Examen EP2 (*Temas 9, 10, 11, 12, 13 y 14)*

- Duración del examen: 2 horas y 30 minutos.
- Los problemas tienen que resolverse en las **HOJAS DE RESPUESTAS**.
- No podéis utilizar calculadora, móvil, apuntes, etc.
- La solución y las notas se publicarán en Atenea mañana 10 de enero del 2023.
- La revisión será a las 12:00 del 11 de enero del 2023 (pasado mañana miércoles) en el aula C6-E101.

Ejercicio 1 *(1.6 puntos)*

El programa en ensamblador SISA que se muestra a continuación se ha traducido a lenguaje máquina situando la sección de datos (`.data`) a partir de la dirección `0x8354` y justo a continuación la sección de código (`.text`).

```
      long=4
      next=2
.data
      vect: .word 2,3,4,5,6,7,8
      aux:  .space 20, 25
            .even
      res:  .space long*2
.text
      movi  R5,lo(vect)
      movhi R5,hi(vect)
      movi  R6, long
loop: LD R4, 0(R5)
      SHL R4, R4, R6
      ST 0x14(R5), R4
      ADDI R5, R5, next
      ADDI R6, R6, -1
      BNZ R6, loop
.end
```

Una vez ensamblado y cargado el programa en la memoria y antes de comenzar su ejecución:

a) ¿A qué direcciones de memoria corresponden las etiquetas o direcciones simbólicas siguientes? *(0.4 puntos)*

`res=0x` `loop=0x`

b) ¿Cuál es la dirección de memoria y su contenido donde han quedado almacenadas las siguientes instrucciones una vez cargado el programa? *(0.4 puntos)*

Instrucción	@ memoria y contenido
`MOVHI R5, hi(vect)`	$Mem_w[0x\quad\quad] = 0x$
`BNZ R6, loop`	$Mem_w[0x\quad\quad] = 0x$

c) Una vez ejecutado el programa en el SISC Von Neumann ¿Cuál es la dirección de la memoria donde ha escrito la última instrucción (`ST 0x14(R5),R4`) y cuál es su contenido? *(0.4 puntos)*

$Mem_w[0x\quad\quad]=0x$

d) Una vez ejecutado el programa en el SISC Von Neumann ¿Cuántas instrucciones se han ejecutado del código que se muestra y cuántos ciclos ha tardado la ejecución completa del programa? (desde el inicio hasta que el registro PC tenga el valor de la etiqueta end). *(0.4 puntos)*

Nº de instruc. ejecutadas =

Nº de ciclos =

Ejercicio 2 *(0.6 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**
`0x532E`	
	`BZ R4, -19`
`0x9877`	

Ejercicio 3 *(2 puntos)*

Cada uno de los apartados pregunta sobre un ciclo concreto de la ejecución de varias instrucciones en el SISC Von Neumann. Escribid el contenido del registro IR, el contenido de la ROM_OUT y 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**.** 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. El programa ensamblador, sustituye los bits no usados del formato de instrucción por 0.

a) Indica el contenido del registro IR. Utilizad el valor 0 para los bits que sean x (solo para este apartado). *(0.4 puntos)*

Nodo / Estado (Mnemo Salida)	**Instrucción en IR** (en ensamblador)	**Valor del IR** (en hexadecimal)
Jalr	`JALR R3,R4`	0x
Ld	`LD R4,6(R1)`	0x
Movhi	`MOVHI R7,127`	0x
Bnz	`BNZ R2,5`	0x

b) Indica el contenido de la ROM_OUT en hexadecimal usando las conexiones en el orden que están en el anexo. Utilizad el valor 0 para los bits que sean x (solo para este apartado). *(0.8 puntos)*

Nodo / Estado (Mnemo Salida)	**Instrucción en IR** (en ensamblador)	**Contenido ROM_OUT** (en hexadecimal)
Jalr	`JALR R3,R4`	0x
Ld	`LD R4,6(R1)`	0x
Movhi	`MOVHI R7,127`	0x
Bnz	`BNZ R2,5`	0x

c) Indica el valor de los de algunos 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 podemos suponer cómo se han implementado las x en la ROM_OUT). *(0.8 puntos)*

Apartado	**Nodo / Estado** (Mnemo Salida)	**Instrucción en IR** (en ensamblador)	**Palabra de Control** @B	Pc/Rx	Ry/N	OP	P/I/L/A	@D	LdPc	Byte	R@/Pc	N (hexa)	ADDR-IO (hexa)
a	Jalr	`JALR R3,R4`											
b	Ld	`LD R4,6(R1)`											
c	Movhi	`MOVHI R7,127`											
d	Bnz	`BNZ R2,5`											

Ejercicio 4 *(0.4 puntos)*

Indicad qué cambios se producen 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 `0x856A`, el contenido de todos los registros pares es `0x2468` y el de los registros impares es `0x1357`, el byte contenido en todas las direcciones pares de la memoria es `0x85` y el de todas las impares es `0x91` 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... y/o MEM_w[0x…]=0x... para indicar cualquier cambio en la memoria y PORTIN[0x...]=0x... para los puertos de entrada y PORTOUT[0x...]=0x... para los de salida.

