

Arquitetura e Organização de Computadores II

Especulação baseada em Hardware

Prof. Nilton Luiz Queiroz Jr.

Especulação baseada em Hardware

- Quanto mais se explora o paralelismo em nível de instrução maior o peso de dependências de controle;
- Processadores de alta capacidade de despacho podem precisar executar mais de um desvio por ciclo;
 - Para explorar mais paralelismo é necessário contornar as dependências de controle



Especulação baseada em Hardware

- Como contornar as dependências de controle?
 - Especular o resultado dos desvios e executá-los como se as escolhas fossem corretas;
 - Uma extensão sutil em relação a previsão de desvio com escalonamento dinâmico;
 - Buscar, enviar e executar instruções como se elas fossem as instruções a serem executadas;
 - Escalonamento dinâmico apenas busca e envia;



Especulação baseada em Hardware

- A especulação baseada no hardware combina três ideias fundamentais:
 - Previsão dinâmica de desvio;
 - Para escolher quais instruções executar;
 - Especulação;
 - Permite que as instruções sejam executadas antes da resolução das dependências de controle;
 - É necessário desfazer os efeitos de uma sequência de instruções que foram executadas e não deviam;
 - Escalonamento dinâmico;



Especulação baseada em Hardware

- A especulação baseada em hardware segue o fluxo previsto de valores de dados para escolher quando executar as instruções;
 - Execução de fluxo de dados;
 - Operações são executadas quando seus dados ficam prontos;



Especulação baseada em Hardware

- Para aplicar a especulação baseada em Hardware pode-se estender o algoritmo de Tomasulo;
 - O bypass deve ser separado dos resultados entre as instruções;
 - Com essa separação uma instrução pode ser executada e enviar seus resultados para outras;
 - Nada será alterado até que seja “confirmado” que aquela instrução é a correta;
 - Usar o valor do bypass é como realizar uma leitura especulativa de registrador;
 - Não se sabe se a instrução está fornecendo o valor correto;
 - Permite-se que a instrução atualize o valor quando ela não for mais especulativa;
 - Quando a predição for “confirmada”;
 - A etapa de atualizar o valor é chamada de confirmação de instrução;
 - Instruction commit;



Especulação baseada em hardware

- A ideia central por trás da implementação da especulação é executar fora de ordem, porém confirmar em ordem;
 - Impede ações irreversíveis até a confirmação da instrução;
- Com especulação é necessário separar:
 - Concluir execução;
 - Confirmar instrução;
 - Para confirmar instrução é necessário buffers no hardware;
 - Buffers de reordenação;
 - Mantem as instruções que terminaram porém ainda não foram confirmadas;



Buffer de reordenação

- Os buffers de reordenação (Reorder buffer - ROB) oferecem registradores adicionais
 - Semelhante as estações de reserva;
- Mantém o resultado do momento em que a instrução termina até o momento que a instrução é confirmada;
 - Isso os torna fonte de operandos para instruções;
 - A renomeação de registradores “ocorre neles”



Buffer de reordenação

- Cada entrada no buffer de reordenação contém quatro campos:
 - O tipo de instrução;
 - O campo de destino;
 - O campo de valor; e
 - O campo que indica se está pronto;



