存 储 系 统

- 随机读写存储器
- 只读存储器和闪速存储器
- 高速存储器
- cache存储器
- 虚拟存储器
- 存储保护

概述

存储器是计算机系统中的记忆设备，用来存放程序和数据。

存储元是存储器中最小的存储单位，可以存储一位二进制代码。由若干个存储元组成一个存储单元，再由许多存储单元组成一个存储器。

存储器有不同的分类方法。

★ 按存储介质分

半导体存储器：主要有MOS型存储器和双极型存储器两大类。

磁表面存储器：在金属或塑料基体上，涂覆一层磁性材料，用磁层存储信息，常见的有磁盘、磁带等。

光存储器：采用激光技术访问的存储器。

★ 按存取方式分

随机存储器：任何存储单元的内容都能被随机存取，且存取时间和存储单元的物理位置无关。如半导体存储器。

顺序存储器：只能按某种顺序来存取，存取时间和存储单元的物理位置有关。如磁带存储器。

磁盘存储器既不像随机存储器那样能随机地访问任一个存储单元，也不像顺序存储器那样完全按顺序存取，而是介于两者之间。存取信息时，第一步指向整个存储器中的某个小区域（磁盘上的磁道）；第二步在小区域内顺序检索，直至找到目的地后再进行读/写操作。其存取时间和信息的物理位置有一定关系。

★ 按存储器的读写功能分

只读存储器(ROM)：存储的内容固定不变，只能读出而不能写入。

随机读写存储器(RAM)：既能读出又能写入。

★ 按信息的可保存性分

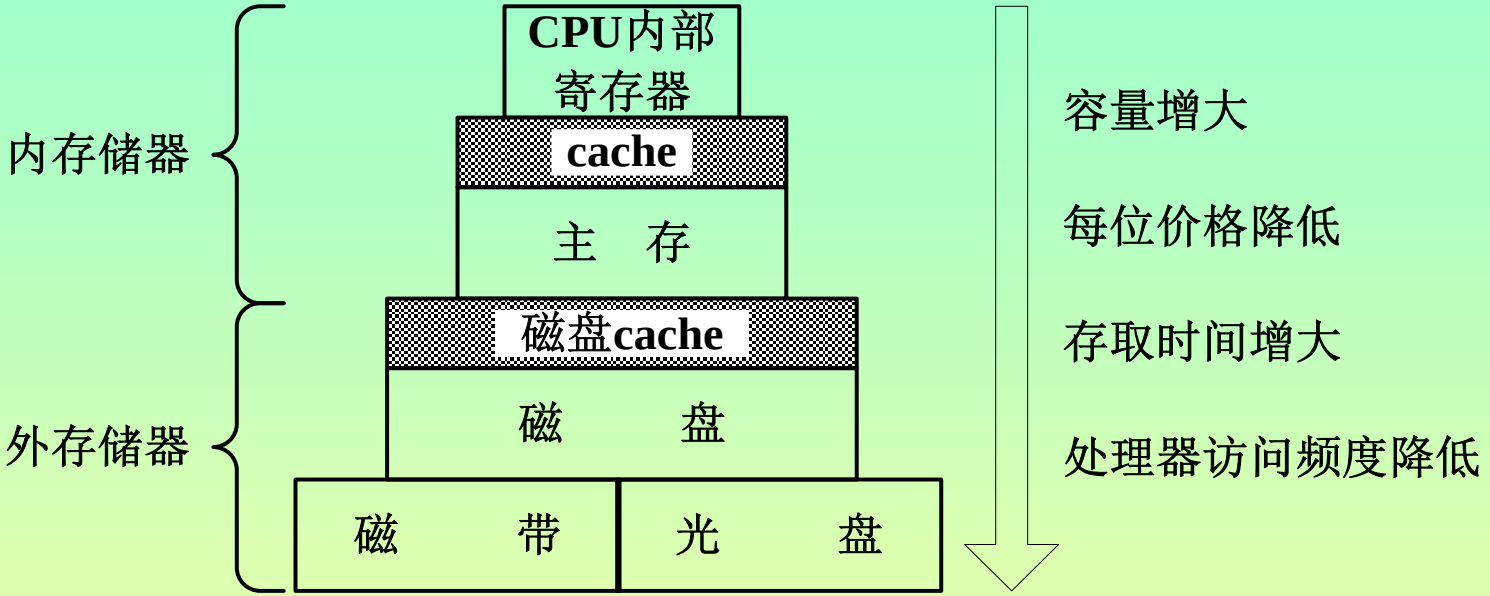
易失性存储器：断电后信息即消失的存储器。

非易失性存储器：断电后仍能保存信息的存储器。

★ 按在计算机系统中的作用分

根据存储器在计算机系统中所起的作用，可分为主存储器、辅助存储器、高速缓冲存储器、控制存储器（用于存放微程序，由ROM构成）。

存储系统的层次结构



为了解决存储容量、存取速度和价格之间的矛盾，计算机中通常采用多级存储器体系结构，即使用高速缓冲存储器、主存储器和外存储器。CPU能直接访问的存储器称为内存储器，包括高速缓冲存储器和主存储器。CPU不能直接访问外存储器，外存储器的信息必须调入内存储器才能被CPU处理。

高速缓冲存储器－主存层次：CPU的处理速度比主存的存取速度快。为弥补主存速度的不足，在主存和CPU之间增加一级高速缓冲存储器 (Cache)。其特点是速度高而容量小。它所存放的是主存中部分内容的复制，是当前最有可能被CPU访问的信息。

从整体看，Cache－主存层次的存取速度接近于Cache的速度，而容量接近于主存的容量。Cache存储器全部由硬件调度，对程序员是透明的。

主存－辅存层次：主存的存储量仍不能满足程序运行的要求，因此利用大容量、低价格的外部存储器作为辅助存储器。当前要用到或经常用到的信息存储在主存，未用到或不常用到的信息存储在辅存，需要时调往主存。主存和辅存一起构成了现在广泛使用的“虚拟存储系统”。

从整体看，主存－辅存层次具有接近于主存的速度和接近于辅存的容量。虚拟存储系统需要由操作系统来调度，因此对系统程序员是不透明的，但对应用程序员是透明的。

存储器的性能指标

存储容量：存储器所包含的存储单元的总数称为存储容量。存储容量用字数或字节数表示。一个字节定义为8个二进制位，一个字包括2个或4个字节。

存取时间：从启动一次存储器操作到完成该操作所经历的时间。例如：读出时间是指从CPU向主存发出有效地址和读命令开始，直到将被选单元的内容读出为止所用的时间；写入时间是指从CPU向主存发出有效地址和写命令开始，直到信息写入被选中单元为止所用的时间。

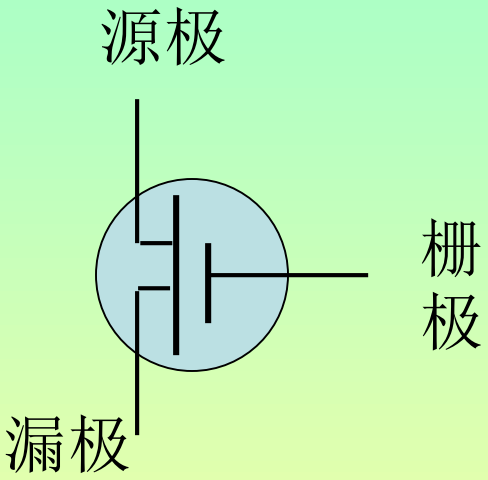
存取周期：连续两次访问存储器操作之间所需要的最短时间。一般情况下，存取周期大于存取时间。这是因为对于任何一种存储器，在读写操作之后，总要有一段恢复内部状态的复原时间。

存储器带宽：又称数据传输率，指单位时间内存储器可读写的数据量，用位/秒或字节/秒度量。由存取周期和字长决定。

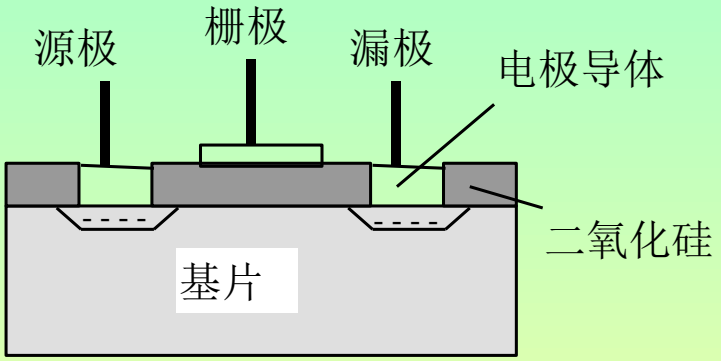
可靠性：用平均无故障时间MTBF来衡量。

其它参数：功耗、价格等。

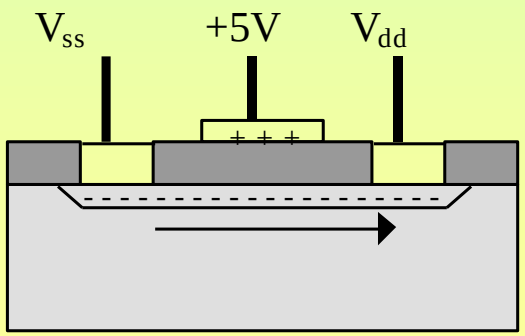
场效应管



场效应管



(a) MOS 晶体管结构

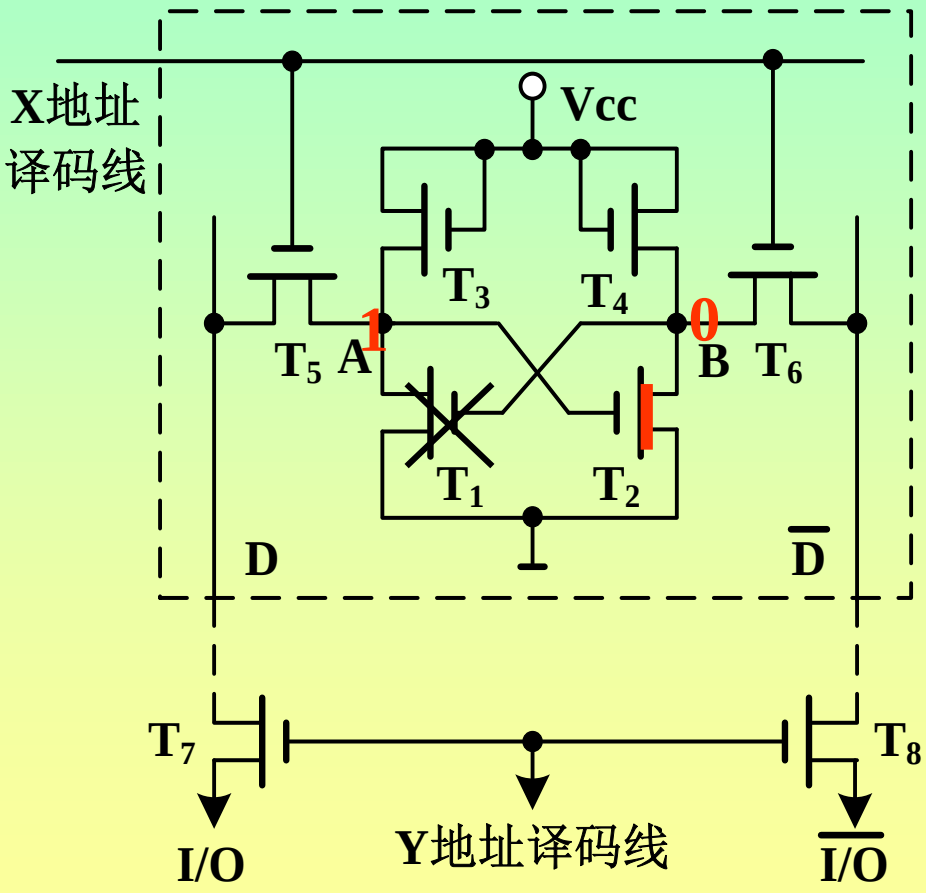


(b) MOS 晶体管导通状态

静态随机读写存储器SRAM

目前广泛使用的半导体存储器是MOS型半导体存储器，可以分为静态MOS型存储器（Static RAM）和动态MOS型存储器（Dynamic RAM）。

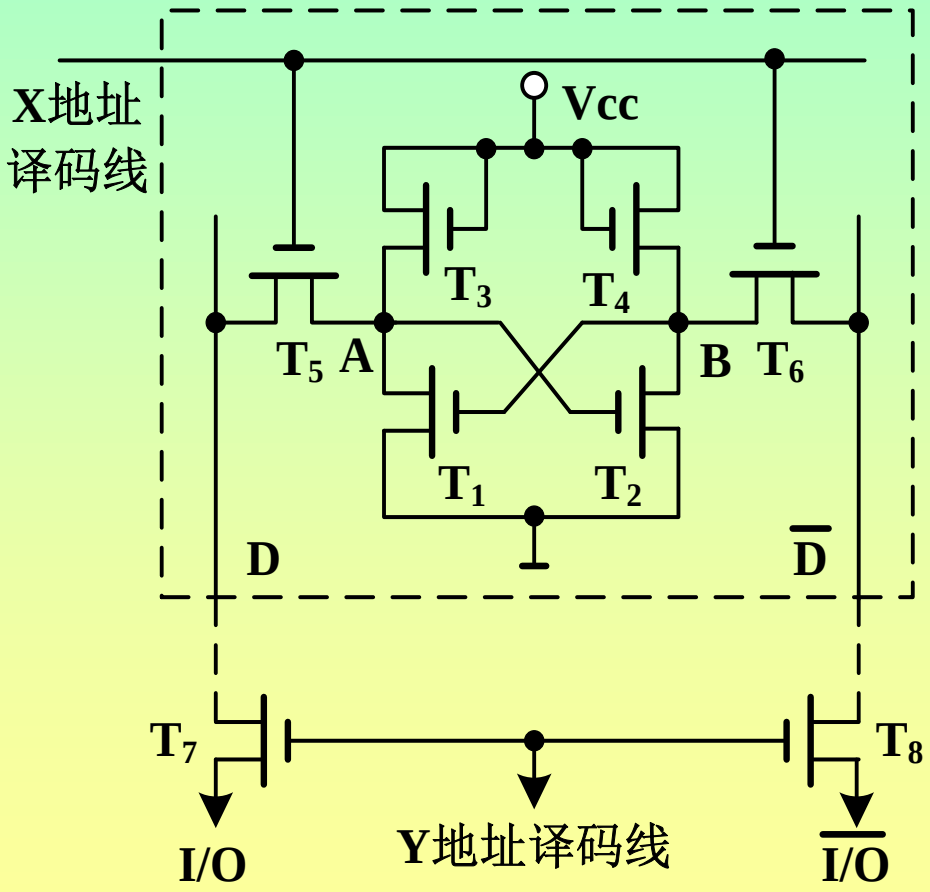
右图是六管SRAM存储元的电路图。T_3、T_4相当于负载电阻，T_1和T_2构成双稳态触发器。若T_1截止，A为高电平，使T_2导通，B为低电平，而B的低电平又使T_1更加截止；反之，若B为高电平，则A为低电平。可见该电路有两个稳定状态，且A和B两点电位总是互反的。如果用A点高电平代表“1”，A点低电平代表“0”，该电路可存储一位二进制数。



T_5、T_6、T_7和T_8为控制管。如果某存储元被选中，X、Y地址译码线均处于高电平，使T_5～T_8导通，输入输出电路I/O和$\overline{\text{I/O}}$分别与A点和B点相连，A点和B点的电平状态就能输出到I/O和$\overline{\text{I/O}}$上，完成读操作。

写操作时，如果要写入“1”，在I/O线上输入高电位，在$\overline{\text{I/O}}$线上输入低电位，开启T_5～T_8四个MOS管把高、低电位分别加在A、B点，使T_1管截止，使T_2管导通，将“1”写入存储元。

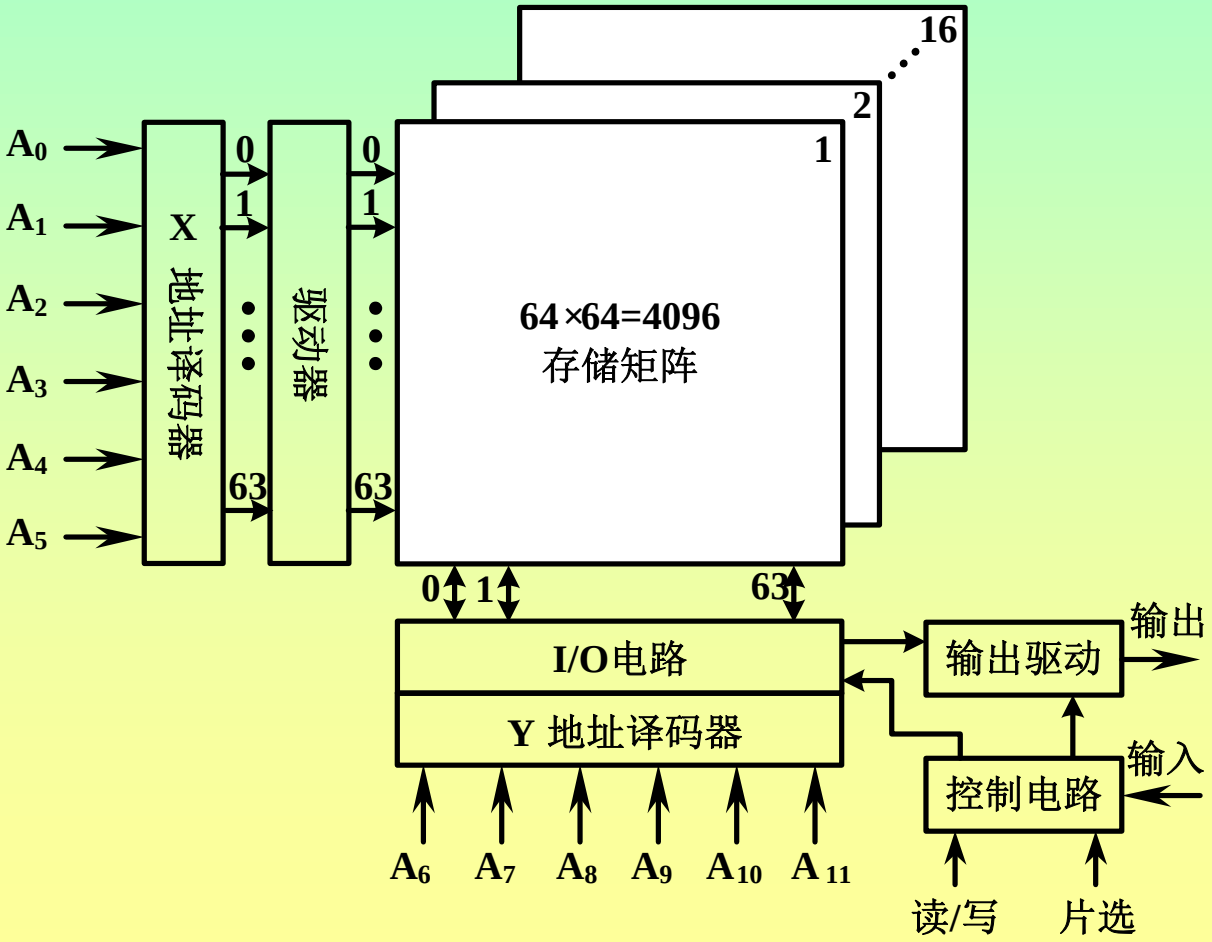
如果要写入“0”，在I/O线上输入低电位，在$\overline{\text{I/O}}$线上输入高电位，打开T_5～T_8，把低、高电位分别加在A、B点，使T_1管导通，T_2管截止，将“0”写入存储元。



SRAM存储器的组成

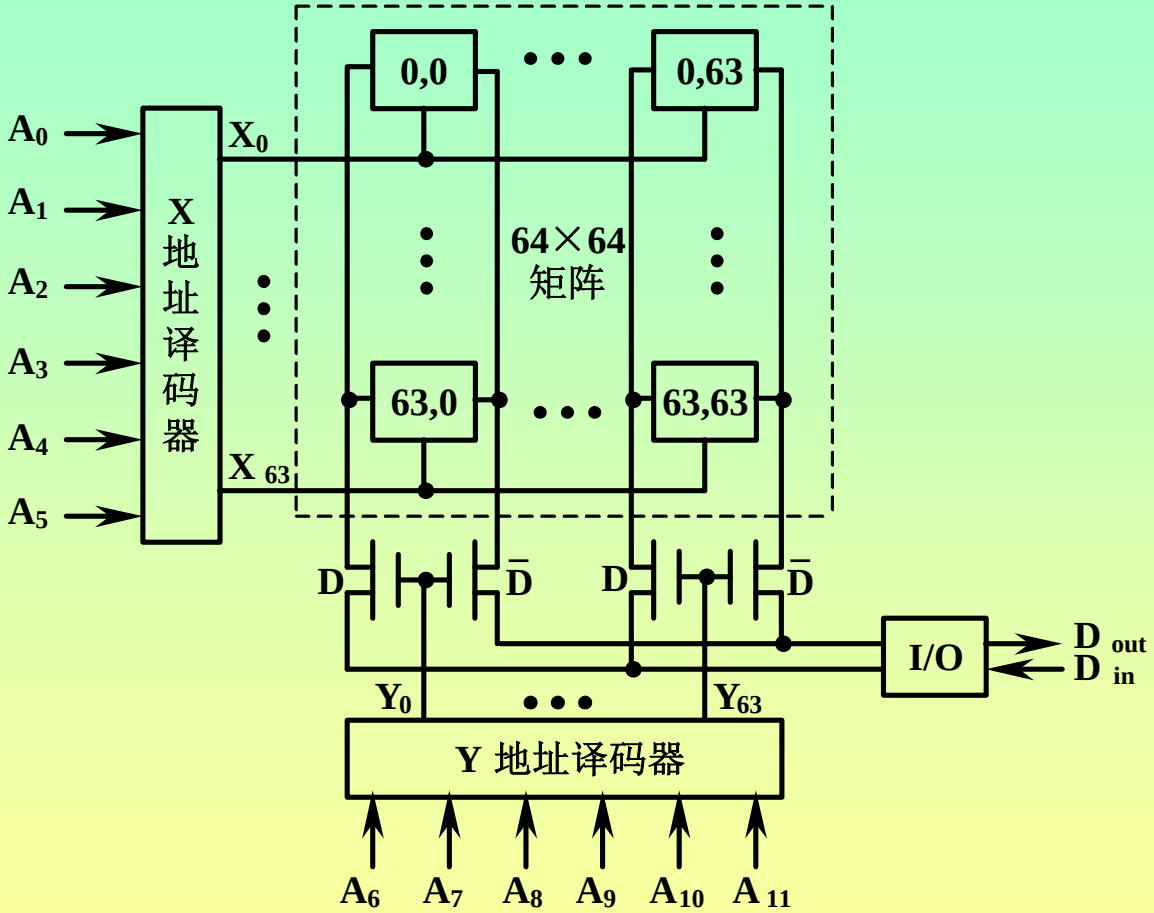
SRAM存储器由存储体、地址译码电路、读写电路和控制电路组成。

存储体：存储体是存储单元的集合。较大容量的存储器中，往往把各个字的同一位集成在一块芯片内，并排列成矩阵形式，由行选择线和列选择线进行选择。例如把4096个字的同一位集成在一块4096×1位的芯片中，用16块这样的芯片就可以组成4096×16位的存储器。



地址译码器：CPU要选择某一存储单元，就在地址总线上输出该单元的地址信号给地址译码器，译码器把二进制代码表示的地址转换成电平信号，选中要访问的存储单元。

右图是双译码结构的地址译码器，包括X向和Y向两个译码器。若每个译码器有n/2个输入端，$2^{n/2}$个输出端，两个译码器交叉译码的结果，可以选择$2^{n/2} \times 2^{n/2} = 2^n$个存储单元。这种结构可以减少译码线数目，适用于大容量存储器。

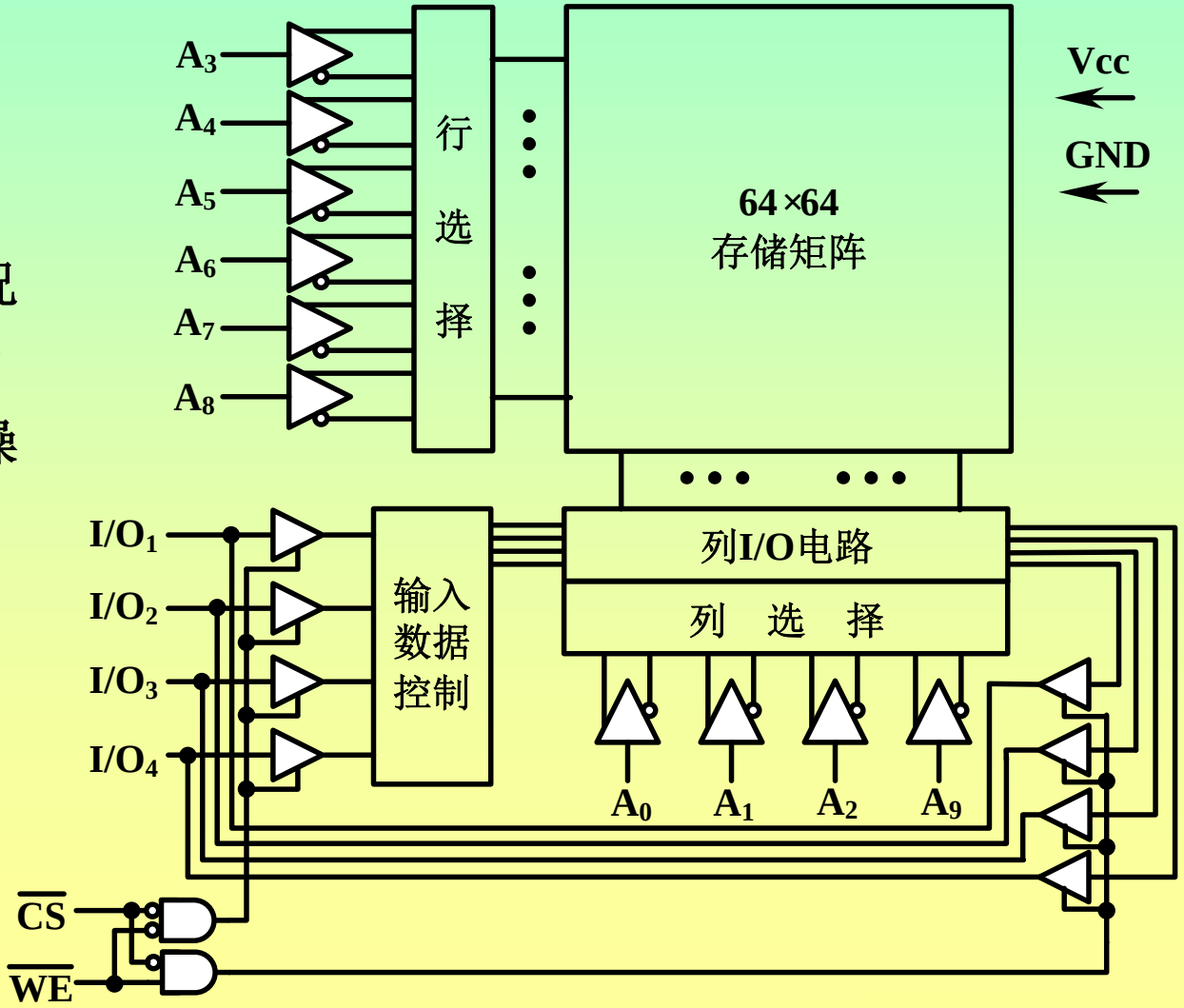


片选与读写控制电路：读写控制线用来控制芯片是进行读操作还是写操作，片选线用来决定该芯片是否被选中。

SRAM芯片2114

2114是1K×4位的SRAM，有10根地址线。A3～A8用于行译码，产生64根行选择线；A0、A1、A2、A9用于列译码，产生16根列选择线。每条列选择线控制4位。

在片选信号$\overline{CS}$有效（低电平）情况下，写命令$\overline{WE}$有效（低电平）进行写操作；$\overline{WE}$无效（高电平）进行读操作。



存储器与CPU的连接

RAM芯片通过地址线、数据线和控制线与外部连接。存储器同CPU的连接就是要完成地址线、数据线和控制线的连接。地址线是单向输入的，数据线是双向的，既可输入也可输出。

