Intel x86 Assembly Language Cheat Sheet

Instruction	Effect	Example
Data movement		
mov src, dest	Copy src to dest	mov $10,%eax
Arithmetic		
add src, dest	Dest = dest + src	add $10, %esi
mul reg	edx:eax = eax * reg (colon means the result spans across two registers)	mul %esi

div reg
idiv reg
	edx = edx:eax mod reg
	eax = edx:eax / reg
		div %edi

inc dest	Increment destination	Inc %eax
dec dest	Decrement destination	dec (%esi)

sbb arg1, arg2	If CF = 1, (this is set by cmp instruction; refer cmp)
	arg2 = arg2 – (arg1 + 1)
	else
	arg2 = arg2 – arg1
		sbb %eax, %ebx

Function Calls		
call label	Push eip, transfer control	call _fib
ret	Pop eip and return	ret
push item	Push item (constant or register) to stack	pushl $32 pushl %eax
pop [reg]	Pop item from stack; optionally store to register	pop %eax popl
Bitwise Operations		
and src,dest	Dest = src & dest	and %ebx, %eax
or src, dest	Dest = src | dest	orl (0x2000), %eax
xor src, dest	Dest = src ^ dest	xor $0xffffff, %eax
shl count, dest	Dest = dest << count	shl $2, %eax
shr count, dest	Dest = dest >> count	shr $4, (%eax)
sal count, dest	Same as shl, shifted bits will be the sign bit	
Conditionals and jumps		
cmp arg1, arg2	If arg1 > arg2 sets CF=1 (carry flag =1) This compares arg1 and arg2; you can use any conditionals jumps below to act upon the result of this comparison	cmp $0, %eax
test reg,imm/reg	Bitwise and of register and constant/register; the next jump command uses the result of this; consider this essentially as same as compare	test %rax, %rcx
je label	Jump to label if arg2 = arg1	je endloop
jne label	Jump to label if arg2 != arg1	jne loopstart
jg label / ja label	Jump to label if arg2 > arg1	jg exit / ja exit
jge label	Jump to label if arg2 >= arg1	jge format_disk
jl label	Jump to label if arg2 < arg1	jl error
jle label	Jump to label if arg2 <= arg1	jle finish
jz label	Jump to label if bits were not set	jz looparound
jnz label	Jump to label if bits were set	jnz error
jump label	Unconditional jump	jmp exit
Miscellaneous		
nop	No-op	nop
lea addr, dest	Move the address calculated to the dest	lea 23(%eax, %ecx,8),%eax
cqto	%rdx:%rax← sign-extend of %rax.	cqto

suffixes b=byte(8), w=word(16), l=long(32), q=quad(64)

base indexed scale displacement 172(%rdi, %rdx,8) = %rdi + 8 * %rdx + 172

Note that not both src and dest can be memory operands at the same time.

register - %eax	fixed address – (0x1000)
constant - $10	dynamic address – (%rsi)