Cheatography

x86 Assembly, C Linking, Loading etc Cheat Sheet

by rwwagner90 via cheatography.com/2239/cs/635/

x86 Assembly Instructions

ADD <dest>, <source>	Adds <source> to <dest>. <dest> may be a register or memory. <source> may Be a register, memory or immediate value.
CALL <loc>	Call a function and return to the next instruction when finished. <proc> may be a relative offset from the current location, a register or memory addr.
CMP <dest>, <source>	Compare <source> with <dest>. Similar to SUB instruction but does not Modify the <dest> operand with the result of the subtraction.
DEC <dest>	Subtract 1 from <dest>. <dest> may be a register or memory.
DIV <divisor>	Divide the EDX:EAX registers (64-bit combo) by <divisor>. <divisor> may be a register or memory.
INC <dest>	Add 1 to <dest>. <dest> may be a register or memory.
JE <loc>	Jump if Equal (ZF=1) to <loc>.
JG <loc>	Jump if Greater (ZF=0 and SF=OF) to <loc>.
JGE <loc>	Jump if Greater or Equal (SF=OF) to <loc>.
JLE <loc>	Jump is Less or Equal (SF<>OF) to <loc>.
JMP <loc>	Jump to <loc>. Unconditional.
JNE <loc>	Jump if Not Equal (ZF=0) to <loc>.
JNZ <loc>	Jump if Not Zero (ZF=0) to <loc>.
JZ <loc>	Jump if Zero (ZF=1) to <loc>.
LEA <dest>, <source>	Load Effective Address. Gets a pointer to the memory expression <source> and stores it in <dest>.

x86 Assembly Instructions (cont)

MOV <dest>, <source>	Move data from <source> to <dest>. <source> may be an immediate value, register, or a memory address. Dest may be either a memory address or a register. Both <source> and <dest> may not be memory addresses.
MOVZBL <dest>, <source>	Zero extend <source> to long and save in <dest>.
MUL <source>	Multiply the EDX:EAX registers (64-bit combo) by <source>. <source> may be a register or memory.
POP <dest>	Take a 32-bit value from the stack and store it in <dest>. ESP is incremented by 4. <dest> may be a register, including segment registers, or memory.
PUSH <value>	Adds a 32-bit value to the top of the stack. Decrements ESP by 4. <value> may be a register, segment register, memory or immediate value.
ROL <dest>, <count>	Bitwise Rotate Left the value in <dest> by <count> bits. <dest> may be a register or memory address. <count> may be immediate or CL register.
ROR <dest>, <count>	Bitwise Rotate Right the value in <dest> by <count> bits. <dest> may be a register or memory address. <count> may be immediate or CL register.
SHL <dest>, <count>	Bitwise Shift Left the value in <dest> by <count> bits. Zero bits added to the least significant bits. <dest> may be reg. or mem. <count> is imm. or CL.
SHR <dest>, <count>	Bitwise Shift Right the value in <dest> by <count> bits. Zero bits added to the least significant bits. <dest> may be reg. or mem. <count> is imm. or CL.

x86 Assembly Instructions (cont)

SUB <dest>, <source>	Subtract <source> from <dest>. <source> may be immediate, memory or a register. <dest> may be memory or a register. (source = dest)->ZF=1, (source > dest)->CF=1, (source < dest)->CF=0 and ZF=0
TEST <dest>, <source>	Performs a logical OR operation but does not modify the value in the <dest> operand. (source = dest)->ZF=1, (source <> dest)->ZF=0.
XCHG <dest, <source>	Exchange the contents of <source> and <dest>. Operands may be register or memory. Both operands may not be memory.
XOR <dest>, <source>	Bitwise XOR the value in <source> with the value in <dest>, storing the result in <dest>. <dest> may be reg or mem and <source> may be reg, mem or imm.

What does a Linker do?

– Merges multiple relocatable (.o) object files into a single executable object file that can loaded and executed by the loader.

– As part of the merging process, resolves external references. • External reference: reference to a symbol defined in another object file.

– Relocates symbols from their relative locations in the .o files to new absolute positions in the executable.

– Updates all references to these symbols to reflect their new positions. • References can be in either code or data

Memory Management

Info