微机原理
期末通关

这门课一半是**背**（引脚、时序、寄存器），一半是**画**（存储器和 8259 的连线）。这本按"**先看懂图，再背数字**"来写。

目录

00 这门课在讲什么
01 微机系统组成与三总线
02 8086 内部结构与寄存器
03 存储器与地址空间
04 引脚与总线时序
05 寻址方式
06 指令系统
07 汇编程序与比较跳转
08 中断系统与中断向量表
09 8259A 可编程中断控制器
10 I/O 接口与端口
11 存储器扩展与地址译码
12 数制、码制与复位
13 8253 定时/计数器
14 8255A 并行接口
15 汇编程序阅读与数据段
16 画图题专项：时序图 / 存储图 / 堆栈图 / 芯片连接图
17 整卷速查：考前 15 分钟

00

这门课在讲什么

约 2 分 · 8 分钟

先看清全图，再决定力气往哪使

> **一句话：**整门课只追一条线 —— 指令怎么从内存进 CPU，算完又怎么送到外设。

? 你在键盘上敲 3+4，屏幕上出现 7。中间发生了什么？
谁把 3 和 4 搬过去的？谁算的？7 又是怎么回来的？

先讲清楚

一、这门课只回答三个问题

一台计算机干的所有事，都能拆成“取指令、算一下、送出去”三件事。

你写一行 **a = 3 + 4**，机器并不认识“加法”这两个字。

它只认两样：**一串 0/1 组成的指令**，和**某个地址上放着的数**。

问题	谁干	在哪几讲
指令从哪来	BIU 按 CS:IP 去内存取	02～04
在哪儿算	EU 里的 ALU	02、05、06
结果送去哪	I/O 接口，再交给外设	08～11

三个问题，正好对上三组线：**地址线说去哪，数据线搬东西，控制线说读还是写。**

> **小结：**这门课不是背“计算机原理”，是沿一条数据通路走一遍，每站停一下。

二、全课地图：四块，一条链

十七讲不是十七个知识点，是四条链，前一条是后一条的地基。

块	讲	解决什么
一、整机	01～02	机器由哪几块拼成，CPU 肚子里有什么
二、存储与总线	03～04	20 位地址怎么拼出来，一个读周期谁先谁后

三、指令	05～07	操作数在哪，有哪些指令，程序怎么跳
四、接口	08～11	中断、8259A、I/O 端口、存储器扩展

四块之外还有几讲是“补丁”：数制与码制、8253 定时器、8255A 并行口、汇编程序阅读、画图专项。

顺序为什么这么排？**先认人（CPU），再认路（地址与时序），再认话（指令），最后接外设。**

> **停一下：**为什么“寻址方式”要排在“指令系统”前面？

一条指令分两半：**操作码 + 操作数**。操作码说“干什么”，操作数说“对谁干”。

寻址方式回答的就是“操作数在哪”：在寄存器里、在内存某地址、还是指令里直接带着。

不会寻址，看到 **MOV AX,[BX]** 就不知道 BX 是数还是地址。

三、卷面长什么样：三种能力

这张卷子只考三种能力：背得准、算得快、连得对。

题型	题量	考什么	怎么练
填空	10 题	数字、名字、作用	把 must 盒背成条件反射
选择	10 题	概念辨析	分清“双向/单向”“字/字节”
简答	3～4 题	说原理、画时序	合上书能画出读周期
设计/计算	2～3 题	地址计算、芯片连线	手算 3 道、手画 2 张

一句话：**会背能拿一半，会画图能拿八成。**

时间要反着分配：**设计题十几分，填空 4 分。**

所以先练“算地址”和“画连线”，零碎时间再背数字。

> **停一下：**只复习一晚上，先做哪类题？

先做设计/计算：一道 10～15 分，步骤分多，套路固定（算容量 → 分组 → 接译码）。

填空靠平时零碎记忆，一晚上硬塞 40 个数字，第二天会混。

四、要练的三个动作：背、算、画

这门课没有“看懂了就自动会”的部分，三个动作都得动手。

动作	例子	练法
背数字	8086：16 位数据线、20 位地址线、6 字节队列	每天 5 分钟出声念
算地址	段 2000H、偏移 0100H → 20100H	纸上算 10 道，不许按计算器
画连线	6264 + 74LS138 + 74LS373	白纸默画，画完对照

每次只练一个。**三个一起练，等于一个都没练。**

> **小结：**背和画各占一半，别偏科。只背不画，设计题空着；只画不背，填空全丢。

五、一讲怎么学：三遍法

每一讲按"看懂 → 背熟 → 做题"三遍过，别想一遍全懂。

1. **第一遍只看图和例子**，不求记住数字，先知道这一讲在解决什么麻烦。
2. **第二遍背 must 盒**，六条，一条一条合上书复述。
3. **第三遍做 ex 和 quiz**，卡住的地方回头只看那一小节。
4. **最后做末尾的真题区**，卷1/卷2/卷3 里属于这一讲的题都在那。

时间上：**一讲 40 分钟，15 分钟看、10 分钟背、15 分钟写。**

看完立刻合上，用一句话说出它解决什么问题。**说不出来，就是没看懂。**

> **停一下：**只有 20 分钟，先看哪部分?

先看 **core**（解决什么）和 **must**（要背什么），再扫一眼 **ex** 的第一步（题长什么样）。

三样加起来 5 分钟。剩下 15 分钟直接做题，卡住再回头翻那一小节。

六、先绕开三条弯路

这三件事现在做，等于把时间扔进水里。

别做	为什么
一上来背指令表	指令 = 操作码 + 操作数。寻址没懂，背了也不知道操作数在哪
跳过时序图	简答就考"读周期里 RD# 什么时候有效"，图本身就是答案

把 8086 和 8088 当一个东西	队列 6 对 4，数据线 16 对 8，两个空一起错

必背 · 六条

微机 = CPU + 存储器 + I/O	三块积木
系统总线 = DB + AB + CB	数据/地址/控制
8086：16 位数据线、20 位地址线	20 根定 1MB
物理地址 = 段 × 16 + 偏移	左移 4 位
接口三件套：8259 / 8253 / 8255	中断/定时/并行
学习链：整机→总线→指令→接口	别跳步

导览 · 例题 一句话走完全课

程序里有一句 **ADD AX,4**，执行前 AX=0003H，结果要显示到屏幕上。问这一路经过哪几块?

第一步：先有指令。编译后这句话变成机器码 **05 04 00**（ADD AX，立即数 4），共 3 字节。

第二步：指令得在内存里。设 CS=2000H、IP=0100H，它落在 **20100H**。

第三步：取指。BIU 按 CS:IP 从 20100H 读 3 字节进 6 字节队列，IP 变 0103H。

第四步：执行。EU 译码出"加法、加 4"，ALU 算 0003H+0004H=**0007H**，写回 AX。

第五步：结果要出门。屏幕不是内存，得走 I/O：**OUT** 指令把 07H 送到接口，比如 8255A。

第六步：靠三组线完成。地址线送端口号，数据线送 07H，控制线把 WR# 拉低表示"写"。

验算：20100H 在 00000H～FFFFFH 内，20 位够 ✓；一次取 3 字节 < 6 字节队列 ✓；0007H 低 8 位有 3 个 1，是奇数，所以 PF=0 ✓。

⚠ **三个坑：**

① **把"微处理器"当"微型计算机"。**错在把一块芯片当整机。问"系统由什么组成"只写 CPU，就漏了存储器和 I/O。

② **背目录当学懂。**错在只记住"第 3 讲讲存储"。换成"16 位寄存器怎么寻 1MB"照样答不出

——目录是路标，不是知识。

③ **拿 8086 的参数套 8088。**队列写成 6/6，数据线写成 16/16。这两数成对出现：**16 位数据线配 6 字节队列，8 位配 4 字节。**

自测 · 3 道

1. 四块内容按什么顺序排？为什么不能先学接口？
2. 卷面上单题分值最高的是哪类题？它主要考哪两个动作？
3. “微处理器”“微型计算机”“微型计算机系统”三者差在哪？

1. 整机→存储与总线→指令→接口。接口讲“CPU 怎么和外面说话”，得先知道 CPU 里有什么、地址怎么给、指令怎么读操作数。

2. 设计/计算题，10～15 分。考“算地址”和“画连线”。

3. 微处理器只是一块 CPU；微型计算机 = CPU + 存储器 + I/O 接口 + 总线；微型计算机系统再配外设和软件。

01

微机系统组成与三总线

约 4 分 · 25 分钟

一台机器由哪几块拼成，又靠什么连起来

一句话：三块积木（CPU、存储器、I/O）用三组公共导线（数据、地址、控制）连成一台机器。

? CPU 怎么知道去哪读程序？读回来的数又走哪根线？
如果每个部件都单独拉线，6 个部件要拉 15 组，机箱里根本塞不下。

先讲清楚

一、把“微机”拆成三块积木

一台微型计算机 = 微处理器 + 存储器 + I/O 接口，外设挂在接口上。

为什么非得三块？只有 CPU 的话，它没地方放程序，也没法和人交换信息。

好比厨房：CPU 是厨师，存储器是冰箱，I/O 接口是传菜口。**厨师再快，没冰箱和传菜口也开不了张。**

部件	干什么	具体例子
微处理器 CPU	运算 + 控制	8086

存储器	存程序和数据	6264（RAM）、2764（ROM）
I/O 接口	给外设当翻译	8255A、8253、8259A

三个词天天混，一次分清：

说法	包含什么
微处理器	只是一块 CPU 芯片
微型计算机	CPU + 存储器 + I/O 接口 + 总线
微型计算机系统	微机 + 外设 + 软件

范围越往下越大：**芯片 ⊂ 整机 ⊂ 系统**。答题时看清问的是哪一个。

二、为什么必须用总线

总线是一组公共导线，所有部件都挂在上面，同一时刻只允许一个部件发话。

先算"两两直连"的账：6 个部件，每两个之间拉一组线，要 $6\times5/2=15$ 组。

再加第 7 个部件，还得补拉 6 组。**机器越复杂，线越多，根本做不下去。**

改成一组总线，6 个部件共用 1 组线，每块板只加 1 个接口。

代价有两条：**一次只能一家说话（分时）；得有一套"谁先说"的规矩（仲裁）。**

所以总线的本质是：**用时间换线数。**

> 停一下：8 个部件两两直连要几组线？用总线呢？

两两直连：$8\times7/2=28$ 组。总线：1 组线 + 8 个接口，省下 27 组。

部件越多越划算，代价只是大家排队用线。

三、三总线：按"传什么"分三组

系统总线按传的东西分成三组：数据总线 DB、地址总线 AB、控制总线 CB。

总线	传什么	方向	8086 根数	决定什么
DB	数据	**双向**	16 根	一次搬几位

AB	地址	**单向**（CPU 发）	20 根	能找多大地方
CB	读写、中断、时钟	有出有入	若干根	什么时候搬

方向是重点：**地址永远是 CPU 送出去的，数据要读要写所以双向。**

控制线"有出有入"：CPU 发 RD#、WR# 是出；存储器回 READY、外设发 INTR 是入。

20 根地址线 ⇒ 2^{20}=1048576 字节 = 1MB，地址 00000H～FFFFFH。

16 根数据线 ⇒ 一次 2 字节 = 1 个字（8086 里"字"就是 16 位）。

换成 8088：地址线同样 20 根，数据线只有 8 根，一次传 1 字节。

> **停一下：**地址线 24 根能寻多大？10 根呢？

2^{24}=16MB；2^{10}=1024 字节 = 1KB。

地址线每多一根，空间翻一倍：20 根 1MB，21 根就 2MB。

四、三组线怎么配合读一次内存

一次读内存 = 地址先稳住、控制信号拉低、数据再回来，三步有严格先后。

三组线不是静线，它们按节拍动作。**节拍错了，读回来就是垃圾。**8086 一个总线周期分 4 个 T：

阶段	总线上发生什么
T1	地址上 AB；ALE 变高，把地址锁进 74LS373
T2	地址撤走，数据线让出来；RD# 变低
T3	存储器把数据放上 DB；CPU 在 T3 末尾采样
T4	撤 RD#，释放总线

为什么地址要锁？**8086 的 AD0～AD15 是地址和数据合用一组引脚**，T1 过后地址就没了。

慢芯片来不及怎么办？拉低 READY，CPU 在 T3 后插一个 Tw 等它。

> **小结：**时序就考这张表：**T1 地址、T2 切换、T3 读写、T4 结束。**

五、CPU 肚子里有什么

微处理器内部 = ALU + 控制器 + 寄存器组，三者用内部总线连起来。

为什么分三块？“算”“指挥”“暂存”是三件不同的事，混在一起没法并行。

部件	干什么	具体例子
ALU	加减、与或非、比较	0003H+0004H=0007H
控制器	取指、译码、发 RD#/WR#	认出 05 是“ADD AX，立即数”
寄存器组	暂存操作数、地址、结果	AX 存 0007H，IP 存下条指令地址

连接它们的是**内部总线**，16 位宽，一次搬一个字。有的教材把它算作第四部分，写四个更全。

两个特殊寄存器：**PC（8086 里叫 IP）存下一条指令的地址；PSW 存运算结果的状态。**

六、程序是怎么跑起来的

程序是一串指令，CPU 干的事就是“取指—译码—执行”的死循环。

一条指令 = **操作码 + 操作数**。操作码说“干什么”，操作数说“对谁干”。

例：**MOV AX,1234H**。MOV 是操作码，AX 和 1234H 是两个操作数。

阶段	谁干	结束时
取指	BIU 按 CS:IP 读指令	IP 加上指令长度
译码	EU 拆出操作码和操作数	知道要干什么
执行	EU 取数、运算、写回	结果进寄存器或内存

一个指令周期 = **取指周期 + 执行周期**，然后接着取下一条。

> **小结：**三步不是严格排队：8086 用 6 字节队列让取指和执行重叠，下一讲讲。

必背 · 六条

微机 = CPU + 存储器 + I/O	三块积木
系统总线 = DB + AB + CB	三总线
DB 双向，AB 单向	方向别写反
20 根地址线 → 1MB	2 的 20 次方
CPU = ALU + 控制器 + 寄存器组	加内部总线是四部分
指令 = 操作码 + 操作数	周期 = 取指 + 执行

总线 · 算 1 总线根数换算

某 8086 系统，地址总线 20 根，数据总线 16 根。求寻址空间、地址范围、一次最多传几字节；再问读 00002H 一个字节时 A0、BHE# 各是什么。

第一步：算寻址空间。2^{20}=1048576 字节 = **1MB**。

第二步：写地址范围。20 位全 0 到全 1，写成 5 位十六进制：**00000H～FFFFFH**。

第三步：算一次传送。16 根数据线 = 16 位 = **2 字节**，正好一个字。

第四步：判断奇偶。00002H 末位是 2，偶地址 ⇒ **A0=0**、BHE#=1，走低 8 位 D7～D0。

第五步：换数据线。数据线只 8 根时一次传 **1 字节**，地址空间仍是 1MB。

验算：20 根 = 5 位十六进制，00000H～FFFFFH 共 2^{20} 个 ✓；16÷8=2 字节 ✓；A0 与 BHE# 不能同时为 1，否则一个字节也读不出 ✓。

答案：1MB；00000H～FFFFFH；2 字节；A0=0、BHE#=1；8 根时 1 字节。

⚠ **三个坑：**

① **地址总线写成双向。**地址只从 CPU（或 DMA 控制器）发出去，对方只会“听”。写成双向就是没分清谁选谁。

② **把“微处理器”当“微型计算机”。**前者是一块芯片，后者是 CPU+存储器+I/O+总线的整机。问“系统组成”只写 CPU，三块变一块。

③ **拿数据线根数当寻址能力。**16 根数据线只说明一次搬 2 字节，和能找多大地方无关；寻址看地址线。

自测 · 3 道

1. 系统总线分哪三种？哪个是单向的？
2. 地址总线 20 根、数据总线 16 根，寻址空间和一次传送各是多少？
3. 一条指令由哪两部分组成？一个指令周期分哪两段？

1. 数据总线 DB（双向）、地址总线 AB（单向）、控制总线 CB。

2. 2^{20} = 1MB；16 位 = 2 字节。

3. 操作码 + 操作数；取指周期 + 执行周期。

2024 卷 · 真题 2 道

2024 · 填1 4 分

○ 会 ○ 卡 ○ 不会

微型计算机系统是指以（ ）为核心，配以（ ）和（ ）构成的系统。

一句话：问微机系统由哪三部分拼成。 ① CPU 是核心，负责运算与控制。 ② 存储器存放程序和数据。 ③ I/O 接口连接外设，三者挂在总线上。 **答案：微处理器（CPU）、存储器、I/O 接口电路（含外设）。**

2024 · 填2 4 分

○ 会 ○ 卡 ○ 不会

系统总线分为（ ）、（ ）、（ ）。

一句话：背总线的三分类。 ① 数据总线 DB 传数据，双向。 ② 地址总线 AB 传地址，单向，20 位定 1MB。 ③ 控制总线 CB 传读写、中断等控制信号。 **答案：数据总线、地址总线、控制总线。**

2022 卷 · 真题 1 道

2022 · 填1 4 分

○ 会 ○ 卡 ○ 不会

微型计算机由（ ）、（ ）、（ ）组成。

一句话：问微机硬件系统由哪几大块组成。 ① 核心是微处理器 CPU，负责运算与控制。 ② 存储器存放当前运行的程序和数据。 ③ I/O 接口与外设负责与外界交换信息。 **答案：中央处理器（CPU）、存储器、I/O 接口及外设。**

2020 卷 · 真题 2 道

2020 · 填1 4 分

○ 会 ○ 卡 ○ 不会

一般的微处理器内部主要由四部分组成：_____、_____、_____和_____。

一句话：背出微处理器内部的四大部件。 ① 运算器 ALU：完成算术与逻辑运算。 ② 控制器：取指、译码、发控制信号。 ③ 寄存器组：暂存操作数、地址、结果和状态。 ④ 内部总线：连接以上部件。 **答案 运算器（ALU）、控制器、寄存器组、内部总线**

2020 · 填2 4 分

○ 会 ○ 卡 ○ 不会

微机系统的总线按传输信息的种类可分为_____、_____和_____。

一句话：按信息种类把系统总线分成三组。 ① 数据总线 DB：双向，根数决定并行位宽。 ② 地址总线 AB：单向输出，根数决定寻址空间。 ③ 控制总线 CB：传读写、中断、DMA 等信号。 ④ 三者合称三总线。 **答案 数据总线、地址总线、控制总线**

02 8086 内部结构与寄存器

约 4 分 · 30 分钟

CPU 肚子里到底装了什么，14 个寄存器各管什么

> **一句话：**8086 劈成 BIU 和 EU 两半，14 个寄存器分工，20 位地址靠“段×16+偏移”拼。

? 16 位的寄存器最大只能表示 65535，可 8086 要访问 1MB。
多出来的那 4 位地址，是从哪儿变出来的?

先讲清楚

一、为什么要劈成 BIU 和 EU

8086 把内部劈成 BIU 和 EU 两半，为的是取指和执行能同时干。

先看“不劈”有多亏：取一条指令要等存储器好几个时钟，这期间 ALU 全在空转。

劈开以后，BIU 管“出门拿货”，EU 管“在家加工”，**拿货和加工可以重叠**。

单元	全称	管什么	里面有什么
BIU	总线接口单元	取指、读写内存和 I/O	4 个段寄存器、IP、指令队列
EU	执行单元	译码、运算、写回结果	ALU、通用寄存器、标志寄存器

边界：**EU 不直接碰总线**，要读内存得向 BIU 提请求。

算笔账：设取指 4 个时钟、执行 4 个时钟，跑 4 条指令。**串行要 8+8+8+8=32 个时钟，重叠只要 8+4+4+4=20 个。**

> **小结：**流水线不缩短单条指令，它缩短一批指令。

二、指令队列：6 字节的小仓库

指令队列是 BIU 预取指令的仓库，8086 是 6 字节，8088 是 4 字节。

为什么按字节不按条？因为指令长短不一，短的 1 字节，长的 6 字节。**队列是字节仓库，不是指令仓库。**

队列没满就继续取；满了就停下，等 EU 取走一条。

什么时候会“断流”？遇到**转移、调用、中断**。队列里存的是“顺序的下一条”，可 CPU 要跳走，**整队作废，按新地址重取。**

8086：数据线 16 位、队列 6 字节、一次取 2 字节。8088：数据线 8 位、队列 4 字节、一次取 1 字节。

两个数成对记：**数据线宽的，队列也深。**

停一下：6 字节队列能装几条指令?

不一定，1～6 条。8086 指令长度 1～6 字节，平均 3 字节的话大约两条。

所以"队列几条"这个问法本身是错的，只能答字节数。

三、14 个寄存器：4+4+4+2

8086 里编程能看见的寄存器一共 14 个，按 4+4+4+2 分成四组。

组	名字	个数	能不能拆 8 位
通用	AX、BX、CX、DX	4	能，拆成 AH/AL
指针/变址	SP、BP、SI、DI	4	不能
段	CS、DS、SS、ES	4	不能
控制	IP、PSW（FLAGS）	2	不能

为什么只有通用寄存器能拆？因为它们要当 8 位数据用。AX=1234H 时，**AH=12H**、**AL=34H**，高字节在前。

四个还各带一点"兼职"：AX 是累加器，乘除和 IN/OUT 默认用它；BX 常当基址；CX 当循环计数。

指针变址四个不能拆：**SP 是堆栈指针，BP 是堆栈里的基址，SI/DI 是源/目的变址。**别写 SPL、SPH，芯片里没有。

四个段寄存器各 16 位，**存的是段基址，不是物理地址**；IP 存下条指令的偏移，PSW 存标志。

四、段寄存器：16 位怎么够到 1MB

物理地址 = 段基址 × 16 + 偏移，这一乘就把 16 位变成了 20 位。

为什么乘 16？因为 16 = 2^4，十六进制里就是**左移一位、末尾补个 0**。

例：CS=2000H、IP=0100H ⇒ 20000H + 0100H = **20100H**。

副作用：**段基址最低 4 位恒为 0**，段只能从 16 字节整数倍开始，1MB 里共 65536 个段起点。

为什么非要这么设计？16 位寄存器最多寻 64KB，20 根地址线要 1MB。**拿两个 16 位凑一个 20 位，是最便宜的解法。**

段寄存器	默认配谁	管什么
CS	IP	取指令
DS	BX、SI、DI、直接地址	数据

SS	SP、BP	堆栈
ES	DI（串操作）	附加段

口诀：**取指 CS:IP，数据默认 DS，堆栈 SS，只有 BP 特殊也配 SS。**

段可以重叠：同一块内存能被两个段盖住。**物理地址唯一，段:偏移 不唯一。**

停一下：DS=2100H、BX=0158H，[BX] 的物理地址是多少?

