Opcode Decoding Tutorial part I

DISCLAIMER

The following document is a study based on information made public by Intel(c). This article represent a comprehensive explanation of the methods behind the opcode generation inside the Intel or Intel compatible microprocessors.

The information revealed here is not intended to harm anybody and the author cannot be held responsible for it's use that led to data loss.

- Yash K.S

------¦ Foreword ¦---

This article mainly explains about One-Byte opcode decoding completely. I assume after understanding this u can explore yourself Two-byte opcode table easily. This article will explains by giving more examples of instruction decoding.

------¦ General Machine Instruction Format ¦---------------------------------

All Intel Architecture instructions are encoded using the subsets of the general machine instruction format shown in Figure A-1.

Each instruction consists of an Opcode, a Register and/or Address mode specifier (if required) consisting of the ModR/M byte and sometimes the scale-index-base (SIB) byte, a displacement (if required), and an immediate data field (if required).

[Figure A-1]

```
+-----------------------------------------------------------------------+
¦                 ¦                 ¦      * * *      ¦                 ¦
¦ 7 6 5 4 3 2 1 0 ¦ 7 6 5 4 3 2 1 0 ¦ 7 6 5 4 3 2 1 0 ¦ 7 6 5 4 3 2 1 0 ¦
+-+-+-+-+-+-+-+-+---+-+-+-+-+-+-+-+---+-+-+-+-+-+-+-+---+-+-+-+-+-+-+-+-+
  ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦   ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦   ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦   ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦
  +-------------------------------+     +-------------+     +-------------+
                  ¦                            ¦                  ¦
                  ¦                            ¦                  ¦
                  ¦                            ¦                  ¦

               Opcode                     ModR/M Byte        SIB Byte
            1 or 2 Bytes
```

d32 ¦ d16 ¦ d8 ¦ none
[Address displacement 4,2,1 Bytes or None]

d32 ¦ d16 ¦ d8 ¦ none
[Immediate displacement 4,2,1 Bytes or None]

* * *

Note : 5 4 3, Sometimes these 3 bits used as Opcode extensions

---[Opcode byte]---

The primary opcode of the instruction is encoded in one or two bytes of the instruction. Some instructions also use an opcode eXtension (Check figure A-1) field encoded in bits of 5,4,3 of the ModR/M byte. These fields define information such as Register encoding, Conditional test performed or Sign eXtension of immediate byte. Everything depends on First byte of instruction.

---[Mod R/M byte]--

After Primary opcode the next byte is ModR/M byte. The ModR/M byte consists of three field of information (Refer Table A-2) :

¦ Bits 7,6 is used as MOD, this bits will combine with the r/m field to form 32 bit possible values : 8 registers and 24 addressing modes.

¦ Bits 5,4,3 is used as REG/OPCODE eXtension, These bits specifies either a Register number or Opcode extension. The purpose of this field is specified in the primary opcode(First byte in Instruction filed).

¦ Bits 2,1,0 is used as R/M, can specify register as an operand or can combine with the MOD field to encode an addressing mode.

---[SIB byte]--

Sometimes encoding of the ModR/M requires a SIB byte to fully specify the addressing form. The base+index and scale+index forms a 32bit addressing require the SIB byte. The SIB byte consists 3 field of information (Refer Table A-3) :

¦ Bits 7,6 is used as SCALE field for specifying scale factor.

¦ Bits 5,4,3 is used as INDEX field for specifying index register.

¦ Bits 2,1,0 is used as BASE field and specifies base register.

---[Displacement and Immediate bytes]--------------------------------------

Some addressing forms includes a displacement following either ModR/M or SIB byte. If displacement required the size can be 1 or 2 or 4 bytes.

The Immediate operand comes always last in instruction field. An immediate operand can be 1 or 2 or 4 bytes.

------¦ Opcode Tables ¦--

Now starts real decoding part. The opcode length may be either 1 byte or 2 byte maximum. First we will go through about 1 byte opcode table because using 1 byte opcode we have to refer 2 byte opcode also. For decoding

1 byte instruction refer Table A-1. In this table, Intel has provided some symbols to specify the operands(It helps for decoding the instruction). This table helps only to get names like ADD, PUSH, MOV etc., and other Addressing modes comes next subsequent table. Following definitions explains the meaning of the symbols (This definitions u have to refer while decoding the instructions) :

All operands are specified in two-character of the form Zz. Where the Uppercase letter 'Z' indicates addressing mode and the Lowercase letter 'z' indicates the type of operand.

---[Codes for Addressing Method ('Z' Types)]--

A   Direct address. The instruction has no ModR/M byte. The address of the operand is coded in the instruction: no base register, index register, or scaling factor can be applied.
for ex : far JMP(EA)

C   The reg field of the ModR/M byte selects the control register.
for ex : MOV (0F20, 0F22)

D   The reg field of the ModR/M byte selects the debug register.
for ex : MOV (0F21, 0F23)

E   A ModR/M byte follows the opcode and specifies the operand. The operand is either a general purpose register or memory address. If it is memory address, the address is computed from a segment register and any of the following register: a base register, an index register, a scaling factor, a displacement.

F   EFLAGS register.

G   The reg field of ModR/M byte selects a general purpose register.
for ex : AX (000)

I   Immediate data. The operand value is encoded in subsequent bytes of the instruction.

J   The instruction contains relative offset to be added to the instruction pointer register.
for ex : JMP(0E9), LOOP

M   The ModR/M may refer to only memory.
for ex : BOUND, LES, LDS, LSS, LFS, LGS, CMPXCHG8B

O   The instruction has no ModR/M byte. The offset of the operand is coded as word or double word in the instruction. No base register, Index register, or Scale factor can be applied.
for ex : MOV (A0-A3)

P   The reg field of the ModR/M byte selects a packed quadword (MMX Registers).

Q   A ModR/M byte followed the opcode and specifies the operand. The operand is either MMX register or Memory address. If it is memory address, the address is computed from a segment register and any of the following values : a base register, an index register, a scaling factor, and a displacement.

R   The Mod field of the ModR/M byte may refer only to a general register.
for ex : MOV (0F20-0F24, 0F26)

S   The reg field of the ModR/M byte selects a segment register.
for ex : MOV (8C, 8E)

T   The reg field of the ModR/M bytes selects a test register.
for ex : MOV (0F24, 0F26)

V   The reg field of the ModR/M byte selects a packed SIMD Floating point register.

W   An ModR/M byte follows the opcode and specifies the operand. The operand is either a SIMD floating-point register or a memory address. If it is a memory address, the address is computed from a segment register and any of the following values: a base register, an index register, a scaling factor, and a displacement.

X   Memory addressed by the DS:SI register pair.
for ex : MOVS, CMPS, OUTS or LODS

Y   Memory addressed by the ES:DI register pair.
for ex : MOVS, CMPS, INS, STOS, or SCAS

---[Codes for Operand ('z' Types)]--

a   Two one-word operands in memory or two double-word operands in memory, depending on operand-size attribute (used only by the BOUND instruction).

b   Byte, regardless of operand-size attribute.

c   Byte or word, depending on operand-size attribute.

d   Doubleword, regardless of operand-size attribute.

dq Double-quadword, regardless of operand-size attribute.

p   32-bit or 48-bit pointer, depending on operand-size attribute.

pi Quadword MMX technology register (e.g. mm0)

ps 128-bit packed FP single-precision data.

q   Quadword, regardless of operand-size attribute.

s   6-byte pseudo-descriptor.

ss Scalar element of a 128-bit packed FP single-precision data.

si Doubleword integer register (e.g., eax)

