第五章：中央处理器

▼ 重点概念总结

指令系统：所有机器指令的集合

1条机器指令：1个微程序

微程序：若干条微指令的组合

微指令：一系列微命令

微命令=微操作

微命令：构成控制序列的最小单位

微操作：微命令的操作过程，执行部件中最基本的操作。

程序→机器指令→微程序→微指令→微命令（微操作）

▼ CPU

功能：**自动完成取指指令和执行指令的任务**

组成：运算器ALU（算术运算&逻辑运算）；Cache（数据和指令存取）；控制器（指令执行控制）

▼ 主要寄存器

1. 数据缓冲寄存器（DR） **运算器**
   - CPU和内存、外部设备的信息中转站
   - 补偿速度差
2. 指令寄存器（IR） **控制器**
   - 保存当前正在执行的一条指令
   - 指令译码器（ID）：译码**操作码**字段
3. 程序计数器（PC） **Cache**
   - 存放将要执行的下一条指令的地址
4. 地址寄存器（AR） **Cache**
   - 保存当前CPU访问的内存单元的地址
5. 通用寄存器（R0-R3） **运算器**

- ALU的工作区

6. 程序状态字（PSW） **运算器**
   - 进位标志（C），溢出标志（V），零标志（Z），负标志（N）等

▼ 操作控制器和时序产生器 控制器

操作控制器：

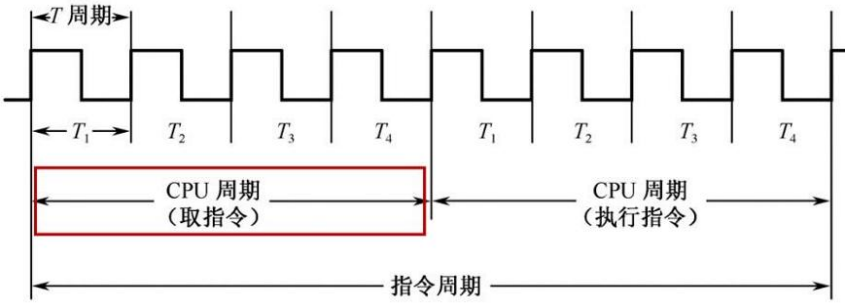
1. 硬布线控制器（时序逻辑）
2. 微程序控制器（存储逻辑）

▼ 指令周期

指令周期：取出、分析（译码）、执行指令所需的全部时间

机器周期（CPU周期）：主存工作周期（存取周期）

时钟周期：控制CPU操作的最小时间单位



1. 单周期CPU

   所有指令执行时间都以**最长时间的指令**为准

