

Programación de la CPU teórica

Codificación de un algoritmo

En este punto vamos a plantear un problema, a partir del problema construiremos su algoritmo y a partir del algoritmo el programa para nuestra CPU teórica, de forma que se aporte una solución general para el problema.

El problema a resolver es el de calcular el máximo de una serie de números positivos. El valor máximo obtenido se guardará en la posición de memoria indicada por **R4**. Los números están almacenados en memoria consecutivamente a partir de la posición indicada por el contenido del registro **R5**. El total de números a chequear para obtener el máximo son 7. En la figura 1 aparece el estado de la memoria y los registros indicados. Entre paréntesis los valores de los números en decimal.

R5	2078	2078	007F	(127)
		2079	008F	(143)
		207A	000C	(12)
		207B	0025	(37)
		207C	003B	(59)
		207D	00C9	(201)
		207E	000D	(13)
R4	2080	2080		

Figura 1 Estado de la memoria y los registros.

El problema planteado ya se vio al estudiar la arquitectura Von Neumann. Entonces se expresó el algoritmo de resolución y se solucionó mediante una máquina específica. Ahora vamos a ver como se resolvería con una máquina genérica como es nuestro computador teórico.

En primer lugar recordaremos el algoritmo utilizado, era:

```

Inicializar máximo ← 0
para cada número de la lista hacer:
    comparar número con máximo
    Si número > máximo
        asignar máximo ← número
Almacenar máximo
    
```

En el algoritmo podemos encontrar varias fases:

Inicialización.- Son los pasos previos que hay que realizar antes de comenzar a ejecutar el algoritmo.

Ejecución.- Una parte debe ejecutarse para cada número de la tabla: comparar el número con el máximo y según el resultado de esta comparación, se cambiará el máximo o no.

El realizar una cosa u otra es posible mediante las instrucciones de control de flujo, salto condicional en este caso, pues la acción a tomar depende del resultado de una comparación.

Resultados.- Una vez realizada la ejecución del algoritmo se almacenará el valor obtenido en la posición indicada.

En nuestro problema hemos de comparar una serie de números almacenados consecutivamente, normalmente a esta estructura se la denomina tabla o array. ¿Como haríamos para comparar todos los números? o lo que es lo mismo ¿Cómo recorreríamos la tabla?

Existen dos alternativas. Podemos comenzar colocando el número de elementos de la tabla en un registro, y cada vez que procesamos un elemento de la tabla decrementamos el valor del registro en una unidad. Cuando al decrementar alcancemos el valor cero, habremos procesado ya todos los elementos de la tabla.

La segunda alternativa consiste en emplear dos registros, uno se carga con el número de elementos de la tabla y el otro inicializado a cero. Cada vez que se procesa un elemento el registro inicializado a cero se incrementa en una unidad. Cuando los dos registros tengan el mismo valor habremos alcanzado el final de la tabla. (Para saber si los dos registros tienen el mismo valor se puede emplear la operación de comparación).

Se utiliza una u otra alternativa según los casos. En nuestro ejemplo emplearemos la primera alternativa.

De forma simbólica el programa se escribiría:

Inicialización:

- | | |
|--------------------------------------|---|
| 1) (R7) máximo \leftarrow 0 | Inicializa el máximo a 0, R7 almacenará el máximo |
| 2) (R6) contador \leftarrow (7) | Inicializo el registro R6 con el número de elementos de la tabla entre los cuales buscaremos el máximo. |

Ejecución:

