LA UNIDAD CENTRAL DE PROCESO

(UCP)

Arquitectura de Equipos y Sistemas Electrónicos

Por: Pedro y Jose Manuel

Sevilla 12 de Febrero de 2000

Unidad Central de Proceso.

LA UNIDAD CENTRAL DE PROCESO.

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

El Microprocesador (uP) es un dispositivo electrónico que reúne en un único elemento las partes fundamentales de una UCP. Por eso, hablar de UCP o de uP resulta totalmente similar en la mayoría de los casos.

Dependiendo del sistema, la UCP puede realizar más o menos funciones que las incluidas en un uP, por eso utilizamos el término UCP para denominar la parte central de los Sistemas Basados en Microprocesadores, y el uP en sí mismo se considera dentro de la UCP como un componente más.

1.- Entorno de sistema.

El estudio de la UCP que desarrollamos en lo que sigue está enfocado a una estructura de sistema mínima constituida por los 4 elementos básicos: **BUS DEL** SISTEMA, **UCP, UCM** y UES. La figura 1 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.

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 2 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 uP. La funcionalidad del uP es lo más universal posible, cubriendo de esta manera la mayoría de las funciones de una UCP. Las funciones particulares o especiales que deseemos en una UCP la hemos de implementar por medio de una circuitería complementaria a la del uP.

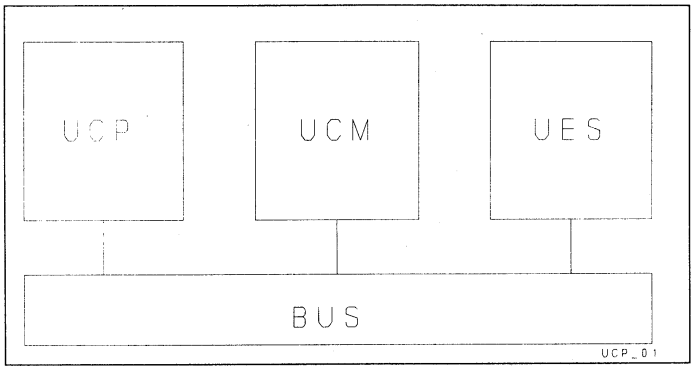


Figura 1. Estructura del sistema.

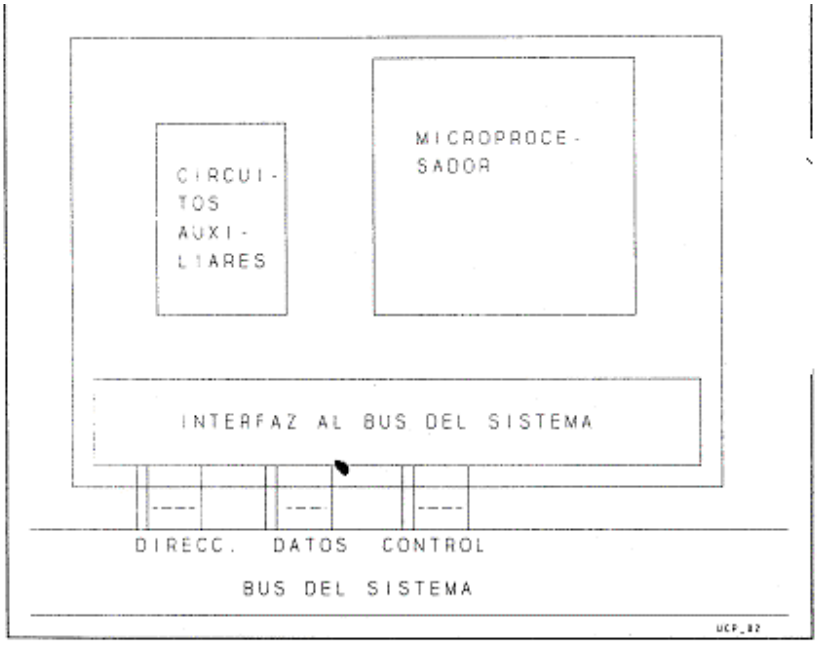


Figura 2. Partes de la UCP.

En sistemas pequeños y medianos, el uP 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 vista a poder acoplar la UCP al bus del sistema, la conectividad de esta unidad ha de estar de acuerdo con las especificaciones del bus. La parte de Interfaz al bus de la UCP tiene esta misión. El interfaz al bus se ocupa de hacer posible la comunicación entre el uP Y El bus del sistema.

2.- Perfil del microprocesador.

En este apartado presentamos la 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 3 se muestra la representación lógica de un uP, sobre la que nos basaremos para el desarrollo siguiente.

2.1.- Señales externas.

Un uP dispone un conjunto de señales para su manejo desde el exterior (ver figura 3). Estas señales se agrupan en tres conjuntos, o buses, cuya agrupación forma el bus local del uP; el bus de datos, el bus de direcciones y el bus de control. En lo que sigue discutimos cada uno de ellos.

Bus de datos

El uP dispone de un conjunto de señales por la cual circula la información desde y hacia el u*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).

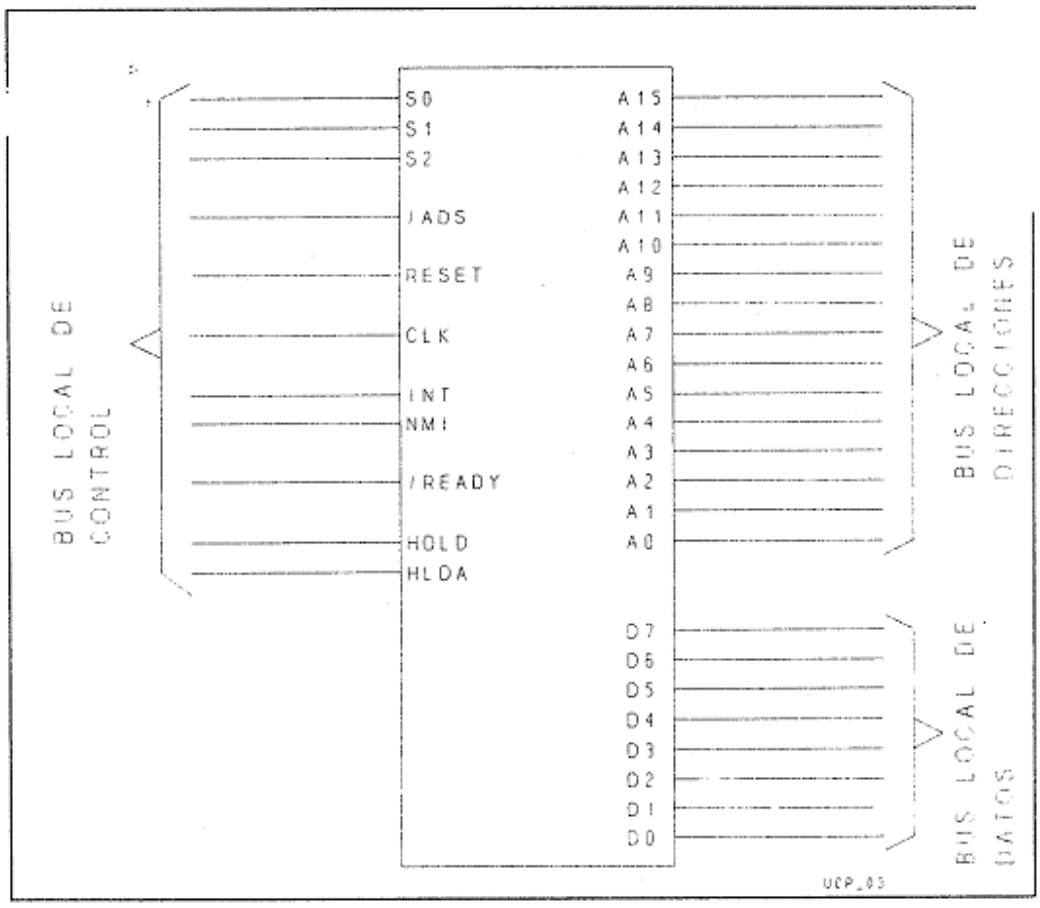


Figura 3. Representación lógica del uP.

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 que se compone del, nombre genérico del bus "D" más un número que identifica su peso binario en la información. Así pues, el bus de datos de nuestro uP, que es de 8 bits, se compone de 8 líneas de señal denominadas D7 a DO.

Bus de direcciones

Para poder localizar la información en la memoria, el uP ha de indicar en qué posición se encuentra por medio de la dirección que corresponda. Para ello dispone de un conjunto de líneas de señal que conforman la dirección. Este conjunto se denomina bus de direcciones (A) y la señales que lo componen toman en nombre genérico del bus (A) más el número que identifica su peso binario en la información de dirección. Asi pues, para nuestro modelo de uP, El bus de direcciones que es de 16 bits, está compuesto por las señales A15 a AO.

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 u*P*. Al contrario de los buses de direcciones y datos, la lineas 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 uP y las de control del bus.

Señales de control del sistema

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

Líneas de estado (S2, SI, SO)

Las combinaciones de estas señales de salida del juP indican la situación (ciclo) en que se encuentra. Con estas tres lineas de estado. El uP identifica los 7 estados diferentes en que se puede encontrar. La codificación puede ser diferente para cada tipo de uP pero en general los ciclos son comunes. La codificación para nuestro uP es la indicada en la tabla 1.

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	Hait
1	1	1	No asignado

Tabla 1. Identificación de los ciclos del uP por las señales de estado. Por medio de las señales de estado podemos identificar la actividad del uP en cada instante. Estas señales son propias del u*P* y no son triestado.

Señales de control del uP

Por medio de estas señales podemos controlar el funcionamiento del uP desde el exterior de éste.

Línea de inicialización (RESET)

La señal RESET inicializa al uP a un estado conocido. Cuando RESET se activa el contador de programa se carga con un valor conocido (O en nuestro uP) relizando el primer fetch 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 uP 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 uP es la entrada de la interrupción enmascarable. Esta señal podrá ser atendida o no por el uP dependiendo de una información interna a éste (máscara de interrupción) que puede 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 uP abandona la secuencia del programa en la que se encuentra y pasa a realizar otro programa denominado de atención a la interrupción.

Interrupción no enmascarable (NMI)

Para esta señal de entrada al uP no existe macara alguna, de tal forma que cuando NMI se activa, el u*P* abandona la secuencia que está realizando para atender a la NMI (rutina de NMI).

Final de ciclo (/READY)

Esta señal de entrada al uP indica el final del ciclo en curso. Cuando el uP inicia un ciclo sobre el bus local, pone en él toda la información necesaria para realizar el ciclo. Una vez hecho ésto, el uP espera que la unidad destino de la transferencia active la señal /READY para indicar que se ha realizado correctamente ésta. El uP iniciará la ejecución del siguiente ciclo máquina en cuanto muestree la señal /READY activada. /READY es activa a nivel bajo.

Señales de control del bus

Por medio de estas señales, los otros elementos del sistema pueden solicitar al uP el control del bus. En otras palabras, se solicita al uP 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 uP y activa a nivel H, y otra de respuesta del uP que es salida y se llama HLDA (activa a nivel H). Cuando HLDA se activa, las lineas de los buses de direcciones, datos y control se encuentran en triestado por parte del uP. Cuando la señal HOLD se desactive, el uP tomara de nuevo el control del bus y desactivará la señal HLDA.

2.2.- Funcionamiento.

El uP 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) b) la memoria de datos

La memoria de programa contiene, convenientemente codificadas en binario, todas las órdenes (instrucciones) que ha de ejecutar el uP. Todas estas instrucciones se colocan ordenada y consecutivas en la memoria de tal forma gue si un programa comienza en la posición N de la memoria es porgue 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 uP toma una a una cada instrucción y la ejecuta. El tiempo gue tarda el uP en ejecutar una instrucción se llama **CICLO DE INSTRUCCION.** Este ciclo de instrucción se compone de un número entero de ciclos de reloj.

La ejecución de cualquier instrucción se compone de tres fases:

a) búsgueda del código de operación

b) búsgueda del operando.

c) ejecución.

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

Durante la fase a) el u*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 gue no exista (cuando el direccionamiento es del tipo implícito) puede que en esta fase se obtenga el dato en sí mismo (direccionamiento inmediato) o 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) pueden existir o no diferenciada de la a) y la b) pues depende de cuanto tiempo necesite el uP 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 o 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 del código.

La figura 4 muestra el cronograma que desarrolla este ciclo. En ella podemos ver que las señales de estado del bus de control (S2, SI, SO) toman el valor correspondiente a la codificación del ciclo de fetch (ver tabla 1). Para acceder al código de operación, el uP 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 linea /READY para indicar al uP que la información existente en el bus de datos es válida. El uP recoge la información del bus de datos y finaliza de este modo el ciclo.

Ciclo de lectura en memoria.

La figura 5 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.

La figura 6 muestra el cronograma desarrollado por el ^P para un ciclo de escritura en memoria. Las señales de identificación de ciclo (S2, SI, SO) toman el estado correspondiente al ciclo en curso. El uP envia por el bus de direcciones la dirección sobre la que se va a escribir . Cuando el uP recibe la señal /READY activada procedente de la memoria, se acaba el ciclo.

