第一章 计算机系统概论

1. 什么是计算机系统、计算机硬件和计算机软件？硬件和软件哪个更重要？

解：P3

计算机系统：由计算机硬件系统和软件系统组成的综合体。

计算机硬件：指计算机中的电子线路和物理装置。

计算机软件：计算机运行所需的程序及相关资料。

硬件和软件在计算机系统中相互依存，缺一不可，因此同样重要。

5. 冯•诺依曼计算机的特点是什么？

解：冯•诺依曼计算机的特点是：P8

- 计算机由运算器、控制器、存储器、输入设备、输出设备五大部件组成；
- 指令和数据以同同等地位存放于存储器内，并可以按地址访问；
- 指令和数据均用二进制表示；
- 指令由操作码、地址码两大部分组成，操作码用来表示操作的性质，地址码用来表示操作数在存储器中的位置；
- 指令在存储器中顺序存放，通常自动顺序取出执行；
- 机器以运算器为中心（原始冯•诺依曼机）。

7. 解释下列概念：

主机、CPU、主存、存储单元、存储元件、存储基元、存储元、存储字、存储字长、存储容量、机器字长、指令字长。

解：P9-10

主机：是计算机硬件的主体部分，由CPU和主存储器MM合成为主机。

CPU：中央处理器，是计算机硬件的核心部件，由运算器和控制器组成；（早期的运算器和控制器不在同一芯片上，现在的CPU内除含有运算器和控制器外还集成了CACHE）。

主存：计算机中存放正在运行的程序和数据的存储器，为计算机的主要工作存储器，可随机存取；由存储体、各种逻辑部件及控制电路组成。

存储单元：可存放一个机器字并具有特定存储地址的存储单位。

存储元件：存储一位二进制信息的物理元件，是存储器中最小的存储单位，又叫存储基元或存储元，不能单独存取。

存储字：一个存储单元所存二进制代码的逻辑单位。

存储字长：一个存储单元所存二进制代码的位数。

存储容量：存储器中可存二进制代码的总量；（通常主、辅存容量分开描述）。

机器字长：指CPU一次能处理的二进制数据的位数，通常与CPU的寄存器位数有关。

指令字长：一条指令的二进制代码位数。

8. 解释下列英文缩写的中文含义：

CPU、PC、IR、CU、ALU、ACC、MQ、X、MAR、MDR、I/O、MIPS、CPI、FLOPS

解：全面的回答应分英文全称、中文名、功能三部分。

CPU：Central Processing Unit，中央处理机（器），是计算机硬件的核心部件，主要由运算器和控制器组成。

PC：Program Counter，程序计数器，其功能是存放当前欲执行指令的地址，并可自动计数

形成下一条指令地址。

IR：Instruction Register，指令寄存器，其功能是存放当前正在执行的指令。

CU：Control Unit，控制单元（部件），为控制器的核心部件，其功能是产生微操作命令序列。

ALU：Arithmetic Logic Unit，算术逻辑运算单元，为运算器的核心部件，其功能是进行算术、逻辑运算。

ACC：Accumulator，累加器，是运算器中既能存放运算前的操作数，又能存放运算结果的寄存器。

MQ：Multiplier-Quotient Register，乘商寄存器，乘法运算时存放乘数、除法时存放商的寄存器。

X：此字母没有专指的缩写含义，可以用作任一部件名，在此表示操作数寄存器，即运算器中工作寄存器之一，用来存放操作数；

MAR：Memory Address Register，存储器地址寄存器，在主存中用来存放欲访问的存储单元的地址。

MDR：Memory Data Register，存储器数据缓冲寄存器，在主存中用来存放从某单元读出、或要写入某存储单元的数据。

I/O：Input/Output equipment，输入/输出设备，为输入设备和输出设备的总称，用于计算机内部和外界信息的转换与传送。

MIPS：Million Instruction Per Second，每秒执行百万条指令数，为计算机运算速度指标的一种计量单位。

9. 画出主机框图，分别以存数指令“STA M”和加法指令“ADD M”（M 均为主存地址）为例，在图中按序标出完成该指令（包括取指令阶段）的信息流程（如→①）。假设主存容量为 256M*32 位，在指令字长、存储字长、机器字长相等的条件下，指出图中各寄存器的位数。

解：主机框图如 P13 图 1.11 所示。

（1）STA M 指令：PC→MAR，MAR→MM，MM→MDR，MDR→IR，

OP(IR) →CU，Ad(IR) →MAR，ACC→MDR，MAR→MM，WR

（2）ADD M 指令：PC→MAR，MAR→MM，MM→MDR，MDR→IR，

OP(IR) →CU，Ad(IR) →MAR，RD，MM→MDR，MDR→X，ADD，ALU→ACC，ACC→MDR，WR

假设主存容量 256M*32 位，在指令字长、存储字长、机器字长相等的条件下，ACC、X、IR、MDR 寄存器均为 32 位，PC 和 MAR 寄存器均为 28 位。

10. 指令和数据都存于存储器中，计算机如何区分它们？

解：计算机区分指令和数据有以下 2 种方法：

- 通过不同的时间段来区分指令和数据，即在取指令阶段（或取指微程序）取出的为指令，在执行指令阶段（或相应微程序）取出的即为数据。
- 通过地址来源区分，由 PC 提供存储单元地址的取出的是指令，由指令地址码部分提供存储单元地址的取出的是操作数。

第 2 章 计算机的发展及应用

1. 通常计算机的更新换代以什么为依据？

答：P22

主要以组成计算机基本电路的元器件为依据，如电子管、晶体管、集成电路等。

2. 举例说明专用计算机和通用计算机的区别。

答：按照计算机的效率、速度、价格和运行的经济性和实用性可以将计算机划分为通用计算机和专用计算机。通用计算机适应性强，但牺牲了效率、速度和经济性，而专用计算机是最有效、最经济和最快的计算机，但适应性很差。例如个人电脑和计算器。

第 3 章 系统总线

1. 什么是总线？总线传输有何特点？为了减轻总线负载，总线上的部件应具备什么特点？

答：P41.总线是多个部件共享的传输部件。

总线传输的特点是：某一时刻只能有一路信息在总线上传输，即分时使用。

为了减轻总线负载，总线上的部件应通过三态驱动缓冲电路与总线连通。

4. 为什么要设置总线判优控制？常见的集中式总线控制有几种？各有何特点？哪种方式响应时间最快？哪种方式对电路故障最敏感？

答：总线判优控制解决多个部件同时申请总线时的使用权分配问题；

常见的集中式总线控制有三种：链式查询、计数器定时查询、独立请求；

特点：链式查询方式连线简单，易于扩充，对电路故障最敏感；计数器定时查询方式优先级设置较灵活，对故障不敏感，连线及控制过程较复杂；独立请求方式速度最快，但硬件器件用量大，连线多，成本较高。

5. 解释下列概念：总线宽度、总线带宽、总线复用、总线的主设备（或主模块）、总线的从设备（或从模块）、总线的传输周期和总线的通信控制。

答：P46。

总线宽度：通常指数据总线的根数；

总线带宽：总线的数据传输率，指单位时间内总线上传输数据的位数；

总线复用：指同一条信号线可以分时传输不同的信号。

总线的主设备（主模块）：指一次总线传输期间，拥有总线控制权的设备（模块）；

总线的从设备（从模块）：指一次总线传输期间，配合主设备完成数据传输的设备（模块），它只能被动接受主设备发来的命令；

总线的传输周期：指总线完成一次完整而可靠的传输所需时间；

总线的通信控制：指总线传送过程中双方的时间配合方式。

6. 试比较同步通信和异步通信。

答：同步通信：指由统一时钟控制的通信，控制方式简单，灵活性差，当系统中各部件工作速度差异较大时，总线工作效率明显下降。适合于速度差别不大的场合。

异步通信：指没有统一时钟控制的通信，部件间采用应答方式进行联系，控制方式较同步复杂，灵活性高，当系统中各部件工作速度差异较大时，有利于提高总线工作效率。

8. 为什么说半同步通信同时保留了同步通信和异步通信的特点？

答:半同步通信既能像同步通信那样由统一时钟控制,又能像异步通信那样允许传输时间不一致,因此工作效率介于两者之间。

10. 为什么要设置总线标准？你知道目前流行的总线标准有哪些？什么叫 plug and play？哪些总线有这一特点？

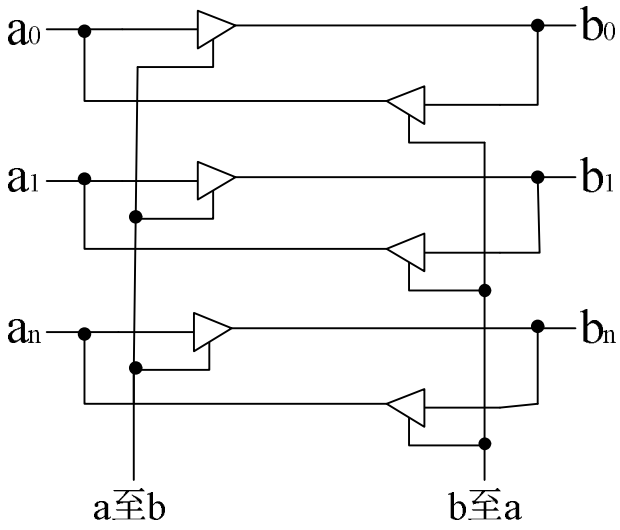
答：总线标准的设置主要解决不同厂家各类模块化产品的兼容问题；

目前流行的总线标准有：ISA、EISA、PCI 等；

plug and play：即插即用，EISA、PCI 等具有此功能。

11. 画一个具有双向传输功能的总线逻辑图。

答：在总线的两端分别配置三态门，就可以使总线具有双向传输功能。



12. 设数据总线上接有 A、B、C、D 四个寄存器，要求选用合适的 74 系列芯片，完成下列逻辑设计：

（1） 设计一个电路，在同一时间实现 D→A、D→B 和 D→C 寄存器间的传送；

（2） 设计一个电路，实现下列操作：

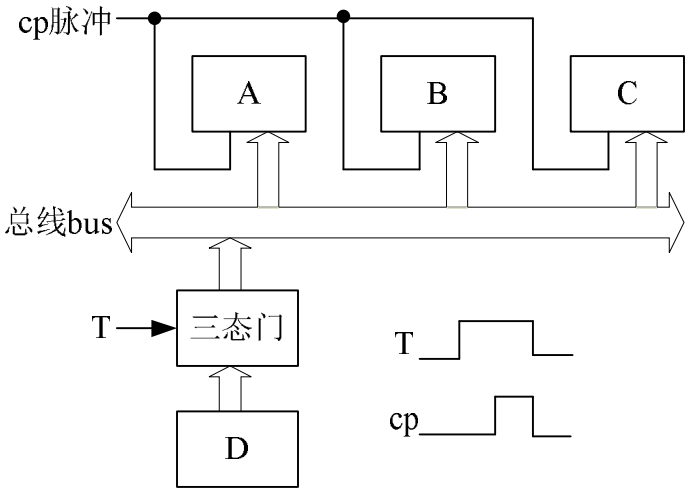
T0 时刻完成 D→总线；

T1 时刻完成总线→A；

T2 时刻完成 A→总线；

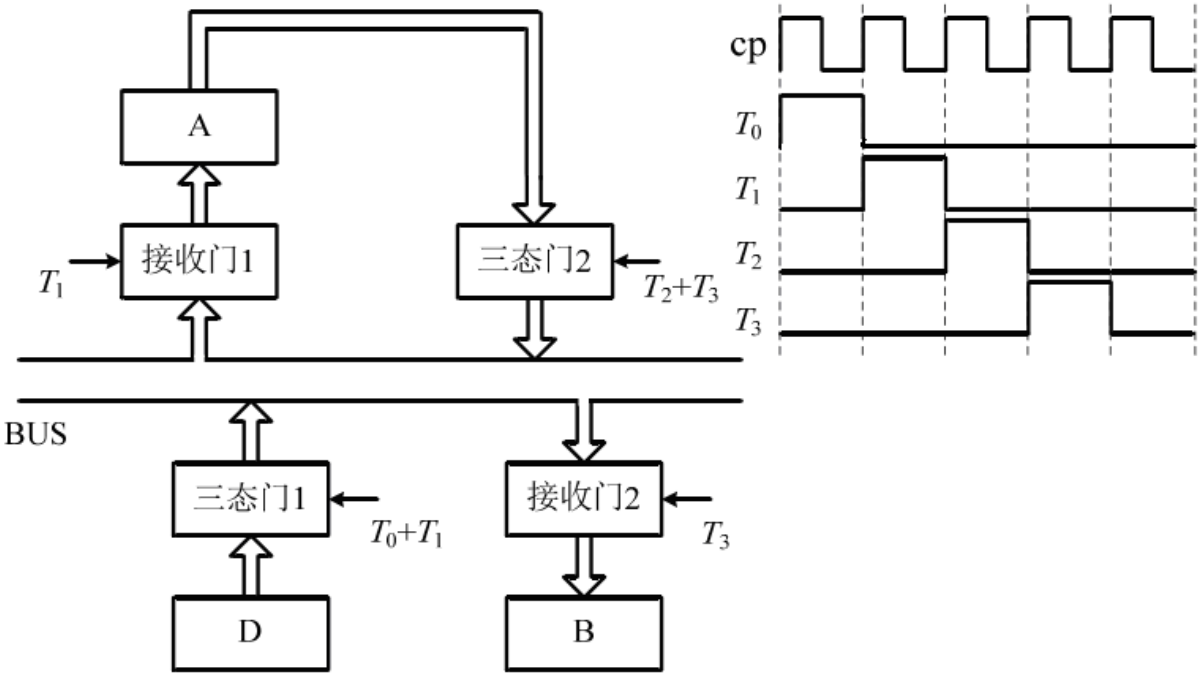
T3 时刻完成总线→B。

解：（1）由 T 打开三态门将 D 寄存器中的内容送至总线 bus，由 cp 脉冲同时将总线上的数据打入到 A、B、C 寄存器中。 T 和 cp 的时间关系如图（1）所示。



图（1）

（2）三态门 1 受 T0＋T1 控制，以确保 T0 时刻 D→总线，以及 T1 时刻总线→接收门 1→A。三态门 2 受 T2＋T3 控制，以确保 T2 时刻 A→总线，以及 T3 时刻总线→接收门 2→B。T0、T1、T2、T3 波形图如图（2）所示。



图(2)

3.14 设总线的时钟频率为 8MHz，一个总线周期等于一个时钟周期。如果一个总线周期中并行传送 16 位数据，试问总线的带宽是多少？

解；总线宽度 = 16 位/8 =2B　　总线带宽 = 8MHz×2B =16MB/s

3.15 在一个 32 位的总线系统中，总线的时钟频率为 66MHz，假设总线最短传输周期为 4 个时钟周期，试计算总线的最大数据传输率。若想提高数据传输率，可采取什么措施？

解法 1： 总线宽度 =32 位/8 =4B　　时钟周期 =1/ 66MHz =0.015μs

总线最短传输周期 =0.015μs×4 =0.06μs

总线最大数据传输率 = 4B/0.06μs =66.67MB/s

解法 2： 总线工作频率 = 66MHz/4 =16.5MHz 总线最大数据传输率=16.5MHz×4B =66MB/s

若想提高总线的数据传输率，可提高总线的时钟频率，或减少总线周期中的时钟个数，或增加总线宽度。

3.16 在异步串行传送系统中，字符格式为：1 个起始位、8 个数据位、1 个校验位、2 个终止位。若要求每秒传送 120 个字符，试求传送的波特率和比特率。

解： 一帧 =1+8+1+2 =12 位 波特率 =120 帧/秒×12 位=1440 波特

比特率 = 1440 波特×（8/12）=960bps 或：比特率 = 120 帧/秒×8 =960bps

第 四 章

0.静态 RAM 与动态 RAM 的区别

静态 RAM(SRAM)速度非常快，只要电源存在内容就不会自动消失。其基本存储电路为 6 个 MOS 管组成 1 位，因此集成度相对较低，功耗也较大。一般高速缓冲存储器用它组成。

动态 RAM(DRAM)的内容在 10-3 或 10-6 秒之后自动消失，因此必须周期性的在内容消失之前进行刷新。由于它的基本存储电路由一个晶体管及一个电容组成，因此它的集成度高，成本较低，另外耗电也少，但它需要一个额外的刷新电路。 DRAM 运行速度较慢， SRAM 比 DRAM 要快 2~5 倍，一般，PC 机的标准存储器都采用 DRAM 组成。

1.辅助储存的技术的指标

(1)记录密度 Dt=1/p Db=f1/πd（min）

(2)储存容量 C=NKS

(3)平均寻址时间 Ta=Tsa+Twd

(4)数据传输率 Dv=Db*V

(5)误码率

2.Cache 的三种映射方式：

（1）直接映射 实现简单，不够灵活。

（2）全相联映射

（3）组相联映射

3. 存储器的层次结构主要体现在什么地方？为什么要分这些层次？计算机如何管理这些层次？

答：存储器的层次结构主要体现在 Cache-主存和主存-辅存这两个存储层次上。

Cache-主存层次在存储系统中主要对 CPU 访存起加速作用，即从整体运行的效果分析，CPU 访存速度加快，接近于 Cache 的速度，而寻址空间和位价却接近于主存。

主存-辅存层次在存储系统中主要起扩容作用，即从程序员的角度看，他所使用的存储器其容量和位价接近于辅存，而速度接近于主存。

综合上述两个存储层次的作用，从整个存储系统来看，就达到了速度快、容量大、位价低的优化效果。

主存与 CACHE 之间的信息调度功能全部由硬件自动完成。而主存与辅存层次的调度目前广泛采用虚拟存储技术实现，即将主存与辅存的一部分通过软硬结合的技术组成虚拟存储器，程序员可使用这个比主存实际空间（物理地址空间）大得多的虚拟地址空间（逻辑地址空间）编程，当程序运行时，再由软、硬件自动配合完成虚拟地址空间与主存实际物理空间的转换。因此，这两个层次上的调度或转换操作对于程序员来说都是透明的。

4. 说明存取周期和存取时间的区别。

解：存取周期和存取时间的主要区别是：存取时间仅为完成一次操作的时间，而存取周期不仅包含操作时间，还包含操作后线路的恢复时间。即：

存取周期 = 存取时间 + 恢复时间

5. 什么是存储器的带宽？若存储器的数据总线宽度为32位，存取周期为200ns，则存储器的带宽是多少？

解：存储器的带宽指单位时间内从存储器进出信息的最大数量。

存储器带宽 = 1/200ns ×32位 = 160M位/秒 = 20MB/秒 = 5M字/秒

注意：字长32位，不是16位。（注：$1ns=10^{-9}s$）

6. 某机字长为32位，其存储容量是64KB，按字编址它的寻址范围是多少？若主存以字节编址，试画出主存字地址和字节地址的分配情况。

解：存储容量是64KB时，按字节编址的寻址范围就是64K，如按字编址，其寻址范围为：

64K / （32/8）= 16K

主存字地址和字节地址的分配情况：（略）。

7. 一个容量为16K×32位的存储器，其地址线和数据线的总和是多少？当选用下列不同规格的存储芯片时，各需要多少片？

1K×4位，2K×8位，4K×4位，16K×1位，4K×8位，8K×8位

解：地址线和数据线的总和 = 14 + 32 = 46根；

选择不同的芯片时，各需要的片数为：

1K×4：（16K×32） / （1K×4） = 16×8 = 128片

2K×8：（16K×32） / （2K×8） = 8×4 = 32片

4K×4：（16K×32） / （4K×4） = 4×8 = 32片

16K×1：（16K×32）/ （16K×1） = 1×32 = 32片

4K×8：（16K×32）/ （4K×8） = 4×4 = 16片

8K×8：（16K×32） / （8K×8） = 2×4 = 8片

9. 什么叫刷新？为什么要刷新？说明刷新有几种方法。

解：刷新：对DRAM定期进行的全部重写过程；

刷新原因：因电容泄漏而引起的DRAM所存信息的衰减需要及时补充，因此安排了定期刷新操作；

常用的刷新方法有三种：集中式、分散式、异步式。

集中式：在最大刷新间隔时间内，集中安排一段时间进行刷新，存在CPU访存死时间。

分散式：在每个读/写周期之后插入一个刷新周期，无CPU访存死时间。

异步式：是集中式和分散式的折衷。

讨论：

1、刷新与再生的比较：

共同点：

- 动作机制一样。都是利用DRAM存储元破坏性读操作时的重写过程实现；
- 操作性质一样。都是属于重写操作。

区别：

- 解决的问题不一样。再生主要解决DRAM存储元破坏性读出时的信息重写问题；刷新主要解决长时间不访存时的信息衰减问题。
- 操作的时间不一样。再生紧跟在读操作之后，时间上是随机进行的；刷新以最大间隔时间为周期定时重复进行。

•动作单位不一样。再生以存储单元为单位，每次仅重写刚被读出的一个字的所有位；刷新以行为单位，每次重写整个存储器所有芯片内部存储矩阵的同一行。

•芯片内部 I/O 操作不一样。读出再生时芯片数据引脚上有读出数据输出；刷新时由于 CAS 信号无效，芯片数据引脚上无读出数据输出（唯 RAS 有效刷新，内部读）。鉴于上述区别，为避免两种操作混淆，分别叫做再生和刷新。

