Compiladores

Professor: Flávio Márcio de Moraes e Silva

Curso: Ciência da Computação

Universidade de Itaúna

O que é um compilador?

- Software que traduz o texto que representa um programa para código de máquina capaz de ser executado pelo computador
- Contudo, um compilador pode ser muito mais do que isso...

Importância

- Papel importante no desenvolvimento de aplicações complexas em tempo útil
- Muitas das matérias abordadas são utilizadas por outras ferramentas:
  - Tradução de linguagens
  - Interpretação de descrições (ex.: html, *postscript*, *latex*, ficheiros *word*)
  - Procura de palavras ou frases por motores de busca

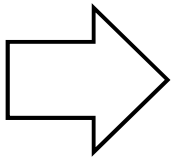
Dilema da Programação

- *Assembly* é fastidioso, erróneo, pouco produtivo, etc.
  - Embora possa permitir melhor desempenho
- Desenho de linguagens de alto-nível
- Como implementar a linguagem?
  - Interpretador (ler, interpretar e executar)
  - Compilador (traduzir)

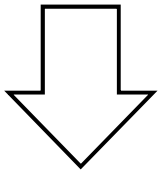
Compilador

Ponto de Origem

Código fonte descrito numa linguagem de alto-nível (ex.: C, C++, Java, Pascal, etc.)



Compilador



Ponto de Destino

Programa em Linguagem de Máquina

Ponto de Origem

```
Int sum(int A[], int N) {
  Int i, sum = 0;
  For(i=0; i<N; i++) {
    sum = sum + A[i];
  }
  return sum;
}
```

- Linguagem imperativa ou de alto nível (C/C++, Java, Pascal, etc.)
  - Estruturas de Estado
    - Variáveis do tipo int, float, char, etc
    - Variáveis do tipo array
    - Registros
    - Atributos de objetos (linguagens orientadas por objetos)
  - Computação
    - Expressões (aritméticas, lógicas, etc.)
    - Fluxo de controle (if, switch, etc.)
    - Estruturas de repetição (while, for, do .. While, etc)
    - Procedimentos (métodos nas linguagens orientadas por objetos)

Ponto de Destino

```
Sum:  Addi  $t0, $0, 0
      Addi  $v0, $0, 0
Loop: beq   $t0, $a1, End
      Add   $t1, $t0, $t0
      Add   $t1, $t1, $t1
      Add   $t1, $t1, $a0
      Lw    $t2, 0($t1)
      Add   $v0, $v0, $t2
      Addi  $t0, $t0, 1
      J       Loop
End:  jr      $ra
```

- Linguagem máquina que descreve o programa com base no ISA (*Instruction-Set Architecture*) do processador
  - Estruturas de Estado
    - Memória
    - Registros (de estado, de propósito geral, etc.)
  - Computação
    - Instruções do ISA (MIPS):
      - Lw $3, 100($2)
      - Add $3, $2, $1
      - Bne $2, $3, label
      - …

Ementa / Conteúdo Programático

- Introdução aos compiladores
  - Conceitos Iniciais
  - Análise léxica
  - Análise Sintática
  - Análise Semântica
  - Tradução para código intermediário
  - Otimização de código
  - Geração de código final
- Teoria da computação x Análise léxica
- Teoria da computação x Análise sintática
  - Top down
  - Bottom up

Conteúdo Programático

- Aulas Teórico-Práticas
  - Exemplos de projeto e programação de algumas etapas de um compilador.
  - Exercícios sobre determinados aspectos realacionados a costrução dos compiladores.

Avaliação

- De acordo com as normas da universidade
- Proposta: 3 provas (25 + 25 + 40 pontos)
- 2 listas de exercícios preparatórias para as provas (5 pontos cada lista)

Bibliografia

- [A] Aho, A., Sethi, R., Ullman, J., Lam, M.. *Compiladores: Princípios, Técnicas e Ferramentas*. Pearson.
- [B] Price, Ana Maria; Toscani, Simão Sirineo. Implementação de Linguagens de Programação: Compiladores. Bookman.