单个芯片的存储容量往往不能满足要求，需要进行扩展。扩展的方法有：位扩展法、字扩展法、字位同时扩展法。

1、存储芯片存储容量为8K×1，16K×8其含义分别表示什么？

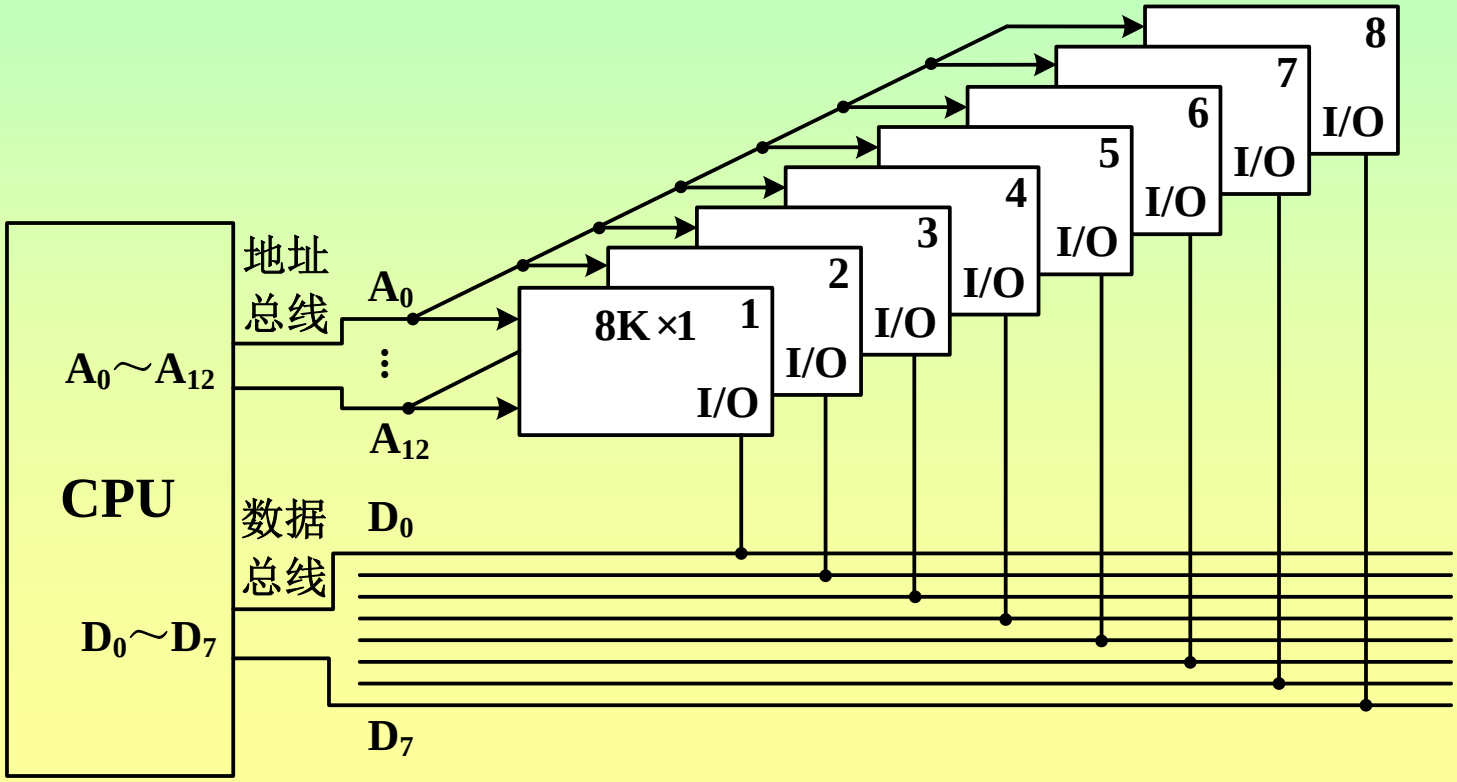
2、存储容量与地址线位数有关还是与数据线位数有关？

3、假设某存储器的地址线为A0~A12，数据线为D0~D7,从地址单元001(H)中读出的数据为03H，则地址线，数据线上的信号分别为多少？

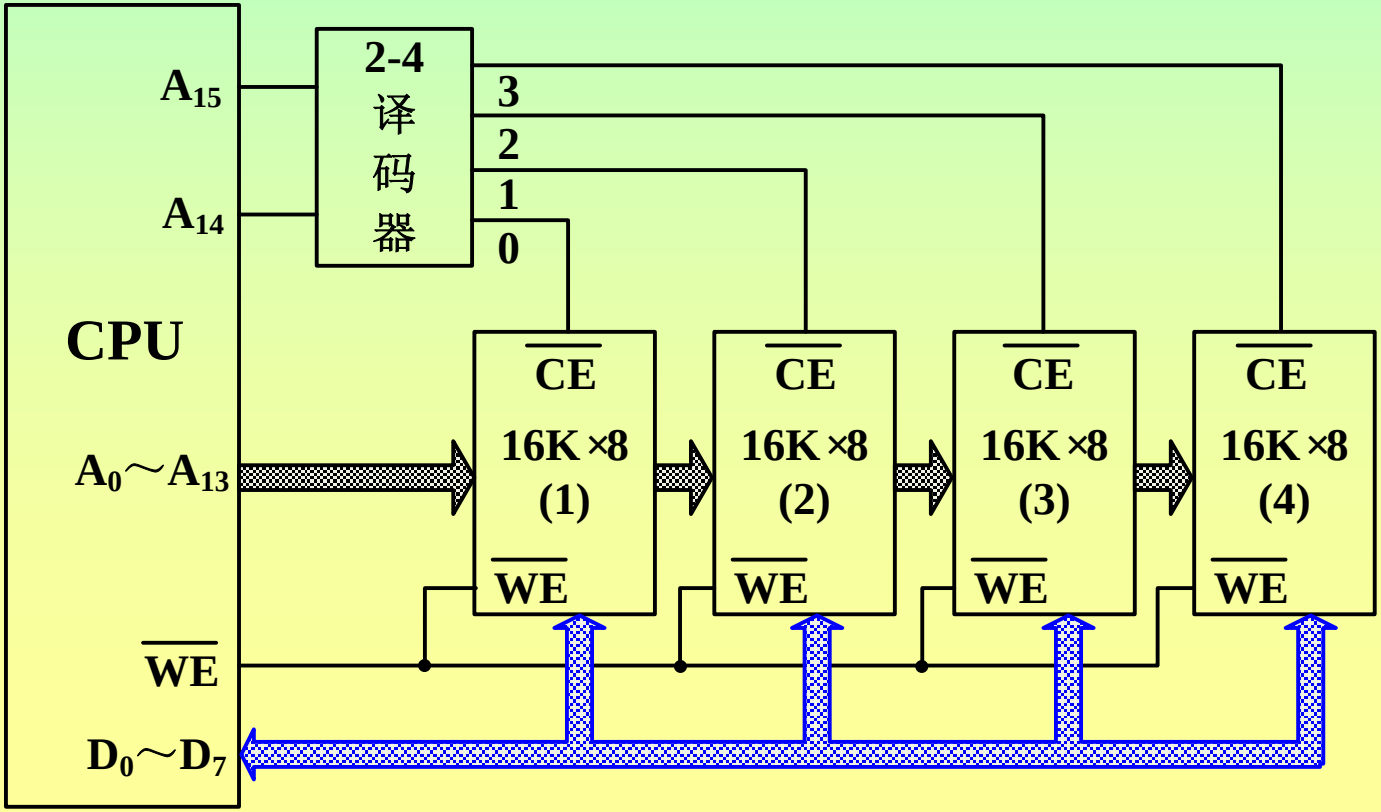
4、能否从一片8K×1的存储芯片中读出数据02（H）？

5、常见的数字换算关系，1K ＝ 2^{10} ＝ 400 （H）

位扩展法：如果存储器芯片的字数满足要求，而位数不够，需进行位扩展。例如：将8K×1的芯片组成8K×8的存储器。方法是将芯片的地址线、控制线并联，数据线分联。对片选信号$\overline{CS}$没有要求，直接接地。每一条地址总线有8个负载，每一条数据总线接一个负载。当给出某个地址时，同时选中8个芯片，将8个芯片上的数据从数据总线上读入或写入。



字扩展法：如果存储器芯片的位数满足要求，而字数不够，需进行字扩展。方法是将芯片的低位地址线、数据线、读写控制线并联，利用高位地址线经译码后作为片选信号。下图利用4片16K×8芯片经字扩展组成64K×8存储器。4个芯片的数据线与数据总线D_0～D_7相连，地址线与地址总线低位地址A_0～A_{13}相连，写允许信号$\overline{WE}$与CPU的$\overline{WE}$相连。高位地址A_{14}和A_{15}经译码器和4个片选端相连。



思考

1、要将多少位的低位地址线进行并联？
多少位的高位地址进行译码？

2、地址译码器要如何选择？选择3：8译码，2：4译码还是其他？
若需要用16K的芯片组成80K的存储器呢？

3、译码器的输出是高电平还是低电平？

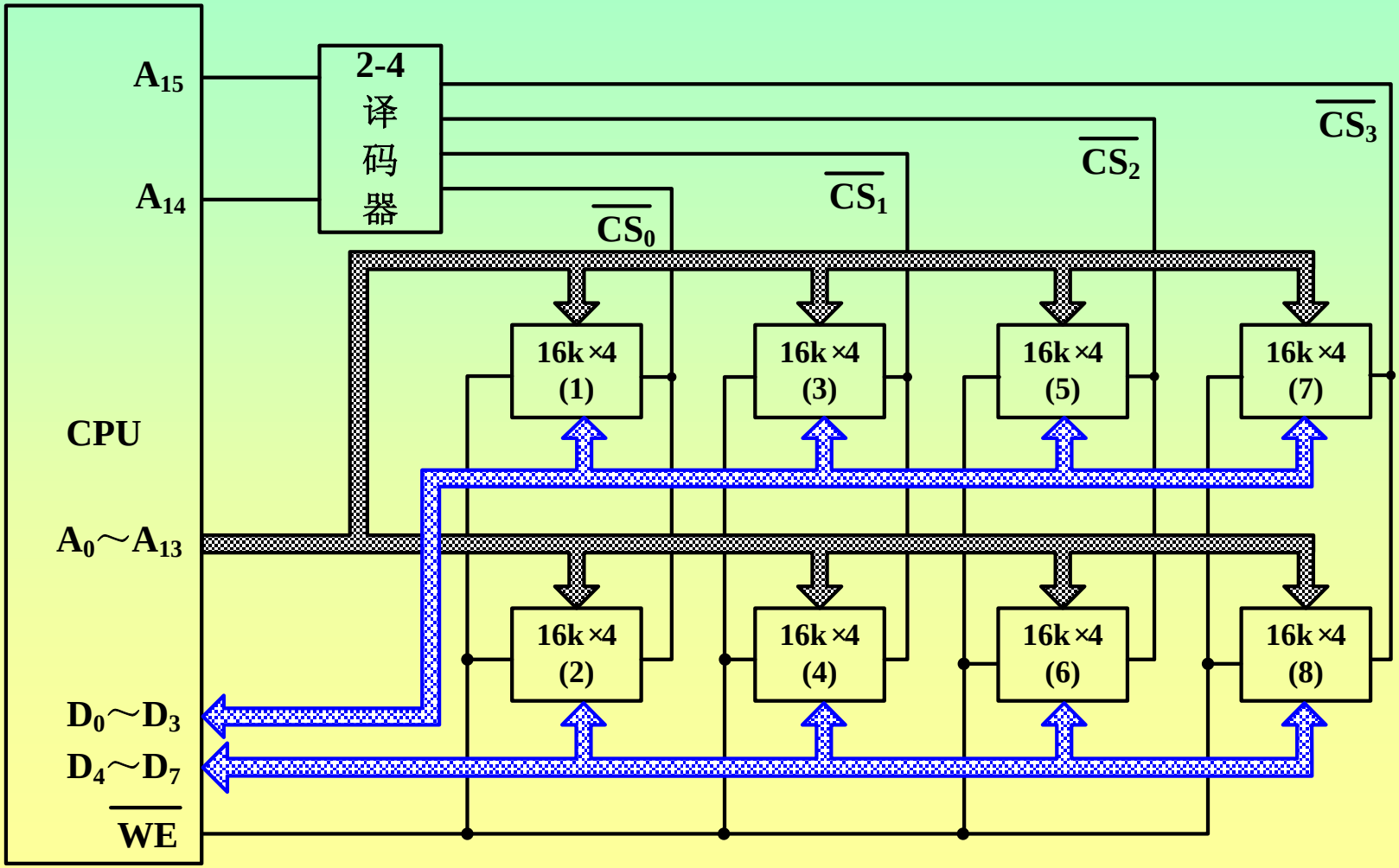
4、写出各个芯片的起始地址

5、当地址为4001H时，是如何选中需要的存储单元？
A0～A13，A14，A15分别为多少？译码结果如何？

同一时刻4个芯片中只能有一个芯片被选中。$A_{15}A_{14}$=00选中第一片，$A_{15}A_{14}$=01选中第二片，……。4个芯片的地址分配如下：

		$A_{15}A_{14}$	A_{13}～A_0	
第一片	最低地址	00	00 0000 0000 0000	＝ 0000 H
	最高地址	00	11 1111 1111 1111	＝ 3FFF H
第二片	最低地址	01	00 0000 0000 0000	＝4000 H
	最高地址	01	11 1111 1111 1111	＝7FFF H
第三片	最低地址	10	00 0000 0000 0000	＝ 8000H
	最高地址	10	11 1111 1111 1111	＝ BFFF H
第四片	最低地址	11	00 0000 0000 0000	＝ C000 H
	最高地址	11	11 1111 1111 1111	＝ FFFF H

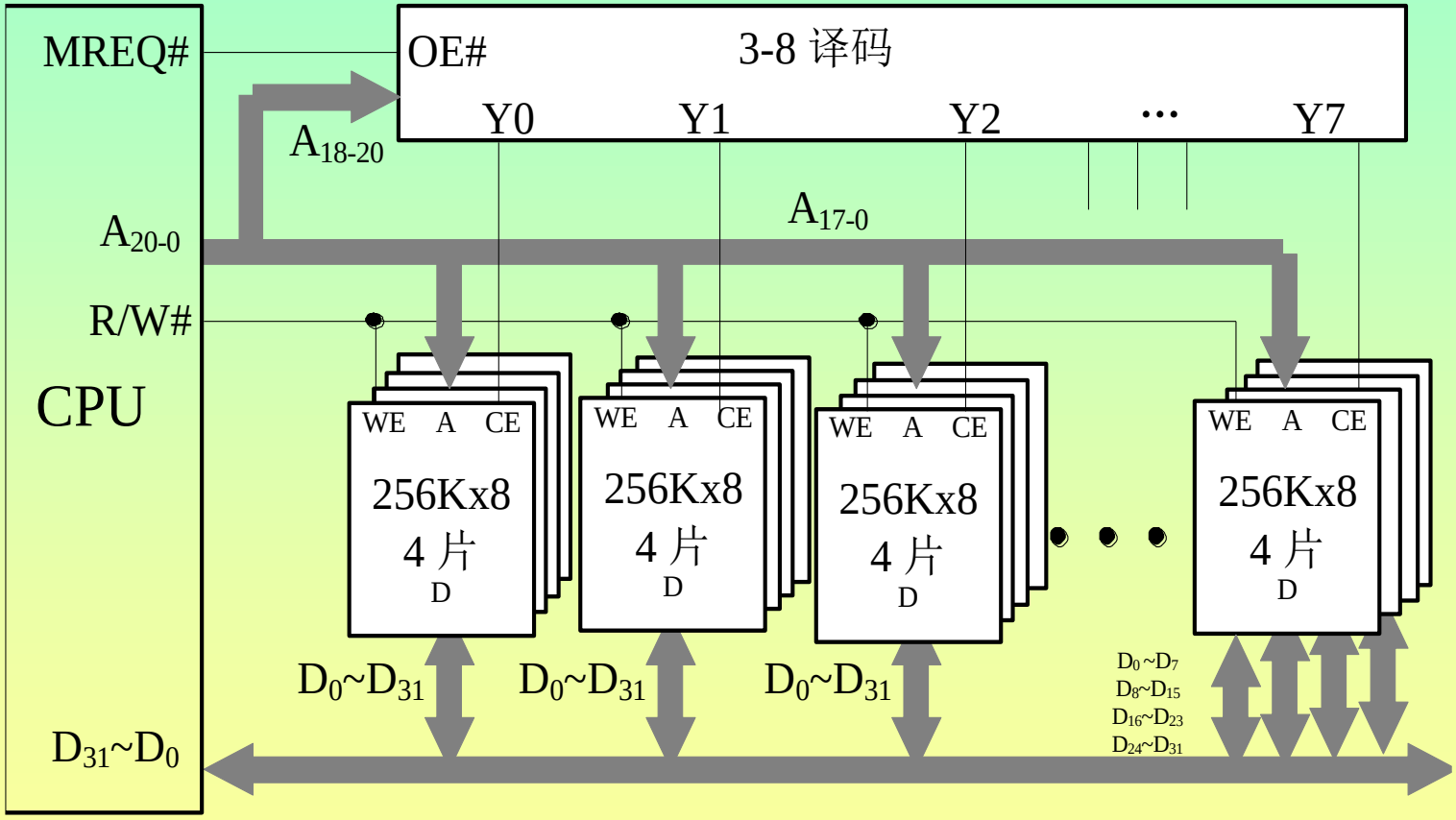
字位同时扩展法：如果存储器芯片的位数和字数都不满足要求，就需要字位同时扩展。用m×n位芯片构成M×N位存储器需要（M/m)×（N/n）个芯片。下图用8片16K×4位芯片构成64K×8位存储器。



字位同时扩展法是以上两种方法的综合，需要完成数据线、地址线、控制线的连接。先完成位扩展，按照位扩展的方式将数据线低位、高位分别接不同芯片，这些位扩展后的芯片形成一组。按照字扩展的方式对这些芯片组的地址线进行连接。注意译码器的选择。最后完成控制线（如读写信号）的连接。

例1：256K×8的芯片组成2M×32的存储器

256K×8的芯片组成2M×32的存储器



例2：为某8位微机系统设计一个具有40 KB RAM和8 KB ROM的存储器。RAM用SRAM芯片6264（8 K×8位）组成，地址从0000 H开始；ROM用EPROM芯片2732（4 K×8位）组成，与RAM地址空间相连。

若芯片的存储容量不一致，如何连接相应的连接线？

$\overline{Y5}$	A12	$\overline{CE}$
0	0	1
0	1	0
1	0	1
1	1	1

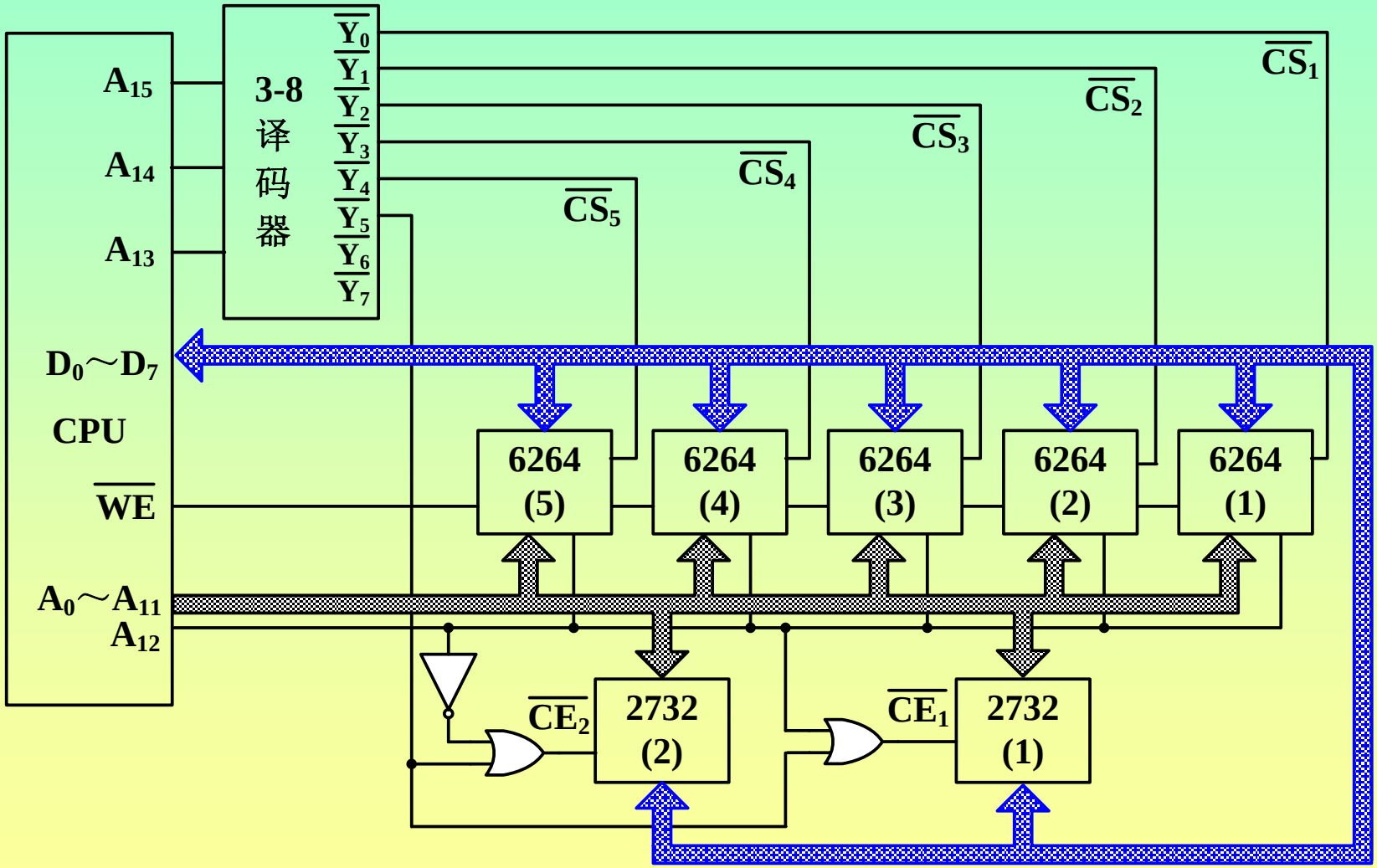
何选择？

少根地址线？4K需要多少根地址线？

地址范围是多少？4K芯片的片选如何连接？

$\overline{Y5}$	A12	$\overline{CE}$
0	0	0
0	1	1
1	0	1
1	1	1

	A15-A13	A12	A0		A15-A13	A12	A0
40K～44K	101	0	0000 0000 0000	～	101	0	1111 1111 1111
44K～48K	101	1	0000 0000 0000	～	101	1	1111 1111 1111


A15
A14
A13
3-8
译
码
器
Y0
Y1
Y2
Y3
Y4
Y5
Y6
Y7
CS1
CS2
CS3
CS4
CS5
D0～D7
CPU
WE
A0～A11
A12
6264
(5)
6264
(4)
6264
(3)
6264
(2)
6264
(1)
CE2
2732
(2)
CE1
2732
(1)

例3：CPU地址总线为A15～A0，数据总线为D7～D0，$\overline{\text{MREQ}}$为允许访存，R/$\overline{\text{W}}$为读写命令。地址空间分配如下：0～8 K为系统程序区，由8K×8的ROM组成；8 K～32 K为用户程序区；由8K×8的RAM芯片组成，地址空间最后2K为系统程序区，由一片2K×8的RAM芯片。请设计存储器系统。

地址空间分配如下：0～8 K为系统程序区，由ROM组成；8 K～32 K为用户程序区；最后2K为系统程序区。

地址空间分配如下：

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	……	A_0	
0	0	0	0	0	0	……	0	一片8 KB EPROM
0	0	0	1	1	1	……	1	
0	0	1	0	0	0	……	0	第一片8 KB SRAM
0	0	1	1	1	1	……	1	
0	1	0	0	0	0	……	0	第二片8 KB SRAM
0	1	0	1	1	1	……	1	
0	1	1	0	0	0	……	0	第三片8 KB SRAM
0	1	1	1	1	1	……	1	
	…		…		…	…		空（30 KB）
1	1	1	1	1	0	……	0	一片2 KB SRAM
1	1	1	1	1	1	……	1	

最后8K地址空间分配如下：

Y7	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	……	A_0		$\overline{CE}$
0	1	1	1	0	0	0	……	0	第一个2 KB 空间	1
0	1	1	1	0	0	1	……	1		
0	1	1	1	0	1	0	……	0	第二个2 KB 空间	1
0	1	1	1	0	1	1	……	1		
0	1	1	1	1	0	0	……	0	第三个2 KB 空间	1
0	1	1	1	1	0	1	……	1		
0	1	1	1	1	1	0	……	0	第四个2 KB 空间	0
0	1	1	1	1	1	1	……	1		

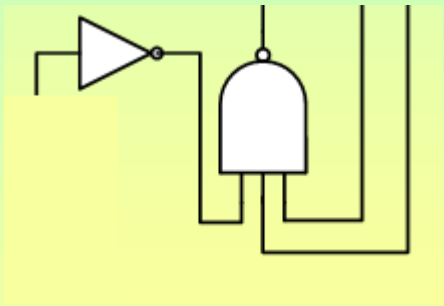
最后2K的片选逻辑电路如何设计？

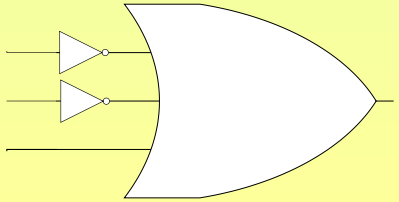
1、与最后2K地址片选有关的信号有哪些？

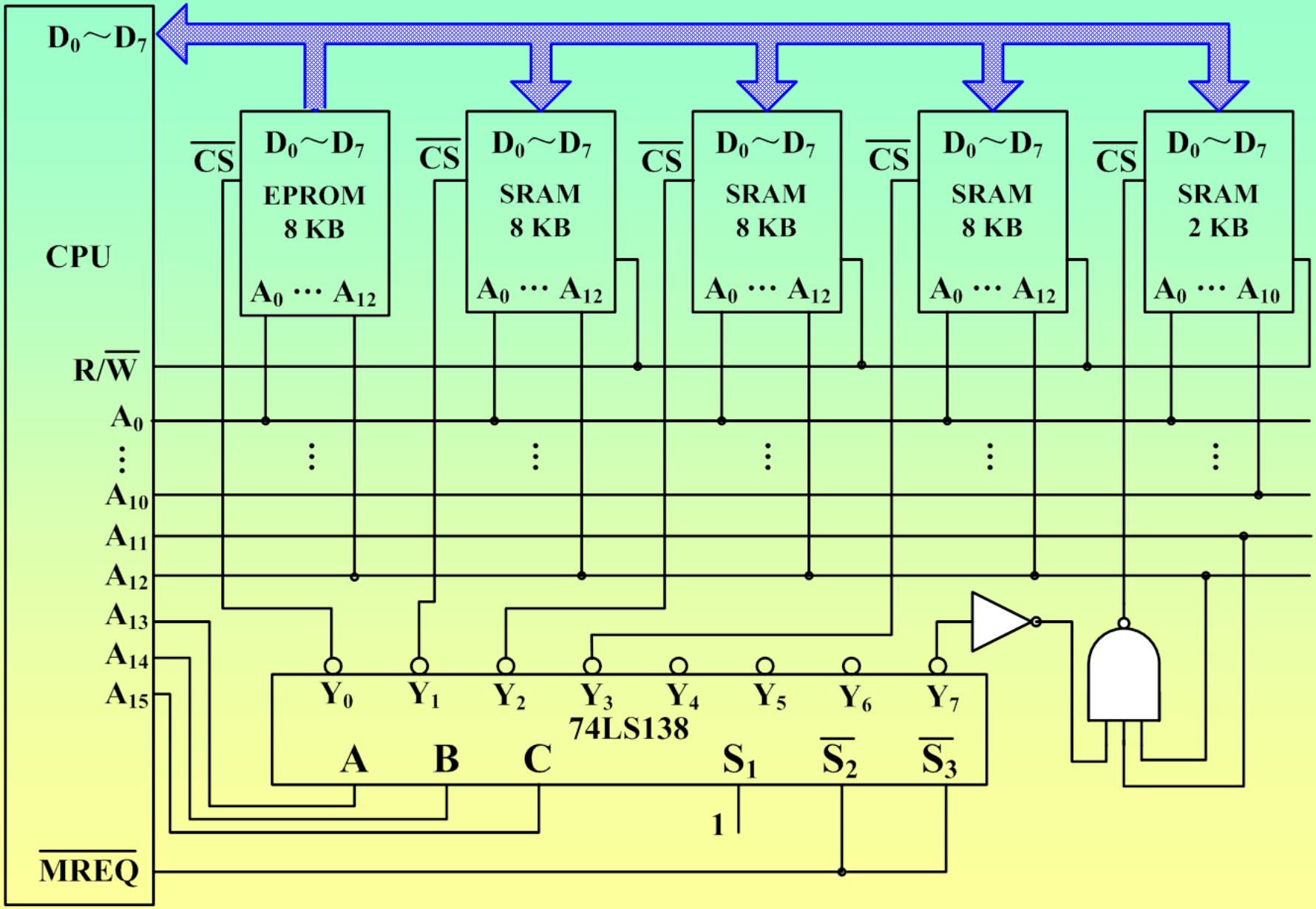
2、

Y7	A12	A11	CE
1	×	×	1
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0

→ 相关的信号都转化为全0或者全1






D0~D7
CPU
R/W
A0
A10
A11
A12
A13
A14
A15
MREQ
CS
D0~D7
EPROM
8 KB
A0 ··· A12
SRAM
8 KB
SRAM
8 KB
SRAM
8 KB
SRAM
2 KB
A0 ··· A10
Y0 Y1 Y2 Y3 Y4 Y5 Y6 Y7
74LS138
A B C S1 S2 S3
1

例4：某8位机器中，地址总线12根，数据总线8根（D7～D0）控制总线中与主存有关的有（访存允许，低电平有效，读写控制信号，高电平为读命令，低电平为写命令）。选片译码电路使用2:4译码器，主存的地址空间分布如下：

1)用1K×8位的ROM芯片构成一个地址空间为2KB的系统程序区；起始地址为000H。

2)用2K×4位的RAM芯片构成2KB的用户程序区,与ROM地址空间相连

要求：

1）计算所需的芯片数目；

2）画出地址空间分布图；

3）画出主存和CPU的连接逻辑图。

1)系统程序区大小2K×8位；用户程序区2K×8位。
故需要芯片：
1K×8位ROM 2片；2K×4位RAM芯片2片。

2）地址空间分布图如图：

000~3FFH	ROM 系统程序区
400~7FFH	
800~FFFH	RAM 用户程序区

1、需要多少地址线对RAM芯片寻址？

用户程序区2K×8位 **A0~A10**

2、ROM芯片需要多少地址线？

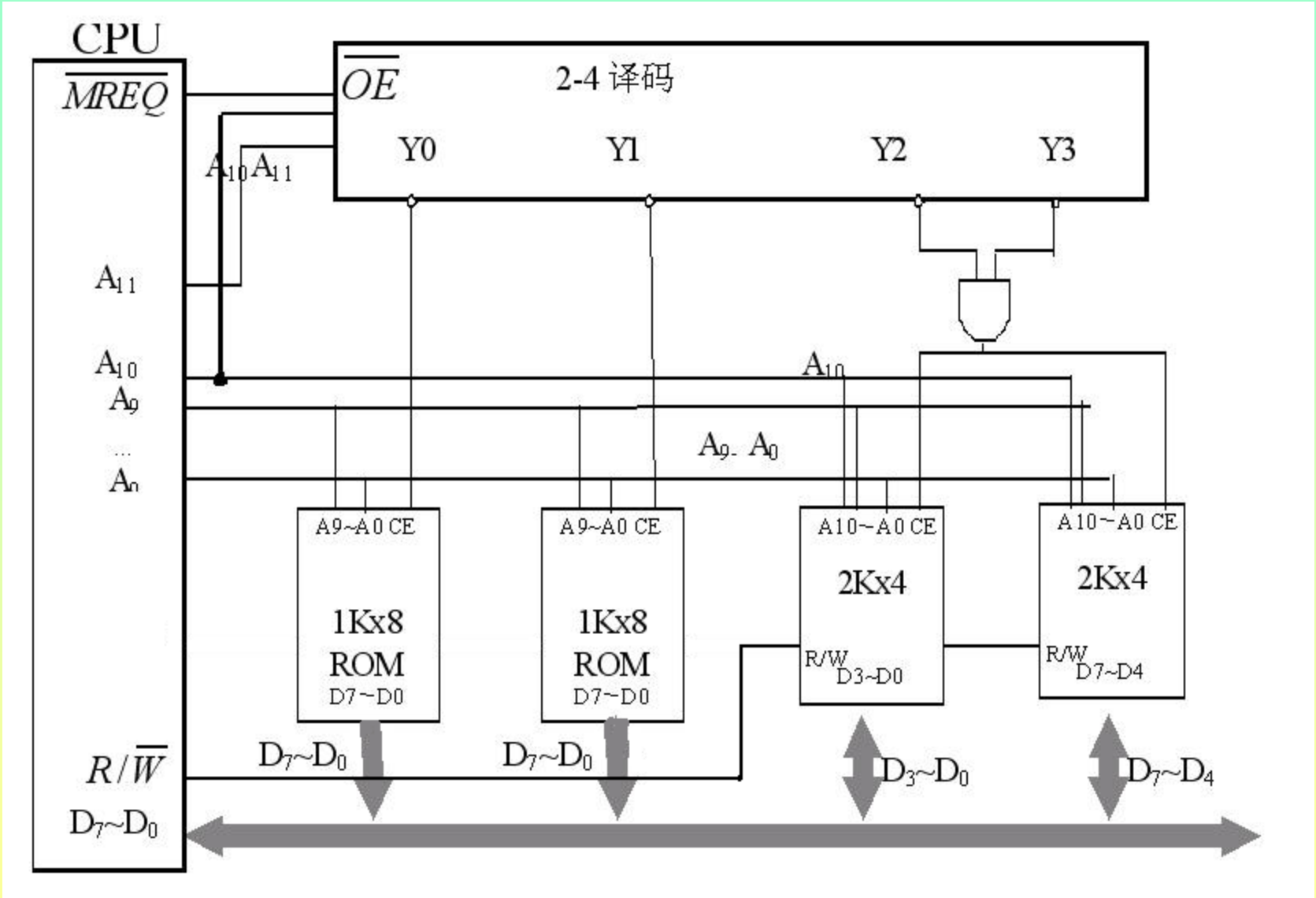
1K×8位 **A0~A9**

3、需要几位的地址译码作为片选？ **A10、A11**

4、地址线A10如何处理？

5、RAM区域对应着几个译码片选信号？

- 分析：使用2：4译码器，译码的使能端～OE连接～MEMQ，将A10、A11进行译码；ROM的地址线A0～A9，2片，其CE分别用Y0、Y1连接；
- RAM芯片，2片，地址线A0～A10，CE分别用Y2、Y3相与后连接；数据线D0～D7直接和ROM相连，分成两组D0～D3、D4～D7和RAM相连；～R/ W和RAM的读写控制端相连。


