Alocação de Registradores

Introdução

- A IR e a seleção de instruções assumiram que o número de registradores era infinito
- Objetivo:
  - Atribuir registradores físicos (da máquina) para os temporários usados nas instruções
  - Se possível, atribuir a fonte e o destino de MOVES para o mesmo registrador
    - Eliminando os MOVES inúteis

Introdução

- Grafo de Interferência (IG):
  - Temos arestas entre t1 e t2 se eles não podem ocupar o mesmo registrador
  - Live ranges têm intersecção
  - Restrições da arquitetura
    - a = a + b não pode ser atribuido ao r12
- O problema se transforma em um problema de coloração de grafos

Coloração do IG

- Queremos colorir o IG com o mínimo de cores possíveis, de maneira que nenhum par de nós conectados por uma aresta tenham a mesma cor
  - Coloração de vértices
  - As cores representam os registradores
  - Se nossa máquina tem k registradores
    - Se encontrarmos uma k-coloração para o IG
      - Essa coloração é uma alocação válida dos registradores

Coloração do IG

- E se não existir uma k-coloração?
  - Então teremos que colocar alguns dos temporários ou variáveis na memória
  - Operação conhecida como *spilling*
- Coloração de vértices é um problema NP-Completo
  - Logo, alocação de registradores por coloração também é
- Existe uma aproximação linear que traz bons resultados

Coloração por Simplificação

- Dividida em 4 fases:

1. Build:
   - Construir o IG
   - Usa a análise de longevidade
2. Simplify:
   - Heurística
   - Suponha que o grafo G tenha um nó *m* com menos de k vizinhos
   - K é o número de registradores
   - Faça G’ = G – {m}
   - Se G’ pode ser colorido com k cores, G também pode

Coloração por Simplificação

2. Simplify:
   - Leva a um algoritmo recursivo (pilha)
     - Repetidamente:
       - Remova nós de grau menor que K
       - Coloque na pilha
     - Cada remoção diminui o grau dos nós em G, dando oportunidades para novas remoções

Coloração por Simplificação

3. Spill:
   - Em algum momento não temos um nó com grau < k
   - A herística falha
   - Temos que marcar algum nó para spill
   - A escolha desse nó é também uma heurística
     - Nó que reduza o grau do maior número de outros nós
     - Nó com menor custo relacionado as operações de memória

Coloração por Simplificação

4. Select:
   - Atribui as cores
   - Reconstrói o grafo G adicionando os nós na ordem determinada pela pilha
   - Quando adicionamos um nó, devemos ter uma cor para ele dado o critério de seleção usado para remover
   - Isso não vale para os nós empilhados marcados como spill
     - Se todos os vizinhos já usarem k cores, não adicionamos no grafo
     - Continua o processo

