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

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

- Duración del examen: 1 hora y 45 minutos.
- 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 XX de diciembre.

Ejercicio 1 **(1 punto)**

Completa el siguiente fragmento de código ensamblador SISA para el procesador SISC Harvard uniciclo (UPG+I/O+MEM) para que guarde en el registro R0 el byte almacenado en la posición de memoria 0x3456.

```
...
MOVI  R1, 0x56
MOVHI R1, 0x34
LDB   R0, 0(R1)
...
```

En el procesador SISC Harvard uniciclo se puede acceder a una misma posición de memoria de varias maneras. ¿De cuantas maneras distintas hubiésemos podido completar el código anterior (teniendo en cuenta que todos los valores numéricos siempre están escritos en hexadecimal) para que realizase el acceso al byte de la posición 0x3456? Indicad el número total y justificadlo.

El número de maneras de acceder a una misma posición de memoria es de 64. La dirección efectiva a la que se accede en la memoria de datos es el resultado de una operación de suma entre el contenido de un registro y una constante de 6 bits. Siempre que el resultado de la suma de estos dos elementos de el mismo valor accederemos a la misma posición de memoria. El registro puede contener cualquier valor de 16 bits, pero la constante de 6 bits sólo puede valer 64 valores distintos.

Criterio de corrección: +0.25 puntos si el fragmento de código es correcto. +0.75 si el valor 64 es correcto y la explicación coherente.

Ejercicio 2 **(1 punto)**

a) Indica el valor que debe tener cada uno de los bits de la palabra de control de la UPG básica (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)*
IN R1 // OUT R5 // MOVEI -, 0x1234	101	xxx	0	10	001	1	001	1	1 2 3 4
SHL R2, R3, R1 // OUT R3	011	001	1	00	111	0	010	1	X X X X

Criterio de corrección: -0.25 puntos por cada fila y columna incorrecta, escogiendo el número mínimo de filas y/o columnas que cubren todos los errores.

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

Mnemotécnico	@A	@B	Rb/N	OP	F	In/Alu	@D	WrD	N *(hexa)*
SUB R3, R1, R7	001	111	1	00	101	0	011	1	X X X X
CMPLTI -,R2,0xFFFC // IN R4	010	xxx	0	01	000	1	100	1	F F F C

Criterio de corrección: -0.25 puntos por cada mnemotécnico incorrecto.

Ejercicio 3 **(1 punto)**

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**
0x677D	STB -3(R3), R5
0xA0FC	IN R0, 252 o IN R0, 0xFC
0x9918	MOVHI R4, 24
0x85FB	BNZ R2, -5

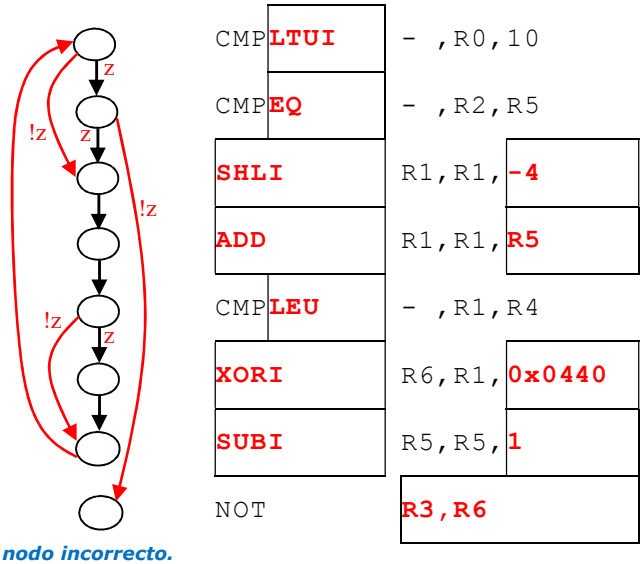
Criterio de corrección: +0.25 puntos por cada fila correcta.

