IA32 gnu Cheat Sheet

Integer Registers (4 byte)

%eip	instruction pointer
%esp	stack pointer
%ebp	base pointer for stack frame
%eax	return values; low of pair for mul/div with %edx; intermediate
%edx	high of pair for mul/div pair with %eax; intermediate
%ecx	intermediate
%ebx	base address for arrays; callee save
%esi	source index; callee save
%edi	destination index; callee save

Constants (immediate)

$*constant* Numeric constants can be base-10 or hexadecimal (begin with 0x).

Memory References

symbol	global basis, external or static variable
$symbol	address of the variable
(%*reg*)	address in %*reg* register
off(%reg)	address = offset + (%*reg*)
(%*reg1*,%*reg2*)	address= (%*reg1*)+(%*reg2*)
off(%*reg1*,%*reg2*)	address=offset+(%*reg1*)+(%*reg2*)
off(%*reg1*,%*reg2*,*sh*)	address=offset+(%*reg1*)+(%*reg2*)**sh* where *sh* is one of 1,2,4,8

Move

mov*S source, dest*	move value from *source* to *dest*
lea*S source, dest*	move address of *source* to *dest*

Arithmetic

add*S op1, op2*	add op1 to op2. Sets OF, ZF, SF.
sub*S op1, op2*	subtract op1 from op2. Sets OF, ZF, SF.
imul*S operand**	multiply operand using %eax %edx pair
idiv*S operand**	divide by operand; result int %eax; rem in %edx
inc*S operand*	increment the operand
dec*S operand*	decrement the operand
neg*S operand*	negate the operand

Shift

sal*S k,reg*	shift arithmetic left *k* bits
sar*S k,reg*	shift arithmetic right *k* bits
shl*S k,reg*	shift logical left *k* bits
shr*S k,reg*	shift logical right *k* bits

Logic

and*S op1, op2*	bitwise p1 & op2; result in op2
or*S op1, op2*	bitwise op1 | op2; result in op2
xor*S op1, op2*	bitwise op1 ^ op2; result in op2
not*S operand*	~ operand; result in operand

Flow

cmp*S op1, op2*	compare operand2:operand1. Sets ZF, SF, OF, and CF.
test*S op1, op2*	this ands the operands. Sets SF, ZF. Usually tests a register with itself to determine if value is negative, zero or positive.
jmp *label*	jump unconditionally to *label*
je/jz *label*	jump equal (ZF on); jump if zero
jne/jnz *label*	jump not equal (ZF off); jump if not zero
Signed Comparisons:	
jle *label*	jump less than or equal ((SF xor OF) or ZF)
jl *label*	jump less than (SF xor OF)
jge *label*	jump greater than or equal (!(SF xor OF) and !ZF)
jg *label*	jump greater than (! (SF xor OF))
Unsigned Comparisons:	
jbe *label*	jump before or equal (ZF on or CF on)
jb *label*	jump before (CF on)
jae *label*	jump after or equal (CF off)
ja *label*	jump after (ZF off and CF off)

Stack

push*S operand*	pushes the operand onto the runtime memory stack
pop*S operand*	pops the top of the stack and stores it in *operand*
leave	prepare to leave the subroutine based on calling convention
call *dest*	using calling convention to invoke the function at *dest*
ret	return to the caller based on the calling convention

Data Size Suffixes

S is the size and must be one of

b	byte (1 byte)
w	word (2 bytes) - based on old hardware where word size was 2 bytes
l	long (4 bytes)
q	quad words (8 bytes)

Condition Flags

OF overflow flag; set when a **signed** arithmetic operation is either too large or too small to fit in the destination**;** set when operands have same sign and sign changes

CF carry flag; set when an **unsigned** arithmetic operation is too large to fit in the destination

ZF zero flag; set when the result is zero; it is ON if a comparison shows values are equal

SF sign flag; set when the result is a negative value

PF parity flag; its parity is even (PE) when an even number of 1 bits in the 8 low order bits.