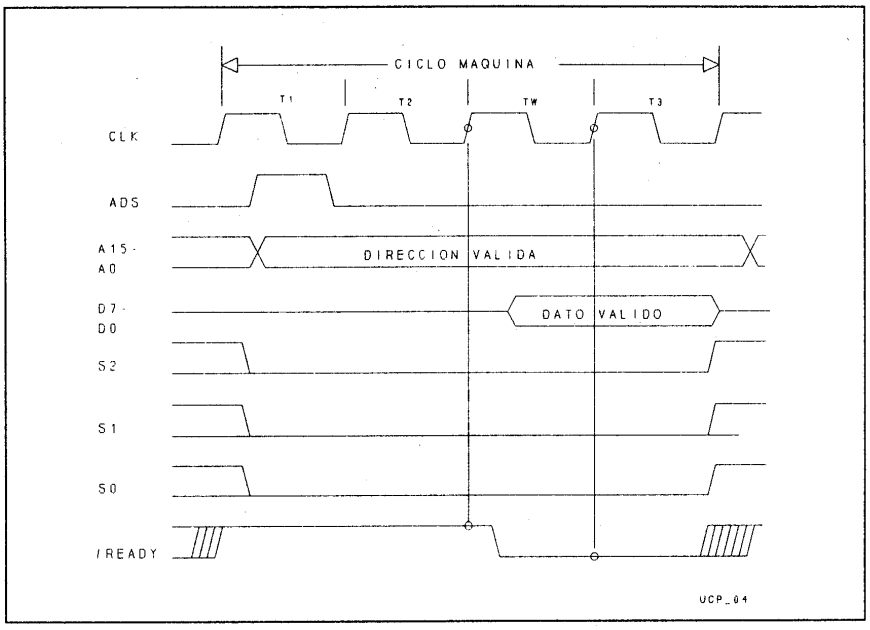


Figura 4. Cronograma del ciclo de búsqueda de código.

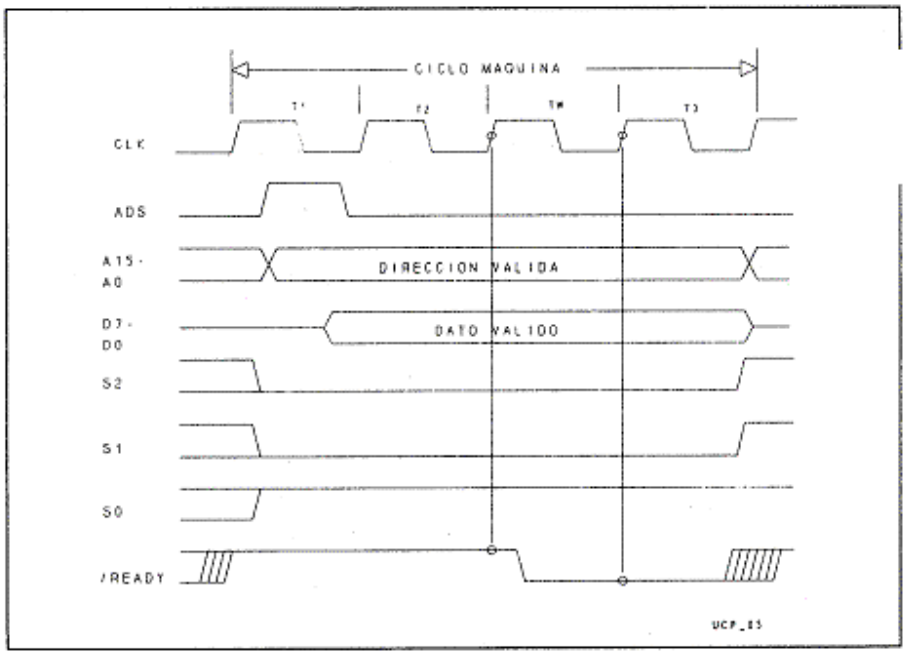


Figura 5. Cronograma del ciclo de lectura en memoria.

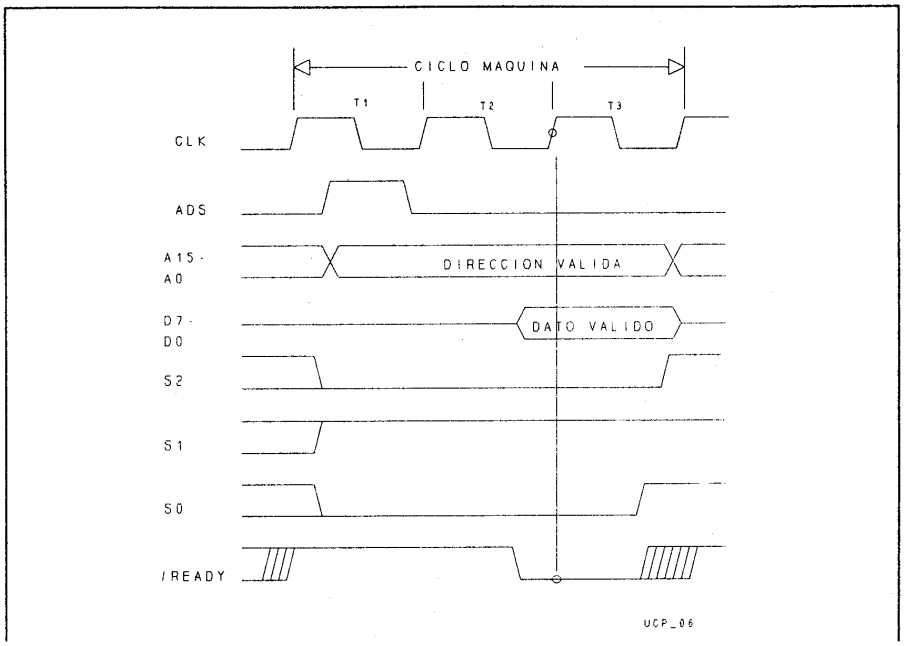


Figura 6. Cronograma del ciclo de escritura en memoria.

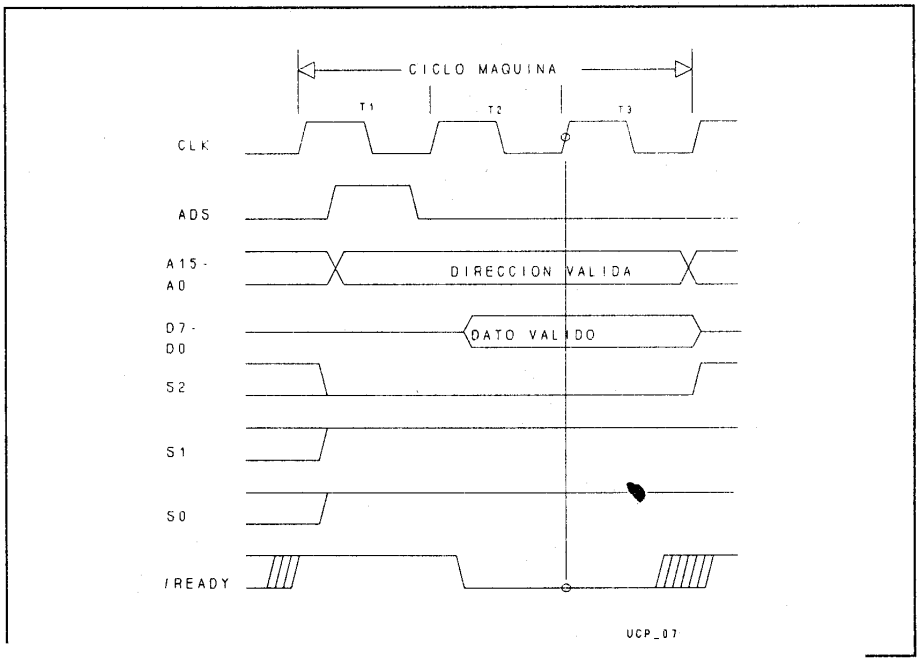


Figura 7. Cronograma del ciclo de lectura en E/S.

Ciclo de lectura en E/S.

La figura 7 muestra el ciclo de lectura en el espacio de 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 8 muestra el desarrollo del ciclo de escritura en E/S.

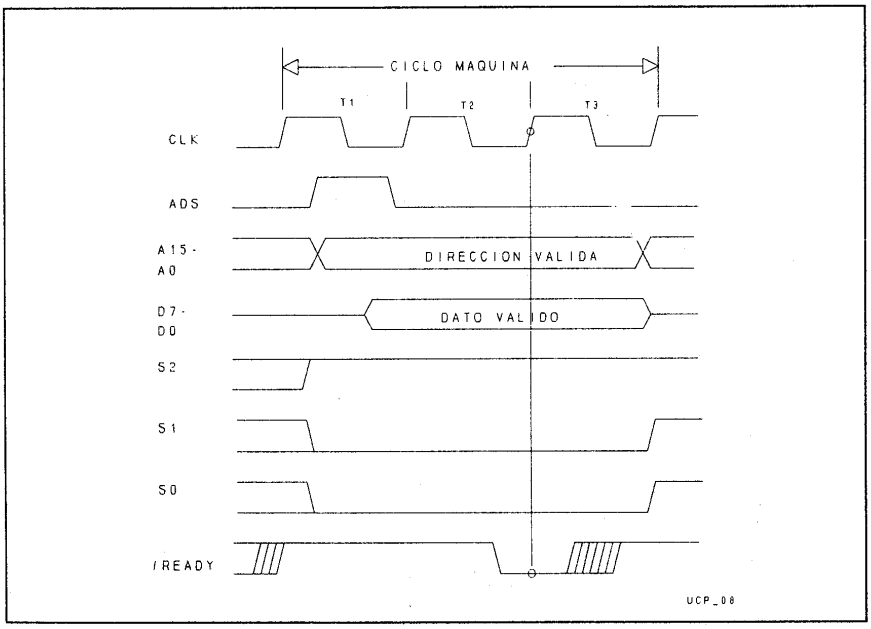


Figura 8. Cronograma del ciclo de escritura en E/S.

3.- Arquitectura del microprocesador.

Un uP típica contiene (ver figura 9) tres unidades básicas:

Registros internos. Unidad Lógica y Aritmética. Unidad de Control.

Cada una de ellas se discuten a continuación.

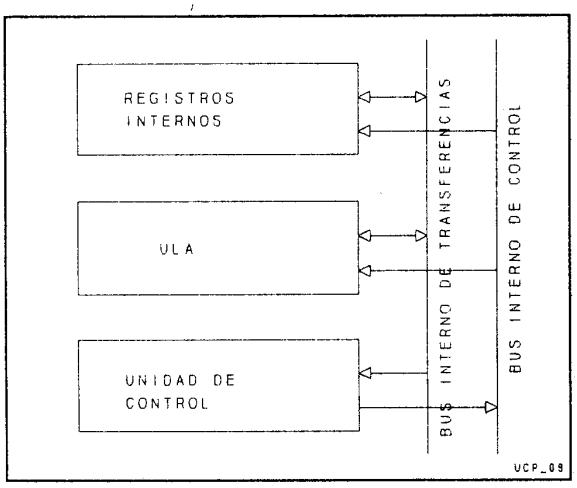


Figura 9. Unidades básicas del uP.

3.1.- Registros internos.

Se denominan registros internos al conjunto de registros que dispone el uP para su funcionamiento.

Los registros internos son unidades de almacenamiento temporal dentro del uP . El tamaño de los registros depende de la función que ha de realizar y del tipo de uP. 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 o de usuario.

Los registros no accesibles (figura 10) son aquellos registros internos al uP, que el usuario no puede controlar directamente su funcionamiento ni su contenido. Son de uso propio y restringido del uP . Desde el punto de vista del usuario, estos registros son totalmente transparentes ya que no afectan, aparentemente, al funcionamiento del uP observado desde el exterior.

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

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 uP (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.

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 algunos de ellos ya que su presencia nos facilitará la comprensión del funcionamiento del conjunto.

Registro de instrucción.

Las instrucciones que ha de ejecutar el uP 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 uP obtiene de la memoria el código de operación de una instrucción lo guarda en un registro denominado registro de instrucción y se mantiente ahí hasta finalizar la ejecución de esa instrucción.

El registro de instrucción es del tamaño adecuado a los códigos utilizados en el uP. 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 uP está formado por las salidas de un registro interno al uP que se denomina registro del bus de direcciones (MAR).

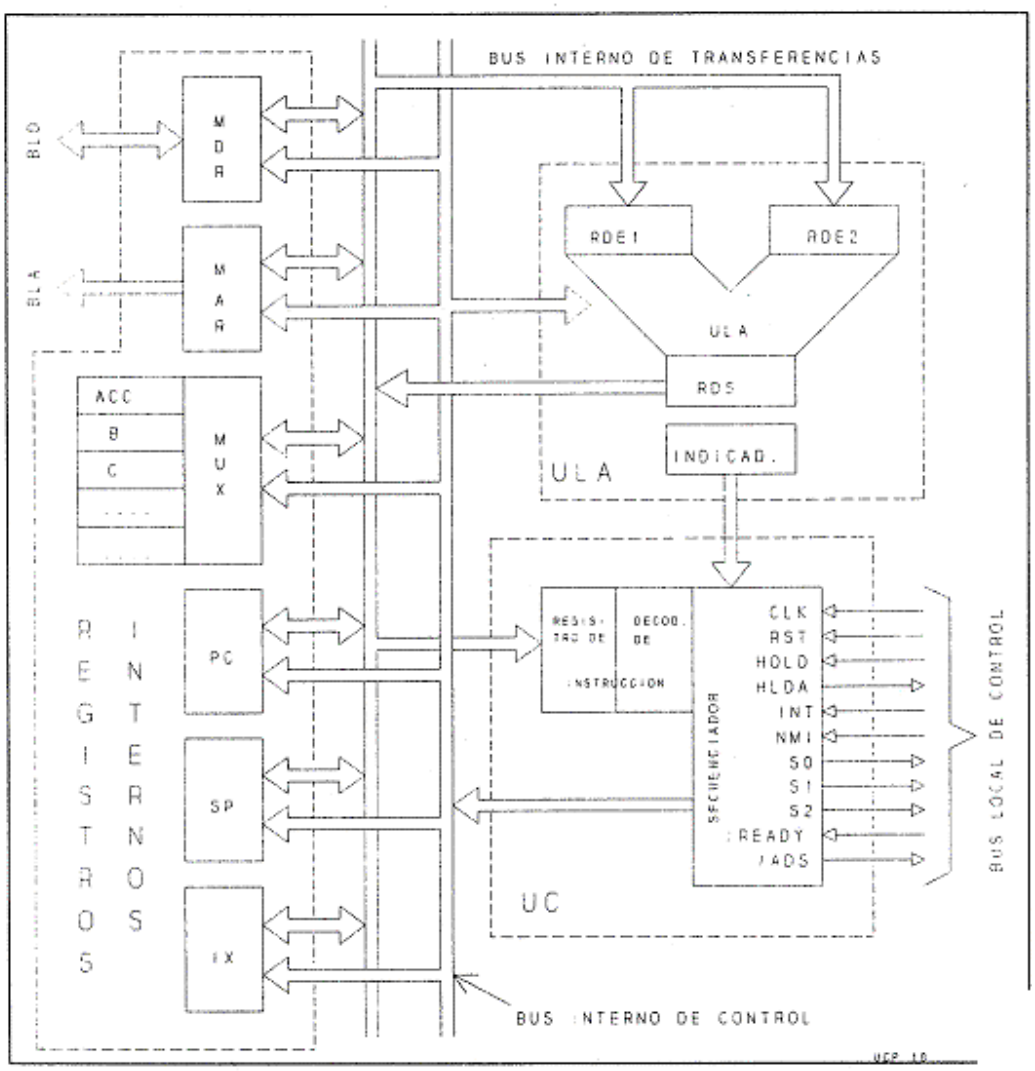


Figura 10. Estructura interna del uP.

