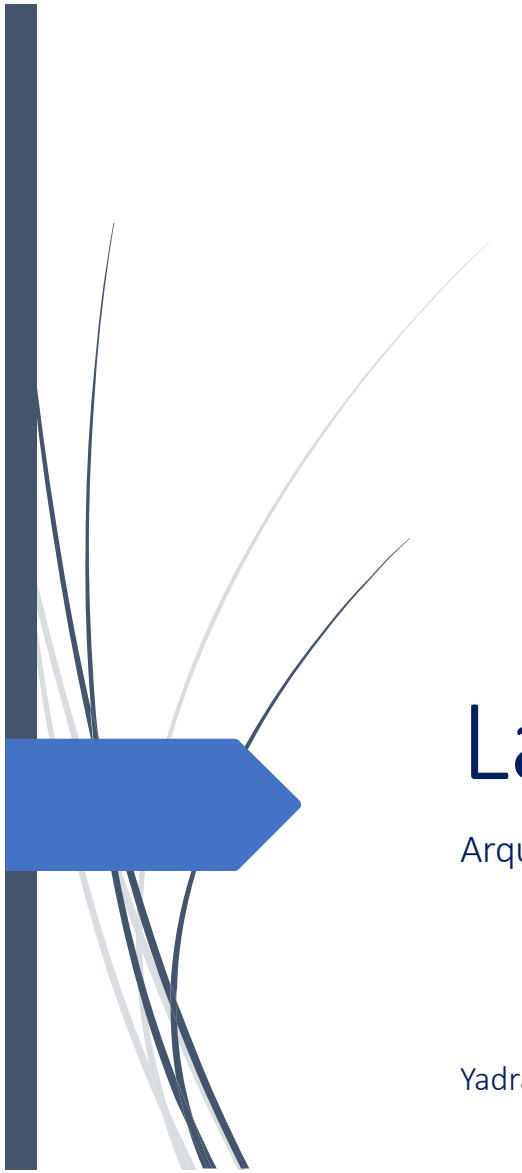


La arquitectura del computador

Arquitectura de Computadores – IIC2343

Yadran Eterovic S. (yadran@uc.cl) y Hans Löbel D.

2025-2

Nuestro **computador básico** tiene todas las funcionalidades básicas:

Registros + unidades de ejecución + unidades de control

Además de hacer cálculos, puede realizar operaciones de control de flujo

Provee modularidad básica, al dar soporte para subrutinas

Nuestro computadoir básico refleja una de muchas **arquitectura**s posibles:

Distintos procesadores pueden diferir en el conjunto de **funcionalidades básicas**

Existen procesadores que son programados de la misma manera (p.ej., AMD e Intel), pero su construcción interna es distinta

Las decisiones en cuanto a cantidad de registros, tamaño de buses, memorias, instrucciones, etc., definen la **arquitectura de un computador**

El/la arquitecto/a computacional debe:

Primero, determinar cuáles atributos son importantes para un nuevo computador

Luego, diseñar un computador para maximizar desempeño y eficiencia energética

… y al mismo tiempo cumplir las restricciones de costo, potencia y disponibilidad

