TEMA 5. CONTROL EN EL SISTEMA COMPUTADOR

Este tema estudia la Unidad de Control del Computador; el elemento del computador Von Neumann que forma parte de la CPU, cuya misión es definir en todo instante el valor de todas las señales de control, tanto las internas a la propia CPU como las correspondientes a la conexión de la misma con el resto de elementos, la Memoria Principal y las Unidades de Entrada/Salida.

Las informaciones que emplean las unidades de control de los computadores son las de tipo instrucción. Una instrucción define la operación a realizar, el/los modo/s de direccionamiento empleado/s y el tipo de operando a considerar. La información relacionada con el tipo de operando, los sistemas de representación, se ha tratado en el Tema 2; en este tema se comienza estudiando tanto los modos de direccionamiento como los tipos de instrucciones que de manera general emplean los sistemas computadores.

La Unidad de Control, para definir en cada instante el valor de las distintas señales de control emplea como informaciones, las correspondientes a la instrucción presente en el registro de instrucción, el contenido del registro de estado y el número de periodo de ejecución de esa instrucción en el que se encuentre.

5.1. INTRODUCCIÓN

El trabajo de un computador consiste en el procesamiento de los datos de acuerdo con un programa de instrucciones almacenadas en la memoria principal. Mientras que la unidad operativa tiene la misión de llevar a cabo las operaciones aritméticas y lógicas que indican las instrucciones, la unidad de control se encarga de interpretar y controlar la ejecución de dichas instrucciones. La memoria principal es la que almacena los datos y el programa en ejecución y, por último, las unidades de E/S transfieren la información entre la CPU y los periféricos en ambos sentidos. Un sistema computador tipo *VON NEUMANN* tiene la estructura representada en la Figura 5.1.

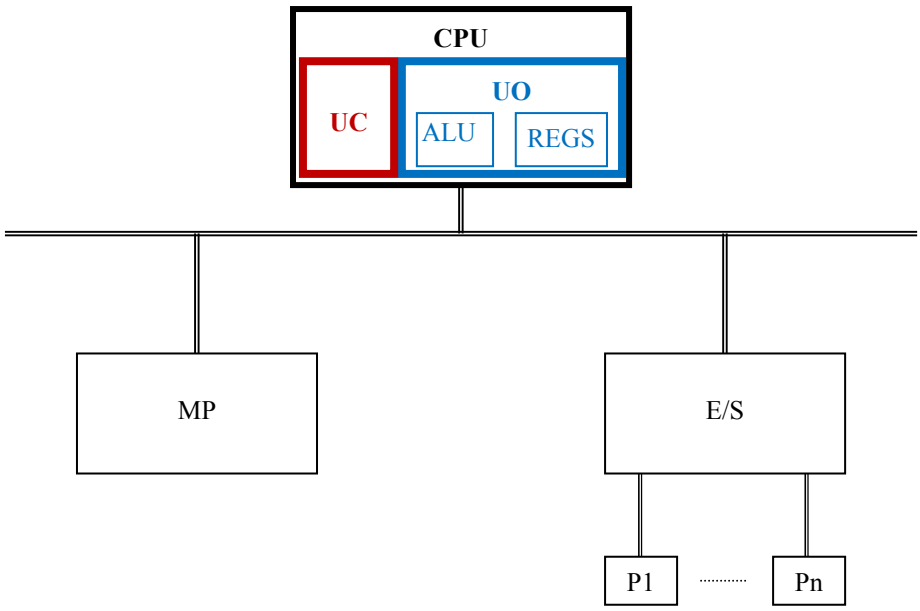

Figura 5.1 *Estructura VON NEUMANN.*

La unidad de control de un sistema computador tiene como función básica la ejecución de la secuencia:

- Tomar de la memoria principal la instrucción que se encuentra en la dirección indicada por el contador de programa, e incrementar adecuadamente este contador.
- Interpretar o decodificar la instrucción leída.
- Ejecutar la instrucción.

Además de las tres funciones básicas de lectura, decodificación y ejecución, la unidad de control se encarga de resolver las situaciones anómalas o de conflicto que puedan ocurrir en el computador y de controlar y comunicarse con los periféricos.

La información que emplea la unidad de control para poder realizar su cometido es:

- Instrucción.
- Registro de estado.
- Contador de periodos.
- Señales de E/S.
- Contador de programa.

La *instrucción* contiene el código de operación e información relativa de los operandos. El código de operación indica la operación a realizar así como la forma de tratar la información relativa a los operandos (modos de direccionamiento) contenida en la instrucción. La unidad de control deberá localizar los operandos, mandarlos, en su caso, a la unidad aritmético – lógica y almacenar el resultado.

El *registro de estado* contiene información relativa a los resultados obtenidos en operaciones anteriores, así como relativa a posibles situaciones anómalas o especiales, tales como desbordamientos, interrupciones, errores de paridad, etc., que exigen una acción especial por parte de la unidad de control. En general podemos decir que la información de estado se emplea para hacer rupturas de secuencia condicionales, del programa en curso. Estas rupturas pueden estar programadas en el propio programa, mediante instrucciones de bifurcación condicionales, o pueden ser automáticas, como ocurre con las interrupciones externas y de situaciones de error (no programadas).

