Lenguajes y Compiladores

Generación de Código Intermedio

Polaca inversa, tercetos, árbol

Sentencias de Control

UNLaM

Árbol Sintáctico Sentencias de Selección

Árbol Sintáctico

Sentencia de Selección (sin else)

1. start -> sel
2. sel -> IF cond THEN accion ENDIF
3. cond -> ID < CTE
4. accion -> ID := exp
5. exp -> exp + term
6. exp -> term
7. term -> term * factor
8. term -> factor
9. factor -> CTE
10. factor -> ID

Reglas aplicadas bajo el análisis SLR

3(a,3) **10**(c) **8** **6** **9**(1) **8** **5** **4**(b) **2** **1**

Sentencia a analizar

```
if   a < 3
then
            b:= c+1
endif
```

Árbol Sintáctico

Sentencia de Selección (sin else)

1. start -> sel
2. sel -> IF cond THEN accion { crear_nodo (IF , cond.Ptr , accion.Ptr) ; } ENDIF
3. cond -> ID < CTE { cond.Ptr = crear_nodo (<,crear_hoja (ID) ,crear_hoja (CTE)) ; }
4. accion -> ID := exp {accion.Ptr = crear_nodo (:= , crear_hoja(ID), exp.Ptr) ; }
5. exp -> exp + term {exp.Ptr = crear_nodo (+, exp.Ptr, term.Ptr) ; }
6. exp -> term { exp.Ptr = term.Ptr; }
7. term -> term * factor {term.Ptr = crear_nodo (*, term.Ptr, factor.Ptr) ; }
8. term -> factor { factor.Ptr = term.Ptr; }
9. factor -> CTE {factor.Ptr = crear_hoja (CTE);}
10. factor -> ID { factor.Ptr = crear_hoja (ID); }

Reglas aplicadas bajo el análisis SLR

3(a,3) **10**(c) **8** **6** **9**(1) **8** **5** **4**(b) **2** **1**

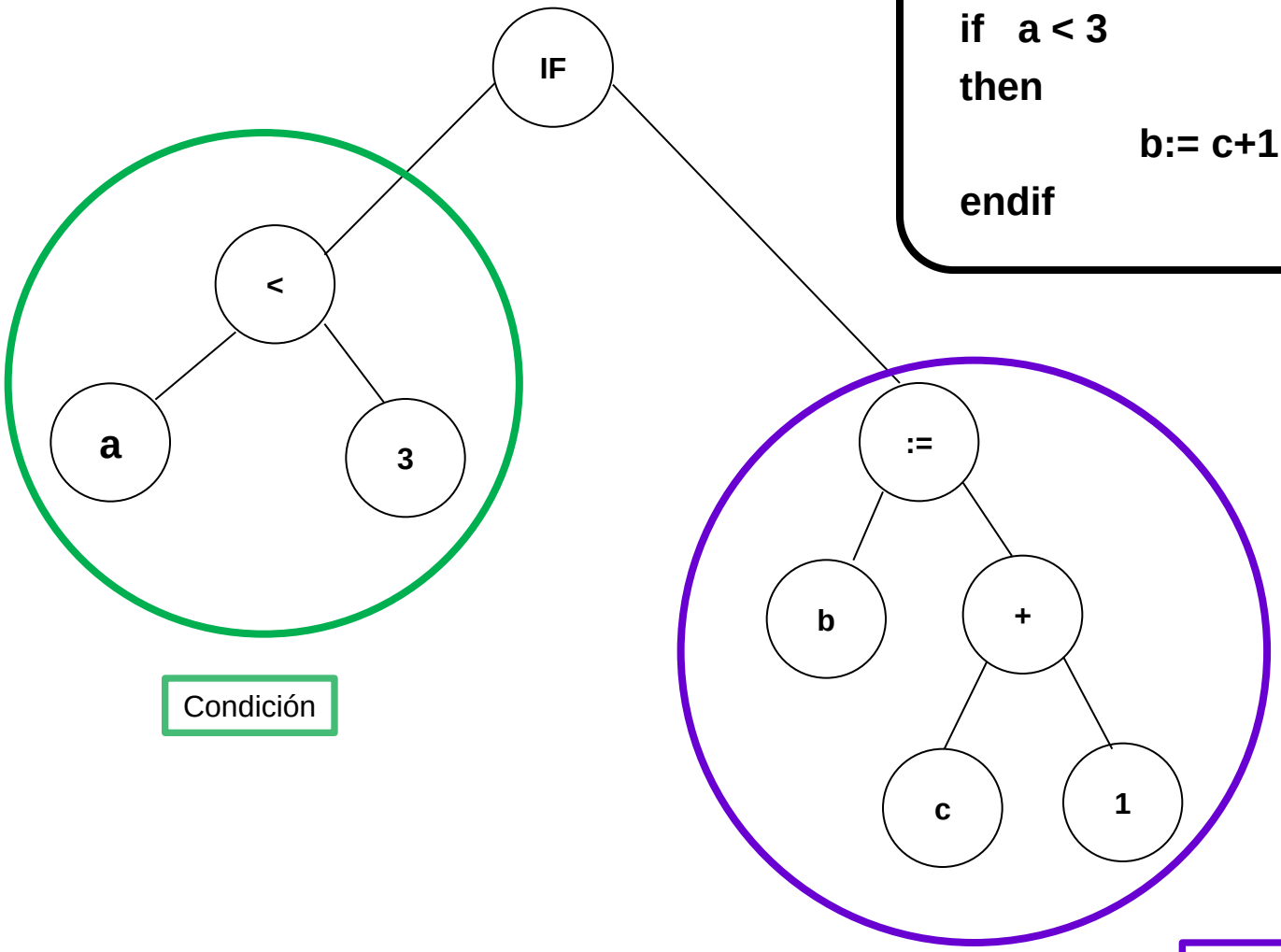
Sentencia a analizar

```
if   a < 3
then
            b:= c+1
endif
```