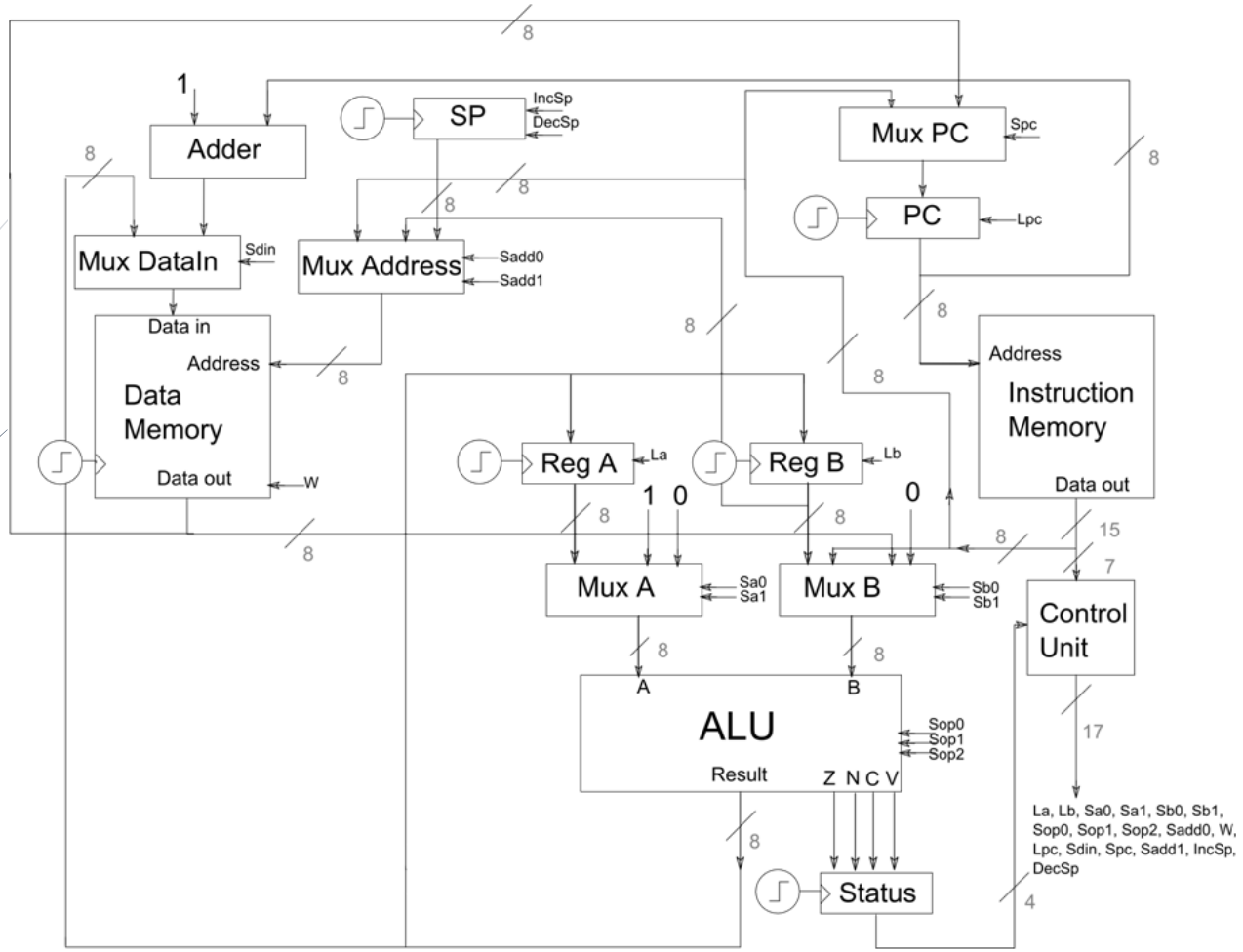
Formalmente, la arquitectura de un procesador está definida por **dos elementos**:

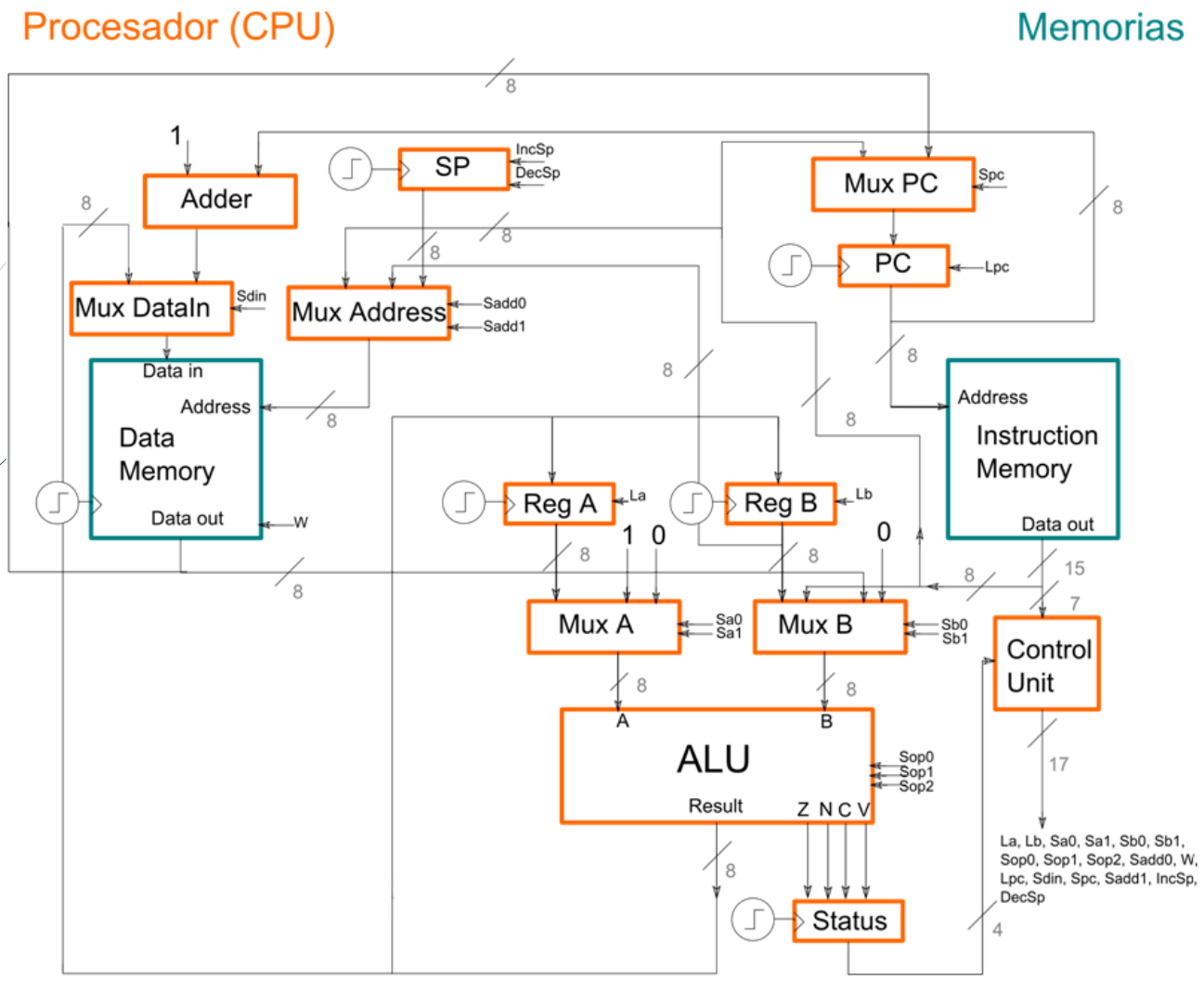
La **microarquitectura** (+ el *hardware*) se refiere a los aspectos de alto nivel del diseño —sistema de memoria, interconexión memoria-CPU, diseño interno de la CPU (aritmética, lógica, saltos y transferencia de datos)—

