**X86 Assembly**

1. **Các tập lệnh cơ bản**
2. Data Transfer
3. Move :

move dest,src

1. Data swap

xchg src,dest

1. Move with zero extend

movzx dest,src

1. Sign Extend

movsx dest,src

1. Move String

movsb

1. Load Effective Address

lea dest,src

1. Control Flow
2. Comparison Instruction

**test** arg2, arg1

**cmp arg1,arg2**

1. Jump Instruction
2. Uncondition Jumps

**jmp loc**

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

**jne** loc

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

**je** loc

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

**jge** loc

**jg** loc

**ja** loc

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

**jl** loc

**jle** loc

**jbe** loc

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

**jo** loc

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

**jz** loc

**jnz** loc

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

**call** proc

**ret** [val]

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

**loop** arg

**loopx** arg

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

**enter** arg

**leave**

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

**Nop**

**Lock**

**wait**

1. Arthmetic instruction
2. Arthmetic **instruction**

**add dest, src**

**sub dest, src**

**mul arg**

**imul dest, src, aux**

**div arg**

**idiv arg**

**neg arg**

1. Carry **Arithmetic Instructions**

**adc** dest, src

**sbb** dest, src

1. Increment and Decrement

**inc arg**

**dec arg**

1. Login Instructions

**and** dest, src

**or** dest, src

**xor** dest, src

**not** arg

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

**shr** dest, src

**shl** dest, src

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

**sar** dest, src

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

**shld** dest, src, cnt

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

**ror** dest, src

**rol** dest, src

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

**rcr** dest, src

**rcl** dest, src

* 1. [Number of arguments](http://en.wikibooks.org/wiki/X86_Assembly/Shift_and_Rotate#Number_of_arguments)
  2. [Notes](http://en.wikibooks.org/wiki/X86_Assembly/Shift_and_Rotate#Notes)

1. Other Instructions
2. X86 Interrupts

**int arg**

1. **Các lệnh khác**
2. **Segment Register Instructions**
3. **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]**

1. **Pop Stack into Word**

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

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

1. **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]**

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

**in{bwl} imm8**

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

**ins{bwl}**

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

**out{bwl} imm8**

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

**outs{bwl}**

1. **Flag Instructions**
2. **Load Flags into AH Register (lahf)**

**lahf**

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

**sahf**

1. **Pop Stack into Flag (popf)**
2. **popf{wl}**
3. **Push Flag Register Onto Stack (pushf)**

**pushf{wl}**

**pushfl**

1. **Complement Carry Flag (cmc)**

**cmc**

1. **Clear Carry Flag (clc)**

**clc**

1. **Set Carry Flag (stc)**

**stc**

1. **Clear Interrupt Flag (cli)**

**cli**

1. **Set Interrupt Flag (sti)**

**sti**

1. **Clear Direction Flag (cld)**

**cld**

1. **Set Direction Flag (std)**

**std**

1. **Arithmetic Logical Instructions**
2. **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]**

1. **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]**

1. **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]**

1. **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]**

1. **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]**

1. **Increment by 1 (inc)**

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

1. **Decrease by 1 (dec)**

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

1. **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]**

1. **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]**

1. **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]**

1. **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]**

1. **One’s Complement Negation (not)**

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

1. **Two’s Complement Negation (neg)**

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

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

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

1. **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]**

1. **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]**

1. **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]**

1. **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]**

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

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

1. **Unsigned Divide (div)**

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

1. **Signed Divide (idiv)**

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

1. **Conversion Instructions**
2. **Convert Byte to Word (cbtw)**

**Cbtw**

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

**Cwtl**

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

**Cwtd**

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

**Cltd**

1. **Decimal Arithmetic Instructions**
2. **Decimal Adjust AL after Addition (daa)**