- | | |
|---|---|
| IR_OTRO: 3) R0 \leftarrow [R5] | Se trae el número a comparar de memoria. Recordar que los datos están almacenados en memoria a partir de la posición indicada por el registro R5. |
| 4) R7 - R0 | <p>Compara el máximo actual con el valor obtenido de memoria. La comparación en la CPU teórica no genera resultado, aunque si modifica los flags del registro de estado.</p> <p>Los posibles resultados serán:</p> <p>≥ 0; El máximo actual es mayor que el número comparado, no se cambiaría el máximo.</p> <p>< 0; El máximo actual es menor que el número comparado, se reemplazaría el máximo actual con el número.</p> <p>El siguiente paso sería la evaluación de los resultados de la comparación. El flag a observar será el flag de signo.</p> |
| 5) Si SF = 0 IR_SIGUE | <p>Si el flag de signo es 0, SF = 0, el resultado de la resta es positivo, luego no se ejecuta la siguiente instrucción, se saltará a la instrucción 7).</p> <p>IR_SIGUE Se denomina etiqueta y sirve para indicar el destino del salto. Aquí indica a donde debemos saltar.</p> |
| 6) R7 \leftarrow R0 | Si el resultado de la comparación del paso 4) ha |

sido un número negativo, $SF = 1$, no se salta en el paso 5) y se ejecuta esta instrucción que sustituye el máximo actual por el valor con el que se comparaba. En R7 tendremos ahora el nuevo máximo.

IR_SIGUE: 7) $R5 \leftarrow R5 + 1$

En caso de que el signo de la comparación fuera positivo, la instrucción 5) provocaría un salto a esta instrucción. La etiqueta a la izquierda indica que es destino del salto.

En esta instrucción se comienzan a dar los pasos para procesar el siguiente número.

1. Incrementamos el valor de R5, ahora R5 contendrá la dirección del siguiente número a procesar, es decir, $2078 + 1 = \underline{2079} + 1 = \underline{207A}$
....

8) $R6 \leftarrow R6 - 1$

2. Hemos comparado el máximo con un número, luego restamos 1 al contador de elementos a procesar y que habíamos inicializado con el número de elementos de la tabla.

9) Si no fin tab. **IR_OTRO:**

3. Hemos de comprobar si ya hemos comparado el máximo con todos los números. En caso de que no sea así debemos continuar con el siguiente número, y en caso afirmativo se procede a almacenar el máximo obtenido.

Para comprobar si hemos procesado todos los números hacemos lo siguiente: En la instrucción 8), si al decrementar en 1 el registro R6 éste toma el valor cero, indicará que no quedan elementos por procesar. Que una operación dé cero como resultado activará el flag de cero, $ZF = 1$. Comprobando en el salto el estado de este flag sabemos si hemos acabado o hemos de repetir el proceso para otro número.

Si el resultado de la operación no es cero y hemos de repetir el proceso, la etiqueta **IR_OTRO** en la instrucción 3), nos indica el destino del salto a realizar.

Observar que en el nuevo ciclo R5 tendrá la dirección del siguiente dato a procesar por la instrucción 7) y todo funcionaría correctamente.

Almacenamiento:

10) $[R4] \leftarrow R7$

Si se ha alcanzado el final de la tabla, es decir, se ha comparado el máximo con todos los números, se procede a guardar el máximo obtenido en la dirección indicada por el registro R4.

Codificación en Mnemónicos

Codificando la representación anterior del algoritmo con los Mnemónicos empleados en nuestra CPU elemental quedará:

	1)	XOR	R7, R7, R7	Inicializo máximo a 0
	2)	MOVL	R6, 07H	Inicializamos el registro con el número de datos
		MOVH	R6, 00H	a comparar, 0007
IR_OTRO:	3)	MOV	R0, [R5]	Carga de valor desde memoria
	4)	COMP	R7, R0	Comparación
	5)	BRNS	IR_SIGUE	Si el signo es positivo (SF = 0) salta
	6)	MOV	R7, R0	Actualiza el máximo
IR_SIGUE:	7)	INC	R5	Incrementa el valor de R5
	8)	DEC	R6	Decrementa el valor de R6
	9)	BRNZ	IR_OTRO	Si R6 es distinto de cero (ZF = 0), otro número.
	10)	MOV	[R4], R7	Almacena el resultado
		FIN		

Antes de pasar el código de Mnemónicos a código de instrucción, hemos de resolver los saltos, es decir, los valores numéricos por los que deben sustituirse las etiquetas **IR_OTRO** e **IR_SIGUE**.

