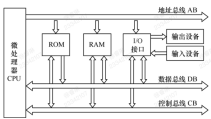


第一章 微机概述

- 地址引脚数**：为 **16** 条，可以寻址 $2^{16} = 64\text{KB}$ 的存储单元。 $2^{10} = 1\text{K}$ ， $2^{20} = 1\text{M}$ ， $2^{30} = 1\text{G}$ ；b 表示位，**B** 表示字节。
- 补码**：补码为原码取反加 1，最高位 **1** 表示负数，**0** 表示正数。例：[-89] 补 = 10100111。十六进制数后面加 **H**，否则默认为十进制。

微处理器	微型计算机中用 CPU 表示，由一片或几片大规模集成电路组成，具有 运算和控制 功能的中央处理器部件。
微型计算机	以 微处理器 为核心，配上 存储器 、 输入输出接口电路 及 系统总线 组成，又称 主机 。
微型计算机系统	以 微型计算机 为中心，配以外围设备、电源和辅助电路（硬件系统），以及指挥工作的 软件系统 。
软件系统	包括： 系统软件 （操作系統、编译器、数据库等）和 应用软件 （Office、微信等）。

●**典型硬件微系统结构**：计算机的经典结构为冯·诺依曼结构（存储程序计算机结构），特点包括：由**运算器、控制器、存储器、输入设备和输出设备**五个基本部分组成；程序和数据以**二进制代码形式**存放在**存储器**中，位置由地址指定，地址码也是二进制；**控制器**根据**存储器中的指令序列（程序）**工作，并由**程序计数器（PC）**控制指令执行，具有判断能力，可根据计算结果选择不同动作流程。**微处理器**是执行指令的核心部件，包含**运算器和控制器**。**存储器**用于存储当前正在使用的程序和数据。**I/O 设备**和**接口电路**实现微处理器与外部设备的连接，如**显示器接口**、**硬盘接口**等。**系统总线**连接微处理器和其他部件，分为**地址总线**、**数据总线**和**控制总线**，分别传输地址、数据和控制信息。



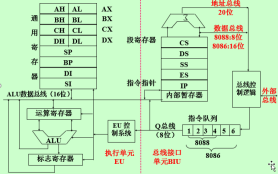
- 数制与计算**
 - 有符号数的机器表示：**真值**指带正负号的实际数值（如 +5，−5），而**机器数**是计算机内部存储的编码形式。有符号数常用三种标准编码：
 - 原码：[X]_原
 - 反码：[X]_反
 - 补码：[X]_补

- 正负数的编码差异**：**正数**：三种表示法完全一致，即 [X]_原 = [X]_反 = [X]_补，符号位为 0，数值位直接表示绝对值。**负数**：三种表示法存在差异，所有区别仅体现在负数的编码规则上。
- 补码的主导地位**：**现代计算机均采用补码**，补码便于硬件实现加法规则，成为实际工程标准。
- 补码加法运算规则**：**核心思想：减法加法，即** $X - Y$ **可转化为** $X + (-Y)$ ，简化硬件设计。
 - 加法： $[X + Y]_{\text{补}} = [X]_{\text{补}} + [Y]_{\text{补}}$
 - 减法： $[X - Y]_{\text{补}} = [X]_{\text{补}} - [Y]_{\text{补}} = [X]_{\text{补}} + [-Y]_{\text{补}}$

符号位参与运算：补码加法时，符号位与数值位同等对待，全部参与二进制加法（逢二进一）。

- 补码求负（变补规则）**： $[-X]_{\text{补}}$ 的求法：将 $[X]_{\text{补}}$ 的**所有位（包括符号位）按位取反，再加 1。即“全字取反加一”。**注意**：**包括符号位**，不区分符号和数值位。**

第二章 微机结构



- 8086/8088 对比**：**8086**：16 位外部数据总线；**8088**：8 位外部数据总线（准 16 位），为兼容 8 位外设，两者均为 **20 位** 地址线（寻址 **1MB**），I/O 寻址 **16 位（64KB）**。

BIU（总线接口单元） 负责**取指、总线操作、物理地址形成**。含指令队列、段寄存器（CS/DS/ES/SS）、IP、地址加法器。
EU（执行单元） 负责译码、执行、运算。含 ALU、通用寄存器、标志寄存器、控制电路。
BIU 取指与 EU 执行重叠，实现流水线雏形，提高总线利用率。

● 物理地址计算 ：	$PA = \text{段基址} \times 10H + \text{偏移地址}$ 。段基址由 段寄存器 （16 位）左移 4 位提供，偏移地址由 IP 或 EU 计算。
指令队列	8086 为 6 字节 ，8088 为 4 字节 。FIFO 原则。当队列空出 2 字节（8086）或 1 字节（8088）时，BIU 自动取指。执行跳转/调用/返回时 清空 。
通用寄存器	AX（累加器），BX（基址），CX（计数），DX（数据）。可拆分为 H/L 8 位使用。
指针与变址	SP（堆栈指针），BP（基址指针），SI（源变址），DI（目的变址）。

第三章 80x86 指令系统

寻址方式 需要看清楚问的是源操作数还是目标操作数的寻址方式，如果是字操作数，要写两个单元的地址。

固定寻址	如 AAA
立即数寻址	操作数直接包含在指令中。举例 MOV AL,15H MOV AX,1234H
寄存器寻址	操作数在寄存器中。例： MOV AX, BX
存储器寻址	比较复杂，见下文详细说明。

- 存储器寻址**：当执行单元 **EU** 需要读/写位于存储器的操作数时，便根据指令给出的寻址方式，由 **EU** 先计算出操作数的有效地址 **EA**（偏移地址），并请求 **BIU**；同时请求 **BIU** 执行一个总线周期，**BIU** 将某个段寄存器的内容左移 4 位，加上 **EU** 送来的有效地址 **EA** 形成 20 位的物理地址，然后执行总线周期，读/写指令所需的操作数。有效地址 **EA**（偏移地址）的值要根据指令所采用的寻址方式计算得出。计算 **EA** 的公式为：

$$EA = \text{基址值} \left\{ \begin{matrix} BX \\ BP \end{matrix} \right. + \text{变址值} \left\{ \begin{matrix} SI \\ DI \end{matrix} \right. + \text{位移量} \left\{ \begin{matrix} 0 \\ 8 \\ 16 \end{matrix} \right.$$

