Arquitetura e Organização de Computadores II

Arquiteturas VLIW

Prof. Nilton Luiz Queiroz Jr.

Arquiteturas VLIW

- Buscam alcançar maior nível de paralelismo de instrução com a execução de instruções longas compostas por diversas operações;
- As palavras consistem de diversas operações de load/store, aritméticas e de controle na mesma palavra;
  - Por exemplo, em uma mesma palavra pode-se ter:
    - Uma operação com inteiros (que também pode ser usada para desvios);
    - Duas operações ponto flutuante;
    - Duas operações de referencia a memória;
- O tamanho da palavra está relacionado com a quantidade de operações que ea contém;

Arquiteturas VLIW

- Arquiteturas VLIW são arquiteturas que utilizam múltiplas unidades funcionais independentes;
- A emissão de instruções ocorre com o “empacotamento” de várias instruções em uma única instrução longa;
- Esse pacote de instruções é executado de maneira concorrente;

Arquiteturas VLIW

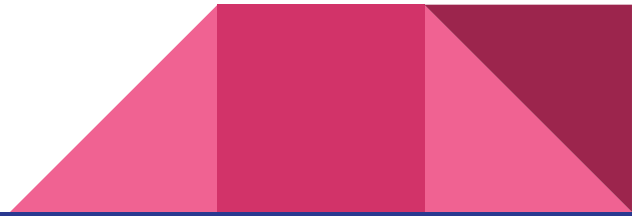
- Arquiteturas VLIW contam com grande auxilio do compilador;
  - O compilador é responsável por escalonar as instruções da maneira mais eficiente possível;
    - Inserir NO-OPs explícitas no código;
  - O compilador deve conhecer os ciclos da arquitetura;
    - Necessário para saber quantos NO-OPs, ou instruções independentes
- O código objeto não é compatível com o de um processador com arquitetura convencional;
  - Em alguns casos não é compatível entre VLIW diferentes;

Arquitetura VLIW

- O tamanho do código costuma ser muito grande;
  - Muitos NO-OPs inseridos;
  - Muito comum usar loop unrolling para melhorar performance;
- Não possuem escalonamento dinâmico;
  - Ou seja, essas arquiteturas não implementam em hardware nem o algoritmo de Tomasulo nem técnicas de especulação;
  - Hardware simplificado;
    - Não precisa de Estações de reserva;

Arquiteturas VLIW

- Alguns problemas não podem ser evitados no tempo de compilação;
  - Alguns casos a quantidade de paralelismo que pode ser extraída é limitada por situações de controle;
- Existem extensões de conjuntos de instruções que podem superar tais problemas:
  - Instruções condicionais ou com predicado (conditional instructions ou predicated instructions);



Instruções com predicado

- Instruções condicionais e pressupostas são usadas para transformar dependência de controle em dependência de dados;
  - Ao invés de usarmos um branch seguido de uma instrução, pode-se usar uma única instrução, que verifica o valor de um de seus operandos e a executa ou então a transforma em um no-op;
- Um exemplo desse tipo de instrução é o move condicional;

Instruções com predicado

- Suponha o seguinte trecho de código:

```
if (a==0){
        s=t;
}
...
```

- Qual seria seu respectivo código em assembly MIPS?

Instruções com predicado

- Suponha o seguinte trecho de código:

```
if (a==0){
        s=t;
}
...
```

- Qual seria seu respectivo código em assembly MIPS?

```
        BNEZ R1,L
        ADDU R2,R3,R0
L:
        ...
```

Instruções com predicado

- Note que no código em assembly existe uma dependência de controle;

```
        BNEZ R1,L
        ADDU R2,R3,R0
L:
        ...
```

- Com uma instrução de move condicional esse código poderia ser transformado no seguinte:

```
CMOVZ   R2, R3, R1
```

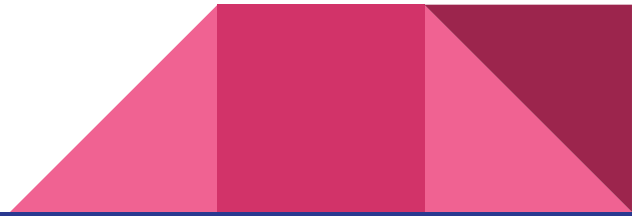
Move o valor de r3 para r2 se r1 for zero;

Instruções com predicado

- Introdução de algumas instruções condicionais, como move condicional, são chamadas de “partial predication”;
- Para usar tais instruções é necessário tomar cuidado com a quantidade de código;
  - Desvios com grande quantidade de instruções podem se tornar ineficientes;
    - Seriam necessários diversos moves condicionais, o que em alguns casos colocaria muitas instruções no-op;
- Outra alternativa é o uso de conjuntos de instruções onde todas instruções são controladas por predicados, ou ainda, “full predication”;

