Clase 12 - Programabilidad de circuitos digitales

IIC1001 - Algoritmos y Sistemas Computacionales

Cristian Ruz – cruz@uc.cl

Miércoles 17-Abril-2024

Departamento de Ciencia de la Computación
Escuela de Ingeniería
Pontificia Universidad Católica de Chile

Contenidos

Contenidos

Contacto

🥇 ignaciomunoz@uc.cl
Ignacio Muñoz
Ayudante jefe

- Coordinación
- Notas de actividades, interrogaciones
- Todo lo que no sé donde más enviar

🐐 vicente.cabra@uc.cl
Vicente Cabra
Ayudante

- Materia

🐚 fernando.concha@uc.cl
Fernando Concha
Ayudante

- Materia

🐸 alejandro.tapia@uc.cl
Alejandro Tapia
Ayudante

- Materia

Contenidos

Temas del curso

Sistemas computacionales

- Representación datos, números y codificación
- Funcionamiento hardware, procesadores y memoria.
- Funcionamiento de sistemas operativos: ejemplo scheduling
- Funcionamiento de Internet
- Herramientas computacionales: github + latex

Algoritmos

- Algoritmos y resolución de problemas
- Eficiencia algorítmica
- Estructuras secuenciales y ordenamiento
- Grafos y árboles

Esta semana

Lunes 15-Abril: Clase

- Multiplexores
- Unidad de control

Miércoles 17-Abril: Clase + Actividad

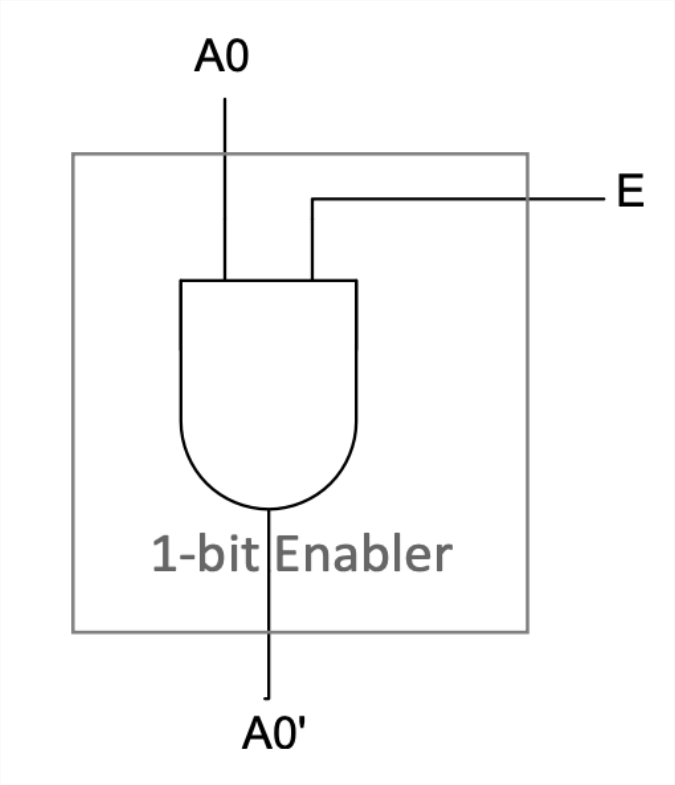
- Repaso de multiplexores y unidad de control
- Actividad en grupos de 3

Contenidos

Enabler

Componente que permite habilitar o deshabilitar una salida

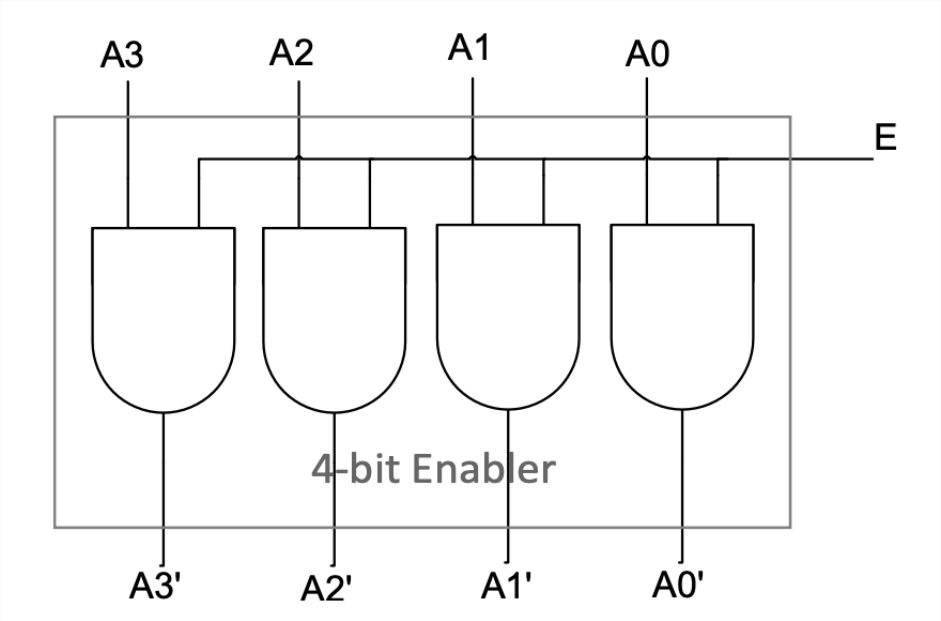
E	A0’
0	0
1	A0



Enabler de 1 bit

Enabler

Se puede implementar para circuitos con más bit de entrada



Enabler de 4 bits

Multiplexor

Buscamos un circuito que tenga más de una entrada, y nos permita **elegir** cuál de ellas queremos leer.

Dos entradas: $A0$, y $B0$.

Una salida: $M0$.

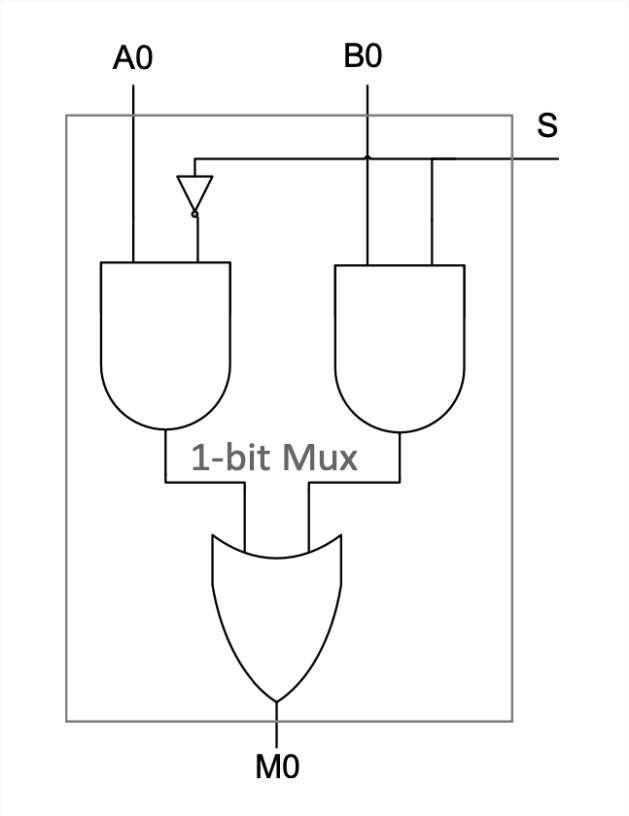
Señal de elección: S. Si $S = 0$, queremos leer $A0$. Si $S = 1$, queremos leer $B0$.