v   Word or doubleword, depending on operand-size attribute.

```
        w   Word, regardless of operand-size attribute.

    [ One-Byte Opcode Table A-1 ]

     0         1         2         3         4         5         6        7
 +---------------------------------------------------------------------------+
 ¦                            ADD                          ¦  PUSH  ¦   POP  ¦
0¦---------------------------------------------------------¦        ¦        ¦
 ¦ Eb,Gb  ¦ Ev,Gv  ¦  Gb,Eb  ¦  Gv,Ev  ¦  AL,Ib  ¦ eAX,Iv  ¦  ES    ¦   ES   ¦
 ¦---------------------------------------------------------+--------+--------¦
 ¦                            ADC                          ¦  PUSH  ¦   POP  ¦
1¦---------------------------------------------------------¦        ¦        ¦
 ¦ Eb,Gb  ¦ Ev,Gv  ¦  Gb,Eb  ¦  Gv,Ev  ¦  AL,Ib  ¦ eAX,Iv  ¦  SS    ¦   SS   ¦
 ¦---------------------------------------------------------+--------+--------¦
 ¦                            AND                          ¦  SEG   ¦        ¦
2¦---------------------------------------------------------¦        ¦   DAA  ¦
 ¦ Eb,Gb  ¦ Ev,Gv  ¦  Gb,Eb  ¦  Gv,Ev  ¦  AL,Ib  ¦ eAX,Iv  ¦  =ES   ¦        ¦
 ¦---------------------------------------------------------+--------+--------¦
 ¦                            XOR                          ¦  SEG   ¦        ¦
3¦---------------------------------------------------------¦        ¦   AAA  ¦
 ¦ Eb,Gb  ¦ Ev,Gv  ¦  Gb,Eb  ¦  Gv,Ev  ¦  AL,Ib  ¦ eAX,Iv  ¦  =SS   ¦        ¦
 ¦---------------------------------------------------------------------------¦
 ¦                             INC general register                          ¦
4¦---------------------------------------------------------------------------¦
 ¦  eAX   ¦  eCX   ¦   eDX   ¦   eBX   ¦   eSP   ¦   eBP   ¦  eSI   ¦  eDI   ¦
 ¦---------------------------------------------------------------------------¦
 ¦                             PUSH general register                         ¦
5¦---------------------------------------------------------------------------¦
 ¦  eAX   ¦  eCX   ¦   eDX   ¦   eBX   ¦   eSP   ¦   eBP   ¦  eSI   ¦  eDI   ¦
 ¦--------+--------+---------+---------+---------+---------+--------+--------¦
 ¦        ¦        ¦  BOUND  ¦  ARPL   ¦   SEG   ¦   SEG   ¦ Operand¦ Address¦
6¦ PUSHA  ¦ POPA   ¦         ¦         ¦         ¦         ¦        ¦        ¦
 ¦        ¦        ¦  Gv,Ma  ¦  Ew,Rw  ¦   =FS   ¦   =GS   ¦  Size  ¦  Size  ¦
 ¦---------------------------------------------------------------------------¦
 ¦                  Short displacement jump of condition (One-Byte)          ¦
7¦---------------------------------------------------------------------------¦
 ¦  JO    ¦  JNO   ¦   JB    ¦   JNB   ¦   JZ    ¦  JNZ    ¦   JBE  ¦  JNBE  ¦
 ¦-----------------+---------+---------+-------------------+-----------------¦
 ¦ Immediate Grp1  ¦         ¦  Grp1   ¦       TEST        ¦      XCHG       ¦
8¦-----------------¦         ¦         +-------------------+-----------------¦
 ¦ Eb,Ib  ¦ Ev,Iv  ¦         ¦  Ev,Iv  ¦  Eb,Gb  ¦  Ev,Gv  ¦  Eb,Gb ¦  Ev,Gv ¦
 ¦--------+------------------------------------------------------------------¦
 ¦        ¦             XCHG word or double-word register with eAX           ¦
9¦  NOP   +------------------------------------------------------------------¦
 ¦        ¦  eCX   ¦   eDX   ¦   eBX   ¦   eSP   ¦   eBP   ¦  eSI   ¦  eDI   ¦
 ¦-------------------------------------+---------+---------+--------+--------¦
 ¦                MOV                  ¦  MOVSB  ¦ MOVSW/D ¦ CMPSB  ¦CMPSW/D ¦
A¦-------------------------------------¦         ¦         ¦        ¦        ¦
 ¦ AL,Ob  ¦ eAX,Ov ¦  Ob,AL  ¦  Ov,eAX ¦  Xb,Yb  ¦  Xv,Yv  ¦  Xb,Yb ¦  Xv,Yv ¦
 ¦---------------------------------------------------------------------------¦
 ¦                     MOV immediate byte into byte register                 ¦
B¦---------------------------------------------------------------------------¦
 ¦   AL   ¦  CL    ¦   DL    ¦   BL    ¦   AH    ¦   CH    ¦   DH   ¦   BH   ¦
 ¦-----------------+-------------------+---------+---------+-----------------¦
 ¦   Shift Grp2    ¦      RET near     ¦   LES   ¦   LDS   ¦       MOV       ¦
```

```
C¦------------------+-------------------¦         ¦         +-----------------¦
 ¦  Eb,Ib ¦ Ev,Iv   ¦   Iw    ¦         ¦ Gv,Mp   ¦ Gv,Mp   ¦ Eb,Ib  ¦ Ev,Iv  ¦
 ¦--------------------------------------+---------+---------+--------+--------¦
 ¦             Shift Grp2               ¦         ¦         ¦        ¦        ¦
D¦--------------------------------------¦  AAM    ¦  AAD    ¦        ¦ XLAT   ¦
 ¦  Eb,1  ¦ Ev,1    ¦ Eb,CL   ¦ Ev,CL   ¦         ¦         ¦        ¦        ¦
 ¦--------+---------+---------+---------+-------------------+-----------------¦
 ¦ LOOPNE ¦ LOOPE   ¦  LOOP   ¦ JCXZ    ¦       IN          ¦      OUT        ¦
E¦        ¦         ¦         ¦         +-------------------+-----------------¦
 ¦  Jb    ¦  Jb     ¦   Jb    ¦  Jb     ¦ AL,Ib   ¦ eAX,Ib  ¦ Ib,AL  ¦ Ib,eAX ¦
 ¦--------+---------+---------+---------+---------+---------+-----------------¦
 ¦        ¦         ¦         ¦  REP    ¦         ¦         ¦    Unary Grp3   ¦
F¦ LOCK   ¦         ¦ REPNE   ¦         ¦  HLT    ¦  CMC    +-----------------¦
 ¦        ¦         ¦         ¦ REPE    ¦         ¦         ¦  Eb    ¦  Ev    ¦
 +----------------------------------------------------------------------------+


      8         9         A         B         C         D         E        F
 +----------------------------------------------------------------------------+
 ¦                           OR                             ¦ PUSH   ¦2-byte  ¦
0+----------------------------------------------------------¦        ¦        ¦
 ¦ Eb,Gb ¦  Ev,Gv   ¦ Gb,Eb   ¦ Gv,Ev   ¦ AL,Ib   ¦eAX,Iv   ¦  CS    ¦escape  ¦
 ¦----------------------------------------------------------+--------+--------¦
 ¦                          SBB                             ¦ PUSH   ¦ POP    ¦
1+----------------------------------------------------------¦        ¦        ¦
 ¦ Eb,Gb ¦  Ev,Gv   ¦ Gb,Eb   ¦ Gv,Ev   ¦ AL,Ib   ¦eAX,Iv   ¦ DS     ¦  DS    ¦
 ¦----------------------------------------------------------+--------+--------¦
 ¦                          SUB                             ¦ SEG    ¦        ¦
2+----------------------------------------------------------¦        ¦ DAS    ¦
 ¦ Eb,Gb ¦  Ev,Gv   ¦ Gb,Eb   ¦ Gv,Ev   ¦ AL,Ib   ¦eAX,Iv   ¦ =CS    ¦        ¦
 ¦----------------------------------------------------------+--------+--------¦
 ¦                          CMP                             ¦ SEG    ¦        ¦
3+----------------------------------------------------------¦        ¦ AAS    ¦
 ¦ Eb,Gb ¦  Ev,Gv   ¦ Gb,Eb   ¦ Gv,Ev   ¦ AL,Ib   ¦eAX,Iv   ¦ =DS    ¦        ¦
 ¦----------------------------------------------------------------------------¦
 ¦                            DEC general register                            ¦
4+----------------------------------------------------------------------------¦
 ¦  eAX  ¦   eCX    ¦  eDX    ¦  eBX    ¦  eSP    ¦  eBP    ¦  eSI   ¦ eDI    ¦
 ¦----------------------------------------------------------------------------¦
 ¦                        POP into general register                           ¦
5+----------------------------------------------------------------------------¦
 ¦  eAX  ¦   eCX    ¦  eDX    ¦  eBX    ¦  eSP    ¦  eBP    ¦ eSI    ¦ eDI    ¦
 ¦-------+----------+---------+---------+---------+---------+--------+--------¦
 ¦ PUSH  ¦  IMUL    ¦ PUSH    ¦ IMUL    ¦ INSB    ¦INSW/D   ¦OUTSB   ¦OUTSW/D ¦
6¦       ¦          ¦         ¦         ¦         ¦         ¦        ¦        ¦
 ¦  Ib   ¦ GvEvIv   ¦  Ib     ¦GvEvIv   ¦ Yb,DX   ¦ Yb,DX   ¦Dx,Xb   ¦DX,Xv   ¦
 ¦----------------------------------------------------------------------------¦
 ¦                 Short-displacement jump on condition(One-Byte)             ¦
7+----------------------------------------------------------------------------¦
 ¦  JS   ¦   JNS    ¦  JP     ¦  JNP    ¦  JL     ¦ JNL     ¦ JLE    ¦ JNLE   ¦
 ¦--------------------------------------+---------+---------+--------+--------¦
 ¦               MOV                    ¦  MOV    ¦  LEA    ¦ MOV    ¦ POP    ¦
8+--------------------------------------¦         ¦         ¦        ¦        ¦
 ¦ Eb,Gb ¦  Ev,Gv   ¦ Gb,Eb   ¦ Gv,Ev   ¦ Ew,Sw   ¦ Gv,M    ¦ Sw,Ew  ¦  Ev    ¦
 ¦-------+----------+---------+---------+---------+---------+--------+--------¦
 ¦       ¦          ¦ CALL    ¦         ¦ PUSHF   ¦ POPF    ¦        ¦        ¦
9¦  CBW  ¦   CWD    ¦         ¦ WAIT    ¦         ¦         ¦ SAHF   ¦ LAHF   ¦
```

```
 ¦       ¦         ¦  Ap     ¦         ¦  Fv     ¦  Fv     ¦        ¦        ¦
 ¦-----------------+---------+---------+---------+---------+--------+--------¦
 ¦      TEST       ¦  STOSB  ¦ STOSW/D ¦  LODSB  ¦ LODSW/D ¦ SCASB  ¦SCASW/D ¦
A+-----------------¦         ¦         ¦         ¦         ¦        ¦        ¦
 ¦ AL,Ib ¦ eAX,Iv  ¦  Yb,AL  ¦  Yv,eAX ¦  AL,Xb  ¦ eAX,Xv  ¦  AL,Xb ¦eAX,Xv  ¦
 ¦---------------------------------------------------------------------------¦
 ¦      MOV immediate word or double into word or double register           ¦
B+---------------------------------------------------------------------------¦
 ¦  eAX  ¦   eCX   ¦   eDX   ¦   eBX   ¦   eSP   ¦   eBP   ¦   eSI  ¦  eDI   ¦
 ¦-------+---------+-------------------+---------+---------+--------+--------¦
 ¦ ENTER ¦         ¦      RET far      ¦  INT    ¦  INT    ¦        ¦        ¦
C¦       ¦  LEAVE  +-------------------¦         ¦         ¦  INTO  ¦  IRET  ¦
 ¦ Iw,Ib ¦         ¦   Iw    ¦         ¦   3     ¦  Ib     ¦        ¦        ¦
 ¦---------------------------------------------------------------------------¦
 ¦                                                                           ¦
D¦                ESC(Escape to coprocessor instruction set)                 ¦
 ¦                                                                           ¦
 ¦---------------------------------------------------------------------------¦
 ¦ CALL  ¦            JNP                ¦       IN          ¦      OUT      ¦
E¦       +-------------------------------+-------------------+-----------------¦
 ¦  Av   ¦   Jv    ¦   Ap    ¦   Jb    ¦  AL,DX  ¦ eAX,DX  ¦ DX,AL  ¦ DX,eAX ¦
 ¦-------+---------+---------+---------+---------+---------+--------+--------¦
 ¦       ¦         ¦         ¦         ¦         ¦         ¦INC/DEC ¦Indirct ¦
F¦  CLC  ¦   STC   ¦   CLI   ¦   STI   ¦   CLD   ¦   STD   ¦        ¦        ¦
 ¦       ¦         ¦         ¦         ¦         ¦         ¦  Grp4  ¦  Grp5  ¦
 +---------------------------------------------------------------------------+

 ---------------------------------------------------------------------------
 NOTES: All blanks in the opcode maps are reserved and should not be depend
 on the operation of these undefined opcodes ( I guess this they can use to
 extended Instruction Set Table further like 3 - byte opcode ;-) )
 ---------------------------------------------------------------------------
```

Below tables are related to Addressing modes using ModR/M and SIB byte.

[32-Bit Addressing Forms with the ModR/M Byte Table A-2]

r8(/r)		AL	CL	DL	BL	AH	CH	DH	BH
r16(/r)		AX	CX	DX	BX	SP	BP	SI	DI
r32(/r)		EAX	ECX	EDX	EBX	ESP	EBP	ESI	EDI
/digit (Opcode)		0	1	2	3	4	5	6	7
REG =		000	001	010	011	100	101	110	111
	Mod R/M	ModR/M Values in Hexadecimal							
[EAX]	00 000	00	08	10	18	20	28	30	38
[ECX]	00 001	01	09	11	19	21	29	31	39
[EDX]	00 010	02	0A	12	1A	22	2A	32	3A
[EBX]	00 011	03	0B	13	1B	23	2B	33	3B
[--] [--]	00 100	04	0C	14	1C	24	2C	34	3C
disp32	00 101	05	0D	15	1D	25	2D	35	3D
[ESI]	00 110	06	0E	16	1E	26	2E	36	3E
[EDI]	00 111	07	0F	17	1F	27	2F	37	3F
disp8[EAX]	01 000	40	48	50	58	60	68	70	78
disp8[ECX]	01 001	41	49	51	59	61	69	71	79
disp8[EDX]	01 010	42	4A	52	5A	62	6A	72	7A
disp8[EBX];	01 011	43	4B	53	5B	63	6B	73	7B
disp8[--] [--]	01 100	44	4C	54	5C	64	6C	74	7C
disp8[EBP]	01 101	45	4D	55	5D	65	6D	75	7D
disp8[ESI]	01 110	46	4E	56	5E	66	6E	76	7E
disp8[EDI]	01 111	47	4F	57	5F	67	6F	77	7F
disp32[EAX]	10 000	80	88	90	98	A0	A8	B0	B8
disp32[ECX]	10 001	81	89	91	99	A1	A9	B1	B9
disp32[EDX]	10 010	82	8A	92	9A	A2	AA	B2	BA
disp32[EBX]	10 011	83	8B	93	9B	A3	AB	B3	BB
disp32[--] [--]	10 100	84	8C	94	9C	A4	AC	B4	BC
disp32[EBP]	10 101	85	8D	95	9D	A5	AD	B5	BD
disp32[ESI]	10 110	86	8E	96	9E	A6	AE	B6	BE
disp32[EDI]	10 111	87	8F	97	9F	A7	AF	B7	BF
EAX/AX/AL	11 000	C0	C8	D0	D8	E0	E8	F0	F8
ECX/CX/CL	11 001	C1	C9	D1	D9	E1	E9	F1	F9
EDX/DX/DL	11 010	C2	CA	D2	DA	E2	EA	F2	FA
EBX/BX/BL	11 011	C3	CB	D3	DB	E3	EB	F3	FB
ESP/SP/AH	11 100	C4	CC	D4	DC	E4	EC	F4	FC
EBP/BP/CH	11 101	C5	CD	D5	DD	E5	ED	F5	FD
ESI/SI/DH	11 110	C6	CE	D6	DE	E6	EE	F6	FE
EDI/DI/BH	11 111	C7	CF	D7	DF	E7	EF	F7	FF

NOTES:

1. [--] [--] means a SIB follows the ModR/M byte.
2. disp8 denotes a 32-bit displacement following the SIB bit, to be sign-extended and added to the Index.
3. disp32 denotes a 32-bit displacement following the SIB bit, to be added to the Index.

