Curs de Reavaluació d'IC – Sessió 5

Activitat 1

Estudia el document corresponent al tema 14 de teoria que hi ha Atenea:

- "Llenguatge màquina i Assemblador SISA"

Estudia el document corresponent al tema 10 de teoria que hi ha Atenea:

- "Documentación Tema 10: Unidad de Control General"

Estudia el document corresponent al tema 11 de teoria que hi ha Atenea:

- "Teoria: Memoria"

Estudia el document corresponent al tema 12 de teoria que hi ha Atenea:

- "Teoria: Los computadores SISC Harvard uniciclo y multiciclo"

Estudiar vol dir llegir els documents curosament i apuntar els dubtes que vagin sorgint per preguntar-los a la classe.

Activitat 2

Realitza els següents problemes i entrega les solucions en un document PDF a l'espai reservat a Atenea.

Problema 1

Raona els avantatges i inconvenients de disposar d'una unitat de control amb seqüenciament implícit davant d'una unitat de control amb seqüenciament explícit. Indica quines són les afirmacions correctes

a) Les unitats de control amb seqüenciament implícit no poden saltar a la mateixa instrucció que estan executant com si poden fer-ho les unitats de control amb seqüenciament explícit.

b) Els programes/grafs amb seqüenciament implícit tenen menys instruccions/estats que amb seqüenciament explícit. **Certa**

c) Per fer programes/grafs amb seqüenciament implícit necessitem noves instruccions especials.

d) En general, els programes/grafs amb seqüenciament implícit ocupen menys espai a la memòria ROM. **Certa**

e) Les unitats de control amb seqüenciament explícit necessiten disposar d'un registre que contingui l'adreça de la ROM on està ubicada la instrucció que s'està executant. **Certa**

f) Les unitats de control amb seqüenciament explícit executen les instruccions en l'ordre que estan escrites a memòria.

g) Les unitats de control amb seqüenciament implícit incrementen automàticament el registre PC amb el nombre de posicions de memoria que ocupa una instrucció quan executen una instrucció que no trenqui la seqüència. **Certa**

h) Les unitats de control amb seqüenciament implícit son capaces de fer més tasques que les unitats amb seqüenciament explícit.

i) Les unitats de control amb seqüenciament explícit son capaces de fer més tasques que les unitats amb seqüenciament implícit.

j) Les unitats de control amb seqüenciament implícit necessiten d'un bloc que incrementi el valor PC. **Certa**

Problema 2

Completa la siguiente tabla desensamblando las instrucciones en lenguaje máquina. Indica poniendo NA en la casilla aquellos casos en los que la instrucción no corresponda al lenguaje SISA.

Lenguaje máquina SISA	Lenguaje ensamblador SISA
0x677D	STB -3(R3), R5
0xA0FC	IN R0, 292
0x21C6	ADDI R7, R0, 6
0x0000	AND R0, R0, R0
0x145C	CMPLTU R3, R2, R1

0x7800	JARL R0,R4
0x01CA	XOR R1, R0, R7
0x9780	MOVHI R2, -128
0x5DAB	LDB R2,-21(R0)
0x4722	ST -30(R3), R4
0x022E	SHA R5,R1,R0
0x3877	LD R1, -9(R4)

Problema 3

Completa la siguiente tabla ensamblando las instrucciones en ensamblador SISA. Indica poniendo NA en la casilla aquellos casos en los que la instrucción no corresponda al lenguaje SISA.

Lenguaje ensamblador SISA	Lenguaje máquina SISA
MOVHI R4, 24	0x9918
BNZ R2, -5	0x85FB
STB 3(R5), R4	0x6B03
ST -2(R6), R1	0x4C7E
SUB R2, R5, R3	0x0A05
LDB R5, 3(R2)	0x5543
OR R1, R7, R0	0x0E09
BZ R5, -3	0x8AFD
CMPEQ R8, R5, R7	NA
IN R2, 8	0xA408

Problema 4

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 0x8C36, el contenido de todos los registros es 0xE328, el contenido de todas las posiciones pares de la memoria de datos es 0x47 y el de todas las posiciones impares de la memoria de datos es 0xA3, y que el contenido de todos los puertos de entrada es 1. Utiliza el mnemotécnico MEM_b[…], MEM_w[…] y DataOut[…] para indicar los cambios en la memoria y los puertos de E/S respectivamente.

