北京郵電大學
Beijing University of Posts and Telecommunications

Computer Architecture

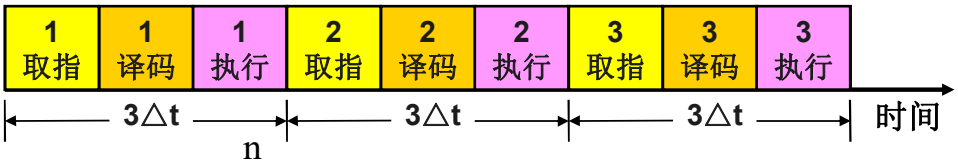
计算机系统结构

（时间\指令级并行）

北京邮电大学

邝 坚 2017年3月

从指令重叠到流水线



$$T = \sum_{i=1}^{n} (t_{取指令i} + t_{分析i} + t_{执行i})$$

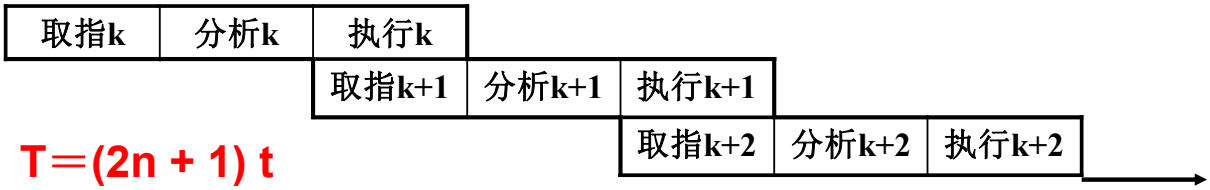
如果每段时间均为t，则执行n条指令所用的时间为：

$T = 3nt$

优点：控制简单，成本低；

缺点：执行速度慢，部件利用率低。

从指令重叠到流水线



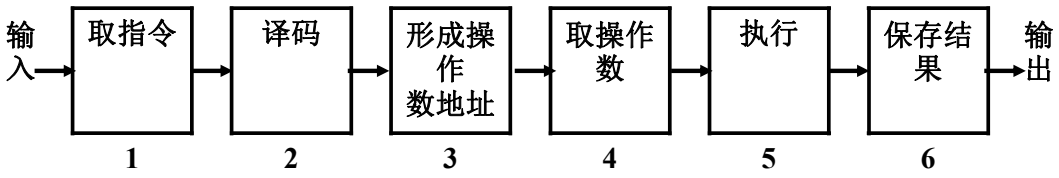
$T = (2n + 1)\,t$

$T = (n + 2)\,t$

优点和缺点？

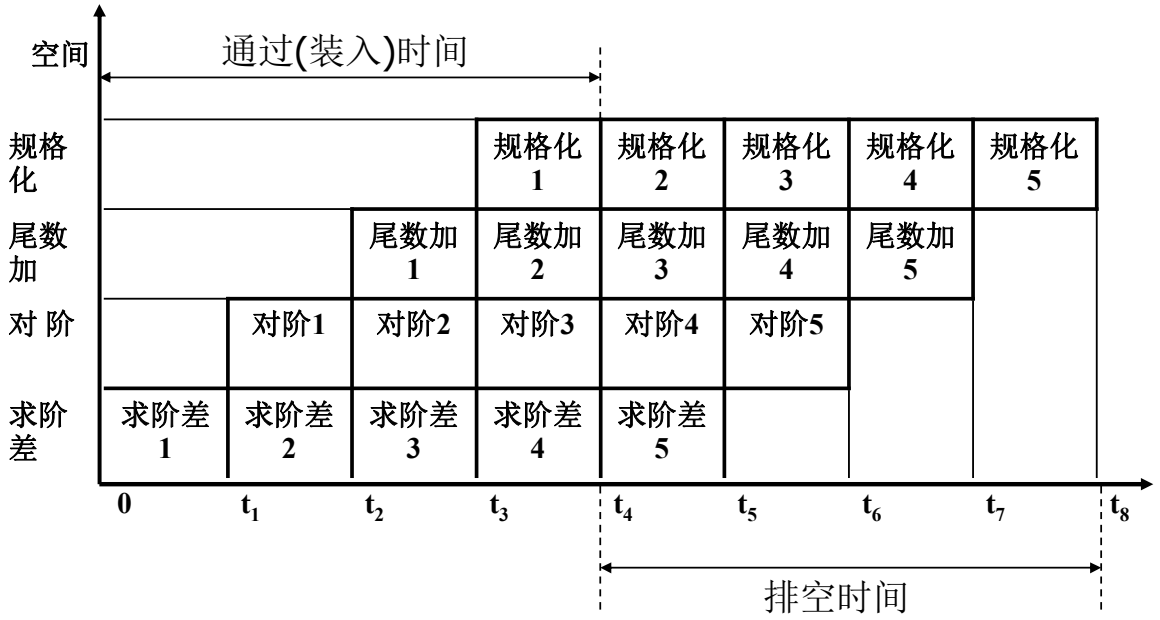
流水线的表示方法

- 连接图 - 逻辑关系

