Examen 3. (Temas 8, 9, 10 y 11)

- Duración del examen: 1 hora y 45 minutos.
- La solución se tiene que escribir en el espacio reservado para ello en el propio enunciado.
- No podéis utilizar calculadora, móvil, apuntes, etc.
- La solución del examen se publicará en Atenea mañana y las notas antes del 7 de Mayo del 2019

Ejercicio 1 *(0,5 puntos)*
Completa el siguiente fragmento de código ensamblador SISA para el procesador SISC Harvard uniciclo (UPG+I/O+MEM) para que inicialice el R0 con el valor de memoria de la posición 0x0101.

```
MOVI R0, 0x____
MOVI R1, 0x____
SHA  R2, _______
ADD  R2, _______
LD    R0, 0(___)
```

Ejercicio 2 *(1 punto)*

1. Indica el valor que debe tener cada uno de los bits de la palabra de control de la UPG básica (sin subsistema de I/O ni memoria) para que realice, durante un ciclo, la acción concreta especificada mediante el mnemotécnico. Indicad con ***x*** las casillas cuyo valor no importe para la ejecución de la instrucción. En caso de que no se pueda realizar la acción tachar **toda la línea** de señales. *(0.5 puntos)*

Mnemotécnico	@A	@B	Rb/N	OP	F	In/Alu	@D	WrD	N (hexa)
SHAI R4, R5, 4									
IN R3//ADD -,R2,R1									

2. Indica el mnemotécnico que corresponde a cada una de las siguientes palabras de control de la UPG básica (sin subsistema de I/O ni memoria). *(0.5 puntos)*

Mnemotécnico	@A	@B	Rb/N	OP	F	In/Alu	@D	WrD	N (hexa)
	xxx	xxx	0	10	001	0	011	1	0001
	000	000	1	01	011	x	xxx	0	XXX

Ejercicio 3 *(1 punto)*
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.

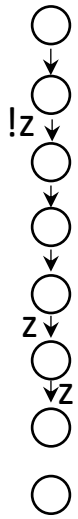
Lenguaje máquina SISA	Lenguaje ensamblador SISA
0x8AFE	
0x087E	
	OR R3, R2, R1
	IN R3, 0x0F

Ejercicio 4 *(3 puntos)*

Dado el siguiente fragmento de código en C (el código no tiene que hacer algo útil) donde todos los datos son **naturales**, Completad el fragmento de…

```
…
while (R0<R5) {
     R5 = R5/2
     R0 = R0 + 1
     if ((R0 == R3) OR (R0 > 99)){
         R0 = R3 – R4
     }
}
…
```

a) … grafo de estados de la UC de **propósito especifico** para que junto con la UPG formen un procesador que realice la funcionalidad descrita en los fragmentos de código anteriores. Indicad los arcos que faltan, las etiquetas de los arcos (z, !z, o nada) y completad las casillas de cada palabra de control que se especifica con mnemotécnicos a la derecha de cada nodo del grafo. *(1.5 punto)*



…
CMP
SH
CMPEQ
CMP
…

b) … programa en lenguaje ensamblador SISA para que el procesador formado por la unidad de control de propósito general (UCG) junto con la UPG realicen las funcionalidades descritas en los fragmentos de código en C (el código no tiene que hacer algo útil). En las comparaciones, hay que interpretar los datos como valores **naturales**. Rellenad la parte subrayada que falta. *(1.5 punto)*

@I-MEM	
	…
0x1000	CMP_____ R7,_____
0x1002	BNZ R7,_____
0x1004	MOV____ R7,_____
0x1006	SH _____
0x1008	ADD _____
0x100A	CMP_____ R7,_____
0x100C	B____ R7,_____
0x100E	MOV____ R6,_____
0x1010	CMP_____ R7, R0,_____
0x1012	B____ R7,_____
0x1014	SUB _____
0x1016	B____ R7,_____
0x1018	BNZ R7, _____
0x101A	…

Ejercicio 5 *(1 punto)*

Escribid sobre la siguiente tabla el valor de los bits que tiene la palabra de control del SISC-Harvard uniciclo (incluyendo la señal *TknBr*) durante el ciclo en que se ejecuta cada una de las instrucciones SISA. Indicad únicamente el valor (0 o 1) de los bits que son estrictamente necesarios para ejecutar correctamente cada instrucción. Para el resto de bits de la palabra de control, que pueden valer 0 o 1 indistintamente para la ejecución correcta de la instrucción, poned **x** (aunque se pueda saber el valor codificando la instrucción). Suponed que antes de ejecutar cada instrucción el contenido de los registros es cero.

