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

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

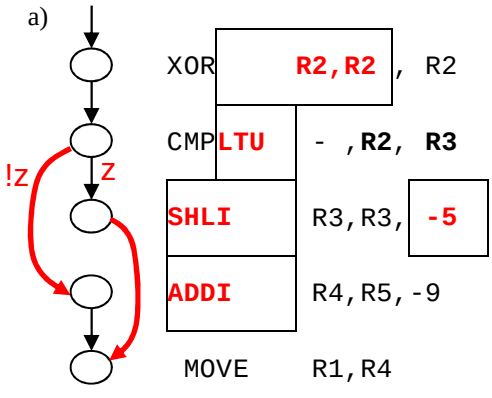
- Duración del examen: 2 horas.
- La solución de cada ejercicio se tiene que escribir en el espacio reservado para ello en el propio enunciado.
- No podéis utilizar calculadora, móvil, apuntes, etc.
- La solución del examen se publicará en Atenea mañana y las notas antes del 1 de diciembre.

***Ejercicio 1*** *(Objetivos 8.x)* **(1 punto)**

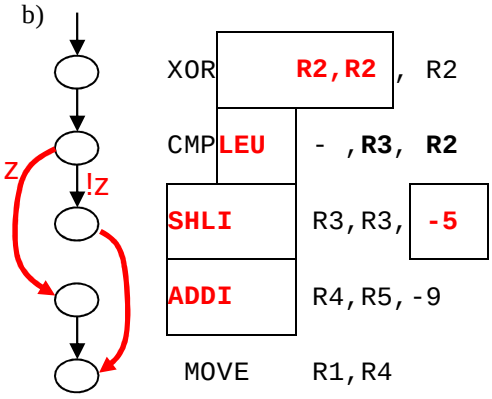
Completad los dos fragmentos de grafo de estados de la UC de **propósito específico** para que junto con la UPG formen un procesador que realice la funcionalidad descrita mediante el siguiente código en C. Indicad los arcos que faltan, las etiquetas de los arcos (z, !z, o nada) y completad las casillas de cada palabra de control que se especifica con mnemotécnicos a la derecha de cada nodo del grafo. Todos los datos son **naturales**. Ambos fragmentos ejecutan la misma funcionalidad pero de forma distinta.

```
R2 = 0;
if (R3<=R2) {
    R3 = R3 / 32;
} else {
    R4 = R5 - 9;
}
R1 = R4;
```

a)




b)




***Criterio de corrección: Ambos apartados se corrigen conjuntamente. -0.25 puntos por el primer nodo incorrecto y -0.50 puntos por cada uno de los siguientes nodos incorrectos.***
**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. Las funcionalidades que sean iguales en ambos apartados y estén mál en ambos, sólo contaran como un error.**
**Hacemos una excepción a esta regla: Si falta la U o la I se descuenta 0.1 por ese nodo si sólo tiene ese fallo.**

***Ejercicio 2*** *(Objetivo 8.x)* **(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)* |
|---|---|---|---|---|---|---|---|---|---|
| SUBI R1, R2, -3 | **010** | **xxx** | **0** | **00** | **101** | **0** | **001** | **1** | **F F F D** |
| MOVEI R3, 0x00B7 | **xxx** | **xxx** | **0** | **10** | **001** | **0** | **011** | **1** | **0 0 B 7** |
| OUT R4 // IN R3 // XORI -, R4, 9 | **100** | **xxx** | **0** | **00** | **010** | **1** | **011** | **1** | **0 0 0 9** |

***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 (sin subsistema de I/O ni memoria). *(0.5 puntos)*

| ***Mnemotécnico*** | @A | @B | Rb/N | OP | F | In/Alu | @D | WrD | N *(hexa)* |
|---|---|---|---|---|---|---|---|---|---|
| **IN R0** | xxx | xxx | x | xx | xxx | 1 | 000 | 1 | X X X X |
| **ANDI -,R4,0xABCD** | 100 | xxx | 0 | 00 | 000 | x | xxx | 0 | A B C D |
| **NOT R0,R0** | 000 | xxx | x | 00 | 011 | 0 | 000 | 1 | X X X X |

