Código Intermediário - Objetivo

Transformar a representação hierárquica em sequencial.

Gerar um representação intermediária.

Simular um programa para uma máquina abstrata.

Código Intermediário - Objetivo

Adiar decisões relacionadas com **características** da máquina alvo.

Características que tornam o processo complexo:

- Quantidade de registradores.
- Localização física das variáveis (memória ou registrador).
- Endereçamento das funções/procedimentos.

Código Intermediário - Aspectos

Mais fácil otimizar código intermediário que código final.

Compiladores Híbridos

- Pode ser a última etapa.
- Normalmente a otimização fica na máquina virtual.

Código Intermediário - Linguagem

Utiliza uma linguagem intermediária entre a fonte e a final.

Deve ter conjunto de instruções que facilite essa transformação.

Linguagem intermediária deve ser:

- Fácil de gerar a partir da árvore sintática
- Fácil de ser transformada em código final.

Código Intermediário - Representação

Linguagem deve ser compatível com as instruções no conjunto de instruções final.

Pode conter mais instruções que o conjunto final.

Cada instrução intermediária deve ser possível traduzir para uma ou mais instruções final.

Código Intermediário - Formato

Código de 3 endereços

Forma intermediária onde cada instrução possui 1 operador e no máximo 3 posições de memória associados.

Formato

OP	Endereço 1	Endereço 2	Endereço 3

Código Intermediário - Exemplo

Instruções de 3 endereços

OP	Endereço 1	Endereço 2	Endereço 3
ATR	destino	origem	-
ADD	resultado	operando1	operando2
JMP	label	-	-
JNZ	label	variável	-
JGT	label	operando1	Operando2

Transformação para Código Intermediário

Cada sub-árvore será tratada individualmente.

Identificar as sub-árvore das regras sintáticas.

Definir um padrão de instruções associada a sub-árvore.

Nem todo sub-árvore terá um padrão associado.

Transformação – Sub-árvores

Cada tipo de sub-árvore vai ter características diferentes.

Declaração: Reservar espaço de armazenamento.

Comando: Cria label para destino do salto.

Expressão: Cria variável temporária para resultado.

Transformação – Regras

BLC -> **begin** COMS **end**
COMS -> COM **;** COMS
COMS -> ε
COM -> ID **:=** EXP
COM -> **if** EXP **then** BLC
COM -> **if** EXP **then** BLC **else** BLC
COM -> **while** EXP BLOCO
EXP -> PARM OP EXP
EXP -> PARM
PARM -> ID | NUM

Transformação – Regras sem instruções

BLC -> **begin** COMS **end**
COMS -> COM ; COMS
COMS -> ε
COM -> ID **:=** EXP
COM -> **if** EXP **then** BLC
COM -> **if** EXP **then** BLC **else** BLC
COM -> **while** EXP BLC
EXP -> PARM OP EXP
EXP -> PARM
PARM -> ID | NUM

