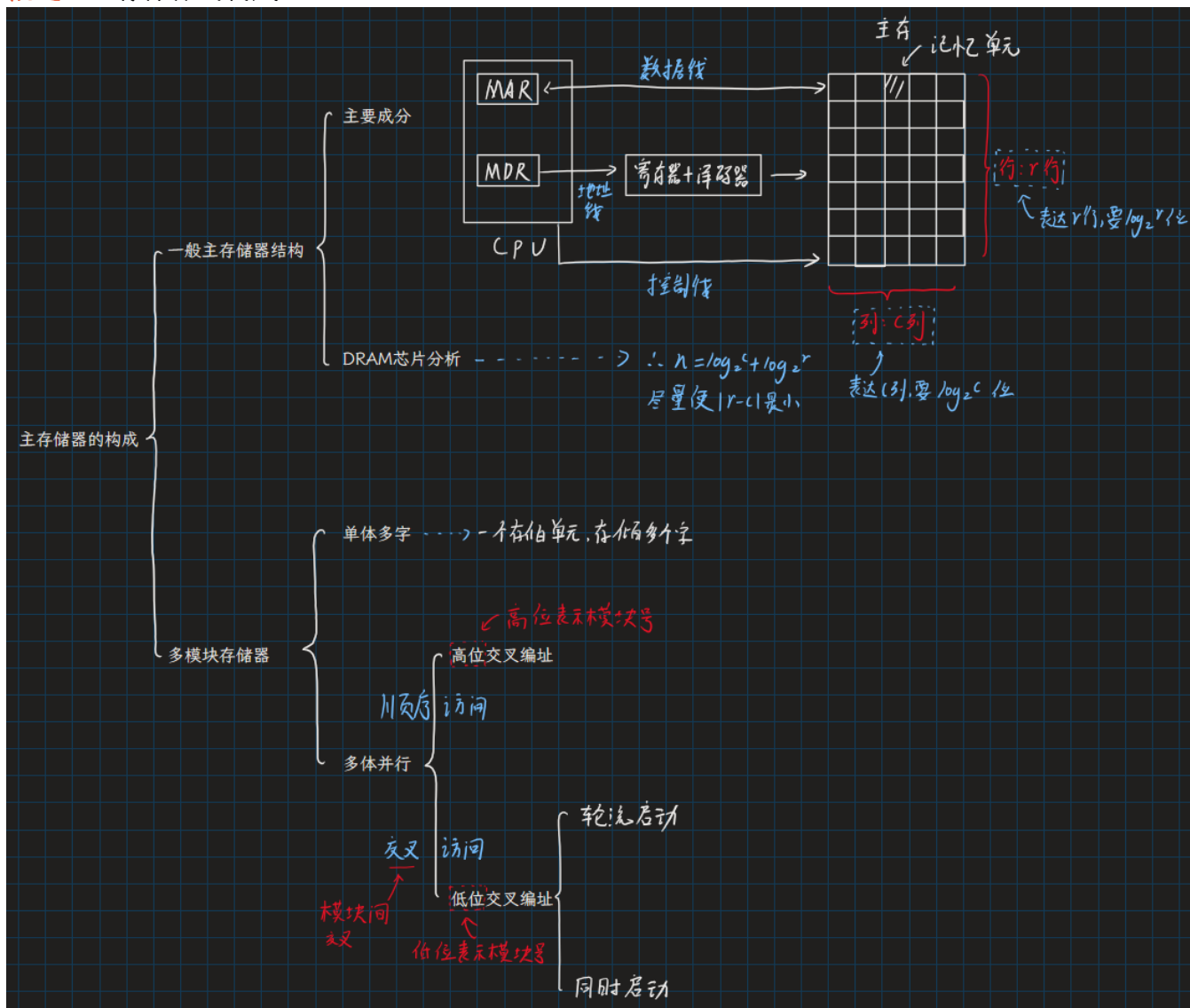


Lecture 9: 知识 - 主存储器

主存储器构成

概述: 主存储器的构成

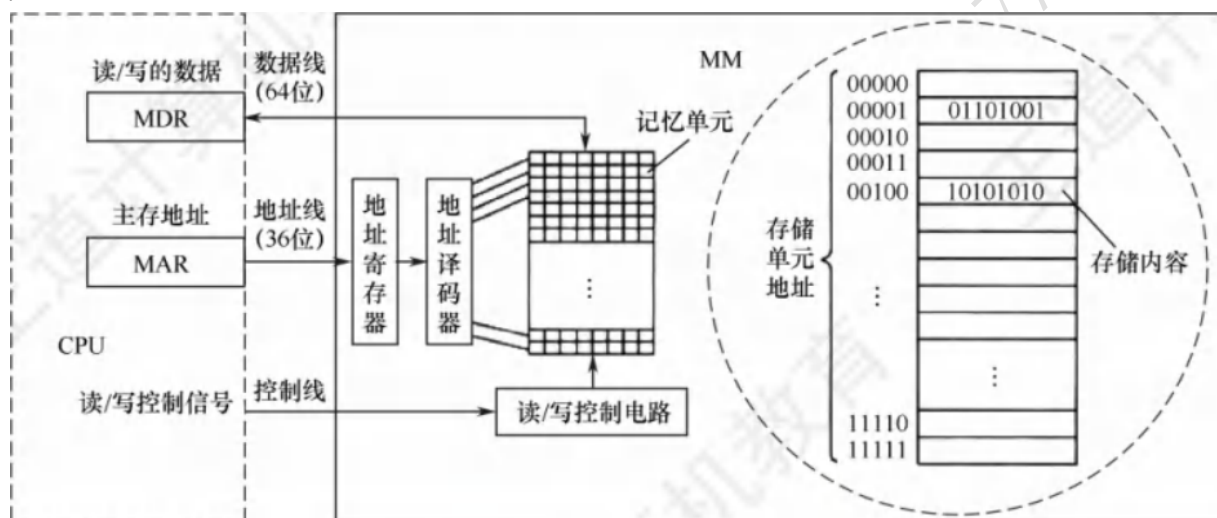


主存储器的存储结构

概念: 主存储器

- 名词:
 - 主存储器 (Main Memory, MM);
- 概念:
 - 记忆单元 (存储元件):
 - 记忆单元: 由一个个存储 0 或 1 的**记忆单元** (也称**存储元件**) 构成的**存储矩阵** (也称**存储体**) 是存储器的核心部件;
 - 存储元件: 是具有两种稳态的能表示二进制 0 和 1 的物理器件;
 - 编址单位:
 - 为了存取存储体中的信息, 必须对存储单元编号 (也称编址);

