Apellidos: Nombre:

1)[4 ptos]Utilizando los elementos hardware proporcionados a continuación (ALU, banco de registros BR, memorias, etc...) diseña la unidad de datos y unidad de control de un procesador de tamaño de palabra de 16 bits para ejecutar las siguientes instrucciones:

Instrucción	Acción	Formato de instrucción
JACNZra	Si AC!=0entonces PC ←(BR[ra])[7:0]	Opcode: 1 1 1 1 1 | ra: | x x x x x x x x
SUBdra, dir	AC←BR[ra]- M(dir)	Opcode: 1 1 1 1 0 | ra: | dir:

Para el diseño de la unidad de datos puedes utilizar los elementos hardware adicionales que creas oportunos (registros, multiplexores, etc). Tanto la memoria de datos como la memoria de instrucciones se direccionan a nivel de palabra.

NOTA: Diseña el camino de datos que incluya únicamente los elementos (incluyendo caminos) mínimos para ejecutar solamente estas instrucciones.
Ten cuidado con el tamaño de los buses, no todos los elementos del diseño tienen el mismo número de bits.

Especificación del funcionamiento de los elementos hardware:

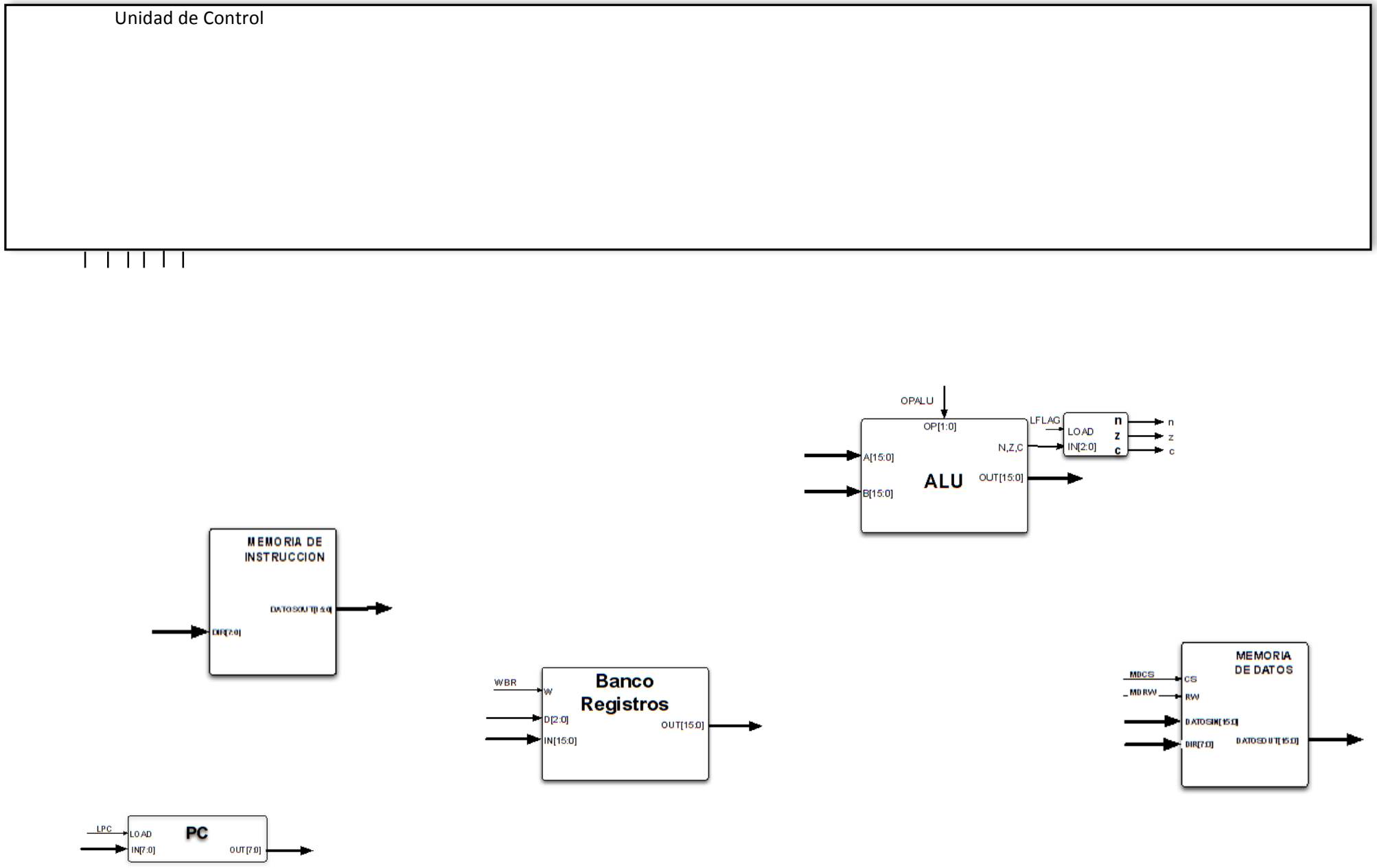
Registros (PC, AC):
- Continuamente el contenido del registro está en OUT[:]
- LOAD = 1 → IN[:] se almacena en el registro en el flanco activo del reloj

Banco de registros (BR):
- Continuamente el contenido del registro especificado por D[2:0] está en OUT[15:0]
- Write = 1 → IN[15:0] se almacena en el registro especificado por D[2:0] en el flanco activo del reloj

Memoria de instrucción:
- Continuamente el contenido de la posición de memoria especificada en DIR[7:0] está en DATOSOUT[15:0]

