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

Examen Final, parte 2

- Duración de esta parte: 2 horas. Escribid las respuestas en el enunciado y entregad todas sus hojas con apellidos y nombre en la cabecera.
- 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 20/06/2023 a las 12:00 en el aula C6-E101.

Ejercicio 1 (1 punto)

Para ser ejecutado en el SISC Von Neumann, las instrucciones del programa en ensamblador SISA, una vez completado, se deberán traducir a lenguaje máquina y cargar en la memoria del computador a partir de la dirección 0x0000. (A cada instrucción le hemos asignado como nombre un número o letra a la izquierda para referirnos a ellas en algunas de las preguntas).

Los datos del programa (que no se muestran en el código) también se deberán cargar en la memoria antes de iniciarse la ejecución. Los datos consisten en un vector de 10 elementos, almacenados en direcciones consecutivas de memoria a partir de la dirección 158. Cada elemento es un número entero codificado en complemento a dos en un byte. V[0] se almacena en la dirección 158, V[1] en la 159 y así sucesivamente. La secuencia de los valores de los elementos del vector (V[0], V[1], V[2]...) almacenados en memoria es: 0x21, 0x00, 0xFF, 0x56, 0x96, 0x83, 0xF3, 0x79, 0x0F, 0xE2.

El programa calcula cuántos de los elementos del vector son negativos y escribe este valor por la impresora.

Al escribir el programa podéis usar las típicas direcciones simbólicas de los puertos de la impresora. No podéis usar etiquetas para los saltos. Se usa `R7` como registro temporal siempre que se puede.

```
1   M [    ]  R0, [    ]
2   M [    ]  R1, [    ]
3   M [    ]  R1, [    ]
4   MOVI       R2, 0x00
5   M [                ]
6   [    ]     R7, 0(R1)
a   C [    ]  R7, R0, R7
b   B [    ]  R7, [    ]
c   A [    ]  R2, [    ]
7   A [    ]  R1, [    ]
8   A [    ]  R3, [    ]
9   [    ]     R3, [    ]
10  [                  ]
11  [                  ]
12  [                  ]
```

a) Completad el código SISA para la funcionalidad descrita.(**0,6 puntos**)

b) Después de ejecutar por quinta vez la instrucción numerada con 6, ¿Cuál es la dirección de memoria a la que se ha accedido y cuál es el valor que se ha escrito en R7 (dar ambos en hexadecimal)? (**0,1 puntos**)

Dir. Mem. = `0x` , `R7` = `0x`

c) ¿Cuántas instrucciones se ejecutan y en cuantos ciclos desde la instrucción numerada con 1 a la numerada con 12 (ambas incluidas)? Suponed, para esta pregunta y la siguiente que cada vez que se lee el registro de estado de la impresora su contenido vale 1. (**0,1 puntos**)

Nº instruc. ejecutadas = Nº ciclos =

d) Reescribid el código para que el tiempo de ejecución sea independiente de los valores concretos de los elementos del vector. Para ello debéis intercambiar las tres instrucciones `a`, `b` y `c` por dos nuevas instrucciones `d` y `e`. Indicad el código ensamblador de las dos instrucciones. ¿En cuantos ciclos se ejecuta el nuevo código? (**0,2 puntos**)

```
d  .........................
e  .........................
```

Nº ciclos =

Ejercicio 2 (0,6 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 que el contenido de todos los registros, `Rk` para k=0,...7, antes de ejecutarse cada instrucción es 0.

Apartado	Nodo (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	Ldb	`LDB R6,0x32(R7)`																			
b	D	`ADD R1,R2,R3`																			
c	Movi	`MOVI R4,0x88`																			

Ejercicio 3 **(0,6 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	
5												Addr
9												Stb
11												Bz
14												Movhi

Ejercicio 4 **(0,6 puntos)**

Indicad qué cambios se producen en el estado del computador SISC Von Neumann con su teclado e impresora 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 los registros es Ri= 3*i, el contenido de cada byte de memoria es MEMb[@]=(@+2)%2^8 (el símbolo % es la operación módulo) y el contenido de todos los puertos de entrada es 1 y el de los de salida es 2. Utilizad la notación MEM_b[0x…]=0x... para indicar cambios en la memoria (si se modifica un word hay que indicar separadamente el cambio en cada uno de los dos bytes) y PORTIN[Dir.Simbolica]=0x... y PORTOUT[Dir.Simbolica]=0x... para los cambios en los puertos de entrada/salida.