Este registro tiene como misión retener todo el tiempo que sea necesario una determinada información de dirección sobre el bus de direcciones, independizándolo de la evolución de los restantes registros internos del uP.

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 uP 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 uP .

Este registro se llama MDR y es de entrada y salida paralelo, bidireccional y si el uP permite el funcionamiento en ADM, la salida hacia bus de datos externo ha de ser triestado.

Registros accesibles.

Como hemos visto antes, el uP dispone de un conjunto de registros 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 uP, aunque en algún caso pueda realizar otras funciones.

Registro contador de programa.

Las instrucciones que conforman un programa se almacenan en la memoria en direcciones consecutivas. El uP 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 uP dispone de un registro destinado a contener la dirección de la instrucción a ser ejecutada. Este registro es el contador de programa (PC). El uP 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 posiciones 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 especifica pero cuyo contenido lo podemos controlar a través de las instrucciones correspondientes. .

Registro puntero de la pila.

El Stack o pila es una zona de memoria de datos reservada al uP y que ésta destinada a almacenar determinada información de importancia. Más adelante al describir el funcionamiento del pila veremos en detalle la utilización de este registro.

Al ser el pila una zona de la memoria, el uP ha de direccionarla para poder leer y escribir sobre ella. Esto lo hace a través de un registro especial denominado puntero del pila (stack pointer, SP). El SP es capaz de volcar su contenido directamente sobre el registro MAR.

El registro puntero del pila (SP) tiene una dimensión determinada que hace fijar la posición y el tamaño del pila en la memoria.

Cada vez que el uP necesita escribir o leer un dato de la pila, vuelca al bus de direcciones (a través del MAR) el contenido del puntero del pila.

El uP maneja, de forma automática, el puntero del 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 uP 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 uP, 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 uP.

Registro de indicadores.

Los indicadores son biestables que utiliza el uP para memorizar determinadas circunstancias ocurridas durante un proceso. Estos indicadores se agrupan en un registro denominado de indicadores (o registro de estado de la máquina, PSW).

Este registro de estado tiene un tamaño variable de un *uP a* otro, dependiendo de las necesidades prevista por el diseñador de la misma.

Es un registro algo particular en cuanto que cada uno de los bits es independiente de los demás.

El uP actualiza el estado de estos indicadores automáticamente como consecuencia del resultado de la instrucción ejecutada, representando sobre ellos el estado en que ha quedado la máquina.

La figura 11 muestra el contenido del registro de indicadores del uP. 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:

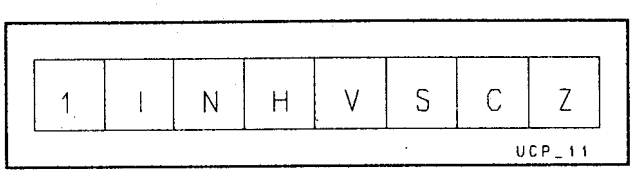


Figura 11. Registro de indicadores del uP.

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 SUBSTRACCION (N).

Ya que el algoritmo para corregir operaciones en BCD es diferente para las operaciones de suma y de resta, este indicador especifica cual fue la última operación ejecutada.

MASCARA DE INTERRUPCION (I).

Este bit permite la entrada o no de una interrupción enmascarable al u*P*. Si la máscara está activada, I="1", el uP no hace caso de la señal de entrada de interrupción enmascarable INT. La interrupción está enmascarada. Si I="0", el uP aceptará la entrada de interrupción por la señal INT.

Hay que anotar 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) (es modificable directamente por el usuario).

Registros de uso general.

Estos son registros internos de el uP que no tienen una función predefinida. Se pueden utilizar como origen o destino de los datos en el u*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 de el uP con un tiempo de acceso muy corto. Esto hace que los fabricantes introduzcan en máximo número posible de estos registros ya que así se aumenta enormemente la velocidad de proceso.

El tamaño de los registros internos suele ser el mismo que el del dato de el uP O un múltiplo o submúltiplo entero.

En muchas ocasiones los registros de uso general tienen funciones de puntero de direcciones para determinadas operaciones sobre la memoria.

3.2.- Unidad Lógica y Aritmética.

La ULA es la parte del u*P* que se encarga de realizar las operaciones lógicas y aritméticas del uP. 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. Los uPs más avanzados incorporan funciones más complejas, como multiplicación y división, que simpifican el algoritmo. Además, existe la posibilidad (en algunos uPs) 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 uP 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 entra en el decodificador de instrucción que lo identifica y activa las señales internas y externas correspondientes del uP.

La UC es un sistema secuencial controlado por el reloj del uP que arranca su funcionamiento a partir de la decodificacion 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 uP. Algunos uPs permiten que sea el propio usuario quien escriba esta memoria del microprograma. Esta posibilidad permite hacer, un uP personalizada. ^\

4.- Funcionamiento del microprocesador.

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

Funcionamiento interno.[1]

Nos basaremos en la arquitectura interna del uP representada en la figura 10 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 uP 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 limita el funcionamiento con procesos totalmente asincronos respecto al uP, 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 uP se pueda compatibilizar con cualquier otro elemento (memoria, dispositivo E/S, etc.) aunque éstos no sean síncronos con el reloj de aquella.

4.1.- La pila.

La pila del uP es una zona de la memoria externa a ella que se utiliza para almacenear 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 vacia 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 vacia.

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.

4.2.- Ciclos funcionales.

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

4.2.1.- Funcionamiento interno.

En la figura 10 podemos ver la existencia de dos buses internos (BIT y BIC) que se utilizan para interconexionar las distintas unidades. Por 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 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 sólo utilizan los 8 bits más bajos (7 a 0) 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 u*P*. Este bus se compone de todas las señales de control que hemos descrito antes.

El interfaz con el exterior del uP 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 uP 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.

Toda instrucción comienza con la búsqueda del código de operación en la memoria, para lo cual la UC hace que el contenido del PC salga al bus de direcciones externo a través del MAR y habilita la entrada de datos desde el exterior (a través del MDR) hacia el registro de instrucción. Una vez decodificada la instrucción se generan (por parte de la UC) los controles necesarios para su ejecución.

4.2.2.- Funcionamiento externo.

Como hemos dicho antes, el funcionamiento del uP es cíclico. El reloj del uP es el que marca cada uno de los pasos del funcionamiento.

La fase de búsqueda del código de operación es un ciclo de máquina particular que recibe el nombre de CICLO **de** BUSQUEDA o FETCH. El comienzo de cada uno de los ciclos de instrucción que realiza el u*P* para ejecutar un programa es un ciclo de búsqueda de código de operación.

Durante el ciclo de búsqueda, el uP desarrolla un acceso de lectura a la memoria de programa para obtener de ella el código de operación de la instrucción que se trate. Este código que se obtiene de la memoria es depositado, durante este ciclo, en el registro de instrucción del uP y es identificado por el decodificador de instrucción. De tal forma que al finalizar el ciclo, el uP ya tiene información de cual es la función a realizar.

Una vez acabada la fase de búsqueda del código de operación, se pasa a la fase de búsqueda del operando. Esta operación ocupará o no otro ciclo máquina dependiendo de donde se encuentre el operando. Más adelante, cuando discutamos los modos de direccionamiento del operando, veremos que hay veces en que el operando se incluye en el mismo código de operación de la instrucción.

Si el operando hay que buscarlo fuera del uP o el resultado hay que depositarlo en la memoria, la instrucción se ve acompañada de algún ciclo máquina de lectura o escritura en memoria o E/S.

Sea, por ejemplo 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 transferencia entre registros) no acceden al exterior del uP.

La secuencia desarrollada hacia el exterior del uP 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 escribir el contenido de un registro interno en una determinada posición de la memoria, la secuencia evolucionan así:

- Leer el código de operación de la instrucción. Ciclo de fetch con un acceso de lectura sobre la memoria de programa.
- Leer 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 el la dirección se compone de dos bytes).
- La dirección de destino leida en los dos ciclos anteriores hay que componerla en el orden correcto y enviarla al registro MAR. Se trata de una operación interna al u*P* que no desarrolla

ciclo externo pero que consume ciclos de reloj en su ejecución.

- Escribir el dato en la dirección indicada por la instrucción. Implica un ciclo de acceso de escritura sobre la memoria de datos.

Como podemos ver, desde el exterior del uP sólo podremos apreciar los ciclos que el uP requiere accesos a su exterior. Internamente ella ejecutará funciones que no tienen representación externa y que sólo podemos observar como ciclos consumidos si actividad aparente.

Ciclos funcionales básicos.

Los ciclos funcionales básicos son todos los ciclos externos diferentes que es capaz de desarrollar el uP durante su funcionamiento. Para cada modelo de uP existe una colección de ciclos básicos que en general, son diferentes de un modelo a otro. Sin embargo todas los uPs realizan una colección común de ciclos que son los que estudiaremos a continuación. Por supuesto que los ciclos aquí discutidos son desarrollados por cada uP diferente de forma diferente.

Ciclo de búsqueda del código de instrucción (fetch).

Como se ha dicho en repetidas ocasiones, toda instrucción comienza con la búsqueda del código de la instrucción. Este ciclo se denomina de fetch y consiste en un acceso de lectura sobre la memoria de programa del sistema. La figura 14 representa el cronograma de este ciclo en el que podemos ver que se desarrolla en tres periodos de reloj (T1 a T3).

El ciclo comienza con el flanco de subida de T1 activándose ADS en el tiempo 1 (tiempo de retardo de ADS medido desde el flanco de subida de T1 al de subida de ADS). ADS se desactiva un tiempo l1 después de la bajada de T1. Las líneas de direcciones se hacen estables en el tiempo 2 desde el flanco ascendente de T1, mientras que la señales de estado identifican el ciclo de fetch en el tiempo 3 desde el flanco ascendente de T1. A partir de éste momento el uP queda a la espera de que la memoria le entregue la información (el código de instrucción en este caso).

La memoria pondrá la información requerida en el bus de datos algún tiempo después de ser válidas la dirección y las señales de control. Este tiempo se denomina tiempo de acceso. La memoria indica que la información está disponible sobre el bus de datos activando la señal /READY del uP.

El uP muestrea el estado de la señal /READY en el flanco ascendente de T3. En este instante, si /READY está activado (/READY=L) se lee la información en el bus de datos local y se termina el ciclo en curso. Si en ese momento de muestreo la señal /READY no está activa (/READY=H), el u*P* espera hasta el siguiente flanco ascendente del reloj para muestrear de nuevo la señal /READY.

El uP requiere que el dato sea estable un tiempo antes del final del ciclo (instante del muestreo de la señal /READY cuando /READY=L, tiempo 6 medido desde que el dato es válido hasta el momento de lectura del mismo por parte del u*P)*, este tiempo se denomina de setup del dato. Así mismo, las información a la entrada del uP ha de cumplir con un tiempo estable después de la lectura por parte de uP, tiempo de hold respecto al flanco en que es leido (tiempo 7).

La activación de la señal /READY indica al uP que el dato está disponible para ser leido del bus y así poder acabar el ciclo. El uP exige a la señal /READY, para poder llevar a cabo su misión, que sea estable en el estado activado durante un periodo de tiempo antes del flanco de reloj con el que se desea acabar el ciclo (tiempo de setup del /READY respecto al reloj. Tiempo 8), así como un tiempo de hold respecto a este mismo flanco (tiempo 9).

Cada ciclo de reloj que se añade al ciclo mínimo que puede desarrollar el uP en espera de la

llegada del /READY se denomina estado de espera o ciclo de espera (wait state).

Lo ideal es hacer que el uP funcione sin estados de espera ya que de esta forma se consigue la máxima velocidad. Esto no es posible en todos los casos y la señal /READY se utiliza para adaptar las diferencias de velocidad entre el uP y los demás elementos (memoria y E/S).

Evidentemente cada ciclo de espera que se introduce en cualquier ciclo máquina del uP penaliza la velocidad de ésta. La tabla siguiente muestra la penalización en % para diferentes estados de espera introducidos en los ciclos del uP.

Como podemos comprobar en la tabla, los estados de espera perjudican seriamente al tiempo de ciclo máquina, tanto más cuanto más rápida estados de espera se introduzcan para realizar el ciclo.

Esta influencia de los estados de espera se difumina algo cuando se relaciona con los tiempos de ciclo de instrucción ya que si bien para el caso en que el ciclo instrucción sea igual al ciclo máquina, el impacto de los estados de espera es el mismo, en los casos en que el ciclo instrucción sea diferente (mayor, por supuesto) al ciclo máquina, el peso de los tiempos añadidos por los estados de espera se reduce.

Duracion ciclo T	Estados espera por ciclo	Incremento ciclo %
3	1	33
	2	66
	3	100
	4	133
4	1	25
	2	50
	3	75
	4	100
5	1	20
	2	40
	3	60
	4	80

Influencia de los estados de espera añadidos sobre los ciclos máquina.

Supongamos, por ejemplo, una instrucción que necesite tres ciclos máquina con 3T de duración y 1 WS cada uno de ellos y que en su ejecución se invierta 16 ciclos de reloj. Esto significa que la incidencia de los tres WS sobre el tiempo total del ciclo de la instrucción es del 23% frente al 33% en un ciclo máquina.