2100H×16 = 21000H，加 0158H = **21158H**。

注意是乘 16（末尾补 0），不是乘 10。补错一位，整题错。

五、PSW：机器怎么记住上次的结果

PSW 用 9 个 1 位的标志记录上次运算的结果，后面的条件跳转全靠它。

CMP AX,BX 比完，JE 要知道“相等吗”。**不能重算，就查 ZF。**

标志	什么时候是 1	管什么
CF	最高位向外出进位/借位	无符号溢出
ZF	结果为 0	相等判断
SF	结果最高位是 1	有符号的负
OF	有符号运算溢出	有符号出错
PF / AF	低 8 位 1 的个数为偶 / 低 4 位进位	校验 / BCD 调整
DF / IF / TF	方向 / 开中断 / 单步	三个控制标志

看三个加法，把 CF 和 OF 彻底分开：

运算	8 位结果	CF	ZF	OF
03H+04H	07H	0	0	0
0FFH+01H	00H	1	1	0
7FH+01H	80H	0	0	1

第二行：无符号 255+1 进位了（CF=1），有符号 −1+1=0 没溢出（OF=0）。

第三行：无符号 127+1 没进位（CF=0），有符号 127+1 超出 +127（OF=1）。

一句话：**CF 管无符号，OF 管有符号。**同一组数，两个标志看的是两件事。

必背 · 六条

BIU 取指，EU 执行	靠队列并行
队列：8086 是 6，8088 是 4	按字节算
14 = 4+4+4+2	通用/指针/段/控制
物理地址 = 段 × 16 + 偏移	乘 10H
CS:IP 取指，SS:SP 堆栈	BP 也配 SS
CF 管无符号，OF 管有符号	两个溢出

寄存器 · 算 1 一条指令走一遍

CS=2000H、IP=0100H、DS=3000H、BX=0020H、SS=4000H、BP=0020H、AX=0000H，30020H 起两字节是 34H、12H。问 **MOV AX,[BX]** 的取指地址、操作数地址、执行结果。

第一步：算取指地址。CS×16 = 20000H，加 IP 0100H ⇒ **20100H**。

第二步：找操作数的段。地址里出现 BX，默认段是 **DS**，不是 CS。

第三步：算操作数地址。3000H×16 = 30000H，加 0020H ⇒ **30020H**。

第四步：读一个字。30020H 是偶地址，A0=0、BHE#=0，两字节一次读完，低 8 位 34H、高 8 位 12H，拼成 **1234H**。

第五步：看标志。MOV 不影响标志，ZF/SF/CF 保持原值。

第六步：改 IP。这条指令机器码 8B 07，占 2 字节，IP=0100H+2=**0102H**。

第七步：换成 [BP]。出现 BP 就改用 SS：4000H×16 = 40000H，加 0020H = **40020H**。

验算：20102H−20100H=2 字节 ✓；30020H 的偏移 0020H 没超 64KB ✓；40020H−30020H=10000H，正好差一个段 ✓。

答案：20100H；30020H；AX=1234H；IP=0102H；[BP] 是 40020H；标志不变。

⚠ **三个坑：**

① **把 SP/BP/SI/DI 拆成 8 位。**只有 AX/BX/CX/DX 能拆。写“SPL”“SPH”的答案直接错，芯片里没这两个寄存器。

② **队列 6/4 记反，或说成“6 条指令”。**队列单位是**字节**；8086 指令 1～6 字节，6 字节平均只装两条。

③ **以为 BP 也默认配 DS。**基址/变址加位移的地址里，**只有 BP 走 SS**；写成 DS，整道地址题从第一步错。

自测 · 3 道

1. BIU 和 EU 各管什么？为什么能同时工作？
2. 8086 有几个寄存器？哪四个不能拆成 8 位？
3. CS=2000H、IP=0100H 的物理地址？换 DS=2000H、BX=0100H 呢？

1. BIU 取指、读写内存和 I/O，带段寄存器、IP、队列；EU 译码执行，带 ALU、通用寄存器、PSW。中间有队列，取指不必等执行。

2. 14 个（4+4+4+2）；SP、BP、SI、DI 不能拆。

3. 都是 20100H。物理地址唯一，但“段:偏移”的写法不唯一。

2024 卷 · 真题 5 道

2024 · 填3 4 分 ○ 会 ○ 卡 ○ 不会

微处理器由（ ）、（ ）、（ ）组成。

一句话：问 CPU 内部的三块。 ① ALU 完成算术与逻辑运算。 ② 控制器取指、译码、发控制信号。 ③ 寄存器组暂存操作数、地址和状态。 **答案：算术逻辑单元 ALU、控制器、寄存器组。**

2024 · 填4 4 分

○ 会 ○ 卡 ○ 不会

程序计数器的作用（ ），标志寄存器的作用（ ）。

一句话：PC 指路、PSW 记事。 ① PC 就是 IP，始终指向下一条待取指令。 ② 每取一个字节，IP 自动加 1。 ③ PSW 记录 CF/ZF/SF/OF，还含 IF、TF、DF。 **答案：PC 存放下一条指令的地址；PSW 记录结果状态并控制工作方式。**

2024 · 填5 4 分

○ 会 ○ 卡 ○ 不会

一条指令的执行分为（ ）和（ ）。

一句话：问指令执行的两个阶段。 ① 取指周期：BIU 按 CS:IP 从内存取指令。 ② 执行周期：EU 译码、取操作数、运算、存结果。 ③ 两阶段由 BIU 与 EU 重叠进行。 **答案：取指周期和执行周期。**

2024 · 填6 4 分

○ 会 ○ 卡 ○ 不会

微机的工作过程是（ ），程序由（ ）组成。

一句话：一句话概括计算机怎么干活。 ① 取指令：按 CS:IP 取指令进队列。 ② 译码：分析操作码，产生控制信号。 ③ 执行：取操作数运算，再回到取指。 **答案：取指令—译码—执行的循环过程；程序由指令组成。**

2024 · 填7 4 分

○ 会 ○ 卡 ○ 不会

8086CPU 采用（ ）工作方式，对内存空间实现（ ）管理。

一句话：问 8086 的两个招牌设计。 ① BIU 预取指令进 6 字节队列，EU 同时执行。 ② 取指与执行并行，即流水线。 ③ 1MB 空间按段管理，段基址×16 加偏移定址。 **答案：流水线（取指与执行并行）；分段管理。**

2023 卷 · 真题 3 道

2023 · 填1 4 分

○ 会 ○ 卡 ○ 不会

8086CPU 继承了如下两种功能单元：（ ）和（ ）

一句话：问 8086 内部按"取指"和"执行"分成的两大功能单元。 ① EU 是执行单元，负责指令译码、执行和运算。 ② BIU 是总线接口单元，负责取指令、读写内存和 I/O。 ③ BIU 里还有段寄存器、IP 和 6 字节指令队列。 **答案：EU（执行单元）和 BIU（总线接口单元）。**

2023 · 填7 4 分

○ 会 ○ 卡 ○ 不会

8086CPU 的指令队列长度：（ ），8088CPU 的指令队列长度：（ ）。

一句话：区分 8086 和 8088 的指令队列深度。 ① 指令队列是 BIU 里的小缓冲，预取的指令排队等 EU 执行。 ② 8086 数据线 16 位，队列 6 字节。 ③ 8088 数据线只有 8 位，队列缩为 4 字节。 **答案：8086 为 6，8088 为 4。**

2023 · 答1 8 分

○ 会 ○ 卡 ○ 不会

8086 中的段寄存器包括哪些，段寄存器的功能有哪些？并逐一说明。

一句话：把 4 个段寄存器逐个点名并说清功能。 ① 共 4 个：CS、DS、SS、ES，各 16 位。 ② CS 是代码段，与 IP 配对形成 CS×16+IP。 ③ DS 是数据段，与 BX、SI、DI 配对。 ④ SS 是堆栈段，与 SP、BP 配对。 ⑤ ES 是附加段，常作串操作的目的段。 ⑥ 分段的意义：16 位寄存器移 4 位才能访问 1MB。 **答案：CS/DS/SS/ES 分别存代码段、数据段、堆栈段、附加段基址，与偏移量合成 20 位物理地址。**

2022 卷 · 真题 7 道

2022 · 填2 4 分

○ 会 ○ 卡 ○ 不会

8086 内部包含（ ）和（ ）。

一句话：考 8086 内部两大功能单元的名字。 ① BIU 总线接口单元：取指、读写内存、管段寄存器。 ② EU 执行单元：译码并执行指令，内含 ALU。 ③ 两者靠指令队列衔接，取指与执行同时进行。 **答案：总线接口单元 BIU、执行单元 EU。**

2022 · 填4 4 分

○ 会 ○ 卡 ○ 不会

8086 是（ ）位的 CPU，有（ ）根数据线，有（ ）根地址线。

一句话：背 8086 的字长和引脚位数。 ① 16 位 CPU，内部一次处理 16 位。 ② 数据线 16 根 D0–D15，一次传一个字。 ③ 地址线 20 根 A0–A19，最大寻址 1MB。 **答案：16 位；16 根数据线；20 根地址线。**

2022 · 填5 4 分

○ 会 ○ 卡 ○ 不会

8086 内部有（ ）个寄存器。

一句话：数 8086 编程可见的寄存器总数。 ① 通用寄存器 8 个：AX、BX、CX、DX、SP、BP、SI、DI。 ② 段寄存器 4 个：CS、DS、SS、ES。 ③ 再加 IP 和 FLAGS，共 14 个。 **答案：14 个。**

2022 · 选1 4 分

○ 会 ○ 卡 ○ 不会

CS、SS、DS、IP 四个寄存器中与其他三个不同类的是（ ）。

一句话：从寄存器堆里挑出不是段寄存器的那个。 ① CS、DS、SS 都是段寄存器，存段基址。 ② 同类的还有 ES。 ③ IP 是指令指针，存的是偏移地址。 **答案：IP。**

2022 · 选5 4 分

○ 会 ○ 卡 ○ 不会

以下哪个是指令指针寄存器（ ）。

一句话：问谁负责指向下一条指令。 ① 指令指针是 IP，存下一条指令的偏移地址。 ② IP 与 CS 组合成 CS:IP，指向当前取指地址。 ③ IP 不能用 MOV 直接改，只能由转移指令改。 **答案：IP。**

2022 · 选6 4 分

○ 会 ○ 卡 ○ 不会

8086 CPU 在工作过程中表示状态的寄存器是（ ）：A.CS　B.DS　C.IP　D.PSW

一句话：问哪个寄存器记录运算状态。 ① CS、DS 是段寄存器，只存段基址。 ② IP 存下一条指令的偏移地址。 ③ PSW（FLAGS）存 ZF、CF、OF 等标志位。 **答案：D，PSW。**

2022 · 答1 8 分

○ 会 ○ 卡 ○ 不会

8086 CPU 是如何实现并行流水线的?

一句话：讲 BIU 与 EU 如何分工同时干活。 ① 8086 内部拆成 BIU 和 EU 两个独立单元。 ② BIU 负责取指，先送进 6 字节指令队列。 ③ EU 从队列取指令译码执行，两者同时工作。 ④ 队列没满时 BIU 继续预取，不必等 EU。 ⑤ 只有遇到转移指令才清空队列重新取指。 **答案：取指与执行分家、用指令队列缓冲，实现边取边执行。**

2020 卷 · 真题 1 道

2020 · 填6　4 分

○ 会　○ 卡　○ 不会

8086CPU 有两个控制寄存器_____和_____。

一句话：IP 指路、FLAGS 记状态。 ① IP：存下一条待取指令的偏移地址，与 CS 拼成物理地址。 ② FLAGS：16 位标志寄存器，含状态标志与控制标志。 ③ 状态标志 ZF、CF、SF 由运算自动置位。 ④ 控制标志 IF、DF、TF 用专用指令改写。 **答案 IP（指令指针）和 FLAGS（标志寄存器）**

03 存储器与地址空间

约 10 分 · 30 分钟

16 位的寄存器，怎么够到 1MB 那么远的地址

一句话：寄存器只有 16 位，8086 用「段 × 16 + 偏移」拼出 20 位物理地址。

? 8086 有 20 根地址线，能寻址 1MB，最大地址 FFFFFH。
可它的寄存器只有 16 位，最大只能装 FFFFH，也就是 64KB。
一个 16 位的数，怎么指向 1MB 里的第 70000H 个字节?

先讲清楚

一、为什么非分段不可

16 位寄存器装不下 20 位地址，这是分段存在的唯一理由。

20 根地址线，每根取 0 或 1，一共 2^{20}=1048576 个字节，也就是 **1MB**。

编号 00000H~FFFFFH，1MB = 1024KB = 1048576 字节。

可 16 位寄存器最大只有 FFFFH = 65535，只够 64KB。差 4 位，怎么办?

拿两个 16 位：一个当**段基址**，一个当**偏移**。

量	位数	范围
物理地址	20 位	00000H~FFFFFH
段基址	16 位	0000H~FFFFH

偏移地址	16 位	0000H~FFFFH

公式：物理地址 = 段基址 × 16 + 偏移地址。

×16 在十六进制里就是**末尾添一个 0**，等价于二进制左移 4 位。

例：CS=2000H，IP=0100H。2000H×16=20000H，再加 0100H，得 **20100H**。

注意是乘 16（10H），不是乘 10。这一步错，整题全错。

> **停一下：**CS=1000H，IP=2345H，取指令的物理地址是多少?

1000H×16 = 10000H；10000H + 2345H = **12345H**。

> **小结：**段值管起点，偏移管走多远；先乘 16，再加。

二、段起点为什么都在 16 字节的整数倍上

段基址要乘 16，所以段起点在二进制里末 4 位恒为 0。

2000H 左移 4 位得 20000H，所以段只能从 **16 字节的整数倍**处开始。

1MB ÷ 16 = 65536，段值 0000H~FFFFH 共 **65536 个**起点；一段最长 **64KB**。

段之间可以重叠，因为段长 64KB，而相邻起点只差 16 字节。

2000H:0010H：2000H×16=20000H，+0010H=20010H。

2001H:0000H：2001H×16=20010H，+0000H=20010H。

同一单元能写成很多组「段:偏移」，这叫**段重叠**，也叫段可滑动。

段值 2000H 的段覆盖 20000H~2FFFFH，整整 64KB。

> **停一下：**段值 3000H 的段，起点和末地址分别是多少?

起点 3000H×16 = **30000H**；末地址 30000H + FFFFH = **3FFFFH**。

> **小结：**段值 16 位、起点 16 字节对齐、段长 64KB、段间可重叠。

三、四个段寄存器各管什么

四个段寄存器各管一片：CS 取指令，DS 放数据，SS 走堆栈，ES 当备用。

每个段寄存器都是 16 位，里面装的是段基址，不是物理地址。

段寄存器	默认搭档	管什么
CS	IP	取指令，永远 CS:IP
DS	BX、SI、DI、直接地址	一般数据
SS	SP、BP	堆栈、局部变量
ES	数据寻址（写 ES:）	附加数据段

取指令只能用 **CS:IP**，不能换成别的段，也不能段超越。

访问数据默认用 DS。只有用 BP 时，默认段换成 SS。

想临时换段，就写「段寄存器:」，这叫**段超越**。

例：MOV AX, ES:[SI] ⇒ ES×16 + SI；DS=2100H、BX=0158H，MOV AX,[BX] ⇒ **21158H**。

> **小结：**取指 CS、数据 DS、堆栈 SS；默认段里只有 BP 是例外。

四、内存按字节编号，字要拆成两半

8086 按字节编址：一个地址对应一个字节，字占两个相邻字节。

1MB 就是 1048576 个单元，每单元 1 字节，从 00000H 编到 FFFFFH。

地址 20000H 指的是**一个字节**，不是一个字。字只是连续取两个字节。

存放字 645FH：**低字节 5FH 放 20000H，高字节 64H 放 20001H**，这叫**小端**。

数据线 16 根，一次能传 2 字节，所以存储器分成两个体：

体	存什么地址	数据线	选中条件
偶体	偶地址 0、2、4…	D7~D0	A0=0
奇体	奇地址 1、3、5…	D15~D8	BHE#=0

A0=0 且 BHE#=0 ⇒ 两个体一起动，一次读写一个字。

A0=1、BHE#=0 ⇒ 只动奇体，读写高字节。

A0=0、BHE#=1 ⇒ 只动偶体，读写低字节。

> **停一下：**字 1234H 从 20000H 开始放，20000H 和 20001H 各是什么？

低字节 34H 放 20000H；高字节 12H 放 20001H。

> **小结：**字节编址，小端低字节在前；A0 管偶体，BHE# 管奇体。

五、存储器芯片：SRAM 和 DRAM

存储器的两大指标是容量和速度；SRAM 不用刷新，DRAM 必须刷新。

内存是 CPU 能按地址直接访问的物理空间，放当前运行的程序和数据。

容量看能存多少字节（KB、MB），速度看读写一次要多久（ns）。

对比	SRAM	DRAM
存什么	触发器，静态	电容电荷，动态
刷新	**不需要**	**需要**，2ms 内刷完所有行
地址线	一次送全	**地址复用**，行、列分两次送
速度、价格	快、贵	慢、便宜、集成度高
例子	6264、6116	内存条里的芯片

DRAM 靠电容存电荷，电荷会漏掉，不补就丢数据，所以必须刷新。

地址复用是因为引脚不够：先送行地址给 **RAS#** 锁存，再送列地址给 **CAS#** 锁存。

所以 DRAM 的地址引脚数大约只要一半。

> **小结：**DRAM 两个关键词 —— 刷新、地址复用；SRAM 两个都不用。

必背 · 六条

物理地址 = 段 × 16 + 偏移	乘 16 就是末尾添 0
1MB = 00000H~FFFFFH	20 根地址线，2^{20} 字节
段起点 16 字节对齐	段可重叠、可滑动
CS 取指、DS 取数、SS 走栈	只有 BP 默认配 SS
字节编址，小端存放	低字节在低地址
SRAM 不刷新，DRAM 刷新	DRAM 还地址复用

综合 · 算 1 10 分　四问一次算完

已知 CS=2000H、IP=0100H；DS=2100H、BX=0158H；内存 20000H 起存放字 645FH。求：(1) 取指令的物理地址；(2) MOV AX,[BX] 的物理地址；(3) 20000H、20001H 两个字节的内容；(4) 2001H:0000H 与 2000H:0010H 是否同一单元。

第一步：认取指组合。取指令固定 CS:IP，段用 CS。

第二步：段乘 16，再加偏移。2000H×16=20000H，+0100H=**20100H**。

第三步：认数据组合。方括号里是 BX，默认段 DS。

第四步：算数据地址。2100H×16=21000H，+0158H=**21158H**。

第五步：拆字。645FH 的低字节是 5FH，高字节是 64H。

第六步：按小端放。20000H 存 **5FH**，20001H 存 **64H**。

第七步：验段重叠。2000H×16+0010H=20010H；2001H×16+0000H=20010H。

第八步：下结论。两个写法指向同一单元 **20010H**，只是偏移不同。

答案：20100H；21158H；5FH 和 64H；是同一单元 20010H。

验算：三个物理地址都在 00000H~FFFFFH 内；64H、5FH 按「高、低」拼回 645FH，没拆反。

⚠ **三个坑：**

坑 1 · 段值乘 10。必须乘 16（10H），2000H 乘 16 就是添 0 得 20000H。把 2000H 当十进制 2000 乘 10，得 20000（十进制）= 4E20H，整题全错。

坑 2 · 看见 BP 还用 DS。BP 默认配 SS。段选错，物理地址可能差 10000H，后面全废。

坑 3 · 以为段不能重叠。同一单元可写成多组「段:偏移」，如 2000H:0010H 和 2001H:0000H。硬说只有一种写法就错了。

自测 · 3 道

1. 8086 的地址空间多大？编号范围是什么？
2. CS=3000H，IP=0200H，物理地址是多少？
3. 段值 2000H 的段，末地址是多少？

1. 1MB，00000H~FFFFFH，共 2^{20}=1048576 个字节单元。

2. 3000H×16=30000H，+0200H=**30200H**。

3. 20000H + FFFFH = **2FFFFH**。

2024 卷 · 真题 2 道

2024 · 答2 8 分

○ 会 ○ 卡 ○ 不会

8086 的寻址范围有多大？如何实现？请以实例说明。

一句话：20 位地址线靠分段拼出来。 ① 20 根地址线，寻址 1MB，即 00000H~FFFFFH。 ② 寄存器只有 16 位，直接寻址最多 64KB。 ③ 于是分段：物理地址=段基址×16+偏移。 ④ 段基址由 CS/DS/SS/ES 给出，段可重叠可滑动。 ⑤ 实例：CS=2000H、IP=1000H ⇒ 21000H。 **关键：段基址左移 4 位，末 4 位恒为 0。 答案：1MB；物理地址=段×16+偏移，如 2000H×16+1000H=21000H。**

2024 · 设2 10 分

○ 会 ○ 卡 ○ 不会

用延迟线实现 DRAM 读写时序，画图说明。（图：RAS#、CAS#、WE#、行列地址、数据的时间关系）

一句话：用延时把 RAS# 和 CAS# 错开，做到"先行后列"。 ① DRAM 地址复用，行、列地址分两次送上同一组引脚。 ② RAS# 下降沿锁存行地址，CAS# 下降沿锁存列地址。 ③ 延迟线把 RAS# 延时后产生 CAS#，保证行地址先稳住。 ④ 写：WE# 在 CAS# 前有效，数据在 CAS# 前后保持稳定。 ⑤ 读：CAS# 有效后经访问时间数据才有效，CPU 此时采样。 ⑥ 收尾：先撤 CAS#，再撤 RAS#，最后撤数据。 ⑦ 刷新：只给 RAS# 就能整行刷新，2ms 内轮完全部行。 **答案：延迟线让 RAS# 先行、CAS# 后到，读写都在 CAS# 有效窗口内完成。**

2023 卷 · 真题 3 道

2023 · 填4 4 分

○ 会 ○ 卡 ○ 不会

DRAM 芯片的地址引线的特点是（ ）。

一句话：问 DRAM 芯片的地址引脚为什么只要一半。 ① DRAM 容量大，每单元独占一根地址线会使引脚爆炸。 ② 地址分两次送：先送行地址由 RAS# 锁存，再送列地址。 ③ 内部有行、列两个锁存器，同一组引脚分时复用。 **答案：地址线复用（行列地址分两次从同一组引脚输入）。**

2023 · 填5 4 分

○ 会 ○ 卡 ○ 不会

8086 微处理器读程序的内存地址计算方式：（ ）。

一句话：背 8086 取指令时的物理地址公式。 ① 8086 用"段地址:偏移地址"两级方式访问 1MB 空间。 ② 段地址左移 4 位（乘 16 或乘 10H）再加偏移。 ③ 取指令默认段是代码段 CS，偏移是指令指针 IP。 **答案：CS×16+IP（即 CS×10H+IP）。**

2023 · 算1 10 分

○ 会 ○ 卡 ○ 不会

BX=0158H，DI=10A5H，位移量=1B57H，DS=2100H，SS=1100H，BP=1011H。求下列指令的物理地址。

一句话：按"默认段×16+有效地址"把五条指令各算一遍。 ① 公式：物理地址 = 段基址×16 + 偏移。 ② 默认段：BX、DI、直接地址用 DS；BP 用 SS。 ③ MOV AX,[1B57H] → 21000H+1B57H=22B57H ④ MOV AX,[BX] → 21000H+0158H=21158H ⑤ MOV AX,[BP+1B57H] → 11000H+1011H+1B57H=13B68H ⑥ MOV AX,[BX][DI] → 21000H+0158H+10A5H=221FDH ⑦ MOV AX,[BP+DI+1B57H] → 11000H+1011H+10A5H+1B57H=14C0DH **答案：22B57H、21158H、13B68H、221FDH、14C0DH。**

2022 卷 · 真题 4 道

2022 · 填6 4 分

○ 会 ○ 卡 ○ 不会

存储器的主要性能指标是（ ）和（ ）。

一句话：问存储器最主要的两项指标。 ① 存储容量指能存多少字节，单位 KB/MB/GB。 ② 存取速度指读/写一次所需时间，越短越好。 ③ 两者互相制约，是选片的首要依据。 **答案：存储容量和存取速度。**

2022 · 填7 4 分

○ 会 ○ 卡 ○ 不会

动态 RAM 的一个重要问题是（ ）。

一句话：问 DRAM 为什么必须定时刷新。 ① DRAM 靠栅极电容存电荷，电荷会慢慢泄漏。 ② 所以必须定期读出再写回，这就是刷新。 ③ 典型要求 2ms 内把所有行轮流刷一遍。 **答案：刷新（不刷新数据会丢失）。**

2022 · 填8 4 分

○ 会 ○ 卡 ○ 不会

内存是存放（ ）信息的（ ）。

一句话：给"内存"下定义。 ① 内存即主存，CPU 可按地址直接访问。 ② 里面放当前正在运行的程序和数据。 ③ 它按单元编址，是真实的物理存储空间。 **答案：存放当前正在运行的程序和数据；物理存储空间。**

2022 · 填9 4 分

○ 会 ○ 卡 ○ 不会

8086 内存地址空间范围（ ），地址寻址范围共有（ ）个内存单元，简称内存是多少（ ）。

一句话：背 8086 地址范围与总容量。 ① 20 根地址线，00000H–FFFFFH。 ② 共 2^20 = 1048576 个单元，每单元 1 字节。 ③ 所以容量是 1MB，即 1024KB。 **答案：00000H–FFFFFH；2^20 个单元；1MB。**

2020 卷 · 真题 3 道

2020 · 选2 4 分

○ 会 ○ 卡 ○ 不会

8086CPU 系统主存储器以（ ）为单位编址。
A、字节 B、字 C、双字 D、八字节

一句话：确认 8086 是按字节编址的。 ① 1MB 空间按字节从 00000H 编到 FFFFFH。 ② 一个字 16 位，占两个相邻字节，低字节在低地址。 ③ 地址 20000H 指的是一个字节，不是一个字。 ④ 字、双字只是连续取几个字节，不改变编址单位。 **答案 A**

2020 · 选13 4 分

○ 会 ○ 卡 ○ 不会

与 SRAM 相比，下面哪一个不是 DRAM 的特点（ ）。
A、集成度高 B、功耗低 C、不需要刷新 D、每 MB 的造价更低

一句话：DRAM 必须刷新，说它"不用刷新"就错了。 ① DRAM 用电容存电荷，电荷会泄漏，必须周期刷新。 ② 一般 2ms 内把所有行刷新一遍，还占用总线。 ③ 集成度高、功耗低、每 MB 便宜确是它的优点。 ④ 题目问"不是特点"，只有 C 把话说反了。 **答案 C**

2020 · 填5 4 分

○ 会 ○ 卡 ○ 不会

字 645FH 在内存中的地址为 20000H，则 20000H 字节内容为_____，20001H 字节内容为_____。

一句话：小端方式，低字节放低地址。 ① 8086 采用小端存放，多字节数据低字节在前。 ② 645FH 的低字节 5FH 放 20000H。 ③ 高字节 64H 放 20001H。 ④ 读回时先取低地址当低位。 **答案 5FH；64H**

04 引脚与总线时序

约 12 分 · 35 分钟

40 根引脚，怎么同时送 20 位地址和 16 位数据

一句话：引脚不够就分时复用，一个总线周期用 T1~T4 四拍完成一次读写。

? 8086 只有 40 个引脚，却要送 20 位地址、16 位数据，还有一堆控制信号。
地址和数据挤在同一组线上，为什么不会打架？
如果地址只在 T1 出现一下，存储器怎么记住它？

先讲清楚

一、40 个引脚装不下，只能分时复用

地址线 20 根加数据线 16 根就是 36 根，再加电源、地、时钟、复位、中断，40 个引脚根本不够。

只能让同一组线分时干两件事，于是 8086 把引脚分成两组复用：

引脚	T1 时送什么	T2 以后送什么
AD0~AD15	地址 A0~A15	数据 D0~D15
A16~A19/S3~S6	地址 A16~A19	状态 S3~S6
BHE#/S7	奇偶体选择 BHE#	状态 S7

一个引脚在不同时间干不同事，这叫**分时复用**。

代价是：**地址只在 T1 出现**，T2 一到，AD 线上就换成数据。

存储器译码要地址稳定一整个周期，地址消失就全乱，所以必须把地址**锁**下来。

> 小结：复用省引脚，地址只在 T1 出现。

二、ALE 加 373，把地址锁住

ALE 在 T1 变高，告诉 373「现在总线上是地址」；ALE 下降沿一到，地址就被锁进 373。

74LS373 是 8 位锁存器：G 高时输出跟着输入走，G 一低就锁住。

接线：**G 接 ALE，OE# 接地**；输入接 AD0~AD15 和 A16~A19，输出 A0~A19。

ALE 高电平代表**地址有效**，不是数据有效。

数据线配一片 **8286** 双向缓冲器：DT/R# 控方向（读收、写发），DEN# 控使能。

> 停一下：ALE 为高电平时，AD0~AD15 上跑的是地址还是数据?