Instrucción a ejecutar	**Cambios en el estado del computador**
`ST -5(R3), R6`	
`BNZ R0, -6`	
`LDB R4,4(R2)`	
`MOVI R6, 126`	

Ejercicio 5 *(0.6 puntos)*

Queremos crear un programa en ensamblador SISA que haga lo mismo que el siguiente programa en C.

```
int subrutina(int a, int b){
      return (a+b)*2;
}

int main() {
      int c,d;
      c=subrutina(14, 35);
      d=subrutina(-8, 73);
      out(21,c+d);
}
```

El programador que lo estaba haciendo se ha ido de la empresa y lo ha dejado incompleto. El código implementado hasta ahora en SISA es el siguiente y desconocemos las instrucciones que irían en los recuadros etiquetados con la letra A (el código de ambos recuadros es exactamente el mismo):

```
subr:  add  R0, R1, R2
       add  R4, R0, R0
       jalr R6, R6
main:  movi R1, 14
       movi R2, 35
       [ A ]
       addi R5, R4, 0
       movi R1, -8
       movi R2, 73
       [ A ]
       add R3, R5, R4
print: out 21, R3
```

El código de la subrutina podría cambiar en un futuro el número de instrucciones o la ubicación en la memoria, pero siempre acabará con la instrucción JALR R6, R6. La rutina principal *main* no cambiará. Indica que instrucciones de código ensamblador SISA irían en el recuadro con la etiqueta A para que funcione correctamente ahora y en un futuro donde el contenido de la subrutina hubiese sido modificado.

Ejercicio 6 *(1.6 puntos)*

Especificad el **camino crítico** (indicando la suma ordenada de los tiempos de propagación de los bloques por los que pasa) y calculad el **tiempo de ciclo mínimo** para que el computador **SISC Von Neumann** pueda ejecutar correctamente el tipo de instrucción SISA que se indica en cada apartado (este sería el tiempo de ciclo mínimo del computador si solo ejecutara instrucciones como la indicada u otras que requieran menor tiempo). No tenéis que añadir ningún porcentaje de seguridad en el cálculo del tiempo de ciclo mínimo. Suponed que los tiempos de propagación de los bloques que forman el computador son los siguientes:

T_p(ROM_Q+) = 50 u.t.

T_p(ROM_OUT) = 70 u.t.

T_p(MUX-2-1) = 50 u.t.

T_p(MUX-4-1) = 100 u.t.

T_p(REG) = 100 u.t. // Tiempo de propagación de un registro.

T_p(REGFILE) = 250 u.t. // Tiempo de lectura del banco de registros

T_p(ALU-slow) = 700 u.t. // Tp de la ALU para las operaciones/funciones lentas: ADD, SUB, CMP*.

T_p(ALU-quick) = 350 u.t. // Tp de la ALU para las operaciones/funciones rápidas: cualquier otra distinta de ADD, SUB, CMP*.

T_{acc}(64Kb MEMORY) = 900 u.t. // Tiempo de acceso (para la lectura o escritura) a la memoria

T_p(AND-2) = T_p(OR-2) = 20 u.t.

T_p(NOT) = 10 u.t.

El tiempo de propagación de un bloque combinacional (T_p) y el tiempo de acceso a memoria para realizar una lectura (T_{acc}) es el tiempo desde que están estables todas las entradas necesarias hasta que se estabilizan las salidas requeridas al valor correcto para las entradas aplicadas. Desconocemos como se han implementado internamente los bloques (y podría ser de forma diferente a los vistos en clase). Recordad que un registro con señal de carga (Ld), REGwLd, está construido con un REG y un MUX-2-1 (no os damos el esquema interno del REGwLd, porque lo tenéis que saber).

a) T_c correspondiente al nodo de **F** (Fetch).

b) T_c correspondiente al nodo de **Addr**.

c) T_c correspondiente al nodo de **Ldb**.

d) T_c correspondiente al nodo de **Movhi**.