2. 多周期CPU

   取指→译码→执行→回写

   时钟周期短，不同指令周期数可不同

   灵活性高

▼ 时序系统

▼ 多级时序信号体系

时序产生器作用：**CPU通过时序信号/周期信息辨认取指周期和执行周期**

1. 硬布线控制器——三级体制

   主状态周期>节拍电位（CPU周期）>节拍脉冲

2. 微程序控制器——二级体制

   节拍电位（CPU周期）>节拍脉冲（T周期）

   节拍脉冲的时间间隔可以不相等

▼ 时序信号控制方式

同步控制

1. 定长机器周期
2. 不定长机器周期
3. 中央控制与局部控制

异步控制

不适用于超高速率场景

联合控制方式（微程序控制器采用）

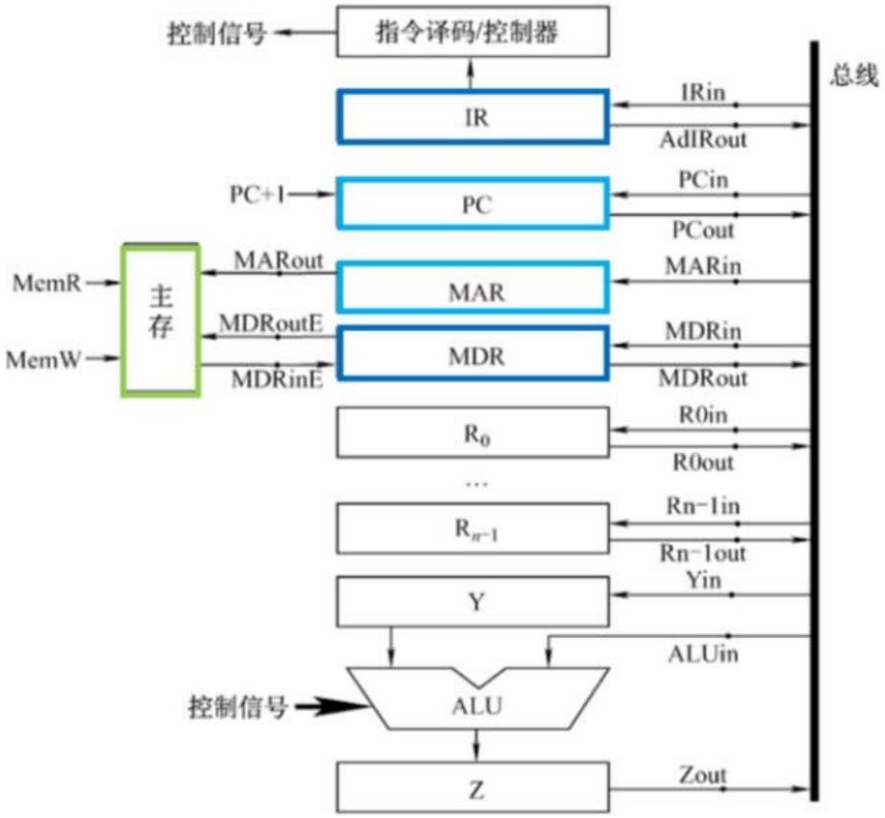
主要同步，耗时长的（IO指令）异步

▼ 数据通路

内部数据通路：寄存器之间传送信息的通路

外部数据通路：借助总线，连接寄存器（MAR、MDR）与存储器和I/O模块

▼ 数据通路例题



写出 ADD[R0]，R1指令的控制信号。功能[R0]+R1→[R0]

取指：

时序	微操作	有效控制信号
1	(PC)→MAR	PCout，MARin
2	M(MAR)→MDR	MDRinE，MemR，MARout
3	(MDR)→IR	MDRout，IRin
4	指令译码	
5	(PC)+1→PC	

执行

时序	微操作	有效控制信号
1	(R0)→MAR	R0out，MARin
2	M(MAR)→MDR	MemR，MARout，MDRinE
3	(MDR)→Y	MDRout，Yin
4	(R1)+(Y)→Z	R1out，ALUin，CU向ALU发控制信号
5	(Z)→MDR	Zout，MDRin

