Tema 1
Introducción a los Microprocesadores

1.1. Introducción

Ante la necesidad de realizar un programa, podemos conectar los componentes (sumadores, desplazadores, memorias, etc.) de forma que realicen determinadas funciones. Esto es conocido como 'hardwired program' o lógica cableada. No obstante, podemos diseñar una configuración hardware de propósito general que realizara funciones aritméticas y lógicas. Con la primera aproximación, el sistema acepta datos y produce resultados. Con la segunda, el sistema acepta datos y señales de control y produce resultados, por lo que para cambiar el programa sólo se necesita proporcionar al sistema un nuevo conjunto de señales de control.

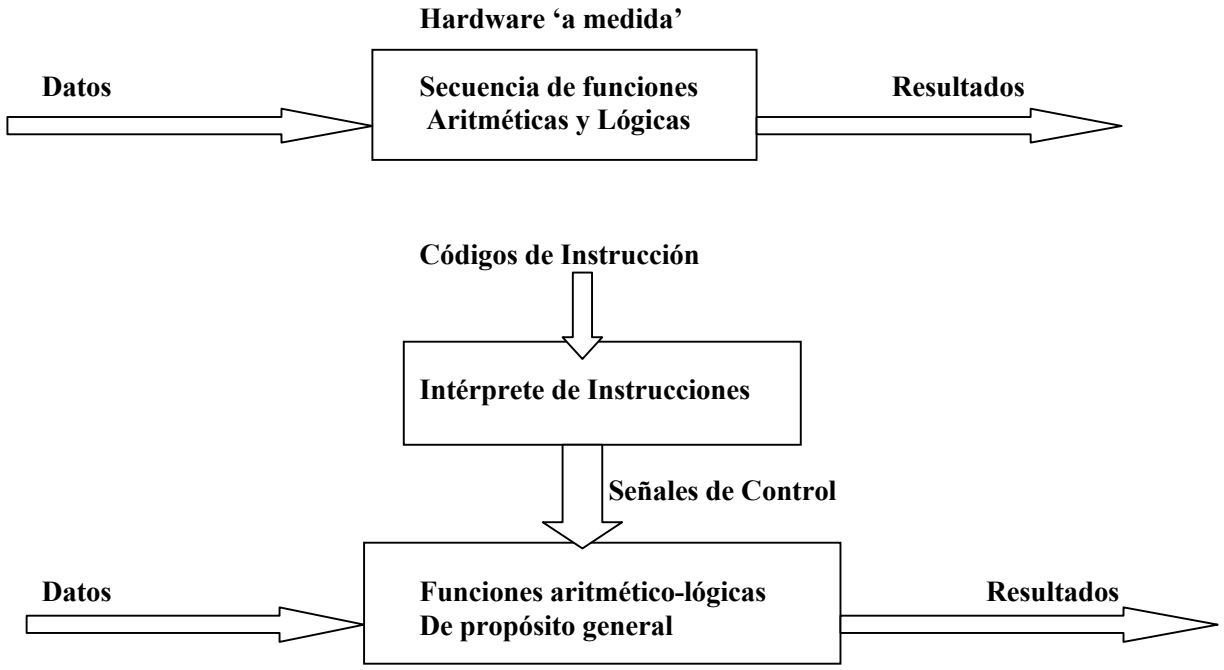


FIGURA 1.1. Aproximaciones hardware y software.

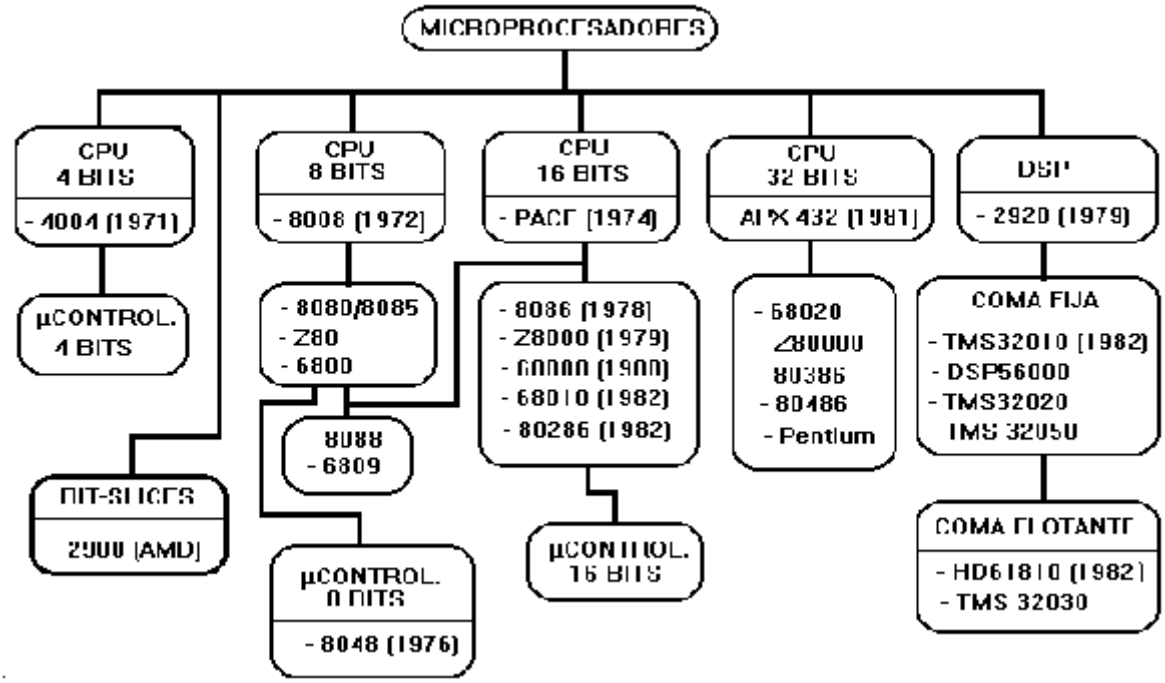
La forma de generar estas señales de control, que serían distintas en cada paso de programa, puede hacerse mediante un código único para cada posible conjunto de señales, y añadir entonces al hardware de propósito general un segmento que acepte estos códigos y genere las señales de control adecuadas. Estos códigos son básicamente las instrucciones que acepta el hardware, y el conjunto de estas instrucciones sería el software del sistema.

El módulo que implementa las funciones aritmético-lógicas junto con el intérprete de las

instrucciones constituyen la CPU del sistema. La necesidad de introducir los datos y almacenar o visualizar los resultados nos llevan a la necesidad de un sistema de E/S. Por último, debido a que los programas necesitan acceder a datos, se necesita una memoria que almacene la información y a la que se pueda acceder en cualquier posición.

Así pues, desde un punto de vista de máxima abstracción, un computador consiste en una CPU, memoria y componentes de E/S, con uno o más módulos de cada tipo. Estos componentes se interconectan de alguna forma para obtener la función básica del computador, que es ejecutar programas.

Un microprocesador es básicamente una ALU y una unidad de control integrada en el mismo chip. Un microcomputador combina un microprocesador con memoria y unidades de entrada/salida de datos. En la figura se muestra la evolución de los µP.



- **µP DE 4 BITS**: Construidos con tecnología PMOS debido a su bajo coste. Estos µP se siguen utilizando para aplicaciones de bajo coste industriales y de consumo.

- **BIT-SLICES**: utilizados **p**ara aplicaciones que requieran gran potencia de computación. Se trata de módulos idénticos que contienen CPUs con datos de pocos bits (típicamente 4). Su disposición en paralelo permite conseguir CPU de longitudes de palabra múltiplos de la básica, con la ventaja de estar realizados con tecnologías rápidas (TTL, ECL), aprovechando la estructura simplificada de cada módulo que no requiere un alto grado de integración. Su uso ha disminuido con la aparición de µP de longitudes de palabra mayores.

- **µP DE 8 BITS**: La segunda generación de µP comenzó en 1972 con el 8008 de Intel. La tecnología NMOS fue adoptada en 1974, permitiendo incrementar sus funciones y velocidad.

Posteriormente fueron desarrollándose procesadores de periféricos para estos dispositivos que soportaban otras funciones de control tales como CRT y unidades de disco. Los µP actuales de 8 bits utilizan generalmente palabras internas de longitud 16 o 32 bits (por ejemplo, el Intel 8088 y el Motorola 6809). Básicamente representan versiones de µP de 16 o 32 bits para aplicaciones de bajo coste.

- **µP DE 16 BITS:** El PACE de National Semiconductor fue el primer µP de 16 bits (1974) con el. El Intel 8086 aparece en 1978, el Zilog Z8000 en 1979 y el Motorola 68000 en 1980. Utilizan tecnología NMOS mejorada (XMOS, HMOS, etc). Las principales novedades que introducen son:
    - Modos de operación protegidos (modo de sistema y modo normal, lo que permite que un programa supervisor, como un sistema operativo, pueda realizar funciones prohibidas generalmente a aplicaciones del usuario).
    - Segmentación y protección de memoria.

- **µP DE 32 BITS:** El primer µP de 32 bits, el Intel APX 432, aparece en 1981. Existen otros procesadores posteriores que son básicamente versiones de 32 bits de modelos anteriores o que mantienen cierta compatibilidad con aquéllos (68020, 80386, etc.). Como características podemos destacar:
    - Mayores densidades de integración y velocidades de reloj → Incremento de la velocidad de proceso.
    - Optimización de arquitecturas: pipe-lines de 3 o más etapas, unidades de control de memoria integradas en el mismo chip, coprocesadores de coma flotante integrados, etc.
    - Nuevas técnicas de manejo de memoria virtual: paginación y técnicas mixtas paginación-segmentación.

- **DSP (**Procesadores Digitales de Señal): microprocesadores de uso específico optimizados para realizar procesado digital de señal en tiempo real. El primer DSP fue el Intel 2920.

1.2. Estructuras de Interconexión

El conjunto de caminos de interconexión entre los distintos módulos del computador se denomina 'estructura de interconexión'. De los diferentes tipos de estructuras de interconexión, la más usual es la interconexión de bus, que es la única que comentaremos en esta introducción, ya que básicamente es la misma que se implementa a nivel interno de la CPU para interconexión de sus elementos constituyentes.

