5.5 硬布线控制器

5.5 硬布线控制器

1、实现方法

- 通过逻辑电路直接连线而产生，又称为组合逻辑控制方式
- 一旦控制部件构成后，除非重新设计和物理上对它重新布线，否则要想增加新的控制功能是不可能的
- 这种逻辑电路是一种由门电路和触发器构成的复杂树形网络，故称之为硬布线控制器

2、设计目标

- 使用最少元件
- 速度最高

5.5 硬布线控制器

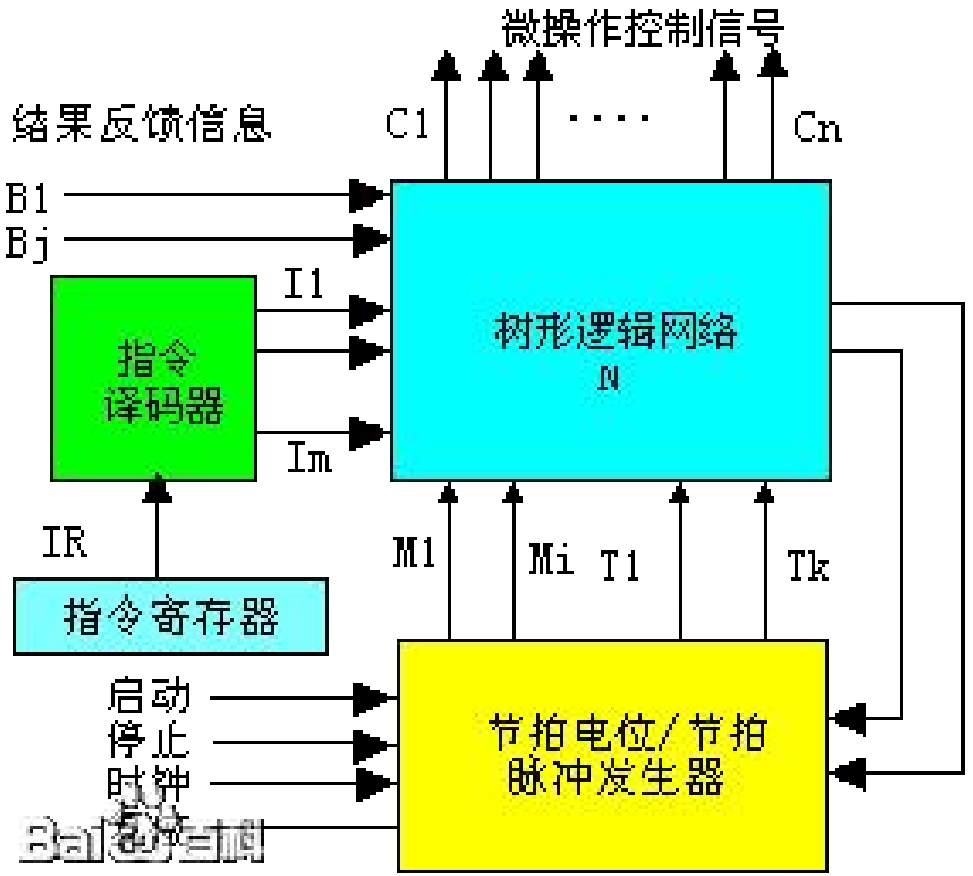
- 当执行不同的机器指令时，通过激活一系列彼此很不相同的控制信号来实现对指令的解释，其结果使得控制器往往很少有明确的结构而变得杂乱无章
- 结构上的这种缺陷使得硬布线控制器的设计和调试非常复杂且代价很大
- 正因为如此，硬布线控制器被微程序控制器所取代

- 与微程序控制相比，硬连线控制的速度较快，其原因是微程序控制中每条微指令都要从控存中读取一次，影响了速度，而硬连线控制主要取决于电路延迟。因此在某些超高速新型计算机结构中，又选用了硬连线控制器，或与微程序控制器混合使用

5.5 硬布线控制器

3、结构方框图

- 逻辑网络的输入信号来源有三个
- 来自指令操作码译码器的输出I_m
- 来自执行部件的反馈信息B_j
- 来自时序产生器的时序信号，包括节拍电位信号M和节拍脉冲信号T，其中节拍电位信号是规定的机器周期（CPU周期）信号，节拍脉冲信号是时钟周期信号



- 逻辑网络的输出信号就是微操作控制信号，它用来对执行部件进行控制

5.5 硬布线控制器

4 微操作控制信号产生

- 在微程序控制器中，微操作控制信号由微指令产生，并且可以重复使用
- 在硬布线控制器中，某一微操作控制信号由布尔代数表达式描述的输出函数产生

- 硬布线控制器的基本原理，归纳起来可叙述为：某一微操作控制信号C是指令操作码译码器输出I_m、时序信号（节拍电位 M_i，节拍脉冲T_k）和状态条件信号B_j的函数，即

$$C=f(I_m, M_i, T_k, B_j)$$

5.5 硬布线控制器

- 一般来说，还要考虑节拍脉冲和状态条件的约束，所以每一个控制信号C，可以由以下形式的逻辑方程来确定：

$$C_n = \sum_i \left(M_i \cdot T_k \cdot B_j \sum_m I_m \right)$$

- 控制信号是用门电路、触发器等许多器件采用组合逻辑设计方法来实现的
- 当机器加电工作时，某一操作控制信号C在某条特定指令和状态条件下，在某一序号的特定节拍电位和节拍脉冲时间间隔中起作用，从而激活这条控制信号线，对执行部件实施控制

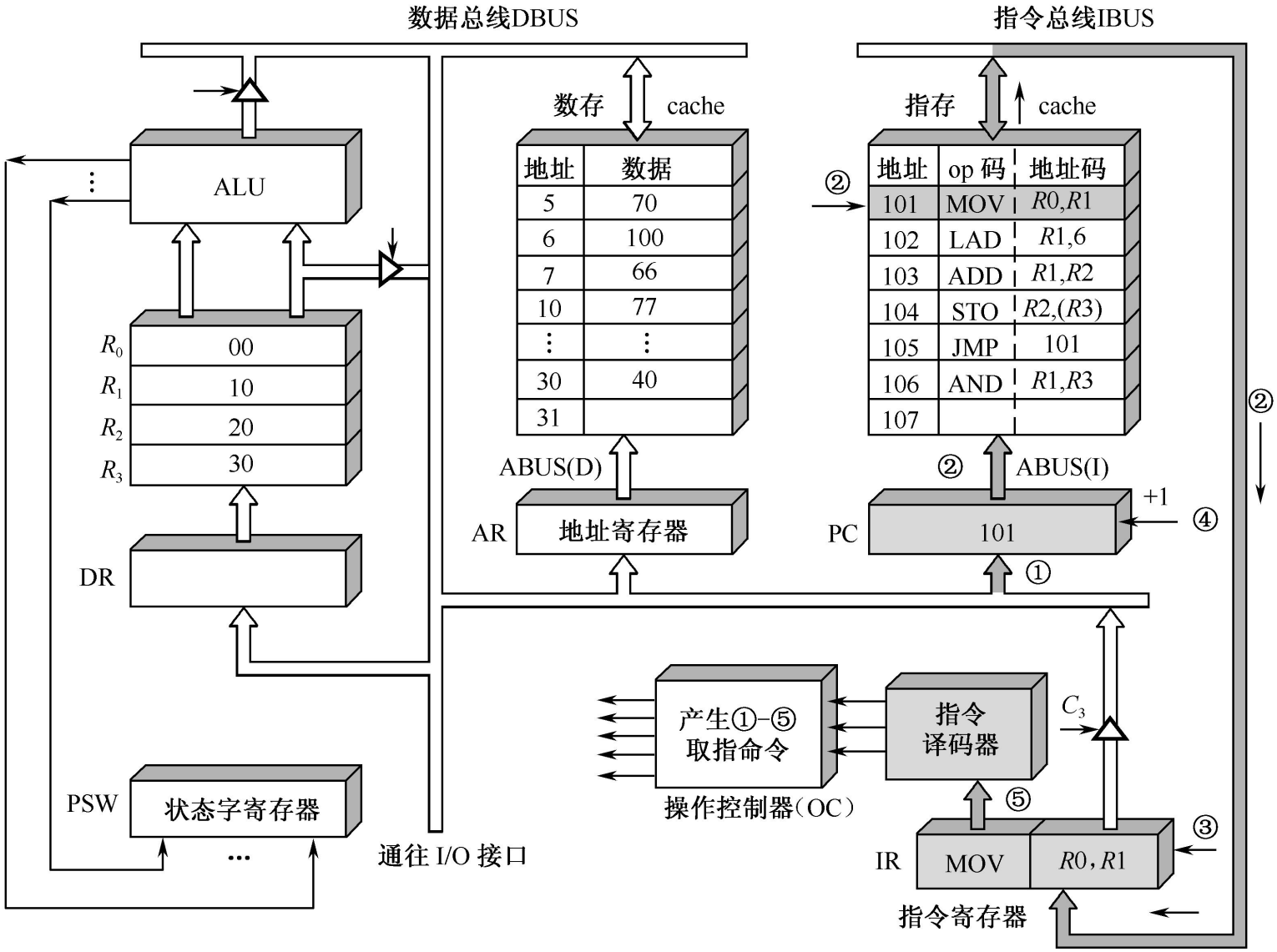
5.5 硬布线控制器

- 设计微操作控制信号的方法和过程是，根据所有机器指令流程图，寻找出产生同一个微操作信号的所有条件，并与适当的节拍电位和节拍脉冲组合，从而写出其布尔代数表达式并进行简化，然后用门电路或可编程器件来实现