Comencemos con la etiqueta **IR_SIGUE**. En la instrucción 5) se comprueba el flag de signo. Si el signo es positivo (SF = 0), o lo que es lo mismo el máximo actual es mayor que el número. En este caso no se producirá cambio en el valor del máximo y se salta a la instrucción 7). Es decir, saltamos dos instrucciones, sin embargo, ha de recordarse que el **contador de programa** o **PC** apunta en todo momento a la siguiente instrucción a la que se está ejecutando. En este caso el **PC** estará apuntando a la instrucción 6), en realidad sólo existe un paso de diferencia, el valor que tomará **IR_SIGUE** es 1. $PC = PC + 1$ y alcanzaremos la nueva instrucción.

Para la etiqueta **IR_OTRO**. En la instrucción 9) se comprueba si al decrementar R6 éste toma el valor cero, si es así no se salta. En caso contrario, aún quedan números por comparar, se saltará a la instrucción 3). Es decir, se retroceden 6 instrucciones. Sin embargo, al igual que en el caso anterior, el **contador de programa** o **PC** está apuntando a la siguiente instrucción a la que se está ejecutando, en este caso la 10). Por tanto, en realidad hemos de retroceder una instrucción más, es decir, 7. **IR_OTRO** tomará el valor -7 pues el salto es hacia atrás, que expresado en complemento a dos y con 8 bits es: 1111 1001 \Rightarrow F9H. $PC = PC + FFF9...$ (incluida la extensión de signo)

Resumiendo, cuando el salto es hacia adelante el **PC** se incrementa en una unidad menos, mientras que cuando el salto es hacia atrás, el **PC** debe decrementarse en una unidad más. Esto se debe a la posición del **PC** una instrucción adelantada.

Codificación en código de instrucción.

Siguiendo la tabla de codificación del juego de instrucciones de nuestra CPU teórica, el algoritmo anterior escrito en Mnemónicos se codificaría en código de instrucción de la siguiente forma:

	1)	XOR	R7, R7, R7	01 100 111 111 111 00
	2)	MOVL	R6, 07H	00 100 110 00000111
		MOVH	R6, 00H	00 101 110 00000000
IR_OTRO:	3)	MOV	R0, [R5]	00 010 000 101 00000
	4)	COMP	R7, R0	01 101 111 000 000 00
	5)	BRNS	IR_SIGUE	11 110 <u>111</u> 00000001 \Rightarrow cond. (SF = 0)
	6)	MOV	R7, R0	00 001 111 000 00000
IR_SIGUE:	7)	INC	R5	100 01 101 00000000
	8)	DEC	R6	100 10 110 00000000
	9)	BRNZ	IR_OTRO	11 110 <u>101</u> 11111001 \Rightarrow cond. (ZF = 0)
	10)	MOV	[R4], R7	00 011 100 111 00000
		FIN		

206B			
206C	67FC		
206D	2607		
206E	2E00		
206F	10A0		
2070	6F00		
2071	F701		
2072	0F00		
2073	8D00		
2074	9600		
2075	F5F9		
2076	1CE0		
2077	PARAR		
2078	007F	(127)	
2079	008F	(143)	
207A	000C	(12)	
207B	0025	(37)	
207C	003B	(59)	
207D	00C9	(201)	
207E	000D	(13)	
2080			

PROGRAMA

DATOS

RESULTADO

Figura 2 Contenido de la memoria después de cargar el programa

Si los códigos de instrucción obtenidos se colocan en memoria a partir de la posición 206C y se expresan en hexadecimal nos quedará la memoria como se puede ver en la figura 2.

A partir de este momento el programa podría ejecutarse en la CPU instrucción por instrucción. Cada instrucción seguiría las fases de, búsqueda de la instrucción, incremento del PC, interpretación de la instrucción y ejecución de la misma.