是**地址**。ALE 高对应 T1；T2 起 AD 线才换成数据。

> 小结：复用解决引脚不够，373 解决地址留不住；ALE 就是 373 的开关。

三、一个总线周期就是四拍

一个基本总线周期固定 4 个时钟周期：T1 送地址，T2 发控制，T3 传数据，T4 收尾。

拍	CPU 干什么	关键信号
T1	地址送上总线，同时锁存	A19~A0、BHE#、ALE 高
T2	撤地址，发读或写命令	RD# 或 WR#、DT/R#、DEN#
T3	数据稳定，完成读写	READY 在这拍采样
T4	撤销命令，释放总线	RD#、WR# 回高

一个 T 是一个时钟周期。5MHz 时，一个 T 是 200ns。

四拍 4×200ns = **800ns**，这是最短总线周期。

一条指令由若干个总线周期组成。**不同指令长度不一样**，没有统一的指令周期。

停一下：时钟 5MHz，一个基本总线周期多少纳秒?

T = 1 ÷ 5MHz = 200ns；4T = **800ns**。

> **小结：**T1 地址、T2 命令、T3 数据、T4 结束。

四、读周期和写周期，差在数据方向

读是「CPU 先给地址、存储器后给数据」；写是「CPU 把地址和数据一起摆出来」。

对比项	读周期	写周期
控制信号	RD#=0	WR#=0
DT/R#	**0**（收）	**1**（发）
数据谁驱动	存储器或外设	CPU
数据什么时候有	T3 前由对方送上	T2 就上总线，保持到 T4
什么时候生效	CPU 在 T3 末采样	存储器在 T3 取走

共同点：T1 都送地址和 BHE#，T2 都发命令，T4 都撤销命令。

M/IO# 一起看：**=1 访存储器，=0 访 I/O 端口**。DEN# 低有效，打开 8286 才通数据。

读周期 T2 要把 AD 线转成输入，有一段高阻；写周期 T2 直接送数。

> **停一下：**读周期里数据是谁送来的？DT/R# 应该给 0 还是 1？

数据由**存储器或外设**送来，CPU 接收，所以 DT/R#=**0**。

写周期才反过来：CPU 送出，DT/R#=1。

> **小结：**读 0 写 1 —— DT/R#：0 收、1 发；读在 T3 末采样，写在 T2 给数。

五、慢设备就插 Tw

设备来不及，就把 READY 拉低；CPU 在 T3 后插入等待周期 Tw，一个 T 一个 T 地等。

READY 是输入。CPU 在 T3 末尾检查它：高就进 T4，低就插一个 Tw。

每个 Tw 末尾再查，直到 READY 变高才走。**Tw 可以插任意多个。**

周期长度 = 4 + Tw 个 T。插 2 个 Tw 是 6 个 T，插 3 个就是 7 个。

例：5MHz 时钟，插 2 个 Tw，6×200ns = **1.2μs**。

> **小结：**READY 低就等，Tw 插在 T3 之后；慢多少，就插多少个。

六、最小模式和最大模式

MN/MX# 接高电平是最小模式，CPU 自己出控制信号；接低电平是最大模式，控制信号交给 8288。

最小模式用于单处理器，外围就是 373、8286、存储器和接口芯片。

最大模式用于多处理器或配 8087。**系统用的 MEMR#、MEMW# 改由 8288 产生。**

它把 S0#~S2# 状态码送给 8288，由 8288 译出存储器读、写、I/O 读、写和中断响应。

连 24~31 脚的定义也换了：最小模式出 ALE、DEN#、DT/R# 等，最大模式出 S0#~S2# 和总线请求。

但地址锁存和数据缓冲不变：**两种模式都要 373 和 8286**。

对比	最小模式	最大模式
MN/MX#	1（接 +5V）	0（接地）
控制信号	8086 自己产生	8288 产生
典型系统	单 CPU	多 CPU、配 8087

> **小结：**最小模式自己干，最大模式请 8288 帮忙；地址锁存两边都少不了。

必背 · 六条

总线周期 = 4 个 T	T1~T4，慢则加 Tw
T1 地址、T2 命令、T3 数据、T4 结束	地址锁存在 T1
Tw 插在 T3 之后	READY 低才插
ALE 高 = 地址有效	接 373 的 G
读 RD#=0、DT/R#=0；写 WR#=0、DT/R#=1	读收写发
MN/MX#=1 最小，=0 最大	最大用 8288

综合 · 时序 1 10 分 算时间、认信号

8086 时钟 5MHz。某次读周期中，READY 在 T3 之后连续 3 个 T 保持低电平。求：(1) 这个总线周期共几个 T、多长时间；(2) ALE、RD#、DT/R#、M/IO# 的电平和有效时刻；(3) 数据在哪一刻被采样。

第一步：算一个 T。T = 1 ÷ 5MHz = **200ns**。

第二步：认基本四拍。没有等待时是 T1~T4，共 4 个 T。

第三步：数等待。READY 低 3 个 T ⇒ 插 3 个 Tw。

第四步：算总拍数。4 + 3 = **7 个 T**。

第五步：算总时间。7×200ns = **1400ns = 1.4μs**。

第六步：认 ALE。T1 高有效，把地址锁进 373。

第七步：认读写信号。读周期 RD#=0，从 T2 起有效，T4 撤销；M/IO#=1 表示访存；DT/R#=0 表示收数据；DEN#=0 打开 8286。

第八步：认采样时刻。数据在 T3 末采样；有 Tw 时，在最后一个 Tw 末采样。

答案：7 个 T、1.4μs；ALE 在 T1 高有效，RD# 从 T2 低有效，DT/R#=0，M/IO#=1；数据在最后一个等待周期末采样。

验算：4T=800ns，3Tw=600ns，800+600=1400ns；7×200ns 也是 1400ns，两路对得上。

⚠ **三个坑：**

坑 1 · 把 Tw 插在 T2 后面。错。READY 要到 T3 才被采样，Tw 只能插在 T3 之后、T4 之前。

坑 2 · 以为 ALE 高表示数据有效。错。ALE 是地址锁存允许，T1 高电平代表 AD 线上是地址。

坑 3 · 读周期 DT/R# 给 1。错。0 收、1 发；读周期 CPU 收数据，必须给 0，给 1 会让 8286 往外推。

自测 · 3 道

1. 一个基本总线周期几个 T？T1 做什么？
2. READY 连续低 3 个 T，总线周期共几个 T？
3. 最大模式下，存储器读、写控制信号由谁产生？

1. 4 个，T1 送地址并靠 ALE 锁存进 373。

2. 4 + 3 = **7 个 T**。

3. 由 **8288** 总线控制器根据 S0#~S2# 译码产生。

2024 卷 · 真题 3 道

2024 · 填8 4 分

○ 会 ○ 卡 ○ 不会

ALE 作用（ ），READY 作用（ ）。

一句话：两个"救场"引脚各管什么。 ① AD15~AD0 分时复用，T1 送地址、T2 起送数据。 ② ALE 在 T1 高有效，把地址锁进 373。 ③ READY 为低时在 T3 后插 Tw，等慢速设备。 **答案：ALE 地址锁存允许；READY 等待请求，协调慢速存储器/外设。**

2024 · 填9 4 分

○ 会 ○ 卡 ○ 不会

一个总线周期包含（ ）个时钟周期。

一句话：问 T1~T4 这四拍。 ① T1 送地址，ALE 高有效锁存。 ② T2 撤地址，发 RD# 或 WR#。 ③ T3 采样数据，T4 结束；慢设备在 T3 后插 Tw。 **答案：4 个时钟周期（T1~T4），必要时插入 Tw。**

2024 · 答1 8 分

○ 会 ○ 卡 ○ 不会

分别画出 8086 读、写总线周期，并分析两者的不同。(图：横轴 T1~T4，画出 A19~A0、ALE、M/IO#、RD#、WR#、DT/R#、D15~D0)

一句话：画四拍，再对比 RD#/WR# 与数据出现的时刻。 ① T1：送地址 A19~A0，ALE 高有效把地址锁进 373。 ② T2：地址撤销，读周期发 RD#，写周期发 WR#。 ③ 读周期数据在 T3 由存储器送上，CPU 在 T3 末采样。 ④ 写周期数据在 T2 就上总线，保持到 T4。 ⑤ T4 撤销控制信号结束；M/IO#=1 选存储器，DT/R# 读 0 写 1。 **关键：读是"先给地址后收数据"，写是"地址与数据几乎同时给出"。 答案：读周期 RD# 有效、DT/R#=0，T3 末采样；写周期 WR# 有效、DT/R#=1，数据 T2 上总线保持到 T4。**

2023 卷 · 真题 3 道

2023 · 填3 4 分

○ 会 ○ 卡 ○ 不会

8086CPU 的许多引线具有双重定义和功能，采用（ ）的方式工作。

一句话：问 8086 引脚为什么要"一根线当两根用"。 ① 40 个引脚要送出 20 位地址和 16 位数据，装不下。 ② 于是同一组引脚在不同 T 状态下承担不同信号，如 AD0~AD15。 ③ 靠 ALE 锁存地址、靠时序区分信号类型。 **答案：分时复用。**

2023 · 答2 8 分

○ 会 ○ 卡 ○ 不会

一个总线周期有多长？其长度取决于哪些因素？CPU 是如何决定一个总线周期的长度的?

一句话：问一个总线周期几个 T、谁能让它变长。 ① 基本总线周期 = 4 个时钟周期 T1~T4。 ② 长度取决于 CPU 时序与存储器、外设的存取速度。 ③ T1 送地址，T2 撤地址发控制，T3 采样 READY。 ④ READY 为低就在 T3 后插入等待周期 Tw。 ⑤ 插几个 Tw 由慢速器件决定，周期 = 4+Tw 个 T。 ⑥ 实例：READY 低 2 个 T，周期就是 6 个 T。 **答案：基本 4 个 T；由器件速度决定；CPU 在 T3 采样 READY，为低则插 Tw。**

2023 · 答3 8 分

○ 会 ○ 卡 ○ 不会

8086 最小模式下 373 锁存器起什么作用？给出一种读或写的总线工作时序。（图：AD0~AD15 与 A16~A19/S3~S6 送入 74LS373，G 接 ALE、OE# 接地，输出 A0~A19；8286 接 D0~D15，方向由 DT/R# 控制。）

一句话：说清 373 为什么必须存在，再给一个写周期时序。 ① AD0~AD15、A16~A19/S3~S6 是地址与数据分时复用的。 ② T1 总线上是地址，T2 起变成数据，地址就消失了。 ③ 373 是带三态输出的 8 位锁存器，G 端由 ALE 控制。 ④ ALE 在 T1 有效，373 趁机锁住地址，输出稳定 A0~A19。 ⑤ 锁存后地址整周期不变，供存储器和外设译码。 ⑥ 写周期：T1 送地址并锁存，T2 发 WR# 且数据上线。 ⑦ T3 写入选通，T4 撤销控制信号，周期结束。 **答案：373 起地址锁存作用；写周期 T1 锁地址、T2 送数据发 WR#、T3 写入、T4 结束。**

2022 卷 · 真题 3 道

2022 · 选2 4 分

○ 会 ○ 卡 ○ 不会

以下说法正确的有几个：①一个总线周期包括 4 个时钟周期；②CPU 引脚有 20 根地址线、16 根数据线；③CPU 的指令周期都一样长。

一句话：逐条判断三句话的真假。 ① ①对：8086 一个基本总线周期就是 T1–T4。 ② ②对：确有 20 根地址线、16 根数据线。 ③ ③错：指令周期长短不一，不同指令差很多。 **答案：①② 正确，③ 错误。**

2022 · 选8 4 分

○ 会 ○ 卡 ○ 不会

如何设置 BHE# 和 A0（ ）可使 8086 一次读写两个字节。

一句话：做字访问时两个引脚该给什么电平。 ① A0=0 选通偶地址体，BHE#=0 选通奇地址体。 ② 两者同时为 0，两个存储体一起工作。 ③ 于是 D0–D15 一次传一个完整的字。 **答案：BHE#=0、A0=0。**

2022 · 答2 8 分 ○ 会 ○ 卡 ○ 不会

说明 8086 CPU 的读、写总线周期时序，并画出波形。

一句话：按 T1–T4 说清读/写周期各步动作。 ① T1：送地址 A19–A0 与 BHE#，ALE 同时有效。 ② T2：地址已锁存，AD 总线转高阻，发 RD# 或 WR#。 ③ T3：数据稳定，CPU 在 T3 末尾采样读入。 ④ T4：撤销 RD#/WR#，周期结束。 ⑤ 外设慢就在 T3 后插 Tw，靠 READY 低电平延长。 ⑥ 写周期在 T2 就把数据送上总线，保持到 T4。 **答案：T1 送地址、T2 发控制、T3 传数据、T4 结束，慢则插 Tw。**

2020 卷 · 真题 1 道

2020 · 选14 4 分 ○ 会 ○ 卡 ○ 不会

8086 总线周期时序中，锁存地址是在（ ）状态。
A、T1 B、T2 C、T3 D、T4

一句话：地址在 T1 送出，靠 ALE 锁存。 ① 一个基本总线周期由 T1、T2、T3、T4 组成。 ② T1 时 CPU 把 A19~A0 送上总线，ALE 变高。 ③ 地址只在 T1 有效，用 373 在 ALE 时锁存。 ④ T2 起地址让出，总线进入读写数据阶段。 **答案 A**

05 寻址方式

约 8 分 · 30 分钟

一个方括号，差出一次访存

一句话：寻址方式就是「操作数在哪、地址怎么算」，关键看方括号和里面的寄存器。

? MOV AX, BX 和 MOV AX, [BX] 只差一个方括号。
一个直接拿寄存器，一个要去内存里取数。
方括号里到底允许写什么？地址又是怎么拼出来的？

先讲清楚

一、操作数只可能在三个地方

操作数只可能在指令里、寄存器里、内存里；方括号就是「去内存」的记号。

如果只用固定地址，遍历 100 个字节的数组就得写 100 条指令，没法循环。

所以地址必须能算：基址加变址加位移，一条指令走遍整块数据。

操作数在哪	例子	访存吗
指令里（立即数）	MOV AX, 1234H	不访存
寄存器里	MOV AX, BX	不访存
内存里	MOV AX, [BX]	**访存**

立即数直接写在机器码里，只能当源操作数，不能当目的操作数。

没有方括号，操作数就在指令或寄存器里；有方括号，才去内存。

方括号里的东西不是数据，是用来**算地址**的线索。

停一下：MOV AX, 1234H 和 MOV AX, [1234H]，哪个要访存?

后者。**MOV AX, [1234H]** 去内存 1234H（还要加段）取数；前者 1234H 直接进 AX。

> 小结：先看有没有方括号；有，就是访存，接下来才算地址。

二、七种寻址方式，看方括号里放了什么

七种寻址方式的区别，全在方括号里放了什么：什么都没有、一个寄存器、两个寄存器、有没有常数。

名称	写法	EA	默认段
立即寻址	MOV AX, 1234H	无	不访存
寄存器寻址	MOV AX, BX	无	不访存
直接寻址	MOV AX, [1B57H]	1B57H	DS
寄存器间接	MOV AX, [BX]	BX	DS
寄存器相对	MOV AX, [BP+1B57H]	BP+1B57H	SS
基址变址	MOV AX, [BX][DI]	BX+DI	DS

相对基址变址	MOV AX, [BP+DI+1B57H]	BP+DI+1B57H	SS

[BX][DI] 也能写成 [BX+DI]，两种写法等价，都表示两个寄存器相加。

寄存器间接除了 [BX]，还可以用 [SI]、[DI]；用 [BP] 时段换成 SS。

名字有规律：**带常数叫「相对」，两个寄存器叫「基址变址」，两样都有叫「相对基址变址」。**

> **小结：**先数方括号里有几个寄存器、有没有常数，名字和 EA 就都出来了。

三、EA 怎么拼，哪些组合非法

EA = 基址 + 变址 + 位移；基址只能用 BX 或 BP，变址只能用 SI 或 DI。

成分	只能放	作用
基址	**BX、BP**	指向数据块开头
变址	**SI、DI**	在块内移动
位移	8 位或 16 位常数	固定偏移

一个方括号里，基址最多一个、变址最多一个、位移一个，三者相加。

合法组合只有这些：BX、BP、SI、DI、BX+SI、BX+DI、BP+SI、BP+DI，每个都能再加位移。

非法：[AX+BX]、[SI+DI]、[BX+BP]、[CX+SI]。AX、CX、DX 不能参与算地址。

为什么限制？指令编码里拿来选寄存器的位有限，8086 只给这 4 个地址寄存器留了位置。

EA 是 16 位，加出来超过 FFFFH 就只留低 16 位，这叫**回绕**。

> **停一下：**[BX+BP] 合法吗？[BP+SI] 默认用哪个段？

[BX+BP] **非法**：两个都是基址，不允许。

[BP+SI] 合法，里面有 BP，默认段是 **SS**。

> **小结：**基址 BX/BP、变址 SI/DI，一个基址配一个变址，再加常数。

四、默认段和段超越

数据寻址的默认段只有一条例外：带 BP 用 SS，其余全用 DS。

寻址形式	默认段

CS:IP 取指令	CS（不能超越）
[BX]、[SI]、[DI]、[BX+SI]、[BX+DI]、直接地址	DS
[BP]、[BP+SI]、[BP+DI]	**SS**
PUSH、POP 用的 SP	SS

想临时换段，就写「段寄存器:」，这叫**段超越**。

例：MOV AX, ES:[SI] ⇒ 物理地址 = ES×16 + SI，不再用 DS。

什么时候必须写？用 BP 却想访问 DS 段，就要写 DS:[BP+...]。

默认段可以省，改默认段必须写。取指令的 CS 不能超越。

停一下：SS=1000H，BP=0100H，MOV AX,[BP] 的物理地址是多少?

有 BP，段用 SS：1000H×16=10000H，+0100H=**10100H**。

> 小结：BP 配 SS，其余配 DS；要改就写段前缀，CS 改不了。

五、算物理地址的三步固定动作

不管题多绕，永远三步：算 EA、定段、段×16 加 EA。

1. 看方括号，算 EA：基址 + 变址 + 位移，只保留 16 位。
2. 定段：有 BP 用 SS，否则 DS；有段前缀就按前缀。
3. 段基址 × 16，再加 EA，得 20 位物理地址。

十六进制加法一位一位加，别跳位；EA 超过 FFFFH 就把最高位丢掉。

例：DS=2000H，BX=0100H，SI=0020H，MOV AX,[BX+SI+10H]。

EA=0100H+0020H+0010H=0130H；段 2000H×16=20000H；物理地址 **20130H**。

方括号里有两个寄存器加一个常数，所以是**相对基址变址寻址**。

最后写两样：**寻址方式名称**和**物理地址**。名字说明怎么找，地址说明在哪。

> 小结：EA 是偏移，不是物理地址；算完 EA 还必须加段。

必背 · 六条

方括号 = 访存	没括号就是立即数或寄存器
EA = 基址 + 变址 + 位移	先算 EA，再拼段
基址 BX/BP，变址 SI/DI	不能乱配
有 BP 用 SS，其余 DS	唯一例外
物理地址 = 段 × 16 + EA	段末尾添 0
改默认段写「ES:」	默认段可省

2023 · 算 1 10 分 五条指令一起算

DS=2100H，SS=1100H，BX=0158H，DI=10A5H，BP=1011H，位移量=1B57H。求下列指令的寻址方式和物理地址：(1) MOV AX,[1B57H]；(2) MOV AX,[BX]；(3) MOV AX,[BP+1B57H]；(4) MOV AX,[BX][DI]；(5) MOV AX,[BP+DI+1B57H]。

第一步：立规则。EA=基址+变址+位移；带 BP 用 SS，其余用 DS。

第二步：第 1 条是直接寻址。EA=1B57H，段 DS：21000H+1B57H=**22B57H**。

第三步：第 2 条是寄存器间接。EA=0158H：21000H+0158H=**21158H**。

第四步：第 3 条是寄存器相对，带 BP。EA=1011H+1B57H=2B68H；11000H+2B68H=**13B68H**。

第五步：第 4 条是基址变址。EA=0158H+10A5H=11FDH；21000H+11FDH=**221FDH**。

第六步：第 5 条是相对基址变址，带 BP。EA=1011H+10A5H+1B57H=3C0DH；11000H+3C0DH=**14C0DH**。

第七步：写名称。依次是直接、寄存器间接、寄存器相对、基址变址、相对基址变址。

答案：22B57H、21158H、13B68H、221FDH、14C0DH。

验算：第 5 条换顺序，10A5H+1B57H=2BFCH，+1011H=3C0DH，仍是 14C0DH；五个结果都在 1MB 内。

⚠ **三个坑：**

坑 1 · 带 BP 还用 DS。[BP+DI+1B57H] 默认段是 SS。按 DS 算得 24C0DH，按 SS 才是

14C0DH，差 10000H。

坑 2 · 把 [AX+BX] 当合法。AX 不是基址也不是变址，不能参与算地址；[SI+DI] 也不行，两个都是变址。

坑 3 · 算完 EA 就停手。EA 只是偏移。EA=3C0DH 不是物理地址，还要加 SS×16，才是 14C0DH。

自测 · 3 道

1. DS=2000H，MOV AX,[1B20H] 属于哪种寻址？物理地址是多少？
2. [BP+SI] 的默认段是哪个？为什么？
3. 下列哪个组合非法：A、[BX+SI] B、[BP+DI] C、[AX+BX] D、[SI+10H]

1. 直接寻址；20000H+1B20H=**21B20H**。

2. SS。里面有 BP，BP 默认配 SS。

3. C。AX 不能当基址或变址。

2022 卷 · 真题 2 道

2022 · 选7 4 分

○ 会 ○ 卡 ○ 不会

寄存器间接寻址，操作数存放在（ ）中。

一句话：问寄存器间接寻址的数据到底在哪。 ① 有效地址 EA 放在寄存器里，如 [BX]、[SI]。 ② 操作数本身在内存单元中，由该 EA 指向。 ③ 寄存器存的是地址而不是数。 **答案：在内存单元中。**

