ARQUITECTURA DE COMPUTADORAS

CAPITULO II

UNIDAD DE CONTROL

II.1 - INTRODUCCIÓN:

La Unidad de Control, según expuse anteriormente, es la encargada de buscar y hacer ejecutar las instrucciones. Es entonces una especie de director de operaciones, que manda a todas las restantes partes de la máquina, lo que deben hacer y en que instante de tiempo hacerlo. Recordemos que en la figura I.24, tenemos un esquema simplificado donde se destacan sus elementos integrantes.

En general la Unidad de Control es la menos conocida de las unidades que conforman un computador, por cuanto allí es donde aún persisten algunos secretos, que los diseñadores se guardan, pues las diferencias entre los diversos procesadores, que aún subsisten, se encuentran en ella.

La función de la misma, es la de buscar instrucciones y ejecutarlas. La primera parte de éste ciclo, que según dijera, es el ciclo de máquina, consiste en tomar la dirección de la próxima instrucción, y buscarla en memoria. La segunda, es interpretar (o decodificar) la parte mando de la instrucción, para proceder a emitir las señales de comando a todas las unidades que deben actuar en la ejecución, en el tiempo adecuado.

II.2 - INSTRUCCIONES:

La operación de la UCP es determinada por la instrucción que está ejecutando. Estas instrucciones son conocidas como instrucciones de máquina, existiendo una gran variedad de ellas. Cada máquina es capaz de ejecutar una cierta cantidad de instrucciones distintas, las que son conocidas como conjunto de instrucciones que es particular para cada máquina.

II.2.1 - ELEMENTOS DE UNA INSTRUCCIÓN DE MÁQUINA:

Cada instrucción debe contener toda la información requerida para que la UCP realice la ejecución. En la figura I.9, donde se indica el diagrama de estados de un ciclo de instrucciones, se tiene una buena forma de mostrar por implicancia los elementos de una instrucción. Ellos son:

- Código de Operación: Especifica la operación a ser llevada a cabo. Esta es indicada por un código binario, también conocido como "Código Operativo".

- Referencia al operando fuente: Esta operación puede comprender una o más fuentes de operandos, o sea datos que son las entradas de la operación.

- Referencia al operando resultado: La operación puede generar un resultado. Generalmente donde hay que almacenarlo.

- Referencia a la próxima instrucción: Lo que indica a la UCP donde buscar la próxima instrucción, luego de haber concluido la ejecución de la actual.

La nueva instrucción a buscar se encuentra alojada en memoria, o en un sistema de memoria virtual, en la principal o en la secundaria (disco). Donde no hay una referencia específica a la próxima instrucción, ella se encuentra en la casilla siguiente, en

cambio cuando se precisa una referencia específica, se debe además suministrar la dirección en la memoria principal o en la virtual.

Los operandos fuente y resultado, pueden encontrarse en alguna de las siguientes tres áreas:

- Memoria principal o virtual: Como referencia a la próxima instrucción debe entregarse la dirección en la cual se encuentra, tanto sea en la memoria principal como en la virtual.

- Registro de la UCP: Con raras excepciones, la UCP contiene uno o más registros que pueden ser referenciados por las instrucciones de máquina. Cuando existe uno solo, la referencia puede ser implícita. Si existen varios, a cada uno se le asigna un número, y la instrucción contendrá el número del referenciado.

- Dispositivo de E/S: La instrucción debe especificar cual módulo y cual dispositivo se utilizarán para la operación. Si se utiliza un sistema de E/S por memoria mapeada, es necesaria la dirección de memoria principal o virtual a la cual se conecta el dispositivo a emplear.

II.2.2 - REPRESENTACIÓN DE INSTRUCCIONES:

En el interior de la computadora, cada instrucción es representada por una secuencia de bits, divididos en campos específicos, que corresponden a cada elemento de la instrucción. La forma de la instrucción es conocida como formato de la misma, que podría ser tal como el indicado en la figura V.1, en la cual se destacan tres campos: el del código operativo, aquí de cuatro bits, el de la referencia al primer operando aquí de 6 bits y el correspondiente a la referencia al segundo operando, que también sería de seis bits.

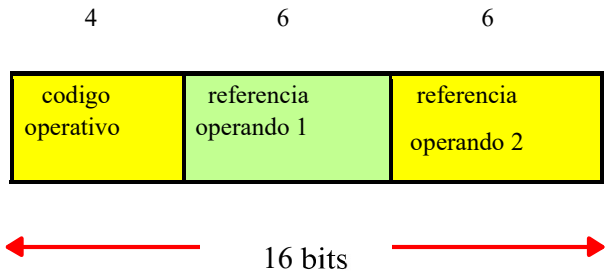


Figura II.1 - Un formato de instrucción simple de dos direcciones.

Dada la dificultad, tanto para el escritor como para el lector, para interpretar las expresiones en código binario de las instrucciones de máquina, se ha convertido en práctica corriente la de utilizar representaciones simbólicas, en las cuales los códigos operativos son reemplazados por abreviaturas nemotécnicas, tales como:

ADD = suma
SUB = sustracción
MPY = multiplicación
DIV = división
LOAD = carga de datos desde la memoria

STOR = almacenamiento de datos en memoria

Por otra parte, también los operandos son representados simbólicamente, así por ejemplo la instrucción:

ADD R,Y

puede significar: Sumar el contenido de la locación Y al contenido del registro R.

En consecuencia es posible escribir un programa en lenguaje máquina de una forma simbólica, donde cada código simbólico es reemplazable por una secuencia de bits. Asimismo el programador especifica la locación correspondiente a cada uno de los símbolos, para lo cual al inicio del programa les asigna una posición de memoria, por ejemplo, en la forma siguiente:

X = 513
Y = 514

y así sucesivamente.

Es posible escribir todo el programa en ésta forma simbólica, y proveer los medios para que un simple programa, traduzca los símbolos a códigos binarios. Estos lenguajes simbólicos, así como sus traductores son conocidos como ensambladores.

II.2.3 - TIPOS DE INSTRUCCIONES:

Una computadora tiene un conjunto de instrucciones tal, que le permite al usuario formular cualquier tarea de procesamiento de datos. Otra forma de expresar lo mismo, es tener en cuenta que una máquina debe operar con lenguajes de alto nivel, que son traducidos a lenguaje de máquina para ser ejecutados, de aquí que el conjunto de instrucciones de máquina, debe ser lo suficientemente amplio como para realizar todo lo solicitado por las instrucciones de alto nivel.

En general pueden darse cuatro categorías o tipos de instrucciones:

- De procesamiento de datos: instrucciones aritméticas y lógicas.

- De almacenamiento de datos: instrucciones con referencia a memoria.

- De movimiento de datos: instrucciones de E/S.

- De Control: Instrucciones de prueba y de bifurcación.

Las instrucciones aritméticas proveen la capacidad de procesar datos numéricos. Las lógicas operan sobre los bits de una palabra, tratándolos individualmente, por tanto permiten la realización de cualquier tipo de operación lógica que el programador desee ejecutar sobre ellos.

Las instrucciones con referencia a memoria, son las encargadas del movimiento de datos entre la memoria y los registros. Las de E/S son necesarias para transferir programas y datos hacia la memoria, así como para que el programador tenga acceso a los resultados.

Las instrucciones de prueba son utilizadas para comprobar el contenido de

una palabra o para determinar el estado de una ejecución. Finalmente, las instrucciones de bifurcación o de ramificación permiten transferir el control del programa a otro punto del mismo, en función de una decisión tomada en base a un cierto cálculo.

II.2.4 - CANTIDAD DE DIRECCIONES:

Si analizamos la secuencia de operaciones a realizar para la ejecución de una instrucción aritmética, vemos que es necesario tener:

1) la ubicación del primer operando
2) la ubicación del segundo operando 3) la ubicación donde guardar el resultado 4) la ubicación de la próxima instrucción.

Lo que nos lleva a una instrucción con cuatro direcciones, la cual es la menos encontrada en la práctica, por cuanto precisa procesadores de alta complejidad, con un complicado ciclo de máquina. Son más utilizados los sistemas de una, dos o tres direcciones. Donde la dirección de la próxima instrucción es implícita (obtenida del contador de programa). La operación es:

$$Y = (A-B)/(C+D*E)$$

INSTRUCCIÓN	SIGNIFICADO
SUB Y,A,B	Y <-- A - B
MPY T,D,E	T <-- D x E
ADD T,T,C	T <-- T + C
DIV Y,Y,T	Y <-- Y ÷ T

a) Con instrucciones de tres direcciones.

MOVE Y,A	Y <-- A
SUB Y,B	Y <-- Y - B
MOVE T,D	T <-- D
MPY T,E	T <-- T x E
ADD T,C	T <-- T + C
DIV Y,T	Y <-- Y ÷ T

b) Con instrucciones de dos direcciones.

LOAD D	AC <-- D
MPY E	AC <-- AC x E
ADD C	AC <-- AC + C
STOR Y	Y <-- AC
LOAD A	AC <-- A
SUB B	AC <-- AC - B
DIV Y	AC <-- AC ÷ Y
STOR Y	Y <-- AC

c) Con instrucciones de una dirección.

Figura II.2 - Programas para la ejecución de la operación

En la figura V.2 se compara la ejecución de una misma operación, mediante procesadores que operan con una, dos y tres direcciones.

Como vemos, al tener mayor cantidad de direcciones, se necesitan menos líneas de programa, pero las instrucciones se vuelven de mayor longitud, precisando más hardware. En realidad parece ser ideal la máquina de dos direcciones, en la cual se permite tener varios registros direccionables, por lo cual pueden hacerse operaciones directamente entre registros, que son más rápidas por no tener que efectuar un acceso a memoria.

La forma más simple es la una dirección, introduciendo una segunda, tal como la de un registro, en forma implícita. Esto fue muy común en las primeras máquinas, usando como registro implícito al acumulador (AC), todas las instrucciones hacían referencia al mismo.

De hecho, es posible tener máquinas con instrucciones sin ninguna dirección (instrucciones de cero direcciones), las que son aplicables a la conformación de memoria especial, denominada memoria de pilas (Stack memory), en la cual la referencia es siempre a la primera casilla de la pila, que es de tipo LIFO (Último que entra primero que sale).

Por razones de practicidad, la mayoría de los procesadores actuales utilizan una mezcla de instrucciones de dos y tres direcciones.

II.3 - TIPOS DE OPERANDOS:

Las instrucciones de máquina, operan sobre datos, los que se pueden catalogar en cuatro categorías principales:

- Direcciones - Números - Caracteres - Datos Lógicos

Según veremos más adelante, las direcciones pueden considerarse como una forma de datos, sobre los cuales pueden realizarse algunas operaciones para determinar la dirección efectiva en la memoria principal o en la virtual, así es que como datos, son números enteros sin signo.

II.3.1 - NÚMEROS.

Todos los lenguajes de máquina incorporan datos de tipo numérico, aún en los sistemas de procesamiento no numérico se necesitan números en diversas aplicaciones, tales como en contadores, ancho de los diversos campos, posiciones de memoria, etc.

Una distinción entre los números utilizados en los cálculos comunes y los aplicados en la computación, es su longitud. En una máquina, la cantidad de cifras siempre es acotada, tanto en su representación en coma fija, como en coma flotante.

Hay tres tipos de datos numéricos representables en computadoras:

- Coma fija o enteros

- Coma flotante o notación científica - Decimales empaquetados

Si bien la representación ideal en la máquina es la binaria, muchas veces es necesario utilizar un sistema decimal tanto para operaciones de E/S, como también, a veces, es conveniente almacenarlos en esa forma. Lógicamente se utiliza un sistema codificado, el BCD, cuya representación es denominada "empaquetada", por cuanto cada dígito decimal precisa cuatro dígitos binarios. Además en caso de utilizar números con signo, el mismo es representado mediante otro paquete de cuatro bits (el signo menos es representado por la serie 1111). Esto hace que el sistema sea muy ineficiente, pero evita el tener que traducir continuamente las cantidades.

Por otra parte, y según vimos en el capítulo I, hay máquinas que proveen instrucciones aritméticas para operaciones codificadas.

II.3.2 - CARACTERES:

Una forma común para los datos es la lista de caracteres, los cuales son representados por códigos de 7 ú 8 bits. El mas común es el ASCII, que con 7 bits permite la representación de 128 caracteres distintos. Lógicamente los numéricos y alfabéticos no son tantos, así que es posible agregar una cantidad de caracteres de control, algunos de los cuales permiten justamente controlar las impresoras. El código ASCII de ocho bits, permite la incorporación del bit de paridad.

II.3.3 - DATOS LÓGICOS:

Normalmente cada palabra es tratada como un dato unitario, sin embargo, a veces es conveniente considerar a una unidad de n bits como n datos de un bit, entonces se los considera como datos lógicos, entre los cuales es posible realizar operaciones lógicas, siendo la más común el desplazamiento de un bit, ya sea a derecha o a izquierda.

También es posible que un mismo dato sea tomado a veces como numérico y a veces como lógico, por ejemplo en los desplazamientos que se necesitan en las operaciones en coma flotante.

II.3.4 - TIPOS DE DATOS PENTIUM.

