

Linguagens e Compiladores

Aula 1:

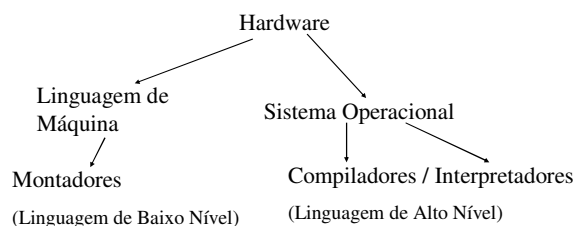
- a) Noções de Compiladores e Interpretadores
- b) Usos da tecnologia de compilação
- c) Compiladores, filtros e pré-processadores
- d) Estruturação lógica de compiladores
- e) Exemplo

Ricardo Luis de Azevedo da Rocha

a) Noções de Compiladores

- Elementos de um sistema de programação
- Conceitos, definições
- Utilização de compiladores
- Exemplo
- Compiladores, filtros e pré-processadores
- Formalização das linguagens de programação
- Estratégia para obtenção de compiladores
- Estruturação lógica e física dos compiladores

Sistema de Programação



Definição

- **Compilador:** Um programa de computador que traduz código escrito em uma determinada linguagem (código-fonte) em código escrito em outra linguagem (linguagem-objeto).

Exemplos:

- Traduzir de C++ para C
- Traduzir de Java para JVM
- Traduzir de C para C (Por que? Para tornar o código mais eficiente, menor, mais rápido, ou mesmo medir desempenho).

b) Usos da Tecnologia de Compilação

- Uso mais comum: traduzir um programa em linguagem de alto nível para código objeto para um processador particular
- Melhorar o desempenho de programas (otimizar)
- Interpretadores: tradutores diretos “on-line”, exemplos Perl, Java
- Paralelização ou vetorização automática
- Ferramentas de depuração
- Segurança: Java VM usa compilação para demonstrar segurança de código
- Formatadores de texto (TEX para PS, PS para PDF)
- Simulação de arquiteturas, Tolerância a falhas, Assinaturas

Chave do Processo

- Habilidade de extrair as propriedades de um programa (análise), e, opcionalmente, transformá-lo (síntese). Exemplo:

```
X = ...
i = 1
While (i <= 100) do
  i = i + 1
  j = i * 4
  N = j ** 2 /* Exponenciação
  Y = X * 2.0
  A[i] = X * 4.0
  B[j] = Y * N
  C[j] = N * Y * C[j]
End While
L1: Print B[1:400]
Print C[1:400]
Stop
```

Resultado da Otimização de Código do Exemplo

```
j = 4          /* (i = 1) * 4: propagação da const
Y = X * 2.0    /* mudança do invariante do laço
While (j <= 400) do
  j = j + 4    /* substituição da variável (forte)
  N = j * j    /* redução (forte)
               /* eliminação de A
  B[j] = Y * N
  C[j] = B[j] * C[j] /* eliminação de sub-expressão
End While
L1: Print B[1:400]
Print C[1:400]
Stop

/* Também aloca registros j, N, Y
```

c) Compiladores, Filtros e Pré-Processadores

- Tradutor genérico: Fonte → Objeto
- Compilador: Alto nível → Outra linguagem
- Montador: Baixo nível → Ling. máquina
- Filtro: Alto nível → Alto nível (conversões)
- Pré-Processador: Semelhante ao Filtro, porém antecede a etapa de compilação, por exemplo, expandindo macros, etc.

Formalização das Linguagens de Programação

- Objetivo: eliminar ambigüidades
- Meio: notação formal
- Característica: linguagens de programação são mais restritivas que as linguagens naturais
- Inviável tratar a comunicação homem-máquina em linguagem puramente natural
- Uso de gramáticas, reconhecedores e dispositivos geradores, descritos em uma *metalinguagem*

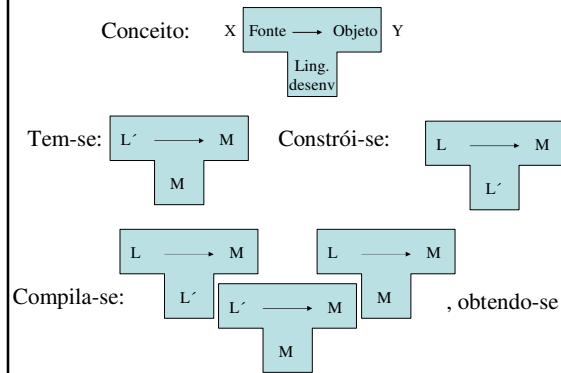
Estratégia para Obtenção de Compiladores

- Três alternativas:
 - Manual: Realizar toda a codificação através de técnicas especiais (os primeiros compiladores foram construídos desta forma)
 - Mista: Utilizar em parte algum programa de auxílio e desenvolver o restante
 - Semi-Automática: Utilizar programas mais modernos e completos, modificando pouco a estrutura gerada
- Definir a linguagem fonte de tradução e a linguagem objeto (máquina objeto), além da linguagem de desenvolvimento

Tipos de Compiladores

- Com relação à máquina hospedeira:
 - Auto-residente: se a máquina hospedeira é a mesma máquina para a qual deve ser gerado código
 - Compilador-Cruzado: se a máquina hospedeira é diferente da máquina para a qual o código deve ser gerado
- Método de “*bootstrapping*” – auto-compiláveis

“Bootstrapping”