[32-Bit Addressing Forms with SIB Table A-3]

r32		EAX	ECX	EDX	EBX	ESP	[*]	ESI	EDI
Base =		0	1	2	3	4	5	6	7
Base =		000	001	010	011	100	101	110	111
	SS Index	ModR/M Values in Hexadecimal							
[EAX]	00 000	00	01	02	03	04	05	06	07
[ECX]	00 001	08	09	0A	0B	0C	0D	0E	0F
[EDX]	00 010	10	11	12	13	14	15	16	17
[EBX]	00 011	18	19	1A	1B	1C	1D	1E	1F
none	00 100	20	21	22	23	24	25	26	27
[EBP]	00 101	28	29	2A	2B	2C	2D	2E	2F
[ESI]	00 110	30	31	32	33	34	35	36	37
[EDI]	00 111	38	39	3A	3B	3C	3D	3E	3F
[EAX*2]	01 000	40	41	42	43	44	45	46	47
[ECX*2]	01 001	48	49	4A	4B	4C	4D	4E	4F
[ECX*2]	01 010	50	51	52	53	54	55	56	57
[EBX*2]	01 011	58	59	5A	5B	5C	5D	5E	5F
none	01 100	60	61	62	63	64	65	66	67
[EBP*2]	01 101	68	69	6A	6B	6C	6D	6E	6F
[ESI*2]	01 110	70	71	72	73	74	75	76	77
[EDI*2]	01 111	78	79	7A	7B	7C	7D	7E	7F
[EAX*4]	10 000	80	81	82	83	84	85	86	87
[ECX*4]	10 001	88	89	8A	8B	8C	8D	8E	8F
[EDX*4]	10 010	90	91	92	93	94	95	96	97
[EBX*4]	10 011	98	89	9A	9B	9C	9D	9E	9F
none	10 100	A0	A1	A2	A3	A4	A5	A6	A7
[EBP*4]	10 101	A8	A9	AA	AB	AC	AD	AE	AF
[ESI*4]	10 110	B0	B1	B2	B3	B4	B5	B6	B7
[EDI*4]	10 111	B8	B9	BA	BB	BC	BD	BE	BF
[EAX*8]	11 000	C0	C1	C2	C3	C4	C5	C6	C7
[ECX*8]	11 001	C8	C9	CA	CB	CC	CD	CE	CF
[EDX*8]	11 010	D0	D1	D2	D3	D4	D5	D6	D7
[EBX*8]	11 011	D8	D9	DA	DB	DC	DD	DE	DF
none	11 100	E0	E1	E2	E3	E4	E5	E6	E7
[EBP*8]	11 101	E8	E9	EA	EB	EC	ED	EE	EF
[ESI*8]	11 110	F0	F1	F2	F3	F4	F5	F6	F7
[EDI*8]	11 111	F8	F9	FA	FB	FC	FD	FE	FF

NOTES: [*] means a disp32 with no base if MOD is 00, [ESP] otherwise. This provides following addressing modes :

disp32[index] (MOD=00)
disp8[EBP][Index] (MOD=01)
disp32[EBP][Index] (MOD=10)

By looking at these tables it is difficult to understand the decoding. I promise U, these Tables and Abbreviations is enough to decode the Instructions. For ur better understanding i will go thru step by step.

First i will translate the above tables into another Table format. So that u can refer below tables to decode the instruction. These tables for looking and decoding it will easier compare to above tables. I will try to make the decoding easier. I will translate three tables One-Byte, ModR/M and SIB.

[One-Byte Opcode Translated Table A-4]

One-Byte Opcode	Mnemonic Identifer
00	ADD { Eb, Gb }
01	ADD { Ev, Gv }
02	ADD { Gb, Eb }
03	ADD { Gv, Ev }
04	ADD { AL, Ib }
05	ADD { eAX, Iv }
06	PUSH { ES }
07	POP { ES }
08	OR { Eb, Gb }
09	OR { Ev, Gv }
0A	OR { Gb, Eb }
0B	OR { Gv, Ev }
0C	OR { AL, Ib }
0D	OR { eAX, Iv }
0E	PUSH { CS }
0F	2- byte { Escape }
10	ADC { Eb, Gb }
11	ADC { Ev, Gv }
12	ADC { Gb, Eb }
13	ADC { Gv, Ev }
14	ADC { AL, Ib }

15	ADC { eAX, Iv }
16	PUSH { SS }
17	POP { SS }
18	SBB { Eb, Gb }
19	SBB { Ev, Gv }
1A	SBB { Gb, Eb }
1B	SBB { Gv, Ev }
1C	SBB { AL, Ib }
1D	SBB { eAX, Iv }
1E	PUSH { DS }
1F	POP { DS }
20	AND { Eb, Gb }
21	AND { Ev, Gv }
22	AND { Gb, Eb }
23	AND { Gv, Ev }
24	AND { AL, Ib }
25	AND { eAX, Iv }
26	SEG { =ES }
27	DAA
28	SUB { Eb, Gb }
29	SUB { Ev, Gv }
2A	SUB { Gb, Eb }
2B	SUB { Gv, Ev }
2C	SUB { AL, Ib }
2D	SUB { eAX, Iv }
2E	SEG { =CS }
2F	DAS
30	XOR { Eb, Gb }
31	XOR { Ev, Gv }

32	XOR { Gb, Eb }
33	XOR { Gv, Ev }
34	XOR { AL, Ib }
35	XOR { eAX, Iv }
36	SEG { =SS }
37	AAA
38	CMP { Eb, Gb }
39	CMP { Ev, Gv }
3A	CMP { Gb, Eb }
3B	CMP { Gv, Ev }
3C	CMP { AL, Ib }
3D	CMP { eAX, Iv }
3E	SEG { =DS }
3F	AAS
40	INC { eAX }
41	INC { eCX }
42	INC { eDX }
43	INC { eBX }
44	INC { eSP }
45	INC { eBP }
46	INC { eSI }
47	INC { eDI }
48	DEC { eAX }
49	DEC { eCX }
4A	DEC { eDX }
4B	DEC { eBX }
4C	DEC { eSP }
4D	DEC { eBP }

4E	DEC { eSI }
4F	DEC { eDI }
50	PUSH { eAX }
51	PUSH { eCX }
52	PUSH { eDX }
53	PUSH { eBX }
54	PUSH { eSP }
55	PUSH { eBP }
56	PUSH { eSI }
57	PUSH { eDI }
58	POP { eAX }
59	POP { eCX }
5A	POP { eDX }
5B	POP { eBX }
5C	POP { eSP }
5D	POP { eBP }
5E	POP { eSI }
5F	POP { eDI }
60	PUSHA
61	POPA
62	BOUND { Gv, Ma }
63	ARPL { Ew, Rw }
64	SEG { =FS }
65	SEG { =GS }
66	Operand { Size }
67	Address { Size }
68	PUSH { Ib }
69	IMUL { Gv,Ev,Iv }
6A	PUSH { Ib }

6B	IMUL { Gv,Ev,Iv }
6C	INSB { Yb, DX }
6D	INSW/D { Yb, DX }
6E	OUTSB { Dx, Xb }
6F	OUTSW/D { Dx, Xv }
70	JO { Ib }
71	JNO { Ib }
72	JB { Ib }
73	JNB { Ib }
74	JZ { Ib }
75	JNZ { Ib }
76	JBE { Ib }
77	JNBE { Ib }
78	JS { Ib }
79	JNS { Ib }
7A	JP { Ib }
7B	JNP { Ib }
7C	JL { Ib }
7D	JNL { Ib }
7E	JLE { Ib }
7F	JNLE { Ib }
80	Immediate Grp1 { Eb, Ib }
81	Immediate Grp1 { Ev, Iv }
82	-Not Defined-
83	Grp1 { Ev, Iv }
84	TEST { Eb, Gb }
85	TEST { Ev, Gv }
86	XCHG { Eb, Gb }

87	XCHG { Ev, Gv }
88	MOV { Eb, Gb }
89	MOV { Ev, Gv }
8A	MOV { Gb, Eb }
8B	MOV { Gv, Ev }
8C	MOV { Ew, Sw }
8D	LEA { Gv, M }
8E	MOV { Sw, Ew }
8F	POP { Ev }
90	NOP
91	XCHG { eAX, eCX }
92	XCHG { eAX, eDX }
93	XCHG { eAX, eBX }
94	XCHG { eAX, eSP }
95	XCHG { eAX, eBP }
96	XCHG { eAX, eSI }
97	XCHG { eAX, eDI }
98	CBW
99	CWD
9A	CALL { Ap }
9B	WAIT
9C	PUSH { Fv }
9D	POPF { Fv }
9E	SAHF
9F	LAHF
A0	MOV { AL,Ob }
A1	MOV { eAX, Ov }
A2	MOV { Ob, AL }
A3	MOV { Ov, eAX }

A4	MOVSB { Xb, Yb }
A5	MOVSW/D { Xv, Yv }
A6	CMPSB { Xb, Yb }
A7	CMPSW/D { Xv, Yv }
A8	TEST { AL, Ib }
A9	TEST { eAX, Iv }
AA	STOSB { Yb, AL }
AB	STOSW/D { Yv, eAX }
AC	LODSB { AL, Xb }
AD	LODSW/D { eAX, Xv }
AE	SCASB { AL, Xb }
AF	SCASW/D { eAX, Xv }
B0	MOV { AL, Ib }
B1	MOV { CL, Ib }
B2	MOV { DL, Ib }
B3	MOV { BL, Ib }
B4	MOV { AH, Ib }
B5	MOV { CH, Ib }
B6	MOV { DH, Ib }
B7	MOV { BH, Ib }
B8	MOV { eAX, Iv }
B9	MOV { eCX, Iv }
BA	MOV { eDX, Iv }
BB	MOV { eBX, Iv }
BC	MOV { eSP, Iv }
BD	MOV { eBP, Iv }
BE	MOV { eSI, Iv }
BF	MOV { eDI, Iv }

C0	Shift Grp2 { Eb, Ib }
C1	Shift Grp2 { Ev, Iv }
C2	RET near { Iw }
C3	RET
C4	LES { Gv, Mp }
C5	LDS { Gv, Mp }
C6	MOV { Eb, Ib }
C7	MOV { Ev, Iv }
C8	ENTER { Iw, Ib }
C9	LEAVE
CA	RET far { IW }
CB	RET
CC	INT { 3 }
CD	INT { Ib }
CE	INTO
CF	IRET
D0	Shif Grp2 { Eb, 1 }
D1	Shif Grp2 { Ev, 1 }
D2	Shif Grp2 { Eb, CL }
D3	Shif Grp2 { Ev, CL }
D4	AAM
D5	AAD
D6	-Not Defined-
D7	XLAT
D8	ESC(Escape to coprocessor instruction set)
D9	ESC(Escape to coprocessor instruction set)
DA	ESC(Escape to coprocessor instruction set)
DB	ESC(Escape to coprocessor instruction set)
DC	ESC(Escape to coprocessor instruction set)

DD	ESC(Escape to coprocessor instruction set)
DE	ESC(Escape to coprocessor instruction set)
DF	ESC(Escape to coprocessor instruction set)
E0	LOOPNE { Jb }
E1	LOOPE { Jb }
E2	LOOP { Jb }
E3	JCXZ { Jb }
E4	IN { AL, Ib }
E5	IN { eAX, Ib }
E6	OUT { Ib, AL }
E7	OUT { Ib, eAX }
E8	CALL { Av }
E9	JNP { near Jv }
EA	JNP { far AP }
EB	JNP { short JB }
EC	IN { AL,DX }
ED	IN { eAX, DX }
EE	OUT { DX, AL }
EF	OUT { DX, eAX }
F0	LOCK
F1	-Not Defined-
F2	REPNE
F3	REP
F4	HLT
F5	CMC
F6	Unary Grp3 { Eb }
F7	Unary Grp3 { Ev }
F8	CLC