Contato com o Professor

- Flávio M. M. e Silva
  - fmms10@gmail.com
  - flaviomarcio@uit.br

Compiladores
Introdução

Prof. Flávio Márcio

Introdução

- **Linguagem ->** Meio de comunicação entre pessoas.
- **Linguagem de programação ->** Meio de comunicação entre pessoas e o computador

Evolução das linguagens de programação:

- Gerações:
- 1) Linguagem de Máquina
- 2) Linguagem simbólica (assembly)
- 3) Orientadas ao usuário (Profissionais de TI)
- 4) Orientadas a aplicação (Usuário comum)
- 5) Linguagens do conhecimento (Int. Artificial)

Tradutores de linguagens de programação:

- Sistemas que aceitam como entrada programas em linguagem de programação (linguagem fonte) e produzem como resultado um programa equivalente em outra linguagem (linguagem objeto).

Classificações

- **Montadores (Assemblers):** converte assembly para linguagem de máquina, na proporção de uma instrução para uma;
- **Macro-Assemblers:** mesmo tipo de conversão dos montadores, só que na proporção de uma instrução para várias;
- **Compiladores:** convertem linguagens de alto nível para assembly ou para linguagem de máquina;
- **Pré-compiladores ou pré-processadores:** convertem instruções em uma linguagem de alto nível estendida para instruções na linguagem de programação original. Surgiram para facilitar a extensão de linguagens.

Classificações

- **Interpretadores:** Recebem como entrada um programa (alto nível) ou seu código intermediário (pré-tradução) e produzem o “efeito de execução” do algoritmo original, sem porém mapeá-lo para linguagem de máquina, mas sim para a linguagem assembly da máquina em questão. São normalmente menores que compiladores e facilitam a construção de linguagens de programação complexas.
- **Desvantagem:** O tempo de execução de um programa interpretado é maior que o necessário para executar um programa objeto.
- **Vantagem:** Possibilita a utilização de uma linguagem em diferentes equipamentos de computação (Arquiteturas de computadores variadas).

