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

第二章 计算机的发展与应用

- 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。

- 3. 什么是摩尔定律？该定律是否永远生效？为什么？
- 答：P23，否，P36

系 统 总 线

第 三 章

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

解：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？

解：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；独立请求方式判优速度最快，但硬件器件用量大，连线多，成本较高。

5. 解释概念：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、总线的通信控制。

解：

总线宽度——指数据总线的位（根）数，用bit（位）作单位。

总线带宽——指总线在单位时间内可以传输的数据总量，相当于总线的数据传输率，等于总线工作频率与总线宽度（字节数）的乘积。

总线复用——指两种不同性质且不同时出现的信号分时使用同一组总线，称为总线的“多路分时复用”。

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——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。

解：

同步通信——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——不由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？

解：

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. 什么是总线标准？为什么要设置总线标准？目前流行的总线标准有哪些？什么是即插即用？哪些总线有这一特点？

解：

总线标准——可理解为系统与模块、模块与模块之间的互连的标准界面。

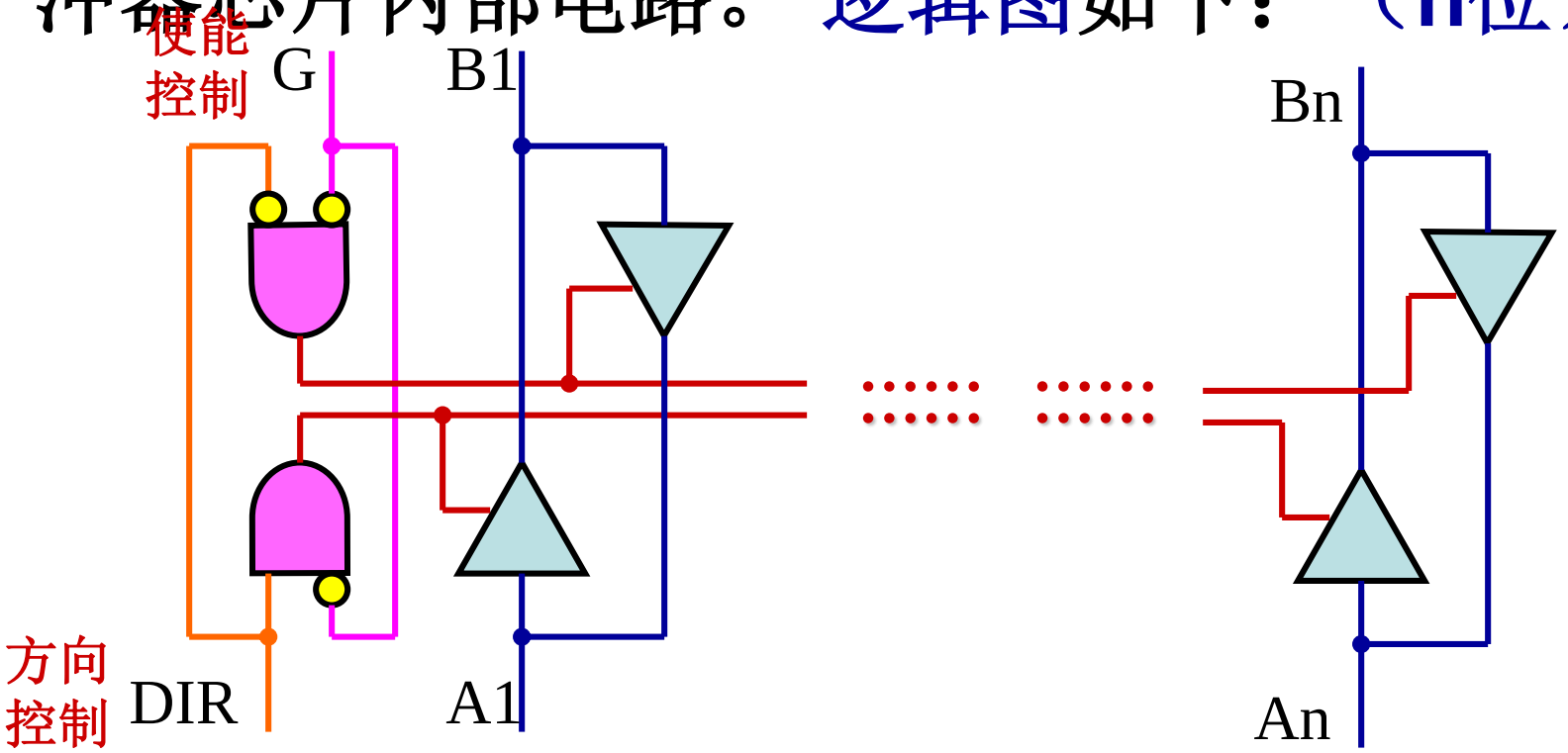
总线标准的设置主要解决不同厂家各类模块化产品的兼容问题；

目前流行的总线标准有：**ISA、EISA、PCI**等；

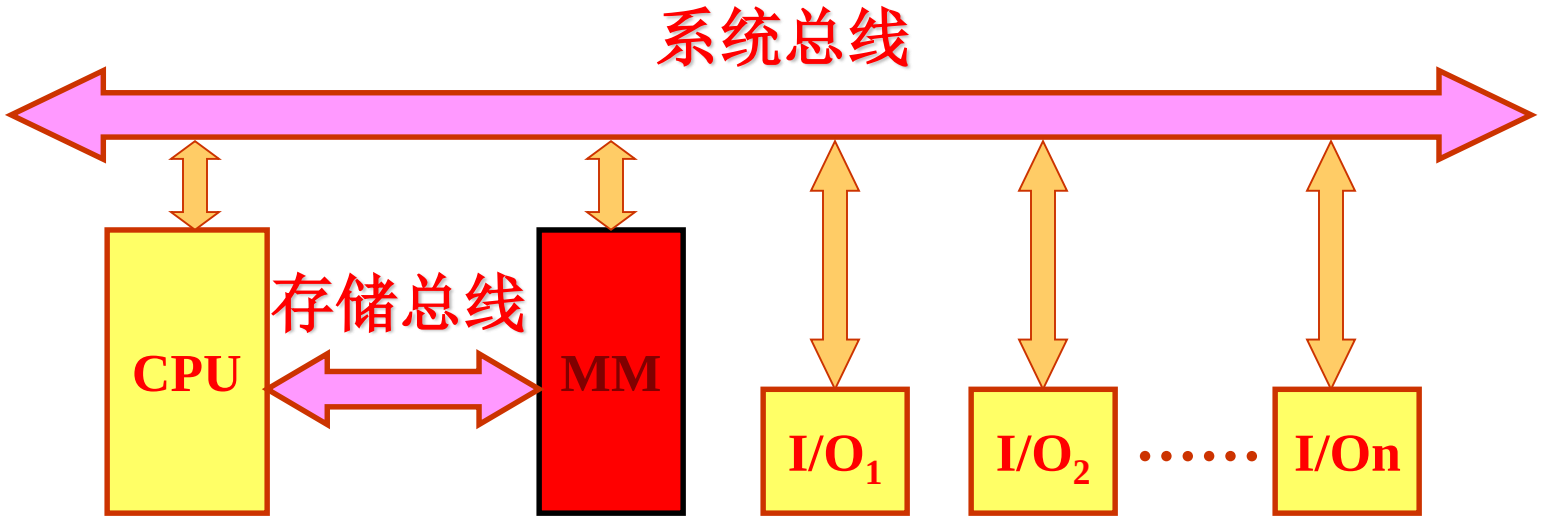
即插即用——指任何扩展卡插入系统便可工作。**EISA、PCI**等具有此功能。

11. 画一个具有双向传输功能的总线逻辑图。

解：此题实际上是要求设计一个双向总线收发器，设计要素为三态、方向、使能等控制功能的实现，可参考**74LS245**等总线缓冲器芯片内部电路。逻辑图如下：（n位）



错误的设计：



这个方案的错误是：

不合题意。按题意要求应画出逻辑线路图而不是逻辑框图。

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；

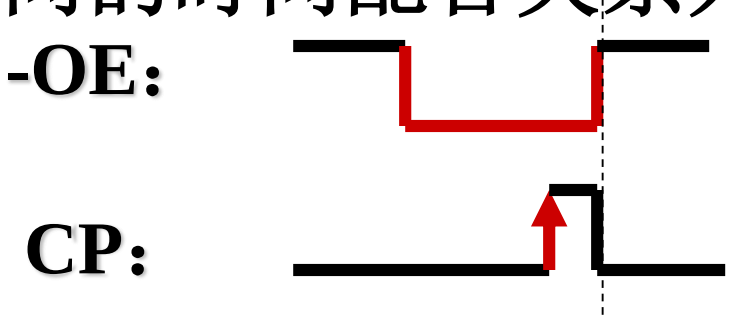
T1时刻完成总线→A；

T2时刻完成A→总线；

T3时刻完成总线→B。

解：

（1）采用三态输出的D型寄存器74LS374做A、B、C、D四个寄存器，其输出可直接挂总线。A、B、C三个寄存器的输入采用同一脉冲打入。注意-OE为电平控制，与打入脉冲间的时间配合关系为：

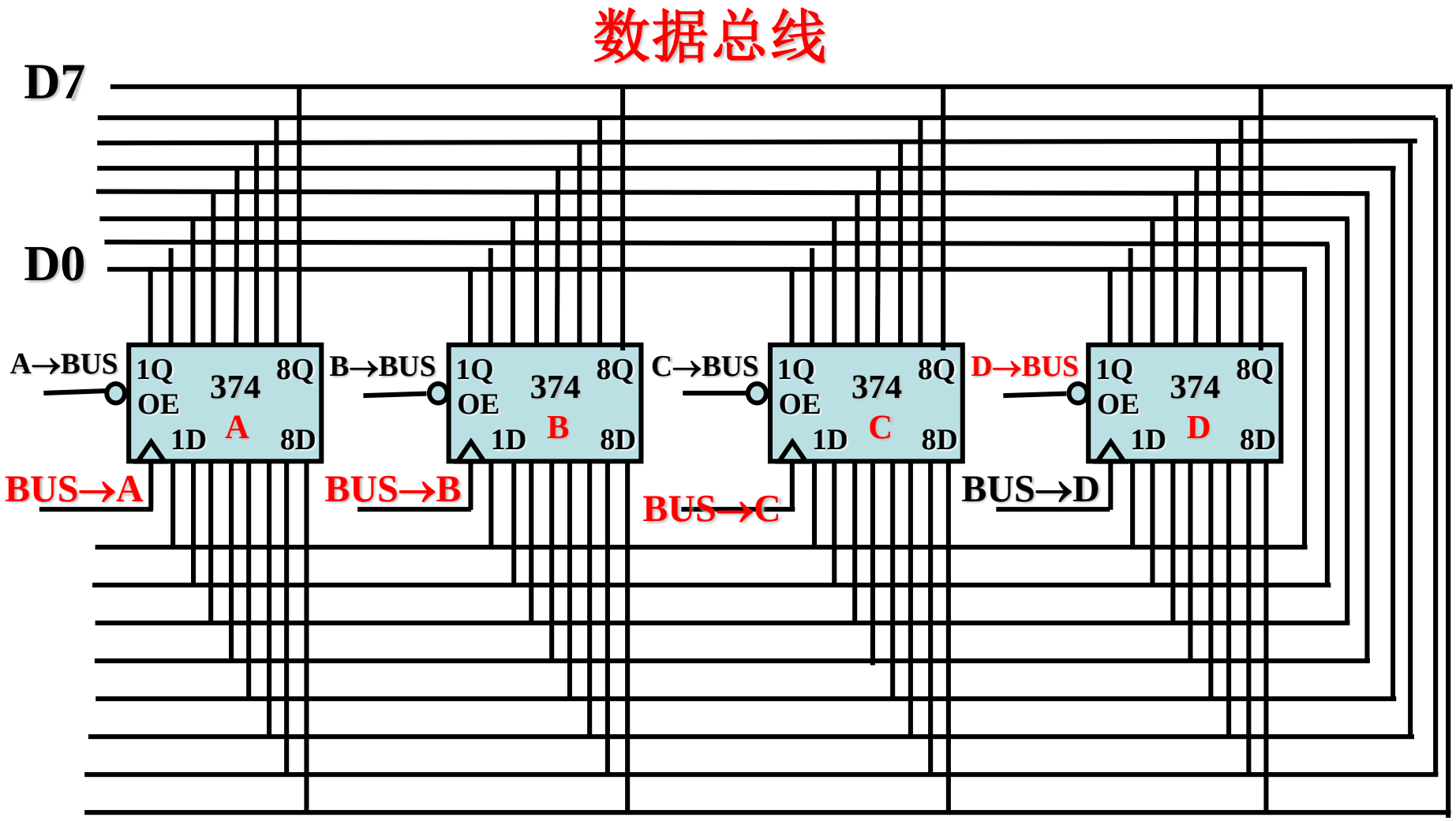


令：BUS→A=BUS→B=BUS→C=CP；

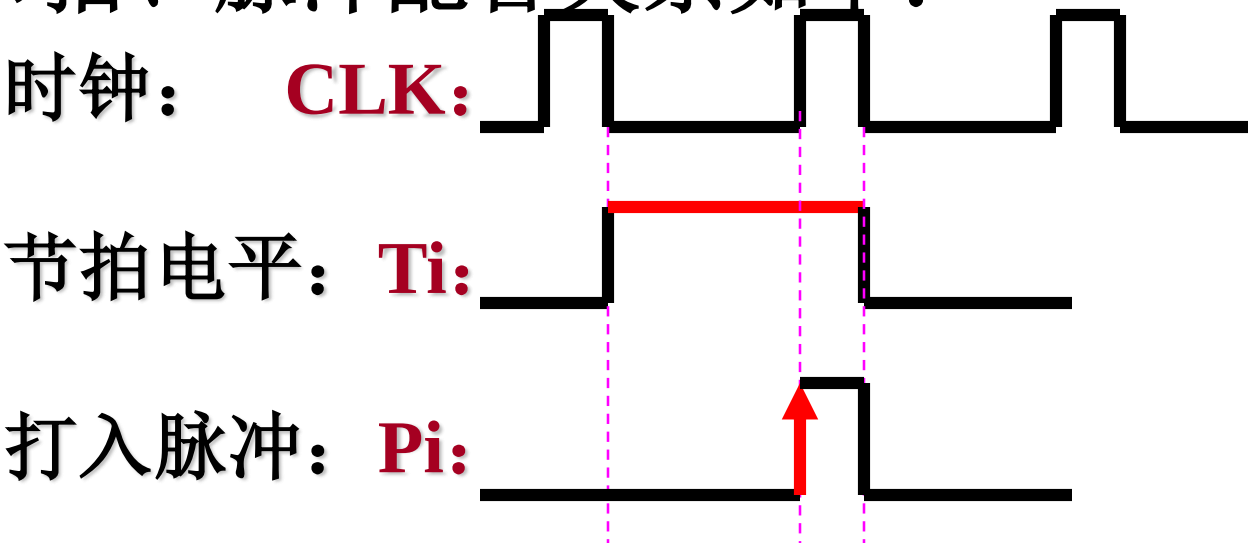
D→BUS= -OE；

当CP前沿到来时，将D→A、B、C。

现以8位总线为例，设计此电路，如下图示：

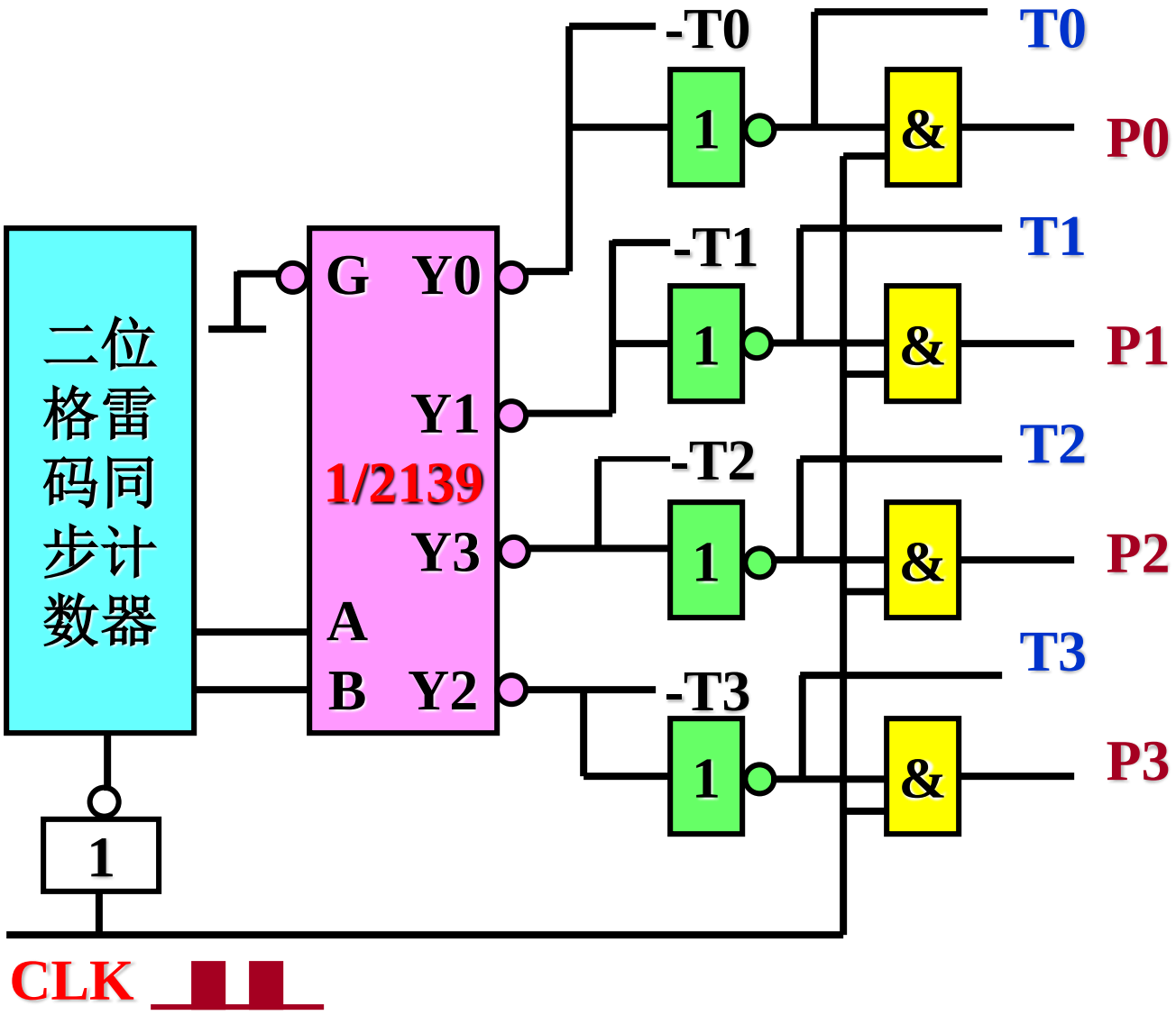


（2）寄存器设置同（1），由于本题中发送、接收不在同一节拍，因此总线需设锁存器缓冲，锁存器采用74LS373（电平使能输入）。节拍、脉冲配合关系如下：

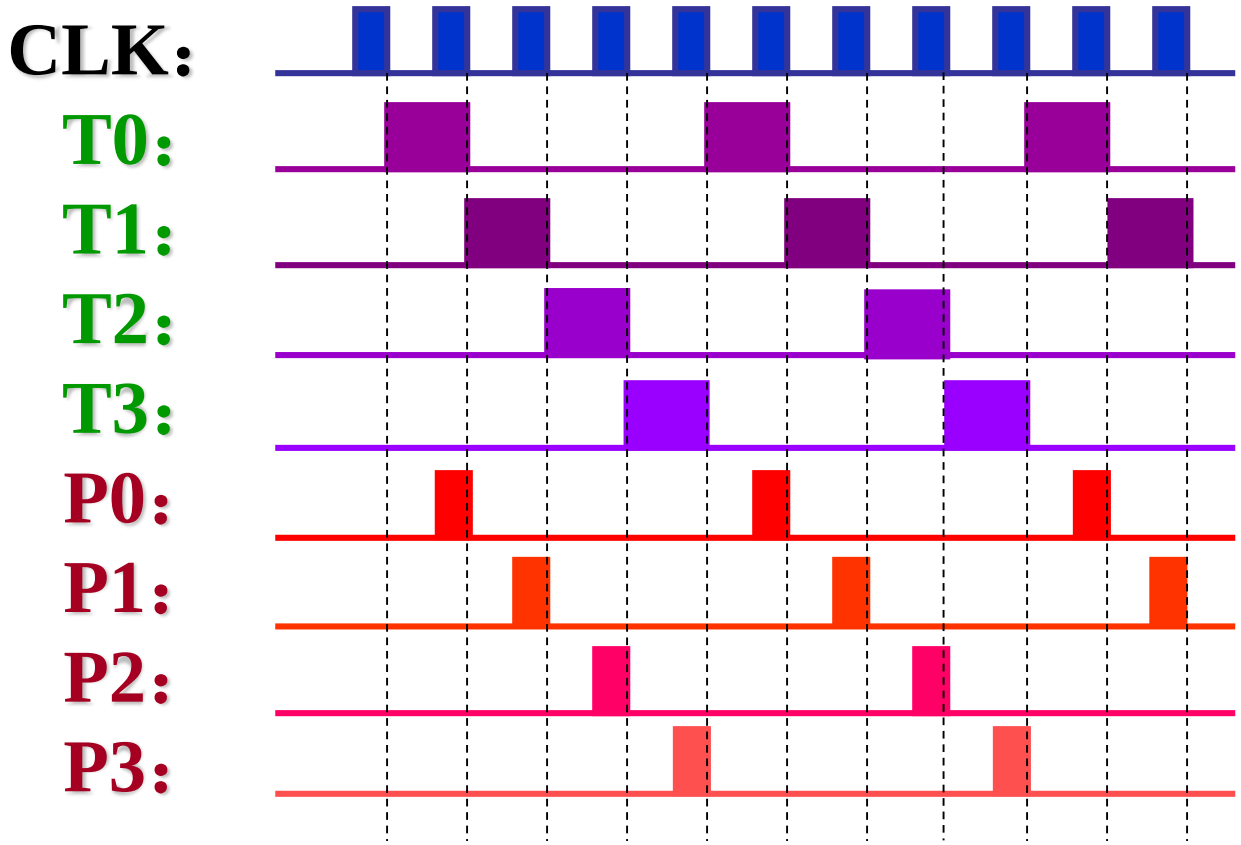


图中，脉冲包在电平中，为了留有较多的传送时间，脉冲设置在靠近电平后沿处。

节拍、脉冲分配逻辑如下：

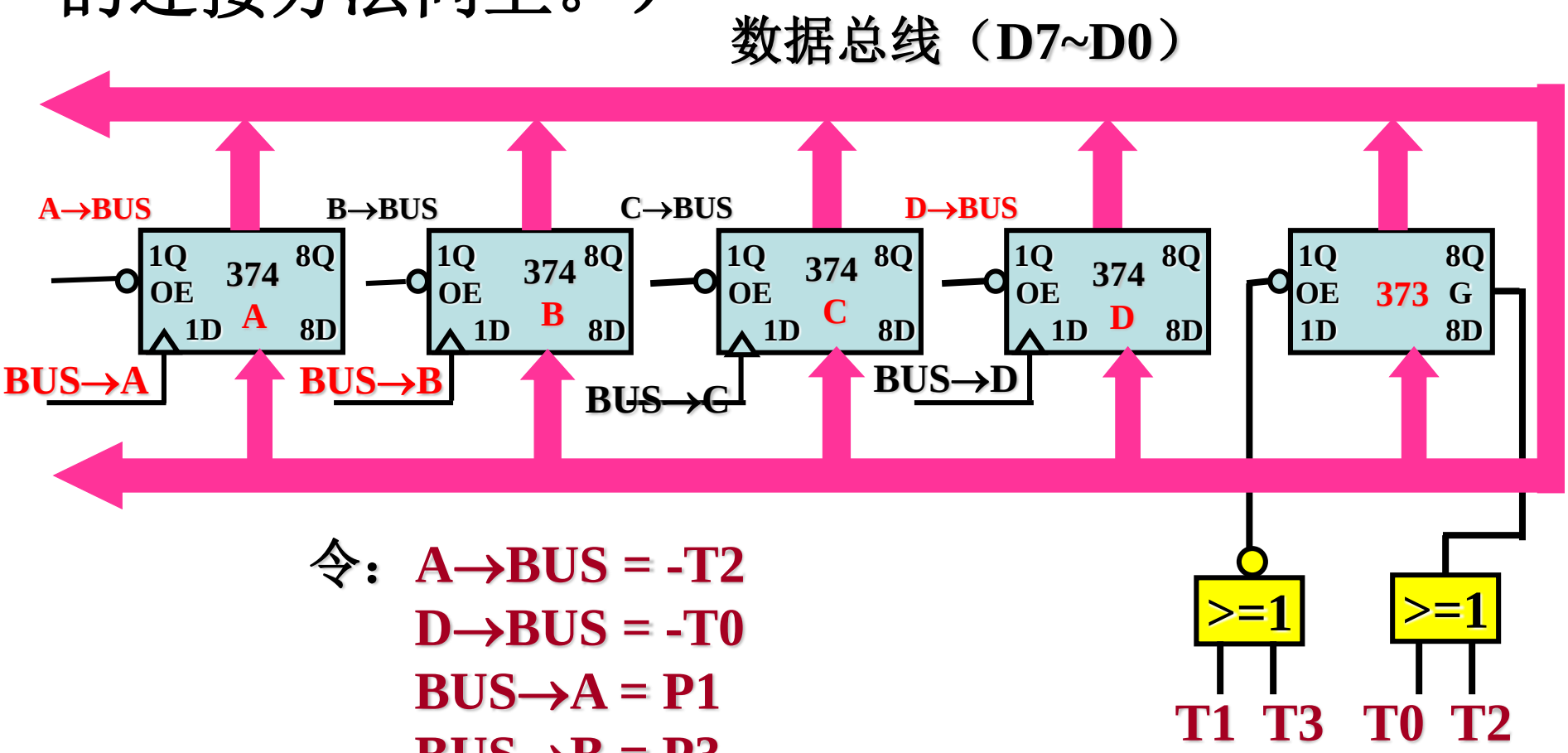

二位格雷码同步计数器
G
A
B
Y0
Y1
Y3
Y2
1/2139
-T0
-T1
-T2
-T3
1
1
1
1
1
&
&
&
&
T0
T1
T2
T3
P0
P1
P2
P3
CLK

节拍、脉冲时序图如下：



以8位总线为例，电路设计如下：

（图中，A、B、C、D四个寄存器与数据总线的连接方法同上。）



令：A→BUS = -T2

D→BUS = -T0

BUS→A = P1

BUS→B = P3

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

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。

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

存 储 器

第 四 章

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—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份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/S = 5M字/秒

注意字长（32位）不是16位。

（注：本题的兆单位来自时间=10^6）

6. 某机字长为32位，其存储容量是64KB，按字编址其寻址范围是多少？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64KB，则：

按字寻址范围 = 64K×8 / 32=16K字

按字节编址时的主存地址分配图如下：

字地址	HB ——字节地址—— LB			
0	0	1	2	3
4	4	5	6	7
8				
……	……	……	……	……
65528				
65532	65532	65533	65534	65535

讨论：

1、 在按字节编址的前提下，按字寻址时，地址仍为16位，即地址编码范围仍为0~64K-1，但字空间为16K字，字地址不连续。

2、 字寻址的单位为字，*不是B（字节）*。

3、 画存储空间分配图时要画出上限。

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；

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 = 32片

4K×8：16K×32 /4K×8 = 4 × 4 = 16片

8K×8：16K×32 / 8K × 8 = 2X4 = 8片

讨论：

地址线根数与容量为2的幂的关系，在此为2^{14}，14根；

数据线根数与字长位数相等，在此为32根。（注：不是2的幂的关系。）

×：$32=2^5$，5根

8. 试比较静态RAM和动态RAM。

答：静态RAM和动态RAM的比较见下表：

特性	SRAM	DRAM
存储信息	触发器	电容
破坏性读出	非	是
需要刷新	不要	需要
送行列地址	同时送	分两次送
运行速度	快	慢
集成度	低	高
发热量	大	小
存储成本	高	低
功耗	高	低
可靠性	高	低
可用性	使用方便	不方便
适用场合	高速小容量存储器	大容量主存

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；

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、分散刷新及异步刷新三种方式的刷新间隔各为多少？

注：该题题意不太明确。实际上，只有异步刷新需要计算刷新间隔。

解：设DRAM的刷新最大间隔时间为2ms，则

异步刷新的刷新间隔 =2ms/256行

=0.0078125ms =7.8125μs

即：每7.8125μs刷新一行。

集中刷新时，

刷新最晚启动时间=2ms-0.1μs×256行

=2ms-25.6μs=1974.4μs

集中刷新启动后，

刷新间隔 = 0.1μs

即：每0.1μs刷新一行。

集中刷新的死时间 =0.1μs×256行

=25.6μs

分散刷新的刷新间隔 =0.1μs×2

=0.2μs

即：每0.2μs刷新一行。

分散刷新一遍的时间

=0.1μs×2×256行 =51.2μs　则

分散刷新时，

2ms内可重复刷新遍数

=2ms/ 51.2μs ≈39遍

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？

（注：将存储器分成若干个容量相等的区域，每一个区域可看做一个页面。）

解：设采用SRAM芯片，

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位 / 4

= 16K × 8位；

组容量 = 页面容量 / 组数

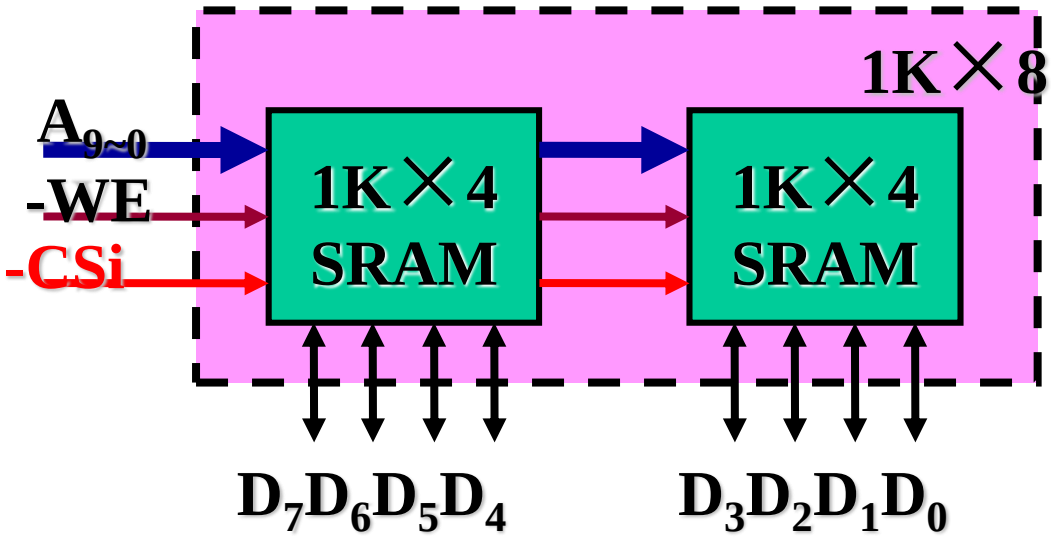
= 16K × 8位 / 16 = 1K × 8位；

组内片数 = 组容量 / 片容量

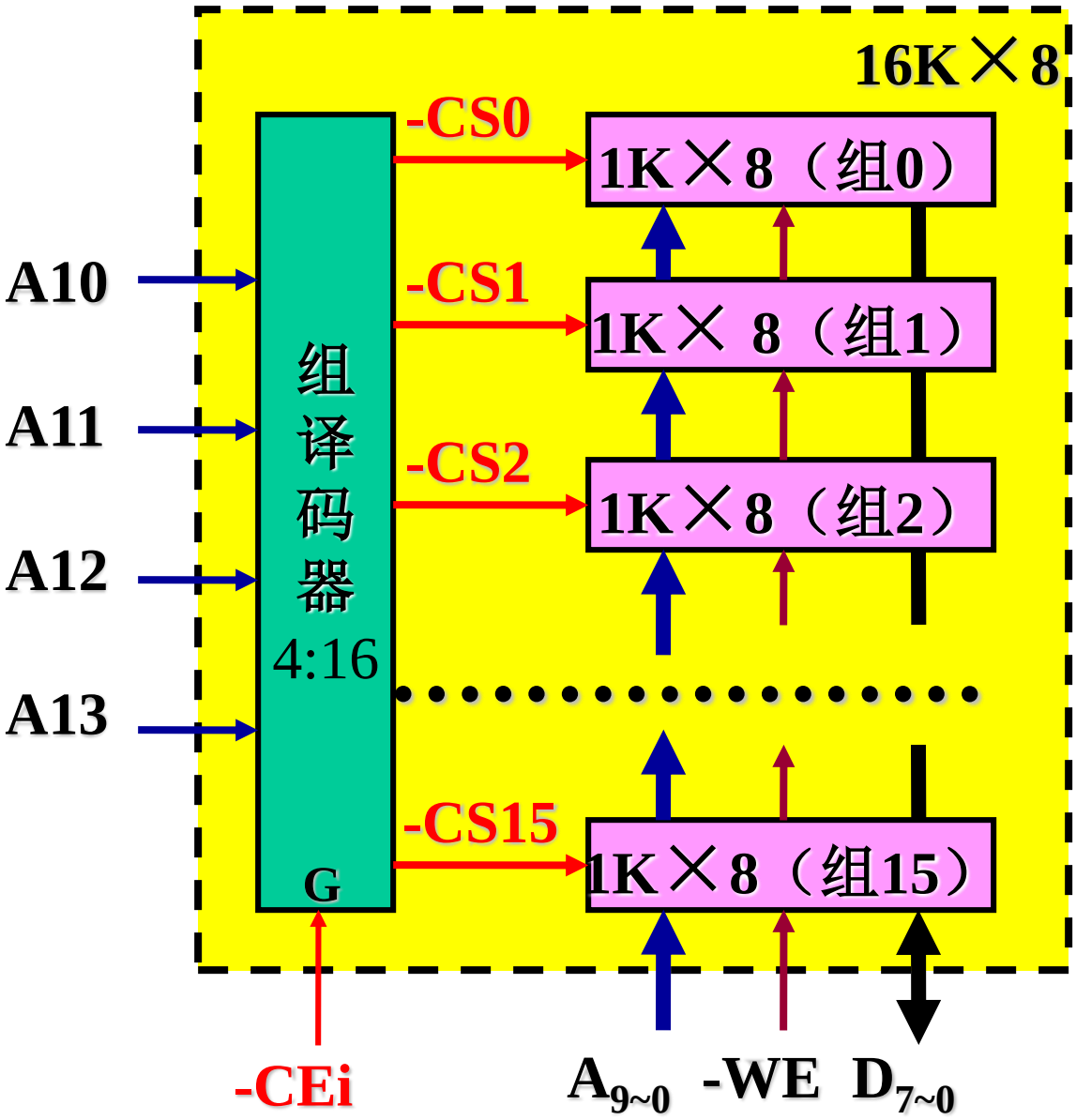
= 1K×8位 / 1K×4位 = 2片

地址分配：

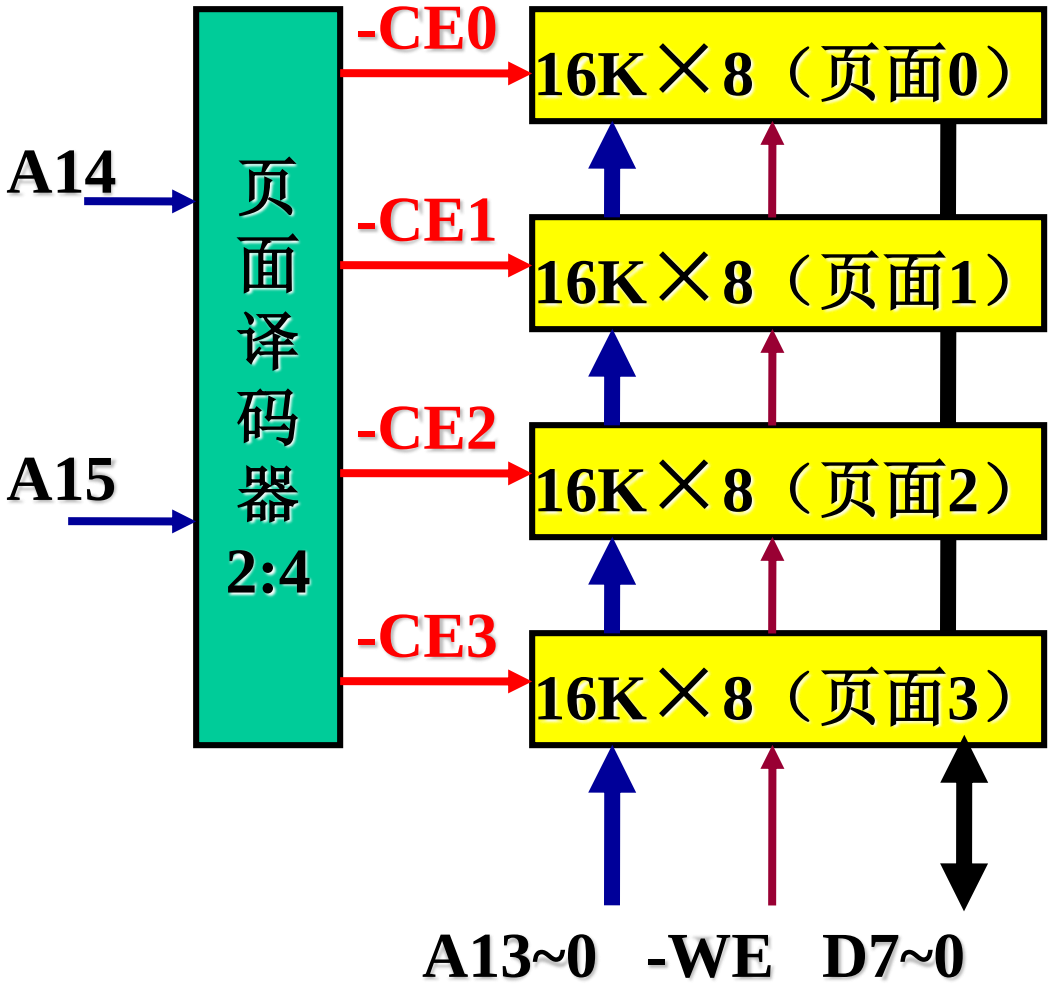
组逻辑图如下：（位扩展）



页面逻辑框图：（字扩展）


16K×8
-CS0
1K×8（组0）
A10
-CS1
1K× 8（组1）
A11
组
译
码
器
4:16
-CS2
1K×8（组2）
A12
A13
-CS15
1K×8（组15）
G
-CEi
A9~0
-WE
D7~0

