Saltos y Subrutinas

IIC2343 - Arquitectura de Computadores

Nicolás Elliott B. (nicolas.elliott@uc.cl)

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
ESCUELA DE INGENIERÍA
PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

(II/2019)

¿Qué necesita?

Programa:

```
public static void promedio(){
        int[] arreglo = new int[]{6,4,2,3,5};
        int n = 5;
        int i = 0;
        float promedio = 0.0;

        while(i < n){
                promedio += arreglo[i];
                i ++;
        }
        promedio /=  n;
        System.out.println(promedio);
}
```

Saltos

Salto incondicional

Secuencia limitada de Fibonacci

Dirección	Instrucción	Operandos	A	B
0x00	MOV	A,0	**0**	?
0x01	MOV	B,1	0	**1**
0x02	ADD	A,B	**1**	1
0x03	ADD	B,A	1	**2**
0x04	ADD	A,B	**3**	2
0x05	ADD	B,A	3	**5**
0x06	ADD	A,B	**8**	5
0x07	ADD	B,A	8	**13**

Salto incondicional

Secuencia «infinita» de Fibonacci

Dirección	Instrucción	Operandos
0x00	MOV	A,0
0x01	MOV	B,1
0x02	ADD	A,B
0x03	ADD	B,A
0x04	JMP	0x02

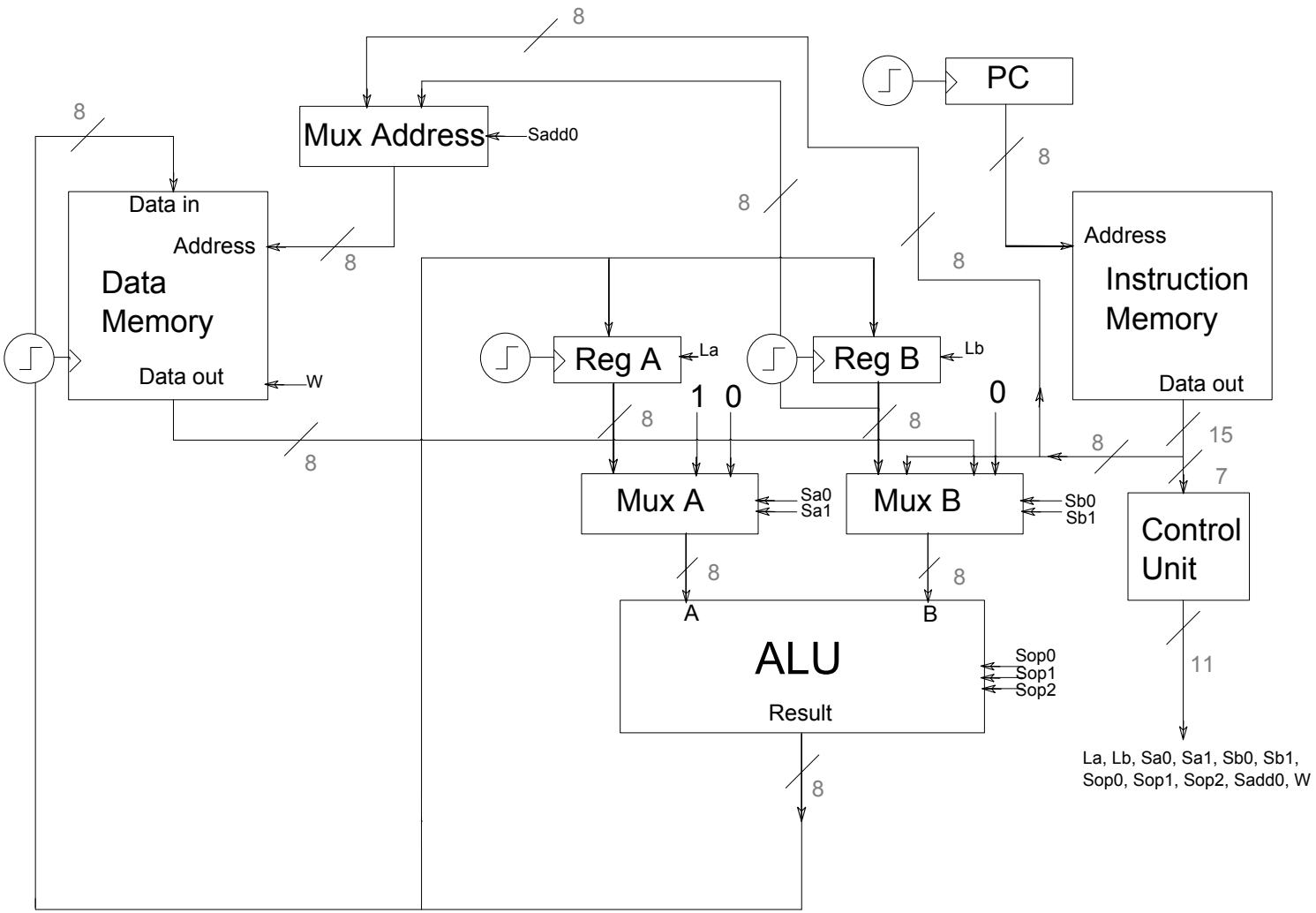
Salto incondicional

Secuencia «infinita» de Fibonacci con label

Dirección	Label	Instrucción	Operandos
0x00		MOV	A,0
0x01		MOV	B,1
0x02	start:	ADD	A,B
0x03		ADD	B,A
0x04		JMP	start