El Pentium puede tratar con datos de 8, 16, 32 y 64 bits, denominando a los conjuntos respectivamente: byte, palabra, doble palabra y cuádruple palabra. No correspondiendo las longitudes de palabra, doble palabra o cuádruple palabra con ninguna forma específica de direccionamiento, sino que pueden comenzar en cualquier celda de un byte, que es la unidad mínima de información admitida, también conocida como media palabra.

Estos son referenciados como tipos generales de datos, además el Pentium reconoce una imponente matriz de tipos de datos específicos que son reconocidos y operados por instrucciones particulares.

II.3.5 - TIPOS DE DATOS EN EL POWER PC.

El Power PC puede manejar datos de 8, 16, 32 o 64 bits de longitud, denominados: byte, media palabra, palabra, y doble palabra. Algunas instrucciones requieren que los operandos en memoria sean alineados entre límites de 32 bits, pero en general no se requiere ninguna alineación. Una propiedad interesante del procesador es que puede utilizar dos estilos para el almacenamiento, uno en el cual el byte menos significativo es almacenado en la dirección menor, y otro en el cual es almacenado en la dirección mayor. La primera es conocida como terminación pequeña, y el otro como terminación mayor.

El procesador de coma fija, reconoce los siguientes tipos de datos:

- Byte sin signo: Puede ser empleado en operaciones lógicas o aritméticas enteras. Son cargados en los registros rellenando con ceros a la izquierda hasta completar su capacidad.

- Media palabra sin signo: Lo mismo que el anterior pero para cantidades de hasta 16 bits.

- Media palabra con signo: Utilizada para operaciones aritméticas, cargada en la memoria mediante extensión del signo a la izquierda del registro. (El signo es repetido en todas las posiciones vacías del registro)

- Palabra sin signo: Utilizada para operaciones lógicas y puntero de direcciones.

- Palabra con signo: Utilizada para operaciones aritméticas.

- Doble palabra sin signo: Utilizada como puntero de direcciones.

- Lista de bits: de 0 a 128 bits de longitud.

Además el Power PC soporta los tipos de datos en coma flotante de simple y doble precisión, dados en la norma IEEE 754.

II.4 - TIPOS DE OPERACIONES:

La cantidad de códigos operativos varía ampliamente de máquina en máquina, pero en todas ellas se realizan los mismos tipos de operaciones. Una clasificación estándar de tipos de operaciones es la siguiente:

- Transferencia de datos - Aritméticas - Lógicas - de Conversión - de E/S - de Control del sistema - de Transferencia del control

En la tabla siguiente se tiene un listado de los tipos de instrucciones más comunes utilizadas dentro de cada categoría.

TIPO	NOMBRE DE LA OPERACIÓN	DESCRIPCIÓN
TRANSFERENCIA DE DATOS	MOVE (Transfer)	Transfiere una palabra o un bloque desde la fuente al destino.
	STORE	Transfiere una palabra desde la UCP a la memoria
	LOAD (Fetch)	Transfiere una palabra de la memoria a la UCP
	EXCHANGE	Intercambia los contenidos de la fuente y el destino.
	CLEAR (Reset)	Transfiere al destino una palabra conteniendo todos ceros
	SET	Transfiere al destino una palabra conteniendo todos unos.
	PUSH	Transfiere una palabra de la fuente al acceso a la pila.
	POP	Transfiere una palabra del acceso a la pila al destino.
ARITMÉTICAS	ADD	Computa la suma de dos operandos
	SUBSTRACT	Computa la resta de dos operandos
	MULTIPLY	Computa el producto de dos operandos
	DIVIDE	Computa el cociente de dos operandos
	ABSOLUTE	Reemplaza al operando por su valor absoluto
	NEGATE	Cambia el signo del operando
	INCREMENT	Suma 1 al operando
	DECREMENT	Resta 1 al operando
LÓGICAS	AND	Conforma la operación indicada entre bits.
	OR	Idem.
	NOT (Complement)	Idem.
	EXCLUSIVE-OR	Idem.
	TEST	Prueba la condición especificada.
	COMPARE	Hace la comparación lógica o aritmética de dos o más operandos.
	SET CONTROL VARIABLES	Conjunto de instrucciones que ubica controles para protección, manejo de interrupciones, timers, etc.
	SHIFT	Desplaza a derecha o izquierda al operando, introduciendo constantes en los extremos.
	ROTATE	Desplaza a derecha o izquierda al operando, en forma circular.

TRANSFERENCIA DEL CONTROL	JUMP (Branch)	Transferencia incondicional, cargando al contador de programa con un nuevo contenido.
	JUMP CONDITIONAL	Prueba la condición indicada y carga o nó al contador de programa con un nuevo contenido.
	JUMP TO SUBROUTINE	Guarda la información de control del programa en ejecución y salta a la nueva dirección especificada.
	RETURN	Reemplaza el contenido de los registros con datos conocidos.
	EXECUTE	Busca los operandos indicados y ejecuta la instrucción sin alterar al contador de programa.
	SKIP	Incrementa al contador de programa para pasar ala próxima instrucción.
	SKIP CONDITIONAL	Prueba la condición especificada y cambia o nó el contenido del contador de programa.
	HALT	Detiene la ejecución del programa.
	WAIT (Hold)	Suspende la ejecución del programa hasta que se cumpla una condición dada.
	NO OPERATION	No se realiza ninguna operación, pero el programa se sigue ejecutando.
ENTRADA /SALIDA	INPUT (Read)	Transfiere datos del dispositivo indicado o puerto de E/S al destino.
	OUTPUT (Write)	Transfiere datos de la fuente especificada al puerto de E/S o dispositivo indicado.
	START I/O	Transfiere instrucciones al procesador de E/S para iniciar la operación de E/S
	TEST I/O	Transfiere información de estado del sistema de E/S al destino indicado.
CONVERSIÓN	TRANSLATE	Traduce los valores de una sección de la memoria, en base a una tabla de correspondencias.
	CONVERT	Convierte el contenido de una palabra de una forma a otra.

II.4.1 - TRANSFERENCIA DE DATOS.

La operación fundamental de toda máquina, es la transferencia de datos. Para ser llevada a cabo, debe especificarse lo siguiente en la instrucción:

- Localización de los operandos fuente y destino: cada locación puede estar en la memoria, en un registro, o en el acceso a una pila.

- Longitud de los datos a transferir.

- Modo de direccionamiento para cada operando.

Cuando los datos están en registros de la UCP, la tarea es la más simple de todas las que se realizan, en cambio cuando por lo menos uno de los datos está en la memoria, las operaciones a ejecutar son:

1 - Calcular la dirección de memoria.
2 - Si la dirección se refiere a memoria virtual, se debe trasladar el dato a la zona de memoria real.
3 - Determinar si el ítem referenciado está en la cache.
4 - Si no está, buscarlo en el módulo de memoria correspondiente.

II.4.2 - ARITMÉTICAS:

Muchas máquinas proveen las operaciones aritméticas básicas de suma, resta, multiplicación y división invariablemente para números enteros con signo. A menudo, también se las provee para números en coma flotante o para decimales empaquetados.

Otras operaciones posibles, incluyen una variedad de instrucciones de operando único, por ejemplo:

- Absolute: toma el valor absoluto del operando.
- Negate: cambia el signo del operando.
- Increment: suma una unidad al operando.
- Decrement: resta una unidad al operando.

La ejecución de una operación aritmética puede incluir operaciones de transferencia de operandos a la unidad aritmética, y entregar la salida de la misma.

II.4.3 - LÓGICAS:

Muchas máquinas proveen una variedad de operaciones para la manipulación de los bits de una palabra, o de otras unidades direccionables, todas basadas en operaciones Booleanas.

II.4.4 - DE CONVERSIÓN:

Las operaciones de conversión son las que permiten modificar el formato de los datos. Por ejemplo, la conversión de binario a BCD.

II.4.5 - DE ENTRADA/SALIDA:

Estas ya han sido indicadas en el capítulo anterior, y como sabemos, incluyen una cierta variedad de aproximaciones, que incluyen la E/S programada, la E/S por DMA, hasta el uso de un procesador de E/S.

II.4.6 - CONTROL DEL SISTEMA:

Las instrucciones de control del sistema son generalmente privilegiadas, por cuanto solo pueden ejecutarse en algún estado especial de la ejecución de un programa, estas instrucciones son reservadas para su utilización en los sistemas operativos.

Como ejemplo, algunas de estas instrucciones pueden alterar el contenido de algunos registros de control, así como pueden modificar los códigos de protección del almacenamiento.

II.4.7 - DE TRANSFERENCIA DEL CONTROL:

En todos los tipos de instrucciones vistos, se especifican los operando, la operación a realizar y que debe hacerse con el resultado, así como la dirección de la próxima instrucción, la que normalmente es la que sigue a la de la que se encuentra en ejecución, y es indicada por el contador de programa.

Pero, una buena parte de las instrucciones de un programa son las encargadas de modificar la secuencia del mismo, o sea hacer que se busque la próxima instrucción en otro lugar de la memoria. En estas instrucciones la acción de la unidad de control, es la de actualizar el contenido del contador de programa, a fin de que apunte a la dirección necesaria.

Existe una cierta cantidad de razones por las cuales es necesario modificar la secuencia de un programa, entre las más importantes, podemos citar:

1 - En los cálculos prácticos, es necesario ejecutar una instrucción más de una vez, generalmente varios miles de veces, lo que requeriría la implementación de un programa con varios millones de instrucciones, para evitarlo, es posible escribir en forma separada el conjunto de instrucciones que será repetido, y acceder a él toda vez que sea necesario.

2 - Virtualmente todo programa implica el tomar algún tipo de decisión. En estos casos la ejecución del programa puede realizarse por diversos caminos según cual sea esa decisión. (por ejemplo la resolución de una raíz cuadrada, donde primero se prueba el signo de la misma, y si es negativo no se calcula)

3 - Para confeccionar correctamente un programa relativamente largo, es conveniente poder separarlo en tareas y programar y correr cada una de ellas por separado.

Las operaciones de transferencia de control más comunes son:

- Bifurcación
- Salto
- Llamado a Subrutina

II.4.7.1 - INSTRUCCIONES DE BIFURCACIÓN:

Una instrucción de bifurcación (branch), también llamada instrucción de salto, tiene como uno de sus operandos, la dirección de la próxima instrucción a ser ejecutada.

El caso más general, es el salto condicionado o condicional, que se realiza cuando se cumple una cierta condición, en caso contrario, o sea que la condición no se cumple, se sigue con el proceso normal del programa.

El salto involucra una actualización del contenido de algunos registros, en especial el contador de programa. Para la generación de la condición, se utilizan dos métodos: primero, en algunas máquinas se provee una condición de uno o varios bits, que se predisponen como resultado de alguna operación.

Como ejemplo, tomemos el caso de una operación aritmética, (ADD, SUB, etc) que puede disponer una condición de dos bits, en uno de cuatro valores posibles: 0, positivo, negativo o rebose, por lo que pueden haber cuatro condiciones de salto:

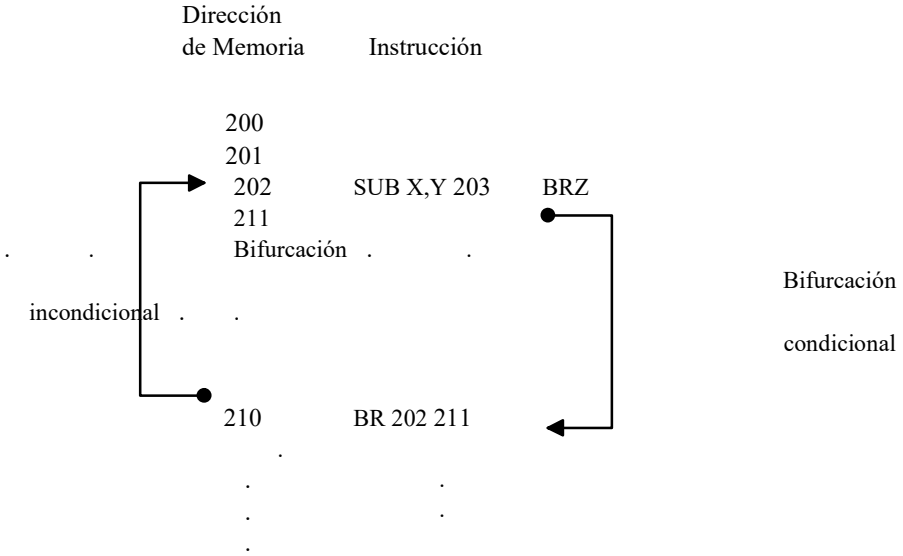
- BRP X: Salto a la dirección X si el resultado es positivo.
- BRN X: Salto a la dirección X si el resultado es negativo. - BRZ X: Salto a la dirección X si el resultado es cero.
- BRO X: Salto a la dirección X si hay un rebose (Overflow).

En todos estos casos el resultado al que se hace referencia es el obtenido de la operación "más recientemente" efectuada.

En caso de instrucciones de tres direcciones, es posible utilizar otra forma, en la cual se realiza una comparación y se especifica la bifurcación en la misma instrucción, por ejemplo:

BRE R1, R2, X = Salta a X si <R1>=<R2>

En la figura V.3, se tiene un ejemplo de estas operaciones, y además se puede notar que las bifurcaciones o saltos pueden ser hacia adelante o hacia atrás. Además se muestra como un salto incondicional puede ser utilizado para realizar un lazo repetido de instrucciones, hasta que el resultado de una operación es igual a cero, lo que ocurre en la operación Y-X.



