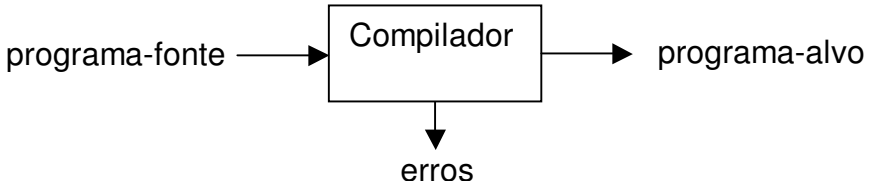
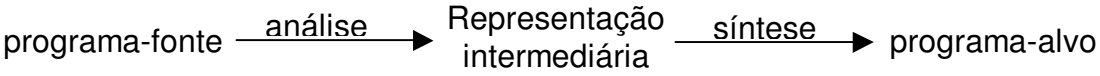
I. Introdução

Um compilador é um programa que lê outro programa escrito em uma linguagem (fonte) e o traduz para outra linguagem (alvo).

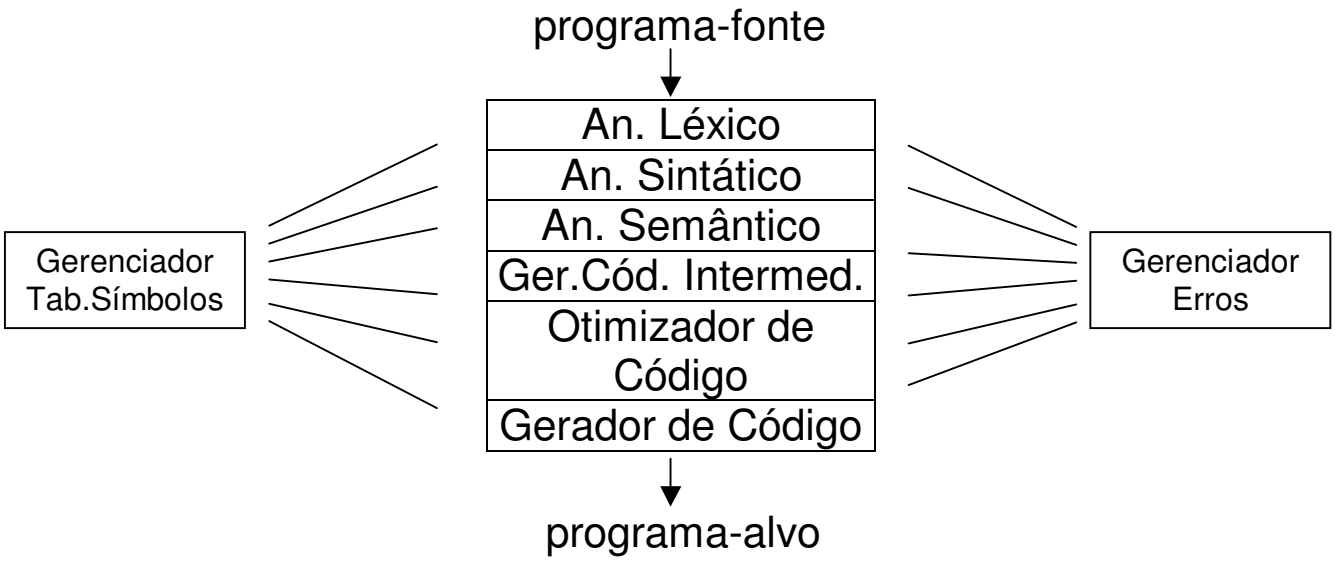


Modelo Análise – Síntese



a) Análise Linear (A. Léxica): a seqüência de caracteres do programa-fonte é lida da esquerda para a direta em busca de elementos da linguagem (tokens).
b) Análise Hierárquica (A. Sintática): os tokens são agrupados em estruturas hierárquicas segundo as regras de produção da gramática.
c) Análise Semântica: os componentes do programa são checados para se garantir sentido às estruturas.

Fases do Compilador



a) Analisador Léxico – Responsável por identificar lexemas no programa-fonte, verificando a correção "ortográfica" do código e os tokens correspondentes.

