开箱手册-计算机组成原理

ywy_c_asm

计算学部金牌讲师团

前言

- 本教程基于2022年秋季学期哈工大计算学部金牌讲师团为2020级的计算机组成原理课程所开设的期末复习讲座的课件，内容详细清楚，是对唐书所谓“知识”的浓缩提炼，适合作为复习/应付/预习期末考试的资料，主要作应试而用，**并不适合用于实用知识学习**。此外如果你是408考研的同学，尽管本教程很大程度上参考了考研真题(以及附带了很多考研真题解析)，但考虑到考研还会考一些期末考试不涉及的知识(此外难度也更大)，不建议完全拿本教程参考。

- 由于作者水平有限，难免会出一些差错，对此提前致以歉意。**此外本教程关于考试内容与重点的划分很大程度上来自个人观点，如果你因此而遭受损失，作者概不负责。**
- 作者在此提供了3次讲座的录屏，可以搭配学习：
  - 讲座1：https://pan.baidu.com/s/1fOFxWwhbjCmArpOGComvmg?pwd=1453 提取码：1453
  - 讲座2：https://pan.baidu.com/s/1JB7wPLLnagweQCtxZTS8_A?pwd=1453 提取码：1453
  - 讲座3：https://pan.baidu.com/s/1iH-PT6EGI3qaduCSA8xH7Q?pwd=1453 提取码：1453
- 作者还提供了配套模拟题：
  https://pan.baidu.com/s/1lY1QIeXhNL0wQ0WXRiGi5A?pwd=1453 提取码：1453
- 感谢你的使用，希望它对你有用。

前人经验-如何应付这门课

- 经过广大20级同学的评价，本课程的实质是**几乎毫无用处的八股文课程**，这一点需要同学们建立认识。作者不在此处作出进一步批判，只对如何应付本课程谈一些经验见解。
- 顾名思义，“八股文课程”的含义就是你需要对宝贵的唐书上的很多东西去死记硬背。例如数据的表示那一章，不管这章讲的是否符合实践需求，是否与计统等课程冲突，你必须无条件服从于唐书的奇怪的定点数表示格式(考试就需要这样写)以及计算流程(这个也是考试必考的)。再如唐书后面处理器部分书上看似是随便拿来举例的指令系统(ADD,STA,LDA)这些，这在现实中是不存在的，但你就是需要记住它们，考试会直接拿来用。总之，这本书上可能会有很多你可能自己也觉得没什么用的枯燥知识—但这门课程要求你记住它们，这就是现实。
- 更重要的是，注意唐书上的例题与作业题，考试会很大程度上以它们的形式(甚至直接搬原题)出题，相比于考研题，唐书上的题是更贴近期末考试的。考试大题的套路是，必定会有一道涉及中断/DMA的分析题、一道数据运算题(定点加减乘除/浮点加减)、一道指令格式设计题、一道给定数据通路结构的微操作节拍安排题、一道多存储芯片电路设计题(2023年居然没考，考的是跟计统一样的Cache分析)。这些题都能在唐书上的题里找到套路式模板。
- 一言以蔽之，你需要的是**无条件服从**，但是，你需要知道自己**并不是真的**在无条件服从，不能将本课程所学知识当作真理。当考试结束后，建议尽快抛弃遗忘它们，那是早该湮没于时代尘埃中的糟粕。

目录

(星号个数表示作者个人亲历期末考试后所认为的重要程度，仅供参考)

注意：2023年期末考试还考查了虚拟内存的内容，这部分和计统是一致的，故不在本教程中阐述。

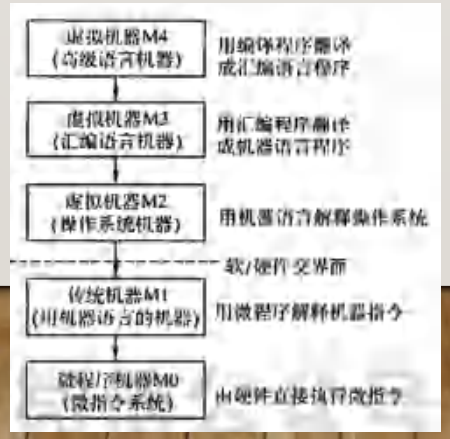
1.计算机系统概论

01. 完整的计算机系统应包括（ ）。
A. 运算器、存储器、控制器　　B. 外部设备和主机
C. 主机和应用程序　　D. 配套的硬件设备和软件系统

- 计算机系统=硬件+软件，软件=**系统软件**+**应用软件**
- 典型的系统软件：操作系统、标准程序库、语言处理（编译链接等）、数据库管理软件、网络管理软件等。

13. 下列（ ）属于应用软件。
A. 操作系统　　B. 编译程序　　C. 连接程序　　D. 文本处理

- 计算机**体系结构**问题：机器是否具备某指令？（指令系统结构，程序员可见）
- 计算机**组成**问题：某指令的器件如何实现？（相同结构，不同组成）
- **编译程序**与**解释程序**的区别：解释程序逐条翻译高级语言并立即执行。
- 五级层次结构的计算机系统：



16. 只有当程序执行时才将源程序翻译成机器语言，并且一次只能翻译一行语句，边翻译边执行的是（ ）程序，把汇编语言源程序转变为机器语言程序的过程是（ ）。
I. 编译　II. 目标　III. 汇编　IV. 解释
A. I、II　　B. IV、II　　C. IV、I　　D. IV、III

2.冯诺伊曼机

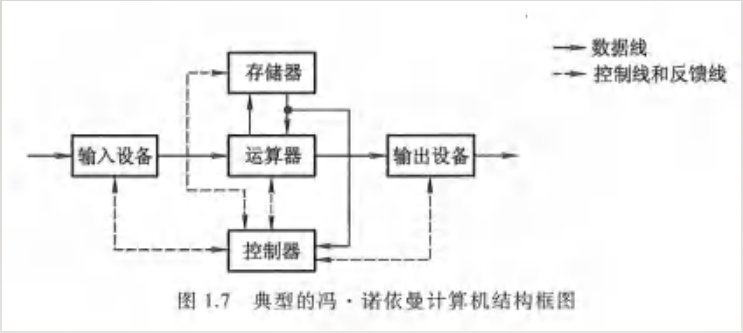


图 1.7　典型的冯·诺依曼计算机结构框图

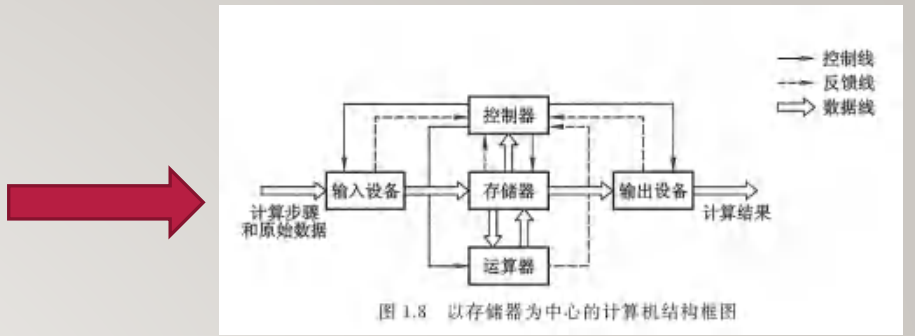


图 1.8　以存储器为中心的计算机结构框图

- 冯诺伊曼机主要特点：
- ①五大部件：**控制器**、**运算器**、**存储器**、**输入设备**、**输出设备**。
- ②**存储程序**：指令与数据以同等地位存储并按地址访问。（最根本特征）
- ③指令与数据以二进制表示。
- ④指令由操作码和地址码组成，操作码表示操作性质，地址码表示操作数位置。
- ⑤指令在存储器中顺序存放，通常顺序执行，可根据条件改变执行顺序。
- ⑥以**运算器为中心**，输入输出数据和存储器之间需要经过运算器。（早期模型）

现代计算机运算器速度极快，故改变为**以存储器为中心**，输入输出设备可直接与存储器交互

25.【2019 统考真题】下列关于冯·诺依曼计算机基本思想的叙述中，错误的是（　）。
A. 程序的功能都通过中央处理器执行指令实现
B. 指令和数据都用二进制数表示，形式上无差别
C. 指令按地址访问，数据都在指令中直接给出
D. 程序执行前，指令和数据需预先存放在存储器中

03. 下列（　）是冯·诺依曼机工作方式的基本特点。
A. 多指令流单数据流
B. 按地址访问并顺序执行指令
C. 堆栈操作
D. 存储器按内容选择地址

08. 在CPU的寄存器中，（ ）对用户是完全透明的。
A. 程序计数器　B. 指令寄存器　C. 状态寄存器　D. 通用寄存器

3.计算机组成框图与工作流程

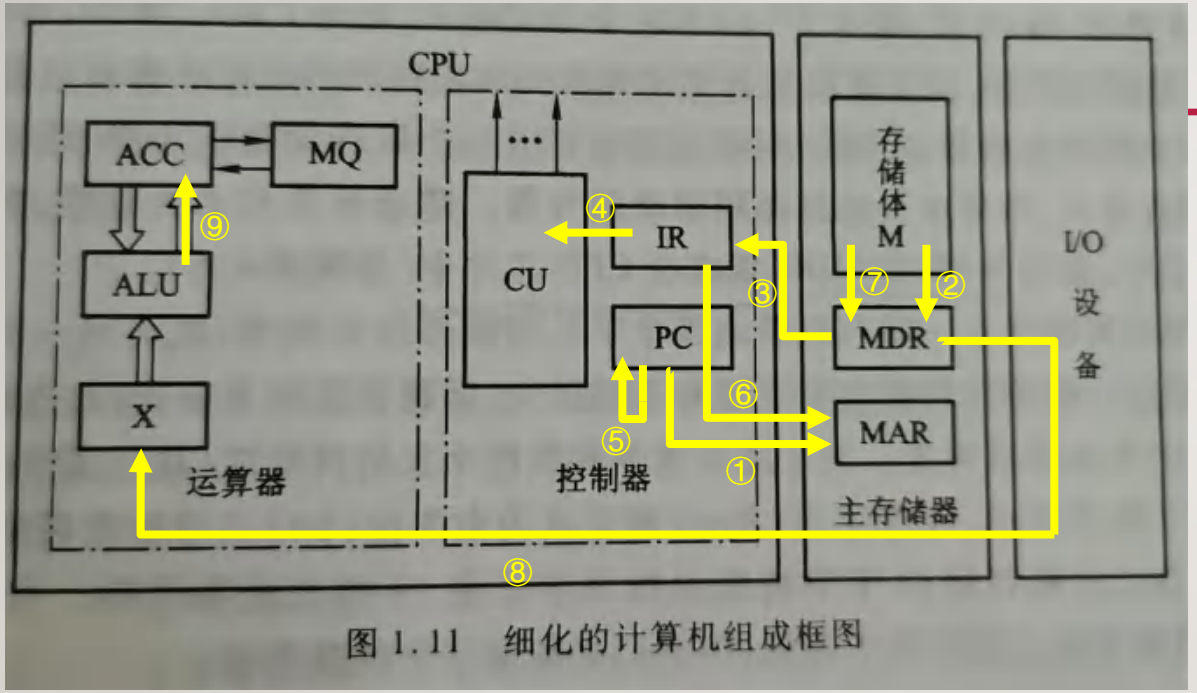


图 1.11　细化的计算机组成框图

加法指令"ADD addr"的执行过程：
（将ACC的值加上地址addr的存储单元的数）
①PC->MAR
②M(MAR)->MDR
③MDR->IR
④OP(IR)->CU
⑤(PC)+1->PC
⑥Ad(IR)->MAR
⑦M(MAR)->MDR
⑧MDR->X
⑨(ACC)+(X)->ACC

取指并译码

当寄存器参与某种运算时需要加括号（书写标准）

CU：控制单元，解释指令，发出控制信号来执行指令
PC：程序计数器，存放当前欲执行指令的地址
IR：指令寄存器，存放当前的指令（操作码OP(IR)，地址码Ad(IR)）
ALU：算术逻辑单元，用来完成算术逻辑运算
ACC：累加寄存器，存放被加数、被减数、被除数、和、差、乘积高位、余数
X：操作数寄存器，存放加数、减数、乘数、除数
MQ：乘商寄存器，存放乘数、乘积低位、商（**w位×w位=2w位，2w位÷w位=w位商w位余数**）
MAR：存储器地址寄存器，存放访问存储单元地址，位数反映存储单元个数
MDR：存储器数据寄存器，存放读写存储器数据，位数与存储字长相等

可在CPU中

05. 组成一个运算器需要多个部件，但下面的（ ）不是组成运算器的部件。
A. 状态寄存器　B. 数据总线　C. ALU　D. 地址寄存器

06. 算术逻辑单元（ALU）的功能一般包括（ ）。
A. 算术运算　B. 逻辑运算
C. 算术运算和逻辑运算　D. 加法运算

05. 存放欲执行指令的寄存器是（ ）。
A. MAR　B. PC　C. MDR　D. IR

06. 在CPU中，跟踪下一条要执行的指令的地址的寄存器是（ ）。
A. PC　B. MAR　C. MDR　D. IR

18. 在CPU的组成中，不包括（ ）。
A. 运算器　B. 存储器　C. 控制器　D. 寄存器

07. CPU不包括（ ）。
A. 地址寄存器　B. 指令寄存器（IR）
C. 地址译码器　D. 通用寄存器

4.计算机性能指标

18.【2013 统考真题】某计算机的主频为 1.2GHz，其指令分为 4 类，它们在基准程序中所占比例及 CPI 如下表所示。

指令类型	所占比例	CPI	指令类型	所占比例	CPI
A	50%	2	C	10%	4
B	20%	3	D	20%	5

该机的 MIPS 数是（ ）。

A. 100　　B. 200　　C. 400　　D. 600

总CPI$= 0.5*2+0.2*3+0.1*4+0.2*5=3$

指令执行速度$=\frac{1.2*10^9 clk/s}{3clk/i}=4*10^8 \text{ips}=400\text{MIPS}$

04. 下列关于机器字长，指令字长和存储字长的说法中，正确的是（ ）。

I. 三者在数值上总是相等的　　II. 三者在数值上可能不等

III. 存储字长是存放在一个存储单元中的二进制代码位数

05. 32 位微机是指该计算机所用 CPU（ ）。

A. 具有 32 位寄存器　　B. 能同时处理 32 位的二进制数

C. 具有 32 个寄存器　　D. 能处理 32 个字符

一般称的“字”/“字长”

- 区分三种字长：（无论如何它们都一定是字节(8位)的整数倍）

长度可以相同，也可以不相同

- **机器字长**：机器一次处理数据的位数，通常与(通用)寄存器位数有关。
- **存储字长**：一个存储单元存储的二进制数据的位数，等于MDR的位数。
- **指令字长**：一条指令的位数，长度一般可变。

21.【2020 统考真题】下列输出的部件中，其位数（宽度）一定与机器字长相同的是（ ）。

I. ALU　　II. 指令寄存器　　III. 通用寄存器　　IV. 浮点寄存器

A. 仅 I、II　　B. 仅 I、III　　C. 仅 II、III　　D. 仅 II、III、IV

03. 以下说法中，错误的是（ ）。

A. 计算机的机器字长是指数据运算的基本单位长度

B. 寄存器由触发器构成

C. 计算机中一个字的长度都是 32 位

D. 磁盘可以永久性存放数据和程序

- 存储容量=存储单元个数×存储字长（单位：b，**一定要区分b(位)与B(字节)**！），现代常用B作为单位。注意：一个地址**唯一**确定一个存储单元，地址就是存储单元编号。
- 运算速度：百万次指令每秒MIPS，浮点运算次数每秒FLOPS，平均指令周期数CPI

注意：在速度/频率/带宽等单位中，K/M/G/T表示10的幂，但在存储容量中表示2的幂

10. 计算机中，CPU 的 CPI 与下列（ ）因素无关。

A. 时钟频率　　B. 系统结构　　C. 指令集　　D. 计算机组织

06. 用于科学计算的计算机中，标志系统性能的最有用的参数是（ ）。

A. 主时钟频率　　B. 主存容量　　C. MFLOPS　　D. MIPS

CPU执行时间$=\frac{\text{指令条数}\times CPI}{\text{时钟频率}}$

指令执行速度$=\frac{\text{指令条数}}{\text{CPU执行时间}}=\frac{\text{时钟频率}}{CPI}$

20.【2017 统考真题】假定计算机 M1 和 M2 具有相同的指令集体系结构（ISA），主频分别为 1.5GHz 和 1.2GHz。在 M1 和 M2 上运行某基准程序 P，平均 CPI 分别为 2 和 1，则程序 P 在 M1 和 M2 上运行时间的比值是（ ）。

A. 0.4　　B. 0.625　　C. 1.6

$\frac{2\times P}{1.5\text{GHz}}:\frac{1\times P}{1.2\text{GHz}}=1.6$

5.定点数的表示方式

对于整数，用逗号将符号位隔开，对于小数，用小数点将符号位隔开（书写标准）

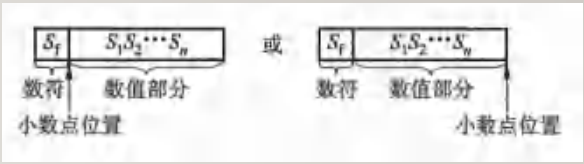
Examples: $(n=4)$

$[+0]_{原}=[+0]_{反}=0,0000\quad [-0]_{原}=1,0000\quad [-0]_{反}=1,1111$

$[+0.75]_{原}=[+0.1100]_{原}=[+0.75]_{反}=0.1100\quad [-0.75]_{原}=1.1100\quad [-0.75]_{反}=1.0011$

$[+12]_{原}=[+1100]_{原}=[+12]_{反}=0,1100\quad [-12]_{原}=1,1100\quad [-12]_{反}=1,0011$

- 首要问题：一个抽象的数字，如何用w个二进制位表示？w位二进制代码以什么方式被**解读**为它所表示的数字？
  - 所谓的“机器数”
  - 所谓的“真值”
- 定点数：在机器级表示中，二进制代码位数固定，并且小数点位置也固定。在本书中一般被划分为纯整数和纯小数（二者其实某种程度上是等价的）。



- 对于有符号数，表示为“符号位+n位二进制数”的形式：实际上表示为$n+1$位！

符号+绝对值

  - **原码**：负数符号位为1，否则为0，剩下的n位表示真值x的**绝对值**，因此有+0和-0两种不同的机器数。对于整数，$x\in[-(2^n-1),2^n-1]$，对于小数，$x\in[-(1-2^{-n}),1-2^{-n}]$

原码不适合加法运算，需要同时设置加法器和减法器

  - **反码**：正数以及+0的反码**跟原码一样**，负数以及-0的反码为将其原码中绝对值按位取反，但**符号位不变**。表示范围同原码，属于原码到补码的中间过渡，现实中没什么用。

10. 一个 $n+1$ 位整数 x 原码的数值范围是（ ）。
A. $-2^n+1<x<2^n-1$　B. $-2^n+1\le x<2^n-1$
C. $-2^n+1<x\le 2^n-1$　D. $-2^n+1\le x\le 2^n-1$

09. 8位原码能表示的不同数据有（ ）个。
A. 15　B. 16　C. 255　D. 256

11. n 位定点整数（有符号）表示的最大值是（ ）。
A. 2^n　B. 2^n-1　C. 2^{n-1}　D. $2^{n-1}-1$

20. 若寄存器内容为 10000000，若它等于-0，则为（ ）。
A. 原码　B. 补码　C. 反码　D. 移码

24. 若二进制定点小数真值是-0.1101，机器表示为 1.0010，则为（ ）。
A. 原码　B. 补码　C. 反码　D. 移码

5.定点数的表示方式-如何理解补码?