Transformação – Expressões

EXP -> PARM OP EXP

OP	Endereço 1	Endereço 2	Endereço 3
OP	TEMP	PARAM	TEMP_EXP

EXP -> PARM

OP	Endereço 1	Endereço 2	Endereço 3
ATR	TEMP	PARAM	-

Transformação – Atribuição

COM -> ID := EXP

OP	Endereço 1	Endereço 2	Endereço 3
ATR	ID	TEMP_EXP	-

Transformação - Exemplo

Código Fonte

A := B + C + D

Regras

EXP -> PARM OP EXP

COM -> ID **:=** EXP

Código Intermediário

ADD	temp1	C	D
ADD	temp2	B	temp1
ATR	A	temp2	

Transformação – Condicional

COM -> **if** EXP **then** BLC

instruções EXP

OP	Endereço 1	Endereço 2	Endereço 3
JMZ	LABEL_FIM	TEMP_EXP	-

instruções BLC

OP	Endereço 1	Endereço 2	Endereço 3
LBL	LABEL_FIM	-	-

Transformação - Condicional

COM -> **if** EXP **then** BLC **else** BLC

instruções EXP

OP	Endereço 1	Endereço 2	Endereço 3
JMZ	LABEL_ELSE	TEMP_EXP	-

instruções BLOCO_IF

OP	Endereço 1	Endereço 2	Endereço 3
JMP	LABEL_FIM	-	-
LBL	LABEL_ELSE	-	-

instruções BLOCO_ELSE

OP	Endereço 1	Endereço 2	Endereço 3
LBL	LABEL_FIM	-	-

Transformação – Repetição

COM -> **while** EXP BLC

OP	Endereço 1	Endereço 2	Endereço 3
LBL	LABEL_INI	-	-

instruções EXP

OP	Endereço 1	Endereço 2	Endereço 3
JMZ	LABEL_FIM	TEMP_EXP	-

instruções BLC

OP	Endereço 1	Endereço 2	Endereço 3
JMP	LABEL_INI	-	-
LBL	LABEL_FIM	-	-

Transformação - Exemplo

Código Fonte

if A > B **then** A := C

Regras

COM -> **if** EXP **then** BLC

COM -> ID **:=** EXP

EXP -> PARM OP EXP

Código Intermediário

Instruções da expressão A > B			
JMZ	Label1	temp1	-
Instruções do bloco A := B			
LBL	label1	-	-

Transformação - Exemplo

Código Fonte

if A > B **then** A := C

Regras

COM -> **if** EXP **then** BLC

COM -> ID **:=** EXP

EXP -> PARM OP EXP

Código Intermediário

GRT	temp1	A	B
JMZ	Label1	temp1	-
ATR	A	C	-
LBL	label1	-	-

Algoritmo - Código Intermediário

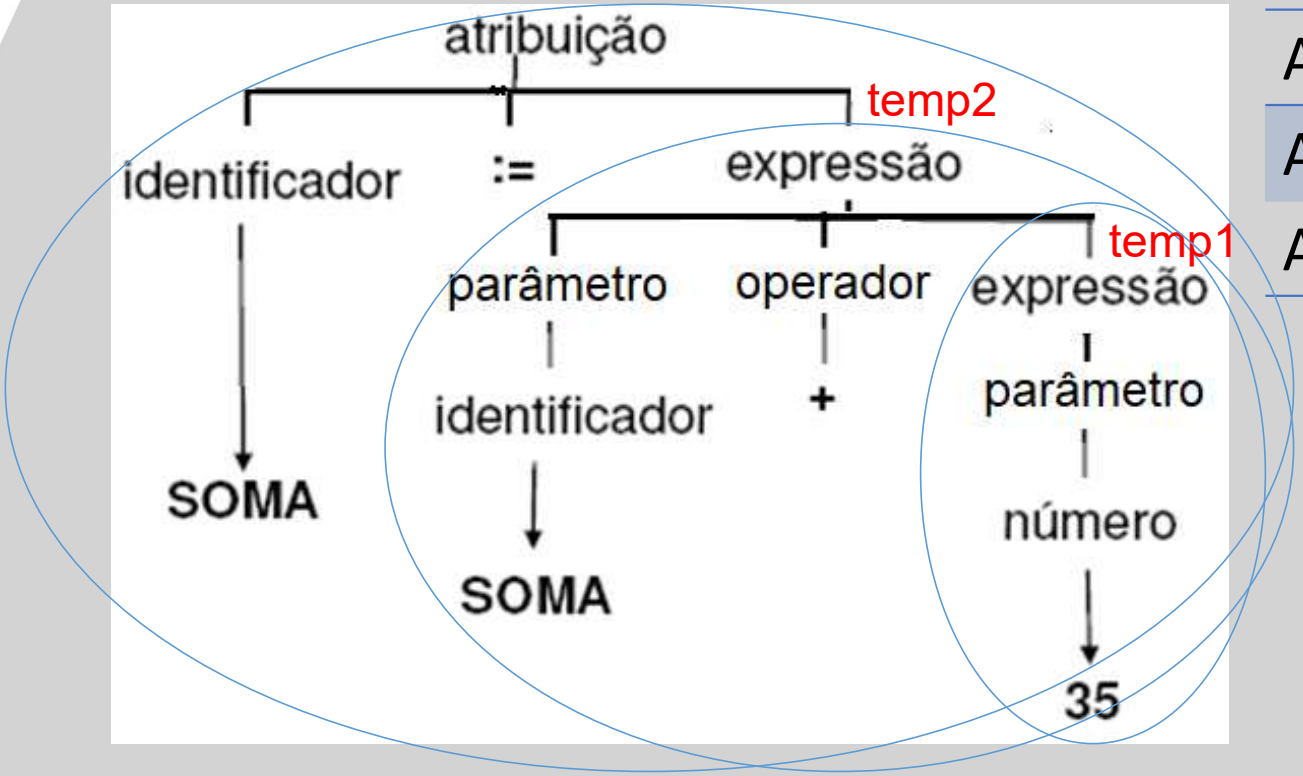
Percorrer a árvore em **pós ordem** gerando o código associado sub-árvore do nó visitado.

