# 计算机系统结构

## 第五讲 指令调度与循环展开

谢长生

武汉光电国家研究中心

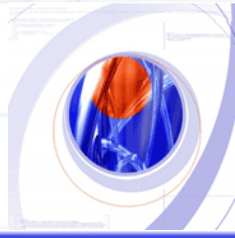
华中科技大学
HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

# 1.开发ILP的两种方法

- 基于硬件的动态开发方法
- 基于软件的静态开发方法

# 2. 指令调度的基本方法

- 指令调度
  - 找出不相关的指令序列，让它们在流水线上重叠并行执行。
- 制约编译器指令调度的因素
  - 程序固有的指令级并行
  - 流水线功能部件的延迟



# 3. 部件延迟

| 产生结果的指令 | 使用结果的指令 | 延迟(cycles) |
| --- | --- | --- |
| 浮点计算 | 另一个浮点计算 | 3 |
| 浮点计算 | 浮点store(S.D) | 2 |
| 浮点Load(L.D) | 浮点计算 | 1 |
| 浮点Load(L.D) | 浮点store(S.D) | 0 |

本节使用的浮点流水线的延迟

# 4. 指令调度实例

例1 对于下面的源代码，转换成MIPS汇编语言，在不进行指令调度和进行指令调度两种情况下，分析其代码一次循环所需的执行时间。

```
for (i=1000; i>0; i--)
    x[i] = x[i] + s;
```

**解：**

**先把该程序翻译成MIPS汇编语言代码**

```
Loop:   L.D      F0, 0(R1)
        ADD.D    F4, F0, F2
        S.D      F4, 0(R1)
        DADDIU R1, R1, #-8
        BNE      R1, R2, Loop
```

# 5. 延迟分析

– 在不进行指令调度的情况下，根据表中给出的浮点流水线中指令执行的延迟，程序的实际执行情况如下：

| | | | 指令流出时钟 |
|---|---|---|---|
| Loop： | L.D | F0, 0(R1) | 1 |
| | (空转) | | 2 |
| | ADD.D | F4, F0, F2 | 3 |
| | (空转) | | 4 |
| | (空转) | | 5 |
| | S.D | F4, 0(R1) | 6 |
| | DADDIU | R1, R1, #-8 | 7 |
| | (空转) | | 8 |
| | BNE | R1, R2, Loop | 9 |
| | (空转) | | 10 |

# 6. 仅使用指令调度优化

– 在用编译器对上述程序进行指令调度以后，程序的执行情况如下：

| | | | 指令流出时钟 |
|---|---|---|---|
| Loop： | L.D | F0, 0(R1) | 1 |
| | DADDIU | R1, R1, #-8 | 2 |
| | ADD.D | F4, F0, F2 | 3 |
| | (空转) | | 4 |
| | BNE | R1, R2, Loop | 5 |
| | S.D | F4, 8(R1) | 6 |

# 7. 进一步的优化：循环展开

- 编译时指令调度是怎样减少整个指令序列在流水线上的执行时间的？
- 指令调度能否跨越分支边界？
- 怎样提高整个执行过程中有效操作的比率？

# 8. 循环展开的基本概念与方法

- 循环展开
  - 把循环体的代码复制多次并按顺序排放， 然后相应调整循环的结束条件。
  - 开发循环级并行的有效方法

**例2 将例1中的循环展开3次得到4个循环体，然后对展开后的指令序列在不调度和调度两种情况下，分析代码的性能。假定R1的初值为32的倍数，即循环次数为4的倍数。消除冗余的指令，并且不要重复使用寄存器。**

# 9. 展开后的代码

- 展开后没有调度的代码如下(需要分配寄存器)

| | | | 指令流出时钟 |
|---|---|---|---|
| Loop: | L.D | F0, 0(R1) | 1 |
| | （空转） | | 2 |
| | ADD.D | F4, F0, F2 | 3 |
| | （空转） | | 4 |
| | （空转） | | 5 |
| | S.D | F4, 0(R1) | 6 |
| | L.D | F6, -8(R1) | 7 |
| | （空转） | | 8 |
| | ADD.D | F8, F6, F2 | 9 |
| | （空转） | | 10 |
| | （空转） | | 11 |
| | S.D | F8, -8(R1) | 12 |
| | L.D | F10, -16(R1) | 13 |
| | （空转） | | 14 |

| | | | 指令流出时钟 |
|---|---|---|---|
| | ADD.D | F12, F10, F2 | 15 |
| | （空转） | | 16 |
| | （空转） | | 17 |
| | S.D | F12, -16（R1） | 18 |
| | L.D | F14, -24（R1） | 19 |
| | （空转） | | 20 |
| | ADD.D | F16, F14, F2 | 21 |
| | （空转） | | 22 |
| | （空转） | | 23 |
| | S.D | F16, -24（R1） | 24 |
| | DADDIUR1, R1, # -32 | | 25 |
| | （空转） | | 26 |
| | BNE | R1, R2, Loop | 27 |
| | （空转） | | 28 |

**50%是空转周期！**

# 10. 展开且再次调度后的代码

- 调度后的代码如下：

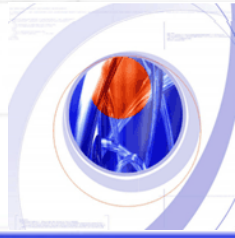
| | | | 指令流出时钟 |
|---|---|---|---|
| Loop: | L.D | F0, 0（R1） | 1 |
| | L.D | F6, -8（R1） | 2 |
| | L.D | F10, -16（R1） | 3 |
| | L.D | F14, -24（R1） | 4 |
| | ADD.D | F4, F0, F2 | 5 |
| | ADD.D | F8, F6, F2 | 6 |
| | ADD.D | F12, F10, F2 | 7 |
| | ADD.D | F16, F14, F2 | 8 |
| | S.D | F4, 0（R1） | 9 |
| | S.D | F8, -8（R1） | 10 |
| | DADDIU | R1, R1, # -32 | 12 |
| | S.D | F12, 16（R1） | 11 |
| | BNE | R1, R2, Loop | 13 |
| | S.D | F16, 8（R1） | 14 |

**结论：通过循环展开、寄存器重命名和指令调度，可以有效开发出指令级并行。**

没有空转周期！

# 11. 循环展开和指令调度的注意事项

- 保证正确性
- 注意有效性
- 使用不同的寄存器
- 删除多余的测试指令和分支指令，并对循环结束代码和新的循环体代码进行相应的修正。
- 注意对存储器数据的相关性分析
- 注意新的相关性



谢谢大家