# 汇编语言指令格式

Intel ：寄存器和和立即数都没有前缀。

AT&T：寄存器前冠以 “％” ，而立即数前冠以“$”。

Intel ：十六进制和二进制立即数后缀分别冠以“h”和“b”

AT&T：十六进制立即数前冠以“0x”。



Intel ：第一个操作数是目的操作数，第二个操作数是源操作数。mov eax,[ecx]

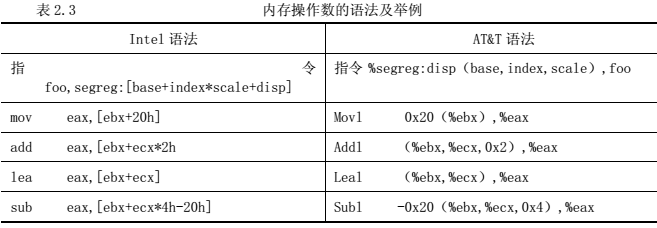
AT&T：第一个数是源操作数，第二个数是目的操作数。movl （%ecx）,%eax

Intel ：基寄存器用“ ［］ ”括起来。mov eax,[ebx+5]

AT&T：基寄存器用“ （） ”括起来。movl 5（%ebx）,%eax

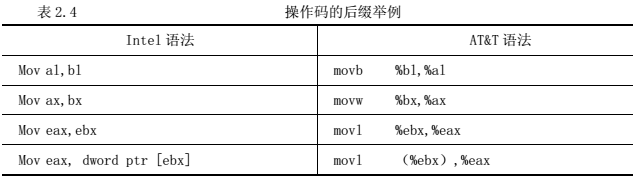
Intel： segreg:[base+index\*scale+disp]，

AT&T：%segreg:disp（base,index,scale） 。



AT&T： 在内存单元操作数的后面有一个后缀。 “l”=32 位， “w”=16 位， “b”=8 位。

Intel：在内存单元操作数的前面加上 byte ptr、word ptr 和 dword ptr，“dword”对应“long” 。



# 汇编程序指令

**.section**

section .data：这种节包含程序已初始化的数据。例如：

hello: .string "Hello world!\n"

hello\_len : .long 13

.section .bss：这个节包含程序还未初始化的数据。例如：

name : .fill 30 # 用来请求用户输入名字

name\_len : .long 0 # 名字的长度（尚未定义）

section .text ：这个节包含程序的代码，它是只读节，而.data 和.bss 是读／写节。

**.ascii**

.ascii：表示零个或多个（用逗号隔开）字符串，并把每个字符串中的字符放在连续的地址单元。（结尾不自动加“0“字节）

.ascii "string"...

**.asciz**

.asciz：z 代表“0” ，即每个字符串结尾自动加一个“0”字节，例如：

.asciz "Unknown interrupt\n"

**.fill repeat, size, value**

.fill ：以size 个字节为一组（不超过8字节），将这一组数据拷贝repeat次，每组数据的最高 4 个字节内容为 0，最低 4 字节内容置为 value。

size 和 value 为可选项。如果第 2 个逗号和 value 值不存在，则假定 value 为 0。如果第 1 个逗号和 size 不存在，则假定 size 为 1。

.fill NR\_CPUS\*4, 8, 0 /\* space for TSS's and LDT's \*/

//因为每个描述符正好占 8 个字节，因此，.fill 给每个 CPU 留有存放 4 个描述符的位置。

**.globl symbol**

.globl ：定义全局变量。

.globl SYMBOL\_NAME（idt）

.globl SYMBOL\_NAME（gdt）

**.quad bignums**

.quad 表示零个或多个 bignums（用逗号分隔），对于每个 bignum，其缺省值是 8 字节整

数。如果 bignum 超过 8 字节，则打印一个警告信息；并只取 bignum 最低 8 字节。

例如，对全局描述符表的填充就用到这个指令：

.quad 0x00cf9a000000ffff /\* 0x10 kernel 4GB code at 0x00000000 \*/

.quad 0x00cf92000000ffff /\* 0x18 kernel 4GB data at 0x00000000 \*/

.quad 0x00cffa000000ffff /\* 0x23 user 4GB code at 0x00000000 \*/

.quad 0x00cff2000000ffff /\* 0x2b user 4GB data at 0x00000000 \*/

**.rept count**

把.rept 指令与.endr 指令之间的行重复 count 次。

.rept 3

.long 0

.endr

相当于

.long 0

.long 0

.long 0

**.space size, value**

.space：保留 size 个字节的空间，每个字节的值为 value。如果逗号和value被省略，则假定 value为 0。

.space 1024 //表示保留 1024 字节的空间，并且每个字节的值为 0。

**.byte 表达式**

.byte 表示零或多个表达式（用逗号隔开） ，每个表达式被放在下一个字节单元。

**.word expressions**

.word：表示一个或多个表达式（用逗号分开），表达式的值占两个字节，例如：

gdt\_descr:

.word GDT\_ENTRIES\*8-1 //表示变量 gdt\_descr 的值为 GDT\_ENTRIES\*8-1

**.long expressions //这与.word 类似**

**.org new-lc , fill**

把当前节的位置计数器提前到 new-lc（New Location Counter） 。

.org 0x2000

ENTRY（pg0） //表示把位置计数器置为 0x2000，这个位置存放的就是临时页表 pg0。

# 汇编文件的编译

使用汇编程序 GAS 和连接程序 ld：

as filename.s -o filename.o

ld filename.o -o filename

使用 gcc：大写的 S 可以使 gcc 自动识别汇编程序中的 C 预处理命令。

gcc -o example example.S