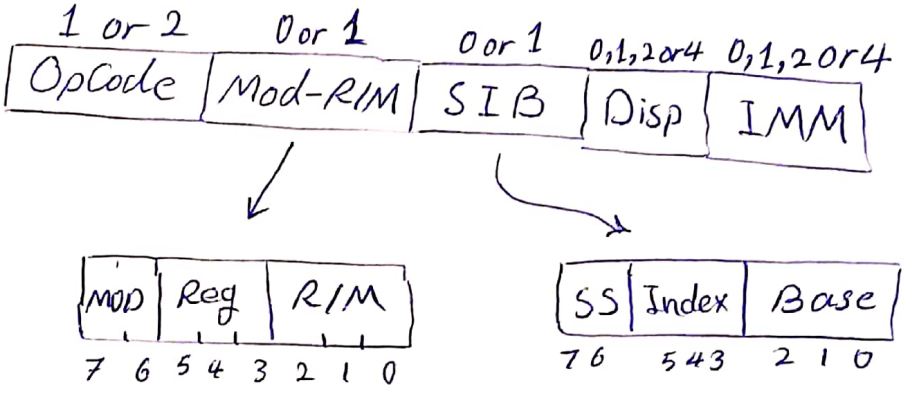
u Seyed Ali Maher u u 9932113 u

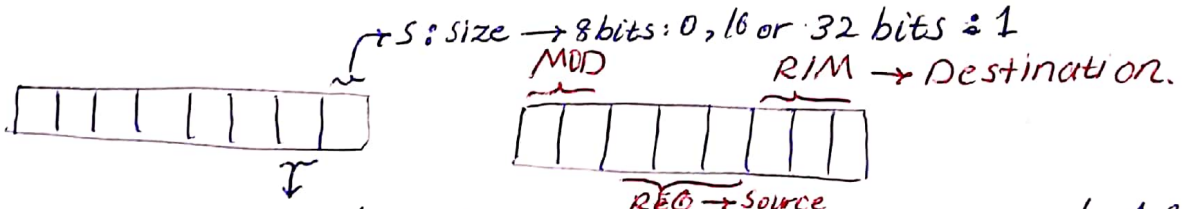
- Assembly Project. Part 1.

4 Instructions { AND, OR / ADD, SUB } only for Regs.

Total format :

NO. of bytes





d: direction → d=0: from reg to mem, d=1: from mem to reg

* Note: for this part because the transferring data is from register to register, So we assume the MOD=11.

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 Source Reg Dst Reg

If $d = 1$: OpCode MOD Dst Reg Source Reg

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 <

0000 0001 1101 0000 → 01 D0

0000 0011 1100 0010 → 03 C2

e.g. OR eax, ebx

0000 1001 1101 1000 → 09 D8

OR

0000 1011 1100 0011 → 0B C3

e.g AND ecx, edx

0010 0001 1101 0001 → 21 D1

0010 0011 1100 1010 → 23 CA

e.g SUB ebx, eax

0010 1001 1100 0011 → 29 C3

0010 1011 1101 1000 → 2B D8

• For 16 bits Reg 8

We use 2 bytes Opcode. Infact a prefix.

X66. → 0110 0110

e.g. ADD ax, bx

0110 0110 0000 0011 1100 0011 → 66 03 C3

0110 0110 0000 0001 1101 1000 → 66 01 D8

e.g SUB cx, 108

0110 0110 1000 0011 1110 1001 0110 1100 → 66 83 E9 6C

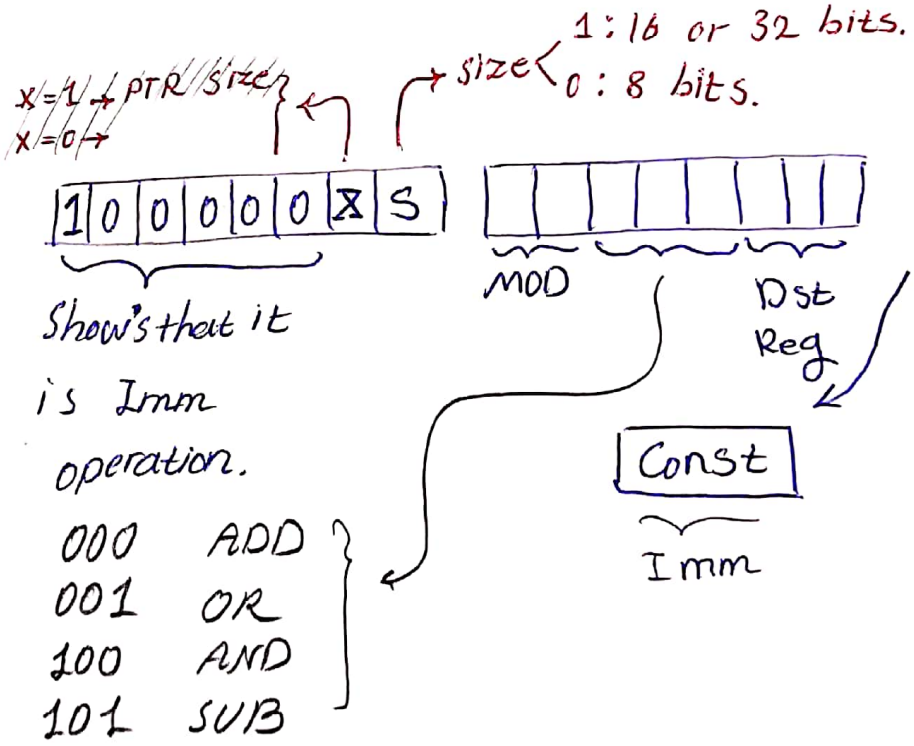
~~0110 0110~~

• Immediate:

e.g. AND edx, 1

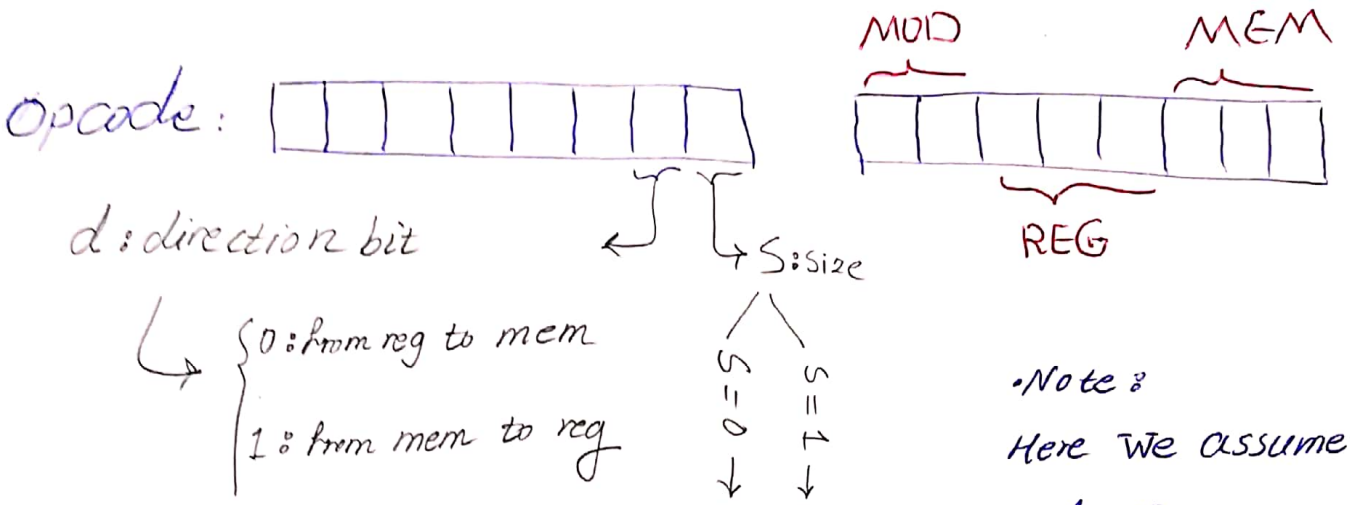
1000 0011 1110 0010 0000 0001

→ 83 E2 01



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

Part 2. Total format is as below:



•Note:
Here we assume mod 00.

Some Examples:

ADD cl, [bl]

0000 0010 0000 1011 → 02 0b

ADD [bl], cl

0000 0000 0000 1011 → 00 0b

ADD eax, [ecx]

0000 0011 0000 0001 → 03 01

ADD [ecx], eax

0000 0001 00000001 → 01 01

Warning: We don't support 16 bits In mod 00. Bcz of it's BUT & IF :)

Tip1: For "ah" and "esp" we use the above total format.

But ~~the~~ in fact we use 100, for SIB mode.

Tip 2: We don't have [ebp] or [ch], because it used for 32 bits displacement-only addressing mode. But here Calculate with above format. e.g.

Add eax, [ebp] → my answer → 03 05

→ real Answer → 03 45 00

Fraunhofer
FKIE
FRAUNHOFER-INSTITUT FÜR KOMMUNIKATION, INFORMATIONSVERARBEITUNG UND ERGONOMIE FKIE

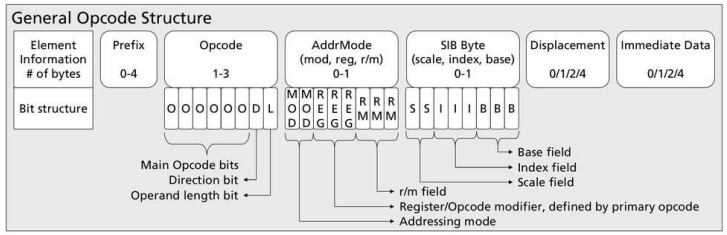
x86 Opcode Structure and Instruction Overview

