2

FORMATOS DE INSTRUCCIÓN

2.1. Introducción

Un programa consta de una secuencia de instrucciones, cada una de las cuales especifica una acción particular. Una parte de la instrucción, llamada **código de operación**, señala la acción que va a ejecutarse. Muchas instrucciones contienen los datos que se usan, o bien especifican donde están. Todo lo relativo a la localización de los datos sobre los que se realiza la operación (es decir, la dirección de los operandos) se denomina **direccionamiento** y será objeto de estudio en el capítulo siguiente. Uno de los objetivos de este capítulo será repartir los bits de la palabra de instrucción entre el código de operación y las especificaciones de operandos.

El propósito de una instrucción es especificar la acción que debe ser realizada y los operandos sobre los que se efectúa esta acción. Dentro de esos operandos pueden quedar incluidos tanto los datos de la operación como los lugares donde deben depositarse los resultados que la instrucción pueda producir. La operación y los operandos se describen mediante diferentes campos en la palabra de instrucción. Los campos de operando contienen las direcciones de los operandos o la información suficiente para de calcularlas. La mayoría de las instrucciones de un procesador tienen una de las siguientes formas:

$$x_n \leftarrow f(x_1, x_2, \ldots, x_n)$$

$$x_n \leftarrow f(x_1, x_2, \ldots, x_{n-1})$$

donde $x_1, x_2, \ldots, x_n$ son los operandos de la instrucción. La representación natural de esta instrucción de n operandos es el formato de la figura 2.1 que comprende un código de operación y la especificación de n operandos.

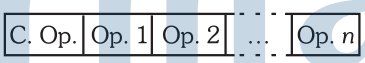


Fig. 2.1. Formato genérico de instrucción.

2.2. Criterios de diseño de los formatos de instrucción

Cuando se diseña un computador ha de elegirse el formato, o formatos, de instrucción considerando multitud de factores. Uno de los factores más importantes a tener en cuenta es la longitud de la palabra de instrucción: ésta puede o no ser igual para todas las instrucciones del computador y puede ser menor, igual o mayor que la longitud de palabra de memoria. Una cuestión a tener en cuenta es que, al menos en principio, las instrucciones cortas son mejores que las largas. Un programa compuesto por n instrucciones de 16 bits ocupa sólo la mitad de espacio que el mismo programa con instrucciones de 32 bits. Esto, en los computadores modernos, no suele suponer un problema; sin embargo, siempre será más rápido leer de memoria un programa que ocupe menos bits, y eso siempre es importante. Por otra parte, en algunas máquinas, es más rápido decodificar instrucciones más cortas. Otro factor a tener en cuenta es que es necesario dejar suficiente espacio en la instrucción para decodificar todas las operaciones deseadas. También conviene que la longitud de instrucción sea múltiplo o submúltiplo de la palabra de memoria.

Una característica importante a tener en cuenta para el diseño de los formatos de instrucción es la **ortogonalidad**. Se dice que dos aspectos de una arquitectura son **ortogonales** si *son independientes*. Esto, en nuestro caso concreto, es aplicable a los códigos de operación y a los modos de direccionamiento de los operandos. También se puede emplear este concepto combinando los aspectos anteriores con los tipos de datos. De esta forma, si todos estos asuntos son independientes, se codificarán en campos distintos y entonces se podrán combinar todas las operaciones con todos los modos de direccionamiento y con todos los tipos de datos. El concepto de ortogonalidad también se manifiesta en que los operandos se codifiquen siempre de la misma forma. Esta idea nos lleva a otro criterio importante a la hora de diseñar los formatos de instrucción de una máquina y es que el número de formatos diferentes en una misma máquina debe ser lo más bajo posible, ya que esto facilitará enormemente la decodificación y permitirá que ésta se pueda hacer totalmente por hardware, evitándose otros tipos de decodificación costosos y lentos.

2.3. Número de direcciones

Una de las formas de reducir el número de bits en el formato de instrucción de la figura 2.1 es especificar explícitamente sólo m de los n operandos, dejando implícita la especificación de los $n - m$ restantes. Se dice que un procesador es de m direcciones si m es el número máximo de operandos explícitos que tiene una instrucción aritmético-lógica genérica.

La elección del número de direcciones fue una de las cuestiones más discutidas en los tiempos de los primeros computadores. En la actualidad, el número de direcciones de las instrucciones depende de la organización interna del procesador. Según se vio en el capítulo anterior, los computadores pueden tener una de las siguientes organizaciones:

- Organización de un solo **acumulador**.
- Organización en **registros de uso general**.
- Organización de **pila**.

Normalmente, los computadores con organización de un solo acumulador tienen instrucciones de una sola dirección ya que el otro operando es el acumulador y el resultado queda en éste mismo. Ejemplo de este tipo de máquinas pueden ser la mayoría de los microprocesadores de 8 bits (como por ejemplo el Z-80).

Las máquinas con organización de registros generales pueden tener instrucciones de dos o tres direcciones, según se especifique o no la dirección del resultado. Ejemplos de este tipo de procesadores son el MC68000, el PDP-11 y el VAX (en este último el número de direcciones puede llegar hasta seis para operaciones complejas).

En las máquinas organizadas en pila no se requieren operandos ya que éstos se localizan en la cima de pila, igual que el resultado. Estas máquinas necesitan instrucciones adicionales para poner y quitar datos de la pila; estas instrucciones sí requieren campo de dirección (PUSH y POP o LOAD y STORE). Un ejemplo de este tipo de máquinas es el HP-3000.

2.4. Instrucciones con longitud variable

Algunas máquinas tienen instrucciones con un numero variable de bits. Esto implica que la primera parte de la palabra de instrucción debe tener información sobre el código de operación y sobre el número de operandos que le siguen. De esta forma, la longitud total de la instrucción depende del número de operandos. Sin embargo, en estas máquinas, el número de bits del código de operación suele ser fijo. Un ejemplo de este tipo de formatos lo constituye el VAX.

2.5. Instrucciones con código de operación de longitud variable

Una máquina con un formato de instrucción con código de operación de longitud fija de n bits puede tener 2^n instrucciones diferentes, que pueden ser monarias, binarias, etc. Sin embargo, puede ser posible mantener una longitud de instrucción fija y variar la longitud del código de operación según diferentes factores, tales como la frecuencia de uso de las instrucciones o el número de operandos de las mismas. En los apartados siguientes se abordará cada uno de estos esquemas.

2.5.1. Codificación de Huffman

Este método de codificación de las instrucciones se adopta aplicando un principio de diseño básico en Arquitectura de Computadores: *mejorar en lo posible los casos más frecuentes*. En nuestro caso, este principio se traduce en que las instrucciones más frecuentes tengan códigos de operación más cortos y así minimizar su tiempo de decodificación o la cantidad de memoria ocupada por el programa. En general este tipo de código se puede emplear siempre que se quiera disminuir el número de bits necesarios para una codificación (**compresión**).

En este punto hay que destacar que la frecuencia (o probabilidad) de utilización de las instrucciones se puede medir de dos formas:

- La **frecuencia estática**, que se refiere a las veces que cada instrucción se utiliza en los programas.
- La **frecuencia dinámica**, que se refiere a las veces que esas instrucciones son ejecutadas, esta frecuencia diferirá de la anterior, por ejemplo, si una instrucción se encuentra en un bucle, ya que se ejecutará muchas veces (frecuencia dinámica alta) pero aparecerá en el programa sólo una vez (frecuencia estática baja).

Según lo que se quiera optimizar en el diseño de un computador se puede utilizar una o la otra. Si se pretende minimizar la cantidad de memoria ocupada por el programa se utilizará la frecuencia estática, ya que la memoria ocupada por el programa depende del número de veces que aparezcan las instrucciones más largas. Sin embargo, si se pretende minimizar el tiempo de decodificación de las instrucciones se utilizará la frecuencia dinámica ya que el número de accesos a memoria depende de las veces que las instrucciones se ejecuten.

Dado un conjunto de n instrucciones con sus probabilidades de aparición (dinámicas o estáticas dependiendo de lo que se pretenda), el código de Huffman se construye atendiendo al siguiente método:

Paso 1: Las probabilidades asociadas a las instrucciones se ordenan en orden decreciente formando una lista L_1 de n elementos.

Paso 2: Los dos elementos menores de la lista obtenida en el primer paso se suman obteniendo así la lista L_2 de $n-1$ elementos. Se vuelve al paso anterior construyendo sucesivas listas L_i ordenadas.