直接寻址	操作数的 EA 由指令直接给出，段地址默认为数据段 DS ，其它数据段应在指令中用段前缀指出。这种寻址方式的指令执行速度较快，主要用于存取位于存储器中的简单单元。例： MOV AX, [1234H]
间接寻址	操作数在存储器中， 存储单元的有效地址由寄存器指出 。 BX、SI、DI —默认数据段 DS BP —默认堆栈段 SS 。 注意 ：间接寻址的地址寄存器只能是 BX、BP、SI、DI ，不能是其它寄存器。指令中地址寄存器要加方括号，如： [BX] 。根据所采用的地址寄存器的不同，间接寻址方式又可分为以下 3 种：

基址寻址	操作数的有效地址由 基址寄存器（BX 或 BP） 的内容和指令中给出的地址位移量（0 位、8 位或 16 位）之和来确定。 $EA = BX/BP + 0\text{位}/8\text{位}/16\text{位位移量}$ 。例： MOV AX, [BX] ，假设 $BX = 1122H$ ， $DS = 3000H$ ，则 $PA = 30000H + 1122H = 31122H$ ， $30000H + 1123H = 31123H$ 。假设 $[31122H] = 34H$ ， $[31123H] = 56H$ ，则指令执行后， AX = 5634H 。 例：假设 $BETA = 8$ ， $DS = 6000H$ ， $BX = 5000H$ ，则 MOV AL, [BX+8] MOV AL, [BX+8] ；“8”是偏移地址位移量，不是乘的倍数 MOV AL, [BX+BETA] MOV AL, BETA [BX] 。注：上面 4 条指令是等价价。 $EA = 5000H + 8 = 5008H$ ， $PA = DS \times 16 + 5008H$ ， $60000H + 5008H = 65008H$ 。假设 $[65008H] = 68H$ ，执行后， AL=68H 操作数的有效地址由变址寄存器（ SI 或 DI ）的内容与指令中给出的地址位移量（0 位、8 位或 16 位）之和来确定。 $EA = SI/DI + 0\text{位}/8\text{位}/16\text{位位移量}$ 。例： MOV BETA[DI], AX MOV BX, [SI+BETA] 操作数的有效地址 EA 为以下三部分之和：基址寄存器（ BX 或 BP ）的值、变址寄存器（ SI 或 DI ）的值、指令中的地址位移量（0 位、8 位或 16 位）。 $EA = [BX/BP] + [SI/DI] + 0/8/16\text{位偏移量}$ 。例： MOV BX, [BX+SI] MOV [BX][DI], AX MOV AX, [BX][SI] MOV AX, 8[BX+SI] MOV AX, [BX+SI+8] MOV AX, ES:[BX+SI+10H] “ES:”一段前缀，指定操作数在附加段，源操作数 $PA = ES \times 16 + BX + SI + 10H$ ， $ES \times 16 + BX + SI + 10H + 1$ MOV AX, [BP+SI+20H] BP —操作数在堆栈段，源操作数 $PA = S \times 16 + BP + SI + 20H$ ， $S \times 16 + BP + SI + 20H + 1$
-------------	---

第四章汇编语言程序设计基础

注意：如果有图片，尽量不要插入大图。建议手绘或者用 TikZ，或者只是文字描述流程。

第五章微机总线技术

- 概念**：● **总线定义**：它是用来**连接各部件的一组通信线**。换言之，总线是一种在多于两个模块（设备或子系统）间传递信息的公共通路。为在各模块之间实现信息共享和交换，总线由传递信息的物理介质以及一套管理信息传输的协议构成。
- **三态逻辑**：除了逻辑 0 和逻辑 1 之外，还有第三种状态——高阻态（Hi-Z）。高阻态相当于断开电路，使得该信号线不对总线产生影响，从而允许多台设备驱动总线。
- **微机总线分类**：按功能可分为**系统总线（内总线）**和**通信总线（外总线）**；按信号传输方式可分为**串行总线**与**并行总线**。

片内总线	又称元件级总线，位于 集成电路芯片内部 ，连接芯片内各功能单元
片总线	连接同一块插件板（PCB）上各个芯片的总线。常见如 IIC 总线、SPI 总线 等。
内总线	又称系统总线、板级总线或微机总线，用于微机系统中各插件之间的信息传输。是我们主要研究的总线类型
外总线	又称通信总线，常见如 USB、PCIe、SATA 等。

双向数据总线，是一个总线周期内可以传送的数据位数，
单向地址总线（通常称为三态逻辑控制），位宽如**16、20 等**，位宽代表了芯片的寻址能力（20 位地址线意味着 2^{20} 的寻址能力）。
控制总线 传送定时与控制信号以协调各部件动作（如读/写、中断请求、复位等），控制总线一般**不采用三态逻辑**。

总线宽度	即数据总线的位数，决定了一个 总线周期内可并行传输的数据位数 。
总线频率	总线的工作时钟频率（单位为 Hz），表示 每秒总线可执行的数据传输次数 。
传输速率	衡量系统性能的最终指标，表示 单位时间内总线可传输的数据总量 （单位为 MB/s）。三者关系：传输速率 =（总线宽度/8）× 总线频率。

常用芯片	● 常用缓冲器芯片 ： 74LS245 （双向总线收发器）、 74LS244 （单向三态缓冲器）、 74LS373 （八位透明锁存器）、 74LS138 （3-8 线译码器）、 74LS157 （4-1 线数据选择器）等。
	● 74LS244 ： 单向三态缓冲器 ，8 路分为两组 两个使能端均为低电平有效 。适用于数据总线的缓冲与隔离。
1A1–1A4	1Y1–1Y4 （由 1G 控制）
2A1–2A4	2Y1–2Y4 （由 2G 控制）