$n=2$ 的例子，补码相当于把数分布在模2^{n+1}的循环上使其易于加减

- **补码**：对于整数$[x]_{补}=(2^{n+1}+x)\ mod\ 2^{n+1}$，对$2^{n+1}$取模相当于**仅取结果低$n+1$位**，真值$x\in[-2^n,2^n-1]$，同理对于小数$[x]_{补}=(2+x)\ mod\ 2$，对2取模相当于**仅取小数点前1位以后的部分**。真值$x\in[-1,1-2^{-n}]$。可以发现"符号位"被赋予了更多的数学含义。

由于补码只有1种0，所以可以比原码多表示一个-2^n

$$y_n,y_{n-1}y_{n-2}\cdots y_0=-2^n y_n+\sum_{i=0}^{n-1}2^i y_i$$

符号位位权为负

- $[x]_{补}+[-x]_{补}=(2^{n+1}+x)+(2^{n+1}-x)\bmod 2^{n+1}=0=[0]_{补}$

这里都是纯粹的$n+1$位二进制加减法

- $[a\pm b]_{补}=(2^{n+1}+a\pm b)\bmod 2^{n+1}=(2^{n+1}+a)\pm(2^{n+1}+b)\bmod 2^{n+1}=[a]_{补}\pm[b]_{补}$
- $[-x]_{补}=[0-x]_{补}=[0]_{补}-[x]_{补}=0-[x]_{补}$
  - 补码取负相当于将这串二进制代码**按位取反再+1**。

我们希望对抽象的数字进行的数学运算，可以在补码表示下对二进制代码直接使用加法器进行$n+1$位二进制加减法！

（这种优美性质是原码不具有的，显然没有$[a+b]_{原}=[a]_{原}+[b]_{原}$）

Examples: $(n=4)$

$$[0]_{补}=100000+00000=100000\xrightarrow{取低n+1位}0,0000$$

$$[-1]_{补}=100000-00001=011111\xrightarrow{取低n+1位}1,1111=1,1110+1=\sim(0,0001)+1=\sim[1]_{补}+1$$

$$[-16]_{补}=100000-10000=010000\xrightarrow{取低n+1位}1,0000$$

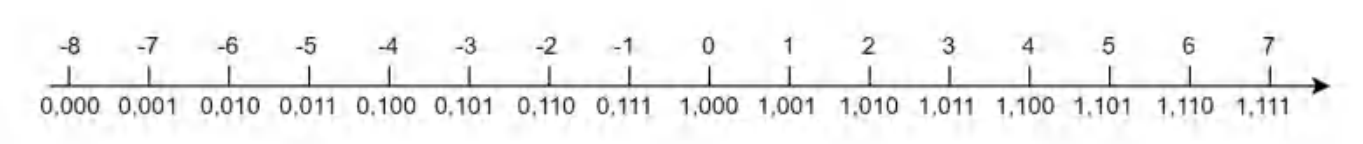
26. 计算机内部的定点数大多用补码表示，以下是一些关于补码特点的叙述：
I. 零的表示是唯一的　II. 符号位可以和数值部分一起参加运算
III. 和真值的对应关系简单、直观　IV. 减法可用加法来实现
在以上叙述中，（　）是补码表示的特点。

08. 若X为负数，则由$[X]_{补}$求$[-X]_{补}$是将（　）。
A. $[X]_{补}$各值保持不变
B. $[X]_{补}$符号位变反，其他各位不变
C. $[X]_{补}$除符号位外，各位变反，末位加1
D. $[X]_{补}$连同符号位一起变反，末位加1

5.定点数的表示方式

注意！在一些场合下移码的偏移量可以是其它值！例如IEEE754的阶码偏移量为2^n-1

- 移码：对于整数$[x]_{移}=2^n+x$，真值$x\in[-2^n,2^n-1]$，对于小数$[x]_{移}=1+x$，真值$x\in[-1,1-2^{-n}]$，对补码的一种修正，使得数字**易于比较大小**（可以直接对$n+1$位的二进制代码做从高位到低位的比较），但还是不方便做加减法……



将所有真值按照大小顺序分布在数轴上（而不是环上）
恰好与补码符号位相反

（题外话：通过以上几例可以发现，整数的2^n等同于小数的1，整数的2^{n+1}等同于小数的2，因此整数与纯小数在很多情况下几乎是等价的……）

14. 下列关于补码和移码关系的叙述中，（ ）是不正确的。
A. 相同位数的补码和移码表示具有相同的数据表示范围
B. 零的补码和移码表示相同
C. 同一个数的补码和移码表示，其数值部分相同，而符号相反
D. 一般用移码表示浮点数的阶，而补码表示定点整数

06. 对真值0表示形式唯一的机器数是（ ）。
A. 原码　B. 补码和移码　C. 反码　D. 以上都不对

25. 下列为8位移码机器数$[x]_{移}$，求$[-x]_{移}$时，（ ）将会发生溢出。
A. 11111111　B. 00000000　C. 10000000　D. 01111111

$x=-128$

5.定点数的表示方式

07. 若$[X]_{补}=1.1101010$，则$[X]_{原}=$（ ）。
A. 1.0010101　B. 1.0010110　C. 0.0010110　D. 0.1101010

符号位=1，负数，其**绝对值** $[-X]_{补}=1.1101010 \xrightarrow{按位取反} 0.0010101 \xrightarrow{低位+1} 0.0010110$，
因此$X=-0.0010110$，$[X]_{原}=1.0010110$

15. 若$[x]_{补}=1,x_1x_2x_3x_4x_5x_6$，其中$x_i$取0或1，若要$x>-32$，应当满足（ ）。
A. x_1为0，其他各位任意　B. x_1为1，其他各位任意
C. x_1为1，$x_2\cdots x_6$中至少有一位为1　D. x_1为0，$x_2\cdots x_6$中至少有一位为1

用4种不同的解读方式解读二进制代码

机器数	原码	反码	补码	移码
0,000	+0	+0	0	-8
0,001	1	1	1	-7
0,010	2	2	2	-6
0,011	3	3	3	-5
0,100	4	4	4	-4
0,101	5	5	5	-3
0,110	6	6	6	-2
0,111	7	7	7	-1
1,000	-0	-7	-8	0
1,001	-1	-6	-7	1
1,010	-2	-5	-6	2
1,011	-3	-4	-5	3
1,100	-4	-3	-4	4
1,101	-5	-2	-3	5
1,110	-6	-1	-2	6
1,111	-7	-0	-1	7

同理

$[-32]_{补}=10000000-0100000=1,100000$，$x$已经是负数，因此剩下的6位应该比二进制数100000大（因为补码**除符号位以外的位权都是正的**，除符号位以外越是1越大）

28. 【2015统考真题】由3个“1”和5个“0”组成的8位二进制补码，能表示的最小整数是（ ）。
$1,0000011=[-128+2+1]_{补}=[-125]_{补}$
A. -126　B. -125　C. -32　D. -3

18. 假定一个十进制数为-66，按补码形式存放在一个8位寄存器中，该寄存器的内容用十六进制表示为（ ）。
A. C2H　B. BEH　C. BDH

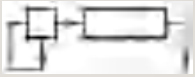
$[-66]_{补}=100000000-01000010=10111110=BEH$

大家仍然需要熟练掌握进制转换！

20. 若寄存器内容为10000000，若它等于-0，则为（ ）。
A. 原码　B. 补码　C. 反码　D. 移码
21. 若寄存器内容为11111111，若它等于+127，则为（ ）。
A. 反码　B. 补码　C. 原码　D. 移码
22. 若寄存器内容为11111111，若它等于-1，则为（ ）。
A. 原码　B. 补码　C. 反码　D. 移码
23. 若寄存器内容为00000000，若它等于-128，则为（ ）。
A. 原码　B. 补码　C. 反码　D. 移码
24. 若二进制定点小数真值是-0.1101，机器表示为1.0010，则为（ ）。
A. 原码　B. 补码　C. 反码　D. 移码

在什么都不知道的情况下，
符号位真不一定能反映正负

6.定点数移位运算

- 逻辑移位（无符号数）：无论左右移都用0补充，用于位操作。
- 算术移位（有符号数）：左移简单处理，右移要考虑符号位（保证右移**在算术意义上除2**），无论如何**左右移都不会改变符号位**。算术左移也不会改变符号位！这个定义和C语言等不同
  - 原码：左移低位补0（丢掉次高位），右移符号位不变，次高位补0。
  - 反码：正数同原码，**负数左移低位补1**，右移符号位不变，次高位补1。再次强调反码只有在负数时才是反的！
  - 补码：左移低位补0，右移符号位不变，移入次高位。 

08. 补码定点整数 0101 0101 左移两位后的值为（ ）。
A. 0100 0111　B. 0101 0100　C. 0100 0110　D. 0101 0101

09. 补码定点整数 1001 0101 右移一位后的值为（ ）。
A. 0100 1010　B. 01001010 1　C. 1000 1010　D. 1100 1010

11. 设机器数字长 8 位（含 1 位符号位），若机器数 BAH 为原码，算术左移 1 位和算术右移 1 位分别得（ ）。
A. F4H, EDH　B. B4H, 6DH　C. F4H, 9DH　D. B5H, EDH

12. 若机器数 BAH 为补码，其余条件同上题，则有（ ）
A. F4H, DDH　B. B4H, 6DH　C. F4H, 9DH　D. B5H, EDH

BAH = 1,0111010 ，无论原码还是补码，左移都在低位补0，符号位不变，为1,1110100 = F4H，原码右移在次高位补0，为1,0011101 = 9DH，补码右移把符号位移入次高位，为1,1011101 = DDH

7.定点数加减运算

一些无溢出双符号位例子：

$a=-0.75,\ b=0.125$

$[a]_{补}=1.0100,\ [b]_{补}=0.0010,\ [-b]_{补}=1.1110$

$a+b$:	$a-b$:	$b+b$:	$-b-b$:
11.0100	11.0100	00.0010	11.1110
+00.0010	+11.1110	+00.0010	+11.1110
11.0110	11.0010	00.0100	11.1100

- 本书(以及现代计算机)只讨论**补码加减法**，因此对二进制代码做简单加减法即可！实际上只需要做加法，减法使用对相反数做加法实现（补码的取负更容易）。
- $n=4$的例子：

$a=-0.75,\ b=0.125$

$[a]_{补}=1.0100,\ [b]_{补}=0.0010,\ [-b]_{补}=1.1110$

1.0100	1.0100
+0.0010	+1.1110
1.0110	1.0010

$[a+b]_{补}=[a]_{补}+[b]_{补}=1.0100+0.0010=1.0110=[-0.625]_{补}$，故$a+b=-0.625$

$[a-b]_{补}=[a]_{补}+[-b]_{补}=1.0100+1.1110=1.0010=[-0.875]_{补}$，故$a-b=-0.875$

$a=-0.75,\ [a]_{补}=1.0100$，计算$a+a$

注意这里是带符号扩展！

1.0100	11.0100
+1.0100	+11.1110
0.1000	10.1000
负+负=非负，溢出！	真正的符号位与本来的符号位不同，溢出！

- 加法溢出判断

进位并不代表溢出！溢出也不一定有进位！

  - 单符号位：两数符号位不同则不溢出，**符号位相同且结果符号位改变**则溢出。
  - “双”符号位：溢出**至多只溢出一位**，因此符号扩展一位做加法，结果现在的最高位表示结果**真正的符号**，不会被影响。因此看“本来的符号位”是否跟真正的符号位不一样即可！相当于对整数补码在模2^{n+2}（对小数在模4）意义下做二进制加法，多保留一位。

14. 在定点运算器中，无论是采用双符号位还是采用单符号位，必须有（ ）。
A. 译码电路，它一般用“与非”门来实现
B. 编码电路，它一般用“或非”门来实现
C. 溢出判断电路，它一般用“异或”门来实现
D. 移位电路，它一般用“与或非”门来实现

位是否相同/不同都可用异或实现

16. 若采用双符号位，则两个正数相加产生溢出的特征时，双符号位为（ ）。
A. 00　B. 01　C. 10　D. 11

18. 在补码的加减法中，用两位符号位判断溢出，两位符号位$S_{s1}S_{s2}=10$时，表示（ ）。
A. 结果为正数，无溢出　B. 结果正溢出
C. 结果负溢出　D. 结果为负数，无溢出

7.定点数加减运算-*深入理解加法溢出

- 以n位+1位符号位的整数补码为例，真值$x \in [-2^n, 2^n - 1]$，那么$a + b$的结果应该在$[-2^{n+1}, 2^{n+1} - 2]$内（这个范围仍然可以用$n + 1$位+1位符号位补码正常表示！因此**双符号位是安全的**！），如果发生正溢出，结果在$[2^n, 2^{n+1} - 2]$内，恰好需要占据原本的符号位（但不会占据更高的位），即$1, xxxxxxxx……$，因此两个正数相加溢出结果为负，同理两个负数相加溢出结果为正。
- 两正数相加溢出，一定是**次高位向符号位发生进位**，两负数相加溢出，一定是**次高位没有向符号位发生进位**，在双符号位下就是这样的情况：

（注：在双符号位下，最高位是真正的符号位，位权为-2^{n+1}，原符号位的位权改为2^n，其实本质上本来就是扩展一位后的正常的$n+1$位+1位符号位的补码！）

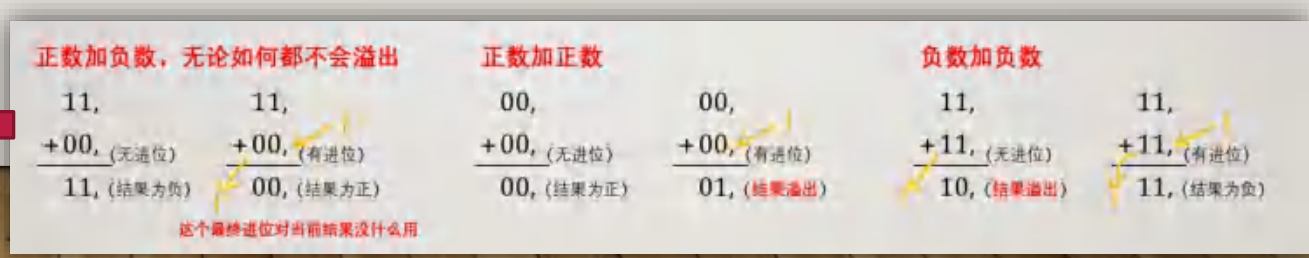
17. 判断加减法溢出时，可采用判断进位的方式，若符号位的进位为 C_0，最高位的进位为 C_1，则产生溢出的条件是（ ）。

I. C_0 产生进位　　II. C_1 产生进位

III. C_0、C_1 都产生进位　　IV. C_0、C_1 都不产生进位

V. C_0 产生进位，C_1 不产生进位　　VI. C_0 不产生进位，C_1 产生进位

A. I 和 II　　B. III　　C. IV　　D. V 和 VI

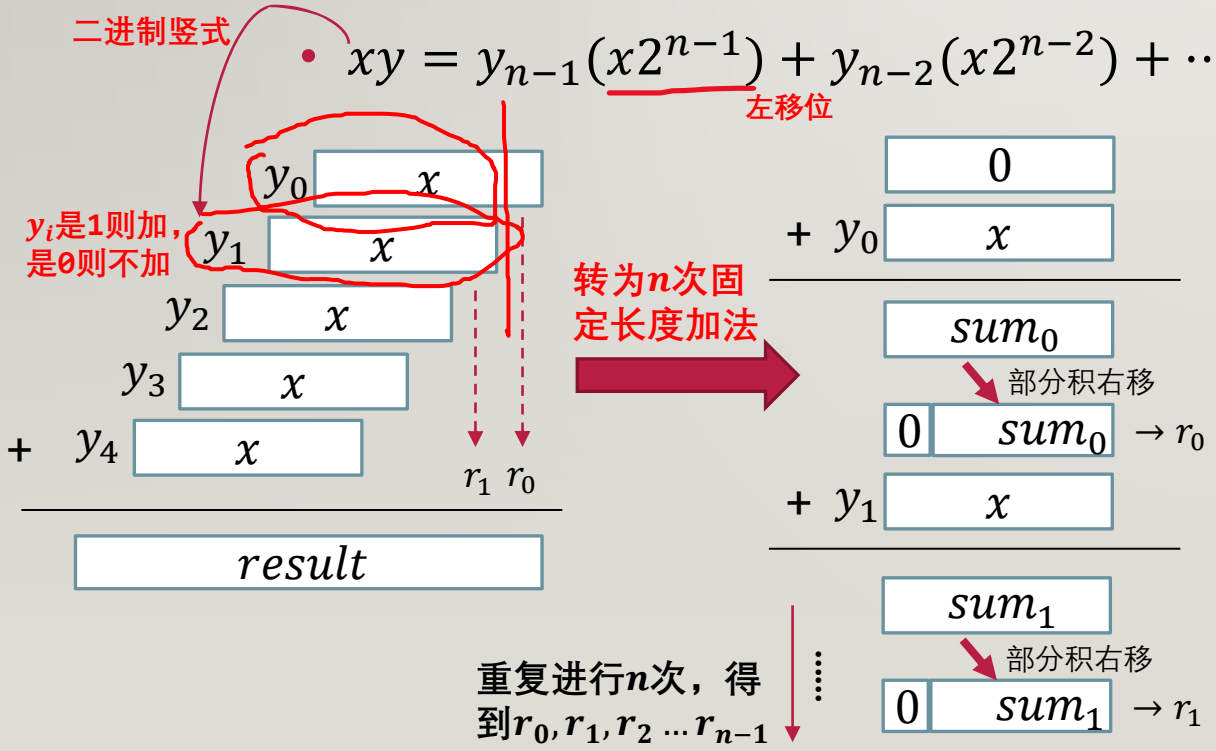


22. 在原码一位乘法中，（ ）。
A. 符号位参加运算
B. 符号位不参加运算
C. 符号位参加运算，并根据运算结果改变结果中的符号位
D. 符号位不参加运算，并根据运算结果确定结果中的符号

20. 原码乘法是（ ）。
A. 先取操作数绝对值相乘，符号位单独处理
B. 用原码表示操作数，然后直接相乘
C. 被乘数用原码表示，乘数去绝对值，然后相乘
D. 乘数用原码表示，被乘数去绝对值，然后相乘

8.定点数乘法运算（乘数$n+1$位，结果$2n+1$位）

- **原码1位乘**：将乘数x和乘数y都取其绝对值，转化为两个n位非负二进制数的乘法，得到一个$2n$位二进制数结果，最终根据乘数的符号决定结果的符号位。
- $y=0,y_{n-1}y_{n-2}y_{n-3}\cdots y_0=y_{n-1}2^{n-1}+y_{n-2}2^{n-2}+\cdots+y_02^0\ \ (y_i\in\{0,1\})$
- $xy=y_{n-1}(x2^{n-1})+y_{n-2}(x2^{n-2})+\cdots+y_0(x2^0)$



乘法过程中涉及到所有数皆非负，但为防止加法溢出，部分积要额外开一位，反正由于右移操作在最终只会得到n位部分积作为结果高位

例： 在4+1位原码下计算$x=15$与$y=-10$的乘积。

$[x]_{原}=0,1111$，$[y]_{原}=1,1010$，绝对值$[x^*]_{原}=0,1111$，$[y^*]_{原}=0,1010$

部分积	乘数	结果低位
0,0000 +0,0000	0,1010　$y_0=0$，不加	空
0,0000 右移一位 0,0000 +0,1111	0,1010　$y_1=1$，加x^*	→0
0,1111 右移一位 0,0111 +0,0000	0,1010　$y_2=0$，不加	→10
0,0111 右移一位 0,0011 +0,1111	0,1010　$y_3=1$，加x^*	→110
1,0010 右移一位 0,1001	不操作，不加	→0110

这里其实是因为符号位恒为0

因此$[x^*y^*]_{原}=0,10010110$，由于是正数乘负数，故结果为负

$[xy]_{原}=1,10010110=[-150]_{原}$

8.定点数乘法运算
(乘数$n+1$位，结果$2n+1$位)

24. 实现N位（不包括符号位）补码一位乘时，乘积为（ ）位。
A. N　　B. $N+1$　　C. $2N$　　D. $2N+1$

- **补码1位乘**（Booth算法）：所有运算全部补码实现，每次扫描乘数的**连续两位**作控制。
- $y = y_n, y_{n-1}y_{n-2}y_{n-3}\cdots y_0 = -y_n 2^n + y_{n-1}2^{n-1} + y_{n-2}2^{n-2} + \cdots + y_0 2^0 \quad (y_i \in \{0,1\})$
- $= -y_n 2^n + y_{n-1}(2^n - 2^{n-1}) + y_{n-2}(2^{n-1} - 2^{n-2}) + \cdots + y_0(2^1 - 2^0)$
- $= 2^n(y_{n-1} - y_n) + 2^{n-1}(y_{n-2} - y_{n-1}) + \cdots + 2^1(y_0 - y_1) + 2^0(y_{-1} - y_0) \quad (y_{-1} = 0)$
- 则$xy = (y_{n-1} - y_n)(x2^n) + (y_{n-2} - y_{n-1})(x2^{n-1}) + \cdots + (y_{-1} - y_0)(x2^0)$

乘数连续两位作控制

y_{i-1}	y_i	$y_{i-1} - y_i$	操作
0	0	0	不加
1	1		
1	0	1	加$[x]_{补}$
0	1	-1	加$[-x]_{补}$

例：在4+1位补码下计算$x = 15$与$y = -10$的乘积。

$[x]_{补} = 0,1111$，$[y]_{补} = 1,0110$，$[-x]_{补} = 1,0001$

一开始要把$[-x]_{补}$预先算出

这里同样也要为避免溢出扩展一位，补码的符号位要参与运算，因此双符号位

这里的右移是算术右移

部分积	乘数	结果低位
00,0000 +00,0000	1,01100　$y_{-1} - y_0 = 0$，不加	空
00,0000 右移一位 00,0000 +11,0001	1,01100　$y_0 - y_1 = -1$，加$[-x]_{补}$	→0
11,0001 右移一位 11,1000 +00,0000	1,01100　$y_1 - y_2 = 0$，不加	→10
11,1000 右移一位 11,1100 +00,1111	1,01100　$y_2 - y_3 = 1$，加$[x]_{补}$	→010
00,1011 右移一位 00,0101 +11,0001	1,01100　$y_3 - y_4 = -1$，加$[-x]_{补}$	→1010
11,0110		1010