Memoria de datos:
- Read = 1 → el contenido de la posición de memoria especificada en DIR[7:0] aparece en DATOSOUT[15:0]
- Write = 1 → DATOSIN[15:0] se almacena en la posición de memoria especificada en DIR[7:0] en el flanco activo del reloj
- Si Read = 0 y Write = 0 → DATOSOUT[15:0] está en alta impedancia (desconectado)

ALU:
- OP[1:0] = 00 → OUT[15:0] = A[15:0] + B[15:0] (Suma aritmética)
- OP[1:0] = 01 → OUT[15:0] = A[15:0] - B[15:0] (Resta aritmética)
- OP[1:0] = 10 → OUT[15:0] = A[15:0] AND B[15:0] (AND bit a bit)
- OP[1:0] = 11 → OUT[15:0] = A[15:0] OR B[15:0] (OR bit a bit)
- Salida N se activa (1) si OUT[15:0] es un número negativo
- Salida Z se activa (1) si OUT[15:0] es cero
- Salida C se activa (1) si se ha producido acarreo en la operación realizada por la ALU


Unidad de Control
OPALU
OP[1:0]
LFLAG
LOAD
IN[2:0]
n
z
c
N,Z,C
A[15:0]
B[15:0]
ALU
OUT[15:0]
MEMORIA DE INSTRUCCION
DATOSOUT[15:0]
DIR[7:0]
WBR
W
D[2:0]
IN[15:0]
Banco Registros
OUT[15:0]
MDCS
MDRW
CS
RW
DATOSIN[15:0]
DIR[7:0]
MEMORIA DE DATOS
DATOSOUT[15:0]
LPC
LOAD
IN[7:0]
PC
OUT[7:0]

2) [0.5 ptos] Representa el siguiente número en formato estándar de punto flotante IEEE754 de 32 bits.

-27.625

3) [0.5ptos]¿A qué número corresponde la siguiente representación del estándar de punto flotante IEEE754 de 32 bits dada en hexadecimal?

FE000000

Representación binaria

Valor representado

4)[1 pto]Representa los siguientes valores enteros siguiendo los sistemas de representación que aparecen en la tabla, utilizando 8 bits para cada caso. En caso de no poder representar el valor en algún sistema, indícalo claramente.

Valor	Binario natural	S-M	BCD	C2	Exceso 64
37					
-96					

5)[0.5 pto]Dada la siguiente cadena binaria de 8 bits, determina el valor numérico que representa si se consideran que está representando números según los sistemas de representación que aparecen en la tabla. Si no representa ningún valor numérico válido indícalo claramente.

Cadena	Binario natural	S-M	C2	Exceso 128	BCD
10101010					

6)[0.5 ptos]Dada la siguiente cadena binaria que ha sido codificada en Hamming SEC-DED ($x_7x_6x_5p_4x_3p_2p_1p$), determina si se ha producido error en uno o dos bits (considera despreciable la probabilidad de error en 3 bits), y si es posible, indica el dato correcto que se quería enviar ($x_7x_6x_5x_3$).

Cadena binaria $x_7x_6x_5p_4x_3p_2p_1p$	Nº de errores	Bits erróneos	Dígito correcto $x_7x_6x_5x_3$
10001111			

7) [3 ptos]Sea una arquitectura RISC con un conjunto de instrucciones similar al del procesador MIPS visto en clase. Su camino de datos presenta una segmentación en 4 etapas y dos memorias una para instrucciones y otra para datos. Esto hace que las instrucciones se ejecuten según se muestra a continuación:

Instrucción	1	2	3	4	5	6	7
Instr. 1	IFD	REX	MEM	WB			
Instr. 2		IFD	REX	MEM	WB		
Instr. 3			IFD	REX	MEW	WB	
Instr. 4				IFD	REX	MEW	WB

donde:

- IFD es la etapa de búsqueda (**lectura de la instrucción de memoria**) y decodificación de instrucción.
- REX es la etapa de búsqueda de operandos (**lectura del banco de registros**), ejecución de operación ó resolución de condiciones de salto y cálculo de dirección para operando destino ó salto y actualización del PC para instrucciones de salto (**escritura en PC**) incondicionales.
- MEM es la etapa de acceso a memoria para instrucciones de carga o almacenamiento (**lectura/escritura en memoria**) y actualización del PC para instrucciones de salto (**escritura en PC**) condicionales.
- WB es la etapa de escritura de resultados en registros (**escritura en banco de registros**) para instrucciones aritmético/lógicas y de carga.

Se ha realizado adelantamiento en el banco de registros, de forma que se puede escribir un determinado registro en un ciclo y en el mismo ciclo leer el valor almacenado.

Representar en un diagrama de ciclos la evolución del cauce para el siguiente trozo de código e **identifica los riesgos** que puedan darse, indicando su tipo (estructural, datos, control) y la causa que los provoca (unidad funcional para los estructurales, número de los registros para los de datos).

Instrucciones	1	2	3	4	5	6	7	8	9
lw $1, 0($2)									
salt: subi $2, $1, 1									
sw $5, 0($2)									
beq $2, $0, etq									
add $5, $5, $1									
j salt									
etq: sub $6, $6, $5									
	¿riesgo?	¿riesgo?	¿riesgo?	¿riesgo?	¿riesgo?	¿riesgo?	¿riesgo?	¿riesgo?	¿riesgo?

(b) Modifica el código para evitar los riesgos. Haz dos versiones, en la primera, inserta el menor número de NOPs necesarios pero sin reordenar el código. En la segunda versiónreordena el código, si es posible, para eliminar el mayor número de NOPs y que el programa sigua siendo correcto.

Código 1	Código 2