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

IIC2343 – Arquitectura de Computadores

Arquitecturas de Computadores

Profesor: Hans Löbel

Computador básico ya tiene todas las funcionalidades "básicas"

- Posee registros y unidades de ejecución y control.
- Además de hacer cálculos, puede realizar operaciones de control de flujo.
- Provee modularidad básica, al dar soporte para subrutinas.

Nuestro computador presenta una de muchas posibles arquitecturas

- Distintos computadores pueden diferir en el conjunto de funcionalidades básicas y fundamentales.
- Por otro lado, existen computadores que son programados de la misma manera (ej. AMD-Intel), pero su construcción interna es distinta.
- Decisiones en cuanto a cantidad de registros, tamaño de buses, memorias, instrucciones, etc., definen la arquitectura de un computador.

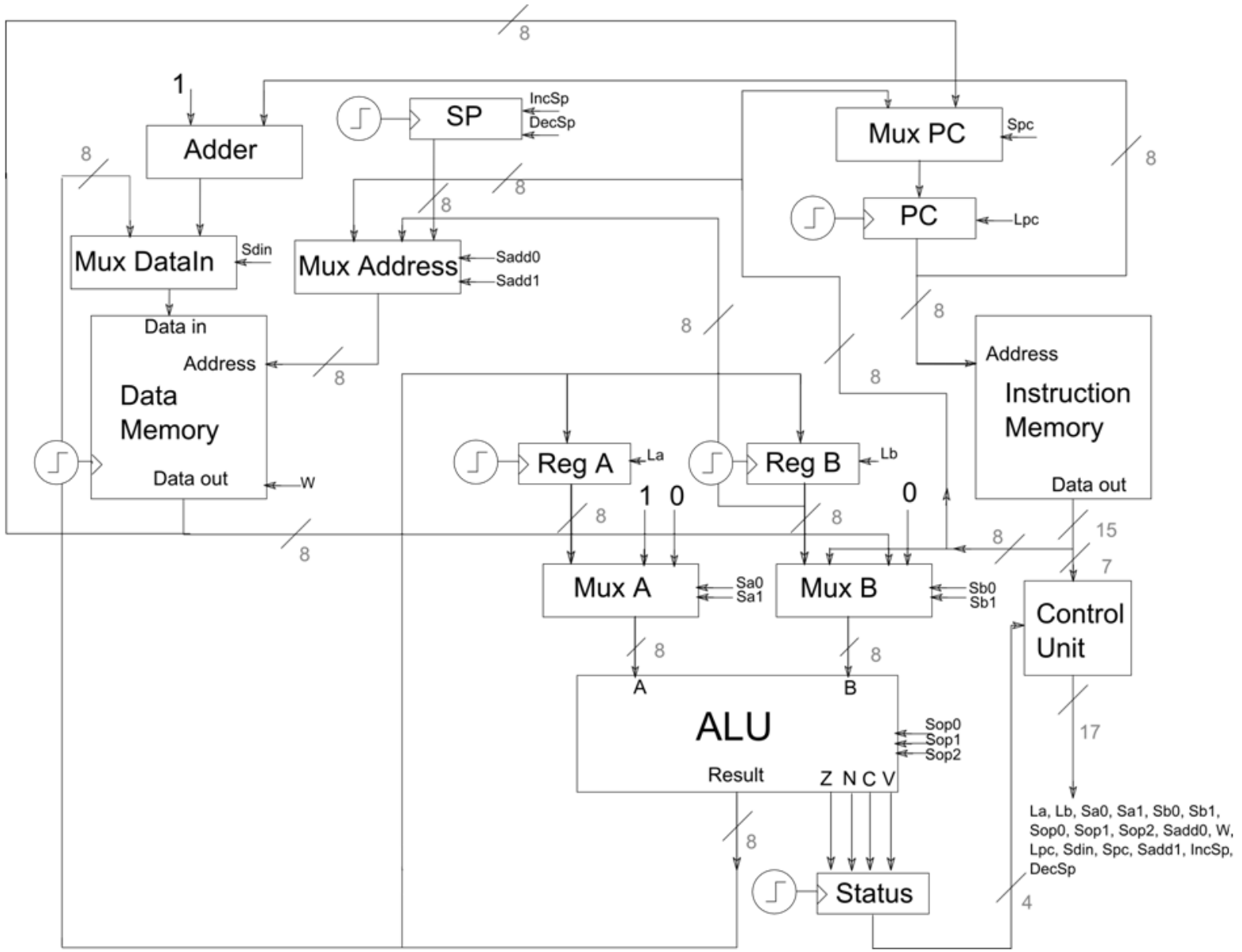
Microarquitectura e ISA definen la arquitectura de un computador

La arquitectura de un computador se define en base a dos elementos:

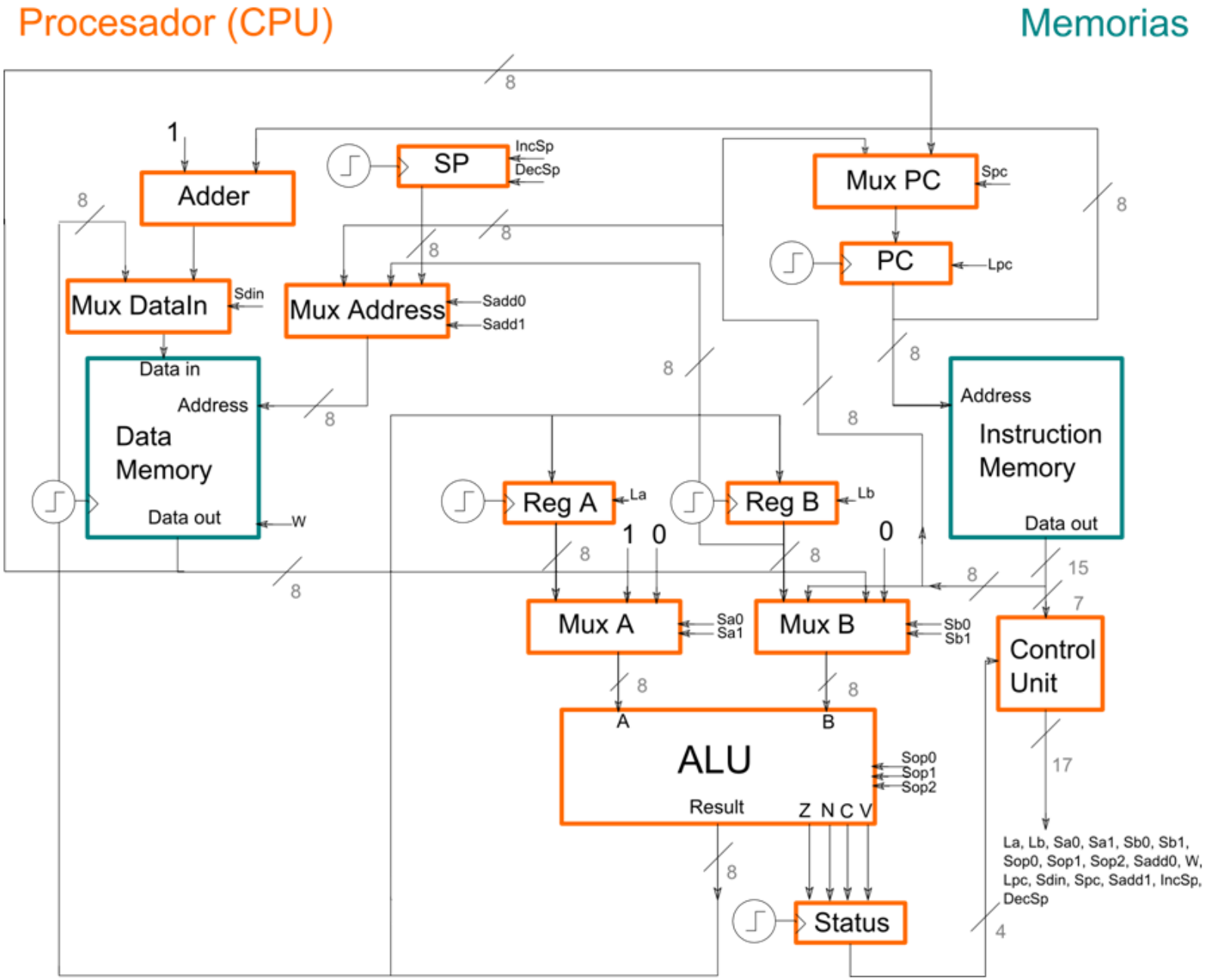
1. Microarquitectura: se refiere a los distintos componentes de hardware que están presentes en el computador.