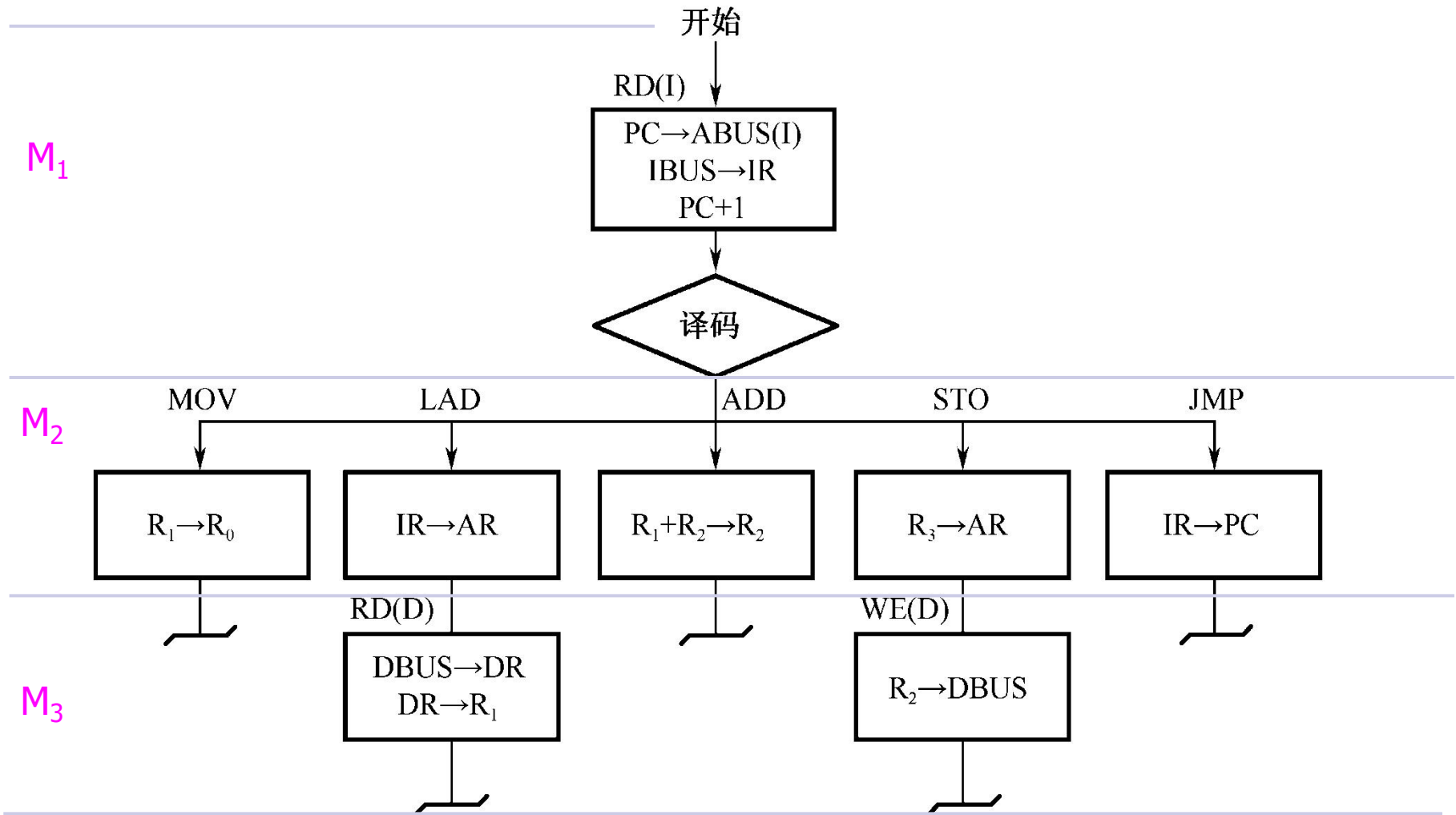
5.5 硬布线控制器

- 例：写出以下操作控制信号RD(I)、RD(D)、WE(D)、LDPC、LDIR、LDAR、LDDR、PC+1、LDR_2的逻辑表达式。其中每个操作控制信号的含义是：
  - RD(I)—指存读命令
  - RD(D)—数存读命令
  - WE(D)———数存写命令
  - LDPC—打入程序计数器
  - LDIR—打入指令寄存器
  - LDAR—打入数存地址寄存器
  - LDDR—打入数据缓冲寄存器
  - PC+1—程序计数器加1
  - LDR_2—打入R_2寄存器

5.5 硬布线控制器


数据总线DBUS
指令总线IBUS
数存
cache
指存
cache
地址
数据
5
70
6
100
7
66
10
77
30
40
31
地址
op 码
地址码
101
MOV
R0,R1
102
LAD
R1,6
103
ADD
R1,R2
104
STO
R2,(R3)
105
JMP
101
106
AND
R1,R3
107
ALU
R0
00
R1
10
R2
20
R3
30
DR
ABUS(D)
AR
地址寄存器
②
ABUS(I)
PC
101
+1
④
①
②
产生①–⑤
取指命令
操作控制器(OC)
指令
译码器
C3
⑤
③
IR
MOV
R0,R1
指令寄存器
PSW
状态字寄存器
通往 I/O 接口

5.5 硬布线控制器


开始
RD(I)
PC→ABUS(I)
IBUS→IR
PC+1
译码
M1
M2
M3
MOV
LAD
ADD
STO
JMP
R1→R0
IR→AR
R1+R2→R2
R3→AR
IR→PC
RD(D)
DBUS→DR
DR→R1
WE(D)
R2→DBUS

5.5 硬布线控制器

- 设M_1、M_2、M_3是三个节拍电位信号；T_1、T_2、T_3、T_4为一个CPU周期中的节拍脉冲信号；MOV、LAD、ADD、STO、JMP分别表示对应机器指令的OP操作码译码输出信号
- 五条指令的微操作控制信号举例：
  - RD(I)=M_1（电位信号）
  - RD(D)=M_3·LAD（电位信号）
  - WE(D)=M_3·T_3·LAD（脉冲信号）
  - LDPC=M_1·T_4+M_2·T_4·JMP
  - LDIR=M_1·T_4
  - LDAR=M_2· (LAD+STO)·T_4
  - LDDR=M_2·T_3(MOV+ADD)+M_3·T_3·LDA
  - PC+1=M_1
  - LDR_2=M_2·T_4·ADD

5.6 传统CPU

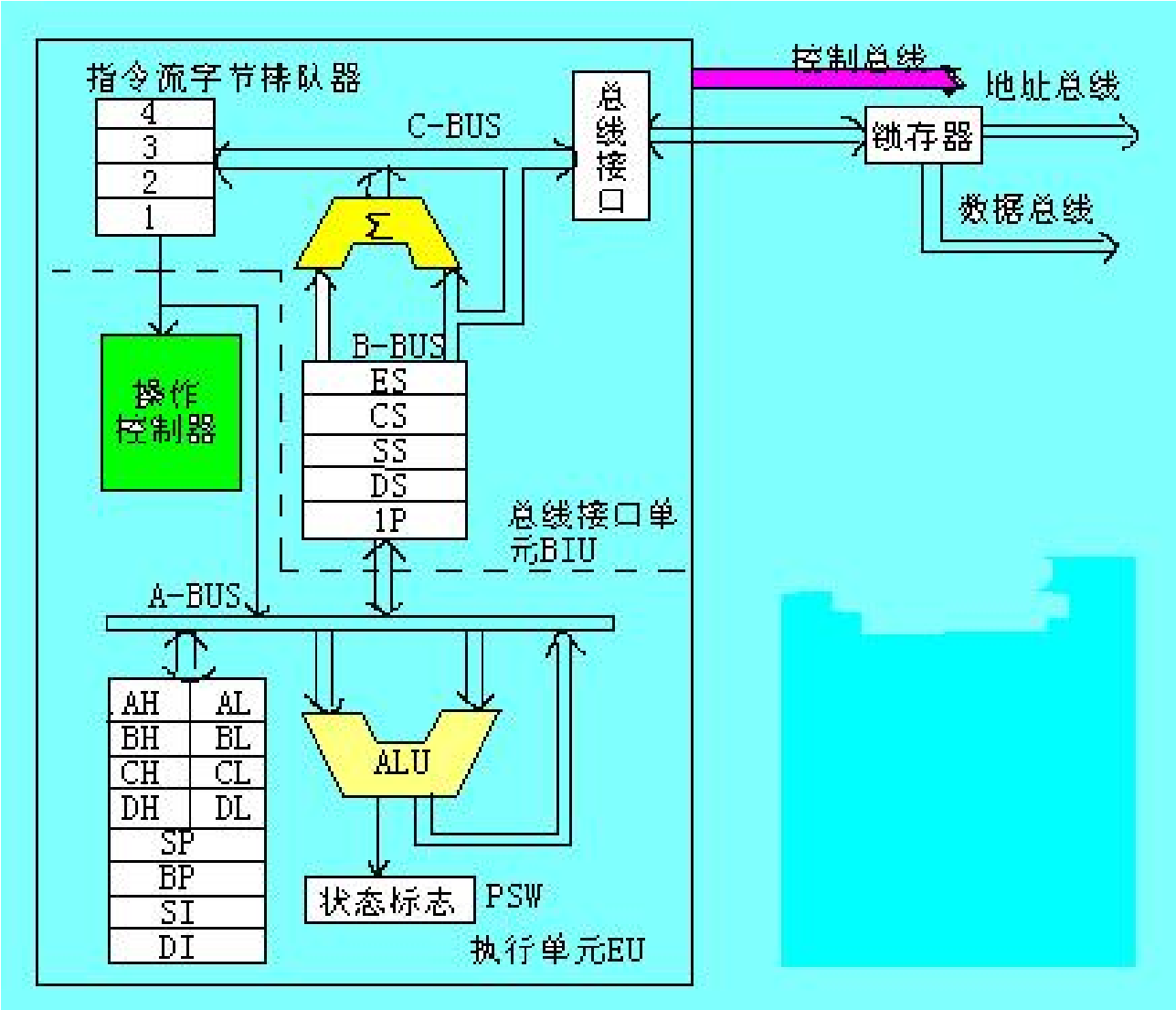
提纲

5.6.1 Intel 8088

- Intel 8088是一种16位微处理器，其内部结构为16位，与外部交换的数据为8位，它可以处理16 位数据（具有16位运算指令，包括乘除法指令），也可以处理8位数据
- 有20条地址线、所以直接寻址能力达到1M字节
- 采用40条引线封装，单相时钟，电源为5V

