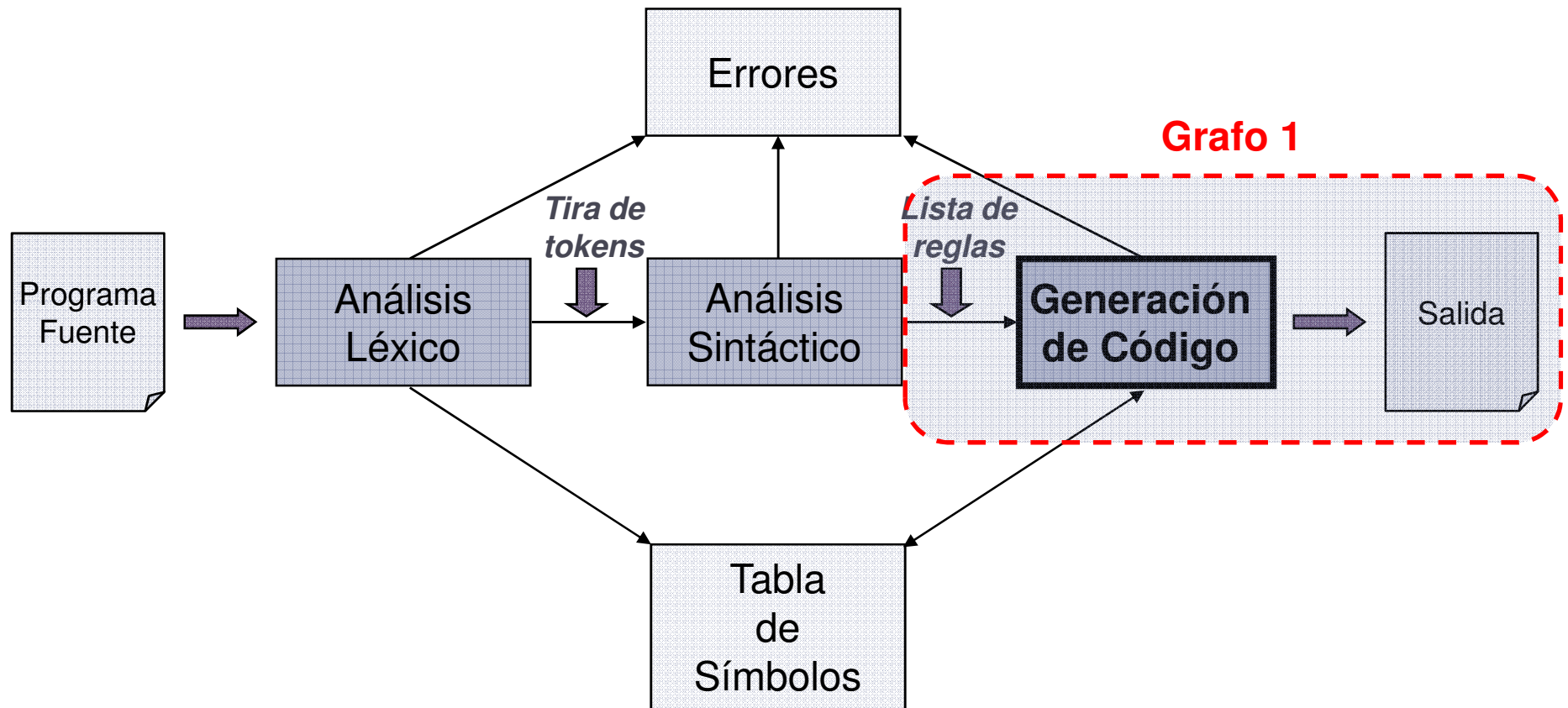


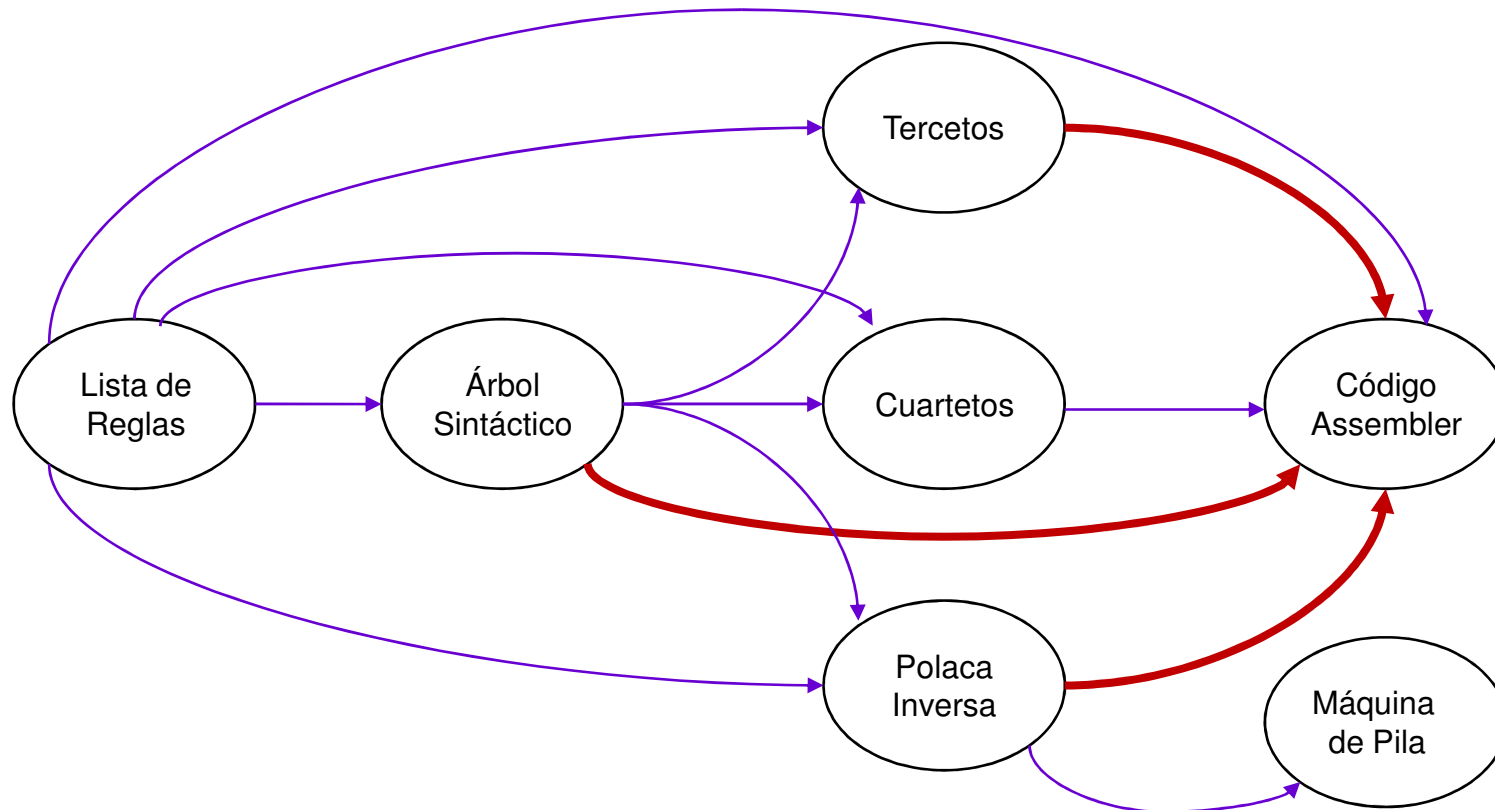
Diseño de Compiladores I

Generación de Código

Fases de la Compilación



Generación de Código



Sentencias de Control

Árbol Sintáctico → Assembler

Polaca Inversa → Assembler

Tercetos → Assembler

Código Assembler para Sentencia IF

IF ($a - b > c + 1$)
THEN $a := b + c$;
ELSE $a := b - c$;

- ▶ MOV R1, _a
- ▶ SUB R1, _b
- ▶ MOV R2, _c
- ▶ ADD R2, 1
- ▶ CMP R1, R2

JLE Label1

- ▶ MOV R1, _b
- ▶ ADD R1, _c
- ▶ MOV _a, R1

JMP Label2

Label1:

- ▶ MOV R1, _b
- ▶ SUB R1, _c
- ▶ MOV, _a, R1

Label2:

SENTENCIA SIGUIENTE AL IF

¿Dónde van las bifurcaciones?

