- Temporizador i8254 y modos de operación. | 51 |
| 8.3.1.- Conexión al sistema. | 51 |
| 8.3.2.- Inicialización..... | 53 |
| 8.4.- Controlador de interrupciones (PIC) i8259. | 54 |
| 8.5.- Controlador de Acceso directo a memoria (ADM) i8237. | 56 |
| 8.5.1.- Conexiones y operación..... | 56 |
| 8.5.3.- Inicialización..... | 58 |
| 8.5.4.- DMA en el IBM PC..... | 58 |

BIBLIOGRAFÍA

Apuntes Electrónica Industrial Escuela Politécnica Universidad de Málaga.
Microprocessors and Interfacing, Douglas V. Hall, Mc Graw-Hill.
Fundamentos de Computadores, De Miguel, P. Ed. Paraninfo.
Intel Data Sheets.

LA UNIDAD CENTRAL DE PROCESO

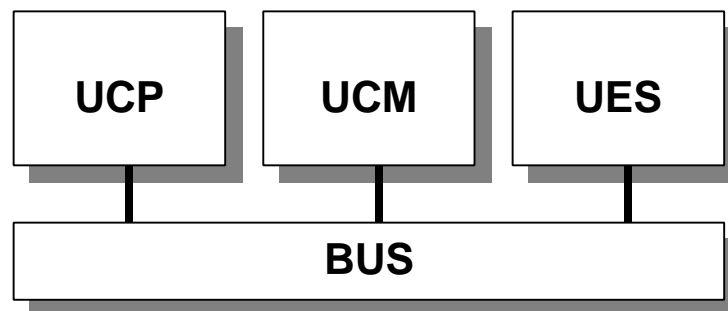
La Unidad Central de Proceso (UCP) es el corazón de los Sistemas Basados en Microprocesador. Esta unidad se encarga de coordinar y sincronizar el funcionamiento de todo el sistema. Sin embargo, por sí sola no forma el sistema, necesita de las demás unidades para su funcionamiento.

El Microprocesador (?P) es un dispositivo electrónico que reúne en un único elemento las partes fundamentales de una UCP. Por eso, UCP o ?P serán sinónimos en la mayoría de los casos.

Dependiendo del sistema, la UCP puede realizar más o menos funciones que las incluidas en un ?P, por eso utilizamos el término UCP para denominar la parte central de los Sistemas Basados en Microprocesadores, y el ?P en sí mismo se considera dentro de la UCP como un componente más. Otros componentes de la UCP como veremos serán: el *generador de reloj*, *controlador de interrupciones*(PIC), *temporizador* y *controlador de acceso directo a memoria*(ADM).

1.- Entorno del sistema.

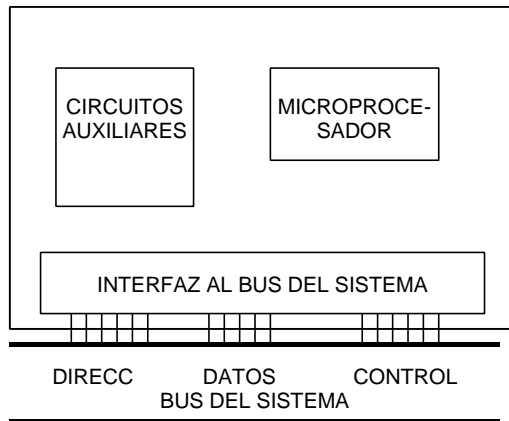
El estudio de la UCP que desarrollamos en lo que sigue está enfocado a una estructura del sistema mínima constituida por los 4 elementos básicos: BUS DEL SISTEMA, UCP, UCM y UES. La figura siguiente muestra el diagrama de bloques de este sistema en el que nos basaremos para el estudio teórico de la UCP.



Este estudio de la UCP no representa a ninguna UCP comercial en concreto sino que se trata de un modelo teórico del tema. Al final estudiaremos los componentes periféricos de INTEL que apoyan el funcionamiento de la UCP, ayudando al sistema para alcanzar otros fines que la UCP no puede o no tiene tiempo de abordar.

1.1.- Elementos de la UCP.

En la concepción actual de Sistema Basado en Microprocesador, la UCP se compone de dos elementos básicos: el Microprocesador y el interfaz al bus del sistema. La figura siguiente muestra los elementos de la UCP.



El Microprocesador

Este elemento es el que reúne la mayor parte de la funcionalidad global de la UCP en cuanto a proceso de la información, ejecución de programas, etc.. Sólo una “pequeña” parte de la UCP queda fuera de las posibilidades del μP . La funcionalidad del μP es lo más universal posible, cubriendo de esta manera la mayoría de las funciones de una UCP. Las funciones particulares o especiales que deseamos en una UCP las hemos de implementar por medio de una circuitería complementaria a la del μP . De esto se encargan los circuitos auxiliares.

En sistemas pequeños y medianos, el μP cubre la totalidad de las funciones exceptuando la adaptación a las características del bus del sistema. De esta función se encarga el interfaz al bus.

Interfaz al bus

Con el fin de poder acoplar la UCP al bus del sistema, la conectividad de esta unidad ha de estar de acuerdo con las especificaciones eléctricas y lógicas del bus. El interfaz al bus se ocupa de hacer posible la comunicación entre el μP y el bus del sistema.

Circuitos Auxiliares

Estos circuitos se encargan de añadir funcionalidad a la UCP que no puede prestar el μP , como por ejemplo realizar una gestión de las interrupciones (PIC) o realizar transferencias de E/S a memoria independientes (ADM), generar estados de espera en el bus para las memorias (generador de reloj) o generar señales para realizar el refresco de la memoria RAM o para seguir el curso de la hora y la fecha (generador de intervalos). El μP se apoya en dispositivos que realizan alguna función específica de las que hemos comentado. Estos dispositivos se conectan al bus y al μP , por lo que colaboran estrechamente con este último.

2.- Perfil del microprocesador.

En este apartado presentamos las ideas básicas de un modelo teórico de UCP desde el punto de vista externo para poder encajarla en el conjunto del sistema. En la figura siguiente se muestra la representación lógica de un μP , sobre la que nos basaremos para el desarrollo posterior.

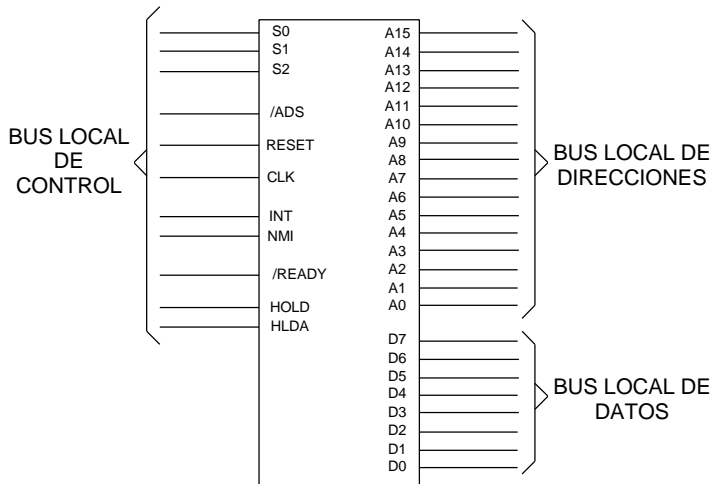
2.1.- Señales externas.

Un μP dispone de un conjunto de señales para su manejo desde el exterior. Estas señales se agrupan en tres conjuntos, o buses, cuya agrupación forma el bus local del μP ; el bus de datos, el bus de direcciones y el bus de control. En lo que sigue discutiremos cada uno de ellos.

Bus de datos.

El ?P dispone de un conjunto de señales por las cuales circula la información desde y hacia el ?P. Este conjunto de señales se le llama bus de datos cuyas características generales son las siguientes:

- Tamaño de 8 bits.
- Bidireccional.
- Triestado (para funcionamiento en ADM).



El bus de datos lo identificamos por el nombre genérico “D” y cada una de las señales que lo componen se identifican por su nombre formado por el nombre genérico del bus más un número que identifica su peso binario en la información. Así pues, el bus de datos de nuestro ?P, que es de 8 bits, se compone de 8 líneas de señal denominadas D7 a D0.

Bus de direcciones

Para poder localizar la información en la memoria, el ?P dispone de un conjunto de líneas de señal que se denomina *bus de direcciones* (A) y las señales que lo componen toman un nombre genérico del bus (A) más el número que identifica su *peso binario* en la información de dirección. Así pues, para nuestro modelo de ?P, el bus de direcciones que es de 16 bits, está compuesto por las señales A15 a A0.

El bus de direcciones presenta las siguientes características:

- ✍ **Tamaño de 16 bits.**
- ✍ **Unidireccional.**
- ✍ **Triestado (para ADM).**

Bus de control

El bus de control está formado por todas las señales que controlan el funcionamiento externo e interno del ?P. Al contrario de los buses de direcciones y datos, las líneas de señal que componen este bus no toman su nombre del genérico del bus sino que disponen de nombre propio que hace referencia a la función que realiza la señal.

Las líneas del bus de control se pueden asociar en tres grupos: las de *control del sistema*, las de *control del ?P* y las de *control del bus*.

Señales de control del sistema

Este grupo de señales son generadas por la Unidad de Control (UC) del μP y son las siguientes:

Líneas de estado (S2, S1, S0)

Las combinaciones de estas señales de salida del μP indican la situación (ciclo máquina) en que se encuentra. Con estas tres líneas de estado el μP identifica los 7 estados diferentes en que se puede encontrar. El μP también puede encontrarse en *estado de HOLD*, o sea con todos sus buses en triestado, pero esto lo indicaría con la señal HLDA a nivel "1". La codificación puede ser diferente para cada tipo de μP pero en general los ciclos son comunes. La codificación para nuestro μP es la indicada en la tabla siguiente:

| S2 | S1 | S0 | Estado |
|----|----|----|---------------------------|
| 0 | 0 | 0 | Lectura de código (fetch) |
| 0 | 0 | 1 | Lectura de memoria |
| 0 | 1 | 0 | Escritura en memoria |
| 0 | 1 | 1 | Lectura en E/S |
| 1 | 0 | 0 | Escritura en E/S |
| 1 | 0 | 1 | Rec. de interrupción |
| 1 | 1 | 0 | Halt |
| 1 | 1 | 1 | No asignado |

Por medio de las señales de estado podemos identificar la actividad del μP en cada instante. Estas señales son propias del μP y no son triestado.

Señales de control del μP

Por medio de estas señales podemos controlar el funcionamiento del μP desde el exterior de este.

Línea de inicialización (RESET) : La señal de RESET inicializa al μP a un estado conocido. Cuando RESET se activa el contador de programa se carga con un valor conocido (0 en nuestro μP) realizando el primer fetch (búsqueda de instrucción) sobre esta dirección.

Señal de reloj CLK : El reloj marca la pauta del desarrollo de todas las funciones, por lo tanto es uno de los factores que establecen la velocidad de ejecución del proceso.

Líneas de interrupción (INT y NMI) : Estas señales de interrupción nos permiten romper desde el exterior del μP la secuencia del programa que se está ejecutando en un instante dado para realizar otro programa (rutina de atención a la interrupción). Disponemos de dos tipos de interrupciones:

Interrupción enmascarable (INT) : La señal de entrada INT al μP es la encargada de realizar la interrupción enmascarable. Esta señal podrá ser atendida o no por el μP dependiendo de una información interna a este (máscara de interrupción) que pueda manejar el usuario desde el propio programa. Si la máscara está activada, la señal en INT no será tenida en

cuenta. Si la máscara de la interrupción está desactivada y se activa la señal INT, el μP abandona la secuencia del programa en la que se encuentra y pasa a realizar otro programa denominado de atención a la interrupción. De todas formas antes de saltar termina la instrucción en curso.

Interrupción no enmascarable (NMI) : El comportamiento es igual que en el caso anterior, con la diferencia de que para esta señal de entrada al μP no existe máscara alguna, de tal forma que cuando NMI se activa, el μP abandona la secuencia que está realizando para atender a la NMI (rutina de NMI).

Final de ciclo (/READY) : Esta señal de entrada al μP indica el final del ciclo en curso. Cuando el μP inicia un ciclo sobre el bus local, pone en él toda la información necesaria para realizar el ciclo. Una vez hecho esto, el μP espera que la unidad destino o fuente de la transferencia active la señal /READY para indicar que se ha realizado correctamente ésta. el μP iniciará la ejecución del siguiente ciclo máquina en cuanto muestree la señal /READY activada. /READY es activa a nivel bajo. En caso contrario esperará un ciclo de reloj más. Normalmente el mismo generador de reloj genera la señal READY insertando un número entero de estados de espera o ciclos de reloj de espera.

Señales de control del bus

Los demás elementos del sistema pueden solicitar al μP el control del bus. Se dice que son *elementos maestros del bus* (bus masters), estos solicitan al μP que desconecte sus líneas de direcciones, datos y control del bus del sistema. Con ello se permite que otro elemento del sistema sea quien controle el bus.

Para implementar esta función son necesarias dos señales, una de petición de bus llamada **HOLD** que es de entrada al μP y activa a nivel H, y otra de respuesta del μP que es salida y se llama **HLDA** (activa a nivel H). Cuando HLDA se activa, las líneas de los buses de direcciones, datos y control se encuentran en triestado por parte del μP . Cuando la señal HOLD se desactive, el μP tomará de nuevo el control del bus y desactivará la señal HLDA.

2.2.- Funcionamiento del μP .

El μP es una máquina de estados que trabaja bajo las órdenes de un programa almacenado previamente en la memoria del sistema (UCM). La memoria del sistema está dividida en dos partes fundamentales:

- a) La **memoria de programa**.
- b) La **memoria de datos**.

La memoria de programa contiene, convenientemente codificadas en binario, todas las órdenes (instrucciones) que ha de ejecutar el μP . Todas estas instrucciones se colocan ordenadas y consecutivas en la memoria de tal forma que si un programa comienza en la posición N de la memoria es porque la primera instrucción de éste está en esa posición, las siguientes instrucciones se encuentran en posiciones N+n siendo n un número entero positivo. El μP toma una a una cada instrucción y la ejecuta. El tiempo que tarda el μP en ejecutar una

instrucción se llama **CICLO DE INSTRUCCIÓN**. Este ciclo de instrucción se compone de un número entero de ciclos de reloj.

La ejecución de cualquier instrucción se compone de tres fases:

- Búsqueda del código de operación.
- Búsqueda del operando.
- Ejecución propiamente.

La fase a) se denomina “**fetch**” y siempre existe en toda instrucción. Por medio del código de operación le decimos al ?P la función que deseamos que realice (sumas, desplazamiento, etc.).

Durante la fase a) el ?P accede de lectura a la memoria de programa y obtiene el código de la instrucción. Se trata, pues, de un ciclo de lectura de memoria de programa.

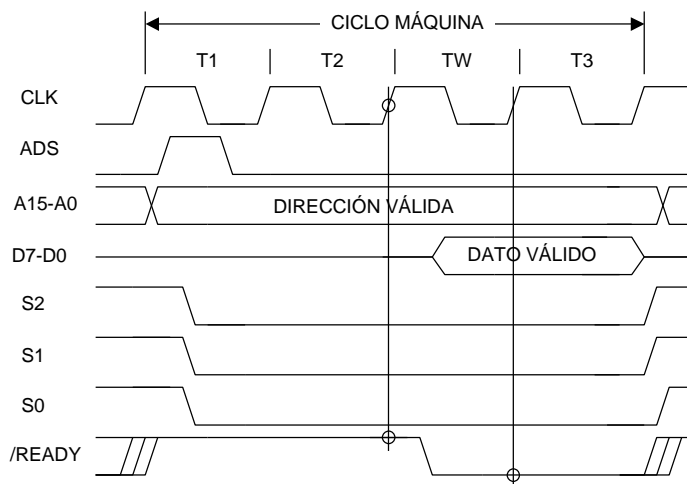
La fase b) es la de entrega del dato sobre el que se va a realizar el código de operación. Dependiendo del tipo de direccionamiento que se utilice para el código de operación (los distintos modos de direccionamiento los analizaremos más adelante), la fase b tiene distinto formato. Puede que no exista (cuando el direccionamiento es del tipo *implícito*) puede que en esta fase se obtenga el dato en sí mismo (direccionamiento *inmediato*) ó puede que en esta fase lo que se obtenga sea una referencia para poder localizar el dato en otra zona (direccionamientos *indirectos*).

La fase b), cuando existe, es uno o varios ciclos de lectura en la memoria de programa o de datos (según sea el direccionamiento).

La fase c) puede existir o no diferenciada de la a) y la b), pues depende de cuanto tiempo necesite el ?P para realizarla. En muchas ocasiones aparece embebida en las fases a) o b) porque el tiempo de ejecución es muy corto.

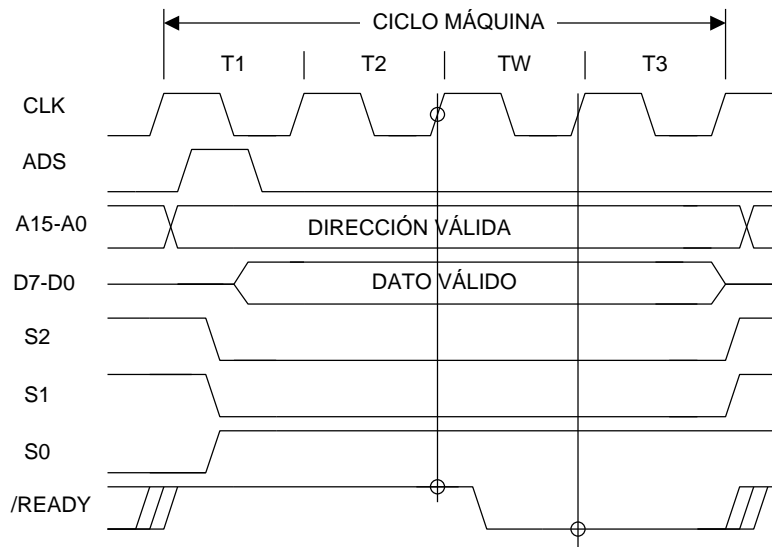
La fase c) se compone de ciclos que pueden ser de cualquier tipo sobre la memoria ó E/S (excepto fetch). Depende del tipo de instrucción que se esté ejecutando. En los apartados que siguen se describen cada uno de los ciclos.

Ciclo de búsqueda de código.



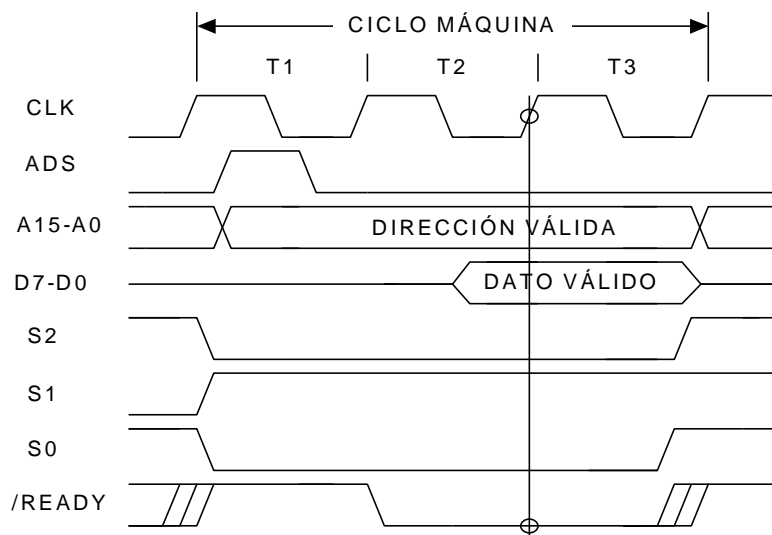
Podemos observar que las señales de estado del bus de control (S2, S1, S0) toman el valor correspondiente a la codificación del ciclo de fetch (ver tabla 1). Para acceder al código de operación, el ?P envía por el bus de direcciones la dirección donde se encuentra el código de operación. Las restantes señales quedan en estado de reposo.

Cuando la memoria entrega la información solicitada activa la línea /READY para indicar al ?P que la información existente en el bus de datos es válida. El ?P recoge la información del bus de datos y finaliza de este modo el ciclo.



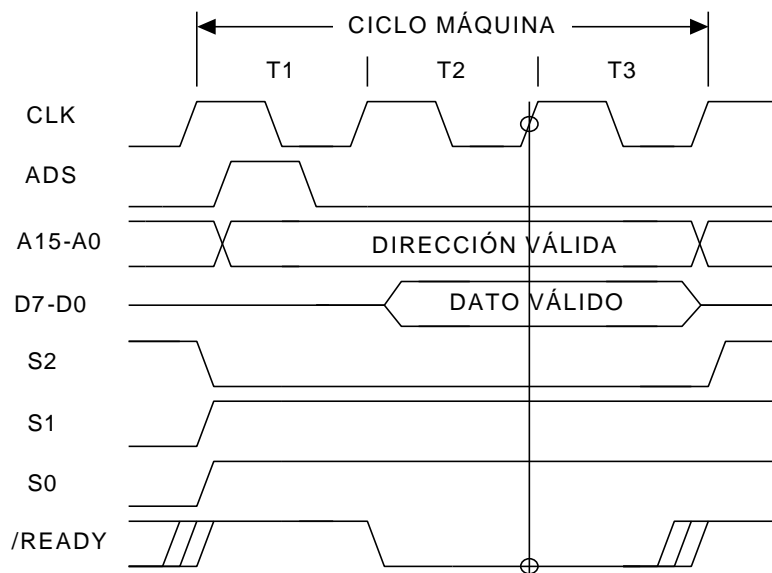
Ciclo de lectura en memoria.

La figura muestra un ciclo de lectura de la memoria (de programa o de datos) en donde las señales de estado nos dan la definición del ciclo en curso. El resto de la secuencia es similar a la del ciclo de fetch.



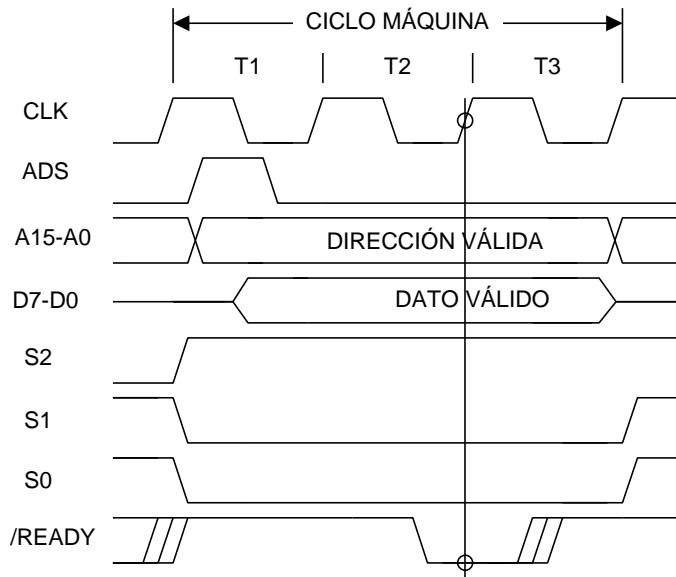
Ciclo de escritura en memoria.

Las señales de identificación del ciclo (S2, S1, S0) toman el estado correspondiente al ciclo en curso. El ?P envía por el bus de direcciones la dirección sobre la que se va a escribir. Cuando el ?P recibe la señal /READY activada procedente de la memoria, se acaba el ciclo.



Ciclo de lectura en E/S

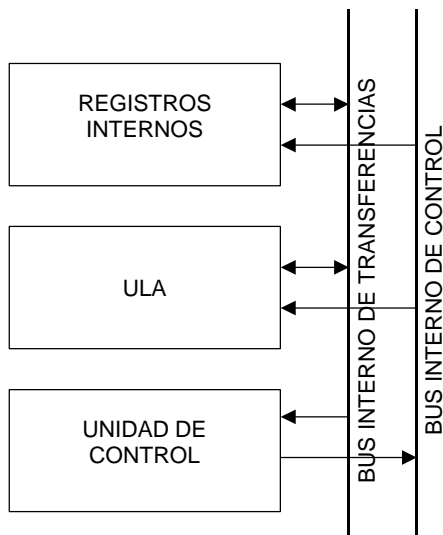
Las señales de estado del bus de control indican el destino del ciclo en curso mientras la dirección indica la posición en el espacio de E/S.



Ciclo de escritura en E/S.

Es similar al ciclo de lectura en memoria salvo que las señales de estado del bus de control tienen el estado correspondiente a este ciclo. La figura siguiente muestra el desarrollo del ciclo de escritura en E/S.

3.- Arquitectura del microprocesador.



Un ?P típico contiene tres unidades básicas:

- ?? **Registros internos.**
- ?? **Unidad Lógica y Aritmética.**
- ?? **Unidad de Control.**

Cada una de ellas se discute a continuación.

3.1.- Registros internos.

Se denominan “registros internos” al conjunto de registros que dispone el ?P para su funcionamiento. Los registros internos son unidades de almacenamiento temporal dentro del ?P. El tamaño de los registros depende de la función que ha de realizar y del tipo de ?P. En general tienen el tamaño del bus de datos o del bus de direcciones.

Clasificación

Los registros internos son de dos tipos:

- **No accesibles.**
- **Accesibles** por el usuario.

