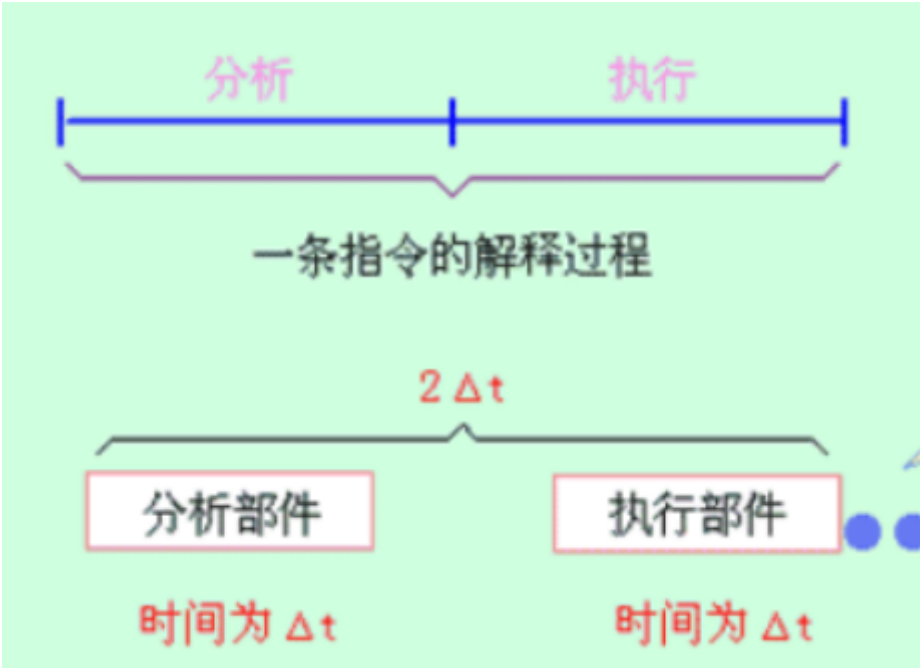
流水线技术

指令流水线通过指令重叠减小 CPI。

指令流水线：

把指令的解释过程分解为“分析”和“执行”两个子过程，并让这两个子过程分别用独立的分析部件和执行部件来实现。 理想情况下速度提高一倍



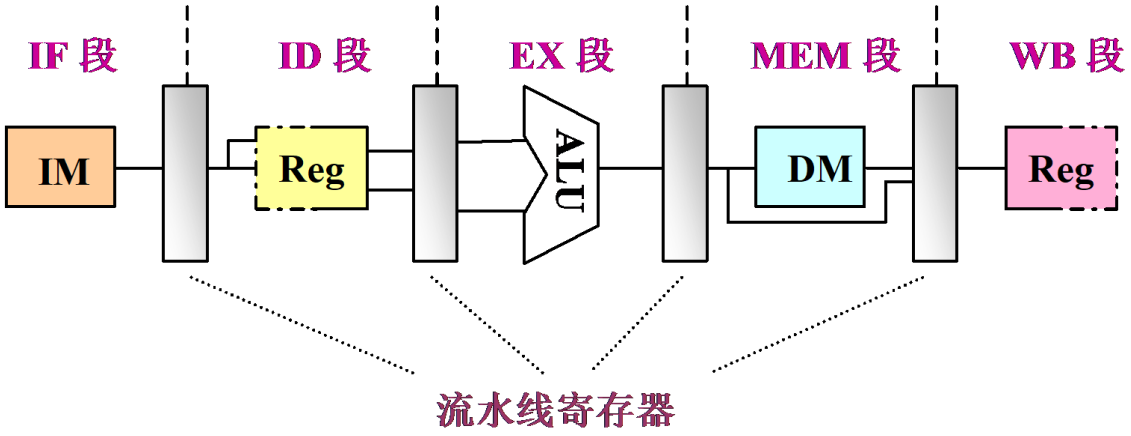
当分析部件完成上一条指令的分析后，立即进入下一条指令执行部件进行执行操作，同时，再分析下一条指令。