Ejercicio 7 *(3.2 puntos)*

Completad el diseño del SISC Von Neumann para que pueda ejecutar, además de las 25 instrucciones originales SISA, una nueva instrucción **ADDBZ**. Se trata de un salto condicional que permite romper el secuenciamiento implícito de las instrucciones y a la vez realizar una suma. Cuyo formato y codificación, la sintaxis ensamblador y la semántica son las siguientes:

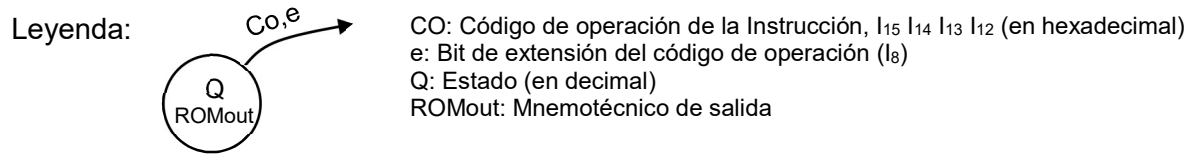
```
Codificación:  1011 ???????????
Sintaxis:      ADDBZ Rd, Rb, N5
Semántica:     PC=PC+2; Rd=Rd+Rb; if (Rd==0) {PC=PC+SE(N5)*2}
```

La instrucción realiza la suma de dos los registros y lo almacena en el registro destino. Si el resultado de esta suma es igual a 0, entonces rompe el secuenciamento implícito saltando, a partir de la siguiente instrucción, tantas instrucciones como indique la constante N5 (similar a los saltos BZ y BNZ). Se pide:

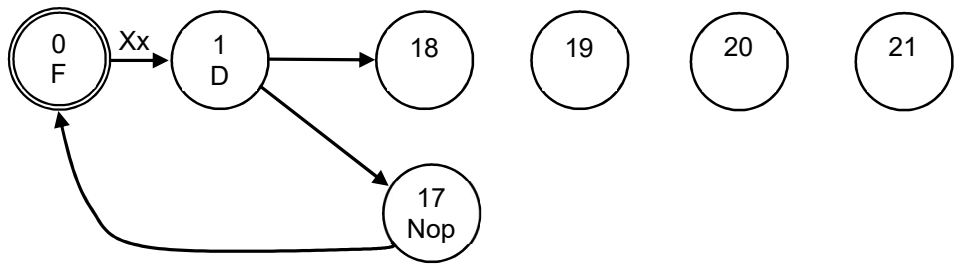
a) Definid el formato de la nueva instrucción **ADDBZ** para que permita implementar la nueva instrucción sin necesidad de modificar el hardware del procesador SISC Von Neumann. Dad el valor ((0, 1, d, a, b, n, x) a cada uno de los bits marcados con un '?' en el formato binario de la codificación de la instrucción. `N5` es un campo de la instrucción que se deberá identificar en el formato de instrucción con `nnnnn` y representa un número entero codificado en 5 bits en Ca2. *(0.4 puntos)*

I_{15} I_{14} I_{13} I_{12}	I_{11} I_{10} I_9 I_8 I_7 I_6 I_5 I_4 I_3 I_2 I_1 I_0
1 0 1 1	

b) Completad el fragmento del grafo de estados de la figura correspondiente al circuito secuencial de la unidad de control para la instrucción **ADDBZ**. Se da la leyenda del grafo y los nodos necesarios para ejecutar las nuevas instrucciones, pero faltan arcos y etiquetas. Dibujad todos los arcos que faltan y todas las etiquetas. En número de nodos necesarios para ejecutar la nueva instrucción dependerá de la solución que propongáis, pero no deberían ser más de 4 y deberá ser coherente con las acciones que pongáis en el apartado c) y la definición del formato del apartado a). Los nodos que os sobren podéis dejarlos sin conectar. *(0.4 puntos)*



Grafo:



c) Completad el contenido 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 ciclos/nodos que requiere la ejecución de la nueva instrucción **ADDBZ** (Fetch, Decode, y los ciclos/nodos de la ejecución propiamente dicha). Para especificar las acciones usad el mismo lenguaje de transferencia de registros que en la documentación. En número de filas/nodos de la tabla a rellenar dependerá de la solución que propongáis y deberá ser coherente con el apartado b). Nota: Existe una solución solo con 4 nodos (F, D y 2 nodos nuevos). *(1 punto)*

Nodo	Mnemotécnico	Acciones
E0	F	IR ← Mem_w[PC] // PC ← PC+2
E1	D	RX ← Ra // RY ← Rb // R@ ← PC+SE(N8)*2
E18		
E19		
E20		
E21		

d) Completad (poniendo 0, 1 o x en cada bit) las filas de la tabla que especifican el contenido de la ROM_OUT para las direcciones de los nodos que hayáis usado para implementar la nueva instrucción. **Poned x siempre que el valor de un bit no importe**. *(1 punto)*

@ROM	Bnz	Bz	WrMem	RdIn	WrOut	WrD	LdIr	Byte	R@/Pc	Alu/R@	Pc/Rx	Ry/N	$P/I/L/A_1$ $P/I/L/A_0$	OP_1 OP_0	MxN_1 MxN_0	MxF	F_2 F_1 F_0	$Mx@D_1$ $Mx@D_0$	***Nodo***
18																			
19																			
20																			
21																			

e) Completad la dirección o contenido, según corresponda, de la ROM_Q+ del SISC Von Neumann. *(0.4 puntos)*

`ROM_Q+[0x0AC]=0x` `ROM_Q+[0x ]=0x0F`