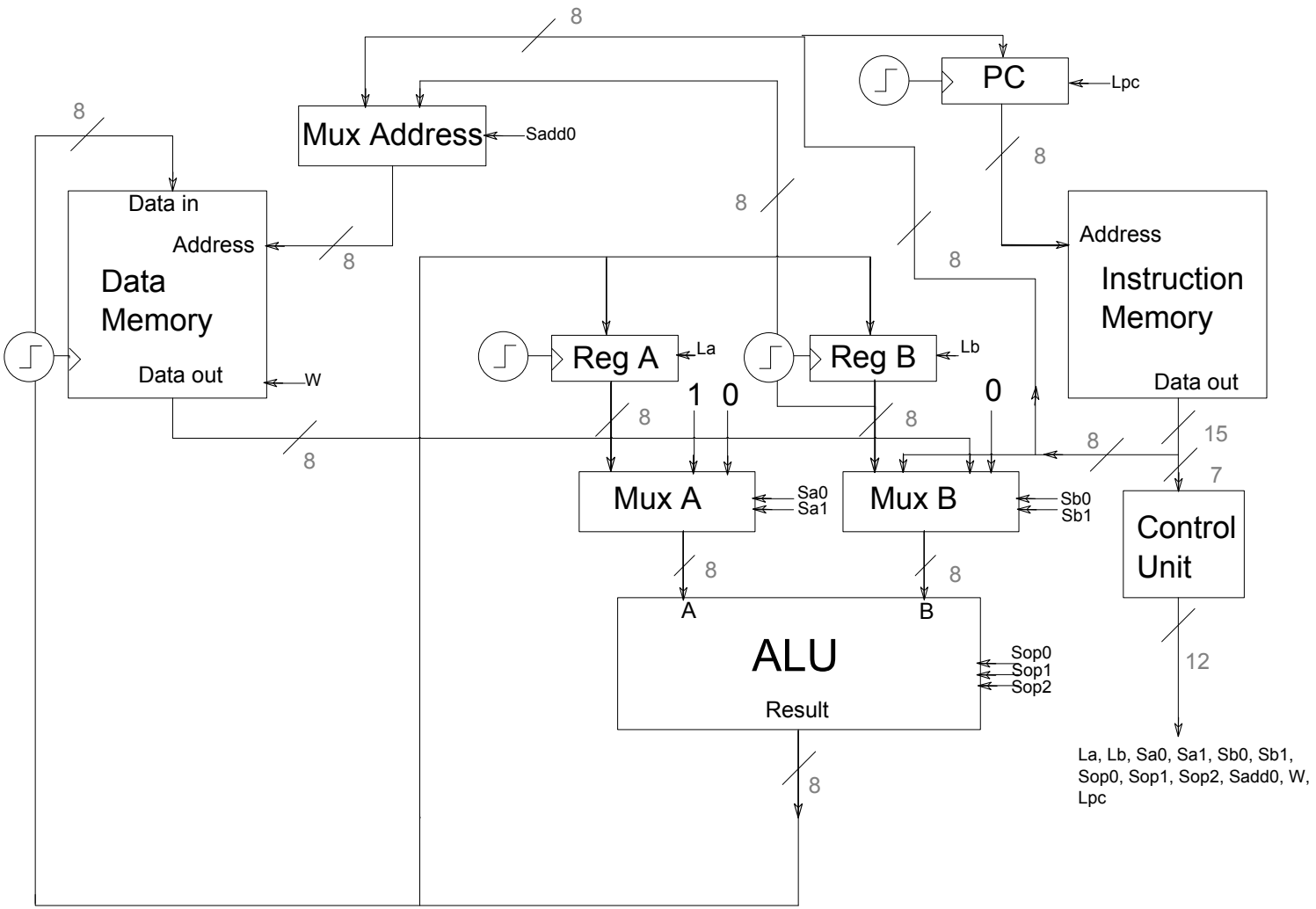
Salto incondicional

Computador con direccionamiento



Salto incondicional

Computador con salto incondicional



Salto condicional

Comparación

- $a == b$, $a != b$, $a > b$, $a < b$,$a \geq b$,$a \leq b$.
- $a - b == 0$, $a - b != 0$, $a - b > 0$, $a - b < 0$,$a - b \geq 0$,$a - b \leq 0$
- `CMP A,B` que ejecuta la resta entre los registros A y B y **no** almacena el resultado.

Salto condicional

Instrucciones de salto para comparación

- `JEQ`: «Jump equal», cuando $a == b$ $(Z = 1)$.
- `JNE`: «Jump not equal», cuando $a! = b$ $(Z = 0)$.
- `JGT`: «Jump greater than», cuando $a > b (Z = 0, N = 0)$.
- `JLT`: «Jump less than», cuando $a < b$ $(N = 1)$.
- `JGE`: «Jump greater or equal than», cuando $a \geq b$ $(N = 0)$.
- `JLE`: «Jump less or equal than», cuando $a \leq b$ $(Z = 1, N = 1)$.

Salto condicional

Condition codes para comparación

- Zero (**Z**): El código de condición cero se puede obtener a partir del resultado de la ALU, haciendo un **nor** entre todos los bits.
- Negative (**N**): El código de condición negativo se puede obtener tomando el bit más significativo del resultado.

Salto condicional

Instrucciones de salto condicional por excepción

Instrucción	Operandos	Operación	Condiciones	Ejemplo de uso
CMP	A,B	A-B		
	A,Lit	A-Lit		CMP A,0
JEQ	Dir	PC = Dir	Z=1	JEQ label
JNE	Dir	PC = Dir	Z=0	JNE label
JGT	Dir	PC = Dir	N=0 y Z=0	JGT label
JLT	Dir	PC = Dir	N=1	JLT label
JGE	Dir	PC = Dir	N=0	JGE label
JLE	Dir	PC = Dir	Z=1 o N=1	JLE label

Salto condicional

Excepción

- Pérdida de un bit, **carry**
- Cambio de inesperado de signo, **overflow**

Salto condicional

Instrucciones de salto por excepción

- `JCR`: «Jump carry», cuando ocurra un carry ($C = 1$).
- `JOV`: «Jump overflow», cuando ocurra un overflow ($V = 1$).

