Examen E4 (temas 12, 13 y 14)

- Duración del examen: 2 horas.
- Los problemas tienen que resolverse en las HOJAS DE RESPUESTAS.
- No podéis utilizar calculadora, móvil, apuntes, etc.
- La solución del examen se publicará en Atenea mañana por la tarde.

Ejercicio 1 **(2 puntos)**

El programa en ensamblador SISA que se muestra a continuación se ha traducido a lenguaje máquina situando la sección de código (`.text`) a partir de la dirección `0x0000` de memoria y a continuación la sección de datos (`.data`). Suponed que justo antes de ejecutarse el programa los registros/puertos de entrada `KEY-STATUS` y `KEY-DATA` contienen 0x0001 y 0x0005 respectivamente.

```
.data
     N = 6   ;tiene que ser <= 10
     F:    .space 1
           .even
     V:    .word 2,-5,264,-63,23
           .word 58,-64,32,0,-7
     W:    .space 20
.text
     L1:   IN      R0, KEY-STATUS
           BZ      R0, L1
           IN      R0, KEY-DATA
           MOVI    R2, lo(V)
           MOVHI   R2, hi(V)
           MOVI    R3, lo(F)
           MOVHI   R3, hi(F)
           MOVI    R1, N
           MOVI    R4, 1
     L2:   LD      R5, 0(R2)
           CMPLT   R6, R5, R0
           BZ      R6, L3
           ST      20(R2), R0
           STB     0(R3), R4     ;Flag=1
     L3:   ADDI    R2, R2, 2
           ADDI    R1, R1, -1
           BNZ     R1, L2
.end
```

a) Una vez cargado el programa en memoria: ¿A qué dirección de memoria corresponde la etiqueta, o dirección simbólica, V? *(0.2 puntos)*

V = 0x

b) Una vez cargado el programa en memoria: ¿Cuál es la dirección de memoria y su contenido donde han quedado almacenadas las siguientes instrucciones? *(0.8 puntos)*

Instrucción	@ memoria y contenido
`MOVI R3, lo(F)`	Mem_w[0x] = 0x
`BZ R6, L3`	Mem_w[0x] = 0x
`ST 20(R2), R0`	Mem_w[0x] = 0x

c) Una vez ejecutado el programa en el computador SISC Von Neumann ¿Cuál es la dirección de memoria donde ha escrito por última vez la instrucción (`ST 20(R2), R0`) y cuál es su contenido? *(0.4 puntos)*

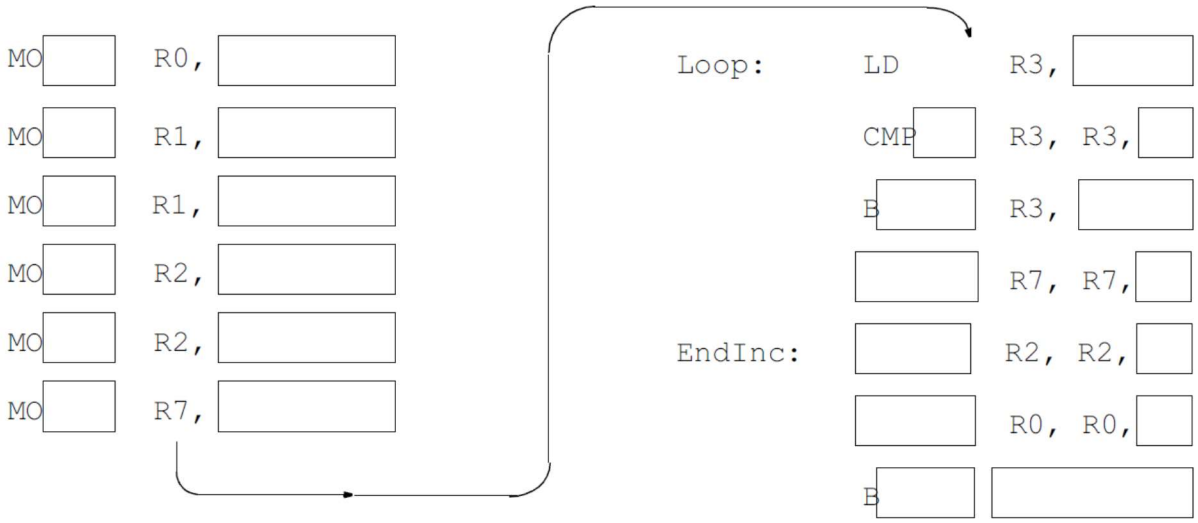
Mem_w[0x] = 0x

d) ¿ Cuántas instrucciones se ejecutan y cuánto tarda en ejecutarse el código en las tres versiones de los computadores SISC? Recordad que el tiempo de ciclo del Harvard uniciclo, el Harvard multiciclo y el Von Neumann es de 4.000, 1.000 y 1.400 u.t. respectivamente. *(0.6 puntos)*

