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

IIC2343 – Arquitectura de Computadores

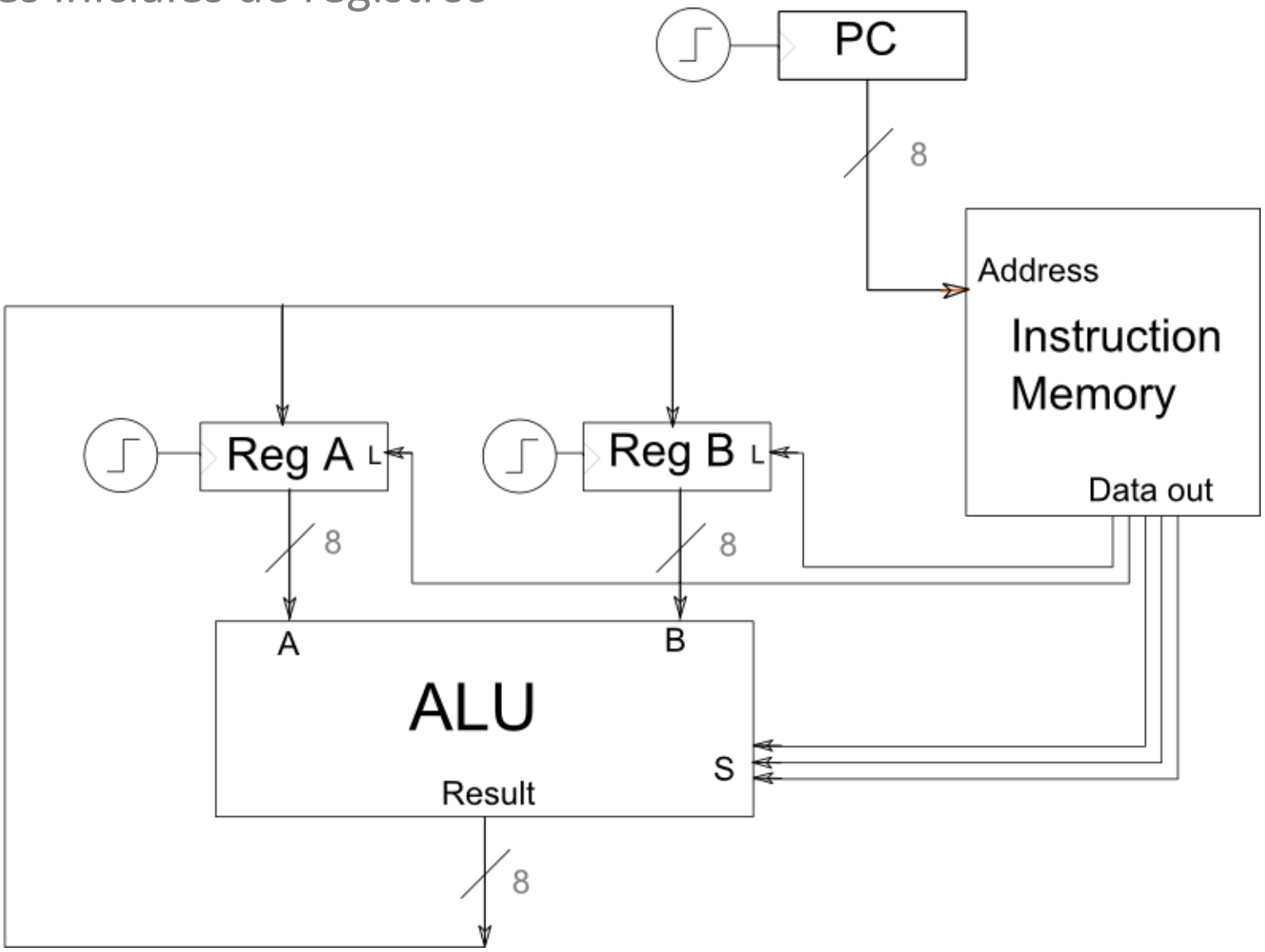
Programabilidad Parte 2 - Computador Básico