浮点加法流水线

把浮点加法的全过程分解为“求阶差”、“对阶”、“尾数相加”、“规格化”四个子过程，并让它们分别用各自独立的部件来实现。理想情况：速度提高3倍

简单的5段流水线

五个阶段： 取指（IF）、译码（ID），执行（EX），存储器访存（MEM，Only Load and Store），写回（WB）。



时 - 空图

时 - 空图从时间和空间两个方面描述了流水线的工作过程。时 - 空图中，横坐标代表时间，纵坐标代表流水线的各个段。

流水线分类

处理级别

操作级流水、指令集流水、处理器级流水（宏流水）

功能多少

单功能流水线：只能完成单一功能 \
多功能流水线：可以完成多种功能

按同一时间内各段之间的连接方式

静态多功能流水线：同一时间内只能输入一串运算操作相同的指令（即单一功能），其效率才能得到发挥。\
动态多功能流水线：同一时间可以允许多种功能的连接方式同时工作。优点：提高流水线效率；缺点：会使流水线控制变得复杂

流水线的性能指标

吞吐量

单位时间流水线所完成（输出）的任务数量

$$TP = \frac{n}{T_{all}}$$

其中：
TP： 吞吐量

n：任务数

T_{all}：完成n个任务所需要的总时间

假设有k段流水线，每段流水线完成所需要的时间为Δt,在理想情况下，完成n个任务需要的总时间为 $(k+n-1)\Delta t$,则在k段流水线下的理想吞吐率

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

并且存在 当 $n \to \infty$是有

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

加速比

完成同一批任务，使用流水线与不使用流水线的所用时间之比。

$$S = \frac{T_{不使用}}{T_{使用}}$$

假设有k段流水线，每段需要的时间为Δt,在理想的使用流水线的情况下，完成n个任务需要的总时间为$(k+n-1)\Delta t$, 在理想的不使用流水线的情况下，完成n个任务的总时间$kn\Delta t$,则：

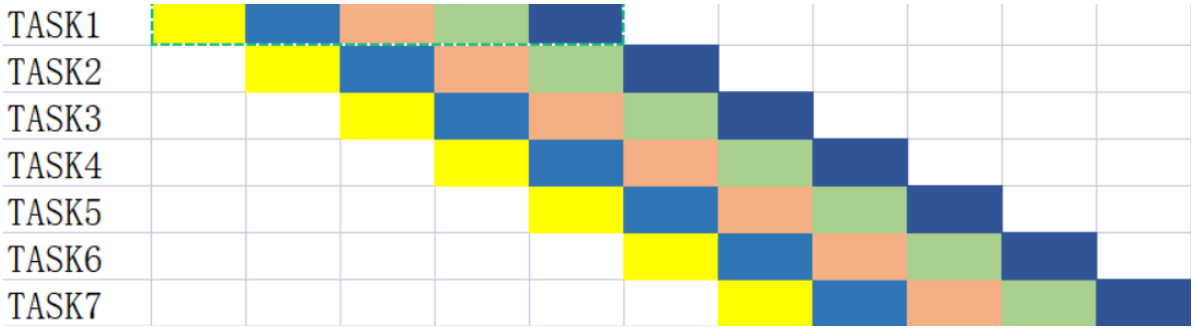
$$S=\frac{T_{不使用}}{T_{使用}}=\frac{kn\Delta t}{(k+n-1)\Delta t}=\frac{kn}{k+n-1}$$

并且存在 当 $n\to\infty$是有

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

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，即流水线设备的利用率。由于流水线有通过时间和排空时间，所以在连续完成n个任务的时间内，各段并不是满负荷地工作。



