Compiladores
Aula 01 - Parte I

Compiladores

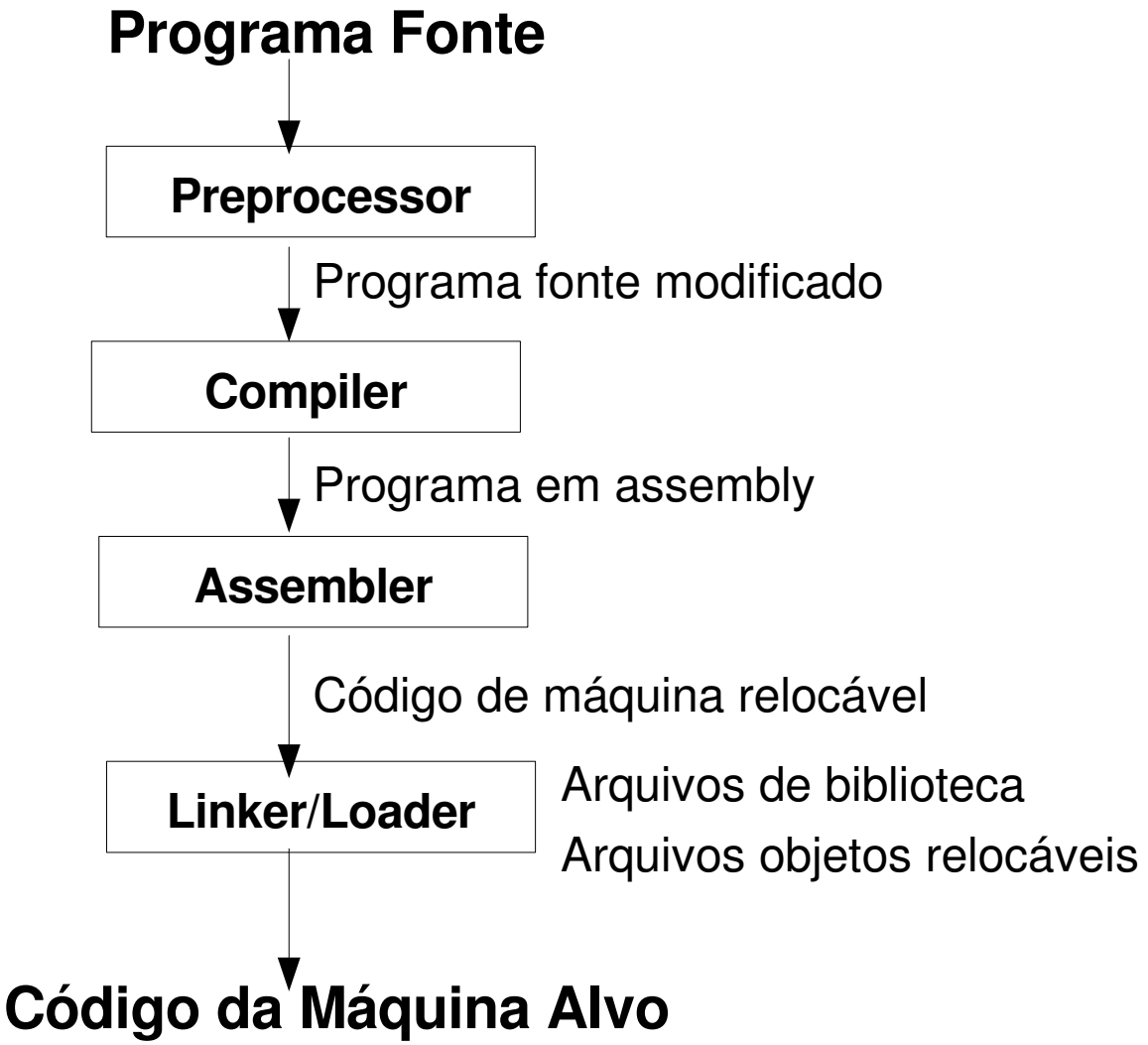
- Linguagens de programação são notações que permitem descrever como programas devem executar em uma máquina
- Mas, antes do programa executar, deve primeiro ser traduzido em uma forma que pode ser inteligível por um computador
- O elemente que permite essa tradução é o **compilador**

Compiladores

- Basicamente, um compilador é um programa que "lê" um programa escrito em uma linguagem – **linguagem fonte** – e o traduz em um programa equivalente em uma **linguagem alvo** (linguagem de máquina)
- Uma função importante do compilador diz respeito à detecção e informe de erros que detecta no programa fonte durante o processo de tradução