2022 · 答3 8 分

○ 会 ○ 卡 ○ 不会

写出下列 5 条指令的寻址方式并求物理地址。DS=2000H、SS=1000H、DI=0100H、BX=0156H、BP=0100H。

一句话：套"段基址×16+偏移"算五个物理地址。 ① 直接寻址：2000H×16+1B20H=21B20H。 ② 寄存器间接 BX：2000H×16+0156H=20156H。 ③ 寄存器相对 BP：1000H×16+0100H+1200H=11300H。 ④ 基址变址 BX+DI：2000H×16+0156H+0100H=20256H。 ⑤ 相对基址变址：1000H×16+0100H+0100H+1257H=11457H。 ⑥ 关键：BP 默认配 SS，其余默认配 DS。 **答案：21B20H、20156H、11300H、20256H、11457H。**

2020 卷 · 真题 1 道

2020 · 选5 4 分

○ 会 ○ 卡 ○ 不会

下列寻址方式中，段前缀不可省略的是（ ）。
A、DS:[BX+ST] B、DS:[BP+ST] C、SS:[BP] D、DS:[DI]

一句话：弄清各寻址方式的默认段寄存器。 ① 默认段：BX/SI/DI 配 DS，BP 配 SS。 ② C 项 SS:[BP] 本来就默认 SS，写不写一样。 ③ D 项 DS:[DI] 与 A 项默认就是 DS，可省。 ④ B 项用 BP 却要落 DS 段，必须显式写 DS:。 **答案 B**

06 指令系统

约 6 分 · 30 分钟

CPU 一辈子只干四件事：搬数、算数、比大小、改路线

一句话：指令分四类 —— 搬、算、串、跳；**算完留下的六个小开关叫标志位**，跳转全靠偷看它。

? 你写下 ADD AX,BX，CPU 除了把和塞进 AX，还顺手拨了 6 个开关。
下一句 JZ 就要看其中一个。**谁拨的、拨成什么，就是这一讲的全部内容。**

先讲清楚

一、一条指令 = 做什么 + 对谁做

任何一条指令都拆成两半：操作码说"做什么"，操作数说"对谁做"。

MOV 是"搬"，AX 是"搬到哪"，BX 是"从哪搬"，三块拼齐才是一条命令。

操作数只可能待在三处：寄存器写名字、存储器加方括号、立即数直接写常数。

方括号是"去这个地址把货取出来"，没括号才是"这个数本人"。

MOV AX,[1200H] 拿内容，MOV AX,1200H 给地址。差一个括号，差很远。

二、数据传送：只搬不算，一个标志都不许动

MOV 只搬数据，绝不改标志位。它没做运算，谈不上进位或溢出。

指令	干什么	要记的点
MOV dst,src	src 复制到 dst	**两边不能都是内存**
PUSH src	压栈	**只能压字**，SP 先减 2
POP dst	出栈	先取数，SP 再加 2
LEA dst,src	取**地址**	不取内容
IN / OUT	读 / 写端口	端口号 > 255 要先放 DX

PUSH 最容易记反：**先 SP−2，再放数**。栈往低地址长，栈顶是最新那个字。

LEA 和 MOV 差在哪？设 BX=1000H，该单元存着 55H：MOV AX,[BX] 得 **0055H**，LEA AX,[BX] 得 **1000H**。

> 停一下：PUSH AX 之后紧跟 POP BX，AX 和 BX 各是什么？

BX 变成原来 AX 的值，AX 一点没变，SP 也回到原处。

所以这两句等价于 **MOV BX,AX**，不是 XCHG —— XCHG 会把 AX 也改掉。

三、算术运算：顺手拨了六个开关

算术指令算完会改标志位，最要紧的四个是 CF、ZF、SF、OF。

为什么非要有标志位？CPU 算完只留结果，"有没有进位""是不是零"没地方放，就塞进标志寄存器。

标志	名字	什么时候是 1
CF	进位	无符号数最高位往上**进 / 借**了一位

ZF	零	结果**正好等于 0**
SF	符号	结果最高位是 1（看着像负数）
OF	溢出	有符号数算出来**超出表示范围**

CF 和 OF 最容易被当成一回事，其实它们回答两个不同问题：**CF 管无符号，OF 管有符号。**

7FH 加 01H 得 80H：最高位没往前进位 ⇒ CF=0；但 127+1=128 超出 +127 ⇒ OF=1。

算术指令的落点：ADD/SUB 四个标志全改；**INC/DEC 单独不动 CF**；MUL 结果在 AX 或 DX:AX；CMP 只减不存。

INC 为什么不碰 CF？做 32 位加法时，低 16 位的进位要留给高位，被冲掉就全错。**这是特意留的口子。**

> 停一下：CMP 为什么不干脆用 SUB?

你只想比大小，不想把原数冲掉。

SUB 会把差写回目的操作数，原来的数就毁了；CMP 只让标志位说话。

四、逻辑与移位：一位一位地动手

逻辑指令按位算，移位指令把整串位往一边推。

指令	规则	典型用途
AND	有 0 就 0	**把某几位清零**
OR	有 1 就 1	**把某几位置 1**
XOR	不同为 1	某几位取反；自己异或自己得 0
SHL / SHR	右边 / 左边补 0	无符号数 ×2 / ÷2
SAL / SAR	SAR 补**符号位**	有符号数 ×2 / ÷2

TEST 和 CMP 是一对：**一个"只测不存"的 AND，一个"只减不存"的 SUB。**ROL/ROR 是循环移位，移出的位绕回另一端。

为什么分 SAL 和 SHR？−8 右移一位该得 −4，SHR 补 0 会得 124，只有 SAR 补符号位才对。

五、串操作：一次搬一批，指针自己走

串操作让 CPU 自动挪指针，一条指令处理一整串数据。

没有它，复制 100 字节要写 100 条 MOV；现在一句 REP MOVSB 就够。

指令	搬什么	指针怎么走
MOVS	[SI] → [DI]	SI、DI 都动
LODS	[SI] → AL / AX	只动 SI
STOS	AL / AX → [DI]	只动 DI
CMPS	[SI] – [DI]	都动，只改标志
SCAS	AL / AX – [DI]	只动 DI，只改标志

指针加还是减，由 **DF** 决定：**DF=0 往上加，DF=1 往下减**。CLD 清 0，STD 置 1。

加多少看后缀：MOVSB 加 1，MOVSW 加 2。**B 是字节，W 是字。**要重复就加 REP，它拿 **CX** 计数。

六、控制转移：不按顺序走，就这几招

控制转移只干一件事：改 IP，让下一条指令换个地方取。

指令	怎么改	典型场合
JMP	无条件跳	跳过一段、回循环头
Jcc	看标志，成立才跳	if、循环出口
LOOP	CX 减 1，不为 0 就跳	计数循环
CALL / RET	压返回地址 / 弹回来	子程序
INT / IRET	压标志和返回地址	中断

CALL 和 JMP 只差一处：**CALL 会把下一条指令的地址压栈**，RET 才能回来。JMP 有去无回。

Jcc 只能段内短跳，位移 –128 ~ +127。目标太远就**先用 Jcc 跳到附近，再用 JMP 过去**。

LOOP 只认 CX，而且**先减再判断**。进循环前 CX=0 会减成 0FFFFH，白转 65536 圈。

> **小结：**传送搬数据、算术逻辑造标志、串操作批量搬、控制转移看标志改 IP。

必背 · 六条

MOV 两边不能都是内存	段寄存器也不收立即数
PUSH：SP 先减 2	POP：先取再 SP+2
LEA 取地址，MOV 取内容	差一个方括号
INC / DEC 不动 CF	其余标志照改
CMP = 减法但不存结果	只留标志
DF=0 加，DF=1 减	REP 用 CX 计数

标志位 8 分 一条一条算

指令序列：MOV AL,0FFH ； ADD AL,01H ； INC AL。逐条写出 AL 与 CF、ZF、SF、OF、PF。

第一步：MOV 放数。AL=0FFH（1111 1111B），标志一个都不动。

第二步：ADD 相加。FFH+01H=100H 要 9 位，AL 只留低 8 位 00H。

第三步：定 CF。最高位往上进出去一个 1 ⇒ CF=1。

第四步：定 ZF、SF。结果 00H ⇒ ZF=1；最高位是 0 ⇒ SF=0。

第五步：定 OF。按有符号看 −1+1=0 没出界 ⇒ OF=0。**有进位不等于有溢出。**

第六步：定 PF。00H 里 1 的个数是 0，偶数 ⇒ PF=1。

第七步：INC 加一。00H+1=01H ⇒ ZF=0、SF=0、PF=0（1 个 1）。

第八步：OF 仍为 0。0+1=1 没出界。注意 INC 改的标志里**没有 CF**。

第九步：CF 保持 1。它还是第二步那个进位，INC 没动它。

验算：CF 和 AL 拼成 9 位是 1:01H=257；原式 255+1+1=257 ✓ 进位没丢。

⚠ **三个坑：**

① **写 MOV [BX],[SI]。**8086 没有内存到内存的通路，两边都是内存就没法搬，必须借 AX 中转。

② **以为 INC 会改 CF。**多字节加法靠 CF 串进位，一句 INC 就把它冲掉，高位直接算错。

③ **拿 OF 当 CF 用。**7FH+1 有 OF=1 却 CF=0，查进位却看 OF，多字节加法全废。

自测 · 3 道

1. MOV AX,[BX] 和 LEA AX,[BX] 差在哪?
2. 执行 INC AX 后 CF 变不变? 为什么这样设计?
3. REP MOVSB，CX=8，DF=0，跑完 SI 和 DI 各变多少?

1. MOV 搬单元里的内容；LEA 搬地址本身。

2. 不变。这样多字节加法里刚产生的进位不会被冲掉。

3. 各加 8。DF=0 往上加，MOVSB 每次走 1 字节。

2023 卷 · 真题 1 道

2023 · 填6 4 分

○ 会 ○ 卡 ○ 不会

一条指令包括（ ）和（ ）两部分。

一句话：问一条指令由哪两部分拼成。 ① 操作码说明"做什么"，决定指令功能。 ② 操作数说明"对谁做"，给出源/目的操作数或地址。 ③ 有的指令没有操作数（如 NOP），但操作码一定不能少。 **答案：操作码和操作数。**

2022 卷 · 真题 1 道

2022 · 填3 4 分

○ 会 ○ 卡 ○ 不会

指令是（ ）(写定义)。

一句话：给"指令"下一个准确定义。 ① 指令是 CPU 能直接识别并执行的命令。 ② 它用二进制代码表示，规定做什么操作。 ③ 一条指令分操作码和操作数两部分。 **答案：规定计算机执行某种操作的二进制代码。**

2020 卷 · 真题 4 道

2020 · 选3 4 分

○ 会 ○ 卡 ○ 不会

下列指令中，有语法错误的是（ ）。

A、MOV [SI],AL　B、IN AL,DX

C、MOV [BX+ST],78　D、MOV AX,[1200]

一句话：找出一条 8086 里根本写不出来的指令。 ① A 项 [SI] 把 AL 送入 SI 指向的单元，合法。 ② B 项 IN AL,DX 由 DX 给端口号，标准间接输入。 ③ C 项 [BX+ST] 里的 ST 不是 8086 寄存器，写法不成立。 ④ 基址变址只允许 BX/BP 配 SI/DI；D 项直接寻址合法。 **答案 C**

2020 · 选6 4 分

○ 会 ○ 卡 ○ 不会

连续执行 PUSH AX 和 POP BX 两条指令，其功能相当于执行（ ）指令。

A、MOV BX,AX　B、MOV AX,BX　C、XCHG AX,BX　D、XCHG BX,AX

一句话：先压栈再出栈，数据只是过了一手。 ① PUSH AX：SP 减 2，把 AX 压到栈顶。 ② POP BX：从栈顶弹出两字节送 BX，SP 加 2。 ③ 结果 BX 等于原 AX，AX 不变，堆栈恢复。 ④ 等价于 MOV BX,AX；XCHG 会把 AX 也改掉，不对。 **答案 A**

2020 · 选8 4 分

○ 会 ○ 卡 ○ 不会

8086 指令系统的指令是（ ）。

A、单字节指令　B、双字节指令　C、固定长指令　D、可变长指令

一句话：8086 指令长度 1~6 字节，不固定。 ① 最短 1 字节，如 NOP、XLAT。 ② 常规寄存器传送多为 2 字节。 ③ 带位移量或立即数的会加长，最长 6 字节。 ④ 长度不固定，所以叫可变长指令。 **答案 D**

2020 · 选10 4 分

○ 会 ○ 卡 ○ 不会

若寄存器中的无符号数末位为零，右移 1 位后，新数值是原数值的（ ）。

A、数值不变　B、1/2　C、2 倍　D、1/4

一句话：无符号数右移一位等于除以 2。 ① 右移时最低位移入 CF，高位补 0，每位权重减半。 ② 末位为 0，移出的是 0，没有信息丢失。 ③ 所以新值正好是原值的 1/2。 ④ 末位为 1 时，结果才是"除以 2 取整"。

答案 B

07

汇编程序与比较跳转

约 6 分 · 30 分钟

同一串 0 和 1，换个读法，大小关系就翻过来了

一句话：CMP 只减不存，跳转只偷看标志；**无符号看 CF，有符号看 SF 和 OF 同不同。**

? 一个字节 0FFH，当无符号是 255，当有符号是 −1。
拿它和 01H 比，谁大？**两种读法给出相反答案，而两条跳转指令都"对"。**

先讲清楚

一、CMP：减了，但把差扔掉

CMP a,b 就是做 a−b，算完把差扔掉，只留标志位。

为什么费这事？你要比大小，又不想把原数冲掉，差本身没用。

拿 AL=05H、BL=0AH 做 CMP AL,BL，逐个数：

标志	为什么变成这样
CF=1	5 减 10 不够减，向上借了一位
ZF=0	差是 0FBH，不等于 0
SF=1	差 0FBH 的最高位是 1
OF=0	5−10=−5，还在 −128~+127 之内

四个标志记住了，剩下全是翻译：**ZF=1 是相等，CF=1 是无符号的小于，SF≠OF 是有符号的小于。**

> 停一下：CMP AL,AL 之后，ZF 和 CF 各是多少?

差是 0 ⇒ **ZF=1**；两个数一样大，不用借位 ⇒ **CF=0**。

所以 CMP AL,AL 常用来测 AL 是不是 0。

二、无符号比较：只问一句"够不够减"

无符号数只看 CF：够减就是不小于，不够减就是小于。

为什么？把数想成一条 0 到 255 的刻度尺。小数减大数一定要借位，借了就说明小的排在前面。

含义	指令	满足的条件
大于	**JA**	CF=0 且 ZF=0
大于等于	JAE	CF=0
小于	**JB**	CF=1
小于等于	JBE	CF=1 或 ZF=1

JA 为什么多要求 ZF=0？因为相等不算"大于"，JA 是严格的。

05H 比 0AH：CF=1 ⇒ **JB 成立**，5 确实小于 10 ✓

助记：**A 是 Above，B 是 Below**，天生配"高低"，和正负号无关。

三、有符号比较：SF 会骗人，得拉上 OF

有符号数的"小于"，是 SF 和 OF 不相等；两者相等才是"不小于"。

为什么不直接看 SF？减法会溢出，溢出把符号位翻了面，SF 就说反话。

看 7FH 比 80H。按有符号读是 127 比 −128，127 明显更大：

量	值	说明
差	7FH−80H=0FFH	借了位，还落成"负数"
SF	1	看着像"小减大"
OF	1	127−(−128)=255，出界

结论	SF=OF	**其实不小于**

两个都是 1，一异或就抵消 —— 这就是全部秘密：**溢出把符号位翻了一次，那就再翻回来。**

同一对数按无符号读：CF=1 ⇒ JB 成立，7FH 确实小于 80H。

一条说大于、一条说小于，两条都没错，因为它们回答的是两个问题。

含义	指令	满足的条件
大于	**JG**	ZF=0 且 SF=OF
大于等于	JGE	SF=OF
小于	**JL**	SF≠OF
小于等于	JLE	ZF=1 或 SF≠OF

四、一张表管两套，别混用

先确定手上数据有没有符号，再选 A/B 系还是 G/L 系。

要判断	无符号	有符号
大于	JA（CF=0,ZF=0）	JG（ZF=0,SF=OF）
大于等于	JAE（CF=0）	JGE（SF=OF）
小于	JB（CF=1）	JL（SF≠OF）
小于等于	JBE（CF=1 或 ZF=1）	JLE（ZF=1 或 SF≠OF）
相等 / 不等	JE / JNE（ZF）	同左

停一下：数组里存的是成绩 0~100，比大小时该用哪套?

用无符号的 JA / JB。成绩是"数量"，不会取负值。

0~100 都小于 80H，两套碰巧一样；换成 ≥80H 的编码，就只有无符号那套对。

五、汇编源程序长什么样

能汇编的源程序 = 段定义 + 伪指令 + 指令，最后 END 收尾。

地址是按"段基址:偏移"拼的，代码和数据得各占一段。

写法	作用
DATA SEGMENT … ENDS	数据段，里面用 DB / DW 放数
CODE SEGMENT … ENDS	代码段，里面放指令
ASSUME CS:CODE,DS:DATA	告诉汇编器谁对应谁
MOV AX,DATA MOV DS,AX	**段寄存器不收立即数**，必须绕一道
END 标号	整个源程序到此结束

伪指令常用四个：

伪指令	干什么
DB / DW / DD	定义字节 / 字 / 双字数据
EQU	给常量起名，如 PORT EQU 60H
OFFSET	取偏移地址，**汇编时就算好**
PTR	临时指定类型，如 BYTE PTR [BX]

OFFSET 和 LEA 都能拿地址，差在时机：**OFFSET 是汇编时的常数，LEA 是运行时算出来的。**

六、读程序题：三步走完

读一段汇编只看三处：初值、循环体、出口条件。

初值看 MOV 给了什么（尤其 CX 和"擂主"），循环体看指针怎么挪，出口看 Jcc 什么时候不再跳。

LOOP 要单独记：它**先 CX−1，再看是不是 0**。所以 CX=4 正好转 4 圈，最后一圈 CX 变 0 就退出。

条件跳转只能跳 −128 ~ +127 字节。目标太远汇编器直接报错，得用 JMP 中转。

> **小结：**CMP 造标志，Jcc 读标志；无符号翻译成 CF，有符号翻译成 SF 与 OF。

必背 · 六条

CMP = 减法但不写回	只改标志
无符号小于 = CF=1	A/B 系
有符号小于 = SF≠OF	G/L 系
相等 = ZF=1	JE / JNE 两套通用
Jcc 范围 −128 ~ +127	只能段内短跳
OFFSET 汇编时，LEA 运行时	都拿地址

读程序 10 分　顺着标志走一遍

SI 指向 5 个字节：05H、0AH、03H、0CH、08H。下面这段跑完，AL 是多少?

```
MOV AL,[SI]
MOV CX,4
L1: INC SI
  CMP AL,[SI]
  JAE L2
  MOV AL,[SI]
L2: LOOP L1
```

第一步：看初值。MOV AL,[SI] 把第一个数 05H 请上来当擂主。

第二步：数圈数。CX=4，后面 4 个数各来挑战一次。

第三步：第一轮。INC SI 指向 0AH，CMP AL,[SI] 就是 05H−0AH。

第四步：读标志。5−10 借位 ⇒ CF=1，ZF=0。

第五步：判 JAE。JAE 要求 CF=0，这里不成立，所以不跳，落到 MOV AL,[SI]。

第六步：换人。AL 从 05H 变成 0AH，擂主更新。

第七步：第二轮。0AH−03H 够减 ⇒ CF=0 ⇒ JAE 成立，直接跳去 L2，擂主不变。

第八步：第三轮。0AH−0CH 借位 ⇒ CF=1 ⇒ 不跳，AL 换成 0CH。

第九步：第四轮。0CH−08H 够减 ⇒ CF=0 ⇒ 不换；LOOP 把 CX 减到 0，循环结束。

验算：AL=0CH=12，五个数 5、10、3、12、8 里最大就是 12 ✓

⚠ **三个坑：**

① **有符号无符号混用**。0FFH 比 01H，JB 说它小、JL 说它大，两句都执行了，程序就精神分裂。

② **以为 CMP 会改操作数**。它是"算完就扔"，AL 和 BL 原封不动，别指望差留在哪。

③ **以为 SF=1 就是小于**。7FH 比 80H 时 SF=1，可 127 大于 −128，溢出让符号位说了假话。

自测 · 3 道

1. CMP AL,BL 之后想表达"AL 无符号大于 BL"，用哪条?
2. SF=1 就一定能断定"小于"吗?
3. 要让下面 10 条指令重复 10 次，CX 该给多少？LOOP 怎么判断?

1. JA。要求 CF=0 且 ZF=0，无符号那套。

2. 不能。有符号要比 SF 和 OF，异号才是小于；只看 SF 会被溢出骗。

3. CX=10。LOOP 先减 1 再判 0，减到 0 就退出，所以正好 10 圈。

08 中断系统与中断向量表

约 10 分 · 35 分钟

CPU 正算着题被打断，回来还得接着算 —— 它靠什么记住路

一句话：中断 = 打断、记路、去服务、再回来；**类型号只干一件事：算出表里那 4 个字节在哪**。

? 键盘被按下，CPU 正在算一道除法。它凭什么知道该跳去哪个服务程序?
靠一个编号，去一张固定地址的表里查第 n 项。项里写着 CS 和 IP。

先讲清楚

一、为什么需要中断

中断是"有事你叫我"，让 CPU 不必死等外设。

不用中断会怎样？CPU 只能循环去问打印机"好了没"。这期间它一步也走不开，纯浪费。

有了中断，外设准备好就拉一下引脚，CPU 停下手上活去处理，处理完再回来。

对比	子程序 CALL	中断 INT

谁发起	程序自己安排	外部事件或异常逼的
压什么	只压返回地址	压返回地址 **+ 标志寄存器**
何时发生	执行到那一句	任意一条指令结束后
返回指令	RET	IRET

多压的那个标志寄存器是关键：服务程序可能改标志，不存下来，回来一判断就错了。

二、中断从哪来：内中断与外中断

CPU 内部产生的叫内中断，从引脚进来的叫外中断。

类别	来源	例子
内中断	CPU 自己	除法出错、单步、断点、溢出、INT n 指令
外中断	外部引脚	**NMI**（不可屏蔽）、**INTR**（可屏蔽）

INTR 能不能被挡住，看 **IF**：IF=1 才响应，IF=0 就当没听见。CLI 清 0，STI 置 1。

NMI 挡不住，因为它接的是掉电、内存出错这类晚一秒就完蛋的事。

> 停一下：执行 CLI 之后，NMI 还能进来吗?

能。CLI 只是把 IF 清 0，挡的是 INTR 这条路。

NMI 根本不看 IF，所以永远进得来 —— 这正是"不可屏蔽"的意思。

内中断不经过 INTA#：类型号是 CPU 内部直接给的，不用问外设要。

几个背下来的类型号：0 除法错、1 单步、3 断点、4 溢出、8 时钟、9 键盘。

三、响应一次中断，CPU 按顺序做五件事

响应 = 两个 INTA# 要来类型号 → 压三个字 → 查表跳过去。

1. 当前这条指令**执行完**才响应，不会干到一半就跑。
2. 看 IF 是否为 1，是 1 才理会 INTR。
3. 发**第一个 INTA#**：通知中断源"我收到了"，这个周期**不取数据**。
4. 发**第二个 INTA#**：中断源把**类型码**送上数据总线。
5. 压栈保护，然后查表，跳到服务程序。

第 5 步做得非常规矩，顺序是：**标志寄存器 → 清 IF 和 TF → CS → IP**。

为什么先清 IF？服务程序刚开个头就被同级打断，会没完没了。想允许嵌套，服务程序里自己 STI。

返回用 **IRET**，它反着弹三个字：IP、CS、标志寄存器。少弹一个，程序就跑飞。

四、中断向量表：类型号到入口的地址簿

表在 00000H~003FFH，256 项，每项 4 字节，第 n 项就在 n×4。

算一下：256 项 × 4 字节 = 1024 字节 = 1KB，正好占最低端那 1KB。

项目	值
位置	**00000H~003FFH**
表项个数	256（类型号 0~255）
每项多大	4 字节
第 n 项地址	**n×4**
项里存什么	低 2 字节 **IP**，高 2 字节 **CS**

为什么 IP 在前？CPU 按地址从低往高取字：先取到的那 2 字节当 IP，后取的当 CS。

每个字内部还是小端：**低字节在低地址**。所以 IP=0150H 存成 50H、01H 两个字节。

随手算三个：类型号 08H → 20H；21H → 84H；0FFH → 3FCH。

> **停一下：**256 项乘 4 字节，表为什么正好 1KB?

