Apellidos y nombre: ..... Grup: ..... DNI:.....

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

- Duración del examen: 1 hora y 45 minutos
- La solución de cada ejercicio 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 19 de mayo.

Ejercicio 1 **(2 puntos)**

a) Indicad el valor que debe tener cada uno de los bits de la palabra de control de la UPG (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 tachad **toda la línea** de señales. *(1 punto)*

Mnemotécnico	@A	@B	Rb/N	OP	F	In/Alu	@D	WrD	N *(hexa)*
`SHLI R2, R4, -2`									
`CMPLT -, R9, R5`									
`OUT R5 // ADD R2, R5, R4`									
`AND R0, R0, R0`									

b) Indicad el mnemotécnico que corresponde a cada una de las siguientes palabras de control de la UPG (sin subsistema de I/O ni memoria). *(1 punto)*

Mnemotécnico	@A	@B	Rb/N	OP	F	In/Alu	@D	WrD	N *(hexa)*
	100	xxx	x	10	000	0	111	1	X X X X
	101	xxx	0	01	001	x	xxx	0	F F 3 2
	000	010	1	00	010	0	001	1	X X X X
	101	010	1	01	011	0	010	1	X X X X

Ejercicio 2 **(0.75 puntos)**

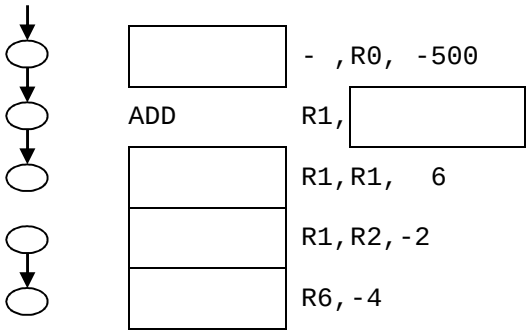
Completad la siguiente tabla ensamblando las instrucciones en ensamblador SISA o desensamblando las instrucciones en lenguaje máquina según sea necesario. Indica poniendo NA en la casilla aquellos casos en los que la instrucción no corresponda al lenguaje SISA.

Lenguaje máquina SISA	**Lenguaje ensamblador SISA**
`0x1034`	
	`LDB R6, -2(R0)`
`0xAFE0`	

Ejercicio 3 **(1,5 puntos)**
Dado el siguiente fragmento de código en C (el código no tiene que hacer algo útil), indicad como se implementaría en un procesador que use la UPG vista en clase, utilizando la UC de **propósito específico** (UCe) y la UP de **propósito general** (UPG). Todos los datos son **enteros**.

```
if (R0 > -500) { R1 = R0+R1-6; }
else  { R1 = R2 / 4; }
R6=-4;
```

a) Completad el fragmentos de grafo de estados de la UC de **propósito específico** para que junto con la UPG formen un procesador que realice la funcionalidad descrita en el fragmento de código anterior. 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. *(0.75 puntos)*



b) Completad el fragmento de programa en lenguaje ensamblador SISA para que el procesador formado por la unidad de control de propósito general (UCG) junto con la UPG realice la funcionalidad descrita en el fragmento de código en C (el código no tiene que hacer algo útil). El código SISA ya escrito siempre utiliza el registro R7 para valores temporales. En las comparaciones, hay que interpretar los datos como valores **enteros**. Rellenad la parte subrayada que falta. *(0.75 puntos)*

@I-Mem	
0x0000	_____ R7, ____
0x0001	MOV____ R7, ____
0x0002	CMP____ R7, R0, R7
0x0003	B_____ R7, ____
0x0004	ADD R1, _____
0x0005	_____ R1, R1, ____
0x0006	B_____ R7, ____
0x0007	MOV____ R7, ____
0x0008	_____ R1, R2, ____
0x0009	_____ R6, -4

Apellidos y nombre: …… Grup: …… DNI:……

Ejercicio 4 **(1,75 puntos)**

Indicad qué cambios se producen 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 0x3000, el contenido de todos los registros es 0xBEBE y que el contenido de todas las posiciones pares de la memoria de datos es 0x34 y el de todas las posiciones impares de la memoria de datos es 0x86. Utilizad los mnemotécnicos MEM_b[…] y MEM_w[…] para indicar los cambios en la memoria (a nivel de byte y word respectivamente).