Campos do buffer de reordenação

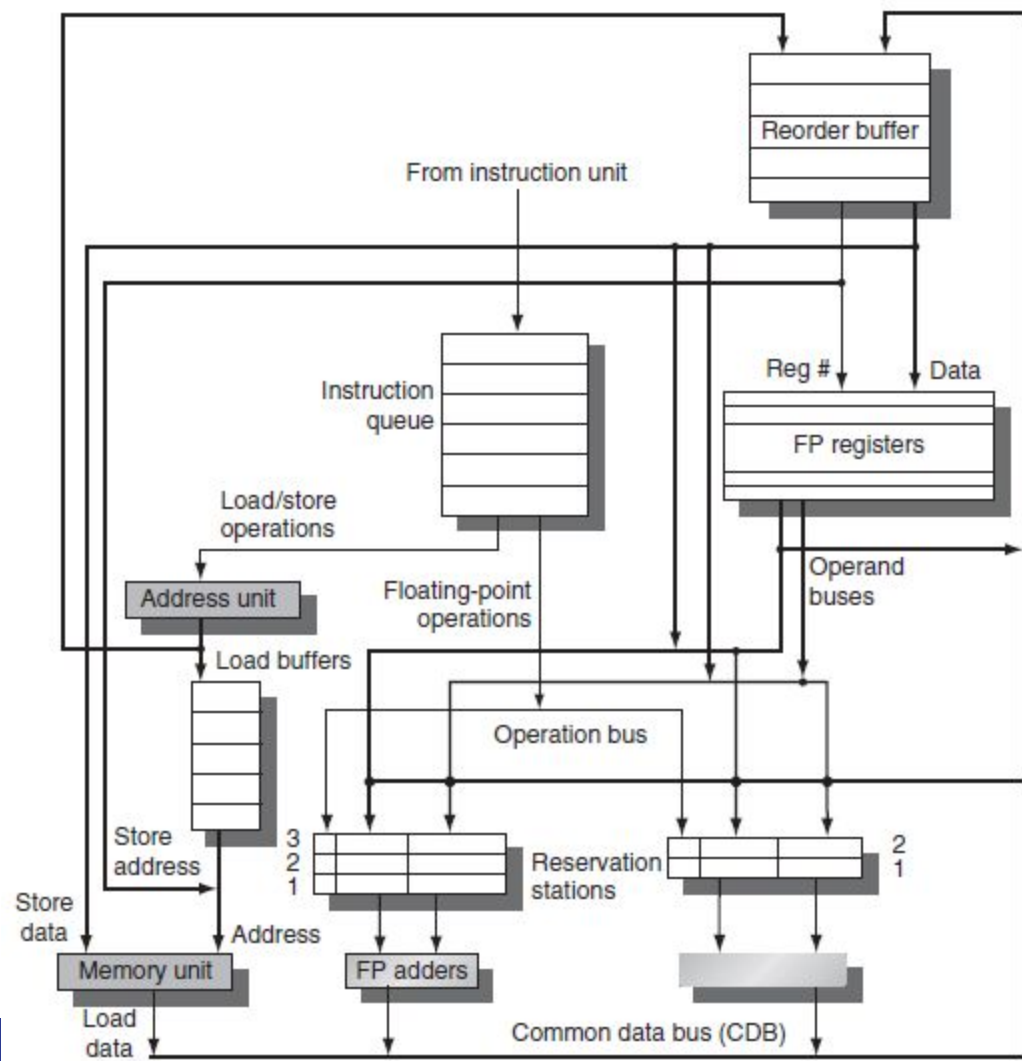
- Tipo de instrução:
 - Indica se a instrução é:
 - Desvio;
 - Não possui destino para o resultado;
 - Sotre:
 - Possui como destino um endereço de memória;
 - Operação de registrador (ULA/UFP ou load);
 - Possui como destino um registrador;
- Campo de destino:
 - Fornece o número do registrador ou endereço de memória;




Campos do buffer de reordenação

- Campo de valor:
 - Usado para armazenar o valor do resultado;
 - Armazena no intervalo de término da execução até o commit;
- Campo pronto:
 - Indica se a instrução terminou ou não sua execução;
 - Quando a instrução estiver pronta o valor do campo de valor pode ser usado;





Especulação baseada em Hardware

- Note que não existem mais buffers de store;
 - O buffer de reordenamento substitui os buffers de store;
 - Os stores ainda são executados em duas etapas;
 - A segunda etapa passa a ser a confirmação da instrução;
 - As estações de reserva ainda são usadas para armazenar instruções entre os momentos que são enviadas e iniciam sua execução;
 - Armazenam operações e operandos;
 - Os resultados são identificados usando um número de entrada do ROB;
 - Anteriormente usava-se o número da estação de reserva;
 - Os ROBs devem ser rastreados pelas estações de reserva;
- 

Caminho da instrução

- A execução de uma instrução passa a ser dividida em 4 etapas:
 - Despacho;
 - Execução;
 - Escrita de resultado;
 - Confirmação (commit);



Caminho da instrução

- Despacho:
 - Carrega a nova instrução da fila de execução;
 - Se houver espaço no ROB e nas estações de reserva envie a instrução;
 - Envie os operandos para estação de reserva se estiverem disponíveis;
 - Atualize as entradas de controle para indicar que os buffers estão ocupados;
 - Envie o número do ROB alocado para o resultado para estação de reserva
 - Para que a estação de reserva possa ler o valor quando ele for enviado ao CDB;



Caminho da instrução

- Execução:
 - Monitore o CDB enquanto os operando não estão prontos;
 - Verificação de hazards RAW;
 - Execute a operação quando os dois operandos estiverem prontos;
 - Stores só necessitam do registrador base disponível nessa etapa;
 - A execução para um store é apenas o cálculo do endereço efetivo



Caminho da instrução

- Escrita do resultado:
 - Escreva o resultado no CDB quando disponível;
 - Para que o ROB e as estações de reserva que precisam dele peguem-o;
 - Marque a estação de reserva como disponível;
 - Para instruções store:
 - Se o valor a ser armazenado está disponível coloque-o no campo vaor do ROB;
 - Caso contrario monitore o CDB;