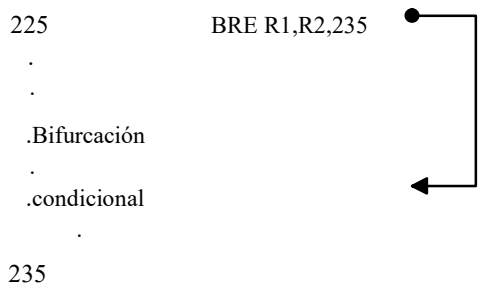


Figura II.3 - Ejemplo de instrucciones de bifurcación.

II.7.2 - INSTRUCCIONES DE SALTO

Otra forma común de transferencia del control, es dado por la instrucción de salto, la cual incluye una dirección implícita. En general el salto consiste en pasar por alto una instrucción, lo cual significa que la dirección de la próxima instrucción es el contenido del contador de programa, más una unidad.

Debido a que la instrucción de salto no precisa contener direcciones, es posible incorporarle otras condiciones en el campo correspondiente a la(s) misma(s). Como ejemplo, podemos considerar la instrucción: Incremente y salte si es cero (ISZ), que puede verse en el siguiente fragmento de programa:

```
 301
 .
 .
 .
309 ISZ  R1
310 BR 301
 311
```

En el mismo, se utilizan dos instrucciones, una de salto (skip) y otra de bifurcación (branch) para implementar un lazo iterativo.

Se carga un número negativo en R1, al que automáticamente se va agregando una unidad en cada lazo realizado, cuando el contenido de R1 llega al valor cero, la instrucción siguiente es salteada, en caso contrario se vuelve a la inicial (301).

II.7.3 - INSTRUCCIONES DE LLAMADO A SUBRUTINAS:

Posiblemente una de las innovaciones más importantes en el desarrollo de los programas, sea la incorporación de las subrutinas, las cuales son un programa autosuficiente, incorporables a un programa mayor. En cualquier punto del programa mayor o principal, es posible invocar (llamar) a una subrutina, con lo cual se transfiere el control a la misma, y se lo devuelve al programa mayor cuando se concluye su ejecución.

Las dos razones de mayor importancia para el uso de las subrutinas, son la

economía y la modularidad. La primera se refiere al ahorro de espacio en memoria y al menor trabajo de programación, la segunda, se refiere a que un gran programa puede ser subdivido en pequeñas unidades, lo cual también facilita la tarea de programación.

El mecanismo de la subrutina, implica dos instrucciones básicas, el llamado a la subrutina y el retorno al programa principal. Una instrucción de llamado o de retorno, es una instrucción de bifurcación o de salto que transfiere el control a otra dirección.

En la figura II.4, se muestra el uso de subrutinas para construir un programa. En éste ejemplo, hay un programa principal que comienza en la dirección 4000, y que incluye un llamado a la subrutina 1, que comienza en la locación 4500, cuando se encuentra la instrucción CALL SUB1 (que es el llamado a la subrutina 1), se suspende la ejecución del programa principal y se transfiere el control a la locación 4500.

Dentro de la subrutina 1, hay dos llamados a la subrutina 2, que comienza en la locación 4800, en ambos casos se suspende la ejecución de la subrutina 1 y se pasa el control a la subrutina 2. La sentencia RETURN hace que el control se transfiera al programa llamante, continuando su ejecución a partir de la instrucción ubicada inmediatamente después de la instrucción de llamado a la subrutina. En la figura V.5 se muestra este comportamiento.

Dirección Memoria Principal

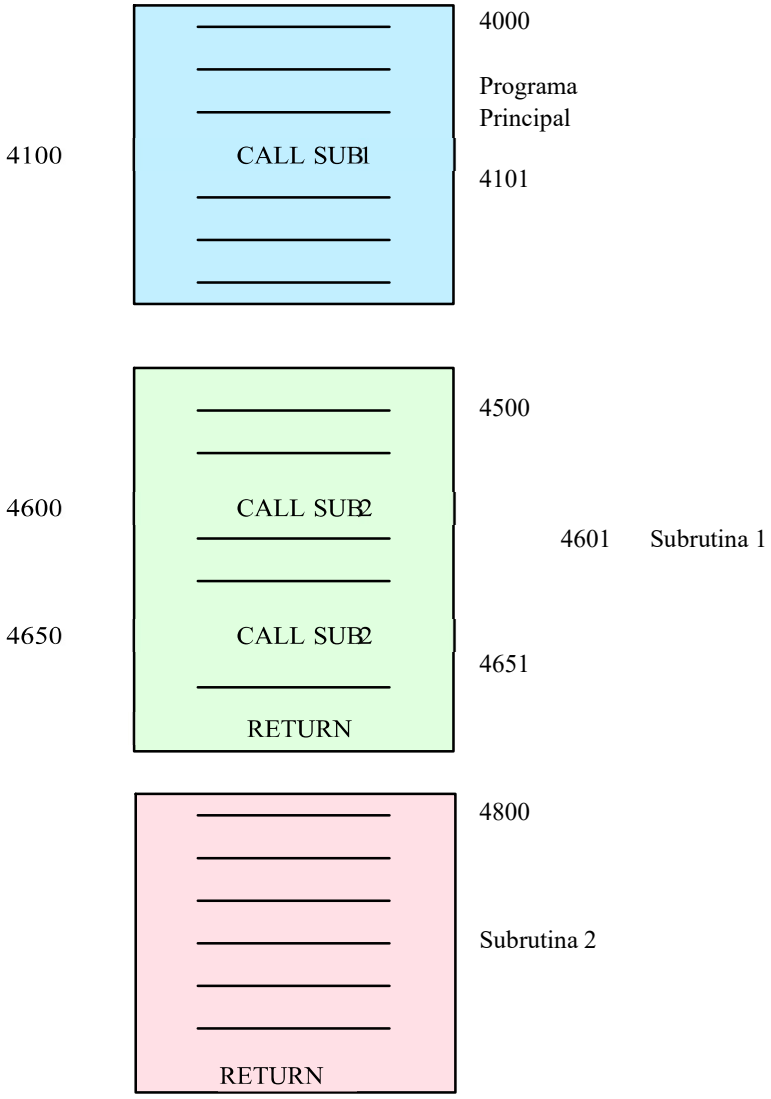


Figura II.4 - Subrutinas Anidadas.

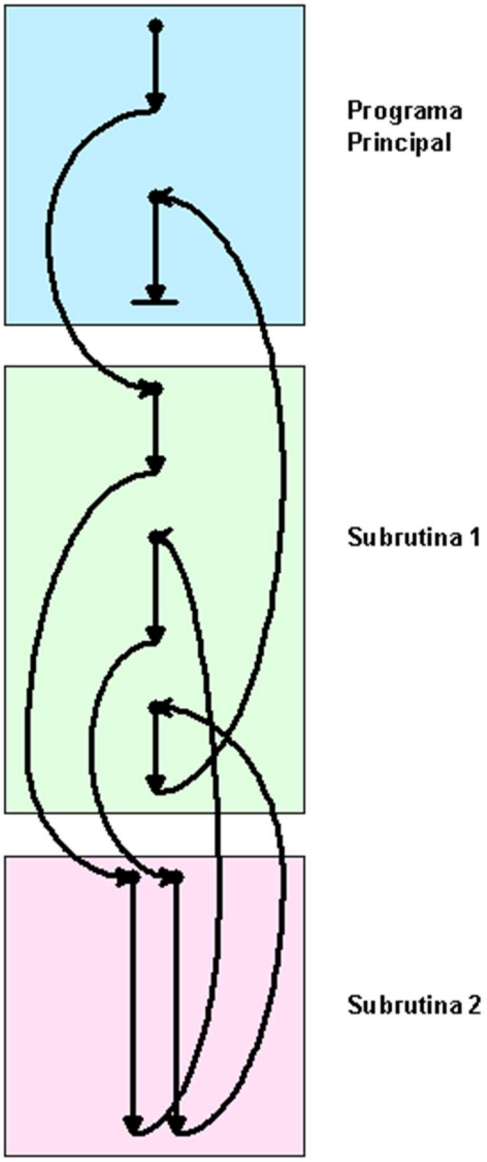


Figura II.5 - Secuencia de Ejecución de las subrutinas anidadas indicadas en la figura II.4.

II.8 - DIRECCIONAMIENTO:

Tal como ya citara, el o los campos de direccionamiento son de longitud limitada, por tanto, para alcanzar a direccionar todas la locaciones de memoria de un sistema, aún las de memoria virtual, es necesario emplear algunas técnicas que relacionen la capacidad disponible con la longitud de los campos. Algunos de estos métodos de direccionamiento son:

- Inmediato.
- Directo.
- Indirecto.
- por Registro.
- Indirecto por Registro.
- por Desplazamiento:
- Relativo.
- por Registro Base.
- Indexado:
- preindexado.
- postindexado.
- por Pila.

En la figura V.7 se indica en forma gráfica como actúa cada uno de estos modos. En la figura, y en lo que sigue, emplearemos la siguiente notación:

M = Parte mando de la Instrucción A = Contenido del campo de dirección
DE = Dirección Efectiva (X) =
Contenido de la locación X.

Para indicar el modo de direccionamiento a la unidad de control se emplean varias aproximaciones, de ellas, las más prácticas parecen ser dos:

- Agregar al mando un campo de modo de direccionamiento.
- Para diferentes códigos operativos, distintos modos de direccionamiento.

El término Dirección Efectiva, se refiere a la dirección real de memoria o de un registro. En sistemas de memoria virtual el concepto se amplia al de dirección virtual de memoria o de un registro.

II.8.1 - DIRECCIONAMIENTO INMEDIATO:

Es la forma más simple, en la cual el contenido del campo de dirección es directamente el operando a emplear. Es utilizado principalmente para definir constantes, o determinar valores iniciales para las variables.

OPERANDO = A

La ventaja que ofrece éste modo, es que no se hace referencia a memoria para obtener el operando, con lo cual se acelera el cálculo en aquellos casos donde esa constante o valor inicial son utilizados reiterativamente.

La desventaja es que el tamaño del número es restringido a la capacidad del campo de dirección, el cual en algunos casos es pequeño comparado con longitud de palabra.

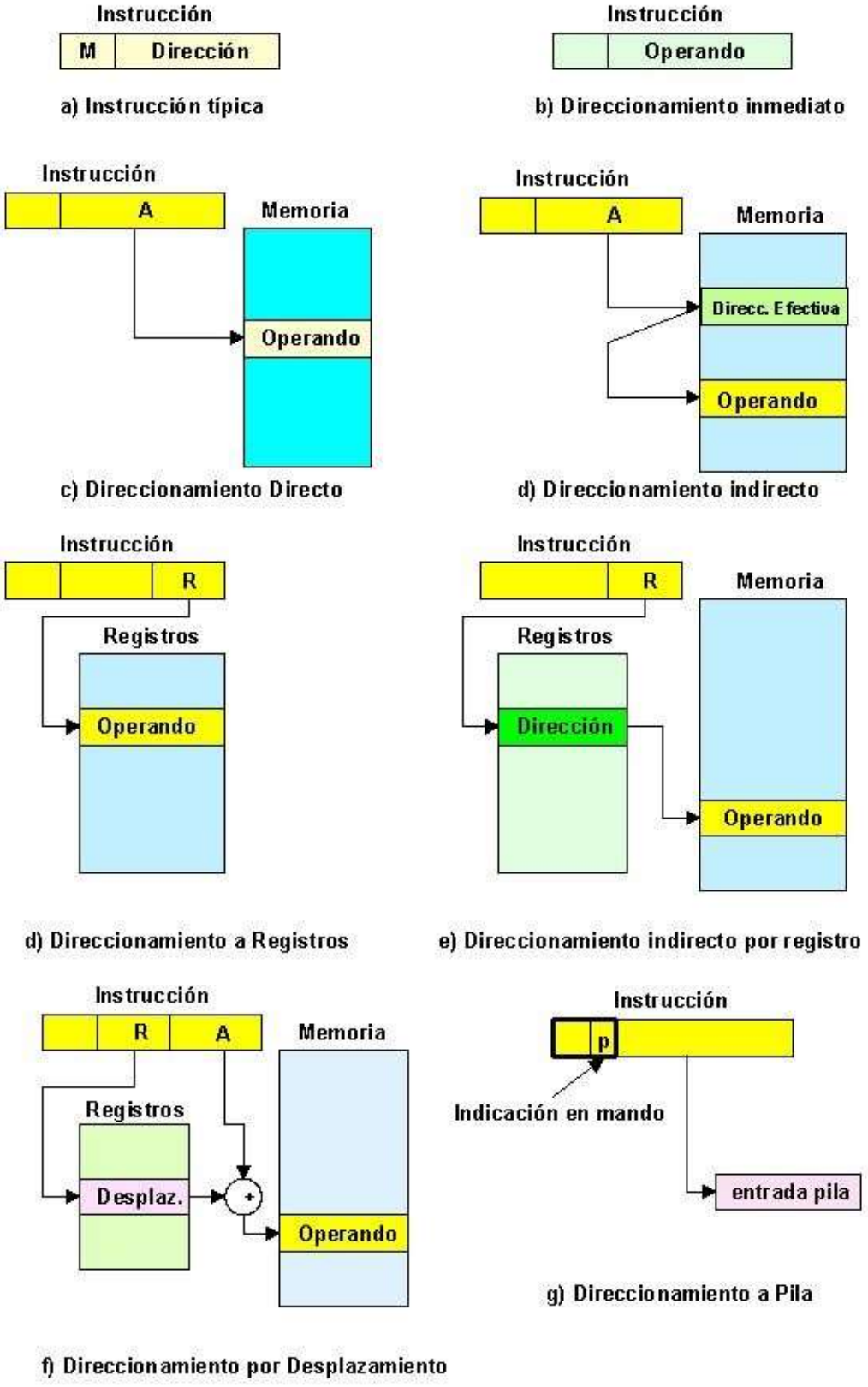


Figura II.7 - Modos de Direccionamiento.