Sistema de Processamento

- Em adição ao compilador, diversos outros programas podem ser requeridos para criar um programa na linguagem alvo



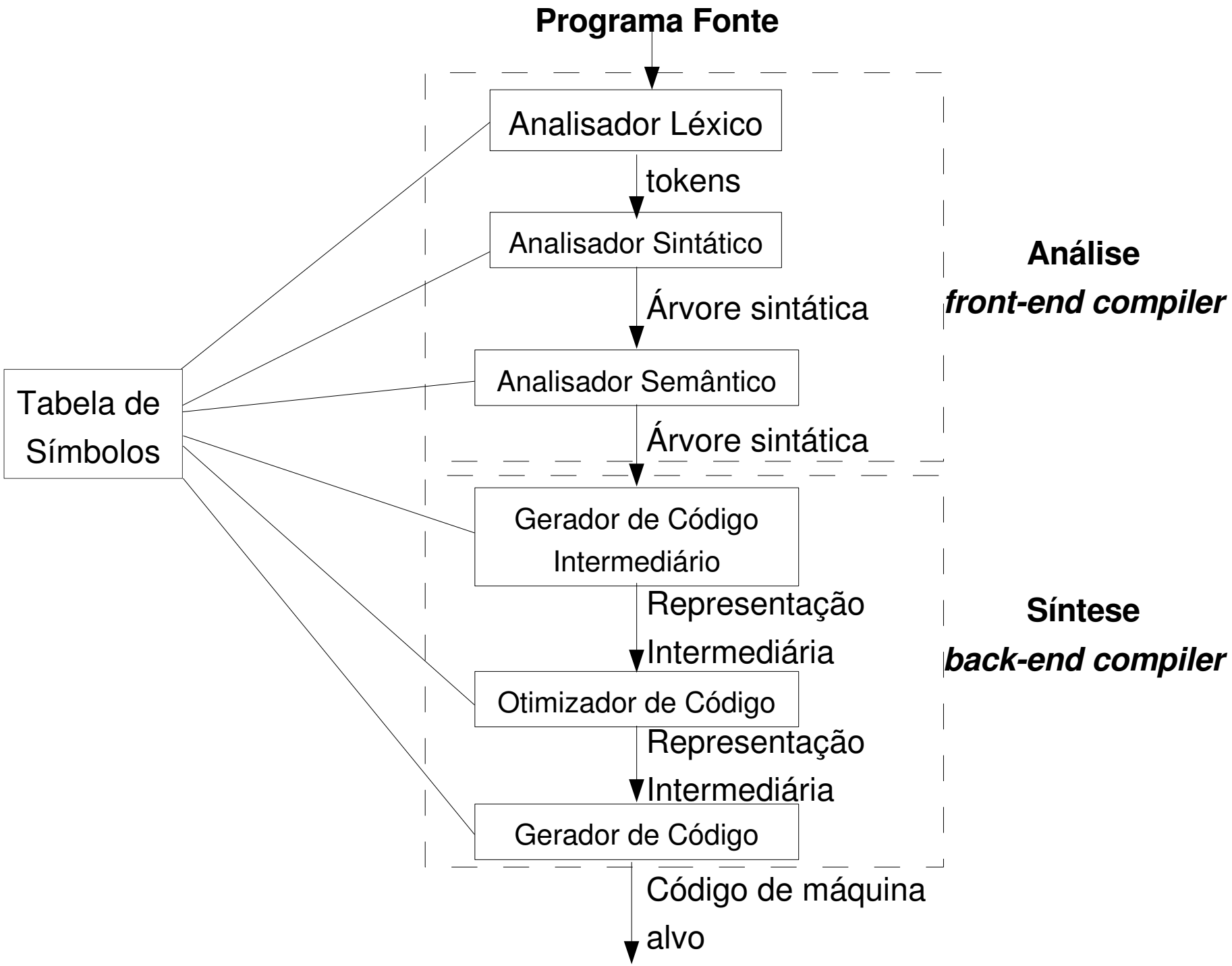
Sistema de Processamento

- Preprocessador: coleta os módulos, em arquivos separados, que compõem o programa fonte; Expande macros para as construções da linguagem fonte
- Compilador: a partir do resultado do preprocessador, produz um programa em linguagem assembly
- O assembler (montador) produz código de máquina relocável a partir da saída do compilador
  - **Relocação é o processo de atribuir endereços de de memória para as várias partes do programa e ajustar o código e dados no programa para refletir os endereços atribuídos**
- O linker resolve endereços de memória externos ao programa fonte (referências a arquivos externos e/ou bibliotecas)
- O loader reserva espaço para armazenar o programa (completo) na memória e configura bits de proteção