Blocos básicos

- Para facilitar o entendimento de arquiteturas com suporte total a predicados é interessante entender o conceito de bloco básico;
- Um bloco básico é um trecho de código onde só existe um ponto de entrada e um ponto de saída;
  - Em outras palavras, é um trecho de código onde não existem jumps, e nenhum jump pula para o meio dele;
  - É importante entender que mais de um local pode “entrar” naquele trecho e ele pode sair para mais de um local;

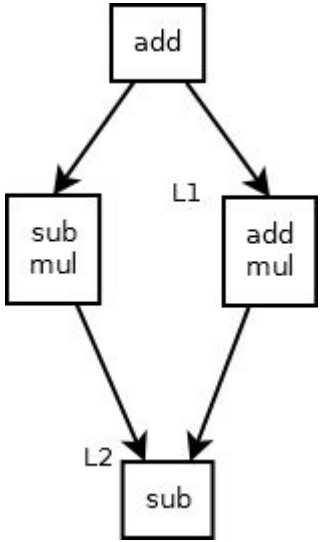


Blocos básicos

Trecho de código

```
        add   r1, r2, r3
        beqz  r1, L1
        sub   r3, r4,r2
        mul   r4, r2,r3
        j     L2
L1:     add    r1, r3,r4
        mul   r3, r5,r6
L2:     sub    r5, r3,r1
```

Blocos básicos

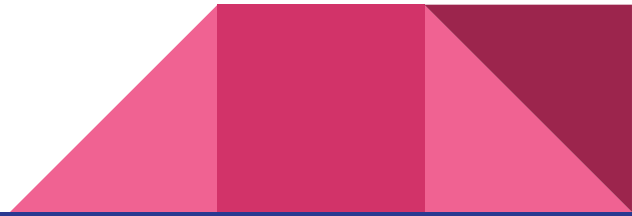


Instruções com predicado

- Uma arquitetura que tem suporte total a instruções com predicado executa instruções em paralelo, onde algumas serão efetivadas e outras não;
- Com o uso de instruções com predicado se torna possível mover instruções de maneira especulativa;
  - Instruções que podem ser muito custosas podem ser movidas acima de um branch tentando poupar tempo;
  - Instruções que estão em dois blocos básicos que são mutuamente exclusivos podem ser emitidas na mesma instrução quando existem unidades funcionais suficientes;
    - No exemplo dos blocos básicos citado anteriormente, as instruções “sub r3, r4,r2” e “add r1, r3 r4” poderiam estar no mesmo pacote de instruções se o hardware tivesse a quantidade suficiente de unidades funcionais;

Arquiteturas VLIW

- Algumas técnicas de compilação que ajudam arquiteturas VLIW são:
  - Loop Unrolling(desdobramento de loop);
  - Pipeline em software (software-pipelined);



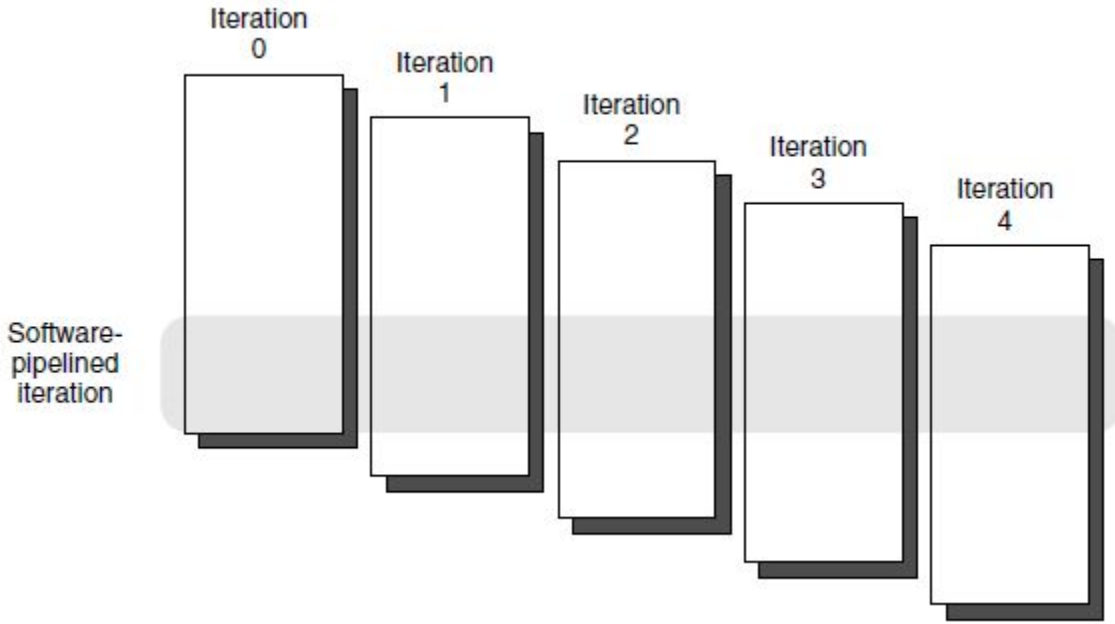
Loop Unrolling

- Aumenta a quantidade de instruções relativas a um branch;
- Multiplica o código do corpo do loop;
  - Reajusta o contador do loop e de endereço apenas uma vez por loop;