时序	微操作	有效控制信号
6	(MDR)→M(MAR)	MemW，MDRoutE，MARout

▼ 典型周期指令

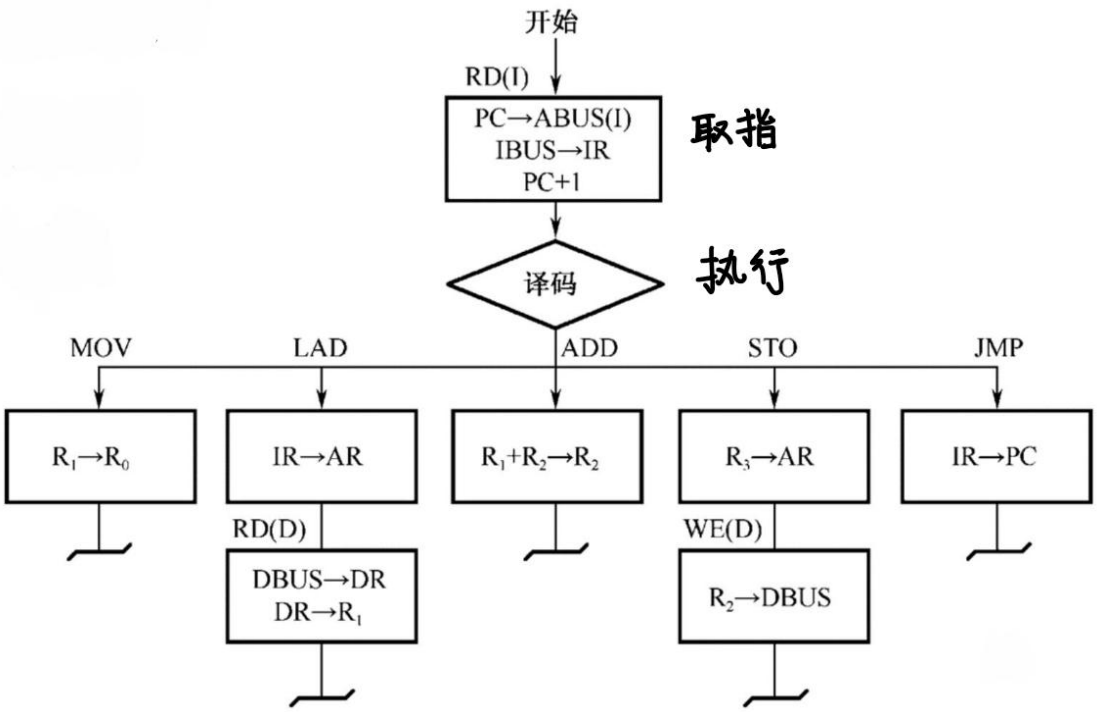
MOV指令、LAD指令、ADD指令、STO指令、JMP指令

▼ 方框图

方框：数据通路

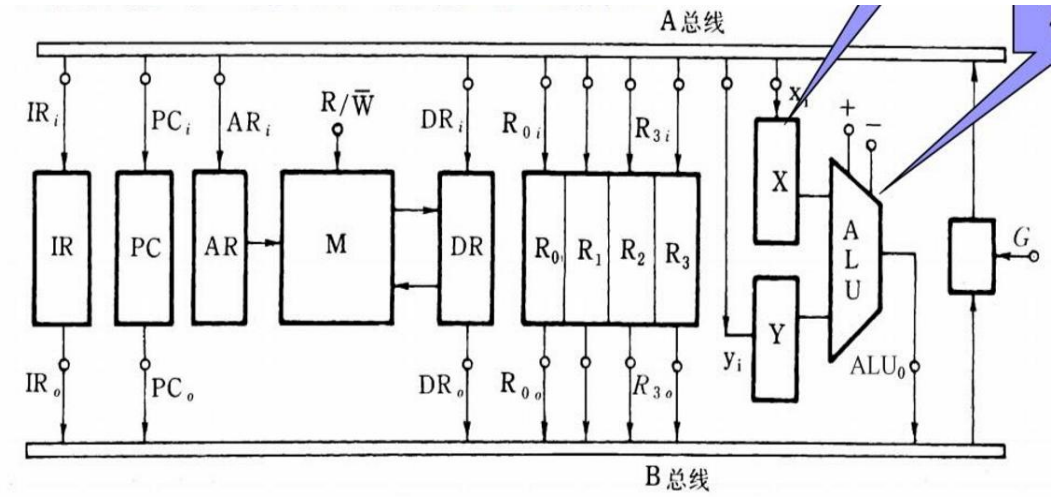