1st \ 2nd	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	ADD						ES PUSH SS	ES POP SS	OR						CS PUSH DS	TWO BYTE
1	ADC								SBB							POP DS
2	AND						ES SEGMENT OVERRIDE SS	DAA	SUB						CS SEGMENT OVERRIDE DS	DAS
3	XOR							AAA	CMP							AAS
4	INC								DEC							
5	PUSH								POP							
6	PUSHAD	POPAD	BOUND	ARPL	FS SEGMENT OVERRIDE	GS	OPERAND SIZE	ADDRESS SIZE	PUSH	IMUL	PUSH	IMUL	INS		OUTS	
7	JO	JNO	JB	JNB	JE	JNE	JBE	JA	JS	JNS	JPE	JPO	JL	JGE	JLE	JG
	Jcc															
8	ADD/ADC/AND/XOR OR/SBB/SUB/CMP				TEST		XCHG		MOV REG				MOV SREG	LEA	MOV SREG	POP
9	NOP	XCHG EAX							CWD	CDQ	CALLF	WAIT	PUSHFD	POPFD	SAHF	LAHF
A	MOV EAX				MOVS		CMPS		TEST		STOS		LODS		SCAS	
B	MOV															
C	SHIFT IMM		RETN		LES	LDS	MOV IMM		ENTER	LEAVE	RETF		INT3	INT IMM	INTO	IRETD
D	SHIFT 1	SHIFT CL			AAM	AAD	SALC	XLAT	FPU							
	ROL/ROR/RCL/RCR/SHL/SHR/SAL/SAR															
E	LOOPNZ	LOOPZ	LOOP	JECXZ	IN IMM		OUT IMM		CALL	JMP	JMPF	JMP SHORT	IN DX		OUT DX	
	CONDITIONAL LOOP															
F	LOCK	ICE BP	REPNE	REPE	HLT	CMC	TEST/NOT/NEG [i]MUL/[i]DIV		CLC	STC	CLI	STI	CLD	STD	INC DEC	INC/DEC CALL/JMP PUSH
	EXCLUSIVE ACCESS		CONDITIONAL REPETITION													

1st \ 2nd	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	{L,S}LDT {L,S}TR VER{R,W}	{L,S}GDT {L,S}IDT {L,S}MSW	LAR	LSL			CLTS		INVD	WBINVD		UD2		NOP		
1	*SSE{1,2,3}*								Prefetch *SSE1*	HINT_NOP						
2	MOV CR/DR								*SSE{1,2}*							
3	WRMSR	RDTSC	RDMSR	RDPMC	SYSENTER	SYSEXIT		GETSEC *SMX*	MOVBE / THREE BYTE		THREE BYTE *SSE4*					
4	CMOV															
5	*SSE{1,2}*															
6	*MMX, SSE2*															
7	*MMX, SSE{1,2,3}, VMX*												*MMX, SSE{2,3}*			
8	JO	JNO	JB	JNB	JE	JNE	JBE	JA	JS	JNS	JPE	JPO	JL	JGE	JLE	JG
	Jcc SHORT															
9	SETO	SETNO	SETB	SETNB	SETE	SETNE	SETBE	SETA	SETS	SETNS	SETPE	SETPO	SETL	SETGE	SETLE	SETG
	SETcc															
A	PUSH FS	POP FS	CPUID	BT	SHLD				PUSH GS	POP GS	RSM	BTS	SHRD		*FENCE	IMUL
B	CMPXCHG		LSS	BTR	LFS	LGS	MOVZX		POPCNT	UD	BT BTS BTR BTC	BTC	BSF	BSR	MOVSX	
C	XADD	*SSE{1,2}*						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

General Opcode Structure



Addressing Modes

mod	00		01		10		11
r/m	16bit	32bit	16bit	32bit	16bit	32bit	r/m // REG
000	[BX+SI]	[EAX]	[BX+SI]+disp8	[EAX]+disp8	[BX+SI]+disp16	[EAX]+disp32	AL / AX / EAX
001	[BX+DI]	[ECX]	[BX+DI]+disp8	[ECX]+disp8	[BX+DI]+disp16	[ECX]+disp32	CL / CX / ECX
010	[BP+SI]	[EDX]	[BP+SI]+disp8	[EDX]+disp8	[BP+SI]+disp16	[EDX]+disp32	DL / DX / EDX
011	[BP+DI]	[EBX]	[BP+DI]+disp8	[EBX]+disp8	[BP+DI]+disp16	[EBX]+disp32	BL / BX / EBX
100	[SI]	*SIB*	[SI]+disp8	*SIB*+disp8	[SI]+disp16	*SIB*+disp32	AH / SP / ESP
101	[DI]	disp32	[DI]+disp8	[EBP]+disp8	[DI]+disp16	[EBP]+disp32	CH / BP / EBP
110	disp16	[ESI]	[BP]+disp8	[ESI]+disp8	[BP]+disp16	[ESI]+disp32	DH / SI / ESI
111	[BX]	[EDI]	[BX]+disp8	[EDI]+disp8	[BX]+disp16	[EDI]+disp32	BH / DI / EDI

SIB Byte Structure

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]	disp32 / disp8+ [EBP] / disp32 + [EBP]
110	--	[ESI]	ESI
111	--	[EDI]	EDI
SIB value = index * scale + base			

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