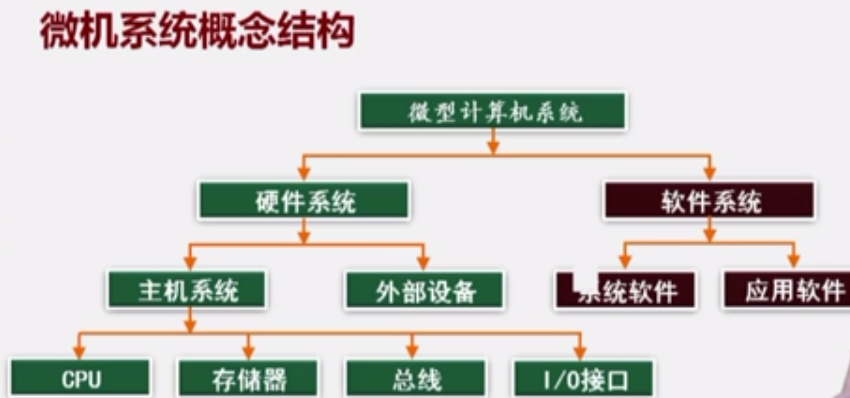
微机原理与接口技术

计算机的主要应用：数值计算、信息处理、**过程控制（本课程研究方向）**。

课程内容：数值信息表示、微型机基本原理、汇编程序设计、半导体存储器及其接口设计、输入输出技术（数字并行接口、模拟量并行接口）

### 一、微型计算机基础概念



1. 计算机系统的组成：硬件系统、软件系统

硬件系统：主机系统、外部设备（不能直接和CPU进行数据交换，需要通过输入输出接口与计算机进行数据交换）

主机系统：CPU、存储器、输入输出系统、总线

a. CPU：运算器、控制器、寄存器组

b. 存储器：内存储器、外存储器（联机外存：硬磁盘、固盘；脱机外存：移动存储设备）【机械硬盘接口：SATA】

内存条上芯片有8片。（内存由内存单元组成，每个单元存放1Byte数据）

内存容量：内存单元的个数。

内存分为：随机存取存储器（RAM）、只读存储器（ROM）

c. 输入/输出（I/O）接口：CPU与外部设备间的桥梁。

功能：数据缓冲寄存、信号电平或类型的转换、实现主机和外设间的运行匹配

d. 总线：一组导线和相关的控制、驱动电路的集合。

地址总线（AB）、数据总线（DB）、控制总线（CB）

【能够与CPU直接进行信息交换的部件属于主机系统，不能够与CPU直接进行信息交换的部件属于外部设备。】

1. 微型计算机的一般工作工程

**取指令** 🡪 **分析指令** 🡪 读取操作数 🡪 **执行指令** 🡪 存放结果

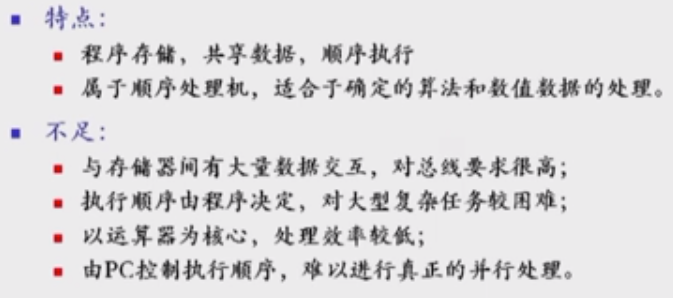
顺序执行、并行执行

加速比S = 顺序执行花费的时间/并行执行花费的时间

1. 冯诺依曼计算机

五个部件：存储器、运算器、控制器、输入设备、输出设备

存储程序工作原理，以运算器为核心。（如今，**大部分以存储器为核心**）



哈佛结构：

指令和数据分别存放在两个独立的存储器模块中。

CPU与存储器间指令和数据传送分别采用两组独立的总线。

1. ASCII码：128个字符（7bit，1Byte中最高位默认为0）

扩展ASCII码：8位（保括一些希腊字母和德语字母）

1. 数值的表示

浮点数，规格化：规格化过程中小数点后第一位必须为1。

规格化结果：正数补码表示为0.1xxx（包括1/2：0.100）；负数补码表示1.0xxx（除-1/2：1.100）。

例：浮点数规范化后，阶码和尾数都用补码表示，都是8位。求：

|  |  |  |  |
| --- | --- | --- | --- |