Se repiten los pasos anteriores hasta la lista L_{n-1} que tendrá dos probabilidades.

Paso 3: Los códigos 0 y 1 (o viceversa) se asignan a cada elemento de la lista L_{n-1}.

Paso 4: A partir de este punto, si una probabilidad con un código c_j, asignado en el paso anterior, de la lista L_i se obtuvo sumando dos probabilidades P_k y P_{k-1} de la lista L_{i-1}, el código correspondiente a las instrucciones con estas probabilidades se formará concatenando el código c_j con un 1 y un 0 respectivamente (o viceversa).

Se repite el paso 4 hasta llegar a la lista L_1.

Como fácilmente puede desprenderse del método anterior, el código de Huffman se basa en la construcción de un árbol binario en que cada rama terminal corresponde a una instrucción. Cada ramificación del árbol supone la adición de un bit; debido a ello, las zonas más ramificadas del árbol representan a las instrucciones menos probables porque el código de Huffman precisamente pretende que las instrucciones menos probables tengan más bits en su código de operación.

Ejemplo 2.1: *Supongamos que se quiere diseñar, por este método, los códigos de operación de un computador cuyo juego de instrucciones, con sus probabilidades dinámicas de aparición son las que aparecen en la tabla 2.1.*

En primer lugar, ordenaremos las probabilidades en orden decreciente para formar la lista L_1 (ver tabla 2.2), después iremos sumando los dos últimos elementos de la lista,

Tabla 2.1. Juego de instrucciones y sus probabilidades dinámicas.

Instrucción	Probabilidad
A	0,50
B	0,27
C	0,05
D	0,10
E	0,08

Tabla 2.2. Ejemplo de códigos de operación según la codificación de Huffman.

Instrucción	L_1	L_2 (sin ordenar)	L_2 (ordenada)	L_3	L_4
A	0,50 **0**	0,50 **0**	0,50 **0**	0,50 **0**	0,50 **0**
B	0,27 **10**	0,27 **10**	0,27 **10**	0,27 **10**	0,50 **1**
D	0,10 **111**	0,10 **111**	0,13 **110**	0,23 **11**	
E	0,08 **1100**	0,13 **110**	0,10 **111**		
C	0,05 **1101**				

reordenando si es necesario, para formar las listas sucesivas hasta L_4. Una vez construidas estas listas asignamos arbitrariamente los códigos 0 y 1 a los dos elementos de la lista L_4. Luego, pasaremos a la lista L_3 y, a los elementos de esta lista que hayan tenido que ser sumados se les asignará el código de la suma añadiendo un 0 o un 1. Se continúa el proceso hasta llegar a la lista L_1 (es decir, la tabla inicial).

En este ejemplo se puede ver que el código de Huffman supone un ahorro del número de bits empleados. Si no se hubiera utilizado, los bits necesarios para la codificación hubieran sido 3. Sin embargo, aplicando el método de Huffman el número medio de bits empleados es:

$$\sum_{i=1}^{n} P_i l_i = 1,86$$

donde P_i representa la probabilidad de cada instrucción y l_i la longitud de su código de operación.

2.5.2. Códigos de operación con extensión

Dado lo engorroso que resulta a veces emplear el código de Huffman por las diferentes longitudes que genera, a veces se utiliza un procedimiento intermedio para la codificación de instrucciones que nos lleva a los códigos de operación con extensión. Este método consiste en asignar una longitud fija a las instrucciones más probables y reservar una serie de códigos para ampliar el código de operación sólo para las instrucciones menos probables. Así, para el ejemplo anterior, podrían quedar los códigos de operación que se ven en la tabla 2.3.

Tabla 2.3. Códigos de operación con extensión.

Instrucción	Probabilidad	C.op.
A	0,50	00
B	0,27	01
D	0,10	10
E	0,08	110
C	0,05	111

Fig. 2.2. Ejemplo de instrucción de tres direcciones.

Se puede observar que ahora hay sólo 2 longitudes diferentes, mientras que con la codificación de Huffman teníamos 4. El uso de pocas longitudes diferentes en el código de operación hace que la decodificación sea más sencilla y el árbol de decodificación menos profundo. La diferencia, a la hora de la decodificación, es que en el código de Huffman el árbol de decodificación es binario mientras que en los códigos de operación con extensión no lo es.

En este caso, la longitud media del código de operación será:

$$\sum_{i=1}^{n} P_i l_i = 2,13$$

Esta técnica también se usa en el caso de que tengamos una longitud de instrucción fija y queramos codificar operaciones con diferente número de operandos según la naturaleza de la operación. Si la longitud de instrucción es fija, el código de operación tendrá longitud variable según el número de operandos. Pudiera aplicarse la técnica anterior de reservar, en los códigos más cortos algún código para ampliar los bits empleados.

Ejemplo 2.2: *Supongamos una máquina de tres direcciones que debe tener instrucciones de cero, uno, dos o tres operandos con una longitud de instrucción de 16 bits, con campos de dirección de 4 bits. Un diseño de formato de instrucción posible puede ser el mostrado en la figura 2.2 Una forma de indicar que la operación no tiene los tres operandos es reservar algún código (o códigos) de operación para indicar que el campo de dirección 1 también corresponde al código de operación y el resto a operandos. Si se reserva el código $1111_{(2}$ para este fin podrá haber hasta 15 instrucciones de tres operandos (desde $0000_{(2}$ hasta $1110_{(2}$). Análogamente, se puede reservar algún código (o códigos) de 8 bits (por ejemplo $11111111_{(2}$) para indicar que el campo de dirección 2 también pertenece al código de operación, esto nos dará otras 15 instrucciones de dos operandos (desde $11110000_{(2}$ a $11111110_{(2}$), se puede seguir extendiendo el código hasta llegar a las instrucciones sin operandos. Si se precisaran más instrucciones de algún tipo se le reservarán más códigos del tipo anterior.*

Una variación de la idea anterior consiste en que alguna parte del código de operación indique la longitud de la instrucción. Por ejemplo los dos primeros bits pueden indicar si la instrucción tiene 1, 2, 3 o 4 bytes según su valor sea 00, 01, 10 o 11. Este enfoque difiere del anterior en que el código de operación no invade los campos de dirección sino la instrucción completa.

2.6. Ejemplos de formatos de instrucción en procesadores reales

En este apartado veremos como las técnicas anteriores se emplean en la práctica. Se verá que se utilizan la mayoría de las técnicas expuestas a excepción de la codificación de Huffman que complica mucho la decodificación; sin embargo, la idea de este tipo de código subyace en algunas de las máquinas.

2.6.1. PDP-11

El PDP-11 (DEC, 1983) es un ejemplo de procesador con longitud de instrucción fija y código de operación variable que emplea la técnica de código de operación con extensión. No obstante, es necesario mencionar que sólo es fija la longitud de la palabra principal de la instrucción, ya que algunos modos de direccionamiento precisan de palabras adicionales. Esta máquina también se caracteriza por su ortogonalidad ya que los operandos casi siempre se codifican de la misma forma y esto hace que la mayoría de las operaciones puedan combinarse con todos los modos de direccionamiento.

Todas las instrucciones del PDP-11 (figura 2.3) constan de una palabra (16 bits) a la que se puede añadir una o dos palabras de información adicional dependiendo del modo de direccionamiento de los operandos. Llamaremos I a la palabra de instrucción y numeraremos sus bits de 15 a 0. Para trabajar con esta máquina siempre se emplea el sistema octal de numeración ya que, como se verá a continuación, muchos de los campos de las instrucciones tienen 3 bits.

La decodificación comienza analizando los bits $I_{14:12}$ y pueden darse tres casos:

1. $I_{14:12} = 1$ a 6: **Instrucciones de dos operandos con direccionamiento general**. El formato de este grupo de instrucciones es el mostrado en la figura 2.3 (a). La codificación de operandos es siempre la misma: 3 bits para el modo de direccionamiento y otros 3 para especificar el registro sobre el que actúa. El bit 15 (S) indica si la operación se realiza sobre byte (1) o sobre palabra (0), excepto para el código 6, en cuyo caso $06_{(8}$ es el código de la suma (ADD) y $16_{(8}$ es el código de la resta (SUB). Ambas operan solamente con palabras.

2. $I_{14:12} = 7$: **Instrucciones de dos operandos con un operando en registro**. En este formato el código de operación se extiende a los bits $I_{11:9}$, como puede observarse en la figura 2.3 (b). A este grupo pertenece excepcionalmente la instrucción SOB, que sirve para programar bucles, en que los últimos 6 bits no corresponden a un operando en la forma modo-registro sino a un desplazamiento positivo de 6 bits que se resta al contador de programa para obtener la dirección de comienzo del bucle.