Árbol Sintáctico

Sentencia de Selección (sin else)

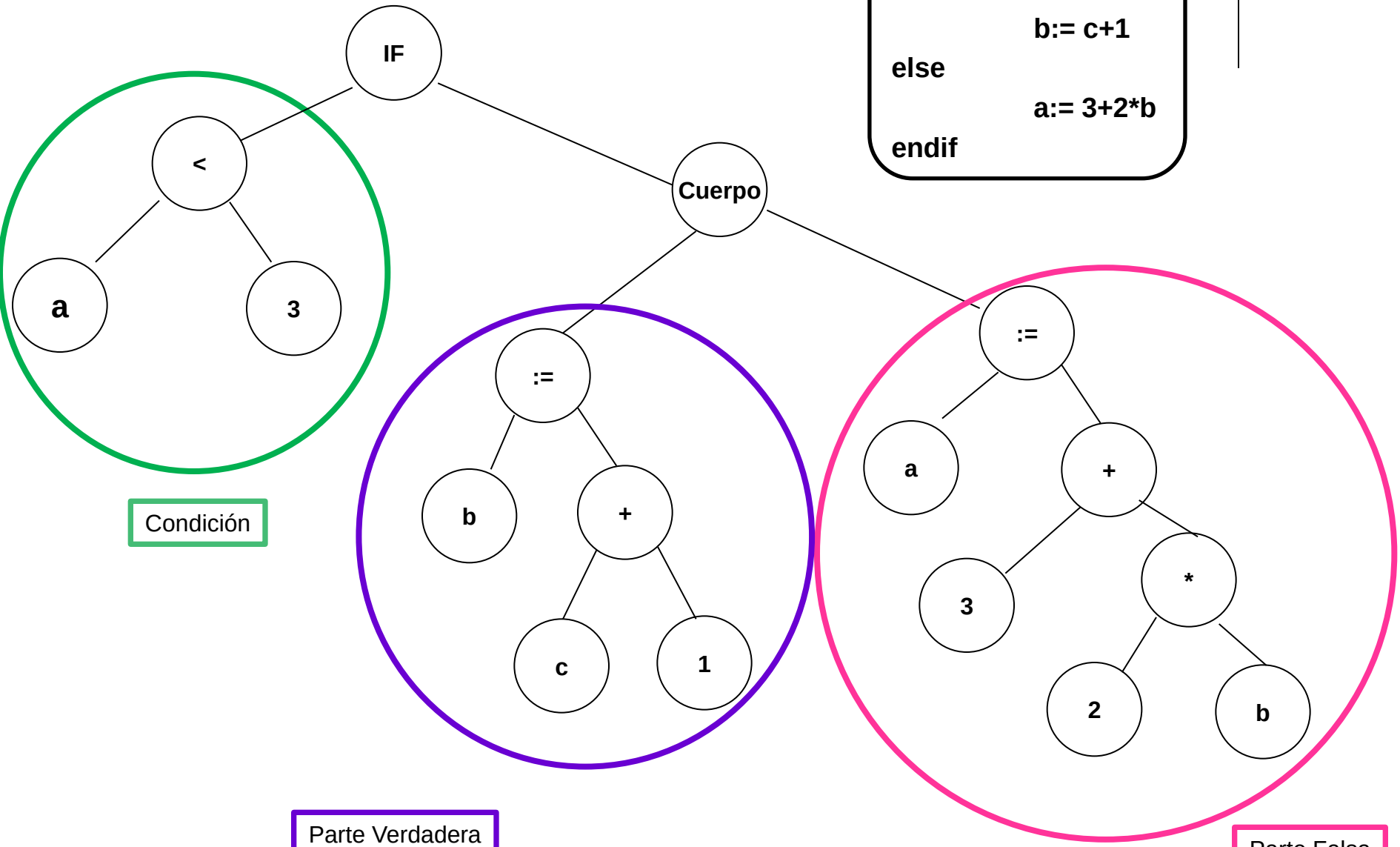

```
if   a < 3
then
            b:= c+1
endif
```