5.6.1 Intel 8088

- 内部结构图

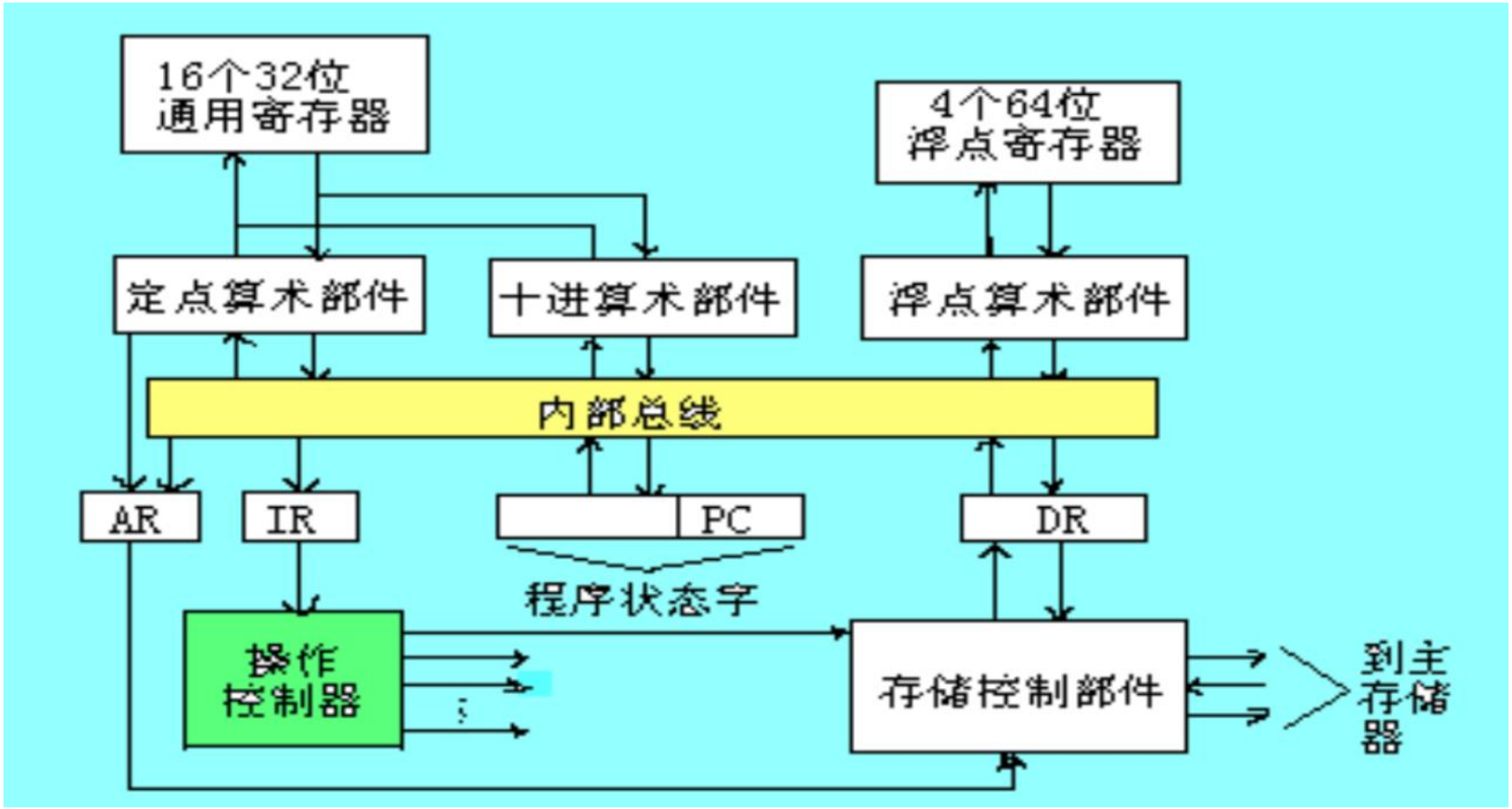


5.6.1 Intel 8088

- CPU从功能上来说分成总线接口单元BIU和执行单元EU两大部分
- BIU负责与存储器和外围设备的接口，即8088 CPU与存储器和外围设备之间的信息传送，都是由BIU进行的。例如，BIU从主存的指定部分取出指令，送到指令流队列中排队；在执行指令时所需要的操作数，也由BIU从主存的指定区域取出，传送到EU部分去执行
- EU部分负责指令的执行
- 取指部分与执行指令部分是独立并行工作的，于是在一条指令的执行过程中，就可以取出下一条（或多条）指令，在指令流队列寄存器中排队，在一条指令执行完以后就可以立即执行下一条指令，减少了CPU为取指令而等待的时间，提高了CPU的利用率，提高了整个系统的运行速度

5.6.1 IBM370

- 内部结构图



5.6.1 IBM370

- IBM370 CPU
  - 1972年，32位
  - ALU部件按功能不同分为三个子部件
    - 定点运算，包括整数计算和有效地址的计算
    - 浮点运算
    - 可变长运算，包括十进制算术运算和字符串操作寄存器结构
  - CPU控制状态
    - 管态：当在执行操作系统的一段程序时，操作系统明确地控制着CPU，这时说 CPU 处于管理状态（简称管态），某些指令只允许在这个状态下执行
    - 当CPU在执行用户程序时，则认为处于正常的解题状态（简称目态）
    - CPU 在任何时刻的状态都是由它的 PSW来说明的

5.7 流水CPU

提纲

5.7.1 并行处理技术

- 并行性（Parrelism）概念
  - 问题中具有可以同时进行运算或操作的特性
  - 例：在相同时延的条件下，用n位运算器进行n位并行运算速度几乎是一位运算器进行n位串行运算的n倍（狭义）

- （广义）含义
  - 只要在同一时刻（同时性）或在同一时间间隔内（并发性）完成两种或两种以上性质相同或不同的工作，他们在时间上相互重叠，都体现了并行性

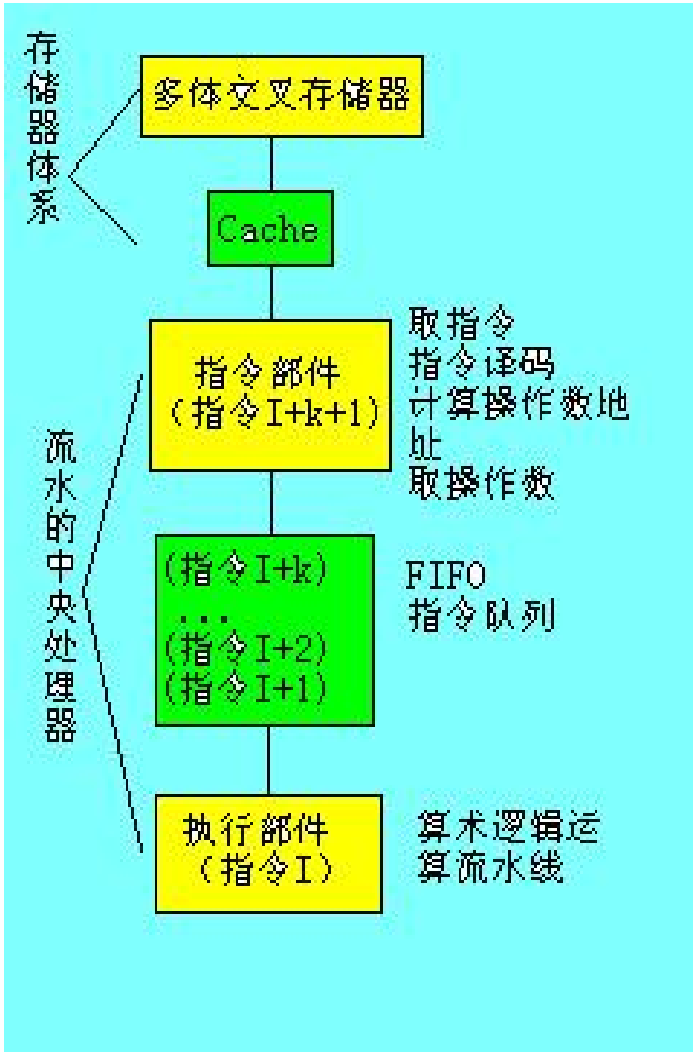
5.7.1 并行处理技术

- 三种形式
  - 时间并行（重叠）：让多个处理过程在时间上相互错开，轮流使用同一套硬件设备的各个部件，以加快硬件周转而赢得速度，实现方式就是采用流水处理部件
  - 空间并行（资源重复）：以数量取胜
    - 它能真正的体现同时性
    - LSI和VLSI为其提供了技术保证
  - 时间+空间并行
    - 时间重叠和资源重复的综合应用
    - Pentium中采用了超标量流水线技术，在一个机器周期中同时执行两条指令

