

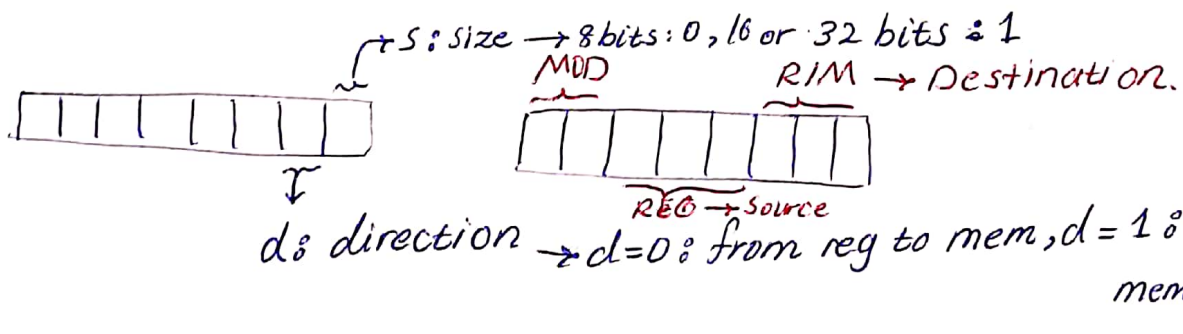
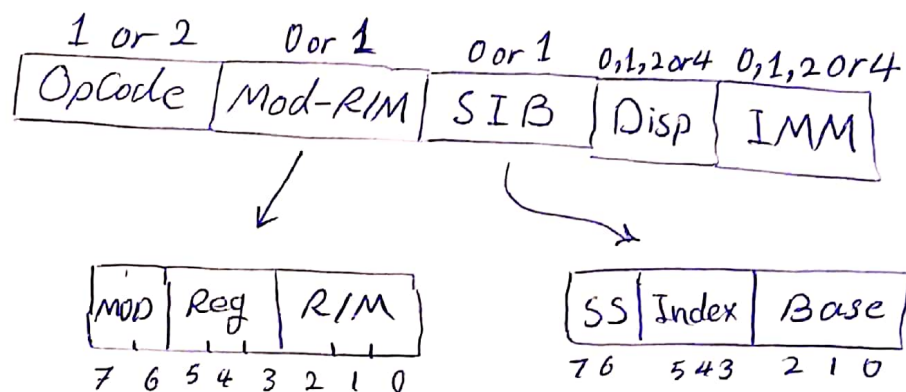
- Assembly Project. Part 2.

4 Instructions { AND, OR
ADD, SUB

only for Regs.

Total format :

NO. of bytes



* Note: for this part because the transferring data is from register to register, so we assume the MOD=11.
~~But~~ It means d=X (direction bit is don't care).
 But it is a bit different if we assume d=0, or d=1.

If $d = 0$: OpCode MOD SourceReg DstReg
 If $d = 1$: OpCode MOD DstReg SourceReg

Which means we have 2 ways to Assemble a machine code for some instructions. But the most common way is the first one. ($d = 0$).

e.g. Add eax, edx

0000 00x1 1101 0000 $\left\{ \begin{array}{l} 0000\ 0001\ 1101\ 0000 \rightarrow 01\ D0 \\ 0000\ 0011\ 1100\ 0010 \rightarrow 03\ C2 \end{array} \right.$

e.g. OR eax, ebx

0000 1001 1101 1000 $\rightarrow 09\ DB$
 or
 0000 1011 1100 0011 $\rightarrow 0B\ C3$

e.g. AND ecx, edx

0010 0001 1101 0001 $\rightarrow 21\ D1$
 0010 0011 1100 1010 $\rightarrow 23\ CA$

e.g. SUB ebx, eax

0010 1001 1100 0011 $\rightarrow 29\ C3$
 0010 1011 1101 1000 $\rightarrow 2B\ D8$

• For 16 bits Reg 8

We use 2 bytes Opcode. Infact a prefix.