Instrucción a ejecutar	Cambios en el estado del computador
`ST -8(R4), R5`	
`LDB R7, 0x3F(R7)`	
`IN R7, KEY-DATA`	

Ejercicio 5 (1,2 puntos)

Vamos a diseñar el SISC2018 Von Neumann para que pueda ejecutar código SISA2018. El SISA2018 es igual al SISA excepto que:

1. El estado del computador del SISA2018 es el mismo que el del SISA más un nuevo registro especial de un bit (biestable) que denominamos `Fz` (Flag z). El `Fz` solamente lo escriben todas las instrucciones de tipo Aritmético-Lógico, AL, y Comparación, CMP, que además de incrementar en dos el PC (como todas las instrucciones) y escribir el registro destino (`Rd`) con el resultado de la operación, modifican `Fz` como se indica a continuación (la siguiente sentencia hay que añadirla a la semántica original de las instrucciones AL y CMP en el SISA2018):

   ```
   if (Rd == 0) Fz = 1; else Fz = 0;
   ```

2. El SISA2018, además de las 25 instrucciones originales (incluida la modificación anterior en las AL y CMP) tiene nuevas instrucciones aritmético-lógicas y de comparación ***Condicionales***, que denominamos ALc y CMPc. Estas nuevas instrucciones condicionales hacen lo mismo que las originales SISA pero solo escriben el resultado en `Rd` si `Fz` vale 0. Las instrucciones ALc y CMPc no escriben `Fz`, solo lo leen para decidir si se escribe `Rd`. La definición de las nuevas instrucciones es:

   `Codificación:` `cccc aaa bbb ddd fff` (excepto la `NOT` que en `bbb` tiene `xxx`)
   Siendo `cccc` igual a `1110` para las instrucciones ALc y `1111` para las CMPc. La codificación del campo `fff` para las instrucciones ALc y CMPc es la misma que para las AL y CMP, respectivamente.

   `Sintaxis:` `MNEMOc Rd, Ra, Rb`
   siendo `MNEMO` cualquiera de los mnemotécnicos de las operaciones `AL` o `CMP` originales.
   Ejemplo de instrucción `ALc`: `SUBc Rd, Ra, Rb`. Ejemplo de `CMPc`: `CMPLEUc Rd, Ra, Rb`.

   `Semántica:` `if (Fz == 0) Rd = Ra op Rb;`
   siendo `op` la operación AL o CMP que corresponde al `MNEMO` (de la instrucción en ensamblador)
   o al código de operación y campo `fff` (de la instrucción en lenguaje máquina).

En general, el SISA2018 es más eficiente que el SISA original ya que se puede reducir, en algunos casos, el número de instrucciones `BZ` y `BNZ` utilizadas. Las primeras dos preguntas de este ejercicio tratan este tema:

a) Completad los código SISA original y SISA2018 para que cada uno de ellos realice la siguiente funcionalidad (La comparación es de números naturales. Usad R7 como registro temporal si es necesario): (**0,1 puntos**)