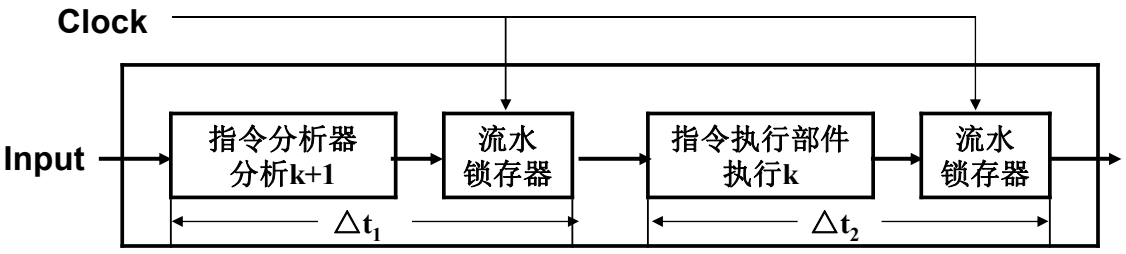


流水线的表示方法

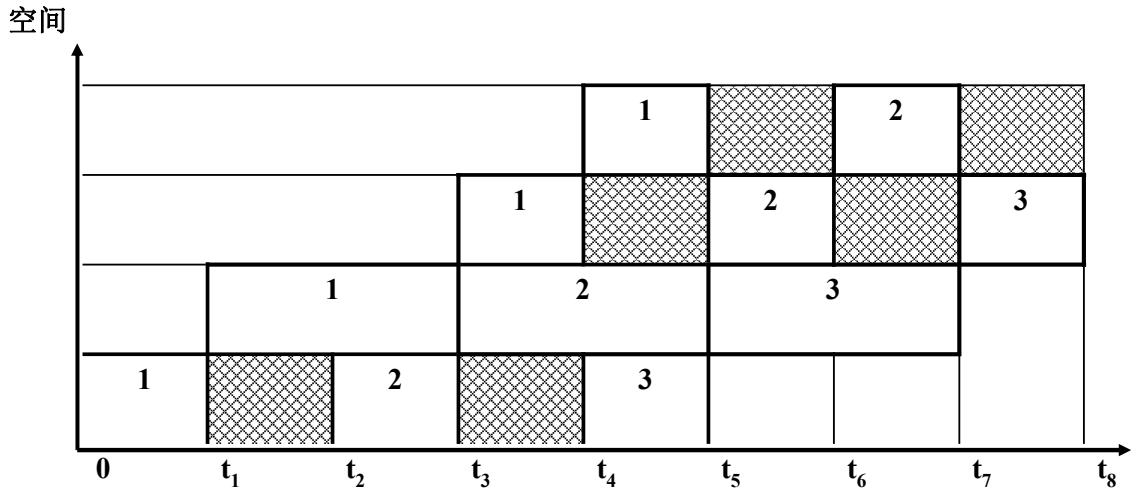
- 时空图 – 时间关系



流水线的锁存器



各段执行时间不相等的流水线

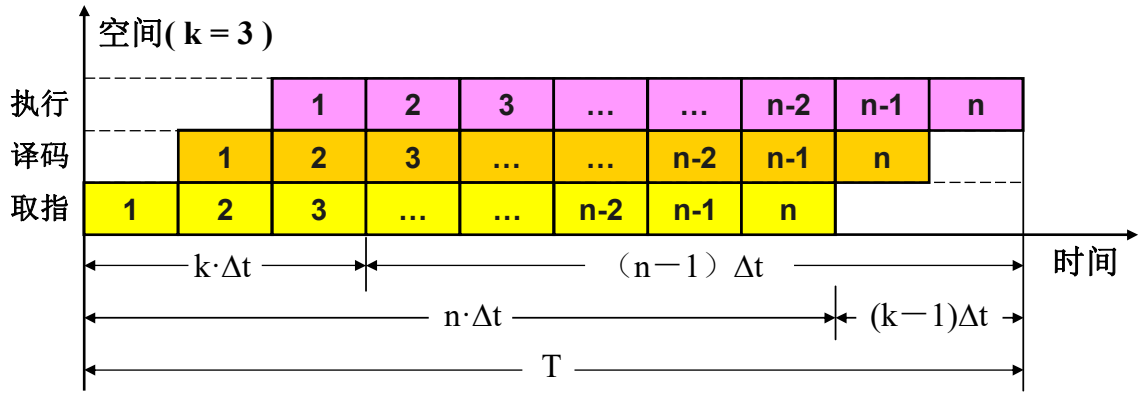


流水线的特点

- 各功能段的时间尽量相等
  - 执行时间最长的段是瓶颈
- 效率
  - 连续不断地使用（同一种功能）
  - 通过(装入)时间 – 第一个任务从进入到流出
  - 排空时间 – 最后一个任务从进入到流出

流水线的性能指标

- 吞吐率（Throughput）
  - 单位时间流水线所完成的任务数量，或输出结果的数量



流水线的性能指标

- 计算流水线吞吐率的最基本公式：

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

- 各段执行时间相等，输入连续任务情况下：完成n个连续任务需要的总时间为：

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