Gerar o código seguindo o padrão associado a regra.

As instruções das sub-árvores filhas são arrumadas conforme o padrão.

Algoritmo - Exemplo

SOMA := SOMA + 35



ATR	temp1	35	
ADD	temp2	SOMA	temp1
ATR	SOMA	temp2	

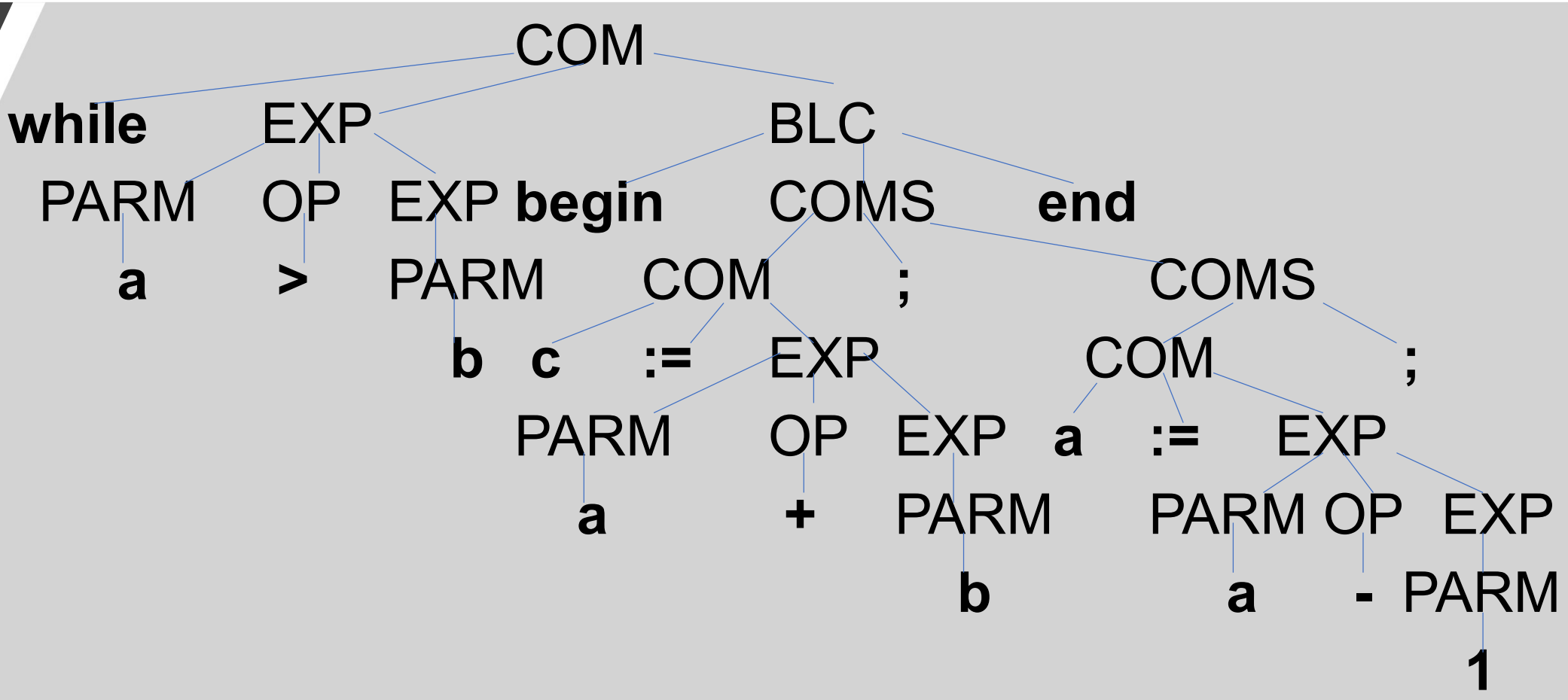
Código Intermediário - Algoritmo

```
função CI( No n ){
        para todos t.filhos faça
                instruções = CI( filho );
        return formatarInstrucoes( instruções, padrão );
}
```