2、CPU 访存周期与存取周期的区别：

CPU 访存周期是从 CPU 一边看到的存储器工作周期，他不一定是真正的存储器工作周期；存取周期是存储器速度指标之一，它反映了存储器真正的工作周期时间。

3、分散刷新是在读写周期之后插入一个刷新周期，而不是在读写周期内插入一个刷新周期，但此时读写周期和刷新周期合起来构成 CPU 访存周期。

4、刷新定时方式有 3 种而不是 2 种，一定不要忘了最重要、性能最好的异步刷新方式。

10. 半导体存储器芯片的译码驱动方式有几种？

解：半导体存储器芯片的译码驱动方式有两种：线选法和重合法。

线选法：地址译码信号只选中同一个字的所有位，结构简单，费器材；

重合法：地址分行、列两部分译码，行、列译码线的交叉点即为所选单元。这种方法通过行、列译码信号的重合来选址，也称矩阵译码。可大大节省器材用量，是最常用的译码驱动方式。

11. 一个 8K×8 位的动态 RAM 芯片，其内部结构排列成 256×256 形式，存取周期为 0.1μs。试问采用集中刷新、分散刷新和异步刷新三种方式的刷新间隔各为多少？

解：采用分散刷新方式刷新间隔为:2ms，其中刷新死时间为：256×0.1μs=25.6μs

采用分散刷新方式刷新间隔为：256×（0.1μs+×0.1μs）=51.2μs

采用异步刷新方式刷新间隔为:2ms

12. 画出用 1024×4 位的存储芯片组成一个容量为 64K×8 位的存储器逻辑框图。要求将 64K 分成 4 个页面，每个页面分 16 组，指出共需多少片存储芯片。

解：设采用 SRAM 芯片，则：

总片数 = （64K×8 位） / （1024×4 位）= 64×2 = 128 片

题意分析：本题设计的存储器结构上分为总体、页面、组三级，因此画图时也应分三级画。首先应确定各级的容量：

页面容量 = 总容量 / 页面数 = 64K×8 / 4 = 16K×8 位，4 片 16K×8 字串联成 64K×8 位

组容量 = 页面容量 / 组数 = 16K×8 位 / 16 = 1K×8 位,16 片 1K×8 位字串联成 16K×8 位

组内片数 = 组容量 / 片容量 = 1K×8 位 / 1K×4 位 = 2 片，两片 1K×4 位芯片位并联成 1K×8 位

存储器逻辑框图：（略）。

13. 设有一个 64K×8 位的 RAM 芯片，试问该芯片共有多少个基本单元电路（简称存储基元）？欲设计一种具有上述同样多存储基元的芯片，要求对芯片字长的选择应满足地址线和数据线的总和为最小，试确定这种芯片的地址线和数据线，并说明有几种解答。

解：存储基元总数 = 64K×8 位 = 512K 位 = 2^{19}位；

思路：如要满足地址线和数据线总和最小，应尽量把存储元安排在字向，因为地址位数和字数成 2 的幂的关系，可较好地压缩线数。

解：设地址线根数为 a，数据线根数为 b，则片容量为：$2^a \times b = 2^{19}$；$b = 2^{19-a}$；

若 a = 19，b = 1，总和 = 19+1 = 20；

a = 18，b = 2，总和 = 18+2 = 20；

a = 17，b = 4，总和 = 17+4 = 21；

a = 16，b = 8，总和 = 16+8 = 24；

…… ……

由上可看出：片字数越少，片字长越长，引脚数越多。片字数减 1、片位数均按 2 的幂变化。

结论：如果满足地址线和数据线的总和为最小，这种芯片的引脚分配方案有两种：地址线 = 19 根，数据线 = 1 根；或地址线 = 18 根，数据线 = 2 根。

14. 某 8 位微型机地址码为 18 位，若使用 4K×4 位的 RAM 芯片组成模块板结构的存储器，试问：

（1）该机所允许的最大主存空间是多少？

（2）若每个模块板为 32K×8 位，共需几个模块板？

（3）每个模块板内共有几片 RAM 芯片？

（4）共有多少片 RAM？

（5）CPU 如何选择各模块板？

解：（1）该机所允许的最大主存空间是：2^{18} × 8 位 = 256K×8 位 = 256KB

（2）模块板总数 = 256K×8 / 32K×8 = 8 块

（3）板内片数 = 32K×8 位 / 4K×4 位 = 8×2 = 16 片

（4）总片数 = 16 片×8 = 128 片

（5）CPU 通过最高 3 位地址译码输出选择模板，次高 3 位地址译码输出选择芯片。地址格式分配如下：

模板号（3位）	芯片号 （3位）	片内地址（12位）

4.15 设 CPU 共有 16 根地址线，8 根数据线，并用 $\overline{\text{MREQ}}$（低电平有效）作访存控制信号，$\text{R}/\overline{\text{W}}$ 作读写命令信号（高电平为读，低电平为写）。现有下列存储芯片：ROM（2K×8 位，4K×4 位，8K×8 位），RAM（1K×4 位，2K×8 位，4K×8 位），及 74138 译码器和其他门电路（门电路自定）。试从上述规格中选用合适芯片，画出 CPU 和存储芯片的连接图。要求：

（1）最小 4K 地址为系统程序区，4096~16383 地址范围为用户程序区；

（2）指出选用的存储芯片类型及数量；

（3）详细画出片选逻辑。

解：（1）地址空间分配图：

系统程序区（ROM 共 4KB）：0000H-0FFFH

用户程序区（RAM 共 12KB）：1000H-FFFFH

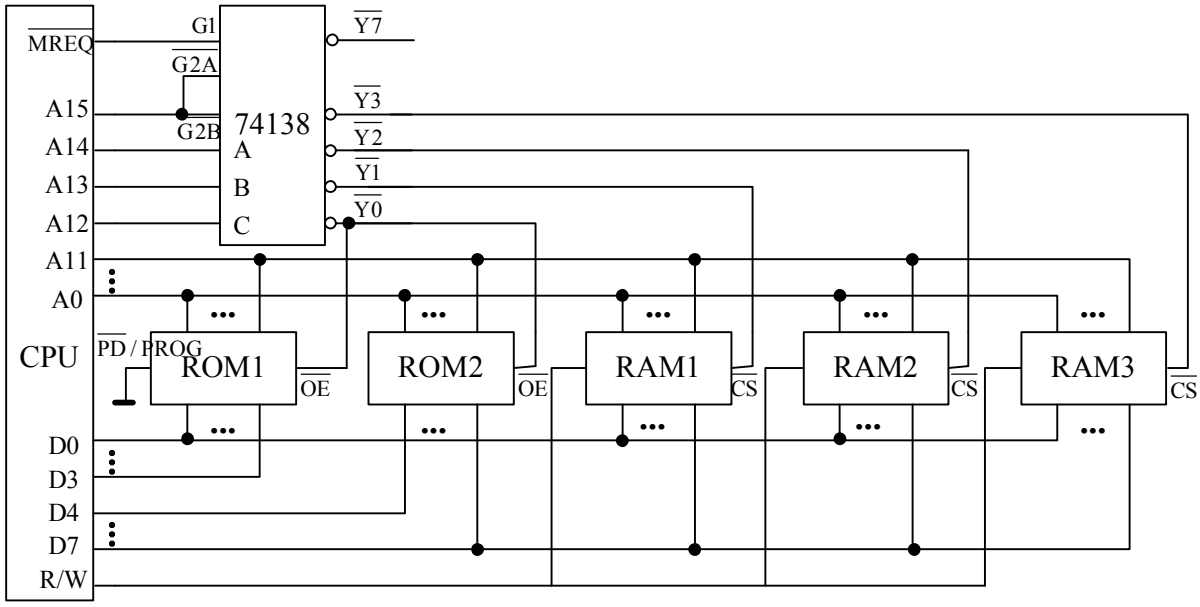
（2）选片：ROM：选择 4K×4 位芯片 2 片，位并联 RAM：选择 4K×8 位芯片 3 片，字串联(RAM1 地址范围为:1000H-1FFFH,RAM2 地址范围为 2000H-2FFFH, RAM3 地址范围为:3000H-3FFFH)

（3）各芯片二进制地址分配如下：

	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
ROM1,2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1
RAM1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
RAM2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1
RAM3	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1

CPU 和存储器连接逻辑图及片选逻辑如下图(3)所示：



图（3）

16. CPU 假设同上题，现有 8 片 8K×8 位的 RAM 芯片与 CPU 相连，试回答：

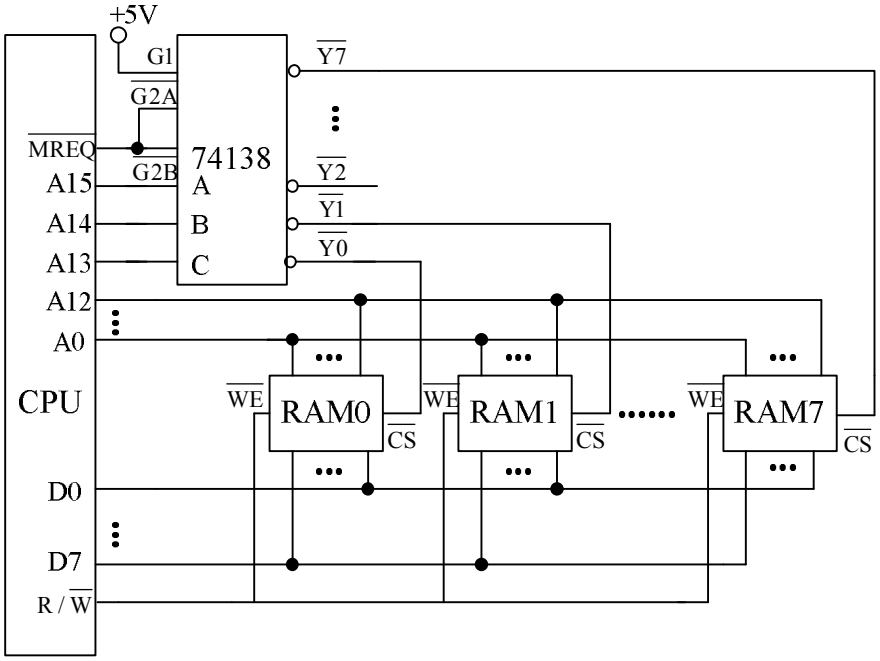
（1）用 74138 译码器画出 CPU 与存储芯片的连接图；

（2）写出每片 RAM 的地址范围；

（3）如果运行时发现不论往哪片 RAM 写入数据后，以 A000H 为起始地址的存储芯片都有与其相同的数据，分析故障原因。

（4）根据（1）的连接图，若出现地址线 A13 与 CPU 断线，并搭接到高电平上，将出现什么后果？

解：（1）CPU 与存储器芯片连接逻辑图：



（2）地址空间分配图：

RAM0:0000H-1FFFH

RAM1:2000H-3FFFH

RAM2:4000H-5FFFH

RAM3:6000H-7FFFH

RAM4:8000H-9FFFH

RAM5:A000H-BFFFH

RAM6:C000H-DFFFH

RAM7:E000H-FFFFH

（3）如果运行时发现不论往哪片 RAM 写入数据后，以 A000H 为起始地址的存储芯片(RAM5)都有与其相同的数据，则根本的故障原因为：该存储芯片的片选输入端很可能总是处于低电平。假设芯片与译码器本身都是好的，可能的情况有：

1）该片的-CS 端与-WE 端错连或短路；

2）该片的-CS 端与 CPU 的-MREQ 端错连或短路；

3）该片的-CS 端与地线错连或短路。

（4）如果地址线 A13 与 CPU 断线，并搭接到高电平上，将会出现 A13 恒为“1”的情况。此时存储器只能寻址 A13=1 的地址空间(奇数片)，A13=0 的另一半地址空间（偶数片）将永远访问不到。若对 A13=0 的地址空间（偶数片）进行访问，只能错误地访问到 A13=1 的对应空间(奇数片)中去。

17. 写出 1100、1101、1110、1111 对应的汉明码。

解：有效信息均为 n=4 位，假设有效信息用 b4b3b2b1 表示

校验位位数 k=3 位，（2^k>=n+k+1）

设校验位分别为 c1、c2、c3，则汉明码共 4+3=7 位，即：c1c2b4c3b3b2b1

校验位在汉明码中分别处于第 1、2、4 位

c1=b4⊕b3⊕b1

c2=b4⊕b2⊕b1

c3=b3⊕b2⊕b1

当有效信息为 1100 时，c3c2c1=011,汉明码为 1110100。

当有效信息为 1101 时，c3c2c1=100,汉明码为 0011101。

当有效信息为 1110 时，c3c2c1=101,汉明码为 1011110。

当有效信息为 1111 时，c3c2c1=010,汉明码为 0110111。

18. 已知收到的汉明码（按配偶原则配置）为 1100100、1100111、1100000、1100001，检查上述代码是否出错？第几位出错？

解：假设接收到的汉明码为：c1’ c2’ b4’ c3’ b3’ b2’ b1’

纠错过程如下：

P1=c1’ ⊕b4’ ⊕b3’ ⊕b1’

P2=c2’ ⊕b4’ ⊕b2’ ⊕b1’

P3=c3’ ⊕b3’ ⊕b2’ ⊕b1’

如果收到的汉明码为 1100100，则 p3p2p1=011，说明代码有错，第 3 位（b4’）出错，有效信息为：1100

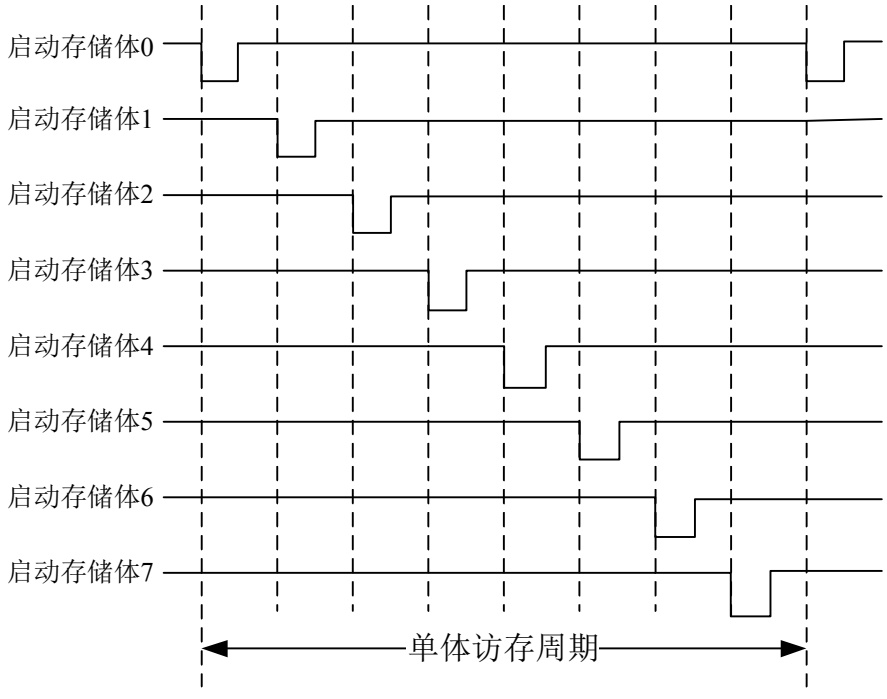
如果收到的汉明码为 1100111，则 p3p2p1=111，说明代码有错，第 7 位（b1’）出错，有效信息为：0110

如果收到的汉明码为 1100000，则 p3p2p1=110，说明代码有错，第 6 位（b2’）出错，有效信息为：0010

如果收到的汉明码为 1100001，则 p3p2p1=001，说明代码有错，第 1 位（c1’）出错，有效信息为：0001

22. 某机字长 16 位，常规的存储空间为 64K 字，若想不改用其他高速的存储芯片，而使访存速度提高到 8 倍，可采取什么措施？画图说明。

解：若想不改用高速存储芯片，而使访存速度提高到 8 倍，可采取八体交叉存取技术，8 体交叉访问时序如下图：



18. 什么是“程序访问的局部性”？存储系统中哪一级采用了程序访问的局部性原理？
解：程序运行的局部性原理指：在一小段时间内，最近被访问过的程序和数据很可能再次被访问；在空间上，这些被访问的程序和数据往往集中在一小片存储区；在访问顺序上，指令顺序执行比转移执行的可能性大（大约 5:1）。存储系统中 Cache—主存层次采用了程序访问的局部性原理。

25. Cache 做在 CPU 芯片内有什么好处？将指令 Cache 和数据 Cache 分开又有什么好处？
答：Cache 做在 CPU 芯片内主要有下面几个好处：

1）可提高外部总线的利用率。因为 Cache 在 CPU 芯片内，CPU 访问 Cache 时不必占用外部总线。

2）Cache 不占用外部总线就意味着外部总线可更多地支持 I/O 设备与主存的信息传输，增强了系统的整体效率。

3）可提高存取速度。因为 Cache 与 CPU 之间的数据通路大大缩短,故存取速度得以提高。

将指令 Cache 和数据 Cache 分开有如下好处：

1）可支持超前控制和流水线控制，有利于这类控制方式下指令预取操作的完成。

2）指令 Cache 可用 ROM 实现，以提高指令存取的可靠性。

3）数据 Cache 对不同数据类型的支持更为灵活，既可支持整数（例 32 位），也可支持浮点数据（如 64 位）。

补充：

Cache 结构改进的第三个措施是分级实现，如二级缓存结构，即在片内 Cache（L1）和主存之间再设一个片外 Cache（L2），片外缓存既可以弥补片内缓存容量不够大的缺点，又可在主存与片内缓存间起到平滑速度差的作用，加速片内缓存的调入调出速度。

30. 一个组相连映射的 CACHE 由 64 块组成，每组内包含 4 块。主存包含 4096 块，每块由 128 字组成，访存地址为字地址。试问主存和高速存储器的地址各为几位？画出主存地址格式。
解：cache 组数：64/4=16 ，Cache 容量为：$64*128=2^{13}$字，cache 地址 13 位

主存共分 4096/16=256 区，每区 16 块

主存容量为：$4096*128=2^{19}$字，主存地址 19 位，地址格式如下：

主存字块标记（8 位）	组地址（4 位）	字块内地址（7 位）

32. 设某机主存容量为 4MB，Cache 容量为 16KB，每字块有 8 个字，每字 32 位，设计一个四路组相联映象（即 Cache 每组内共有 4 个字块）的 Cache 组织，要求：
（1）画出主存地址字段中各段的位数；
（2）设 Cache 的初态为空，CPU 依次从主存第 0、1、2……99 号单元读出 100 个字（主存一次读出一个字），并重复按此次序读 8 次，问命中率是多少？
（3）若 Cache 的速度是主存的 6 倍，试问有 Cache 和无 Cache 相比，速度提高多少倍？
答：
（1）由于容量是按字节表示的，则主存地址字段格式划分如下：

8	7	2	3	2

（2）由于题意中给出的字地址是连续的，故（1）中地址格式的最低 2 位不参加字的读出操作。当主存读 0 号字单元时，将主存 0 号字块（0~7）调入 Cache（0 组 x 号块），主存读 8 号字单元时，将 1 号块（8~15）调入 Cache（1 组 x 号块）…… 主存读 96 号单元时，将 12 号块（96~103）调入 Cache（12 组 x 号块）。
≈ 共需调 100/8 13 次，就把主存中的 100 个数调入 Cache。除读第 1 遍时 CPU 需访问主存 13 次外，以后重复读时不需再访问主存。则在 800 个读操作中：

访 Cache 次数=（100-13）+700=787 次

≈ 0.98 ≈ Cache 命中率=787/800 98%

（3）设无 Cache 时访主存需时 800T（T 为主存周期），加入 Cache 后需时：

（131.167+13）T≈T/6+13T×787

144.167T≈

5.55 倍≈则：800T/144.167T

有 Cache 和无 Cache 相比，速度提高 4.55 倍左右。

38. 磁盘组有六片磁盘，每片有两个记录面，存储区域内径 22 厘米，外径 33 厘米，道密度为 40 道/厘米，内层密度为 400 位/厘米，转速 2400 转/分，问：

（1）共有多少存储面可用？

（2）共有多少柱面？

（3）盘组总存储容量是多少？

（4）数据传输率是多少？

解：

（1）若去掉两个保护面，则共有：

6 × 2 - 2 = 10 个存储面可用；

（2）有效存储区域

=（33-22）/ 2 = 5.5cm

柱面数 = 40 道/cm × 5.5= 220 道

=π （3）内层道周长=22 69.08cm

道容量=400 位/cm×69.08cm

= 3454B

面容量=3454B × 220 道

= 759，880B

盘组总容量 = 759，880B × 10 面

= 7，598，800B

（4）转速 = 2400 转 / 60 秒

= 40 转/秒

数据传输率 = 3454B × 40 转/秒

= 138，160 B/S

注意：

1）计算盘组容量时一般应去掉上、下保护面；

的精度选取不同将引起答案不同，一般取两位小数；π2）

盘组总磁道数（=一个盘面上的磁道数）≠3）柱面数

4）数据传输率与盘面数无关；

5）数据传输率的单位时间是秒，不是分。

4.39 某磁盘存储器转速为 3000 转/分，共有 4 个记录盘面，每毫米 5 道，每道记录信息 12 288 字节，最小磁道直径为 230mm，共有 275 道，求：