● 74LS245 ：	又称 收发器 。在 A 端 和 B 端 之间， 并联了两个方向相反的单向三态门 。拥有两个控制信号： E 全局使能信号 ，决定整个双向通道是否工作或处于高阻态。 低电平有效 。 DIR 方向控制信号 ，在通被 E 使能时，选择数据是 $A \rightarrow B$ 还是 $B \rightarrow A$ 传输。
● 74LS373 ：	由两个独立逻辑单元组成：

D 型锁存器	存储/保持数据（由 G 控制） 。使能时，锁存器处于 透明状态 ，输出 1Q 跟随输入 1D 的逻辑电平变化；非使能时，锁存器数据被 锁存 ， 保存上一瞬间 1D 上的逻辑状态 。
三态输出门	负责控制数据是否被驱动到输出总线（由 OE 控制，使能时输出，否则为 高阻态 ）。

最小方式	MN/MX# 连接到高电平（ V_{CC} ）时激活。CPU 自行产生总线控制信号（直接输出控制信号，如 INTA#、ALE、RD#、WR#、M/I/O# 等）。适用于 单处理器系统 ，CPU 直接控制总线，无需外部总线控制逻辑。
最大方式（Maximum Mode）	MN/MX# 连接到低电平（GND）时激活。CPU 不直接产生控制信号，而是输出编码后的状态信号。引脚 24-31 被重新定义。

SO#、S1#、S2#	输出状态，需外接 8288 解码生成 RD#、WR#、ALE 等
RQ#/GTO#、RQ#/GT1#	请求/准许信号，用于总线仲裁
LOCK#	总线锁定信号，保证原子操作，适用于多处理器系统

总线时序	
名词解释	一个基本的总线周期需要至少 4 个时钟周期 （即 4 个 T 状态，通常标记为 T1、T2、T3、T4）。

时序 信息在总线上的出现不仅要有**空间顺序**，还要有**严格的顺序和准确的时间**。这种时间和逻辑上的配合关系被称为**时序**。

时钟 由时钟发生器产生的具有固定频率和占空比的脉冲序列。是整个微机系统的时间基准，所有部件的动作都必须与此信号同步。

主频 时钟的频率，衡量 CPU 处理速度的一个指标。

主频的倒数（ $T = 1/f$ ），CPU 通过总线对存储器或 I/O 端口进行**一次完整的访问（读或写）**所需的时间。

常用信号	除了 CLK 外的常用控制信号包括：
M/I/O	存储器/I/O 控制信号 ，用于区分 CPU 是访问存储器（ $M/I/O = 1$ ）还是访问 I/O 端口（ $M/I/O = 0$ ）。
DT/R#	数据发送/接收信号 ，在 T_1 周期用于指示 245 等数据缓冲芯片的传输方向。
RD#	读控制信号 。 RD 信号为低电平时，表示 8086 CPU 执行读操作。在 DMA 方式时， RD 处于高阻态。
WR#	写控制信号 （输出，三态）。当 8086 CPU 对存储器或 I/O 端口进行写操作时， WR 为低电平。
ALE	地址锁存允许信号 （输出）。8086 CPU 在总线周期的第一个时钟周期内发出的正脉冲信号，其下降沿用来把地址/数据总线（ $AD_{15} \sim AD_0$ ）以及地址/状态总线（ $A_{19}/S_6 \sim A_{16}/S_3$ ）中的地址信息锁存入地址锁存器中。
DEN#	数据缓冲器使能信号 。
BHE#/S7	数据高允许/状态信号 S_7 信号。分时复用的双重总线，在总线周期开始的 T_1 周期，作为总线高半部分允许信号，低电平有效。在总线周期的其他 T 周期，该引脚输出状态信号 S_7 。在 DMA 方式下，该引脚为高阻态。

BHE#	A0	总线高允许信号，低电平有效。用于控制数据总线高 8 位（D15–D8）的读写。地址最低位，访问偶地址时，A0 必然为 0。
-------------	-----------	---

奇偶地址访问方式 ：	
BHE# = 0， A0 = 0	访问 16 位偶地址 （全字访问），数据线 D15–D0 全部有效。
BHE# = 1， A0 = 0	访问 8 位偶地址 ，仅 D7–D0 有效（低字节）。
BHE# = 0， A0 = 1	访问 8 位奇地址 ，仅 D15–D8 有效（高字节）。
BHE# = 1， A0 = 1	无效访问（不选中任何字节）。

复位、启动时序	● RESET 信号 ：需保持高电平至少 4 个时钟周期 。有效期间总线处于 高阻态 （浮空）。
----------------	---

CS：IP	FFFFH：0000H （启动地址 FFFF0H ）
标志位	1F （允许中断），其余清零
DS/SS/ES	0000H
指令队列	清空

● **复位执行流**：CPU 复位后 **CS:IP = FFFFH:0000H**（物理地址 **FFFF0H**）。从该处取出 **JMP** 指令，跳转至低地址空间的系统初始化程序（如 BIOS）。**原因**：从 **FFFF0H** 到内存顶端 **FFFFFFH** 仅**16 字节**，空间极小，无法容纳完整程序。

最小方式总线读操作时序 8086 CPU 需要与外部存储器或 I/O 端口交换数据，或者需要填充指令队列时，必须执行一个总线周期（四个或更多时钟周期，考虑到 T_W 的存在）。

T1	确定 M/\overline{IO} 状态；输出地址信息； ALE 发出正脉冲（下降沿锁存地址）； DT/R 置低（接收）。
T2	$AD_{15} \sim AD_0$ 进入高阻态； \overline{RD} 与 \overline{DEN} 有效（低电平）；输出状态信号 $S_3 \sim S_7$ 。
T3/Tw	外设将数据驱动至总线。在 T_3 前沿采样 READY 信号，若为 0 则插入等待周期 T_W 。
T4	在 T_4 前沿（即 T_3/T_W 结束处）采样数据；撤销 \overline{RD} 、 \overline{DEN} ，总线周期结束。