F9	STC
FA	CLI
FB	STI
FC	CLD
FD	STD
FE	INC/DEC { Grp4 }
FF	INC/DEC { Grp5 }

[32-Bit Addressing Forms with the ModR/M Byte Translated Table A-5]

ModR/M Byte	Src/Dst, Src/Dst Operand
00	[EAX], EAX/AX/AL
01	[ECX], EAX/AX/AL
02	[EDX], EAX/AX/AL
03	[EBX], EAX/AX/AL
04	[--][--], EAX/AX/AL
05	[disp32], EAX/AX/AL
06	[ESI], EAX/AX/AL
07	[EDI], EAX/AX/AL
08	[EAX], ECX/CX/CL
09	[ECX], ECX/CX/CL
0A	[EDX], ECX/CX/CL
0B	[EBX], ECX/CX/CL
0C	[--][--], ECX/CX/CL
0D	[disp32], ECX/CX/CL
0E	[ESI], ECX/CX/CL
0F	[EDI], ECX/CX/CL
10	[EAX], EDX/DX/DL
11	[ECX], EDX/DX/DL
12	[EDX], EDX/DX/DL

13	[EBX], EDX/DX/DL
14	[--][--], EDX/DX/DL
15	[disp32], EDX/DX/DL
16	[ESI], EDX/DX/DL
17	[EDI], EDX/DX/DL
18	[EAX], EBX/BX/BL
19	[ECX], EBX/BX/BL
1A	[EDX], EBX/BX/BL
1B	[EBX], EBX/BX/BL
1C	[--][--], EBX/BX/BL
1D	[disp32], EBX/BX/BL
1E	[ESI], EBX/BX/BL
1F	[EDI], EBX/BX/BL
20	[EAX], ESP/SP/AH
21	[ECX], ESP/SP/AH
22	[EDX], ESP/SP/AH
23	[EBX], ESP/SP/AH
24	[--][--], ESP/SP/AH
25	[disp32], ESP/SP/AH
26	[ESI], ESP/SP/AH
27	[EDI], ESP/SP/AH
28	[EAX], EBP/BP/CH
29	[ECX], EBP/BP/CH
2A	[EDX], EBP/BP/CH
2B	[EBX], EBP/BP/CH
2C	[--][--], EBP/BP/CH
2D	[disp32], EBP/BP/CH
2E	[ESI], EBP/BP/CH

2F	[EDI], EBP/BP/CH
30	[EAX], ESI/SI/DH
31	[ECX], ESI/SI/DH
32	[EDX], ESI/SI/DH
33	[EBX], ESI/SI/DH
34	[--][--], ESI/SI/DH
35	[disp32], ESI/SI/DH
36	[ESI], ESI/SI/DH
37	[EDI], ESI/SI/DH
38	[EAX], EDI/DI/BH
39	[ECX], EDI/DI/BH
3A	[EDX], EDI/DI/BH
3B	[EBX], EDI/DI/BH
3C	[--][--], EDI/DI/BH
3D	[disp32], EDI/DI/BH
3E	[ESI], EDI/DI/BH
3F	[EDI], EDI/DI/BH
40	[disp8+EAX], EAX/AX/AL
41	[disp8+ECX], EAX/AX/AL
42	[disp8+EDX], EAX/AX/AL
43	[disp8+EBX], EAX/AX/AL
44	[disp8+[--][--]], EAX/AX/AL
45	[disp8+EBP], EAX/AX/AL
46	[disp8+ESI], EAX/AX/AL
47	[disp8+EDI], EAX/AX/AL
48	[disp8+EAX], ECX/CX/CL
49	[disp8+ECX], ECX/CX/CL
4A	[disp8+EDX], ECX/CX/CL
4B	[disp8+EBX], ECX/CX/CL

4C	[disp8+[--][--]], ECX/CX/CL
4D	[disp8+EBP], ECX/CX/CL
4E	[disp8+ESI], ECX/CX/CL
4F	[disp8+EDI], ECX/CX/CL
50	[disp8+EAX], EDX/DX/DL
51	[disp8+ECX], EDX/DX/DL
52	[disp8+EDX], EDX/DX/DL
53	[disp8+EBX], EDX/DX/DL
54	[disp8+[--][--]], EDX/DX/DL
55	[disp8+EBP], EDX/DX/DL
56	[disp8+ESI], EDX/DX/DL
57	[disp8+EDI], EDX/DX/DL
58	[disp8+EAX], EBX/BX/BL
59	[disp8+ECX], EBX/BX/BL
5A	[disp8+EDX], EBX/BX/BL
5B	[disp8+EBX], EBX/BX/BL
5C	[disp8+[--][--]], EBX/BX/BL
5D	[disp8+EBP], EBX/BX/BL
5E	[disp8+ESI], EBX/BX/BL
5F	[disp8+EDI], EBX/BX/BL
60	[disp8+EAX], ESP/SP/AH
61	[disp8+ECX], ESP/SP/AH
62	[disp8+EDX], ESP/SP/AH
63	[disp8+EBX], ESP/SP/AH
64	[disp8+[--][--]], ESP/SP/AH
65	[disp8+EBP], ESP/SP/AH
66	[disp8+ESI], ESP/SP/AH
67	[disp8+EDI], ESP/SP/AH

68	[disp8+EAX], EBP/BP/CH
69	[disp8+ECX], EBP/BP/CH
6A	[disp8+EDX], EBP/BP/CH
6B	[disp8+EBX], EBP/BP/CH
6C	[disp8+[--][--]], EBP/BP/CH
6D	[disp8+EBP], EBP/BP/CH
6E	[disp8+ESI], EBP/BP/CH
6F	[disp8+EDI], EBP/BP/CH
70	[disp8+EAX], ESI/SI/DH
71	[disp8+ECX], ESI/SI/DH
72	[disp8+EDX], ESI/SI/DH
73	[disp8+EBX], ESI/SI/DH
74	[disp8+[--][--]], ESI/SI/DH
75	[disp8+EBP], ESI/SI/DH
76	[disp8+ESI], ESI/SI/DH
77	[disp8+EDI], ESI/SI/DH
78	[disp8+EAX], EDI/DI/BH
79	[disp8+ECX], EDI/DI/BH
7A	[disp8+EDX], EDI/DI/BH
7B	[disp8+EBX], EDI/DI/BH
7C	[disp8+[--][--]], EDI/DI/BH
7D	[disp8+EBP], EDI/DI/BH
7E	[disp8+ESI], EDI/DI/BH
7F	[disp8+EDI], EDI/DI/BH
80	[disp32+EAX], EAX/AX/AL
81	[disp32+ECX], EAX/AX/AL
82	[disp32+EDX], EAX/AX/AL
83	[disp32+EBX], EAX/AX/AL
84	[disp32+[--][--]], EAX/AX/AL

85	[disp32+EBP], EAX/AX/AL
86	[disp32+ESI], EAX/AX/AL
87	[disp32+EDI], EAX/AX/AL
88	[disp32+EAX], ECX/CX/CL
89	[disp32+ECX], ECX/CX/CL
8A	[disp32+EDX], ECX/CX/CL
8B	[disp32+EBX], ECX/CX/CL
8C	[disp32+[--][--]], ECX/CX/CL
8D	[disp32+EBP], ECX/CX/CL
8E	[disp32+ESI], ECX/CX/CL
8F	[disp32+EDI], ECX/CX/CL
90	[disp32+EAX], EDX/DX/DL
91	[disp32+ECX], EDX/DX/DL
92	[disp32+EDX], EDX/DX/DL
93	[disp32+EBX], EDX/DX/DL
94	[disp32+[--][--]], EDX/DX/DL
95	[disp32+EBP], EDX/DX/DL
96	[disp32+ESI], EDX/DX/DL
97	[disp32+EDI], EDX/DX/DL
98	[disp32+EAX], EBX/BX/BL
99	[disp32+ECX], EBX/BX/BL
9A	[disp32+EDX], EBX/BX/BL
9B	[disp32+EBX], EBX/BX/BL
9C	[disp32+[--][--]], EBX/BX/BL
9D	[disp32+EBP], EBX/BX/BL
9E	[disp32+ESI], EBX/BX/BL
9F	[disp32+EDI], EBX/BX/BL
A0	[disp32+EAX], ESP/SP/AH

A1	[disp32+ECX], ESP/SP/AH
A2	[disp32+EDX], ESP/SP/AH
A3	[disp32+EBX], ESP/SP/AH
A4	[disp32+[--][--]], ESP/SP/AH
A5	[disp32+EBP], ESP/SP/AH
A6	[disp32+ESI], ESP/SP/AH
A7	[disp32+EDI], ESP/SP/AH
A8	[disp32+EAX], EBP/BP/CH
A9	[disp32+ECX], EBP/BP/CH
AA	[disp32+EDX], EBP/BP/CH
AB	[disp32+EBX], EBP/BP/CH
AC	[disp32+[--][--]], EBP/BP/CH
AD	[disp32+EBP], EBP/BP/CH
AE	[disp32+ESI], EBP/BP/CH
AF	[disp32+EDI], EBP/BP/CH
B0	[disp32+EAX], ESI/SI/DH
B1	[disp32+ECX], ESI/SI/DH
B2	[disp32+EDX], ESI/SI/DH
B3	[disp32+EBX], ESI/SI/DH
B4	[disp32+[--][--]], ESI/SI/DH
B5	[disp32+EBP], ESI/SI/DH
B6	[disp32+ESI], ESI/SI/DH
B7	[disp32+EDI], ESI/SI/DH
B8	[disp32+EAX], EDI/DI/BH
B9	[disp32+ECX], EDI/DI/BH
BA	[disp32+EDX], EDI/DI/BH
BB	[disp32+EBX], EDI/DI/BH
BC	[disp32+[--][--]], EDI/DI/BH
BD	[disp32+EBP], EDI/DI/BH

BE	[disp32+ESI], EDI/DI/BH
BF	[disp32+EDI], EDI/DI/BH
C0	EAX/AX/AL, EAX/AX/AL
C1	ECX/CX/CL, EAX/AX/AL
C2	EDX/DX/DL, EAX/AX/AL
C3	EBX/BX/BL, EAX/AX/AL
C4	ESP/SP/AH, EAX/AX/AL
C5	EBP/BP/CH, EAX/AX/AL
C6	ESI/SI/DH, EAX/AX/AL
C7	EDI/DI/BH, EAX/AX/AL
C8	EAX/AX/AL, ECX/CX/CL
C9	ECX/CX/CL, ECX/CX/CL
CA	EDX/DX/DL, ECX/CX/CL
CB	EBX/BX/BL, ECX/CX/CL
CC	ESP/SP/AH, ECX/CX/CL
CD	EBP/BP/CH, ECX/CX/CL
CE	ESI/SI/DH, ECX/CX/CL
CF	EDI/DI/BH, ECX/CX/CL
D0	EAX/AX/AL, EDX/DX/DL
D1	ECX/CX/CL, EDX/DX/DL
D2	EDX/DX/DL, EDX/DX/DL
D3	EBX/BX/BL, EDX/DX/DL
D4	ESP/SP/AH, EDX/DX/DL
D5	EBP/BP/CH, EDX/DX/DL
D6	ESI/SI/DH, EDX/DX/DL
D7	EDI/DI/BH, EDX/DX/DL
D8	EAX/AX/AL, EBX/BX/BL
D9	ECX/CX/CL, EBX/BX/BL