Salto condicional

Condiciones de overflow

Operación	A	B	Resultado	Ejemplo (1 byte)
$A + B$	≥ 0	≥ 0	< 0	$127 + 4 = -125$
$A + B$	< 0	< 0	≥ 0	$-127 + -4 = 125$
$A - B$	≥ 0	< 0	< 0	$127 - -4 = -125$
$A - B$	< 0	≥ 0	≥ 0	$-127 - 4 = 125$

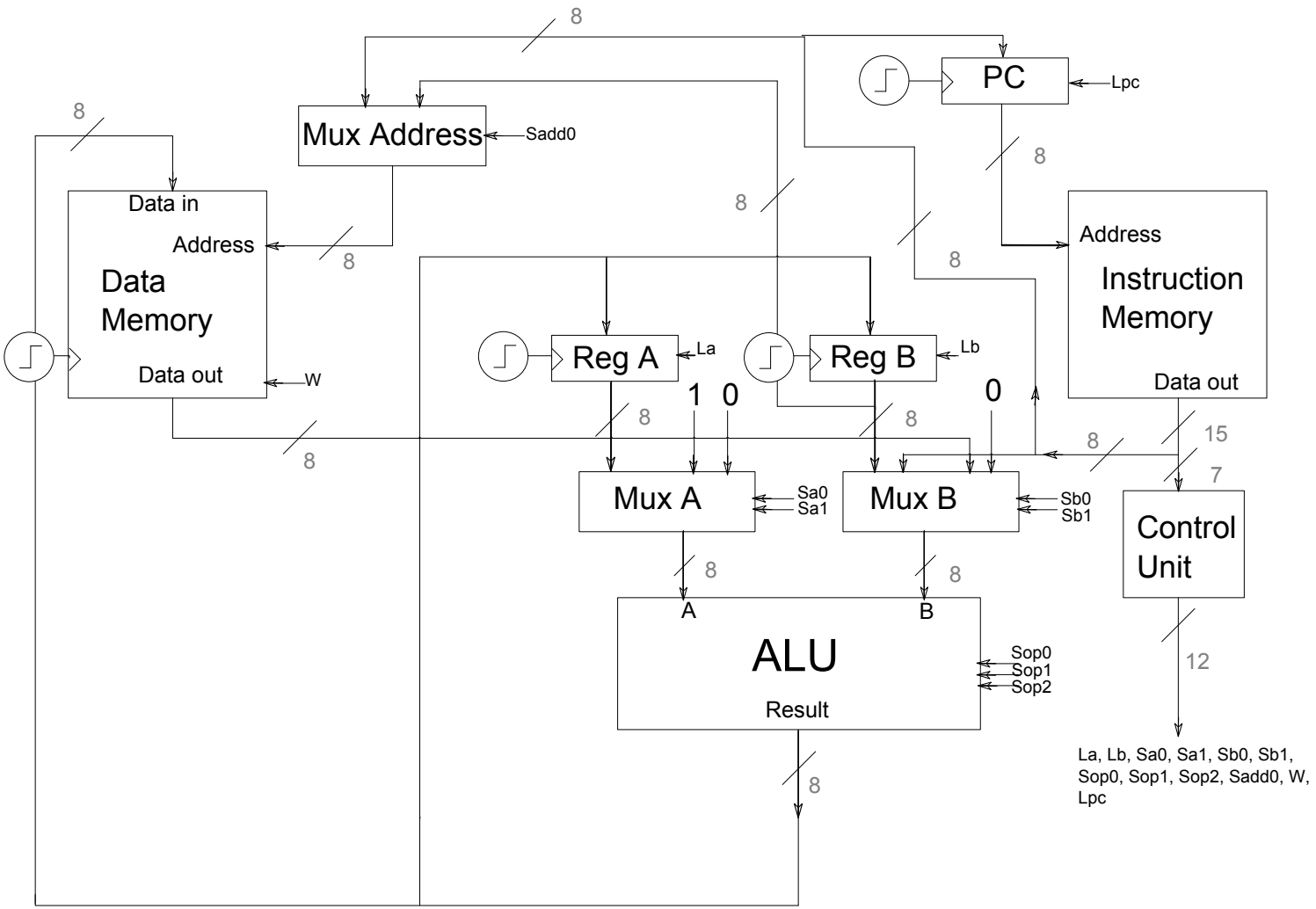
Salto condicional

Instrucciones de salto condicional por excepción

Instrucción	Operandos	Operación	Condiciones	Ejemplo de uso
JCR	Dir	PC = Dir	C=1	JCR label
JOV	Dir	PC = Dir	V=1	JOV label

