Apellidos y nombre: ..... Grup: ..... DNI:.....

Examen 3. (Temas 8, 9, 10 y 11)

- 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.
- La solución del examen se publicará en Atenea mañana y las notas antes del 1 de diciembre.

Ejercicio 1 **(0.5 puntos)**

Raona els avantatges i inconvenients de disposar d'una unitat de control amb seqüenciament implícit davant d'una unitat de control amb seqüenciament explícit. Marca amb una creu totes les afirmacions correctes (*nota:les respostes incorrectes restaran nota sobre les correctes*)

- ☐ Les unitats de control amb seqüenciament implícit no poden saltar a la mateixa instrucció que estan executant com si poden fer-ho les unitats de control amb seqüenciament explícit.
- ☐ Els programes/grafs amb seqüenciament implícit tenen menys instruccions/estats que amb seqüenciament explícit.
- ☐ Per fer programes/grafs amb seqüenciament implícit necessitem noves instruccions especials.
- ☐ En general, els programes/grafs amb seqüenciament implícit ocupen menys espai a la memòria ROM.
- ☐ Les unitats de control amb seqüenciament explícit necessiten disposar d'un registre que contingui l'adreça de la ROM on està ubicada la instrucció que s'està executant.
- ☐ Les unitats de control amb seqüenciament explícit executen les instruccions en l'ordre que estan escrites a memòria.
- ☐ Les unitats de control amb seqüenciament implícit incrementen automàticament el registre PC amb el nombre de posicions de memoria que ocupa una instrucció quan executen una instrucció que no trenqui la seqüència.
- ☐ Les unitats de control amb seqüenciament implícit son capaces de fer més tasques que les unitats amb seqüenciament explícit.
- ☐ Les unitats de control amb seqüenciament explícit son capaces de fer més tasques que les unitats amb seqüenciament implícit.
- ☐ Les unitats de control amb seqüenciament implícit necessiten d'un bloc que incrementi el valor PC.

Ejercicio 2 **(1 punto)**

a) Indica el valor que debe tener cada uno de los bits de la palabra de control de la UPG (sin subsistema de I/O ni memoria) para que realice, durante un ciclo, la acción concreta especificada mediante el mnemotécnico. Indicad con ***x*** las casillas cuyo valor no importe para la ejecución de la instrucción. En caso de que no se pueda realizar la acción tachar **toda la línea** de señales. *(0.5 puntos)*

Mnemotécnico	@A	@B	Rb/N	OP	F	In/Alu	@D	WrD	N *(hexa)*
MOVEI R0, 0xABCD									
IN R7 // XORI –, R4, –6									
CMPLEU R3, R1, R0									

b) Indica el mnemotécnico que corresponde a cada una de las siguientes palabras de control de la UPG (sin subsistema de I/O ni memoria). *(0.5 puntos)*

Mnemotécnico	@A	@B	Rb/N	OP	F	In/Alu	@D	WrD	N *(hexa)*
	100	001	1	00	100	0	011	1	X X X X
	111	xxx	0	01	011	x	xxx	0	0 0 0 A
	101	xxx	x	00	011	1	000	1	X X X X

Ejercicio 3 **(2 puntos)**

Dados los dos siguientes fragmentos de código en C (el código no tiene que hacer algo útil), indicad como se implementarían cada uno en un procesador que use la UPG vista en clase, utilizando la UC de **propósito específico** (UCe) y la UP de **propósito general** (UPG). Todos los datos son **naturales**.