存储器逻辑框图：（字扩展）


A14
A15
页面译码器2:4
-CE0
-CE1
-CE2
-CE3
16K×8（页面0）
16K×8（页面1）
16K×8（页面2）
16K×8（页面3）
A13~0
-WE
D7~0

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。

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、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：

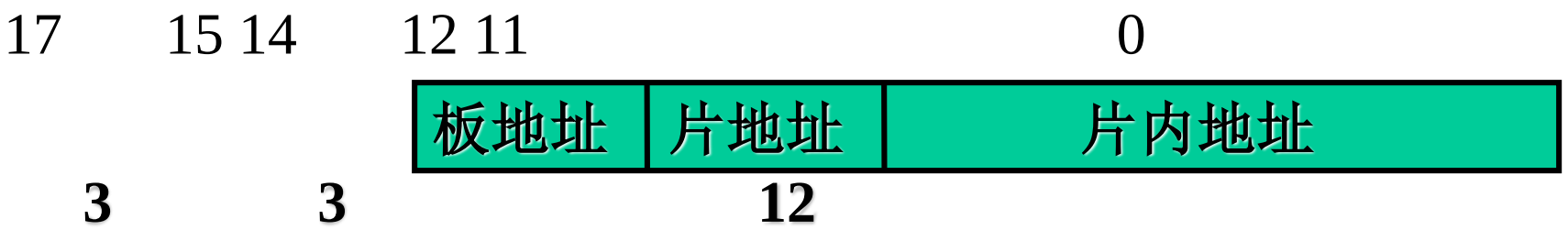
（1）2^{18} = 256K，则该机所允许的最大主存空间是256K×8位（或256KB）；

（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位地址译码选板，次高3位地址译码选片。地址格式分配如下：



15. 设CPU共有16根地址线，8根数据线，并用-MREQ（低电平有效）作访存控制信号，R/-W作读/写命令信号（高电平为读，低电平为写）。现有这些存储芯片：

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的芯片，画出CPU和存储芯片的连接图。要求如下：

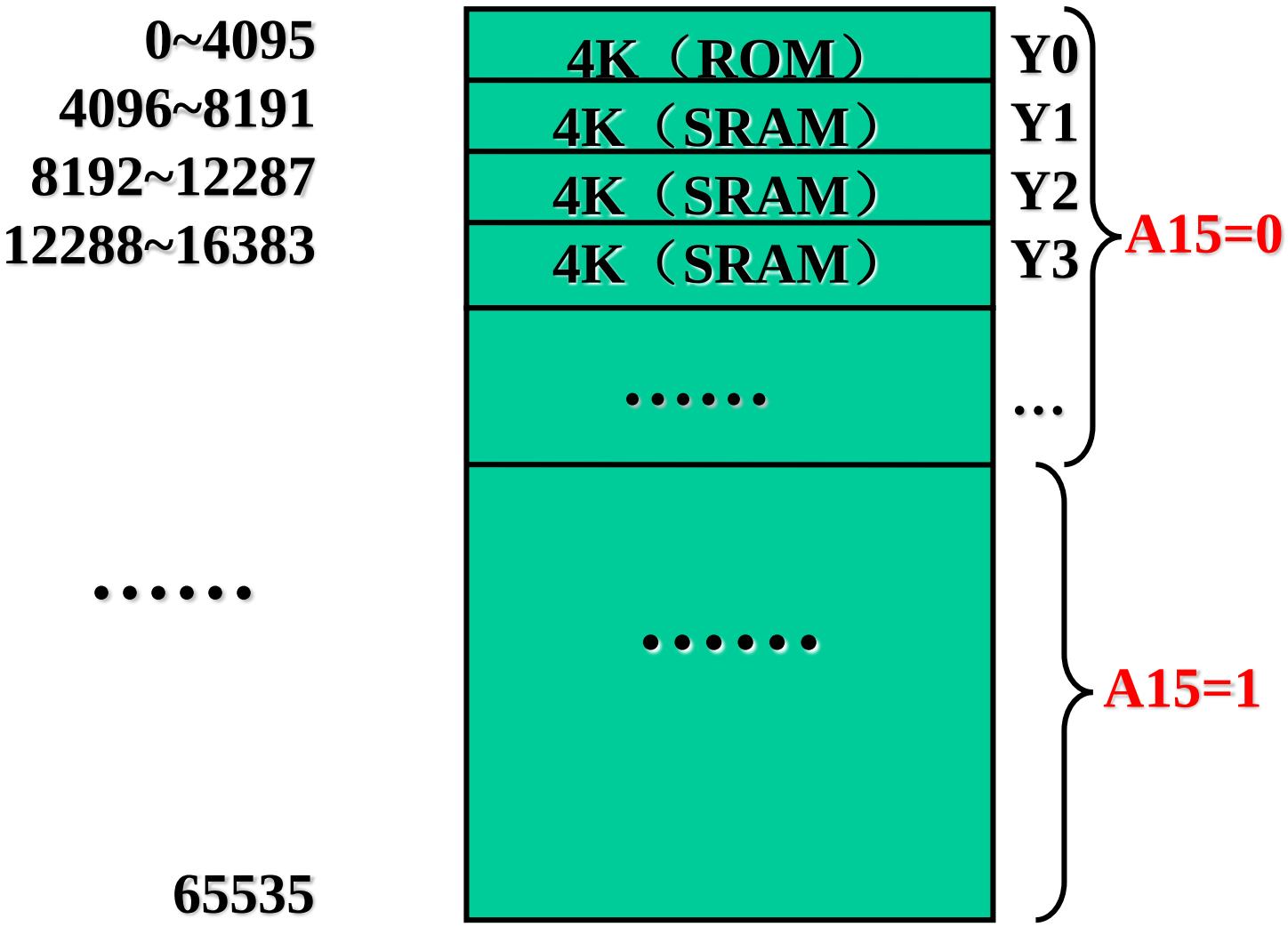
（1）最小4K地址为系统程序区，4096~16383地址范围为用户程序区；

（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：

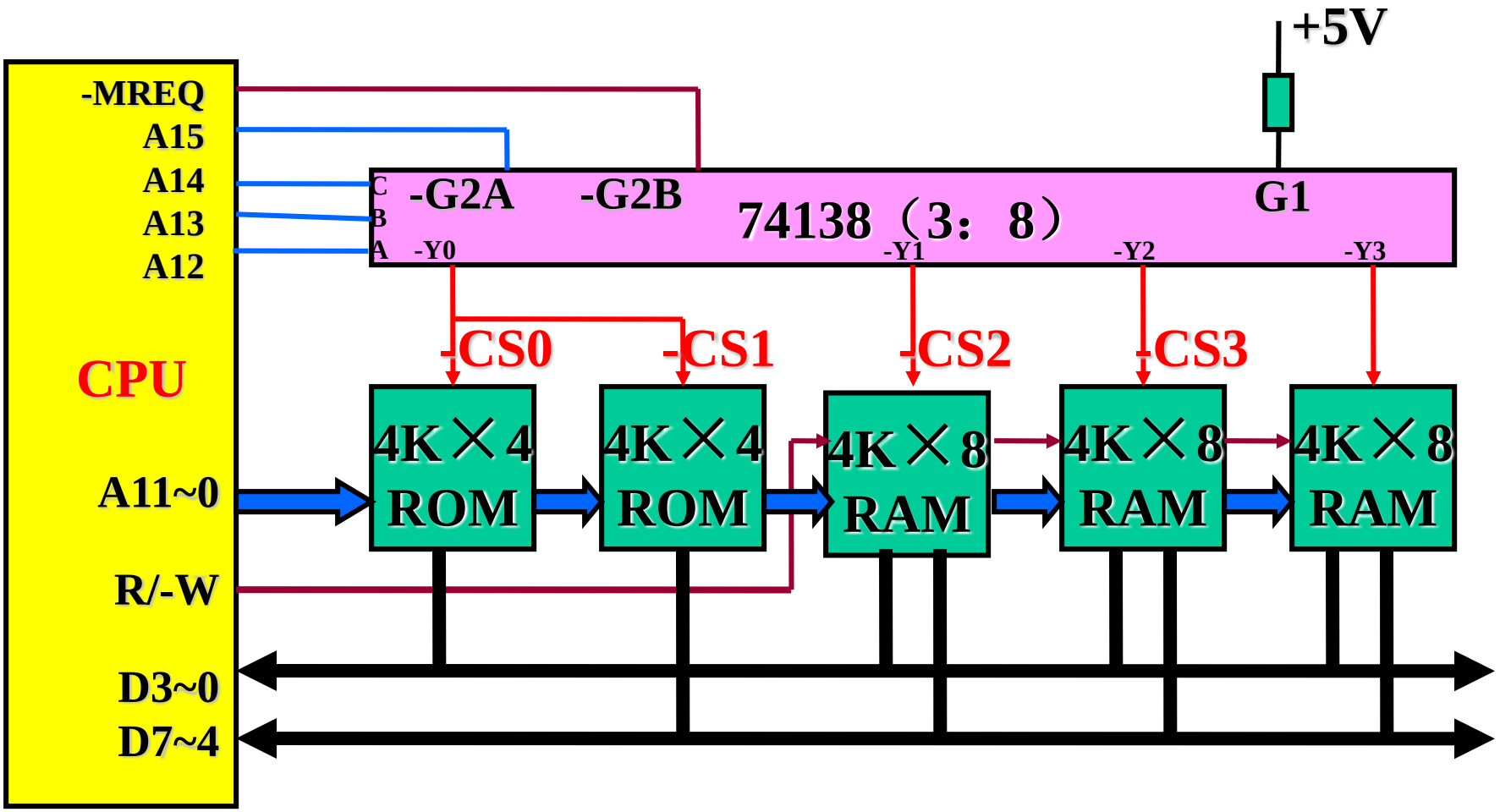
（1）地址空间分配图如下：



（2）选片：ROM：4K × 4位：2片；

RAM：4K × 8位：3片；

（3）CPU和存储器连接逻辑图及片选逻辑：



讨论：

1）选片：当采用字扩展和位扩展所用芯片一样多时，选位扩展。

理由：字扩展需设计片选译码，较麻烦，而位扩展只需将数据线按位引出即可。

本题如选用2K×8 ROM，则RAM也应选2K×8的。否则片选要采用二级译码，实现较麻烦。

当需要RAM、ROM等多种芯片混用时，应尽量选容量等外特性较为一致的芯片，以便于简化连线。

2）应尽可能的避免使用二级译码，以使设计简练。但要注意在需要二级译码时如果不使用，会使选片产生二意性。

3）片选译码器的各输出所选的存储区域是一样大的，因此所选芯片的字容量应一致，如不一致时就要考虑二级译码。

4）其它常见错误：

× EPROM的PD端接地；

(PD为功率下降控制端，当输入为高时，进入功率下降状态。因此PD端的合理接法是与片选端-CS并联。)

× ROM连读/写控制线-WE；

(ROM无读/写控制端)

注：该题缺少“系统程序工作区”条件。

16. CPU假设同上题，现有8片8K×8位的RAM芯片与CPU相连。

（1）用74138译码器画出CPU与存储芯片的连接图；

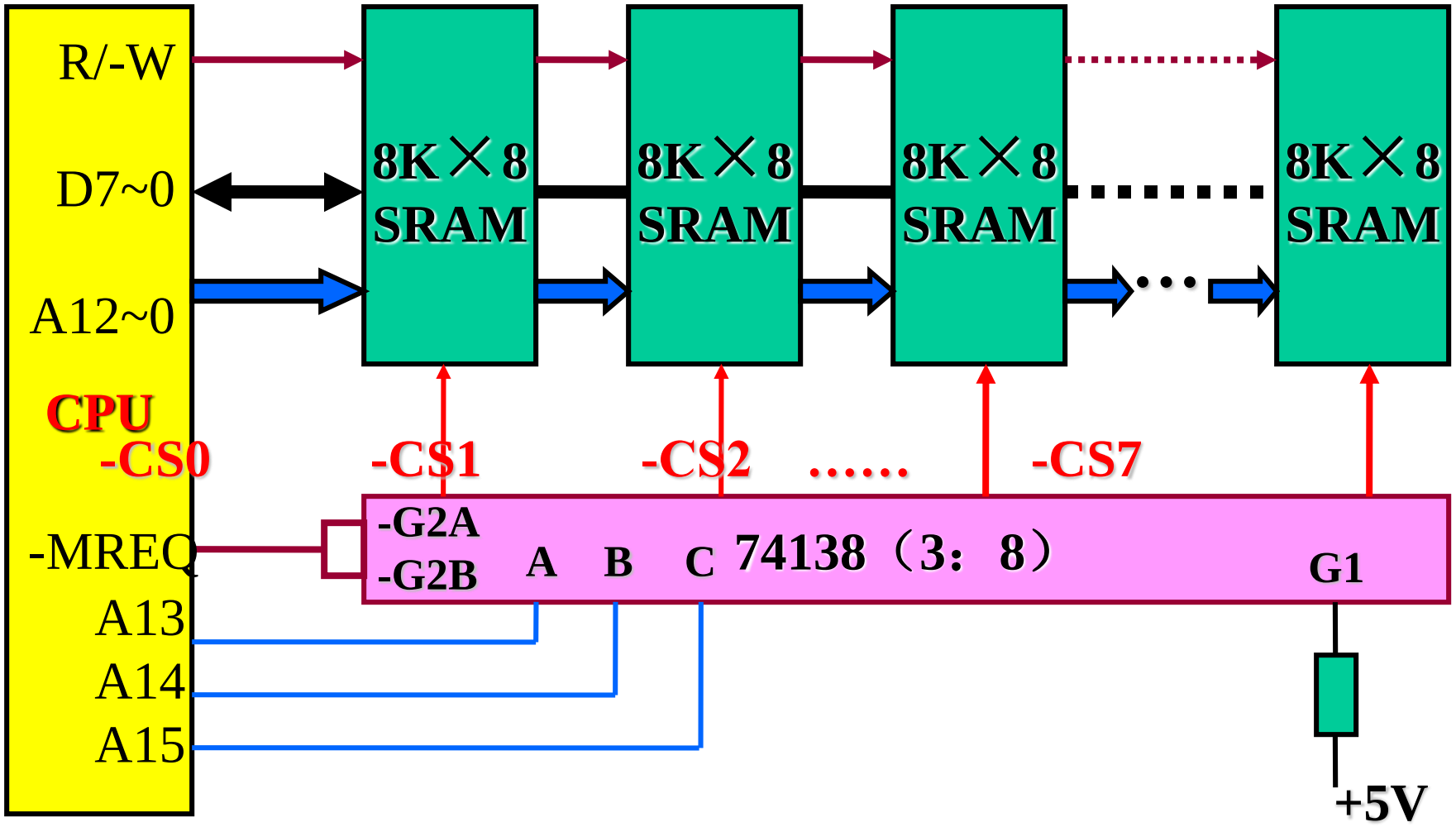
（2）写出每片RAM的地址范围；

（3）如果运行时发现不论往哪片RAM写入数据，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。

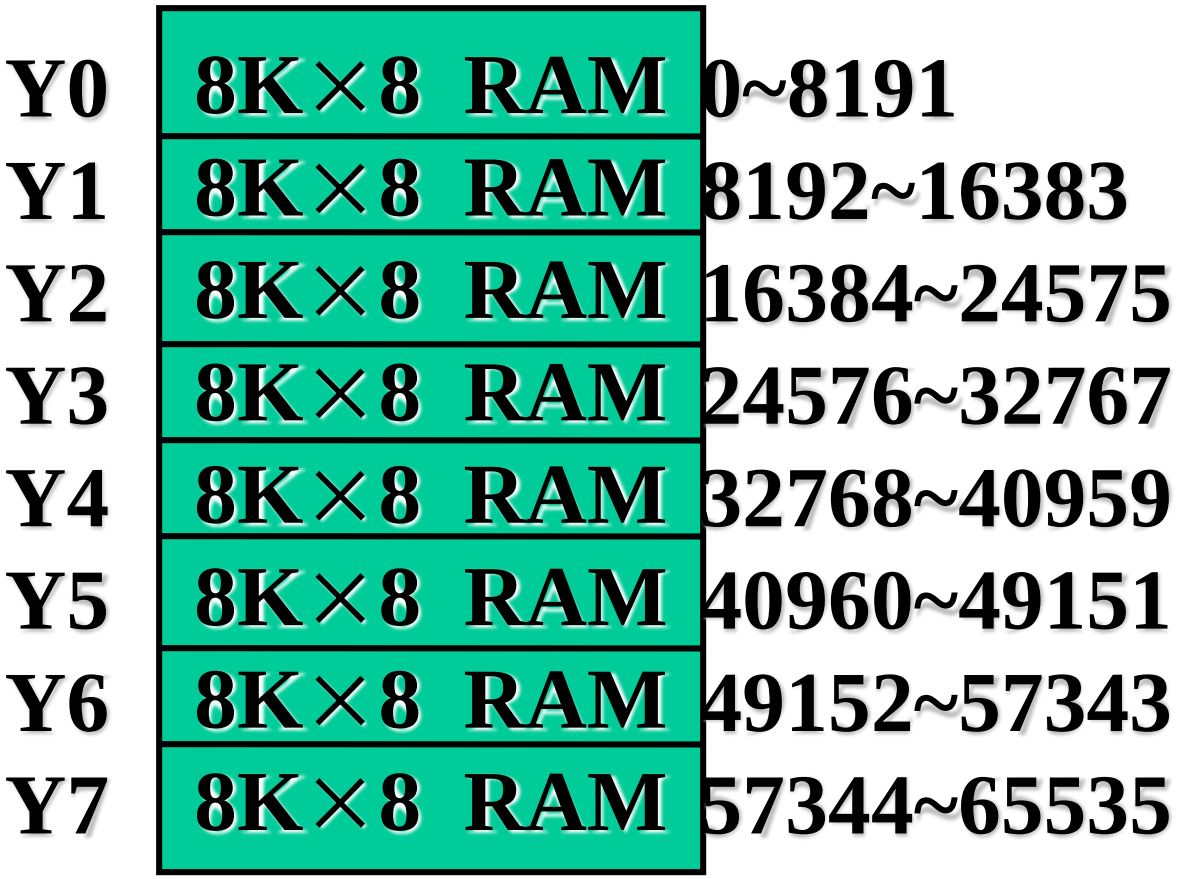
（4）根据（1）的连接图，若出现地址线A13与CPU断线，并搭接到高电平上，将出现什么后果？

解：

（1）CPU与存储器芯片连接逻辑图：



（2）地址空间分配图：


Y0
8K×8 RAM
0~8191
Y1
8K×8 RAM
8192~16383
Y2
8K×8 RAM
16384~24575
Y3
8K×8 RAM
24576~32767
Y4
8K×8 RAM
32768~40959
Y5
8K×8 RAM
40960~49151
Y6
8K×8 RAM
49152~57343
Y7
8K×8 RAM
57344~65535

（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，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。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；

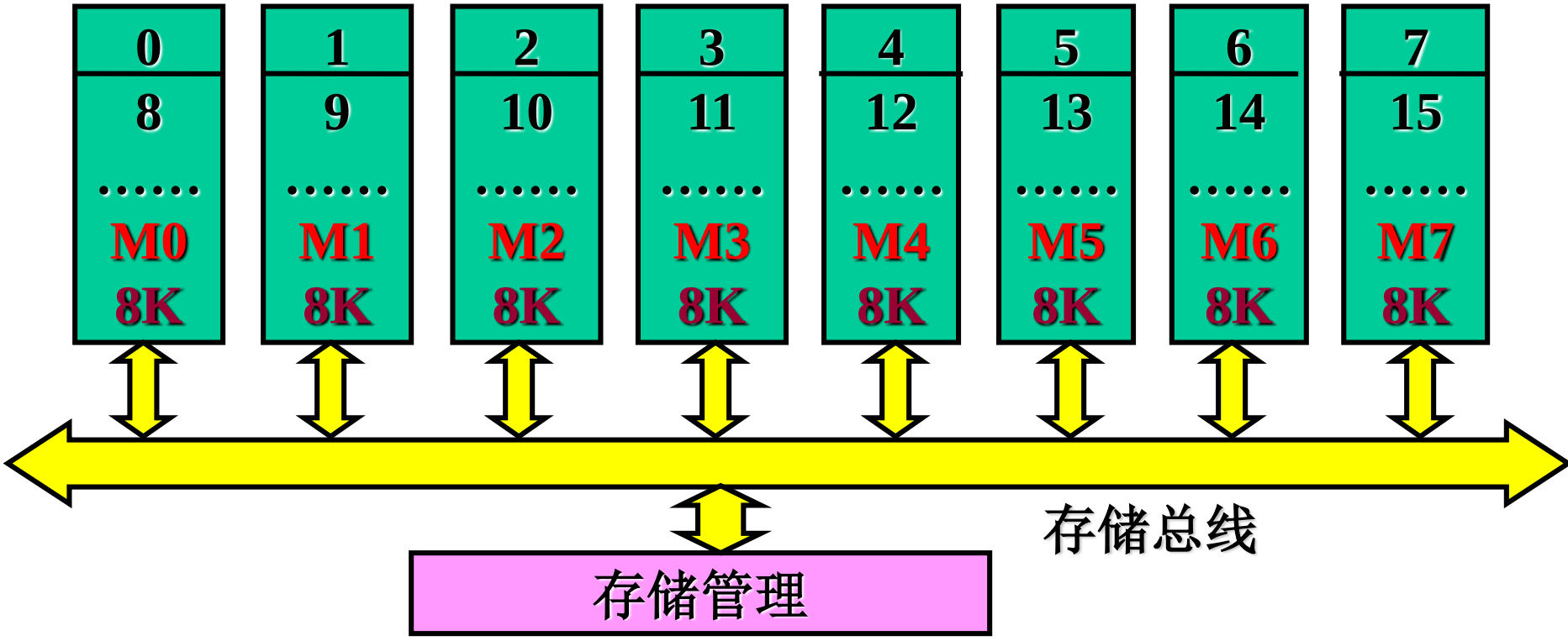
3）该片的-CS端与地线错连或短路；

在此，假设芯片与译码器本身都是好的。

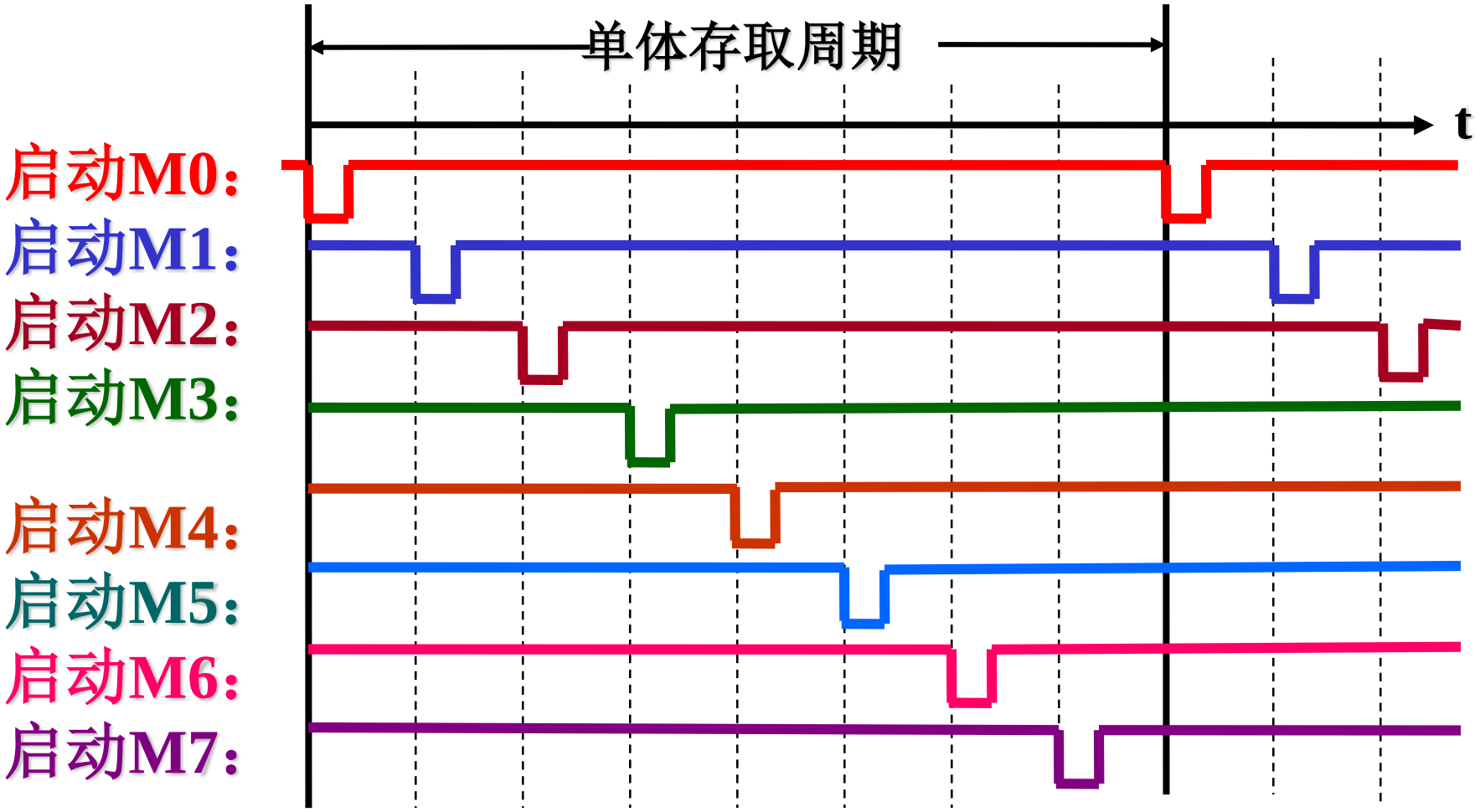
（4）如果地址线A13与CPU断线，并搭接到高电平上，将会出现A13恒为“1”的情况。此时存储器只能寻址A13=1的地址空间，A13=0的另一半地址空间将永远访问不到。若对A13=0的地址空间进行访问，只能错误地访问到A13=1的对应空间中去。

22. 某机字长为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，可采取多体交叉存取技术，图示如下：



8体交叉访问时序：

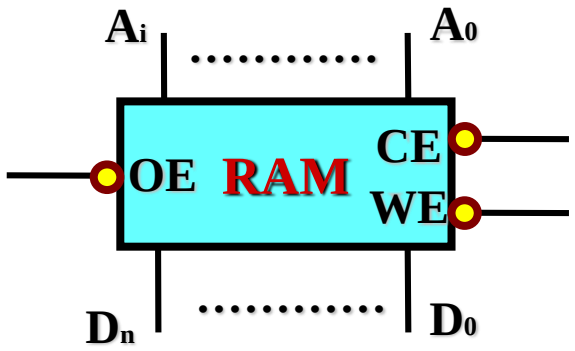


由图可知：每隔1/8个存取周期就可在存储总线上获得一个数据。

23. 设CPU共有16根地址线，8根数据线，并用M/-IO作为访问存储器或I/O的控制信号（高电平为访存，低电平为访I/O），-WR（低电平有效）为写命令，-RD（低电平有效）为读命令。设计一个容量为64KB的采用低位交叉编址的8体并行结构存储器。现有右图所示的存储芯片及138译码器。

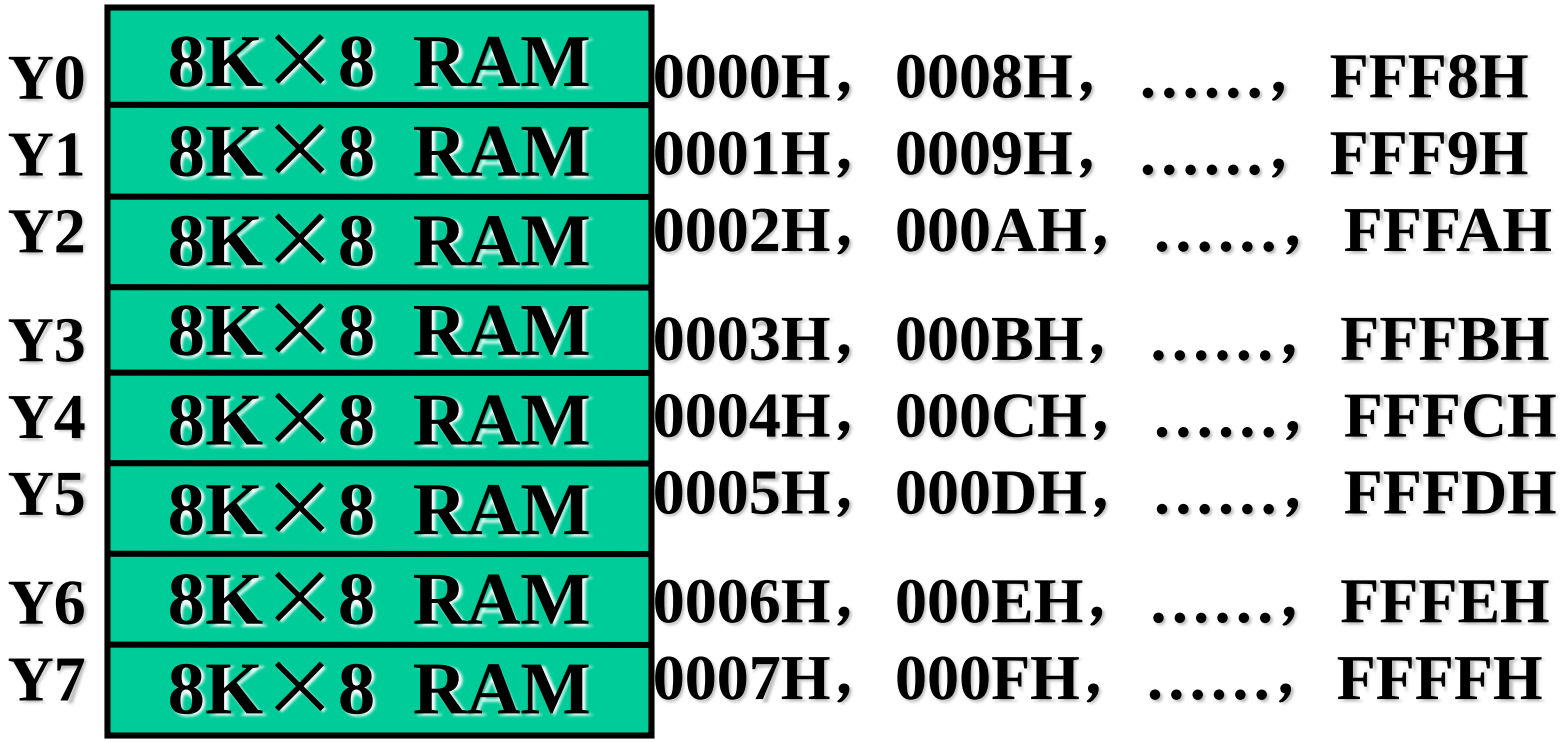
画出CPU和存储芯片（芯片容量自定）的连接图，并写出图中每个存储芯片的地址范围（用十六进制数表示）。

-OE 允许读
-WE 允许写
-CE 片选

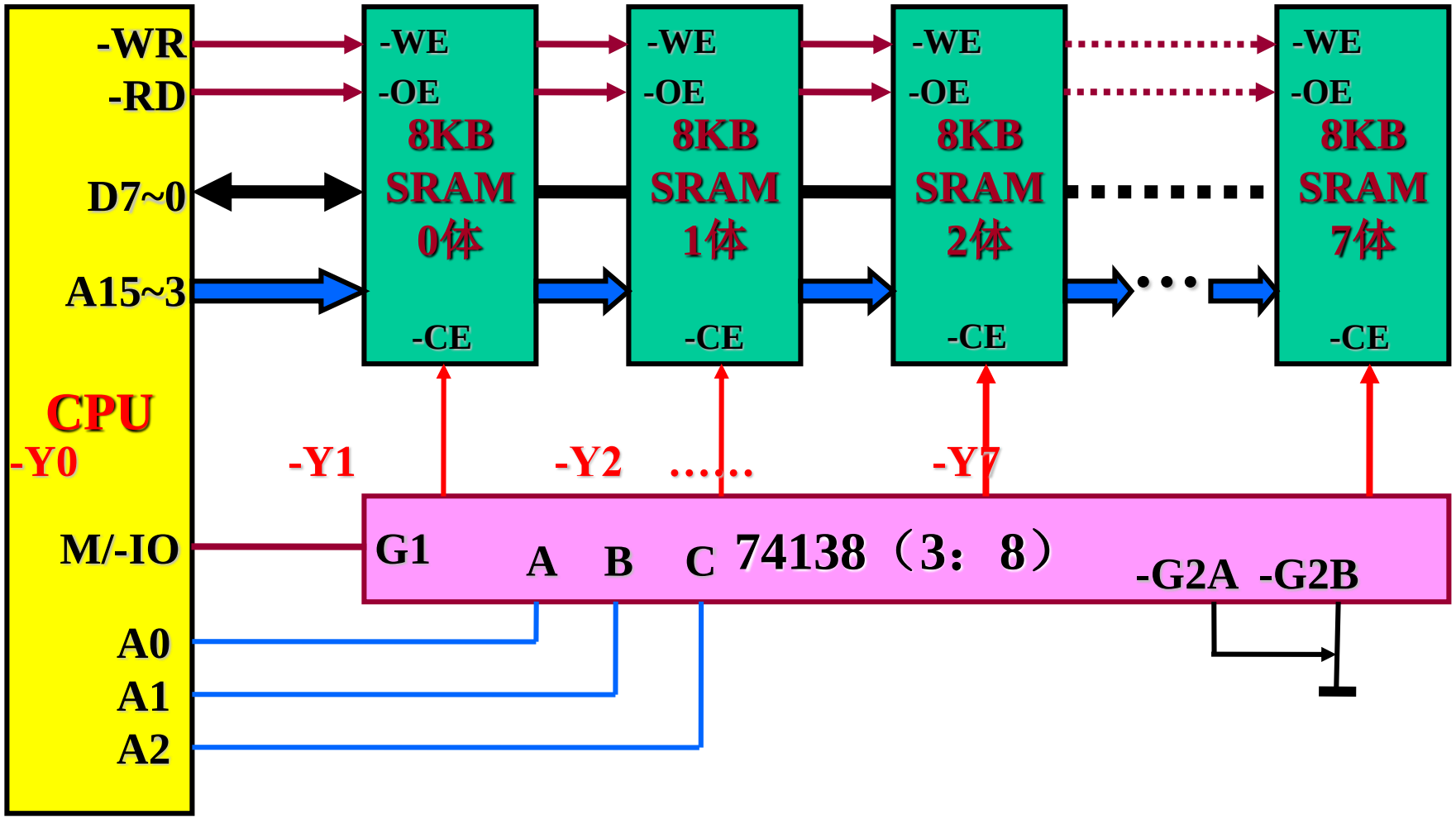


解：芯片容量=64KB/8=8KB

每个芯片（体）的地址范围以8为模低位交叉分布如下：



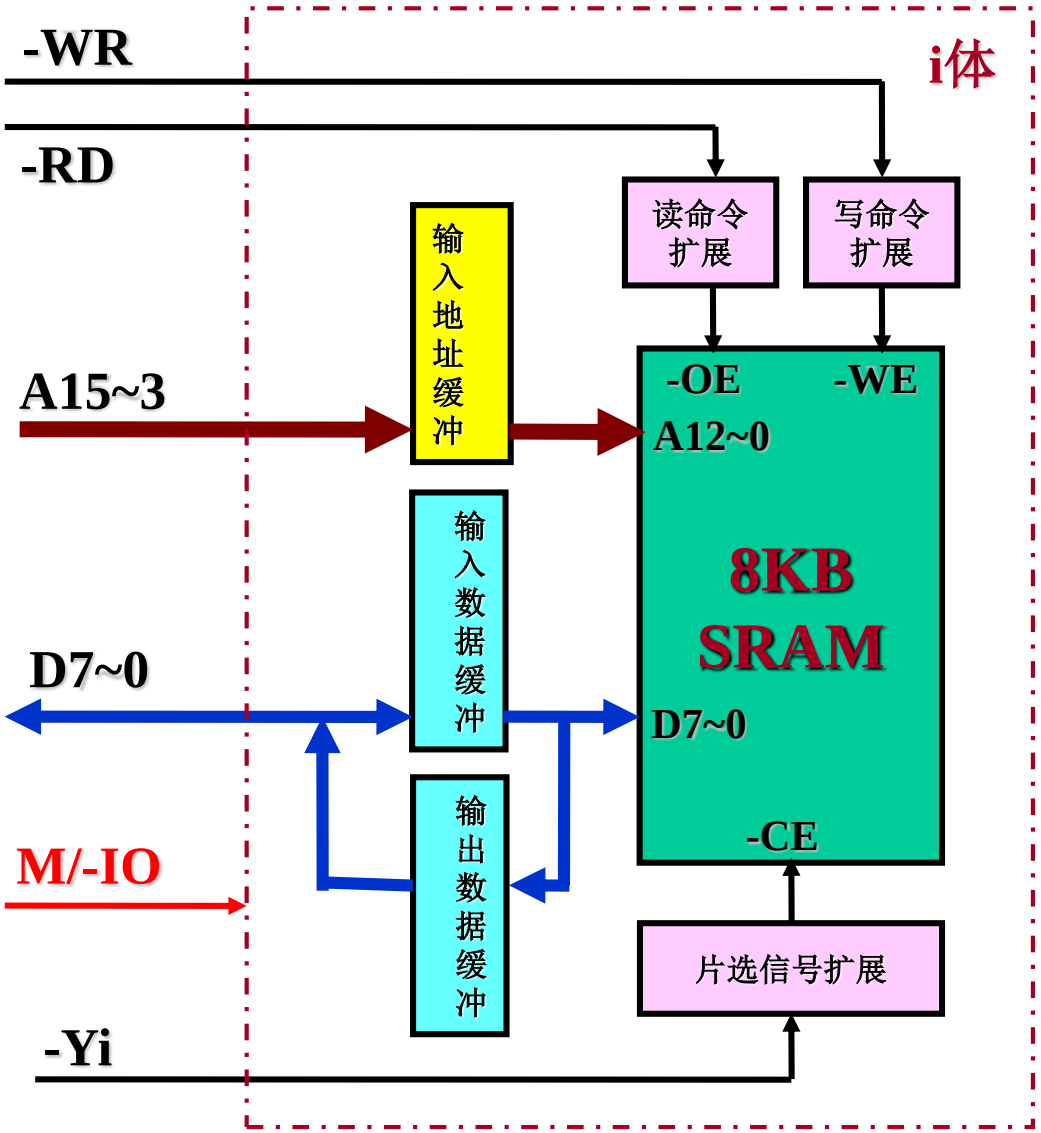
方案1：8体交叉编址的CPU和存储芯片的连接图：



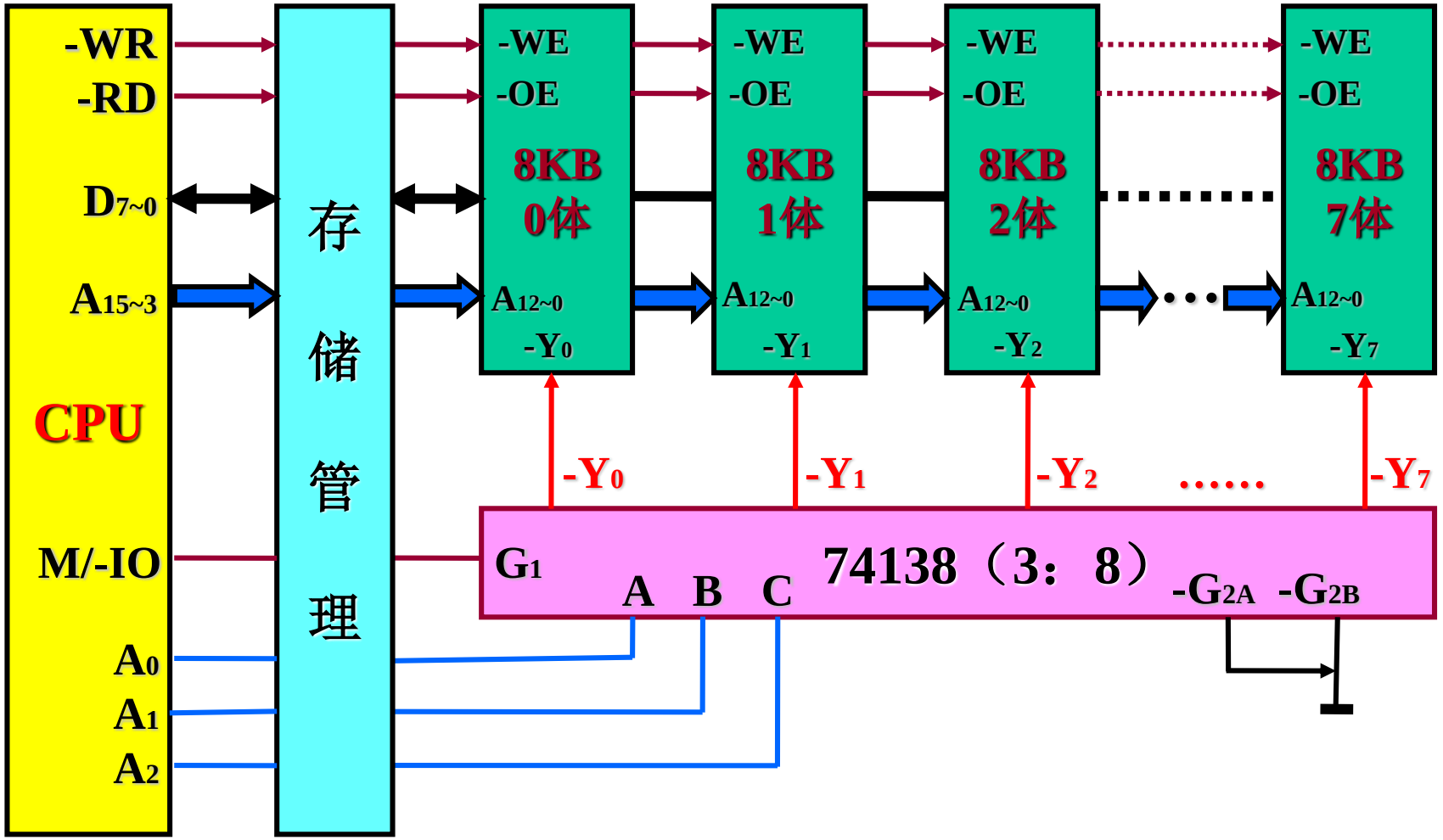
注：此设计方案只能实现八体之间的低位交叉寻址，但不能实现八体并行操作。

方案2：8体交叉并行存取系统体内逻辑如下：

由于存储器单体的存取周期为T，而CPU的总线访存周期为（1/8）T，故体内逻辑要支持单体的独立工作速率。因此在SRAM芯片的外围加了地址、数据的输入/输出缓冲装置，以及控制信号的扩展装置。



