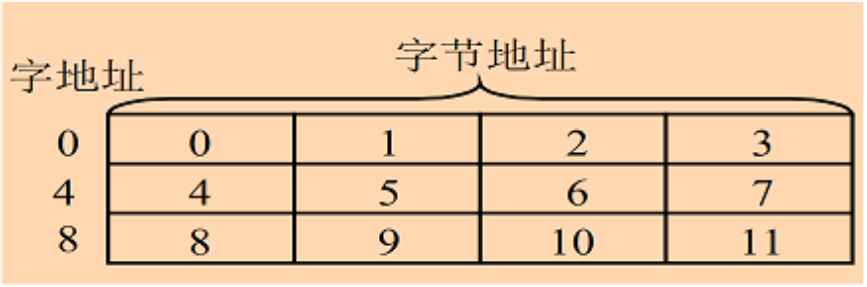
存储系统和结构

笔记本： 计算机组成原理
创建时间： 2021/6/27 12:12 **更新时间：** 2021/6/28 17:10
作者： 134exetj717

本章内容，我们主要讲讲存储系统和结构。

- 首先，我们需要知道，存储系统具体都有哪些分类呢？
  - 作用分类：高速缓冲存储器、主存储器、辅助存储器
  - 存取方式分类：
    - 随机存取存储器（RAM）：对存储器的内容随机的存取
    - 只读存储器（ROM）
    - 顺序存取存储器（SAM）：其内容只能按照某种顺序存取，如果有说明，那么一般都是执行完一条指令之后，指令地址自动+1
    - 直接存取存储器（DAM）：第一步直接指向存储器的某个小区域；第二步在小区域内顺序检索或等待
  - 存储介质分类：
    - 磁芯存储器：信息可以长期存储，不会因断电而丢失；但其读出是破坏性读出，读完以后数据就是消失
    - 半导体存储器：信息会因为断点而丢失
    - 磁表面存储器：用磁层存储信息，容量大、价格低、存取速度慢，故多用作辅助存储器
    - 光存储器：采用激光技术，一般分为只读式、一次写入式、可改写式三种，存储容量大，目前使用最广泛的辅助存储器
  - 按信息的可保护性分类：
    - 易失性存储器：断电后存储信息即消失的存储器，如半导体RAM
    - 非易失性存储器：断电后信息仍存在，如以上1，3，4三种
- 了解了分类以后，我们需要知道存储系统的层次结构是怎样的呢？
  - 三级存储系统：高速缓冲存储器、主存储器、辅助存储器
  - 其中该三级系统可以分为两个层次：Cache-主存存储层次（Cache存储系统）、主存-辅存层次（虚拟存储系统）。
  - 那么我们会疑惑，为什么需要这三级存储系统呢？
    - 原因是：为了解决存储容量、存取速度和价格之间的矛盾。
- 众所周知，存储系统都是由很多个地址位构成的，而数据都存储在地址中，那么我们如何对其编制呢？
  - 按字编址：编制单位==计算机字长
    - 一行代表一个计算机字长，如果我们的存储字包含4个单独字节，计算机字长32位，那么每一行都有4个存储字，因为一个字节8位，总共满足32位。因此每一行的开头地址为4的整数倍。
    - 大端方案：字地址等于最高有效字节地址
    - 小端方案：字地址等于最低有效字节地址
    - 