El *contador de periodos* le permite diferenciar los correspondientes periodos de que consta la instrucción. Hay que tener en cuenta que la ejecución de cada instrucción requiere realizar una serie de pequeños pasos que llamaremos operaciones elementales, tales como suma de base más desplazamiento, lectura del operando, incremento del contador de programa, ejecución de una operación aritmética, etc. La ejecución de cada una de estas operaciones elementales, requiere la activación de una serie de señales de control. Será, por tanto, el OBJETIVO de la unidad de control, “la generación de las señales de control que permiten realizar las distintas operaciones elementales de cada instrucción”.

Las *señales de E/S* permiten el diálogo con los periféricos. Estas señales de E/S vienen a través de las unidades de E/S ya que, como sabemos, los periféricos se conectan al sistema computador a través de sus unidades de E/S.

Por último, la unidad de control emplea la información del *contador de programa* para producir el secuenciamiento de las instrucciones. La unidad de control se encargará de ir incrementando adecuadamente este registro y de cargarlo, en su caso, con nuevos valores para producir las bifurcaciones, salvando su contenido anterior si éstas son con retorno.

5.2. LENGUAJE MÁQUINA. INSTRUCCIONES MÁQUINA

El *lenguaje máquina* es el que puede interpretar y ejecutar directamente el computador. Está compuesto por una serie de instrucciones que forman parte del juego o repertorio de instrucciones del computador. Generalmente todas las instrucciones máquina son muy simples.

Las cuatro propiedades más importantes que cumplen las instrucciones máquina son:

- Realizan una única y sencilla operación. De esta forma se simplifica la operación de decodificación.
- Emplean un número fijo de operandos, los cuales adoptan una representación determinada.
- La codificación de las instrucciones se hace de forma sistemática con el fin de simplificar la decodificación posterior por parte de la unidad de control.
- Las instrucciones son *autocontenidas* e *independientes*, lo cual significa que poseen toda la información precisa para su ejecución y no dependen de la posición que ocupen en la memoria o en el programa.

El *repertorio de instrucciones* de un sistema computador es el conjunto de instrucciones que es capaz de interpretar el mismo. Su definición y características vienen fijadas por los siguientes puntos:

a) Las representaciones posibles de los datos.
b) Los modos de direccionamiento.
c) Las operaciones que se pueden efectuar.
d) El formato de las instrucciones (modo en el que se codifican).

La definición y las características de un juego de instrucciones vienen fijadas por:

- *Representaciones posibles de los datos.*
- *Modos de direccionamiento.*
- *Operaciones que se pueden efectuar.*
- *Formato de las instrucciones.*

5.3. MODOS DE DIRECCIONAMIENTO. TIPOS

Los *modos de direccionamiento* son las diversas formas que disponen las instrucciones para determinar el valor de un operando o la posición de un operando o una instrucción. La denominación de modo de direccionamiento proviene del hecho de que, normalmente, se especifica la dirección donde se encuentra el dato o la instrucción.

Un esquema representativo de los *modos de direccionamiento generales*, los definidos según estándares, se muestran en la tabla 5.1. Aunque es difícil que los repertorios de instrucciones de los computadores comerciales respondan exactamente a dicha clasificación, la definición de los mismos nos va a permitir comprender fácilmente los modos de direccionamiento que corresponden a cada uno de dichos computadores comerciales.

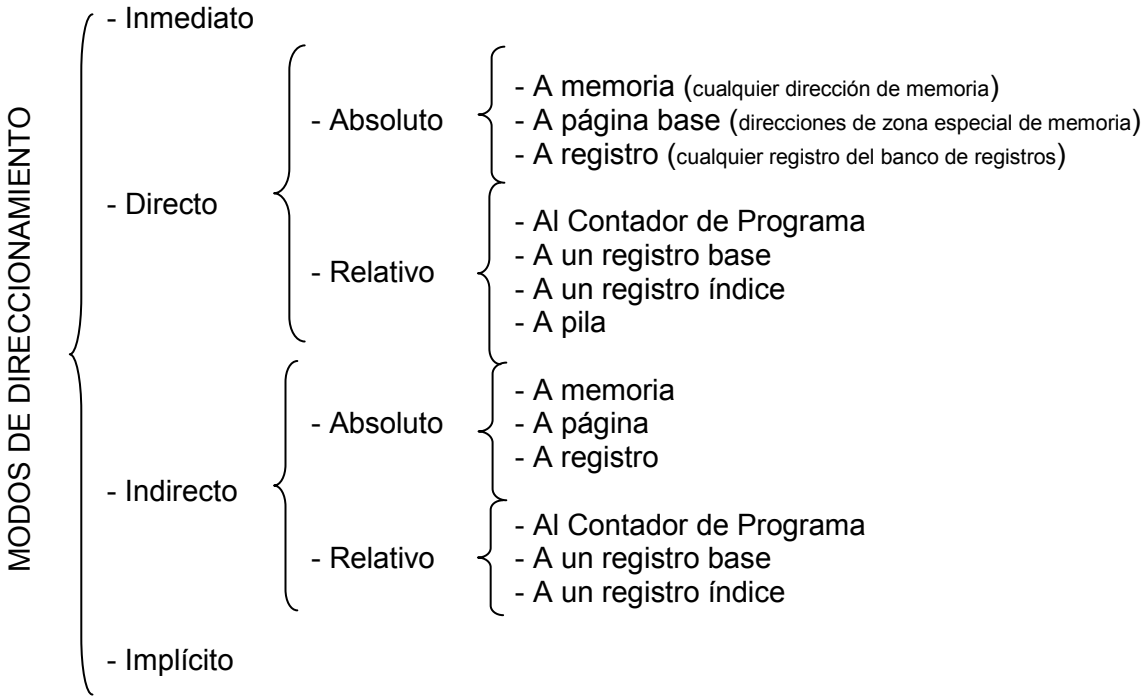

Tabla 5.1 *Esquema de los Modos de Direccionamiento.*

DIRECCIONAMIENTO INMEDIATO. El objeto, que en este caso es un operando, se encuentra contenido en la propia instrucción (Figura 5.2).