A estrutura de um compilador

- Um compilador pode ser dividido em dois componentes principais: análise e síntese
- A **análise** (front-end) recebe o programa fonte, verifica as partes constituintes desse programa e impõe uma estrutura gramatical sobre elas
  - Cria uma representação intermediária desse programa fonte;
  - Coleta informações sobre o programa fonte e as armazena em uma Tabela de Símbolos;
  - Fornece mensagens indicando possíveis erros na sintaxe e/ou semântica do programa
- A **síntese** (back-end) recebe uma representação intermediária do programa e constrói o programa para a máquina alvo
- O processo de compilação opera em uma seqüência de fases, onde, em cada fase, a representação do programa é modificada

A estrutura de um compilador



A estrutura de um compilador

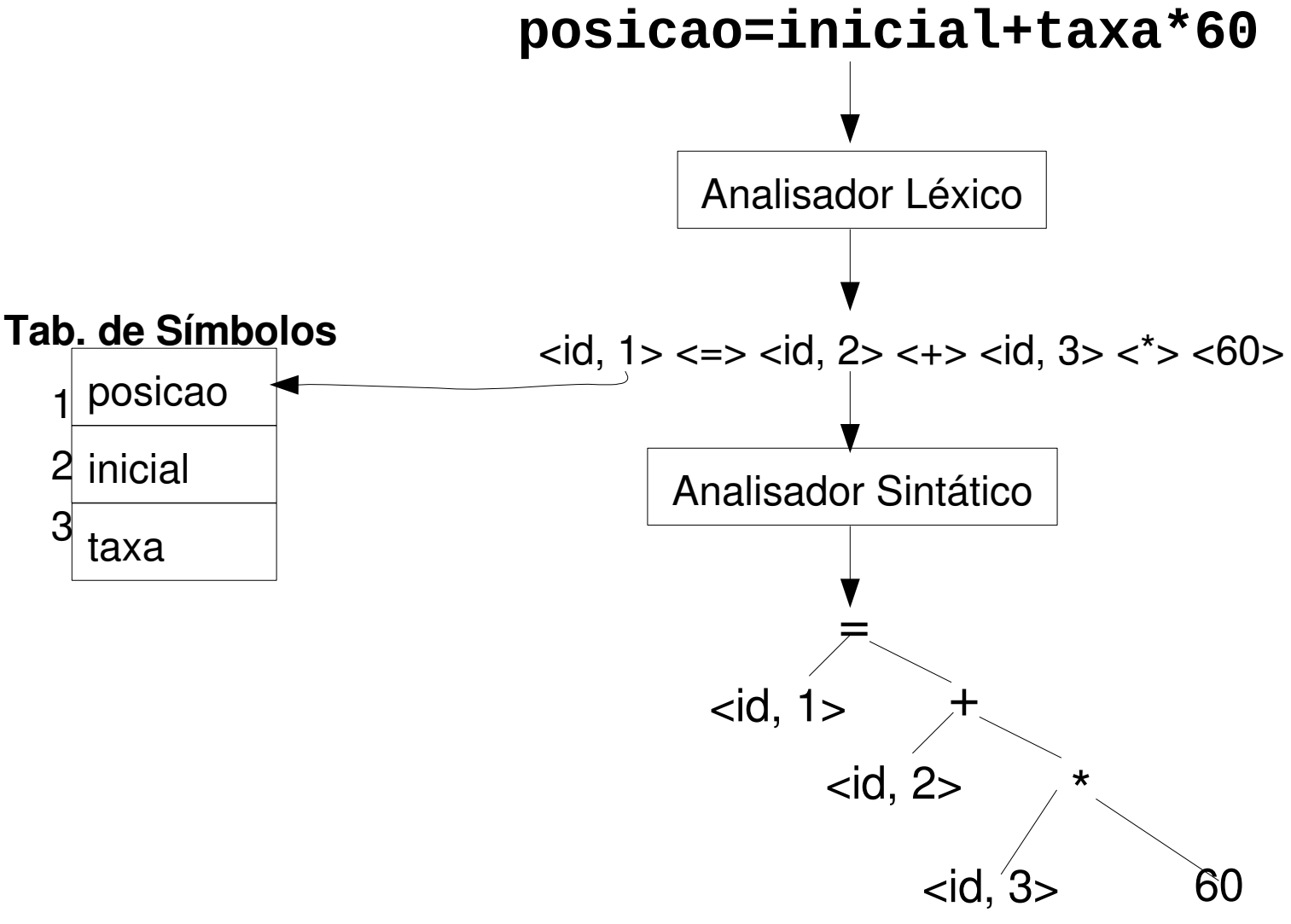
- Tabela de Símbolos: É uma estrutura de dados para armazenamento e recuperação de identificadores.
  - Contém um registro (nome, atributo) para cada identificador.
  - Deve ser projetada para pesquisar e recuperar registros rapidamente.
    - Comumente implementada como tabelas de dispersão
- Analisador Léxico (*scanner*): Lê o fluxo de caracteres do programa fonte e agrupa esses caracteres em lexemas. Para cada lexema, o analisador produz um token na forma **<nome_token, attributo_token>**
- Analisador Sintático (parser): Usa os tokens para produzir uma representação na forma de árvore a fim de representar a estrutura gramatical do fluxo de tokens

