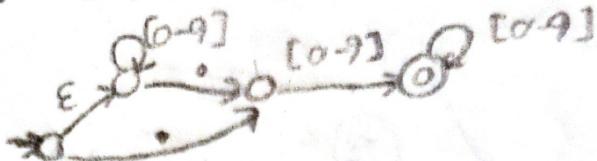
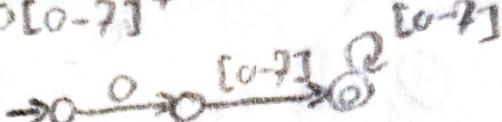


Teste EN 2025 (Teste reconstruído de memória, pode haver coisas erradas)

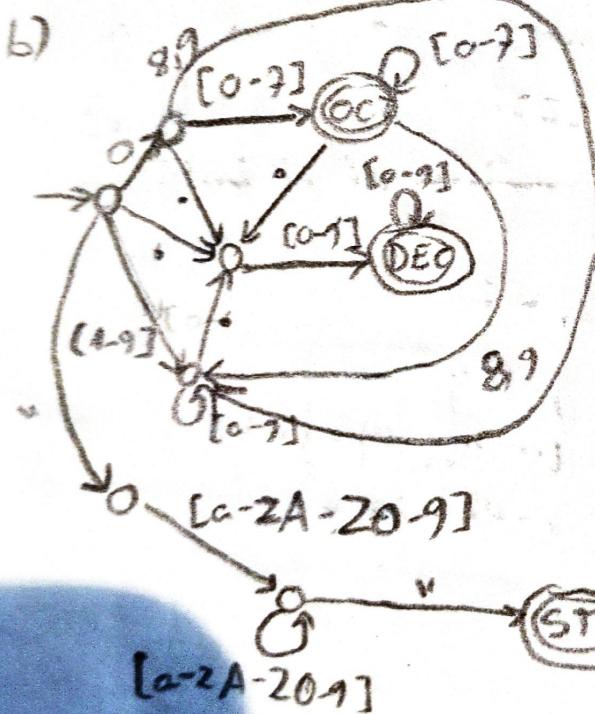
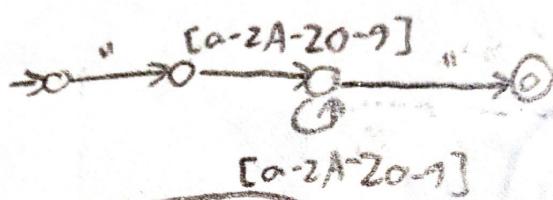
a) DEC $[0-9]^*$, $[0-9]^+$



OCT $0[0-7]^+$



STR "a-zA-Z0-9"



c) 0 "a" 00 "b" 00.48.7 "5.40"

ERR: 0 "

STR: "a"

OCT: 00

STR: "b"

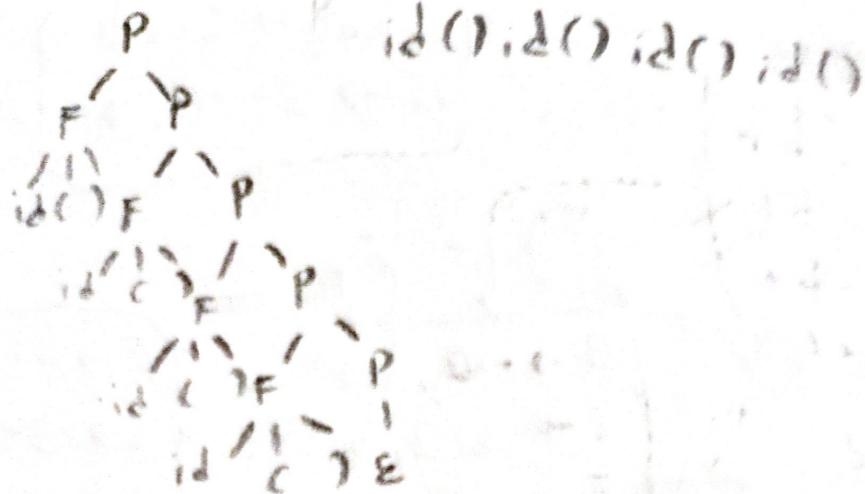
DEC: 00.48

DEC: .7

ERR: 5.