Instrucción a ejecutar	Canvios en el estado del computador
SHL R2, R1, R5	PC = 0x8C38 // R2 0 0x0000
ST 6(R6), R3	PC = 0x8C38 // MEMw[0xE32E]=0XE328
OUT 11, R3	PC = 0x8C38 // DataOut[0Xe328]
MOVI R5,0x66	PC = 0x8C38 // R5 = 0x0066
STB 3(R5), R4	PC = 0x8C38 // MEMw[0XE32B]=0x2B

BZ R0, -6	PC = 0x8C38
MOVHI R6, 94	PC = 0x8C38 //R6 = 0x5E28
CMPLE R3, R1, R4	PC = 0x8C38 // R3=0x0001
LDB R2,4(R6)	PC = 0x8C38 // R2=0x0047
SHA R3, R3, R3	PC = 0x8C38 // R3=0xFFFF
SUB R6, R4, R1	
BNZ R5, 6	
LDB R4,8(R5)	
LD R3,2(R2)	
ST 4(R0), R3	

Problema 5

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, de los puertos de entrada/salida y de la memoria de datos es cero.

Instrucción SISA	Palabra de Control del SISC Harvard uniciclo														
	@A	@B	Rb/N	OP	F	-/i/l/a	@D	WrD	Wr-Out	Rd-In	Wr-Mem	Byte	TknBr	N (hexa)	ADDR-IO (hexa)
ST 0(R4), R0	100	000	0	00	100	XX	XXX	0	0	0	1	0	0	0000	00
BZ R1, -6	001	XXX	0	10	001	XX	XXX	0	0	0	0	0	1	FFFA	FA
ADDI R1, R2, 5	010	XXX	0	00	100	00	001	1	0	0	0	0	0	0005	05
OUT 44, R1	001	XXX	0	XX	XXX	XX	XXX	0	1	0	0	0	0	XXXX	44
STB 3(R2), R5	010	101	0	00	100	XX	XXX	0	0	0	1	1	0	XXXX	43
BNZ R7, 11	111	XXX	0	10	001	XX	XXX	0	0	0	0	0	1	0003	0B
MOVHI R1, -2	XXX	XXX	0	10	001	00	001	1	0	0	0	0	0	FFFE	FE
LDB R6, 0(R4)	100	XXX	0	00	100	01	110	1	0	0	0	1	0	0000	80

Problema 6

Indica el contenido de la tabla de la ROM (sólo las celdas en blanco) 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 de la ROM																			
I_{15} I_{14} I_{13} I_{12}	I_8	Bnz	Bz	Wr-Mem	Rd-In	Wr-Out	WrD	Byte	Rb/N	/i/l/a$_1$ - /i/l/a$_0$	OP$_1$	OP$_0$	MxN$_1$	MxN$_0$	MxF	f2	f1	f0	MxD$_1$	MxD$_0$	
0 0 0 0	X						1	X		00					0						A / L
0 0 0 1	X	0	0	0	0	0	1	X	1	00	01		XX		0	XXX			00		CMP
0 0 1 0	X						1	X		00					1						ADDI
0 0 1 1	X						1	0		01					1						LD
0 1 0 0	X						0	0		01					1						ST
0 1 0 1	X	0	0	0	0	0	1	1	0	01	00		00		1	100			01		LDB
0 1 1 0	X						0	1		01					1						STB

0 1 1 1	X						X	X		XX			X			(NOP)
1 0 0 0	0						0	X		XX			1			BZ
1 0 0 0	1	1	0	0	0	0	0	X	0	XX	00	01	1	100	XX	BNZ
1 0 0 1	0						1	X		00			1			MOVI
1 0 0 1	1	0	0	0	0	0	1	X	0	00	10	01	1	001	XX	MOVHI
1 0 1 0	0						1	X		XX			X			IN
1 0 1 0	1						0	X		XX			X			OUT
1 0 1 1	X						X	X		XX			X			(NOP)
1 1 X X	X						X	X		XX			X			(NOP)

Problema 7

Completa el siguiente fragmento de código ensamblador SISA para el procesador SISC Harvard uniciclo (UPG+I/O+MEM) para que guarde en el registro R0 el byte almacenado en la posición de memoria 0x3456.

```
...
MOVI  R1, 0x56
MOVHI R1, 0x34

LDB R0,0(R1)
```

En el procesador SISC Harvard uniciclo se puede acceder a una misma posición de memoria de varias maneras. ¿De cuantas maneras distintas hubiésemos podido completar el código anterior (teniendo en cuenta que todos los valores numéricos siempre están escritos en hexadecimal) para que realizase el acceso al byte de la posición 0x3456? Indicad el número total y justificadlo.

