x86汇编(AT&T)

基础语法

1. 寄存器使用 `%` 前缀，立即数使用 `$` 前缀，十六进制前 `0x`
2. 操作码后缀 `l` 代表32bit， `w` 代表16bit， `b` 代表8bit
3. 指令格式以及常见指令:

```
opcode src, dest

//运算指令
addl %edx, %eax    //edx+eax，结果放到eax l表示32位，后面指令位64位
add  $5,%r10      // 5 + r10，结果放到r10
div  %r10         //rax 除以r10，商放到rax，余数放到rdx
inc  %r10         // r10 加1
mul  %r10         // 将rax乘以 r10, 将结果放到rax中，溢出部分放到rdx

//拷贝指令
mov %r10,%r11    //将r10寄存器的值赋值给r11 ;
mov $99,%r10      //将立即数99赋值给r10寄存器;
mov %r10,(%r11)  // 将r10的值拷贝到r11寄存器中的数值指向的内存地址上;
mov (%r10),%r11   // 将r10中数值指向的内存地址上的内容拷贝到r11;
push %r10         // 将r10的值放到栈上 ;
pop %r10          // 将栈顶的值pop到r10寄存器上。

```

3. 寄存器修饰:

    1. `r-` 表示 64 位的寄存器，例如 `%rax`
    2. `e-` 表示 32 位的寄存器，例如 `%eax`

3. 没有，表示 16 位的寄存器，例如 `%ax`
4. x 换成 l 表示低位。例如 `%al` 表示 `%ax` 的低 8 位。
5. x 换成 h 表示高位。例如 `%ah` 表示 `%ax` 的高 8 位。

4. 常见寄存器(以64bit为例):

寄存器	使用限制	使用说明
rax	否	1，系统调用时，调用号； 2，函数返回值； 3，除法运算中，存放除数、以及运算结果的商； 4，乘法运算中，存放被乘数、以及运算结果；
rbx	是，被调用者保存	1，在32位模式下，用来存放GOT的地址；
rcx	否	1，函数调用时，第4个参数 2，有时用作counter；
rdx	否	1，函数调用时，第3个参数； 2，除法运算中，存放运算结果的余数； 3，乘法运算中，存放运算结果溢出的部分；
rbp	**是，被调用者保存**	**frame pointer，存放当前函数调用时栈的基地址**
rsp	**是，被调用者保存**	**时时刻刻指向栈顶**
rdi	否	1，函数调用时，第1个参数；
		2， rep movsb中的目的寄存器；
rsi	否	1，函数调用时，第2个参数；
		2， rep movsb中的源寄存器；
r8	否	1，函数调用时，第5个参数
r9	否	1，函数调用时，第6个参数
r10	否	
r11	否	
r12	是，被调用者保存	
r13	是，被调用者保存	
r14	是，被调用者保存	
r15	是，被调用者保存	

> eax, ebx, ecx, edx, esi, edi, ebp, esp等都是X86 汇编语言中CPU上的通用寄存器的名称，是32位的寄存器。如果用C语言来解释，可以把这些寄存器当作变量看待。
>
> 这些32位寄存器有多种用途，但每一个都有“专长”，有各自的特别之处。
>
> eax 是"累加器"(accumulator), 它是很多加法乘法指令的缺省寄存器。
>
> ebx 是"基地址"(base)寄存器, 在内存寻址时存放基地址。
>
> ecx 是计数器(counter), 是重复(REP)前缀指令和LOOP指令的内定计数器。
>
> edx 则总是被用来放整数除法产生的余数。
>
> esi, edi, 分别叫做"源/目标索引寄存器"(source/destination index),因为在很多字符串操作指令中, DS:ESI指向源串,而ES:EDI指向目标串.
>
> ebp是"基址指针"(BASE POINTER), 它最经常被用作高级语言函数调用的"框架指针"(frame pointer)

reference

1. AT&T的简要概括
2. x86 Assembly Guide