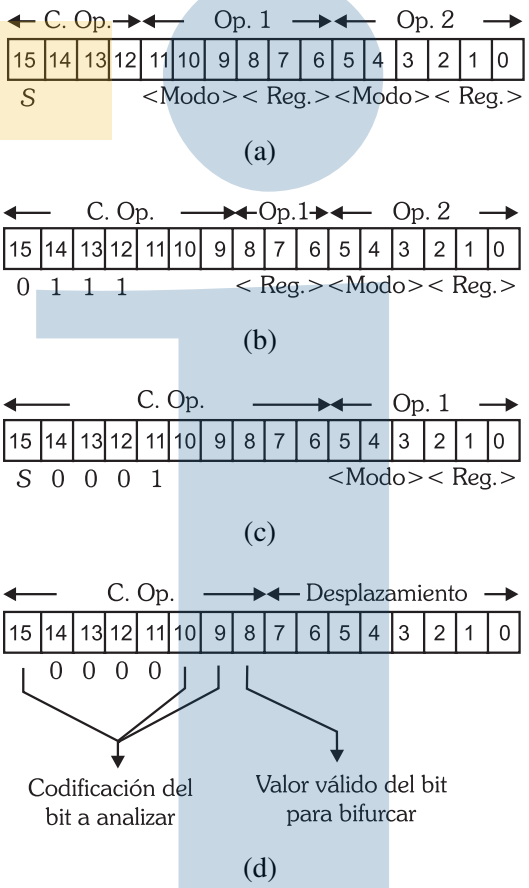


Fig. 2.3. Formatos de instrucción del PDP-11: (a) Instrucciones de dos operandos, (b) Instrucciones de dos operandos con uno de ellos en un registro, (c) Instrucciones de un operando y (d) Instrucciones de bifurcación.

3. $I_{14:12} = 0$: Este es un nuevo caso de extensión de código. Se mira bit I_{11} pudiéndose dar dos casos:

   $I_{11} = 1$: **Instrucciones de un operando**. El código de operación se extiende a los bits $I_{10:6}$, siendo el formato el mostrado en la figura 2.3 (c). El bit S sigue indicando el tamaño del operando. Existen algunas excepciones en este grupo, tal es el caso de la instrucción JSR (llamada a subrutina) que se diferencia del resto en que $I_{10:9} = 0$. Esta instrucción tiene dos operandos (Ver apéndice C).

   $I_{11} = 0$: En este caso el código se extiende 3 bits más y si $I_{15:8} = 0$ entonces, en general, se trata de **instrucciones sin operandos** entre las que destacan las instrucciones de manipulación de los bits de condición. Si no es así, es decir, si $I_{15:8} \neq 0$, se trata de las **instrucciones de bifurcación**. El formato de éstas se muestra en la figura 2.3 (d). En ella, el campo de codificación de bit se refiere al bit que determina la condición y el campo de valor válido se refiere al valor que tiene que tener el bit codificado anteriormente para que la condición se cumpla. Por ejemplo, en la instrucción BEQ la condición de bifurcación es $Z = 1$; por tanto el bit en cuestión es Z, que se codifica 001, y el valor de ese bit para bifurcar es 1, por tanto el código de BEQ

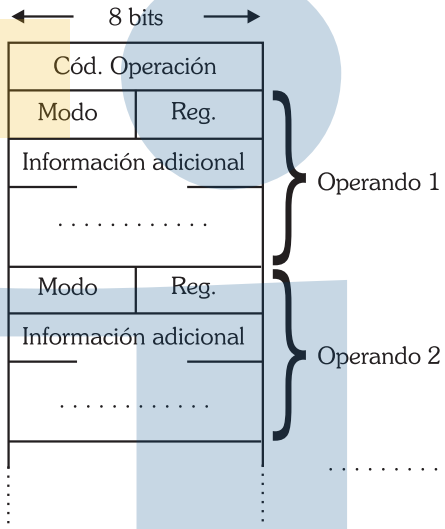


Fig. 2.4. Formato general de instrucción del VAX.

es 00000011. Se recordará que los bits de condición de esta máquina eran 4 (N, Z, V y C), sin embargo se reserva 3 bits para codificarlos (8 posibilidades). Esto es debido a que hay que codificar combinaciones de varios de estos bits para las comparaciones con y sin signo; también se reserva un código para la bifurcación incondicional (BR).

El lector al que las últimas líneas le parezcan demasiado complicadas puede hacer caso omiso de las mismas y volver a ellas después de haber leído el capítulo 4 en que se explicará el funcionamiento de las instrucciones de bifurcación. Para detalles sobre los códigos de los bits de condición, y en general sobre los códigos de operación del PDP-11, consúltese el apéndice C. Se recomienda analizar en profundidad el citado apéndice para comprender la filosofía de los formatos de instrucción de esta máquina.

2.6.2. VAX

Los formatos del VAX (DEC, 1986) se caracterizan por tener instrucciones de longitud variable mientras, sin embargo, su código de operación es de longitud fija. Los formatos del VAX son más simples que los del PDP-11 porque son aún más ortogonales.

La mayoría de las operaciones tienen un byte de código de operación y un byte más por cada operando al que puede agregarse información adicional que puede ocupar de uno a cinco bytes; esta información sirve para especificar desplazamientos, operandos inmediatos, etc. En las instrucciones aritméticas, los tres primeros bits del código de operación indican el tipo de los operandos (100: byte; 101: palabra; 110: doble palabra, etc.) y el último, el número de éstos (0, dos operandos; 1, tres operandos). Los operandos pueden oscilar entre cero y seis. El formato general de las instrucciones es el mostrado en la figura 2.4. El byte de especificación de operando tiene dos partes: 4 bits (*nibble*) para especificar el modo de direccionamiento y otros 4 para especificar uno de los 16 registros de propósito general. Todos los operandos van en el mismo formato y pueden combinarse todos los códigos de operación con todos los modos

de direccionamiento (ortogonalidad). Puede verse que los formatos están bastante inspirados en los del PDP-11, aunque hay importantes diferencias; la diferencia esencial radica en que el VAX tiene código de operación de longitud fija, aunque hay que mencionar que existen algunas instrucciones cuyo código de operación tiene dos bytes, esto se consigue reservando algunos valores en el primer byte (FDH, FEH y FFH) para extender el código de operación al siguiente. También es necesario mencionar que las instrucciones de bifurcación tienen un formato ligeramente diferente: además del byte de código de operación tienen uno o dos bytes más que indican un desplazamiento que se suma al contador de programa para obtener la dirección de bifurcación. Este operando no tiene por tanto, la codificación típica de los demás en la forma modo-registro.

2.6.3. MC68000 y derivados

Los formatos del MC68000 (Harman, 1989) se diseñaron teniendo en cuenta el mismo principio que inspiró el código de Huffman, es decir *mejorar en lo posible los casos más frecuentes*. En nuestro caso eso se traduce en conseguir que las instrucciones de uso más frecuente tengan códigos de operación más cortos. Las instrucciones de uso más frecuente son las de transferencia, por ello, en el MC68000 la instrucción con código más corto es MOVE cuyo código de operación tiene sólo 2 bits.

En el diseño de esta máquina también se nota la influencia del PDP-11 sobre todo en la codificación de los operandos que es muy parecida (modo-registro). Sin embargo, en el MC68000 se aplica la extensión del campo de modo al campo de registro si aquél vale $111_{(2}$; esto se aplica a los modos de direccionamiento que no operan sobre ningún registro. Estos pormenores se estudiarán detalladamente en el capítulo 3. Por otra parte, los formatos del MC68000 suponen un interesante contraste con los del PDP-11 en cuanto a la ortogonalidad. En el MC68000 se prefirió un mayor aprovechamiento de los bits de la palabra de instrucción a costa de prohibir muchas combinaciones de modos de direccionamiento. Se ha seguido una técnica similar a la empleada en el PDP-11 en cuanto a la información adicional necesaria en algunos modos de direccionamiento. Otra complicación respecto al PDP-11 es la adición de un nuevo tipo de dato (doble palabra) lo que supone un bit más para codificar el tamaño del operando.