DEC: 5.40 ERR: "\$"

2 a)



b)

Expandido a gramática, temos:

Uma gramática NL é LL(1) se,

$P \rightarrow id()P$ para algum $A \rightarrow \alpha \mid B$
 $\boxed{id \mid var}P$ $First(\alpha) \cap First(B) = \emptyset$

$\mid \epsilon$ isso verifica-se nesta gramática
 para

$First(id()P) \cap First(id \mid var P) = \{id\}$
 então a gramática NL é LL(1)

$$1 \quad P \rightarrow id P'$$

$$2 \quad 1 \epsilon$$

$$3 \quad P' \rightarrow ()P$$

$$4 \quad 1 \mid var P$$

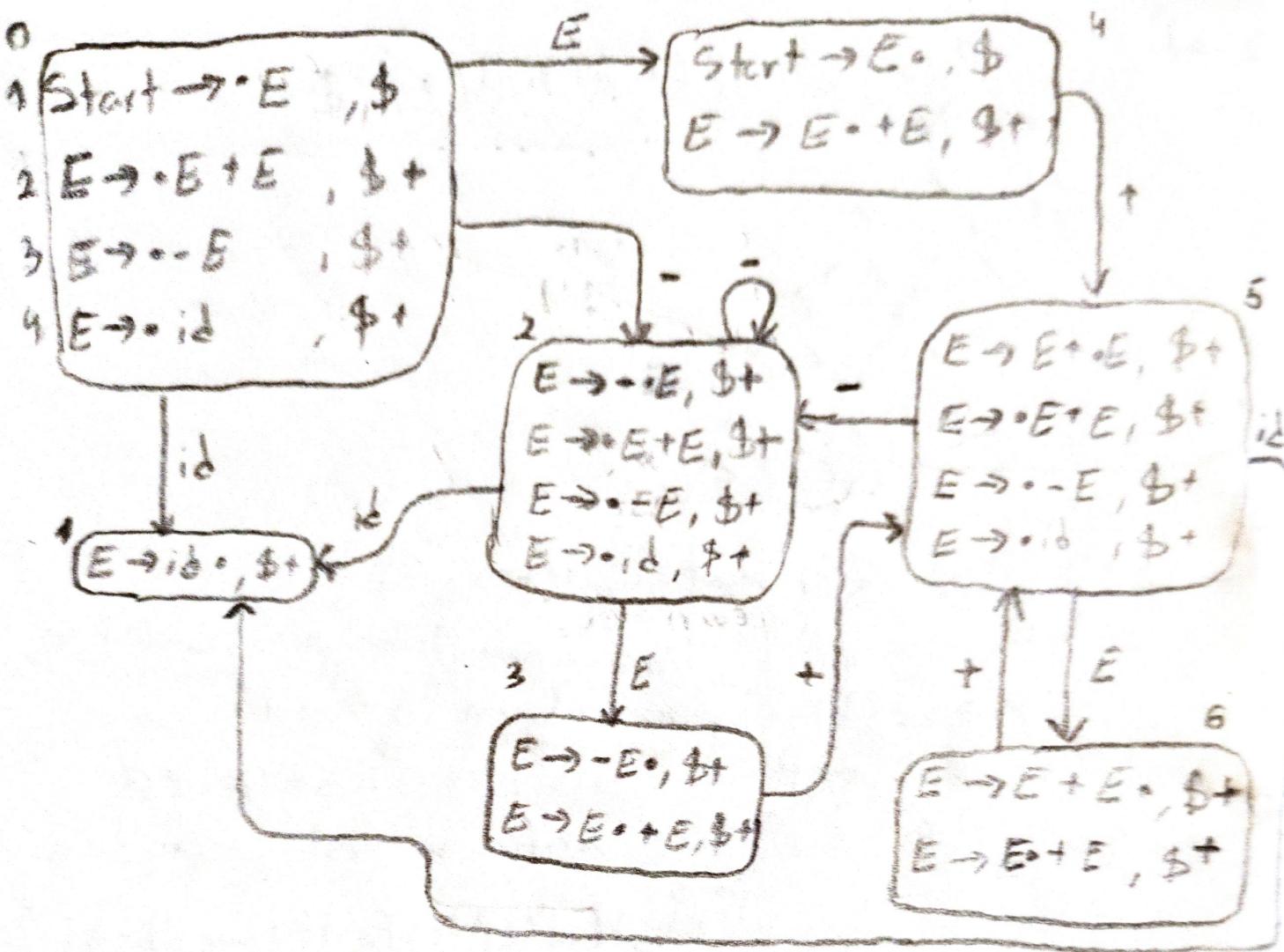
$$First(P) = \{id, \epsilon\}$$

$$First(P') = \{(), var\}$$

$$Follow(P) = \{\$\} \cup Follow(P) = \{\$\}$$

$$Follow(P') = Follow(P) = \{\$\}$$

	()	id	var	\$
P				
P'	3	1	4	



b)

	ACTION GOTG			
	+	-	id	\$
0	s2	s1		r9
1	r4			r4
2	s2	s1		r3
3	r3			r5
4	r5			Acc
5	s2	s1		r6
6	r3 / r5			r2

$T[3, +] = r3 \} \text{ Prioridade}$
 $T[6, +] = r2 \} \text{ Associativa à esquerda}$

c) $\text{Start} \rightarrow E \quad \{ \$\$ = \text{newnode}(\text{Program}, \text{NULL}),$
 $\text{addchild}(\$ \$, \$ 1); \}$