Árbol Sintáctico

Sentencia de Selección (con else)


```
if  a < 3
then
            b:= c+1
else
            a:= 3+2*b
endif
```

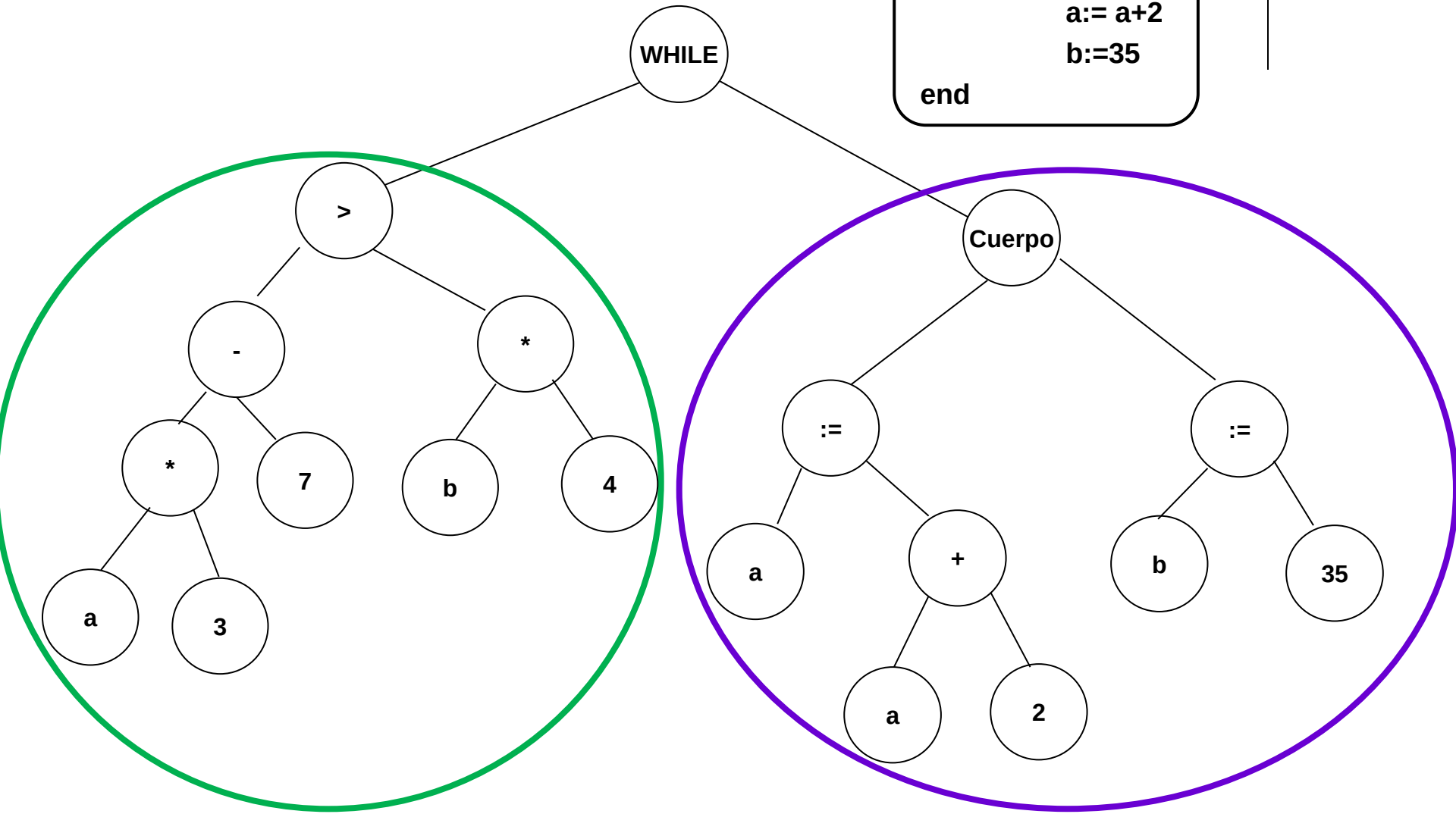


Árbol Sintáctico Sentencias de Iteración

Árbol Sintáctico

Sentencia de iteración (while)


```
While   a*3-7 > b*4
            a:= a+2
            b:=35
end
```



Condición

Cuerpo

Polaca Inversa Sentencias de Selección

Polaca inversa

Sentencia de selección (sin else)

1) start -> sel
2) sel -> IF cond THEN accion ENDIF
3) cond -> ID < CTE
4) accion -> ID := exp
5) exp -> exp + term
6) exp -> term
7) term -> term * factor
8) term -> factor
9) factor -> CTE
10) factor -> ID

Sentencia IF

```
if a< 3
then
        b:= c+1
endif
```

Reglas aplicadas bajo el análisis SLR

3 (a,3) **10**(c) **8 6 9**(1) **8 5**(+) **4** (b) **2 1** (:=)

Condición (Regla 3)

	19	20	21	22	23	24	25	26	27	28	29
	a	3	CMP	BGE	29	c	1	+	b	:=	
	Lado Izquierdo Comparación	Lado Derecho Comparación	Reservado			Then					

Polaca inversa

Sentencia de selección (sin else)

	19	20	21	22	23	24	25	26	27	28	29
	a	3									
	Lado Izquierdo Comparación	Lado Derecho Comparación	Reservado			Then					

¿Cómo se llenan los espacios reservados?

En Assembler las comparaciones se realizan por diferencia…

LI-LD < 0 ➡ **then**

LI-LD >= 0 ➡ **nada**

Sentencia IF

```
if a< 3
then
        b:= c+1
endif
```

Polaca inversa

Sentencia de selección (sin else)

Valor Original	Valor "Assembler"
>=	BLT (Less Than)
>	BLE (Less Equal)
<=	BGT (Greather Than)
<	BGE (Greather Equal)
<>	BEQ (EQual)
