Estructura de Computadores

Tema 1. El sistema computador

- Estructura general de un computador. Arquitectura de Von Neumann.
- Unidades generales: procesador, memoria, entrada/salida, unidad de control.
- Niveles de descripción.
- Código máquina y ensamblador.
- Características de los juegos de instrucciones. Métodos de descodificación. Direccionamientos.
- Clasificación del procesador según el número de referencias a memoria.

---

1. Ejercicios Resueltos

1.1.

¿Cuál es la diferencia entre una instrucción de direccionamiento inmediato, una de direccionamiento directo y una de direccionamiento indirecto? ¿Cuántas referencias de memoria se necesitan para cada uno de estos tipos de instrucciones para traer un operando a un registro del procesador? Poner ejemplos de instrucciones del 8086/88 de los distintos modos de direccionamiento.

Solución

Cuando una instrucción incluye un valor en el operando se dice que trabaja en modo de **direccionamiento inmediato**. La representación en código máquina de estas instrucciones ocupan dos posiciones consecutivas en la memoria:

- la primera palabra (o byte) contiene el código de operación,
- la segunda palabra contiene el valor del dato.

Para ejecutar esta instrucción el procesador tiene que acudir a la memoria dos veces.

En el 80x86 el operando se incluye en el segmento de código como parte de la instrucción. Por ejemplo,

```
ADD     AX,1234h
```

hará que el procesador sume el valor numérico 1234h al contenido actual del registro `AX`. Su codificación:

05	34	12
OPCODE	operando	

exige acceder a la memoria sólo dos veces: para extraer el opcode y para extraer el operando.

En la familia 80x86 no todas las instrucciones pueden aceptar operandos inmediatos.

Se dice que una instrucción emplea un **modo de direccionamiento absoluto o directo** si el campo operando contiene la dirección donde se encuentra el valor con el que se va a trabajar.

El código objeto correspondiente a esta instrucción ocupa dos posiciones de memoria consecutivas, sin embargo, al contrario del direccionamiento inmediato, el contenido de la posición de memoria que sigue al opcode corresponde a la dirección del operando.

Para ejecutar esta instrucción el procesador tiene que realizar tres accesos a memoria: uno para extraer el código de la operación, otro para obtener la dirección del operando y un tercero para obtener el propio operando.

En el 80x86, en las instrucciones que utilizan este modo de direccionamiento, un valor de 16 bits forma parte de la instrucción, interpretándose dicho valor como una dirección de memoria. Es el contenido de dicha dirección el operando que se requiere en la instrucción. El tamaño del operando queda determinado por el opcode de la instrucción. Ejemplo:

```
ADD     AX,[1234h]
```

que se codifica como `03  06  34  12`
suma el contenido de 16 bits de la dirección 1234h al valor actual de `AX`, y

```
ADD     AL,[1234h]
```

que se codifica `02  06  34  12`
suma a `AL` el byte direccionado por 1234h.

Cuando una instrucción especifica una dirección que contiene a la dirección del operando se tiene un **direccionamiento indirecto**. Habitualmente la dirección especificada en la instrucción es la de un registro, hablándose de **direccionamiento indirecto por registro**, considerándose varias técnicas: indexado, relativo a base, etc.

Una instrucción con auténtico direccionamiento indirecto requiere de cuatro accesos a memoria: extracción del opcode de la instrucción, extracción de la dirección que acompaña al opcode como valor (su segunda palabra), esta segunda palabra proporciona la dirección en donde se encuentra la dirección del operando, lo cual requiere dos accesos más a la memoria.

En el caso del direccionamiento indirecto por registro, que es el más habitual, en la palabra del opcode ya se encuentra codificado el registro que contiene la dirección del operando, por tanto en este caso bastan tres accesos a la memoria.

Ejemplo en el 80x86:

```
ADD     AX,[BX]          ;  03  07
ADD     AX,[SI]          ;  03  04
ADD     AX,[BX+0100h]    ;  03  87  00  01
```

1.2.

Enumerar cuatro unidades principales de un sistema computador.

Solución

Unidad Central de Procesamiento (CPU), Unidad de Memoria, Unidad de Entrada y Salida, Unidad de Interconexión. (Entrada, Salida, Memoria, Camino de Datos, Control).

1.3.