Profesor: Hans Löbel

Todavía restan varias cosas para llamar a “esto” un computador

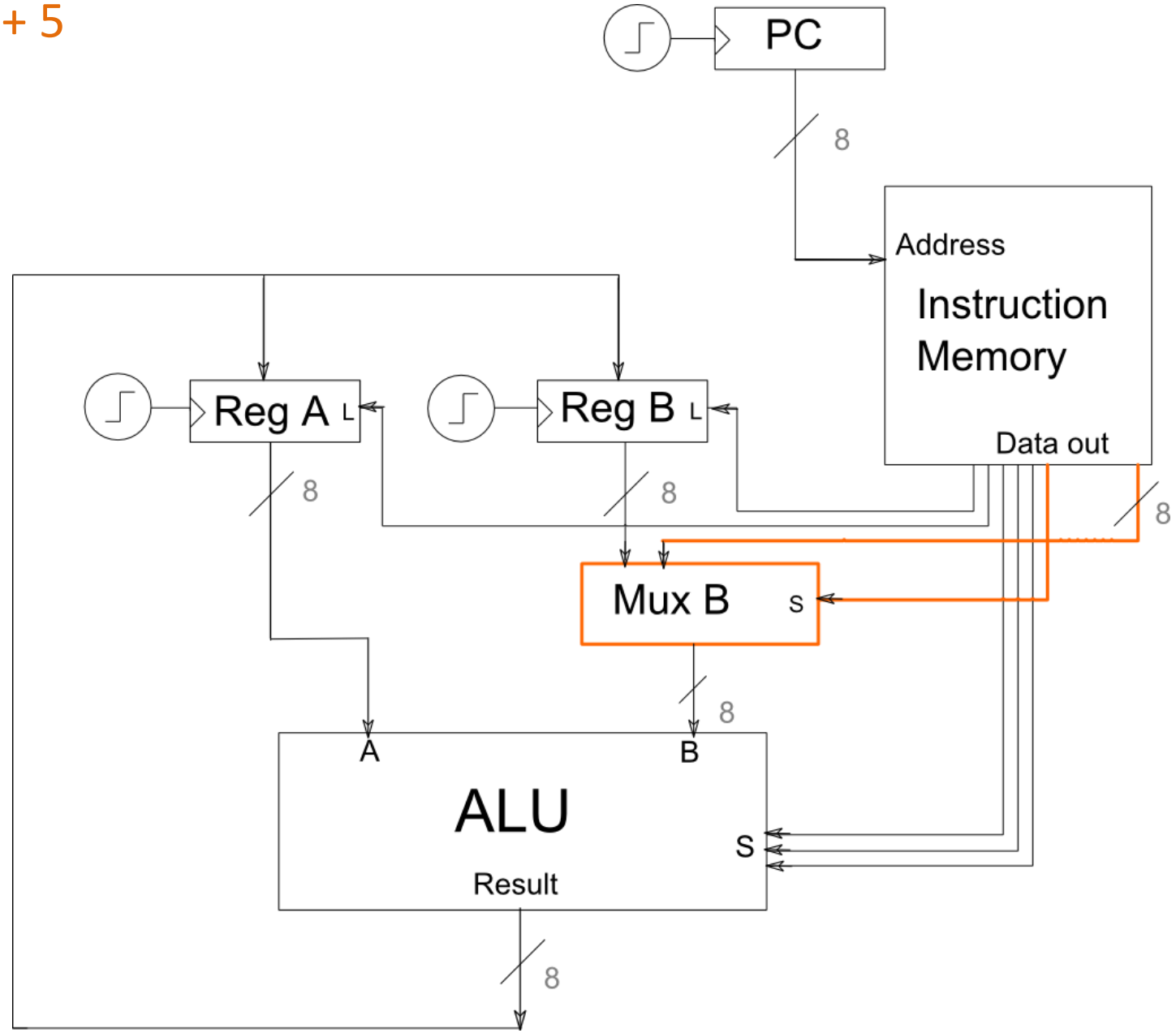
- Tenemos limitaciones en los valores iniciales de los registros.
- No tenemos donde almacenar más de dos datos.
- No podemos programar esta máquina de manera razonable.

