

Arquitetura e Organização de Computadores II

Múltiplo Despacho

Prof. Nilton Luiz Queiroz Jr.

Escalonamento dinâmico e especulação

- Escalonamento dinâmico de instruções e especulação procuram alcançar uma quantidade ideal de 1 ciclo para cada instrução;
- Se tornaria mais interessante se fosse possível alcançar uma menos de 1 ciclo para cada instrução;
 - Ou seja, a quantidade de instruções executadas por ciclo superasse 1;
- Porém, se a quantidade de instruções despachadas por ciclo é de 1 não se torna possível finalizar mais de uma instrução por ciclo;



Processadores de múltiplo despacho

- Processadores de múltiplo despacho contornam esse tipo de situação;
 - Eles conseguem emitir mais de uma instrução por ciclo de clock;
- Eles podem ser divididos em três principais tipos:
 - Superescalares escalonados estaticamente;
 - Very Long Instruction Word (VLIW)
 - Superescalares escalonados dinamicamente;



Processadores de múltiplo despacho

- Processadores superescalares emitem um número variado de instruções por clock;
 - Executam em ordem quando são escalonados estaticamente;
 - Executam fora de ordem quando são escalonados dinamicamente;
- Processadores VLIW enviam um número fixo de instruções por clock;
 - As instruções são pré formatadas como uma única instrução;
 - Pacote de instruções;
 - São escalonados pelo compilador;



Processadores de múltiplo despacho

- Os processadores superescalares com escalonamento estático estão mais próximos dos VLIW;
 - Ambos são escalonados estaticamente;
- Os processadores superescalares com escalonamento tem desvantagens conforme a taxa de despacho cresce;



Processadores de múltiplo despacho

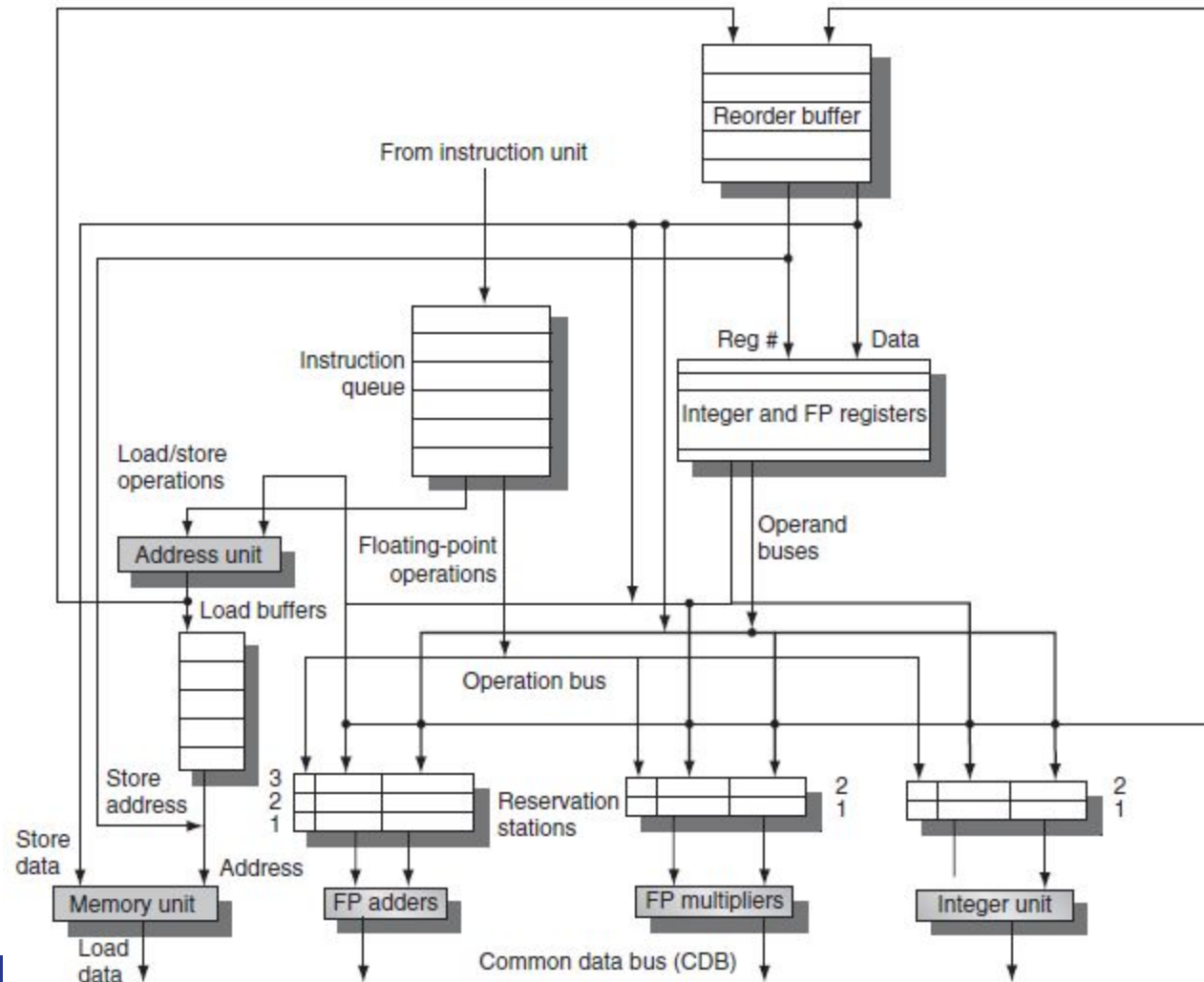
- Processadores modernos costumam juntar:
 - Escalonamento dinâmico;
 - Especulação;
 - Múltiplo despacho;
 - Em geral são três ou mais instruções enviadas por clock;