El ciclo termina con la desactivación de las señales de estado con un tiempo de retardo 10 respecto al flanco descendente de 13.

Ciclo de lectura en memoria.

Es un ciclo que desarrolla el uP cuando necesita leer información, no de código, de la memoria, ya sea de la de programa o de la de datos. La figura 15 muestra el cronograma correspondiente a este ciclo que presenta el mismo aspecto que el ciclo de fetch discutido antes salvo el estado de las señales de estado que adquieren el estado lógico adecuado a la identificación del ciclo en curso.

La figura 15 muestra el ciclo de lectura mínimo (sin estados de espera, O wait states).

Ciclo de escritura en memoria.

El uP utiliza este ciclo para depositar información (dirección o dato) en la memoria. La figura 16 muestra cronograma correspondiente a este ciclo. En esta ocasión, el uP vuelca al bus de datos la información a ser escrita. El tiempo 8 es el tiempo que el uP tarda en volcar el dato al bus medido desde el flanco de subida de T2 al dato válido. El uP espera, igual que en los casos anteriores, que la circuitería lógica de la memoria active la señal /READY para acabar el ciclo. Conforme ésto sucede, el uP espera aún un tiempo para retirar los datos (tiempo 9) para asegurar un tiempo mínimo de hold en el bus. El tiempo 9 se mide desde el flanco ascendente siguiente al final del ciclo hasta la retirada del dato (paso a triestado).

Ciclo de lectura en E/S.

Es un ciclo similar al de lectura en memoria salvo que ahora las señales de estado toman la combinación correspondiente. La figura 17 muestra el cronograma correspondiente a este ciclo.

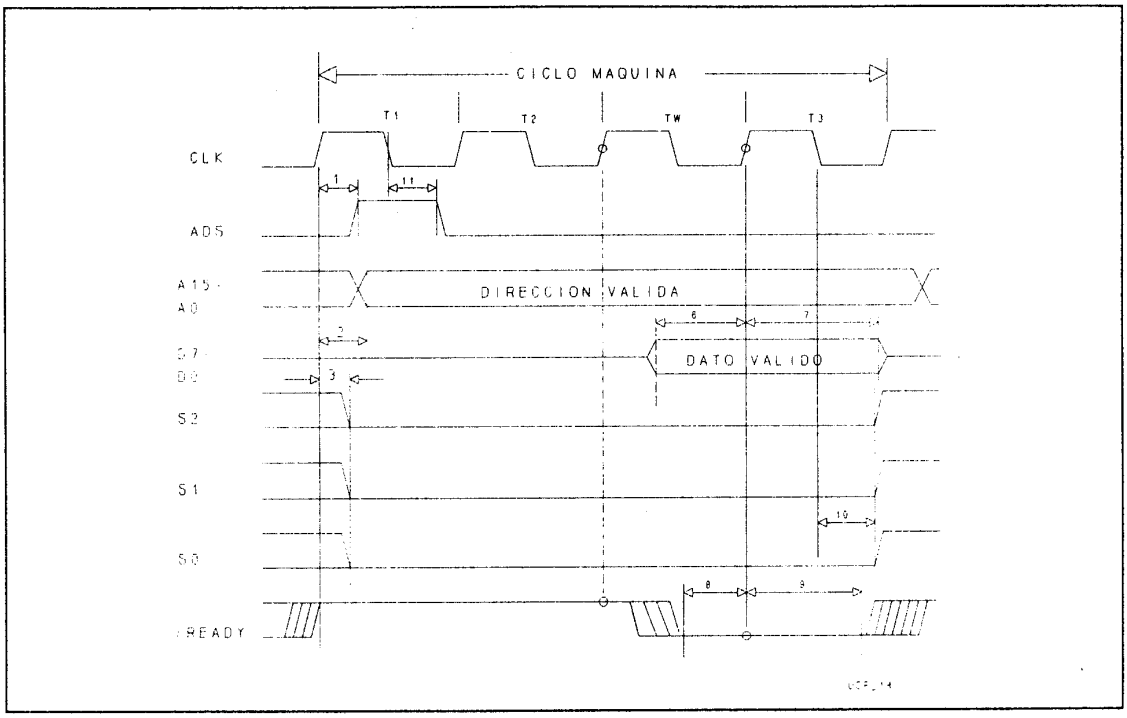


Figura 14. Cronograma externo del ciclo Fetch.

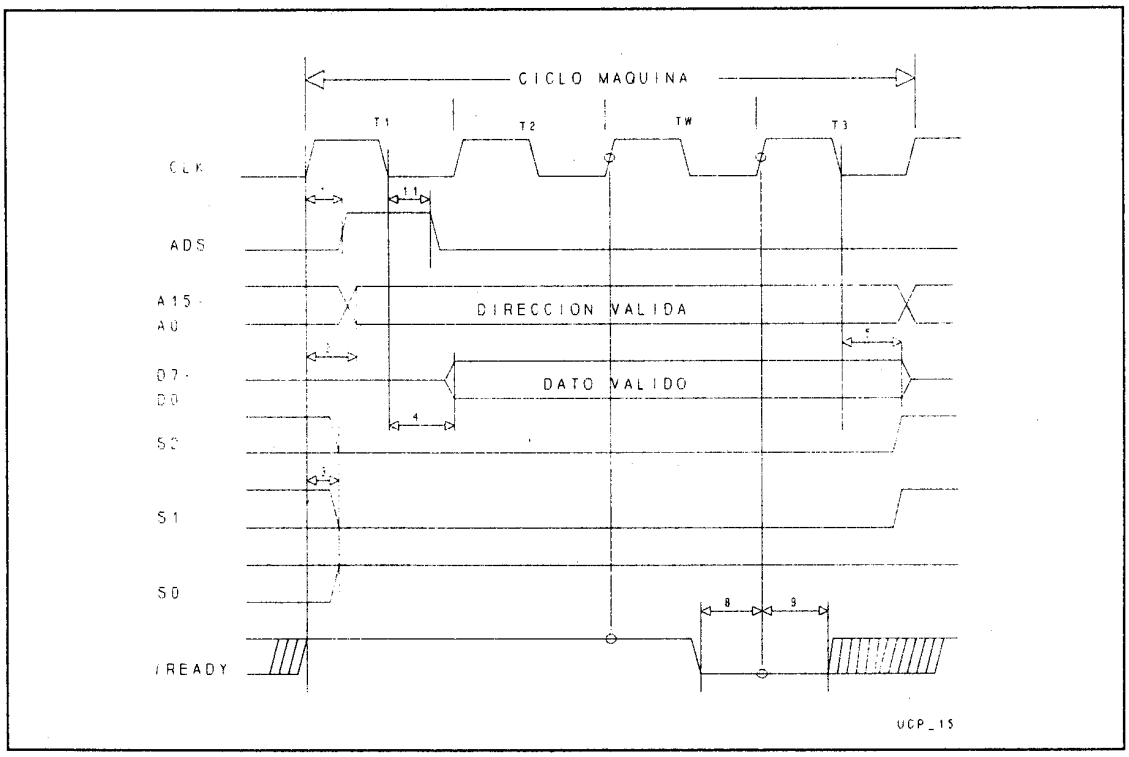


Figura 15. Cronograma del ciclo externo de escritura en memoria**.**

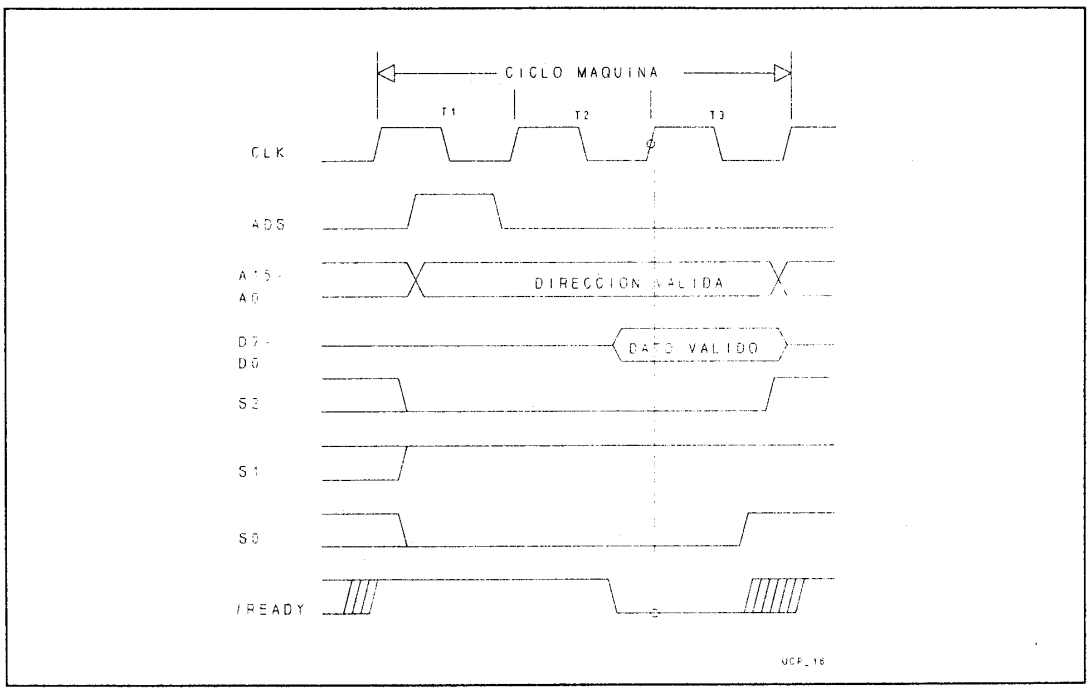


Figura 16. Cronograma del ciclo externo de escritura en memoria**.**

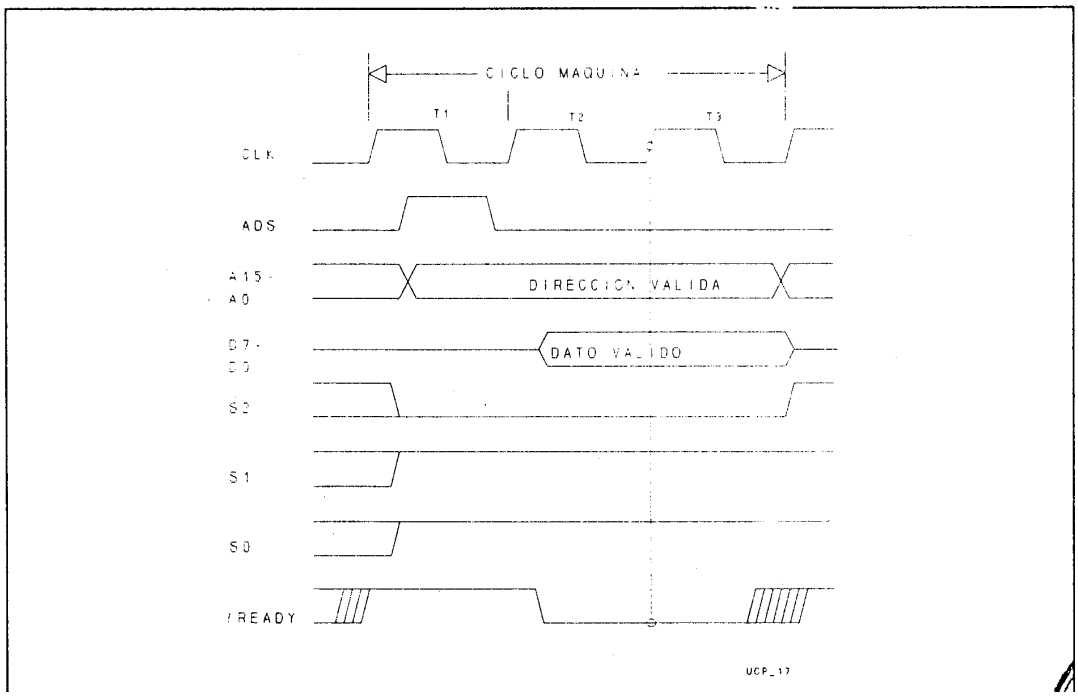


Figura 17. Cronograma del ciclo externo de lectura en E/S.

Ciclo de escritura en E/S.

Es similar al ciclo de escritura en memoria salvo que está identificado como de E/S por las señales de estado del uP. La figura 18 muestra el cronograma de este ciclo.

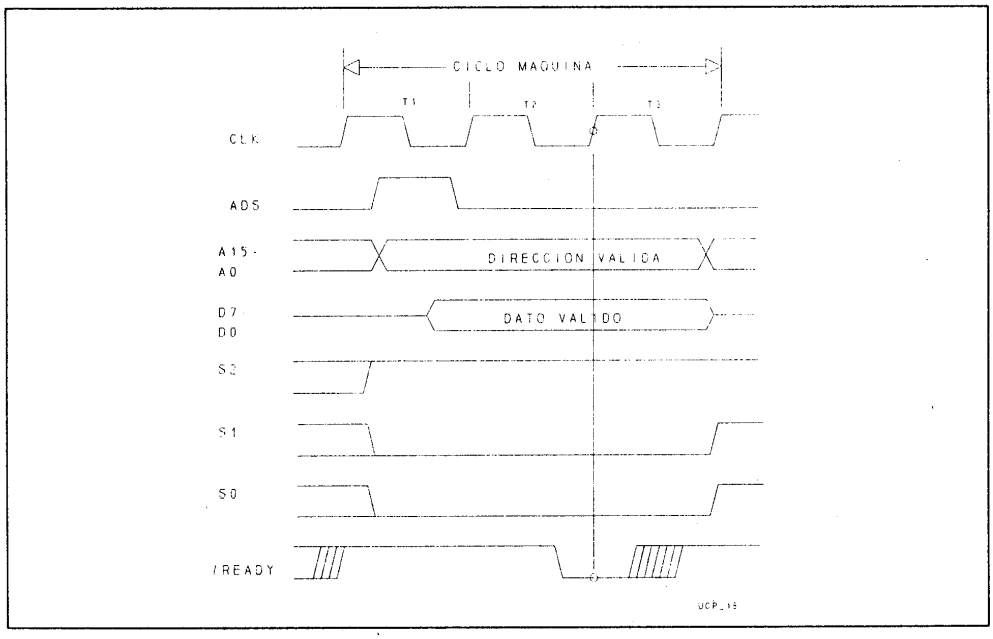


Figura 18. Cronograma del ciclo externo de escritura en E/S.

