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 antes del miércoles 24 a la noche, excepto que el profesor indique otra cosa.

Ejercicio 1 **(3 puntos)**

El programa ensamblador de la derecha se ha traducido a lenguaje máquina para ser ejecutado en el SISC Von Neumann, situando la sección `.text` a partir de la dirección 0x8000 de memoria y a continuación la sección `.data`.

```
.data
     N = 5
V:   .word 7,3,7890,750,67
     .word 43,64,32,0,7
     Uno=1
     Tres=3
W:   .space 10,0
     .even
     .space 10,0
.text
     MOVI   R0,N
     MOVI   R1, lo(V)
     MOVHI  R1, hi(V)
     MOVI   R2, Uno
     MOVI   R3, Tres
L:   LD     R4, 0(R1)
     SHL    R5, R4, R3
     SHL    R6, R4, R2
     ADD    R4, R5, R6
     ST     0x14(R1), R4
     ADDI   R1,R1,2
     ADDI   R0,R0,-1
     BNZ    R0, L
```

a) Una vez cargado el programa en memoria:

- ¿A qué dirección de memoria corresponde la etiqueta, o dirección simbólica, `W`? (**0,25 puntos**)

`W = 0x`

- ¿Cuál es la dirección de memoria y su contenido donde han quedado almacenadas cada una de las siguientes instrucciones? (**0,75 puntos**)

`MOVI R1, lo(V)` => Mem_w[0x] = 0x

`SHL R5, R4, R3` => Mem_w[0x] = 0x

`BNZ R0, L` => Mem_w[0x] = 0x

b) Una vez ejecutado el programa en el computador SISC Von Neumann ¿Cuál es la dirección de memoria donde ha escrito por última vez la instrucción `ST 0x14(R1),R4` y cuál es su contenido? (**1 punto**)

Mem_w[0x] = 0x

c) ¿Cuánto tarda en ejecutarse el programa en las tres versiones de los computadores SISC? Suponed que se han fabricado los computadores con una tecnología tal que el tiempo de ciclo del Harvard uniciclo, el Harvard multiciclo y el Von Neumann es de 4.000, 1.000 y 1.500 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.(**1 punto**).

Tejec(Harv.uni.) = ; Tejec(Harv.multi.) = ; Tejec(V.Neumann) = ; x =

Ejercicio 2 **(2 puntos)**

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 se refiere cada apartado (Es conveniente que paséis el contenido del IR de ensamblador a binario). **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 la UC en el anexo. Suponed que el contenido de todos los registros durante el ciclo a que hace referencia cada apartado es 0xFADO.

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	Movhi	`MOVHI R3,0x8F`																			
b	D	`BNZ R7,-4`																			
c	Addi	`ADDI R1,R3,-8`																			
d	Stb	`STB 0x20(R3),R1`																			

Ejercicio 3 **(1,5 puntos)**

Completad las filas y columnas SIN SOMBREAR en la siguiente tabla que representa en forma compacta el 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. Tened en cuenta que cuando se ejecuta una instrucción con un código de operación que no coincide con ninguno del SISA no se debe modificar el estado del computador, excepto el PC que se debe incrementar adecuadamente para pasar a ejecutar la siguiente instrucción (podemos decir que estos códigos de operación no usados en ninguna de las 25 instrucciones SISA codifican instrucciones NOP).

@ROM	Bnz	Bz	WrMem	RdIn	WrOut	WrD	LdIr	Byte	R@/Pc	Alu/R@	Pc/Rx	Ry/N	P/IL/A1	P/IL/A0	OP1	OP0	MxN1	MxN0	MxF	F2	F1	F0	Mx@D1	Mx@D0	
0																									F
1																									D
2																									Al
3																									Cmp
4																									Addi
5																									Addr
6																									Ld
7																									St
8																									Ldb
9																									Stb
10																									Jalr
11																									Bz
12																									Bnz
13																									Movi
14																									Movhi
15																									In
16																									Out
17..31																									Nop

Ejercicio 4 **(0,5 punto)**

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 0xCAFE, el contenido de todos los registros es 0xDAD0 y que el contenido de todas las direcciones pares de la memoria es 0xCE y el de todas las impares es 0xDA. Utilizad el mnemotécnico $MEM_b[\ldots]$=... y/o $MEM_w[\ldots]$=... para indicar los cambios en la memoria.

