- Duración del examen: 2 horas
- La solución a cada ejercicio debe escribirse en el espacio reservado para ello en el propio enunciado.
- No podéis utilizar calculadora, móvil, apuntes, ...
- La solución al examen se publicará mañana en Atenea y las notas se publicarán en una semana

Ejercicio 1 (1,5 puntos) Criterio: 0,5 por fila, 0,4 con una señal errónea, 0,25 con dos señales erróneas, 0 si hay 3 o más errores

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.

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	AI	`ADD R2, R5, R3`	101	011	0	1	00	100	00	010	1	0	0	0	x	0	x	x	x	xxxx	D4
b	D	`MOVHI R2, 0x85`	010	110	1	0	00	100	xx	xxx	0	0	0	0	0	0	x	x	x	FF0A	85
c	Ldb	`LDB R2, 3(R1)`	001	010	x	x	xx	xxx	01	010	1	0	0	0	x	0	1	x	1	xxxx	83

Ejercicio 2 (1 punto) Criterio: 0,25 puntos por fila, correción binaria

Indicad qué cambios se producen en el estado en el SISC Von Neumann después de ejecutar cada una de las instrucciones de la tabla suponiendo que antes de ejecutarse cada una de ellas `PC=0x6ABC`, `Ri=0x443F` y que el contenido del word de memoria i-ésimo és (i+2) módulo 2^{16}. Utilizad el mnemotécnico $MEM_b[...] = ...$ y/o $MEM_w[...] = ...$ para indicar cambios en la memoria.

Instrucción a ejecutar	Cambios en el estado del computador
`LDB R1, 1(R6)`	`R1 = 0x0042, PC = 0x6ABE`
`MOVHI R2, 0xFA`	`R2 = 0xFA3F, PC = 0x6ABE`
`BNZ R3, -1`	`Ningún cambio (PC pasa a 0x6ABE y después a 0x6ABC)`
`JALR R4, R5`	`R4= 0x6ABE, PC = 0x443E`

Ejercicio 3 (1 punto) Criterio: 0,25 cada fila, 0,1 si hay 1 error, 0 si hay 2 o más errores

Completad las filas y columnas de la siguiente tabla que representa un subconjunto 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	WrD	R@/PC	Pc/Rx	Byte	Ry/N	Estado
0	1	1	0	0	1	0	0	Fetch
8	0	x	1	1	x	1	x	Ldb
10	1	x	1	x	0	x	x	Jalr
14	0	x	1	x	0	x	0	Movhi

Ejercicio 4 (2 puntos)

El programa ensamblador de la derecha se ha traducido a lenguaje máquina para ser ejecutado en el SISC Von Neumann, situando la sección `.data` a partir de la dirección `0xA000` de memoria y justo a continuación la sección `.text`.

```
.data
V1:     .word 1, 2, 4, 8
        .word 16, 32, 64
        .word -1
V2:     .byte 7, 6, 5, 4
        .byte 3, 2, 1
        .even
V3:     .word 0

.text
        MOVI  R0, lo(V1)
        MOVHI R0, hi(V1)
        MOVI  R1, lo(V2)
        MOVHI R1, hi(V2)
        MOVI  R2, 0
        MOVI  R3, 0xFF
L1:     LD    R4, 0(R0)
        CMPEQ R5, R3, R4
        BNZ   R5, L2
        LDB   R6, 0(R1)
        SHL   R4, R4, R6
        ADD   R2, R2, R4
        ADDI  R0, R0, 2
        ADDI  R1, R1, 1
        BZ    R5, L1
L2:     MOVI  R7, lo(V3)
        MOVHI R7, hi(V3)
        ST    0(R7), R2
.end
```

a) Una vez cargado el programa en memoria:

- ¿A qué dirección de memoria corresponden las etiquetas, o direcciones simbólicas, `L1` y `V2`? (0,5 puntos) Criterio: 0,25 cada una, binario