Processadores de múltiplo despacho

- Suponha a extensão de um processador que tem implementado o algoritmo de tomasulo para dar suporte a um pipeline superescalar;
 - Assuma duas instruções despachadas por ciclo;
 - Uma UF de inteiros e uma UF de ponto flutuante;
- A interação de inteiros e pontos flutuantes é crucial, é interessante que o hardware também implemente a técnica de tomasulo para valores inteiros;
- Para tratar exceções é interessante também utilizar um hardware com especulação;





Processadores de múltiplo despacho

- Ao enviar duas instruções no mesmo ciclo de clock deve não é desejável que se altere a ordem de emissão de ambas;
 - Isso alteraria a semântica do programa;
- Para isso deve-se melhorar a lógica de emissão;
 - A lógica de conclusão também deve ser melhorada, pois é desejável que seja possível confirmar mais de uma instrução por ciclo;



Processadores de múltiplo despacho

- Emitir mais de uma instrução por ciclo é complexo;
 - As instruções que estão sendo emitidas podem depender umas das outras;
- A emissão de duas instruções sem levar em conta as dependências provavelmente geraria resultados incorretos em suas execuções;
- Duas técnicas foram usadas para enviar múltiplas instruções:
 - Emitir instruções em metade de um ciclo;
 - Difícil extensão para despachos maiores;
 - Desenvolver a lógica necessária para emissão;
 - Verificar todas as possíveis dependências entre as instruções
- Para a ampliação da lógica de emissão todas as possíveis dependências entre as instruções devem ser verificadas e tratadas;

Processadores de múltiplo despacho

- É possível generalizar o comportamento da emissão múltipla com os seguintes passos:
 - Definir estações de reserva e ROB para todas instruções a serem emitidas;
 - É possível fazer essa definição antes de se conhecer todos tipos de instruções;
 - Realizam-se os ROBs
 - Garante que existem estações de reserva o suficiente para as instruções;
 - Uma opção é limitar a quantidade de instruções de uma mesma espécie por despacho;
 - Caso somente uma parte das instruções possam ser emitidas quebra-se o conjunto;
 - As demais serão emitidas no próximo despacho
 - Analisar todas as dependências entre as instruções no conjunto de despacho;

Processadores de múltiplo despacho

- Quanto a dependências:
 - Se existir dependências entre duas instruções no mesmo conjunto de despacho use o número do ROB atribuído para atualizar a tabela de reserva para a instrução dependente;
 - Caso contrário use a tabela de reserva existente e a informação do ROB para atualizar as entradas da tabela de reservas para as instruções enviadas;
- Isso tudo deve ser feito em um único ciclo de clock;
 - Uma operação “atômica”;



