Ayudantía 11

Andrés González

Laurence Golborne

Rodrigo Alonso

Susana Figueroa

Tomás Contreras

003-1040559 1250 003-77156.8 1760 0009-14563.7 73273

Temas

- Lógica Digital
- Representaciones numéricas
- Hardware sin manejo de memoria
- Hardware con manejo de memoria

01
02
03
04
05
06

Pregunta 1.a

Convierta a complemento de 2 el número $(106)_{10}$:

106 % 2 = 0

53 % 2 = 10   * 106 / 2 = 53.

26 % 2 = 010

13 % 2 = 1010

6 % 2   = 01010

3 % 2   = 101010

1 % 2   = 1101010

-> Si se trata de un número de 32 bits, tenemos que rellenar con ceros a la izquierda:

0000000 0000000 00000000 01101010

Pregunta 1.b

Transforme a base decimal el número $11111111\ 11111111\ 11111110\ 10100100_2$ en complemento a 2.

Inverso aditivo:

11111111 11111111 11111110 10100100 -> lo negamos

00000000 00000000 00000001 01011011 -> sumamos 1

00000000 00000000 00000001 01011100

Luego:

$1x2^8 + 0x2^7 + 1x2^6 + 0x2^5 + 1x2^4 + 1x2^3 + 1x2^2 + 0x2^1 + 0x2^0$

$= 256 + 0 + 64 + 0 + 16 + 8 + 4 + 0 + 0 = 348$

Pero, se trata de un negativo, por lo que tenemos que el número es -348.

Pregunta 2

Hacer una compuerta XOR con compuertas NAND.

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

A	B	A NAND B
0	0	1
0	1	1
1	0	1
1	1	0

Pregunta 2

Hacer una compuerta XOR con compuertas NAND.

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

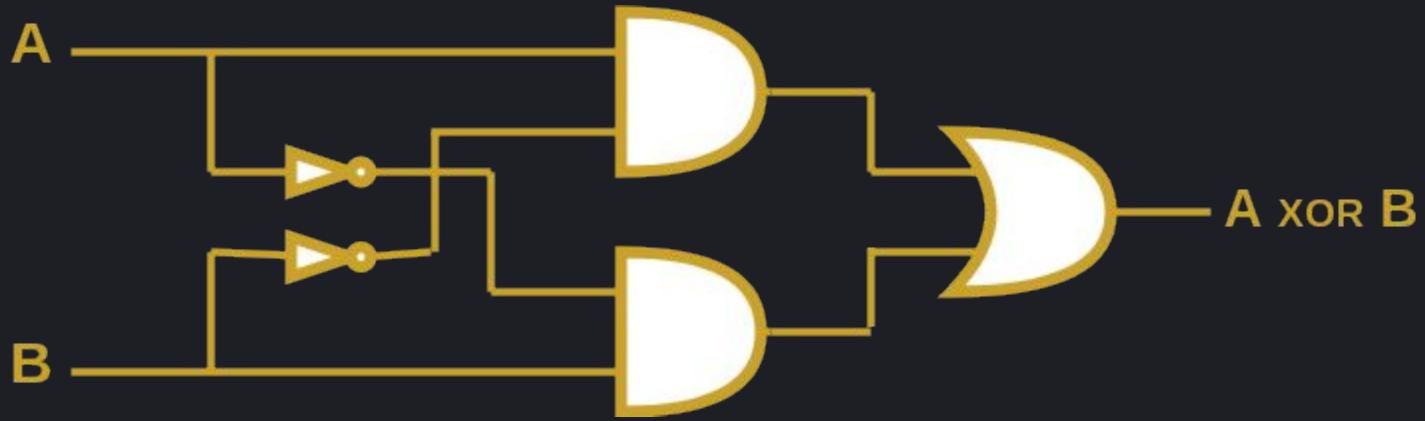
Pregunta 2

Hacer una compuerta XOR con compuertas NAND.