`L1 = 0xA026`	`V2 = 0xA010`

- ¿Cuál es el word almacendo en las siguientes direcciones de memoria? (0,5 puntos) Criterio: 0,25 por apartado, binario

Mem_W`[0xA010] = 0x0607`
Mem_W`[0xA02A] = 0x8B06`

b) Una vez ejecutado el programa en el computador SISC Von Neumann ¿Cuál es la dirección de memoria escrita por la instrucción `ST`?¿Cuál es el valor escrito? (0,75 puntos) Criterio: 0,25 la dirección , 0,5 el contenido, binario

Mem_W`[0xA018] = 0x0380`

c) Indicad el número total de instrucciones que ejecuta el programa, así como cuántas son lentas y cuántas son rápidas (0,25 puntos) Criterio: binario

$N_{total} = 75$	$N_{lentas} = 16$	$N_{rápidas} = 59$

Ejercicio 5 (1 punto) Criterio: 0,25 puntos cada respuesta, criterio binario

a) Indicad cuál es contenido, en formato hexadecimal, de las siguientes direcciones de la ROM_Q+ del computador Von Neumann:

ROM_Q+[0x031] = 0x0C	ROM_Q+[0x0A8] = 0x07

b) Indicad qué dirección(es) de la ROM_Q+ contienen las siguientes transciones. Indicad las direcciones en formato binario, indicando con x los bits que no importen.

De Decode a Cmp = 00001 0001 x	De Decode a Addr =00001 0011 x, 00001 0100 x, 00001 0101 x, 00001 0110 x

Ejercicio 6 (3,5 puntos = 0,25 + 1,5 (0,5 + 1) + 1,75 (0,25 + 0,25 + 1 + 0,25))

Al analizar los programas que se ejecutan en el computador SISC von Neumann, se detecta que es habitual la ejecución de dos instrucciones **consecutivas** (p.e. `ADD R1, R1, R3` y `OUT 27, R1`) en la que:

- la primera es una suma (o una resta) donde el registro destino coincide con el primer registro fuente
- la segunda envía el resultado del cálculo a un puerto de salida

Por lo tanto, el diseñador del lenguaje máquina SISA se plantea ampliar el repertorio de instrucciones con una nueva instrucción que realice ambas tareas: el cálculo (suma o resta) y el envío del resultado a un puerto de salida.

- Sintaxis: `COMB Ra, Rb, N`
- Codificación: `1111 aaa bbb nnnnnn`
- Semántica: `PC=PC+2; if (b > 3) Ra = Ra - Rb; else Ra = Ra + Rb; OutPort[N] = Ra;`

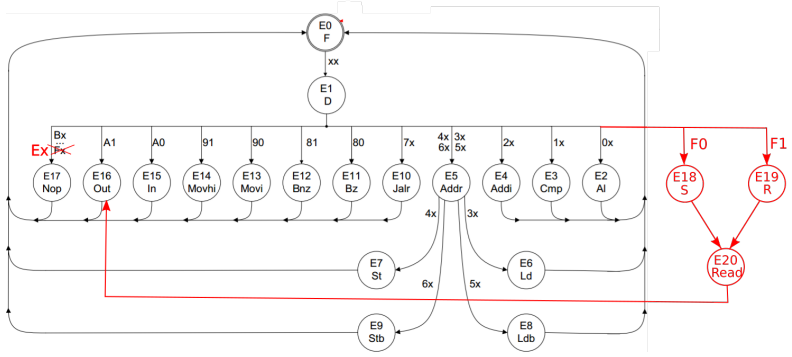
  Tened en cuenta que, según la semántica de la instrucción, **es el identificador del registro b y no su contenido** el que indica si se debe hacer una suma o una resta. Por ejemplo, las instrucciones consecutivas `ADD R5, R5, R1` y `OUT 3, R5` podrán ser substituidas por la instrucción `COMB R5, R1, 3`. Análogamente, `SUB R2, R2, R6` y `OUT 9, R2` podrán ser substituidas por `COMB R2, R6, 9`.