S	M0
0	A0
1	B0

Multiplexor

Señal de elección: S. Si $S = 0$, queremos leer $A0$. Si $S = 1$, queremos leer $B0$.

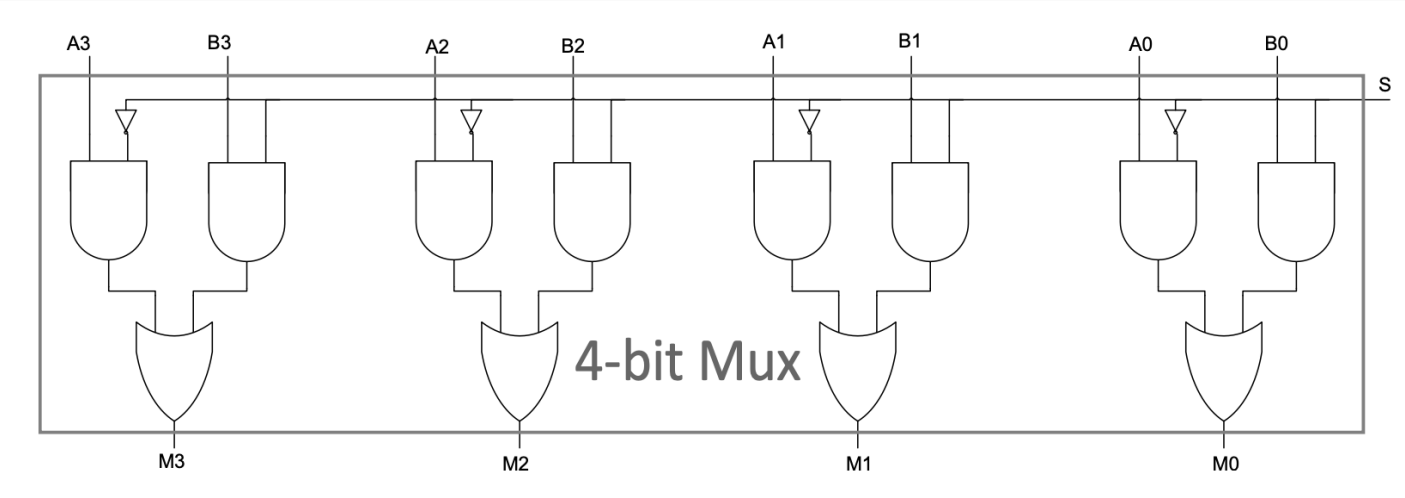
S	M0
0	A0
1	B0



Multiplexor de 2 entradas de 1 bit de datos

Multiplexor

Se puede implementar con más bits

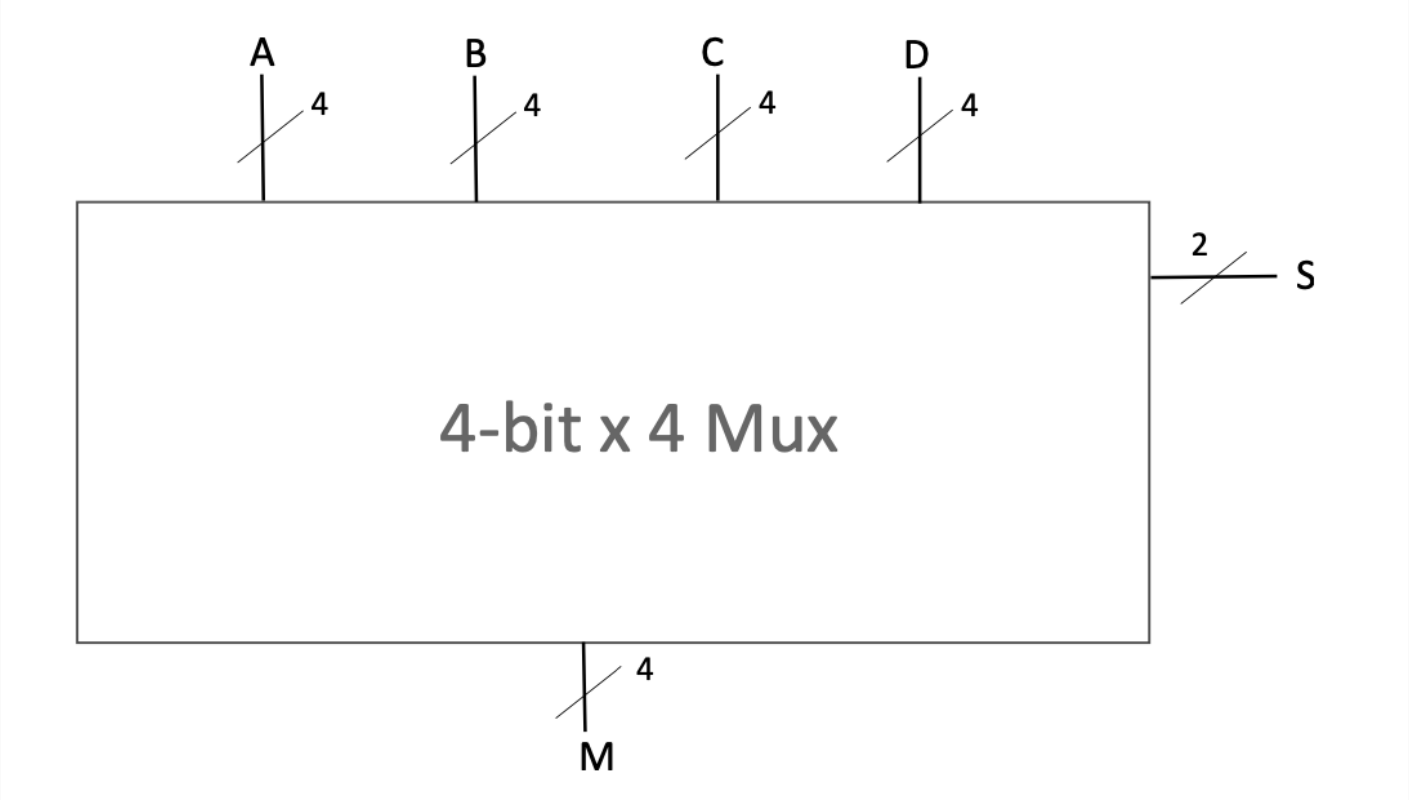


S1	S1	M
0	0	A
0	1	B
1	0	C