Máquina programable aún está limitada por valores iniciales de registros



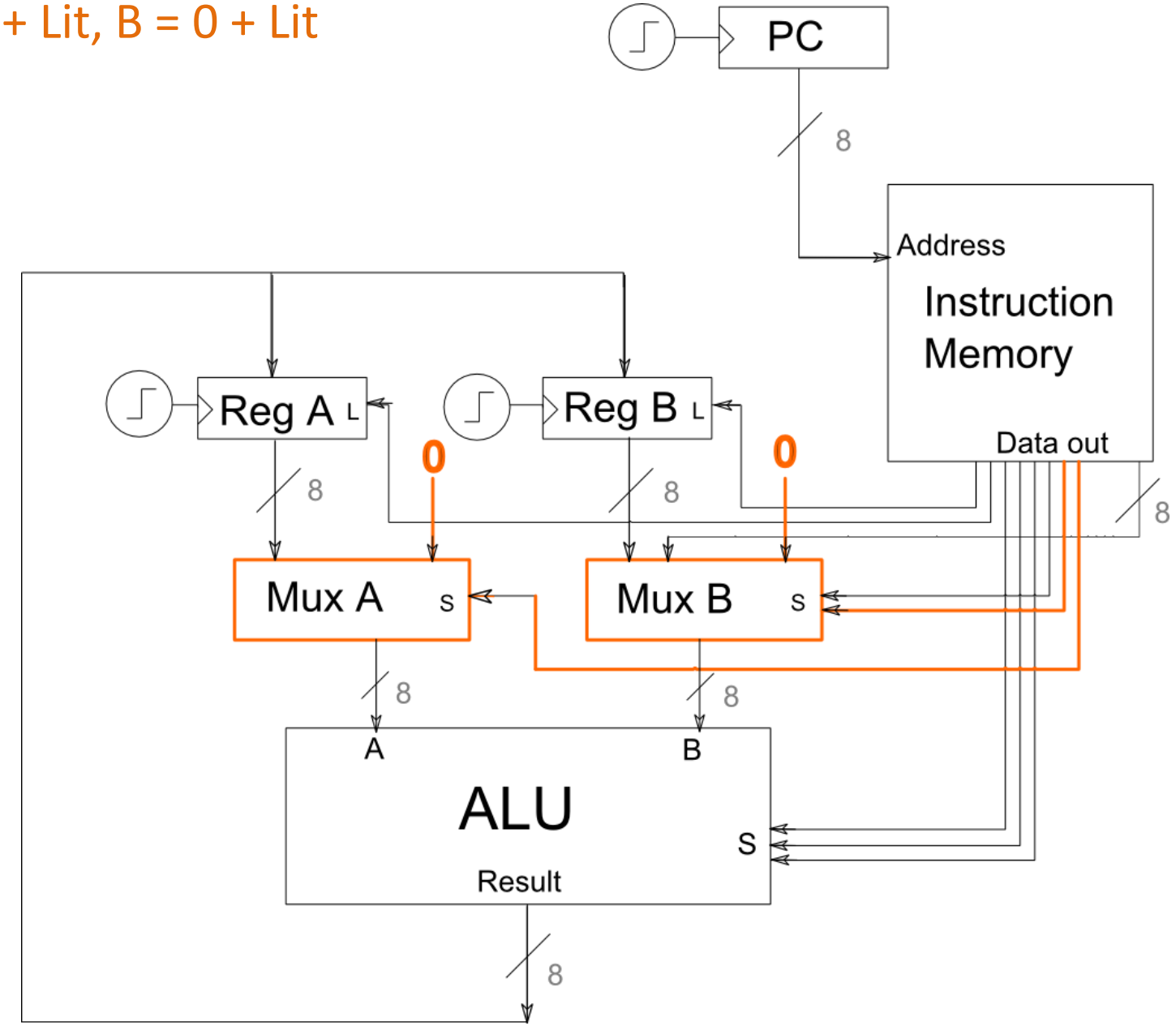
Literales aumentan poder expresivo

Ej.: A = A + 5



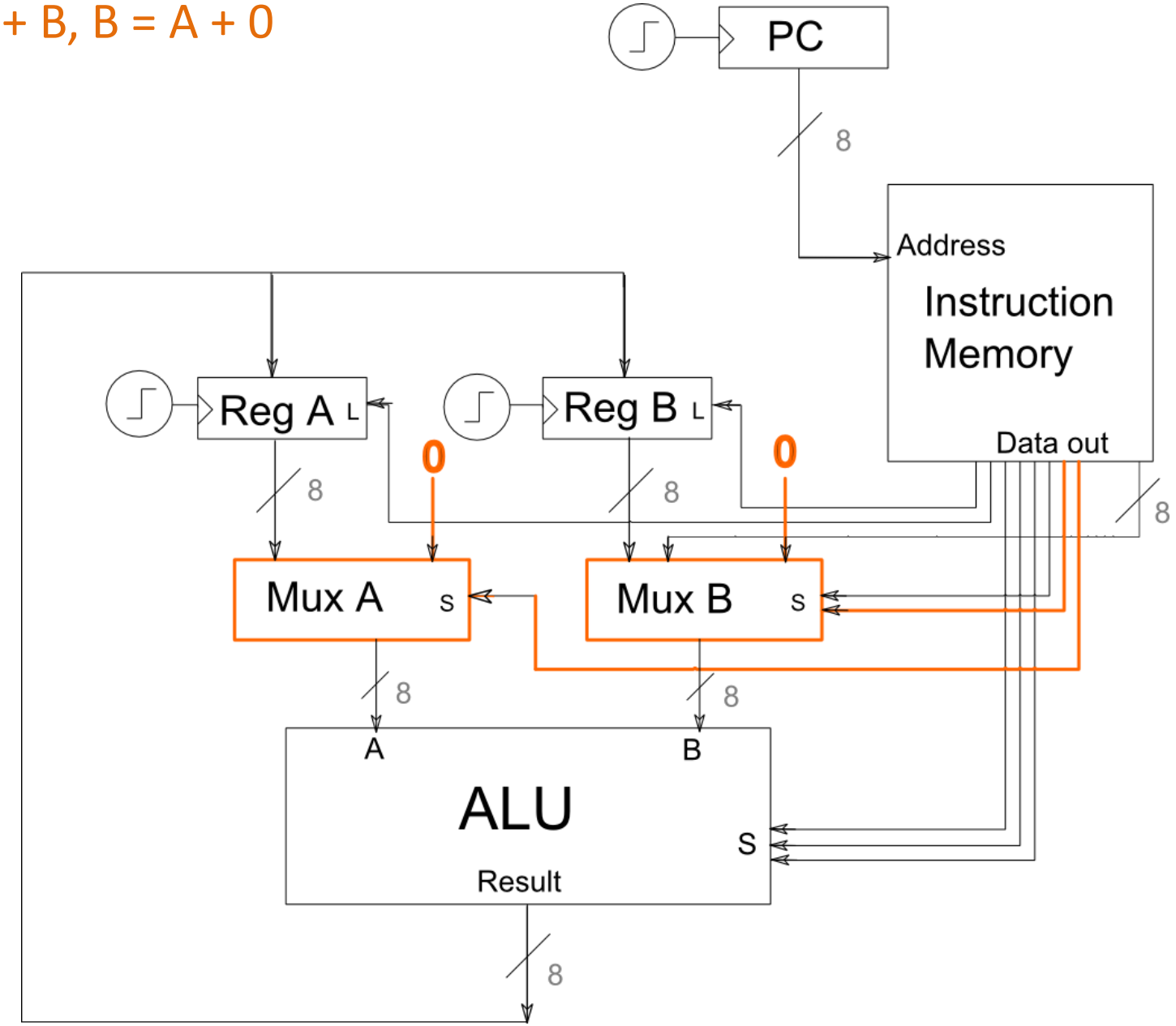
Ahora podemos inicializar registros

Ej.: A = 0 + Lit, B = 0 + Lit



Y también copiarlos

Ej.: A = 0 + B, B = A + 0



Tenemos palabras de control de 8 bits.

Podemos escribir en la memoria cualquier cosa (esto es malo)

Pero tenemos solo 28 instrucciones válidas.