II.8.2 - DIRECCIONAMIENTO DIRECTO:

Esta es la forma más simple y directa, dado que el contenido del campo de dirección es directamente la dirección efectiva del operando en la memoria.

DE = A

Esta técnica fue común en las primeras computadoras, y aún se lo utiliza en varias familias de pequeñas computadoras. Su única y limitativa desventaja, es que solo se puede direccionar memorias de poca capacidad.

II.8.3 - DIRECCIONAMIENTO INDIRECTO:

Dado que la principal desventaja del direccionamiento directo es que el campo de dirección es de longitud inferior al de una palabra, lo cual limita la capacidad de direccionar, en el modo indirecto, el campo de dirección en realidad indica una palabra de memoria donde está contenida la dirección completa (de una palabra) del operando.

DE = (A)

La ventaja del método es obviamente el que con una longitud de N bits es posible direccionar 2^N posiciones de memoria, pero con la desventaja de requerir dos accesos a memoria para obtener el operando.

II.8.4 - DIRECCIONAMIENTO A REGISTROS.

Este modo es similar al direccionamiento directo, solo que se hace referencia a un registro en vez que a una posición de memoria.

DE = R

Normalmente, un campo de direcciones que hace referencia a registros, es de solamente tres o cuatro bits, dado que se deben referenciar hasta 8 o 16 registros de propósitos generales.

La ventaja es que no se hace referencia a memoria para buscar un operando, y solo se necesita un pequeño campo de dirección. La desventaja es que el espacio de direcciones es extremadamente limitado.

II.8.5 - DIRECCIONAMIENTO INDIRECTO POR REGISTROS.

Este modo es completamente análogo al direccionamiento indirecto, solo que en vez de acceder a memoria para hallar la dirección, la misma se encuentra en un registro de la UCP.

DE = (R)

Las ventajas y limitaciones son análogas a las del direccionamiento indirecto.

II.8.6 - DIRECCIONAMIENTO POR DESPLAZAMIENTO:

Es el más poderoso modo de direccionamiento, que combina las ventajas del direccionamiento directo con las direccionamiento indirecto por registros.

$$DE = A + (R)$$

Requiere que haya dos campos de direccionamiento, uno referente al registro a utilizar y otro al desplazamiento a agregar al contenido de ese registro. Los tres usos más comunes de este tipo de direccionamiento serán vistos a continuación.

II.8.6.1 - DIRECCIONAMIENTO RELATIVO:

En este caso, la referencia implícita es el contado de programa (CP), a cuyo contenido es sumado el contenido del campo de dirección para producir la DE. Normalmente el contenido del campo de dirección es tratado como un número en complemento a dos, por lo que el desplazamiento puede ser hacia adelante o hacia atrás.

$$DE = (CP) + (A)$$

La dirección efectiva es función de la dirección de la instrucción, dada por el CP, por tanto esto permite una efectiva economía de bits en el direccionamiento.

II.8.6.2 - DIRECCIONAMIENTO POR REGISTRO BASE.

En este caso, el registro referenciado contiene la dirección de memoria, y el campo de dirección de la instrucción, contiene el desplazamiento, en forma tal que:

$$DE = (R) + (A)$$

No solo reúne las ventajas del modo anterior, sino que también permite direccionar todo un programa a partir de una cierta dirección de base, de aquí el nombre de registro base, lo cual permite la carga simultánea de varios programas, cada uno de los cuales es referenciado mediante un registro.

II.8.6.3 - DIRECCIONAMIENTO INDEXADO:

La interpretación del término indexación es la siguiente: El campo de dirección hace referencia a una dirección de memoria, y el registro referenciado contiene un desplazamiento positivo a partir de ésta dirección. (Esto es justamente lo contrario del direccionamiento por registro base)

Si bien la forma de calcular la dirección es la misma, en este caso el registro, generalmente llamado registro de índice, incrementa en una unidad cada vez que es referenciado, o sea que es:

$$DE = A + (R)\ (R) = (R) + 1$$

En algunas máquinas, es posible emplear conjuntamente el direccionamiento indirecto y el indexado, pasando ésta forma a tener el nombre de postindexado. El cálculo de la dirección efectiva es:

$$DE = (A) + (R)$$

El método preindexado, consiste en que la indexación es efectuada antes de calcular la dirección:

$$DE = (A + (R))$$

En la figura V.8 se tiene el diagrama correspondiente a cada caso.

Es así que este tipo de direccionamiento tiene gran utilidad para referenciar matrices o conjuntos de datos ubicados ordenadamente en memoria, así como para guardar sus resultados.

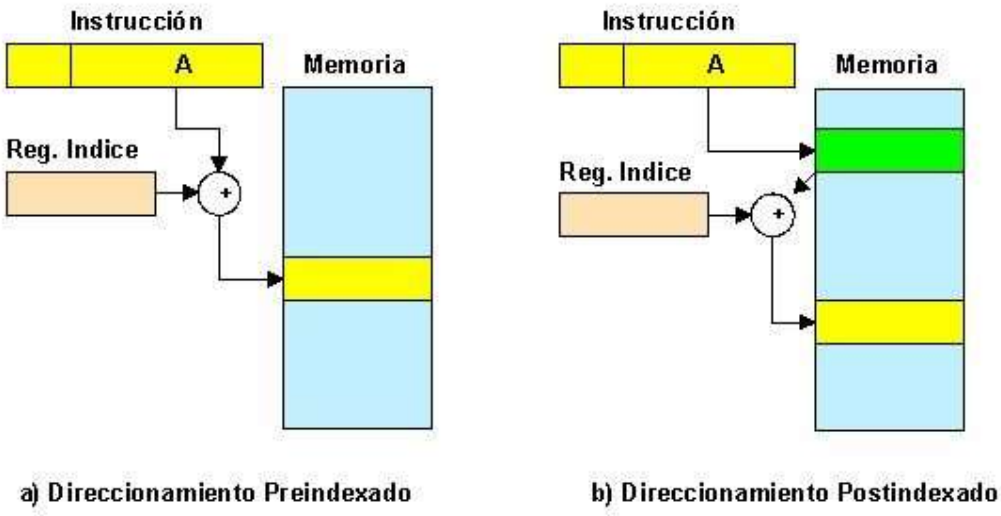


Figura II.8 -Direccionamiento Indexado.

Existen otras formas, en las cuales se aplica simultáneamente el direccionamiento relativo y el indexado, o se utiliza un índice relativo, pero estos casi no son utilizados en aplicaciones prácticas.

II.8.7 - DIRECCIONAMIENTO DE LA PILA.

Tal como antes definimos, la pila es una matriz lineal de posiciones de memoria, la cual es direccionada mediante indicación de la primera o la última, según sea la convención adoptada. Cada vez que se introduce un dato, toda la pila se mueve un lugar, dando cabida en el alojamiento de entrada al nuevo dato, así como cada vez que se referencia, extrayendo un dato, la pila se mueve dejando accesible el almacenado inmediatamente antes del extraído.

Por tanto, la manera práctica de hacer esto, no es mover todos los datos, sino

tomar una cierta cantidad de posiciones de memoria y un puntero de pila, el que apunta siempre a la última casilla ocupada. Cada vez que se introduce un dato el puntero cambia a la casilla recién ocupada, y cada vez que se extrae uno, el puntero vuelve atrás.

El valor del puntero de la pila se mantiene en un registro, por lo que el direccionamiento de la pila es en realidad un direccionamiento relativo al contenido del mismo. También ésta es una forma implícita de direccionamiento, por lo que no es necesario hacer referencia a memoria en el campo de direcciones, sino que siempre se opera en el registro de puntero de pila.

II.8.7.1 - DIRECCIONAMIENTO DE CARGA/ALMACENAMIENTO.

El Power PC provee dos modos de direccionamiento alternativos para las instrucciones de carga/almacenamiento (Load/Store).

Con el direccionamiento indirecto, la instrucción incluye un desplazamiento de 16 bits para ser sumado al contenido de un registro de base, que puede ser cualquiera de los registros de propósitos generales. Asimismo, la instrucción puede especificar que la recientemente calculada dirección efectiva, puede ser realimentada al registro de base, cuyo contenido resultará actualizado. Este modo es utilizado para indexar progresivamente las matrices ubicadas dentro de lazos.

La otra técnica, consiste en aplicar el direccionamiento indirecto indexado, con lo cual la instrucción referencia un registro de base y un registro de índice, que pueden ser cualquiera de los registros de propósitos generales. La dirección efectiva es la suma de los contenidos de estos dos registros. Por supuesto que otra vez puede emplearse la opción de actualización a la nueva dirección efectiva.

II.9 - FORMATOS DE LAS INSTRUCCIONES.

El formato de una instrucción define la disposición de los bits en la misma, en función de sus partes constituyentes.

La instrucción debe incluir un código operativo, y ninguno o uno o varios operandos referenciados en forma explícita o implícita, empleando uno de los modos de direccionamiento indicados anteriormente. Obviamente, el formato de la instrucción debe incluir también en forma explícita o implícita el modo de direccionamiento de cada operando.

En muchos conjuntos de instrucciones, se utiliza más de un formato para las mismas. El diseño del formato es un arte complejo, por lo que existe una asombrosa variedad.

II.9.1 - LONGITUD DE UNA INSTRUCCIÓN:

La selección de la longitud de una instrucción es un factor que afecta y a su vez es afectada por el tamaño de la memoria, su organización, la estructura de buses, la complejidad de la UCP, su velocidad, etc. La decisión determina la riqueza y la flexibilidad de la máquina, vista desde la posición del programador en lenguaje ensamblable.

Un poderoso repertorio de instrucciones, hace el trabajo del programador más simple, pero encarece el hardware de la máquina, y reduce su capacidad de almacenamiento.

A la inversa, con un repertorio simple, se aumenta el trabajo del programador, pero se hace más eficiente la utilización de la memoria y se aumenta la velocidad de cálculo.

Normalmente la longitud de la instrucción es igual al ancho del bus, aunque se implementan sistemas en los cuales una es múltiplo de la otra. En caso que la instrucción sea de mayor longitud que el bus, la transferencia de la misma desde la memoria se hará en más de un acceso. Mientras que cuando ocurre lo contrario, se desperdicia capacidad.

Lo lógico es que todo guarde una armónica proporción, entonces la longitud de una instrucción deberá ser igual a la longitud de una palabra, interpretando como tal a la mínima unidad direccionable en memoria, y el ancho del bus deberá ser igual a esta longitud.

II.9.2 - UBICACIÓN DE LOS BITS.

Para una dada longitud de instrucción, hay una clara relación entre la cantidad de códigos operativos posibles y la capacidad de direccionamiento. Una gran cantidad de códigos operativos, obviamente necesitan una mayor cantidad de bits, restando los dedicados al campo de dirección.

Un posible refinamiento es el de utilizar códigos operativos de longitud variable, pero habrá una longitud mínima, a partir de la cual se agregarán algunos bits, lo cual siempre termina por restar capacidad de direccionamiento.

Los siguientes factores interrelacionados determinan la cantidad de bits dedicados al direccionamiento:

- Cantidad de modos de direccionamiento: Algunas veces el modo de direccionamiento es implícito en el código operativo, pero cuando hay varios de ellos, es necesario agregar un campo para determinarlo, lo cual implica la utilización de algunos bits.

- Cantidad de operandos: Hemos visto anteriormente cuantas direcciones se necesitan para programar eficientemente, digamos que en la actualidad se prefieren las instrucciones con dos campos de dirección, en el cual los dos operandos requieren su referencia al modo de direccionamiento.

- Registros y memoria: Toda máquina debe poseer registros para permitir la realización de operaciones sobre los operandos. Cuando hay uno solo, generalmente el acumulador, la referencia al mismo es implícita, y no se utilizan bits para hacerlo, sin embargo, la programación con uso de un solo registro es muy pesada, requiriendo una gran cantidad de instrucciones. Aún con varios registros, los bits necesarios para direccionarlos son unos pocos, además esos registros pueden ser utilizados para referenciar operandos. Estudios recientes indican que es preferible tener de 8 a 32 registros visibles al usuario.