Los registros no accesibles son aquellos registros internos al μP , que el usuario no puede controlar directamente su funcionamiento ni su contenido. Son de uso propio y restringido del μP . Desde el punto de vista del usuario, estos registros son totalmente transparentes ya que no afectan, aparentemente, al funcionamiento del μP observado desde el exterior.

Los registros accesibles son aquellos registros internos al μP que el usuario puede controlar su funcionamiento y contenido desde el exterior del μP . Este manejo se hace a través del juego de instrucciones que dispone el μP .

Dentro de los registros de usuario se distinguen dos tipos diferentes, los de *uso específico* y los de *uso general*. Los de uso específico son aquellos cuya función está determinada por el fabricante del μP (como por ejemplo el contador de programa). Los de uso general no tienen una función predeterminada sino que, por el contrario, es el usuario quien lo aplica según sean sus necesidades.

3.1.1.- Registros no accesibles

Aunque desde el punto de vista del usuario, la función de éstos registros es totalmente transparente y, además no se dispone de medios para actuar directamente sobre ellos, sí es interesante conocer la existencia de alguno de ellos ya que su presencia nos facilitará la comprensión del funcionamiento del conjunto.

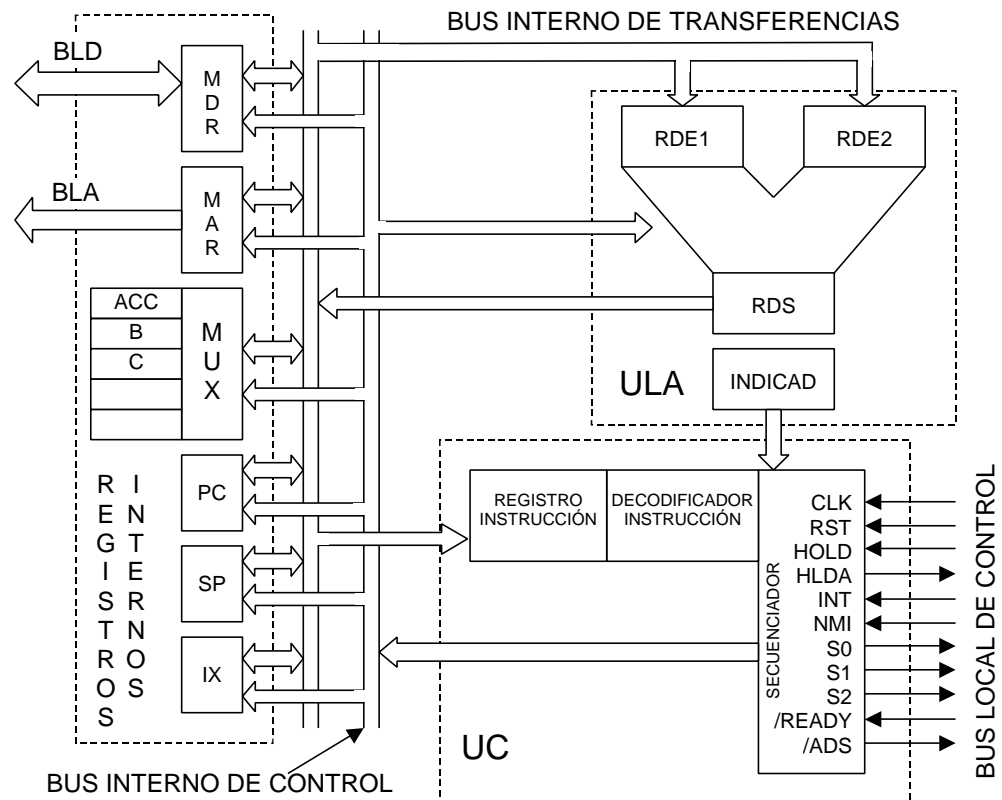
Registro de instrucción : Las instrucciones que ha de ejecutar el μP se encuentran codificadas como un dato en la memoria, de donde las extrae para su ejecución. Las instrucciones se componen de varias partes, el *código de operación* que es realmente la orden codificada, y el *operando*. Cuando el μP obtiene de la memoria el código de operación de una instrucción lo guarda en un registro denominado registro de instrucción y se mantiene ahí hasta finalizar la ejecución de esa instrucción.

El registro de instrucción es del tamaño adecuado a los códigos utilizados en el μP . Lo más frecuente es que los códigos tengan el tamaño igual al dato (8 o 16 bits).

La salida del registro de instrucción sirve como entrada al *decodificador de instrucción*, pero esto lo veremos más adelante al hacer la descripción del funcionamiento. Este registro no es modificable directamente por el usuario ya que no dispone de una instrucción para ello.

Registro del bus de direcciones : El bus de direcciones del μP está formado por las salidas de un registro interno al μP que se denomina registro del bus de direcciones (MAR). Este registro tiene como misión mantener una determinada información de dirección sobre el bus de direcciones, independizándolo de la evolución de los restantes registros internos al μP .

Las fuentes que entregan direcciones al registro del bus de direcciones son varias. Una de ellas es el registro *Contador de Programa* (PC) que ya hemos citado y del que hablaremos con mayor detalle más adelante. Este registro es el que más frecuentemente utiliza el MAR ya que cada instrucción a ser ejecutada necesita ser buscada previamente a través de su dirección contenida en el PC. El *registro puntero de la pila* también accede al MAR cuando su contenido se utiliza para dar una dirección. Así mismo, el MAR puede ser cargado desde cualquier otro registro interno capaz de ser puntero de dirección. El MAR es cargado directamente desde una determinada instrucción si ésta contiene una *dirección de dato*.



El MAR es un registro unidireccional de entrada y salida en paralelo. Si el ?P puede funcionar en ADM, este registro además dispone de salidas triestado.

Registro del bus de datos : Es similar al registro del bus de direcciones salvo que ahora este registro se encarga de conectar con el bus de datos. Por él pasan todos los datos y códigos solicitados y entregados por el ?P. Este registro se llama MDR y es de *entrada y salida paralelo, bidireccional* y si el ?P permite el funcionamiento en ADM, la salida hacia el bus de datos externo ha de ser *triestado*.

3.1.2.- Registros accesibles

Como hemos visto antes, el ?P dispone de un conjunto de registro que son accesibles al usuario. Dentro de ellos hay dos tipos, los de *uso específico* y los de *uso general*. A estos últimos se denominan también *registros de usuario*.

Registros de uso específico

Los registros internos de uso específico son aquellos que tienen asignada una función determinada dentro del ?P, aunque en algún caso puedan realizar otras funciones.

Registro contador de programa : Las instrucciones que constituyen un programa se almacenan en la memoria en direcciones consecutivas. El ?P para poder ejecutar este programa ha de acceder a la memoria de programa a leer estas instrucciones para saber qué operación ha de ejecutar. Para ello es necesario direccionar adecuadamente la memoria.

Cada posición de la memoria está numerada unívocamente para poder distinguirla de las restantes. El número que identifica cada posición de la memoria se llama dirección. El ?P dispone de un registro destinado a contener la dirección de la instrucción a ser ejecutada. Este registro es el contador de programa (PC). El ?P avanza en la ejecución del programa incrementando este registro cada vez que ejecuta una instrucción.

El programador almacena los programas en posiciones sucesivas y crecientes de la memoria; según esto, el comienzo del programa estará en posiciones más bajas, y el final en las más altas.

Esto no es cierto rigurosamente en todos los casos ya que no siempre es posible escribir los programas en direcciones consecutivas. En multitud de ocasiones hemos de cambiar el orden de ejecución del programa debiéndose ejecutar instrucciones que se encuentran en posiciones no consecutivas. Esto no es un problema, ya que se dispone de instrucciones de “salto” que permiten alterar el contenido del contador de programa al valor necesario.

Como vemos, el PC es un registro con una función específica pero cuyo contenido lo podemos controlar a través de las instrucciones correspondientes.

Registro puntero de la pila : El Stack ó pila es una zona de la memoria de datos reservada al ?P y que ésta destinada a almacenar determinada información de importancia. Más adelante al describir el funcionamiento de la pila veremos en detalle la utilización de este registro.

Al ser la pila una zona de la memoria, el ?P ha de direccionarla a través de un registro especial denominado *puntero de la pila (stack pointer, SP)*. El SP es capaz de volcar su contenido directamente sobre el registro MAR. El registro puntero de la pila (SP) tiene una dimensión determinada que obliga a fijar la posición y el tamaño de la pila en la memoria. Cada vez que el ?P necesita escribir o leer un dato de la pila, vuelca al bus de direcciones (a través del MAR) el contenido del puntero de la pila.

El ?P maneja de forma *automática* el puntero de pila en determinadas circunstancias, como veremos más adelante, pero también puede ser modificado a voluntad por el usuario a través de las instrucciones correspondientes. Este registro puede ser *cargado directamente* con una información o puede ser incrementado o decrementado.

Registro índice : Este es un registro con una utilización determinada, pero que en muchas ocasiones puede ser utilizado como un registro general.

El registro índice tiene como misión *servir de puntero de direcciones* para unas determinadas funciones. El ?P utiliza el contenido de este registro para suministrar o calcular la dirección de algún dato cuando la instrucción así lo ordene (direccionamiento indexado).

Registro acumulador : Este registro es el registro destino de la *salida de la ULA* y además puede ser cargado con cualquier valor a través de las instrucciones correspondientes. Se encuentra, pues, en la línea divisoria entre los registros de uso específicos y los de uso general.

Tiene el mismo tamaño que el dato del ?P, y puede ser utilizado como origen o destino de datos o direcciones (teniendo siempre en cuenta su tamaño). Es uno de los registros más utilizados del ?P.

Registro de indicadores : Los indicadores son biestables que utiliza el ?P para memorizar determinadas *condiciones* ocurridas durante un proceso. Estos indicadores se agrupan en un *registro de indicadores (o registro de estado de la máquina, PSW Program Status Word)*.

Este registro de estado tiene un tamaño variable de un ?P a otro, dependiendo de las necesidades previstas por el diseñador del mismo. Es un registro algo particular en cuanto a que cada uno de los bits es independiente de los demás y el ?P o el usuario puede activarlos o desactivarlos por separado ya sea directa o indirectamente.

El ?P actualiza el estado de estos indicadores automáticamente como consecuencia del resultado de la instrucción ejecutada, representando sobre ellos el estado en el que ha quedado la máquina. La figura siguiente muestra el contenido del registro de indicadores del ?P.



Cada uno de estos bits recibe un nombre propio y tienen una función diferente. Lo más general es incluir en este registro los siguientes bits:

- **INDICADOR DE CERO (Z).** Se activa cuando el resultado de la operación es nulo. En caso contrario, se desactiva.
- **INDICADOR DE ACARREO (C).** Se activa cuando se produce un acarreo en la última operación realizada. Se desactiva si no se produce. En las operaciones de resta, representa al acarreo correspondiente (borrow).
- **INDICADOR DE SIGNO (S).** Refleja el estado del bit 7 del último resultado de la última operación hecha sobre la ULA. Está pensado para ser utilizado con la representación de magnitud y signo. Si S=1 indica que el resultado es negativo.
- **INDICADOR DE REBOSAMIENTO (V).** Señaliza un error en el proceso de cálculo en complemento a dos por sobrepasar el valor máximo en el resultado de la operación. (+127, -128).
- **INDICADOR DE ARRASTRE INTERMEDIO (H).** Señaliza el arrastre producido del bit 3 al 4 como resultado de la operación. Este indicador está pensado para ser utilizado en operaciones con representación de números en BCD.
- **INDICADOR DE SUBTRACCIÓN (N).** Ya que el algoritmo para corregir operaciones en BCD es diferente para las operaciones de suma y resta, este indicador especifica cual fue la última operación ejecutada y así permitir corrección automática.
- **MÁSCARA DE INTERRUPCIÓN (I).** Este bit permite la entrada o no de una interrupción enmascarable al ?P. Si la máscara está activada, I="1", el ?P no hace caso de la señal de entrada de interrupción enmascarable INT. La interrupción está enmascarada. Si I="0", el ?P aceptará la entrada de interrupción por la señal INT.

Hay que hacer notar que el estado activo de los bits del registro de estado es el estado H (nivel alto), mientras que el reposo lo es el nivel L (bajo). Hemos dicho antes que estos bits los puede activar y desactivar el usuario. Esto es así en algunos casos, pero en otros no existe una instrucción directa que lo realiza sino que se ha de hacer a través de otra instrucción, esto es, de una *forma indirecta* (p. e.: si deseamos activar el indicador Z, podemos hacer una operación de resta del acumulador con él mismo. El resultado es 0 y Z=1).

Registros de uso general : Estos son registros internos del ?P que no tienen una función predefinida. Se pueden utilizar como origen o destino de los datos en el ?P y pueden intervenir en algunas operaciones aritméticas y lógicas de la ULA.

Los registros de uso general se utilizan como *elementos de memoria* dentro del ?P con un tiempo de acceso muy corto. Esto hace que los fabricantes introduzcan el máximo número posible de estos ya que así se aumenta enormemente la velocidad de proceso. El tamaño de los registros internos suele ser el mismo que el del dato del ?P o un múltiplo o submúltiplo entero. En muchas ocasiones los registros de uso general tienen también funciones de puntero de direcciones para determinadas operaciones sobre la memoria.

3.2.- Unidad Lógica y Aritmética.

La ULA es la parte del ?P que se encarga de realizar las *operaciones lógicas y aritméticas*. Las operaciones aritméticas que realiza son las dos básicas: suma y resta. Las operaciones lógicas que es capaz de realizar son: AND, OR, inversión, XOR, desplazamientos y rotaciones.

Con este conjunto de *operaciones básicas* es posible realizar cualquier función por compleja que sea. El problema se traslada a un algoritmo que lo realice con estas operaciones básicas. Los ?Ps más avanzados incorporan funciones más complejas, como multiplicación y división, que simplifican el algoritmo. Además, existe la posibilidad (en algunos ?Ps) de utilizar un *coprocesador aritmético* que se encarga de realizar todas las funciones de la ULA con una velocidad mayor ya que permite trabajar en *punto flotante*.

3.3.- Unidad de control

La Unidad de Control (UC) dirige el funcionamiento de la ULA, de los registros internos y genera las señales del bus de control externo del ?P de acuerdo con la instrucción que esté en ejecución en cada momento.

La UC reacciona frente al *código de operación* de la instrucción que se ha cargado en el registro de instrucción. El código pasa al decodificador de instrucción que lo identifica y activa las señales internas y externas correspondientes del ?P para realizar la instrucción.

La UC es un *sistema secuencial* controlado por el reloj del ?P que arranca su funcionamiento a partir de la *decodificación del código de operación* cargado en el registro de instrucción generando las señales necesarias para ejecutar la instrucción en curso. El ciclo de funcionamiento de la UC no es fijo puesto que depende del código cargado. El comienzo del ciclo es siempre la lectura de la memoria de programa del código de operación que define la instrucción a ejecutar. Durante la ejecución de toda instrucción, la UC prepara la dirección del siguiente código en el contador de programa (PC).

La UC realiza su secuencia de acuerdo a un *microprograma* que hay grabado en una memoria fija en su interior y que es capaz de identificar (decodificar) todo el juego de instrucciones del ?P. Algunos ?P permiten que sea el propio usuario quien escriba esta memoria del microprograma. Esta posibilidad permite hacer un ?P personalizado.

4.- Funcionamiento del microprocesador.

En este apartado discutiremos el funcionamiento del μP desde dos perspectivas distintas; desde el interior del μP y desde el exterior. La visión interna del funcionamiento es básica para un buen entendimiento del comportamiento externo.

Funcionamiento interno

Nos basaremos en la arquitectura del μP representada por la figura anterior de la estructura interna de la UCP, en la que podemos distinguir los tres elementos principales; los *registros internos*, la *Unidad Lógica y Aritmética* y la *Unidad de Control*.

El funcionamiento de los tres elementos básicos citados son coordinados por la UC. El funcionamiento del μP es siempre cíclico y controlado por el reloj que marca la pauta para que la máquina de estados de la UC evolucione de un estado al siguiente. Es, por tanto, un **sistema secuencial síncrono**. Esta característica no impide el funcionamiento con procesos totalmente asíncronos respecto al μP , como veremos más adelante cuando hablemos de las interrupciones.

Además, la UC es capaz de detener la secuencia (congelarla) en determinados puntos de ella para adaptarse a las distintas velocidades del resto de los elementos que conforman el sistema. Esto hace que el μP se pueda compatibilizar con cualquier otro elemento (memoria, dispositivo E/S, etc.) aunque éstos no sean síncronos con el reloj de aquella, gracias a la señal /READY.

Descripción de las señales de control internas del μP

En los apartados que siguen establecemos la definición de las distintas señales del bus de control interno que genera la UC del μP . Para ello establecemos, en primer lugar, la nomenclatura que utilizamos y a continuación hacemos la descripción detallada.

Nomenclatura genérica : Las señales de control internas al μP utilizan una **nomenclatura particular** que está relacionada con la funcionalidad de cada una de ellas. Desde un punto de vista general, esta nomenclatura consta de dos campos separados por un guión bajo (_).




El **primer campo** indica el tipo de señal y el nivel lógico activo (o flanco activo). El símbolo "/" indica que la señal es activa a nivel bajo (o el flanco activo es el flanco de bajada). Si no se pone este símbolo, la señal es activa a nivel alto (o el flanco activo es el flanco de subida).

El **segundo campo** da información del usuario de la señal indicando el nombre (reducido) del destinatario de la misma. Cada nombre de señal consta de ocho símbolos en total, incluyendo el símbolo "/" (si existe) y el guión bajo "_". Por ejemplo, la señal: /OE_PC indica que su nombre es OE, que es activa a nivel bajo y que el destinatario es el PC. En reposo se encuentra a nivel H.

Descripción de las señales




En lo que sigue haremos una descripción detallada de cada una de las señales necesarias para el control, por parte de la UC del ?P, de los distintos elementos.

Señales de control de salida : Cada elemento que es capaz de suministrar información al bus interno de datos o al bus local de direcciones o de datos del ?P dispone de un control para permitir su conexión (y desconexión) al bus correspondiente. Esta señal se denomina genéricamente /OE_XXXX (Output Enable) en donde XXXX representa el nombre del destinatario. Son **activas a nivel bajo** de tal forma que en reposo se encuentran a nivel H manteniendo al elemento activo desconectado del correspondiente bus. Por ejemplo:

| | | |
|---------|---|---|
| /OE_MAR |  | Control de salida del registro MAR. |
| /OE_PC |  | Control de salida del contador de programa. |
| /OE_RDS |  | Control de salida del registro RDS de la ULA. |

Señales de escritura : Casi todos los registros internos del ?P pueden ser cargados (escritos) desde el bus de datos interno con una información. El MAR además puede ser cargado desde el bus de datos local del ?P. Para poder realizar esta maniobra es necesario disponer de una señal que realice la escritura en los diferentes registros. Genéricamente esta señal se denomina WR_XXXX. Es **activa con su flanco ascendente** por lo que en reposo se encuentra en nivel H o L (es más interesante mantener el nivel H como la situación de reposo ya que representa un mayor margen al ruido. El flanco ascendente en este caso se consigue por medio de la generación de un pulso a nivel L).

Por ejemplo:

| | | |
|--------|---|--|
| WR_MDR |  | Escritura en el registro de datos del bus. |
| WR_SP |  | Escritura en el puntero de pila. |
| WR_INS |  | Escritura en el registro de instrucción. |

Controles de los registros contadores : Algunos de los registros internos del ?P tienen características especiales. Además de poder ser escritos y leídos en paralelo pueden ser **incrementados o decrementados** como contadores. Estos registros son el PC y el SP.

El **contador de programa (PC)** actúa normalmente como contador incrementándose en una unidad con una señal de reloj denominada CK_PC que es activa con el flanco de subida, por lo que normalmente se encuentra en estado “0”. Cada vez que el contenido del PC es sacado al bus de direcciones para acceder a esa dirección de memoria, el PC se ve incrementado en una unidad. De esta forma, el PC siempre contiene la siguiente dirección que ha de ser accedida.

El PC, por otro lado, también puede ser **cargado con una información** de dirección desde el bus interno (para realizar un cambio en la secuencia del programa, saltos), por ello dispone de la señal WR_PC que realiza esta función de carga.

El registro puntero de pila SP trabaja normalmente como **contador arriba-abajo**, por lo que necesita dos señales para su control. La señal /INC_SP indica al SP (cuando se activa a nivel L) que actúe como un contador hacia arriba. Cuando /INC_SP está a nivel H, el SP

trabaja como contador hacia abajo. El reloj para incrementar o decrementar el SP se denomina CK_SP y es activo por flanco de subida.

El SP puede ser cargado con una información de dirección desde el bus interno del ?P (por ejemplo, para cambiar la situación de la pila en la memoria), por lo que dispone de una señal WR_SP que realiza esta función con el flanco ascendente.

Señales de control de los registros de uso general : Los registros de uso general se encuentran agrupados bajo el control de un multiplexor. La UC controla las señales de este MUX para seleccionar y escribir o leer uno de los registros asociados en cada ocasión.

Para la *selección de uno de los registros*, disponemos de tres señales de control SEL_MUX2, SEL_MUX1 y SEL_MUX0 cuyas combinaciones direccionan uno de los ocho registros disponibles como se indica en la tabla siguiente:

| SEL_MUX2 | SEL_MUX1 | SEL_MUX0 | Registro seleccionado |
|----------|----------|----------|-----------------------|
| 0 | 0 | 0 | A |
| 0 | 0 | 1 | B |
| 0 | 1 | 0 | C |
| 0 | 1 | 1 | D |
| 1 | 0 | 0 | E |
| 1 | 0 | 1 | F |
| 1 | 1 | 0 | REG1 |
| 1 | 1 | 1 | REG2 |

La señal ***/OE_MUX es el control de salida*** del registro seleccionado por el multiplexor. Es activa a nivel L. Cuando /OE_MUX=L la información contenida en el registro seleccionado por la combinación de las señales de control del MUX es volcado al bus interno.

Los registros A a F son registros accesibles de uso general, los REG1 y 2 son registros de almacenamiento temporal no accesibles para el funcionamiento de la UCP con algunas instrucciones. Estos registros de 8 bits se conectan al bus interno sobre la parte baja y parte alta de forma que se pueda volcar su contenido tanto a la parte baja del bus interno (mediante /OE_MUX_L) o a la parte alta del bus interno (mediante /OE_MUX_H).

La señal ***WR_MUX es el control de escritura*** del registro seleccionado por el multiplexor. Es activa con flanco ascendente.

Señales de control de la ULA : La Unidad Lógica y Aritmética realiza diversas funciones que han de ser seleccionadas cada una de ellas en cada instante por la UC en función de la instrucción a ejecutar. Las *señales que seleccionan estas funciones* son cuatro y se denominan SEL_ULA3, SEL_ULA2, SEL_ULA1 y SEL_ULA0.

Las distintas combinaciones de estas señales definen la función de la ULA como se indica en la tabla que sigue:

| S E L 3 | S E L 2 | S E L 1 | S E L 0 | Función | Exp. |
|------------------|------------------|------------------|------------------|------------------|--|
| 0 | 0 | 0 | 0 | SUMA | $(RDE1 + RDE2)$ |
| 0 | 0 | 0 | 1 | RESTA | $(RDE1 - RDE2)$ |
| 0 | 0 | 1 | 0 | INCREM. | $(RDE1 + 1)$ |
| 0 | 0 | 1 | 1 | DECREM. | $(RDE1 - 1)$ |
| 0 | 1 | 0 | 0 | OR | $(RDE1 \# RDE2)$ |
| 0 | 1 | 0 | 1 | AND | $(RDE1 \& RDE2)$ |
| 0 | 1 | 1 | 0 | XOR | $(RDE1 \oplus RDE2)$ |
| 0 | 1 | 1 | 1 | INVERSION | $(/RDE1)$ |
| 1 | 0 | 0 | 0 | ROT. LOG. IZQ. | La rotación lógica a izq. consiste en desplazar cada bit n del registro RDE1 a la posición n+1. El contenido original del bit 7 pasa al bit 0. |
| 1 | 0 | 0 | 1 | ROT. LOG. DER. | La rotación lógica a der. consiste en desplazar cada bit n del registro RDE1 a la posición n-1. El contenido original del bit 0 pasa al bit 7. |
| 1 | 0 | 1 | 0 | ROT. ARIT. IZQ. | La rotación aritmética a la izq. consiste en desplazar el bit n de RDE1 a la posición n+1. El bit 7 original se coloca en el bit de CARRY del registro de estado y el contenido original del CARRY pasa a la posición 0. |
| 1 | 0 | 1 | 1 | ROT ARIT. DER. | La rotación aritmética a la der. consiste en desplazar el bit n de RDE1 a la posición n-1. El bit 0 original se coloca en el bit de CARRY del registro de estado y el contenido original del CARRY pasa a la posición 7. |
| 1 | 1 | 0 | 0 | DESP. LOG. IZQ. | El desplazamiento lógico es similar a la rotación lógica sin la realimentación entre el bit 0 y el 7. |
| 1 | 1 | 0 | 1 | DESP. LOG. DER. | El desplazamiento lógico es similar a la rotación lógica sin la realimentación entre el bit 0 y el 7. |
| 1 | 1 | 1 | 0 | DESP. ARIT. IZQ. | El desplazamiento aritmético es similar a la rotación aritmética sin la realimentación entre CARRY y bit 0. |
| 1 | 1 | 1 | 1 | DESP. ARIT. DER. | El desplazamiento aritmético es similar a la rotación aritmética sin la realimentación entre CARRY y bit 0. |

REGISTROS DE ENTRADA DE LA ULA : Los dos registros de entrada de la ULA (RDE1 y RDE2) son registros de entrada y salida en paralelo de 8 bits de tamaño. Sólo disponen de señales de control para la *carga de la información* (**WR_RDE1** y **WR_RDE2**) ya que su salida se encuentra conectada directamente a la entrada de la ULA. Cuando WR_RDEn cambia de estado L \rightarrow H la información existente en el bus interno se escribe en el registro que corresponda de entrada a la ULA.