Exemplo de uma lógica de emissão

- Suponha a emissão de duas instruções;
 - Um load de FP seguido de uma operação de FP;
- As etapas do despacho devem ser:
 - Atualizar as tabelas de reserva para instrução de load que tem um único operando;
 - Como ela é a primeira instrução do conjunto, ocorreria de maneira semelhante a um load sem despacho múltiplo;
 - Atualizar a estação de reserva para apontar para o load
 - Observe que existem três casos:
 - Primeiro operando do load anterior;
 - Segundo operando do load anterior;
 - Ambos operandos do load anterior;
 - A dependência deve ser analisada durante o processamento e as entradas do ROB devem ser alocadas durante essa etapa do despacho para que não ocorram erros na atualização das tabelas de reserva

Exemplo de uma lógica de emissão

- Supondo que o segundo operando venha de uma instrução já despachada, essa etapa seria muito parecida com a verificação do segundo operando de uma operação FP com despacho simples;
 - Deve-se levar em consideração as atualizações feitas na tabela em caso de dependência;
- Por fim atualizam-se as tabelas para a operação de ponto flutuante;
 - É independente do Load;
 - Se mais instruções, seguintes a essa dependessem do resultado produzido por ela, as instruções deveriam pegar os valores alterados por essa tabela de reserva



Exemplo de uma lógica de emissão

- Atualizar as tabelas de reserva para instrução de load que tem um único operando;

```
if (RegisterStat[rs1].Busy){  
    h ← RegisterStat[rs1].Reorder;  
    if (ROB[h].Ready){  
        RS[r1].Vj ← ROB[h].Value;  
        RS[r1].Qj ← 0;  
    } else {  
        RS[r1].Qj ← h;  
    }  
} else {  
    RS[r1].Vj ← Regs[rs];  
    RS[r1].Qj ← 0;  
}  
RS[r1].Busy ← yes;
```

```
RS[r1].Dest ← b1;  
ROB[b1].Instruction ← Load;  
ROB[b1].Dest ← rd1;  
ROB[b1].Ready ← no;  
RS[r].A ← imm1;  
RegisterStat[rt1].Reorder ← b1;  
RegisterStat[rt1].Busy ← yes;  
ROB[b1].Dest ← rt1;
```

Exemplo de uma lógica de emissão

- Atualizar a estação de reserva para apontar para o load
 - Para o primeiro operando;

$RS[r2].Qj \leftarrow b1;$

- Supondo que o segundo operando venha de uma instrução despachada em ciclos anteriores;

```
if (RegisterStat[rt2].Busy){  
    h  $\leftarrow$  RegisterStat[rt2].Reorder;  
    if (ROB[h].Ready){  
        RS[r2].Vk  $\leftarrow$  ROB[h].Value;  
        RS[r2].Qk  $\leftarrow$  0;  
    }else {  
        RS[r2].Qk  $\leftarrow$  h;  
    }  
}  
  
} else {  
    RS[r2].Vk  $\leftarrow$  Regs[rt2];  
    RS[r2].Qk  $\leftarrow$  0;  
}  
RegisterStat[rd2].Reorder  $\leftarrow$  b2;  
RegisterStat[rd2].Busy  $\leftarrow$  yes;  
ROB[b2].Dest  $\leftarrow$  rd2;
```


Exemplo de uma lógica de emissão

- Por fim atualizam-se as tabelas para para a operação de ponto flutuante;

```
RS[r2].Busy  $\leftarrow$  yes;  
RS[r2].Dest  $\leftarrow$  b2;  
ROB[b2].Instruction  $\leftarrow$  FP operation;  
ROB[b2].Dest  $\leftarrow$  rd2;  
ROB[b2].Ready  $\leftarrow$  no;
```



Processadores de múltiplo despacho

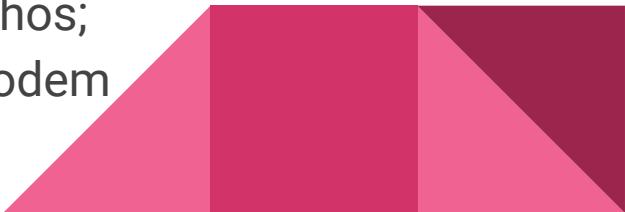
- Também é necessário completar e confirmar diversas instruções em um ciclo;
 - Esse processo é um pouco mais fácil que o de despacho;
- Como as instruções foram emitidas, suas dependências já foram resolvidas;