Instrucción a ejecutar	Cambios en el estado del computador
`LDB R7, 0x3F(R5)`	
`STB -7(R7), R6`	

Ejercicio 5 **(3 puntos)**

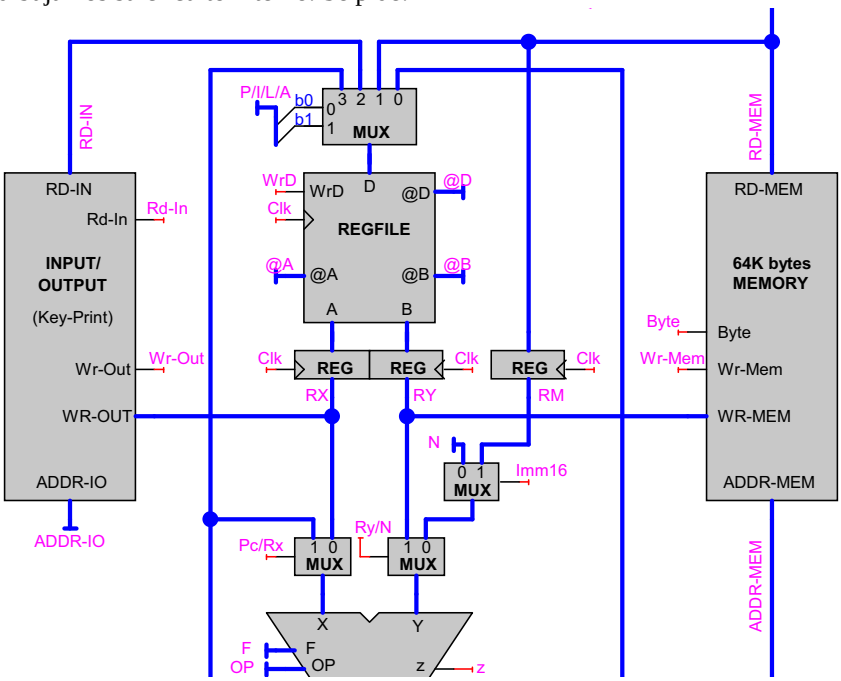
Completad el diseño del SISC Von Neumann para que pueda ejecutar, además de las 25 instrucciones originales SISA, las nuevas 7 instrucciones Aritmético-Lógicas y las 5 de Comparación en las que el segundo operando fuente es un inmediato de 16 bits. La instrucción NOT no tiene segundo operando por lo que no existe la NOT con inmediato de 16 bits. Cada nueva instrucción ocupa dos palabras consecutivas en memoria, la de dirección @, donde se encuentran los 16 bits de más peso de la instrucción y la de dirección @+2 donde se encuentran los 16 bits del inmediato (N16 = nnnnnnnnnnnnnnnn). Una vez ejecutada la instrucción el PC debe quedar incrementado en 4 para apuntar a la siguiente instrucción en secuencia. El nuevo formato y codificación, la sintaxis ensamblador y la semántica de las nuevas instrucciones es la siguiente:

`Codificación:` `1011 aaa e xx ddd fff nnnnnnnnnnnnnnn`
con `e` = 0 para las operaciones `AL` y con `e` = 1 para las `CMP`. El campo `fff` codifica la operación a realizar de la misma manera que se codifica para las instrucciones `AL` y `CMP` originales

`Sintaxis:` `mnemoI16 Rd, Ra, N16`
siendo `mnemo` cualquiera de los mnemotécnicos de las operaciones `AL` o `CMP` originales.
Ejemplo de instrucción `AL`: `SUBI16 Rd, Ra, N16`. Ejemplo `CMP`: `CMPLEUI16 Rd, Ra, N16`.

Semántica: `Rd = Ra op N16`
siendo `op` la operación AL o CMP que corresponde al `mnemo` (de la instrucción en ensamblador) o al campo `fff` (de la instrucción en lenguaje máquina).