Instrucción a ejecutar	**Cambios en el estado del computador**
`STB -2(R6), R2`	
`BZ R7, -6`	
`SHA R4, R2, R0`	
`LDB R4,3(R2)`	

Ejercicio 5 (**1,5 puntos)**

Indicad el contenido de la tabla de la ROM (**sólo las celdas sombreadas**) correspondiente al bloque ROM_CRTL_ LOGIC. Indicad los valores que tomarían las señales para ejecutar correctamente las instrucciones. Indicad 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/a1	-/i/l/ao	OP_1	OP_0	MxN1	MxN0	MxF	f2	f1	f0	MxD1	MxD0	
0	0	0	0	X																					A/L
0	0	0	1	X																					CMP
0	0	1	0	X																					ADDI
0	0	1	1	X																					LD
0	1	0	0	X																					ST
0	1	0	1	X																					LDB
0	1	1	0	X																					STB
0	1	1	1	X																					-
1	0	0	0	0																					BZ
1	0	0	0	1																					BNZ
1	0	0	1	0																					MOVI
1	0	0	1	1																					MOVHI
1	0	1	0	0																					IN
1	0	1	0	1																					OUT
1	0	1	1	X																					-
1	1	X	X	X																					-

Ejercicio 6 **(2,5 puntos)**

Sobre un SISC-Harv uniciclo (UCG+UPG+IO+Memoria) y dado un vector almacenado en memoria que contiene números naturales codificados en 16 bits, queremos calcular cuántos elementos del vector son pares y cuántos son impares.

- El número de elementos del vector, N, se leerá del teclado. Nos garantizan que N será mayor que cero.
- El primer elemento del vector se almacena a partir de la posición `0xD0CE`. Los siguientes elementos se almacenan en posiciones consecutivas.
- El resultado debe mostrarse por impresora. En primer término la cantidad de números pares y a continuación la cantidad de números impares.
- Los puertos de entrada/salida son los vistos en teoría (`KEY-DATA, KEY-STATUS, PRINT-STATUS, PRINT-DATA`) con efecto lateral.

Nos facilitan una versión incompleta de la solución. Completad (en los espacios subrayados) el código ensamblador SISA para que realice la función anteriormente descrita.

```
                              ; Inicializaciones
MOV___ R3, _________________
MOV___ R3, _________________  ; carga dirección de inicio del vector
MOVI   R4, 0x00               ; contador números pares R4 = 0x0000
MOVI   R5, 0x01               ; máscara R5 = 0x0001
_____________________________
_____________________________
_____________________________  ; lectura de N sobre R1 desde teclado
OR     R2, R1, R1             ; copia de R1 sobre R2
                              ; Bucle principal
LD     R6, _________________
AND    ____________________
B______________________________
ADDI   R4, R4, 1
______ R3, _________________
______ R1, _________________
B_____ R1, _________________
                              ; Presentación resultados
_____________________________  ; guarda en R2 la cantidad de números impares
_____________________________
_____________________________
_____________________________  ; imprime cantidad de números pares
_____________________________
_____________________________
_____________________________  ; imprime cantidad de números impares
```

Además, contestad las siguientes preguntas:

1. Si los datos del vector fuesen naturales codificados en 8 bits almacenados en posiciones consecutivas, ¿cómo debería modificarse el código anterior?

2. Con la instrucción `OR R2, R1, R1` (octava instrucción del programa) copiamos el valor actual de R1 sobre R2. Indicad otra instrucción SISA que permita realizar la misma operación.