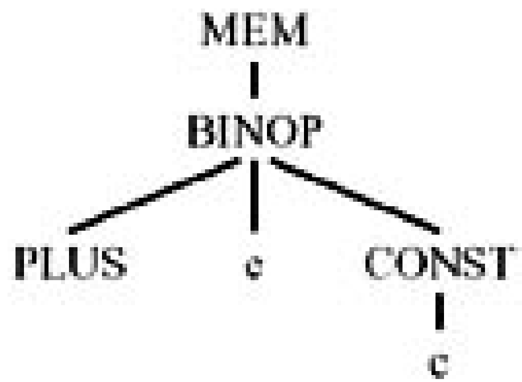
Seleção de Instruções

Sandro Rigo
sandro@ic.unicamp.br

Introdução

- A árvore da IR expressa uma operação "simples" em cada nó
  - Acesso à memória
  - Operador Binário
  - Salto condicional
- Instruções da máquina podem realizar uma ou mais dessas operações

Introdução

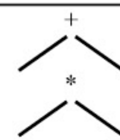
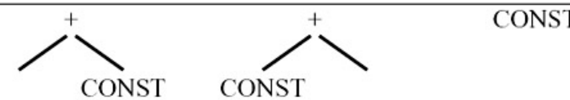
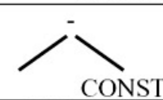
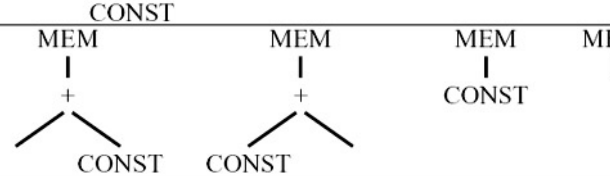
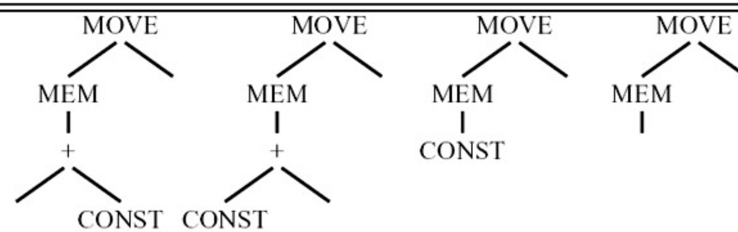
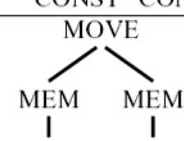


- Que instrução seria essa?
- Encontrar o conjunto de instruções de máquinas que implementa uma dada árvore da IR é o objetivo da ***Seleção de Instruções***

Padrões de Árvores

- Expressão as instruções da máquina
- Seleção de instruções:
    - Cubra a árvore da IR com o *menor* número de padrões existentes para a máquina alvo
- Exemplo:
    - Máquina Jouette
        - r0 contém sempre zero

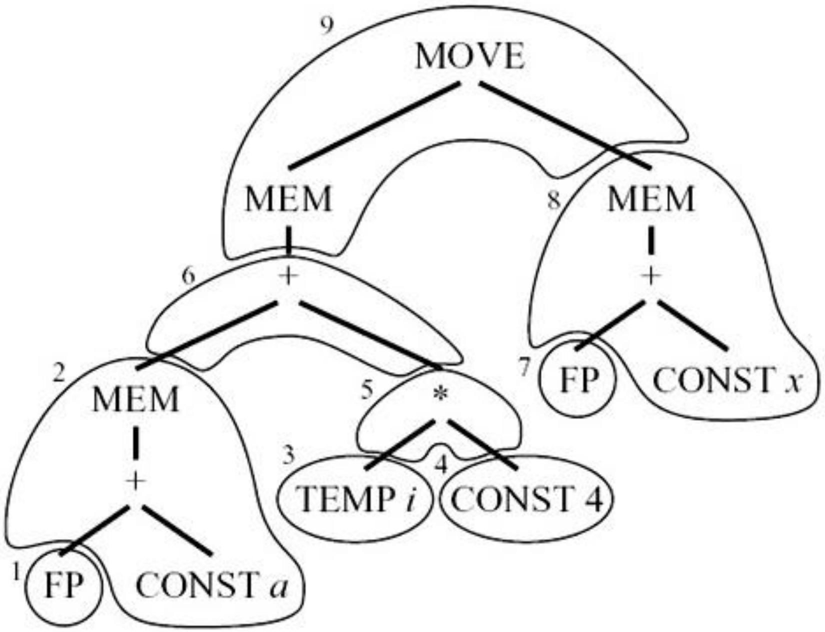
Padrões - Jouette

Name	*Effect*	*Trees*
—	r_i	TEMP
ADD MUL	$r_i \leftarrow r_j + r_k$ $r_i \leftarrow r_j \times r_k$	
SUB DIV	$r_i \leftarrow r_j - r_k$ $r_i \leftarrow r_j / r_k$	
ADDI	$r_i \leftarrow r_j + c$	
SUBI	$r_i \leftarrow r_j - c$	
LOAD	$r_i \leftarrow M[r_j + c]$	
STORE	$M[r_j + c] \leftarrow r_i$	
MOVEM	$M[r_j] \leftarrow M[r_i]$	

