**X86 Assembly**

(theo linkt<http://en.wikibooks.org/wiki/360_Assembly> thì e thấy trong assembly có khoảng trên 360 tập lệnh hoặc tổng cộng các lệnh trong cuốn IA-32 Intel® Architecture Volume 2 Instruction Set Reference cũng tầm 360 tập lệnh )

* **Các tập lệnh cơ bản**
* Data Transfer
* Move :

move dest,src

* Data swap

xchg src,dest

* Move with zero extend

movzx dest,src

* Sign Extend

movsx dest,src

* Move String

movsb

* Load Effective Address

lea dest,src

* Control Flow
* Comparison Instruction

**test** arg2, arg1

**cmp arg1,arg2**

* Jump Instruction
* Uncondition Jumps

**jmp loc**

* [Jump on Equality](http://en.wikibooks.org/wiki/X86_Assembly/Control_Flow)

**jne** loc

* [Jump on Inequality](http://en.wikibooks.org/wiki/X86_Assembly/Control_Flow)

**je** loc

* [Jump if Greater](http://en.wikibooks.org/wiki/X86_Assembly/Control_Flow)

**jge** loc

**jg** loc

**ja** loc

* [Jump if Less](http://en.wikibooks.org/wiki/X86_Assembly/Control_Flow)

**jl** loc

**jle** loc

**jbe** loc

* [Jump on Overflow](http://en.wikibooks.org/wiki/X86_Assembly/Control_Flow)

**jo** loc

* [Jump on Zero](http://en.wikibooks.org/wiki/X86_Assembly/Control_Flow)

**jz** loc

**jnz** loc

* [Function Calls](http://en.wikibooks.org/wiki/X86_Assembly/Control_Flow)

**call** proc

**ret** [val]

* [Loop Instructions](http://en.wikibooks.org/wiki/X86_Assembly/Control_Flow)

**loop** arg

**loopx** arg

* [Enter and Leave](http://en.wikibooks.org/wiki/X86_Assembly/Control_Flow)

**enter** arg

**leave**

* [Other Control Instructions](http://en.wikibooks.org/wiki/X86_Assembly/Control_Flow)

**Nop**

**Lock**

**wait**

* Arthmetic instruction
* Arthmetic **instruction**

**add dest, src**

**sub dest, src**

**mul arg**

**imul dest, src, aux**

**div arg**

**idiv arg**

**neg arg**

* Carry **Arithmetic Instructions**

**adc** dest, src

**sbb** dest, src

* Increment and Decrement

**inc arg**

**dec arg**

* Login Instructions

**and** dest, src

**or** dest, src

**xor** dest, src

**not** arg

* Shift and Rotate Instruction
* [Logical Shift Instructions](http://en.wikibooks.org/wiki/X86_Assembly/Shift_and_Rotate)

**shr** dest, src

**shl** dest, src

* [Arithmetic Shift Instructions](http://en.wikibooks.org/wiki/X86_Assembly/Shift_and_Rotate)

**sar** dest, src

* [Extended Shift Instructions](http://en.wikibooks.org/wiki/X86_Assembly/Shift_and_Rotate)

**shld** dest, src, cnt

* [Rotate Instructions](http://en.wikibooks.org/wiki/X86_Assembly/Shift_and_Rotate)

**ror** dest, src

**rol** dest, src

* [Rotate With Carry Instructions](http://en.wikibooks.org/wiki/X86_Assembly/Shift_and_Rotate)

**rcr** dest, src

**rcl** dest, src

* [Number of arguments](http://en.wikibooks.org/wiki/X86_Assembly/Shift_and_Rotate)
* [Notes](http://en.wikibooks.org/wiki/X86_Assembly/Shift_and_Rotate)
* Other Instructions
* X86 Interrupts

**int arg**

* **Các lệnh khác**
* **Segment Register Instructions**
* **Load Full Pointer (lds,les, lfs, lgs, and lss)**