La palabra básica de instrucción del MC68000, a la que llamaremos I, tiene 16 bits que numeraremos de 15 a 0. Comenzaremos analizando los bits $I_{15:14}$, pueden darse dos casos:

1. $I_{15:14} = 0$. Se pasan a analizar los dos siguientes bits ($I_{13:12}$). Si éstos no son 0 se trata de la instrucción MOVE, en cuyo formato (figura 2.5 (a)), los bits $I_{13:12}$ indican el tamaño de los operandos. Si $I_{13:12} = 0$ (código prohibido de tamaño), se analiza I_8 lo que nos lleva a dos situaciones:

   $I_8 = 1$: Son operaciones de bit individual en que el número del bit con que se pretende operar reside en un registro de datos; su formato se muestra en la figura 2.5 (b). En estas instrucciones uno de los modos está prohibido, lo que se aprovecha para codificar otras operaciones.

   $I_8 = 0$: Instrucciones aritméticas y lógicas con un operando inmediato (constante) que reside en la palabra siguiente; su formato puede verse en la figura 2.5 (c). Debe notarse que la codificación del tamaño de los operandos es distinta a la de MOVE.

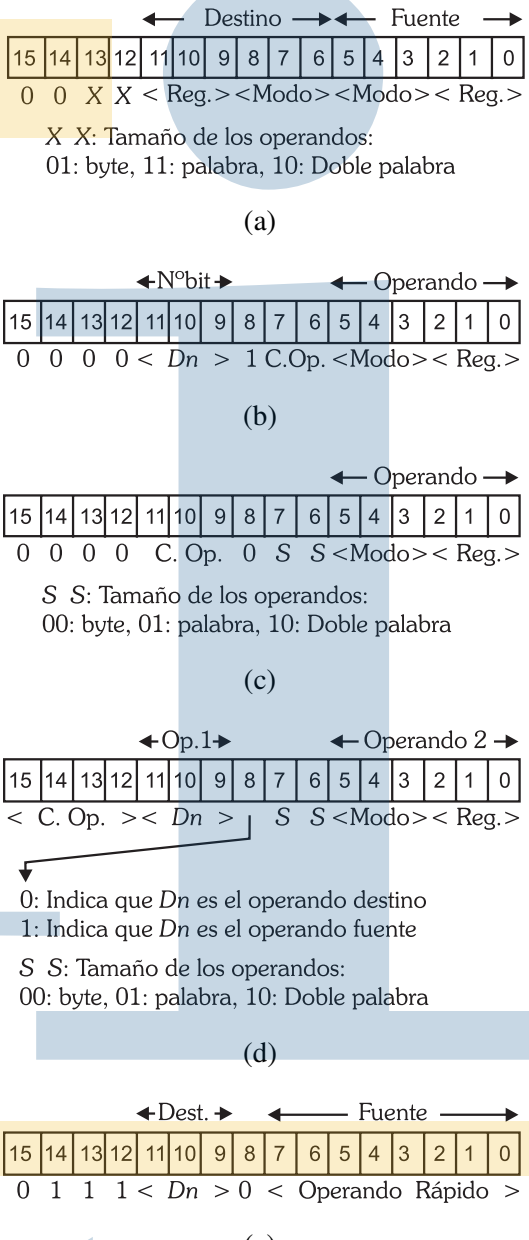
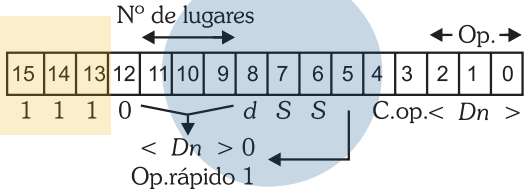


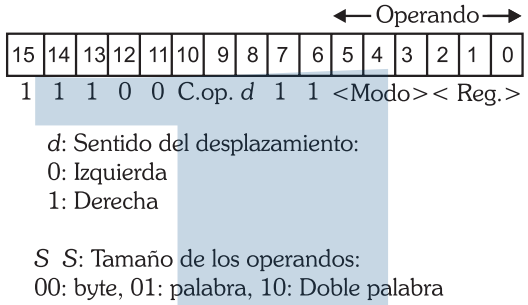
Fig. 2.5. Formatos de instrucción del MC68000: (a) Instrucción MOVE, (b) Operaciones de bit individual, (c) Operaciones aritmético-lógicas con un operando inmediato, (d) Operaciones aritmético-lógicas y (e) Instrucción MOVEQ (Continúa).

El código de tamaño de la figura 2.5 (c) es el usado en todas las instrucciones que precisan señalar el tipo del operando.

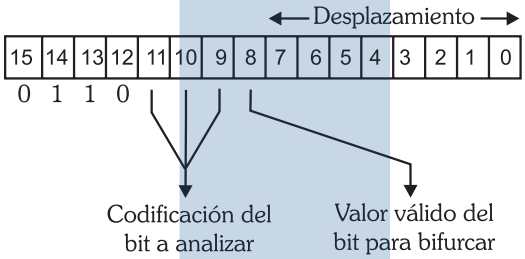
2. $I_{15:14} \neq 0$: En este caso el código se extiende por lo menos 2 bits más. Estudiemos todos los casos que pueden darse:

   $I_{15:12} =$ 8H, 9H, BH, CH ó DH: Operaciones aritméticas y lógicas (formato de la figura 2.5 (d) aunque existen variantes).

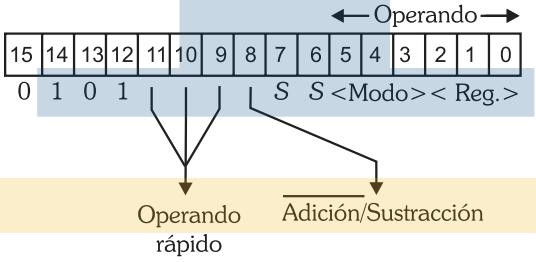




(f)

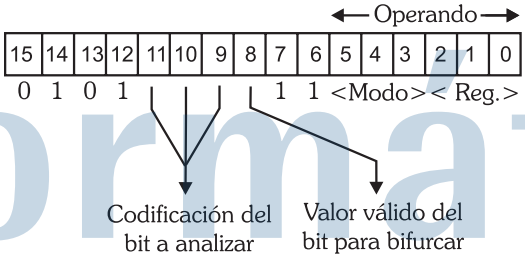


(g)



S S: Tamaño de los operandos:
00: byte, 01: palabra, 10: Doble palabra

(h)



(i)

Fig. 2.5. Formatos de instrucción del MC68000: (f) Instrucciones de desplazamiento y rotación, (g) Instrucciones de bifurcación, (h) Instrucciones de adición y sustracción rápidas, (i) Instrucción Scc (Conclusión).

$I_{15:12}$ = 7H: Se trata de la instrucción MOVEQ (formato de la figura 2.5 (e)). MOVEQ forma parte de las instrucciones con **operandos inmediatos rápidos** que son constantes pequeñas que se codifican en la palabra básica de instrucción.

$I_{15:12}$ = EH: Operaciones de desplazamiento y rotación. Su formato es uno de los mostrados en la figura 2.5 (f) que se diferencian en el valor de los bits $I_{7:6}$ que en primer caso indican el tamaño y en otro toman el valor $11_{(2}$ (tamaño prohibido). En el primer formato el número de desplazamientos viene dado por un operando inmediato rápido de 3 bits o por el contenido de un registro de datos (bits $I_{11:9}$) y el operando siempre reside en un registro de datos. Las instrucciones correspondientes al segundo formato sólo desplazan un lugar pero tienen la ventaja de que el operando tiene la forma general modo-registro.

$I_{15:12}$ = 6H: Instrucciones de bifurcación con el formato de la figura 2.5 (g). En esta figura, el campo de codificación de bit se refiere al bit que determina la condición y el campo de valor válido se refiere al valor que tiene que tener ese bit para que la condición se cumpla. La codificación de los bits puede verse en el apéndice E. El nemónico de estas instrucciones es Bcc donde cc depende de la condición. Si el desplazamiento es nulo significa que el desplazamiento tiene 16 bits que se codifican en la palabra siguiente. En las instrucciones de bifurcación se nota mucho la influencia del PDP-11.

$I_{15:12}$ = 5H: El formato de estas instrucciones varía en función de los bits $I_{7:6}$: si no valen $11_{(2}$ se trata de ADDQ y SUBQ (suma y resta con operando rápido) cuyo formato se muestra en la figura 2.5 (h). Si los bits que indican el tamaño ($I_{7:6}$) valen $11_{(2}$ (valor prohibido) tenemos la instrucción Scc (formato de la figura 2.5 (i)). Esta instrucción pone 1's en el operando si la condición se cumple y 0's en caso contrario. La forma de codificar la condición es igual que en las instrucciones de bifurcación. Si en esta instrucción se utiliza el direccionamiento 001 (prohibido) se tiene la instrucción DBcc en que el campo de registro de datos; esta instrucción sirve para programar bucles.