|  | 阶码 | 尾数 | 值(阶码 尾数) |
| 最大正数 | 最大正数 | 最大正数 |  |
| 最小正数 | 最小负数 | 最小正数 |  |
| 最大负数 | 最小负数 | 最大负数 |  |
| 最小负数 | 最大正数 | 最小负数 |  |

【注：最后两项颇有争议。有人认为最大负数为1.0111111，最小负数为1.0000000】

1. 无符号数加减运算溢出判断：最高位向更高位有进位（或借位）时产生溢出。

有符号数加减运算溢出判断：最高位进位状态⊕次高位进位状态 = 1时产生溢出。

【】

### 二、微处理器与总线

1. 8088/8086（第三代处理器）：8088是inter公司对之前机器的兼容版本，8088和8086内部宽度都是16位，8086对外总线也是16位，而8088对外总线是8位。

【8088/8086地址总线20位，8088数据总线8位，8086数据总线16位】

1. 8088/8086 CPU特点：
2. 采用并行流水线工作方式

CPU内部通过设置指令预取队列实现。

1. 对内存空间实行分段管理（实地址模式下的寻址）

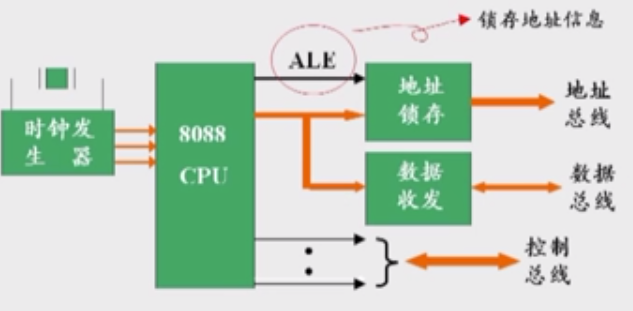
将内存分为4个段并设置地址段寄存器，以实现对1MB空间的寻址。

1. 支持协处理器

主要用来实现浮点数的计算。

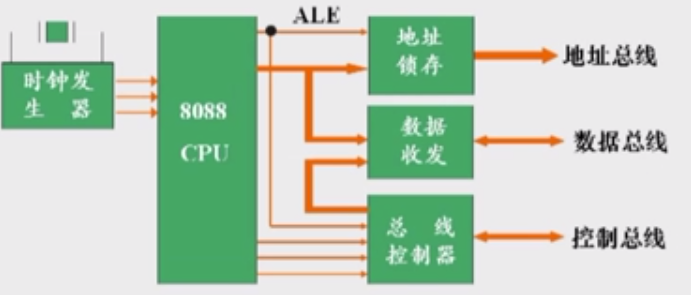
1. 8088/8086 CPU的两种工作模式
2. 最小模式

最小模式为单处理器模式，所有控制信号由微处理器产生。



1. 最大模式

最大模式为多处理器模式，部分控制信号由外部总线控制器产生【用于包含协处理器的情况下】



8088的工作模式是由引线的状态决定。为0工作在最大模式，为1工作在最小模式。

1. 8088的主要引线及内部结构

8088是一个40个引脚的一个双列直插式芯片。

* 8088最小模式下的主要引脚信息

1. 完成一次访问内存或接口所需要的主要信号

地址线和数据线：

* AD0-AD7：低8位地址和低8位数据信号分时复用。在传送地址信号时为单向，传送数据信号时为双向。
* A8-A15：8位地址信号。
* A16-A19：高4位地址信号，与状态信号分时复用。

主要控制信号：【#代表低电平有效】

* #WR：写信号
* #RD：读信号
* IO/#M：为“0”表示访问内存，为“1”表示访问接口
* #DEN：低电平有效，允许进行读/写操作（数据收发器）
* DT/#R：数据收发器的传送方向控制（数据收发器）
* ALE：地址锁存信号
* RESET：复位信号