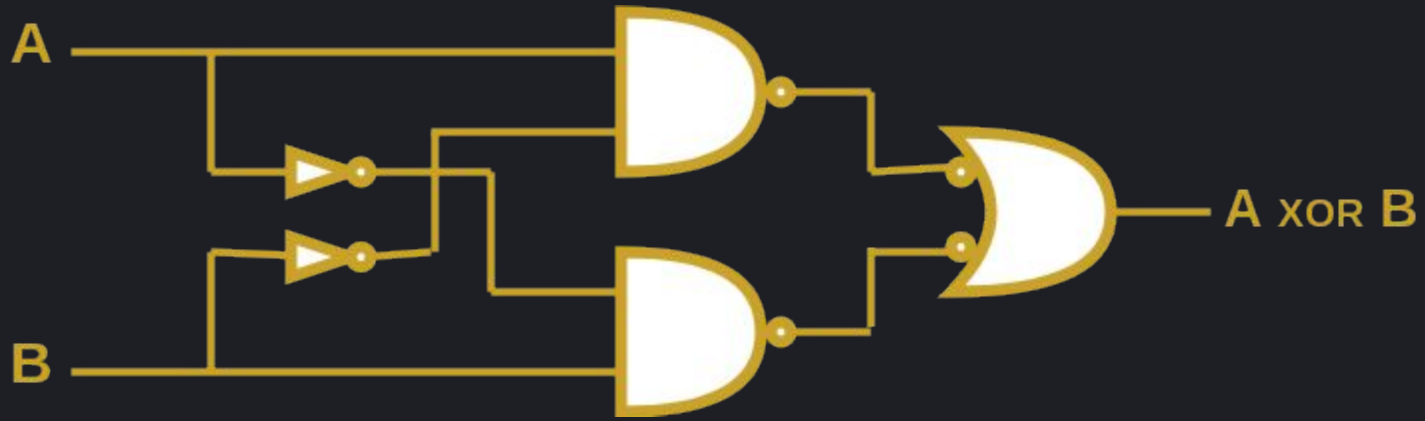
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