$I_{15:12}$ = 4H: Dependiendo del bit I_8 se puede tratar de las instrucciones aritméticas y lógicas de un operando (con el formato de la figura 2.5 (c)), de instrucciones sin operandos o de instrucciones diversas con el formato de la figura 2.5 (b).

2.6.4. IBM-360/370

Los formatos de esta serie de computadores se caracterizan por tener código de operación de longitud fija y longitud total de instrucción variable. No es ortogonal y los modos de direccionamiento se codifican en el código de operación. Tiene cinco formatos de instrucción denominados RR, RX, RS, SI y SS, en función de los modos de direccionamiento utilizados en los operandos. Todos los formatos tienen 8 bits de código de operación, de los cuales los dos primeros indican el tipo de formato:

- Formato RR (registro-registro): ambos operandos radican en registros de propósito general. Este formato es el mostrado en la figura 2.6 (a).

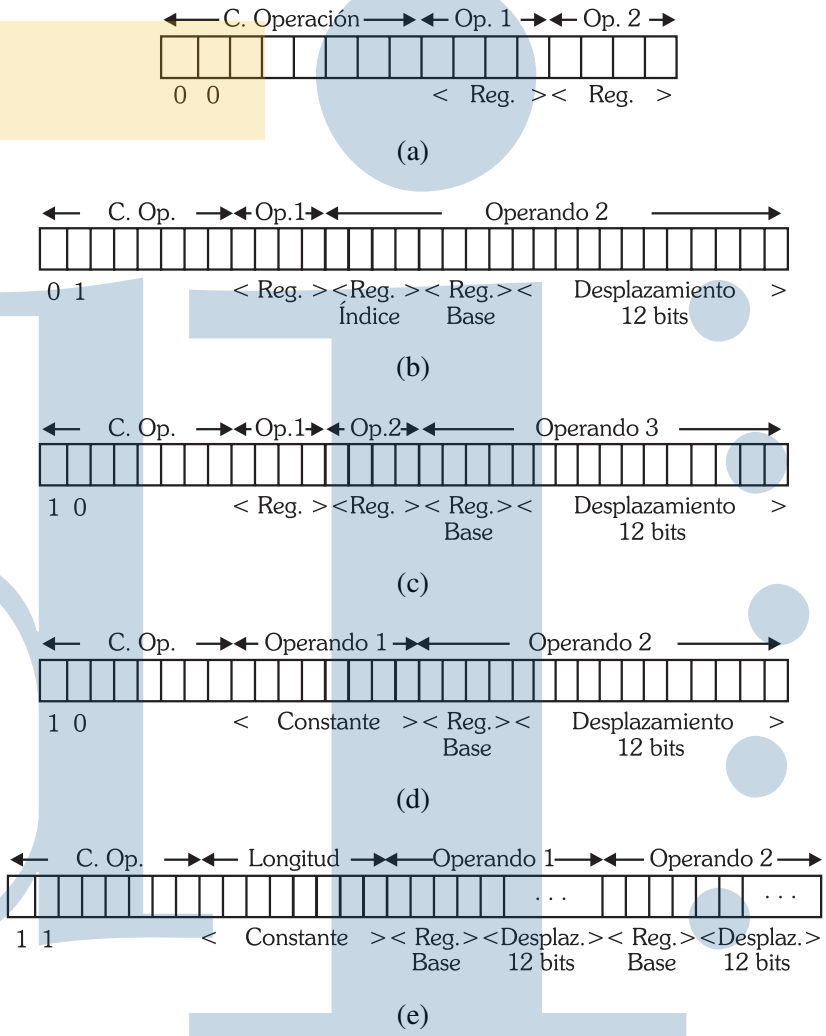


Fig. 2.6. Formatos de instrucción de los computadores IBM de las series 360 y 370: (a) Formato RR, (b) Formato RX, (c) Formato RS, (d) Formato SI y (e) Formato SS.

- Formato RX (registro-índice): un operando reside en un registro y el otro en memoria especificándose con direccionamiento indexado. El formato RX se muestra en la figura 2.6 (b).
- Formato RS (registro-memoria): dos operandos residen en registros y un tercero en memoria. Este formato se muestra en la figura 2.6 (c).
- Formato SI (memoria-inmediato): un operando está en memoria mientras que el otro es una constante (direccionamiento inmediato). Este formato se muestra en la figura 2.6 (d). Este formato se especifica con el mismo código que el anterior (10) porque es incompatible con el en el mismo código de operación.
- Formato SS (memoria-memoria): ambos operandos están en memoria. Este tipo de operaciones se emplea para transferir bloques de datos entre lugares diferentes de la memoria, por eso hay que especificar la longitud del bloque que tiene que ser constante. Este formato se muestra en la figura 2.6 (e) y consta de 48 bits.

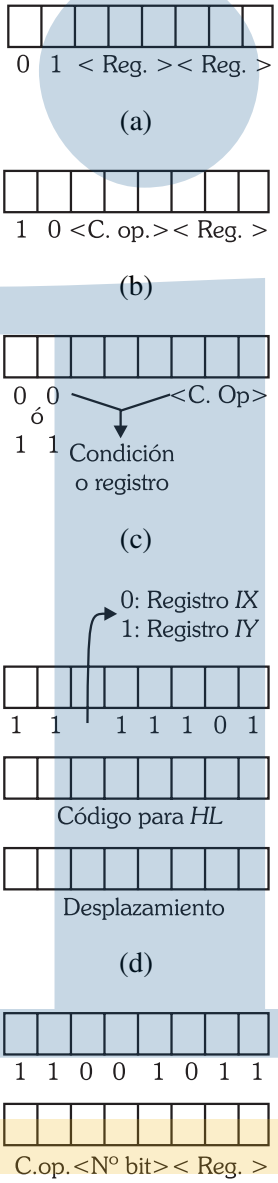


Fig. 2.7. Formatos de instrucción del microprocesador Z-80: (a) Instrucción LD, (b) Instrucciones aritméticas y lógicas de 8 bits, (c) Instrucciones con $I_{7:6} = 0$ o $I_{7:6} = 3$, (d) Instrucciones que manejan los registros índice y (e) Instrucciones de bit individual.

2.6.5. Z-80

La característica fundamental del diseño de los formatos de instrucción de Z-80 fue la decisión de hacerlos compatibles a nivel de códigos de operación con los del microprocesador 8080 de Intel (Zaks, 1982); esto complicó algo los formatos que hubieran sido más sencillos si se hubiera decidido diseñar de nuevo todos los formatos. En los formatos que se verán a continuación se nota claramente qué instrucciones pertenecían al 8080 y cuáles son de nueva incorporación al Z-80.

Las instrucciones pueden tener de uno a cuatro bytes, de los cuales uno o los dos primeros corresponden a los códigos de operación y el resto a desplazamientos, operandos inmediatos, etc. Llamaremos I al byte principal del código de operación y numeraremos sus bits de 7 a 0. Para decodificar la instrucción se comienza analizando los 2 primeros bits de mayor orden ($I_{7:6}$). Pueden darse los siguientes casos:

1. $I_{7:6} = 1$: Se trata de las instrucciones de carga entre registros (LD) de 8 bits cuyo formato se muestra en la figura 2.7 (a).

2. $I_{7:6} = 2$: Son instrucciones aritméticas y lógicas de 8 bits con el formato de la figura 2.7 (b). Como puede verse el código se extiende tres bits más.

3. $I_{7:6} = 0$: En este caso el código también se extiende pero a los 3 últimos bits. El formato general de estas instrucciones se muestra en la figura 2.7 (c) y son transferencias entre la memoria y los registros, cargas de constantes, instrucciones aritméticas o lógicas de 16 bits y saltos relativos.