A estrutura de um compilador

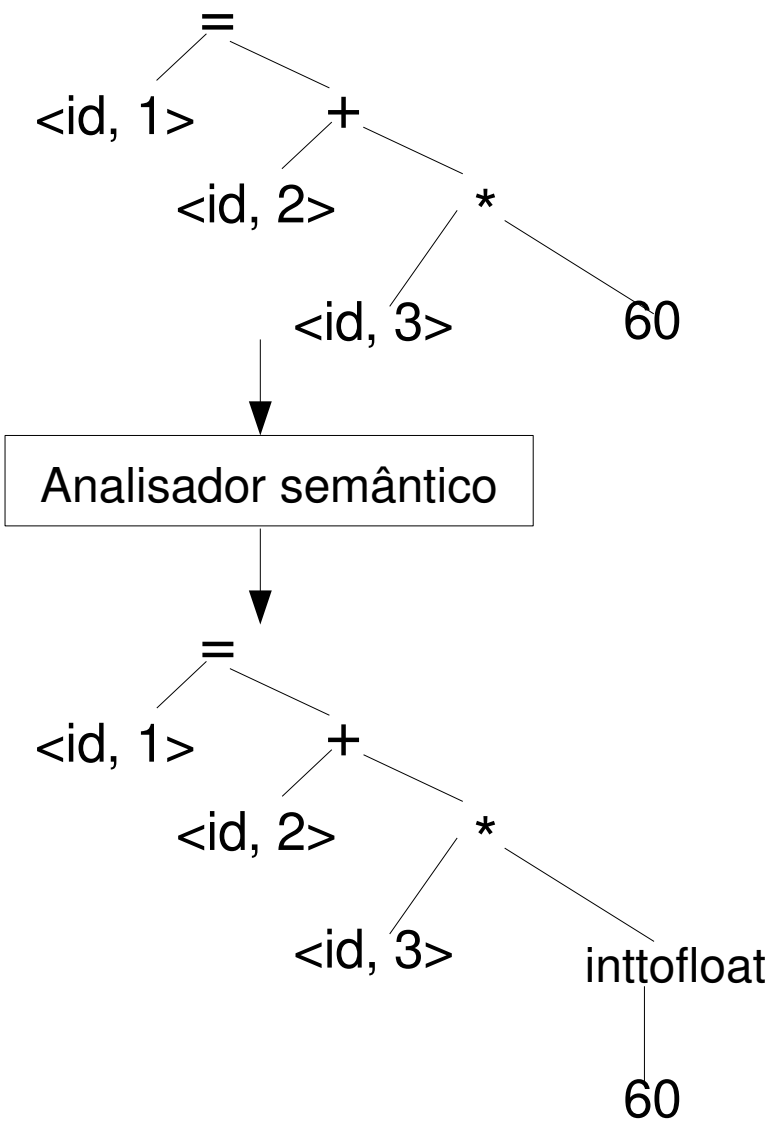
- Analisador Semântico: Utiliza a árvore sintática e a Tab. De Símbolos para verificar a consistência semântica do programa para a definição da linguagem
    - **Uma atividade importante aqui é a verificação de tipos**
- Geração de código intermediário: Produz uma representação intermediária do programa fonte a partir da árvore sintática
    - **Uma representação intermediária bem conhecida é a representação de código de três endereços**
- Otimização de código: Emprega algoritmos que procuram melhorar a qualidade do código intermediário visando gerar um código de máquina melhor (mais rápido e/ou menor)
    - **Atualmente, esta fase é cada vez mais complexa devido à complexidade das arquiteturas de computadores**
- Geração de código: Recebe como entrada e representação intermediária e mapeia esse programa para a linguagem alvo (código de máquina) da máquina que deverá executar o programa