2. Arquitectura del set de instrucciones (ISA): se refiere al tipo, formato, características, etc., de las instrucciones soportadas por el computador. En resumen, lo que tenga que ver con la programación de un computador.

Revisemos la microarquitectura de nuestro computador básico

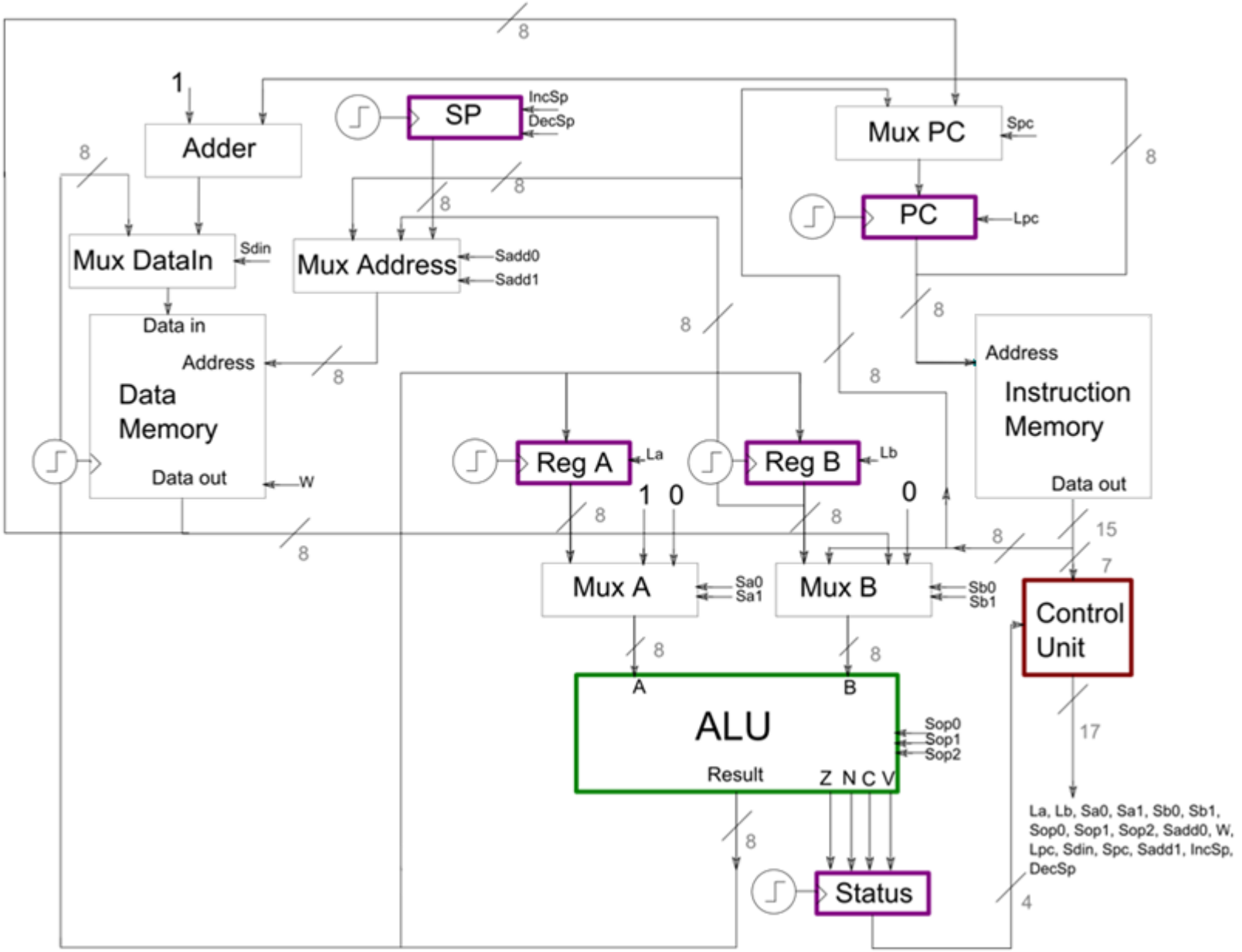


Revisemos la microarquitectura de nuestro computador básico



Revisemos la microarquitectura de nuestro computador básico

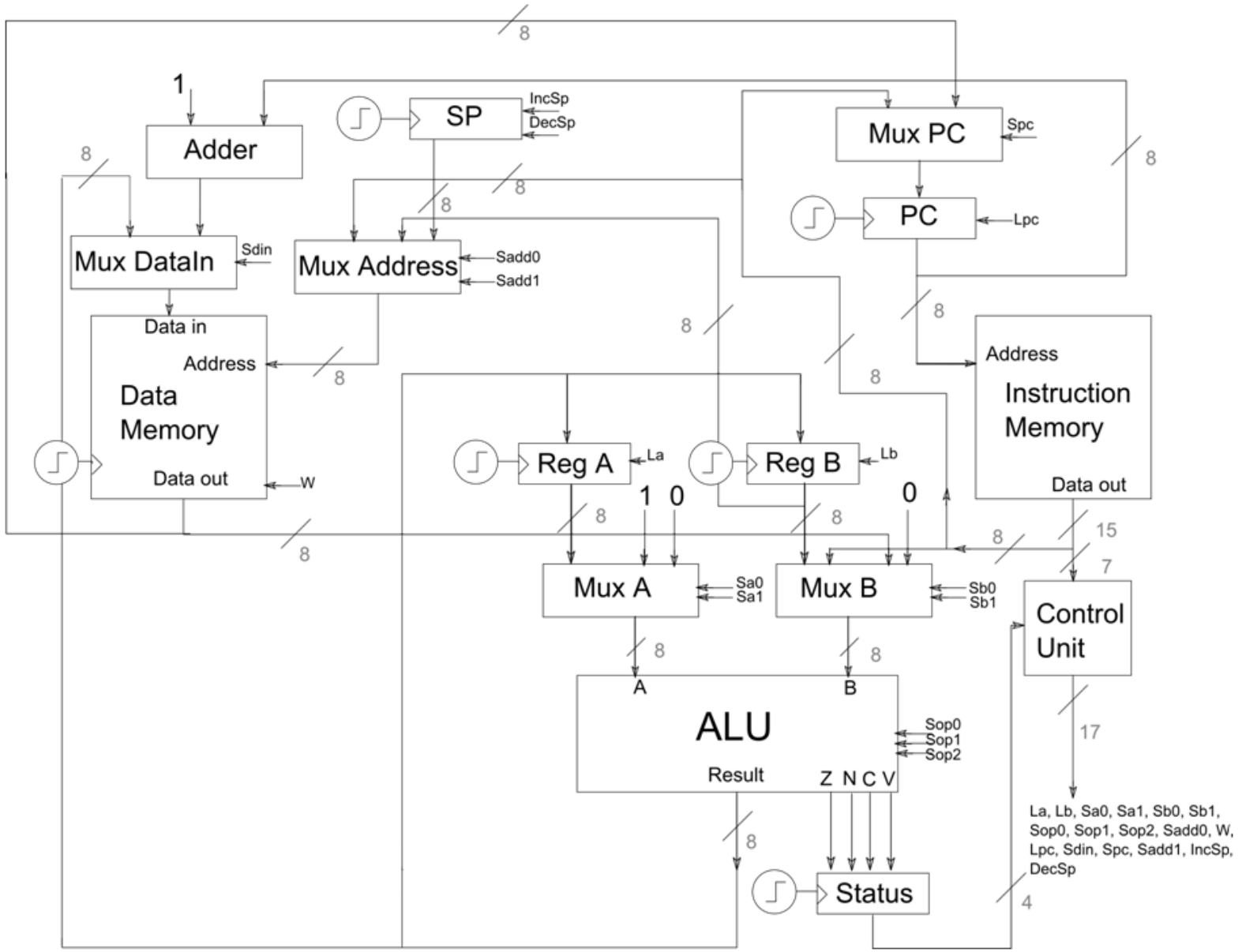
Registros, Unidad de ejecución, Unidad de control

