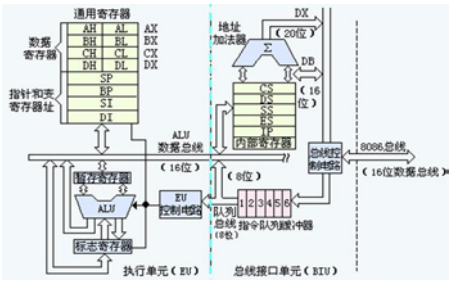
1.1 微机系统 MCS

微型计算机系统 MCS 包括硬件和软件；硬件：微机 MC、外围设备和电源；MC：微处理器 MP、存储器、I/O 设备和系统总线；MP：算术逻辑单元 ALU、控制单元 CU 和寄存器阵列 RA；系统总线（连接 CPU、I/O 设备和存储器）：地址总线 AB、数据总线 DB 和控制总线 CB

1.2 微型计算机

（单总线结构，双总线结构，双重总线结构；地址总线，数据总线和控制总线；面向系统的总线结构）；微型计算机系统（硬盘是外设）



微处理器：

ALU：执行算术和逻辑操作以及循环移位等。两个操作数一般来自累加器 **A**ccumulator 和内部 DB，可以是数据寄存器 **D**ata**R**egister 或寄存器阵列 RA 中的。运算结果送回 A 暂存。

控制部件：产生一定的时序，控制指令所规定的操作的执行。(1)指令寄存器 **I**nstruction**R**egister 存放从存储器取出的将要执行的指令。(2)指令译码器 **I**nstruction**D**ecoder 对指令寄存器 IR 中的指令进行译码，确定对应操作。(3)可编程逻辑阵列 **P**rogrammable**L**ogic**A**rray 产生取指令和执行指令所需的操作控制信号。

内部寄存器：

累加器 A：最常用的寄存器。在进行算术逻辑运算时：运算前保存一个操作数；运算后保存结果。

数据寄存器 DR：暂存数据或指令。从存储器读出时，若读出指令，指令通过内部 DB 送到指令寄存器 IR；若读出数据，则通过内部 DB 送到有关的寄存器或运算器。向存储器写入数据时，经 DR 通过 DB 写入。

程序计数器 PC：PC 中存放待取出指令的地址。根据 PC 中的指令地址从存储器中取出要执行的指令。程序通常按顺序执行。PC 能够自动加 1，因此总是指向下一字节或下一条指令(对单字节指令而言)所在的地址。

地址寄存器 AR:存放正要取出的指令的地址或操作数的地址。取指令时，将 PC 中的指令地址送到 AR，据此从存储器中取出指令；取操作数时，将操作数地址通过内部 DB 送到 AR，据此从存储器中取出操作数；向存储器存入数据时，先将待写入数据的地址送到 AR，据此向存储器写入数据

标志寄存器 F：寄存指令产生的结果或状态的标志信号。

存储器：存放用二进制代码的数据和程序。字节 byte：8 位；字 word：两个字节，16 位；字长：DB 一次操作能处理的位数即位长，并由此而定义是多少位的计算机。

1.3 冯•诺依曼型数字计算机工作原理

计算机系统由运算器、控制器、存储器、输入输出设备组成。指令：操作码+操作 **3.1 8086/8088 微处理器**

8086：16 位，20 根地址线，可寻址内存 2^{20}=1M
16 根 DB，地址以段地址和偏移地址分开存储

8088：8 根 DB 以兼容原外围芯片，其它与 8086 一致

BIU：4 个 16 位段寄存器和 16 位 IP

Code**S**egment：码段寄存器
Digit**S**egment：数据段寄存器
Extra**S**egment：附加数据段寄存器
Stack**S**egment：堆栈段寄存器
Instruction**P**ointer：指针寄存器