BUFFER DE SALIDA DE LA ULA : La salida de la ULA es un buffer (registro) que dispone únicamente de *control de salida* (**/OE_BDS**) ya que mientras las señales de entrada sean registradas, la salida se mantiene estable y, por tanto, no necesita ser registrada. Cuando /OE_BDS=L el resultado de salida de la ULA pasa al bus interno.

Señales de control de los registros del interfaz : Denominamos registro de interfaz a los registros del ?P que permiten la *comunicación con el exterior* de ésta. Son dos los registros de interfaz; el registro MDR que conecta el ?P con el bus de datos local y es por donde circula la información de datos y códigos hacia y desde el ?P. El otro registro de interfaz es el MAR que permite al ?P conectarse con el bus de direcciones local. Por medio del registro MAR, el ?P indica al resto de los elementos del sistema la dirección a la que desea acceder (de lectura o escritura) ya sea de memoria o de entrada/salida.

REGISTRO DEL BUS DE DATOS MAR : El MDR es un registro de entrada y salida en paralelo con características particulares; es *registrado* en el sentido bus interno \nless bus local y es *transparente* (no registrado) en sentido bus local \nless bus interno. Una señal de escritura (WR_MDR) permite la carga de este registro desde el bus interno del ?P. Cuando WR_MDR sube, la información existente en ese instante en el bus interno del ?P se introduce en el registro MDR (los 8 bits más bajos, 7 a 0 ya que se trata de un registro de 8 bits).

Para poder ser bidireccional, la salida ha de poder ser seleccionada en función de una señal que indica el *sentido de la información* para poder hacer las transacciones entre ambos buses (del bus local al bus interno o del bus interno al bus local). Estas características hacen que el registro MDR disponga de una *señal para el control del triestado* (/OE_MDR), una señal para el *control de la dirección del dato* (DIR_MDR, hacia BDI o hacia BD). Cuando DIR_MDR=H el registro es transparente y la información fluye desde el bus local hacia el bus interno del ?P. Si DIR_MDR=L el registro es registrado y la información fluye desde el bus interno hacia el bus local del ?P.

La señal /OE_MDR controla la salida en ambos sentidos del flujo de la información sobre el bus que corresponda (interno o local). Cuando /OE_MDR=L y DIR_MDR=L, el contenido del MDR es volcado al bus local del ?P. Cuando /OE_MDR=L y DIR_MDR=H, el contenido del bus local es transferido al bus interno del ?P.

REGISTRO DEL BUS DE DIRECCIONES : El conjunto de señales de control que gobiernan el funcionamiento del registro MAR son consecuencia de su actividad dentro del ?P. Este registro es de entrada y salida paralelo (entrada desde el bus interno y salida al bus local del ?P) y es unidireccional. Sólo son necesarias dos señales de control; el *reloj de carga* (WR_MAR) y el *control del triestado* de salida hacia el bus de direcciones local (/OE_MAR).

Cuando la señal WR_MAR sube de L a H, el contenido del bus interno del ?P se introduce en el registro MAR. Cuando /OE_MAR=L el contenido del registro MAR aparece sobre el bus local del ?P. Esta señal estará normalmente activada ya que sólo en los casos de acceso directo a memoria (ADM) se pide al ?P que desconecte sus líneas del bus.

Señales de control del registro de instrucción : El registro de instrucción es de 8 bits de entrada y salida en paralelo. Su salida está conectada continuamente al decodificador de instrucciones. Por ello este registro sólo necesita de una señal de reloj para efectuar la carga del código desde el bus interno del ?P (WR_INS). La escritura en el registro de instrucción se realiza con el flanco de subida de WR_INS.

Señales de control del registro índice : El registro índice es de 16 bits y de entrada y salida en paralelo. Las señales que controlan el funcionamiento de este registro son: el *reloj de carga* (**WR_IX**) y el *control de salida* al bus interno del ?P (**/OE_IX**).

Señales de control del registro de estado : El registro de estado (llamado también PSW, *Processor Status Word*) se compone, como ya sabemos, de registros independientes de 1 bit, cada uno de ellos dispone de una señal de control para ser escrito independientemente y una única señal que controla la salida de todos los bits al bus interno (**/OE_PSW**). Cada una de las señales de escritura del PSW se denominan como se indica a continuación:

| Bit | Nombre | Descripción | Señal de escritura |
|-----|--------|------------------|--------------------|
| 0 | Z | Cero | WR_Z |
| 1 | C | Arrastre (carry) | WR_C |
| 2 | S | Signo | WR_S |
| 3 | V | Rebosamiento | WR_V |
| 4 | H | Arr. intermedio | WR_H |
| 5 | N | Substracción | WR_N |
| 6 | I | Mascara INT | WR_I |

Algunas de estas señales de escritura del PSW evolucionan simultáneamente. Estas señales son: **WR_Z**, **WR_C**, **WR_S**, **WR_V**, **WR_H** y **WR_N**, ya que son consecuencia directa del resultado de la operación que esté realizando la ULA. La señal **WR_C** puede ser manejada, además, directamente por el usuario por medio de instrucciones del ?P (instrucción de puesta a 1 del carry o de puesta a 0). Cada vez que la ULA realiza una operación lógica o aritmética, la UC reescribe los bits citados del PSW para actualizar su información. El bit de máscara de interrupción (I) es manejado por otros eventos distintos a las operaciones de la ULA del ?P. Puede ser escrito por el usuario por medio de instrucciones directas. La máscara de interrupción será manejada de forma automática por determinados procesos dentro del ?P que veremos en apartados posteriores (interrupciones).

| Registro | Tamaño (bits) | Descripción | Controles |
|-----------------|---------------|------------------|---|
| MAR | 16 | Dir. bus local | /OE_MAR, WR_MAR |
| MDR | 8 | Datos bus local | /OE_MDR, WR_MDR, DIR_MDR |
| PC | 16 | Cont. programa | /OE_PC, CK_PC, WR_PC |
| SP | 16 | Puntero pila | /OE_SP, CK_SP, WR_SP, /INC_SP |
| IX | 16 | Índice | /OE_IX, WR_IX |
| RDE1 | 8 | Entrada 1 ULA | WR_RDE1 |
| RDE2 | 8 | Entrada 2 ULA | WR_RDE2 |
| BDS | 8 | Salida ULA | /OE_BDS |
| FLAGS | 8 | Reg. de indicad. | Cada uno de los bits de este registro se maneja independientemente según la tabla anterior. |
| REG_INTR | 8 | Reg. instrucción | WR_INS |
| A | 8 | Registro A | Estos registros se controlan a través de las señales del MUX según tabla adjunta. |
| B | 8 | Registro B | |
| C | 8 | Registro C | |
| D | 8 | Registro D | |
| E | 8 | Registro E | |
| F | 8 | Registro F | |
| REG1 | 8 | Reg. interno | |
| REG2 | 8 | Reg. interno | |

4.1.- La Pila.

La pila del ?P no es más que una zona de la memoria externa que se utiliza para almacenar datos de importancia vital para determinadas funciones. Estas funciones (interrupciones, llamadas, etc.) que las veremos más adelante, requieren que en un determinado momento se pueda guardar en la memoria (en la pila) una determinada información. La dirección en donde se encuentra la primera dirección vacía de la pila está dada por el contenido del **puntero de la pila** (SP) que es un registro de 16 bits cuyo contenido lo inicializa el usuario cargando el valor correspondiente por medio del programa.

La **pila crece hacia abajo**, es decir, cada vez que se introduce una información en ella por medio de la dirección dada por el registro SP, el contenido de éste se decrementa en una unidad para apuntar a la siguiente dirección vacía. Cada vez que se extrae de la pila una información por medio de la dirección indicada por el registro SP, el contenido de éste queda incrementado en una unidad respecto al valor inmediato anterior.

Además, cuando se introduce en la pila una información de 16 bits (una dirección de retorno, por ejemplo) por medio del SP, se hace en el orden siguiente:

| Dirección | Contenido |
|-----------|--|
| SP | Parte alta de la información (bits 8 a 15) |
| SP-1 | Parte baja de la información (bits 7 a 0) |

Es decir, en la dirección más alta se deposita la parte alta de la información, mientras que en la dirección más baja se deposita la parte baja de la información.

4.2.- Ciclos fundamentales.

En lo que sigue discutimos el funcionamiento del microprocesador desde dos puntos de vista: **interno y externo**. Con ello lograremos comprender a fondo la evolución del sistema y nos permitirá realizar más fácilmente las aplicaciones.

4.2.1.- Funcionamiento interno.

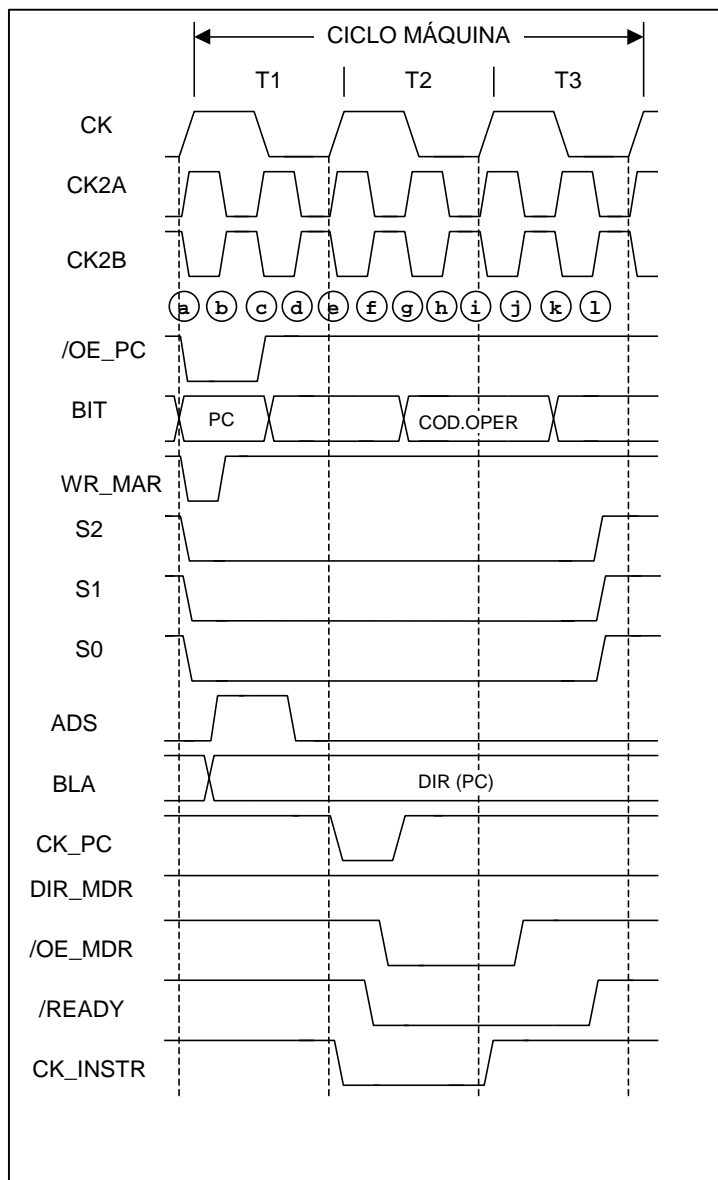
En la figura anterior del diagrama de bloques interno de la UCP podemos ver la existencia de dos buses internos (BIT y BIC) que se utilizan para interconexión de las distintas unidades. Por el BIT (*Bus Interno de Transferencias*) fluyen las distintas informaciones (datos y direcciones) entre los registros internos. Ya que por este bus ha de circular tanto información de 8 bits como de 16 bits (datos, códigos y direcciones) **el tamaño del bus BIT es de 16 bits** de los cuales los registros de 8 bits solo utilizan los 8 más bajos (7 a 0, exceptuando REG1 y REG2 que se utilizan para intercambiar operandos entre la parte baja y alta del BIT) mientras que los registros de 16 bits utilizan el bus completo o cada uno de los bytes independientemente.

Por el bus BIC (*Bus Interno de Control*) se transmiten las órdenes desde la UC a los restantes elementos del ?P. Este bus se compone de **todas las señales de control** que hemos descrito antes. El interfaz con el exterior del ?P se realiza por medio del registro MAR para las direcciones, del registro MDR para los datos y las señales de control del secuenciador de la UC (las señales del bus interno de control no son necesarias en el bus externo de control).

Básicamente, el funcionamiento del μP evoluciona en tres fases: *búsqueda del código de operación*, *búsqueda del operando* y *ejecución de la instrucción*. Esta no es más que la evolución lógica para hacer algo: averiguar qué es lo que hay que hacer, saber los elementos que intervienen y, por último, hacerlo.

FASE 1. BÚSQUEDA DEL CÓDIGO DE OPERACIÓN.

Esta instrucción en código máquina se compone de dos bytes, uno para el *código de operación* de 8 bits y un segundo byte que es el *dato* que hay que sumar al acumulador. El otro operando (el contenido del acumulador) está **direccionado intrínsecamente en el código** (como veremos más adelante al hablar de la codificación de las instrucciones).



En la primera fase del proceso (búsqueda del código de operación), el μP **envía la dirección de la instrucción a ejecutar al bus de direcciones**. Para ello activa la salida del PC al BIT y carga el MAR con ésta información. Simultáneamente, se activa la salida del MAR hacia BLA (Bus local de direcciones). Las señales de control correspondientes al ciclo a realizar en la memoria de datos (lectura de código de operación) han sido activadas por la UC (estas señales son las de estado del μP) hacia el BIC (Bus Interno de Control). Conforme sale la dirección al bus de direcciones (el MAR ha sido cargado con el contenido del PC), el PC recibe un impulso de reloj del contador y **se incrementa** en una unidad.

Una vez que el dato procedente de la memoria llega al MDR, la UC lo transfiere desde aquí al registro de instrucción. El μP averigua entonces la función a realizar (**decodificando el código** de operación en el decodificador de instrucción) y deduce que uno de los operandos es el contenido del acumulador y que el otro operando ha de buscarlo en la siguiente dirección de memoria.

En la figura podemos ver que el μP utiliza **dos relojes**, CK es suministrado externamente y CK2A y CK2B que son generados internamente a partir de CK y son de frecuencia doble que el suministrado y desplazados 180 grados. Disponiendo de estos dos relojes, el μP mejora los tiempos de respuesta funcionando únicamente con los flancos ascendentes de CK2A y CK2B.

El ciclo de fetch se compone de tres ciclos del reloj externo CK. El ciclo comienza en el instante a) con la subida de CK en T1. En este momento se activa la señal /OE_PC que hace que el contenido del PC sea colocado en el bus interno del ?P. También en este instante baja WR_MAR preparándose para escribir la dirección del PC en el registro MAR. Las señales de estado para la identificación del ciclo se sitúan en la combinación adecuada a partir de este momento ($S_0=S_1=S_2=0$).

En el momento b) se activa ADS que *señala el comienzo del ciclo* y se escribe en el registro MAR el contenido del PC subiendo la señal WR_MAR. Esto hace que la dirección válida aparezca en el bus local de direcciones (BAL) un instante después. En c) se desactiva la salida del PC (/OE_OC=H) dejando el BIT libre. En d) se desactiva ADS. En e) se prepara la señal de escritura del registro de instrucción para recibir el código. En g) se incrementa el contenido del PC en uno con la subida de CK_PC. En este momento se activa /OE_MDR para dar entrada al dato desde el bus de datos local hacia el BIT.

En el momento i) el ?P verifica si la señal /READY se encuentra activa (L). Si /READY se activa antes del tiempo necesario para cumplir con el tiempo de setup respecto al flanco de subida de CK2A la UC determina que ha de recoger la información del BIT y sube la señal CK_INS. Si /READY no hubiera estado activa en el momento citado antes, el ?P deja pasar un periodo de CK creando un periodo de espera (T_w) y vuelve a realizar la comprobación en el siguiente flanco ascendente de T3. Esto se repite hasta que en un determinado flanco de CK la señal /READY se verifique que está activa y el ?P continua con el tiempo T3 del ciclo en curso.

Esta forma de prolongar los ciclos es igual en cualquier ciclo que requiera la señal /READY para acabarlo y permite el acoplamiento perfecto de dispositivos de distintas características temporales al ?P (velocidades). Como consecuencia de ello, si algún dispositivo no activa la señal /READY, el ?P se queda esperándolo por tiempo indefinido y el sistema queda detenido. Esta señal normalmente se encargara de activarla un contador programado para generar un número de *tiempos de espera* predeterminado.

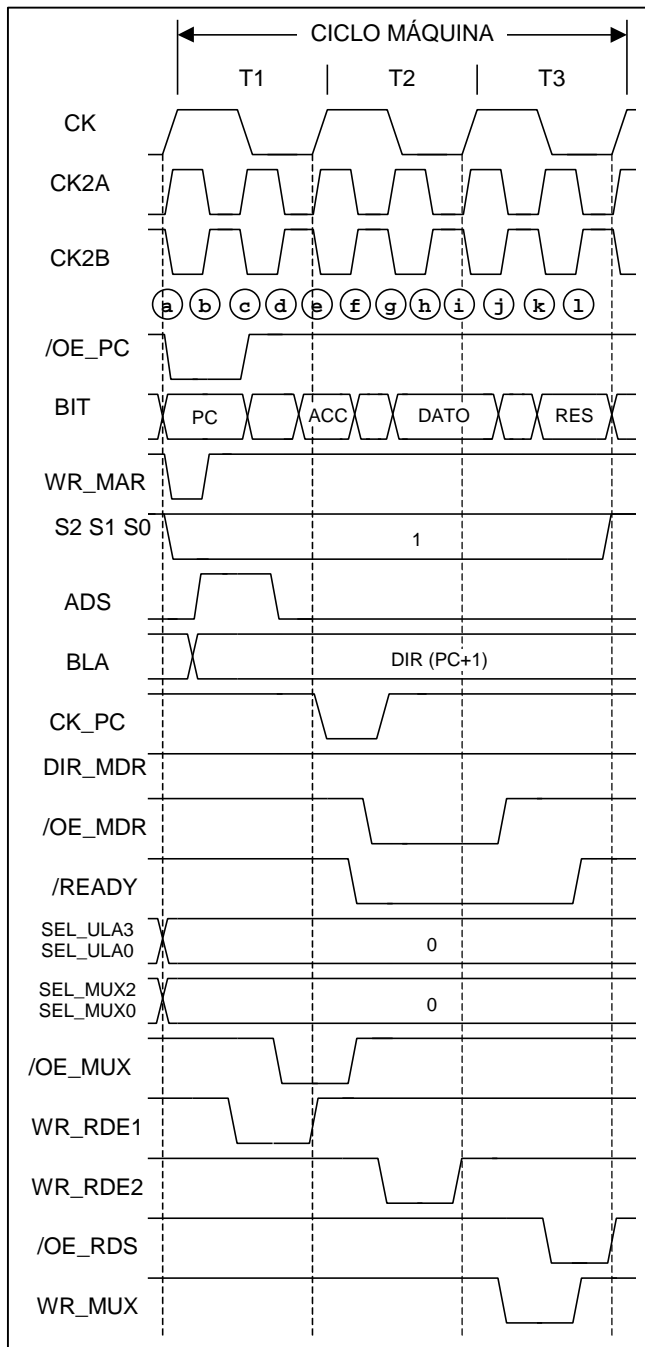
En la figura anterior, el ?P encuentra activada la señal /READY en el flanco de subida de T3, por lo que el dato es transferido por MDR al registro de instrucciones en donde es depositado en el instante i). En el momento j) queda libre el BIT pues se desactiva la señal /OE_MDR y en k) se desactivan las señales de estado del ?P (S_2-S_0) indicando con ello el final de este ciclo. La decodificación de la instrucción entrada en el registro de instrucción se realiza *inmediatamente* a partir del momento i) con lo cual el ?P al llegar al final del ciclo ya “sabe” lo que ha de hacer a continuación.

En la figura anterior se ha representado el contenido del BIT en cada momento. Podemos comprobar que el bus está ocupado solo por el contenido del PC durante la primera parte y por el MDR en la segunda.

FASE 2. BÚSQUEDA DEL OPERANDO.

Como consecuencia de la decodificación del código de operación por parte de la UC, ésta genera órdenes para realizar varias funciones. La decodificación de este código de operación obliga al secuencial de la UC a realizar un *ciclo adicional* para buscar el otro operando en la dirección siguiente de memoria, para lo cual el contenido del PC se vuelca al bus externo ($PC \nrightarrow BIT \nrightarrow MAR$) y se activan las señales correspondientes para leer la siguiente dirección de

memoria cuyo contenido es el segundo operando. A continuación da orden al MUX (Multiplexor de los registros accesibles) para volcar al bus interno el contenido del acumulador y cargarlo en el registro de entrada de la ULA (Acc \rightarrow BIT \rightarrow RDE1). Así mismo, la UC ha activado los controles de la ULA para realizar la operación de sumar.



Una vez que el registro MAR ha sido cargado con el contenido del PC, éste recibe un nuevo impulso y se incrementa en una unidad para apuntar a una nueva dirección de la memoria de programa (la dirección del siguiente código de operación). Cuando el dato procedente de la memoria de programa se encuentra disponible, la UC transfiere la información desde el MDR al otro registro de entrada de la ULA (MDR \rightarrow BDI \rightarrow RDE2).

En la figura observamos que el ciclo se compone de tres periodos de reloj y que el desarrollo no solo lee de la memoria el operando sino que prepara la ULA para la operación que se va a realizar y carga ésta con el otro operando.

La primera parte de este ciclo máquina es similar al fetch en cuanto que el PC da la dirección en donde se encuentra el operando y la UC activa las señales de estado para indicar el tipo de ciclo en curso. Pero además la UC pone en el momento a) la **combinación de control** adecuada para la ULA. En el espacio de tiempo que hay entre c) y g) el ?P realiza la **transferencia del otro operando** desde el acumulador al registro de entrada de la ULA. En a) selecciona el registro fuente (0 = acumulador) y en d) vuelca su contenido al BIT recogiénolo en el registro RDE1 de la ULA en el momento e). En f) queda libre el BIT para la entrada del operando desde la memoria.

El ciclo termina colocando el resultado de la operación en el acumulador. En k) el contenido del registro de salida de la ULA es volcado al BIT y en l) es escrito en el acumulador.

FASE 3. EJECUCIÓN.

Esta fase está dentro de la fase 2 como se ha descrito antes. La fase de ejecución de una instrucción puede emplear más ciclos de reloj que en el caso analizado. En estos casos el ?P

utiliza tantos ciclos de reloj como necesite para ejecutar la orden. Se dice en estos casos que el ?P está realizando **ciclos internos**. Durante estos periodos el ?P puede mantenerse sin activar ninguna de las señales externas sin que ello implique problema alguno.

Al finalizar la fase de ejecución acaba lo que se llama ciclo de instrucción del ?P. Los *bits de estado* que dependen del resultado de la ULA se actualizan durante esta fase en el instante 1) quedando el PSW con el contenido correcto al finalizar el ciclo de instrucción.

4.2.2.- Funcionamiento externo.

En el caso de la instrucción ejecutada en el apartado anterior, visto desde el exterior, se realizan dos accesos a la memoria, uno para el fetch y otro para obtener el segundo operando (ya que el primero es el contenido del acumulador). Las funciones internas (tales como las transferencias entre registros) no acceden al exterior del ?P. Como sucedió en el análisis anterior cuando el ?P hizo la transferencia del dato desde el acumulador al registro de entrada de la ULA. La secuencia desarrollada hacia el exterior del ?P evoluciona así:

- **Ciclo de búsqueda del código de operación.** Acceso de lectura sobre la memoria de programa.
- **Ciclo de lectura de dato.** Acceso de lectura sobre la memoria de programa para obtener el dato.