在乘数末尾添一个$y_{-1} = 0$

注意最后一定要扫描到乘数符号位！因此需做$n+1$次加法

因此$[xy]_{补} = 1,01101010 = [-150]_{补}$

21. x、y为定点整数，其格式为1位符号位、n位数值位，若采用补码一位乘法实现乘法运算，则最多需要（ ）次加法运算。
A. $n-1$　　B. n　　C. $n+1$　　D. $n+2$

9.定点数除法运算

- 在本书中，我们只需要考虑被除数x和除数y都是$n+1$位纯小数，只需得到一个$n+1$位的纯小数商，且必须有$|x|<|y|$。

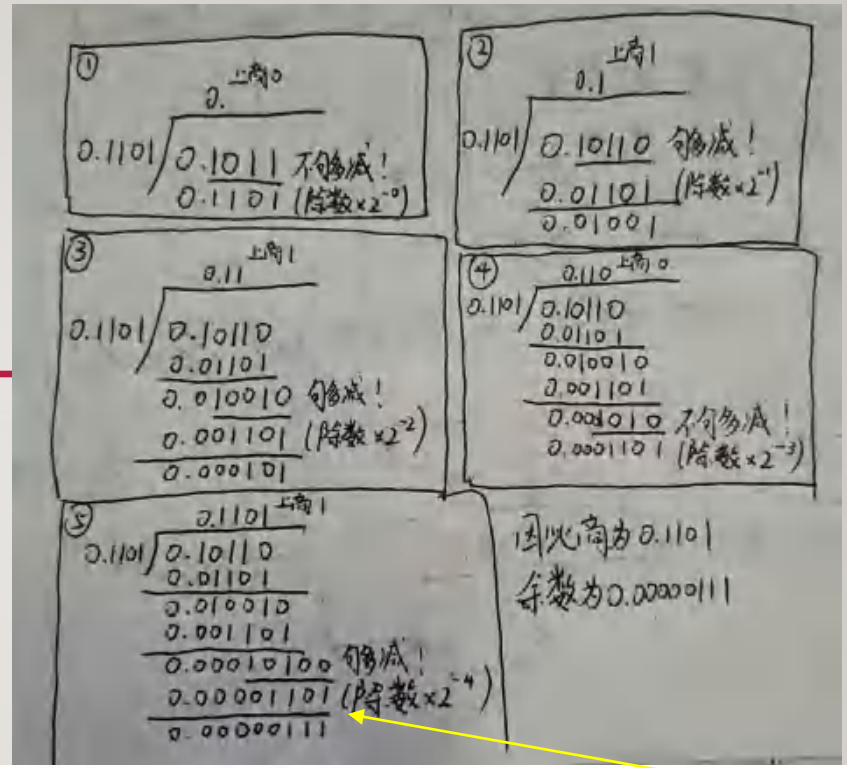
- **恢复余数法**：仅计算$\frac{|x|}{|y|}$，不断尝试将当前被除数的剩余部分(余数)R减去除数y^*，如果能减$(R-y^*\geq 0)$，就在这一位上商1并且$R\leftarrow R-y^*$，如果不能减$(R-y^*<0)$就上商0。这一轮做完后令R左移一位，即$R\leftarrow 2R$。（**由常规的除法竖式直接导出**）

最终得到的结果仍需像原码一位乘一样配符号

- **加减交替法**：对恢复余数法的简化(不恢复余数)，维护一个R'为这一轮**已经减过除数y^*的余数R**，若$R'\geq 0$说明这一轮余数能减，就在这一位上商1并且$R'\leftarrow 2R'-y^*$，否则说明这一轮余数不能减，上商0并且$R'\leftarrow 2R-y^*=2(R'+y^*)-y^*=2R'+y^*$。

这里自动同时做了恢复余数、左移和下一轮减除数

因此根据R'的符号决策是加y^*还是$-y^*$



例：对被除数$x=0.101$和除数$y=0.111$做除法。

$[x]_{补}=0.101$，$[y^*]_{补}=0.111$，$[-y^*]_{补}=1.001$

被除数剩余部分R'	操作	商
0.101 +1.001	开始令$R'=x-y^*$ (如果结果非负则除法溢出)	
1.110 左移一位1.100 +0.111	R'负，上商0，加$[y^*]_{补}$	0.
0.011 左移一位0.110 + 1.001	R'正，上商1，加$[-y^*]_{补}$	0.1
1.111 左移一位1.110 +0.111	R'负，上商0，加$[y^*]_{补}$	0.10
0.101	R'正，上商1	0.101

在算法中不断让被除数剩余部分左移，实际上是除数在右移，总之它俩要对齐

25. 在原码不恢复余数法（又称原码加减交替法）的算法中，（ ）。
A. 每步操作后，若不够减，则需恢复余数
B. 若为负商，则恢复余数
C. 整个算法过程中，从不恢复余数
D. 仅当最后一步不够减时，才恢复一次余数

被除数剩余部分并不是真正的余数！
若要得到最终余数可能需要对其加上除数进行恢复，并且还得右移n位

10.浮点数的表示方式

17. 设浮点数共12位。其中阶码含1位阶符共4位，以2为底，补码表示；尾数含1位数符共8位，补码表示，规格化。则该浮点数所能表示的最大正数是（ ）。
A. 2^7　　B. 2^8　　C. 2^8-1　　D. 2^7-1

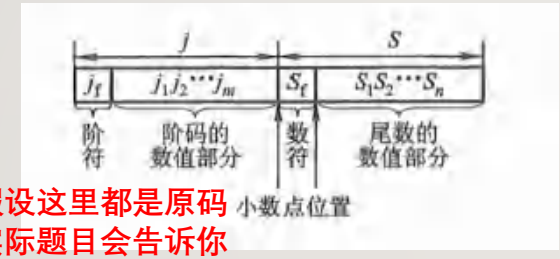
$j_{max}=2^3-1=7, S_{max}=1-2^{-7}, N_{max}=2^7-1$

11. 在浮点数编码表示中，（ ）在机器数中不出现，是隐含的。
A. 阶码　　B. 符号　　C. 尾数　　D. 基数

15. 在浮点运算中，下溢指的是（ ）。
A. 运算结果的绝对值小于机器所能表示的最小绝对值
B. 运算的结果小于机器所能表示的最小负数
C. 运算的结果小于机器所能表示的最小正数
D. 运算结果的最低有效位产生的错误

- 浮点数$N=S\times r^j$，S为**尾数**，j为**阶码**，r为**基数**(2,4,8…)
- 本书的抽象格式：

注意不要将其与现实中IEEE754标准搞混



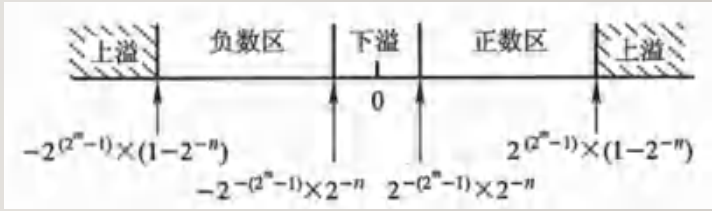
假设这里都是原码
实际题目会告诉你

阶码最大为2^m-1
最小为$-(2^m-1)$

尾数绝对值最大为$1-2^{-n}$，最小为2^{-n}

太小的不能表示，尾数/阶码位数有限

太大的不能表示，阶码位数有限



（对于原码而言）

- **规格化数**：强制令尾数除符号位外的最高位为1（绝对值$\in[2^{-1},1-2^{-n}]$），这样能够尽可能多地保留数字位数，精度更高。

13. 采用规格化的浮点数最主要是为了（ ）。
A. 增加数据的表示范围　　B. 方便浮点运算
C. 防止运算时数据溢出　　D. 增加数据的表示精度

- *IEEE754标准：尾数绝对值为1.xxxx的小数(规格化)，阶码用移码表示，尾数用原码表示，32位单精度数$|S|=24,|j|=8$，64位双精度数$|S|=53,|j|=11$。

（本书对此并不做过多涉及）

阶码越长，范围越大；尾数越长，精度越高

02. 长度相同但格式不同的两种浮点数，假设前者阶码长、尾数短，后者阶码短、尾数长，其他规定均相同，则它们可表示的数的范围和精度为（ ）。
A. 两者可表示的数的范围和精度相同　　B. 前者可表示的数的范围大但精度低
C. 后者可表示的数的范围大且精度高　　D. 前者可表示的数的范围大且精度高

长度不变，基数越大，范围越大，但分布稀疏，精度更低

03. 长度相同，格式相同的两种浮点数，假设前者基数大，后者基数小，其他规定均相同，则它们可表示的数的范围和精度为（ ）。
A. 两者可表示的数的范围和精度相同　　B. 前者可表示的数的范围大但精度低
C. 后者可表示的数的范围大且精度高　　D. 前者可表示的数的范围大且精度高

11.浮点数加减运算

28.【2015 统考真题】下列有关浮点数加减运算的叙述中，正确的是（ ）。
I. 对阶操作不会引起阶码上溢或下溢　大阶都没溢出，这肯定不会
II. 右规和尾数舍入都可能引起阶码上溢　尾数舍入可能再次右规
III. 左规时可能引起阶码下溢
IV. 尾数溢出时结果不一定溢出　尾数溢出时右规即可
A. 仅II、III　B. 仅I、II、IV　C. 仅I、III、IV　D. I、II、III、IV

07. 下列关于对阶操作说法正确的是（ ）。
A. 在浮点加减运算的对阶操作中，若阶码大小，则尾数左移
B. 在浮点加减运算的对阶操作中，若阶码增大，则尾数右移；若阶码减小，则尾数左移
C. 在浮点加减运算的对阶操作中，若阶码增大，则尾数右移
D. 以上都不对　一定要搞清楚在数值不变的情况下尾数移位时阶码怎么变！

- 浮点数加减运算的5个步骤：（题目中尾数一般都是补码）
  - ①**对阶**，将阶码小的**增大阶码**并**右移尾数**，使两数阶码对齐。（“小阶向大阶看齐”）
  - ②**尾数求和**，一般用双符号位补码加减。
  - ③**规格化**，使得结果尾数S满足$\frac{1}{2} \le |S| < 1$，正数$[S]_{补} = 00.1xxxx$，负数$[S]_{补} = 11.0xxxx$
    - 特别地，$S = -1$时，$[-1]_{补} = 11.0000$，尽管不满足$\frac{1}{2} \le |S| < 1$，但**位满足性质，认为是规格化数**。
      $S = -\frac{1}{2}$时，$[-\frac{1}{2}]_{补} = 11.1000$，尽管满足$\frac{1}{2} \le |S| < 1$，但**位不满足性质，不认为是规格化数**！
    - 当加法溢出得到01.xxxx或10.xxxx时，进行**右规**，尾数右移1位，阶码+1。
    - 当结果为11.1xxx或00.0xxx时，进行**左规**，尾数左移，阶码减小，直到满足规格化为止。
  - ④**舍入**，一般用于右规后的结果，也可以用于对阶时右移的尾数，将右移出的部分舍入掉。
    - **0舍1入**法：右移出的最高位是1，则在**末位+1**，道理同四舍五入。$0.001110 \to 0.0100$　$0.001101 \to 0.0011$
    - **恒置1**法：只要右移出至少1位，无论如何都将末位置1。$0.001110 \to 0.0011$　$0.001000 \to 0.0011$
  - ⑤**溢出判断**，判断阶码是否超出范围，若下溢直接置机器0，若上溢可能要引发异常。

为了使尾数作纯小数加减

18. 若浮点数的尾数用补码表示，则下列（ ）中的尾数是规格化数形式。
A. 1.11000　B. 0.01110　C. 0.01010　D. 1.00010

必须保证小数点后1位与符号位相反

这主要是为方便硬件判断而规定的，并且这俩特例是补码特有的，原码只需满足绝对值至少$\frac{1}{2}$即可

遇到$-\frac{1}{2}$时需要将其左规变为-1

05. 在规格化浮点运算中，若某浮点数为 $2^{-5} \times 1.10101$，其中尾数为补码表示，则该数（ ）。
A. 不需规格化　B. 需右移规格化
C. 需将尾数左移一位规格化　D. 需将尾数左移两位规格化

除非结果是0，否则一定可以左规成功

可能需要再次进行右规！

11.浮点数加减运算-Example

22.【2009 统考真题】浮点数加、减运算过程一般包括对阶、尾数运算、规格化、舍入和判断溢出等步骤。设浮点数的阶码和尾数均采用补码表示，且位数分别为 5 和 7（均含 2 位符号位）。若有两个数 $X=2^7\times29/32$ 和 $Y=2^5\times5/8$，则用浮点加法计算 $X+Y$ 的最终结果是（ ）。

A. 00111 1100010　B. 00111 0100010　C. 01000 0010001　D. 发生溢出

- $X = 2^{00,111} \times 00.11101$，$Y = 2^{00,101} \times 00.10100$

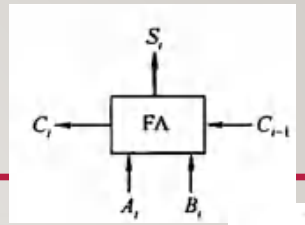
一般题目构造的数不会让你在对阶时就考虑舍入……

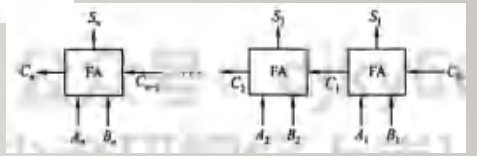
- ①将阶码更小的Y与X对阶：$Y = 2^{00,111} \times 00.00101$，尾数右移两位（缩小4倍），阶码+2
- ②尾数相加：$00.11101 + 00.00101 = 01.00010$，则$X + Y = 2^{00,111} \times 01.00010$
- ③加法溢出，则右规：$2^{00,111} \times 01.00010 \xrightarrow{\text{尾数右移1，阶码+1}} 2^{01,000} \times 00.100010$
- ④舍入：若采取0舍1入，右规时移出的是0，直接舍，$X + Y = 2^{01,000} \times 00.10001$
- ⑤判断溢出：阶码为$01,000$，正溢出，因此结果上溢！！

12.加法器与并行进位链

07. 加法器采用并行进位的目的是（ ）。
A. 增强加法器功能　　B. 简化加法器设计
C. 提高加法器运算速度　　D. 保证加法器可靠性

01. 并行加法器中，每位全和的形成除与本位相加二数数值位有关外，还与（ ）有关。
A. 低位数值大小　　B. 低位数的全和
C. 高位数值大小　　D. 低位数送来的进位





并行加法器

- 一位**全加器**：输入A_i, B_i以及上一位进位C_{i-1}，输出这一位结果S_i以及进位C_i
  - $S_i = A_i \oplus B_i \oplus C_{i-1}$，$C_i = A_iB_i + (A_i + B_i)C_{i-1}$
    本地进位d_i(G)　传递条件t_i(P)
    必须知道上一位进位才可得出这一位结果，进位是瓶颈
- **串行进位**加法器：简单地将n个全加器按进位级联形成n位加法器，进位信号需要串行地一级一级传播，速度慢。

03. 在串行进位的并行加法器中，影响加法器运算速度的关键因素是（ ）。
A. 门电路的级延迟　　B. 元器件速度
C. 进位传递延迟　　D. 各位加法器速度的不同

- **并行进位**加法器：将进位信号C_i的递推式**展开**为一个关于前面所有位输入的巨大表达式，这个表达式可以使用复杂的多输入与/或门构造，进位可并行产生。
- **单重分组跳跃进位**加法器：每组内并行进位，组与组之间进位信号串行级联，还是慢。
- **双重分组跳跃进位**加法器：

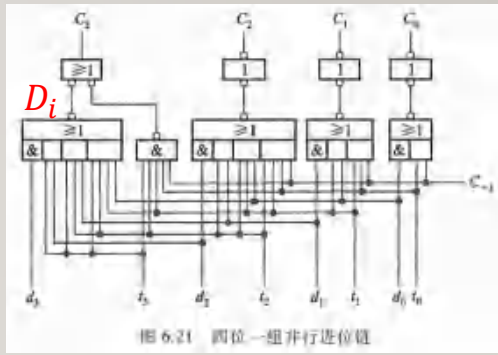


图 6.21　四位一组并行进位链

$d_i = A_iB_i, t_i = A_i + B_i$
$C_0 = d_0 + t_0C_{-1}$
$C_1 = d_1 + t_1d_0 + t_1t_0C_{-1}$
$C_2 = d_2 + t_2d_1 + t_2t_1d_0 + t_2t_1t_0C_{-1}$
$C_3 = d_3 + t_3d_2 + t_3t_2d_1 + t_3t_2t_1d_0 + t_3t_2t_1t_0C_{-1}$
D_i　　T_i

74181/182的G和P引脚相当于D_i和T_i
（考试或许会出，这图记住就好）

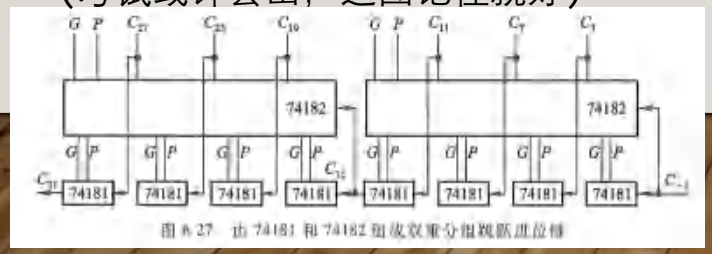


图 6.27　由 74181 和 74182 组成双重分组跳跃进位链

$D_7 + T_7D_8 + T_7T_8C_{-1}$　　$D_8 + T_8C_{-1}$

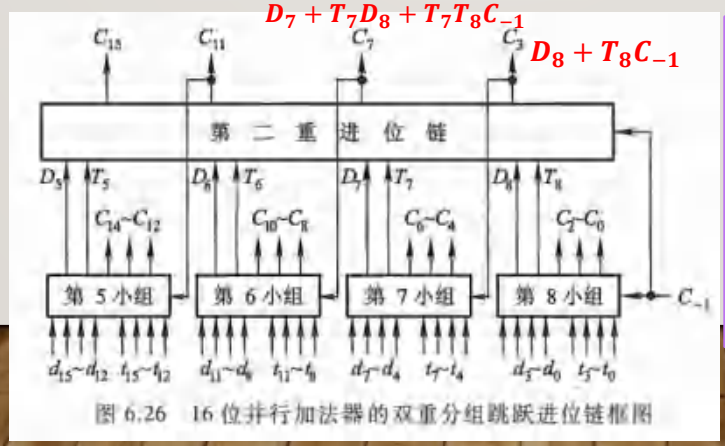
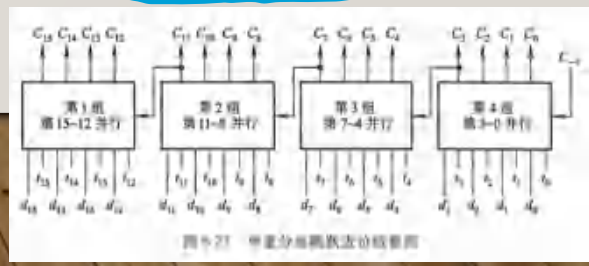


图 6.26　16 位并行加法器的双重分组跳跃进位链框图

D_i, T_i仅与这一组的输入有关，与进位无关
①同时产生D_i, T_i
②大组产生进位输出C_3, C_7, C_{11}, C_{15}，将它们输入给小组
③小组产生每一位的结果
这样组之间可并行工作！



13.指令系统概述

01. 以下有关指令系统的说法中，错误的是（ ）。
A. 指令系统是一台机器硬件能执行的指令全体
B. 任何程序运行前都要先转换为机器语言程序
C. 指令系统是计算机软/硬件的界面
D. 指令系统和机器语言是无关的

- **指令系统**是全部机器指令的集合，反映机器的功能。
- 指令由操作码和地址码构成，**操作码**反映操作类型，位数反映指令种类数，**地址码**可以是立即数、寄存器地址、存储器地址，反映指令操作数。指令的操作数类型可以是地址、数字、字符和逻辑数据。指令的长度为**指令字长**，可固定也可变。
- 指令的操作类型包括：**数据传送**、**算术逻辑操作**、**移位**、转移(**无条件转移**、**条件转移**、**调用**、**返回**、**陷阱**)、**输入输出**和**特权指令**等。

04. 程序控制类指令的功能是（ ）。
A. 进行算术运算和逻辑运算
B. 进行主存与CPU之间的数据传送
C. 进行CPU和I/O设备之间的数据传送
D. 改变程序执行的顺序

06. 下列指令中应用程序不准使用的指令是（ ）。
A. 循环指令　B. 转换指令　C. 特权指令　D. 条件转移指令

需要在某个寄存器/内存位置/栈顶保存返回地址

- **CISC**(复杂指令系统计算机)和**RISC**(精简指令系统计算机)：RISC选用频度高的简单指令，指令长度固定，指令格式和寻址方式少，使用load/store访存，多个通用寄存器，采用流水线使得大部分指令花费一个周期(CPI趋近于1)，对编译优化友好，时钟周期短，易于设计。