ESTRUCTURA DE BUS

Se denomina 'bus del sistema' al bus que conecta los componentes principales del computador (CPU, Memoria e E/S). Un bus del sistema consiste generalmente de 50 a 200 líneas independientes. Estas líneas están clasificadas en tres grupos funcionales: datos, direcciones y control. Además se dispone del bus de alimentación para cada módulo.

El bus de datos transmite datos entre módulos. Consta de 8, 16 o 32 líneas (bits) generalmente, lo que se denomina anchura del bus. Este factor es clave para el rendimiento del sistema. Por ejemplo, si la anchura del bus es 8, deberá acceder 2 veces a memoria para obtener una instrucción, por una sola vez en el caso de un bus de 16.

El bus de direcciones determina la fuente o destino de los datos del bus de datos. La anchura del bus de direcciones determina la capacidad máxima de memoria del sistema. Además, las líneas de este bus se utilizan también para direccionar los módulos E/S. Generalmente, los bits más significativos se utilizan para seleccionar un módulo particular del bus y los menos significativos seleccionan una posición concreta de memoria o un puerto dentro del módulo seleccionado.

El bus de control se utiliza para controlar el acceso a y el uso de los buses de datos y direcciones, ya que éstos son compartidos por todos los módulos, por lo que es necesario establecer un control de los mismos. Las señales de control transmiten información de comando y sincronización entre módulos. Las señales más importantes son:

- Escritura en Memoria: el dato presente en el bus de datos se copia en la dirección de memoria seleccionada.
- Lectura de memoria: el contenido de la dirección seleccionada se transfiere al bus de datos.
- Escritura E/S: Se saca un dato por un puerto E/S.
- Lectura E/S: Un dato de un puerto E/S seleccionado se transfiere al bus de datos.
- Reconocimiento de transferencia: Indica que se ha aceptado un dato desde el puerto o escrito en el mismo.
- Petición de Bus: Indica que un módulo solicita el control del bus.
- Reconocimiento de Bus: Indica que el módulo peticionario ha sido habilitado como controlador del bus.
- Petición de interrupción: Indica que existe una interrupción pendiente de servicio.
- Reconocimiento de interrupción: Indica que la interrupción pendiente ha sido aceptada.
- Reloj: Usado para sincronizar las operaciones del bus.
- Reset: inicialización de los módulos.

JUAN F. GUERRERO MARTÍNEZ
JOSÉ V. FRANCÉS VILLORA

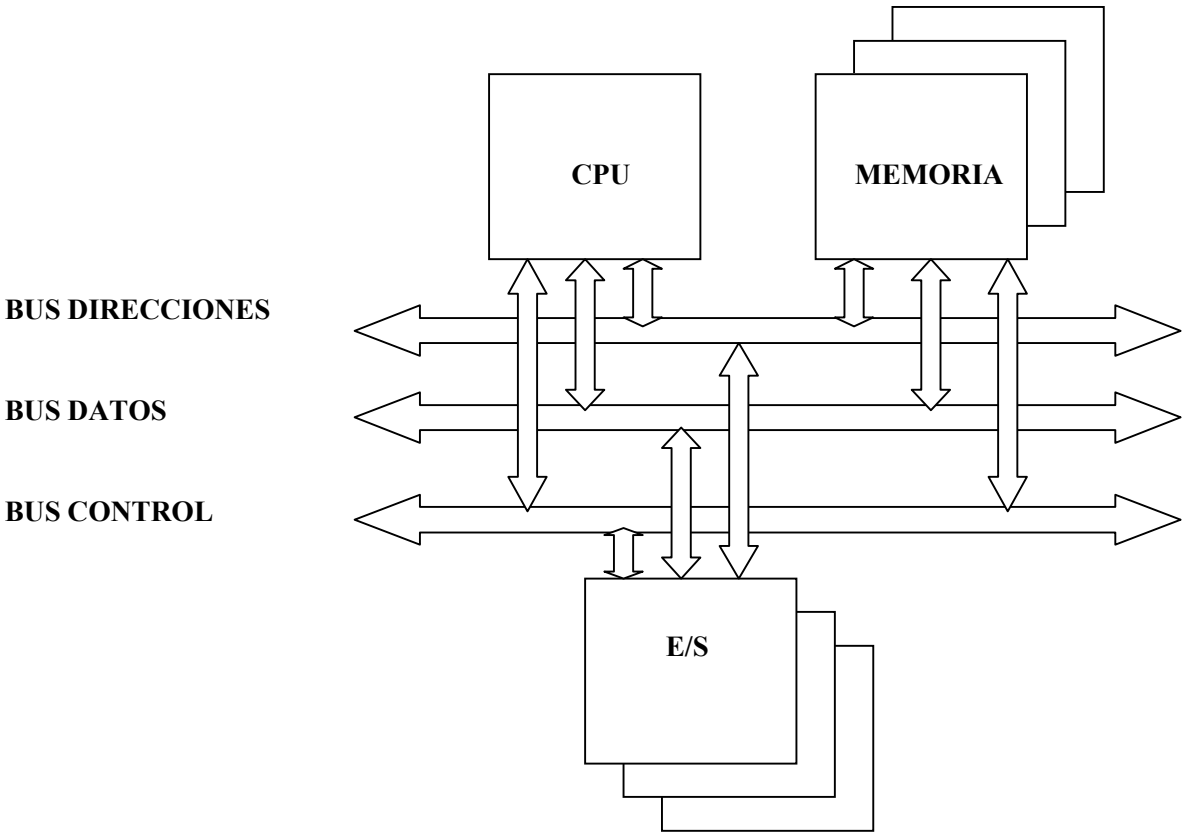


FIGURA 1.2. Estructura de interconexión de bus.

El bus opera de la siguiente forma:

- Un módulo desea enviar datos a otro: 1) obtiene el uso del bus; 2) transfiere los datos a través del bus.
- Un módulo desea obtener datos de otro módulo: 1) Obtiene el uso del bus; 2) solicita la transferencia al otro módulo mediante las líneas de dirección y control apropiadas, y esperar hasta el envío de los datos desde el otro módulo.

Físicamente, el bus del sistema consiste en conductores eléctricos paralelos, dispuestos en la placa de c.i., y que se extiende a todos los componentes del sistema, cada uno de los cuales se conecta a algunas o a todas las líneas del bus. Una forma usual y económica de implementar este tipo de bus es conectar a él diferentes placas que contienen los módulos, lo que hace al sistema fácilmente expansible sin más que añadir nuevas placas, a la vez que facilita la determinación de fallos.

1.3. Funcionamiento de la Unidad Central de Proceso

La CPU lee ('fetch') instrucciones de la memoria (una cada vez), la ejecuta y busca la siguiente, ininterrumpidamente. El procesamiento de una instrucción se denomina 'ciclo de instrucción' como se muestra en la figura. El proceso se interrumpe sólo si la máquina es desconectada, ocurre algún tipo de error o se encuentra una instrucción HALT.

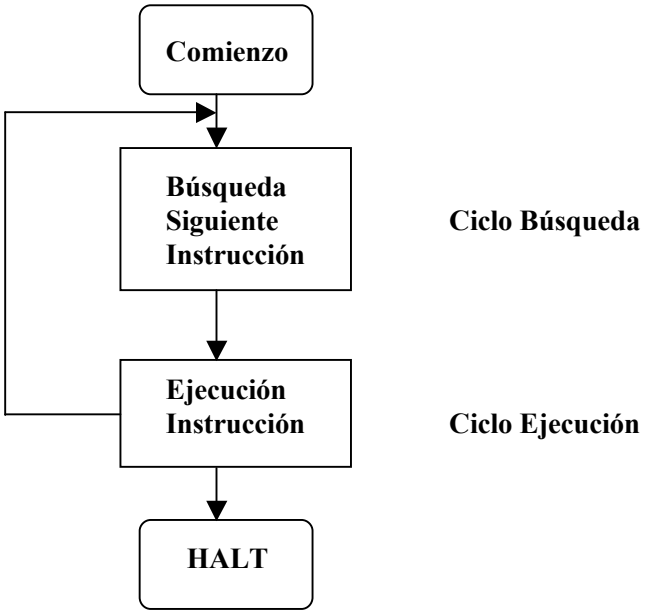


FIGURA 1.3. Ciclo básico de instrucción.

CICLO DE BUSQUEDA Y EJECUCION.

La búsqueda de instrucción se realiza secuencialmente, si no se especifica lo contrario. La instrucción buscada está codificada en binario y especifica la acción a realizar por la CPU. La CPU interpreta esta instrucción e implementa las siguientes funciones:

- CPU-Memoria: transferencia de datos en cualquiera de los sentidos.
- CPU-E/S: transferencia de datos con módulos de entrada/salida.
- Procesado de datos: implementación de operaciones aritméticas y/o lógicas.
- Control: control del flujo del programa, a través de saltos en la búsqueda de nuevas instrucciones motivados por resultados y órdenes anteriores.

Para algunas instrucciones, pueden necesitarse más de un acceso a memoria y de accesos a E/S. El caso más general de un ciclo de instrucción puede representarse como un diagrama de

estados, algunos de los cuales pueden ser nulos y algunos pueden ejecutarse más de una vez para una instrucción determinada.

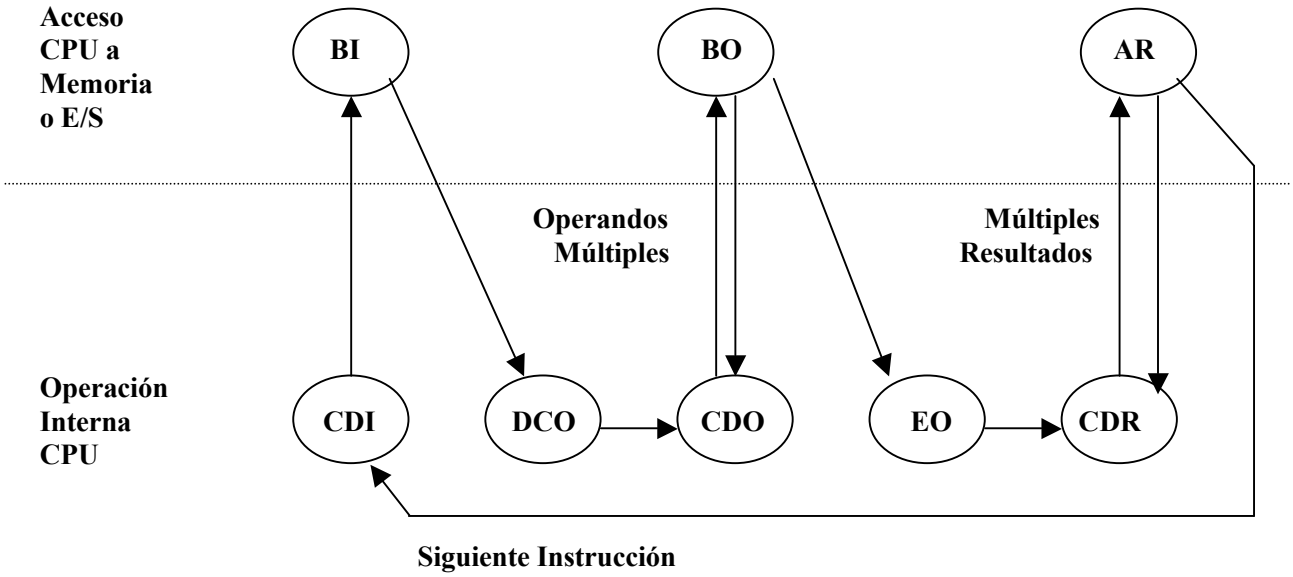


FIGURA 1.4. Diagrama de estados del ciclo de instrucción.

