

- 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)

Cada apartado 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 un bit sea irrelevante en dicho ciclo (aunque se pudiera saber su valor).** Para cada apartado/fila se indica el nodo/estado de la UC y la instrucción SISA almacenada en el IR en ese ciclo. Si os resulta necesario, podéis suponer que el contenido del registro Ri es i.

Apartado	Nodo/Estado (Mnemo Salida)	Instrucción en IR (en ensamblador)	Palabra de Control																	
			@A	@B	Pc/Rx	Ry/N	OP	F	P//L/A	@D	WrD	Wr-Out	Rd-In	Wr-Mem	Ldlr	LdPc	Byte	Alu/R@	R@/Pc	N (hexa)
a	In	IN R7, 65																		
b	D	BNZ R2, 16																		
c	Stb	STB -3(R6), R5																		

Ejercicio 2 (1 punto)

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=0xACDC, Ri=0xABBA y que el contenido del byte de memoria i-ésimo es (i+2) módulo 2⁸. Utilizad el mnemotécnico $MEM_b[...] = \dots$ y/o $MEM_w[...] = \dots$ para indicar cambios en la memoria.

Instrucción a ejecutar	Cambios en el estado del computador
LD R1, 1(R6)	
MOVHI R2, 0x26	
BNZ R3, -7	
JALR R4, R5	

Ejercicio 3 (1 punto)

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	Bz	WrOut	R@/PC	Alu/R@	OP1	OP0	MxN1	MxN0	MxF	Estado
0										Fetch
8										Ldb
10										Jalr
14										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 `.text` a partir de la dirección `0xBE00` de memoria y justo a continuación la sección `.data`.

a) Una vez cargado el programa en memoria:

- ¿A qué dirección de memoria corresponden las etiquetas, o direcciones simbólicas, `L1` y `V`? (0,5 puntos)

<code>L1 = 0x</code>	<code>V = 0x</code>
----------------------	---------------------

- ¿Cuál es la codificación SISA de las instrucciones `MOVHI R0, hi(V)` y `BNZ R4, L1`? (0,5 puntos)

<code>MOVHI R0, hi(V) = 0x</code>
<code>BNZ R4, L1 = 0x</code>

b) Una vez ejecutado el programa, ¿cuál es el contenido del vector `V`? (indicad la lista de valores como *words* separados por comas y en el orden de almacenamiento) (0,75 puntos)

--

c) Indicad el número de instrucciones que ejecuta cada mitad del programa, desglosadas en lentas y rápidas, (0,25 puntos)

1ª mitad	$N_{lentas_m1} =$	$N_{rápidas_m1} =$
2ª mitad	$N_{lentas_m2} =$	$N_{rápidas_m2} =$

```

N = 8 ; Asumimos N par

.data
V:      .space 2*2*N

.text
; primera mitad
MOVI    R0, lo(V)
MOVHI   R0, hi(V)
MOVI    R1, lo(N)
MOVHI   R1, hi(N)
MOVI    R2, 0
MOVI    R3, 0
L1:     ADD    R2, R2, R3
        ST     0(R0), R2
        ADDI   R0, R0, 2
        ADDI   R3, R3, 1
        CMPLTU R4, R3, R1
        BNZ    R4, L1

; segunda mitad
MOVI    R2, 0
MOVI    R3, 0
L2:     ADD    R2, R2, R3
        ST     0(R0), R2
        ADDI   R3, R3, 1
        ADD    R2, R2, R3
        ST     2(R0), R2
        ADDI   R0, R0, 4
        ADDI   R3, R3, 1
        CMPLTU R4, R3, R1
        BNZ    R4, L2

.end

```

Ejercicio 5 (1 punto)

a) Indicad cuál es contenido de las siguientes direcciones de la `ROM_Q+` del computador Von Neumann:

<code>ROM_Q+[0x010] = 0x</code>	<code>ROM_Q+[0x14F] = 0x</code>
---------------------------------	---------------------------------

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

De <code>Addr</code> a <code>Stb</code> =	De <code>Decode</code> a <code>Movi</code> =
---	--

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

El diseñador del lenguaje máquina SISA considera que es preciso añadir una nueva instrucción al repertorio de instrucciones. Se trata de una instrucción de salto indexado (*indexed jump*, JI) que permite saltar a la dirección de código indicada por la posición n -ésima de un vector almacenado en memoria, donde n es un natural menor que 128. Como la instrucción JALR, esta nueva instrucción también devuelve la dirección de código correspondiente al secuenciamiento implícito.

