

CURS 1 - Tehnici de compilare

Limbaje de calculator

- 1) Limbaje cod mornă (nivel 0)
 - 2) Limbaje de asamblare (nivel 1)
 - adrese hexazecimale
 - adrese la registri
 - memorie pentru o structură
 - 3) Limbaje de nivel înalt (nivel 2)
 - 4) Limbaje specializate

PROCESARE DE LIMBAJE:

Compilatorul

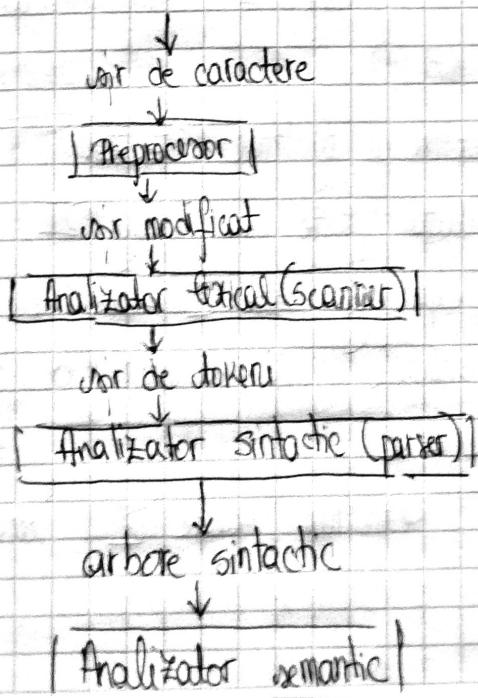
program $\xrightarrow{\text{surse}}$ compilatorul $\xrightarrow{\quad}$ programul final
(limbaj de înțelit) \qquad (cod intermediar sau cod machine)

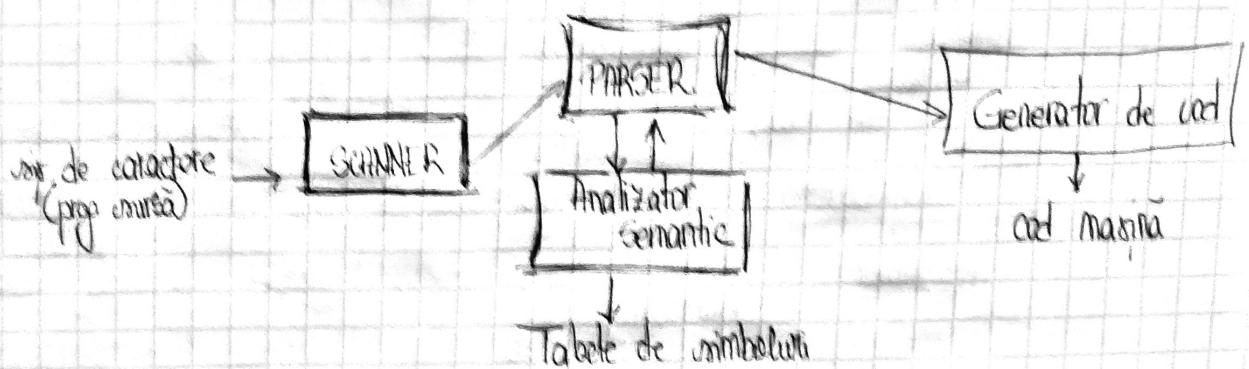
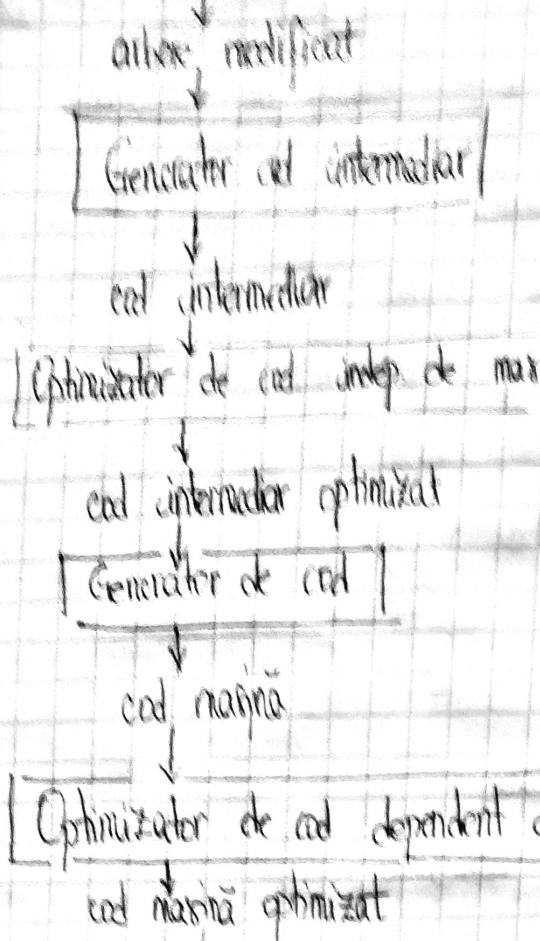
Interpretatorul: translatează direct instrucțiunile programului cursă

Asamblador: program writes in
climbey de
asambleo → Asamblador → cod máquina

Preprocessor : Linker
Loader

FAZELE COMPILĂRII





ANALIZA LEXICALĂ: constă în tracaj. emisiunii de caractere coresp. progr. cursă antr-un sir de tokeni

Token = unitate lexicală (ex. identificatori, cuv. cheie, operatori, comentarii, separatori)

- detectă unități dintr-un cu ajutorul RE-TEX

Expresie regulată peste un limbaj?