Caminho da instrução

- Confirmação (Commit):
 - Ocorre quando a instrução alcança a primeira posição na fila de confirmação;
 - Existem três diferentes sequências ações para um commit:
 - Quando uma instrução alcança o início da fila e seu resultado está presente no buffer, assim o registrador é atualizado e a instrução removida;
 - No caso de store o valor atualizado é a memória;
 - Quando um desvio com predição incorreta atinge o início da fila o ROB é esvaziado e a execução é reiniciada no sucessor correto do desvio;
 - Quando um desvio com predição correta atinge o início da fila ele é terminado;




Especulação baseada em Hardware

- Como o commit é feito em ordem o modelo de interrupção é preciso;
 - Quando uma instrução gera uma interrupção, todas as instruções seguintes podem ser esvaziadas do ROB;
- Alguns arquitetos e usuários decidiram que exceções de ponto flutuante imprecisas são aceitáveis;
 - Pois com uma delas o programa provavelmente terminará;
- Outras instruções precisam continuar com a execução normal;



Especulação baseada em Hardware

- Processadores que especulam tentam recuperar o mais cedo possível após um desvio ser mal previsto;
 - Pode-se limpar o ROB para todas instruções que vem após o desvio;
 - Processadores especulativos são mais sensíveis à predição de desvio;
 - Os impactos causados por um desvio mal previsto são maiores;
 - Exceções são tratadas pelo seu não reconhecimento até que estejam prontas para serem confirmadas;
 - Se a instrução especulada levantar uma exceção será feito um registro no ROB;
 - Caso um desvio que a precede for tomado, e sua especulação tenha sido feita errada ela sera apagada;
 - Caso ela atinja o início do ROB a exceção será tomada;
- 

Especulação baseada em Hardware

- Existe uma diferença importante no modo que os stores são tratados no Algoritmo de Tomasulo e em um processador especulativo:
 - Assim que um store alcançou a escrita do resultado ele pode atualizar a memória (desde que o resultado a ser escrito esteja disponível);
 - Num processador especulativo os stores só atualizam a memória quando alcançam o início do ROB;
 - Isso garante que a memória não seja atualizada até que uma instrução não seja mais especulativa;



Especulação baseada em Hardware

- O algoritmo a ser visto apresenta uma simplificação significativa para stores:
 - Exige espera no estágio de escrita de resultado pelo registrador operando-fonte;
 - O valor é então movido para o campo Vk da estação de reserva do store para o campo Valor da entrada equivalente ao store no ROB;
 - Esse valor não precisa chegar imediatamente antes do store ser confirmado;
 - Pode ser colocado diretamente na entrada de store do ROB pela instrução de origem;
 - Para isso o hardware deve acompanhar o valor de origem a ser armazenado na entrada de store do ROB e pesquisando o ROB a cada término de instrução para pesquisar os stores dependentes;



Especulação baseada em Hardware

- Instruções ULA/FPU

- Despacho

```
if (RegisterStat[rs].Busy){  
    h ← RegisterStat[rs].Reorder;  
    if (ROB[h].Ready){  
        RS[r].Vj ← ROB[h].Value;  
        RS[r].Qj ← 0;  
    }else{  
        RS[r].Qj ← h;  
    }  
} else {  
    RS[r].Vj ← Regs[rs];  
    RS[r].Qj ← 0;  
}  
RS[r].Busy ← yes;  
RS[r].Dest ← b;
```

```
ROB[b].Instruction ← opcode;  
ROB[b].Dest ← rd;  
ROB[b].Ready ← no;  
if (RegisterStat[rt].Busy){  
    h ← RegisterStat[rt].Reorder;  
    if (ROB[h].Ready){  
        RS[r].Vk ← ROB[h].Value;  
        RS[r].Qk ← 0;  
    }else {  
        RS[r].Qk ← h;  
    }  
} else {  
    RS[r].Vk ← Regs[rt];  
    RS[r].Qk ← 0;  
}  
RegisterStat[rd].Reorder ← b;  
RegisterStat[rd].Busy ← yes;  
ROB[b].Dest ← rd;
```

Especulação baseada em Hardware

- Execução

Compute results—operands are in V_j and V_k

- Write Back

```
b ← RS[r].Dest;
RS[r].Busy ← no;
∀ x(
  if (RS[x].Qj==b) {
    RS[x].Vj ← result;
    RS[x].Qj ← 0
  }
);
∀ x(
  if (RS[x].Qk==b) {
    RS[x].Vk ← result;
    RS[x].Qk ← 0
  }
);
ROB[b].Value ← result;
ROB[b].Ready ← yes;
```