Ejercicio 4 **(2 puntos)**

Dado el siguiente fragmento de código en C (el código no tiene que hacer algo útil), indicad como se implementarían 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**.

```
while ((R0<10) || (R2!=R5)) {
    R1=R1/16+R5;
    if (R1>R4)
        R6=XOR(R1,0x0440);
    R5--;
}
R3=not(R6);
```

a) Completad el fragmento 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. *(1 punto)*



Criterio de corrección: -0.2 puntos por cada nodo incorrecto.
Un nodo es erróneo si falta alguno de los arcos que salen de él, si alguna etiqueta es incorrecta o los destinos de alguno de sus arcos es incorrecto. También es incorrecto un nodo si la salida especificada mediante mnemotécnicos (operación, registros o valor inmediato) es incorrecta. Hacemos una excepción a esta regla: Si falta la I se descuenta 0.1 por ese nodo si sólo tiene ese fallo. Si falta la U en varios nodos solos se descuenta una vez.

b) Completad el fragmento 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. *(1 punto)*

@I-Mem	
0x0000	**MOVI** R7, 10
0x0002	CMP**LTU** R7, R0, R7
0x0004	B**NZ** R7, **2**
0x0006	CMP**EQ** R7, R2, R5
0x0008	B**NZ** R7, **10**
0x000A	**MOVI** R7, **-4**
0x000C	**SHL** R1, R1, **R7**
0x000E	ADD R1, R1, **R5**
0x0010	CMP**LEU** R7, R1, R4
0x0012	B**NZ** R7, **3**
0x0014	MOVI R7, **0x44**
0x0016	S**HL/A** R7, R7, R7
0x0018	**XOR** R6, R1, R7
0x001A	**ADDI** R5, R5, **-1**
0x001C	B**NZ** **R7**, **-15**
0x001E	**NOT** R3, **R6**

Criterio de corrección: -0.1 puntos por la primera instrucción incorrecta y -0.2 puntos por cada una de las siguientes instrucciones incorrectas.

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

***Ejercicio 5* (1 punto)**

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 del SISC Harvard uniciclo**														
	@A	@B	Rb/N	OP	F	-/i/l/a	@D	WrD	Wr-Out	Rd-In	Wr-Mem	Byte	TknBr	N *(hexa)*	ADDR-IO *(hexa)*
LDB R6, 0(R4)	100	xxx	0	00	100	01	110	1	0	0	0	1	0	0 0 0 0	X X
BNZ R4, 7	100	xxx	x	10	000	xx	xxx	0	0	0	0	x	0	0 0 0 E	X X
IN R1, 5	xxx	xxx	x	xx	xxx	10 1x	001	1	0	1	0	x	0	X X X X	0 5
CMPEQ R2, R3, R0	011	000	1	01	011	00	010	1	0	0	0	x	0	X X X X	X X

Criterio de corrección: -0.25 puntos por cada fila y columna incorrecta, escogiendo el número mínimo de filas y/o columnas que cubren todos los errores.

***Ejercicio 6* (1 puntos)**

Indica el contenido de la tabla de la ROM (sólo filas indicadas) 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	f_2	f_1	f_0	MxD_1	MxD_0	
0	0	0	0	X	0	0	0	0	0	1	x	1	0	0	0	0	x	x	0	x	x	x	0	0	*A / L*
0	1	0	0	X	0	0	1	0	0	0	0	0	x	x	0	0	0	0	1	1	0	0	x	x	*ST*
1	0	0	0	0	0	1	0	0	0	0	x	x	x	x	1	0	1	0	1	0	0	0	x	x	*BZ*
1	0	0	1	1	0	0	0	0	0	1	x	0	0	0	1	0	0	1	1	0	1	0	1	0	*MOVHI*

Criterio de corrección: -0.25 puntos por cada fila y columna incorrecta, escogiendo el número mínimo de filas y/o columnas que cubren todos los errores.

***Ejercicio 7* (1 punto)**

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 0x2A34, el contenido de todos los registros es 0xC328 y que el contenido de todas las posiciones pares de la memoria de datos es 0x15 y el de todas las posiciones impares de la memoria de datos es 0x73. 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**
`SHL R2, R1, R5`	**R2=0x2800 PC=0x2A36**
`ST 6(R6), R3`	**MEMw[0xC32E]=0xC328 PC=0x2A36**
`OUT 11, R3`	**DataOut[11]=0xC328 PC=0x2A36**
`MOVI R5,0x66`	**R5=0x0066 PC=0x2A36**

Criterio de corrección: -0.25 puntos por cada fila incorrecta sin contar los PC que siguen el secuenciammiento implicito. Si los PC que siguen el secuenciamento implicito estan mal sólo descuentan -0.25 puntos adicionales.

Ejercicio 8 **(1 punto)**

Se ha conectado a la UPG un dispositivo externo de entrada que nos envía valores y que tiene el registro de status en la dirección 4 del espacio de direccionamiento de entrada y el de datos en la 8. Este dispositivo tiene un efecto lateral en la lectura/escritura del dato sobre su registro de estado.