1. 与外部同步控制信号

READY：外部同步控制信号，CPU访问一次内存或接口时，在第三个时钟周期后检测该信号，若为高电平表示访问正常，否则表示未就绪，需要在第三个周期后插入一个Twait周期，重复以上检测。（正常情况下，完成一次访问需要4个时钟周期）【一个总线周期等于4个时钟周期，每个时钟周期大致】



1. 中断请求和响应信号

INTR：外部可屏蔽中断请求输入端

NMI：外部非屏蔽中断请求输入端

#INTA：中断相应输出端

1. 总线保持和响应信号（CPU工作在直接存储器存取的时候：内存可以和外设直接进行数据交互）

HOLD：总线保持请求信号输入端。当CPU以外的其他设备要求占用总线时，通过该引脚向CPU发出请求。

HLDA：总线保持相应信号输出端。CPU对HOLD信号的响应信号。

* 8088和8086引线功能的比较

1. 数据总线宽度不同

8088的外部总线宽度是8位，8086是16位。

1. 访问存储器和输入输出控制信号含义不同

8088——IO/M=0表示访问内存。

8086——IO/M=1表示访问内存。

1. 其他部分引线功能的区别

* 8088内部结构（所有CPU都包含运算器、控制器、寄存器组三个部分）

1. 执行单元（EU）

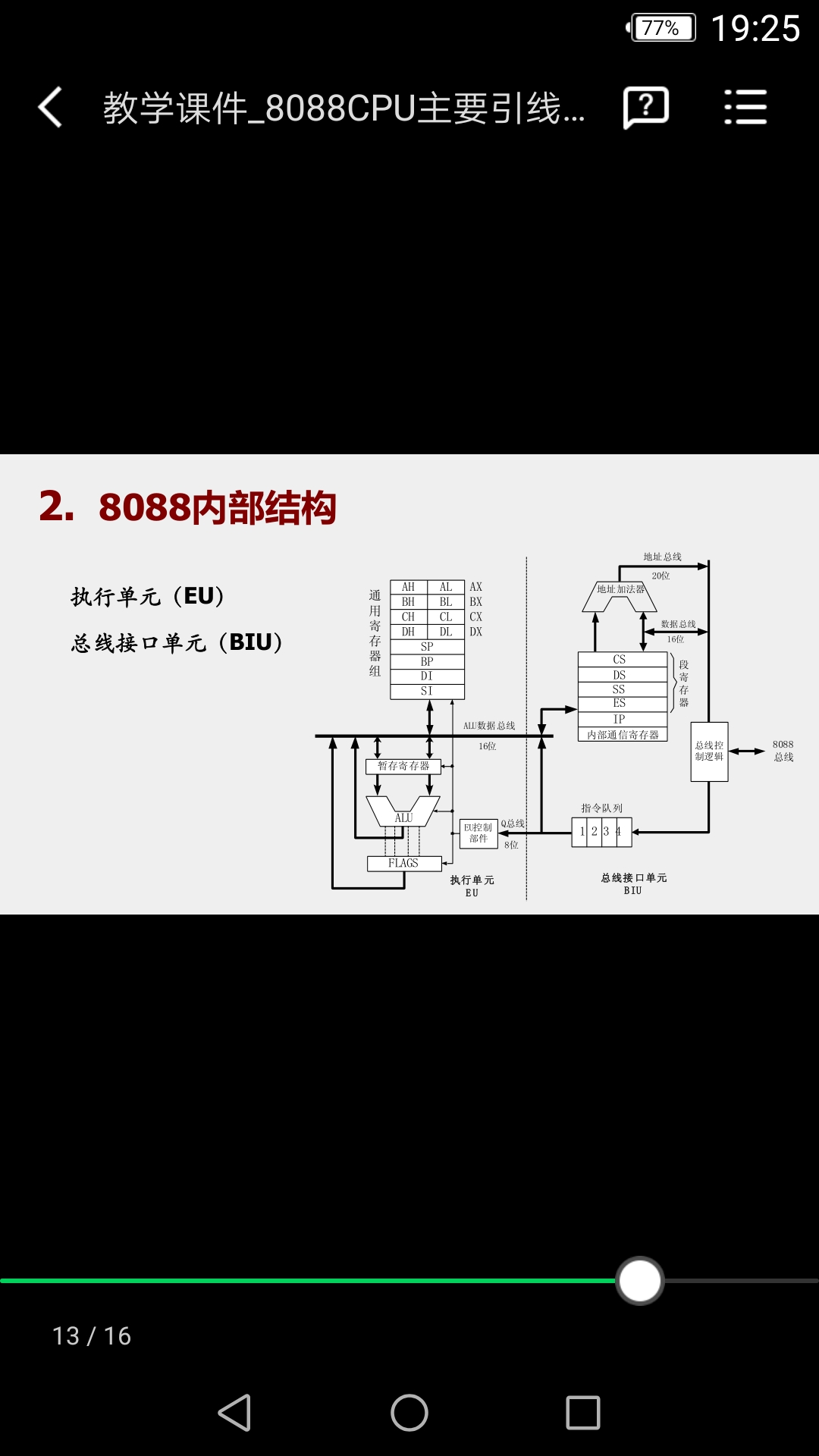
运算器

8个通用寄存器

1个标志寄存器FLAGS（标志两个数相加，有无进位、有位溢出等）

EU部分控制电路

1. 总线控制单元（BIU）



1. 内部寄存器

含14个16位寄存器，按功能分为三类：

1. 8个通用寄存器

* 数据寄存器（每个可拆分为两个8位寄存器）：AX、BX、CX、DX

AX：累加器，性能略高，普遍用来存放中间运算结果。所有I/O指令都通过AX与接口传送信息。

BX：基址寄存器，在间接寻址中用于存放基地址。

CX：计数寄存器，用于在循环或串操作指令终存放计数值。

DX：数据寄存器，在间接寻址的I/O指令中存放I/O端口地址；在32位乘除法运算时，存放高16位数。

- 地址指针寄存器：SP、BP