Ciclo de reconocimiento de interrupción.

Cuando el uP reconoce una interrupción enmascarable realiza un ciclo especial para indicarlo ya que afecta al funcionamiento de los elementos exteriores a ella cuando el uP funciona con interrupciones vectorizadas o autovectorizadas. La figura 19 muestra el desarrollo del "ciclo, en ella podemos ver 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 uP el final del ciclo. En, este caso el uP no necesita entregar dirección alguna.

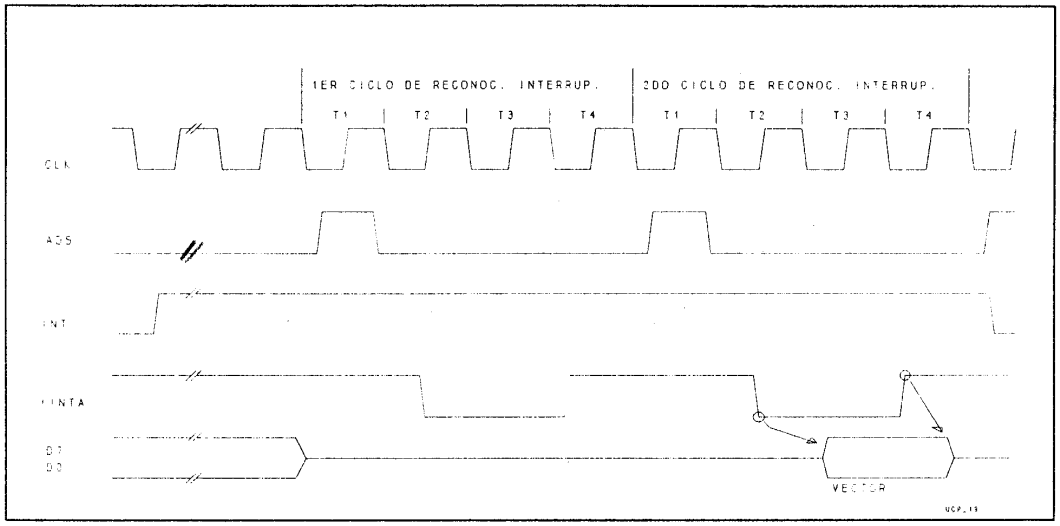


Figura 19. Ciclo INTA.

Ciclo Hold.

El uP realiza un ciclo hold cuando se activa la señal HOLD. Cuando esto sucede, el uP acaba el ciclo máquina en que está en curso y a continuación entra en estado hold.

La situación de hold del uP es señalizada por la señal HDLA. Cuando HDLA se activa (HDLA=H), el uP está en estado hold, con ello indica que las líneas de direcciones, datos y control del uP están en alta impedancia de forma que otro dispositivo puede tomar el control del bus local.

La señal HDLA no pasa a triestado ya que ha de señalar la situación de hold del uP. Esta situación de desconexión del uP de los buses se mantiene mientras que esté activa la señal HOLD. Durante este tiempo, el uP 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.

Ciclo de Halt.

El uP 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 uP se detiene en su ejecución y sólo sale de esta situación si recibe una interrupción. El cronograma de la figura 20 corresponde a este ciclo.

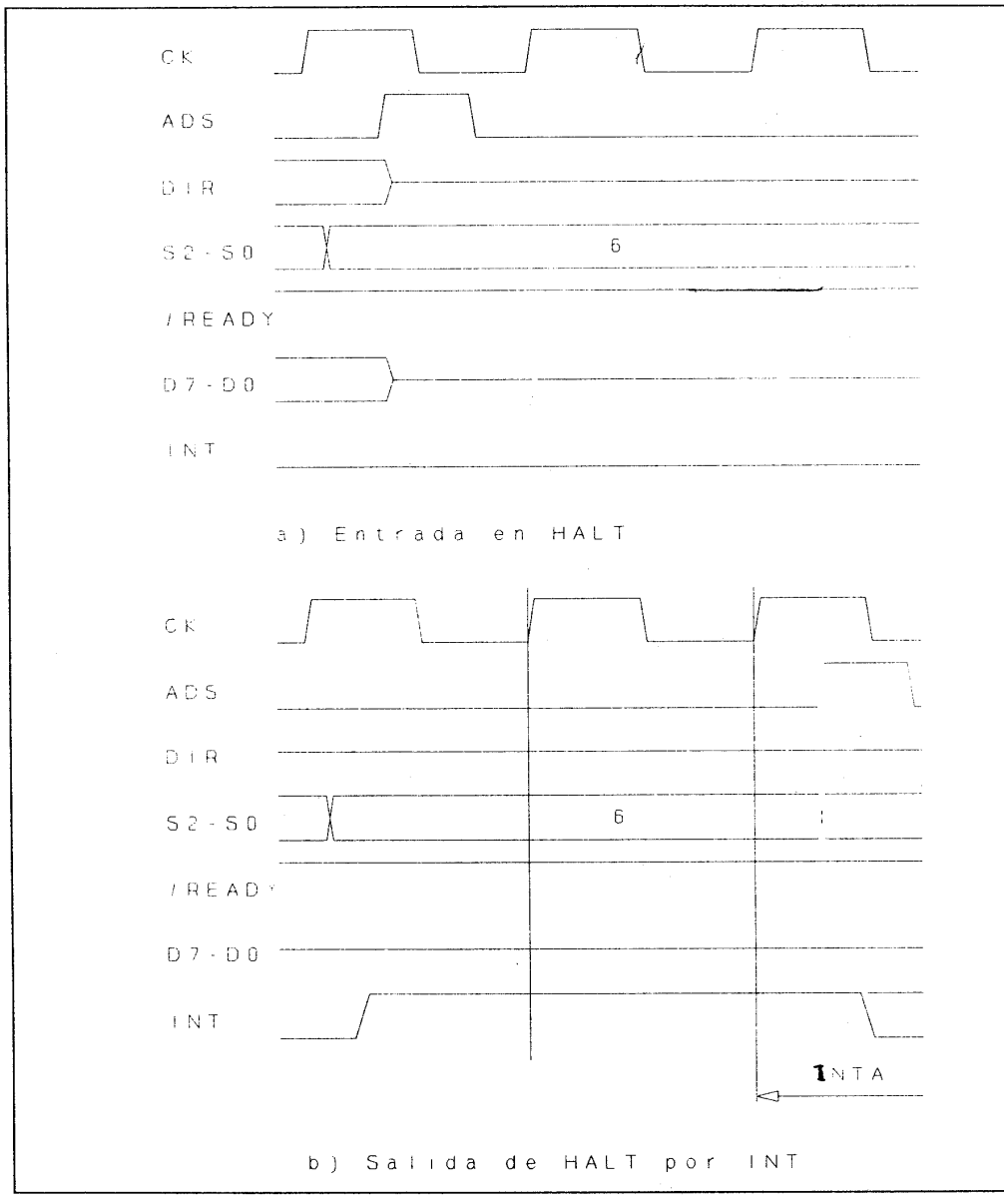


Figura 19 Cronograma del ciclo HALT.

Cuando la señal HOLD se desactiva, el uP 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 la señal HOLD desactivada, es desactivada la señal HLDA para indicar la salida del estado hold del uP.

La figura 21 muestra el cronograma desarrollado por este ciclo. En ella podemos ver que HOLD sólo se muestrea al final del ciclo máquina en curso (en el mismo instante que se muestrea ready activado), y que la respuesta por parte del uP se produce a partir del siguiente flanco de reloj. En ese momento se desconectan las linaes de los buses de direcciones y datos y las señales de control. También en ese momento se activa HDLA.

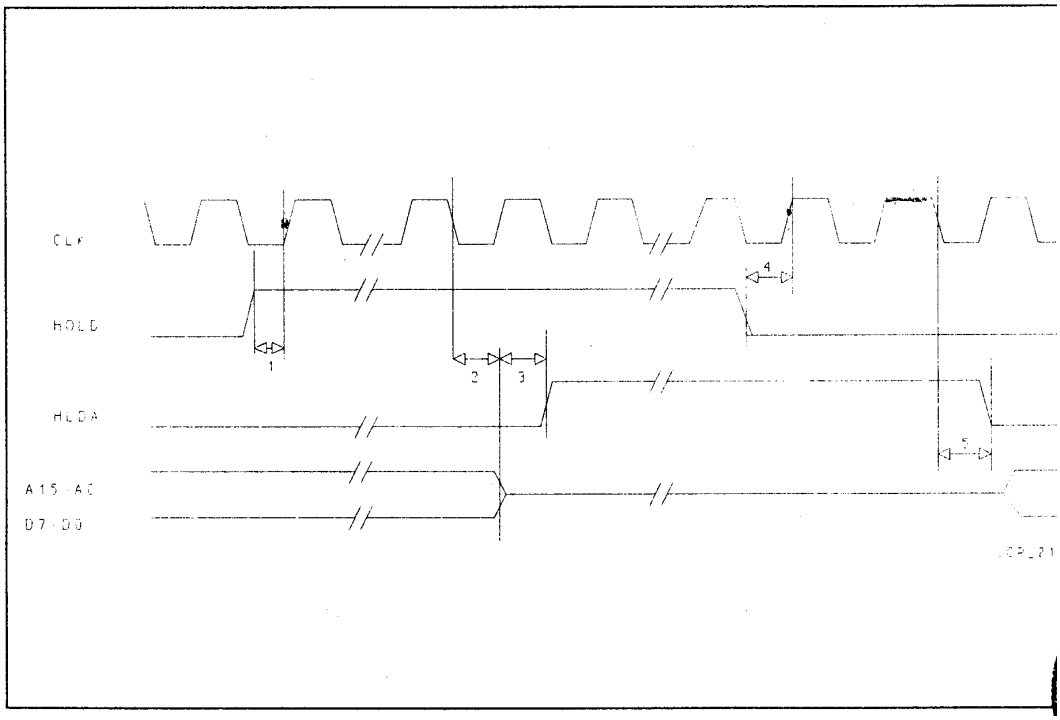


Figura 21. Cronograma del ciclo HOLD.

El tiempo que transcurre desde que se activa HOLD hasta que el uP responde correspondientemente se denomina tiempo de latencia del hold y es variable de un uP a otra 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 u*P* para funciones internas en las que no se producen ciclos en el bus local.

4.3.- Definiciones de tiempos.

Los cronogramas mostrados de los distintos ciclos que desarrolla el uP se acotan en tiempo respecto al reloj CK de la unidad. El uP modelo que estamos utilizando evoluciona con el flanco ascendente del reloj CK. Por ello todas las acotaciones van referidas siempre a un flanco de subida de CK. Los números que aparecen en los distinto cronogramas es la referencia a la cota en tiempo característica del uP.

Los ciclos de reloj necesarios en cada ciclo de u*P se* nombran como T1, T2, etc. El ciclo máquina mínimo de nuestro uP consta de tres ciclos de reloj (TI, T2 y T3). Todos los ciclos máquina comienzan con la subida de la señal ADS y terminan con la llegada de la señal /READY. ADS la suministra el uP indicando el comienzo del ciclo mientras que /READY la suministra el dispositivo destino de la operación (memoria o E/S), por lo que son los dispositivos externos al uP los que determinan la duración exacta del ciclo. Cuando el ciclo es de lectura (de código o de dato sobre memoria o E/S), /READY se activará de tal forma que los datos sean válidos a la entrada del uP al menos el tiempo de setup. Así mismo, la propia señal /READY tiene una especificación de tiempo de setup para asegurar que el uP reconocerá el final del ciclo. Las definiciones de estos tiempos se dan a continuación.

Descripción

1 11	Retardo de ADS
2 2	Retardo de las direcciones (A15-AO) Desconexión de direcciones
3 10	Retardo (S2, SI, S0) Retardo de estado
8 9	Setup de /READY Hold de /READY
6 7 5 4	Setup de datos (D7-DO) Hold de datos (D7-DO) Hold de datos Retardo de datos (D7-DO)
11 12	Setup de INT Hold de INT
1 4 3 5	Setup de HOLD Setup de HOLD Retardo de HLDA Retardo de HLDA

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 (figura 22).

CODIGO DE OPERACION.

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 tiene 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 interpetación 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, sólo 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 al 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 altas. Nosotros describiremos las más básicos 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
- Indirecto
- Indirecto registrado
- Basado
- Indexado
- Indexado a base
- A bit

A continuación discutimos cada uno de ellos.

Direccionamiento Implícito.

Cuando la.operaión 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.

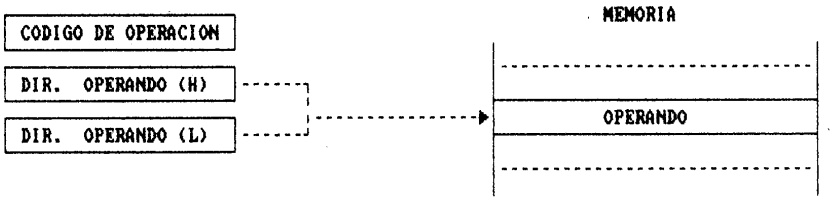
El direccionamiento es inmediato cuando el contenido del campo del operando es el propio operando.

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 OADH en el acumulador con direccionamiendo directo (LD A,OADH), es un ejemplo de este caso.

Direccionamiento Directo.

Cuando se utiliza este modo de direccionamiento, e1 campo de operando de la instrucción, no contiene al operando en sí mismo .sino la dirección, donde se encuentra realmente éste en la memoria,

Figura 22. Estructura de una instrucción.



Figura, 23. Direccionamiento directo.