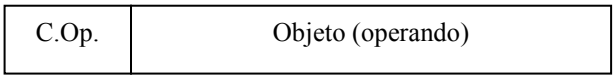

Figura 5.2 *Formato de una instrucción con modo de direccionamiento inmediato.*

DIRECCIONAMIENTO DIRECTO ABSOLUTO. Se llama *direccionamiento directo* al que expresa la **dirección real** del objeto, en contraposición con el indirecto. El *direccionamiento absoluto* expresa que la instrucción contiene la **dirección efectiva** del objeto, **sin compactar**. *Direccionamiento directo absoluto* significa que la instrucción contiene la dirección real, sin compactar, del objeto (Figura 5.3).

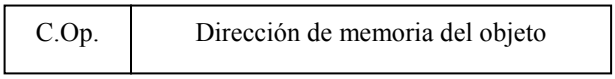

Figura 5.3 *Formato de una instrucción con modo de direccionamiento directo absoluto a memoria.*

El direccionamiento directo absoluto a memoria admite tres variantes:

- *DIRECTO ABSOLUTO A MEMORIA*. La dirección contenida en la instrucción es una *dirección completa* de la memoria principal (la dirección queda definida con el número de bits necesarios para seleccionar una posición del conjunto de posiciones que tiene la memoria principal).

- *DIRECTO ABSOLUTO A PÁGINA BASE*. La información contenida en la instrucción es una *dirección limitada* que hace referencia a una parte determinada del mapa de memoria (por ejemplo, el direccionamiento por página cero).

- *DIRECTO ABSOLUTO A REGISTRO*. La información contenida en la instrucción es el *identificador de un registro* en el que se halla almacenado el objeto.

DIRECCIONAMIENTO DIRECTO RELATIVO. La instrucción no contiene la dirección del objeto, sino un *desplazamiento* sobre una dirección marcada por un "puntero". La dirección efectiva se calcula sumando el desplazamiento al puntero de referencia situado en un registro. La principal ventaja de este modo de direccionamiento es que usa menos bits en la instrucción, aunque tiene el inconveniente de tenerse que realizar una operación de suma para encontrar dicha dirección efectiva. El desplazamiento puede ser tanto positivo como negativo. Las bases del código reentrante y reubicable, así como algunas técnicas de protección de memoria, residen en este tipo de direccionamiento (Figura 5.4).

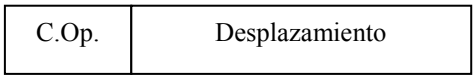

Figura 5.4 *Formato de una instrucción con modo de direccionamiento directo relativo.*

Según el registro puntero que se use y su tratamiento, se producen diversas variantes del direccionamiento relativo:

- *DIRECTO RELATIVO A PC*. Se usa el registro contador de programa como registro puntero. Es un direccionamiento muy adecuado para alcanzar instrucciones próximas a las que se está ejecutando (por ejemplo en las bifurcaciones que dan lugar a los bucles) (Figura 5.5).

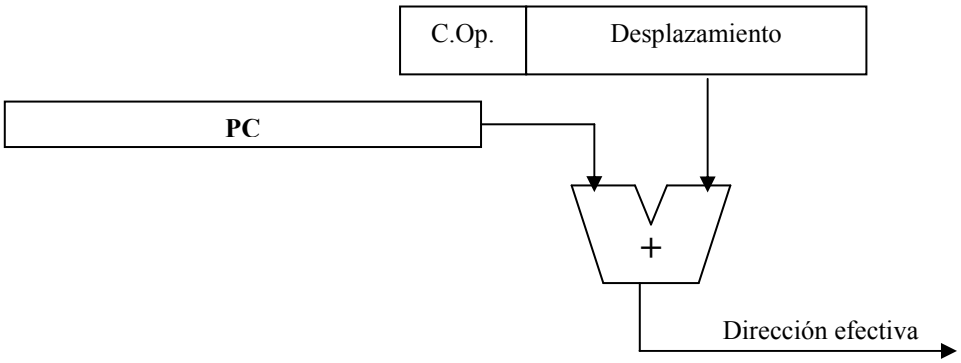

Figura 5.5 *Direccionamiento directo relativo a PC.*

- *DIRECTO RELATIVO A REGISTRO BASE*. Se usa como registro puntero un registro base, de los varios con los que puede contar la CPU. En este caso, la instrucción debe contener además del desplazamiento, el identificador del registro base seleccionado. Este direccionamiento se diferencia del relativo a registro índice en que el valor del registro base no suele modificarse y el del índice sí.

- *DIRECTO RELATIVO A REGISTRO ÍNDICE*. Se puede decir que es una variante del direccionamiento anterior, en el que el registro puntero es un registro índice, cuyo valor se modifica para ir recorriendo los elementos de una tabla o vector. Es muy frecuente que se realice de forma automática el autoincremento o autodecremento del registro índice (adaptándose siempre estos incrementos y decrementos a la longitud de los operandos). Las variantes posibles son:
    - Con *PRE-AUTOINCREMENTO*. Primero se produce el incremento del registro índice y posteriormente se produce la suma del desplazamiento.
    - Con *PRE-AUTODECREMENTO*. Primero se produce el decremento del registro índice y posteriormente se produce la suma del desplazamiento.
    - Con *POST-INCREMENTO* o *AUTOINCREMENTO*. Primero calcula la suma y luego se produce el incremento del registro índice.