04. 【2009统考真题】下列关于RISC的说法中，错误的是（ ）。
A. RISC普遍采用微程序控制器
B. RISC大多数指令在一个时钟周期内完成
C. RISC的内部通用寄存器数量相对CISC多
D. RISC的指令数、寻址方式和指令格式种类相对CISC少

RISC使用组合逻辑进行控制

02. 下列描述中，不符合RISC指令系统特点的是（ ）。
A. 指令长度固定，指令种类少
B. 寻址方式种类尽量减少，指令功能尽可能强
C. 增加寄存器的数目，以尽量减少访存次数
D. 选取使用频率最高的一些简单指令，以及很有用但不复杂的指令

CISC指令功能是比较强的

05. 【2011统考真题】下列指令系统的特点中，有利于实现指令流水线的是（ ）。
I. 指令格式规整且长度一致
II. 指令和数据按边界对齐存放
III. 只有Load/Store指令才能对操作数进行存储访问
A. 仅I、II　B. 仅II、III　C. 仅I、III　D. I、II、III

14.寻址方式

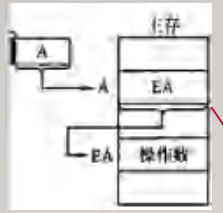
10. 相对寻址方式中，指令所提供的相对地址实质上是一种（ ）。
A. 立即数
B. 内存地址
C. 以本条指令在内存中首地址为基准位置的偏移量
D. 以下条指令在内存中首地址为基准位置的偏移量

24.【2016 统考真题】某指令格式如下所示。

OP	M	I	D

其中 M 为寻址方式，I 为变址寄存器编号，D 为形式地址。若采用先变址后间址的寻址方式，则操作数的有效地址是（ ）。
A. I+D　　B. (I)+D　　C. ((I)+D)　　D. ((I))+D

20.【2011 统考真题】偏移寻址通过将某个寄存器的内容与一个形式地址相加来生成有效地址。下列寻址方式中，不属于偏移寻址方式的是（ ）。
A. 间接寻址　　B. 基址寻址　　C. 相对寻址　　D. 变址寻址



- 指令操作数的寻址需要在指令中给出**寻址特征**和**形式地址**(A)字段，操作数实际地址为EA。
- ①**立即寻址**，A不是操作数地址，而是操作数本身(立即数)。

05. 在指令寻址的各种方式中，获取操作数最快的方式是（ ）。
A. 直接寻址　　B. 立即寻址　　C. 寄存器寻址　　D. 间接寻址

04. 简化地址结构的基本方法是尽量采用（ ）。
A. 寄存器寻址　　B. 隐地址　　C. 直接寻址　　D. 间接寻址

- ②**直接寻址**，**EA=A**，访问1次主存，A字段位数限制寻址范围，只能访问固定地址，不灵活。
- ③**隐含寻址**，操作数地址隐含在操作码中，例如ADD隐含操作数在ACC中，能缩短指令长度。
- ④**间接寻址**，**EA=(A)**，实际地址在存储器中，分为1次和多次(需要**额外1位**来判断是否为最终地址)，可以将操作数寻址范围扩大到存储字长，但需要**至少2次**访存。

03. 为了缩短指令中某个地址段的位数，有效的方法是采取（ ）。
A. 立即寻址　　B. 变址寻址　　C. 基址寻址　　D. 寄存器寻址

- ⑤**寄存器寻址**，**EA=Ri**，操作数在指定编号的寄存器中，无需访存，且指令长度短。
- ⑥**寄存器间接寻址**，**EA=(Ri)**，比间接寻址少访存一次。
- ⑦**基址寻址**，**EA=A+(BR)**，BR是专用的**基址寄存器**(用户一般**不可**修改)，或者可显式指定通用寄存器，可以实现主存空间分段以及多道程序等(操作系统管理)。
- ⑧**变址寻址**，**EA=A+(IX)**，IX是专用的**变址寄存器**(用户**可**修改)，或者可显式指定通用寄存器，地址A应该**保持不变**，可以实现数组访问(A为数组起始地址，IX为变化的下标)。

08. （ ）便于处理数组问题。
A. 间接寻址　　B. 变址寻址　　C. 相对寻址　　D. 基址寻址

对于取指来说又称跳跃寻址

- ⑨**相对寻址**，**EA=(PC)+A**，相对当前程序地址进行寻址，A作为可正可负的位移量使用补码表示，可以实现位置无关的浮动程序代码。**注意：这里的PC是取完指令进行自增后的PC！要考虑当前指令长度！**
- ⑩**堆栈寻址**，堆栈指针寄存器SP指向栈顶元素，入栈和出栈会给SP增量。

09. 堆栈寻址方式中，设 A 为累加器，SP 为堆栈指示器，M_{sp} 为 SP 指示的栈顶单元，若进栈操作的动作顺序是 (A)→M_{sp}，(SP)−1→SP，则出栈操作的动作顺序是（ ）。
A. $(M_{sp})\rightarrow A$，$(SP)+1\rightarrow SP$　　B. $(SP)+1\rightarrow SP$，$(M_{sp})\rightarrow A$

无论如何入栈出栈必须对称

14.寻址方式-一些题

19.【2009 统考真题】某机器字长为 16 位，主存按字节编址，转移指令采用相对寻址，由 2 字节组成，第一字节为操作码字段，第二字节为相对位移量字段。假定取指令时，每取一字节 PC 自动加 1。若某转移指令所在主存地址为 2000H，相对位移量字段的内容为 06H，则该转移指令成功转移后的目标地址是（ ）。

A. 2006H　　B. 2007H　　C. 2008H　　D. 2009H

取完指令后新PC=2000H+2=2002H
跳转直接在新PC上+6，为2008H

17. 设相对寻址的转移指令占 3B，第一字节为操作码，第二、三字节为相对位移量（补码表示），而且数据在存储器中采用以低字节为字地址的存放方式，每当 CPU 从存储器取出一字节时，即自动完成(PC) + 1→PC。若 PC 的当前值为 240（十进制），要求转移到 290（十进制），则转移指令的第二、三字节的机器代码是（ ）；若 PC 的当前值为 240（十进制），要求转移到 200（十进制），则转移指令的第二、三字节的机器代码是（ ）。

A. 2FH、FFH　　B. D5H、00H　　C. D5H、FFH　　D. 2FH、00H

取完指令后新PC=240+3=243
转移到290的位移量为290-243=47=002FH，2FH和00H
转移到200的位移量为200-243=-43=FFD5H，D5H和FFH

07. 一条双字长的取数指令（LDA）存于存储器的 200 和 201 单元，其中第一个字为操作码 OP 和寻址特征 M；第二个字为形式地址 A。假设 PC 的当前值为 200，变址寄存器 IX 的内容为 100，基址寄存器的内容为 200，存储器相关单元的内容如下表所示：

地址	201	300	400	401	500	501	502	700
内容	300	400	700	501	600	700	900	401

新PC=202

A

下表的各列分别为寻址方式、该寻址方式下的有效地址及取数指令执行结束后累加器（AC）的内容，试补全下表：

寻址方式	有效地址（EA）	累加器（AC）的内容
立即寻址		300
直接寻址	300	400
间接寻址	(300)=400	700
相对寻址	202+300=502	900
变址寻址	300+100=400	700
基址寻址	300+200=500	600
先变址后间址	(300+100)=700	401
先间址后变址	(300)+100=500	600

15.指令格式设计

图 4.1 扩展操作码技术

4位操作码1111未用，作为8位操作码前缀

8位操作码11111111未用，作为12位操作码前缀

11. 某指令系统有 200 条指令，对操作码采用固定长度二进制编码，最少需要用（ ）位。
A. 4 B. 8 C. 16 D. 32

- n 位字段能够表示**至多 2^n** 个模式，这可以用于操作码、寄存器编号、寻址特征位等。
- 操作码是一种前缀编码，短操作码不能是长操作码的**前缀**，否则无法区分指令。设计操作码时要先分配短操作码，再将**未使用**的短码作为长操作码的前缀。**“扩展操作码”**
- 注意考虑指令字长、指令中的字段、每个字段有多少位。

12. 在指令格式中，采用扩展操作码设计方案的目的是（ ）。
A. 减少指令字长度
B. 增加指令字长度
C. 保持指令字长度不变而增加指令的数量
D. 保持指令字长度不变而增加寻址空间

28.【2020 统考真题】某计算机采用 16 位定长指令字格式，操作码位数和寻址方式位数固定，指令系统有 48 条指令，支持直接、间接、立即、相对 4 种寻址方式。在单地址指令中，直接寻址方式的可寻址范围是（ ）。

指令码6位，寻址特征位2位，地址码8位

A. 0～255 B. 0～1023 C. −128～127 D. −512～511

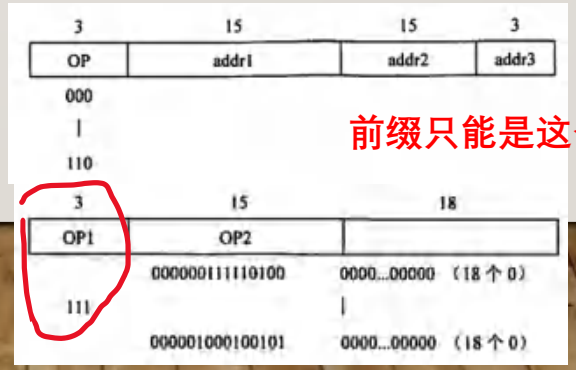
指令字长必为8倍数

14.【2017 统考真题】某计算机按字节编址，指令字长固定且只有两种指令格式，其中三地址指令 29 条，二地址指令 107 条，每个地址字段为 6 位，则指令字长至少应该是（ ）。

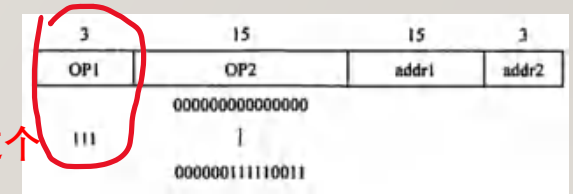
短操作码至少5位

A. 24 位 B. 26 位 C. 28 位 D. 32 位

03. 在一个 36 位长的指令系统中，设计一个扩展操作码，使之能表示下列指令：
1）7 条具有两个 15 位地址和一个 3 位地址的指令。
2）500 条具有一个 15 位地址和一个 3 位地址的指令。
3）50 条无地址指令。



前缀只能是这个



短操作码还剩$2^6-29=35$个未用，它们可以作为长操作码的前缀，长操作码可以有$35\times 2^{12-6}=2240$个，因此还可以再添加$2240-107=2133$条二地址指令！

16.指令周期

15. （ ）可区分存储单元中存放的是指令还是数据。
A. 控制器　B. 运算器　C. 存储器　D. 数据通路

17. 【2009 统考真题】冯·诺依曼计算机中指令和数据均以二进制形式存放在存储器中，CPU 区分它们的依据是（ ）。
A. 指令操作码的译码结果　B. 指令和数据的寻址方式
C. 指令周期的不同阶段　D. 指令和数据所在的存储单元

取指阶段从存储器取出的是指令
执行阶段从存储器取出的是数据

07. 以下叙述中，错误的是（ ）。
A. 指令周期的第一个操作是取指令
B. 为了进行取指操作，控制器需要得到相应的指令
C. 取指操作是控制器自动进行的
D. 指令执行时有些操作是相同或相似的

04. 指令（ ）从主存中读出。
A. 总是根据程序计数器　B. 有时根据程序计数器，有时根据转移指令
C. 根据地址寄存器　D. 有时根据程序计数器，有时根据地址寄存器

- **指令周期**是CPU从开始取指到指令执行结束的时间，被划分为四个阶段：

必须有 • **取指周期**，PC->MAR->M->MDR->IR，PC+1->PC

05. 在一条无条件跳转指令的指令周期内，程序计数器（PC）的值被修改了（ ）次。
A. 1　B. 2　C. 3　D. 不能确定

书上明确说了取指时必须自增PC

仅间址寻址指令有 • **间址周期**，取出间址寻址的有效地址，(1次间址)Ad(IR)->MAR->M->MDR->Ad(IR)

必须有 • **执行周期**，指令真正的执行过程，对不同指令都是不同的。

13. CPU 响应中断的时间是（ ）。
A. 一条指令执行结束　B. I/O 设备提出中断
C. 取指周期结束　D. 指令周期结束

仅有中断响应时有 • **中断周期**，检查并响应中断，保存PC(特定位置或栈顶)，向量地址->PC，关中断

机器周期通常可以由较慢的存取时间确定

- 多级时序系统：指令周期由多个**机器周期**构成，每个机器周期对应一个阶段，机器周期由多个**时钟周期(节拍)**构成，时钟周期是最小时间单位。

01. 计算机工作的最小时间周期是（ ）。
A. 时钟周期　B. 指令周期　C. CPU 周期　D. 工作脉冲

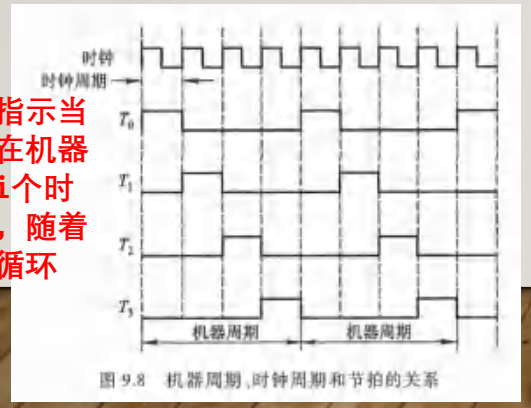


图 9.8　机器周期、时钟周期和节拍的关系

节拍信号指示当前是否处在机器周期的第i个时钟周期内，随着机器周期循环

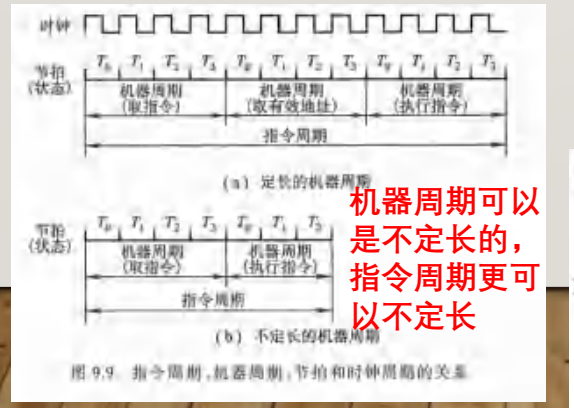


图 9.9　指令周期、机器周期、节拍和时钟周期的关系

机器周期可以是不定长的，指令周期更可以不定长

11. 下列说法中，合理的是（ ）。
A. 执行各条指令的机器周期数相同，各机器周期的长度均匀
B. 执行各条指令的机器周期数相同，各机器周期的长度可变
C. 执行各条指令的机器周期数可变，各机器周期的长度均匀
D. 执行各条指令的机器周期数可变，各机器周期的长度可变

09. 由于 CPU 内部操作的速度较快，而 CPU 访问一次存储器的时间较长，因此机器周期通常由（ ）来确定。
A. 指令周期　B. 存取周期　C. 间址周期　D. 中断周期

若机器周期定长，则取决于最慢的阶段

10. 以下有关机器周期的叙述中，错误的是（ ）。
A. 通常把通过一次总线事务访问一次主存或 I/O 的时间定为一个机器周期
B. 一个指令周期通常包含多个机器周期
C. 不同的指令周期所包含的机器周期数可能不同
D. 每个指令周期都包含一个中断响应机器周期

17.流水线

M个阶段对应部件

这样的图需要会画

图 8.18 各段时间相等的流水线时空图

14.【2013 统考真题】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（ ）。 1.0×10^9条指令/s

20.【2020 统考真题】下列给出的处理器类型中，理想情况下，CPI 为 1 的是（ ）。
I. 单周期 CPU II. 多周期 CPU III. 基本流水线 CPU IV. 超标量流水线 CPU
A. 仅 I、II B. 仅 I、III C. 仅 II、IV D. 仅 III、IV

18.【2018 统考真题】[illegible] 5 个子功能 [illegible] 90ps、50ps、50ps、70ps 和 50ps，[illegible] 20ps。[illegible] CPU 时钟周期至少为（ ）。
A. 60ps B. 70ps C. 80ps D. 100ps

12.【2009 统考真题】某计算机的指令流水线由 4 个功能段组成，[illegible] 90ns、80ns、70ns 和 60ns，[illegible] CPU 周期至少是（ ）。
A. 90ns B. 80ns C. 70ns D. 60ns

- **流水线**将指令的执行阶段划分为多个使用独立器件的阶段，每个阶段时间为Δt(取决于**最慢**的阶段)，可以使连续多条指令在处理器上重叠执行。影响流水线的3种相关：

这些阶段之间要插入寄存器

13.【2010 统考真题】下列不会引起指令流水线阻塞的是（ ）。
A. 数据旁路 B. 数据相关 C. 条件转移 D. 资源冲突

  - **结构相关**：不同指令在重叠执行时争用同一部件，主要是访存指令和下一条指令的取指对于存储器的竞争。可以暂停下条指令的取指，或将指令和数据分开存储，或在执行时用队列进行指令预取。
  - **数据相关**：在正常的按序流水线中因对同一寄存器先写后读(RAW)引起(非按序也可WAW/WAR)，可以暂停后续指令执行，或使用**定向(旁路)**，不等结果送回寄存器时就给后续指令使用。
  - **控制相关**：由转移指令引起，干扰后续指令的正常取指，可以暂停+分支预测，尽早生成转移目标地址。

18. [illegible]
A. [illegible]
B. [illegible]
C. [illegible]
D. [illegible]

- 衡量流水线性能的3个量：**吞吐率**为单位时间内完成的指令数，理想情况最大吞吐率$T_{pmax}=\frac{1}{\Delta t}$，$n$条指令$m$阶段的实际吞吐率$T_p=\frac{n}{m\Delta t+(n-1)\Delta t}=\frac{T_{pmax}}{1+\frac{m-1}{n}}$。**加速比**$S_p=\frac{\text{非流水线时间}}{\text{流水线时间}}=\frac{nm\Delta t}{m\Delta t+(n-1)\Delta t}=\frac{m}{1+\frac{m-1}{n}}$，**效率**$E=\frac{\sum\text{每个阶段工作时间}}{\sum\text{每个阶段总时间}}=\frac{m\times n\Delta t}{m\times(m\Delta t+(n-1)\Delta t)}=\frac{n}{1+\frac{m-1}{n}}$。

流水线实际时间为$m\Delta t+(n-1)\Delta t$

04. 流水 CPU 是由一系列称为“段”的处理线路组成的。一个 m 段流水线稳定时的 CPU 的吞吐能力，与 m 个并行部件的 CPU 的吞吐能力相比，（ ）。
A. 具有同等水平的吞吐能力

- 优化流水线的3种高级技术：**超标量**(每个时钟周期同时并发多条独立指令，需要多个执行部件)，**超流水**(将流水线中的每个阶段继续流水线化)，**超长指令字**(将多条能并行的指令编译为一条超长指令)。 都需要编译器配合优化

02. 下列关于超标量流水线的描述中，不正确的是（ ）。
A. 在一个时钟周期内一条流水线可执行一条以上的指令
B. 一条指令分为多段指令由不同电路单元完成
C. [illegible]
D. [illegible]

16.【2017 统考真题】下列关于超标量流水线特性的叙述中，正确的是（ ）。
I. [illegible]
II. 能在一个时钟周期内同时发射多条指令
III. [illegible]
A. 仅 II B. 仅 I、III C. 仅 II、III D. I、II 和 III

11. 关于流水线技术的说法中，错误的是（ ）。
A. [illegible]
C. [illegible] RAW、WAR 和 WAW 中，[illegible] RAW [illegible]
D. [illegible]

18.指令微操作

这些微操作的标准写法需要掌握其套路，作为八股，考完扔了即可

- 取指周期：①PC→MAR，②1→R，③M(MAR)→MDR，④MDR->IR，⑤OP(IR)->CU，⑥(PC)+1→PC
- 间址周期：①Ad(IR)->MAR，②1→R，③M(MAR)→MDR，④MDR→Ad(IR)
- 执行周期对不同的指令不同：（**以下唐书制造的指令需要大家记住**）
  - 清除累加器指令CLA：0→ACC
  - 累加器取反指令COM：$\overline{ACC}$→ACC
  - 算术右移1位指令SHR：L(ACC)→R(ACC)，ACC_0→ACC_0
  - 循环左移1位指令CSL：R(ACC)→L(ACC)，ACC_0→ACC_n
  - 停机指令STP：0→G

唐书在这里取0为最高位

需要访存，注意访存套路

  - 加法指令ADD X：①Ad(IR)→MAR，②1→R，③M(MAR)→MDR，④(ACC)+(MDR)→ACC
  - 存数指令STA X：①Ad(IR)→MAR，②1→W，③ACC→MDR，④MDR→M(MAR)
  - 取数指令LDA X：①Ad(IR)→MAR，②1→R，③M(MAR)→MDR，④MDR→ACC
  - 无条件转移指令JMP X：Ad(IR)→PC
  - 累加器负则转移指令BAN X：$ACC_0 \cdot Ad(IR) + \overline{ACC_0} \cdot PC \rightarrow PC$