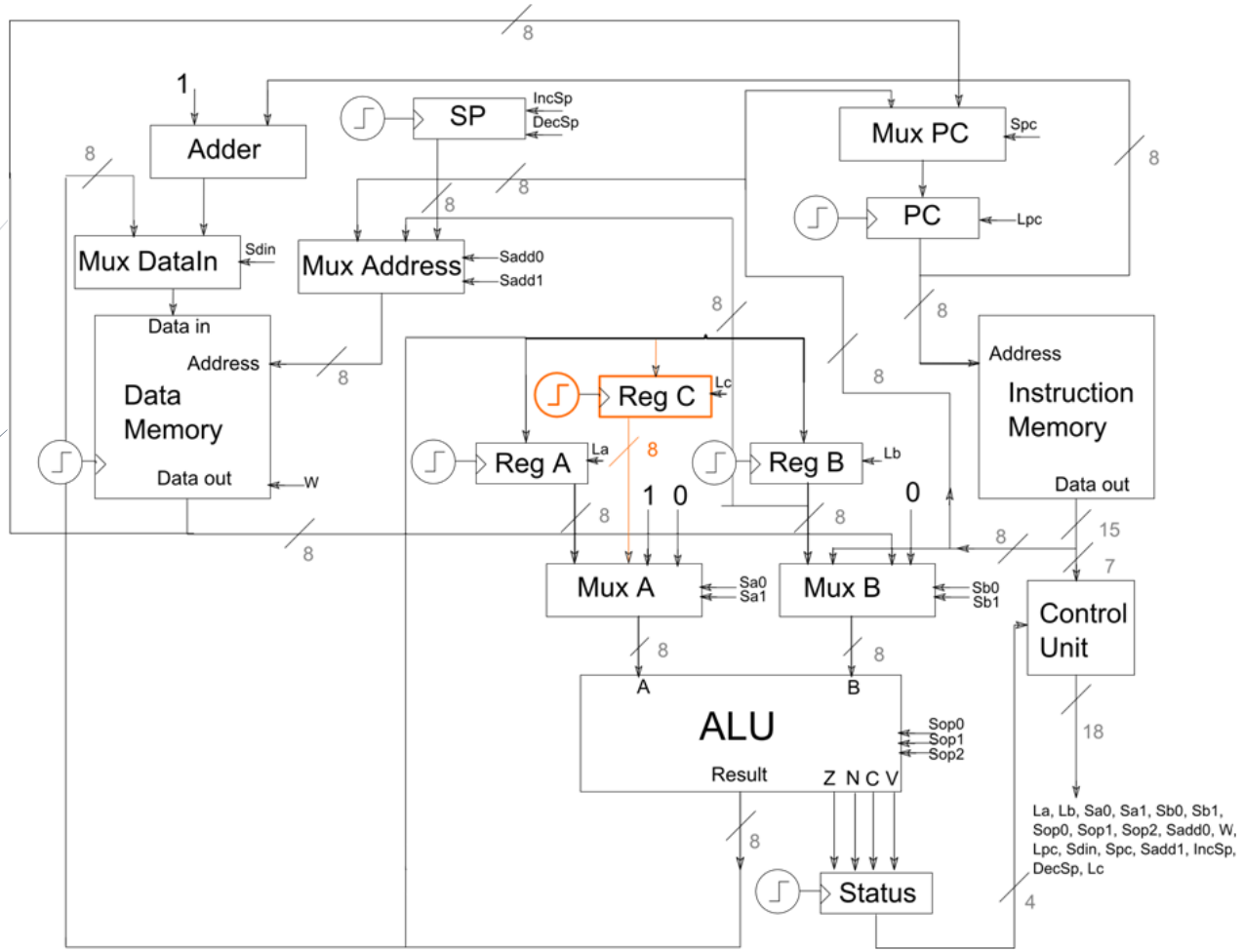
… más el diseño lógico detallado (p.ej., la frecuencia del reloj) y la tecnología de empaquetamiento