**Daa**

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

**Das**

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

**Aaa**

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

**aas**

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

**Aam**

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

**aad**

1. **Coprocessor Instructions**
2. **Wait (wait, fwait)**

**wait**

**fwait**

1. **String Instructions**
2. **Move Data from String to String (movs)**

**movs{bwl}**

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

1. **Compare String Operands (cmps)**

**cmps{bwl}**

1. **Store String Data (stos)**

**stos{bwl}**

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

**lods{bwl}**

1. **Compare String Data (scas)**

**scas{bwl}**

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

**xlat**

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

**rep**

**repnz**

**repz**

1. **Procedure Call and Return Instructions**
2. **Far Call — Procedure Call (lcall)**

**lcall immptr**

**lcall \*mem48**

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

**call disp32**

**call \*r/m32**

1. **Return from Procedure (ret)**

**ret**

**ret imm16**

1. **Long Return (lret)**

**lret**

**lret imm16**

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

**enter imm16, imm8**

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

**Leave**

1. **Jump Instructions**
2. **Jump if ECX is Zero (jcxz)**

**jcxz disp8**

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

**loop disp8**

**loopnz disp8**

**loopne disp8**

**loopz disp8**

**loope disp8**

1. **Jump (jmp, ljmp)**

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

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

**ljmpimmPtr**

**ljmp\*mem48**

**jcc disp{8|32}**

1. **Interrupt Instructions**
2. **Call to Interrupt Procedure (int, into)**

**int 3**

**int imm**

**into**

1. **Interrupt Return (iret)**

**Iret**

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

**sldtr/m16**

1. **Store Task Register (str)**

**str r/m16**

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

**lldt r/m16**

1. **Load Task Register (ltr)**

**ltr r/m16**

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

**verr r/m16**

**verw r/m16**

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

**sgdt mem48**

**sidt mem48**

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

**lgdt mem48**

**lidt mem48**

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

**smsw r/m16**

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

**lmsw r/m16**

1. **Load Access Rights (lar)**

**lar r/m32, reg32**

1. **Load Segment Limit (lsl)**

**lsl r/m32, reg32**

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

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

**arplr16, r/m16**

1. **Bit Instructions**
2. **Bit Scan Forward (bsf)**

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

1. **Bit Scan Reverse (bsr)**

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

1. **Bit Test (bt)**

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

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

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

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

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

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

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

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

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

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

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

1. **Exchange Instructions**
2. **Compare and Exchange (cmpxchg)[486]**

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

1. **Floating-Point Transcendental Instructions**
2. **Floating-Point Sine (fsin)**

**Fsin**

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

**Fcos**

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

**fld1**

**fld12**

**fld12**

**fldpi**

**fldlg**

**fldln**

**fldz**

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

**fldcwr/m16**

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

**fldenvmem**

1. **Miscellaneous Floating-Point Instructions**
2. **Floating-Point Different Reminder (fprem)**

**fprem1**

1. **Floating-Point Comparison Instructions**
2. **Floating-Point Unsigned Compare (fucom)**