CPU
MREQ
A11
A10
A9
An
R/W
D7~D0
OE
2-4 译码
Y0
Y1
Y2
Y3
A10 A11
A10
A9~A0
A9~A0 CE
1Kx8
ROM
D7~D0
A10~A0 CE
2Kx4
R/W
D3~D0
D7~D4

注意：

- A10既参加译码又作为片内地址寻址

A11 A10 A9 A8 … … A0

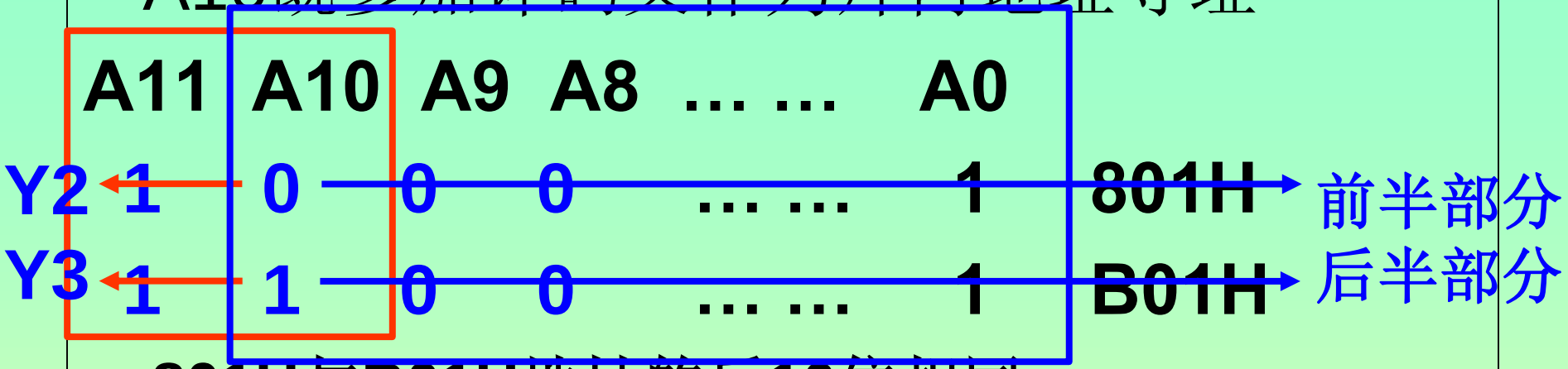
1 0 0 0 … … 1 801H

1 1 0 0 … … 1 B01H

801H与B01H地址的后10位相同

- 译码片选Y2 Y3 同时用来片选RAM

- A10既参加译码又作为片内地址寻址



801H与B01H地址的后10位相同

- 译码片选Y2 Y3 同时用来片选RAM

思考：A10的两种作用会不会起“冲突”？

不会！

A10与A11配合，选Y2或Y3(A10=0或1)

A10与A0~A9配合，片内选址，

芯片的前半部分（A10=0）

或后半部分（A10=1）

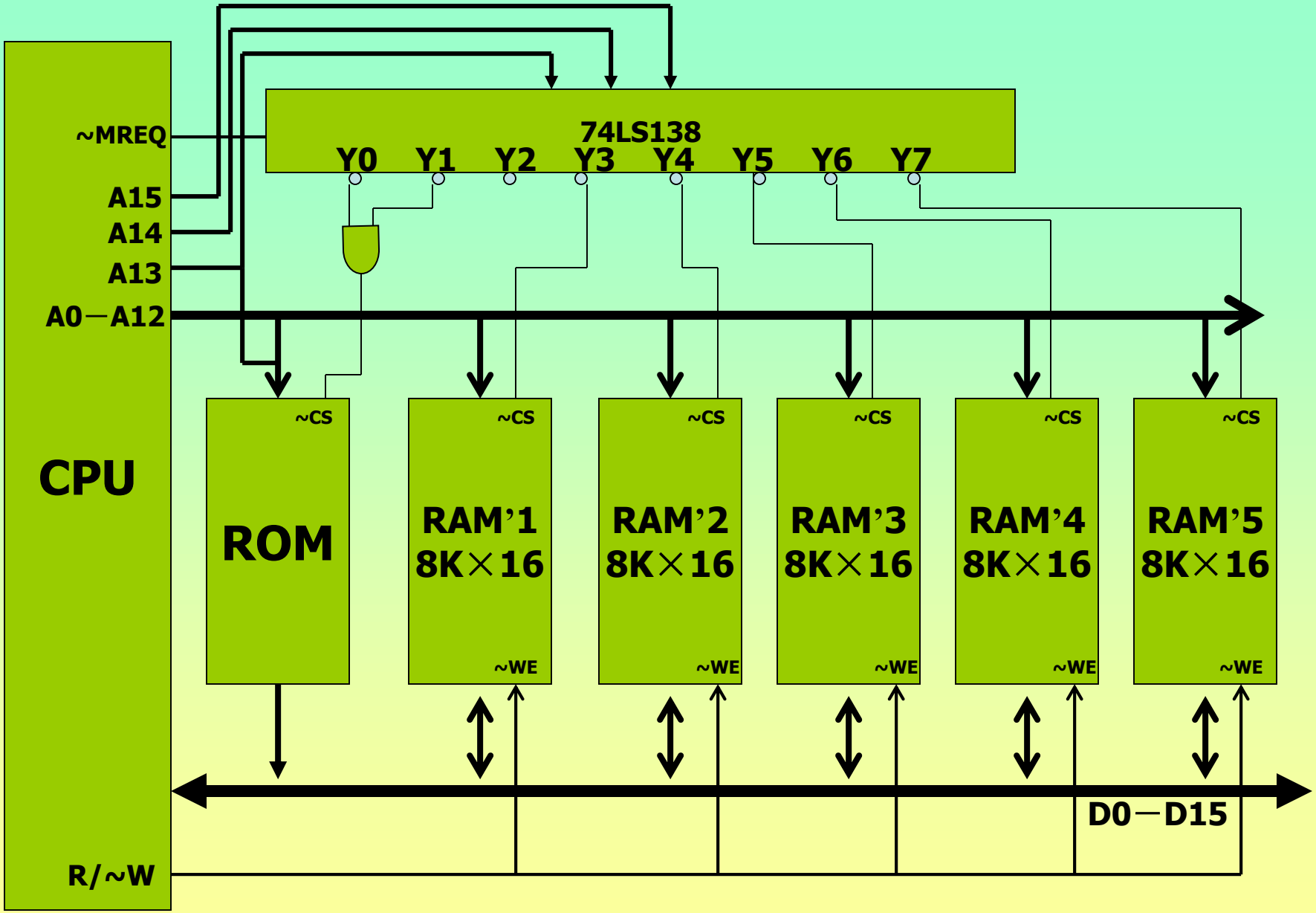
例5：P111.7

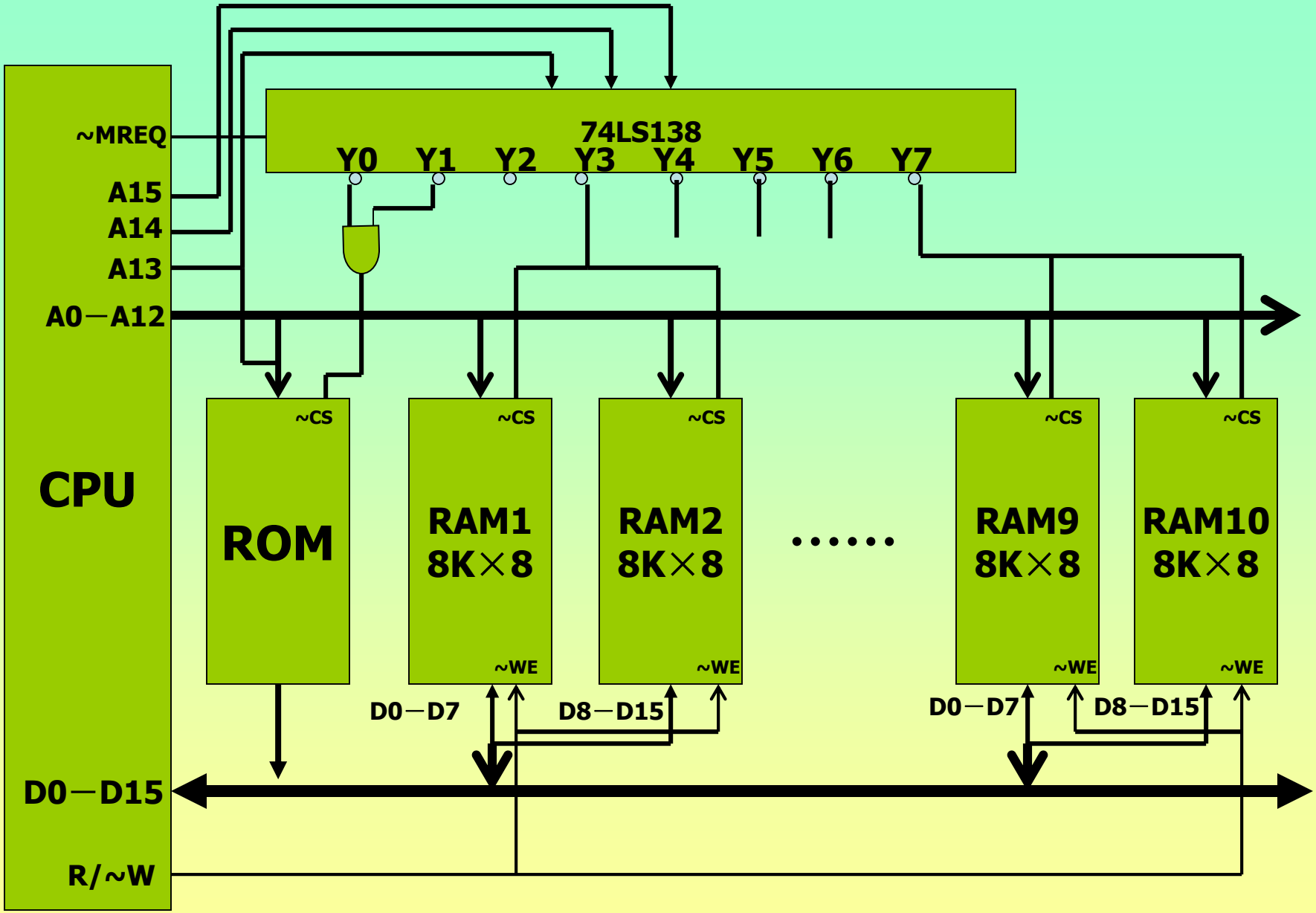
地址空间0000H—3FFFH为ROM。RAM（8K×8）组成40K×16，起始地址为6000H。

例5：P125.7

- 解：存储器的地址空间分布如下图所示：

- RAM1’～ RAM5’是8K×16的模块，需要两片的8K×8的RAM采用位扩展的方法组成。
- 存储器的地址空间可以看做分为8组，每组8K，其中ROM区域看做2组，所以需要采用3：8译码来控制片选信号。（也可以采用将各个模块的起止地址写出，找出其中的规律进行译码。提示：采用地址线的高2位进行译码，就是将地址空间平均分为4组，高3位译码就是将地址空间平均分为8组）
- CPU的R/W信号与RAM的~WE连接，ROM只读不写。
- ROM区域的地址空间有16K，16K=2^14，所以ROM的片内寻址地址需要有14位，A13既作为片内寻址，也用来译码片选。
- ROM区域的地址空间有16K，译码后的结果对应着两组，因为低电平有效，所以采用与门来实现。


~MREQ
A15
A14
A13
A0—A12
R/~W
CPU
74LS138
Y0 Y1 Y2 Y3 Y4 Y5 Y6 Y7
~CS
ROM
RAM'1
8K×16
RAM'2
8K×16
RAM'3
8K×16
RAM'4
8K×16
RAM'5
8K×16
~WE
D0—D15


~MREQ
A15
A14
A13
A0—A12
CPU
D0—D15
R/~W
74LS138
Y0 Y1 Y2 Y3 Y4 Y5 Y6 Y7
~CS
ROM
RAM1
8K×8
~WE
RAM2
8K×8
.....
RAM9
8K×8
RAM10
8K×8
D0—D7
D8—D15

- ROM芯片为只读存储器，
  所以数据只读不写，数据线为单向的，
  不需要读写控制信号RW

存储器与CPU的连接完成数据线、地址线、控制线的连接

数据线：位扩展的方式，位数不够时将高位、低位数据线分别连接；数据线是双向的（ROM除外）。

控制线：读写信号并联接入CPU；译码器的片选信号也直接接入CPU的访存允许；存储器片选信号由高位地址经过译码器得到的结果给出，注意两种情况：一个译码结果对应多个芯片的片选，或几个译码结果对应一个芯片的片选，有时还需要与低位一起经过门电路后进行片选。RW读写控制信号要接RAM。

地址线：高位进行译码，低位直接接入芯片，注意接入芯片进行片内选址的地址位数，芯片多大，就需要多少根片内地址。两种情况：位数不够或位数太多。地址线为单向的。

地址线：

1、16K 8K 4K 2K 1K 别需要多少根地址线？

片内寻址就与芯片的容量大小有关

2、5片8K芯片组成40K，高几位进行译码？

高3位，A13\A14\A15

3、A12\A13\A14三位进行地址译码时，一个译码的结果(Y0-Y7)对应多大的地址空间？

A0-A11，2^12 = 4K

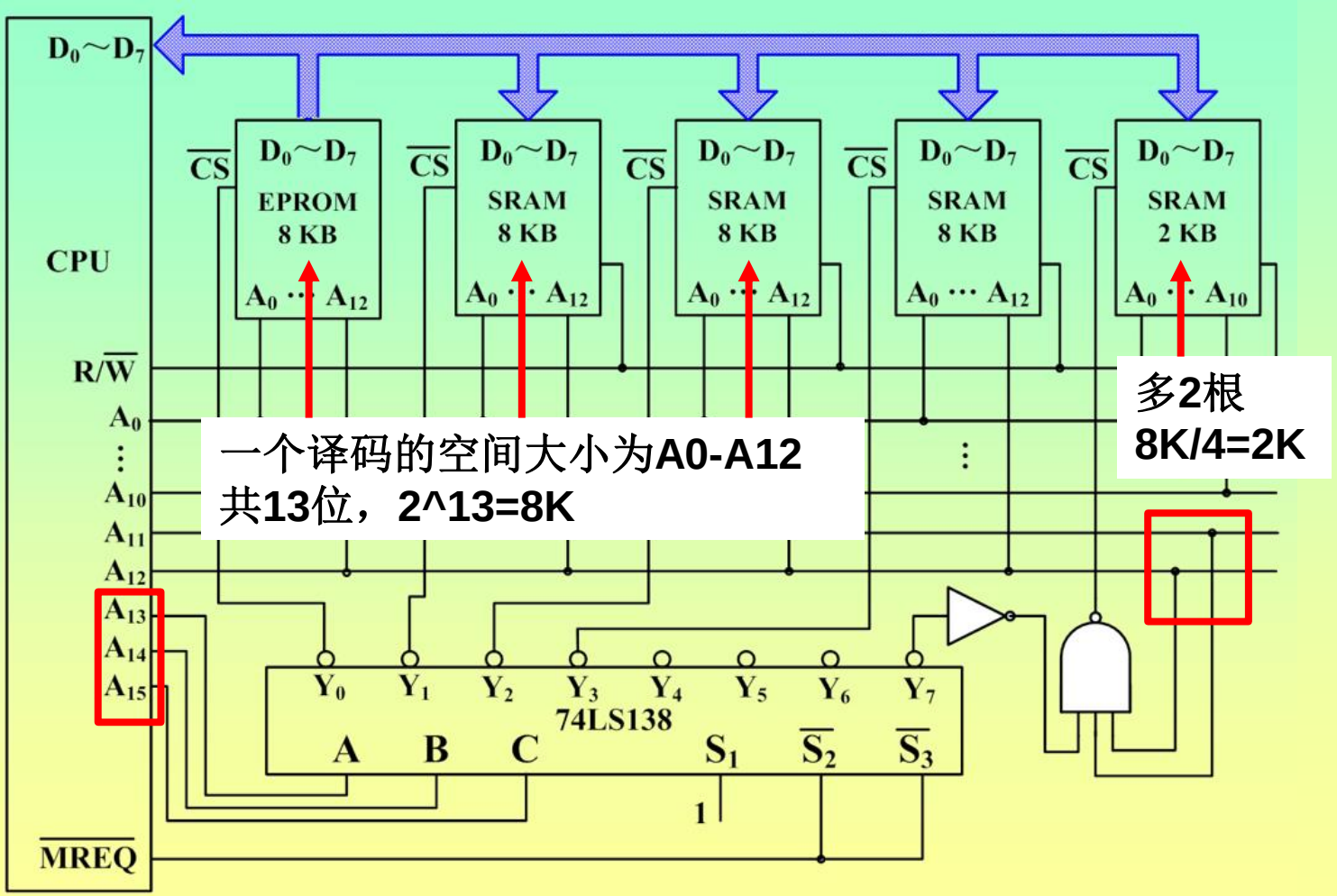
4、若译码的结果(Y0-Y7)对应4K地址空间，如何选择其中的1K的地址空间？

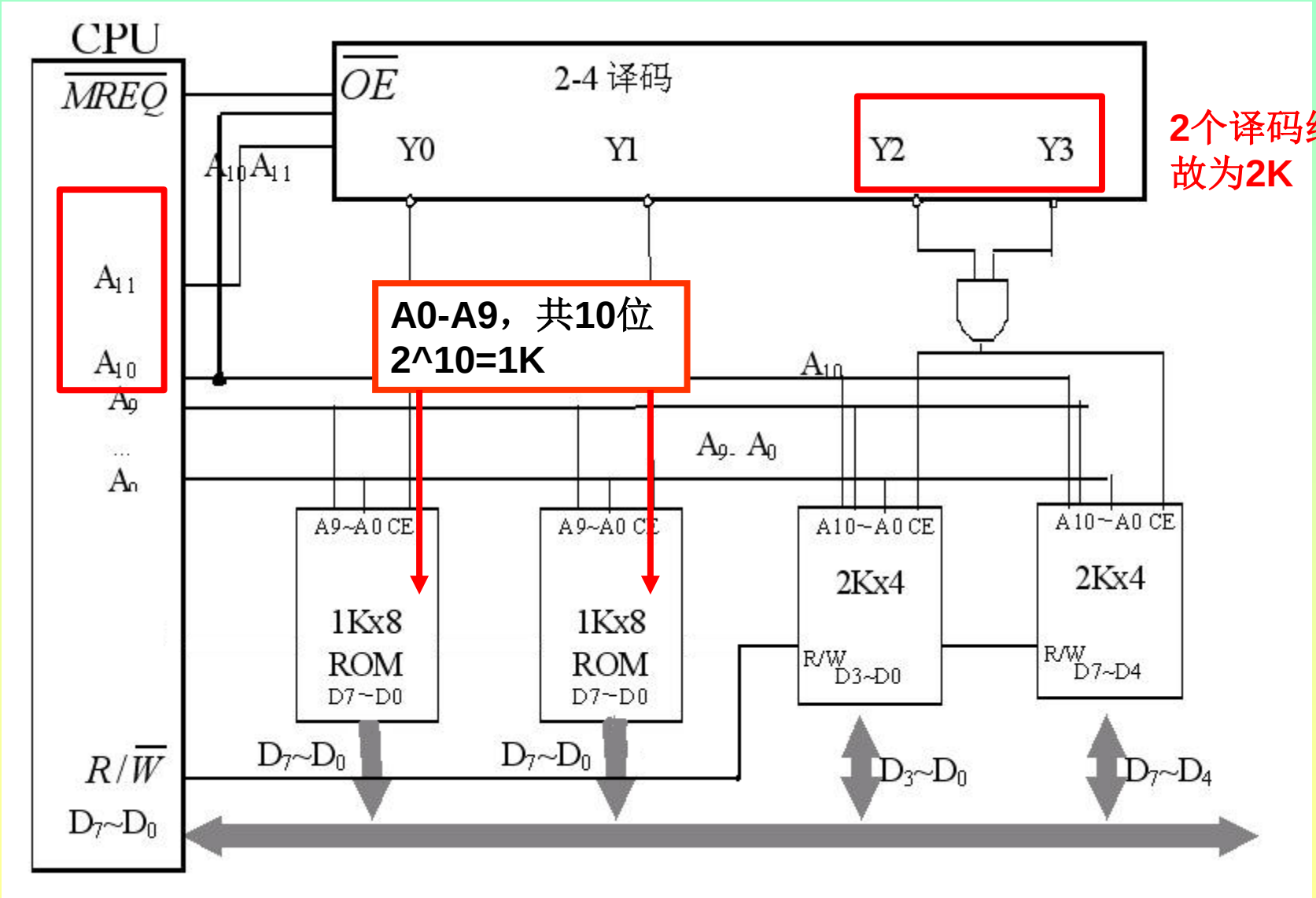
4K/1K = 4 = 2^2 高2位来处理

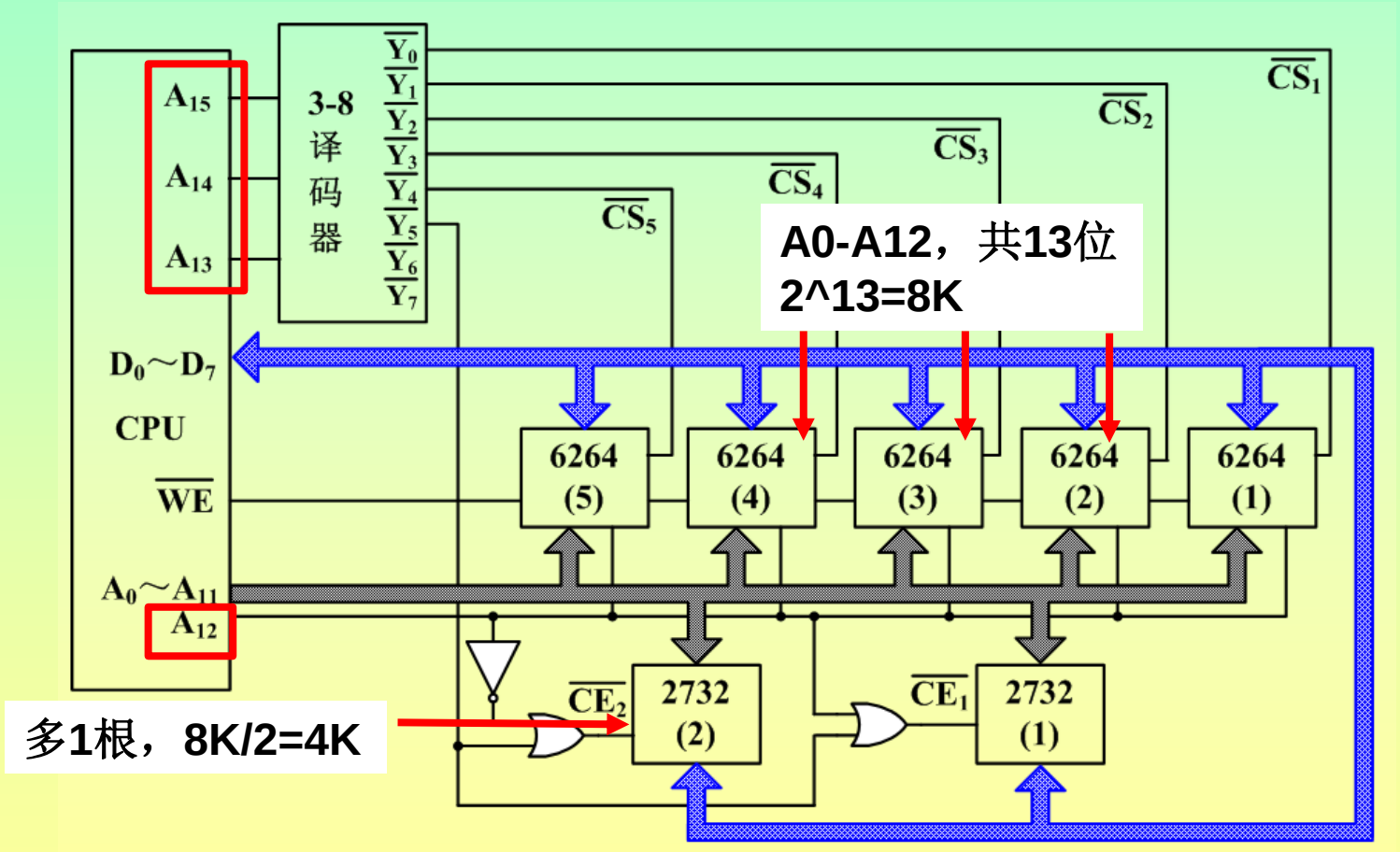
1K= 2^10 A0-A9 接下去2位 A10 A11

地址线：

学会从CPU的连接图上反推断地址线的根数。




CPU
MREQ
A11
A10
A9
...
A0
R/W
D7~D0
OE
2-4 译码
Y0
Y1
Y2
Y3
A10 A11
A0-A9，共10位
2^10=1K
2个译码结果
故为2K
A10
A9~A0
A9~A0 CE
1Kx8
ROM
D7~D0
A10~A0 CE
2Kx4
R/W
D3~D0
D7~D4
D7~D0
D3~D0

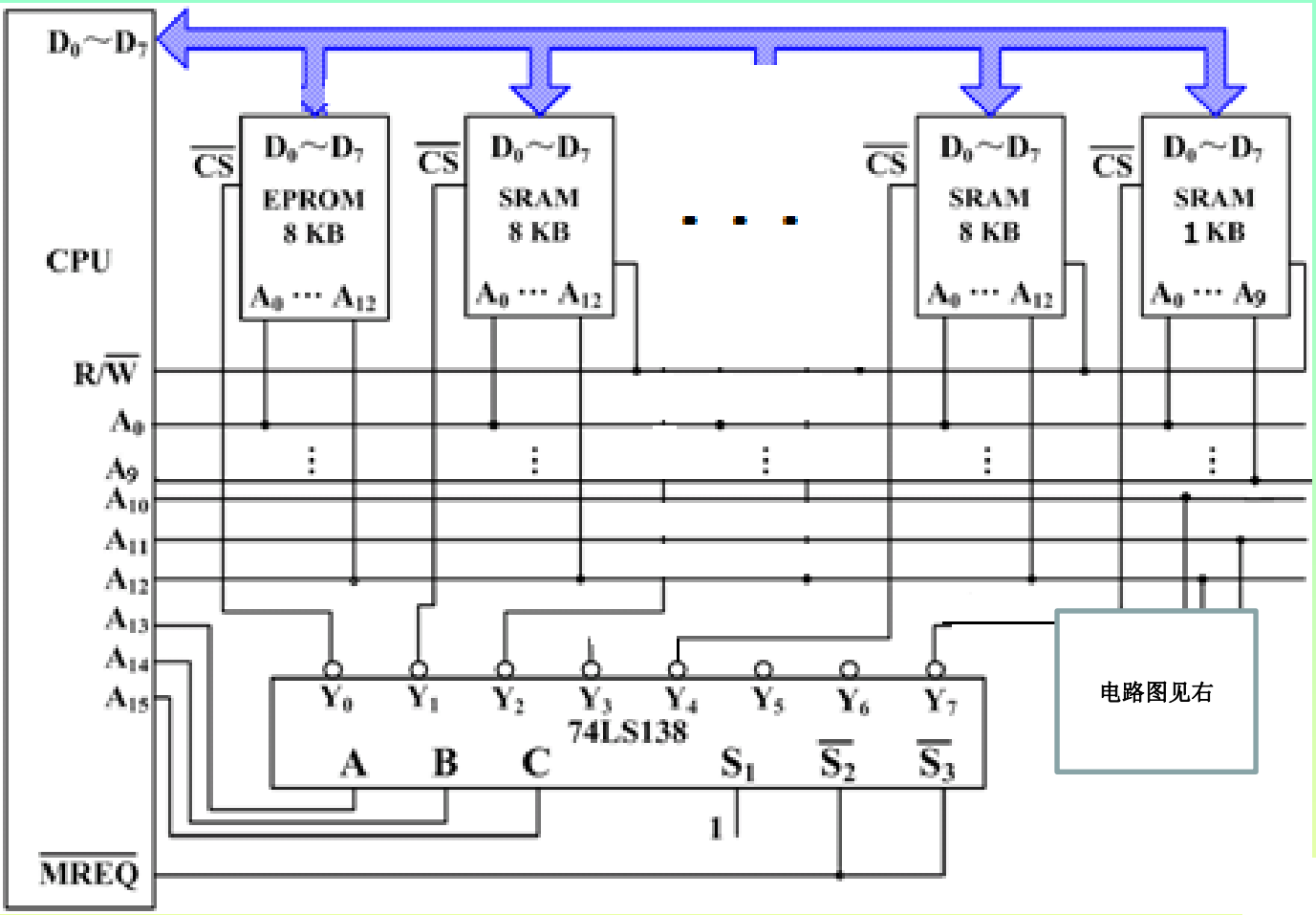

A15
A14
A13
3-8
译
码
器
Y0
Y1
Y2
Y3
Y4
Y5
Y6
Y7
CS1
CS2
CS3
CS4
CS5
A0-A12，共13位
2^13=8K
D0～D7
CPU
WE
A0～A11
A12
6264
(5)
6264
(4)
6264
(3)
6264
(2)
6264
(1)
CE2
2732
(2)
CE1
2732
(1)
多1根，8K/2=4K

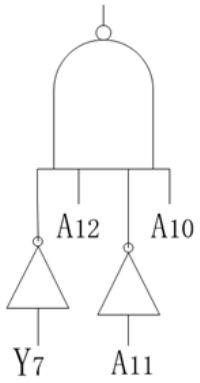
三、某 CPU 地址总线为 A15～A0，数据总线为 D7～D0，$\overline{MREQ}$为允许访存信号，RAM 芯片有$\overline{CS}$和$\overline{WE}$信号控制端。存储空间分配如下：系统程序区 8KB，起始地址为 0000H，由 8K×8 的 ROM 组成；用户程序区共 32KB，起始地址为 2000H，由 8K×8 的 RAM 芯片组成；系统程序区 1KB，起始地址为 F400H，由 1K×8 的 RAM 芯片组成。存储器按字节访问，请设计该系统。