**lds{wl} mem[32|48], reg[16|32]**

**les{wl} mem[32|48], reg[16|32]**

**lfs{wl} mem[32|48], reg[16|32]**

**lgs{wl} mem[32|48], reg[16|32]**

**lss{wl} mem[32|48], reg[16|32]**

* **Pop Stack into Word**

**pop{wl} r/m[16|32]**

**pop{l} [%ds|%ss|%es|%fs|%gs]**

* **Push Word/Long onto Stack**

**push{wl}r/m[16|32]**

**push{wl}imm[8|16|32]**

**push{l} [%cs|%ds|%ss|%es|%fs|%gs]**

* **I/O Instructions**
* **Input from Port (in, ins)**

**in{bwl} imm8**

**in{bwl} (%dx)**

**ins{bwl}**

* **Output from Port (out, outs)**

**out{bwl} imm8**

**out{bwl} (%dx)**

**outs{bwl}**

* **Flag Instructions**
* **Load Flags into AH Register (lahf)**

**lahf**

* **Store AH into Flags (sahf)**

**sahf**

* **Pop Stack into Flag (popf)**
* **popf{wl}**
* **Push Flag Register Onto Stack (pushf)**

**pushf{wl}**

**pushfl**

* **Complement Carry Flag (cmc)**

**cmc**

* **Clear Carry Flag (clc)**

**clc**

* **Set Carry Flag (stc)**

**stc**

* **Clear Interrupt Flag (cli)**

**cli**

* **Set Interrupt Flag (sti)**

**sti**

* **Clear Direction Flag (cld)**

**cld**

* **Set Direction Flag (std)**

**std**

* **Arithmetic Logical Instructions**
* **Integer Addition (add)**

**add{bwl} reg[8|16|32], r/m[8|16|32]**

**add{bwl} r/m[8|16|32], reg[8|16|32]**

**add{bwl} imm[8|16|32], r/m[8|16|32]**

* **Integer Add With Carry (adc)**

**adc{bwl} reg[8|16|32], r/m[8|16|32]**

**adc{bwl} r/m[8|16|32], reg[8|16|32]**

**adc{bwl} imm[8|16|32], r/m[8|16|32]**

* **Integer Subtraction (sub)**

**sub{bwl} reg[8|16|32], r/m[8|16|32]**

**sub{bwl} r/m[8|16|32], reg[8|16|32]**

**sub{bwl} imm[8|16|32], r/m[8|16|32]**

* **Integer Subtraction With Borrow (sbb**

**sbb{bwl} reg[8|16|32], r/m[8|16|32]**

**sbb{bwl} r/m[8|16|32], reg[8|16|32]**

**sbb{bwl} imm[8|16|32], r/m[8|16|32]**

* **Compare Two Operands (cmp)**

**cmp{bwl} reg[8|16|32], r/m[8|16|32]**

**cmp{bwl} r/m[8|16|32], reg[8|16|32]**

**cmp{bwl} imm[8|16|32], r/m[8|16|32]**

* **Increment by 1 (inc)**

**inc{bwl} r/m[8|16|32]**

* **Decrease by 1 (dec)**

**dec{bwl}r/m[8|16|32]**

* **Logical Comparison or Test (test)**

**test{bwl}reg[8|16|32], r/m[8|16|32]**

**test{bwl}r/m[8|16|32], reg[8|16|32]**

**test{bwl}imm[8|16|32], r/m[8|16|32]**

* **Shift (sal, shl, sar, shr)**

**sal{bwl} imm8, r/m[8|16|32]**

**sal{bwl} %cl, r/m[8|16|32]**

**shl{bwl} imm8, r/m[8|16|32]**

**shl{bwl} %cl, r/m[8|16|32]**

**sar{bwl} imm8, r/m[8|16|32]**

**sar{bwl} %cl, r/m[8|16|32]**

**shr{bwl} imm8, r/m[8|16|32]**

**shr{bwl} %cl, r/m[8|16|32]**

* **Double Precision Shift Left (shld)**

**shld{wl}imm8, reg[16|32], r/m[16|32]**

**shld{wl}%cl, reg[16|32], r/m[16|32]**

* **Double Precision Shift Right (shrd)**

**shrd{wl}imm8, reg[16|32], r/m[16|32]**

**shrd{wl}%cl, reg[16|32], r/m[16|32]**

* **One’s Complement Negation (not)**

**not{bwl} r/m[8|16|32]**

* **Two’s Complement Negation (neg)**

**neg{bwl} r/m[8|16|32]**

* **Check Array Index Against Bounds (bound)**

**bound{wl}reg[16|32], r/m[16|32]**

* **Logical And (and)**