==	BNE (Not Equal)

Sentencia IF

if a < 3
then
b:= c+1
endif

20	21	22	23	24
	CMP	BGE	29	

Cuando falla la comparación, salta a la celda 29

Polaca inversa

Sentencia de selección (sin else)

	19	20	21	22	23	24	25	26	27	28	29
	a	3	CMP	BGE	29	c	1	+	b	:=	
	Lado Izquierdo Comparación	Lado Derecho Comparación	**Reservado**			Then					

Así es el resultado del if…

Pero esto se debe llenar a medida que el parser pasa por las reglas de la gramática…

Existe un algoritmo…

Polaca inversa

Sentencia de selección (sin else)

Algoritmo (uso de pila)

Fin de Condición ➡ Apilar # celda actual

Fin del then ➡ Desapilar X (tope de la pila)
X = # celda actual

	19	20	21	22	23	24	25	26	27	28	29
	a	3	CMP	BGE	29	c	1	+	b	:=	
	Lado Izquierdo Comparación	Lado Derecho Comparación	Reservado			Then					

Polaca inversa

Sentencia de selección (sin else)

1. start -> sel
2. sel -> IF cond THEN { insertar CMP; insertar BGE;apilar #celda actual ; avanzar ; } accion {desapilar X (tope de la pila) ; escribir en X #celda actual } ENDIF
3. cond -> ID < CTE { insertar id ; insertar cte ; }
4. accion -> ID := exp { insertar id ; insertar := ; }
5. exp -> exp + term { insertar +; }
6. exp -> term
7. term -> term * factor { insertar * ; }
8. term -> factor
9. factor -> CTE { insertar cte ; }
10. factor -> ID { insertar id ; }

Sentencia a analizar

```
if   a < 3
then
            b:= c+1
endif
```

Reglas aplicadas bajo el análisis SLR

3 (a,3) **10**(c) **8 6 9**(1) **8 5**(+) **4** (b) **2 1** (:=)

Polaca inversa

Sentencia de selección (con else)