¿Dónde van las etiquetas?

Código Assembler para Sentencia WHILE

```
WHILE ( a + b <= c - d ) {  
    a := b + c;  
    c := d - 3;}
```

Label1:

- ▶ MOV R1, _a
- ▶ ADD R1, _b
- ▶ MOV R2, _c
- ▶ SUB R2, _d
- ▶ CMP R1, R2

JG Label2

- ▶ MOV R1, _b
- ▶ ADD R1, _c
- ▶ MOV _a, R1
- ▶ MOV R1, _d
- ▶ SUB R1, 3
- ▶ MOV _c, R1

JMP Label1

Label2:

SENTENCIA SIGUIENTE AL WHILE

¿Dónde van las bifurcaciones?

¿Dónde van las etiquetas?

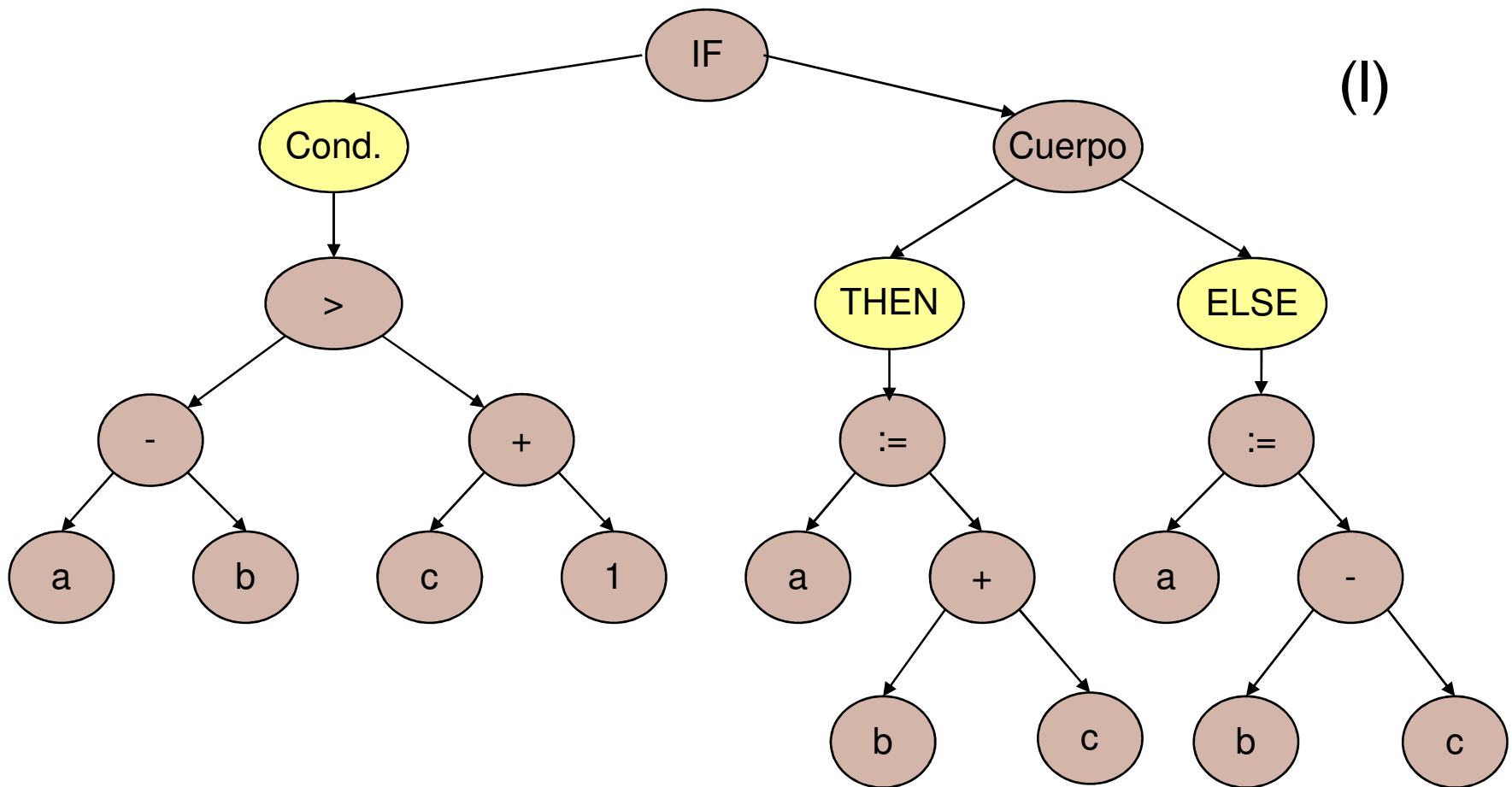
Árbol Sintáctico → ASSEMBLER

Sentencia IF

Generación de Código

Árbol Sintáctico → Assembler

IF (a - b > c + 1) THEN a := b + c; ELSE a := b - c;



(I)

Generación de Código

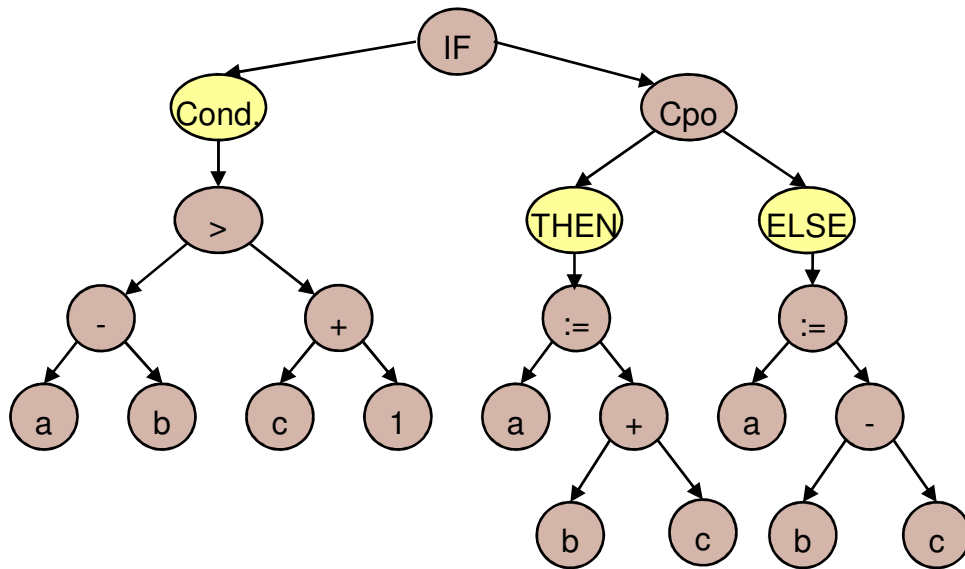
Árbol Sintáctico → Assembler

IF (a - b > c + 1)
THEN a := b + c;
ELSE a := b - c;

- ▶ MOV RI, _a
- ▶ SUB RI, _b

L - L - L - L
O - L - L - L
O - L - L - L

(I)