- Cantidad de conjuntos de registros: Ciertas máquinas poseen un conjunto de 8 a 16 registros de propósitos generales. Estos pueden ser utilizados para almacenar datos y también direcciones en el caso de direccionamiento por desplazamiento. En la actualidad, la tendencia es hacia dos o más bancos de

registros especializados, (por ejemplo: para datos y para desplazamientos) tanto en microprocesadores como en supercomputadoras. La ventaja principal, es que para un cierto número de registros, el salto de uno a otro conjunto requiere muy pocos bits de la instrucción. Por ejemplo, con dos conjuntos de ocho registros, solo se requieren tres bits para identificar a cada uno de ellos, mientras que el conjunto puede ser determinado implícitamente por el código operativo.

- Rango de direcciones: Para direcciones con referencia a memoria el rango de direcciones que puede ser referenciado está dado por la cantidad de bits del campo de direcciones. Dado que esto impone severas restricciones, el direccionamiento directo es raramente utilizado. Con el sistema del desplazamiento, el rango se amplía hasta la capacidad del registro de direcciones. Asimismo, es conveniente tener grandes desplazamientos, lo cual nuevamente requiere un gran número de bits en el campo de direcciones de la instrucción.

- Granularidad de la dirección: Para direcciones que hacen referencia a la memoria, otro factor a tener en cuenta es la granularidad de la dirección. En un sistema con palabras de 16 o 32 bits, una dirección puede referenciar a una u otra longitud de palabra. Para la manipulación de caracteres es necesario que la unidad direccionable sea un byte, pero ello lleva a que para una misma memoria, la cantidad de bits de la dirección sea mayor.

De esto se concluye que el proyectista está enfrentado a una serie de requerimientos encontrados, por lo que debe considerar una solución de compromiso. Si bien se han realizado muchos estudios sobre el tema, aún no hay una clara referencia que indique el tamaño más conveniente de la instrucción y sus campos.

II.12 - OPERACION DE LA UNIDAD DE CONTROL.

Después de ver como son las instrucciones y los datos, pasaremos a tratar la actuación de los circuitos encargados de hacerlas cumplir.

Sabemos que la función de una computadora es ejecutar programas, y la ejecución de un programa consiste en la ejecución de una serie de ciclos de instrucción, a razón de una instrucción por ciclo.

También debemos recordar que la secuencia de ciclos de instrucción, no necesariamente responde a la secuencia en que se han escrito las instrucciones, por cuanto existen las instrucciones de bifurcación.

II.12.1 - MICRO-OPERACIONES:

Hemos visto que un ciclo de instrucción en realidad está formado por una cierta cantidad de eventos, en primer lugar, por un sub-ciclo de búsqueda y por un subciclo de ejecución, que a su vez están compuestos por una serie de pequeñas operaciones, tales como la habilitación de compuertas para el envío de señales, la espera por una respuesta de otro módulo (por ejemplo, el de memoria), la remisión de señales de desplazamiento, el traslado de la información, etc.

Este conjunto de ordenes, necesarias para el cumplimiento de un ciclo de instrucción, puede ser tomado como formado por un conjunto de micro-operaciones.

II.12.1.1 - EL SUB-CICLO DE BUSQUEDA:

Comenzaremos observando como es un sub-ciclo o ciclo de búsqueda, que tiene lugar al comienzo de un ciclo de instrucción y es utilizado para buscar una instrucción en la memoria.

A fin de proponer un punto de partida para el análisis, podemos suponer una organización tal como la indicada en la figura II.9, y además hay cuatro registros involucrados:

- Registro de direcciones de memoria (MAR): Conectado a las líneas del bus de direccionamiento y especifica la dirección de memoria donde se debe escribir o leer.

- Registro Buffer de memoria (MBR): Está conectado a las líneas del bus de datos, conteniendo el valor a ser almacenado o el último valor leído en memoria.

- Contador de Programa (PC): Retiene la dirección de la próxima instrucción a ser buscada.

- Registro de Instrucciones (IR): Retiene la última instrucción buscada.

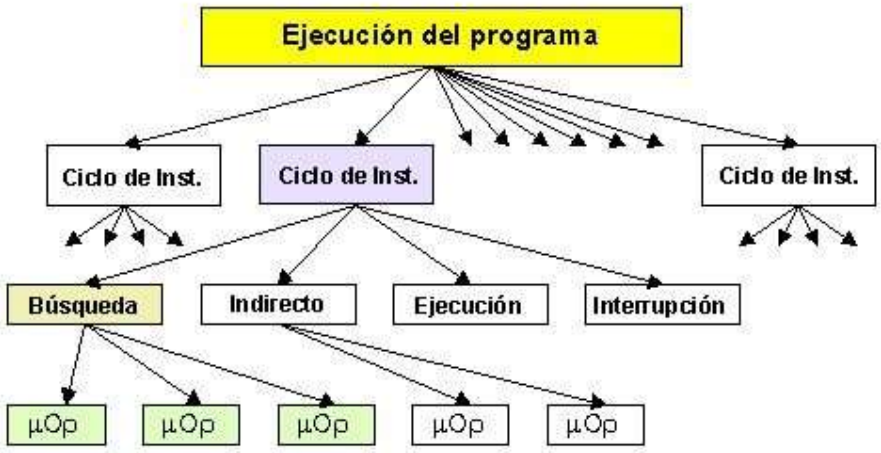


Figura II.9 - Elementos de la ejecución de un programa.

La secuencia de eventos de un sub-ciclo de búsqueda, es la siguiente:

1 - La Dirección de la próxima instrucción a ser ejecutada es tomada del PC, y entregada al bus de direcciones, al cual está conectado el MAR.

2 - Se emite desde la Unidad de Control al mando LEER, el que es entregado al bus de control. Es ubicada la dirección en la memoria y el contenido de la correspondiente posición es entregado al MBR, el cual está conectado al bus de datos.

3 - Se suma una unidad al contenido del PC.

4 - Se pasa el contenido del bus de datos al IR.

Estos eventos, en cuanto no se interactúen entre sí, pueden realizarse al mismo tiempo, es así que podemos escribir simbólicamente:

t1: MAR <-- (PC) t2: MBR <-- (Memoria)
PC <-- (PC) + 1
t3: IR <-- (MBR)

Toda máquina posee un reloj central que emite una serie continua de pulsos regularmente espaciados, en consecuencia, todas las unidades de tiempo (intervalos entre dos pulsos) son iguales. La indicación t1, t2, t3 indica instantes de tiempo correspondientes a los pulsos del reloj, por lo que cada una de las operaciones es llevada a cabo en el mismo tiempo, y así también ocurre en t2, donde se agrega la actualización del PC, lo cual no interfiere con ninguna otra operación.

II.12.1.2 - EL CICLO INDIRECTO:

Una vez que se ha traído una instrucción al IR, el próximo paso, en el más general de los casos, será el de buscar los operandos.

Continuando con el simple ejemplo que estamos viendo, suponiendo que las instrucciones son de una dirección, con la posibilidad de tener direccionamiento directo e indirecto. Si la instrucción especifica una dirección indirecta, un ciclo, que denominaremos indirecto, deberá preceder al ciclo de ejecución. Este ciclo indirecto, comprende las siguientes micro-operaciones:

t1: MAR <-- (IR(Dirección))
t2: MBR <-- (Memoria) t3:
IR(Dirección) <-- (MBR(Dirección))

Lo que significa que el campo de dirección es transferido al MAR, donde luego es utilizado para buscar la dirección del operando. Con el contenido de la dirección encontrada, es actualizado el contenido del IR, por lo que ahora contiene la dirección directa del operando.

A partir de éste momento el IR se encuentra en el mismo estado que si no se hubiera utilizado el direccionamiento indirecto, y está listo para iniciar el sub-ciclo de ejecución.

II.12.1.3 - EL CICLO DE INTERRUPCIÓN:

Al concluir el sub-ciclo de ejecución, se realiza la prueba para determinar si

se ha presentado alguna interrupción válida. Si la prueba resulta afirmativa, se produce un ciclo de interrupción, el que suele ser muy diferente para distintos procesadores. Por lo que tomaremos una muy simple secuencia de eventos:

t1: MBR <-- (PC) t2: MAR <-- Guarda-dirección PC <-- Dirección-rutina t3: Memoria <-- (MBR)

En el primer paso, el contenido del PC es transferido al MBR, en forma tal que se puede salvar la dirección de retorno de la interrupción. A continuación el MAR es cargado con la dirección en la cual debe ser salvado el contenido del PC, y el contador es cargado con la dirección de inicio de la rutina de interrupción. Estas dos acciones pueden ser agrupadas en una micro-operación.

Sin embargo, dado que hay muchos procesadores que proveen varios tipos o niveles de interrupciones, la operación puede precisar más micro-operaciones para llegar a salvar el contenido del PC.

De cualquier manera una vez salvados los datos necesarios, la UCP está preparada para seguir con el próximo ciclo de instrucción.

II.12.1.4 - EL SUB-CICLO DE EJECUCIÓN:

Los sub-ciclos de búsqueda, el indirecto y el de interrupción, son simples y predecibles, por cuanto cada uno implica una pequeña secuencia fija de microoperaciones, y en algunos casos, la misma micro-operación es repetida varias veces.

En el sub-ciclo de ejecución en cambio, si la máquina tiene N códigos operativos distintos, hay N secuencias de micro-operaciones diferentes.

Para demostrarlo, tomemos una instrucción hipotética, y desarrollémosla:

ADD R1, X

en la cual se suma el contenido de la locación X al contenido del registro R1, la secuencia de micro-operaciones puede ser:

t1: MAR <-- (IR(Dirección))
t2: MBR <-- Memoria t3:
R1 <-- (R1) + (MBR)

lo que significa que comenzamos con el IR que contiene la instrucción ADD. En el primer paso, la parte dirección de IR es llevada al MAR, referenciando así a una posición de memoria, la cual es leída y su contenido se lleva al MBR. Finalmente, los contenidos de R1 y MBR son sumados en la ULA, y el resultado almacenado en R1.

Supongamos ahora un ejemplo un poco más complicado, tomando la instrucción para incrementar y saltar si es cero, que se escribe:

ISZ X

lo que significa que el contenido de la locación X es incrementado en una unidad, y si el resultado es cero, se transfiere el control del programa pasando por alto la instrucción siguiente.

La posible secuencia de micro-operaciones, puede ser:

t1: MAR <-- (IR(dirección)) t2:
MBR <-- Memoria
t3: MBR <-- (MBR) + 1 t4:
Memoria <-- (MBR)
Si ((MBR)=0) luego (PC<--(PC)+1)

Esta es una acción condicionada, el contador es incrementado en una unidad si el contenido del MBR es cero. Esta prueba y acción puede se implementada en una sola micro-operación, y nótese que además esta micro-operación puede ser llevada a cabo simultáneamente mientras el valor actualizado del contenido del MBR es almacenado en memoria.

Para terminar, consideremos un llamado a una subrutina, para lo cual tomemos una instrucción bifurque y guarde la dirección:

BSA X

La dirección de la instrucción que sigue es almacenada en la locación X, y la ejecución continúa en la locación X+1. La dirección almacenada será posteriormente utilizada para el retorno.

La secuencia de micro-operaciones necesaria puede ser la siguiente:

t1: MAR <-- (IR(Dirección))
MBR <-- (PC) t2: PC <--
(IR(Dirección)) Memoria <--
(MBR)
t3: PC <-- (PC) + 1

La dirección contenida en el PC al comienzo de la instrucción es la dirección de la próxima instrucción en la secuencia. Ella es almacenada en la dirección indicada por el IR. La dirección es luego incrementada para proveer la dirección de la instrucción para el próximo ciclo.

II.12.1.5 - EL CICLO DE INSTRUCCIÓN:

Hemos visto como cada fase del ciclo de instrucción puede ser descompuesta en una serie de micro-operaciones elementales. En nuestro ejemplo, hay una secuencia para cada sub-ciclo, o sea que hay una secuencia de micro-operaciones por cada código operativo.

Para completar la imagen, necesitamos enlazar estas secuencias de microoperaciones, lo que es indicado en la figura V.10, donde suponemos tener un nuevo

registro de dos bits, denominado Código del Ciclo de Instrucción (ICC), el cual indica el estado de la UCP para cada porción del ciclo que se está ejecutando. Según lo indicado en la figura V.16, estos estados corresponden a:

00: Búsqueda
01: Indirecto
10: Ejecute
11: Interrupción

Al final de cada uno de los cuatro sub-ciclos, ICC es cargado apropiadamente. El ciclo indirecto siempre es seguido por el de ejecución. El de interrupción siempre es seguido por el de búsqueda y finalmente para ambos, el de búsqueda y el de ejecución, el próximo sub-ciclo depende del estado del sistema.

II.12.2 - CONTROL DE LA UCP.

Como resultado del análisis precedente, hemos descompuesto el funcionamiento de la UCP en operaciones elementales, denominadas micro-operaciones, por cuanto nuestra motivación es la de entender como pueden entregarse las señales de control provenientes de la Unidad de Control, para obtener este funcionamiento.