5.7.2 流水CPU的结构

1. 流水计算机的系统组成

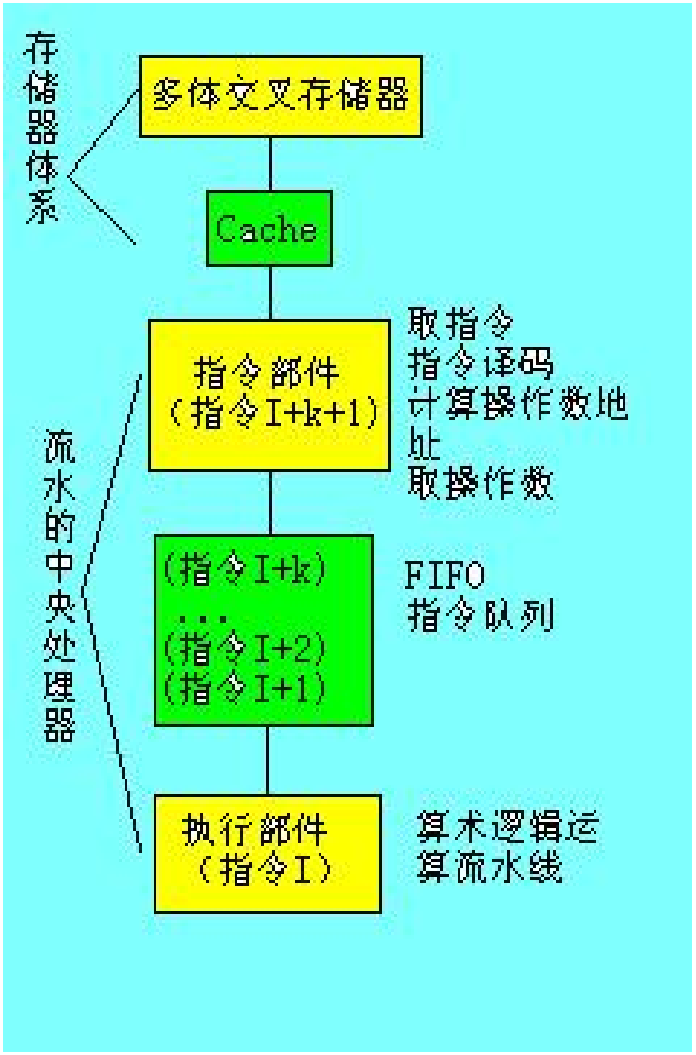
- 存储器体系：主存通常采用多体交叉存储器，以提高访问速度；Cache是一个高速缓冲存储器，用以弥补主存和CPU速度上的差异
- 流水方式CPU：指令部件、指令队列、执行部件
- 指令流水线
  - 指令部件本身又构成一个流水线，即指令流水线，它由取指令、指令译码、计算操作数地址、取操作地址等几个过程段组成



5.7.2 流水CPU的结构

- 指令队列：是一个先进先出（FIFO）的寄存器队列，用于存放经过译码的指令和取来的操作数，它也是由若干个过程段组成的流水
- 执行部件：可以具有多个算术逻辑运算部件，这些部件本身又用流水线方式构成

- 当执行部件正在执行第I条指令时、指令队列中存放着第I+1，I + 2，…，I + k条指令，而与此同时，指令部件正在取第了I + k + 1条指令



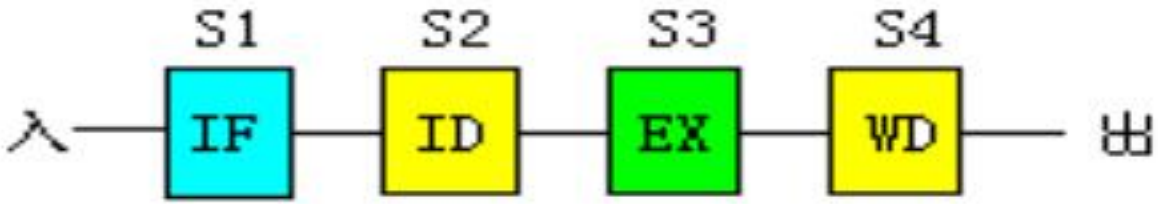
5.7.2 流水CPU的结构

- 执行段的速度匹配问题：通常采用并行的运算部件及部件流水线的工作方式来解决
- 一般采用的方法包括
  - （1）将执行部件分为定点执行部件和浮点执行部件两个可并行执行的部分，分别处理定点运算指令和浮点运算指令
  - （2）在浮点执行部件中，又有浮点加法部件和浮点乘/除部件，它们也可以同时执行不同的指令
  - （3）浮点运算部件都以流水线方式工作

5.7.2 流水CPU的结构

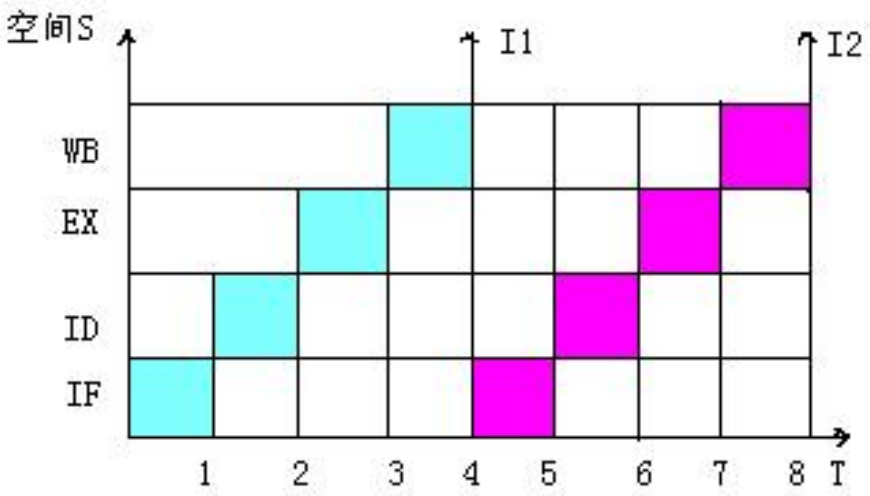
2. 流水CPU的时空图

- 一个指令周期的任务分解：
  - IF（Instruction Fetch取指）
  - ID（Instruction Decode指令译码）
  - EX（Execution执行）
  - WB（Write Back写回）

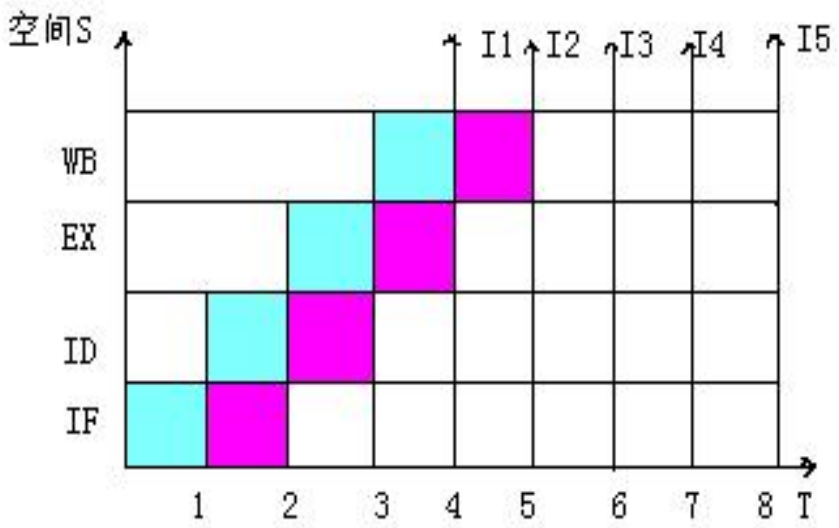


(a) 一个指令流水线过程段

5.7.2 流水CPU的结构



非流水线CPU时空图



标量流水线CPU时空图

- 各个过程段之间设有高速缓冲寄存器，以暂时保存上一过程段子任务处理的结果。在统一的时钟控制下，数据从一个过程段流向相邻的过程段

5.7.2 流水CPU的结构

- 在流水线计算机中，指令周期由几个流水周期组成，流水周期可以由指令执行步骤中最慢的一个步骤的时间来决定
- 流水线中每一级的操作都由统一的时钟来同步，在流水线满负荷时，每一个流水周期可以输出一个结果
- 在流水线计算机中，当任务饱满时，任务源源不断地输入流水线，不论有多少级过程段，每隔一个时钟周期都能输出一个任务。从理论上说，一个具有k级过程段的流水线处理n个任务需要的时钟周期数为：

$$T_k = k + (n - 1)$$

- 其中k个时钟周期用于处理第一个任务，k个周期后、流水线被装满，剩余的n一1个任务只需n-1个周期就完成了

5.7.2 流水CPU的结构

- 如果用非流水线处理器处理这n个任务，则所需时钟周期数为

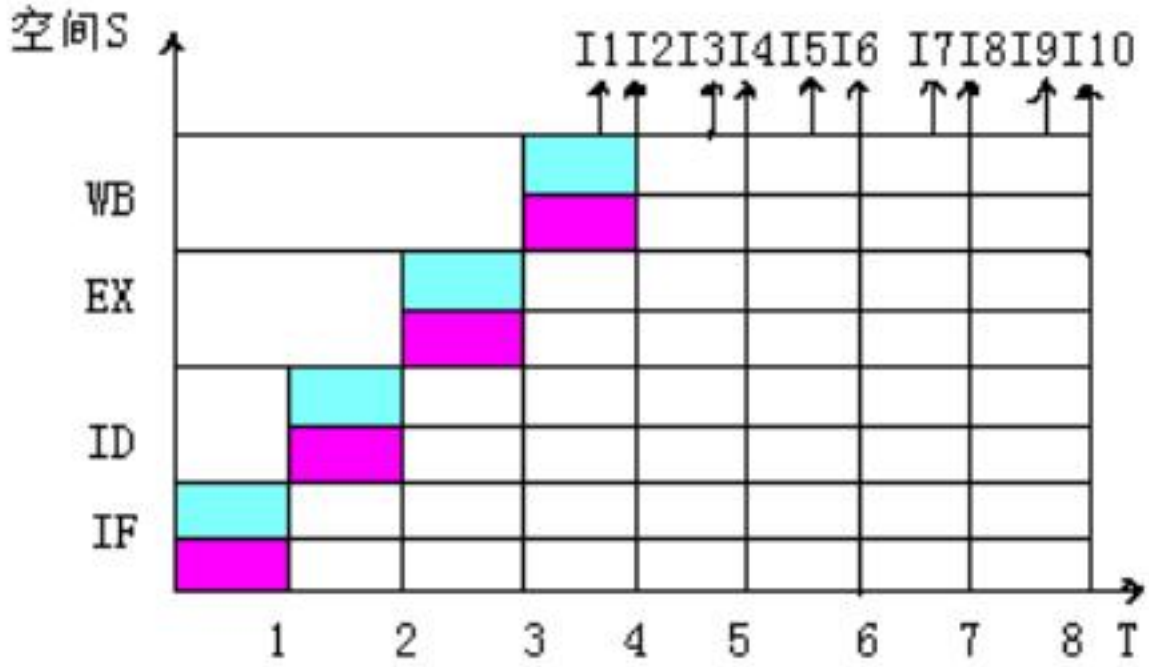
$$T_l = n \times \mathrm{k}$$

- 我们将T_l和T_k的比率定义为k级线性流水处理器的加速比：

$$C_\mathrm{k} = \frac{T_l}{T_k} = \frac{n \times \mathrm{k}}{\mathrm{k}+(n-1)}$$