CPU和各体的连接图：由于存储器单体的工作速率和总线速率不一致，因此各体之间存在总线分配问题，存储器不能简单地和CPU直接相连，要在存储管理部件的控制下连接。



24. 一个4体低位交叉的存储器，假设存取周期为T，CPU每隔1/4存取周期启动一个存储体，试问依次访问64个字需多少个存取周期？

解：本题中，只有访问第一个字需一个存取周期，从第二个字开始，每隔1/4存取周期即可访问一个字，因此，依次访问64个字需：

存取周期个数 =(64-1)×(1/4)T+T

=（63/4+1）T =15.75+1 =16.75T

与常规存储器的速度相比，加快了：（64-16.75）T =47.25T

注：4体交叉存取虽然从理论上讲可将存取速度提高到4倍，但实现时由于并行存取的分时启动需要一定的时间，故实际上只能提高到接近4倍。

25. 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。

26. 计算机中设置Cache的作用是什么？能不能把Cache的容量扩大，最后取代主存，为什么？

答：计算机中设置Cache主要是为了加速CPU访存速度；

不能把Cache的容量扩大到最后取代主存，主要因为Cache和主存的结构原理以及访问机制不同（主存是按地址访问，Cache是按内容及地址访问）。

27. Cache制作在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）。

补充讨论：

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（主存—L2—L1）。

28. 设主存容量为256K字，Cache容量为2K字，块长为4。

（1）设计Cache地址格式，Cache中可装入多少块数据？

（2）在直接映射方式下，设计主存地址格式。

（3）在四路组相联映射方式下，设计主存地址格式。

（4）在全相联映射方式下，设计主存地址格式。

（5）若存储字长为32位，存储器按字节寻址，写出上述三种映射方式下主存的地址格式。

29. 假设CPU执行某段程序时共访问Cache命中4800次，访问主存200次，已知Cache的存取周期是30ns，主存的存取周期是150ns，求Cache的命中率以及Cache-主存系统的平均访问时间和效率，试问该系统的性能提高了多少？

30. 一个组相联映射的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Cache的地址各为几位？画出主存的地址格式。

31. 设主存容量为1MB，采用直接映射方式的Cache容量为16KB，块长为4，每字32位。试问主存地址为ABCDEH的存储单元在Cache中的什么位置？

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射（即Cache每组内共有4个字块）的Cache组织。

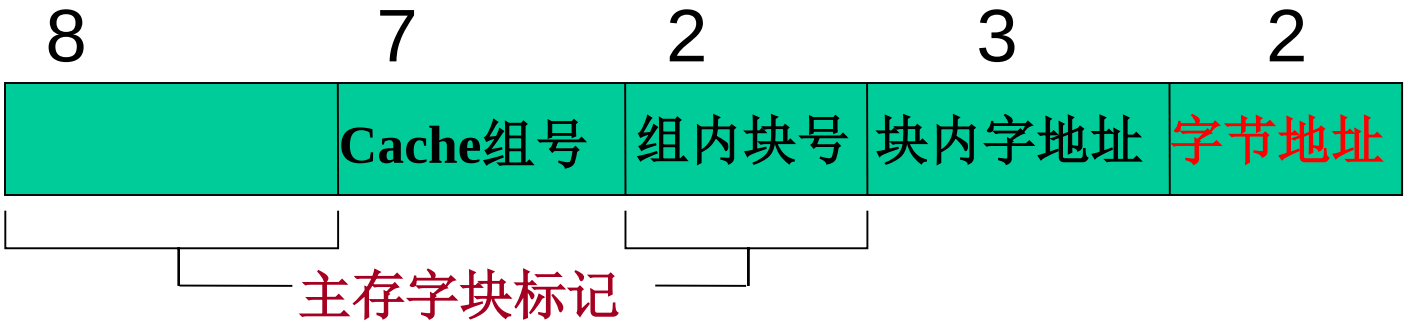
（1）画出主存地址字段中各段的位数；

（2）设Cache的初态为空，CPU依次从主存第0、1、2……89号单元读出90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，速度约提高多少倍？

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：



（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0号块），主存读8号字单元时，将1号块（8~15）调入Cache（1组0号块）…… 主存读89号单元时，将11号块（88~89）调入Cache（11组0号块）。

共需调90/8 ≈12次，就把主存中的90个字调入Cache。除读第1遍时CPU需访问主存12次外，以后重复读时不需再访问主存。则在90×8 =720个读操作中：

访Cache次数 =（90-12）+630 =708次

Cache命中率 =708/720 ≈0.98 ≈98%

（3）设无Cache时访主存需时720T（T为主存周期），加入Cache后需时：

708×T/6+12T =（118+12）T

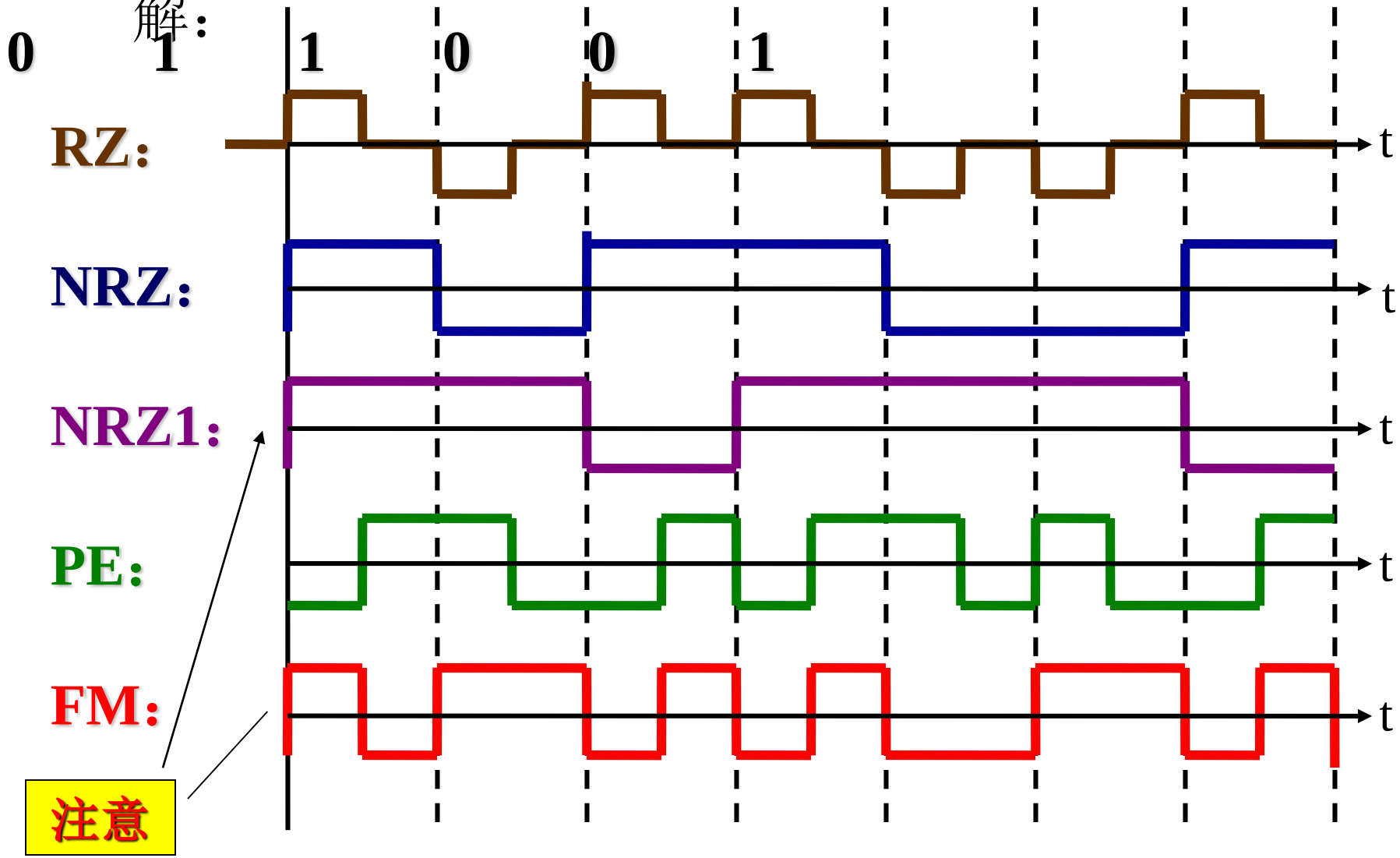
=130T

则：720T/130T ≈5.54倍

有Cache和无Cache相比，速度提高了4.54倍左右。

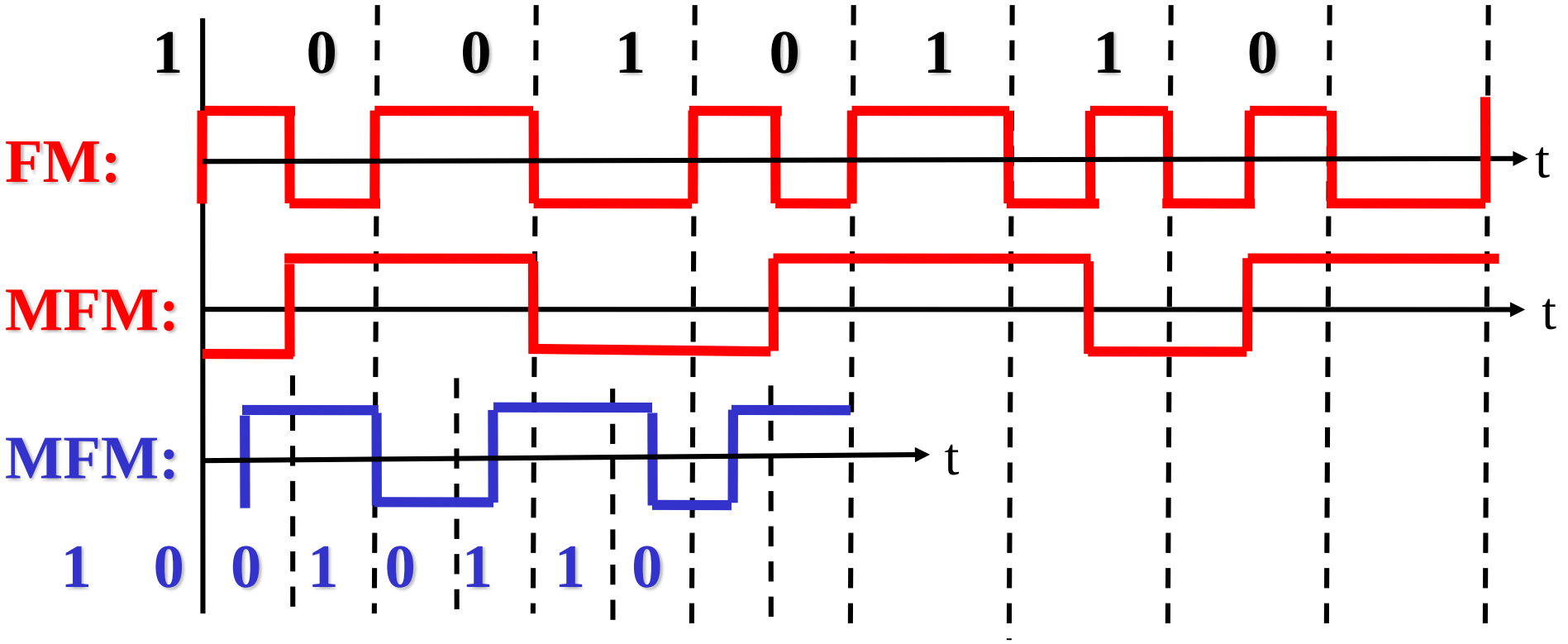
35. 画出RZ、NRZ、NRZ1、PE、FM写入数字串1011001的写电流波形图。

解：



36. 以写入1001 0110为例，比较调频制和改进调频制的写电流波形图。

解：写电流波形图如下：



频率提高一倍后的MFM制。

比较：

1）FM和MFM写电流在位周期中心处的变化规则相同；

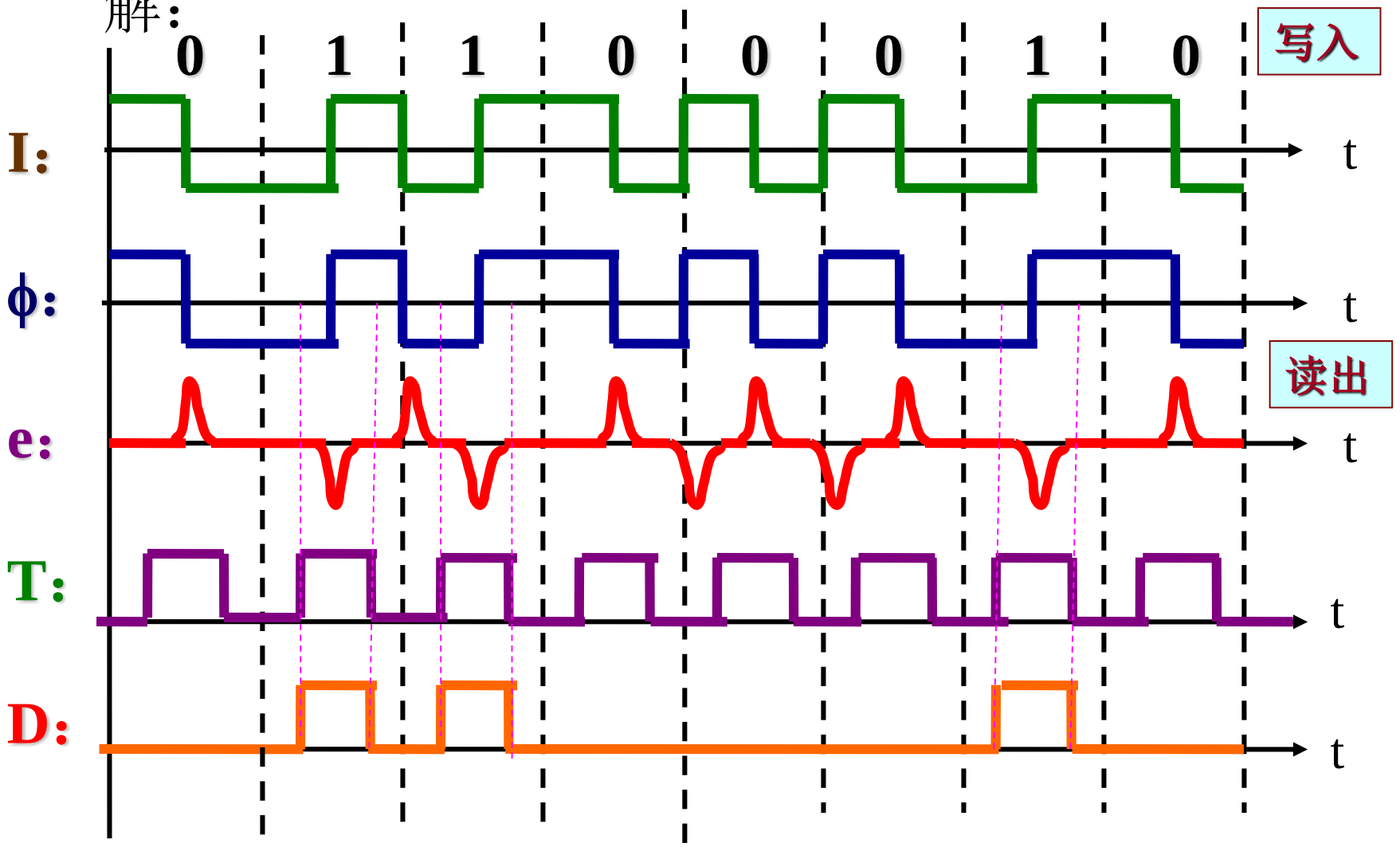
2）MFM制除连续一串“0”时两个0周期交界处电流仍变化外，基本取消了位周期起始处的电流变化；

3）FM制记录一位二进制代码最多两次磁翻转，MFM制记录一位二进制代码最多一次磁翻转，因此MFM制的记录密度可提高一倍。上图中示出了在MFM制时位周期时间缩短一倍的情况。由图可知，当MFM制记录密度提高一倍时，其写电流频率与FM制的写电流频率相当；

4）由于MFM制并不是每个位周期都有电流变化，故自同步脉冲的分离需依据相邻两个位周期的读出信息产生，自同步技术比FM制复杂得多。

37. 画出调相制记录01100010的驱动电流、记录磁通、感应电势、同步脉冲及读出代码等几种波形。

解：



注意：

1）画波形图时应严格对准各种信号的时间关系。

2）读出感应信号不是方波而是与磁翻转边沿对应的尖脉冲；

3）同步脉冲的出现时间应能“包裹”要选的读出感应信号，才能保证选通有效的读出数据信号，并屏蔽掉无用的感应信号。PE记录方式的同步脉冲应安排对准代码周期的中间。

4）最后读出的数据代码应与写入代码一致。

38. 磁盘组有6片磁盘，最外两侧盘面可以记录，存储区域内径22cm，外径33cm，道密度为40道/cm，内层密度为400位/cm，转速3600转/分。

（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）共有：6×2 = 12个存储面可用；

（2）有效存储区域 =（33-22）/2

= 5.5cm

柱面数 = 40道/cm × 5.5cm= 220道

（3）内层道周长=22πcm= 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×12面

= 9，118，560B

（4）转速 = 3600转 / 60秒 = 60转/秒

数据传输率 = 3454B × 60转/秒

= 207，240 B/S

注意：

1）π的精度选取不同将引起答案不同，一般取两位小数；

2）柱面数≠盘组总磁道数（=一个盘面上的磁道数）

3）数据传输率与盘面数无关；

4）数据传输率的单位时间是秒，不是分。

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。

40. 采用定长数据块记录格式的磁盘存储器，直接寻址的最小单位是什么？寻址命令中如何表示磁盘地址？

答：采用定长数据块记录格式，直接寻址的最小单位是一个记录块（数据块），寻址命令中可用如下格式表示磁盘地址：

台号	柱面(磁道)号	盘面(磁头)号	扇区号

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

$M(x)\cdot x^4$ =110 0000

$M(x)\cdot x^4/G(x)$ =110 0000/11011

=100+1100/11011　　R(x) =1100

$M(x)\cdot x^4+R(x)$ =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。

42. 有一个（7，4）码，生成多项式$G(x)=x^3+x+1$，写出代码1001的循环冗余校验码。

解：编码过程如下：

$M(x)=1001$　　$n=4$

$G(x)=x^3+x+1=1011$

$k+1=4$　　$k=3$

$M(x)\cdot x^3=1001\ 000$

$M(x)\cdot x^3/G(x)=1001\ 000/1011$

$=1010+110/1011$　　$R(x)=110$

$M(x)\cdot x^3+R(x)=1001\ 000+110$

$=1001\ 110$ =CRC码

由于码制和生成多项式均与教材上的例题4.15相同，故此（7，4）码的出错模式同表4.6。

输入输出系统

第 五章

补充题：

一、某CRT显示器可显示64种ASCII字符，每帧可显示72字×24排；每个字符字形采用7×8点阵，即横向7点，字间间隔1点，纵向8点，排间间隔6点；帧频50Hz，采取逐行扫描方式。假设不考虑屏幕四边的失真问题，且行回扫和帧回扫均占扫描时间的20%，问：

1）显存容量至少有多大？

2）字符发生器（ROM）容量至少有多大？

3）显存中存放的是那种信息？

4）显存地址与屏幕显示位置如何对应？

5）设置哪些计数器以控制显存访问与屏幕扫描之间的同步？它们的模各是多少？

6）点时钟频率为多少？

解：1）显存最小容量=72×24×8 =1728B

2）ROM最小容量=64×8行×8列

= 512B（含字间隔1点，或512×7位）

3）显存中存放的是ASCII码信息。

4）显存每个地址对应一个字符显示位置，显示位置自左至右，从上到下，分别对应缓存地址由低到高。

5）设置点计数器、字计数器、行计数器、排计数器控制显存访问与屏幕扫描之间的同步。

它们的模计算如下：

点计数器模 = 7+1 = 8

行计数器模 = 8 + 6 = 14

字、排计数器的模不仅与扫描正程时间有关，而且与扫描逆程时间有关，因此计算较为复杂。

列方程：（72+x）× 0.8 = 72

（24+y）× 0.8 = 24

解方程得：x = 18，y = 6，则：

字计数器模 = 72 + 18 = 90

排计数器模 = 24 + 6 = 30

6）点频 = 50Hz × 30排 × 14行 × 90字 × 8点 = 15 120 000Hz

= 15.12MHz

讨论：

1、VRAM、ROM容量应以字或字节为单位；

2、字模点阵在ROM中按行存放，一行占一个存储单元；

3、显存中存放的是ASCII码而不是像素点；

4、计算计数器的模及点频时应考虑回扫时间。

二、有一编码键盘，其键阵列为8行×16列，分别对应128种ASCII码字符，采用硬件扫描方式确认按键信号，问：

1）扫描计数器应为多少位？

2）ROM容量为多大？

3）若行、列号均从0开始编排，则当第5行第7列的键表示字母“F”时，CPU从键盘读入的二进制编码应为多少（设采用奇校验）？

4）参考教材图5.15，画出该键盘的原理性逻辑框图；

5）如果不考虑校验技术，此时ROM是否可省？

解：1）扫描计数器 = 7位

（与键的个数有关）

2）ROM容量 = 128 × 8 = 128B

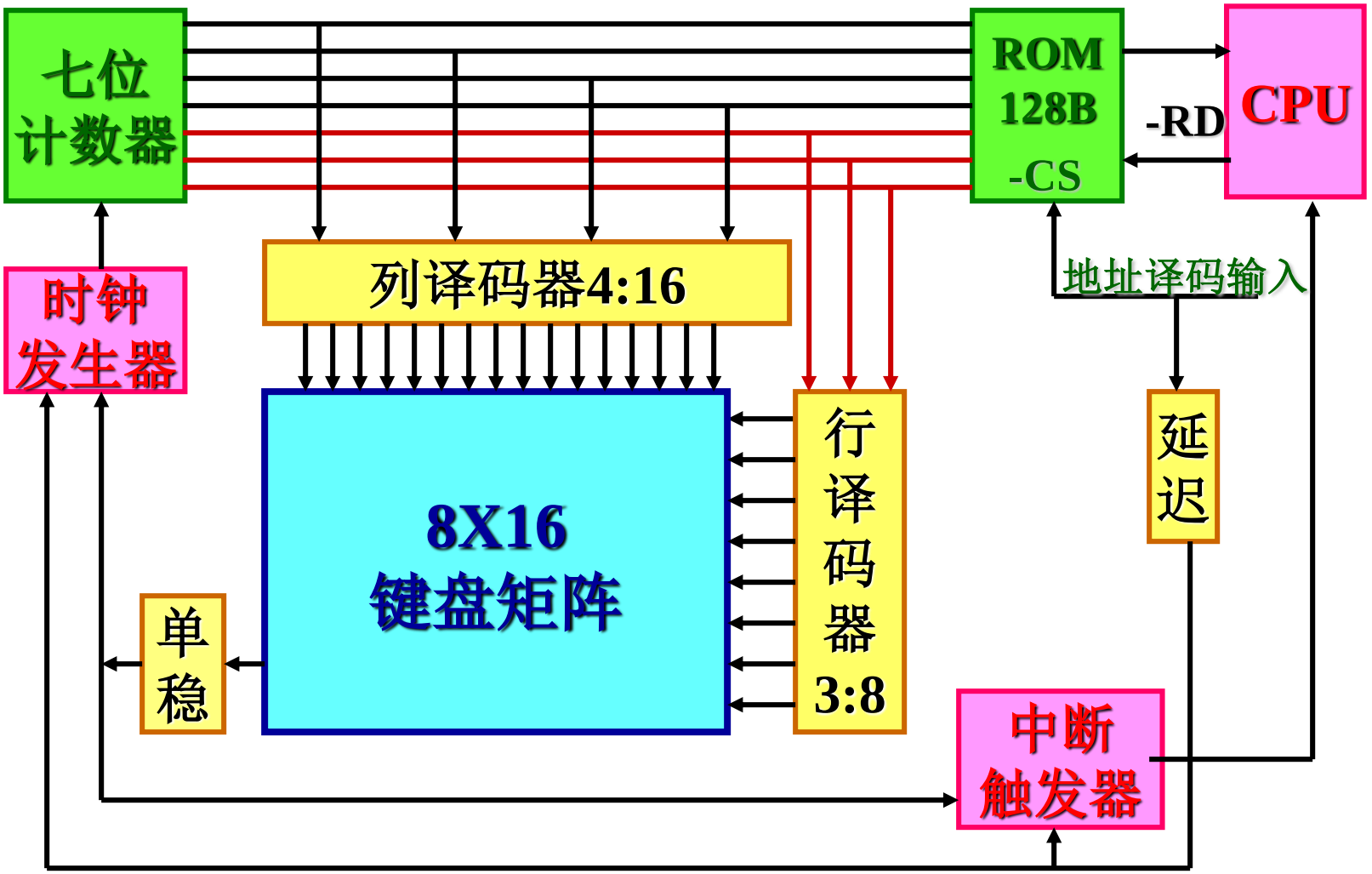
（与字符集大小有关）

3）CPU从键盘读入的应为字符“F”的ASCII码= 01000110（46H），其中最高位为奇校验位（注：不是位置码）。

4）该键盘的原理性逻辑框图见下页，与教材图5.15类似，主要需标明参数。

5）如果不考虑校验技术，并按ASCII码位序设计键阵列（注意），则ROM编码表可省，此时7位计数器输出值（扫描码或键位置码）即为ASCII码。

该键盘的原理性逻辑框图如下：



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。

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。

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：显示缓冲存储器中存放着一屏要显示的字符ASCII码信息，它的作用是支持屏幕扫描时的反复刷新；

只读存储器中存放着字符集中所有字符的点阵信息，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?它与端口有何区别？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端口一般指I/O接口中的各种寄存器。为了便于程序对这些寄存器进行访问，通常给每个寄存器分配一个地址编号，这种编号被称为I/O端口地址，相应的寄存器也叫作I/O端口。

I/O接口和I/O端口是两个不同的概念。一个接口中往往包含若干个端口，因此接口地址往往包含有若干个端口地址。

由于I/O设备的物理结构和工作速率一般与主机差异很大，无法直接相连，因此通常通过I/O接口进行连接。

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。

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中断向量表表项地址编号，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JMP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。

22. 程序查询方式和程序中断方式都是通过“程序”传送数据，两者的区别是什么？

答：程序查询方式通过“程序”传送数据时，程序对I/O的控制包括了I/O准备和I/O传送两段时间。由于I/O的工作速度比CPU低得多，因此程序中要反复询问I/O的状态，造成“踏步等待”，严重浪费了CPU的工作时间。

而程序中断方式虽然也是通过“程序”传送数据，但程序仅对I/O传送阶段进行控制，I/O准备阶段不需要CPU查询。故CPU此时照样可以运行现行程序，与I/O并行工作，大大提高了CPU的工作效率。

25. 根据以下要求设计一个产生**3**个设备向量地址的电路。

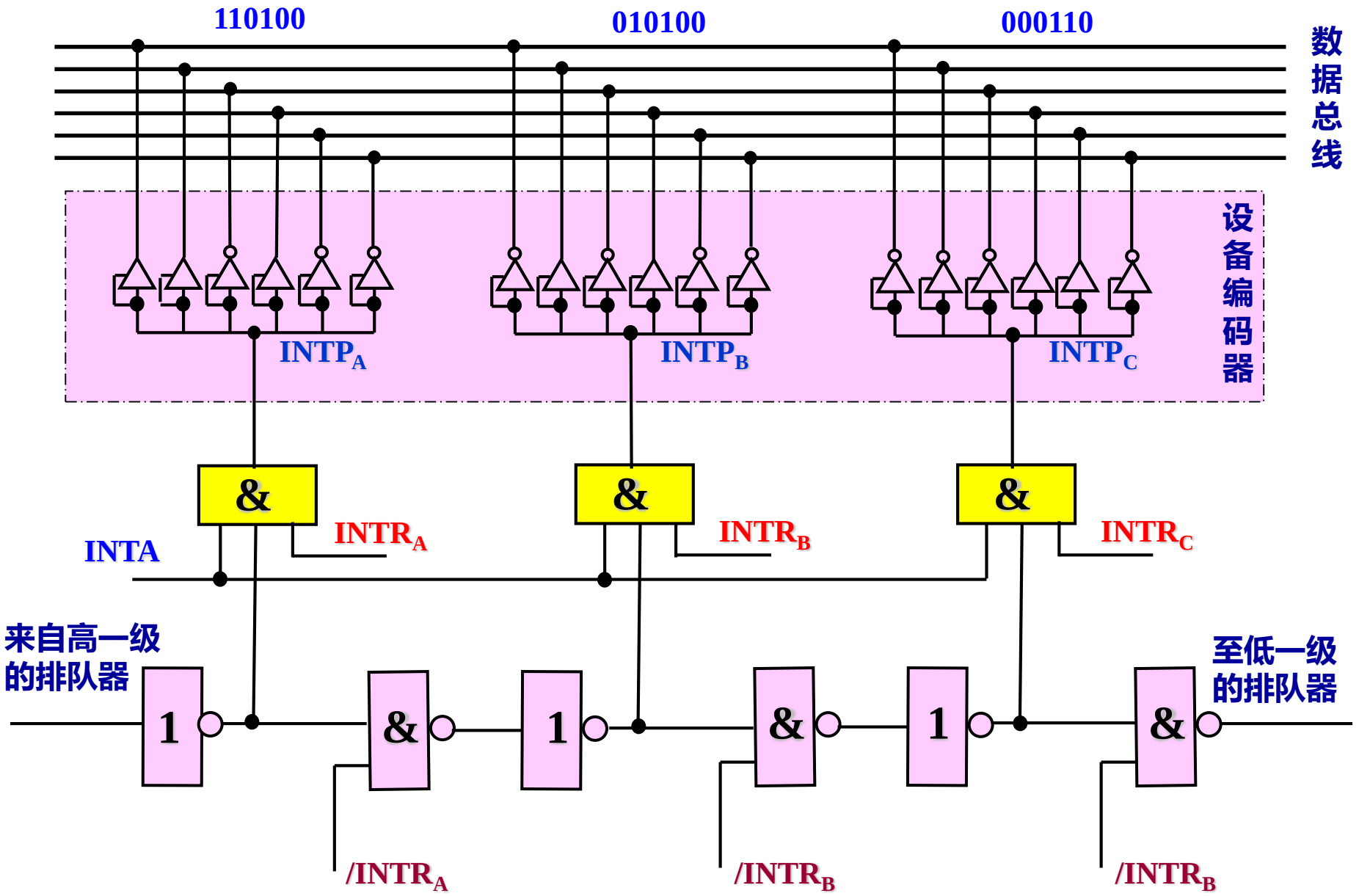
（**1**）**3**个设备的优先级按**A→B→C**降序排列。

（**2**）**A**、**B**、**C**的向量地址分别为**110 100**、**010 100**、**000 110**。

（**3**）排队器采用链式排队电路。

（**4**）当**CPU**发来中断响应信号**INTA**时，可将向量地址取至**CPU**。

解：此题与教材例**5.2**类似，可参考设计。该设备向量地址的电路如下：


110100
010100
000110
数据总线
设备编码器
INTP_A
INTP_B
INTP_C
&
&
&
INTA
INTR_A
INTR_B
INTR_C
来自高一级的排队器
1
&
1
&
1
&
至低一级的排队器
/INTR_A
/INTR_B
/INTR_B