256×4=1024 字节，正好 1KB，所以表从 00000H 排到 003FFH，一个字节不多。

这也是为什么表项不能改成 2 字节：**CPU 硬件就按 ×4 去查**，改了就对不上号。

表里给的只是入口的"段:偏移"，真地址还要拼：**物理地址 = CS×16 + IP**。

五、优先级、嵌套与返回

高优先级的中断可以打断低优先级的服务程序，这叫中断嵌套。

几个中断同时来，谁先谁后由优先级定；8086 外面一般挂 8259A 来排这个队。

响应时 IF 已被清零，所以默认是"处理完这一个再理下一个"，不会套娃。

服务程序里执行 STI，才允许更高优先级的插进来；出口的 IRET 会把原来的 IF 还回去。

> **小结：**类型号 → n×4 → 取 4 字节 → 得 IP 和 CS → CS×16+IP 就是入口。

必背 · 六条

向量表 00000H~003FFH	1KB，256 项
第 n 项地址 = n×4	不是 n×2
低 2 字节 IP，高 2 字节 CS	低字节在低地址
响应送 2 个 INTA#	第二个才送类型码
入栈 FLAGS→CS→IP	IRET 反着弹
INTR 受 IF 管，NMI 不受	CLI / STI 只关 INTR

向量表 8 分 正向写、反向验

类型号 0AH 的服务程序入口是 2B00H:0150H。写出向量表里 4 个字节的地址和内容，再反算物理地址。

第一步：算表项地址。0AH×4=28H，所以从 00028H 开始占 4 个字节。

第二步：拆 IP。IP=0150H，低字节 50H、高字节 01H。

第三步：按小端写 IP。00028H=50H，00029H=01H，低字节在前。

第四步：拆 CS。CS=2B00H，低字节 00H、高字节 2BH。

第五步：接着写 CS。0002AH=00H，0002BH=2BH。

第六步：连起来看。28H~2BH 依次是 50H、01H、00H、2BH。

第七步：反算物理地址。2B00H×16=2B000H，加 0150H 得 **2B150H**。

第八步：验算拼接。取回 50 01 得 0150H，取回 00 2B 得 2B00H，和题目给的一致 ✓

第九步：验边界。最后一项 0FFH 落在 3FCH~3FFH，正好压住表尾 003FFH ✓

答案：00028H=50H、00029H=01H、0002AH=00H、0002BH=2BH，物理地址 2B150H。

⚠ **三个坑：**

① **INTR 和 INTA# 搞反。**INTR 是外设**提请求**的进线，INTA# 是 CPU **作答**的出线，一个是问、一个是答。

② **表项地址忘乘 4。**每项 4 字节不是一个字节；写成 n×2，类型号 08H 会算成 10H，差了整

整两项。

③ **IP 和 CS 写反**。低两字节必须是 IP，写反了 CS 变成 0150H，跳过去的位置南辕北辙。

自测 · 3 道

1. 类型号 20H 的表项首地址是多少?
2. 响应外部中断要送几个 INTA#? 第二个周期干什么?
3. IRET 从栈里弹出几个字? 顺序是什么?

1. 20H×4=80H，即 00080H。

2. 2 个。第二个周期中断源把类型码送上数据总线。

3. 3 个：先弹 IP，再弹 CS，最后弹标志寄存器。

2024 卷 · 真题 1 道

2024 · 填10 4 分

○ 会 ○ 卡 ○ 不会

一个中断响应周期包含（ ）个总线周期，第一个周期（ ），第二个周期（ ）。

一句话：两个 INTA# 各干什么活。 ① 第一个 INTA# 有效，通知中断源 CPU 已响应。 ② 该周期不取数据，只作应答。 ③ 第二个 INTA# 有效，中断源把类型码送总线。 **答案：2 个总线周期；第一个通知外设，第二个外设送类型码。**

2023 卷 · 真题 3 道

2023 · 填2 4 分

○ 会 ○ 卡 ○ 不会

8086 在响应外部中断请求时，由（ ）引脚送出（ ）个负脉冲。

一句话：数 8086 响应可屏蔽中断时 INTA# 打几个负脉冲。 ① 外部请求由 INTR 进来，CPU 响应后连发两个 INTA#。 ② 第一个 INTA# 通知中断源"已响应"，但不取数据。 ③ 第二个 INTA# 让外设或 8259 把中断类型码送上数据总线。 **答案：INTA# 引脚送出 2 个负脉冲。**

2023 · 填8 4 分

○ 会 ○ 卡 ○ 不会

8086 系统的断电保护的主要工作是（ ）。

一句话：问断电或中断前必须做的那件保命事。 ① 断电会打断程序，CPU 内部寄存器内容会被冲掉。 ② 必须把断点 CS:IP、标志寄存器、通用寄存器压栈保存。 ③ 恢复后依次弹出，程序从断点继续跑。 **答案：保存现场（保护断点和现场信息）。**

2023 · 填10 4 分

○ 会 ○ 卡 ○ 不会

8086 系统中中断向量表的作用：（ ），共包含（ ）个入口地址。

一句话：问中断向量表存什么、一共多少项。 ① 每个表项 4 字节：低 2 字节存 IP，高 2 字节存 CS。 ② 类型码 n 的表项首地址 = n×4。 ③ 类型码 0~255 共 256 种，故共 256 个入口。 ④ 表固定在 00000H~003FFH，共 1KB。 **答案：存放中断服务程序入口地址（CS:IP），共 256 个入口。**

2022 卷 · 真题 3 道

2022 · 选3 4 分

○ 会 ○ 卡 ○ 不会

8086 CPU 执行中断的引脚有哪些?

一句话：背与中断有关的三个引脚。 ① NMI：非屏蔽中断请求，上升沿触发。 ② INTR：可屏蔽中断请求，高电平有效，受 IF 控制。 ③ INTA#：中断响应信号，CPU 回两个负脉冲。 **答案：INTR、NMI、INTA#。**

2022 · 选4 4 分

○ 会 ○ 卡 ○ 不会

（ ）中断源不是内中断：A.单步　B.非屏蔽　C.溢出　D.软件

一句话：分清内中断（异常）与外中断。 ① 内中断由 CPU 内部产生：除法溢出、单步、软件中断。 ② 外中断来自芯片外部引脚：NMI 与 INTR。 ③ 非屏蔽中断从 NMI 引脚进来，属外中断。 **答案：B 非屏蔽中断。**

2022 · 选9 4 分

○ 会 ○ 卡 ○ 不会

中断向量表的存储空间是（ ）。

一句话：背中断向量表的位置和大小。 ① 共 256 个中断类型，每个占 4 字节。 ② 每项存 IP 和 CS 各 2 字节，共 1KB。 ③ 类型 n 的入口在 4n 处。 **答案：00000H–003FFH，共 1KB。**

2020 卷 · 真题 3 道

2020 · 选7 4 分

○ 会 ○ 卡 ○ 不会

中断向量存放在 0000H:003CH 开始的四个单元，依次为 34H、FEH、00H、0F0H，则类型码和入口地址分别为（ ）。

A、0EH,34FE:00F0H　B、0EH,F000:FE34H

C、0FH,F000:FE34H　D、0FH,00F0:34FEH

一句话：从向量表地址反推类型码和入口。 ① 类型码 = 向量地址 ÷ 4 = 003CH ÷ 4 = 0FH。 ② 低两字节是 IP，高两字节是 CS，小端存放。 ③ IP = FE34H，CS = F000H。 ④ 入口写作 段:偏移，即 F000:FE34H。 **答案 C**

2020 · 选12 4 分

○ 会 ○ 卡 ○ 不会

对于以下各种中断，只有（ ）需要硬件提供中断类型码。

A、INTO　B、INT n　C、NMI　D、INTR

一句话：只有 INTR 的类型码来自外部芯片。 ① 软件中断 INT n 的 n 由指令直接给出。 ② INTO 固定 4 号，除法出错固定 0 号，NMI 固定 2 号。 ③ INTR 的类型码由 8259A 在第二个 INTA# 送上数据总线。 ④ 所以需要硬件提供类型码的只有 INTR。 **答案 D**

2020 · 填4 4 分 ○ 会 ○ 卡 ○ 不会

可屏蔽中断 INTR 在中断请求时，CPU 无响应的最可能的原因是_____。

一句话：INTR 被 IF=0 屏蔽在门外了。 ① INTR 是可屏蔽中断，响应与否由 IF 决定。 ② IF=0 表示关中断，请求被挡住。 ③ 其次才是正在执行不可中断指令或更高级中断。 ④ 题目问"最可能"，标准答法就是 IF=0。
答案 IF=0（CPU 处于关中断状态）

09

8259A 可编程中断控制器

约 10 分 · 35 分钟

八路外设抢一根中断线，谁先说话、CPU 怎么知道是谁

一句话：8259A 是 8 路"叫号机"——**排队、屏蔽、报号**：4 个 ICW 配一次，3 个 OCW 随时调。

? 键盘、硬盘、网卡、打印机，八路设备都想打断 CPU。
可 CPU 只有一根 INTR，**既不知道谁在举手，也不知道该先理谁**。
8259A 就插在中间当"前台"。

先讲清楚

一、为什么需要 8259A

没有它，八路设备没法共用一根中断线。三个麻烦全落在它身上：判优、屏蔽、报号。

它的活	具体怎么做
判优	IR0 最高、IR7 最低，同时来先服务号小的
屏蔽	IMR 里写 1 的那一路，举手也不理
报号	把 IRn 翻成"基址 + n"的类型码送 CPU

芯片里还记着三本账，命令字写的都是它们：

名字	记什么	怎么变
IRR	谁在举手	来信号置 1，被响应后清 0

ISR	谁在处理	响应时置 1，收到 EOI 才清 0
IMR	谁被拉黑	由 OCW1 直接写

只接 IR3，它一按，IRR 变成 **0000 1000B**；CPU 响应后 IRR 清、ISR 置位。

服务程序末尾发 EOI，ISR 才清 0。**不发，这一路以后就永远静默。**

二、ICW1：初始化的大门钥匙

ICW1 一写，芯片就进入初始化模式，后面的字按顺序排队。

它靠 **D4=1** 认出自己，端口用 **A0=0**；此后 A0=1 的写操作，依次是 ICW2、ICW3、ICW4。

位	名字	管什么
D0	IC4	1 = 后面还要写 ICW4
D1	SNGL	0 = 级联，1 = 单片
D3	LTIM	1 = 电平触发，0 = 边沿触发
D4	—	**固定 1**，标志"我是 ICW1"

没列的位写 0。

例：边沿触发 + 级联 + 要 ICW4 ⇒ **0001 0011B = 13H**。

三、ICW2：类型码基址

ICW2 只干一件事：规定 IR0 对应哪个类型码。

余下七路不用写，芯片自己加：**IRn = 基址 + n**。例：写 40H ⇒ IR3=43H、IR7=47H。

IR 脚	IR0	IR1	IR3	IR6	IR7
类型码	40H	41H	43H	46H	47H

低 3 位为什么要求写 0？这三位由 IR 编号填，你写多少都会被盖掉。

CPU 拿到 43H，就去向量表 **43H×4 = 10CH** 取入口地址。

> **停一下：**基址 40H，IR5 的类型码是多少？入口地址存在哪四个字节？

40H + 5 = **45H**；45H×4 = **114H**，占 114H~117H。

四、ICW3 和 ICW4：级联与收尾

ICW3 说"谁挂在谁身上"，ICW4 说"用哪种 CPU、怎么结束"。

ICW3 有两种写法，因为主片和从片要的信息正好相反。

身份	写什么	例
主片	哪几个 IR 挂了从片，对应位写 1	IR0、IR1 ⇒ **03H**
从片	自己挂在主片第几号 IR，就写编号	挂 IR0 ⇒ **00H**

单片时 SNGL=1，**ICW3 整条跳过**，写了反而错位。

位	名字	作用
D0	μPM	**1 = 8086 模式**，必写 1
D1	AEOI	1 = 自动结束；正常 EOI 写 0

例：8086 模式、正常 EOI、非缓冲 ⇒ **0000 0001B = 01H**。

> **停一下：**主片 ICW3 写成 05H，是什么意思?

05H = 0000 0101B：**IR0 和 IR2 各挂一片从片**，其余六路直连。

五、三个 OCW：运行中随时改

ICW 开机配一次，OCW 在程序里随时可写。

命令字	端口	干什么
OCW1	A0=1	写屏蔽字 IMR，1 = 屏蔽
OCW2	A0=0	发 EOI、改优先级
OCW3	A0=0	选择读 IRR/ISR、查询

A0=0 的端口挤着三个字，靠 **D4、D3** 认：**1x** 是 ICW1、**00** 是 OCW2、**01** 是 OCW3。

例：屏蔽 IR6、IR7 ⇒ OCW1 = **1100 0000B = C0H**，写 21H 口。

例：普通 EOI ⇒ **0010 0000B = 20H**，写 20H 口。

级联时 EOI 发两次：**先给从片，再给主片**。

停一下：ICW2 和 OCW1 都写 21H，芯片靠什么分清?

靠时机。初始化序列里 A0=1 收的是 ICW2、ICW3、ICW4。

四个 ICW 收完，A0=1 才轮到 OCW1。

六、初始化顺序：一步都不能错

顺序固定：ICW1 → ICW2 →（ICW3）→（ICW4），中间不许插别的写操作。

芯片内部有个计数器：写完 ICW1 等 ICW2，写完 ICW2 等 ICW3。

步	写什么	端口	什么时候写
1	ICW1	20H（A0=0）	总要写
2	ICW2	21H（A0=1）	总要写
3	ICW3	21H	只有级联才写
4	ICW4	21H	ICW1 的 D0=1 才写

完整例子：端口 20H/21H，边沿、级联、基址 40H、IR0/IR1 挂从片、8086 模式。

1. MOV AL,13H　OUT 20H,AL（ICW1）
2. MOV AL,40H　OUT 21H,AL（ICW2）
3. MOV AL,03H　OUT 21H,AL（ICW3，主片位图）
4. MOV AL,01H　OUT 21H,AL（ICW4）
5. MOV AL,C0H　OUT 21H,AL（OCW1）

必背 · 六条

ICW1：D4=1，写 A0=0	初始化的钥匙
类型码 = 基址 + IR 号	ICW2 定基址
主片 ICW3 写位图，从片写编号	03H / 00H
ICW4 的 D0 = 1	8086 模式
OCW1 屏蔽、OCW2 结束、OCW3 读状态	A0 是 1/0/0
ICW1→2→3→4	不能插队

8259A 初始化 10 分 一路走到底

主片端口 20H/21H，基址 40H，边沿触发，IR0、IR2 各挂一片从片。
从片基址顺排 48H、50H，8086 模式、正常 EOI。求主片四个 ICW、两从片 ICW3、从片 B 的 IR5。

第一步：ICW1。边沿 D3=0、级联 D1=0、要 ICW4 D0=1、D4=1 ⇒ **13H**，写 20H。

第二步：ICW2。基址 40H，低 3 位补 0 ⇒ **40H**，写 21H。

第三步：主片 ICW3。IR0、IR2 置 1 ⇒ 0000 0101B = **05H**。

第四步：从片 ICW3。从片 A 挂 IR0 写 **00H**；从片 B 挂 IR2 写 **02H**。

第五步：ICW4。D0=1，其余 0 ⇒ **01H**，写 21H。

第六步：排基址。主片 40H~47H，从片 A 48H~4FH，从片 B **50H~57H**。

第七步：求类型码。从片 B 的 IR5 = 50H + 5 = **55H**。

第八步：验算。主片 IR2=42H，与 55H 不撞车 ✓ 55H×4=154H 在表内 ✓

答案：主片 13H / 40H / 05H / 01H；从片 A 00H、从片 B 02H；IR5 = 55H。

⚠ **三个坑：**
① **漏写 ICW4。**ICW1 的 D0 已写 1，芯片就在等 ICW4；你不写，下一条屏蔽字被当成 ICW4 吃掉，整片不受控。
② **主从 ICW3 写反。**主片要位图（IR0、IR1 写 03H），从片要编号（挂 IR1 写 01H）。反了，主片指的和从片报的对不上，第二个 INTA# 取不到类型码。
③ **级联只发一次 EOI。**从片的 ISR 还亮着，那一路以后不再响应。顺序：先从片，后主片。

自测 · 3 道

1. ICW1 = 1AH，说明哪三件事?
2. 基址 40H，IR6 的类型码是多少？入口在向量表哪几个字节?
3. 运行中只屏蔽 IR3，写哪个命令字、哪个端口、什么值?

1. 1AH = 0001 1010B：电平触发、单片、不要 ICW4。

2. 46H；46H×4 = 118H，占 118H~11BH。

3. OCW1，端口 21H，值 F7H。

2024 卷 · 真题 1 道

2024 · 设3 10 分

○ 会 ○ 卡 ○ 不会

画出 3 片 8259A 级联的连接图，并分析中断处理过程。

一句话：主片带两从片，主片判优、从片报类型码。 ① 主片 IR0、IR1 各接一片从片的 INT，构成两级。 ② 主片 SP#/EN# 接 +5V，从片接地。 ③ 三片 CAS0~CAS2 并联，主片靠它指出选中的从片。 ④ 数据线 D7~D0 并联，各片 CS# 独立占不同端口。 ⑤ ICW3：主片 03H（IR0、IR1 挂从片），从片 00H、01H。 ⑥ 中断源：主片 IR2~IR7 六个 + 两从片 16 个 = 22 个。 ⑦ 请求路径：从片 INT→主片 IR→主片 INT→CPU 的 INTR。 ⑧ 第一个 INTA#：主片把从片 ID 送上 CAS0~CAS2。 ⑨ 第二个 INTA#：被选从片把类型码送数据总线。 ⑩ CPU 用 4n 查向量表；结束先给从片 EOI，再给主片 EOI。 **关键：级联时类型码由从片提供，主片只管判优和转发 ID。 答案：3 片管 7×3+1=22 个中断；主片判优、CAS 选从片、从片送类型码，EOI 发两次。**

2023 卷 · 真题 1 道

2023 · 算3 10 分

○ 会 ○ 卡 ○ 不会

用三片 8259 实现对 22 个可屏蔽中断请求的管理，说明配置方式与响应过程。

一句话：先用 7n+1 算片数，再配好 ICW3 说清响应过程。 ① 单片 8259 有 IR0~IR7 八个中断请求输入端。 ② 主片一个 IRi 要接从片 INT，这个脚就被从片占了。 ③ 每加一片从片净增 7 个：8+7(n−1)=7n+1。 ④ 取 n=3 得 7×3+1=22，正好 22 个。 ⑤ 配置：1 主 + 2 从，从片 INT 接主片 IR0、IR1。 ⑥ 主片 ICW3 写 03H（IR0、IR1 位置 1）。 ⑦ 从片 1 接 IR0 写 00H，从片 2 接 IR1 写 01H。 ⑧ 主片剩下 IR2~IR7 直连 6 个，6+8+8=22。 ⑨ 从片 INT 送主片 IRi，主片向 CPU 发 INT，CPU 回第一个 INTA#。 ⑩ 第一个 INTA# 期间主片把级联号送 CAS0~CAS2 选中从片。 ⑪ 第二个 INTA# 期间从片送类型码，CPU 查表转服务程序。 **答案：1 主 2 从；主片 ICW3=03H，从片 00H、01H；经 CAS 级联，两个 INTA# 取类型码。**

2022 卷 · 真题 2 道

2022 · 选10 4 分

○ 会 ○ 卡 ○ 不会

8259 中断类型码的初始化命令字是（ ）。

一句话：问四个 ICW 里谁管类型码。 ① ICW1 管触发方式、单片还是级联。 ② ICW2 写类型码基址，即 IR0 对应的类型码。 ③ ICW3 管级联，ICW4 管缓冲与自动结束。 **答案：ICW2。**

2022 · 设2 10 分

○ 会 ○ 卡 ○ 不会

系统有 10 个中断源，用 8259A 管理。(1) 需要几片，几个主几个从；(2) 画级联图；(3) 主从片 ICW3；(4) 主片 IR0 与 IR7 的类型码。

一句话：一片主片带一片从片，凑够 10 路中断。 ① 单片只有 8 路 IR，10 路不够，必须级联。 ② 1 主 1 从：主片 7 路 + 从片 8 路 = 15 路 ≥ 10。 ③ 从片 INT 接主片 IR0，主片这一路被从片占用。 ④ 主片 CAS0–CAS2 接从片 CAS0–CAS2。 ⑤ 两片数据线并联，主片 INT 接 CPU 的 INTR。 ⑥ CPU 的 INTA# 同时接主片和从片。 ⑦ 主片 ICW3=01H（bit0=1 表示 IR0 挂从片）。 ⑧ 从片 ICW3=00H（表示接在主片 IR0）。 ⑨ 主片 ICW2 定基址 40H，则 IR0=40H。 ⑩ 片内 IRn=基址+n，所以主片 IR7=47H。 ⑪ 从片基址顺排 48H，其 IR0–IR7 为 48H–4FH。 **答案：1 主 1 从；主片 ICW3=01H，从片 00H；IR0=40H、IR7=47H。**

10 I/O 接口与端口

约 4 分 · 25 分钟

外设都挂在总线上，CPU 凭什么只跟其中一个说话

一句话：端口是外设在 I/O 空间的门牌号；8086 用**独立编址**，靠 IN/OUT 按号取送。

? 键盘按一下，这个字节要进 CPU。
可总线上还挂着硬盘、网卡、打印机，它们都盯着同一组数据线。
CPU 怎么保证这一次读到的是键盘，不是别人？

先讲清楚

一、外设为什么要先经过"接口"

外设的电平、速度、数据格式都跟总线对不上，中间必须有接口。

键盘一秒几个字节，硬盘一秒几十兆；总线按自己的节拍走，快慢差太远，硬接就丢数据。

接口要干四件事，少一件就接不通：

活	不做会怎样
缓冲、锁存	数据在总线上只停几十纳秒，不锁住就没了
电平、格式	串口给的是串行比特，总线要 8 位并行
地址译码	不认门牌，CPU 一读，所有外设一起往线上送数
状态与命令	CPU 不知道外设忙不忙，也没法让它启动

小结：接口是外设的"翻译 + 门卫"，CPU 只见接口，不见外设。

二、一个接口里通常有三个端口

数据、状态、控制三类信息，各走各的门。

先分清两个词：**接口是那块电路，端口是板上能被 CPU 读写的寄存器。**

为什么三个端口不合用一个？方向不同、时机不同，混在一起就会读错。

端口	方向	装什么	打印机上的例子
数据	双向	真正要传的数据	要打的那个字符
状态	外设 → CPU	忙、闲、就绪、出错	bit7=1 表示"我忙着"
控制	CPU → 外设	启动、复位、选通	写 1 让它开始打印

所以打印机接口至少占三个地址，往往是连着的三个号：

数据 **378H**、状态 **379H**、控制 **37AH**，这就是并口的经典分配。

三、独立编址还是统一编址

独立编址：端口自己一套地址空间；统一编址：端口借用存储器地址。

8086 选的是独立编址，所以内存 1MB 和 I/O 64KB 是两套互不重叠的空间。

对比项	独立编址（8086）	统一编址
地址空间	单独 64KB，0000H~FFFFH	从内存里划一块

访问指令	专用 IN / OUT	普通 MOV，寻址方式随便用
怎么区分	M/IO# = 0 表示访端口	靠地址范围，不用另加信号
好处	不占内存，程序一眼看出是 I/O	指令多，寻址灵活
坏处	指令少，只有 AL/AX 能进出	吃掉内存地址，译码更麻烦

为什么分开？20 根地址线管 1MB 内存已经很紧，再切一块给 I/O 不划算。

而且 **M/IO#** 一拉低，整个系统就知道"这次找端口，不是找内存"。

> **停一下：**统一编址的系统里，读端口用 IN 还是 MOV?