- Con *POST-DECREMENTO* o *AUTODECREMENTO*. Primero calcula la suma y luego se produce el decremento del registro índice.

- *DIRECTO RELATIVO A PILA*. Se trata de un caso particular del direccionamiento relativo, muy utilizado. La CPU debe disponer de uno o varios *registros punterors de pila* (*SP*), los cuales contienen la dirección de memoria principal donde se encuentra la cabecera de la pila. La pila puede crecer según direcciones ascendentes o descendentes de memoria. La instrucciones con direccionamiento a pila son muy compactas puesto que existiendo únicamente un registro puntero de pila, la instrucción no requiere información alguna sobre dirección.

DIRECCIONAMIENTO INDIRECTO. En este caso se comienza con un direccionamiento directo (absoluto o relativo) que proporciona una dirección que contiene la dirección del objeto, y no el propio objeto. En consecuencia, para conocer el valor del objeto hay que realizar uno o varios accesos adicionales a la memoria principal. La *indirección* se puede añadir a cualquiera de los direccionamientos absolutos o relativos vistos anteriormente con el direccionamiento directo (Figura 5.6).

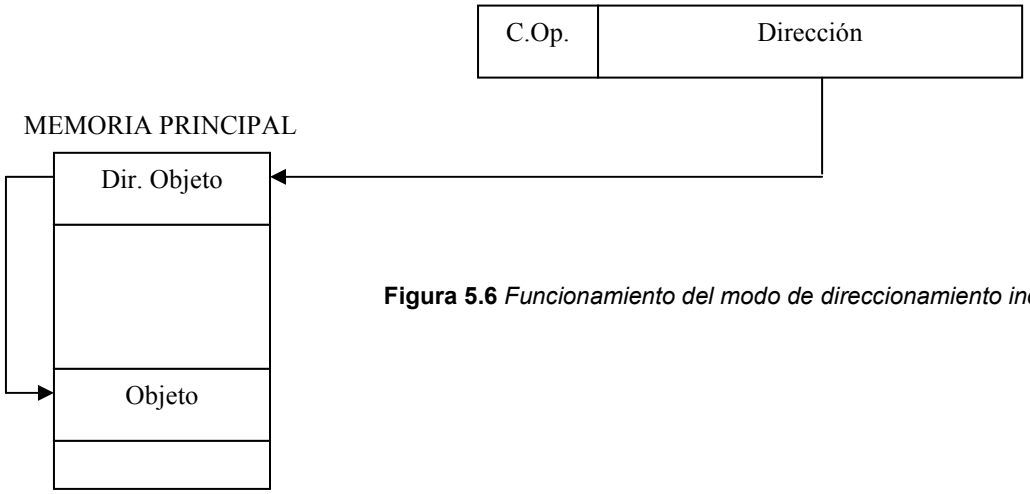

Figura 5.6 *Funcionamiento del modo de direccionamiento indirecto.*

DIRECCIONAMIENTO IMPLÍCITO. En este caso, la instrucción no contiene la información sobre la situación del objeto porque éste se encuentra en un lugar determinado (registro o posición de memoria). Es decir, en este caso, la situación del objeto está definida en el propio código de operación de la instrucción. Este direccionamiento presenta la ventaja de no ocupar espacio en la instrucción, aunque la hace muy restringida y poco flexible.

5.4. CARACTERÍSTICAS Y TIPOS DE INSTRUCCIONES

El juego de instrucciones de un computador debe ser *completo* y *eficaz*. Un juego de instrucciones *completo* permite calcular en un tiempo finito cualquier *tarea computable*. Además, el juego de instrucciones ha de ser *eficaz*, lo que equivale a permitir una velocidad de cálculo alta sin una excesiva complejidad de la unidad de control.

Se dice que una *tarea* es *computable* cuando puede calcularse en una *Máquina de Turing* con un número de pasos finito. La Máquina de Turing utiliza sólo cuatro instrucciones:

- Escribir
- Mover a la izquierda una posición y leer
- Mover a la derecha una posición y leer
- Parar

Definir el juego de instrucciones de un computador es uno de los puntos más críticos de su diseño.

El juego de instrucciones de un computador ha de ser:

- *Regular*. Es decir, no presentar casos especiales.
- *Ortogonal*. Cada operación debe poder realizarse con cualquier tipo de operando y con cualquier tipo de direccionamiento.

Tal como establece Fairclough, las instrucciones más utilizadas en los computadores pueden clasificarse en ocho grupos:

- **Movimiento o transferencia de datos**:
  Permiten repetir en el operando destino la información almacenada en el operando origen, sin que éste último se modifique. Ejemplos de estas instrucciones son: MOVE, STORE, LOAD, MOVE BLOCK, MOVE MULTIPLE, EXCHANGE, CLEAR, SET, PUSH, POP, ...