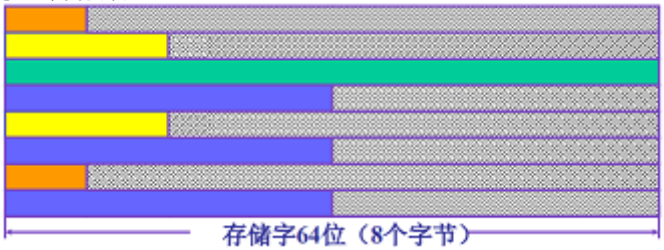
  - 按字节编址：编制单位==1个字节

- 紧缩存放

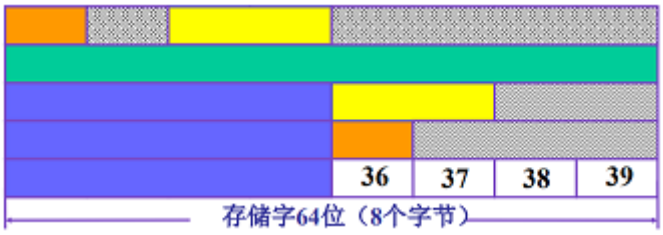
1）紧缩存放：4种不同长度的数据一个紧接着一个存放。

不浪费存储器资源，但一个数据可能跨字存放，要读数要访存两次

- 字边界存放



- 整数边界存放

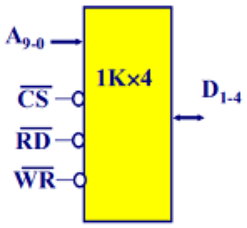


  - 按位编址：编制单位==1bit

- 要了解存储系统，我们就需要了解其内部的芯片。我们知道，存储体是由多个芯片组合而成，比如半导体随机存储器和只读存储器。仅仅知道数据在主存中的存放方式是不够的，我们还需要知道如何调用这些地址，即地址译码如何运作呢？
  - 单译码方式（一维地址译码方式，字选法）

以 1K×4 的芯片为例

（1K 个单元地址，每单元 4 位可同时读/写）



地址线： 10 条， A_9-A_0 (2^{10}=1K)

数据线： 4 条双向，D_{1-4}

片选信号：$\overline{CS}$，低电平时芯片才能读写

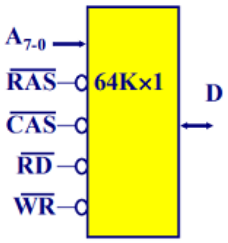
读写控制：$\overline{RD}$ 为低时读出；$\overline{WR}$ 为低时写入

（有的芯片用 $\overline{WE}$ 控制读写(0-读，1-写)

  -
  - 双译码方式（二维地址译码）

以 64K×1 的芯片为例（需 16 位地址 （2^{16}=64K））

采用二维地址译码



用 8 根地址线(分时复用)：

$\overline{RAS}$ 下降沿时从地址线送入行地址 A_{15-8}

$\overline{CAS}$ 下降沿时从地址线送入列地址 A_{7-0}

数据线： 1 条双向线

（有的芯片用 2 条单向线 Din, Dout）

读写控制：$\overline{RD}$ 为低时读出；$\overline{WR}$ 为低时写入

（有的芯片用 $\overline{WE}$ 控制读写(0-读，1-写)）

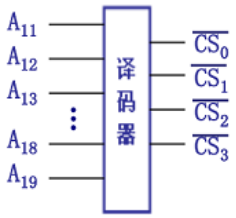
DRAM芯片一般没有片选端，用$\overline{RAS}$和$\overline{CAS}$信号代替片选

- 了解了地址译码的原理，那么对于RAM，她是如何存储信息的呢？主要依靠栅极电容。在我们使用的过程中，电容肯定会存在损耗，为了让我们存储体能够长期保持数据，我们比如给芯片，比如RAM补充栅极电容。那么，我们如何补充呢？当然，SRAM不需要刷新，而DRAM需要。
  - 首先，我们需要重申，为什么每隔一段时间都需要刷新RAM芯片？
    - 原因是：为了维持DRAM记忆单元的存储信息。
  - 可能我们会问，多久需要进行一次刷新呢？
    - 这主要是根据栅极电容上电荷的泄放速度来决定。一般选定的最大刷新间隔为 2ms 或者 4ms 甚至更大。也就是说，在规定时间内，我们必须将全部存储体刷新一遍。
  - 读到这里，我们可能会有疑惑，刷新和重写（再生）都类似于在重新写数据，那么两者有什么区别呢？
    - 重写是随机的，只有在某个存储单元被破坏读出之后才需要重写；而刷新是定时的，即使许多记忆单元长期未被访问，若不及时补充电荷的话，信息也可能会丢失
    - 重写一般是按存储单元进行的，而刷新通常是以存储矩阵中的一行为单位进行
  - 谈到这里，我们可能会问，什么是存储矩阵？
    - 首先，存储矩阵是由芯片组成的；
    - 其次，芯片是由存储容量（地址线）和数据线组成的；
    - 最后，我们可以用存储容量，比如 1024kb，的位数当作矩阵的列；用数据线的位数，当作矩阵的行。
    - 也就是说，刷新是指刷新一整条数据线的一个位。
  - 假如给定了最大刷新间隔 2ms ，以及芯片的记忆单元数 n*n ，和芯片的横向数量 m，那么我们怎么求芯片刷新周期数呢？
    - 很简单，我们用 n 就可以得到最终答案-->刷新周期数（要注意看是求芯片刷新数还是存储体刷新数）
    - 关键点在于，芯片是 正方形的存储矩阵
- 从存储体的基本分类功能，到数据的存放，再到地址译码，又有了芯片内部的刷新，由宏观到微观，那么此刻，我们了解了内部构造，重新从宏观角度出发，若芯片不够大，需要多片来实现一个存储体，我们该如何进行主存容量的扩展呢？
  - 字，位，字位扩展
    - 字扩展：字就相当于容量，更多的代表了地址数量，字扩展表示芯片的地址容量不足以制造该存储器，需要增加芯片；
    - 位扩展：数据总线一次所能并行传送信息的位数，也叫数据通路宽度。一般的位数表示数据传输的宽度，当芯片的位宽小于存储器的宽度时，需要进行位扩展；
    - 字位扩展：当芯片的容量，位宽都小于存储器时，就需要字位同时扩展。
    - 比如：
      - 前提：有一个 4k*8 的芯片，需要制造一个 8k*16 的存储器。