26. 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。

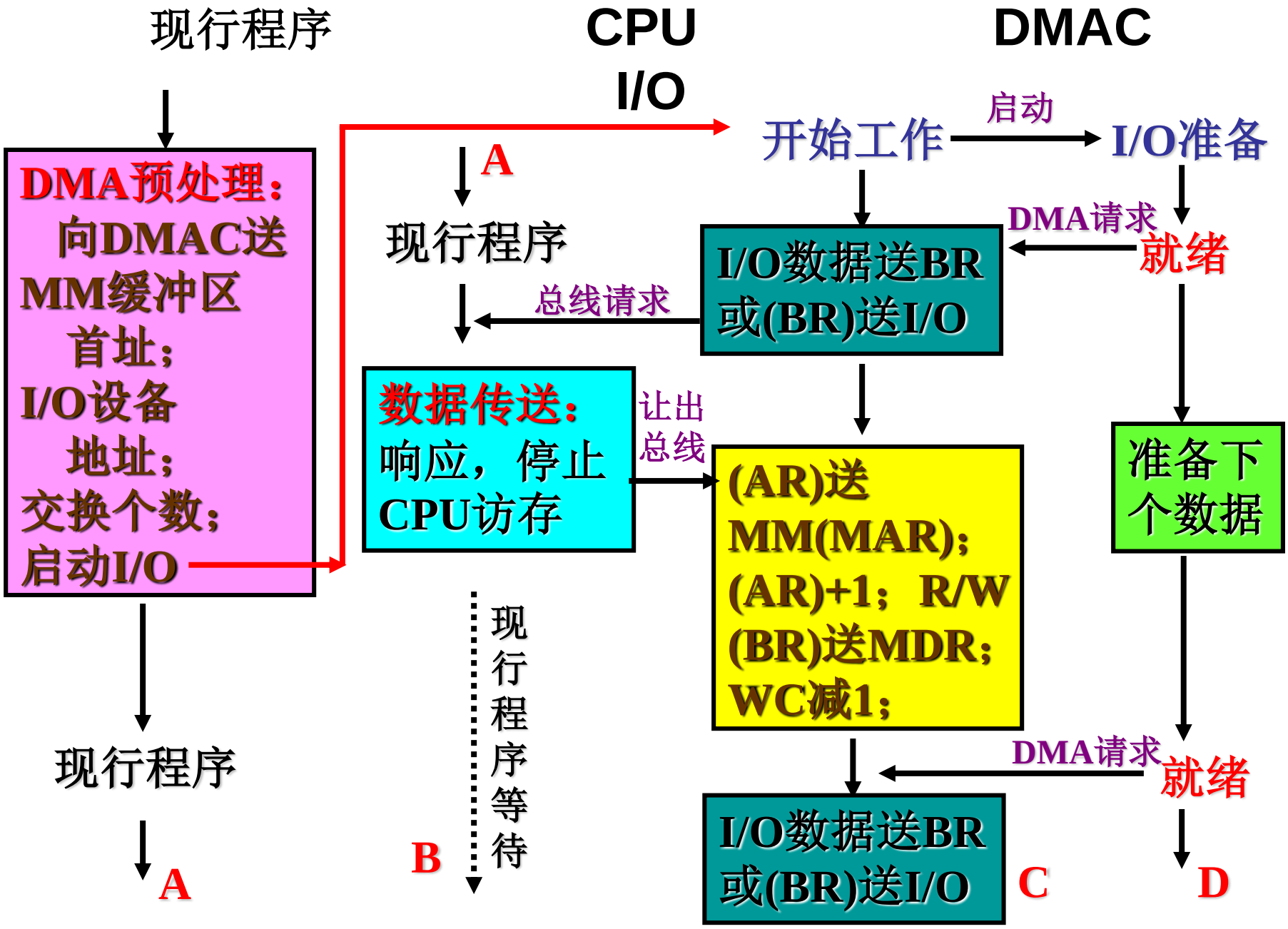
28. CPU对DMA请求和中断请求的响应时间是否相同？为什么？

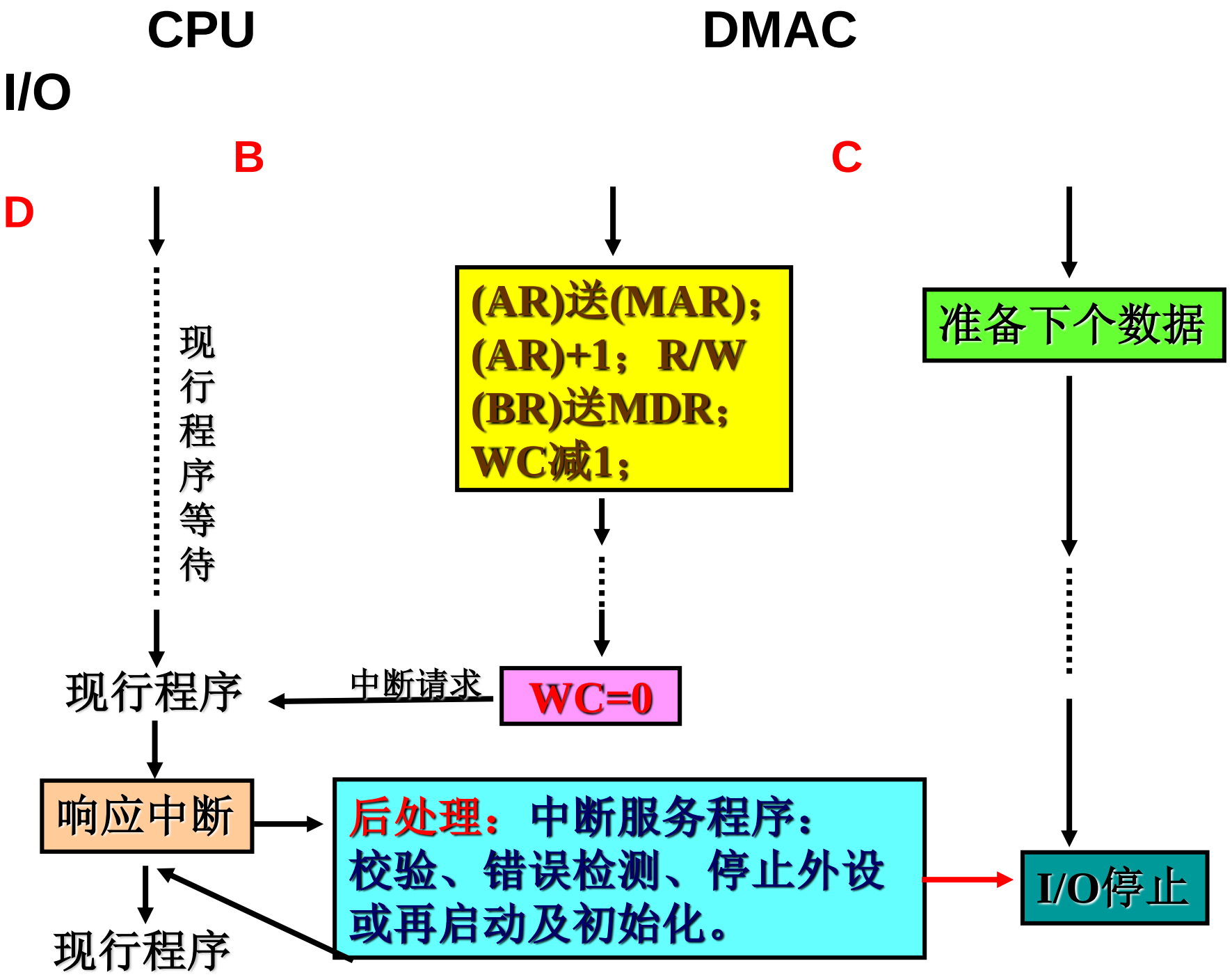
解： CPU对DMA请求和中断请求的响应时间不相同，因为两种方式的交换速度相差很大，因此CPU必须以更短的时间间隔查询并响应DMA请求（一个存取周期末）。

30. 在**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。

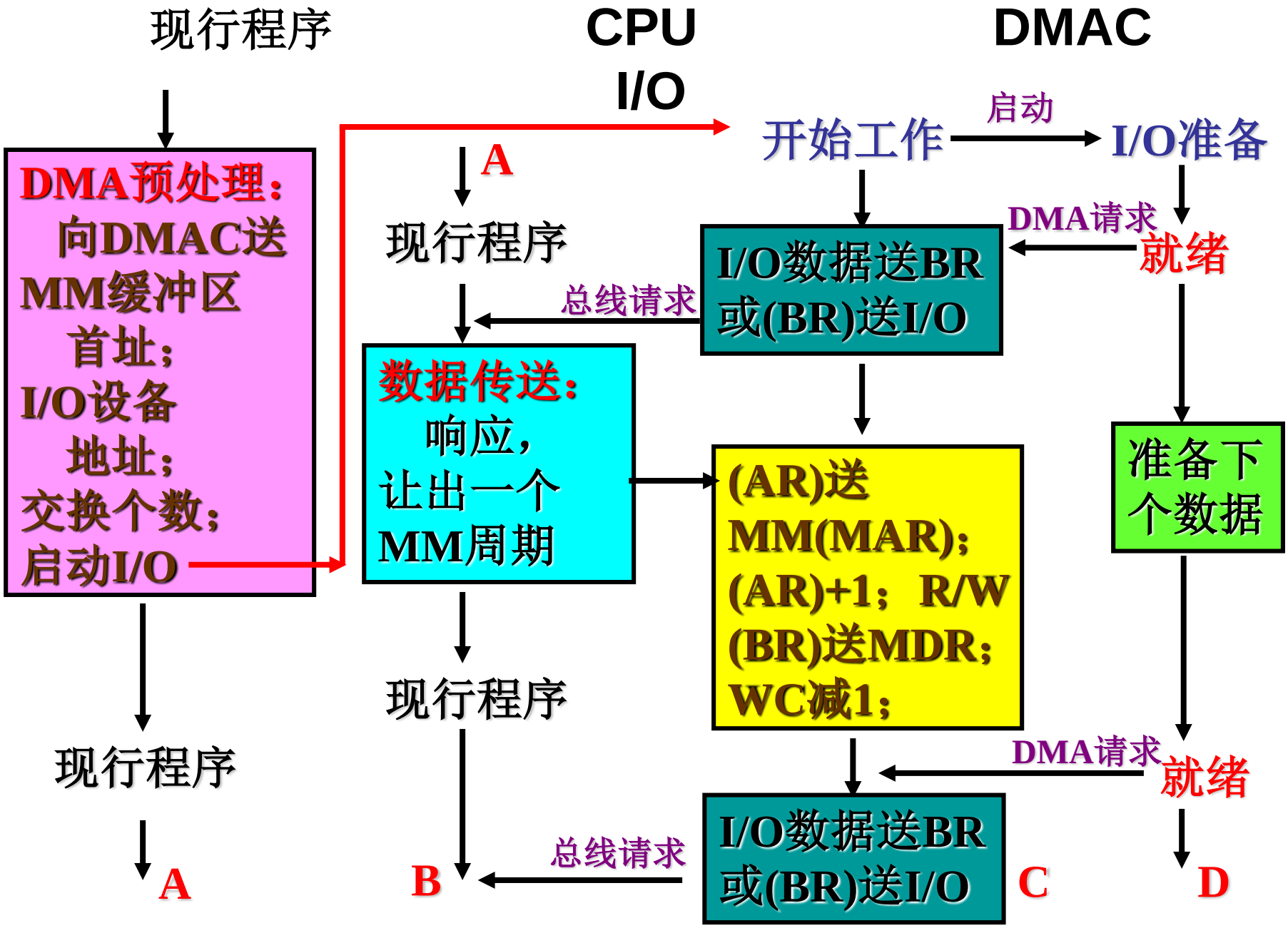
解：两种**DMA**方式的工作流程见下页，其主要区别在于传送阶段，现行程序是否完全停止访存。

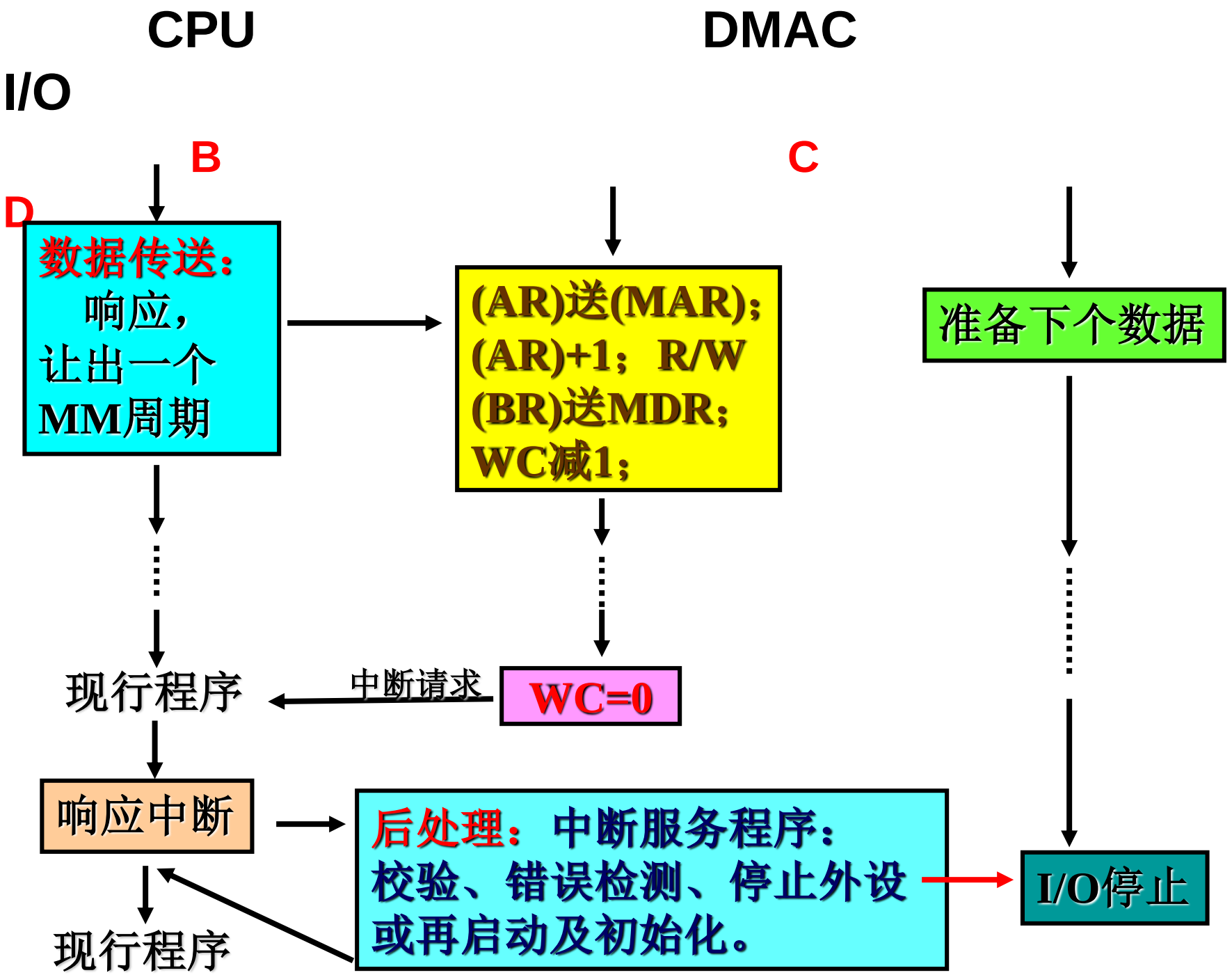
停止CPU访存方式的DMA工作流程如下：




CPU
DMAC
I/O
B
C
D
现行程序等待
(AR)送(MAR);
(AR)+1; R/W
(BR)送MDR;
WC减1;
准备下个数据
现行程序
中断请求
WC=0
响应中断
后处理：中断服务程序：
校验、错误检测、停止外设
或再启动及初始化。
I/O停止
现行程序

周期窃取方式的DMA工作流程如下：




CPU
DMAC
I/O
B
C
D
数据传送：
响应，
让出一个
MM周期
(AR)送(MAR)；
(AR)+1； R/W
(BR)送MDR；
WC减1；
准备下个数据
现行程序
中断请求
WC=0
响应中断
后处理：中断服务程序：
校验、错误检测、停止外设
或再启动及初始化。
I/O停止
现行程序

31. 假设某设备向CPU传送信息的最高频率是40 000次/秒，而相应的中断处理程序其执行时间为40μs，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？

解：该设备向CPU传送信息的时间间隔

$=1/40K=0.025\times10^3=25\mu s < 40\mu s$

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。

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，25μs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，40μs时响应；

50μs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80μs时响应；

75μs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；

32. 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**25μs**，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？

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。

道容量=1KB×8÷16 =1K ×8 ×8 ÷16

=1K ×4=4K字

数传率=4K字×3000转/分

=4K字×50转/秒 =200K字/秒

一个字的传送时间=1/200K字/秒≈5μs

注：在此1K=1024，来自数据块单位缩写。

由上计算知：5 μs<<25 μs，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）。

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）。

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。

补充题：

一、某CRT显示器可显示64种ASCII字符，每帧可显示72字×24排；每个字符字形采用7×8点阵，即横向7点，字间间隔1点，纵向8点，排间间隔6点；帧频50Hz，采取逐行扫描方式。假设不考虑屏幕四边的失真问题，且行回扫和帧回扫均占扫描时间的20%，问：

1）显存容量至少有多大？

2）字符发生器（ROM）容量至少有多大？

3）显存中存放的是那种信息？

4）显存地址与屏幕显示位置如何对应？

5）设置哪些计数器以控制显存访问与屏幕扫描之间的同步？它们的模各是多少？

6）点时钟频率为多少？

二、有一编码键盘，其键阵列为8行×16列，分别对应128种ASCII码字符，采用硬件扫描方式确认按键信号，问：

1）扫描计数器应为多少位？

2）ROM容量为多大？

3）若行、列号均从0开始编排，则当第5行第7列的键表示字母“F”时，CPU从键盘读入的二进制编码应为多少（设采用奇校验）？

4）参考教材图5.15，画出该键盘的原理性逻辑框图；

5）如果不考虑校验技术，此时ROM是否可省？

三、一针式打印机采用7×9点阵打印字符，每行可打印132个字符，共有96种可打印字符，用带偶校验位的ASCII码表示。问：

1）打印缓存容量至少有多大？

2）字符发生器容量至少有多大？

3）列计数器应有多少位？

4）缓存地址计数器应有多少位？

解：

1）打印缓存最小容量 = 132×8

= 132B

（考虑偶校验位）

2）ROM最小容量 = 96×7列×9行

= 672×9位

返回目录

3）列计数器 = 3位

（7列向上取2的幂）

4）缓存地址计数器 = 8位

（132向上取2的幂）

讨论：

1、由于针打是按列打印，所以ROM一个存储单元中存一列的9个点，则容量为672×9位；

2、列计数器是对列号进行计数，所以模=7，3位（模不等于位数）；

3、同样缓存地址计数器模=132，8位。

计算机的运算方法

第　六　章

1. 最少用几位二进制数即可表示任一五位长的十进制正整数？

解：五位长的十进制正整数中，最大的数99999满足条件：2^{16}（=65536）<99999<2^{17}（=131072），故最少用17位二进制数即可表示任一五位长的十进制正整数。

2. 已知$X=0.a_1a_2a_3a_4a_5a_6$（a_i为0或1），讨论下列几种情况时a_i各取何值。

（1）$X > 1/2$；　　（2）$X \geq 1/8$；

（3）$1/4 \geq X > 1/16$

解：（1）若要$X > 1/2$，只要$a_1=1$，$a_2 \sim a_6$不全为0即可（a_2 or a_3 or a_4 or a_5 or $a_6 = 1$）；

（2）若要$X \geq 1/8$，只要$a_1 \sim a_3$不全为0即可（a_1 or a_2 or $a_3 =1$），$a_4 \sim a_6$可任取0或1；

（3）若要$1/4 \geq X > 1/16$，只要$a_1=0$，a_2可任取0或1；

当$a_2=0$时，若$a_3=0$，则必须$a_4=1$，且a_5、a_6不全为0（a_5 or $a_6=1$；若a3=1，则a_4~a_6可任取0或1；

当$a_2=1$时， a_3~a_6可任取0或1。

3. 设x为整数，$[x]_{补}=1$，$x_1x_2x_3x_4x_5$，若要求 x < -16，试问 x_1~x_5 应取何值？

解：若要x < -16，需 $x_1=0$，x_2~x_5 任意。（注：负数绝对值大的反而小。）

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		原码	反码	补码
十进制	二进制			
-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；

$[x]_{补}$	$[x]_{原}$	x（二进制）	x（十进制）
1.1100	1.0100	-0.0100	-1/4
1.1001	1.0111	-0.0111	-7/16
0.1110	0.1110	+0.1110	+7/8
1.0000	无	-1.0000	-1
1，0101	1，1011	-1011	-11
1，1100	1，0100	-0100	-4
0，0111	0，0111	+0111	+7
1，0000	无	-10000	-16

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$x \geq 0$，则

$[x]_{补}=[x]_{原}$成立；

若$x < 0$，则当$x= -1/2$时，

$[x]_{补}=[x]_{原}$成立。

当x为整数时，若$x \geq 0$，则

$[x]_{补}=[x]_{原}$成立；

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。

9. 当十六进制数**9B**和**FF**分别表示为原码、补码、反码、移码和无符号数时，所对应的十进制数各为多少（设机器数采用一位符号位）？

解：真值和机器数的对应关系如下：

十六进制	真值	无符号数	原码	反码	补码	移码
9BH	二进制 十进制	1001 1011 155	-11 011 -27	-1100100 -100	-1100101 -101	+11011 +27
FFH	二进制 十进制	1111 1111 255	-1111111 -127	-0000000 -0	-0000001 -1	+1111111 +127

注意：1）9BH、FFH为机器数，本身含符号位。

2）移码符号位与原、补、反码相反，数值同补码。

10. 在整数定点机中，设机器数采用一位符号位，写出±0的原码、补码、反码和移码，得出什么结论？

解：0的机器数形式如下：

真值	原码	补码	反码	移码
+0	0，00…0	0，00…0	0，00…0	1，00…0
-0	1，00…0	0，00…0	1，11…1	1，00…0

结论：补、移码0的表示唯一，原、反码不唯一。

注意：本题不用分析不同编码间的其他特性。

11. 已知机器数字长为4位（其中1位为符号位），写出整数定点机和小树定点机中原码、补码和反码的全部形式，并注明其对应的十进制真值。

解：机器数与对应的真值形式如下：

	真值（二进制）	真值（十进制）	原码	反码	补码
整数	+111	+7	0，111	同原码	同原码
	+110	+6	0，110		
	+101	+5	0，101		
	+100	+4	0，100		
	+011	+3	0，011		
	+010	+2	0，010		
	+001	+1	0，001		
	+000	+0	0，000		

续表1：

	真值（二进制）	真值（十进制）	原码	反码	补码
整数	-1000	-8	无	无	1，000
	-111	-7	1，111	1，000	1，001
	-110	-6	1，110	1，001	1，010
	-101	-5	1，101	1，010	1，011
	-100	-4	1，100	1，011	1，100
	-011	-3	1，011	1，100	1，101
	-010	-2	1，010	1，101	1，110
	-001	-1	1，001	1，110	1，111
	-000	-0	1，000	1，111	0，000

续表2：

	真值（二进制）	真值（十进制）	原码	反码	补码
小数	+0.111	+7/8	0.111	同原码	同原码
	+0.110	+3/4	0.110		
	+0.101	+5/8	0.101		
	+0.100	+1/2	0.100		
	+0.011	+3/8	0.011		
	+0.010	+1/4	0.010		
	+0.001	+1/8	0.001		
	+0.000	+0	0.000		

续表3：

	真值（二进制）	真值（十进制）	原码	反码	补码
小数	-1.000	-1	无	无	1.000
	-0.111	-7/8	1.111	1.000	1.001
	-0.110	-3/4	1.110	1.001	1.010
	-0.101	-5/8	1.101	1.010	1.011
	-0.100	-1/2	1.100	1.011	1.100
	-0.011	-3/8	1.011	1.100	1.101
	-0.010	-1/4	1.010	1.101	1.110
	-0.001	-1/8	1.001	1.110	1.111
	-0.000	-0	1.000	1.111	0.000

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、7.375、-86.5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。

（注：题意中应补充规格化数的要求。）

解：据题意画出该浮点数的格式：

1	4	1	10
阶符	阶码	数符	尾数

注意：

1）正数补码不“变反+1”。

2）机器数末位的0不能省。

将十进制数转换为二进制：

$x_1=51/128=(0.011\ 001\ 1)_2$

$=2^{-1}\times(0.110\ 011)_2$

$x_2=-27/1024=(-0.000\ 001\ 101\ 1)_2$

$=2^{-5}\times(-0.110\ 11)_2$

$x_3=7.375=(111.011)_2$

$=2^{3}\times(0.111\ 011)_2$

$x_4=-86.5=(-1\ 010\ 110.1)_2$

$=2^{7}\times(-0.101\ 011\ 01)_2$

则以上各数的浮点规格化数为：

（1）$[x_1]_{浮}$=1，0001；0.110 011 000 0

（2）$[x_1]_{浮}$=1，1111；0.110 011 000 0

（3）$[x_1]_{浮}$=0，1111；0.110 011 000 0

000 0

（1）$[x_3]_{浮}$=0，0011；0.111 011 000 0

（2）$[x_3]_{浮}$=0，0011；0.111 011 000 0

（3）$[x_3]_{浮}$=1，0011；0.111 011 000 0

（1）$[x_4]_{浮}$=0，0111；1.101 011 010 0

（2）$[x_4]_{浮}$=0，0111；1.010 100 110 0

数符	阶符	阶码	尾数

（3）$[x_4]_{浮}$=1，0111；1.010 100 110 0

此时只要将上述答案中的数符位移到最前面即可

注：以上浮点数也可采用如下格式

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：$N_{+max}=2^{15}\times(1-2^{-10})$

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：$N_{+min}=2^{-16}\times 2^{-1}=2^{-17}$

r=16时，最大正数的浮点格式为：

0，1111；0.1111 1111 11

其真值为：$N_{+max}=16^{15}\times(1-2^{-10})$

非零最小规格化正数浮点格式为：

1，0000；0.0001 0000 00

其真值为：$N_{+min}=16^{-16}\times 16^{-1}=16^{-17}$

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（2^{15}）< 6万 <65536（2^{16}），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

按此格式，该浮点数上溢的条件为：阶码 ≥ 32

该浮点数格式如下：

1	5	1	25

阶符	阶　值	数符	尾　数

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、尾数用补码表示（此时阶码为最小阶、尾数为零，而移码的最小码值正好为“0”，补码的零的形式也为“0”，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）。分别写出正数和负数的表示范围；

（注：加条件：阶原尾原非规格化数。）

（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：各种表示方法数据范围如下：（1）无符号整数：$0 \sim 2^{16}-1$，

即：0 ~ 65535；

（2）原码定点小数：

$1-2^{-15} \sim -(1-2^{-15})$

（3）补码定点小数：

$1-2^{-15} \sim -1$

（4）补码定点整数：$2^{15}-1 \sim -2^{15}$，

即：32767 ~ -32768；

（5）原码定点整数：

$2^{15}-1 \sim -(2^{15}-1)$，

即：32767 ~ -32767；

（6）据题意画出该浮点数格式：

1	5	1	9
阶符	阶码	数符	尾数

由于题意中未指定该浮点数所采用的码制，则不同的假设前提会导致不同的答案，示意如下：

1）当采用阶原尾原非规格化数时，

最大正数=0，11 111；0.111 111 111

最小正数=1，11 111；0.000 000 001

则正数表示范围为：

$2^{31}\times(1-2^{-9})\sim 2^{-31}\times 2^{-9}$

最大负数=1，11 111；1.000 000 001

最小负数=0，11 111；1.111 111 111

则负数表示范围为：

$2^{-31} \times (-2^{-9}) \sim -2^{31} \times (1-2^{-9})$

2）当采用阶移尾原非规格化数时，

正数表示范围为：

$2^{31} \times (1-2^{-9}) \sim 2^{-32} \times 2^{-9}$

负数表示范围为：

$2^{-32} \times (-2^{-9}) \sim -2^{31} \times (1-2^{-9})$

注：零视为中性数，不在此范围内。

（7）当机器数采用补码规格化形式时，若不考虑隐藏位，则

最大正数=0，11 111；0.111 111 111

最小正数=1，00 000；0.100 000 000

其对应的正数真值范围为：

$2^{31}\times(1-2^{-9})\sim 2^{-32}\times 2^{-1}$

最大负数=1，00 000；1.011 111 111

最小负数=0，11 111；1.000 000 000

其对应的负数真值范围为：

$-2^{-32}\times(2^{-1}+2^{-9})\sim 2^{31}\times(-1)$

注意：

1）应写出可表示范围的上、下限精确值（用≥或≤，不要用>或<）。

2）应用十进制2的幂形式分阶、尾两部分表示，这样可反映出浮点数的格式特点。括号不要乘开，不要用十进制小数表示，不直观、不精确且无意义。

3）原码正、负域对称，补码正、负域不对称，浮点数阶、尾也如此。特别要注意浮点负数补码规格化范围。（满足条件：数符⊕MSB位=1）

17. 设机器数字长为8位（含1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。

$[x_1]_{原}$=0.001 1010；

$[x_2]_{原}$=1.110 1000；

$[x_3]_{原}$=1.001 1001；

$[y_1]_{补}$=0.101 0100；

$[y_2]_{补}$=1.110 1000；

$[y_3]_{补}$=1.001 1001；

$[z_1]_{反}$=1.010 1111；

$[z_2]_{反}$=1.110 1000；

$[z_3]_{反}$=1.001 1001。

解：算术左移一位：

$[x_1]_{原}$=0.011 0100；正确

$[x_2]_{原}$=1.101 0000；溢出（丢1）出错

$[x_3]_{原}$=1. 011 0010；正确

$[y_1]_{补}$=0. 010 1000；溢出（丢1）出错

$[y_2]_{补}$=1.101 0000；正确

$[y_3]_{补}$=1.011 0010；溢出（丢0）出错

$[z_1]_{反}$=1. 101 1111；溢出（丢0）出错

$[z_2]_{反}$=1. 101 0001；正确

$[z_3]_{反}$=1.011 0011；溢出（丢0）出错

算术左移两位：

$[x_1]_{原}$=0.110 1000；正确

$[x_2]_{原}$=1.010 0000；溢出（丢11）出错

$[x_3]_{原}$=1. 110 0100；正确

算术左移两位：

$[y_1]_{补}$=0. 101 0000；溢出（丢10）出错

$[y_2]_{补}$=1.010 0000；正确

$[y_3]_{补}$=1.110 0100；溢出（丢00）出错

$[z_1]_{反}$=1. 011 1111；溢出（丢01）出错

$[z_2]_{反}$=1. 010 0011；正确

$[z_3]_{反}$=1.110 0111；溢出（丢00）出错

算术右移一位：

$[x_1]_{原}$=0.000 1101；正确

$[x_2]_{原}$=1.011 0100；正确

$[x_3]_{原}$=1.000 1100(1)；丢1，产生误差

$[y_1]_{补}$=0.010 1010；正确

$[y_2]_{补}$=1.111 0100；正确

$[y_3]_{补}$=1.100 1100(1)；丢1，产生误差

算术右移一位：

$[z_1]_{反}$=1.101 0111；正确

$[z_2]_{反}$=1.111 0100(0)；丢0，产生误差

$[z_3]_{反}$=1.100 1100；正确

算术右移两位：

$[x_1]_{原}$=0.000 0110（10）；产生误差

$[x_2]_{原}$=1.001 1010；正确

$[x_3]_{原}$=1.000 0110（01）；产生误差

$[y_1]_{补}$=0.001 0101；正确

$[y_2]_{补}$=1.111 1010；正确

$[y_3]_{补}$=1.110 0110（01）；产生误差

$[z_1]_{反}$=1.110 1011；正确

$[z_2]_{反}$=1.111 1010（00）；产生误差

$[z_3]_{反}$=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 0010）$_2$

B= -13/32=（-0.011 0100）$_2$

$[A]_{补}$=0.001 0010

$[B]_{补}$=1.100 1100

$[A+B]_{补}$= 0. 0 0 1　0 0 1 0

\+ 1. 1 0 0　1 1 0 0

1. 1 0 1　1 1 1 0 ——无溢出

A+B=（ -0.010 0010）$_2$ = -17/64

（2）A=19/32=（0.100 1100）$_2$

B= -17/128=（-0.001 0001）$_2$

$[A]_{补}$=0.100 1100

$[B]_{补}$=1.110 1111

$[-B]_{补}$=0.001 0001

$[A-B]_{补}$= 0. 1 0 0　1 1 0 0

\+ 0. 0 0 1　0 0 0 1

0. 1 0 1　1 1 0 1 ——无溢出

A-B=（0.101 1101）$_2$ = 93/128

（3）A= -3/16=（-0.001 1000）$_2$

B=9/32=（0.010 0100）$_2$

$[A]_{补}$=1.110 1000

$[B]_{补}$= 0.010 0100

$[A+B]_{补}$= 1. 1 1 0　1 0 0 0

+　0. 0 1 0　0 1 0 0

0. 0 0 0　1 1 0 0 —— 无溢出

A+B=（0.000 1100）$_2$ = 3/32

（4）A= -87=（-101 0111）$_2$

B=53=（110 101）$_2$

$[A]_{补}$=1，010 1001

$[B]_{补}$=0，011 0101

$[-B]_{补}$=1，100 1011

$[A-B]_{补}$= 1，0 1 0　1 0 0 1

+　1，1 0 0　1 0 1 1

0，1 1 1　0 1 0 0 —— 溢出

A-B=（-1，000 1100）$_2$ = -140

（5）A=115=（111 0011）$_2$

B= -24=（-11 000）$_2$

$[A]_{补}$=0，111 0011

$[B]_{补}$=1，110 1000

$[A+B]_{补}$= 0，1 1 1　0 0 1 1

+　1，1 1 0　1 0 0 0

0，1 0 1　1 0 1 1——无溢出

A+B=（101 1011）$_2$ = 91

注意：1、单符号位运算要用单符号位的判断方法判溢出；

2、结果的真值形式上要和原始数据一致。

20. 用原码一位乘、两位乘和补码一位乘（Booth算法）、两位乘计算$x \cdot y$。

（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）$[x]_{原}=x=0.110111$，$[y]_{原}=1.101110$

$x^*=0.110111$，$y^*=0.101110$

$x_0=0$，$y_0=1$，$z_0=x_0 \oplus y_0=0 \oplus 1=1$

$x^* \times y^*=0.100\ \ 111\ \ 100\ \ 010$

$[x \times y]_{原}=1.100\ \ 111\ \ 100\ \ 010$

$x \cdot y= -0.\ 100\ \ 111\ \ 100\ \ 010$

原码一位乘：

部分积	乘数y*	
0.000 000	.1 0 1 1 1 0	—— +0
→1 0.000 000	0.1 0 1 1 1	—— +x*
+ 0.110 111		
0.110 111		
→1 0.011 011	1 0.1 0 1 1	—— +x*
+ 0.110 111		
1.010 010		
→1 0.101 001	0 1 0 .1 0 1	—— +x*
+ 0.110 111		
1.100 000		
→1 0.110 000	0 0 1 0.1 0	—— +0
→1 0.011 000	0 0 0 1 0.1	—— x*
+ 0.110 111		
1.001 111		
→1 0.100 111	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
	000.000 000	00.1011 10	0
+	001.101 110	+2x*	
	001.101 110		0
→2	000.011 011	10 00.10 11	
+	111.001 001	$+[-x^*]_{补}$	
	111.100 100		1
→2	111.111 001	00 1000.10	
+	111.001 001	$+[-x^*]_{补}$	
	111.000 010		1
→2	111.110 000	10 0010 00.	
+	000.110 111	+x*	
	000.100 111	10 0010	0

结果同一位乘，$x\cdot y= -0.100\ 111\ 100\ 010$

$[x]_{补}=x=0.110111$

$[y]_{补}=1.010010$

$[-x]_{补}=1.001001$

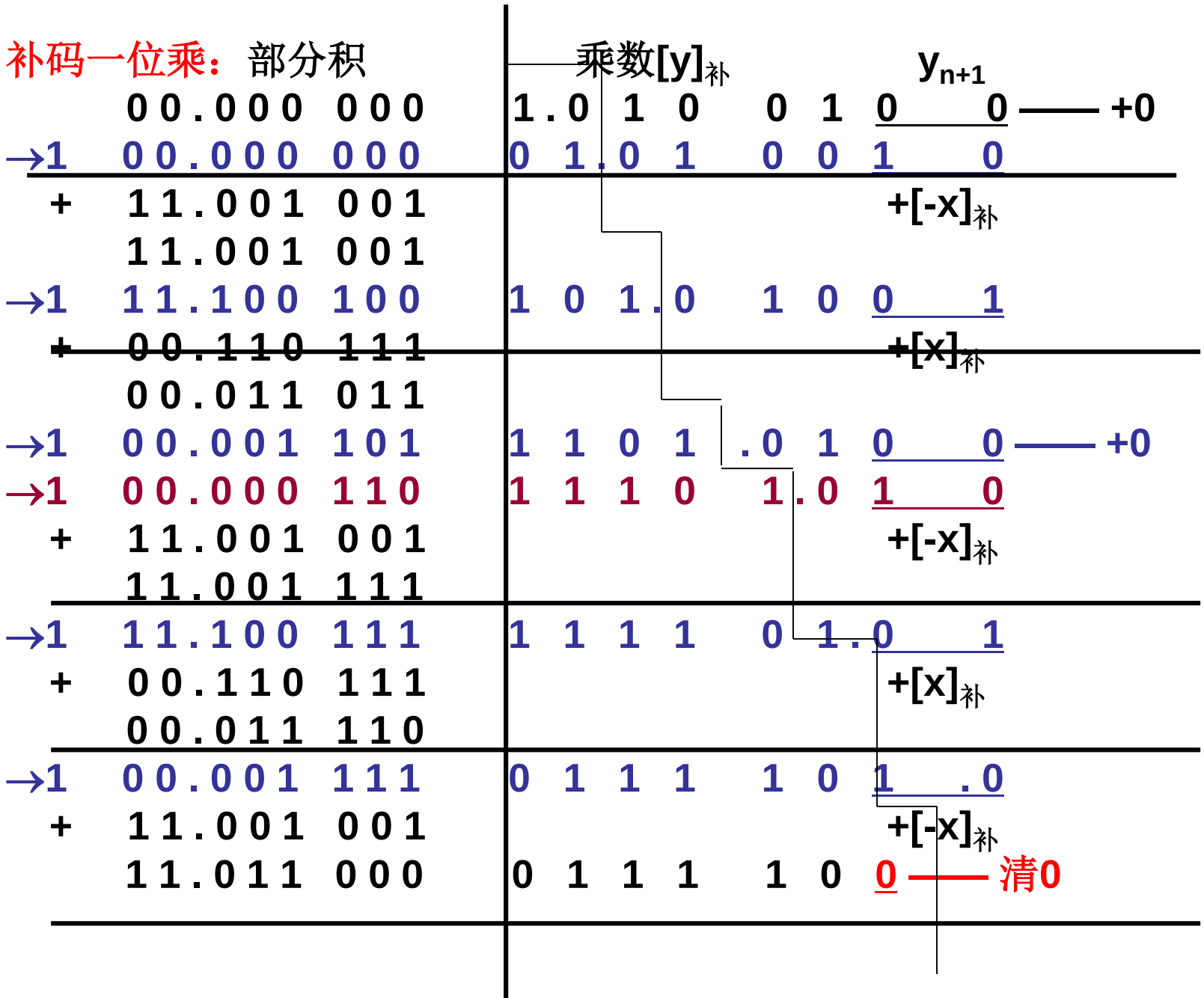
$[2x]_{补}=01.101110$

$[-2x]_{补}=10.010010$

$[x\times y]_{补}=1.011\ \ 000\ \ 011\ \ 110\ 0$

$x\cdot y=-0.100\ \ 111\ \ 100\ \ 010\ \ 0$

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
00.000 000
1.0 1 0 0 1 0 0 —— +0
→1 00.000 000
0 1.0 1 0 0 1 0
+ 11.001 001
+[-x]补
11.001 001
→1 11.100 100
1 0 1.0 1 0 0 1
+ 00.110 111
+[x]补
00.011 011
→1 00.001 101
1 1 0 1 .0 1 0 0 —— +0
→1 00.000 110
1 1 1 0 1.0 1 0
+ 11.001 001
+[-x]补
11.001 111
→1 11.100 111
1 1 1 1 0 1.0 1
+ 00.110 111
+[x]补
00.011 110
→1 00.001 111
0 1 1 1 1 0 1 .0
+ 11.001 001
+[-x]补
11.011 000
0 1 1 1 1 0 0 —— 清0

