Apellidos y Nombre: .....Grupo:.....DNI: .....

Examen 4 (temas 12, 13 y 14)

- Duración del examen: 2 horas. La solución de cada ejercicio se tiene que escribir en el espacio reservado para ello en el propio enunciado.
- No podéis utilizar calculadora, móvil, apuntes, etc. En hoja aparte se os da una "chuleta" con información útil para realizar los ejercicios.
- La solución del examen se publicará en Atenea mañana por la tarde y las notas antes del próximo lunes noche. En general, para revisar la nota de este examen debéis enviar un email a vuestro profesor de teoría el próximo miércoles, excepto que el profesor indique otra cosa.

Ejercicio 1 **(3 puntos)**

El programa en ensamblador, una vez completado, se traducirá a lenguaje máquina para ser ejecutado en el SISC Von Neumann situando la sección `.text` a partir de la dirección 512 de memoria y a continuación la sección `.data`. En esta implementación el puerto de datos de la impresora (con efecto lateral sobre el de estado) es el 30 y el de estado el 20. Puede ser que en el programa haya más etiquetas de las necesarias.

```
.data
        N=0x0006; 0<N<23
V1:     .byte 50,200,142,62,160,71
        .space 16
        .even
V2:     .byte 42,128,127,255,130,30
        .space 16
.text
L0:     MOVI    R7,  N
        MOVI    R1,
        MOVHI   R1,
        ADDI    R2, R1,
L1:     L       R3,       (R1)
        L       R4,       (R2)
L2:     IN      R6,
        B       ,
L3:     CMPLT   R5,  R3,  R4
        B       R5,
        OUT     ,   R3
        B       R5,
L4:     OUT     ,   R4
L5:     A       R1,  R1,
        A       R2,  R2,
        A       R7,  R7,
L6:     B       R7,
.end
```

a) Completad la escritura del programa para que imprima el mayor entre `V1[k]` y `V2[k]` para `k=0, 1,... N-1`, considerando los elementos de `V1` y `V2` como números **enteros** codificados en complemento a dos con **8 bits**. (Los elementos de los vectores se han definido como naturales en decimal y son interpretados como enteros en Ca2). Los elementos `V1[0]` y `V2[0]` se encuentran almacenados en memoria en las direcciones simbólicas `V1` y `V2` respectivamente. Solo se procesan los `N` primeros elementos de los vectores, a razón de uno por iteración. El programa esta escrito para `0<N<23`: (**1 punto**)

b) Una vez cargado el programa en memoria:

- ¿A qué dirección de memoria corresponde cada una de las siguientes etiquetas, o direcciones simbólicas, y qué palabra (word) contiene? (**0.75 puntos**)

`V2:`	=>	Mem_w[0x	] = 0x
`L0:`	=>	Mem_w[0x	] = 0x
`L3:`	=>	Mem_w[0x	] = 0x

- Una vez ejecutado el programa en el computador SISC Von Neumann ¿Cuál es la secuencia de los 3 primeros datos impresos, expresados en hexadecimal (el primero os lo damos nosotros)? (**0,5 puntos**)

0x0032, ,

c) Responder considerando que cada vez que se termina de ejecutar la instrucción `L2` el registro `R6` contiene un 1 y que la mitad de las veces que se ejecuta L3 el resultado es verdadero. ¿Cuántas instrucciones se ejecutan? ¿Cuánto tarda en ejecutarse el código en el Harvard uniciclo y en el Von Neumann suponiendo que los tiempos de ciclo son 2.000 y 1.000 u.t. respectivamente? ¿Cuánto vale x para que sea cierta la siguiente afirmación? "El computador Von Neumann es un x% **más lento** que el Harvard uniciclo, ejecutando este código.(**0,75 puntos**)

InstrucEjec = ; Tejec(Harvard uniciclo) = ; Tejec(V.Neumann) = ; x =

Ejercicio 2 **(1 punto)**

Cada uno de los apartados pregunta sobre un ciclo concreto de la ejecución de una instrucción en el SISC Von Neumann. Escribid el valor de los bits de la **palabra de control** que genera el bloque **SISC CONTROL UNIT** durante el ciclo a que hace referencia cada apartado. **Poned x siempre que no se pueda saber el valor de un bit** (ya que no sabemos cómo se han implementado las x en la ROM_OUT). Para cada apartado/fila se indica el nodo/estado de la UC en ese ciclo y la instrucción (en ensamblador) que está almacenada en el IR en ese ciclo. Podéis ver el grafo de estados de Moore de la UC en el anexo. Suponed que el contenido de todos los registros, `Rk` para k=0,...7, antes de ejecutarse cada instrucción es 0.