- **关键状态位**： S_4 、 S_3 组合指示当前段寄存器：**00:ES, 01:SS, 10:CS, 11:DS**。 S_5 指示中断允许标志 **IF**。

最小方式总线写操作时序	基本和读操作类似，区别在于数据传输方向相反。
--------------------	------------------------

T1	输出地址； ALE 发出正脉冲（锁存地址）； DT/R 置高（发送）。
T2	WR 信号变低（有效）；CPU 直接驱动数据到 AD 总线 （不进入高阻态）。
T3/Tw	采样 READY 信号；数据在总线上保持稳定，等待外设响应。
T4	\overline{WR} 上升沿触发外设写入数据；撤销 \overline{WR} 、 \overline{DEN} ，周期结束。

- **读写区别**：写操作中 **A0 总线在 T_2 不进入高阻态**，由 CPU 持续驱动；数据在 \overline{WR} 上升沿锁存。

中断时序	当 CPU 接收到外部中断请求（ INTR ）且中断允许标志 IF = 1 时，CPU 会执行一个中断响应操作。这个操作在总线时序上表现为 两个连续的总线周期 。
-------------	--

第一周期	CPU 发出第一个 INTA 负脉冲，用于 中断握手 ，此周期不传输数据。
第二周期	CPU 发出第二个 INTA 负脉冲，8259A 将 中断类型码 送往数据总线 $AD_7 \sim AD_0$ 。
后续动作	CPU 在 T_4 前沿采样类型码，计算向量表地址并跳转至 中断服务程序（ISR） 。

- **注意**：两个周期之间可能插入 T_I （空闲态）以兼容外设速度。

第六章存储器系统

存储器分类与基本指标

按读写功能 读写存储器（RAM）：可读写，数据暂存；只读存储器（ROM）：只能读，存固件/引导程序。

按存储介质 半导体存储器：如 DRAM、ROM，速度快，体积小；磁存储器：如硬盘、磁带，容量大，速度慢。

按存取方式 随机存取存储器（RAM）：存取时间与位置无关；顺序存取存储器：如磁带，存取时间与位置有关。

按信息保存性 易失性存储器：断电丢失，如 RAM；非易失性存储器：断电保存，如 ROM、硬盘、SSD。

内外存区别：	
内存	存放当前运行的程序和数据。特点： 快 、容量小、随机存取， CPU 可直接通过系统总线访问 。通常由半导体存储器（RAM、ROM）构成。
外存	存放非当前使用的程序和数据。特点： 慢 、容量大、顺序/块存取， CPU 不能直接访问 ，需通过 I/O 接口电路 调入内存。如硬盘、U 盘、移动硬盘等。

RAM 类型特点：	
SRAM	静态 RAM，利用 双稳态触发器 存储逻辑 0/1， 无需刷新 ，只要不掉电信息不丢失。集成度低，外围控制电路简单，常用于小容量存储（如 Cache ）。
DRAM	动态 RAM，利用 MOS 管栅极 分电容存储电荷， 需定期刷新 。集成度高，外围控制电路复杂，常用于大容量存储（如 主存 ）。

存储容量	$N \times M$ （字数 \times 字长）
字数 N	单元总数，决定地址线数量 k ($2^k = N$)
字长 M	每单元位数，决定数据线位宽

常用存储芯片：	
6264 (SRAM)	$8K \times 8$ (8KB)，13 根地址线 ($2^{13} = 8K$)，8 根数据线
2114 (SRAM)	$1K \times 4$ ，10 根地址线，4 根数据线（常两片并联组成 8 位）
2764 (EPROM)	$8K \times 8$ (8KB)，13 根地址线，8 根数据线，用于存储固件

- 地址译码：**将 CPU 高位地址信号转换为芯片片选信号 **CS#**。常用 **74LS138**（3-8 译码器）实现。

Bit（位）	最小存储单位，对应硬件中的 双稳态触发器 状态。
Byte（字节）	基本处理单位， 1 B = 8 bits 。
Word（字）	基本处理单位， 1 Word = 2 Bytes 。
常用容量换算	1KB=2 ¹⁰ B，1MB=2 ²⁰ B，1GB=2 ³⁰ B。

基本性能指标	<p>存取时间：从 CPU 发出地址信号到数据有效（读）或写入完毕（写）的时间。</p> <p>存储周期：进行一次完整读写所需的最小时间间隔。</p> <p>存取周期 > 存储周期（内部电路需恢复时间）。</p> <p>$t_{cyc}(R)$：连续两次读最小间隔；$t_{cyc}(W)$：连续两次写最小间隔。</p>
--------	--

计算末尾地址的公式为：

末尾地址 = 首地址 + 存储容量 - 1

注：减 1 是因为地址是从 0 开始计数的，且首地址本身占了一个存储单元。

常见芯片	
通用引脚特点	<p>地址线 $A_0 \sim A_n$：接地址总线 AB，输入信号，决定芯片容量 $N = 2^{n+1}$。</p> <p>数据线 $D_0 \sim D_m$：接数据总线 DB，双向（RAM）或输出（ROM），决定字长 M。</p> <p>片选线 $\overline{CS}/\overline{CE}$：由高位地址译码产生，低电平有效，用于选中该芯片。</p> <p>读写线：读允许 \overline{OE} 接 CPU 的 \overline{RD}；写允许 \overline{WE} 接 CPU 的 \overline{WR}。</p>

6264 组织	地址线 $A_0 \sim A_{12}$ ($2^{13} = 8K$)；数据线 $D_0 \sim D_7$ 。
双片选机	$\overline{CS}1$ （低有效）和 $\overline{CS}2$ （低有效）。选中条件： $\overline{CS}1 = 0$ 且 $\overline{CS}2 = 1$ 。
读写控制	\overline{OE} （输出允许，接系统 \overline{RD} ）； \overline{WE} （写允许，接系统 \overline{WR} ）。