补码两位乘：

部分积	乘数	y_{n+1}
000.000 000	11.010 010	0
+ 110.010 010		+[-2x]补
110.010 010		
→2 111.100 100	10 11.0 10 0	1
+ 000.110 111		+[x]补
000.011 011		
→2 000.000 110	11 10 1 1.0 1	0
+ 000.110 111		+[x]补
000.111 101		
→2 000.001 111	01 11 1 0 11	.0
+ 111.001 001		+[-x]补
111.011 000	01 11 1 0 0	0.清0

结果同补码一位乘， x·y= -0. 100 111 100 010 00

（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

x_0=1，y_0=1，$z_0=x_0 \oplus y_0=1 \oplus 1=0$

$[x]_{补}$=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

$[x \times y]_{原}$=0.000 111 100 011

$[x \times y]_{补}$=0.000 111 100 011 0

x·y= 0. 000 111 100 011

运算过程如下：

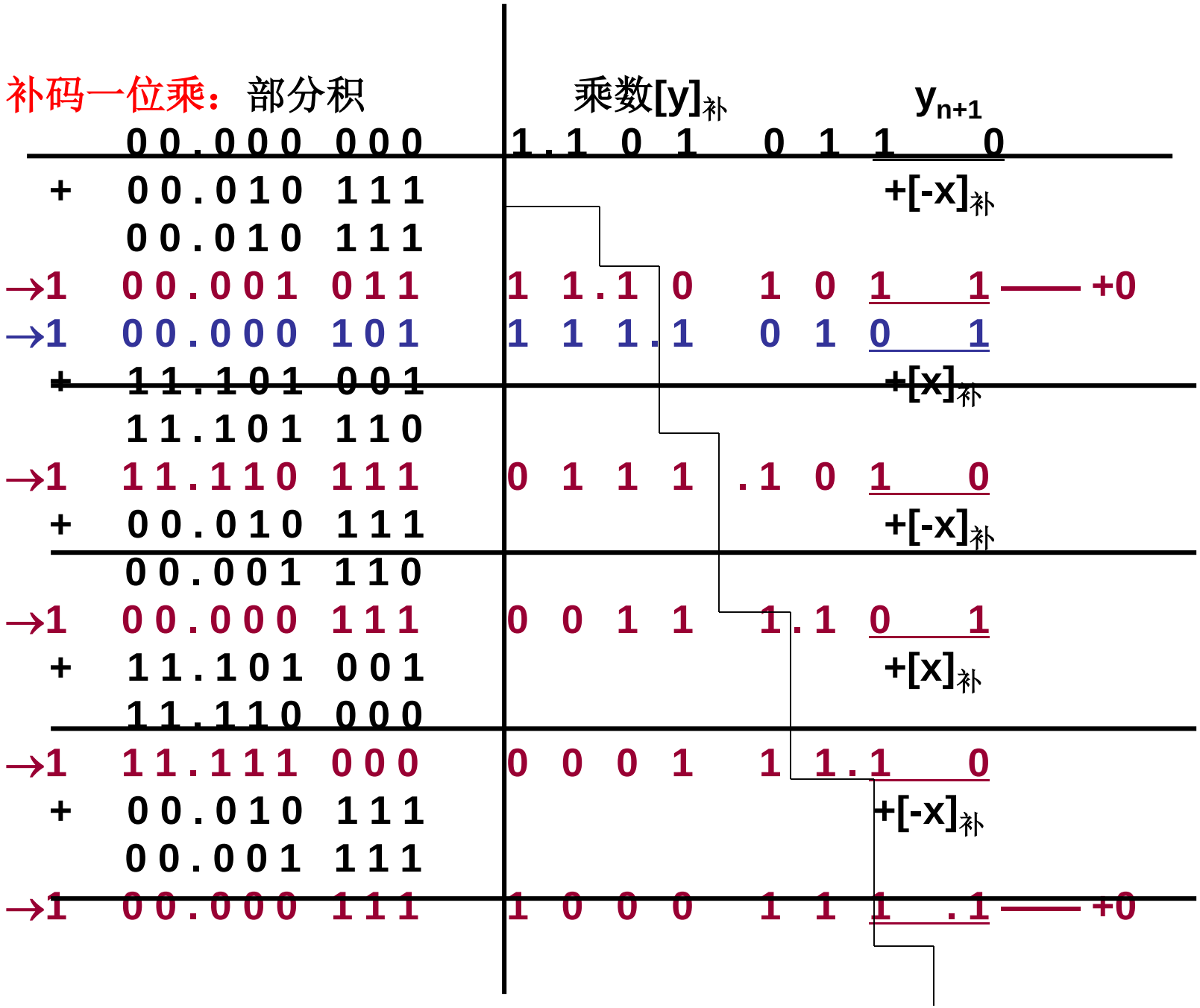
原码一位乘：

部分积	乘数y*	
0.000 000	.0 1 0 1 0 1	—— +x*
+ 0.010 111		
0.010 111		
→1 0.001 011	1.0 1 0 1 0	—— +0
→1 0.000 101	1 1.0 1 0 1	—— +x*
+ 0.010 111		
0.011 100		
→1 0.001 110	0 1 1 .0 1 0	—— +0
→1 0.000 111	0 0 1 1.0 1	—— +x*
+ 0.010 111		
0.011 110		
→1 0.001 111	0 0 0 1 1.0	—— +0
→1 0.000 111	1 0 0 0 1 1	

原码两位乘：

部分积	乘数y*	Cj
000.000 000	00.010 1 0 1	0
+ 000.010 111	+x*	
000.010 111		0
→2 000.000 101	11 0 0.0 1 0 1	
+ 000.010 111	+x*	
000.011 100		0
→2 000.000 111	00 1 1 0 0.0 1	
+ 000.010 111	+x*	
000.011 110		0
→2 000.000 111	10 0 0 1 1 0 0.	
	+0	

结果同一位乘， $x \cdot y = 0.000\ 111\ 100\ 011$


补码一位乘：部分积
乘数[y]补
yn+1
00.000 000
1.1 0 1 0 1 1 0
+ 00.010 111
+[-x]补
00.010 111
→1 00.001 011
1 1.1 0 1 0 1 1 +0
→1 00.000 101
1 1 1.1 0 1 0 1
+ 11.101 001
+[x]补
11.101 110
→1 11.110 111
0 1 1 1 .1 0 1 0
+ 00.010 111
+[-x]补
00.001 110
→1 00.000 111
0 0 1 1 1.1 0 1
+ 11.101 001
+[x]补
11.110 000
→1 11.111 000
0 0 0 1 1 1.1 0
+ 00.010 111
+[-x]补
00.001 111
→1 00.000 111
1 0 0 0 1 1 1 .1 +0

补码两位乘：

部分积	乘数	y_{n+1}
000.000 000	11.101	0 11 0
+ 000.010 111		+[-x]补
000.010 111		
→2 000.000 101	11 11.1	0 10 1
+ 000.010 111		+[-x]补
000.011 100		
→2 000.000 111	00 11 1	1.10 1
+ 000.010 111		+[-x]补
000.011 110		
→2 000.000 111	10 01 1	11 .1
		清0 +0

结果同补码一位乘， $x \cdot y = 0.000\ 111\ 100\ 011\ 00$

（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

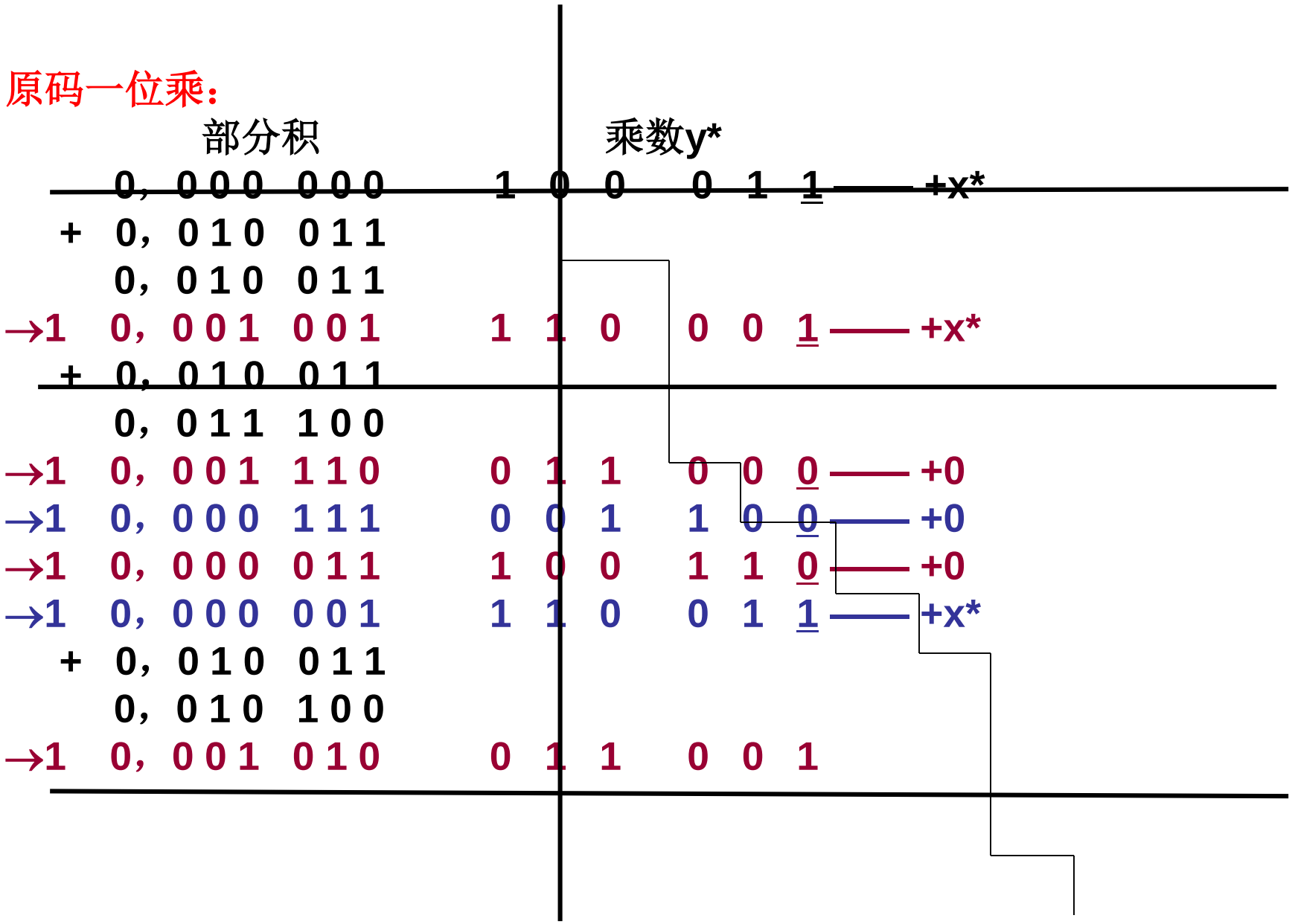
x_0=0，y_0=0，$z_0=x_0 \oplus y_0=0 \oplus 0=0$

$x \cdot y= x^* \times y^*=[x \times y]_{原}=[x \times y]_{补}$

= 0，001 010 011 001

=（665）$_{10}$

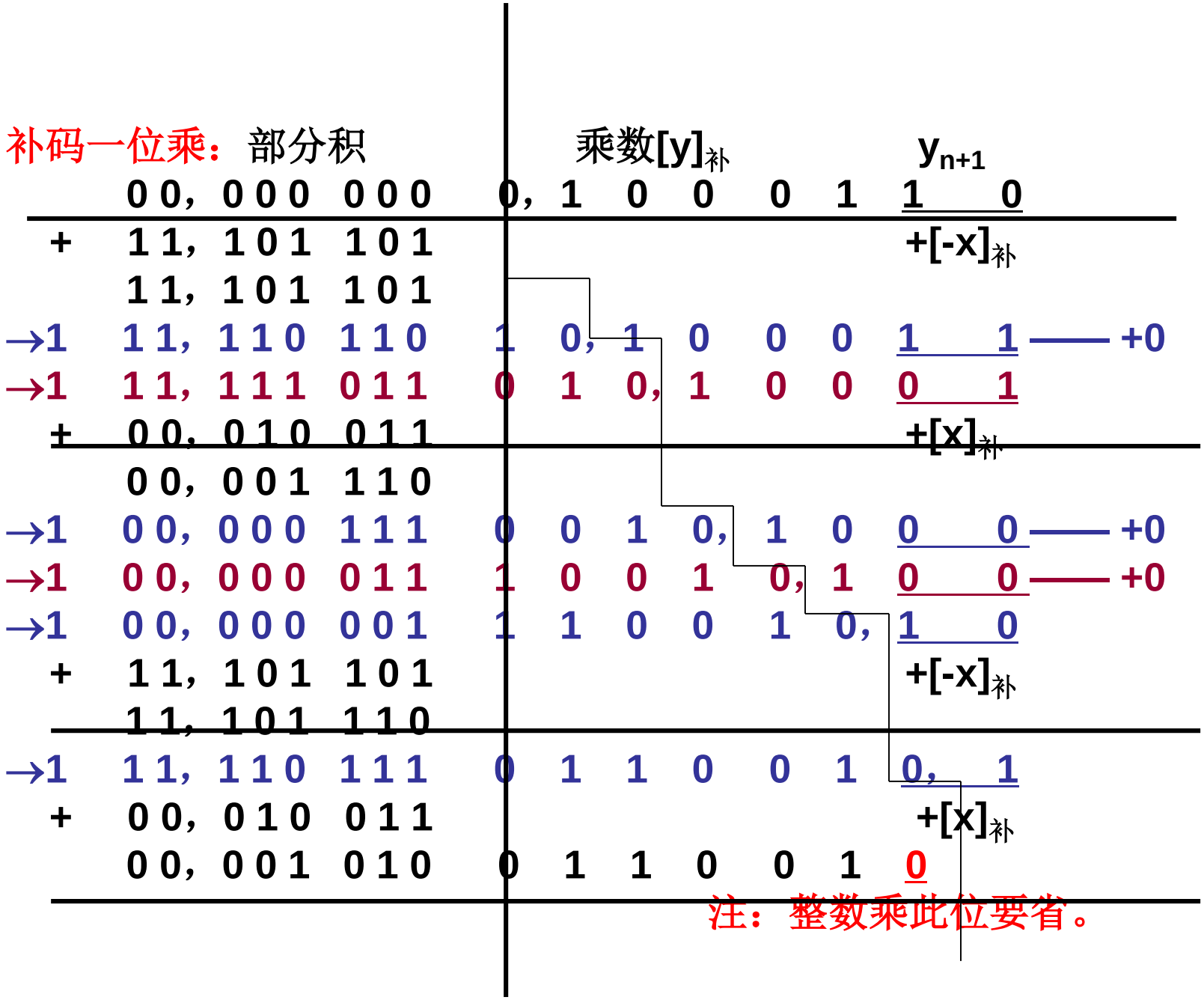
运算过程如下：


原码一位乘：
部分积
乘数y*
0，000 000 1 0 0 0 1 1 +x*
+ 0，010 011
0，010 011
→1 0，001 001 1 1 0 0 0 1 +x*
+ 0，010 011
0，011 100
→1 0，001 110 0 1 1 0 0 0 +0
→1 0，000 111 0 0 1 1 0 0 +0
→1 0，000 011 1 0 0 1 1 0 +0
→1 0，000 001 1 1 0 0 1 1 +x*
+ 0，010 011
0，010 100
→1 0，001 010 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 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 1　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
00，000 000
0，1 0 0 0 1 1 0
+ 11，101 101
+[-x]补
11，101 101
→1 11，110 110
1 0，1 0 0 0 1 1
+0
→1 11，111 011
0 1 0，1 0 0 0 1
+ 00，010 011
+[x]补
00，001 110
→1 00，000 111
0 0 1 0，1 0 0 0
+0
→1 00，000 011
1 0 0 1 0，1 0 0
+0
→1 00，000 001
1 1 0 0 1 0，1 0
+ 11，101 101
+[-x]补
11，101 110
→1 11，110 111
0 1 1 0 0 1 0，1
+ 00，010 011
+[x]补
00，001 010
0 1 1 0 0 1 0
注：整数乘此位要省。

补码两位乘：

	部分积	乘数	y_{n+1}
	0 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		
→2	1 1 1，1 1 1　0 1 1	0 1　0　0，1　0　0　0　1	
+	0 0 0，0 1 0　0 1 1		$+[x]_{补}$
	0 0 0，0 0 1　1 1 0		
→2	0 0 0，0 0 0　0 1 1	1 0　0　1　0　0，1　0　0	
+	1 1 1，0 1 1　0 1 0		$+[-2x]_{补}$
	1 1 1，0 1 1　1 0 1		
→2	1 1 1，1 1 0　1 1 1	0 1　1　0　0　1　0　0，1	
+	0 0 0，0 1 0　0 1 1		+0
	0 0 0，0 0 1　0 1 0	0 1　1　0　0　1　0　0	—省

结果同补码一位乘，　$x \cdot y = 0,001\ 010\ 011\ 001$

（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

$x_0=0$，$y_0=1$，$z_0=x_0 \oplus y_0=0 \oplus 1=1$

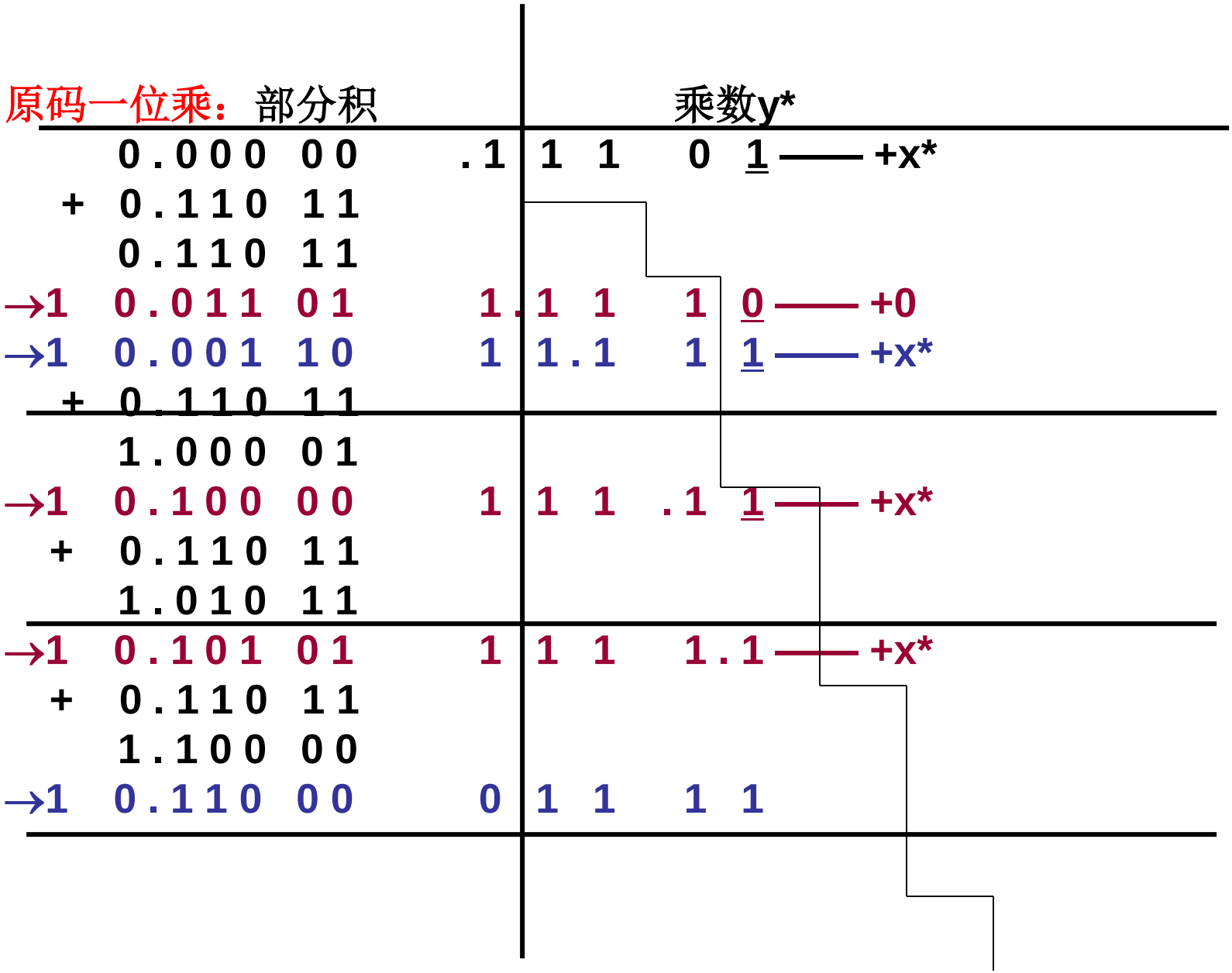
$x^* \times y^*$=0.110 000 111 1

$[x \times y]_{原}$=1.110 000 111 1

$[x \times y]_{补}$=1.001 111 000 10

$x \cdot y$= -0. 110 000 111 1

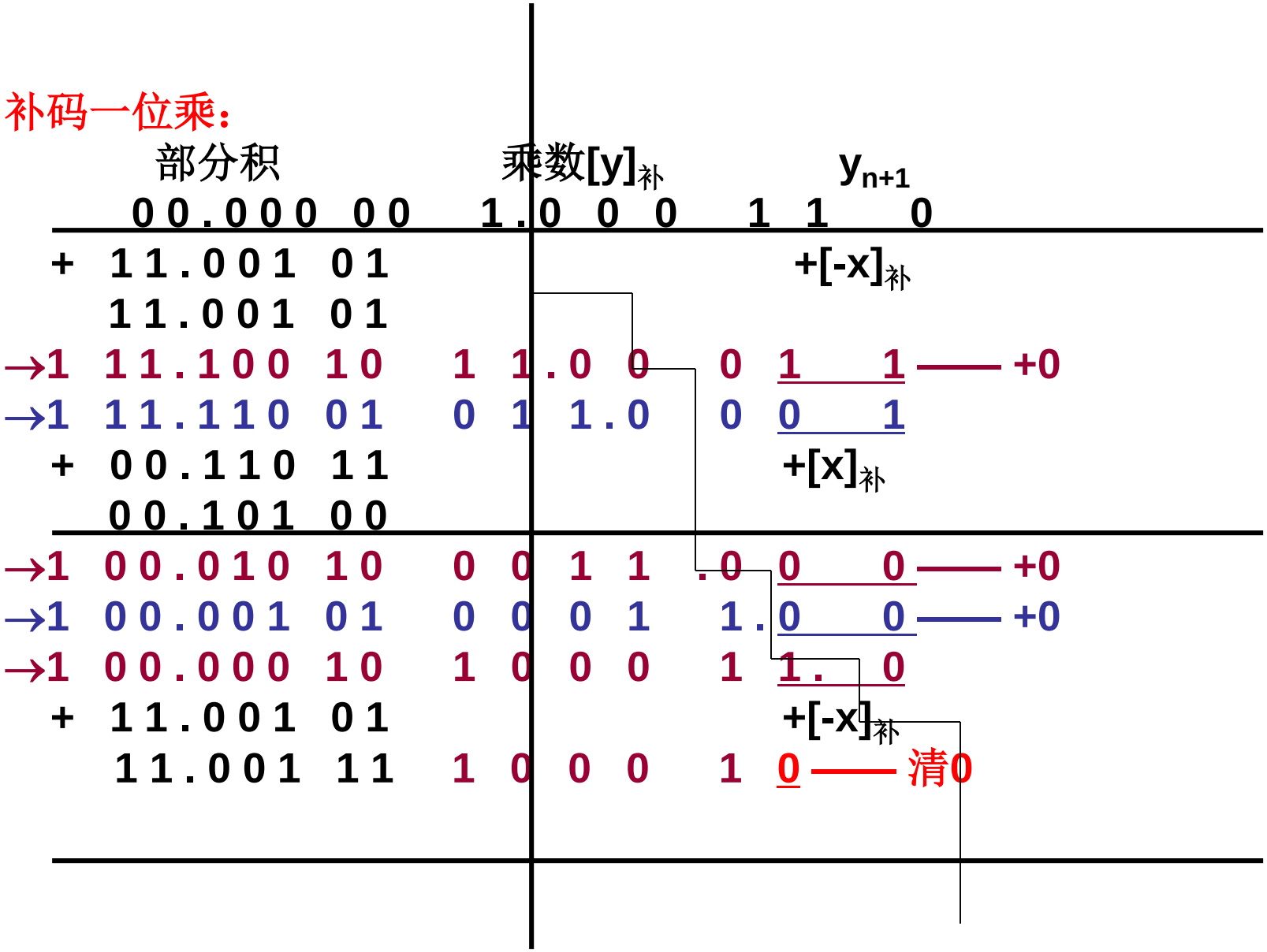
运算过程如下：


原码一位乘：部分积
乘数y*
0.000 00 .1 1 1 0 1 —— +x*
+ 0.110 11
0.110 11
→1 0.011 01 1. 1 1 0 —— +0
→1 0.001 10 1 1.1 1 1 —— +x*
+ 0.110 11
1.000 01
→1 0.100 00 1 1 1 .1 1 —— +x*
+ 0.110 11
1.010 11
→1 0.101 01 1 1 1 1.1 —— +x*
+ 0.110 11
1.100 00
→1 0.110 00 0 1 1 1 1

原码两位乘：

	部分积	乘数y*	Cj
	000.000 00	0.1 1 1 0 1	0
+	000.110 11	+x*	
	000.110 11		0
→2	000.001 10	1 1 0.1 1 1	
+	111.001 01	+[-x*]补	
	111.010 11		1
→2	111.110 10	1 1 1 1 .0 1	
+	001.101 10	+2x*	
	001.100 00		0
→1	000.110 00	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
00.000 00
1.0 0 0 1 1 0
+ 11.001 01
+[-x]补
11.001 01
→1 11.100 10
1 1.0 0 0 1 1
+0
→1 11.110 01
0 1 1.0 0 0 1
+ 00.110 11
+[x]补
00.101 00
→1 00.010 10
0 0 1 1 .0 0 0
+0
→1 00.001 01
0 0 0 1 1.0 0
+0
→1 00.000 10
1 0 0 0 1 1. 0
+ 11.001 01
+[-x]补
11.001 11
1 0 0 0 1 0
清0

补码两位乘：

	部分积	乘数	y_{n+1}
	000.000 00	1.0 0	0 1 1 0
+	111.001 01		+[-x]补
	111.001 01		
→2	111.110 01	0 1 1.0	0 0 1
+	000.110 11		+[x]补
	000.101 00		
→2	000.001 01	0 0 0 1	1.0 0
+	110.010 10		+[-2x]补
	110.011 11		
→1	111.001 11	1 0 0 0	1 0. —— 清0

结果同补码一位乘， $x \cdot y = -0.110\ 000\ 111\ 10$

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）$x^*=[x]_{原}=[x]_{补}=x=0.100\ \ 111$

$y^*=[y]_{原}=[y]_{补}=y=0.101\ \ 011$

$[-y^*]_{补}=[-y]_{补}=1.010\ \ 101$

$q_0=x_0 \oplus y_0=0 \oplus 0=0$

$x \div y=x^* \div y^*=[x \div y]_{原}=0.111\ \ 010$

$r^*=0.000\ 010 \times 2^{-6}=0.000\ 000\ 000\ 010$

计算过程如下：

原码加减交替除法：

被除数（余数）	商
0.100 111	0.000 000
+ 1.010 101	试减，+[-y*]补
1.111 100	
1← 1.111 000	0.
+ 0.101 011	r<0，+y*
0.100 011	
1← 1.000 110	0.1
+ 1.010 101	r>0，+[-y*]补
0.011 011	
1← 0.110 110	0.1 1
+ 1.010 101	r>0，+[-y*]补
0.001 011	

续：

	被除数（余数）	商
1←	0.010 110	0.111
+	1.010 101	r>0，+[-y*]补
	1.101 011	
1←	1.010 110	0.1 110
+	0.101 011	r<0，+y*
	0.000 001	
1←	0.000 010	0.11 101
+	1.010 101	r>0，+[-y*]补
	1.010 111	1← 0.111 010
+	0.101 011	r<0，+y*（恢复余数）
	0.000 010	

补码加减交替除法：

被除数（余数）	商
00.100 111	0.000 000
+ 11.010 101	试减，x、y同号，+[-y]补
11.111 100	
1← 11.111 000	0.
+ 00.101 011	r、y异号，+[y]补
00.100 011	
1← 01.000 110	0.1
+ 11.010 101	r、y同号， +[-y]补
00.011 011	
1← 00.110 110	0.1 1
+ 11.010 101	r、y同号， +[-y]补
00.001 011	

续：　被除数（余数）　　商

1←　00.010　110　　0.111

+　11.010　101　　r、y同号，　$+[-y]_{补}$

11.101　011

1←　11.010　110　　0.1　1 1 0

+　00.101　011　　r、y异号，$+[y]_{补}$

00.000　001

1←　00.000　010　　0.1 1　1 0 1

+　11.010　101　　r、y同号，　$+[-y]_{补}$

11.010　111　　1← 0.1 1 1　0 1 1 —— 恒置1

+　00.101　011　　r、x异号，（恢复余数）

00.000　010　　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

$q_0 = x_0 \oplus y_0 = 1 \oplus 0 = 1$

x*÷y*= 0.110 00

$[x \div y]_{原}$=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	
0.101 01	0.000 00	
+ 1.001 01		试减，+[-y*]补
1.110 10		
1← 1.101 00	0.	
+ 0.110 11		r<0，+y*
0.011 11		
1← 0.111 10	0.1	
+ 1.001 01		r>0，+[-y*]补
0.000 11		
1← 0.001 10	0.11	
+ 1.001 01		r>0，+[-y*]补
1.010 11		

续：

	被除数（余数）	商
1←	0.101 10	0.110
+	0.110 11	r<0，+y*
	1.100 01	
1←	1.000 10	0.1 100
+	0.110 11	r<0，+y*
	1.111 01	1← 0.11 000
+	0.110 11	r<0，+y*（恢复余数）
	0.110 00	

被除数（余数）	商	
11.010 11	0.000 00	
+ 00.110 11		试减，x、y异号，+[y]补
00.001 10		
1← 00.011 00	1.	
+ 11.001 01		r、y同号，+[-y]补
11.100 01		
1← 11.000 10	1.0	
+ 00.110 11		r、y异号，+[y]补
11.111 01		
1← 11.110 10	1.0 0	
+ 00.110 11		r、y异号，+[y]补
00.101 01		

续：

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

$x^*= [x]_{原}= [x]_{补}= x=0.101\ 00$

$[y]_{原} = 1.100\ 01$

$y^* = 0.100\ 01$

$[-y^*]_{补}=1.011\ 11$

$[y]_{补}= 1.011\ 11$

$[-y]_{补}= 0.100\ 01$

$q_0 = x_0 \oplus y_0 = 0 \oplus 1 = 1$

$x^*÷y^*= 1.001\ 01$ —— 溢出

$[x÷y]_{原}$：无定义

$x÷y = -1.001\ 01$

$r^*=0.010\ 11\times2^{-5}$

$=0.000\ 000\ 101\ 1$

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		
+	1.011 11	r>0，	
	1.101 01		
1←	1.010 10	1.0	
+	0.100 01	r<0， +y*	
	1.110 11		
1←	1.101 10	1.0 0	
+	0.100 01	r<0， +y*	
	0.001 11		

注：溢出，可停止运算，转溢出处理。

续：

	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← 1.00 101
		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	

续：

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，$+[-y]_{补}$
00.010 11	1← 0.1 1011 —— 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：$q_f = x_0 \oplus y_0 = 0 \oplus 1 = 1$

$[x \div y]_{补}$=10.110 11，x÷y= -1.001 01

判溢出：$q_f \oplus q_0 = 1 \oplus 0 = 1$，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）$x=13/32=(0.011\ \ 01)_2$

$y=-27/32=(-0.110\ \ 11)_2$

$x^*=[x]_{原}=[x]_{补}=x=0.011\ \ 01$

$[y]_{原}=1.110\ \ 11$

$y^*=0.110\ \ 11$

$[-y^*]_{补}=1.001\ \ 01$

$[y]_{补}=1.001\ \ 01$

$[-y]_{补}=0.110\ \ 11$

$q_0=x_0 \oplus y_0=0 \oplus 1=1$

$x^* \div y^*=0.011\ \ 11$

$[x \div y]_{原}=1.011\ \ 11$

$x \div y=(-0.011\ \ 11)_2=-15/32$

$r^*=0.010\ \ 11 \times 2^{-5}$

$=0.000\ \ 000\ \ 101\ \ 1$

原码加减交替除法：

被除数（余数）	商
0.011 01	0.000 00
+ 1.001 01	试减，+[-y*]补
1.100 10	
1← 1.001 00	0.
+ 0.110 11	r<0， +y*
1.111 11	
1← 1.111 10	0.0
+ 0.110 11	r<0， +y*
0.110 01	
1← 1.100 10	0.0 1
+ 1.001 01	r>0， +[-y*]补
0.101 11	

续：

	被除数（余数）	商
1←	1.011 10	0.011
+	1.001 01	r>0，+[-y*]补
	0.100 11	
1←	1.001 10	0.0 11 1
+	1.001 01	r>0，+[-y*]补
	0.010 11	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	
00.011 01	0.000 00	
+ 11.001 01	试减，x、y异号，	$+[y]_{补}$
11.100 10		
1← 11.001 00	1.	
+ 00.110 11	r、y同号，	$+[-y_{补}$
11.111 11		
1← 11.111 10	1.1	
+ 00.110 11	r、y同号，	$+[-y_{补}$
00.110 01		
1← 01.100 10	1.10	
+ 11.001 01	r、y异号，	$+[y_{补}$
00.101 11		

续：

被除数（余数）	商
1← 01.011 10	1.100
+ 11.001 01	r、y异号， $+[y]_{补}$
00.100 11	
1← 01.001 10	1.1 000
+ 11.001 01	r、y异号，$+[y]_{补}$
00.010 11	1← 1.1 0001 —— 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、原码加减交替除和补码加减交替除法最多各需多少时间？

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

补码加减交替除最多需时=

=1μs×（16+1）+1μs×15=32μs

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）$[x]_{补}=1,\ 101;\ 0.101\ 100$

$[y]_{补}=1,\ 110;\ 1.100\ 100$

注：为简单起见，源操作数可直接写成浮点格式，不必规格化。

1）对阶：

$[\Delta E]_{补}=[Ex]_{补}+[-Ey]_{补}$

=11，101+00，010=11，111

$[\Delta E]_{补}<0$，应Ex向Ey对齐，则：

$[Ex]_{补}+1$=11，101+00，001

=11，110

$[\Delta E]_{补}+1$=11，111+00，001

=00，000=0

至此， Ex=Ey，对毕。

$[x]_{补}$=1，110；0.010 110

2）尾数运算：

$[Mx]_{补}+[My]_{补}$= 0 0 . 0 1 0　1 1 0

+ 1 1 . 1 0 0　1 0 0

1 1 . 1 1 1 0 1 0

$[Mx]_{补}+[-My]_{补}$= 0 0 . 0 1 0　1 1 0

+ 0 0 . 0 1 1　1 0 0

0 0 . 1 1 0　0 1 0

3）结果规格化：

$[x+y]_{补}$=11，110；11.111 010

=11，011；11.010 000

（左规3次，阶码减3，尾数左移3位）

$[x-y]_{补}$=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），则

$[x]_{补}=1，110；1.101\ \ 111$

2）尾数运算：

$$[Mx]_{补}+[My]_{补}=11.101\ \ 111$$
$$+\ 11.100\ \ 001$$
$$11.010\ \ 000$$

$$[Mx]_{补}+[-My]_{补}=11.101\ \ 111$$
$$+\ 00.011\ \ 111$$
$$00.001\ \ 110$$

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；00.111 000

（左规2次，阶码减2，尾数左移2位）

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：

$[\Delta E]_{补}=[Ex]_{补}+[-Ey]_{补}$

=00，101+11，100=00，001

$[\Delta E]_{补}>0$，应Ey向Ex对齐，则：

$[Ey]_{补}+1$=00，100+00，001

=00，101

$[\Delta E]_{补}+[-1]_{补}$=00，001+11，111

=00，000=0

至此， Ey=Ex，对毕。

$[y]_{补}$=0，101；1.111 000（1）

2）尾数运算：

$[Mx]_{补}+[My]_{补}$= 1 1 . 0 1 1　0 1 1

+ 1 1 . 1 1 1　0 0 0（1）

1 1 . 0 1 0　0 1 1（1）

$[Mx]_{补}+[-My]_{补}$= 1 1 . 0 1 1　0 1 1

+ 0 0 . 0 0 0　1 1 1（1）

1 1 . 1 0 0　0 1 0（1）

3）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

（左规1次，阶码减1，尾数左移1位）

4）舍入：

$[x+y]_{补}$=00，101；11.010 011（舍）

$[x-y]_{补}$不变。

$[x-y]_{补}$=00，100；11.000 101

5）溢出：无

则：$x+y=2^{101}\times$（-0.101 101）

$x-y=2^{100}\times$（-0.111 011）

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）$[2^5\times(11/16)]+[2^4\times(-9/16)]$