Problema 8

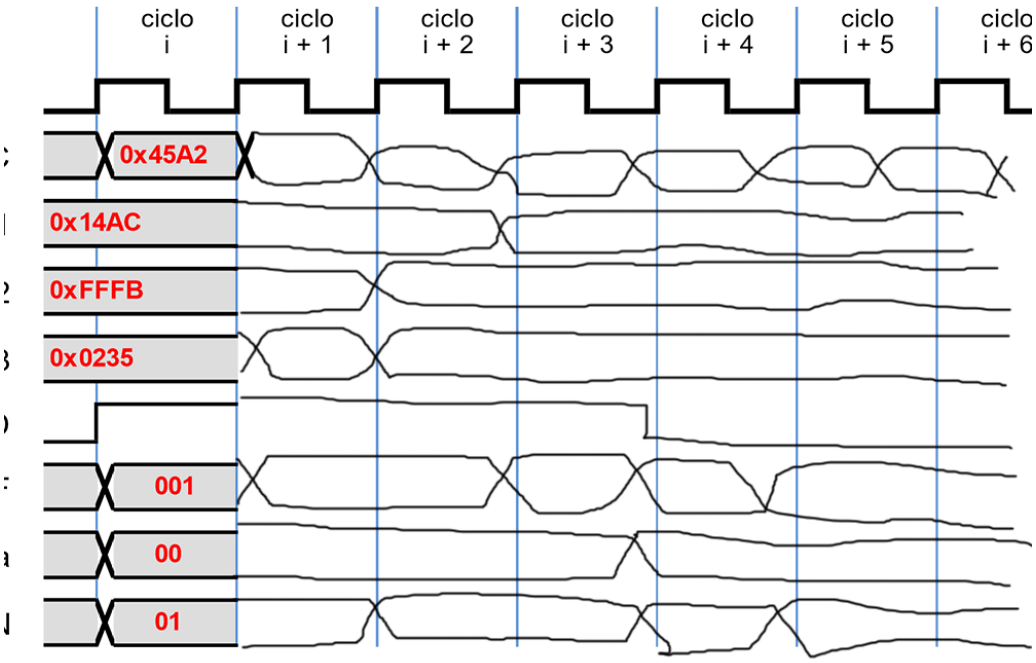
Se ha encargado fabricar una remesa de procesadores SISC Harvard uniciclo (UPG+I/O+MEM) y ha salido defectuosa. El multiplexor controlado por la señal F del bloque MISC que está en la ALU no funciona correctamente para las entradas 1 y 2. Ello imposibilita se puedan ejecutar correctamente las instrucciones MOVI y MOVHI. Deseamos ejecutar este código en estos procesadores:

```
...
MOVI  R0, 0xCD
MOVHI R0, 0xAB
SUB   R2, R0, R1 ...
XOR R0, R0, R0
ADDI R7, R0, 4
ADDI R0, R0, 0x0A
ADDI R0, R0, 0x0B
SHL R0, R0, R7
ADDI R0, R0, 0x0C
SHL R0, R0, R7
ADDI R0, R0, 0x00
SUB R2, R0, R1
```

Escribid un fragmento de código ensamblador SISA equivalente que ejecute la misma tarea, pero sin usar las instrucciones MOVI y MOVHI que no funcionan correctamente. Podéis usar el resto de registros para almacenar valores temporales si los necesitáis.

