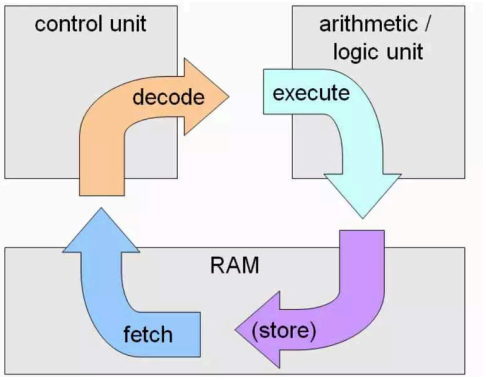
Escuela Davinci, 2024

Carrera Analista de Sistemas
Materia Arquitectura y Sistemas Operativos

¿Cómo el procesador ejecuta instrucciones?

Para ejecutar un programa el microprocesador ejecuta una a una las líneas de instrucción siguiendo el ciclo de instrucción (Fetch – Decode – Execute – End).



1. Fetch: En esta etapa el procesador lee la instrucción de memoria, la dirección de esta instrucción está definida por el registro PC.

2. Decode: En esta etapa se decodifica la secuencia de bits leída en la anterior etapa y se determinan las acciones que el procesador va a realizar para ejecutar la instrucción.

3. Ejecución: En esta etapa se ejecutan las acciones determinadas en la anterior etapa hasta completar la ejecución total de la instrucción. El resultado se guarda en algún registro o en la memoria (depende la instrucción).

4. Store: En esta etapa se almacena el resultado de la anterior etapa en la memoria, sin embargo no todas las instrucciones almacenan sus resultados en memoria.

Registros del procesador

El Registro Contador de Programa PC (Program Counter) es también llamado IP (Puntero de Instrucciones) y contiene la dirección de memoria de la siguiente instrucción a ejecutar.

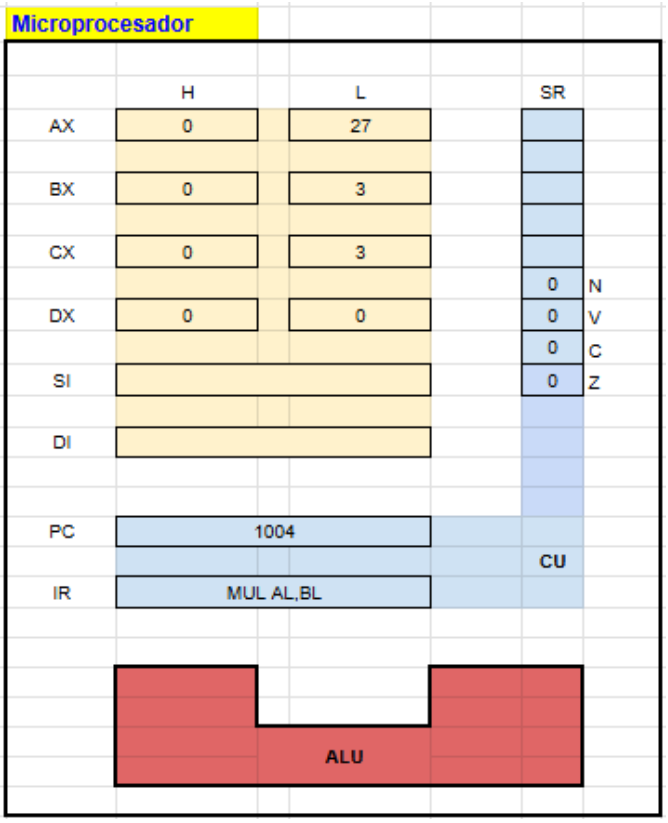
El Registro de Instrucciones IR (Instruction Register) contiene la instrucción en ejecución.

El Registro Puntero de Pila SP (Stack Pointer), contiene la dirección de memoria de un dato en segmento de pila de un proceso.

Los Registros SI (Source Index) y DI (Destination Index), también contienen direcciones de memoria y son usados en los direccionamientos de datos Directo e Indirecto.

Los registros A, B, C, D son de propósito general, ayudan en las operaciones aritméticas (+ - x /) y lógicas (and or xor) conteniendo a los operandos.

Otro tipo de registros como el Registro de Estado SR (Status Register) se dividen en banderas o flags que son bits que se activan bajo una condición.



La bandera Z vale ‘1’ si el resultado de una operación da cero, caso contrario Z vale ‘0’, del mismo modo la bandera C depende de si hubo acarreo o no, la bandera V depende de si hubo desbordamiento o no, la bandera N vale ‘1’ si el resultado de una operación es negativo. Existen muchos más registros y banderas.

