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 jueves noche. En general, para revisar la nota de este examen debéis enviar un email a vuestro profesor de teoría el próximo viernes, 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 0x0100 de memoria y a continuación la sección `.data`.

```
.data
     N = 6; tiene que ser <=
          ; que 10 y > que 0.
A:   .byte 10
     .even
V1:  .word 2,-5,264,-63,23
     .word 58,-64,32,0,-7
     .even
V2:  .space 20
.text
L1:  MOVI   R0, lo(V1)
     MOVHI  R0, hi(V1)
     MOVI   R1, N
     MOVI   R2, 0
L2:  LD     R7, 0(R0)
     ADD    R2, R2, R7
     ADDI   R1,R1,-1
     ADDI   R0,R0,2
     BNZ    R1, L2
L3:  MOVI   R7, lo(V2)
     MOVHI  R7, hi(V2)
     ST     4(R7), R2
.end
```

a) Una vez cargado el programa en memoria:

- ¿A qué dirección de memoria corresponde la etiqueta, o dirección simbólica, `V2`? (**0,25 puntos**)

`V2 = 0x`

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

`MOVI R0, lo(V1)`	=>	Mem_w[0x	] = 0x
`LD R7, 0(R0)`	=>	Mem_w[0x	] = 0x
`BNZ R1, L2`	=>	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 la instrucción (`ST 4(R7),R2`) y cuál es su contenido? (**1 punto**)

Mem_w[0x] = 0x

c) ¿Cuánto tarda en ejecutarse el código 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 3.500, 900 y 1.500 u.t. respectivamente.

¿Cuánto vale x para que sea cierta la siguiente afirmación? "El computador Harvard multiciclo es un x% **más rápido** que el Von Neumann, ejecutando este código.(**1 punto**).

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

Ejercicio 2 **(1,5 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 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, para responder al apartado b, que el contenido de `R1` antes de ejecutarse la instrucción `BNZ R1, -58` era 0xCAFE.

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	LdIr	LdPc	Byte	Alu/R@	R@/Pc	N (hexa)	ADDR-IO (hexa)
a	Addr	`ST 18(R1), R1`																			
b	Bnz	`BNZ R1, -58`																			
c	Movi	`MOVI R3, 0x80`																			

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

Completad las filas y columnas sombreadas 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 **(1 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 0x3DF8, el contenido de todos los registros es 0x7788 y que el contenido de todas las direcciones pares de la memoria es 0x85 y el de todas las impares es 0x00. 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 R0, -1(R6)`	
`STB -7(R0), R2`	
`MOVHI R2, 0xF0`	
`BZ  R4, 0xFFF6`	

Ejercicio 5 **(3 puntos)**

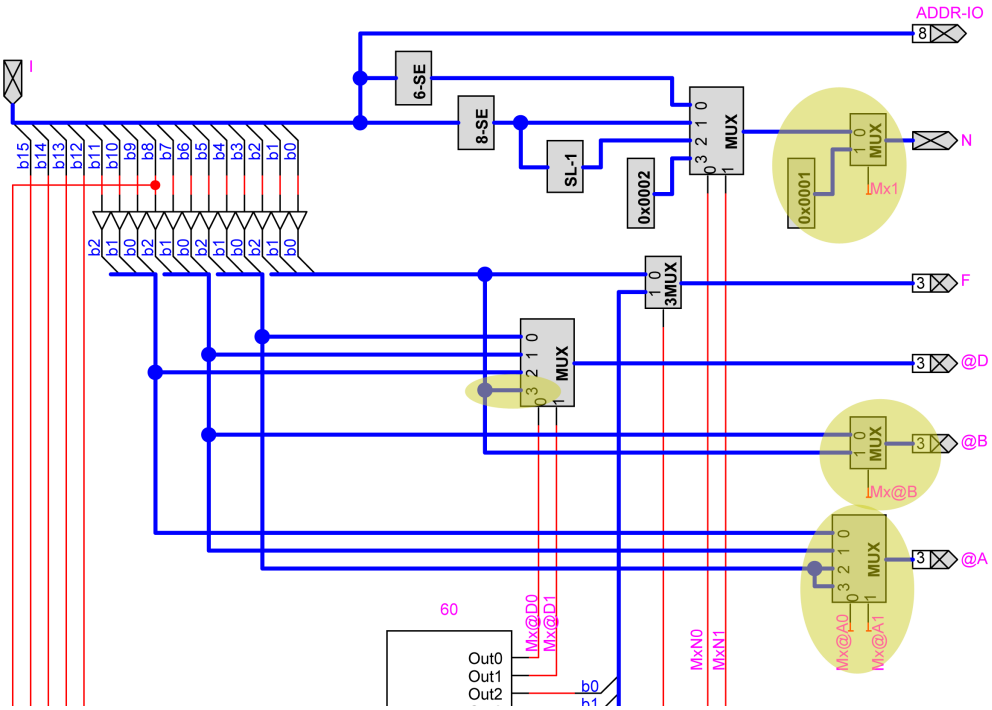
Completad el diseño del SISC Von Neumann para que pueda ejecutar, además de las 25 instrucciones originales SISA, la nueva instrucción ACCUMV, que tiene el formato y codificación, la sintaxis ensamblador y la semántica siguientes (en la semántica se especifica el incremento del PC ya que esta instrucción puede modificar el PC de forma poco convencional):

```
Codificación:  1011 aaa bbb ddd ccc
Sintaxis:      ACCUMV Rd, Ra, Rb, Rc
Semántica:     PC=PC+2; Rc=Mem_w[Ra]; Rd=Rd+Rc; Ra=Ra+2; Rb=Rb-1; if (Rb!=0) PC=PC-2;
```