SP：堆栈指针寄存器，内容为栈顶的偏移地址。（有堆栈时，该寄存器即专用寄存器）

BP：基址指针寄存器，常用于在访问内存时存放内存单元的偏移地址。

【BX vs BP：作为通用寄存器，二者均可用于存放数据；作为基址寄存器，用BX表示所寻找的数据在数据段，用BP则表示数据在堆栈段。】

- 变址寄存器：SI、DI

SI：源变址寄存器

DI：目的变址寄存器

【变址寄存器在指令中常用于存放数据在内存中的地址。】

1. 4个段寄存器（存放相应逻辑段的段基地址，**段寄存器的值表明相应逻辑段在内存中的位置**）

逻辑段类型：

代码段：存放指令代码 CS

数据段：存放操作的数据 DS

附加段：存放操作的数据 ES

堆栈段：存放暂时不用但需保存的数据 SS

【在一个程序模块中，一种逻辑段只能有一个】

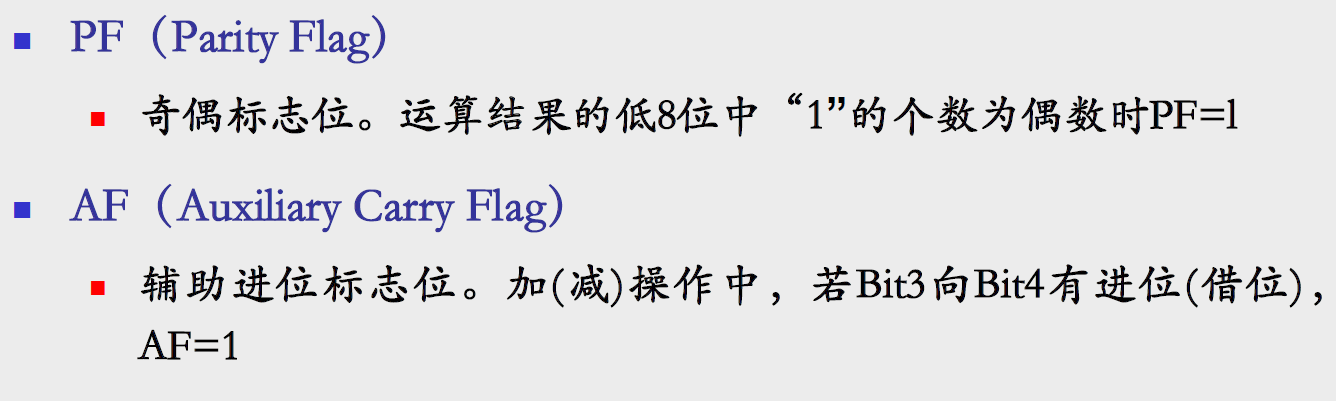
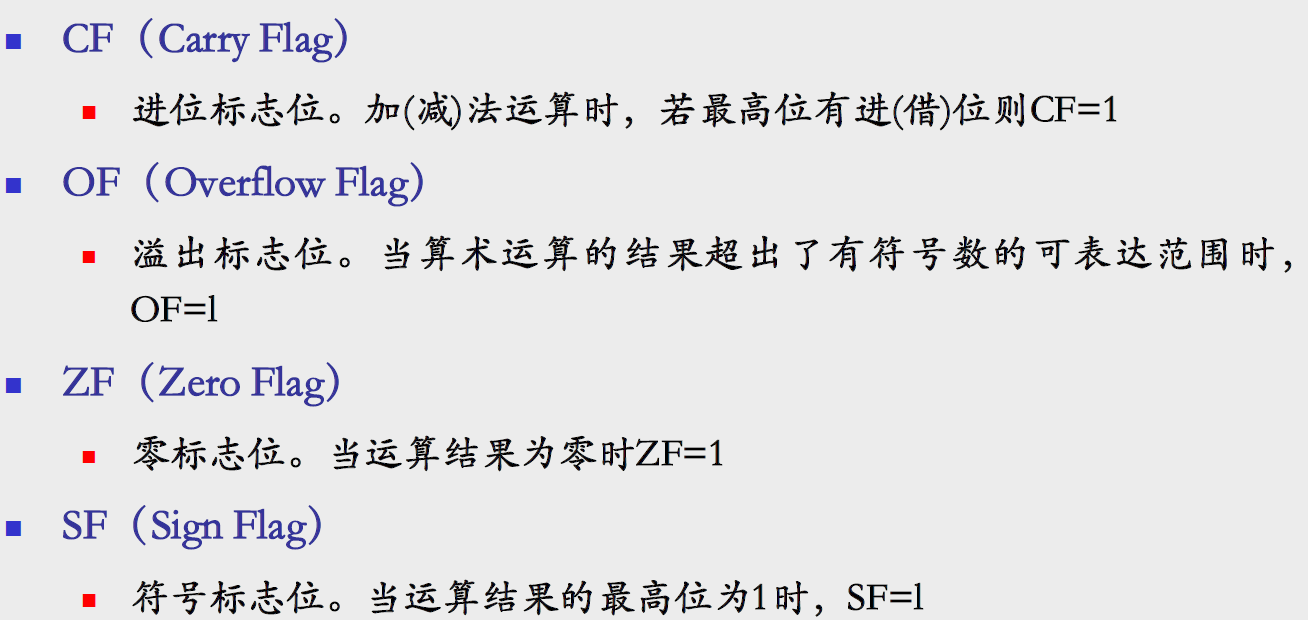
1. 2个控制寄存器