其中：k 为流水线的段数，Δt为时钟周期。

- 吞吐率为：

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

流水线的性能指标

- 最大吞吐率为：

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

- 各段执行时间不相等，输入连续任务
  - 吞吐率：

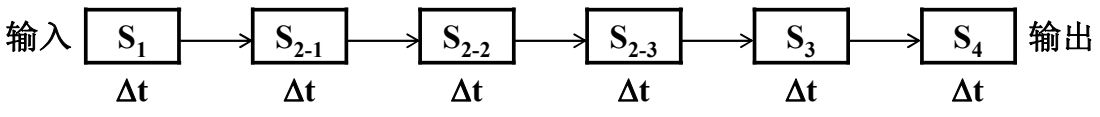
$$TP = \frac{n}{\sum_{i=1}^{k} \Delta t_i + (n-1)\max(\Delta t_1, \Delta t_2, \cdots, \Delta t_k)}$$

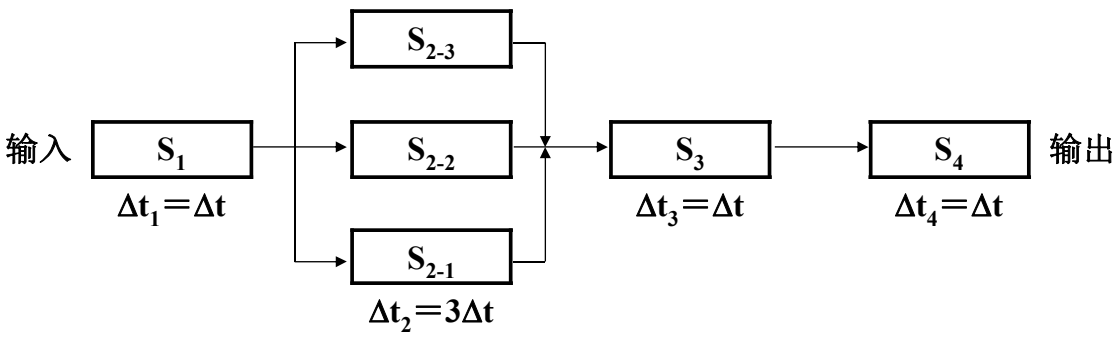
  - 最大吞吐率：

$$TP_{max} = \frac{1}{\max(\Delta t_1, \Delta t_2, \cdots, \Delta t_k)}$$

流水线的性能指标

- 解决“瓶颈”





