Apellidos y Nombre: .....Grupo:.....DNI: .....

Examen 4 (temas 12, 13 y 14)

- 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. En hoja aparte se os da una "chuleta" con información útil para realizar los ejercicios.
- La solución del examen se publicará en Atenea mañana por la tarde y las notas antes del próximo lunes 6 de junio a las 10 de la mañana. A las 11:00 se publicaran las notas finales provisionales (medias ponderadas de los parciales y el Lab,) y a las 12, solo una hora más tarde, se realizará la revisión de estas notas, que incluye la revisión del E4.

Ejercicio 1 **(3 puntos)**

El programa en ensamblador, una vez completado, se traducirá a lenguaje máquina para ser ejecutado en el SISC Von Neumann situando la sección `.data` a partir de la dirección 256 de memoria y a continuación la sección `.text` (esta sección tiene **20 instrucciones que no se muestran** antes de la etiqueta `L1`). Puede ser que haya más etiquetas de las necesarias.

```
.data
        N=1000
V:      .byte 23,57,40,51,253
        .space 995, 0
        .even
        Masc=0x
NumP:   .space 2
.text
   • • •
L1:     M       R0,
        M       R0,
        M       R1,
        M       R1,
        M       R2,
        A       R3,
L2:     L       R7,
        A       R7,
        S       R3,
L3:     A       R1,
        A       R0,
L4:     BNZ     R0, L2
        M       R7,
        M       R7,
   • • •
.end
```

a) Completad el fragmento de código ensamblador SISA para que escriba en `NumP` (para ser usado en partes del programa que no se muestran) el número de datos (donde cada dato es un número natural codificado en binario en un byte de memoria), de entre los `N`, del vector `V` que son pares (un número par codificado en binario tiene el bit de menor peso a 0). El núcleo del código consta de un bucle de `N` iteraciones, una por dato. Todas las iteraciones ejecutan las mismas instrucciones, independientemente de que el dato analizado sea par o impar. En cada iteración se accede al dato, se observa si es impar y se actualiza el registro `R3` (que después de las `N` iteraciones contendrá el número de datos pares). Finalmente, se escribe R3 en la dirección de memoria `NumP`. Antes de entrar en el bucle se carga N en `R0` y en `R3` (en cada iteración se actualiza `R0` con el número de datos que faltan por procesar). En `R1` se carga la dirección, `V`, del primer dato a analizar, `V[0]`, y se actualiza antes de terminar cada iteración para que apunte al siguiente dato. `R2` se usa para mantener una máscara, invariante en cada iteración, necesaria para detectar si el dato es impar. Se usa `R7` para almacenar valores temporales. No se usa ningún otro registro. (**1 punto**)

b) Una vez cargado el programa en memoria, ¿A qué dirección de memoria corresponde cada etiqueta y qué palabra (word) contiene? (**0,75 puntos**)

`V:`	=>	Mem_w[0x	] = 0x
`L1:`	=>	Mem_w[0x	] = 0x
`L4:`	=>	Mem_w[0x	] = 0x

c) Cuándo se ejecuta la instrucción `L3` por tercera vez ¿Cuál es, en este momento, el valor contenido en `R3`? (**0,5 puntos.**) `R3 =`

d) Del código mostrado, ¿cuántas instrucciones se ejecutan? ¿Cuánto tardan en ejecutarse en el Harvard uniciclo y en el Von Neumann suponiendo tiempos de ciclo de 2.000 y 1.000 u.t. respectivamente? ¿Cuánto vale x para que sea cierta la siguiente afirmación? "El computador Von Neumann es un x% **más lento** que el Harvard uniciclo, para este código.(**0,75 punt.**)

InstrucEjec = ; Tejec(V.Neumann) = ; x =

Ejercicio 2 **(1,25 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	D	`STB 0x2C(R2),R7`																			
b	Bz	`BZ R5,-61`																			
c	Addr	`LD R6,-21(R2)`																			

Ejercicio 3 **(1,5 puntos)**

Completad la tabla que representa en forma compacta parte del contenido de la ROM_OUT de la unidad de control del SISC Von Neumann. Poned x siempre que un bit pueda valer tanto 0 como 1.

@ROM	Bnz	Ldlr	R@/Pc	Alu/R@	Pc/Rx	Ry/N	MxN1	MxN0	MxF	Mx@D1	Mx@D0	
0												F
9												Stb
11												Bz
14												Movhi

Ejercicio 4 **(1,25 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 0xF0F8, el contenido de todos los registros es 0x6789 y que el byte contenido en todas las direcciones pares de la memoria es 0x83 y el de todas las impares es 0x01. Utilizad la notación $MEM_b[\ldots]$=... y/o $MEM_w[\ldots]$=... para indicar los cambios en la memoria.