1）计算所需各个芯片数。（2 分）

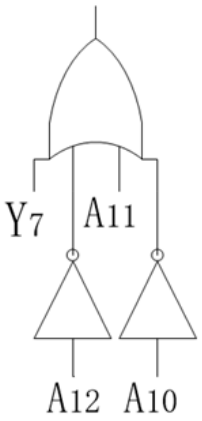
2）画出地址空间分布图；（2 分）

3）画出 RAM、ROM 与 CPU 的连接图以及相关的控制信号。（12 分）


D0~D7
CPU
R/W
A0
A9
A10
A11
A12
A13
A14
A15
MREQ
CS
D0~D7
EPROM
8 KB
A0 … A12
SRAM
8 KB
A0 … A12
SRAM
8 KB
A0 … A12
SRAM
1 KB
A0 … A9
Y0 Y1 Y2 Y3 Y4 Y5 Y6 Y7
74LS138
A B C S1 S2 S3
1
电路图见右


A12
A10
Y7
A11

或


Y7
A11
A12
A10

- 评分标准：

1、基本结构对（有译码、有芯片）

2、有MREQ信号

3、A13、A14、A15作为高位译码信号

4、RAM有WE信号，ROM不能有WE

5、片选的CS有连到3:8译码端Y0-Y4，Y7

6、地址线8K的为A0-A12

7、地址线1K的为A0-A9

8、数据线D0-D7有标

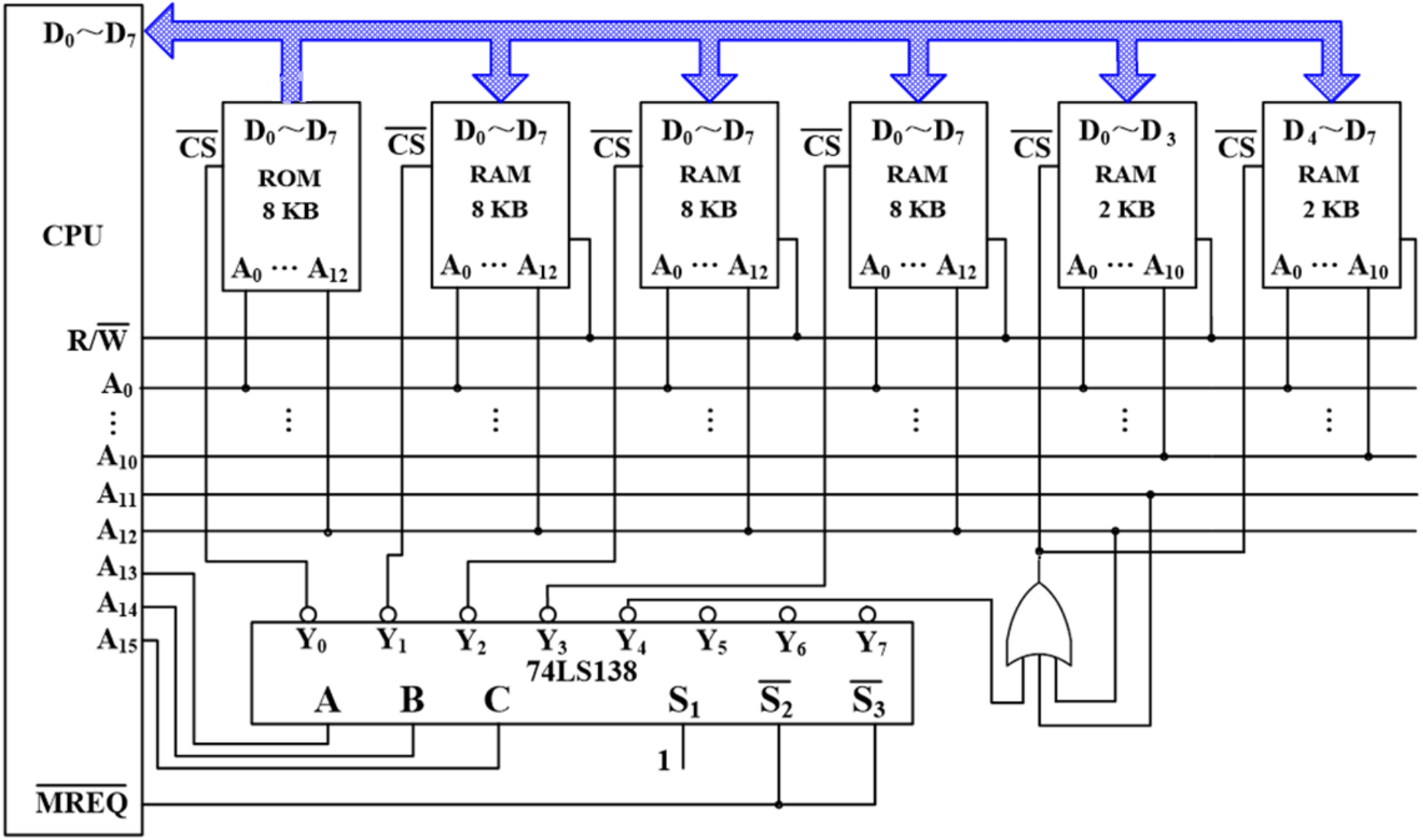
9、ROM数据线单向的

以上一点为1分

10、最后的门电路图 （3分）

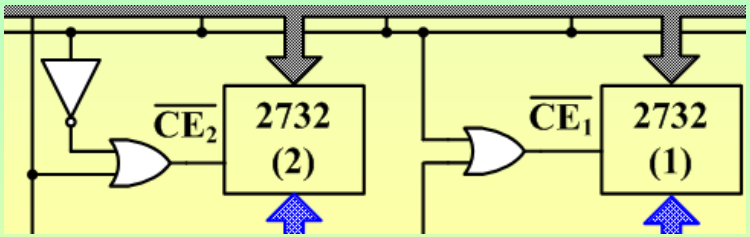
二、 （14 分）某 CPU 地址总线为 A0～A15，数据总线为 D0～D7，$\overline{MREQ}$为允许访存，R/$\overline{W}$为读写命令。地址空间分配如下：0～8 K 为系统区，由 8K×8 的 ROM 组成；8 K～32 K 为用户区，由 8K×8 的 RAM 芯片组成；程序区有 2K，由 2K×4 的 RAM 芯片组成，程序区的地址空间与用户区的地址空间相连。

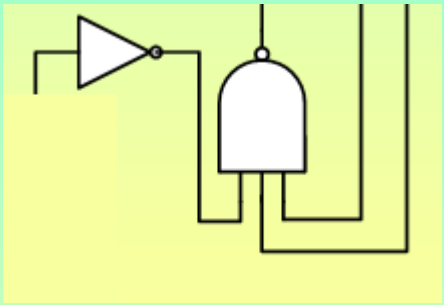
1. 计算所需的芯片。
2. 画出地址空间分布图。
3. 请画出 CPU 与存储器系统的连线，并注明相关的控制信号。

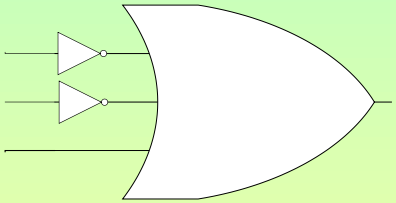

D0~D7
CPU
R/W
A0
A10
A11
A12
A13
A14
A15
MREQ
CS
D0~D7
ROM
8 KB
A0 ··· A12
RAM
8 KB
RAM
8 KB
RAM
8 KB
D0~D3
RAM
2 KB
A0 ··· A10
D4~D7
RAM
2 KB
Y0 Y1 Y2 Y3 Y4 Y5 Y6 Y7
74LS138
A B C S1 S2 S3
1

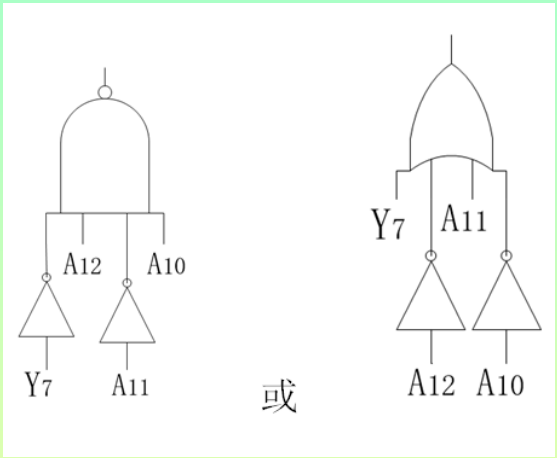
- 有CPU、芯片、译码器（1分）
- A13-A15接138译码器，Y0-Y3外接片选（1分）
- 最后2K位扩展，标明D0-D3，D4-D7（1分）
- 8K芯片有标明D0-D7 （1分）
- R/W信号有接芯片,ROM不能接R/W（1分）
- 8K芯片的地址线为A0-A12（1分）
- 2K芯片地址线为A0-A10（1分）
- MREQ有接（1分）
- 最后2K的片选接对 （1分）

思 考









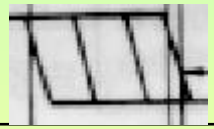
- 不用门电路，用相对应的2:4（3:8）译码器是否可以实现所需功能？

时序图

时序图提供的信息：

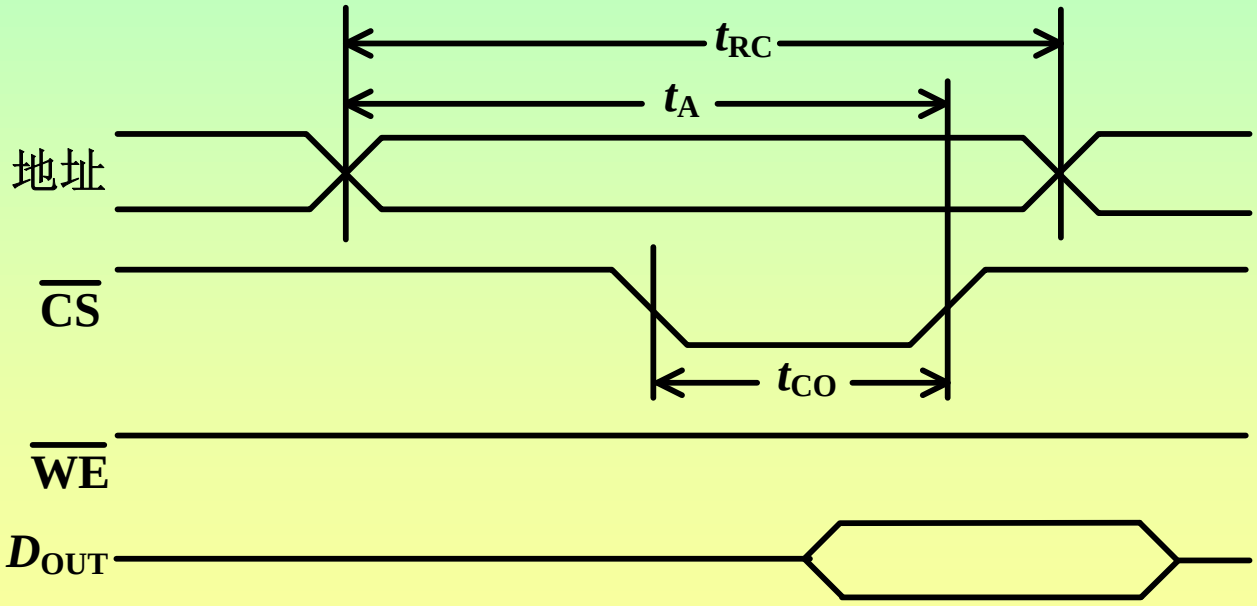
- 各输入信号的有效情况
- 各输入信号的极限时间参数（如最小脉冲宽度）
- 各输入信号之间应遵循的极限时间配合关系（如建立时间）
- 各输出的最大传输延迟

常用的时序符号

符号	输入	输出
———	稳定的‘H’或‘L’态	稳定的‘H’或‘L’态
	由‘H’向‘L’变化，斜线为过渡区	
	允许由‘L’向‘H’变化或由‘H’向‘L’变化	不知状态如何变化
		中线为高阻态
	中间为信息有效，其它的为信息无效	

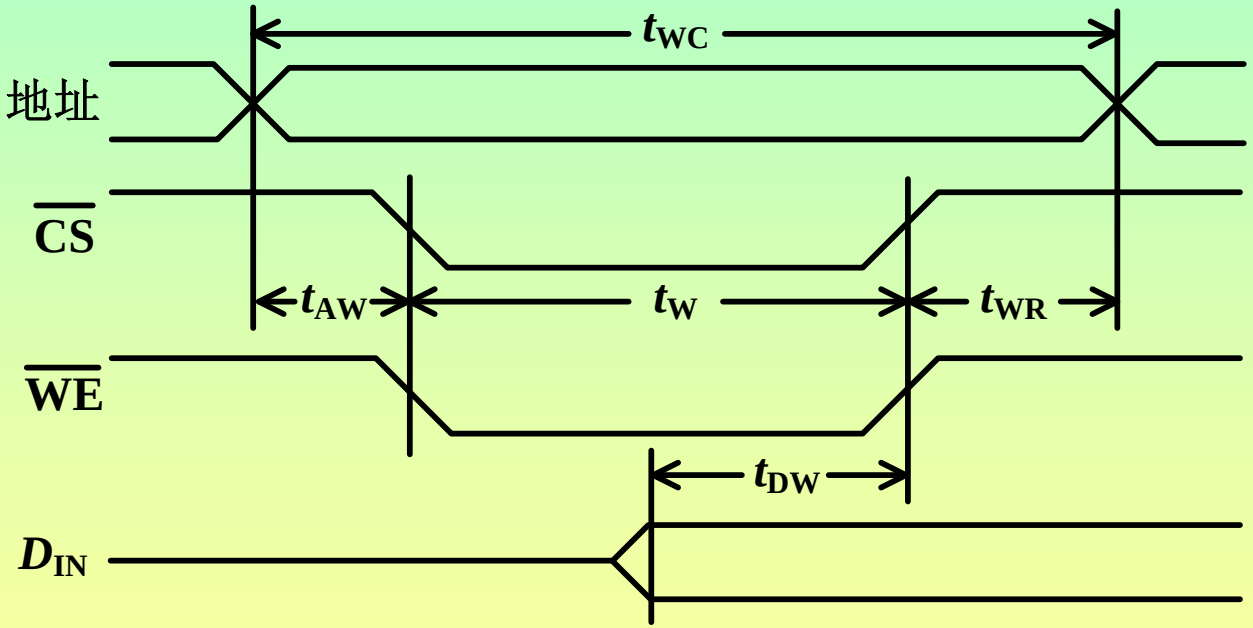
存储器的读、写周期

读周期：从给出有效地址，经过译码电路、驱动电路延迟，到读出选中单元内容，再经过I/O电路延迟后在外部数据总线上稳定地出现所读出的数据信息，这一过程所需时间为t_A，称为读出时间。读周期t_{RC}表示存储器进行两次连续读操作所必须间隔的时间，它大于等于读出时间。片选信号$\overline{CS}$必须保持到数据稳定输出，t_{CO}为片选保持时间。读周期中$\overline{WE}$为高电平。



t_{RC} — 读周期
t_A — 读出时间
t_{CO} — 片选保持时间

写周期：要实现写操作，要求$\overline{CS}$和$\overline{WE}$都为低电平。为使数据总线上的数据可靠写入存储器，要求$\overline{CS}$和$\overline{WE}$同时有效的宽度至少为t_W。为了使地址变化期间不会把信息写入错误地址，WE在地址变化期间必须为高电平。为了保证$\overline{WE}$、CS无效前把数据可靠地写入，数据必须在t_{DW}前稳定地出现在数据总线上。



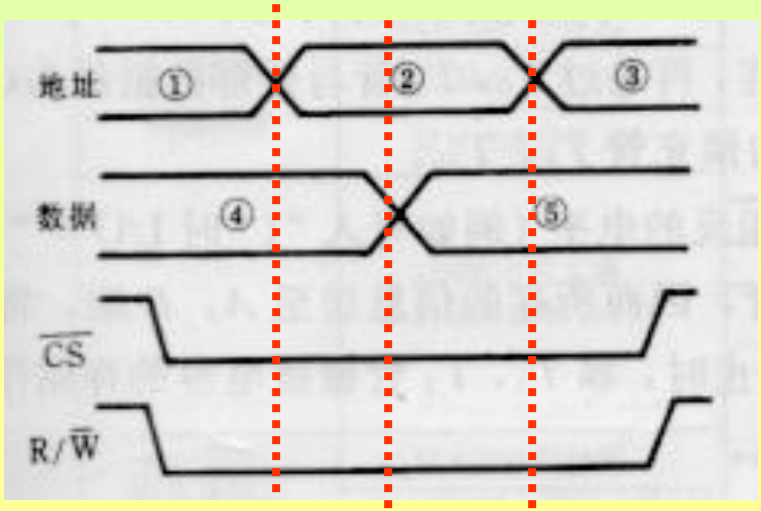
t_{WC} — 写周期
t_W — 写数时间
t_{WR} — 写恢复时间
t_{DW} — 数据有效时间

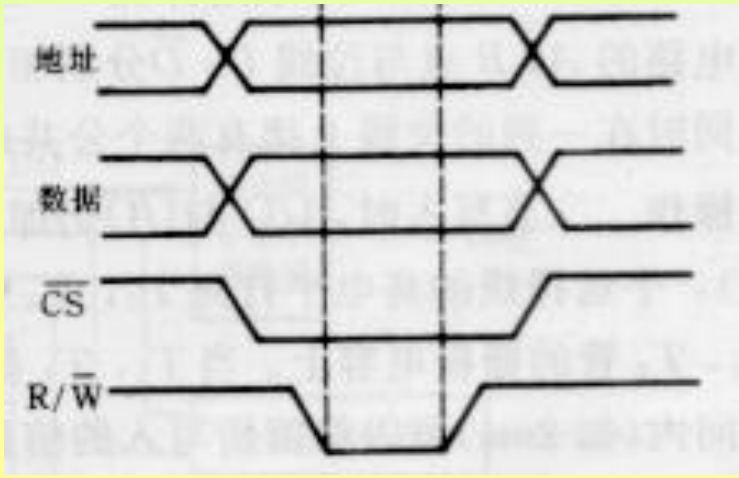
如图为SRAM的写入时序图，请指出图中错误地方，并画出正确的时序图

读写信号加负脉冲时，地址线和数据线应稳定

错误1：读写信号加低电平时，数据线改变，会存入新的数据

错误2：读写信号加低电平时，地址线改变，会存入新的地址



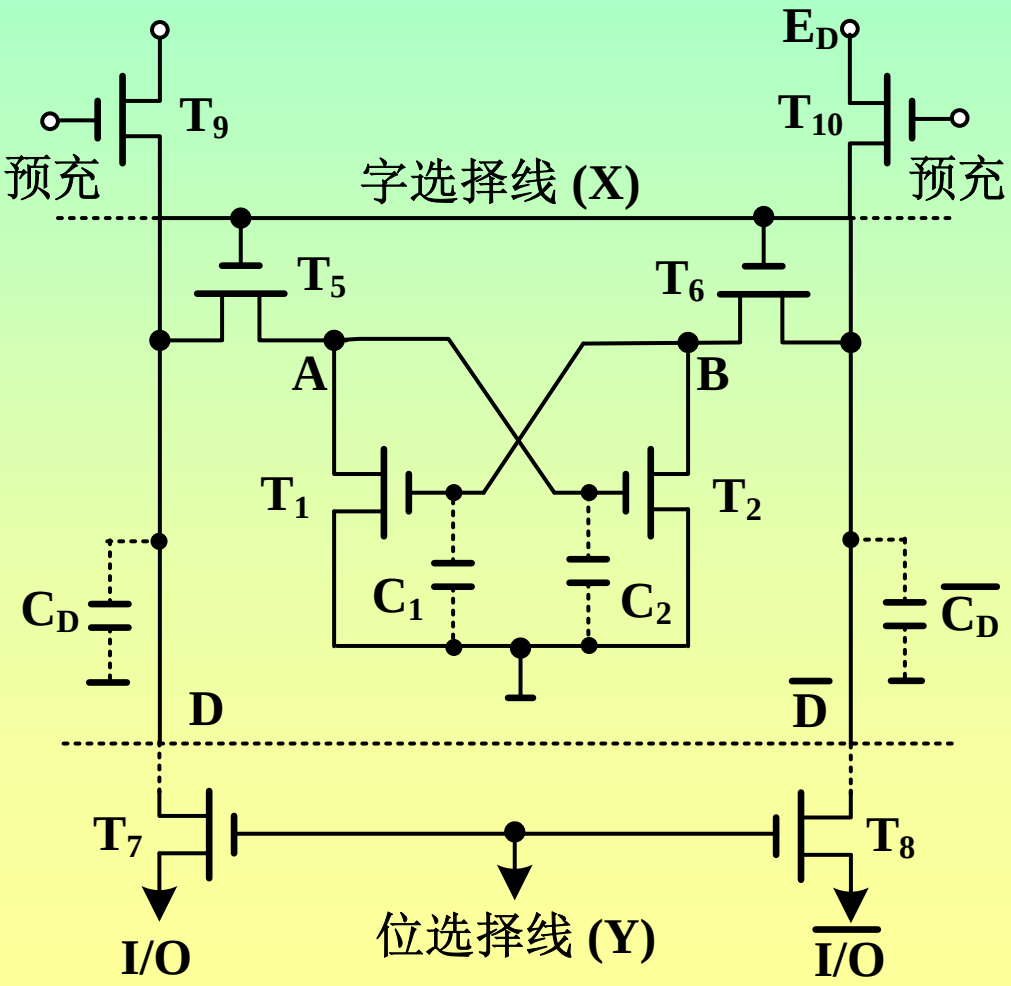


动态随机读写存储器DRAM

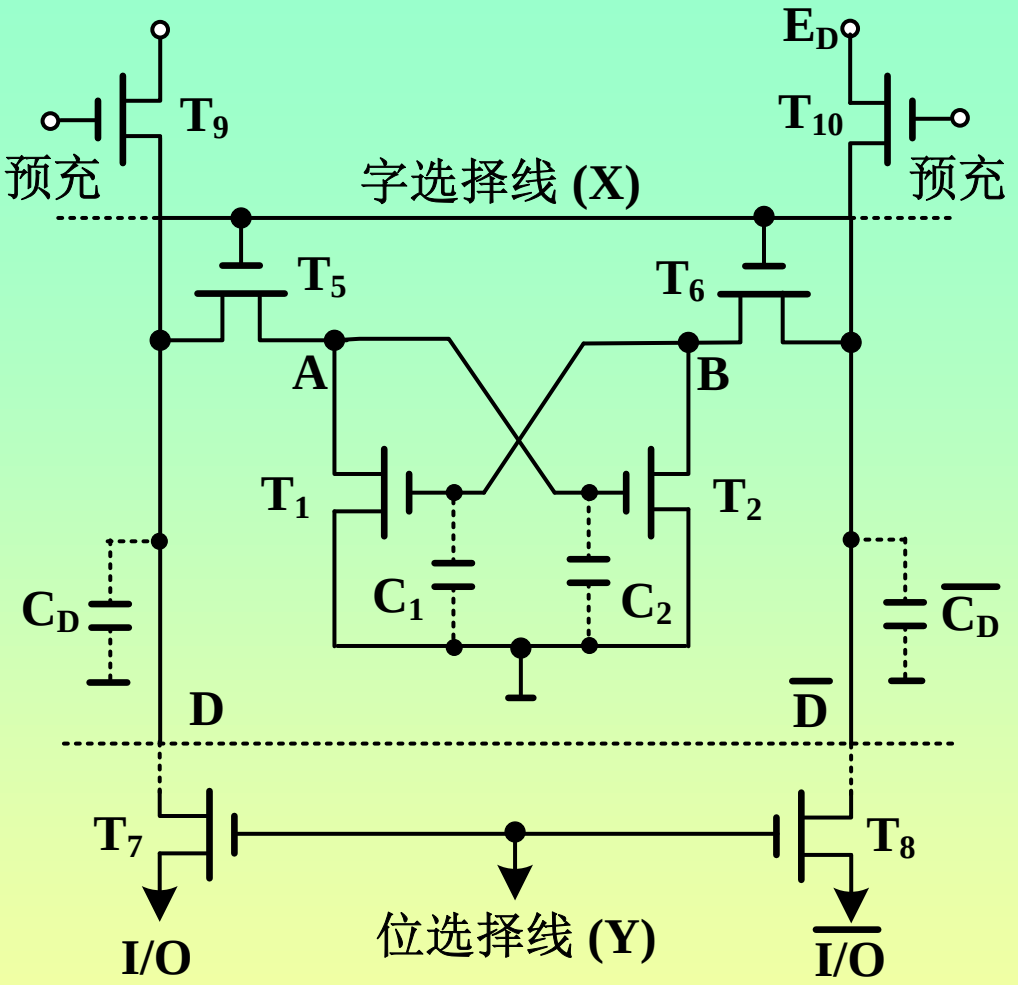
将六管静态存储元电路的负载管T_3和T_4去掉，就得到四管动态存储元电路。数据以电荷的形式存储在T_1和T_2的栅极电容C_1和C_2上。

若C_2充电，C_1未充电，则T_2导通，T_1截止，A为高电平，B为低电平，这一状态为存储元的“1”状态。若C_1充电，C_2未充电，则A低电平，B为高电平，这一状态为存储元的“0”状态。

写操作时，字选择线X和位选择线Y都为高电平，T_5～T_8导通，给I/O和$\overline{I/O}$加相反电平，把信息存储在栅极电容C_1和C_2上。



读操作时，先经过T_9和T_{10}对位线D和D上的分布电容C_D和C_D充电，然后打开T_5和T_6管。假定存储元处于“1”状态，则A为高电平，B为低电平，T_1截止，T_2导通。C_D上的电荷经T_2释放掉，C_D上的电荷可以保持，A点和B点的状态就转移到位线D和D上。同时，C_D上的电荷可以向C_2补充，所以读出的过程也是刷新的过程。



由于栅极电容存在泄漏，必须每隔一段时间对C_1或C_2上的电荷进行补充，称为“刷新”。刷新的过程和读操作类似，先对位线的分布电容C_D和C_D充电，再给字选择线加一个脉冲，打开T_5和T_6，由分布电容对栅极电容充电

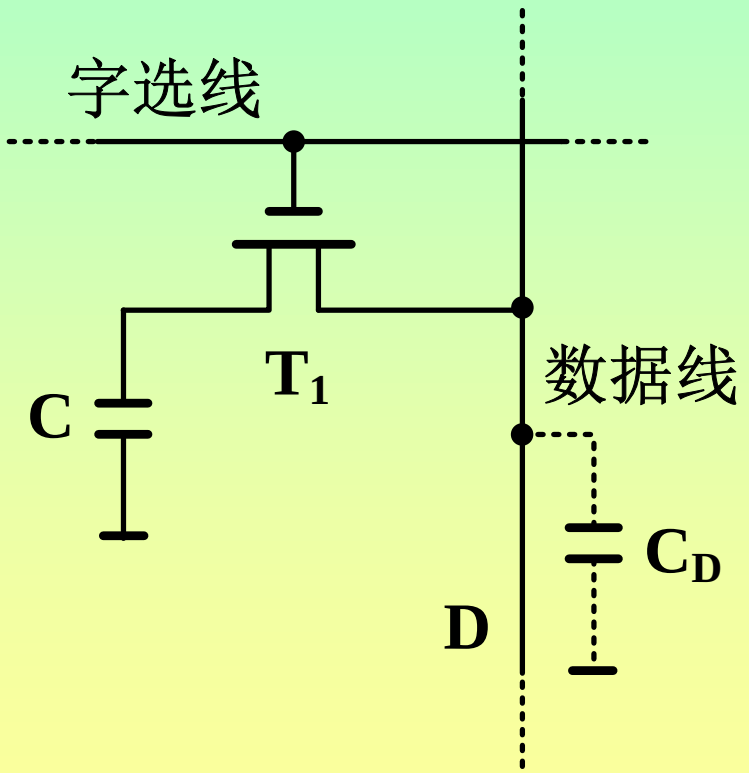
单管动态存储元

单管动态存储元是所有元电路中最简单的一种。虽然它的外围控制电路复杂，但由于在提高集成度上所具有的优势，使它成为目前大容量DRAM的首选存储元。

写操作时，字选线给出高电平，T_1导通，数据线上的数据被存入C。

读操作时，字选线同样给出高电平，C经过T_1向数据线上的分布电容C_D充电，使数据线获得存储元的信息。

由于$C_D \gg C$，数据线上读出的电压值很低，而且C上的电荷消耗殆尽，是一种破坏性读出。这种类型DRAM需要有灵敏的读出放大器，并且需要对存储元的信息进行恢复。



DRAM芯片2116

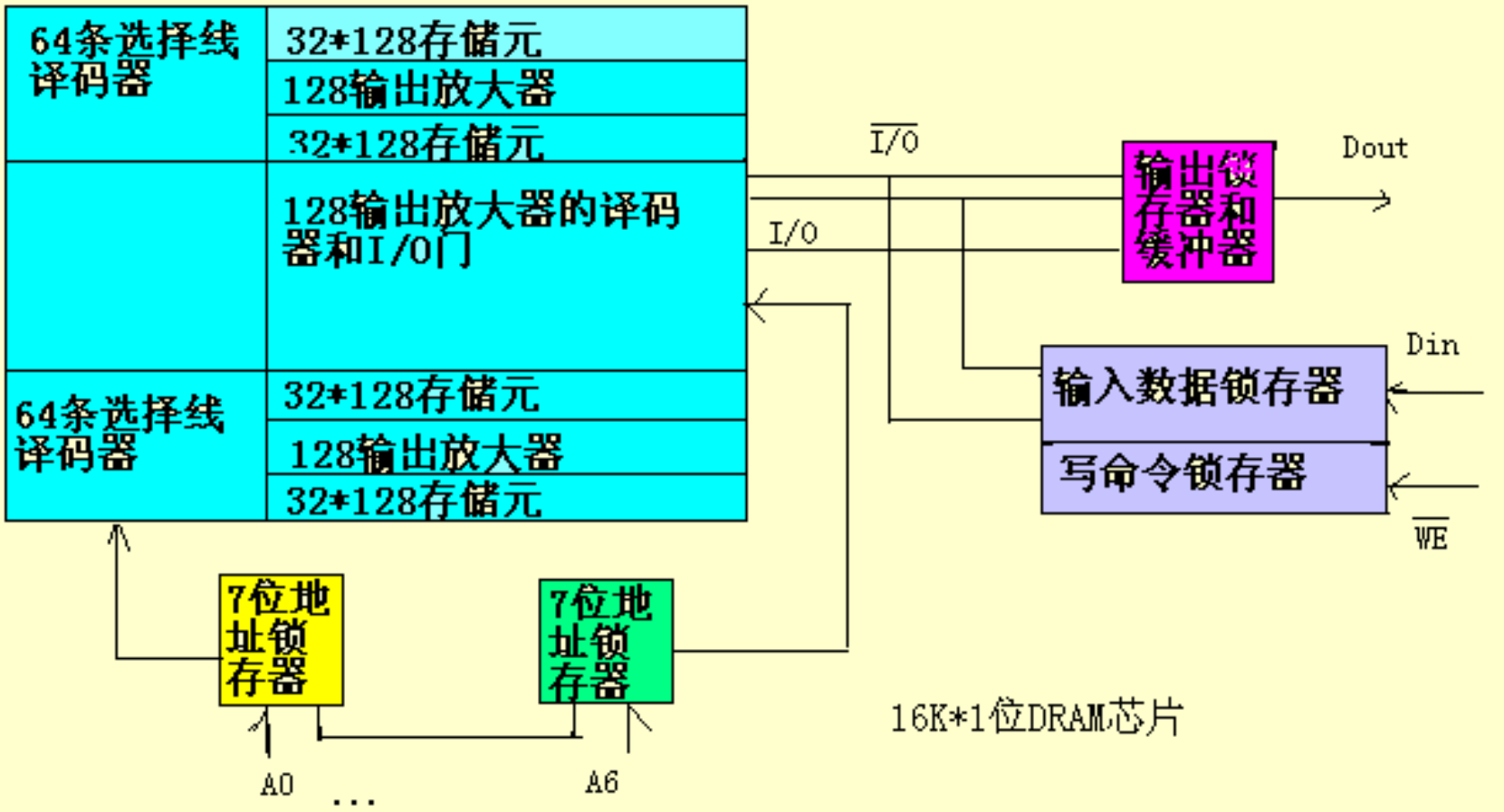
2116是16K×1位的DRAM，存储元排列成128×128矩阵。为减少引线数目，采用地址复用技术，地址引线只有7条。片内设有行地址锁存器和列地址锁存器，通过7条地址线接收CPU分时发送的地址：由行地址选通信号$\overline{\text{RAS}}$把先出现的7位行地址送至行地址锁存器，再由随后出现的列地址选通信号$\overline{\text{CAS}}$把7位列地址送至列地址锁存器。7条行地址线也用作刷新地址，实现按行刷新。