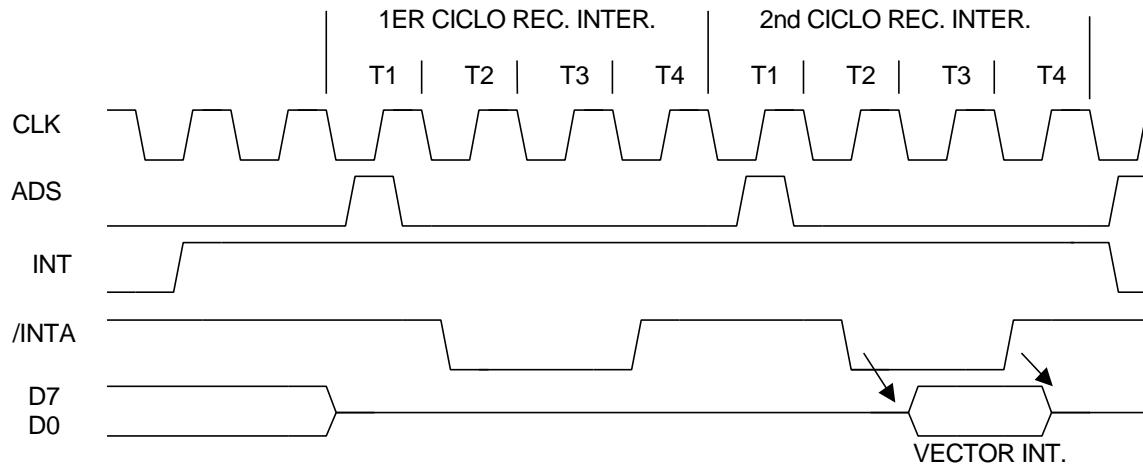
Si la instrucción fuera de escritura del contenido de un registro interno en una determinada posición de la memoria, la secuencia evoluciona así:

- **Lectura del código de operación de la instrucción.** Ciclo de fetch con un acceso de lectura sobre la memoria de programa.
- **Lectura de la dirección en donde se va a escribir el dato.** Búsqueda de la dirección de destino (operando de la instrucción). Se realizan dos accesos de lectura sobre la memoria de programa (ya que suponemos que la dirección se compone de dos bytes).
- **La dirección de destino** leída en los dos ciclos anteriores hay que componerla en el orden correcto y enviarla al registro MAR. Se trata de una operación interna al ?P que no desarrolla ciclo externo, pero que consume ciclos de reloj para su ejecución.
- **Escritura del dato en la dirección indicada por la instrucción.** Implica un ciclo de acceso de escritura sobre la memoria de datos.

Como podemos observar, desde el exterior del ?P solo podemos apreciar los ciclos para los cuales el ?P requiere acceso al exterior. Internamente se ejecutarán funciones que no tienen representación externa y que solo podemos observar como ciclos consumidos sin actividad aparente.

Ciclos funcionales básicos : Los ciclos funcionales básicos son todos los ciclos externos diferentes que es capaz de desarrollar el ?P durante su funcionamiento. Para cada modelo de ?P existe una colección de **ciclos básicos** que en general, son diferentes de un modelo a otro. Sin embargo, todos los ?Ps realizan una colección común de ciclos que son los que estudiaremos a continuación. Por supuesto que los ciclos aquí discutidos son desarrollados por cada ?P diferente de forma diferente. Ya los hemos analizado, por lo cual no insistiremos en ellos.

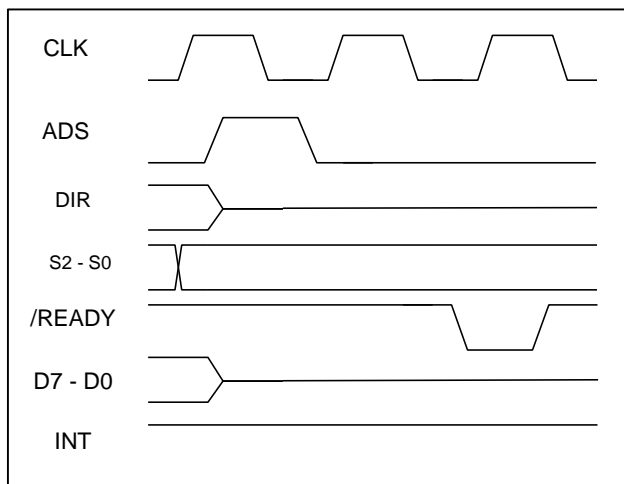
Ciclo de reconocimiento de interrupción : Cuando el ?P reconoce una interrupción enmascarable realiza un ciclo especial para indicarlo ya que afecta al funcionamiento de los elementos exteriores del ?P cuando este funciona con interrupciones vectorizadas o autovectorizadas. La figura siguiente muestra el desarrollo del ciclo.



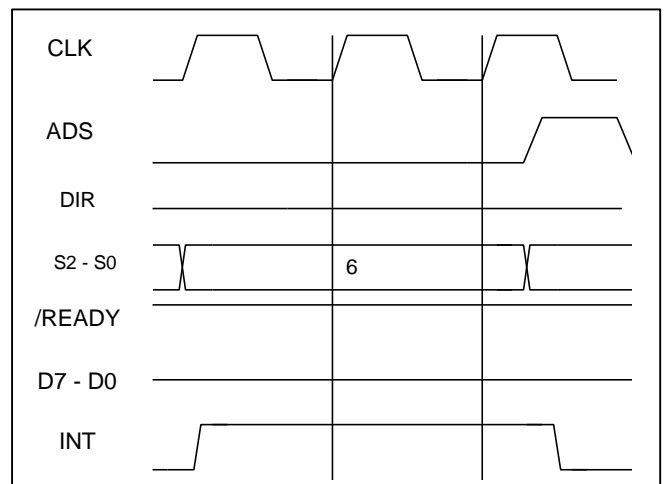
Podemos observar que las señales de estado toman la combinación adecuada para indicar esta situación. El dispositivo que interrumpe vuelca el vector y activa la señal /READY para indicar al ?P el final del ciclo. En este caso el ?P no necesita entregar dirección alguna.

Ciclo de Halt : El ?P dispone de una instrucción para detener el proceso y quedar a la espera de la llegada de una interrupción. Esta instrucción se denomina HALT. Cuando el programa ejecuta una instrucción HALT el ?P se detiene en su ejecución y solo sale de esta situación si recibe una interrupción. El cronograma de la figura siguiente corresponde a este ciclo.

Entrada en HALT



Salida de HALT por INT

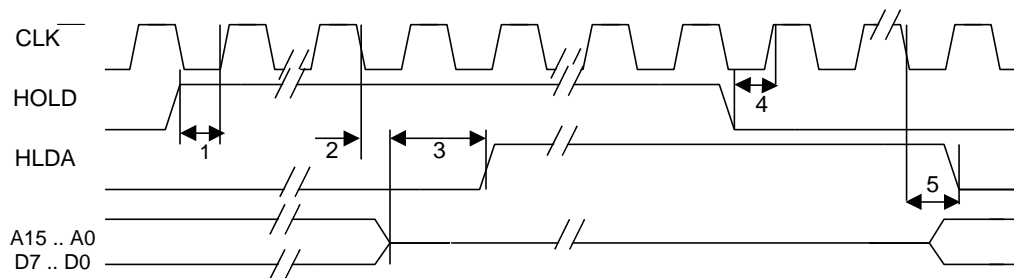


Ciclo Hold : El ?P realiza un ciclo hold cuando se activa la señal HOLD. Cuando esto sucede, el ?P acaba el ciclo máquina en que está en curso y a continuación entra en estado hold. La situación de hold del ?P es señalizada por la señal HLDA. Cuando HLDA se activa (HLDA=H), el ?P está en estado hold, con ello indica que las líneas de direcciones, datos y

control del ?P están en alta impedancia de forma que otro dispositivo puede tomar el control del bus local.

La señal HLDA no pasa a triestado ya que ha de **señalar la situación de hold** del ?P. Esta situación de desconexión del ?P de los buses se mantiene mientras que esté activa la señal HOLD. Durante este tiempo, el ?P no realiza ninguna función, ha detenido el proceso al final de un ciclo máquina y espera la desactivación de la señal HOLD para continuar con el siguiente ciclo máquina.

Cuando la señal HOLD se desactiva, el ?P lo encontrará desactivado en el momento de muestreo que corresponda (flanco de subida de cualquier periodo del reloj durante el estado hold) y en el ciclo de reloj siguiente **reanuda su actividad normal** tomando de nuevo el control de los buses y realizando el siguiente ciclo máquina. Inmediatamente después de muestrear el ?P la señal HOLD desactivada, es desactivada la señal HLDA para indicar la salida del estado hold del ?P. La figura siguiente muestra el cronograma desarrollado por este ciclo. En ella podemos ver que HOLD solo se muestrea al final del ciclo máquina en curso (en el mismo instante que se muestrea ready activado), y que en la respuesta por parte del ?P se introduce a partir del siguiente flanco de reloj. En ese momento se desconectan las líneas de los buses de direcciones y datos y las señales de control. También en ese momento se activa HLDA.



El tiempo que transcurre desde que se activa HOLD hasta que el ?P responde se denomina **tiempo de latencia del hold** y es variable de un ?P a otro, ya que dependen por un lado de la longitud de cada ciclo máquina que sea capaz de realizar, y por otro, del tiempo que utilice el ?P para funciones internas en las que no se producen ciclos en el bus local.

5.- Juego de Instrucciones.

Toda UCP dispone de un conjunto de códigos capaz de ser identificados por su decodificador de instrucción. A este conjunto se denomina juego de instrucciones de la UCP y será más o menos extenso dependiendo del modelo de UCP.

5.1.- Estructura.

Cualquier instrucción de una UCP se compone de dos campos denominados **código de operación** y **operando** respectivamente.

| |
|---------------------|
| CODIGO DE OPERACIÓN |
|---------------------|

| |
|----------|
| OPERANDO |
|----------|

CODIGO DE OPERACIÓN : El código de operación es una *información codificada* que informa a la UCP de la función que debe realizar. La codificación es particular para cada UCP y tienen una relación muy estrecha con el microprograma contenido en la UC para la decodificación de la instrucción. El código de operación contiene, además, información que indica a la UCP cómo se va a localizar el operando de esa instrucción (lo que veremos como direccionamiento).

Normalmente, los códigos de operación de una UCP no tienen la misma interpretación en otra UCP (salvo algunos casos concretos de UCPs pertenecientes a la misma familia o compatibles). Esto hace que un programa escrito para una determinada UCP no sea, por código, utilizable en otra, salvo los casos anotados.

En las UCPs de 8 bits de datos, el código de operación suele estar formado por un único byte, lo que significa disponer de hasta 256 posibles códigos diferentes. Algunas UCPs expanden estas posibilidades utilizando un byte como precódigo y otro como código de operación realmente. Esto hace que, en principio, cada precódigo expande cada una de las combinaciones en 256 posibilidades más.

OPERANDO : El campo del operando de una instrucción es la parte de la instrucción que da información sobre el operando. Esta información puede ser tan clara como que se trate del propio operando o, dependiendo del tipo de instrucción, una *información para poder localizarlo*, es decir, una dirección directamente o algún dato para poder formar la dirección del operando.

La existencia del campo de operando está condicionada al *tipo de direccionamiento* indicado en el código de operación de la instrucción que corresponda. Así pues, si la instrucción es para sumar dos datos que están contenidos en sendos registros internos (uno de ellos en el acumulador) para dejar el resultado en el acumulador, solo es necesaria la existencia del campo del código de operación ya que éste informa que se trata de una operación interna a la UCP donde tanto el origen de los datos como el destino del resultado son registros internos.

En las UCPs de 8 bits, el campo de operando, cuando existe, suele tener un tamaño de uno o dos bytes.

5.2.- Modos de direccionamiento.

Como decíamos en apartados anteriores, la instrucción consta de un campo denominado de código de operación y otro de operando. El código, además de identificar la instrucción, indica el modo de localizar el operando (modo de direccionamiento). Por eso, el campo del operando contiene o bien el operando directamente o bien una información suficiente para encontrarlo. Los modos de direccionamiento son los *distintos métodos* que utiliza la UCP para localizar el operando.

Tipos de direccionamiento.

Las posibilidades de modos diferentes de direccionamiento son muy numerosas. Nosotros describiremos las más básicas ya que las demás son combinaciones de éstas. Tenemos que anotar que referiremos los modos de direccionamiento a nuestra UCP modelo que dispone de 8 bits de datos y 16 de direcciones. Esto no quita generalidad a las descripciones y nos permite concretar los dibujos y los ejemplos.

Los distintos tipos de direccionamientos son los que siguen:

- ✍ *Implícito*
- ✍ *Inmediato*
- ✍ *Directo*
- ✍ *Extendido*
- ✍ *Indirecto*
- ✍ *Indirecto registrado*
- ✍ *Basado*
- ✍ *Indexado*
- ✍ *Indexado a base*
- ✍ *A bit*
- ✍ *Otros*

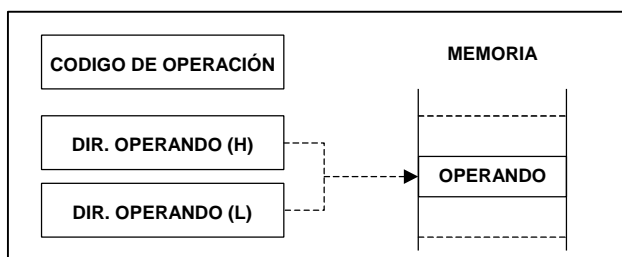
A continuación discutimos cada uno de ellos.

Direccionamiento Implícito : Cuando la operación se realiza sobre **registros internos de la UCP** obligatoriamente, la instrucción se compone únicamente de la parte de código ya que éste en sí mismo implica el registro sobre el que ejecutar. Por ejemplo es el caso de la instrucción de incrementar el contenido del acumulador: INC A. Las instrucciones con este modo de direccionamiento no tienen campo de operando.

Direccionamiento Inmediato : El direccionamiento es inmediato cuando el contenido del campo del operando es el **propio operando**. En nuestra UCP la instrucción consta de dos o tres bytes (según que la instrucción opere sobre registros de 8 o 16 bits, respectivamente), uno para el código de operación y otro para el operando. La instrucción que carga el dato 0ADH en el acumulador con direccionamiento directo (MOV A, #0ADH), es un ejemplo de este caso.

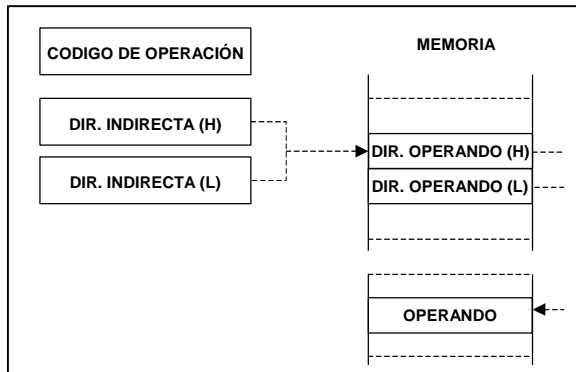
Direccionamiento Directo : Cuando se utiliza este modo de direccionamiento, el campo de operando de la instrucción no contiene al operando en sí mismo sino la **dirección donde se encuentra** realmente éste en la memoria.

La dirección del operando puede estar formada por uno o dos bytes, lo que hace que la instrucción completa sea de dos o tres bytes. Si la dirección es de un byte, el operando se encuentra obligatoriamente en las 256 posiciones de memoria más bajas, a estas direcciones algunos autores las denominan **página cero** y lo describen como **direccionamiento a página 0**. La figura siguiente muestra el direccionamiento directo.

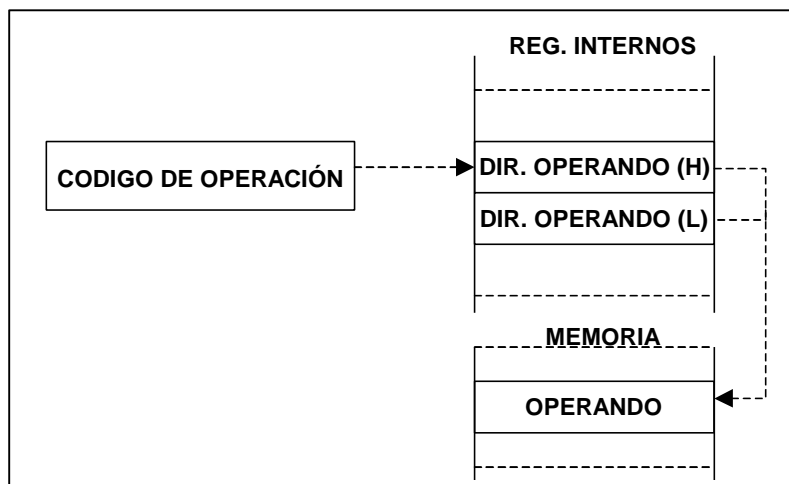


Si la dirección se compone de dos bytes, el operando se puede encontrar en una de las 64K posiciones de memoria posibles. En este caso el direccionamiento pasa a llamarse **EXTENDIDO**.

Direccionamiento Indirecto : Cuando se utiliza este modo de direccionamiento, el contenido del campo de operando de la instrucción es una *dirección de la memoria en donde se encuentra la dirección efectiva del operando*. Es decir, la UCP obtiene del campo de operando una dirección a la que accede de lectura obteniendo de ella la dirección real del operando. En la figura se muestra este modo de direccionamiento.

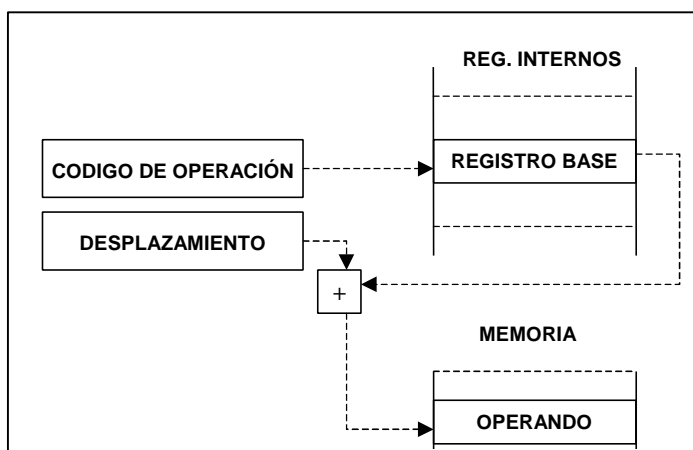


Este modo de direccionamiento es bastante sofisticado y no es frecuente verlo implementado en UCP comerciales. La ejecución de una instrucción con este tipo de direccionamiento en una UCP como el modelo nuestro implica *seis accesos a memoria* para conseguir el operando.



Algunas UCPs, como el 8085 o el Z-80, implementan una versión de este direccionamiento que se denomina ***direccionamiento INDIRECTO REGISTRADO***. Para ello utilizan un par de registros internos como punteros de memoria cuyo contenido indica la dirección del operando. Esto significa que para ejecutar esta instrucción se ha de realizar con anterioridad la carga de estos registros.

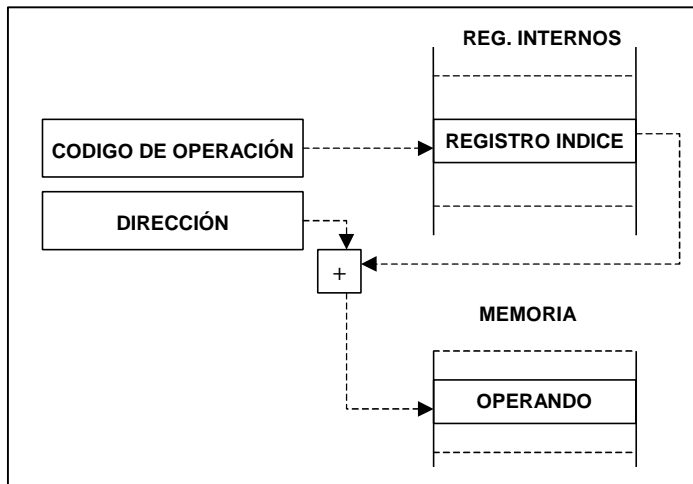
Direccionamiento Basado : Este modo de direccionamiento del operando implica a un registro interno de la UCP denominado *registro Base*. El código de operación hace referencia a este registro base que contiene una dirección (dirección base). La instrucción, además del código, contiene un valor para el desplazamiento de la dirección base. La dirección efectiva del operando se calcula sumando el valor del desplazamiento al contenido del registro base.



La dirección efectiva del operando se calcula sumando el valor del desplazamiento al contenido del registro base.

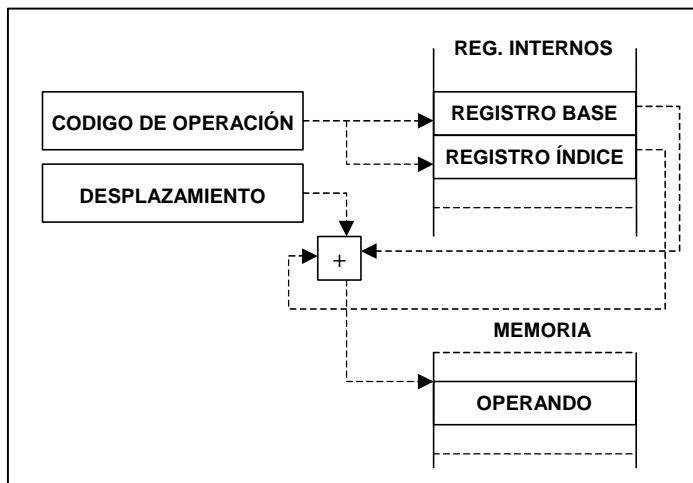
Hay que observar que este modo de direccionamiento es un direccionamiento indirecto registrado modificado. El valor del desplazamiento puede darse con un número con o sin signo, según sea la UCP, esto cambia el campo de aplicación en cada caso.

Direccionamiento Indexado



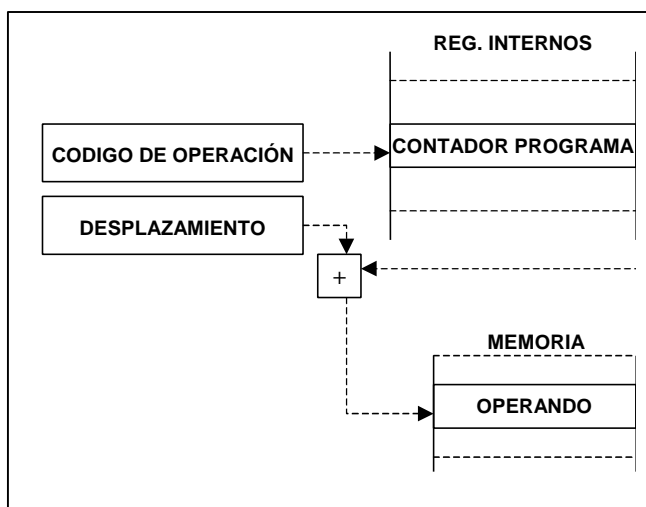
Se trata de una estructura similar a la anterior, salvo que ahora, el desplazamiento no lo entrega la instrucción sino que se encuentra contenido en un registro interno a la UCP denominado *registro índice*. La instrucción contiene en el campo de operando una dirección completa. La dirección efectiva del operando se calcula sumando el contenido del registro índice a la dirección dada.

Direccionamiento Indexado a base



Este es un modo de direccionamiento que combina los dos anteriores, el direccionamiento basado y el indexado. El código de operación implica a dos registros internos a la UCP, uno hace de registro base y el otro de registro índice. Además, la instrucción aporta un desplazamiento. La dirección efectiva del operando se calcula sumando el contenido de los registros implicados y el desplazamiento aportado por la instrucción.

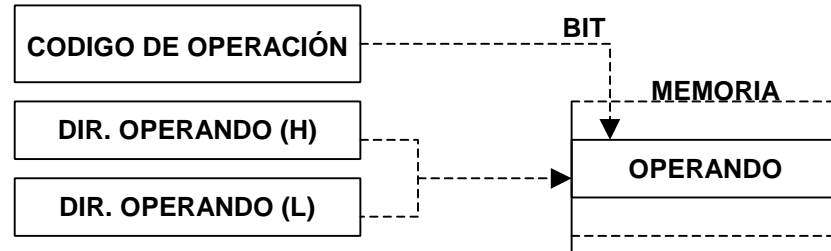
Direccionamiento Relativo



Este es un tipo de *direccionamiento basado* en donde el valor de base es el contenido del contador de programa. La instrucción aporta un valor de desplazamiento para la base. El cálculo de la dirección efectiva se realiza sumando el valor de la base con el valor del desplazamiento.

Si el desplazamiento se entrega en *complemento a dos*, este permitirá desplazarse hacia adelante o hacia atrás, respecto al PC. Este direccionamiento es muy útil en determinados tipos de *saltos* (saltos relativos).

Direccionamiento a un Bit : Hasta ahora, los distintos modos de direccionamiento están enfocados a localizar un operando y éste es de 8 bits (un byte). Si *dentro de este byte* nos interesa manejar el estado de un bit determinado (de 0 a 7), necesitamos realizar otras operaciones adicionales (AND Y OR de bits). Pero si la UCP dispone del direccionamiento a bit, se puede manejar directamente con una única instrucción.



Este modo de direccionamiento utiliza un *doble direccionamiento*, por un lado ha de localizar el byte al que pertenece el bit deseado y por otro lado, el bit en sí mismo. La localización del byte se hace por uno de los modos de direccionamientos ya descritos, mientras que la localización del bit se hace por medio de tres bits del código de operación de la instrucción con este modo de direccionamiento (ocho posibilidades).

5.3.- Clasificación de las instrucciones.

Cada UCP dispone de un juego de instrucciones propio que la caracteriza. Sin embargo, las instrucciones de cualquier UCP pueden clasificarse en los grupos que siguen:

- /// Instrucciones de carga.
- /// Instrucciones lógicas.
- /// Instrucciones aritméticas.
- /// Instrucciones de saltos.
- /// Instrucciones de control.
- /// Instrucciones de E/S.
- /// Otras instrucciones.

De cada uno de los grupos hacemos una discusión en los apartados que siguen. Antes de comenzar a analizar las instrucciones describimos la nomenclatura utilizada en el texto que sigue.