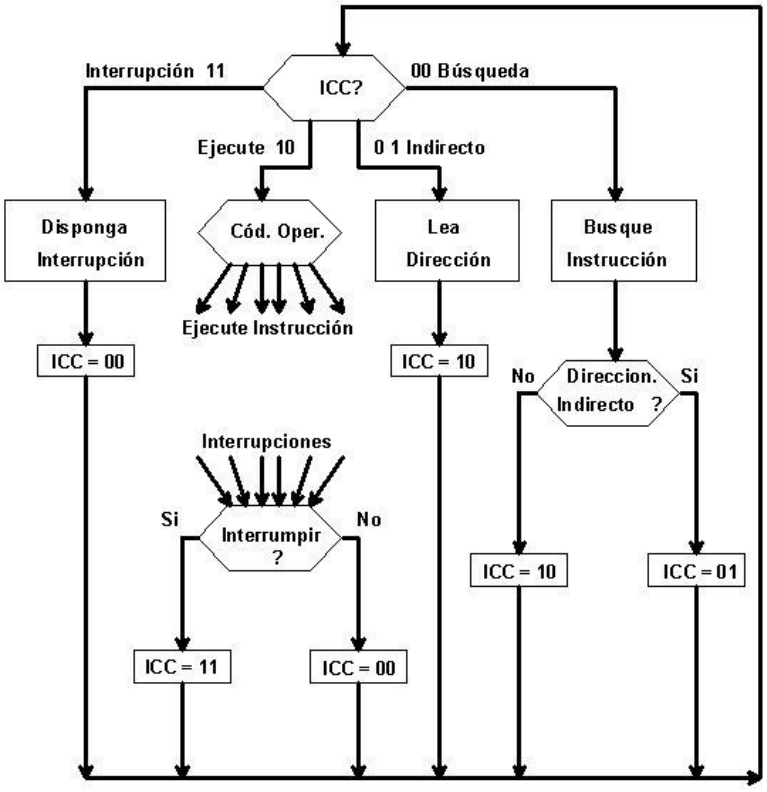


Figura II.10 - Diagrama de flujo del ciclo de instrucción.

II.12.2.1 - REQUERIMIENTOS FUNCIONALES:

Para reducir la operación de la UCP a su nivel fundamental, es necesario definir exactamente que debe hacer la unidad de control para que ocurra exactamente una determinada acción.

Esto hará que podamos definir las necesidades o requerimientos funcionales de la Unidad de Control, o sea, que funciones deberá realizar en cada caso, con lo cual podremos proceder a su diseño lógico.

Con la información que poseemos, podemos indicar cuales son los tres pasos principales para caracterizar la Unidad de Control:

1 - Definir los elementos básicos de la UCP.

2 - Describir las micro-operaciones que la UCP realiza.

3 - Determinar las funciones que la unidad de control debe cumplir para provocar la ejecución de las micro-operaciones indicadas. A este punto, los dos primeros procesos ya los tenemos en claro, por lo que podemos resumir los resultados:

1 - La CPU debe contener los siguientes elementos funcionales:

- ULA
- Registros
- Rutas de datos internas - Rutas de datos externas
- Unidad de control

2 - La ejecución de un programa consiste de operaciones que implican a estos elementos de la CPU, pudiendo clasificar las micro-operaciones a llevar cabo en alguna de las siguientes categorías:

1 - Transferir datos desde y entre registros.
2 - Transferir datos de un registro a un módulo de E/S 3 - Transferir datos de un módulo de E/S a un registro
4 - Llevar a cabo una operación lógica o aritmética, utilizando registros para la entrada y la salida.

3 - Todas las micro-operaciones necesarias para completar un ciclo de instrucción, incluyendo todas las micro-operaciones para ejecutar cualquier instrucción, caen dentro de una de estas cuatro categorías.

Ahora podemos decir algo más sobre como funciona la Unidad de Control, ella lleva a cabo dos tareas básicas:

- Secuenciamiento: La Unidad de control hace que la UCP vaya pasando por una secuencia de micro-operaciones, en la secuencia apropiada, basada en el programa que se está ejecutando.

- Ejecución: La unidad de control hace que cada micro-operación sea cumplida.

II.12.2.2 - SEÑALES DE CONTROL:

La descripción funcional anterior, nos dice sobre que debe hacer la Unidad de control, la llave de como opera, es el uso de señales de control.

Para que la Unidad de Control lleve a cabo sus funciones, debe tener entradas que le permitan determinar el estado del sistema y salidas que le posibiliten controlar el comportamiento del sistema. Estas son las especificaciones externas, internamente, la unidad de control debe tener la lógica requerida para llevar a cabo las funciones de secuenciamiento y ejecución.

Comencemos por tratar de comprender la interacción entre la unidad de control y los demás componentes de la UCP. En la figura II.11, tenemos un modelo general de la unidad de control, que es el mismo presentado en la figura II.1, pero dispuesto ahora para poner en evidencia las señales de entrada y las de salida.

Las entradas son:

- Reloj: La unidad de Control provee una o una serie de micro-operaciones en cada pulso de reloj, por tanto esto es denominado "ciclo del procesador" o "ciclo de reloj".

- Registro de Instrucciones: El código operativo de la presente instrucción es utilizado para determinar cuales son las micro-operaciones a llevar a cabo durante el ciclo de ejecución.

- Banderas: Estas son señales necesarias para que la unidad de control determine el estado de la UCP y de los resultados anteriormente obtenidos en la ALU. Por ejemplo, en la instrucción ISZ, la unidad de control debe incrementar al PC si la bandera "0" está activa.

- Señales de Control del Bus de Control: Un sector del bus de control debe proveer las señales de solicitud de interrupción y de reconocimiento.

Las salidas son:

- Señales de control para la UCP: de las cuales hay dos tipos, las que provocan el movimiento de datos entre registros, y las que activan funciones específicas de la ALU.

- Señales de control para el bus de control: Que también son de dos tipos, las que son destinadas a la memoria y las que actúan sobre los módulos de E/S.

Los nuevos elementos introducidos en la figura II.11, son las señales de

control, de las cuales, según dijimos, hay tres tipos: Las que activan funciones de la ALU, las que activan rutas de señales y las que son para el bus de control o para las interfases externas. Todas estas señales son aplicadas directamente como señales binaras a la entrada de compuertas lógicas.

Consideremos ahora nuevamente el ciclo de búsqueda, para ver como la unidad de control lleva a cabo justamente ese control.

Recordemos que la unidad de control sabe permanentemente en que lugar está del ciclo de instrucción, por tanto, en un dado punto, se da cuenta que está por empezar un sub-ciclo de búsqueda. El primer paso es el de transferir el contenido del PC al MAR, por tanto, deberá activar la señal de control que abre las compuertas lógicas existentes entre los bits del PC y los bits del MAR.

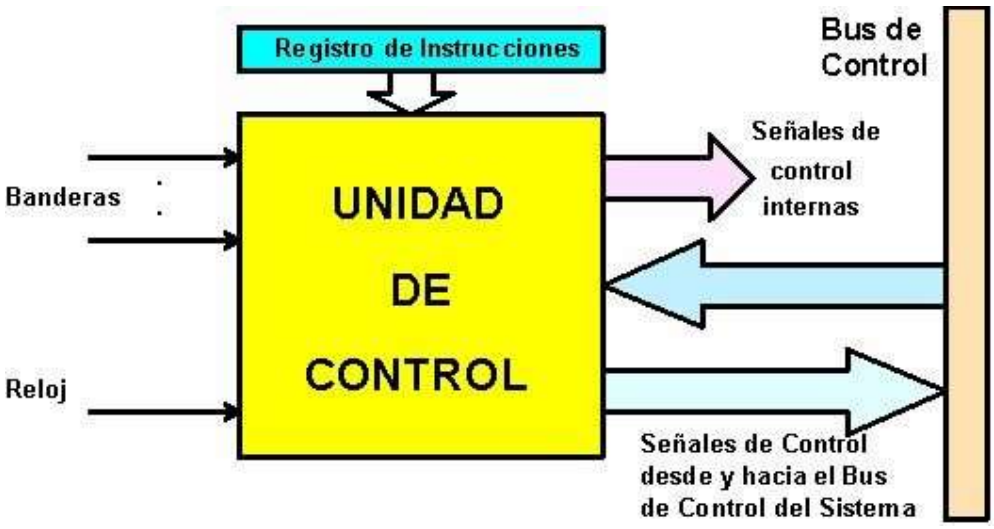


Figura II.11 - Modelo Simplificado de la Unidad de Control.

El próximo paso es leer una palabra en la memoria, pasándola al MBR e incrementando al PC, por lo que deberá enviar las siguientes señales de control:

1 - Una señal que abra las compuertas para que el contenido del MAR pase al bus de direcciones.

2 - Una señal de control para lectura de la memoria, al bus de control.

3 - Una señal de control que abra las compuertas para que el contenido del bus de datos sea entregado al MBR.

4 - Señales de control para sumar una unidad al contenido del PC.

A continuación, la Unidad de Control envía una señal de control que abre las compuertas existentes entre el MBR y el IR, con lo cual se completa el sub-ciclo de búsqueda.

Para seguir con el ciclo de instrucción, ahora la Unidad de Control deberá

saber que tipo de sub-ciclo llevar a cabo para la búsqueda del operando, si es el directo o el indirecto, para decidirlo, examina el IR. Tanto el ciclo indirecto como el de interrupción, operan en modo similar.

Para realizar el sub-ciclo de ejecución, la Unidad de Control examina el Código Operativo, y en base a él determina cuales es la secuencia de micro-operaciones a realizar.

II.13 - IMPLEMENTACION DE LA UNIDAD DE CONTROL:

Hasta ahora hemos estudiado la Unidad de Control en función de señales de entrada, de salida y de funciones. Es tiempo de que veamos como puede implementarse. Hay una amplia variedad de técnicas que pueden ser aplicadas, la mayoría de las cuales cae en una de las siguientes dos categorías:

- IMPLEMENTACIÓN CABLEADA.
- IMPLEMENTACIÓN MICROPROGRAMADA.

En la primera, la Unidad de Control, es esencialmente un circuito combinacional, donde las entradas son transformadas en un conjunto de señales lógicas, que son las señales de control.

II.13.1 - ENTRADAS A LA UNIDAD DE CONTROL:

La figura II.11 pinta bastante bien como es una Unidad de Control simplificada, en ella vemos las señales de entrada salida y de función que hemos descrito, consideremos ahora el registro de instrucciones (IR). La Unidad de Control hace uso del código operativo para llevar a cabo diferentes acciones para diferentes instrucciones. Mediante un decodificador, es posible tener que para cada código, haya una sola salida. En general, el decodificador puede tener n entradas binarias y dar 2^n salidas también binarias, en consecuencia, habrá una salida por cada uno de los códigos operativos previstos. En la tabla siguiente tenemos un ejemplo de como ocurre este proceso:

I_1	I_2	I_3	I_4	O_1	O_2	O_3	O_4	O_5	O_6	O_7	O_8	O_9	O_{10}	O_{11}	O_{12}	O_{13}	O_{14}	O_{15}	O_{16}
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
1	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0

1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Lógicamente que un decodificador para una UC, debe ser algo más sofisticado que el indicado, por cuanto se debe tener en cuenta las diferentes longitudes de los códigos operativos o su presencia en diferentes partes de la instrucción. De cualquier manera, de ésta forma es posible analizarlo.

La sección de reloj de la UC, entrega una secuencia continua de pulsos, los cuales se emplean para determinar la duración de las micro-operaciones. Pero además, es necesario emitir diferentes señales de control, en diferentes instantes de tiempo, dentro de un ciclo de instrucción. Por tanto será necesario un contador y un circuito combinacional para permitir la presencia de estas señales. Este circuito es conocido con el nombre de circuito de tiempo. Con estos dos refinamientos, el circuito simplificado de la UC, queda como el indicado en la figura II.12.

II.13.2 - LÓGICA DE LA UNIDAD DE CONTROL.

Para definir la implementación cableada de la Unidad de Control, todo lo que queda por hacer es discutir la lógica interna que permitirá obtener las señales de control.

Esencialmente, lo que se debe hacer es derivar la función booleana de cada señal en función de las entradas. Esto es mejor explicado mediante un ejemplo.

Supongamos una sencilla UCP, que posee un solo acumulador AC, e indiquemos los caminos entre los distintos elementos. Estos serán los caminos de datos. Asimismo, tal como se indica en la figura II.13, proveamos de compuertas a los caminos, compuertas (indicadas mediante círculos) que serán habilitadas o abiertas desde la unidad de control, mediante las señales C_1 ..... $C_{13,}$ y también se tienen otras señales, que son las que predisponen o conectan los circuitos internos de la ULA para que resuelva la operación indicada por la instrucción.

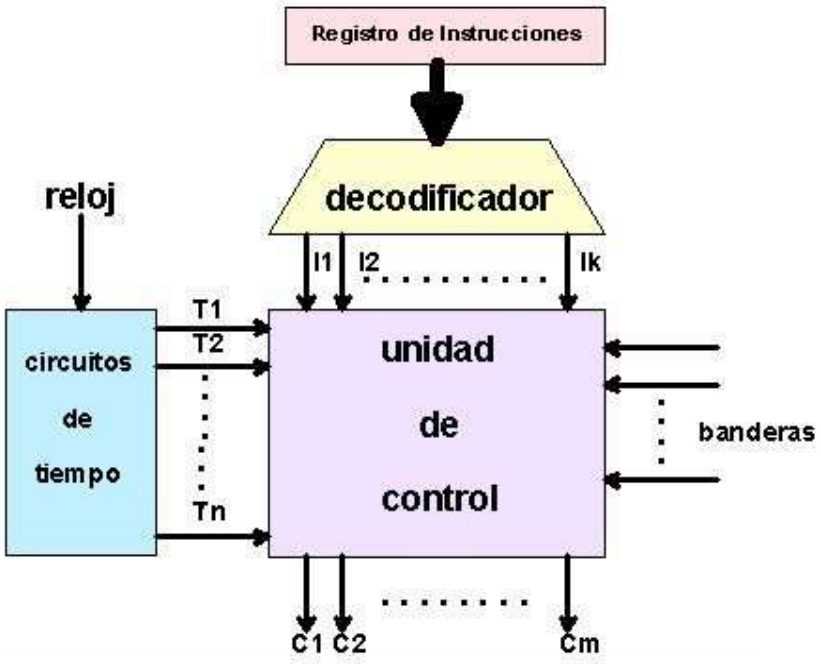


Figura II.12 - Unidad de Control con entradas decodificadas.

