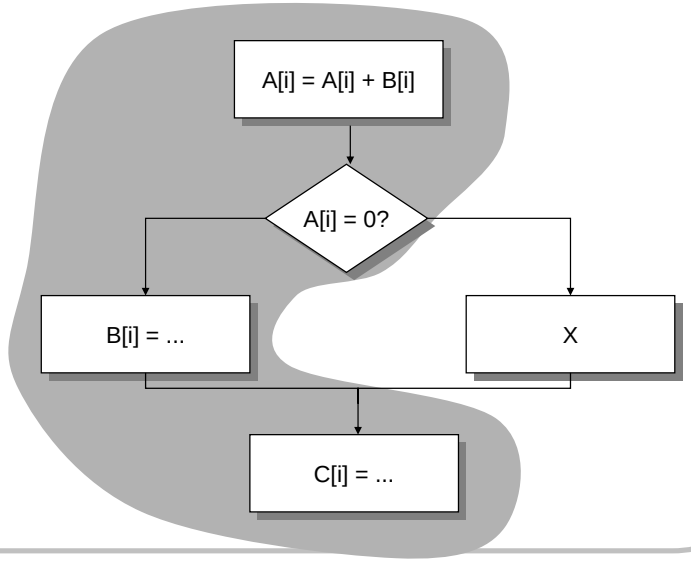
Aula 12: Término de Pipelines Avançados Hierarquia de Memória — Motivação, Definições, Quatro Perguntas Fundamentais

Trace Scheduling

- Permite encontrar paralelismo cruzando branches de IFs vs. branches de LOOPs
- Dois passos:
  - *Trace Selection*
    - » Encontre sequência mais provável de blocos básicos (*trace*) a partir de sequência estaticamente prevista de código sequencial
  - *Trace Compaction*
    - » Comprima *trace* no menor número de instruções VLIW o possível
    - » Necessita de manter informações para recuperação em caso de previsão errônea

Seleção de *Trace*



Suporte em HW para Mais ILP

- *Speculation*: permite uma instrução executar sem nenhuma consequência, mesmo se branch não for tomado (“HW undo”)
- Geralmente combinado com escalonamento dinâmico
- Tomasulo: separa bypass especulativo dos resultados do bypass real dos resultados
  - – Quando instr. não for mais especulativa, escreva os resultados (*instruction commit*)
  - – Executa fora de ordem, mas *commit* em ordem

Suporte em HW para Mais ILP

- Necessita de buffer em HW para resultados de instruções *uncommitted*: *reorder buffer*
  - *Reorder buffer* pode ser fonte de operandos
  - Quando instr. *commit*, resultado pode ser encontrado nos registradores
  - 3 campos: tipo instr, destino, valor
  - Use número do *reorder buffer* ao invés do número da *reservation station*

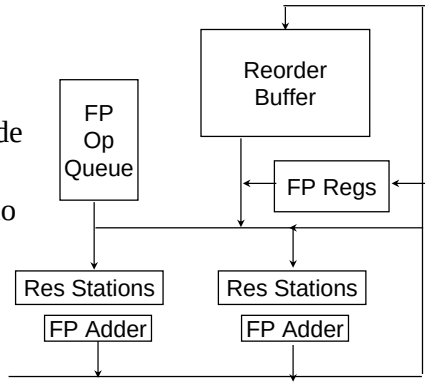


Figura 4.34, página 311

Quatro Passos do Algoritmo de Tomasulo com Especulação

1. Issue—carrega instrução da fila de instruções de FP

   Se *reservation station* ou *reorder buffer slot* livre, issue instr & envie operandos & num. *reorder buffer* para destino.

2. Execução—(EX)

   Quando ambos operandos estiverem na *reservation station*, execute;

3. Escrita de Resultado—término da execução(WB)

   Escreva no CDB para todas as FUs & reorder buffer aguardando resultado; marque *reservation station* livre.