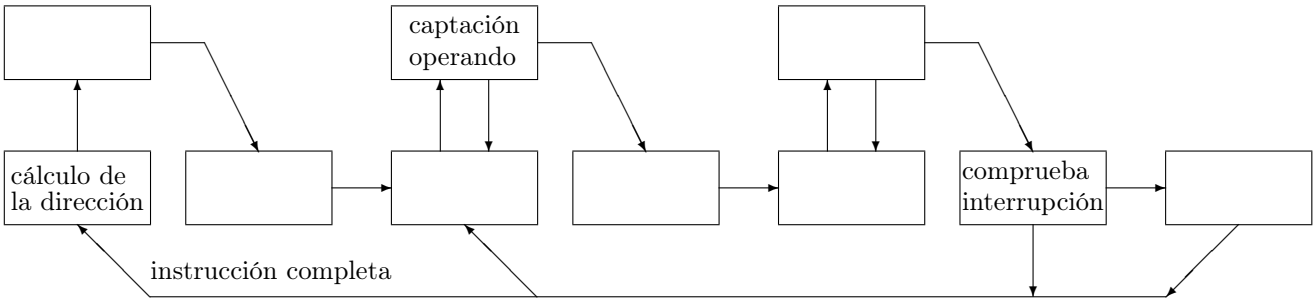
Nombrar al menos cuatro subciclos, o fases, de un ciclo de instrucción.

Solución

Captación o extracción de la instrucción, decodificación, acceso a operandos, ejecución, interrupción.

1.4.

Completar las etiquetas de los cuadros de la figura.



Solución

captación de la instrucción
decodificación de la operación de la instrucción
cálculo de la dirección del operando
operación con los datos (ejecución)
cálculo de la dirección del operando
almacenamiento de operando
interrupción

1.5.

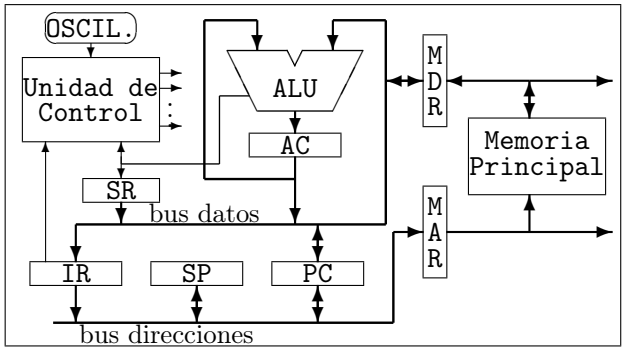
Un microprocesador tiene la palabra de instrucción de 8 bits de tamaño y un campo de dirección es de tres bits. En el repertorio de instrucciones hay 20 instrucciones sin operandos y 13 instrucciones con un operando. ¿Cuántas instrucciones de dos operandos puede tener como máximo?

Solución

2 instrucciones de dos operandos.
Dos operandos de tres bits dejan espacio para un opcode de sólo 2 bits en una palabra de instrucción de 8 bits. Las alternativas son: 4 instrucciones de dos operandos y nada más. O bien tres instrucciones de dos operandos y reservar uno de los códigos para expandir el opcode hasta 8 instrucciones más de un operando. Como no es suficiente, se pueden reservar dos de los códigos. Así, se tienen dos instrucciones de dos operandos y hasta 16 instrucciones de un operando. El mismo método puede usarse para tener instrucciones de cero operandos. Cada código que se expanda proporciona 8 instrucciones más.

1.6.

En la máquina de la figura ¿qué secuencia de operaciones elementales se realizan en la instrucción de llamada a subrutina? Los registros `PC` y `SP` son contadores.

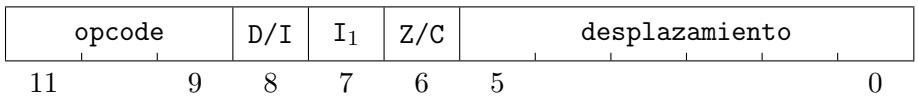


Solución

MAR ← (PC)
MDR ← Mem[MAR]
IR ← (MDR) || PC ← PC + 1
SP ← SP - 1
MAR ← SP || MDR ← (PC)
Mem[MAR] ← (MDR)
PC ← IR<dirección>

1.7.

Asumir que una instrucción tiene el siguiente formato:



donde `D/I` = directo/indirecto y `Z/C` = *zero/current page*.

El contenido de los registros, en hexadecimal, es:

Registro índice I_1	`492h`
Contador de programa (`PC`)	`924h`

El direccionamiento por registro índice de este computador significa que se suma el contenido del registro índice al desplazamiento. Suponer que hay 2^{12} posiciones de memoria, que comprenden 2^6 páginas de 2^6 palabras. Determinar las direcciones efectivas, en hexadecimal, de las siguientes instrucciones:

(a) `295h` **(b)** `66Ah`

Solución

(a) instrucción = $295h$ = 0010 1001 0101, luego

`opcode`	=	001	
`D/I`	=	0	⇒ indirecto
I_1	=	1	⇒ indexado
`Z/C`	=	0	⇒ página cero
`desplazamiento`	=	01 0101	= $015h$