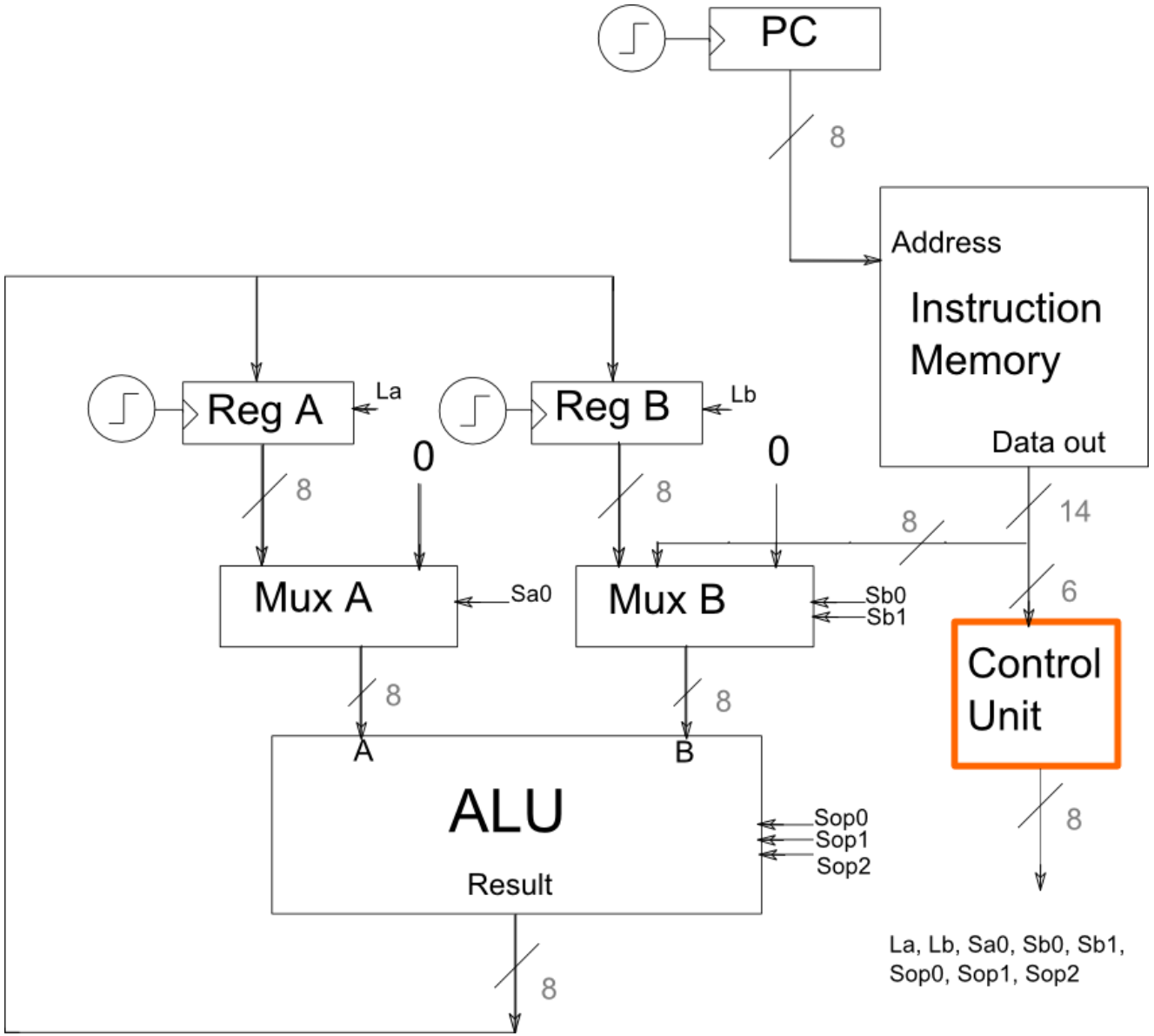
La	Lb	Sa0	Sb0	Sb1	Sop2	Sop1	Sop0	Operación
1	0	1	0	0	0	0	0	A=B
0	1	0	1	1	0	0	0	B=A
1	0	1	0	1	0	0	0	A=Lit
0	1	1	0	1	0	0	0	B=Lit
1	0	0	0	0	0	0	0	A=A+B
0	1	0	0	0	0	0	0	B=A+B
1	0	0	0	1	0	0	0	A=A+Lit
1	0	0	0	0	0	0	1	A=A-B
0	1	0	0	0	0	0	1	B=A-B
1	0	0	0	1	0	0	1	A=A-Lit
1	0	0	0	0	0	1	0	A=A and B
0	1	0	0	0	0	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

Solucionamos esto usando opcodes.

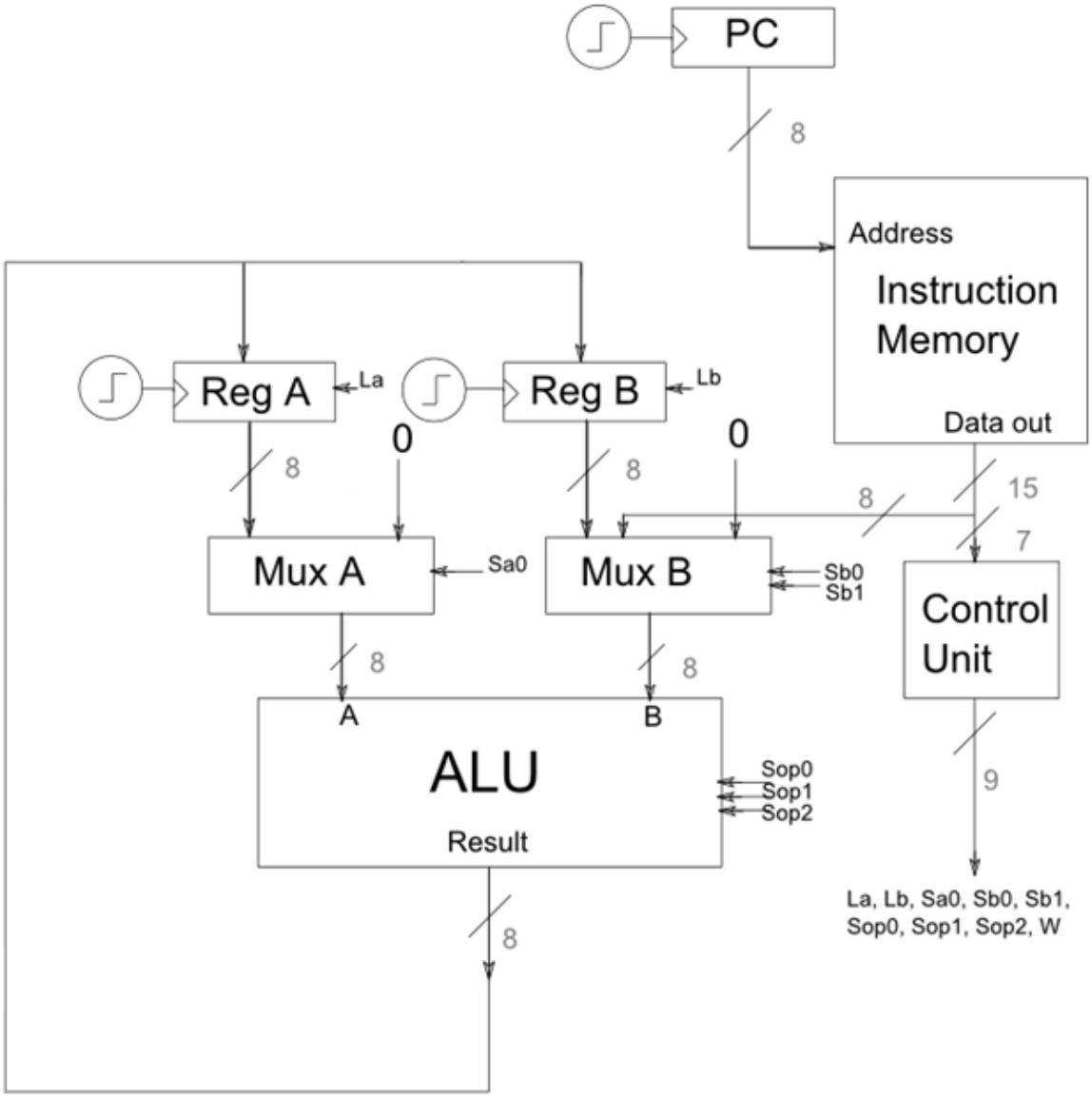
Cada uno se asocia a una instrucción.

