处理器体系结构

第四章 处理器的微架构B

--流水线基础

(Micro-processor Architecture)

目录

4.2.1 流水线简介

一个在死后被尊称为“科学管理之父”的人；

一个影响了流水线生产方式产生的人；

一个被社会主义伟大导师列宁推崇备至的人；

一个影响了人类工业化进程的人。

一个由于视力被迫辍学的人；

一个被工人称为野兽般残忍的人；

一个与工会水火不容，被迫在国会上作证的人；

一个被现代管理学者不断批判的人。

Frederick Winslow Taylor, 1856—1915

4.2.1 流水线简介

流水线是如何优化效率的?

流水线能够提升多少的效率?

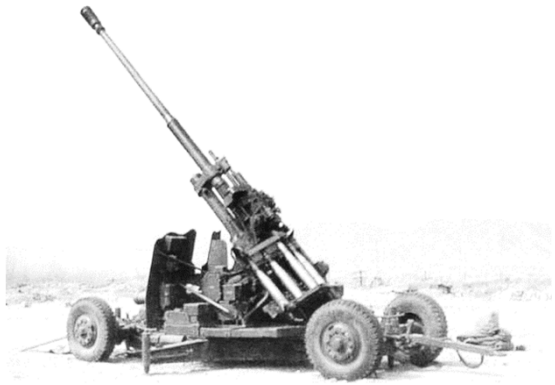
流水线的级数对效率的影响?

如何提升流水线的效率?

4.2.1 流水线简介

如何提高高射炮的射击速度?

- **填装炮弹：10秒**
- **瞄准射击：10秒**



4.2.1 流水线简介

如何提高高射炮的射击速度?

- **填装炮弹：5+5秒**
- **瞄准射击：5+5秒**

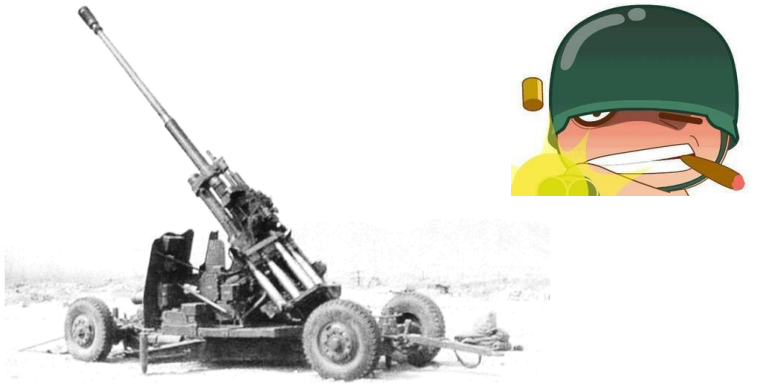
双管高炮

每分钟12发

4.2.1 流水线简介

如何提高高射炮的射击速度?