EA = (`página`×2^6 + I_1 + `desplazamiento`)
= ($000h + 492h + 015h$)
= ($4A7h$) (indirecto)

(b) instrucción = 66Ah = 0110 0110 1010, luego

$$\begin{aligned}
\texttt{opcode} &= 011 \\
\texttt{D/I} &= 0 && \Rightarrow \text{indirecto} \\
\texttt{I}_1 &= 0 && \Rightarrow \text{no indexado} \\
\texttt{Z/C} &= 1 && \Rightarrow \text{página actual en el } \texttt{PC} \\
\texttt{desplazamiento} &= 10\ 1010 && = 02\text{A}h \\
\texttt{página} &= \texttt{PC}\times 2^{-6} && = 924h \times 2^{-6} = 1001\ 0010\ 0100 >> 6 = 100100 = 024h
\end{aligned}$$

$$\begin{aligned}
\text{EA} &= (\ \texttt{página}\times 2^6 + \texttt{desplazamiento}\ ) \\
&= (\ 900h + 02\text{A}h\ ) \\
&= (92\text{A}h) \qquad \text{(indirecto)}
\end{aligned}$$

1.8.

Disponemos de un μP de 32 bits cuyas instrucciones de 32 bits están compuestas por dos campos: el primer byte contiene el código de operación (`OPCODE`) y los restantes, un operando inmediato o una dirección de operando.

a) ¿Cuál es el tamaño de memoria que se puede direccionar directamente?
b) ¿Cuántos bits necesita el contador de programa (`PC`) y el registro de instrucciones (`IR`)?
c) Discutir el impacto que se produciría en la velocidad del sistema si el μP tiene:

1. Un bus de direcciones de 16 bits y un bus de datos de 16 bits.
2. Un bus de dirección de 32 bits y un bus de datos de 16 bits.

Solución

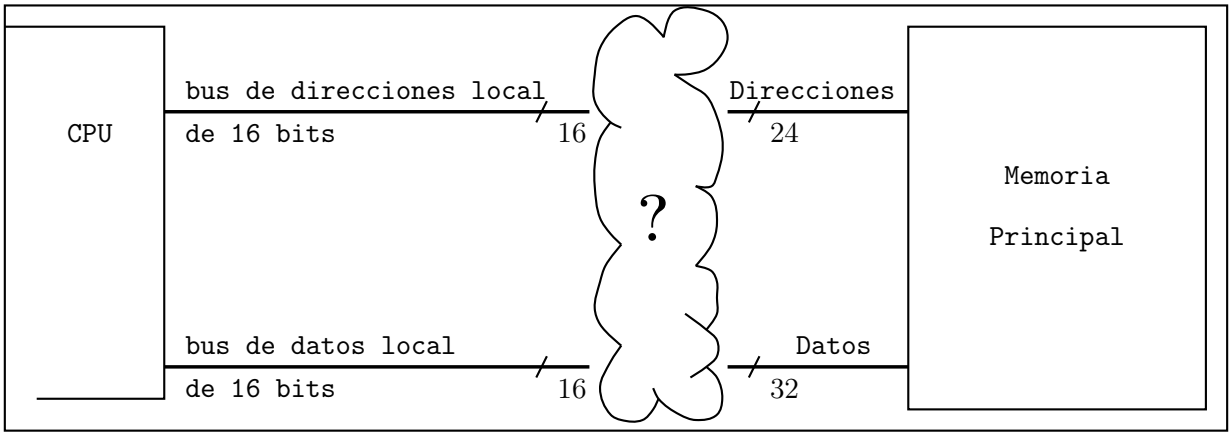
De los 32 bits de la instrucción, si 8 bits son del opcode quedan 24 bits para la dirección de un operando.
(a) Con 24 bits se pueden direccionar hasta 2^{24} = 16 M palabras.

Respuesta:	16 M palabras

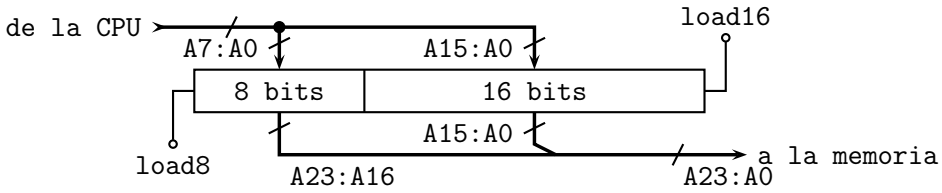
(b) El contador del programa debe acceder a todo este espacio direccionable, luego el PC necesita tener 24 bits.
El registro de instrucciones debe contener a toda la instrucción de 32 bits.