Algoritmo - Exemplo

```
...
while a > b
begin
        c := a + b;
        a := a – 1;
end
...
```

Algoritmo – Exemplo



Algoritmo – Exemplo

COM -> **while** EXP BLOCO

LBL | LABEL_INI | - | -

instruções EXP

JMZ | LABEL_FIM | TEMP_EXP | -

instruções BLOCO

JMP | LABEL_INI | - | -

LBL | LABEL_FIM | - | -

Algoritmo – Exemplo

COM -> **while** EXP BLOCO

LBL | LABEL_INI | - | -

instruções EXP

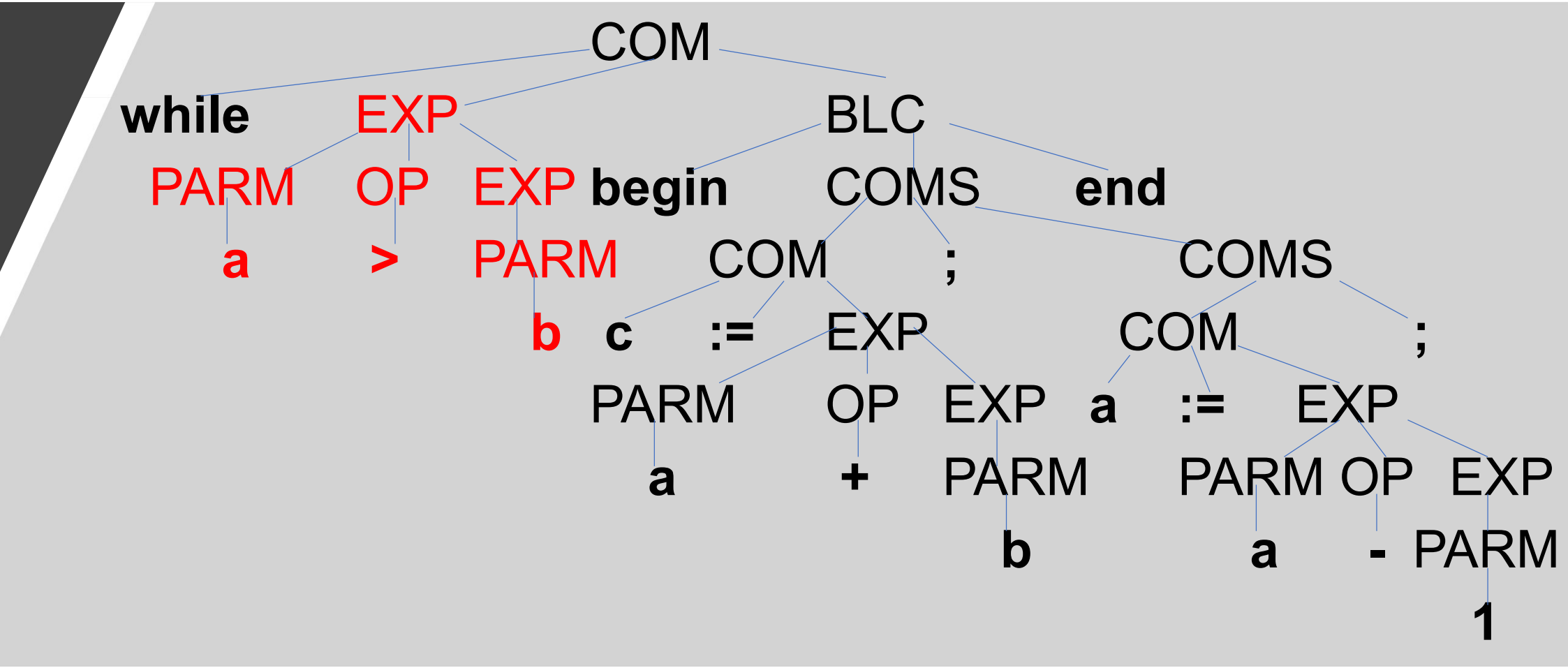
JMZ | LABEL_FIM | TEMP_EXP | -

instruções BLOCO