Padrões - Jouette

- Primeira linha não gera instrução
  - TEMP é implementado como registrador
- Duas últimas instruções não geram resultado em registrador
  - Alterações na memória
- Uma instrução pode ter mais de um padrão associado
- Objetivo é cobrir a árvore toda, sem sobreposição entre padrões

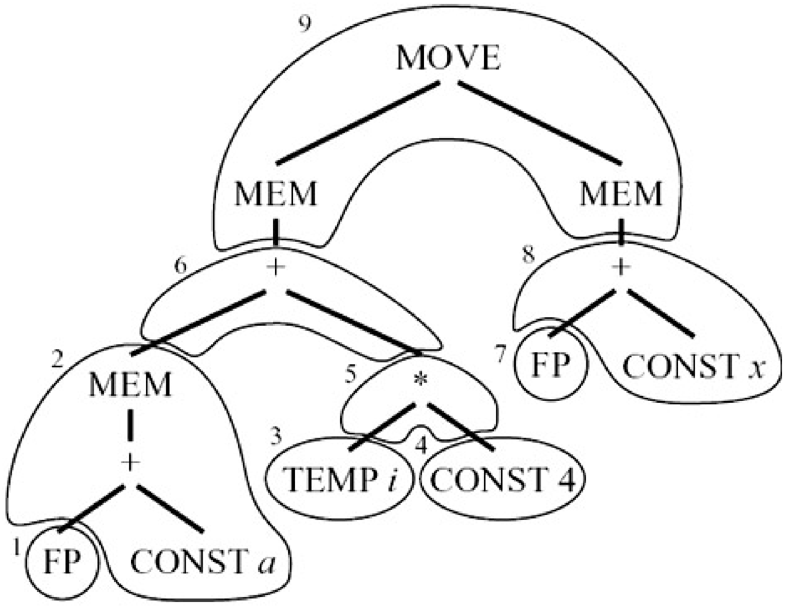
Seleção de Instruções



2	LOAD	$r_1 \leftarrow M[\mathbf{fp} + a]$
4	ADDI	$r_2 \leftarrow r_0 + 4$
5	MUL	$r_2 \leftarrow r_i \times r_2$
6	ADD	$r_1 \leftarrow r_1 + r_2$
8	LOAD	$r_2 \leftarrow M[\mathbf{fp} + x]$
9	STORE	$M[r_1 + 0] \leftarrow r_2$

- Que operação seria essa?

Seleção de Instruções



2	LOAD	$r_1 \leftarrow M[\mathbf{fp} + a]$
4	ADDI	$r_2 \leftarrow r_0 + 4$
5	MUL	$r_2 \leftarrow r_i \times r_2$
6	ADD	$r_1 \leftarrow r_1 + r_2$
8	ADDI	$r_2 \leftarrow \mathbf{fp} + x$
9	MOVEM	$M[r_1] \leftarrow M[r_2]$

- A cobertura não é única!

Seleção de Instruções

- ADDI $r1 \leftarrow r0 + a$
- ADD $r1 \leftarrow \mathbf{fp} + r1$
- LOAD $r1 \leftarrow M[r1 + 0]$
- ADDI $r2 \leftarrow r0 + 4$
- MUL $r2 \leftarrow ri \times r2$
- ADD $r1 \leftarrow r1 = r2$
- ADDI $r2 \leftarrow r0 + x$
- ADD $r2 \leftarrow \mathbf{fp} + r2$
- LOAD $r2 \leftarrow M[r2 + 0]$
- STORE $M[r1 + 0] \leftarrow r2$

- Tente cobrir a árvore com padrões de um nó apenas

Optimal e Optimum

- Queremos a cobertura que nos traga o menor custo
  - Normalmente a menor
  - Caso as instruções tenham latências diferentes
    - A de menor tempo total
- Cada instrução recebe um custo
  - A melhor cobertura da árvore é a que a soma dos custos dos padrões utilizados é a menor possível
  - Este é o *optimum*

Optimal e Optimum

- Uma cobertura onde nenhum par de padrões adjacentes possa ser combinado em um par de menor custo é *optimal*
- Caso haja um padrão que possa ser quebrado e diminua o custo total, ele deve ser descartado
- Optimum => optimal
- Optimal <≠ optimum

Optimal e Optimum

- No exemplo anterior assuma:
  - MOVEM tem custo m
  - Todas as outras têm custo 1
  - O que acontece com as duas coberturas apresentadas se
    - m = 0, 1 ou 2?