流水线的性能指标

- 加速比（Speedup）
  - 同一任务，不使用流水线所使用时间与使用流水线所用时间比

$$S = \frac{\text{顺序执行时间 } T_s}{\text{流水线执行时间 } T_k}$$

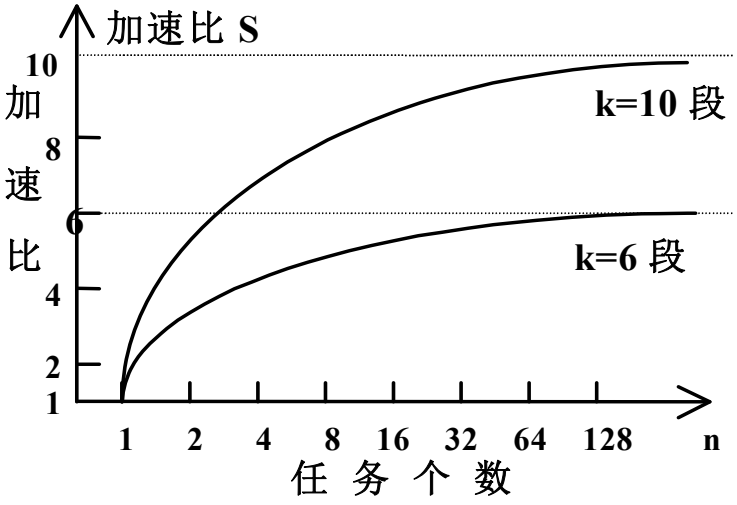
- 各段执行时间相等，输入连续任务
  - 加速比：

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

  - 最大加速比为：

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

流水线的性能指标



流水线的性能指标

- 各段执行时间不相等，输入连续任务情况下，实际加速比：

$$S = \frac{n \cdot \sum_{i=1}^{k} \Delta t_i}{\sum_{i=1}^{k} \Delta t_i + (n-1) \cdot \max(\Delta t_1, \Delta t_2, \cdots, \Delta t_k)}$$

流水线的性能指标

- 效率（Efficiency）- 流水线设备的利用率

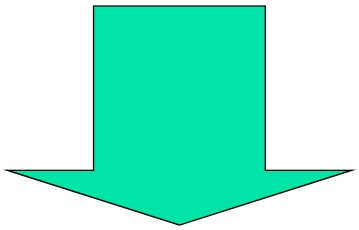
$$E = \frac{n\text{个任务占用有效面积}}{\text{对应总面积}}$$

- 各流水段执行时间相等，输入n个连续任务
- 流水线的效率为：$E = \frac{k \cdot n \cdot \Delta t}{k \cdot (k+n-1) \cdot \Delta t} = \frac{n}{k+n-1}$
- 流水线的最高效率为：$E_{\max} = \lim_{n \to \infty} \frac{n}{k+n-1} = 1$