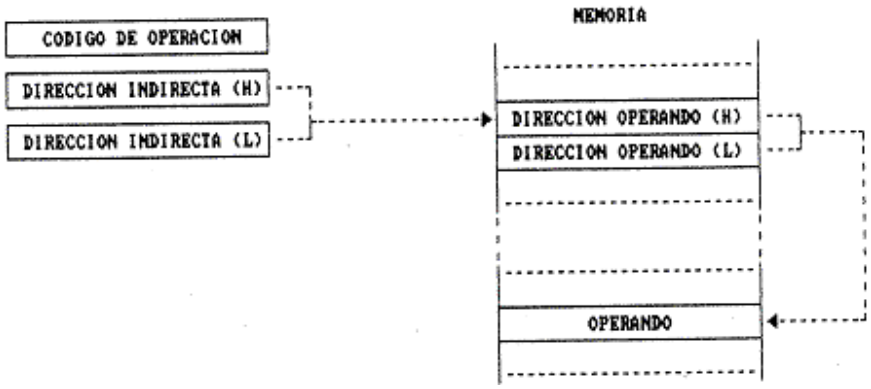


Figura 24. Direccionamiento indirecto.

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 posiones de memoria más bajas, a estas direcciones algunos autores les denominan página cero y lo describen como direccionamiento a página 0. La figura 23 muestra el direccionamiento directo.

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 24 se muestra este modo de direccionamiento.

Este modo de direccionamiento es bastante sofisticado y no es frecuente verlo implementado en UCPs 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 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 indican la dirección del operando. Esto significa que para ejecutar esta instrucción se ha de realizar con anterioridad la carga de estos registros. La figura 25 muestra este caso.

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 figura 26 muestra este modo de direccionamiento.

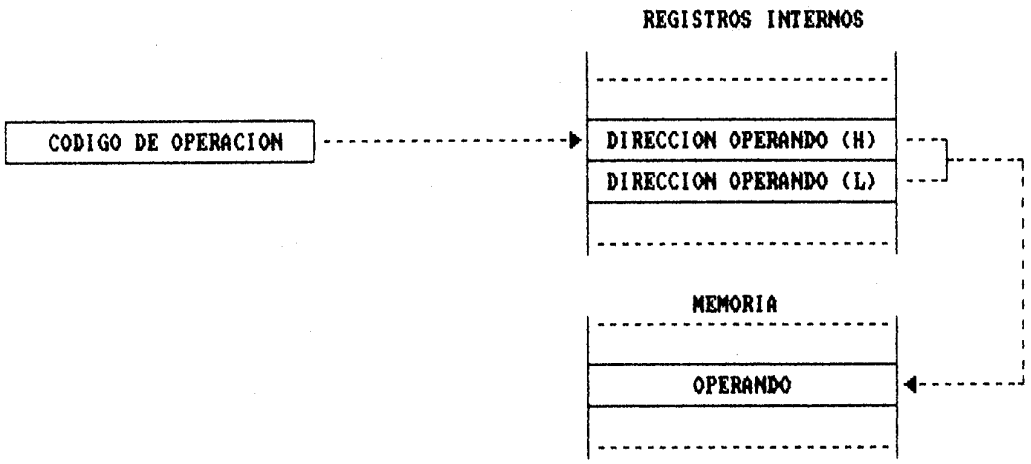


Figura 25. Direccionamiento indirecto registrado,

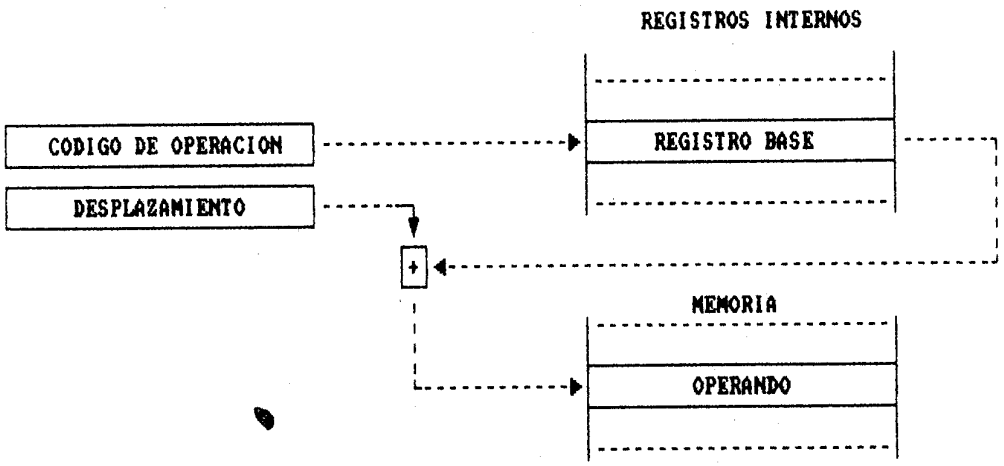


Figura 26. Direccionamiento Basado.

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. La figura 27 muestra este caso.

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. En la figura 28 se muestra este mecanismo.

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. En la figura 29 se muestra este modo de direccionamiento.

Si el desplazamiento se entrega en complemento a dos, este direccionamiento permitirá desplazarse hacia adelante o hacia atrás, respecto al PC.

Este direccionamiento es muy útil en determinados tipos de saltos. (Saltos relativos).

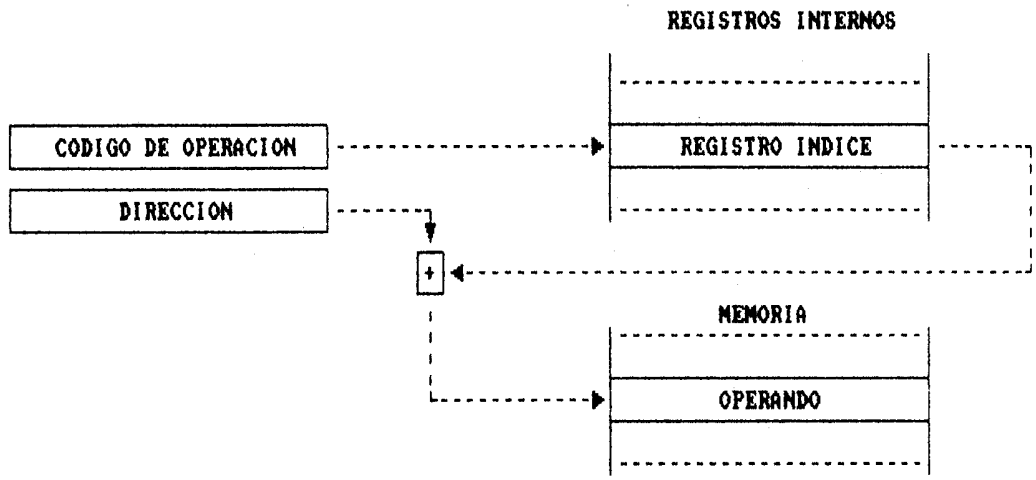


Figura 27. Direccionamiento Indexado.

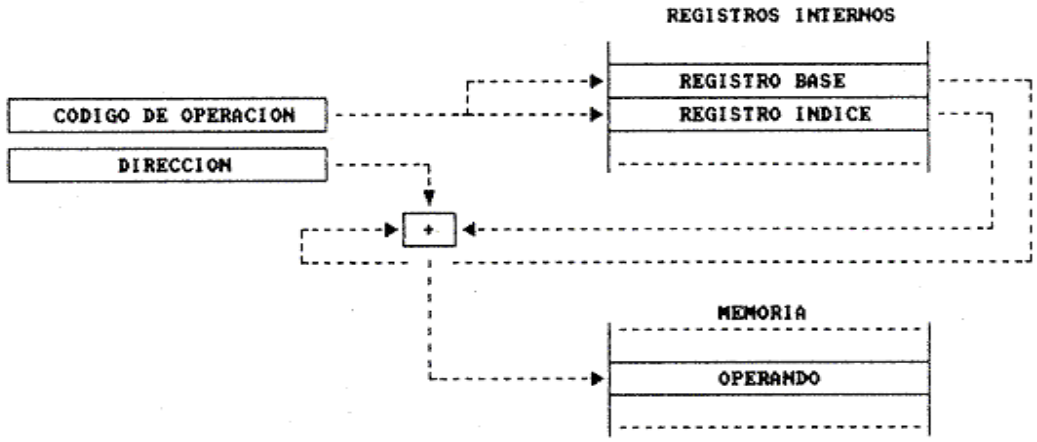


Figura. 28. Direccionamiento Indexado a Base.

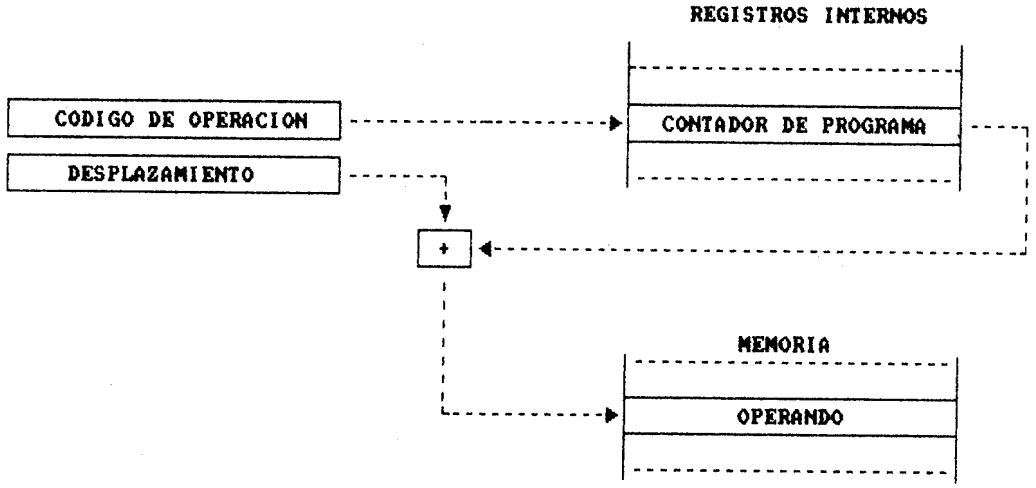


Figura 29, Direccionamiento Relativo.

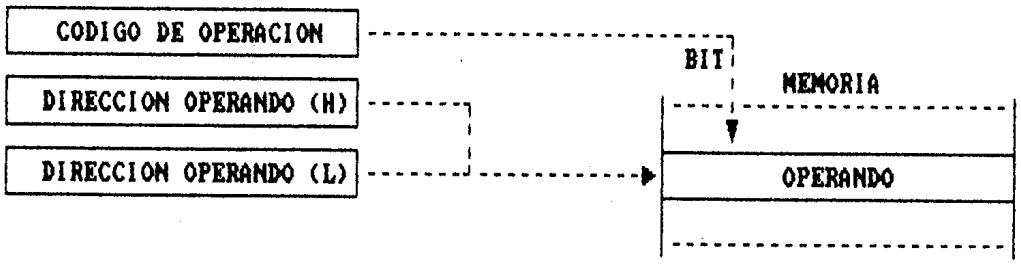


Figura 30. Direccionaimiento a BIT.

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 O a 7), necesitamos realizar otras operaciones adicionales. 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 direccionamiento 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. En la figura 30 se ha representado este modo de direccionamiento en donde se utiliza el modo de direccionamiento directo para localizar el byte en cuestión.

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 cargas.
- 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) La UCP es un dispositivo digital y como tal, sólo puede tratar con los niveles lógicos H y L que en lógica positiva se corresponde con el ‘1’ y el ‘0’ del algebra Booleana. Por lo tanto, para indicarle a la UCP que deseamos realizar una suma del acumulador con en 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 tendremos que decirle a la UCP que realice la instrucción:

11000101 00100000 o lo que es lo mismo: C5 20 H

De esta forma le hablamos directamente a la UCP pero nos encontramos con el problema de que el códiqo 11000101 no es demasiado cómodo de entender para nosotros. Además nos es muy fácil confundirnos con un código parecido como el 011000101 que no indica una suma sino una carga de registro interno.

Por ello, las instrucciones las representamos por un conjunto de letras que denominamos nemónico de la intrucción. Siempre la escribimos con mayúsculas. 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 nemónico.

Algunos autores denominan a este nemónico como lenguaje ensamblador o lenguaje máquina. Si escribimos nuestros programas en nemónico, 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 nemónico y obtener el código binario correspondiente. Esta tarea la dejamos para que la realice un programa que es capaz le interpretar el programa en nemónico y producir una salida en código binario. Este programa se denomina programa ensamblador.

5.3.1.- Funcionamiento de las instrucciones

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

Instrucciones de carga.

El grupo de instrucciones de carga permite el movimiento 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 LD (LOAD) y se dividen en tres grupos, las que cargan registros internos de 8 bits, las gue 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.

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 al 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).

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.

Instrucciones de saltos.

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 todo él, 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 existen dos posibilidades y hay 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

A continuación hablamos de cada uno de ellos.

Saltos cortos.

Se denominan así los saltos cuya longitud está limitada a un determinado rango de direcciones.

En los uPs de 8 bits, este tipo de salto utilizan un sólo byte para dar o calcular la dirección del salto con lo que la longitud máxima, del salto es de 128

Saltos largos.

Al contrario de los saltos cortos, éstos permiten alcanzar la. totalidad- de.las direcciones posibles de la memoria. Por lo tanto, la instrucción completa constara del código de operación y de un operando que será la dirección completa del salto o una referencia a ella, 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)

Saltos absolutos.

En los saltos absolutos, la dirección a la que hay que saltar no tiene una relación directa con la pos ición donde se encuentra la instrucción de salto.

La dirección del salto se especifica por diferentes métodos, según sea el direccionamiento de la instrucción del salto.

Saltos relativos.

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 esta 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 direccionemiento 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.

El salto máximo 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 del 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 en retroceso. cuando es en el 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 han de correr en posiciones de memoria no conocidas de antemano, 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 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 necesarias. Por ejemplo la función de conversión de código hexa 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 esta función desde cualquier punto del programa.

Las subrutinas tienen una estructura determinada que no entraremos a discutir en este apartado ya que corresponde al sw, pero si es necesario indicar que dispone de una entrada y de 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 instrución 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 la 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 subrutina. 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. Más adelante discutiremos este punto.

