第三章　流水线技术

一、流水线的基本概念及分类：

静（动）态流水线、单（多）流水线、（非）线性流水线

单功能流水线：只能完成一种固定功能的流水线。

多功能流水线：流水线的各段可以进行不同的连接，以实现不同的功能。

静态流水线：在同一时间内，多功能流水线中的各段只能按同一种功能的连接方式工作。

动态流水线：在同一时间内，多功能流水线中的各段可以按照不同的方式连接，同时执行多种功能。

线性流水线：流水线的各段串行连接，没有反馈回路。数据通过流水线中的各段时，每一个段最多只流过一次。

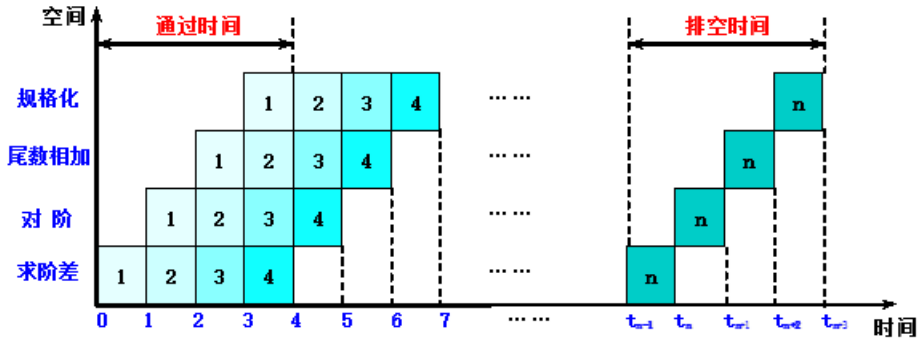
非线性流水线：流水线中除了有串行的连接外，还有反馈回路。

流水线技术核心:部件功能专用化.

时间最长的段为流水线的瓶颈。

二、流水线表示

时空图 、连接图



通过时间：第一个任务从进入流水线到流出结果所需的时间。又称装入时间。

排空时间：最后一个任务从进入流水线到流出结果所需的时间。

流水线性能计算和分析 ： 吞吐率、加速比、效率

1、吞吐率

1 吞吐率

定义：单位时间内流水线所完成的务数量或输出结果的数量。

$$TP = \frac{n}{T_k}$$

吞吐率 $TP = \frac{n}{T_k}$

n：任务数

T_k：处理完成n个任务所用的时间

n：任务数

T_k：处理完n个任务所用时间

流水线完成 n 个连续任务所需要的总时间（假设一条 k 段线性流水线）

$T_k = k\Delta t + (n-1)\Delta t = (k+n-1)\Delta t$

所以：$k\Delta t + (n-1)\Delta t = (k+n-1)\Delta t$

k段线性流水线，n为任务数

- 流水线的实际吞吐率

$$TP = \frac{n}{(k+n-1)\Delta t}$$

- 最大吞吐率

$$TP_{max} = \lim_{n\to\infty} \frac{n}{(k+n-1)\Delta t} = \frac{1}{\Delta t}$$

当流水线各段执行时间不等时，流水线实际吞吐率：

分母中第一部分是流水线完成第 1 个任务所用时间，第二部分是完成其余 n－1 个任务所用时间。

执行时间长的一个流水段成为整个流水线的瓶颈。

两种解决流水线瓶颈的方法：

解决瓶颈的方法

① 把瓶颈段细分为多个子流水段。

② 重复设置瓶颈段，让多个瓶颈流水段并行工作

重复设置瓶颈段，使多个瓶颈并行

2、加速比

加速比：完成同样一批任务，不使用流水线所用的时间与使用流水线所用的时间之比。

$$S = \frac{T_s}{T_k}$$

$S = \frac{T_s}{T_k}$.

T_s：不使用流水线（顺序执行）所用的时间

T_k ：使用流水线后所用的时间

Ts：不使用流水.