- Cálculo.dirección.instrucción (cdi): Determina la dirección de la siguiente instrucción a ejecutar. Generalmente supone sumar 1 a la dirección de la instrucción previa.

- Búsqueda.instrucción (bi): transfiere la instrucción desde memoria a la CPU.
- Decodificación.código.operación (dco): analiza la instrucción para determinar el tipo de operación a ejecutar y los operandos necesarios.

- Cálculo.dirección.operandos (cdo): si la operación hace referencia a un operando almacenado en memoria o accesible via E/S, se determina la dirección del operando.

- Búsqueda.operando (bo): transferencia del operando desde memoria o E/S.

- Ejecución.operación (eo): realiza la operación indicada.

- Cálculo.dirección.resultado (cdr): si la operación hace referencia a un resultado a almacenar en memoria o accesible vía E/S, se determina la dirección del resultado.

- Almacenamiento.resultado (ar): escribe el resultado en memoria o E/S.

Los estados de la parte superior del diagrama están relacionados con un intercambio entre la CPU y memoria o E/S, mientras que los de la inferior se refieren a operaciones internas de la CPU.

CICLO DE INTERRUPCION.

Las interrupciones, que implican un mecanismo por el cual se interrumpe el procesado normal de la CPU, responden a la necesidad de incrementar la eficiencia de procesado. En algunos casos, la CPU está conectada a dispositivos de E/S lentos. En estos casos conviene que la CPU no esté sirviendo continuamente al periférico, sino que éste avise de cuándo ha terminado la operación anterior y está listo para recibir otra. Este aviso se realiza a través de una señal de petición de interrupción ('interrupt request'), a lo que la CPU responde interrumpiendo el programa en curso, saltando a un subprograma de servicio al periférico, y reasumiendo la ejecución original posteriormente.

Para acomodar las interrupciones, es necesario añadir un ciclo de interrupción. En este ciclo, la CPU comprueba si se ha producido alguna interrupción. Si no se ha producido, se continúa con la siguiente instrucción. Si se ha producido, se ejecutan los siguientes pasos:

1. Guarda el contexto del programa actualmente en ejecución. Esto significa guardar la dirección de la siguiente instrucción a ejecutar y cualquier otro dato relevante para la CPU.

2. La siguiente instrucción a ejecutar se bifurca al programa de servicio de interrupción. Cuando se termina de ejecutar este programa, se recupera el contexto anterior.

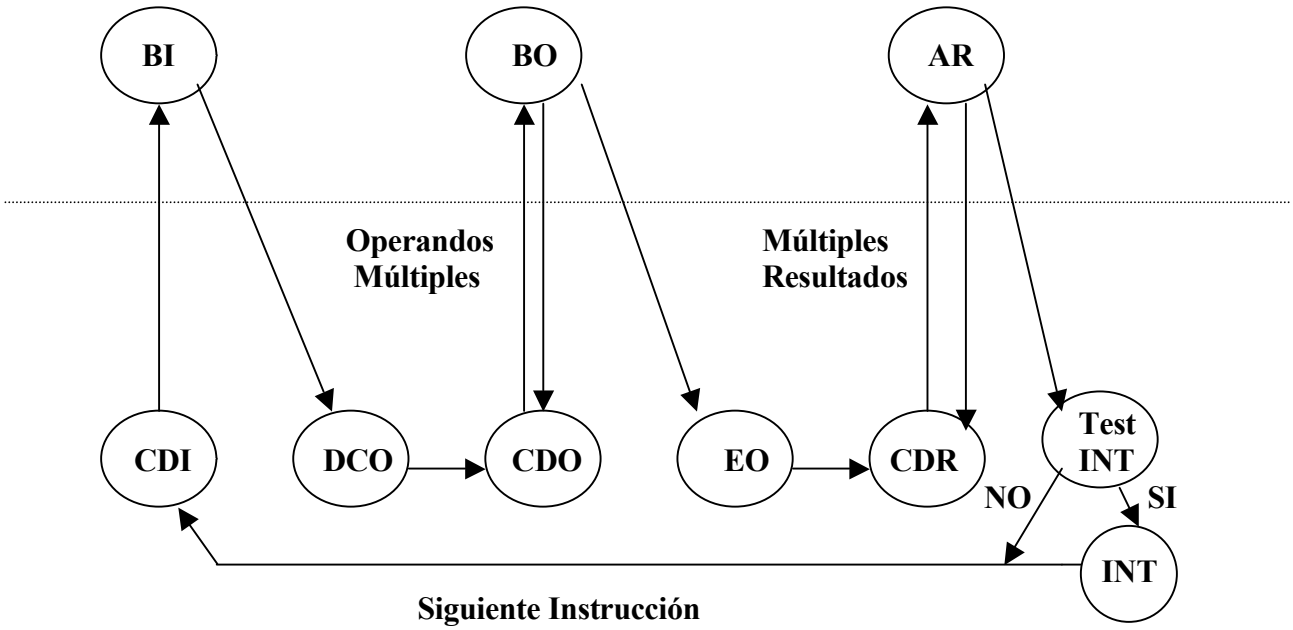


FIGURA 1.5. Ciclo de instrucción con interrupciones.

Las interrupciones no siempre se sirven inmediatamente. La CPU puede deshabilitar o habilitar todas o algunas líneas de interrupción. Generalmente, mientras se está sirviendo una, se

suelen deshabilitar para evitar interrupciones anidadas.

1.4. Estructura del Procesador

Para comprender la organización de la CPU, consideremos los requerimientos exigidos:

- Buscar instrucciones: lectura de memoria.
- Interpretar instrucciones: decodificación para determinar acciones a realizar.
- Buscar datos: lectura de datos de memoria y/o E/S.
- Procesar datos: implementar operaciones aritméticas o lógicas.
- Escribir resultados: sobre memoria y/o E/S.

Para realizar estas funciones, la CPU necesita almacenar temporalmente datos: debe memorizar la posición de la última instrucción para determinar la siguiente y contener los operandos mientras se ejecutan las instrucciones. Esta memoria interna a la CPU estará compuesta por los registros.

En la siguiente figura se muestra una estructura interna de la CPU. Además de los componentes básicos ya presentados (ALU y UC), comprende los caminos de transferencia de datos y de control, incluyendo el bus interno, necesario para transferir datos entre los diferentes registros y la ALU, puesto que ésta opera sólo con datos contenidos en registros, no con memoria externa. Existe, pues, una similaridad entre la estructura interna de la CPU y la estructura externa del computador: en ambos casos existen unas unidades funcionales conectadas entre ellas por caminos o buses.

En los siguientes apartados estudiaremos la estructura de estos bloques funcionales: Registros, ALU y Unidad de Control, así como el tipo de operaciones que ejecuta la CPU y el control de las mismas.

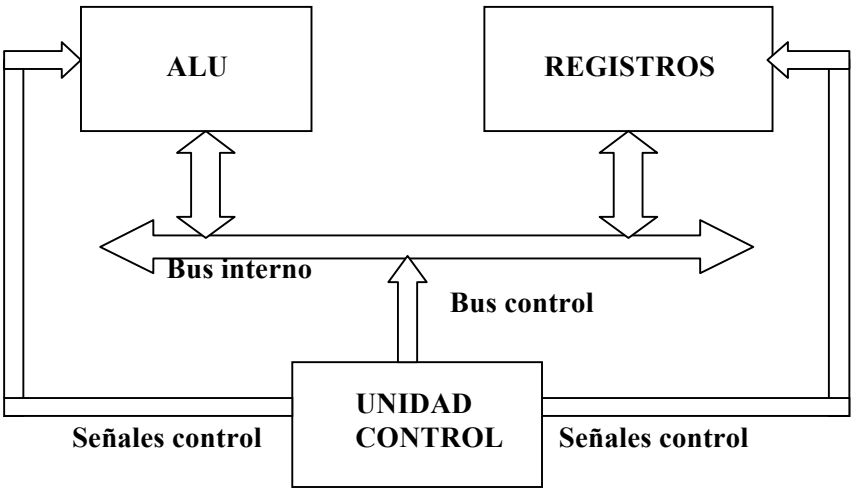


Figura 1.6. Estructura interna de la CPU.

1.5. Organización de Registros.

Los registros en la CPU sirven para dos funciones:

- Registros visibles para el usuario: permiten al programador minimizar las referencias a la memoria principal mediante la optimización de su uso.

- Registros de control y estado: son utilizados por la unidad de control para controlar la operación de la CPU y por los programas del sistema operativo para controlar la ejecución de programas.

REGISTROS VISIBLES PARA EL USUARIO.

Un registro visible para el usuario es aquél que puede referenciarse por medio del lenguaje máquina que la CPU ejecuta. Se pueden caracterizar en las siguientes categorías:

- Propósito general.
- Datos.
- Direcciones.
- Códigos condicionales.

Los registros de propósito general son utilizados por el programador para diversas funciones. En principio cualquier registro puede utilizarse para contener el operando de cualquier instrucción. No obstante, pueden existir restricciones (por ejemplo, pueden existir registros para operaciones en coma flotante).