Ninguna instrución de salto modifica el estado del registro de estado salvo el retorno de interrupción.

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 máscara de interrupción, etc.

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.

Otras instrucciones.

Algunos autores incluyen en este grupo instrucciones como NOP (no operación).

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.

Observar, así mismo, que quedan aún 84 posibles códigos sin utilizar (256 - 172 = 84).

6.- Interfaz con el bus del sistema.

Hasta ahora hemos visto cómo la UCP se relaciona con el entorno en que se encuentra desde el punto de vista teórico del modelo utilizado. 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. Sin embargo este bus local tiene una relación muy estrecha con el bus del sistema, ya sea directa o indirectamente. En muchas ocasiones nos encontraremos que el bus del sistema y el bus local son la misma cosa, tienen la misma definición. No es este el caso de nuestro modelo teórico puesto que la definición de señales en nuestro bus no se corresponda completamente con la definición del bus de nuestro sistema. Esto es así para desarrollar el caso más general que cubre la mayoría de las aplicaciones. 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 (ver figura 31):

- Interfaz de direcciones**.**

- Interfaz de datos.

- Interfaz de control.

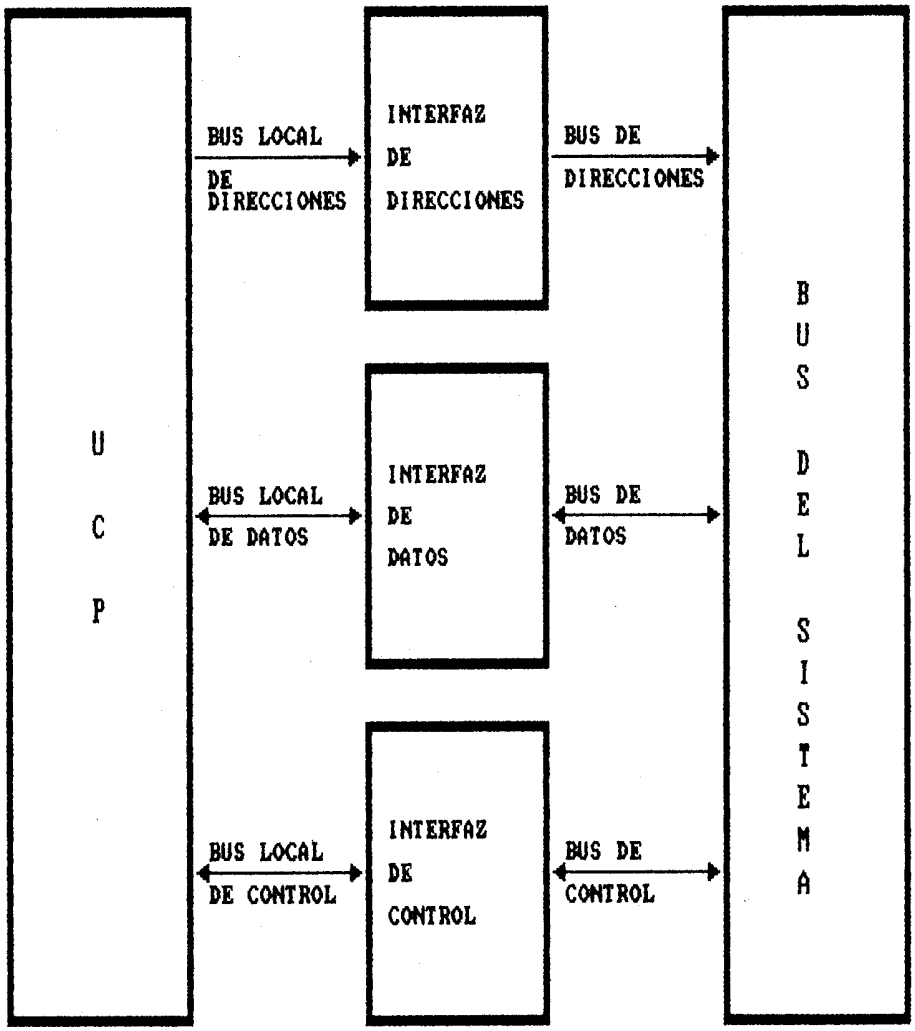


Figura 31. Interfaz de la UCP con el bus del sistema.

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 (figura 32).

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

Interfaz de datos.

El bus de datos local de la UCP así como el del sistema, es bidireccional. Esta característica hace que el interfaz de datos se implemento con dispositivos transceptores (registrados o no). Estos

transceptores disponen de varias señales de control. Básicamente requieren dos señales de control, cuando no son registrados, más una señal de reloj en caso de ser registrados. La 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. La figura 33 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 una 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 la señales no presentes y adaptar las concordantes.

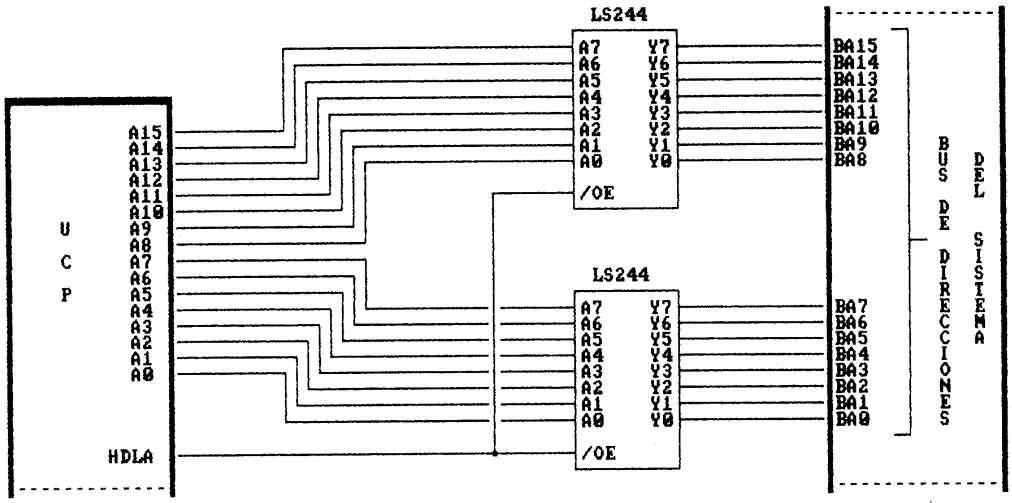


Figura 32. Interfaz de direcciones de la UCP.

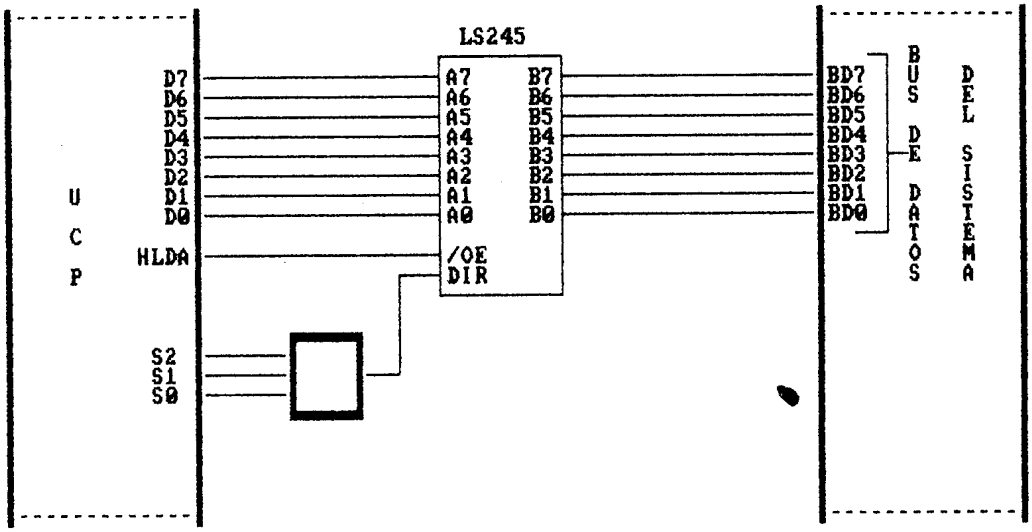


Figura 33. Interfaz de datos de la UCP.

Este interfaz tiene los tres 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 34 muestra esta conexión en donde podemos ver que la señales HOLD, HLDA se corresponden con esta parte del interfaz.

En la figura 34 podemos ver que la señales /RD, /WR, /MEM, /ES, /HALT se obtienen del bloque generador cuyas entradas son las señales S2, SI, SO 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 34. Todas ellas han de ser atendidas por la UCP.

7. Interrupciones.

El funcionamiento de un sistema basado en microprocesador raramente no se ve influenciado por los elementos externos. La forma de coordinar la dedicación del sistema a los elementos externos es por medio de las interrupciones del sistema. Estas interrupciones del sistema son atendidas por la UCP que abandona la ejecución del programa en curso para ello.

El tema de las interrupciones es tan importante dentro de los SBM (sistemas basados en microprocesador) que dedicamos un capítulo independiente para ello.

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 atiende a la interrupción (no la reconoce). Si está desactivada sí la reconoce.

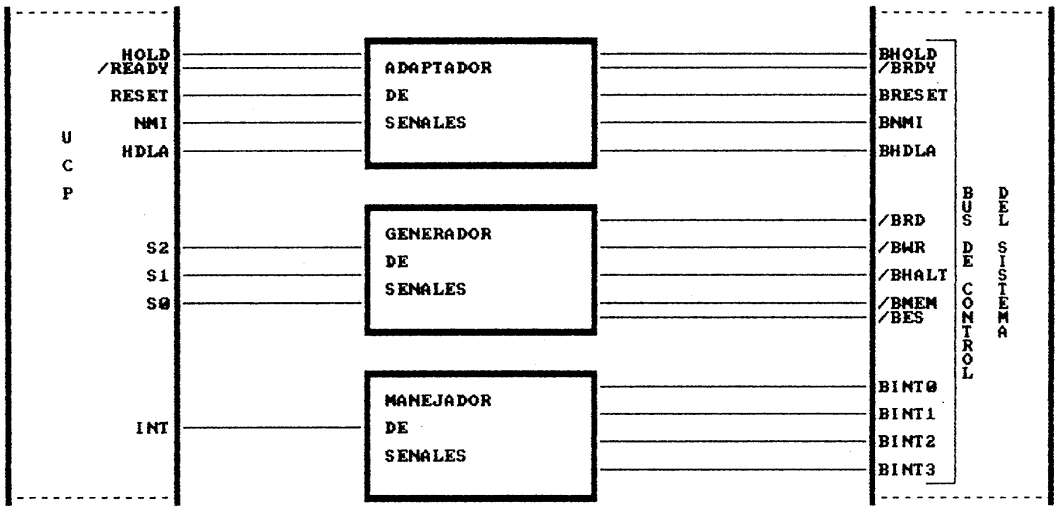


Figura. 34, Interfaz de control de la UCP.

- **Vector de interrupción**. Información volcada al bus de datos durante un 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 la atención a una 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 enmacarable. Estos nombres hacen referencia al bit de máscara de interrupción que dispone la UCP en su interior. La máscara de interrupción sólo es válida para la interrupción enmascarables.

Prioridad de las interrupciones.

Ya que la UCP admite dos tipos de interrupciones, hay que conocer las características respecto a ellas y de su arbitración por parte de la UCP. Esto último es lo que vamos a discutir a continuación.

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 directamente congela 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 35 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 si la encuentra activa entra directamente al modo ADM independientemente de si existe o no petición de interrupción NMI o INT. Si el READY corresponde con un final de ciclo instrucción, la UCP verifica el estado de las peticiones de interrupción NMI e INT y si hay alguna activa la memoriza en un biestable de interrupción pendiente. Cuando la UCP no está en ADM 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 dentro de la UCP, la señal HOLD es la más prioritaria, a continuación le sigue la NMI y, por último la INT.

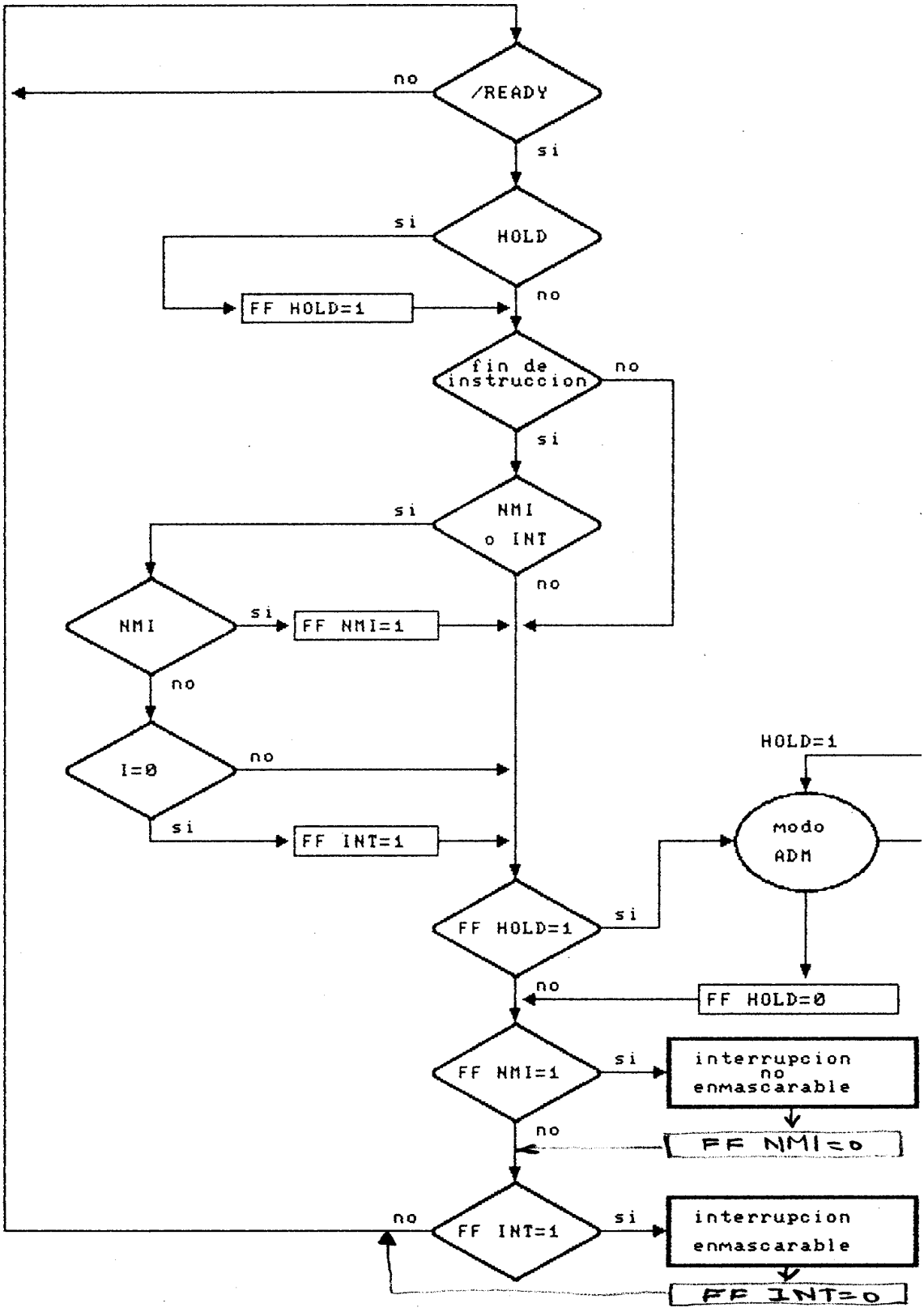


Figura 35. Diagrama de flujo de la secuencia de interrupciones.

