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

Examen Final, parte II

- Duración de esta parte: 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 se publicará mañana y las notas antes de una semana. La revisión será el 21 de junio a las 11:00 en el aula D6-114.

Ejercicio 1 **(0,4 puntos)**

El programa ensamblador se ha traducido a lenguaje máquina para ser ejecutado en el SISC Von Neumann, situando la sección `.data` a partir de la dirección 0x80A0 de memoria y a continuación la sección `.text`.

Una vez ensamblado y cargado el programa en memoria:

a) ¿A qué direcciones de memoria corresponden las etiquetas, o direcciones simbólicas siguientes?

C=0x	D=0x	L2=0x

b) ¿Cuál es la dirección de memoria y su contenido donde han quedado almacenadas las siguientes instrucciones?

`MOVHI R2, HI(C)`	=> Mem_w[0x	] = 0x
`BZ R0, L1`	=> Mem_w[0x	] = 0x
`LD R5, 0(R2)`	=> Mem_w[0x	] = 0x

```
.data
     N = 5
A:   .space 1
B:   .byte -27
     .even
C:   .word 2,-5,264,-63,23
D:   .byte 58,-64,32,0,-7
     .even
E:   .space 20
.text
L1:  IN      R0, KEY-STATUS
     BZ      R0, L1
     IN      R0, KEY-DATA
     MOVI    R2, LO(C)
     MOVHI   R2, HI(C)
     MOVI    R1, N
     MOVI    R4, 1
L2:  LD      R5, 0(R2)
     ...
```

Ejercicio 2 ***(0,6 puntos)***

Indicad qué cambios hay 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 0x3C18, el contenido de todos los registros es 0x9ABC y que el contenido de todas las direcciones pares de la memoria es 0xA5 y el de todas las impares es 0x00. Utiliza el mnemotécnico MEM_b[…]=... y/o MEM_w[…]=... para indicar los cambios en la memoria.

Instrucción a ejecutar	Cambios en el estado del computador
`LDB R2, -1(R6)`	
`STB -4(R1), R2`	
`ST 1(R3), R2`	

Ejercicio 3 ***(0,75 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 Von Neuman 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). La segunda y tercera columna definen la situación en la que se encuentra la Unidad de Control (UC) para cada apartado/fila: nodo/estado de la UC en ese ciclo e 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 R4 antes de ejecutarse la instrucción `BNZ R4, -7` es 0xFFFF.

			Palabra de Control																			
Apartado	**Nodo/Estado** (Mnemo Salida)	**Instrucción en IR** (en ensamblador)	@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	Addi	`ADDI R5, R2, -10`																			
b	Bnz	`BNZ R4, -7`																			
c	Addr	`LD R1, 28(R2)`																			

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

Completad la siguiente 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	R@/Pc	Pc/Rx	MxN1	MxN0	
5						Addr
8						Ldb
11						Bz
14						Movhi

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

Completad el diseño del SISC Von Neumann para que pueda ejecutar, además de las 25 instrucciones originales SISA, la nueva instrucción de salto `BRBkZ` (Branch to Register if Bit k equals Zero), que tiene la sintaxis ensamblador, el formato y codificación y la semántica siguientes:

Sintaxis: `BRBkZ Ra, Rb, k`
(`k` es un número entero con rango de 0 a 15 ambos incluidos))

Codificación: `1011 aaa bbb nnnnnn`
(ni `aaa ni bbb` pueden ser `000`) (`N6` = `k-1`)

Semántica: `PC= PC+2; if (Ra<k> == 0) PC = Rb;`
(`R0` queda modificado con un valor no definido)

`Ra<k>` con $0 \leq k \leq 15$ es el bit `k` del contenido del registro `Ra`. La instrucción `BRBkZ` es una instrucción de salto condicional: se rompe la ejecución en secuencia si el bit `k` de `Ra` es cero. Sería lógico, de cara a ensamblar la instrucción, que los 6 bits, `nnnnnn`, del campo `N6` de la instrucción codificaran en binario el valor `k`, pero ello requeriría modificar la UC para generar una mascara adecuada. Para evitar este cambio en la UC, el programa ensamblador genera el campo `N6` con el valor `k-1`. Por ejemplo para la instrucción `BRBkZ R1, R2, 2` el programa ensamblador la traduce a lenguaje máquina como `1011 001 010 000001`. De esta forma la UC puede generar una máscara adecuada usando la constante 0x0002 que ya se encuentra en la UC original.

La dirección destino de salto tomado esta contenida en `Rb`. Si no se cumple la condición de salto se pasa a ejecutar la siguiente instrucción en secuencia: la que está en la dirección `PC+2`. La ejecución de esta instrucción usa `R0` como registro temporal, para almacenar ciertos valores necesarios para la ejecución de la instrucción en el computador. `R0` quedará modificado después de la ejecución de la instrucción con un valor que no queda definido en la especificación de la semántica de la instrucción, porque no tiene interés.