02. 在组合逻辑控制器中，微操作控制信号的形成主要与（ ）信号有关。

A. 指令操作码和地址码　　B. 指令译码信号和时钟

C. 操作码和条件码　　D. 状态信息和条件

18.指令微操作：节拍划分

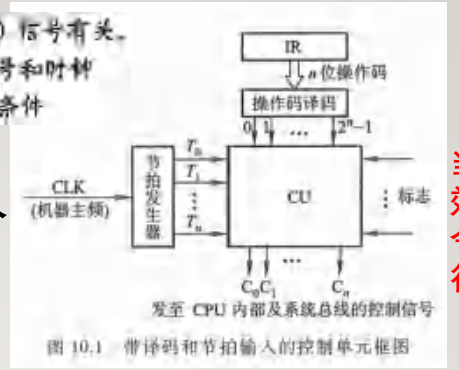


图 10.1　带译码和节拍输入的控制单元框图

当某个节拍信号有效时，进行这个指令在这个节拍应进行的微操作

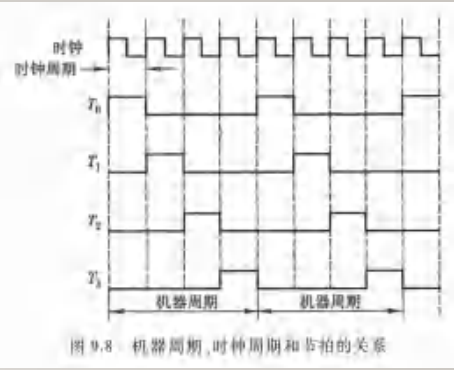


图 9.8　机器周期、时钟周期和节拍的关系

每个机器周期包含3个节拍 T_0, T_1, T_2，唐书将仅有1个微操作的指令安排在 T_2，T_0, T_1为空

- 基于组合逻辑的控制单元可以接受节拍信号的输入，在不同的节拍下生成对应的控制信号，使指令微操作按节拍有序进行，每个节拍可以容纳多个简单或独立操作。
- 取指周期：T_0{PC→MAR,1→R}，T_1{M(MAR)→MDR,(PC)+1→PC}，T_2{MDR→IR,OP(IR)→CU}
- 间址周期：T_0{Ad(IR)→MAR,1→R}，T_1{M(MAR)→MDR}，T_2{MDR→Ad(IR)}
- 执行周期：
  - 加法指令ADD X：T_0{Ad(IR)→MAR,1→R}，T_1{M(MAR)→MDR}，T_2{(ACC)+(MDR)→ACC}
  - 存数指令STA X：T_0{Ad(IR)→MAR,1→W}，T_1{ACC→MDR}，T_2{MDR→M(MAR)}
  - 取数指令LDA X：T_0{Ad(IR)→MAR,1→R}，T_1{M(MAR)→MDR}，T_2{MDR→ACC}
  - 无条件转移指令JMP X：T_2{Ad(IR)→PC}
  - 累加器负则转移指令BAN X：$T_2\{ACC_0 \cdot Ad(IR)+ACC_0 \cdot PC \rightarrow PC\}$

19.数据通路

05 【2016 统考真题】单周期处理器中所有指令的指令周期为一个时钟周期。下列关于单周期处理器的叙述中，错误的是（ ）。

A. 可以采用单总线结构数据通路 由于数据冲突无法在一个周期内实现

B. 处理器时钟频率较低

C. 在指令执行过程中控制信号不变

D. 每条指令的 CPI 为 1

03. 采用 CPU 内部总线的数据通路与不采用 CPU 内部总线的数据通路相比，（ ）。

A. 前者性能较高　　B. 后者的数据冲突问题较严重

C. 前者的硬件量大、实现难度高　　D. 以上说法都不对

- 数据通路可以有非总线方式和内部总线方式两种实现：

CU产生的控制信号控制数据通路开闭以及寄存器的输入输出，实现微操作

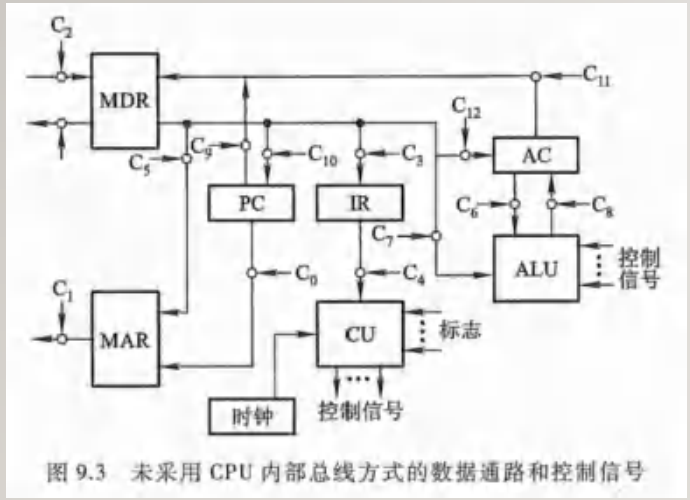
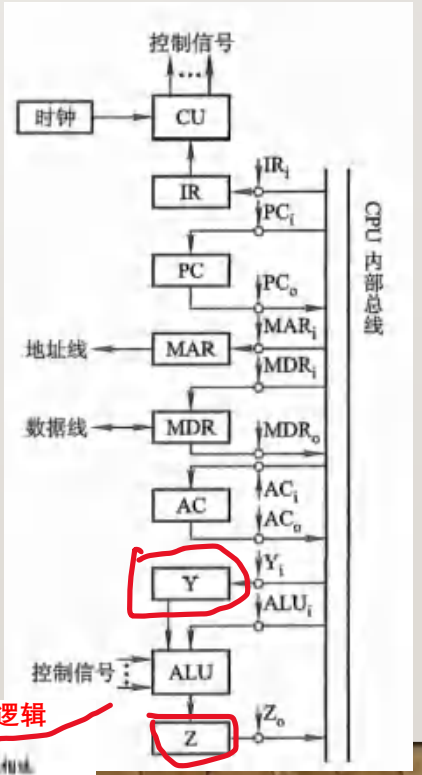


图 9.3　未采用 CPU 内部总线方式的数据通路和控制信号

取指周期:
①C_0有效，PC→MAR
②C_1有效，传送地址
③1→R，发出读命令
④C_2有效，M(MAR)→MDR
⑤C_3有效，MDR→IR
⑥C_4有效，OP(IR)→CU
⑦(PC)+1→PC
间址周期:
①C_5有效，MDR→MAR
(实际上取的是指令的Ad字段)
②C_1有效，传送地址
③1→R，发出读命令
④C_2有效，M(MAR)→MDR
⑤C_3有效，MDR→Ad(IR)

执行周期：(以间址加法ADD @X为例)
①C_5有效，MDR→MAR
(实际上取的是指令的Ad字段)
②C_1有效，传送地址
③1→R，发出读命令
④C_2有效，M(MAR)→MDR
⑤C_6, C_7有效，设置ALU输入
⑥对ALU发加信号
⑦C_8有效，结果送回ACC

内部总线节省连线，但可能有数据冲突



02. 在单总线的 CPU 中，（ ）。

A. ALU 的两个输入端及输出端都可与总线相连

B. 为 ALU 的两个输入端可与总线相连，但输出端需通过暂存器与总线相连

C. ALU 的一个输入端可与总线相连，其输出端也可与总线相连

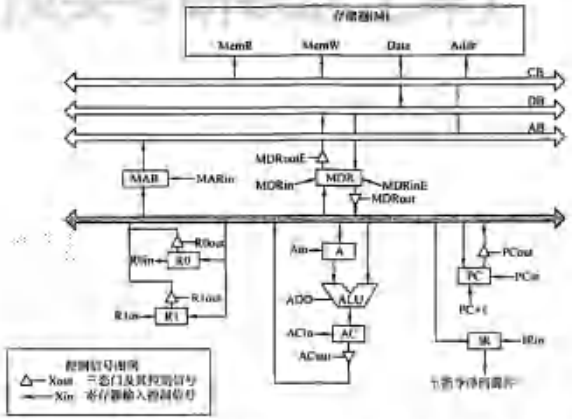
✓ D. ALU 只能有一个输入端可与总线相连，另一输入端需通过暂存器与总线相连

注意ALU是组合逻辑

取指周期:
①PC_o, MAR_i有效，PC→**Bus**→MAR
②1→R，发出读命令
③M(MAR)→MDR
④MDR_o, IR_i有效，MDR→**Bus**→IR，且CU开始译码
⑤(PC)+1→PC
间址周期:
①MDR_o, MAR_i有效，MDR→**Bus**→MAR
(实际上取的是指令的Ad字段)
②1→R，发出读命令
③M(MAR)→MDR
④MDR_o, IR_i有效，MDR→**Bus**→Ad(IR)
执行周期：(以间址加法ADD @X为例)
①MDR_o, MAR_i有效，MDR→**Bus**→MAR
(实际上取的是指令的Ad字段)
②1→R，发出读命令
③M(MAR)→MDR
④MDR_o, Y_i有效，MDR→**Bus**→Y
⑤AC_o, ALU_i有效，ACC→**Bus**→ALU输入
⑥对ALU发加信号，(ACC)+(Y)→Z
⑦Z_o, AC_i有效，Z→**Bus**→ACC

19.数据通路——一些照本宣科之题

01.【2009统考真题】某计算机字长16位，采用16位定长指令字结构，部分数据通路结构如下图所示，图中所有控制信号为1时表示有效，为0时表示无效。例如，控制信号MDRinE为1表示允许数据从DB打入MDR，MDRin为1表示允许数据从总线打入MDR。假设MAR的输出一直处于使能状态。加法指令"ADD (R1), R0"的功能为(R0)+((R1))→(R1)，即将R0中的数据与R1的内容所指主存单元的数据相加，并将结果送入R1的内容所指主存单元中保存。



下表给出了上述指令取指和译码阶段每个节拍（时钟周期）的功能和有效控制信号，请按表中描述方式用表格列出指令执行阶段每个节拍的功能和有效控制信号。

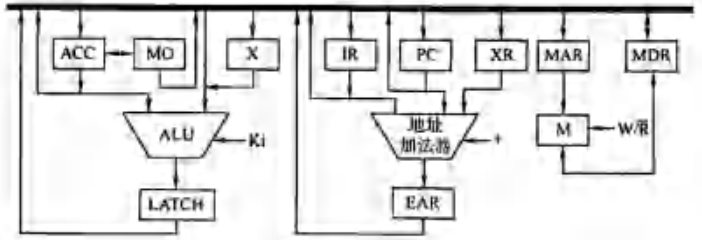
时　钟	功　能	有效控制信号
C1	MAR←(PC)	PCout, MARin
C2	MDR←M(MAR) PC←(PC)+1	MemR, MDRinE, PC+1
C3	IR←(MDR)	MDRout, IRin
C4	指令译码	无
C5	MAR←(R1)	R1out, MARin
C6	MDR←M(MAR)	MemR, MDRinE
C7	A←(MDR)	MDRout, Ain
C8	AC←(A)+(R0)	R0out, Add, ACin
C9	MDR←(AC)	ACout, MDRin
C10	M(MAR)←(MDR)	MDRoutE, MemW

07. 已知单总线计算机结构如下图所示，其中M为主存，XR为变址寄存器，EAR为有效地址寄存器，LATCH为暂存器，假设指令地址已存在于PC中，请给出ADD X, D指令周期信息流程和相应的控制信号。说明：

1）ADD X, D指令字中，X为变址寄存器XR，D为形式地址。

2）寄存器的输入/输出均采用控制信号控制，如PC_i表示PC的输入控制信号，MDR_o表示MDR的输出控制信号。

3）凡需要经过总线的传递，都需要注明，如(PC)→MAR，相应的控制信号为PC_o和MAR_i。



周　期	微　操　作	有效控制信号
取指周期	(PC)→MAR	PC_o, MAR_i
	M(MAR)→MDR (PC)+1→PC	MAR_o, R/W, MDR_i +1
	(MDR)→IR	MDR_o, IR_i
执行周期	(XR)+Ad(IR)→EAR	XR_o, IR_{ad}, +, EAR_i
	(EAR)→MAR	EAR_o, MAR_i
	M(MAR)→MDR	MAR_o, R/W, MDR_i
	(MDR)→X	MDR_o, X_i
	(ACC)+(X)→LATCH	ACC_o, X_o, K_i=+, $LATCH_i$
	(LATCH)→ACC	$LATCH_o$, ACC_i

07. 以下说法中，正确的是（ ）。
A. 采用微程序控制器是为了提高速度
B. 控制存储器由高速 RAM 电路组成
C. 微指令计数器决定指令执行顺序
D. 一条微指令存放在控制器的一个控制存储器单元中

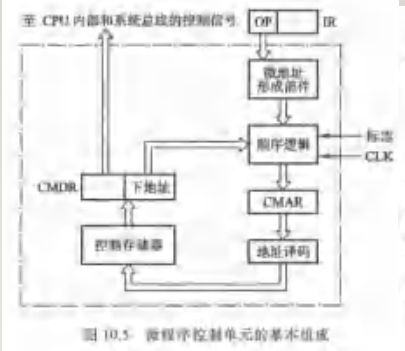


图 10.5 微程序控制单元的基本组成

20. 【2009 统考真题】相对于微程序控制器，硬布线控制器的特点是（ ）。
A. 指令执行速度慢，指令功能的修改和扩展容易
B. 指令执行速度慢，指令功能的修改和扩展难
C. 指令执行速度快，指令功能的修改和扩展容易
D. 指令执行速度快，指令功能的修改和扩展难

微程序CU可以修改控存来实现指令功能修改

03. 在微程序控制器中，形成微程序入口地址的是（ ）。
A. 机器指令的地址码字段
B. 微指令的微地址码字段
C. 机器指令的操作码字段
D. 微指令的微操作码字段

20.微指令与微程序

微程序CU是除了组合逻辑CU(硬布线控制器)之外的另一种实现形式

- 一条机器指令对应一个**微程序**，一个微程序包含若干条**微指令**，每个微指令表示一些微操作命令。
- 指令对应的微程序存储在**控制存储器**(ROM)中，对应的地址和数据寄存器为CMAR和CMDR，每条微指令都需要显式给出下条微指令的地址(下地址字段)，执行指令时先将指令对应的微程序首地址送入CMAR，循环以下步骤：①CM(CMAR)→CMDR，②由微指令发出微命令(控制信号)，③Ad(CMDR)→CMAR

05. 微程序控制器的速度比硬布线控制器慢，主要是因为（ ）。
D. 增加了从控制存储器读取微指令的时间

15. 在微程序控制方式中，以下说法正确的是（ ）。
I. 采用微程序控制器的处理器称为微处理器
II. 每条机器指令由一段微程序来解释执行
III. 在微指令的编码中，效率最低的是直接编码方式
IV. 水平型微指令能充分利用数据通路的并行结构
A. I、II　B. II、IV　C. I、III　D. III、IV

直接编码位数多，但执行效率高

- 微指令主要分为两种：
  - **水平型微指令**：一次并行执行多个微命令。
  - **垂直型微指令**：类似RISC，具有微操作码和地址码，仍然是“指令”，可进一步拆成毫微指令……

垂直型微指令位数少，1次能做的微命令少，微程序结构长

- 水平型微指令主要有这样两种编码方式：
  - **直接编码**方式：微指令每位表示一个微命令的控制信号。
  - **字段直接编码**方式：划分为若干段，每段使用译码器产生1个微命令(每段至多对应2^i-1种互斥的微命令，留1个状态表示不发任何信号) 最多同时发出段数个微命令

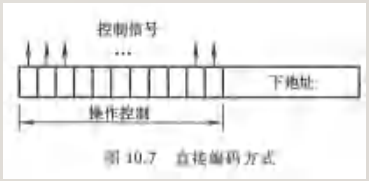


图 10.7 直接编码方式

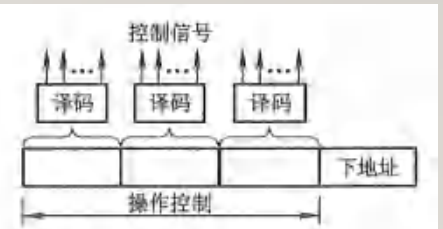


图 10.8 字段直接编码方式

21. 【2012 统考真题】某计算机的控制器采用微程序控制方式，微指令中的操作控制字段采用字段直接编码法。共有 33 个微命令，构成 5 个互斥类，分别包含 7、3、12、5 和 6 个微命令，则操作控制字段至少有（ ）。位数：3,2,4,3,3，注意每段多加1个空状态
A. 5 位　B. 6 位　C. 15 位　D. 33 位

22. 【2014 统考真题】某计算机采用微程序控制器，共有 32 条指令，公共的取指令微程序包含 2 条微指令，各指令对应的微程序平均由 4 条微指令组成，采用断定法（下地址字段法）确定下条微指令地址，则微指令中下地址字段的位数至少是（ ）。
A. 5　B. 6　C. 8　D. 9

控存中一共要存32*4+2=130条微指令，控存地址应为8位

20.微指令与微程序：指令微程序设计

每执行一条微指令都要在下一节拍设置CMAR

取指周期：

T_0 PC→MAR,1→R

T_1 **Ad(CMDR)→CMAR**

T_2 M(MAR)→MDR,(PC)+1→PC

T_3 **Ad(CMDR)→CMAR**

T_4 MDR→IR,OP(IR)→微地址形成部件

T_5 **微地址形成部件→CMAR**

指令JMP X执行周期：

T_0 Ad(IR)→PC

T_1 **Ad(CMDR)→CMAR(跳转到取指微程序)**

指令ADD X执行周期：

T_0 Ad(IR)→MAR,1→R

T_1 **Ad(CMDR)→CMAR**

T_2 M(MAR)→MDR

T_3 **Ad(CMDR)→CMAR**

T_4 (ACC)+(MDR)→ACC

T_5 **Ad(CMDR)→CMAR(跳转到取指微程序)**

2）写出硬布线控制器完成 STA X（X 为主存地址）指令发出的全部微操作命令及节拍安排。

3）若采用微程序控制，还需增加哪些微操作？

2）微操作命令及节拍安排如下：

T_0 PC→MAR，1→R

T_1 M(MAR)→MDR，(PC)+1→PC

T_2 MDR→IR，OP(IR)→ID

T_0 Ad(IR)→MAR，1→W

T_1 ACC→MDR

T_2 MDR→M(MAR)

3）若采用微程序控制，还需增加下列微操作：

取指周期：

Ad(CMDR)→CMAR

OP(IR)→CMAR

执行周期：

Ad(CMDR)→CMAR

21.存储器概述

01. 磁盘属于（ ）类型的存储器。
A. 随机存取存储器（RAM）　B. 只读存储器（ROM）
C. 顺序存取存储器（SAM）　✓D. 直接存取存储器（DAM）

- 按存取方式，存储器可分为**随机存储器**(RAM)、**只读存储器**(ROM)、**顺序存取存储器**(磁带)、**直接存取存储器**(磁盘)。
  - 任一单元都可随机存取，时间与物理位置无关
  - 需要按物理地址先后顺序寻址，耗时不同
  - 直接访问与顺序访问结合
- 按字寻址、按半字寻址、按字节寻址：将存储器按某一字长划分成若干个单元。
  - “字”一般指机器字长

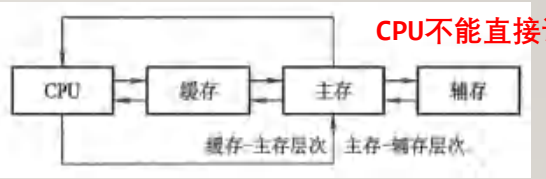
03. 设机器字长为32位，一个容量为16MB的存储器，CPU按半字寻址，其可寻址的单元数是（ ）。
A. 2^{24}　✓B. 2^{23}　C. 2^{22}　D. 2^{21}
2B

07. 设机器字长为64位，存储容量为128MB，若按字编址，它可寻址的单元个数是（ ）。
A. 16MB　✓B. 16M　C. 32M　D. 32MB

- 三级存储系统：**缓存-主存**层次解决CPU与主存速度不匹配的问题(纯硬件,Cache)，**主存-辅存**层次解决存储器容量问题(硬件+OS,分页虚拟存储)。

10. 在多级存储体系中，“Cache-主存”结构的作用是解决（ ）的问题。
A. 主存容量不足　B. 主存与辅存速度不匹配
C. 辅存与CPU速度不匹配　✓D. 主存与CPU速度不匹配

13. 下列关于多级存储系统的说法中，正确的有（ ）。
✓ 多级存储系统是为了降低存储成本　仅对应用程序透明
✗ 虚拟存储器中主存和辅存之间的数据调动对任何程序员是透明的
✗ CPU寄存器与Cache直接交换信息，CPU与主存或辅存总是要经过Cache
A. 仅I　B. 仅I和II　C. I、II和III　D. 仅II

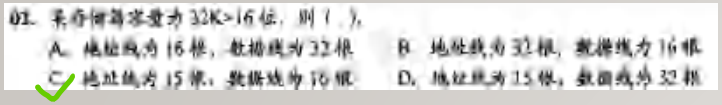


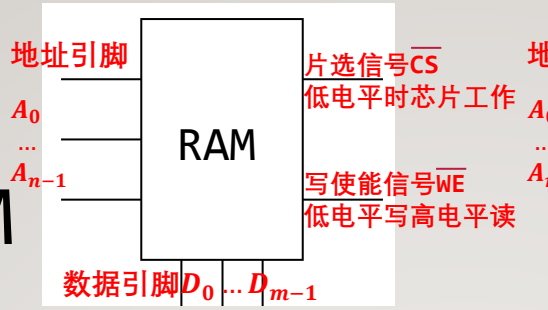
CPU不能直接访问辅存!