菱形：判别

~公操作：执行完毕

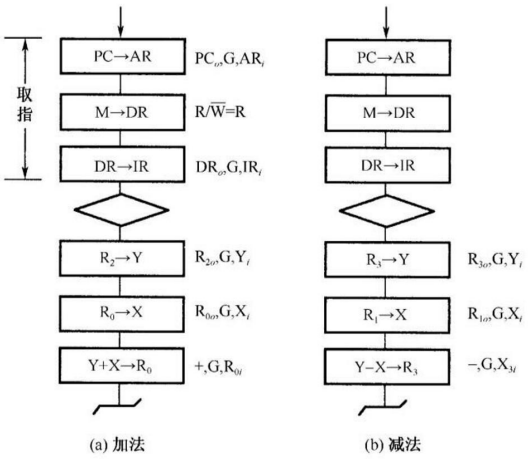


▼ 指令流程图

▼ 例1



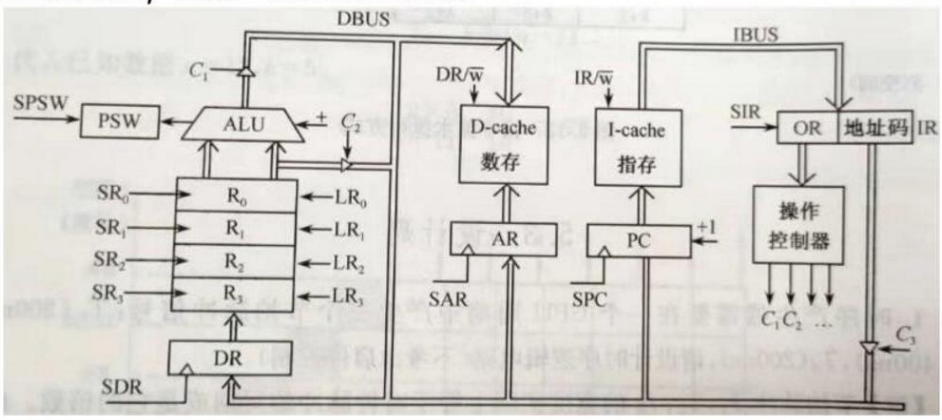
画出（1）ADD R0，R2 （2）SUB R1，R3的指令周期流程图，并列出相应的微操作信号。（小圈代表控制信号）

