Arquitetura de Computadores

Paralelismo em Nível de Instrução

Paralelismo em Nível de Instrução

- Todos os processadores a partir de 1985 usam pipelining para sobrepor a execução de instruções e melhorar seu desempenho.
- Essa potencial sobreposição de instruções é chamada de Paralelismo em Nível de Instruções, **ILP**- do inglês- ***Instruction Level Parallelism***.

Paralelismo em Nível de Instrução

- A quantidade de paralelismo disponível dentro de um bloco básico é bem pequena.
- Um bloco básico é uma sequência de código em linha reta sem ramificações exceto para a entrada e sem ramificações exceto na saída.
- Para programas RISC típicos, a frequência média de desvio dinâmico geralmente está entre 15% e 25%, o que significa que entre três e seis instruções são executadas entre um par de desvios, geralmente essas instruções dependem uma das outras o que não permite paralelismo.
- Para obter melhorias substanciais de desempenho, devemos explorar o ILP em vários blocos básicos.

Paralelismo em Nível de Instrução

- A maneira mais simples e comum de aumentar o ILP é explorar o paralelismo entre as iterações de um laço(loop). Esse tipo de paralelismo é frequentemente chamado de **paralelismo em nível de laço**.
- Cada iteração do laço pode se sobrepor a qualquer outra iteração, embora dentro de cada iteração do laço haja pouca ou nenhuma oportunidade de sobreposição.

Paralelismo em Nível de Instrução

- Examinaremos várias técnicas para converter esse paralelismo em nível de laço em paralelismo em nível de instrução. Basicamente, essas técnicas funcionam desenrolando o laço estaticamente pelo compilador ou dinamicamente pelo hardware.

Paralelismo em Nível de Instrução

- Existe um problema ao explorar o ILP, a dependência de dados.
- As dependências são uma propriedade dos programas.
- Se uma determinada dependência resulta na detecção de um conflito e se esse conflito atualmente causa um stall são propriedades da organização do pipeline.

Paralelismo em Nível de Instrução

- Existem três tipos diferentes de dependências:
  - dependências de dados (também chamadas de dependências de dados verdadeiras),
  - dependências de nome; e
  - dependências de controle.
- Uma instrução j é dependente de dados da instrução i se qualquer um dos seguintes ocorrer:
  - A instrução i produz um resultado que pode ser usado pela instrução j.
  - A instrução j é dependente de dados da instrução k, e a instrução k é dependente de dados da instrução i.

Paralelismo em Nível de Instrução

- Por exemplo, considere a seguinte sequência de código RISC-V que incrementa um vetor de valores na memória (começando em 0(x1) terminando com o último elemento em 0(x2)) por um escalar no registrador f2.

```
Loop: fld f0,0(x1)        //carrega o 1º elemento do vetor em f0
      fadd.d f4,f0,f2     //soma o escalar em f2 a f0 e guarda em f4
      fsd f4,0(x1)        //armazena o resultado de volta no vetor
      addi x1,x1,-8       //decrementa o ponteiro em 8 bytes – próx.vetor
      bne x1,x2,Loop      //salta, se não igual x1 e x2, para loop
```

Paralelismo em Nível de Instrução

- As dependências de dados nesta sequência de código envolvem dados de ponto flutuante:

```
Loop: fld f0,0(x1)
      fadd.d f4,f0,f2     // esta instrução depende da anterior - f0
      fsd f4,0(x1)        // esta instrução depende da anterior - f4
```

- e dados inteiros:

```
      addi x1,x1,-8
      bne x1,x2,Loop  // esta instrução depende da anterior - x1
```

Paralelismo em Nível de Instrução

- Quando uma instrução é dependente de dados de outra é necessário que a ordem de execução das instruções sejam mantidas, logo não podem ser executadas simultaneamente ou serem completamente sobrepostas.
- Uma dependência de dados determina três coisas:

  (1) a possibilidade de um risco,

  (2) a ordem em que os resultados devem ser calculados e

  (3) um limite superior de quanto paralelismo pode ser explorado.