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

```
if  (R5 >= R6) R0 = R0 + R1;
```

```
SISA:   C      R7,
        B      R7,
        A      R0,

SISA2018:  C      R7,
           A      R0,
```

b) ¿Cuanto vale x para que sea cierta la siguiente afirmación? "El código SISA del apartado anterior ejecutado en el SISC Von Neumann es un x% más lento que el código SISA2018 ejecutado en el SISA2018 Von Neumann". Suponed que los dos computadores tienen el mismo tiempo de ciclo, que es Tc=2000 u.t., que al ejecutarse el código `R5` es mayor o igual que `R6` y que en el SISC2018 las instrucciones tardan el mismo número de ciclos en ejecutarse que en el SISC (las ALc y CMPc tardan lo mismo que las AL y CMP). (**0,1 puntos**)

x = %

A continuación vamos a modificar el diseño del SISC Von Neumann para que pueda ejecutar el lenguaje máquina SISA2018 y convertirlo así en el computador SISC2018 Von Neumann. Para ello, solamente se requieren hacer los siguientes cambios en la Unidad de Control (UC) del SISC Von Neumann:

1. Se añade un biestable D activado por flanco ascendente de la señal de reloj, que implementa el registro especial de estado, de 1 bit, `Fz` (la salida de este biestable es `Fz`). Este biestable lo podríamos disponer en la UP, pero lo dispondremos en la UC, ya que esto es indiferente.
2. La ROM_OUT deberá generar dos nuevas señales de salida, denominadas Cond y LdFz, para poder implementar las diferencias entre SISA y SISA2018. La señal Cond valdrá 1 en el ultimo ciclo de ejecución de cualquiera de las nuevas instrucciones Condicionales y 0 en cualquier otro caso. La señal LdFz sirve, como se indica en el punto 4, para gestionar la carga de Fz. La ROM_Q+ también deberá cambiar para adaptarse a los nuevos nodos.
3. Se cambia el nombre de la señal de salida de la ROM_OUT, WrD, que pasará a llamarse WrD1, para diferenciarla de la señal WrD que sale de la Lógica de Control para ir a la UP como parte de la palabra de control, ya que ahora no siempre WrD (de la palabra de control) será igual a WrD1 (de la salida de la ROM_OUT del SISC2018). No importará el valor que tenga WrD1 en el ultimo ciclo de ejecución de cualquiera de las nuevas instrucciones Condicionales, por lo que en estos casos se especifica como x.
4. Se añade un pequeño circuito **secuencial** en la UC, que denominamos **FzC** (*Fz Control*), que contiene al biestable Fz. La salida del circuito FzC es la salida Fz del biestable. Al circuito le entran, además de la señal Clk, las señales LdFz y z, para gestionar su funcionamiento: `if (LdFz(c) == 1) Fz(c+1) = z(c) else Fz(c+1) = Fz(c);` para cualquier ciclo `c = 0, 1, 2...` siendo (para cualquier señal binaria x) x(c) el valor de la señal x en el ciclo `c`. Los nombres de las entradas y salidas indican su conexionado en la UC.
5. Se añade un pequeño circuito **combinacional** en la UC, denominado **CIC** (*Conditional Instrucction Control*), que genera la señal WrD, que forma parte de la palabra de control del SISC2018, en función de sus señales de entrada: Cond, Fz y WrD1.

Responded a las siguientes preguntas sobre el diseño del SISC2018:

c) Dibujad el grafo de estados del circuito lógico secuencial FzC, sin olvidar la leyenda del grafo. Escribid la expresión lógica en suma de productos mínima (por Karnaugh) del estado siguiente y de la salida del circuito (Dibujando primero los dos mapas de Karnaugh con las agrupaciones de unos). Dibujad el esquema lógico del circuito secuencial implementando los circuitos combinacionales del estado siguiente y de la salida en suma de productos mínima (que son el resultado de la minimización por Karnaugth anterior) usando puertas Not, And y Or. (**0,2 puntos**)