1	1	D

Multiplexor de 2 entradas de 4 bit de datos

Multiplexor

En notación de buses



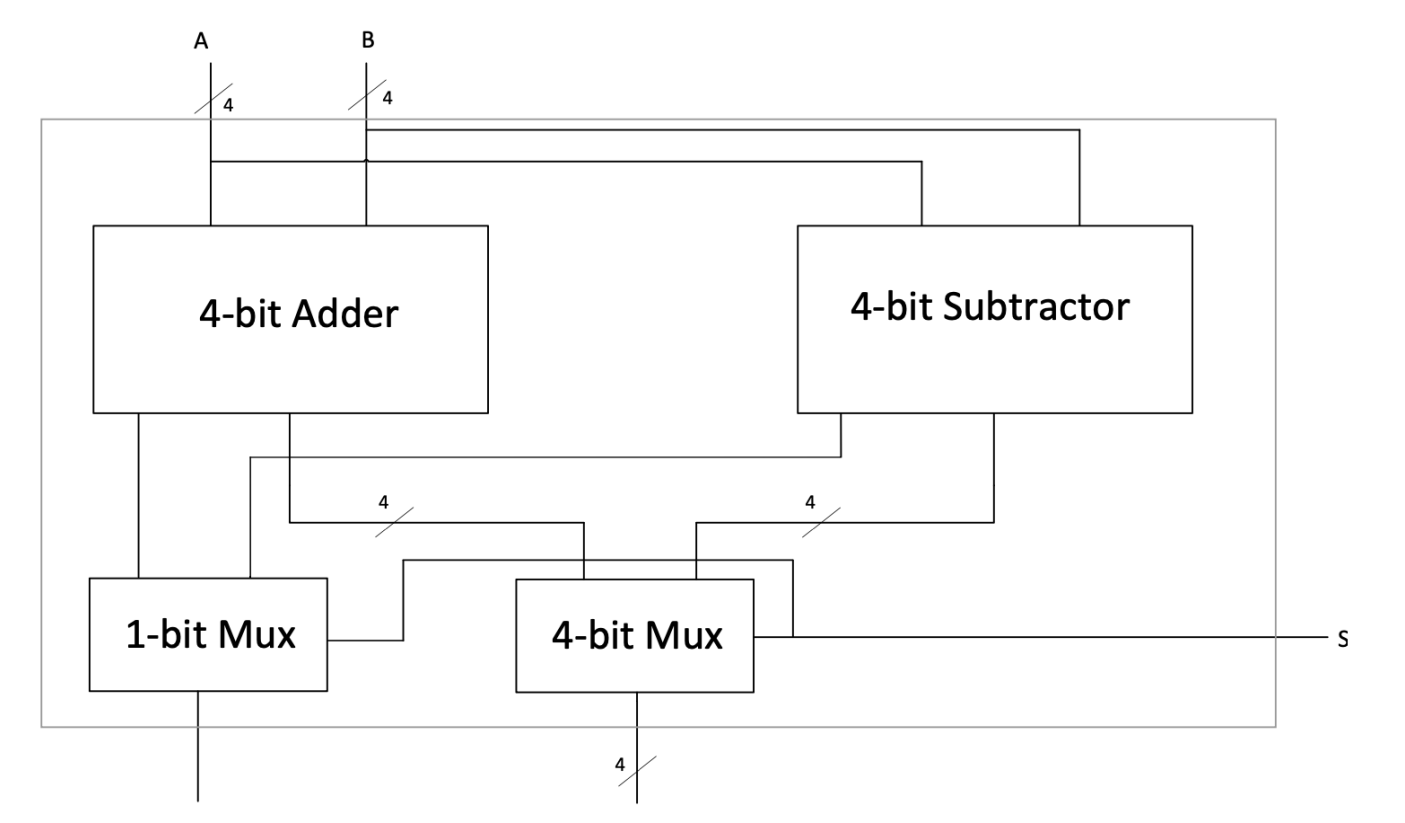
Multiplexor de 4 entradas de 4 bit de datos

Unidad de selección

Se pueden combinar operaciones y seleccionar usando multiplexores.

Es una primera aproximación a una **unidad aritmético-lógica** (ALU)