- 编址单位是指**具有相同地址的那些存储元件构成的一个单位**，可以按字节编址，也可以按字编址；
- 现代计算机通常采用字节编址方式，此时存储体内的一个地址中有 **1** 字节；
- 分析：
 - 图示采用 **64** 位数据线，所以在按 **字节编址方式** 下，每次最多可以存取 **8** 个单元的内容；
 - **地址线的位数**决定了主存地址空间的最大可寻址范围；
 - 例如：**36** 位地址的最大寻址范围为 $(0, 2^{36} - 1)$ ，即地址从 **0** 开始编号；
- 注意：
 - **数据线的位数通常等于存储字长**，因此 **MDR** 的位数通常等于**存储字长**；
 - 若数据线的位数不等于存储字长，则 **MDR** 的位数由数据线的位数决定；
- 图示：



概念： **MAR** 和 **MDR** 位数与地址线/数据线数的关系、寻址范围的计算

- 指令执行过程中需要访问主存时，**CPU** 首先把**被访问单元的地址**送到 **MAR** 中；
- 然后通过**地址线**将主存地址 $\rightarrow$ 主存中的地址寄存器，以便地址译码器进行译码，选中相应单元，同时 **CPU** 将读/写信号通过控制线送到主存的读/写控制电路；
- 若是写操作：
 - 则 **CPU** 同时将要写的信息送到 **MDR** 中，在**读/写控制电路**的控制下，经数据线将信号写入选中的单元；
- 若是读操作：
 - 则**主存**读出选中单元的内容送至数据线，然后被送到 **MDR** 中；
 - **MDR** 的位数与数据线的位数相同，**MAR** 的位数与地址线的位数相同；

DRAM 芯片分析

题型： **DRAM** 芯片的地址引脚复用技术

- 概念：
 - **DRAM** 芯片容量较大、地址位数较多，为了减少芯片的**地址引脚数**，通常采用地址引脚复用技术；
- 实现：

- 行地址和列地址通过相同的引脚分先后两次输入，这样地址引脚数可减少一半；

题型： DRAM 芯片行、列数的优化原则

- 假定有一个 $2^n \times b$ 位的 DRAM 芯片的存储阵列，其行数为 r 位，列数为 c 位，则 $2^n = r \times c$
- 存储阵列的地址位数为 n ，其中行地址位数为 $\log_2 r$ ，列地址位数为 $\log_2 c$ ，即 $n = \log_2 c + \log_2 r$
- 为减少地址引脚数，应尽量使行、列位数相同，即满足 $|r - c|$ 最小；
- 又由于 DRAM 按行刷新，为减少刷新开销，应使行数较少，因此还需满足 $r \leq c$ ；

多模块存储器

概念：多模块存储器是一种空间并行技术，利用多个结构完全相同的存储模块，并行工作（因为 CPU 的速度比存储器快得多，所以如果可以同时从 n 个存储器读取就可以提高利用率），提高存储器的吞吐率；

单体多字存储器

概念：单体多字存储器

- 概念：
 - 在单体多字系统中，每个存储单元存储 m 个字，总线宽度也为 m 个字，一次并行读出 m 个字；
 - 在一个存取周期内，从同一地址取出 m 条指令，然后将指令逐条送至 CPU 执行，即每隔 $1/m$ 存取周期，CPU 向主存取一条指令；
- 优点：
 - 提高了单体存储器的工作速度；
- 缺点：
 - 只有指令和数据在**主存中连续存放时**，这种方法才能有效提升存取速度；
 - 一旦**遇到转移指令**，或操作数不能连续存放时，这种方法的提升效果就不明显；

多体并行存储器

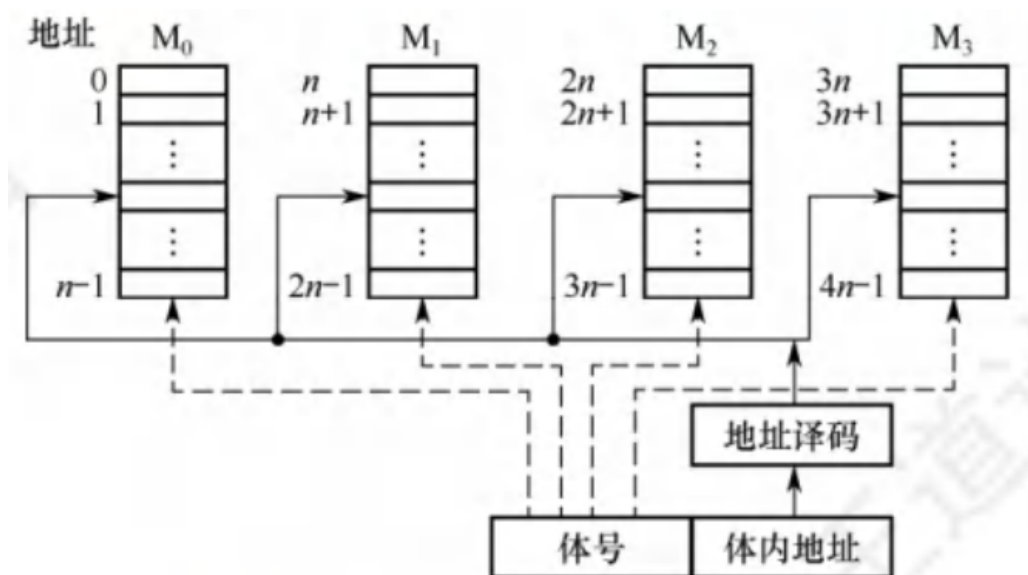
引入：多体并行存储器

- 介绍：
 - 多体并行存储器由多体模块组成，每个模块都有相同的容量和存取速度，各模块都有**独立的读/写控制电路、地址寄存器和数据寄存器**；
- 特点：
 - 不同的存储器之间，既能并行工作，又能交叉工作；
- 编址方式：
 - 高位交叉编址
 - 低位交叉编址