Respuesta:	PC de 24 bits IR de 32 bits

(c.1) Si el bus de direcciones fuese de 16 bits y el bus de datos también de 16 bits, entonces el acceso a una palabra de memoria requeriría al menos 2 ciclos de acceso, con la memoria paginada o segmentada, o bien un ciclo de máquina adicional a los dos de acceso para renovar el registro de página/segmento.



El interfaz requiere unos registros donde se mantenga la información que se transmite a trozos. Un registro de direcciones, de 24 bits al menos, que lo escribe la CPU en dos veces:



La CPU debe efectuar dos escrituras por el bus de direcciones (local) para poner toda la dirección en el registro de 24 bits.

El registro superior de 8 bits puede considerarse como un registro de página. Si las direcciones se mueven en un rango (de hast 64 K palabras) en el que no se modifica esta parte de la dirección, la CPU puede ahorrarse la escritura del registro de 8 bits en esos casos, mientras las direcciones se muevan dentro de la "página".

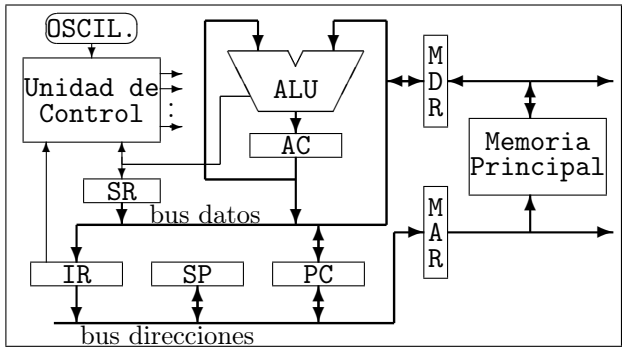
(c.2) Si el bus de datos es de 32 bits el acceso a la memoria se realizará en un solo ciclo, con la memoria paginada o segmentada, y se necesitará un ciclo de máquina adicional cuando se requiera renovar el registro de página/segmento.

1.9.

En el repertorio de instrucciones de la máquina de la figura se encuentran las siguientes:

Mnemónico		Descripción
`LOAD`	`X`	carga el acumulador con una palabra de la memoria
`STORE`	`X`	guarda el acumulador en la memoria
`ADD`	`X`	suma una palabra de la memoria al acumulador
`CALL`	`X`	llamada de subrutina
`JUMP`	`X`	salto incondicional
`Jcond`	`X`	salto condicional (según valores de `SR`)
`PUSH`	`X`	guarda una variable en la pila
`PUSHA`		guarda el acumulador en la pila

La ALU, además de sumar es capaz de restar (`SUB`), incrementar y decrementar (`INC`, `DEC`), poner a cero y a uno (`CLR`, `SET`), operaciones lógicas (`AND`, `OR`, ...) y otras. También tiene las instrucciones `RET`, `POP` y `POPA`. El registro de estado (`SR`) proporciona los bits típicos de acarreo, signo, cero, ...



Se pide:

a) Escribir todas las micro-operaciones (operaciones elementales) que se precisan para ejecutar cada una de las instrucciones de la tabla.

b) Dibujar el esquema de la máquina mejorada con la adición de dos registros índices `I` y `J`.

Solución

(a) La fase de extracción es común a todas las instrucciones:

$$\begin{array}{ll} \text{MAR} \leftarrow \text{PC} & \text{; a través del bus de direcciones} \\ \text{MDR} \leftarrow \text{M[MAR]} & \\ \text{IR} \leftarrow \text{MDR} & \text{; a través del bus de datos} \\ \quad \| \ \text{PC} \leftarrow \text{PC} + 1 & \text{; por sus propios medios} \end{array}$$

Las distintas fases de ejecución de las instrucciones de la tabla asumen que la referencia al dato operando (X) es a la memoria en modo directo.

(a.1) `LOAD`

$$\begin{array}{l} \text{MAR} \leftarrow \text{IR<X>} \\ \text{MDR} \leftarrow \text{M[MAR]} \\ \text{AC} \leftarrow \text{ALUop(pasa B)} \end{array}$$

(a.2) `STORE`

$$\begin{array}{l} \text{MAR} \leftarrow \text{IR<X>} \\ \text{MDR} \leftarrow \text{Ac} \\ \text{M[MAR]} \leftarrow \text{MDR} \end{array}$$

(a.3) `ADD`

$$\begin{array}{l} \text{MAR} \leftarrow \text{IR<X>} \\ \text{MDR} \leftarrow \text{M[MAR]} \\ \text{Ac} \leftarrow \text{ALUop(suma A y B)} \end{array}$$