Estrutura básica de um compilador

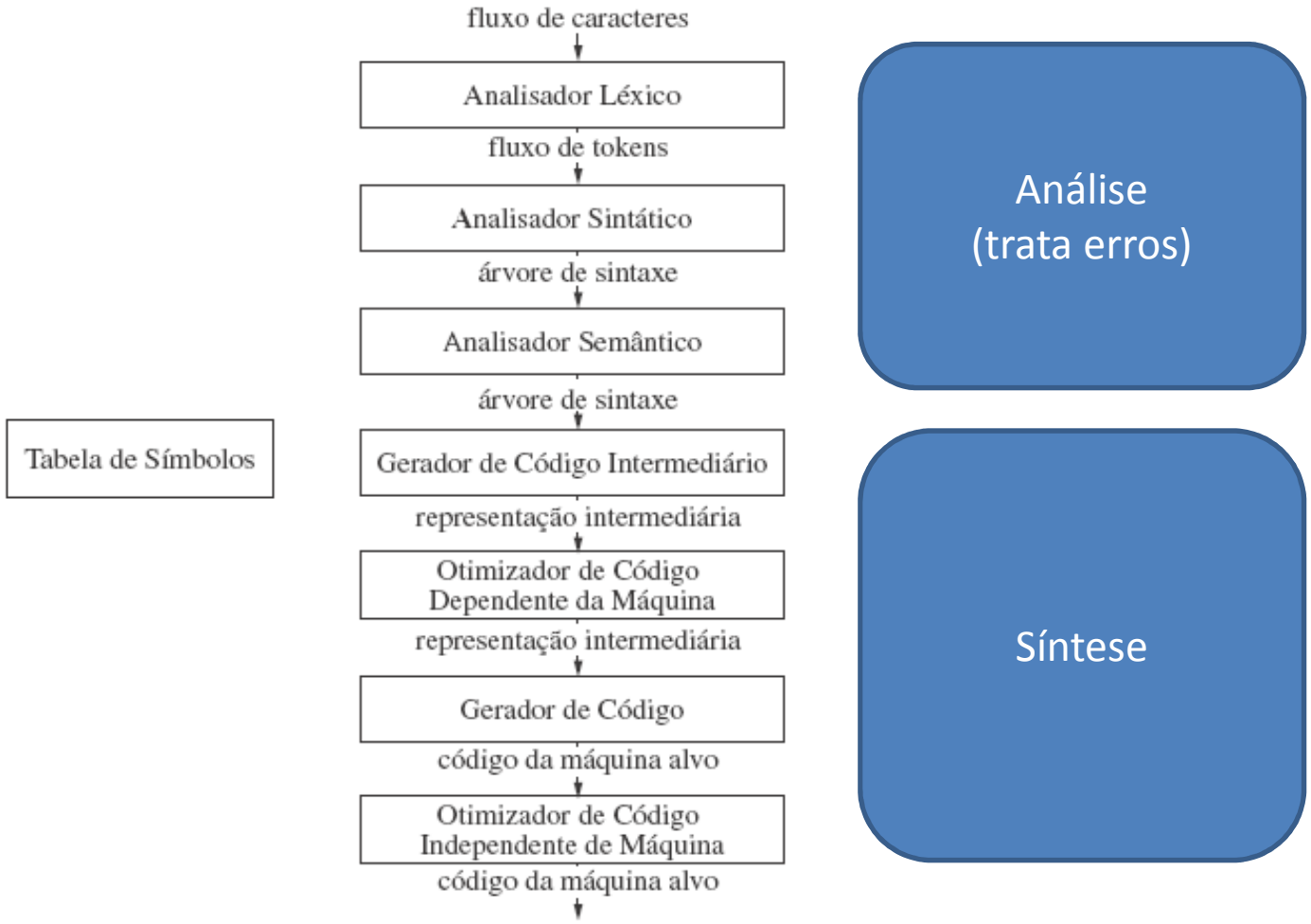


FIGURA 1.6 Fases de um compilador.

Análise x Síntese

- Divisão do processo
  - Análise: quebrar o programa fonte em diversas construções criando uma representação intermediária. Analisando a corretude do programa fonte.
  - Síntese: gerar o programa destino a partir de uma representação intermediária.

Processos

- A compilação é acompanhada de diversos processos auxiliares até que se gere um programa executável: pré-processador -> compilador -> montador -> linkeditor -> carregador
- **Pré-processador:** produzir a entrada para o compilador.
  - agrupar os diversos módulos que compõem o programa fonte. Ex: #include
  - processamento de macros: definição de macros ou uso (expansão da construção associada). Ex: #define
  - saída: programa fonte

Processos

- **Compilador:**
  - efetuar a tradução do programa fonte para o código assembly.
  - Posteriormente usa um “montador” para produzir um código de máquina relativo (realocável) que é passado direto ao linkeditor .
- B := A + 2 ---> Fonte
- Código Assembly
- MOV A, R1 (load)
- ADD #2, R1 (add)
- MOV R1, B (store)

Processos

- **Montador:** efetuar a tradução do programa assembly em um código de máquina relativo. Trabalhando em passos:
- Passo 1: determina-se todos os identificadores que denotam posição de memória:
- Tabela de Símbolos
- Identificador Endereço
- A 0
- B 4
- Passo 2: tradução da entrada para a sequência de bits equivalente
- Operador Regist. Modo End. Endereço
- 0001 01 00 00000000 *
- 0011 01 10 00000010
- 0010 01 00 00000100 *

Processos

- **Linkeditor:** construir um único programa a partir de diversos arquivos de códigos relativos:
- Juntar resultados de diferentes compilações inclusive de bibliotecas fornecidas pelo sistema, resolvendo ainda qualquer referência externa.

Processos

- **Loader (Carregador)**: altera os endereços relativos para absolutos, carregando o programa e os dados na memória.
- O código do exemplo anterior sendo carregado na posição 8 de memória. Produziria o seguinte código:
- 0001 01 00 00001000
- 0011 01 10 00000010 <----- valor absoluto de 2
- 0010 01 00 00001100

Processos

- É importante relatar que os processos acima nem sempre são facilmente identificados como módulos de um compilador, muitas vezes uma ou mais tarefas são sobrecarregadas dentro de um mesmo procedimento.
- O mesmo pode acontecer com as diferentes etapas de análise descritas a seguir.