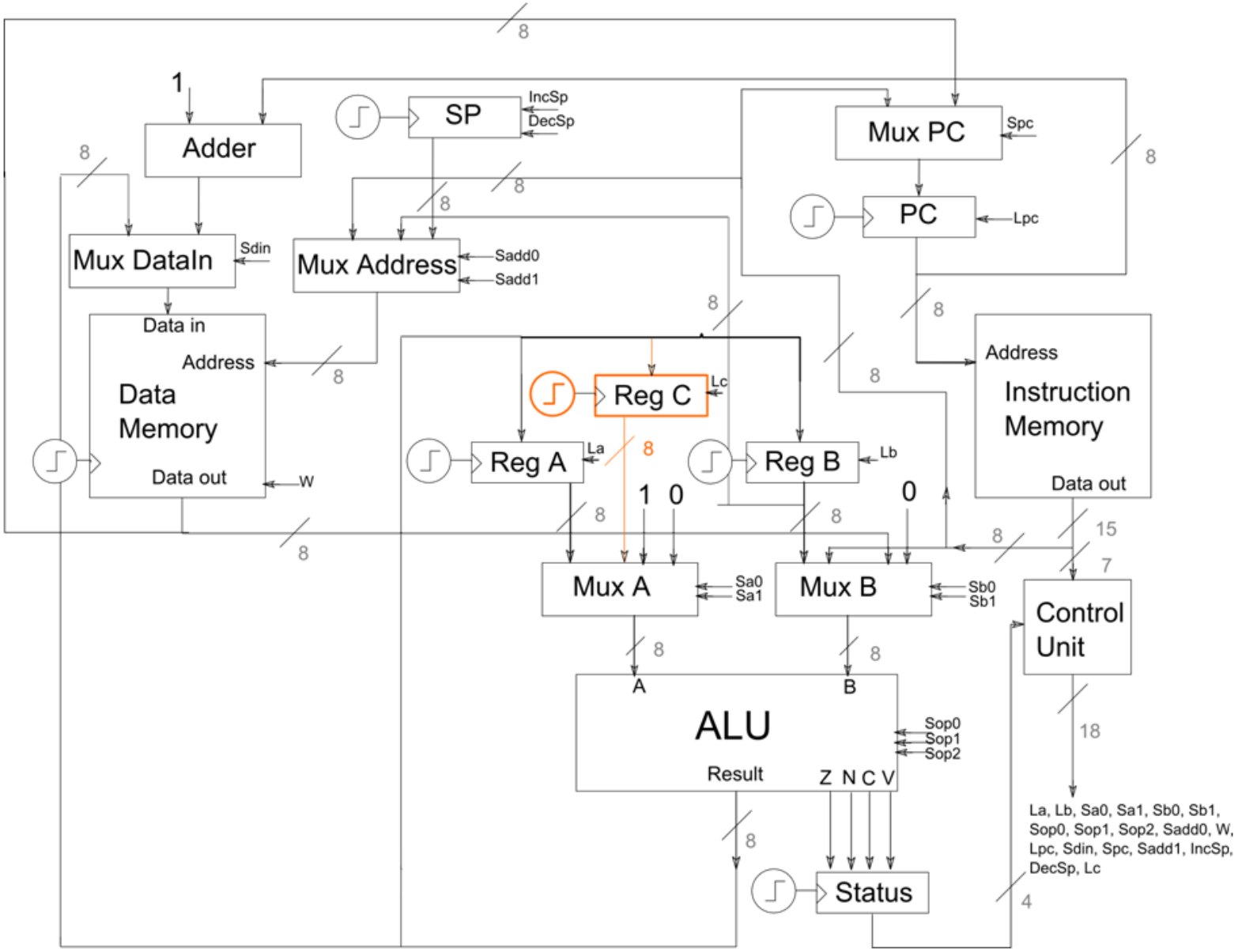


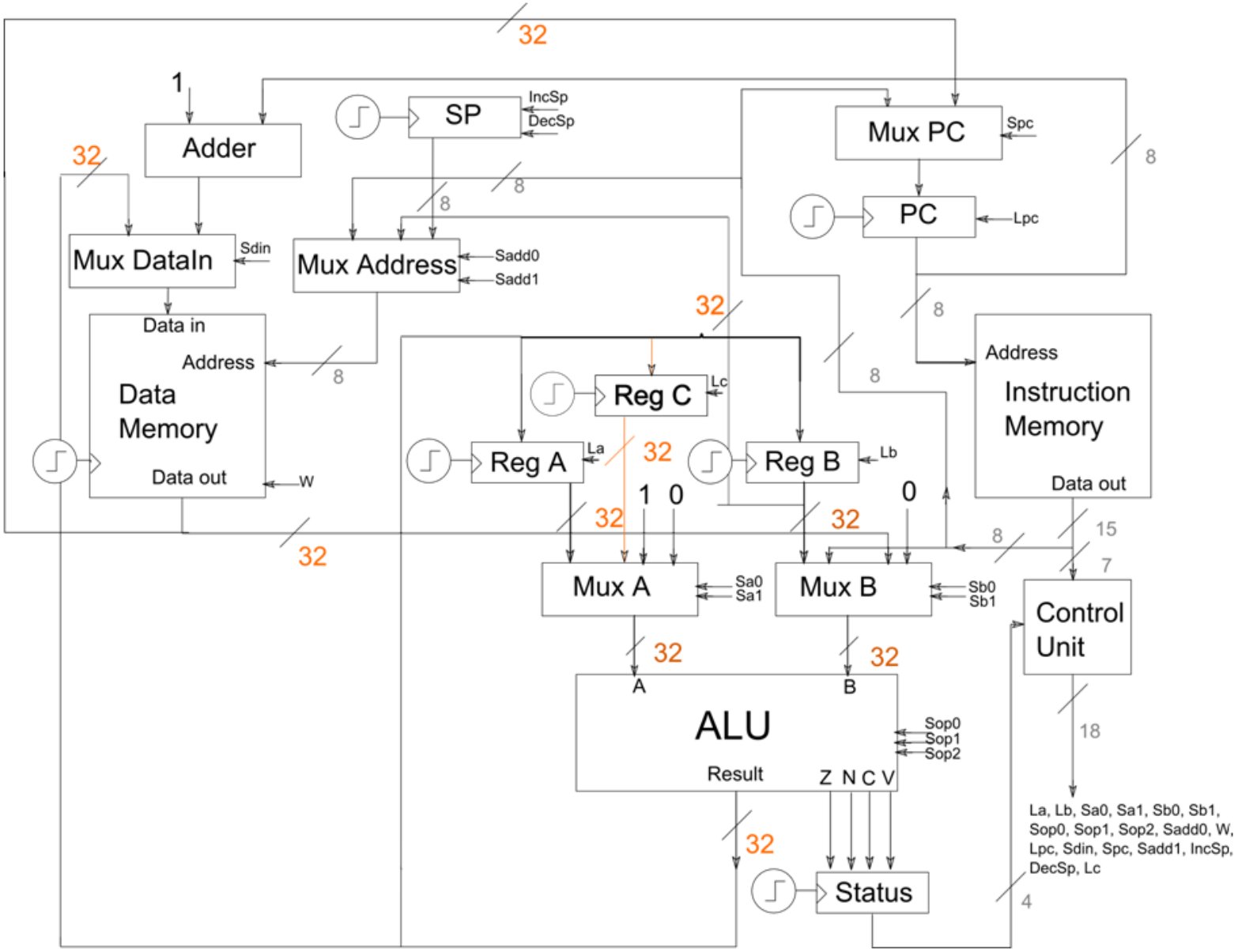
¿Cuál es la microarquitectura de nuestro computador?

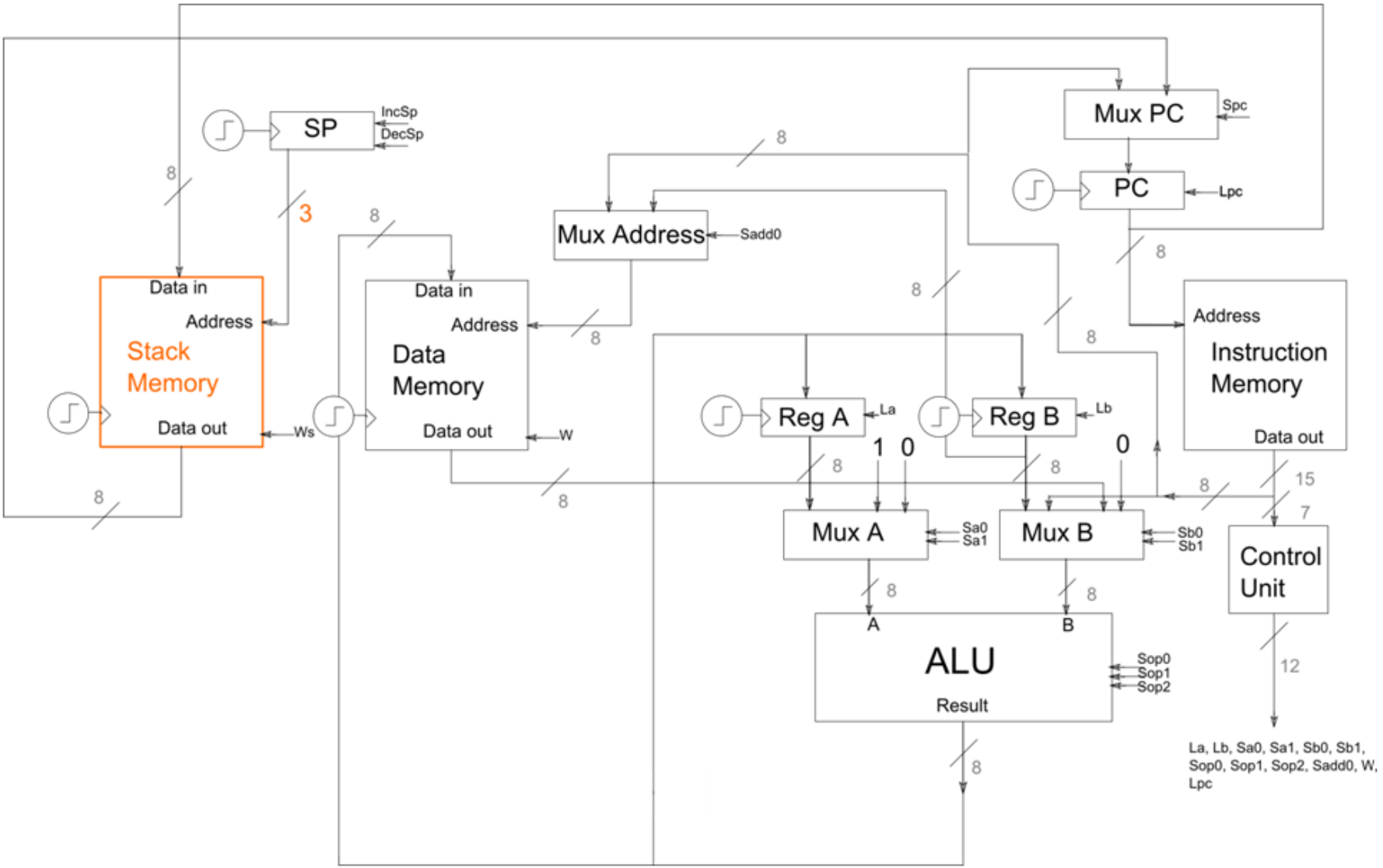
- Registros: A, B, SP, PC, Status
- Unidad de control: Simple (Hardwired)
- Tamaños: Regs., dir. mem., etc., 8 bits
- Unidad de ejecución: ALU
- Condition Codes: Z, N, C, V
- Stack: En memoria