(a.4) `CALL`

$$\begin{array}{l} \text{MDR} \leftarrow \text{PC} \ \| \ \text{SP} \leftarrow \text{SP} + 1 \\ \text{MAR} \leftarrow \text{SP} \ \| \ \text{PC} \leftarrow \text{IR<X>} \\ \text{M[MAR]} \leftarrow \text{MDR} \end{array}$$

(a.5) `JUMP`

$$PC \leftarrow IR\langle X\rangle$$

(a.6) `Jcond`

$$\text{if cond then } PC \leftarrow IR\langle X\rangle$$

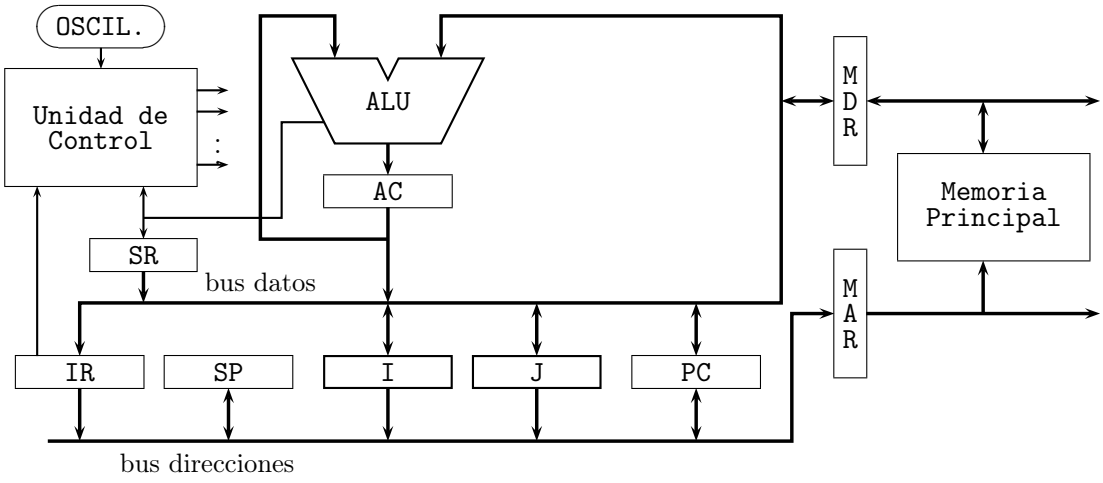
(a.7) `PUSH`

$$MAR \leftarrow IR\langle X\rangle \parallel SP \leftarrow SP + 1$$
$$MDR \leftarrow M[MAR]$$
$$MAR \leftarrow SP$$
$$M[MAR] \leftarrow MDR$$

(a.8) `PUSHA`

$$SP \leftarrow SP + 1$$
$$MAR \leftarrow SP \parallel MDR \leftarrow Ac$$
$$M[MAR] \leftarrow MDR$$

(b) Los registros índices deben contener direcciones, por tanto deberán estar conectados al bus de direcciones; por otro lado deben tener la posibilidad de ser inicializados, bien con variables procedentes de la memoria o con valores resultado de operaciones de la ALU, por tanto también deberán estar conectados al bus de datos.



1.10.

Sea una arquitectura de 16 bits con una memoria de 32 KB, 16 registros y 40 instrucciones diferentes. Se desea implementar la instrucción `ADDM`, la cual suma el contenido de un registro con el contenido de una dirección de memoria principal, almacenando el resultado en esta posición de memoria.

(a) Indicar el formato de la instrucción para cada uno de los siguientes modos de direccionamiento de la memoria: direccionamiento directo, direccionamiento de registro, direccionamiento relativo a registro base, direccionamiento relativo a contador de programa y direccionamiento indirecto.

(b) ¿Cuántos accesos a memoria se debe realizar en cada caso? Justificar la respuesta.

Solución

1. La instrucción es de la forma:

```
ADDM    mem, reg
```

de modo que hay que especificar dos direcciones, una de memoria que hace de fuente y destino a la vez, y otra de registro que hace de segunda fuente: $mem \leftarrow mem + R_i$. El formato de la instrucción constará de tres o de cuatro campos, según el modo de direccionamiento. El campo del código de operación (opcode) es fijo, de $\lceil \log_2(40) \rceil = 6$ bits. El campo del registro fuente también es fijo de $\log_2(16) = 4$ bits (= valor de i).