（1）磁盘存储器的存储容量；

（2）最高位密度（最小磁道的位密度）和最低位密度；

（3）磁盘数据传输率；

（4）平均等待时间。

解： （1）存储容量 = 275 道×12 288B/道×4 面 = 13 516 800B

（2）最高位密度 = 12 288B/230π≈17B/mm≈136 位/mm（向下取整）

最大磁道直径=230mm+275 道/5 道 × 2= 230mm + 110mm = 340mm

最低位密度 = 12 288B / 340π≈11B/mm≈92 位 / mm （向下取整）

（3）磁盘数据传输率= 12 288B × 3000 转/分=12 288B × 50 转/秒=614 400B/S

（4）平均等待时间 = 1/50 / 2 = 10ms

讨论：1、本题给出的道容量单位为字节，因此算出的存储容量单位也是字节，而不是位；

2、由此算出的位密度单位最终应转换成 bpm(位/毫米)；

3、平均等待时间是磁盘转半圈的时间，与容量无关。

4.41 设有效信息为 110，试用生成多项式 G(x) =11011 将其编成循环冗余校验码。

解：编码过程如下：

M(x) =110 n =3

G(x) =11011 k+1 =5 k =4

M(x)·x4 =110 0000

M(x)·x4/G(x) =110 0000/11011=100+1100/11011 R(x) =1100

M(x)·x4+R(x) =110 0000+1100=110 1100 =CRC 码 （7，3）码

注：此题的 G(x)选得不太好，当最高位和最低位出错时，余数相同，均为 0001。此时只能检错，无法纠错

第 五 章

1. I/O 有哪些编址方式？各有何特点？

解：常用的 I/O 编址方式有两种： I/O 与内存统一编址和 I/O 独立编址；

特点： I/O 与内存统一编址方式的 I/O 地址采用与主存单元地址完全一样的格式，I/O 设备和主存占用同一个地址空间，CPU 可像访问主存一样访问 I/O 设备，不需要安排专门的 I/O 指令。

I/O 独立编址方式时机器为 I/O 设备专门安排一套完全不同于主存地址格式的地址编码，此时 I/O 地址与主存地址是两个独立的空间，CPU 需要通过专门的 I/O 指令来访问 I/O 地址空间。

讨论：I/O 编址方式的意义：

I/O 编址方式的选择主要影响到指令系统设计时 I/O 指令的安排，因此描述其特点时一定要说明此种 I/O 编址方式对应的 I/O 指令设置情况。

× I/O 与内存统一编址方式将 I/O 地址看成是存储地址的一部分，占用主存空间；

问题：确切地讲，I/O 与内存统一编址的空间为总线空间，I/O 所占用的是内存的扩展空间。

I/O 独立编址方式有明显的 I/O 地址标识，× 而 I/O 与内存统一的编址方式没有；

问题：无论哪种编址方式，I/O 地址都是由相应的指令提供的，而地址本身并没有特殊的标识。

2. 简要说明 CPU 与 I/O 之间传递信息可采用哪几种联络方式？它们分别用于什么场合？

答： CPU 与 I/O 之间传递信息常采用三种联络方式：直接控制（立即响应）、 同步、异步。适用场合分别为：

直接控制适用于结构极简单、速度极慢的 I/O 设备，CPU 直接控制外设处于某种状态而无须联络信号。

同步方式采用统一的时标进行联络，适用于 CPU 与 I/O 速度差不大，近距离传送的场合。

异步方式采用应答机制进行联络，适用于 CPU 与 I/O 速度差较大、远距离传送的场合。

讨论：注意 I/O 交换方式、I/O 传送分类方式与 I/O 联络方式的区别：

串行、并行 I/O 传送方式常用于描述 I/O 传送宽度的类型；

I/O 交换方式主要讨论传送过程的控制方法；

I/O 联络方式主要解决传送时 CPU 与 I/O 之间如何取得通信联系以建立起操作上的同步配合关系。

× 同步方式适用于 CPU 与 I/O 工作速度完全同步的场合。

问题： I/O 要达到与 CPU 工作速度完全同步一般是不可能的。同步方式的实质是"就慢不就快"，如采用同步方式一般 CPU 达不到满负荷工作。

5.3 .I/O 设备与主机交换信息时，共有哪鸡枞控制方式？简述它们的特点。

(1）程序直接控制方式：也称查询方式，采用该方式，数据在 CPU 和外设间的传送完全靠计算机程序控制，CPU 的操作和外围设备操作同步，硬件结构简单，但由于外部设备动作慢，浪费 CPU 时间多，系统效率低。

（2）程序中断方式：外设备准备就绪后中断方式猪肚通知 CPU，在 CPU 相应 I/O 设备的中断请求后，在暂停现行程序的执行，转为 I/O 设备服务科明显提高 CPU 的利用率，在一定程度上实现了主机和 I/O 设备的并行工作，但硬件结构负载，服务开销时间大、

（3）DMA 方式与中断方式一样，实现了主机和 I/O 设备的并行工作，由于 DMA 方式直接依靠硬件实现贮存与 I/O 设备之间的数据传送，传送期间不需要 CPU 程序干预，CPU 可继续执行原来的程序，因此 CPU 利用率和系统效率比中断方式更高，但 DMA 方式的硬件结构更为复杂。

6. 字符显示器的接口电路中配有缓冲存储器和只读存储器，各有何作用？

解：显示缓冲存储器的作用是支持屏幕扫描时的反复刷新；只读存储器作为字符发生器使用，他起着将字符的 ASCII 码转换为字形点阵信息的作用。

8. 某计算机的 I/O 设备采用异步串行传送方式传送字符信息。字符信息的格式为一位起始位、七位数据位、一位校验位和一位停止位。若要求每秒钟传送 480 个字符，那么该设备的数据传送速率为多少？

解：480×10=4800 位/秒=4800 波特；

波特——是数据传送速率波特率的单位。

注：题意中给出的是字符传送速率，即：字符/秒。要求的是数据传送速率，串行传送时一般用波特率表示。

两者的区别：字符传送率是数据的"纯"有效传送率，不含数据格式信息；波特率是"毛"传送率，含数据格式信息。

10. 什么是 I/O 接口?为什么要设置 I/O 接口？I/O 接口如何分类？

解： I/O 接口一般指 CPU 和 I/O 设备间的连接部件； I/O 接口分类方法很多，主要有：

按数据传送方式分有并行接口和 串行接口两种；

按数据传送的控制方式分有程序控制接口、程序中断接口、DMA 接口三种。

5.12 结合程序查询方式的接口电路，说明其工作过程。

解：程序查询接口工作过程如下（以输入为例）：1）CPU 发 I/O 地址→地址总线→接口→设备选择器译码→选中，发 SEL 信号→开命令接收门；2）CPU 发启动命令→ D 置 0，B 置 1 →接口向设备发启动命令→设备开始工作；3）CPU 等待，输入设备读出数据→ DBR；4）外设工作完成，

完成信号→接口→ B 置 0，D 置 1；5）准备就绪信号→控制总线→ CPU；6）输入：CPU 通过输入指令（IN）将 DBR 中的数据取走；

若为输出，除数据传送方向相反以外，其他操作与输入类似。工作过程如下：1）CPU 发 I/O 地址→地址总线→接口→设备选择器译码→选中，发 SEL 信号→开命令接收门；2）输出： CPU 通过输出指令（OUT）将数据放入接口 DBR 中； 3）CPU 发启动命令→ D 置 0，B 置 1 →接口向设备发启动命令→设备开始工作；4）CPU 等待，输出设备将数据从 DBR 取走；5）外设工作完成，完成信号→接口→ B 置 0，D 置 1；6）准备就绪信号→控制总线→ CPU，CPU 可通过指令再次向接口 DBR 输出数据，进行第二次传送。

13. 说明中断向量地址和入口地址的区别和联系。

解：

中断向量地址和入口地址的区别：

向量地址是硬件电路（向量编码器）产生的中断源的内存地址编号，中断入口地址是中断服务程序首址。

中断向量地址和入口地址的联系：

中断向量地址可理解为中断服务程序入口地址指示器（入口地址的地址），通过它访存可获得中断服务程序入口地址。 (两种方法：在向量地址所指单元内放一条 JUM 指令；主存中设向量地址表。参考 8.4.3）

讨论：

硬件向量法的实质：

当响应中断时，为了更快、更可靠的进入对应的中断服务程序执行，希望由硬件直接提供中断服务程序入口地址。但在内存地址字较长时这是不可能的。因此由硬件先提供中断源编号、再由编号间接地获得中断服务程序入口地址。这种中断源的编号即向量地址。

由于一台计算机系统可带的中断源数量很有限，因此向量地址比内存地址短得多，用编码器类逻辑部件实现很方便。

14. 在什么条件下，I/O 设备可以向 CPU 提出中断请求？

解：I/O 设备向 CPU 提出中断请求的条件是：I/O 接口中的设备工作完成状态为 1（D=1），中断屏蔽码为 0 （MASK=0），且 CPU 查询中断时，中断请求触发器状态为 1（INTR=1）。

15. 什么是中断允许触发器？它有何作用？

解：中断允许触发器是 CPU 中断系统中的一个部件，他起着开关中断的作用（即中断总开关，则中断屏蔽触发器可视为中断的分开关）。

16. 在什么条件和什么时间，CPU 可以响应 I/O 的中断请求？

解：CPU 响应 I/O 中断请求的条件和时间是：当中断允许状态为 1（EINT=1），且至少有一个中断请求被查到，则在一条指令执行完时，响应中断。

17. 某系统对输入数据进行取样处理，每抽取一个输入数据，CPU 就要中断处理一次，将取样的数据存至存储器的缓冲区中，该中断处理需 P 秒。此外，缓冲区内每存储 N 个数据，主程序就要将其取出进行处理，这个处理需 Q 秒。试问该系统可以跟踪到每秒多少次中断请求？

解：这是一道求中断饱和度的题，要注意主程序对数据的处理不是中断处理，因此 Q 秒不能算在中断次数内。

N 个数据所需的处理时间=P×N+Q 秒

平均每个数据所需处理时间= （P×N+Q） /N 秒；
求倒数得：
该系统跟踪到的每秒中断请求数=N/（P×N+Q）次。

19. 在程序中断方式中，磁盘申请中断的优先权高于打印机。当打印机正在进行打印时，磁盘申请中断请求。试问是否要将打印机输出停下来，等磁盘操作结束后，打印机输出才能继续进行？为什么？

解：这是一道多重中断的题，由于磁盘中断的优先权高于打印机，因此应将打印机输出停下来，等磁盘操作结束后，打印机输出才能继续进行。因为打印机的速度比磁盘输入输出的速度慢，并且暂停打印不会造成数据丢失。

讨论：
× 打印机不停，理由有如下几种：
打印内容已存入打印机缓存；×
问题：1）如果打印机无缓存呢？
2）如果打印机有缓存，还需要用程序中断方式交换吗？（应用 DMA)
由于在指令执行末查中断，因此执行打印指令时不会响应磁盘中断。×
问题：打印中断处理程序=打印指令？
采用字节交叉传送方式，当两者同时请求中断时，先响应盘，再响应打印机，交叉服务。×
问题：这是程序中断方式吗？
由于打印机速度比 CPU 慢得多，CPU 将数据发送给打印机后，就去为磁盘服务，而这时打印机可自己慢慢打印。×
问题：停止打印机传送=停止打印机动作？
我有打印机，感觉上打印机工作是连贯的；×
问题：人的感觉速度=计算机工作速度？

28. CPU 对 DMA 请求和中断请求的响应时间是否一样？为什么？

解： CPU 对 DMA 请求和中断请求的响应时间不一样，因为两种方式的交换速度相差很大，因此 CPU 必须以更短的时间间隔查询并响应 DMA 请求（一个存取周期末）。

讨论：
CPU 对 DMA 的响应是即时的；×
随时都能响应？
CPU 响应 DMA 的时间更短；×
× DMA 比中断速度高；
短、高或不一样的具体程度？
× 不一样。因为 DMA 与 CPU 共享主存，会出现两者争用主存的冲突，CPU 必须将总线让给 DMA 接口使用，常用停止 CPU 访存、周期窃取及 DMA 与 CPU 交替访存三种方式有效的分时使用主存；
这种情况仅仅存在于 DMA 与中断程序之间吗？答非所问。

30. DMA 的工作方式中，CPU 暂停方式和周期挪用方式的数据传送流程有何不同？画图说明。

解：两种 DMA 方式的工作流程见下页，其主要区别在于传送阶段，现行程序是否完全停止访存。

停止 CPU 访存方式的 DMA 工作流程如下：

现行程序　　CPU　　DMAC　　I/O

　　CPU　　DMAC　　I/O

　　B　　C　　D

周期窃取方式的 DMA 工作流程如下：

现行程序　　CPU　　DMAC　　I/O

　　CPU　　DMAC　　I/O

　　B　　C　　D

31. s，试问该外设是否可用程序中断方式与主机交换信息，为什么？μ假设某设备向 CPU 传送信息的最高频率是 40K 次/秒，而相应的中断处理程序其执行时间为 40

sμ　解：该设备向 CPU 传送信息的时间间隔 =1/40K=0.025×103=25 < sμ40

则：该外设不能用程序中断方式与主机交换信息，因为其中断处理程序的执行速度比该外设的交换速度慢。

讨论：

s）比较接近，传送过程会频繁的打断 CPU 执行主程序，而执行中断服务程序，因此不能用程序中断方式……。μs）与中断处理时间（40μ×I/O 传送（25

错：此时 CPU 还有可能执行主程序吗？

举例说明：　（输入）

假设初始 CPU 空闲，则当 I/O 将第一个数据放在接口的数据缓冲寄存器中后，向 CPU 发第一个中断请求，CPU 立即响应；

I/O 设备匀速运行， s 时响应；μs 后，第二个中断请求到来，CPU 正在执行中断程序接收第一个数据， 40μ25

s 时响应；μs 后，第三个中断请求到来，CPU 正在执行中断程序接收第二个数据，要到 80μ50

s 后，第四个中断请求到来，但此时第三个中断请求还没有响应，则放在数据缓冲寄存器中的第三个数据来不及接收，被第四个数据冲掉；μ75

讨论：

s，CPU 大部分时间处于"踏步等待"状态；μ 交换一次用时 25+40=65×

sμ　　错 1：25 I/O 传送间隔主要指设备准备数据的时间（输入），这段时间设备与 CPU 并行工作。

错 2：程序中断不存在踏步等待。

×　　10-6=1×40×40K.6 秒，时间过长，用程序中断不划算；

中断处理程序执行时间=？×　　错 1：设备传送频率

错 2：越慢速的设备越适合用中断。

若外设与 CPU 之间有足够大的缓冲区，则可以用程序中断方式；×

如果安排足够大的缓冲区，为何不用 DMA 方式？

讨论（续）：

两者速度相差较小没有必要用中断。×

32. 是否可采用一条指令执行结束时响应 DMA 请求的方案，为什么？若不行，应采取什么方案？μ设磁盘存储器转速为 3000 转/分，分 8 个扇区，每扇区存储 1K 字节，主存与磁盘存储器数据传送的宽度为 16 位（即每次传送 16 位）。假设一条指令最长执行时间是 25

解：先算出磁盘传送速度，然后和指令执行速度进行比较得出结论。

16÷16 =1K ×8 ×8 ÷道容量=1KB×8

=1K ×4=4K 字

数传率=4K 字×3000 转/分

=4K 字×50 转/秒 =200K 字/秒

sμ5≈一个字的传送时间=1/200K 字/秒

注：在此 1K=1024，来自数据块单位缩写。

5 sμ<<25 s，所以不能采用一条指令执行结束响应 DMA 请求的方案，应采取每个 CPU 机器周期末查询及响应 DMA 请求的方案（通常安排 CPU 机器周期=MM 存取周期）。μ

讨论：

扇面、扇段和扇区：扇面指磁盘分区后形成的扇形区域；扇段指扇面上一个磁道所对应的弧形区域；扇区通常用来泛指扇面或扇段。由于磁盘是沿柱面存取而不是沿扇面存取，因此习惯上扇区即指扇段，不用特别说明也不会引起误会。

问题：是否磁盘转一圈读完所有扇区上的磁道？

答：应为：磁盘转一圈读完一个磁道上的所有扇区，然后转到下一盘面的同一位置磁道接着读(如果文件未读完的话)。

× s，CPU 工作周期大于主存周期，应采用 DMA 与 CPU 交替访存；μs，CPU 执行指令 20μs 内主存占用 5μ不行，在 25

s；μ错 1：题意为 CPU 执行指令 25

指令周期；≠ 错 2：CPU 工作周期=内存周期（同步控制）而

× 不行，传送间隔=20ms，远大于指令执行周期，应在 DMA 接口设一小容量存储器，可减少 DMA 传送占用总线时间；

对于想采用 DMA 的慢速设备（像打印机等），可采用此法，对于磁盘不需要。另外，占用总线时间较长的 DMA 传送为停止 CPU 访存 DMA，如采用周期窃取方式的 DMA，每次传送只占一个主存周期时间。

33. 试从下面七个方面比较程序查询、程序中断和 DMA 三种方式的综合性能。

（1）数据传送依赖软件还是硬件；

（2）传送数据的基本单位；

（3）并行性；

（4）主动性；

（5）传输速度；

（6）经济性；

（7）应用对象。

解：比较如下：

（1）程序查询、程序中断方式的数据传送主要依赖软件，DMA 主要依赖硬件。 （注意：这里指主要的趋势）

（2）程序查询、程序中断传送数据的基本单位为字或字节，DMA 为数据块。

（3）程序查询方式传送时，CPU 与 I/O 设备串行工作；

程序中断方式时，CPU 与 I/O 设备并行工作，现行程序与 I/O 传送串行进行；

DMA 方式时，CPU 与 I/O 设备并行工作，现行程序与 I/O 传送并行进行。

（4）程序查询方式时，CPU 主动查询 I/O 设备状态；

程序中断及 DMA 方式时，CPU 被动接受 I/O 中断请求或 DMA 请求。

（5）程序中断方式由于软件额外开销时间比较大，因此传输速度最慢；

程序查询方式软件额外开销时间基本没有，因此传输速度比中断快；

DMA 方式基本由硬件实现传送，因此速度最快；

注意：程序中断方式虽然 CPU 运行效率比程序查询高，但传输速度却比程序查询慢。

（6）程序查询接口硬件结构最简单，因此最经济；

程序中断接口硬件结构稍微复杂一些，因此较经济；

DMA 控制器硬件结构最复杂，因此成本最高；

（7）程序中断方式适用于中、低速设备的 I/O 交换；

程序查询方式适用于中、低速实时处理过程；

DMA 方式适用于高速设备的 I/O 交换；

讨论：

问题 1：这里的传送速度指 I/O 设备与主存间，还是 I/O 与 CPU 之间？

答：视具体传送方式而定，程序查询、程序中断为 I/O 与 CPU 之间交换，DMA 为 I/O 与主存间交换。

问题 2：主动性应以 CPU 的操作方式看，而不是以 I/O 的操作方式看。

程序查询方式：以缓冲器容量（块、二进制数字）为单位传送；×

×　　程序中断方式：以向量地址中的数据（二进制编码）为单位传送；

DMA：传送单位根据数据线的根数而定；×

30. 什么是多重中断？实现多重中断的必要条件是什么？

解：多重中断是指：当 CPU 执行某个中断服务程序的过程中，发生了更高级、更紧迫的事件，CPU 暂停现行中断服务程序的执行，转去处理该事件的中断，处理完返回现行中断服务程序继续执行的过程。

实现多重中断的必要条件是：在现行中断服务期间，中断允许触发器为 1，即开中断。

第　六　章

2. 已知 X=0.a1a2a3a4a5a6（ai 为 0 或 1），讨论下列几种情况时 ai 各取何值。

（1）X >　1/8；　≥1/2；

（2）X　 X≥

（3）1/4　> 1/16

解：　（1）若要 X > 1/2，只要 a1=1，a2~a6 不全为 0 即可（a2 or a3 or a4　1/8，只要 a1~a3 不全为 0 即可（a1 or a2 or a3 =1），≥or a5 or a6 = 1）；　　（2）若要 X　a4~a6 可任取 0 或 1；

X≥（3）若要 1/4　> 1/16，只要 a1=0，a2 可任取 0 或 1；　　当 a2=0 时，若 a3=0，则必须 a4=1，且 a5、a6 不全为 0（a5 or a6=1；若 a3=1，则 a4~a6 可任取 0 或 1；　　当 a2=1 时，　a3~a6 可任取 0 或 1。

3. 设 x 为整数，[x]补=1，x1x2x3x4x5，若要求 x < -16，试问 x1~x5 应取何值？

解：若要 x < -16，需 x1=0，x2~x5 任意。（注：负数绝对值大的补码码值反而小。）

6.4　设机器数字长为 8 位（含 1 位符号位在内），写出对应下列各真值的原码、补码和反码。

-13/64，29/128，100，-87

解：真值与不同机器码对应关系如下：

真　　值				
十进制	二进制	原　码	反　码	补　码
-13/64	-0.00 1101	1.001 1010	1.110 0101	1.110 0110

29/128	0.001 1101	0.001 1101	0.001 1101	0.001 1101
100	110 0100	0,110 0100	0,110 0100	0,110 0100
-87	-101 0111	1,101 0111	1,010 1000	1,010 1001

5. 已知[x]补，求[x]原和 x。 [x1]补=1. 1100； [x2]补=1. 1001； [x3]补=0. 1110； [x4]补=1. 0000； [x5]补=1，0101； [x6]补=1，1100； [x7]补=0，0111； [x8]补=1，0000；

解：[x]补与[x]原、x 的对应关系如下：

6. 设机器数字长为 8 位（含 1 位符号位在内），分整数和小数两种情况讨论真值 x 为何值时，[x]补=[x]原成立。

解： 当 x 为小数时，若 x >0 [x]补=[x]原成立； 若 x < 0，则当 x= -1/2 时， [x]补=[x]原成立。 0，则 [x]补=[x]原成立； 若 x≥当 x 为整数时，若 x < 0，则当 x= -64 时， [x]补=[x]原成立。

7. 设 x 为真值，x*为绝对值，说明[-x*]补=[-x]补能否成立。

解：当 x 为真值，x*为绝对值时，[-x*]补=[-x]补不能成立。 [-x*]补=[-x]补的结论只在 x>0 时成立。当 x<0 时，由于[-x*]补是一个负值，而[-x]补是一个正值，因此此时[-x*]补不等于[-x]补。

8. 讨论若[x]补>[y]补，是否有 x>y?