La instrucción se usa para acumular sobre Rd la suma de los elementos de un vector de Rb words (con Rb_u>0) almacenados en memoria a partir de la dirección contenida en Ra, usando Rc como registro temporal.

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

Para poder ejecutar la nueva instrucción solamente se ha modificado la unidad de control del computador añadiendo dos nuevos MUX-2-1 con señales de selección Mx@B y Mx1 y un MUX-4-1 con señales Mx@A1 y Mx@A0 y se ha modificando la entrada 3 del MUX-4-1 que obtiene @D, que antes no se usaba. La siguiente figura muestra la parte de la Unidad de Control que ha cambiado respecto de la original. Las cuatro nuevas señales de selección se conectan a la salida de la ROM_OUT "por nombre" para simplificar el dibujo del esquema lógico. Además de estos nuevos multiplexores, ha cambiado la ROM_OUT que ahora tiene 28 bits por palabra (los 24 originales más los 4 nuevos). Aunque el número de palabras de la ROM_OUT es el mismo que en la versión original del computador hay seis palabras, que antes implementaban cada una de ellas la instrucción NOP, que ahora se usan para implementar la nueva instrucción. Por último, también debe cambiar el contenido de algunas palabras de la ROM_Q+. 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 8 ciclos/nodos que requiere la ejecución de la nueva instrucción (Fetch, Decode, y los 6 ciclos/nodos de la ejecución propiamente dicha): F, D, Acc1, Acc2... Acc6.Para especificar las acciones se ha usado el mismo lenguaje de transferencia de registros que en la documentación (con el que ya hemos completado las acciones del nodo E0/F. (**1 punto**)

Nodo/Estado		Acciones
Número	Mnem.	
E0	F	IR ← MEMw[PC] // PC ← PC+2
E1	D	[ ] // [ ] // RY ← Rb
E17	Acc1	R@ ← [ ]
E18	Acc2	Rc ← [ ] // [ ]
E19	Acc3	Ra ← [ ] // [ ] // RY ← Rc
E20	Acc4	Rd ← [ ]
E21	Acc5	R@ ← [ ] // [ ]
E22	Acc6	[ ] // if([ ]) [ ]

b) Completad (poniendo 0, 1 o x en cada bit) las filas/columnas marcadas con fondo gris 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 (Acc1) etc. (**1,5 puntos**)

@ROM	Mx@A1	Mx@A0	Mx@B	Mx1	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	
0																													F
1																													D
2																													AI
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																													Acc1
18																													Acc2
19																													Acc3
20																													Acc4
21																													Acc5
22																													Acc6
23..31																													Nop

c) Completad el fragmento del grafo de estados del circuito secuencial de la unidad de control necesario para ejecutar completamente la nueva instrucción ACCUMV. Se da la leyenda del grafo y todos los nodos, pero solo un arco al que le falta la etiqueta. Dibujad todos los arcos que faltan y todas las etiquetas. (**0,25 puntos**)

d) Indicad la dirección o las direcciones de la memoria ROM_Q+ y su contenido para implementar correctamente el paso del nodo/estado E1 al E17. (**0,25 puntos**)