（2）$[2^{-3}\times(13/16)]-[2^{-4}\times(-5/8)]$

（3）$[2^3\times(13/16)]\times[2^4\times(-9/16)]$

（4）$[2^6\times(-11/16)]\div[2^3\times(-15/16)]$

（5）$[2^3\times(-1)]\times[2^{-2}\times57/64]$

（6）$[2^{-6}\times(-1)]\div[2^7\times(-1/2)]$

（7）3.3125+6.125

（8）14.75-2.4375

解：设机器数采用阶补尾补形式：

（1）$x=2^5\times(11/16)=2^{101}\times0.101100$

$y=2^4\times(-9/16)=2^{100}\times(-0.100100)$则：$[x]_{阶补尾补}$=00，101；00.101100

$[y]_{阶补尾补}$=00，100；11.011100

1）对阶：

$[\Delta E]_{补}=[Ex]_{补}+[-Ey]_{补}$

=00，101+11，100=00，001

$[\Delta E]_{补}>0$，应Ey向Ex对齐，则：

$[Ey]_{补}+1$=00，100+00，001=00，101

$[\Delta E]_{补}+[-1]_{补}$=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：$x+y=2^{100}\times(0.110\ 100)$

$=2^{4}\times(13/16)$

（2）$[2^{-3}\times(13/16)]-[2^{-4}\times(-5/8)]$

$x=2^{-3}\times(13/16)=2^{-011}\times 0.110\ 100$

$y=2^{-4}\times(-5/8)=2^{-100}\times(-0.101000)$

$[x]_{阶补尾补}=11，101；00.110100$

$[y]_{阶补尾补}=11，100；11.011000$

1）对阶：

$[\Delta E]_{补}=[Ex]_{补}+[-Ey]_{补}$

$=11，101+00，100=00，001$

$[\Delta E]_{补}>0$，应Ey向Ex对齐，则：

$[Ey]_{补}+1=11，100+00，001=11，101$

$[\Delta E]_{补}+[-1]补=00，001+11，111=0$

至此， Ey=Ex，对毕。

$[y]_{补}=11，101；11.101100$

2）尾数运算：

$$[Mx]_{补}+[-My]_{补}= \begin{array}{r} 00.110100 \\ +\ 00.010100 \\ \hline 01.001000 \end{array}$$

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。

5）溢出：无

则：$x-y=2^{-010}\times（0.100\ 100）$

$=2^{-2}\times（9/16）$

（3）$[2^3\times(13/16)]\times[2^4\times(-9/16)]$

$x=2^3\times(13/16)=2^{011}\times$（0.110 100）

$y=2^4\times(-9/16)=2^{100}\times$（-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。

$[Mx\times My]_{补}$=11.100 010（110 000 00）

3）结果规格化：左规1位。

$[x\times y]_{补}$=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	y_{n+1}
000.000 000	11.011	100 0
+ 000.000 000		$+[-0]_{补}$
000.000 000		
→2 000.000 000	00 11.0	1 1 1 0
+ 111.001 100		$+[-x]_{补}$
111.001 100		
→2 111.110 011	00 00 1	1.0 1 1
+ 001.101 000		$+[2x]_{补}$
001.011 011		
→2 000.010 110	11 000	0 1 1 .0
+ 111.001 100		$+[-x]_{补}$
111.100 010	11 000	0 0 0(清0)

4）舍入：设采用0舍1入法，应舍：

$[x\times y]_{阶补尾补}$=0，110；1.000 101

5）溢出：无

$x\times y=2^{110}\times(-0.111\ \ 011)$

$=2^{6}\times(-59/64)$

（4） $[2^{6}\times(-11/16)]\div[2^{3}\times(-15/16)]$

$x=2^{6}\times(-11/16)=2^{110}\times(-0.101\ \ 100)$

$y=2^{3}\times(-15/16)=2^{011}\times(-0.111\ \ 100)$

$[x]_{阶补尾补}$=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	
11.010 100	0.000 000	试减，
+ 00.111 100		Mx、My同号，+[-My]补
00.010 000		
1← 00.100 000	0.	
+ 11.000 100		r、My异号，+[My]补
11.100 100		
1← 11.001 000	0.1	
+ 00.111 100		r、My同号，+[-My]补
00.000 100		
1← 00.001 000	0.10	
+ 11.000 100		r、My异号，+[My]补
11.001 100		

续： 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 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， $+[My]_{补}$

$[Mx \div My]_{补} = 0.101\ 111$， $[r]_{补} = 1.001\ 000$

$r = -0.111\ 000 \times 2^{-6} = -0.000\ 000\ 111\ 000$

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$x \cdot y$，且结果保留1倍字长。

（1）$x=2^{-100}\times 0.101101$，

$y=2^{-011}\times(-0.110101)$；

（2）$x=2^{-011}\times(-0.100111)$，

$y=2^{101}\times(-0.101011)$。

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：（算法一：补码两位乘比较法）

部分积	乘数	y_{n+1}
000.000 000	11.001	0 11 0
+ 111.010 011		+[-x]补
111.010 011		
→2 111.110 100	11 11.0	0 10 1
+ 111.010 011		+[-x]补
111.000 111		
→2 111.110 001	11 11 1.0	0 1
+ 000.101 101		+[x]补
000.011 110		
→2 000.000 111	10 11 1	1 1 .0
+ 111.010 011		+[-x]补
111.011 010	10 11 1	1 0 0(清0)

2）尾数相乘：（算法二：补码一位乘比较法）

部分积	乘数	y_{n+1}
00.000 000	1.001 011	0
+ 11.010 011		$+[-x]_{补}$
11.010 011		
→1 11.101 001	1 1.00 101	1 — +0
→1 11.110 100	1 1 1.0 010	1
+ 00.101 101		$+[x]_{补}$
00.100 001		
→1 00.010 000	1 1 1 1. 001	0
+ 11.010 011		$+[-x]_{补}$
11.100 011		
→1 11.110 001	1 1 1 1 1.00	1
+ 00.101 101		$+[x]_{补}$
00.011 110		
→1 00.001 111	0 1 1 1 1 1.0	0 — +0
→1 00.000 111	1 0 1 1 1 1 1.	0
+ 11.010 011		$+[-x]_{补}$
11.011 010	1 0 1 1 1 1 0.	— (清0)

$[Mx \times My]_{补}$=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：

$[x \times y]_{阶移尾补}$=0，001；1.011 011

5）溢出：无

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

（2）$x = 2^{-011} \times$（-0.100 111）

$y = 2^{101} \times$（-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：（算法一：补码两位乘比较法）

部分积	乘数	y_{n+1}
000.000 000	11.010	10
+ 111.011 001		$+[x]_{补}$
111.011 001		
→2 111.110 110	01 11.0	10
+ 111.011 001		$+[x]_{补}$
111.001 111		
→2 111.110 011	11 01 1	1.010
+ 111.011 001		$+[x]_{补}$
111.001 100		
→2 111.110 011	00 11 0	111.0
+ 000.100 111		$+[-x]_{补}$
000.011 010	00 11 0	10 0(清0)

$[Mx \times My]_{补}$=0.011 010（001 101 00）

2）尾数相乘：（算法二：补码一位乘比较法）

部分积	乘数	y_{n+1}
00.000 000	1.0 1 0 1 0 1 0	
+ 00.100 111		+[-x]补
00.100 111		
→1 00.010 011	1 1.0 1 0 1 0 1	
+ 11.011 001		+[x]补
11.101 100		
→1 11.110 110	0 1 1.0 1 0 1 0	
+ 00.100 111		+[-x]补
00.011 101		
→1 00.001 110	1 0 1 1. 0 1 0 1	
+ 11.011 001		+[x]补
11.100 111		
→1 11.110 011	1 1 0 1 1.0 1 0	
+ 00.100 111		+[-x]补
00.011 010		
→1 00.001 101	0 1 1 0 1 1.0 1	
+ 11.011 001		+[x]补
11.100 110		
→1 11.110 011	0 0 1 1 0 1 1. 0	
+ 00.100 111		+[-x]补
00.011 010	0 0 1 1 0 1 0. — (清0)	

$[Mx \times My]_{补}$=0.011 010（001 101 0）

3）结果规格化：

$[x\times y]_{阶移尾补}=$

$=1，010；0.011\ \ 010（001\ \ 101\ \ 00）$

$=1，001；0.110\ \ 100（011\ \ 010\ \ 0）$

（左规1次，阶码减1，尾数左移1位）

4）舍入：设采用0舍1入法，应舍：

$[x\times y]_{阶移尾补}=1，001；0.110\ \ 100$

5）溢出：无

$x\times y=2^{001}\times 0.110\ \ 100$

注意：采用阶移尾补格式是指：参加运算的数是阶移尾补格式，用阶移尾补算法计算，运算结果是阶移尾补格式。

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。（注：改用阶移尾原格式作。）

（1）$x=2^{101}\times 0.100111$，

$y=2^{011}\times(-0.101011)$；

（2）$x=2^{110}\times(-0.101101)$，

$y=2^{011}\times(-0.111100)$。

解：先将x、y转换成机器数形式：

（1）$[x]_{阶移尾原}$=1，101；0.100 111

$[y]_{阶移尾原}$=1，011；1.101 011

1）阶码相减：

$[Ex]_{移}+[-Ey]_{补}$=01，101+11，101

=01，010（无溢出）

2）尾数相除：（原码加减交替除法）

被除数（余数）	商	
00.100 111	0.000 000	试减，
+ 11.010 101		+[-My*]补
11.111 100		r＜0，商0
1← 11.111 000	0.	
+ 00.101 011		+My*
00.100 011		r＞0，商1
1← 01.000 110	0.1	
+ 11.010 101		+[-My*]补
00.011 011		r＞0，商1
1← 00.110 110	0.11	
+ 11.010 101		+[-My*]补
00.001 011		r＞0，商1

续： 被除数（余数）	商
1← 00.010 110	0.111
+ 11.010 101	+[-My*]补
11.101 011	r＜0，商0
1← 11.010 110	0.1 1 1 0
+ 00.101 011	+My*
00.000 001	r＞0，商1
1← 00.000 010	0.1 1 1 0 1
+ 11.010 101	+[-My*]补
11.010 111	1← 0.1 1 1 0 1 0，r＜0，商0
+ 00.101 011	（恢复余数）
00.000 010	且r、My同号， +[-My]补

Mx*÷My*= 0.111 010，$[Mx \div My]_{原}$= 1.111 010

r*= 0 .000 010 × 2^{-6} =0.000 000 000 010

3）结果规格化：已是规格化数。

4）舍入：已截断法舍入。

5）溢出：无

$[x \div y]_{阶移尾原}$=1，010；1.111 010

$x \div y = 2^{010} \times (-0.111\ 010)$

（2）$x = 2^{110} \times (-0.101\ 101)$

$y = 2^{011} \times (-0.111\ 100)$

$[x]_{阶移尾原}$=1，110；1.101 101

$[y]_{阶移尾原}$=1，011；1.111 100

1）阶码相减：

$[Ex]_{移}+[-Ey]_{补}$=01，110+11，101

=01，011（无溢出）

2）尾数相除：（原码加减交替除法）

被除数（余数）	商	
00.101 101	0.000 000	试减，
+ 11.000 100		$+[-My^*]_{补}$
11.110 001		r＜0，商0
1← 11.100 010	0.	
+ 00.111 100		$+My^*$
00.011 110		r＞0，商1
1← 00.111 100	0.1	
+ 11.000 100		$+[-My^*]_{补}$
00.000 000		r＞0，商1
1← 00.000 000	0.11	
+ 11.000 100		$+[-My^*]_{补}$
11.000 100		r＜0，商0

续： 被除数（余数）	商
1← 1 0 . 0 0 1 0 0 0	0 . 1 1 0
+ 0 0 . 1 1 1 1 0 0	+My*
1 1 . 0 0 0 1 0 0	r＜0，商0
1← 1 0 . 0 0 1 0 0 0	0.1 1 0 0
+ 0 0 . 1 1 1 1 0 0	+My*
1 1 . 0 0 0 1 0 0	r＜0，商0
1← 1 0 . 0 0 1 0 0 0	0.1 1 0 0 0
+ 0 0 . 1 1 1 1 0 0	+My*
1 1 . 0 0 0 1 0 0	1← 0.1 1 0 0 0 0，r＜0，商0
+ 0 0 . 1 1 1 1 0 0	恢复余数，+My*
0 0 . 0 0 0 0 0 0	

$Mx^*÷My^*= [Mx÷My]_{原}= 0.110\ \ 000$

$r^*= -0.000\ \ 000\times2^{-6} = -0.000\ 000\ 000\ 000$

注：由于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。

3）结果规格化：已是规格化数。

4）舍入：已截断法舍入。

5）溢出：无

$[x \div y]_{\text{阶移尾原}} = 1，011；0.110\ \ 000$

$x \div y = 2^{011} \times 0.110\ \ 000$

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：

$$T=2ty\times32=64ty=64\times30=1920ns$$

不满足0.6μs的加法时间限制，不能用。（设1ty=30ns）

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×8组=20ty=20×30=600ns

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。

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×6组=15ty=15×30=450ns

除进位外还有150ns（约5ty）左右的时间供加法开销，较充裕。

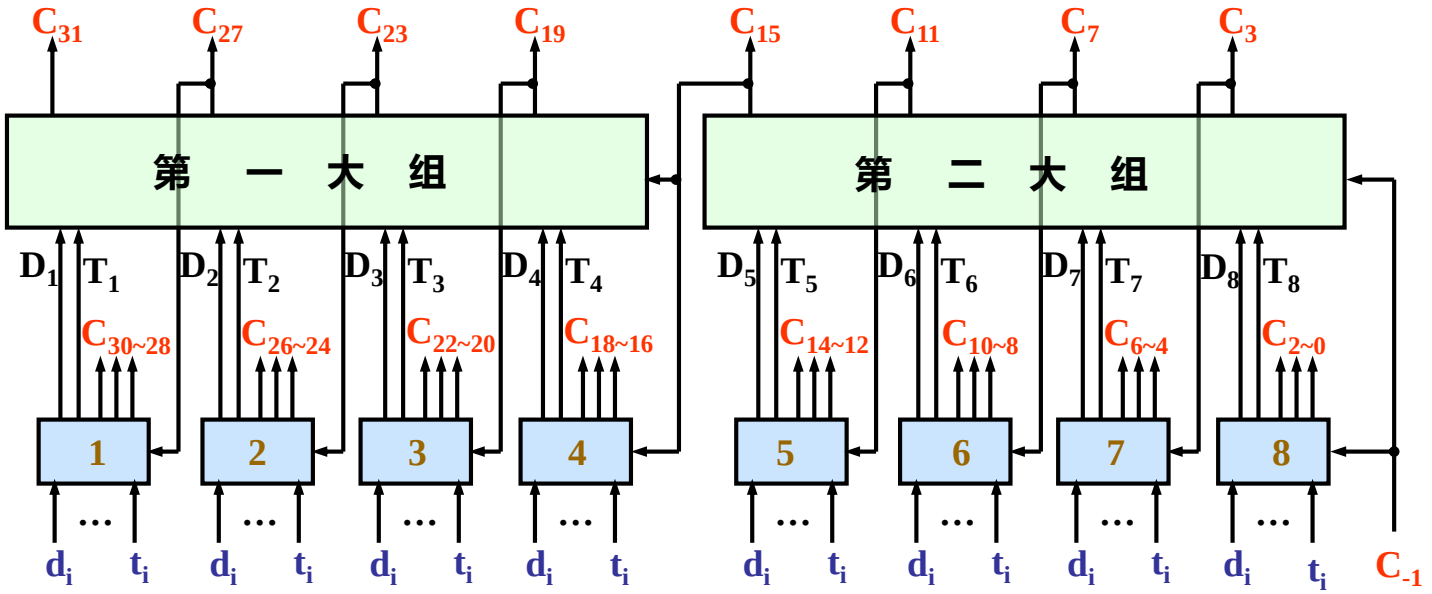
3）若采用双重分组跳跃进位（二级先行—级联进位），4位一小组，4小组为一大组分组，则32位进位需：

$T=2.5ty\times4级=10ty=10\times30=300ns$

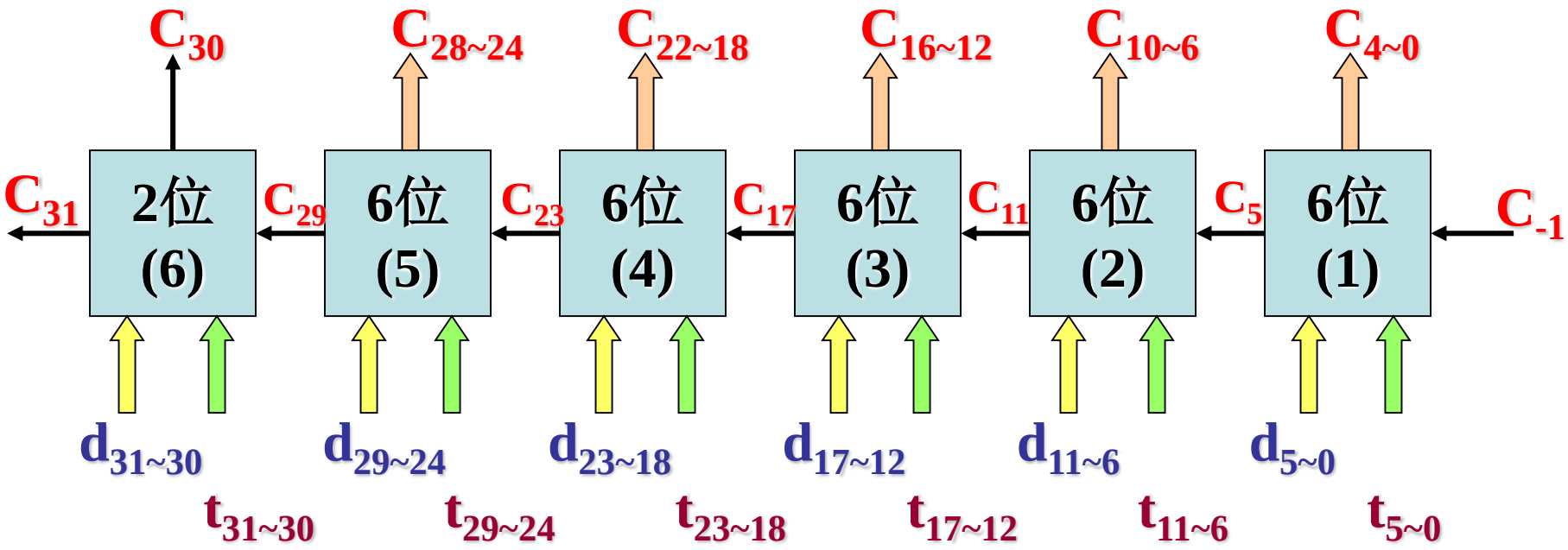
完全满足0.6μs的加法时间限制，可以使用。

双重分组跳跃进位
（两级先行进位）

32位双重分组跳跃进位的进位链框图见教材286页图6.23。

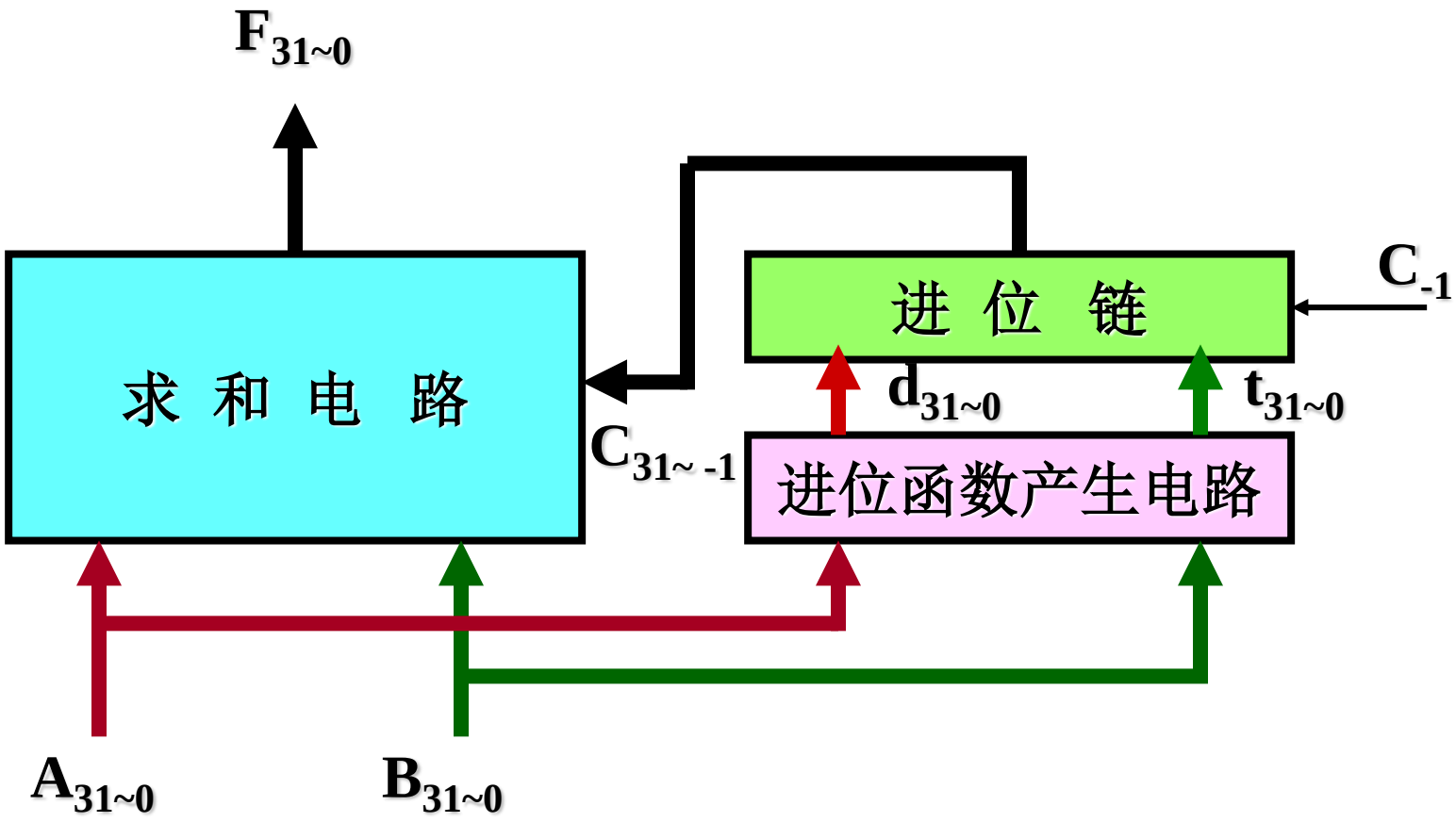


6位一组单重分组跳跃进位的进位链框图如下：



注：一个完整的加法器还应考虑di、ti产生电路、求和电路等。

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。

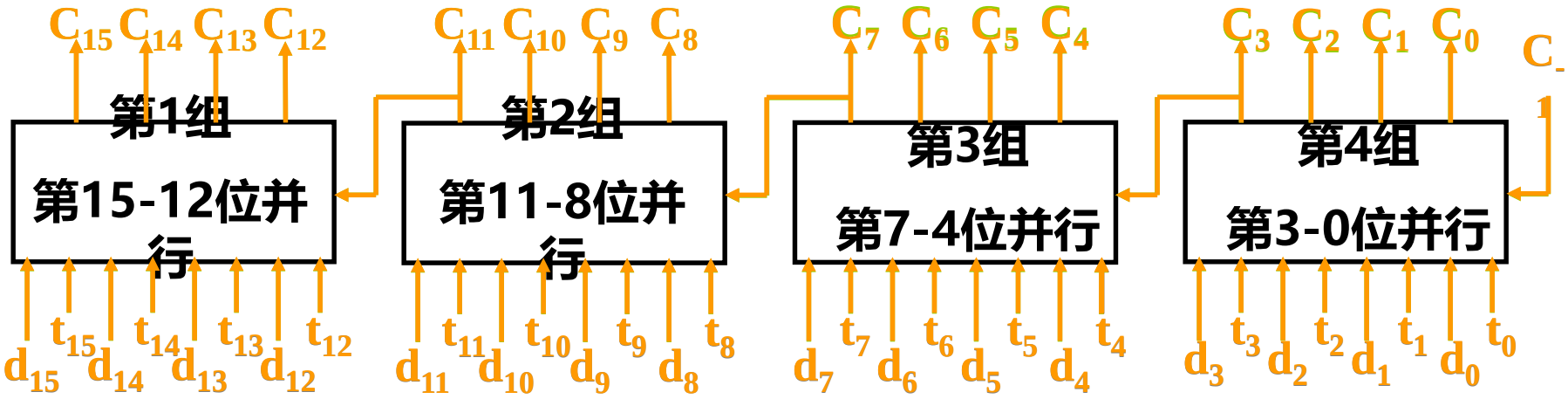
（3）用74181和74182画出单重和双重分组的并行进位链框图。

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。

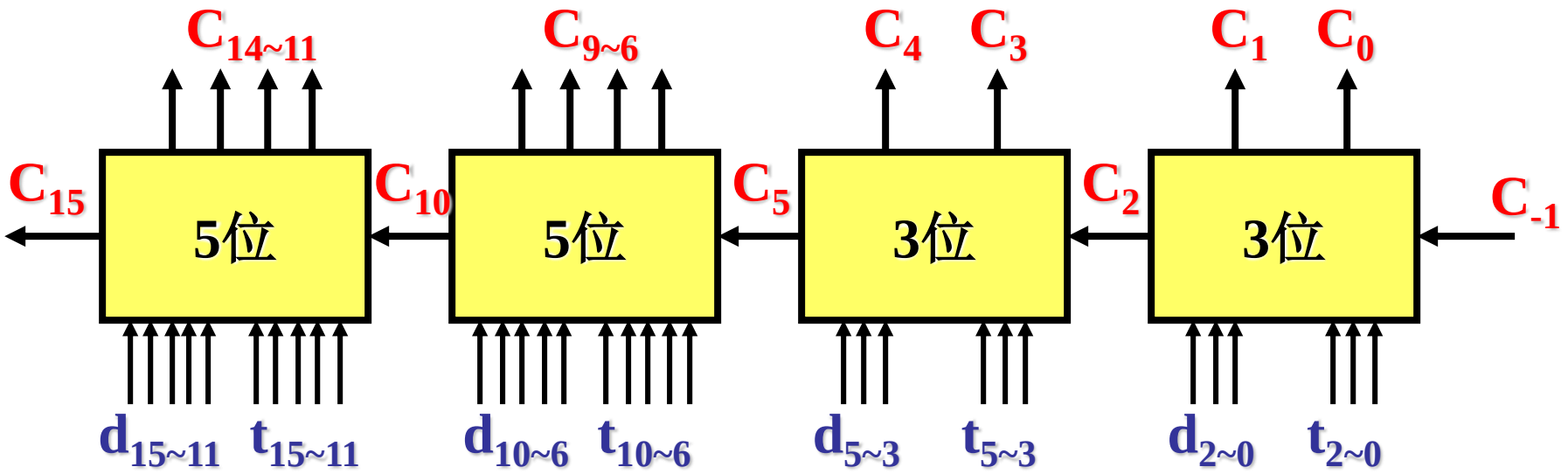
单重分组跳跃进位
（一级先行进位）

16位并行加法器进位链框图（6.22）



注意：16位一级先行进位加法器最长进位延迟时间为$10t_y$。

5—5—3—3分组的16位单重分组并行进位链框图如下：



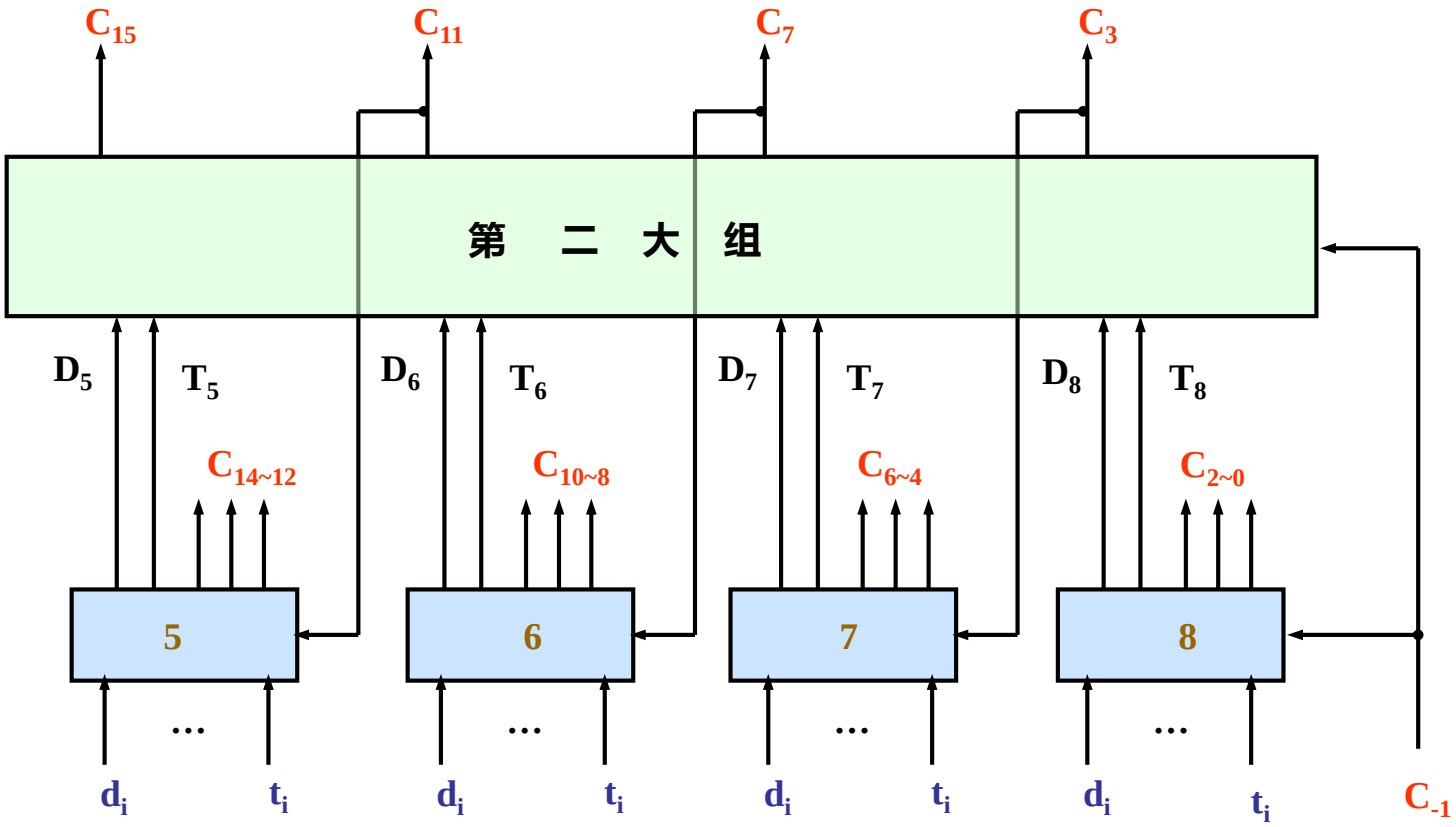
4—4—4—4分组的进位时间=2.5ty×4=10ty；

5—5—3—3分组的进位时间=2.5ty×4=10ty；

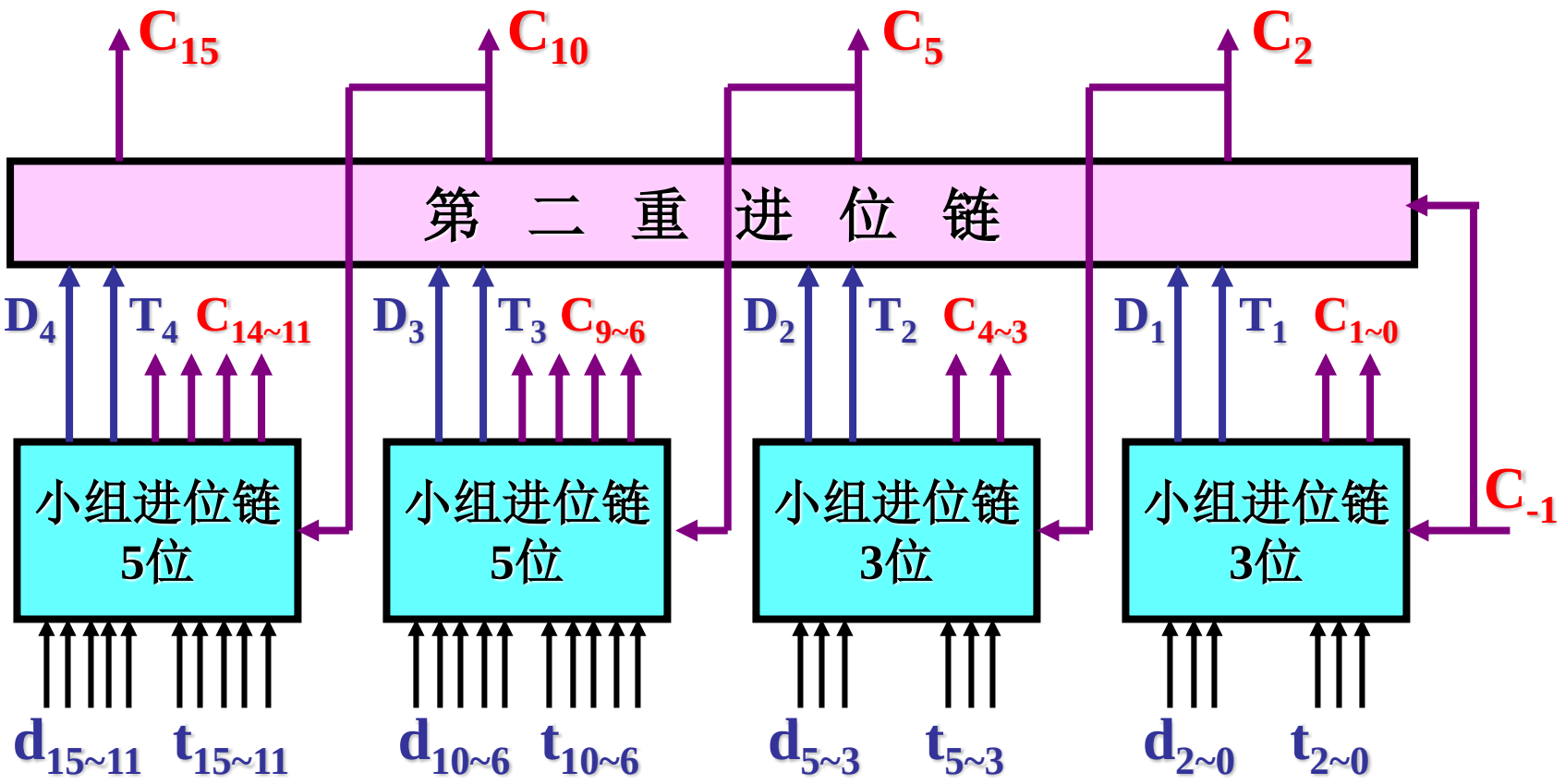
两种分组方案最长加法时间相同。

结论：单重分组并行进位的最长进位时间只与组数有关，与组内位数无关。

（2）4—4—4—4分组的16位双重分组并行进位链框图见教材288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：