- 当$\mathrm{n} \gg \mathrm{k}$时，$C_\mathrm{k} \to \mathrm{k}$，这就是说，理论上k级线性流水线处理器几乎可以提高k倍速度
- 但实际上由于存储器冲突、数据相关、程序分支和中断，这个理想的加速比不一定能达到
- 在指令流水中，执行（EX）这一级往往需要的时间最长（例如浮点运算），因此可以进一步分为多级的流水线

5.7.2 流水CPU的结构



超标量流水线CPU时空图

- 具有两条以上的指令流水线
- 上图中流水线满载时，每一个时钟周期可以执行2条指令
- 采用时间和空间并行技术

5.7.2 流水CPU的结构

2. 流水线（Pipelining）CPU的分类

- 按并行等级分为:
  - 算术流水线：运算操作步骤并行，如流水线加法器、流水线乘法器、流水线快速傅立叶变换器等。现代计算机中已广泛采用了流水线的算术运算器
  - 指令流水线：指令步骤并行，例如将指令流的外理过程划分为取指、译码、取操作数，执行这几个并行处理的过程段，目前，几乎所有的高性能计算机都采用了指令流水线
  - 处理机流水线（宏流水线）：指程序步骤的并行，由一串级联的处理机构成流水线的各个过程段，每台处理机负责某一特定的任务，数据流从第一台处理机输入，经处理后被送入与第二台处理机相联的缓冲存储器中，第二台处理机从该存储器中取出数据进行处理、然后传送给第三台处理机，如此串联下去

5.7.3 流水线中的主要问题

1、资源相关

- 资源相关：多条指令进入流水线后在同一机器时钟周期内争用同一个功能部件
  - 解决办法：后边指令拖一拍再推进或增设一个功能部件

指令 \ 时钟	1	2	3	4	5	6	7	8
I1 (Load)	IF	ID	EX	MEM	WB			
I2		IF	ID	EX	MEM	WB		
I3			IF	ID	EX	MEM	WB	
I4				IF	ID	EX	MEM	WB
I5					IF	ID	EX	MEM

  - 增设一个存储器，将指令和数据分别放在两个存储器中

5.7.3 流水线中的主要问题

2、数据相关问题

- 如果必须等前一条指令执行完毕后，才能执行后一条指令，那么这两条指令就是数据相关的
- 例：两条指令发生数据相关冲突RAW(Read After Write)

```
ADD     R1,   R2,   R3        R2+R3-->R1
SUB     R4,   R1,   R5        R1-R5-->R4
AND     R6,   R1,   R7        R1^R7-->R6
```

时钟 / 指令	1	2	3	4	5	6	7	8
ADD	IF	ID	EX	MEM	WB			
SUB		IF	ID	EX	MEM	WB		
AND			IF	ID	EX	MEM	WB	

5.7.3 流水线中的主要问题

- 解决方法：流水CPU的运算器中特意设置若干运算结果缓冲器，暂时保留运算结果，以便于后继指令直接使用，称为“向前”或定向传送技术

5.7.3 流水线中的主要问题

- 例：流水线中有三类数据相关冲突：写后读（RAW）相关；读后写（WAR）相关；写后写（WAW）相关。判断以下三组指令各存在哪种类型的数据相关。

(1) I1 ADD R1，R2，R3 ；（R2）+（R3）->R1

I2 SUB R4，R1，R5 ；（R1）-（R5）->R4

(2) I3 STO M（x），R3 ；（R3）->M(x)，M(x)是存储器单元

I4 ADD R3，R4，R5 ；（R4）+（R5）->R3

(3) I5 MUL R3，R1，R2 ；（R1）×（R2）->R3

I6 ADD R3，R4，R5 ；（R4）+（R5）->R3

5.7.3 流水线中的主要问题

- 第（1）组指令中，I1指令运算结果应先写入R1，然后在I2指令中读出R1内容。由于I2指令进入流水线，变成I2指令在I1指令写入R1前就读出R1内容，发生RAW相关。
- 第（2）组指令中，I3指令应先读出R3内容并存入存储单元M（x），然后在I4指令中将运算结果写入R3。但由于I4指令进入流水线，变成I4指令在I3指令读出R3内容前就写入R3，发生WAR相关。
- 第（3）组指令中，如果I6指令的加法运算完成时间早于I5指令的乘法运算时间，变成指令I6在指令I5写入R3前就写入R3，导致R3的内容错误，发生WAW相关。

5.7.3 流水线中的主要问题

3、控制相关

- 控制相关冲突是由转移指令引起的。当执行转移指令时，依据转移条件的产生结果，可能顺序执行下一条指令；也可能转移到新的目标地址取指令，从而使流水线发生断流。
- 解决方式：
  - 延迟转移法：由编译程序重排指令序列来实现。基本思想是“先执行再转移”，即发生转移取时并不排空指令流水线，而是让紧跟在转移指令之后已进入流水线的少数几条指令继续完成。如果这些指令是与转移指令结果无关的有用指令，那么延迟损失时间片正好得到了有效的利用
  - 转移预测法：硬件方法来实现，依据指令过去的行为来预测将来的行为。通过使用转移取和顺序取两路指令预取队列器以及目标指令cache,可将转移预测提前到取指阶段进行，以获得良好的效果

5.7.3 流水线中的主要问题

4、流水线的拥挤问题

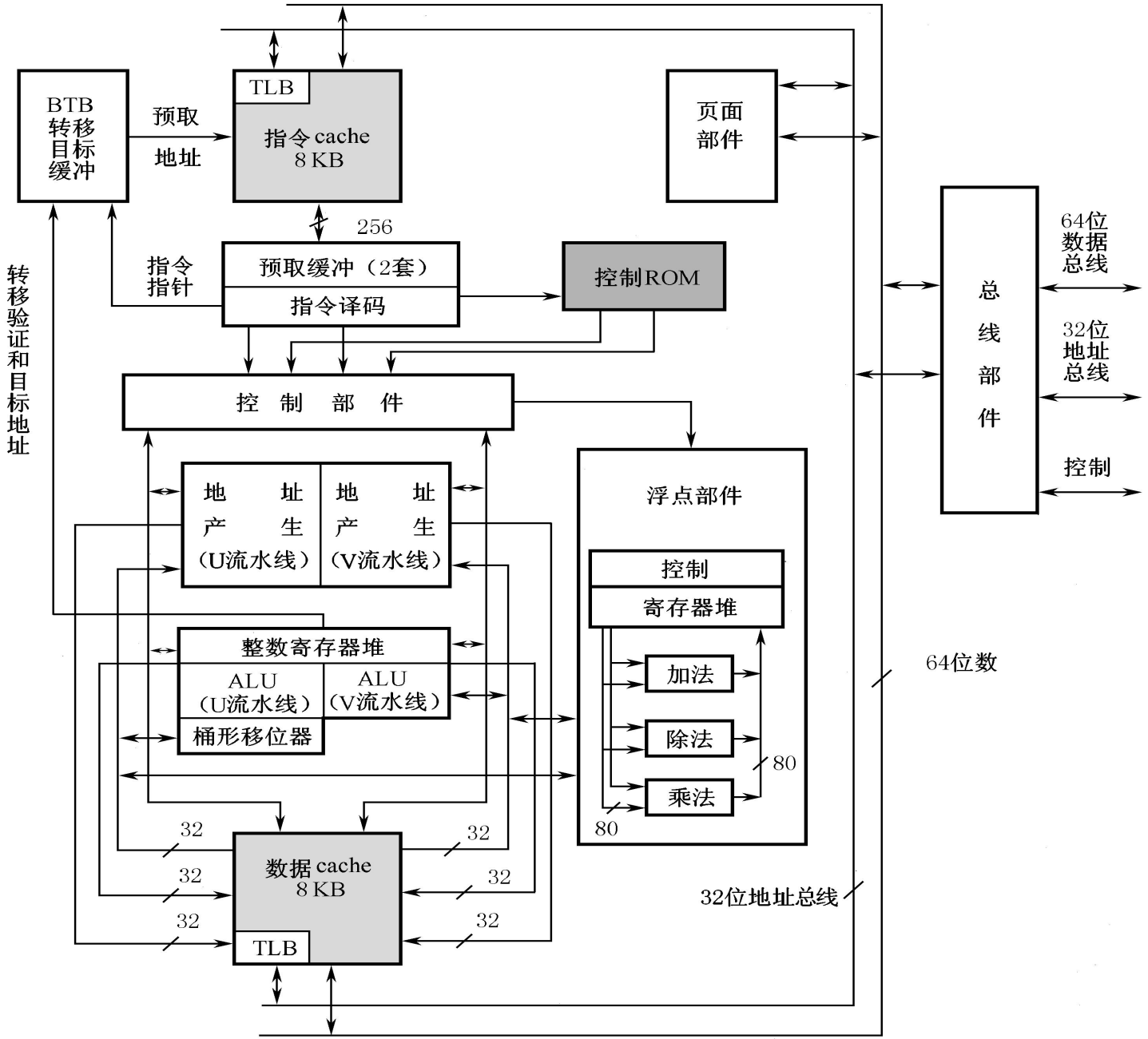
- 在流水线中，各个过程段的处理速度通常是不完全相同的，当某一段的处理速度很慢时，就会成为流水线的瓶颈
- 解决的办法：
  - 把瓶颈段进一步分成几个过程段，或是利用资源重复原理提高瓶颈段的处理速度，这样流水线中的拥挤状况就会得到缓解，从而提高流水线吞吐率
  - 设置缓冲器来弥补过程段之间的速度差异。通常，在处理时间不固定的过程段的前后都设置缓冲器，这样，从前面的缓冲器中能够得到源源不断的信息（指令或操作数），处理后的结果也能够及时存入后面的缓冲器中