Apartado	Nodo/Estado (Mnemo Salida)	Instrucción en IR (en ensamblador)	Palabra de Control																		
			@A	@B	Pc/Rx	Ry/N	OP	F	P/I/L/A	@D	WrD	Wr-Out	Rd-In	Wr-Mem	Ldlr	LdPc	Byte	Alu/R@	R@/Pc	N (hexa)	ADDR-IO (hexa)
a	D	`STB 17(R5), R6`																			
b	Bnz	`BNZ R7, -15`																			
c	Addr	`LDB R0,-3(R4)`																			
d	Movi	`MOVI R6, 0x90`																			

Escribid el contenido del IR en hexadecimal para cada apartado/fila de la tabla anterior:

a) IR=0x , b) IR=0x , c) IR=0x , d) IR=0x

Ejercicio 3 **(1,25 puntos)**

Completad la tabla que representa en forma compacta parte del contenido de la ROM_OUT de la unidad de control del SISC Von Neumann. Poned x siempre que un bit pueda valer tanto 0 como 1.

@ROM	Bz	Ldlr	R@/Pc	Alu/R@	Pc/Rx	Ry/N	MxN1	MxN0	MxF	Mx@D1	Mx@D0	
7												St
8												Ldb
11												Bz
15												In

Ejercicio 4 **(0,75 puntos)**

Indicad qué cambios se producen en el estado del computador SISC Von Neumann después de ejecutar cada una de las instrucciones de la tabla suponiendo que antes de ejecutarse cada una de ellas el PC vale `0xF0F8`, el contenido de todos los registros pares es 0x6799 y de los impares es 0x7FFF y que el byte contenido en todas las direcciones pares de la memoria es 0x87 y el de todas las impares es 0x78. Utilizad la notación $MEM_b[\ldots]$=... y/o $MEM_w[\ldots]$=... para indicar los cambios en la memoria.

Instrucción a ejecutar	Cambios en el estado del computador
`LDB R0, -3(R4)`	
`MOVI R2, 0x93`	
`STB 0x27(R3),R6`	

Ejercicio 5 **(0,5 puntos)**

Completad la dirección o contenido, según corresponda, de la ROM_Q+ del SISC Von Neumann.

ROM_Q+ [0x0AB] = 0x , ROM-Q+ [0x] = 0x0E

Ejercicio 6 **(3,5 puntos)**

Completad el diseño del SISC Von Neumann para que pueda ejecutar, además de las 25 instrucciones originales SISA, la nueva instrucción `BLT`, que tiene el formato y codificación, la sintaxis ensamblador y la semántica siguientes:

```
Codificación:  1111 aaa bbb nnnnnn
Sintaxis:      BLT Ra, Rb, N6
Semántica:     PC=PC+2; if(Ra_s<Rb_s)PC=PC+SE(N6)*2
```

La nueva instrucción se puede usar para hacer algo “equivalente” a lo que se hace ejecutando el siguiente código SISA:

```
CMPLT Rd, Ra, Rb
BNZ   Rd, N8
```

La ventaja de usar la nueva instrucción, `BLT Ra, Rb, N6`, es que no se requiere usar el registro temporal `Rd`, mientras que la desventaja es que la dirección destino de salto tomado no puede estar tan alejada de la dirección de la instrucción de salto como lo

puede estar usando las dos instrucciones del lenguaje SISA original. Vamos a proponer dos implementaciones de BLT (que son independientes, como si fueran dos ejercicios distintos).

Apartado A. (1,25 puntos)

En esta primera implementación, además de modificar el contenido de la ROM_OUT y de la ROM_Q+, cosa imprescindible para añadir una nueva instrucción, se le añade a la unidad de control un MUX-2-1 (con señal de selección Mx, que genera la ROM_OUT, y que solamente valdrá 1 en alguna de las fases de ejecución de BLT). La ejecución de BLT solamente requiere 4 ciclos de ejecución, nodos F, D, Blt1 y Blt2. Se pide:

a) Indica claramente, con texto y/o con un dibujo, donde se conectan las entradas de datos 0 y 1 del MUX-2-1 y donde se conecta su salida, para el correcto funcionamiento del computador con la nueva instrucción BLT. Si se responde incorrectamente este apartado, el resto del Apartado A se considerará incorrecto. **(0,25 puntos)**

b) Completad el contenido de la tabla que indica, mediante una fila para cada nodo, la acción o acciones en paralelo que se realiza en el computador en cada uno de los ciclos/nodos que requiere la ejecución, propiamente dicha, de la nueva instrucción (Blt1 y Blt2). Usad el mismo lenguaje de transferencia de registros que en la documentación. **(0,5 puntos)**