Si en algún momento el programa tiene que hacer un salto o “jump”, se ejecuta alguna instrucción que modifica el PC, de ese modo se “desvía” el flujo de ejecución del programa (entiéndase como GOTO en lenguaje C).

Una instrucción como if-else es un salto condicional, si se cumple una condición X se ejecutan unas líneas caso contrario se ejecutan otras líneas distintas. Esto mismo en bajo nivel (Assembler) podría ser: si la bandera Z está en ‘1’ modificamos PC y ejecutamos un bloque de instrucciones, caso contrario ejecutamos un bloque de instrucciones distinto.

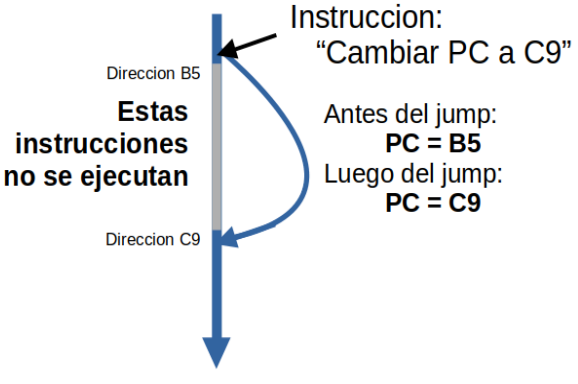


Figura. Jump sin condicion

Ejemplo 1.-
Se tiene el siguiente programa cargado en memoria principal listo para ejecutarse, este programa calcula 3 elevado a 5, el resultado de esta operación se deja en el registro A.

MEMORIA PRINCIPAL (RAM)...		
DIR (Hex)	**INSTR. (ASM)**	**DESCRIPCIÓN**
1000h	MOV CL,4	Mover 4 inmediatamente a la parte baja de registro C
1001h	MOV AL,3	Mover 3 inmediatamente a la parte baja de registro A
1002h	MOV BL,3	Mover 3 inmediatamente a la parte baja de registro B
1003h	MUL AL,BL	Multiplicar AL y BL, resultado en AL
1004h	SUB CL,1	Descontar en 1 a CL (Z puede ser afectado)
1005h	JZ <1007h>	Si Z activo, afecta a PC (sobreescribe 1007h en PC)
1006h	JMP <1003h>	Afecta a PC (sobreescribe 1003h en PC)
1007h	HLT	No operation (Fin de la ejecución)

Solución ejemplo 1.-

Corremos dicho programa mostrando la evolución de los valores de los registros usados, en la tabla.

VISTO DESDE EL MICROPROCESADOR...						
IR	**PC**	**AL**	**BL**	**CL**	**Z**	**Notas y comentarios…**
-	1000	-	-	-	0	
MOV CL,4	1001	-	-	**4**	0	
MOV AL,3	1002	**3**	-	4	0	
MOV BL,3	1003	3	**3**	4	0	
MUL AL,BL	1004	**9**	3	4	0	
SUB CL,1	1005	9	3	**3**	0	
JZ <**1007**>	1006	9	3	3	0	
JMP <1003>	**1003**	9	3	3	0	
MUL AL,BL	1004	**27**	3	3	0	
SUB CL,1	1005	27	3	**2**	0	
JZ <**1007**>	**1006**	27	3	2	0	
JMP <1003>	**1003**	27	3	2	0	
MUL AL,BL	1004	**81**	3	2	0	
SUB CL,1	1005	81	3	**1**	0	
JZ <**1007**>	**1006**	81	3	1	0	
JMP <1003>	**1003**	81	3	1	0	
MUL AL,BL	1004	**243**	3	1	0	
SUB CL,1	1005	243	3	**0**	**1**	Z se pone en 1 ya que CL-1 da 0
JZ <**1007**>	**1007**	243	3	0	**0**	Z vuelve a cero inmediatamente después para no afectar a otros programas que usen el bit Z para su funcionamiento
HLT	**1008**	243	3	0	0	Observar que PC indica seguir con la siguiente dirección, donde habrán otras instrucciones correspondientes a otro programa…

Ejemplo 2.-
Se tiene el siguiente programa cargado en memoria principal listo para ejecutarse, este programa realiza la división de 7 (Dividendo) entre 2 (Divisor), el resultado de esta operación se deja en el registro A (Cociente) y en el registro D (Residuo).