$E \rightarrow E + E \quad \{ \$ \$ = \text{newnode}(ADD, \text{NULL}),$
 $\text{addchild}(\$ \$, \$ 1),$
 $\text{addchild}(\$ \$, \$ 3); \}$

$1 - E \quad \{ \$ \$ = \text{newnode}(MIN, \text{NULL}),$
 $\text{addchild}(\$ \$, \$ 2); \}$

$1 \text{ id} \quad \{ \$ \$ = \text{newnode}(ID, \$ 1, \text{lexeme}); \}$

4 a)

sqr^t:

1 $x := \text{param}[0]$

2 $err := 0,00000G$

3 $div := 2$

4 $try := x$

5 $t_1 := try / div$

6 $t_2 := x / t_1$

7 $next := t_1 + t_2$

8 $t_3 := next - try$

9 if FALSE $t_3 < err$ goto 13

10 $t_4 := try - next$

11 if FALSE $t_4 < err$ goto 13

12 goto 15

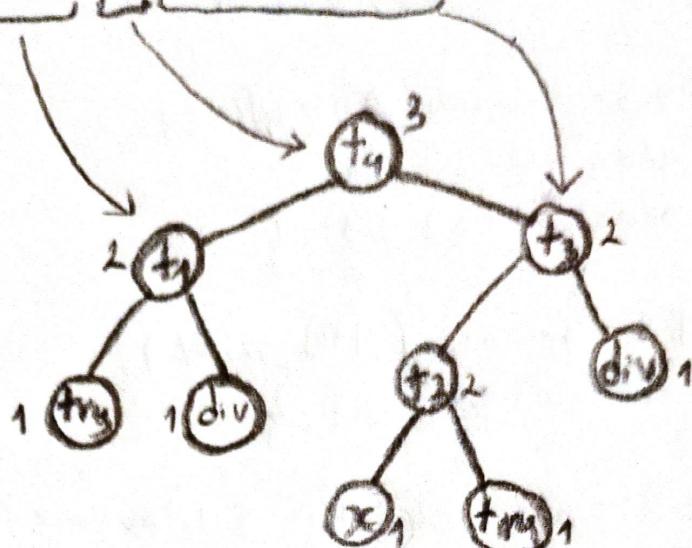
13 $try := next$

14 goto 5

15 $t_5 := try$

16 return try

b) $(try / div) + (x / try / div)$



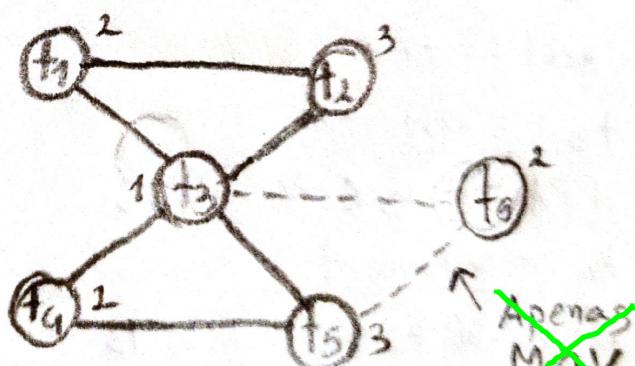
O número de Ershov da root é 3, portanto são precisos, no mínimo 3 registros.

Como?

Lifetimes

$$\begin{array}{l}
 t_1 := try \\
 t_2 := div \\
 t_3 := t_1 / t_2 \\
 t_4 := x \\
 t_5 := t_4 / t_3 \\
 t_6 := t_3 + t_5
 \end{array}
 \quad \left| \begin{array}{c} t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \end{array} \right| \quad \left| \begin{array}{c} t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \end{array} \right|$$

Podemos fazer o grafo de interferência



Colorindo de forma greedy, começando em t_3 , temos:

Node	Cor
t_1	2
t_2	3
t_3	1
t_4	2
t_5	3
t_6	2

Então, com os registros R1, R2 e R3 podemos recresver a expressão como:

MOV R2 try
 MOV R3 div
 DIV R1 R2 R3
 MOV R2 x
 DIV R3 R2 R1
 ADD R2 R1 R3

4 e) Return value, old stack pointer, frame pointer, parâmetros, espaço para spilling, variáveis locais e temporárias

Alternativas :- Passar parâmetros por registros

✓ velocidade

X maior chance de spilling

- Passar parâmetros pelo heap:

X não é otimizado pois "double" é de tamanho estático

- Saved registers; caso haja registros convencionados "callee-saver"

- Guardar return address num registro; neste caso é adequado porque se trata de uma função folha

- Usar display, lambda lifting ou apenas um static link para voltar à frame da caller function.

- etc ...