x64 ASM 常用汇编指令

语法习惯

这里主要说AT&T风格的汇编语言风格。因为gdb看反汇编默认的风格就是AT&T风格的，Intel风格的这里就不做介绍。

立即数，$ 开头

寄存器，% 开头

取地址里面的值，偏移量(%寄存器)

// 除了 lea 取地址指令外，lea就是取地址

load effecive address

整形操作通用后缀，后缀 [b w l q 1 2 4 8] byte word l… quadruple， 表示多少字节

浮点指令还有三个 [s d t 4 8 16] single double extended (修饰精度: precision)

S -> Scr, D -> Dst, I -> Immediate, R -> Register

//表示源，目的操作数，立即数，寄存器

常用指令

数据传送

mov S, D D <- S

movabs I, R D <- S

// I 表示 immediate date，立即数 常数

// S 表示 Src，立即数，内存，寄存器 都可以

// D 表示 Dst，立即数，内存，寄存器 都可以

// R 表示 Register，寄存器，如下面表示指令的目的操作数只能是寄存器

movslq S, R // move sign-extend double world (eg, int 转 long 赋值)

movsbq S, R // move sign-extend byte

movzbq S, R // move zero-extend byte

栈帧指令

// 单操作数指令

// 这里可以这里理解 S D, Src 读操作，Dst 写操作

push S 常用来存 frame pointer 调用栈的一层函数栈帧

push S, 完成这些操作

%rsp = %rsp - 8, 压栈导致线程执行调用栈增长，栈向低地址增长，x64一个地址 8 个 Byte

%rsp = S, S 放到 %rsp 指向的栈顶地址

pop D

pop D, 完成这些操作

D = %rsp, 当前存在%rsp指向的 stack pointer地址中的值赋给 D %rsp = %rsp + 8, 弹栈导致线程执行调用栈减小，栈向高地址回缩

leave

ret

操作地址

// x64 汇编，一个地址 8 字节， 所以后缀为 q （quad word） quadruple

// word 由于历史遗留表示 2 个 Byte，后缀 w

leaq S, D (src, dest) D <- &S

单目运算

incq D D <- D + 1 // 比如指针加加

decq D D <- D - 1

negq D D <- -D

notq D D <- ~D

双目运算

addq S, D D = D + S

subq S, D D = D - S

imulq S, D D = D * S

xorq S, D D = D ^ S

orq S, D D = D | S

andq S, D D = D & S

移位指令

salq k, D D = D << k (left shift)

shlq k, D D = D << k (== salq)

sarq k, D D = D >> k (right shift)

shrq k, D D = D >> k (logical right shift)

控制指令

// 下面是判断指令，配合置标志位和条件跳转实现流程控制

cmpq S2, S1 S1 - S2

testq S2, S1 S1 & S2

// 常用于测试操作数中某位是否为1，而且不会影响目的操作数

置标志位

CF：进位标志寄存器，它记录无符号操作的溢出，当溢出时会被设为1。

ZF：零标志寄存器，当计算结果为0时将会被设为1。

SF：符号标志寄存器，当计算结果为负数时会被设为1。

OF：溢出标志寄存器，当计算结果导致了补码溢出时，会被设为1。

条件赋值

// 双目指令 D S，配合上面满足条件的 D 赋值给 S , 对应条件运算符

cmove equal

cmovne

cmovs Negative

cmovns

cmovg greater (signed)

cmovge

cmovl less

cmovle

// g greater 这么判断 ~(SF^OF)&~ZF（有符号的大于）

// 还是判断 a - b 值，^ xor 异或 => 同0异1

// l less 有符号小于，判断 SF^OF，less 判断 负溢出为正

// 举例 less 负溢出举例

(lldb) p/d (int)-2147483648

(int) $6 = -2147483648

(lldb) p/d (int)-2147483648 - 2

(int) $7 = 2147483646

// 举例 below, CF是无符号溢出标志, a - b 结果溢出了，则代表 a 是小于 b

cmova (above) (a : unsigned)

cmovae

cmovb (below)

cmovbe

条件跳转

jmp Label

jmp *Operand

je == 0

jne

js Negative

jns

jg greter (signed)

jge

jl less

jle

ja above (unsigned)

jae

jb below

jbe

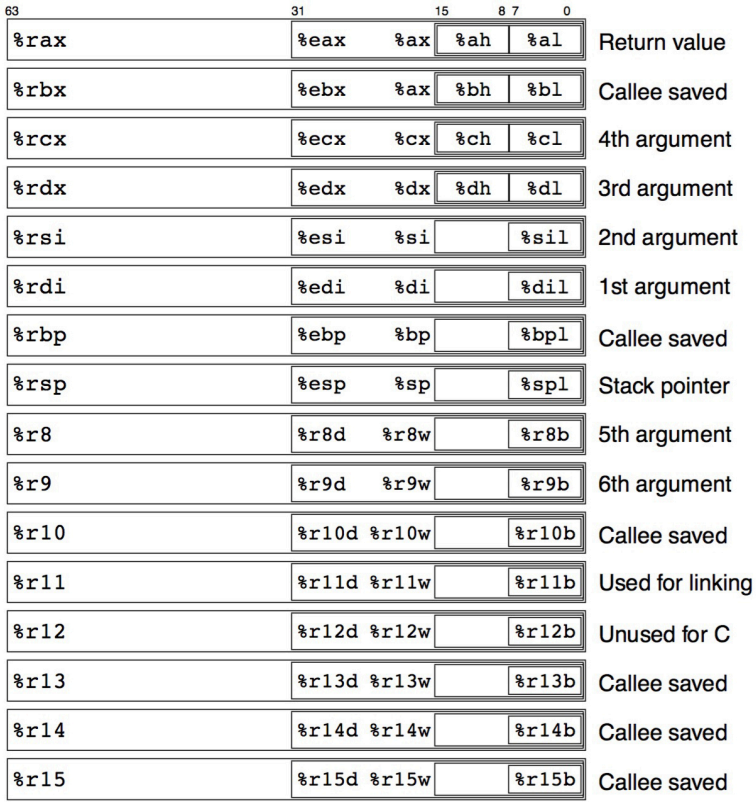
浮点操作

Floating instruction

具体看asm64-handout.pdf的介绍。

寄存器

register



C 数据类型

c_data

C declaration	Intel data type	GAS suffix	x86-64 Size (Bytes)
char	Byte	b	1
short	Word	w	2
int	Double word	l	4
unsigned	Double word	l	4
long int	Quad word	q	8
unsigned long	Quad word	q	8
char *	Quad word	q	8
float	Single precision	s	4
double	Double precision	d	8
long double	Extended precision	t	16

参考文献

asm64-handout.pdf

CSAPP

CSAPP read note