- **填装炮弹：10秒**
- **瞄准射击：5秒**

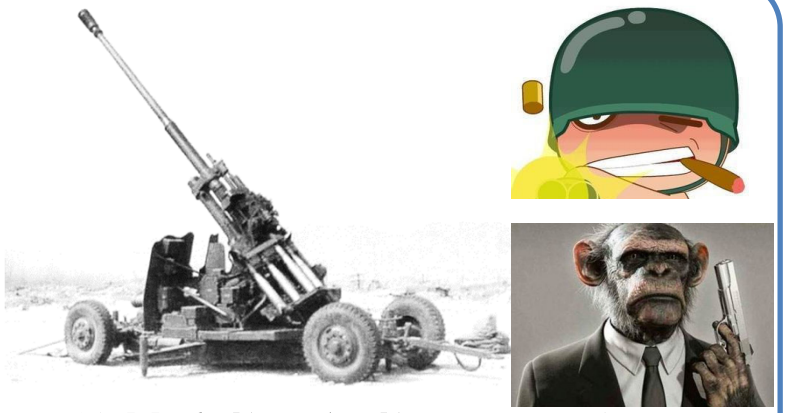


单管高炮+神炮手

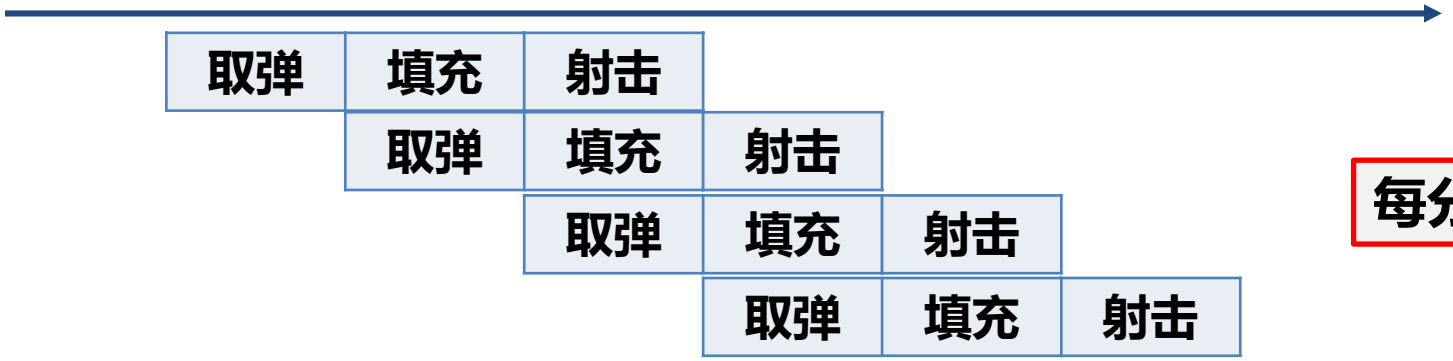
4.2.1 流水线简介

如何提高高射炮的射击速度?

- **填装炮弹：10秒 =取弹(5s)+填充(5s)**
- **瞄准射击：5秒**



单管高炮+神炮手+程序猿

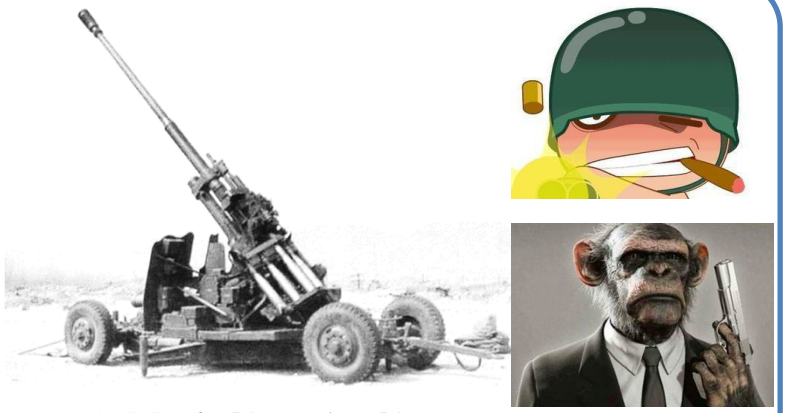


每分钟12发

4.2.1 流水线简介

如何提高高射炮的射击速度?

- **填装炮弹：10秒 ->自动填装器×2!**
- **瞄准射击：5秒**



单管高炮+神炮手+程序猿

每分钟12发

目录

4.2.2 MIPS中的流水线

最基本的3级流水线：

1. IF：从程序存储器中取出指令
2. ID：指令译码、寄存器读
3. EX：执行操作或计算地址

- 低功耗领域的ARM7采用3级流水

最经典的五级流水线：

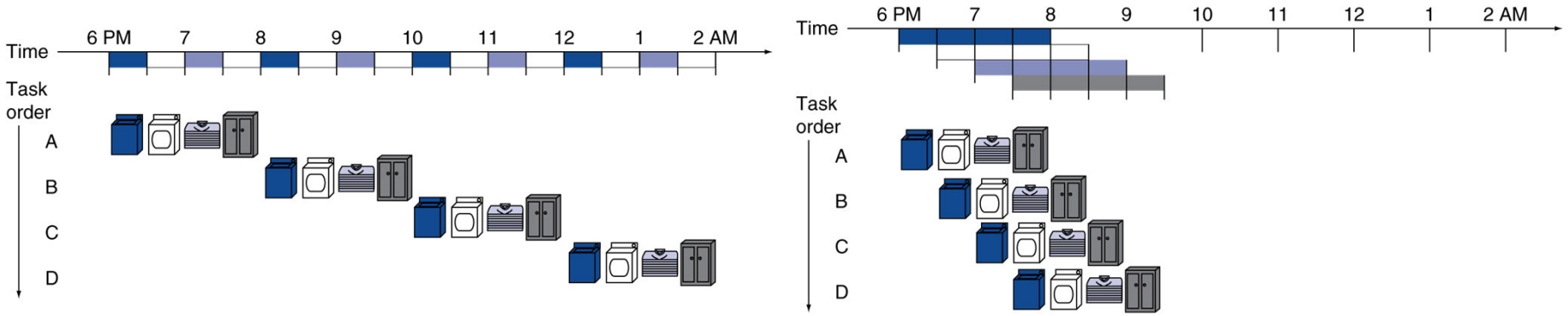
1. IF：从程序存储器中取出指令
2. ID：指令译码、寄存器读
3. EX：执行操作或计算地址
4. MEM：访问数据存储器
5. WB：将结果写回到寄存器

