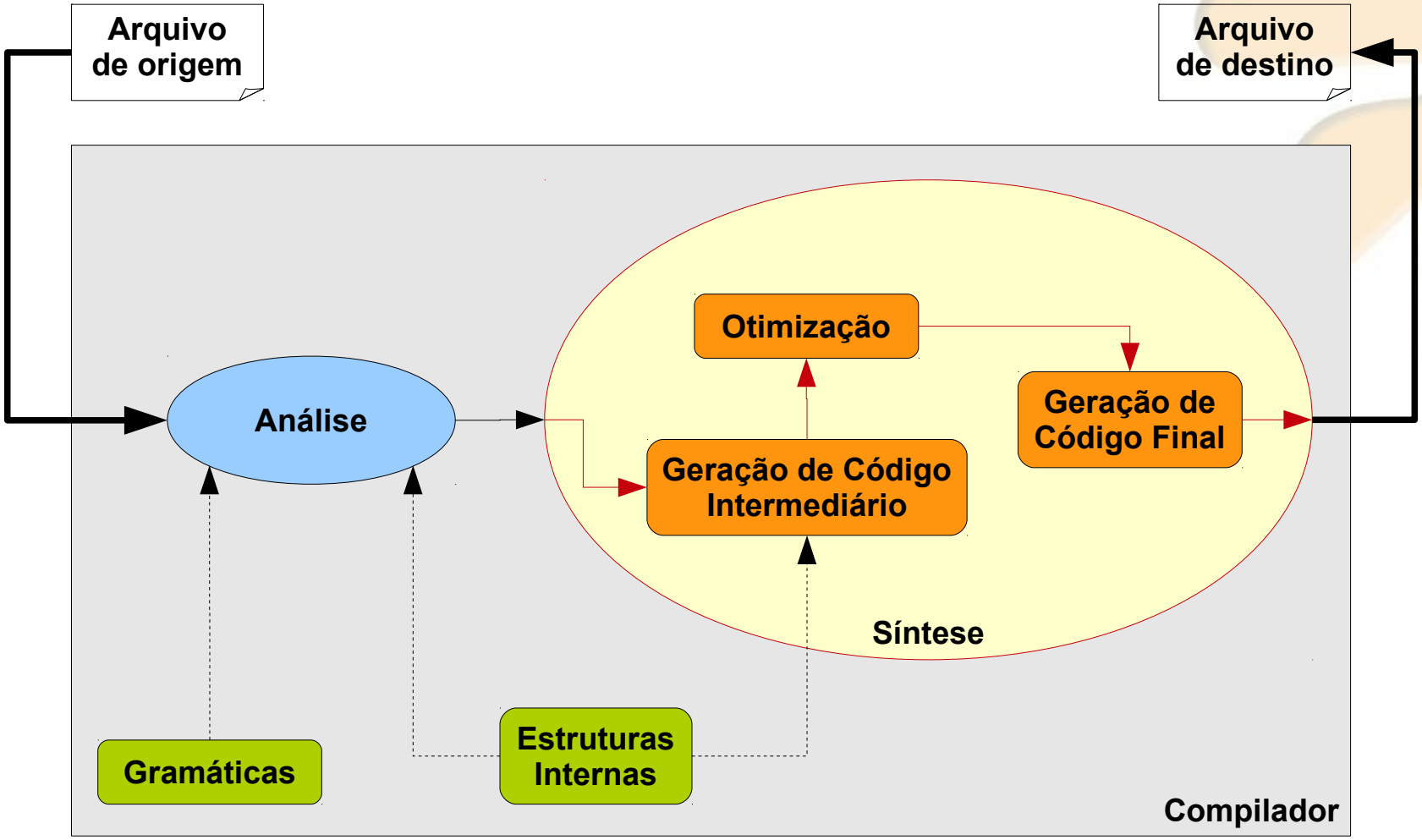
Compiladores

Otimização de Código

Cristiano Lehrer, M.Sc.

ybadoo

Atividades do Compilador



Introdução (1/2)

- O problema da geração de código eficiente envolve aspectos de uso de memória e de velocidade de execução:
  - Esses aspectos são muitas vezes conflitantes, pois normalmente ganhos no espaço utilizado implicam perdas no tempo de execução e vice-versa.
- Geração de código ótimo é um problema indecidível:
  - O que se faz é utilizar heurísticas para tentar otimizar o código ao máximo.
  - Isso faz com que a solução encontrada não seja a melhor possível.
- Como a otimização de código consome tempo de compilação, ela somente deve ser implementada se a utilização do compilador realmente exige um código objeto eficiente.