5.7.4 Pentium CPU

- Pentium CPU （第一代）
  - 1989年初，0.8um工艺，310万晶体管
  - 有60M和66MHz外频两种版本
  - 5V电压，功耗20W
  - 超标量流水线结构
    - 486有一条流水线
    - Pentium有U和V两条指令流水线
      - ✓ U流水线可以执行所有的整数和浮点指令
      - ✓ V流水线可以执行简单的整数和FXCH浮点指令（浮点交换）
  - 双重分离式Cache（指令和数据），减少等待和搬移数据时间
  - 32位CPU，外部数据总线宽度64位，外部地址总线宽度36位

5.7.4 Pentium CPU

- 操作控制器采用硬布线控制和微程序控制相结合的方式。大多数简单指令用硬布线控制实现，在一个时钟周期内执行完毕。对微程序实现的指令，也在2—3个时钟周期内执行完毕
- 非固定长度指令格式，9种寻址方式，191条指令，兼具有RISC和CISC特性，不过我们还是将其看成CISC
- 提供了更加灵活的存储器寻址结构，可以支持传统的4k大小的页面，也可以支持4M大小的页面
- 动态转移预测技术
  - BTB（Branch Target Buffer，转移目标缓冲器）
- TLB（Translation Lookaside Buffer，后备缓冲器，后援缓冲器），里面存放的是一些页表文件（虚拟地址到物理地址的转换表），又称为快表技术，用来缓存最近访问的虚拟页


BTB
转移
目标
缓冲
预取
地址
TLB
指令cache
8 KB
页面
部件
256
指令
指针
预取缓冲（2套）
指令译码
控制ROM
转移验证和目标地址
控 制 部 件
总
线
部
件
64位
数据
总线
32位
地址
总线
控制
地 址
产 生
(U流水线)
地 址
产 生
(V流水线)
浮点部件
控制
寄存器堆
整数寄存器堆
ALU
(U流水线)
ALU
(V流水线)
桶形移位器
加法
除法
乘法
80
80
64位数
32
32
32
32
32
32
数据cache
8 KB
TLB
32位地址总线

5.8 RISC CPU

5.8 RISC CPU

- 特点
  - 采用流水线技术
  - 简单而统一格式的指令译码
  - 大部分指令可以单周期执行
  - 只有LOAD/STORE可以访问存储器
  - 简单的寻址方式
  - 采用延迟转移技术
  - 采用LOAD延迟技术
  - 三地址指令格式
  - 较多的寄存器
  - 对称的指令格式

5.8 RISC CPU

- 实例 MC88110
  - Motorola公司的产品
  - CPU结构框图（见下图）
  - 12个执行功能部件
    - 取数/存数（读写）部件、整数运算部件（2个）、浮点加法部件、乘法部件、除法部件、图形处理部件（2个）、位处理部件、用于管理流水线的超标量指令派遣/转移部件
  - 3个Cache（指令，数据和目标指令（保存转移目标指令））
  - 两个寄存器堆
    - 通用寄存器堆，用于整数和地址指针
    - 扩展寄存器堆，用于浮点数
  - 六条80位宽的内部总线相连