如图是，一个五段流水线执行七个任务，总的执行时间$(k+n-1)\Delta t=11\Delta t$, 其中每一个阶段的效率都是相等的，且都为$e_i=\frac{7}{11}$, 所以总的效率

$$E=\frac{\sum_{i=1}^{k}e_i}{k}=e_i=\frac{7}{11}$$

设 k 段流水线，每段的流水线工作效率，为e_k,
则

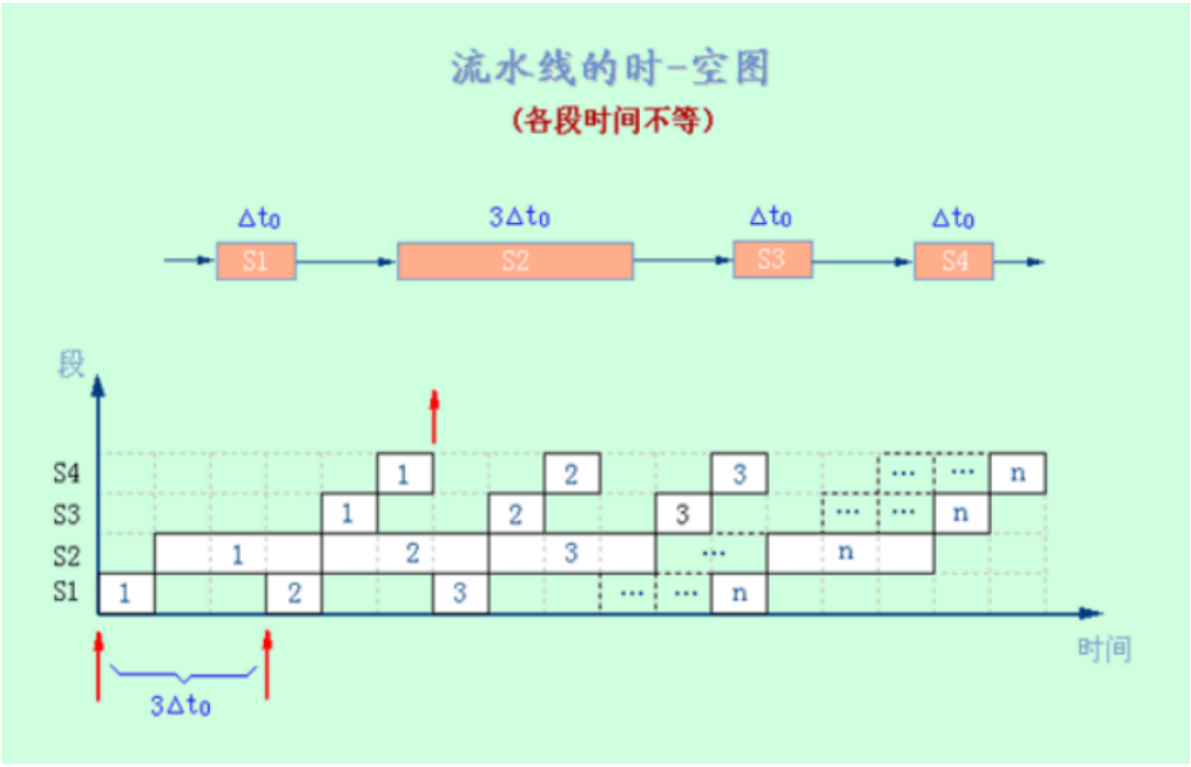
$$E=\frac{\sum_{i=1}^{k}e_i}{k}=e_i=\frac{n}{(k+n-1)\Delta t}=TP\Delta t=\frac{S}{k}$$

当 n >> k (n → ∞) 时,

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

流水线的最慢的段

事实上，流水线的每个段并不是都是相等的，在各段时间不完全相等的流水线中，时间最长的段叫做**瓶颈段**。



如图在这个 4段 流水线中，总共执行 n 个任务，可以显而易见的是部件二，占据了任务的一大部分\
当执行一个任务时候，$t_1 = 6\Delta t_0$。\
当执行两个任务时候，$t_2 = \Delta t_0 + 2\times 3\Delta t_0 + \Delta t_0 + \Delta t_0 = 6\Delta t_0 + 3\Delta t_0$。 \
.... \
当执行n个任务的时候$t_n = \Delta t_0 + (n)\times 3\Delta t_0 + \Delta t_0 + \Delta t_0 = 6\Delta t_0 + (n-1)\times 3\Delta t_0$