Se desea que este sistema vaya leyendo indefinidamente los datos que se reciban por el dispositivo de entrada mientras el valor de estos datos sea distinto de 0, y los vaya almacenando consecutivamente en la memoria de datos a partir de la posición 0x0064. Los datos recibidos son de 16 bits y primero se almacenará el valor recibido y luego su valor negado bit a bit. Por ejemplo, si el primer valor recibido es el 0xABCD, este valor se almacenará en la posición 0x0064 y en la posición 0x0066 se almacenará su valor negado (0x5432), cuando llegue el segundo valor por el dispositivo de entrada, por ejemplo el valor 0x1234, este valor se almacenará en la posición de memoria 0x0068 y su valor negado (0xEDCB) en la posición 0x006A, etc. Cuando el dato recibido sea 0, el sistema debe quedarse en un bucle infinito sin hacer nada.

Usando el procesador SISC Harvard uniciclo, escribid el programa en código ensamblador SISA para que realice la función anteriormente descrita.

```
MOVI R0, 0x64
IN   R7, 4
BZ   R7, -2
IN   R1, 8
BZ   R1, -1
ST   0(R0), R1
NOT  R1, R1
ST   2(R0), R1
ADDI R0, R0, 4
MOVI R7, 0
BNZ  R7, -10
```

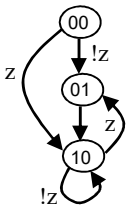
```
MOVI R0, 0x64
IN   R7, 4
BZ   R7, -2
IN   R1, 8
BZ   R1, -1
ST   0(R0), R1
NOT  R2, R1
ST   2(R0), R2
ADDI R0, R0, 4
BNZ  R1, -9
```

Hay muchas soluciones posibles, esto son 2 ejemplos

Criterio de corrección: +1 punto si el código hace la función descrita. En caso de que el código no sea correcto: +0.25 puntos si se hace correctamente la lectura del dato por el puerto, +0.25 puntos si se hace el bucle infinito, +0.25 si se hace la operación descrita y se almacenan los valores en la memoria, +0.25 si se incremente el registro con la dirección base, se inicializa correctamente el registro iterador y se salta al principio.

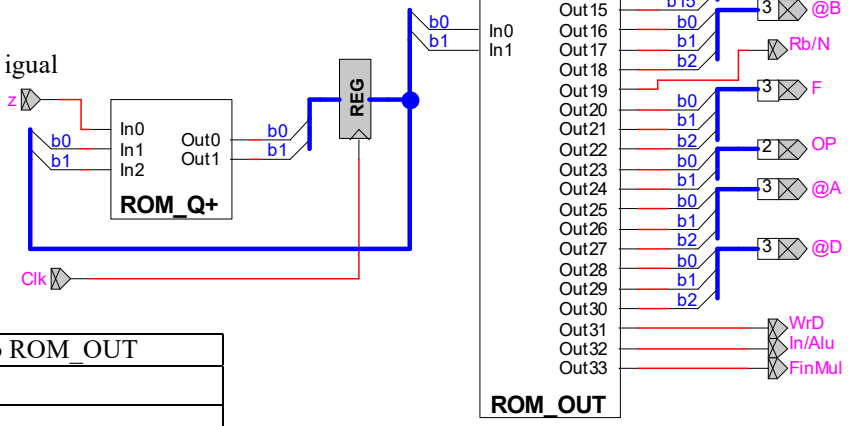
Ejercicio 9 **(1 punto)**

Dado el siguiente grafo de estados de una UC de propósito específico para que, junto con la UPG básica, formen un PPE que lo ejecute. En la figura se muestra el esquema de la unidad de control correspondiente a una implementación con dos roms. Una rom para el estado siguiente (ROM_Q+) y otra rom para las salidas (ROM_OUT).




```
ADD R0, R1, R2  // FinMul=0
MOVEI R3,0xFABC // FinMul=1
SHLI R4, R5, -2 // FinMul=0
```

Indica el contenido de las ROMS en hexadecimal de igual modo como lo hicisteis en la práctica 4. Utilizad el valor 0 para los bits que sean x. Si los contenidos no están en hexadecimal no se corregirá el ejercicio.



@ROM	Contenido ROM_Q+
@0	1
@1	2
@2	2
@3	2
@4	2
@5	1
@6	0
@7	0

@ROM	Contenido ROM_OUT
@0	0824A0000
@1	2B110FABC
@2	0CA70FFFE
@3	000000000

Criterio de corrección ROM_Q+ (0.25 puntos): binario. +25 puntos si es completamente correcta
Criterio de corrección ROM_OUT (0.75 puntos): -0.25 puntos por cada fila incorrecta.