- a) Puesto que la UCP es un *dispositivo digital*, solo puede tratar los niveles lógicos H y L que en lógica positiva se corresponden con el 1 y el 0 del álgebra Booleana. Por lo tanto, para indicarle a la UCP que deseamos realizar una suma del acumulador con el dato (32, por ejemplo) tenemos que indicárselo por medio del *código correspondiente* a esta instrucción y, además le hemos de dar el dato también en binario. De esta forma si C5H se corresponde con el código de operación de la instrucción de suma, tendremos que decirle a la UCP que realice la instrucción:

11000101 001000000

o lo que es lo mismo:

C5 20 H

De este modo le hablamos directamente a la UCP pero nos encontramos con el problema de que el código 11000001001 no es demasiado cómodo de entender para nosotros. Además nos es muy fácil confundirnos con un código parecido como el 0110001010 que no indica una suma sino una carga de registro interno. Por ello, las instrucciones las representamos por un conjunto de letras que denominamos *mnemónico* de la instrucción. Cada instrucción tiene para

cada UCP una codificación unívoca e invariable en binario (que es el único lenguaje que es capaz de reconocer la UC de la UCP) y, por lo tanto una representación única en mnemónico.

Algunos autores denominan a este mnemónico como *lenguaje ensamblador*, dejando la acepción *lenguaje máquina* para la representación binaria del código de operación. Si escribimos nuestros programas en mnemónico, hemos de traducirlos al lenguaje binario para que sea asimilable por la UCP. Realizar esto es una tarea incómoda que consiste en ir consultando en una tabla cada mnemónico y obtener el código binario correspondiente. Esta tarea la dejamos para que la realice un programa que es capaz de interpretar el programa en mnemónico y producir una salida en código binario. Este programa se denomina *programa ensamblador*.

Para la descripción de las instrucciones de la UCP utilizaremos algunos acuerdos de representación que nos van a facilitar el trabajo.

- b) Los **registros internos de uso general** de 8 bits de la UCP los denominaremos, de una forma genérica, por “r”. De esta forma, en la instrucción que aparezca una “r” podremos poner en su lugar cualquiera de los registros de 8 bits, esto es: A, B, C, D, E o F.
- c) Los **registros internos de 16 bits** de la UCP los denominamos por la letra “R”. R representa cualquier registro interno de 16 bits como el SP, el IX o el PC.

RL es la parte baja del registro (bits 7 a 0) como PCL o IXL y RH se refiere a la parte alta del registro (bits 15 a 8) como SPH o PCH.

- d) Cuando en una instrucción tenemos que poner un **dato de 8 bits**, lo representaremos, de una forma genérica, por la letra “d”.
- e) Cuando la **información sea de 16 bits**, lo representaremos por “dd” (se trate de una dirección o de un dato).
- f) Cuando nos sea necesario hacer referencia al **contenido de una posición** de la memoria cuya dirección sea dd, lo representaremos poniendo la dirección entre paréntesis. Es decir que (dd) significa que hacemos referencia al contenido de la dirección dd. Igualmente si hablamos del dato (IX+d) nos estamos refiriendo al dato contenido en la dirección dada por IX+d.
- g) El **bit de Carry** del registro de estado lo representaremos por CY cuando sea posible confundirlo con el registro C de la UCP.

En lo que sigue discutiremos en detalle el funcionamiento de cada uno de los tipos de instrucciones. Esta discusión nos define gran parte de la funcionalidad del ?P y es fundamental su conocimiento para poder desarrollar los programas que controlan una determinada aplicación.

5.3.1.- Instrucciones de carga

El grupo de instrucciones de carga permite la **copia de datos** entre registros internos o posiciones de la memoria. Tanto el origen como el destino del dato puede ser un registro interno o posición de memoria externa.

Estas instrucciones reciben el nombre genérico de **MOV (MOVIMIENTO)** y se dividen en tres grupos, las que cargan registros internos de 8 bits, las que cargan registros internos de 16 bits y las que cargan posiciones de la memoria. Ninguna de ellas afecta al registro de estado ya que en esta función no interviene la ULA. Por ello el registro de estado queda como sigue:

Indicador I N H V S C Z

Estado nc nc nc nc nc nc nc

- a) **Instrucciones de carga de registros de 8 bits** : Estas instrucciones se encargan de cargar un registro interno de la UCP con una información. El origen de esta información depende del modo de direccionamiento que se utilice. Con estas instrucciones podemos mover datos entre cualquiera de los registros internos y traer un dato de cualquier posición de la memoria.

| Direccionamiento | Sintaxis | Función | Longitud (bytes) |
|------------------|------------|-------------------------|------------------|
| Implícito | MOV r,r' | $r \not\Leftarrow r'$ | 1 |
| Inmediato | MOV r,d | $r \not\Leftarrow d$ | 2 |
| Directo | MOV r,(dd) | $r \not\Leftarrow (dd)$ | 3 |

- b) **Instrucciones de carga de registros de 16 bits** : Estas instrucciones nos permiten cargar los distintos registros de 16 bits de la UCP con un determinado valor no siendo posible, sin embargo la transferencia entre registros.

| Direccionamiento | Sintaxis | Función | Longitud (bytes) |
|------------------|-----------|--|------------------|
| Inmediato | MOV R,dd | RL \Leftarrow (PC+1) RH \Leftarrow (PC+2) | 3 |
| Directo | LD R,(dd) | RL \Leftarrow (dd) RH \Leftarrow (dd+1) | 3 |

- c) **Instrucciones de carga de memoria** : De forma similar a las instrucciones anteriores, éstas nos permiten cargar desde un registro interno de la UCP a la memoria.

| Direccionamiento | Sintaxis | Función | Longitud (bytes) |
|------------------|-----------------|--------------------------|------------------|
| Implícito | MOV (dd),r | (dd) \Leftarrow r | 3 |
| Inmediato | MOV (dd),d | (dd) \Leftarrow d | 4 |
| Directo | MOV (dd),(dd) | (dd) \Leftarrow (dd) | 5 |
| Directo-Index | MOV (IX+d),(dd) | (IX+d) \Leftarrow (dd) | 4 |
| Indexado | MOV (dd),(IX+d) | (dd) \Leftarrow (IX+d) | 4 |

5.3.2.- Instrucciones lógicas

Las instrucciones pertenecientes a este grupo realizan funciones lógicas entre dos operandos. Las **funciones lógicas** que realizan son: AND, OR, XOR, Inversión, Desplazamiento y Rotación. Estas instrucciones tienen siempre como uno de los operandos el acumulador de la UCP (registro A). El otro operando puede ser el contenido de un registro interno o de una posición de la memoria. El resultado siempre se vuelca sobre el acumulador.

Todas las instrucciones lógicas se realizan por medio de la ULA y, por lo tanto, el registro de estado se ve afectado por el resultado de la operación (salvo casos excepcionales).

- a) **Función AND** : Estas instrucciones realizan la función lógica AND entre dos operandos bit a bit. El registro de estado resulta modificado por la ejecución de una instrucción de este tipo según se indica a continuación.

| | | | | | | | |
|------------------|----------|-----------------------|---|---|------------------|---|---|
| Indicador | I | N | H | V | S | C | Z |
| Estado | nc | 0 | 1 | 0 | c | 0 | c |
| Direccionamiento | Sintaxis | Función | | | Longitud (bytes) | | |
| Implícito | AND r | $A \leftarrow A \& r$ | | | 1 | | |

| | | | |
|-----------|------------|--------------------------------|---|
| Inmediato | AND d | $A \not\leftarrow A \& d$ | 2 |
| Directo | AND (dd) | $A \not\leftarrow A \& (dd)$ | 3 |
| Indexado | AND (IX+d) | $A \not\leftarrow A \& (IX+d)$ | 2 |

- b) **Función OR** : Estas instrucciones realizan la función lógica OR entre dos operandos bit a bit. El registro de estado resulta modificado por la ejecución de una instrucción de este tipo según se indica a continuación.

| | | | | | | | |
|------------------|-----------|---------------------------|---|---|------------------|---|---|
| Indicador | I | N | H | V | S | C | Z |
| Estado | nc | 0 | 0 | 0 | c | 0 | c |
| <hr/> | | | | | | | |
| Direccionamiento | Sintaxis | Función | | | Longitud (bytes) | | |
| Implícito | OR r | A ← A # r | | | 1 | | |
| Inmediato | OR d | A ← A # d | | | 2 | | |
| Directo | OR (dd) | A ← A # (dd) | | | 3 | | |
| Indexado | OR (IX+d) | A ← A # (IX+d) | | | 2 | | |

- c) **Función XOR** : Estas instrucciones realizan la función lógica OR entre dos operandos bit a bit. El registro de estado resulta modificado por la ejecución de una instrucción de este tipo según se indica a continuación.

| | | | | | | | |
|------------------|------------|----------------|---|---|------------------|---|---|
| Indicador | I | N | H | V | S | C | Z |
| Estado | nc | 0 | 1 | 0 | c | 0 | c |
| | | | | | | | |
| Direccionamiento | Sintaxis | Función | | | Longitud (bytes) | | |
| Implícito | XOR r | A ← A ? r | | | 1 | | |
| Inmediato | XOR d | A ← A ? d | | | 2 | | |
| Directo | XOR (dd) | A ← A ? (dd) | | | 3 | | |
| Indexado | XOR (IX+d) | A ← A ? (IX+d) | | | 2 | | |

- d) **Función inversión** : Esta función denominada de inversión o complemento, solo se realiza sobre el contenido del acumulador. El registro de estado se ve modificado como sigue:

| | | | | | | | |
|------------------|----------|--------------------|--------|---------|------------------|---------|---------|
| Indicador Estado | I nc | N 1 | H 1 | V nc | S nc | C nc | Z nc |
| Direccionamiento | Sintaxis | Función | | | Longitud (bytes) | | |
| Implícito | CPL A | $A \nleftarrow A'$ | | | 1 | | |

- e) **Desplazamientos** : Las instrucciones de desplazamiento se realizan siempre sobre el contenido del acumulador, por lo que el direccionamiento es siempre implícito. Se distinguen dos tipos de desplazamientos: con carry y sin carry según sea un caso u otro se corresponde con la denominación de desplazamiento aritmético (con carry) y desplazamiento lógico (sin carry).

Las instrucciones de desplazamiento con carry modifican el contenido del registro de estado como sigue:

| | | | | | | | |
|------------------|---------|--------|--------|---------|--------|--------|--------|
| Indicador Estado | I nc | N 0 | H 0 | V nc | S c | C c | Z c |
|------------------|---------|--------|--------|---------|--------|--------|--------|

Mientras que las instrucciones de desplazamiento sin carry modifican el contenido del registro de estado como sigue:

| Indicador | I | N | H | V | S | C | Z |
|------------------|----------|---------|---|----|------------------|----|---|
| Estado | nc | 0 | 0 | nc | c | nc | c |
| Direccionamiento | Sintaxis | Función | | | Longitud (bytes) | | |
| Implícito | SRA | | | | 1 | | |
| Implícito | SRCA | | | | 1 | | |
| Implícito | SLA | | | | 1 | | |
| Implícito | SLCA | | | | 1 | | |

- f) **Rotaciones** : Las instrucciones de rotación se realizan siempre sobre el contenido del acumulador por lo que el direccionamiento es siempre implícito. Se distinguen dos tipos de desplazamientos: con carry y sin carry según sea un caso u otro se corresponde con la denominación de rotación aritmética (con carry) y rotación lógica (sin carry).

Las instrucciones de rotación con carry modifican el contenido del registro de estado como sigue:

| | | | | | | | |
|-----------|----|---|---|----|---|----|---|
| Indicador | I | N | H | V | S | C | Z |
| Estado | nc | 0 | 0 | nc | c | nc | c |

Mientras que las instrucciones de rotación sin carry modifican el contenido del registro de estado como sigue:

| Indicador | I | N | H | V | S | C | Z |
|------------------|----------|---------|---|----|------------------|----|---|
| Estado | nc | 0 | 0 | nc | c | nc | c |
| Direccionamiento | Sintaxis | Función | | | Longitud (bytes) | | |
| Implícito | RRA | | | | 1 | | |
| Implícito | RRCA | | | | 1 | | |
| Implícito | RLA | | | | 1 | | |
| Implícito | RLCA | | | | 1 | | |

5.3.3.- Instrucciones aritméticas

Las operaciones aritméticas se realizan por medio de este grupo de instrucciones. Siempre trabajan con dos operandos, uno de los cuales es el contenido del acumulador. El otro operando puede ser el contenido de un registro interno o de una posición de memoria.

Las instrucciones aritméticas realizan las funciones de suma, resta, comparación, etc. Estas instrucciones se realizan, también, por medio de la ULA y el registro de estado de la UCP se ve afectado por el resultado de la operación ejecutada.

- a) **Instrucción de suma** : Estas instrucciones realizan por medio de la ULA, la suma aritmética de dos operandos. Como consecuencia de ello, el registro de estado se ve modificado como sigue:

| | | | | | | | |
|-----------|----|---|---|---|---|---|---|
| Indicador | I | N | H | V | S | C | Z |
| Estado | nc | 0 | c | c | c | c | c |

Son posibles dos instrucciones de suma: con carry y sin carry.

| Direccionamiento | Sintaxis | Función | | | Longitud (bytes) | | |
|------------------|----------|---------|--|--|------------------|--|--|
|------------------|----------|---------|--|--|------------------|--|--|

| | | | |
|-----------|------------|------------------------------------|---|
| Implícito | ADD A,r | $A \not\leftarrow A + r$ | 1 |
| Inmediato | ADD d | $A \not\leftarrow A + d$ | 2 |
| Directo | ADD (dd) | $A \not\leftarrow A + (dd)$ | 3 |
| Indexado | ADD (IX+d) | $A \not\leftarrow A + (IX+d)$ | 2 |
| Implícito | ADC A,r | $A \not\leftarrow A + r + CF$ | 1 |
| Inmediato | ADC d | $A \not\leftarrow A + d + CF$ | 2 |
| Directo | ADC (dd) | $A \not\leftarrow A + (dd) + CF$ | 3 |
| Indexado | ADC (IX+d) | $A \not\leftarrow A + (IX+d) + CF$ | 2 |

- b) **Instrucciones de resta** : Estas instrucciones realizan por medio de la ULA, la resta aritmética de dos operandos. Como consecuencia de ello, el registro de estado se ve modificado como sigue:

| | | | | | | | |
|-----------|----|---|---|---|---|---|---|
| Indicador | I | N | H | V | S | C | Z |
| Estado | nc | 1 | c | c | c | c | c |

Son posibles dos tipos de instrucciones de resta: con carry y sin carry.

| Direccionamiento | Sintaxis | Función | Longitud (bytes) |
|------------------|------------|------------------------------------|------------------|
| Implícito | SUB A,r | $A \not\leftarrow A - r$ | 1 |
| Inmediato | SUB d | $A \not\leftarrow A - d$ | 2 |
| Directo | SUB (dd) | $A \not\leftarrow A - (dd)$ | 3 |
| Indexado | SUB (IX+d) | $A \not\leftarrow A - (IX+d)$ | 2 |
| Implícito | SBC A,r | $A \not\leftarrow A - r + CF$ | 1 |
| Inmediato | SBC d | $A \not\leftarrow A - d + CF$ | 2 |
| Directo | SBC (dd) | $A \not\leftarrow A - (dd) + CF$ | 3 |
| Indexado | SBC (IX+d) | $A \not\leftarrow A - (IX+d) + CF$ | 2 |

- c) **Instrucciones de incremento y decremento** : Por medio de estas instrucciones podemos incrementar y decrementar el contenido de cualquiera de los registros de 8 bits o de una posición de la memoria. El resultado se deja en el mismo registro o posición de la memoria indicada. El registro de estado se ve modificado por la ejecución de una de estas instrucciones como se indica a continuación:

| | | | | | | | |
|-----------|----|---|---|---|---|----|---|
| Indicador | I | N | H | V | S | C | Z |
| Estado | nc | 1 | c | c | c | nc | c |

| Direccionamiento | Sintaxis | Función | Longitud (bytes) |
|------------------|------------|------------------------------------|------------------|
| Implícito | INC r | $r \not\leftarrow r + 1$ | 1 |
| Directo | INC (dd) | $(dd) \not\leftarrow (dd) + 1$ | 3 |
| Indexado | INC (IX+d) | $(IX+d) \not\leftarrow (IX+d) + 1$ | 2 |
| Implícito | DEC r | $r \not\leftarrow r - 1$ | 1 |
| Directo | DEC (dd) | $(dd) \not\leftarrow (dd) - 1$ | 3 |
| Indexado | DEC (IX+d) | $(IX+d) \not\leftarrow (IX+d) - 1$ | 2 |

5.3.4.- Instrucciones de salto

Este conjunto de instrucciones permiten que un programa tenga **continuidad lógica** aunque no exista la continuidad física. Resulta casi imposible poder disponer de un programa con una continuidad física completa a lo largo de este, ya que lo normal es que en algún momento, por necesidades de la función a realizar, sea necesario **tomar una decisión** dependiente de algún parámetro. La toma de decisión implica, que existan dos posibilidades y haya que elegir una de ellas. Como mucho, una podría mantener la continuidad física, pero para la otra es imposible.

Esto hace que este conjunto de instrucciones sea de las más utilizadas y que los diseñadores de UCPs se esfuercen en potenciarlas. Hay varios tipos de salto:

- /// **Largos**
- /// **Cortos**
- /// **Absolutos**
- /// **Relativos**
- /// **Incondicionales**
- /// **Condicionales**
- /// **A subrutinas**
- /// **Retornos**

Salto corto : Se denominan así los saltos cuya *longitud está limitada* a un determinado rango de direcciones. En los CPUs de 8 bits, este tipo de salto utiliza un solo byte para dar o calcular la dirección de salto con lo que la longitud máxima de salto es de 256 posiciones.

Salto largo : Al contrario de los saltos cortos, éstos permiten alcanzar la *totalidad de las direcciones* posibles de la memoria. Por lo tanto, la instrucción completa constará del código de operación y de un operando que será la dirección completa del salto o una referencia a él, según sea el modo de direccionamiento seleccionado. En el caso de direccionamiento directo para nuestro modelo de UCP, la instrucción se compone de tres bytes, uno de código de operación y dos de operando (dirección de destino del salto).

Salto absoluto : En los saltos absolutos, la dirección a la que hay que saltar no tiene una relación directa con la posición donde se encuentra la instrucción del salto. La dirección del salto se especifica por diferentes métodos, según sea el direccionamiento de la instrucción de salto.

Salto relativo : En los saltos relativos la dirección a la que hay que saltar se calcula a partir de la posición donde se encuentra la instrucción del salto, es decir, del contenido del contador de programa. El salto puede ser *hacia adelante o hacia atrás*, siempre respecto a la referencia. La dirección a la que hay que saltar se calcula sumando o restando un valor denominado *desplazamiento (offset)* al contenido del PC. Para realizar esa operación, el desplazamiento se da en **complemento a dos**: de esta forma se simplifica el cálculo de la dirección a saltar.

Hemos de tener en cuenta que en el momento en que la UCP va a realizar el cálculo de la dirección a la que ha de saltar, el contador de programa se encuentra incrementado en varias unidades desde que se comenzó el fetch de la instrucción de salto relativo. Concretamente, en nuestro modelo, las diferentes instrucciones de salto con direccionamiento relativo constan de dos bytes, el código y el desplazamiento. Por lo tanto, el contador de programa se encontrará **incrementado en dos** respecto al comienzo de la instrucción.

La distancia máxima posible en este tipo de salto depende del tamaño del desplazamiento. En las UCPs de 8 bits, este desplazamiento es de 8 bits. En este caso, la instrucción de salto relativo ocupa dos bytes, uno para el código y otro para el desplazamiento. Si la dirección de la instrucción de salto relativo es n , la instrucción siguiente se encuentra en la dirección $n+2$.

Por otro lado, la representación en complemento a dos con 8 bits nos permite un **desplazamiento máximo de +127 a -127**. Todo esto hace que el desplazamiento real sea diferente si es en avance o retroceso. Cuando es en avance, el desplazamiento máximo es de $127+2=129$ posiciones de memoria, ya que el cálculo se realiza desde la dirección de la instrucción siguiente. En cambio si es en retroceso, el desplazamiento máximo es de $127-2=125$ posiciones.

Los saltos relativos son muy interesantes en programas que se han de ejecutar en posiciones de memoria no conocidas de antemano ("**reubicabilidad**" o en inglés *relocatibily*, se dice que el programa es reubicable), ya que, al no depender el salto de la posición absoluta, el programa se puede situar en cualquier posición de la memoria.

Saltos incondicionales : Son los saltos que se realizan sin necesidad de consultar ningún parámetro condicionante. Es el más simple de los saltos y puede ser corto, largo, absoluto o relativo.

Saltos condicionales : La ejecución de las instrucciones de salto condicionales depende de que se cumpla la condición del salto. Los **condicionantes** de estos saltos son los bits del registro de estado de la UCP, que según estén activados o no se producirá el salto o no, según indique la instrucción. Los saltos condicionales también pueden ser largos, cortos, absolutos o relativos.

Saltos a subrutinas : A lo largo de un programa nos encontramos habitualmente **funciones que se repiten** de forma necesaria. Por ejemplo, la función de conversión de código hexadecimal a ascii es una de las más usuales para poder presentar los datos por una pantalla. Cuando esto sucede, la función se desarrolla en un **trozo de programa** que se llama subrutina que tiene la particularidad de ser un recurso común al resto del programa. Es decir, puede ser utilizada desde cualquier punto del programa.

Las subrutinas tienen una estructura determinada que no entraremos a discutir en este apartado ya que corresponde al software, pero sí es necesario indicar que dispone de una **entrada y una salida**. Por la entrada se llega a la subrutina con los datos que han de ser tratados por ella y una vez tratados, por la salida obtenemos el resultado.

La forma de llegar a la entrada de una subrutina es por medio de una instrucción de **llamada a subrutina** (CALL) que es un tipo particular de salto. Cuando ejecutamos una instrucción CALL, la UCP desarrolla una secuencia especial (como consecuencia de la decodificación del código de operación) que consta de dos pasos principales: **guarda el contenido del contador de programa** en la pila y **salta a la dirección indicada**. Con ello se consigue salvar la dirección de la siguiente instrucción a ejecutar después de la llamada para continuar con el programa. La posición de la pila en que se introduce la dirección de retorno está controlada por el puntero de pila (SP) que lo maneja de forma automática la UCP.

A continuación, la subrutina toma el control y ejecuta su actividad. La subrutina dispone de una instrucción especial al final de ella que permite a la UCP continuar en el punto en que llamó a la subrutina. Se trata de una **instrucción de retorno (RETURN)** cuya función es inversa a la de llamada en cuanto que obtiene de la pila la dirección que introdujo la llamada y ejecuta un salto a esta dirección.

Retornos : Las instrucciones de retorno de subrutina son las complementarias de las de salto a subrutinas. Ellas se encargan de recuperar la dirección que se guardó al hacer el salto a la subrutina en el stack y continuar el programa por donde iba.

Otra instrucción de retorno es la de **retorno de interrupción**. El funcionamiento básico es el mismo, salvo que se utiliza cuando el salto se produce por causa de una interrupción. La diferencia más importante es que el retorno de interrupción *restaura el valor de la máscara de interrupción*. Ninguna instrucción de salto excepto el retorno de interrupción modifica el contenido del registro de estado de la UCP. De todas las posibilidades para las instrucciones de salto en la UCP nos quedamos con las siguientes:

a) Instrucciones de salto incondicional.

| Direccionamiento | Sintaxis | Función | Longitud (bytes) |
|------------------|----------|------------------------|------------------|
| Directo | JP dd | PC \leftarrow dd | 3 |
| Relativo | JR d | PC \leftarrow PC + d | 2 |

b) Salto condicional : Las condiciones para los saltos son en función de los **bits Z y C del registro de estado**. Disponemos de condiciones para la activación y la no activación de ambos bits.

| Direccionamiento | Sintaxis | Función | Longitud (bytes) |
|------------------|----------|------------------------------|------------------|
| Inmediato | JP C,dd | Si C=1, PC \leftarrow dd | 3 |
| Inmediato | JP NC,dd | Si C=0, PC \leftarrow dd | 3 |
| Inmediato | JP Z,dd | Si Z=1, PC \leftarrow dd | 3 |
| Inmediato | JP NZ,dd | Si Z=0, PC \leftarrow dd | 3 |
| Relativo | JR C,d | Si C=1, PC \leftarrow PC+d | 2 |
| Relativo | JR NC,d | Si C=0, PC \leftarrow PC+d | 2 |
| Relativo | JR Z,d | Si Z=1, PC \leftarrow PC+d | 2 |
| Relativo | JR NZ,d | Si Z=0, PC \leftarrow PC+d | 2 |

c) Instrucciones de llamada a subrutina.

| Direccionamiento | Sintaxis | Función | Longitud (bytes) |
|------------------|----------|--|------------------|
| Inmediato | CALL dd | (SP) \leftarrow PCH (SP+1) \leftarrow PCL PC \leftarrow dd | 3 |

d) Instrucciones de retorno.

| Direccionamiento | Sintaxis | Función | Longitud (bytes) |
|------------------|----------|---|------------------|
| Implícito | RET | PCL \leftarrow (SP+1) PCH \leftarrow (SP+2) I \leftarrow I' | 1 |
| Implícito | RETI | PCL \leftarrow (SP+1) PCH \leftarrow (SP+2) | 1 |

5.3.5.- Instrucciones de control

Estas son instrucciones que se utilizan para manejar las posibilidades de *programación de la UCP*. Entre ellas tenemos las que manejan los bits del registro de estado, seleccionan o no la entrada de interrupción, etc.

a) Instrucciones de manejo del bit de carry : Disponemos de una instrucción para activar directamente el bit de carry del registro de estado. Para desactivarlo se puede utilizar, por ejemplo, la instrucción OR A.

| Direccionamiento | Sintaxis | Función | Longitud (bytes) |
|------------------|----------|------------|------------------|
| Implícito | SCF | $C \neq 1$ | 1 |

- b) **Instrucciones de manejo de la máscara de interrupción** : Por medio de estas instrucciones podemos activar y desactivar la máscara de la interrupción.

| Direccionamiento | Sintaxis | Función | Longitud (bytes) |
|------------------|----------|------------|------------------|
| Implícito | EI | $I \neq 1$ | 1 |
| Implícito | DI | $I \neq 0$ | 1 |

- c) **Instrucción de parada** : Por medio de la instrucción HALT podemos detener la ejecución de la UCP en espera de la llegada de una interrupción.

| Direccionamiento | Sintaxis | Función | Longitud (bytes) |
|------------------|----------|---------|------------------|
| Implícito | HALT | Parada | 1 |

- d) **Instrucción de no hacer nada** : En ocasiones es necesario disponer de una instrucción que haga correr un ciclo máquina sin afectar más que al contenido del PC que se incrementa en una unidad.

| Direccionamiento | Sintaxis | Función | Longitud (bytes) |
|------------------|----------|----------------|------------------|
| Implícito | NOP | $PC \neq PC+1$ | 1 |

5.3.6.- Instrucciones de Entrada/Salida

Son las encargadas de manejar el *mapa de direcciones de E/S* y efectúan las operaciones de entrada y salida de datos a y desde los periféricos.