4. $I_{7:6} = 3$: A este grupo pertenece un numeroso conjunto de instrucciones que pueden estar en uno de los dos siguientes casos:

   a) Instrucciones que pertenecían al juego del Intel 8080: Estas instrucciones tienen código de operación diferente de DD, FD, ED o CB (H) y tienen el formato ya mostrado en la figura 2.7 (c). Entre ellas están las instrucciones de salto absoluto, llamada y retorno de subrutina, manejo de pila (PUSH y POP) y algunas más.

   b) Instrucciones de nueva incorporación al juego del Z-80: Se caracterizan porque el valor de su primer byte es uno de los siguientes: DD, FD, ED y CB (Hex.) y extienden su código de operación al byte siguiente. En los dos primeros casos (DD y FD) se trata de instrucciones existentes en el 8080 pero que incorporan direccionamiento indexado o manejan los registros índice IX e IY (estos registros no existían en el 8080). El formato de esas instrucciones es el mostrado en la figura 2.7 (d). El código DD corresponde a operaciones sobre IX y el FD a operaciones sobre IY. El byte siguiente corresponde al código de la instrucción que se trate como si se aplicara al registro HL. Si se trata de operaciones con direccionamiento indexado hay que añadir un byte para especificar el desplazamiento. El código ED corresponde a algunas instrucciones de nueva incorporación en el Z-80 como las operaciones sobre bloques y el código CB corresponde a nuevas instrucciones de desplazamiento y rotación o a las instrucciones de bit individual cuyo formato se muestra en la figura 2.7 (e). Estas instrucciones también admiten direccionamiento indexado y entonces su código de operación tiene 3 bytes.

Se sugiere la lectura y análisis del apéndice F para completar de forma más detallada el estudio de los formatos de instrucción de este microprocesador.

2.6.6. i-8086 y derivados

En el i-8086, los formatos de instrucción son muy irregulares (Brey, 2000). Este procesador es del tipo registro-memoria, por lo que no puede trabajar, en general, con dos operandos en

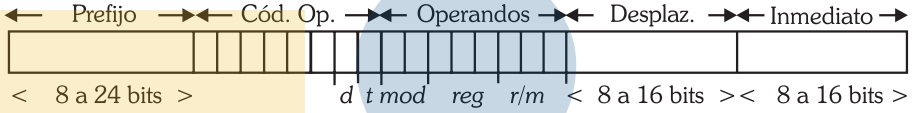


Fig. 2.8. Formato general de las instrucciones del microprocesador 8086.

memoria: normalmente si uno de los operandos está en memoria el otro radica en un registro. Si el computador fuera ortogonal no se tendría este inconveniente porque todos los operandos se codificarían de la misma forma. Esto hace que la mayoría de las instrucciones tengan que tener un bit que indique quién es el operando fuente: el registro o el operando en memoria.

El formato general de las instrucciones es el mostrado en la figura 2.8. En este formato, el único campo que aparece siempre es el de código de operación que tiene 8 bits. Los demás campos son opcionales según el tipo de instrucción y los modos de direccionamiento empleados. Dentro del código de operación hay instrucciones en que el último bit (t) indica el tamaño del operando (0: byte, 1: palabra) y el anterior (d) indica quién es el operando destino (0: el operando destino se especifica mediante los campos mod y r/m y puede residir en memoria o en un registro, 1: el operando destino es un registro que se especifica mediante el campo reg). El segundo campo indica la localización de ambos operandos, uno está especificado por mod y r/m y el otro es un registro que se especifica mediante el campo reg, como se vio antes el que determina cuál de los operandos es fuente o destino es el bit d. Los campos de desplazamiento e inmediato sólo se usan cuando aparecen operandos que utilizan determinados direccionamientos. El prefijo, que, si existe, puede tener de 1 a 3 bytes sirve para modificar algunas circunstancias de la instrucción (repeticiones, indicaciones sobre el registro de segmento a utilizar por el operando en memoria, etc.).

En el 80386 los formatos de instrucción son más complicados; las diferencias radican en la mayor longitud de los campos (el código de operación puede tener 2 bytes, y los campos de desplazamiento e inmediato hasta 32), y en la adición de un nuevo campo opcional de 1 byte antes del campo de desplazamiento. Este byte se usa en algunos modos de direccionamiento nuevos y se denomina SIB (*scale, index, base*: escala, índice, base)

2.6.7. Arquitectura Alpha

Esta máquina, como buen ejemplo de las máquinas RISC, tiene formatos de instrucción muy sencillos y regulares. Su regularidad comienza en el hecho de que todas las instrucciones y códigos de operación tienen longitud fija (32 y 6 bits, respectivamente). Nunca se añaden informaciones adicionales a los 32 bits de instrucción, como se ha visto que hacen otros procesadores. Sin embargo, hay instrucciones que tienen códigos de operación secundarios. Llamaremos I a la palabra de instrucción y numeraremos sus bits de derecha a izquierda.

Esta máquina tiene varios formatos de instrucción (figura 2.9) (Sites, 1992) que se diferencian por el valor del código de operación principal (bits $I_{31:26}$). Existen los siguientes formatos:

1. Instrucciones sobre **memoria**: es el formato de las instrucciones LOAD y STORE (carga y almacenamiento) para diferentes tipos de datos. Este formato se muestra en la figura 2.9 (a). Existe una variante de este formato en que el campo de desplazamiento se

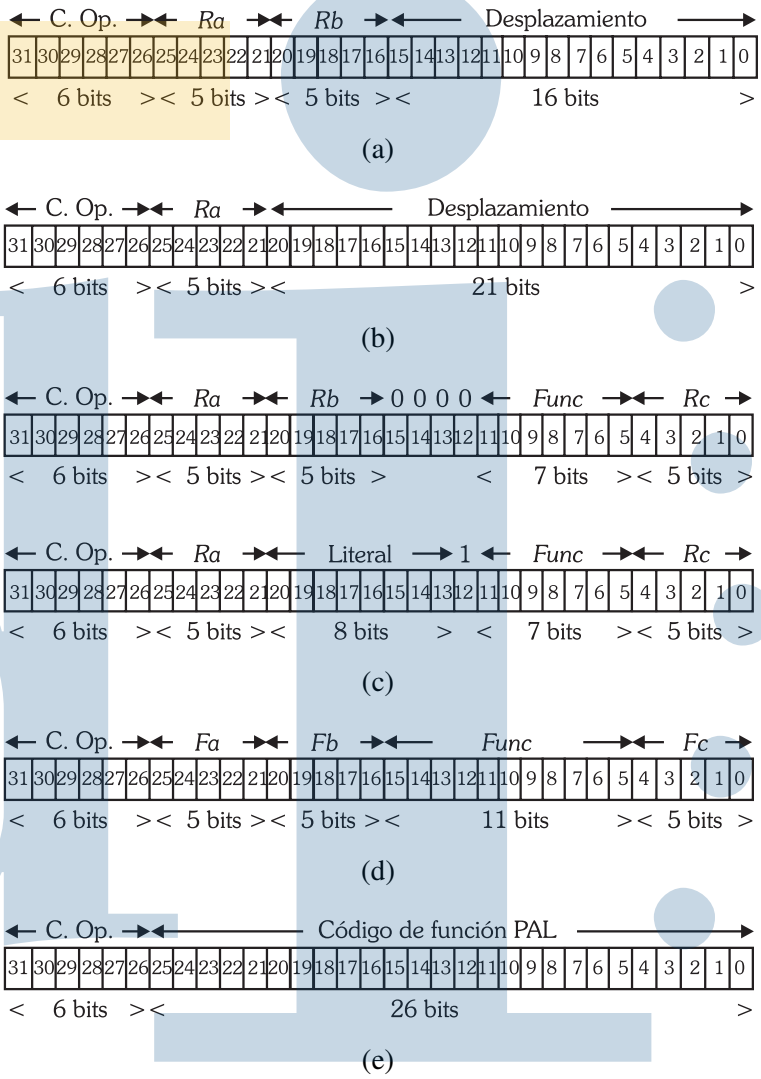


Fig. 2.9. Formatos de instrucción de la arquitectura Alpha: (a) Instrucciones sobre memoria, (b) Instrucciones de bifurcación, (c) Instrucciones enteras, (d) Instrucciones de punto flotante y (e) Instrucciones PAL.

utiliza como **código de función** (código de operación secundario) y otra para instrucciones de salto en que los dos bits más altos del desplazamiento se usan para realizar una predicción de la dirección de salto. Ambos casos se caracterizan por valores específicos en el código de operación principal.

2. Instrucciones de **bifurcación**: este formato se muestra en la figura 2.9 (b) donde el campo de desplazamiento representa la diferencia entre la dirección de bifurcación y el contenido del contador de programa en complemento a 2 (medido en dobles palabras).