  Asumimos que el sistema de entrada/salida dispondrá de un máximo de 64 puertos de salida. El controlador descartará los 2 bits altos del bus ADDR-IO generado por el control del procesador.

a) Suponiendo que la implementación de la nueva instrucción tarde 5 ciclos (incluyendo Fetch y Decode) y que no impacte en el T_c, consideramos un programa que ejecute 2.400 instrucciones rápidas y 700 lentas: entre las 2.400 rápidas, hay 375 parejas de instrucciones `ADD/OUT` y 125 parejas `SUB/OUT` de las indicadas anteriormente. Reescribimos el programa utilizando la instrucción `COMB`. **Indicad qué porcentaje de reducción en el tiempo de ejecución** observaríamos respecto al tiempo de ejecución del programa original. **(Indicad la expresión del cálculo en función del número y tipo de instrucciones así como el resultado final)** (0,25 puntos) Criterio: binario (si es correcto el resultado o los cálculos indicados)

$NumCiclos_{original} = 3{\cdot}2.400 + 4{\cdot}700 = 10.000$
$NumCiclos_{nuevo} = 3{\cdot}(2.400\text{-}2{\cdot}(375{+}125)) + 4{\cdot}700{+}\ 5{\cdot}(375{+}125) = 3{\cdot}1.400 + 4{\cdot}700 + 5{\cdot}500 = 9.500$
El programa con la nueva instrucción tardará un 5 % menos que el original.

b) **Sin modificar el hardware** y sólo modificando el contenido de las ROM's, completad el diseño del computador para que ejecute, **además** de las instrucciones originales, la instrucción `COMB` en 5 ciclos (incluyendo F y D).
b1) Indicad qué modificaciones introduciríais en el grafo de estados de la unidad de control (0,5 puntos) Criterio: binario



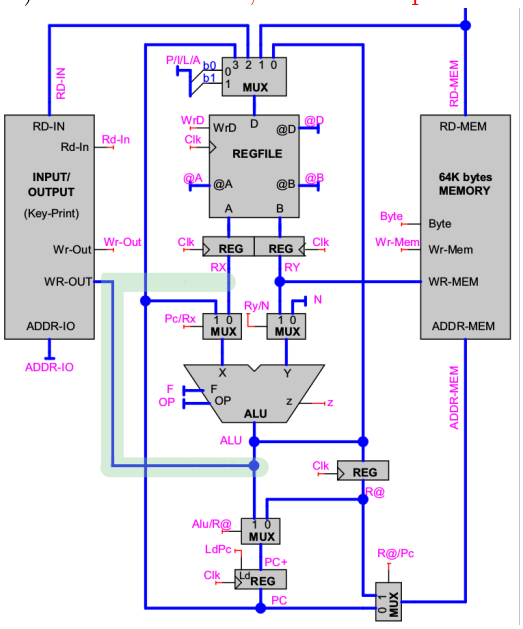
b2) Indicad el contenido de las filas de la ROM_OUT que sea preciso modificar así como las acciones a realizar (la tabla adjunta tiene el número suficiente de filas, incluso es posible que no sean necesarias todas) (1 punto) Criterio: 0,25 las acciones (0,15 si falta una); 0,75 la romout (0,25 por fila, 0,15 con un error, 0 con dos o más errores)

@ROM	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
18	0	0	0	0	0	1	0	x	x	x	0	1	0	0	0	0	x	x	1	1	0	0	1	0
19	0	0	0	0	0	1	0	x	x	x	0	1	0	0	0	0	x	x	1	1	0	1	1	0
20	0	0	0	0	0	0	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