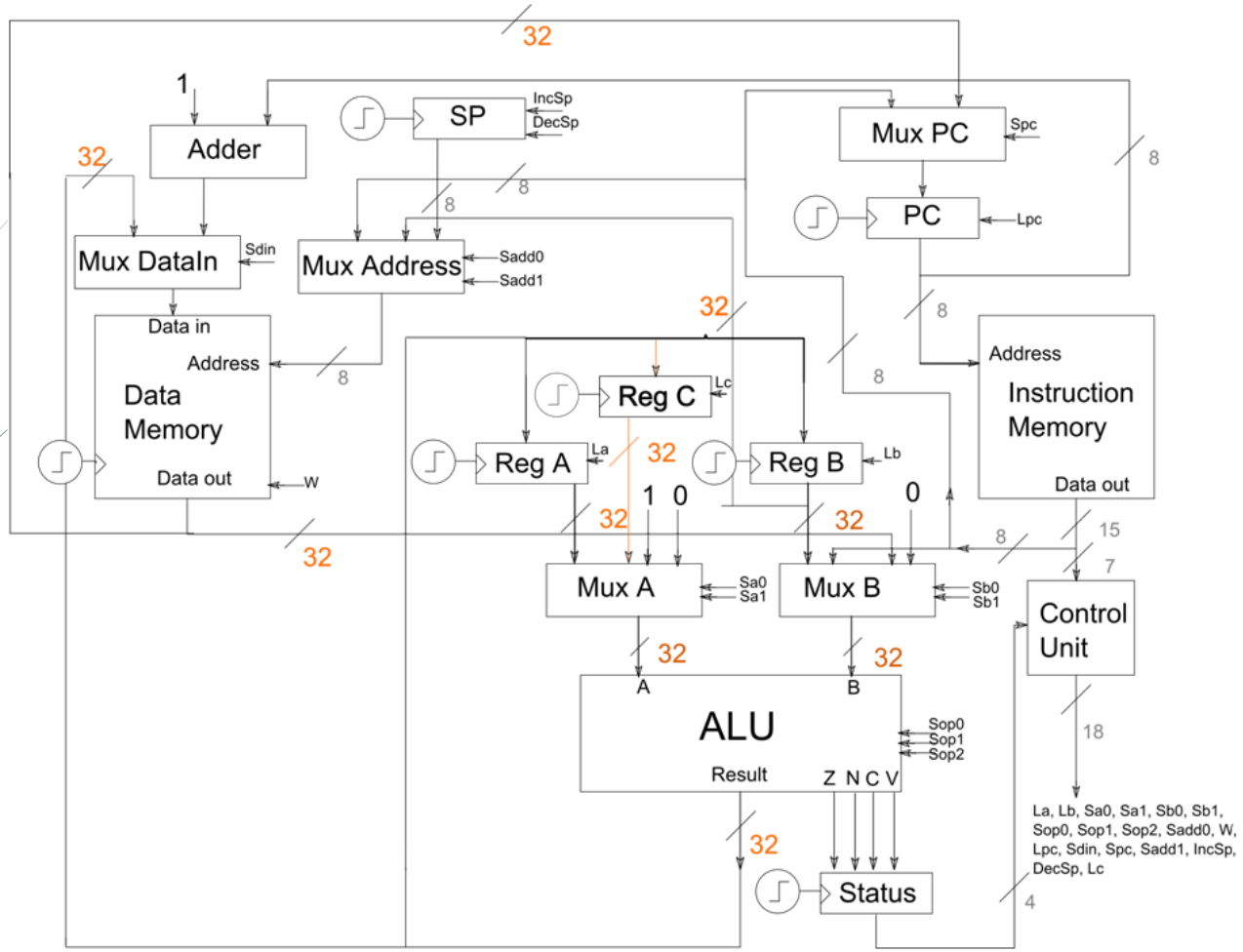
La **arquitectura del conjunto de instrucciones (ISA)** se refiere a las instrucciones que el computador es capaz de ejecutar y que son visibles para el programador