5.8 RISC CPU

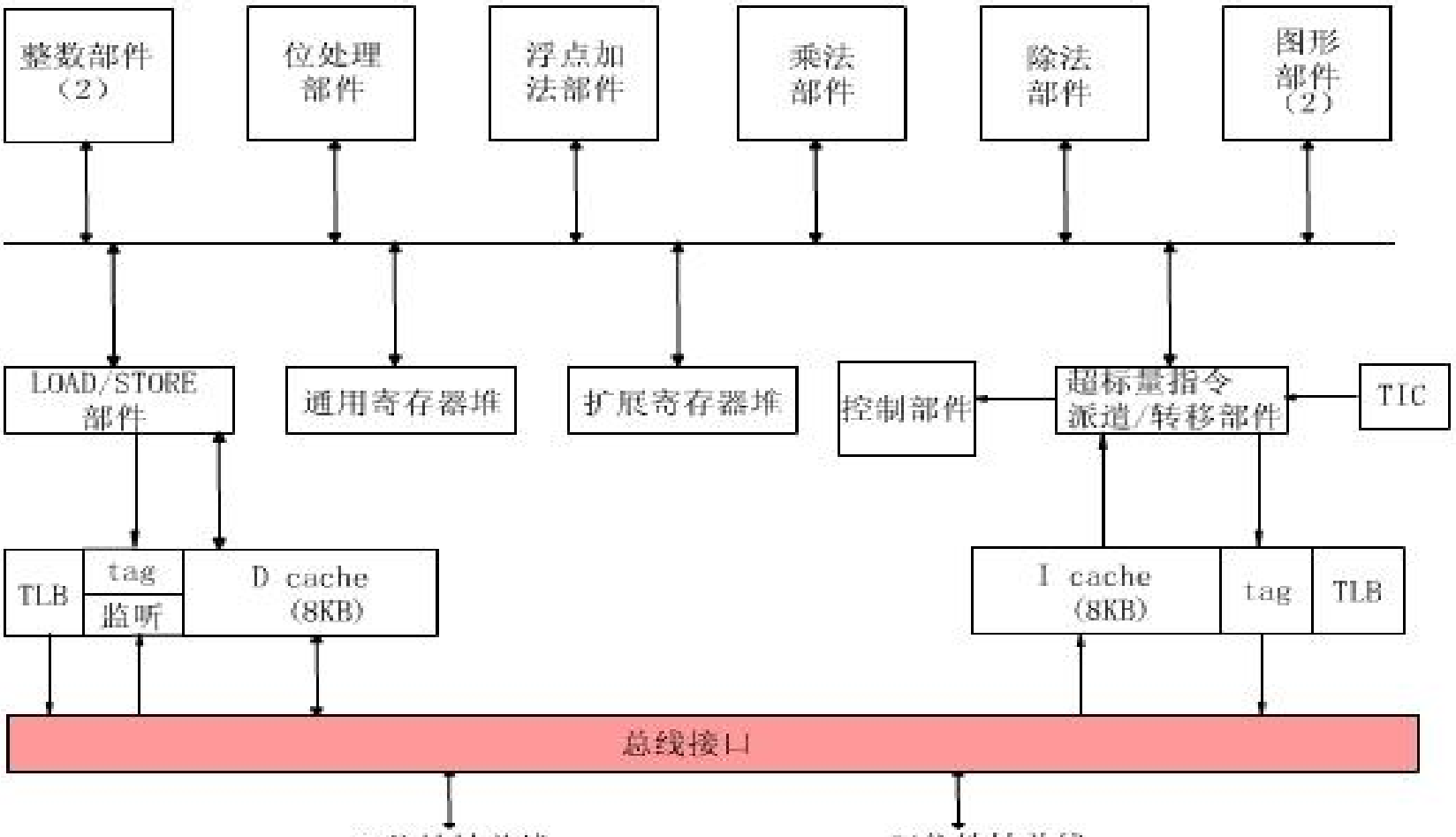


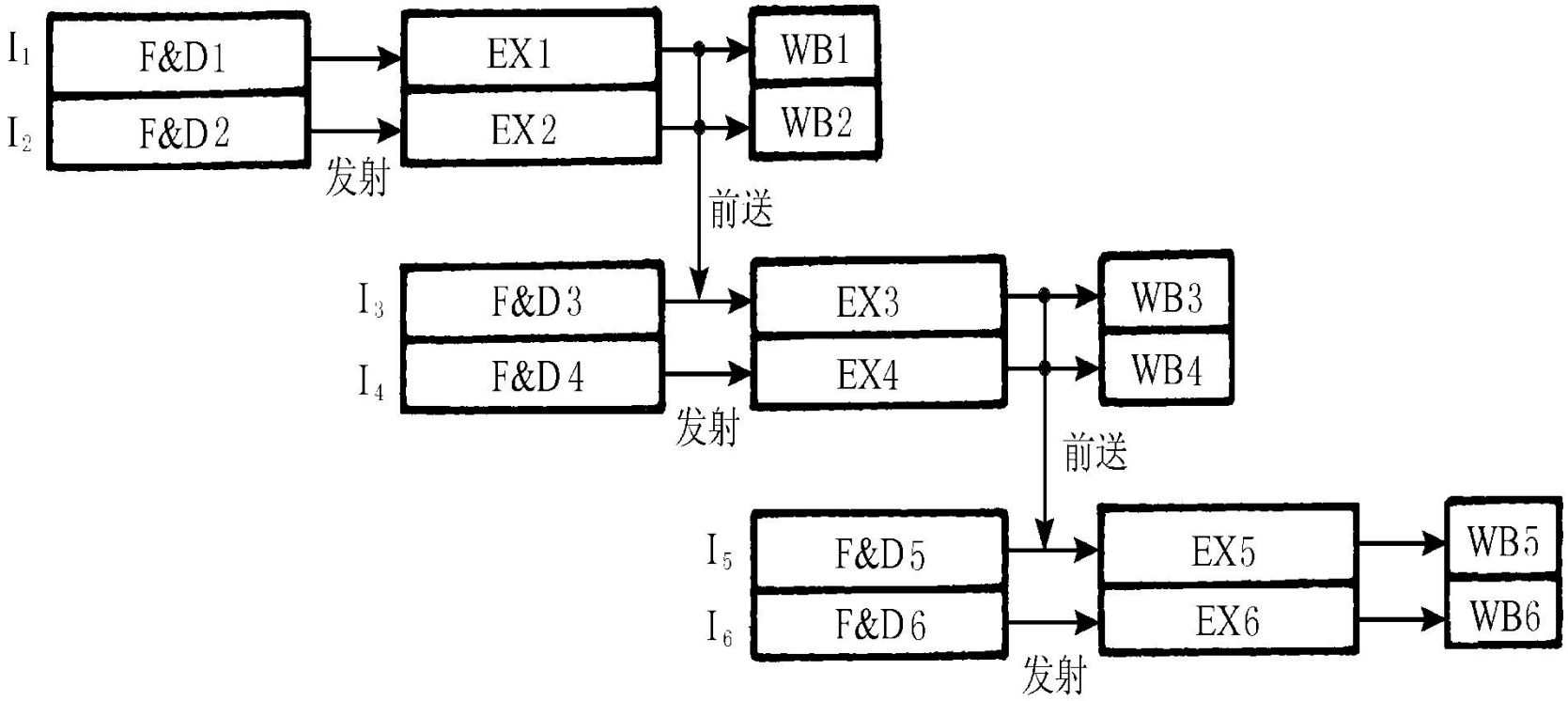
图5.41 88110处理器结构框图

5.8 RISC CPU

- MC88110的指令流水线——超标量流水线CPU
  - F&D：取指和译码段需要一个时钟周期，取一对指令并译码，并从寄存器堆取操作数，然后判断是否把指令发射到EX段
  - EX：执行段，大都只需要一个时钟周期，支持经定向电路提前传送到ALU，可直接被当前进入EX的指令所使用
  - WB：写回段，只需要时钟周期的一半

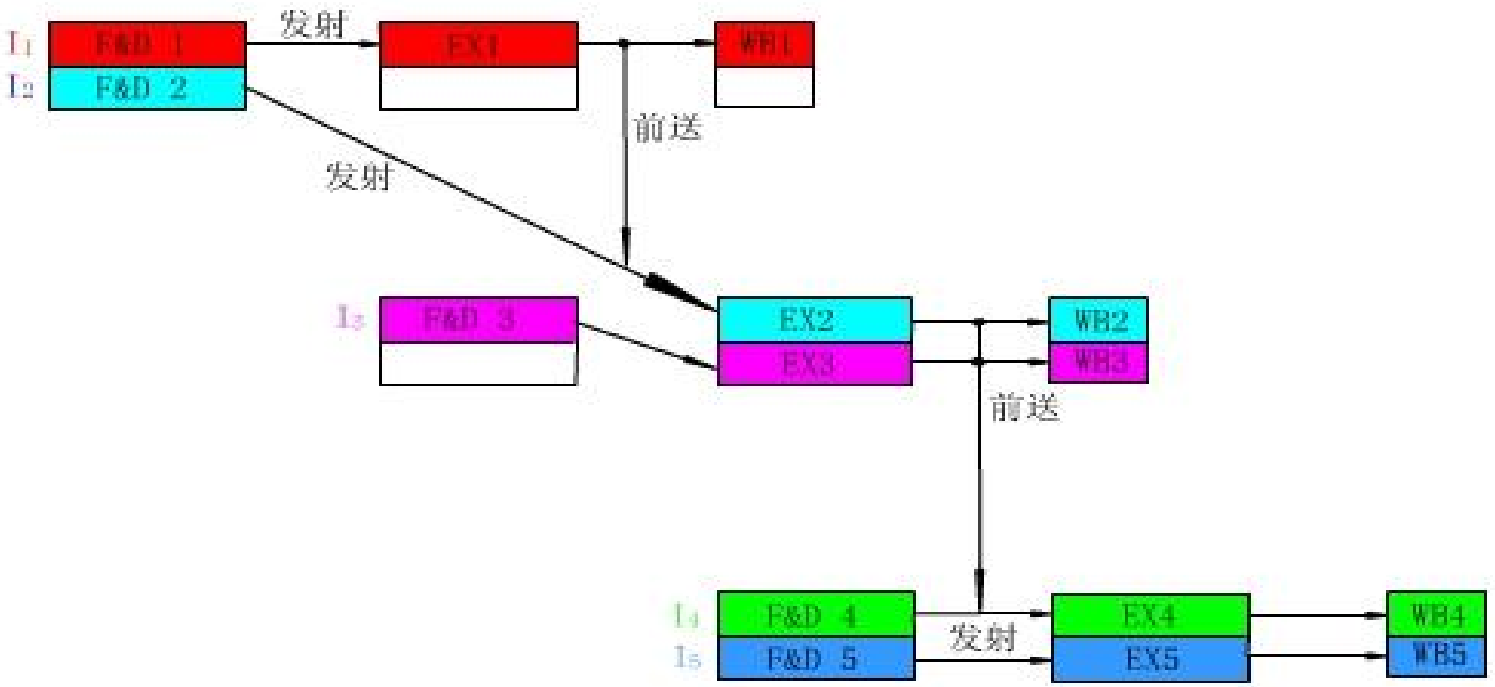
5.8 RISC CPU

- 指令动态调度策略（按序发射，按序完成）
  - 按序发射
    - 取两条指令，配对发送，一个周期可以有两条指令执行完毕



5.8 RISC CPU

- 第一条指令由于资源相关或数据相关,则这两条指令都不发射
- 若第一条指令能发射,第二条不能发射，只发射第1条指令到EX段，第二条指令等待并新取一条指令与之配对等待发射



超标量流水线指令配对情况

5.8 RISC CPU

- 几个问题:

- 怎样判断能否发射呢?
  - 可以采用计分牌的方法
- 如何保证按序完成?
  - FIFO指令队列

5.8 RISC CPU

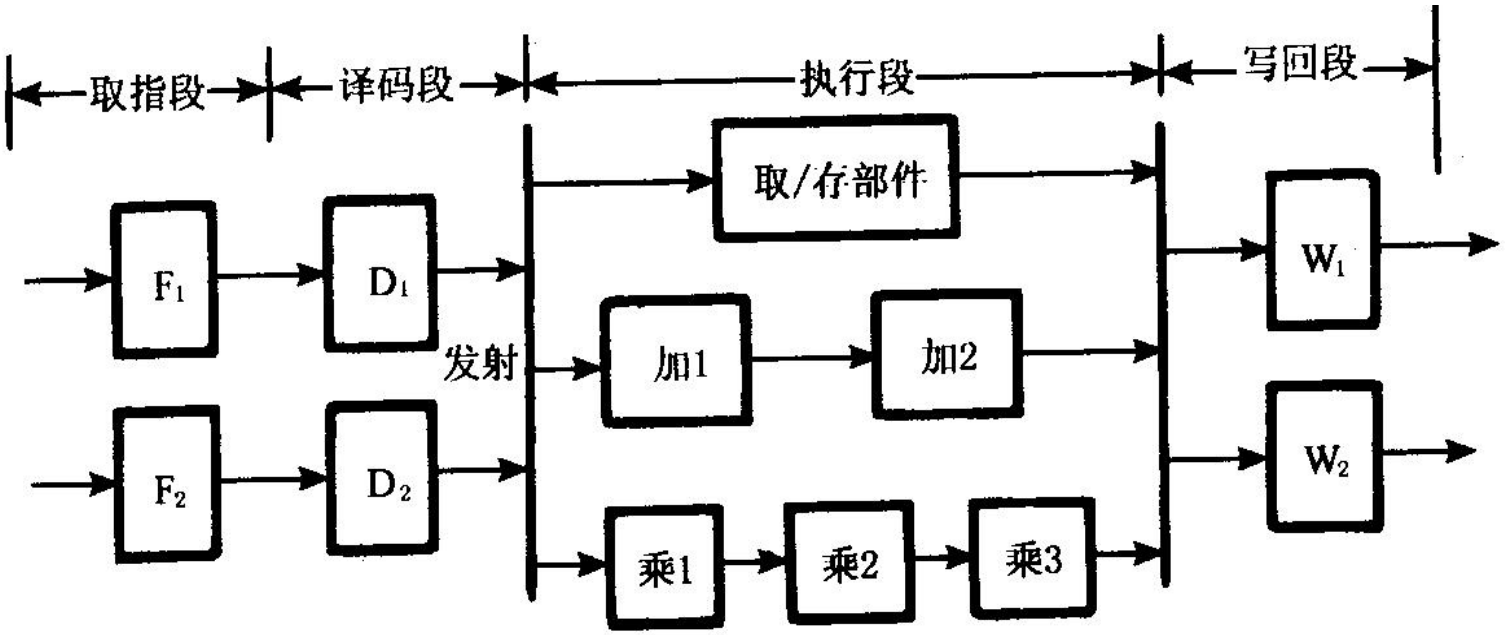
- 计分牌：
  - 计分牌是一个位向量，每一位对应寄存器堆中的一个寄存器
  - 指令发射时，目的寄存器在计分牌中相应位置1，写回后清0
  - 判断指令可否发射的条件是：
    - 该指令的所有目的寄存器、源寄存器在向量位中对应的位都为0
    - 否则，等待这些位清除

5.8 RISC CPU

- FIFO队列
  - FIFO队列称为历史缓冲器，每当一条指令发射后，副本传入FIFO队列队尾
  - 只有当前面的指令执行完毕，才到达队首，
  - 执行完毕后，离开队列