$$f(x,y) = (A \cdot \overline{B}) + (\overline{A} \cdot B)$$

Pregunta 2



Pregunta 2



Pregunta 2

Pregunta 2

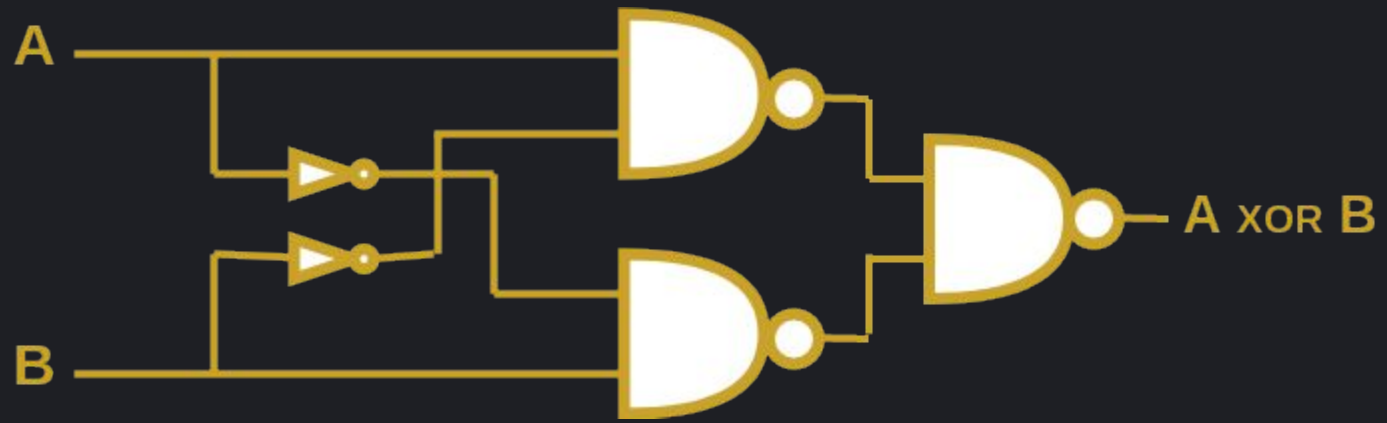
A	B	Output
0	0	1
0	1	1
1	0	1
1	1	0

Pregunta 2

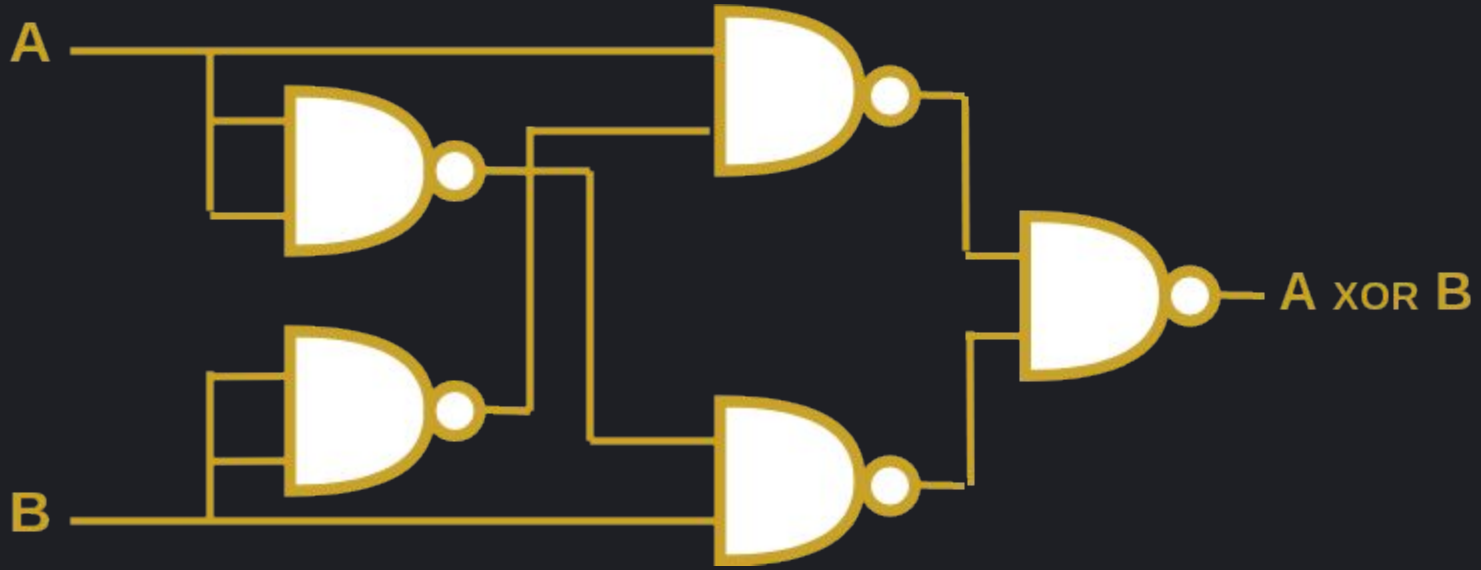
A
B

A	B	A NAND B
0	0	1
0	1	1
1	0	1
1	1	0

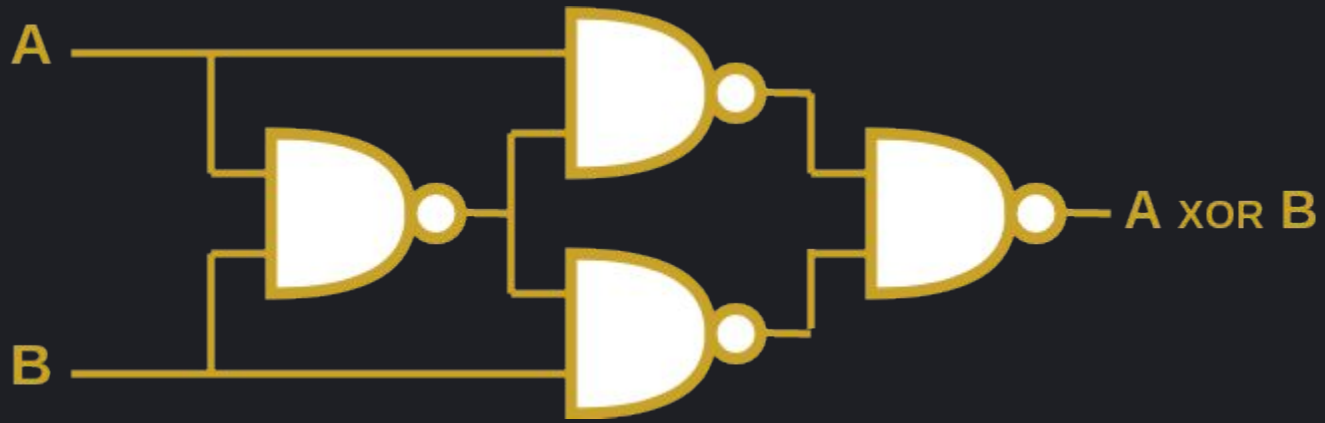
Pregunta 2



Pregunta 2

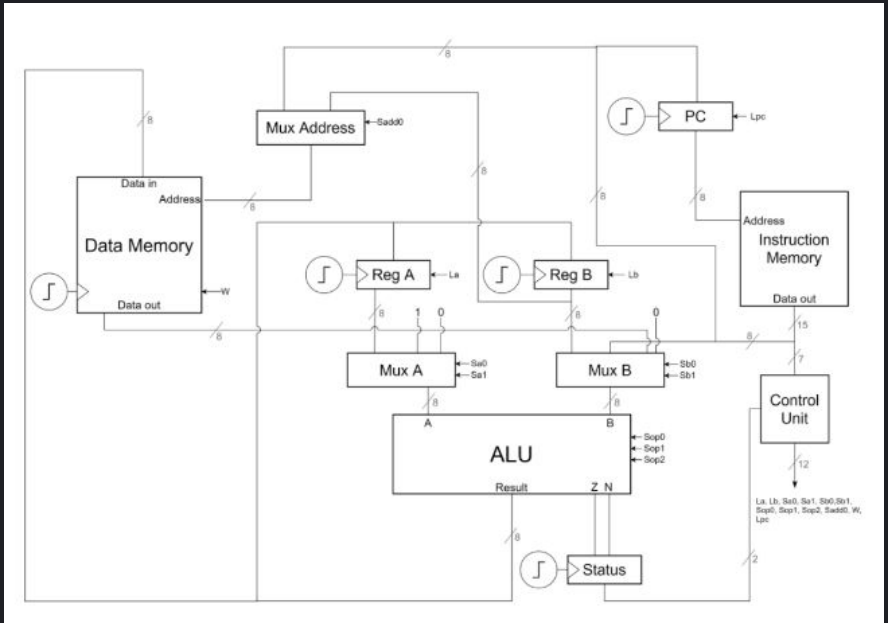


Pregunta 2



Pregunta 3