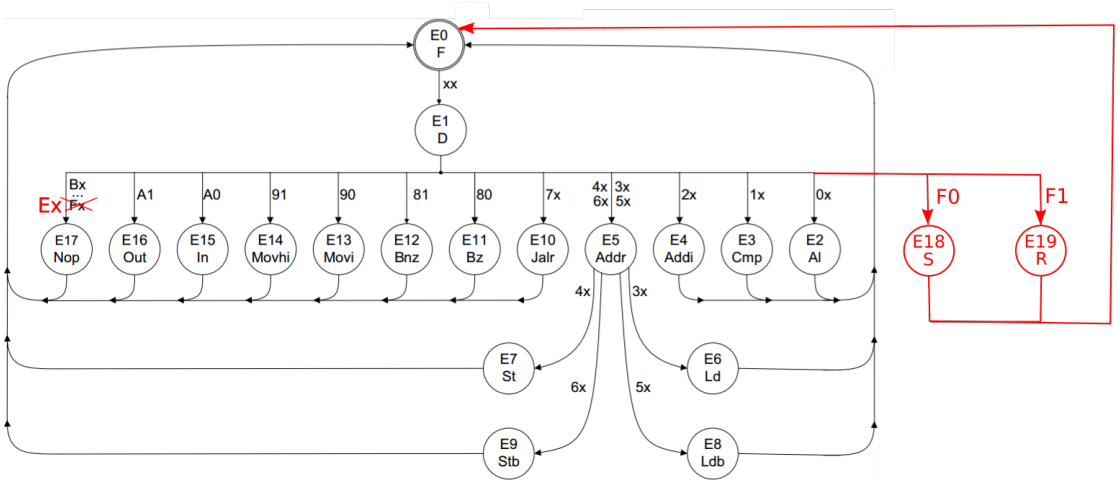
Acciones asociadas al estado (en lenguaje de transferencia de registros)
Ra = RX + RY
Ra = RX - RY
RX = Ra

c) **Si pudiéramos modificar el hardware de la Unidad de Proceso (pero no el de la Unidad de Control),**

c1) Proponed qué modificaciones introduciríais al hardware de la UP para poder reducir el tiempo de ejecución de la instrucción `COMB` a 3 ciclos (incluyendo Fetch y Decode). Sólo es necesario modificar el punto de conexión del bus WR_OUT en la UP. (0,25 puntos) Criterio: binario, también se puede conectar desde entrada del REGFILE



c2) Indicad cómo habría que modificar consecuentemente el grafo de estados de la UC (0,25 puntos) Criterio: binario



c3) Indicad el contenido de las filas de la ROM_OUT que sea preciso modificar así como las acciones a realizar (la tabla adjunta tiene el número suficiente de filas, incluso es posible que no sean necesarias todas) (1 punto) Criterio: como apartado b2)

@ROM	Bnz	Bz	WrMem	RdIn	WrOut	WrD	Ldlr	Byte	R@/Pc	Alu/R@	Pc/Rx	Ry/N	P/I/LA1	P/I/LA0	OP1	OP0	MxN1	MxN0	MxF	F2	F1	F0	Mx@D1	Mx@D0
18	0	0	0	0	1	1	x	x	x	x	0	1	0	0	0	0	x	x	1	1	0	0	1	0
19	0	0	0	0	1	1	x	x	x	x	0	1	0	0	0	0	x	x	1	1	0	1	1	0
16	0	0	0	0	1	0	x	x	x	x	0	x	x	x	1	0	x	x	1	0	0	0	x	x

Acciones asociadas al estado (en lenguaje de transferencia de registros)

Ra = RX+RY || OutP[N]=RX+RY
Ra = RX-RY || OutP[N]=RX-RY
OutP[N] = ALU (RX)

c4) Actualizad el cálculo del apartado a) considerando que la instrucción `COMB` tarda 3 ciclos. (0,25 puntos) Criterio: como apartado a)

$NumCiclos_{original}$ = 3·2.400 + 4·700 = 10.000
$NumCiclos_{nuevo}$ = 3·(2.400-2·(375+125)) + 4·700 + 3·(375+125) = 3·1.400 + 4·700 + 3·500 = 8.500
El programa con la nueva instrucción tardará un 15 % menos que el original.