Exemplo

- Considere o seguinte trecho de código em um processador com múltiplo despacho sem especulação e com especulação;

```
Loop:   LD      R2,0(R1)
        DADDIU  R2,R2,#1
        SD      R2,0(R1)
        DADDIU  R1,R1,#8
        BNE     R2,R3,LOOP
```

- Suponha que existam unidades funcionais de inteiros para cálculo de endereço, de operações e avaliação de desvios;
 - Suponha que os branches devem ser enviados sozinhos;
 - Suponha que até duas instruções de qualquer tipo podem ser confirmadas no mesmo ciclo;
- 

Exemplo

- Quais seriam os ciclos que cada instrução faria as seguintes microoperações
 - Despacho;
 - Execução;
 - Acesso à memória;
 - somente loads e stores;
 - Escrita nos CDB e nos registradores;
 - No caso com especulação seria no CDB e nos ROB's;
 - Commit (no caso de especulação);



Sem especulação

Iteration number	Instructions	Issues at clock cycle number	Executes at clock cycle number	Memory access at clock cycle number	Write CDB at clock cycle number
1	LD R2,0(R1)	1	2	3	4
1	DADDIU R2,R2,#1	1	5		6
1	SD R2,0(R1)	2	3	7	
1	DADDIU R1,R1,#8	2	3		4
1	BNE R2,R3,LOOP	3	7		
2	LD R2,0(R1)	4	8	9	10
2	DADDIU R2,R2,#1	4	11		12
2	SD R2,0(R1)	5	9	13	
2	DADDIU R1,R1,#8	5	8		9
2	BNE R2,R3,LOOP	6	13		
3	LD R2,0(R1)	7	14	15	16
3	DADDIU R2,R2,#1	7	17		18
3	SD R2,0(R1)	8	15	19	
3	DADDIU R1,R1,#8	8	14		15
3	BNE R2,R3,LOOP	9	19		

Com especulação

Iteration number	Instructions	Issues at clock number	Executes at clock number	Read access at clock number	Write CDB at clock number	Commits at clock number
1	LD R2,0(R1)	1	2	3	4	5
1	DADDIU R2,R2,#1	1	5		6	7
1	SD R2,0(R1)	2	3			7
1	DADDIU R1,R1,#8	2	3		4	8
1	BNE R2,R3,LOOP	3	7			8
2	LD R2,0(R1)	4	5	6	7	9
2	DADDIU R2,R2,#1	4	8		9	10
2	SD R2,0(R1)	5	6			10
2	DADDIU R1,R1,#8	5	6		7	11
2	BNE R2,R3,LOOP	6	10			11
3	LD R2,0(R1)	7	8	9	10	12
3	DADDIU R2,R2,#1	7	11		12	13
3	SD R2,0(R1)	8	9			13
3	DADDIU R1,R1,#8	8	9		10	14
3	BNE R2,R3,LOOP	9	13			14

Exemplo

- Observe que:
 - As instruções com especulação, mesmo contendo uma etapa a mais, terminam no ciclo 14;
 - As instruções sem especulação, mesmo com uma quantidade menor de etapas terminam no ciclo 19;
 - Isso ocorre devido ao atraso causado na espera do branch;