Tk：使用流水.

- 流水线各段时间相等（都是△t）
  - 一条k段流水线完成n个连续任务的时间：

    $T_k = (k+n-1)\Delta t$
  - 顺序执行n个任务所需要的时间：

    $T_s = nk\Delta t$

实际加速比：

$$S = \frac{n \cdot k\Delta t}{(k+n-1)\Delta t} = \frac{n \cdot k}{k+n-1}$$

$S = \frac{n \cdot k \cdot \Delta t}{(n-1+k)\Delta t} = \frac{nk}{n-1+k}$

- 最大加速比

  n=∞时，最大加速比

$$S_{max} = \lim_{n\to\infty} \frac{n \cdot k}{k+n-1} = k$$

（流水线段数）

当 $n=\infty$ 时，$S_{max} = k$.

3、流水线效率

流水线效率：流水线中的设备实际使用时间与整个运行时间的比值，又称流水线设备利用率。

流水线有通过时间和排空时间，在连续完成 n 个任务的时间内，各段并不是满负荷地工作。

流水线效率.

各段时间相等

- 各段的效率e_i相同

$$e_1 = e_2 = \cdots = e_k = \frac{n\Delta t}{T_k} = \frac{n}{k+n-1}$$

$E = \frac{\text{实际时间}}{\text{总时间}}$.