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

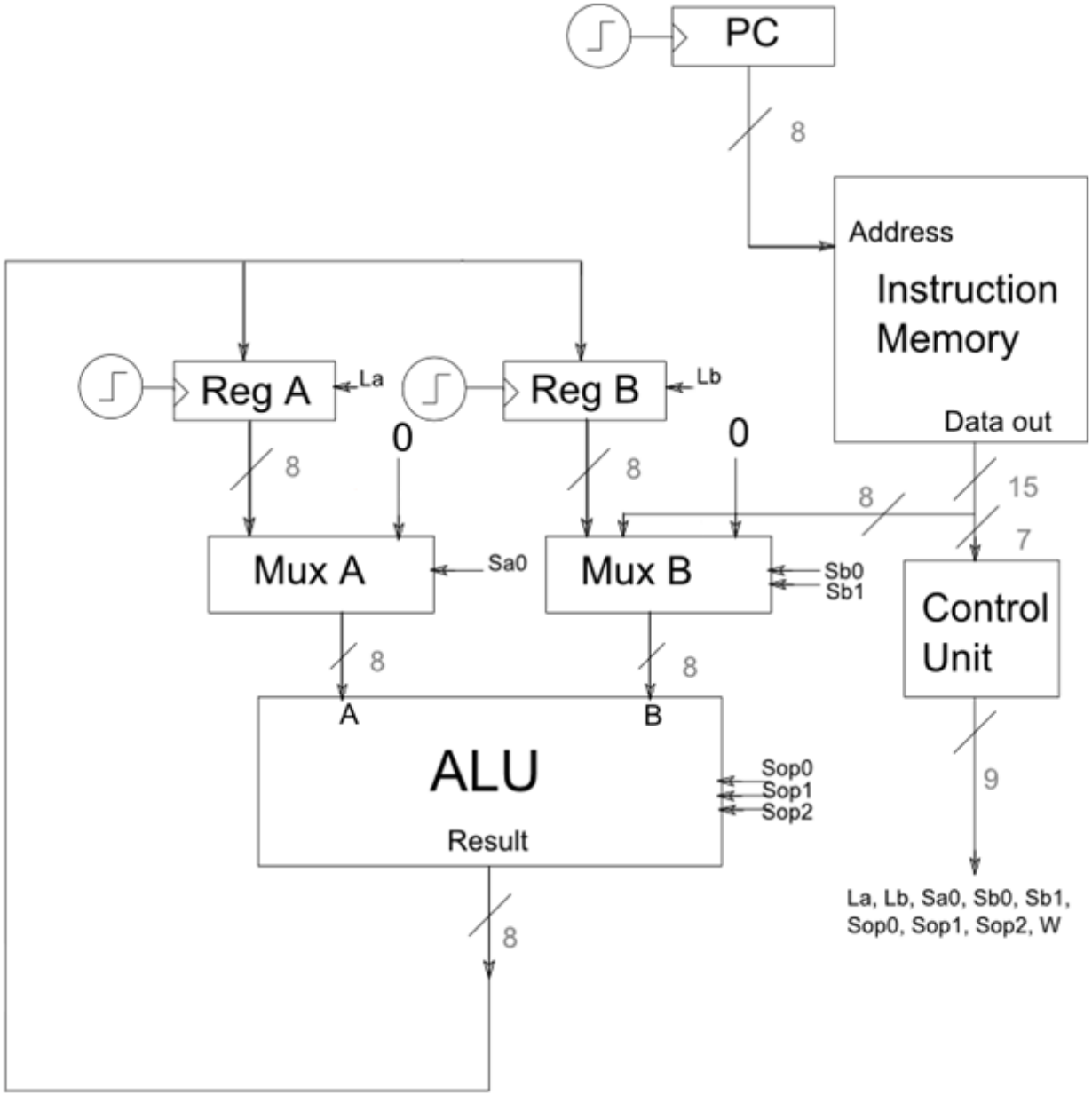
La CU traduce opcodes a señales de control y se implementa con una ROM (pero puede ser más compleja)