- **Ruptura de secuencia**:
  Alteran la secuencia normal de ejecución del programa. Hay varios tipos:
  - *Bifurcaciones o saltos condicionales*: Dan lugar a dos secuencias distintas según se cumpla o no la condición. Las condiciones de bifurcación se establecen sobre los biestables de estado (del registro de estado). Pueden

establecerse distintas condiciones {ZERO (Z), NOT ZERO (NZ), EQUAL (E), NOT EQUAL (NE), CARRY (C), NOT CARRY (NC), ... }

- *Bifurcaciones o saltos incondicionales*: En este caso no se evalúa ninguna condición; la ejecución de estas instrucciones implican la realización de la bifurcación.
- *Bifurcaciones con retorno*: Estas bifurcaciones salvan la dirección de la instrucción a partir de donde se encuentra la siguiente instrucción, es decir, el valor de PC (registro contador de programa). Con esto se puede retornar al punto donde se produjo la bifurcación y continuar en la siguiente dirección. Se usan en las llamadas a subrutina (CALL o BRANCH TO SUBROUTINE). La dirección de retorno se almacena en distintos lugares como pueden ser un registro especial, un registro general, en la propia subrutina, en la pila, ...

  Las denominaciones más frecuentes son BRANCH, CALL, RETURN, SKIP, RETURN WITH SKIP, …

- **Aritméticas**:
  Son las que se refieren a las operaciones que puede llevar a cabo la unidad aritmética, como son, entre otras: ADD, ADD WITH CARRY, SUBSTRACT, SUBSTRACT WITH BORROW, INCREMENT, DECREMENT, MULTIPLY, DIVIDE, EXTEND, NEGATE, ABSOLUTE.

- **Lógicas**:
  Realizan operaciones lógicas en todos los bits de los operandos, de forma independiente (AND, OR, NOT, XOR, ...).

- **De comparación**:
  La operación COMPARE consiste en restar o hacer la operación EXCLUSIVE-OR bit a bit, entre dos o más operandos. No se obtiene ni queda almacenado el resultado de la operación y sólo quedan afectados los biestables de estado. A este tipo de instrucciones le sigue una bifurcación condicional en base a los biestables que modifica. En muchos computadores existe la operación TEST, que consiste en una comparación con cero.

- **De bit**:
  Comprueban un bit del operando y su valor lo refleja en el biestable de estado ZERO. Ejemplos de estas instrucciones: BIT TEST, BIT SET, BIT CLEAR, ...

- **De desplazamiento**:
  Entre las más importantes se encuentran la instrucción SHIFT (desplazamiento lógico o aritmético de los bits del operando a derecha o izquierda) y ROTATE (operación de rotación a derecha o izquierda de los bits del operando; en la rotación puede incluirse el acarreo).

- **De entrada y salida, y diversas**:
  Son en realidad instrucciones de transferencia, con la peculiaridad de que el destino o el origen es un registro de un dispositivo de E/S (que está asociado a un periférico). Algunos ejemplos de instrucciones de este tipo son: INPUT, OUTPUT, TEST IO, CONTROL IO, ...

  A veces los juegos de instrucciones disponen de algunas especiales que no pueden incluirse en ninguno de los grupos descritos anteriormente, como son HALT, WAIT, CONVERT, NO OPERATION, ...

Las *instrucciones máquina* son cadenas de unos y ceros totalmente inexpresivas para el ser humano. Para su fácil identificación se usan *nemotécnicos*, que son un conjunto de letras representativas de la operación que realiza la instrucción recogidas de su nombre en inglés. Este tipo de representación es el que emplea el ***lenguaje ensamblador***.

5.5. FORMATOS DE INSTRUCCIONES. CARACTERÍSTICAS

La representación de la instrucción se denomina *formato*. El *formato* especifica el significado de cada uno de los bits que constituyen la instrucción, denominándose *longitud del formato* al número de bits que lo componen.

Para simplificar su decodificación, la instrucción se divide en una serie de *campos* (cadenas de bits contiguos) estando referido cada campo a un tipo de información específico.

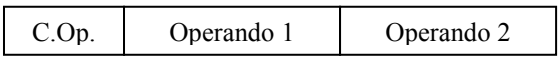

Figura 5.7 *Ejemplo de formato de instrucción*

Existen dos tipos básicos de *campos*:

- *Código de operación*: Indica la operación a realizar y el tratamiento que hay que dar al resto de informaciones de la instrucción.
- *Campo de dirección*: Especifica la dirección de un dato, resultado o instrucción a la que se bifurca.

CARACTERÍSTICAS DEL FORMATO DE INSTRUCCIÓN

De manera general, podemos decir que todos los formatos de instrucción de los computadores cumplen las siguientes características:

- Un computador tiene uno o varios formatos de instrucción. Las distintas instrucciones se encajan en estos formatos. Cuantos menos formatos se tengan, más sencilla será la unidad de control que deba interpretarlos.
- Los formatos son sistemáticos. Los campos del mismo tipo tienen la misma longitud y ocupan la misma posición dentro de las instrucciones. De esta forma se simplifica la decodificación y se minimizan los caminos internos del computador necesarios para mover la información de la instrucción.
- Para acortar el tamaño de los formatos se emplean ampliamente las técnicas de direccionamiento implícito.
- Los tamaños de los formatos encajan fácilmente en la palabra de la máquina.
- Cuando el computador tiene varios formatos, el código de operación diferencia entre ellos.

En los formatos, los tamaños de los campos que expresan direcciones, ya sea del banco de registros o de la memoria principal, deben corresponder a los espacios diseccionados (bancos de registros o mapas de memoria y/o E/S).

Ejemplo de diseño de formatos de instrucción del computador:

Supongamos un computador con palabras de 32 bits y 32 registros de 32 bits cada uno de ellos. Este computador tiene 64 operaciones y sus modos de direccionamiento son los siguientes:

- *Directo absoluto*
- *Indirecto absoluto*
- *Directo relativo:*
    - *A registro base*
    - *A contador de programa*

Se pide diseñar los dos formatos de instrucciones (1. Directo absoluto o indirecto absoluto y 2. Directo relativo) de dos operandos, y de modelo de ejecución Reg-Mp.