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

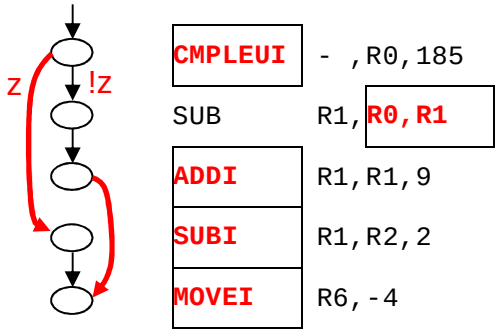
***Ejercicio 3*** *(Objetivos 8.x)* **(2 puntos)**
Dados los dos siguientes fragmentos de código en C (el código no tiene que hacer algo útil), indicad como se implementarían cada uno en un procesador que use la UPG vista en clase, utilizando la UC de **propósito específico** (UCe) y la UP de **propósito general** (UPG). Todos los datos son **naturales**.

Fragmento 1
```
if (R0<=185) { R1=R0-R1+9; }
else  { R1=R2-2; }
R6=-4;
```

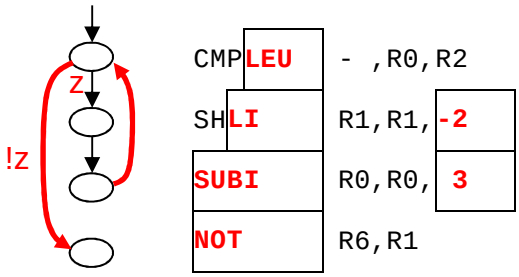
Fragmento 2
```
while (R0>R2) {
      R1=R1/4;
      R0=R0-3;
}
R6=not(R1);
```

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)*




***Criterio de corrección: Cada fragmento vale 0.5 puntos y se corrigen por separado. -0.25 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. Las funcionalidades que sean iguales en ambos apartados y estén mál en ambos, sólo contaran como un error.**
**Hacemos una excepción a esta regla: Si falta la U o la I se descuenta 0.1 por ese nodo si sólo tiene ese fallo.**

b) Completad los fragmentos de programa en lenguaje ensamblador SISA-I 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-I 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 | MOVI R7, 0xB9 |
| 0x0001 | MOVHI R7, 0x00 |
| 0x0002 | CMPLEU R7, R0, R7 |
| 0x0003 | BZ R7, 3 |
| 0x0004 | SUB R1, R0, R1 |
| 0x0005 | ADDI R1, R1, 9 |
| 0x0006 | BNZ R7, 1 |
| 0x0007 | ADDI R1, R2, -2 |
| 0x0008 | MOVI R6, -4 |

Fragmento 2 *(0.5 puntos)*

| @I-Mem | |
|---|---|
| 0x0000 | CMPLEU R7, R0, R2 |
| 0x0001 | BNZ R7, 4 |
| 0x0002 | MOVI R7, -2 |
| 0x0003 | SHL R1, R1, R7 |
| 0x0004 | ADDI R0, R0, -3 |
| 0x0005 | BNZ R7, -6 |
| 0x0006 | NOT R6, R1 |

***Criterio de corrección: Cada fragmento vale 0.5 puntos y se corrigen por separado. -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 4*** *(Objetivos X)* **(0.5 puntos)**

Completa la siguiente tabla ensamblando las instrucciones en ensamblador SISA-I 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-I.

| Lenguaje máquina SISA-I | Lenguaje ensamblador SISA-I |
|---|---|
| 0x616C | **STB -20(R0), R5** |
| **0xAC80** | IN R6, 128 |
| 0x3DFE | **LD R7, -2(R6)** |

