IBM-PC

第一章 基础知识

- 熟悉0-9的ASCII码

第二章 80x86计算机组织

2.1 80x86微处理器

名词术语：

- 主频：芯片所用的主时钟频率。（**运行速率**）
- 数据总线
  - 数据总线宽度
  - 外部数据总线宽度
  - 地址总线宽度（**存储器的最大范围**）
- 指令执行时间
- 工作方式：
  - (1)实模式:段寄存器保护实际的段地址,但只限于1M的寻址
  - (2)保护模式:有更大寻址能力,使用各种表格,段的实际地址放在描述符表中,使用段选择器
  - (3)虚拟模式:提供比主存更大的空间,程序可放在外存上,执行时调入,模拟多个 CPU

2.2 基于微处理器的计算机系统

2.2.1 硬件

组成：[p17 图2.1]

- 中央处理器（运算器+控制器）、存储器、输入/输出子系统（利用总线连接）

2.2.2 软件

计算机软件：

- **系统软件**：产生、准备和执行用户程序所必需的。（厂家提供）
- **用户软件**：用户自行编制的程序。

操作系统（**系统软件的核心**）：

> 系统指令的集合，主要部分是常住监督程序

- **I/O驱动程序**：对I/O设备进行控制或管理
- **文件管理程序**
- **文本编辑程序**： 如记事本、写字板
- **翻译程序**
  - **汇编程序**： Asm、Masm、Tasm
  - **解释程序**

    - **编译程序**
- **连接程序**：程序、库文件或其他已翻译的子程序连接在一起,形成机器可执行程序
- **装入程序**：把程序从外存送到内存调试
- **调试程序**

机器语言和汇编语言的特点：

- 汇编语言和机器语言一一对应
- 能够精确逼真地描述计算机执行步骤
- 可移植性差，复杂；需要了解计算机硬件系统

计算机软件层次图[p18 图2.2]

2.3 中央处理器

2.3.1 中央处理机CPU的组成

CPU的组成：

- 算数逻辑部件ALU (arithmetic logic unit)
- 控制逻辑
- 工作寄存器

2.3.2 8086/8088的寄存器组

- 寄存器：可见寄存器、不可见寄存器
- 可见寄存器：
    - 通用寄存器
    - 专用寄存器
    - 段寄存器

通用寄存器

数据寄存器（通用）：AX,BX,CX,DX (16bytes)

- 可用作8位寄存器用：AH, AL ... DH, DL等（将数据寄存器 一分为二）
- AX(accumulator): 累加器
    - 算数运算
- BX(base): 基址寄存器、通用寄存器
- CX(count): 计数器
- DX(data): 数据寄存器 / 双字长寄存器(存放高位数据)、通用寄存器
    - DX和AX组合变成双字长：DX高，AX低
- 变址寄存器（通用）：SP(堆栈指针寄存器), BP, SI, DI, 16位
    - SP:堆栈指针寄存器
    - BP:基址指针寄存器
    - SI:源变址寄存器
    - DI:目的变址寄存器

> SP用来指示栈顶的偏移地址；
>
> BP可作为堆栈区中的一个基地址，以便访问堆栈中的其他信息

- 专用寄存器: IP、SP、FLAGS, 16位

    - IP: 指令寄存器 总是指向下一条指令的首地址（不定长）
    - FLAGS: 程序状态字寄存器PSW 发生某些条件，在标志位保存结果
        - 条件码标志：
            - OF 溢出，
            - SF 符号，
            - ZF 零，
            - CF 机器数进位，
            - AF 半字节进位（十进制进位），
            - PF 奇偶
        - 控制标志位：DF 方向（串处理指令/矩阵运行是正向还是反向进行）
        - 系统标志：TF 每条指令产生中断，IF 中断，IOPL 权限
- 段寄存器
    - 用来直接或间接地存放段地址
    - 代码段 CS
    - 数据段 DS
    - 堆栈段 SS
    - 附加段 ES：存放数据；数据拷贝时，目标段地址存ES。