连续编址方式

概念：连续编址（又称为 高位交叉编址方式）

- 概念：
 - 高位地址表示模块号（体号），低位地址表示模块内地址（体内地址）；
 - 在高位交叉方式下，总把**低位的体内地址**送到高位体号确定的模块内进行译码；
 - 访问一个连续主存块时，总是先在一个模块内访问，等到**该模块访问完**才转到下一个模块访问，CPU 总是按顺序访问存储模块；
- 特点：
 - 由于各模块不能被并行访问，因而**不能提高存储器的吞吐率**；
- 图示：



交叉编址方式

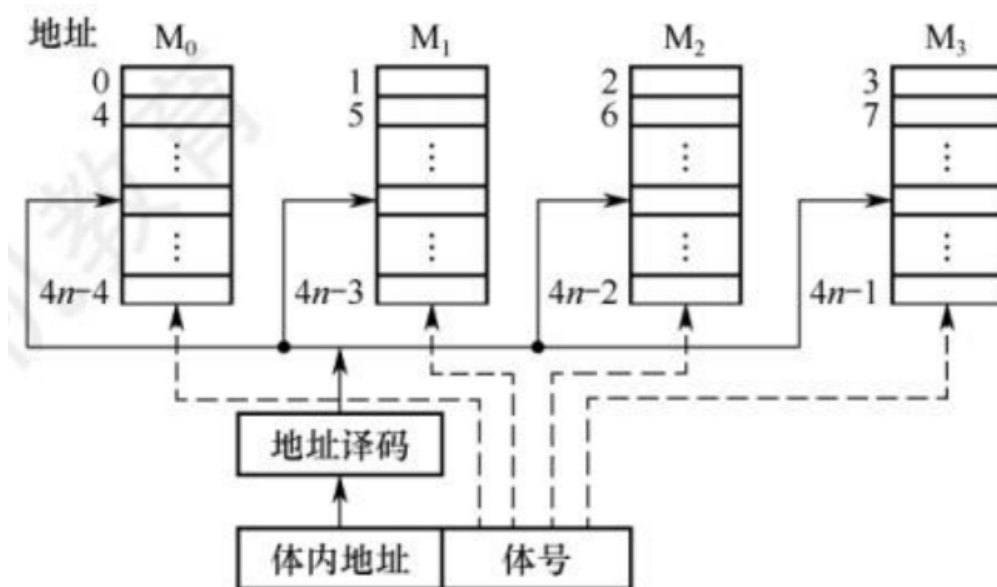
交叉编址方式的实现

概念：低位交叉编制 - 交叉方式

- 概念：
 - 低位地址为模块号（体号），高位地址为模块内地址（体内地址）；
 - 并且总是把**高位的体内地址**送到由**低位体号**所确定的模块内进行译码；
 - 每个块按“模 m ”交叉编址；
 - 模块号 = 单元地址 $\bmod m$ ；
 - 程序连续存放在相邻模块中，因此称采用此编址方式的存储器为**交叉存储器**；
- 举例：
 - 假定有 m 个模块，每个模块有 k 个单元，则：
 - 单元 $0, m, 2m, 3m, 4m, \dots, (k-1)m$ 位于 M_0
 - 单元 $1, m+1, 2m+1, 3m+1, 4m+1, \dots, (k-1)m+1$ 位于 M_1 ；
 - 单元 $2, m+2, 2m+2, 3m+2, 4m+2, \dots, (k-1)m+2$ 位于 M_2 ；
 - 以此类推；
- 补充：
 - 模块数一般是 2 的幂次方，这样电路比较简单；
 - 但如果想要减少存储体冲突，也可以使用质数个模块；
- 模块数分析：

- 轮流启动：
 - 若每个模块一次读/写的位数正好等于数据总线位数，模块的存取周期为 T ，总线周期为 r ；
 - 为实现轮流启动方式，存储器交叉模块数应大于或等于 $m = \frac{T}{r}$ ，所以模块数小于或等于 $\frac{1}{m}$
- 同时启动：
 - 若所有模块一次并行读/写的总位数（各个模块读写的位数之后）正好等于数据总线位数，则可以同时启动所有模块进行读/写；

• 图示：



低位交叉编制的两种启动方式

概念： 轮流启动

- 概念：
 - 模块的存取周期为 T ；
 - 总线周期为 r ；
 - 轮流启动的时间间隔 = $\frac{1}{m} = \frac{r}{T} >$ 模块个数的倒数
- 实现：
 - 按每隔 $1/m$ 个存取周期轮流启动各模块，则每隔 $1/m$ 个存取周期就可读出或写入一个数据，存取速度提高 m 倍；
 - 交叉存储器要求其模块数大于或等于 m ，以保证启动某模块后经过 $m * r$ 的时间后再次启动该模块时，其上次的存取操作已经完成（以保证流水线不间断）；
- 时间：
 - 分析：
 - 连续存取 m 个字所需的时间为：