DA	EDX/DX/DL, EBX/BX/BL
DB	EBX/BX/BL, EBX/BX/BL
DC	ESP/SP/AH, EBX/BX/BL
DD	EBP/BP/CH, EBX/BX/BL
DE	ESI/SI/DH, EBX/BX/BL
DF	EDI/DI/BH, EBX/BX/BL
E0	EAX/AX/AL, ESP/SP/AH
E1	ECX/CX/CL, ESP/SP/AH
E2	EDX/DX/DL, ESP/SP/AH
E3	EBX/BX/BL, ESP/SP/AH
E4	ESP/SP/AH, ESP/SP/AH
E5	EBP/BP/CH, ESP/SP/AH
E6	ESI/SI/DH, ESP/SP/AH
E7	EDI/DI/BH, ESP/SP/AH
E8	EAX/AX/AL, EBP/BP/CH
E9	ECX/CX/CL, EBP/BP/CH
EA	EDX/DX/DL, EBP/BP/CH
EB	EBX/BX/BL, EBP/BP/CH
EC	ESP/SP/AH, EBP/BP/CH
ED	EBP/BP/CH, EBP/BP/CH
EE	ESI/SI/DH, EBP/BP/CH
EF	EDI/DI/BH, EBP/BP/CH
F0	EAX/AX/AL, ESI/SI/DH
F1	ECX/CX/CL, ESI/SI/DH
F2	EDX/DX/DL, ESI/SI/DH
F3	EBX/BX/BL, ESI/SI/DH
F4	ESP/SP/AH, ESI/SI/DH
F5	EBP/BP/CH, ESI/SI/DH
F6	ESI/SI/DH, ESI/SI/DH

F7	EDI/DI/BH, ESI/SI/DH
F8	EAX/AX/AL, EDI/DI/BH
F9	ECX/CX/CL, EDI/DI/BH
FA	EDX/DX/DL, EDI/DI/BH
FB	EBX/BX/BL, EDI/DI/BH
FC	ESP/SP/AH, EDI/DI/BH
FD	EBP/BP/CH, EDI/DI/BH
FE	ESI/SI/DH, EDI/DI/BH
FF	EDI/DI/BH, EDI/DI/BH

[32-Bit Addressing Forms with SIB byte Translated Table A-6]

SIB Byte	Src/Dst, Src/Dst Operand
00	[EAX], EAX
01	[EAX], ECX
02	[EAX], EDX
03	[EAX], EBX
04	[EAX], ESP
05	[EAX], [*]
06	[EAX], ESI
07	[EAX], EDI
08	[ECX], EAX
09	[ECX], ECX
0A	[ECX], EDX
0B	[ECX], EBX
0C	[ECX], ESP
0D	[ECX], [*]
0E	[ECX], ESI
0F	[ECX], EDI

10	[EDX], EAX
11	[EDX], ECX
12	[EDX], EDX
13	[EDX], EBX
14	[EDX], ESP
15	[EDX], [*]
16	[EDX], ESI
17	[EDX], EDI
18	[EBX], EAX
19	[EBX], ECX
1A	[EBX], EDX
1B	[EBX], EBX
1C	[EBX], ESP
1D	[EBX], [*]
1E	[EBX], ESI
1F	[EBX], EDI
20	none, EAX
21	none, ECX
22	none, EDX
23	none, EBX
24	none, ESP
25	none, [*]
26	none, ESI
27	none, EDI
28	[EBP], EAX
29	[EBP], ECX
2A	[EBP], EDX
2B	[EBP], EBX
2C	[EBP], ESP

2D	[EBP], [*]
2E	[EBP], ESI
2F	[EBP], EDI
30	[ESI], EAX
31	[ESI], ECX
32	[ESI], EDX
33	[ESI], EBX
34	[ESI], ESP
35	[ESI], [*]
36	[ESI], ESI
37	[ESI], EDI
38	[EDI], EAX
39	[EDI], ECX
3A	[EDI], EDX
3B	[EDI], EBX
3C	[EDI], ESP
3D	[EDI], [*]
3E	[EDI], ESI
3F	[EDI], EDI
40	[EAX*2], EAX
41	[EAX*2], ECX
42	[EAX*2], EDX
43	[EAX*2], EBX
44	[EAX*2], ESP
45	[EAX*2], [*]
46	[EAX*2], ESI
47	[EAX*2], EDI
48	[ECX*2], EAX

49	[ECX*2], ECX
4A	[ECX*2], EDX
4B	[ECX*2], EBX
4C	[ECX*2], ESP
4D	[ECX*2], [*]
4E	[ECX*2], ESI
4F	[ECX*2], EDI
50	[EDX*2], EAX
51	[EDX*2], ECX
52	[EDX*2], EDX
53	[EDX*2], EBX
54	[EDX*2], ESP
55	[EDX*2], [*]
56	[EDX*2], ESI
57	[EDX*2], EDI
58	[EBX*2], EAX
59	[EBX*2], ECX
5A	[EBX*2], EDX
5B	[EBX*2], EBX
5C	[EBX*2], ESP
5D	[EBX*2], [*]
5E	[EBX*2], ESI
5F	[EBX*2], EDI
60	none, EAX
61	none, ECX
62	none, EDX
63	none, EBX
64	none, ESP
65	none, [*]

66	none, ESI
67	none, EDI
68	[EBP*2], EAX
69	[EBP*2], ECX
6A	[EBP*2], EDX
6B	[EBP*2], EBX
6C	[EBP*2], ESP
6D	[EBP*2], [*]
6E	[EBP*2], ESI
6F	[EBP*2], EDI
70	[ESI*2], EAX
71	[ESI*2], ECX
72	[ESI*2], EDX
73	[ESI*2], EBX
74	[ESI*2], ESP
75	[ESI*2], [*]
76	[ESI*2], ESI
77	[ESI*2], EDI
78	[EDI*2], EAX
79	[EDI*2], ECX
7A	[EDI*2], EDX
7B	[EDI*2], EBX
7C	[EDI*2], ESP
7D	[EDI*2], [*]
7E	[EDI*2], ESI
7F	[EDI*2], EDI
80	[EAX*4], EAX
81	[EAX*4], ECX

82	[EAX*4], EDX
83	[EAX*4], EBX
84	[EAX*4], ESP
85	[EAX*4], [*]
86	[EAX*4], ESI
87	[EAX*4], EDI
88	[ECX*4], EAX
89	[ECX*4], ECX
8A	[ECX*4], EDX
8B	[ECX*4], EBX
8C	[ECX*4], ESP
8D	[ECX*4], [*]
8E	[ECX*4], ESI
8F	[ECX*4], EDI
90	[EDX*4], EAX
91	[EDX*4], ECX
92	[EDX*4], EDX
93	[EDX*4], EBX
94	[EDX*4], ESP
95	[EDX*4], [*]
96	[EDX*4], ESI
97	[EDX*4], EDI
98	[EBX*4], EAX
99	[EBX*4], ECX
9A	[EBX*4], EDX
9B	[EBX*4], EBX
9C	[EBX*4], ESP
9D	[EBX*4], [*]
9E	[EBX*4], ESI

9F	[EBX*4], EDI
A0	none, EAX
A1	none, ECX
A2	none, EDX
A3	none, EBX
A4	none, ESP
A5	none, [*]
A6	none, ESI
A7	none, EDI
A8	[EBP*4], EAX
A9	[EBP*4], ECX
AA	[EBP*4], EDX
AB	[EBP*4], EBX
AC	[EBP*4], ESP
AD	[EBP*4], [*]
AE	[EBP*4], ESI
AF	[EBP*4], EDI
B0	[ESI*4], EAX
B1	[ESI*4], ECX
B2	[ESI*4], EDX
B3	[ESI*4], EBX
B4	[ESI*4], ESP
B5	[ESI*4], [*]
B6	[ESI*4], ESI
B7	[ESI*4], EDI
B8	[EDI*4], EAX
B9	[EDI*4], ECX
BA	[EDI*4], EDX

BB	[EDI*4], EBX
BC	[EDI*4], ESP
BD	[EDI*4], [*]
BE	[EDI*4], ESI
BF	[EDI*4], EDI
C0	[EAX*8], EAX
C1	[EAX*8], ECX
C2	[EAX*8], EDX
C3	[EAX*8], EBX
C4	[EAX*8], ESP
C5	[EAX*8], [*]
C6	[EAX*8], ESI
C7	[EAX*8], EDI
C8	[ECX*8], EAX
C9	[ECX*8], ECX
CA	[ECX*8], EDX
CB	[ECX*8], EBX
CC	[ECX*8], ESP
CD	[ECX*8], [*]
CE	[ECX*8], ESI
CF	[ECX*8], EDI
D0	[EDX*8], EAX
D1	[EDX*8], ECX
D2	[EDX*8], EDX
D3	[EDX*8], EBX
D4	[EDX*8], ESP
D5	[EDX*8], [*]
D6	[EDX*8], ESI
D7	[EDX*8], EDI

D8	[EBX*8], EAX
D9	[EBX*8], ECX
DA	[EBX*8], EDX
DB	[EBX*8], EBX
DC	[EBX*8], ESP
DD	[EBX*8], [*]
DE	[EBX*8], ESI
DF	[EBX*8], EDI
E0	none, EAX
E1	none, ECX
E2	none, EDX
E3	none, EBX
E4	none, ESP
E5	none, [*]
E6	none, ESI
E7	none, EDI
E8	[EBP*8], EAX
E9	[EBP*8], ECX
EA	[EBP*8], EDX
EB	[EBP*8], EBX
EC	[EBP*8], ESP
ED	[EBP*8], [*]
EE	[EBP*8], ESI
EF	[EBP*8], EDI
F0	[ESI*8], EAX
F1	[ESI*8], ECX
F2	[ESI*8], EDX
F3	[ESI*8], EBX

F4	[ESI*8], ESP
F5	[ESI*8], [*]
F6	[ESI*8], ESI
F7	[ESI*8], EDI
F8	[EDI*8], EAX
F9	[EDI*8], ECX
FA	[EDI*8], EDX
FB	[EDI*8], EBX
FC	[EDI*8], ESP
FD	[EDI*8], [*]
FE	[EDI*8], ESI
FF	[EDI*8], EDI