- Sintaxis: JI Ra, N7
- Codificación: 1111 aaa x 0nnn nnnn
- Semántica: $\text{tmp} = \text{PC} + 2$; $\text{PC} = \text{Mem}_w[\text{Ra} + 2 \cdot \text{N7}]$; $\text{Ra} = \text{tmp}$;

La codificación de la instrucción exige que el bit 7 valga '0'; en cambio, el valor del bit 8 es indiferente.

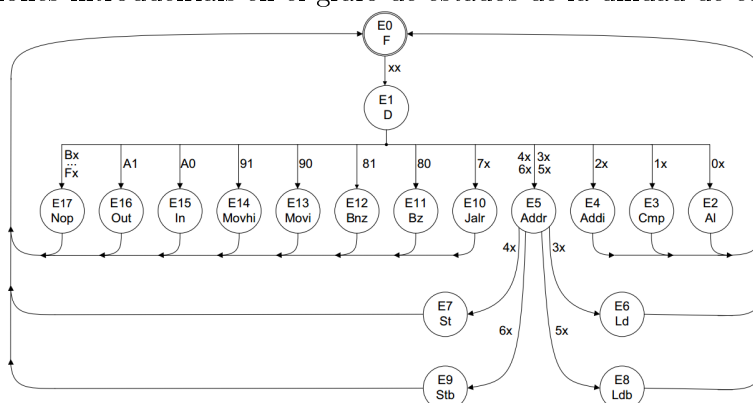
Tened en cuenta que Ra es tanto registro fuente como registro destino.

Por ejemplo, si $\text{R3}=0\text{xBECA}$, $\text{PC}=0\text{xAD10}$ y el vector de *words* almacenado a partir de la dirección 0xBECA contiene los *words* 0xABBA , 0xACDC , 0xCAFE , 0xBEBE , 0xBABE , 0xFACE , ... en este orden, la ejecución de JI R3, 0 provocaría que $\text{PC}=0\text{xABBA}$ y $\text{R3}=0\text{xAD12}$; en cambio, la ejecución de JI R3, 5 provocaría que $\text{PC}=0\text{xFACE}$ y $\text{R3}=0\text{xAD12}$.

- a) Consideramos un programa que ejecuta 2.000 instrucciones rápidas y 1.000 de acceso a memoria. Reescribimos el programa utilizando la instrucción JI y obtenemos una nueva versión del programa que ejecuta 1.600 instrucciones rápidas, 900 instrucciones de acceso a memoria y 200 instrucciones JI. Suponiendo que la implementación de la nueva instrucción tarde 6 ciclos (incluyendo Fetch y Decode) y que no impacte en el T_c , **indicad cuántos ciclos tarda la ejecución de cada versión del programa y qué porcentaje de reducción en el tiempo de ejecución observaríamos respecto al tiempo de ejecución del programa original. (Indicad el cálculo en función del número y tipo de instrucciones así como el resultado final)** (0,25 puntos)

- 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 JI en 6 ciclos (incluyendo F y D).

- b1) Indicad qué modificaciones introduciríais en el grafo de estados de la unidad de control (0,5 puntos)