En primer lugar hay que decir que el *modelo de ejecución de un computador* expresa el lugar donde se encuentran los operandos que intervienen en una operación. El ejemplo hace referencia a un computador con modelo de ejecución Reg-Mp; esto nos indica que, en las operaciones diádicas, un operando se encuentra en

un registro y el otro se encuentra en una posición de memoria, y el resultado se almacenará en el registro que contenía uno de los operandos. Por lo tanto, para este tipo de instrucciones, el formato de la instrucción tendrá tres campos: código de operación, indicación de un registro del banco de registros e indicación de una dirección de memoria:

Cód. Op.	Reg.	Dir. Mp.

Figura 5.8 *Formato de la instrucción del ejemplo planteado.*

Vamos a determinar el número de bits que tiene cada uno de los campos del formato de la instrucción:

- **Campo Código de Operación**: Se especifica en el enunciado que el computador puede realizar hasta 64 *operaciones diferentes*. A continuación se dice también que tiene 4 modos de direccionamiento. Para calcular el número de bits que codifican el campo código de operación, consideraremos que todas las operaciones soportan todos los modos de direccionamiento Por tanto, el número "x" de bits definidos en el campo código de operación tiene que ser tal que $2^x \geq 64 \cdot 4 = 256 = 2^8$. Luego el menor "x" posible es x = 8.
- **Campo Registro**: Se dice en el enunciado que el número de registros que tiene el computador es 32. Por lo tanto, el campo de la instrucción que especifique uno de los operandos (que además suponemos que va a ser el destino del resultado en las operaciones diádicas) debe tener un número "y" de bits tal que $2^y \geq 32 = 2^5$. Por lo tanto, el menor "y" posible es y = 5.
- **Campo Dirección de Memoria Principal**: Este campo, dependiendo del modo de direccionamiento especificado, define una dirección de memoria (direccionamientos directo e indirecto absolutos) o un registro y un desplazamiento con respecto a ese registro (direccionamientos directos relativos). Como el enunciado especifica que el número total de bits de la palabra del computador es 32, si suponemos que una instrucción ocupa una palabra (podría definirse por más o menos de una palabra), el número "z" de bits que tendrá este campo será: z = 32 – (8+5) = 19. Estos 19 bits definen el campo de dirección de memoria principal: con los *modos de direccionamiento directo e indirecto absolutos*, definen una dirección de memoria principal; con el *modo de direccionamiento directo relativo a un registro base*, debe definir tanto el desplazamiento como el registro con respecto al cual está definido ese desplazamiento [el subcampo registro lo definirán 5 bits puesto que son 32 los registros que tiene el computador (se está suponiendo que son más de 16 los registros que se pueden considerar como bases, de esos 32) y el subcampo desplazamiento lo definirán 19 – 5 = 14 bits]; y, con el *modo de direccionamiento directo relativo al contador de programa*, como el registro contador de programa estamos suponiendo que es único y hemos supuesto que en el código de operación se especifica si es direccionamiento directo relativo al contador de programa, no es necesario explicitar en este campo el registro con respecto al que se define el desplazamiento, teniendo que definirse únicamente el desplazamiento y, por tanto, este desplazamiento para este modo de direccionamiento puede definirse con más bits [hasta 19 (hay que considerar un rango máximo)] o seguir considerando el mismo número de bits que con el resto de direccionamientos relativos (no habría que definir los 5 bits del subcampo registro). También se podría haber considerado el direccionamiento directo relativo al contador de programa como un caso no particular de direccionamiento relativo y, a la hora de calcular el número de códigos de operación diferentes considerar 64 operaciones con 3 modos de direccionamiento (64 · 3 = 192 códigos de operación), siguiéndose necesitando 8 bits para el código de operación, y teniendo que especificar en el subcampo registro una combinación de bits que hiciera referencia al registro contador de programa.

Por tanto, hay que considerar los dos formatos de instrucción representados en la figura 5.9.

Cód. Op. (8 bits)	Reg. (5 bits)	Dir. Mp. (19 bits)

1. Direccionamientos Directo e Indirecto absolutos.

Cód. Op. (8 bits)	Reg. (5 bits)	RB (5 bits)	Desplaz. (14 bits)

2. Direccionamientos Directo relativos a RB y a PC.

Figura 5.9 *Formato de las instrucciones del ejemplo planteado.*

5.6. RUPTURAS DE SECUENCIA NO PROGRAMADAS. INTERRUPCIONES Y EXCEPCIONES O CEPOS

La *unidad de control* estudiada hasta el momento ejecuta, de manera inamovible, las instrucciones sucesivas y contiguas de un programa. Solamente cuando se encuentra una bifurcación se rompe esta secuencia y se salta a ejecutar en otra zona de memoria.

Ahora bien, hay situaciones en las que puede interesar parar la ejecución de un programa, para pasar a ejecutar otro. Esto ocurre, por ejemplo, en los siguientes casos:

- Cuando aparece un error de ejecución, tal como desbordamiento del resultado en una operación aritmética.
- Cuando aparece un error de paridad en la lectura de la memoria.
- Al intentar ejecutar una instrucción con código de operación prohibido.
- Al intentar acceder a una zona de memoria protegida.
- Cuando un periférico requiera la atención de la unidad de control de proceso.