流水线的性能指标

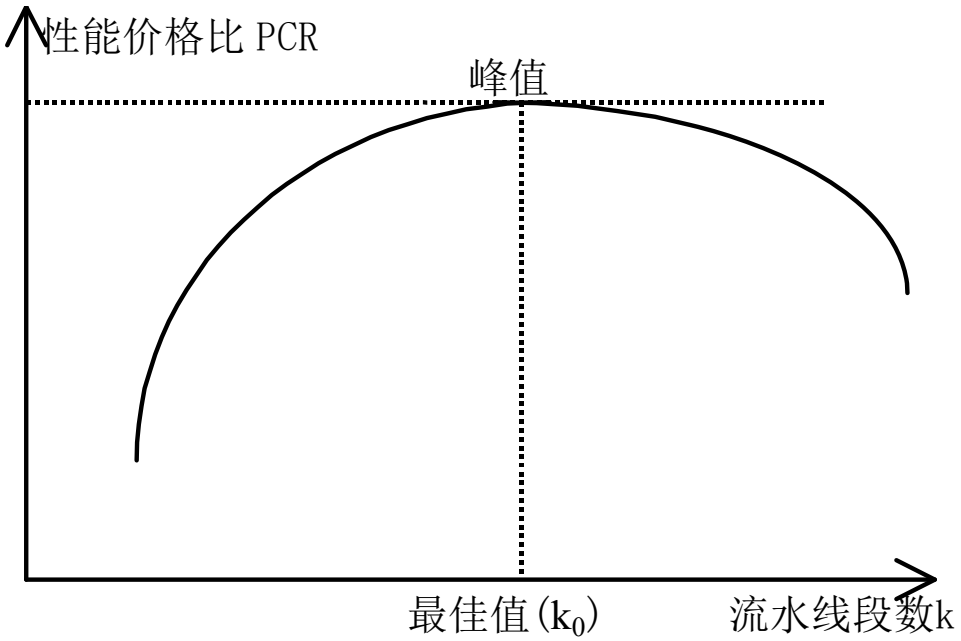
- 流水线的吞吐率、加速比与效率的关系：

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



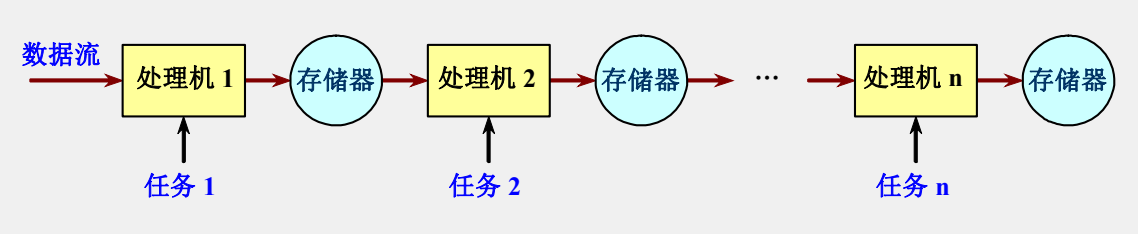
$E = TP \cdot \Delta t$，$S = k \cdot E$

最佳流水段数



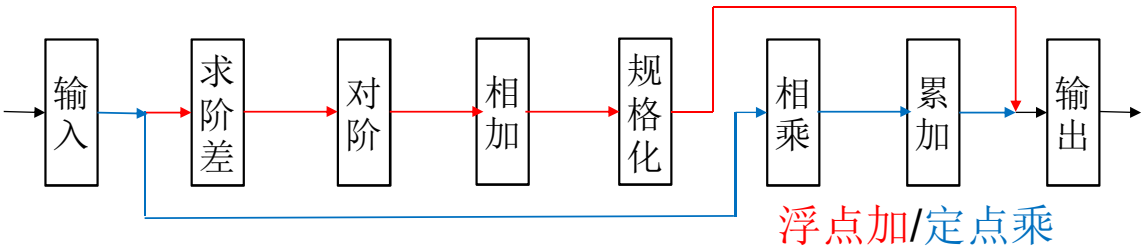
流水线的分类

- 按等级 - 部件级、处理机级和系统级流水线
  - 部件级 - 把处理机中的部件分段相互连接，使运算操作按流水方式处理，如浮点加，也称运算操作流水线(Arithmetic Pipeline)
  - 处理机级 – 把指令的执行过程分解为若干个子过程，每个在独立的功能部件中执行，即指令流水线(Instruction Pipeline)
  - 系统级 – 把多个处理机串行连接，对同一数据流进行处理，每个完成任务中的一部分。前一台的结果放入存储器，作为后一台的输入，又称宏流水线(Macro Pipeline)



流水线的分类

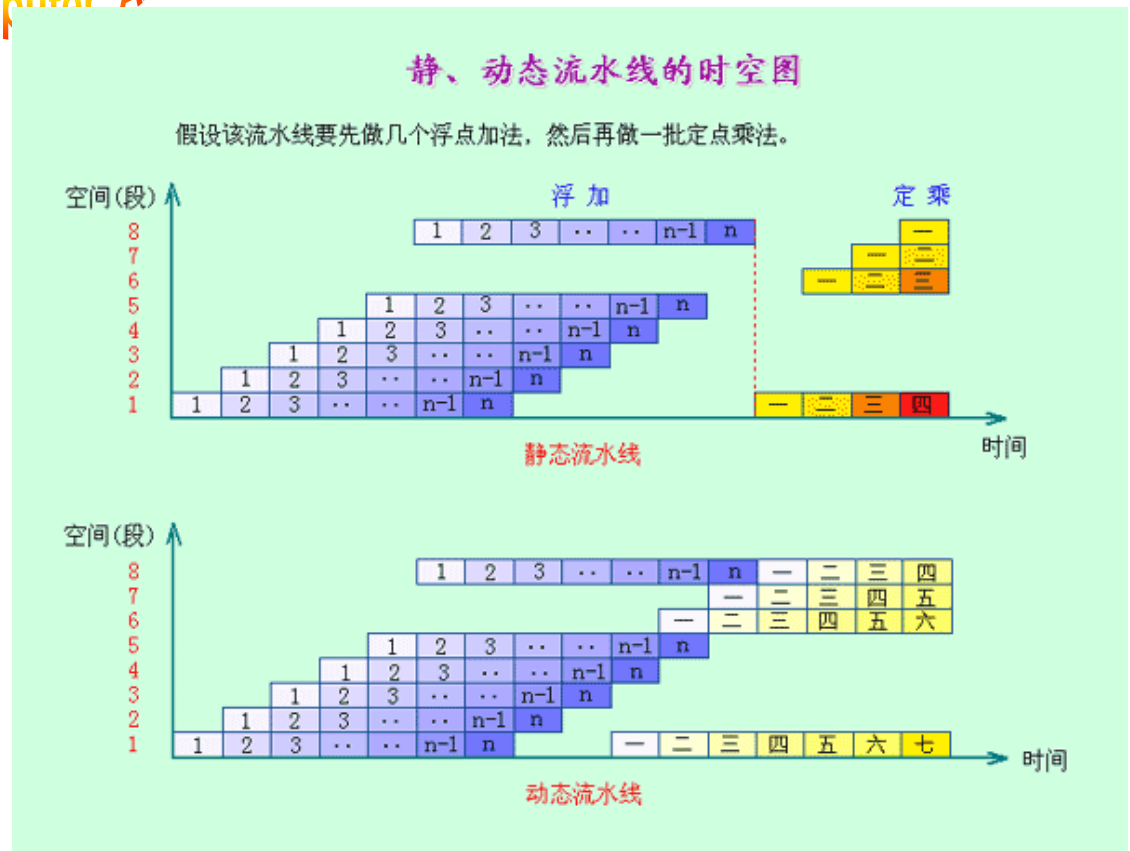
- 按完成功能的多倍性 - 单功能与多功能流水线
  - 单功能(Unifunction Pipeline) – 流水线各段之间的连接固定不变，只能完成一种功能
  - 单功能(Multifunction Pipeline) – 段之间的连接可以变化，不同的连接方式可以完成不同的功能