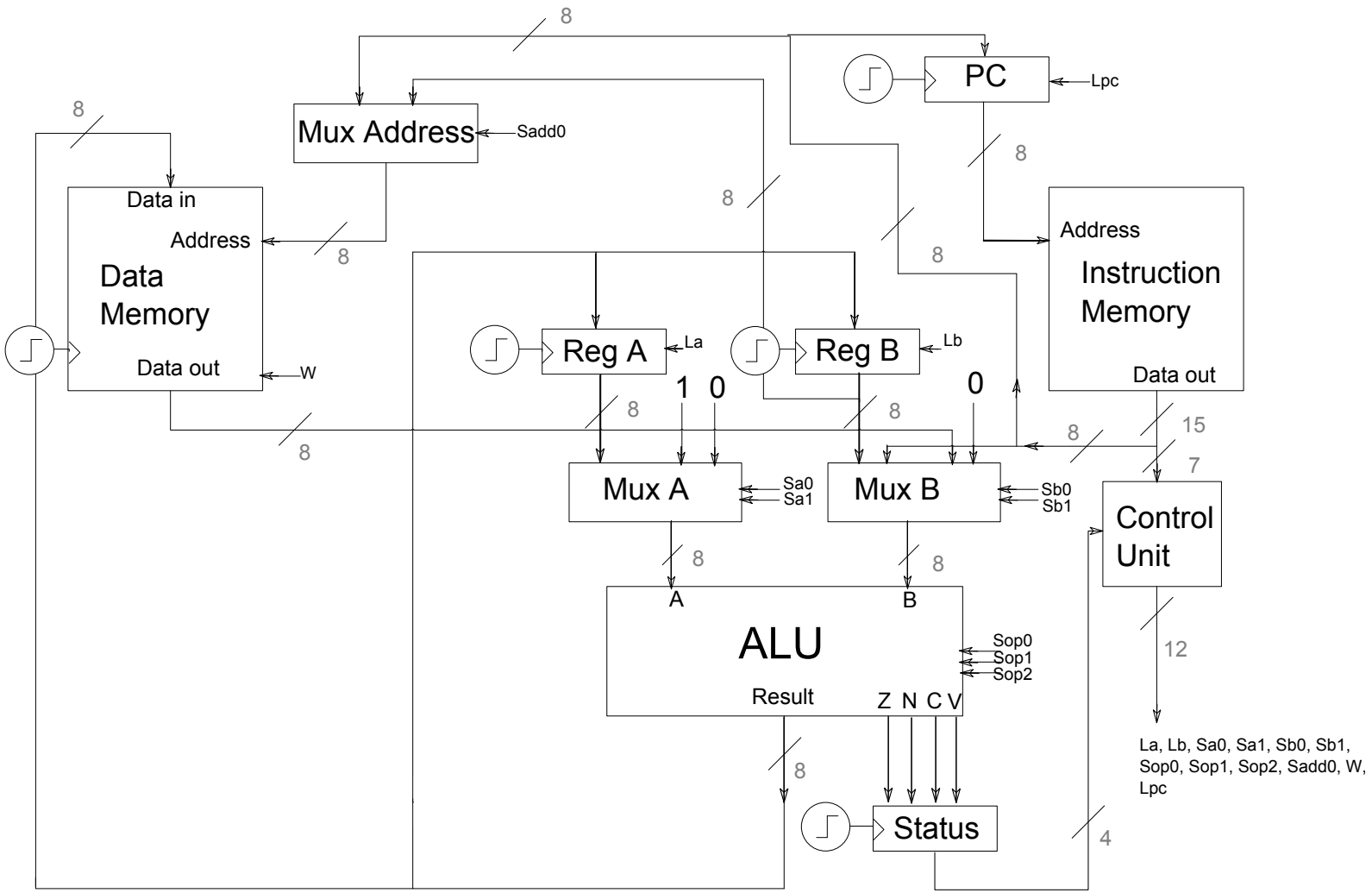
Saltos

Computador con salto incondicional



Saltos

Computador con salto condicional



Salto condicional

Instrucciones de salto del computador básico.

Instrucción	Operandos	Operación	Condiciones	Ejemplo de uso
CMP	A,B	A-B		
	A,Lit	A-Lit		CMP A,0
JMP	Dir	PC = Dir		JMP label
JEQ	Dir	PC = Dir	Z=1	JEQ label
JNE	Dir	PC = Dir	Z=0	JNE label
JGT	Dir	PC = Dir	N=0 y Z=0	JGT label
JLT	Dir	PC = Dir	N=1	JLT label
JGE	Dir	PC = Dir	N=0	JGE label
JLE	Dir	PC = Dir	Z=1 o N=1	JLE label
JCR	Dir	PC = Dir	C=1	JCR label
JOV	Dir	PC = Dir	V=1	JOV label

¿Qué necesita?