Processadores com despacho múltiplo

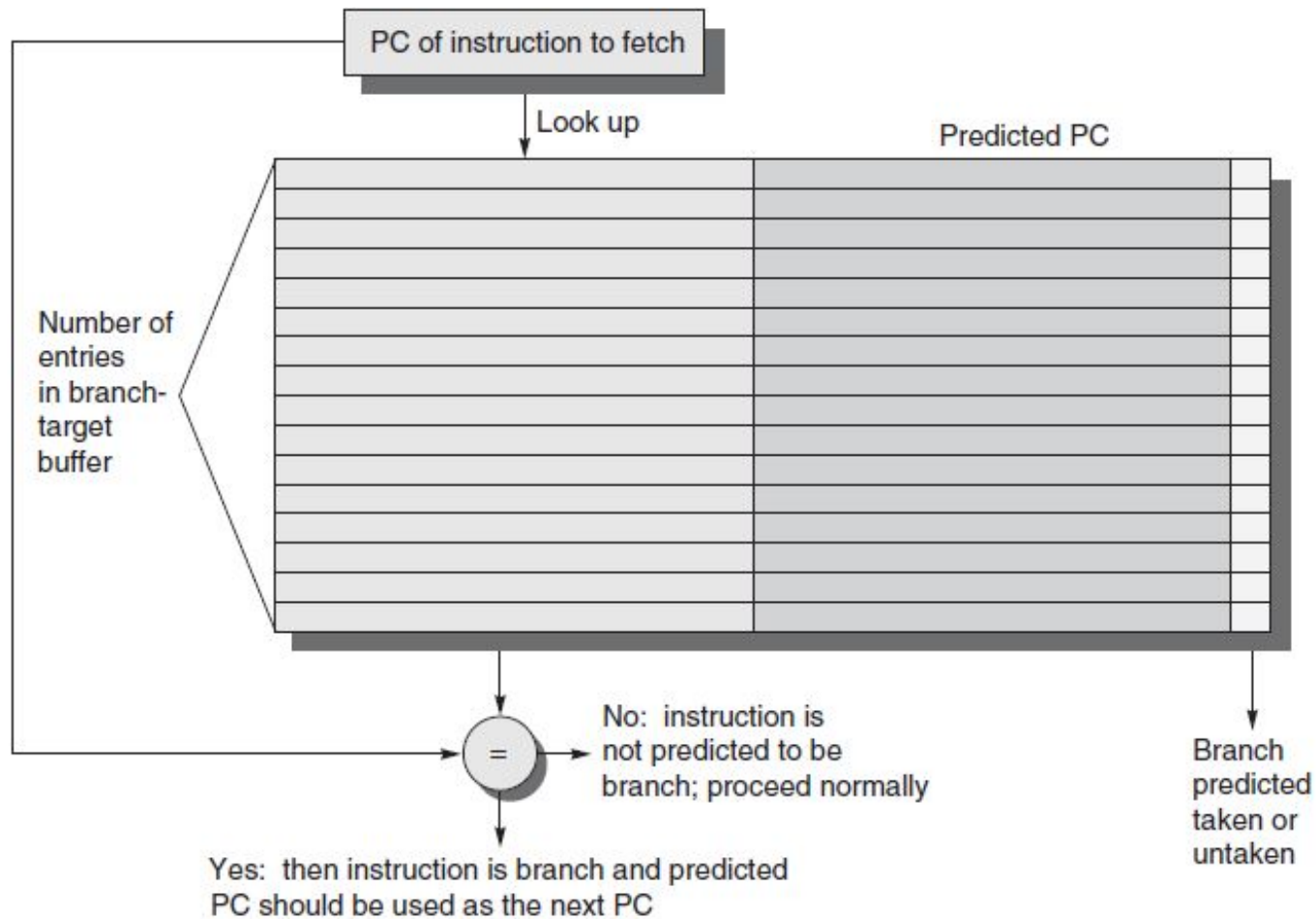
- Processadores com despacho múltiplo requerem que a média de instruções buscadas por clock seja tão grande quanto a média de despacho;
 - O maior desafio é lidar com desvios;
- Dentre as maneiras temos:
 - Branch Target-Buffer (tabela de histórico de desvios);
 - Preditor de endereço de retorno