> 一般，段地址+0000+IP（端内偏移地址）；
>
> 拷贝数据的时候，目标数据和源数据一般处于两个段，DS和ES（目标）。

2.4 存储器

> 访问时，程序地址+段地址+基地址

2.4.1 存储单元的地址和内容

- 位，字节和字
    - 位：0 / 1
    - 字节：8bit = 1 字节
    - 字长：16位 = 2个字节，32位 = 4个字节，双字

2.4.2 实模式存储器寻址

- 存储器地址的分段
    - 分配给每个字节单元的编号，**从0开始，顺次加1**.
    - 每个字节单元有唯一一个地址：**物理地址**
    - 8086/8088：20位，地址范围：00000H~FFFFFH
    - 80386：32位
    - Pentium II： 36位
    - 存储单元的内容
        - 低位低地址，高位高地址。
        - 例：（0004）=1234H 中 12H存0005H地址，34H存0004H地址
        - 同一个地址既可看作**字节单元**，又可看作**字单元地址**
        - () 表示取内容，(()) ：地址的内容
        - 段：将存储器划分成段，每个段的大小可达64K，每段内地址可以用16位表示。
        - 小段：从0地址开始。
        - 物理地址 = 16d * 段地址 （ 相当于在末尾添加四个0 ） + 偏移地址
- 段寄存器

    - 段寄存器与放偏移地址的寄存器的默认组合[p30 表2.3]
        - CS + IP
        - SS + SP / BP
        - DS + BX / DI / SI 或16位数
        - ES + DI
- *保护模式
    - 逻辑地址
    - 描述符：描述端的大小、端在存储器中的位置及控制和状态信息
        - 基地址：段起始地址
        - 界限：段长度
        - 访问权：控制信息
        - 附加字段：表示该段的一些属性

2.5 外部设备（后面详解）

- 输入、输出设备
- 通信方式

第三章 80x86的指令系统和寻址方式

> 重点：
>
> 1. 数据传送指令、算数运算指令、逻辑运算和移位指令、控制转移指令
> 2. 能够根据不同的寻址方式掌握对于...

指令系统概述

格式：操作码（只有一个）+ 操作数（n个）

3.1 80x86的寻址方式

> **考试不会出现32位的指令**

3.1.1 与数据有关的寻址方式

- 8086/80286系列
    - 立即寻址
    - 寄存器寻址
    - 直接寻址
    - 寄存器间接寻址
    - 寄存器相对寻址
    - 基址变址寻址
    - 相对基址变址寻址
- 80386及后继机型
    - 比例变址寻址
    - 基址比例变址寻址
    - 相对基址比例变址寻址

立即寻址方式 immediate addressing

```
MOV AL 5  不限位数
MOV AX, 3064H   // (AX) = (AH , AL) = (30H, 64H) 32位不能用
MOV EAX, 12345678H  *32位可用
```

寄存器寻址方式 register addressing

- 方向：右 → 左
- 源寄存器和目的寄存器的位数，类型必须一致

```
// 设 (AX) = 3046H, (BX) = 1234H
MOV AX, BX
// 结果： AX = 1234H, (BH) = 1234H
```

有效地址计算公式

EA = 基址 + （变址 * 比例因子） + 位移量

- EA： efficient address
- 基址：数据段中数组或者字符串的首地址
- 变址：数组中的某个元素或者字符串中某个字符
- 比例因子：1, 2, 3 ,4

之后的寻址方式，都存在的首地址存放在DS中，在指令中指定的所有地址量，都是一个相对位移量。

直接寻址方式

```
// e.g (DS) = 3000H
MOV AX, [2000H]
// OUT: (AX) = 3050H

// 会根据VALUE类型判断 是立即还是直接寻址。
MOV AX, VALUE
MOV AX, [VALUE]

// 当VALUE在附加段中，应指定段跨越前缀：
MOV AX, ES:VALUE
MOV AX, ES:[VALUE]
```

寄存器间接寻址方式 register indirect addressing

```
// 模板：MOV AX [BX/BP/SI/DI]
// 区别首地址存放的寄存器：
// 物理地址 = DS + BX/SI/DI
// 物理地址 = SS + BP (堆栈段普通，一般用于找参数)

// (DS) = 2000H, (BX) =1000H
// 取 21000H 地址内存中数据
MOV AX,[BX]
```