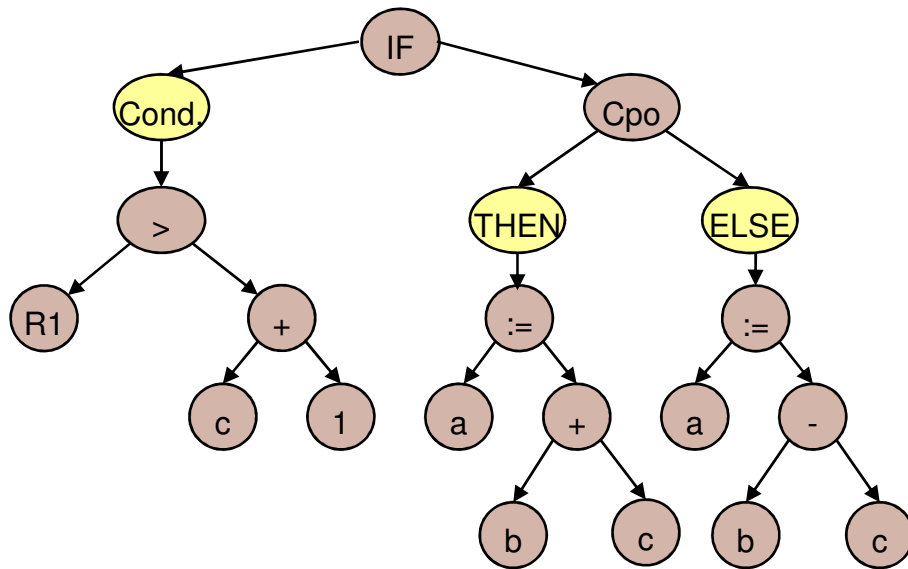
Generación de Código

Árbol Sintáctico → Assembler

IF (a - b > c + 1)
THEN a := b + c;
ELSE a := b - c;

- ▶ MOV R1, _a L - L - L - L
- ▶ SUB R1, _b O - L - L - L
- ▶ MOV R2, _c O - O - L - L
- ▶ ADD R2, I O - O - L - L

(I)



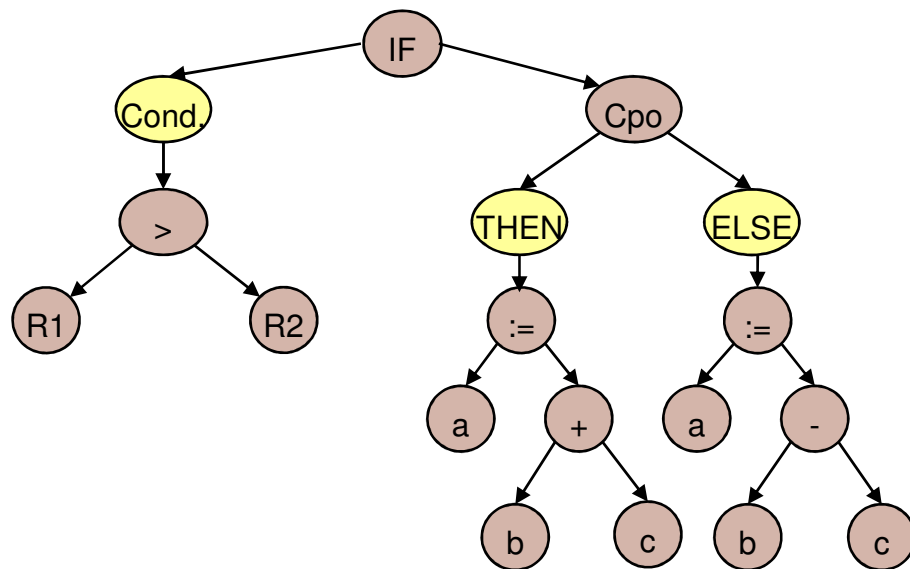
Generación de Código

Árbol Sintáctico → Assembler

IF (a - b > c + 1)
THEN a := b + c;
ELSE a := b - c;

- ▶ MOV R1, _a L - L - L - L
- ▶ SUB R1, _b O - L - L - L
- ▶ MOV R2, _c O - O - L - L
- ▶ ADD R2, 1 O - O - L - L
- ▶ CMP R1, R2 L - L - L - L

(I)

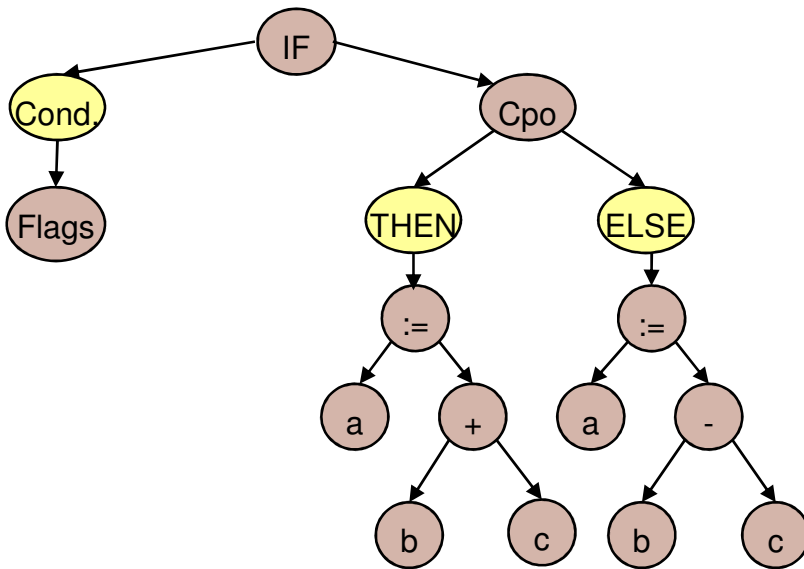


Generación de Código

Árbol Sintáctico → Assembler

IF (a - b > c + 1)
 THEN a := b + c ;
 ELSE a := b - c ;

(I)



- ▶ MOV R1, _a L - L - L - L
- ▶ SUB R1, _b O - L - L - L
- ▶ MOV R2, _c O - O - L - L
- ▶ ADD R2, 1 O - O - L - L
- ▶ CMP R1, R2 L - L - L - L

(Si la raíz del subárbol es Cond)

JLE LabelI

Apilo LabelI

- ▶ MOV R1, _b O - L - L - L
- ▶ ADD R1, _c O - L - L - L
- ▶ MOV _a, R1 L - L - L - L



Árbol Sintáctico → Assembler

IF ($a - b > c + 1$)

THEN $a := b + c$;

ELSE $a := b - c$;

- ▶ MOV R1, _a
- ▶ SUB R1, _b
- ▶ MOV R2, _c
- ▶ ADD R2, 1
- ▶ CMP R1, R2

JLE Label1

- ▶ MOV R1, _b
- ▶ ADD R1, _c
- ▶ MOV _a, R1

JMP Label2

Label1:

- ▶ MOV R1, _b
- ▶ SUB R1, _c
- ▶ MOV, _a, R1

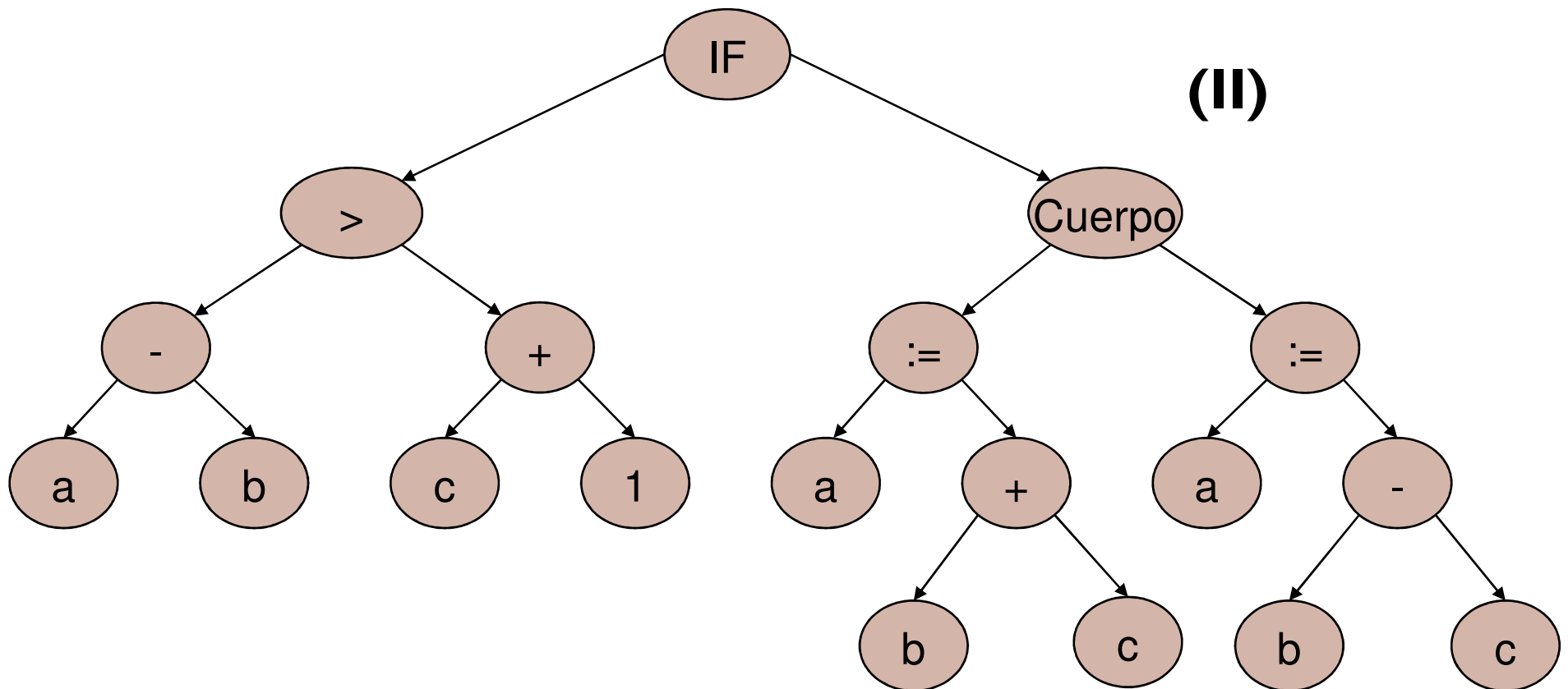
Label2:



Árbol Sintáctico → Assembler

Ejemplo:

IF (a - b > c + 1) THEN a := b + c; ELSE a := b - c;



Generación de Código

Árbol Sintáctico → Assembler

IF (a - b > c + 1)

THEN a := b + c;

ELSE a := b - c;

▶ MOV RI, _a

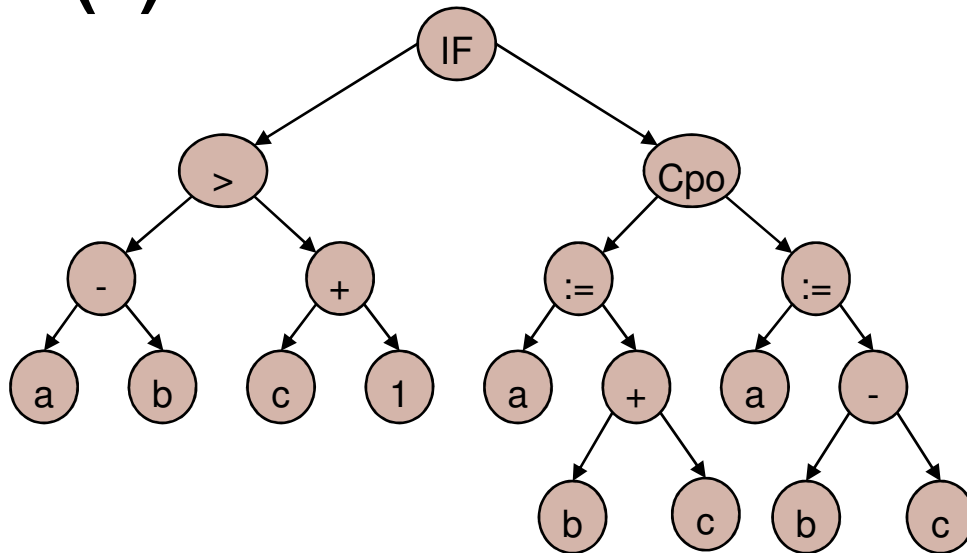
▶ SUB RI, _b

L - L - L - L

O - L - L - L

O - L - L - L

(II)



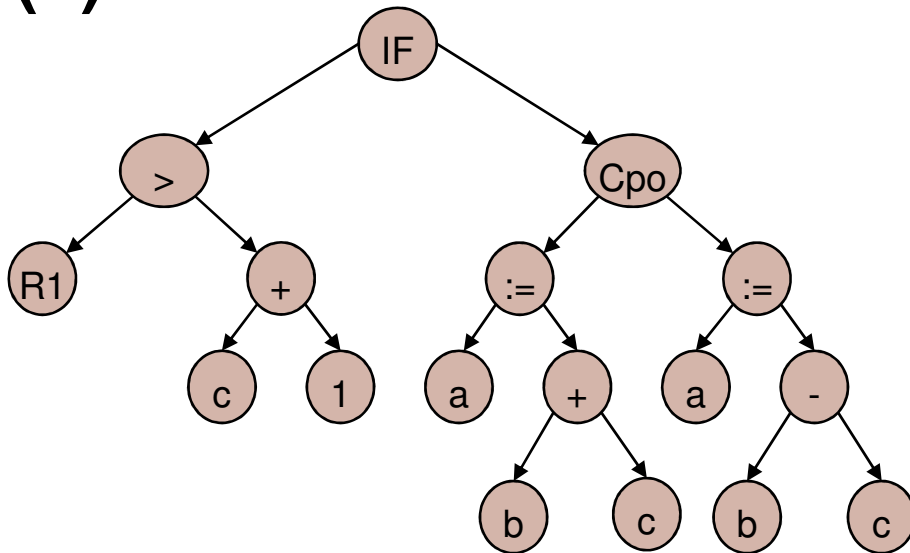
Generación de Código

Árbol Sintáctico → Assembler

IF (a - b > c + 1)
THEN a := b + c;
ELSE a := b - c;

- ▶ MOV R1, _a L - L - L - L
- ▶ SUB R1, _b O - L - L - L
- ▶ MOV R2, _c O - O - L - L
- ▶ ADD R2, 1 O - O - L - L

(II)



Generación de Código

Polaca Inversa → ASSEMBLER

Sentencia IF

Generación de Código

Código Assembler para Sentencia IF

IF ($a - b > c + 1$)
THEN $a := b + c$;
ELSE $a := b - c$;

- ▶ MOV R1, _a
- ▶ SUB R1, _b
- ▶ MOV R2, _c
- ▶ ADD R2, 1
- ▶ CMP R1, R2

JLE Label1

- ▶ MOV R1, _b
- ▶ ADD R1, _c
- ▶ MOV _a, R1

JMP Label2

Label1:

- ▶ MOV R1, _b
- ▶ SUB R1, _c
- ▶ MOV, _a, R1

Label2:

SENTENCIA SIGUIENTE AL IF

¿Dónde van las bifurcaciones?

¿Dónde van las etiquetas?