En algunos casos, estos registros pueden utilizarse para funciones de direccionamiento (por ejemplo, desplazamiento, indirecto por registro). En otros casos, existe una distinción clara entre registros de datos y direcciones. Los primeros sólo pueden contener datos, mientras que los segundos pueden ser también de propósito general o estar dedicados a un modo particular de direccionamiento. Por ejemplo:

- Punteros de segmento: en máquinas con direccionamiento segmentado, un registro segmento contiene la dirección de la base del segmento. Puede haber múltiples registros, por ejemplo uno para el sistema operativo y otro para el proceso actual.
- Registros índice: Se utilizan para direccionamiento indexado y pueden ser autoindexados.
- Puntero de pila: si existe direccionamiento por pila visible al usuario, el pila está generalmente en memoria y hay un registro dedicado que apunta a la cima. Esto permite direccionamiento implícito.

Con el uso de registros especializados, pueden estar implícitos en el código de operación, ahorrando bits del especificador, que sólo deberá distinguir uno entre un conjunto determinado y no uno de entre todos los registros. No obstante, la excesiva especialización limita la flexibilidad

del programador.

El número de registros afecta también al conjunto de instrucciones ya que son necesarios más bits de la instrucción para seleccionar el registro. En máquinas convencionales, el óptimo suele estar entre 8 y 32.

La longitud del registro es otro factor importante. Los registros de direcciones deben ser al menos de igual longitud que la dirección a seleccionar. Los de datos pueden trabajar con longitudes diferentes, por lo que en ocasiones se disponen contiguos para usarlos con valores de doble longitud.

Existe una gran variedad de organización de registros, dependiendo de la máquina, pero hay cuatro registros que son básicos para la ejecución de una instrucción:

- Contador de programa (PC): contiene la dirección de la instrucción a buscar. Es incrementado generalmente por la CPU después de cada búsqueda, por lo que siempre apunta a la siguiente instrucción a ejecutar.
- Registro de instrucciones (IR): contiene la última instrucción buscada. La instrucción es analizada a nivel de código y operandos.
- Registro de dirección de memoria (MAR): contiene la dirección de una posición de memoria. Se utiliza para la búsqueda de instrucciones y/o operandos. Está conectado al bus de direcciones.
- Registro buffer de memoria (MBR): contiene un dato leído de memoria o a escribir en ella. Está conectado al bus de datos y los registros intercambian datos con memoria a través de él.

Internamente, los datos deben acceder a la ALU para realizar operaciones. La ALU puede tener acceso directo al MBR y a los registros, o bien puede disponer de registros buffer adicionales, que sirven como registros entrada/salida a la ALU e intercambian datos con MBR y registros.

Los registros de códigos de condición o estado contienen flags que son activados por la CPU como resultado de operaciones. El código puede ser testado posteriormente para operaciones de salto condicional. En general, pueden ser leídos, pero no modificados directamente por el programador. Todas las CPU incluyen uno o varios de este tipo de registros, conocidos como 'palabra de estado de programa (PSW)'. Los indicadores más usuales son:

- Signo: contiene bit de signo de la última operación realizada.
- Cero: '1' cuando el resultado es 0.
- Acarreo: '1' cuando se produce un sobrepasamiento del último bit como resultado de una operación. Utilizado en operaciones aritméticas de precisión extendida.
- Igual: '1' cuando una comparación lógica es igual.
- Sobrepasamiento: indica overflow aritmético.
- Habilitación interrupción: habilita o deshabilita interrupciones.
- Supervisor: Indica cuándo la CPU está trabajando en modo supervisor o usuario. Algunas instrucciones privilegiadas y algunas áreas de memoria sólo pueden utilizarse en modo supervisor.

1.6. Unidad Aritmético Lógica

La Unidad Aritmético-Lógica (ALU) es el elemento funcional de la CPU encargada de la ejecución de instrucciones. Dicha ejecución viene determinada por las órdenes de la Unidad de Control, y el resultado de la instrucción activa los correspondientes indicadores de estado (Signo, Cero, Acarreo, Sobrepasamiento...) que son utilizados por la Unidad de Control para determinar el funcionamiento de las siguientes instrucciones. La siguiente figura muestra la estructura general de una ALU.

El banco de registros proporciona los operandos al elemento de cálculo (operador), y almacena resultados intermedios y finales. Es frecuente la existencia de uno de estos registros, denominado ***acumulador*** , que suele trabajar como registro resultado de la operación y tiene un tratamiento especial respecto de las instrucciones que pueden ejecutarse con él (mayor cantidad y tipos que con los otros registros).

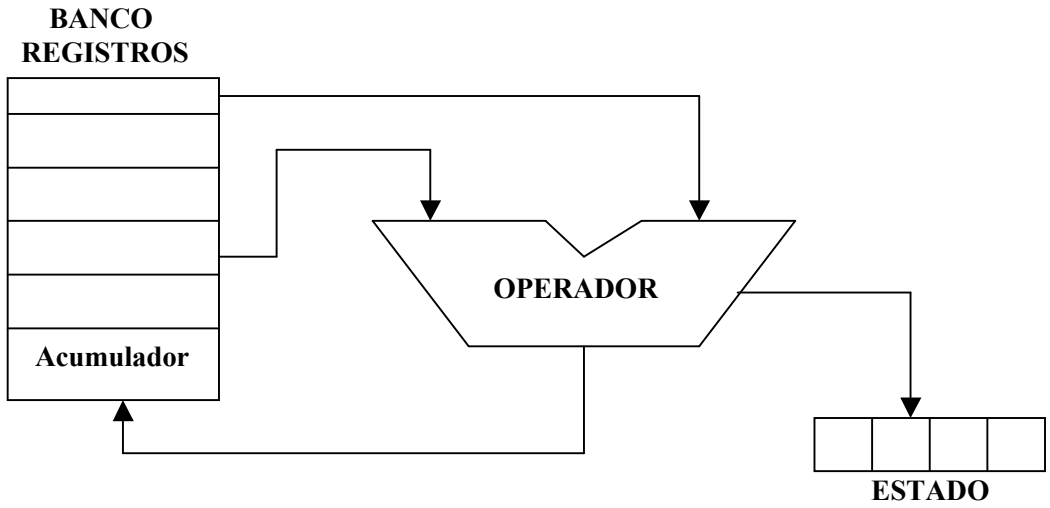


Figura 1. 7. Estructura genérica de la Unidad Aritmético-Lógica.

El operador suele estar implementado con circuitos combinacionales, y la Unidad de Control determina su operación en el caso de instrucciones complejas.

Los indicadores de estado ('*flags*') están implementados con biestables, aunque a nivel de acceso suelen organizarse como parte del registro de estado comentado anteriormente. Su información permite a la Unidad de Control determinar la opción de ejecución en instrucciones de bifurcación condicionales.

1.7. Conjunto de Instrucciones

La operación de la CPU está determinada por las instrucciones que ejecuta, que suelen denominarse instrucciones máquina. Puesto que la CPU implementa distintas funciones, éstas se reflejan en distintos tipos de instrucciones. El conjunto de las diferentes instrucciones que puede

ejecutar una CPU se denomina conjunto de instrucciones.

Elementos de una instrucción maquina

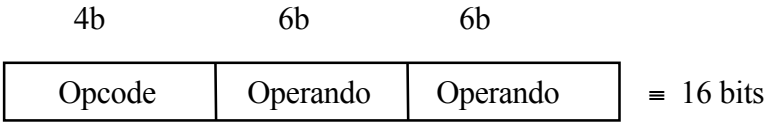
Cada instrucción debe contener la información que necesita la CPU para su ejecución. Si nos basamos en el ciclo de instrucción, podemos definir los elementos de una instrucción:

- Código de operación: especifica el tipo de operación a realizar (suele representarse de forma simbólica: mnemónico).
- Referencia a operando fuente: la operación puede implicar uno o más operandos fuente o de entrada.
- Referencia a operando resultado: operando donde se almacenará el resultado.
- Referencia a la siguiente instrucción: le informa a la CPU dónde buscar la siguiente instrucción a ejecutar. En el caso más general, la siguiente instrucción será la inmediatamente posterior en el programa, y no es necesaria una referencia explícita, pero sí en caso de saltos, llamadas a subrutinas, etc.

Los operandos pueden localizarse en:

- Memoria principal o virtual: debe proporcionarse la dirección.
- Registros CPU: si sólo existe uno, la referencia es implícita. Si hay más, cada registro tendrá un identificador.
- Dispositivo E/S: debe proporcionarse la dirección.

Cada instrucción se representa por una secuencia de bits y está dividida en campos, cada uno de los cuales contiene uno de los elementos comentados. Esta estructura se conoce como formato de instrucción. En la siguiente figura se muestra un ejemplo:



Tipos de operación

Aunque el conjunto de instrucciones varía de una CPU a otra, existen algunos tipos generales de instrucción que son comunes a todas las máquinas. Estos tipos se pueden clasificar en:

- Transferencia de datos. Deben especificar: 1) localización de los operandos fuente y destino; 2) longitud de los datos a transferir; 3) Modo de direccionamiento.
- Operaciones aritméticas. Los tipos usuales son: 1) Suma; 2) Resta; 3)

Multiplicación; 4) División; 5) Valor absoluto; 6) Negación; 7) Incremento; 8) Decremento.