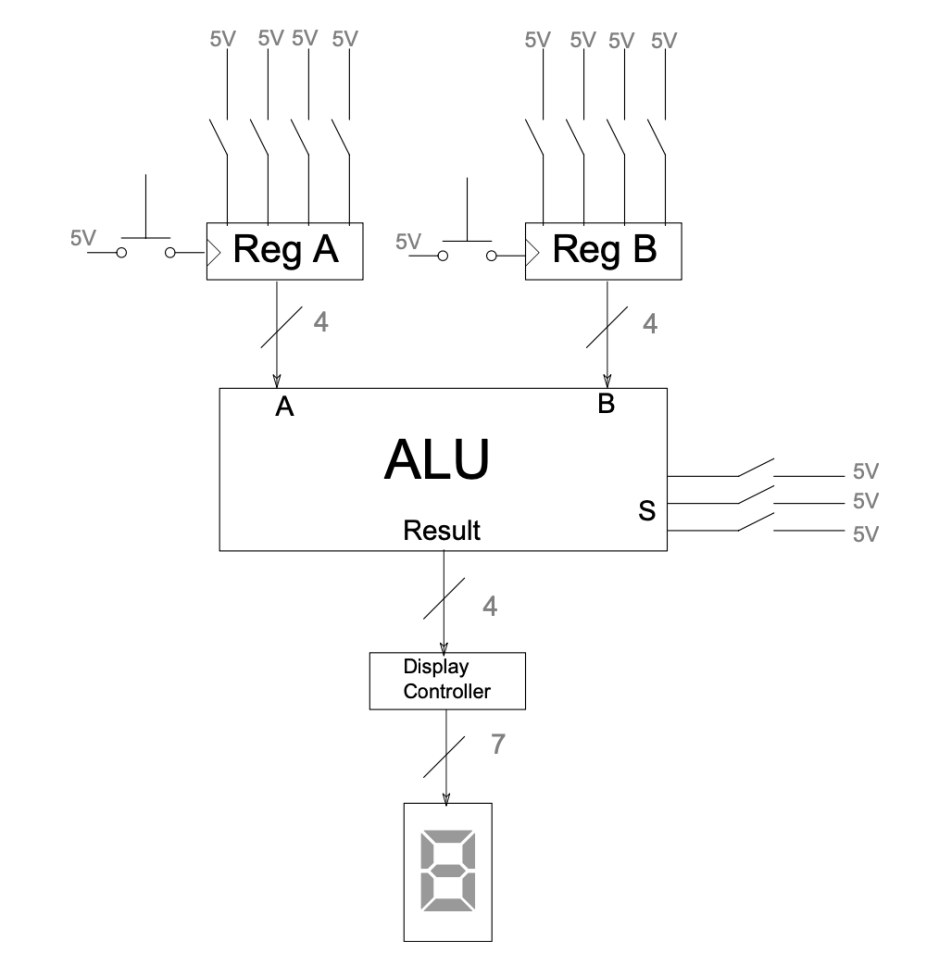
$S = 0$: Suma, $S = 1$:, Resta



Sumador-restador de 4 bit de datos

Contenidos

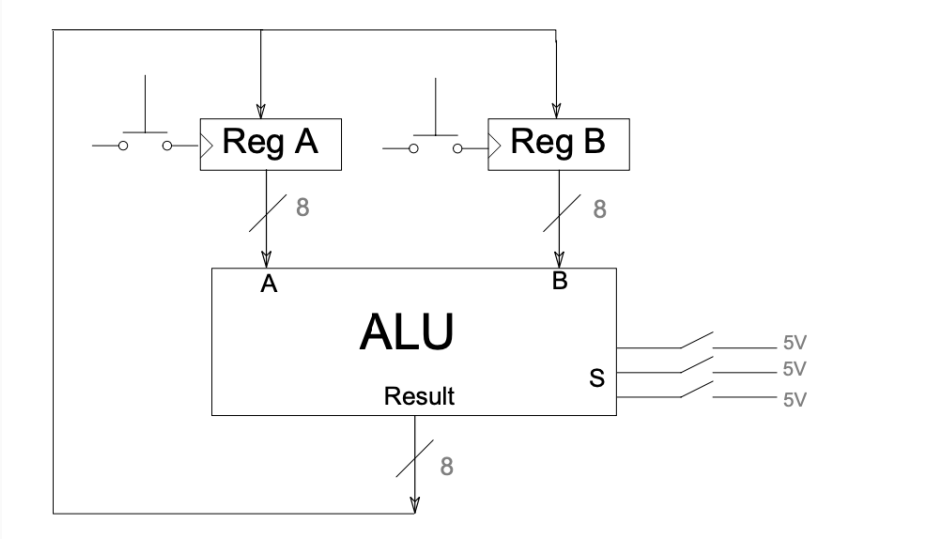
Acumulación de operaciones



select	s2	s1	s0	operación
0	0	0	0	Suma
1	0	0	1	Resta
2	0	1	0	And
3	0	1	1	Or
4	1	0	0	Not A
5	1	0	1	Xor
6	1	1	0	Shift Left A
7	1	1	1	Shift Right A

Calculador de 4 bits con ingreso de usuario y display

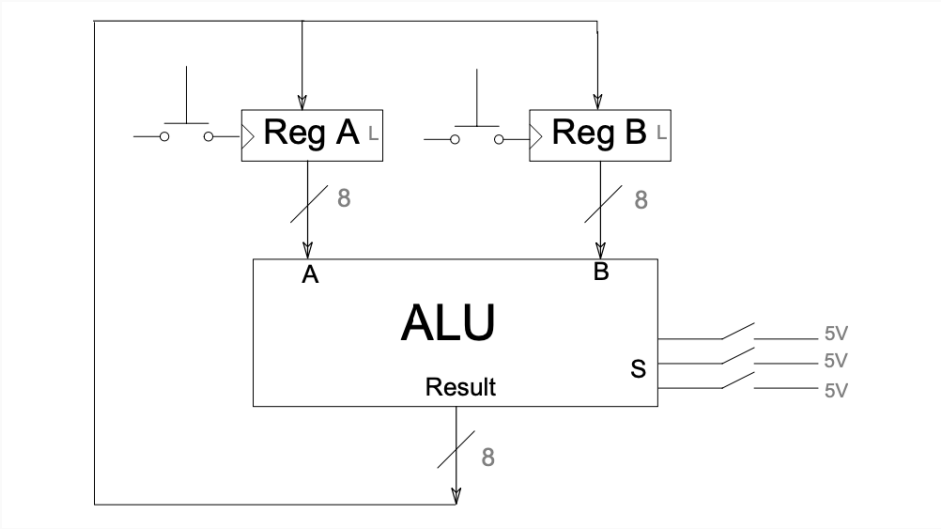
Acumulación de operaciones



select	s2	s1	s0	operación
0	0	0	0	Suma
1	0	0	1	Resta
2	0	1	0	And
3	0	1	1	Or
4	1	0	0	Not A
5	1	0	1	Xor
6	1	1	0	Shift Left A
7	1	1	1	Shift Right A

Calculadora de 4 bits con **registros acumuladores**

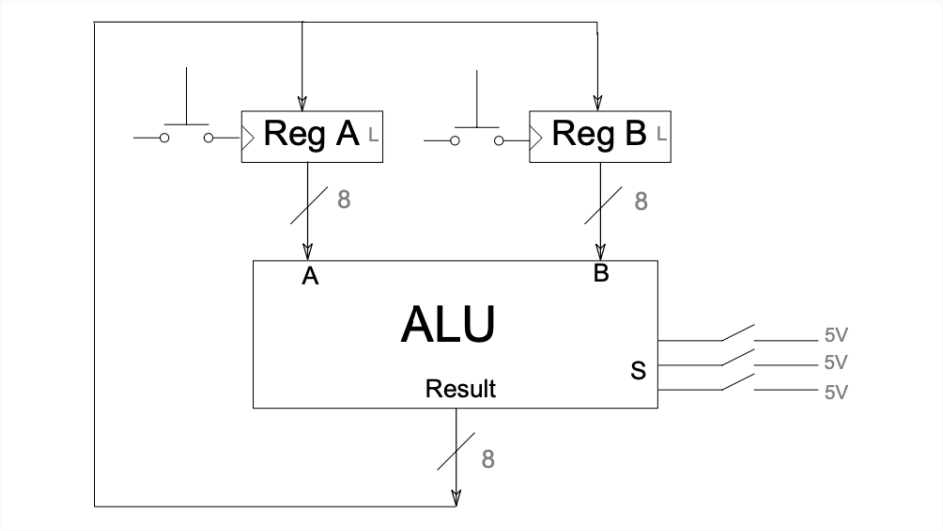
Acumulación de operaciones



la	lb	s2	s1	s0	operación
1	0	0	0	0	A=A+B
0	1	0	0	0	B=A+B
1	0	0	0	1	A=A-B
0	1	0	0	1	B=A-B
1	0	0	1	0	A=A and B
0	1	0	1	0	B=A and B
1	0	0	1	1	A=A or B
0	1	0	1	1	B=A or B
1	0	1	0	0	A=notA
0	1	1	0	0	B=notA
1	0	1	0	1	A=A xor B
0	1	1	0	1	B=A xor B
1	0	1	1	0	A=shift left A
0	1	1	1	0	B=shift left A
1	0	1	1	1	A=shift right A
0	1	1	1	1	B=shift right A

