**\_02\_寄存器（CPU工作原理）**

**0.基础知识概述**

（1）CPU概述

一个典型的CPU，由运算器、控制器、寄存器等器件组成。这些CPU内部器件由内部总线相连。

内部总线：连接CPU内部的各器件。

外部总线：连接CPU与主板上的其它器件。

注意：8086CPU是一个16位CPU，表现为以下几个特征：

①运算器一次最多可以处理16位的数据。

②寄存器的最大宽度为16位。

③寄存器和运算器之间的通路是16位的。

（2）寄存器概述

8086CPU中，所有寄存器都是16个比特位，也就是2字节（1个字）。

8086CPU有14个寄存器，分别是：AX、BX、CX、DX、SI、DI、SP、BP、IP、CS、SS、DS、ES、PSW

**①数据寄存器：**

**AX (Accumulator Register)：累加器寄存器，常用于算术和逻辑运算，并存储中间结果。**

**BX (Base Register)：基址寄存器，常用于与内存地址相关的操作。**

**CX (Count Register)：计数寄存器，常用于循环操作和字符串操作中的计数值。**

**DX (Data Register)：数据寄存器，常用于 I/O 指令和乘法/除法操作。**

**②指针和索引寄存器：**

**SI (Source Index)：源索引寄存器，常用于字符串操作中的源地址。**

**DI (Destination Index)：目标索引寄存器，常用于字符串操作中的目标地址。**

**③段寄存器：**

**CS (Code Segment)：代码段寄存器，存储当前执行的指令的段地址。**

**DS (Data Segment)：数据段寄存器，存储数据操作数的默认段地址。**

**SS (Stack Segment)：堆栈段寄存器，存储堆栈段的段地址。**

**ES (Extra Segment)：附加段寄存器，在字符串操作中作为额外数据段的段地址。**

**④指令指针寄存器：**

**IP (Instruction Pointer)：指令指针寄存器，存储 CPU 当前执行的指令的偏移地址。在 8086 的实际编程中，你通常会看到 EIP（扩展指令指针），但在 8086 的上下文中，它通常被称为 IP。**

**⑤标志寄存器：**

**FLAGS：标志寄存器，存储处理器状态和控制位，如溢出、符号、零、进位等。这些标志在算术、逻辑和比较操作后被设置或清除。**

**1.通用寄存器**

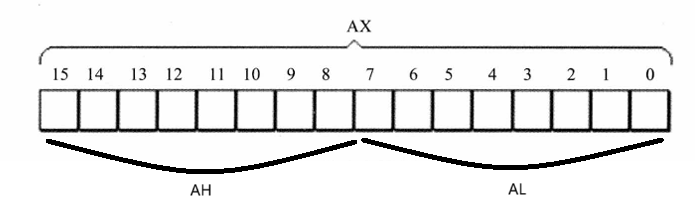
（1）基础介绍

通用寄存器常用于存放一般性数据。

8086CPU的通用寄存器为：AX、BX、CX、DX。

AX、BX、CX、DX均为16位寄存器，以AX为例：

AX可拆分为两个独立的8位寄存器AH（High）和AL（Low）使用，其中AH表示高8位，AL表示低8位。（BX、CX、DX的拆分同理）



注意事项：

①若把AX看作AH、AL，这两个寄存器虽然内存空间连续，但是它们是两个独立的寄存器，二者互不影响。

e.g. MOV AX,00FF // 此时AX为00FF

ADD AL,01 // 此时AX为0000

这里，操作的对象是AL，则AL和AH是两个独立的寄存器，AL的值不会溢出到AH中去。

**2.8086CPU的寻址方式（重点）**

（1）基础概念

①物理地址：对于CPU而言，计算机中所有的内存单元构成的存储空间在是一个一维的线性空间，每个存储单元唯一对应一个物理地址。