Especulação baseada em Hardware

- Commit

```
d ← ROB[h].Dest;
if (ROB[h].Instruction==Branch){
    if (branch is mispredicted){
        clear ROB[h], RegisterStat;
        fetch branch dest;
    }
} else {
    Regs[d] ← ROB[h].Value;
}
ROB[h].Busy ← no;
if (RegisterStat[d].Reorder==h) {
    RegisterStat[d].Busy ← no;
}
```



Especulação baseada em Hardware

- Instruções Load

- Despacho

```
if (RegisterStat[rs].Busy){
    h ← RegisterStat[rs].Reorder;
    if (ROB[h].Ready){
        RS[r].Vj ← ROB[h].Value;
        RS[r].Qj ← 0;
    }else{
        RS[r].Qj ← h;
    }
} else {
    RS[r].Vj ← Regs[rs];
    RS[r].Qj ← 0;
}
RS[r].Busy ← yes;
RS[r].Dest ← b;
```

```
ROB[b].Instruction ← opcode;
ROB[b].Dest ← rd;
ROB[b].Ready ← no;
```

```
RS[r].A ← imm;
RegisterStat[rt].Reorder ← b;
RegisterStat[rt].Busy ← yes;
ROB[b].Dest ← rt;
```



Especulação baseada em Hardware

- Execução

```
RS[r].A  $\leftarrow$  RS[r].Vj + RS[r].A;  
Read from Mem[RS[r].A]
```

- Write Back

```
b  $\leftarrow$  RS[r].Dest;  
RS[r].Busy  $\leftarrow$  no;  
 $\forall x$ (  
  if (RS[x].Qj==b) {  
    RS[x].Vj  $\leftarrow$  result;  
    RS[x].Qj  $\leftarrow$  0  
  }  
);  
  
 $\forall x$ (  
  if (RS[x].Qk==b) {  
    RS[x].Vk  $\leftarrow$  result;  
    RS[x].Qk  $\leftarrow$  0  
  }  
);  
ROB[b].Value  $\leftarrow$  result;  
ROB[b].Ready  $\leftarrow$  yes;
```

Especulação baseada em Hardware

- Commit

$d \leftarrow \text{ROB}[h].\text{Dest};$

$\text{Regs}[d] \leftarrow \text{ROB}[h].\text{Value};$

$\text{ROB}[h].\text{Busy} \leftarrow \text{no};$

```
if (RegisterStat[d].Reorder==h) {  
    RegisterStat[d].Busy ← no;  
}
```



Especulação baseada em Hardware

- Instruções Store

- Despacho

```
if (RegisterStat[rs].Busy){  
    h ← RegisterStat[rs].Reorder;  
    if (ROB[h].Ready){  
        RS[r].Vj ← ROB[h].Value;  
        RS[r].Qj ← 0;  
    }else{  
        RS[r].Qj ← h;  
    }  
} else {  
    RS[r].Vj ← Regs[rs];  
    RS[r].Qj ← 0;  
}  
RS[r].Busy ← yes;  
RS[r].Dest ← b;
```

```
ROB[b].Instruction ← opcode;  
ROB[b].Dest ← rd;  
ROB[b].Ready ← no;
```

```
RS[r].A ← imm;
```



Especulação baseada em Hardware

- Execução

$ROB[h].Address \leftarrow RS[r].Vj + RS[r].A;$

- Write Back

$ROB[h].Value \leftarrow RS[r].Vk;$

- Commit

$d \leftarrow ROB[h].Dest;$

$Mem[ROB[h].Destination] \leftarrow ROB[h].Value;$

```
ROB[h].Busy ← no;  
if (RegisterStat[d].Reorder==h) {  
    RegisterStat[d].Busy ← no;  
}
```




Especulação baseada em Hardware

- Evita-se hazards do tipo WAW e WAR na memória;
 - A fase de commit os elimina;
 - Atualização da memória ocorre em ordem;
- Hazards RAW são mantidos por duas restrições:
 - Loads não podem iniciar sua segunda etapa quando stores no ROB tem o mesmo endereço;
 - Mantém-se a ordem do programa para o cálculo de endereço efetivo de um load com relação a todos stores anteriores;



Especulação baseada em Hardware

- Para sistemas com especulação baseada em Hardware é importante mensurar o tamanho dos buffers de reordenação;
 - Buffers muito grandes fazem com que várias instruções saiam do buffer com previsões erradas;
 - Buffers muito pequenos causam muitos stalls;
 - Na realidade a especulação baseada em Hardware pode ser mais útil em programas de inteiros;
 - Costumam ter código em que o comportamento de desvio é menos previsível;
 - Especulação baseada em Hardware é mais interessante em processadores com múltiplo despacho;
- 

Referências

PATTERSON, D. A.; HENNESSY, J. L. Computer Architecture: A Quantitative Approach. Fifth Edition