Programa:

```
public static void promedio(){
        int[] arreglo = new int[]{6,4,2,3,5};
        int n = 5;
        int i = 0;
        float promedio = 0.0;

        while(i < n){
                promedio += arreglo[i];
                i ++;
        }
        promedio /=  n;
        System.out.println(promedio);
}
```

Funciones

Son:

- Invocables
- Reciben Parámetros
- Retornan algo
- Re utilizables

Subrutinas

Subrutinas

Instrucciones

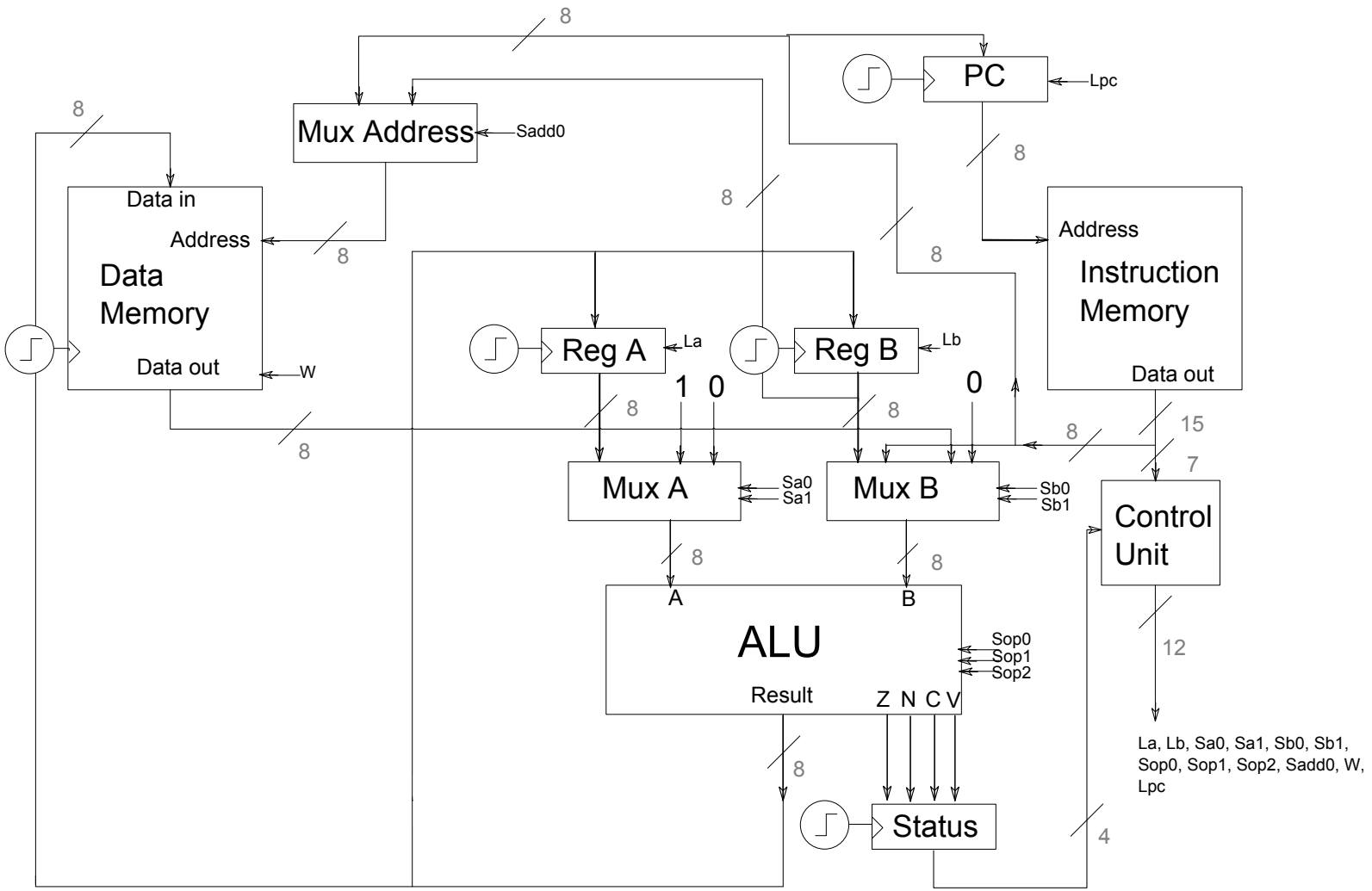
- CALL label: Salta a un punto arbitrario del código.
- RET:Regresa a la sección del código de origen.

