5.优化程序性能

5.1 优化编译器的能力和局限性

大多数编译器，包括GCC，向用户提供了一些对他们所使用的优化的控制。编译器必须很小心地对程序只使用安全的优化。
比如，两个指针有可能指向同一个内存位置的情况（内存别名使用)在优化时就会导致问题。如果编译器不能确定两个指针是否指向同一个位置，就必须假设什么情况都有可能，就限制了可能的优化策略。

5.2 表示程序性能

引入每元素周期数（CPE）来表示程序性能。

5.3 消除循环的低效率

将循环不变值外提。

5.5 减少过程调用

即减少函数调用。

5.6 消除不必要的内存引用

（也注意到类似内存别名使用时造成的影响）
使用中间变量同时写回，就可以避免别名错误。

5.7 理解现代处理器

利用处理器微体系结构的优化，也就是处理器用来执行指令的底层系统设计。

5.7.1 整体操作

在每个指令周期执行多个操作，且是乱序的。

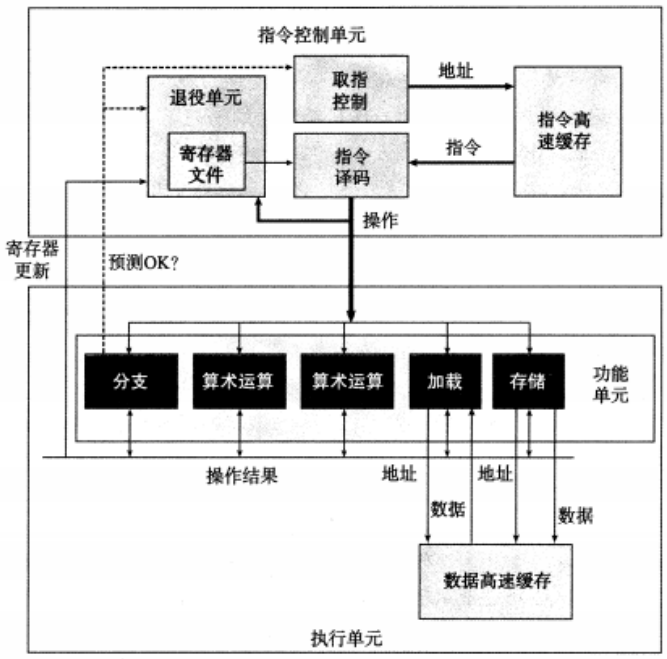


图 5-11　一个乱序处理器的框图。指令控制单元负责从内存中读出指令，并产生一系列基本操作。然后执行单元完成这些操作，以及指出分支预测是否正确

指令完成后结果先存在退役单元中，直到确定生效后再执行所有对程序寄存器的更新。如果预测错误，则这条指令会被清空。

为了加速一条指令到另一条指令的结果的传送，许多此类信息在执行单元之间交换，执行单元可以直接将结果发送给彼此。

控制操作数在执行单元间传送的最常见的机制称为寄存器重命名。当一条更新寄存器r的指令译码时，产生标记t，得到一个指向该操作结果的唯一的标识符。条目(r,t)被加入到一张表中，该表维护着每个程序寄存器r与会更新该寄存器的操作的标记t之间的关系。当随后以寄存器r作为操作数的指令译码时，发送到执行单元的操作会包含t作为操作数源的值。当某个执行单元完成第一个操作时，会生成一个结果（v，t），指明标记为t的操作产生值v。当一条被译码的指令需要寄存器r，而又没有标记与这个寄存器相关联，那么可以直接从寄存器文件中获取这个操作数。

5.7.2 功能单元的性能

运算	整数			浮点数		
	延迟	发射	容量	延迟	发射	容量
加法	1	1	4	3	1	1
乘法	3	1	1	5	1	2
除法	3 ~ 30	3 ~ 30	1	3 ~ 15	3 ~ 15	1

延迟是表示完成该运算需要的最小时钟周期数；发射是两个连续的同类型的运算之间需要的最小时钟周期数；容量表示能够执行该运算的功能单元的数量。

（延迟并不影响发射时间）

关键路径才是影响程序运行时间的部分（这次迭代需要用到上一次迭代计算出来的值）

5.8 循环展开

```
/* Combine 2 elements at a time */
for (i = 0; i < limit; i+=2) {
    acc = (acc OP data[i]) OP data[i+1];
}
```

增加每次迭代计算的元素的数量，减少循环的迭代次数。

首先，它减少了不直接有助于程序结果的操作的数量（即非关键路径中的操作），第二，它提供了一些方法，可以进一步变化代码，减少整个计算中关键路径上的操作数量。

5.9 提高并行性

5.9.1 多个累计变量

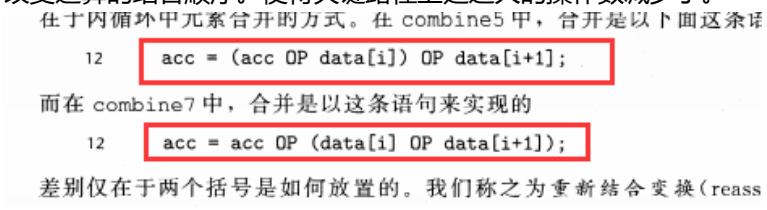
```
/* Combine 2 elements at a time */
for (i = 0; i < limit; i+=2) {
    acc0 = acc0 OP data[i];
    acc1 = acc1 OP data[i+1];
}

/* Finish any remaining elements */
for (; i < length; i++) {
    acc0 = acc0 OP data[i];
}
```

可以通过将同一组合运算分割成两个或更多的部分，并在最后合并结果来提高性能。
通常，只有保持能够执行该操作的所有功能单元的流水线都是满的，程序才能达到这个操作的吞吐量界限。对延迟为L，容量为C的操作而言，要求循环展开因子k>=C X L。
浮点乘法和加法是不可结合的。

5.9.2 重新结合变换

改变运算的结合顺序。使得关键路径上延迟大的操作数减少了。



5.11一些限制因素

关键路径上的所有延迟之和等于T，则这个程序最少需要T个周期才能执行完。
如果一个程序需要N个某种运算的计算，而微处理器之后C个能执行这个操作的功能单元，并且这些单元的发射时间为I，则这个程序最少需要N X I / C个周期

5.11.1 寄存器溢出

如果并行度P超过了可用的寄存器数量，就会将临时变量存储在内存中，通常是在运行时堆栈上分配空间。

5.11.2 分支预测和预测错误处罚

5.12 理解内存性能

5.12.1 加载的性能

只有两个加载单元

5.12.2 存储的性能

存储的操作。可能会有写读相关，就会延迟