流水线的分类

- 静态和动态流水线 – 多功能流水线的进一步分类
  - 静态(Static Pipeline) - 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(Dynamic Pipeline) -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

流水线的分类



流水线的分类

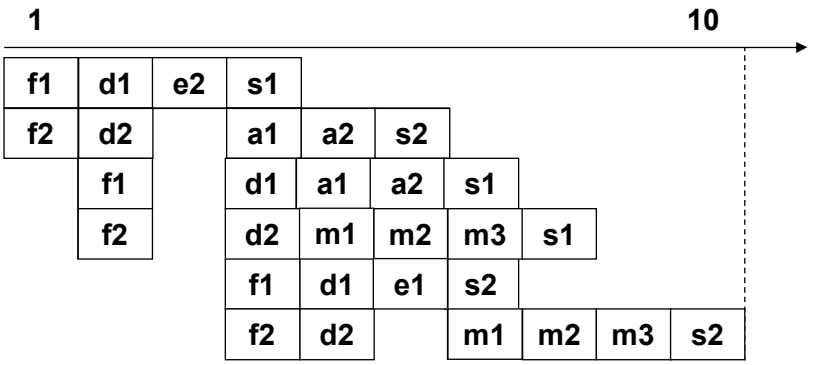
- 按流水线中是否有反馈回路分类 - 线性与非线性流水线
  - 线性流水线(Linear Pipeline) - 流水线的各段串行连接，没有反馈回路。数据通过流水线中的各段时，每一个段最多只流过一次
  - 非线性流水线(Nonlinear Pipeline) - 流水线中除了有串行的连接外，还有反馈回路，例如主参考书P57，图3.6

流水线的分类

- 根据任务流入和流出的顺序是否分类 -顺序与乱序流水线
  - 顺序流水线(In-order Pipeline) - 流水线输出端任务流出的顺序与输入端任务流入的顺序完全相同。每一个任务在流水线的各段中是一个跟着一个顺序流动的
  - 乱序流水线(Out-of-order Pipeline) - 流水线输出端任务流出的顺序与输入端任务流入的顺序可以不同，允许后进入流水线的任务先完成（从输出端流出）。也称为无序流水线、错序流水线、异步流水线