$$t_1 = T + (m - 1) * r$$

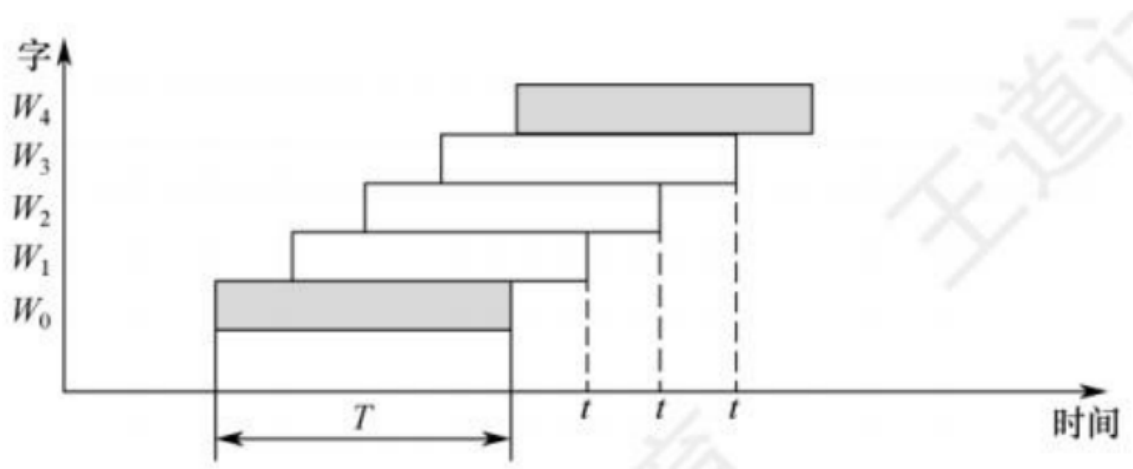
- T ：读取完第一个模块的所画时间为模块的存取周期 T ；
- $(m - 1) * r$ ：因为是按照流水线方式进行存取，所以在进行第一个 T 同时，也开始了其他模块的存储，因此在完成 T 之后、完成 W_1 只需要再加上一个总线周期 r 即

可；

- 比较：

- 而顺序方式连续读取 m 个字所需的时间为 $t_2 = mT$
- 交叉存储器的带宽相比顺序大大提高；

- 图示：

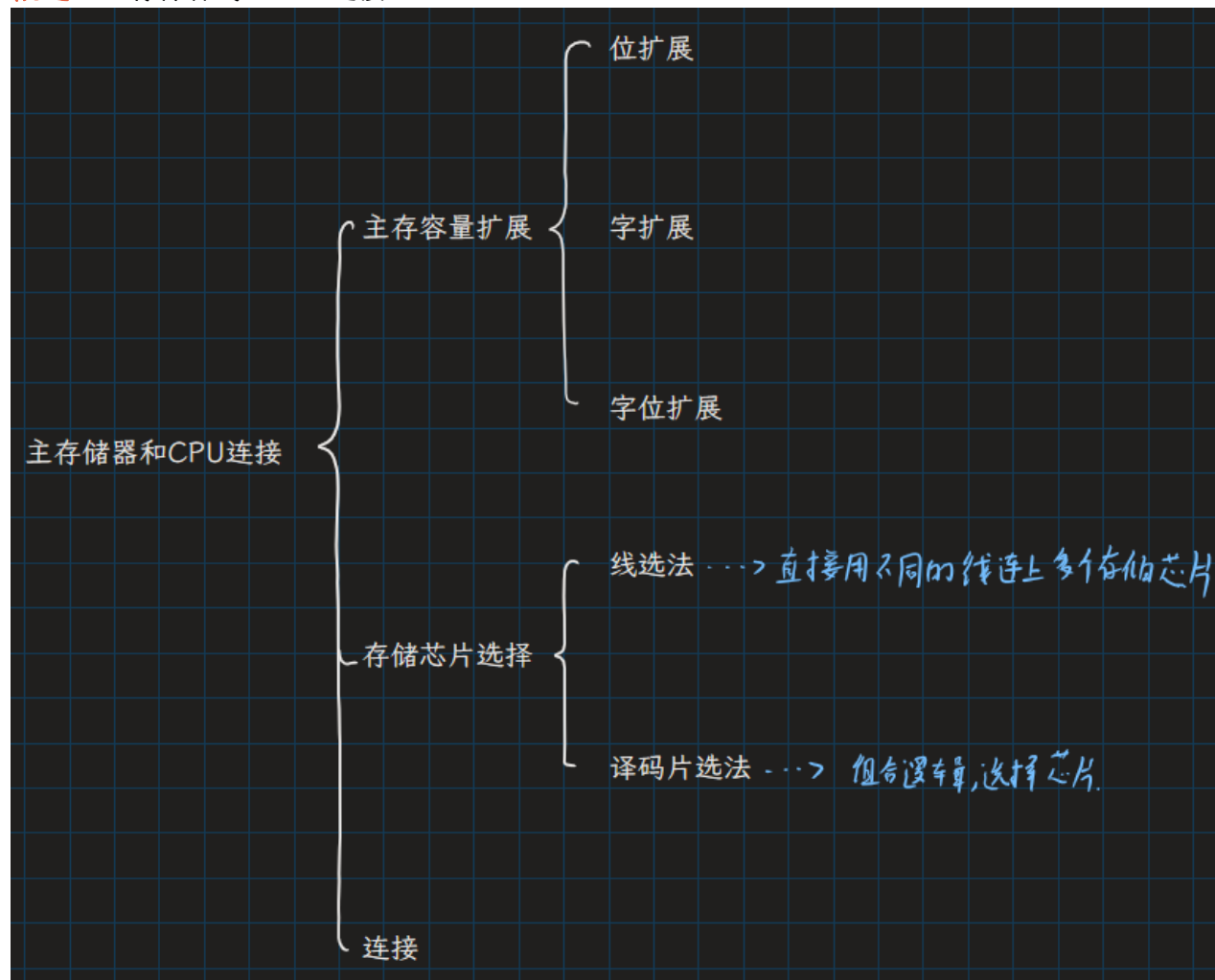


概念： 同步启动方式

- 设每个模块一次读/写的位数为 16 位，模块数 $m=4$ ，数据总线位数为 64 位，4 个模块一共提供 64 位，正好构成一个存储字，因此应该同时启动 4 个模块进行并行读/写；