Paralelismo em Nível de Instrução

- Uma dependência de dados limita a quantidade de paralelismo em nível de instrução que podemos explorar.
- Uma dependência pode ser superada de duas maneiras diferentes:

  (1) mantendo a dependência, mas evitando um conflito, e

  (2) eliminando uma dependência transformando o código

Paralelismo em Nível de Instrução

- O **escalonamento do código** é o principal método usado para evitar um risco sem alterar uma dependência, e esse escalonamento pode ser feito tanto pelo compilador quanto pelo hardware.
- Um valor de dados pode fluir entre instruções por meio de registradores ou por locais de memória. Quando o fluxo de dados ocorre através de um registrador, a detecção da dependência é direta porque os nomes dos registradores são fixados nas instruções, embora fique mais complicado quando os ramos intervêm e as questões de correção forçam um compilador ou hardware a ser conservador.
- As dependências que fluem pelos locais de memória são mais difíceis de detectar porque dois endereços podem se referir ao mesmo local, mas parecem diferentes: Por exemplo, 100(x4) e 20(x6) podem ser endereços de memória idênticos.

Paralelismo em Nível de Instrução

- Dependência de Nomes
  - Uma dependência de nome ocorre quando duas instruções usam o mesmo registrador ou local de memória, chamado de nome, mas não há fluxo de dados entre as instruções associadas a esse nome. Existem dois tipos de dependências de nomes entre uma instrução i que precede a instrução j na ordem do programa:
    - Uma **antidependência** entre a instrução i e a instrução j ocorre quando a instrução j escreve um registrador ou local de memória que a instrução i lê. A ordenação original deve ser preservada para garantir que i leia o valor correto
    - Uma **dependência de saída** ocorre quando a instrução i e a instrução j escrevem no mesmo registrador ou local de memória. A ordenação entre as instruções deve ser preservada para garantir que o valor finalmente escrito corresponda à instrução j.

Paralelismo em Nível de Instrução

- Antidependências e dependências de saída são dependências de nome, ao contrário de dependências de dados verdadeiras, porque não há valor sendo transmitido entre as instruções.
- Logo podem ser executadas simultaneamente ou em ordem trocada se o nome usado for trocado para evitar conflitos.

Paralelismo em Nível de Instrução

- Conflito de Dados
  - Existe um conflito sempre que há uma dependência de nome ou dados entre as instruções, e elas estão próximas o suficiente para que a sobreposição durante a execução altere a ordem de acesso ao operando envolvido na dependência.
  - Por causa da dependência, devemos preservar o que é chamado de **ordem do programa** - isto é, a ordem em que as instruções seriam executadas se executadas sequencialmente uma de cada vez, conforme determinado pelo programa fonte original.

Paralelismo em Nível de Instrução

- Os conflitos de dados podem ser classificados em três tipos, dependendo da ordem dos acessos de leitura e gravação nas instruções:
- Considere duas instruções i e j, com i precedendo j na ordem do programa. Os possíveis riscos de dados são:
  - RAW (read after write)—j tenta ler uma fonte antes de i escrevê-la, então j incorretamente obtém o valor antigo. Este conflito é o tipo mais comum e corresponde a uma verdadeira dependência de dados. A ordem do programa deve ser preservada para garantir que j receba o valor de i.

Paralelismo em Nível de Instrução

- WAW (write after write)—j tenta escrever um operando antes de ser escrito por i. As escritas acabam sendo realizadas na ordem errada, deixando o valor escrito por i ao invés do valor escrito por j no destino. Este perigo corresponde a uma dependência de saída.
- WAR (write after read)—j tenta escrever um destino antes de ser lido por i, então i obtém incorretamente o novo valor. Este perigo surge de uma antidependência (ou dependência do nome).