流水线的分类

```
LOAD   R1,M(A)
ADD    R2,R2,R1
ADD    R3,R3,R4
MUL    R4,R4,R5
NEG    R6,R6
MUL    R6,R6,R7
```

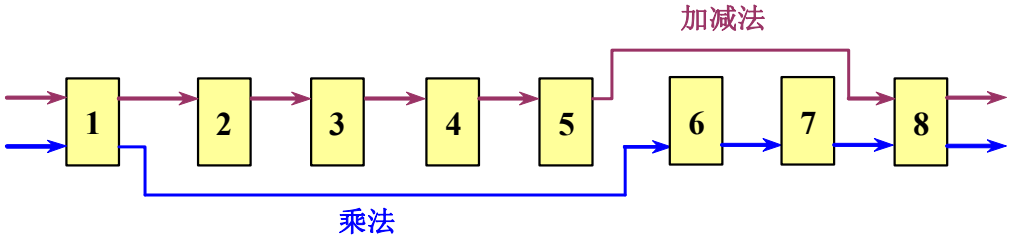


流水线的性能分析举例

例3.1 设在下图所示的静态流水线上计算：

$$\prod_{i=1}^{4}(A_i + B_i)$$

流水线的输出可以直接返回输入端或暂存于相应的流水寄存器中，试计算其吞吐率、加速比和效率。



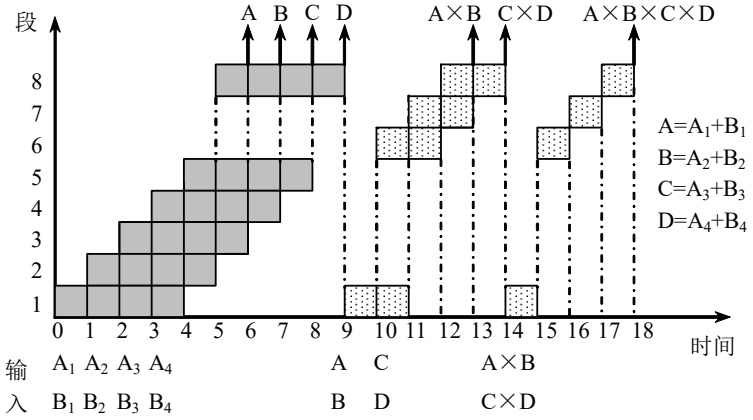
（每段的时间都为△t）

流水线的性能分析举例

解：（1）选择适合于流水线工作的算法

- 先计算A_1+B_1、A_2+B_2、A_3+B_3和A_4+B_4；
- 再计算$(A_1+B_1)\times(A_2+B_2)$和$(A_3+B_3)\times(A_4+B_4)$；
- 然后求总的乘积结果。

（2）画出时空图



流水线的性能分析举例

- 主要原因
  - 多功能流水线在做某一种运算时，总有一些段是空闲的。
  - 静态流水线在进行功能切换时，要等前一种运算全部流出流水线后才能进行后面的运算。
  - 运算之间存在关联，后面有些运算要用到前面运算的结果。
  - 流水线的工作过程有建立与排空部分。

流水线的性能分析举例

（3）计算性能

- 在18个△t时间中，给出了7个结果。吞吐率为：

$$TP = \frac{7}{18\Delta t}$$

- 不用流水线，由于一次求和需6△t，一次求积需4△t，
  则产生上述7个结果共需（4×6+3×4）△t = 36△t，加速比为

$$S = \frac{36\Delta t}{18\Delta t} = 2$$

- 流水线的效率

$$E = \frac{4\times 6 + 3\times 4}{8\times 18} = 0.25$$

可以看出，在求解此问题时，该流水线的效率不高。

流水线设计中的若干问题

- 瓶颈问题
  - 理想情况下，流水线在工作时，其中的任务是同步地每一个时钟周期往前流动一段。
  - 当流水线各段不均匀时，机器的时钟周期取决于瓶颈段的延迟时间。
  - 在设计流水线时，要尽可能使各段时间相等。

流水线设计中的若干问题

- 流水线的额外开销
  - 流水寄存器延迟 - 流水寄存器需要建立时间和传输延迟
    - 建立时间：在触发写操作的时钟信号到达之前，寄存器输入必须保持稳定的时间。
    - 传输延迟：时钟信号到达后到寄存器输出可用的时间。
  - 时钟偏移开销
    - 流水线中，时钟到达各流水寄存器的最大差值时间。（时钟到达各流水寄存器的时间不是完全相同）

流水线设计中的若干问题

- 几个问题
  - 流水线并不能减少（而且一般是增加）单条指令的执行时间，但却能提高吞吐率。
  - 增加流水线的深度（段数）可以提高流水线的性能。
  - 流水线的深度受限于流水线的额外开销。
  - 当时钟周期小到与额外开销相同时，流水已没意义。因为这时在每一个时钟周期中已没有时间来做有用的工作。
- 冲突问题
  - 流水线设计中要解决的重要问题之一。