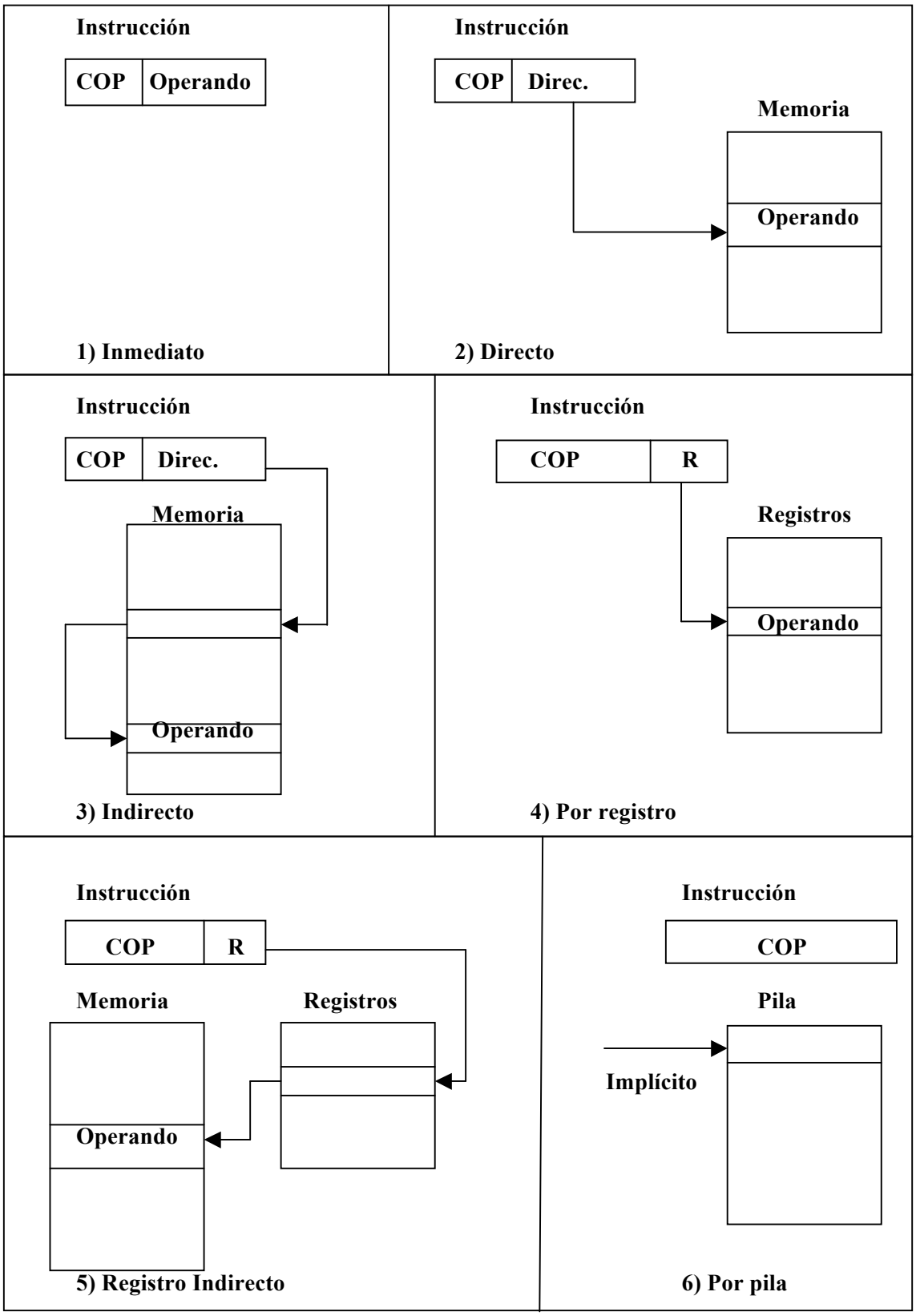
- Operaciones lógicas. NOT, OR, AND, XOR, desplazamientos lógicos a derecha e izquierda, desplazamientos aritméticos a derecha e izquierda, rotaciones a derecha e izquierda.
- Conversión. Modifican el formato de los datos (por ejemplo, conversión decimal a binaria o entero a coma flotante).
- E/S. Operaciones de lectura/escritura/control de módulos de E/S.
- Control del sistema. Generalmente son operaciones especiales que pueden ejecutarse sólo cuando el procesador se encuentra en un un determinado estado o ejecutando un programa privilegiado. Suelen estar reservadas al SO (por ejemplo, modificación de registros de control o acceso a bloques de control de proceso en sistemas multiprogramados).
- Transferencia de control. Los tipos usuales son: 1) Salto (condicional e incondicional); 2) Llamada a subrutina.

Direccionamiento

En la siguiente figura se muestran los modos de direccionamiento de operandos más comunes:

Para determinar qué modo de direccionamiento se está utilizando en una instrucción, existen diferentes soluciones. Puede utilizarse diferentes códigos de operación para cada modo. También pueden reservarse uno o más bits para determinar el modo de direccionamiento.

* Direccionamiento inmediato. No necesita referencias a memoria para obtener el operando, pero el valor de la constante está limitado por el número de bits del campo.
* Direccionamiento directo. Requiere sólo un acceso a memoria sin cálculos previos de dirección, pero el margen de memoria que puede direccionar está limitado por el número de bits del campo correspondiente al operando.
* Direccionamiento indirecto. Permite aumentar el margen de memoria direccionada respecto al caso anterior, pero requiere dos accesos a memoria, uno para conseguir la dirección y otro para obtener el valor del operando.
* Direccionamiento por registro. Similar al directo, pero en este caso, el campo de dirección contiene referencia a un registro interno en lugar de memoria. Puesto que el número de registros no suele ser grande, el campo de dirección puede ser pequeño, y no se requieren accesos a memoria.
* Direccionamiento indirecto por registro. Similar al indirecto, pero se evita el primer acceso a memoria para buscar la dirección, que se toma de un registro.


Instrucción
COP
Operando
1) Inmediato
Instrucción
COP
Direc.
Memoria
Operando
2) Directo
Instrucción
COP
Direc.
Memoria
Operando
3) Indirecto
Instrucción
COP
R
Registros
Operando
4) Por registro
Instrucción
COP
R
Memoria
Registros
Operando
5) Registro Indirecto
Instrucción
COP
Pila
Implícito
6) Por pila

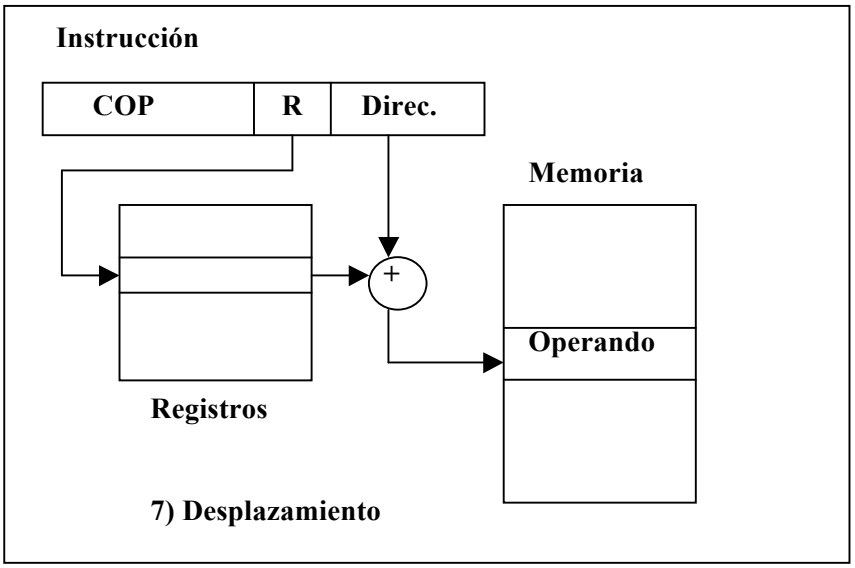


Figura 1. 9. Modos de direccionamiento.

* Direccionamiento por desplazamiento. Combina el direccionamiento directo y el indirecto por registro. Necesita dos campos de direcciones, con al menos uno explícito. El valor contenido en uno de ellos (A) se utiliza directamente. El otro referencia un registro cuyo contenido se suma a A para producir la dirección efectiva. Los tipos más comunes son:
    * Direccionamiento relativo. El contenido del registro PC se suma a A para producir el desplazamiento, que será relativo a la posición actual.
    * Direccionamiento basado en registro. Similar al anterior, pero el registro es otro diferente del PC.
    * Indexado. El campo de dirección referencia una dirección de memoria principal, y el registro referenciado contiene un desplazamiento respecto a esa dirección.
* Direccionamiento por pila. La pila es una estructura LIFO, que tiene reservada un bloque de direcciones de memoria. Asociado con la pila existe un registro puntero que contiene la dirección de la cima de la pila. Por tanto, referencias a las posiciones de memoria del pila se realizan mediante el puntero de pila y se comporta como direccionamiento indirecto por registro.

1.8. Operación de la Unidad de Control

Microoperaciones

La ejecución de un programa consiste en la ejecución secuencial de instrucciones. Cada instrucción se ejecuta durante un ciclo de instrucción, que está constituido por subciclos más cortos (búsqueda operación, búsqueda indirecta, ejecución, interrupción. Las características de cada subciclo implican una o más operaciones menores, denominadas microoperaciones.

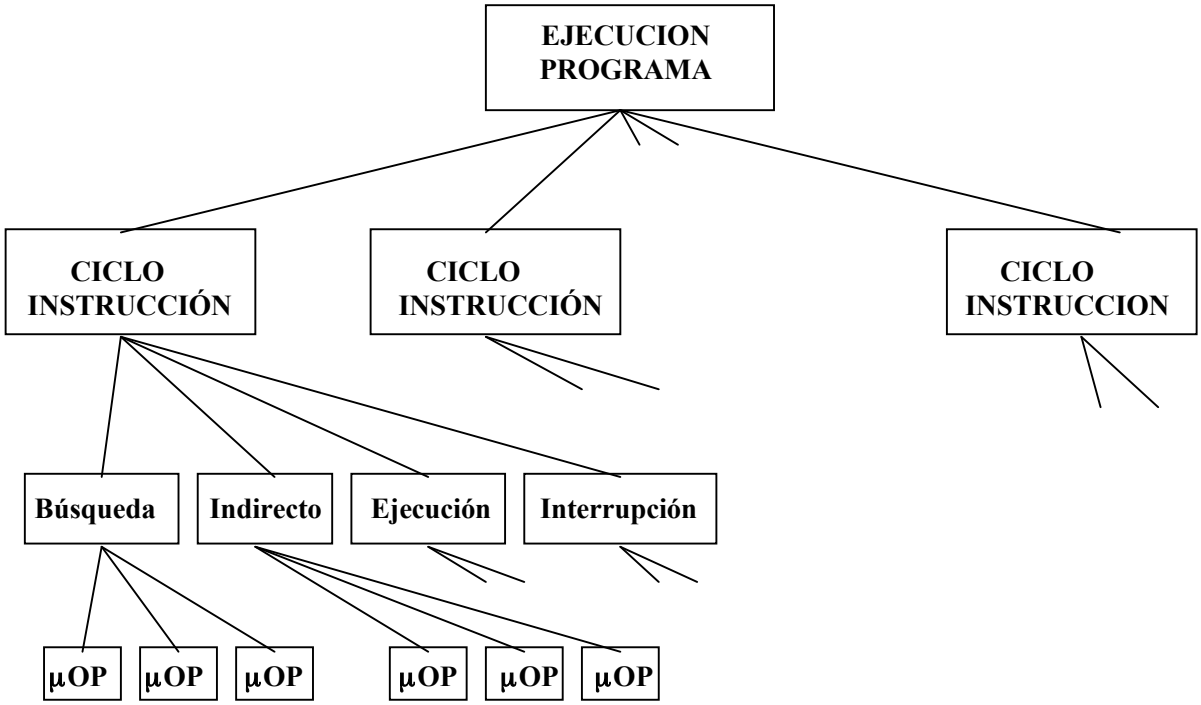


Figura 1.10. Elementos constituyentes de la ejecución de un programa.