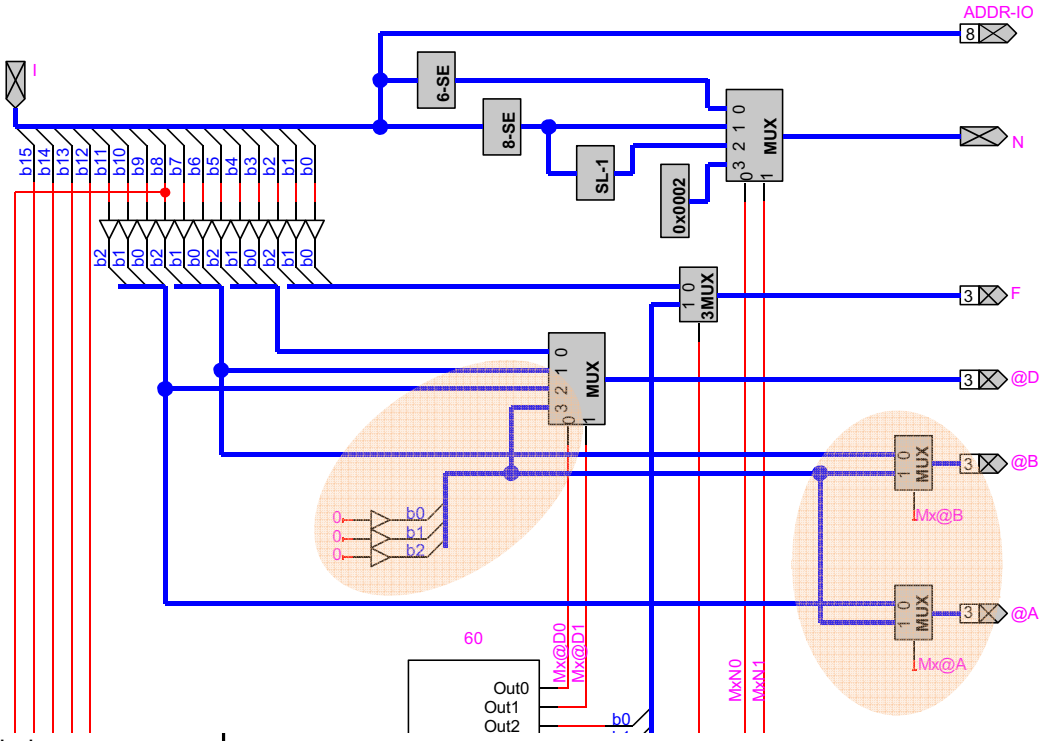
Para poder ejecutar la nueva instrucción solamente se ha modificado la unidad de control del computador añadiendo dos nuevos MUX-2-1 cuyas señales de selección, Mx@A y Mx@B, son generadas por la ROM_OUT y generando tres bits a cero para poder leer y escribir `R0` sin que esté explicitado el campo 000 en los 16 bits de la instrucción.

La siguiente figura muestra la parte de la Unidad de Control que ha cambiado respecto de la original. Las dos nuevas señales se conectan a la salida de la ROM_OUT "por nombre" para simplificar el dibujo del esquema lógico. Además de estos dos multiplexores ha cambiado la `ROM_OUT` que ahora tiene 26 bits por palabra (los 24 originales más los dos nuevos). Aunque el número de palabras de la `ROM_OUT` es el mismo que en la versión original del computador hay cinco palabras, que antes implementaban cada una de ellas la instrucción `NOP`, que ahora se usan para implementar la nueva instrucción.

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 7 ciclos/nodos que requiere la ejecución de la nueva instrucción (Fetch, Decode, y los cinco ciclos/nodos de la ejecución propiamente dicha): F, D, Brb1, Brb1... Brb5. 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)

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

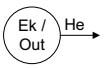


Nodo		
Número	Mnemotécnico	Acciones
E0	F	IR ← Mem$_w$[PC] // PC ← PC+2
E1	D	R@ ← PC+SE(N8)*2 // [ ] // [ ]
E17	Brb1	R0 ← [ ]
E18	Brb2	RX ← [ ]
E19	Brb3	[ ] ← SHL(RX, SE(N6)) // [ ]
E20	Brb4	[ ] // [ ] // [ ]
E21	Brb5	AND(RX, RY) // if ([ ]) [ ]

b) Completad el fragmento del grafo de estados del circuito secuencial de la unidad de control necesario para ejecutar completamente la nueva instrucción BRBkZ, que se muestra en la parte de la izquierda de la siguiente caja. 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. (0,1 punto)

c) Completad las filas/columnas marcadas con fondo gris de la tabla que se encuentra a la derecha de la siguiente caja y que especifica el contenido de la ROM-OUT. Indicad los bits (0, 1 o x) de las palabras/filas de la ROM_OUT para cada estado del fragmento del grafo que habéis dibujado en el apartado c para que implemente las acciones especificadas en el apartado b. Indicad también los bits de las nuevas dos señales (columnas) Mx@A y Mx@B para todas las filas de la tabla. La dirección 0 de la ROM corresponde al estado E0 (F) , la 1 al E1 (D)... la dirección 17 al estado E17 (Brb1) etc. (0,4 puntos)

Leyenda:



Ek: Estado k (k=número de orden en decimal).
Out: mnemotécnico de salida.
H: código de operación (dígito hexadecimal).
e: extesión del código de operación (bit).

@ROM	Mx@A	Mx@B	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
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															Brb1
18															Brb2
19															Brb3
20															Brb4
21															Brb5
22..31															Nop