Pipeline em software

- Escalona as instruções de maneira que uma das instruções do segundo laço será enviada no mesmo pacote de alguma instrução do primeiro laço;
  - É necessário fazer loop unrolling para aplicar essa técnica
- É a técnica de software correspondente ao o algoritmo de Tomasulo;

Pipeline em software



Processador Itanium

- Um exemplo de processador VLIW-style é o Itanium
  - Itanium
  - Itanium 2
- A arquitetura é chamada de EPIC (Explicitly Parallel Instruction Computer) que é um pouco diferente das primeiras VLIWs;
  - Mais flexíveis que VLIWs comuns;
    - Usa indicadores que indicam possíveis dependências entre instruções;
      - Pode reduzir o tamanho de código;
  - Suporte mais extensivo para especulação em software;
- Conjunto de instruções IA-64;

Processadores Itanium

- Os registradores do conjunto de instruções IA-64 tem as seguintes características:
  - 128 registradores de 64 bits para uso geral;
  - 128 registradores de 64 bits para ponto flutuante;
  - 64 registradores para predicado;
  - Registradores usados para mapeamento de memória, controle do sistema, performance counters e comunicações com o SO;
- Os registradores inteiros e de ponto flutuante dão suporte a rotação de registradores;
  - Essa funcionalidade de rotação de registradores é usada para facilitar a alocação de registradores em software pipeline, pois são necessárias muitas renomeações de registradores;

Processadores Itanium

- Cada palavra consiste em 128 bits;

template	Instrução 1	Instrução 2	instrução 3

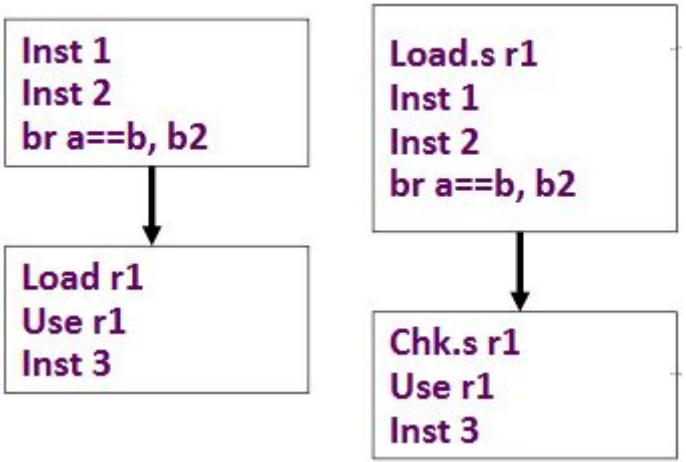
  - 5 bits de template
    - Indicam:
      - Quais as paradas estão no grupo;
      - Se pode ser executado em paralelo com o vizinho;
      - Quais unidades funcionais necessárias para cada instrução;
    - Graças ao template não é necessário colocar NO-OPs para preencher um grupo;
      - VLIWs normais fazem isso;
  - 41 bits para cada uma das instruções;

Processador Itanium

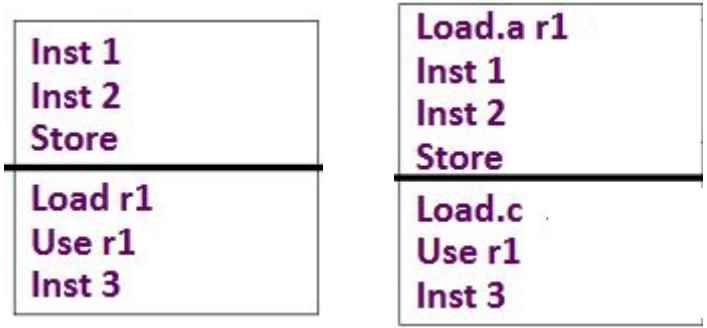
- Oferece suporte a especulação pelo software;
  - O compilador pode mover load para antes do desvio que o controla;
    - Transforma um Load em Load.s (load especulativo);
    - Em outras palavras, o load muda de bloco básico;
  - Usa uma rotina check.s para verificar se ocorreu algum problema na instrução especulativa anterior, e caso tenha, pode desviar para uma rotina para corrigir o problema;
  - A especulação de dados é feita com um load avançado;
    - Load carrega o dado e armazena numa espécie de tabela que fez load em determinado endereço;
    - Store, ao escrever, invalida todas entradas da tabela que tem aquele resultado;
    - No local que deveria ser colocado o load orignal é usado um tipo especial de load para verificar se algum store escreveu naquele endereço;

Processador Itanium

Load especulativo



Load Avançado



Referencias

HENNESSY, John L.;PATTERSON, David A. Computer architecture: a quantitative approach. Elsevier, 2011.

Wentzlaff, David. Vector, SIMD, and GPUs. Notas de Aula.

Silva, Gabriel P. Arquiteturas VLIW. Notas de Aula.