CICLO DE BUSQUEDA.

El ciclo de búsqueda se produce al comienzo de cada ciclo de instrucción y produce que una instrucción se cargue en la CPU desde memoria. Consideraremos la estructura básica de 4 registros internos MAR (Memory Adress Register), MBR (Memory Buffer Register), PC (Program Counter), IR (Instruction Register).

MAR	
MBR	
PC	Direc. Instr.
IR	

1) Comienzo

MAR	Direc. Instr.
MBR	Instrucción
PC	Direc. Instr.+1
IR	

3) 2° estado

MAR	Direc. Instr.
MBR	
PC	Direc. Instr.
IR	

2) 1er estado

MAR	Direc. Instr.
MBR	Instrucción
PC	Direc. Instr.+1
IR	Instrucción

4) 3er estado

Figura 1.11. Secuencia de eventos en el ciclo de búsqueda.

Al comienzo del ciclo, la dirección de la siguiente instrucción a ejecutar está en PC. El primer paso es llevar esta dirección al MAR, puesto que es el único registro conectado a las líneas de direcciones del bus del sistema. El segundo paso es pasar este contenido al bus, ejecutar un comando READ mediante el bus de control y el resultado contenido en el bus de datos es copiado en el MBR. Necesitamos incrementar el PC. Puesto que estas dos operaciones (leer dato y sumar 1 al PC) no interfieren entre sí, pueden realizarse simultáneamente. El tercer paso es mover el contenido de MBR a IR, dejando libre MBR para un posible ciclo indirecto.

Por tanto, el ciclo de búsqueda consiste de este modo con tres pasos y 4 microoperaciones. Cada microoperación implica movimiento de datos fuera o dentro del registro. Mientras no interfieran entre sí, pueden realizarse a la vez. Simbólicamente puede escribirse el proceso como:

$$t_1: MAR \leftarrow (PC)$$

$$t_2: MBR \leftarrow Memory$$
$$PC \leftarrow (PC)+1$$

$$t_3: IR \leftarrow (MBR)$$

Estamos suponiendo que el reloj del sistema emite pulsos de igual duración que definen unidades de tiempo. Cada microoperación deberá realizarse en una unidad de tiempo. La notación t_i representa sucesivas unidades de tiempo.

Control de la CPU

La unidad de control realiza dos tareas básicas:

- Secuenciamiento: Fuerza a la CPU a pasar a través de una serie de microoperaciones en la secuencia apropiada, basada en el programa que se está ejecutando.
- Ejecución: Produce la ejecución de cada microoperación.

SEÑALES DE CONTROL.

Para realizar sus funciones, la unidad de control debe disponer de entradas que le permitan determinar el estado del sistema, y salidas que le permitan controlar su comportamiento.
Esto constituye las especificaciones externas de la unidad de control. Internamente debe disponer de la lógica requerida para realizar las funciones de secuenciación y ejecución.

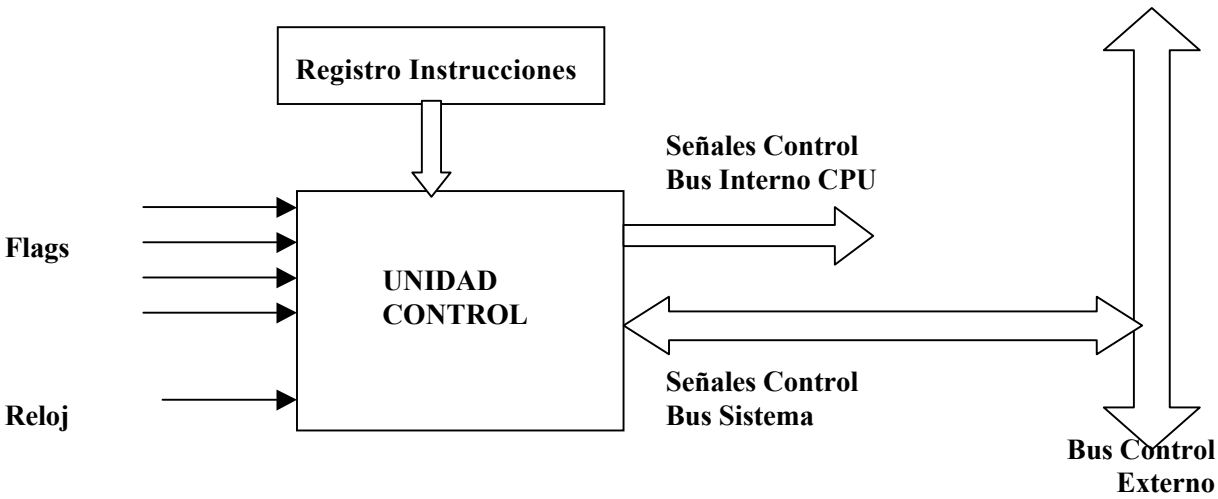


Figura 1.13. Modelo de unidad de control.

Las entradas son:

- Reloj. La unidad de control realiza una o varias microinstrucciones en una unidad de tiempo. Esto se conoce también como ciclo del procesador o ciclo de reloj.
- Registro de instrucciones. El código de operación se utiliza para determinar qué microinstrucciones realizar durante el ciclo de ejecución.
- Flags. Se utilizan para determinar el estado de la CPU e indicar el resultado de operaciones ALU previas.
- Señales de control del Bus de Control. Este bus proporciona señales a la unidad de control, como las interrupciones y control del bus.

Las salidas son:

- Señales de control internas a la CPU. Existen dos tipos: las que producen la transferencia de datos de un registro a otro y las que activan funciones ALU.
- Señales de control al Bus de Control. Hay dos tipos: señales de control de memoria y de E/S.

Todas estas señales de control se conectan en última instancia a puertas lógicas individuales.

Control microprogramado

Los métodos de implementación de la unidad de control pueden clasificarse en dos categorías:

- Implementación cableada.
- Implementación microprogramada.

En la primera, la unidad de control es básicamente un circuito combinacional, en el que las entradas lógicas son transformadas en señales lógicas de salida que constituyen las señales de control. No obstante, en una CPU actual, en el que la unidad de control puede ser muy compleja, el número de ecuaciones booleanas que definen el circuito combinacional puede ser muy grande. Además, resulta difícil testar el circuito , por las múltiples combinaciones a comprobar, así como posteriores modificaciones.

La implementación microprogramada se basa en la codificación de las microoperaciones en un lenguaje simbólico, lo que es conocido como lenguaje de microprogramación. Cada línea describe un conjunto de microoperaciones que ocurren simultáneamente, y se denomina microinstrucción. Una secuencia de microinstrucciones se denomina microprograma o 'firmware'. Para cada microoperación la unidad de control debe general señales de control determinadas, por lo que pueden codificarse en una palabra de control en la que hay un bit por cada línea de control.

No obstante, utilizar una palabra de control de este tipo podría dar problemas cuando la secuencia de microoperaciones no coincide entre instrucciones diferentes (por ejemplo, a veces se necesita un ciclo indirecto y a veces no). Podemos aproximarnos a la solución almacenando las palabras de control en una memoria en la que cada palabra tenga una única dirección. De esta manera, se puede añadir un campo de dirección a cada palabra de control que indique dónde se almacena la siguiente palabra de control a ejecutar si se cumplen determinadas condiciones y algunos bits para especificar la condición.

El resultado de esto se conoce como microinstrucción horizontal. Tiene un bit para cada línea de control interna de la CPU y un bit para cada línea del bus de control externo. Hay un campo de condición que indica la condición para la que se producirá el salto, y otro campo con la dirección de la siguiente instrucción a ejecutar en este caso. La microinstrucción se interpretará de la siguiente forma:

1. Para ejecutarla, poner a estado alto todas las líneas de control cuyo bit representativo esté a '1', y a bajo las que estén a '0'. Las señales resultantes pueden realizar una o más microoperaciones.

2. Si la condición indicada no se cumple, ejecutar la siguiente microinstrucción.

3. Si la condición se cumple, ejecutar la microinstrucción indicada por el campo de direcciones.

En microinstrucciones verticales, se utiliza un código para cada acción a realizar, y un decodificador convierte este código en señales individuales de control. La ventaja de este tipo es que es más compacto que la microinstrucción horizontal a expensas de un pequeño incremento de lógica y retraso temporal.

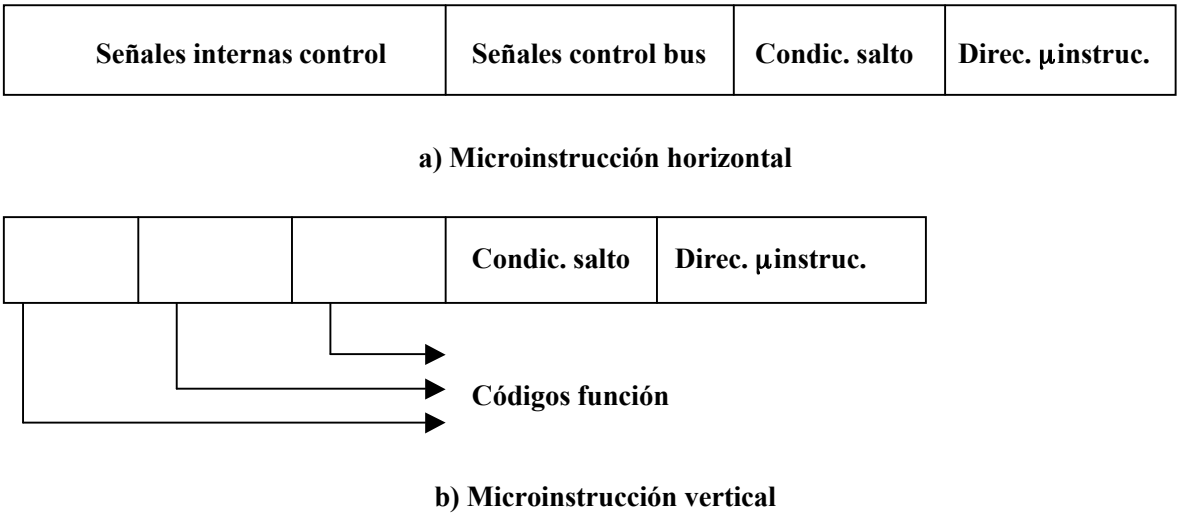


Figura 1.15. Formatos de microinstrucciones.