Calculadora de 4 bits con **registros acumuladores** y señales de carga

Acumulación de operaciones

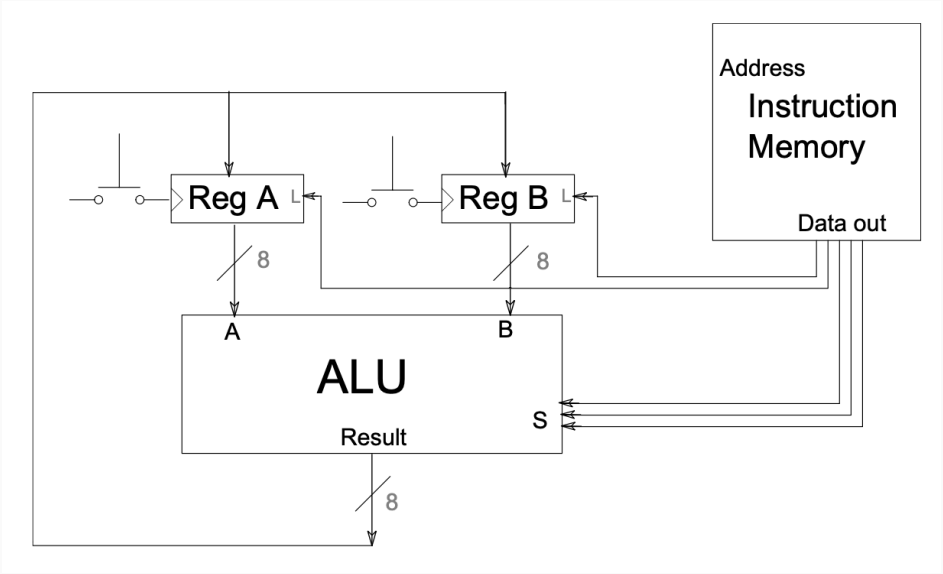


la	lb	s2	s1	s0	operación
1	0	0	0	0	A=A+B
0	1	0	0	0	B=A+B
1	0	0	0	1	A=A-B
0	1	0	0	1	B=A-B
1	0	0	1	0	A=A and B
0	1	0	1	0	B=A and B
1	0	0	1	1	A=A or B
0	1	0	1	1	B=A or B
1	0	1	0	0	A=notA
0	1	1	0	0	B=notA
1	0	1	0	1	A=A xor B
0	1	1	0	1	B=A xor B
1	0	1	1	0	A=shift left A
0	1	1	1	0	B=shift left A
1	0	1	1	1	A=shift right A
0	1	1	1	1	B=shift right A

la	lb	s2	s1	s0	operación	A	B
0	0	-	-	-	-	**0**	1
1	0	0	0	0	A=A+B	**1**	1
0	1	0	0	0	B=A+B	1	**2**
1	0	0	0	0	A=A+B	**3**	2
0	1	0	0	0	B=A+B	3	**5**
1	0	0	0	0	A=A+B	**8**	5
0	1	0	0	0	B=A+B	8	**13**

Calculadora de 4 bits con **registros acumuladores**, señales de carga y programa

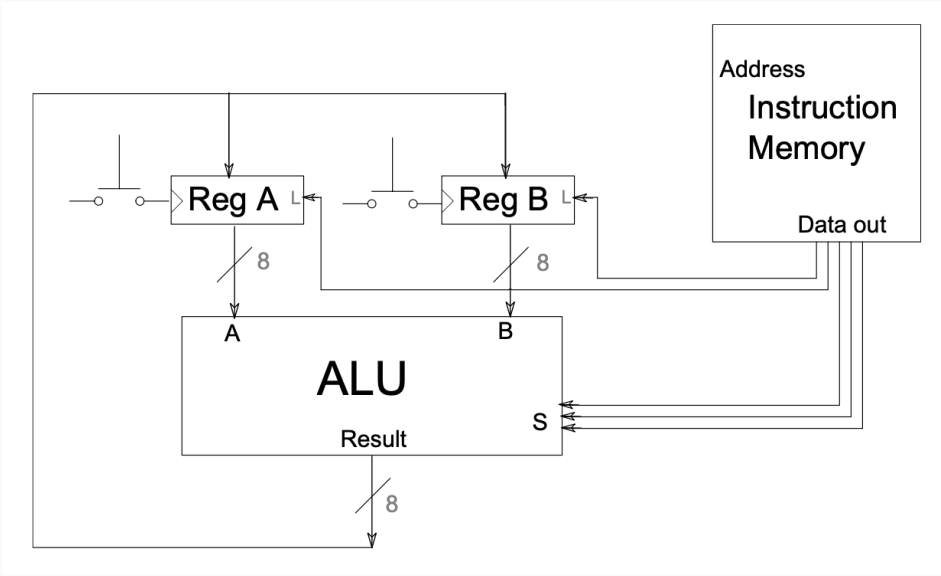
Almacenamiento de instrucciones



la	lb	s2	s1	s0	operación
1	0	0	0	0	A=A+B
0	1	0	0	0	B=A+B
1	0	0	0	1	A=A-B
0	1	0	0	1	B=A-B
1	0	0	1	0	A=A and B
0	1	0	1	0	B=A and B
1	0	0	1	1	A=A or B
0	1	0	1	1	B=A or B
1	0	1	0	0	A=notA
0	1	1	0	0	B=notA
1	0	1	0	1	A=A xor B
0	1	1	0	1	B=A xor B
1	0	1	1	0	A=shift left A
0	1	1	1	0	B=shift left A
1	0	1	1	1	A=shift right A
0	1	1	1	1	B=shift right A

Usando una memoria de instrucciones se puede almacenar un programa

Almacenamiento de instrucciones



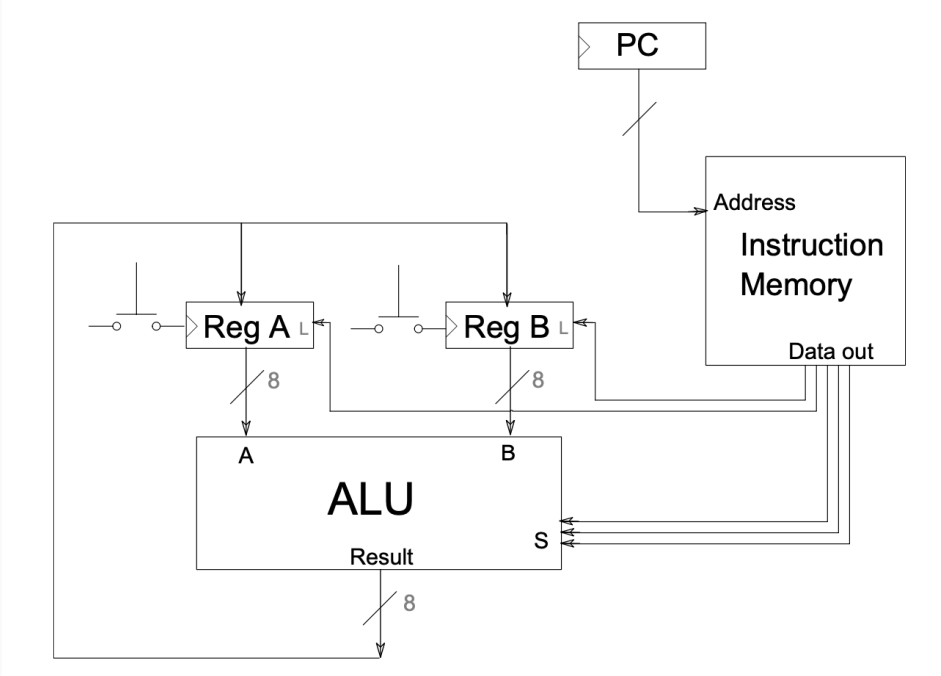
la	lb	s2	s1	s0	operación
1	0	0	0	0	A=A+B
0	1	0	0	0	B=A+B
1	0	0	0	1	A=A-B
0	1	0	0	1	B=A-B
1	0	0	1	0	A=A and B
0	1	0	1	0	B=A and B
1	0	0	1	1	A=A or B
0	1	0	1	1	B=A or B
1	0	1	0	0	A=notA
0	1	1	0	0	B=notA
1	0	1	0	1	A=A xor B
0	1	1	0	1	B=A xor B
1	0	1	1	0	A=shift left A
0	1	1	1	0	B=shift left A
1	0	1	1	1	A=shift right A
0	1	1	1	1	B=shift right A

la	lb	s2	s1	s0	operación	A	B
0	0	-	-	-	-	**0**	1
1	0	0	0	0	A=A+B	**1**	1
0	1	0	0	0	B=A+B	1	**2**
1	0	0	0	0	A=A+B	**3**	2
0	1	0	0	0	B=A+B	3	**5**
1	0	0	0	0	A=A+B	**8**	5
0	1	0	0	0	B=A+B	8	**13**

dirección	instrucción
0000	10000
0001	01000
0010	10000
0011	01000
0100	10000
0101	01000