IP：指令指针寄存器，其内容为下一条要取的指令的偏移地址。

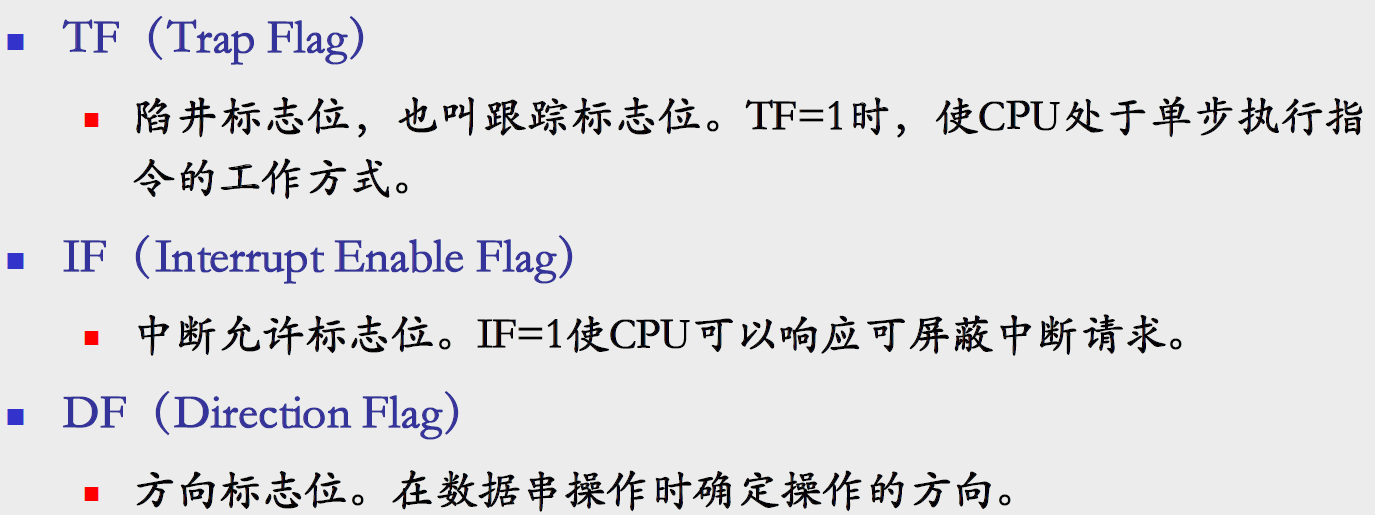
FLAGS：标志寄存器，存放运算结果的特征。（其中只有9位是有意义的）

6个状态标志位：表示运算结果的特征，CF、OF、ZF、SF、AF、PF

3个控制标志位：表示CPU当前的工作状态，IF、TF、DF



【注意：CF（无符号数溢出）、OF（有符号数溢出）、ZF、SF针对8位和16位都有效；PF、AF只针对低8位有效，AF表示第3位向第4位的进位，从0开始数】



【注意：TF=1表示中断】

1. 内存储器分段管理

* 内存单元的编址

内存每个单元的地址在逻辑上都由两部分组成：

* 段（基）地址

指示存储单元在整个内存空间中处于哪个段。

* 段内地址（相对地址/偏移地址）

指示存储单元在段中的相对位置（与段中第1个单元的距离）

* 实地址模式下的内存地址转换

物理地址 = 段基地址 \* 10H + 偏移地址

注意：1. 内存分段时，段首都是以节为边界起始的，即段首地址都必须能被16整除

2. 理论上，可以产生64K个逻辑段，但存放段地址的只有4个段寄存器。

3. 内存的分段是逻辑分段，不是物理分段。各个逻辑段在地址上可以相连，可以部分重合，也可以完全重合。

4. 每个内存单元具有唯一的物理地址，但可能有多个逻辑地址。

* 堆栈段

栈首即段首。若栈顶(SP为栈顶偏移地址) = 栈底，表示空栈；若栈顶 = 栈首，表示满栈。

1. 系统总线

* 总线时序

CPU工作时序：CPU各引脚信号在时间上的关系。

总线周期：CPU完成一次访问内存（或接口）操作所需要的时间。一个总线周期至少包括4个时钟周期。

* 总线
* 是一组导线和相关的控制、驱动电路的集合。是计算机系统中各部件之间传输地址、数据和控制信息的通道。
* 按层次结构分类：CPU总线、系统总线（连接CPU和接口间的总线，常见的两种标准为PCI总线和PCIE总线）、外部总线（接口和外设间的总线，常见的是USB总线）
* 今天的微机中大部分都是多总线结构。
* 总线的基本功能：