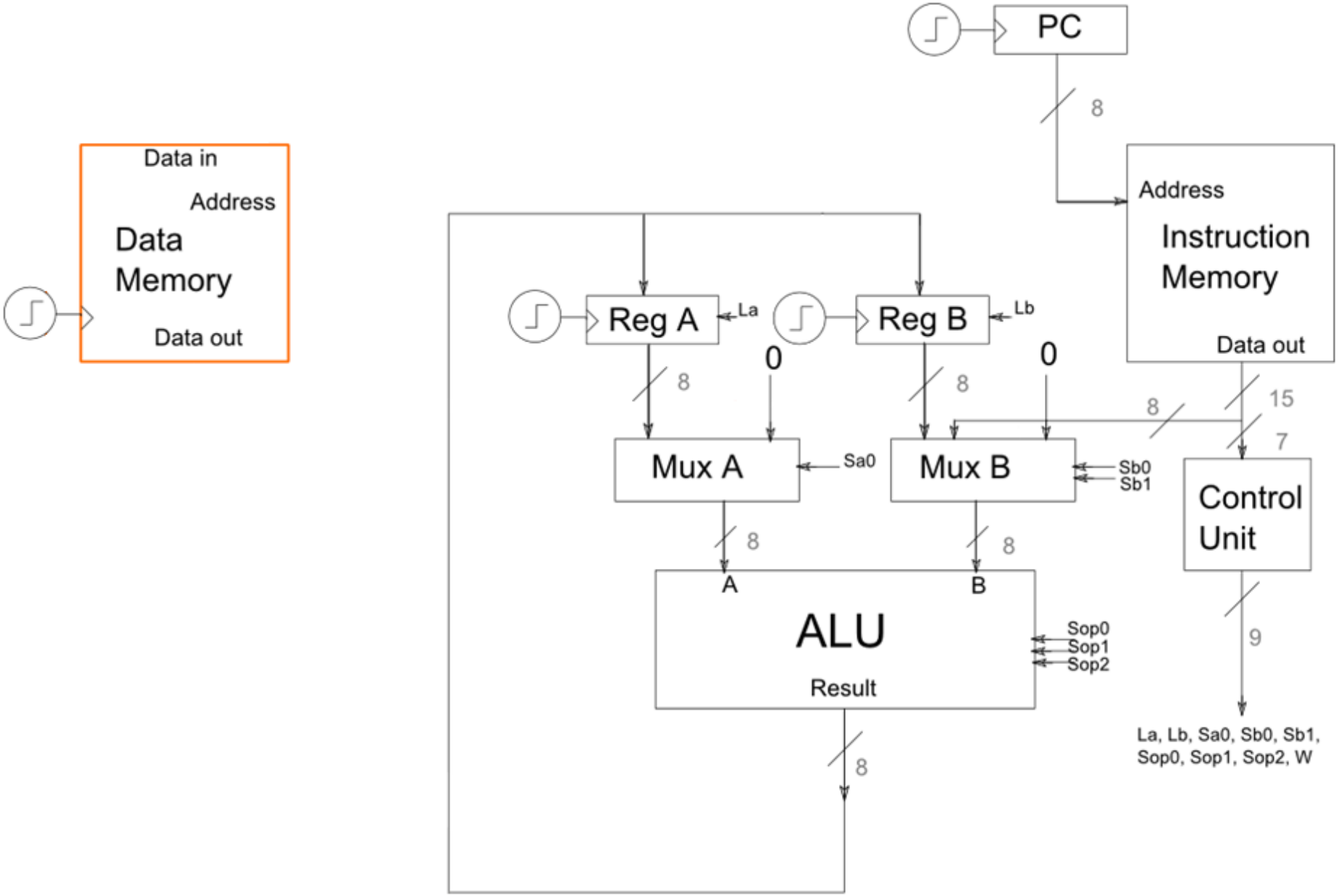
Aún necesitamos una manera para almacenar una mayor cantidad de datos



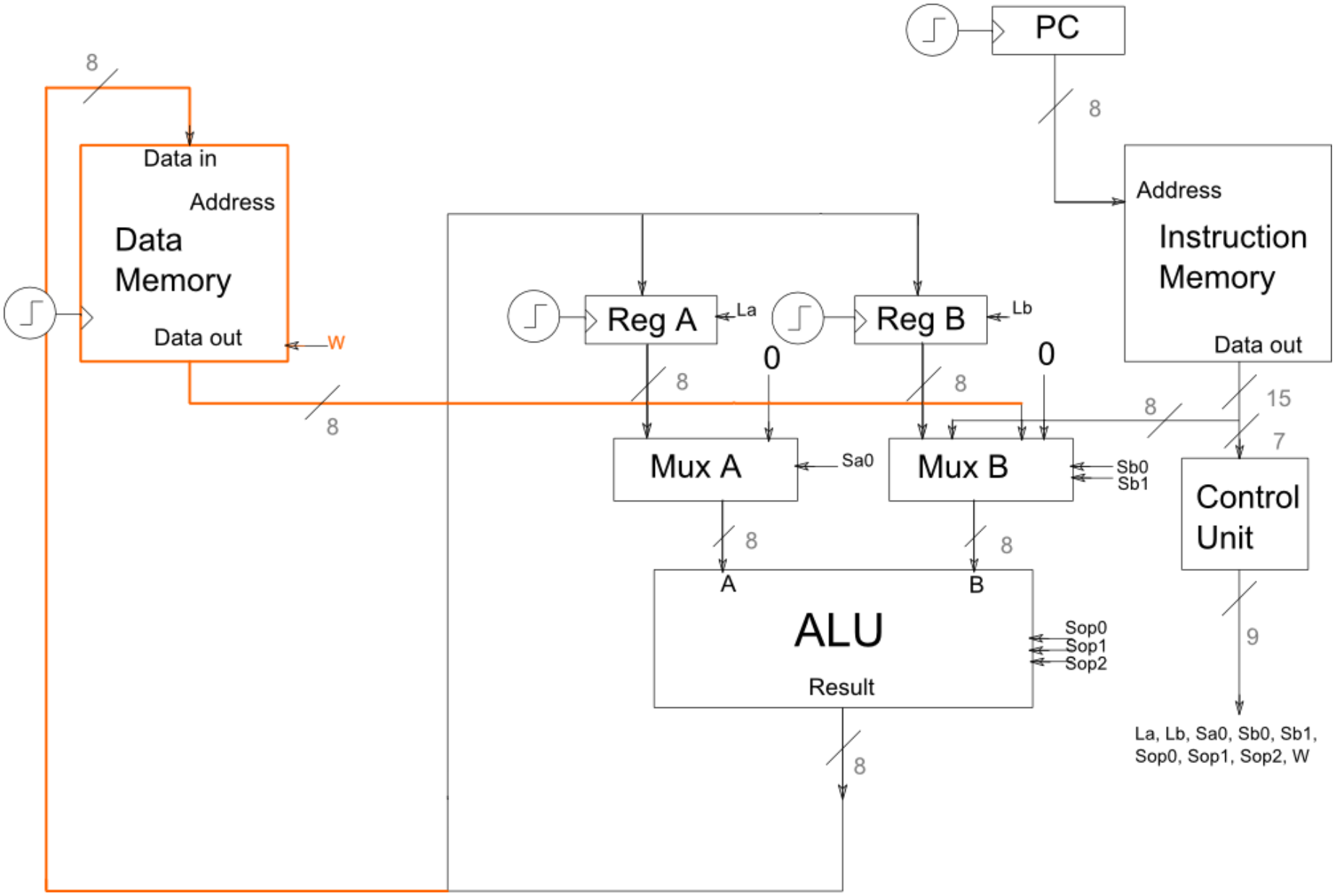
¿Donde podemos agregar una memoria?