2114 容量与组织	$1K \times 4$ 位（0.5 KB）；地址线 $A_0 \sim A_9$ (2^{10})，数据线 $D_1 \sim D_4$
位扩展	8 位系统需 两片 并联，分别负责高 4 位和低 4 位

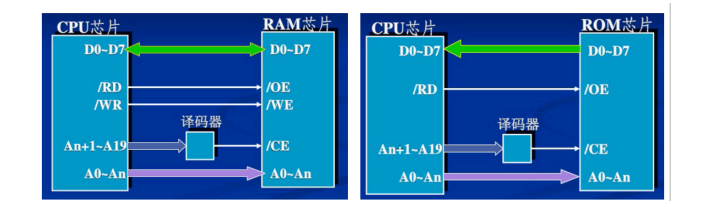
由于只有一个 \overline{WE} 而没有 \overline{OE} ，因此 2114 的读写模式由 \overline{CS} 和 \overline{WE} 共同决定：

写模式	$\overline{CS} = 0, \overline{WE} = 0$ ，数据端口为输入（DIN）
读模式	$\overline{CS} = 0, \overline{WE} = 1$ ，数据端口为输出（DOUT）
待机模式	$\overline{CS} = 1$ ，输出为 高阻态（Hi-Z） ，实现总线上隔离

容量与组织	地址线 $A_0 \sim A_{12}$ ；数据线 $D_0 \sim D_7$ 。与 6264 引脚兼容。
CE#	片选使能，低电平有效。控制芯片激活及低功耗模式。
OE#	输出使能，低电平有效。读操作时打开输出缓冲器。
Pgm#	编程脉冲， 低电平有效 。烧录时施加负脉冲，正常读取时置高。
Vpp	编程电压引脚，烧录时需施加高压（12.5V 或 21V）。

在对芯片进行数据烧录（编程）时，需要在 PGM# 施加特定宽度（如 50ms）的负脉冲，配合 V_{pp} 引脚的高压（通常 12.5V 或 21V），将数据写入**浮栅晶体管**（EPROM 没有 \overline{WE} 引脚，因为它在正常工作时不可写。）

低位地址线用于片内寻址，高位地址线用于片选寻址：



存储器扩展

- 存储器扩展：**当单片芯片容量（字数 N 或字长 M ）不足以满足系统需求时，需通过多片组合进行扩展。

位扩展：增加**字长**，字数不变。 $N \times M \rightarrow N \times (M \times k)$ 。各片**地址线、读写线并联**，数据线分别连接 CPU 数据总线的不同位。

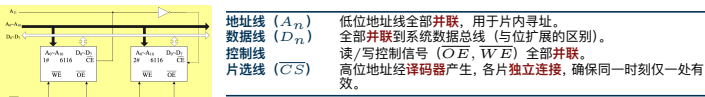
字扩展：增加**字数**（容量），字长不变。 $N \times M \rightarrow (N \times k) \times M$ 。各片**数据线、低位地址线并联**，高位地址线经**译码器**产生片选信号 \overline{CS} 。

位和字节扩展：同时增加字长和字数。通常先通过位扩展组成满足字长要求的“存储组”，再通过字扩展（译码片选）连接各组。

位扩展：当芯片字长小于 CPU 数据总线宽度时，通过多片并联增加**存储字长**。如两片 $1K \times 4$ 的 2114 并联可组成 $1K \times 8$ 存储器。

地址/片选/读写线	全部 并联 。所有芯片接收相同地址，由同一片选信号同时选中，并同步执行读/写操作。
数据线 (D_n)	分段连接 。各芯片的数据端分别连接 CPU 数据总线的不同位段（如一片接 $D_0 \sim D_3$ ，另一片接 $D_4 \sim D_7$ ）。

字节扩展	当芯片字长满足但容量不足时，增加 存储单元总数 （地址空间深度）。
------	--



- 核心逻辑：**通过高位地址译码，将不同芯片映射到系统内存空间的不同地址段（如 $0000H \sim 1FFFFH$ ）。

位和字节扩展：当芯片字长和容量均不足时，需同时进行扩展。

- 芯片数计算：**所需总芯片数 $Z = (M/L) \times (N/K)$ 。其中 M/L 为字扩展组数， N/K 为位扩展组数。L 为单片字长，K 为单片字数。

第一步：位扩展：将 N/K 片并联，地址/读写线并联，数据线分段连接，构成一个字长满足要求的存储组（Bank）。

第二步：字扩展：将 M/L 个存储组级联，数据/低位地址线并联，高位地址经译码器产生各组的片选信号 \overline{CS} 。

地址译码电路设计（片选）	<ul style="list-style-type: none"> 译码电路设计： <ol style="list-style-type: none"> 片内寻址：根据芯片容量 N 确定低位线数 k ($2^k = N$)，如 4KB 需 $A_{11} \sim A_0$。 片选逻辑：高位地址通过译码器（如 74LS138）或逻辑门产生 $\overline{CS}\#$。 范围确定：首地址为高位固定、低位全 0；末地址为高位固定、低位全 1。
--------------	--

CPU 芯片	RAM 芯片
D0-D7	D0-D7
R/D	R/D
W/R	W/R
A0-A19	A0-A19
A0-A0	A0-A0
A0-A0	A0-A0

全译码	CPU 全部地址线 均被利用。低位用于片内寻址， 剩余所有高位 参与译码产生片选信号 $\overline{CS}\#$ 。特点： 地址唯一性 （不重叠），地址空间连续。
部分译码	仅利用部分高位地址线 参与译码。存在未使用的“悬空”地址线（Don't Care）。特点：电路简单，但会导致 地址重叠 （镜像），即一个物理单元对应多个逻辑地址。