   a) En el caso de direccionamiento directo hay que guardar una dirección absoluta de memoria de $\log_2(32\ \text{KB}) = 15$ bits. En total, pues, $6+15+4 = 25$ bits, lo que implica a dos palabras.
   b) En el caso de direccionamiento de registro ($mem = [R_k]$) sólo se necesita especificar el número del registro que contiene la dirección de la memoria, es decir 4 bits. En total, $6+4+4 = 14$ bits, por lo que basta una palabra.
   c) En el caso de direccionamiento relativo a registro base la dirección de memoria se construye sumando al contenido del registro base un desplazamiento ($mem = [R_k] + desplazamiento$). Hay que especificar el registro base (número k) y el desplazamiento, de 15 bits. Es un caso de instrucción con cuatro campos. En total, $6+4+15+4 = 29$ bits, luego hacen falta dos palabras para la instrucción.
   d) El caso del direccionamiento relativo al contador de programa es semejante al anterior, con la salvedad que el registro no hay que especificarlo, es implícito su uso, por lo que tan sólo hay que especificar el desplazamiento de 15 bits. En total, pues, 25 bits como en el primer caso.
   e) En el direccionamiento indirecto hay que especificar una dirección de memoria, con lo que el formato es como en el primer caso, de 25 bits.

2. ... en proceso ...

1.11.

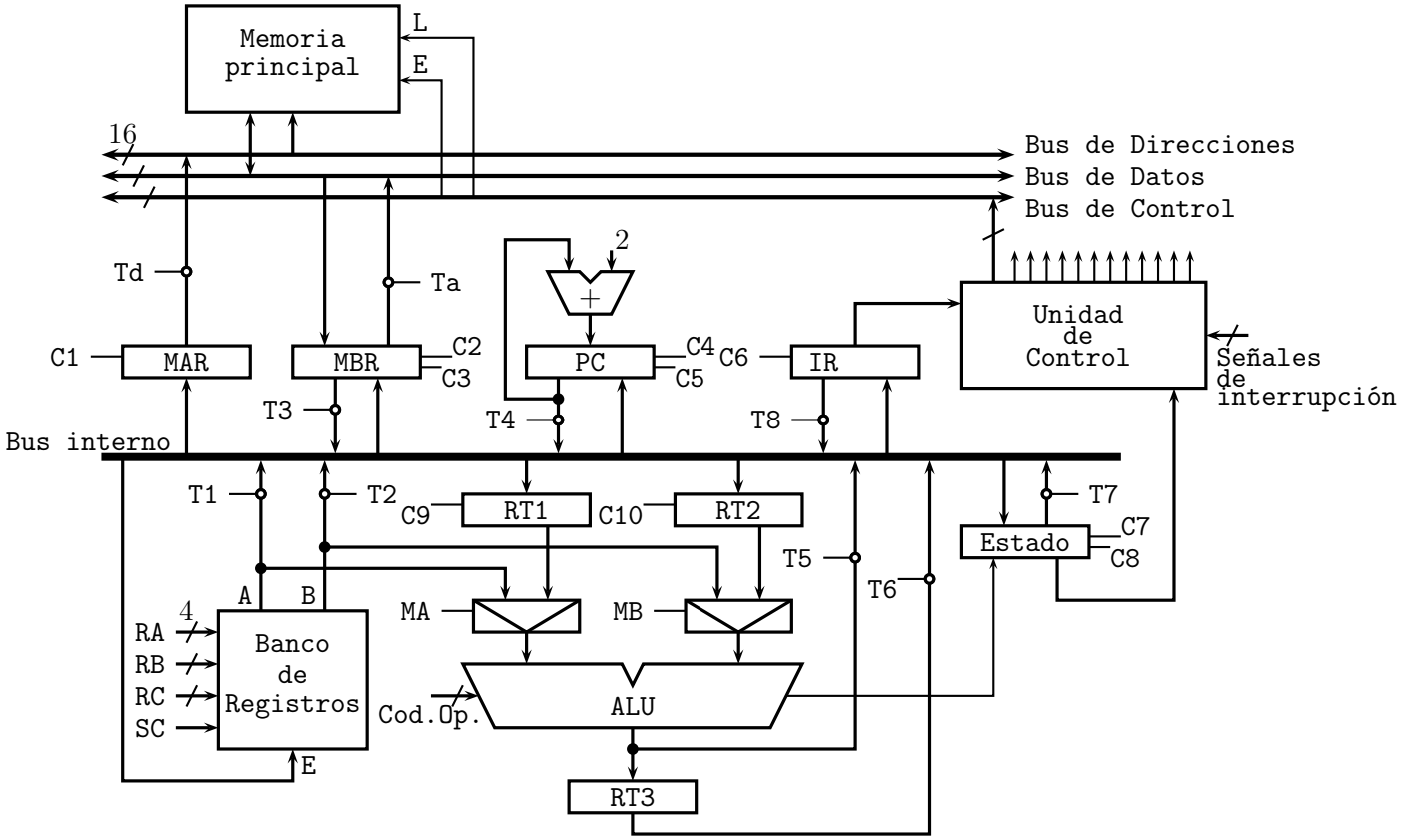
Considérese un procesador de 16 bits con la estructura que se muestra en la figura siguiente, y con una ALU capaz de realizar 16 operaciones aritméticas y lógicas. El procesador necesita un ciclo para decodificar la instrucción y la memoria un ciclo para acceder a su contenido, ya sea en operaciones de lectura como de escritura. La memoria se direcciona por palabras. Considérese que en este procesador el puntero de pila (R15) apunta a la última posición ocupada de la pila. Se pide:

(a) Indicar las operaciones elementales de la instrucción `PUSH R1`, que inserta un elemento en la pila. Indicar además, para cada operación elemental, las señales de control necesarias.