没有专门的片选线$\overline{\text{CE}}$。使用中可用行选信号$\overline{\text{RAS}}$和列选信号$\overline{\text{CAS}}$兼做片选。

数据线不是输入、输出共有的双向线，而是两根分设的输入、输出线D_{in}和D_{out}，且有各自的锁存器。

只设一根读写控制线$\overline{\text{WE}}$，为低电平时写入，为高电平时读出。

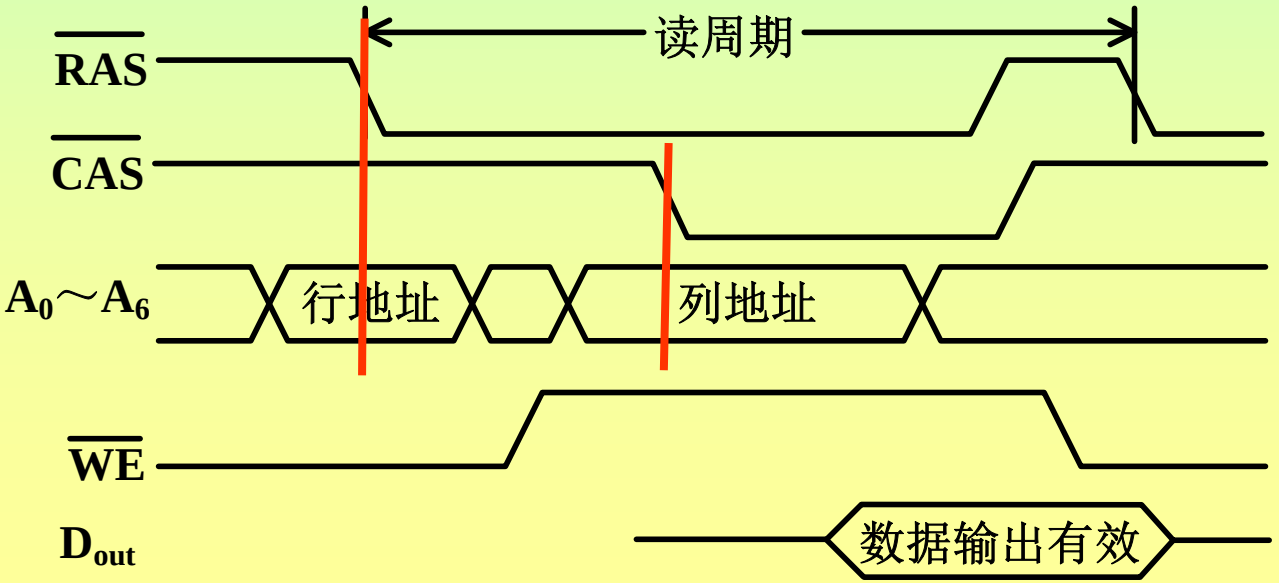
DRAM芯片2116


64条选择线
译码器
32*128存储元
128输出放大器
32*128存储元
128输出放大器的译码器和I/O门
64条选择线
译码器
32*128存储元
128输出放大器
32*128存储元
I/O
I/O
输出锁存器和缓冲器
Dout
Din
输入数据锁存器
写命令锁存器
WE
7位地址锁存器
7位地址锁存器
A0
...
A6
16K*1位DRAM芯片

读、写周期

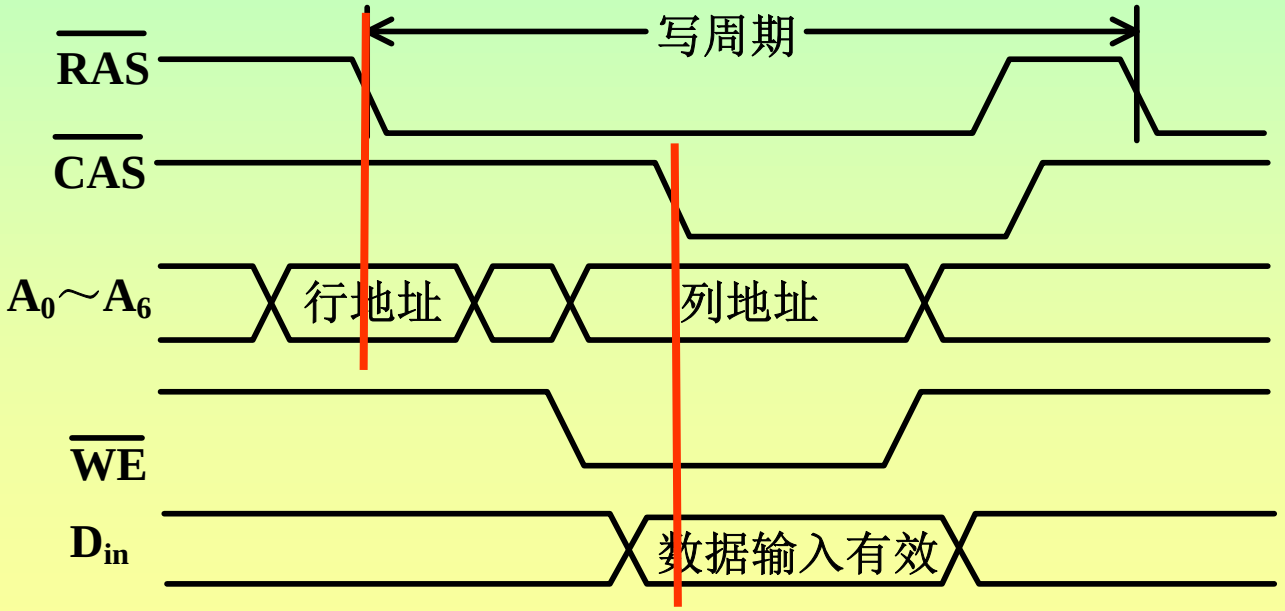
读周期：在读周期，A_0～A_6先收到7位行地址，然后行选信号$\overline{RAS}$＝0，通过脉冲的下降沿将行地址锁存。地址锁存后，可以撤消A_0～A_6的行地址。为提高读出速度，可以在产生列选信号之前产生读命令（$\overline{WE}$＝1）。然后A_0～A_6收到7位列地址，列选信号（$\overline{CAS}$＝0）通过脉冲的下降沿将列地址锁存。数据输出有效后，可以撤消行选信号、列选信号和读命令。

行地址信号在$\overline{RAS}$有效之前有效，且在$\overline{RAS}$有效后保持一段时间，将数据锁存，列地址也相同。



写周期：先在地址端准备好行地址，然后发行选信号（$\overline{\text{RAS}}=0$）和写命令（$\overline{\text{WE}}=0$）。行地址锁存后，可撤消芯片地址端的行地址。之后，在地址端准备好列地址，在数据输入端准备好写入数据，发列选信号。数据可靠写入后，可以撤消输入数据、行选信号、列选信号和写命令。

没有专门的片选线CE，用行选信号$\overline{\text{RAS}}$和列选信号$\overline{\text{CAS}}$兼做片选。



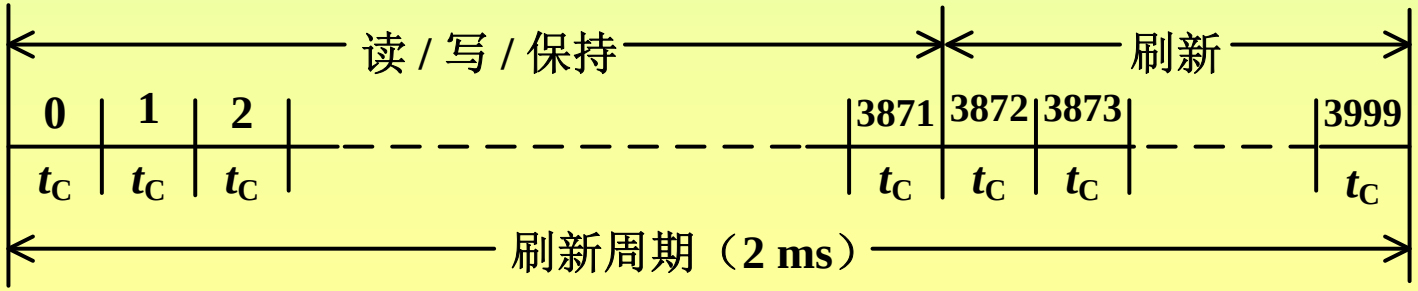
DRAM的刷新

DRAM的存储元利用栅极电容有无电荷表示信息。每隔一定时间必须对所有存储元的栅极电容补充电荷，称为刷新。刷新过程中只改变行选择地址，每次刷新一行。另外，刷新时不需要片选信号。

刷新有三种方式：集中式、分散式和异步式。以16K×1位的芯片为例，其存储元排列成128×128矩阵。

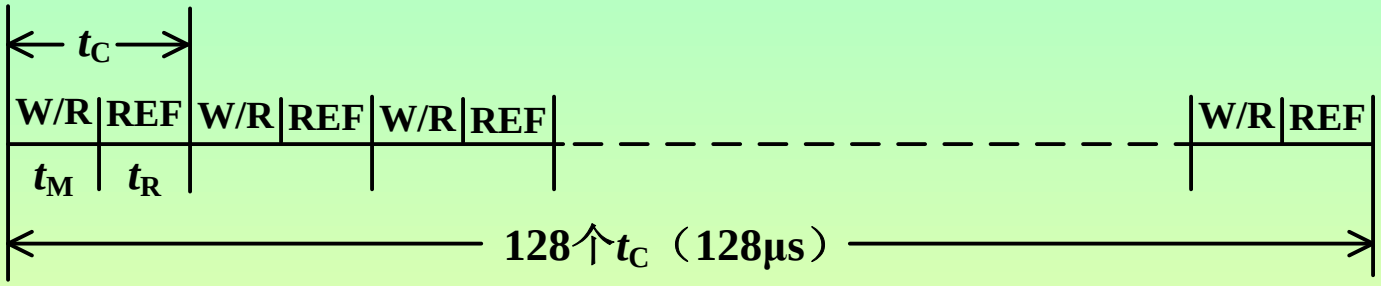
集中式：在每个刷新间隔内，前一段时间进行读写操作或保持，后一段时间集中进行刷新。设读写周期为t_C＝0.5 μs，刷新间隔为2 ms，相当于4000个读写周期。利用最后128个t_C进行刷新，前3872个t_C用来读写或保持。

在集中刷新的时间内不能进行存取访问，称为死时间。

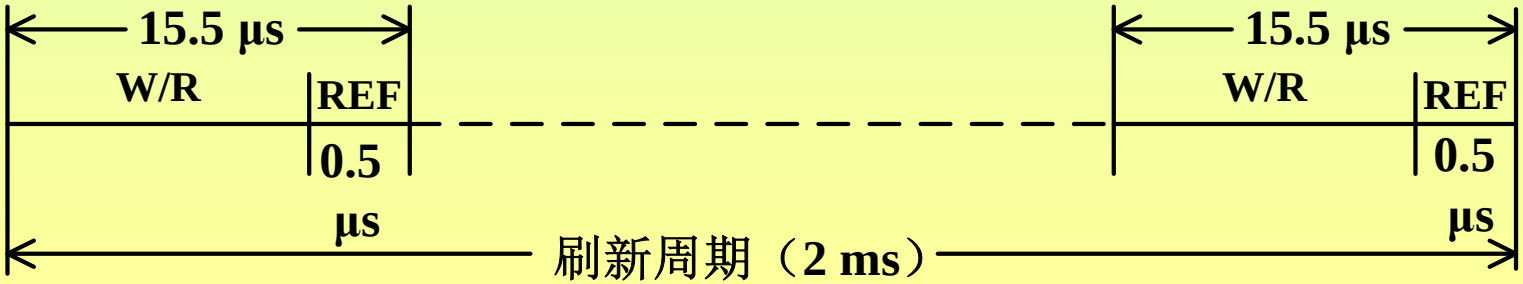


分散式：把每个存储系统周期t_C分成两半，前半段时间t_M用来读写或保持，后半段时间t_R用来刷新。每过128个系统周期，存储器就刷新一遍。如果读写周期为0.5 μs，存储器系统周期为1 μs，则刷新间隔为128 μs。

这种方式不存在死时间。但刷新过于频繁，影响系统速度。



异步式：是前两种方式的结合。把2 ms的刷新间隔分成128段，每段约15.5 μs。其中前15 μs用于读写或保持，后0.5 μs用于刷新。



DRAM的控制电路

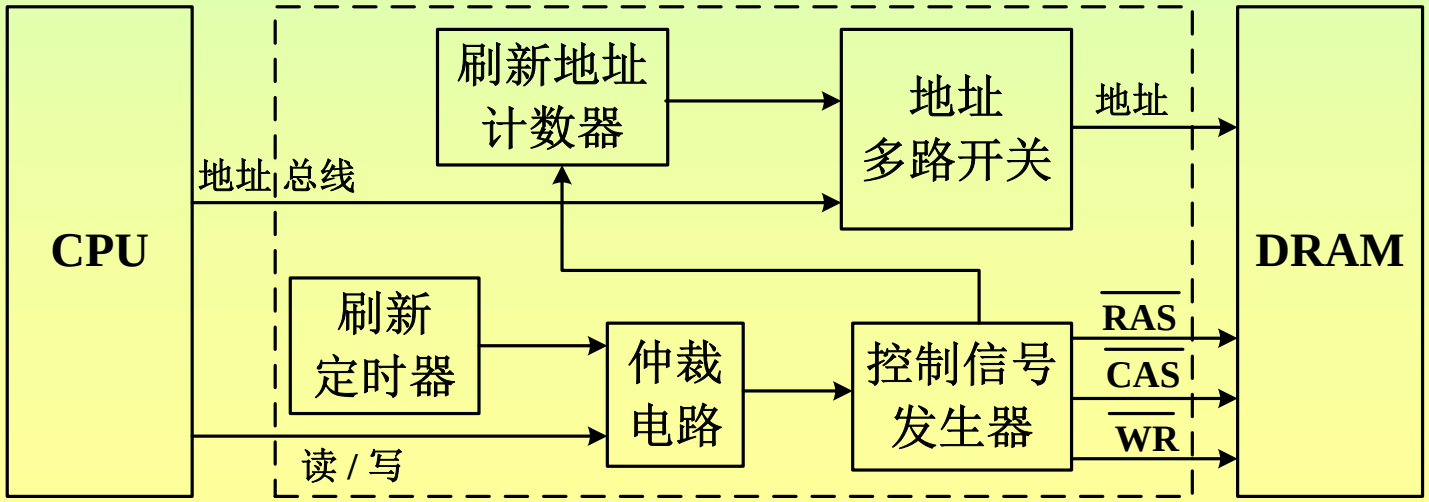
地址多路开关：向DRAM分时提供行地址和列地址，刷新时提供刷新地址。

刷新定时器：定时产生刷新请求。

刷新地址计数器：只用$\overline{\text{RAS}}$信号的刷新操作，需提供刷新地址计数器。

仲裁电路：来自CPU的访问存储器请求和来自刷新定时器的刷新请求同时出现，由仲裁电路对二者的优先权进行裁定。

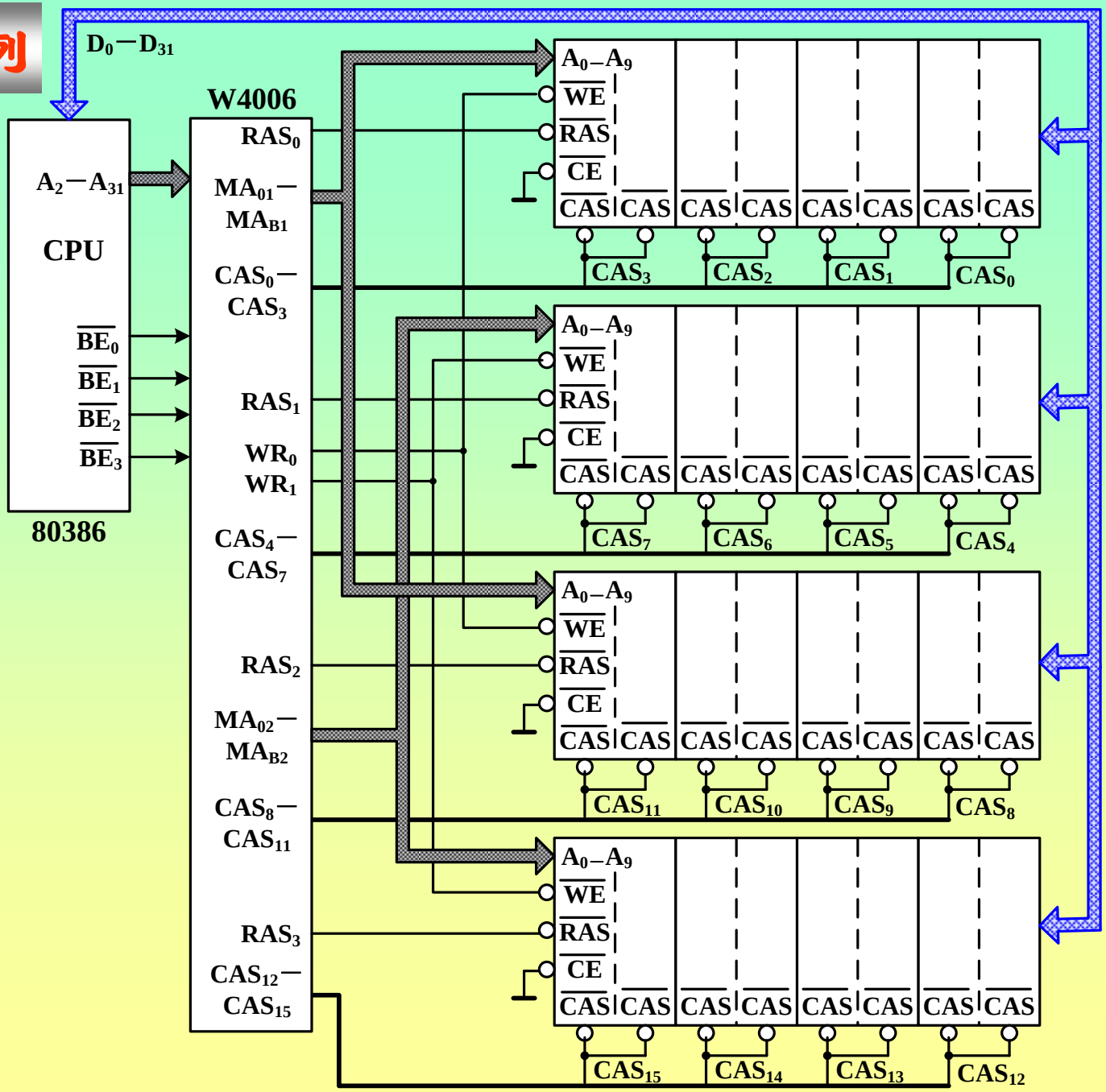
控制信号发生器：提供行地址选通信号$\overline{\text{RAS}}$、列地址选通信号$\overline{\text{CAS}}$和写命令$\overline{\text{WR}}$，对存储器进行存取或刷新。



主存组成实例

80386是32位处理器。地址线为A_2－A_{31}（没有A_0，A_1），用于确定一个4字节单元的地址；$\overline{BE_0}$－$\overline{BE_3}$指明访问该4字节中的那些字节。

控制两个存储器进行交叉访问。



上图利用8片1M×4位的芯片进行位扩展，得到1M×32位的存储模块，再利用4个这样的模块进行字扩展，得到4M×32位的主存储器。通过DRAM控制器W4006AF和CPU相连。

RAS_0－RAS_3：对存储器的行选通信号，分别与每个存储模块对应。

MA_{01}－MA_{B1}：对存储器的地址信号，对应于RAS_0和RAS_2指向的存储模块。

MA_{02}－MA_{B2}：对存储器的地址信号，对应于RAS_1和RAS_3指向的存储模块。

WR_0－WR_1：对存储器的写允许信号。

CAS_0－CAS_{15}：对存储器的列选通信号，由字节使能信号$\overline{BE_0}$－$\overline{BE_3}$产生：

$\overline{BE_0}$：产生CAS_0、CAS_4、CAS_8、CAS_{12}，选中D_7－D_0；

$\overline{BE_1}$：产生CAS_1、CAS_5、CAS_9、CAS_{13}，选中D_{15}－D_8；

$\overline{BE_2}$：产生CAS_2、CAS_6、CAS_{10}、CAS_{14}，选中D_{23}－D_{16}；

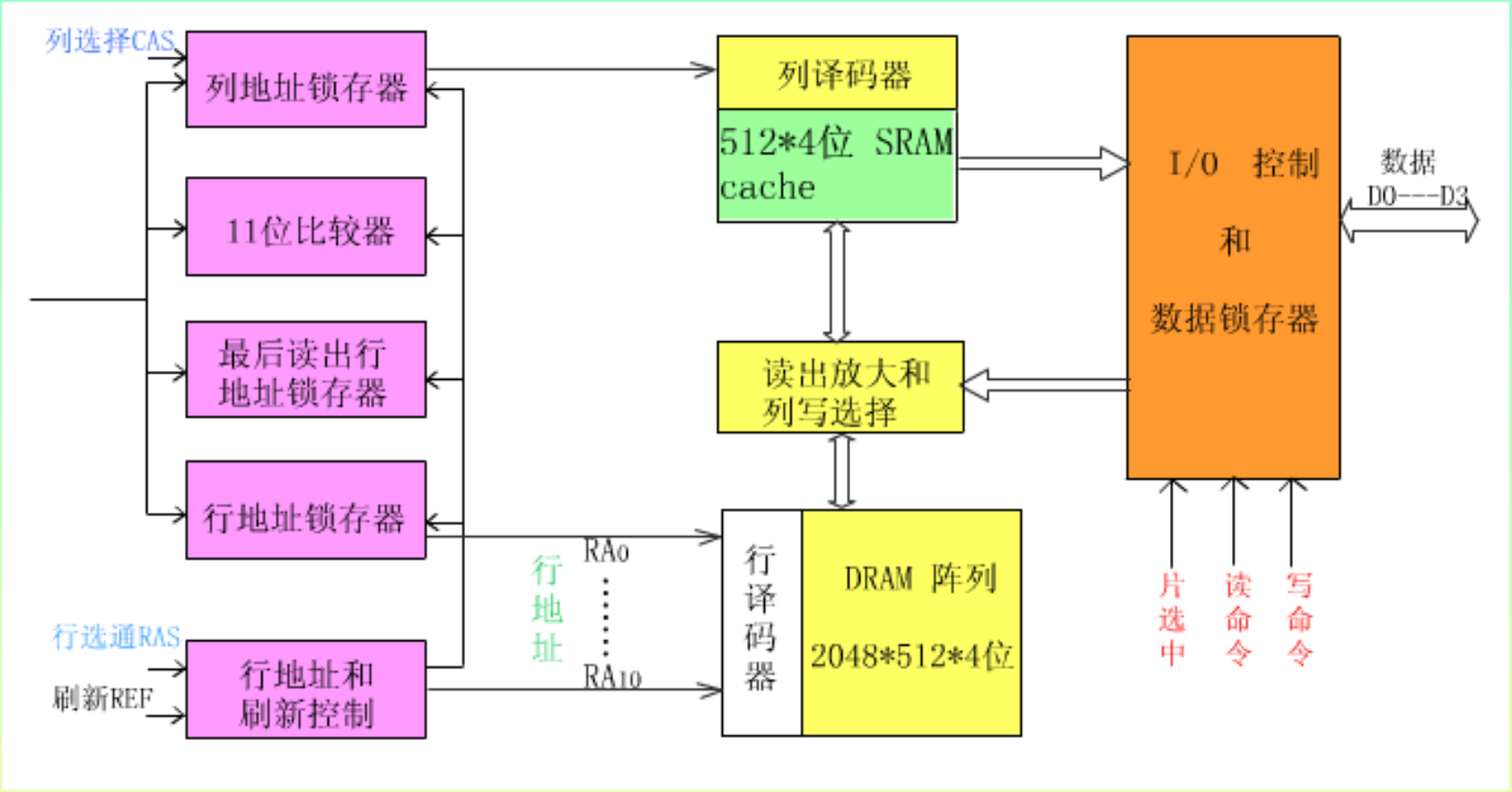
$\overline{BE_3}$：产生CAS_3、CAS_7、CAS_{11}、CAS_{15}，选中D_{31}－D_{24}；

新型DRAM芯片

CDRAM：增强型DRAM（cache DRAM）。它是在DRAM芯片内集成了一个SRAM作为高速缓存。

以1M×4位CDARM为例，其SRAM为512×4位。CDRAM的存储元排列成2048×512×4位。地址引脚11根，采用地址复用技术。首先在行选通信号作用下锁存11位行地址，从2048行中选择一行，将该行的512×4位数据送入SRAM暂存；然后在列选通信号作用下，锁存9位列地址。在读命令有效时，从512列中选择一列，将其4位数据送至数据线D_0－D_3。

下一次读取时，将输入行地址与上一次读取的行地址比较，若相同表示该行数据仍在SRAM中，只需根据列地址从SRAM中选择一列输出即可；若不同则更新SRAM和最后读出行地址锁存器。这种方式对成块传送非常有利。如果读取时地址高11位不变而低9位连续变化，就可以使SRAM中位组连续读出，这称为猝发式读取。



CDRAM的这种结构的两个优点：

●在SRAM读出期间可同时对DRAM阵列进行刷新。

●芯片内的数据输出路径与输入路径是分开的，允许在写操作完成的同时来启动同一行的读操作。

SDRAM：同步DRAM（Synchronous DRAM）。传统的DRAM与CPU之间采用异步方式交换数据。CPU发出地址后，经过一定的延迟，数据才读出或写入。这段时间内，CPU必须等待而不能做其它工作。而SDRAM和CPU共用系统时钟，同步工作。由于CPU知道SDRAM要多少个时钟周期才能响应，在SDRAM执行CPU请求时，CPU可离开并执行其它任务。例如系统时钟周期为10 ns，SDRAM读出延迟为60 ns，则CPU可以把地址放入锁存器，在存储器读操作期间去进行其它操作，6个时钟周期后再回来取从存储器读出的数据。如果DRAM是异步工作的，CPU只能等待60 ns。

DDR SDRAM：双倍数据速率SDRAM（Double Data Rate SDRAM）。其核心建立在SDRAM基础上，与SDRAM的区别是DDR SDRAM可以在时钟脉冲的上升沿和下降沿读出数据，不需要提高时钟频率就可以将SDRAM的速度加倍。

SRAM和DRAM的比较

（1）DRAM使用单管存储元，存储容量较大，约为SRAM的4倍。由于DRAM采用地址复用技术，引脚数目比SRAM少得多，封装尺寸也比较小。

（2）DRAM的价格比较便宜，约为SRAM的1/4。

（3）由于使用动态存储元，DRAM的功耗约为SRAM的1/6。

（4）DRAM的速度比SRAM低。

（5）DRAM需要刷新，外围电路复杂。

（6）SRAM一般用作容量不大的高速缓冲存储器，DRAM一般用作主存。

（7）它们的共同特点是断电后存储的信息消失，属于易失性存储器。

只读存储器

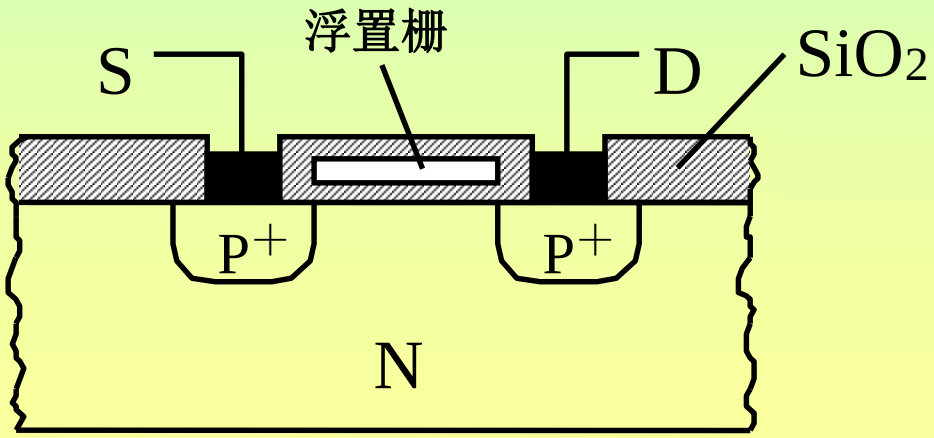
只读存储器（Read Only Memory，ROM）在正常工作时只能读出，不能写入。ROM是非易失性存储器，切断电源后，存储的信息也不会丢失。

掩模式只读存储器（Masked ROM，MROM）：它的内容由厂家按用户提出的要求在芯片的生产过程中直接写入的，写入后内容无法改变。MROM的优点是可靠性高，集成度高，形成批量之后价格便宜。缺点是灵活性差。