②8086CPU：是16位CPU，其运算器和寄存器都是16位的。

②8086计算机的地址总线：是20位地址总线，对应2^20个地址空间，即寻址能力为1M。

（2）**物理地址的计算：段地址\*16+偏移地址=物理地址**

①使用该计算方式的原因：

**8086计算机的地址总线是20位的，所以其物理地址也是20位的。**

**8086CPU本身是16位CPU（寄存器也是16位），所以其段地址和偏移地址都是16位的。**

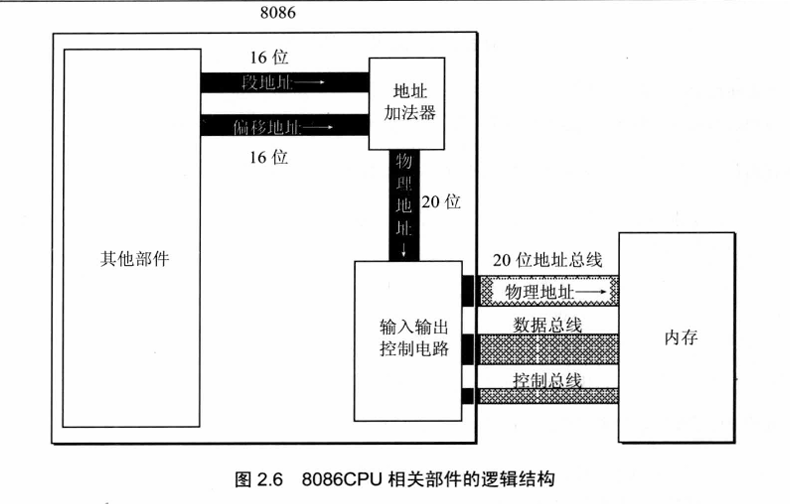
**因此，8086CPU内部需要用两个16位地址来合成20位的物理地址，即：“段地址\*16+偏移地址=物理地址”。**

②8086CPU物理地址的计算方法（以下三种方法等价）

**段地址<<4 + 偏移地址 = 物理地址**

**段地址\*16 + 偏移地址 = 物理地址**

**段地址\*10H + 偏移地址 = 物理地址**



（3）段的概念

段：Segment

注意事项：内存本身是连续的，并没有被分段。段的划分来自于CPU，由于8086CPU用“段地址\*16+偏移地址=物理地址”的方式计算物理地址，才使得我们可以用分段的方式来管理内存。

以后，根据编程时的需要，可将若干连续的内存单元看作一个段，用“段地址\*16”来定位段的起始地址（基地址），用偏移地址定位段中的内存单元。

注意：由于偏移地址是16位，对应2^16个存储单元（0~FFFFH），所以一个段的长度最大为64KB。

**3.CS和IP**

（1）段寄存器

8086CPU有4个段寄存器：CS（Code Segment）、DS（Data Segment）、SS（Stack Segment）、ES（Extra Segment）。

在8086CPU要访问内存时，由这4个段寄存器提供内存单元的段地址。

**（2）CS和IP**

**① CS（Code Segment）：代码段寄存器**

**功能：用于存储当前指令所在的段地址**

**② IP（Instruction Pointer）：指令指针寄存器**

**功能：用于存储当前指令在代码段中的偏移地址**

**性质：**

**<1>存放下一条指令：IP寄存器中，实际存储的是将要执行的下一条指令的偏移地址。**

**<2>自增性：每当CPU执行完一条指令后，IP寄存器中的值会自动递增，指向下一条指令。**

**③CS和IP的关系：**

**CS和IP共同指示了当前CPU要从内存中读取指令的地址。其中，CS存放当前指令所在的段地址；IP存放当前指令所在代码段中的偏移地址。**

**指令的逻辑地址：CS:IP**

**指令的物理递增：CS中的值\*16 + IP中的值**

（3）CPU执行汇编指令的过程

①从CS:IP指向的内存单元中读取指令，读取的指令进入指令缓冲器。

②IP=IP+所读取指令的长度，从而指向下一条指令。

③执行指令，并跳转回步骤①，重复这一过程。

**（4）修改CS和IP中的值**

修改CS:IP，即改变CPU将要执行的目标指令的地址。

注意：MOV指令不能修改CS和IP寄存器中的值。若想修改这两个寄存器中的值，需要JMP指令。

①同时修改CS:IP的值

语法：JMP 段地址:偏移地址

e.g. JMP 2AE3:3

功能：将CS的值变为2AE3，IP的值变为0003。（下一条将执行地址为2AE3:3的指令）

②只修改IP的值

语法：JMP某一合法寄存器

e.g. JMP AX

功能：将AX中的值赋给IP。（下一条将执行段内的偏移地址为AX中值的指令）