Análise Semântica e Representação Intermédia

Compiladores, Aula Nº 23

João M. P. Cardoso

Conversão para a IR de nível baixo

- Converte fluxo de controlo estruturado em fluxo de controlo baseado em saltos (não estruturado)
  - Saltos condicionais e incondicionais
- Converte modelo de memória estruturado em modelo de memória planar
  - Endereçamento planar para variáveis
  - Endereçamento planar para Arrays
- Continua independente da linguagem

Representação do Programa

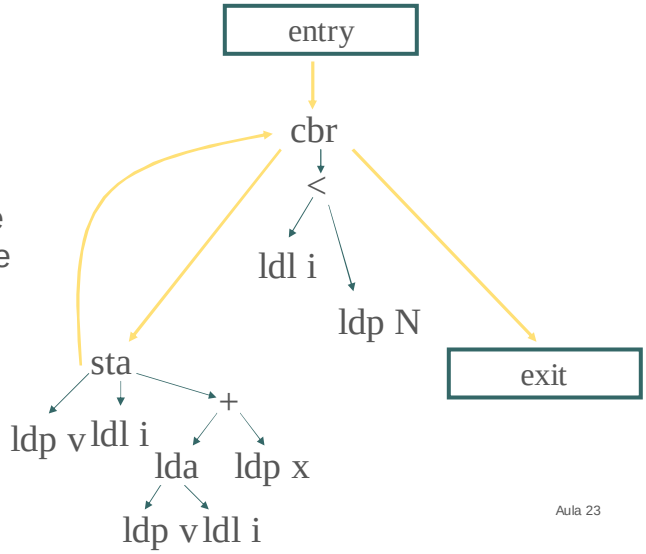
- *Control Flow Graph* (CFG): grafo de fluxo de controlo
  - Nós do CFG são nós de instruções
    - stl, sta, cbr, ldl, lda, ldp são nós de instruções
    - +, <, ... são nós de expressões
  - Laços no CFG representam o fluxo de controlo
  - *Forks* em instruções de salto condicional
    - Representam dois ou mais caminhos possíveis
  - *Merges* quando o controlo pode alcançar um ponto por caminhos múltiplos
  - Um nó de entrada (entry) e um nó de saída (exit)

Exemplo: CFG

```
while (i < N)
  v[i] = v[i]+x;
```

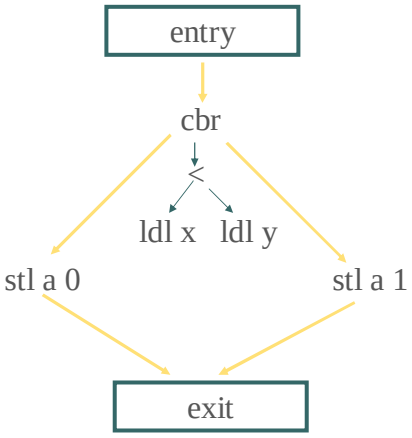
Laços entre Instruções e expressões

Laços de fluxo de controlo



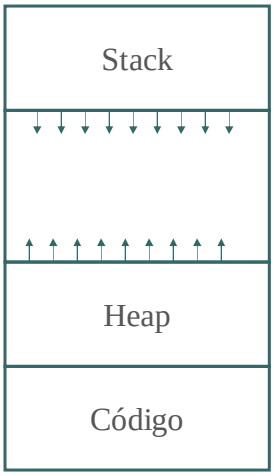
Exemplo: CFG

```
if (x < y) {
    a = 0;
} else {
    a = 1;
}
```



Modelo de Memória da Máquina Alvo

- Uma memória planar
  - Composta por palavras
  - Endereçável ao byte
- Nós modelam instruções Load e Store
  - ld addr,offset – resultado é o conteúdo de memória no local: addr+offset
  - st addr, offset, valor – escreve valor no local: addr+offset
  - Substituir nós: lda e ldl por nós ld
  - Substituir nós: sta e stl por nós st

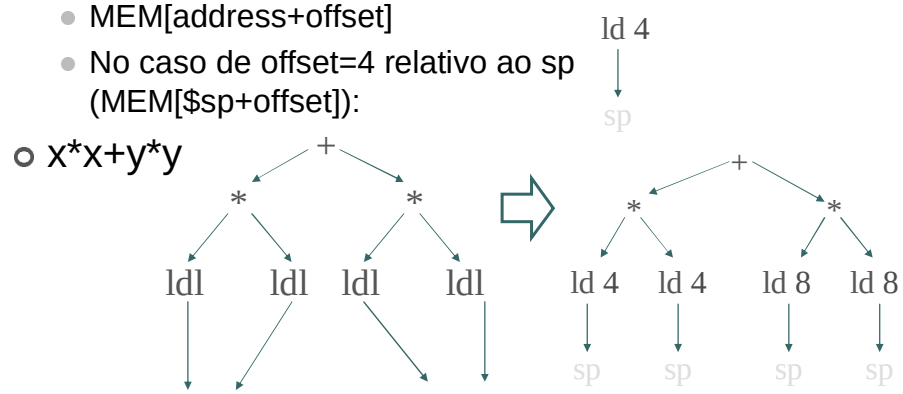


locais (alguns parâmetros)

