# x86-64汇编速成

## 简介

我们将在此篇文档中只提到一部分很小的x86-64汇编。

## 环境

x86-64下，共有%rax、%rbx、%rcx、%rdx、%rsi、%rdi、%rsp（栈顶指针）、%rbp（栈底指针）、%r8、%r9、%r10、%r11、%r12、%r13、%r14、%r15共16个64位通用寄存器。也包括%xmm0、%xmm1、%xmm2、%xmm3、%xmm4、%xmm5、%xmm6、%xmm7至少8个用于存储浮点数的浮点数寄存器。需要强调的是，浮点数和整数在计算机内的存储方式不同，所以对于浮点数有一套自己的寄存器以及运算指令。

## 指令集

见表

|  |  |
| --- | --- |
| 指令 | 说明 |
| movq %rax, %rbx | 将%rax寄存器的数据移动到%rbx |
| movq -8(%rsp), %rbx | 将%rsp-8所代表的的内存地址处的数据移动到%rbx |
| movq %rbx, -8(%rsp) | 将%rbx的数据移动到%rsp-8所代表的的内存地址处 |
| 注意，数据传送只支持寄存器-寄存器，以及寄存器-内存，但不支持内存-内存形式 | |
| movsd -8(%rsp), %xmm0 | 将%rsp-8所代表的的内存地址处的64位数据移动到浮点数寄存器%xmm0 |
| movsd %xmm0, -8(%rsp) | 将%xmm0浮点数寄存器的值移动到%rsp-8所代表的的内存地址处 |
| call func1 | 调用func1处标记的函数 |
| ret | 返回 |
| addq %rax, %rbx | %rbx = %rbx + %rax |
| subq %rax, %rbx | %rbx = %rbx - %rax |
| idivq %rcx | 用[%rdx:%rax]构成的128位整数除以%rcx，并且把商存在%rax，把余数存在%rdx中 |
| cqto | 一般与idivq连用，将%rax中的值拓展前64位到%rdx |
| imulq %rax, %rbx | %rbx = %rbx \* %rax |
| andq %rax, %rbx | %rbx = %rbx 按位与 %rax |
| orq %rax, %rbx | %rbx = %rbx 按位或 %rax |
| notq %rax | %rax按位取反 |
| xorq %rax, %rbx | %rbx = %rbx 按位异或 %rax |
| jmp .tag | 跳到.tag标记处 |
| cmpq %rax, %rbx | 比较%rax和%rbx，一般后面直接接条件跳转语句，例如je |
| testq %rax, %rax | 测试%rax，通常下句用jz或者jnz，连起来表示如果%rax为0/非0则跳转到 |
| jz .tag | 如果为0则跳转到.tag |
| jl/jle/je/jne/jg/jge .tag | （整型）小于、小于等于、等于、不等于、大于、大于等于则跳转到.tag，l即less，g即greater |
| jb/jbe/ja/jae/jp .tag | （浮点数）小于、小于等于、大于、大于等于、等于则跳转到.tag，b即below，a即above |
| addsd %xmm0, %xmm1 | 浮点数%xmm1 = %xmm1 + %xmm0 |
| subsd %xmm0, %xmm1 | 浮点数%xmm1 = %xmm1 - %xmm0 |
| mulsd %xmm0, %xmm1 | 浮点数%xmm1 = %xmm1 \* %xmm0 |
| divsd %xmm0, %xmm1 | 浮点数%xmm1 = %xmm1 / %xmm0 |
| pushq %rax | %rax入栈，且栈顶延伸 |
| popq %rax | 栈顶元素弹出到%rax，栈顶缩减 |

特别强调，整数除法的一般格式为，例如被除数在-8(%rsp)内存位置处，除数在-16(%rsp)处，则为

movq -8(%rsp), %rax

cqto

movq -16(%rsp), %rcx

idivq %rcx

之后余数在%rdx，商在%rax。

另外，我们所使用到的浮点数和整数指令，都是针对数据宽度为64位。

## 跳转标记

跳转格式

.tag1:

movq %rax, %rbx

…语句

jmp .tag1

则当执行到jmp .tag1时，将跳转到 movq %rax, %rbx语句。

## 后记

由于x86-64汇编本身是一个非常庞大的内容，有兴趣的同学可以参阅《深入理解计算机系统》的第二部分，该部分讲述了汇编运行的一般原理，针对x86-64，有x86-64汇编手册可以查阅。