- 早期的MIPS、ARM9采用这种流水线，后来的处理器也能看到它的影子，包括x86处理器

->流水线能够提升多少效率?

4.2.2 MIPS中的流水线--流水线带来的加速

四级流水线的加速比?



执行4次的加速比：8/3.5=2.3

执行n次的加速比：2n/(0.5n+1.5)≈4

如果每一步操作的执行时间都相等，则使用m级流水，执行n次的加速比为：

n:[(n+m-1)/m]≈m

->每一步的执行时间不等怎么办?

4.2.2 MIPS中的流水线--流水线带来的加速

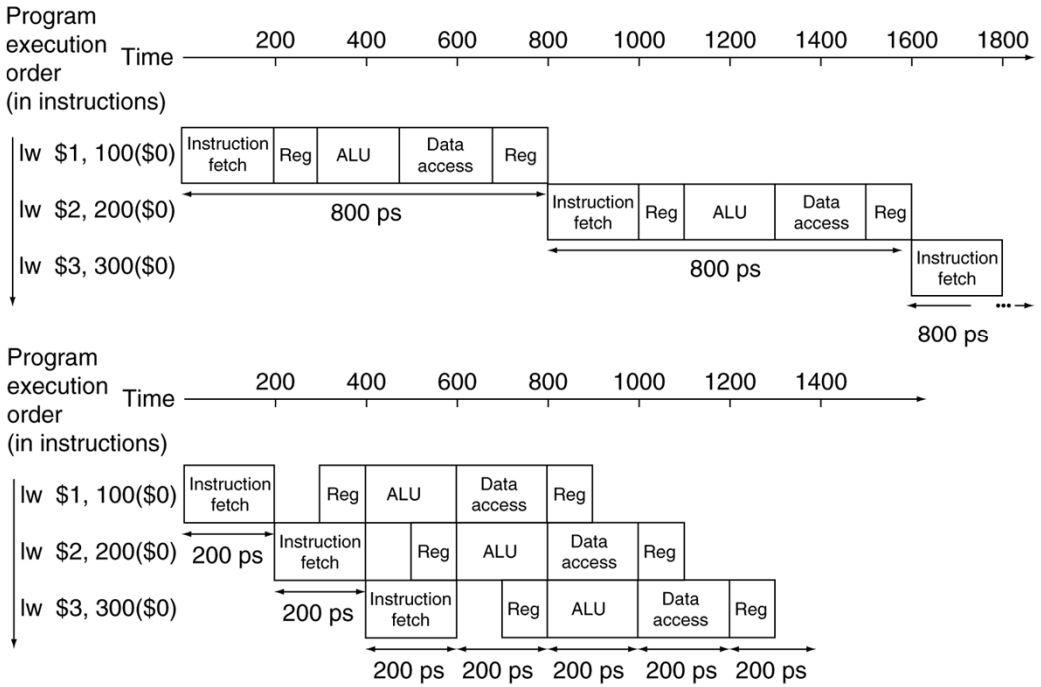
思考：对于经典五级流水线的处理器，假设ID和WB阶段各需要100ps，其它阶段需要200ps，则指令的执行需要多长时间？(与单周期处理器对比)

指令	IF	ID	EX	MEM	WB	单周期处理器	流水线处理器
lw	200ps	100 ps	200ps	200ps	100ps	**800ps**	**1000ps**
sw	200ps	100 ps	200ps	200ps		**700ps**	**1000ps**
R-format	200ps	100 ps	200ps		100ps	**600ps**	**1000ps**
beq	200ps	100 ps	200ps			**500ps**	**1000ps**

->对于单条指令，流水线执行实际上需要更长的时间！

4.2.2 MIPS中的流水线--流水线带来的加速

执行3次取数指令，则单周期和流水线版本的处理器各需多少执行时间?