***Criterio de corrección: -0.25 puntos por cada fila incorrecta.***

***Ejercicio 5*** *(Objetivos X)* **(1.25 puntos)**

Escribid sobre la siguiente tabla el valor de los bits que tiene la palabra de control del SISC-Harv uniciclo (UCG+UPG+subsistema IO+Memoria) durante el ciclo en que se ejecuta cada una de las instrucciones SISA-I que se indican. Poned x siempre no importe el valor de ese bit para la ejecución correcta de la instrucción (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 y salida y de la memoria de datos es cero.

| Instrucción SISA-I | 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)* |
| MOVHI R3, -2 | **011** | **xxx** | **0** | **10** | **010** | **00** | **011** | **1** | **0** | **0** | **0** | **x** | **0** | **FFFE** | **XX** |
| STB 6(R2), R0 | **010** | **000** | **0** | **00** | **100** | **xx** | **xxx** | **0** | **0** | **0** | **1** | **1** | **0** | **0006** | **XX** |
| BNZ R1, -4 | **001** | **xxx** | **x** | **10** | **000** | **xx** | **xxx** | **0** | **0** | **0** | **0** | **x** | **0** | **FFF8** | **XX** |
| IN R4, 198 | **xxx** | **xxx** | **x** | **xx** | **xxx** | **10** | **100** | **1** | **0** | **1** | **0** | **x** | **0** | **XXXX** | **C6** |
| LD R2,3(R0) | **000** | **xxx** | **0** | **00** | **100** | **01** | **010** | **1** | **0** | **0** | **0** | **0** | **0** | **0003** | **XX** |

***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*** *(Objetivos X)* **(1.25 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 0xA722, el contenido de todos los registros es 0xFFFC y que el contenido de todas las posiciones pares de la memoria de datos es 0x34 y el de todas las posiciones impares de la memoria de datos es 0x56. 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 | |
|---|---|---|
| LDB R5, -1(R4) | **R5=0x0056** | **PC=0xA724** |
| STB 6(R0), R3 | **$MEM_b$[0x0002]=0xFC** | **PC=0xA724** |
| BNZ R7, 6 | | **PC=0xA730** |
| SHL R4, R2, R0 | **R4=0x0FFF** | **PC=0xA724** |
| LD R4,0(R2) | **R4=0x5634** | **PC=0xA724** |

***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 7*** *(Objetivos X)* **(1.5 puntos)**

Indica el contenido de la tabla de la ROM (sólo las celdas sombreadas) 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/a1 | -/i/l/ao | $OP_1$ | $OP_0$ | MxN1 | MxN0 | MxF | f2 | f1 | f0 | MxD1 | MxD0 | |
| 0 | 0 | 0 | 0 | X | | | | | | | | | 0 | 0 | | | | | 0 | | | | 0 | 0 | A / L |
| 0 | 0 | 0 | 1 | X | | | | | | | | | 0 | 0 | | | | | 0 | | | | 0 | 0 | CMP |
| 0 | 0 | 1 | 0 | X | | | | | | | | | 0 | 0 | | | | | 1 | | | | 0 | 1 | ADDI |
| 0 | 0 | 1 | 1 | X | | | | | | | | | 0 | 1 | | | | | 1 | | | | 0 | 1 | LD |
| 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 |
| 0 | 1 | 0 | 1 | X | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | LDB |
| 0 | 1 | 1 | 0 | X | | | | | | | | | x | x | | | | | 1 | | | | x | x | STB |
| 0 | 1 | 1 | 1 | X | | | | | | | | | x | x | | | | | x | | | | x | x | - |
| 1 | 0 | 0 | 0 | 0 | | | | | | | | | x | x | | | | | 1 | | | | x | x | BZ |
| 1 | 0 | 0 | 0 | 1 | | | | | | | | | x | x | | | | | 1 | | | | x | x | BNZ |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | x | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | MOVI |
| 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 |
| 1 | 0 | 1 | 0 | 0 | | | | | | | | | 1 | 0 | | | | | x | | | | 1 | 0 | IN |
| 1 | 0 | 1 | 0 | 1 | | | | | | | | | x | x | | | | | x | | | | x | x | OUT |
| 1 | 0 | 1 | 1 | X | | | | | | | | | x | x | | | | | x | | | | x | x | - |
| 1 | 1 | X | X | X | | | | | | | | | x | x | | | | | x | | | | x | x | - |

