6- Estudio de un S.E.T.I. básico I (Modelo de programación)

BIBLIOGRAFÍA

[Eck 95]

David J. Eck,

"The Most Complex Machine",

A. K. Peters.

german@we.lc.ehu.es
http://158.227.6.118:8080/german

Registros internos

Acumulador (16 bits)

Flag (1 bit)

Formato de instrucción

I_5	I_4	I_3	I_2	I_1	I_0	10 bits (datos)

operación	*dir. directo*	*dir. inmediato*	*dir. indirecto*
Sumar al AC	000000 0 ADD	010000 16 ADD-C	100000 32 ADD-I
Restar del DC	000001 1 SUB	010001 17 SUB-C	100001 33 SUB-I
Y lógico con el AC	000010 2 AND	010010 18 AND-C	100010 34 AND-I
O lógico con el AC	000011 3 OR	010011 19 OR-C	100011 35 OR-I
NO lógico del AC	000100 4 NOT		
Desplazar a izquierda el AC	000101 5 SHL		
Desplazar a derecha el AC	000110 6 SHR		
Incrementar el AC	000111 7 INC		
Decrementar el AC	001000 8 DEC		
Cargar AC de memoria	001001 9 LOD	011001 25 LOD-C	101001 41 LOD-I
Descargar AC en memoria	001010 10 STO		101010 42 STO-I
Parar	001011 11 HLT		
Saltar	001100 12 JMP		101100 44 JMP-I
Saltar si AC=0	001101 13 JMZ		101101 45 JMZ-I
Saltar si AC<0	001110 14 JMN		101110 46 JMN-I
Saltar si FLAG activado	001111 15 JMF		101111 47 JMF-I

Programa ejemplo: $\sum_{i=100}^{1} i$

Con etiquetas			Con direcciones			Código Máquina	
	LOD-C	100	0	LOD-C	100	0	0110010001100100
	STO	i	1	STO	13	1	0010100000001101
	LOD-C	0	2	LOD-C	0	2	0110010000000000
	STO	Suma	3	STO	14	3	0010100000001110
Ciclo:	LOD	Suma	4	LOD	14	4	0010010000001110
	ADD	i	5	ADD	13	5	0000000000001101
	STO	Suma	6	STO	14	6	0010100000001110
	LOD	i	7	LOD	13	7	0010010000001101
	DEC		8	DEC		8	0010000000000000
	JMZ	Fin	9	JMZ	12	9	0011010000001100
	STO	i	10	STO	13	10	0010100000001101
	JMP	Ciclo	11	JMP	4	11	0011000000000100
Fin:	HLT		12	HLT		12	0010110000000000
i:	DATA		13			13	0000000000000000
Suma:	DATA		14			14	0000000000000000

Cuestiones/ejercicios

1. Las instrucciones se codifican con 6 bits, de modo que hay 64 códigos posibles. No obstante sólo disponemos de 31 instrucciones, por lo que hay 33 códigos que no tienen un significado para la máquina. ¿Que deberá hacer ésta en caso de encontrarse con uno de tales códigos?.
2. Podemos almacenar caracteres ASCII en memoria en formato “empaquetado” situando dos caracteres en cada posición. No obstante para su manipulación puede ser más cómodo tener uno en cada dirección con los 8 bits más altos a cero. Escribir dos rutinas para realizar las conversiones en uno y otro sentido para estos dos formatos.
3. Escribir una rutina para multiplicar dos números naturales (no es preciso controlar las situaciones de “rebose” -overflow- del resultado).