La siguiente figura muestra cómo pueden ordenarse estas microinstrucciones en una 'memoria de control'. Las microinstrucciones de cada rutina se ejecutan secuencialmente. Cada rutina termina con un salto que indica por dónde se debe continuar. El ciclo de ejecución representa las distintas rutinas que implementan las instrucciones máquina a ejecutar dependiendo del código de operación actual.

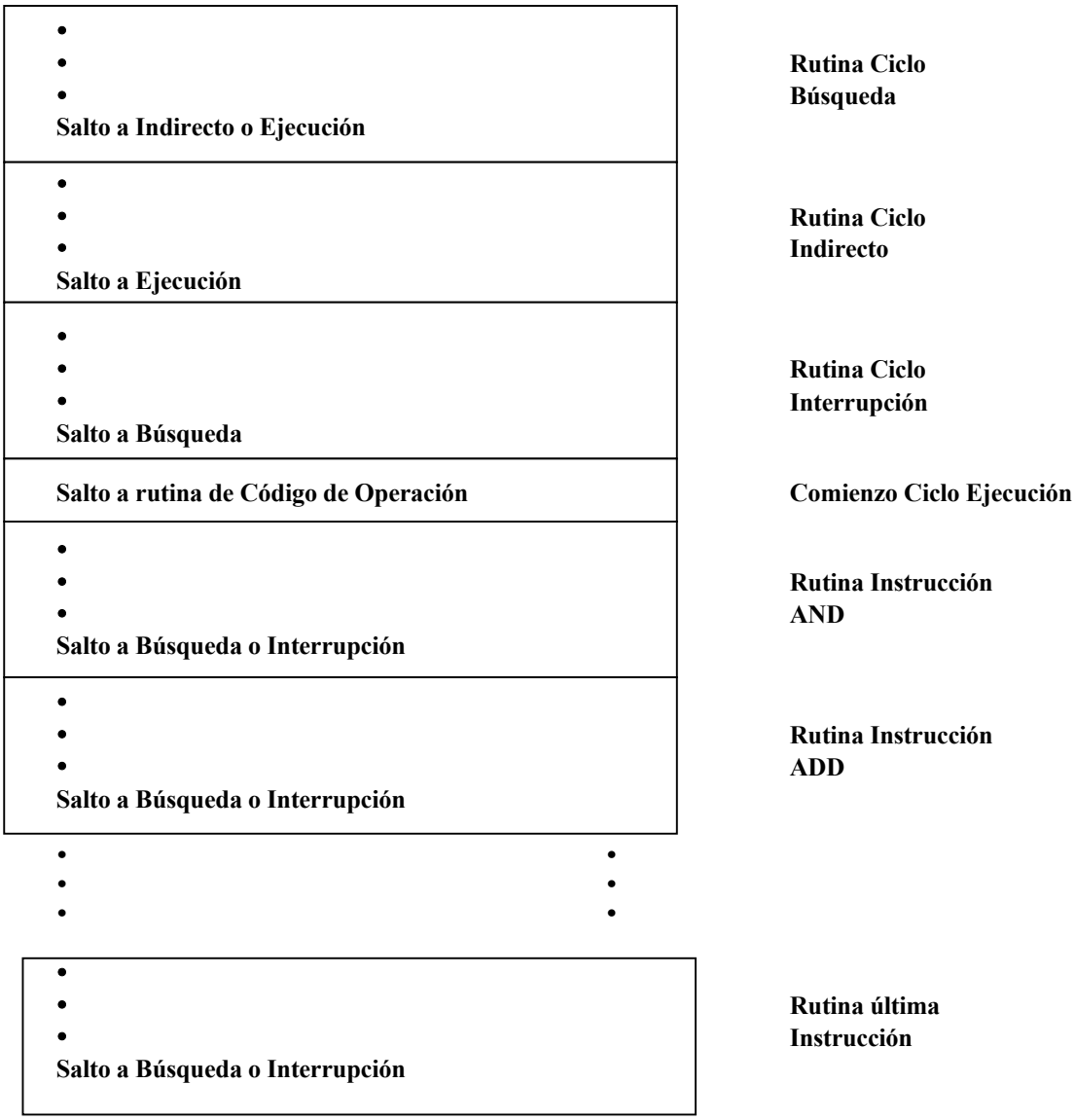


Figura 1.16. Organización de la memoria de control.

Esta organización da información sobre el funcionamiento de la unidad de control para un computador en particular, y proporciona también un método de implementarla.

Unidad de control microprogramada

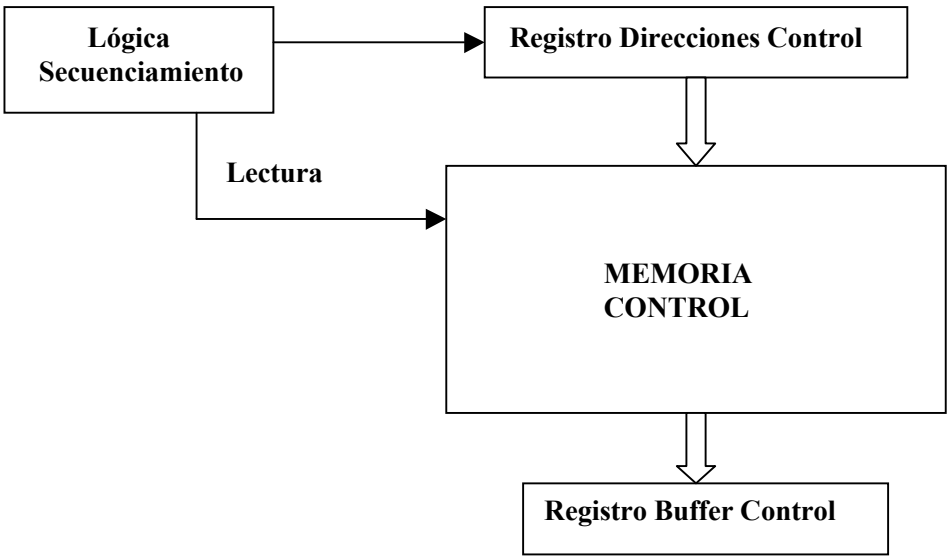


Figura 1.17. Microarquitectura de la unidad de control.

La figura muestra los elementos clave de la implementación microprogramada. El conjunto de microinstrucciones se almacena en la memoria de control. El registro de direcciones de control contiene la dirección de la siguiente microinstrucción a ejecutar. Cuando se lee una microinstrucción de la memoria, se transfiere al registro buffer de control. La parte izquierda de este registro (que coincide con la de la microinstrucción horizontal de la figura 1.15.) conecta con las líneas de control que salen de la unidad de control. Por tanto, leer una microinstrucción de la memoria de control es equivalente a ejecutarla. El tercer elemento es una unidad de secuenciamiento que carga el registro de direcciones de control y fuerza un comando de lectura.

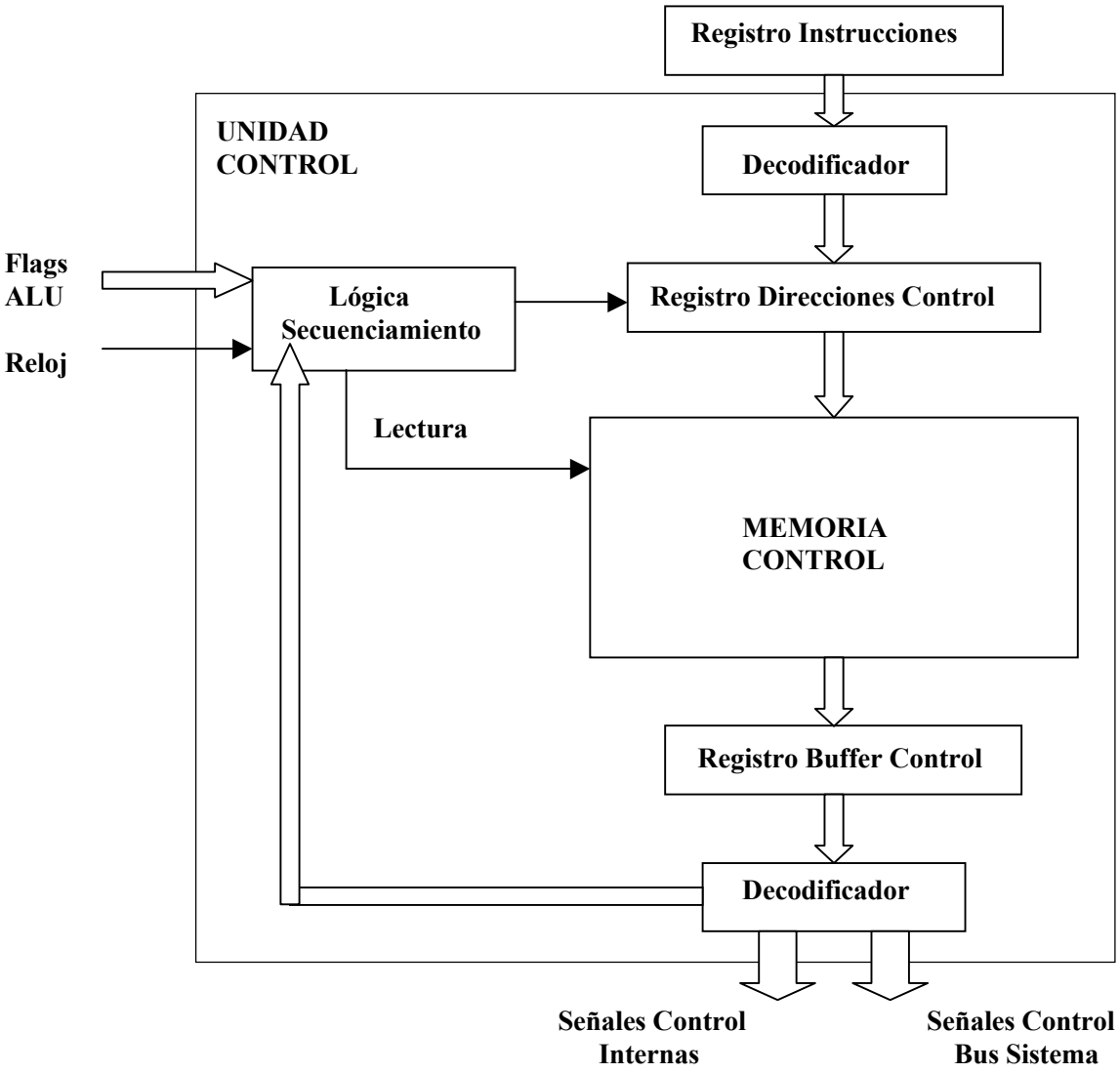


Figura 1.18. Funcionamiento de la unidad de control microprogramada.