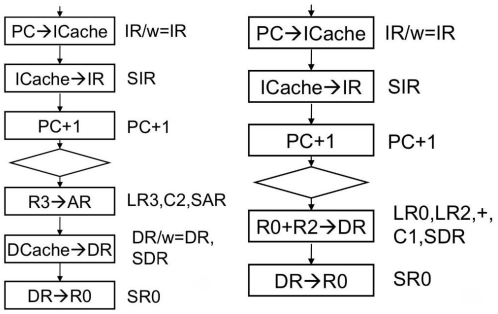


▼ 例2

CPU数据通路如图，单线箭头信号均为微操作控制信号，例如LR0表示读出R0寄存器，画出如下指令的指令周期流程图并标明控制信号

- LDA (R3), R0：读出以（R3）为地址的数存单元存入R0寄存器中
- ADD R2，R0：R0+R2→R0



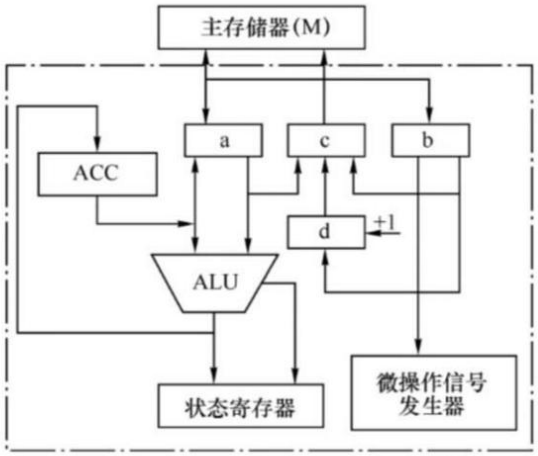


▼ 例3

图5-11是一个简化了的CPU与主存连接结构示意图。寄存器包括ACC、MAR、MDR、PC、IR。各部件间连线表示数据通路，PC具有自增功能

1）a、b、c、d名称

2）画出指令ADD Y的数据通路（Y为主存地址，功能为ACC+（Y）→ACC）



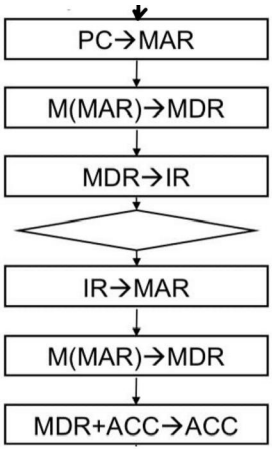
CPU与主存连接结构示意图

（1）a：MDR（主存的双向通路）

b：IR（连接微操作信号）

c：MAR（主存单向通路）

d：PC（+1）

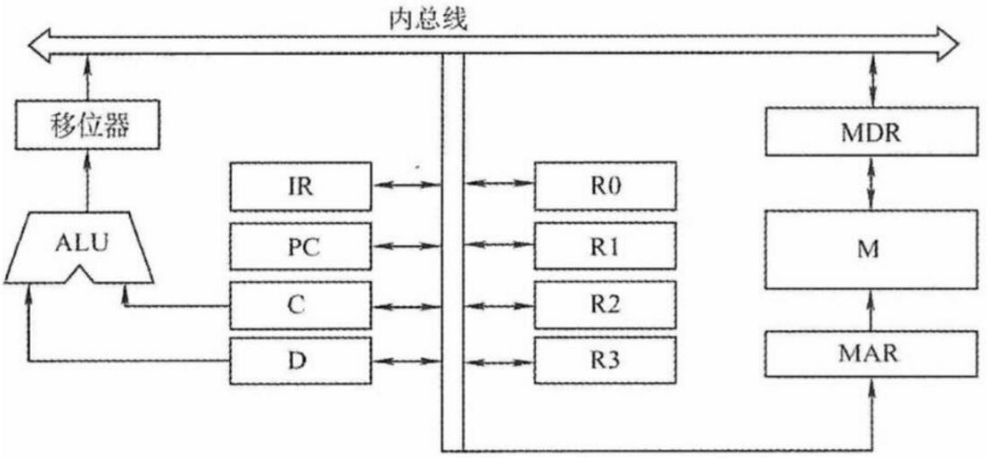


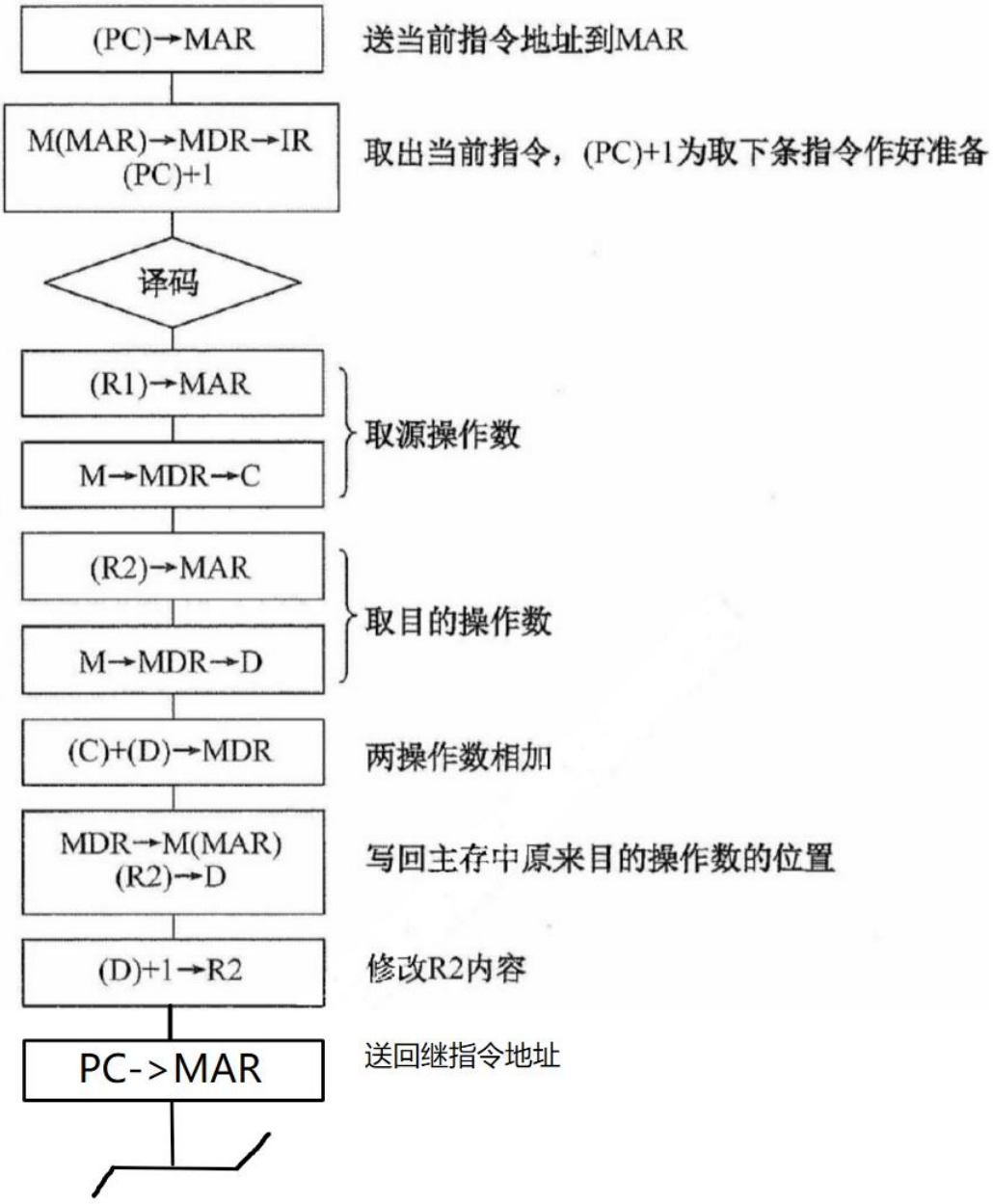
▼ 例4

某计算机主要功能部件如下图所示，C/D为暂存器

1）请补充各部件之间的主要连线，并注明连线数据方向