Usando una memoria de instrucciones se puede almacenar un programa

Almacenamiento de instrucciones



la	lb	s2	s1	s0	operación
1	0	0	0	0	A=A+B
0	1	0	0	0	B=A+B
1	0	0	0	1	A=A-B
0	1	0	0	1	B=A-B
1	0	0	1	0	A=A and B
0	1	0	1	0	B=A and B
1	0	0	1	1	A=A or B
0	1	0	1	1	B=A or B
1	0	1	0	0	A=notA
0	1	1	0	0	B=notA
1	0	1	0	1	A=A xor B
0	1	1	0	1	B=A xor B
1	0	1	1	0	A=shift left A
0	1	1	1	0	B=shift left A
1	0	1	1	1	A=shift right A
0	1	1	1	1	B=shift right A

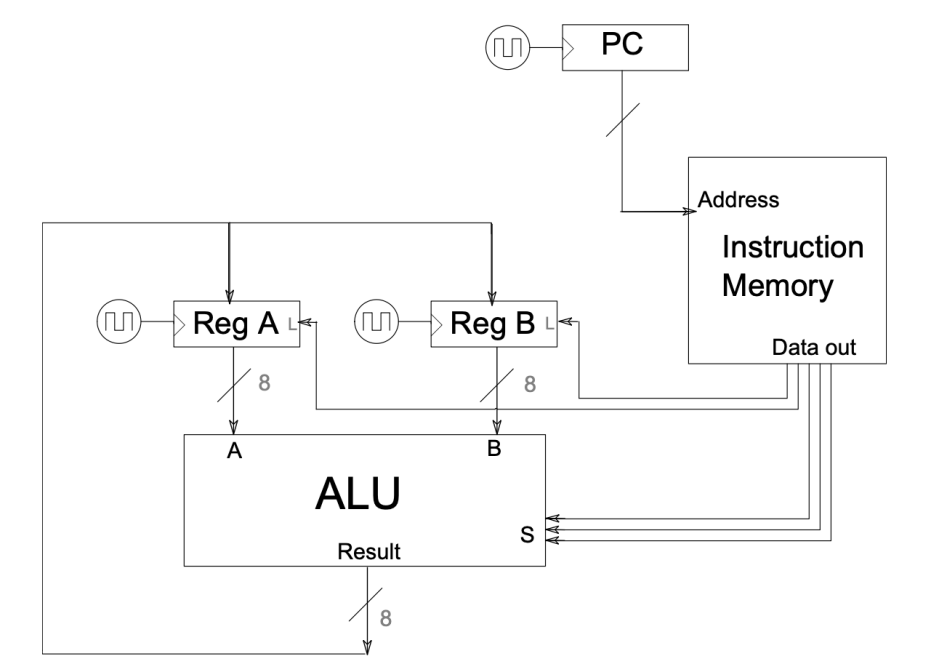
program counter	instrucción	operación
0000	10000	A=A+B
0001	01000	B=A+B
0010	10000	A=A+B
0011	01000	B=A+B
0100	10000	A=A+B
0101	01000	B=A+B

dirección	instrucción
0000	10000
0001	01000
0010	10000
0011	01000
0100	10000
0101	01000

Registro **Programa Counter (PC)** indica en qué instrucción vamos a ejecutar

Contenidos

Ejecución automática



la	lb	s2	s1	s0	operación
1	0	0	0	0	A=A+B
0	1	0	0	0	B=A+B
1	0	0	0	1	A=A-B
0	1	0	0	1	B=A-B
1	0	0	1	0	A=A and B
0	1	0	1	0	B=A and B
1	0	0	1	1	A=A or B
0	1	0	1	1	B=A or B
1	0	1	0	0	A=notA
0	1	1	0	0	B=notA
1	0	1	0	1	A=A xor B
0	1	1	0	1	B=A xor B
1	0	1	1	0	A=shift left A
0	1	1	1	0	B=shift left A
1	0	1	1	1	A=shift right A
0	1	1	1	1	B=shift right A

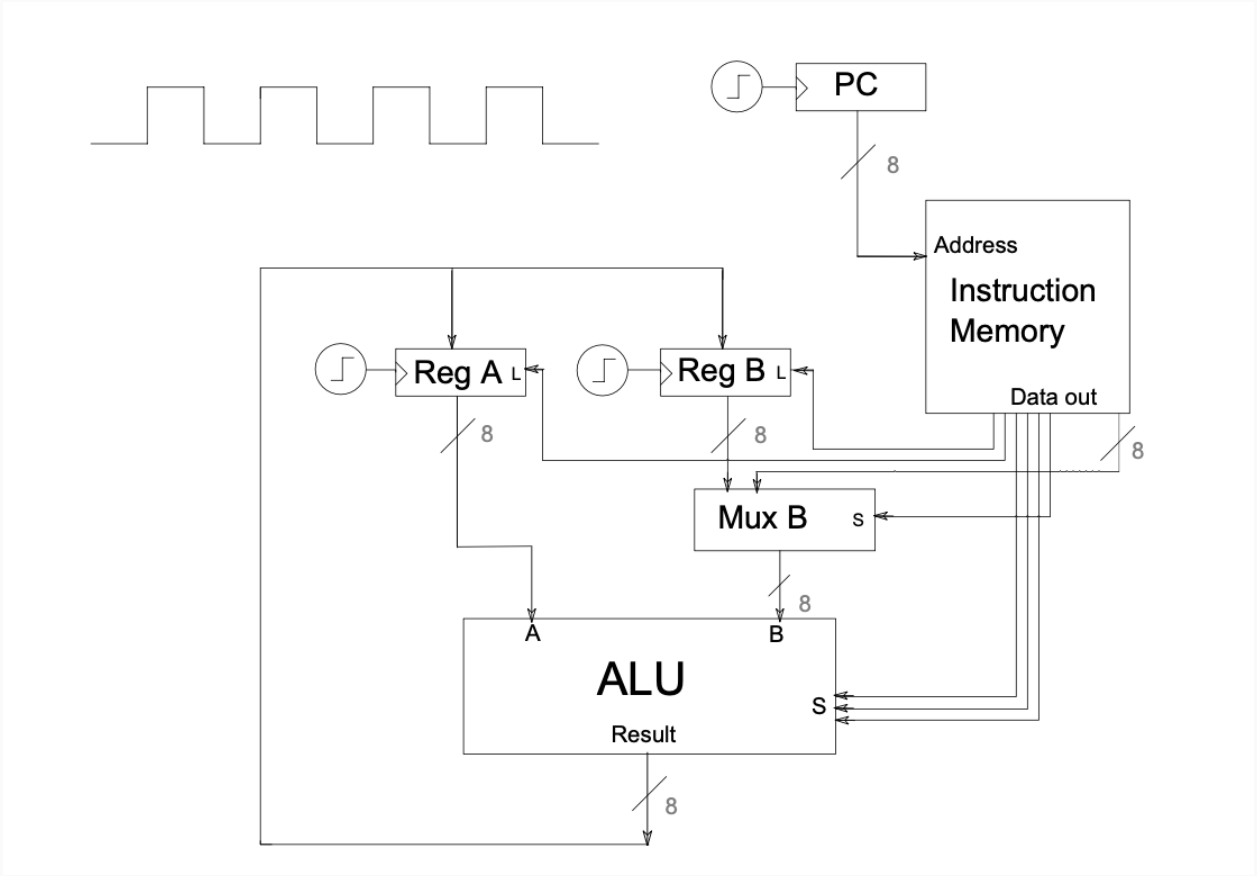
program counter	instrucción	operación
0000	10000	A=A+B
0001	01000	B=A+B
0010	10000	A=A+B
0011	01000	B=A+B
0100	10000	A=A+B
0101	01000	B=A+B

