和 | | 汇编语法的主要区别

October 29, 2011 | 分类：技术

作为一个爱折腾的大好青年，补番之余还要补一些 Linux 下的基础，比如 GDB 的正确使用方法。但无论是看 `gdb` 还是 `gcc -S` 里的汇编，感觉都不能一下子接受这种设定。

后来发现，虽然同为 x86 汇编，但语法也分两大流派：之前上学时学的 Intel 语法，以及流行于 Unix/Linux 平台上的 AT&T 语法。

首先，两者最让人纠结的区别就是源操作数、目标操作数的顺序。AT&T 语法先写源操作数，再写目标操作数；Intel 语法先写目标操作数，再写源操作数：

AT&T

```
movl %esp, %ebp
```

Intel

```
MOV EBP, ESP
```

然后，另一个明显的区别就是指令的命名（或者说，操作数大小的指定方式）。AT&T 语法将操作数的大小表示在指令的后缀中（b、w、l）；Intel 语法将操作数的大小表示在操作数的前缀中（BYTE PTR、WORD PTR、DWORD PTR）：

AT&T

```
decw (%eax)
```

Intel

```
DEC WORD PTR [EBX]
```

再者，各种取址方式的表示。AT&T 语法总体上是 `offset(base, index, width)` 的格式；Intel 语法总体上是 `[INDEX * WIDTH + BASE + OFFSET]` 的格式：

AT&T

```
movl           0x0100, %eax
movl           (%esi), %eax
movl         -8(%ebp), %eax
```

Intel

```
MOV EAX, [0100]
MOV EAX, [ESI]
MOV EAX, [EBP-8]
MOV EAX, [EBX*4+0100]
MOV EAX, [EDX+EBX*4+8]
```

另外，各种非十进制数制下数字的表示方法。AT&T 语法用前缀表示数制（ox、o、ob）；Intel 语法用后缀表示数制（h、o、b）：

AT&T

```
movl 0x8   , %eax
movl 010   , %eax
movl 0b1000, %eax
```

Intel

```
MOV EAX,    8h
MOV EAX,   10o
MOV EAX, 1000b
```

最后就是零碎的东西的表示方法了。AT&T 语法要在常数前加 $、在寄存器名前加 % 符号；Intel 语法没有相应的东西要加：

AT&T

```
subl $0x30, %eax
```

Intel

```
SUB EAX, 30
```

于是，以上就是 AT&T 和 Intel 汇编语法的主要区别了……吧？

汇编

添加新评论 »

Name 称呼 *