En estos casos, y otros muchos, debe producirse una *ruptura de secuencia no programada*, puesto que no se produce por instrucciones de bifurcación del programa en ejecución.

El mecanismo para producir estas rupturas de secuencia no programadas es similar al empleado para las bifurcaciones condicionales. En efecto, todas las condiciones que pueden producir esta ruptura de secuencia automática se reflejan en sendos biestables de estado. Además, *todas las instrucciones llevan al final de su cronograma una bifurcación condicional implícita* sobre el valor de estos biestables. De esta forma, cuando uno de ellos se activa, se corta la ejecución del programa en curso y se bifurca.

Cuando la causa de la bifurcación no programada pertenece al exterior de la CPU, se dice que se ha producido una *interrupción*. Por ejemplo, una unidad de disco interrumpe a la CPU cuando termina una operación de E/S. Cuando la causa de la bifurcación no programada es de origen interno a la CPU se habla de una *excepción* o *cepo* (*TRAP*) o *interrupción interna*.

Queda por determinar la forma en que se genera la dirección de bifurcación para estos casos:

- Una técnica consiste en *bifurcar a una posición prefijada o precableada*, lo cual es relativamente frecuente para los casos de los cepos o traps.

- Otra técnica empleada para las interrupciones externas, es que *el propio periférico*, que produce la interrupción, *proporciona la dirección de bifurcación* (*interrupción vectorizada*). Se analizará con detalle en el siguiente tema, Tema 6, relativo a la E/S.

En general, la dirección de bifurcación corresponde a la dirección donde se encuentra la primera instrucción de un programa del sistema operativo, que trata el cepo o la interrupción y, posteriormente, restituye el control al programa interrumpido.

5.7. ESTADO DEL COMPUTADOR

Un aspecto muy importante de las interrupciones internas y externas es que introducen una bifurcación, rompiendo la secuencia normal de ejecución de un programa. En general, una vez tratada la interrupción, deberá volverse al programa interrumpido, para que pueda seguir realizando su trabajo. Ahora bien, el computador tiene, además de la memoria principal, una serie de elementos de memoria que se van modificando de forma dinámica. Para que el programa interrumpido pueda seguir funcionando correctamente, estos elementos deberán volver a tomar el mismo valor que tenían cuando el programa fue interrumpido. Además, la zona de memoria principal asignada deberá haber sido respetada.

Llamaremos *estado del computador* al contenido de estos elementos de memoria internos, necesarios para que un programa pueda seguir funcionando correctamente. Dependiendo de lo complejo que sea el computador, su estado vendrá definido por más o menos elementos de memoria, pero, de forma general se pueden destacar los siguientes:

- *Contador de Programa*. Registro PC del ejemplo.

- *Registros Generales*. Banco de registros del ejemplo.

- *Registros Aritmétidos Auxiliares*. Del tipo del registro RA, siempre que su valor pueda tener sentido entre instrucciones sucesivas.

- *Biestables de Estado Aritmético*. Como los biestables de acarreo, desbordamiento, cero o negativo.

- *Biestables de Estado de E/S*. Como los biestables de inhibición de interrupción, de máscara o de nivel, que ya analizaremos en el tema correspondiente a la E/S.

- *Biestables o Registros de Modificación del Mapa de Memoria*. Tales como los biestables o los registros de ampliación del mapa de memoria vistos en el tema que trata sobre la memoria del computador.

- *Registros de Clave o de Protección de Memoria,* que limitan la zona de memoria principal accesible para un usuario.

- *Biestables de Control Residual o que Definen el Nivel de Ejecución del Computador*. Vistos en el tema correspondiente al estudio del repertorio de instrucciones y a los modos de direccionamiento, cuando se añadían unos bits adicionales para completar el código de operación y, además así se podían tener instrucciones privilegiadas en el sistema preparado para multiprogramación.

A título de ejemplo se muestra la Figura 5.31 el registro de estado del microprocesador comercial M68000 de Motorola.

Puede observarse que este microprocesador tiene su palabra de estado dividida en dos partes, una de sistema y otra de usuario. Ello se debe a que esta máquina tiene dos *niveles de operación*: *sistema* y *usuario*, definidos por el contenido del *bit S*. Cuando se está a *nivel de usuario*, los bits 8–15 del registro de estado son leíbles pero no se pueden modificar. Obsérvese que la *máscara de interrupción* solamente se puede modificar en el *nivel de sistema*.