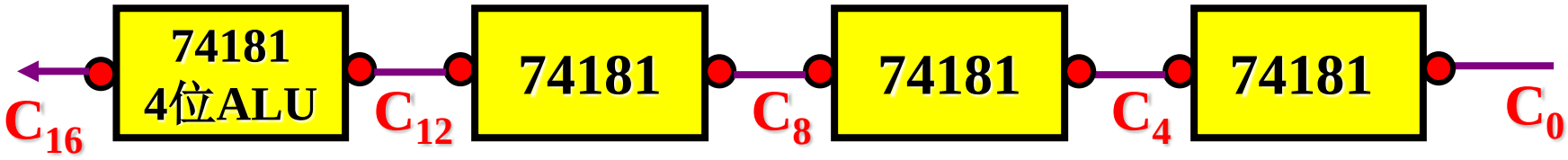
4—4—4—4分组的进位时间=2.5ty×3=7.5ty；

5—5—3—3分组的进位时间=2.5ty×3=7.5ty；

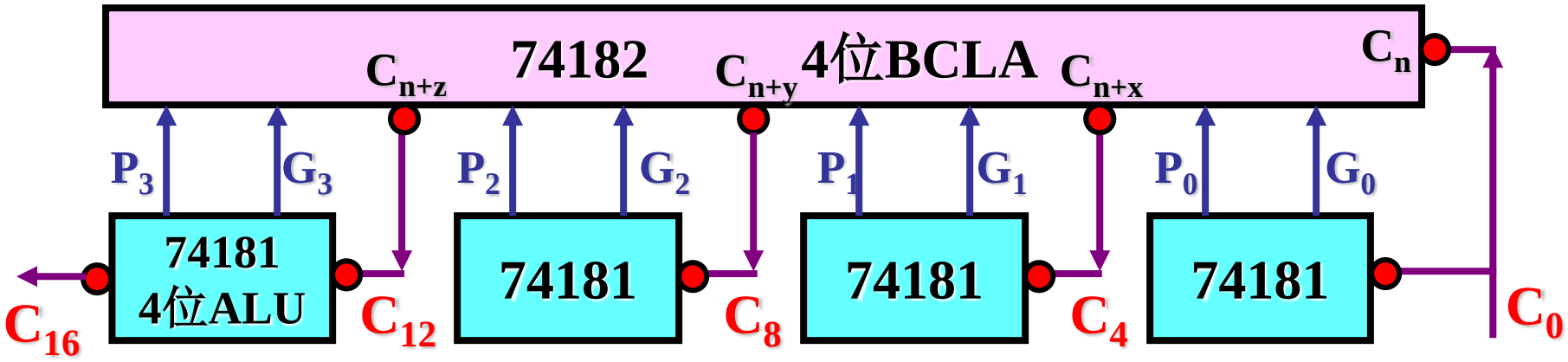
两种分组方案最长加法时间相同。

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）：



双重分组16位并行加法器逻辑图如下（正逻辑）：



图中，设与进位无关的引脚省略不画，不用的引脚也省略不画。

注意：

1）181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的一般是双重以上分组跳跃进位；

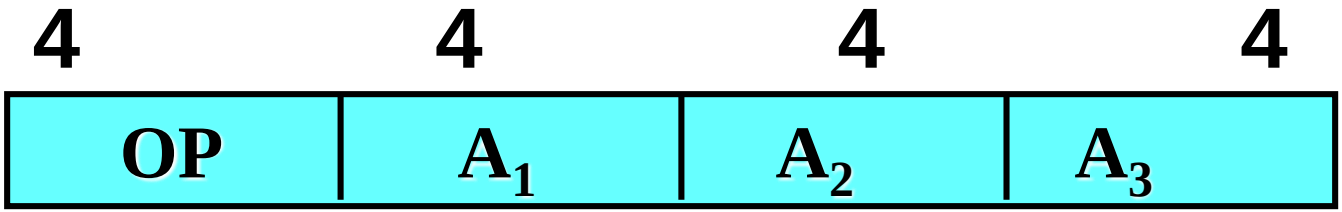
6）单重分组跳跃进位是并行进位和串行进位技术的结合；双重分组跳跃进位是二级并行进位技术；特别注意在位数较少时，双重分组跳跃进位可以采用全先行进位技术实现；位数较多时，可采用双重分组跳跃进位和串行进位技术结合实现。

指　令　系　统

第　七　章

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：



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：

指令操作码分配方案

4位OP

0000，
……，
0111，　} A_1，A_2，A_3；8条三地址指令

1000，0000，
……，……，
1000，1111，　} A_2，A_3；16条二地址指令

1001，0000，0000，
……，……，……，
1001，0110，0011，　} A3；100条一地址指令

1001，0110，0100，
……，……，……，
1001，1111，1111，　} 冗余编码

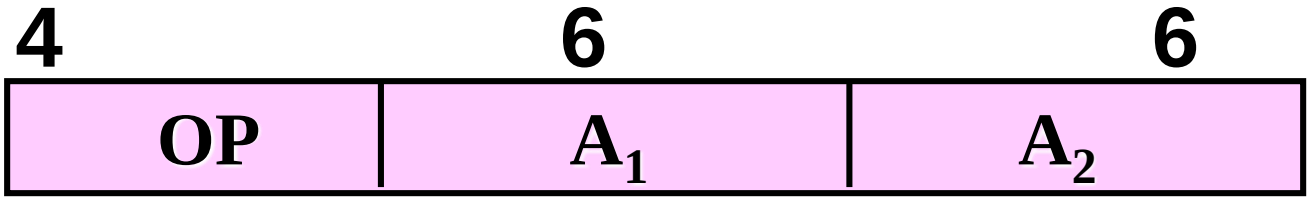
可用来扩充一、零地址指令条数

1010，
……，
1111，　} 冗余编码

可用来扩充三、二、一、零地址指令条数

7. 设指令字长为16位，采用扩展操作码技术，每个操作数的地址为6位。如果定义了13条二地址指令，试问还可安排多少条一地址指令？

解：二地址指令格式如下：



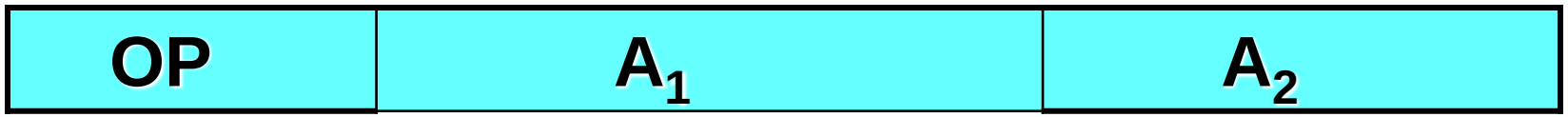
设二地址指令格式为该指令系统的基本格式，4位操作码共有16种编码，其中13种用来定义二地址指令，还剩3种可用作扩展标志。如不考虑零地址指令，该指令系统最多还能安排：

一地址指令条数 $=3\times2^6=192$条

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种，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：

4　　6　　6



此时，无论指令中有几个地址，指令格式都不变。

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：

$K_{max}=16-1-1=14$种

2）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，

$K=2^4-(N/2^6+M/2^{12})$；

（$N/2^6+M/2^{12}$ 向上取整）

当$(N/2^6+M/2^{12})\leq 1$时，K最大，

则二地址指令最多有：

$K_{max}=16-1=15$种（只留一种编码作扩展标志用。）

讨论：此时，一地址指令条数为：

$N=（2^4-K）\times 2^6-M/2^6$；

（$M/2^6$向上取整）。

零地址指令条数为：

$M=2^{16}-2^{12}K-2^6N$；

当K最大时（K=15），一地址指令最多有：

$N_{max}=64-1=63$种；

零地址指令最多有：

$M_{max}=64$种

注意：应首先根据题意画出指令基本格式。

10. 试比较基址寻址和变址寻址。

解：比较如下：

1）都可有效地扩大指令寻址范围。

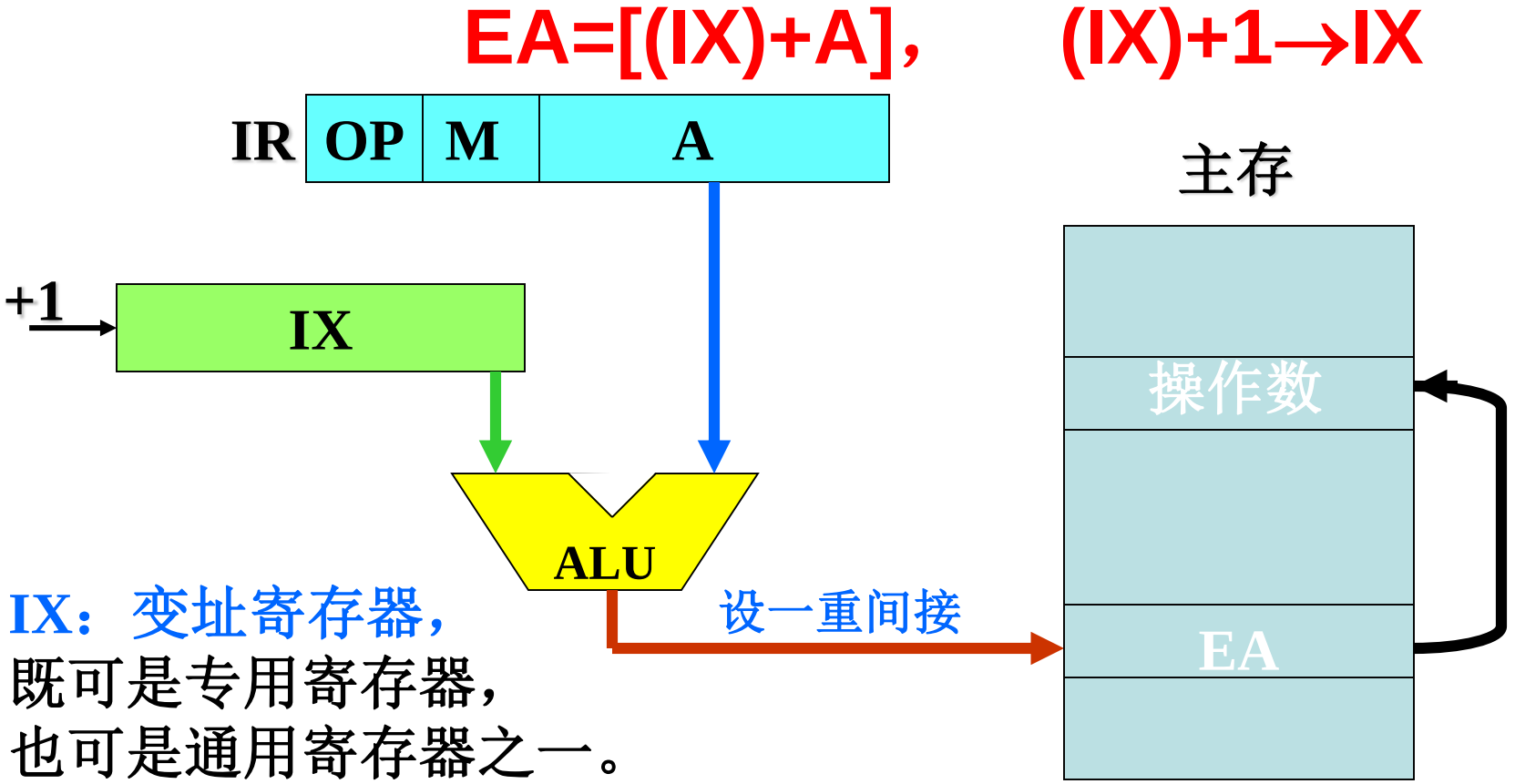
2）基址寻址时，基准地址由基址寄存器给出，地址的改变反映在位移量A的取值上；变址寻址时，基准地址由A给出，地址的改变反映在变址值的自动修改上，变址值由变址寄存器给出。

3）基址寄存器内容通常由系统程序设定，变址寄存器内容通常由用户设定。

4）基址寻址适用于程序的动态重定位，变址寻址适用于数组或字符串处理，适用场合不同。

11. 画出先变址再间址及先间址再变址的寻址过程示意图。

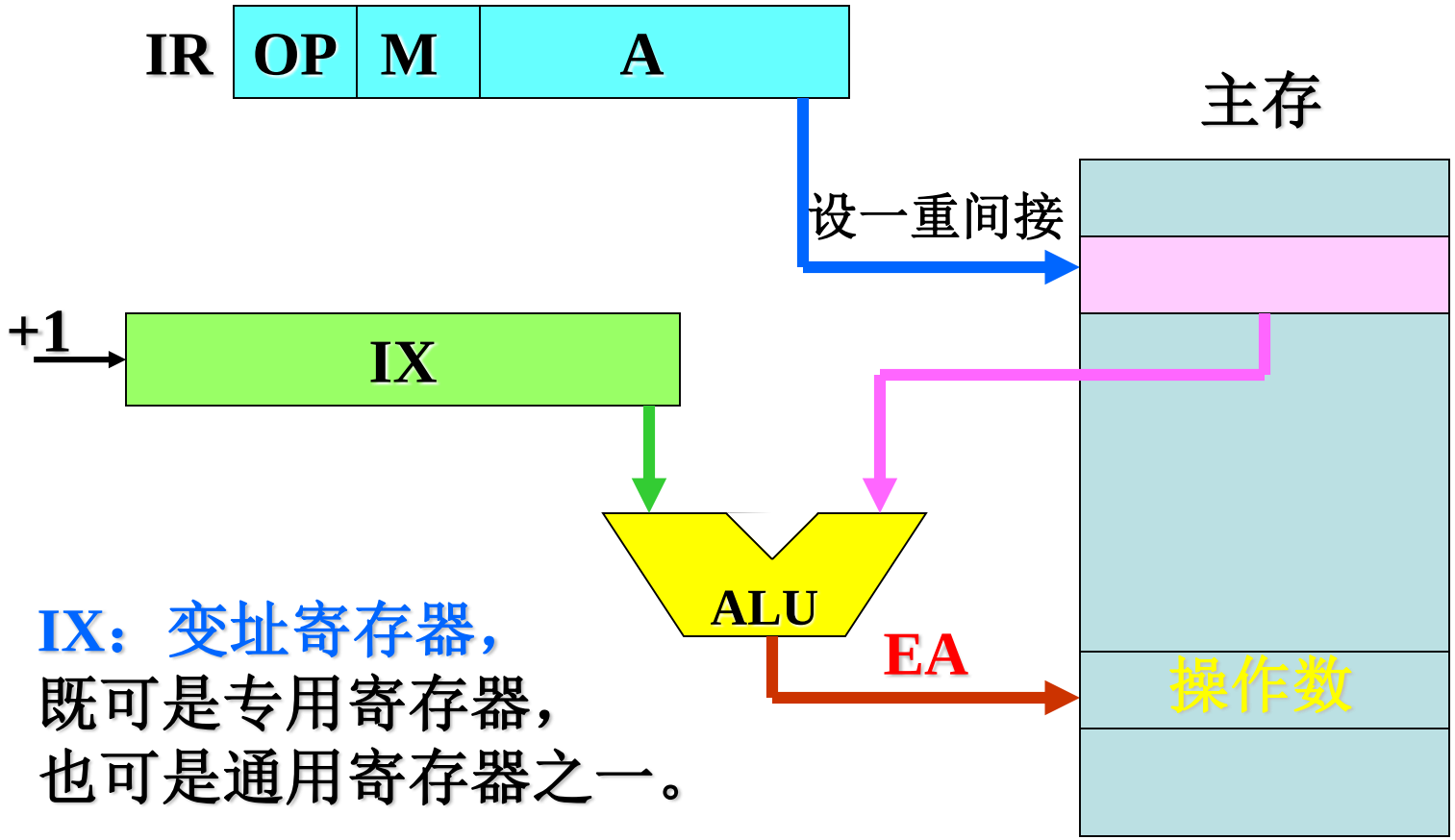
解：1）先变址再间址寻址过程简单示意如下：



IX：变址寄存器，
既可是专用寄存器，
也可是通用寄存器之一。

2）先间址再变址寻址过程简单示意如下：

EA=(IX)+(A)，　(IX)+1→IX



IX：变址寄存器，
既可是专用寄存器，
也可是通用寄存器之一。

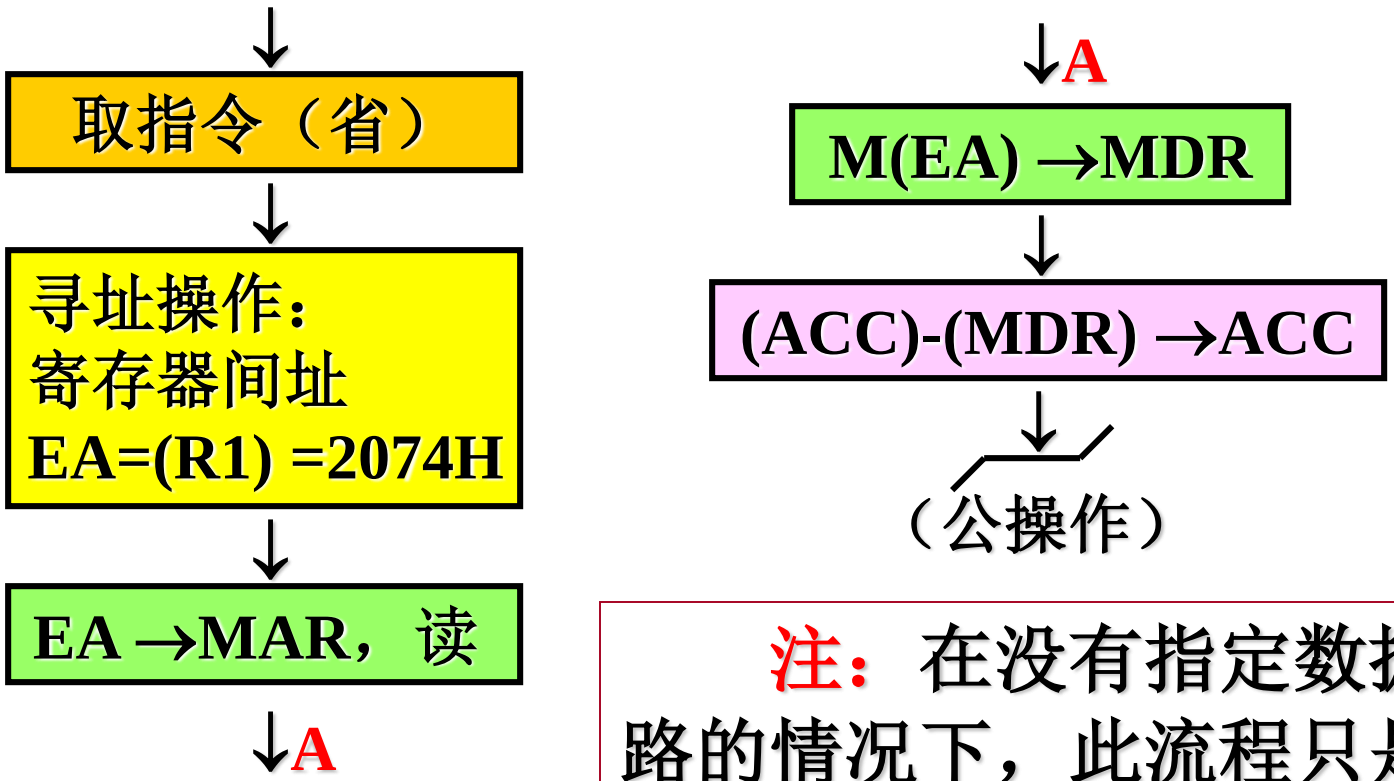
注意：

1）英文缩写EA表示有效地址，不能乱用。

2）示意图中应标明EA（有效地址）的位置。

12. 画出“**SUB @R1**”指令对操作数的寻址及减法过程的流程图。设被减数和结果存于**ACC**中，@表示间接寻址，**R1**寄存器的内容为**2074H**。

解：**SUB @R1**指令寻址及减法过程的流程图：

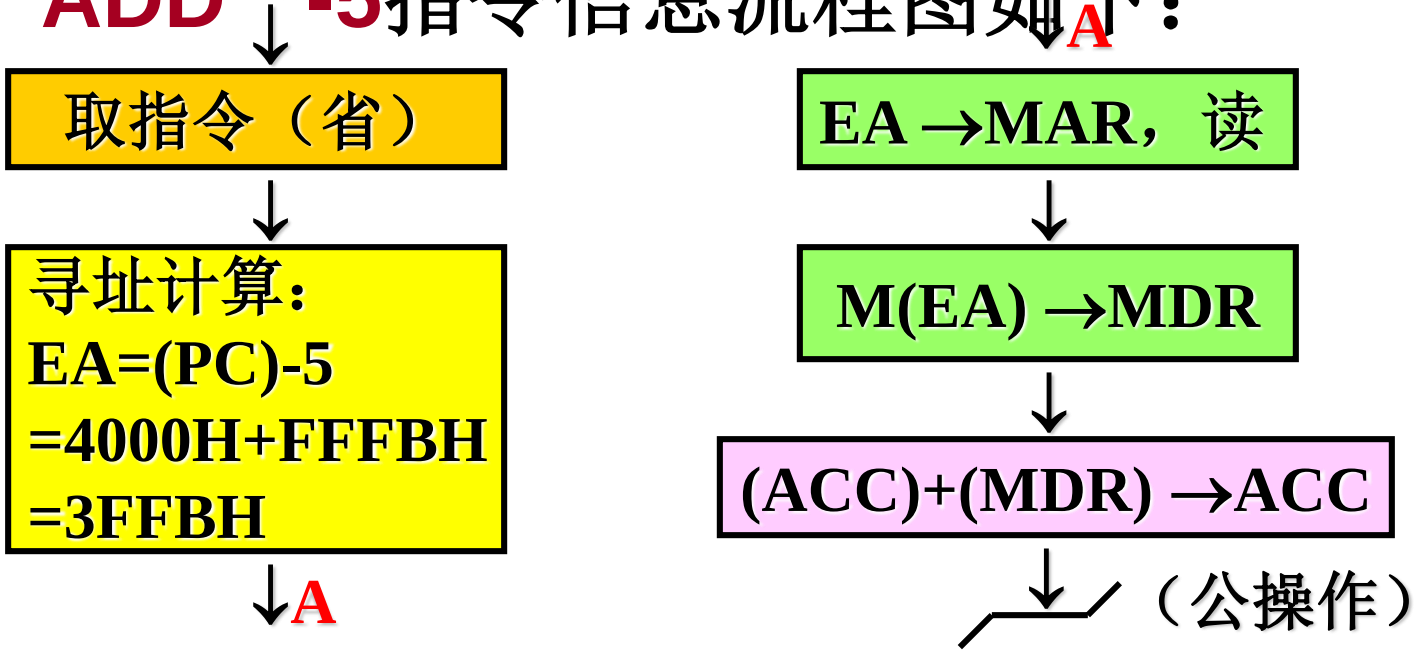


注：在没有指定数据通路的情况下，此流程只是一个粗略的示意。

13. 画出执行“**ADD *-5**”指令（*为相对寻址特征）的信息流程图。设另一个操作数和结果存于**ACC**中，并假设**（PC）=4000H**。

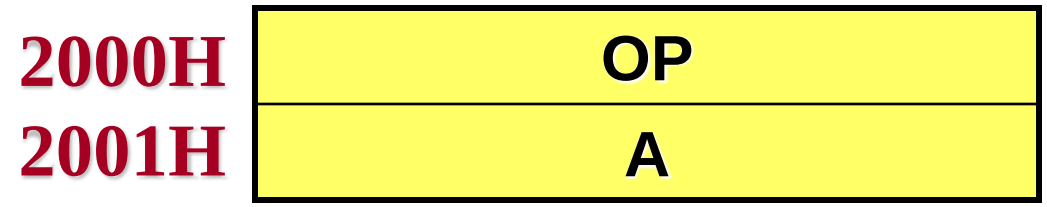
解：由于本题未指定数据通路结构，因此只能大概地排一下信息流程图，并且流程图中突出寻址过程的实现。

ADD *-5指令信息流程图如下：



14. 设相对寻址的转移指令占两个字节，第一个字节是操作码，第二个字节是相对位移量，用补码表示。假设当前转移指令第一字节所在的地址为**2000H**，且**CPU**每取出一个字节便自动完成（**PC**）**+1→PC**的操作。试问当执行“**JMP *+8**”和“**JMP *-9**”指令时，转移指令第二字节的内容各为多少？

解：据题意，相对寻址的转移指令格式如下：



当执行JMP指令时，指令第二字节的内容不变，PC的内容变为2002H。此时转移指令第二字节内容各为：

A1= +8 = 0000 1000 = 08H

A2= -9 = 1111 0111 = F7H

其有效地址各为：

EA1= (PC) +8 = 2002H+0008H

= 200AH

EA2= (PC) –9 =2002H+FFF7H

= 1FF9H

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，试回答以下问题。

（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）一次间接寻址和多次间接寻址的寻址范围。

（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）。

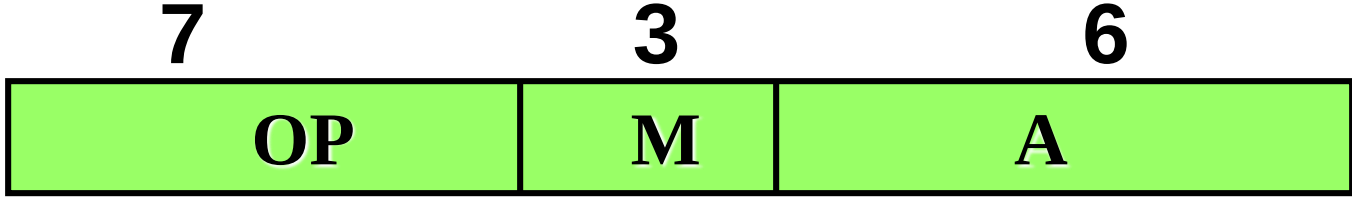
（6）上述六种寻址方式的指令中哪一种执行时间最短，哪一种最长，为什么？哪一种便于程序浮动，哪一种最适合处理数组问题？

（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：



各字段的作用：

OP——操作码字段，提供至少108种指令操作码；

M——寻址方式码字段，指出6种寻址方式；

A——形式地址字段，给出寻址所需的形式地址。

（2）A为6位，该指令直接寻址的最大范围为2^6=64字；

（3）一次间址的寻址范围为2^{16}=64K字；

多次间址的寻址范围为2^{15}=32K字；

（4）立即数的范围：若采用补码表示为1FH~20H；十进制表示为31~ -32；无符号数为0~63；

（5）相对寻址的位移量范围在采用补码表示时同立即数范围，为31~ -32；

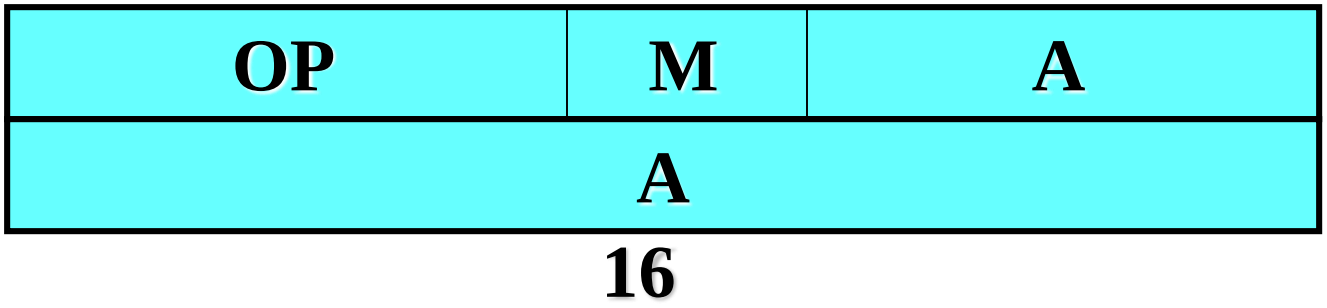
（6）六种寻址方式中，立即寻址指令执行时间最短，因为此时不需寻址；

间接寻址指令执行时间最长，因为寻址操作需访存一次到多次；

相对寻址便于程序浮动，因为此时操作数位置可随程序存储区的变动而改变，总是相对于程序一段距离；

变址寻址最适合处理数组问题，因为此时变址值可自动修改而不需要修改程序。

（7）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：



图中，指令的第一字保持原来格式不变，形式地址A扩展到第2个字。这样，直接寻址时，EA=A=16+6=22位，正好可访问4M地址空间。由于A的扩展，变址、基址、相对、立即数等寻址方式也扩展到22位。

（8）如使一条转移指令能转移到主存的任一位置，可采用上述双字长一地址指令，通过选用合适的寻址方式完成。（如选用直接寻址就可转移到主存任一位置，但选用相对寻址则只能在±2M范围内转移。）

除此之外，（7）、（8）两题也可通过段寻址方式达到扩大寻址空间的目的（此时不需修改指令格式）。总之，不论采取何种方式，最终得到的实际地址应是22位。

方案二：

（7）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（16位），再由硬件自动完成段寻址，最后得22位物理地址。

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：

（7）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。

通过基址寻址与段寻址获得实际地址的区别：

1）基址寻址的基地址一般比较长（≥存储器地址位数），位移量比较短（=形式地址位数），相加后得到的有效地址长度=基地址长度。此时主存不分段。

实际地址=有效地址=基地址+位移量

段寻址是基址寻址的一种变种，当基地址短于存储地址时，基址寻址就变成了段寻址，基地址就叫做段地址，此时主存分段。

实际地址=段地址×偏移量+段内位移量（有效地址）

2）基址寻址一般在机器字长≥存储地址长度的机器中，可直接通过寻址计算获得实际地址。

在机器字长≤存储地址长度的机器中，由于CPU内部数据通路的限制，编程指定的任何一种寻址计算得到的有效地址长度都等于机器字长，为获得更长的地址字，硬件自动通过段寻址计算出存储器实际地址。此时除ALU之外，硬件还要增设专用的地址加法器。

相关问题：

* 一般：机器字长=存储字长；
* CPU中所有寄存器（包括基址寄

* 通常：指令字长不一定等于机器字长。早期的小型机由于字长较短，指令常以机器字长为单位变化（几字长指令，如PDP-11机），目前以字节长为单位变化（几字节指令）的较多。习题中指令字长=机器字长的假设只是为简单起见；

* 当设指令字长=存储字长（=机器字长）时，如用立即寻址，由于立即数由形式地址直接给出，而形式地址的位数肯定不足一个字长，因此立即寻址非常适用于编程给出短常数的场合。

提示：寻址方式的正确选择与编程技巧有关。

17. 举例说明哪几种寻址方式在指令的执行阶段不访问存储器？哪几种寻址方式在指令的执行阶段只需访问一次存储器？完成什么样的指令，包括取指令在内共访问存储器4次？

解：举例如下：

1）一地址指令在执行阶段不访存的寻址方式有：寄存器寻址、立即寻址。

2）一地址指令在执行阶段只访存一次的寻址方式有：寄存器间接寻址、直接寻址、基址寻址、变址寻址、相对寻址、页面寻址。

3）包括取指在内共访存四次的指令有：二重间址的一地址指令；一重间址的二地址指令，当另一操作数采用直接、基址、变址、相对、页面、寄存器间接寻址时。

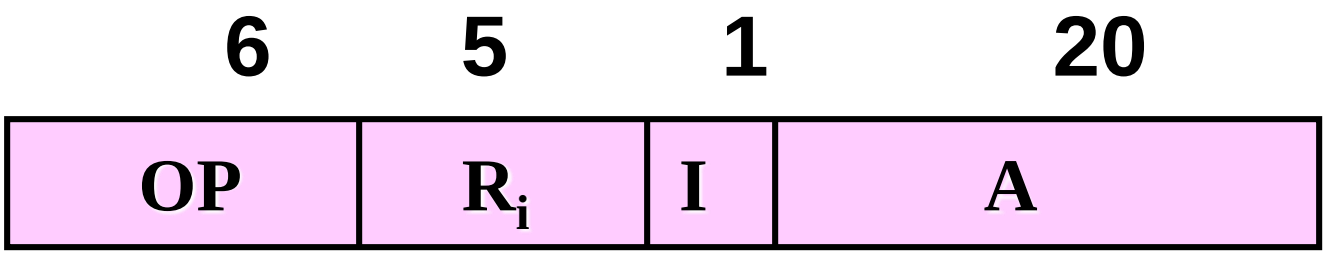
19. CPU内有32个32位的通用寄存器，设计一种能容纳64种操作的指令系统。假设指令字长等于机器字长，试回答以下问题。

（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：



直接寻址的最大空间=2^{20}=1M字

此指令格式的设计有较大的发挥余地，为简化设计，在此采用紧贴题意的答题方式，即只按题意要求的因素设计，不考虑扩展因素。

（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：

6	5	1	1	5	14
OP	R_i	I	B	BR_i	A

其中：B可省（B为基址寻址标志），BRi为基址寄存器号。基址寻址时：

寻址的最大空间=2^{32}=4G字

其寻址范围仅与基址位数有关，与形式地址位数无关。

CPU的结构和功能

第　八　章

2. 什么是指令周期？指令周期是否有一个固定值？为什么？

解：指令周期是指一条指令从开始取指令直到指令执行完这段时间。

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。

讨论：指令周期长度不一致的根本原因在于设计者，为了提高CPU运行效率而这样安排的，与指令功能不同及指令实际执行时间不同没有什么必然关系。

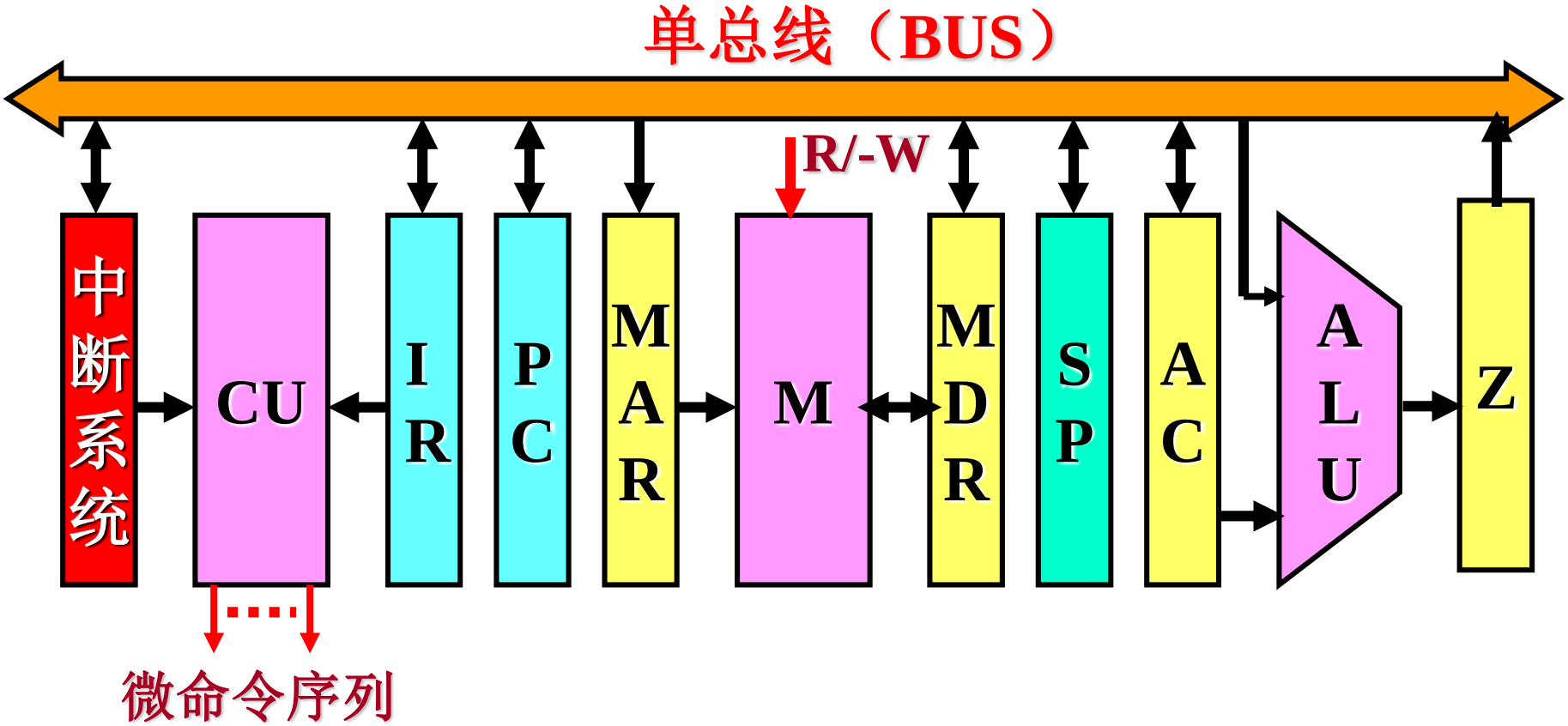
4. 设CPU内有下列部件：PC、IR、SP、AC、MAR、MDR和CU，要求：

（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。

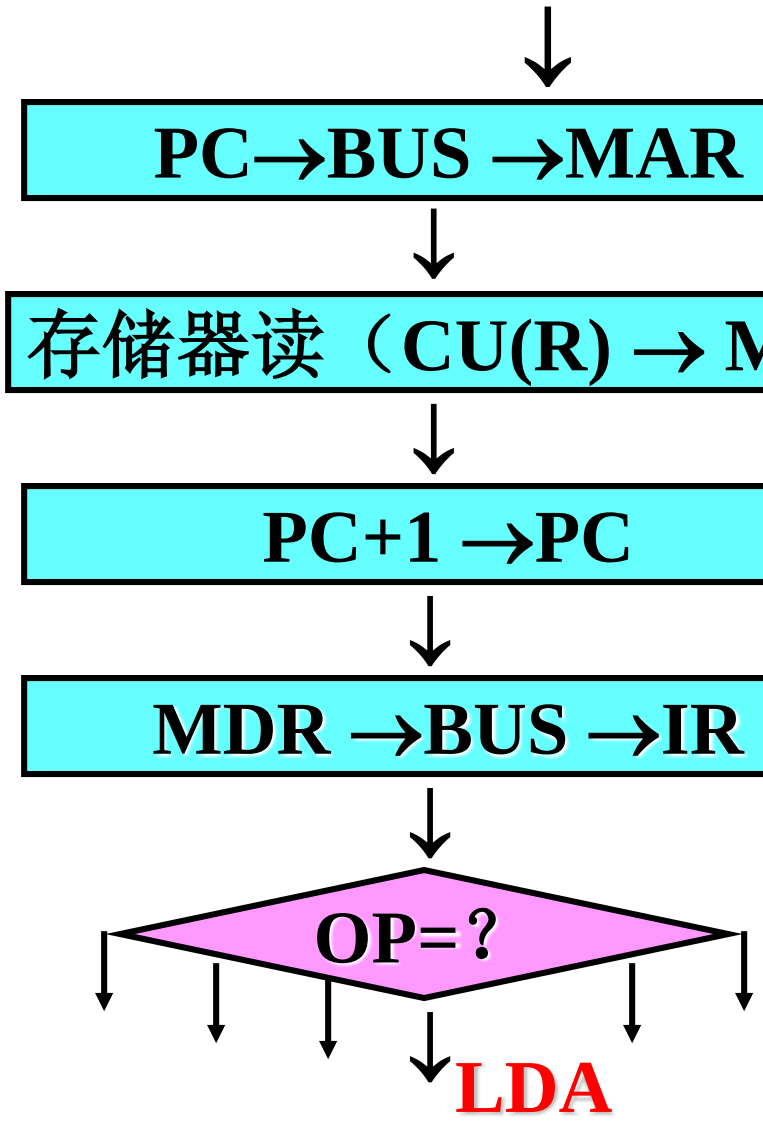
解：CPU中的数据流向与所采用的数据通路结构直接相关，不同的数据通路中的数据流是不一样的。常用的数据通路结构方式有直接连线、单总线、双总线、三总线等形式，目前大多采用总线结构，直接连线方式仅适用于结构特别简单的机器中。