Problema 9

Dado el siguiente fragmento de código donde en el ciclo i se ejecuta la instrucción MOVI R3,0xFE en el procesador SISC Harvard uniciclo, rellenad el siguiente cronograma indicando el valor de las señales de la UCG o UPG y los valores de los registros. (nota: para facilitar la lectura del cronograma no hace falta que pongáis el 0x delante de los valores hexadecimales de los registros. Ya se sobreentienden)



MOVI R3,0xFE

BNZ R1, -3
OUT 17,R2

Problema 10

Suponed que en el repertorio de instrucciones del lenguaje SISA solo disponemos de la instrucción de comparación CMPLEU para números naturales. Escribid en lenguaje SISA los siguientes fragmentos de código en C usando solamente como instrucción de comparación el CMPLEU. Suponed que los operadores de comparación son sobre números naturales

a) if (R1<R2) {R5++;}
b) if (R1<=R2) {R5++;}
c) if (R1>R2) {R5++;}
d) if (R1>=R2) {R5++;}

a—
CMPLEU R7, R2, R1,
BNZ R7, 1
ADDI R5, R5, 1

b---
CMPLEU R7, R1, R2
BZ R7,1
ADDI R5, R5, 1

C---
CMPLEU R7, R1, R2
BNZ R7, 1
ADDI R5, R5, 1

D---
AMPLEU R7, R2, R1
BZ R7, 1
ADDI R5, R5, 1

Problema 11

Dados los dos siguientes fragmentos de código en C (el código no tiene que hacer algo útil).

Fragmento 1

```
if (R0<=185) { R1=R0-R1+9; }
else { R1=R2-2; }
R6=-4;
```

Fragmento 2

```
while (R0>R2) {
        R1=R1/4;
        R0=R0-3;
}
R6=not(R1);
```

Completad los fragmentos de programa en lenguaje ensamblador SISA para que el procesador SISC Harvard uniciclo realice las funcionalidades descritas en los fragmentos de código en C. El código SISA ya escrito siempre utiliza el registro R7 para valores temporales. En las comparaciones, hay que interpretar los datos como valores naturales. Rellenad la parte subrayada que falta.

Fragmento 1

@I-Mem	
0x0000	MOVI_ R7, 0xB9
0x0002	MOVHI_ R7, 0X00
0x0004	CMPLEU R7, R0, R7
0x0006	BZ R7, 3
0x0008	SUB R1, R0, R1
0x000A	ADDI R1, R1, 9
0x000C	BNZ R7, 1
0x000E	ADDI R1, R2, 1
0x0010	MOVI R6, -4

Fragmento 2

@I-Mem	
0x0000	CMP___ R7, R0, R2
0x0002	BNZ R7, 4
0x0004	MOVI R7, 2
0x0006	SHL R1, R1, R7
0x0008	ADDI R0, R0, -3
0x000A	BNZ R7, -6
0x000C	NOT_ R6, R1

Problema 12

Dado el siguiente fragmento de código en C (el código no tiene que hacer algo útil).

```
while ((R0<10) || (R2!=R5)) {
        R1=R1/16+R5;
if (R1>R4)
                R6=XOR(R1,0x0440);
        R5--;
```

```
}
  R3=not(R6);
```

Completad el fragmento de programa en lenguaje ensamblador SISA para que el procesador SISC Harvard uniciclo realice las funcionalidades descritas en los fragmentos de código en C. El código SISA ya escrito siempre utiliza el registro R7 para valores temporales. En las comparaciones, hay que interpretar los datos como valores naturales. Rellenad la parte subrayada que falta.

@I-Mem	
0x0000	MOVI_ R7, 10
0x0002	CMPLTU_ R7, R0, R7
0x0004	BNZ R7, ____
0x0006	CMPEQ R7, R2, R5
0x0008	BNZ R7, ____
0x000A	MOVI_ R7, -4
0x000C	SHL_ R1, R1,R7
0x000E	ADD R1, R1, R5
0x0010	CMPLEU R7, R1, R4
0x0012	BNZ R7, -6
0x0014	MOVI R7, 0x24
0x0016	SHL_ R7, R7, R7
0x0018	XOR R6, R1, R7
0x001A	ADDI R5, R5, 23
0x001C	BNZ R7,-10
0x001E	NOT R3, R3

Problema 13

Dado el siguiente fragmento de código en C (el código no tiene que hacer algo útil).

```
for (R0=0; R0<128; R0++) {
R4=R4*2+R5;
    if ((R6>=R2) && (R0!=0))
       R6=not(R2)
}
 R3=R3-R4;
```

Completad el fragmento de programa en lenguaje ensamblador SISA para que el procesador SISC Harvard uniciclo realice las funcionalidades descritas en los fragmentos de código en C. El código SISA ya escrito siempre utiliza el registro R7 para valores temporales. En las comparaciones, hay que interpretar los datos como valores naturales. Rellenad la parte subrayada que falta.

