Examen E4 (temas 12, 13 y 14)

- Duración del examen: 2 horas.
- Los problemas tienen que resolverse en las HOJAS DE RESPUESTAS.
- No podéis utilizar calculadora, móvil, apuntes, etc.
- La solución del examen se publicará en Atenea mañana por la tarde.

Ejercicio 1 (2.6 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 `0x0100` y la sección de código (.text) a partir de la dirección `0xC000`. Antes de la primera instrucción que se muestra (`MOVI R5,0`) hay 150 instrucciones que no se muestran y después de la última instrucción que se muestra (`ST 0(R2),R5`) hay 20 instrucciones que tampoco se muestran.

```
.data
       long=4
       A: .word 1,-4, 15, 0, 12
       B: .word 13, 2, 14, -2, 9
       C: .space A
       D: .space 1
.text
       . . .
       MOVI  R5, 0
       MOVI  R1, lo(A)
       MOVHI R1, hi(A)
       MOVI  R2, lo(C)
       MOVHI R2, hi(C)
       ADDI  R0, R1, long
loop:  LD    R3, 0(R1)
       ADDI  R1, R1, 2
       LD    R4, long(R1)
       ADD   R7, R3, R4
       ADD   R3, R3, R7
       ST    0(R2), R3
       ADDI  R2, R2, 2
       ADD   R5, R5, R3
       CMPLEU  R7, R1, R0
       BNZ   R7, loop
endlp: ST    0(R2), R5
       . . .
.end
```

a) Una vez cargado el programa en la memoria y antes de comenzar su ejecución ¿Cuál es la dirección de la memoria y su contenido donde han quedado almacenadas las siguientes instrucciones? *(0.6 puntos)*

Instrucción	@ memoria Mem[0x????]	Contenido memoria
`MOVHI R2, hi(C)`	0x	0x
`LD  R4, long(R1)`	0x	0x
`BNZ R7, loop`	0x	0x

b) Una vez ejecutado el programa en el SISC Von Neumann ¿Cuál es la dirección de la memoria de datos donde ha escrito la última instrucción (`ST 0(R2),R5`) y cuál es su contenido? Suponed que las instrucciones que hay antes de la primera instrucción del fragmento de programa (`MOVI R5,0`) no modifican el estado del computador excepto el PC. *(0.4 puntos)*

c) ¿Cuántas instrucciones se ejecutan del código que se muestra y cuantos ciclos tarda desde se ejecuta la primera instrucción del código (`MOVI R5,0`) hasta que se ejecuta la última instrucción del código (`ST 0(R2),R5`), ambas incluidas? ¿Cuánto tarda en ejecutarse el código en el Harvard uniciclo, en el Hardard multiciclo y en el Von Neumann suponiendo que los tiempos de ciclo son 3.000, 2000 y 1.000 u.t. respectivamente? *(0.7 puntos)*

Nº de instruc. ejecutadas =

Nº de ciclos (H. uniciclo)= Nº de ciclos (H. multiciclo)= Nº de ciclos (Von Neumann)=

Tejec(Harvard uniciclo) = Tejec(Harvard multiciclo) = Tejec(Von Neumann) =

d) Supongamos que el código ha cambiado y entre la primera instrucción del bucle y la instrucción de salto hay más de 200 instrucciones.

```
.text
       . . .
loop:  LD    R3, 0(R1)
       . . .                ; suponed que hay 200 instrucciones
       CMPLEU  R7, R1, R0
       BNZ   R7, loop
endlp: ST    0(R2), R5
       . . .
.end
```

Ahora la instrucción de salto ya no puede ser directamente un BNZ ya que esta fuera del alcance de la constante que se puede codificar en el formato de instrucción de los saltos. Suponed que el PC de la instrucción BNZ vale 0x8500, ¿Cuál sería la dirección de la instrucción anterior y posterior más alejadas a las que podríamos acceder con la instrucción de salto? *(0.4 puntos)*

@Mem instrucción anterior = 0x

@Mem instrucción posterior = 0x

e) Una posible solución a este inconveniente es la que se presenta incompleta a continuación. Completa los huecos que faltan para que el código siga funcionando aunque haya más de 200 instrucciones entre origen y el destino del salto. *(0.5 puntos)*
Recordad que la semántica del JALR es:

Sintaxis ensamblador:	`JALR Rd, Ra`
Semántica:	`PC = PC+2; tmp=Ra&(~1); Rd=PC; PC=tmp;`
Lenguaje máquina:	`0111 aaa ddd xxxxxx`

```
loop:  LD    R3, 0(R1)
       . . .                ; suponed que hay 200 instrucciones
       CMPLEU  R7, R1, R0
       BZ    R7, endlp

       ________________

       ________________
       JALR  _________
endlp: ST    0(R2), R5
```

Ejercicio 2 **(1.2 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 0x26AC, el contenido de todos los registros pares es 0x6789 y el de los registros impares es 0x0123, el byte contenido en todas las direcciones pares de la memoria es 0x64 y el de todas las impares es 0x32 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 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 9(R2), R5`	
`BZ R0, -6`	
`MOVHI R6, 94`	

Ejercicio 3 (4.7 puntos)

Una de las sucesiones más simples y famosas de las matemáticas es la sucesión de Fibonacci (a veces mal llamada serie de Fibonacci) que es la sucesión infinita de números naturales como la siguiente: 0,1,1,2,3,5,8,13,21,34,55,89,144,233,377, … La sucesión comienza con los números 0 y 1, (f_0=0 y f_1=1) y a partir de estos, cada elemento de la sucesión es la suma de los dos anteriores ($f_n=f_{n-1}+f_{n-2}$). A los elementos de esta sucesión se les llama números de Fibonacci.

a) Completad el código SISA para que calcule 500 elementos de la sucesión de Fibonacci y los almacene en memoria. Suponed que en la posición de memoria 12346 ya se encuentra el primer valor de la sucesión (f_0) y en la siguiente posición, la 12348, el segundo valor (f_1). El código debe rellenar las siguientes 500 posiciones de memoria con los siguientes valores de la sucesión. Los valores son números naturales de 16 bits. En caso de que el valor de la sucesión calculado exceda los 16 bits el programa se queda con los 16 bits de menor peso. *(1.0 puntos)*

Nota: Para saber qué acciones concretas hace el algoritmo os puede ayudar leer el apartado c) de este ejercicio

@Mem				
0x0000	.data			
0x0001	.text	MO___	R5, _____	; R5=primera posición de memoria
0x0002		MO___	R5, _____	
0x0003		MO___	R6, _____	; R6=última posición de memoria
0x0004		MO___	R6, _____	
0x0005		LD	R0, _____	; primer valor de la sucesión
0x0006		_____	R1, ____(R5)	; segundo valor de la sucesión
0x0007	bucle:	A_____	R2, R0, ___	
0x0008		_____	0(____),___	
0x0009		_____	R0, R1, ___	
0x000A		_____	R1, R2, ___	
0x000B		A_____	R5, R5,____	
0x000C		CMPLE	R7, R5, ___	
0x000D		_____	R7, _____	
0x000E	.end			

b) ¿Cuántos ciclos y unidades de tiempo tarda en ejecutarse el código anterior en Harvard uniciclo y en el Von Neumann, considerando que el tiempo de ciclo del Harvard uniciclo es de 4000 u.t. y el del V.Neumann es de 1000 u.t.? *(0.4 puntos)*

Código en el Harvard uniciclo: Número de ciclos= Tejec=

Código en el Von Neumann: Número de ciclos= Tejec=

c) Queremos acelerar la ejecución del código. Para ello vamos a modificar el computador Von Neumann para añadirle una nueva instrucción que haga el cálculo de los números de Fibonacci como la siguiente:

Binario: `1011 aaa bbb ddd xxx`
Ensamblador: `FIB Rd, Ra, Rb`
Semántica: `Rd = Ra + Rb ; Ra = Rb ; Rb = Rd`

Completad el diseño del SISC Von Neumann para que pueda ejecutar, además de las 25 instrucciones originales SISA, la nueva instrucción FIB, que tiene el formato y codificación, la sintaxis ensamblador y la semántica anteriores.