X66. \rightarrow 0110 0110,

e.g. ADD ax, bx

0110 0110 0000 0011 1100 0011 \rightarrow 66 03 C3
 0110 0110 0000 0001 1101 1000 \rightarrow 66 01 D8

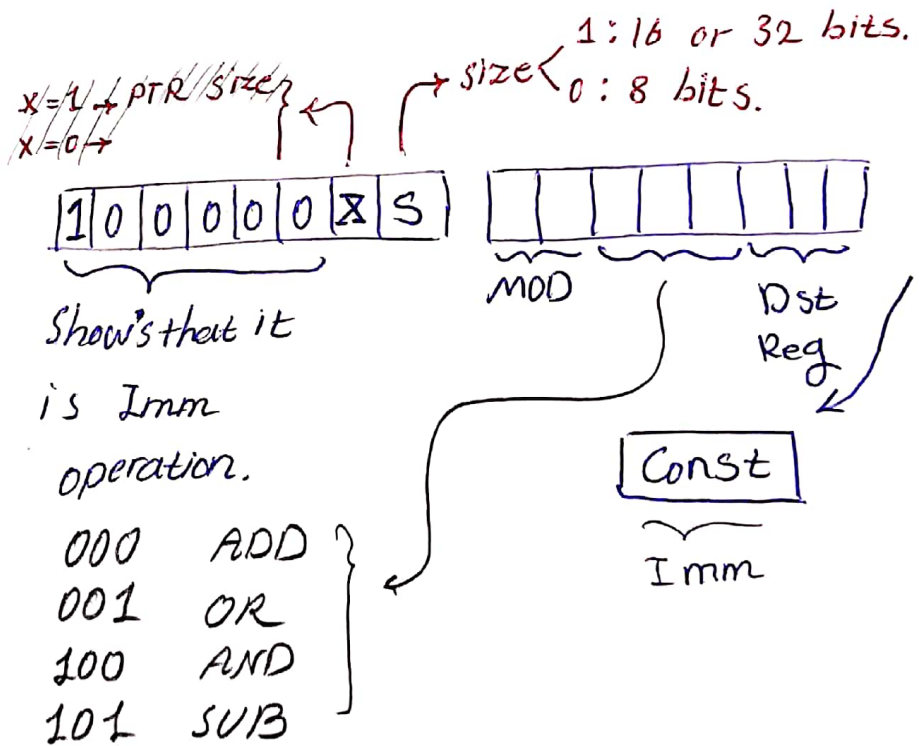
e.g. SUB cx, 108

0110 0110 1000 0011 1110 1001 0110 1100 \rightarrow 66 83 E9 6C
~~0110 0110~~

• Immediate :

e.g. AND edx, 1

1000 0011 1110 0010 0000 0001
 \rightarrow 83 E2 01



* Exception: The Above format isn't correct only for one amount (case) \rightarrow Instruction al, Const



Fraunhofer

FKIE

FRAUNHOFER-INSTITUT FÜR KOMMUNIKATION, INFORMATIONSVERARBEITUNG UND ERGONOMIE FKIE

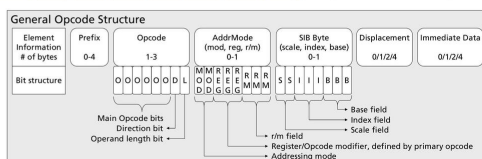
x86 Opcode Structure and Instruction Overview