解：若[x]补>[y]补，不一定有 x>y。 [x]补 > [y]补时 x > y 的结论只在 x > 0、y > 0，及 x<0、y<0 时成立。当 x>0、 y<0 时，有 x>y，但由于负数补码的符号位为 1，则[x]补<[y]补。同样，当 x<0、 y >0 时，有 x < y，但[x]补>[y]补。

注意： 1）绝对值小的负数其值反而大，且负数的绝对值越小，其补码值越大。因此， 当 x<0、y<0 时，若[x]补>[y]补，必有 x>y。 2）补码的符号位和数值位为一体，不可分开分析。 3）完整的答案应分四种情况分析，但也可通过充分分析一种不成立的情况获得正确答案。 4）由于补码 0 的符号位为 0，因此 x、y=0 可归纳到>0 的一类情况讨论。 5）不考虑不同数字系统间的比较。（如有人分析 x、y 字长不等时的情况，无意义。）

12. 设浮点数格式为：阶码 5 位（含 1 位阶符），尾数 11 位（含 1 位数符）。写出 51/128、-27/1024 所对应的机器数。要求如下：

（1）阶码和尾数均为原码。

（2）阶码和尾数均为补码。

（3）阶码为移码，尾数为补码。

解：据题意画出该浮点数的格式：

阶符 1 位	阶码 4 位	数符 1 位	尾数 10 位

将十进制数转换为二进制：x1= 51/128= 0.0110011B= 2^{-1} * 0.110 011B

x2= -27/1024= -0.0000011011B = 2^{-5}*(-0.11011B)

则以上各数的浮点规格化数为：

（1）[x1]浮=1，0001；0.110 011 000 0

[x2]浮=1，0101；1.110 110 000 0

（2）[x1]浮=1，1111；0.110 011 000 0

[x2]浮=1，1011；1.001 010 000 0

（3）[x1]浮=0，1111；0.110 011 000 0

[x2]浮=0，1011；1.001 010 000 0

13. 浮点数格式同上题，当阶码基值分别取 2 和 16 时，（1）说明 2 和 16 在浮点数中如何表示。（2）基值不同对浮点数什么有影响？（3）当阶码和尾数均用补码表示，且尾数采用规格化形式，给出两种情况下所能表示的最大正数和非零最小正数真值。

解：（1）阶码基值不论取何值，在浮点数中均为隐含表示，即：2 和 16 不出现在浮点格式中，仅为人为的约定。

（2）当基值不同时，对数的表示范围和精度都有影响。即：在浮点格式不变的情况下，基越大，可表示的浮点数范围越大，但精度越下降。（3）r=2 时，最大正数的浮点格式为：0，1111；0.111 111 111 1 其真值为：N+max=215×（1-2-10） 非零最小规格化正数浮点格式为：1，0000；0.100 000 000 0 其真值为：N+min=2-16×2-1=2-17 r=16 时，最大正数的浮点格式为：0，1111；0.1111 1111 11 其真值为：N+max=1615×（1-2-10） 非零最小规格化正数浮点格式为：1，0000；0.0001 0000 00 其真值为：N+min=16-16×16-1=16-17

14. 设浮点数字长为 32 位，欲表示±6 万间的十进制数，在保证数的最大精度条件下，除阶符、数符各取一位外，阶码和尾数各取几位？按这样分配，该浮点数溢出的条件是什么？

解：若要保证数的最大精度，应取阶的基=2。 若要表示±6 万间的十进制数，由于 32768（215）< 6 万 <65536（216），则：阶码除阶符外还应取 5 位（向上取 2 的幂）。 故：尾数位数=32-1-1-5=25 位 25（32） 该浮点数格式如下： 1 5 1 ≥ 按此格式，该浮点数上溢的条件为：阶码 25

15. 什么是机器零？若要求全 0 表示机器零，浮点数的阶码和尾数应采取什么机器数形式？

解：机器零指机器数所表示的零的形式，它与真值零的区别是：机器零在数轴上表示为"0"点及其附近的一段区域，即在计算机中小到机器数的精度达不到的数均视为"机器零"，而真零对应数轴上的一点（0 点）。若要求用"全 0"表示浮点机器零，则浮点数的阶码应用移码、尾数用补码 66 拼起来正好为一串 0 的形式）。

16. 设机器数字长为 16 位，写出下列各种情况下它能表示的数的范围。设机器数采用一位符号位，答案均用十进制表示。

（1）无符号数；

（2）原码表示的定点小数。

（3）补码表示的定点小数。

（4）补码表示的定点整数。

（5）原码表示的定点整数。

（6）浮点数的格式为：阶码 6 位（含 1 位阶符），尾数 10 位（含 1 位数符）。分别写出其正数和负数的表示范围。

（7）浮点数格式同（6），机器数采用补码规格化形式，分别写出其对应的正数和负数的真值范围。

解：（1）无符号整数：0 —— 2^{16} - 1，即：0—— 65535；

无符号小数：0 —— 1 - 2^{-16} ，即：0 —— 0.99998；

（2）原码定点小数：-1 + 2^{-15}——1 - 2^{-15} ，即：-0.99997 —— 0.99997

（3）补码定点小数：- 1——1 - 2^{-15} ，即：-1——0.99997 亚运会

（4）补码定点整数：-2^{15}——$2^{15}-1$，即：-32768——32767

（5）原码定点整数：$-2^{15}+1$——$2^{15}-1$，即：-32767——32767

（6）据题意画出该浮点数格式，当阶码和尾数均采用原码，非规格化数表示时：

最大负数= 1，11 111；1.000 000 001 ，即 $-2^{-9}\times2^{-31}$

最小负数= 0，11 111；1.111 111 111，即 $-(1-2^{-9})\times2^{31}$

则负数表示范围为：$-(1-2^{-9})\times2^{31}$ —— $-2^{-9}\times2^{-31}$

最大正数= 0，11 111；0.111 111 111，即 $(1-2^{-9})\times2^{31}$

最小正数= 1，11 111；0.000 000 001，即 $2^{-9}\times2^{-31}$

则正数表示范围为：$2^{-9}\times2^{-31}$ ——$(1-2^{-9})\times2^{31}$

（7）当机器数采用补码规格化形式时，若不考虑隐藏位，则

最大负数=1，00 000；1.011 111 111，即 $-2^{-1}\times2^{-32}$

最小负数=0，11 111；1.000 000 000，即 -1×2^{31}

则负数表示范围为：-1×2^{31} —— $-2^{-1}\times2^{-32}$

最大正数=0，11 111；0.111 111 111，即 $(1-2^{-9})\times2^{31}$

最小正数=1，00 000；0.100 000 000，即 $2^{-1}\times2^{-32}$

则正数表示范围为：$2^{-1}\times2^{-32}$ ——$(1-2^{-9})\times2^{31}$

17. 设机器数字长为 8 位（包括一位符号位），对下列各机器数进行算术左移一位、两位，算术右移一位、两位，讨论结果是否正确。

[x1]原=0.001 1010；[y1]补=0.101 0100；[z1]反=1.010 1111；

[x2]原=1.110 1000；[y2]补=1.110 1000；[z2]反=1.110 1000；

[x3]原=1.001 1001；[y3]补=1.001 1001；[z3]反=1.001 1001。

解：算术左移一位：

[x1]原=0.011 0100；正确

[x2]原=1.101 0000；溢出（丢 1）出错

[x3]原=1.011 0010；正确

[y1]补=0.010 1000；溢出（丢 1）出错

[y2]补=1.101 0000；正确

[y3]补=1.011 0010；溢出（丢 0）出错

[z1]反=1.101 1111；溢出（丢 0）出错

[z2]反=1.101 0001；正确

[z3]反=1.011 0011；溢出（丢 0）出错

算术左移两位：

[x1]原=0.110 1000；正确

[x2]原=1.010 0000；溢出（丢 11）出错

[x3]原=1.110 0100；正确

[y1]补=0.101 0000；溢出（丢 10）出错

[y2]补=1.010 0000；正确

[y3]补=1.110 0100；溢出（丢 00）出错

[z1]反=1.011 1111；溢出（丢 01）出错

[z2]反=1.010 0011；正确

[z3]反=1.110 0111；溢出（丢 00）出错

算术右移一位：

[x1]原=0.000 1101；正确

[x2]原=1.011 0100；正确

[x3]原=1.000 1100(1)；丢 1，产生误差

[y1]补=0.010 1010；正确

[y2]补=1.111 0100；正确

[y3]补=1.100 1100(1)；丢 1，产生误差

[z1]反=1.101 0111；正确

[z2]反=1.111 0100(0)；丢 0，产生误差

[z3]反=1.100 1100；正确

算术右移两位：

[x1]原=0.000 0110（10)；产生误差

[x2]原=1.001 1010；正确

[x3]原=1.000 0110（01)；产生误差

[y1]补=0.001 0101；正确

[y2]补=1.111 1010；正确

[y3]补=1.110 0110（01)；产生误差

[z1]反=1.110 1011；正确

[z2]反=1.111 1010（00)；产生误差

[z3]反=1.110 0110（01)；产生误差

18. 试比较逻辑移位和算术移位。

解：逻辑移位和算术移位的区别：　　逻辑移位是对逻辑数或无符号数进行的移位，其特点是不论左移还是右移，空出位均补 0，移位时不考虑符号位。　　算术移位是对带符号数进行的移位操作，其关键规则是移位时符号位保持不变，空出位的补入值与数的正负、移位方向、采用的码制等有关。补码或反码右移时具有符号延伸特性。左移时可能产生溢出错误，右移时可能丢失精度。

19. 设机器数字长为 8 位（含 1 位符号位)，用补码运算规则计算下列各题。

（1）A=9/64，　B=-13/32，求 A+B。

（2）A=19/32，B=-17/128，求 A-B。

（3）A=-3/16，B=9/32，求 A+B。

（4）A=-87，B=53，求 A-B。

（5）A=115，B=-24，求 A+B。

解：（1）A=9/64= 0.001 0010B, B= -13/32= -0.011 0100B

[A]补=0.001 0010, [B]补=1.100 1100

[A+B]补= 0.0010010 + 1.1001100 = 1.1011110 ——无溢出

A+B= -0.010 0010B = -17/64

（2）A=19/32= 0.100 1100B, B= -17/128= -0.001 0001B

[A]补=0.100 1100, [B]补=1.110 1111 , [-B]补=0.001 0001

[A-B]补= 0.1001100 + 0.0010001= 0.1011101 ——无溢出

A-B= 0.101 1101B = 93/128B

（3）A= -3/16= -0.001 1000B, B=9/32= 0.010 0100B

[A]补=1.110 1000, [B]补= 0.010 0100

[A+B]补= 1.1101000 + 0.0100100 = 0.0001100 —— 无溢出

A+B= 0.000 1100B = 3/32

（4） A= -87= -101 0111B, B=53=110 101B

[A]补=1 010 1001, [B]补=0 011 0101, [-B]补=1 100 1011

[A-B]补= 1 0101001 + 1 1001011 = 0 1110100 —— 溢出

（5）A=115= 111 0011B, B= -24= -11 000B

[A]补=0 1110011, [B]补=1，110 1000

[A+B]补= 0 1110011 + 1 1101000 = 0 1011011——无溢出

A+B= 101 1011B = 91

6.20 用原码一位乘、两位乘和补码一位乘（Booth 算法）、两位乘计算 x·y。

（1）x= 0.110 111，y= -0.101 110;

（2）x= -0.010 111，y= -0.010 101;

（3）x= 19， y= 35;

（4）x= 0.110 11， y= -0.111 01。

解：先将数据转换成所需的机器数，然后计算，最后结果转换成真值。

（1）[x]原=x=0.110111，[y]原=1.101110

x*=0.110111， y*=0.101110

x0=0，y0=1，z0=x0 ⊕ y0=0 ⊕ 1=1

x*×y*=0.100 111 100 010

[x×y]原=1.100 111 100 010

x·y= -0. 100 111 100 010

原码一位乘：

	部分积	乘数 y*	
	0．0 0 0 0 0 0	．1 0 1 1 1 0	—— +0
→1	0．0 0 0 0 0 0	0．1 0 1 1 1	—— +x*
+	0．1 1 0 1 1 1		
	0．1 1 0 1 1 1		
→1	0．0 1 1 0 1 1	1 0．1 0 1 1	—— +x*
+	0．1 1 0 1 1 1		
	1．0 1 0 0 1 0		
→1	0．1 0 1 0 0 1	0 1 0 ．1 0 1	—— +x*
+	0．1 1 0 1 1 1		
	1．1 0 0 0 0 0		
→1	0．1 1 0 0 0 0	0 0 1 0．1 0	—— +0
→1	0．0 1 1 0 0 0	0 0 0 1 0．1	—— x*
+	0．1 1 0 1 1 1		
	1．0 0 1 1 1 1		
→1	0．1 0 0 1 1 1	1 0 0 0 1 0	

2x*=01.101110，[-x*]补=[-x]补=1.001001

原码两位乘：

	部分积	乘数	Cj
	0 0 0．0 0 0 0 0 0	0 0．1 0 1 1 1 0	0
+	0 0 1．1 0 1 1 1 0	+2x*	
	0 0 1．1 0 1 1 1 0		0
→2	0 0 0．0 1 1 0 1 1	1 0 0 0．1 0 1 1	
+	1 1 1．0 0 1 0 0 1	+[-x*]补	
	1 1 1．1 0 0 1 0 0		1

→2　1 1 1．1 1 1　0 0 1　　0 0　1　0　0　0．1　0
　+　1 1 1．0 0 1　0 0 1　　+[-x*]补
　　1 1 1．0 0 0　0 1 0　　1
→2　1 1 1．1 1 0　0 0 0　　1 0　0　0　1　0　0　0．
　+　0 0 0．1 1 0　1 1 1　　+x*
　　0 0 0．1 0 0　1 1 1　　1 0　0　0　1　0　　0
结果同一位乘，x・y= -0. 100　111　100　010
[x]补=x=0.110111
[y]补=1.010010
[-x]补=1.001001
[2x]补=01.101110
[-2x]补=10.010010
[x×y]补=1.011　000　011　110 0
x・y= -0.100　111　100　010　0
补码一位乘、两位乘运算过程如下：
补码一位乘：部分积　　乘数[y]补　　yn+1
　　0 0．0 0 0　0 0 0　　1．0　1　0　　0　1　0　　0 —— +0
→1　0 0．0 0 0　0 0 0　　0　1．0　1　　0　0　1　　0
　+　1 1．0 0 1　0 0 1　　+[-x]补
　　1 1．0 0 1　0 0 1
→1　1 1．1 0 0　1 0 0　　1　0　1．0　　1　0　0　　1
　+　0 0．1 1 0　1 1 1　　+[x]补
　　0 0．0 1 1　0 1 1
→1　0 0．0 0 1　1 0 1　　1　1　0　1　．0　1　0　　0 —— +0
→1　0 0．0 0 0　1 1 0　　1　1　1　0　　1．0　1　　0
　+　1 1．0 0 1　0 0 1　　+[-x]补
　　1 1．0 0 1　1 1 1
→1　1 1．1 0 0　1 1 1　　1　1　1　1　　0　1．0　　1
　+　0 0．1 1 0　1 1 1　　+[x]补
　　0 0．0 1 1　1 1 0
→1　0 0．0 0 1　1 1 1　　0　1　1　1　　1　0　1　　．0
　+　1 1．0 0 1　0 0 1　　+[-x]补
　　1 1．0 1 1　0 0 0　　0　1　1　1　　1　0　0 —— 清0
（2） x= -0.010111，　　y= -0.010101
　　[x]原=1.010111，　[y]原=1.010101
　　x*=0. 010111 ，　y*=0. 010101
　　[-x*]补=1.101001，2x*=0.101110
　　[-2x*]补=1.010010
　　x0=1，y0=1，z0=x0 ⊕ y0=1 ⊕ 1=0
　　[x]补=1.101001，[y]补=1.101011
　　[-x]补=0.010111，[2x]补=1.010010
　　[-2x]补=0.101110
　　x*×y*=0.000　111　100　011
　　[x×y]原=0.000　111　100　011

[x×y]补=0.000 111 100 011 0

x·y= 0. 000 111 100 011

运算过程如下：

原码一位乘：

	部分积	乘数 y*	
	0．0 0 0 0 0 0 0	．0 1 0 1 0 1	——— +x*
+	0．0 1 0 1 1 1		
	0．0 1 0 1 1 1		
→1	0．0 0 1 0 1 1	1．0 1 0 1 0	——— +0
→1	0．0 0 0 1 0 1	1 1．0 1 0 1	——— +x*
+	0．0 1 0 1 1 1		
	0．0 1 1 1 0 0		
→1	0．0 0 1 1 1 0	0 1 1 ．0 1 0	——— +0
→1	0．0 0 0 1 1 1	0 0 1 1．0 1	——— +x*
+	0．0 1 0 1 1 1		
	0．0 1 1 1 1 0		
→1	0．0 0 1 1 1 1	0 0 0 1 1．0	——— +0
→1	0．0 0 0 1 1 1	1 0 0 0 1 1	

原码两位乘：

	部分积	乘数 y*	Cj
	0 0 0．0 0 0 0 0 0	0 0．0 1 0 1 0 1	0
+	0 0 0．0 1 0 1 1 1	+x*	
	0 0 0．0 1 0 1 1 1		0
→2	0 0 0．0 0 0 1 0 1	1 1 0 0．0 1 0 1	
+	0 0 0．0 1 0 1 1 1	+x*	
	0 0 0．0 1 1 1 0 0		0
→2	0 0 0．0 0 0 1 1 1	0 0 1 1 0 0．0 1	
+	0 0 0．0 1 0 1 1 1	+x*	
	0 0 0．0 1 1 1 1 0		0
→2	0 0 0．0 0 0 1 1 1	1 0 0 0 1 1 0 0．	
			+0

结果同一位乘， x·y= 0. 000 111 100 011

补码一位乘：

	部分积	乘数[y]补	yn+1
	0 0．0 0 0 0 0 0 0	1．1 0 1 0 1 1	0
+	0 0．0 1 0 1 1 1		+[-x]补
	0 0．0 1 0 1 1 1		
→1	0 0．0 0 1 0 1 1	1 1．1 0 1 0 1	——— +0
→1	0 0．0 0 0 1 0 1	1 1 1．1 0 1 0	1
+	1 1．1 0 1 0 0 1		+[x]补
	1 1．1 0 1 1 1 0		
→1	1 1．1 1 0 1 1 1	0 1 1 1 ．1 0 1	0
+	0 0．0 1 0 1 1 1		+[-x]补
	0 0．0 0 1 1 1 0		
→1	0 0．0 0 0 1 1 1	0 0 1 1 1．1 0	1

+ 1 1．1 0 1 0 0 1 +[x]补

1 1．1 1 0 0 0 0

→1 1 1．1 1 1 0 0 0 0 0 1 1 1．1 0

+ 0 0．0 1 0 1 1 1 +[-x]补

0 0．0 0 1 1 1 1

→1 0 0．0 0 0 1 1 1 1 0 0 0 1 1 1．1 —— +0

（3） x= 19， y= 35

x=（10 011）2，y=（100 011）2

x*= [x]原= [x]补= 0，010 011

y*= [y]原= [y]补= 0，100 011

[-x*]补= [-x]补= 1，101 101

2x*= [2x]补= 0，100 110

[-2x*]补= [-2x]补= 1，011 010

x0=0，y0=0，z0=x0 ⊕ y0=0 ⊕ 0=0

x·y= x*×y*= [x×y]原= [x×y]补

= 0，001 010 011 001

= （665）10

运算过程如下：

原码一位乘：

部分积 乘数 y*

0，0 0 0 0 0 0 1 0 0 0 1 1 —— +x*

+ 0，0 1 0 0 1 1

0，0 1 0 0 1 1

→1 0，0 0 1 0 0 1 1 1 0 0 0 1 —— +x*

+ 0，0 1 0 0 1 1

0，0 1 1 1 0 0

→1 0，0 0 1 1 1 0 0 1 1 0 0 0 —— +0

→1 0，0 0 0 1 1 1 0 0 1 1 0 0 —— +0

→1 0，0 0 0 0 1 1 1 0 0 1 1 0 —— +0

→1 0，0 0 0 0 0 1 1 1 0 0 1 1 —— +x*

+ 0，0 1 0 0 1 1

0，0 1 0 1 0 0

→1 0，0 0 1 0 1 0 0 1 1 0 0 1

原码两位乘：

部分积 乘数 y* Cj

0 0 0，0 0 0 0 0 0 0，1 0 0 0 1 1 0

+ 1 1 1，1 0 1 1 0 1 +[-x*]补

1 1 1，1 0 1 1 0 1 1

→2 1 1 1，1 1 1 0 1 1 0 1 0 0，1 0 0 0

+ 0 0 0，0 1 0 0 1 1 +x*

0 0 0，0 0 1 1 1 0 0

→2 0 0 0，0 0 0 0 1 1 1 0 0 0，1 0

+ 0 0 0，1 0 0 1 1 0 +2x*

0 0 0，1 0 1 0 0 1 0

→2　0 0 0，0 0 1　0 1 0　0 1　1 0　0 1　0 0，

+0

结果同一位乘， x • y= 0，001 010 011 001

补码一位乘：部分积　乘数[y]补　yn+1

0 0，0 0 0 0　0 0 0　0，1　0　0　0　1　1　0

+　1 1，1 0 1　1 0 1　+[-x]补

1 1，1 0 1　1 0 1

→1　1 1，1 1 0　1 1 0　1　0，1　0　0　0　1　1 —— +0

→1　1 1，1 1 1　0 1 1　0　1　0，1　0　0　0　1

+　0 0，0 1 0　0 1 1　+[x]补

0 0，0 0 1　1 1 0

→1　0 0，0 0 0　1 1 1　0　0　1　0，1　0　0　0 —— +0

→1　0 0，0 0 0　0 1 1　1　0　0　1　0，1　0　0 —— +0

→1　0 0，0 0 0　0 0 1　1　1　0　0　1　0，1　0

+　1 1，1 0 1　1 0 1　+[-x]补

1 1，1 0 1　1 1 0

→1　1 1，1 1 0　1 1 1　0　1　1　0　0　1　0，　1

+　0 0，0 1 0　0 1 1　+[x]补

0 0，0 0 1　0 1 0　0　1　1　0　0　1　0

注：整数乘此位要省。

（4） x= 0. 110　11，　y= -0.111　01

x*= [x]原= [x]补= 0. 110　11

[y]原=1.111　01，y*=0. 111　01

[y]补=1.000　11

[-x*]补= [-x]补= 1.001　01

2x*= [2x]补= 01.101　10

[-2x*]补= [-2x]补= 10.010　10

x0=0，y0=1，z0=x0 ⊕ y0=0 ⊕ 1=1

x*×y*=0.110　000　111　1

[x×y]原=1.110　000　111　1

[x×y]补=1.001　111　000　10

x • y= -0. 110　000　111　1

运算过程如下：

原码一位乘：部分积　乘数 y*

0 . 0 0 0　0 0　. 1　1　1　0　1 —— +x*

+　0 . 1 1 0　1 1

0 . 1 1 0　1 1

→1　0 . 0 1 1　0 1　1 . 1　1　1　0 —— +0

→1　0 . 0 0 1　1 0　1　1 . 1　1　1 —— +x*

+　0 . 1 1 0　1 1

1 . 0 0 0　0 1

→1　0 . 1 0 0　0 0　1　1　1　. 1　1 —— +x*

+　0 . 1 1 0　1 1

1 . 0 1 0　1 1

→1　0．1 0 1　0 1　　1　1　1　　1．1 —— +x*
　+　0．1 1 0　1 1
　　1．1 0 0　0 0
→1　0．1 1 0　0 0　　0　1　1　　1　1