- a) **Instrucciones de entrada** : Por medio de estas instrucciones tenemos acceso a los registros de los periféricos de entrada que estén situados en el mapa de E/S de la UCP.

| Direccionamiento | Sintaxis | Función | Longitud (bytes) |
|------------------|----------|--------------|------------------|
| Inmediato | IN A,d | $A \neq (d)$ | 2 |

- b) **Instrucciones de salida** : Por medio de estas instrucciones tenemos acceso a los registros de los periféricos de salida que estén situados en el mapa de E/S de la UCP.

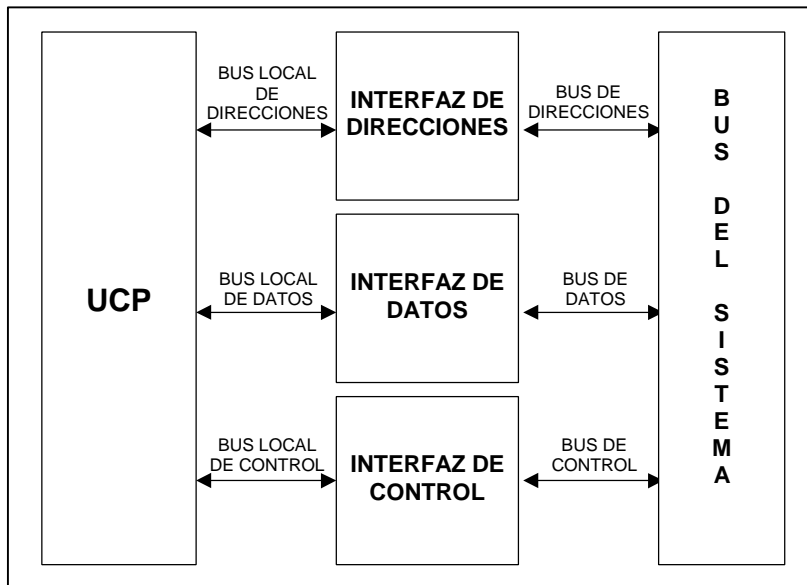
| Direccionamiento | Sintaxis | Función | Longitud (bytes) |
|------------------|-----------|--------------|------------------|
| Inmediato | OUT (d),A | $(d) \neq A$ | 2 |

5.3.7.- Otras instrucciones

Algunos autores incluyen en este grupo instrucciones como NOP que ya hemos citado. Conviene observar que el total de instrucciones diferentes discutidas es de 24 y, sin embargo, el total de códigos diferentes es de 172. Esto es debido a los *distintos modos de direccionamiento* que permiten algunas instrucciones, que hacen que estas sean realmente diferentes. Observar, así mismo, que quedan aún 84 posibles códigos de nuestro microprocesador sin utilizar ($256 - 172 = 84$).

6.- Interfaz con el bus del sistema.

La UCP presenta un conjunto de señales a las que hemos denominado buses y que ahora renombramos como **BUS LOCAL DE LA UCP** para diferenciarlo del bus del sistema. En muchas ocasiones nos encontraremos que el bus del sistema y el bus local son la misma cosa, tienen la misma definición. La circuitería que adapta el bus local de la UCP al bus del sistema es lo que denominamos **interfaz de la UCP** con el bus del sistema. En general, el interfaz de la UCP con el bus del sistema se compone de tres partes:



≡≡ **Interfaz de direcciones.**

≡≡ **Interfaz de datos.**

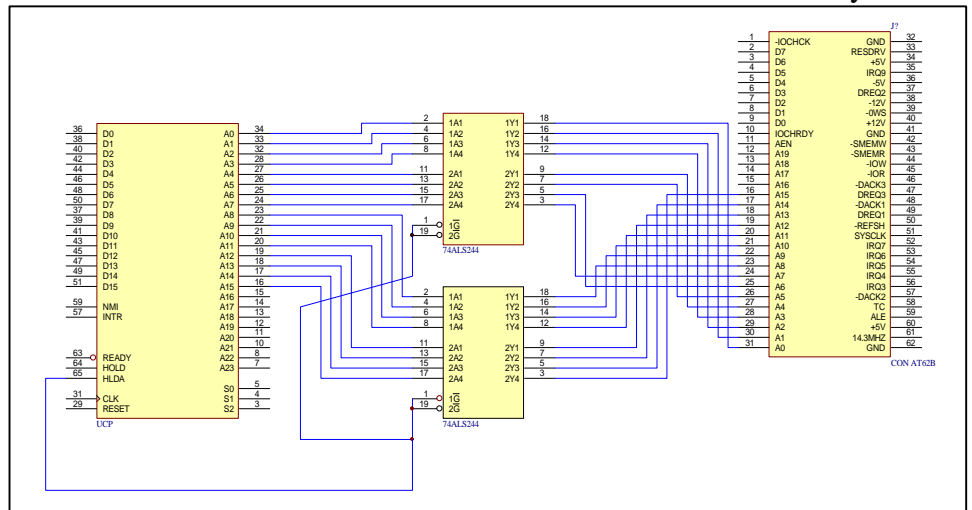
≡≡ **Interfaz de control.**

Interfaz de direcciones :

El interfaz del bus de direcciones tiene como misión adaptar las características del bus local de direcciones al del sistema. Este interfaz está compuesto por **dispositivos transmisores unidireccionales** que pueden ser latcheados o no y con salida triestado. El bus de direcciones del sistema es un

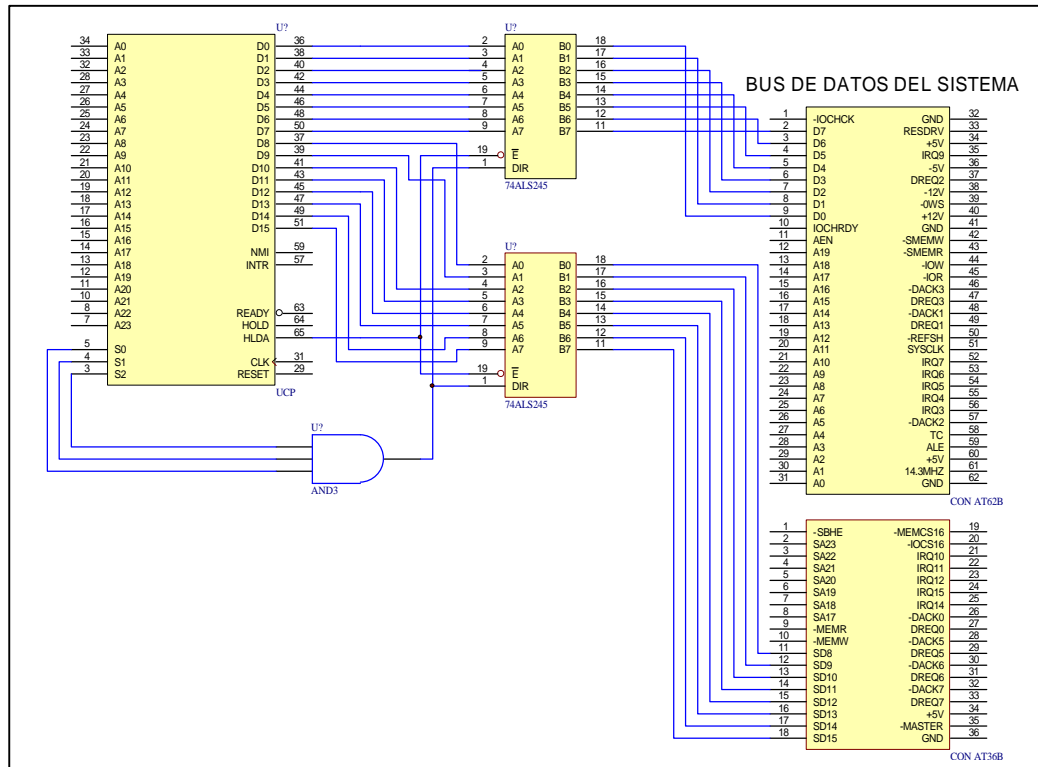
bus no multiplexado y, por tanto, la conexión de estos transmisores es directa, se conectan de forma que la entrada es el bus de direcciones del bus local mientras que la salida se conecta al bus de direcciones del sistema.

El interfaz del bus de direcciones tiene como finalidad el aislamiento entre el bus local y el del sistema, evitando de esta forma posibles averías de la UCP por choques de señales (**"bus contention"**), cortocircuitos, etc. A su vez, estos transmisores aumentan el fan-out de las señales de acuerdo a las especificaciones del bus.



Básicamente requieren *dos señales de control*, cuando no son registrados, más una señal de reloj en caso de ser registrados.

Las dos señales comunes a ambos controlan el *sentido de la información* (hacia el bus local de la UCP o hacia el bus del sistema) una de ella (**DIR**) mientras que la otra controla el *triestado de salida* del transceptor (**/G**). La figura siguiente muestra una conexión típica de estos transceptores.



Interfaz de control : Con frecuencia, el bus de control del bus del sistema en el que se encuentra ubicada la UCP no coincide con el bus de control del bus local de la UCP, no tanto por las diferencias de *características eléctricas* (fan-out, niveles, etc) como por la propia *definición de señales* en ambos buses de control. Concretamente, el bus de nuestro sistema modelo requiere de señales que no existen en el bus de control de la UCP. Este interfaz del bus de control tiene como misión generar las señales no presentes y adaptar las concordantes.

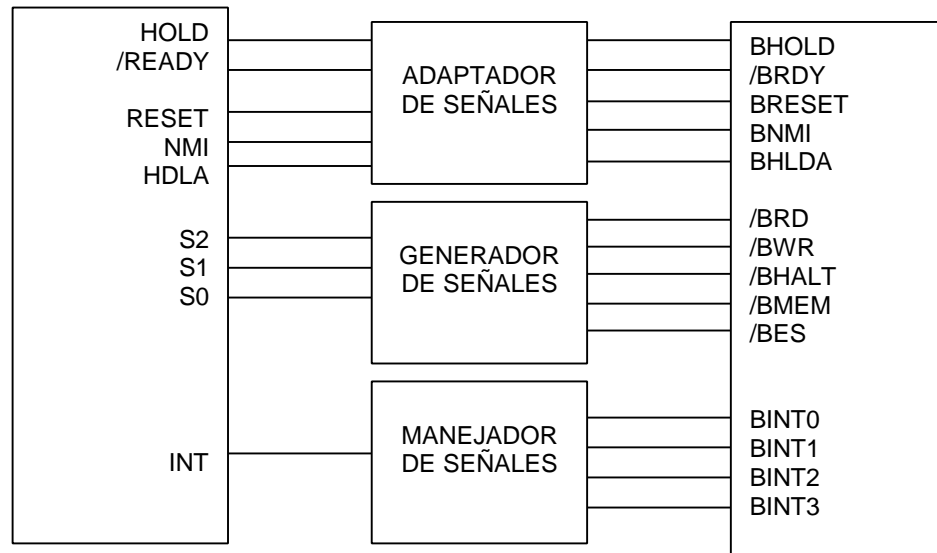
Este interfaz lo podemos dividir en dos partes principales:

- ✍️ **Adaptación.**
- ✍️ **Generación.**
- ✍️ **Manejo.**

La **adaptación de señales** es un proceso similar al que realiza el interfaz del bus de direcciones, utilizando transmisores si las señales son unidireccionales o transceptores si se trata de señales bidireccionales. La figura siguiente muestra esta conexión en donde podemos ver que las señales HOLD, HLDA se corresponden con esta parte del interfaz.

Podemos ver que las señales /RD, /WR, /MEM, /ES, /HALT se obtienen del bloque generador cuyas entradas son las señales S2, S1, S0 procedentes del bus local. El **bloque generador** puede ser un sistema combinacional o secuencial, según se necesite.

La parte de **manejo de señales** implica directamente a las interrupciones enmascarables procedentes del bus del sistema hacia la UCP. La UCP dispone en general de una entrada de interrupción enmascarable mientras que en el bus del sistema pueden existir varias (4 en nuestro modelo de bus de sistema) como podemos ver en la figura. Todas ellas han de ser atendidas por la UCP. En este caso el manejador es el chip **controlador de interrupciones** (PIC).



7.- Interrupciones.

La forma de que el sistema atienda de **forma rápida** a los elementos externos es por medio de las interrupciones del sistema. La UCP abandona de forma momentánea la ejecución del programa en curso para atender la rutina de interrupción. Para explicar los distintos mecanismos existentes de manejo y control de las interrupciones es necesario conocer los cuatro conceptos siguientes:

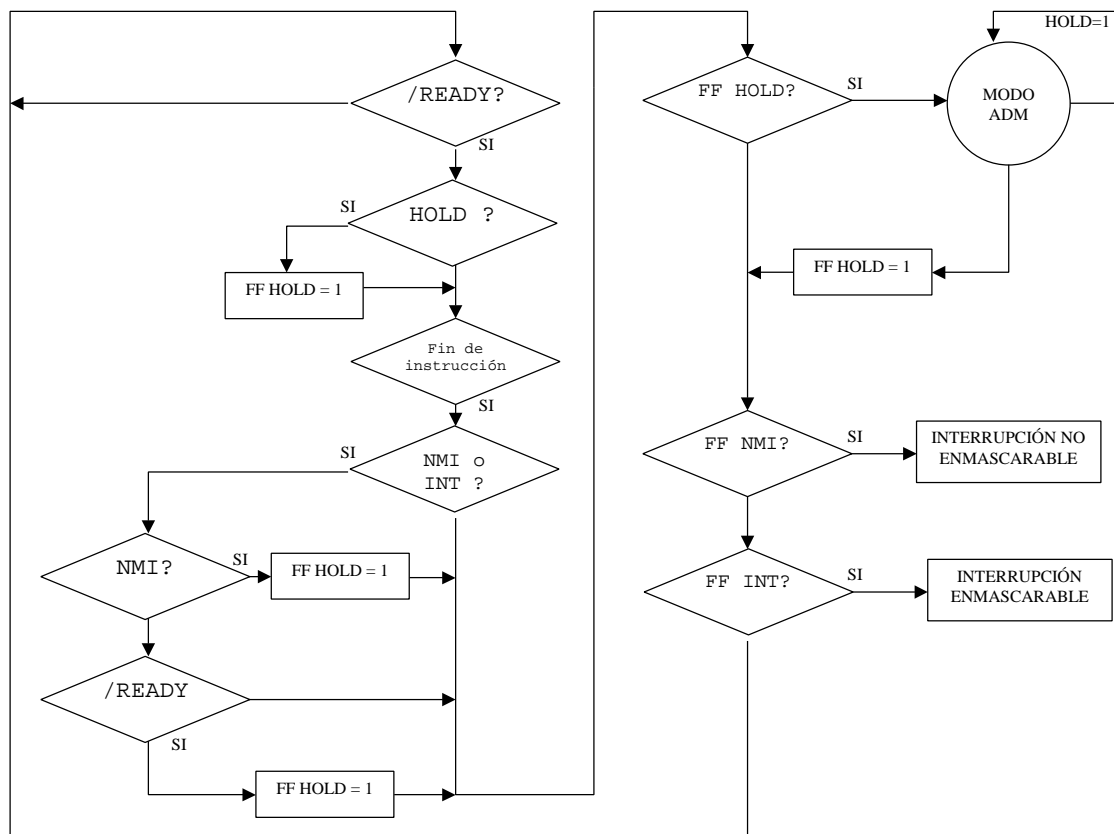
- **Máscara:** Registro interno de la UCP que permite o no que una interrupción enmascarable sea aceptada (reconocida). Cuando la máscara está activada la UCP no recibe la interrupción (no la reconoce). Si está desactivada sí la reconoce.
- **Vector de interrupción:** Información volcada al bus de datos durante el ciclo de reconocimiento de interrupción que sirve para que la UCP pueda localizar la fuente de la interrupción y proceder a atenderla correctamente.
- **Prioridad:** Preferencia en el tiempo en la atención a la interrupción de un determinado dispositivo cuando coinciden varias peticiones de interrupción.
- **Anidamiento:** Es la posibilidad de que la ejecución del programa que atiende una interrupción (rutina de interrupción) se vea interrumpida por la llegada de otra interrupción más prioritaria. La viabilidad del anidamiento supone que las rutinas de interrupción que lo permiten desactivan la máscara de la interrupción de la UCP, ya que por defecto, la máscara se activa automáticamente al aceptar una interrupción.

7.1.- Tipos de interrupciones.

En los sistemas basados en microprocesadores se distinguen **dos tipos de interrupciones**; la interrupción que obliga al sistema a tenerla en cuenta y, por tanto, atenderla con la máxima urgencia y la que tiene ciertas tolerancias que permiten a la UCP no abandonar inmediatamente su trabajo para atenderla. La primera se llama **interrupción no enmascarable** y la segunda **interrupción enmascarable**. Estos nombres hacen referencia al bit de máscara de interrupción que dispone la UCP en su interior. La máscara de interrupción solo es válida para la interrupción enmascarable.

Prioridad de las interrupciones : Las entradas de interrupción a la UCP son **señales asíncronas** respecto al reloj de ésta. Las interrupciones desvían la atención de la UCP hacia rutinas concretas de atención a ellas parando el proceso en curso. La señal HOLD es también una señal asíncrona a la UCP pero no se trata como una interrupción. HOLD no requiere de ninguna rutina de atención.

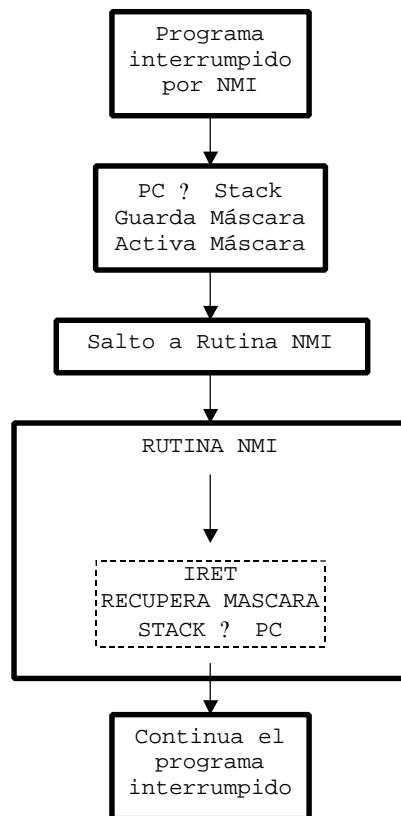
HOLD congela directamente el funcionamiento de la UCP una vez terminado el ciclo máquina en curso. Por tanto, mientras la UCP esté en hold, **no será atendida ninguna interrupción**. La figura siguiente muestra la secuencia que sigue la UCP en cada ciclo máquina para comprobar el estado de las señales de entrada de HOLD, INT y NMI. En ella podemos observar que una vez llegado el final del ciclo máquina (/READY activo) la UCP investiga el estado de la señal HOLD y si la encuentra activa entra directamente al modo HOLD independientemente de si existe o no petición de interrupción NMI o INT. Si hay alguna petición activa la memoriza en un biestable de **interrupción pendiente**. Cuando la UCP no está en HOLD se comprueba la existencia de alguna interrupción pendiente de atender y si existe, la UCP la atiende.



Conviene insistir en que en orden de **prioridades de las señales** dentro de la UCP, la señal HOLD es la más prioritaria, a continuación le sigue la NMI y, por último la INT.

Interrupción no enmascarable (NMI) : La llegada de una NMI al μP implica de modo automático que el μP ha de:

- a) **Terminar la instrucción en curso.**
- b) **Atender la interrupción.**



La NMI no tiene más espera que la finalización de la instrucción en curso. La figura muestra la secuencia funcional que desarrolla el μP a la aceptación de una **interrupción no enmascarable NMI**. La atención a la interrupción supone la existencia de una rutina que se encarga de atender esta interrupción. Esta rutina está localizada en una determinada posición de la memoria fijada por el fabricante (o sea es una **interrupción autovectorizada**) y a la que accede la CPU mediante un salto a ella.

Para poder mantener la **coherencia lógica** del programa y poder continuar el trabajo en el punto en que fue interrumpido, el μP desarrolla, a la llegada de la NMI, una secuencia similar a la desarrollada en la ejecución de la instrucción CALL. Cuando la UCP recibe una NMI y termina con la instrucción en curso, guarda la dirección contenida en el PC en la pila como **dirección de retorno**, activa la máscara de interrupción y salta a la dirección de la rutina NMI.

La UCP **impide automáticamente** que pueda atenderse otra interrupción enmascarable durante la atención a la NMI ya que activa la máscara de interrupción. Esta acción es normal en las funciones del HW a la llegada de una interrupción cualquiera. Si se desea permitir la entrada de interrupciones, ha de ser el programa quien lo autorice desactivando la máscara.

La rutina de atención a la NMI, una vez que termina de realizar su función puede retomar el programa que dejó al llegar la NMI en el punto en donde fue interrumpido ejecutando una **instrucción de retorno de interrupción**. El retorno de interrupción de NMI recupera la dirección de retorno desde la pila reponiendo la máscara de la interrupción a su valor original anterior al salto a la rutina de tratamiento de la NMI. La interrupción NMI, por su carácter prioritario, se utiliza para **casos de alarmas graves y emergencias**, como puede ser la aparición de errores en la memoria o fallo en la alimentación del sistema.

La señal NMI es una **entrada asíncrona** respecto al reloj de la UCP pero ha de cumplir con los tiempos de setup y hold respecto al flanco de subida de CLK para que la UCP sea capaz de reconocerla. Por lo que se deduce que el retardo máximo que sufre una NMI desde su llegada a su atención es el **tiempo del ciclo de instrucción más largo** (incluyendo los estados de espera correspondientes si los hay).

La NMI **no será atendida** si la UCP está en hold ya que mientras la señal HOLD se encuentre activa, la UCP no es operativa.

Interrupción enmascarable (INT) : La interrupción más frecuentemente utilizada en los sistemas basados en microprocesadores es la interrupción enmascarable ya que el programa que se ejecuta en el sistema puede controlar por medio del **bit de máscara** la entrada de esta interrupción. En lo que sigue utilizaremos la palabra interrupción (simplemente interrupción) para referirnos sin más a la interrupción enmascarable, mientras que cuando hagamos referencia a la NMI especificaremos que se trata de una interrupción no enmascarable.