Now, take a look at decoding in very easy way. It is like substituting the simple Math formula ;-).

```
1) Opcode          :  00

    Example            :  00 D8
    Decoding method    :
     ADD { Eb, Gb }                                       ( Table A-4 )
     ADD { EAX/AX/AL, EBX/BX/BL }                         ( Table A-5 )
    Final instruction  :  ADD AL, BL

    Example            : 00 44 BB 40
    Decoding method    :
     ADD { Eb, Gb }                                       ( Table A-4 )
     ADD { [disp8+[--][--]], EAX/AX/AL }                  ( Table A-5 )
     ADD { [disp8+([EDI*4],EBX]), EAX/AX/AL }             ( Table A-6 )
     ADD { [disp8+EDI*4+EBX], EAX/AX/AL }
    Final instruction  : ADD byte ptr [40+EDI*4+EBX], AL

    Note : [--][--] indicates u have to consider next byte as SIB Byte
           and refer SIB table for forming complete instruction.

   2) Opcode           : 01

      Example          : 01 FC
      Decoding method  :
       ADD { Ev, Gv }                                     ( Table A-4 )
       ADD { ESP/SP/AH, EDI/DI/BH }                       ( Table A-5 )
      Final instruction: ADD ESP, EDI

      Example          : 01 05 08 BD 41 00
      Decoding method  :
```

```
    ADD { Ev, Gv }                                      ( Table A-4 )
    ADD { [disp32], EAX/AX/AL }                         ( Table A-5 )
   Final instruction: ADD dword ptr ds:[ 0041BD08 ], EAX

2) Opcode           : 02

   Example          : 02 E6
   Decoding method  :
    ADD { Gb, Eb }                                      ( Table A-4 )
    ADD { ESI/SI/DH, ESP/SP/AH }                        ( Table A-5 )
   Final instruction: ADD DH, AH

   Example          : 02 6C 73 74
   Decoding method  :
    ADD { Gb, Eb }                                      ( Table A-4 )
    ADD { EBP/BP/CH, [disp8+[--][--]] }                 ( Table A-5 )
    ADD { EBP/BP/CH, [disp8+([ESI*2], EBX)]  }          ( Table A-6 )
    ADD { EBP/BP/CH, [disp8+ESI*2+EBX] }

   Final instruction: ADD CH, byte ptr [74+ESI*2+EBX]

   Note : Eb, Gb means u can substitue directly from translated table.
          Gb, Eb means just reverse the content after getting from
          translated table, i mean Reverse source and destination.



3) Opcode           : 03

   Example          : 03 C8
   Decoding method  :
    ADD { Gv, Ev }                                      ( Table A-4 )
    ADD { ECX/CX/CL, EAX/AX/AL }                        ( Table A-5 )
   Final instruction: ADD ECX, EAX

   Example          : 03 C8
   Decoding method  : 03 6C 24 10
    ADD { Gv, Ev }                                      ( Table A-4 )
    ADD { EBP/BP/CH, [disp8+[--][--]] }                 ( Table A-5 )
    ADD { EBP/BP/CH, [disp8+(none, ESP)] }              ( Table A-6 )
    ADD { EBP/BP/CH, [disp8+ESP] }

   Final instruction: ADD EBP, dowrd ptr [10+ESP]

   Note : 'none' means nothing :-)

4) Opcode           : 04

   Example          : 04 30
   Decoding method  :
    ADD { AL, Ib }                                      ( Table A-4 )
   Final instruction: ADD AL, 30

5) Opcode           : 05

   Example          : 05 00 01 00 00
   Decoding method  :
    ADD { eAX, Iv }                                     ( Table A-4 )
   Final instruction: ADD EAX, 00000100

6) Opcode           : 06

   Example          : 06
   Decoding method  :
    PUSH { ES }                                         ( Table A-4 )
   Final instruction: PUSH ES

7) Opcode           : 07

   Example          : 07
   Decoding method  :
    POP { ES }                                          ( Table A-4 )
   Final instruction: POP ES

8) Opcode           : 08

   Example          : 08 CC
   Decoding method  :
    OR { Eb, Gb }                                       ( Table A-4 )
    OR { ESP/SP/AH, ECX/CX/CL }                         ( Table A-5 )
   Final instruction: OR AH, CL

   Example          : 08 34 40
   Decoding method  :
    OR { Eb, Gb }                                       ( Table A-4 )
    OR { [--][--], ESI/SI/DH }                          ( Table A-5 )
    OR { [([EAX*2],EAX)], ESI/SI/DH }                   ( Table A-6 )
    OR { [EAX*2+EAX], ESI/SI/DH }
   Final instruction: OR [EAX*2+EAX], DH

9) Opcode           : 09

   Example          : 09 C9
   Decoding method  :
    OR { Ev, Gv }                                       ( Table A-4 )
    OR { ECX/CX/CL, ECX/CX/CL }                         ( Table A-5 )
   Final instruction: OR ECX, ECX

   Example          : 09 7C 88 44
   Decoding method  :
    OR { Ev, Gv }                                       ( Table A-4 )
    OR { [disp8+[--][--]], EDI/DI/BH }                  ( Table A-5 )
    OR { [disp8+([ECX*4], EAX)], EDI/DI/BH }            ( Table A-6 )
    OR { [disp8+ECX*4+EAX], EDI/DI/BH }
   Final instruction: OR dword ptr [44+ECX*4+EAX], EDI

10) Opcode          : 0A

    Example         : 0A C0
    Decoding method :
     OR { Gb, Eb }                                      ( Table A-4 )
     OR { EAX/AX/AL, EAX/AX/AL }                        ( Table A-5 )
    Final instruction: OR AL, AL
    Example             : 0A 05 0A 06 0A 0A
    Decoding method     :
     OR { Gb, Eb }                                      ( Table A-4 )
     OR { EAX/AX/AL, [disp32] }                             ( Table A-5 )
    Final instruction: OR AL, byte ptr [ 0A0A060A ]

11) Opcode          : 0B

    Example         : 0B C0
    Decoding method :
     OR { Gv, Ev }                                      ( Table A-4 )
     OR { EAX/AX/AL, EAX/AX/AL }                        ( Table A-5 )
    Final instruction: OR EAX, EAX

    Example             : 0B 34 ED 04 9F 41 00
    Decoding method     :
     OR { Gv, Ev }                                      ( Table A-4 )
     OR { ESI/SI/DH, [--][--] }                         ( Table A-5 )
     OR { ESI/SI/DH, ([EBP*8], [*]) }                   ( Table A-6 )
     OR { ESI/SI/DH, [EBP*8+00419F04]  }
    Final instruction: OR ESI, dword ptr [EBP*8+00419F04]

    Note : [*], refer the 'Note' of Table A-3.

12) Opcode          : 0C

    Example         : 0C 01
    Decoding method :
     OR { AL, Ib }                                      ( Table A-4 )
    Final instruction: OR AL, 01

13) Opcode          : 0D

    Example         : 0D 00 04 00 00
    Decoding method :
     OR { eAX, Iv }                                         ( Table A-4 )
    Final instruction: OR EAX, 00000400

14) Opcode          : 0E

    Example         : 0E
    Decoding method :
     PUSH { CS }                                            ( Table A-4 )
    Final instruction: PUSH CS

15) Opcode          : 0F
    ( Refer Next Artical ;-) )

16) Opcode          : 10

    Example         : 10 DC
    Decoding method :
     ADC { Eb, Gb }                                         ( Table A-4 )
     ADC { ESP/SP/AH, EBX/BX/BL }                           ( Table A-5 )
    Final instruction: ADC AH, BL
    Example         : 10 AD 22 46 23 FF
    Decoding method :
     ADC { Eb, Gb }                                         ( Table A-4 )
     ADC { [disp32+EBP], EBP/BP/CH }                        ( Table A-5 )
     ADC { [FF234622+EBP], CH
    Final instruction: ADC byte ptr [FF234622+EBP], CH

17) Opcode          : 11

    Example         : 11 F5
    Decoding method :
     ADC { Ev, Gv }                                         ( Table A-4 )
     ADC { EBP/BP/CH, ESI/SI/DH }                           ( Table A-5 )
    Final instruction: ADC EBP, ESI

    Example         : 11 8C 78 00 C0 4F D9
    Decoding method :
     ADC { Ev, Gv } ( Table A-4 )
     ADC { [disp32+[--][--]], ECX/CX/CL }                   ( Table A-5 )
     ADC { [disp32+([EDI*2], EAX), ECX/CX/CL }              ( Table A-6 )
     ADC { [disp32+EDI*2+EAX), ECX/CX/CL }
    Final instruction: ADC dword ptr [D94FC000+EDI*2+EAX], ECX

18) Opcode          : 12

    Example         : 12 D3
    Decoding method :
     ADC { Gb, Eb } ( Table A-4 )
     ADC { EDX/DX/DL, EBX/BX/BL }                           ( Table A-5 )
    Final instruction: ADC DL, BL

    Example         : 12 04 12
    Decoding method :
     ADC { Gb, Eb }                                         ( Table A-4 )
     ADC { EAX/AX/AL, [--][--] }                            ( Table A-5 )
     ADC { EAX/AX/AL, ([EDX], EDX) }                        ( Table A-6 )
     ADC { EAX/AX/AL, [EDX+EDX] }
    Final instruction: ADC AL, byte ptr [ EDX+EDX ]

19) Opcode          : 13

    Example         : 13 F8
    Decoding method :
     ADC { Gv, Ev }                                         ( Table A-4 )
     ADC { EDI/DI/BH, EAX/AX/AL }                           ( Table A-5 )
    Final instruction: ADC EDI, EAX

    Example         : 13 54 F5 BF
    Decoding method :
     ADC { Gv, Ev }                                         ( Table A-4 )
     ADC { EDX/DX/DL, [disp8+[--][--]] }                    ( Table A-5 )
     ADC { EDX/DX/DL, [disp8+([ESI*8], [*])] }              ( Table A-6 )
     ADC { EDX/DX/DL, [ESI*8+EBP+disp8] }
     ADC { EDX/DX/DL, [ESI*8+EBP-41] }
    Final instruction: ADC EDX, dword byte ptr [ESI*8+EBP-41]

    Note : If Most significant bit in data is 1 then,
           (disp8 - FF)+1 or (disp32 - FFFFFFFF)+1.

20) Opcode          : 14

    Example         : 14 6C
    Decoding method :
     ADC { AL, Ib }                                         ( Table A-4 )
    Final instruction: ADC AL, 6C

21) Opcode          : 15

    Example         : 15 01 00 00 00
    Decoding method :
        ADC { eAX, Iv }                                     ( Table A-4 )
    Final instruction: ADC EAX, 00000001

22) Opcode          : 16

    Example         : 16
    Decoding method :
     PUSH { SS }                                            ( Table A-4 )
    Final instruction: PUSH SS

23) Opcode      : 17

    Example         : 16
    Decoding method :
     POP { SS }                                             ( Table A-4 )
    Final instruction: POP SS

24) Opcode      : 18

    Example         : 18 D6
    Decoding method :
     SBB { Eb, Gb }                                         ( Table A-4 )
     SBB { ESI/SI/DH, EDX/DX/DL }                           ( Table A-5 )
    Final instruction: SBB DH, DL

    Example         : 18 B4 B0 04 F5 4C 96
    Decoding method :
     SBB { Eb, Gb }                                         ( Table A-4 )
     SBB { [disp32+[--][--]], ESI/SI/DH }                   ( Table A-5 )
     SBB { [disp32+([ESI*4], EAX)], ESI/SI/DH }             ( Table A-6 )
     SBB { [disp32+ESI*4+EAX], ESI/SI/DH }
    Final instruction: SBB byte ptr [ 964CF504+ESI*4+EAX ], DH

25) Opcode      : 19

    Example         : 19 FA
    Decoding method :
     SBB { Ev, Gv }                                         ( Table A-4 )
     SBB { EDX/DX/DL, EDI/DI/BH }                           ( Table A-5 )
    Final instruction: SBB EDX, EDI

    Example         : 19 5C 33 2D
    Decoding method :
     SBB { Ev, Gv }                                         ( Table A-4 )
     SBB { [disp8+[--][--]], EBX/BX/BL }                    ( Table A-5 )
     SBB { [disp8+([ESI], EBX)], EBX/BX/BL }                ( Table A-6 )
     SBB { [disp8+ESI+EBX], EBX/BX/BL }
    Final instruction: SBB dword ptr [2D+ESI+EBX], EBX

26) Opcode      : 1A

    Example         : 1A C0
    Decoding method :
     SBB { Gb, Eb }                                         ( Table A-4 )
     SBB { EAX/AX/AL, EAX/AX/AL }                           ( Table A-5 )
    Final instruction: SBB AL, AL

    Example         : 1A 43 01
    Decoding method :
     SBB { Gb, Eb }                                         ( Table A-4 )
     SBB { EAX/AX/AL, [disp8+EBX] }                         ( Table A-5 )
    Final instruction: SBB AL, byte ptr [01+EBX]

27) Opcode      : 1B

    Example         : 1B F5
    Decoding method :
     SBB { Gv, Ev }                                         ( Table A-4 )
     SBB { ESI/SI/DH, EBP/BP/CH }                           ( Table A-5 )
    Final instruction: SBB ESI, EBP

    Example         : 1B 54 24 14
    Decoding method :
     SBB { Gv, Ev }                                         ( Table A-4 )
     SBB { EDX/DX/DL, [disp8+[--][--]] }                    ( Table A-5 )
     SBB { EDX/DX/DL, [disp8+(none, ESP)] }                 ( Table A-6 )
     SBB { EDX/DX/DL, [disp8+ESP] }
    Final instruction: SBB EDX, dword ptr [14+ESP]

28) Opcode      : 1C

    Example         : 1C AC