**Fucomfreg**

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

**Fucompfreg**

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

**Fucompp**

1. **Load and Move Instructions**
2. **Load Effective Address (lea)**

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

1. **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]**

1. **Move Segment Registers (movw)**

**movwsreg,r/m16**

**movwr/m16, sreg**

1. **Move Control Registers (mov)**

**mov{l}creg, reg32**

**mov{l}reg32, creg**

1. **Move Debug Registers (mov)**

**mov{l}dreg, reg32**

**mov{l}reg32, dreg**

1. **Move Test Registers (mov)**

**mov{l}treg, reg32**

**mov{l}reg32, treg**

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

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

**movsxwl r/m16, reg32**

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

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

**movzwl r/m16, reg32**

1. **Pop Instructions**
2. **Pop All General Registers (popa)**

**popa{wl}**

1. **Push Instructions**
2. **Push All General Registers (pusha)**

**pusha{wl}**

1. **Rotate Instructions**
2. **Rotate With Carry Left (rcl)**

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

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

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

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

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

1. **Rotate Left (rol)**

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

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

1. **Rotate Right (ror)**

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

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

1. **Byte Instructions**
2. **Byte Set On Condition (setcc)**

**setcc r/m8**

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

**bswapreg[16|32]**

1. **Exchange Instructions**
2. **Exchange And Add (xadd) [486]**

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

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

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

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

**Wbinvd**

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

**Invd**

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

**invlpgmem32**

1. **LOCK Preﬁx (lock)**

**Lock**

1. **No Operation (nop)**

**Nop**

1. **Halt (hlt)**

**hlt**

**Address Prefix**

**addr16**

**Data Prefix**

**data16**

1. **Real Transfer Instructions**
2. **Load Real (fld)**

**fld{lst}**

1. **Store Real (fst)**

**fst{ls}**

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

**fstp{lst}**

1. **Exchange** Registers (fxch)

**Fxch**

1. **Integer Transfer Instructions**
2. **Integer Load (fild)**
3. **Integer Store (fist)**
4. **Integer Store and Pop (fistp)**
5. **Packed Decimal Transfer Instructions**
6. **Packed Decimal (BCD) Load (fbld)**
7. **Packed Decimal (BCD) Store and Pop (fbstp)**
8. **Addition Instructions**
9. **Real Add (fadd)**
10. **Real Add and Pop (faddp)**
11. **Integer Add (fiadd)**
12. **Subtraction Instructions**
13. **subtract Real and Pop (fsub)**
14. **Subtract Real (fsubp)**
15. **Subtract Real Reversed (fsubr)**
16. **Subtract Real Reversed and Pop (fsubrp)**
17. **Integer Subtract (fisubrp)**
18. **Integer Subtract Reverse (fisubr)**
19. **Multiplication Instructions**
20. **Multiply Real (fmul)**
21. **Multiply Real and Pop (fmulp)**
22. **nteger Multiply (fimul)**
23. **Division Instructions**
24. **Divide Real (fdiv)**
25. **Divide Real and Pop (fdivp)**
26. **Divide Real Reversed (fdivr)**
27. **Divide Real Reversed and Pop (fdivrp)**
28. **Integer Divide (fidiv)**
29. **Integer Divide Reversed (fidivr)**
30. **Miscellaneous Arithmetic Operations**
31. **Square Root (fsqrt)**
32. **Scale (fscale)**
33. **Partial Remainder (fprem)**
34. **Round to Integer (frndint)**
35. **Extract Exponent and Signiﬁcand (fxtract)**
36. **Absolute Value (fabs)**
37. **Change Sign (fchs)**
38. **Comparison Instructions**
39. **Compare Real (fcom)**
40. **Compare Real and Pop (fcomp)**
41. **Compare Real and Pop Twice (fcompp)**
42. **Integer Compare (ficom)**
43. **Integer Compare and Pop (ficomp)**
44. **Test (ftst)**
45. **Examine (fxam)**
46. **Transcendental Instructions**
47. **Partial Tangent (fptan)**
48. **Partial Arctangent (fpatan)**
49. **Constant Instructions**
50. **Load log2 E (fldl2e)**
51. **Load log2 10 (fldl2t)**
52. **Load log 2 (fldlg2)**
53. **Load loge 2 (fldln2)**
54. **Load pi (fldpi)**
55. **Load + 0 (fldz)**
56. **Processor Control Instructions**
57. **Initialize Processor (finit, fnint)**
58. **No Operation (fnop)**
59. **Save State (fsave, fnsave)**
60. **Store Control Word (fstcw, fnstcw)**
61. **Store Environment (fstenv, fnstenv)**
62. **Store Status Word (fstsw, fnstsw)**
63. **Restore State (frstor)**
64. **CPU Wait (fwait, wait)**
65. **Clear Exceptions (fclex, fnclex)**
66. **Decrement Stack Pointer (fdecstp)**
67. **Free Registers (ffree)**
68. **Increment Stack Pointer (fincstp)**
69. **F**
70. **Ff**
71. **F**
72. **Ff**
73. **f**