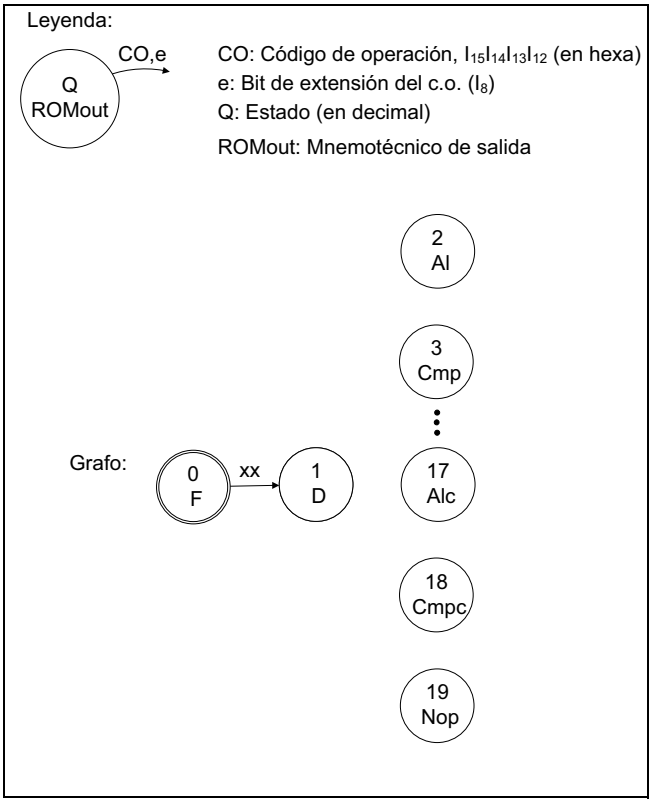
Leyenda	**Grafo**	**Mapa y expresión Q+**	**Mapa y expresión Salida**	**Esquema lógico del circuito**

d) Escribid la expresión lógica en suma de minterms de la salida del circuito combinacional CIC (WrD) ordenando las entradas así: Cond, Fz, WrD1. (**0,1 puntos**)

WrD(Cond, Fz, WrD1) =

e) Completad el fragmento del grafo de la UC necesario para ejecutar las instrucciones AL y CMP originales, las nuevas ALc y CMPc del SISA2018 y todas las que tienen un código ilegal (que se ejecutan en Nop). Se dan todos los nodos necesarios y su codificación, pero solo un arco con su etiqueta. Dibujad todos los arcos que faltan con sus etiquetas. (**0,1 puntos**)

f) Completad (poniendo 0, 1 o x en cada bit) las casillas vacías de las tablas que especifican parte del contenido de la ROM_OUT para que se ejecuten correctamente todas las instrucciones SISA2018, poniendo el máximo número de x posibles. Damos la señal Cond totalmente completada y la señal WrD1 parcialmente completada, de acuerdo con la definición que hemos dado de ellas anteriormente. La dirección 0 de la ROM_OUT corresponde al estado 0 (F), la 1 al 1 (D), la 2 y 3 a las instrucciones aritmético-lógicas y de comparación (que han modificado su semántica) y las direcciones 17 y 18 a los dos nuevos estados Alc y Cmpc. En la segunda tabla, se representan las dos columnas de la ROM_OUT para las nuevas salidas Cond y LdFz, que se han dibujado en horizontal para ocupar menos espacio (En la fila de cabecera se indica la dirección de la ROM_OUT y el mnemotécnico de salida. Se han sombreado las casillas que no tenéis que completar por haberlo hecho en la primera tabla). (**0,5 puntos = 0,2** (LdFz) **+ 0,3** (el resto)



@ROM	Cond LdFz	Bnz	Bz	WrMem	RdIn	WrOut	WrD1	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	Mnemo
0	0																									F
1	0																									D
2	0																									Al
3	0																									Cmp
…		…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
17	1						x																			Alc
18	1						x																			Cmpc

	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 Alc	18 Cmpc	19 Nop
Cond					0	0	0	0	0	0	0	0	0	0	0	0	0			
LdFz																				

g) Indicad la dirección o las direcciones (en binario, usando x para representar varias direcciones cuyo contenido de la ROM es el mismo) de la memoria ROM_Q+ y su contenido o sus contenidos (en Hexa) para implementar correctamente el arco del nodo/estado 1 al 18 y todos los arcos que salen del nodo 18. (**0,1 puntos**)