- 地址重叠计算：**若有 n 根高位地址线未参与译码，则一个物理单元对应 2^n 个重叠地址。

存储器与 CPU 连接

第七章 I/O 接口

概念

- 端口：**接口电路中用于缓存数据、状态、及控制信息的部件，分为：**数据端口、状态端口、控制端口**。

- 接口电路：**计算机系统中包含多个不同功能的接口电路，**每个接口电路又可能包含 1 个或多个端口**。
- 寻址端口方法：**先找到端口所在的接口电路芯片（片选），在该芯片上找到具体访问的端口（**片内地址**）。若接口中仅有一个端口，则找到芯片即找到端口；若接口中有多个端口，则找到芯片后需再找端口。每个端口地址 = **片选地址（高位地址）** + **片内地址**。

- 8086/8088 的 I/O 端口编址：**采用 **I/O 独立编址方式**；I/O 操作只使用 20 根地址线上的 **16 根（A15 ~ A0）**；可寻址的 I/O 端口数为 $2^{16} = 64K$ (65536) 个；I/O 地址范围是 **0~0FFFFH**。

8259A

- 8259A 特性：功能：**8259A 是一个功能很强的中断扩充和多中断管理芯片，具有**中断扩展、自动提供中断类型码、中断优先级裁决**等中断管理功能。**可编程：**内部有多个寄存器以及功能部件都是可编程的，使用方便。**级联扩展：**单片可连接 8 个中断请求源，多片级联可扩展到 64 级中断。通过编程可设置**中断触发方式、中断类型码、中断屏蔽方式、中断优先级方式、中断结束方式**等。

- 8259A 内部结构：**数据总线缓冲器：三态、双向、8 位寄存器。**读写控制逻辑：**接收 CPU 的读写控制信号。**级联缓冲/比较器：**支持单片或多片级联，主片/从片管理，最多可扩展到 64 级中断。

控制逻辑：向片内各部件发送控制信号，向 CPU 发送中断请求信号 \overline{INT} ，接收 CPU 回送的 $\overline{INTA}\#$ 信号，控制 8259A 进入中断管理状态。

中断请求寄存器（IRR）：8 位寄存器，记录外部中断请求， $\overline{IR}i$ 有请求时 \overline{IRR} 的相应位 \overline{Di} 置 1，中断响应后清除。

中断屏蔽寄存器（IMR）： \overline{IMR} 中 \overline{Di} 位为 1 时禁止对应 $\overline{IR}i$ 请求，为 0 时允许。

优先级判选器：对 \overline{IRR} 中未屏蔽的中断进行优先级比较，选出当前优先级最高的中断请求。

中断服务寄存器（ISR）：记录 CPU 当前正在服务的中断标志， $\overline{IR}i$ 请求响应时 \overline{ISR} 相应位置 1，复位由中断结束方式决定。

第八章常用接口技术

8255A

- 8255 功能介绍：**8255 是一种可编程的并行通信接口芯片，可用于 CPU 和外设之间进行并行数据传输。内部有**三个 8 位的数据端口**，有三种工作方式。端口号的 $A0A1$ 为 00、01、10 分别表示读写 **A、B、C 口**；11 表示只写控制寄存器。

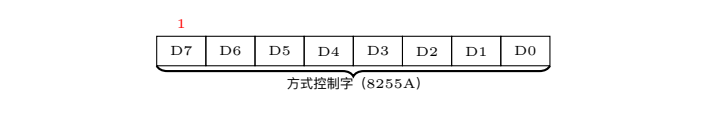
并行通信：	指 多位数据同时进行传送 的方式，其特点是 传输速度快 。
方式 0	基本的输入/输出方式，A/B/C 口均可用，C 口仅需设置方向。
方式 1	选通的输入/输出方式，A/B 口可用，支持中断方式传输，C 口部分引脚用于控制和中断信号。
方式 2	选通的双向传输方式，仅 A 口可用，支持双向传输和中断，C 口部分引脚用于控制和中断信号。
A 口	可工作在方式 0、1、2，方式 2 时支持双向传输。
B 口	可工作在方式 0，仅支持单向传输。
C 口	只能工作在方式 0，高四位和低四位可分开使用，方式 1/2 时部分引脚用于控制和中断。

方式 1、2：选通输入输出方式，可以**中断方式**传输，且 **C 口**会固定的引脚用作**控制联络信号和中断请求信号**。

仅 **A 口**工作在**方式 2** 时，可以**双向传输**，A 口工作在方式 0、1 及 B、C 口只能单向传输。

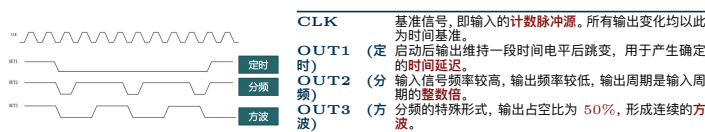
- 控制字：**控制字分为**端口的方式选择控制字**（可使 8255 的 3 个数据端口工作在不同的方式）和 **C 口的按位位置**

和**复位控制字**（可使 C 口的任意一位位置和复位）。**控制字送入的端口为最后一个端口**



8253

- 概述：**8253 是一种**可编程的计数器/定时器**接口芯片，最高计数频率为 **2MHz**，可用于产生各种**定时波形**，也可用于**对外部事件计数**。内部有三个**独立的 16 位减一计数器**（互不干扰，支持二进制（Binary）或 二-十进制（BCD 码）计数），通过设置控制字，各计数器可以工作于 **6 种工作方式**。



CLK	基准信号，即输入的 计数脉冲源 。所有输出变化均以此为时间基准。
OUT1（定时）	启动后输出维持一段时间电平后跳变，用于产生确定的 时间延迟 。
OUT2（分频）	输入信号频率较高，输出频率较低，输出周期是输入周期的 整数倍 。
OUT3（方波）	分频的特殊形式，输出占空比为 50% ，形成连续的 方波 。