4. Commit—grave resultado com resultado de *reorder*

   Quando instr. no topo do *reorder buffer* & resultado estiver presente, grave resultado em registrador ou memória

Limites para ILP

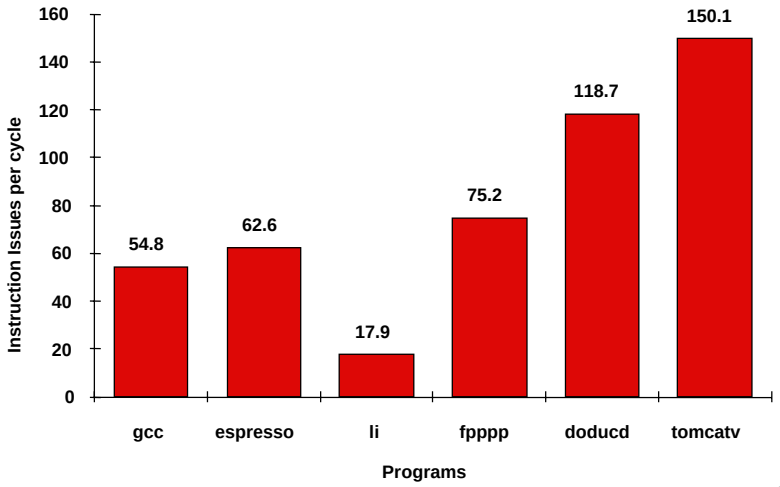
Modelo inicial de hardware; compiladores MIPS

1. *Register renaming*–número infinito de registradores virtuais para evitar todos os hazards de WAW & WAR
2. *Branch prediction*–perfeito; sem previsões erradas
3. *Jump prediction*–previsões perfeitas => máquina com especulação perfeita & buffer de instruções ilimitado
4. *Análise de sinônimos de memória*–endereços são conhecidos e stores podem ser movidos antes de loads se os endereços não forem iguais

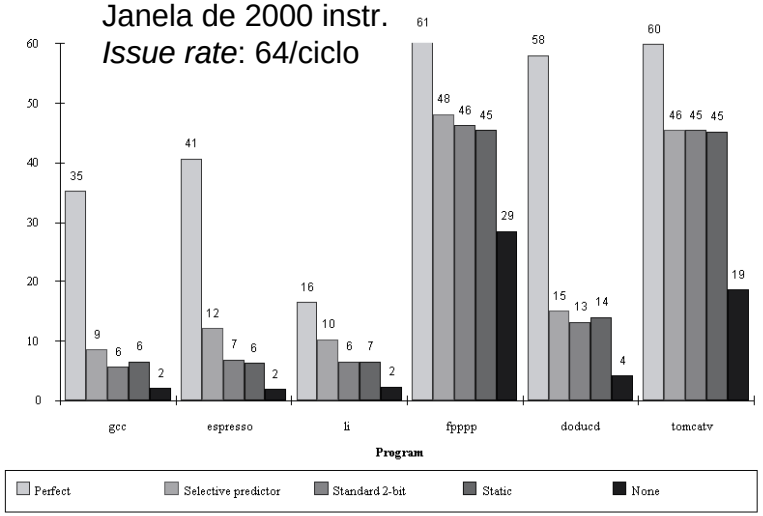
1 ciclo de latência para todas as instruções

Limite Superior para ILP

(Figure 3.1, p 157)