Ex: A rosa é vermelha
- A → artigo
- Rosa → substantivo
- é → verbo
- Vermelha → adjetivo

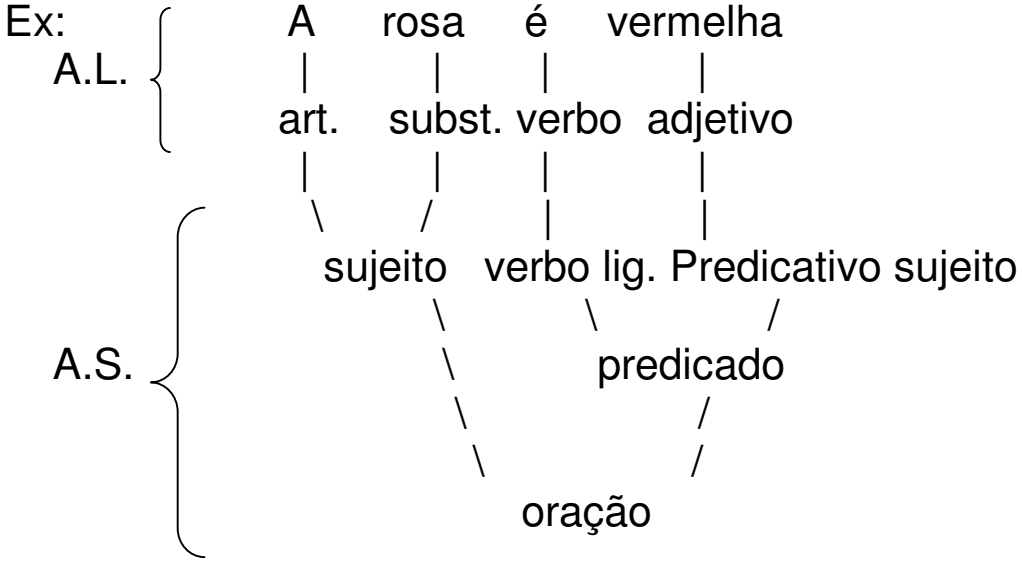
Ex: A roza é vermelha
- roza ? (erro léxico)

Ex: A := 10.0 {
- A→ identificador
- := → símbolo reservado de atribuição
- 10.0 → constante

Ex: A:=10.0.0 { 10.0.0 ? (erro léxico)

Pergunta: Quais os itens presentes no programa-fonte ? Eles estão grafados corretamente ?

b) An. Sintático – Responsável por agrupar os itens léxicos (tokens) em estruturas hierárquicas. Cada token identificado pelo An. Léxico é armazenado na tabela de símbolos e sua posição verificada através da gramática.



Ex:

Rosa	a	é	vermelha
subst.	art.	verbo	adj.

A. L. esta ok !

Porém há um erro sintático, pois o sujeito deve ser formado pelo artigo seguido do substantivo e não o contrário.

Ex:

G: S→ id := E
E→ id + F
F→ const * id | id * id

A	:=	B	+	10	*	C
id	:=	id	+	const.	*	id

A. L. e A. S. ok !

Ex:

A	:=	B	C	+
id	:=	id	id	+

A. L. ok !

A. S. não ok !

/

Pergunta: Os itens (tokens) estão na ordem correta ?

c) An. Semântico: Responsável por verificar a coerência das estruturas, quanto ao seu contexto, tipos de operandos e operadores, declaração de variáveis e constantes, entre outros.

Ex: A rosa é vermelho A.L. e A.S. Ok !

Mas há um erro de concordância !

Ex: VAR A: INTEGER;

a:= 10.0;

Mas há incompatibilidade de tipos !

Pergunta: Os meus elementos e estruturas fazem sentido dentro do contexto ?

d) Gerenciador da Tabela de Símbolos: Responsável por registrar informações sobre os tokens do programa fonte, com relação a tipos, escopo de validade, parâmetros de funções, métodos de passagem dos parâmetros, entre outros. Estas informações são usadas para conversão de tipos e alocação na memória na fase de geração de código.

e) Gerenciador de Erros: Responsável por tratar erros durante as fases de compilação, de forma a permitir que esta prossiga e possibilite a análise de outras partes do p.fonte.

f) Gerador de Código Intermediário: Responsável por representar as estruturas sintáticas encontradas na forma de um programa para uma máquina abstrata. Esta representação intermediária facilita a conversão do p. fonte para o p. alvo.