- **存取时间**为从发出读写命令到存储器完成操作的时间，**存取周期**为连续两次独立操作之间所需最小间隔(一般**长于**存取时间)，**存储器带宽**为单位时间内存储器存取的信息量，**取决于存取周期**(而不是存取时间!)。

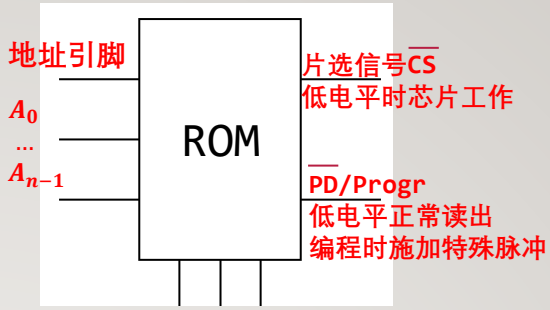
06. 若某存储器存储周期为250ns，每次读出16位，该存储器的数据传输率是（ ）。
A. 4×10^6B/s　B. 4MB/s　✓C. 8×10^6B/s　D. 8×2^{20}B/s

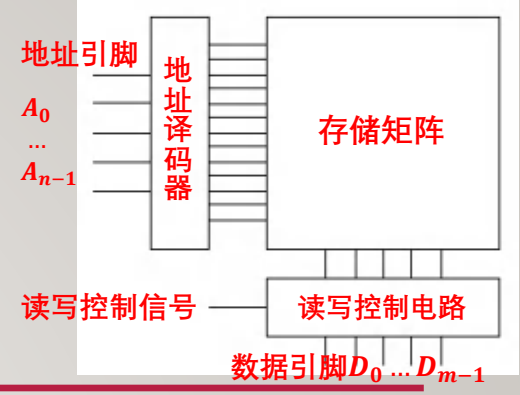
22.RAM与ROM





存储容量=$2^n \times m$位

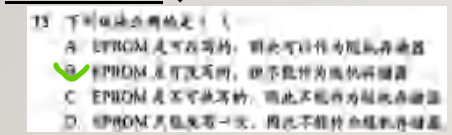


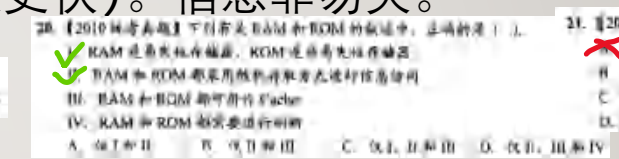


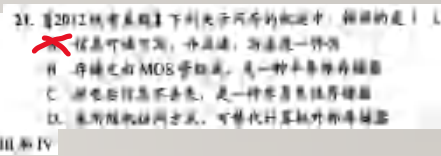
编程：彻底擦除并改写整块ROM内的信息

- ROM分为**掩膜ROM**(不可编程)、**PROM**(可编程一次)、**EPROM**(可紫外线擦除编程多次)、**EEPROM**(可电擦除编程多次)、**Flash**(类似EEPROM但速度更快)。信息非易失。

注意：ROM不属于随机存储器，但它可以以随机存取方式访问信息

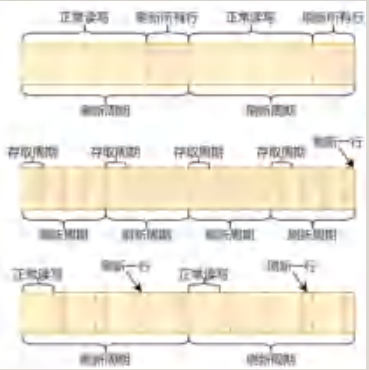




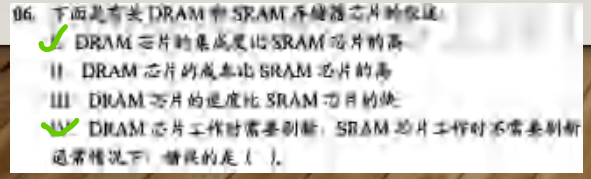


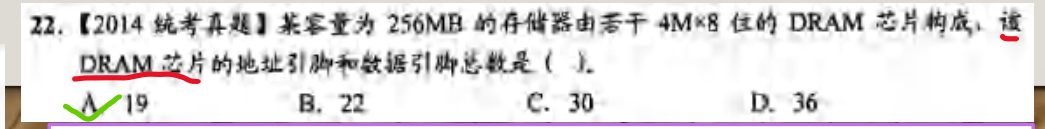
- **SRAM**(静态RAM)基于六MOS管触发器，不需刷新，速度快，功耗大，一般用于Cache。

- **DRAM**(动态RAM)基于电容，集成度高，成本低，容量大，速度慢，行地址和列地址分两次送入(地址引脚复用，一般**折半**处理)，引脚更少。需要定时刷新，将**一行**的原信息读出放大再重新写入，每个刷新周期内都要将所有行刷新一遍。有三种刷新方式：



  - **集中刷新**，每个刷新周期的末尾将所有行统一刷新一次，这会使存储器有一段时间无法读写操作。死时间
  - **分散刷新**，每个存取周期都要刷新一行，刷新周期可以更短，无死时间，但存取周期强制变长。
  - **异步刷新**，每个刷新周期划分为行数个时隙，每个时隙刷新一行，死时间和存取周期短。

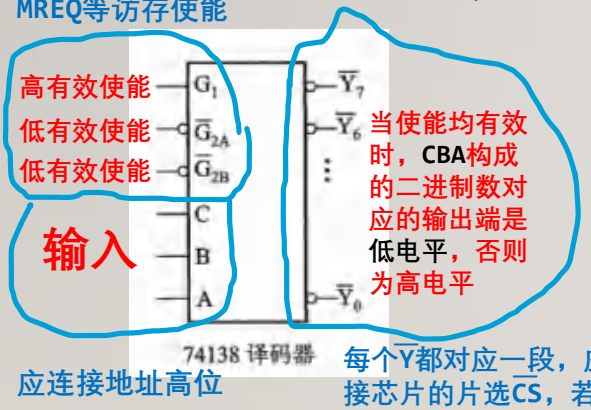




地址22位，对于DRAM折半为11位行列地址进行引脚复用，因此11+8=19

23.多存储芯片电路设计

- 关键：给定一个地址，这个地址对应的存储单元在哪些芯片里，哪些芯片可以工作(**片选信号$\overline{CS}$**)。**位扩展**需要让一些芯片同时工作并将数据线并起来，**字扩展**需要给地址空间**分段**，每段对应一些芯片，需要使用**译码器**通过给定地址的高位判断是哪一段，生成片选信号。



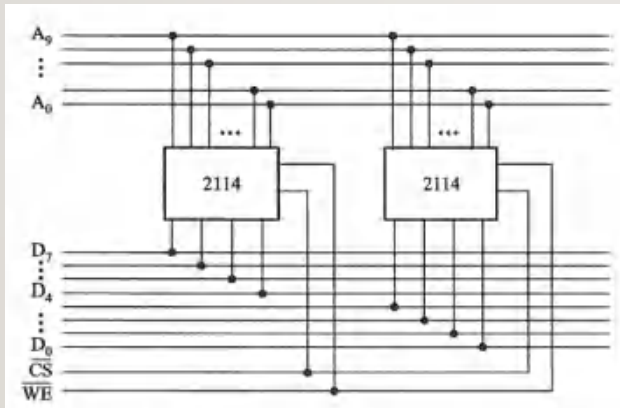


图 4.32 由 2 片 1 K×4 位的芯片组成 1 K×8 位的存储器

简单位扩展，逻辑上每个8位存储单元都对应两个芯片中同一个物理位置的4位存储单元，亦即拆成高低4位，两个芯片要同时工作，接受同样的地址和控制信号

相当于一个1-2译码器，通过A_{10}判断哪个CS低有效

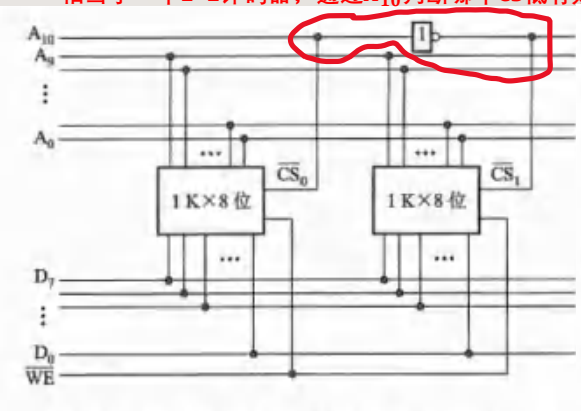
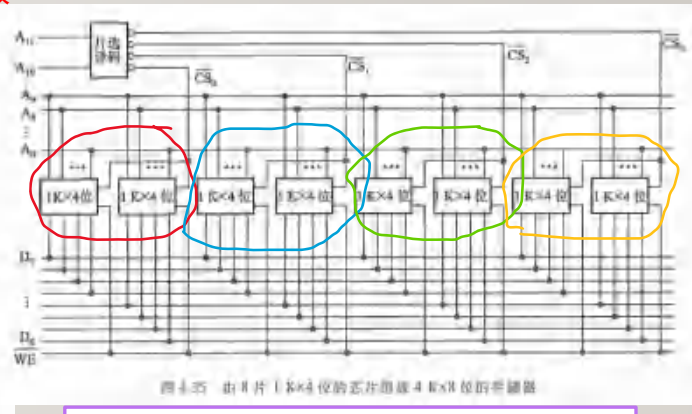


图 4.34 由 2 片 1 K×8 位的芯片组成 2 K×8 位的存储器

简单字扩展，将2K地址空间拆成两个1K(10位)的段，段内地址为地址低10位$A_9 \dots A_0$，地址最高位A_{10}决定这地址在哪个段，每段对应一个芯片

段0：0x000~0x3FF
段1：0x400~0x7FF



简单字位扩展，将4K地址空间拆成4个1K的段，通过地址高2位A_{11}, A_{10}决定这地址在哪个段，使用2-4译码器实现片选，每段进行位扩展，对应2个4位芯片，构成8位的逻辑存储单元

段0：0x000~0x3FF
段1：0x400~0x7FF
段2：0x800~0xBFF
段3：0xC00~0xFFF

23.多存储芯片电路设计

例 4.1　设 CPU 有 16 根地址线，8 根数据线，并用 MREQ 作为访存控制信号（低电平有效），用 WR 作为读/写控制信号（高电平为读，低电平为写）。现要用下列存储芯片：1 K×4 位 RAM，4 K×8 位 RAM，8 K×8 位 RAM，2 K×8 位 ROM，4 K×8 位 ROM，8 K×8 位 ROM 及 74138 译码器和若干门电路，如图 4.36 所示。画出 CPU 与存储器的连接图，要求如下：

① 主存地址空间分配：

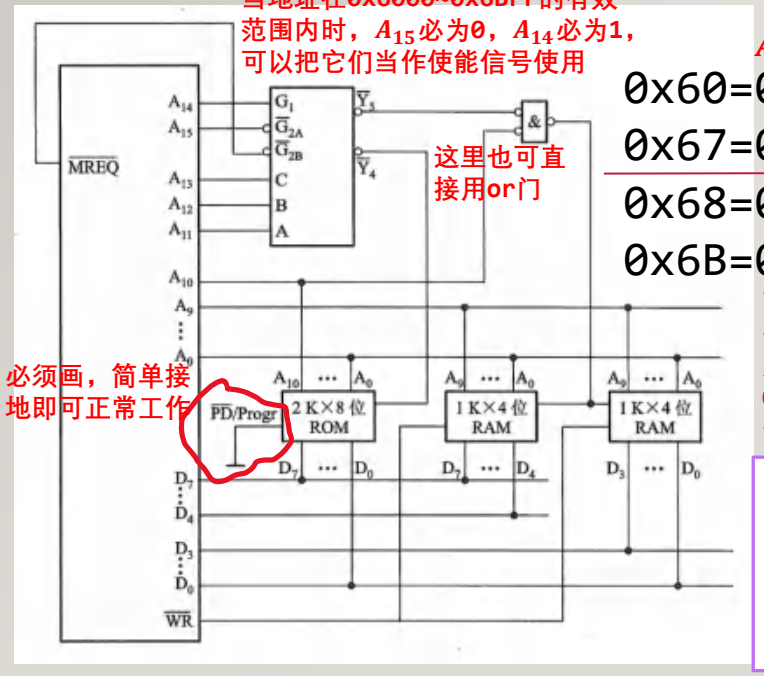
6000H～67FFH 为系统程序区。ROM

6800H～6BFFH 为用户程序区。RAM

② 合理选用上述存储芯片，说明各选几片。

③ 详细画出存储芯片的片选逻辑图。

当地址在0x6000~0x6BFF的有效范围内时，A_{15}必为0，A_{14}必为1，可以把它们当作使能信号使用



必须画，简单接地即可正常工作

$A_{11}=0$是选中ROM的充要条件

$A_{15}A_{14}A_{13}A_{12}A_{11}A_{10}A_{09}A_{08}$

0x60=0 1 1 0 0 0 0 0

0x67=0 1 1 0 0 1 1 1

0x68=0 1 1 0 1 0 0 0

0x6B=0 1 1 0 1 0 1 1

ROM段地址空间2K
段内地址11位
1片2K*8位ROM

RAM段地址空间1K
段内地址10位
2片1K*4位RAM(位扩展)

高4位事实上无需区分，但原则上它们为0110是芯片工作的必要条件

在高4位满足的情况下$A_{11}A_{10}=10$是选中RAM的充要条件

该电路的逻辑：

首先$MREQ, A_{14}, A_{15}$必须为0

当$A_{13}A_{12}A_{11}=100$时，ROM选中

当$A_{13}A_{12}A_{11}=101$且$A_{10}=0$时，RAM选中

若地址不落在这两个段里则任何芯片都不能工作!

例 4.3　设 CPU 有 20 根地址线和 16 根数据线，并用 IO/M 作为访存控制信号，RD 为读命令，WR 为写命令。CPU 可通过 BHE 和 A_0 来控制按字节或字两种形式访存（如表 4.1 所示）。要求采用图 4.39 所示的芯片，门电路自定。试回答：

（1）CPU 按字节访问和按字访问的地址范围各是多少？

（2）CPU 按字节访问时需分奇偶体，且最大 64 KB 为系统程序区，与其相邻的 64 KB 为用户程序区。写出每片存储芯片所对应的二进制地址码。

（3）画出对应上述地址范围的 CPU 与存储芯片的连接图。

段内地址为低16位，通过高4位区分段

ROM：1111

RAM：1110

表 4.1　例 4.3 CPU 访问形式与 BHE 和 A_0 的关系

BHE	A_0	访问形式
0	0	字
0	1	奇字节
1	0	偶字节
1	1	不访问

无论如何BHE=0是高8位数据有效的充要条件

0字访问必须让地址对齐

仅数据线高8位有效

仅数据线低8位有效

需要将奇偶地址字节分别存在两个芯片里，连接数据线高低8位，通过A_0, BHE确定让哪个芯片工作(或者同时工作)，芯片内地址为$A_{15}\dots A_1$

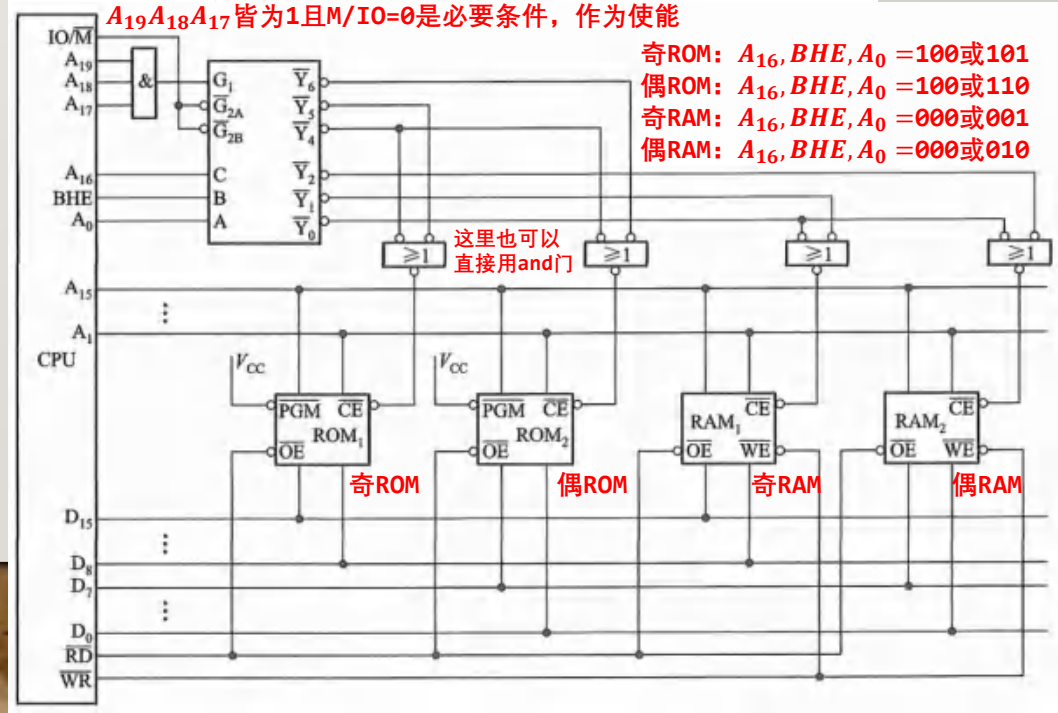
$A_{19}A_{18}A_{17}$皆为1且M/IO=0是必要条件，作为使能

奇ROM：A_{16}, BHE, A_0 =100或101

偶ROM：A_{16}, BHE, A_0 =100或110

奇RAM：A_{16}, BHE, A_0 =000或001

偶RAM：A_{16}, BHE, A_0 =000或010



24. 多体存储系统

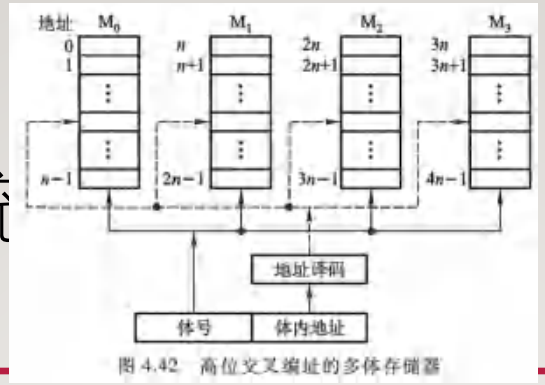


图 4.42 高位交叉编址的多体存储器

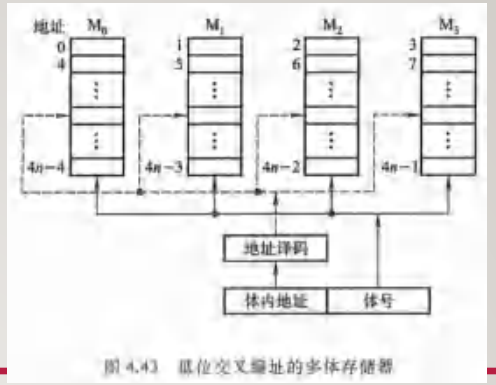


图 4.43 低位交叉编址的多体存储器

- **高位交叉方式**(顺序编址)：简单分段，每个存储体对应一段连续地址，通过地址高位确定要访问的存储体。适合多个设备同时并行使用主存的场景。

18. 下列说法中，正确的是（ ）
- ✗ 高位多体交叉存储器能很好地满足程序的局部性原理
- ✓ 高位四体交叉存储器可能在一个存储周期内连续访问 4 个模块

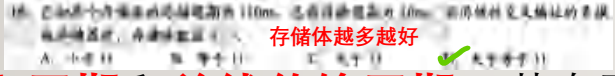
不适合快速访问连续地址

并行模块当然可以，尽管这样无法做到连续地址访问

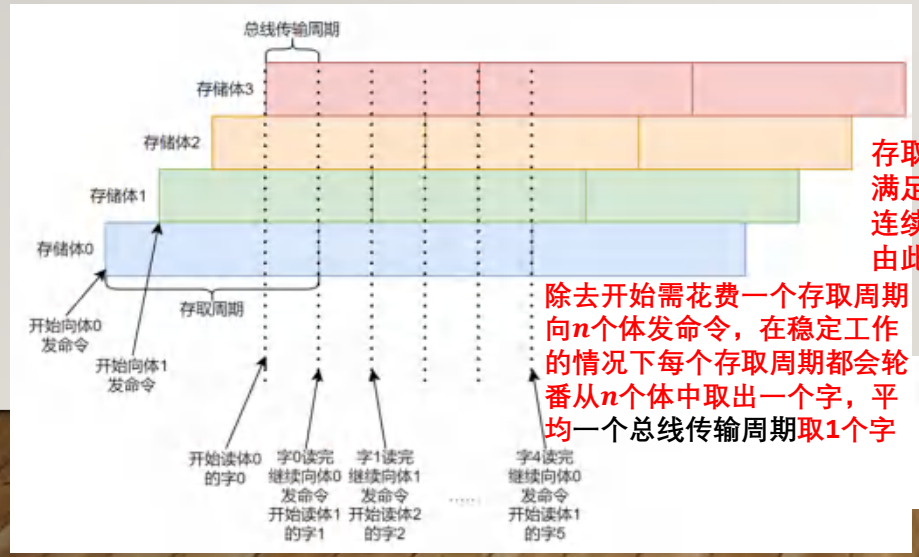
- **低位交叉方式**(交叉编址)：通过地址低位(**地址模n**)确定要访问的存储体，每个存储体仅对应所有地址低$\log n$位为固定值的存储单元。这样可以以流水线方式提高**连续地址**访问的带宽。