Nº de ciclos (H. uniciclo)=	Nº de ciclos (H. multiciclo)=	Nº de ciclos (V.Neumann)=
Tejec(Harv.uni.) =	Tejec(Harv.multi.) =	Tejec(V.Neumann) =

Ejercicio 2 (2 puntos)

El vector de 100 números naturales `V[100]` se encuentra en la memoria a partir de la dirección simbólica `V` (`V[0]` en la dirección `V`, `V[1]` en la `V+2`, etc.). Completad el código SISA para que deje en `R7` el número de elementos del vector que son mayores que 38164. `R0` controla que se procesen los 100 elementos, `R1` contiene la constante 38164, `R2` es un puntero a los elementos de `V` y `R3` mantiene temporalmente resultados intermedios.



Ejercicio 3 (1.5 puntos)

Cada uno de los apartados pregunta sobre un ciclo concreto de la ejecución de varias instrucciones en el SISC Von Neumann. Escribid el contenido del registro IR, el contenido de la ROM_OUT y 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**.** 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. Podéis ver el grafo de estados de Moore de la UC en el anexo. Suponed que el contenido de todos los registros, `Rk` para k=0,...7, antes de ejecutarse cada instrucción es 0.

a) Indica el contenido del registro IR. *(0.3 puntos)*

Nodo / Estado (Mnemo Salida)	**Instrucción en IR** (en ensamblador)	**Valor del IR** (en hexadecimal)
D	`ADD R1,R2,R3`	0x
Movhi	`MOVHI R6,2`	0x
Out	`OUT 73, R2`	0x

b) Indica el contenido de la ROM_OUT en hexadecimal usando las conexiones en el orden que están en el anexo (o chuletario). Utilizad el valor 0 para los bits que sean x (solo para este apartado). *(0.6 puntos)*

Nodo / Estado (Mnemo Salida)	**Instrucción en IR** (en ensamblador)	**Contenido ROM_OUT** (en hexadecimal)
D	`ADD R1,R2,R3`	0x
Movhi	`MOVHI R6,2`	0x
Out	`OUT 73, R2`	0x

c) Indica 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 podemos suponer cómo se han implementado las x en la ROM_OUT). *(0.6 puntos)*

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	LdIr	LdPc	Byte	Alu/R@	R@/Pc	N (hexa)	ADDR-IO (hexa)
a	D	`ADD R1,R2,R3`																			
b	Movhi	`MOVHI R6,2`																			
c	Out	`OUT 73, R2`																			

Ejercicio 4 (1.5 puntos)

Especificad el **camino crítico** (indicando la suma ordenada de los tiempos de propagación de los bloques por los que pasa) y calculad el **tiempo de ciclo mínimo** para que el computador **SISC Von Neumann** pueda ejecutar correctamente el tipo de instrucción SISA que se indica en cada apartado (este sería el tiempo de ciclo mínimo del computador si solo ejecutara instrucciones como la indicada u otras que requieran menor tiempo). No tenéis que añadir ningún porcentaje de seguridad en el cálculo del tiempo de ciclo mínimo. Suponed que los tiempos de propagación de los bloques que forman el computador son los siguientes:

T_p(ROM_Q+) = 50 u.t.

T_p(ROM_OUT) = 80 u.t.

T_p(MUX-2-1) = 50 u.t.

T_p(MUX-4-1) = 100 u.t.

T_p(REG) = 100 u.t. // Tiempo de propagación de un registro.

T_p(REGFILE) = 200 u.t. // Tiempo de lectura del banco de registros

T_p(ALU-slow) = 700 u.t. // Tp de la ALU para las operaciones/funciones lentas: `ADD`, `SUB`, `CMP*`.

T_p(ALU-quick) = 300 u.t. // Tp de la ALU para las operaciones/funciones rápidas: cualquier otra distinta de `ADD`, `SUB`, `CMP*`.