Introdução (2/2)

- Processo de otimização desenvolve-se em duas fases:
  - Otimização do código intermediário:
    - Técnicas para eliminar atribuições redundantes, suprimir sub-expressões comuns, eliminar temporários desnecessários, trocar instruções de lugar, de modo a obter um código intermediário melhor.
  - Otimização do código objeto:
    - Realizada através da troca de instruções de máquina por instruções mais rápidas e da melhor utilização de registradores.

Otimização de Código Intermediário

- A fase de otimização do código intermediário vem logo após a fase de geração desse código e tem o objetivo de tornar o código intermediário mais apropriado para a produção de código objeto eficiente, tanto em relação ao tamanho como ao tempo de execução.
- A técnica de otimização de código intermediário consiste em identificar segmentos sequenciais do programa, chamados blocos básicos, representá-los através de grafos dirigidos e submetê-los a um processo de otimização.
- Um **bloco básico** é um trecho de programa que inicia por um comando líder e que não apresenta comandos de desvio, a não ser eventualmente o último, o qual pode ser um comando de desvio.

ybadoo

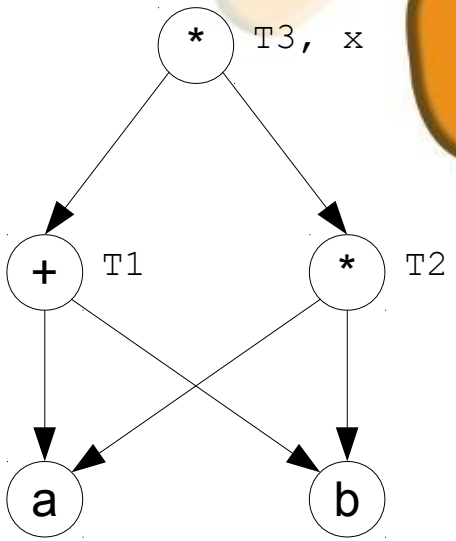
Representação de Blocos Básicos Através de Grafos (1/2)

- Todo bloco básico pode ser representado através de um grafo acíclico dirigido (GAD), também denominado **grafo de fluxo**, que permite:
  - Identificação de subexpressões comuns num bloco.
  - Identificação de variáveis usadas dentro do bloco mas que foram computadas fora (variáveis de entrada).
  - Identificação de variáveis cujos valores são computados dentro do bloco e usadas fora do bloco (variáveis de saída).
- Um GAD para um bloco básico é um grafo acíclico orientado tal que:
  - As folhas representam variáveis ou constantes.
  - Os nodos internos representam operadores e valores computados.

Representação de Blocos Básicos Através de Grafos (2/2)

```
x := (a + b) * (a - b)
```

	oper	arg1	arg2	result
(0)	+	a	b	T1
(1)	-	a	b	T2
(2)	*	T1	T2	T3
(3)	:=	T3		x



ybadoo

Algoritmo para Construir o GAD de um Bloco (1/4)

- O algoritmo supõe que cada instrução do código intermediário segue um dos seguintes três formatos:
  - (i) `x = y op z`
  - (ii) `x = op y`
  - (iii) `x = y`
- Instruções do tipo `if-goto` são tratadas como no caso (i).

Algoritmo para Construir o GAD de um Bloco (2/4)