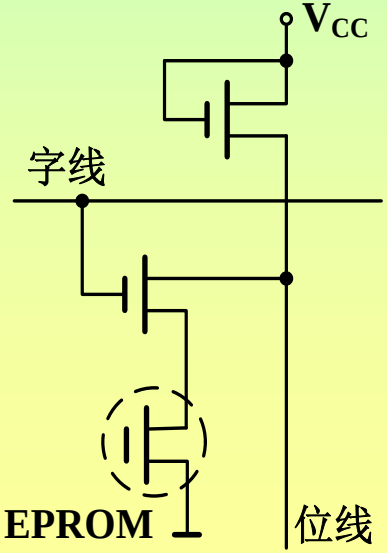
一次编程只读存储器（Programmable ROM，PROM）：PROM允许用户利用专门的编程器写入自己的程序，但一旦写入后，其内容无法改变。PROM产品出厂时，所有记忆单元均制成“0”（或“1”），用户根据需要可自行将其中某些记忆单元改为“1”（或“0”）。双极型PROM有两种结构，一种是熔丝烧断型，另一种是PN结击穿型，由于它们的写入都是不可逆的，所以只能进行一次性写入。

可擦除可编程只读存储器（Erasable PROM，EPROM）：用户可以对其内容多次改写。有光擦除（UVEPROM）和电擦除（EEPROM）两种。

光擦除EPROM的存储单元如下图。在出厂时所有储存单元的浮置栅都没有电荷，处于截止状态，相当于存“1”。写入“0”时，在D和S之间加高电压，使D、S间被击穿，有电子通过SiO_2绝缘层注入浮置栅。高压电源去除后，浮置栅上的电荷无处泄漏，使D、S间形成导电沟道，MOS管导通，相当于存“0”。如果用紫外光照射存储单元，浮置栅上的电荷形成光电流泄漏走，存储单元恢复“1”状态。



P沟道增强型浮栅雪崩注入MOS管



EPROM芯片2716

2716是2K×8位光擦除EPROM。有11条地址线A_0－A_{10}，7条用于行译码，4条用于列译码；有8条数据线D_0－D_7。工作电源V_{CC}为＋5V，编程电源V_{PP}在脱机编程时加＋25V，正常工作时接＋5V。

$\overline{CS}$为片选端，PD/PGM为功率下降/编程输入端。当PD/PGM接高电平时，芯片功耗下降75%，输出端呈高阻态。正常工作时，$\overline{CS}$和PD/PGM连在一起，没有选中的片子就工作在功耗下降方式下。

编程时，$\overline{CS}$接高电平，在PD/PGM端加TTL高电平脉冲，数据线上的输入就会写入指定的存储单元。

引脚 / 操作	PD/PGM	$\overline{CS}$	V_{PP}	V_{CC}	D_0～D_7
读	低	低	+ 5V	+ 5V	输出
未选中	无关	高	+ 5V	+ 5V	高阻
功率下降	高	无关	+ 5V	+ 5V	高阻
编程	脉冲	高	+ 25V	+ 5V	输入

闪速存储器

闪速存储器是一种高密度、非易失性半导体存储器，其存储单元由单支叠栅MOS管构成，可在联机状态下进行电擦除、改写。其特点有：

非易失性：SRAM和DRAM的信息在断电后丢失，需要磁盘作为后援存储器。而闪存的存储单元与E^2PROM类似，利用MOS管的浮置栅极是否存储电荷表示信息，具有非易失性。

廉价的高密度：相同存储容量的闪存与DRAM相比，位成本接近，但闪存节省了后援存储器（硬盘）的成本和空间。

可直接执行：闪存可与CPU直接相连，省去了从磁盘到RAM的加载步骤。

固态性能：闪存是一种低功耗、高密度且没有机电移动装置的半导体技术，适合于便携式计算机和移动存储设备。

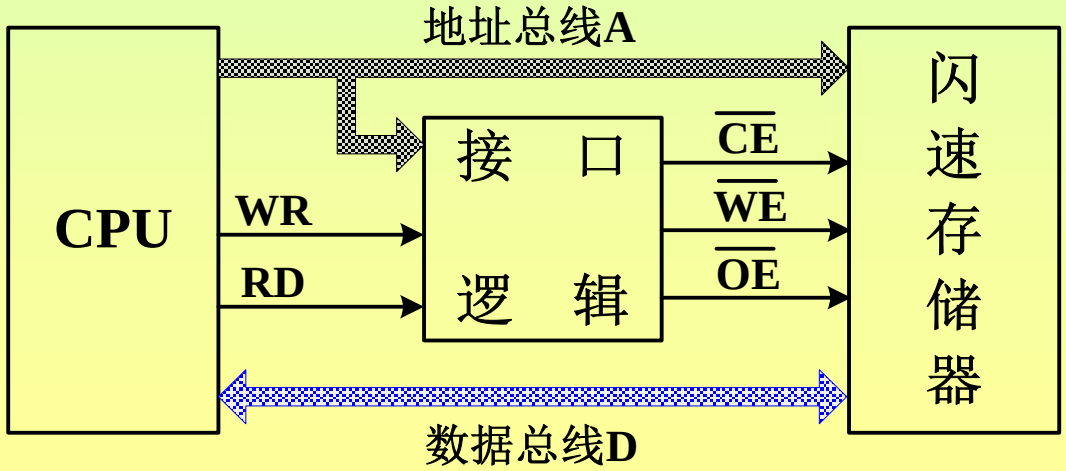
闪存的外部接口有三类：地址总线，数据总线，控制端有片选信号$\overline{CE}$，输出允许信号$\overline{OE}$和写命令信号$\overline{WE}$；另有对器件供电的V_{CC}（＋5V）和擦除/编程电源V_{PP}（＋12V）及地线。

P86表3.3 从各种存储器的性能比较中，FLASH具有明显优点。

28F256A的工作模式

操作 \ 引脚		V_{PP}	A_0	A_9	$\overline{CE}$	$\overline{OE}$	$\overline{WE}$	DQ_0～DQ_7
只读	读	V_{PPL}	A_0	A_9	0	0	1	数据输出
	输出禁止	V_{PPL}	×	×	0	1	1	高阻态
	等待	V_{PPL}	×	×	1	×	×	高阻态
读写	读	V_{PPH}	A_0	A_9	0	0	1	数据输出
	输出禁止	V_{PPH}	×	×	0	1	1	高阻态
	等待	V_{PPH}	×	×	1	×	×	高阻态
	写	V_{PPH}	A_0	A_9	0	1	0	数据输入

闪存与CPU的连接：



V_{PP}引脚接低电压（V_{PPL}）时，28F256A是一个只读存储器，可实现读、等待、输出禁止和读系统标识符等操作；V_{PP}接高电压（V_{PPH}）时，除实现上述功能外，还可实现存储器内容的变更，如擦除和编程。

读操作：片选信号$\overline{CE}$和输出允许信号$\overline{OE}$都有效（低电平）时实现数据输出。V_{PP}接高电压时，读操作可输出阵列数据、系统标识符代码，还可输出数据实现擦除/编程校验；V_{PP}接低电压时，只能输出阵列数据。

输出禁止操作：输出允许端$\overline{OE}$接高电平时，28F256A被禁止输出，数据输出引脚处于高阻态。

等待操作： 片选信号$\overline{CE}$处于高电平时，器件功耗大大降低，输出端呈高阻态。

标识符操作：当$\overline{CE}$和$\overline{OE}$均为低电平时，令A_9升至高电压，从地址0000H和0001H可读出厂家代码和器件代码。

写操作：V_{PP}接高电压，$\overline{CE}$＝0且$\overline{WE}$＝0时，实现写操作。

高速存储器

CPU和主存储器在速度上是不匹配的，限制了CPU性能的发挥；另一方面，在一个CPU周期内可能需要几个存储单元的数据，这种情况也成为限制高速计算机的主要问题。

解决CPU和主存之间速度差异的主要途径有：

① 在CPU内部设置更多的通用寄存器，存放运算的中间结果，减少访问主存的次数。

② 在CPU和主存之间插入高速缓冲存储器（Cache），缩短读出时间。

③ 采用并行操作的存储器，如双端口存储器、多模块交叉存储器和相联存储器。

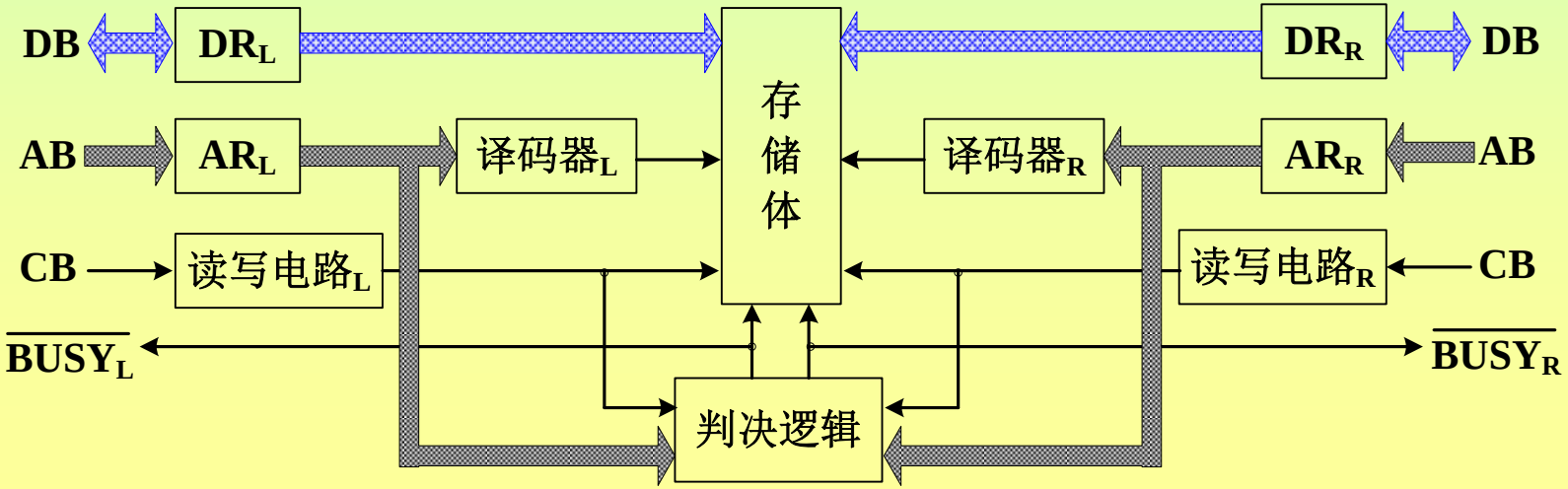
④ 主存储器采用更高速的技术来缩短读出时间，或加长存储器的字长。

双端口存储器

存储器一方面要不断接受CPU的访问，另一方面还要与外围设备交换信息。普通的RAM只有一套访问端口，在任一时刻只能接受一个部件的访问，而且一个存储周期内只能读/写一次，是串行的工作模式。

双端口RAM有两个相互独立的访问端口，即在一个存储体中配置左右两套地址寄存器（AR_L，AR_R）、地址译码器、数据缓冲寄存器（DR_L，DR_R）和读写控制电路。两个端口分别连接两套独立的总线。

IDT7133是2K×16位双端口RAM。两个端口具有各自的地址线（A_0～A_{10}）、数据线（I/O_0～I/O_{15}）和控制线（$R/\overline{W}$，$\overline{CE}$，$\overline{OE}$，$\overline{BUSY}$）。

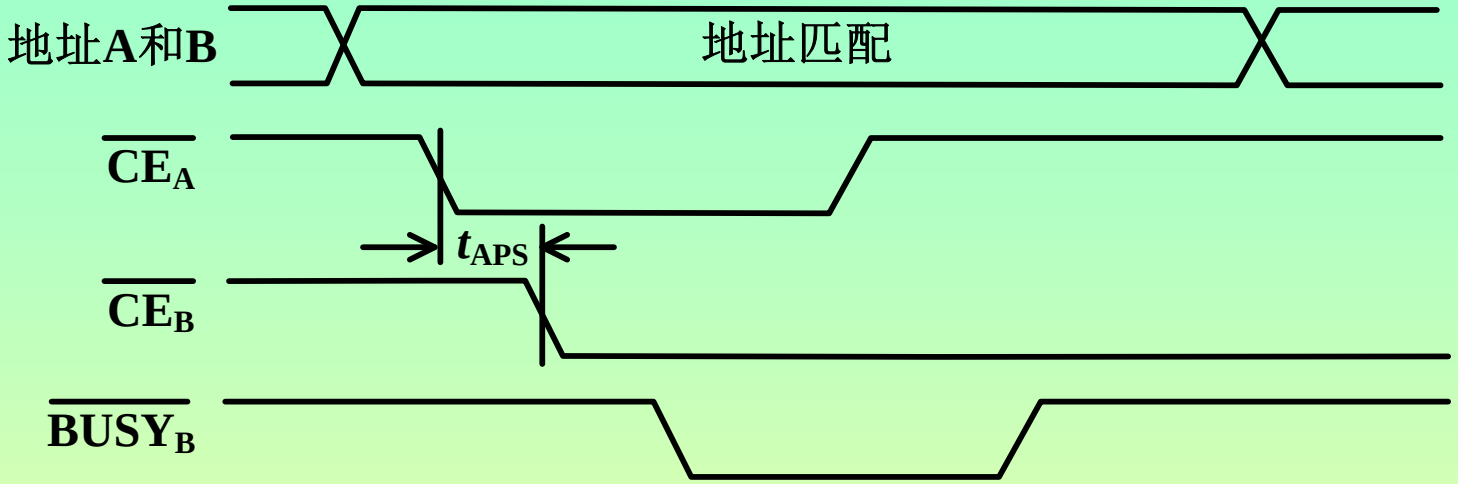


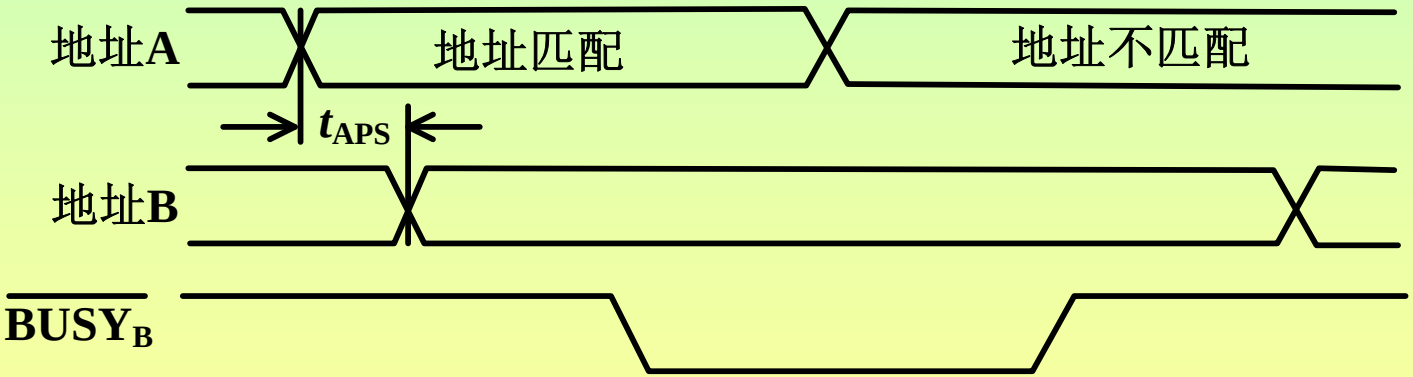
当两个端口地址不同时，在两个端口进行读写操作，不会发生冲突。任意端口被选中时，就可以对地址码指定的存储单元进行存取。每个端口有自己的片选控制$\overline{CE}$、输出驱动控制$\overline{OE}$和读写控制R/$\overline{W}$。读操作时，$\overline{CE}$为低电平，R/$\overline{W}$为高电平，$\overline{OE}$为低电平，输出驱动器打开，从存储矩阵读出的数据就出现在数据总线上；写操作时，$\overline{CE}$为低电平，R/$\overline{W}$为低电平，将数据缓冲寄存器的数据写入存储矩阵。

左端口或右端口						功　能
R/$\overline{W}_{LB}$	R/$\overline{W}_{UB}$	$\overline{CE}$	$\overline{OE}$	I/O_{0-7}	I/O_{8-15}	
×	×	1	×	Z	Z	功率下降模式，高阻抗输出
0	0	0	×	数据入	数据入	低位和高位字节数据写入存储器
0	1	0	0	数据入	数据出	低位字节写入，高位字节输出
1	0	0	0	数据出	数据入	低位字节输出，高位字节写入
0	1	0	1	数据入	Z	低位字节写入存储器
1	0	0	1	Z	数据入	高位字节写入存储器
1	1	0	0	数据出	数据出	存储器数据输出至低位和高位字节
1	1	0	1	Z	Z	高阻抗输出

两个端口对同一存储单元进行访问，会发生冲突。此时由判决逻辑对端口的优先权进行判决。在判决中失败的端口置$\overline{\text{BUSY}}$标志，暂时关闭该端口。

地址匹配发生在$\overline{\text{CE}}$有效之前，则利用$\overline{\text{CE}}$信号进行判决。



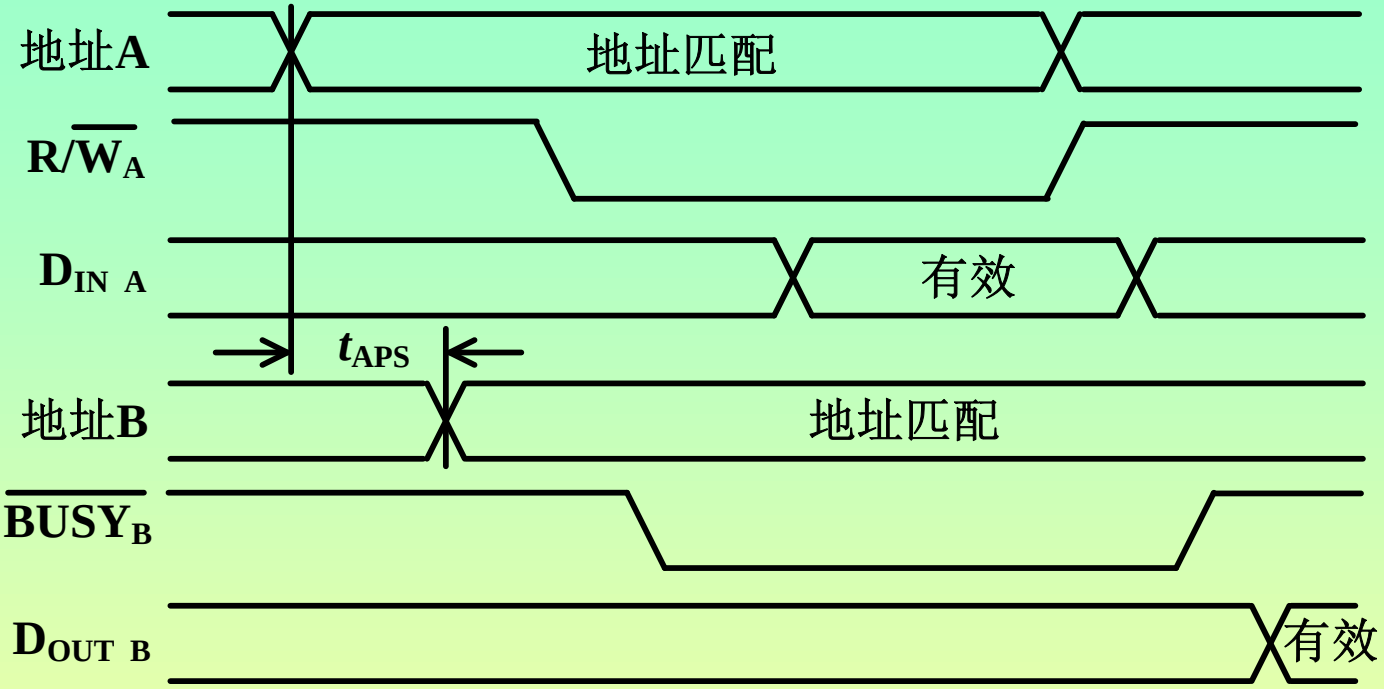


$\overline{\text{CE}}$在地址匹配之前变低，则利用地址信号进行判决。

❶ 端口A可以指左端口也可以指右端口，端口B是和A相反的端口；

❷ t_{APS}不能小于 5 ns，否则无法确定那个端口的$\overline{\text{BUSY}}$信号变成低电平，但不会出现两个端口的$\overline{\text{BUSY}}$同时变成低电平。

下图是端口A对某一存储单元进行写操作，端口B对同一存储单元进行读操作的时序。端口A的地址首先有效，因此在判决中获得优先权。



双端口RAM的应用：

① 实现CPU和DMA设备同时并行访问主存，避免相互干扰。

② 在多机系统中，用双端口或多端口存储器实现多机之间的信息交换。

③ 多级存储体系中的Cache采用双端口，一端面向CPU，一端面向主存，CPU访问Cache和Cache与主存交换数据可并行进行。

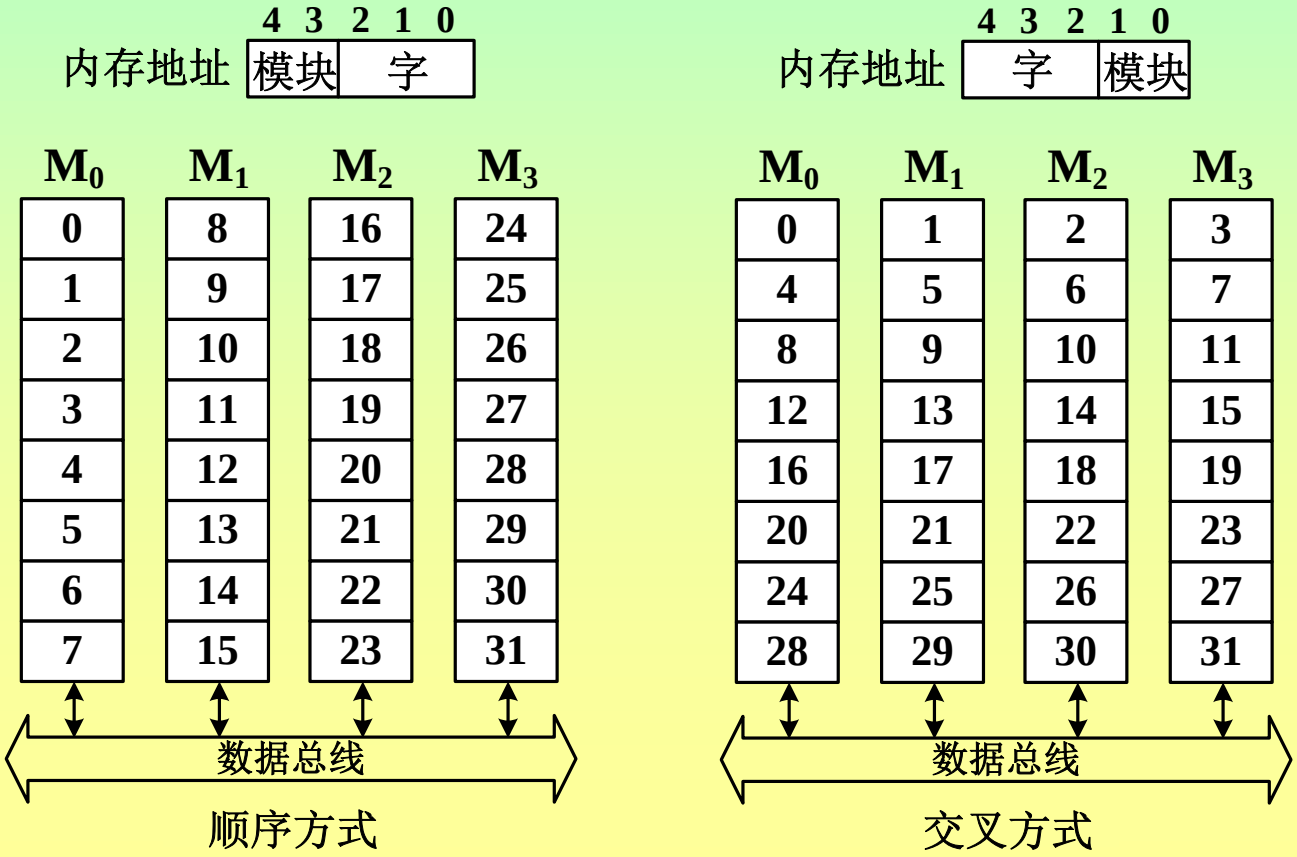
④ 内部总线为多总线结构的CPU中，用作通用寄存器为运算器的两个输入端并行提供操作数。

多模块交叉存储器

存储器地址在各模块中的安排方式有两种：

顺序方式：某一模块出现故障时，其他模块可以照常工作，通过增添模块来扩充存储器容量比较方便。各模块串行工作，存储器的带宽受到限制。

交叉方式：连续地址分布在相邻的模块内，同一个模块内的地址是不连续的。对连续字的成块传送可实现多模块流水式并行存取，提高存储器带宽。



每个模块有各自的读写控制电路、AR和DR。CPU同时访问四个模块，由存储器控制部件控制它们分时使用数据总线进行信息传递。

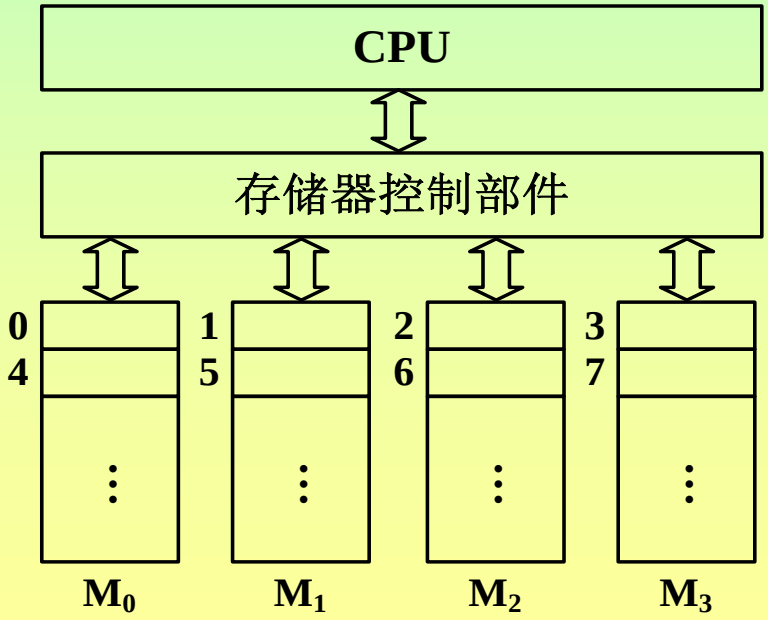
设模块存取一个字的存储周期为T，总线传送周期为τ，存储器的交叉模块数为m，为了实现流水线方式存取，应当满足

$T=m\tau$　　（$m=T/\tau$称为交叉存取度）

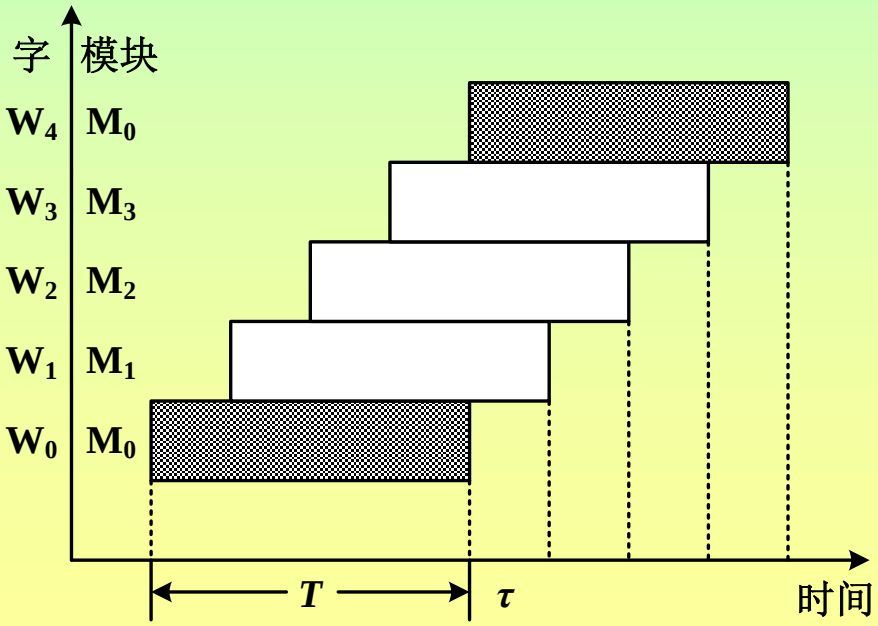
连续读取n个字所需的时间为

$t_1=T+(n-1)\tau$

而顺序方式存储器连续读取n个字所需时间为$t_2=nT$。交叉存储器的带宽大大提高。



四模块交叉存储器结构



四模块交叉存取的工作时序

例：设存储器容量为32字，字长64位，模块数m=4，分别用顺序方式和交叉方式进行组织。存储周期T=200ns，数据总线宽度为64位，总线传送周期 τ =50ns。连续读4个字，问顺序存储器和交叉存储器的带宽各是多少?