EU：16 位 ALU、16 位 F、数据暂存寄存器、16 位通用寄存器

Accumulator**X**：
Base**X**(Register)：
Count**X**：
Data**X**：
Stack**P**ointer：指向堆栈顶端
Base**P**ointer：默认段寄存器为 SS
Source**I**ndex：
Destination**I**ndex：
Carry**F**：进位或借位时为 1
Parity**F**：结果低 8 位中含有偶数个 1 时为 1
Auxiliary**F**：低位向高位进位或借位时为 1
Zero**F**：结果为 0 时为 1
Sign**F**：即结果的最高位
Overflow**F**：结果的符号位错误时为 1
Direction**F**：控制数据串操作指令方向，为 1 时递减
Interruptenable**F**：为 1 时允许接受中断请求
Trap**F**：为 1 时单步调试

总线周期：

读：

CLK	T1	T2	T3	T4
AD_{15}~AD_0	输出地址		高阻态 O→I	输入数据
A_{19}/S_6~A_{16}/S_3	输出地址		输出状态	

写：

CLK	T1	T2	T3	T4
AD_{15}~AD_0	输出地址		输出数据	
A_{19}/S_6~A_{16}/S_3	输出地址		输出状态	

若存储器/外设慢，则在 T3 前向 CPU 发出“未准备好”，CPU 在 T3 后进入 Tw 直至收到“已准备好”

3.2 8086/8088 的最小/最大工作方式

Byte**H**igh**E**nable/S7：数据总线高 8 位使能和状态复用信号，在总线周期 T1 状态，$\overline{BHE}$有效，表示数据线上高 8 位数据有效，在 T2～T4 状态，$\overline{BHE}$/S7 输出状态信息。S7 在 8086 中未定义。

MN/$\overline{MX}$：最小/最大方式

$\overline{RD}$：读信号，输出，三态。$\overline{RD}$有效，表示 CPU 执行一个对存储器或 I/O 端口的读操作，在一个读操作的总线周期中，$\overline{RD}$在 T2～T3 状态中有效，为低电平。

READY：准备好信号，输入。

$\overline{TEST}$：测试信号，输入。$\overline{TEST}$和 WAIT 结合使用，在 CPU 执行 WAIT 时空转进行等待。只有当 8086 检测到 $\overline{TEST}$有效时，才结束等待，继续执行 WAIT 后的指令。

INTR：可屏蔽中断请求，输入。IF&&INTR==1 时，CPU 结束当前指令后，响应 INTR 中断。

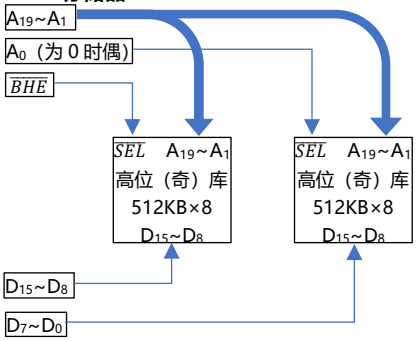
Non**M**askable**I**nterrupt：非屏蔽中断请求，输入。NMI 不受 IF 的影响。当 CPU 检测到 NMI 有一个上升沿触发的信号以后，CPU 执行完当前指令便响应中断类型号为 2 的非屏蔽中断请求。

CLK：时钟信号，输入。占空比为 33%。由 8284 提供

RESET：输入。维持 4 个周期有效时，总线无效，CPU 中止操作并清零 F，IP，DS，SS，ES 及指令队列，设 CS 为 FFFFH（-1）。撤除时，从 FFFF0H 开始执行程序。