Instrucción SISA	@A	@B	Rb/N	OP	F	-/i/I/a	@D	WrD	Wr-Out	Rd-In	Wr-Mem	Byte	TknBr	N (hexa)	ADDR-IO (hexa)
STB 0(R0), R0															
OR R2, R3, R4															
BNZ R7, 0xA															
IN R2, 0x11															

Ejercicio 6 *(1 puntos)*

Indica el contenido de la tabla de la ROM (solo filas indicadas) correspondiente al bloque ROM_CRTL_ LOGIC. Indica los valores que tomarían las señales para ejecutar correctamente las instrucciones. Indica con x los valores de los bits del contenido de la ROM que puedan valer 0 o 1.

Dirección ROM		Contenido ROM														
I_{15} I_{14} I_{13} I_{12}	I_8	Bnz	Bz	Wr-Mem	Rd-In	Wr-Out	Byte	Rb/n	-i/I/a_1 -i/I/a_0	OP_1 OP_0	MxN_1 MxN_0	MxF	F_2 F_1 F_0	MxD_1 MxD_0	
1 0 0 0	0														BZ
1 0 0 1	0														MOVI
0 1 0 0	x														ST
1 0 1 0	1														OUT

Ejercicio 7 *(1 punto)*

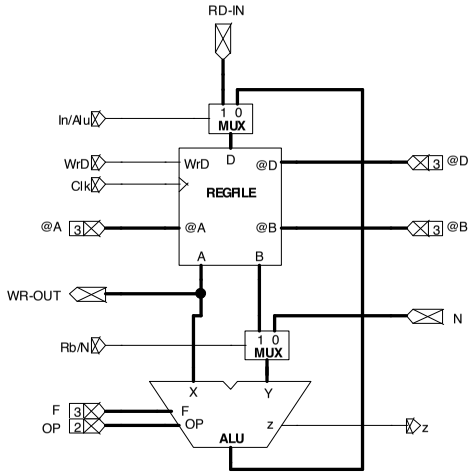
Indicad qué cambios hay en el estado del computador después de ejecutar cada una de las instrucciones de la tabla suponiendo que **antes de ejecutarse cada una** de ellas el PC vale 0x200A, el contenido de todos los registros necesarios es R0=0xF000, R3=0x000A, R5=0x0000, y que el contenido de todas las posiciones pares de la memoria de datos es 0x2 y el de todas las posiciones impares de la memoria de datos es 0x1. Asumid todos los registros de E/S contienen el valor 0x0001 y que el teclado está formado por un port de datos (Keydata) y uno de status (KeyStat). Utiliza el mnemotécnico MEMb[...], MEMw[...] y Port[...] para indicar los cambios en la memoria y los puertos de E/S respectivamente.

Instrucción	Cambios en el estado del computador
XOR R3, R3, R3	
BZ R5, -3	
IN R3, KeyData	
LD R5, 0xA(R0)	

Ejercicio 8 *(1,5 puntos)*

Dado el computador SISC Harvard uniciclo (formado por UCG+UPG+IOkey-print+MEM), escribid un fragmento de código ensamblador SISA que lea dos datos de teclado (ports KeyData y KeyStatus). El primero indica una **dirección de memoria que guardaremos en R0** (seguro que siempre es par) y el segundo un **valor entero de 16 bits que guardaremos en R1**. Después miraremos si el valor (16 bits) que hay en la posición de memoria que indicada por el primer valor es menor que el valor que hemos leído en segundo lugar. Si el valor de memoria es menor, lo reemplazaremos por el valor leído por teclado, en caso contrario no haremos nada y volveremos a leer otros dos números. Para cualquier registro temporal emplearemos el R7. El código no puede tener más de 11 instrucciones. Si tiene más, se tendrá un 0.