Transforme la arquitectura del computador básico para que esta soporte como tipos de datos nativos números enteros de 8 y 16 bits de manera independiente.



Pregunta 3

Transforme la arquitectura del computador básico para que esta soporte como tipos de datos nativos números enteros de 8 y 16 bits de manera independiente.

Solución:

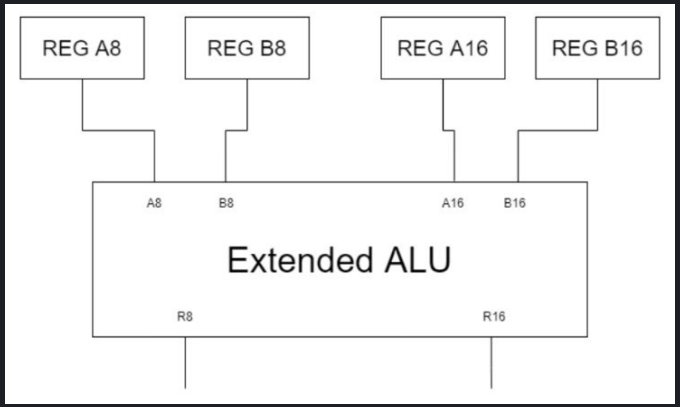
Agregar **registros extendidos** de 16 bits independientes a una **ALU extendida** que pueda recibir a 4 registros (2 de 8 bits y 2 de 16 bits).