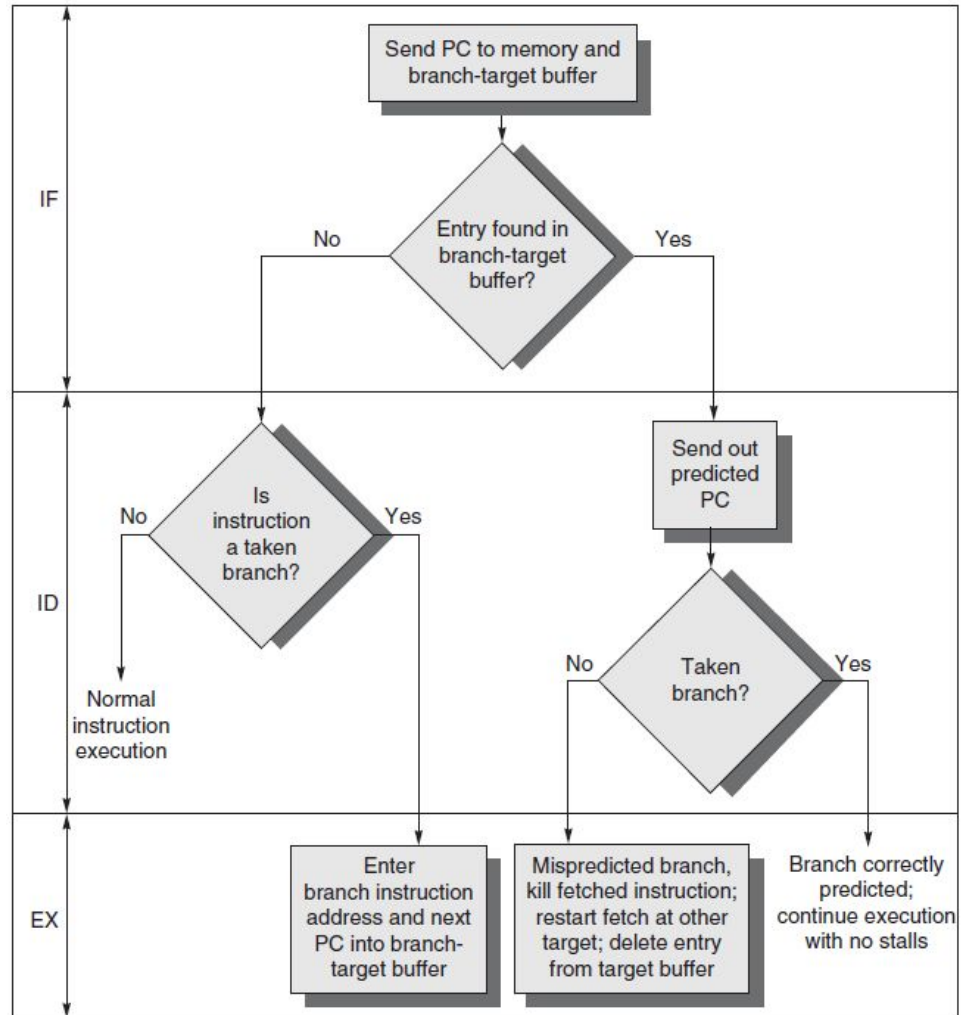


Melhorando a busca de instruções

- Branch Target-Buffer (BTB):
 - A tabela guarda o program counter (PC) da instrução e qual o próximo program counter;
 - Deve-se saber se a instrução é um desvio que será tomado ou não;
 - Se for um branch tomado então o PC será alimentado com o próximo program counter
 - Quando o endereço da instrução atual é encontrado na tabela inicia-se a execução com o PC predito;
 - Só é necessário armazenar os desvios tomados, dado que os desvios não tomados tem seu fluxo de execução sequencial;







Melhorando a busca de instruções

- **Preditor de endereço de retorno**
 - É necessário prever desvios indiretos:
 - A maior parte deles vem de retornos de procedimentos;
 - Vários locais diferentes podem chamar a mesma função
 - Endereços de retorno diferentes;
 - O uso um pequeno buffer operando como pilha para armazenar o endereço de retorno é mais interessante que BTBs
 - Chamadas de diferentes lugares podem fazer com que o buffer esqueça o endereço de retorno de chamadas passadas;
 - Tanto a família intel Core quanto a AMD Phenon usam, além de outros, esse método;



Melhorando a busca de instruções

- Projetos mais recentes empregam unidades de busca de instruções integradas;
 - Unidades que alimentam o pipeline com instruções;
 - A unidade de busca de instruções integrada tem as seguintes funções:
 - Predição de desvio;
 - Pré busca de instrução;
 - Busca instruções a frente;
 - Acesso e buffer de instrução de memória:
 - Lida com dificuldades em acesso a cache que surgem ao buscar mais de uma instrução;
 - Exemplo: acesso a mais de uma linha na cache para buscar as instruções;

Referências