2）画出ADD [R1],[R2]+指令周期流程图，其中源操作数地址在R1，目标操作数地址为自增型寄存器间接寻址（先间址计算，地址再+1），结果写回R2寄存器



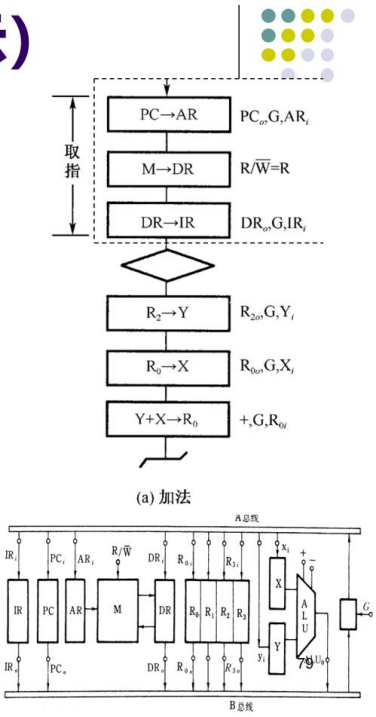

(PC)→MAR
送当前指令地址到MAR
M(MAR)→MDR→IR
(PC)+1
取出当前指令，(PC)+1为取下条指令作好准备
译码
(R1)→MAR
M→MDR→C
取源操作数
(R2)→MAR
M→MDR→D
取目的操作数
(C)+(D)→MDR
两操作数相加
MDR→M(MAR)
(R2)→D
写回主存中原来目的操作数的位置
(D)+1→R2
修改R2内容
PC->MAR
送回继指令地址

▼ 例5

微程序控制（程序化表示）

- 根据表格情况，调整微指令内容

微地址	微操作命令	功能
		取指阶段 uAR=OP;微地址由操作码决定
……	……	……
		ADD R2，R0： R2+R0→R0 注：ADD语句 OP码为0101



微地址	微操作命令	功能
0	PC_o,G,AR_i, ,R/W=R uAR=uAR+1	取指阶段 uAR=OP;微地址由操作码决定
1	DR_o,G,IR_i uAR=OP	
……	……	……
5	R_{2o},G,Y_i uAR=uAR+1	ADD R2，R0： R2+R0→R0 注：ADD语句 OP码为0101
6	R_{0o},G,X_i uAR=uAR+1	
7	+,G,R_{0i},uAR=0	