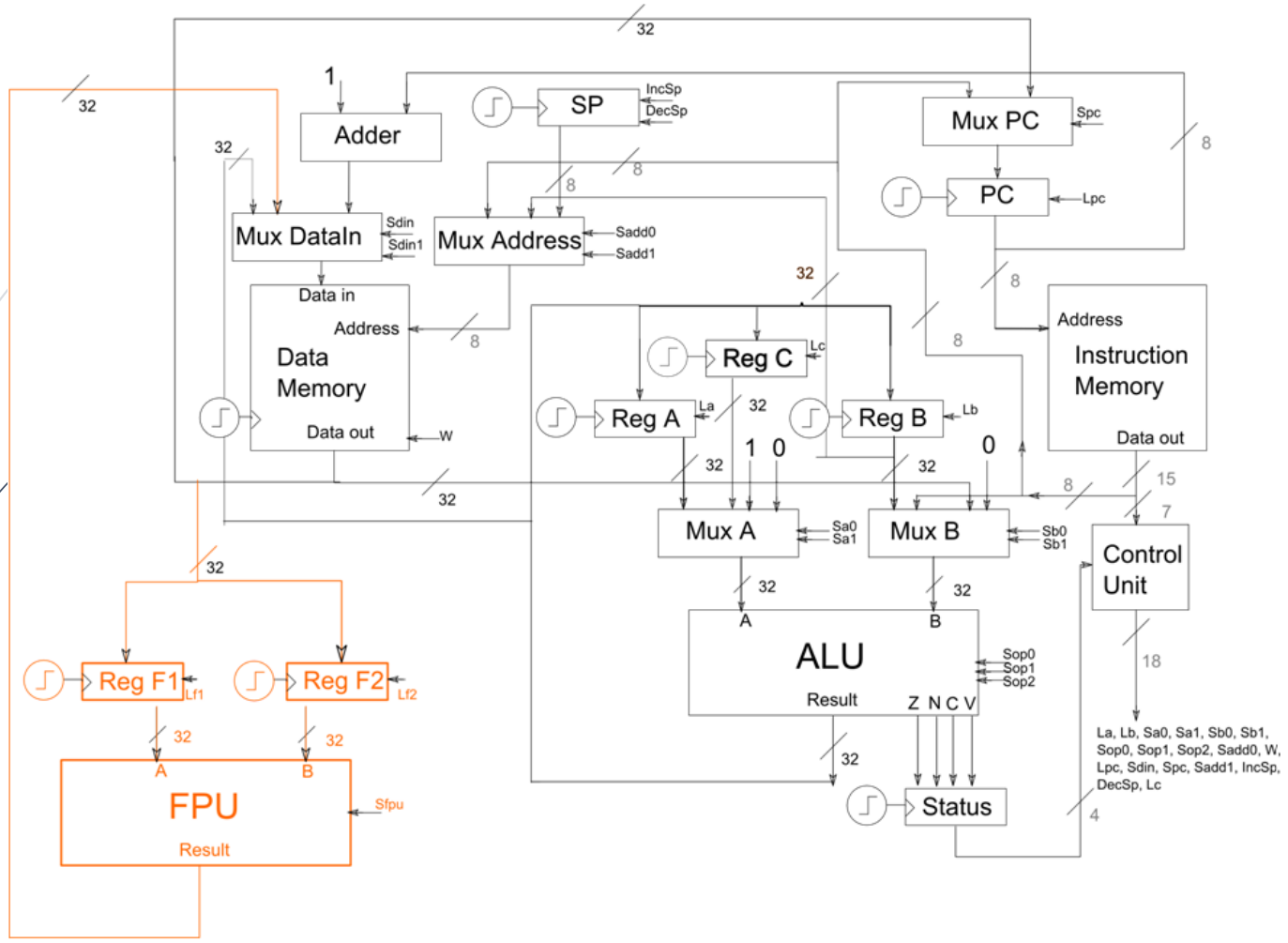
… es decir, lo que está relacionado con la programación del computador y que sirve de frontera entre el software y el hardware