故实际吞吐率：

$$TP = \frac{n}{6\Delta t_0 + (n-1)\times 3\Delta t_0}$$

当 n → ∞ 时

$$TP_{max} = \frac{1}{3\Delta t_0}$$

所以，当k段流水线，每一段流水线的时间为$\Delta t_0, \Delta t_1, \Delta t_2, \ldots, \Delta t_{k-1}$的时候，执行n条指令，总耗时：

$$t_n = \sum_{i=0}^{k-1} \Delta t_i + (n-1)\times \max(\Delta t_0, \Delta t_1, \Delta t_2, \ldots, \Delta t_{k-1})\$\$ \tag{1}$$

$$TP = \frac{n}{t_n} = \frac{n}{\sum_{i=0}^{k-1} \Delta t_i + (n-1)\times \max(\Delta t_0, \Delta t_1, \Delta t_2, \ldots, \Delta t_{k-1})} \tag{2}$$

当 n → ∞ 时，

$$TP_{max} = \frac{1}{max(\Delta t_0, \Delta t_1, \Delta t_2, \ldots, \Delta t_{k-1})} \tag{3}$$

$$S = \frac{T_{unuse}}{T_{use}} = \frac{\sum_{i=0}^{k-1} \Delta t_i}{\sum_{i=0}^{k-1} \Delta t_i + (n-1)\times \max(\Delta t_0, \Delta t_1, \Delta t_2, \ldots, \Delta t_{k-1})} \tag{4}$$

并且存在 当 $n \to \infty$是有

$$S \quad \sum_{i=0}^{k-1} \Delta t_i \tag{5}$$

$$\mathfrak{I}_{max} = \frac{1}{max(\Delta t_0, \Delta t_1, \Delta t_2, \ldots, \Delta t_{k-1})} \tag{5}$$

此时，效率变为

$$E = \frac{n \times \sum_{i=0}^{k-1} \Delta t_i}{k \times [\sum_{i=0}^{k-1} \Delta t_i + (n-1) \times \max(\Delta t_0, \Delta t_1, \Delta t_2, \ldots, \Delta t_{k-1})]} \tag{6}$$

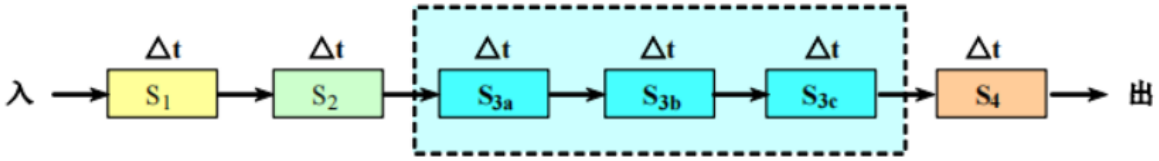
从时空图上来看

$$E = \frac{n\text{个任务实际占用的时空区}}{k\text{个段的个段总的时空区}} \tag{7}$$

解决流水线的瓶颈

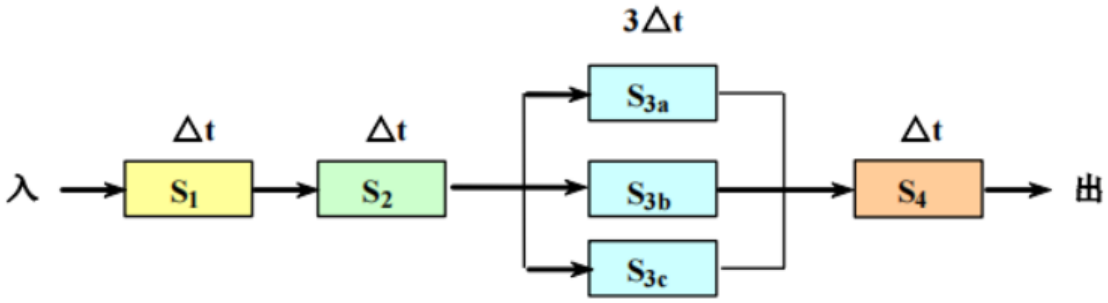
细分瓶颈段

将瓶颈段分割成更小的段的串联，如图:

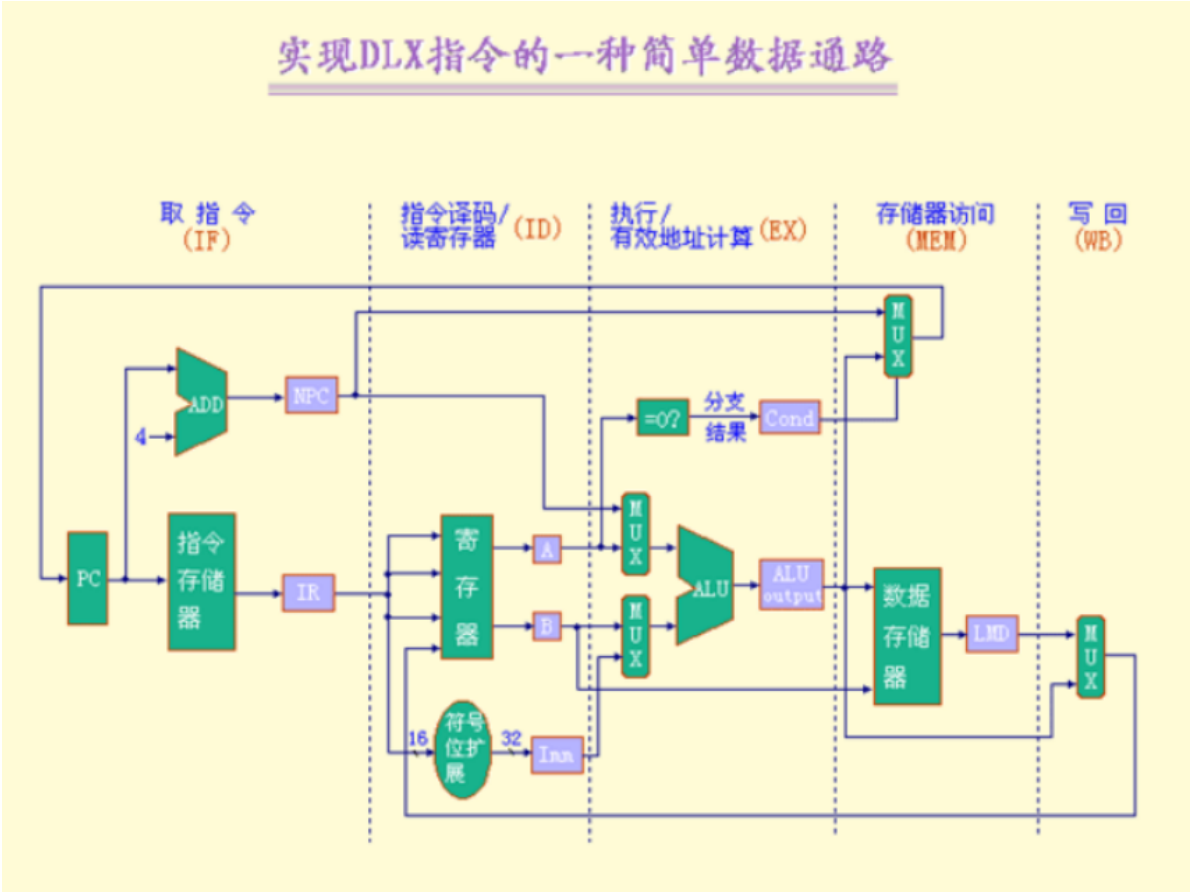


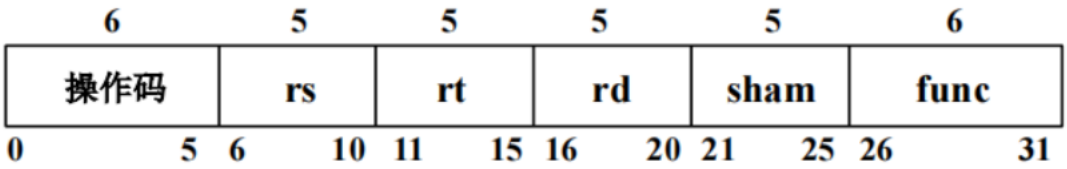
重复设置瓶颈段

将瓶颈段重复的设置，让不同的指令可以同时执行，而不冲突，如图:



DLX的基本流水线





取指令周期（IF）

把指令送入指令寄存器IR，再把PC指向下一条指令（PC + 4）

指令译码/读寄存器周期（ID）

把寄存器（IR[IR$_{6..10}$], 指令的6-10位是寄存器R_s的编号）中的值送入A，把寄存器（IR[IR$_{11..15}$], 指令的11-15位是寄存器R_t的编号）中的值送入B, 对于立即数Imm，机器会将16位的立即数符号拓展为32位。

执行/有效地址计算周期（EX）

不同的指令有不同的操作。

存储器访问： $ALU_{out} \leftarrow A + Imm$

寄存器—寄存器 ALU 操作： $ALU_{out} \leftarrow A\ op\ B$

寄存器—立即值 ALU 操作： $ALU_{out} \leftarrow A\ op\ Imm$

分支操作:
$ALU_{out} \leftarrow PC + Imm$ 或 $Cond \leftarrow (A\ op\ 0)$

存储器访问/分支完成周期（MEM）

在该周期处理的DLX指令只有*Load*、*Store*和分支指令。

存储器访问:
Mem [ALU_{out}] ← B 或者 LMD ← Mem[ALU_{out}]

分支操作：如果条件成立，则将ALU_{out}送到PC

写回周期（WB）

不同指令在该周期完成的工作也不一样
寄存器—寄存器型 ALU 指令： 将ALU_{out}送到指令的第16-20位所指的寄存器中。

寄存器—立即值型 ALU 指令： 将ALU_{out}送到指令的第11-15位所指的寄存器中。

Load 指令： 将LMD送到指令的第11-15位所指的寄存器中。

表3.1 DLX流水线的每个流水段的操作

流水段	任何指令类型		
IF	IF/ID.IR ← Mem[PC] IF/ID.NPC,PC ← (if EX/MEM.cond {EX/MEM.NPC} else {PC+4});		
ID	ID/EX.A ← Regs[IF/ID.$IR_{6..10}$]; ID/EX.B ← Regs[IF/ID.$IR_{11..15}$]; ID/EX.NPC ← IF/ID.NPC; ID/EX.IR ← IF/ID.IR; ID/EX.Imm ← $(IR_{16})^{16}$##$IR_{16..31}$;		
	ALU 指令	Load/Store 指令	分支指令
EX	EX/MEM.IR ← ID/EX.IR; EX/MEM.ALUOutput ← ID/EX.A op ID/EX.B 或 EX/MEM.ALUOutput ← ID/EX.A op ID/EX.Imm; EX/MEM.cond ← 0;	EX/MEM.IR ← ID/EX.IR; EX/MEM.ALUOutput ← ID/EX.A + ID/EX.Imm;	EX/MEM.ALUOutput ← ID/EX.NPC + ID/EX.Imm; EX/MEM.cond ← (ID/EX.A op 0);
流水段	任何指令类型		
	ALU 指令	Load/Store 指令	分支指令
MEM	MEM/WB.IR ← EX/MEM.IR; MEM/WB.ALUOutput ← EX/MEM.ALUOutput;	MEM/WB.IR ← EX/MEM.IR; MEM/WB.LMD ← Mem[EX/MEM.ALUOutput]; 或 Mem[EX/MEM.ALUOutput] ← EX/MEM.B;	
WB	Regs[MEM/WB.$IR_{16..20}$]← MEM/WB.ALUOutput; 或 Regs[MEM/WB.$IR_{11..15}$]← MEM/WB.ALUOutput;	Regs[MEM/WB.$IR_{11..15}$]← MEM/WB.LMD;	

流水线的相关

流水线中的相关是指相邻或相近的两条指令因存在某种关联，后一条指令不能在原指定的时钟周期开始执行。

结构相关 (结构冒险)