Polaca Inversa → Assembler

IF (a - b > c + l)

THEN a := b + c;

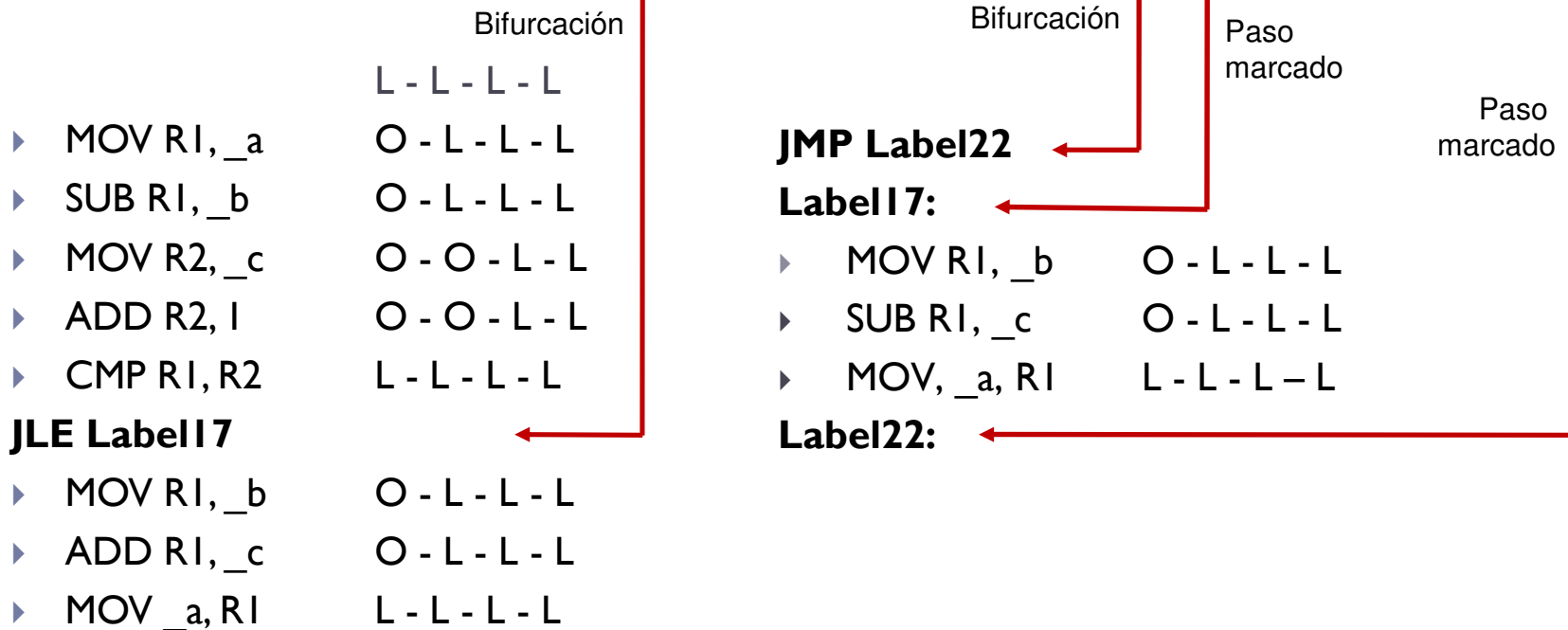
ELSE a := b - c;

Alternativa 1: 2 pasadas

➤ Pasada 1: Se marcan los destinos de las bifurcaciones

➤ Pasada 2: Se genera código

a	b	-	c	l	+	>	17	BF	a	b	c	+	:=	22	BI	a	b	c	-	:=	...
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22



Polaca Inversa → Assembler

IF (a – b > c + l)

THEN a := b + c;

ELSE a := b – c;

Alternativa 2.a:

- Durante la generación de la Polaca, se marcan los destinos de las bifurcaciones.
- En una sola pasada, se genera código.

a	b	-	c	l	+	>	17	BF	a	b	c	+	:=	22	Bl	a	b	c	-	:=	...
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22

- MOV R1, _a L - L - L - L
- SUB R1, _b O - L - L - L
- MOV R2, _c O - O - L - L
- ADD R2, 1 O - O - L - L
- CMP R1, R2 L - L - L - L

JLE Label17

- MOV R1, _b O - L - L - L
- ADD R1, _c O - L - L - L
- MOV _a, R1 L - L - L - L

JMP Label22

- Label17:** ←
- MOV R1, _b O - L - L - L
 - SUB R1, _c O - L - L - L
 - MOV, _a, R1 L - L - L - L

Label22: ←

Paso
marcado

Paso
marcado



Polaca Inversa → Assembler

IF ($a - b > c + 1$)

THEN $a := b + c$;

ELSE $a := b - c$;

- ▶ MOV R1, _a
- ▶ SUB R1, _b
- ▶ MOV R2, _c
- ▶ ADD R2, 1
- ▶ CMP R1, R2

JLE Label17

- ▶ MOV R1, _b
- ▶ ADD R1, _c
- ▶ MOV _a, R1

JMP Label22

Label17:

- ▶ MOV R1, _b
- ▶ SUB R1, _c
- ▶ MOV, _a, R1

Label22:



Polaca Inversa → Assembler

IF (a – b > c + l)

THEN a := b + c;

ELSE a := b – c;

Alternativa 2.b:

- Durante la generación de la Polaca, se agregan pasos con las etiquetas.
- En una sola pasada, se genera código.

a	b	-	c	l	+	>	17	BF	a	b	c	+	:=	23	BI	L17	a	b	c	-	:=	L23	...
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24

- MOV R1, _a L - L - L - L
- SUB R1, _b O - L - L - L
- MOV R2, _c O - O - L - L
- ADD R2, 1 O - O - L - L
- CMP R1, R2 L - L - L - L

JLE L17

- MOV R1, _b O - L - L - L
- ADD R1, _c O - L - L - L
- MOV _a, R1 L - L - L - L

Etiqueta

JMP L23

L17:

- MOV R1, _b O - L - L - L
- SUB R1, _c O - L - L - L
- MOV, _a, R1 L - L - L - L

L23:

Etiqueta



Polaca Inversa → Assembler

IF ($a - b > c + 1$)

THEN $a := b + c$;

ELSE $a := b - c$;

- ▶ MOV R1, _a
- ▶ SUB R1, _b
- ▶ MOV R2, _c
- ▶ ADD R2, 1
- ▶ CMP R1, R2

JLE L17

- ▶ MOV R1, _b
- ▶ ADD R1, _c
- ▶ MOV _a, R1

JMP L23

L17:

- ▶ MOV R1, _b
- ▶ SUB R1, _c
- ▶ MOV, _a, R1

L23:



Tercetos → ASSEMBLER

Sentencia IF

Generación de Código

Tercetos → Assembler

IF ($a - b > c + l$)

THEN $a := b + c$;

ELSE $a := b - c$;

9.	...
10.	(-, a , b)
11.	(+ , c , l)
12.	(> , [10] , [11])
13.	(BF , [12] , <u>17</u>)
14.	(+ , b , c)
15.	(:= , a , [14])
16.	(BI , <u>19</u> , -)
17.	(- , b , c)
18.	(:= , a , [17])
19.	FUERA DEL IF

¿Dónde van las etiquetas?



Tercetos → Assembler

IF (a - b > c + l)

THEN a := b + c;

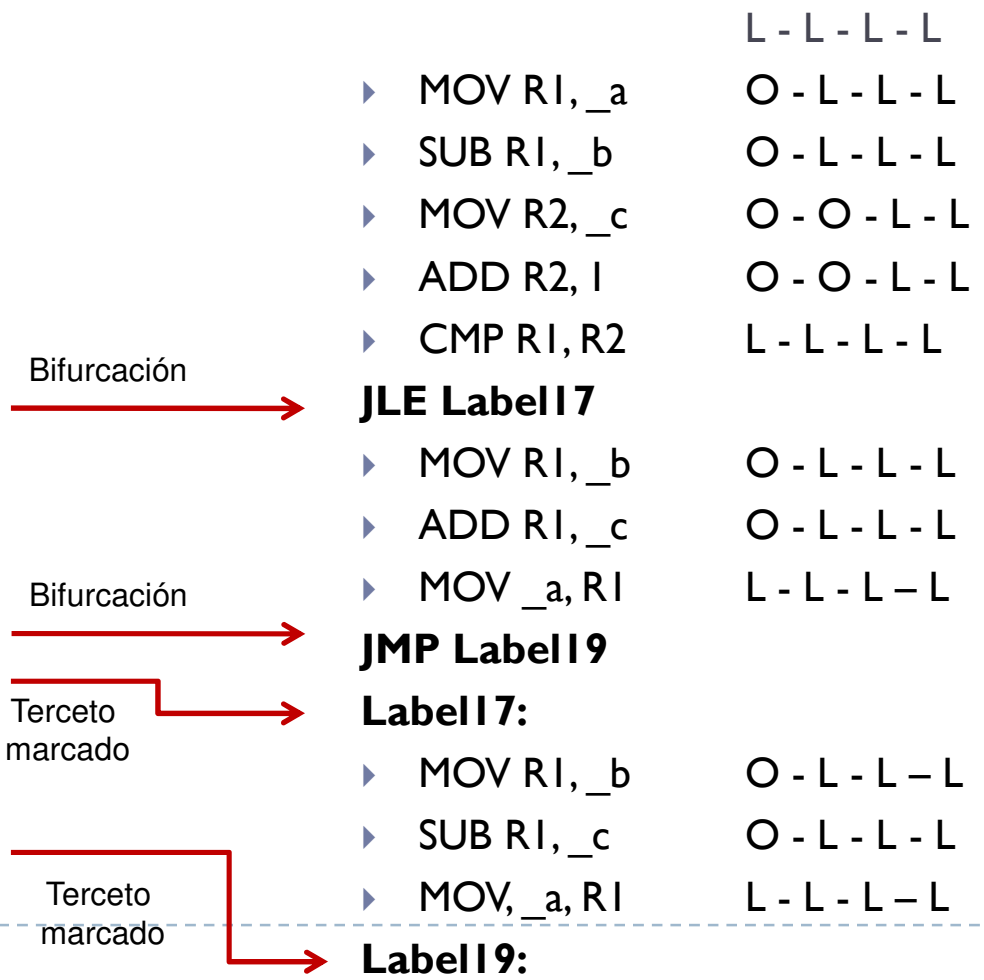
ELSE a := b - c;