单周期处理器：
2400ps

流水线处理器：
1400ps

->连续执行多条指令时，流水线的速度优势才能体现

执行n次取数指令，则单周期和流水线版本的处理器各需多少执行时间?

单周期处理器：800n ps

流水线处理器：200n+800 ps

4.2.2 MIPS中的流水线--流水线与ISA设计

为提升流水线处理器的加速比，需要：指令每个阶段所需的执行时间尽可能相等

- 流水线的级数需要合理，有时候需要适当的折中

针对流水线优化，MIPS ISA进行了以下设计：

1. 所有指令都是32位的
   - 易于取指和译码(IF、ID)
   - ps：x86每条指令的长度为1~17个字
2. 较少和规整的指令格式
   - 可以一个周期内完成译码和读寄存器(否则需要多一级流水)
3. 存取数的地址计算
   - 在EX阶段计算数据访问的地址(否则需要额外的地址计算阶段)
4. 存储器数据对齐(4的倍数)
   - 只需一个周期完成存储器访问

执行速度(流水线加速比、主频)、指令数量、控制复杂度、面积、功耗等因素的折中

目录

4.2.3 冒险 (Hazards)

冒险：处理器在下一个时钟周期不能执行下一条指令的情况

高射炮在打坦克

- **结构冒险：所需要的资源正在被使用**

新炮弹还没造好

- **数据冒险：需要等待先前指令完成其数据的读/写**

一群灰机飘过，不知道打哪个了！

- **控制冒险：控制的决策依赖于先前指令的执行结果**

4.2.3 冒险 (Hazards)--结构冒险

产生原因：资源使用的冲突

It means that the hardware cannot support the combination of instructions that we want to execute in the same clock cycle.

MIPS指令集在设计的时候就考虑了如何流水，如何有效避免结构冒险

假设：数据存储器和指令存储器是在一起，那么存取数的时候就不能取指，从而在IF阶段产生冒险

->需要分离的程序/数据存储

*Hazard、Stall、bubble的关系?