5.8 RISC CPU

- 例：超标量流水线结构如下
  - 有四个段
  - F、D、W需1个时钟周期，E有1、2、3种时钟周期



(a)超标量流水模型结构

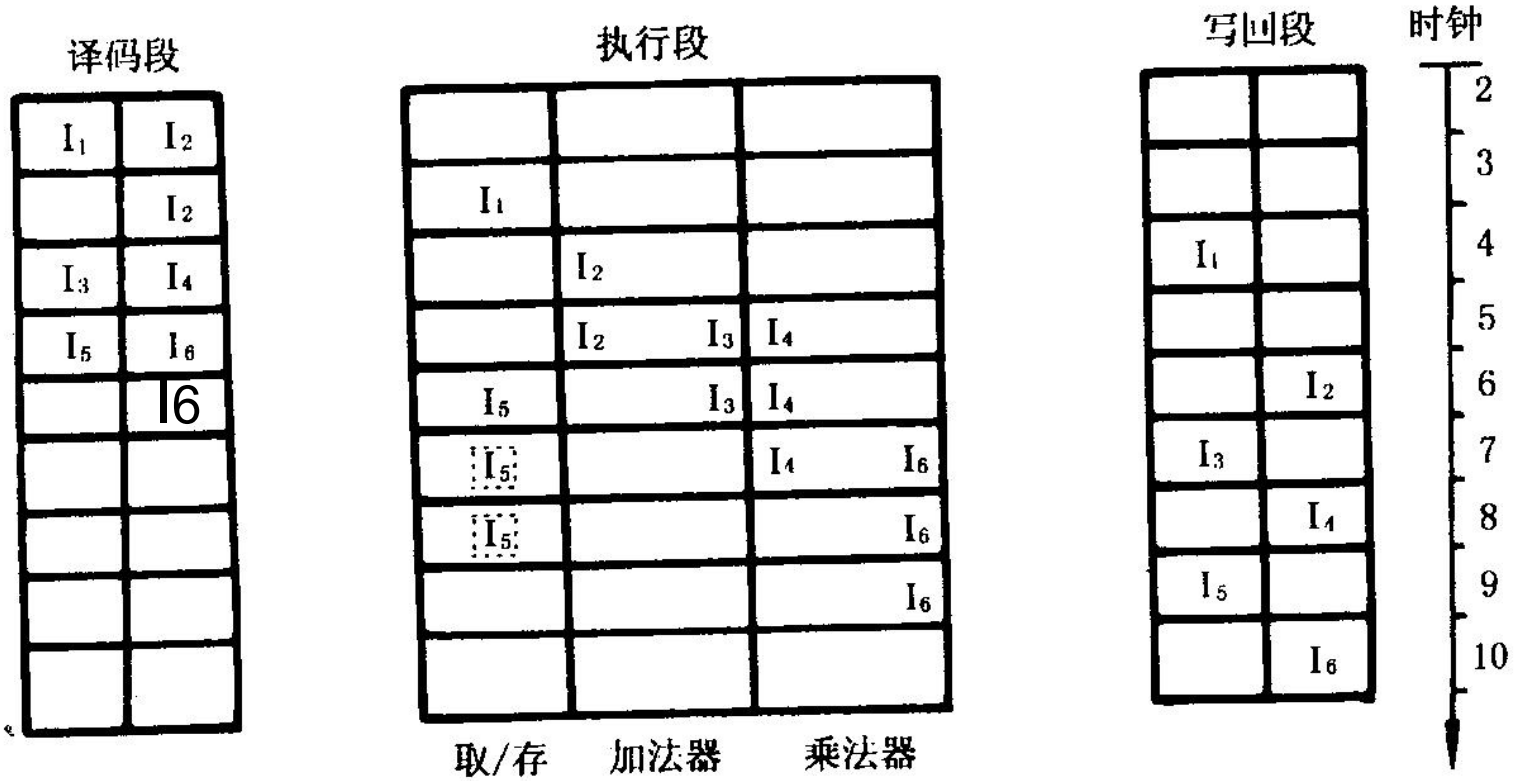
5.8 RISC CPU

I1	LDA	R1,	A	RAW
I2	ADD	R2,	R1	
I3	ADD	R3,	R4	WAR
I4	MUL	R4,	R5	
I5	LDA	R6,	B	WAW
I6	MUL	R6,	R7	

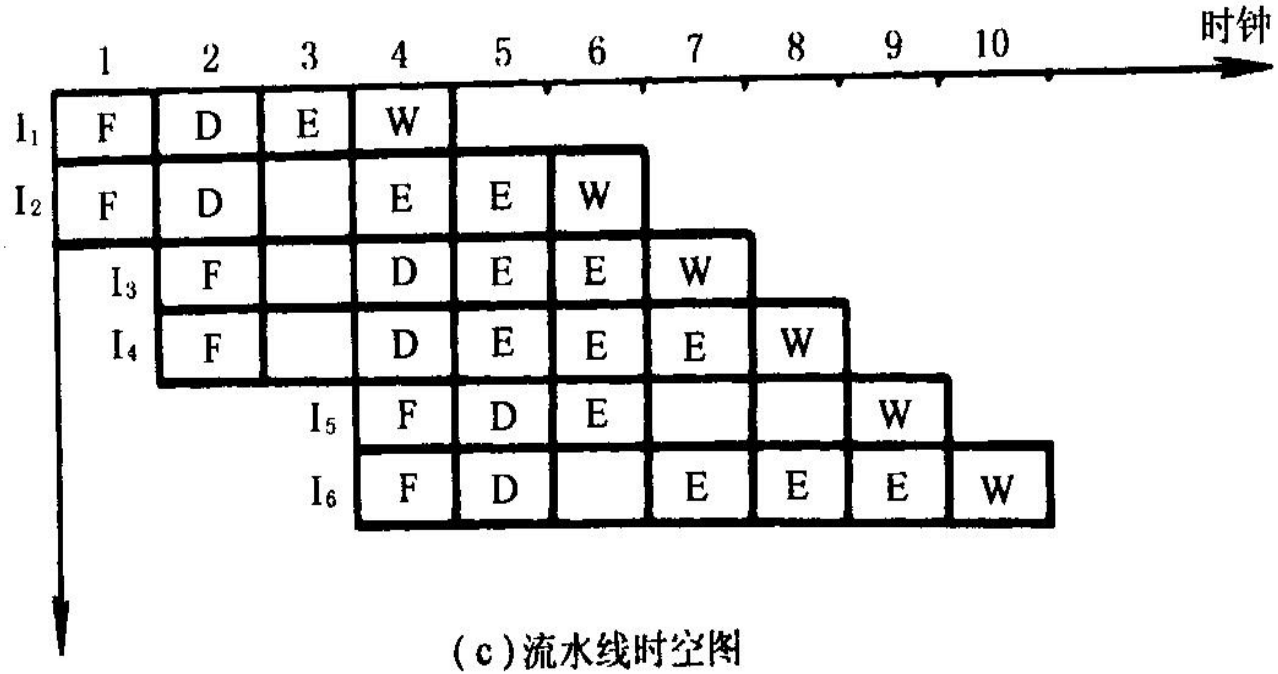
画出按序发射按序完成各段推进情况图

画出按序发射按序完成的流水线时空图

5.8 RISC CPU



5.8 RISC CPU



（c）流水线时空图

5.9 多媒体CPU

5.9 多媒体CPU

- 多媒体概念
  - 指利用计算机来综合、集成地处理文字、图形、图像、声音、视频、动画等媒体，从而形成的一种全新的信息传播和处理的计算机技术。主要特征：
    - 信息表示的数字化
    - 处理的集成性
    - 系统的交互性

5.9 多媒体CPU

- 主要技术问题——压缩和解压缩技术
- 静态
  - 640*480的256色图像约占640*480*1B=307200B = 300K
  - 640*480的24Bit彩色图像约占640*480B*3=921600B=900K
- 动态
  - 每秒钟30桢（播放256色）
  - 则每秒钟处理300K*30=9M，而ISA总线的传输率只有5MBPS
- 结论：多媒体信息量大，给信息处理和传输带来了困难

5.9 多媒体CPU

- 解决方法：
  - 压缩技术
    - JPEG（Joint Photographic Experts Group）
    - MPEG（Moving Picture group）
  - 软件技术
    - 多媒体OS
    - 多媒体处理软件
  - 硬件技术
    - MMX（多媒体扩展技术）
    - 动态执行技术

5.9 多媒体CPU

- MMX（多媒体扩展技术）
  - MMX是Intel为增强处理器的多媒体能力而提出的解决方案，它是57个多媒体指令集合。这些指令是为高效地处理视频、声音和图形数据而专门设计的

5.9 多媒体CPU

- 动态执行技术
  - 通过预测程序流来调整指令的执行，分析程序的数据流来选择指令执行的最佳顺序
  - 涉及数据相关性、指令调度法、转移预测法、指令的发射顺序和完成顺序等流水技术
  - 适合MMX指令的加速执行
  - 实现的关键技术：
    - 取消“取指”、“执行”的时间顺序，采用指令缓冲池（开辟一个较长的指令窗口，乱序执行）
    - 允许执行单元在一个较大的范围内调遣和执行译码过的指令

- 分析微程序控制器和硬布线控制器的异同点。在微程序控制器中，微程序计数用uAR来代替，试间是否可以用具有计数功能的存储器地址寄存器MAR来代替提序计数器PC？为什么？

类别 / 对比项目	微程序控制器	硬布线控制器
工作原理	微操作控制信号以微程序的形式存放在控制存储器中，执行指令时读出即可	微操作控制信号由组合逻辑电路根据当前的指令码、状态和时序，即时产生
执行速度	慢	快
规整性	较规整	烦琐、不规整
应用场合	CISC CPU	RISC CPU
易扩充性	易扩充修改	困难

- 不能。因为控存中只有微指令，为了降低成本，可以用具有计数功能的微地址寄存器(μMAR)来代替μPC 。而主存中既有指令又有数据,它们都以二进制代码形式出现，取指令和数据时地址的来源是不同的。取指令:(PC)→ MAR；取数据：地址形成部件→ MAR，所以不能用 MAR代 替PC 。

作业

- 第四章：7、8、9
- 第五章：6、10、16