Pregunta 3

Transforme la arquitectura del computador básico para que esta soporte como tipos de datos nativos números enteros de 8 y 16 bits de manera independiente.

Solución:

Agregar **registros extendidos** de 16 bits independientes a una **ALU extendida** que pueda recibir a 4 registros (2 de 8 bits y 2 de 16 bits).



Pregunta 3

Transforme la arquitectura del computador básico para que esta soporte como tipos de datos nativos números enteros de 8 y 16 bits de manera independiente.

Solución:

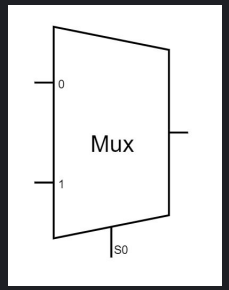
Además se debe modificar el **almacenamiento en memoria** para que este pueda soportar 16 bits. Esto se puede hacer de muchas maneras distintas. Una puede ser añadiendo un multiplexor para que la memoria sepa cuando deba manejar 8 bits o 16 bits, o bien se pueden definir 2 ciclos, para que se lea/escriba una palabra en cada ciclo y así completar los 16 bits.

Pregunta 3

Transforme la arquitectura del computador básico para que esta soporte como tipos de datos nativos números enteros de 8 y 16 bits de manera independiente.

Solución:

Además se debe modificar el **almacenamiento en memoria** para que este pueda soportar 16 bits. Esto se puede hacer de muchas maneras distintas. Una puede ser añadiendo un multiplexor para que la memoria sepa cuando deba manejar 8 bits o 16 bits, o bien se pueden definir 2 ciclos, para que se lea/escriba una palabra en cada ciclo y así completar los 16 bits.



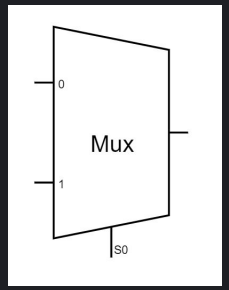
Si no hay un multiplexor, que puede pasar con respecto a los 8 bits?

Pregunta 3

Transforme la arquitectura del computador básico para que esta soporte como tipos de datos nativos números enteros de 8 y 16 bits de manera independiente.

Solución:

Además se debe modificar el **almacenamiento en memoria** para que este pueda soportar 16 bits. Esto se puede hacer de muchas maneras distintas. Una puede ser añadiendo un multiplexor para que la memoria sepa cuando deba manejar 8 bits o 16 bits, o bien se pueden definir 2 ciclos, para que se lea/escriba una palabra en cada ciclo y así completar los 16 bits.



Si no hay un multiplexor, que puede pasar con respecto a los 8 bits?

Los carry pueden irse al mundo de 16 bits 😱

Pregunta 3

Transforme la arquitectura del computador básico para que esta soporte como tipos de datos nativos números enteros de 8 y 16 bits de manera independiente.

Solución:

No debemos olvidar que también se debe modificar la **ISA** para esta arquitectura

ADD8, ADD16, SUB8, SUB16, …

Los **opcodes** correspondientes también deben ser creados.