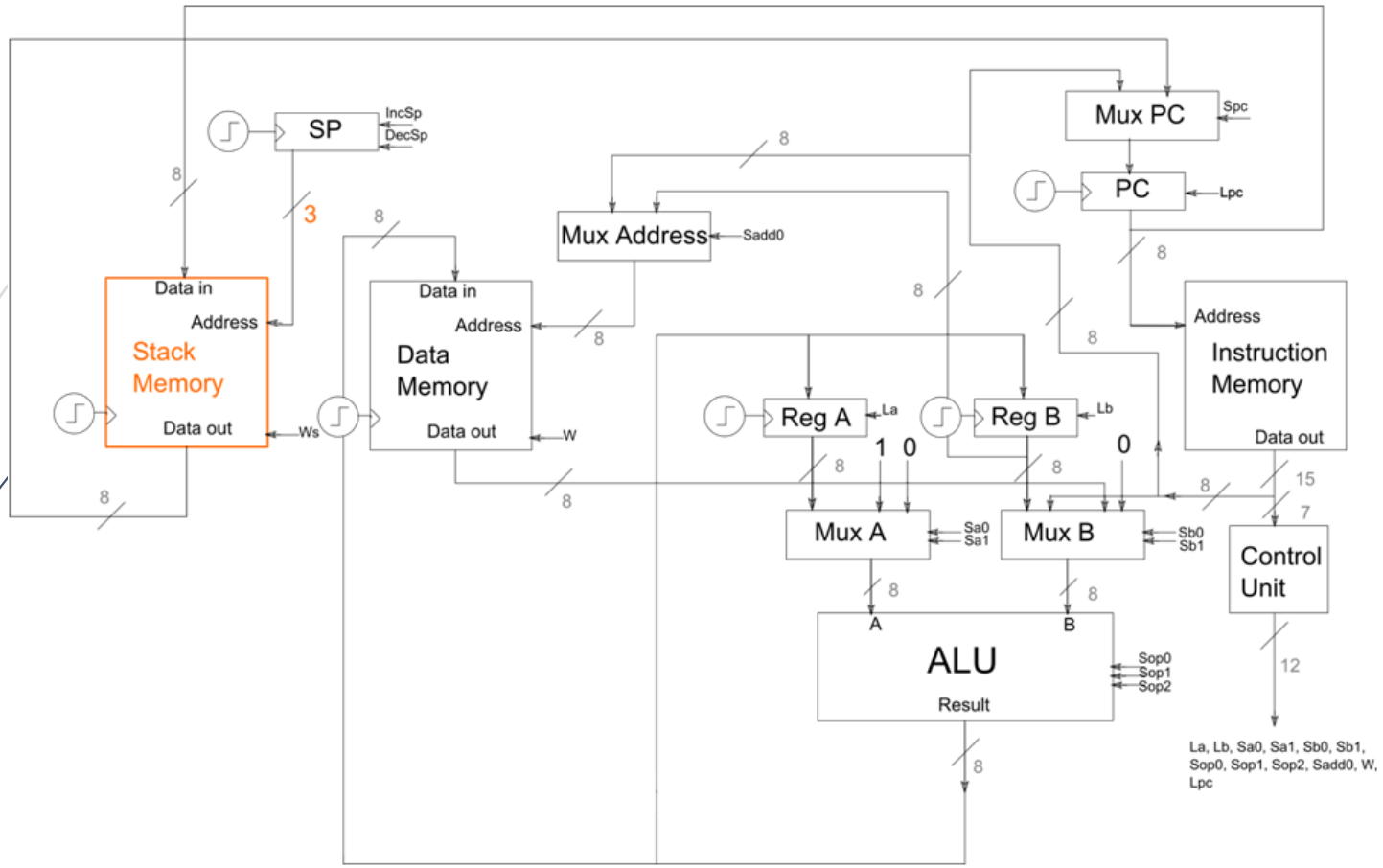

8
1
Adder
SP
IncSp
DecSp
Mux PC
Spc
PC
Lpc
Mux DataIn
Sdin
Mux Address
Sadd0
Sadd1
Data in
Address
Data
Memory
Data out
W
Address
Instruction
Memory
Data out
Reg A
La
Reg B
Lb
1 0
0
15
7
Mux A
Sa0
Sa1
Mux B
Sb0
Sb1
Control
Unit
A
B
ALU
Result
Z N C V
Sop0
Sop1
Sop2
17
La, Lb, Sa0, Sa1, Sb0, Sb1,
Sop0, Sop1, Sop2, Sadd0, W,
Lpc, Sdin, Spc, Sadd1, IncSp,
DecSp
Status
4


Procesador (CPU)
Memorias
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 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
Z N C V
Sop0
Sop1
Sop2
17
La, Lb, Sa0, Sa1, Sb0, Sb1,
Sop0, Sop1, Sop2, Sadd0, W,
Lpc, Sdin, Spc, Sadd1, IncSp,
DecSp
Status
4