原码两位乘：

部分积	乘数 y*	Cj
0 0 0．0 0 0　0 0	0．1 1 1　0 1	0
+ 0 0 0．1 1 0　1 1	+x*	
0 0 0．1 1 0　1 1		0
→2 0 0 0．0 0 1　1 0	1 1 0．1　1 1	
+ 1 1 1．0 0 1　0 1	+[-x*]补	
1 1 1．0 1 0　1 1		1
→2 1 1 1．1 1 0　1 0	1 1 1 1 ．0 1	
+ 0 0 1．1 0 1　1 0	+2x*	
0 0 1．1 0 0　0 0		0
→1 0 0 0．1 1 0　0 0	0 1 1 1　1 0．	
		+0

结果同一位乘， x·y= -0. 110　000　111　1

补码一位乘：

部分积	乘数[y]补	yn+1
0 0．0 0 0　0 0	1．0 0 0　1	1　0
+ 1 1．0 0 1　0 1	+[-x]补	
1 1．0 0 1　0 1		
→1 1 1．1 0 0　1 0	1 1．0 0　0	1　1 —— +0
→1 1 1．1 1 0　0 1	0 1 1．0　0	0　1
+ 0 0．1 1 0　1 1	+[x]补	
0 0．1 0 1　0 0		
→1 0 0．0 1 0　1 0	0 0 1 1 ．0	0　0 —— +0
→1 0 0．0 0 1　0 1	0 0 0 1　1．0	0　0 —— +0
→1 0 0．0 0 0　1 0	1 0 0 0　1	1．　0
+ 1 1．0 0 1　0 1	+[-x]补	
1 1．0 0 1　1 1	1 0 0 0　1	0 —— 清 0

6.21　用原码加减交替法和补码加减交替法计算 x÷y。

（1）x=0.100111，y=0.101011；

（2）x=-0.10101，　y=0.11011；

（3）x=0.10100，　y= -0.10001；

（4）x=13/32，　　y= -27/32。

解：

（1）x*=[x]原=[x]补=x= 0.100　111

y*=[y]原=[y]补=y= 0.101　011

[-y*]补=[-y]补=1.010　101

q0=x0⊕y0=0 ⊕0=0

x÷y=x*÷y*=[x÷y]原=0.111　010

r*=0.000 010×2-6=0.000 000 000 010

计算过程如下：

原码加减交替除法：

	被除数（余数）	商
	0．1 0 0　1 1 1	0．0 0 0　0 0 0
+	1．0 1 0　1 0 1	试减，+[-y*]补
	1．1 1 1　1 0 0	
1←	1．1 1 1　0 0 0	0．
+	0．1 0 1　0 1 1	r<0，+y*
	0．1 0 0　0 1 1	
1←	1．0 0 0　1 1 0	0.1
+	1．0 1 0　1 0 1	r>0，+[-y*]补
	0．0 1 1　0 1 1	
1←	0．1 1 0　1 1 0	0.1 1
+	1．0 1 0　1 0 1	r>0，+[-y*]补
	0．0 0 1　0 1 1	
	被除数（余数）	商
1←	0．0 1 0　1 1 0	0．1 1 1
+	1．0 1 0　1 0 1	r>0，+[-y*]补
	1．1 0 1　0 1 1	
1←	1．0 1 0　1 1 0	0.1　1 1 0
+	0．1 0 1　0 1 1	r<0，+y*
	0．0 0 0　0 0 1	
1←	0．0 0 0　0 1 0	0.1 1　1 0 1
+	1．0 1 0　1 0 1	r>0，+[-y*]补
	1．0 1 0　1 1 1	1← 0.1 1 1　0 1 0
+	0．1 0 1　0 1 1	r<0，+y*（恢复余数）
	0．0 0 0　0 1 0	

补码加减交替除法：

	被除数（余数）	商
	0 0．1 0 0　1 1 1	0．0 0 0　0 0 0
+	1 1．0 1 0　1 0 1	试减，x、y 同号，+[-y]补
	1 1．1 1 1　1 0 0	
1←	1 1．1 1 1　0 0 0	0．
+	0 0．1 0 1　0 1 1	r、y 异号，+[y]补
	0 0．1 0 0　0 1 1	
1←	0 1．0 0 0　1 1 0	0.1
+	1 1．0 1 0　1 0 1	r、y 同号，+[-y]补
	0 0．0 1 1　0 1 1	
1←	0 0．1 1 0　1 1 0	0.1 1
+	1 1．0 1 0　1 0 1	r、y 同号，+[-y]补
	0 0．0 0 1　0 1 1	
被除数（余数）		商
1←	0 0．0 1 0　1 1 0	0．1 1 1

```
     +   1 1 . 0 1 0   1 0 1              r、y 同号，  +[-y]补
         1 1 . 1 0 1   0 1 1
1←     1 1 . 0 1 0   1 1 0                0.1   1 1 0
     +   0 0 . 1 0 1   0 1 1              r、y 异号，+[y]补
         0 0 . 0 0 0   0 0 1
1←     0 0 . 0 0 0   0 1 0              0.1 1   1 0 1
     +   1 1 . 0 1 0   1 0 1              r、y 同号，  +[-y]补
         1 1 . 0 1 0   1 1 1     1← 0.1 1 1   0 1 1 —— 恒置 1
     +   0 0 . 1 0 1   0 1 1              r、x 异号，(恢复余数)
         0 0 . 0 0 0   0 1 0         且 r、y 异号，  +[y]补
```

注：恒置 1 引入误差。 x÷y=[x÷y]补= 0.111 011

[r]补=0.000 010，r=r*=0.000 000 000 010

（2）x= -0.101 01，y=0.110 11

[x]原=1.101 01

x*= 0.101 01

y* = [y]原 = [y]补= y = 0.110 11

[-y*]补= [-y]补= 1.001 01

[x]补= 1.010 11

q0 = x0 ⊕ y0 = 1 ⊕ 0 = 1

x*÷y*= 0.110 00

[x÷y]原=1.110 00

x÷y = -0.110 00

r*=0.110 00×2-5

=0.000 001 100 0

计算过程如下：

原码加减交替除法：

```
     被除数（余数）              商
     0 . 1 0 1   0 1        0 . 0 0 0   0 0
+    1 . 0 0 1   0 1            试减，+[-y*]补
     1 . 1 1 0   1 0
1←   1 . 1 0 1   0 0                          0 .
+    0 . 1 1 0   1 1            r<0，+y*
     0 . 0 1 1   1 1
1←   0 . 1 1 1   1 0                        0.1
+    1 . 0 0 1   0 1            r>0，  +[-y*]补
     0 . 0 0 0   1 1
1←   0 . 0 0 1   1 0                      0.1 1
+    1 . 0 0 1   0 1            r>0，  +[-y*]补
     1 . 0 1 0   1 1
被除数（余数）              商
1←   0 . 1 0 1   1 0                  0 . 1 1 0
+    0 . 1 1 0   1 1            r<0，  +y*
     1 . 1 0 0   0 1
1←   1 . 0 0 0   1 0            0.1   1 0 0
```

+ 0．1 1 0 1 1 r<0，+y*

1．1 1 1 0 1 1← 0.1 1 0 0 0

+ 0．1 1 0 1 1 r<0， +y*（恢复余数）

0．1 1 0 0 0

补码加减交替除法：

被除数（余数） 商

1 1．0 1 0 1 1 0．0 0 0 0 0

+ 0 0．1 1 0 1 1 试减，x、y 异号，+[y]补

0 0．0 0 1 1 0

1← 0 0．0 1 1 0 0 1．

+ 1 1．0 0 1 0 1 r、y 同号，+[-y]补

1 1．1 0 0 0 1

1← 1 1．0 0 0 1 0 1.0

+ 0 0．1 1 0 1 1 r、y 异号， +[y]补

1 1．1 1 1 0 1

1← 1 1．1 1 0 1 0 1.0 0

+ 0 0．1 1 0 1 1 r、y 异号， +[y]补

0 0．1 0 1 0 1

被除数（余数） 商

1← 0 1．0 1 0 1 0 1．0 0 1

+ 1 1．0 0 1 0 1 r、y 同号， +[-y]补

0 0．0 1 1 1 1

1← 0 0．1 1 1 1 0 1.0 0 1 1

+ 1 1．0 0 1 0 1 r、y 同号，+[-y]补

0 0．0 0 0 1 1 1← 1.0 0 1 1 1 —— 恒置 1

+ 1 1．0 0 1 0 1 r、x 异号，（恢复余数）

1 1．0 1 0 0 0 且 r、y 同号，+[-y]补

注：恒置 1 引入误差。

[r]补=1.010 00， r= -0.000 001 100 0

[x÷y]补=1.001 11，x÷y= -0.110 01

（3）x= 0.101 00，y= -0.100 01

x*= [x]原= [x]补= x=0.101 00

[y]原 = 1.100 01

y* = 0.100 01

[-y*]补=1.011 11

[y]补= 1.011 11

[-y]补= 0.100 01

q0 = x0 ⊕ y0 = 0 ⊕ 1 = 1

x*÷y*= 1.001 01 —— 溢出

[x÷y]原：无定义

x÷y = -1.001 01

r*=0.010 11×2-5

=0.000 000 101 1

计算过程如下：

原码加减交替除法：

	被除数（余数）	商
	0．101　00	0．000　00
+	1．011　11	试减，+[-y*]补
	0．000　11	
1←	0．001　10	1．
+	1．011　11	r>0，+[-y*]补
	1．101　01	
1←	1．010　10	1.0
+	0．100　01	r<0，+y*
	1．110　11	
1←	1．101　10	1.00
+	0．100　01	r<0，+y*
	0．001　11	
	被除数（余数）	商
1←	0．011　10	1．001
+	1．011　11	r>0，+[-y*]补
	1．111　01	
1←	1．110　10	1.0　010
+	0．100　01	r<0，+y*
	0．010　11	1← 1.00　1011
		r>0，结束

注：当 x*>y*时产生溢出，这种情况在第一步运算后判断 r 的正负时就可发现。此时数值位占领小数点左边的 1 位，原码无定义，但算法本身仍可正常运行。

补码加减交替除法：

	被除数（余数）	商
	00．101　00	0．000　00
+	11．011　11	试减，x、y 异号，+[y]补
	00．000　11	
1←	00．001　10	0．
+	11．011　11	r、y 异号，+[y]补
	11．101　01	
1←	11．010　10	0.1
+	00．100　01	r、y 同号，+[-y]补
	11．110　11	
1←	11．101　10	0.11
+	00．100　01	r、y 同号，+[-y]补
	00．001　11	
	被除数（余数）	商
1←	00．011　10	0．110
+	11．011　11	r、y 异号，+[y]补
	11．111　01	
1←	11．110　10	0.1　101
+	00．100　01	r、y 同号，+[-y]补

0 0 . 0 1 0　1 1　　1← 0.1　1 0 1 1 —— 恒置1

r、x同号，结束

[r]补=0.010　11，r=r*=0.000　000　101　1

真符位的产生：qf = x0 ⊕ y0 = 0 ⊕ 1 = 1

[x÷y]补=10.110　11，x÷y= -1.001　01

判溢出：qf　⊕ q0 = 1 ⊕ 0 = 1，溢出

注：由于本题中x*>y*，有溢出。除法运算时一般在运算前判断是否x*>y*，如果该条件成立则停止运算，转溢出处理。但此算法本身在溢出情况下仍可正常运行，此时数值位占领小数点左边的1位，商需设双符号位（变形补码），以判溢出。采用这种方法时运算前可不判溢出，直接进行运算，运算完后再判溢出。

（4）x=13/32=（0.011　01）2

y= -27/32=（-0.110　11）2

x*= [x]原= [x]补= x=0. 011　01

[y]原 = 1.110　11

y* = 0.110　11

[-y*]补=1.001　01

[y]补= 1.001　01

[-y]补= 0.110　11

q0 = x0 ⊕ y0 = 0 ⊕ 1 = 1

x*÷y*= 0.011　11

[x÷y]原=1.011　11

x÷y =（-0.011　11）2 = -15/32

r*=0.010　11×2-5

=0.000　000　101　1

原码加减交替除法：

	被除数（余数）	商
	0 . 0 1 1　0 1	0 . 0 0 0　0 0
+	1 . 0 0 1　0 1	试减，+[-y*]补
	1 . 1 0 0　1 0	
1←	1 . 0 0 1　0 0	0 .
+	0 . 1 1 0　1 1	r<0，+y*
	1 . 1 1 1　1 1	
1←	1 . 1 1 1　1 0	0.0
+	0 . 1 1 0　1 1	r<0，+y*
	0 . 1 1 0　0 1	
1←	1 . 1 0 0　1 0	0.0 1
+	1 . 0 0 1　0 1	r>0，+[-y*]补
	0 . 1 0 1　1 1	

被除数（余数）	商
1← 1 . 0 1 1　1 0	0 . 0 1 1
+ 1 . 0 0 1　0 1	r>0，+[-y*]补
0 . 1 0 0　1 1	
1← 1 . 0 0 1　1 0	0.0　1 1 1
+ 1 . 0 0 1　0 1	r>0，+[-y*]补

0．0 1 0　1 1　　1← 0.0 1　1 1 1

r>0，结束

补码加减交替除法：

	被除数（余数）	商
	0 0．0 1 1　0 1	0．0 0 0　0 0
+	1 1．0 0 1　0 1	试减，x、y 异号，+[y]补
	1 1．1 0 0　1 0	
1←	1 1．0 0 1　0 0	1．
+	0 0．1 1 0　1 1	r、y 同号，+[-y]补
	1 1．1 1 1　1 1	
1←	1 1．1 1 1　1 0	1.1
+	0 0．1 1 0　1 1	r、y 同号，+[-y]补
	0 0．1 1 0　0 1	
1←	0 1．1 0 0　1 0	1.1 0
+	1 1．0 0 1　0 1	r、y 异号，+[y]补
	0 0．1 0 1　1 1	

	被除数（余数）	商
1←	0 1．0 1 1　1 0	1．1 0 0
+	1 1．0 0 1　0 1	r、y 异号，+[y]补
	0 0．1 0 0　1 1	
1←	0 1．0 0 1　1 0	1.1　0 0 0
+	1 1．0 0 1　0 1	r、y 异号，+[y]补
	0 0．0 1 0　1 1	1← 1.1　0 0 0 1 —— 恒置 1
		r、x 同号，结束

[r]补=0.010　11，r=r*=0.000　000　101　1

[x÷y]补=1.100　01，x÷y=（-0.011　11）2 = -15/32

22. 设机器字长为 16 位（含 1 位符号位），若一次移位需 1μs，一次加法需 1 μs，试问原码一位乘、补码一位乘、原码加减交替除法和补码加减交替除法各最多需多少时间？　　解：原码一位乘最多需时　　1μs×15（加）+ 1μs×15（移位）=30μs　　补码一位乘最多需时　　1μs×16+1μs×15 = 31μs　　原码加减交替除最多需时　　1μs×（16+1）+1μs×15=32μs（或 33μs）　　补码加减交替除最多需时　　1μs×（16+1）+1μs×15=32μs (或 33μs)　　(包括最后恢复余数！)

25. 对于尾数为 40 位的浮点数（不包括符号位在内），若采用不同的机器数表示，试问当尾数左规或右规时，最多移位次数各为多少？　　解：对于尾数为 40 位的浮点数，若采用原码表示，当尾数左规时，最多移位 39 次；反码表示时情况同原码；若采用补码表示，当尾数左规时，正数最多移位 39 次，同原码；负数最多移位 40 次。当尾数右规时，不论采用何种码制，均只需右移 1 次。

26. 按机器补码浮点运算步骤，计算[x±y]补.

（1）$x=2^{-011}\times$ 0.101 100，$y=2^{-010}\times$（-0.011 100）；

（2）$x=2^{-011}\times$（-0.100 010），$y=2^{-010}\times$（-0.011 111）；

（3）$x=2^{101}\times$（-0.100 101），$y=2^{100}\times$（-0.001 111）。

解：先将 x、y 转换成机器数形式：

(1) $x=2^{-011}\times$ 0.101 100，$y=2^{-010}\times$（-0.011 100）

[x]补=1，101；0.101 100, [y]补=1，110；1.100 100

[Ex]补=1,101, [y]补=1,110, [Mx]补=0.101 100, [My]补=1.100 100

1）对阶：

[ΔE]补=[Ex]补+[-Ey]补 = 11,101+ 00,010=11,111 < 0，

应 Ex 向 Ey 对齐，则：[Ex]补+1=11，101+00，001=11，110 = [Ey]补

[x]补=1，110；0.010 110

2）尾数运算：

[Mx]补+[My]补= 0.010 110 + 11.100 100=11.111010

[Mx]补+[-My]补=0.010 110 + 00.011100= 00.110 010

3）结果规格化：

[x+y]补=11，110；11.111 010 = 11，011；11.010 000 （尾数左规 3 次，阶码减 3）

[x-y]补=11，110；00.110 010, 已是规格化数。

4）舍入：无

5）溢出：无

则：$x+y=2^{-101}\times$（-0.110 000）

$x-y=2^{-010}\times$0.110 010

(2) $x=2^{-011}\times$（-0.100010），$y=2^{-010}\times$（-0.011111）

[x]补=1，101；1.011 110, [y]补=1，110；1.100 001

1） 对阶：过程同(1)的 1），则

[x]补=1，110；1.101 111

2）尾数运算：

[Mx]补+[My]补= 11.101111 + 11. 100001 = 11.010000

[Mx]补+[-My]补= 11.101111 + 00.011111 = 00.001110

3）结果规格化：

[x+y]补=11，110；11.010 000,已是规格化数

[x-y]补=11，110；00.001 110 =11，100；00.111000 （尾数左规 2 次，阶码减 2）

4）舍入：无

5）溢出：无

则：$x+y=2^{-010}\times$（-0.110 000）

$x-y=2^{-100}\times$0.111 000

(3) $x=2^{101}\times$（-0.100 101），$y=2^{100}\times$（-0.001 111）

[x]补=0，101；1.011 011, [y]补=0，100；1.110 001

1）对阶：

[ΔE]补=00，101+11，100=00，001 >0，应 Ey 向 Ex 对齐，则：

[Ey]补+1=00，100+00，001=00，101=[Ex]补

[y]补=0，101；1.111 000（1）

2）尾数运算：

[Mx]补+[My]补= 11.011011+ 11.111000（1）= 11.010011（1）

[Mx]补+[-My]补= 11.011011+ 00.000111（1）= 11.100010（1）

2） 结果规格化：

[x+y]补=00，101；11.010 011（1），已是规格化数

[x-y]补=00，101；11.100 010（1）=00，100；11.000 101 （尾数左规 1 次，阶码减 1）

4）舍入：

[x+y]补=00，101；11.010 011（舍）

[x-y]补 不变

5）溢出：无

则：x+y=2^{101}×（-0.101 101）

x-y =2^{100}×（-0.111 011）

6.27、假设阶码取 3 位，尾数取 6 位（均不包括符号位），计算下列各题。

（1）[25×(11/16)]+[24×(-9/16)]

（2）[2-3×(13/16)]-[2-4×(-5/8)]

（3）[23×(13/16)]×[24×(-9/16)]

（4）[26×(-11/16)]÷[23×(-15/16)]

（5）[23×(-1)] ×[2-2×57/64]

（6）[2-6×(-1)]÷[27×(-1/2)]

