Geração de Código

Algoritmo de Escalonamento de instruções – List Scheduling

Geração de Código

- O algoritmo de list scheduling é um algoritmo amplamente utilizado para determinar o momento (quando) e o local (onde) em que cada instrução deve executar sobre um processador específico
- Embora aplicado aqui para o caso de um processador simples (sem recursos de paralelismo) pode ser estendido para processadores com múltiplas unidades funcionais e que exploram diferentes granularidades de paralelismos

Algoritmo de list scheduling

- Entrada: um DAG montado a partir de uma expressão
- Saída: seqüência de instruções escalonadas (DAG escalonado)
- Método: o algoritmo utiliza alguns conjuntos para determinar a ordem de execução das instruções. Utiliza 2 fases.

Algoritmo de list scheduling

- Método: fase 1
  - Atravessar o DAG das folhas para a raíz, com DAG “invertido”, computando:
    - Delay(n)= Latência(n) (Etime(n)), se n é uma folha

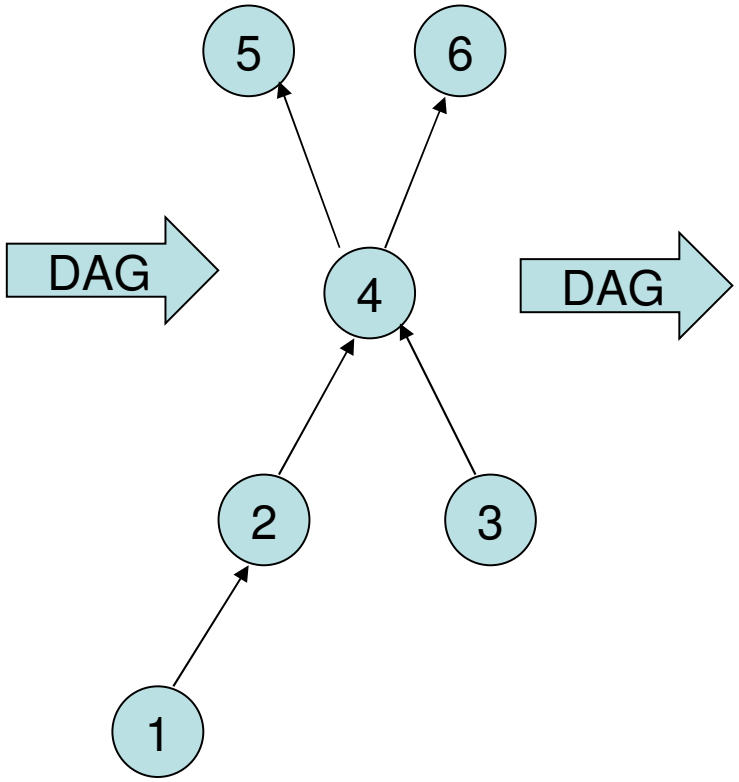
      atraso máximo (soma da latência de n + atrasos máximos dos demais nós do caminho crítico) para uma das raízes, caso n não é uma folha

Algoritmo de list scheduling

- Método: fase 2
  - Atravessar o DAG das raízes para as folhas (considere DAG “invertido”), computando:
    - Cands=conjunto de nós candidatos (independentes) para serem escalonados
    - MCands=Conjunto de nós em Cands com maior atraso (maior Delay)
    - ECands=conjunto de nós em MCands cujo tempo de permanência em MCands é menor ou igual a variavel Curtime
  - Retirar de ECands o nó com menor Etime e adicionar para o conjunto de nós já escalonados
  - Repetir esse passo até que o conjunto sched (conjunto de nós já escalonados) contém todos os nós do DAG

Algoritmo de list scheduling

- Exemplo: Escalonar o código a seguir usando o algoritmo de list scheduling
- 1) z=10
- 2) y=x+5
- 3) z=20
- 4) k=y+z
- 5) i=k*2
- 6) j=k+1



Nó	Etime
1	1
2	1
3	1
4	1
5	1
6	2

Algoritmo de list scheduling

- Exemplo: Fase 1: Computar delay

Nó	Delay
1	5
2	4
3	4
4	3
5	1
6	2

Algoritmo de list scheduling

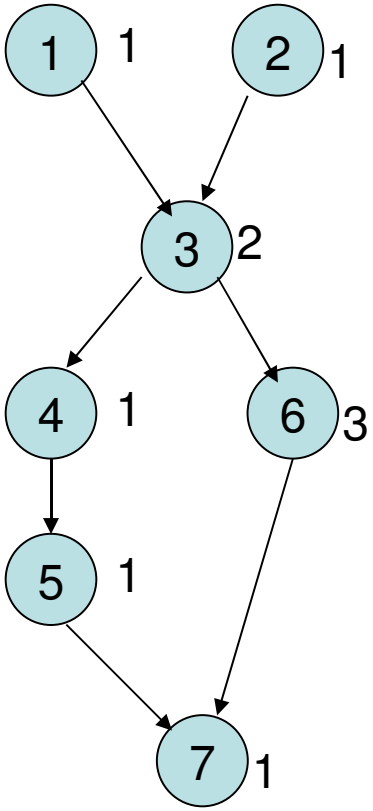
- Exemplo: Fase 2: computar conjuntos e determinar o escalonamento

Cands	MCands	ECands	Sched	Curtime
1,3	1	1	[1]	1
3,2	3,2	3	[1,3]	2
2	2	2	[1,3,2]	3
4	4	4	[1,3,2,4]	4
5,6	6	6	[1,3,2,4,6]	5
5	5	5	[1,3,2,4,6, 5]	6

  - A seqüência de nós escalonados é, então, 1, 3, 2, 4, 6, 5

Algoritmo de list scheduling

- Exercício 1: Executar o algoritmo de list scheduling no DAG a seguir. Os números à direita dos nós indica a latência



Questões:

2) Qual é o menor escalonamento possível se não houvesse restrições de recursos?

3) E se todas as operações fossem independentes?

Algoritmo de list scheduling

- Exercício 2: Considere agora a existência de uma unidade de memória (para instruções lw e sw) e uma unidade aritmética (para instruções lógicas e aritméticas). As operações de lw têm latência=2. As demais operações possuem latência=1