0 - expr. reg. peste Σ care descrie limbajul \emptyset

1 - curşorul de lungime 0 $\{ \}$

$\forall a \in \Sigma$

$\{ a \}$

$S \xrightarrow{*} p, q$ expr. reg. peste Σ care descriu ab. $L(p), L(q)$

Autor

$$\begin{aligned}
 p_1 q_2 \text{ este } w_1 \text{ parte } \subseteq \text{este} \text{ din } L(p) \cup L(q) \\
 p_1 q_2 \\
 p_1 q_2 \\
 (p) \\
 (q)
 \end{aligned}$$

Proprietăți:

$$L_1 L_2 \subseteq \Sigma^* \quad L_1, L_2 = \{w_1, w_2\} \text{ cu } w_1 \in L_1, w_2 \in L_2$$

$$L = \Sigma^*, L^* = \{a, b, \lambda, aa, ab, ba, bb, aaa\}$$

Automate finite determinante (neDeterminante) (nedet. cu λ -transiții)

$$A = (Q, \Sigma, \delta, q_0, F)$$

Q - mulțime finită de stări

Σ - alfabetul autonotăllui

$$\delta: Q \times \Sigma \rightarrow Q \quad \text{funcția de tranziție}$$

$$\delta: Q \times \Sigma \rightarrow 2^Q \quad \delta: Q \times (\Sigma \cup \{\lambda\}) \rightarrow 2^Q$$

$q_0 \in Q$, stare initială

$F \subseteq Q$, stări finale

Descriere instantanee a lui A:

$(p, w) \quad p \in Q$, stare curentă

$w \in \Sigma^*$ apărut recent sau nu

Migare a lui A:

$$(p, \alpha w) \mapsto (q, w) \iff q = \delta(p, \alpha)$$

$p, q \in Q$, $\alpha \in \Sigma$, $w \in \Sigma^*$, \mapsto zero sau mai multe migări

$$L(A) = \{ w \in \Sigma^* \mid (q_0, w) \mapsto (q, \lambda), q \in F \}$$

$$\} \text{ Pd. } q \in Q, \langle q \rangle = \{ s \in Q \mid \exists p_0, p_1, \dots, p_n \in Q, p_0 = p, p_n = q, p_i \in \delta(p_{i-1}, \alpha) \text{ } i=1, \dots, n \}$$

λ -echidere de Q

Obs:

$$g \in \langle g \rangle$$

$$\langle\langle g \rangle\rangle = \langle g \rangle$$

$$M \subseteq Q$$

$$\langle M \rangle = \bigcup_{g \in M} \langle g \rangle$$

$$L_{AFD} = L_{AFN} = L_{AFN^2} = L_3 = L_{ExpRep}$$

Scannerul (analizatorul lexical) este o subiectă:

- scanăază fișierul cu programul cursă
- elimină (ignoră) comentarii, spații etc.
- furnizează parserului tokenul curent
- semnalizează eventuale erori lexicale

DESCRIERE LEXICALĂ:

Def: Numim descriere lexicală poste Σ o exp. reg. de forma:

$$e = (e_1 | e_2 | \dots | e_n)^*$$

e_i exp. reg. poste Σ (care reprezintă o cat de tokeni;
 Σ mult. caract. permise)

Def: Fie $w \in L(e)$. Numim interpretator al lui w un sir de forma:

$$(w_1, i_1), \dots, (w_t, i_t)$$

$$w = w_1 \dots w_t, 1 \leq i_1, \dots, i_t \leq m$$

$$w_j \in L(e_j) \quad j = 1, \dots, t$$

Def: Spunem că e este ambiguă dacă $\exists w \in L(e)$ care are 2 interpretări distincte.

Ex: $P_xuma = (\text{identifier} | \text{int} | \text{comment} | \text{slash} | \text{spaces} | \text{semicolon} | \text{equal})^*$

$$\text{letter} = ("A" | "B" | "C" | "D")^*$$

$$\text{digit} = ("0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9")^*$$

$\text{int} = \text{digit digit}^*$
 $\text{comment} = "/*" (\text{not star}) | "*/" (\text{not slash}) \vee /*/$
 $\text{slash} = "/"$
 $\text{space} = (" ")^*$
 $\text{semicolon} = ";"$
 $\text{equals} = "="$
 $L(\text{not star}) = \Sigma - \{ /* \}$
 $L(\text{not slash}) = \Sigma - \{ "/" \}$

$$\Sigma = \{ A, B, C, D, \dots, 9, +, 3, 4, \dots, \}$$

+ , 13

NPC

$(A, \text{identif})$ $(Bc, \text{identif})$
 $(AB, \text{identif})$ $(C, \text{identif})$
 $(A, \text{identif})$ $(B, \text{identif})$ $(\alpha, \text{identif})$
 $(ABC, \text{identif})$

Def.: Spunem că o des. lexicală $e = (e_1 e_2 \dots e_n)^*$ este orientată dreapta dacă pentru $(V) w \in L(e)$ și pt. $V(w_{i+1}), \dots, (w_i, v)$ a lui w , w_j este cel mai lung prefix al vîrfului $w_j w_{j+1} \dots w_t$.

50% lab
2 proiecte

In: Se poate decide algoritmic dacă o deservire lexicală este orientată dreapta.

forma:

50% examen
cu materiale
- exercitii

care are

$\text{int} \neq \text{int} \text{ equals}^*$

$\text{int} \neq \text{int}$