- Analisador léxico
- Analisador sintático
- Analisador semântico
- Gerador de código intermediário
- Otimizador de código
- Gerador de código objeto

```

graph LR
    CF[Código fonte] --> AL[Analísador Léxico]
    AL --> A[Átomos]
    A --> AS[Analísador Sintático]
    AS --> ASint[Árvore sintática]
    ASint --> ASem[Analísador Semântico]
    ASem --> GCI[Gerador de Código Int.]
    GCI --> RI[Repres. Interna]
    RI --> OC[Otimizador de Código]
    OC --> RI2[Repres. Interna]
    RI2 --> GC[Gerador de Código]
    GC --> CO[Código Objeto]
    ASem --> CO
  
```

O diagrama ilustra o processo de compilação em várias etapas:

- Código fonte** é processado pelo **Analísador Léxico** para identificar palavras, operadores, números, etc., resultando em **Átomos**.
- Os **Átomos** são processados pelo **Analísador Sintático** para identificar a estrutura de expressões, declarações, etc., resultando na **Árvore sintática**.
- A **Árvore sintática** é processada pelo **Analísador Semântico** para identificar o sentido, tipos, etc.
- O **Analísador Semântico** também gera o **Código Objeto** diretamente.
- A **Árvore sintática** também é usada pelo **Gerador de Código Int.** para gerar a **Repres. Interna**.
- A **Repres. Interna** é processada pelo **Otimizador de Código** para transformar a representação interna e modificar o código (desempenho), resultando novamente em **Repres. Interna**.
- A **Repres. Interna** é processada pelo **Gerador de Código** para gerar o **Código Objeto**.

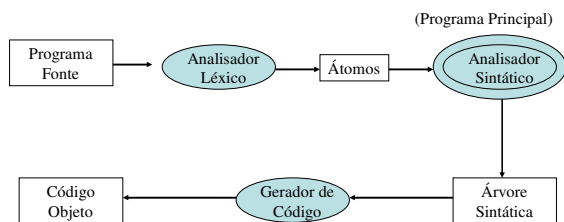
```
graph TD; A[Programa Fonte  
(Linguagem de alto nível)] --> B[Interpretador]; A --> C[Compilador]; B --> D[Resultados da execução]; D --> E[Hardware]; C --> F[Programa Objeto  
(Linguagem intermediária)]; C --> G[Programa Objeto  
(Linguagem relocável)]; F --> H[Interpretador  
Ling interm.]; F --> I[Programa Objeto  
(Linguagem de Máquina)]; G --> J[Relocador]; I --> J; K[Bibliotecas] --> J; J --> E;
```

O diagrama ilustra o fluxo de um programa de alto nível até a execução no hardware. O processo começa com o **Programa Fonte (Linguagem de alto nível)**, que pode ser processado diretamente por um **Interpretador** para gerar **Resultados da execução** no **Hardware**, ou então ser compilado por um **Compilador**. O compilador gera um **Programa Objeto (Linguagem intermediária)**, que pode ser interpretado por um **Interpretador Ling interm.** para gerar resultados de execução, ou então ser convertido para um **Programa Objeto (Linguagem de Máquina)**. Este último pode ser executado diretamente no hardware ou passar por um **Relocador** (que também incorpora **Bibliotecas**) para gerar o executável final no hardware.

Organização Física

- Quanto ao número de passos:
 - Em um único passo (ex. Delphi, Turbo Pascal)
 - Em múltiplos passos (ex. compiladores COBOL para Mainframes, compiladores da Microsoft mais antigos)
- Quanto à hierarquia interna
 - Dirigidos por sintaxe (mais comum)
 - Dirigidos pela análise léxica
 - Dirigidos pela geração de código

Dirigidos por Sintaxe



Considerações Custo-benefício

- Tarefa de compilação de uma linguagem de alto nível não é trivial
- Sem compiladores não é possível utilizar uma linguagem de alto nível (justificativa de construção)
- Número de linhas de programa geradas é constante para cada programador, e os programas ficam menores
- Número de erros diminui pela verificação dos compiladores
- Engenharia de Software, considerações de eficiência
- Dificuldade: Um compilador Pascal com otimização demanda aproximadamente 30 Homens-Ano

e) Exemplo

Sample Program

```
public class first {  
    public static void main(String argv[])  
  
        int x;  
        x = 19;  
        x = x*x;  
    }  
}
```

Compilação

Output Bytecode

```
Compiled from first.java  
public class first extends java.lang.Object {  
    public static void main(java.lang.String[]);  
    public first();  
    Method void main(java.lang.String[])  
0      bipush 19  
2      istore_1  
3      iload_1  
4      iload_1  
5      imul  
6      istore_1  
7      return  
}
```

Continuação

Byte Code Continued

```
Method first()  
0      aload_0  
1      invokevirtual #3 <Method java.lang.Object.<init>()V>  
4      return  
}
```

Bibliografia

- "Introdução à Compilação", J. J. Neto, LTC, 1987 (**Livro-Base**)
- "Compilers: Principles, Techniques and Tools", Alfred Aho, Ravi Sethi and Jeffrey Ullman, Addison-Wesley, 2007
- "Introdução à compilação", I. Ricarte, ed Campus, 2008
- "Projeto Moderno de Compiladores – Implementação e Aplicações", D. Grune, H. Bal, C. Jacobs, K. G. Langendoen, ed. Campus, 2002
- "Compiladores: princípios e práticas", K. C. Loudon, ed. Thompson, 2004
- "Implementação de Linguagens de Programação: Compiladores", A. M. A. Price & S. S. Toscani, ed. Sagra-Luzzatto, 2002
- Trembley & Sorenson – The theory and practice of compiler writing, McGraw-Hill, 1985
- Robert W. Sebesta – Programming language concepts, Addison-Wesley, 2007.
