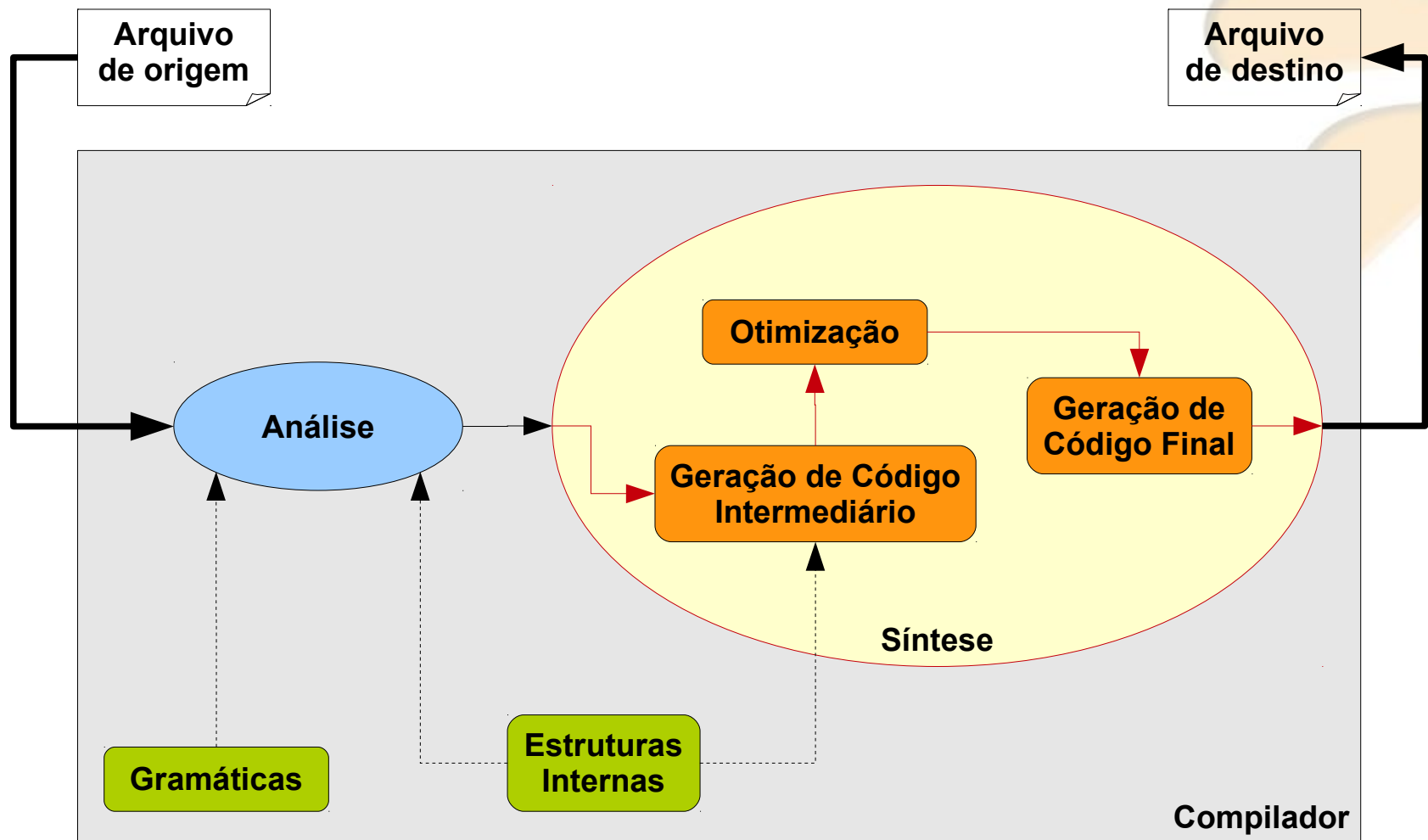


Compiladores

Geração de Código Objeto

Cristiano Lehrer, M.Sc.

Atividades do Compilador



Introdução

- Os principais requisitos impostos a geradores de código objeto são os seguintes:
 - O código gerado deve ser correto e de alta qualidade.
 - O código deve fazer uso efetivo dos recursos da máquina.
 - O código gerado deve executar eficientemente.
- O problema de gerar código ótimo é insolúvel (indecidível) como tantos outros.
- Na prática, devemos contentar-nos com técnicas heurísticas que geram bom código (não necessariamente ótimo).