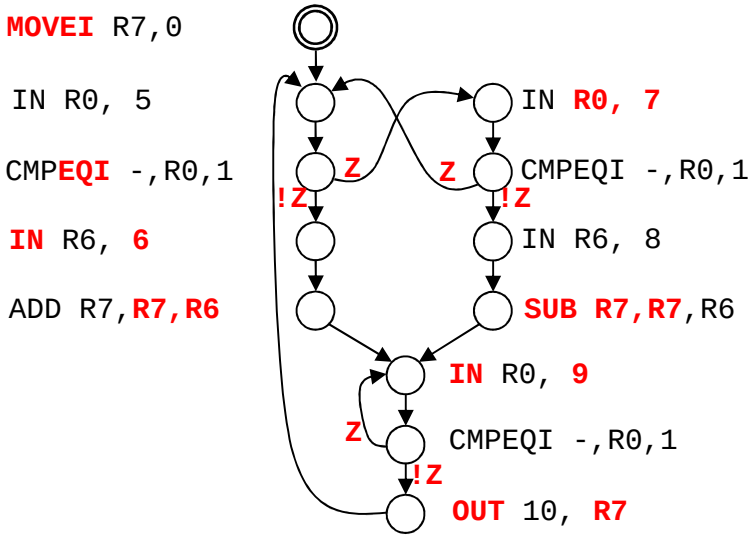
***Criterio de corrección: -0.25 puntos por cada fila y columna incorrecta, escogiendo el número mínimo de filas y/o columnas que cubren todos los errores.***

***Ejercicio 8*** *(Objetivos X)* **(1.5 puntos)**
Se han conectado a la UPG dos dispositivos externos de entrada (*dispA* y *dispB*) que nos envían valores enteros y un dispositivo externo de salida (*dispIMP*) al cual le podemos enviar datos. Todos estos dispositivos tienen un efecto lateral sobre su registro de estado.

Se desea realizar una función que vaya leyendo los datos de ambos dispositivos en el orden en el que se vayan detectando. Partiremos de un registro inicializado a cero, cada vez que llegue un valor por el *dispA* se le sumará a este registro y cada vez que llegue un valor por el *dispB* se le restará. A cada actualización del valor de este registro se enviará al *dispIMP*.

Suponiendo que el dispositivo *dispA* tiene el registro de status en la dirección 5 del espacio de direccionamiento de entrada y el de datos en la 6; que el *dispB* tiene el registro de status en la dirección 7 y el de datos en la 8; y que el *dispIMP* tiene el registro de status en la dirección 9 de entrada y el de datos en la 10 del espacio de direccionamiento de salida, completad:

a) 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. *(0.75 puntos)*




***Criterio de corrección: -0.25 puntos por cada nodo incorrecto.***

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

**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. Las funcionalidades que sean iguales en ambos apartados y estén mál en ambos, sólo contaran como un error.**
**Hacemos una excepción a esta regla: Si falta la U o la I se descuenta 0.1 por ese nodo si sólo tiene ese fallo.**

b) el fragmento de código ensamblador SISA-I para que realice la función anteriormente descrita. *(0.75 puntos)*

```
      MOVI R7,0
lecA: IN   R0, 5        (statusA)
      BZ   R0, 3        (lecB)
      IN   R6, 6        (datoA)
      ADD  R7,R7,R6
      BNZ  R0, 4        (impr)
lecB: IN   R0, 7        (statusB)
      BZ   R0, -7       (lecA)
      IN   R6, 8        (datoB)
      SUB  R7,R7,R6
impr: IN   R0, 9        (statusIMP)
      BZ   R0, -2       (impr)
      OUT  10, R7       (datoIMP)
      BNZ  R0, -13      (lectA), también puede ir a lecB (-8)
```

***Criterio de corrección: -0.25 puntos por cada instrucción incorrecta.***