3. Instrucciones **enteras**: estas instrucciones tienen dos tipos de formatos que se muestran en la figura 2.9 (c). Ambos difieren en el valor del bit I_{12}: en el primer caso los dos operandos se encuentran en registros (Ra y Rb) y el destino es el registro Rc; en el segundo, uno de los operandos es un literal, es decir, constante, que reside en los bits $I_{20:13}$.

4. Instrucciones de **punto flotante**: su formato se muestra en la figura 2.9 (d). Los operandos se hallan en los registros de punto flotante Fa y Fb y el resultado queda en Fc. Estas instrucciones trabajan con varios formatos de punto flotante diferentes.

5. Instrucciones PAL (Privileged Architecture Library): estas instrucciones sirven para hacer llamadas de bajo nivel dependientes del sistema operativo. Su formato se muestra en la figura 2.9 (e).

2.6.8. Arquitectura SPARC

Este ejemplo (Weaver & Germond, 1994) también es un caso típico de arquitectura RISC. Una característica importante de sus formatos es que su palabra de instrucción tiene una longitud completamente fija de 32 bits, incluyendo código de operación y especificaciones de operandos. Nunca se añade información adicional a esos 32 bits.

En un primer análisis de los formatos de instrucción de esta arquitectura (figura 2.10), puede parecer que su variedad es grande. En realidad hay muy pocos formatos principales con algunas variaciones. El formato principal queda determinado por los dos bits de orden más alto de la palabra de instrucción, dependiendo de esos dos bits, el código puede extenderse a los bits 24:22 o a los bits 24:19 y en algún caso excepcional también a los bits 13:5. Por otra parte, si se observan conjuntamente todos los formatos se verá que las mismas informaciones se codifican casi siempre en los mismos bits, aunque sea en formatos distintos, con muy pocas excepciones. Esto es muy típico de los procesadores RISC. Otro detalle que se puede observar, es que uno solo de los formatos (el correspondiente a la figura 2.10 (d)) incluye un porcentaje muy amplio de instrucciones de la máquina, que además son las que se usan con más frecuencia. Por el contrario, el variado conjunto de formatos de la figura 2.10 (g) corresponde a instrucciones de uso escaso.

Bibliografía y referencias

BREY, B.B. 2000. *Intel Microprocessors 8086/8088, 80186/80188, 80286, 80386, 80486, Pentium, and Pentium Pro Processor Architecture, Programming, and Interfacing*. 5 edn. Pearson Education. Existe traducción al castellano: Los microprocesadores Intel : arquitectura, programación e interfaces de los procesadores 8086/8088, 80186/80188, 80286, 80386 y 80486 Pentium, Pentium Pro y Pentium II, Pearson Education, 2001.

CRAWFORD, J, & GELSINGER, P. 1987. *Programming the 80386 (Featuring 80386/80387)*. Sybex. Existe traducción al castellano: Programación del 80386/387, Anaya Multimedia, 1991.

DEC. 1983. *PDP-11 Architecture Handbook*. Digital Equipment Corporation.

DEC. 1986. *VAX Architecture Handbook*. Digital Equipment Corporation.

HARMAN, T.L. 1989. *The Motorola MC68020 and MC68030 Microprocessors, Assembly Language, Interfacing and Design*. Vol. II. Prentice-Hall International.

RODRÍGUEZ ROSELLÓ, M.A. 1987. *8088-8086/8087. Programación en ensamblador en entorno MS-DOS*. Anaya Multimedia.

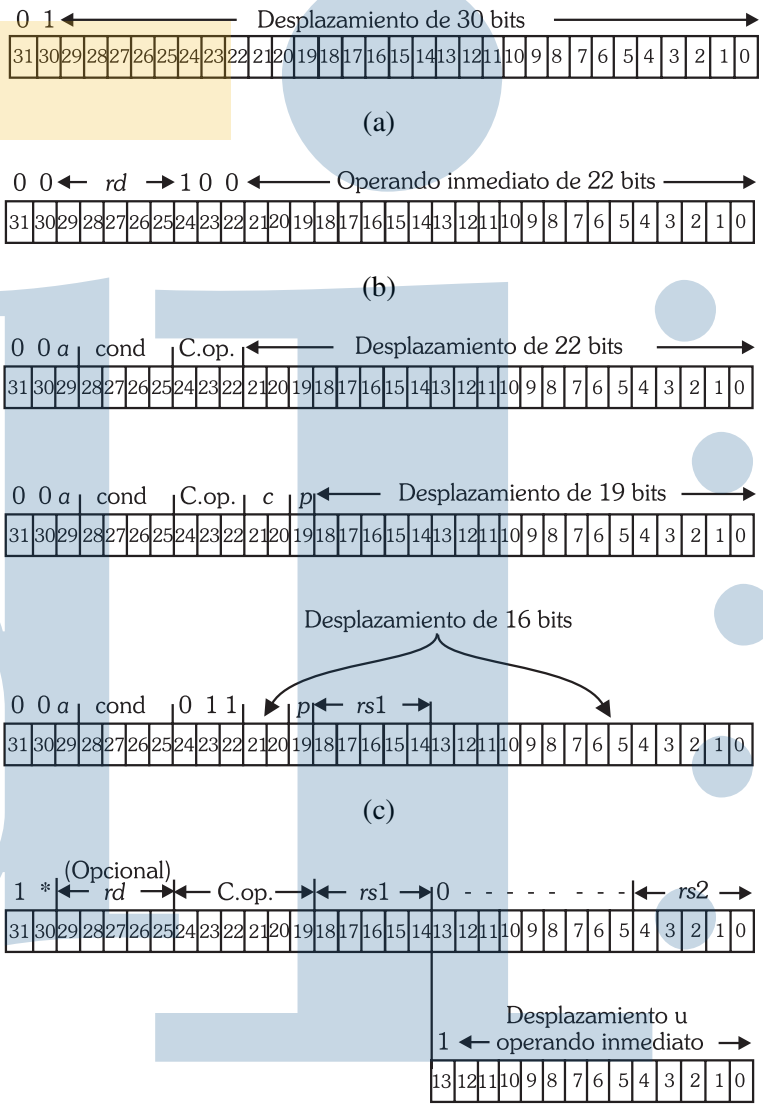


Fig. 2.10. Formatos de instrucción de la arquitectura SPARC: (a) Instrucción CALL, (b) Instrucción SETHI, (c) Instrucciones de bifurcación, (d) Instrucciones de carga, almacenamiento, aritméticas, lógicas, salto, retorno, SAVE y RESTORE (Continúa).

SITES, R.L. 1992. *Alpha Architecture Reference Manual*. Digital Press.

TANENBAUM, A.S. 2006. *Structured Computer Organization*. 5 edn. Prentice-Hall International. Existe traducción al castellano de la edición anterior: Organización de computadores: un enfoque estructurado, 4ª edición, Prentice-Hall Hispanoamericana, 2000.

WEAVER, D.L., & GERMOND, T. (EDITORES). 1994. *The SPARC Architecture Manual, version 9*. Prentice-Hall International.

ZAKS, R. 1982. *Programming the Z-80*. Sybex. Existe traducción al castellano: Programación del Z-80, Anaya Multimedia, 1985.