VISTO DESDE LA MEMORIA PRINCIPAL (RAM)...		
DIRECCION	**INSTRUCCION**	**DESCRIPCIÓN**
0x099	MOV AH,0	Cargar 0 en registro AH
0x100	MOV DH,7	Cargar 7 en registro DH
0x101	MOV BH,2	Cargar 2 en registro BH
0x102	SUB DH,BH	Restar DH y BH, resultado en DH (N afectado)
0x103	JS 0x106	Saltar si N=1, afecta a PC
0x104	INC AH	AH se incrementa en 1
0x105	JMP 0x102	Saltar siempre, afecta a PC
0x106	ADD DH,BH	Sumar DH y BH, resultado en DH
0x107	HLT	Fin de la ejecución

Solución ejemplo 2.-

Corremos dicho programa mostrando la evolución de los valores de los registros usados, en la tabla.

VISTO DESDE EL MICROPROCESADOR					
Reg. Instruccion	**Cont de Prog**				
IR	**PC**	**Reg AH**	**Reg BH**	**Reg DH**	**flag N**
x	099	x	x	x	0
MOV AH,0	100	0	x	x	0
MOV DH,7	101	0	x	7	0
MOV BH,2	102	0	2	7	0
SUB DH,BH	103	0	2	5	0
JS 0x106	104	0	2	5	0
INC AH	105	1	2	5	0
JMP 0x102	102	1	2	5	0
SUB DH,BH	103	1	2	3	0
JS 0x106	104	1	2	3	0
INC AH	105	2	2	3	0
JMP 0x102	102	2	2	3	0
SUB DH,BH	103	2	2	1	0
JS 0x106	104	2	2	1	0
INC AH	105	3	2	1	0
JMP 0x102	102	3	2	1	0
SUB DH,BH	103	3	2	-1	1
JS 0x106	106	3	2	-1	0
ADD DH,BH	107	3	2	1	0
HLT	108	3	2	1	0

Ejemplo 3.-
Se tiene el siguiente programa cargado en memoria principal listo para ejecutarse, este programa desplaza el dato cargado en AL 1 bit hacia la izquierda, 3 veces.

VISTO DESDE LA MEMORIA PRINCIPAL (RAM)...		
DIREC (Hex)	**INSTR (Asm)**	**DESCRIPCIÓN**
FA5	MOV AL,1	Mover 1 al registro AL
FA6	MOV CL,3	Mover 3 al registro CL
FA7	SHL AL,1	Desplazar (shift) los bits de AL una posición a la izq.
FA8	SUB CL,1	Descontar en 1 a CX
FA9	JZ FAB	Si Z activo saltar a 0xFAB
FAA	JMP FA7	Saltar a 0xFA7 (sin condición)
FAB	HLT	Fin de la ejecución

VISTO DESDE EL MICROPROCESADOR					
IR	**PC (hex)**	**Reg AL (bin)**	**Reg CL**	**flag Z**	**Comentario**
x	FA5	-	-	0	
MOV AL,1	FA6	0000 0001	-	0	El valor en AL es 1 visto en decimal
MOV CL,3	FA7	0000 0001	3	0	
SHL AL,1	FA8	0000 0010	3	0	
SUB CL,1	FA9	0000 0010	2	0	
JZ FAB	FAA	0000 0010	2	0	
JMP FA7	FA7	0000 0010	2	0	
SHL AL,1	FA8	0000 0100	2	0	
SUB CL,1	FA9	0000 0100	1	0	
JZ FAB	FAA	0000 0100	1	0	
JMP FA7	FA7	0000 0100	1	0	
SHL AL,1	FA8	0000 1000	1	0	
SUB CL,1	FA9	0000 1000	0	1	
JZ FAB	FAB	0000 1000	0	0	
HLT	FAC	0000 1000	0	0	El valor en AL es 4 visto en decimal

TAREA 3

A) Investigar.-
Qué acciones realizan las siguientes instrucciones en assembly, con un pequeño ejemplo cada una:

- SHL,
- SHR,
- ROL y
- ROR.

B) Resolver.-
Se tiene el siguiente programa cargado en memoria principal listo para ejecutarse, este programa desplaza el dato cargado en AX 1 bit hacia la derecha, 3 veces.

Ejecuta este programa siguiendo el ciclo de instrucción del procesador y anotando todos los resultados en una tabla. Sugerencia, trabajar el valor de AX en binario (hacer conversiones en tal caso).

VISTO DESDE LA MEMORIA PRINCIPAL (RAM)...		
DIREC (Hex)	**INSTR (Asm)**	**DESCRIPCIÓN**
FA5	MOV AX,80h	Mover 80 (en hex) al registro AX
FA6	MOV CL,3	Mover 3 al registro CL
FA7	SHR AX,1	Desplazar (shift) los bits de AL una posición a la der.
FA8	SUB CL,1	Descontar en 1 a CX
FA9	JZ FAB	Si Z activo saltar a 0xFAB
FAA	JMP FA7	Saltar a 0xFA7 (sin condición)
FAB	HLT	Fin de la ejecución