JMP | LABEL_INI | - | -

LBL | LABEL_FIM | - | -

Algoritmo - Exemplo



Algoritmo – Exemplo

COM -> **while** EXP BLOCO

LBL | LABEL_INI | - | -

instruções EXP

JMZ | LABEL_FIM | TEMP_EXP | -

instruções BLOCO

JMP | LABEL_INI | - | -

LBL | LABEL_FIM | - | -

Algoritmo - Exemplo

COM -> **while** EXP BLOCO

LBL | LABEL_INI | - | -

instruções EXP

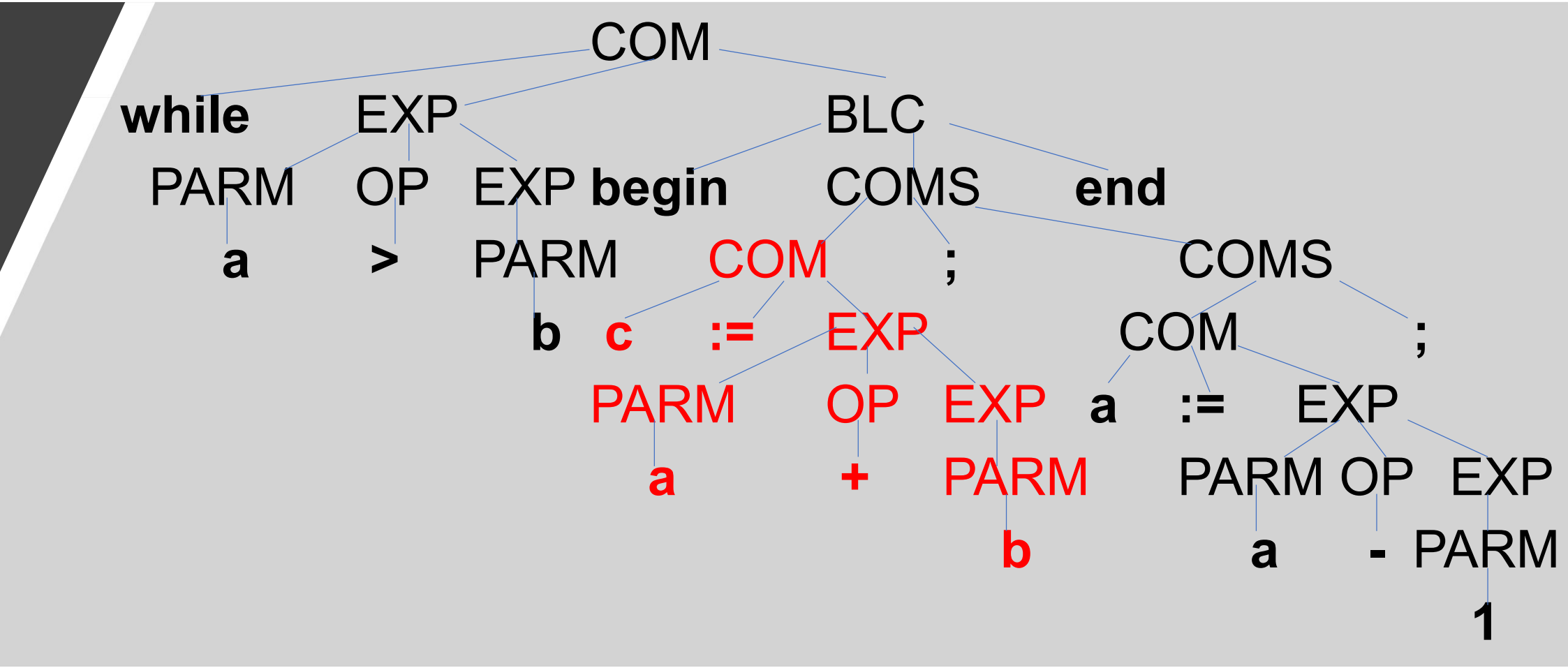
JMZ | LABEL_FIM | TEMP_EXP | -

instruções BLOCO

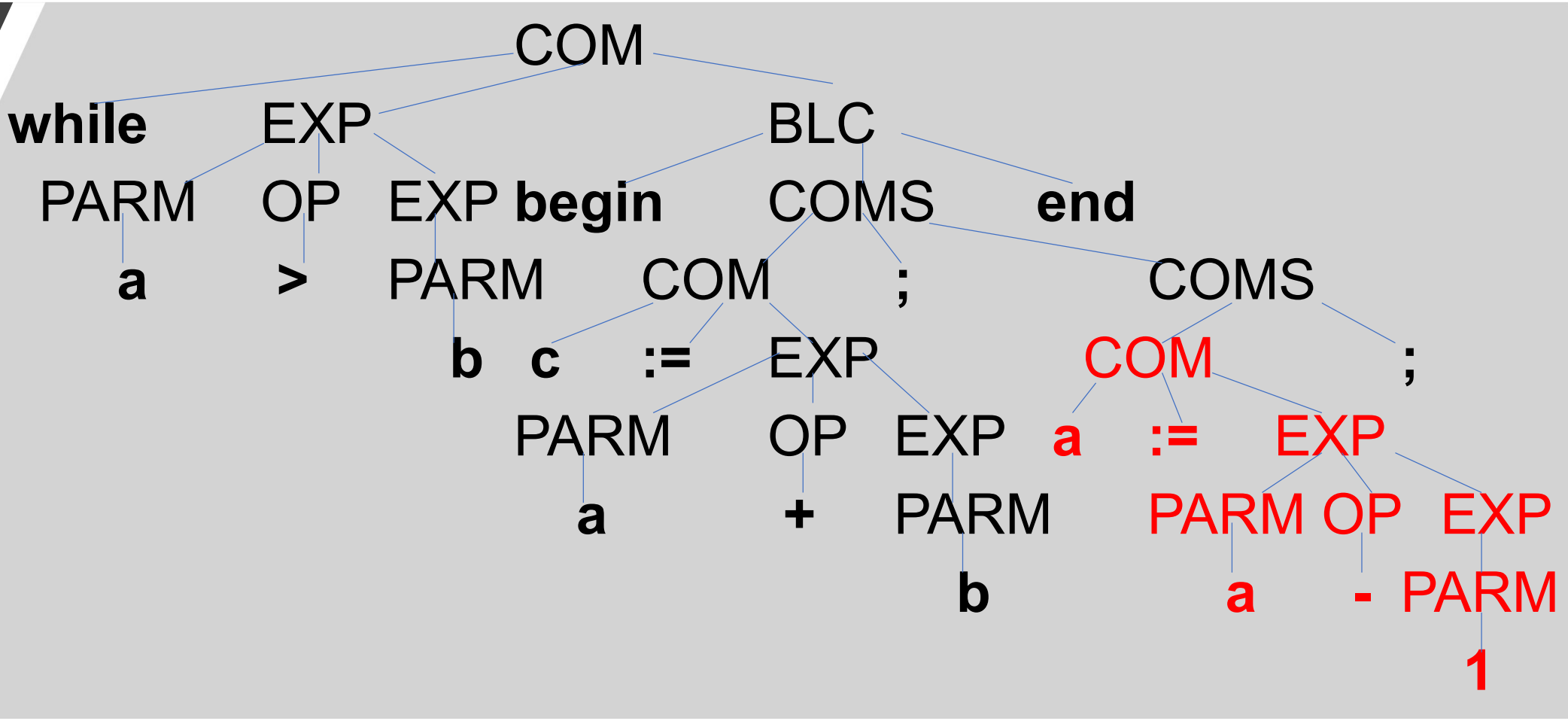
JMP | LABEL_INI | - | -

LBL | LABEL_FIM | - | -

Algoritmo – Exemplo



Algoritmo – Exemplo



Algoritmo – Exemplo

COM -> **while** EXP BLOCO

LBL | LABEL_INI | - | -

instruções EXP

JMZ | LABEL_FIM | TEMP_EXP | -

instruções BLOCO

JMP | LABEL_INI | - | -

LBL | LABEL_FIM | - | -

Algoritmo - Exemplo

```
...
while a > b
begin
        c := a + b;
        a := a – 1;
end
...
```

LBL	label1	-	-
GTR	temp1	a	b
JMZ	label2	temp1	-
ADD	temp2	a	b
ATR	c	temp2	-
SUB	temp3	a	1
ATR	a	temp3	-
JMP	label1	-	-
LBL	label2	-	-