UPG básica



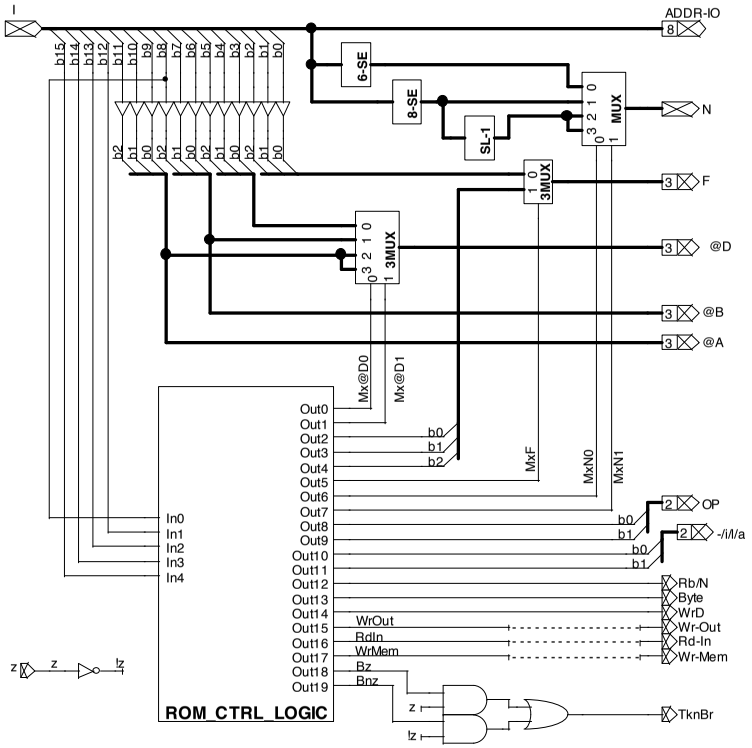
Formato Instrucciones SISA-I

15 14 13 12	11 10 9	8 7 6	5 4 3	2 1 0	Name	Mnemonic
0 0 0 0	a a a	b b b	d d d	f f f	Logic and Aritmetic Operations	AND, OR, XOR, NOT, ADD, SUB, SHA, SHL
0 0 0 1	a a a	b b b	d d d	f f f	Compare Signed and Unsigned	CMPLT, CMPLE, -,CMPEQ, CMPLTU, CMPLEU, -, -
0 0 1 0	a a a	d d d	n n n n n n		Add Immediate	ADDI
0 0 1 1	a a a	d d d	n n n n n n		Load	LD
0 1 0 0	a a a	b b b	n n n n n n		Store	ST
0 1 0 1	a a a	d d d	n n n n n n		Load Byte	LDB
0 1 1 0	a a a	b b b	n n n n n n		Store Byte	STB
0 1 1 1						Branch future extension
1 0 0 0	a a a	0	n n n n n n		Branch on Zero	BZ
		1			Branch on Not Zero	BNZ
1 0 0 1	d d d	0	n n n n n n		Move Immediate	MOVI
	a a a d d d	1			Move Immediate High	MOVHI
1 0 1 0	d d d	0	n n n n n n		Input	IN
	a a a	1			Output	OUT
1 0 1 1						Future extensions
1 1 x x						

Funcionalidades ALU

F			OP			
b_2	b_1	b_0	11	10	01	00
0	0	0	---	X	CMPLT (X,Y)	AND(X,Y)
0	0	1	---	Y	CMPLE (X,Y)	OR(X,Y)
0	1	0	---	MOVHI (X,Y)	---	XOR(X,Y)
0	1	1	---	---	CMPEQ (X,Y)	NOT(X)
1	0	0	---	---	CMPLTU (X,Y)	ADD(X,Y)
1	0	1	---	---	CMPLEU (X,Y)	SUB(X,Y)
1	1	0	---	---	---	SHA(X,Y)
1	1	1	---	---	---	SHL(X,Y)

Lógica de control del SISC Harvard Uniciclo



Computador SISC Harvard Uniciclo (UCG + UPG + IO + MEM)