一般题目所说的“交叉”都是低位交叉

显然这样对局部性原理友好



考虑存储体的**存取周期**和**总线传输周期**，若存取周期$=n\times$总线传输周期，那么应采取n体低位交叉：



字0,1,2,3,4,5,6,7,8,9…
分别对应
体0,1,2,3,0,1,2,3,0,1…

存取周期为T,总线传输周期为t
满足$T=nt$
连续取连续存储的m个字的时间为$T+(m-1)t$，
由此可推出平均带宽

除去开始需花费一个存取周期向n个体发命令，在稳定工作的情况下每个存取周期都会轮番从n个体中取出一个字，平均一个总线传输周期取1个字

24. 【2015 统考真题】某计算机使用四体交叉编址存储器，假定在存储器总线上出现的主存地址（十进制）序列为 8005, 8006, 8007, 8008, 8001, 8002, 8003, 8004, 8000，则可能发生访存冲突的地址对是（ ）。考虑地址模4的值，对应哪个体

A. 8004 和 8008　B. 8002 和 8007　C. 8001 和 8008　D. 8000 和 8004

必须隔4个间隔才能再次访问体0

访存地址	8005	8006	8007	8008	8001	8002	8003	8004	8000
模块序号	1	2	3	0	1	2	3	0	0

25. 【2017 统考真题】某计算机主存按字节编址，由 4 个 64M×8 位的 DRAM 芯片采用交叉编址方式构成，并与宽度为 32 位的存储器总线相连，主存每次最多读写 32 位数据。若 double 型变量 x 的主存地址为 804 001 0AH，则读取 x 需要的存储周期数是（ ）。

A. 1　B. 2　C. 3　D. 4

double为8个连续存储的8位字，第一个存储周期向4个体发命令，接下来2个存储周期分别读4个字

17. 某机器采用四体低位交叉存储器，现分别执行下述操作：① 读取 6 个连续地址单元中存放的存储字，重复 80 次；② 读取 8 个连续地址单元中存放的存储字，重复 60 次。则①，②所花费的时间之比为（ ）。

$(8\times 80):(8\times 60)=4:3$

A. 1:1　B. 2:1　C. 4:3　D. 3:4

对于每轮6个连续地址访问你需要访问体0,1,2,3,0,1，然后空两个传输周期才能开始下一轮的0,1,2,3,0,1，因此每一轮要8个传输周期！而每轮8个连续地址访问只需要0,1,2,3,0,1,2,3,不需间隔，可以完全连续访问！

25.汉明校验

纠正1位错能力-即在仅出现至多一位错的假设下，不管哪位(包括加的校验位)出错都能纠正

- 对于n位01串，向其中添加k位使得能够纠正存储/传输时出现的1位错，必须满足$2^k \geq n+k+1$，汉明码就是恰好能满足这个下界的1位纠错码。

一共$n+k$位，可能在其中1位出错，或者没错，共$n+k+1$种情况，添加的k位需要能反映出任何一种情况

例：要传送0101，按配偶原则进行汉明校验：

①$n=4$，根据$2^k \geq n+k+1$，添加$k=3$个校验位。

②要传送的7位01串为$C_1C_2C_3C_4C_5C_6C_7$(下标从1开始)，令$C_3C_5C_6C_7=0101$(正常数据)，$C_1C_2C_4$为校验位(下标为2^i)，它们分别负责对二进制第i位为1的下标进行偶校验。

③3=011，5=101，6=110，7=111

若题目让你按配奇原则那么你要把校验位取反，使得集合内1的个数是奇数

对0位为1的{1,3,5,7}偶校验：$C_1=C_3\oplus C_5\oplus C_7=0$
对1位为1的{2,3,6,7}偶校验：$C_2=C_3\oplus C_6\oplus C_7=1$
对2位为1的{4,5,6,7}偶校验：$C_4=C_5\oplus C_6\oplus C_7=0$

传送串0100101

但第6位出错，变成0100111

④接收方校验这3个集合是否出错：

$$P_0=C_1\oplus C_3\oplus C_5\oplus C_7=0$$
$$P_1=C_2\oplus C_3\oplus C_6\oplus C_7=1$$
$$P_2=C_4\oplus C_5\oplus C_6\oplus C_7=1$$

这说明，0位为1的集合{1,3,5,7}有偶数个1，“认为”全部正确，1位为1的集合{2,3,6,7}有奇数个1，有错，2位为1的集合{4,5,6,7}有奇数个1，有错

⑤因此，出现错误的下标必然满足第0位为0，第1位为1，第2位为1，即$P_2P_1P_0=110=6$，将C_6比特翻转得到0100101，因此原始数据为0101

若为配奇原则那么1为正确0为错误，取反，出错下标为$\overline{P_2P_1P_0}$

汉明码这样设计的核心思想在于添加的k个校验位互不影响，这几个由下标二进制位确定的集合互相交叉，通过判断所有集合是否出错/无错，每一个下标都能被唯一地交叉出来！

26.Cache

19.【2014 统考真题】采用指令 Cache 与数据 Cache 分离的主要目的是（ ）。

A. 降低 Cache 的缺失损失　　B. 提高 Cache 的命中率

C. 降低 CPU 平均访存时间　　D. 减少指令流水线资源冲突 ✓

流水线中不同指令的取指和访存可能同时进行，指令数据Cache分立可以避免冲突

20.【2016 统考真题】有如下 C 语言程序段：

```
for(k=0; k<1000; k++)
    a[k] = a[k]*32;
```

若数组 a 和变量 k 均为 int 型，int 型数据占 4B，数据 Cache 采用直接映射方式，数据区大小为 1KB，块大小为 16B，该程序段执行前 Cache 为空，则该程序段执行过程中访问数组 a 的 Cache 缺失率约为（ ）。

A. 1.25%　　B. 2.5%　　C. 12.5% ✓　　D. 25%

共访存2000次，每轮循环的第二次一定不缺失，每块有4个int，在直接映射下连续访问地址，每隔4轮循环(每访存8次)就要调入新块，因此缺失率12.5%

02. 某计算机的主存地址位数为 32 位，按字节编址。假定数据 Cache 中最多存放 128 个主存块，采用四路组相联方式，块大小为 64B，每块设置了 1 位有效位。采用一次性回写策略，为此每块设置了 1 位"脏"位，要求：

1）分别指出主存地址中标记（Tag）、组号（Index）和块内地址（Offset）三部分的位置与位数。低6位块内地址(64B)，中间5位组地址(32组)，高21位标记

2）计算该数据 Cache 的总位数。$128 \times (64 \times 8 + 21 + 1 + 1) = 68480$

- **命中率**=Cache命中次数/总访存次数，**平均访问时间**=命中率*Cache存取周期+(1-命中率)*主存存取周期，**Cache效率**=Cache存取周期/平均访问时间。 一定要正确读题区分"命中率""不命中率""缺失率"

原则上，无论如何Cache是每次访存都要先访问的，无论命中与否，本书默认"主存存取周期"里包含了Cache的时间

"效率"一定是理想:实际

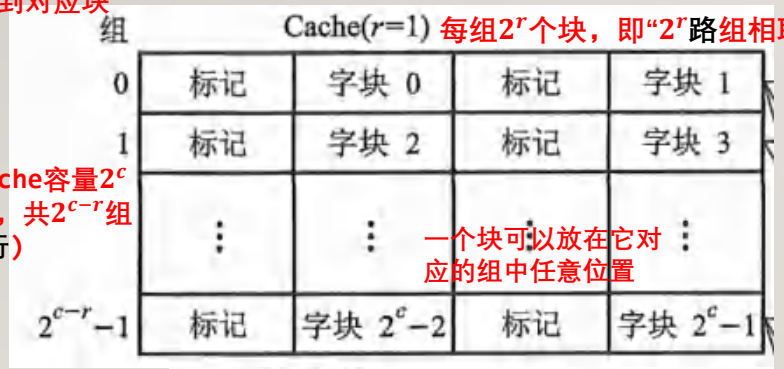
- 3种替换方式：先进先出FIFO(替换掉最早的块)，**最近最少使用LRU**，随机法。

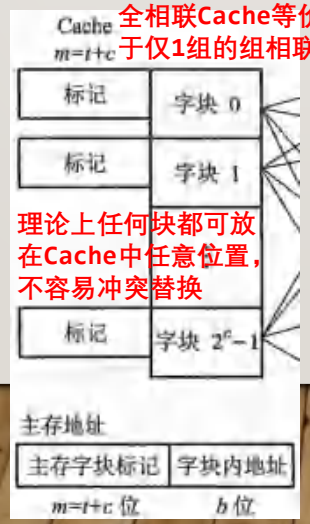
可记录最近一次使用的时间

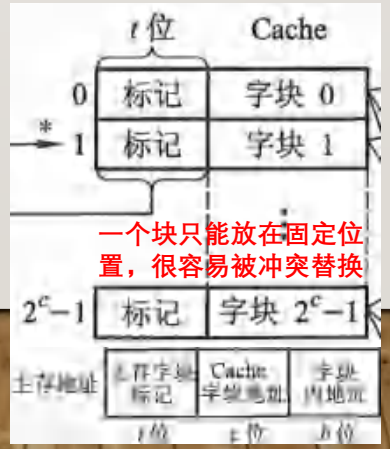
- 2种写操作：**写直达**(每次写操作既写Cache又写主存，慢但保证一致性)，**写回**(写操作仅写Cache，每块增加一个**修改位**，替换出Cache时将修改过的块写回主存，可能有一致性问题)。

给定一个主存地址，先通过组地址直接判断是哪组，再在组内通过对标记挨个比较找到对应块

- 3种Cache：**组相联**，**直接映射**，**全相联**。







05. 某32位计算机的Cache容量为16KB，Cache行的大小为16B，若主存与Cache地址映像采用直接映像方式，则主存地址为0x1234E8F8的单元装入Cache的地址是（ ）。

A. 0001000100101　　B. 0100010011010

C. 1010001111000 ✓　　D. 1101001101000

低4位为块内地址(16B)，0x8=1000

Cache内1K个块，中间10位Cache块地址，0x28F=1010001111

则Cache地址为10100011111000

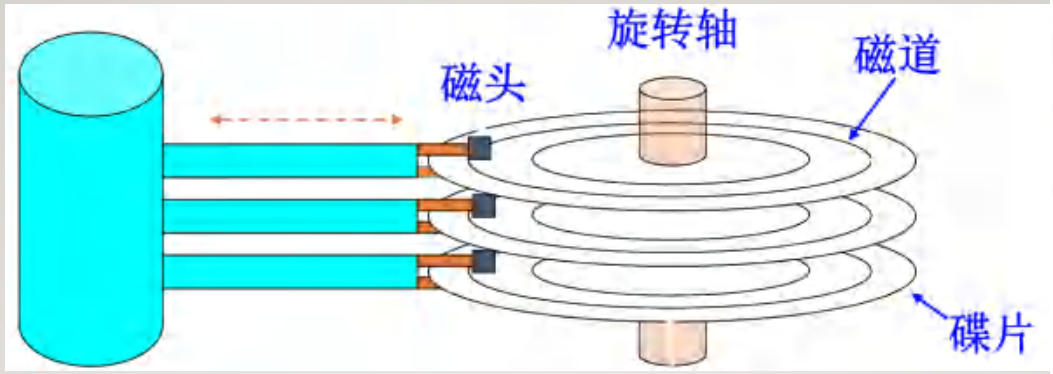
22.【2021 统考真题】若计算机主存地址为 32 位，按字节编址，Cache 数据区大小为 32KB，主存块大小为 32B，采用直接映射方式和回写（Write Back）策略，则 Cache 行的位数至少是（ ）。

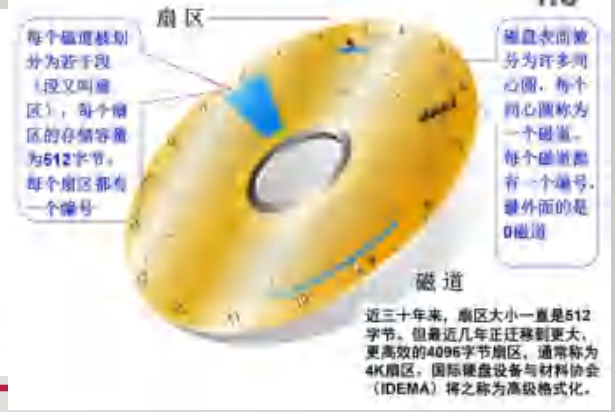
A. 275 ✓　　B. 274　　C. 258　　D. 257

块内地址5位(32B)，Cache内1K个块，Cache块地址10位，剩下32-10-5=17位作标记，需要记录在Cache行中。再加一个有效位和修改位(写回)，因此256+17+1+1=275

无论如何Cache块由有效数据、标记和有效位以及可能的修改位构成，但Cache容量一般只计算有效数据，但如果是总位数那就要算上附加的位

27.磁盘





- **道密度**反映半径上单位长度的磁道数(道/mm)，**位密度**反映磁道上单位长度的数据量(bits/mm)
- 磁盘数据访问过程：磁头先寻磁道，等待磁盘旋转到数据起始位置，再随着磁盘旋转进行连续数据传输。数据传输率=磁道数据量×1s内转圈数
- 磁盘平均存取时间=**平均寻道时间**+**平均旋转等待时间**+**数据传输时间**

0.5×旋转一周时间　　取决于转速和位密度(磁道数据量)

（反正就是简单小学数学计算……）

平均寻址时间

01. 一个磁盘的转速为 7200 转/分，每个磁道有 160 个扇区，每个扇区有 512 字节，则在理想情况下，其数据传输率为（　）。1s120转，传输$120\times160\times512=9600KB$数据

A. 7200×160KB/s　B. 7200KB/s　C. 9600KB/s　D. 19200KB/s

04. 【2013 统考真题】某磁盘的转速为 10000 转/分，平均寻道时间是 6ms，磁盘传输速率是 20MB/s，磁盘控制器延迟为 0.2ms，读取一个 4KB 的扇区所需的平均时间约为（　）。

A. 9ms　B. 9.4ms　C. 12ms　D. 12.4ms

平均旋转等待时间$=0.5\times\frac{60s}{10000rpm}=3ms$

数据传输时间$=\frac{4KB}{20MB/s}=0.19ms$

平均读取时间$=6ms+3ms+0.19ms+0.2ms=9.4ms$

01. 某个硬磁盘共有 4 个记录面，存储区域内半径为 10cm，外半径为 15.5cm，道密度为 60 道/cm，外层位密度为 600bit/cm，转速为 6000 转/分。

1）硬磁盘的磁道总数是多少？

2）硬磁盘的容量是多少？

3）假定每个扇区的容量为 512B，每个磁道有 12 个扇区。寻道的平均等待时间为 10.5ms，试计算磁盘平均存取时间。

1）有效存储区域 = 15.5 − 10 = 5.5cm，道密度 = 60 道/cm，因此每个面为 60×5.5 = 330 道，即有 330 个柱面，因此磁道总数 = 4×330 = 1320 个磁道。

2）外层磁道的长度为 $2\pi R=2\times3.14\times15.5=97.34cm$。

每道信息量 = 600bit/cm×97.34cm = 58404bit = 7300B。

利用 1）的结果，可得磁盘总容量 = 7300B×1320 = 9636000B（非格式化容量）。

3）读一个扇区中数据所用的时间 = 找磁道的时间 + 找扇区的时间 + 磁头扫过一个扇区的时间。

找磁道的时间是指磁头从当前所处磁道运动到目标磁道的时间，一般选用磁头在磁盘径向方向上移动 1/2 个半径长度所用的时间为平均值来估算，题中给出的是 10.5ms。

找扇区的时间是指磁头从当前所处扇区运动到目标扇区的时间，一般选用磁盘旋转半周所用的时间作为平均值来估算，题中给出磁盘转速为 6000 转/分，即 100 转/秒，所以磁盘转一周用时 10ms，转半周用时 5ms。

题中给出每个磁道有 12 个扇区，磁头扫过一个扇区用时为 10/12 = 0.83ms，因此磁盘平均存取时间为 10.5 + 5 + 0.83 = 16.33ms。

28. 总线

27. 【2015 统考真题】下列有关总线定时的概述中，错误的是（ ）。
A. 异步通信方式中，全互锁协议最慢
B. 异步通信方式中，非互锁协议的可靠性最差
C. 同步通信方式中，同步时钟信号可由各设备提供
D. 半同步通信方式中，握手信号的采样由同步时钟控制

时钟必须是统一提供的！不然难以同步

请求信号 回答信号 撤销请求 撤销回答

09. 在手术过程中，医生将手伸出，等护士将手术刀递上，待医生握紧后，护士才松手。若把医生和护士视为两个通信模块，上述动作相当于（ ）。
A. 同步通信 B. 异步通信的全互锁方式
C. 异步通信的半互锁方式 D. 异步通信的不互锁方式

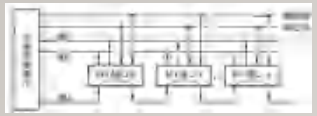
19. 【2009 统考真题】假设某系统总线在一个总线周期中并行传输 4 字节信息，一个总线周期占用 2 个时钟周期，总线时钟频率为 10MHz，则总线带宽是（ ）。
A. 10MB/s B. 20MB/s C. 40MB/s D. 80MB/s

$4B \times \frac{10MHz}{2} = 20MB/s$

总线带宽一般用字节每秒

20. 【2010 统考真题】下列选项中的英文缩写均为总线标准的是（ ）。
A. PCI、CRT、USB、EISA B. ISA、CPI、VESA、EISA
C. ISA、SCSI、RAM、MIPS D. ISA、EISA、PCI、PCI-Express

- 常见的总线标准：ISA,EISA,VESA,PCI,PCI-E,AGP,USB,RS-232C



- 总线判优(仲裁)控制：主设备控制总线，需要决定哪个提出请求的从设备使用总线。

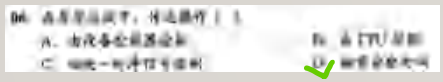
  - **链式查询**：总线同意信号BG一个设备一个设备往下传递，直到遇见一个提出请求的从设备为止。
  - **计数器定时查询**：增加设备地址线，主设备轮流询问设备号，直到询问到提出请求从设备为止。
  - **独立请求**：每个从设备与主设备单独连接请求/响应信号线，主设备内置判优电路。
- **同步通信**：双方通过统一时钟信号进行数据传输，要求速度一致，一个传输周期有多个时钟周期。

一个传输周期并行传输总线宽度位数的数据
传输周期的倒数是总线的工作频率 —这和时钟频率不同

- **异步通信**：基于“请求-等待-回答”模式，无统一时钟同步。

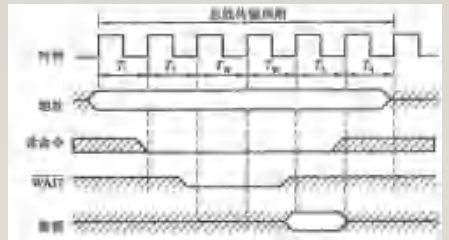
需要握手信号

  - **不互锁**(发起请求后无需回答信号，等待固定时间)，**半互锁**(有回答信号，收到回答信号后撤销请求)，**全互锁**(撤销请求后回答信号才撤销)。



在串行传输时需要有一些附加位

  - 异步串行通信：**波特率**(单根信号线上每秒传多少比特)，**比特率**(每秒传多少比特**有效**数据)。
- **半同步通信**：有统一时钟信号，但可通过等待信号WAIT插入若干等待周期T_w。