Considerações

- Basicamente, quatro aspectos devem ser considerados no projeto de geradores de código:
 - Forma do código objeto a ser gerado:
 - Linguagem absoluta, relocável ou *assembly*.
 - Seleção das instruções de máquina:
 - A escolha da sequência apropriada pode resultar num código mais curto e mais rápido.
 - Alocação de registradores.
 - Escolha da ordem de avaliação:
 - A determinação da melhor ordem para execução das instruções é um problema insolúvel.
 - Algumas computações requerem menos registradores para resultados intermediários.

Forma do Código Objeto (1/3)

- Linguagem absoluta:
 - A geração de um programa em linguagem absoluta de máquina tem a vantagem de que o programa objeto pode ser armazenado numa área de memória fixa e ser imediatamente executado.
 - Compiladores deste tipo são utilizados em ambientes universitários, onde é altamente conveniente diminuir o custo de compilação.
 - Os compiladores que geram código absoluto e executam-no imediatamente são conhecidos como *load and go compilers*.

Forma do Código Objeto (2/3)

- Linguagem relocável:
 - A geração de código em linguagem de máquina relocável permite a compilação separada de subprogramas.
 - Módulos e objetos relocáveis podem ser ligados e carregados por um Ligador-Carregador.
 - Essa estratégia dá flexibilidade para compilar subrotinas separadamente e para chamar outros programas previamente compilados.

Forma do Código Objeto (3/3)

- Linguagem *assembly*:
 - A tradução para linguagem *assembly* facilita o processo de geração de código.
 - São geradas instruções simbólicas e podem ser usadas as facilidades de macro instruções.
 - O preço pago é um passo adicional:
 - Tradução para linguagem de máquina.
 - É uma estratégia razoável, especialmente, para máquinas com pouca memória, nas quais o compilador deve desenvolver-se em vários passos.

Alocação de Registradores (1/2)

- Instruções com registradores são mais curtas e mais rápidas do que instruções envolvendo memória.
- Portanto, o uso eficiente de registradores é muito importante.
- A atribuição ótima de registradores a variáveis é muito difícil, e muito problemática quando a máquina trabalha com registradores aos pares (para instruções de divisão e multiplicação), ou provê registradores específicos para endereçamentos e para dados.

Alocação de Registradores (2/2)

- Operação de divisão nos computadores IBM/360 e IBM/370:
 - $D\ r, m$
 - O dividendo de 64 bits deve estar armazenado em um par de registradores par-ímpar, representado por r .
 - m contém o divisor.
 - Após a divisão, o registrador par contém o resto e o ímpar contém o quociente.

$t = a + b$	L	R0, a	
	A	R0, b	
$t = t + c$	A	R0, c	
$t = t / d$	SRDA	R0, 32	desloca o dividendo para R1 e limpa R0
	D	R0, d	
	St	R1, t	

Máquina Objeto (1/2)

- Familiaridade com a máquina e com o conjunto de instruções é pré-requisito para projetar um bom gerador de código.
- Formato das instruções: `op fonte, destino`
- Instruções:
 - `ADD R1, R2 R2 = R2 + R1 (adição)`
 - `SUB R1, R2 R2 = R2 – R1 (subtração)`
 - `MUL R1, R2 R2 = R2 * R1 (multiplicação)`
 - `DIV R1, R2 R2 = R2 / R1 (divisão)`
 - `LOAD M, R Registrador = Memória (carregamento)`
 - `STORE R, M Memória = Registrador
 (armazenamento)`
 - `COPY R1, R2 R2 = R1 (cópia)`

Máquina Objeto (2/2)

- **$a := b + c$**

- LOAD $b, R0$ $R0 = b$
- LOAD $c, R1$ $R1 = c$
- ADD $R1, R0$ $R0 = R0 + R1$
- STORE $R0, a$ $a = R0$

- **$a := b + c * d$**

- LOAD $d, R0$ $R0 = d$
- LOAD $c, R1$ $R1 = c$
- MUL $R0, R1$ $R1 = R1 * R0$
- LOAD $b, R0$ $R0 = b$
- ADD $R1, R0$ $R0 = R0 + R1$
- STORE $R0, a$ $a = R0$