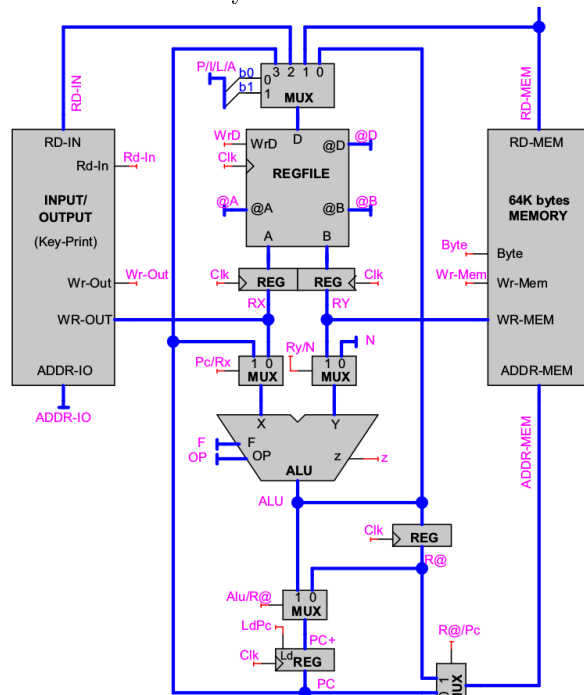


- b2) Indicad el contenido de las filas de la ROM_OUT que sea preciso modificar así como las acciones a realizar (la tabla adjunta tiene el número suficiente de filas, incluso es posible que no sean necesarias todas) (1 punto)

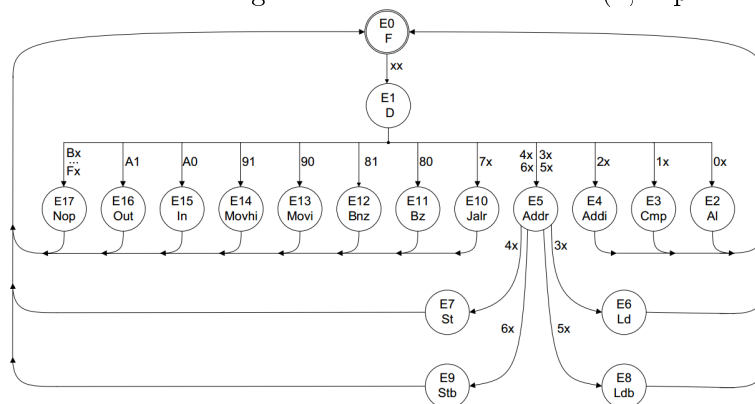
@ROM	Bnz	Bz	WrMem	RdIn	WrOut	WrD	Ldlr	Byte	R@/Pc	Alu/R@	Pc/Rx	Ry/N	P//L/A1	P//L/A0	OP1	OP0	MxN1	MxN0	MxF	F2	F1	F0	Mx@D1	Mx@D0

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

c) Si pudiéramos modificar el hardware de la Unidad de Proceso y el de la Unidad de Control),
c1) ¿Cómo modificaríais el hardware de la UPG para reducir el tiempo de ejecución de la instrucción JI a 4 ciclos (incluyendo Fetch y Decode)? Podéis modificar buses y añadir multiplexor(es), buses y señales de control pero no podéis modificar ni los bloques ni el uso de las señales de control existentes. Para poder mantener el tiempo de ciclo, no es válido calcular la dirección de memoria y acceder a memoria en el mismo ciclo. (0,25 puntos)



c2) ¿Cómo modificaríais consecuentemente el grafo de estados de la UC? (0,25 puntos)



c3) Indicad el contenido total de las filas de la ROM_OUT que sea preciso añadir así como las acciones a realizar; disponéis de espacio para indicar dos nuevas señales de control (es posible que no sean necesarias todas). Indicad también el valor de la(s) nueva(s) señal(es) en al menos dos de las filas de la ROM_OUT ya existentes en las que la(s) nueva(s) señal(es) de control no tenga(n) el valor x. La tabla adjunta tiene el número suficiente de filas, incluso es posible que no sean necesarias todas. (1 punto)

@ROM	New1	New2	Bnz	Bz	WrMem	RdIn	WrOut	WrD	Ldlr	Byte	R@/Pc	Alu/R@	Pc/Rx	Ry/N	P//L/A1	P//L/A0	OP1	OP0	MxN1	MxN0	MxF	F2	F1	F0	Mx@D1	Mx@D0

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

c4) Actualizad el cálculo del apartado a) considerando que la instrucción JI tarda 4 ciclos. (0,25 puntos)