Subrutinas

- Parámetros: Memoria o Registros.
- Retorno: : Memoria o Registros.

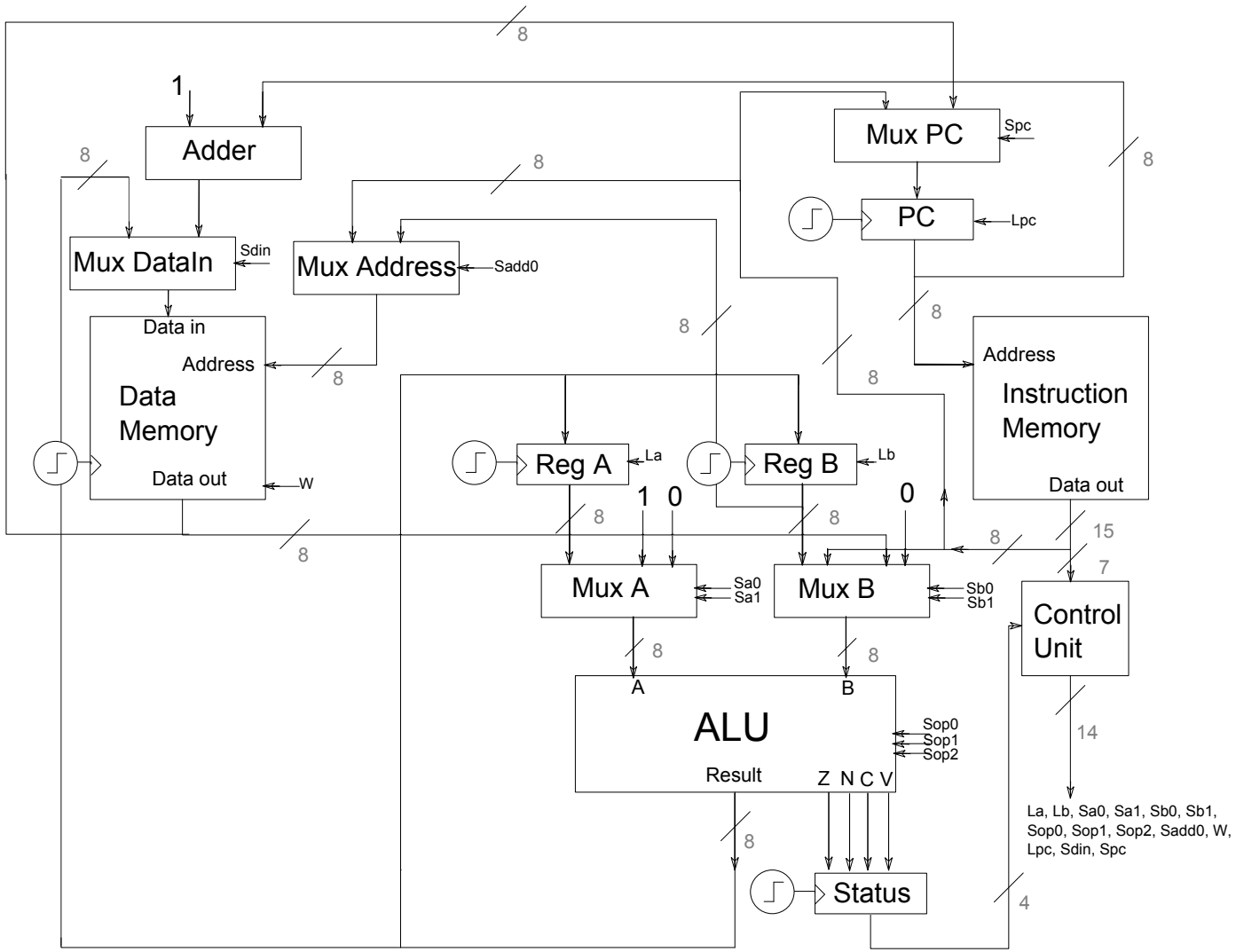
Subrutinas

Computador sin subrutinas



Subrutinas

Computador con almacenamiento del program counter

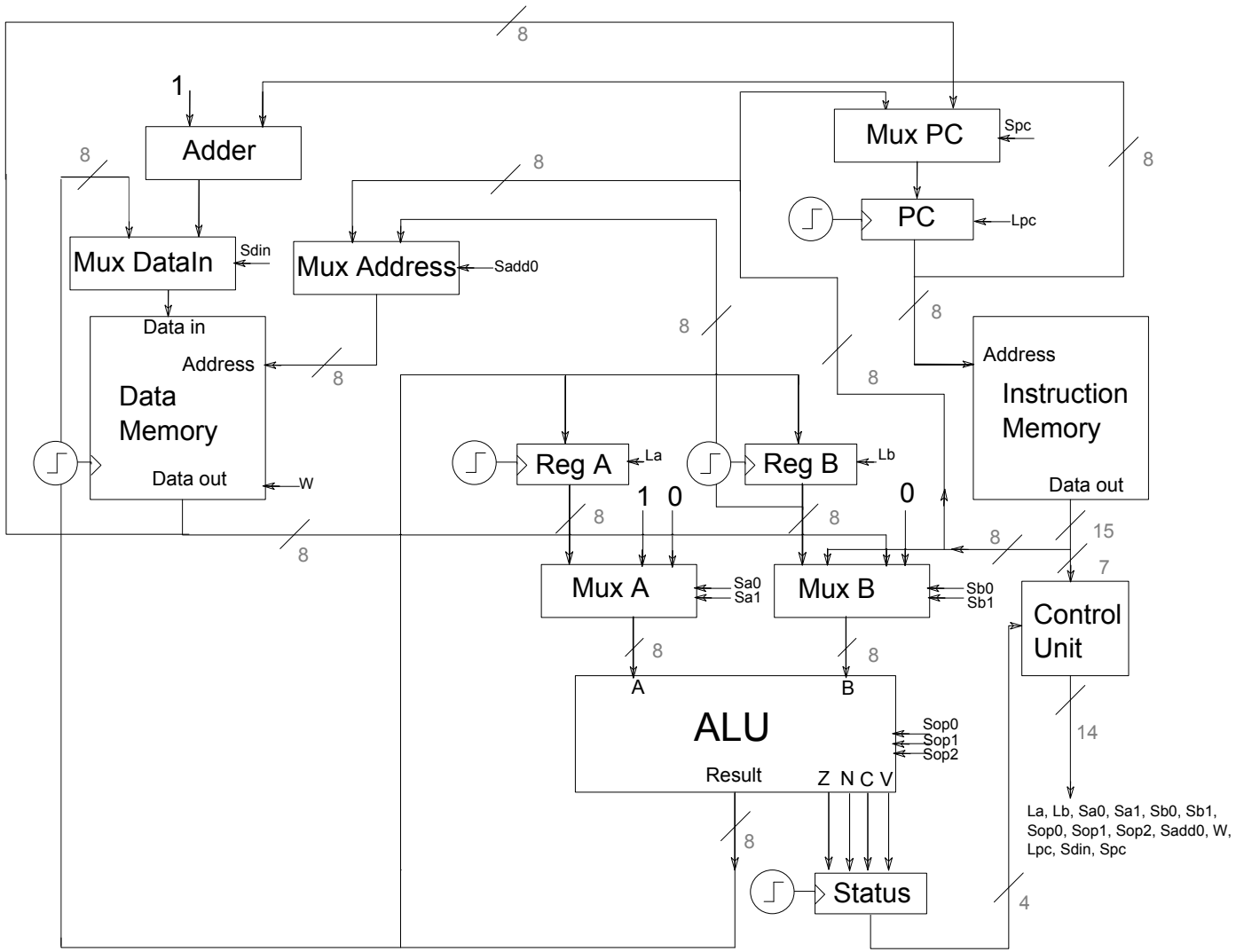


Subrutinas

- ¿ En que parte de la memoria guardamos el PC?
- Última dirección de memoria: 255