        - 说明：芯片容量位 m字*n位，存储器容量为 M字*N位。
        - 分析：
            - m==M，n<N -> 位扩展
            - m<M，n==N -> 字扩展
            - m<M，n<N ->字位扩展
        - 解题：　　字位扩展，（8/4）*（16/8）= 4；因此需要 4片芯片
- 主存容量扩展完毕以后，我们又如何控制每个芯片的运作呢？
    - 片选：选择存储器芯片，由高位地址决定
        - 线选法
            - 1个位对应控制1个片内地址（由于是1-1对应关系，因为执行速度非常快）

芯片	A_{19}～A_{15} (不用)	A_{14}～A_{11} (片选地址)	A_{10}～A_0 (片内地址)	地址范围（空间）
0#	00000	1 1 1 0	000 0000 0000 …… 111 1111 1111	07000～077FFH
1#	00000	1 1 0 1	000 0000 0000 …… 111 1111 1111	06800～06FFFH
2#	00000	1 0 1 1	000 0000 0000 …… 111 1111 1111	05800～05FFFH
3#	00000	0 1 1 1	000 0000 0000 …… 111 1111 1111	03800～03FFFH

        - 全译码法
            - 1个位可以控制2个片内地址（类似于3-8译码器这种，以少定多，但是比较复杂）



芯片	A_{19}～A_{11} (片选用)	A_{10}～A_0 (片内地址)	地址范围(空间)
0#	0000 0000 0	000 0000 0000 …… 111 1111 1111	00000～007FFH (0～2K-1)
1#	0000 0000 1	000 0000 0000 …… 111 1111 1111	00800～00FFFH (2K～4K-1)
2#	0000 0001 0	000 0000 0000 …… 111 1111 1111	01000～017FFH (4K～6K-1)
3#	0000 0001 1	000 0000 0000 …… 111 1111 1111	01800～01FFFH (6K～8K-1)

            全译码法特点：每片芯片的地址范围是唯一确定且连续的，不会产生地址重叠的存储器，对译码电路要求较高。
        - 部分译码法
            - 共有3个位，那么用2个位控制2个片内地址，最后再用1个位控制2个片内地址

芯片	A_{19}～A_{13} A_{12}～A_{11} (不用) (参与译码)	A_{10}～A_0 (片内地址)	地址范围（空间）
0#	0000 0000 0	000 0000 0000 …… 111 1111 1111	00000～007FFH (0～2K-1)
1#	0000 0000 1	000 0000 0000 …… 111 1111 1111	00800～00FFFH (2K～4K-1)
2#	0000 0001 0	000 0000 0000 …… 111 1111 1111	01000～017FFH (4K～6K-1)
3#	0000 0001 1	000 0000 0000 …… 111 1111 1111	01800～01FFFH (6K～8K-1)

        - 注意：在片选的过程中，只有横向的不同芯片才能参与片选，注意看下面细节，因为两个芯片合成一个数据总线相当于一个芯片：

假定芯片的读写控制也采用两条线(RD和WR)，则存储器的组成逻辑图为：