**and{bwl} reg[8|16|32], r/m[8|16|32]**

**and{bwl} r/m[8|16|32], reg[8|16|32]**

**and{bwl} imm[8|16|32], r/m[8|16|32]**

* **Logical Inclusive OR (or)**

**or{bwl} reg[8|16|32], r/m[8|16|32]**

**or{bwl} r/m[8|16|32], reg[8|16|32]**

**or{bwl} imm[8|16|32], r/m[8|16|32]**

* **Logical Exclusive OR (xor)**

**xor{bwl} reg[8|16|32], r/m[8|16|32]**

**xor{bwl} r/m[8|16|32], reg[8|16|32]**

**xor{bwl} imm[8|16|32], r/m[8|16|32]**

* **Signed Multiply (imul)**

**imulb r/m8**

**imulw r/m16**

**imul{l} r/m32**

**imul{wl}r/m[16|32], reg[16|32]**

**imul{bwl}imm[16|32], r/m[16|32], reg[16|32]**

* **Unsigned Multiplication of AL, AX or EAX(mul)**

**mul{bwl} r/m[8|16|32]**

* **Unsigned Divide (div)**

**div{bwl} r/m[8|16|32]**

* **Signed Divide (idiv)**

**idiv{bwl}r/m[8|16|32]**

* **Conversion Instructions**
* **Convert Byte to Word (cbtw)**

**Cbtw**

* **Convert Word to Long (cwtl)**

**Cwtl**

* **Convert Signed Word to Signed Double Word (cwtd)**

**Cwtd**

* **Convert Signed Long to Signed Double Long (cltd)**

**Cltd**

* **Decimal Arithmetic Instructions**
* **Decimal Adjust AL after Addition (daa)**

**Daa**

* **Decimal Adjust AL after Subtraction (das)**

**Das**

* **ASCII Adjust after Addition (aaa)**

**Aaa**

* **ASCII Adjust after Subtraction (aas)**

**aas**

* **ASCII Adjust AX after Multiply (aam)**

**Aam**

* **ASCII Adjust AX before Division (aad**

**aad**

* **Coprocessor Instructions**
* **Wait (wait, fwait)**

**wait**

**fwait**

* **String Instructions**
* **Move Data from String to String (movs)**

**movs{bwl}**

**movs{bwl} m[8|16|32], reg[16|32]**

* **Compare String Operands (cmps)**

**cmps{bwl}**

* **Store String Data (stos)**

**stos{bwl}**

* **The Load String Operand (lods)**

**lods{bwl}**

* **Compare String Data (scas)**

**scas{bwl}**

* **Look-Up Translation Table (xlat)**

**xlat**

* **Repeat String Operation (rep, repnz, repz)**

**rep**

**repnz**

**repz**

* **Procedure Call and Return Instructions**
* **Far Call — Procedure Call (lcall)**

**lcall immptr**

**lcall \*mem48**

* **Near Call — Procedure Call (call)**

**call disp32**

**call \*r/m32**

* **Return from Procedure (ret)**

**ret**

**ret imm16**

* **Long Return (lret)**

**lret**

**lret imm16**

* **Enter/Make Stack Frame for Procedure Parameters (enter)**

**enter imm16, imm8**

* **High Level Procedure Exit (leave)**

**Leave**

* **Jump Instructions**
* **Jump if ECX is Zero (jcxz)**

**jcxz disp8**

* **Loop Control with CX Counter (loop, loopnz, loopz)**

**loop disp8**

**loopnz disp8**

**loopne disp8**

**loopz disp8**

**loope disp8**

* **Jump (jmp, ljmp)**

**jmp disp{8|16|3**

**jmp \*r/m{16|32}**

**ljmpimmPtr**

**ljmp\*mem48**

**jcc disp{8|32}**

* **Interrupt Instructions**
* **Call to Interrupt Procedure (int, into)**

**int 3**

**int imm**

**into**

* **Interrupt Return (iret)**

**Iret**

* **Protection Model Instructions**
* **Store Local Descriptor Table Register (sldt)**