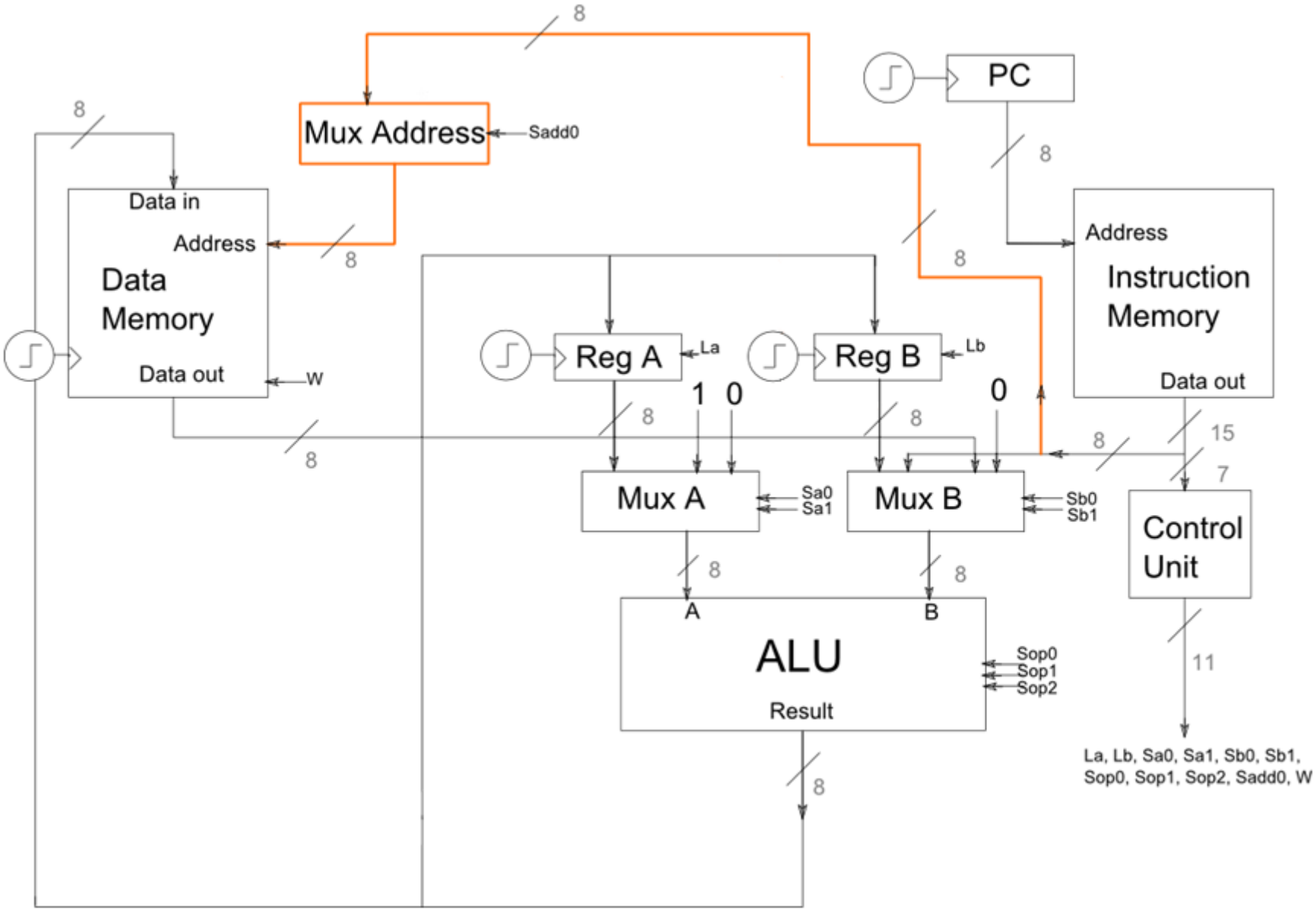
¿Donde podemos agregar una memoria?