最小工作方式：只用一个处理器，所有总线控制信号由 8086/8088 产生。

3.3 存储器



偏移地址：相对段首地址的偏移量，16 位

逻辑地址：段地址：偏移地址

物理地址：20 位，段地址左移+偏移地址

有效地址：组成偏移地址的地址。

EA=基地址 BX/BP+变地址 SI/DI+位移量 D（0/8/16）

堆栈：

SP：0000→FFFE→FFFC→……，每次必定操作两个字节

3.4 指令系统

1) 固定寻址：某些操作被指定了寄存器
2) 立即数寻址：须为 8/16 位整数，只能为源操作数，且须与目标操作数字长一致
3) 寄存器寻址：源与目标字长须一致
4) 存储器寻址：
   a) 直接寻址：直接给出 EA：[1680H]，段地可隐含或用 ES、SS 指定；操作数地址可用变量
   b) 间接寻址：由 BX/BP+SI/DI 组成，实际运算为加法，可以单独用，代码中不加符号。e.g.:[BX][SI]or[BP]。基址加变址寻址：必须用 BYTE PTR、WORD PTR 或 DWORD PTR 确定类型
5) 其它寻址：
   a) 串操作指令：源串的首字/字节须在 SI，目标首字/字节须在 DI。自动遍历
   b) I/O 端口：
      i. 立即数：IN AL, n;0<n<255；
      ii. 间接：用 DX，0~65535
   c) 转移类指令：段内/段间 + 直接/间接 转移 JMP 指令

所有指令不能用两个存储器寻址方式，目标不为立即数

1) 数据传送

MOV d,s; 不影响标志位
PUSH s; 不可为立即数 POP d;
XCHG d,s; 不可为立即数
XLAT; 查表
LEA d,s; load s 的 EA 到 d，EA 由 s 寻址方式决定
LDS d,s; 同上，且段地址存入 DS LES d,s; ES
LAHF; 将 F 低位装入 AH，影响 F SAHF; 相反
PUSHF; 将整个 F 压入栈 POPF;弹出 F
IN 累加器,端口; 输入字节/字 累加器：AL/AX
OUT 端口,累加器; 输出 端口：8 位立即数或 DX

2) 算术运算

ADD d,s ADC d,s; d=d+s+CF INC d; 加 1
SUB d,s SBB d,s; d=d-s-CF DEC d; 减 1
NEG d; 取负，有符号 CMP d,s; 比较
MUL s; 无符乘法，被乘数 AL/AX，结果 AX/DX+AX
s 不能为立即数，只影响 CF 和 OF
IMUL s; 有符乘法，(和 MUL) 进到高位时，CF=OF=1
DIV s; 无符除法，被除数 AX/DX+AX，除数 s，商 AL/AX，余数 AH/DX
IDIV s; 有符除法，都是有符数，(和 DIV) 不改 F
除数为 0 或商溢出等错误，由系统直接转入 0 型中断
商溢出，是指被除数高一半的绝对值大于除数的绝对值时，商超出了 16 位的表示范围（字操作）或 8 位的表示范围（字节操作）。
CBW; 字节扩展为字
AH= 00H，当 AL 的最高有效位为 0 时
AH= FFH，当 AL 的最高有效位为 1 时
CWD; 字扩展为双字
DX=0000H 当 AX 的最高有效位为 0 时
DX=FFFFH 当 AX 的最高有效位为 1 时
都不影响状态标志位。
ASCII 转换成 BCD 码只要把 ASCII 码的高 4 位清零
DAA;加法的十进制调整 DAS;减法（AX）
AAA;加法的 ASCII 调整 AAS;减法(影响 AF,CF)(AL)
调整产生的进/借位值存入 AH
AAM 乘法
AAD 除法的 ASCII 调整(和 AAM：AX，影响 SF,ZF, PF)

3) 逻辑运算和移位循环指令

AND d,s OR d,s NOT d XOR d,s
TEST d,s; 只改 F 的按位与，不返回值
SHL d,count; 逻辑左移 SHR d,count; 逻辑右移
SAL d,count; 算术左移 SAR d,count; 算术右移
ROL d,count; 循环左移 ROR d,count; 循环右移
RCL d,count; 带进位循环左移
RCR d,count; 带进位循环右移 count 为 1 或 CL