Fragmento 1

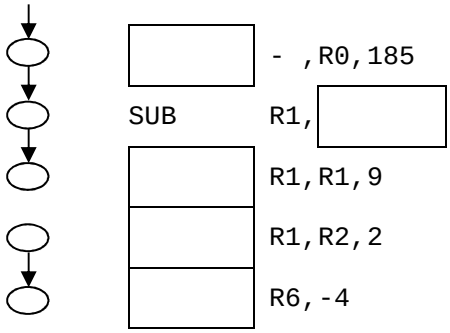
```
if (R0<=185) { R1=R0-R1+9; }
else  { R1=R2-2; }
R6=-4;
```

Fragmento 2

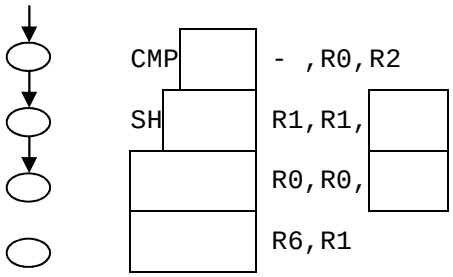
```
while (R0>R2) {
      R1=R1/4;
      R0=R0-3;
}
R6=not(R1);
```

a) Completad los dos fragmentos de grafo de estados de la UC de **propósito específico** para que junto con la UPG formen un procesador que realice la funcionalidad descrita en los fragmentos de código anteriores. Indicad los arcos que faltan, las etiquetas de los arcos (z, !z, o nada) y completad las casillas de cada palabra de control que se especifica con mnemotécnicos a la derecha de cada nodo del grafo.

Fragmento 1 *(0.5 puntos)*



Fragmento 2 *(0.5 puntos)*



b) Completad los fragmentos de programa en lenguaje ensamblador SISA para que el procesador formado por la unidad de control de propósito general (UCG) junto con la UPG realicen las funcionalidades descritas en los fragmentos de código en C (el código no tiene que hacer algo útil). El código SISA ya escrito siempre utiliza el registro R7 para valores temporales. En las comparaciones, hay que interpretar los datos como valores **naturales**. Rellenad la parte subrayada que falta.

Fragmento 1 *(0.5 puntos)*

@I-Mem	
0x0000	_____ R7, 0xB9
0x0002	MOV___ R7, ____
0x0004	CMP___ R7, R0, R7
0x0006	B____ R7, ____
0x0008	SUB R1, _____
0x000A	_____ R1, R1, 9
0x000C	B____ R7, ____
0x000E	_____ R1, R2, ____
0x0010	_____ R6, -4

Fragmento 2 *(0.5 puntos)*

@I-Mem	
0x0000	CMP___ R7, R0, R2
0x0002	B____ R7, ___
0x0004	MOVI R7, ___
0x0006	S_____ R1, R1, ___
0x0008	ADDI R0, R0, ___
0x000A	B_____ R7, ___
0x000C	_____ R6, R1

Ejercicio 4 **(0.5 puntos)**

Completa la siguiente tabla ensamblando las instrucciones en ensamblador SISA o desensamblando las instrucciones en lenguaje máquina según sea necesario. Indica poniendo NA en la casilla aquellos casos en los que la instrucción no corresponda al lenguaje SISA.

Lenguaje máquina SISA	Lenguaje ensamblador SISA
0x022E	
	BZ R5, -3
0x3877	

Apellidos y nombre: ..... Grup: ..... DNI:.....

Ejercicio 5 **(0.75 puntos)**

Escribid sobre la siguiente tabla el valor de los bits que tiene la palabra de control del SISC-Harvard uniciclo (incluyendo la señal *TknBr*) durante el ciclo en que se ejecuta cada una de las instrucciones SISA. Indicad únicamente el valor (0 o 1) de los bits que son estrictamente necesarios para ejecutar correctamente cada instrucción. Para el resto de bits de la palabra de control, que pueden valer 0 o 1 indistintamente para la ejecución correcta de la instrucción, poned x (aunque se pueda saber el valor codificando la instrucción). Suponed que antes de ejecutar cada instrucción el contenido de los registros, de los puertos de entrada/salida y de la memoria de datos es cero.