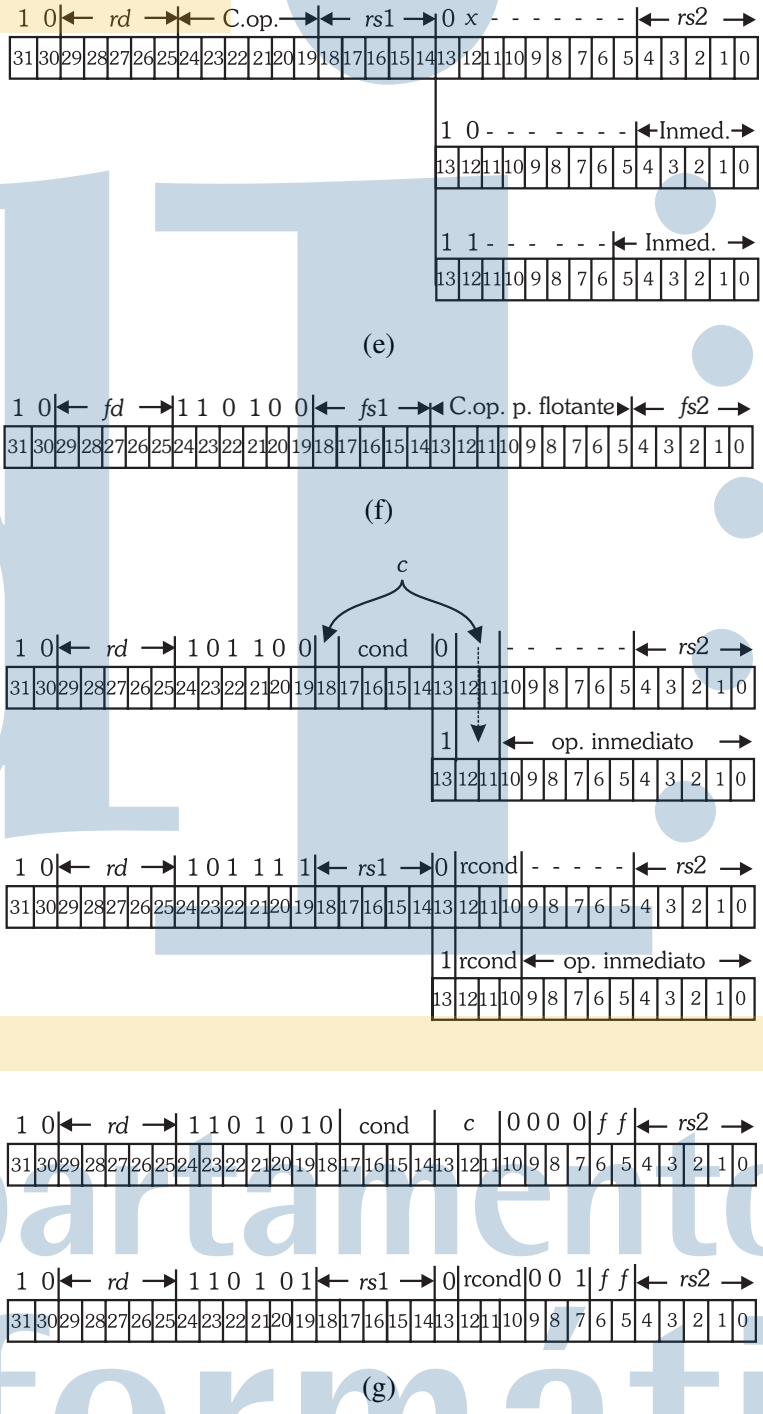


Fig. 2.10. Formatos de instrucción de la arquitectura SPARC: (e) Instrucciones de desplazamiento, (f) Instrucciones aritméticas de punto flotante y (g) Instrucciones de transferencia condicional (Conclusión).

CUESTIONES Y PROBLEMAS

2.1 Para un computador cuya longitud de palabra de instrucción es de 36 bits y que tiene 8 registros de uso general, diseñar los formatos de instrucción para que permitan soportar un juego de instrucciones con:

- 7 instrucciones con dos direcciones de 15 bits y un número de registro.
- 453 instrucciones con una dirección de 15 bits y un número de registro.
- 54 instrucciones sin operandos.

2.2 Un computador tiene instrucciones de 16 bits. Sus direcciones de operandos se especifican con campos de 6 bits. La máquina necesita 14 instrucciones de dos operandos y 20 instrucciones sin operandos. Explicar cuál será el número máximo de instrucciones de un operando que podrá tener este computador.

2.3 Cierta máquina tiene instrucciones de 32 bits y campos de operando de 8 bits. Este computador tiene instrucciones de 0, 1, 2 y 3 direcciones. En la máquina son necesarias 251 instrucciones de 3 operandos, 523 de 1 operando y 87 instrucciones sin operandos. ¿Cuál es el número máximo de instrucciones de 2 direcciones que puede tener ese computador?

2.4 Un computador tiene instrucciones con longitud fija de 16 bits. Sus direcciones de operandos se especifican en campos de 6 bits. La máquina tiene n instrucciones de dos operandos y m instrucciones sin operandos. Explicar cuál será el número máximo de instrucciones de un operando que podrá tener esta máquina.

2.5 Se desea diseñar los códigos de operación de un computador de forma que las instrucciones de ejecución más frecuente tengan códigos de operación más cortos. Sabiendo que las instrucciones del computador, junto con sus probabilidades dinámicas relativas son las mostradas en la tabla 2.4:

a) Elaborar el diseño con el código de Huffman.

b) Realizar el diseño mediante un código de operación con extensión.

Tabla 2.4.

Instrucción	Probabilidad dinámica
MOV	0,46
ADD	0,02
SUB	0,02
JMP	0,20
JMPN	0,11
JMPC	0,09
JMPZ	0,10

Tabla 2.5.

Instrucción	Probabilidad estática	Probabilidad dinámica
MOV	0,12	0,25
LOAD	0,17	0,15
STORE	0,20	0,17
BR	0,15	0,12
BNE	0,07	0,10
BPL	0,11	0,07
INC	0,05	0,03
DEC	0,10	0,06
SUB	0,02	0,03
ADD	0,01	0,02

2.6 a) Diseñar dos tablas de códigos de operación para las instrucciones de una máquina cuyas probabilidades se muestran en la tabla 2.5 de forma que se minimicen:

- La cantidad de memoria ocupada por los códigos de operación.
- El tiempo de decodificación de las instrucciones.

b) Calcular la longitud media del código de operación en los dos casos anteriores.

2.7 Diseñar los formatos de instrucción, tanto en cuanto a código de operación como a operandos, para un procesador con la estructura de la figura 2.11. La máquina se supone ortogonal, la palabra básica de instrucción tiene 16 bits y sus especificaciones son:

Repertorio de instrucciones:

- 15 instrucciones de 2 operandos.
- 9 instrucciones de 1 operando.
- 6 instrucciones que operan sobre un registro.
- 8 instrucciones sin operandos.
- 11 instrucciones de bifurcación con desplazamiento de 8 bits.

Operandos:

- Debe haber 7 modos de direccionamiento que operan sobre alguno de los registros generales y 6 que no precisan registro.

2.8 Diseñar los formatos de instrucción, tanto en cuanto a código de operación como a operandos, para una máquina con la estructura de la figura 2.11. La palabra básica de instrucción tiene 16 bits y el computador se supone ortogonal. Se deben cumplir las especificaciones siguientes:

Operandos:

- Debe haber 6 modos de direccionamiento que operan sobre uno de los registros de uso general y 8 que no precisan registro.

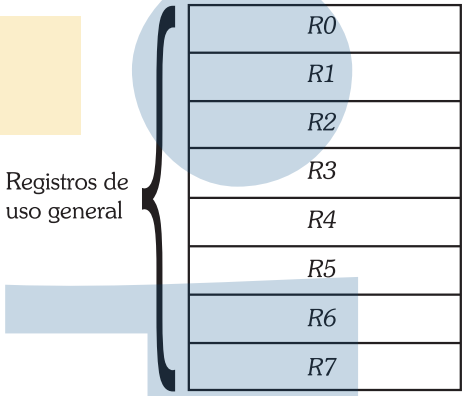


Fig. 2.11.

Repertorio de instrucciones:

- 15 instrucciones de 2 operandos.
- 6 instrucciones de un operando.
- 14 instrucciones que operan sobre un registro.
- 10 instrucciones sin operandos.
- 14 instrucciones de bifurcación con un desplazamiento comprendido entre -128 y 127.

2.9 Se quiere evaluar la expresión $W = (X*Y-Z)/(P+Q)$. Se dispone de cuatro máquinas de 0, 1, 2 y 3 direcciones. Los juegos de instrucciones de esas máquinas se muestran en la tabla 2.6. En ella, M es una dirección de memoria de 16 bits y A, B y C son direcciones de memoria de 16 bits o bien especificaciones de registros de 4 bits. Se supone que las máquinas de 2 y 3 direcciones son ortogonales y que, por tanto, los operandos de las instrucciones pueden residir tanto en registros como en memoria. Se sabe que los códigos de operación de todas las máquinas tienen 8 bits y que esos códigos incluyen la información sobre el modo de direccionamiento de los operandos cuando sea necesario. La longitud de las instrucciones es variable para todas las máquinas. ¿Cuántos bits necesitará un programa escrito en cada una de las máquinas para evaluar la variable W?

Tabla 2.6.

Máquina de 0 direcciones	Máquina de 1 dirección	Máquina de 2 direcciones	Máquina de 3 direcciones
PUSH M	LOAD M	MOV A,B	MOV A,B
POP M	STORE M		
ADD	ADD M	ADD A,B	ADD A,B,C
SUB	SUB M	SUB A,B	SUB A,B,C
MUL	MUL M	MUL A,B	MUL A,B,C
DIV	DIV M	DIV A,B	DIV A,B,C