1) start -> sel
2) sel -> IF cond THEN accion ENDIF
3) sel -> IF cond THEN accion ELSE accion ENDIF
4) cond -> IF < CTE
5) accion -> ID:= exp
6) exp -> exp + term
7) exp -> term
8) term -> term * factor
9) term -> factor
10) factor -> CTE
11) factor -> ID

Sentencia IF

```
if a< 3
then
        b:= c+1
else
        a:=5
endif
```

Reglas aplicadas bajo el análisis SLR

4(a,3) **11**(c) **9 7 10**(1) **9 6**(+) **5**(b) **10**(5) **9 7 5**(a) **3 1**(:=)

	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
	a	3	CMP	BGE	31	c	1	+	b	:=	BI	34	5	a	:=	
	Lado Izquierdo Comparación	Lado Derecho Comparación	Reservado			Then					Reservado		Else			

Polaca inversa

Sentencia de selección (con else)

Algoritmo (uso de pila)

Fin de Condición → Apilar # celda actual

Fin del then → Desapilar X (tope de la pila)
X = # celda actual
Apilar # celda actual + 1

Fin del else → Desapilar X (tope de la pila)
X = # celda actual

	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
	a	3	CMP	BGE	31	c	1	+	b	:=	BI	34	5	a	:=	
	Lado Izquierdo Comparación	Lado Derecho Comparación	Reservado			Then					Reservado		Else			

Polaca inversa

Sentencia de selección (con else)

1. start -> sel
2. sel -> IF cond THEN accion ENDIF
3. sel -> IF cond THEN {insertar CMP; insertar BGE; apilar #celda actual ; avanzar ;} accion ELSE { insertar BI; desapilar X (tope de la pila) ; escribir en X #celda actual +1 ; apilar # celda actual; avanzar} accion {desapilar X (tope de la pila) ; escribir en X #celda actual} ENDIF
4. cond -> id < cte { insertar id ; insertar cte ; }
5. accion -> id := exp { insertar id ; insertar := ; }
6. exp -> exp + term { insertar +; }
7. exp -> term
8. term -> term * factor { insertar * ; }
9. term -> factor
10. factor -> CTE { insertar cte ; }
11. factor -> ID { insertar id ; }

Sentencia a analizar

```
If a < 3
then
    b:= c+1
else
   a:= 28
endif
```

Reglas aplicadas bajo el análisis SLR

4(a,3) **11**(c) **9 7 10**(1) **9 6**(+) **5**(b) **10**(28) **9 7 5**(a) **3 1**(:=)

Polaca Inversa - Sentencias de Iteración

Polaca inversa

Sentencia de iteración (while)

Así es el resultado de la polaca inversa para un ciclo while sobre el siguiente programa

```
while a*3-7 <= b*4
            a:= a+2
 end
```

27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46
	a	3	*	7	-	b	4	*	CMP	BGT	46	a	2	+	a	:=	BI	28	
	Lado IZQ Comparación					**Lado DER Comp.**			**Reser-vado**			**Cuerpo While**					**Reser-vado**		

Polaca inversa

Sentencia de iteración (while)

Algoritmo (uso de pila)

Comienzo → Apilar # celda actual

Fin de la Condición → Apilar # celda actual

Fin del While → Desapilar Z (tope de la pila)
Z = # celda actual + 1
Desapilar Z (tope de la pila)
celda actual = Z

27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46
	a	3	*	7	-	b	4	*	CMP	BGT	46	a	2	+	a	:=	BI	28	
		Lado IZQ Comparación				Lado DER Comp.			Reservado			Cuerpo While					Reservado		

Polaca inversa

Sentencia de iteración (while)

1) start -> ciclo
2) ciclo -> WHILE {apilar celda actual ;escribir ET ; } cond {apilar # celda actual ; avanzar} accion {escribir BI ; desapilar Z (tope de pila); escribir en Z celda actual + 1; desapilar Z (tope de pila); escribir Z en celda actual } END
3) cond -> exp <= exp {escribir CMP ; escribir BGT}
4) accion -> id := exp {escribir id ; escribir :=}
5) exp -> exp + term {escribir +;}
6) exp -> exp - term {escribir -;}
7) exp -> term
8) term -> term * factor {escribir *;}
9) term -> factor
10) factor -> CTE {escribir cte;}
11) factor -> ID {escribir id;}

Sentencia WHILE

```
while a*3-7 <= b*4
            a:= a+2
 end
```

Reglas a aplicar

11(a) 9 10(3) 8 7 10(7) 9 6 11(b) 9 10(4) 8 7 3 11(a) 9 7 10(2) 9 5 4 2 1

¿Preguntas?