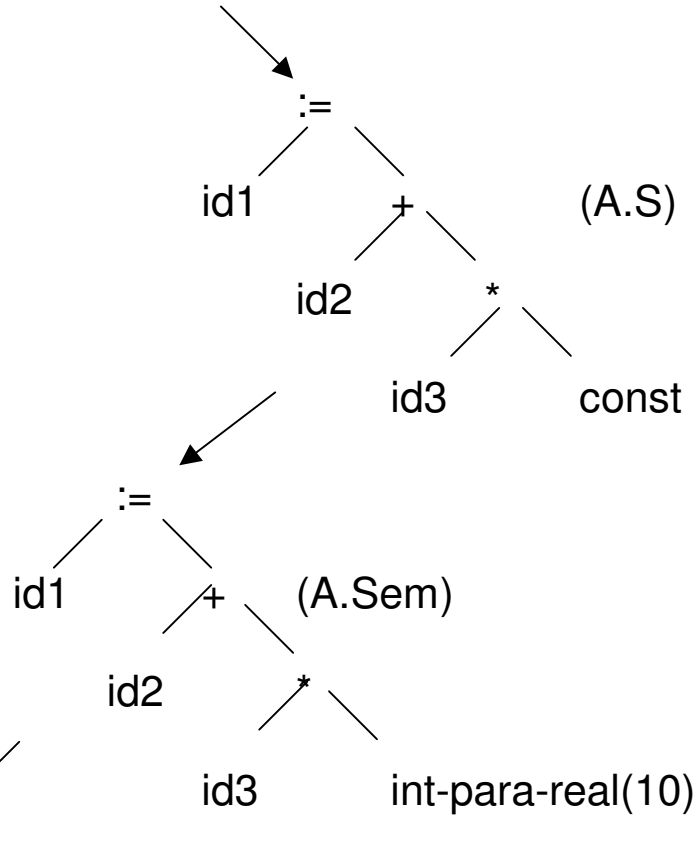
g) Otimizador de Código: Permite melhorar a representação intermediária, com o objetivo de produzir um código final mais eficiente.

h) Gerador de Código: Realiza a conversão do código intermediário para o código alvo.

Ex:

A := B + C * 10 (P.F.)

id1 := id2 + id3 * const (A.L.)



temp1 := int-para-real(10)
temp2 := id3 * temp1 (G.C.I.)
temp3 := id2 + temp2
id1 := temp3

(O.C.)

temp1 := id3 * 10.0
id1 := id2 + temp1

(G.C.)

```
movf    id3, R2
mulf    #10.0, R2
movf    id2, R1
addf    R2, R1
movf    R1, id1
```

Agrupamento de Fases

Blocos

As diversas fases do processo de compilação podem ser agrupadas em dois blocos:

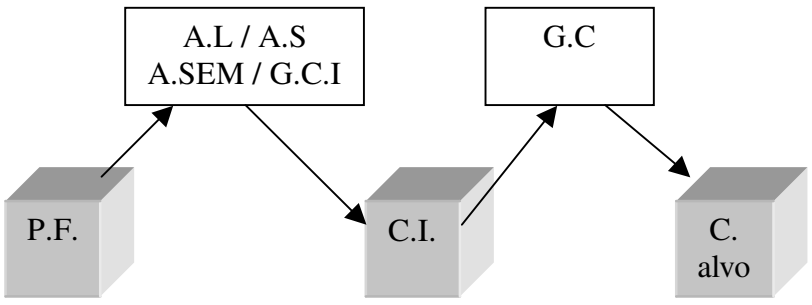
1. Front-End (Vanguarda) – inclui as fases que dependem fundamentalmente da linguagem: A.L., A.S., A.SEM., G.C.I.
2. Back-End (Retaguarda) – inclui a maior parte da otimização de código e o G.C., fases que dependem fundamentalmente da máquina-alvo.

Para se mudar de máquina-alvo, só há necessidade de alteração do back-end, ficando o front-end inalterado.

Passos

O número de passos do compilador está relacionado ao número de vezes em que é necessário ler o arquivo que contém as instruções do programa (mesmo que modificado). Quanto menor o número de passos menos tempo será gasto com a leitura dos arquivos, mas o processo pode se tornar mais complexo e exigir mais memória para se manterem todas as representações carregadas simultaneamente.

Ex.: Compilador de Dois Passos



6. Sistemas de Programação

Tradutor – Programa que converte um programa-fonte para um programa-alvo. Em um processo de tradução, a conversão acontece em um instante distinto da execução do programa-alvo.

Interpretador – No processo de interpretação, o programa-fonte é convertido e imediatamente executado, sem geração do arquivo-objeto.

Montadores (ASSEMBLERS) – Programas que convertem um programa-fonte em ASSEMBLY para um programa-objeto, convertendo os mnemônicos em linguagem de máquina. O tratamento de rótulos pode exigir que o processo de montagem ocorra cm dois passos (retro-correção).

Linkeditor – Procedimentos e programas podem ser compilados separadamente para arquivos-objetos e depois ligados para se transformarem em um módulo de carga absoluta. Um arquivo-objeto possui: campos de identificação e referências, além do código propriamente dito. O linkeditor atualiza referências de endereços para que o programa executável esteja consistente e possa ser carregado com um módulo único.