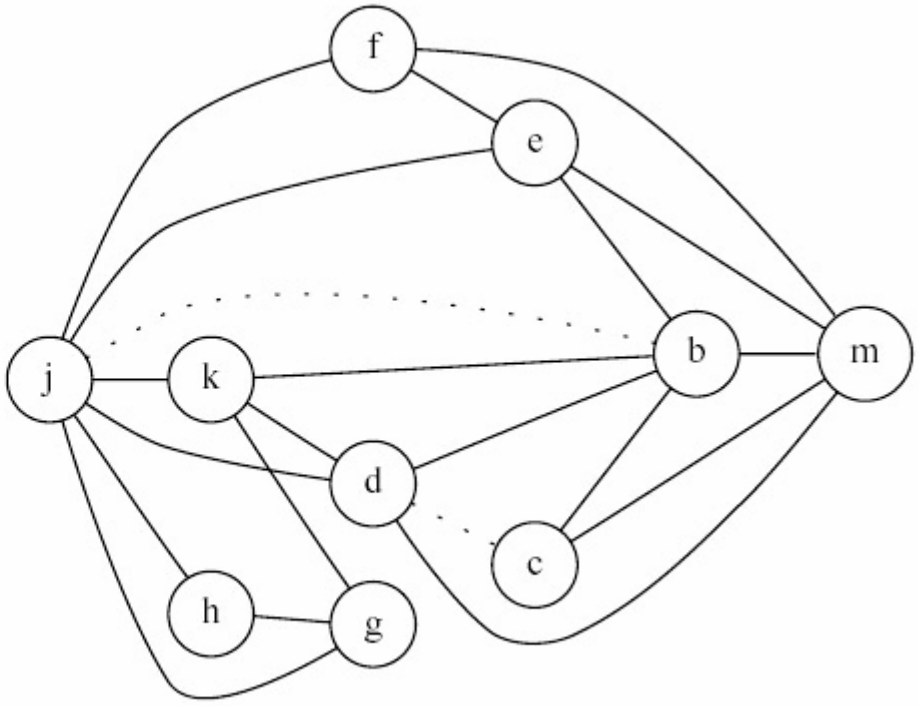
Coloração por Simplificação

5. Start Over:
   - Pode ser que o Select não consiga atribuir uma cor a algum nó
   - Reescreve o programa para pegar esse valor da memória antes de cada uso e armazená-lo de volta após o uso
   - Isso gera novos temporários
     - Com live ranges mais curtas
   - O algoritmo é repetido desde a construção do IG
   - O processo acaba quando Select tiver sucesso para todos os vértices

Exemplo

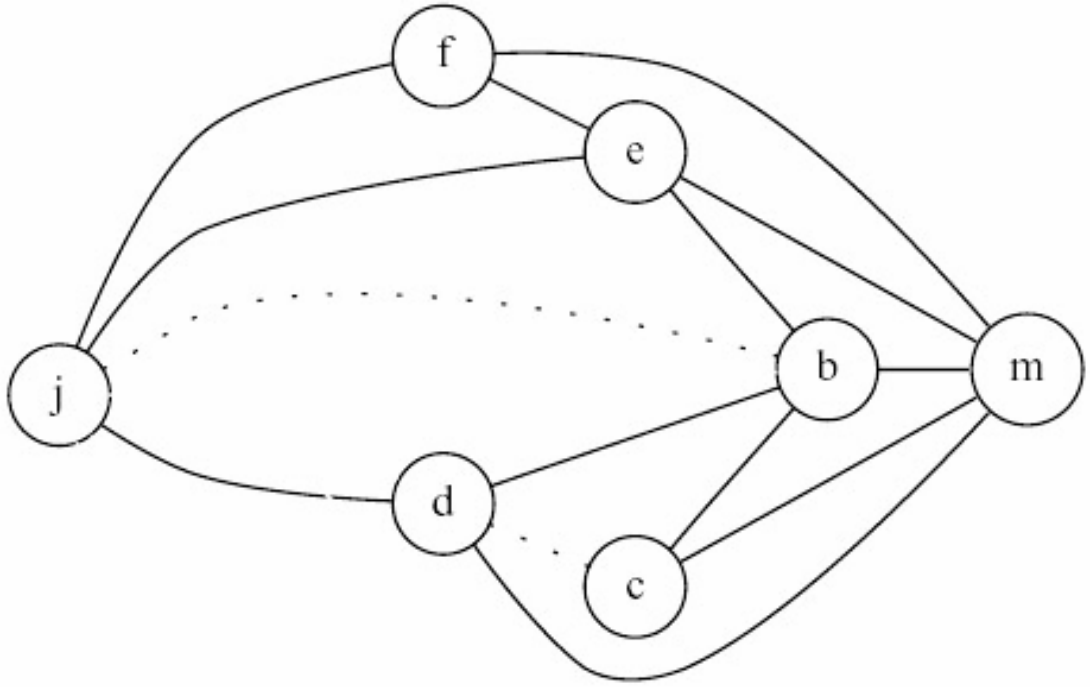
- Suponha que temos 4 registradores

```
live-in: k j
        g := mem[j+12]
        h := k - 1
        f := g * h
        e := mem[j+8]
        m := mem[j+16]
        b := mem[f]
        c := e + 8
        d := c
        k := m + 4
        j := b
live-out: d k j
```



Exemplo

- Removendo h, g , k



Exemplo

- Final

m	1
c	3
b	2
f	2
e	4
j	3
d	4
k	1
h	2
g	4

(a) stack (b) assignment