主存储器与 CPU 连接

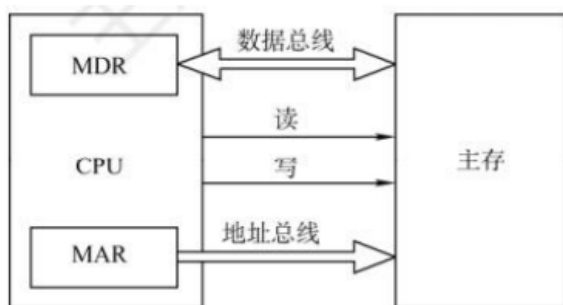
概述：主存储器与 CPU 连接



连接原理

概念：主存储器和三总线

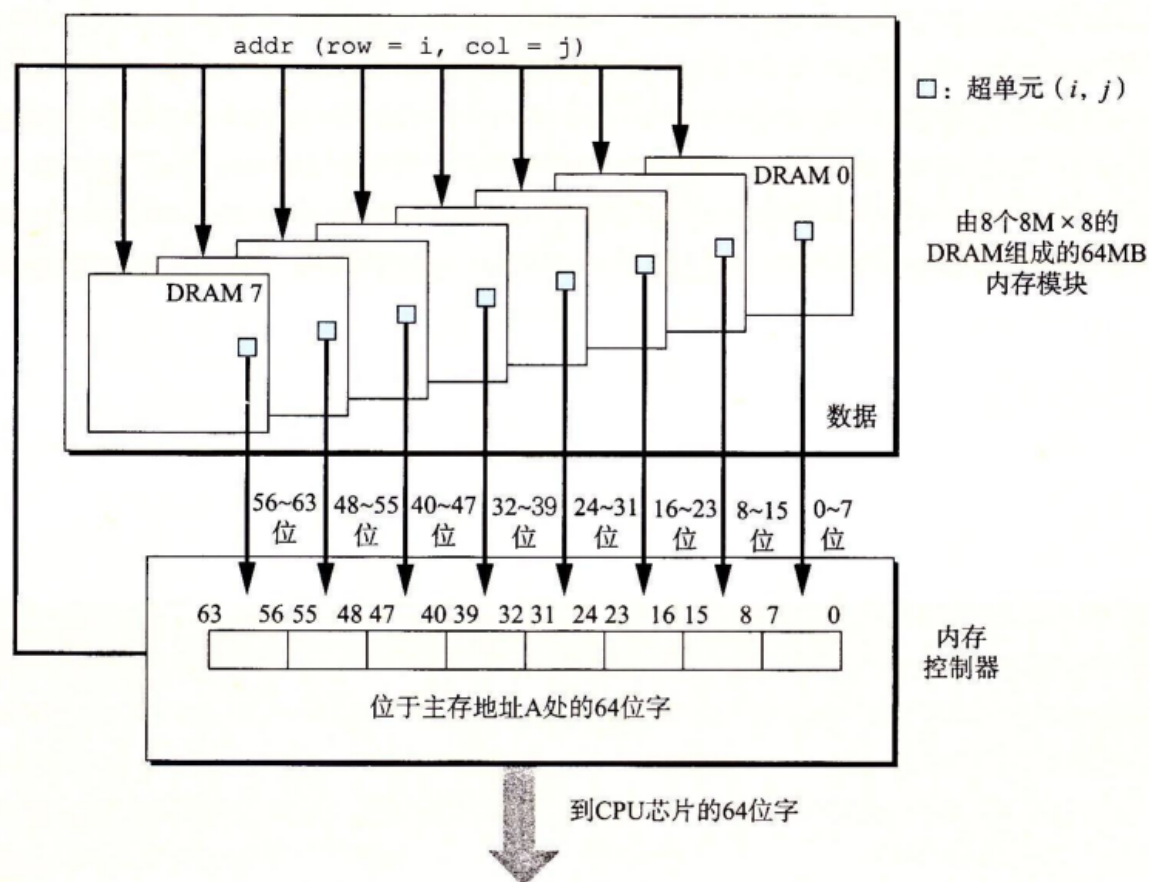
- 概念：
 - 1. 主存储器通过数据总线、地址总线、控制总线与 CPU 连接；
 - 2. **数据总线**的位数与工作频率的乘积正比于数据传输速率；
 - 3. **地址总线**的位数决定了可寻址的最大内存空间；
 - 4. **控制总线** (读/写) 指出总线周期的类型和本次输入/输出操作完成的时刻；
- 图示：



补充：CPU 与主存中数据交互

- 概念：

- DRAM 的芯片封装在以下内存模块当中，它被插到主板的扩展槽上；
- 图示是使用 8 个 64 Mbit 的 8M*8 的 DRAM 芯片，总共存储 64MB；
- 每个超单元存储主存的一个字节，且用相应的超单元地址 (i,j) 为 8 个超单元带代表主存中字节地址 A 处的 64 位字；
- 即：DRAM 0 存储第一个字节（低位），DRAM 1 存储第二个字节，一次类推，它们一起传入数据线后、构成一个 64 位字；
- 取出数据：
 - 要取出内存地址 A 处的一个字，内存控制器将 A 转换成一个超单元地址 (i, j)，并将它发送到内存模块，然后内存模块再将 i 和 j 广播到每个 DRAM；
 - 作为响应、每个 DRAM 输出它的 (i, j) 超单元的 8 位内容；
 - 模块中的电路收集这些输出，并把它们合并成一个 64 位字，再返回给内存控制器；
- 图示：



主存容量的扩展

概念：主存容量扩展的原因

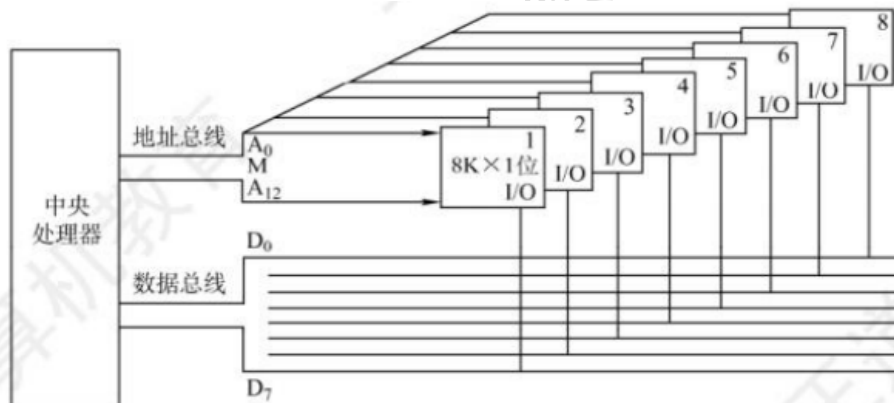
- 原因：
 - 单个芯片的容量是有限的，因此通过存储器芯片扩展技术，将多个芯片集成在一个内存条上；
 - 由多个内存条及主板上的 ROM 芯片 组成计算机所需的主存空间，再通过总线与 CPU 相连；
- 三种扩展：
 - 1. 位扩展：8 个 8k*1位 -> 一个 8k*8位；