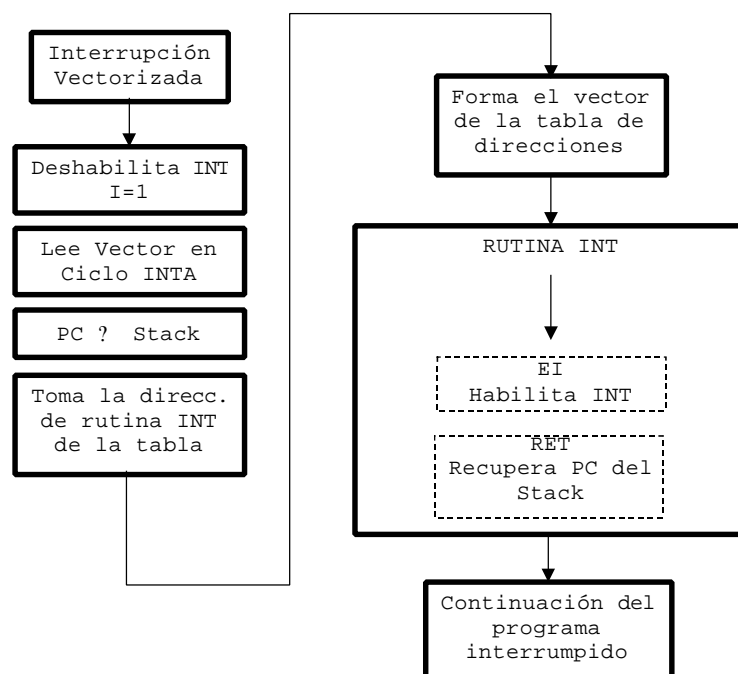
La aceptación de una interrupción por la UCP significa obligatoriamente (como para la NMI) la existencia de un programa de atención a esta interrupción (*rutina de interrupción*). Dependiendo del **método utilizado** para localizar esta rutina de interrupción, las interrupciones pueden ser de tres tipos o modos diferentes:

☞☞ **Vectorizadas.**

☞☞ **Autovectorizadas.**

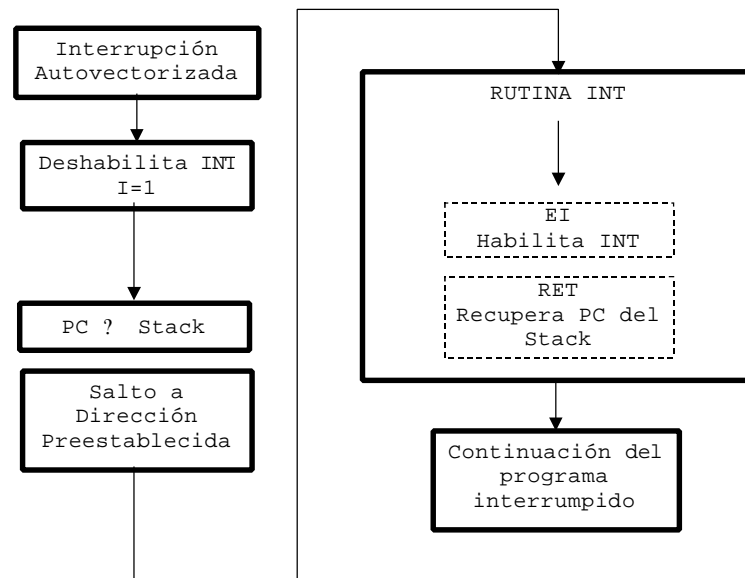
☞☞ **No vectorizada.**

Interrupciones Vectorizadas : En este caso, el dispositivo que interrumpe **suministra la dirección de una tabla** de entrada a las rutinas de interrupción. La UCP obtiene de la dirección dada la dirección real de la rutina de interrupción. El vector volcado al bus puede suministrar la dirección completa o solamente parte de ella, en este último caso, un registro interno de la UCP suministra el resto de la dirección. La figura siguiente muestra la secuencia en este tipo de interrupción.

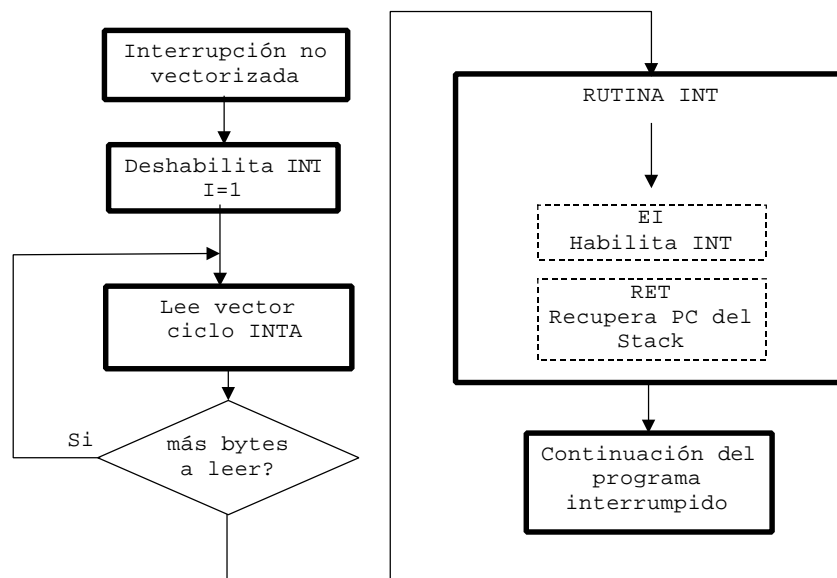


Interrupciones Autovectorizadas : En este caso, la UCP **no necesita de vector externo** para localizar la rutina de interrupción. La rutina de interrupción se encuentra en una dirección fija conocida por la UCP, por lo que directamente, al aceptar la interrupción, la UCP

salta a esta dirección comenzando en ese punto la ejecución de la rutina. La figura siguiente muestra la secuencia desarrollada en este modo de interrupción.



Interrupciones no vectorizadas : La característica de este modo de interrupción es que la UCP *recoge del bus de datos, durante el ciclo de reconocimiento de interrupción, una información (no vector) que introduce en el registro de instrucción y lo decodifica como si se tratara de un código de operación* ejecutándolo a continuación. Este código lo vuelca al bus de datos el dispositivo que genera la interrupción durante el ciclo de reconocimiento de ésta. Este código corresponde generalmente a una **instrucción de salto** a la rutina de interrupción. Esta instrucción puede ser de uno o varios bytes. Si es de un byte, al finalizar el ciclo INTA la UCP salta inmediatamente a la rutina. Si se trata de una instrucción de más de un byte, la UCP leerá los restantes bytes hasta completar la dirección de salto, y a continuación salta.



8.- El PC de IBM.

Aunque el diseño original del IBM PC está *algo anticuado* pues data del año 1982, demuestra bastante bien los conceptos de DMA, Interfaz de interrupciones y otros que vamos a estudiar aquí. Casi todos nuestros comentarios de la operación del PC también resultan válidos para un PC/AT y posterior, y como veremos luego, un conocimiento del PC básico es un buen punto de partida para entender sistemas más modernos de última generación.

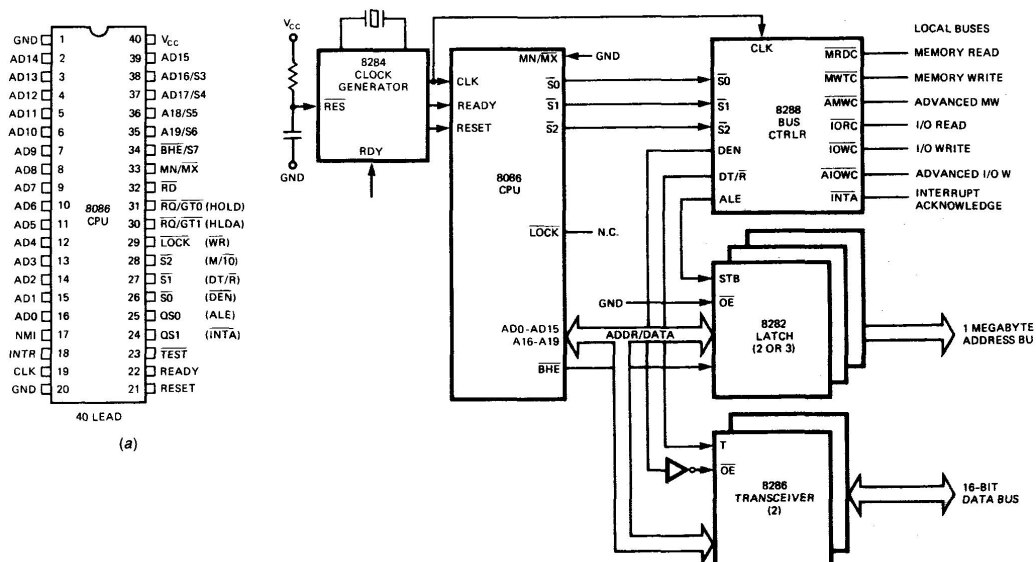
En la figura de la página siguiente podemos observar un diagrama de bloques de la circuitería de la placa base de un IBM PC. Empezando por la izquierda del diagrama pasamos por la CPU 8088 y el controlador de interrupciones 8259A. Bajo el procesador principal 8088 vemos el procesador matemático auxiliar 8087.

La siguiente línea vertical de dispositivos a la derecha consiste en los buffers del bus de direcciones, los del bus de datos, y el chip controlador de bus 8288. Se necesita un chip controlador de bus para generar las señales de control de bus a partir de las señales S0 a S2 del 8088. Los buses de estos dispositivos atraviesan el dibujo y van a parar a conectores de las *placas periféricas* de 62 patillas (BUS ISA en versión 8 bits) de forma que el 8088 se comunique con las tarjetas de expansión así como con la ROM, RAM y puertos de la placa madre. Esto es lo que hace que el sistema sea suficiente versátil para todas las aplicaciones.

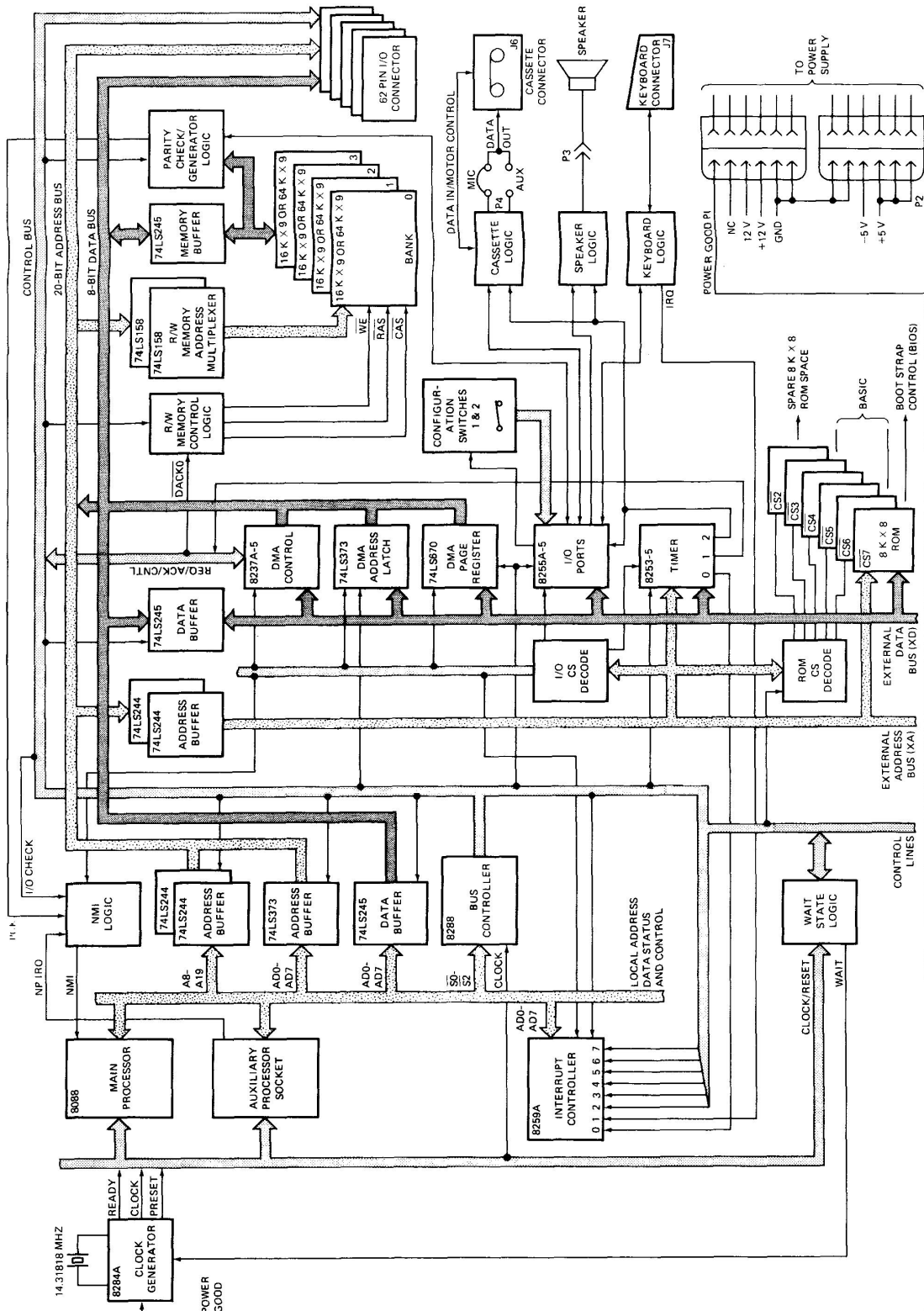
En el centro, abajo localizamos la memoria ROM, la lógica que controla el teclado, etc, en medio a la derecha y la **RAM dinámica** arriba, a la derecha. Finalmente, observamos la columna de dispositivos que contiene el **controlador DMA 8237A-5**. Comenzando por la parte inferior de esta columna vemos un **temporizador programable 8253-5**. Justo encima tenemos el dispositivo **controlador de puertos programable 8255A-5**.

8.1.- La CPU 8086.

La figura siguiente muestra el diagrama de patillas del 8086. Si la patilla 33, MN/MX se encuentra a nivel bajo, el 8086 opera en **modo máximo** y las patillas 24 a 31 generan las señales que se encuentran cerca de las patillas en la figura anterior. En el modo máximo, las señales de control del bus se mandan hacia fuera de forma codificada sobre las líneas de estado S0, S1 y S2.

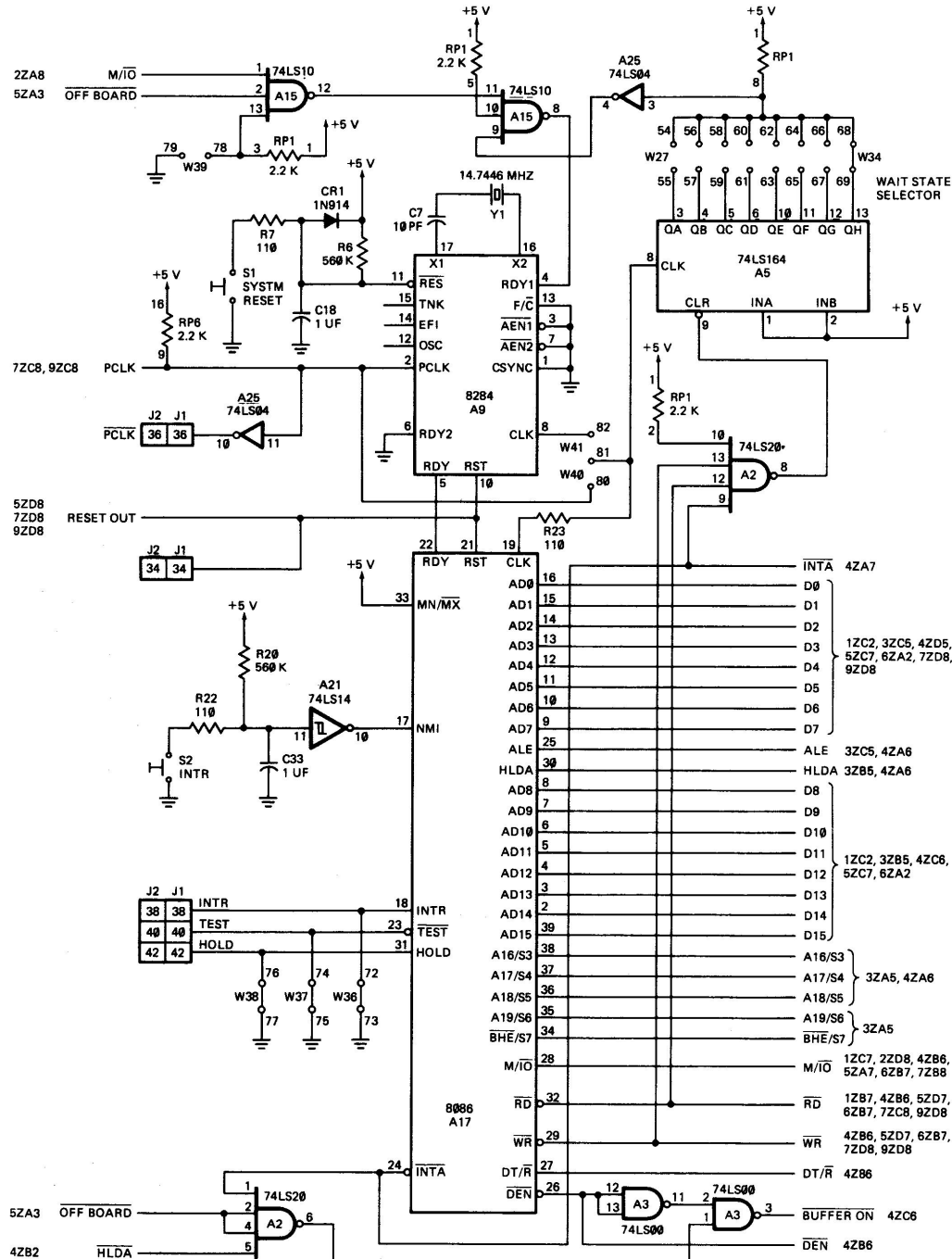


Un dispositivo controlador externo como el Intel 8288 se emplea para generar las señales de control del bus necesarias a partir de estas líneas. La figura muestra los nombres expandidos para cada una de las señales de control del bus generadas por el 8288. Se emplean ***latches óctuples*** 8282 (equivalentes al 74LS374 del tema 6) para demultiplexar las señales de direcciones y ***transceptores bidireccionales*** 8286 (equivalentes al 74LS245) para amplificar el bus de datos de forma que pueda controlar muchos dispositivos.



8.2.- *Generador de Reloj y control del bus.*

El primer elemento que nos encontramos en el sistema es el generador de reloj i8284. Como se observa en la figura siguiente este dispositivo tiene un cristal de cuarzo de 14.7456 conectado.

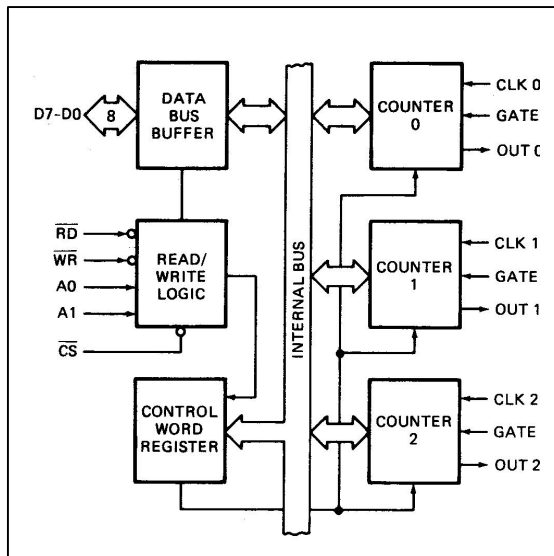


La frecuencia de este cristal *se divide por 3 para producir la señal de reloj* que se manda al 8086. Por tanto, la frecuencia de reloj de funcionamiento es de 4.915 MHz. Otras señal de reloj denominada PCLK, producida también por este dispositivo, tiene una frecuencia mitad de la de reloj, 2.54 MHz. Esta señal se emplea como señal de reloj de propósito general en el sistema. La señal hardware RST y la RDY también pasan a través del 8284 para *sincronizarla con la señal de reloj* antes de enviarse al 8086.

Como observamos en la figura, una gran cantidad de circuitería se emplea para poder insertar varios *tiempos de espera* en el ciclo máquina mediante un registro de desplazamiento que retarda la activación de la señal RDY1.

8.3.- Temporizador i8254 y modos de operación.

El Intel 8254 contiene *tres contadores de 16 bits* que pueden ser programados para trabajar en varios modos diferentes, como muestra el diagrama de bloques. La gran ventaja de estos contadores frente a los contadores revisados en el tema 4, es que podemos cargar una cuenta en ellos, inicializarlos y pararlos con instrucciones del programa. Estos dispositivos se dicen que son *programables por software*. Para programarlo, mandamos los bytes de cuenta y de control como si los enviáramos a cualquier puerto de E/S.



Si observamos a la izquierda del diagrama de bloques, veremos líneas que hacen de interfaz con el bus del sistema. Dispone de una entrada /CS que se activará mediante un decodificador de direcciones cuando se acceda al dispositivo, y dispone de *dos entradas de dirección*, A0 y A1, para permitir el direccionamiento de los tres contadores o el registro de control del dispositivo.

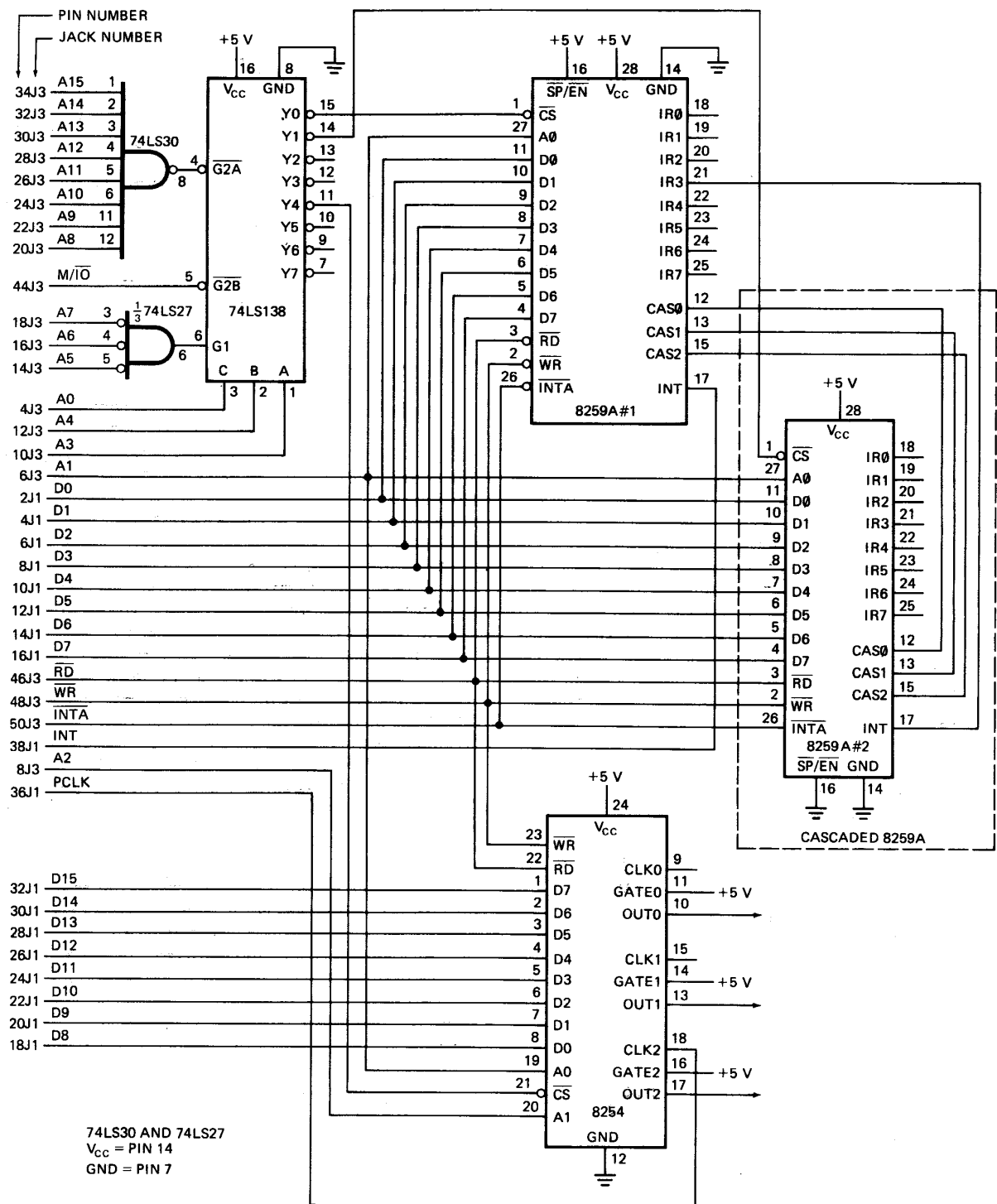
La *entrada GATE* de cada contador nos permite arrancar o parar el contador con una señal hardware externa. Si la entrada GATE está a "1", el contador está habilitado, si la entrada está a "0", el contador está deshabilitado. La señal de salida de cada contador aparece en la patilla OUT.

8.3.1.- Conexión al sistema.

La figura siguiente muestra las conexiones para añadir un contador 8254 y un PIC 8259 a un sistema basado en el 8086. El 74LS138 se emplea para producir señales /CS para el 8254, el 8259 y cualquier otro dispositivo E/S que se quiera añadir. Para que cualquier señal de salida de este componente se active, las entradas de habilitación G1, /G2A y /G2B deben activarse simultáneamente. Esto tiene lugar, como muestra la interfaz *hardware-software* para la dirección base FF00H.

Solo una de las salidas del decodificador puede encontrarse activa. Conectaremos la señal de dirección A0 a la entrada C para que la mitad de las salidas del decodificador corresponda a direcciones pares y la otra mitad a direcciones impares. Hacemos esto porque los *dispositivos son de 8 bits y el bus de datos de 16 bits*, por lo que conectaremos un dispositivo de E/S a la parte baja del bus de datos y el otro a la parte alta del bus de datos. Los datos de direcciones pares aparecen con A0="0" en el bus bajo de datos D0..D7 y los datos de direcciones impares con A0="1" aparecen en el bus alto de datos D8..D15.