4) 串处理指令

自动修改 SI/DI，串长≤64KB
MOVS 目标串，源串; 串传送
MOVSB (目标串，源串); 字节传送
MOVSW (目标串，源串); 字传送 DS:SI→ES:DI
CMPSB / CMPSW (目标串,源串); 串比较，只改 ZF
SCASB / SCASW (目标串); 串搜索
ZF=1：找到；CF=0：终未找到；AX/AL：关键字
LODSB / LODSW (源串); 读串，存入 AX/AL
STOSB / STOSW (目标串); 将 AX/AL 写入目标串
REP ;重复执行串指令，(CX)=重复次数

①CX=0 时，串指令执行完毕，否则执行② ~ ④
②CX—
③执行串指令 MOVS 或 STOS
④重复执行①
REPE/REPZ　　!CX||!ZF 时停止
REPNE/REPNZ　　!CF||ZF 时停止

5)　程序控制

JMP 标号;　跳转　　CALL;　　调用过程;
RET 弹出值;　　从过程返回
JA/JNBE, JAE/JNB, JB/JNAE, JBE/JNA (无符号数)
JG/JNLE, JGE/JNL, JL/JNGE, JLE/JNG (有符号数)
JC,JNC CF 转移　JO,JNO OF 转移　JS,JNS　SF 转移
JP/JPE、JNP/JPO　PF 转移
JZ/JE、JNZ/JNE　ZF 转移　　都有操作数
LOOP　!CX 循环　LOOPE/LOOPZ　ZF&&!CX
LOOPNE/LOOPNZ　!ZF&&!CX　JCXZ　!CX
INT 中断类型　INTO　溢出中断　IRET 中断返回
JMP SHORT LABLE　　;段内短
JMP LABLE　　;段内近（缺省）
JMP reg\JMP WORD PTR OPR ;段内间接
JMP FAR PTR label　　;段间直接
JMP DWORD PTR [SI]　;段间间接
;所有条件转移都是短转移

6)　处理器控制指令

CLC　清 CF　STC　置 1CF　CMC　取反 CF
CLD 清 DF　STD 置 1DF　CLD 清 DF　STD 置 1DF
WAIT　等待，与 ESC 呼应　ESC　交权给协处理器
LOCK (前缀) 锁住总线　HLT　暂停　NOP　空操作

4.1　程序设计语言概述

DB：1B；DW：2B；DD：4B；DQ：8B；DT：10B
例 1：DW：16 位偏址；DD：16 位基址和 16 位偏址

```
FOO SEGMENT AT 55H   ;段基址为 55H
      ZERO     DB 0
      ONE   DW ONE          ;0001H
      TWO   DD TWO          ;0055H,0003H
      FOUR DW FOUR+5        ;7+5=12
      SIX    DW ZERO-TWO    ; 0-3=-3
      ATE    DB 5*6         ; 30
FOO ENDS
```

例 2：STR1 DB 'AB'　　;将 A 放低字节;
　　STR2 DW 'AB'　　;反之

4.2　8086/8088 汇编语言的基本语法

每个段都有一个名字(叫段名)，以 SEGMENT 作为段的开始，以 ENDS 作为段的结束，这两者(伪指令)前面都要冠以相同的名字。　　段可以从性质上分为代码段、堆栈段、数据段和附加段 4 种,但代码段与堆栈段是不可少的,数据段与附加段可根据需要设置。

1)　指令语句的格式

[标号:] [前缀] 指令助记符 [操作数表] [;注释]

2)　伪指令语句的格式

[名字] 伪指令助记符 [参数表] [; 注释]

标号

1）NEAR：只能被标号所在段内的转移和调用指令访问（即段内转移）
2）FAR：可以被其他段（不是标号所在段）的转移和调用指令访问（即段间转移），也可为段内转移。

操作数

- 当基址寄存器为 BP 时,相对应的段寄存器为 SS(堆栈段寄存器)。
- 对串操作，源串起始地址为 DS:SI，允许使用段超越前缀修改源串段地址。目的串起始地址为 ES:DI，禁止用段超越前缀修改目的串地址 ES。
- 可以采用“段超越”前缀，来改变段寄存器。

　　ADD AL, SS:[DI+3]

运算符

算术运算符：+、 -、*、/、MOD、SHR、SHL
逻辑运算符：AND、OR、XOR、NOT
关系运算符：EQ(=),NE(≠),LT(<),GT(>),LE(≤),GE(≥)
当关系成立的时候，结果为 FFFFH，否则为 0.