Algoritmos

- Achar coberturas “optimais” é mais fácil
- CISC
  - Dada a complexidade das instruções, os padrões costumam ser grandes
  - A diferença entre optimal e optimum se torna mais considerável
- RISC
  - Instruções simples levam a padrões pequenos
  - Custo costuma ser mais uniforme
  - A diferença entre optimal e optimum praticamente desaparece

Maximal Munch

- Encontra cobertura optimal
- Bastante simples
  - Inicie na raiz
  - Encontre o maior padrão que possa ser encaixado nesse nó
    - Cubra o raiz e provavelmente outros nós
  - Repita o processo para cada sub-árvore a ser coberta
- A cada padrão selecionado, uma instrução é gerada
- Ordem inversa da execução! A raiz é a última a ser executada

Maximal Munch

- O maior padrão é aquele com maior número de nós
- Se dois padrões do mesmo tamanho encaixam, a escolha é arbitrária
- Facilmente implementado através de funções recursivas
  - Ordene as cláusulas com a prioridade de tamanho dos padrões
  - Se para cada tipo de nó da árvore existir um padrão de cobertura de um nó, nunca pode ficar travado.

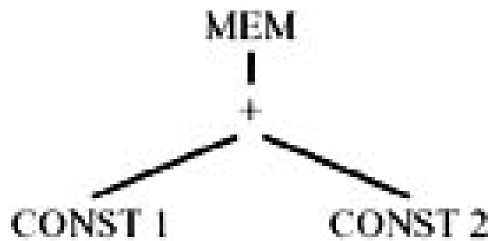
Maximal Munch

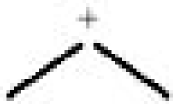
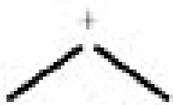
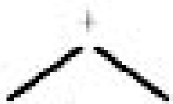
- Faça a cobertura Maximal Munch das seguintes árvores:
  - MOVE(MEM(+(+(CONST*1000*, MEM(TEMP*x*)), TEMP*fp*)), CONST*0*)
  - BINOP(MUL, CONST*5*, MEM(CONST*100*))

Programação Dinâmica

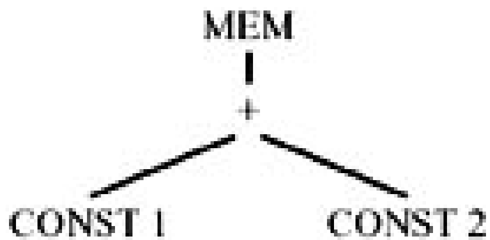
- Encontra um cobertura ótima (optimum)
- PD monta uma solução ótima baseada em soluções ótimas de sub-problemas
- O algoritmo atribui um custo a cada nó da árvore
  - A soma do custo de todas as instruções da melhor cobertura da sub-árvore com raiz no respectivo nó
  - Para um dado nó n
    - Encontra o melhor custo para suas sub-árvores
    - Analisa os padrões que podem cobrir n
    - Algoritmo Botton-up

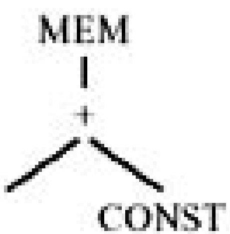
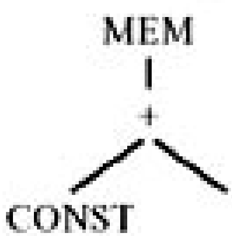
Programação Dinâmica



Tile	Instruction	Tile Cost	Leaves Cost	Total Cost
	ADD	1	1+1	3
 CONST	ADDI	1	1	2
 CONST	ADDI	1	1	2

Programação Dinâmica



Tile	Instruction	Tile Cost	Leaves Cost	Total Cost
	LOAD	1	2	3
	LOAD	1	1	2
	LOAD	1	1	2

Programação Dinâmica

- Após computar o custo da raiz, emitir as instruções
- Emissão de (n)
  - Para cada folha f ***do padrão*** selecionado para n, execute emissão(f)
  - Emita a instrução do padrão de n

Comentários sobre Eficiência

- Seja:
  - T: números de padrões diferentes
  - K: no. médio de nós não-folhas dos padrões casados
  - K’: maior # de nós a serem olhados para identificar quais padrões casam a uma dada sub-árvore. Aprox. o tamanho do maior padrão
  - T’: média de padrões diferentes que casam em cada nó
  - N: # nós da árvore
  - RISC típico:
    - T=50, K=2, K’=4, T’=5
- Maximal Munch: N/K * (K’+ T’)
- Programação Dinâmica: N * (K’+ T’)
  - Requer duas passadas na árvore

Comentários sobre Eficiência

- Ambos são lineares
- Seleção de instruções é bastante rápida comparada com outras fases da compilação
- Até analise léxica pode ser mais demorada