Modifiquemos un poco la microarquitectura del computador básico




8
1
Adder
SP
IncSp
DecSp
Mux PC
Spc
PC
Lpc
Mux DataIn
Sdin
Mux Address
Sadd0
Sadd1
Data in
Address
Data Memory
Data out
W
Reg C
Lc
Reg A
La
Reg B
Lb
Address
Instruction Memory
Data out
15
7
1 0
0
Mux A
Sa0
Sa1
Mux B
Sb0
Sb1
Control Unit
A
B
ALU
Result
Z N C V
Sop0
Sop1
Sop2
18
La, Lb, Sa0, Sa1, Sb0, Sb1, Sop0, Sop1, Sop2, Sadd0, W, Lpc, Sdin, Spc, Sadd1, IncSp, DecSp, Lc
Status
4


32
1
Adder
32
SP
IncSp
DecSp
8
8
Mux PC
Spc
8
PC
Lpc
Mux DataIn
Sdin
Mux Address
Sadd0
Sadd1
32
8
Data in
Address
Data
Memory
Data out
W
8
Reg C
Lc
Reg A
La
32
Reg B
Lb
1 0
0
32
32
8
Address
Instruction
Memory
Data out
15
7
32
Mux A
Sa0
Sa1
Mux B
Sb0
Sb1
Control
Unit
32
32
A
B
ALU
Sop0
Sop1
Sop2
Result
Z N C V
18
La, Lb, Sa0, Sa1, Sb0, Sb1,
Sop0, Sop1, Sop2, Sadd0, W,
Lpc, Sdin, Spc, Sadd1, IncSp,
DecSp, Lc
32
Status
4

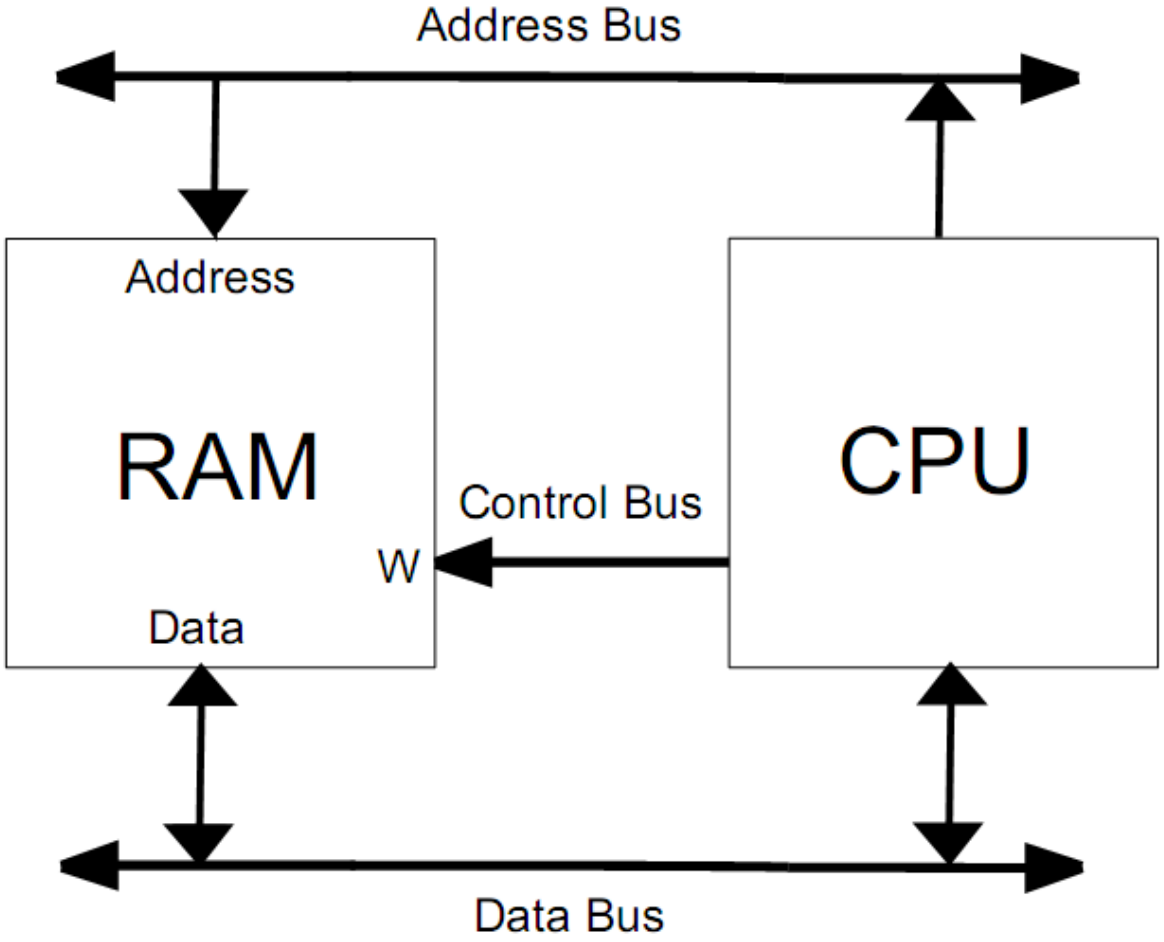

SP
IncSp
DecSp
3
Mux PC
Spc
PC
Lpc
Mux Address
Sadd0
Data in
Address
Stack Memory
Data out
Ws
Data in
Address
Data Memory
Data out
W
Reg A
La
Reg B
Lb
1 0
0
Address
Instruction Memory
Data out
15
7
Mux A
Sa0
Sa1
Mux B
Sb0
Sb1
Control Unit
A
B
ALU
Result
Sop0
Sop1
Sop2
12
La, Lb, Sa0, Sa1, Sb0, Sb1, Sop0, Sop1, Sop2, Sadd0, W, Lpc
8