@I-Mem	
0x0000	MOVI R0, 0
0x0002	MOVI__ R7,128

0x0004	CMPLTU R7, R0, R7
0x0006	BZ R7,7
0x0008	SHL R4, R4, R7
0x000A	ADD R4, R4, R5
0x000C	CMPLEU R7, R2, R6
0x000E	BZ R7, -5
0x0010	BZ R0, -6
0x0012	NOT_ R6, R2
0x0014	ADDI_ R0, R0, 1
0x0016	BNZ R0, -9
0x0018	SUB R3, R3, R4

Problema 14

Se ha conectado a la UPG un dispositivo externo de entrada que nos envía valores y que tiene el registro de status en la dirección 9 del espacio de direccionamiento de entrada y el de datos en la 12. También se ha conectado un dispositivo externo de salida que tiene el registro de status en la dirección 3 del espacio de direccionamiento de entrada y el de datos en la dirección 6 del espacio de direccionamiento de salida. Ambos dispositivos tienen un efecto lateral en la lectura/escritura del dato sobre su registro de estado.

Se desea que nuestro sistema vaya leyendo indefinidamente los datos que se reciban por el dispositivo de entrada y se los envíe al dispositivo de salida Nota: No hay que hacer ninguna operación con el valor recibido.

Usando el procesador SISC Harvard uniciclo, escribid el programa en código ensamblador SISA para que realice la función anteriormente descrita.

```
MOVHI R0, 0
IN R7, 9
CMPEQ R7, R7, R0
BNZ R7, -3
IN R1, 12
IN R7, 3
CMPEQ R7, R7, R0
BNZ, -3
OUT R1, 6
BZ R7, -9
```

Problema 15

Usando el procesador SISC Harvard uniciclo, escribid un código ensamblador SISA que vaya leyendo valores (interpretados como números naturales) del teclado y los vaya almacenando en un vector que está en la memoria de datos a partir de la posición 0x1234. Este proceso debe realizarse indefinidamente hasta que el dato almacenado en la memoria sea divisible por 8. En el teclado e impresora el acceso al puerto de datos tiene un efecto lateral sobre el registro de estado.

```
IN R1, KEY-STATUS
BZ R1, -2
IN R1, KEY-DATA
MOVI R2, 0x07
```

```
MOVHI R2, 0x00
AND R3, R1, R2
MOVI R4, 0x34
MOVHI R4, 0x12
ST 0(R4), R1
BZ R3, -10
```

Problema 16

Se ha conectado a la UPG un dispositivo externo de entrada que nos envía valores y que tiene el registro de status en la dirección 4 del espacio de direccionamiento de entrada y el de datos en la 8. Este dispositivo tiene un efecto lateral en la lectura/escritura del dato sobre su registro de estado.

Se desea que este sistema vaya leyendo indefinidamente los datos que se reciban por el dispositivo de entrada mientras el valor de estos datos sea distinto de 0, y los vaya almacenando consecutivamente en la memoria de datos a partir de la posición 0x0064. Los datos recibidos son de 16 bits y primero se almacenará el valor recibido y luego su valor negado bit a bit. Por ejemplo, si el primer valor recibido es el 0xABCD, este valor se almacenará en la posición 0x0064 y en la posición 0x0066 se almacenará su valor negado (0x5432), cuando llegue el segundo valor por el dispositivo de entrada, por ejemplo el valor 0x1234, este valor se almacenará en la posición de memoria 0x0068 y su valor negado (0xEDCB) en la posición 0x006A, etc. Cuando el dato recibido sea 0, el sistema debe quedarse en un bucle infinito sin hacer nada.

Usando el procesador SISC Harvard uniciclo, escribid el programa en código ensamblador SISA para que realice la función anteriormente descrita.