La dirección base del 8254 es FF01H. Otras conexiones del 8254 son las líneas del sistema /RD y /WR que permiten leer y escribir el 8254; ocho líneas de datos, empleadas para acceder a bytes de control, bytes de estado y valores de cuenta; y líneas de dirección A1 y A2 para seleccionar el registro de control o uno de los tres contadores del 8254.



| A8-A15 | A5-A7 | A4 | A3 | A2 | A1 | A0 | M/I \bar{O} | Y OUTPUT SELECTED | SYSTEM BASE ADDRESS | DEVICE |
|------------------|-------|----|----|----|----|----|---------------|----------------------|------------------------|----------|
| 1 | 0 | 0 | 0 | X | X | 0 | 0 | 0 | F F 0 0 | 8259A #1 |
| 1 | 0 | 0 | 1 | X | X | 0 | 0 | 1 | F F 0 8 | 8259A #2 |
| 1 | 0 | 1 | 0 | X | X | 0 | 0 | 2 | F F 1 0 | 8254 |
| 1 | 0 | 1 | 1 | X | X | 0 | 0 | 3 | F F 1 8 | |
| 1 | 0 | 0 | 0 | X | X | 1 | 0 | 4 | F F 0 1 | |
| 1 | 0 | 0 | 1 | X | X | 1 | 0 | 5 | F F 0 9 | |
| 1 | 0 | 1 | 0 | X | X | 1 | 0 | 6 | F F 1 1 | |
| 1 | 0 | 1 | 1 | X | X | 1 | 0 | 7 | F F 1 9 | |
| ALL OTHER STATES | | | | | | | | NONE | | |

8.3.2.- Inicialización.

Cuando se conecta cualquier dispositivo programable, éste se encuentra en un estado indefinido. Para inicializarlo correctamente tenemos que tener en cuenta la siguiente *serie de pasos* :

1. Determinar la ***dirección base*** en el sistema para el dispositivo. Hacemos esto a partir de la circuitería de decodificación o mediante una tabla de interfaz HW-SW. En nuestro ejemplo vale FF01H.
2. Usamos la ***hoja de datos del fabricante*** para determinar las direcciones internas para cada uno de los registros de control, puertos, temporizadores, registros de estado, etc., del dispositivo. La tabla muestra las ***direcciones internas*** para los tres contadores y el registro de control para el 8254. A0 en esta tabla representa la entrada A0 del dispositivo, y A1 representa la entrada A1 del dispositivo. No podemos emplear la línea de direcciones del sistema A0 como una de estas porque ya la hemos usado como entrada en el decodificador de direcciones y no interviene en la selección de los registros internos.

| A1 | A0 | SELECTS |
|----|----|-----------------------|
| 0 | 0 | COUNTER 0 |
| 0 | 1 | COUNTER 1 |
| 1 | 0 | COUNTER 2 |
| 1 | 1 | CONTROL WORD REGISTER |

(a)

| SYSTEM ADDRESS | 8254 PART |
|----------------|-------------|
| F F 0 1 | COUNTER 0 |
| F F 0 3 | COUNTER 1 |
| F F 0 5 | COUNTER 2 |
| F F 0 7 | CONTROL REG |

3. Sumamos cada una de las direcciones internas a la dirección base para determinar la ***dirección en el sistema*** de cada registro del dispositivo. Tenemos que hacer esto para conocer la dirección real donde tenemos que escribir los bytes a programar. En nuestro ejemplo todas las direcciones son impares porque el dispositivo está conectado a la parte alta del bus de datos.
4. Mirar en la hoja del fabricante el ***formato de las palabras de control*** que tenemos que escribir en el dispositivo para inicializarlo. Para dispositivos diferentes, la palabra de control tiene formatos diferentes. Para inicializar el 8254, escribimos una palabra de control adecuada en el registro de control para cada contador que queramos usar. La figura siguiente muestra el formato de la palabra de control.
5. Construir la palabra de control necesaria para inicializar el dispositivo para la aplicación. Construimos esta palabra de control bit a bit. Resulta útil escribir las ocho cajas de la figura de la derecha para no dejarnos atrás ninguno. Hay que conocer el significado de cada bit de la palabra de control y escribirlo debajo de cada bit. Esta documentación tendrá un valor incalculable cuando tengamos que modificar un programa o reparar una placa defectuosa.

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|-----|-----|-----|-----|----|----|----|-----|
| SC1 | SC0 | RW1 | RW0 | M2 | M1 | M0 | BCD |

SC – SELECT COUNTER:

| SC1 | SC0 | |
|-----|-----|---|
| 0 | 0 | SELECT COUNTER 0 |
| 0 | 1 | SELECT COUNTER 1 |
| 1 | 0 | SELECT COUNTER 2 |
| 1 | 1 | READ-BACK COMMAND (SEE READ OPERATIONS) |

RW – READ/WRITE:

| RW1 | RW0 | |
|-----|-----|--|
| 0 | 0 | COUNTER LATCH COMMAND (SEE READ OPERATIONS) |
| 0 | 1 | READ/WRITE LEAST SIGNIFICANT BYTE ONLY. |
| 1 | 0 | READ/WRITE MOST SIGNIFICANT BYTE ONLY. |
| 1 | 1 | READ/WRITE LEAST SIGNIFICANT BYTE FIRST, THEN MOST SIGNIFICANT BYTE. |

M – MODE:

| M2 | M1 | M0 | |
|----|----|----|--------------------------------------|
| 0 | 0 | 0 | MODE 0 – INTERRUPT ON TERMINAL COUNT |
| 0 | 0 | 1 | MODE 1 – HARDWARE ONE-SHOT |
| X | 1 | 0 | MODE 2 – PULSE GENERATOR |
| X | 1 | 1 | MODE 3 – SQUARE WAVE GENERATOR |
| 1 | 0 | 0 | MODE 4 – SOFTWARE TRIGGERED STROBE |
| 1 | 0 | 1 | MODE 5 – HARDWARE TRIGGERED STROBE |

BCD:

| 0 | BINARY COUNTER 16-BITS |
|---|--|
| 1 | BINARY CODED DECIMAL (BCD) COUNTER (4 DECADES) |

NOTE: DON'T CARE BITS (X) SHOULD BE 0 TO INSURE COMPATIBILITY WITH FUTURE INTEL PRODUCTS.

6. Finalmente, escribimos las palabras de control que hemos construido a la dirección del registro de control para el dispositivo. En el caso del 8254, también mandaremos la cuenta inicial a cada uno de los registros de cuenta.

Una palabra de control se envía de forma separada a cada contador que queramos usar del dispositivo. Sin embargo, el 8254 solo tiene una dirección para el registro de control. El truco aquí es que las palabras de control para todos los contadores se escriben en la misma dirección del dispositivo. Como indica la figura usamos los dos bits de mayor peso como control de que contador queremos inicializar con la palabra de control. Por ejemplo, si queremos escribir una palabra de control para el contador 0, hacemos el bit SC1 de la palabra de control “0” y el bit SC0 “0”.

Los contadores de 16 bits son contadores hacia abajo. Esto significa que el número en el contador se decrementa en cada pulso de reloj. Podemos programarlo para contar en BCD o en binario. Los detalles del funcionamiento de los diferentes modos de conteo se pueden encontrar en el manual del fabricante del dispositivo.

8.4.- Controlador de interrupciones (PIC) i8259.

Cuando el 8086 responde a una señal de interrupción INTR, su respuesta es algo diferente a la respuesta a otras interrupciones, esta es vectorizada. El tipo de interrupción se envía al 8086 desde un dispositivo hardware externo como el 8259A, controlador de interrupciones con prioridad.

Veamos primero como funcionan las interrupciones con el 8086. Si la máscara de interrupciones del 8086 está puesta y la entrada INTR recibe una señal a nivel alto, el 8086:

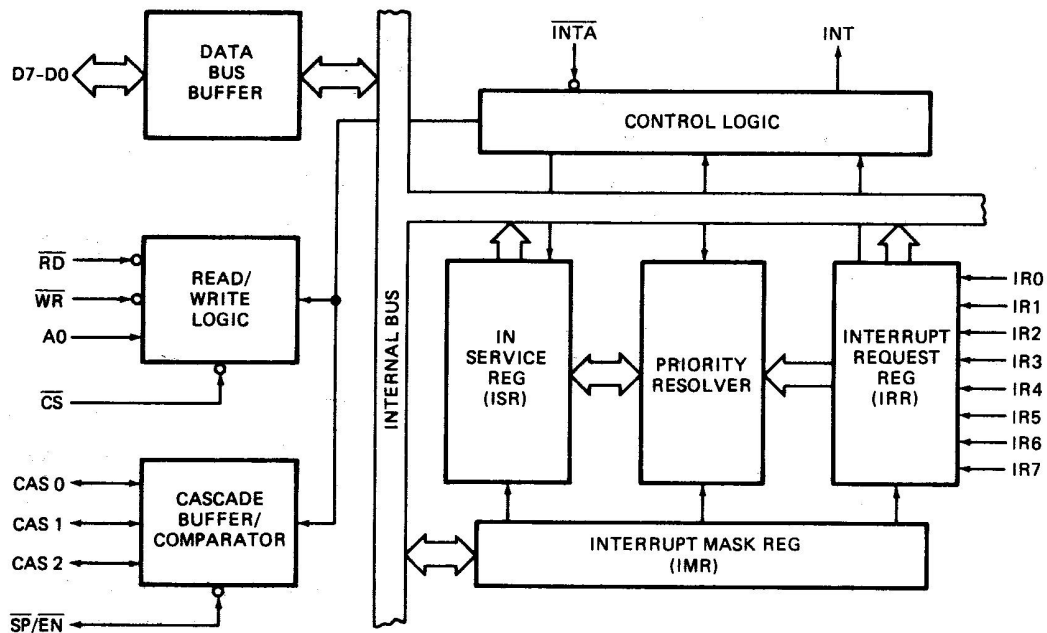
- 1- Manda dos pulsos de reconocimiento de interrupción sobre su patilla /INTA a la patilla /INTA del PIC. El pulso /INTA hace que el 8259A mande el tipo deseado de interrupción al 8086 sobre el bus de datos.
- 2- Multiplica el tipo de interrupción que recibe por 4 para producir una dirección en la tabla de vectores de interrupción (interrupción vectorizada).
- 3- Guarda los flags (registro de indicadores) en la pila.
- 4- Habilita la máscara de interrupción.
- 5- Introduce la dirección de retorno en la pila.
- 6- Obtiene la dirección de la rutina de tratamiento de interrupción de la tabla de vectores de interrupción.
- 7- Ejecuta la rutina de servicio de interrupción.

Observemos el **diagrama de bloques del 8259**. El bus de datos permite al 8086 mandar palabras de control al PIC y leer los códigos de estado. Las entradas /RD y /WR controlan estas transferencias cuando el dispositivo se ha seleccionado al colocar la entrada de selección de chip /CS a nivel bajo.

Las **ocho entradas de interrupción**, etiquetadas IR0 a IR7 permiten que una señal aplicada a cualquiera de estas entradas provoque que el 8259 active su patilla de salida INTR. Si ésta patilla se conecta a la entrada de interrupción del 8086 y se encuentra su máscara desinhibida, se produce una respuesta a interrupción.

La entrada /INTA del 8259 se conecta a la correspondiente salida del 8086. El 8259 usa el primer pulso del ciclo INTA para realizar algunas actividades. Cuando recibe el segundo pulso, coloca el **tipo de interrupción** sobre el bus de datos. El tipo de interrupción que manda al procesador depende de la entrada IR que se recibió y del tipo de interrupción que programamos al PIC en el momento de su inicialización. El 8259 une todas las señales de interrupción en una sola y manda al procesador el tipo especificado para cada interrupción.

Si dos interrupciones suceden simultáneamente, se producirá la **interrupción más prioritaria**, que será la que esté conectada a una entrada con menor peso, o sea IR0 tendrá más prioridad que todas las demás.



El **registro de máscara de interrupción (IMR)** se emplea para inhibir (enmascarar) o habilitar (desenmascarar) cada entrada de interrupción por separado. Cada bit en este registro corresponde a la entrada de interrupción con el mismo número. Activamos una interrupción escribiendo un byte con un 0 en la posición del bit correspondiente.

El **registro de petición de interrupción (IRR)** mantiene la pista de que peticiones de interrupción se han producido. Si una entrada de interrupción tiene una señal de interrupción, el bit correspondiente en el registro estará a "1".

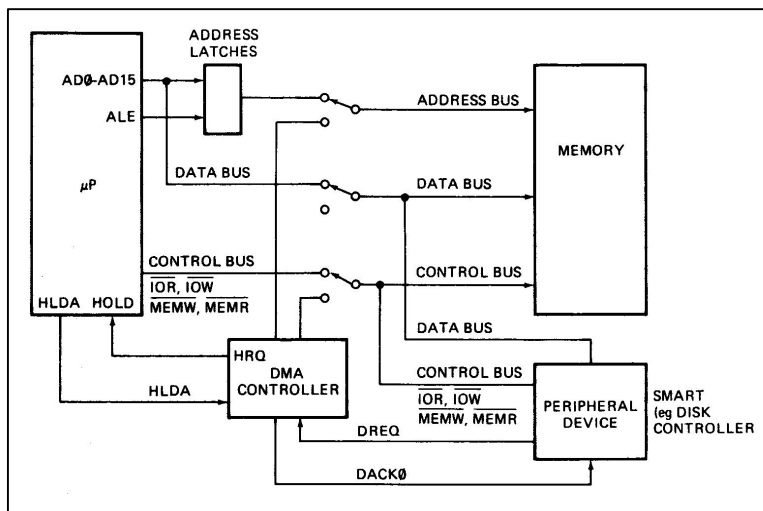
El **registro de peticiones en servicio (ISR)**, mantiene las entradas de interrupción que se están sirviendo. Para cada entrada que se esté sirviendo, el bit correspondiente estará a "1". Esto se hace para **no admitir interrupciones de menor prioridad** que la que se está sirviendo y permitir también que la interrupción que se está realizando pueda ser interrumpida por otra de mayor prioridad. Por tanto, la rutina de servicio de interrupción debe poner este bit a 0 cuando termine de realizar su tarea y **antes de retornar** al programa principal mediante una operación de escritura en el puerto correspondiente.

8.5.- Controlador de Acceso directo a memoria (ADM) i8237.

En algunas aplicaciones, como la transferencia de datos a memoria desde un *disco óptico o magnético*, los datos llegan del disco tan rápido que no se podrían leer mediante instrucciones del programa. En estos casos empleamos un dispositivo hardware denominado controlador de acceso directo a memoria o controlador DMA para realizar la transferencia de datos. El controlador DMA toma prestado temporalmente el bus de direcciones, el bus de datos y el bus de control del microprocesador y transfiere directamente los bytes de datos del controlador de disco a una serie de direcciones de memoria. Como la transferencia de datos se controla totalmente por hardware, es mucho más rápida que si la hiciéramos mediante instrucciones de programa. Un **controlador DMA** también puede transferir datos desde memoria a un puerto. Algunos dispositivos DMA pueden incluso realizar **transferencias memoria a memoria** para implementar transferencias de bloques rápidas.

8.5.1.- Conexiones y operación.

Antes de ver el conexionado real del controlador DMA con el 8086, veamos un diagrama de bloques. El concepto principal a entender es que el microprocesador y el controlador DMA comparten el uso de los buses de direcciones, datos y control. Los tres conmutadores del diagrama intentan mostrar como se transfiere el control de los buses.



Cuando el sistema se enciende, los conmutadores están en la posición de arriba, de forma que **los buses están conectados del microprocesador a la memoria principal y periféricos**. Inicializamos todos los dispositivos programables del sistema y vamos a ejecutar nuestro programa hasta que necesitemos, por ejemplo, leer de un fichero del disco. Para leer un fichero del disco mandamos una serie de comandos al controlador de disco, pidiéndole que busque y

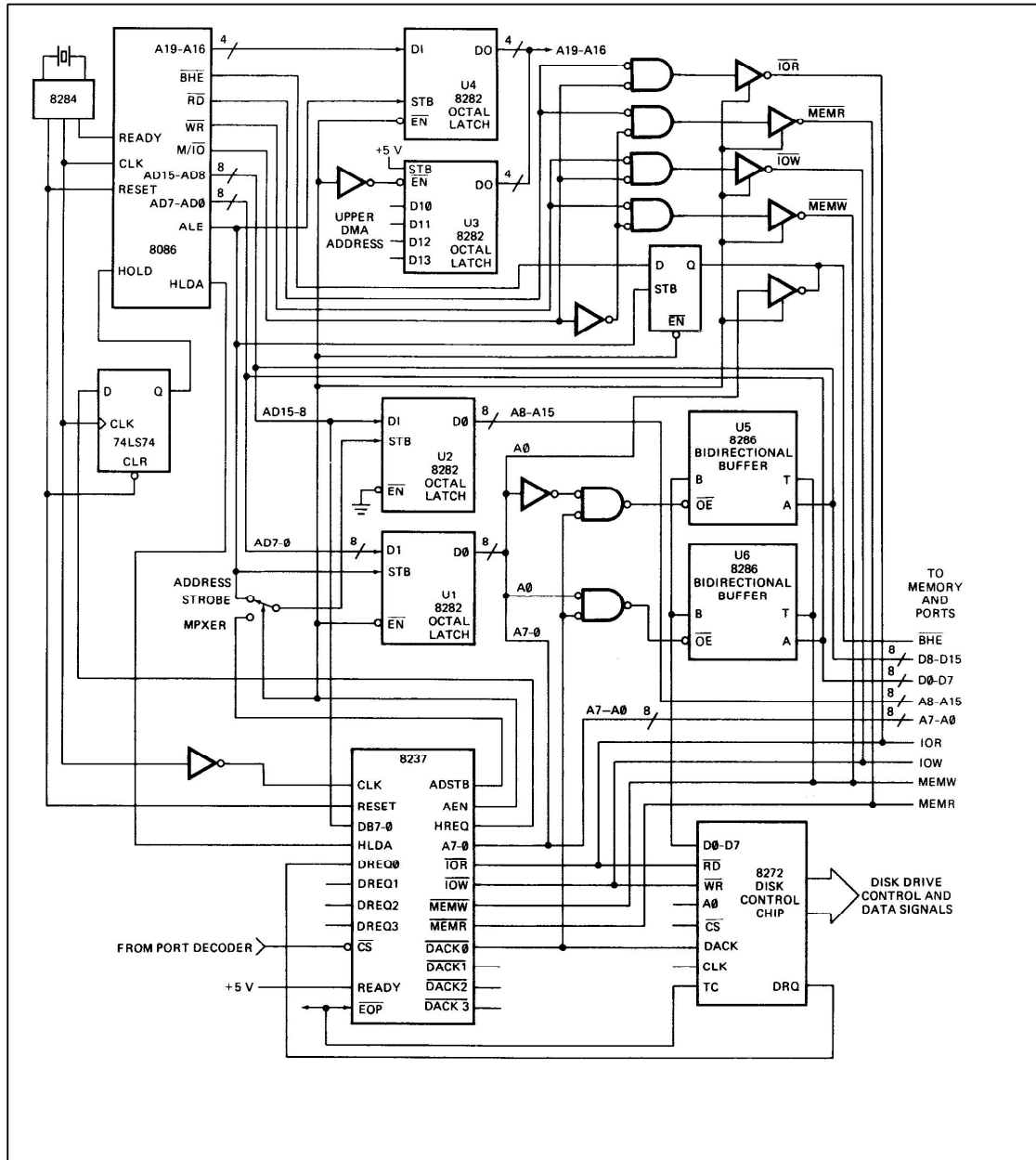
lea el bloque de datos deseado del disco. Cuando el controlador tiene el primer byte del bloque listo, manda una señal de petición de DMA, DREQ, al controlador DMA.

Si esta entrada (canal) del controlador no está enmascarada, el controlador enviará una petición de HOLD, HRQ a la entrada de HOLD del microprocesador. El microprocesador responderá a esta entrada colocando sus buses en estado de alta impedancia y mandando una señal de reconocimiento de HOLD, HLDA, al controlador DMA.

Cuando el controlador DMA recibe la señal HLDA, **mandará una señal de control** que coloca los tres conmutadores abajo en su posición DMA. Esto desconecta el procesador de los buses y conecta el controlador DMA a los buses.

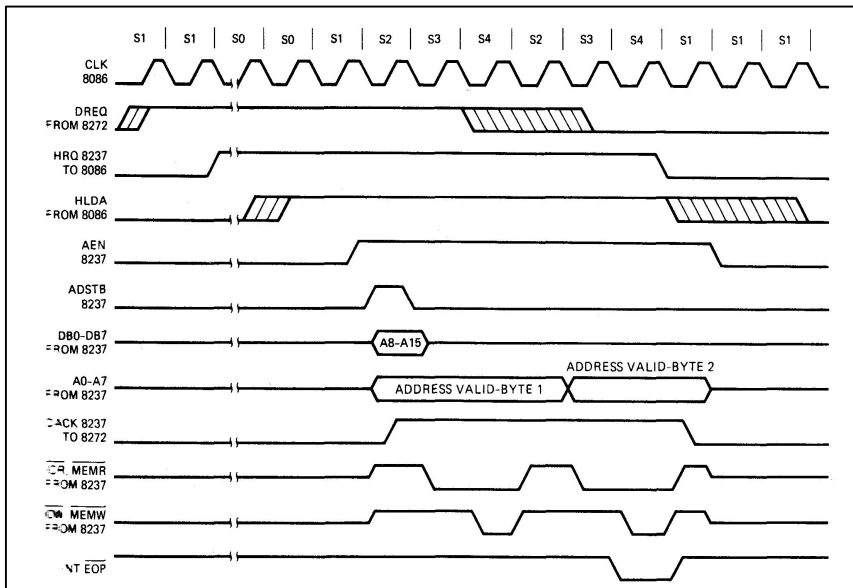
Cuando el controlador DMA **toma el control de los buses**, manda las direcciones de memoria donde se escribirá el primer byte de datos proveniente del controlador de disco. Luego el controlador DMA envía una señal DACK al dispositivo controlador de disco, DACK0, para

indicarle que tenga preparado otro byte de datos. Finalmente, el controlador DMA activa las líneas /MEMW y /IOR del bus de control. La activación de la señal /MEMW permite a la memoria direccionada aceptar el dato escrito a ella. La activación de la señal /IOR permite al controlador de disco sacar el byte de datos desde el disco al bus de datos. El byte de dato se transfiere a continuación directamente del controlador de disco a la dirección de memoria si pasar a través de la UCP ni del controlador de ADM.



8.5.2.- Diagrama de tiempos de la transferencia.

La figura siguiente muestra la secuencia de señales que tiene lugar en una transferencia DMA en un sistema como el de la figura anterior.



Comenzamos cuando el 8237 activa AEN y toma el control de los buses. Entonces, manda los 8 bits bajos de la dirección de memoria en sus patillas A7..A0 y los 8 bits altos de la dirección de memoria en las patillas DB7..DB0. El 8237 introduce un pulso en ADSTB para latched estos bits de direcciones en el 8282 y luego quita estos bits del bus de datos. Al mismo tiempo manda la señal DACK al controlador de disco para informar que se va a realizar una transferencia.

Ahora que todo está listo, el 8237 activa dos señales del bus de control para habilitar realmente la transferencia. Para una transferencia de memoria al controlador de disco, activará /MEMR e /IOW. Para una transferencia del controlador de disco a memoria, activará /MEMW y /IOR. Hay que tener en cuenta que el 8237 **no tiene que poner una dirección de E/S** para habilitar el controlador de disco en la transferencia. Cuando está programado en modo DMA, el controlador de disco solo necesita /IOR o /IOW para habilitar la transferencia. Notar también el 8237 no pondrá una nueva dirección desde A8 a A15 cuando se haga una nueva transferencia, a no ser que estos bits deban ser cambiados. Esto ahorra tiempo durante transferencias de múltiples bytes.

Cuando el número programado de bytes se han transferido, el controlador DMA manda un pulso a su patilla fin de proceso, /EOP, desactivando su petición HOLD al 8086 y pone su señal AEN a nivel bajo. Esto devuelve los buses al 8086.

8.5.3.- Inicialización.

El 8237 está conectado en el sistema como un puerto más, por lo que tenemos que escribir palabras de inicialización como si estuviésemos tratando con cualquier otro puerto de cualquier periférico. Además, varios 8237 se puede conectar en cascada en una configuración maestro-esclavo para ofrecer más canales de entrada y cada dispositivo debe inicializarse.

Como se muestra en las etiquetas de las patillas del 8237, el dispositivo cuenta con cuatro entradas de petición DMA o canales. Para cada canal necesitamos escribir una palabra de comando que especifique la **operación**, la dirección de **memoria de comienzo** y el **número de bytes** a transferir. Cada canal del 8237 se puede programar para transferir un byte en cada petición, un bloque de bytes en cada petición o para transferir bytes hasta que reciba una señal de espera en la entrada/salida /EOP.

8.5.4.- DMA en el IBM PC.

Veamos la sección DMA en el diagrama de bloques de la placa madre del PC. El 8237^a-5 es el controlador de DMA. El 74LS373 se emplea para guardar los 8 bits superiores de la

dirección DMA. El 74LS670 se usa para sacar los bits A16..A19 de la dirección de transferencia DMA.

Para que las placas de periféricos se puedan comunicar con la circuitería de la placa madre mediante DMA, las líneas de señal DMA se conectan también a los conectores de expansión ISA. Las **patillas de petición DRQ1..DRQ3** permiten a las placas periféricas realizar peticiones del bus. Un controlador de disco, por ejemplo, podría pedir la transferencia DMA de un bloque de datos desde la memoria del sistema. Cuando el controlador DMA toma control del bus **activa las señales DACK1..DACK3**. La **señal AEN** se usa para tomar la dirección DMA del bus. Cuando el número programado de bytes se han transferido, la **patilla TC** pasa a nivel alto, señalando el final de la transferencia.