Instrucción a ejecutar	Cambios en el estado del computador
`STB -3(R3), R5`	
`MOVI R7, 0x93`	
`BNZ R2, 0x81`	

Ejercicio 5 **(3 puntos)**

Completad el diseño del SISC Von Neumann para que pueda ejecutar, además de las 25 instrucciones originales SISA, la nueva instrucción `SETIM+`, que tiene el formato y codificación (incompleto), la sintaxis ensamblador y la semántica siguientes:

```
Codificación:  1111 ??? ??? ??????
Sintaxis:      SETIM+  Rd, (Ra)     ; Ra y Rd no deben ser el mismo registro
Semántica:     if (Mem_b[Ra]<0> == 1) Rd = 1 else Rd = 0; Ra=Ra+1;
```

Donde `Mem_b[Ra]<0>` hace referencia al bit de menor peso del byte que se encuentra en la dirección de memoria contenida en `Ra`. Al ejecutarse la instrucción escribe en `Rd` un 1 si el byte almacenado en memoria en la dirección contenida en `Ra`, interpretado como un número natural codificado en binario con 8 bits, es impar y escribe un 0 si es par. Además, después de acceder a memoria, incrementa en 1 el contenido de `Ra`, para dejarlo apuntando al siguiente byte de memoria.

La instrucción deja de tener sentido si se usa el mismo registro como `Ra` y como `Rb`: no tiene sentido `SETIM+ R7, (R7)`

Para poder ejecutar la nueva instrucción solo hay que añadir un nuevo multiplexor de buses, MUX-2-1, en la unidad de control (con señal de selección Mx, que valdrá 0 en el nodo/ciclo Decode, D) y modificar parte del contenido de la ROM_Q+ y de la ROM_OUT. Se pide:

a) Proponed el formato y la codificación de la nueva instrucción (cada bit puede valer `0`, `1`, `a`, `b`, `d`, `n`, `x`) sabiendo que el campo del código de operación es: `1111` (que en el SISA original no se usaba). Solo hay una posible codificación que permite implementar esta instrucción con las restricciones enunciadas: es la clave para poder hacer bien el resto de apartados. (**0,25 puntos**)

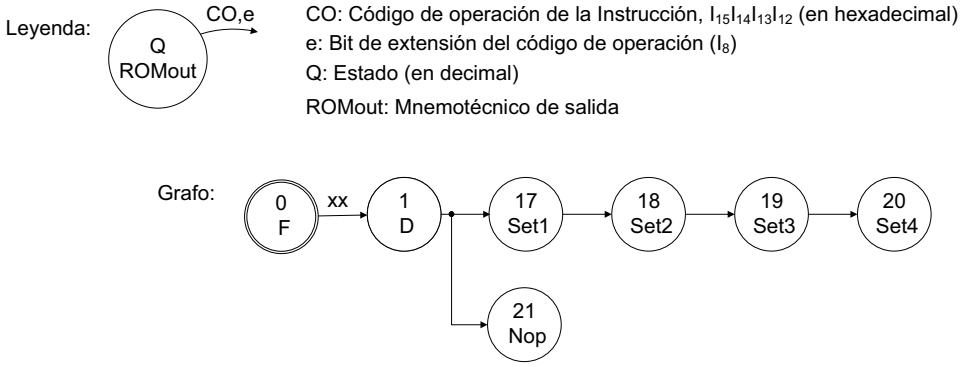
`Codificación:` `1111`

Apellidos y Nombre: .....Grupo:.....DNI: .....

b) Completad el fragmento del grafo de estados del circuito secuencial de la unidad de control necesario para ejecutar completamente la nueva instrucción SERIM+ y todas las que tienen un código ilegal. Se da la leyenda del grafo y todos los nodos necesarios, pero solo un arco con su etiqueta. Dibujad todos los arcos que faltan con sus etiquetas.

Completad, también, 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 6 ciclos/nodos que requiere la ejecución de la nueva instrucción (Fetch, Decode, y los 4 ciclos/nodos de la ejecución propiamente dicha): F, D, Set1, Set2, Set3 y Set4. Para especificar las acciones usad el mismo lenguaje de transferencia de registros que en la documentación.

Completad también la frase enmarcada sobre el nuevo MUX-2-1. (**1,5 = 0,25 + 1 + 0,25 puntos**))



Nodo/Estado		Acciones
Número	Mnem.	
0	F	
1	D	
17	Set1	
18	Set2	
19	Set3	
20	Set4	

El nuevo MUX-2-1 con señal de selección Mx se usa para generar el campo/bit ..... de la palabra de control

c) Completad (poniendo 0, 1 o x en cada bit) las filas de la tabla que especifica parte del contenido de la ROM_OUT para que se ejecuten correctamente todas las instrucciones, poniendo el máximo número de x posibles. La dirección 0 de la ROM corresponde al estado 0 (F), la 1 al 1 (D)... la dirección 17 al estado 17 (Strr1)... la dirección 19 al estado 19 (Strr3) y de la 20 a la 31 el estado Nop (No operación, que se usa para que las instrucciones con un código de operación ilegal no modifiquen el estado del computador y pasen a ejecutar la siguiente instrucción en secuencia). (**1 punto**)

@ROM	Mx	Bnz	Bz	WrMem	RdIn	WrOut	WrD	LdIr	Byte	R@/Pc	Alu/R@	Pc/Rx	Ry/N	P/I/L/A1	P/I/L/A0	OP1	OP0	MxN1	MxN0	MxF	F2	F1	F0	Mx@D1	Mx@D0	
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
1																										D
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
17																										Set1
18																										Set2
19																										Set3
20																										Set4
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…

d) Indicad la dirección o las direcciones de la memoria ROM_Q+ y su contenido o sus contenidos para implementar correctamente el paso del nodo/estado 1 al 17. (**0,25 puntos**)