Alternativa 1: 2 pasadas

➤ Pasada 1: Se marcan los destinos de las bifurcaciones

➤ Pasada 2: Se genera código

9.	...
10.	(-, a , b)
11.	(+ , c , l)
12.	(> , [10] , [11])
13.	(BF , [12] , <u>17</u>)
14.	(+ , b , c)
15.	(:= , a , [14])
16.	(BI , <u>19</u> , -)
17.	(- , b , c)
18.	(:= , a , [17])
19.	FUERA DEL IF



Tercetos → Assembler

IF (a - b > c + l)

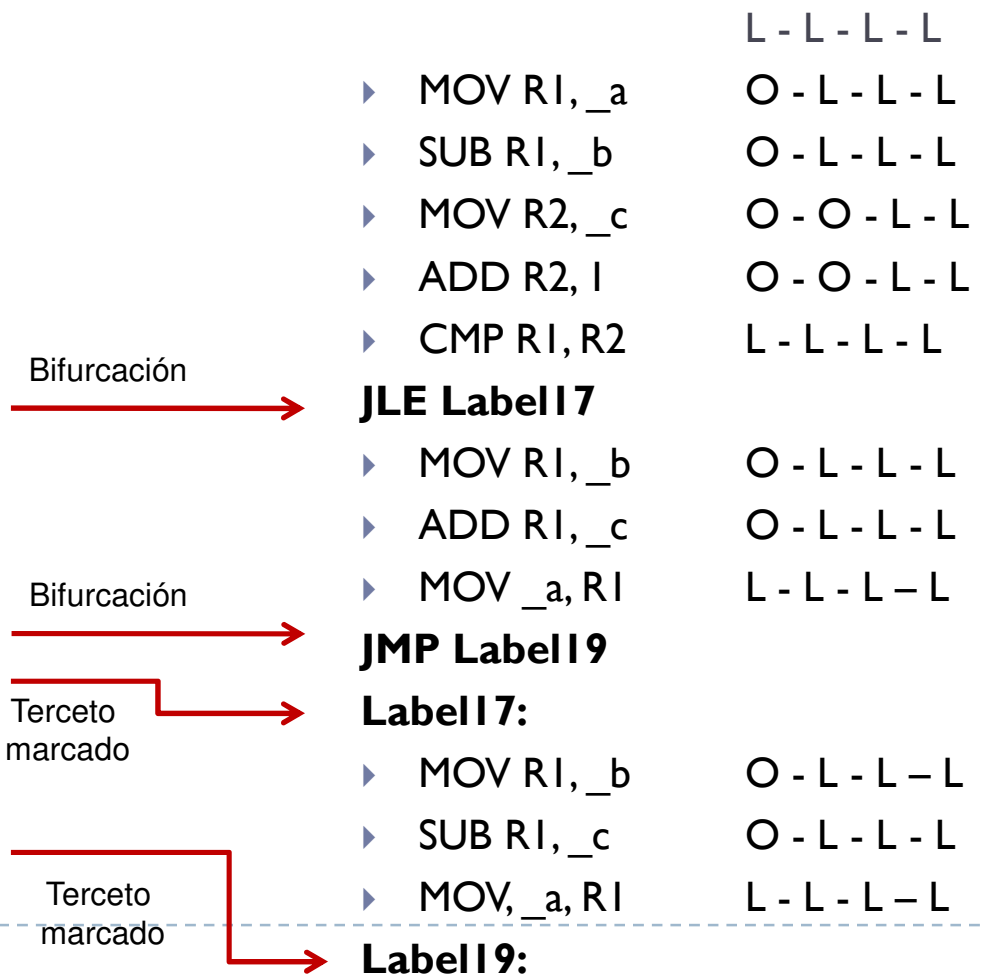
THEN a := b + c;

ELSE a := b - c;

Alternativa 2.a:

- Durante la generación de los Tercetos, se marcan los destinos de las bifurcaciones
- En una sola pasada, se genera código.

9.	...
10.	(-, a , b)
11.	(+ , c , l)
12.	(> , [10] , [11])
13.	(BF , [12] , <u>17</u>)
14.	(+ , b , c)
15.	(:= , a , [14])
16.	(BI , <u>19</u> , -)
17.	(- , b , c)
18.	(:= , a , [17])
19.	FUERA DEL IF



Tercetos \rightarrow Assembler

IF ($a - b > c + 1$)
THEN $a := b + c$;
ELSE $a := b - c$;

- ▶ MOV R1, _a
- ▶ SUB R1, _b
- ▶ MOV R2, _c
- ▶ ADD R2, 1
- ▶ CMP R1, R2

JLE Label17

- ▶ MOV R1, _b
- ▶ ADD R1, _c
- ▶ MOV _a, R1

JMP Label19

Label17:

- ▶ MOV R1, _b
- ▶ SUB R1, _c
- ▶ MOV, _a, R1

Label19:



Tercetos → Assembler

IF (a - b > c + l)

THEN a := b + c;

ELSE a := b - c;

Alternativa 2.b:

- Durante la generación de los Tercetos, se agregan tercetos etiqueta.
- En una sola pasada, se genera código.

9.	...
10.	(-, a , b)
11.	(+ , c , l)
12.	(> , [10] , [11])
13.	(BF , [12] , <u>17</u>)
14.	(+ , b , c)
15.	(:= , a , [14])
16.	(BI , <u>20</u> , -)
17.	(Label17 , - , -)
18.	(- , b , c)
19.	(:= , a , [17])
20.	(Label20 , - , -)

Terceto etiqueta



Terceto etiqueta



▶	MOV RI, _a	L - L - L - L
▶	SUB RI, _b	O - L - L - L
▶	MOV R2, _c	O - L - L - L
▶	ADD R2, l	O - O - L - L
▶	CMP RI, R2	O - O - L - L
	JLE Label17	L - L - L - L
▶	MOV RI, _b	O - L - L - L
▶	ADD RI, _c	O - L - L - L
▶	MOV _a, RI	L - L - L - L
	JMP Label20	
	Label17:	
▶	MOV RI, _b	O - L - L - L
▶	SUB RI, _c	O - L - L - L
▶	MOV, _a, RI	L - L - L - L
	Label20:	

Tercetos \rightarrow Assembler

IF ($a - b > c + 1$)
THEN $a := b + c$;
ELSE $a := b - c$;

- ▶ MOV R1, _a
- ▶ SUB R1, _b
- ▶ MOV R2, _c
- ▶ ADD R2, 1
- ▶ CMP R1, R2

JLE Label17

- ▶ MOV R1, _b
- ▶ ADD R1, _c
- ▶ MOV _a, R1

JMP Label20

Label17:

- ▶ MOV R1, _b
- ▶ SUB R1, _c
- ▶ MOV, _a, R1

Label20:



Árbol Sintáctico → ASSEMBLER
Polaca Inversa → ASSEMBLER
Tercetos → ASSEMBLER

Sentencia WHILE

Código Assembler para Sentencia WHILE

```
WHILE ( a + b <= c - d ) {  
    a := b + c;  
    c := d - 3;}
```

Label1:

- ▶ MOV R1, _a
- ▶ ADD R1, _b
- ▶ MOV R2, _c
- ▶ SUB R2, _d
- ▶ CMP R1, R2

JG Label2

- ▶ MOV R1, _b
- ▶ ADD R1, _c
- ▶ MOV _a, R1
- ▶ MOV R1, _d
- ▶ SUB R1, 3
- ▶ MOV _c, R1

JMP Label1

Label2:

SENTENCIA SIGUIENTE AL WHILE

¿Dónde van las bifurcaciones?