【解】

顺序存储器和交叉存储器连续读出m=4个字的信息总量都是

q=64位×4=256位

顺序存储器和交叉存储器连续读出4个字所需的时间分别是：

$t2=mT=4\times200ns=800ns=8\times10^{-7}s$;

$t1=T+(m-1)\tau=200ns+3\times50ns=350ns=3.5\times10^{-7}s$

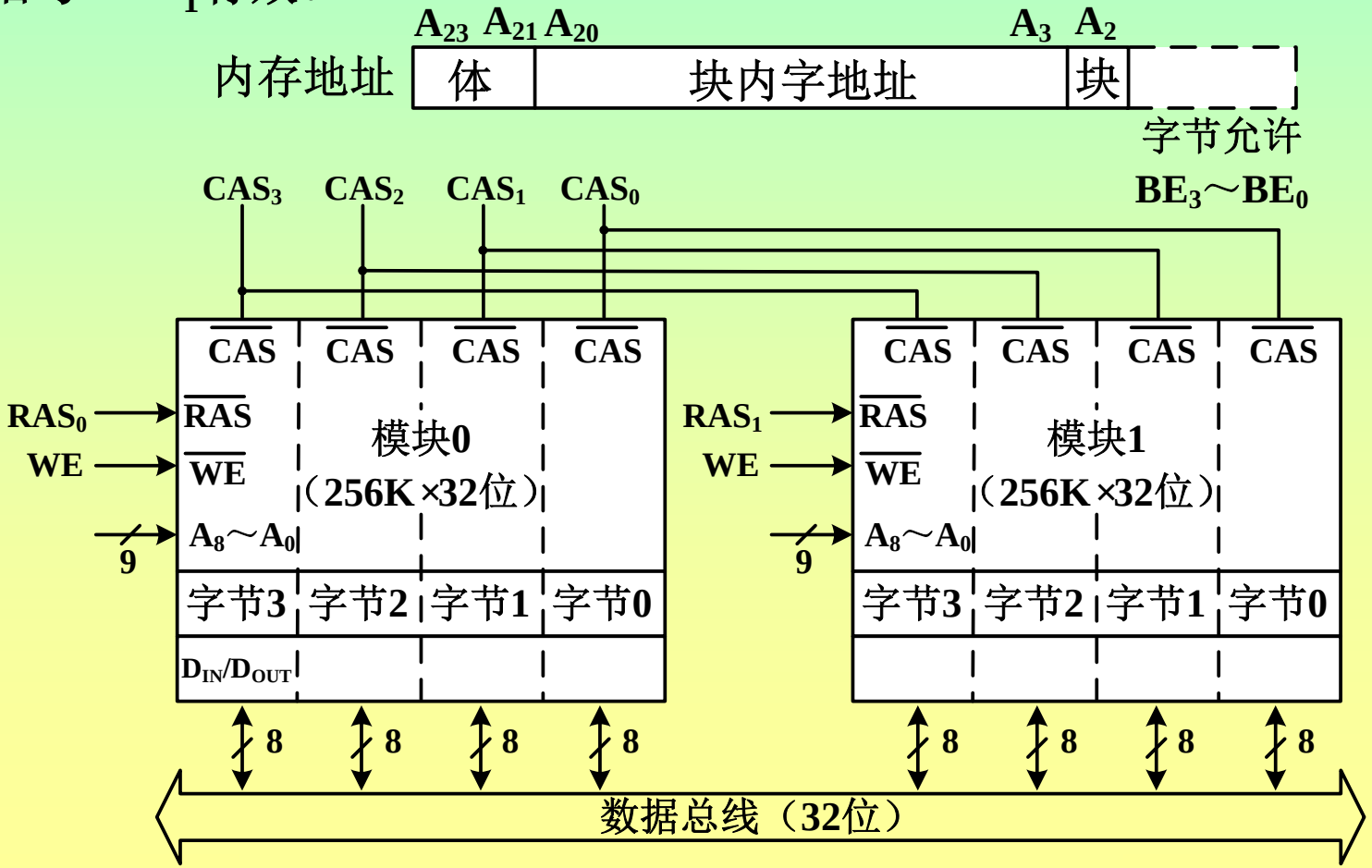
顺序存储器和交叉存储器的带宽分别是：

$W2=q/t2=256\div(8\times10^{-7})=32\times10^{7}$［位/s］;

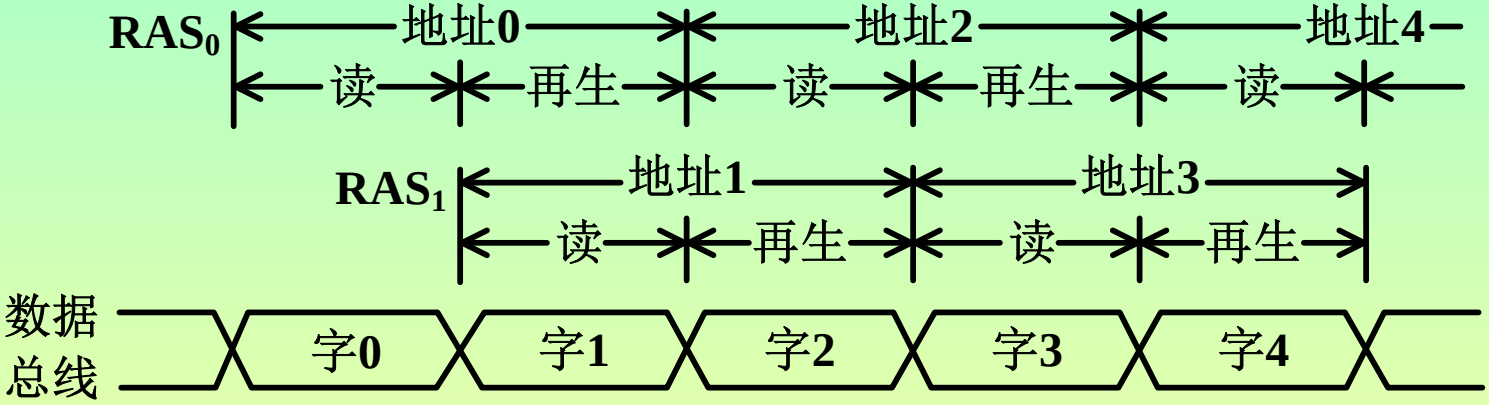
$W1=q/t1=256\div(3.5\times10^{-7})=73\times10^{7}$［位/s］

二模块交叉存储器实例

每个模块为256K×32位，两个模块构成一个2MB的存储体，全系统共8个存储体。地址总线24位，高3位选择存储体；A_{20}～A_3的18位地址用于模块内的256K个存储字选择，分成行、列地址送至芯片的9位地址引脚；A_2用于模块选择，A_2＝0时模块0的行选择信号RAS_0有效，A_2＝1时模块1的行选择信号RAS_1有效。



单管DRAM存储元的读出为破坏性读出，读取后要立即按照读出信息予以再生。若CPU先后两次读取的存储字使用同一RAS选通信号，CPU在接收到第一个存储字之后必须插入等待状态，直至前一存储字再生完毕才开始第二个存储字的读取。若采用$m=2$的交叉存取度的成块传送，两个连续地址字的读取之间不必插入等待状态（零等待存取）。

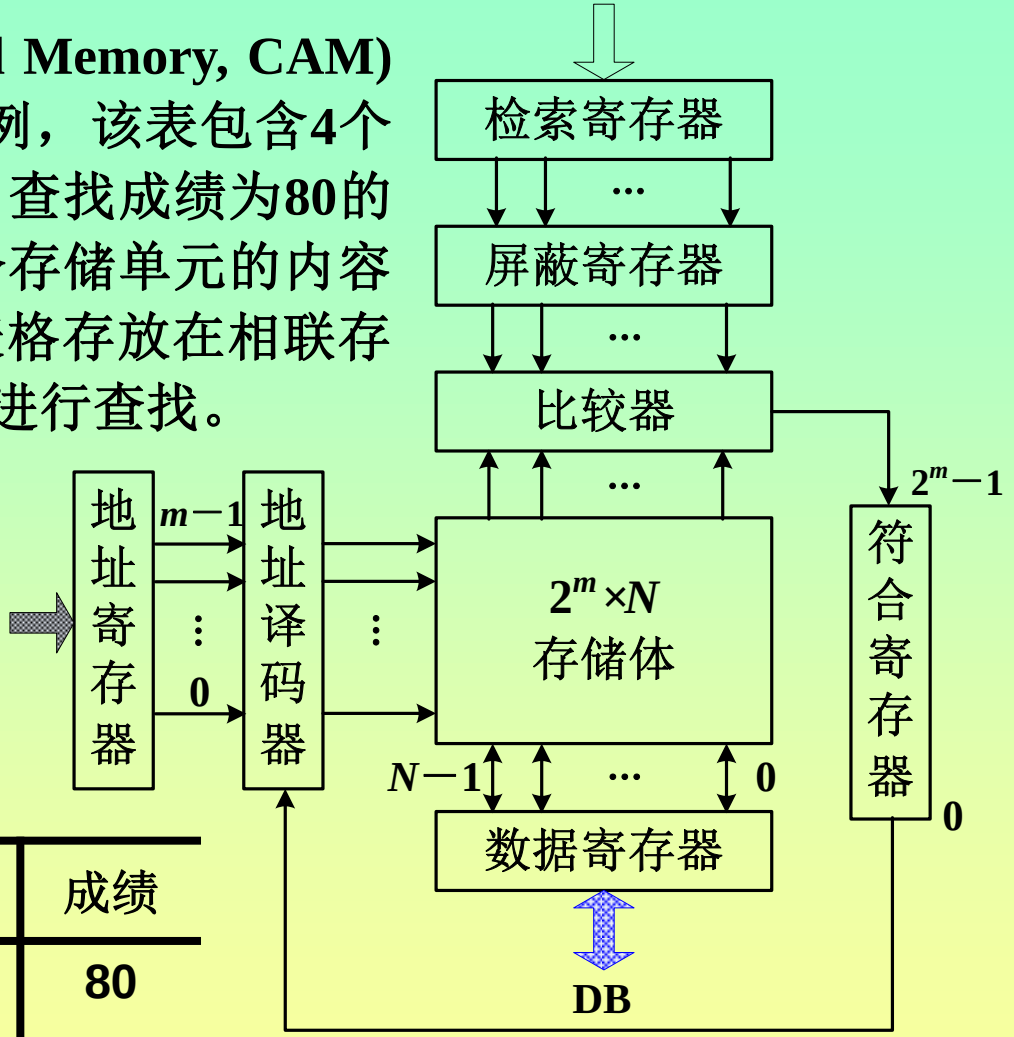


多模块交叉存储器适用于并行处理，在读取连续数据块时可获得高效率，是大、中型机内存的典型结构。多模块交叉存储器要求模块数为2的整数幂，任意模块出现故障都会影响整个地址空间。有些机器采用质数个模块，如我国的银河机$m=31$，其目的是减少存储器冲突。程序转移时的非顺序性会降低多模块交叉存储器的效率。如CDC6600 $m=32$，据测试，带宽为10字/T，还不到理想效率的1/3。

相联存储器

相联存储器 (Content Addressed Memory, CAM) 是按内容访问的存储器。以下表为例，该表包含4个记录，每个记录包含4个字段。为了查找成绩为80的人，可以从地址*n*开始逐一读出每个存储单元的内容，再对第4字段进行判断；也可将表格存放在相联存储器中，用第4字段作为关键字直接进行查找。

相联存储器的基本原理是把存储单元所存内容的一部分作为检索项(即关键字项)，去检索该存储器，并将存储器中与该检索项符合的存储单元内容进行读出或写入。



相联存储器结构

物理地址	学号	姓名	出生年月	成绩
n	001	AA	1981.1	80
n+1	002	BB	1981.2	85
n+2	003	CC	1981.3	90
n+3	004	DD	1981.4	95

相联存储器包括以下几部分：

① **存储体：**用高速半导体存储器构成。有2^m个存储单元，每个存储单元为N位。

② **检索寄存器：**用于存放检索内容，长度为N位。检索时，取其中若干位作为检索项。

③ **屏蔽寄存器：**用于存放屏蔽码，长度为N位。检索寄存器中检索项对应的屏蔽寄存器中的相应位为1，非检索项对应的位为0，被屏蔽掉。

④ **符合寄存器：**用于记录各存储单元的相应位与检索项的符合情况。符合寄存器的位数等于存储单元数，每一位对应一个存储单元。

⑤ **比较器：**将检索项和各存储单元相应位进行比较，比较结果送符合寄存器相应位。

⑥ **数据寄存器：**存放从存储体中读出的数据和向存储体写入的数据。

在计算机系统中，相联存储器主要用于存放Cache的行地址，或用于存放虚拟存储器中的分段表、页表和快表。

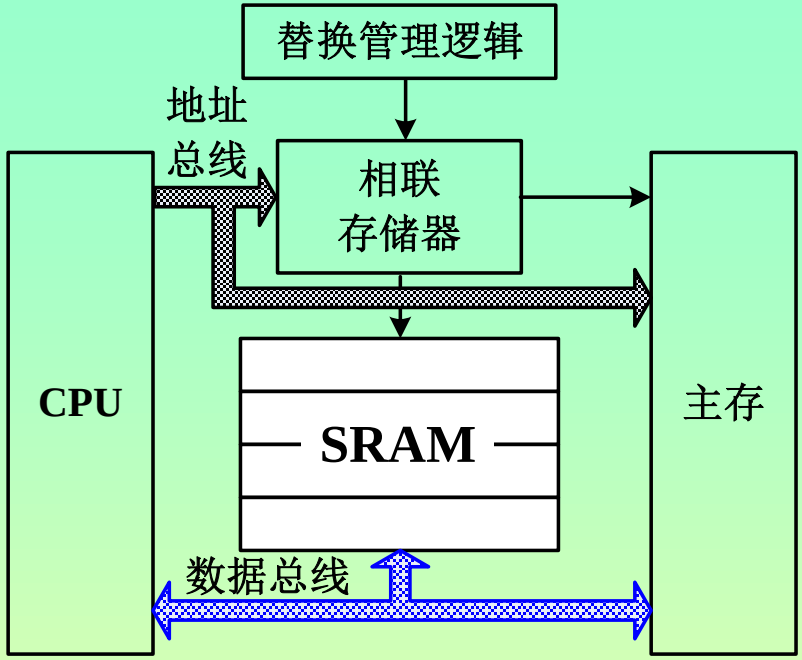
高速缓冲存储器Cache

Cache是为了解决CPU与主存速度不匹配而采取的技术，其有效性利用了程序的局部性原理。

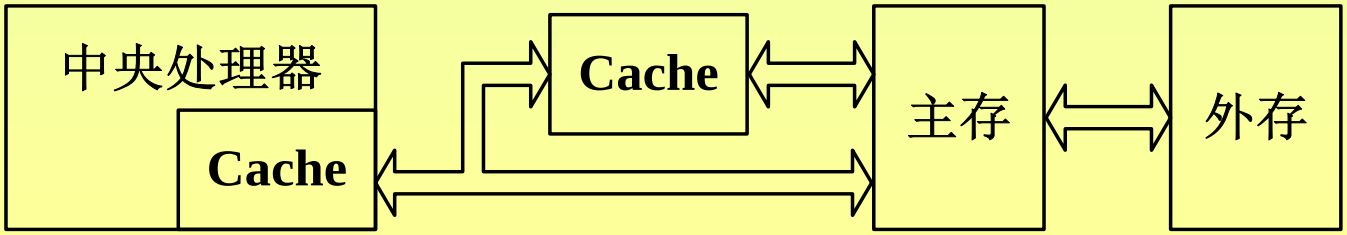
程序的局部性有两个方面的含义：时间局部性和空间局部性。时间局部性是指如果一个存储单元被访问，则可能该单元会很快被再次访问。这是因为程序存在着循环。空间局部性是指如果一个存储单元被访问，则该单元邻近的单元也可能很快被访问。这是因为程序中大部分指令是顺序存储、顺序执行的，数据也是以向量、数组、树、表等形式簇聚地存储在一起的。

高速缓冲技术就是利用程序的局部性原理，把程序中正在使用的部分存放在一个高速的容量较小的Cache中，使CPU的访存操作大多数针对Cache进行，从而使程序的执行速度大大提高。

主存中的数据以块为单位存放在Cache中，Cache中存放的主存块的地址保存在相联存储器里。CPU访问主存时，将地址送到相联存储器进行比较。若在Cache中则从Cache送往CPU，否则从主存送往CPU。



Cache采用SRAM器件，其存取速度接近CPU的速度。目前集成在CPU中的Cache有三级（L_1 L_2 L_3 Cache），早些年安装在主板上的Cache称为二级Cache（L_2 Cache）。



Cache的命中率

增加Cache的目的，就是在性能上使主存的平均读出时间尽可能接近Cache的读出时间。因此，cache的命中率应接近于1。

在一个程序执行期间，设N_c表示Cache完成存取的总次数，N_m表示主存完成存取的总次数，命中率h定义为：

$$h = \frac{N_c}{N_c + N_m}$$

若t_c表示命中时的Cache访问时间，t_m表示未命中时的主存访问时间，$1-h$表示未命中率，则Cache/主存系统的平均访问时间t_a为：

$$t_a = ht_c + (1-h)t_m$$

设$r = t_m/t_c$表示主存慢于Cache的倍率，e表示访问效率，则有：

$$e = \frac{t_c}{t_a} = \frac{t_c}{ht_c + (1-h)t_m} = \frac{1}{h + (1-h)r} = \frac{1}{1 + (1-h)(r-1)}$$

为提高访问效率，命中率h越接近1越好。命中率h与程序的行为、Cache的容量、组织方式、块的大小有关。

CPU执行一段程序时，cache完成存取的次数为1900次，主存完成存取的次数为100次，已知cache存取周期为50ns，主存存取周期为250ns，求cache/主存系统的效率和平均访问时间

解：

h=Nc/(Nc+Nm)=1900/(1900+100)=0.95

r=tm/tc=250ns/50ns=5

e=1/(r+(1-r)h)=1/(5+(1-5)×0.95)=83.3%

ta=tc/e=50ns/0.833=60ns

什么是平均访问时间？

平均访问时间＝访问的总时间/访问的总次数

访问的总时间＝**cache**访问时间＋主存访问的时间

＝**cache**访问的平均时间× **cache**访问的次数＋主存访问的平均时间×主存访问的次数

ta＝（250×100+50×1900）÷（1900+100 ）＝ 60

主存与Cache的地址映射

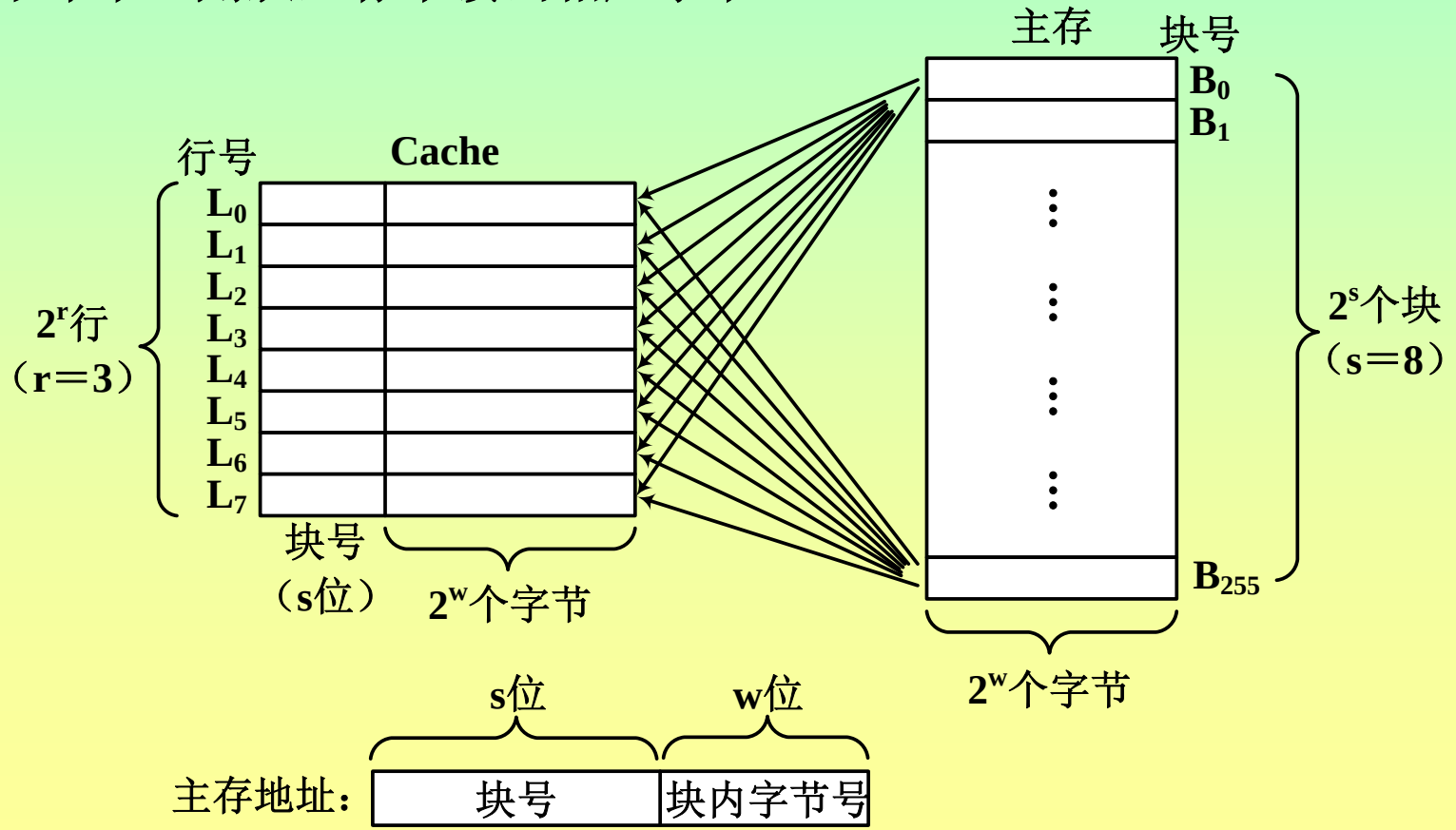
与主存相比，Cache的容量很小。它保存的只是主存内容的一个子集。为了把主存中的数据放到Cache中，必须应用某种方法把主存地址定位到Cache中，称为地址映射。当CPU访问存储器时，它给出的主存地址会自动变换为Cache地址。这个变换的过程是由硬件实现的，对程序员是透明的。

Cache的数据以“行”为单位，用L_i表示，其中$i=0, 1, \ldots, 2^r-1$，共2^r行；主存的数据以“块”为单元，用B_j表示，其中$j=0, 1, \ldots, 2^s-1$，共2^s块。Cache的行和主存的块是等长的，每行（块）由2^w个连续字节组成。

地址映射方式有全相联方式、直接方式和组相联方式三种。

全相联映射方式：主存的任意一块可以存放在Cache的任意一行中。Cache的每一行有s位的标记，用于保存该行所存放的主存块的块号。

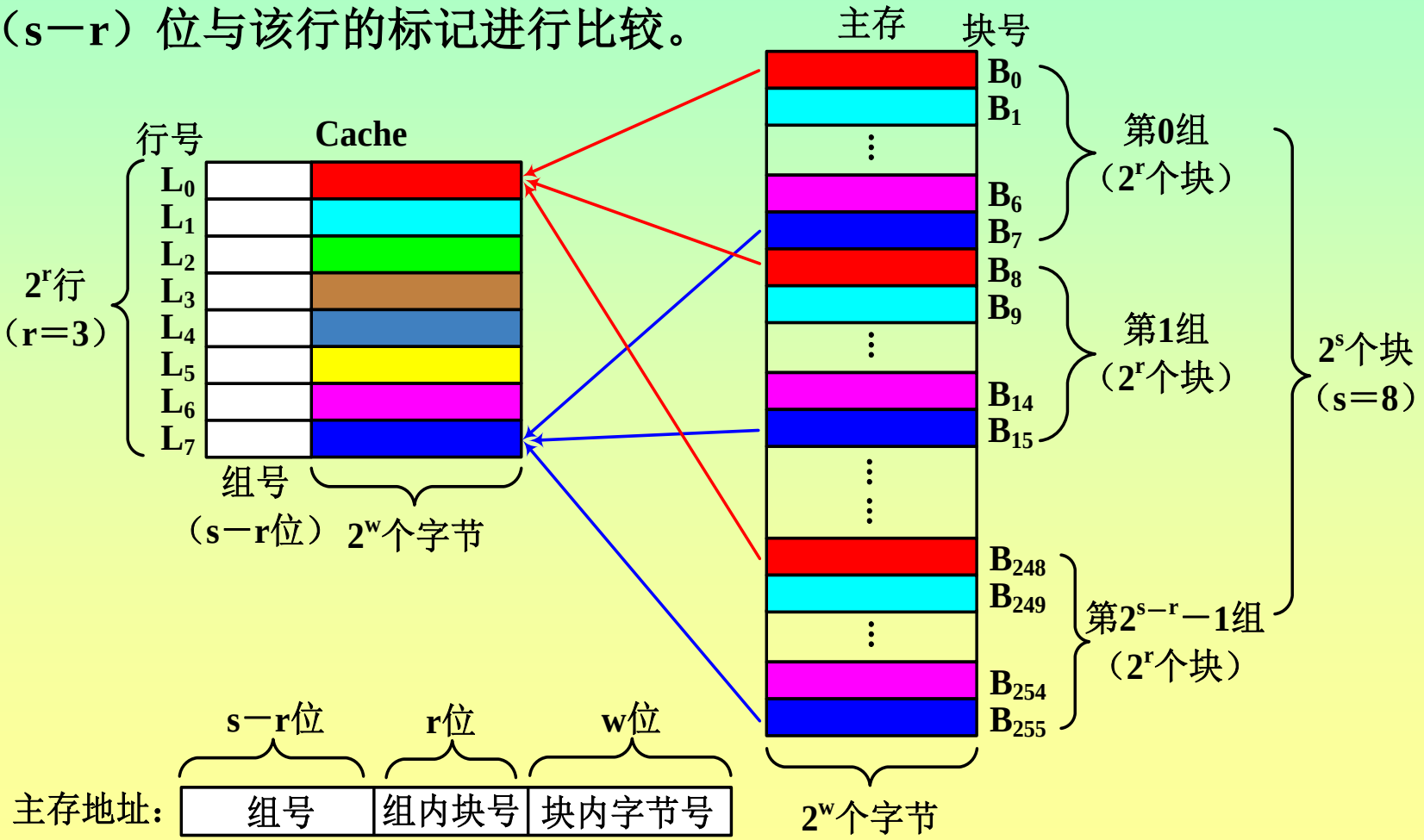
CPU给出的主存地址，高s位作为主存块的块号与Cache所有行的标记进行比较，若与某行的标记相同（命中），则利用低w位从该行读出相应字节；若不命中，则从主存中读出相应字节。



直接映射方式：每个主存块只能存放在Cache的一个特定行中。Cache行号i与主存块号j关系为：

$$i = j \bmod 2^r$$

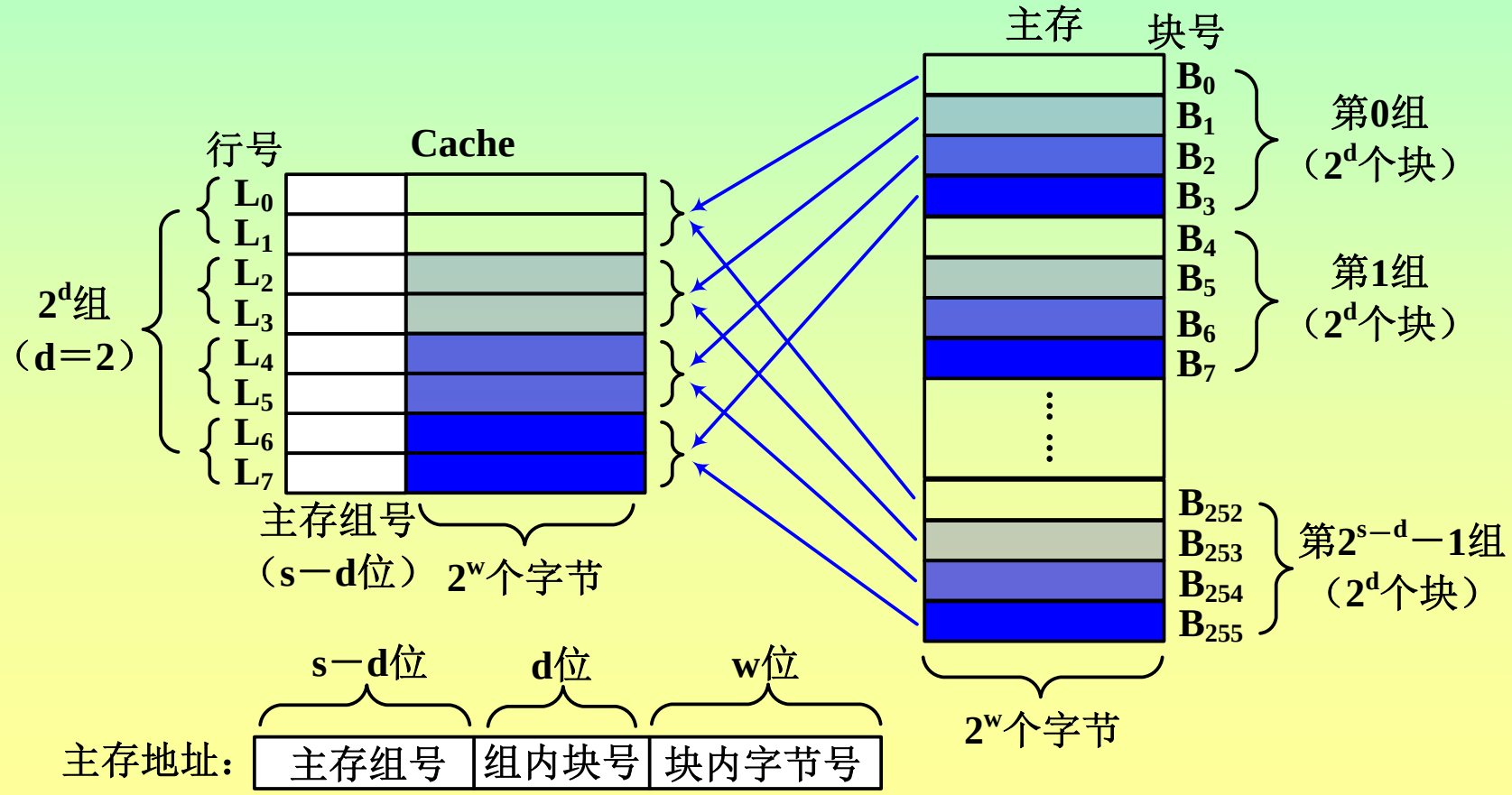
利用主存地址的中间r位确定可以保存该主存块的Cache行，再将主存地址的高（s－r）位与该行的标记进行比较。



组相联映射方式：将Cache划分为2^d组，每组2^{r-d}行；主存每2^d块作为一组，组中的每一块可以存放到Cache的2^{r-d}行中。

利用主存地址的中间d位确定可以保存该主存块的2^{r-d}个Cache行，再将主存地址的高（s－d）位与这2^{r-d}个Cache行的标记进行比较。

若Cache的每组有v行，称为v路组相联。



三种映射方式的比较：

全相联映射方式最为灵活，因为主存的任意一块可以保存在Cache的任意一行中，Cache利用率高。但Cache的标记位太长，而且需要将主存地址的高s位与所有Cache行的标记进行比较，硬件成本高，比较时间长。

直接映射方式的灵活性差，每个主存块只能存放在Cache的特定行中，如果相距为2^r的整数倍的两个主存块都需要存放在Cache中，就会发生冲突。这种方式的优点是硬件简单，只需要将主存地址的高（s－r）位与一个Cache行的标记进行比较。

组相联映射方式是前两种方式的折中。每个主存块可以存放在几个Cache行中，具有一定的灵活性，降低了冲突的可能。主存地址的高（s－d）位只需要和几个Cache行的标记进行比较，硬件成本相对较低。

cache命中率、效率及其相关计算

地址映射的方式及其特点

行号、标记（Tag）、页内地址、组号等位数的计算

地址映射、比较过程

替换策略

Cache应尽量保存最新的数据，必然产生替换。

对于直接映射方式，每个主存块只有一个Cache行可以存放，只需把该行的原主存块换出即可。

对于全相联和组相联映射方式，需要一定的替换策略。

最不经常使用算法（Least Frequently Use, LFU）：每行设置一个计数器，新行建立后从0开始计数。每命中一次，命中行的计数器加1。需要替换时，将计数值最小的行换出，其它行计数器清0。

近期最少使用算法（Least Recently Use, LRU）：每行设置一个计数器，新行建立时计数器为0。每命中一次，命中行计数器清0，其它行计数器加1。需要替换时，将计数值最大的行换出。

LRU保护了刚建立的行和刚刚命中的行，符合程序的局部性原理。

对于2路组相联的Cache，每个主存块只能存放在Cache中一个特定组的两行中，只需用一个二进制位作为标记。若一组中A行刚建立或刚命中，则将此位置1；B行刚建立或刚命中，则将此位置0。需要替换时，检查此二进制位的值，为0则替换A行，为1则替换B行。奔腾CPU的数据Cache是2路组相联结构，就是采用这种简化的LRU算法。

先进先出算法（First In First Out, FIFO）：按调入Cache的先后顺序决定淘汰的顺序，将最先进入Cache的块换出。这种方法要求记录每块进入Cache的先后次序。这种方法容易实现，其缺点是可能会把一些需要经常使用的程序块（如循环程序）也作为最早进入Cache的块替换掉。