（7）3.3125+6.125

（8）14.75-2.4375

解：设机器数采用阶补尾补形式：

（1）x= 25×(11/16)= 2101×0.101100

y= 24×(-9/16)=2100×(-0.100100)则： [x]阶补尾补=00，101；00.101100

[y]阶补尾补=00，100；11.011100

1）对阶：

[ΔE]补=[Ex]补+[-Ey]补

=00，101+11，100=00，001

[ΔE]补>0，应 Ey 向 Ex 对齐，则：

[Ey]补+1=00，100+00，001=00，101

[ΔE]补+[-1]补=00，001+11，111=0

至此， Ey=Ex，对毕。

[y]补=00，101；11.101110

2）尾数运算：

[Mx]补+[My]补= 0 0 . 1 0 1 1 0 0

+ 1 1 . 1 0 1 1 1 0

0 0 . 0 1 1 0 1 0

3）结果规格化：左规 1 位

[x+y]补=00，101；00.011 010

=00，100；00.110 100

4）舍入：不需舍入。

5）溢出：无

则：x+y=2100×（0.110 100）

=24×（13/16）

（2）[2-3×(13/16)]-[2-4×(-5/8)]

x= 2-3×(13/16)= 2-011×0.110 100

y= 2-4×(-5/8)=2-100×(-0.101000)

[x]阶补尾补=11，101；00.110100

[y]阶补尾补=11，100；11.011000

1）对阶：

[ΔE]补=[Ex]补+[-Ey]补

=11，101+00，100=00，001

[ΔE]补>0，应 Ey 向 Ex 对齐，则：

[Ey]补+1=11，100+00，001=11，101

[ΔE]补+[-1]补=00，001+11，111=0

至此， Ey=Ex，对毕。

[y]补=11，101；11.101100

2）尾数运算：

[Mx]补+[-My]补= 0 0 . 1 1 0 1 0 0

+ 0 0 . 0 1 0 1 0 0

0 1 . 0 0 1 0 0 0

3）结果规格化：右规

[x-y]补=11，101；01.001 000

=11，110；00.100 100

4）舍入：不需舍入。

（3）[23×(13/16)]×[24×(-9/16)]

x= 23×(13/16)=2011×（0.110 100）

y= 24×(-9/16)=2100×（-0.100 100）

[x]阶补尾补=00，011；0.110 100

[y]阶补尾补=00，100；1.011 100

1）阶码相加：

[Ex]补+[Ey]补=00，011+ 00，100

=00，111（无溢出）

2）尾数相乘：

补码两位乘比较法，见下页。

[Mx × My]补=11.100 010（110 000 00）

3）结果规格化：左规 1 位。

[x×y]补=0，111；1.100 010(110 000 00)

=0，110；1.000 101(100 000 0)

2）尾数相乘： （补码两位乘比较法）

	部分积	乘数	yn+1
	0 0 0 . 0 0 0 0 0 0	1 1 . 0 1 1 1 0 0	0
+	0 0 0 . 0 0 0 0 0 0		+[-0]补
	0 0 0 . 0 0 0 0 0 0		
→2	0 0 0 . 0 0 0 0 0 0	0 0 1 1 . 0 1 1 1	0
+	1 1 1 . 0 0 1 1 0 0		+[-x]补
	1 1 1 . 0 0 1 1 0 0		
→2	1 1 1 . 1 1 0 0 1 1	0 0 0 0 1 1 . 0 1 1	1
+	0 0 1 . 1 0 1 0 0 0		+[2x]补

0 0 1 . 0 1 1　0 1 1

→2　0 0 0 . 0 1 0　1 1 0　　1 1　0 0 0　　0　1 1 . 0

+　1 1 1 . 0 0 1　1 0 0　　+[-x]补

1 1 1 . 1 0 0　0 1 0　　1 1　0 0 0　　0　0 0(清 0)

4）舍入：设采用 0 舍 1 入法，应舍：

[x×y]阶补尾补=0，110；1.000 101

5）溢出：无

x×y=2110×（-0.111　011）

= 26×（-59/64）

（4）[26×(-11/16)]÷[23×(-15/16)]

x= 26×(-11/16)=2110×（-0.101　100）

y= 23×(-15/16)=2011×（-0.111　100）

[x]阶补尾补=00，110；1.010 100

[y]阶补尾补=00，011；1.000　100

1）阶码相减：

[Ex]补+[-Ey]补=00，110+ 11，101=00，011（无溢出）

2）尾数相除：（补码加减交替除法）

被除数（余数）　　商

1 1 . 0 1 0　1 0 0　　0 . 0 0 0　0 0 0　　试减，

+　0 0 . 1 1 1　1 0 0　　Mx、My 同号，+[-My]补

0 0 . 0 1 0　0 0 0

1←　0 0 . 1 0 0　0 0 0　　0 .

+　1 1 . 0 0 0　1 0 0　　r、My 异号，+[My]补

1 1 . 1 0 0　1 0 0

1←　1 1 . 0 0 1　0 0 0　　0.1

+　0 0 . 1 1 1　1 0 0　　r、My 同号，+[-My]补

0 0 . 0 0 0　1 0 0

1←　0 0 . 0 0 1　0 0 0　　0.1 0

+　1 1 . 0 0 0　1 0 0　　r、My 异号，+[My]补

1 1 . 0 0 1　1 0 0

被除数（余数）　　商

1←　1 0 . 0 1 1　0 0 0　　0 . 1 0 1

+　0 0 . 1 1 1　1 0 0　　r、My 同号，+[-My]补

1 1 . 0 1 0　1 0 0

1←　1 0 . 1 0 1　0 0 0　　0.1　0 1 1

+　0 0 . 1 1 1　1 0 0　　r、My 同号，+[-My]补

1 1 . 1 0 0　1 0 0

1←　1 1 . 0 0 1　0 0 0　　0.1 0　1 1 1

+　0 0 . 1 1 1　1 0 0　　r、My 异号，+[-My]补

0 0 . 0 0 0　1 0 0　　1← 0.1 0 1　1 1 1 —— 恒置 1

+　1 1 . 0 0 0　1 0 0　　r、Mx 异号，（恢复余数）

1 1 . 0 0 1　0 0 0　　且 r、My 异号，+[My]补

[Mx÷My]补= 0.101　111，[r]补=1.001　000

r= -0 .111 000 × 2-6 =-0.000 000 111 000

29. 设浮点数阶码取 3 位，尾数取 6 位（均不包括符号位），要求阶码用移码运算，尾数用补码运算，计算 x·y，且结果保留 1 倍字长。（1）x=2-100× 0.101101，y=2-011×（-0.110101）；（2）x=2-011×（-0.100111），y=2101×（-0.101011）。 解：先将 x、y 转换成机器数形式：（1）[x]阶移尾补=0，100；0.101 101 [y]阶移尾补=0，101；1.001 011 1）阶码相加： [Ex]移+[Ey]补=00，100+11，101 =00，001（无溢出）

2）尾数相乘：（补码两位乘比较法） 部分积 乘数 yn+1 0 0 0 . 0 0 0 0 0 0 1 1 . 0 0 1 0 1 1 0 + 1 1 1 . 0 1 0 0 1 1 +[-x]补 1 1 1 . 0 1 0 0 1 2 1 1 1 . 1 1 0 1 0 0 1 1 1 1 . 0 0 1 0 1 + 1 1 1 .→1 0 1 0 0 1 1 +[-x]补 1 1 1 . 0 0 0 1 1 1 2 1 1 1 . 1 1 0 0 1 1 1 1 1 1 . 0→ 0 1 + 0 0 0 . 1 0 1 1 0 1 +[x]补 0 0 0 2 0 0 0 . 0 0 0 1 1 1 →. 0 1 1 1 1 0 1 0 1 1 1 1 1 . 0 + 1 1 1 . 0 1 0 0 1 1 +[-x]补 1 1 1 . 0 1 1 0 1 0 1 0 1 1 1 1 0 0(清 0)

[Mx × My]补=1.011 010（101 111 00） 3）结果规格化：已是规格化数。 4）舍入：设采用 0 舍 1 入法，应入： [x×y]阶移尾补=0，001；1.011 011 5）溢出：无 x×y=2-111×（-0.100 101） 方案二：采用阶补尾原格式计算： [x]阶补尾原=1，100；0.101 101 [y]阶补尾原=1，101；1.110 101 1）阶码相加： [Ex]补+[Ey]补=11，100+11，101 =11，001（无溢出）

原码一位乘： 部分积 乘数 y* 0 . 0 0 0 0 0 0 . 1 1→ 1 0 1 0 1 —— +x* + 0 . 1 0 1 1 0 1 0 . 1 0 1 1 0 1 0 . 0 0 1 0 1 1 → 0 . 0 1 0 1 1 0 1 . 1 1 0 1 0 —— +0 0 1 . 1 1 0 1 —— +x* + 0 . 1 0 1 1 0 1 0 . 1 1 1 0 0 1 0 . 0 0 1 1→1 0 . 0 1 1 1 0 0 0 0 1 . 1 1 0 —— +0 →0 1 0 0 0 0 1 . 1 1 —— +x* + 0 . 1 0 1 1 0 1 0 . 1 1 1 1 0 . 0 1 1 1 0 1 1 0 0 0 1 . 1 —— +x* + 0 . 1→ 0 1 1 1 0 . 1 0 0 1 0 1 0 1 0 0→0 1 1 0 1 1 . 0 0 1 0 1 0 0 1

[Mx × My]原=1.100 101（010 001） 3）结果规格化：已是规格化数。 4）舍入：设采用 0 舍 1 入法，应舍： [x×y]阶补尾原=1，001；1.100 101 5）溢出：无 x×y=2-111×（-0.100 101） （2）x=2-011×（-0.100 111） y=2101×（-0.101 011） [x]阶移尾补=0，101；1.011 001 [y]阶移尾补=1，101；1.010 101 1）阶码相加： [Ex]移+[Ey]补=00，101+00，101 =01，010（无溢出）

2）尾数相乘：（补码两位乘比较法） 部分积 乘数 yn+1 0 0 0 . 0 0 0 0 0 0 1 1 . 0 1 0 1 0 1 0 + 1 1 1 . 0 1 1 0 0 1 +[x]补 1 1 1 . 0 1 1 2 1 1 1 . 1 1 0 1 1 0 0 1 1 1 . 0 1 1 0 0 1 1→ 0 0 1 1 1 . 0 1 1 0 0 1 +[x]补 1 1 1 . 0 0 1 1 1 1 2 1 1 1 . 1 1 0 0 1 1 1 1 0 1 1 → 1 . 0 1 0 + 1 1 1 . 0 1

1 0 0 1 +[x]补 2 1 1 1 . 1 1 0 0 1→1 1 1 . 0 0 1 1 0 0 1 0 0 1 1 0 1 1 . 0 + 0 0 0 . 1 0 0 1 1 1 +[-x]补 0 0 0 . 0 1 1 0 1 0 0 0 1 1 0 1 0 0(清 0) [Mx × My]补=0.011 010 ◎001 101 00）

3）结果规格化： [x×y]阶移尾补= = 1，010；0.011 010（001 101 00） = 1，001；0.110 100（011 010 0） 4）舍入：设采用 0 舍 1 入法，应舍： [x×y]阶移尾补=1，001；0. 110 100 5）溢出：无 x×y=2001×0.110 100 方案二：采用阶补尾原格式计算： [x]阶补尾原=1，101；1.100 111 [y]阶补尾原=0，101；1.101 011 1）阶码相加： [Ex]补+[Ey]补=11，101+ 00，101 =00，010（无溢出）

原码一位乘： 部分积 乘数 y* 0 . 0 0 0 0 0 0 0 . 1 0 1 0 1 1 0 . 0 1 0 0 1 1 →—— +x* + 0 . 1 0 0 1 1 1 0 . 1 0 0 1 1 1 1 . 1 0 1 0 1 —— +x* + 0 . 1 0 0 1 1 1 0 . 1 1 1 0 1 1 0 . 0 0 1 1→1 0 . 0 1 1 1 0 1 0 1 0 1 . 1 0 1 0 —— +0 →0 1 0 1 0 1 . 1 0 1 —— +x* + 0 . 1 0 0 1 1 1 0 . 1 1 0 1 0 . 0 0→1 0 . 0 1 1 0 1 0 1 1 0 1 . 1 0 —— +0 → 1 0 1 1 1 0 1 0 1 1 0 1 . 1 —— +x* + 0 . 1 0 0 1 1 1 0 . 1 0 . 0 1 1 0 1 0 0 0 1 1 0 1→1 1 0 1 0 0

[Mx × My]原=0.011 010（001 101） 3）结果规格化： [x×y]阶补尾原= = 0，010；0.011 010（001 101） = 0，001；0.110 100（011 01） 4）舍入：设采用 0 舍 1 入法，应舍： [x×y]阶补尾原=0，001；0. 110 100 5）溢出：无 x×y=2001×0.110 100

30. 机器数格式同上题，要求阶码用移码运算，尾数用补码运算，计算 x÷y。 （1）x=2101× 0.100111， y=2011×（-0.101011）； （2）x=2110×（-0.101101）， y=2011×（-0.111100）。

解：先将 x、y 转换成机器数形式： （1）[x]阶移尾补=1，101；0.100 111 [y]阶移尾补=1，011；1.010 101 1）阶码相减： [Ex]移+[-Ey]补=01，101+11，101 =01，010（无溢出）

2）尾数相除： （补码加减交替除法） 被除数（余数） 商 0 0 . 1 0 0 1 1 1 0 . 0 0 0 0 0 0 0 试减， + 1 1 . 0 1 0 1 0 1 Mx、My 异号，+[My]补 1 1 . 1 1 1 1 . 1 1 1 0 0 0 1 . + 0 0 . 1 0 1 0←1 1 0 0 1 0 1 . 0 0 0 1 1←1 1 r、My 同号，+[-My]补 0 0 . 1 0 0 0 1 1 1 0 1.0 + 1 1 . 0 1 0 1 0 1 r、My 异号， +[My]补 0 0 . 1 1 0 1 1 0 1.0 0 + 1 1 . 0←0 0 . 0 1 1 0 1 1 1 1 0 1 0 1 r、My 异号， +[My]补 0 0 . 0 0 1 0 1 1

续： 被除数（余数） 0 0 . 0 1 0 1 1 0 1 . 0 0 0 + 1 1 . 0 1 0← 商 1 1 1 . 0 1 0 1 1← 1 0 1 r、My 异号， +[My]补 1 1 . 1 0 1 0 1 1 1 1 0 1.0 0 0 1 + 0 0 . 1 0 1 0 1 1 r、My 同号，+[-My]补 0 0 . 0 0 0 0 1 0 1.0 0 0 1 0 + 1 1 .←0 0 . 0 0 0 0 0 1 1 1.0 0 0 1←0 1 0 1 0 1 r、My 异号， +[My]补 1 1 .

0 1 0　1 1 1　1 0 1 —— 恒置 1　+　0 0 . 1 0 1　0 1 1　　r、Mx 异号，(恢复余数)　　0 0 . 0 0 0　0 1 My]补= 1.000 101，　[r]补=0.000 010　　r= 0 .000÷0　　且 r、My 同号，　+[-My]补　　[Mx　2-6 =0.000 000 000 010×010 3)结果规格化：已是规格化数。4)舍入：已恒置 1 舍入。5)溢出：无　　y=2010×(-0.111 011)　方案二：采用阶补尾原形式：　　÷y]阶移尾补=1，010；1.000 101　　x ÷[x [x]阶补尾原=0，101；0.100 111　　[y]阶补尾原=0，011；1.101 011　1）阶码相减：　　[Ex]补+[-Ey]补=00，101+11，101　　=00，010（无溢出）

2）尾数相除：（原码加减交替除法）　　被除数（余数）　　商　　0 0 . 1 0 0　1 1 1　　0 . 0 0 0　0 0 0　　试减，　　+　1 1 . 0 1 0　1 0 1　　+[-My*] 补　　1 1 . 1 1 1　0 0 0　　0 .　+　0 0 .←1 1 . 1 1 1　1 0 0 1　0←0，+My*　　0 0 . 1 0 0　0 1 1　1<1 0 1　0 1 1　　r 1 . 0 0 0　1 1 0　　0.1　+　1 1 . 0 1 0　1 0 1　　0 0 . 1 1 0　1 1 0　　←0，+[-My*]补　　0 0 . 0 1 1　0 1 1 1> r0，+[-My*]补　　0 0 . 0 0> 0.1 1　+　1 1 . 0 1 0　1 0 1　　r 1　0 1 1

0 0 . 0 1 0　1 1 0　　←续：　被除数（余数）　　商 1 0，+[-My*]补　　1 1 . 1>0 . 1 1 1　+　1 1 . 0 1 0　1 0 1　　r　1 1 . 0 1 0　1 1 0　　0.1　1 1 0　+　0 0 . 1 0 1 ←0 1　0 1 1 1　0 0 . 0 0 0　0←0，+My*　　0 0 . 0 0 0　0 0 1 1<0 1 1　　r 0，>1 0　　0.1 1　1 0 1　+　1 1 . 0 1 0　1 0 1　　r　0.1 1 1　0 1 0　+　0 0 . 1 0 1　0 1←+[-My*]补　　1 1 . 0 1 0　1 1 1　1 0，恢复余数，+My*　　0 0 . 0 0 0　0 1 0　　<1　　r　2-6 =0.000 000 000 010×My]原= 1.111 010　　r*= 0 .000 010 ÷[Mx ÷y]阶补尾原=0，010；1.111 010　　x ÷3）结果规格化：已是规格化数。 4）舍入：无 5）溢出：无　　[x y=2010×（-0.111 010） ⑵ x=2110×（-0.101 101）　　y=2011×（-0.111 100）　　[x]阶移尾补=1，110；1.010 011　　[y]阶移尾补=1，011；1.000 100　1）阶码相减：　　[Ex]移+[-Ey]补=01，110+11，101　　=01，011（无溢出）

2）尾数相除：（补码加减交替除法）　　被除数（余数）　　商　　1 1 . 0 1 0　0 1 1　　0 . 0 0 0　0 0 0　　试减，　　+　0 0 . 1 1 1　1 0 0　　Mx、My 同号，+[-My]补　　0 0 . 0　0 0 . 0 1 1　1 1 0　　0 .　+　1 1 . 0 0 0　←0 1　1 1 1 1　1 1 . 0 0 0　1 0←1 0 0　　r、My 异号，+[My]补　　1 1 . 1 0 0　0 1 0　1 0　　0.1　+　0 0 . 1 1 1　1 0 0　　r、My 同号，　+[-My]补　　0 0 . 0 0 0　0 0 0　　0.1 0　+　1 1 .←　0 0 . 0 0 0　0 0 0 1 0 0 0　1 0 0　　r、My 异号，　+[My]补　　1 1 . 0 0 0　1 0 0

续：　被除数（余数）　1 0 . 0 0 1　0 0 0　　0 . 1 0 1　+　0 0 . 1 1←　　商 1　1 0 . 0 0 1　0←1　1 0 0　　r、My 同号，　+[-My]补　　1 1 . 0 0 0　1 0 0 1 0 0　　0.1　0 1 1　+　0 0 . 1 1 1　1 0 0　　r、My 同号，+[-My]补　　1 0 . 0 0 1　0 0 0　　0.1 0　1 1 1　+　0 0←　1 1 . 0 0 0　1 0 0 1　0.1 0 1 ←. 1 1 1　1 0 0　　r、My 同号，+[-My]补　　1 1 . 0 0 0　1 0 0　1 My]补= 0.101 111，÷1 1 1 —— 恒置 1　　r、Mx 同号，结束。[Mx 2-6 = -0.000 000 111 100×[r]补=1.000 100=[My]补 r= -0.111 100

注：由于补码加减交替除法算法中缺少对部分余数判“0”的步骤，因此算法运行中的某一步已除尽时，算法不会自动停止，而是继续按既定步数运行完。此时商由算法本身的这一缺陷引入了一个误差，而余数的误差正好等于除数。　　商的误差引入的原因：当 r、My 同号时，此题中表示够减（r、Mx 同号）；当 r、My 异号时，此题中表示不够减（r、Mx 异号）；因此，当 r=0 时，被判为不够减（实际上应为够减），商 0（实际上应商 1），由此引入了误差。

3）结果规格化：已是规格化数。　y=2011× 0.101 111÷y]阶移尾补=1，011；0.101 111　x ÷4）舍入：已恒置 1 舍入。 5）溢出：无　[x 方案二：采用阶补尾原形式：　[x]阶补尾原=0，110；1.101 101　[y]阶补尾原=0，011；1.111 100　1）阶码相减：　[Ex]补+[-Ey]补=00，110+11，101　=00，011（无溢出）

2）尾数相除：（原码加减交替除法）　被除数（余数）　商　0 0．1 0 1　1 0 1　0．0 0 0　0 0 0　试减，　+　1 1．0 0 0　1 0 0　1 1．1 0 0　0 1 0　←　+[-My*]补　1 1．1 1 0　0 0 1 1 0，+My*　0 0<　0．　+　0 0．1 1 1　1 0 0　r　0 0．1 1 1　1 0 0　0.1　+ 1 1．0 0 0←．0 1 1　1 1 0　1　0 0．0 0←0，+[-My*]补　0 0．0 0 0　0 0 0 1>　1 0 0　r　0，>0　0 0 0　0.1 1　+　1 1．0 0 0　1 0 0　r　+[-My*]补　1 1．0 0 0　1 0 0

1 0 .←续：　被除数（余数）　商 1 0 0 1　0 0 0　0．1 1 0　+　0 0．1 1 1　1 0 0　1 0．0 0 1　0 0 0　0.1←0，+My*　1 1．0 0 0　1 0 0 1<　r　0，+My*　1 1．0 0<　1 0 0　+　0 0．1 1 1　1 0 0　r　1 0．0 0 1　0 0 0　0.1 1　0 0 0　+　0 0．1 1 1　1←0　1 0 0 1　0.1 1 0　0 0 0←0，+My*　1 1．0 0 0　1 0 0　1<0 0　r　0，恢复余数，+My*　0 0．0 0 0　0 0 0<　+　0 0．1 1 1　1 0 0　r　2-6 = -0.000 000 000 000×My]原= 0.110 000，r= -0.000 000÷[Mx