**sldtr/m16**

* **Store Task Register (str)**

**str r/m16**

* **Load Local Descriptor Table Register (lldt)**

**lldt r/m16**

* **Load Task Register (ltr)**

**ltr r/m16**

* **Verify a Segment for Reading or Writing (verr, verw)**

**verr r/m16**

**verw r/m16**

* **Store Global/Interrupt Descriptor Table Register (sgdt, sidt)**

**sgdt mem48**

**sidt mem48**

* **Load Global/Interrupt Descriptor Table (lgdt, lidt)**

**lgdt mem48**

**lidt mem48**

* **Store Machine Status Word (smsw)**

**smsw r/m16**

* **Load Machine Status Word (lmsw)**

**lmsw r/m16**

* **Load Access Rights (lar)**

**lar r/m32, reg32**

* **Load Segment Limit (lsl)**

**lsl r/m32, reg32**

* **Clear Task-Switched (clts)**

**Adjust RPL Field of Selector (arpl)**

**arplr16, r/m16**

* **Bit Instructions**
* **Bit Scan Forward (bsf)**

**bsf{wl} r/m[16|32], reg[16|32]**

* **Bit Scan Reverse (bsr)**

**bsr{wl} r/m[16|32], reg[16|32]**

* **Bit Test (bt)**

**bt{wl} imm8, r/m[16|32]**

**bt{wl} reg[16|32], r/m[16|32]**

* **Bit Test And Complement (btc)**

**btc{wl} imm8, r/m[16|32]**

**btc{wl} reg[16|32], r/m[16|32]**

* **Bit Test And Reset (btr)**

**btr{wl} imm8, r/m[16|32]**

**btr{wl} reg[16|32], r/m[16|32]**

* **Bit Test And Set (bts)**

**bts{wl} imm8, r/m[16|32]**

**bts{wl} reg[16|32], r/m[16|32]**

* **Exchange Instructions**
* **Compare and Exchange (cmpxchg)[486]**

**cmpxchg{bwl}reg[8|16|32], r/m[8|16|32]**

* **Floating-Point Transcendental Instructions**
* **Floating-Point Sine (fsin)**

**Fsin**

* **Floating-Point Cosine (fcos)**

**Fcos**

* **Floating-Point Load One (fld)**

**fld1**

**fld12**

**fld12**

**fldpi**

**fldlg**

**fldln**

**fldz**

* **Processor Control Floating-Point Instructions**
* **Floating-Point Load Control Word (fldcw)**

**fldcwr/m16**

* **Floating-Point Load Environment (fldenv)**

**fldenvmem**

* **Miscellaneous Floating-Point Instructions**
* **Floating-Point Different Reminder (fprem)**

**fprem1**

* **Floating-Point Comparison Instructions**
* **Floating-Point Unsigned Compare (fucom)**