流水线的性能指标
Dependences/Hazards

流水线参考模型- MIPS 5段流水

	ALU	Load/Store	Branch
IF		取　指	
ID		译码, 读寄存器堆	
EX	执行	计算访存有效地址	计算目标地址, 设置条件码
MEM	（空操作）	访问存储器	若条件成立, 转移目标地址送PC
WB	计算结果写回寄存器堆	Load数据写回寄存器堆	（空操作）

RISC & PowerPC

现实及问题

```
SUB   R1,R9,R6    ;R9 - R6 → R1
ADD   R3,R1,R2    ;R1 + R2 → R3
ORI   R5,R3,0x1   ;R3 xor 0x1 → R5
```

指令之间在R1，R3上存在先写后读相关

	ALU
IF	取 指
ID	译码，读寄存器堆
EX	执行
MEM	（空操作）
WB	计算结果写回寄存器堆

现实

理想时空图

SUB	IF	ID	EX	MEM	WB 写R1		
ADD		IF	ID 读R1	EX	MEM	WB 写R3	
ORI			IF	ID 读R3	EX	MEM	WB

流水线冲突/冒险

实际的时空图

SUB	IF	ID	EX	ME	WB						
ADD		IF	*stall*	*stall*	*stall*	ID	EX	ME	WB		
ORI						IF	*stall*	*stall*	*stall*	ID	EX

流水线冲突/冒险(Hazard) – that prevent the next instruction in the instruction stream from executing during its designated clock cycle.

数据冲突/冒险(Data hazard)

结构冲突/冒险(Structural hazard)

控制冲突/冒险(Control hazard)

冲突源于相关 - Dependence

- 相关：两条指令之间存在某种依赖关系。
  - 如果两条指令相关，则它们就有可能不能在流水线中重叠执行或者只能部分重叠执行。
- 相关有3种类型
  - 数据相关（也称真数据相关）
  - 名相关
  - 控制相关

数据相关

- 数据相关
  - 对于两条指令i（在前，下同）和j（在后，下同），如果下述条件之一成立，则称指令j与指令i数据相关。
    - 指令j使用指令i产生的结果；
    - 指令j与指令k数据相关，而指令k又与指令i数据相关。
  - 数据相关具有传递性。
  - 数据相关反映了数据的流动关系，即如何从其产生者流动到其消费者。

数据相关

- 例如：下面这一段代码存在数据相关。

```
Loop:  L.D     F0, 0(R1)      // F0为数组元素
       ADD.D   F4, F0, F2     // 加上F2中的值
       S.D     F4, 0(R1)      // 保存结果
       DADDIU  R1, R1, -8     // 数组指针递减8个字节
       BNE     R1, R2, Loop   // 如果R1≠R2，则分支
```

数据相关

- 当数据的流动是经过寄存器时，相关的检测比较直观和容易。
- 当数据的流动是经过存储器时，检测比较复杂。
  - 相同形式的地址其有效地址未必相同，如不同指令中的10(R5)。
  - 形式不同的地址其有效地址却可能相同。

名(name)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，则称这两条指令存在名相关。

- 指令j与指令i之间的名相关有两种：
  - 反(Anti)相关：如果指令j写的名与指令i读的名相同，则称指令i和j发生了反相关。

    指令j写的名＝指令i读的名

  - 输出(Output)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。

    指令j写的名＝指令i写的名

名(name)相关

```
DIV.D   F2，F6，F4
ADD.D   F6，F0，F12          ;DIV.D和ADD.D存在反相关
SUB.D   F8，F6，F14
```

- 名相关的两条指令之间并没有数据的传送。
- 如果一条指令中的名改变了，并不影响另外一条指令的执行。

- 换名(renaming)技术 - 通过改变指令中操作数的名来消除名相关。对于寄存器操作数进行换名称为寄存器换名(Register Renaming)。如：

```
DIV.D   F2，F6，F4
ADD.D   S，F0，F12           ;反相关消除
SUB.D   F8，S，F14
```

既可以用编译器静态实现，也可以用硬件动态完成。

控制相关

- 控制(Control)相关是指由分支指令引起的相关。
  - 为了保证程序应有的执行顺序，必须严格按控制相关确定的顺序执行。
- 典型的程序结构是“if-then”结构，如：

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

控制相关

- 控制相关带来了以下两个限制：
  - 与一条分支指令控制相关的指令不能被移到该分支之前，否则这些指令就不受该分支控制。
    - 对上例，then 部分中的指令不能移到if语句之前。
  - 如果一条指令与某分支指令不存在控制相关，就不能把该指令移到该分支之后。
    - 对上例子，不能把S移到if语句的then部分中。