# AT&T汇编基础

生成汇编指令，使用AT&T汇编

定义段空间

.section .data          有初值数据段

.section .bss             无初值数据段

.section .text           代码段

程序的开始和结束

.section .data

a:     .int   4               定义变量a值为4

.section .bss

         .lcomm    b,      2       定义变量b大小为2字节

.section .text

.globl                \_start                程序的开始

\_start:

         movl         $1,   %eax        程序的结束

         movl         $0,   %ebx

         int             $0x80

常量前要加$符号，寄存器前要加%

直接使用变量a，使用的是a的值；使用$a，使用的是a的地址

**常用寄存器**

32位通用寄存器：%eax, %ebx, %ecx, %edx

16位通用寄存器：%ax, %bx, %cx, %dx

8位通用寄存器：         %ah, %al, %bh, %bl, %ch, %cl, %dh, %dl

段寄存器：    cs, ds, ss, es, fs, gs（很少使用，不用修改）

段寄存器配对：

cs:eip(代码段指针) ds:esi(数据段), es:edi, ss:esp(栈顶指针), ss:ebp(栈底指针)

**数据类型**

.ascii                 字符串

.short                短整型 16位2字节

.int .long          长整型 32位4字节

.byte                 一个字节

.float                浮点单精度

.double             浮点双精度

.comm              名称,长度       长度为字节大小，定义一个为初始化的变量放在bss中

**寻址**

base\_address(offset\_address, index, size)

得到地址为base\_address + offset\_address + index \* size

例如 -4(%eax, %ebx, 2)         得到地址为 -4 + %eax + %ebx \* 2

其中base\_address可以为变量或者常量，offset\_address和index必须为寄存器

**常用指令**

|  |  |
| --- | --- |
| **movb** | b表示一个字节 |
| **movw** | w表示2字节short型整数 |
| **movl** | l表示4字节long型整数 |

|  |  |  |
| --- | --- | --- |
| **指令** | **指令格式** | **指令说明** |
| **movb, movw, movl** | movl 源，目的 | 源地址数据移动到目的地址 |
| **addb, addw, addl** | addl 源，目的 | 目的地址 = 目的地址 + 源地址 |
| **subb, subw, subl** | subl 源，目的 | 目的地址 = 目的地址 - 源地址 |
| **mulb, mulw, mull** | mull 源 | 8位，AX = AL \* 源 16位，DX:AX = AX \* 源 32位，EDX:EAX = EAX \* 源 |
| **divb, divw, divl** | divl 源 | 8位，AL = AX / 源      AH = AX % 源 16位，AX = DX:AX / 源   DX = DX:AX % 源 32位，EAX = EDX:EAX / 源 EDX = EDX:EAX % 源 |
| **cmp** | cmp 源，目的 | 源和目的比较，后面接jz、jnz、ja、jb、jae、jbe |
| **jz** | jz 目的代码 | 如果源和目的相等跳转 |
| **jnz** | jnz 目的代码 | 如果源和目的不相等跳转 |
| **ja** | ja 目的代码 | 目的 > 源 |
| **jb** | jb 目的代码 | 目的 < 源 |
| **jae** | jae 目的代码 | 目的 >= 源 |
| **jbe** | jbe 目的代码 | 目的 <= 源 |
| **pushw, pushl** | pushl 源 | 将源地址数据压栈 |
| **popw, popl** | popl 目的 | 将栈顶元素弹出到目的地址 |

**浮点数运算**

浮点数运算使用8个专用寄存器 st(0) ~ st(7)，形成一个栈，st(0)为栈顶, s(7)为栈底

|  |  |  |
| --- | --- | --- |
| **指令** | **指令格式** | **指令说明** |
| **finit** | finit | 启动浮点数运算器 |
| **flds, fldl** | fldl 源 | 将源地址浮点值压入栈顶，即st(0)，原来的st(0)移动到st(1)，依次执行  flds用于float，fldl用于double |
| **fstps, fstpl** | fstpl 目的 | 将栈顶元素弹出到目的地址，即st(0)弹出到目的地址，st(1)移动到st(0)，依次执行 |
| **fadd** | fadd %st(x), %st(0)  fadd %st(0), %st(x) | %st(0) = %st(0) + %st(x)  %st(x) = %st(x) - %st(0) |
| **fsub, fmul, fdiv** |  | 用法同fadd |