¿Dónde van las etiquetas?

Árbol Sintáctico → ASSEMBLER

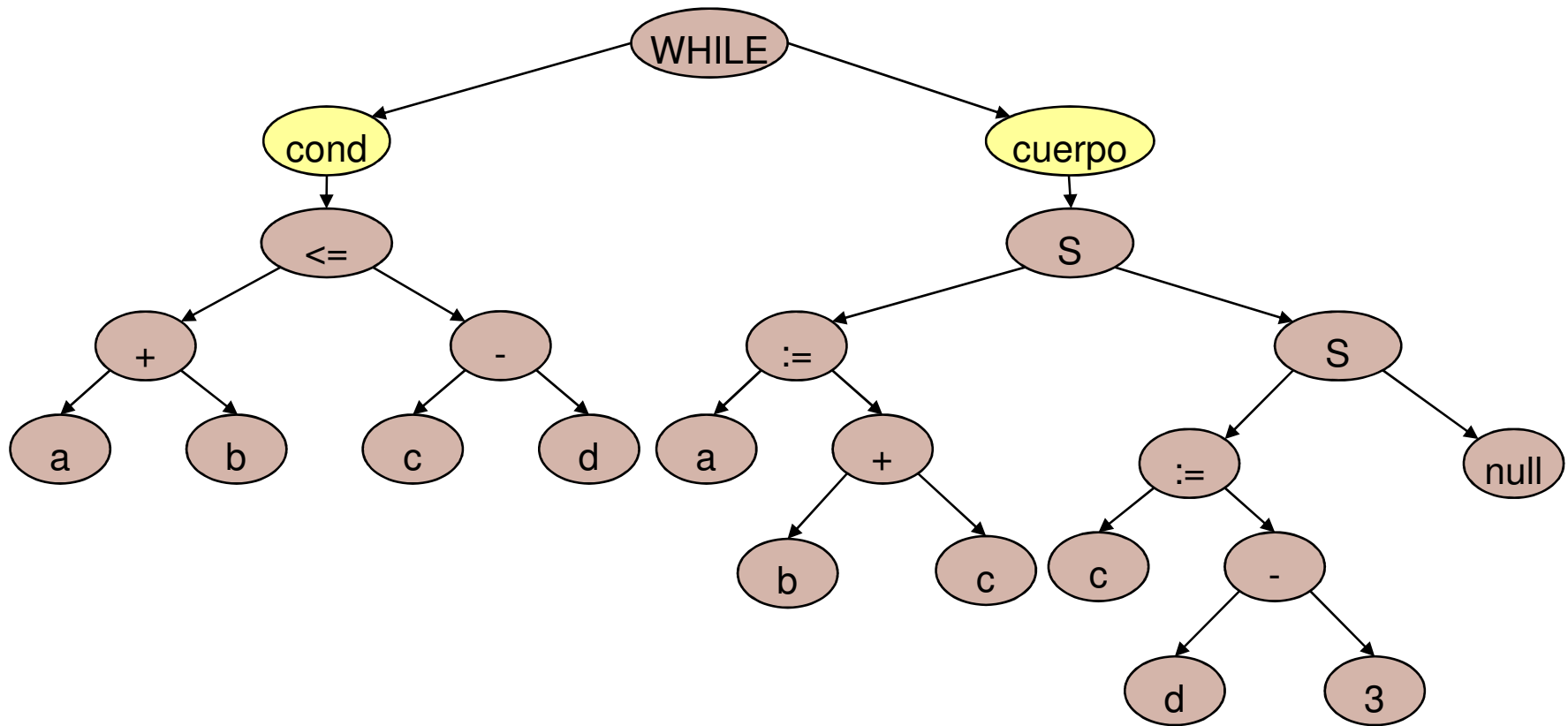
Sentencia WHILE

Generación de Código

Árbol Sintáctico → Assembler

Ejemplo:

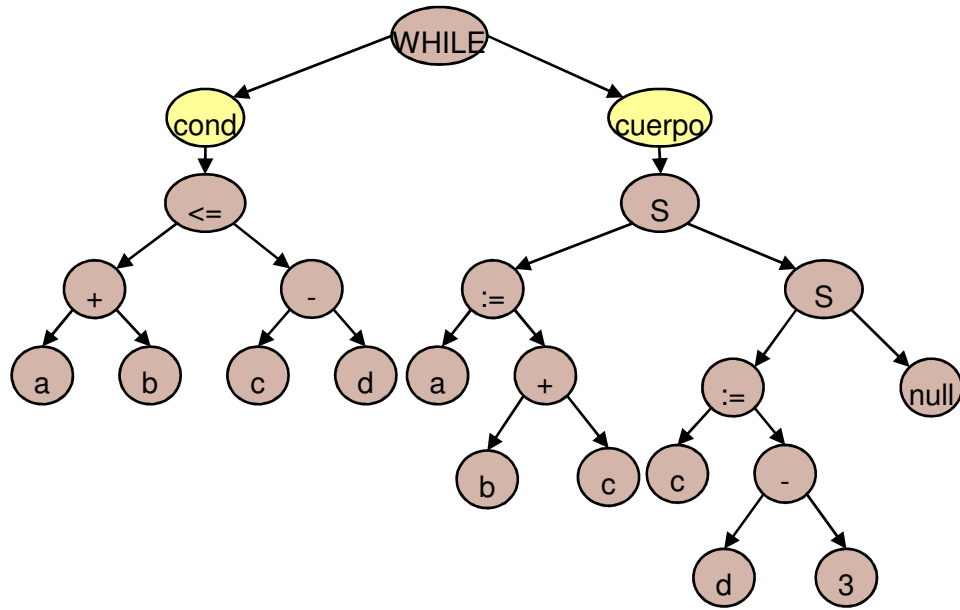
WHILE (a + b <= c - d) { a := b + c; c := d - 3; }



Generación de Código

Árbol Sintáctico → Assembler

```
WHILE ( a + b <= c - d ) {
    a := b + c;
    c := d - 3; }
```



Label1: **Apilo Label1**

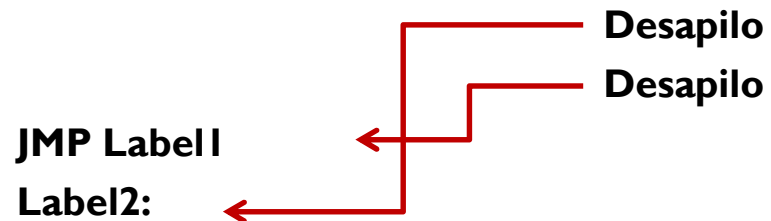
- ▶ MOV R1, _a O - L - L - L
- ▶ ADD R1, _b O - L - L - L
- ▶ MOV R2, _c O - O - L - L
- ▶ SUB R2, _d O - O - L - L
- ▶ CMP R1, R2 L - L - L - L

(Si la raíz del subárbol es cond o es hijo izq de WHILE,

JG Label2 **Apilo Label2**

- ▶ MOV R1, _b O - L - L - L
- ▶ ADD R1, _c O - L - L - L
- ▶ MOV _a, R1 L - L - L - L
- ▶ MOV R1, _d O - L - L - L
- ▶ SUB R1, 3 O - L - L - L
- ▶ MOV _c, R1 L - L - L - L