**Fucomfreg**

* **Floating-Point Unsigned Compare And Pop (fucomp)**

**Fucompfreg**

* **Floating-Point Unsigned Compare And Pop Two (fucompp)**

**Fucompp**

* **Load and Move Instructions**
* **Load Effective Address (lea)**

**lea{wl} r/m[16|32], reg[16|32]**

* **Move (mov)**

**mov{bwl}imm[8|16|32], r/m[8|16|32]**

**mov{bwl}reg[8|16|32], r/m[8|16|32]**

**mov{bwl}r/m[8|16|32], reg[8|16|32]**

* **Move Segment Registers (movw)**

**movwsreg,r/m16**

**movwr/m16, sreg**

* **Move Control Registers (mov)**

**mov{l}creg, reg32**

**mov{l}reg32, creg**

* **Move Debug Registers (mov)**

**mov{l}dreg, reg32**

**mov{l}reg32, dreg**

* **Move Test Registers (mov)**

**mov{l}treg, reg32**

**mov{l}reg32, treg**

* **Move With Sign Extend (movsx)**

**movsx{wl}r/m8, reg[16|32]**

**movsxwl r/m16, reg32**

* **Move With Zero Extend (movzb)**

**movzb[wl]r/m8, reg[16|32]**

**movzwl r/m16, reg32**

* **Pop Instructions**
* **Pop All General Registers (popa)**

**popa{wl}**

* **Push Instructions**
* **Push All General Registers (pusha)**

**pusha{wl}**

* **Rotate Instructions**
* **Rotate With Carry Left (rcl)**

**rcl{bwl}imm8, r/m[8|16|32]**

**rcl{bwl}%cl, r/m[8|16|32]**

* **Rotate With Carry Right (rcr)**

**rcr{bwl}imm8, r/m[8|16|32]**

**rcr{bwl}%cl, r/m[8|16|32]**

* **Rotate Left (rol)**

**rol{bwl}imm8, r/m[8|16|32]**

**rol{bwl}%cl, r/m[8|16|32]**

* **Rotate Right (ror)**

**ror{bwl}imm8, r/m[8|16|32]**

**ror{bwl}%cl, r/m[8|16|32]**

* **Byte Instructions**
* **Byte Set On Condition (setcc)**

**setcc r/m8**

**Byte Swap (bswap) [486]**

**bswapreg[16|32]**

* **Exchange Instructions**
* **Exchange And Add (xadd) [486]**

**xadd{bwl}reg[8|16|32], r/m[8|16|32]**

* **Exchange Register / Memory With Register (xchg)**

**xchg{bwl}reg[8|16|32], r/m[8|16|32]**

* **Miscellaneous Instructions**
* **Write Back and Invalidate Cache (wbinvd) [486 only]**

**Wbinvd**

* **Invalidate (invd) [486 only]**

**Invd**

* **Invalidate Page (invlpg) [486 only**

**invlpgmem32**

* **LOCK Preﬁx (lock)**

**Lock**

* **No Operation (nop)**

**Nop**

* **Halt (hlt)**

**hlt**

**Address Prefix**

**addr16**

**Data Prefix**

**data16**

* **Real Transfer Instructions**
* **Load Real (fld)**

**fld{lst}**

* **Store Real (fst)**

**fst{ls}**

* **Store Real and Pop (fstp)**

**fstp{lst}**

* **Exchange** Registers (fxch)

**Fxch**

* **Integer Transfer Instructions**
* **Integer Load (fild)**
* **Integer Store (fist)**
* **Integer Store and Pop (fistp)**
* **Packed Decimal Transfer Instructions**
* **Packed Decimal (BCD) Load (fbld)**
* **Packed Decimal (BCD) Store and Pop (fbstp)**
* **Addition Instructions**
* **Real Add (fadd)**
* **Real Add and Pop (faddp)**
* **Integer Add (fiadd)**
* **Subtraction Instructions**
* **subtract Real and Pop (fsub)**
* **Subtract Real (fsubp)**
* **Subtract Real Reversed (fsubr)**
* **Subtract Real Reversed and Pop (fsubrp)**
* **Integer Subtract (fisubrp)**
* **Integer Subtract Reverse (fisubr)**
* **Multiplication Instructions**
* **Multiply Real (fmul)**
* **Multiply Real and Pop (fmulp)**
* **nteger Multiply (fimul)**
* **Division Instructions**
* **Divide Real (fdiv)**
* **Divide Real and Pop (fdivp)**
* **Divide Real Reversed (fdivr)**
* **Divide Real Reversed and Pop (fdivrp)**
* **Integer Divide (fidiv)**
* **Integer Divide Reversed (fidivr)**
* **Miscellaneous Arithmetic Operations**
* **Square Root (fsqrt)**
* **Scale (fscale)**
* **Partial Remainder (fprem)**
* **Round to Integer (frndint)**
* **Extract Exponent and Signiﬁcand (fxtract)**
* **Absolute Value (fabs)**
* **Change Sign (fchs)**
* **Comparison Instructions**
* **Compare Real (fcom)**
* **Compare Real and Pop (fcomp)**
* **Compare Real and Pop Twice (fcompp)**
* **Integer Compare (ficom)**
* **Integer Compare and Pop (ficomp)**
* **Test (ftst)**
* **Examine (fxam)**
* **Transcendental Instructions**
* **Partial Tangent (fptan)**
* **Partial Arctangent (fpatan)**
* **Constant Instructions**
* **Load log2 E (fldl2e)**
* **Load log2 10 (fldl2t)**
* **Load log 2 (fldlg2)**
* **Load loge 2 (fldln2)**
* **Load pi (fldpi)**
* **Load + 0 (fldz)**
* **Processor Control Instructions**
* **Initialize Processor (finit, fnint)**
* **No Operation (fnop)**
* **Save State (fsave, fnsave)**
* **Store Control Word (fstcw, fnstcw)**
* **Store Environment (fstenv, fnstenv)**
* **Store Status Word (fstsw, fnstsw)**
* **Restore State (frstor)**
* **CPU Wait (fwait, wait)**
* **Clear Exceptions (fclex, fnclex)**
* **Decrement Stack Pointer (fdecstp)**
* **Free Registers (ffree)**
* **Increment Stack Pointer (fincstp)**
* **F**
* **Ff**
* **F**
* **Ff**
* **f**