15-213/18-213 Exam Notes (Spring 2013)

Jumps

Jump	Condition
`jmp`	1
`je`	ZF
`jne`	~ZF
`js`	SF
`jns`	~SF
`jg`	~(SF^OF)&~ZF
`jge`	~(SF^OF)
`jl`	(SF^OF)
`jle`	(SF^OF)|ZF
`ja`	~CF&~ZF
`jb`	CF

Arithmetic Operations

Format		Computation
`addl`	*Src,Dest*	Dest = Dest + Src
`subl`	*Src,Dest*	Dest = Dest - Src
`imull`	*Src,Dest*	Dest = Dest * Src
`sall`	*Src,Dest*	Dest = Dest << Src
`sarl`	*Src,Dest*	Dest = Dest >> Src
`shrl`	*Src,Dest*	Dest = Dest >> Src
`xorl`	*Src,Dest*	Dest = Dest ^ Src
`adnl`	*Src,Dest*	Dest = Dest & Src
`orl`	*Src,Dest*	Dest = Dest | Src

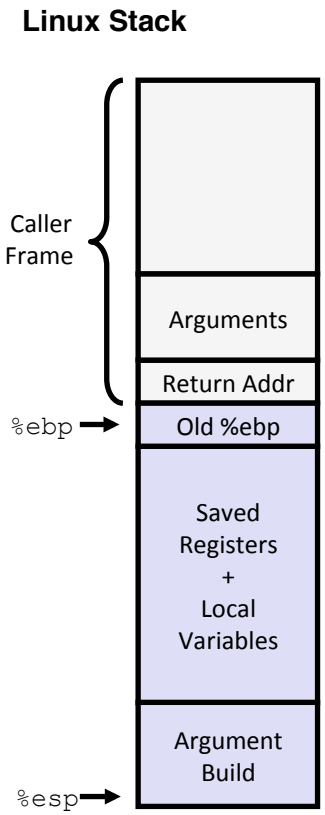
Memory Operations

Format	Computation
`(Rb, Ri)`	Mem[Reg[Rb]+Reg[Ri]]
`D(Rb,Ri)`	Mem[Reb[Rb]+Reg[Ri]+D]
`(Rb,Ri,S)`	Mem[Reg[Rb]+S*Reg[Ri]]

Registers

63	31	15	8 7	0	
`%rax`	`%eax`	`%ax`	`%ah`	`%al`	Return value
`%rbx`	`%ebx`	`%bx`	`%bh`	`%bl`	Callee saved
`%rcx`	`%ecx`	`%cx`	`%ch`	`%cl`	Argument #4
`%rdx`	`%edx`	`%dx`	`%dh`	`%dl`	Argument #3
`%rsi`	`%esi`	`%si`		`%sil`	Argument #2
`%rdi`	`%edi`	`%di`		`%dil`	Argument #1
`%rbp`	`%ebp`	`%bp`		`%bpl`	Callee saved
`%rsp`	`%esp`	`%sp`		`%spl`	Stack Pointer
`%r8`	`%r8d`	`%r8w`		`%r8b`	Argument #5
`%r9`	`%r9d`	`%r9w`		`%r9b`	Argument #6
`%r10`	`%r10d`	`%r10w`		`%r10b`	Reserved
`%r11`	`%r11d`	`%r11w`		`%r11b`	Used for linking
`%r12`	`%r12d`	`%r12w`		`%r12b`	Callee saved
`%r13`	`%r13d`	`%r13w`		`%r13b`	Callee saved
`%r14`	`%r14d`	`%r14w`		`%r14b`	Callee saved
`%r15`	`%r15d`	`%r15w`		`%r15b`	Callee saved

Linux Stack



Specific Cases of Alignment (IA32)

1 byte: char, …

no restrictions on address

2 bytes: short, …

lowest 1 bit of address must be 0_2

4 bytes: int, float, char *, …

lowest 2 bits of address must be 00_2

8 bytes: double, …

Windows (and most other OS's & instruction sets):

lowest 3 bits of address must be 000_2

Linux:

lowest 2 bits of address must be 00_2

i.e., treated the same as a 4-byte primitive data type

12 bytes: long double

Windows, Linux:

lowest 2 bits of address must be 00_2

i.e., treated the same as a 4-byte primitive data type

C Data Type	Intel IA32	x86-64
char	1	1
short	2	2
int	4	4
long	4	8
long long	8	8
float	4	4
double	8	8
long double	10/12	10/16
pointer	4	8

Specific Cases of Alignment (x86-64)

1 byte: char, …

no restrictions on address

2 bytes: short, …

lowest 1 bit of address must be 0_2

4 bytes: int, float, …

lowest 2 bits of address must be 00_2

8 bytes: double, char *, …

Windows & Linux:

lowest 3 bits of address must be 000_2

16 bytes: long double

Linux:

lowest 3 bits of address must be 000_2

i.e., treated the same as a 8-byte primitive data type

Byte Ordering

4-byte variable 0x01234567 at 0x100

Big Endian

Least significant byte has highest address

0x100	0x101	0x102	0x103
01	23	45	67

Little Endian

Least significant byte has lowest address

0x100	0x101	0x102	0x103
67	45	23	01

Floating Point

Bias = $2^{k-1} - 1$