    Decoding method :
     SBB { AL, Ib }                                         ( Table A-4 )
    Final instruction: SBB AL, AC

29) Opcode      : 1D

    Example         : 1D 00 00 00 01
    Decoding method :
     SBB { eAX, Iv }                                        ( Table A-4 )
    Final instruction: SBB EAX, 01000000

30) Opcode      : 1E

    Example         : 1E
    Decoding method :
     PUSH { DS }                                            ( Table A-4 )
    Final instruction: PUSH DS

31) Opcode      : 1F
```

```
    Example         : 1F
    Decoding method :
     POP { DS }                                             ( Table A-4 )
    Final instruction: POP DS

32) Opcode      : 20

    Example         : 20 D7
    Decoding method :
     AND { Eb, Gb }                                         ( Table A-4 )
     AND { EDI/DI/BH, EDX/DX/DL }
    Final instruction: AND BH, DL

    Example         : 20 6C 6F 61
    Decoding method :
     AND { Eb, Gb }                                         ( Table A-4 )
     AND { [disp8+[--][--]], EBP/BP/CH }                    ( Table A-5 )
     AND { [disp8+([EBP*2], EDI)], EBP/BP/CH }              ( Table A-6 )
     AND { [disp8+EBP*2+EDI], EBP/BP/CH }
    Final instruction: AND byte ptr [disp8+EBP*2+EDI], CH

33) Opcode      : 21

    Example         : 21 40 00
    Decoding method :
     AND { Ev, Gv }                                         ( Table A-4 )
     AND { [disp8+EAX], EAX/AX/AL }                         ( Table A-5 )
    Final instruction: AND dword ptr [00+EAX], EAX

    Example         : 21 BC B0 C4 00 00 00
    Decoding method :
     AND { Ev, Gv }                                         ( Table A-4 )
     AND { [disp32+[--][--]], EDI/DI/BH }                   ( Table A-5 )
     AND { [disp32+([ESI*4], EAX)], EDI/DI/BH }             ( Table A-6 )
     AND { [disp32+ESI*4+EAX], EDI/DI/BH }
    Final instruction: AND dword ptr [000000C4+ESI*4+EAX], EDI

34) Opcode      : 22

    Example         : 22 FD
    Decoding method :
     AND { Gb, Eb }                                         ( Table A-4 )
     AND { EDI/DI/BH, EBP/BP/CH }                           ( Table A-5 )
    Final instruction: AND BH, CH

    Example         : 22 9C 3D DF 4E AB D0
    Decoding method :
     AND { Gb, Eb }                                         ( Table A-4 )
     AND { EBX/BX/BL, [disp32+[--][--]] }                   ( Table A-5 )
     AND { EBX/BX/BL, [disp32+([EDI],[*])] }                ( Table A-5 )
     AND { EBX/BX/BL, [EDI+EBP+D0AB4EDF] }
    Final instruction: AND BL, byte ptr [EDI+EBP+D0AB4EDF]

35) Opcode      : 23

    Example         : 23 CF
    Decoding method :
     AND { Gv, Ev }                                         ( Table A-4 )
     AND { ECX/CX/CL, EDI/DI/BH }                           ( Table A-5 )
    Final instruction: AND ECX, EDI

    Example         : 23 75 F4
    Decoding method :
     AND { Gv, Ev }                                         ( Table A-4 )
     AND { [disp8+EBP], ESI/SI/DH }                         ( Table A-5 )
    Final instruction: AND byte ptr [ EBP-0C ], DH

36) Opcode      : 24

    Example         : 24 01
    Decoding method :
     AND { AL, Ib }                                         ( Table A-4 )
    Final instruction: AND AL, 01

37) Opcode      : 25

    Example         : 25 FF FF 00 00
    Decoding method :
     AND { eAX, Iv }                                        ( Table A-4 )
    Final instruction: AND EAX, 0000FFFF

38) Opcode      : 26

    Description     : This opcode indicates ES segment register as Prefix
                      for memory locations.

    Example         : 26  ( Segment Prefix )
                      00 43 6F ( Decode like normal instructions )
    Decoding method :
     SEG { =ES }                                            ( Table A-4 )

     ADD { Eb, Gb }                                         ( Table A-4 )
     ADD { [disp8+EBX], EAX/AX/AL }                         ( Table A-5 )
     ADD { [6F+EBX], EAX/AX/AL }

    Final instruction: AND byte ptr ES:[6F+EBX], AL

    Example         : 26 ( Segment Prefix )
                      27 ( Decode like normal instructions )
    Decoding method :
     SEG { =ES }                                            ( Table A-4 )
     DAA                                                    ( Table A-4 )

    Final instruction: DAA

    Note : No need to use Segment as prefix in this case.

39) Opcode      : 27

    Example         : 27
    Decoding method :
     DAA                                                    ( Table A-4 )
    Final instruction: DAA
```

```
40) Opcode      : 28

    Example         : 28 D7
    Decoding method :
     SUB { Eb, Gb }                                      ( Table A-4 )
     SUB { EDI/DI/BH, EDX/DX/DL }                        ( Table A-5 )
    Final instruction: SUB BH, DL

    Example         : 28 14 40
    Decoding method :
     SUB { Eb, Gb }                                      ( Table A-4 )
     SUB { [--][--], EDX/DX/DL }                         ( Table A-5 )
     SUB { ([EAX*2], EAX), EDX/DX/DL }                   ( Table A-6 )
     SUB { [EAX*2+EAX], EDX/DX/DL }

    Final instruction: SUB byte ptr [EAX*2+EAX], DL

41) Opcode      : 29

    Example         : 29 40 00
    Decoding method :
     SUB { Ev, Gv }                                      ( Table A-4 )
     SUB { [disp8+EAX], EAX/AX/AL }                      ( Table A-5 )
    Final instruction: SUB byte ptr [00+EAX], EAX

    Example         : 29 45 EC
    Decoding method :
     SUB { Ev, Gv }                                      ( Table A-4 )
     SUB { [disp8+EBP], EAX/AX/AL }                      ( Table A-5 )
    Final instruction: SUB byte ptr [EBP-14], EAX

42) Opcode      : 2A

    Example         : 2A C1
    Decoding method :
     SUB { Gb, Eb }                                      ( Table A-4 )
     SUB { EAX/AX/AL, ECX/CX/CL }                        ( Table A-5 )
    Final instruction: SUB AL, CL

    Example         : 2A 8C 30 F6 27 00 00
    Decoding method :
     SUB { Gb, Eb }                                      ( Table A-4 )
     SUB { ECX/CX/CL, [disp32+[--][--]] }                ( Table A-5 )
     SUB { ECX/CX/CL, [disp32+([ESI], EAX)] }            ( Table A-6 )
     SUB { ECX/CX/CL, [disp32+ESI+EAX] }
    Final instruction: SUB CL, byte ptr [000027F6+ESI+EAX]

43) Opcode      : 2B

    Example         : 2B C1
    Decoding method :
     SUB { Gv, Ev }                                      ( Table A-4 )
     SUB { EAX/AX/AL, ECX/CX/CL }                        ( Table A-5 )
    Final instruction: SUB EAX, ECX

    Example         : 2B 2B
    Decoding method :
     SUB { Gv, Ev }                                         ( Table A-4 )
     SUB { EBP/BP/CH, [EBX] }                               ( Table A-5 )
    Final instruction: SUB EBP, byte ptr [EBX]

44) Opcode      : 2C

    Example         : 2C 31
    Decoding method :
     SUB { AL, Ib }                                         ( Table A-4 )
    Final instruction: SUB AL, 31

45) Opcode      : 2D

    Example         : 2D 10 BA 41 00
    Decoding method :
     SUB { eAX, Iv }                                        ( Table A-4 )
    Final instruction: SUB EAX, 0041BA10

46) Opcode      : 2E

    Description     : This opcode indicates CS segment register as Prefix
                      for memory locations.

    Example         : 2E ( Segment Prefix )
                      00 3C 70 ( Decode like normal instructions )
    Decoding method :
     SEG { =CS }                                            ( Table A-4 )

     ADD { Eb, Gb }                                         ( Table A-4 )
     AND { [--][--], EDI/DI/BH }                            ( Table A-5 )
     AND { ([ESI*2], EAX), EDI/DI/BH }                      ( Table A-6 )
     AND { [ESI*2+EAX], EDI/DI/BH }
    Final instruction: AND byte ptr CS:[ESI*2+EAX], BH

    Example         : 2E ( Segment Prefix )
                      8B C0 ( Decode like normal instructions )
    Decoding method :
     SEG { =CS }                                            ( Table A-4 )

     MOV { Gv, Ev }                                         ( Table A-4 )
     MOV { EAX/AX/AL, EAX/AX/AL }                           ( Table A-5 )
    Final instruction: MOV EAX, EAX

47) Opcode      : 2F

    Example         : 2F
    Decoding method :
     DAS                                                    ( Table A-4 )
    Final instruction: DAS

48) Opcode      : 30

    Example         : 30 D5
    Decoding method :
     XOR { Eb, Gb }                                         ( Table A-4 )
     XOR { EBP/BP/CH, EDX/DX/DL }                           ( Table A-5 )
    Final instruction: XOR CH, DL

    Example         : 30 84 40 00 20 84 40
    Decoding method :
     XOR { Eb, Gb }                                         ( Table A-4 )
     XOR { [disp32+[--][--]], EAX/AX/AL }                   ( Table A-5 )
     XOR { [disp32+([EAX*2], EAX)], EAX/AX/AL }             ( Table A-6 )
     XOR { [disp32+EAX*2+EAX], EAX/AX/AL }
    Final instruction: XOR byte ptr [40842000+EAX*2+EAX], AL

49) Opcode      : 31

    Example         : 31 F5
    Decoding method :
     XOR { Ev, Gv }                                         ( Table A-4 )
     XOR { EBP/BP/CH, ESI/SI/DH }                           ( Table A-5 )
    Final instruction: XOR EBP, ESI

    Example         : 31 74 79 70
    Decoding method :
     XOR { Ev, Gv }                                         ( Table A-4 )
     XOR { [disp8+[--][--]], ESI/SI/DH }                    ( Table A-5 )
     XOR { [disp8+([EDI*2], ECX)], ESI/SI/DH }              ( Table A-6 )
     XOR { [disp8+EDI*2+ECX], ESI/SI/DH }
    Final instruction: XOR dword ptr [70+EDI*2+ECX], ESI

50) Opcode      : 32

    Example         : 32 C0
    Decoding method :
     XOR { Gb, Eb }                                         ( Table A-4 )
     XOR { EAX/AX/AL, EAX/AX/AL }                           ( Table A-5 )
    Final instruction: XOR AL, AL

    Example         : 32 44 00 CF
    Decoding method :
     XOR { Gb, Eb }                                         ( Table A-4 )
     XOR { EAX/AX/AL, [disp8+[--][--]] }                    ( Table A-5 )
     XOR { EAX/AX/AL, [disp8+([EAX], EAX)] }                ( Table A-6 )
     XOR { EAX/AX/AL, [disp8+EAX+EAX] }
    Final instruction: XOR AL, byte ptr [EAX+EAX-31]

51) Opcode      : 33

    Example         : 33 F6
    Decoding method :
     XOR { Gv, Ev }                                         ( Table A-4 )
     XOR { ESI/SI/DH, ESI/SI/DH }                           ( Table A-5 )
    Final instruction: XOR ESI, ESI

    Example         : 33 64 41 75
    Decoding method :
     XOR { Gv, Ev }                                         ( Table A-4 )
     XOR { ESP/SP/AH, [disp8+[--][--]] }                    ( Table A-5 )
     XOR { ESP/SP/AH, [disp8+([EAX*2], ECX)] }              ( Table A-6 )
     XOR { ESP/SP/AH, [disp8+EAX*2+ECX] }
    Final instruction: XOR ESP, dword ptr [75+EAX*2+ECX]
```

```
52) Opcode      : 34

    Example         : 34 40
    Decoding method :
     XOR { AL, Ib }                                         ( Table A-4 )
    Final instruction: XOR AL, 40

53) Opcode      : 35

    Example         : 35 40 00 9F 35
    Decoding method :
     XOR { eAX, Iv }                                        ( Table A-4 )
    Final instruction: XOR EAX, 359F0040

54) Opcode      : 36

    Description     : This opcode indicates SS segment register as Prefix
                      for memory locations.

    Example         : 36 ( Segment Prefix )
                      30 32 ( Decode like normal instructions )
    Decoding method :
     SEG { =SS }                                            ( Table A-4 )

     XOR { Eb, Gb }                                         ( Table A-4 )
     XOR { [EDX], ESI/SI/DH }                               ( Table A-5 )
    Final instruction: XOR byte ptr SS:[EDX], DH

    Example         : 36 ( Segment Prefix )
                      93 ( Decode like normal instructions )
    Decoding method :
     SEG { =SS }                                            ( Table A-4 )

     XCHG { eAX, eBX }                                      ( Table A-4 )
    Final instruction: XCHG EAX, EBX

55) Opcode      : 37

    Example         : 37
    Decoding method :
     AAA                                                    ( Table A-4 )