用 **MOV**。端口就是一块内存单元，没有单独的 IN/OUT 指令。

四、IN 和 OUT：两条指令的规矩

端口号不超过 FFH 就写进指令，超过 FFH 必须先送 DX。

原因很土：指令编码里只给端口号留了 1 个字节，装不下 16 位。

写法	能寻址的端口	例子
IN AL/AX, imm8	00H~FFH	IN AL,60H
IN AL/AX, DX	0000H~FFFFH	MOV DX,3F8H；IN AL,DX
OUT imm8, AL/AX	00H~FFH	OUT 61H,AL
OUT DX, AL/AX	0000H~FFFFH	OUT DX,AX

两条死规矩，写错汇编直接报错：

- 数据只能是 **AL 或 AX**，不能是 BL、BX；
- 间接寻址只能用 **DX**，别的寄存器不行。

AL 配 8 位端口，AX 配 16 位端口，一次传两个字节。

> **停一下：**端口 3F8H，能直接写 IN AL,3F8H 吗?

不能。3F8H 超过 FFH，先 MOV DX,3F8H，再 IN AL,DX。

五、端口译码：给每个端口发门牌

译码就是：地址等于我这个号时，只让一片芯片的 CS# 变低。

做法和存储器译码一样，只多一条：**M/IO# 必须为 0**，否则访存也会选中端口。

例：给 200H~207H 八个端口译码，用一片 74LS138。

地址位	200H~207H 的取值	接到哪
A9	恒为 1	G1（要 1）
A15~A10、A8~A3	全为 0	或门 → G2B#（要 0）
M/IO#	访端口时为 0	G2A#（要 0）
A2~A0	000~111 变化	C、B、A

十二根固定位只要有一根是 1，或门输出 1，138 立刻停工。

于是 Y0#~Y7# 依次对应 **200H~207H**，一次只选中一个。

为什么只把 A2~A0 接 CBA？因为只有这三位在变，变一次就换一个端口。

口诀：**会变的接 CBA，固定的接使能，访端口别忘了 M/IO#。**

> 停一下：A9 为什么接 G1，不接 CBA?

A9 在这八个地址里恒为 1，是**固定位**，固定位管"准不准进"。

CBA 要接会变的 A2~A0，才能八选一。

六、数据怎么进出：三种方式

查询是 CPU 等，中断是外设叫，DMA 是别人替 CPU 搬。

方式	谁在忙	代价
程序查询	CPU 反复读状态端口	CPU 空转，效率最低
中断	外设就绪才打断 CPU	每次都要保护和恢复现场
DMA	DMAC 接管总线，内存与外设直搬	要抢总线，电路最贵，速度最快

键盘偶尔来一个字节，用中断；硬盘、网卡成块高速，用 DMA。

必背 · 六条

三种端口：数据、状态、控制	方向各不同
内存 1MB，I/O 64KB	两套空间
访端口时 M/IO# = 0	区分访存
≤FFH 直接寻址，超过 FFH 用 DX	分界线
IN/OUT 数据只能 AL 或 AX	死规矩
固定位接使能，可变位接 CBA	译码口诀

端口译码 8 分 从地址到指令

某接口占 200H~207H 八个端口，用 74LS138 译码。写出地址位分配和连线，再写出把 203H 端口内容读进 AL 的指令。

第一步：写二进制。200H = 0000 0010 0000 0000B，207H = 0000 0010 0000 0111B。

第二步：找可变位。首末只有低 3 位不同 ⇒ A2~A0 可变，A15~A3 固定。

第三步：读固定值。A9 = 1，A15~A10 与 A8~A3 全为 0。

第四步：接选择端。C=A2、B=A1、A=A0，Y0#~Y7# 依次是 200H~207H。

第五步：接使能。G1=A9；G2A#=M/IO#；G2B#=十二根固定位的或。

第六步：写指令。203H 超过 FFH，走 DX：MOV DX,203H；IN AL,DX。

第七步：验算。203H 的 A2A1A0=011=3 ⇒ 选中 Y3#；A9=1、其余固定位为 0 ⇒ 使能成立 ✓

答案：A2~A0 接 CBA，A9 接 G1，M/IO# 接 G2A#，其余固定位经或门接 G2B#；读 203H 用 MOV DX,203H / IN AL,DX。

⚠ **三个坑：**

① **端口超过 FFH 还写立即数。**"IN AL,3F8H" 汇编直接报错；有的汇编器截成低 8 位，你就读到了另一个设备。

② **把数据放进 BX 再 IN。**IN/OUT 只认 AL、AX，别的寄存器不行，这是指令编码定死的。

③ **译码漏掉 M/IO#。**访存时端口也被选中，读一次内存就误触发一次外设，系统看起来"随机死机"。

自测 · 3 道

1. 8086 的 I/O 空间多大？为什么正好是这么大？
2. 端口 2F8H，写出把它的内容读进 AL 的两条指令。
3. 独立编址靠哪根信号区分访存和访端口？

1. 64KB。I/O 只用 A15~A0 十六根地址线，$2^{16}=65536$。

2. MOV DX,2F8H；IN AL,DX。

3. M/IO#：为 0 访端口，为 1 访内存。

2023 卷 · 真题 1 道

2023 · 填9　4 分

○ 会　○ 卡　○ 不会

I/O 接口包括三种 I/O 端口：( )、( )、( )，端口通过（ ）与 CPU 进行通信。

一句话：背 I/O 接口的三种端口和它们的连接方式。 ① 数据端口：暂存 CPU 与外设之间传送的数据。 ② 状态端口：外设把"忙/闲/就绪"送给 CPU 读。 ③ 控制端口：CPU 把启动、复位等命令写给外设。 ④ 端口都挂在系统总线上，通过总线通信。 **答案：数据端口、状态端口、控制端口；通过总线。**

2022 卷 · 真题 1 道

2022 · 填10　4 分

○ 会　○ 卡　○ 不会

CPU 与外设传输的三种信息是（　）、（　）和（　）。

一句话：问 CPU 与外设间传哪三类信息。 ① 数据信息：真正要传的数据。 ② 状态信息：外设忙闲、就绪，如 READY。 ③ 控制信息：CPU 发给外设的命令。 **答案：数据信息、状态信息和控制信息。**

2020 卷 · 真题 5 道

2020 · 选11 4 分

○ 会 ○ 卡 ○ 不会

8086 的内存空间和 I/O 空间是（ ）。

A、单独编址，分别是 1MB 和 64KB　B、单独编址，都是 1MB

C、统一编址，都是 64KB　D、统一编址，都是 1MB

一句话：内存 1MB、I/O 64KB，两套空间分开编址。 ① 8086 用 M/IO# 区分访存与访端口，空间分开。 ② 内存 1MB，用 A19~A0 共 20 根地址线。 ③ I/O 64KB，用 A15~A0 共 16 根，DX 可放满 16 位。 ④ 独立编址不占内存空间，端口只能用 IN/OUT 访问。 **答案 A**

2020 · 选15 4 分

○ 会 ○ 卡 ○ 不会

CPU 与慢速设备进行数据传送时，采用（ ）方式可提高 CPU 的效率。

A、查询　B、中断　C、DMA　D、无条件传送

一句话：DMA 不占用 CPU，效率最高。 ① 查询方式 CPU 反复读状态，一直空转。 ② 中断方式每传一个字节都要保护现场。 ③ DMA 由 DMAC 接管总线，内存与设备间直接搬数据。 ④ 传送期间 CPU 可继续执行程序。 **答案 C**

2020 · 填7 4 分

○ 会 ○ 卡 ○ 不会

8086CPU 的 I/O 指令采用间接寻址时，使用的间接寄存器是_____。

一句话：端口间接寻址只能用 DX。 ① 直接寻址 IN AL,80H，端口号只能 8 位，最大 FFH。 ② 间接寻址 IN AL,DX，端口号先送 DX。 ③ DX 可寻址 0000H~FFFFH 全部 64K 端口。 ④ 这是指令规定，换别的寄存器汇编通不过。 **答案 DX**

2020 · 填8 4 分

○ 会 ○ 卡 ○ 不会

对于微机而言，任何新增的外部设备，最终总是要通过_____与主机相接。

一句话：外设必须经接口才能挂上总线。 ① 外设的速度、电平、数据格式与总线不匹配。 ② 接口要做缓冲、锁存、电平转换与地址译码。 ③ 所以任何外设最终都通过接口（适配器）与主机相连。 **答案 接口（接口电路 / 适配器）**

2020 · 填9 4 分

○ 会 ○ 卡 ○ 不会

8086CPU 用_____指令从端口读入数据，用_____指令向端口写入数据。

一句话：IN 读端口、OUT 写端口。 ① 输入用 IN，如 IN AL,80H 或 IN AL,DX。 ② 输出用 OUT，如 OUT 80H,AL 或 OUT DX,AL。 ③ 端口号大于 FFH 时必须先送 DX。 ④ IN/OUT 只访问 I/O 空间，不碰内存。 **答案 IN；OUT**

11 存储器扩展与地址译码

约 15 分 · 40 分钟

手上只有小芯片，怎么给 CPU 凑出一大块内存

一句话：小芯片拼大内存，只干两件事 —— **数据线怎么分**（解决位宽），**地址线怎么切**（解决片选）。

? 要给 8086 装 32KB 内存，手头只有 4 片 6264，每片 8KB。
问题：哪根线接哪根？CPU 读 70003H 时，怎么知道去第几片里找？

先讲清楚

一、先认清 6264 是什么

规格写着 **8K×8**：8192 个单元，每单元 8 位（1 字节）⇒ 共 **8KB**。

要访问 8192 个单元，需要几根地址线?

$2^{13}=8192$ ⇒ **13 根**（A12~A0）；再加 8 根数据线（D7~D0）。

引脚	作用
CE#	片选。给 0 才选中我
OE#	读的时候给 0
WE#	写的时候给 0

带 # 就是**低电平有效**——给 0 才干活。这条后面要反复用。

二、4 片怎么摆：横着接长，还是竖着接宽

横着接（地址接长）：4 片排一排，各管 8KB 地址段。数据线仍 8 根。

竖着接（数据接宽）：两片一组，一片给低 8 位、一片给高 8 位，合起来 16 位。

8086 数据线有 **16 根**，一次读 2 字节 ⇒ 必须竖着接，否则浪费一半。

于是 4 片分成 **2 组 × 2 片**：

组	接法	谁管它
偶体	接 D7~D0	A0 = 0 时工作
奇体	接 D15~D8	BHE# = 0 时工作

为什么这样分？因为 8086 规定：**偶地址走低 8 位，奇地址走高 8 位**。一个字（两个字节）正好一片给一半。

> 停一下：每组 2 片，每组多大？

两片各 8KB，共用一个地址 ⇒ 8192 个"字"，每字 16 位 = **16KB**。

两组合计 32KB ✓

三、把地址范围翻译成二进制（整道题的钥匙）

70000H~77FFFH，先算容量：**77FFFH − 70000H + 1 = 8000H = 32KB**。

再写成 20 位二进制，逐位比较：

位	首	末	结论
A19~A15	01110	01110	**固定** → 去译码
A14	0	1	**可变** → 分组

A13~A0	全 0	全 1	全变 → 接芯片

这张表就是判分点。固定的位判断"是不是我的地盘"，可变的位用来分片。

A14 只有 0/1 两种取值，正好分两组：

- A14=0 → 70000H~73FFFH（16KB）
- A14=1 → 74000H~77FFFH（16KB）

四、74LS138 是干什么的

CPU 只会输出电压，不会"判断地址是不是 01110"。**得有东西把地址翻译成"选第几片"**——这就是译码器。

138 是 3-8 译码器：给 3 根输入（C、B、A），8 个输出（Y0#~Y7#）里只选一个变低。

但 3 根不够判断 5 位，所以它还有 **3 个使能端**，全部满足才工作：

引脚	要什么电平
G1	1（高）
G2A#	0（低）
G2B#	0（低）

接线规则一句话：固定位接使能端，可变位接 CBA。

本题：G1=A17（要 1）、G2A#=A19（要 0）、G2B#=A15（要 0）；C=A18、B=A16、A=A14。

验一下：A14=0 时 CBA=110=6 ⇒ 选 **Y6#**；A14=1 时 CBA=111=7 ⇒ 选 **Y7#**。正好对上两组。

> 停一下：为什么 A14 接 CBA，不接使能端?

使能端是"一票否决"，不满足就 8 个输出全灭。

A14 要在两个输出间**二选一**，所以必须接 CBA。

口诀：**会变的接 CBA，固定的接使能。**

五、片内地址：为什么要"错一位"

138 用掉了 A19~A14，剩下 A13~A0。可 6264 只有 13 根地址线。

关键：8086 是 16 位机，**A0 不表示"第几个字节"，它表示"低字节还是高字节"**。

所以芯片的 A12~A0 要接 CPU 的 **A13~A1**，错开一位。

不算错的话，A0 会被当成地址低位，同一个 16 位字会被拆到两个不同单元，全乱。

> **小结：**8 位芯片接 16 位 CPU，地址线**必错一位**；接 8 位 CPU 就不用错。

六、最后一步：把片选分开

Y6# 是"每 16KB 一个"，但每 16KB 里有**两片**，得再分。

依据就是 A0 和 BHE#：

- Y6# **或** A0 → 偶片 CE#（只有 A0=0 时输出才为低）
- Y6# **或** BHE# → 奇片 CE#

为什么用"或"？因为 CE# 低有效：**两个输入都为低，输出才低。**

数据与控制线：

CPU	接到
D7~D0	偶片 D7~D0
D15~D8	奇片 D7~D0
RD# / WR#	OE# / WE#
M/IO#	=1（访存）
地址线	先过 74LS373 锁存

373 一定要画，因为 8086 的地址数据是复用的，不锁存地址就没了。

必背 · 六条

片数 = 总容量 ÷ 单片容量	不是地址线相除
容量 = 末 − 首 + 1	别忘 +1
芯片 A12~A0 接 A13~A1	8086 错一位
G1=1，G2A#=G2B#=0	三个都要满足
Y# 低有效	一次只选一个
Y# 与 A0/BHE# 相或	再分奇偶体

2024 · 设 1 15 分 完整走一遍

4 片 6264 接到 8086 的 70000H~77FFFH，138 译码。求片数、地址分配、译码连线。

第一步：算容量。77FFFH−70000H+1 = 8000H = 32KB。4×8KB = 32KB，够。

第二步：找固定位。A19~A15=01110 固定，A14 可变，A13~A0 全变。

第三步：分组。A14=0 管 70000H~73FFFH，A14=1 管 74000H~77FFFH。

第四步：接使能。G1=A17、G2A#=A19、G2B#=A15。

第五步：接选择。C=A18、B=A16、A=A14 ⇒ Y6#/Y7# 各管一组。

第六步：接片内地址。芯片 A12~A0 接 CPU 的 A13~A1。

第七步：分奇偶。Y6#∨A0→偶片 CE#，Y6#∨BHE#→奇片 CE#；Y7# 同理。

第八步：控制线。RD#→OE#，WR#→WE#，M/IO#=1，地址过 373。

第九步：验算。读 70555H：A19~A14=011100 ⇒ Y6# 有效；A0=1 ⇒ 奇片工作。落在范围内 ✓

答案：Y6# 管 70000H~73FFFH，Y7# 管 74000H~77FFFH；G1=A17、G2A#=A19、G2B#=A15、C=A18、B=A16、A=A14；地址接 A13~A1；Y# 与 A0/BHE# 相或分奇偶体。

⚠ **三个坑：**

① **片数拿地址线相除**（20÷13）。那是线数，不是容量。正确是 32KB÷8KB=4。

② **使能端接反。**A19 要 0 却接 G1（要 1），这片内存永远选不中，整题作废。记法：带 # 接 0。

③ **不分奇偶体。**一个片选给两片，两片同时输出，数据线打架。16 位机必扣分。

自测 · 3 道

1. B8000H~BFFFFH 是多少 KB?
2. 6264 为什么是 13 根地址线?
3. G1 要高电平还是低电平?

1. BFFFFH−B8000H+1 = 8000H = 32KB。

2. 8192 个单元，2^{13}=8192。

3. 高电平。只有 G2A#、G2B# 要低。

2024 卷 · 真题 2 道

2024 · 答3 8 分

○ 会 ○ 卡 ○ 不会

部分地址译码电路存在问题，请用实例说明。

一句话：部分译码会造出"一单元多地址"。 ① 全译码：所有高位地址都参与译码，地址唯一。 ② 部分译码：只接部分高位线，漏掉的线造成地址重叠。 ③ 实例：2KB 的 6116 只用 A15 反相接 CS#。 ④ A19~A16 悬空，00000H 与 10000H 都选中同一单元。 ⑤ 改法：A19~A16 也接入 138 使能端，做成全译码。 **关键：判据只有一条——高位地址是否全部参与译码。 答案：部分译码产生地址重叠（映像区），扩展时必冲突，必须全译码。**

2024 · 设1 10 分

○ 会 ○ 卡 ○ 不会

用 4 块 6264（8K×8）连接到 8086 的 70000H~77FFFH，用 74LS138 译码，画出与 8086 的连接。

一句话：32KB 分两组，每组奇偶各一片，138 出两个片选。 ① 容量：77FFFH−70000H+1=8000H=32KB，4 片正好。 ② A13~A1 并联接 4 片的 A12~A0，管片内 8K。 ③ 偶片接 D7~D0 由 A0 选通，奇片接 D15~D8 由 BHE# 选通。 ④ 138 使能：G1=A17，G2A#=A19，G2B#=A15。 ⑤ 选择端：C=A18、B=A16、A=A14，只用 Y6#、Y7#。 ⑥ Y6# 对应 70000H~73FFFH，Y7# 对应 74000H~77FFFH。 ⑦ 片选合成：Y6# 与 A0 或后送偶片 CS#，与 BHE# 或后送奇片 CS#。 ⑧ 控制：RD# 接 OE#，WR# 接 WE#，M/IO#=1，AD 经 373 锁存。 **关键：A19~A15 恒为 01110，只有 A14 在切两组。 答案：Y6# 管 70000H~73FFFH，Y7# 管 74000H~77FFFH，A0/BHE# 再分奇偶体，4 片占满 32KB。**

2023 卷 · 真题 1 道

2023 · 算2 10 分

○ 会 ○ 卡 ○ 不会

用两片 6116（地址 11 位）连接到 8086CPU，实现 16 位数据读写，画连接图并给出地址空间。

一句话：把两片 8 位 6116 并成 16 位，用 A0 与 BHE# 分奇偶体。 ① 6116 是 2K×8 的 SRAM，有 A0~A10 共 11 根地址线。 ② 两片并成 2K×16：偶地址走低 8 位，奇地址走高 8 位。 ③ 偶片片选由"译码输出+A0"驱动，A0=0 时选偶片。 ④ 奇片片选由"译码输出+BHE#"驱动，BHE#=0 时选奇片。 ⑤ 偶片接 D0~D7，奇片接 D8~D15，CPU 的 A1~A11 接两片 A0~A10。 ⑥ 译码用 74138：C、B、A 接高位地址，输出经或门作片选。 ⑦ 地址空间 40000H~40FFFH，容量 4KB，即 2K 个字。 ⑧ 只有 A0=0 且 BHE#=0，两片同时选中，才是一次 16 位读写。 **答案：奇偶分体并联成 2K×16，A0 选偶片、BHE# 选奇片，地址空间 40000H~40FFFH。**

2022 卷 · 真题 2 道

2022 · 答4 8 分

○ 会 ○ 卡 ○ 不会

74LS138 的输出 Y0、Y4、Y6、Y7 分别接不同存储器时，对应地址范围是多少？（图：CBA 接 A15–A13，Y 接各片 CE#）

一句话：把 64K 一页切成 8 块、每块 8K。 ① 设 CBA 接 A15–A13，高 4 位定在一页。 ② 每片 8KB，8 片拼成一个 64KB 页。 ③ Y0 最低块：00000H–01FFFH。 ④ Y4 第 5 块：08000H–09FFFH。 ⑤ Y6 第 7 块：0C000H–0DFFFH。 ⑥ Y7 最高块：0E000H–0FFFFH。 ⑦ 每块 2000H=8KB，8 块正好到 FFFFH。 **答案：00000H–01FFFH、08000H–09FFFH、0C000H–0DFFFH、0E000H–0FFFFH。**

2022 · 设1　10 分

○ 会　○ 卡　○ 不会

用 8088 CPU、4 片 6264 和 74LS138 构成存储系统，地址范围 B8000H–BFFFFH。(1) 几片 6264；(2) 地址分配表；(3) 画连接图。

一句话：把 32KB 窗口平分成 4 片 8KB 的 6264。 ① 范围：BFFFFH–B8000H+1=8000H，即 32KB。 ② 6264 每片 8KB，32K÷8K=4，正好 4 片。 ③ 8088 是 8 位数据线，每片直接挂 D0–D7，不分奇偶体。 ④ 片内 A12–A0 四片并联，统一接 8088 的 A12–A0。 ⑤ 片选靠 74138：C 接 GND，B 接 A14，A 接 A13。 ⑥ 使能：G1 接 A15，G2A# 接 A18，G2B# 接 A19·A17·A16。 ⑦ 4 个输出 Y0–Y3 分别接 4 片 CE#。 ⑧ 分配表：Y0→B8000H–B9FFFH，Y1→BA000H–BBFFFH。 ⑨ Y2→BC000H–BDFFFH，Y3→BE000H–BFFFFH。 ⑩ OE# 接 RD#，WE# 接 WR#，数据线接 D0–D7。 ⑪ A19–A0 先用 373 在 ALE 时锁存，再送存储器和译码器。 **答案：4 片；Y0–Y3 依次 B8000H–B9FFFH、BA000H–BBFFFH、BC000H–BDFFFH、BE000H–BFFFFH。**

2020 卷 · 真题 1 道

2020 · 选4　4 分

○ 会　○ 卡　○ 不会

现有 4K×8 位的 RAM 芯片，它具有的地址线条数应是（　）条。

A、10　B、11　C、12　D、13

一句话：用 4K=2^12 直接算出地址线。 ① 4K = 4096 = 2^12，共 4096 个单元。 ② 每单元 8 位正好一字节，不需再选字节。 ③ 地址线 = log2(4096) = 12 条。 ④ 8 位是数据线宽度，与地址线无关。 **答案 C**

12 数制、码制与复位

约 6 分 · 25 分钟

同一个字节，凭什么能读成 255、−1 和 99

> **一句话：** 0 和 1 本身没有含义：**补码**让减法变加法，**BCD** 给人看十进制，**复位**把 CPU 送到 FFFF0H。

? 同一个字节 1111 1111，可以有三种读法：255、−1、或者 99。
三个都对——**看你拿哪套"码"去读它**。
这一讲把这三套读法一次讲清。

先讲清楚

一、为什么是二进制，又为什么老写十六进制

硬件只认"有电/没电"，所以一切从二进制开始。

一根线两种状态：好做、抗干扰、便宜。要它认十种电平，成本翻几倍还容易读错。

可二进制太长：1024 要写 10 位，人眼一看就花。于是每 **4 位缩成 1 位十六进制**。

十进制	0	1	7	10	15
二进制	0000	0001	0111	1010	1111
十六进制	0	1	7	A	F

所以 1011 0100B 一眼拆成 1011 和 0100 ⇒ **B4H**；反过来，一位拆四位。

小结：十六进制不是另一种数，只是二进制的缩写。

二、进制转换只有三个动作

二转十按权展开，十转二除 2 取余，二与十六四位一组。

权就是 2 的幂，从右往左数：第 0 位是 $2^0=1$，第 3 位是 $2^3=8$。

第几位	7	6	5	4	3	2	1	0
权值	128	64	32	16	8	4	2	1

例：1011.11B = 8+2+1 + 0.5+0.25 = **11.75**。

例：1234H = $1\times16^3+2\times16^2+3\times16+4$ = 4096+512+48+4 = **4660**。

十进制转二进制：整数一直除 2，余数**倒着读**；小数一直乘 2，整数**顺着读**。

例：100 一路除 2，余数倒读得 1100100B，补成 8 位 = **0110 0100B = 64H**。

停一下：把 1101 0110B 写成十六进制，再按权展开成十进制。

四位一组：1101 = D，0110 = 6 ⇒ **D6H**。

按权展开：128+64+16+4+2 = **214**。

三、原码、反码、补码：为什么要多此一举

因为补码能把减法变成加法，CPU 只要一个加法器。

没有补码，算 5−3 要专门的减法电路；有了补码，算 5+(−3) 就行。

道理像钟表：3 点，往前走 10 小时和往后退 2 小时，指针停在同一格。

表盘只有 12 格，12 就是它的"模"，减 2 等于加 10。

8 位有 256 个格，模就是 256，所以 **−n 的补码 = 256 − n**。

码	+5	−5	怎么来的
原码	0000 0101	1000 0101	最高位当符号位，1 表示负
反码	0000 0101	1111 1010	负数：符号位不动，其余取反
补码	0000 0101	**1111 1011**	反码加 1

速算法：从右往左找**第一个 1**，它和它右边的位不动，左边全取反。

验 −5：0000 0101 的第一个 1 在 bit0，左边取反 ⇒ 1111 1011 ✓

范围记死：8 位 **−128~+127**，16 位 **−32768~+32767**。

两个特殊值：80H 是 −128，FFH 是 −1；00H 的补码就是它自己。

> 停一下：8 位补码 80H 换算成十进制是多少?