寄存器相对寻址方式 register relative addressing

```
// EA = (BX/BP/SI/DI) + 8bit/ 16bit 位移量
// 物理地址 = (DS) + (BX/SI/DI) + 8bit/ 16bit 位移量
// 物理地址 = (SS) + (BP) + 8bit/ 16bit 位移量

MOV AX, COUNT[SI]
MOV AX, [COUNT + SI]
```

基址变址寻址方式 based indexed addressing

```
// 物理地址 = (DS) + (BX) + (SI/DI)
// 物理地址 = (SS) + (BP) + (SI/DI)

MOV AX, [BX][DI]
```

相对基址变址寻址方式 relative based indexed addressing

```
MOV AX, MASK[BX][SI]
```

3.1.2 与转移地址有关的寻址方式

> CS: 当前代码段的地址
> IP：当前偏移地址
> 条件转移指令：段内间接寻址的8位位移量
> JMP 和 CALL 指令：四种寻址方式的任意一种
> 段内转移：直接把求得的转向的有效地址 => IP，只改变IP
> 短剑转移：IP和CS都要修改

- 转移分类
  - 条件转移：段内转移
  - 无条件转移：段内/间转移

段内直接寻址 intrasegment direct addressing

```
// 当前 (IP) = 3000 H, NEW_ADDR = 3050H, 则偏移量 = 50H
JMP SHORT NEW_ADDR
```

- EA : 当前IP寄存器的内容 + 指令中指定的8bit/16bit位移量
- 条件转移/非条件转移
- 段内直接短转移：位移量8位 `JMP NEAR PTR PROGIA`
- 段内直接近跳转：位移量16位 `JMP SHORT QUEST`

段内间接寻址 intrasegment indirect addressing

```
JMP BX
JMP WORD PTR[BP + TABLE] // WORD PTR 为操作符，说明它是一个段内转移
```

- EA: 寄存器 / 存储单元 的内容
  - 用数据寻址方式任意一种（立即数除外）

段间直接寻址 intersegment direct addressing

```
JMP FAR PTR NEXTROUTINT
// NEXTROUTINT 转向的符号地址
// FAR PTR 段间转移操作符
```

- EA：指令中**直接**提供转向段地址，偏移地址

段间间接寻址 intersegment indirect addressing

```
JMP DWORD PTR [INTERS + BX]
// DWORD PTR 双字节操作符
// [INTERS + BX] 说明数据寻址方式为直接变址寻址方式
```

- EA：存储器中的两个相继字的内容取代IP和CS
    - 用数据寻址方式任意一种（立即数和寄存器直接除外）

3.2 程序占有的时间和执行时间

3.3 80x86的指令系统

分类

> 数据传送指令
> 算术指令
> 逻辑指令
> 串处理指令
> 控制转移指令
> 处理机控制指令

3.3.1 数据传送指令

> 功能：数据、地址、立即数 => 寄存器 / 存储单元

1. 通用数据传送指令

- MOV
- PUSH
- POP
- XCHG

```
1. MOV DST, SRC   // (DST) <- (SRC)

   MOV指令的7种格式 书 p48

* 注意：
   立即数不能直接送段寄存器
   目的寄存器 != 立即数/ CS寄存器
   双操作数指令不允许两个操作数都是用存储器
      MOV [SI], TABLE (ERROR!!)
      TABLE 变量 : 存储器地址; [SI] 把内容作为地址
```

```
2. PUSH SRC   // (SP) <- (SP)-2
              // ((SP)+1,(SP)) <- (SRC)

  SP: 堆栈栈顶指针 ；地址越小，越往上

3. PUSH SRC   // (DST) <- ((SP)+1,(SP))
              // (SP) <- (SP) + 2

* 注意：
    PUSH 和 POP 指令不影响标志位
    POP指令不能使用CS寄存器


4. XCHG OPR1, OPR2

* 注意：
 两个操作数必有一个在寄存器中。
 可以用除立即寻址以外的任何寻址方式。
 不影响标志位。
```

补充内容

```
// 默认指向第一个数据，可以对于地址进行加减来取数
DATA_BYTE DB 10, 4, 10H      // 地址+1：取得4，地址+2：取得10H
DATA_WORD DW 100, 100H, -5   // 地址+2：取得100H，地址+4：取得-5 双字
DATA_DW DD 3*20, 0FFFDH      // 地址+4：取0FFFDH
```