随机替换：从Cache中随机选择一行换出。

Cache的写策略

Cache的内容只是主存内容的副本，应当和主存内容保持一致。CPU对Cache的写入改变了Cache的内容，应设法使Cache和主存保持一致。常用的写策略有两种：

写回法（Write Back）： CPU写Cache命中时，只修改Cache内容，而不立即写入主存。只有当此行被换出时才写回主存。这种方法使Cache对CPU－主存之间的读写操作都能起到高速缓冲作用。每个Cache需配置一个修改标志位。

全写法（Write Through）： 写Cache命中时，Cache和主存同时进行写操作。Cache在写操作时无高速缓冲功能。

现代计算机中，能够访问主存的设备不止一个。一个计算机可以有多个CPU，它们有各自的Cache和公用的主存；有些I/O设备也可以独立进行主存的读写。为了保证各个CPU的Cache及主存之间数据的一致性，需要更复杂的技术。

总线监听协议

每个Cache不断地监视地址总线，检测总线上其它CPU或I/O设备对存储器的写操作，并把对自身的写操作发布到总线上。监听协议有两种：写－无效协议和写－更新协议。写－无效协议是指某Cache数据被修改后使所有其它Cache中的相应数据副本失效；写－更新协议是指某Cache数据被修改后，广播修改后的数据以更新所有Cache中的相应数据副本。

奔腾CPU中采用的MESI协议属于写－无效协议。每个Cache行有4种状态，用两位进行标记，并根据监听结果和本地CPU的动作进行状态转换。

M (Modified): 修改态。此Cache行已被本地CPU修改，且没有写回主存。

E (Exclusive): 专有态。此Cache行和主存内容相同，其它Cache中没有副本。

S (Shared): 共享态。此Cache行有效，其它Cache中可能有该行的有效副本。

I (Invalid): 无效态。此Cache行无效。

虚拟存储器

虚拟存储器是指主存－辅存层次在操作系统的调度下，以透明的方式给用户提供一个比实际主存空间大得多的程序空间。

程序员编程时所用的地址称为逻辑地址或虚拟地址（虚地址），主存的实际地址称为物理地址（实地址）。程序每次访问主存时，要进行虚、实地址的转换，通常用软、硬件结合的办法实现。

主存－辅存层次和Cache－主存层次有很多相似之处，它们都包括一个容量较小速度较高的存储器和一个容量较大速度较低的存储器，都采用地址变换及映射方法和替换策略，都基于程序局部性原理。它们遵循的原则是：

把程序中最近常用的部分驻留在高速存储器中，一旦这部分变得不常用了，就把它们送回低速存储器；这种换入换出对应用程序是透明的；力图使存储系统的性能接近高速存储器，容量和价格接近低速存储器。

Cache－主存层次主要是为了提高存储系统的速度，而主存－辅存层次主要是为了扩大程序可用的地址空间。

虚拟存储器的管理方式有三种：段式、页式和段页式。

程序可以按照逻辑结构划分为多个相对独立的段。将主存按段分配的管理方式称为段式管理。段的大小取决于程序的逻辑结构，可长可短。用段表来指明各段在主存中的位置，段表本身也是主存中一个可再定位段。段的逻辑独立性使它易于管理、修改和保护。由于段长不固定，给主存空间的分配带来麻烦，容易形成碎片。

页式管理是把主存空间划分成长度固定的页。优点是页的起点和终点地址固定，给造页表带来方便，对主存空间的浪费小。由于页不是逻辑上独立的实体，处理、保护和共享都不如段式管理方便。

段页式管理是将分段和分页结合起来，程序按模块分段，段内再分页。用段表和页表进行两级定位管理。

页式管理

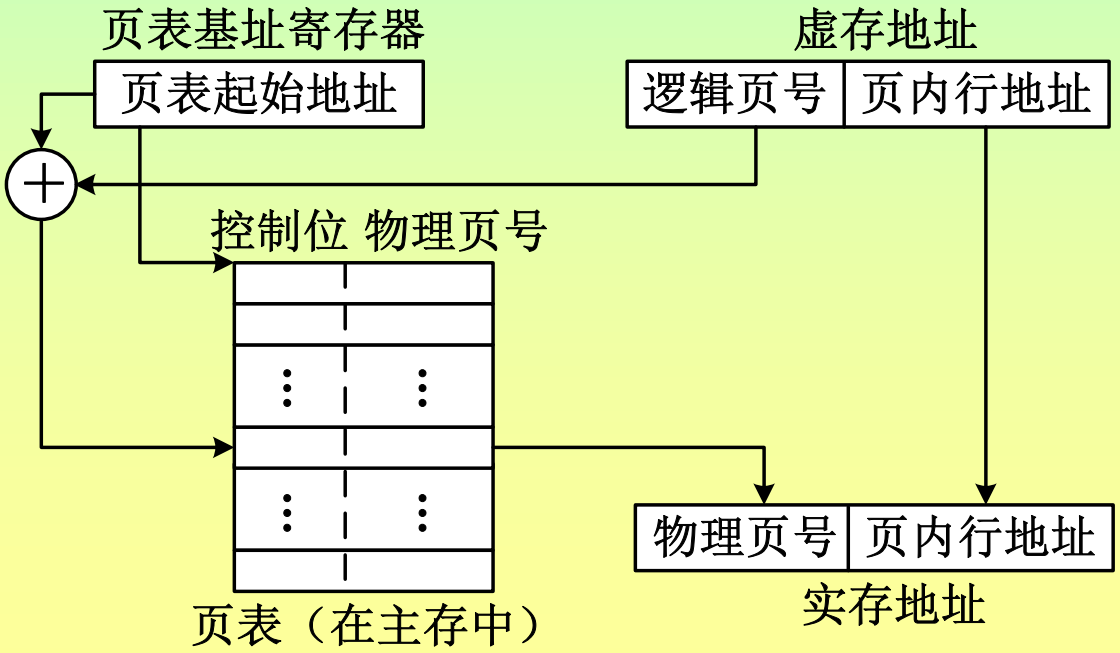
虚拟空间分成逻辑页，主存空间也分成同样大小的物理页。

虚存地址分为两个字段：高字段为逻辑页号，低字段为页内行地址。

实存地址也分两个字段：高字段为物理页号，低字段为页内行地址。

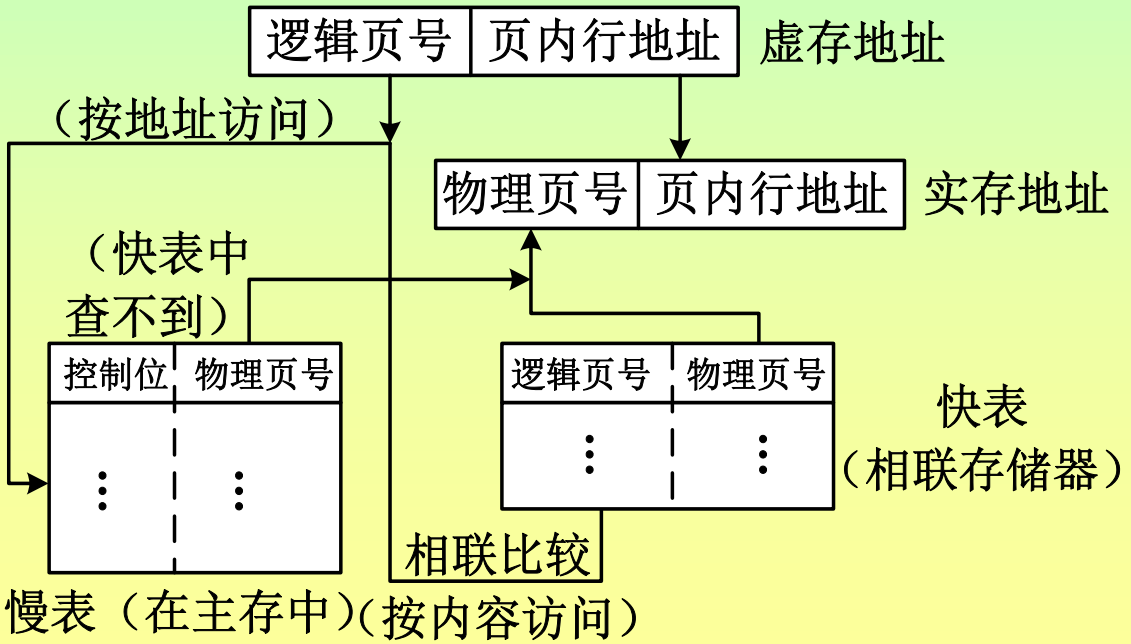
两者的页内行地址是相同的。虚实地址的转换通过页表实现。页表中每一个逻辑页号有一个表项，表项内容包含该逻辑页的物理页号，用它作为实存地址的高字段，与虚存地址的页内行地址字段相拼接，产生完整的实主存地址。

页表的表项中通常还有装入位、修改位、保护位等控制位。装入位表明该逻辑页是否调入主存，访问没有装入的页会启动输入输出系统，从辅存装入该页。修改位指出主存中页面是否被修改。

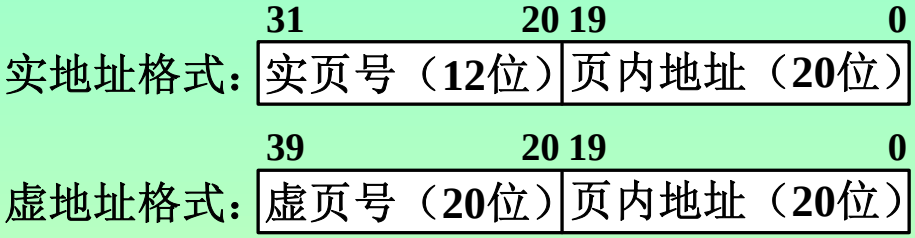


访问存储器时，首先要查找页表，根据查表获得实存地址再次访问主存。因此需要两次访问主存。可以把页表中最活跃的部分存放在高速存储器中组成快表。为了提高查找速度，可以引入硬件支持，如采用相联存储器。

快表是慢表中部分内容的副本。查表时，用逻辑页号同时查找快表和慢表。快表命中则使慢表的查找作废。快表查不到则等待慢表的查找结果，并将逻辑页号和物理页号送入快表。

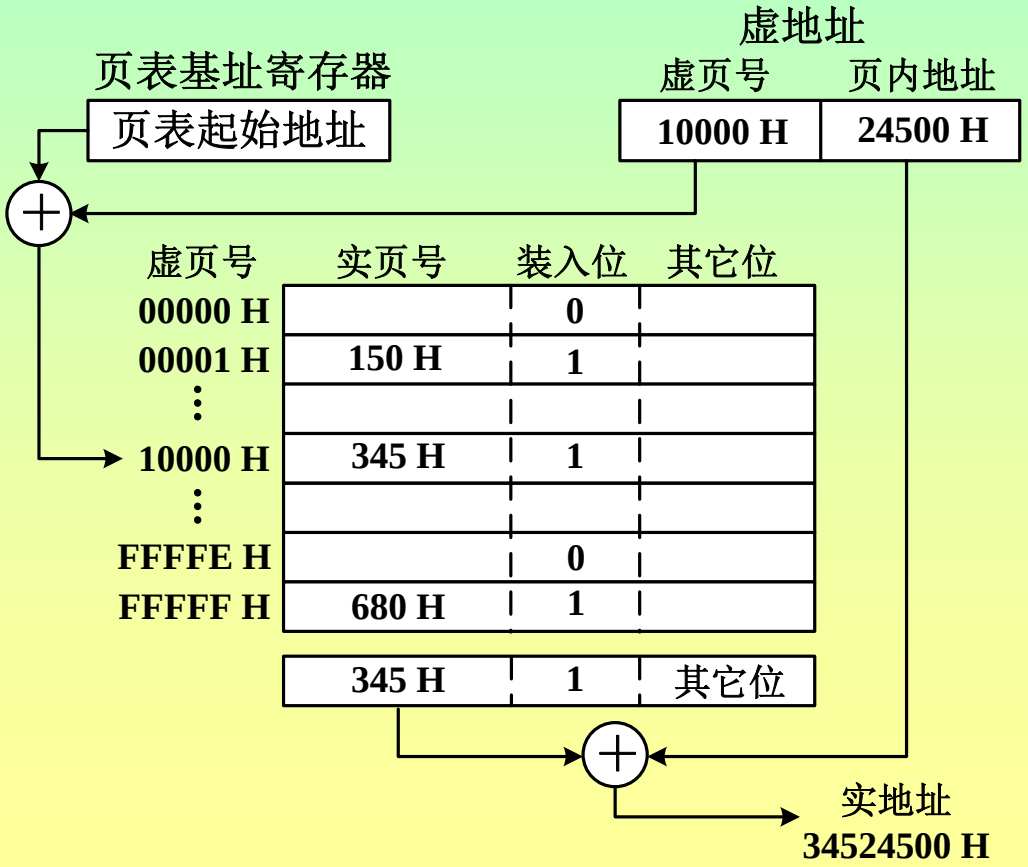


例：某计算机的虚拟存储系统有40位虚地址，32位实地址。页的大小为1MB，有装入位、修改位、保护位和使用位四个控制位。所有虚页都在使用中。



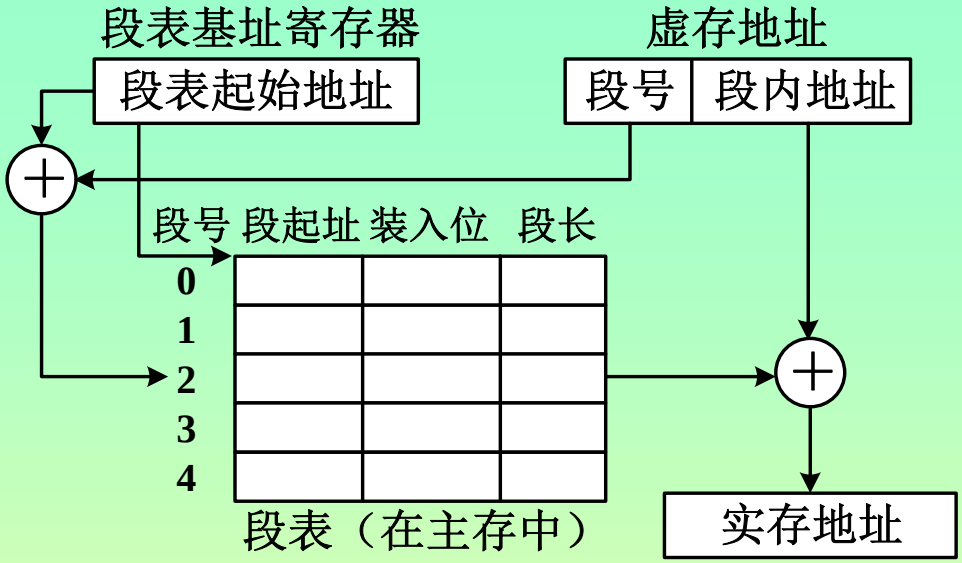
每个页表项的长度为实页号位数（12位）＋控制位（4位）＝16位。

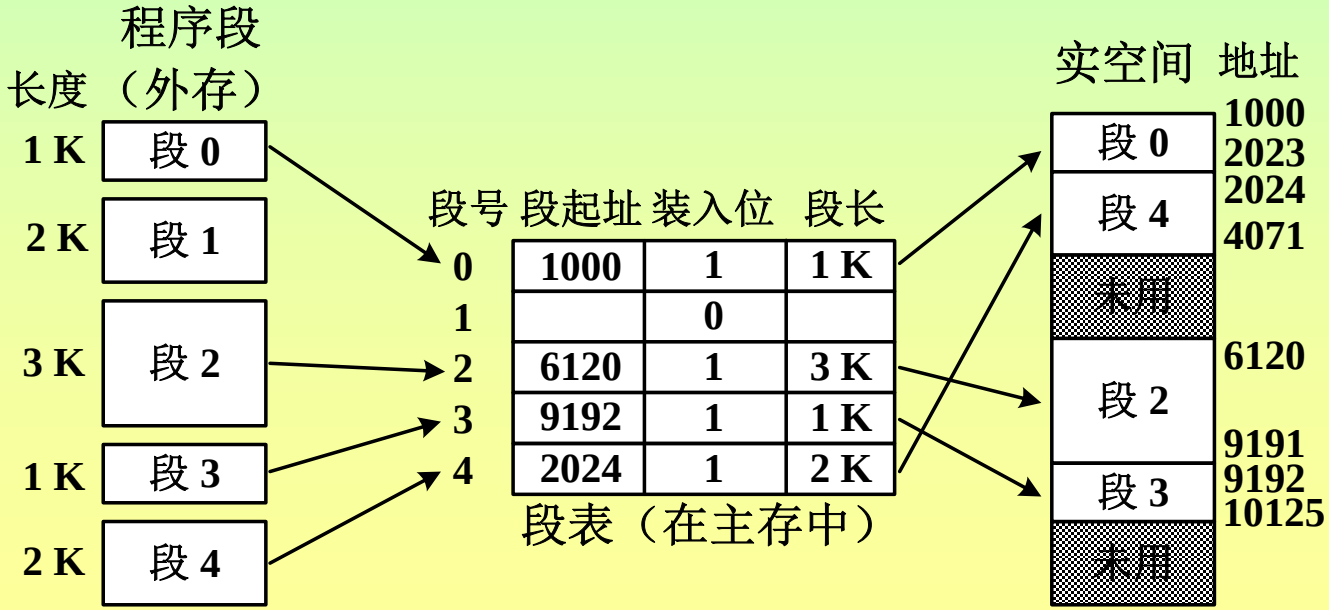
虚页号20位，因此有2^{20}＝1M个虚页，页表大小为1M×16位。



段式管理

段是按照程序的逻辑结构划分的，各个段的长度因程序而异。虚地址由段号和段内地址组成。





为了把虚地址变换成实主存地址，需要一个段表。由于段的长度不固定，段表中需要有长度指示。段表也是一个段。

段页式管理

把程序按逻辑单位分段以后，再把每段分成固定大小的页。程序对主存的调入调出是按页面进行的，但它又可以按段实现共享和保护，兼备页式和段式的优点。

每道程序是通过一个段表和一组页表来进行定位的。段表中的每个表项对应一个段，每个表项包括该段的页表起始地址及该段的控制保护信息。由页表指明该段各页在主存中的位置以及是否已装入、已修改等状态信息。

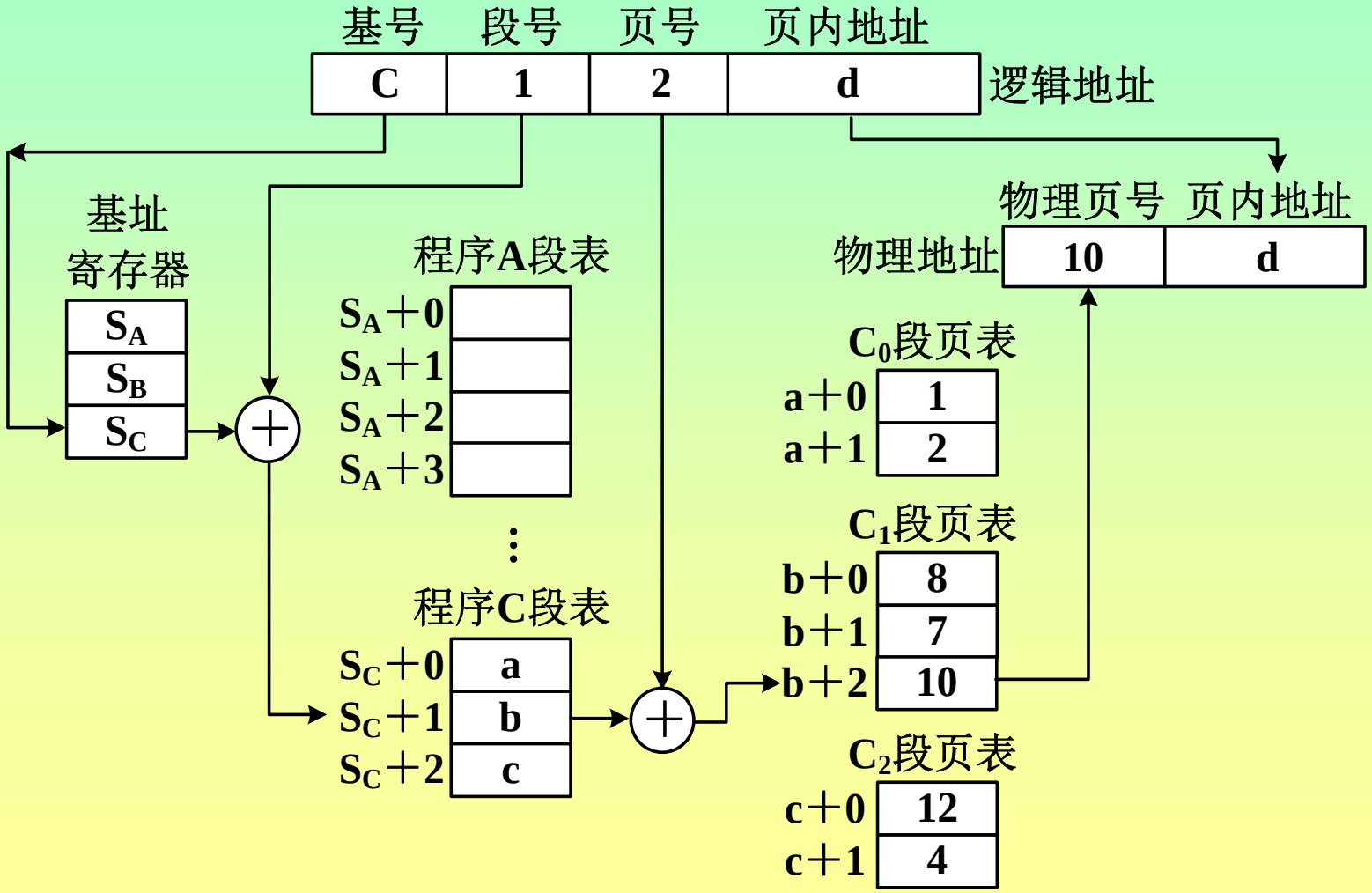
缺点是在映象过程中需要多次查表。

如果有多个用户在机器上运行，多道程序的每一道需要一个基号，由它指明该道程序的段表起始地址。

虚拟地址格式如下：

基号	段号	页号	页内地址

例：有三道程序（A、B、C），每道程序有一张段表，段表起始地址分别为S_A、S_B、S_C，每段有一张页表。段表的每行就是相应页表的起始地址，页表的每行就是相应的物理页号。



替换算法

虚拟存储器中的替换策略一般采用LRU算法、LFU算法 、FIFO算法，或将两种算法结合起来使用。对于将被替换出去的页面，假如该页调入主存后没有被修改，就不必进行处理，否则就把该页重新写入外存。为此，在页表的每一行应设置一修改位。

例：主存只有a、b、c三个页，组成a进c出的FIFO队列，进程访问页面的序列是0，1，2，4，2，3，0，2，1，3，2号。若采用①FIFO算法，②FIFO算法+LRU算法，求两种替换策略情况下的命中率。

页面访问序列		0	1	2	4	2	3	0	2	1	3	2	命中率
FIFO	a	0	1	2	4	4	3	0	2	1	3	3	2/11=18.2%
	b		0	1	2	2	4	3	0	2	1	1	
	c			0	1	1	2	4	3	0	2	2	
FIFO + LRU	a	0	1	2	4	2	3	0	2	1	3	2	3/11=27.3%
	b		0	1	2	4	2	3	0	2	1	3	
	c			0	1	1	4	2	3	0	2	1	

虚拟存储器实例

奔腾CPU的存储管理部件MMU包括分段部件SU和分页部件PU两部份，可单独或同时工作。有三种虚拟地址模式。

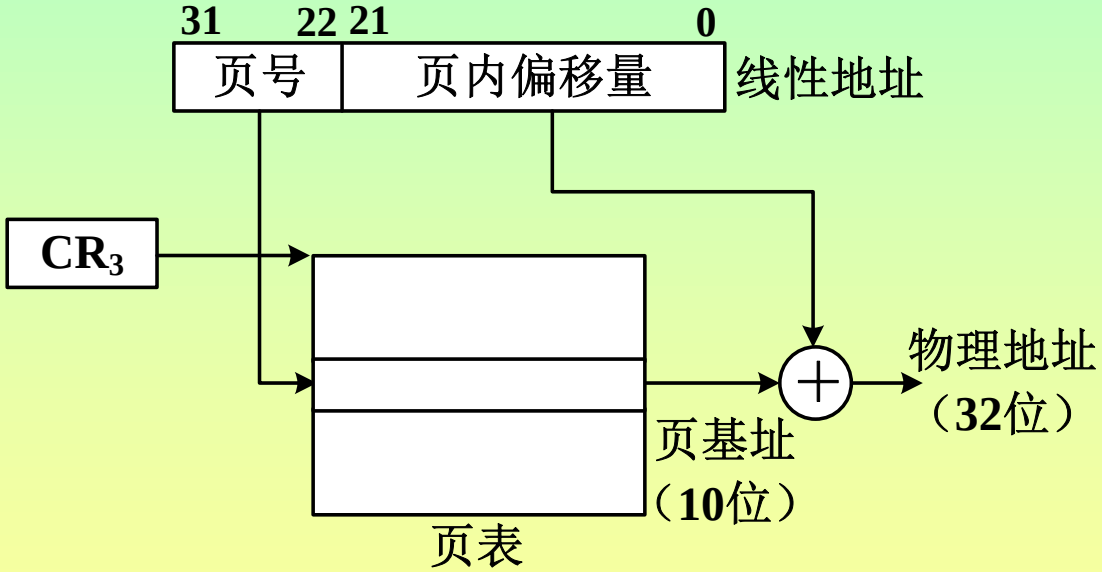
分段不分页模式： 虚拟地址由一个16位的段选择符和一个32位的偏移量组成。段选择符的最低两位是特权级，高14位指定具体的段。一个进程可拥有的最大虚拟地址空间为$2^{14+32}=2^{46}=64$ TB。

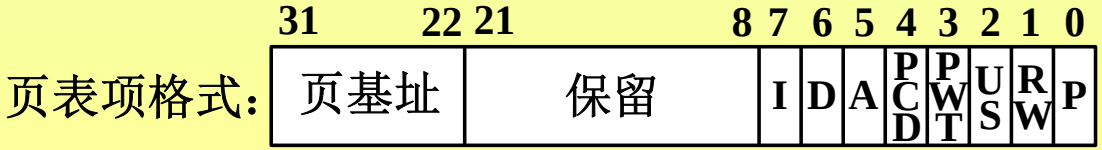
分段分页模式：在分段基础上增加分页存储管理。分页管理包括两级页表，分别称为页目录表和页表。SU部件转换后的32位地址称为线性地址，包括10位页目录索引、10位页表索引和12位页内偏移量。再由PU部件完成两级页表的查找，得到20位的实页号，和12位页内偏移量拼接，得到32位物理地址。进程的最大虚拟地址空间也是64 TB。

不分段分页模式：这种模式下SU不工作，只是分页部件PU工作。32位虚拟地址被看成是由页目录、页表、页内偏移量三个字段组成。由PU完成虚拟地址到物理地址的转换。进程的最大虚拟地址空间是4 GB。

分页地址转换

奔腾CPU有两种分页方式：一种是4 KB的页，使用二级页表（页目录表、页表）进行地址转换；另一种是4 MB的页，用单级页表进行转换。后一种方式下，32位线性地址分为高10位的页号和低22位的页内偏移量。全系统只有一张页表，控制寄存器CR_3指向页表的起始地址。页表有1 K个表项，每项32位。





存储保护

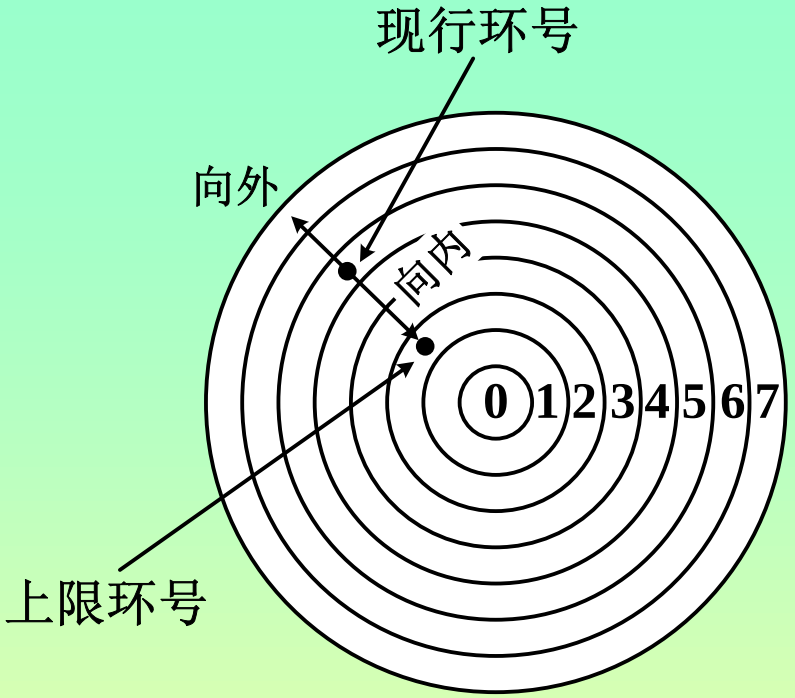
存储保护的目的是防止一个用户程序出错而破坏其他用户的程序和系统软件，还要防止一个用户程序不合法地访问不是分配给它的主存区域。包括存储区域保护和访问方式保护。

页表和段表保护：每个程序的段表和页表都有自身的保护功能。每个程序的虚页号是固定的，如果虚页号出错，则在页表中找不到，也就访问不了主存。段表和页表的保护功能相同。另外，段表项中包含段长，段长通常就是该段包含的页数。如果页号大于段长，说明该页号非法。

键保护：为主存的每一页配一个键，称为存储键。访问键赋予每道程序。当数据要写入主存的某一页时，访问键和存储键比较。二者相符则允许访问该页，否则拒绝访问。

另外还有取数保护。为每页设置一个一位的取数键。取数键为0则该页只受存数保护；为1则受到存取保护，只有访问键和存储键相同的程序才能存取这些页。

环保护：将系统程序和用户程序分层，每层叫做一个环。环号越大，等级越低。在现行程序运行前由操作系统定好程序各页的环号。程序可以访问任何外层空间；访问内层空间则需由操作系统来判断这个向内访问是否合法。每个程序有上限环号，程序不能访问低于上限环号的存储区域。



访问方式保护：对主存信息的使用有三种方式：读 (R)、写 (W)和执行 (E)。相应的访问保护方式就是R、W、E三种方式的逻辑组合。

逻辑组合	含义	逻辑组合	含义
$\overline{R+W+E}$	不允许任何访问	$\overline{(R+E)}\cdot W$	只能写访问
$R+W+E$	可进行任何访问	$(R+E)\cdot\overline{W}$	不准写访问
$(R+W)\cdot\overline{E}$	只能读写，不可执行	$R\cdot\overline{(W+E)}$	只能读访问
$\overline{(R+W)}\cdot E$	只能执行，不可读写	$\overline{R}\cdot(W+E)$	不准读访问