Arrays

Exemplo:

- ld address offset
  - MEM[address+offset]
  - No caso de offset=4 relativo ao sp (MEM[$sp+offset]):
- x*x+y*y



Descritor local para x (4)

Descritor local para y (8)

Parâmetros

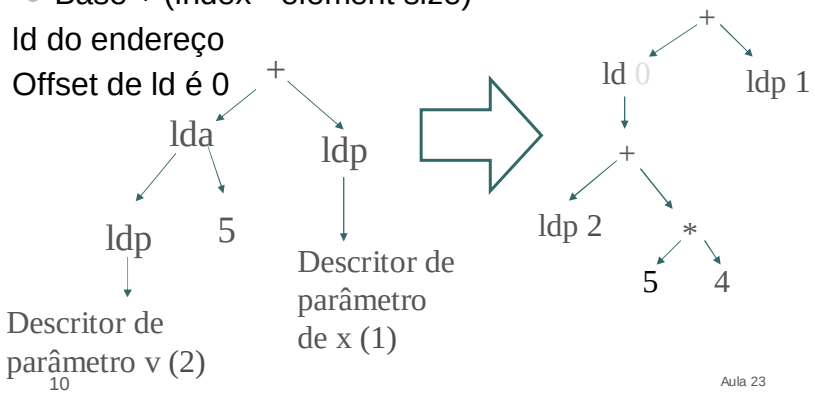
- Muitas máquinas têm convenções nas chamadas
  - Primeiro parâmetro no registo 5, segundo parâmetro no registo 6, ...
  - ver $a0, $a1, … do MIPS
- As convenções variam com a máquina
- Vamos assumir que cada parâmetro é uma palavra
- Vamos endereçar os parâmetros pelo número
  - ldp <número do parâmetro>

Acesso a elementos de um Array

- Assumir que a variável aponta para o primeiro elemento do array
- Elementos do array em posições contíguas
- Qual é o endereço: v[5]?
  - v é um array de inteiros: assumir inteiros de 4 bytes
  - (endereço em v) + (5*4)
- Determinar endereço
  - Base do Array + (index * element size)

Exemplo: v[5]+x

- Conversão de nós lda para nós ld
- Determinar endereço
  - Base + (index * element size)
- ld do endereço
- Offset de ld é 0



Variáveis Locais

- Assumir que são alocadas na pilha de chamadas
  - Endereçamento realizado usando *offsets* a partir do apontador da pilha
- Relembrar:
  - pilha cresce para baixo e por isso os *offsets* são positivos
  - Símbolo especial sp contém apontador para a pilha

Acções na invocação de funções (relembrar)

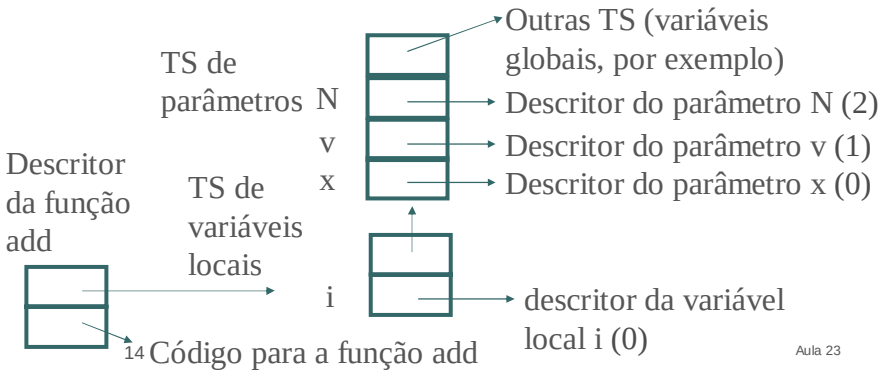
- Invocadora
  - Definir parâmetros de acordo com a convenção de invocações
  - Definir endereço de retorno utilizando a convenção de invocações
  - Saltar para a função invocada
- Invocada
  - Alocar stack frame = deslocar para baixo o apontador da pilha (sp)
  - computar
  - Definir o valor de retorno de acordo com a convenção de invocações
  - Libertar stack frame = deslocar para cima o apontador da pilha (sp)
  - Retornar para a função invocadora

Gestão da Pilha (relembrar)