➢ 整条流水线的效率为：

$$E=\frac{e_1+e_2+\cdots+e_k}{k}=\frac{ke_1}{k}=\frac{kn\Delta t}{kT_k}$$

➢ 可以写成：

$$E=\frac{n}{k+n-1}$$

➢ 最高效率为：

$$E_{\max}=\lim_{n\to\infty}\frac{n}{k+n-1}=1$$

当n>>k时，E≈1。

- 流水线各段执行时间均相等，流水线极限效率

$$E=\frac{k\cdot n\Delta t}{k(k+n-1)\Delta t}=\frac{n}{k+n-1}$$

当n>>k时，流水线的效率接近最大值1。

- 流水线效率与吞吐率关系

$$E=P\cdot\Delta t$$

$$P=\frac{n}{(k+n-1)\Delta t}$$

- 流水线效率与加速比关系

$$E=\frac{S}{k}$$

$$S=\frac{n\cdot k\Delta t}{(k+n-1)\Delta t}=\frac{n\cdot k}{k+n-1}$$

- 效率E＝1时，实际加速比S达到最大， 即S＝k

三、多功能\静（动）态流水线时空图

四、流水线相关与冲突

五、经典五段流水线：

一条指令的执行过程分为以下 5 个周期：

取指令周期（IF）

指令译码/读寄存器周期（ID）

执行/有效地址计算周期（EX）

存储器访问 / 分支完成周期（ME）

取指、译码、执行、ME(访存)/分支完成、写回、

写回周期（WB）

各段完成的操作：

取指令周期（IF）

以程序计数器 PC 中内容为地址，从存储器中取出指令放入指令寄存器 IR；同时 PC 值加 4（假设每条指令占 4 个字节），指向顺序的下一条指令。

指令译码/读寄存器周期（ID）

指令译码、分析，并访问通用寄存器，读出所需的操作数。

执行/有效地址计算（EX）（不同指令所进行的操作不同）

load 和 store 指令：形成访存有效地址。ALU 把指令中所指定的寄存器的内容与偏移量相加。

ALU 指令（寄存器－寄存器）：对从通用寄存器组中读出的数据进行运算。

ALU 指令（寄存器－立即数）：ALU 对从通用寄存器组中读出的操作数和指令中给出的立即数并进行运算。

分支指令：ALU 把指令中给出的偏移量与 PC 值相加，形成转移目标的地址。同时，根据前一个周期读出的操作数，判断或确定分支是否成功。

存储器访问 / 分支完成周期（MEM）

该周期只有访问指令（load、store）和分支指令。

ALU 类型指令在此周期不做任何操作。

load 指令：从存储器中读出相应的数据；

store 指令：把指定的数据写入指出的存储器单元。

分支指令：分支“成功”，就把转移目标地址送入 PC。

写回周期（WB）

把结果数据写入通用寄存器组。

ALU 运算指令：结果数据来自 ALU。

load 指令：结果数据来自存储器

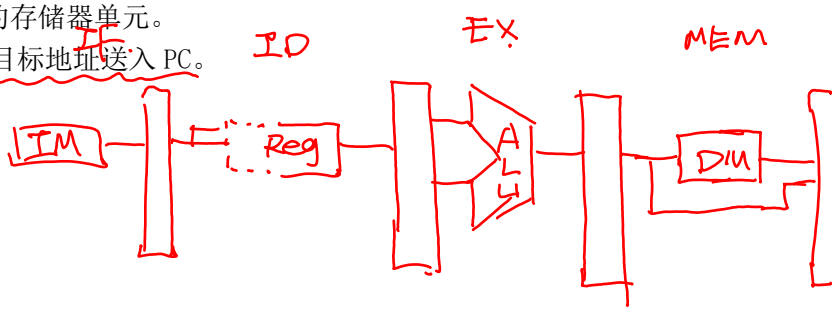
即：

分支指令和 store 指令需要 4 个周期；

其它指令需要 5 个周期才能完成。

ID 段和 WB 段都要访问同一寄存器文件。

ID 段：读　　WB 段：写



数据相关（真相关）/名相关/控制相关

指令相关：两条指令之间存在依赖关系

① **数据相关**　流水线中多条指令并行执行时 ，执行中可能导致数据供求关系上违反原定的次序，称为数据相关。 如以下两条指令：

ADD　R1，R2，R3　；(R2)＋(R3)→R1

SUB　R4，R1，R5　；(R1)－(R5)→R4

即：对于两条指令 i（在前）和 j（在后），下述条件之一成立，则称指令 j 与指令 i 数据相关

指令 j 使用（读）指令 i 产生（写 ）的结果

指令 j 与指令 k 数据相关，而指令 k 又与指令 i 数据相关。

WR （ WFR）数据相关有传递性。

指令 i（在前）和 j（在后）有数据传递。

② 名相关

名：指令所访问的寄存器或存储器单元的名称。

如果两条指令使用相同的名，但是它们之间并没有数据流动，称这两条指令存在名相关。

其中名相关又分为两种：读写相关（反相关）和输出相关

反相关：如果指令 i 读的名与指令 j 写的名相同，则称指令 i 和 j 发生了反相关（读写相关）。

指令 i 读的名＝指令 j 写的(寄存器)名

读后写.

输出相关：如果指令 j 和指令 i 写相同的名，则称指令 i 和 j 发生了输出相关（写写相关）

指令 j 写的（寄存器)名＝指令 i 写的名

写后写.

③ **控制相关**

控制相关是指由分支指令引起的相关。

包括无条件转移、条件转移、子程序调用、中断等，它们属分支指令，控制相关可能改变程序方向，造成流水线断流。

```
if p1 {
   S1;
      };
S;
if p2 {
     S2;
        };
```

WB

p1 ,S1 控制相关；p2 ,S2 控制相关；

p1 , p2 与 S 均无(控制) 相关

Dog

结构冲突、数据冲突、控制冲突：

结构冲突：硬件资源满足不了指令重叠执行的要求。

数据冲突：当指令在流水线中重叠执行时，因需要用到前面指令的执行结果而发生的冲突。

控制冲突：分支指令和其它会改变 PC 值的指令所引起的。

数据冲突的各种形式（写后读，写后写，读后写等）

减少**数据冲突**的方法：延迟、定向、编译

解决**控制冲突**的方法：排空、预测、延迟分支、编译

只有 Load/store 占用 MEM.!

ALU 不占用 MEM