为简单起见，本题采用单总线将题意所给部件连接起来，框图如下：



（1）假设为一重间址，在上述数据通路中，完成间接寻址的取数指令LDA@X的数据流如下页：

LDA@X指令周期流程图：



说　明

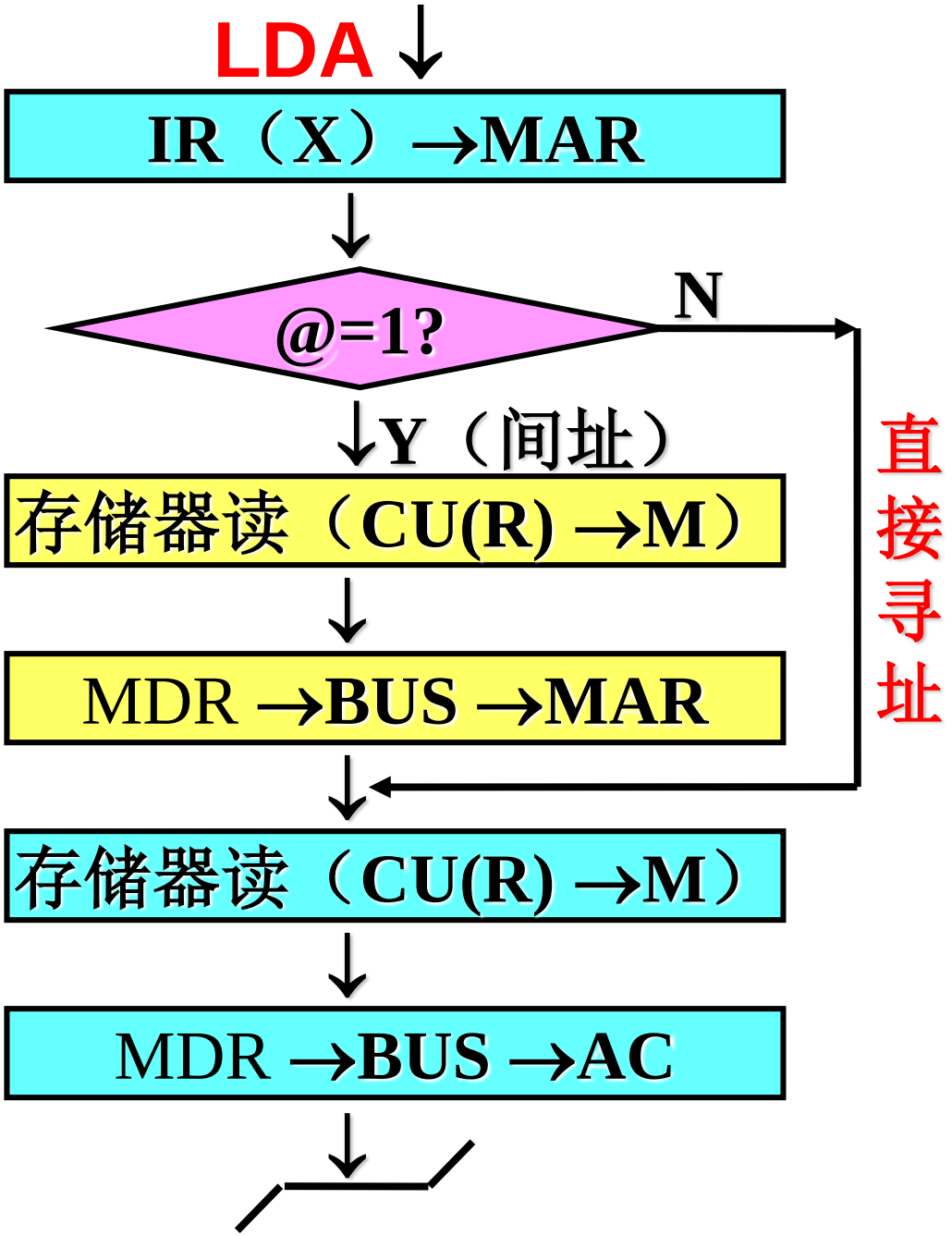
送指令地址

CU向存储器发读令（读出指令）

指向下一指令地址

取出指令

指令译码



说　明

形式地址X送MAR

间接标志判断，本题为@=1

CU发读令(读EA)

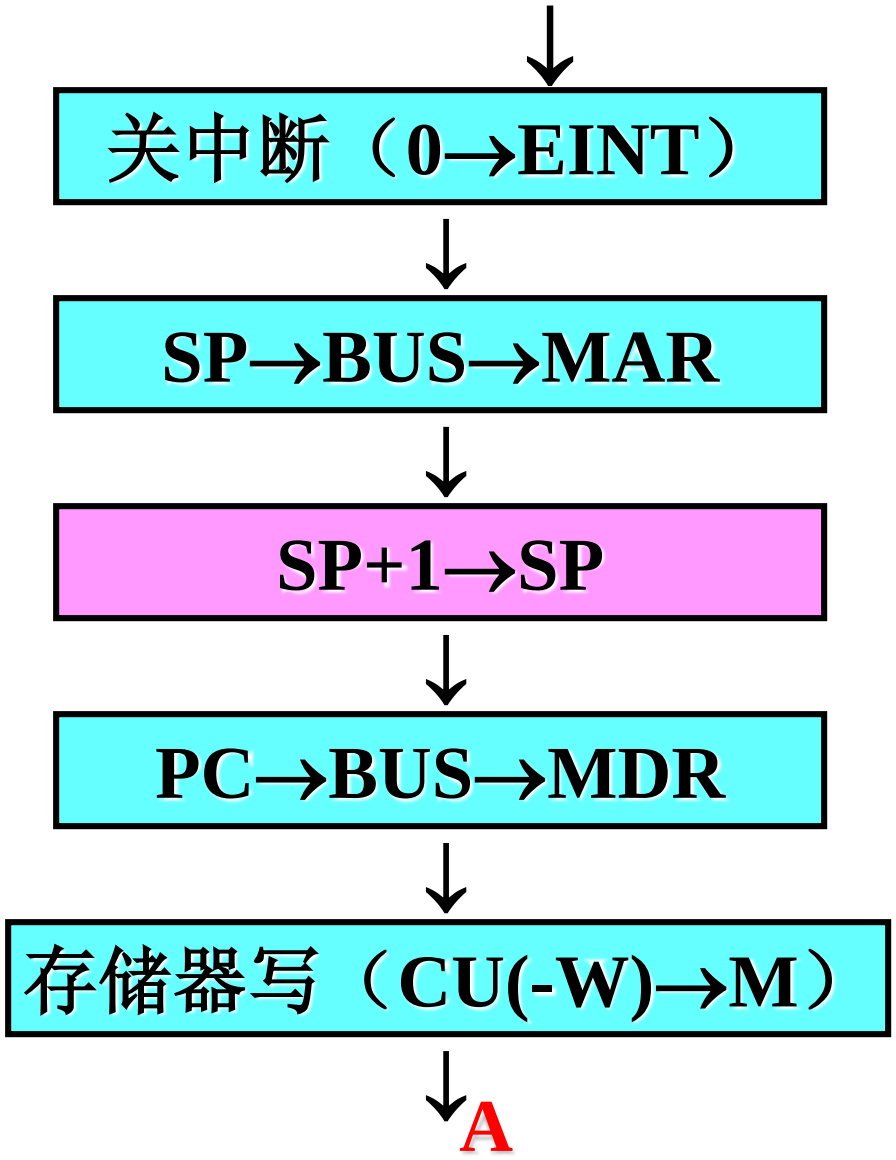
有效地址送MAR

CU发读令(读数据)

数据放入AC

指令末的公操作

（2）中断周期流程图如下：　　说　明

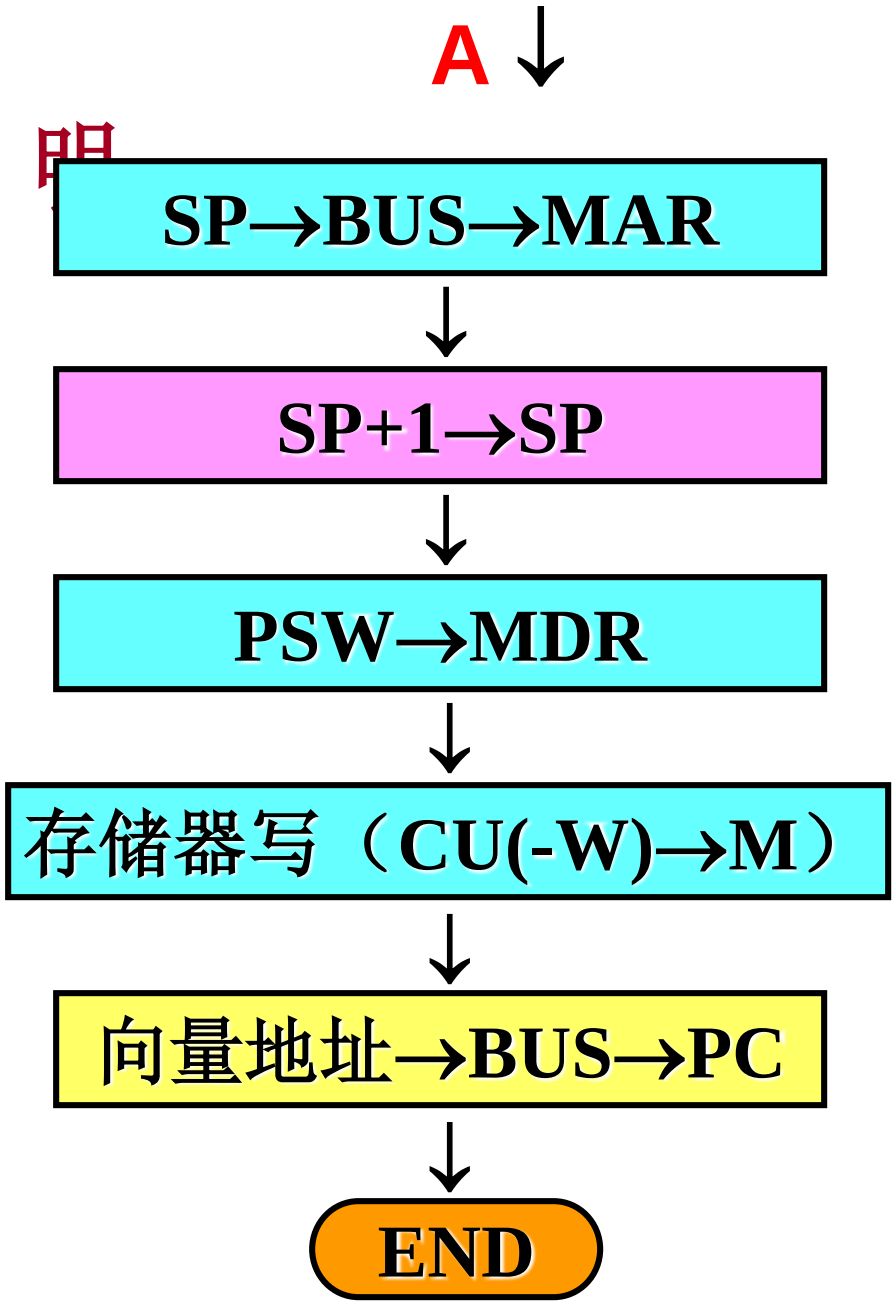


堆栈栈顶地址送MAR

修改堆栈指针

断点送内存

进栈


A
说
SP→BUS→MAR
栈顶地址送MAR
SP+1→SP
修改栈指针
PSW→MDR
程序状态字送内存
存储器写（CU(-W)→M）
进栈
向量地址→BUS→PC
转中断服务程序入口
END
中断周期结束

讨论：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，既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，既强调一个“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：从CPU机器周期的时序层次来看，中断周期前是指令的执行阶段。中断周期后是取指令阶段。在中断周期CPU应完成关中断、保存断点和转中断服务程序入口三个操作。

16. 计算机为了管理中断，在硬件上通常有哪些设置？各有何作用？对指令系统有何考虑？

解：计算机为了管理中断，在硬件上设有专门处理中断的机构——中断系统。它通常包括：中断请求寄存器、中断优先级排队器、向量编码器、中断允许触发器（EINT）、中断标记触发器（INT）、中断屏蔽触发器（寄存器）等。功能如下：

中断请求寄存器——对中断源发来的一过性中断请求信号进行登记；

中断优先级排队器——对同时提出的多个中断请求信号进行裁决，选出一个最紧迫的进行响应；

向量编码器——向量中断时，用来产生向量地址；

中断允许触发器（EINT）——CPU中的中断总开关，完成开、关中断状态的设置；

中断标记触发器（INT）——用来建立中断周期状态。INT=1，表示进入中断周期，即开始执行中断隐指令；

中断屏蔽触发器——对于可屏蔽的中断源进行开、关中断操作，可视为各中断源的中断分开关；

采用程序中断技术时，指令系统中往往有相关指令支持。常见的指令有：开中断、关中断、中断返回等。

17. 在中断系统中，INTR、INT、EINT这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。

讨论：

回答时首先应给出该触发器的**中文名称**，然后说明其**主要作用**。

× 当进入中断周期时，INT=1；

(INT=1时，进入中断周期)

× INT与EINT配合使用以实现关中断功能，即INT=1，反相后使EINT=0；

(关中断并不是INT的主要功能，进入中断周期后要执行中断隐指令的全部三个功能)

× INT表示自愿中断，完成系统调用；

(尽管INT触发器的英文缩写与INT指令助记符完全相同，但它们一个是**硬件设置**，一个是**软中断指令**，其作用完全不同)

× INT标记目前是否正在运行中断程序；

(INT标记在运行中断程序时已不存在)

× INT表示处于中断状态中；

(INT并不是在整个中断过程中都存在)

× INT判断中断过程中是否接受其它中断请求，INT=0时，开中断，允许中断嵌套；

(INT标记与中断嵌套技术没有任何关系。它不能表示出中断过程中是否接受其它中断请求，INT=0也不表示开中断)

× EINT判断CPU是否响应中断请求；

(CPU根据EINT状态决定是否响应中断请求)

× 当CPU响应中断时，EINT置1；

(当EINT=1时，允许CPU响应中断)

× EINT确保CPU响应中断后，不受新的中断干扰；

(CPU响应中断在先，进入中断周期后才使EINT=0，仅在单重中断时，整个中断过程保持EINT=0，不接受新的中断请求)

× EINT表示中断隐指令，INT起关中断作用；

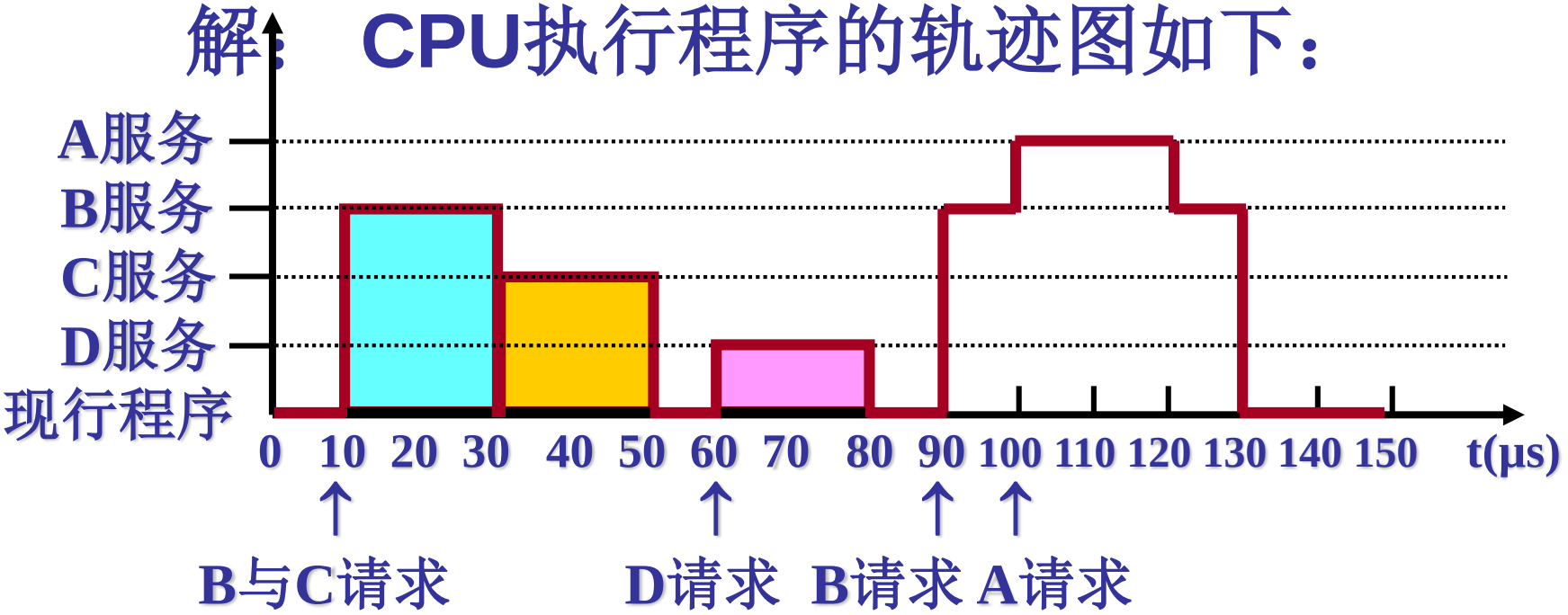
(把EINT和INT的作用搞反了)

× INTR=1，判断哪个中断源有请求；

(INTR对中断源的请求进行登记，当INTR=1时，表示有请求)

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：CPU执行程序的轨迹图如下：



这是一个多重中断的程序运行轨迹，图中忽略了中断响应时间。

25. 设某机有五个中断源L_0、L_1、L_2、L_3、L_4，按中断响应的优先次序由高向低排序为$L_0 \rightarrow L_1 \rightarrow L_2 \rightarrow L_3 \rightarrow L_4$，现要求中断处理次序改为$L_1 \rightarrow L_4 \rightarrow L_2 \rightarrow L_0 \rightarrow L_3$，根据下面的格式，写出各中断源的屏蔽字。

中断源	屏蔽字 0	1	2	3	4
L_0	1	0	0	1	0
L_1	1	1	1	1	1
L_2	1	0	1	1	0
L_3	0	0	0	1	0
L_4	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。

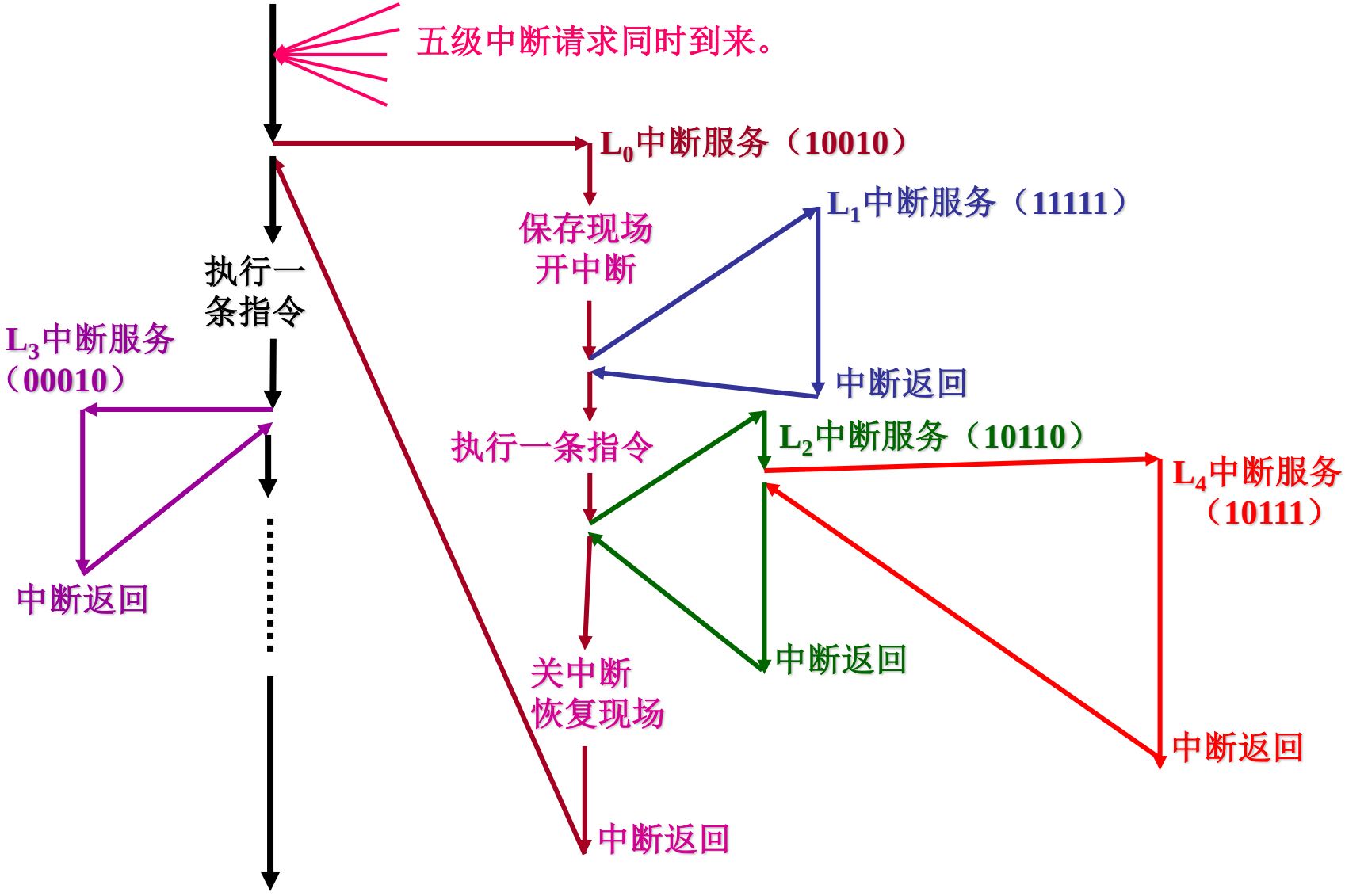
为了使所有中断都能得到及时响应，现行程序的中断屏蔽字一般设为全开放（全0）状态。

讨论：按照修改过的优先次序，当五个中断请求信号同时到来时，CPU中断处理过程如下图：

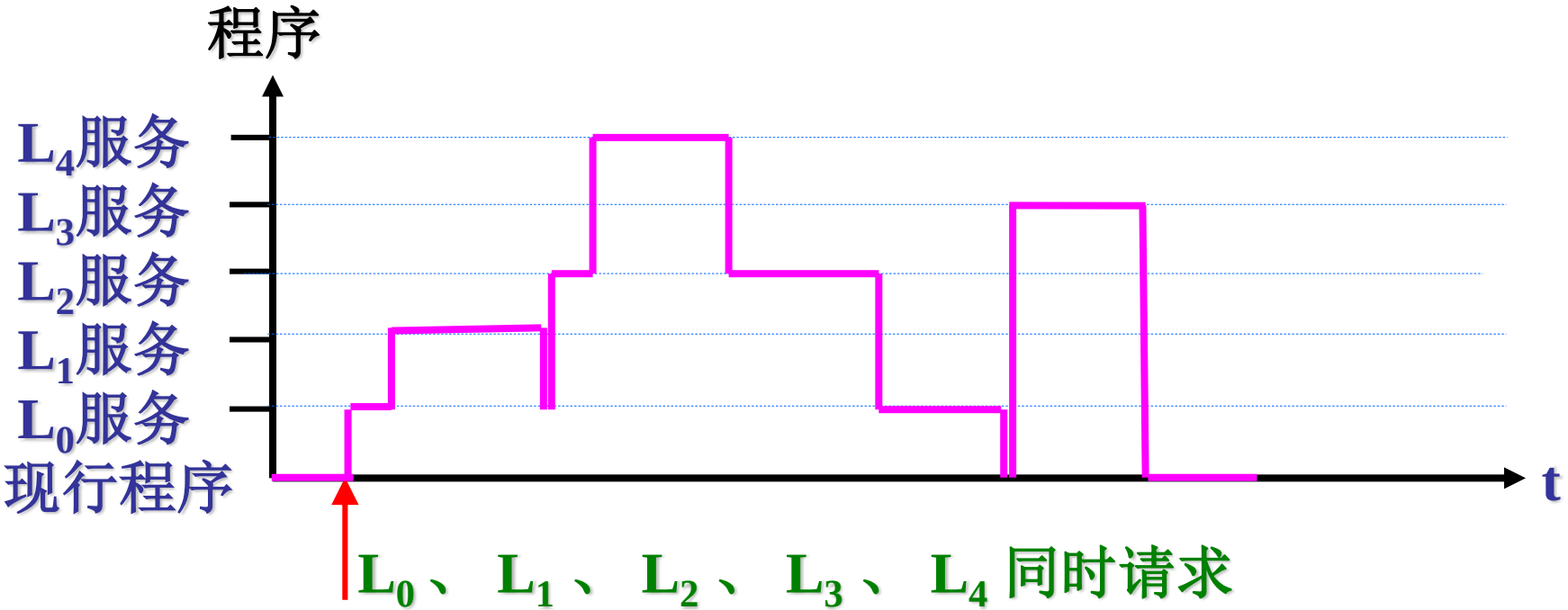
图中括号内为各程序的屏蔽码。

注意：中断屏蔽码的判优作用体现在对低级中断请求的屏蔽上，对于多个同时到来的高级中断请求信号之间则只有开放作用，没有判优作用。此时还需依赖硬件排队线路完成进一步的判优。

现行程序（00000）


五级中断请求同时到来。
L_0中断服务（10010）
保存现场
开中断
L_1中断服务（11111）
中断返回
执行一
条指令
L_3中断服务
（00010）
中断返回
执行一条指令
L_2中断服务（10110）
L_4中断服务
（10111）
中断返回
关中断
恢复现场
中断返回
中断返回

中断处理过程示意图（画法二：时空图表示）

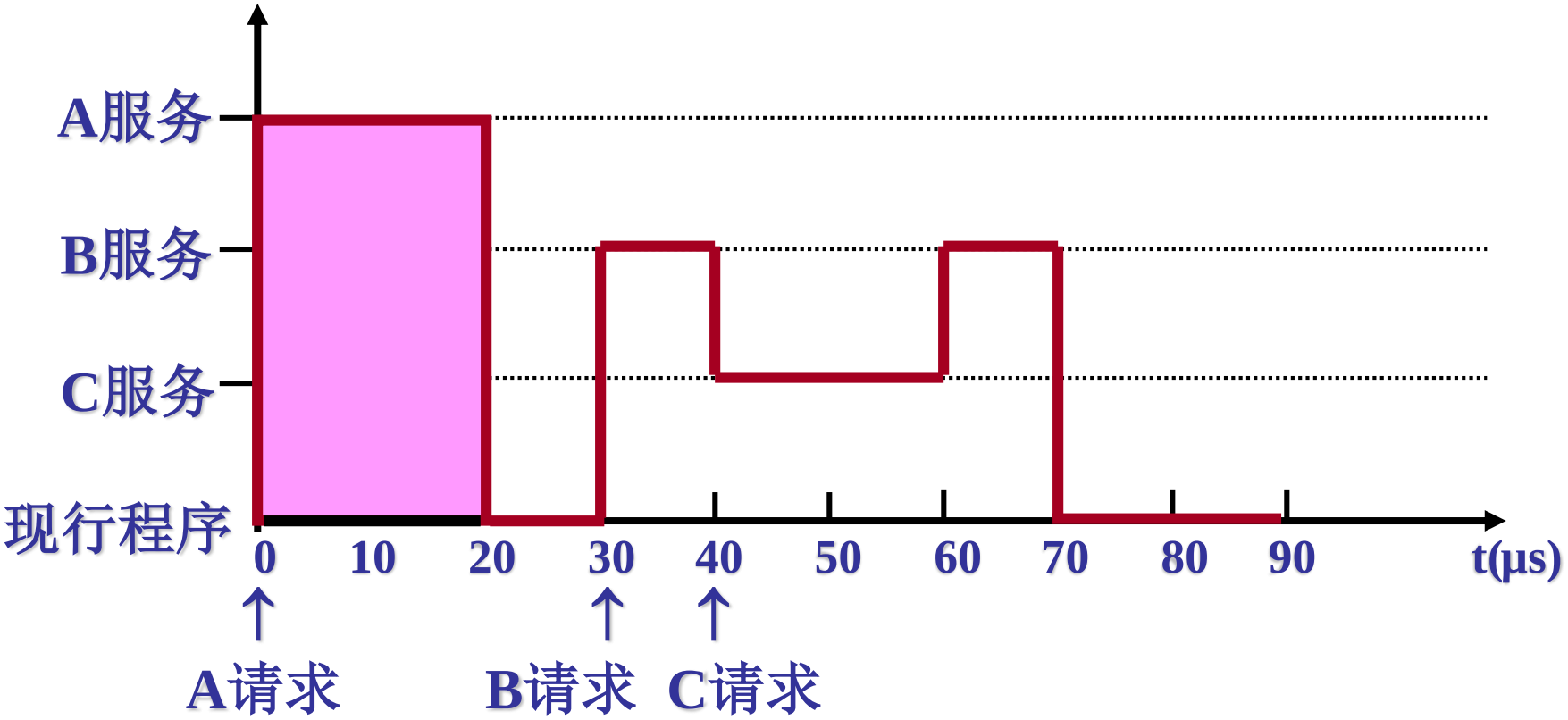


26. 设某机配有A、B、C三台设备，其优先级按A→B→C降序排列，为改变中断处理次序，它们的中断屏蔽字设置如下：

设备	屏蔽字
A	1 1 1
B	0 1 0
C	0 1 1

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20μs。

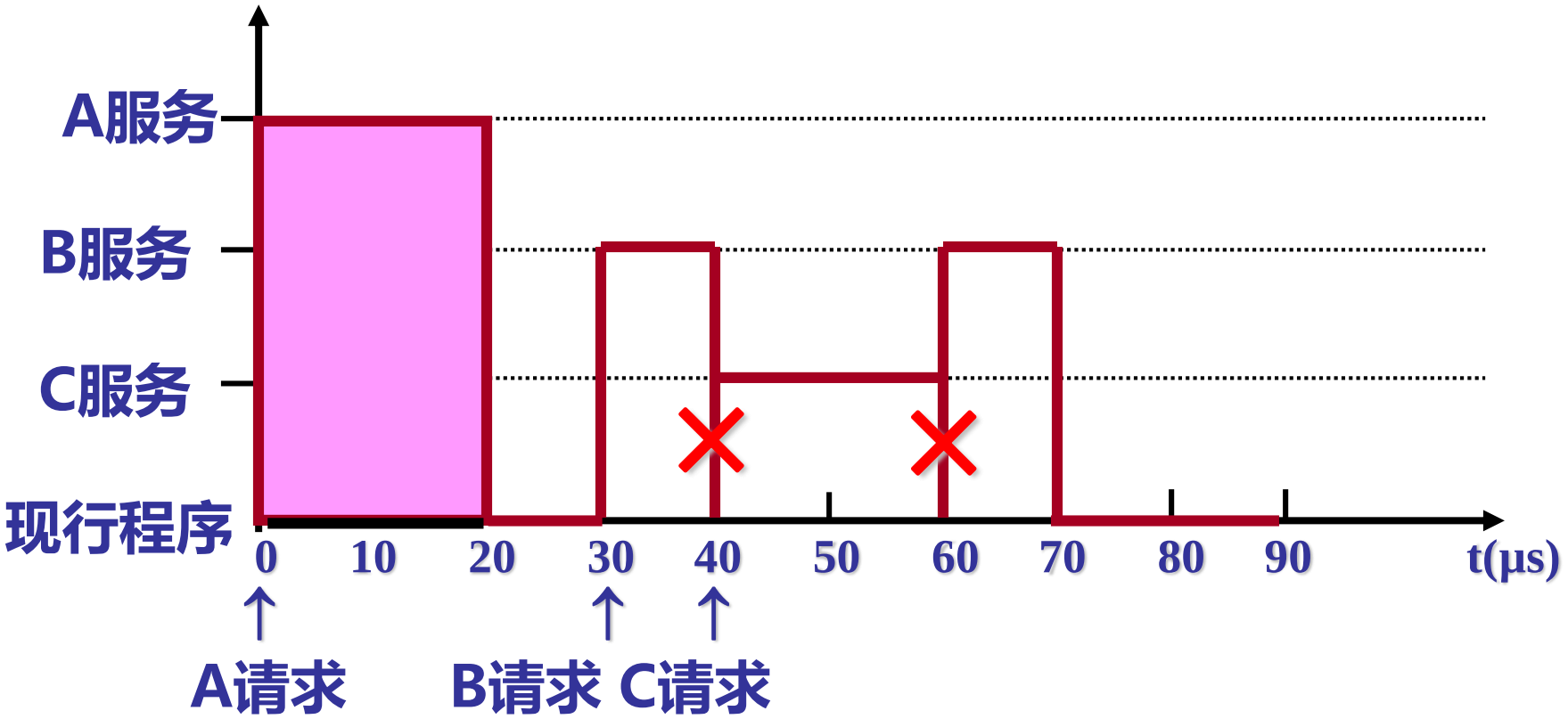
解： CPU执行程序的轨迹图如下：



这是一个多重中断的程序运行轨迹，图中忽略了中断响应时间。

主要注意问题：1）轨迹的连续性；2）程序的转出、返回轨迹及时刻；　3）现行程序在坐标系中的位置。

讨论：当从B中断转到C中断时，不返回现行程序，下述程序运行轨迹是错误的：

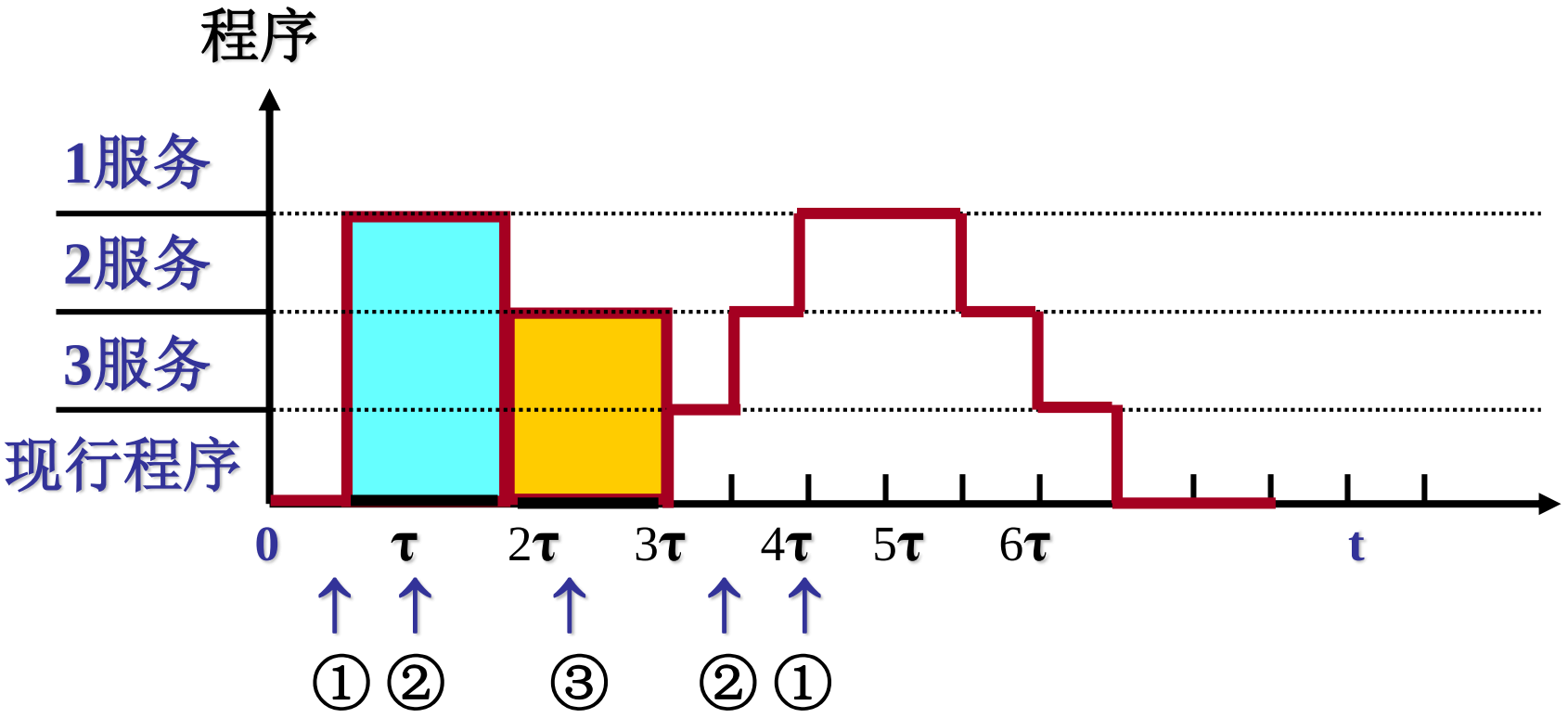


注意现行程序的运行轨迹在横坐标上，即此程序运行轨迹是相对于现行程序而言的。

27. 设某机有3个中断源，其优先级按1→2→3降序排列。假设中断处理时间均为T，在下图所示的时间内共发生5次中断请求，图中①表示1级中断源发出中断请求信号，其余类推，画出CPU执行程序的轨迹。

解： CPU执行程序的轨迹图见下页：

CPU执行程序的轨迹图



28. 设某机有4个中断源1、2、3、4，其响应优先级按1→2→3→4降序排列，现要求将中断处理次序改为4→1→3→2。根据下图给出的4个中断源的请求时刻，画出CPU执行程序的轨迹。设每个中断源的中断服务程序时间均为20μs。

解： CPU执行程序的轨迹图见下页：

CPU执行程序的轨迹图：