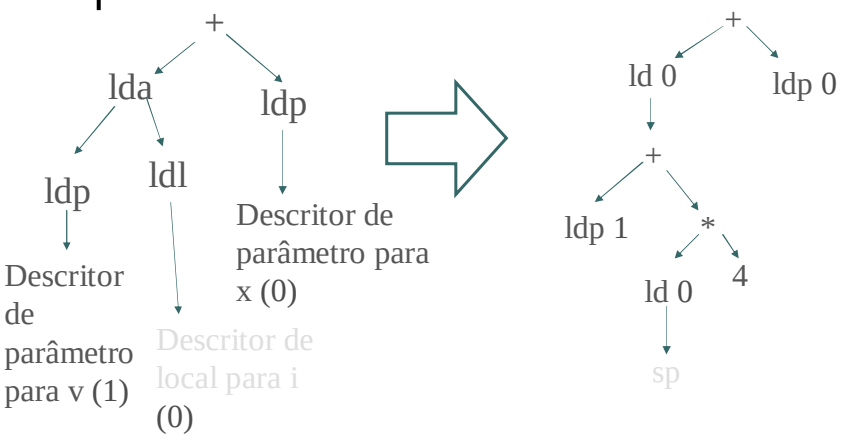
- Determinar tamanho da *stack frame*
  - Alocar quando se entra na função
  - Libertar imediatamente antes do retorno da função
  - Guarda todas as variáveis locais
  - Mais espaço para parâmetros (quando estes ultrapassam em número o número de registos convencionados como argumentos de funções)
  - Assume que todas as variáveis locais e os parâmetros têm o comprimento de uma palavra
- Determinar *offsets* das variáveis locais e dos parâmetros
- Computar *offsets* das variáveis locais e dos parâmetros
  - Guardados nas tabelas de símbolos de locais e de parâmetros
  - Continua a usar nós ldp para aceder aos parâmetros

Eliminação de nós ldl

- Uso de *offsets* na tabela de símbolos locais e sp
- Substituir nós ldl por nós ld
- Exemplo de *offsets* para locais e parâmetros



Exemplo: v[i]+x



Nós Enter e Exit para a função add

```
void add(int x, int[] v, int N) {
   int i;
   ...
}
```

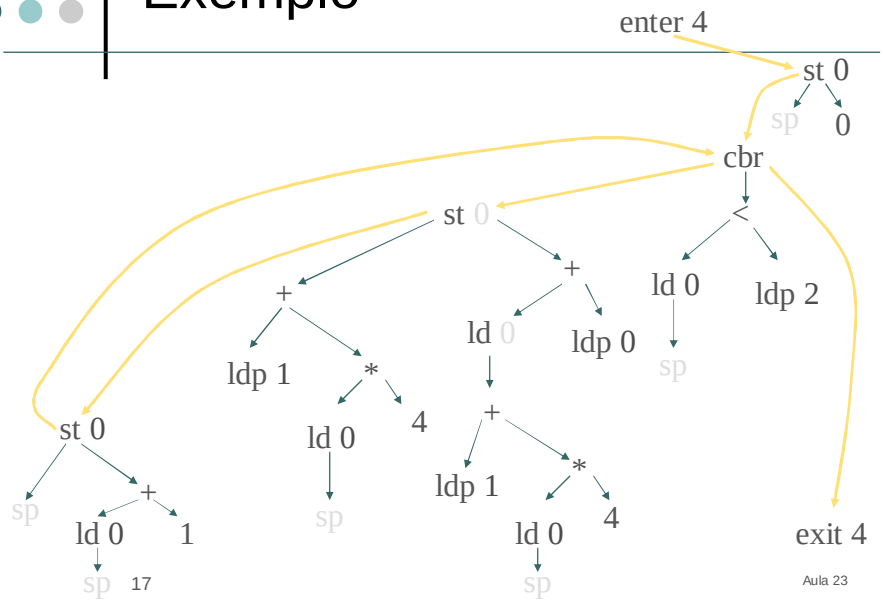
enter **4**

....

exit **4**

- Qual o espaço na pilha para a função add?
  - **4 bytes (espaço para i)**
    - Assumindo palavras de 4 bytes
    - Assumindo parâmetros da função em registos usados para passar argumentos
- Nós enter e exit são anotados com o valor do espaço na pilha necessário para a função

Exemplo



Sumário da IR de nível baixo

- Acessos a arrays traduzidos para nós ld ou st
  - Endereço é o endereço base do array (apontador) + (index * element size)
- Acessos locais traduzidos para nós ld ou st
  - Endereço em sp, offset é o offset local
- Acesso a parâmetros traduzidos para:
  - Instruções lpd – especificar número de parâmetro
- Nós Enter e Exit de uma função identificam tamanho da pilha utilizado

Sumário

- Tradução de árvores sintácticas para IR de nível alto
  - Preserva o fluxo de controlo estruturado
  - Representação eficiente para análise de nível alto e optimizações
- Tradução de IR de nível alto para IR de nível baixo
  - Espaço de endereçamento planar
  - Remoção da estrutura do fluxo de controlo, substituição por saltos condicionais
- Movimento em direcção à máquina alvo