Examen E3 *(21 de noviembre del 2019)* IC – 19-20-Q1
Apellidos y nombre: …… Grupo:……

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 se publicará en Atenea mañana y las notas antes del 28 de noviembre 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 guarde un 0 en la posición 0x0007 de memoria. ***(Corrección: 0,5 binario)***

```
XOR  R0, R0, R0  ; R0 = 0
ADDI R1, R0, 0x1
SHA  R2, R1, R1  ; R2 = 10
ADD  R2, R2, R1  ; R2 = 11
SHA  R2, R2, R1  ; R2 = 110
ADD  R2, R2, R1  ; R2 = 111
ST   0(R2), R0   ; MEM[0+7] = 0
```

Ejercicio 2 *(1 punto)*

a) 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)* ***(Corrección: 0,25 por fila correcta)***

Mnemotécnico	@A	@B	Rb/N	OP	F	In/Alu	@D	WrD	N (hexa)
XORI R4, R5, 4	101	xxx	0	00	010	0	100	1	0x0004
IN R3//AND -,R3,R3	011	011	1	00	000	1	011	1	XXXX

b) 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)* ***(Corrección: 0,25 por fila correcta)***

Mnemotécnico	@A	@B	Rb/N	OP	F	In/Alu	@D	WrD	N (hexa)
ADD R1, R1, R2	001	010	1	00	100	0	001	1	XXXX
MOVEI R1, 0xFF43	xxx	xxx	0	10	001	0	001	1	FF43

Ejercicio 3 *(1 punto)*
Completa la 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. ***(Corrección: 0,25 por fila correcta)***

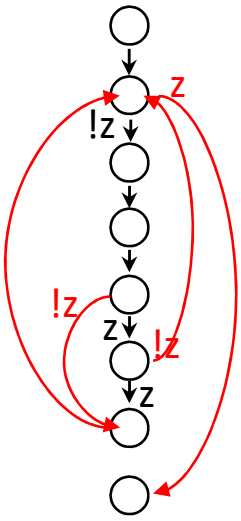
Lenguaje máquina SISA	Lenguaje ensamblador SISA
0x96AB	MOVI R3,0xAB
0x0772	XOR R6, R3, R5
0x025F	SHL R3, R1, R1
0xAB0F	OUT 0x0F, R5

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 = R5/4
   R0 = R0 XOR R4
   if ((R0 == R3) OR (R0 > 0x1F4)){
            R0 = 0xFF
   }
}
```

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)* ***(Corrección: -0,25 por cada nodo con error)***



…
CMPLTU -,R0, R5
SHLI R5, R5, -2
XOR R0, R0, R4
CMPEQ -, R0, R3
CMPLEUI -,R0,0x1F4
MOVEI R0, 0xFF
…

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)* ***(Corrección: -0,25 por cada nodo con error. Excepción: si el error viene directamente de apartado a no baja)***

@I-MEM	
	…
0x1000	CMPLEU R7, R5, R0
0x1002	BNZ R7, 12
0x1004	MOVI R7, -2
0x1006	SHL R5, R5, R7
0x1008	XOR R0, R0, R4
0x100A	CMPEQ R7, R0, R3
0x100C	BNZ R7, 4
0x100E	MOVI R6, 0xF4
0x1010	MOVHI R6, 0x01
0x1012	CMPLTU R7, R6, R0
0x1014	BZ R7, -11
0x1016	MOVI R0, 0xFF
0x1018	MOVHI R0, 0x00
0x101A	BNZ R7, -14
0x101C	…

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. ***(Corrección: -0,25 por cada fila o columna con error contando el menor número de filas/columnas que cubran todos los errores)***

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)
MOVHI R0, 0xFF	000	xxx	0	10	010	00	000	1	0	0	0	x	0	FFFF	XX
XOR R1, R1, R1	001	001	1	00	010	00	001	1	0	0	0	x	0	XXXX	XX
BZ R7, 0xA	111	xxx	x	10	000	xx	xxx	0	0	0	0	x	1	0014	XX
OUT 0x11, R2	010	xxx	x	xx	xxx	xx	xxx	0	1	0	0	x	0	XXXX	11

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. ***(Corrección: -0,25 por cada fila o columna con error contando el menor número de filas/columnas que cubran todos los errores)***

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_{a1} -/i/I_{a0}	OP_1 OP_0	MxN_1 MxN_0	MxF	F_2 F_1 F_0	MxD_1 MxD_0	
1 0 1 0	0	0	0	0	1	0	x	x	10	xx	xx	x	xxx	10	IN
1 0 0 1	0	0	0	0	0	0	x	0	00	10	01	1	001	10	MOVI
0 1 0 1	x	0	0	0	0	0	1	0	01	00	00	1	100	01	LDB
1 0 0 0	1	1	0	0	0	0	x	x	xx	10	10	1	000	xx	BNZ

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 0x1000, el contenido de todos los registros necesarios es R0=0x00FF, 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) y que el sistema de ES tiene el efecto lateral del protocolo de cuatro fases. Utiliza el mnemotécnico MEMb[...], MEMw[...] y Port[...] para indicar los cambios en la memoria y los puertos de E/S respectivamente. ***(Corrección: 0,25 por cada casilla correcta)***

Instrucción	Cambios en el estado del computador
MOVHI R0, 0x1	R0 = 0x01FF PC = 0x1002
BZ R5, 5	PC = 0x100C
IN R3, KeyData	R3 = 0x0001, Port[KeyStat] = 0 PC = 0x1002
ST 0xA(R5), R0	MEMw[0x000A] = 0x00FF PC = 0x1002

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 un dato de teclado (ports KeyData y KeyStatus). **Guarde el valor leído en el registro R0. En el Registro R1 guarde los 4 bits de menos peso extendiendo el signo a los 16 bits del contenido del R0.** Después guardaremos los 8 bits de menos peso de R1 en la posición de memoria almacenada en R5. Para cualquier registro temporal emplearemos el R7. No podéis emplear ninguna operación de comparación (CMP*). El Código no debe tener más de 11 líneas.
(Corrección:
0,5 por E/S correcta
0,25 por store correcto
0,25 por obtener los 4 bits de menos peso
0,5 por extender bien el signo)

Opción 1:

```
IN R0, KeyStatus
BZ R0, -2
IN R0, KeyData          ; R0 tiene el valor leído por teclado
MOVI R7,0x0F            ; R7 = 0000 0000 0000 1111
AND R1, R0, R7          ; R1 tiene los 4 bits de menos peso de R0
MOVI R7,0x08            ; R7 = 0000 0000 0000 1000
AND R7, R1, R7          ; Si R7 vale 0 es que el bit de mas peso de los 4 bit que quedan es 0
                        ; Si vale 1 habrá que extender el signo
BZ R7, 2
MOVI R7, 0xF0           ; R7 = 1111 1111 1111 0000
OR R1, R7, R1           ; R1 = 1111 1111 1111 4 bits de R1
STB 0(R5), R1           ; Guardamos los 8 bits de menos peso de R1 en R5
```

Opción 2:

```
IN R0, KEY_STATUS
BZ R0, -2
IN R0, KEY_DATA
MOVI R7,0x0F
AND R1, R0, R7
MOVI R7,12
SHA/L R1,R1,R7
MOVI R7, -12
SHA R1,R1,R7
STB 0(R5),R1
```

Opción 3: Una optimización sobre la opción 2

```
IN R0, KEY_STATUS
BZ R0, -2
IN R0, KEY_DATA
MOVI R7,12
SHA/L R1,R0,R7
MOVI R7, -12
SHA R1,R1,R7
STB 0(R5),R1
```