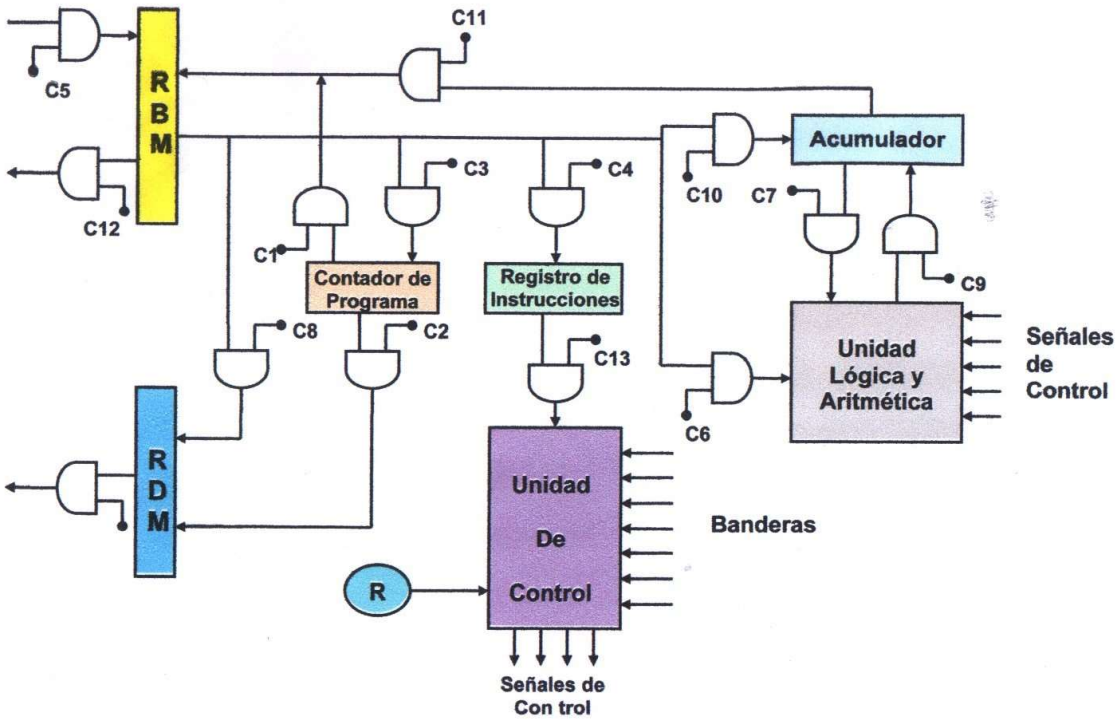


Figura II.13 - Rutas de Datos y Señales de Control.

Asimismo, en la siguiente tabla tenemos las micro-operaciones a realizar, y las señales de control correspondientes.

Micro-operacion	Timing	Señales de Control activas.
Búsqueda:	t_1 : MAR <-- (PC)	C_2

	t_2 : MBR <-- Memoria	C_5 ; C_R
	PC <-- (PC) + 1	
	t_3 : IR <-- (MBR)	C_4
Indirecto:	t_1 : MAR <-- (IR(Direcc)	C_8
	t_2: MBR <-- Memoria	C_5 ; C_R
	t_3: IR(Dirección) <-- (MBR(Dirección)	C_4
Interrupción:	t_1 : MBR <-- (PC)	C_1
	t_2: MAR <-- Guarda-Direcc	
	PC <-- Dir. Rutina	
	t_3 : Memoria <-- (MBR)	C_{12} ; C_W

En la tabla además se ha indicado C_R y C_W con el significado de señal de control de lectura y señal de control de escritura para el bus del sistema, que no están indicadas en el gráfico.

Consideremos ahora solamente la señal C_5, que es la que provoca que los datos sean leídos desde el bus externo y pasen al MBR, la cual es utilizada dos veces en la tabla anterior. Definamos también dos nuevas señales de control, P y Q, con la siguiente interpretación:

PQ = 00: Sub-ciclo de Búsqueda
PQ = 01: Sub-ciclo indirecto
PQ = 10: Sub-ciclo de ejecución
PQ = 11: Sub-ciclo de Interrupción

La expresión booleana que define a C_5 es:

$$C_5 = \overline{P}\,\overline{Q}T_2 + \overline{P}QT_2$$

Lo cual indica que la señal debe ser aplicada en el segundo período de tiempo, o sea en t_2, en ambos sub-ciclos, el de búsqueda y el indirecto.

Esta expresión no está completa, dado que C_5 también es necesaria durante el sub-ciclo de ejecución. Para nuestro ejemplo, supongamos que hay solo tres instrucciones que permiten la lectura desde la memoria, LDA, ADD y AND, por lo que podemos redefinir nuestra C_5 como:

$$C_5 = \overline{P}\,\overline{Q}T_2 + \overline{P}QT_2 + P\overline{Q}LDA + ADD + AND)T_2$$

El mismo proceso puede ser repetido para cada señal de control a generar, y entonces el resultado será un conjunto de ecuaciones booleanas a resolver para implementar el combinacional de control.

Podemos intuir que para los actuales procesadores, con un complejo y extenso conjunto de instrucciones, la cantidad de ecuaciones booleanas a resolver se vuelve muy grande, de varios cientos de ellas, con lo cual el problema resulta altamente dificultoso.

II.14 - CONTROL MICROPROGRAMADO.

El término "Microprograma" fue utilizado por primera vez por Maurice Wilkes en la década del 50, proponiendo un nuevo diseño de la unidad de control que evitara las complejidades del modelo cableado.

Aparece por primera vez en la IBM/360, aunque no en los modelos de mayor tamaño, luego con la disponibilidad de grandes memorias ROM de semiconductor, se desarrollo ampliamente, tanto que en la actualidad es el sistema utilizado por casi todos, en mayor o menor medida.

II.14.1 - CONCEPTOS BÁSICOS:

La unidad de control, tal como ha sido descripta anteriormente, parece ser un dispositivo razonablemente simple, pero sin embargo su implementación es a veces sumamente dificultosa, y sometida a diversos errores, los cuales se advierten al tener que disponer cables adicionales al circuito impreso en la zona correspondiente.

II.14.1.1 - MICROINSTRUCCIONES.

Consideremos nuevamente la última tabla, donde se indican las micro-operaciones descriptas en notación simbólica. Esta notación se parece notablemente a un lenguaje de programación, y de hecho lo es, y es denominado "Lenguaje de Microprogramación". Cada línea describe un conjunto de micro-operaciones que debe ser llevada a cabo en un cierto instante de tiempo, y que es conocida como "Microinstrucción".

La secuencia de estas microinstrucciones es conocida con el nombre de "Microprograma", o también "Firmware", término éste que refleja el hecho de que un microprograma es la vía intermedia entre el Software y el Hardware. ¿Cómo podemos utilizar esto para diseñar unidades de control?

Consideremos que por cada micro-operación todo lo que tiene que hacer es generar un conjunto de señales de control. De aquí que por cada micro-operación, cada línea de control emergente está en "1" o en "0". Esta condición puede ser representada por una palabra binaria, que tiene un dígito para cada una de estas líneas.

En consecuencia, podemos construir la llamada "palabra de control", en la cual cada bit representa una línea de control, por tanto cada micro-operación puede ser representada por una combinación diferente de unos y ceros en la palabra de control.

A continuación, podemos pensar en un listado de palabras de control, que

lógicamente representará una secuencia de micro-operaciones llevadas a cabo por la unidad de control. Por otra parte, debemos reconocer que la secuencia de microoperaciones no es fija, dado que a veces podemos tener un ciclo indirecto y a veces no.

Pongamos ahora la secuencia de palabras de control en una memoria, con cada palabra teniendo una sola y única dirección. Luego, agreguemos un campo de dirección a cada una, indicando la dirección de la próxima palabra de control a ser ejecutada si se cumple con cierta condición, para lo cual deberán agregarse algunos bits para indicar ésta condición.

El resultado es conocido como "microinstrucción horizontal", y es mostrado en la figura V.14.a, de donde se desprende el formato siguiente:

Hay un bit por cada línea de control interna y otro por cada línea de control del bus de control. También hay un campo de condición y hay un campo con la dirección de la microinstrucción que debe ejecutarse a continuación. Tal microinstrucción debe interpretarse como sigue:

1 - Para ejecutar esta microinstrucción, active todas las líneas de control correspondientes al bit "1" y desactive todas las líneas de control correspondientes al bit "0".

2 - Si la condición indicada por los bits de condición es falsa, ejecute la próxima instrucción en la secuencia.

3 - Si la condición indicada por los bits de condición es verdadera, la próxima microinstrucción a ser ejecutada es la indicada en el campo de dirección.

Lógicamente este sistema precisa muchas posiciones binarias, por tanto conviene emplear un sistema de codificación, lo cual nos lleva a una "Microinstrucción vertical", que es la indicada en la figura II.14.b.

En la figura II.16 se tienen los elementos clave de tal implementación. El conjunto de microinstrucciones es almacenado en la memoria de control, el registro de direcciones de control contiene la dirección de la próxima microinstrucción que debe ser leída. Cuando es leída una microinstrucción en la memoria de control, es transferida al registro buffer del control.

La parte derecha de dicho registro, se conecta con las líneas de control que salen de la unidad de control, así que leer una microinstrucción en la memoria de control es lo mismo que ejecutarla.

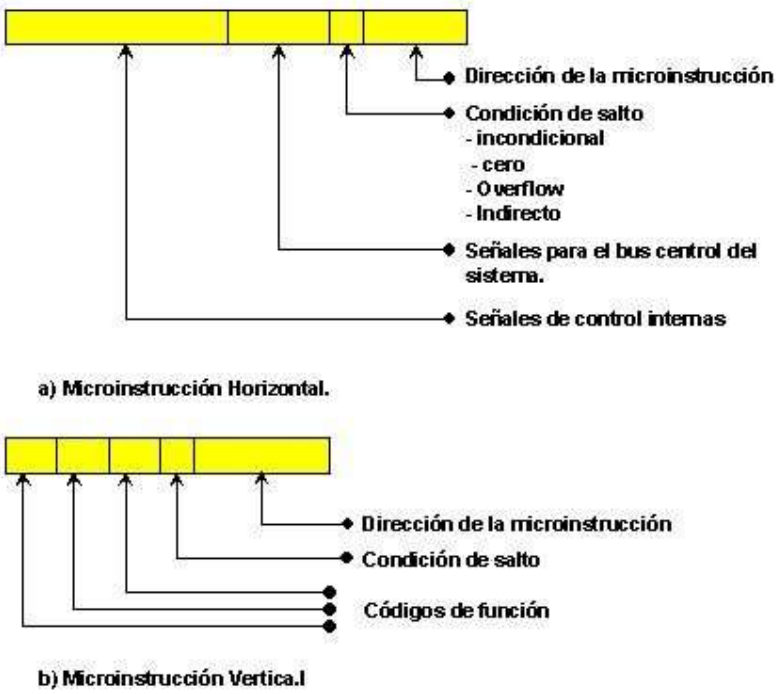


Figura II.14 - Formatos típicos de microinstrucciones.

En la figura II.15, se muestra como las palabras de control pueden ser dispuestas en una "memoria de control". Las microinstrucciones de cada rutina deben ser ejecutadas secuencialmente. Cada rutina termina con un una bifurcación o un salto, indicando donde se debe buscar la próxima.

La Memoria de control de la figura II.15, puede tomarse como una concisa descripción de la operación completa de la Unidad de Control, dado que describe la secuencia de micro-operaciones a ser llevada a cabo durante cada sub-ciclo (búsqueda, indirecto, ejecución e interrupción), y además indica el secuenciamiento de estos.

II.14.1.2 - UNIDAD DE CONTROL MICROPROGRAMADA.