Pregunta 3

Instrucción	Operandos	Opcode	La	Lb	Sa0	Sb0	Sb1	Sop2	Sop1	Sop0	Operación
MOV	A,B	000000	1	0	1	0	0	0	0	0	A=B
	B,A	000001	0	1	0	1	1	0	0	0	B=A
	A,Lit	000010	1	0	1	0	1	0	0	0	A=Lit
	B,Lit	000011	0	1	1	0	1	0	0	0	B=Lit
ADD	A,B	000100	1	0	0	0	0	0	0	0	A=A+B
	B,A	000101	0	1	0	0	0	0	0	0	B=A+B
	A,Lit	000110	1	0	0	0	1	0	0	0	A=A+Lit
SUB	A,B	000111	1	0	0	0	0	0	0	1	A=A-B
	B,A	001000	0	1	0	0	0	0	0	1	B=A-B
	A,Lit	001001	1	0	0	0	1	0	0	1	A=A-Lit
AND	A,B	001010	1	0	0	0	0	0	1	0	A=A and B
	B,A	001011	0	1	0	0	0	0	1	0	B=A and B
	A,Lit	001100	1	0	0	0	1	0	1	0	A=A and Lit
OR	A,B	001101	1	0	0	0	0	0	1	1	A=A or B
	B,A	001110	0	1	0	0	0	0	1	1	B=A or B
	A,Lit	001111	1	0	0	0	1	0	1	1	A=A or Lit
NOT	A,A	010000	1	0	0	0	0	1	0	0	A=notA
	B,A	010001	0	1	0	0	0	1	0	0	B=notA
XOR	A,A	010010	1	0	0	0	0	1	0	1	A=A xor B
	B,A	010011	0	1	0	0	0	1	0	1	B=A xor B
	A,Lit	010100	1	0	0	0	1	1	0	1	A=A xor Lit
SHL	A,A	010101	1	0	0	0	0	1	1	0	A=shift left A
	B,A	010110	0	1	0	0	0	1	1	0	B=shift left A
SHR	A,A	010111	1	0	0	0	0	1	1	1	A=shift right A
	B,A	011000	0	1	0	0	0	1	1	1	B=shift right A

Pregunta 3

Transforme la arquitectura del computador básico para que esta soporte como tipos de datos nativos números enteros de 8 y 16 bits de manera independiente.

Solución:

No debemos olvidar que también se debe modificar la **ISA** para esta arquitectura

ADD8, ADD16, SUB8, SUB16, …

Los **opcodes** correspondientes también deben ser creados. Se re-aprovechan los preexistentes para 8 bit, luego se crean nuevos opcodes para ADD16, SUB16, AND16, etc.

El opcode más alto ocupado en el assembly básico es el $011000_2 = 24_{10}$ y el máximo es $111111_2 = 63_{10}$, por lo que se tienen suficientes opcodes desocupados para agregar este duplicado.

Además, es necesario agregar las señales de control apropiadas para manejar 8 o 16 bits, dependiendo del opcode en uso.