Instrucción SISA	**Palabra de Control**														
	@A	@B	Rb/N	OP	F	-/i/l/a	@D	WrD	Wr-Out	Rd-In	Wr-Mem	Byte	TknBr	N *(hexa)*	ADDR-IO *(hexa)*
STB 3(R2), R5															
BNZ R7, 11															
MOVHI R1, -2															

Ejercicio 6 **(0.75 puntos)**

Indicad qué cambios hay en el estado del computador después de ejecutar cada una de las instrucciones de la tabla suponiendo que **antes de ejecutarse cada una** de ellas el PC vale 0x8C36, el contenido de todos los registros es 0xFFDA y que el contenido de todas las posiciones pares de la memoria de datos es 0x83 y el de todas las posiciones impares de la memoria de datos es 0x24. Utiliza el mnemotécnico $MEM_b[\ldots]$, $MEM_w[\ldots]$ y DataOut[…] para indicar los cambios en la memoria y los puertos de E/S respectivamente.

Instrucción a ejecutar	**Cambios en el estado del computador**
`BNZ R5, 6`	
`LDB R4,8(R5)`	
`MOVHI R7,3`	

Ejercicio 7 **(1.5 puntos)**

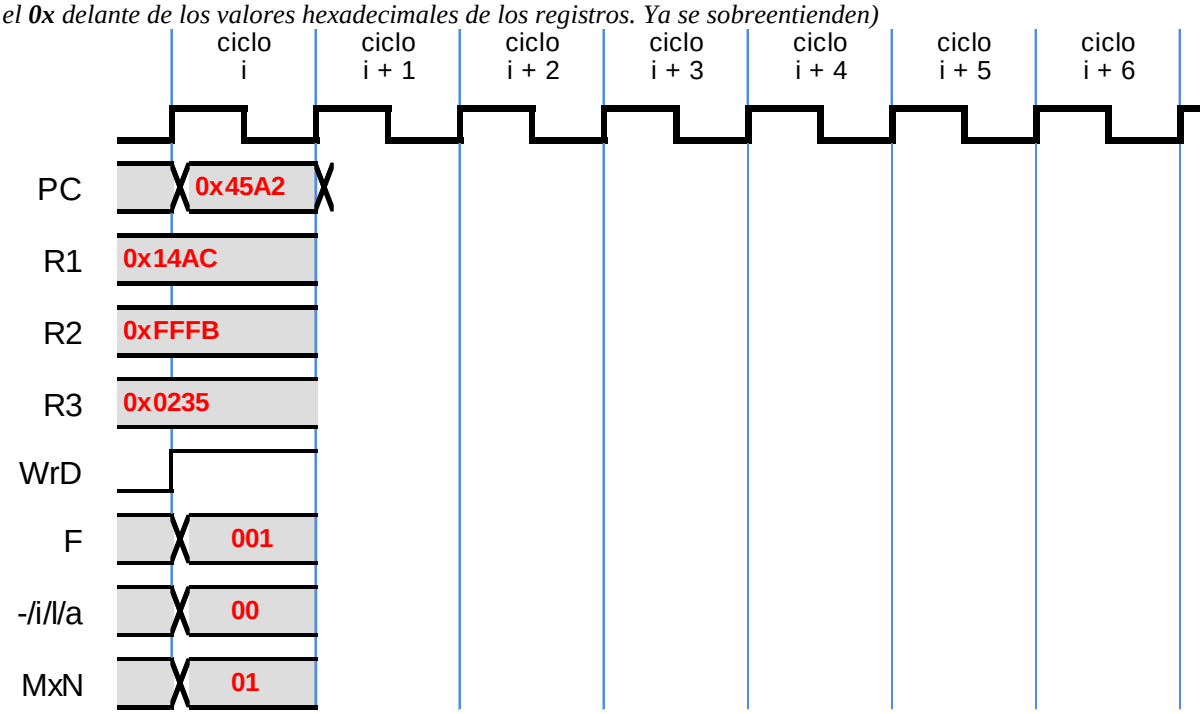
Indica el contenido de la tabla de la ROM (sólo las celdas en blanco) correspondiente al bloque ROM_CRTL_ LOGIC. Indica los valores que tomarían las señales para ejecutar correctamente las instrucciones. Indica con x los valores de los bits del contenido de la ROM que puedan valer 0 o 1.