- 2. 字扩展：4 个 $16k \times 8$ 位 $\rightarrow$ 一个 64×8 位
- 3. 字位扩展；

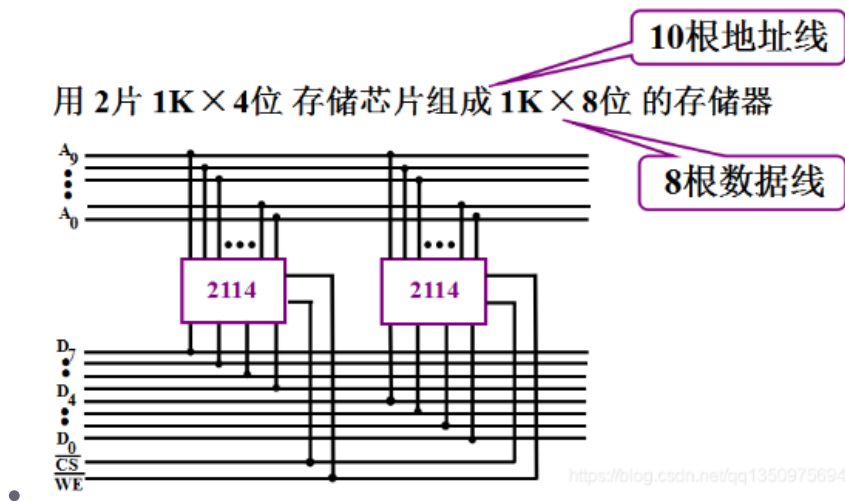
位扩展法

概念：什么是位扩展

- 核心：
 - 数据总线的线数 $>$ 单个存储芯片的数据位数
- 概念：
 - 位扩展是指**对字长进行扩展** (增加存储字长)；
 - 位扩展就是增大数据的范围；
 - 当 CPU 的 系统数据线数 $>$ 存储芯片的数据位数 时，必须对存储芯片扩位，使其**数据位数与 CPU 的数据线数相等**；
- 举例一：
 - 分析：
 - 有 8 片 $8K \times 1$ 位的 RAM 芯片，组成 $8K \times 8$ 位的存储器；
 - 其 8 片 RAM 的地址线 $A_{12} \sim A_0 + CS + WE$ 连在一起，每片的数据线以此作为 CPU 数据线的一位；
 - 图示：



- 举例二：
 - 分析：
 - 有 2 片芯片 $1K \times 4$ 位的 RAM 芯片，组成 $1K \times 8$ 位的存储器；
 - 其中一片的存储位是四位的 D_0 到 D_3 ，对应范围是 $0000 - 1111B = 0 - 15$ ；
 - 变为八位之后、将两个存储器用八位的线连接起来，得到 D_0 到 D_7 对应范围变成 $00000000 - 11111111B = 0 - 255$ ；
 - 对应的图上的连线 $\rightarrow$
 - 两个芯片、八根数据线，从 D_0 到 D_7
 - 一个芯片是数据的高四位，一个芯片是数据的低四位；
 - 图示：



字扩展法

概念：字扩展法

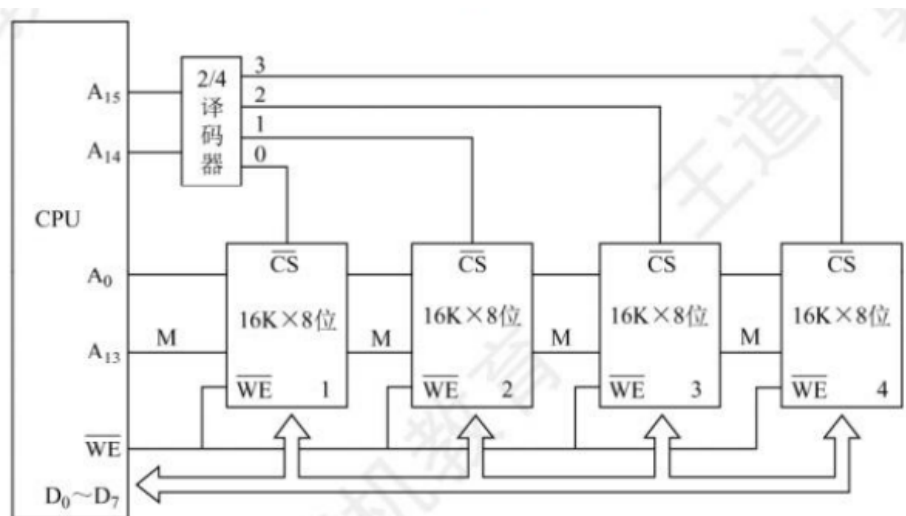
- 核心：
 - 系统地址线位数 > 芯片地址线位数
- 概念：
 - 字扩展是指对**存储字的数量进行扩展**，而存储字的位数满足系统要求；
 - 系统数据线位数等于芯片数据线位数，系统地址线位数多于芯片地址线位数；
- 字扩展的连接方式：
 - 各芯片的地址线与系统地址线的**低位对应相连**；
 - 芯片的数据线和读/写控制线与系统总线相应并联：由系统地址线的高位译码得到各芯片的片选信号；

