# 计算机系统
# 汇编初步

**湖南大学**

**《计算机系统》课程教学组**

# 机器指令

**缺点**：用机器语言编写程序有很高的要求和许多不便。

机器指令是CPU能直接识别并执行的指令，它的表现形式是二进制编码。CPU只能执行机器指令。

**优点**：编写出来的程序执行效率高，CPU严格按照程序员的要求去做，没有多余的额外操作。

# 机器指令

**机器指令举例：**

00001001010101100110010000101011

01101000110101111110000110011000

……

```
00000000 <_start>:
   0:   90
   1:   b8 04 00 00 00
   6:   bb 01 00 00 00
   b:   b9 05 00 00 00
  10:   ba 0d 00 00 00
  15:   89 c3
  17:   b8 01 00 00 00
  1c:   bb 00 00 00 00
  21:   cd 80
```

- 为了改善机器指令的可读性，选用了一些能反映机器指令功能的单词或词组来代表该机器指令：MOV, ADD, SUB等等
- 不再关心机器指令的具体二进制编码。与此同时，也把CPU内部的各种资源符号化，使用该符号名也等于引用了该具体的物理资源，如EAX, ESP等等。

# 通用寄存器

80386有如下通用寄存器：

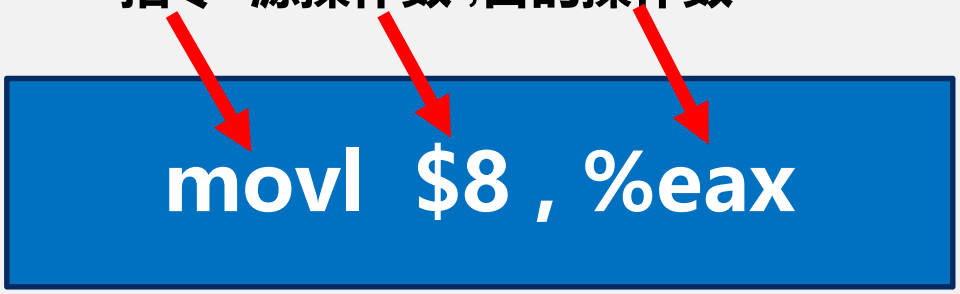
- AX,BX,CX,DX ；SI,DI ； SP,BP; （16位）
- AH,AL ; BH,BL; CH,CL; DH,DL（8位）
- EAX,EBX,ECX,EDX; ESI,EDI; ESP,EBP（32位）

段寄存器：DS，ES，SS ……

# AT&T
# 汇编代码

**格式：** 指令 源操作数 ,目的操作数

**示例：**




- 操作数
  - **立即数（immediate）**
  - **寄存器（register）**
  - **存储器（memory）**

**注意：本课程涉及的机器CPU计算总是仅从寄存器直接存/取数据**

# 汇编示例

## 1005.s

```
.section .text
  .global _start
_start:
     movl $4, %eax
     movl $1, %ebx
     movl $5, %ecx
     movl $13, %edx
     movl $1, %eax
     movl $0, %ebx
     int  $0x80
```

## 数据传送

`movl`：　用于传送32位的长字值

`movw`：　用于传送16位的字值

`movb`：　用于传送8位的字值

示例：1006.s

## 寻址方式

```
movl $1,%eax                    #立即数寻址 - 1005.s
movl %edx,%eax                  #寄存器寻址 - 1006.s

movl 0x08048054,%eax            #绝对寻址 - 1007.s
```

x/4bt 查看内存内容（b-单字节，h-双字节，w-四字 节，g-八字节）

x 按十六进制格式显示变量。d 按十进制格式显示变量。

u 按十六进制格式显示无符号整型。o 按八进制格式显示变量。

t 按二进制格式显示变量。a 按十六进制格式显示变量。

c 按字符格式显示变量。f 按浮点数格式显示变量

# 寻址方式

```
movl $1,%eax                    #立即数寻址 - 1005.s
movl %ebx,%eax                  #寄存器寻址 - 1006.s

movl 0x08048056, %ecx           #绝对寻址 - 1007.s
movw (%ecx), %ax                #间接寻址 - 1008.s
```

# 寻址方式

```
movl $1,%eax                    #立即数寻址 - 1005.s
movl %ebx,%eax                  #寄存器寻址 - 1006.s

movl 0x08048054,%eax            #绝对寻址 - 1007.s
movl (%ebx),%eax                #间接寻址 - 1008.s
movl 0x8(%ebx),%eax             #基址偏移量寻址 – 1008.s
```

# 寻址方式

```
movl  $1,%eax                  #立即数寻址 - 1005.s
movl  %ebx,%eax                #寄存器寻址 -  1006.s

movl  0x08048054,%eax          #绝对寻址 -  1007.s
movl  (%ebx),%eax              #间接寻址 -  1008.s
movl  0x8(%ebx),%eax           #基址偏移量寻址 – 1008.s
movl  (%ebx,%edx),%eax         #变址寻址 - 1009.s
```

# 寻址方式

```
movl  $1,%eax                  #立即数寻址 - 1005.s
movl  %ebx,%eax                #寄存器寻址 -  1006.s

movl  0x08048054,%eax          #绝对寻址 -  1007.s
movl  (%ebx),%eax              #间接寻址 -  1008.s
movl  0x8(%ebx),%eax           #基址偏移量寻址 – 1008.s
movl  (%ebx,%edx),%eax         #变址寻址 - 1009.s
movl  0x8(%ebx,%edx),%eax      #变址基址寻址 – 1009.s

movl  (%ebx,%ecx,0x2),%eax     #比例变址寻址 - 1010.s
```

# 寻址方式

```
movl  $1,%eax                    #立即数寻址 - 1005.s
movl  %ebx,%eax                  #寄存器寻址 -  1006.s

movl  0x08048054,%eax            #绝对寻址 -  1007.s
movl  (%ebx),%eax                #间接寻址 -  1008.s
movl  0x8(%ebx),%eax             #基址偏移量寻址 – 1008.s
movl  (%ebx,%edx),%eax           #变址寻址 - 1009.s
movl  0x8(%ebx,%edx),%eax        #变址基址寻址 – 1009.s

movl  (%ebx,%ecx,0x2),%eax       #比例变址寻址 - 1010.s
movl  0x8(%ebx,%ecx,0x2),%eax    #比例变址基址寻址  - 1010.s
```

## 数据传送

### 使用mov指令在内存和寄存器之间传送数据

**把数据值传送到内存 (1011.s)**

--增加了.section .data段，申明变量

--print &value，查看value变量在内存中的地址

--x/4bt 查看内存内容（b-单字节，h-双字节，w-四字 节，g-八字节）

# 数据传送

## 内存

| | |
|---|---|
| 0000 | Value1_1st Byte |
| 0001 | Value1_2nd Byte |
| 0010 | Value1_3rd Byte |
| 0011 | Value1_4th Byte |
| 0100 | Value2_1st Byte |
| 0101 | Value2_2nd Byte |
| 0110 | Value3 |
| 0111 | 00000000 |
| 1000 | 00000000 |
| 1001 | 00000000 |
| 1010 | 00000000 |
| 1011 | 00000000 |
| 1100 | 00000000 |
| 1101 | 00000000 |
| 1110 | 00000000 |
| 1111 | 00000000 |

## 数据传送

### 获取变量的地址

获得变量在内存中的地址并传送给寄存器 (1012.s)

两种方法

-- (1) movl $value1,%

-- (2) lea value1,%edi　　lea—Load Effective Address

# 下一节：汇编进阶

**湖南大学**

**《计算机系统》课程教学组**