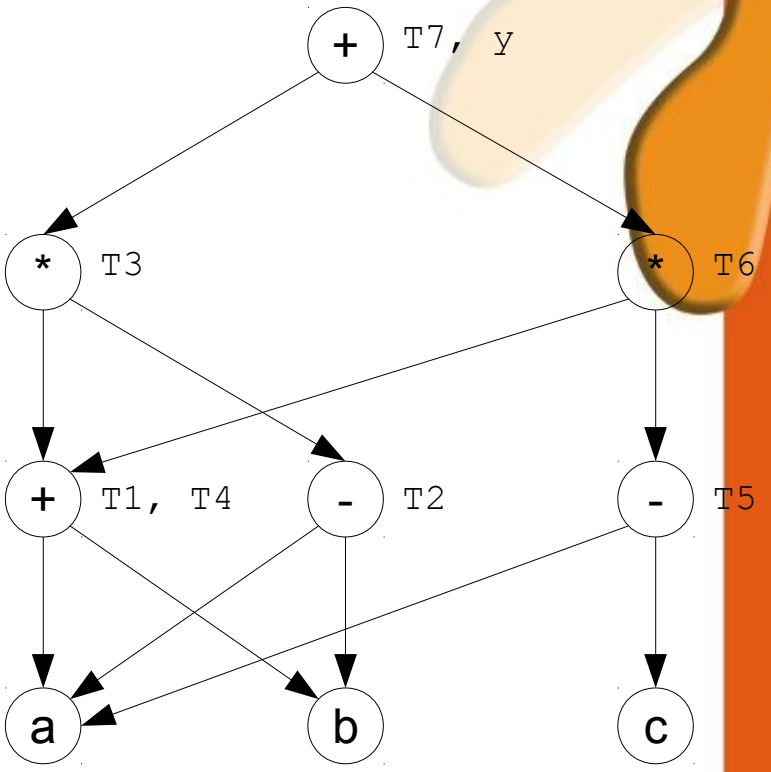
- Para cada instrução do bloco básico, execute os passos (1) e (2):
  - (1) Se o nodo **y** ainda não existe no grafo, crie uma folha para **y**.
    - Tratando-se do caso (i), faça o mesmo para **z**.
  - (2) No caso (i), verifique se existe um nodo **op** com filhos **y** e **z** (nessa ordem).
    - Caso exista, chame-o, também de **x**.
    - Senão crie um nodo **op** com nome **x** e dois arcos dirigidos do nodo **op** para **y** e para **z**.
  - (2) No caso (ii), verifique se existe um nodo **op** com um único filho **y**.
    - Se não existir, crie tal nodo e um arco orientado desse nodo para **y**.
    - Chame de **x** o nodo criado ou encontrado.
  - (2) No caso (iii), chame também de **x** o nodo **y**.

Algoritmo para Construir o GAD de um Bloco (3/4)

```
y := ((a + b) * (a - b)) + ((a + b) * (a - c))
```

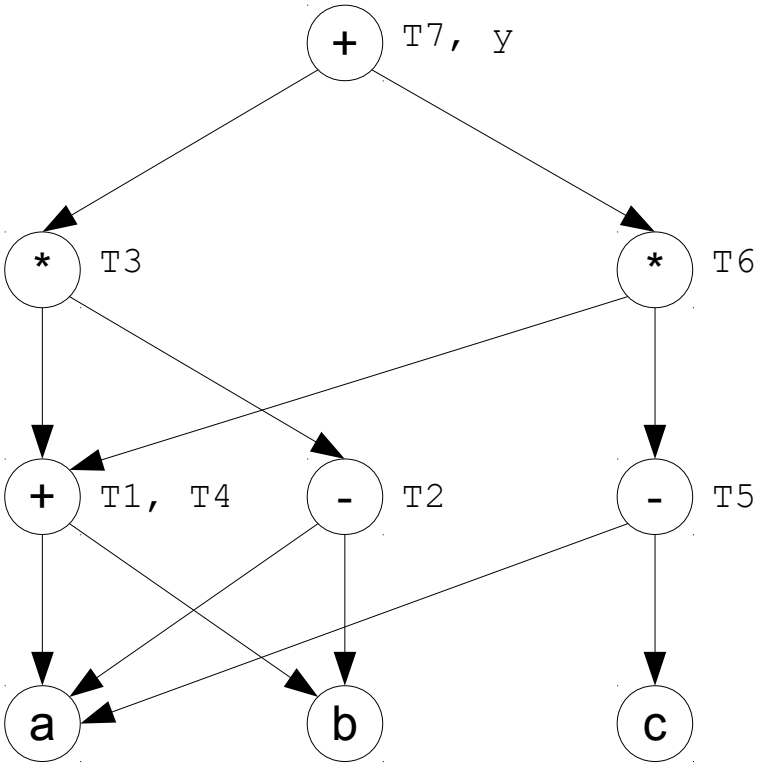
	oper	arg1	arg2	result
(0)	+	a	b	T1
(1)	-	a	b	T2
(2)	*	T1	T2	T3
(3)	+	a	b	T4
(4)	-	a	c	T5
(5)	*	T4	T5	T6
(6)	+	T3	T6	T7
(7)	:=	T7		Y

Código de três endereços original



Algoritmo para Construir o GAD de um Bloco (4/4)

```
y := ((a + b) * (a - b)) + ((a + b) * (a - c))
```



	oper	arg1	arg2	result
(0)	+	a	b	T1
(1)	-	a	b	T2
(2)	*	T1	T2	T3
(4)	-	a	c	T5
(5)	*	T1	T5	T6
(6)	+	T3	T6	T7
(7)	:=	T7		Y

Código de três endereços simplificado

ybadoo