La memoria de control de la figura II.16, contiene un programa que describe el comportamiento de la unidad de control, en consecuencia, podemos intuir que para implementar la unidad de control, simplemente debemos ejecutar el programa

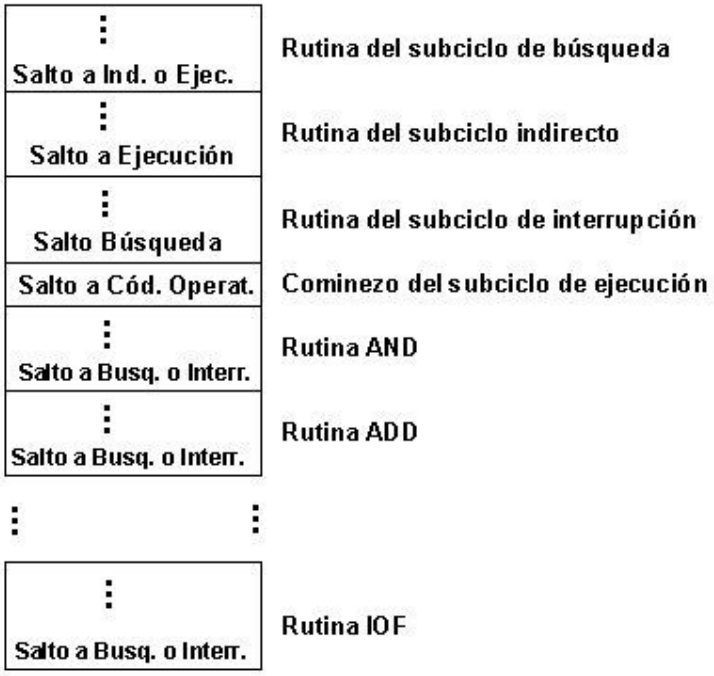


Figura II.15 -Organización de la Memoria de Control.

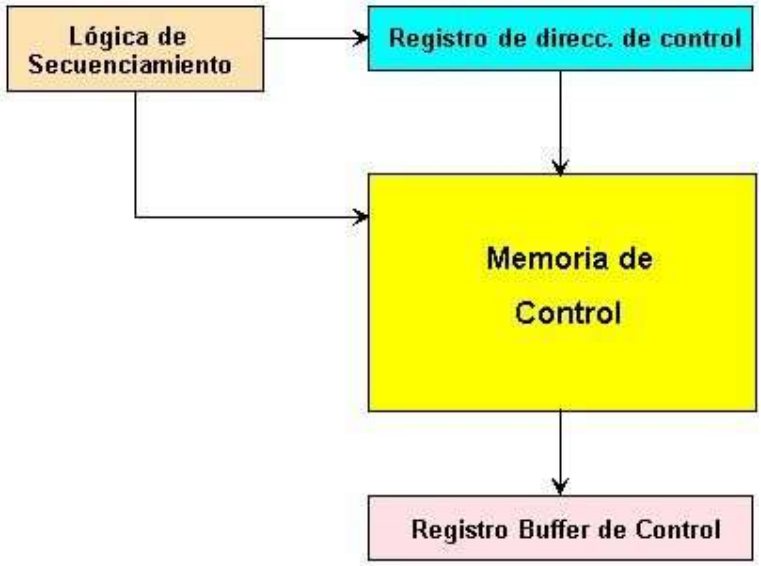


Figura II.16 - Microarquitectura de la Unidad de Control.

El tercer elemento mostrado en la figura es la unidad de secuenciamiento que alimenta al registro de direcciones de control y distribuye el comando leído.

A fin de examinar esta estructura más en detalle, pasemos al diagrama de la figura II.17, que es comparable al de la figura II.16, que corresponde al modelo de unidad de control planteado, dado que tiene las mismas entradas (IR, Banderas de la ULA, reloj) y salidas (señales de control). Esta Unidad de Control funciona de la siguiente manera:

1 - Para ejecutar una instrucción, la lógica de secuenciamiento entrega un comando de lectura a la memoria de control.

2 - La palabra cuya dirección es especificada por el registro de direcciones del control, es pasada al registro buffer del control.

3 - El contenido del registro buffer de control genera las señales de control y la información de próxima dirección que necesita la unidad lógica de secuenciamiento.

4 - La Unidad lógica de secuenciamiento carga una nueva dirección en el registro de direcciones del control, en base a lo indicado por la información de nueva dirección proveniente del registro buffer del control y de las banderas de la ULA.

TODO ESTO OCURRE DURANTE UN SOLO PULSO DE RELOJ.

El último paso indicado, necesita alguna elaboración. En el final de cada microinstrucción, la unidad lógica de secuenciamiento carga una nueva dirección en el registro de direcciones del control, dependiendo del valor de las banderas de la ULA y el registro buffer del control, en base al árbol de decisiones realizado de la siguiente manera:

- Obtener la próxima instrucción: Sumar 1 al registro de direcciones del control.

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

- Saltar a una rutina de instrucción de máquina: Carga el registro de direcciones del control de acuerdo al código operativo contenido en el IR.

La misma figura II.17, muestra dos módulos denominados decodificador. El dispuesto en la parte superior, traduce el código operativo contenido en el IR a una dirección de memoria de control.

El decodificador inferior, no es utilizado con microinstrucciones horizontales, pero si es necesario cuando se emplean microinstrucciones verticales.

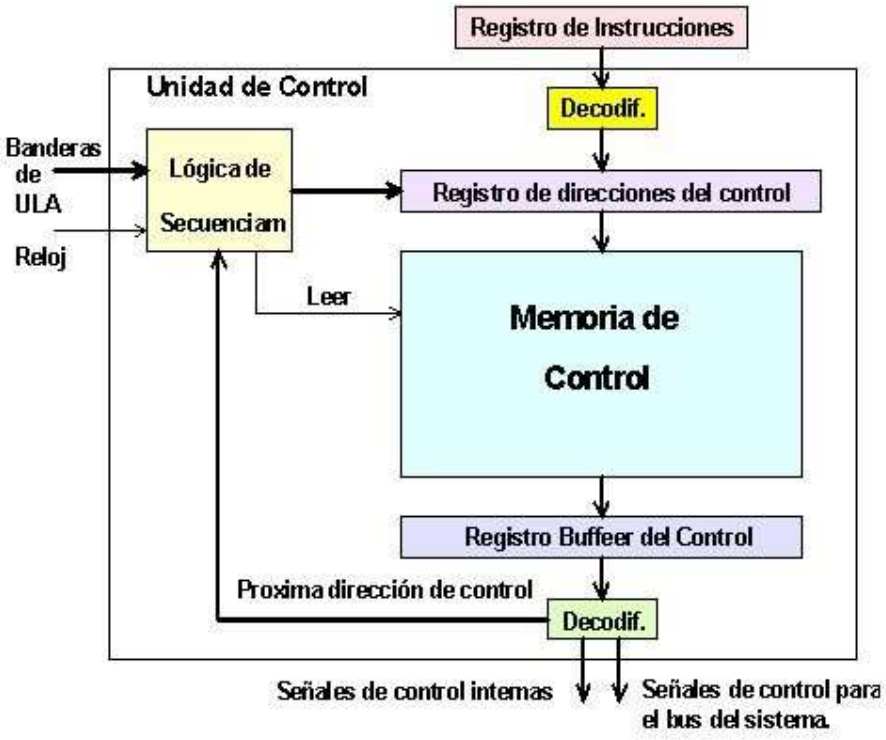


Figura II.17 - Funcionamiento de la Unidad de Control microprogamada.

II.13.1.3 - CONTROL DE WILKES.

Tal como hemos citado, Wilkes propuso por vez primera el uso de un control microprogramado, el que diagramó según lo indicado en la figura II.18. El corazón del sistema es una matriz de diodos, parcialmente llena. Durante un ciclo de máquina, una fila de la matriz es activada mediante un pulso, lo que genera señales en los puntos donde están conectados lo diodos (marcados con puntos en el diagrama).

La primera parte de la fila genera señales de control, que son las enviadas a controlar la UCP, mientras que la segunda parte genera la dirección de la fila que debe ser pulsada en el próximo ciclo de máquina. En consecuencia, cada línea de la matriz es una microinstrucción, y la matriz en si misma es la memoria de control.

En el inicio de un ciclo, la dirección de la fila a ser pulsada es contenida en el Registro I. Esta dirección es la entrada del decodificador, el que, cuando es activado por el pulso de reloj, activa esa fila de la matriz.

De acuerdo con lo que indiquen las señales de control, durante el mismo ciclo, se pasa al Registro II ya sea el código operativo contenido en el registro de instrucciones, o la segunda parte de la fila pulsada, que contiene la dirección de la próxima microinstrucción a ser ejecutada.

El contenido del Registro II es pasado al Registro I por un pulso de reloj. En consecuencia, se utilizan pulsos de reloj alternados para activar una fila de la matriz, y para transferir el contenido del Registro II al Registro I.

Esta disposición de dos registros es necesaria por cuanto el decodificador es simplemente un circuito combinacional, y con un solo registro, la salida se vuelve entrada durante el mismo ciclo, provocando una condición inestable.

Este esquema es muy similar a la microprogramación horizontal descripta anteriormente. La principal diferencia es que en la descripción previa, el registro de direcciones del control puede ser incrementado en una unidad para entregar la nueva dirección. En el esquema de Wilkes, la próxima dirección es contenida en la misma microinstrucción.

Para permitir la bifurcación, una fila debe contener dos partes dirección, controladas por una señal de condición, tal como se muestra en la figura.

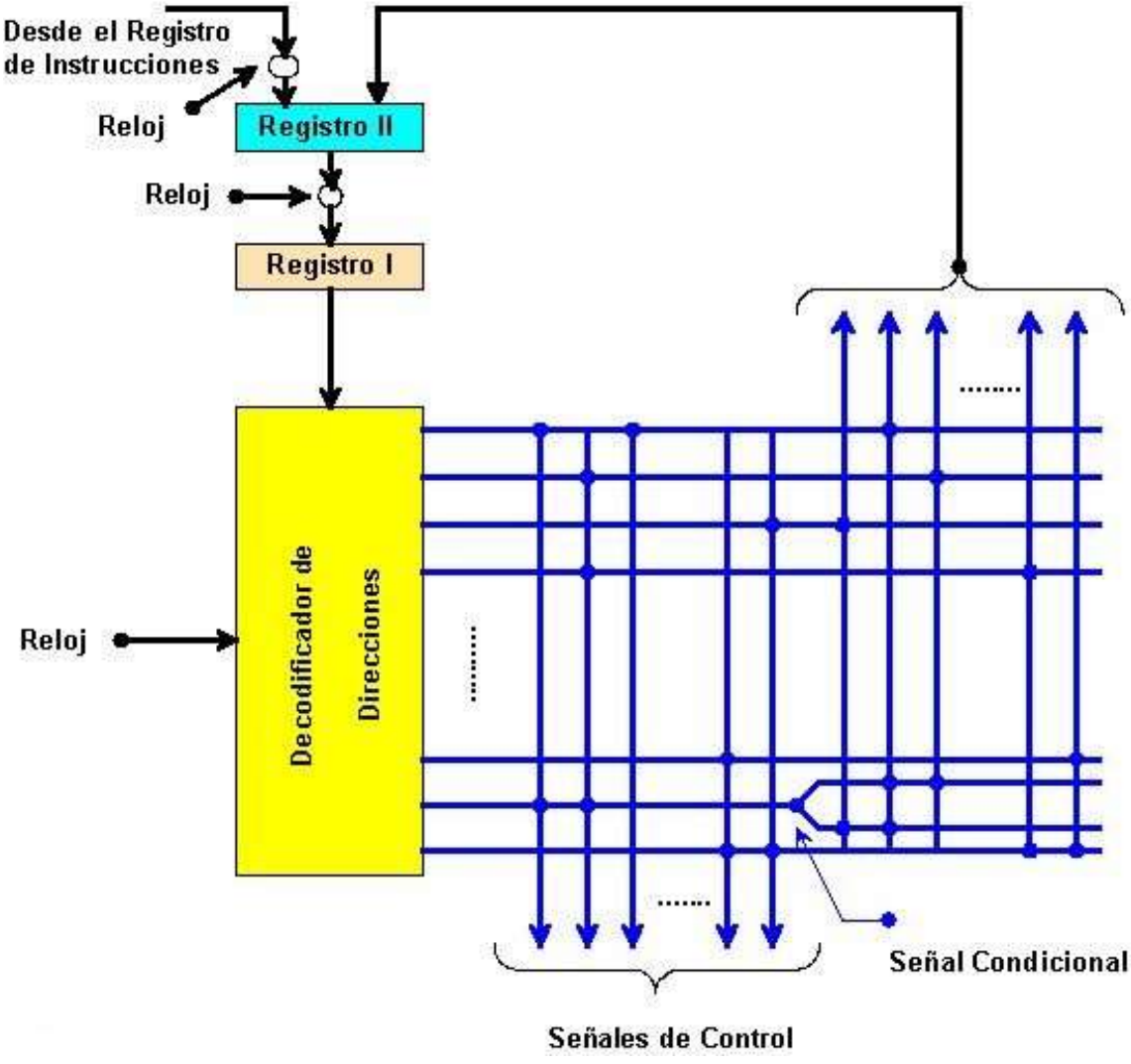


Figura II.18 - Unidad de control microprogramada según Wilkes.

II.13.2 - VENTAJAS Y DESVENTAJAS DE LA MICROPROGRAMACIÓN.

La principal ventaja es poder implementar una unidad de control en forma bastante sencilla. Además el microprograma es almacenado en una ROM, al cambiarse la misma es posible cambiar la actuación de la máquina.

Uno de los hechos importantes, es la flexibilidad del sistema y la posibilidad de emular otras máquinas, dado que pueden hacerse que una máquina trabaje con el conjunto de instrucciones de máquina de otra, a través de la microprogramación.

Como principal desventaja, es que la velocidad de la unidad de control microprogramada es mucho menor que la del sistema cableado.