Arquitecturas de von Neumann y Harvard se utilizan en distintos casos

La memoria presenta una división entre 2 grandes paradigmas dentro de la arquitectura de los computadores:

- Arquitectura Harvard: presenta memorias independientes para instrucciones y para datos.
- Arquitectura von Neumann: utiliza una sola memoria compartida entre instrucciones y datos. Permite escribir instrucciones como si estas fueran datos.

En von Neumann, el bus de instrucciones se agrega al bus bidireccional de datos



ISA especifica como escribir los programas para el computador

- Tipos de Instrucciones: carga, aritméticas,…
- Tipos de datos
- Modos de direccionamiento de memoria
- Manejo del stack
- Formato de instrucción
- Palabras por instrucción
- Ciclos por instrucción

ISA especifica como escribir los programas para el computador

Instrucción	Operandos	Opcode	Condition	Lpc	La	Lb	Sa0,1	Sb0,1	Sop0,1,2	Sadd0,1	Sdin0	Spc0	W	IncSp	DecSp
MOV	A,B	0000000		0	1	0	ZERO	B	ADD	-	-	-	0	0	0
	B,A	0000001		0	0	1	A	ZERO	ADD	-	-	-	0	0	0
	A,Lit	0000010		0	1	0	ZERO	LIT	ADD	-	-	-	0	0	0
	B,Lit	0000011		0	0	1	ZERO	LIT	ADD	-	-	-	0	0	0
	A,(Dir)	0000100		0	1	0	ZERO	DOUT	ADD	LIT	-	-	0	0	0
	B,(Dir)	0000101		0	0	1	ZERO	DOUT	ADD	LIT	-	-	0	0	0
	(Dir),A	0000110		0	0	0	A	ZERO	ADD	LIT	ALU	-	1	0	0
	(Dir),B	0000111		0	0	0	ZERO	B	ADD	LIT	ALU	-	1	0	0
	A,(B)	0001000		0	1	0	ZERO	DOUT	ADD	B	-	-	0	0	0
	B,(B)	0001001		0	0	1	ZERO	DOUT	ADD	B	-	-	0	0	0
	(B),A	0001010		0	1	0	A	ZERO	ADD	B	ALU	-	1	0	0
ADD	A,B	0001011		0	1	0	A	B	ADD	-	-	-	0	0	0
	B,A	0001100		0	0	1	A	B	ADD	-	-	-	0	0	0
	A,Lit	0001101		0	1	0	A	LIT	ADD	-	-	-	0	0	0
	A,(Dir)	0001110		0	1	0	A	DOUT	ADD	LIT	-	-	0	0	0
	A,(B)	0001111		0	0	1	A	DOUT	ADD	B	-	-	0	0	0
	(Dir)	0010000		0	0	0	A	B	ADD	LIT	ALU	-	1	0	0
SUB	A,B	0010001		0	1	0	A	B	SUB	-	-	-	0	0	0
	B,A	0010010		0	0	1	A	B	SUB	-	-	-	0	0	0
	A,Lit	0010010		0	1	0	A	LIT	SUB	-	-	-	0	0	0
	A,(Dir)	0010011		0	1	0	A	DOUT	SUB	LIT	-	-	0	0	0
	A,(B)	0010100		0	1	0	A	DOUT	SUB	B	-	-	0	0	0
	(Dir)	0010101		0	0	0	A	B	SUB	LIT	ALU	-	1	0	0
AND	A,B	0010110		0	1	0	A	B	AND	-	-	-	0	0	0
	B,A	0010111		0	0	1	A	B	AND	-	-	-	0	0	0
	A,Lit	0011000		0	1	0	A	LIT	AND	-	-	-	0	0	0
	A,(Dir)	0011001		0	1	0	A	DOUT	AND	LIT	-	-	0	0	0
	A,(B)	0011010		0	1	0	A	DOUT	AND	B	-	-	0	0	0
	(Dir)	0011011		0	0	0	A	B	AND	LIT	ALU	-	1	0	0
OR	A,B	0011100		0	1	0	A	B	OR	-	-	-	0	0	0
	B,A	0011101		0	0	1	A	B	OR	-	-	-	0	0	0
	A,Lit	0011110		0	1	0	A	LIT	OR	-	-	-	0	0	0
	A,(Dir)	0011111		0	1	0	A	DOUT	OR	LIT	-	-	0	0	0
	A,(B)	0100000		0	1	0	A	DOUT	OR	B	-	-	0	0	0
	(Dir)	0100001		0	0	0	A	B	IR	LIT	ALU	-	1	0	0
NOT	A,A	0100010		0	1	0	A	-	NOT	-	-	-	0	0	0
	B,A	0100011		0	0	1	A	-	NOT	-	-	-	0	0	0
	(Dir)	0100111		0	0	0	A	B	NOT	LIT	ALU	-	1	0	0