分析：字扩展后存储芯片的地址范围

- 概念：
 - 基础信息：
 - 图示是 4 片 $16K \times 8$ 位的 RAM 芯片；
 - 其中的 8 位表示数据线的位数，表示取 8 位的数据出来；
 - 数据线（数据）： $D_0 \sim D_7$
 - 4 片 RAM 芯片的数据线 $D_0 - D_7$ 连接在一起；
 - 片选位（模块地址）： $A_{14} \sim A_{15}$
 - 将 $A_{15}A_{14}$ 作为片选信号；
 - $A_{15}A_{14} = 00$ 时选 1 号芯片；
 - $A_{15}A_{14} = 01$ 时选 2 号芯片；
 - $A_{15}A_{14} = 10$ 时选 3 号芯片；
 - $A_{15}A_{14} = 11$ 时选 4 号芯片；
 - 地址线（片内地址）： $A_0 \sim A_{13}$
 - 第一片：
 - 最低地址：0000, 0000, 0000, 0000
 - 最高地址：0011, 1111, 1111, 1111

- 第二片：
 - 最低地址：0100,0000,0000,0000
 - 最高地址：0111,1111,1111,1111
- 两外两片以此类推；

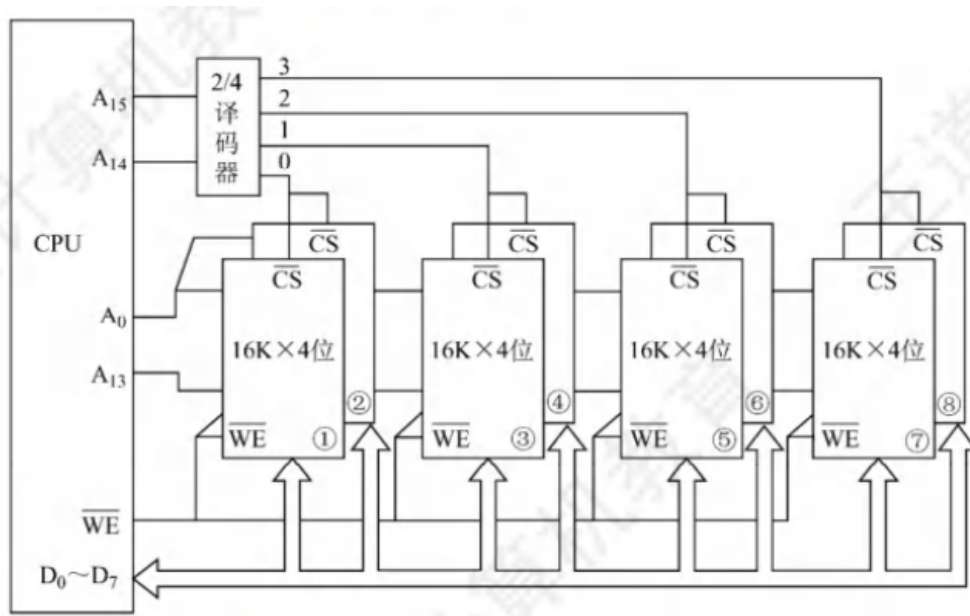
• 图示：



字位扩展法

概念：字位扩展法

- 概念：
 - 字位同时扩展是前两种扩展的组合；
 - 这种方式既增加**存储字的数量**，又增加**存储字长**；
- 连接方式：
 - 将位扩展的芯片作为一组，各组的连接方式与位扩展的相同；
 - 由系统地址线高位译码产生若干片选信号，分别接到各组芯片的片选信号；
- 分析：
 - 基本信息：
 - 用 8 片 16K*4位 的 RAM 芯片组成 64K*8位 的存储器；
 - 扩展：
 - 每两片构成一组 16K*8 位的存储器：位扩展；
 - 共四组构成 64K*8位 的存储器：字扩展；
 - 片选：
 - $A_{15}A_{14} = 00$ 时选第一组芯片；
 - $A_{15}A_{14} = 01$ 时选第二组芯片；
 - $A_{15}A_{14} = 10$ 时选第三组芯片；
 - $A_{15}A_{14} = 11$ 时选第四组芯片；
- 图示：



存储芯片的地址分配与片选

概念：片选与字选

- 片选：
 - CPU 要实现对存储单元的访问，首先要选择存储芯片，即进行片选；
- 字选：
 - 在选定的芯片中**选择具体的存储单元**，以进行数据的读/写，即进行字选；
 - 芯片内的字选通常是由 CPU 送出的 N 条低位地址线完成；
 - N 由片内存储容量 2 决定
 - 片选信号的产生方法分为线选法和译码片选法；

线选法

概念：线选法

- 概念：
 - 线选法用**除片内寻址外的高位地址线**直接连接至各个存储芯片的片选端；
 - 当某位地址线信息为"0"时，就选中与之对应的存储芯片；
 - 这些片选地址线每次寻址时**只能有一位有效**，不允许同时有多位有效，这样才能保证每次只选中一个芯片（或芯片组）；
- 举例：
 - 假设有 4 片 $2K \times 8$ 位的存储芯片使用线选法 $\rightarrow$ 构成 $8K \times 8$ 位的存储器；
 - 片选信号如下：

芯片	$A_{14} \sim A_{11}$
0#	1 1 1 0
1#	1 1 0 1
2#	1 0 1 1
3#	0 1 1 1

- 低位用于片内寻址，高位用于片选；

- 特点：
 - 优点：
 - 不需要地址译码器，线路简单；
 - 缺点：
 - **地址空间不连续**，选片的地址线必须分时位低电平，不能充分利用系统的存储器空间；
 - 即：地址资源的浪费；