dirección	instrucción
0000	10000
0001	01000
0010	10000
0011	01000
0100	10000
0101	01000

Usando una memoria de instrucciones se puede almacenar un programa

Contenidos

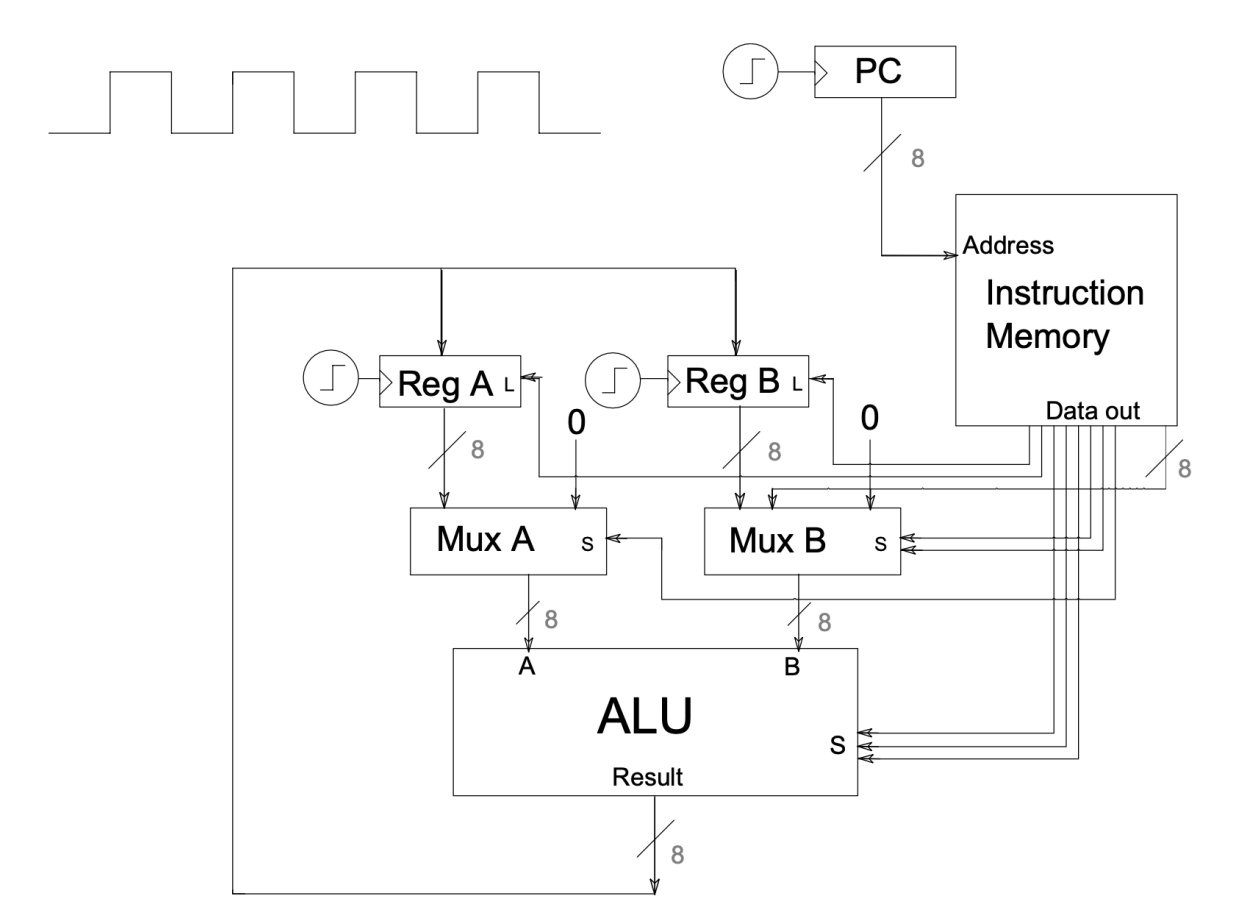
Operaciones con literales



Computador simple con carga de literales

Operaciones: $A = A + literal$

Carga de literales



Computador simple con carga de literales

Operaciones: $A = A + 0$

Contenidos

Unidad control

La	Lb	Sa0	Sb0	Sb1	Sop2	Sop1	Sop0	Operación
1	0	1	0	0	0	0	0	A=B
0	1	0	1	1	0	0	0	B=A
1	0	1	0	1	0	0	0	A=Lit
0	1	1	0	1	0	0	0	B=Lit
1	0	0	0	0	0	0	0	A=A+B
0	1	0	0	0	0	0	0	B=A+B
1	0	0	0	1	0	0	0	A=A+Lit
1	0	0	0	0	0	0	1	A=A-B
0	1	0	0	0	0	0	1	B=A-B
1	0	0	0	1	0	0	1	A=A-Lit
1	0	0	0	0	0	1	0	A=A and B
0	1	0	0	0	0	1	0	B=A and B
1	0	0	0	1	0	1	0	A=A and Lit
1	0	0	0	0	0	1	1	A=A or B
0	1	0	0	0	0	1	1	B=A or B
1	0	0	0	1	0	1	1	A=A or Lit
1	0	0	0	0	1	0	0	A=notA
0	1	0	0	0	1	0	0	B=notA
1	0	0	0	1	1	0	0	A=notLit
1	0	0	0	0	1	0	1	A=A xor B
0	1	0	0	0	1	0	1	B=A xor B
1	0	0	0	1	1	0	1	A=A xor Lit
1	0	0	0	0	1	1	0	A=shift left A
0	1	0	0	0	1	1	0	B=shift left A
1	0	0	0	0	1	1	1	A=shift right A
0	1	0	0	0	1	1	1	B=shift right A