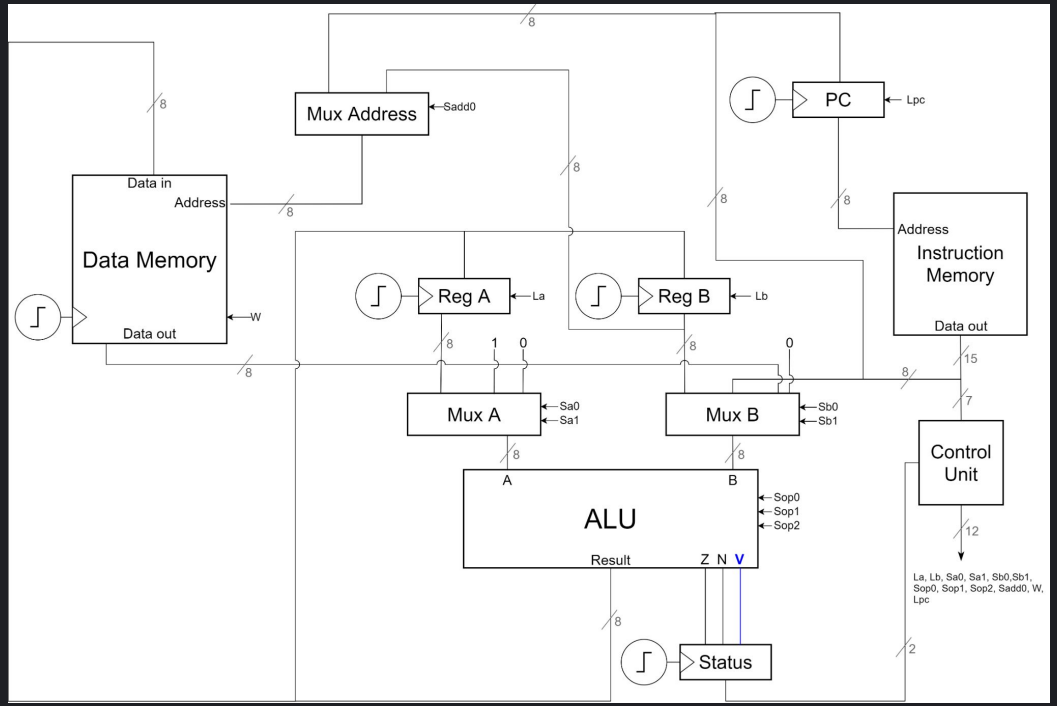
Pregunta 4

Si quisiéramos detectar automáticamente la ocurrencia de overflow en la ALU del computador básico, es posible agregar como output de esta, un condition code V (overflow) que tome el valor de 1 en caso de que ocurra overflow y 0 cuando no ocurre. Esta variable depende de otras cuatro variables binarias:

- A_n-1 (bit más significativo de A)
- B_n-1 (bit más significativo de B)
- S_n-1 (bit más significativo del resultado)
- Op (operación: 0: suma; 1:resta)

Escriba la tabla de verdad para el condition code V en base a estas cuatro variables.

Pregunta 4


Mux Address
Sadd0
PC
Lpc
Data in
Address
Data Memory
Data out
W
Reg A
La
Reg B
Lb
Address
Instruction Memory
Data out
Mux A
Sa0
Sa1
Mux B
Sb0
Sb1
Control Unit
A
B
ALU
Result
Z N V
Sop0
Sop1
Sop2
La, Lb, Sa0, Sa1, Sb0,Sb1, Sop0, Sop1, Sop2, Sadd0, W, Lpc
Status

Pregunta 4

Para poder escribir la tabla de verdad de V, tenemos que encontrar para qué combinación de valores de A_n-1, B_n-1, S_n-1 y Op ocurre overflow.

Para la suma **(Op = 0)** tenemos que:

- Si los dos números son de signo opuesto (A_n-1 != B_n-1), no puede ocurrir overflow.
- Si los dos números son positivos **(A_n-1 = 0, B_n-1 = 0)**, existe overflow cuando el resultado es negativo **(S_n-1 = 1)**.
- Si los dos números son negativos **(A_n-1 = 1, B_n-1 = 1)**, existe overflow cuando el resultado es positivo **(S_n-1 = 0)**.

Pregunta 4

Para la resta **(Op = 1)** tenemos que:

- Si los dos números son de signo opuesto (A_n-1 != B_n-1), no puede ocurrir overflow.
- Si A es positivo **(A_n-1 = 0)** y B es negativo **(B_n-1 = 1)**, ocurre overflow cuando el resultado es negativo **(S_n-1 = 1)**.
- Si A es negativo **(A_n-1 = 1)** y B es positivo **(B_n-1 = 0)**, ocurre overflow cuando el resultado es positivo **(S_n-1 = 0)**.

Pregunta 4

Entonces, solo ocurre overflow en los siguientes casos:

- En una suma **(Op = 0)**, si los dos números son positivos **(A_n-1 = 0, B_n-1 = 0)**, existe overflow cuando el resultado es negativo **(S_n-1 = 1)**.
- En una suma **(Op = 0)**, si los dos números son negativos **(A_n-1 = 1, B_n-1 = 1)**, existe overflow cuando el resultado es positivo (S_n-1 = 0).
- En una resta **(Op = 1)**, si A es positivo **(A_n-1 = 0)** y B es negativo **(B_n-1 = 1)**, ocurre overflow cuando el resultado es negativo **(S_n-1 = 1)**.
- En una resta **(Op = 1)**, si es negativo **(A_n-1 = 1)** y B es positivo **(B_n-1 = 0)**, ocurre overflow cuando el resultado es positivo **(S_n-1 = 0)**.

Pregunta 4

Op	A_{n-1}	B_{n-1}	S_{n-1}	overflow
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

GL; HF!

q(≧▽≦q)

003-1040559 1250 003-77156.8 1760 0009-14563.7 73273