Interrupción no enmascarable (NMI).

La llegada de una NMI al uP implica de modo automático que la uP 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 en el momento de la llegada de NMI.

La figura 36 muestra la secuencia funcional que desarrolla el uP 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 memoria fijada por el fabricante y a la que accede la UCP por medio de un salto a ella. Para poder mantener la coherencia lógica del programa y poder continuar el trabajo en el punto en que fue interrumpido, la uP 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 la normal en las funciones del HW a la llegada de una interrupción. 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 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 Cl 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)**.**

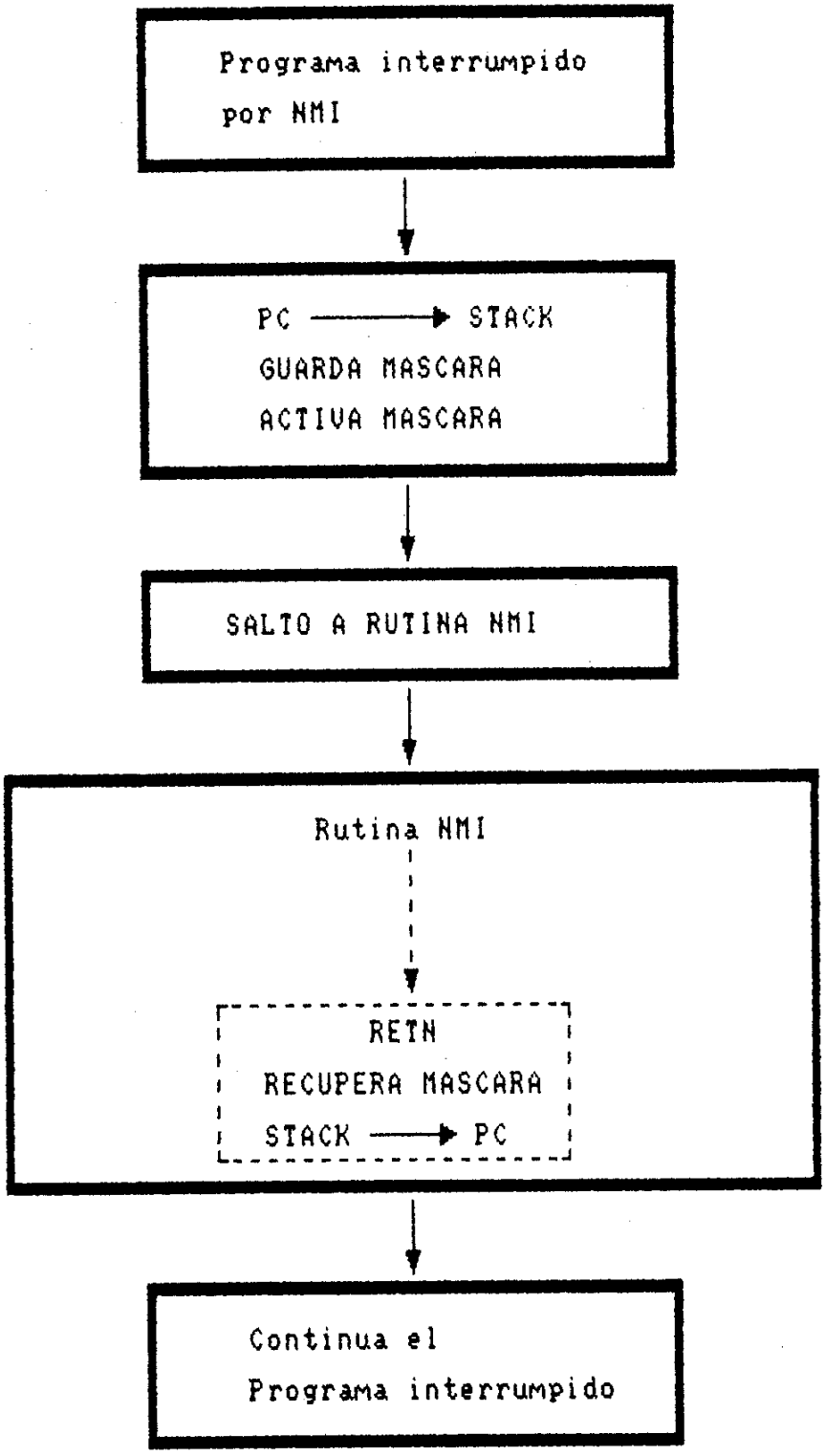


Figura 36. Secuencia funcional seguida por la UCP a la aceptación de una interrupción NMI.

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 corre sobre 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 enmacarable, mientras que cuando hayamos de hacer referencia a la NMI especificaremos que se trata de una interrupción no enmascarable.

La aceptación de una interrupción enmascarable 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 vectorizadas.

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 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 37 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 esa dirección comenzando en ese punto la ejecución de la rutina. La figura 38 muestra la secuencia desarrollada en este modo de interrupción.

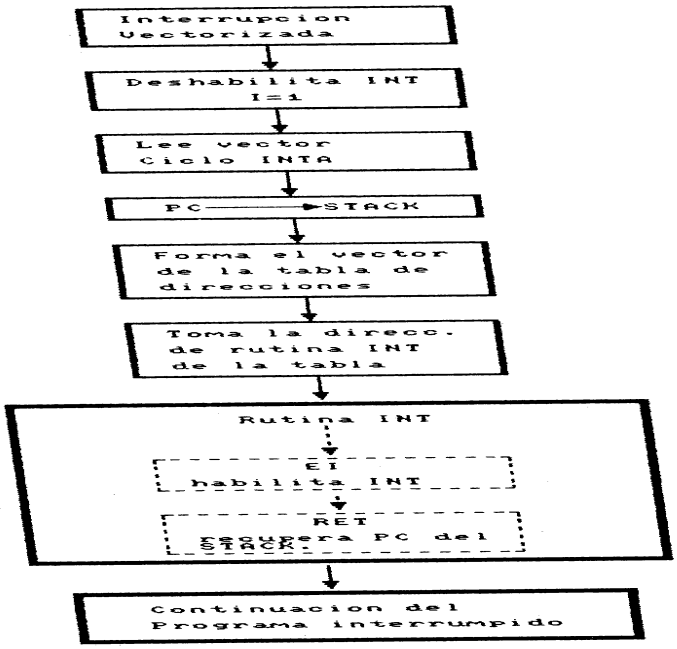
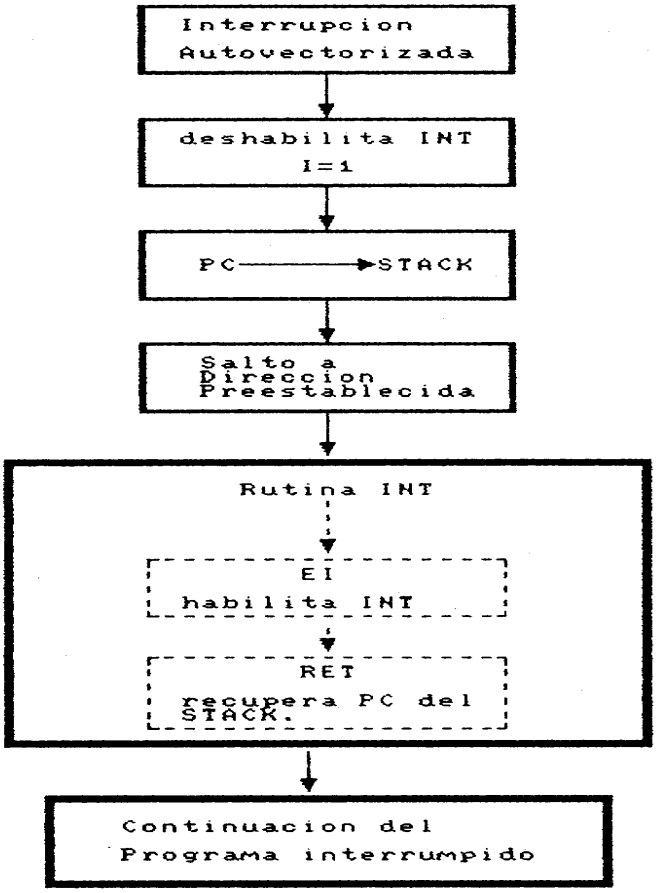


Figura 37. Secuencia de la interrupción vectorizada.



Figura, 38. Secuencia de la interrupción autovectorizada.

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 (figura 38). Este código lo vuelca al bus de datos el dispositivo que genera la interrupción durante el ciclo de reconocimiento de ésta. Este código corresponde 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.

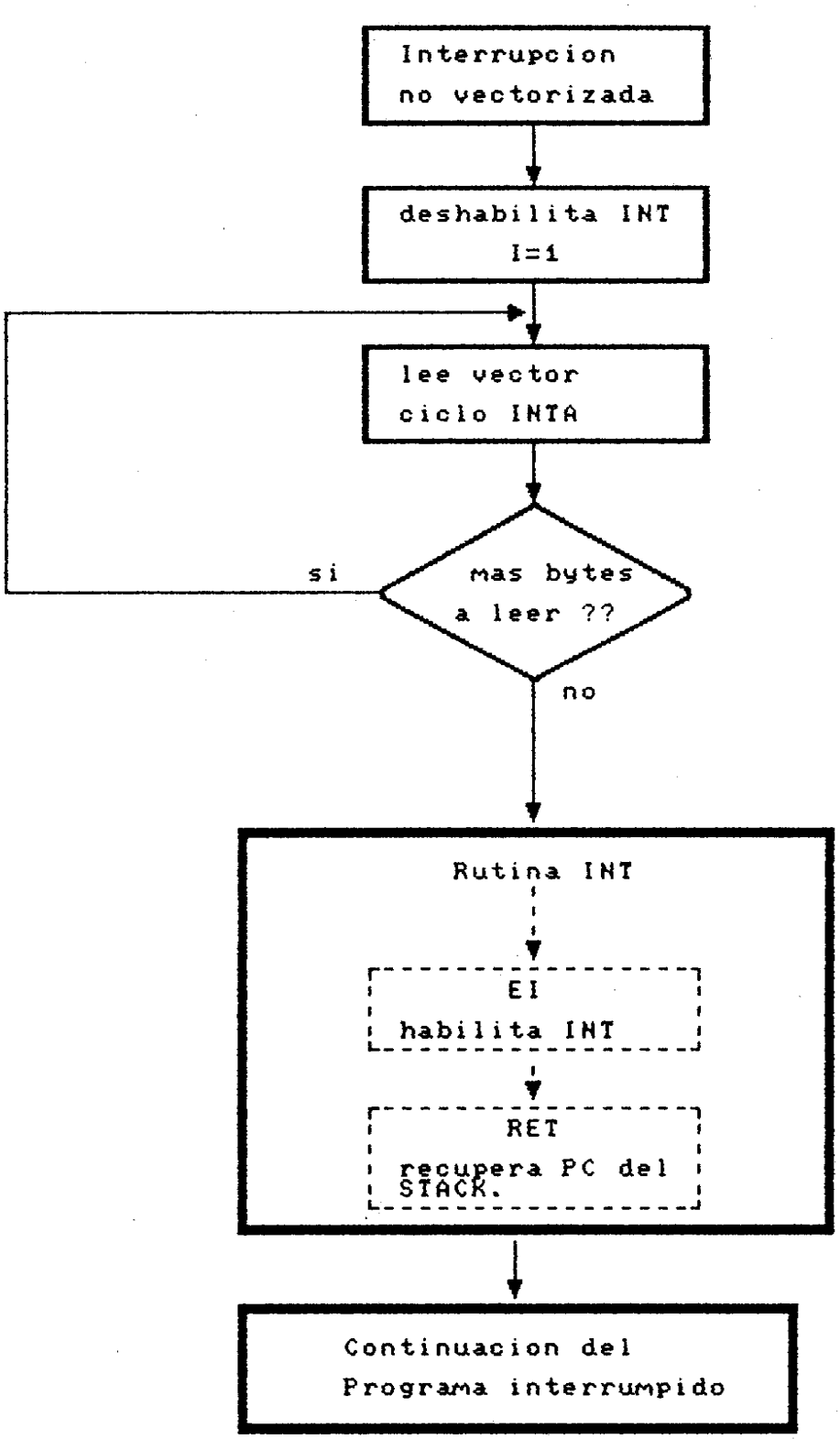


Figura 39. Secuencia de la interrupción no vectorizada.