流水线出现了Hazard，怎么办？

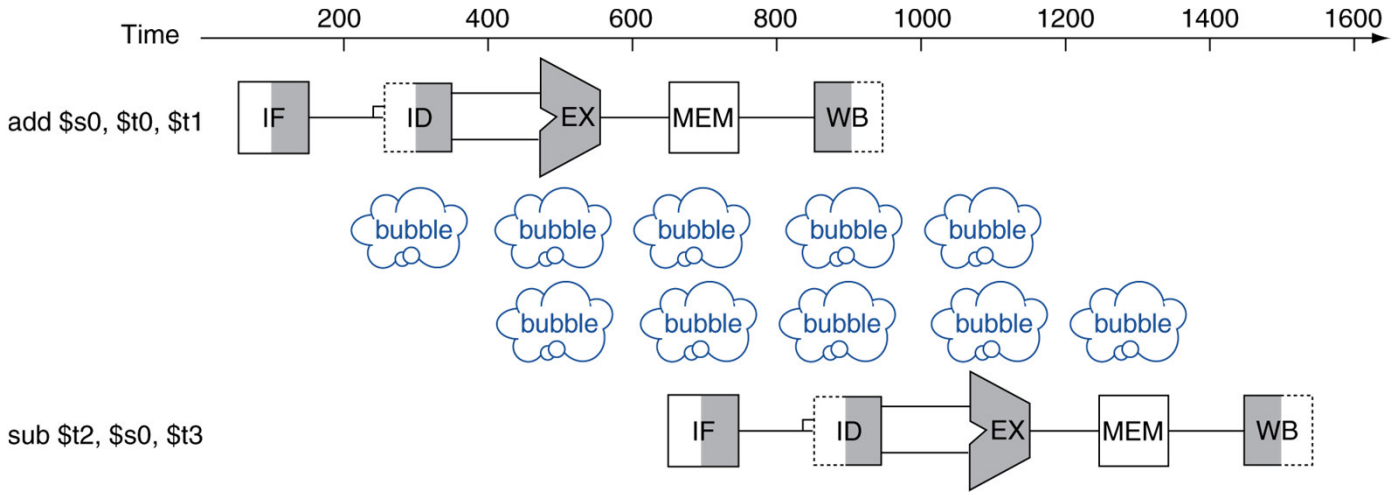
引入stall(pipeline stall)，停顿一会儿，解决hazard

Bubble：pipeline stall的nickname

4.2.3 冒险 (Hazards)--数据冒险

产生原因：指令的执行依赖于先前指令的执行结果

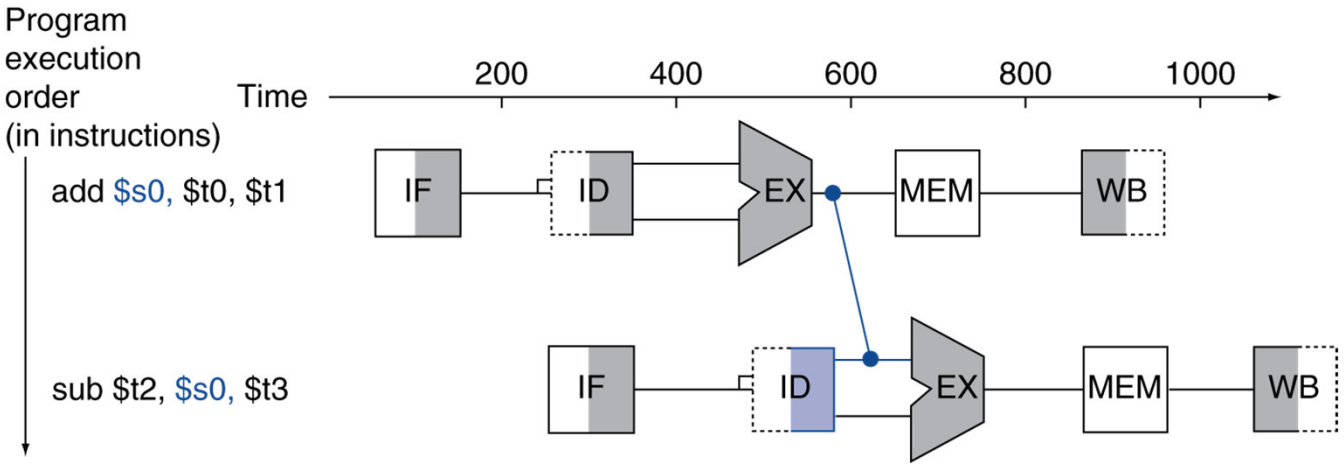
比如： `add $s0, $t0, $t1`
`sub $t2, $s0, $t3`



如何消除？： 在EX执行完毕（而非等待WB完毕）使用数据

4.2.3 冒险 (Hazards)--通过旁路解决数据冒险

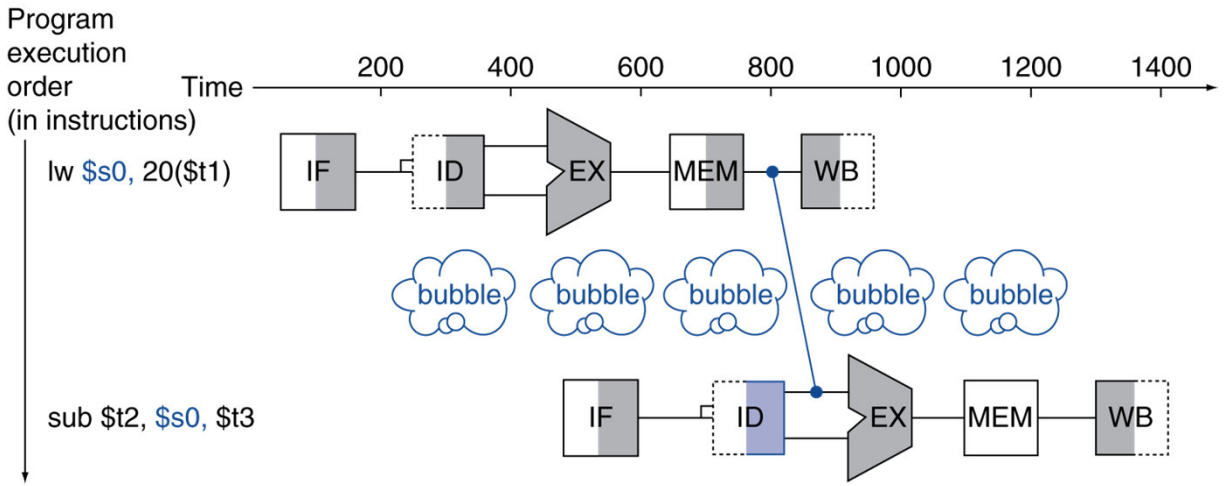
旁路： 在数据通路中加入额外的连接



思考：如果所需要的数据来自MEM阶段呢?