Para su implementación, además de modificar el contenido de la ROM_OUT y de la ROM_Q+, cosa imprescindible para añadir una nueva instrucción, se le añade a la unidad de control un MUX-2-1 (con señal de selección Mx, que genera la ROM_OUT, y que solamente valdrá 1 en alguna de las fases de ejecución de FIB). La ejecución de la instrucción `FIB` solamente requiere 5 ciclos de ejecución, nodos F, D, Fib1, Fib2 y Fib3. Todos los registros (Ra, Rb y Rd) deben ser distintos para que la instrucción se ejecute correctamente en 5 ciclos. Se pide:

c1) Indica claramente, con texto y/o con un dibujo, donde se conectan las entradas de datos 0 y 1 del MUX-2-1 y donde se conecta su salida, para el correcto funcionamiento del computador con la nueva instrucción FIB. Si se responde incorrectamente este apartado, el resto del apartado (subapartados c2, c3 y c4) se considerará incorrecto. *(0.5 puntos)*

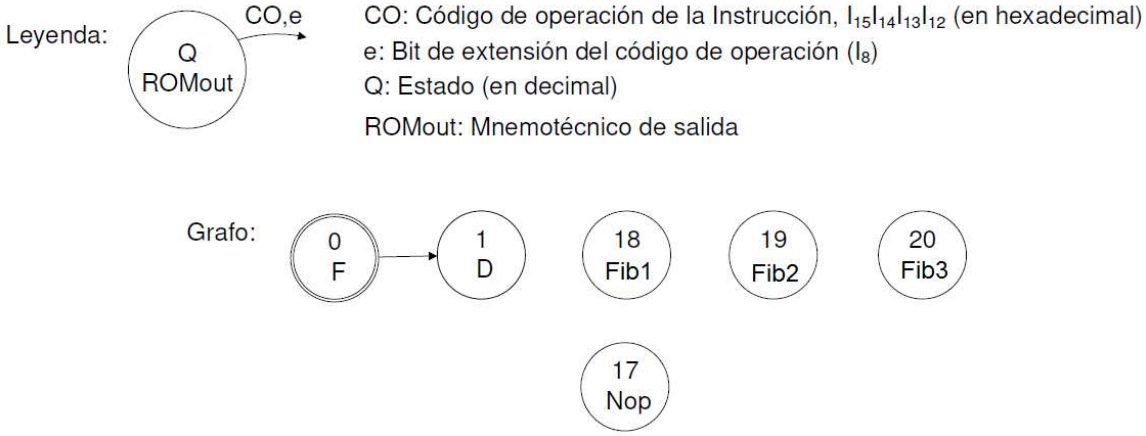
c2) Completad el contenido de la tabla que indica, mediante una fila para cada nodo, la acción o acciones en paralelo que se realiza en el computador en cada uno de los ciclos/nodos que requiere la ejecución, propiamente dicha, de la nueva instrucción (Fib1, Fib2 y Fib3). Usad el mismo lenguaje de transferencia de registros que en la documentación. *(0.6 puntos)*

Nodo / Estado		Acciones
Número	Mnem.	
18	Fib1	
19	Fib2	
20	Fib3	

c3) Completad (poniendo 0, 1 o x en cada bit) las tres filas de la tabla que especifican el contenido de la ROM_OUT para las direcciones 18 (Fib1) y 19 (Fib2) y 20 (Fib3). **Poned x siempre que el valor de un bit no importe**. *(0.6 puntos)*

@ROM	Mx	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	F_2	F_1	F_0	Mx@D_1	Mx@D_0	
18																									Fib1
19																									Fib2
20																									Fib3

c4) Completad el fragmento del grafo de estados del circuito secuencial de la unidad de control de la figura. Se da la leyenda del grafo y todos los nodos necesarios para ejecutar las dos nuevas instrucciones, pero faltan arcos y etiquetas. Dibujad todos los arcos que faltan y todas las etiquetas. Os pedimos que no dibujéis ningún otro nodo. *(0.5 puntos)*



d) 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 ]=0x0C`

e) Reescribe el código correspondiente al bucle del apartado a) que haga la misma tarea con menos instrucciones y usando la nueva instrucción FIB. *(0.5 puntos)*

f) ¿Cuántos ciclos y unidades de tiempo tarda ahora en ejecutarse el código completo con la modificación escrita en el apartado e) en el nuevo SISC Von Neumann (Tc=1000 u.t.)? *(0.2 puntos)*

Número de ciclos= Tejec=

Ejercicio 4 (1.5 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 / Estado** (Mnemo 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	F	`JALR R1,R2`																			
b	Bz	`BZ R6, -2`																			
c	Addr	`LDB R2,3(R1)`																			