−128，8 位补码能表示的最小负数。

它没有对应的正数，所以取补码还是 80H。

四、8 位扩成 16 位：符号扩展

高位全复制符号位：正数补 0，负数补 1。

为什么不能一律补 0？负数高 8 位补 0，值就从 −90 变成 +166。

例：A6H = 1010 0110B，符号位是 1 ⇒ **1111 1111 1010 0110B = FFA6H**。

验算：A6H 按无符号是 166，166−256 = **−90**；FFA6H 是 65446，65446−65536 = **−90** ✓

指令 **CBW** 把 AL 符号扩展到 AX；**CWD** 把 AX 扩展到 DX:AX。

五、BCD：给人看的十进制

BCD 是"4 位二进制硬表示 1 位十进制"，只用到 0000~1001。

为什么要它？二进制转十进制要除法，而电子秤、计算器只想直接把数字显示出来。

例：59 的二进制是 0011 1011B = 3BH，BCD 却是 **0101 1001B = 59H**。

看着像"把十进制抄进十六进制"，这正是 BCD 的特点，也是最容易混的地方。

类型	一字节放几位	59 的写法
压缩 BCD	两位	59H
非压缩 BCD	一位，高 4 位补 0	05H 和 09H

BCD 直接相加会错：7+8 的二进制和是 0FH，BCD 要的是 15H。

所以算完要调整：**DAA** 加 6 修正，**DAS** 减 6 修正。

停一下：BCD 码 1001 0011B 表示的十进制是多少?

93：高 4 位 1001 是 9，低 4 位 0011 是 3，一位一位读。

按二进制读成 147 就错了，那已经不是在读 BCD。

六、复位：CPU 醒来先看哪里

复位后 CS=FFFFH、IP=0000H，第一条指令在 FFFF0H。

为什么定在最高端？那里通常放 ROM，断电不丢，正好装开机程序。

物理地址 = 段×16 + 偏移 = FFFFH×16 + 0000H = **FFFF0H**。

复位后的状态	值
CS	**FFFFH**
IP	**0000H**
DS、ES、SS	0000H
标志寄存器	全清，IF=0（关中断）
指令队列	清空

FFFF0H 到 FFFFFH 只有 **16 个字节**，装不下初始化程序。

所以那里只放一条 **JMP**，跳到 ROM 里真正的入口。

小结：复位不给地址，只给一个固定动作：从 FFFF0H 取第一条指令。

必背 · 六条

4 位二进制 = 1 位十六进制	四位一组
补码 = 反码 + 1	正数三码相同
8 位补码 −128~+127	80H 是 −128
符号扩展：高位全复制符号位	正补 0 负补 1
BCD：4 位二进制表示 1 位十进制	59H 表示 59
复位 CS=FFFFH、IP=0000H	从 FFFF0H 执行

数制与码制 6 分 补码 + BCD

把 −37 写成 8 位补码，再符号扩展成 16 位；另外写出十进制 47 的压缩 BCD。

第一步：先写绝对值。37 = 32+4+1 ⇒ 0010 0101B。

第二步：加符号位得原码。最高位写 1 ⇒ 1010 0101B。

第三步：符号位不动，其余取反。反码 = 1101 1010B。

第四步：反码加 1。补码 = **1101 1011B = DBH**。

第五步：速算法复查。0010 0101 从右往左第一个 1 在 bit0，左边全反 ⇒ 1101 1011 ✓

第六步：符号扩展。符号位是 1，高 8 位补 1 ⇒ **1111 1111 1101 1011B = FFDBH**。

第七步：验算。DBH 是 219，219−256 = −37；FFDBH 是 65499，65499−65536 = −37 ✓

第八步：写 BCD。十位的 4 进高 4 位，个位的 7 进低 4 位 ⇒ **0100 0111B = 47H**。

答案：−37 的 8 位补码是 DBH，16 位是 FFDBH；47 的压缩 BCD 是 47H。

⚠ **三个坑：**

① **把 BCD 当二进制。**59 的 BCD 是 59H，不是 3BH；题目要 BCD 你写 3BH，显示出来就是 3B。看到"BCD"，就一位一位按十进制读。

② **符号扩展补错方向。**负数的符号位是 1，高 8 位必须补 1；补 0 会把 −90 变成 +166，数值直接错。

③ **复位入口写成 00000H。**CS 是 FFFFH，不是 0000H；段基址折算后第一条指令在 FFFF0H，写成 00000H 或 FFFFH 都错。

自测 · 3 道

1. 47H 按 BCD 读是多少？按无符号二进制读又是多少？
2. 8 位补码 A6H 扩成 16 位是多少？
3. 8086 复位后执行的第一条指令在哪个物理地址？

1. BCD 读 47；无符号读 71（4×16+7）。

2. FFA6H：符号位是 1，高 8 位补 FF。

3. FFFF0H。

2020 卷 · 真题 3 道

2020 · 选1 4 分

○ 会 ○ 卡 ○ 不会

8086/8088 复位后从（ ）地址处开始执行程序。
A、00000H B、FFFF0H C、FFFFFH D、FFF0H

一句话：记住复位后 CS=FFFFH、IP=0000H。 ① 复位时 CS 置 FFFFH，IP 清 0000H。 ② 物理地址 = FFFFH×16 + 0000H = FFFF0H，在最高端。 ③ 该处一般放一条 JMP，跳到初始化程序。 ④ 不是 FFFFH，也不是 00000H。 **答案 B**

2020 · 选9 4 分

○ 会 ○ 卡 ○ 不会

X 的 8 位补码是 10110100，则 X 的 16 位补码是（ ）。
A、0000000010110100 B、1000000010110100
C、1111111110110100 D、0111111110110100

一句话：负数符号扩展，高位全补 1。 ① 10110100 最高位是 1，说明 X 是负数。 ② 8 位扩 16 位要做符号扩展，高 8 位补符号位。 ③ 符号位为 1 就补 8 个 1，低 8 位不变。 ④ 结果 11111111 10110100；只有正数才补 0。 **答案 C**

2020 · 填3 4 分

○ 会 ○ 卡 ○ 不会

二进制数 1011.11B 转化为十进制数为_____。

一句话：按权展开，整数和小数分别算。 ① 整数部分 1011B = 8+0+2+1 = 11。 ② 小数部分 .11B = 1/2 + 1/4 = 0.75。 ③ 相加得 11.75。 ④ 验算：11.75×4 = 47 = 101111B。 **答案 11.75**

13

8253 定时/计数器

约 12 分 · 40 分钟

CPU 不会数时间，8253 替它数

> **一句话：** 把"要等多久"换算成"数几个时钟"，数满一个周期，OUT 就动一下。

? 时钟 1MHz，每 1μs 来一个脉冲，要产生 1 秒方波就得数 1000000 下。
可一个通道最多只数 65536 下，差着一位数量级，怎么办?

先讲清楚

一、为什么需要 8253

软件延时既不准又占着 CPU，交给硬件计数器才靠得住。

想延时 1 秒只能让 CPU 空跑，次数还会被主频、中断改来改去。

8253 的做法是：外面给一个节奏稳定的时钟，它只负责数数。

数到约定的个数，OUT 脚就给一个信号，CPU 不用守着。

内部是 **3 个独立的 16 位减法计数器**，各带 CLK、GATE（允许计数）、OUT 三根脚。

A1A0	端口	典型用途
00	通道 0	分频、定时（最常用）
01	通道 1	DRAM 刷新
10	通道 2	接扬声器发声
11	**控制口**	只写，用来下命令

基址 300H ⇒ 三个通道是 300H~302H，**控制口是 303H**。

> **小结：**8253 = 3 个通道 + 1 个控制口，占 4 个地址。

二、方式控制字：8 位各管一件事

控制字就是给通道立规矩：选哪个通道、初值怎么装、按什么规律输出。

位	管什么	取值
D7 D6	选通道	00=通道0，01=通道1，10=通道2
D5 D4	初值怎么读写	01=只低 8 位，10=只高 8 位，**11=先低后高**
D3 D2 D1	工作方式	000~101 依次对应方式 0~5
D0	计数制	0=二进制，1=BCD

例：通道 0、先低后高、方式 3、二进制 ⇒ 00 11 011 0 = **36H**。

最高两位改成 01，其余照抄，就是通道 1 的 **76H**。

D5D4 填 11 最省心：16 位初值一律写两次，先低后高。

> **停一下：**通道 2、只写低字节、方式 2、二进制，控制字是多少?

D7D6=10（通道 2），D5D4=01（只低字节），D3D2D1=010（方式 2），D0=0。

拼起来 10 01 010 0 = **94H**。

三、六种工作方式：先吃透 2 和 3

方式 2 出周期性窄脉冲，方式 3 出周期性方波，这两个占了绝大多数题。

方式	名字	OUT 长什么样
0	计数结束中断	计到 0 变高，只此一次
1	可编程单稳	GATE 上升沿后出负脉冲，宽 N 个时钟
2	**分频器**	连续输出，每周期低 1 个时钟、高 N−1 个
3	**方波发生器**	连续输出，高低各占一半
4	软件触发选通	写完初值出一个负脉冲

5	硬件触发选通	GATE 上升沿后出一个负脉冲

方式 2 的关键是"低 1 个时钟"：周期仍是 N 个时钟，只是低电平很窄。

> **小结：**题目说"方波"就写方式 3，说"分频""连续脉冲"就写方式 2。

四、计数初值：把时间换成个数

初值 = 要等的时长 ÷ 一个时钟的时长，也就是输入频率 ÷ 输出频率。

$N=\frac{f_{in}}{f_{out}}$　一秒里来多少个时钟，就除出这一秒该数多少下。

例：1MHz ⇒ 一个时钟 1μs；要 1kHz ⇒ 输出周期 1000μs，$N=\frac{1000}{1}=1000$。

1000 写成十六进制是 **03E8H**。

读数制	最大值	超出会怎样
二进制（D0=0）	**65536**	装不下，必须级联
BCD（D0=1）	**10000**	装不下，必须级联

初值写 **0000H** 表示 **65536**，这是硬件约定，不是 0。

五、初始化顺序：先立规矩，再装数

顺序固定：先写方式字，再写初值；16 位初值分两次，先低字节后高字节。

1. 算出初值 N，检查没超过 65536。
2. 拼好方式字，写到控制口。
3. 把 N 拆成低字节、高字节。
4. 先写低字节，再写高字节。

为什么必须先写方式字？它靠这个字节认定"下面是初值，而且先低后高"。

顺序反了，第一个初值字节会被当成方式字，配置全乱。

程序（基址 300H，通道 0，N=1000）：

MOV DX,303H → MOV AL,36H → OUT DX,AL，写方式字。

MOV DX,300H → MOV AX,1000 → OUT DX,AL，送低字节 E8H。

MOV AL,AH → OUT DX,AL，送高字节 03H。

停一下：1000 明明是一个 16 位数，为什么 OUT 两次就送完了？

1000 = 03E8H，低字节 E8H 在 AL，高字节 03H 在 AH。

第一次送 AL，再把 AH 挪进 AL 送第二次。拆字节由汇编器按小端摆好。

六、超过 65536：两级级联

一个通道装不下，就把前一级 OUT 接到后一级 CLK，分频倍数相乘。

1MHz 出 1Hz，总倍数 1000000。拆成 1000 × 1000，每级都落在范围内。

级	输入	初值	输出
通道 0	1MHz	1000	1kHz（周期 1ms）
通道 1	1kHz	1000	1Hz（周期 1s）

接线：**OUT0 接 CLK1，OUT1 就是最后输出。**

两个通道各写自己的方式字：通道 0 是 36H，通道 1 是 76H。

级联的初值是**相乘**不是相加：1000 × 1000 = 1000000。

> **小结：**先算总倍数，再拆成两个 ≤65536 的因数，一级接一级往下传。

必背 · 六条

N = 输入频率 ÷ 输出频率	也可以用时长相除
方式 2 分频，方式 3 方波	周期性的只有这两个
36H / 76H	通道 0 / 通道 1，方式 3，先低后高
先控制字，后初值	顺序反了配置全乱
16 位初值写两次	先低字节，再高字节
OUT0 接 CLK1，倍数相乘	这才是级联

8253 定时 12 分 从需求到程序段

8253 基址 300H，CLK0 接 2MHz 时钟。要求 OUT0 输出周期 5ms 的方波。求工作方式、计数初值、方式字，并写初始化程序段。

第一步：定方式。要"方波"，高低各半 ⇒ **方式 3**。

第二步：算时钟周期。2MHz ⇒ $T_{clk}=\frac{1}{2\times 10^6}=0.5\mu s$。

第三步：把时间换成个数。5ms = 5000μs，$N=\frac{5000}{0.5}=10000$。

第四步：查范围。10000 < 65536，一个通道够，不用级联。

第五步：转十六进制。10000 = 2710H（2×4096 + 7×256 + 1×16 = 10000）。

第六步：拼方式字。通道 0、先低后高、方式 3、二进制 ⇒ 00 11 011 0 = **36H**。

第七步：写方式字。MOV DX,303H；MOV AL,36H；OUT DX,AL。

第八步：写初值。MOV DX,300H；MOV AX,2710H；OUT DX,AL 送低字节 10H；MOV AL,AH；OUT DX,AL 送高字节 27H。

第九步：验算。10000 个 0.5μs = 5ms，高 2.5ms、低 2.5ms，确实是方波 ✓

答案：方式 3；初值 10000（2710H）；方式字 36H。

⚠ **三个坑：**
① **16 位初值只写一次。**只送低字节，高字节留着上一次的垃圾，实际初值可能差 256 倍。
② **要方波写成方式 2。**方式 2 的低电平只有 1 个时钟，出来是窄脉冲，不是方波。
③ **超过 65536 还硬塞。**初值被截断，输出频率完全不对，必须拆成两级相乘。

自测 · 3 道

1. 通道 2、方式 2、只写低字节、二进制，方式字是多少?
2. 2MHz 时钟送进 CLK0，初值装 4000，OUT0 的频率是多少?
3. 1MHz 要出周期 0.5s 的方波，为什么必须级联？怎么拆?

1. 10 01 010 0 = **94H**。

2. 2000000 ÷ 4000 = **500Hz**。

3. 需要 500000 分频，超过 65536。拆成 500 × 1000，OUT0 接 CLK1。

2020 卷 · 真题 2 道

2020 · 填10 4 分

○ 会 ○ 卡 ○ 不会

利用 8253 通道 0 产生周期 5ms 的方波，输入时钟 1MHz，则工作方式为_____，计数初值为_____，工作方式字是_____。

一句话：方波选方式 3，5ms÷1μs 得初值 5000。 ① 输入 1MHz，即每 1μs 一个时钟，5ms 需 5000 个脉冲。 ② 初值 N = 5ms ÷ 1μs = 5000 = 1388H。 ③ 方波对应方式 3，高低各占一半。 ④ 控制字：通道0、先低后高、方式3、二进制 = 00110110B = 36H。 ⑤ 5000 未超 65536，不必级联。 **答案 方式 3；初值 5000（1388H）；方式字 36H**

2020 · 分析1

○ 会　○ 卡　○ 不会

8086 最大方式系统中一片 8253，端口地址 301H、303H、305H、307H，产生周期 1 秒信号。完善连接图（含译码）并写程序段。

一句话：8253 接高 8 位，两级级联凑出 1 秒。 ① 301H/303H/305H/307H 全是奇地址，A0 恒为 1。 ② 说明 8253 数据线接高 8 位 D8~D15。 ③ 地址间隔 2：8086 的 A1 接 8253 的 A0、A2 接 A1。 ④ 片选由 A15~A3 译码，并与 BHE# 相与，保证只在高字节周期选中。 ⑤ 1MHz 下 1 秒要 10^6 个脉冲，远超 65536。 ⑥ 级联：通道 0 先 1000 分频得 1ms，通道 1 再 1000 分频。 ⑦ 控制字：通道 0 用 36H，通道 1 用 76H（方式 3）。 ⑧ 代码：MOV DX,307H；MOV AL,36H；OUT DX,AL；MOV DX,301H；MOV AX,1000；OUT DX,AL；MOV AL,AH；OUT DX,AL ⑨ 通道 1 换成 76H、口地址 303H，初值同样 1000。 ⑩ 硬件：1MHz 接 CLK0，OUT0 接 CLK1，OUT1 输出 1Hz。 **答案 方式 3 两级级联，控制字 36H 与 76H，OUT1 输出 1 秒方波**

14 8255A 并行接口

约 10 分 · 35 分钟

24 根线，方向全由一个字决定

> **一句话：**控制字最高位是分水岭：写 **1** 定方向，写 **0** 改 C 口某一根线。

? CPU 输出一个字节就转身走了，可 LED 要一直亮着。
这个字节寄存在哪？谁保证它不消失？

先讲清楚

一、为什么需要并行接口

CPU 只会"读一个字节、写一个字节"，锁存、缓冲、握手这些活得有人替它干。

CPU 往数据总线写一个字节，几个时钟周期后就撤走了。LED 只会闪一下。

所以要有一个输出锁存器，把电平一直保持住，直到下次改写。

输入反过来：按键只按下几毫秒，CPU 读的那一刻不一定正好按着，需要缓冲。

8255A 就干这两件事：**24 根 I/O 线，分 A、B、C 三组，方向可以由程序改。**

A1A0	端口	说明

00	A 口（PA7~PA0）	8 位，可入可出
01	B 口（PB7~PB0）	8 位，可入可出
10	C 口（PC7~PC0）	8 位，常拆成两半用
11	**控制口**	只写，用来下命令

基址 260H ⇒ 260H 是 A 口，261H 是 B 口，262H 是 C 口，**263H 是控制口**。

复位后 24 根线**全是方式 0 输入**，不写控制字，输出脚推不动外设。

> **小结：** 4 个地址、3 组端口，控制口只进不出。

二、A 组和 B 组：C 口被劈成两半

三组端口不是平级的：A 口配 PC7~PC4 叫 A 组，B 口配 PC3~PC0 叫 B 组。

因为控制字就是按这两组定方式、定方向的。

组	包含	谁能定它的方式
A 组	A 口 + PC7~PC4	控制字 D6 D5
B 组	B 口 + PC3~PC0	控制字 D2

C 口不是一个整体：高 4 位和低 4 位可以一个输入、一个输出。

所以控制字里方向位有 D4、D3、D1、D0 四个，分别管 A 口、C 高半、B 口、C 低半。

记忆：**A 组管"高"，B 组管"低"**。

三、三种工作方式

方式 0 是基本 I/O，方式 1 加握手，方式 2 是双向，而且只有 A 口有。

方式	名字	特点
0	基本 I/O	三组口都是普通输入输出，不要联络信号
1	选通 I/O	带 STB#、IBF、INTR 等握手线，C 口让出 3 根
2	双向总线	**只有 A 口能设**，PC7~PC3 五根线全给它

方式 0 最常用：接 LED、开关、数码管，都是它。

方式 1 是"我准备好了"和"我收到了"的对话，靠 C 口那几根线传话。

B 口最高只能到方式 1，**做不了方式 2**。

> **停一下：**B 组的方式位只有 D2 一位，那 B 口能有几种方式?

一位只能表示两种：D2=0 是方式 0，D2=1 是方式 1。

A 组用 D6D5 两位，才有 00/01/1x 三种，其中 1x 是方式 2。

四、方式控制字：逐位拼

D7=1 才是方式字，剩下 7 位从高到低，一位管一件事。

位	管什么	取值
D7	字类型	**1**=方式控制字
D6 D5	A 组方式	00=方式0，01=方式1，1x=方式2
D4	A 口方向	**1=输入，0=输出**
D3	PC7~PC4 方向	1=输入，0=输出
D2	B 组方式	0=方式0，1=方式1
D1	B 口方向	1=输入，0=输出
D0	PC3~PC0 方向	1=输入，0=输出

这里和直觉相反：1 表示输入，0 表示输出。

例：A 入、B 入、C 全出、都是方式 0 ⇒ 1 00 1 0 0 1 0 = **92H**。

例：A 出、B 出、C 全出 ⇒ 1000 0000B = **80H**。

写入就一句：MOV DX,263H；MOV AL,92H；OUT DX,AL。

五、C 口位操作字：只动一根线

D7=0 时，这条命令只改 C 口的一根线，别的 23 根纹丝不动。

位	管什么	取值
D7	字类型	**0**=位操作字

D6 D5 D4	不用	随便写，习惯写 0
D3 D2 D1	选哪一根	000=PC0 … 111=PC7
D0	要什么电平	1=置高，0=拉低

例：PC3 置高 ⇒ 0000 0111B = **07H**；PC3 拉低 ⇒ **06H**。

例：PC7 拉低 ⇒ 0000 1110B = **0EH**。

为什么不重写方式字去改电平？因为方式字里**没有电平这一项**。

而且重写方式字会把整组方向重新定一遍，别的输出脚可能被顺手清掉。

停一下：让 PC0 输出高电平，往控制口写什么？

D7=0，D3D2D1=000（选 PC0），D0=1（置高）。

0000 0001B = **01H**。

六、初始化程序与实际用法

先写方式字定方向，再读写数据口；中途只翻某根线就用位操作字。

典型题：分时采集 8 组 16 位数据，找出最大值，用 LED 显示组号。

引脚	干什么
PC2~PC0	输出组号 S2~S0，选第几组
A 口	读低字节
B 口	读高字节
PC7~PC5	接 LED，显示最大值的组号

A、B 读入，C 全输出，方式 0 ⇒ 控制字 **92H**。

程序段：MOV DX,263H；MOV AL,92H；OUT DX,AL（初始化）。

选组：MOV DX,262H；MOV AL,BL；OUT DX,AL。

读高字节：MOV DX,261H；IN AL,DX；MOV AH,AL。

读低字节：MOV DX,260H；IN AL,DX，此时 AX 就是 16 位数据。

> **小结：**方向靠方式字，组号靠 C 口低 3 位，比较靠 A、B 拼字。

必背 · 六条

D7=1 方式字，D7=0 位操作字	先看最高位分家
1=输入，0=输出	和直觉相反
A 组 = A 口 + PC7~PC4	B 组 = B 口 + PC3~PC0
方式 2 只有 A 口	B 口最多到方式 1
92H：A 入 B 入 C 全出	采集题标配
复位后 24 根线全输入	不写方式字没法输出

8255A 控制字 10 分 拼字 + 改电平

8255A 基址 280H。A 口方式 0 输出，B 口方式 1 输入，PC7~PC4 输出，PC3~PC0 输入。求方式控制字和初始化程序；再把 PC6 置高、再拉低。

第一步：判类型。题目要"定方向"，不是改某一位 ⇒ D7=1。

第二步：A 组方式。方式 0 ⇒ D6D5=00。

第三步：A 口方向。输出 ⇒ D4=0。

第四步：C 口高半。输出 ⇒ D3=0。

第五步：B 组方式。方式 1 ⇒ D2=1。

第六步：B 口方向。输入 ⇒ D1=1；C 口低半输入 ⇒ D0=1。

第七步：拼。1 00 0 0 1 1 1 = **87H**。

第八步：写初始化。MOV DX,283H；MOV AL,87H；OUT DX,AL。

第九步：改 PC6。D3D2D1=110，置高 D0=1 ⇒ 0000 1101B = 0DH；拉低 ⇒ 0CH。

第十步：验算。87H 展开：D7=1、D6D5=00、D4=0、D3=0、D2=1、D1=1、D0=1，与要求逐位对上 ✓

答案：控制字 87H；置高写 0DH，拉低写 0CH。

⚠ **三个坑：**
① **把 1 当输出。**想让全部输出就按 1 填，写成 9BH 甚至 FFH，结果 24 根全是输入。
② **让 B 口做方式 2。**D2 只有一位，写 1 得到的是方式 1，不是双向；双向只有 A 口。
③ **只改一根线却重写方式字。**整组方向被重置，本来输出高的脚会被一起改掉；该用位操作字。

自测 · 3 道

1. A 口输入、B 口输出、C 口全输入、都用方式 0，控制字是多少?
2. 要让 PC0 输出高电平，控制口写什么?
3. A 组包含哪些引脚? B 组呢?

1. 1001 1011B = **9BH**（D4=1、D3=1、D1=0、D0=1）。

2. 0000 0001B = **01H**。

3. A 组 = A 口 + PC7~PC4；B 组 = B 口 + PC3~PC0。

2020 卷 · 真题 1 道

2020 · 分析2

○ 会　○ 卡　○ 不会

一片 8255A（端口 260H~263H）分时采集 8 组 16 位无符号数据，找出最大值并用 C 口高三位（PC7~PC5）的 LED 指示序号。完善硬件并写程序段。

一句话：A/B 口读数据、C 口选组点灯。 ① 260H=A 口、261H=B 口、262H=C 口、263H=控制口。 ② A 口读低字节，B 口读高字节，拼成一个字。 ③ PC2~PC0 输出选择码 S2~S0，PC7~PC5 接 LED。 ④ 控制字：方式 0、A/B 入、C 全出 = 10010010B = 92H（D3=0，PC7~PC5 要输出）。 ⑤ 16 位无符号比较要用 CMP 配 JBE，不能写 JG/JL。 ⑥ BH 记最大值的组号，SI 记最大值，BL 记当前组号。 ⑦ 代码：MOV DX,263H；MOV AL,92H；OUT DX,AL；MOV DX,262H；MOV AL,BL；OUT DX,AL；MOV DX,261H；IN AL,DX；MOV AH,AL；MOV DX,260H；IN AL,DX；CMP AX,SI；JBE SKIP；MOV SI,AX；MOV BH,BL ⑧ 采完 8 组把 BH 左移 5 位输出到 C 口点灯。 ⑨ 硬件：AEN 接 74138 使能，IOR#/IOW# 接 RD#/WR#。 ⑩ A1、A0 接 8255 的 A1、A0，高位译码出 CS#。 **答案 控制字 92H；A/B 输入拼字，C 口低 3 位选组、高 3 位点灯，比较用 JBE**

15 汇编程序阅读与数据段

约 15 分 · 40 分钟

不用上机，拿张纸把寄存器的值一步步抄下来

一句话：先算数据占几字节，再标跳转往哪走，最后拿具体数走一遍。

? 给你 8 行汇编，问"AL 最后是多少"，没有机器、不能运行。
你怎么算得出来?