(*b*) Indicar las operaciones elementales de la instrucción `POP R1`, que extrae un elemento de la pila. Indicar además, para cada operación elemental, las señales de control necesarias.

(*c*) Indicar las operaciones elementales de la instrucción `CALL DESPL`, que salta a ejecutar el código situado en `DESPL` tras guardar en la pila la dirección de la siguiente instrucción. Indicar además, para cada operación elemental, las señales de control necesarias.

(*d*) Indicar las operaciones elementales de la instrucción `RET`, que extrae una dirección de la pila y salta allí para seguir ejecutando instrucciones. Indicar además, para cada operación elemental, las señales de control necesarias.



Solución

1. Instrucción `PUSH`:

c1	`MAR` ← `PC`	; `T4,C1`
c2	Lectura, `PC` ← `PC+1`, `MBR` ← `Mem`	; `L,C4,C2,Td`
c3	`RI` ← `MBR`	; `T3,C6`
c4	Decodificación	
c5	`SP` ← `SP-1`	; `RA=R15,MA=0,Cod.Op.=`Sumar 1`,T5,RC=R15,SC`
c6	`MAR` ← `SP`	; `RA=R15,T1,C1`
c7	`MBR` ← `R1`	; `RA=R1,T1,C3`
c8	Escritura, `Mem` ← `MBR`	; `E,Td,Ta`

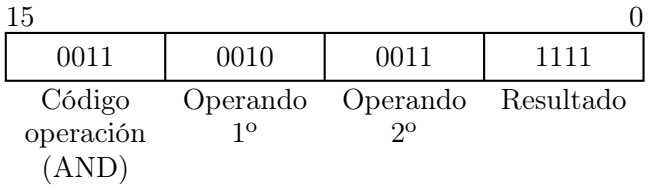
2. Instrucción `POP`: ... en proceso ...

1.12.

Un computador maneja una Memoria Principal de 16 posiciones de 8 bits cada una, cuyos contenidos se muestran en el cuadro siguiente.

Dirección	Memoria Principal
0000	0000 0000
0001	1111 1111
0010	0000 0000
0011	1111 1111
0100	0000 0000
0101	1111 1111
0110	0000 0000
0111	1111 1111
1000	0000 0000
1001	1111 1111
1010	0000 0000
1011	1111 1111
1100	0000 0000
1101	1111 1111
1110	0000 0000
1111	1111 1111

El formato de una determinada instrucción para este computador es el siguiente:



Sabiendo que los operandos y el resultado que participan en la instrucción se expresan en el formato con la dirección de la memoria donde se ubican y que el primer campo de la instrucción representada en la figura anterior corresponde al código asignado a la instrucción lógica AND, averiguar cómo se modifican los contenidos de la memoria después de haber ejecutado dicha instrucción.

Solución

El primer operando que interviene en la operación AND se encuentra en la posición de memoria 0010 y su contenido es 000 000. El segundo operando ocupa la dirección 0011 y contiene 111 111. La operación será:

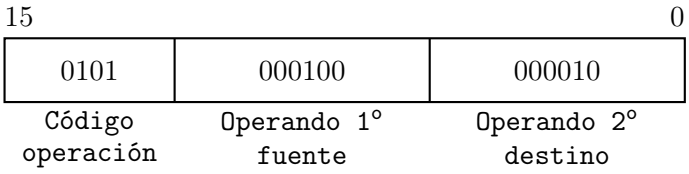
$$\textbf{0000 0000 AND 1111 1111} = \textbf{0000 0000}$$

El resultado de esta instrucción se deposita en la dirección 1111, que inicialmente contenía el valor 1111 1111, con lo que pasará a contener el valor 000 000 después de ejecutar la instrucción con el formato de la figura del enunciado. El resto de las posiciones de la memoria no se modificará.

1.13.

Se dispone de un computador en el que el formato de las instrucciones sólo especifica dos operandos: el fuente y el destino. Los campos para dichos operandos constan de 6 bits y expresan la dirección de la memoria principal donde residen.

Indicar las posiciones de la memoria que se modificarán después de ejecutar la instrucción cuyo formato es el que se muestra en la siguiente figura.