Dirección ROM					Contenido de la ROM																		
I_{15}	I_{14}	I_{13}	I_{12}	I_8	Bnz	Bz	Wr-Mem	Rd-In	Wr-Out	WrD	Byte	Rb/N	-/i/l/a_1	-/i/l/a_0	OP_1	OP_0	MxN_1	MxN_0	MxF	f2	f1	f0	MxD_1	MxD_0	
0	0	0	0	X																					*A / L*
0	0	0	1	X																					*CMP*
0	0	1	0	X																					*ADDI*
0	0	1	1	X																					*LD*
0	1	0	0	X																					*ST*
0	1	0	1	X																					*LDB*
0	1	1	0	X																					*STB*
0	1	1	1	X																					*(NOP)*
1	0	0	0	0																					*BZ*
1	0	0	0	1																					*BNZ*
1	0	0	1	0																					*MOVI*
1	0	0	1	1																					*MOVHI*
1	0	1	0	0																					*IN*
1	0	1	0	1																					*OUT*
1	0	1	1	X																					*(NOP)*
1	1	X	X	X																					*(NOP)*

Ejercicio 8 **(1 punto)**

Dado el siguiente fragmento de código donde en el *ciclo i* se ejecuta la instrucción `MOVI R3,0xFE`, rellenad el siguiente cronograma indicando el valor de las señales de la UCG o UPG y los valores de los registros. *(nota: para facilitar la lectura del cronograma no hace falta que pongáis el **0x** delante de los valores hexadecimales de los registros. Ya se sobreentienden)*

```
MOVI R3,0xFE
MOVHI R3,0x00
XOR R2, R2, R1
SHL R1, R1, R3
BNZ R1, -3
OUT 17,R2
```

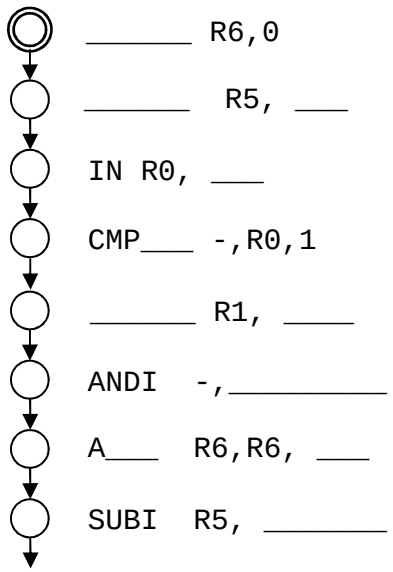


Ejercicio 9 **(2 puntos)**

Se ha conectado a la UPG un dispositivo externo de entrada que nos envía valores **naturales 16 bits** y que tiene el registro de status en la dirección 5 del espacio de direccionamiento de entrada y el de datos en la 6. Este dispositivo tiene un efecto lateral en la lectura del dato sobre su registro de estado.

Este dispositivo de entrada nos envía de forma asíncrona 200 valores y se desea saber cuántas operaciones de multiplicación darían overflow si cada dato recibido lo multiplicásemos por el valor 16. El resultado de esta cuenta debe dejarse en el registro R6.

a) Completad el grafo de estados si estuviésemos utilizando una unidad de control específica (UCe) junto a la UPG para que realice la función anteriormente descrita. Indicad los arcos y las etiquetas de los arcos (z, !z, o nada) que falten (en caso que falten) y completad las casillas de cada palabra de control. *(1 punto)*



b) Usando el procesador SISC Harvard uniciclo, completad el fragmento de código ensamblador SISA para que realice la función anteriormente descrita. Una vez finalizada la operación, el programa deberá dejar el resultado de 16 bits en la posición 0xF006 de la memoria de datos *(1 punto)*

```
M___   R6, 0
M___   R5, ___
M___   R5, ___
IN     R0, 5
B___   R0,___
____   R1, 6
MOVI   R2, ____
MOVHI  R2, ____
A____  R3, R1, R2
B___   R3, ___
ADDI   R6, R6, ___
ADDI   R5, R5, ___
B___   R5, ___
S___   _____, R6
```