Unidad de control contiene instrucciones posible y señales de control

Unidad control

Opcode	La	Lb	Sa0	Sb0	Sb1	Sop2	Sop1	Sop0	Operación
000000	1	0	1	0	0	0	0	0	A=B
000001	0	1	0	1	1	0	0	0	B=A
000010	1	0	1	0	1	0	0	0	A=Lit
000011	0	1	1	0	1	0	0	0	B=Lit
000100	1	0	0	0	0	0	0	0	A=A+B
000101	0	1	0	0	0	0	0	0	B=A+B
000110	1	0	0	0	1	0	0	0	A=A+Lit
000111	1	0	0	0	0	0	0	1	A=A-B
001000	0	1	0	0	0	0	0	1	B=A-B
001001	1	0	0	0	1	0	0	1	A=A-Lit
001010	1	0	0	0	0	0	1	0	A=A and B
001011	0	1	0	0	0	0	1	0	B=A and B
001100	1	0	0	0	1	0	1	0	A=A and Lit
001101	1	0	0	0	0	0	1	1	A=A or B
001110	0	1	0	0	0	0	1	1	B=A or B
001111	1	0	0	0	1	0	1	1	A=A or Lit
010000	1	0	0	0	0	1	0	0	A=notA
010001	0	1	0	0	0	1	0	0	B=notA
010010	1	0	0	0	0	1	0	1	A=A xor B
010011	0	1	0	0	0	1	0	1	B=A xor B
010100	1	0	0	0	1	1	0	1	A=A xor Lit
010101	1	0	0	0	0	1	1	0	A=shift left A
010110	0	1	0	0	0	1	1	0	B=shift left A
010111	1	0	0	0	0	1	1	1	A=shift right A
011000	0	1	0	0	0	1	1	1	B=shift right A

OPCODE permite identificar instrucciones

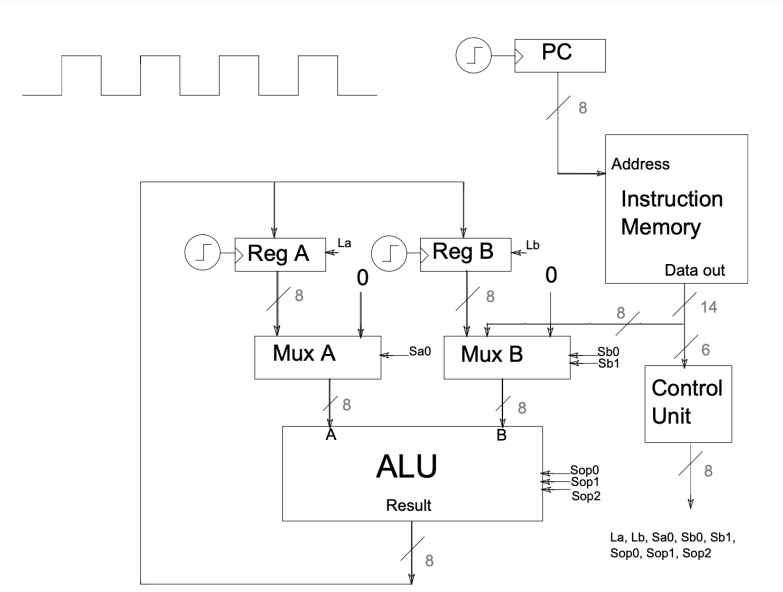
Unidad control

Instrucción	Opcode	La	Lb	Sa0	Sb0	Sb1	Sop2	Sop1	Sop0	Operación
MOVAB	000000	1	0	1	0	0	0	0	0	A=B
MOVBA	000001	0	1	0	1	1	0	0	0	B=A
MOVAL	000010	1	0	1	0	1	0	0	0	A=Lit
MOVBL	000011	0	1	1	0	1	0	0	0	B=Lit
ADDA	000100	1	0	0	0	0	0	0	0	A=A+B
ADDB	000101	0	1	0	0	0	0	0	0	B=A+B
ADDL	000110	1	0	0	0	1	0	0	0	A=A+Lit
SUBA	000111	1	0	0	0	0	0	0	1	A=A-B
SUBB	001000	0	1	0	0	0	0	0	1	B=A-B
SUBL	001001	1	0	0	0	1	0	0	1	A=A-Lit
ANDA	001010	1	0	0	0	0	0	1	0	A=A and B
ANDB	001011	0	1	0	0	0	0	1	0	B=A and B
ANDL	001100	1	0	0	0	1	0	1	0	A=A and Lit
ORA	001101	1	0	0	0	0	0	1	1	A=A or B
ORB	001110	0	1	0	0	0	0	1	1	B=A or B
ORL	001111	1	0	0	0	1	0	1	1	A=A or Lit
NOTA	010000	1	0	0	0	0	1	0	0	A=notA
NOTB	010001	0	1	0	0	0	1	0	0	B=notA
XORA	010010	1	0	0	0	0	1	0	1	A=A xor B
XORB	010011	0	1	0	0	0	1	0	1	B=A xor B
XORL	010100	1	0	0	0	1	1	0	1	A=A xor Lit
SHLA	010101	1	0	0	0	0	1	1	0	A=shift left A
SHLB	010110	0	1	0	0	0	1	1	0	B=shift left A
SHRA	010111	1	0	0	0	0	1	1	1	A=shift right A
SHRB	011000	0	1	0	0	0	1	1	1	B=shift right A

Instrucciones pueden tener un nombre simbólico (assembler)

Unidad control

Instrucción	Opcode	La	Lb	Sa0	Sb0	Sb1	Sop2	Sop1	Sop0	Operación
MOVAB	000000	1	0	1	0	0	0	0	0	A=B
MOVBA	000001	0	1	0	1	1	0	0	0	B=A
MOVAL	000010	1	0	1	0	1	0	0	0	A=Lit
MOVBL	000011	0	1	1	0	1	0	0	0	B=Lit
ADDA	000100	1	0	0	0	0	0	0	0	A=A+B
ADDB	000101	0	1	0	0	0	0	0	0	B=A+B
ADDL	000110	1	0	0	0	1	0	0	0	A=A+Lit
SUBA	000111	1	0	0	0	0	0	0	1	A=A-B
SUBB	001000	0	1	0	0	0	0	0	1	B=A-B
SUBL	001001	1	0	0	0	1	0	0	1	A=A-Lit
ANDA	001010	1	0	0	0	0	0	1	0	A=A and B
ANDB	001011	0	1	0	0	0	0	1	0	B=A and B
ANDL	001100	1	0	0	0	1	0	1	0	A=A and Lit
ORA	001101	1	0	0	0	0	0	1	1	A=A or B
ORB	001110	0	1	0	0	0	0	1	1	B=A or B
ORL	001111	1	0	0	0	1	0	1	1	A=A or Lit
NOTA	010000	1	0	0	0	0	1	0	0	A=notA
NOTB	010001	0	1	0	0	0	1	0	0	B=notA
XORA	010010	1	0	0	0	0	1	0	1	A=A xor B
XORB	010011	0	1	0	0	0	1	0	1	B=A xor B
XORL	010100	1	0	0	0	1	1	0	1	A=A xor Lit
SHLA	010101	1	0	0	0	0	1	1	0	A=shift left A
SHLB	010110	0	1	0	0	0	1	1	0	B=shift left A
SHRA	010111	1	0	0	0	0	1	1	1	A=shift right A
SHRB	011000	0	1	0	0	0	1	1	1	B=shift right A



Computador básico utiliza unidad de control