T_{acc}(MEMORY) = 1000 u.t. // Tiempo de acceso (para la lectura o escritura) a la memoria

T_p(AND-2) = T_p(OR-2) = 20 u.t.

T_p(NOT) = 10 u.t.

El tiempo de propagación de un bloque combinacional (T_p) y el tiempo de acceso a memoria para realizar una lectura (T_{acc}) es el tiempo desde que están estables todas las entradas necesarias hasta que se estabilizan las salidas requeridas al valor correcto para las entradas aplicadas. Desconocemos como se han implementado internamente los bloques (y podría ser de forma diferente a los vistos en clase). Recordad que un registro con señal de carga (Ld), REGwLd, está construido con un REG y un MUX-2-1 (no os damos el esquema interno del REGwLd, porque lo tenéis que saber).

a) T_c correspondiente al nodo de **D** (decode).

b) T_c correspondiente al nodo de **Bnz**.

c) T_c correspondiente al nodo de **Cmp**.

Ejercicio 5 **(3 puntos)**

Apartado a)

Vamos a analizar diferentes aspectos de la ejecución, en el computador SISC Von Neumann, del siguiente fragmento de código:

```
MOVI R7, -2
SHA  R0, R1, R7
```

a.1) Si antes de comenzar a ejecutar la primera instrucción el estado del computador es el siguiente:

`PC=0x348E,  R0=0x348E,  R1=0x82A5,  R7=0x0002`

¿Completad el estado del computador que queda después de ejecutada la segunda instrucción? *(0.6 puntos)*

`PC=0x  R0=0x  R1=0x  R7=0x`

MEM_b`[0x348E]=0x` MEM_b`[0x348F]=0x`

a.2) Sea Nc el número de ciclos que tarda en ejecutarse el código. ¿Cuánto vale Nc? *(0.2 puntos)*

`Nc=`

a.3) Completad las 4 filas de la siguiente tabla que hacen referencia a los ciclos de ejecución 3, 4, 5 y 6 (los ciclos de ejecución se numeran desde el 1 al Nc). Podéis ver los mnemotécnicos de salida de cada Nodo en el grafo de la chuleta. En la columna Acciones usad el lenguaje de transferencia de registros de la documentación de la asignatura (o una notación igual de precisa). Recordad que en la Palabra de control compactada solo hay que expresar el valor de los campos que son significativos en ese ciclo (no hay que indicar los campos que valen x ni las señales de permiso de modificación del estado que valen 0). Completad esta tabla para el **caso general** del código **`MOVI Rd,N8`** y **`SHA Rd,Ra,Rb`**. *(0.8 puntos)*

Ciclo	Nodo Mnemo Salida	Acciones	Palabra de control compactada
3			
4			
5			
6			

Apartado b)

Completad el diseño del SISC Von Neumann para que pueda ejecutar, además de las 25 instrucciones originales SISA, la nueva instrucción

`SHAI Rd, Ra, N5`

que al ejecutarse modifica el estado del computador de la misma forma que lo hace la instrucción SHA pero siendo el segundo operando un inmediato de 5 bits que codifica un número entero en complemento a 2. Con esta nueva instrucción podemos hacer lo mismo que en el SISA original requiere dos instrucciones (como las dos del fragmento de código del *Apartado a*).

La nueva instrucción se tiene que poder ejecutar sin efectuar ningún cambio en el computador excepto el contenido de la ROM_Q+ y el de la ROM_OUT. La nueva instrucción se tiene que ejecutar en el mínimo número de ciclos. La ejecución de las 25 instrucciones SISA originales se hace exactamente igual que antes de introducir la nueva instrucción. Se pide:

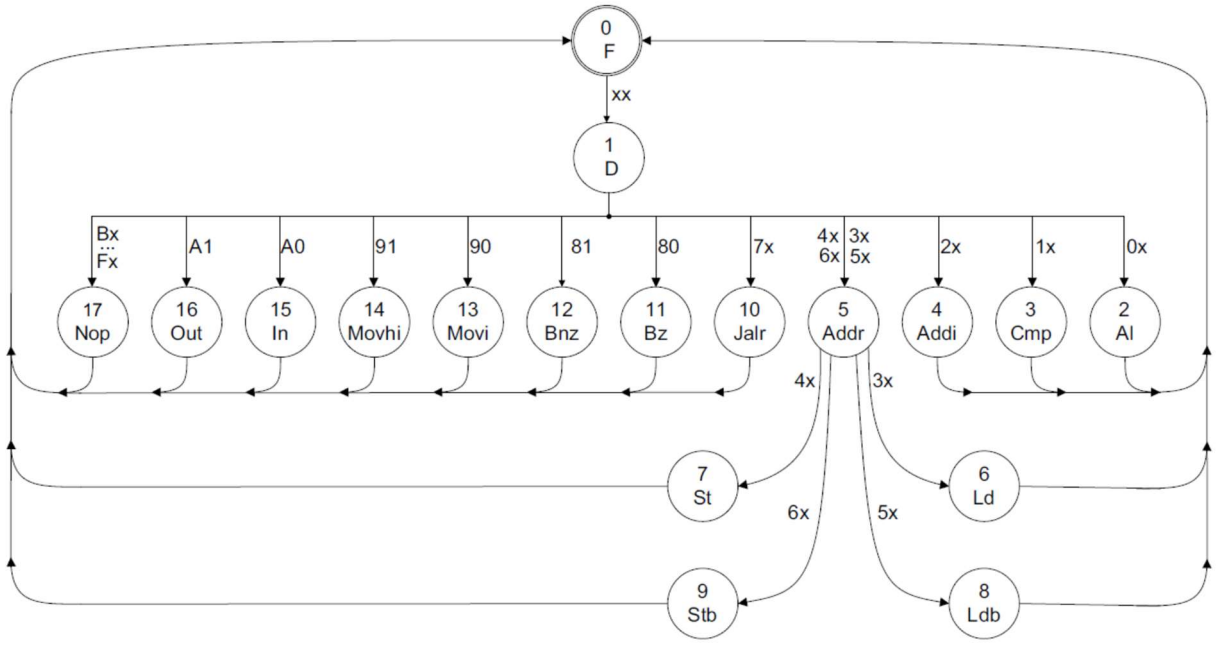
b.1) Proponed el formato y la codificación en binario de la nueva instrucción (cada bit puede valer 0, 1, a, b, d, n, x) y escribid la semántica usando un fragmento de código en un lenguaje de alto nivel tipo C, como el empleado para esto en la documentación. *(0.2 puntos)*

`Codificación:`

`Sintaxis:  SHAI Rd, Ra, N5`

`Semántica:`

b.2) Modificad el grafo de estados de la unidad de control (añadiendo nodos y arcos y/o modificando los ya existentes) dibujando sobre y/o al lado del siguiente grafo original. *(0.3 puntos)*



b.3) Completad la tabla usando una fila para cada nuevo nodo que hayáis añadido al grafo (no son más de 2 nodos). *(0.3 puntos)*

Nodo			
Estado (decimal)	**Mnemo salida**	**Acciones**	**Palabra de control compactada**

b.4) Indicad la dirección o las direcciones (en binario, con x cuando sea posible para referirnos a más de una dirección) de la ROM_Q+ y su contenido (en hexadecimal) para implementar correctamente el arco que va del nodo/estado 1 (con mnemotécnico de salida D) al primer nuevo nodo usado para implementar la nueva instrucción. *(0.3 puntos)*

	Dirección o direcciones (en binario)	**Contenido** (en hexa)
Arco del nodo 1 (D) al nodo		

b.5) Completad (poniendo 0, 1 o x en cada bit) la siguiente tabla que especifica, mediante una fila por cada nuevo nodo añadido al grafo de estados de la unidad de control (no son más de 2 nodos), la dirección (en decimal) de la ROM_OUT y su contenido para que se ejecute correctamente la nueva instrucción, poniendo el máximo número de x posibles. En la columna de la derecha poned el mnemotécnico de salida que habéis usado para nombrar cada nuevo nodo en el grafo del apartado b.2). *(0.3 puntos)*

@ROM	Bnz	Bz	WrMem	RdIn	WrOut	WrD	LdIr	Byte	R@/Pc	Alu/R@	Pc/Rx	Ry/N	$P/I/L/A_1$	$P/I/L/A_0$	OP_1	OP_0	MxN_1	MxN_0	MxF	F_2	F_1	F_0	$Mx@D_1$	$Mx@D_0$	**Nodo (Mnemo Salida)**