(Si la raíz del subárbol es cuerpo o hijo der. de WHILE,



▶ SENTENCIA SIGUIENTE AL WHILE

Árbol Sintáctico → Assembler

```
WHILE ( a + b <= c - d ) {  
    a := b + c;  
    c := d - 3;}
```

Label1:

- ▶ MOV R1, _a
- ▶ ADD R1, _b
- ▶ MOV R2, _c
- ▶ SUB R2, _d
- ▶ CMP R1, R2

JG Label2

- ▶ MOV R1, _b O - L - L - L
- ▶ ADD R1, _c O - L - L - L
- ▶ MOV _a, R1 L - L - L - L
- ▶ MOV R1, _d O - L - L - L
- ▶ SUB R1, 3 O - L - L - L
- ▶ MOV _c, R1 L - L - L - L

JMP Label1

Label2:

- ▶ SENTENCIA SIGUIENTE AL WHILE

Polaca Inversa → ASSEMBLER

Sentencia WHILE

Generación de Código

Polaca Inversa \rightarrow Assembler

```
WHILE ( a + b <= c - d ) {  
  a := b + c;  
  c := d - 3;}
```

a	b	+	c	d	-	<=	22	BF	b	c	+	a	:=	d	3	-	c	:=		BI	...
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22

¿Dónde van las etiquetas?



Polaca Inversa → Assembler

```
WHILE ( a + b <= c - d ) {
  a := b + c;
  c := d - 3;}
```

Alternativa 1: 2 pasadas

- Pasada 1: Se marcan los destinos de las bifurcaciones
- Pasada 2: Se genera código

a	b	+	c	d	-	<=	22	BF	b	c	+	a	:=	d	3	-	c	:=		BI	...
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22

Label1:

- MOV R1, _a O - L - L - L
- ADD R1, _b O - L - L - L
- MOV R2, _c O - O - L - L
- SUB R2, _d O - O - L - L
- CMP R1, R2 L - L - L - L

JG Label22

- MOV R1, _b O - L - L - L
- ADD R1, _c O - L - L - L
- MOV _a, R1 L - L - L - L

- MOV R1, _d O - L - L - L
- SUB R1, 3 O - L - L - L
- MOV _c, R1 L - L - L - L

JMP Label1

Label22:

- SENTENCIA SIGUIENTE AL WHILE



Polaca Inversa → Assembler

WHILE (a + b <= c - d) { a := b + c; c := d - 3; }

Alternativa 2.a:

Durante la generación de la Polaca, se marcan los destinos de las bifurcaciones.

En una sola pasada, se genera código.

a	b	+	c	d	-	<=	22	BF	b	c	+	a	:=	d	3	-	c	:=	I	BI	...
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22

Alternativa 2.b:

Durante la generación de la Polaca, se agregan pasos con las etiquetas.

En una sola pasada, se genera código.

L1	a	b	+	c	d	-	<=	23	BF	b	c	+	a	:=	d	3	-	c	:=	I	BI	L23	...
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	



Polaca Inversa → Assembler

```
WHILE ( a + b <= c - d ) {  
    a := b + c;  
    c := d - 3;}
```

Label1:

- ▶ MOV R1, _a
- ▶ ADD R1, _b
- ▶ MOV R2, _c
- ▶ SUB R2, _d
- ▶ CMP R1, R2

JG Label22/23

- ▶ MOV R1, _b O - L - L - L
- ▶ ADD R1, _c O - L - L - L
- ▶ MOV _a, R1 L - L - L - L
- ▶ MOV R1, _d O - L - L - L
- ▶ SUB R1, 3 O - L - L - L
- ▶ MOV _c, R1 L - L - L - L

JMP Label1

Label22/23:

- ▶ SENTENCIA SIGUIENTE AL WHILE

Tercetos → ASSEMBLER

Sentencia WHILE

Generación de Código

Tercetos \rightarrow Assembler

```
WHILE (a + b <= c - d) {  
  a := b + c;  
  c := d - 3;}  
-----
```

24.	...
25.	(+ , a , b)
26.	(- , c , d)
27.	(<= , [25] , [26])
28.	(BF , [27] , <u>34</u>)
29.	(+ , b , c)
30.	(:= , a , [29])
31.	(- , d , 3)
32.	(:= , c , [31])
33.	(BI , <u>25</u> , -)
34.	SENT_SIG_AL_WHILE

¿Dónde van las etiquetas?



Tercetos → Assembler

```
WHILE ( a + b <= c - d ) {
  a := b + c;
  c := d - 3;}
```

Alternativa 1: 2 pasadas

➤ Pasada 1: Se marcan los destinos de las bifurcaciones

➤ Pasada 2: Se genera código

24.	...
25.	(+ , a , b)
26.	(- , c , d)
27.	(<= , [25] , [26])
28.	(BF , [27] , <u>34</u>)
29.	(+ , b , c)
30.	(:= , a , [29])
31.	(- , d , 3)
32.	(:= , c , [31])
33.	(BI , <u>25</u> , -)
34.	SENT_SIG_AL_WHILE

Label25:

- ▶ MOV R1, _a O - L - L - L
- ▶ ADD R1, _b O - L - L - L
- ▶ MOV R2, _c O - O - L - L
- ▶ SUB R2, _d O - O - L - L
- ▶ CMP R1, R2 L - L - L - L

JG Label34

- ▶ MOV R1, _b O - L - L - L
- ▶ ADD R1, _c O - L - L - L
- ▶ MOV _a, R1 L - L - L - L
- ▶ MOV R1, _d O - L - L - L
- ▶ SUB R1, 3 O - L - L - L
- ▶ MOV _c, R1 L - L - L - L

JMP Label25

Label34:

- ▶ SENTENCIA SIGUIENTE AL WHILE

Tercetos → Assembler

```
WHILE ( a + b <= c - d ) { a := b + c; c := d - 3; }
```

Alternativa 2.a:

Durante la generación de los Tercetos, se marcan los destinos de las bifurcaciones

En una sola pasada, se genera código.

24.	...
25.	(+ , a , b)
26.	(- , c , d)
27.	(<= , [25] , [26])
28.	(BF , [27] , <u>34</u>)
29.	(+ , b , c)
30.	(:= , a , [29])
31.	(- , d , 3)
32.	(:= , c , [31])
33.	(BI , <u>25</u> , -)
34.	FUERA_DEL_WHILE

Tercetos → Assembler

```
WHILE ( a + b <= c - d ) { a := b + c ; c := d - 3 ; }
```

Alternativa 2.b:

Durante la generación de los Tercetos, se agregan tercetos etiqueta.
En una sola pasada, se genera código.

24.	...
25.	(Label25, -, -)
26.	(+ , a , b)
27.	(- , c , d)
28.	(<= , [26] , [27])
29.	(BF , [28] , <u>35</u>)
30.	(+ , b , c)
31.	(:= , a , [30])
32.	(- , d , 3)
33.	(:= , c , [32])
34.	(BI , <u>25</u> , -)
35.	(Label 35 , - , -)
36.	FUERA_DEL_WHILE



Polaca Inversa → Assembler

```
WHILE ( a + b <= c - d ) {  
    a := b + c;  
    c := d - 3;}
```

Label25:

- ▶ MOV R1, _a
- ▶ ADD R1, _b
- ▶ MOV R2, _c
- ▶ SUB R2, _d
- ▶ CMP R1, R2

JG Label34/35

- ▶ MOV R1, _b O - L - L - L
- ▶ ADD R1, _c O - L - L - L
- ▶ MOV _a, R1 L - L - L - L
- ▶ MOV R1, _d O - L - L - L
- ▶ SUB R1, 3 O - L - L - L
- ▶ MOV _c, R1 L - L - L - L

JMP Label25

Label34/35:

- ▶ SENTENCIA SIGUIENTE AL WHILE

¿PREGUNTAS?