“变量”和“标号”的区别

变量指向的是数据段或者附加段中的某个数据项，标号指向的是代码段某条指令
变量的类型是数据项存取单位的大小，标号的类型是 NEAR 或 FAR

地址表达式的运算规则

1)变量或者标号加上或者减去某个结果为整数的数值表达式，其结果仍为变量或者标号，指向某存储单元。
2)同一段内的变量或者标号可以相减,其结果不是地址,而是一个数值，该数值表示两者间相距的字节数。

合成运算符

PTR:指定操作数的类型属性,它优先于隐含的类型属性。格式：类型 PTR 变量[± 常数表达式]
类型:BYTE,WORD,DWORD,FWORD,QWORD, TBYTE
THIS:建立指定类型的存储器地址或者操作数,但不分配存储区。

例：DATAB EQU THIS BYTE
　　DATAW DW ?

1. DB 是唯一能定义字符串的伪操作，字符串用 ‘’ 括起来，串中的每个字符占用一个字节，在存储器内用相应的 ASCII 码表示。
2.当 $ 用在伪指令的参数字段时，它所表示的是地址计数器的当前值

操作数为地址表达式：数据定义时，出现在地址表达式中的变量表示该变量的偏移地址，用根据偏移地址计算出的地址表达式的值初始化相应的存储单元。

格式 1：DW 地址表达式
功能：　用该地址表达式的计算值初始化相应的存储字
格式 2：DD 地址表达式
功能：地址表达式结果对应的段地址(高位字)和偏移地址（低位字）来初始化相应的两个存储字。

```
名字 EQU 表达式
名字 = 表达式
变量名/标号名 LABEL 类型
```

1. EQU 伪操作不允许重复定义

```
TMP EQU 5
TMP EQU TMP+1
```

2.在 EQU 语句的表达式中，如果有变量或标号的表达式，则在该语句前应该先给出它们的定义。

伪指令语句

段名 SEGMENT [定位类型] [组合类型] [‘类别’]
……
段名 ENDS
注意:
（1）SEGMENT 和 ENDS 成对出现
（2）段名由用户命名。
（3）段体：对于数据段、附加段和堆栈段来说，段体内一般是存储单元的定义、分配等伪指令语句；对于代码段中则主要是指令及伪指令语句。
（4）定位类型：表示该段对起始边界地址的要求：
B Y T E：起始地址无要求
W O R D：起始地址须为偶数
P A R A：起始地址：XXXX0H
P A G E：起始地址：XXX00H，PARA 为缺省类型

组合类型	说 明
NONE(缺省)	该段为私有段，连接时将不与其它模块中的同名段合并
PUBLIC	该段连接时将与其它同名段连接在一起，连接次序由连接命令指定
COMMON	该段在连接时与其它同名段有相同的起始地址，所以会产生覆盖
AT表达式	可以用于指定段地址，段地址＝表达式的值，其值必为16位,但AT不能用来指定代码段
MEMORY	该段在连接时被放在所有段的最后（地址最高部分），如果几个段均为该类型，则认为第一个为MEMORY类型，其余为COMMON类型
STACK	将多个同名堆栈段连接在一起，SP设置在第一个堆栈段的开始，SS和SP一起指向堆栈段的开始位置

ASSUME 伪指令：建立段和段寄存器关系的伪指令
格式：ASSUME 段寄存器名: 段名，…
例: ASSUME CS: SEGA, DS: SEGB, SS:NOTHING
* ASSUME CS:SEGA 表示 CS 被设定为以 SEGA 为段名的代码段的段地址寄存器