ISA especifica como escribir los programas para el computador

Instrucción	Operandos	Opcode	Condition	Lpc	La	Lb	Sa0,1	Sb0,1	Sop0,1,2	Sadd0,1	Sdin0	Spc0	W	IncSp	DecSp
XOR	A,B	0101000		0	1	0	A	B	XOR	-	-	-	0	0	0
	B,A	0101001		0	0	1	A	B	XOR	-	-	-	0	0	0
	A,Lit	0101010		0	1	0	A	LIT	XOR	-	-	-	0	0	0
	A,(Dir)	0101011		0	1	0	A	DOUT	XOR	LIT	-	-	0	0	0
	A,(B)	0101100		0	1	0	A	DOUT	XOR	B	-	-	0	0	0
	(Dir)	0101101		0	0	0	A	B	XOR	LIT	ALU	-	1	0	0
SHL	A,A	0101110		0	1	0	A	-	SHL	-	-	-	0	0	0
	B,A	0101111		0	0	1	A	-	SHL	-	-	-	0	0	0
	(Dir)	0110011		0	0	0	A	B	SHL	LIT	ALU	-	1	0	0
SHR	A,A	0110100		0	1	0	A	-	SHR	-	-	-	0	0	0
	B,A	0110101		0	0	1	A	-	SHR	-	-	-	0	0	0
	(Dir)	0111001		0	0	0	A	B	SHR	LIT	ALU	-	1	0	0
INC	B	0111010		0	0	1	ONE	B	ADD	-	-	-	0	0	0
CMP	A,B	0111011		0	0	0	A	B	SUB	-	-	-	0	0	0
	A,Lit	0111100		0	0	0	A	LIT	SUB	-	-	-	0	0	0
JMP	Dir	0111101		1	0	0	-	-	-	-	-	LIT	0	0	0
JEQ	Dir	0111110	Z=1	1	0	0	-	-	-	-	-	LIT	0	0	0
JNE	Dir	0111111	Z=0	1	0	0	-	-	-	-	-	LIT	0	0	0
JGT	Dir	1000000	N=0 y Z=0	1	0	0	-	-	-	-	-	LIT	0	0	0
JLT	Dir	1000001	N=1	1	0	0	-	-	-	-	-	LIT	0	0	0
JGE	Dir	1000010	N=0	1	0	0	-	-	-	-	-	LIT	0	0	0
JLE	Dir	1000011	N=1 o Z=1	1	0	0	-	-	-	-	-	LIT	0	0	0
JCR	Dir	1000100	C=1	1	0	0	-	-	-	-	-	LIT	0	0	0
JOV	Dir	1000101	V=1	1	0	0	-	-	-	-	-	LIT	0	0	0
CALL	Dir	1000101		1	0	0	-	-	-	SP	PC	LIT	1	0	1
RET		1000110		0	0	0	-	-	-	-	-	-	0	1	0
		1000111		1	0	0	-	-	-	SP	-	DOUT	0	0	0
PUSH	A	1001000		0	0	0	A	ZERO	ADD	SP	ALU	-	1	0	1
PUSH	B	1001001		0	0	0	ZERO	B	ADD	SP	ALU	-	1	0	1
POP	A	1001010		0	1	0	-	-	-	-	-	-	0	1	0
		1001011		0	1	0	ZERO	DOUT	ADD	SP	ALU	-	0	0	0
POP	B	1001100		0	0	1	-	-	-	-	-	-	0	1	0
		1001101		0	0	1	ZERO	DOUT	ADD	SP	ALU	-	0	0	0

RISC y CISC presentan soluciones con distinto foco para un mismo problema

Implementación de ISA responde generalmente a uno de dos paradigmas:

- RISC: Instrucciones pequeñas y simples. Diseñado para minimizar complejidad del hardware. Énfasis en el software.
- CISC: Muchas instrucciones y de alta complejidad. Énfasis en el hardware.

¿Cuál es la arquitectura del set de instrucciones de nuestro computador?

- Tipos de inst.: Carga, aritmética, salto, …
- Tipos de dato: Entero binario con y sin signo
- Direccionamiento: Directo, indirecto por reg.
- Manejo stack: General
- Formato de inst.: Mixto (Inst. + 0, 1 ó 2 args.)
- Palabras por inst.: 1 (salvo RET y POP)
- Ciclos por inst.: 1 (salvo RET y POP)
- RISC

Finalicemos esta unidad con un pequeño ejercicio

Se desea modificar la arquitectura del computador básico, para que soporte de manera nativa el uso de número reales.

- Modifique la microarquitectura para soportar de manera nativa el uso y operaciones de número reales.
- Modifique la ISA par dar soporte a las instrucciones relacionadas con el uso de números reales.

No tenemos idea como representar, almacenar y operar números reales

¿Cómo escribimos comúnmente números racionales?

- Podemos expandir usando exponentes negativos

$$123{,}45 = 1\cdot10^{2} + 2\cdot10^{1}+3\cdot10^{0}+4\cdot10^{-1}+5\cdot10^{-2}$$

- En binario es lo mismo

$$101{,}01 = 1\cdot2^{2} + 0\cdot2^{1}+1\cdot2^{0}+0\cdot2^{-1}+1\cdot2^{-2} = 5{,}25$$

¿Cómo lo hacemos para pasar de decimal a binario?

Necesitamos obtener el valor de una división en binario

- Por ejemplo:

$$10^{-1} = 0,1 = \frac{1}{10} = \frac{(1)_2}{(1010)_2}$$

- Y el resultado de esa división es: $0,0\overline{0011}$

- Decimal finito pasa a ser infinito en binario = Pésimo
- Todo depende de la base elegida (ej. 1/3 en ternario)

El secreto oscuro de la programación

- Los resultados de los ejemplos son inesperados

¿Por qué pasa esto?

- Estos números usan memoria finita para manejar rangos muy grandes y densos.
- Luego, existe un trade-off entre rango y precisión.

Caso real: Misil Patriot[1]

- 28 personas murieron en 1991, debido al mal funcionamiento de un misil Patriot.
- El Patriot es un sistema defensivo para interceptar objetivo aéreos, que utiliza misiles.
- Error fue ocasionado por aproximación de un decimal finito mediante un número binario infinito.
- Debido al error, el sistema no siguió correctamente al objetivo y el misil nunca fue disparado.

[1] www.youtube.com/watch?v=EMVBLg2MrLs

Representación de punto fijo

- Dados *n* dígitos (bits), estos se dividen de manera fija para representar signo, parte entera y parte fraccional.
- Pro: simple y rápido
- Contra: rango pequeño
- Idea: mover (flotar) la coma (punto)

Representación de punto flotante

- Basada en notación científica normalizada
- Dos elementos centrales: significante y exponente
- Codifica la posición del punto

- Pro: gran rango
- Contra: pérdida de precisión

IEEE754, el formato más usado para números de punto flotante

float (32 bits):