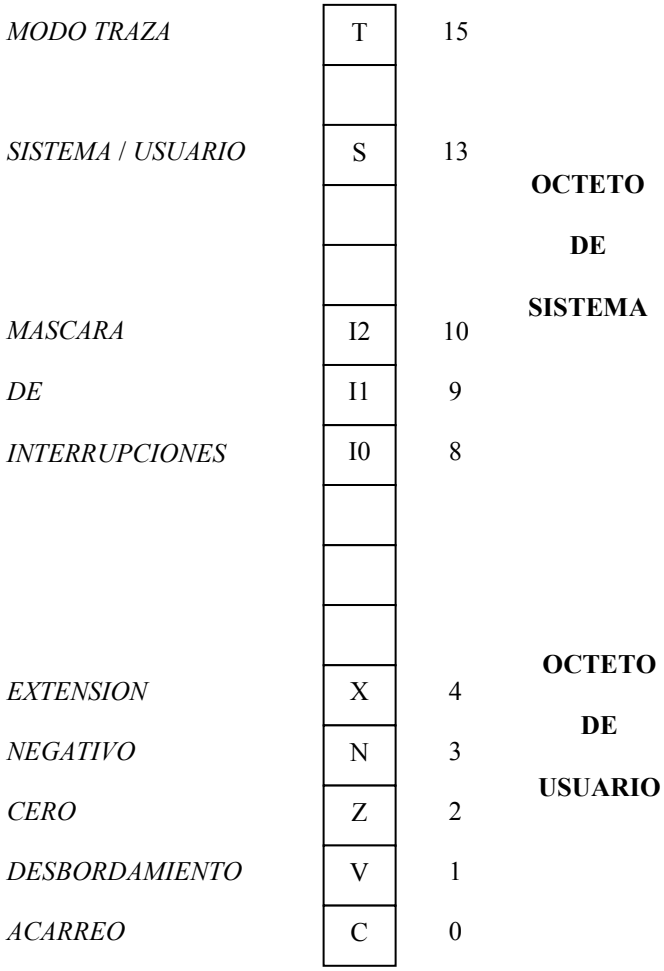

Figura 5.31. *Registro de estado del microprocesador 68000*

Evidentemente, siempre que se quiera poder volver a un punto determinado de un programa, como el retorno de las interrupciones, hay que salvar previamente el estado del computador, para volverlo a restituir y poder seguir con el programa, como si no hubiera pasado nada. La Figura 5.32 muestra cómo se va almacenando el estado, en el caso de emplearse una pila para ese almacenamiento; se supone que se está ejecutando el programa A y que ocurren los siguientes hechos.

- En el instante *T 1* aparece un Trap de desbordamiento.
- En el instante *T 2* finaliza el tratamiento del Trap y se retorna al programa A.
- En el instante *T 3* aparece una interrupción de un Terminal de usuario.
- En el instante *T 4* aparece una interrupción de la unidad de disco.

- En el instante *T 5* aparece una interrupción del reloj del sistema.

- En el instante *T 6* se termina de tratar la interrupción del reloj, por lo que se retorna al programa de tratamiento de la interrupción de disco.

- En el instante *T 7* se termina de tratar la interrupción de disco, por lo que se retorna al programa de tratamiento de la interrupción del Terminal de usuario.

- En *T 8* se termina el tratamiento de la interrupción del Terminal de usuario y se retorna al programa A.

- Etc.

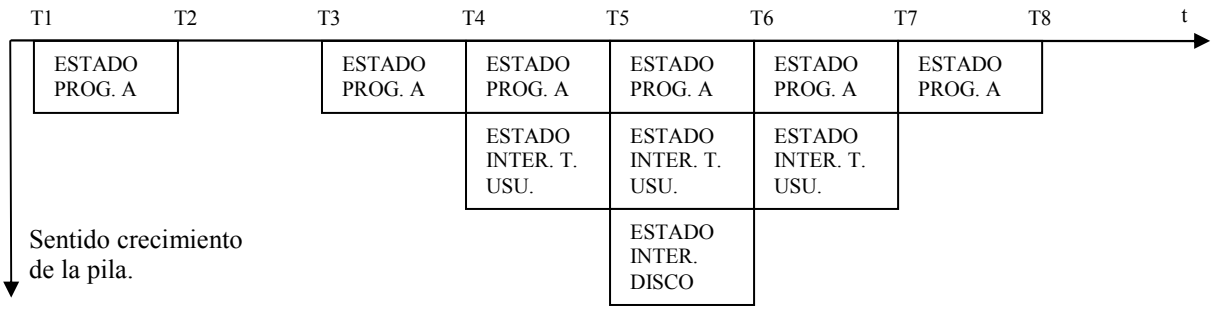

Figura 5.32. *Empleo de la pila para almacenamiento del estado*

5.8. ARRANQUE DEL COMPUTADOR

Un aspecto importante en el diseño de un computador es su *arranque*. En efecto, al estar la memoria principal, los registros y los biestables fabricados con semiconductores, cuando se activa la alimentación toman un estado desconocido de antemano. Esto quiere decir que el *contador de programa*, los *biestables de estado*, el *registro de instrucción*, los *registros generales* y los *registros internos* tomarán valores desconocidos, y que, por supuesto, la *memoria principal* no tendrá ningún programa concreto.

Sin embargo, una vez alimentada la máquina comenzaría con la secuencia de ejecución normal; tomaría el valor desconocido del registro PC, leería un valor desconocido de la memoria principal y trataría de ejecutarlo.

Para evitar esta situación, y que el computador arranque correctamente al encenderlo, se genera una *señal de inicialización* (*reset general*) que se emplea para dar

un valor determinado a los registros y biestables. La solución más corriente consiste en poner todos estos elementos a cero, y asignar el código de operación de valor cero a la instrucción de *no operación* (*NOP*). Así, de esta forma se garantiza que en el arranque el computador inicie la ejecución en una posición de memoria determinada (por ejemplo en la 0) y con unas condiciones internas previamente establecidas y aseguradas. En este caso, bastaría con diseñar la memoria principal de forma que en esa *zona de inicio* se emplease memoria que tuviese una información determinada y definida previamente (normalmente de tipo ROM), para que la ejecución comience de forma correcta.

En general, esta memoria ROM es de tamaño reducido y se limita a tener un pequeño *Programa Cargador*, llamado *BOOT–STRAP*, que tiene por objetivo traer a memoria principal desde un periférico de almacenamiento auxiliar (disco, diskette, cinta, etc.), el sistema operativo (o parte de él) que se va a quedar residente en memoria.