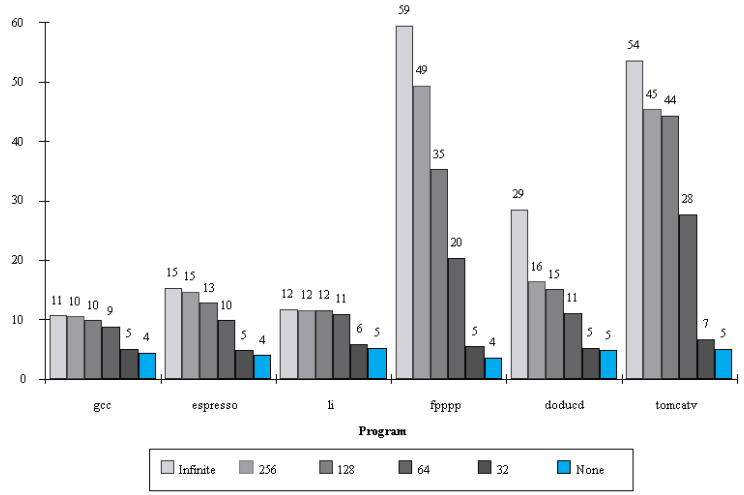
HW Real: Impacto de Branches



Perfect Pick Cor. or BHT BHT (512) Profile

HW Real: Impacto de Registradores

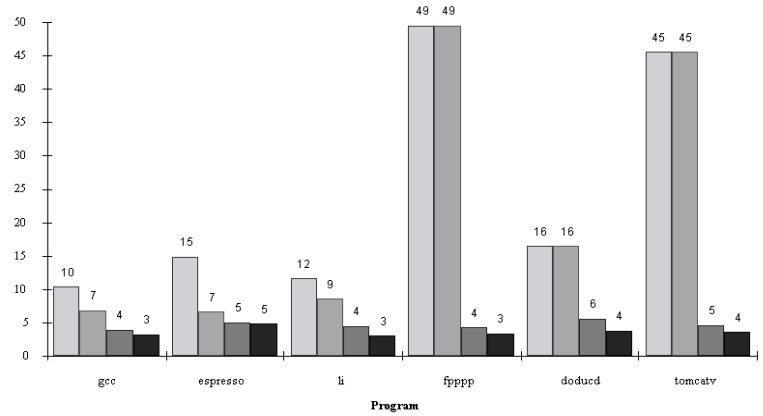
Figure 3.4, p 163



Infinite 256 128 64 32 None

HW Real: Impacto de Alias

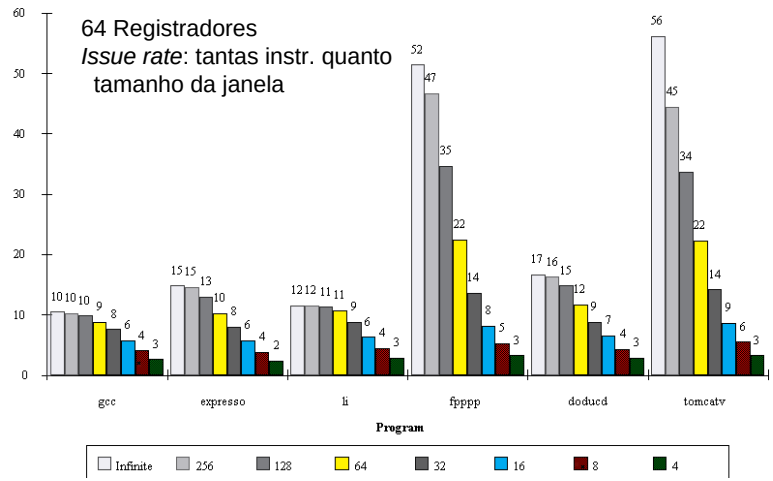
Figure 3.6, p 164



Perfect Global/Stack perf; heap conflicts Inspec. Assem. None

HW Real: Impacto da Janela de Instrs

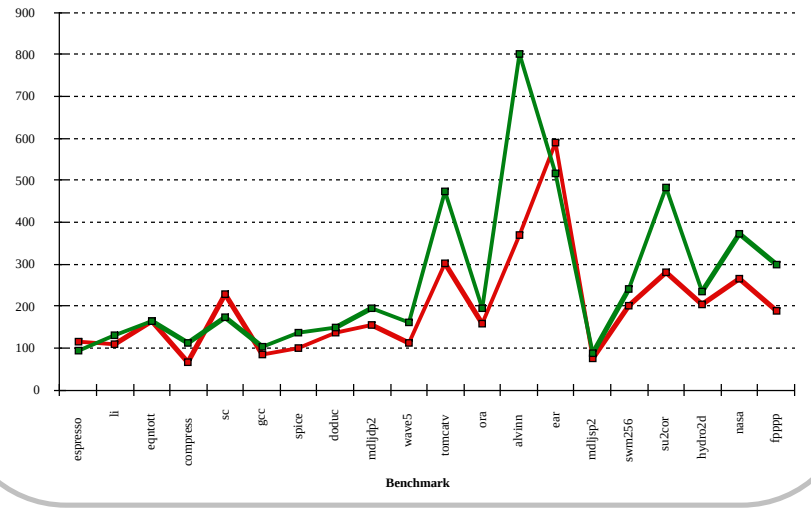
(Figure 4.48, Page 332)



Infinite 256 128 64 32 16 8 4

Força Bruta vs. Rapidez

- 8-scalar IBM Power-2 @ 71.5 MHz (5 stage pipe)
  vs. 2-scalar Alpha @ 200 MHz (7 stage pipe)



Conclusões

- Mais transistores gastos para manter compatibilidade do que para melhorar performance
- Processador superescalar adiciona complexidade cujo preço pode ser muito alto
  - O que é melhor superescalar 8x ou superescalar 2x com um clock mais rápido?
  - Custo do chip x performance da máquina
- Tamanho de janela é um dos obstáculos principais:
  - Janela de 32 instrs -> 900 comparações por ciclo