3）结果规格化：已是规格化数。　y=2011× 0.110 000 29.÷y]阶补尾原=0，011；0.110 000　x ÷4）舍入：无 5）溢出：无　[x

31.设机器字长为 32 位，用与非门和与或非门设计一个并行加法器（假设与非门的延迟时间为 30ns，与或非门的延迟时间为 45ns），要求完成 32 位加法时间不得超过 0.6μs。画出进位链及加法器逻辑框图。

解：首先根据题意要求选择进位方案：　1）若采用串行进位链（行波进位），则在 di、ti 函数的基础上，实现 32 位进位需要的时间为：　30=1920ns　不满足 0.6μs 的加法时间限制，不能用。（设 1ty=30ns）　×32=64ty=64×T=2ty 30=600ns　×8 组=20ty=20×2）若采用单重分组跳跃进位（级连方式），则在 di、ti 的基础上，4 位一组分组，32 位进位需：　T=2.5ty　刚好满足 0.6 μs 加法时间的限制，

考虑到一次加法除进位时间外，还需 di、ti 函数的产生时间、和的产生时间（最高位和）等因素，故此进位方案仍不适用。　30=450ns　×6 组=15ty=15×结论：若采用单重分组跳跃进位，小组规模需在 6 位以上较为合适。即：　T=2.5ty 除进位外还有 150ns（约 5ty）左右的时间供加法开销，较充裕。　3）若采用双重分组跳跃进位（二级先行—级联进位），4 位一小组，4 小组为一大组分组，则 32 位进位需：　30=300ns　完全满足 0.6μs 的加法时间限制，可以使用。×4 级=10ty=10×T=2.5ty

32 位双重分组跳跃进位的进位链框图见教材 287 页图 6.23。　6 位一组单重分组跳跃

进位的进位链框图如下：

加法器逻辑框图如下。图中，进位链电路可选上述两种方案之一。

32. 设机器字长为16位，分别按4、4、4、4和5、5、3、3分组后，

（1）画出按两种分组方案的单重分组并行进位链框图，并比较哪种方案运算速度快。

（2）画出按两种分组方案的双重分组并行进位链框图，并对这两种方案进行比较。

（3）用74181和74182画出单重和双重分组的并行进位链框图。

解：（1）4—4—4—4分组的16位单重分组并行进位链框图见教材286页图6.22。

5—5—3—3分组的16位单重分组并行进位链框图如下：

（2）4—4—4—4分组的16位双重分组并行进位链框图见教材289页图6.26。

5—5—3—3分组的16位双重分组并行进位链框图如下：

5—5—3—3分组的进位时间=2.5ty×3=7.5ty；

4—4—4—4分组的进位时间=2.5ty×3=7.5ty；

可见，两种分组方案最长加法时间相同。

结论：双重分组并行进位的最长进位时间只与组数和级数有关，与组内位数无关。

（3）单重分组16位并行加法器逻辑图如下（正逻辑）：

注意： 1）74181芯片正、负逻辑的引脚表示方法；

2）为强调可比性，5-5-3-3分组时不考虑扇入影响；

3）181芯片只有最高、最低两个进位输入/输出端，组内进位无引脚；

4）181为4位片，无法5-5-3-3分组，只能4-4-4-4分组；

5）单重分组跳跃进位只用到181，使用182的一定是双重以上分组跳跃进位；

6）单重分组跳跃进位是并行进位和串行进位技术的结合；双重分组跳跃进位是二级并行进位技术；特别注意在位数较少时，双重分组跳跃进位可以采用全先行进位技术实现;位数较多时，可采用双重分组跳跃进位和串行进位技术结合实现。

第 七 章

0，RISC 和CISC 差异很大，它们主要有： （1） 指令系统：RISC 设计者把主要精力放在那些经常使用的指令上，尽量使它们具有简单高效的特色。对不常用的功能，常通过组合指令来完成。因此，在RISC 机器上实现特殊功能时，效率可能较低。但可以利用流水技术和超标量技术加以改进和弥补。而CISC 计算机的指令系统比较丰富，有专用指令来完成特定的功能。因此，处理特殊任务效率较高。 （2） 存储器操作：RISC 对存储器操作有限制，使控制简单化；而CISC 机器的存储器操作指令多，操作直接。 （3） 程序：RISC 汇编语言程序一般需要较大的内存空间，实现特殊功能时程序复杂，不易设计；而CISC 汇编语言程序编程相对简单，科学计算及复杂操作的程序设计相对容易，效率较高。 （4） 中断：RISC 机器在一条指令执行的适当地方可以响应中断;而CISC 机器是在一条指令执行结束后响应中断。 （5）CPU：

RISC CPU 包含有较少的单元电路，因而面积小、功耗低；而 CISC CPU 包含有丰富的电路单元，因而功能强、面积大、功耗大。　　（6） 设计周期：RISC 微处理器结构简单，布局紧凑，设计周期短，且易于采用最新技术；CISC 微处理器结构复杂，设计周期长。　　（7） 用户使用：RISC 微处理器结构简单，指令规整，性能容易把握，易学易用；CISC 微处理器结构复杂，功能强大，实现特殊功能容易。　　（8） 应用范围：由于 RISC 指令系统的确定与特定的应用领域有关，故 RISC 机器更适合于专用机；而 CISC 机器则更适合于通用机。

1. 零地址指令的操作数来自哪里？？各举一例说明。

答：零地址指令的操作数来自 ACC，为隐含约定。

在一地址指令中，另一个操作数的地址通常可采用 ACC 隐含寻址方式获得。

2. 对于二地址指令而言，操作数的物理地址可安排在什么地方？举例说明。

答：对于二地址指令而言，操作数的物理地址可安排在寄存器内、指令中或内存单元内等。

7.6 某指令系统字长为 16 位，地址码取 4 位，试提出一种方案，使该指令系统有 8 条三地址指令、16 条二地址指令、100 条一地址指令。

解：三地址指令格式如下：

4	4	4	4
OP	**A1**	**A2**	**A3**

解题思路：以三地址指令格式为该指令系统的基本格式。以此格式为基础，采用扩展操作码技术，设计出题意所要求的地址码结构的指令。

指令操作码分配方案如下：

4 位 OP

0000，

……，　　A1，A2，A3；8 条三地址指令

0111，

1000，0000，

……，……，　　A2，A3；16 条二地址指令

1000，1111，

1001，0000，0000，

……，……，……，　　A3；100 条一地址指令

1001，0110，0011，

1001，0110，0100，

……，……，……，　　冗余编码

1001，1111，1111，　　可用来扩充一、零地址指令条数

1010，

……，　　冗余编码

1111，　　可用来扩充三、二、一、零地址指令条数

8. 某机指令字长 16 位，每个操作数的地址码为 6 位，设操作码长度固定，指令分为零地址、一地址和二地址三种格式。若零地址指令有 M 条，一地址指令有 N 种，则二地址指令最多有几种？若操作码位数可变，则二地址指令最多允许有几种？

解：1）若采用定长操作码时，二地址指令格式如下：

OP（4 位）	A1（6 位）	A2（6 位）

设二地址指令有 K 种，则：$K=2^4-M-N$

当 M=1（最小值），N=1（最小值）时，二地址指令最多有：Kmax=16-1-1=14 种

3） 若采用变长操作码时，二地址指令格式仍如 1）所示，但操作码长度可随地址码的个数而变。此时，$K= 2^4 -（N/2^6 + M/2^{12}）$；

当（$N/2^6 + M/2^{12}$）≤1 时（$N/2^6 + M/2^{12}$ 向上取整），K 最大，则二地址指令最多有：Kmax=16-1=15 种（只留一种编码作扩展标志用。）

11. 画出先变址再间址及先间址再变址的寻址过程示意图。

解：1）先变址再间址寻址过程简单示意如下：

EA=[(IX)+A]，IX→(IX)+1

2）先间址再变址寻址过程简单示意如下：EA=(IX)+(A)，IX→ (IX)+1

16. 某机主存容量为 4M×16 位，且存储字长等于指令字长，若该机指令系统可完成 108 种操作，操作码位数固定，且具有直接、间接、变址、基址、相对、立即等六种寻址方式，试回答：（1）画出一地址指令格式并指出各字段的作用；

（2）该指令直接寻址的最大范围；

（3）一次间址和多次间址的寻址范围；

（4）立即数的范围（十进制表示）；

（5）相对寻址的位移量（十进制表示）；

（6）上述六种寻址方式的指令哪一种执行时间最短？哪一种最长？为什么？哪一种便于程序浮动？哪一种最适合处理数组问题？

（7）如何修改指令格式，使指令的寻址范围可扩大到 4M？

（8）为使一条转移指令能转移到主存的任一位置，可采取什么措施？简要说明之。

解：（1）单字长一地址指令格式：

OP（7 位）	M（3 位）	A（6 位）

OP 为操作码字段，共 7 位，可反映 108 种操作；

M 为寻址方式字段，共 3 位，可反映 6 种寻址操作；

A 为地址码字段，共 16-7-3=6 位。

（2）直接寻址的最大范围为 2^6=64。

（3）由于存储字长为 16 位，故一次间址的寻址范围为 2^{16}；若多次间址，需用存储字的最高位来区别是否继续间接寻址，故寻址范围为 2^{15}。

（4）立即数的范围为-32——31（有符号数），或 0——63（无符号数）。

（5）相对寻址的位移量为-32——31。

（6）上述六种寻址方式中，因立即数由指令直接给出，故立即寻址的指令执行时间最短。间接寻址在指令的执行阶段要多次访存(一次间接寻址要两次访存，多次间接寻址要多次访存)，故执行时间最长。变址寻址由于变址寄存器的内容由用户给定，而且在程序的执行过程中允许用户修改，而其形式地址始终不变，故变址寻址的指令便于用户编制处理数组问题的程序。相对寻址操作数的有效地址只与当前指令地址相差一定的位移量，与直接寻址相比，更有利于程序浮动。

（7）**方案一**：为使指令寻址范围可扩大到 4M，需要有效地址 22 位，此时可将单字长一地址指令的格式改为双字长，如下图示：

OP（7 位）	MOD（3 位）	A（高 6 位）
A（低 16 位）		

方案二：如果仍采用单字长指令（16 位）格式，为使指令寻址范围扩大到 4M，可通过段寻址方案实现。安排如下：

硬件设段寄存器 DS（16 位），用来存放段地址。在完成指令寻址方式所规定的寻址操作

后，得有效地址 EA（6 位），再由硬件自动完成段寻址，最后得 22 位物理地址。 即：物理地址=（DS）×2^6 + EA

注：段寻址方式由硬件隐含实现。在编程指定的寻址过程完成、EA 产生之后由硬件自动完成，对用户是透明的。

方案三：在采用单字长指令（16 位）格式时，还可通过页面寻址方案使指令寻址范围扩大到 4M。安排如下：

硬件设页面寄存器 PR（16 位），用来存放页面地址。指令寻址方式中增设页面寻址。当需要使指令寻址范围扩大到 4M 时，编程选择页面寻址方式，则：EA =（PR）‖A （有效地址=页面地址“拼接”6 位形式地址），这样得到 22 位有效地址。

（8）为使一条转移指令能转移到主存的任一位置，寻址范围须达到 4M，除了采用(7) 方案一中的双字长一地址指令的格式外，还可配置 22 位的基址寄存器或 22 位的变址寄存器，使 EA =(BR) + A （BR 为 22 位的基址寄存器)或 EA =（IX）+ A(IX 为 22 位的变址寄存器)，便可访问 4M 存储空间。还可以通过 16 位的基址寄存器左移 6 位再和形式地址 A 相加，也可达到同样的效果。

总之，不论采取何种方式，最终得到的实际地址应是 22 位。

19. 某 CPU 内有 32 个 32 位的通用寄存器，设计一种能容纳 64 种操作的指令系统。假设指令字长等于机器字长，试回答以下问题：

（1）如果主存可直接或间接寻址，采用寄存器—存储器型指令，能直接寻址的最大存储空间是多少？画出指令格式并说明各字段的含义。

（2）在满足（1）的前提下，如果采用通用寄存器作基址寄存器，则上述寄存器—存储器型指令的指令格式有何特点？画出指令格式并指出这类指令可访问多大的存储空间？

解：（1）如采用 RS 型指令，则此指令一定是二地址以上的地址格式，指令格式如下：

OP（6 位）	R（5 位）	I(1 位)	A（20 位）

操作码字段 OP 占 6 位，因为 2^6>=64；

寄存器编号 R 占 5 位，因为 2^5>=32；

间址位 I 占 1 位，当 I=0，存储器寻址的操作数为直接寻址，当 I=1 时为间接寻址；

形式地址 A 占 20 位，可以直接寻址 2^{20}字。

（2）如采用基址寻址，则指令格式中应给出基址寄存器号，以指定哪一个通用寄存器用作基址寄存器。指令格式变为：

OP（6 位）	源 R（5 位）	I（1 位）	X（1 位）	目标 R（5 位）	A（14 位）

增加寻址特征位 X，当 X=1 时，以目标寄存器 R 作为基址寄存器进行基址寻址。

基址寻址可访问存储空间为：2^{32}字。

6. 某机字长 16 位，存储器直接寻址空间为 128 字，变址时的位移量为-64~+63，16 个通用寄存器均可作为变址寄存器。采用扩展操作码技术，设计一套指令系统格式，满足下列寻址类型的要求：（1）直接寻址的二地址指令 3 条；（2）变址寻址的一地址指令 6 条；（3）寄存器寻址的二地址指令 8 条；（4）直接寻址的一地址指令 12 条；（5）零地址指令 32 条。试问还有多少种代码未用？若安排寄存器寻址的一地址指令，最多还能容纳多少条？

解：题意分析：设指令字长=机器字长，128 字的直接寻址空间要求形式地址 A 为 7 位，-64~+63 的位移量也需 7 位（6 位加 1 位符号位），16 个通用寄存器作变址寄存器需 4 位变址寄存器号，则指令格式为：（1）直接寻址的二地址指令：2 7 7

（3）寄存器寻址的二地址指令：8 4 4

（6）若安排寄存器寻址的一地址指令，指令格式应为： 12 4

操作码编码分配： 0 0 0 1 A1，A2； 3 条直接寻址的二地址指令。 1 0 1 1 0 0 0 …… …… IX，A；6 条变址寻址的一地址指令。 1 1 1 0 1 1 1 1 1 0 0 0 …… …… …… Ri，Rj； 1 1 1 1 0 1 1 1 8 条寄存器寻址的二地址指令。 1 1 1 1 1 1 0 0 0 0 …… …… …… A； 1 1 1 1 1 1 0 1 1 12 条直接寻址的一地址指令。

续： 1 1 1 1 1 1 0 0 0 0 0 …… …… …… …… Ri； 1 1 1 1 1 1 0 1 1 1 30 条寄存器寻址的 1 1 1 1 1 1 1 1 1 0 0 0 一地址指令。 …… …… …… …… （利用 30 个冗余编码） 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 0 0 0 0 …… …… …… …… …… 32 条 0 地址 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 指令

不画所设计的指令格式图，只分配指令操作码编码。 ×讨论： 指令格式中安排寻址方式字段，例： 2 3 4 7 ×

评注：这是一道指令格式设计题，本题已给出了各种指令所需的条数，因此，在根据题意画出各种指令的格式后，剩下的工作就是要为每一条指令分配编码。在采用扩展操作码技术分配指令编码时，扩展的基本方法是在所设计的指令系统中，选定一种操作码位数最少的指令格式作为基本格式，然后在这种基本格式的基础上进行操作码编码的扩展。为便于硬件译码结构的实现，编码分配应尽量做到有序、有规律。特别是扩展标志码的选择，应尽量采用特征较强的编码，象全‘1’编码等。另外，应在某类指令的编码全部安排完后，再考虑安排扩展标志码，以避免漏排或重码等不必要的混乱。

第八章

1、什么是中断

在 CPU 运行过程中，由于内部或外部某个随机事件的发生，使 CPU 暂停正在运行的程序，而转去执行处理引起中断事件的程序，完成后返回原来的程序继续执行。这个过程称为中断。

2. 什么是指令周期？指令周期是否有一个固定值？为什么？

解：指令周期是指取出并执行完一条指令所需的时间。

由于计算机中各种指令执行所需的时间差异很大，因此为了提高 CPU 运行效率，即使在同步控制的机器中，不同指令的指令周期长度都是不一致的，也就是说指令周期对于不同的指令来说不是一个固定值。

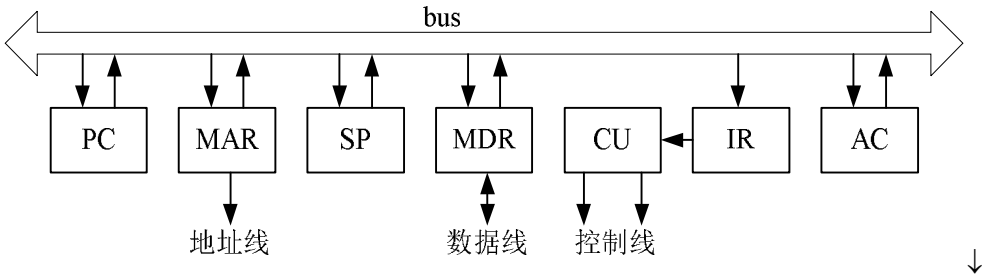
4. 设 CPU 内有下列部件：PC、IR、SP、AC、MAR、MDR 和 CU。

（1）画出完成间接寻址的取数指令 LDA@X（将主存某地址单元 X 的内容取至 AC 中）的数据流（从取指令开始）。

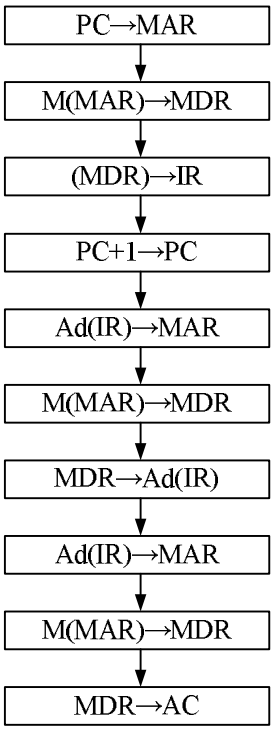
（2）画出中断周期的数据流。

解：CPU 中的数据流向与所采用的数据通路结构直接相关，不同的数据通路中的数据流是不一样的。常用的数据通路结构方式有直接连线、单总线、双总线、三总线等形式，目前大多采用总线结构，直接连线方式仅适用于结构特别简单的机器中。

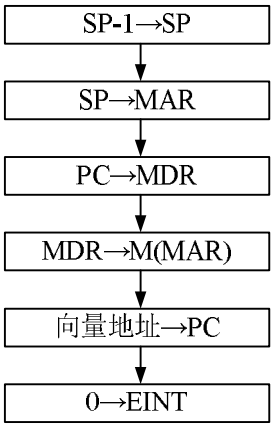
为简单起见，本题采用单总线将题中所给部件连接起来，框图如下：



（1）LDA@X 指令周期数据流程图：



（2）中断周期流程图如下：



注：解这道题有两个要素，首先要根据所给部件设计好数据通路，即确定信息流动的载体。其次选择好描述数据流的方法，无论采用什么样的表达方式，其关键都要能清楚地反映数据在通路上流动的顺序，即强调一个“流”字。较好的表达方式是流程图的形式。

5. 中断周期前是什么阶段？中断周期后又是什么阶段？在中断周期 CPU 应完成什么操作？

答：中断周期前是执行周期，中断周期后是取指周期。在中断周期，CPU 应完成保存断点、将中

断向量送 PC 和关中断等工作。

7. 什么叫系统的并行性？粗粒度并行和细粒度并行有何区别？

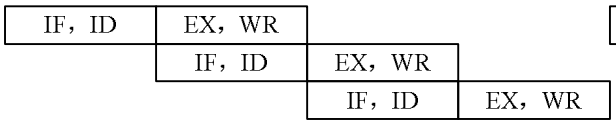
答：所谓并行性包含同时性和并发性。同时性是指两个或两个以上的事件在同一时刻发生，并发性是指两个或多个事件在同一时间段发生。即在同一时刻或同一时间段内完成两个或两个以上性质相同或性质不同的功能，只要在时间上存在相互重叠，就存在并行性。

并行性又分为粗粒度并行和细粒度并行两类。粗粒度并行是指在多个处理机上分别运行多个进程，由多台处理机合作完成一个程序，一般用算法实现。细粒度并行是指在处理机的指令级和操作级的并行性。

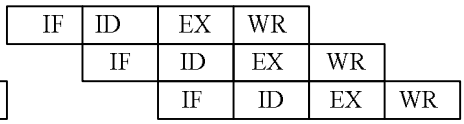
8. 什么是指令流水？画出指令二级流水和四级流水的示意图，它们中哪个更能提高处理机速度，为什么？

答：指令流水是指将一条指令的执行过程分为 n 个操作时间大致相等的阶段，每个阶段由一个独立的功能部件来完成，这样 n 个部件就可以同时执行 n 条指令的不同阶段，从而大大提高 CPU 的吞吐率。