先讲清楚

一、为什么程序能"手算"

CPU 每一步只做一件事，寄存器就是你手边的草稿纸。

一条指令的活儿很小：取个数、算一下、存回去，最多再改几个标志位。

所以读程序不是"读懂英文"，是**照着顺序抄状态**。

准备一张表：AX、BX、CX、SI、ZF、CF。每执行一条，改一格。

题目问"结果是多少"，答案就在这张表的最后一行。

读之前先分类：这段是顺序、分支，还是循环？分类定了，走法就定了。

二、数据段：先算清楚每项占几字节

DB 占 1 字节，DW 占 2 字节，DD 占 4 字节，DUP 是重复，EQU 不占空间。

伪指令	占几个字节	例子
DB	1	DB 12H
DW	2	DW 3568H
DD	4	DD 0
n DUP(x)	n × 单元大小	4 DUP(20H) 占 4 字节
EQU	**0**	CNT1 EQU 3568H

为什么要先算字节？因为后面所有偏移都是从这推出来的，错一格，后面全错位。

程序里的标号，值就是它的**偏移地址**；OFFSET 标号 就是把这个偏移取出来。

物理地址 = 段基址 × 16 + 偏移。段基址 8850H、偏移 5500H ⇒ 8DA00H。

字和双字一律**低字节存低地址**。DW 3568H 落到 5504H，存成 68H、35H。

> 小结：EQU 只是给常数起名，ORG 只是把偏移拨到某处，两者都不产生字节。

三、画存储图：三步走

按顺序排偏移，按类型算字节，把字拆成低字节在前。

题（2020 卷）：数据段段地址 8850H，画出 5500H~550FH 的存储情况。

CNT1 EQU 3568H；CNT2 EQU 110H；ORG 5500H。

偏移	字节	来自哪一句
5500~5503H	20 20 20 20	VCHAR DB 4 DUP(20H)
5504~5505H	68 35	DW CNT1 = 3568H
5506~5507H	10 01	DW CNT2 = 0110H
5508~5509H	04 55	DW VTIMCNT（偏移 5504H）
550A~550DH	08 55 50 88	DD PVT1（偏移 5508H、段 8850H）
550E~550FH	00 00	DW 0

两个容易卡的点：DW 变量名，存的是那个变量的**偏移**，不是它的内容。

DD 变量名，存的是 32 位指针：**先低字放偏移，再高字放段**。

> 停一下：ORG 5500H 后面写 DB 3 DUP(0FFH)，下一项的偏移是多少？

3 个字节占 5500H、5501H、5502H。

下一项从 **5503H** 开始。

四、第一遍读程序：分块，标跳转

先把程序切成顺序段、分支段、循环段，在每个跳转后面写上"跳去哪、什么条件"。

结构	长什么样	怎么找
顺序	一条接一条往下走	没有 JMP、没有 Jcc
分支	两条路，只走一条	一个 CMP 配一个 Jcc
循环	往回跳，做同样的事	跳转目标在**前面**，还有 CX

动手办法：给每个标号编行号，把 Jcc 后面的目标改成行号，走向就一目了然。

看到 **LOOP** 就知道是循环，看到 **JMP** 就知道这段到此为止。

别一上来猜功能。**先把走向画出来，功能是走完几遍之后自然浮出来的。**

五、条件跳转：看标志位，不看数字大小

CMP 只做减法、只改标志、不存结果；跳不跳看的是标志位。

指令	什么时候跳	用在
JE / JNE	相等 / 不等（ZF）	判等
JC / JNC	有借位 / 无借位（CF）	无符号比大小
JA / JAE	大于 / 大于等于	**无符号**
JB / JBE	小于 / 小于等于	**无符号**
JG / JGE / JL / JLE	大于、小于…	**有符号**

例：CMP AL,30H 后 JC ⇒ **AL 小于 30H**；CMP AL,3AH 后 JNC ⇒ **AL 大于等于 3AH**。

为什么要分有符号无符号？因为 F5H 当无符号是 245，当有符号是 −11。

用错指令，比较结果正好反过来，最大值会被找成最小值。

停一下：CMP AL,80H 之后 JB 成立，说明 AL 大概在什么范围?

JB 是"无符号小于"，所以 AL 落在 00H~7FH。

如果用 JL（有符号），80H 会被当成 −128，结论完全不同。

六、循环与堆栈

循环三件套：CX 装次数、指针自己 INC、LOOP 回头。

LOOP 只做两件事：CX 减 1；CX 不为 0 就跳回去。

LOOP 不管地址，所以 SI、DI 必须自己在循环体里加。

堆栈换一套规矩：**PUSH 先 SP 减 2 再存，POP 先取再 SP 加 2。**

栈往低地址长，所以**栈底是最高地址**，栈顶由 SP 指着。

例：SS=5600H、SP=0070H、栈区 0000H~0300H ⇒ 栈顶 56070H，栈底 56300H。

PUSH AX（AX=2010H）⇒ SP=006EH，10H 存 5606EH，20H 存 5606FH。

> 小结：循环看 CX，堆栈看 SP；一个往上加地址，一个往下减地址。

必背 · 六条

DB / DW / DD = 1 / 2 / 4 字节	画存储图的尺子
低字节存低地址	字和双字都这样
EQU 不占空间，ORG 定起点	偏移从 ORG 排起
物理地址 = 段 × 16 + 偏移	别忘乘 16
无符号 JA/JB，有符号 JG/JL	CMP 之后选对
PUSH 先 SP−2，POP 后 SP+2	栈向低地址长

读程序 12 分　找出 5 个字节里的最大值

数据段：BUF DB 12H,0F5H,07H,9CH,33H；CNT EQU 5；MAX DB ?

程序段（DS 已指向数据段）：

```
MOV SI,OFFSET BUF
MOV CX,CNT
MOV AL,[SI]
INC SI
DEC CX
L1:   CMP [SI],AL
JBE L2
MOV AL,[SI]
L2:   INC SI
LOOP L1
MOV MAX,AL
```

问：循环跑几轮？MAX 里最后是多少？

第一步：看数据。BUF 有 5 个字节，CNT=5，MAX 占 1 字节。

第二步：找结构。L2 之后 LOOP L1，目标在前面 ⇒ 这是**循环**。

第三步：初始化。SI 指向 12H，CX=5，AL 取第一个数 12H。

第四步：先把第一个数处理掉。INC SI 后 SI 指 F5H；DEC CX 后 CX=4，循环只剩 4 轮。

第五步：第一轮。CMP F5H,12H ⇒ F5H 更大，JBE 不跳，AL 改成 F5H。

第六步：第二轮。07H 比 F5H 小，JBE 成立，跳过，AL 还是 F5H。

第七步：第三轮。9CH 也比 F5H 小，跳过；CX 减到 1。

第八步：第四轮。33H 更小，跳过；CX 减到 0，LOOP 不回头。

第九步：收尾。MOV MAX,AL ⇒ MAX = **0F5H**。

验算：5 个数里最大是 F5H=245。9CH 当无符号是 156，比 245 小，所以没被误判成 −100 ✓

答案：MAX = 0F5H，功能是求 5 个无符号字节的最大值。

⚠ **三个坑：**

① **把 EQU 当变量。**以为 CNT1 占 2 字节，后面偏移全体后移，整张存储图作废。

② **画存储图高字节放低地址。**3568H 写成 35 68，正确是 68 35，一个字节都不能错。

③ **CMP 后该用 JA 却用 JG。**F5H 是 245 还是 −11？无符号题用有符号指令，最大最小会找反。

自测 · 3 道

1. DW 1234H, 5678H 占几个字节？按地址从低到高依次是什么？
2. ORG 2000H 后写 DB 3 DUP(0FFH)，再写 DW 0，这个字的偏移是多少？
3. PUSH AX 之后 SP 怎么变？AX=2010H、SP=0070H 时，两个字节各存哪？

1. 4 个字节：34H 12H 78H 56H。

2. 3 个字节占 2000H~2002H，这个字从 **2003H** 开始。

3. SP 减 2 变成 006EH；10H 存 006EH，20H 存 006FH。

2020 卷 · 真题 3 道

2020 · 程序1

○ 会 ○ 卡 ○ 不会

子程序：

```
  CVT:CMP AL,30H
      JC L1
      CMP AL,3AH
      JNC L1
      AND AL,0FH
      JMP L2
  L1:MOV AL,0FFH
  L2:RET
```

问：① AL=34H 结果；② AL=8AH 结果；③ 最多几种输出？功能？

一句话：判断 AL 是不是 ASCII 数字，是就转 BCD。 ① 两条 CMP 把范围卡在 30H~39H，即字符 '0'~'9'。② AL=34H 在范围内，两跳都不成立，执行 AND AL,0FH。 ③ 34H AND 0FH = 04H。 ④ AL=8AH：与 3AH 比较无借位，JNC L1 成立。 ⑤ 跳到 L1 执行 MOV AL,0FFH，用 FFH 标记非数字。 ⑥ 输出 11 种：00H~09H 共 10 种，再加 FFH。 ⑦ 功能：ASCII 数字转 BCD，非数字返回 FFH。 **答案 ① 04H；② 0FFH；③ 11 种**

2020 · 程序2

○ 会　○ 卡　○ 不会

数据段段地址 8850H：

```
CNT1 EQU 3568H
CNT2 EQU 110H
ORG 5500H
VCHAR DB 4 DUP(20H)
VTIMCNT DW CNT1,CNT2
PVT1 DW VTIMCNT
PPVT1 DD PVT1
DW 0
```

画出 8850:5500H~550FH 各单元的存储情况。

一句话：按变量顺序排偏移，低字节在低地址。 ① 物理地址 = 88500H+5500H = 8DA00H，共 16 单元。② VCHAR 四个 20H：5500H~5503H。③ VTIMCNT 第一字 3568H：5504H=68H、5505H=35H。④ 第二字 0110H：5506H=10H、5507H=01H。⑤ PVT1 存偏移 5504H：5508H=04H、5509H=55H。⑥ PPVT1 存偏移 5508H 与段 8850H：550A~550DH=08H,55H,50H,88H。⑦ 末尾 DW 0：550EH=00H、550FH=00H。 **答案 20 20 20 20 | 68 35 10 01 | 04 55 | 08 55 50 88 | 00 00**

2020 · 程序3

○ 会　○ 卡　○ 不会

堆栈区 5600H:0000H~0300H，(SP)=0070H。问：1）栈顶地址；2）栈底地址；3）AX=2010H 压栈后画图并求 SP。

一句话：SP 指栈顶，压栈先减 2 再写。 ① 栈区物理地址 56000H~56300H。② 栈顶由 SP 指示：56000H+0070H = 56070H。③ 栈底是最高地址：56000H+0300H = 56300H。④ PUSH AX 先 SP←006EH，再写新栈顶。⑤ 低字节 10H 存 5606EH，高字节 20H 存 5606FH。⑥ 出栈时先读低地址送 AL，再读高地址送 AH，SP 加 2。 **答案 1）56070H；2）56300H；3）5606EH=10H、5606FH=20H，SP=006EH**

16

画图题专项：时序图 / 存储图 / 堆栈图 / 芯片连接图

约 10 分 · 40 分钟

四张图共用一套起手式：先摆空架子，再往里填

> **一句话：**画图题不考画技，考顺序 —— **格子先画，线名先写，电平最后填。**

? 有人一上手就画 ALE 的高电平，画到 T4 才发现格子不够，只能涂掉重来。
空架子摆对了，图就成了一半。

先讲清楚

一、落笔前先量"画多大"

第一笔不是内容，是定"这张图占多大地方"。

图	先数什么	占多大
时序图	几个 T	整页宽，一格 2~3cm
存储图	共几字节	竖排，一格一行
堆栈图	栈区范围	竖半页，高地址在上
连接图	几片芯片	横排，左 CPU 右芯片

第二笔打格钉位置，第三笔才写内容；**信号名写左边，时间和地址写在下边。**

> **总结四步：**定位置 → 画格子 → 写名字 → 填内容。

二、时序图：先 4 个格，再 6 条线

时序图是"同一段时间里几根线各自的电平"，所以先画时间尺子，再画线。

1. 横轴画 4 个**等宽**格，标 T1~T4；要插等待，就在 T3 后加一格 Tw
2. 左侧从上到下画 6 条**平行横线**，间距一样
3. 线左边写名字：CLK、A19~A0、ALE、M/IO#、RD#、D15~D0
4. 一格填一件事，先填 CLK

读周期里，六根线在四格里这样走：

信号	T1	T2	T3	T4

CLK	↑	↑	↑	↑
A19~A0	**地址有效**	撤销	—	—
ALE	**高**	低	低	低
M/IO#	1	1	1	1
RD#	高	**低**	低	高
D15~D0	高阻	高阻	**数据**	撤销

写周期只改两处：WR# 从 T2 起变低，数据 T2 就上线。

写是 CPU 往外送，器件得提前看到；读是器件送回，T3 才交。

老师第一眼看这三格：

- **ALE 只高 1 格**，两格就是锁存两次
- **地址与 ALE 同宽**，T2 必须撤
- **读数据从 T3 起**，早画等于时序反了

停一下：设备慢，Tw 插在哪一格？READY 什么时候被看？

Tw 插在 **T3 之后、T4 之前**，CPU 在 T3 末尾采样 READY。

READY 为低就插一格，插几格由设备速度决定。

三、芯片连接图：先摆方框，再连三类线

连接图分三列摆：左边 CPU，中间 373 和 138，右边存储芯片。

左列 CPU 引三束线；中上 373 管锁存，中下 138 管片选；右列芯片横排，**偶上奇下**。

只连三类线：

线	从哪来	到哪去
地址（高位）	A19~A14	138 的 G1、G2A#、G2B#、CBA
地址（片内）	A13~A1	过 373，再进芯片 A12~A0
片选	Y0#~Y7#	与 A0 / BHE# 相或，再进 CE#
数据	D7~D0 / D15~D8	偶片 / 奇片的数据脚

控制	RD#、WR#、M/IO#	OE#、WE#、参与使能

三样不画就白画：**373、138 的三个使能端、A0 与 BHE# 分奇偶。**

373 不能省：地址只在 T1 出现，不锁存就没了。

> 停一下：Y6# 出来了，为什么不能直接接两片 CE#?

Y6# 管的是 16KB 一整块，里面还分偶体和奇体两片。

直接并联，读一个字时两片同时输出，数据打架；所以要再与 A0、BHE# 相或，一路一片。

四、存储图：一列格子，低字节在上

存储图就是一列竖格子：左边地址，中间一个字节，右边变量名。

先数字节：

写法	占几格	说明
DB / DW / DD	1 / 2 / 4	按单元大小
4 DUP(20H)	4	次数乘单元
EQU	**0**	常量，不占存储

4 字节的例子，从 5500H 起：

地址	内容	变量
5500H	11H	BUF 第 1 格
5501H	11H	BUF 第 2 格
5502H	**34H**	NUM 低字节
5503H	12H	NUM 高字节

规矩两条：**低字节永远在上面那格**，因为上面就是小地址；双字先写偏移、再写段地址。

五、堆栈图：竖着长，SP 是一个箭头

堆栈图里高地址画在上面，SP 箭头指着"下一个要写的位置"。

栈向下长：压栈地址变小，新内容在旧内容下面。

1. 竖排栈区，最上写栈底（最高地址），最下是最低地址
2. 算出当前 SP，在那一格左边画箭头，标 SP=xxxxH
3. PUSH：**先 SP 减 2，再写两个字节**，箭头下移两格
4. 两个字节里，**低字节写在下面那格**（低地址）

SS=5600H，SP=0070H，AX=2010H，执行 PUSH AX：

量	值	怎么来的
栈底	56300H	56000H + 0300H
旧栈顶	56070H	56000H + 0070H
新 SP	**006EH**	0070H – 2
5606EH	**10H**	低字节落低地址
5606FH	20H	高字节落高地址

图上必须出现三样：**栈底那一格、SP 箭头、两格里各写了什么。**

POP 反过来：先读两个字节，再 SP 加 2，箭头往上。

必背 · 六条

定位置 → 画格 → 写名 → 填内容	四图通用
ALE 只高 1 个 T	地址只在 T1 有效
读：数据 T3 起；写：T2 起	方向定时间
一格一字节，低字节在上	DB1 DW2 DD4
栈向下长，PUSH 先减 2	高地址在上
373 + 138 使能 + A0/BHE#	连接图三件套

读总线周期 10 分 从白纸开始

题目：画 8086 读存储器总线周期，标 A19~A0、ALE、M/IO#、RD#、D15~D0。

第一步：定格。横轴 4 个等宽格，每格 2.5cm，写 T1~T4。

第二步：画线。从上到下 6 条平行线，间距一样。

第三步：写名。名字统一放左边，写完不再挪。

第四步：填 CLK。每格一个方波，上升沿落在格线上。

第五步：填地址。T1 整格有效，标 A0=0、BHE#=0；T2 起画虚线示撤销。

第六步：填 ALE。T1 整格高，T2 掉低并保持到 T4，宽度正好一格。

第七步：填 M/IO#。读存储器，四格全 1。

第八步：填 RD#。T1 高，T2 起低，T4 回高。

第九步：填数据。T1 T2 画斜线示高阻，T3 起有效，T4 撤。

验算三问：ALE 只有一格高？地址 T2 就走了？数据 T3 才出现？三个"是"就对了。

要插等待，就在 T3 后多画一格 Tw，其余信号原样不动。

⚠ **三个坑：**

① **ALE 画满整行。**它只在 T1 高，画两格等于地址锁了两次，整图作废。

② **高字节写在上面。**1234H 写成上 12H、下 34H，地址就反了。

③ **栈底画在下面。**栈往低地址长，栈底必须在上；画反了 SP 往哪移也全反。

自测 · 3 道

1. 时序图里 ALE 的高电平占几个 T？地址持续到第几格？
2. DW 1234H、5678H 从 5500H 排，四格分别是什么？
3. SP=0070H，PUSH AX（AX=2010H）后 SP 是多少？低字节写在哪格？

1. 占 1 个 T；地址到 T1 结束，T2 起撤销。

2. 5500H=34H、5501H=12H、5502H=78H、5503H=56H。

3. SP=006EH；10H 在 5606EH，20H 在 5606FH。

17 整卷速查：考前 15 分钟

考前 15 分钟 · 只看第一步

不复习知识，只记"拿到题第一行写什么"

一句话：每道题的第一行只写**一个数或一个式子**，写对了，后面自己会走。

? 考场上最慌的不是不会，是**盯着题不知道从哪落笔**。
下面这张表，就是那第一行。

先讲清楚

一、八类题，第一行写什么

每类题的第一行都是固定的，背下来就有起手。

题型	第一行写什么	接着做	收尾
求物理地址	**段×16 + 偏移**	方括号里先算偏移，BP 配 SS	5 位十六进制
存储器扩展	**片数 = 总容量 ÷ 单片**	末−首+1，列地址表	138 使能 + A0/BHE#
画时序图	**4 个等宽 T 格**	6 条线，一格填一件事	标 ALE 与数据位置
画存储/堆栈图	**一共几字节**	DB1 DW2 DD4，低字节在上	标变量名、SP 箭头

8259 中断	**7n+1 定片数**	ICW1→ICW4 按序写	两个 INTA# 的响应
8253 定时	**初值 = 输入频率 ÷ 输出频率**	控制字 + 方式	写出波形周期
8255 接口	**控制字（D7=1）**	A/B/C 口地址	数据往哪个口走
中断向量表	**表地址 = 类型号×4**	4 字节：IP 低、IP 高、CS 低、CS 高	00000H 起查

第一行永远是"算一个数"或"画一个框"，不是写结论。

因为式子写出来，错在哪一眼能看见；直接写答案，错了没处查。

二、三条通用规矩

所有大题都守这三条，守住了很少丢冤枉分。

规矩	怎么做
先式子，后答案	把 32KB÷8KB=4 写出来，再写"4 片"
单位不省	KB、Hz、ms 全写，换算先换成同一档
算完回代	地址代回范围、频率代回周期，对得上才收笔

举例：算完初值 200，代回 1MHz ÷ 200 = 5kHz，对得上才算完。

三、十五分钟怎么花

5 分钟看表，5 分钟背必背，5 分钟默写。

- 前 5 分钟：把上面八行从头到尾读一遍，只记第一行
- 中 5 分钟：盖住右半边，看题型说第一行
- 后 5 分钟：默写下面六条，写不出的那一条再回去看

> **进考场前默念：**先算数，再画图，最后写结论。

必背 · 六条

物理地址 = 段×16 + 偏移	BP 配 SS
片数 = 容量 ÷ 单片	容量 = 末−首+1
一个总线周期 = 4 个 T	Tw 插 T3 后
向量表地址 = 类型号×4	00000H 起
8259 级联 7n+1	3 片 = 22 级
8253 初值 = fin ÷ fout	方式 3 出方波

⚠ **三个坑：**

① **只写答案不写式子。**数字一错，老师找不到你在哪一步错，过程分也没了。

② **单位混着代。**32KB 直接除 8（其实是 8KB），答案变成 4000 片。

③ **公式背了不会套。**先找题目给的三个量，缺哪个就先算哪个。

自测 · 3 道

1. 1234:5678H 的物理地址是多少?
2. 32KB 空间用 8KB 芯片，要几片?
3. 一个总线周期几个 T? Tw 插在哪?

1. 12340H + 5678H = 179B8H。

2. 32KB ÷ 8KB = 4 片。

3. 4 个 T（T1~T4）；Tw 插在 T3 之后。