- 1 bit de signo, 8 bit exponente, 23 bit significante
- Significante normalizado
- Exponente desfasado en 127
- 0: exponente = 0, significante = 0
- ±∞: exponente = 11111…, significante = 0
- *NaN*: exponente = 11111…, significante ≠ 0

double: 64 bits, reglas similares

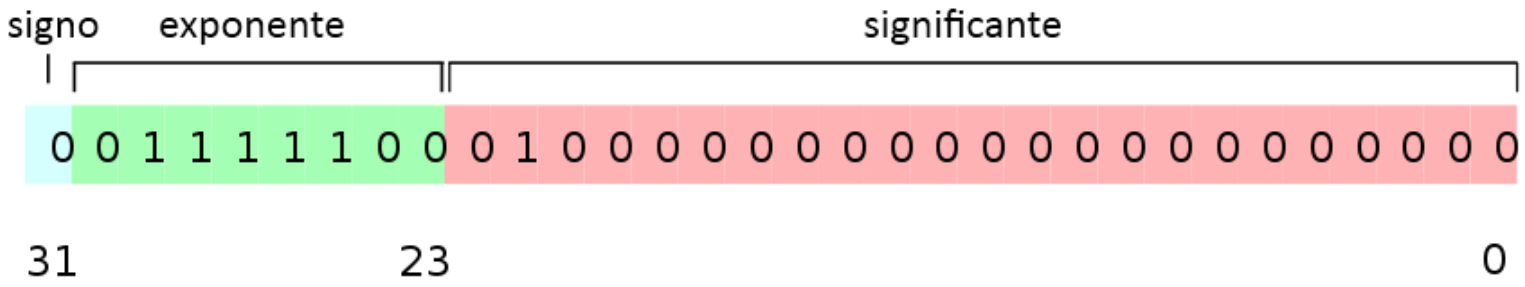
$X = (-1)^{signo} \cdot 1.significante \cdot 2^{exponente-127}$

$0.00101b = (1.01 \cdot 10^{-11})_2 = 1 \cdot 2^{-3} + 1 \cdot 2^{-5} = 0.15625$

signo = 0

significante = 01

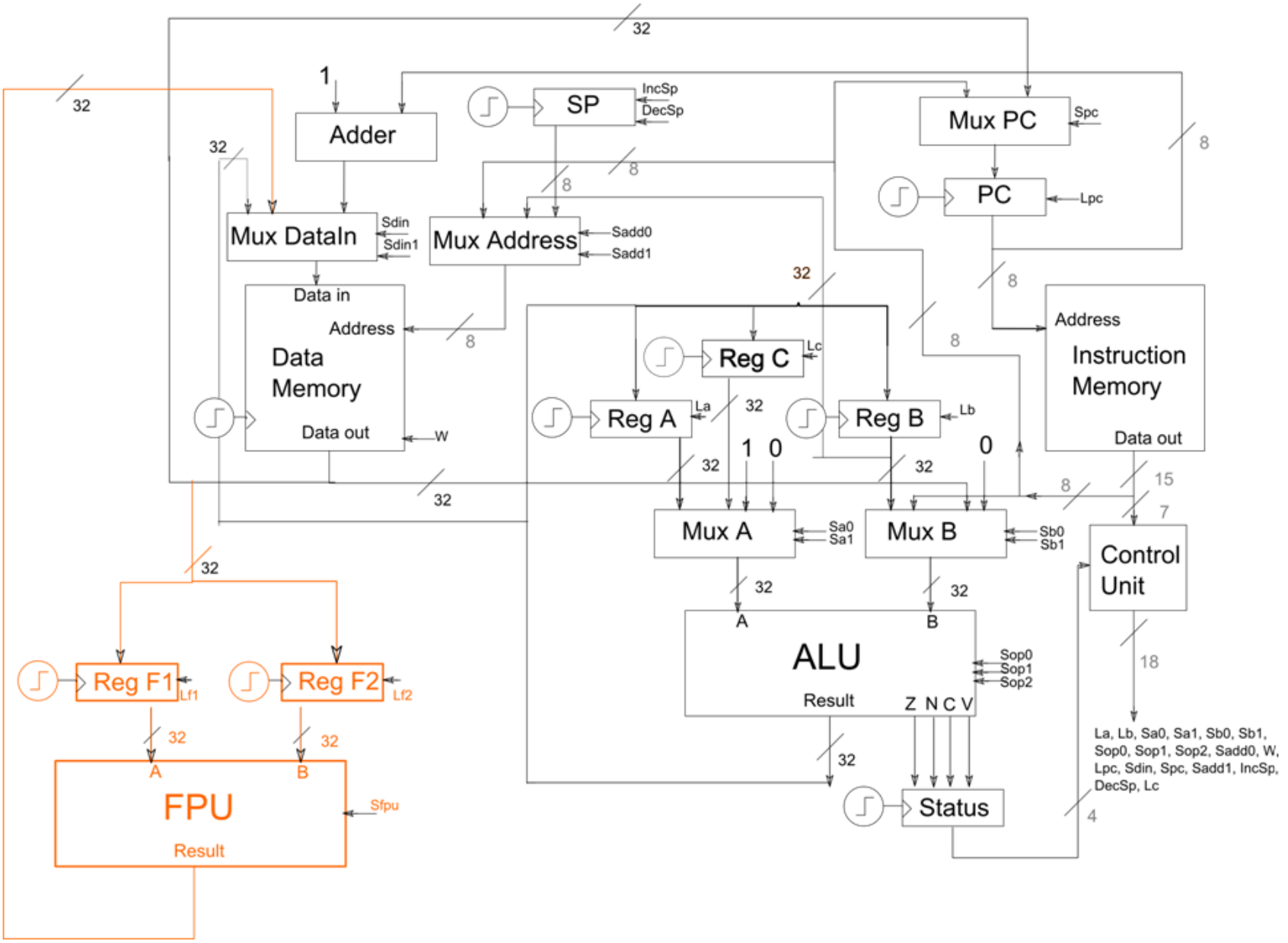
exponente = 124 = 01111100



Representaciones alternativas

- Decimales como números enteros
  - Se utiliza como unidad el menor valor decimal requerido
  - Elimina problemas de aproximación, pero tiene poco rango
- Punto flotante con base decimal
  - Se cambia la base de 2 a 10
  - Elimina problemas de aproximación, pero resulta muy lento
  - Ideal para cálculos “humanos”
- Punto flotante con base decimal y precisión arbitraria
  - Tamaño asignado a significante y exponente se aumenta de acuerdo a las necesidades
  - Lentísimo, pero el más adecuado para cálculos “humanos”

Volvamos ahora al código y luego al ejercicio


32
1
Adder
SP
IncSp
DecSp
Mux PC
Spc
PC
Lpc
8
Mux DataIn
Sdin
Sdin1
Mux Address
Sadd0
Sadd1
Data in
Address
Data
Memory
Data out
W
Reg C
Lc
Reg A
La
Reg B
Lb
1 0
0
Address
Instruction
Memory
Data out
15
7
Mux A
Sa0
Sa1
Mux B
Sb0
Sb1
Control
Unit
A
B
ALU
Sop0
Sop1
Sop2
Result
Z N C V
18
La, Lb, Sa0, Sa1, Sb0, Sb1,
Sop0, Sop1, Sop2, Sadd0, W,
Lpc, Sdin, Spc, Sadd1, IncSp,
DecSp, Lc
4
Status
Reg F1
Lf1
Reg F2
Lf2
FPU
Sfpu
Result

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

IIC2343 – Arquitectura de Computadores

Arquitecturas de Computadores

Profesor: Hans Löbel