- **分离式通信**：设备在准备数据时不占用总线，利用效率高。

02. 某机器 I/O 设备采用异步串行传送方式传送字符信息，字符信息格式为 1 位起始位，7 位数据位，1 位校验位和 1 位停止位，若要求每秒传送 480 个字符，则该设备的数据传输率为（ ）。
A. 380b/s B. 4800B/s C. 480B/s D. 4800b/s

波特率，每个字符包括附加位共10bits

29.I/O系统

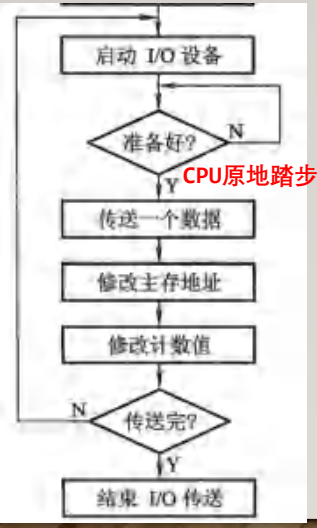
01. 在统一编址的方式下，区分存储单元和I/O设备是靠（ ）。
A. 不同的地址码　　B. 不同的地址线
C. 不同的控制线　　D. 不同的数据线

10.【2014统考真题】下列有关I/O接口的叙述中，错误的是（ ）。
A. 状态端口和控制端口可以合用同一个寄存器 例如触发器D和B
B. I/O接口中CPU可访问的寄存器称为I/O端口
C. 采用独立编址方式时，I/O端口地址和主存地址可能相同
D. 采用统一编址方式时，CPU不能用访存指令访问I/O端口

- I/O设备需要编址，可以采取统一编址(映射到存储器地址空间,直接使用访存指令实现I/O)或者不统一编址(和存储器地址独立,使用专用I/O指令)。
- **I/O端口**是接口电路中CPU可以**直接读写**的寄存器，多个I/O端口加上逻辑电路构成**I/O接口**，接口是主机与外设之间的界面，进行数据/控制命令/状态的信息交换，实现数据缓冲(解决数据传送时速度不匹配问题)，使用设备选择电路(SEL)进行地址译码，转换信息格式(串/并、数/模等)等功能。

11.【2017统考真题】I/O指令实现的数据传送通常发生在（ ）。
A. I/O设备和I/O端口之间　　B. 通用寄存器和I/O设备之间
C. I/O端口和I/O端口之间　　D. 通用寄存器和I/O端口之间

02. 下列功能中，属于I/O接口的功能的是（ ）。
I. 数据格式的转换　　II. I/O过程中错误与状态检测
III. I/O操作的控制与定时　　IV. 与主机和外设通信



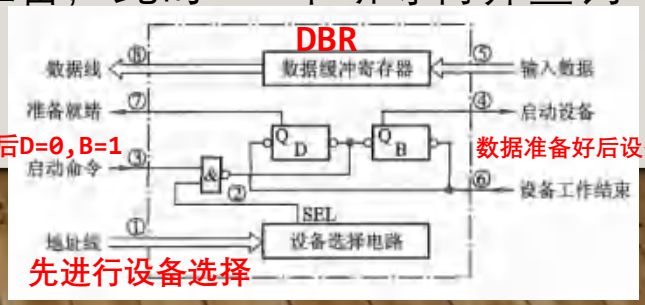
CPU原地踏步等待

- I/O接口中的寄存器：数据缓冲寄存器DBR，完成触发器D，工作触发器B，中断请求触发器INTR，中断屏蔽触发器MASK。
- CPU与I/O设备的数据传送可以使用程序查询方式、程序中断方式、DMA方式。

在本课程一般认为设备每次进行一个字的传送

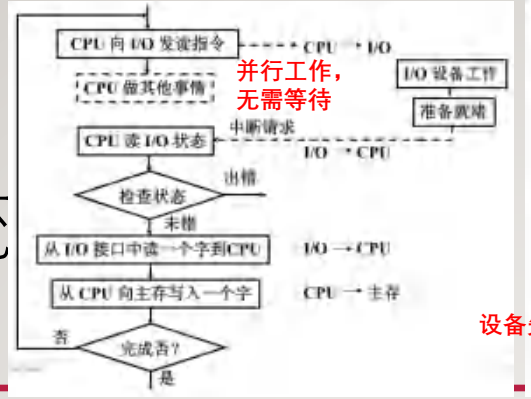
- **程序查询方式**：CPU向I/O设备发命令，I/O设备进行数据准备，此时CPU不断等待并查询I/O设备状态(触发器D)，直到I/O设备准备好数据为止。

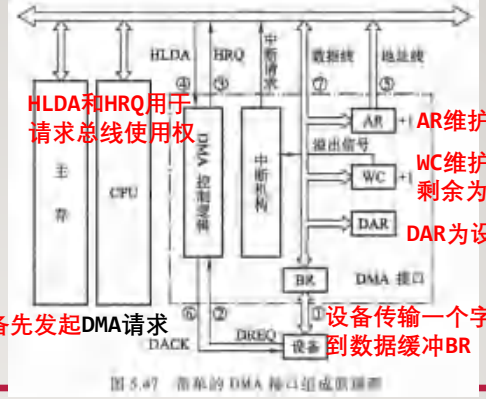
这样的话CPU与I/O设备串行工作，CPU不能干其它事，效率低

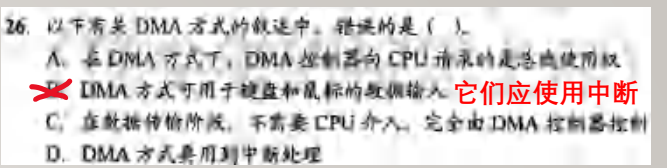


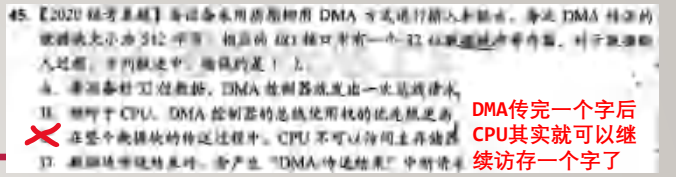
CPU发出启动命令后D=0,B=1
数据准备好后设备令D=1,B=0
先进行设备选择

29.I/O系统









更适合处理设备异常事件

- **程序中断方式**：CPU向I/O设备发命令，I/O设备进行数据准备，此时CPU**执行其它工作**。当数据准备完成时，设备向CPU发出中断请求，CPU响应中断并执行中断处理程序，从I/O接口处读写一个字。这样可以尽量使CPU与I/O设备并行工作，但数据的传送仍需CPU主动完成。

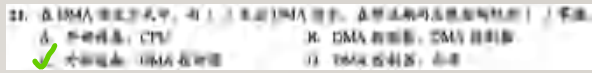
更适合大批量数据的快速连续传输

- **DMA**(直接内存访问)方式：在I/O接口设置DMA控制器，和主存有总线连接。DMA采取**批量**数据读写的方式，CPU需要先设置DMA控制器(一批有多少字、主存起始地址等)，然后CPU执行其它工作，DMA控制器进行I/O设备与主存之间的数据传输。每传输一个字都要取得总线控制权，当一批数据传输完成后**发起中断**告知CPU。DMA有3种总线占用方式：

无论如何必有DMA接口

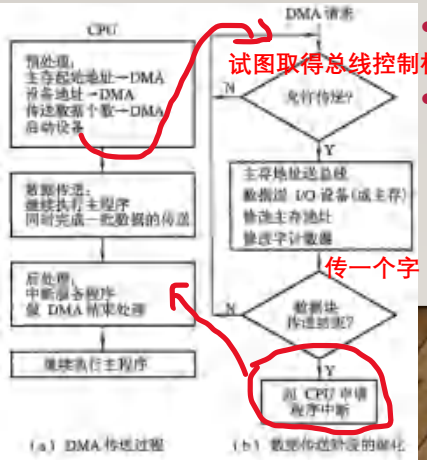
对于外设来说，DMA请求优先级比中断请求更高，因为DMA一般用于高速设备

  - 直接令CPU停止访问主存，在传输这批数据期间DMA控制器始终占用总线。效率较低。
  - 将CPU工作周期划分为CPU访存周期和DMA周期，DMA可以在DMA周期时占用总线。

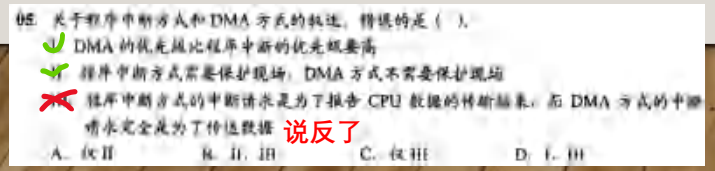


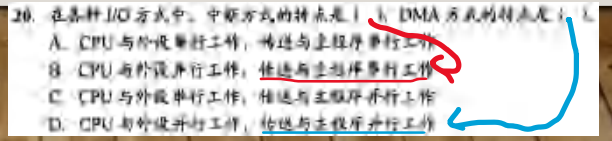
注意这个不是DMA请求

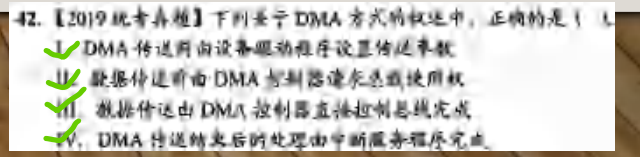
  - **周期挪用(窃取)**，对于DMA要传输的每个字(花费一个存取周期)，需要提出总线占用请求，若CPU未访问主存则直接占用总线，若CPU正在占用总线则等待CPU当前存取周期结束，若与CPU同时发起请求，则使DMA**优先占用总线**。总而言之这样最多会使得CPU**延缓一个存取周期**，效率较高。



而不是中断那样传一个字就要执行一次中断处理程序



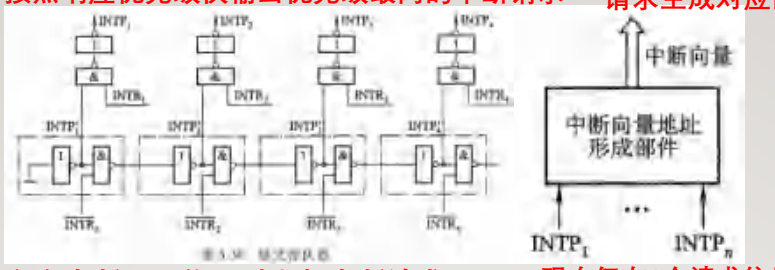




30.中断机制

按照响应优先级仅输出优先级最高的中断请求

编码器根据这唯一的中断请求生成对应的向量地址



多个中断源可能同时发起中断请求INTR

现在仅有1个请求信号有效

31. 【2009 统考真题】下列选项中，能引起外部中断的事件是（ ）。
A. 键盘输入　B. 除数为 0　C. 浮点运算下溢　D. 访存缺页

- 中断源：自愿中断(INT指令,实现系统调用)，程序性事故(溢出等)，硬件故障，外部I/O设备(数据传输或引发事件)。可分为**可屏蔽中断**和**不可屏蔽中断**。

28. 中断发生时，程序计数器内容的保护和更新是由（ ）完成的。
A. 硬件自动　B. 遗嘱指令和转移指令
C. 转移指令　D. 中断服务程序

- CPU在执行完一条指令后，发出中断查询信号，若有中断请求则响应中断，进入中断周期，执行“**中断隐指令**”：保存当前PC，将**向量地址**送PC，**关中断**(EINT=0)。本课程认为向量地址就是**中断服务程序**的入口地址。

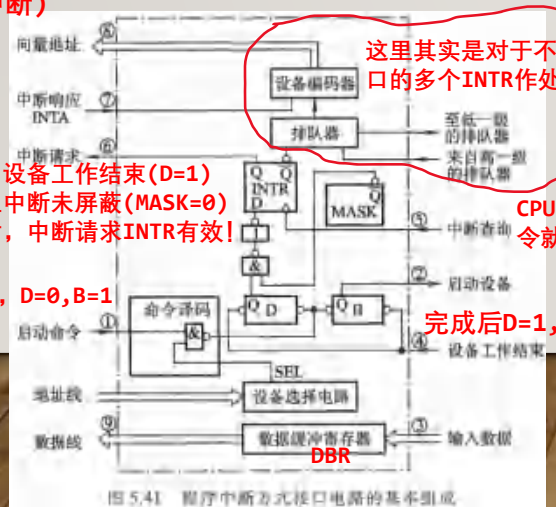
并不是真正的指令，只是CPU自动执行的固定步骤

不同类型中断有不同向量地址，对应不同处理程序

10. 中断响应是在（ ）。
A. 一条指令执行开始　B. 一条指令执行中间
C. 一条指令执行之末　D. 一条指令执行的任何时刻

- 对于I/O设备，不同接口对应不同的中断请求信号INTR，需要**中断排队器**按照固定优先级决定CPU响应同时发生的多个不同INTR中的哪个，再交由**编码器**生成对应的向量地址。

程序中断方式：
(在数据准备完毕后发起中断)



这里其实是对于不同接口的多个INTR作处理

当设备工作结束(D=1)且中断未屏蔽(MASK=0)时，中断请求INTR有效!

CPU每执行完一条指令就发出中断查询

CPU发起启动命令，D=0,B=1

完成后D=1,B=0，通过D发出INTR

01. 设置中断排队判优逻辑的目的是（ ）。
A. 产生中断源编码
B. 使同时提出的请求中的优先级别最高者得到及时响应
C. 使 CPU 能方便地转入中断服务子程序
D. 提高中断响应速度

35. 【2012 统考真题】响应外部中断的过程中，中断隐指令完成的操作，除保护断点外，还包括（ ）。
I. 关中断
II. 保存通用寄存器的内容　保护现场是中断服务程序做的
III. 形成中断服务程序入口地址并送 PC
A. 仅 I、II　B. 仅 I、III　C. 仅 II、III　D. I、II、III

30.中断机制

32.【2010 统考真题】单级中断系统中，中断服务程序内的执行顺序是（ ）。
I. 保护现场 II. 开中断 III. 关中断 IV. 保存断点 V. 中断事件处理 VI. 恢复现场 VII. 中断返回
A. I→V→VI→II→VII　B. III→I→V→VII
C. III→IV→V→VI→VII　D. IV→I→V→VI→VII

禁止嵌套，最后才打开中断

46.【2021 统考真题】下列是关于多重中断系统中 CPU 响应中断的叙述，错误的是（ ）。
A. 仅在用户态（执行用户程序）下，CPU 才能检测和响应中断　这样显然实现不了多重中断吧
B. CPU 只有在检测到中断请求信号后，才会进入中断响应周期
C. 进入中断响应周期时，CPU 一定处于中断允许（开中断）状态
D. 若 CPU 检测到中断请求信号，则一定存在未被屏蔽的中断请求信号

39.【2017 统考真题】下列关于多重中断系统的叙述中，错误的是（ ）。
A. 在一条指令执行结束时响应中断
B. 中断处理期间 CPU 处于关中断状态
C. 中断请求的产生与当前指令的执行无关
D. CPU 通过采样中断请求信号检测中断请求

- 中断服务程序：①**保护现场**，②中断服务，③**恢复现场**，④中断返回。

将通用寄存器/状态寄存器压栈

- **单重中断**不允许中断嵌套，在最后中断返回前重新打开中断(EINT=1)。**多重中断**允许具有**优先级**的中断嵌套，在一开始保护现场后就开中断，且需要中断屏蔽机制。

- 中断优先级分为响应优先级和处理优先级，**响应优先级**由硬件排队电路设置(固定不可变)，决定CPU可以响应同时发来的多个中断请求中的哪个。**处理优先级**由中断屏蔽字设置(可以软件方式改变)，在多重中断下决定CPU在处理中断A的过程中是否可以暂时搁置A去**嵌套**响应突然发来的中断B(因为B有比A更高的处理优先级)。

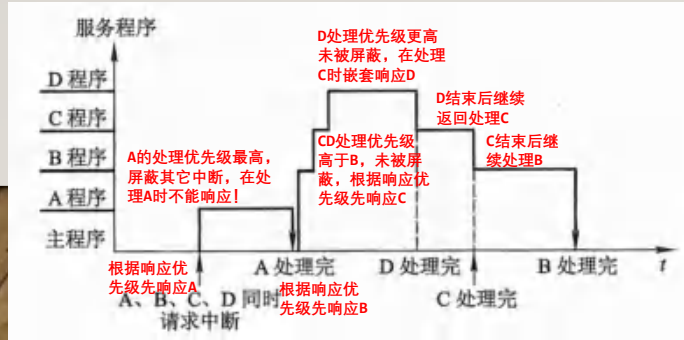
这两种优先级可以相同亦可不同，正确看待它们!

这一行为决定了处理优先级

- 在多重中断下每个中断的服务程序一开始会设置中断屏蔽字(每一位对应每个I/O接口的中断屏蔽触发器MASK)，把处理优先级**更低**的所有中断(包括自己)屏蔽掉。

响应优先级：A>B>C>D
处理优先级：A>D>C>B

中断源	新屏蔽字
A	1 1 1 1
B	0 1 0 0
C	0 1 1 0
D	0 1 1 1



44.【2020 统考真题】外部中断包括不可屏蔽中断（NMI）和可屏蔽中断。下列关于外部中断的叙述中，错误的是（ ）。
A. CPU 处于关中断状态时，也能响应 NMI 请求
B. 一旦可屏蔽中断请求信号有效，CPU 将立即响应
C. 不可屏蔽中断的优先级比可屏蔽中断的优先级高
D. 可通过中断屏蔽字设置可屏蔽中断的处理优先级

13. 某计算机有 4 级中断，优先级从高到低为 1→2→3→4。若将优先级顺序修改，修改后 1 级中断的屏蔽字为 1101，2 级中断的屏蔽字为 0100，3 级中断的屏蔽字为 1111，4 级中断的屏蔽字为 0101，则修改后的优先顺序从高到低为（ ）。
A. 1→2→3→4　B. 3→1→4→2　C. 1→3→4→2　D. 2→1→3→4

33.【2011 统考真题】某计算机有五级中断 $L_4 \sim L_0$，中断屏蔽字为 $M_4M_3M_2M_1M_0$，$M_i = 1$ $(0 \le i \le 4)$ 表示对 L_i 级中断进行屏蔽。若中断优先级从高到低的顺序是 $L_4 \to L_0 \to L_2 \to L_1 \to L_3$，则 L_1 的中断处理程序中设置的中断屏蔽字是（ ）。
A. 11110　B. 01101　C. 00011　D. 01010

*附录：你或许应该知道的一些“名词”缩写

(这些仅仅是为了让你更好地理解它们)

- **ALU**，Arithmetic Logic Unit，算术逻辑单元
- **CU**，Control Unit，控制单元
- **ACC**，Accumulation Register，累加寄存器
- **MQ**，Multiplication and Quotient Register，乘商寄存器
- **PC**，Program Counter，程序计数器
- **IR**，Instruction Register，指令寄存器
- **MAR**，Memory Address Register，存储器地址寄存器
- **MDR**，Memory Data Register，存储器数据寄存器
- **MIPS**，Million Instructions Per Second，百万条指令每秒
- **CPI**，Clock Per Instruction，周期数每条指令
- **RAM**，Random Access Memory，随机存储器
- **ROM**，Read Only Memory，只读存储器
- **PROM**，Programmable ROM，可编程ROM
- **EPROM**，Erasable PROM，可擦除可编程ROM
- **EEPROM**，Electrically EPROM，电可擦除可编程ROM
- **SRAM**，Static RAM，静态RAM
- **DRAM**，Dynamic RAM，动态RAM
- **SDRAM**，Synchronous DRAM，同步动态RAM
- **CS**，Chip Select，片选信号
- **WE**，Write Enable，写使能信号
- **MREQ**，Memory Request，访存请求信号
- **LRU**，Least Recently Used，最近最少使用
- **rpm**，round per minute，(磁盘)转数每分钟
- **DMA**，Direct Memory Access，直接内存访问
- **D**，Done，设备完成状态触发器
- **B**，Busy，设备忙状态触发器
- **INTR**，Interrupt Request，中断请求
- **EINT**，Enable of Interrupt，中断允许
- **DBR**，Data Buffer Register，数据缓冲寄存器
- **EA**，Effective Address，有效地址
- **BR**，Base Register，基址寄存器
- **IX**，Index Register，变址寄存器

*附录：你或许应该知道的一些“名词”缩写

- **SP**，**S**tack **P**oint，栈指针寄存器
- **CISC**，**C**omplex **I**nstruction **S**et **C**omputer，复杂指令系统计算机
- **RISC**，**R**educed **I**nstruction **S**et **C**omputer，精简指令系统计算机
- **RAW**，**R**ead **A**fter **W**rite，(流水线)读后写相关
- **OP**，**Op**eration (Code)，指令操作码
- **Ad**，**Ad**dress (Code)，指令地址码
- **CMAR**，**C**ontrol **M**emory **A**ddress **R**egister，控存地址寄存器
- **CMDR**，**C**ontrol **M**emory **D**ata **R**egister，控存数据寄存器
- **LDA**，**L**oad to **A**CC，取数到ACC指令
- **STA**，**St**ore from **A**CC，将ACC存到主存指令

EOF