• 系统总线接口（连接 CPU）：	
$\overline{D}_7 \sim \overline{D}_0$	双向数据总线，用于 CPU 读写控制字或计数值。
\overline{CS}	片选信号，低电平有效，由地址译码电路产生。
$\overline{RD}/\overline{WR}$	读/写信号，低电平有效，控制数据传输方向。
A_1, A_0	端口选择：00、01、10 分别对应计数器 0、1、2；11 为控制寄存器。

• 计数通道信号（连接外设）：	
$\overline{CLK}0_{-2}$	时钟输入，计数脉冲源， 下降沿 触发减 1 操作。
$\overline{GATE}0_{-2}$	门控输入，用于启动、停止或暂停计数过程。
$\overline{OUT}0_{-2}$	输出信号，计数完成或达到条件时输出特定波形。

- 控制寄存器：**只能进行**写操作**，不能读。CPU 通过向此寄存器写入“控制字”来设定各通道的 $\overline{工作方式}$ 。

- 计数通道结构：**包含三个独立的计数通道（0、1、2），内部由以下三部分配合实现“预置 \rightarrow 减计数 \rightarrow 读出”逻辑：

初值寄存器（CR） **16 位**。用于预置计数的起始值。

执行部件（CE） **16 位减法计数器**。在 \overline{CLK} 信号驱动下进行减 1 计数。

输出锁存器（OL） **16 位**。用于锁存当前计数值供 CPU 读取，不影响计数进行。

8253 没有状态寄存器，CPU **无法直接读取**状态寄存器来获知当前工作状态或回读控制字。这与具有状态回读功能的 8254 不同。

- 端口地址分配：**8253 占用 4 个 I/O 端口。CPU 通过 A_1, A_0 选择：

00	计数通道 0
01	计数通道 1
10	计数通道 2
11	控制字寄存器

• 实例 1：连续地址（8088）：	• 实例 2：偶地址对齐（8086）：
设定	地址范围：0380H ~ 0383H。地址连续（步长为 1）。
分析	0380H (... 000) $\rightarrow A_1A_0 = 00$ ；0381H (... 001) $\rightarrow A_1A_0 = 01$ ；0382H (... 010) $\rightarrow A_1A_0 = 10$ ；0383H (... 011) $\rightarrow A_1A_0 = 11$ 。
连接	CPU 的 A_1, A_0 直接连接到 8253 的 A_1, A_0 。每一个逻辑地址都对应一个物理端口，无地址间隙。

- 读/写逻辑真值表：**

片选前缀：所有操作必须在 $\overline{CS} = 0$ 时有效。

写 ($\overline{WR} = 0$)：根据 A_1, A_0 写入通道 0/1/2 的计数初值寄存器，或写入控制字寄存器。

读 ($\overline{RD} = 0$)：根据 A_1, A_0 读取通道 0/1/2 的当前计数值。注意：**控制字寄存器只写不读**。

- 8 位总线与 16 位计数器接口：**8253 内部的计数初值寄存器（CR）和输出锁存器（OL）都是 16 位的，但外部数据总线 ($\overline{D}_7 \sim \overline{D}_0$) 只有 8 位。必须通过**两次 I/O 操作**来完成一个 16 位数据的传输，由控制寄存器中的控制字来指定读写顺序（如：先读/写低 8 位，再读/写高 8 位）。

计数器选择	读写操作	工作方式	计数制
$\overline{SC}1, \overline{SC}0$	$\overline{RL}1, \overline{RL}0$	$\overline{M}2, \overline{M}1$	$\overline{M}0$
D7	D6	D5	D4
D3	D2	D1	D0
8253 控制字格式			
$\overline{SC}1, \overline{SC}0$	计数器选择。因共用端口（3 个计数器共用一个控制端口），需指定目标通道。		
$\overline{RL}1, \overline{RL}0$	00: CNT0；01: CNT1；10: CNT2；11: 非法。		
	读写格式。定义 CR/OL 的读写位宽及顺序。		
	00: 锁存命令（不停止计数读取）；01: 仅低 8 位；10: 仅高 8 位；11: 先低后高（16 位常用）。		
$\overline{M}2 \sim \overline{M}0$	工作方式。设定方式 0 ~ 5。		
\overline{BCD}	计数制。0: 二进制（FFFFH）；1: BCD 码（9999）。		

● **实例 1: 8 位读写模式:** 需求: 计数器 0, 方式 2, 仅使用低 8 位, 初值 100, 二进制计数。地址: $70H \sim 73H$ 。

SC **00** (选择计数器 0)
RL **01** (只读/写低 8 位)
M **010** (方式 2, 分频器)
BCD **0** (二进制计数)
控制字 00010100*B* = **14H**

汇编实现:

- MOV AL, 14H
- OUT 73H, AL ; 写控制字
- MOV AL, 100 ; 初值 100
- OUT 70H, AL ; 写低 8 位

● **锁存命令:** 解决在计数器运行过程中读取数值不稳定的问题。通过向控制寄存器写入 $RL_1RL_0 = 00$ 的控制字, 将当前计数值复制到 **输出锁存器 (OL)** 中保持不变, 而 **执行部件 (CE)** 继续计数。

● **锁存读出示例:**

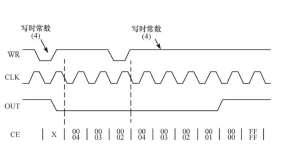
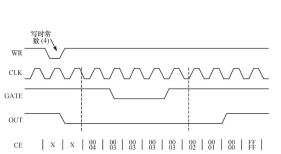
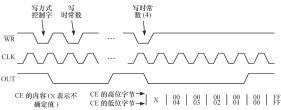
锁存机制	向控制口发送控制字, 其中 $RL_1RL_0 = 00$ 。8253 接收后锁存当前值, 不影响内部计数。 读取计数器 0 当前值: $SC = 00, RL = 00$, 其余位无关, 控制字为 00H 。 1. MOV AL, 00H 2. OUT 73H, AL (向控制口发送锁存命令) 3. IN AL, 70H (从数据口读取低 8 位) 4. IN AL, 70H (从数据口读取高 8 位) 读出操作必须符合初始化时设定的 RL 格式 (如 16 位模式需连续两次)。
实例分析 代码流程	
注意	