1 st	2 nd	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		ADD					ES	ES	OR						CS	TWO	BYTE
1		ADC					PUSH	POP	SBB						PUSH	POP	DS
2		AND					ES	DAA	SUB						CS	DAS	
3		XOR					SEGMENT	AAA	CMP						DS	AAS	
4		INC							DEC								
5		PUSH							POP								
6		PUSHAD	POPAD	BOUND	ARPL	FS	GS	OPERAND	ADDRESS		PUSH	IMUL	PUSH	IMUL	INS	OUTS	
7		JO	JNO	JB	JNB	JE	JNE	JBE	JA	JS	JNS	JPE	JPO	JL	JGE	JLE	JG
8		ADD/ADC/AND/XOR				TEST	XCHG	MOV REG				MOV	SREG	LEA	MOV	SREG	POP
9		NOP				XCHG EAX				CWD	CDQ	CALL	WAIT	PUSHD	POPF	SAHF	LAHF
A		MOV EAX				MOVS	CMP	TEST	STOS	LODS	SCAS						
B		MOV															
C		SHIFT IMM	RETN	LES	LDS	MOV IMM	ENTER	LEAVE	RETF	INT3	INT	INT	INT	INT	INT	INT	INT
D		SHIFT 1	SHIFT CL	AAM	AAD	SALC	XLAT	FPU									
E		LOOPNZ	LOOPZ	LOOP	JECXZ	IN IMM	OUT IMM	CALL	JMP	JMPF	JMP SHORT	IN	DX	OUT	DX		
F		LOCK	ICE	REP	REP	HLT	CMC	TEST/NOT/NEG	CLC	STC	CLI	STI	CLD	STD	INC	DEC	INP/OUT

1 st	2 nd	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		ILS/DT	ILS/DT	LAR	LSL			CLTS		INVD	WBINVD		UD2		NOP		
1		SSE{1,2,3}								Prefetch		HINT_NOP					
2		MOV CR/DR								SSE{1,2}							
3		WRMSR	RDPMC	RDMSR	RDPMC	SYSENTER	SYSEXIT			GETSEC	MOVBE /	THREE					
4		CMOV															
5		SSE{1,2}															
6		MMX, SSE2															
7		MMX, SSE{1,2,3}, VMX															
8		JO	JNO	JB	JNB	JE	JNE	JBE	JA	JS	JNS	JPE	JPO	JL	JGE	JLE	JG
9		SETO	SETNO	SETB	SETNB	SETE	SETNE	SETBE	SETA	SETS	SETNS	SETPE	SETPO	SETL	SETGE	SETLE	SETG
A		PUSH	POP	CPUI	BT	SHLD				PUSH	POP	RSM	BTS	SHRD	*FENCE	IMUL	
B		CMPXCHG	LSS	BTR	LFS	LGS	MOVZX	POPCNT	UD	BT	BTR	BTC	BSF	BSR	MOVSB		
C		XADD								CMPXCHG		BSWAP					
D		MMX, SSE{1,2,3}															
E		MMX, SSE{1,2}															
F		MMX, SSE{1,2,3}															

Arithmetic & Logic
Memory
Stack
Control Flow & Conditional

Prefix
System & I/O
No Operation (NOP) / Multiple Instructions / Extended Instruction Set



mod	00	01	10	11
r/m	8bit	8bit	8bit	8bit
000	8bit	8bit	8bit	8bit
001	8bit	8bit	8bit	8bit
010	8bit	8bit	8bit	8bit
011	8bit	8bit	8bit	8bit
100	8bit	8bit	8bit	8bit
101	8bit	8bit	8bit	8bit
110	8bit	8bit	8bit	8bit
111	8bit	8bit	8bit	8bit

encoding	scale (2bit)	index (3bit)	base (3bit)
000	2 ⁰ =1	[EAX]	EAX
001	2 ¹ =2	[ECX]	ECX
010	2 ² =4	[EDX]	EDX
011	2 ³ =8	[EBX]	EBX
100	...	none	ESP
101	...	[EBP]	EBP
110	...	[ESI]	ESI
111	...	[EDI]	EDI

v1.0 – 30.08.2011

Contact: Daniel Plohmann – +49 228 73 54 228 – daniel.plohmann@fkie.fraunhofer.de

Source: Intel x86 Instruction Set Reference
Opcode table presentation inspired by work of Ange Albertini