Para poder ejecutar la nueva instrucción se ha modificado la unidad de proceso, UP, y la unidad de control, UC. En la UP se ha añadido un camino que conecta el bus RD-MEM con la entrada 0 del MUX-2-1 con señal de selección Ry/N a través de un nuevo registro, llamado RM, y de un nuevo MUX-2-1, con señal de selección Imm16, como se muestra en la figura. Ahora, el bus N, que genera la UC, llega a la ALU a través de estos dos multiplexores (cuándo Imm16=0 y Ry/N=0).En la UC solo se requiere modificar el contenido de algunas palabras de la ROM_Q+ (ya que el nuevo grafo de la UC tiene tres nuevos estados) y de la ROM_OUT (que hora tiene un bit más de salida, Imm16, que forma parte de la nueva palabra de control). No ha hecho falta añadir ninguna nueva lógica en la UC, por lo que no dibujamos su circuito interno. Se pide:



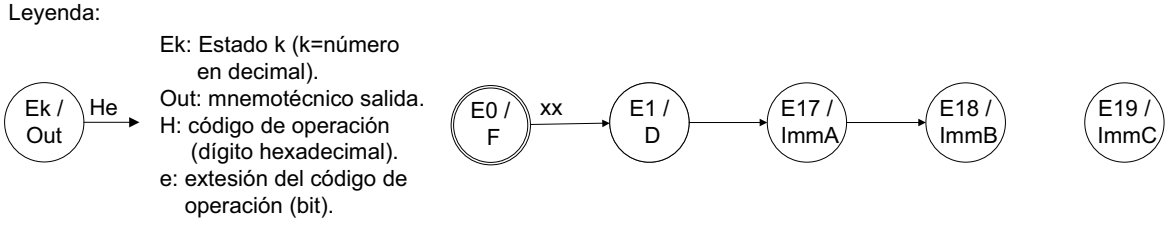
a) Completad el contenido de las cajas vacías de la siguiente tabla que indica, mediante una fila para cada nodo del grafo de estados de la unidad de control, la acción (o acciones en paralelo) que se realiza en el computador en cada uno de los nodos que se requieren para ejecutar las nuevas instrucciones (Fetch, Decode, y los 3 nodos nuevos): F, D, ImmA, ImmB e ImmC. Para especificar las acciones se usa el mismo lenguaje de transferencia de registros que en la documentación (con el que ya hemos especificado algunas acciones). (**1 punto**)

Nodo/Estado		Acciones
Número	Mnem.	
E0	F	`IR ← MEMw[PC]` //
E1	D	// `RX ← Ra` // `RY ← Rb`
E17	ImmA	// // `RX ← Ra`
E18	ImmB	`Rd ←`
E19	ImmC	`Rd ←`

b) Completad (poniendo 0, 1 o x en cada bit) la columna Imm16 y las 3 filas sin sombrear de la tabla que especifica el contenido de la ROM_OUT para que se ejecuten correctamente todas las instrucciones, poniendo el máximo número de x posibles. La dirección 0 de la ROM corresponde al estado E0 (F), la 1 al E1 (D), la dirección 17 al estado E17 (ImmA), etc. (**1 punto**)

@ROM	Imm16	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	Node
0																										F
1																										D
2																										Al
3																										Cmp
4																										Addi
5																										Addr
6																										Ld
7																										St
8																										Ldb
9																										Stb
10																										Jalr
11																										Bz
12																										Bnz
13																										Movi
14																										Movhi
15																										In
16																										Out
17																										ImmA
18																										ImmB
19																										ImmC
20..31																										Nop

c) Completad el fragmento del grafo de estados del circuito secuencial de la unidad de control necesario para ejecutar completamente las nuevas instrucciones. Se da la leyenda del grafo y todos los nodos, pero faltan arcos etiquetas. Dibujad todos los arcos que faltan y todas las etiquetas. No os pedimos que dibujéis el nodo Nop. (**0,5 puntos**)



d) Indicad la dirección o las direcciones (en binario con x cuando sea posible para referirnos a más de una dirección) de la memoria ROM_Q+ y su contenido (en hexadecimal) para implementar correctamente el paso del nodo/estado E1 (D) al E17 (ImmA) y del E17 ((ImmA) al E18 (ImmB). (**0,5 puntos**)

Arco del nodo E1 (D) al E17 (ImmA):

Arco del nodo E17 (ImmA) al E18 (ImmB):