Las funciones de la unidad de control son las siguientes:

1. Al ejecutar una instrucción, la unidad lógica de secuenciamiento realiza una operación de lectura de la memoria de control.

2. La palabra cuya dirección se especifica en el registro de direcciones de control se copia en el registro buffer de control.

3. El contenido del registro buffer de control genera señales de control e información sobre la siguiente dirección para la unidad lógica de secuenciamiento.

4. La unidad lógica de secuenciamiento carga una nueva dirección en el registro de direcciones de control basándose en la información sobre la siguiente dirección del buffer de control y los flags de la ALU.

Al finalizar cada microinstrucción, la lógica de secuenciamiento carga una nueva dirección en el registro de direcciones. Dependiendo de los flags de la ALU y del buffer de control, existen tres posibilidades de decisión:

- Tomar la siguiente instrucción: sumar 1 al registro de direcciones.

- Saltar a una nueva rutina basándose en una microinstrucción de salto: cargar el campo de dirección del buffer de control en el registro de direcciones.

- Saltar a una rutina de servicio de instrucción máquina: cargar el registro de control basándose en el código de operación de IR.

El decodificador superior de la figura traslada el código de operación de IR a la dirección de la memoria de control. El decodificador inferior no se usa en microinstrucciones horizontales, sino en verticales, utilizándose para la decodificación de las señales de control.

La principal ventaja de usar microprogramación para implementar unidades de control es la simplificación de diseño, lo que conlleva menor coste y menos posibilidades de error. Una unidad cableada contiene lógica compleja, mientras que la microprogramada sólo necesita decodificadores y lógica de secuenciación.

La principal desventaja de la microprogramación es su menor velocidad frente a la cableada para una tecnología dada. No obstante, la microprogramación es la técnica dominante en los computadores actuales.

1.10. Estructura segmentada

La segmentación de instrucciones aprovecha la circunstancia de que una instrucción debe pasar por diferentes etapas en su ejecución, por lo que varias instrucciones pueden ejecutarse a distintos niveles de forma simultánea. Este proceso se denomina 'pipelining' ya que, como ocurre en una 'pipeline', se aceptan nuevas entradas por un lado antes de que las anteriores produzcan resultados por el otro lado.

Como aproximación, supongamos la instrucción subdividida en dos etapas: búsqueda y ejecución. Habrá momentos durante la ejecución en que no se accederá a la memoria principal. Este tiempo puede utilizarse para buscar la siguiente instrucción a realizar.

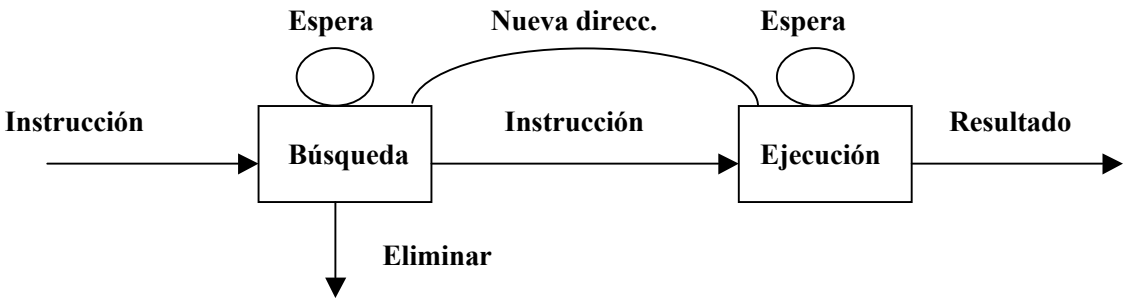


Figura 1.19. Pipeline de instrucciones de dos etapas.

En este caso, la pipeline tiene dos etapas independientes. La primera busca una instrucción y la almacena. Cuando la segunda etapa está libre, la primera pasa la instrucción. Mientras la segunda está ejecutándola, la primera aprovecha los ciclos en que no se utiliza la memoria para buscar y almacenar la siguiente. Esto se conoce como 'pre-búsqueda de instrucción' o 'búsqueda solapada'. En principio, si los tiempos de búsqueda y ejecución fueran iguales, este método proporcionaría el doble de velocidad de la máquina. En la práctica, no suele ser así por dos razones:

1. El tiempo de ejecución es generalmente mayor que el de búsqueda.

2. Instrucciones de salto condicional no permiten conocer la siguiente instrucción a ejecutar hasta la ejecución de la previa, por lo que en principio se deberá esperar en el proceso de búsqueda.

Puede reducirse este segundo tiempo buscando la siguiente operación después de un salto condicional. Si no se cumple el salto, no habrá pérdida de tiempo. Si se cumple, la instrucción siguiente se descarta.

Para incrementar la velocidad, la pipeline puede descomponerse en más etapas. Consideremos la siguiente descomposición de una instrucción:

1. Búsqueda de instrucción (FI): lee la siguiente instrucción a ejecutar y la copia en el buffer.

2. Decodificar instrucción (DI): determina el código de operación y los especificadores de operandos.

3. Cálculo de operandos (CO): calcula la dirección efectiva de cada operando fuente.

4. Búsqueda de operandos (FO): busca cada operando en memoria. Los operandos en registros no necesitan ser buscados.

5. Ejecución de la instrucción (EI): realiza la operación indicada y almacena el resultado, si es necesario, en la posición especificada.

Con esta descomposición, las diversas etapas tienen aproximadamente la misma duración. Suponiendo igual duración, y no permitiendo que más de una etapa que acceda a memoria esté activa a la vez, la pipeline de 5 etapas puede reducir el tiempo de ejecución para tres instrucciones de 15 unidades a 9. FI siempre implica acceso a memoria, pero FO y EI no siempre lo necesitan. No obstante, en este caso se ha supuesto que lo hacen.

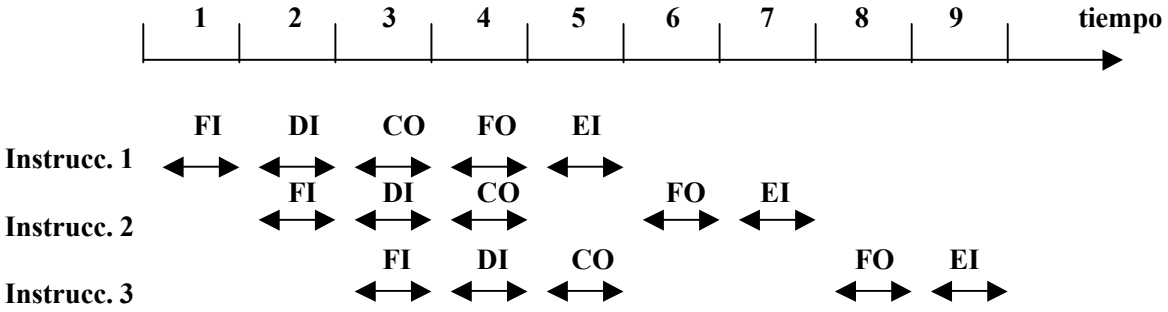


Figura 1.20. Diagrama de tiempos para la operación pipeline.

No obstante, algunos factores siguen limitando las ventajas de la pipeline. Si las cinco etapas no son iguales, habrá alguna espera implicada en las diversas etapas. También los saltos condicionales y las solicitudes de interrupción reducen la efectividad.

Otro problema se refiere a que la etapa CO puede depender del contenido de un registro que es modificado por la instrucción anterior todavía en la pipeline. Pueden darse también otros conflictos referidos a registros o memoria. El sistema debe disponer de lógica adecuada para resolver estos conflictos.

Además, para mejorar la eficiencia de la pipeline, se deberá disponer de lógica adicional. Si, por ejemplo, una etapa EI no necesita acceder a memoria, otra etapa de otra instrucción que sí lo necesite podrá ejecutarse en paralelo.

A la vista de esta discusión, puede parecer que un mayor nº de etapas en una pipeline implica una mayor velocidad de proceso. No obstante, existen dos factores en contra:

- Cada etapa de la pipeline tiene asociado un tiempo utilizado en movimientos entre buffers y otras funciones de preparación internas que puede ser significativo cuando el nº de etapas se incrementa, reduciéndose por tanto el tiempo de cada una de ellas.

- La cantidad de lógica necesria para controlas las dependencias de memoria y registros y ara optimizar el uso de la pipeline crece muy rápidamente con el nº de etapas, puediéndose dar el caso de que la lógica de control de las etapas es más compleja que las

etapas a controlar.

Uno de los principales problemas en el diseño de pipelines es asegurar un flujo estable de instrucciones. El principal impedimento es el salto condicional. Para solucionar esto, hay diversas aproximaciones:

1. Canales múltiples. Una pipeline simple es penalizada por una instrucción de salto puesto que debe escoger una de dos instrucciones a buscar, y puede escoger erróneamente. Si se dispone de varios canales, cada uno puede cargarse con una opción. Un problema de esta aproximación es la posibilidad de que una nueva instrucción de salto entre en la pipeline (en ambos canales) antes de que la lógica resuelva el salto original. Se necesitarían, pues, múltiples canales para evitar esto, con lo que el hardware se complica.

2. Prebúsqueda de la opción del salto. Cuando se reconoce un salto, se busca también la siguiente instrucción a él, a la vez que la posible. Esta última se mantiene hasta la ejecución de la operación salto. Si se da, se continúa con esa variación.

3. Predicción del salto. Se utilizan diversas técnicas para predecir si el salto se dará o no. Se basan en análisis del histórico de instrucciones anteriores, o en alguna medida dinámica de la frecuencia reciente de saltos.