注意:
1. 其中段寄存器名必须是 CS、DS、ES 和 SS 中的一个，而段名必须是由 SEGMENT 定义的段名。
2. 由于 ASSUME 伪指令只是指定某个段分配给哪一个段寄存器，它并不能把段地址装入段寄存器中，所以在代码段中，还必须把段地址装入相应的段寄存器中
3. 在程序中不需要用指令装入代码段的段地址，因为在程序初始化时,装入程序已将代码段的段地址装入 CS 寄存器

ORG 伪指令:

ORG 伪指令用来表示起始的偏移地址，紧接着 ORG 的数值就是偏移地址的起始值。ORG 常用在数据段指定数据的存储地址，有时也用来指定代码段的起始地址。
*ORG 指出其后的程序段或数据块的起始地址偏移量
SEG：段地址；OFFSET：偏移地址；SIZE：数据字节总数；LENGTH 数据项总数

过程定义伪指令

格式：

```
过程名 PROC Attribute
……
过程名 ENDP
```

过程名：为一个标识符，是子程序入口的符号地址，与标号的作用相同。
属性：
(1) 若调用程序和过程在同一个代码段中，则用 NEAR
(2) 若调用程序和过程不在同一个代码段中，则用 FAR
分类：
(1) 外部过程：调用该过程的主程序与该过程不在同一源文件中。此时需在主程序文件中说明该过程(设过程名为 PROCD):　　EXTRN PROCD: FAR
同时在定义该过程的程序文件中说明该过程可以被其它程序文件调用:　　PUBLIC PROCD
(2) 内部过程：调用该过程的主程序和该过程在同一个源文件中。内部过程又分为段内过程和段外过程。
INT n 中断指令（interrupt），n 为中断类型号
执行操作:
① 入栈保存 FLAGS：SP← SP - 2，(SP) ← FLAGS
② 入栈保存返回地址:　　SP← SP - 2，(SP) ← CS
　　SP ← SP - 2，(SP) ← IP
③ 转中断处理程序：IP ← n×4，CS ← n×4 + 2
INTO 溢出则中断（中断类型为 4），相当于 INT 4
执行操作：若 OF=1（有溢出），则:
① 入栈保存 F：SP ← SP - 2，(SP) ← FLAGS
② 入栈保存返回地址:　　SP←SP - 2，(SP) ← CS
　　SP←SP - 2，(SP) ← IP
③转中断处理程序: IP←4×4=10H,CS←4×4 + 2= 12H
注意:
●根据中断类型号（n×4）到中断向量表中取得中断例行程序的入口地址，分别将对应的内容送给 IP 和 CS，可以以实现调用中断例行程序的功能。隐含的中断类型号为 3，即 INT 等价于 INT3
● INT 指令(包括 INTO)执行后，把 IF 和 TF 置 0，但不影响其它标志位。
IRET 中断返回指令（return from interrupt)
执行操作:
① 返回地址出栈：IP ← (SP)，SP ← SP + 2
　　CS ←(SP)，SP ← SP + 2
② FLAGS 出栈：FLAGS ← (SP)，SP ← SP + 2
调用 DOS 或 BIOS 中断功能时，有以下基本步骤:
1、 将调用参数装入指定的寄存器中;
2、 如需功能号，把它装入 AH;
3、 按中断号调用 DOS 或 BIOS 中断;
4、 检查返回参数是否正确。
注：中断向量分配情况
0～1FH, 80H～F0H 为 BIOS 中断向量号
20H～3FH 为 DOS 中断向量号
40H～7FH 为用户备用

功能	矢量号	功能号	备注
返回 DOS	21H	4CH	
设置中断向量	21H	25H	DS=段地址；DX=偏移地址；AL=中断类型码
键盘输入一字符并显示	21H	01H	AL=输入字符的 ASCII 码
字符串输入	21H	0AH	DS:DX 指向输入缓冲区；输入缓冲区第一个单元设置字符串最大输入长度；输入缓冲区第二个单元为实际输入字符串长度
显示字符	21H	02H	DL=显示字符的 ASCII 码
显示以‘$’结尾的字符串	21H	09H	DS:DX 指向字符串的首地址