1. 数据传送：同步、半同步（类似于READY接口）、异步（需要接受到应答信号）
2. 仲裁控制
3. 出错处理
4. 总线驱动

* 总线主要性能指标

总线带宽（B/s）：单位时间内总线上可传送的数据量。总线带宽=位宽 \* 工作频率

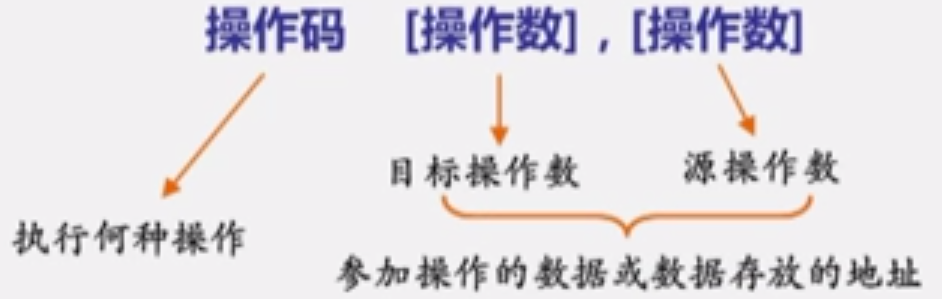
1. 指令系统
2. 基本概念

指令：控制计算机完成某种操作的命令。

指令系统：处理器所能识别的所有指令的集合。

指令的兼容性：同一系列机的指令都是兼容的。

1. 指令格式



零操作数指令：操作码

单操作数指令：操作码 操作数

双操作数指令：操作码 操作数 操作数

多操作数指令：三操作数及以上

1. 指令中的操作数

立即数、寄存器、存储器

【立即数无法作为目标操作数】

1. 寻址方式

* 操作数由指令直接给出

立即寻址 MOV AX,1200H（**立即数1200H在编译时被置于代码段**）

* 存放于寄存器中的寻址方式

寄存器寻址 MOV AX,BX

* 存放于存储器中的寻址方式

方括号中是偏移地址，逻辑段的段基地址通过默认或重设方式给出。

【存储器操作数的字长本身不确定，其字长取决于指令中另一个寄存器操作数，或通过其他方式指定字长】

1. 直接寻址 MOV AX,[1200H]

直接寻址方式下，操作数默认在数据段，但可以重设（MOV AX,ES:[1200H]）

1. 寄存器间接寻址 MOV AX,[BX]