● **相关名词解释:**

CLK 脉冲	指 CLK 引脚上的信号单元。在计数过程中, 每一个 CLK 脉冲的 下降沿 到来时, 计数器减 1。 与“计数通道”同义 指通过指令写入计数器的值, 等同于 初值 。输出波形的周期或延时由该值决定: 时间 = 初值 $\times T_{CLK}$ 。
计数器 时常数	

方式 0 ● **功能定义:** 主要用于**定时中断**。给定时间 t_0 , 到达后输出信号通知 CPU。

初始化	写入方式控制字后, 输出引脚 OUT 变为 低电平 。 GATE 是硬件门控信号, 在方式 0 中充当“计数使能开关”。	
计数过程	写入初值后开始减 1 计数, 期间 OUT 保持 低电平 。	暂停计数, 计数器保持当前值不变, 忽略 CLK 脉冲。
计数结束	当计数值减到 0 时, OUT 立即跳变为高电平 。 1 产生的 上升沿信号 通常连接 CPU 中断请求引脚。	允许计数, 计数器正常工作。



● **方式 0 编程示例:** 已知端口 $40H \sim 43H$, 计数器 0, 方式 0, 初值 1500, 二进制计数。

控制字 SC=00, RL=11, M=000, BCD=0 \rightarrow **30H**
时常数 1500 = 05*DCH* (低位 **DCH**, 高位 **05H**)

汇编实现:

- MOV DX, 43H ; 指向控制口
- MOV AL, 30H ; 写控制字
- OUT DX, AL
- MOV DX, 40H ; 指向计数器 0 数据口
- MOV AX, 1500 ; AX = 05DCH
- OUT DX, AL ; 写低 8 位 (DCH)
- MOV AL, AH ; 取高 8 位 (05H)
- OUT DX, AL ; 写高 8 位

● **核心逻辑:** 写初值 \rightarrow OUT 低 \rightarrow 计数 $\rightarrow 0 \rightarrow$ OUT 高。**定时时长** = $(N + 1) \times T_{CLK}$ 。可通过 GATE 硬件信号暂停, 或重写初值延长定时。

方式 1 ● **功能定义:** **单脉冲形成**。即产生一个宽度可控的负脉冲。

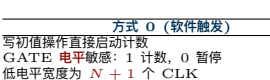
触发源	硬件触发 。区别于方式 0 的软件写入触发, 方式 1 必须由 GATE 引脚的 上升沿 (由低电平变高电平) 来触发。
脉冲宽度	输出负脉冲的宽度为 $N \times T_{CLK}$, 其中 N 为预置的初值。

● **实例 2: 16 位读写模式:** 需求: 计数器 1, 方式 1, 16 位 (先低后高), 初值 1234, BCD 码, 地址: $70H \sim 73H$ 。

SC **01** (选择计数器 1)
RL **11** (先低 8 位后高 8 位)
M **001** (方式 1, 单稳态)
BCD **1** (BCD 码计数)
控制字 01110011*B* = **73H**

汇编实现:

- MOV AL, 73H
- OUT 73H, AL ; 写控制字
- MOV AX, 1234H ; BCD 码初值
- OUT 71H, AL ; 写低 8 位
- MOV AL, AH
- OUT 71H, AL ; 写高 8 位



相同点 均为减 1 计数; 工作时输出低电平; 均具备定时功能。

● **实例 1-单通道编程:**
需求: 计数器 2, 方式 1, 初值 15, 仅低 8 位。地址: 控制口 **COUNT2**, 计数器 2 **COUNTC**。

控制字 SC = 10, RL = 01, M = 000 \rightarrow 001, BCD = 0 \rightarrow **92H**
MOV AL, 92H
OUT COUNTD, AL ; 写控制字
MOV AL, 15
OUT COUNTC, AL ; 写初值

汇编实现

初始化 写入方式控制字后 **OUT** **变高**; 写入初值后 OUT 保持高电平。
触发时刻 **GATE 上升沿**触发。在随后的下一个 CLK 下降沿, OUT 由高变低。
计数与结束 计数器减 1 期间 OUT 为低; 减至 0 时 OUT 跳变为高。低电平宽度 = $N \times T_{CLK}$ 。

● **可重触发性:** 在脉冲未结束时, 若 **GATE** 再次出现**上升沿**, 8253 会在下一个 CLK 将初值寄存器 (CR) 的值**重新装入**执行单元 (CE), 使计数器重新开始, 从而**延长 OUT 低电平宽度**。

● **修改初值 (方式 1):** 在脉冲输出过程中, 若 CPU 修改计数初值:

系统响应 新初值 N_{new} 仅存入 **初值寄存器 (CR)**。当前计数器 (CE) 不受影响, 继续按旧值 N_{old} 减至 0, 当前脉冲宽度不变。
生效时刻 仅在**下一次 GATE 上升沿**到来时, 新值 N_{new} 才从 CR 装入 CE。
结论 写入新初值**不会立即重启**计数, 而是预置给下一次触发使用。

方式 0 (软件触发)	方式 1 (硬件触发)
写初值操作直接启动计数 GATE 电平敏感: 1 计数, 0 暂停 低电平宽度为 $N + 1$ 个 CLK	GATE 上升沿触发启动 GATE 边沿敏感: 上升沿重触发 低电平宽度为 N 个 CLK

● **实例 2-波形分析与混合:**

需求: 根据波形反推。OUT0 为方式 0, OUT1 为方式 1。初值均为 7。

CNT0 控 SC = 00, RL = 01, M = 000 \rightarrow 制字 **10H**
CNT1 控 SC = 01, RL = 01, M = 001 \rightarrow 制字 **52H**
汇编实现
MOV AL, 10H; OUT COUNTD, AL
MOV AL, 52H; OUT COUNTD, AL
MOV AL, 7
OUT COUNTA, AL ; CNT0 初值
OUT COUNTB, AL ; CNT1 初值