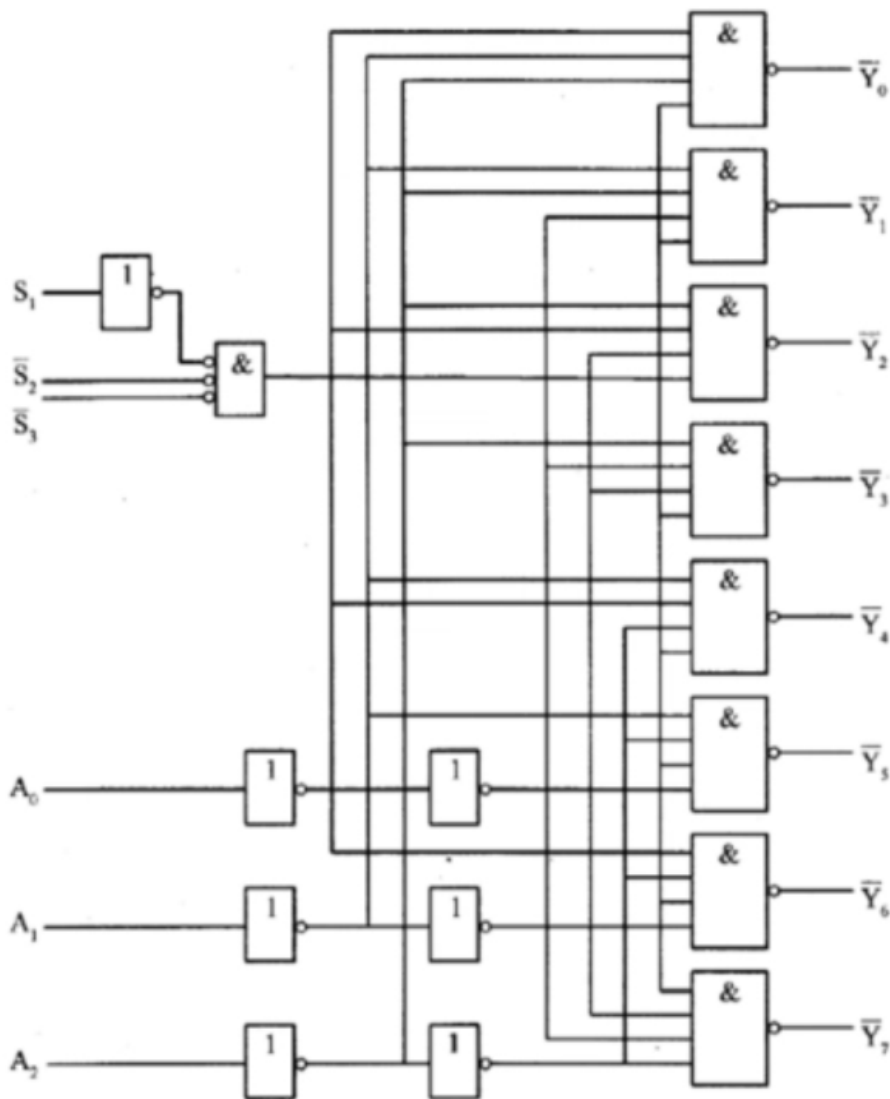
译码片选法

概念：译码片选法

- 概念：
 - 用除片内寻址外的**高位地址线通过地址译码器产生片选信号**；
- 举例：
 - 当前地址线为 16 位，数据线为 8 位；
 - 用 8 片 8K*8位 的存储芯片组成 64K*8位 的存储器 -> 需要 8 个片选信号；
 - 若采用线选法，除去片内寻址的 13 位地址线，只剩下 3 位，无法产生 8 位片选信号；
 - 此时可以使用译码片选法：用一片 74LS138 作为**地址译码器**，高 3 位用于片选；

补充：使用 74LS138 进行 3-8 译码

- 概念：
 - 74LS138 译码器可将一个三位二进制输入转换为八个输出使能信号；
 - 通过设置合适的输入信号，74LS138 译码器可以根据输入的二进制编码来选择相应的输出线激活；
- 举例：74LS138 与数码管
 - 将 74LS138 的输出与七段数码管的对应段连接，根据输入信号的不同，通过译码器的输出控制各段的亮灭，实现数字显示；
- 举例：74LS138 与地址译码器
 - 地址译码器用于将特定的地址编码映射到对应的设备或存储器单元，74LS138 译码器可用作地址解码器的基本构件；
- 图示：



存储器与 CPU 连接

概述：存储器的 CPU 的连接

- 连接主要分为 5 个方面，首先是确定存储芯片，在确定存储芯片后，进行存储器和 CPU 的各个线路的连接：

第一步：合理选择存储芯片

- 概念：
 - 主要指存储芯片的类型 (RAM 或 ROM) 和数量的选择；
 - 在考虑芯片数量时，要尽量使连线简单、方便
- 选择策略：
 - 通常选用 ROM 存放系统程序、标准子程序和各类常数，RAM 则是为用户编程而设置的；

第二步：地址线的连接

- 概念：
 - 存储芯片的容量不同，其地址线数也不同，而 CPU 的地址线数往往比存储芯片的地址线数要多；

- 通常将 CPU 地址线的低位与存储芯片的地址线相连，以选择芯片中的某一单元 (字选)，这部分的译码是由芯片的片内逻辑完成的；
- 而 CPU 地址线的高位则在扩充存储芯片时使用，用来选择存储芯片 (片选)，这部分译码由外接译码器逻辑完成；
- 举例：
 - 假设 CPU 的地址线位 16 位，即 $A_0 \sim A_{15}$ ，但此时 $1K * 4$ 位的存储芯片仅有 10 根地址线；
 - 此时可以将 CPU 的地址线的 $A_0 \sim A_9$ 和存储芯片的地址线相连；

第三步：数据线的连接

- CPU 的数据线数与存储芯片的数据线数不一定相等，在相等时可直接相连、在不等时必须对存储芯片扩位，使其数据位数与 CPU 的数据线数相等；

概念四：读/写命令线的连接

- CPU 读/写命令线一般可直接与存储芯片的读/写控制端相连，通常高电平为读，低电平为写；
- 有些 CPU 的读/写命令线是分开的 (读为 RD，写为 WE，均为低电平有效)；
- 此时 CPU 的读命令线应与芯片的允许读控制端相连，而 CPU 的写命令线则应与芯片的允许写控制端相连；

概念五：片选线的连接

- 片选线的连接是 CPU 与存储芯片连接的关键，存储器由许多存储芯片叠加而成，哪一片被选中完全取决于该存储芯片的片选控制端 CS 是否能接收到来自 CPU 的片选有效信号；
- 片选有效信号与 CPU 的访存控制信号 MREQ (低电平有效) 有关；
- 因为只有当 CPU 要求访存时，才要求选中存储芯片；
- 若 CPU 访问 I/O，则 MREQ 为高，表示不要求存储器工作；