但此处仅有4个通用寄存器可用于存放偏移地址（间址寄存器）：BX、BP、SI、DI

操作数的默认段地址取决于使用的间址寄存器（可以段重设）：

BX、SI、DI → 默认在数据段

BP → 默认在堆栈段

1. 寄存器相对寻址

MOV AX,[BX + DATA]（DATA可以为8位也可以为16位）

MOV AX,[BX]DATA 或者 MOV AX, DATA [BX] 也表示相同含义

相对寻址主要用于一位数组的操作。常将位移量作为“表头”地址，间址寄存器的值作为表内相对地址。

1. 基址变址寻址

操作数的偏移地址为 一个基址寄存器（BX/BP）的内容 + 一个变址寄存器（SI/DI）的内容。

操作数的段地址由选择的基址寄存器决定：基址寄存器为BX，默认为数据段；基址寄存器为BP，默认在堆栈段。

1. 相对基址变址寻址

基址寄存器 + 变址寄存器 + 位移量

操作数的段地址由选择的基址寄存器决定。

基址变址相对寻址主要用于二维表格操作。

1. 隐含寻址

MUL BL（AX = AL\* BL）

1. 8086指令集（共96条）

* 数据传送

1. 通用数据传送指令

* 一般数据传送指令MOV

注意：a. 两个操作数字长必须相同。

b. 两操作数不允许同时为存储器操作数。

c. 两操作数不允许同时为段寄存器。

d. 在源操作数是立即数时，目标操作数不能是段寄存器。

e. IP和CS不作为目标操作数，FLAGE一般也不作为操作数在指令中出现。

* 堆栈操作指令【先进后出、以字为单位】

压栈指令：PUSH OPRD

指令执行过程：a. SP -2 → SP（下面为高地址，上面为低地址）

b. 操作数高字节 → SP + 1

c. 操作数低字节 → SP

出栈指令：POP OPRD

指令执行过程：a. SP → 操作数低字节

b. SP + 1 → 操作数高字节

c. SP +2 → SP

不能使用POP给CS赋值

【注意：对于单操作数的指令，操作数不能是立即数；如果操作数是内存单元，要显式地给出操作数的字长；单操作数指令的操作数都是16位。】

* 交换指令 XCHG REG,REG

要求：两个操作数中不能出现立即数；两个操作数至少有一个是寄存器；不允许使用段寄存器。

* 查表指令 XLAT

说明：用BX的内容代表表格首地址，AL内容为表内位移量，将结果送入到AL。

* 字位扩展指令

说明：该指令均针对的是有符号数的扩展，有符号数扩展将符号位扩展到高位。

CBW 将AL内容扩展到AX

CWD 将AX内容扩展到DX AX

1. 输入输出指令

I/O端口：I/O接口中用于存储数据、可以直接被CPU访问的寄存器。

计算机输入输出系统中可以包含若干接口控制电路（芯片），每个接口中都包含了1个或多个端口。

* IN acc,PORT
* OUT PORT,acc

【acc代表AL/AX，但是不能是AH，PORT是端口地址】

* 直接寻址

端口地址为8位时，指令中直接给出8位端口地址，最大可寻址256个接口。

* 间接寻址

端口地址为16位时，指令中的端口地址必须由DX指定，最大可寻址64K个接口。

1. 地址传送指令

* 取偏移地址指令LEA REG,MEM（取近地址指针）

将变量的16位偏移地址写入到目标寄存器。（**源操作数一定是存储器操作数**）

LEA针对的变量是一个静地址指针，即在本数据段内，不考虑段基地址。

* LDS指令（取远地址指针，多模块编程使用）

LDS REG,MEM 将源操作数的偏移地址送到目标寄存器，将源操作数的段地址送到DS

* LES指令（取远地址指针，多模块编程使用）

LES REG,MEM 将源操作数的偏移地址送到目标寄存器，将源操作数的段地址送到ES

1. 标志传送指令

* LAHF：将FLAGES的低8位装入到AH

SAHF：将AH的写入到FLAGES的低8位

隐含操作数AH

* PUSHF

POPF

隐含操作数FLAGES

【除标志传送指令外，其他指令的执行都对标志位不产生影响。】

* 算术运算

1. 加法指令

* 普通加法指令ADD
* 带进位的加法指令ADC（**多用于多字节数的相加，使用前要将CF清零**。）

ADD、ADC的执行会对全部6个标志位产生影响。

* 加1指令INC

操作数不能是段寄存器和立即数，**指令执行不会对CF造成影响**。

1. 减法指令

* 普通减法指令SUB

SUB OPRD1, OPRD2：OPRD1 - OPRD2 → OPRD1

* 考虑借位的减法指令SBB
* 减1指令DEC（标志位影响同INC）
* 比较指令CMP

CMP OPRD1, OPRD2：OPRD1 - OPRD2

指令执行的结果不影响目标操作数，仅影响标志位。主要用于比较两个数的大小。

CMP AX, BX

* + 两个无符号数的比较：

若AX ≥ BX → CF = 0

若AX < BX → CF = 1

若AX = BX → CF = 1, ZF = 1

* + 两个带符号数的比较：

两个数的大小由OF和SF共同决定。

OF和SF状态相同时，AX ≥ BX

OF和SF状态不同时，AX < BX

* 求补指令NEG

NEG OPRD：0 – OPRD → OPRD

在绝大多数情况下，CF = 1，除了OPRD=0。

当操作数为80H(-128)或者8000H(-32768)，则执行后，结果不变，但OF置1，其他情况下OF均置0。

对一个负数取补码就相当于用零减去此数。

* 逻辑运算和移位
* 串操作
* 程序控制
* 处理器控制