多条指令在同一时刻同时争用一个资源而形成的冲突。

消除办法

暂停等待、资源重复

控制相关 (控制冒险)

执行分支指令的结果有两种

- 分支成功：PC值改变为分支转移的目标地址。
  在条件判定和转移地址计算都完成后，才改变PC值。
- 不成功或者失败：PC的值保持正常递增，指向顺序的下一条指令。

分支需要解决两个问题

- 分支目标地址（转移成功意谓着PC值不是 PC+4）
- CC是否有效，这两点在MIPS中都在流水线的靠后段中确定

指令通常是顺序执行的，在遇到一些改变指令执行顺序的情况，如转移、返回、调用，会改变PC值，会造成断流的情况。

消除办法

分支预测，尽早生成转移地址

预取转移成功和不成功的两个控制流方向上的目标地址

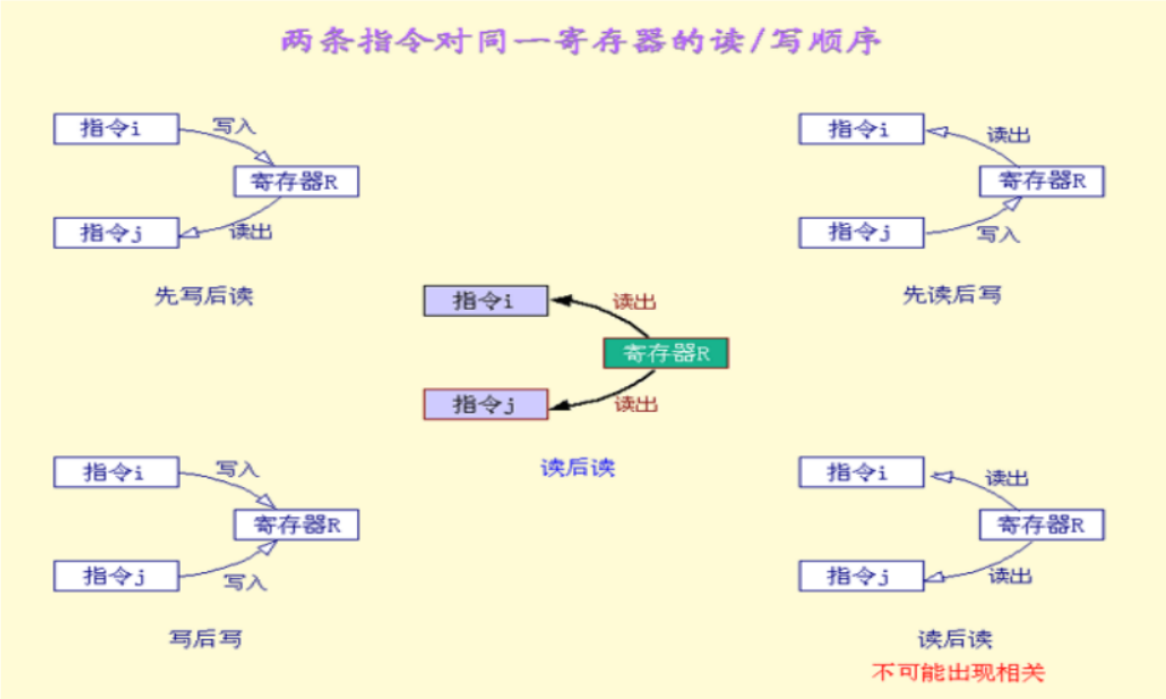
加快和形成条件码

提高转移方向上的猜准率

"冻结"或者"排空"流水线 。(简单)

数据相关 (数据冒险)

下一条指令会用到当前指令输出的结果，会造成数据冲突。



写后读数据相关 （RAW）

当前指令在写回后，下一条才能读

读后写数据相关 (WAR)

当前指令在读出完毕后，下一条才能写回

写后写数据相关 (WAW)