4.2.3 冒险 (Hazards)--“取数-使用”型数据冒险

即使是使用旁路，也不能避免产生bubble：



例：对于下列程序，使用旁路和不使用旁路执行，分别需要多少周期？

```
lw   $t1, 0($t0)
lw   $t2, 4($t0)
add  $t3, $t1, $t2
sw   $t3, 12($t0)
lw   $t4, 8($t0)
add  $t5, $t1, $t4
sw   $t5, 16($t0)
```

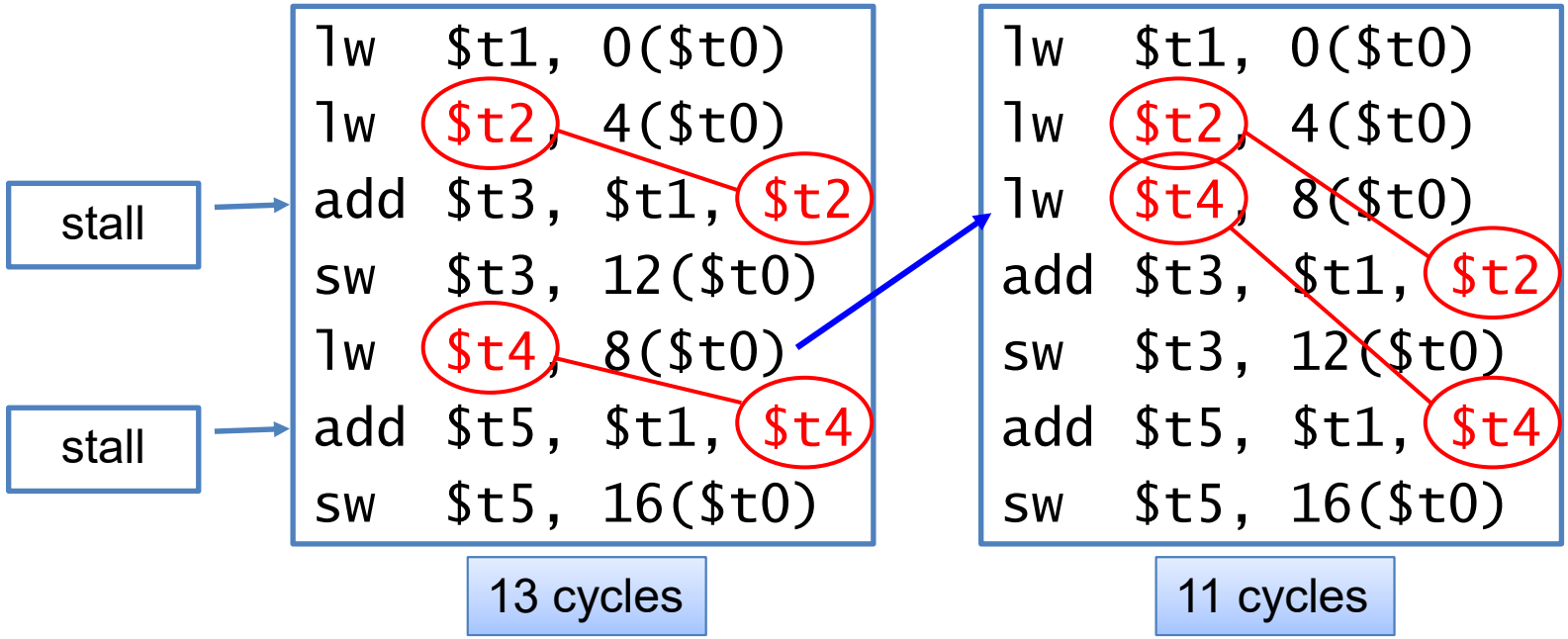
不使用旁路：19 cycles

使用旁路：13 cycles

->能否更快?