▼ 微程序控制器

把操作控制信号编制成微指令，存到控制存储器。运行时，从控存中取出微指令，产生指令运行所需的操作控制信号。

用软件方法来设计硬件

微命令：构成控制序列的最小单位

微操作：微命令的操作过程，执行部件中最基本的操作。

- 相容与互斥

微指令：微命令的组合

操作控制字段：微操作码 产生各个微操作控制信号

顺序控制字段：微地址码 产生下一条要执行的微指令地址

微程序：一系列微指令的有序集合

一条微程序对应一条机器指令

机器指令存在**主存储器**，**微指令**存储在**控制存储器**。

微程序控制器：

控制存储器（核心部件）

微指令寄存器

微地址形成部件

微地址寄存器

▼ 微指令格式

水平型微指令

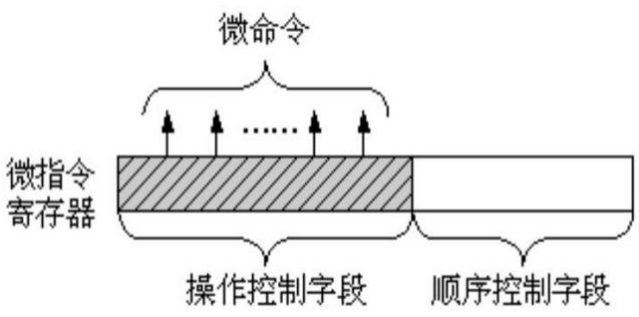
并行能力强、执行时间短、微指令字长微程序短

控制字段	判别测试字段	下地址字段

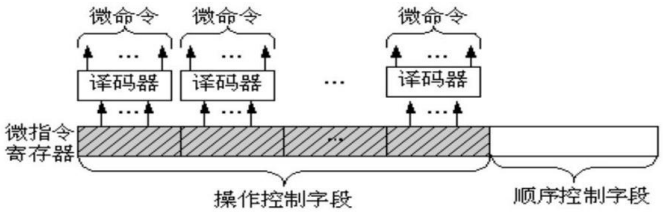
▼ 微指令结构

微命令编码：

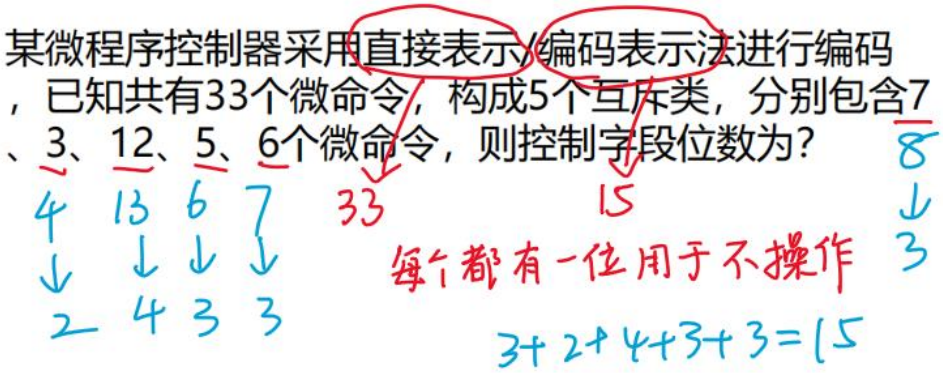
1. 直接表示法



2. 编码表示法（利用**互斥性**）



3. 混合表示法



垂直型微指令

并行差、执行时间长、微指令字短微程序长

000	源寄存器编址	目标寄存器编址	其他

▼ 微地址形成

计数器方式：

顺序执行：后继微地址=现行微地址+"1"

跳转：由微指令的转移地址段来形成转移微地址

不能实现两路以上的并行微程序转移，执行速度低

某微程序控制器采用计数器方式生成下地址，已知共有32条指令，公共取指微程序包含2条微指令，每条指令平均由4条微指令组成，微指令中下地址字段位数至少是多少位?

A 6

B 7

C 8

D 9

32x4+2=130

$2^7=128$

多路转移方式

▼ 流水线

时间并行：流水线技术

空间并行：超标量技术

时间+空间并行：超标量流水线

流水线技术

- 连续的任务
- 每个功能部件之后都要有缓冲寄存器（各级流水独立）
- 满载时效率最高

流水线周期=执行时间最长的指令（例，取指3ns，分析2ns，执行4ns，取4ns）

流水线执行时间 =（取指令t+分析t+执行t）+（总指令数-1）*流水线周期

流水线技术指标：吞吐率=总指令数/流水线执行时间

▼ 流水线指标

吞吐率：单位时间执行指令条数

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

18. 某 CPU 主频为 1.03GHz，采用 4 级指令流水线，每个流水段的执行需要 1 个时钟周期。假定 CPU 执行了 100 条指令，在其执行过程中，没有发生任何流水线阻塞，此时流水线的吞吐率为 C。