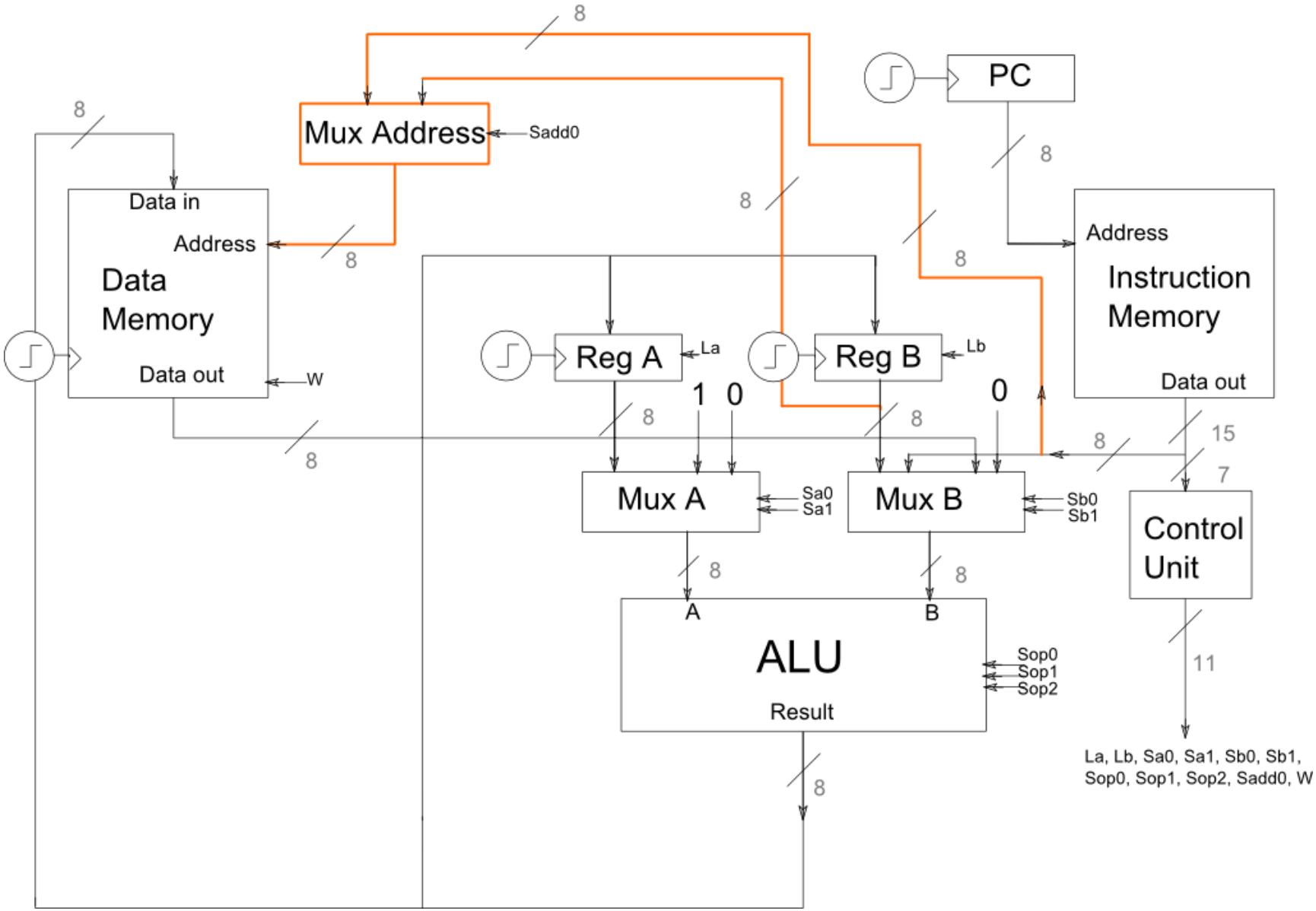
Ahora podemos almacenar variables



Podemos direccionar la memoria mediante un literal…
(direccionamiento directo)



…o también mediante el registro B (direccionamiento indirecto por registro)



¿Cómo podemos darle órdenes (programar) a la máquina programable?

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

Assembly nos permite trabajar de forma más natural y con menor redundancia

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

Ocupando el assembly recién descrito, escriba un programa que realice las siguientes operaciones:

- Cargar los valores 5 y 6 en los registros A y B respectivamente
- Guardar en A la suma de los valores almacenados en A y B
- Guardar en A la resta de los valores almacenados en A y B
- Setear en 0 el valor de B

Necesitamos dar soporte para "variables" y direccionamiento en el assembly

- Separamos el código en 2 segmentos: datos y código
- En el de datos definimos las "variables" indicando su contenido y un *label* para facilitar el acceso
- En el de código, va lógicamente el código

Dirección	Label	Instrucción/Dato
	DATA:	
0x00	var0	Dato 0
0x01	var1	Dato 1
0x02	var2	Dato 2
0x03		Dato 3
0x04		Dato 4
	CODE:	
0x00		Instrucción 0
0x01		Instrucción 1
0x02		Instrucción 2
0x03		Instrucción 3
0x04		Instrucción 4

Necesitamos dar soporte para "variables" y direccionamiento en el assembly

- Para el direccionamiento, usaremos ()
- (*label*) para el direccionamiento directo

  MOV A, (*var*) significa guardar en A el contenido de la dirección de memoria indicada por el label *var*

- (B) para el indirecto

  MOV A, (B) significa guardar en A el contenido de la dirección de memoria indicada en B

Instrucción	Operandos	Operación	Ejemplo de uso
MOV	A,(Dir)	A=Mem[Dir]	MOV A,(var1)
	B,(Dir)	B=Mem[Dir]	MOV B,(var2)
	(Dir),A	Mem[Dir]=A	MOV (var1),A
	(Dir),B	Mem[Dir]=B	MOV (var2),B
	A,(B)	A=Mem[B]	-
	B,(B)	B=Mem[B]	-
	(B),A	Mem[B]=A	-
ADD	A,(Dir)	A=A+Mem[Dir]	ADD A,(var1)
	A,(B)	A=A+Mem[B]	-
	(Dir)	Mem[Dir]=A+B	ADD (var1)
SUB	A,(Dir)	A=A-Mem[Dir]	SUB A,var1
	A,(B)	A=A-Mem[B]	-
	(Dir)	Mem[Dir]=A-B	SUB (var1)
AND	A,(Dir)	A=A and Mem[Dir]	AND A,(var1)
	A,(B)	A=A and Mem[B]	-
	(Dir)	Mem[Dir]=A and B	-
OR	A,(Dir)	A=A or Mem[Dir]	OR A,(var1)
	A,(B)	A=A or Mem[B]	-
	(Dir)	Mem[Dir]=A or B	OR (var1)
NOT	(Dir)	Mem[Dir]=not A	NOT (var1)
XOR	A,(Dir)	A=A xor Mem[Dir]	XOR A,(var1)
	A,(B)	A=A xor Mem[B]	-
	(Dir)	Mem[Dir]=A xor B	XOR (var1)
SHL	(Dir)	Mem[Dir]=shift left A	SHL (var1)
SHR	(Dir)	Mem[Dir]=shift right A	SHR (var1)
INC	B	B=B+1	-
INC	(B)	(B)=(B)+1	-
INC	(Dir)	(Dir)=(Dir)+1	INC (var)

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

IIC2343 – Arquitectura de Computadores

Programabilidad Parte 2 - Computador Básico

Profesor: Hans Löbel