Nodo/Estado		
Número	Mnem.	Acciones
18	Blt1	
19	Blt2	

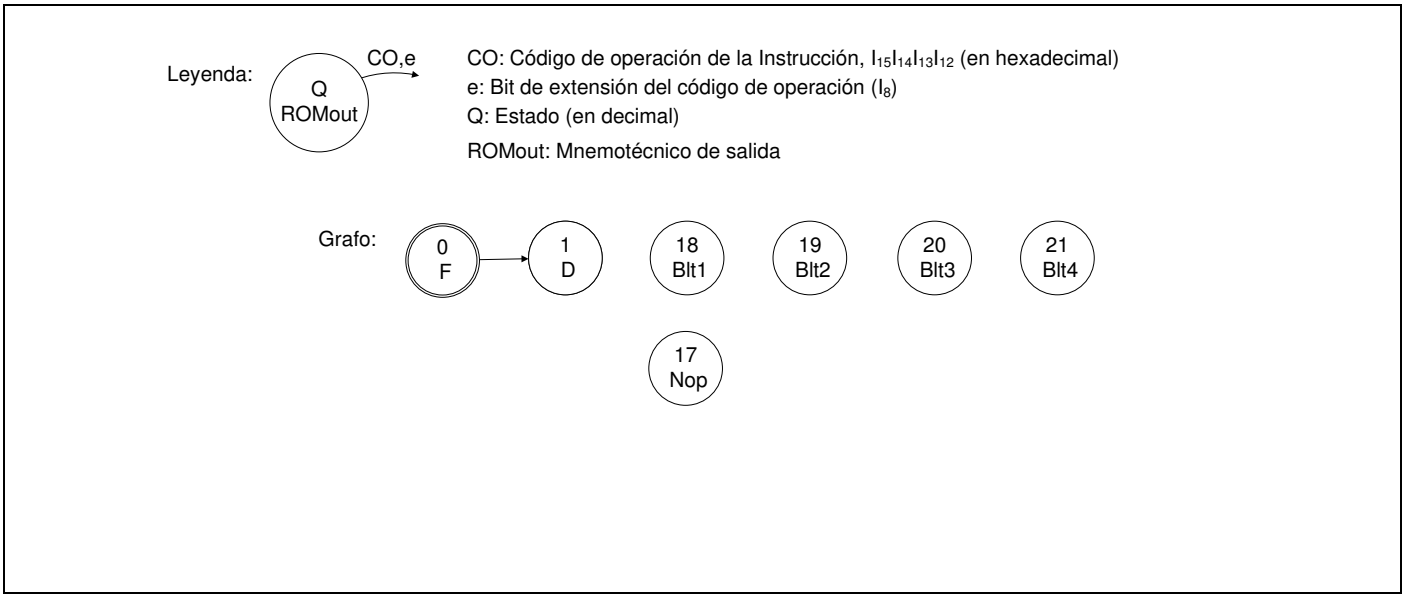
c) Completad (poniendo 0, 1 o x en cada bit) las dos filas de la tabla que especifican el contenido de la ROM_OUT para las direcciones 18 (Blt1) y 19 (Blt2) (**0,5 puntos**)

@ROM	Mx	Bnz	Bz	WrMem	RdIn	WrOut	WrD	LdIr	Byte	R@/Pc	Alu/R@	Pc/Rx	Ry/N	P/I/L/A1	P/I/L/A0	OP1	OP0	MxN1	MxN0	MxF	F2	F1	F0	Mx@D1	Mx@D0	
18																										Blt1
19																										Blt2

Apartado B. (2,25 puntos)

En esta segunda implementación solamente se puede modificar el contenido de la ROM_Q+ y de la ROM_OUT de la unidad de control (no se añade ningún hardware más). La ejecución de la nueva instrucción tarda 6 ciclos (F, D, Blt1, Blt2, Blt3 y Blt4). Se pide:

a) Completad el fragmento del grafo de estados del circuito secuencial de la unidad de control necesario para ejecutar completamente la nueva instrucción BLT y todas las que tienen un código ilegal (código de operación de instrucciones que no están definidas). Se da la leyenda del grafo y todos los nodos necesarios. Faltan algunos arcos, así como las etiquetas de los arcos. Los nuevos nodos/estados se codifican en decimal como 18, 19, 20 y 21 y los mnemotécnicos de sus salidas son Blt1, Blt2... Dibujad todos los nodos, arcos y etiquetas que faltan. **(0,25 puntos)**



b) Completad el contenido de la tabla que indica, mediante una fila para cada nodo, la acción o acciones en paralelo que se realiza en el computador en cada uno de los ciclos/nodos que requiere la ejecución de la nueva instrucción (F, D, Blt1, Blt2...) Para especificar las acciones usad el mismo lenguaje de transferencia de registros que en la documentación. **(1 punto)**

Nodo/Estado		
Número	Mnem.	Acciones
0	F	
1	D	
18	Blt1	`R@ ← PC+SE(N6)  //  RX ← Ra  //  RY ← Rb`
19	Blt2	
20	Blt3	
21	Blt4	

c) Completad (poniendo 0, 1 o x en cada bit) las filas F, D, Blt2, Blt3 y Blt4 de la tabla que especifica parte del contenido de la ROM_OUT, poniendo el máximo número de x posibles. La dirección 0 de la ROM corresponde al estado 0 (F), la 1 al 1 (D), la dirección 19 al estado 19 (Blt2)... (Las fila 18 (Blt1) esta ensombrecida para que no la completéis y así poder responder el apartado en menos tiempo). (**1 punto**)

@ROM	Bnz	Bz	WrMem	RdIn	WrOut	WrD	Ldlr	Byte	R@/Pc	Alu/R@	Pc/Rx	Ry/N	P/I/L/A1	P/I/L/A0	OP1	OP0	MxN1	MxN0	MxF	F2	F1	F0	Mx@D1	Mx@D0	
0																									F
1																									D
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
18																									Blt1
19																									Blt2
20																									Blt3
21																									Blt4