A. 0.25×10^9 条指令/秒　　B. 0.97×10^9 条指令/秒

C. 1.0×10^9 条指令/秒　　D. 1.03×10^9 条指令/秒

$k\Delta t+(n-1)\Delta t=4\times1+99\times1=103$ 个时钟周期

时钟周期 $=\frac{1}{f}=\frac{1}{1.03\times10^9}$ s　$Tp=\frac{100}{103\times\frac{1}{1.03\times10^9}}$

加速比：非流水执行时间/流水执行时间

理想：

$C_k = \frac{T_L}{T_K} = \frac{nk}{k+n-1}$

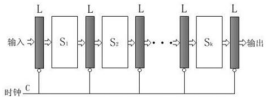
实际：

K级流水CPU：

- 每级流水所需时间t_i，
- 缓冲寄存器的延时 t_l，
- 每隔 $t = max\{t_i\} + t_l$完成一条指令
- 完成一条指令所需时延 $T = \Sigma t_i + Kt_l$

四级流水CPU，每级流水所需时间为200ps，
缓冲寄存器延迟50ps
流水线满载时，完成相邻两条指令时间间隔？
完成一条指令延时？ 200+50
200*4+50*4=1000

A 800ps, 1000ps
B 200ps, 800ps
C 200ps, 1000ps
D 250ps, 1000ps



20. 若某计算机最复杂指令的执行需要完成 5 个子功能，分别由功能部件 A～E 实现，各功能部件所需时间分别为 80ps、50ps、50ps、70ps 和 50ps，采用流水线方式执行指令，流水段寄存器延时为 20ps，则 CPU 时钟周期至少为_____。

A. 60ps B. 70ps C. 80ps D. 100ps

100 70 70 90 70 D

▼ 三种冲突/冒险

▼ 结构冒险：资源相关

多条指令同时争用同一部件

解决方法：空泡技术、硬件结构修改（独立的I-Cache和D-Cache）

▼ 数据冒险

RAW、WAW、WAR

解决方法：空泡技术、数据旁路（前递）

▼ 控制冒险

指令转移破坏流水线结构

解决方法：空泡技术、延迟转移、转移预测法

▼ 例题：

【例4】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

- 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。

（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

- 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。

（3）I5　MUL R3，R1，R2　；（R1）×（R2）->R3
　　I6　ADD R3，R4，R5　；（R4）+（R5）->R3

- 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。

36

▼ 综合例题

现有四级流水线，各部件分别完成取指、译码与取数、执行、回写操作。现假设各操作完成时间分别为100ns、100ns、80ns、50ns，忽略缓冲器时延。试回答如下问题：

（1）流水线操作周期如何设计？该流水线满载情况下的加速比为多少？

答：100ns、Ck=330ns/100ns=3.3

（2）试分析如下指令是否会产生冒险，若有，说明对应冒险种类

ADD R1，R2，R3　；（R2）+（R3）->R1

SUB R4，R1，R5　；（R1）-（R5）->R4

答：数据冒险（RAW）

（3）如果在硬件上加以改进（避免对应冒险），请说明改进方法，并说明流水线至少应推迟多少时间?

答：采用数据前递技术，不需要推迟时间

39

例题. 假设某指令流水线采用“按序发射，按序完成”方式，没有采用转发技术处理数据相关，并且同一寄存器的读和写操作不能在同一个时钟周期内进行。若高级语言程序中某赋值语句为x=a+b，x、a和b均为int型变量，它们的存储单元地址分别表示为[x]、[a]和[b]。该语句对应的指令序列及其在指令流中的执行过程如下图所示。

```
I1   LOAD   R1, [a]    M[a] —> R1
I2   LOAD   R2, [b]    M[b] —> R2
I3   ADD    R1, R2     ( R1 ) + ( R2 ) —> R2
I4   STORE  R2, [x]    ( R2 ) —> M[x]
```

IF ID EX MEM WB
取指 取数 写回

则这4条指令执行过程中I3的ID段和I4的IF段被阻塞的原因各是什么？

I3与I1和I2存在数据相关；

I4的IF段必须在I3进入ID段后才能开始，否则会覆盖IF段锁存器的内容

	时间单元													
指令	1	2	3	4	5	6	7	8	9	10	11	12	13	14
I_1	IF	ID	EX	M	WB									
I_2		IF	ID	EX	M	WB								
I_3			IF				ID	EX	M	WB				
I_4							IF				ID	EX	M	WB