指令二级流水和四级流水示意图如下：



二级指令流水示意图



四级指令流水示意图

四级流水更能提高处理机的速度。分析如下：

假设 IF、ID、EX、WR 每个阶段耗时为 t，则连续执行 n 条指令

采用二级流水线时，耗时为：$4t+(n-1)2t=(2n+2)t$

采用四级流水线时，耗时为：$4t+(n-1)t=(n+3)t$

在 $n>1$ 时，$n+3<2n+2$，可见四级流水线耗时比二级流水线耗时短，因此更能提高处理机速度。

17. 在中断系统中 INTR、INT、EINT 三个触发器各有何作用？

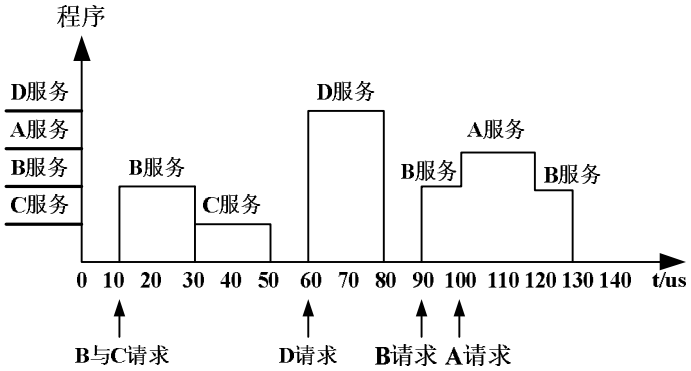
解：INTR——中断请求触发器，用来登记中断源发出的随机性中断请求信号，以便为 CPU 查询中断及中断排队判优线路提供稳定的中断请求信号。

EINT——中断允许触发器，CPU 中的中断总开关。当 EINT=1 时，表示允许中断（开中断），当 EINT=0 时，表示禁止中断（关中断）。其状态可由开、关中断等指令设置。

INT——中断标记触发器，控制器时序系统中周期状态分配电路的一部分，表示中断周期标记。当 INT=1 时，进入中断周期，执行中断隐指令的操作。

24. 现有 A、B、C、D 四个中断源，其优先级由高向低按 A、B、C、D 顺序排列。若中断服务程序的执行时间为 20μs，请根据下图所示时间轴给出的中断源请求中断的时刻，画出 CPU 执行程序的轨迹。

解：A、B、C、D 的响优先级即处理优先级。CPU 执行程序的轨迹图如下：



25. 某机有五个中断源 L0、L1、L2、 L3、L4，按中断响应的优先次序由高向低排序为 L0→L1→L2→L3→L4，根据下示格式，现要求中断处理次序改为 L1→L4→L2→L0→L3，根据下面的格式，写出各中断源的屏蔽字。

解：各中断源屏蔽状态见下表：

中断源	屏蔽字				
	0	1	2	3	4
I0	1	0	0	1	0
I1	1	1	1	1	1
I2	1	0	1	1	0
I3	0	0	0	1	0
I4	1	0	1	1	1

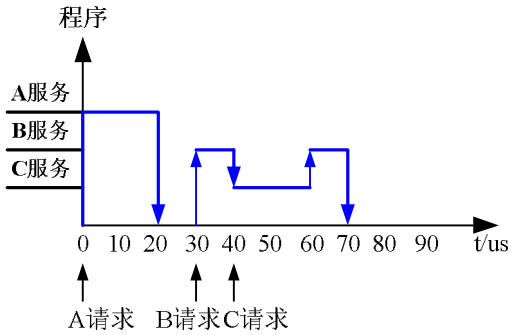
表中：设屏蔽位=1，表示屏蔽；屏蔽位=0，表示中断开放。

26. 设某机配有 A、B、C 三台设备，其优先顺序按 A→B→C 降序排列，为改变中断处理次序，它们的中断屏蔽字设置如下：

设备	屏蔽字
A	111
B	010
C	011

请按下图所示时间轴给出的设备请求中断的时刻，画出 CPU 执行程序的轨迹。设 A、B、C 中断服务程序的执行时间均为 20μ s。

解：A、B、C 设备的响应优先级为 A 最高、B 次之、C 最低，处理优先级为 A 最高、C 次之、B 最低。CPU 执行程序的轨迹图如下：



第 九 章

2. 控制单元的功能是什么？其输入受什么控制？

答：控制单元的主要功能是发出各种不同的控制信号。其输入受时钟信号、指令寄存器的操作码字段、标志和来自系统总线的控制信号的控制。

3. 什么是指令周期、机器周期和时钟周期？三者有何关系？

答：CPU每取出并执行一条指令所需的全部时间叫指令周期；

机器周期是在同步控制的机器中，执行指令周期中一步相对完整的操作（指令步）所需时间,通常安排机器周期长度等于主存周期；

时钟周期是指计算机主时钟的周期时间，它是计算机运行时最基本的时序单位，对应完成一个微操作所需时间，通常时钟周期等于计算机主频的倒数。

4. 能不能说机器的主频越快，机器的速度就越快，为什么？

解：不能说机器的主频越快，机器的速度就越快。因为机器的速度不仅与主频有关，还与数据通路结构、时序分配方案、ALU运算能力、指令功能强弱等多种因素有关，要看综合效果。

5. 设机器A的主频为8MHz，机器周期含4个时钟周期，且该机的平均指令执行速度是0.4MIPS,试求该机的平均指令周期和机器周期，每个指令周期中含几个机器周期？如果机器B的主频为12MHz，且机器周期也含4个时钟周期，试问B机的平均指令执行速度为多少MIPS?

解：先通过A机的平均指令执行速度求出其平均指令周期，再通过主频求出时钟周期，然后进一步求出机器周期。B机参数的算法与A机类似。计算如下：

A机平均指令周期=1/0.4MIPS=2.5μs

A机时钟周期=1/8MHz=125ns

A机机器周期=125ns×4=500ns=0.5μs

A机每个指令周期中含机器周期个数=2.5μs÷0.5μs=5个

B机时钟周期 =1/12MHz 83ns

B机机器周期 =83ns×4=332ns

设B机每个指令周期也含5个机器周期，则：

B机平均指令周期=332ns×5=1.66μs

B机平均指令执行速度=1/1.66μs=0.6MIPS

结论：主频的提高有利于机器执行速度的提高。

6. 设某机主频为8MHz，每个机器周期平均含2个时钟周期，每条指令平均有4个机器周期，试问该机的平均指令执行速度为多少MIPS？若机器主频不变，但每个机器周期平均含4个时钟周期，每条指令平均有4个机器周期，则该机的平均指令执行速度又是多少MIPS？由此可得出什么结论？

解：先通过主频求出时钟周期，再求出机器周期和平均指令周期，最后通过平均指令周期的倒数求出平均指令执行速度。计算如下：

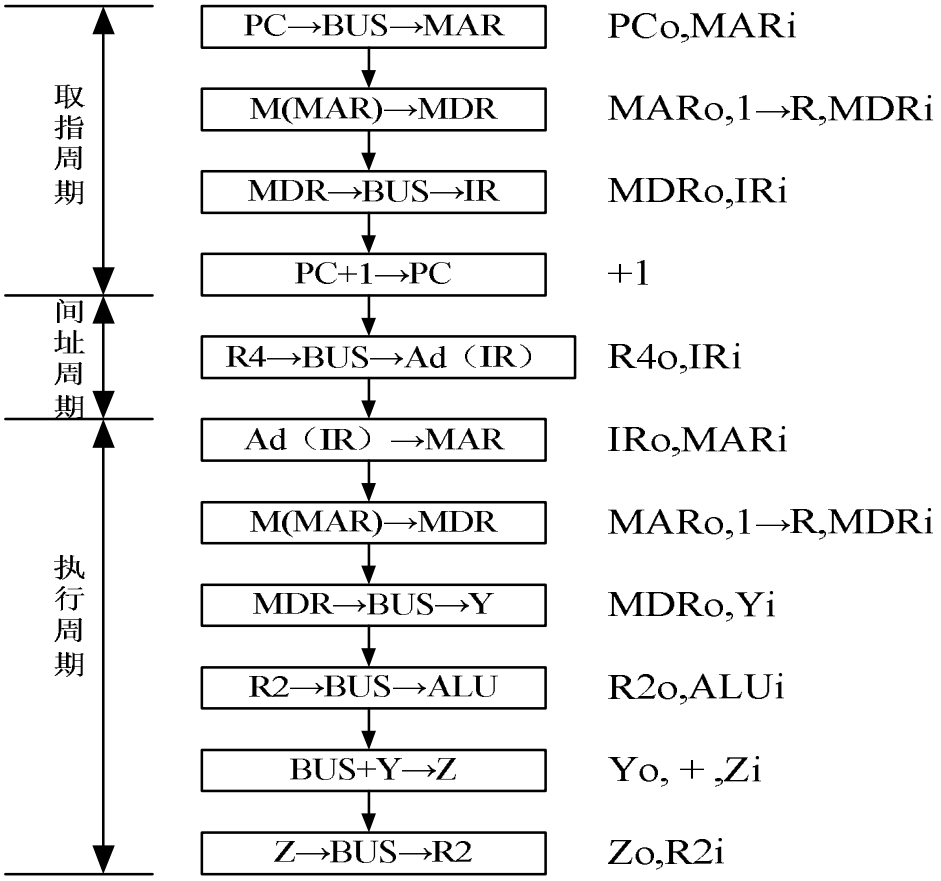
时钟周期=1/8MHz=0.125×10^{-6}s

机器周期=0.125×10^{-6}s×2=0.25×10^{-6}s

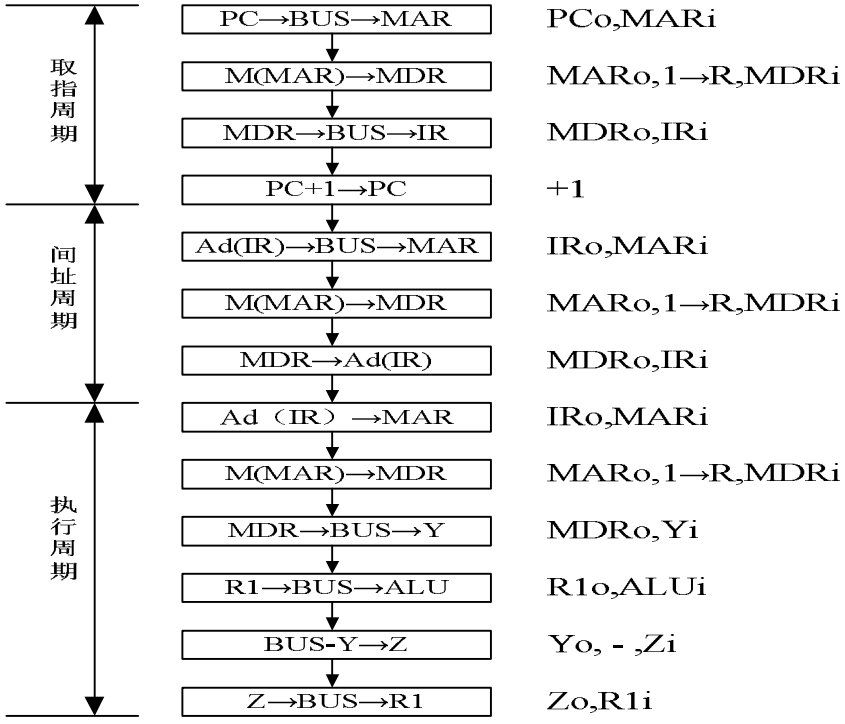
平均指令周期=0.25×10^{-6}s×4=10^{-6}s

平均指令执行速度=1/10^{-6}s=1MIPS

当参数改变后：机器周期= 0.125×10-6s×4=0.5×10-6s
平均指令周期=0.5×10-6s×4=2×10-6s
平均指令执行速度=1/（2×10-6s） =0.5MIPS
结论：两个主频相同的机器，执行速度不一定一样。
7. 某CPU的主频为10MHz，若已知每个机器周期平均包含4个时钟周期，该机的平均指令执行速度为1MIPS，试求该机的平均指令周期及每个指令周期含几个机器周期？若改用时钟周期为0.4μs的CPU芯片，则计算机的平均指令执行速度为多少MIPS？若要得到平均每秒80万次的指令执行速度，则应采用主频为多少的CPU芯片？ 解：先通过主频求出时钟周期时间，再进一步求出机器周期和平均指令周期。
时钟周期=1/10MHz=0.1×10-6s
机器周期=0.1×10-6s×4=0.4×10-6s
平均指令周期=1/1MIPS=10-6s
每个指令周期所含机器周期个数=10-6s /0.4×10-6s =2.5个
当芯片改变后：机器周期=0.4μs×4=1.6μs
平均指令周期=1.6μs×2.5=4μs
平均指令执行速度=1/4μs=0.25MIPS
若要得到平均每秒80万次的指令执行速度，则:
平均指令周期=1/0.8MIPS=1.25×10-6=1.25μs
机器周期=1.25μs÷2.5=0.5μs
时钟周期= 0.5μs÷4=0.125μs
CPU主频=1/0.125μs=8MHz
8. 某计算机的主频为 6MHz，各类指令的平均执行时间和使用频度如下表所示，试计算该机的速度（单位用 MIPS 表示），若上述 CPU 芯片升级为 10MHz，则该机的速度又为多少？
解: (1)指令平均运行时间 =（0.6×0.35+0.8×0.45+10×0.05+1.4×0.15）μs= 1.28μs
机器平均运行速度 = 1/1.28μs≈0.78MIPS
(2)时钟周期 = 1/6MHz≈0.167μs
指令平均运行周期数 = 1.28μs ÷0.167μs≈7.66CPI
若CPU芯片升级为10MHz，时钟周期 = 1/10MHz=0.1μs
指令平均运行时间 = 0.1μs×7.66=0.766μs
机器平均运行速度 ＝1/0.766μs≈1.3MIPS
9. 试比较同步控制、异步控制和联合控制的区别。
答：同步控制是指任何一条指令或指令中任何一个微操作的执行都是事先确定的，并且都受同一基准时标的时序信号所控制的方式。异步控制无基准时标信号，微操作的时序是由专门的应答线路控制，即控制单元发出执行某一微操作的控制信号后，等待执行部件完成了该操作后发回“回答”或“结束”信号，再开始新的微操作。联合控制是同步控制和异步控制相结合的方式，即大多数操作（如 CPU 内部各操作）在同步时序信号的控制下进行，少数时间难以确定的微操作（如涉及 I/O 操作）采用异步控制
9.13. 设 CPU 内部结构如图 9.4 所示，此外还设有 R1~R4 四个寄存器，它们各自的输入和输出端都与内部总线相通，并分别受控制信号控制（如 R2i 为寄存器 R2 的输入控制；R2o 为 R2 的输出控制）。要求从取指令开始，写出完成下列指令所需的全部微操作和控制信号。
（1）ADD R2，@R4 ； ((R2)+((R4)) →R2，寄存器间接寻址)
（2）SUB R1,@mem ； ((R1)-((mem)) →R1，存储器间接寻址)
解：（1）ADD R2，@R4 的指令周期信息流程图及微操作控制信号如下：



（2）SUB R1,@mem 指令周期信息流程图及微操作控制信号如下：



第　十　章

1. 假设响应中断时，要求将程序断点存在堆栈内，并且采用软件办法寻找中断服务程序的入口地址，试写出中断隐指令的微操作及节拍安排。

解：设软件查询程序首址为0 号内存单元，则中断隐指令的微操作命令及节拍安排如下：

T0 0 SP

T1 ， SP→ MAR

T2 SP→W，SP+1 M（MAR）

T3 PSW→MAR，MDR→SP

T4 1→ W，SP+1→MDR， →M（MAR） PC→EINT 由于题意中没有给出确切的数据通路结构，故上述节拍分配方案的并行性较低。→PC，MDR→0 →MDR，1→MAR

2. 写出完成下列指令的微操作及节拍安排（包括取指操作）。

（1）指令ADD R1，X 完成将R1 寄存器的内容和主存X 单元的内容相加，结果存于R1 的操作。

（2）指令ISZ X 完成将主存X 单元的内容增1，并根据其结果若为0，则跳过下一条指令执行。

解：设采用单总线结构的CPU 数据通路如下图所示，且ALU 输入端设两个暂存器C、D（见17题

图）。并设采用同步控制，每周期3 节拍：

PC MAR SP MDR CU IR AC

bus

地址线数据线控制线

（1）指令ADD R1，X 的微操作及节拍安排如下：

取指周期：T0 PC→MAR，1→ R

T1 M(MAR) →MDR，PC+1→PC

T2 MDR→IR,OP(IR) →ID

执行周期1：T0 Ad（IR）→MAR，1→R

T1 M(MAR) →MDR

T2 MDR→D

执行周期2：T0 R1→C

T1 +

T2 ALU→R1

（2）指令ISZ X 的微操作及节拍安排:

取指周期同（1）:略

执行周期1: T0 Ad（IR）→MAR，1→R

T1 M(MAR)→MDR

T2 MDR→C，+1→ALU

执行周期2：T0 ALU→MDR，1→W

T1 (PC+1)·Z+ PC· Z →PC

3. 按序写出下列程序所需的全部微操作命令及节拍安排。

解：由于题意未明确要求采用何种控制器结构，故仍按较简单的组合逻辑时序关系安排节拍（单总线、同步控制，假设同上题）： LDA 206 指令： ID →IR，OP(IR)→MDR T2 MDR→R T1 PC+1，M(MAR)→MAR，1→取指周期： T0 PC AC→MDR T2 MDR→R T1 M(MAR)→MAR，1→执行周期： T0 206(IR)

ADD C →MDR，AC→R T1 M(MAR)→MAR，1→207 指令：取指周期：同上。 执行周期 1： T0 207(IR) AC BAN 204 指令：取指周期：同上。 →D 执行周期 2： T0 T1 T2 +，ALU→T2 MDR PC→ 执行周期：（设 N 为结果为负标志） T0 T1 T2 N·204(IR)

STA 205 指令： -W T2 →MDR，0→MAR T1 AC→ 取指周期：同上。 执行周期： T0 205(IR) G →M(MAR) STP 指令： 取指周期：同上。 执行周期： T0 T1 T2 0→MDR （G 为停机标志。）

6. 已知带返转指令的含义如下图所示，写出机器在完成带返转指令时，取指阶段和执行阶段所需的全部微操作及节拍安排。 主程序 子程序

解：假设同上题，仍按组合逻辑、单总线、同步控制安排，带返转指令的全部微操作及节拍如下： ID →IR,OP(IR)→MDR T2 MDR→R T1 PC+1,M(MAR)→MAR,1 → 取 指 周 期 ： T0 PC M(MAR) → W T2 MDR→MDR),1→MDR(PC→MAR T1 M+1→执行周期： T0 Ad(IR) PC)→PC (MAR +1→ K+1

12. 能否说水平型微指令就是直接编码的微指令，为什么？

解：不能说水平型微指令就是直接编码的微指令，因为符合水平型微指令特征的微指令都属于水平型微指令，常见的有：直接编码、字段直接编码、字段间接编码，及混合编码等。直接编码的微指令只是最典型的一种。

15. 设控制存储器的容量为512×48位，微程序可在整个控存空间实现转移，而控制微程序转移的条件共有4个（采用直接控制），微指令格式如下：

解：因为控制存储器共有512*48=2^9*48

所以，下址字段应有9位，微指令字长48位

又因为控制微程序转移的条件有4个，4+1<=2^3

所以判断测试字段占3位

因此控制字段位数为：48-9-3=36

微指令格式为：

48 13 12 10 9 1

控制字段 测试字段 下址字段

21. 下表给出8条微指令I1~I8及所包含的微命令控制信号，设计微指令操作控制字段格式，要求所使用的控制位最少，而且保持微指令本身内在的并行性。

解：为使设计出的微指令操作控制字段最短，并且保持微指令本身内在的并行性，应采用混合编码法。首先找出互斥的微命令组，为便于分析，将微命令表重画如下：

由表中微命令的分布情况可看出：a、b、c、d、e微命令的并行性太高，因此不能放在同一字段中。另外，由分析可知，在2、3、4分组的互斥组中，3个一组的微命令互斥组对控制位的压缩作用最明显。因此，应尽可能多的找出3个一组的互斥组。现找出的互斥组有：cfj，dij，efh，fhi，bgj，ehj，efj„„等等。

从中找出互不相重的互斥组有两个：dij，efh。则：微指令操作控制字段格式安排如下：

1 1 1 b c efh dij

1 2 2

a

19. 假设机器的主要部件有：程序计数器 PC，指令寄存器 IR，通用寄存器 R0、R1、R2、R3，暂存器 C、D，ALU，移位器，存储器地址寄存器 MAR，存储器数据寄存器 MDR 及存储矩阵 M。 （1）要求采用单总线结构画出包含上述部件的硬件框图，并注明数据流动方向。 （2）画出 ADD（R1），（R2）指令在取指阶段和执行阶段的信息流程图。 R1 寄存器存放源操作数地址，R2 寄存器存放目的操作数的地址。 （3）写出对应该流程图所需的全部微操作命令。

解： （1）采用单总线结构的 CPU 硬件框图如下：

（2）ADD（R1），（R2） ↓（3）对应该流程图所 指令流程图如下： 需的全部微操作命令。

R→R1o，MARi 1 R MDRo，Di→MDRo，Ci R2o，MARi 1

-W 公操作→+，D，MDRi 0

20. 假设机器的主要部件同 17 题，外加一个控制门 G。 （1）要求采用双总线结构（每组总线的数据流动方向是单向的），画出包含上述部件的硬件框图，并注明数据流动方向。 （2）画出 SUB R1 操作的指令周期信息流程图（假设指令地址已放在 PC 中），并列出相应的微操作控制信号序列。→R1，R3 完成（R1）-（R3）

解： （1）双总线结构的 CPU 硬件框图如下：

（2） SUB R1，R3 指令周期流程图如下：

R1o，G，Ci R2o，G，Di -，D，G，R1i