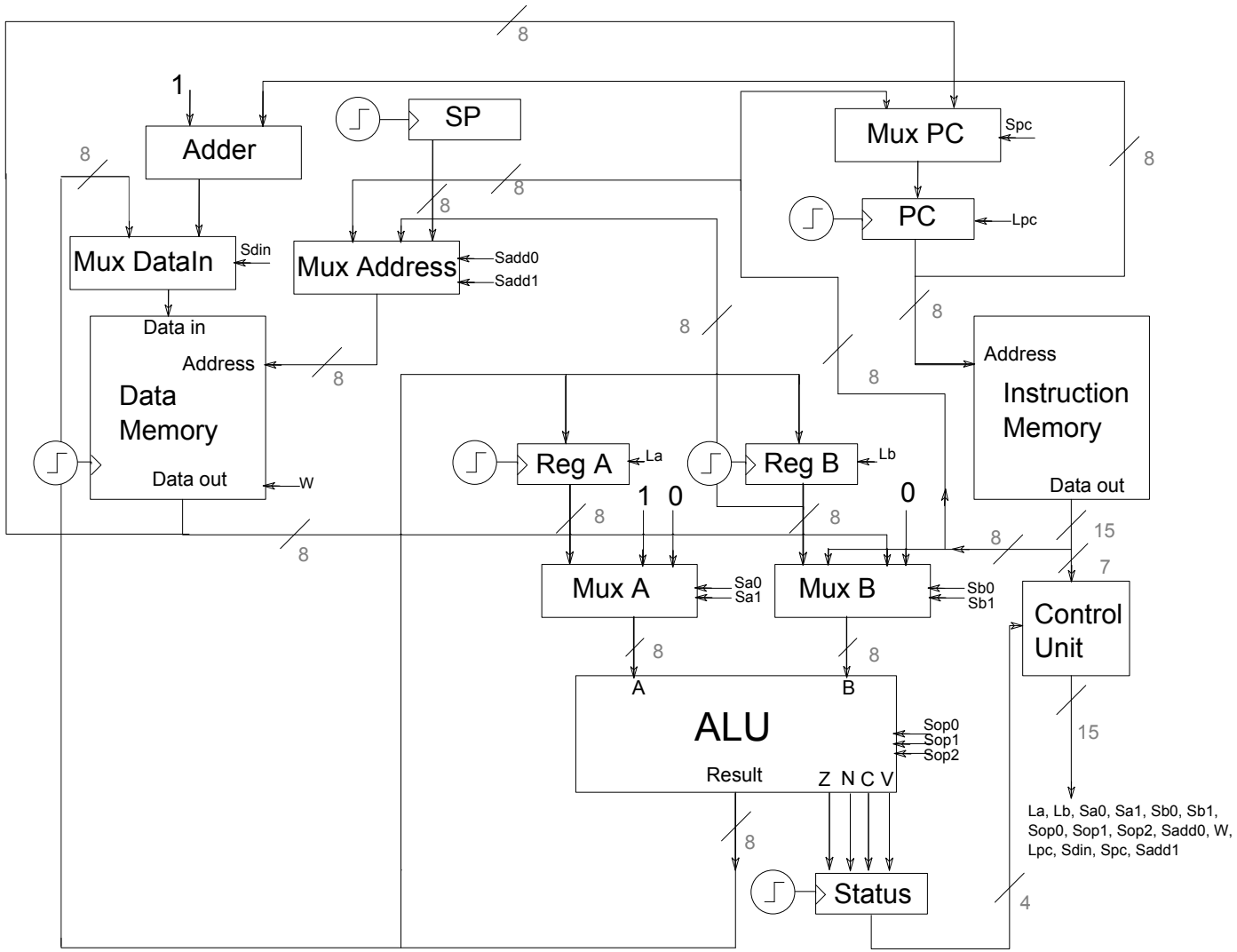
Subrutinas

Computador con almacenamiento del program counter



Subrutinas

Computador con subrutina



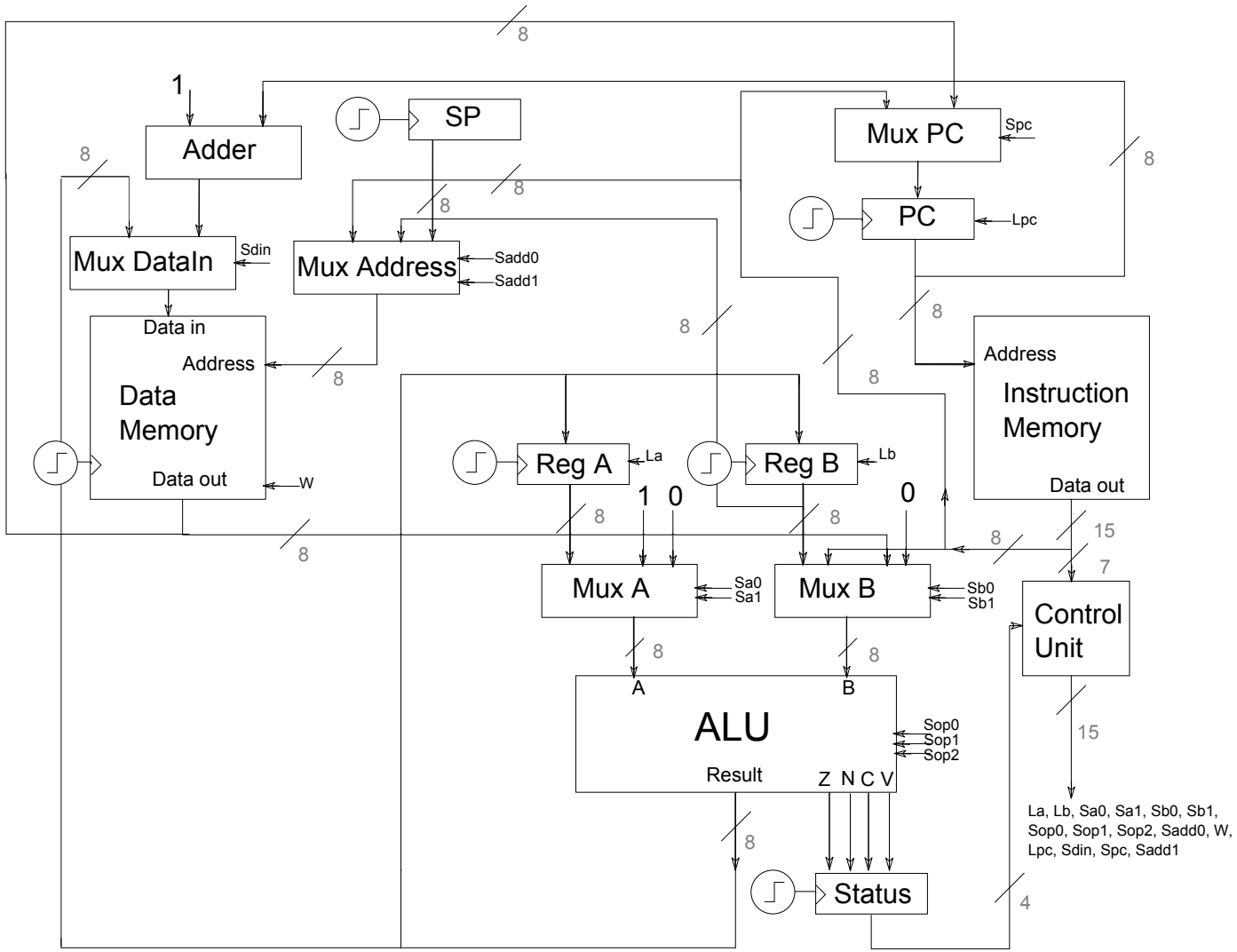
Subrutinas

- ¿Podemos anidar Subrutinas?