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

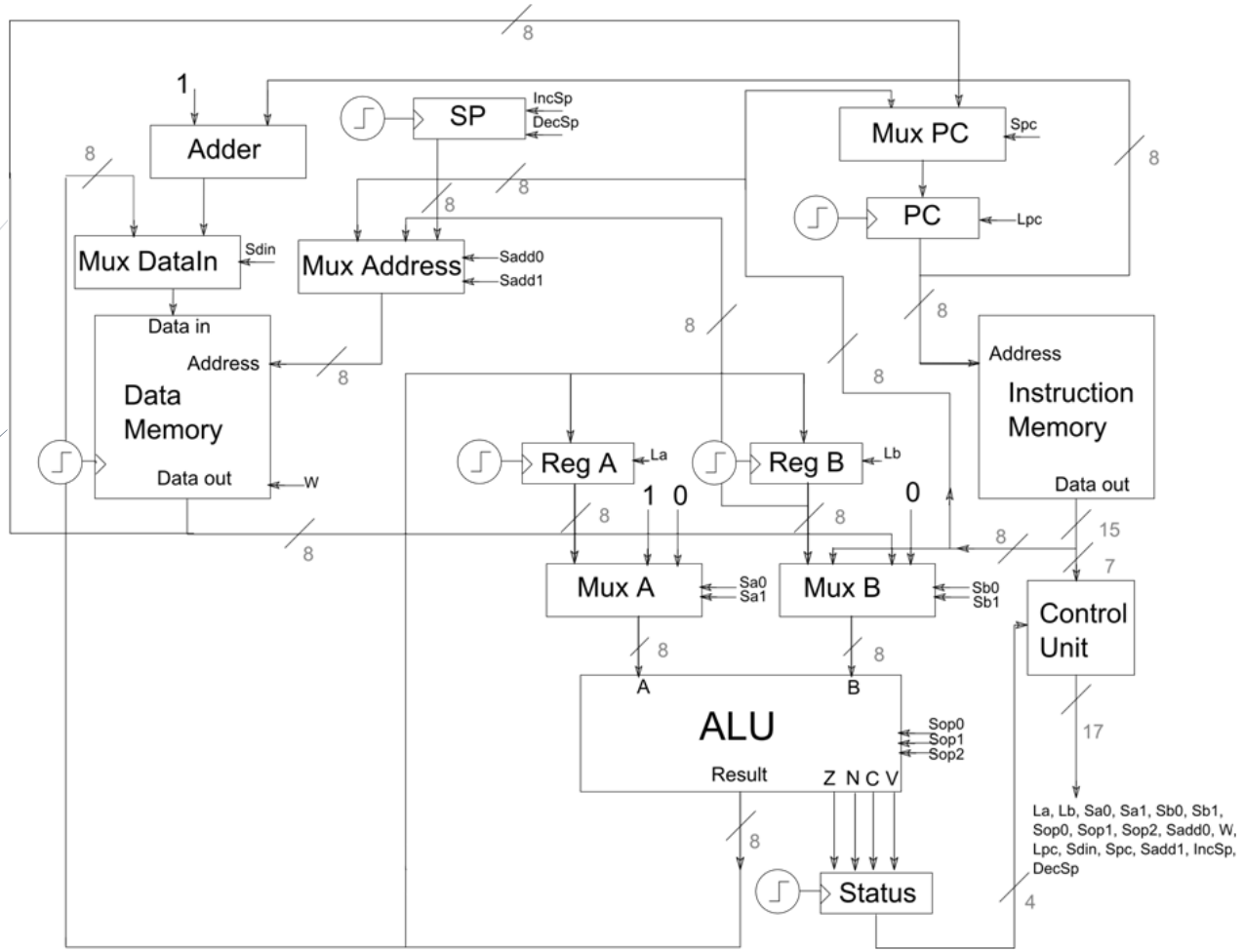

32
1
Adder
SP
IncSp
DecSp
Mux PC
Spc
PC
Lpc
8
Mux DataIn
Sdin
Sdin1
Mux Address
Sadd0
Sadd1
Data in
Address
Data
Memory
Data out
W
Reg C
Lc
Reg A
La
Reg B
Lb
1 0
0
Address
Instruction
Memory
Data out
15
7
Mux A
Sa0
Sa1
Mux B
Sb0
Sb1
Control
Unit
18
A
B
ALU
Sop0
Sop1
Sop2
Result
Z N C V
Status
La, Lb,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


SP
IncSp
DecSp
3
8
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

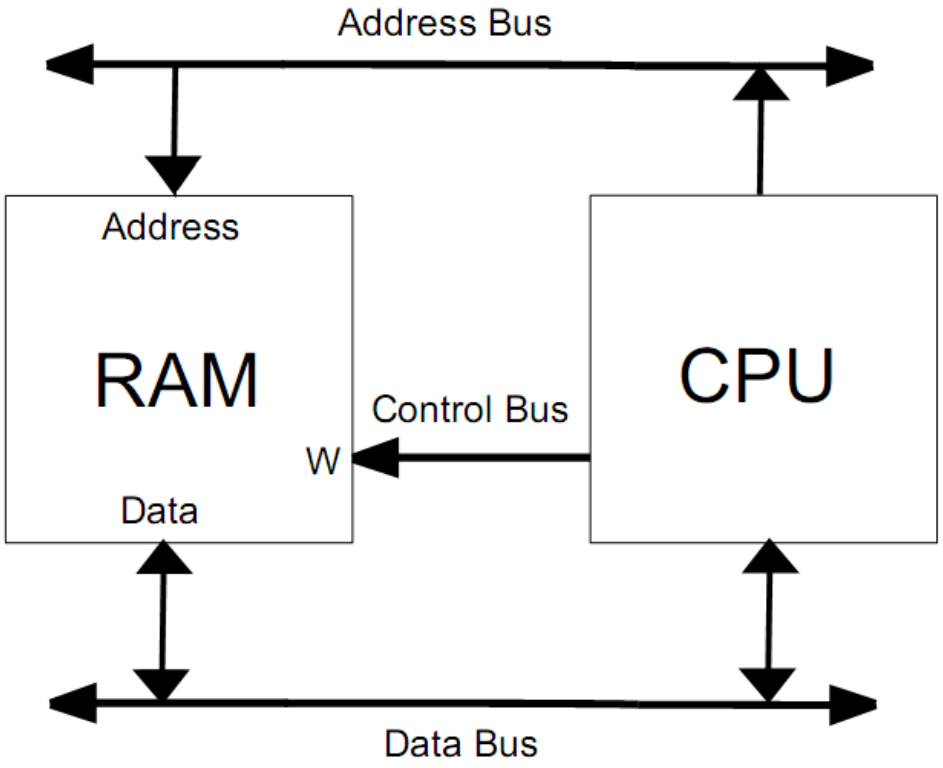
En los inicios de los computadores modernos, aparecieron **dos enfoques** relativos a la organización de la memoria:

En la **arquitectura Harvard** hay memorias independientes para instrucciones y para datos; p.ej., nuestro computador básico

En la **arquitectura Von Neumann** hay una sola memoria compartida por instrucciones y datos


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 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
Z N C V
Sop0
Sop1
Sop2
17
La, Lb, Sa0, Sa1, Sb0, Sb1,
Sop0, Sop1, Sop2, Sadd0, W,
Lpc, Sdin, Spc, Sadd1, IncSp,
DecSp
Status
4

En la **arquitectura Von Neumann** el bus de instrucciones se agrega al bus bidireccional de datos



La ISA especifica cómo escribir programas para el computador:

tipos de instrucciones

tipos de datos

modos de direccionamiento

manejo del stack

formato de las instrucciones

número de palabras por instrucción

número de ciclos por instrucción

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

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

La implementación de la **ISA** responde a uno de dos grandes paradigmas:

En **RISC (reduced instruction set computer)** las instrucciones son pocas y simples, de manera de minimizar la complejidad del hardware y poner énfasis en el software

En **CISC (complex instruction set computer)** las instrucciones son muchísimas y complejas, de manera que el énfasis está en el hardware

La ISA del computador básico es RISC:

tipos de instrucciones:	aritméticas, control de flujo, almacenamiento (*store*) y carga (*load*)
tipos de datos:	entero binario con y sin signo
modos de direccionamiento:	directo, indirecto por registro
manejo del stack:	general
formato de las instrucciones:	mixto (instr. + 0, 1, 2 args.)
número de palabras por instrucción:	1 (excepto **RET** y **POP**)
número de ciclos por instrucción:	1 (excepto **RET** y **POP**)

Siete puntos de vista para analizar y comparar ISAs:

Clase: Hoy todas son *arquitecturas de registros de propósito general*, en que los operandos son registros o localidades en memoria; p.ej., la ISA 80x86 tiene 16 registros de proósito general y 16 registros para datos de punto flotante; la ISA RISC-V tiene 32 registros de proósito general y 32 registros para punto flotante.

Hay ISAs *registro-memoria* (muchas instrucciones tienen acceso a memoria, p.ej., 80x86) e ISAs *load-store* (sólo las instrucciones de tipos *load* y *store* tienen acceso a memoria, p.ej., RISC-V y la ISA ARMv8)

Direccionamiento de memoria: Hoy todos los procesadores usan *direccionamiento de bytes* para los operandos que están en memoria.

P.ej., ARMv8 exige que los operandos estén *alineados*; 80x86 y RISC-V no lo exigen, pero los accesos son más rápidos si los operandos están alineados

Tipos y tamaños de operandos: 8 bits (carácter ASCII), 16 bits (carácter Unicode o media palabra), 32 bits (entero o palabra), 64 bits (entero largo o palabra doble), y punto flotante IEEE 754 de 32 bits y de 64 bits

Siete puntos de vista para analizar y comparar ISAs (*continuación*):

Operaciones: Transferencia de datos, aritmética-lógicas, control, punto flotante

Instrucciones del flujo de control: Saltos condicionales, saltos incondicionales, llamadas a procedimientos, y *returns*

80x86, ARMv8 y RISC-V usan direccionamiento relativo al PC: la dirección destino (a la cual se "salta") se especifica mediante un campo que se suma al PC

Modos de direccionamiento (para los operandos): Pueden especificar **registros**, operandos **constantes** (*literal* o *inmediate*), y **direcciones de memoria** (donde residen los operandos) mediante *displacement* (una constante se suma al valor de un registro para formar la dirección de memoria)

RISC-V tiene estos tres modos; ARMv8 también, además de direccionamiento relativo al PC y suma de dos registros; 80x86 también, incluyendo tres variaciones de *displacement*: sin registro (absoluto), y dos variaciones de dos registros

Codificación: Las instrucciones pueden tener longitud fija o longitud variable …