当前指令在写回后，下一条才能继续写回

消除办法

数据旁路技术（定向技术）：设置专门的数据通路，在不等待把数据送回寄存器组的时候，直接读取ALU的计算结果作为自己的输入
编译优化：调整指令执行顺序

暂停：暂停一个或多个周期，直到冲突消失。包括硬件阻塞Stall和软件阻塞NOP（空指令）

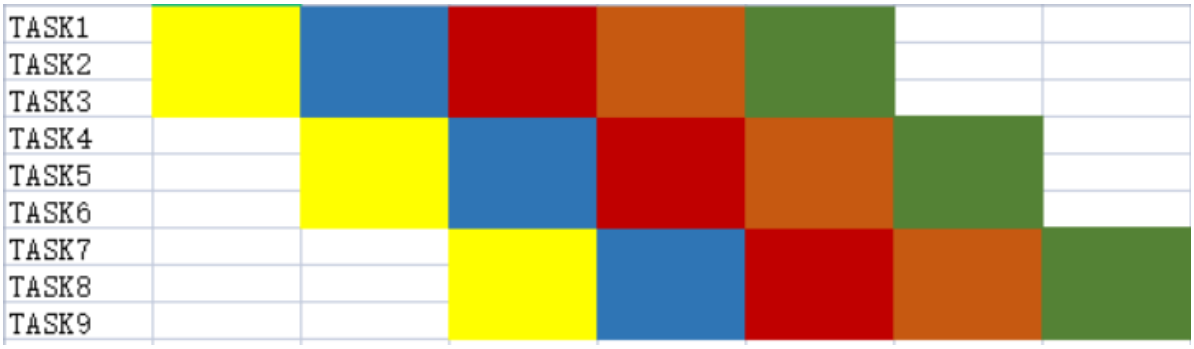
减少分支延时的方法

- 硬件的方法
  - 修改数据通路：使得目标地址和分支条件尽早确定，其中之一尽早确定是没有用的
    - » 判断是否为0可以在ID段确定
    - » 使用另一个加法器计算
      - 可以在ID段计算BTA(分支目标地址)，即在ID段形成下一条指令地址，两种可能（BTA, PC+4），选择哪一个取决于ID段确定的CC
    - » 必要时使用互锁机制来插入Stall
  - 设计合适的ISA
    - » e.g. BNEZ, BEQZ on MIPS 使得CC可以在ID段确定
- 软件（通过编译器）的方法：
  - 调度一些指令放入分支的延迟槽中
  - 预测的方法：统计分支成功和失败的情况，提高预测精度

高级流水线技术

超标量流水线

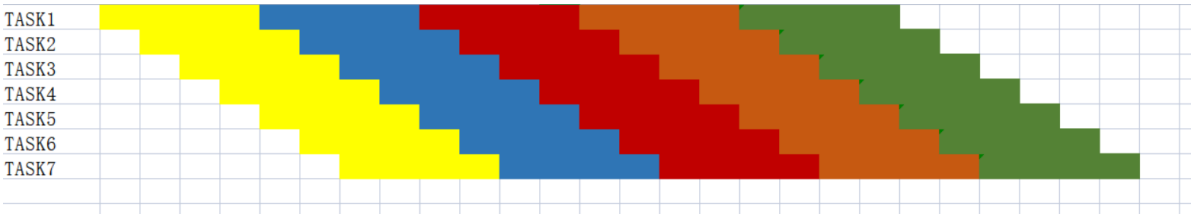
动态多发射技术，每个时钟周期内可以并发多条独立的指令，以并行的方式编译和执行，需配置多个功能部件。**不能调整指令的执行顺序**，通过编译优化技术可以挖掘其更多的并行性。



超长指令字长

静态多发射技术，由编译程序完成，将多条可以并发执行的程序拼接成一条具有多个操作码的超长指令字执行，需配置多个功能部件。

超流水线技术



流水线级数越多，时钟周期就越短，吞吐率就越高。
通过提高流水线主频来提高流水线性能。但是流水线级数越多，用于流水寄存器的开销越大，故不是越多越好。
超流水线CPU在充满后，每个时钟周期还是执行一条指令CPI = 1，但其主频更高。
多发射技术，CPI < 1，每个时钟周期可以处理多条指令。