La unidad de control generará las señales adecuadas para llevar a cabo todos los pasos, los tres primeros pasos son comunes a todas las instrucciones y por tanto las señales de control serán las mismas para todas las instrucciones. Las señales generadas para la ejecución de la instrucción, fase cuatro, dependerá de la instrucción a ejecutar

Conceptos de lenguajes de programación, compilador y código máquina.

Normalmente a la representación de un algoritmo mediante símbolos o expresiones, conducentes a su resolución mediante una máquina de aplicación genérica se le denomina programa.

Lenguaje de Programación

Es un conjunto de reglas semánticas y sintácticas diseñadas para definir estructuras de datos y algoritmos, elementos que forman los programas que deben ser ejecutados por un computador.

- **Reglas semánticas:** ¿Qué se puede hacer con el lenguaje? Son las diferentes construcciones que se pueden hacer con el lenguaje, bucles, asignaciones, bifurcaciones condicionales, etc.
- **Reglas sintácticas:** ¿Cómo se deben expresar las reglas del lenguaje? Es decir, como debe expresarse cada instrucción de forma que sea comprensible, hace referencia a la forma de los Mnemónicos en el caso de la CPU teórica. Sería la ortografía del lenguaje de programación.

Cuando las órdenes o las instrucciones expresadas en el lenguaje hacen referencia directamente a instrucciones o elementos del procesador (P.ej. los registros), se denomina a este tipo de lenguaje **ensamblador**. Si esto no es así se dice que el lenguaje es de **alto nivel**, (P.ej. Pascal, C, etc.)

Al texto escrito según las normas de un lenguaje ensamblador o de un lenguaje de alto nivel se le denomina **programa fuente**.

En nuestro ejemplo el algoritmo de cálculo del máximo se ha representado mediante un programa que calcula el máximo, la representación se ha realizado primeramente en un lenguaje simbólico mediante los Mnemónicos de la CPU teórica, éste es un lenguaje de programación. Como en este lenguaje se hace referencia a los registros se denomina lenguaje ensamblador y al conjunto del programa, programa fuente.

Código Máquina

Se llama código máquina al conjunto de secuencias de bits que pueden ser interpretadas como instrucciones por la CPU de un computador.

Cuando se ha codificado nuestro programa en el código de instrucciones, lo que se ha realizado es la obtención del código máquina. Es decir, se obtienen las secuencias de bits que llegarían a la CPU, que ésta decodificaría y ejecutaría para realizar el programa.

Compilador

Al paso de transferir el programa fuente, escrito por nosotros, a código máquina, entendible por la máquina, se le llama compilación. El elemento que realiza la transferencia o compilación se denomina compilador. En nuestro ejemplo somos nosotros los que pasamos los Mnemónicos a código de instrucciones luego nosotros actuamos como compilador.

Lo normal es que un compilador sea un programa capaz de generar a partir de un programa fuente escrito en un determinado lenguaje (bien sea ensamblador o de alto nivel) código máquina para un determinado procesador.

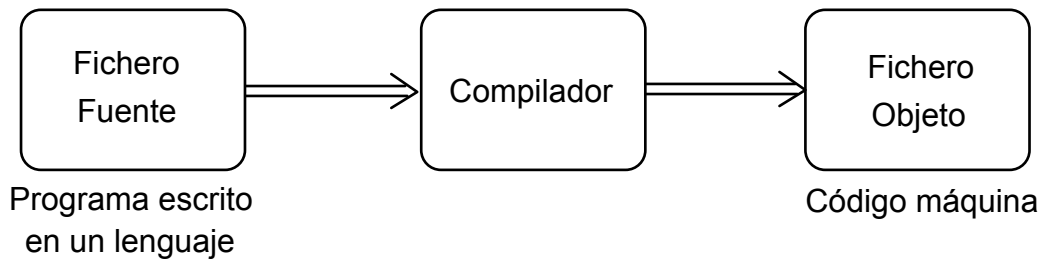


Figura 3 Modo de trabajo de un compilador