Stack

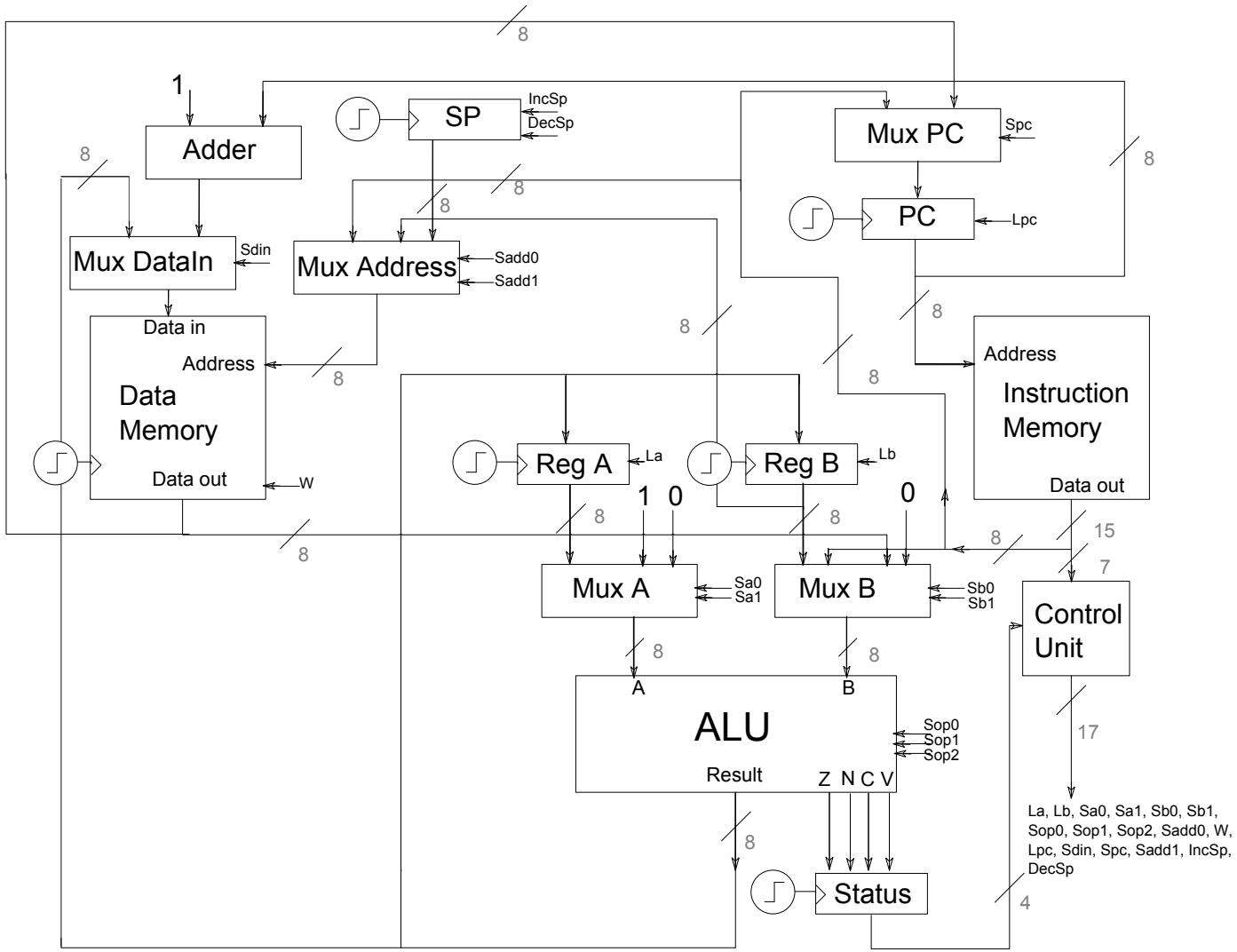
Stack

Computador con subrutina



Stack

Computador con subrutinas



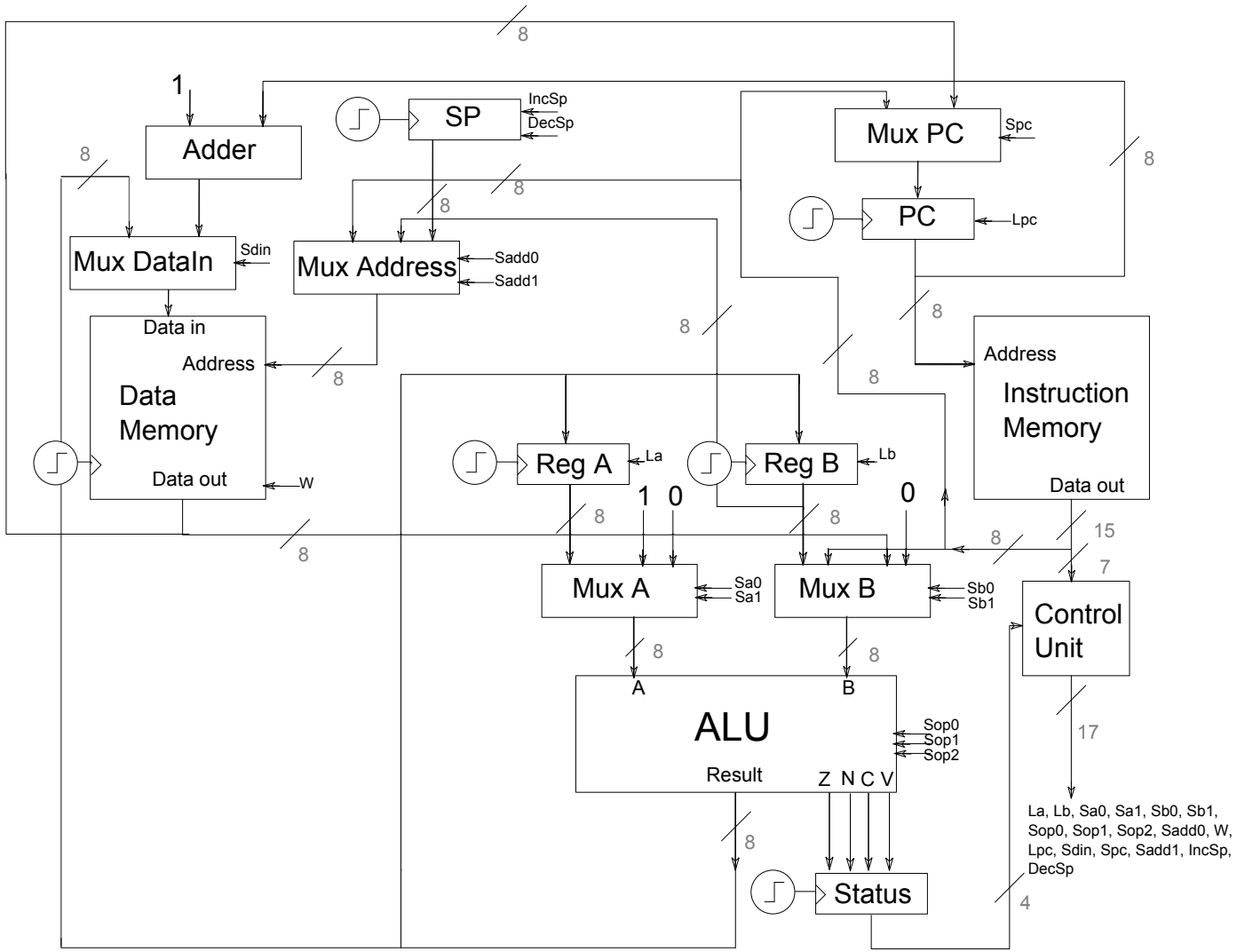
Stack

Instrucciones

- CALL label:
  1. Guardar PC + 1 en la posición actual del SP
  2. Decrementar en 1 el SP
  3. Guardar la dirección de la subrutina en PC
- RET:
  1. Incrementar en 1 el SP
  2. Guardar el valor de memoria apuntado por el SP incrementado en PC

Stack

Computador con subrutinas



Stack

De uso general

- PUSH Reg: Almacena un registro en el stack
- POP Reg: Recupera un valor del stack y lo almacena en un registro

Salto condicional

Instrucciones de salto del computador básico.

Instrucción	Operandos	Operación	Condiciones	Ejemplo de uso
CALL	Dir	Mem[SP] = PC + 1, SP- -, PC = Dir		CALL func
RET		SP++		-
		PC = Mem[SP]		-
PUSH	A	Mem[SP] = A, SP- -		-
PUSH	B	Mem[SP] = B, SP- -		-
POP	A	SP++		-
		A = Mem[SP]		-
POP	B	SP++		-
		B = Mem[SP]		-

Stack

Computador con almacenamiento del program counter