定义字符串

```
MESSAGE DB 'HELLO'
// DB 与 DW 定义单元存放相反
        DB 'AB' // 41: 'A', 42: 'B'
        DW 'AB' // 41: 'B', 42: 'A'
// 操作数 ? 空单元，保留存储单元但不存入数据
```

3. 地址传送指令

> 功能：主存按源地址的寻址方式计算出偏移地址，把地址存入REG
> 类比MOV：MOV存的是内容，LEA存的是地址

- LEA
- LDA,LES,LFS,LGS,LSS

```
1. LEA REG, SRC（重点）
- 目的操作数：16bit / 32bit 寄存器
- 源操作数：除立即数和寄存器外的任一一种寻址方式

* 注意：
指令不影响标志位。
LEA AX, ARR <==> MOV AX, OFFSET ARR //等价，其中OFFSET求出单个变量ARR的地址

2. LDS/... REG, SRC
把地址装入指定的寄存器。例：将下一字单元地址装入DS寄存器。
```

4. 标志寄存器传送指令（影响标志位）

- LAHF：标志送AH
- SAHF：AH送标志寄存器

- PUSHF/PUSHFD
- POPF/POPFD

格式：直接使用指令即可。
e.g LAHF //标志送AH

5. 类型转换指令

- CBW：字节转换为字指令
- CWD：自转和为双字指令

```
1. CBW // 低位：AL，高位：AH，即存在AX中。
符号：原最高位=0，新最高位=0
      原最高位=1，新最高位=0FFH
2. CWD // 低位：AX， 高位DX
```

3.3.2 算术指令

> 单操作数：不可以用立即数
> 双操作数：必须有一个操作数在寄存器中。

1.加法指令

- ADD
- ADC
- INC

```
1. ADD DST, SRC
操作： (DST) <- (SRC) + (DST)

// 带进位加法
2. ADC DST, SRC
操作： (DST) <- (SRC) + (DST) + CF

3. INC OPR
操作：(OPR) <- (OPR) +  1
```

3.3.3 逻辑指令

3.3.4 串处理指令

- MOVS 串传送

与REP配合： MOVS, STOS, LODS, INS, OUTS

```
'''
REP重复操作直到计数器寄存器Count Reg的内容为0为止。
地址长16位（一个字）时，CX 是 Count Reg

'''
格式：
1. REP string primitive

2. MOVS DST, SRC  //需要表明 -> 操作数是双字？字？字节？
   MOVSB (字节)
   MOVSW (字)

e.g. MOVS ES: BYTE PTR[DI], DS: [SI]

* 当方向标志 DF = 0时，用+，DF=1时，用-。
  当地址长度16位时，DST用DI(目的变址寄存器)，SRC用SI(源变址寄存器)
  MOVS指令，不影响条件码
```

MOVS指令：将数据段中的郑传数据传送到附加段

- 源串必须在数据段中，目的串必须在附加段中。
- 源串允许使用段跨越前缀来修改。
- 与REP联用，必须先把数据串的长度值送到计数寄存器Count Reg中

```
CLD (clear direction flag) // DF = 0
STD (set direction flag) // DF = 1
```

3.3.5 控制转移指令

*3.3.6 处理机控制指令

第四章 汇编语言程序格式

4.1 汇编程序功能

运行汇编语言程序的步骤：

- 用编辑程序建立ASM源文件
- MASM程序： ASM -> OBJ
- LINK程序： OBJ -> EXE
- DOS命令键入文件名执行程序

汇编程序的主要功能(p117)

4.2 伪操作

> 执行时期：汇编程序对源程序汇编期间 （由汇编程序处理）

4.2.1 处理器选择伪操作

`.8086` 选择8086指令系统

- 默认值为 `.8086` ， 一般放在整个程序最前面
- 放在程序中，例：在程序中用到一条80486指令，在该指令前放置 `.486`

4.2.2 段定义伪操作

1. 完整的段定义伪操作

段定义伪操作

```
segment name  SEGMENT
  ... # 数据段/附加段/堆栈段：存储单元的定义、分配
      # 代码段：指令、伪操作
segment name  ENDS


ASSUME assignment,...,assignment
```