    Final instruction: AAA

56) Opcode      : 38

    Example         : 38 E0
    Decoding method :
     CMP { Eb, Gb }                                         ( Table A-4 )
     CMP { EAX/AX/AL, ESP/SP/AH }                           ( Table A-4 )
    Final instruction: CMP AL, AH

    Example         : 38 07
    Decoding method :
     CMP { Eb, Gb }                                         ( Table A-4 )
     CMP { [EDI], EAX/AX/AL }                               ( Table A-5 )
    Final instruction: CMP byte ptr [EDI], AL
```

```
57) Opcode      : 39

    Example         : 39 06
    Decoding method :
     CMP { Ev, Gv }                                         ( Table A-4 )
     CMP { [ESI], EAX/AX/AL }                               ( Table A-5 )
    Final instruction: CMP dword ptr [ESI], EAX

    Example         : 39 7C 24 3C
    Decoding method :
     CMP { Ev, Gv }                                         ( Table A-4 )
     CMP { [disp8+[--][--]], EDI/DI/BH }                    ( Table A-5 )
     CMP { [disp8+(none, ESP)], EDI/DI/BH }                 ( Table A-6 )
     CMP { [disp8+ESP], EDI/DI/BH }
    Final instruction: CMP dword ptr [3C+ESP], EDI

58) Opcode      : 3A

    Example         : 3A D1
    Decoding method :
     CMP { Gb, Eb }                                         ( Table A-4 )
     CMP { EDX/DX/DL, ECX/CX/CL }                           ( Table A-5 )
    Final instruction: CMP DL, CL

    Example         : 3A 5C 57 49
    Decoding method :
     CMP { Gb, Eb }                                         ( Table A-4 )
     CMP { EBX/BX/BL, [disp8+[--][--]] }                    ( Table A-5 )
     CMP { EBX/BX/BL, [disp8+([EDX*2], EDI)] }              ( Table A-6 )
     CMP { EBX/BX/BL, [disp8+EDX*2+EDI] }
    Final instruction: CMP BL, byte ptr [49+EDX*2+EDI]

59) Opcode      : 3B

    Example         : 3B C1
    Decoding method :
     CMP { Gv, Ev }                                         ( Table A-4 )
     CMP { EAX/AX/AL, ECX/CX/CL }                           ( Table A-5 )
    Final instruction: CMP EAX, ECX

    Example         : 3B 0C ED 40 96 40 00
    Decoding method :
     CMP { Gv, Ev }                                         ( Table A-4 )
     CMP { ECX/CX/CL, [--][--] }                            ( Table A-5 )
     CMP { ECX/CX/CL, ([EBP*8],[*]) }                       ( Table A-6 )
     CMP { ECX/CX/CL, [EBP*8+00409640] }
    Final instruction: CMP ECX, dword ptr [EBP*8+00409640]

60) Opcode      : 3C

    Example         : 3C 40
    Decoding method :
     CMP { AL, Ib }                                         ( Table A-4 )
    Final instruction: CMP AL, 40

61) Opcode      : 3D
```

```
    Example         : 3D 76 01 00 00
    Decoding method :
     CMP { eAX, Iv }                        ( Table A-4 )
    Final instruction: CMP EAX, 00000176

62) Opcode      : 3E

    Description     : This opcode indicates DS segment register as Prefix
                      for memory locations.

    Example         : 3E ( Segment Prefix )
                      01 00 ( Decode like normal instructions )

    Decoding method :
     SEG { =DS }                            ( Table A-4 )

     ADD { Ev, Gv }                         ( Table A-4 )
     ADD { [EAX], EAX/AX/AL }               ( Table A-6 )
    Final instruction: ADD dword ptr DS:[EAX], EAX

    Example         : 3E ( Segment Prefix )
                      49 ( Decode like normal instructions )

    Decoding method :
     SEG { =DS }                            ( Table A-4 )

     DEC { eCX }                            ( Table A-4 )
    Final instruction: DEC ECX

63) Opcode      : 3F

    Example         : 3F
    Decoding method :
     AAS                                    ( Table A-4 )
    Final instruction: AAS

64) Opcode      : 40

    Example         : 40
    Decoding method :
     INC { eAX }                            ( Table A-4 )
    Final instruction: INC EAX

65) Opcode      : 41

    Example         : 41
    Decoding method :
     INC { eCX }                            ( Table A-4 )
    Final instruction: INC ECX

66) Opcode      : 42

    Example         : 42
    Decoding method :
     INC { eDX }                            ( Table A-4 )
    Final instruction: INC EDX
```

```
67) Opcode      : 43

    Example         : 43
    Decoding method :
     INC { eBX }                                            ( Table A-4 )
    Final instruction: INC EBX

68) Opcode      : 44

    Example         : 44
    Decoding method :
     INC { eSP }                                            ( Table A-4 )
    Final instruction: INC ESP

69) Opcode      : 45

    Example         : 45
    Decoding method :
     INC { eBP }                                            ( Table A-4 )
    Final instruction: INC EBP

70) Opcode      : 46

    Example         : 46
    Decoding method :
     INC { eSI }                                            ( Table A-4 )
    Final instruction: INC ESI

71) Opcode      : 47

    Example         : 47
    Decoding method :
     INC { eDI }                                            ( Table A-4 )
    Final instruction: INC EDI

72) Opcode      : 48

    Example         : 48
    Decoding method :
     DEC { eAX }                                            ( Table A-4 )
    Final instruction: DEC EAX

73) Opcode      : 49

    Example         : 49
    Decoding method :
     DEC { eCX }                                            ( Table A-4 )
    Final instruction: DEC ECX

74) Opcode      : 4A

    Example         : 4A
    Decoding method :
     DEC { eDX }                                            ( Table A-4 )
    Final instruction: DEC EDX
```

```
75) Opcode      : 4B

    Example         : 4B
    Decoding method :
     DEC { eBX }                                      ( Table A-4 )
    Final instruction: DEC EBX

76) Opcode      : 4C

    Example         : 4C
    Decoding method :
     DEC { eSP }                                      ( Table A-4 )
    Final instruction: DEC ESP

77) Opcode      : 4D

    Example         : 4D
    Decoding method :
     DEC { eBP }                                      ( Table A-4 )
    Final instruction: DEC EBP

78) Opcode      : 4E

    Example         : 4E
    Decoding method :
     DEC { eSI }                                      ( Table A-4 )
    Final instruction: DEC ESI

79) Opcode      : 4F

    Example         : 4F
    Decoding method :
     DEC { eDI }                                      ( Table A-4 )
    Final instruction: DEC EDI

80  Opcode      : 50

    Example         : 50
    Decoding method :
     PUSH { eAX }                                     ( Table A-4 )
    Final instruction: PUSH EAX

81) Opcode      : 51

    Example         : 51
    Decoding method :
     PUSH { eCX }                                     ( Table A-4 )
    Final instruction: PUSH ECX

82) Opcode      : 52

    Example         : 52
    Decoding method :
     PUSH { eDX }                                     ( Table A-4 )
    Final instruction: PUSH EDX

83) Opcode      : 53
```

```
    Example         : 53
    Decoding method :
     PUSH { eBX }                                          ( Table A-4 )
    Final instruction: PUSH EBX

84) Opcode      : 54

    Example         : 54
    Decoding method :
     PUSH { eSP }                                          ( Table A-4 )
    Final instruction: PUSH ESP

85) Opcode      : 55

    Example         : 55
    Decoding method :
     PUSH { eBP }                                          ( Table A-4 )
    Final instruction: PUSH EBP

86) Opcode      : 56

    Example         : 56
    Decoding method :
     PUSH { eSI }                                          ( Table A-4 )
    Final instruction: PUSH ESI

87) Opcode      : 57

    Example         : 57
    Decoding method :
     PUSH { eDI }                                          ( Table A-4 )
    Final instruction: PUSH EDI

88) Opcode      : 58

    Example         : 58
    Decoding method :
     POP { eAX }                                           ( Table A-4 )
    Final instruction: POP EAX

89) Opcode      : 59

    Example         : 59
    Decoding method :
     POP { eCX }                                           ( Table A-4 )
    Final instruction: POP ECX

90) Opcode      : 5A

    Example         : 5A
    Decoding method :
     POP { eDX }                                           ( Table A-4 )
    Final instruction: POP EDX

91) Opcode      : 5B
```

```
    Example         : 5B
    Decoding method :
     POP { eBX }                                            ( Table A-4 )
    Final instruction: POP EBX

92) Opcode      : 5C

    Example         : 5C
    Decoding method :
     POP { eSP }                                            ( Table A-4 )
    Final instruction: POP ESP

93) Opcode      : 5D

    Example         : 5D
    Decoding method :
     POP { eBP }                                            ( Table A-4 )
    Final instruction: POP EBP

94) Opcode      : 5E

    Example         : 5E
    Decoding method :
     POP { eSI }                                            ( Table A-4 )
    Final instruction: POP ESI

95) Opcode      : 5F

    Example         : 5F
    Decoding method :
     POP { eDI }                                            ( Table A-4 )
    Final instruction: POP EDI

96) Opcode      : 60

    Example         : 60
    Decoding method :
     PUSHA                                                  ( Table A-4 )
    Final instruction: PUSHA

97) Opcode      : 61

    Example         : 61
    Decoding method :
     POPA                                                   ( Table A-4 )
    Final instruction: POPA

98) Opcode      : 62

    Example         : 62 3A
    Decoding method :
     BOUND { Gv, Ma }                                       ( Table A-4 )
     BOUND { EDI/DI/BH, [EDX] }                             ( Table A-5 )
    Final instruction: BOUND EDI, qword ptr [ EDX ]

    Example         : 62 74 72 61
    Decoding method :
     BOUND { Gv, Ma }                                       ( Table A-4 )
     BOUND { ESI/SI/DH, [disp8+[--][--]] ) }                ( Table A-5 )
     BOUND { ESI, [disp8+([ESI*2], EDX)] }                  ( Table A-6 )
     BOUND { ESI, [disp8+ESI*2+EDX] }
    Final instruction: BOUND ESI, qword ptr [61+ESI*2+EDX]

    Note : Refer the abbrivations in beginning of the artical.
           qword = 8 bytes

99) Opcode      : 63

    Example         : 63 00
    Decoding method :
     ARPL { Ew, Rw }                                        ( Table A-4 )
     ARPL { [EAX], EAX/AX/AL }                              ( Table A-5 )
    Final instruction: ARPL word ptr [EAX], AX

    Example         : 63 74 69 76
    Decoding method :
     ARPL { Ew, Rw }                                        ( Table A-4 )
     ARPL { [disp8+[--][--]], ESI/SI/DH }                   ( Table A-5 )
     ARPL { [disp8+([EBP*2], ECX)], ESI/SI/DH }             ( Table A-6 )
     ARPL { [disp8+EBP*2+ECX], ESI/SI/DH }
    Final instruction: ARPL word ptr [76+EBP*2+ECX], SI



100) Opcode         : 64

     Description    : This opcode indicates FS segment register as Prefix
                      for memory locations.

     Example        : 64 ( Segment Prefix )
                      89 25 00 00 00 00 (Decode like normal instructions)
     Decoding method:
      SEG { =FS }                                       ( Table A-4 )

      MOV { Ev, Gv }                                    ( Table A-4 )
      MOV { [disp32], ESP/SP/AH }                       ( Table A-5 )

     Final instruction  : MOV dword ptr FS:[00000000], ESP

     Example        : 64 ( Segment Prefix )
                      43 (Decode like normal instructions)
     Decoding method:
      SEG { =FS }                                       ( Table A-4 )

      INC { eBX }                                       ( Table A-4 )
     Final instruction  : INC EBX
```

------------------------------[Final Word]------------------------------

Here ends my first Article. I hope this will be useful to you. If u find any mistakes or comments or any other thing you feel u can contact me at:

info@yashks.com

```
+----------------------------------------------------------------------------+
¦                         Yash K.S - 2001                                    ¦
+----------------------------------------------------------------------------+
```