Se conocen los siguientes datos:

(a) El código de operación 0101 corresponde a la instrucción OR.
(b) (000100) = 01010101.
(c) (000010) = 10101010.

Solución

El resultado de la operación lógica OR es:

$$\textbf{01010101 OR 10101010 = 1111 1111}$$

Como el resultado de la instrucción se deposita en el operando destino, que corresponde a la dirección 000 010, ésta es la única que altera el contenido en la memoria, que pasa de contener 10101010 a 1111 1111.

1.14.

Un computador dispone en su Camino de Datos de un Banco de Registros con ocho diferentes (`R0, R1, ..., R7`). En el formato de las instrucciones se especifican los operandos fuente y destino, que se supone están depositados en alguno de los registros. Los registros se seleccionan mediante 3 bits (`000: R0, 001: R1, ..., 111: R7`).

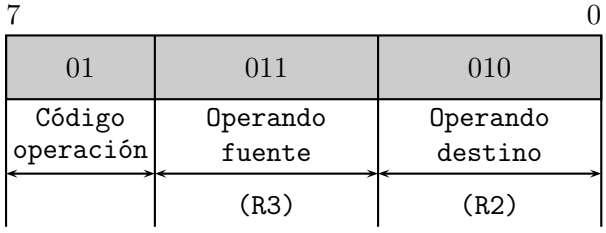
Sabiendo que el código de operación asignado a la instrucción de SUMAR es el 01:

(a) ¿De cuántos bits se compone el formato de la instrucción: SUMAR EL CONTENIDO DE R2 Y R3 DEPOSITANDO EL RESULTADO EN R2?
(b) Dibujar el formato de la instrucción anterior.

Solución

(a) El formato de la instrucción tiene tres campos: Código de la operación, Operando Fuente y Operando Destino.
El Código de Operación tiene dos bits y los operandos se expresan mediante tres bits.
El formato de la instrucción consta de 8 bits.

(b) En la figura siguiente se representa el formato de la instrucción `SUMAR R3, R2`. Téngase en cuenta que `R2` es el operando destino.

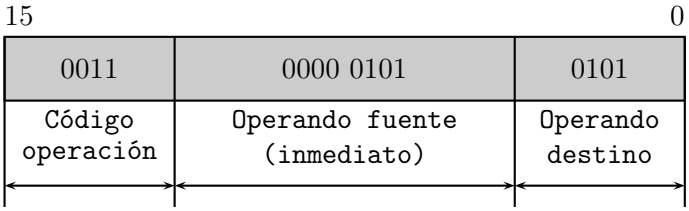


1.15.

Un determinado computador dispone de instrucciones cuyo formato consta de tres campos, con las siguientes especificaciones:

1. El código de operación es de cuatro bits.
2. El operando fuente es un valor inmediato de 8 bits.
3. El operando destino consta de 4 bits que forman la dirección de la memoria principal donde se encuentra el segundo operando y donde se deposita el resultado.

Si se tiene que ejecutar la instrucción cuyo formato se muestra en la figura siguiente, indicar las modificaciones que ocasionará en las posiciones de la memoria. Se sabe que el código de operación 0011 corresponde a la operación de SUMAR y que en la operación de memoria 0101 existe el dato 000 000 antes de ejecutar la instrucción.



Solución

La instrucción expresada en la figura anterior suma el valor inmediato del operando fuente que se indica en el formato con el contenido de la dirección 0101, dejando el resultado en esta última.

$$\begin{array}{r} 0000\ 0101 \\ +\ 0000\ 0000 \\ \hline 0000\ 0101 \end{array}$$

El resultado se deposita en el operando destino, que es la posición de memoria cuya dirección es la 0101, que será la única que modifique su valor después de ejecutar la instrucción.

1.16.

Si el código de operación de la instrucción XOR en un computador es 0101 1010 y los dos operandos (fuente y destino) se localizan en posiciones de la memoria, cuyas direcciones constan de 8 bits.

(a) Interpretar el significado del formato de la instrucción de 3 bytes que se indica:

0101 1010 0000 1000 0000 0010

(b) Codificar dicha instrucción en lenguaje hexadecimal.

Solución

(a) El formato de la instrucción indica que hay que realizar una operación lógica XOR entre los contenidos de las posiciones de memoria de direcciones 8 (fuente) y 2 (destino), dejando el resultado de la operación en la posición de dirección 2.

(b) **5A 08 02**

1.17.

Un sistema computador tiene 24 líneas de direcciones y 16 líneas de datos en su bus externo. ¿Qué capacidad máxima de memoria puede tener?

RESPUESTA.- 2^{24} palabras de 16 bits (16 Mwords).