Exemplo

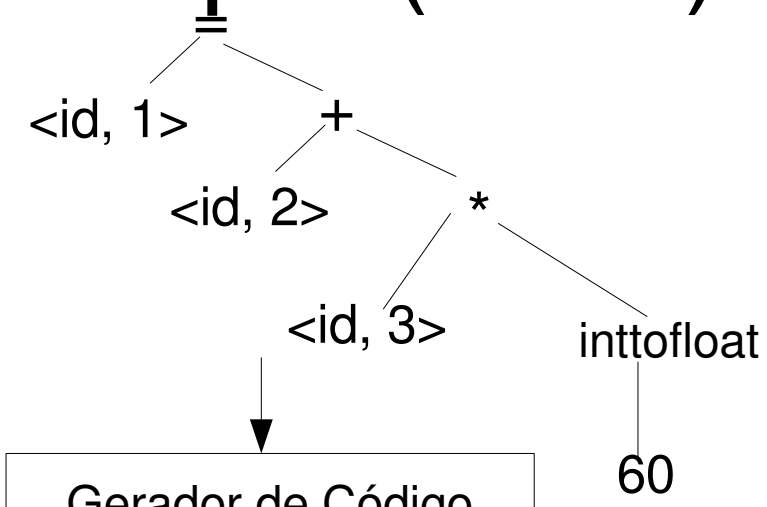
- Exemplo: Resultado da compilação do trecho de código a seguir em cada fase do compilador:



Exemplom (cont.)



Exemplo (cont.)



Gerador de Código Interm.

t1=inttofloat(60)
t2=id3*t1
t3=id2+t2
id1=t3

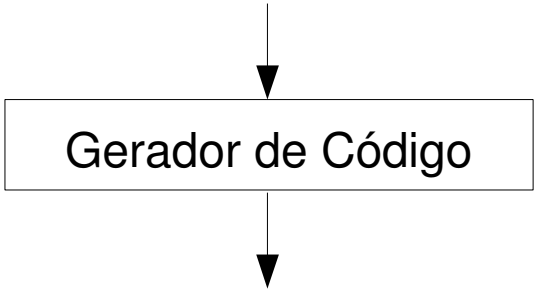
Otimizador de Código

t1=id3*60.0
id1=id2+t1

Exemplo (cont.)

t1=id3*60.0

id1=id2+t1




```
lwf r2, id3
mulf r2, r2, #60.0
lwf r1, id2
addf r1, r1, r2
swf r1, id1
```

Ferramentas para Construção de Compiladores

- Projetista/desenvolvedor de compiladores pode lançar mão de ferramentas de software para minimizar o tempo de desenvolvimento de um compilador. Algumas ferramentas comumente usadas incluem:

  - Geradores de parsers: produzem automaticamente analisadores sintáticos a partir de uma descrição da gramática da linguagem. Ex: bison, yacc
  - Geradores de scanners: produzem analisadores léxicos de uma descrição de expressões regulares. Ex: lex, flex
  - Tradutores dirigidos a sintaxe: produzem rotinas para manipular árvores sintáticas e gerar código intermediário
  - Geradores de gerador de código: produzem geradores de código a partir de regras para tradução da linguagem intermediária para a linguagem alvo
  - Analisadores de fluxo de dados: conjuntos de algoritmos para coleta de informações sobre o fluxo de dados do programa
  - Ferramentas para construção de compiladores: fornecem um conjunto integrado de rotinas para construção de várias fases de um compilador. Ex: fenix

Observações finais

- A necessidade de utilização de compiladores aconteceu a partir da década de 50 com a linguagem FORTRAN
- Projeto da linguagem de programação e do compilador estão intimamente relacionados
- As fases de um compilador não são, necessariamente, seqüênciais
- Algumas fases podem não existir ➞ otimização de código
- O código objeto final pode ser gerado em linguagem de máquina ou então em linguagem de montagem (assembly); Neste último caso, será necessário ainda um montador
- Compiladores são de extrema importância para fazer arquiteturas de computadores utilizáveis
  - **Muitas arquiteturas utilizam compiladores para avaliar os conceitos arquiteturais antes do Hw ser efetivamente construído**