4.2.3 冒险 (Hazards)--通过指令排序避免“读取-使用”型冒险

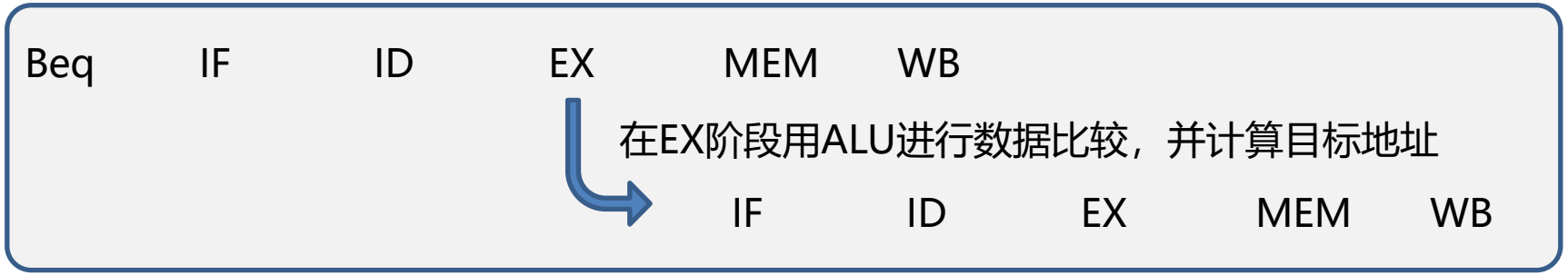
重新排序指令，用于避免“读取-使用”型冒险



4.2.3 冒险 (Hazards)--控制冒险

也叫作分支冒险，即决策导致的冒险：

- **下一条指令的取指依赖于分支结果**
- **流水线不能保证总能取到正确的指令(PC+4)**

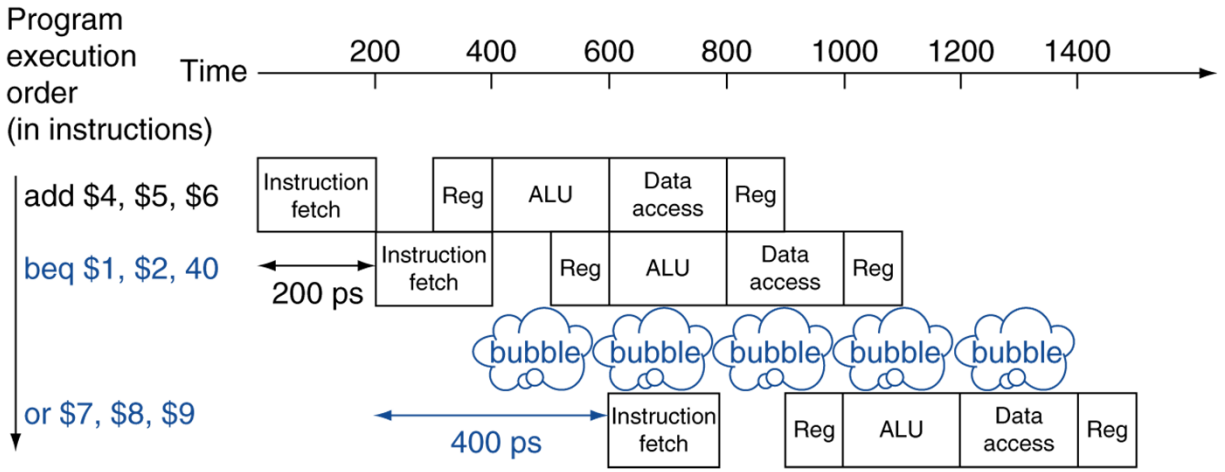


解决方案：

- **需要在流水线中尽早完成决策**
- **在ID阶段增加硬件(计算分支地址、决策是否分支)**

4.2.3 冒险 (Hazards)--控制冒险

即便在ID阶段决策并计算地址，也不能避免冒险：



如果流水线长，则无法较早确立分支结果

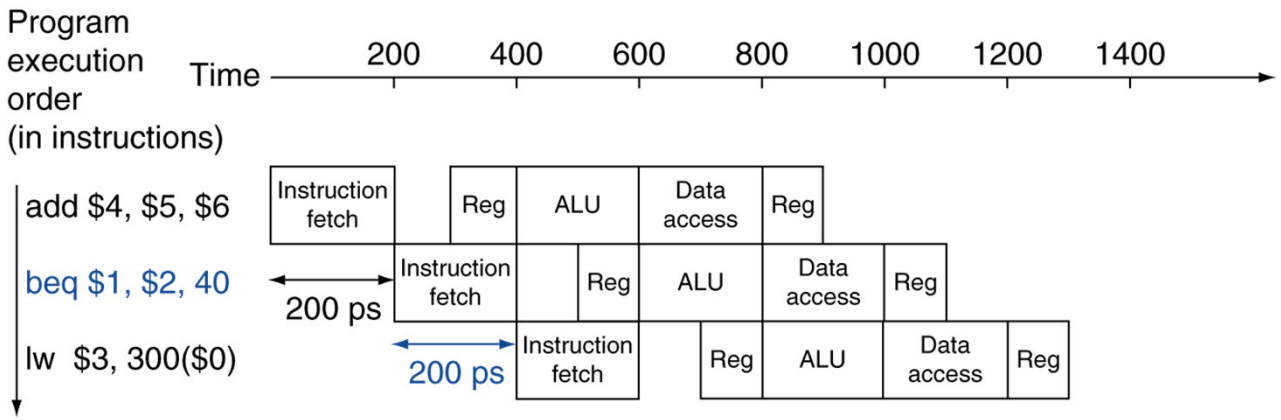
->将产生较多的阻塞

预测分支的结果：只有预测错误时，才会产生阻塞

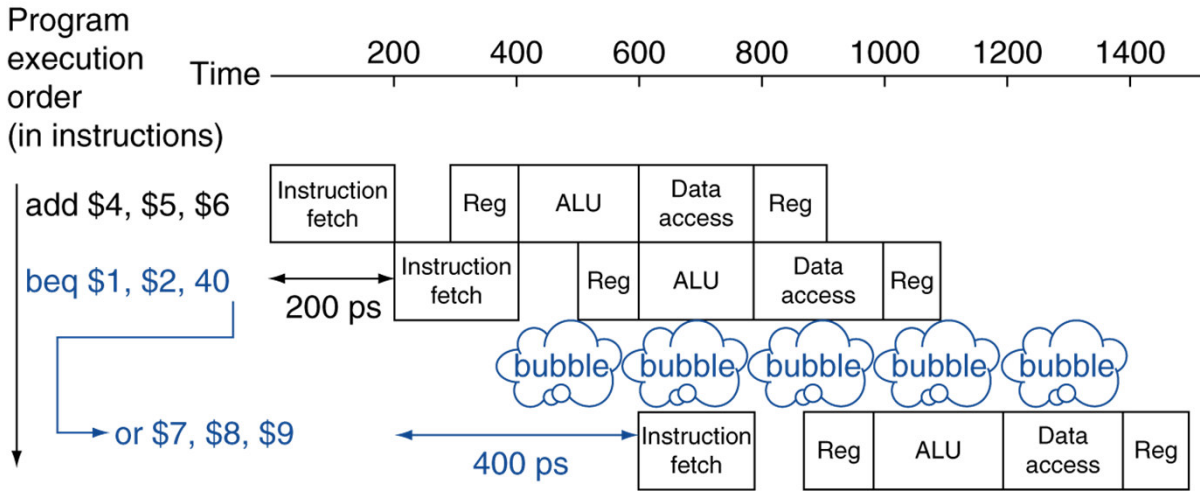
- **预测分支发生/不发生**
- **在分支指令后立马开始新的取指**

4.2.3 冒险 (Hazards)--分支预测

预测正确：



预测错误：



4.2.3 冒险 (Hazards)--更加可行的分支预测

静态分支预测

- **基于典型分支行为**
- **例如：对于循环语句中的判定，预测发生向前跳转**

动态分支预测

- **硬件检测实际的分支行为，对最近的分支历史进行记录**
- **假设分支保持原先的趋势，如果错误，则阻塞并重新取指，更新历史数据**

MIPS还常常采用延迟分支

- **不论是否发生分支，都继续执行下一条指令(与分支无关的指令)**

  ie：由编译器在beq后面增加一条与分支无关的指令，这样的话，不论是否分支，

  这一条指令的执行一定是有用的

小结

理想情况下，x级流水，执行n次的加速比为：(相比单周期处理器)

$$n:[(n+x-1)/x]$$

实际影响流水线加速比的因素：

流水线级数、流水线的划分 、阻塞

冒险：结构冒险、数据冒险、控制冒险

MIPS指令集针对流水线的优化：

1. 所有指令都是32位的
2. 较少和规整的指令格式
3. 存取数的地址计算
4. 存储器数据对齐(4的倍数)