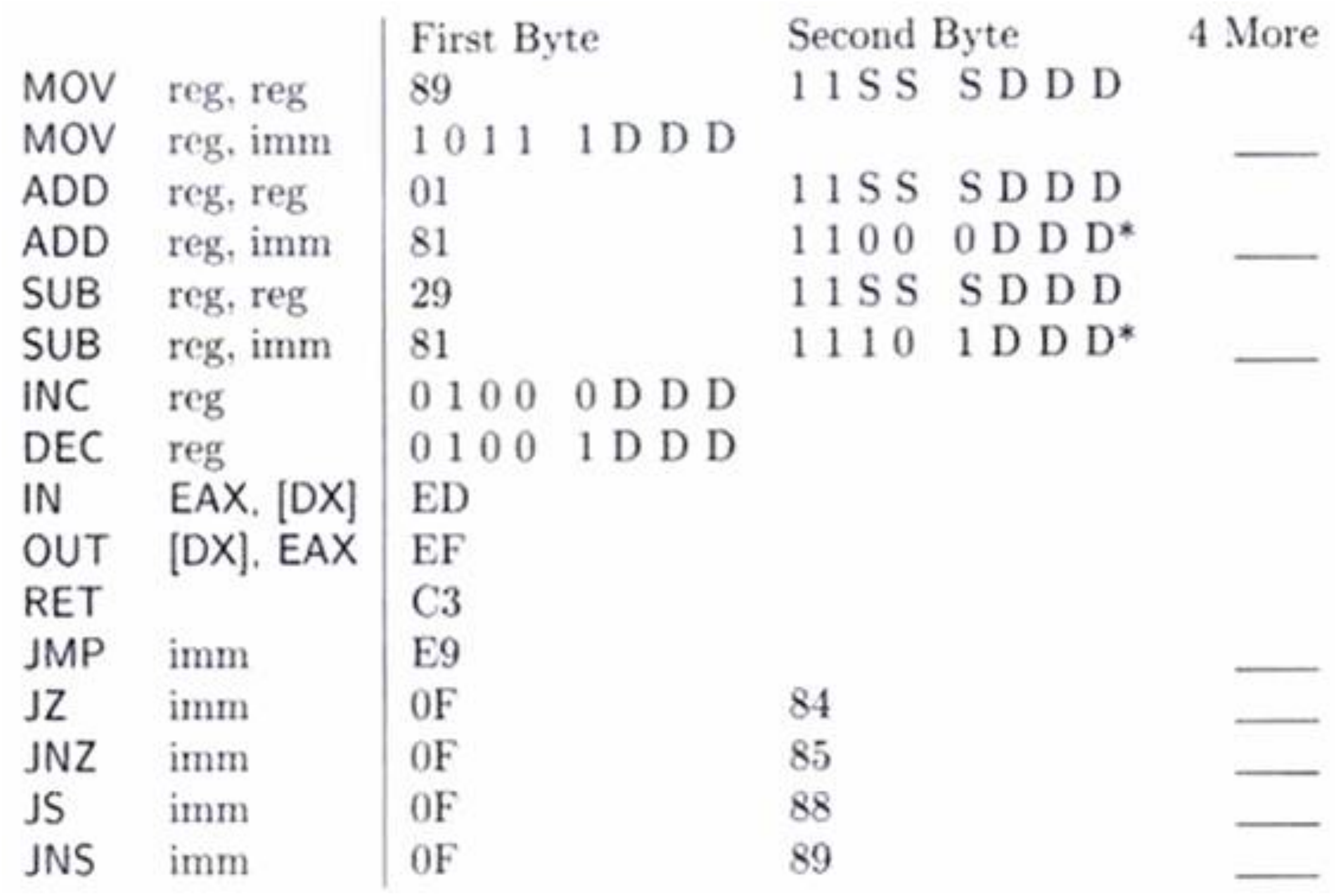
Machine Language

Machine language

- The language seen by the processor
- Assembly language specifies machine code
- Assemble
  - The process of converting assembly code into machine code
- Assembler
  - A program which takes assembly language as input and produces machine language as output

Assembling simple program

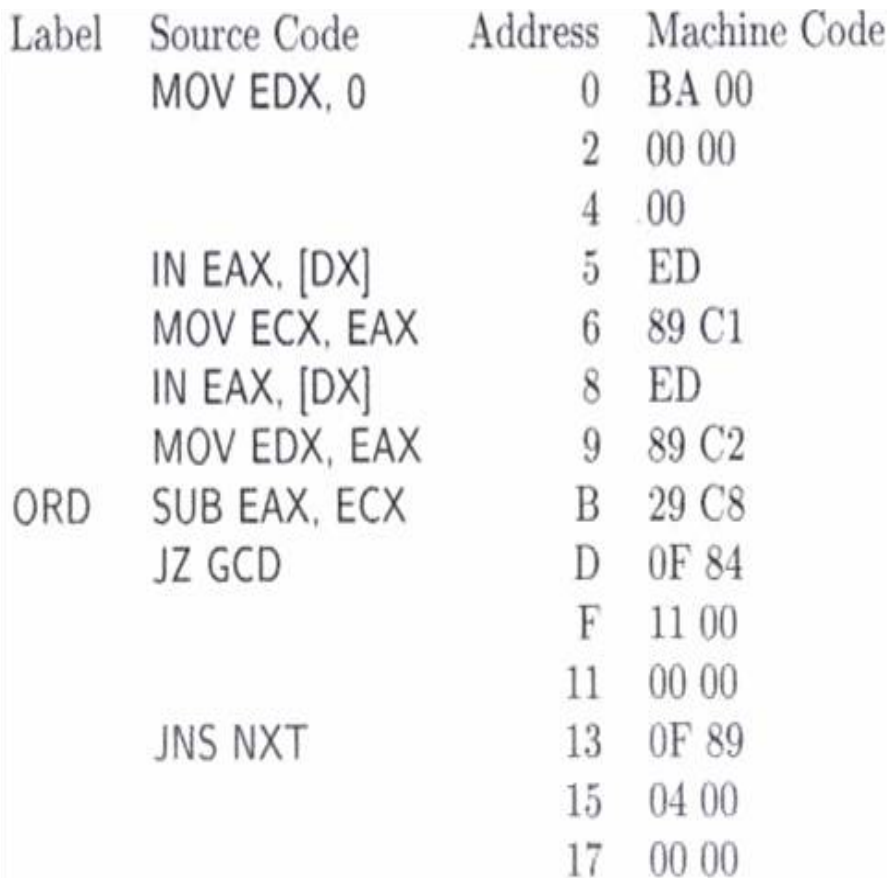


		First Byte	Second Byte	4 More
MOV	reg, reg	89	1 1 S S S D D D	
MOV	reg, imm	1 0 1 1 1 D D D		——
ADD	reg, reg	01	1 1 S S S D D D	
ADD	reg, imm	81	1 1 0 0 0 D D D*	——
SUB	reg, reg	29	1 1 S S S D D D	
SUB	reg, imm	81	1 1 1 0 1 D D D*	——
INC	reg	0 1 0 0 0 D D D		
DEC	reg	0 1 0 0 1 D D D		
IN	EAX, [DX]	ED		
OUT	[DX], EAX	EF		
RET		C3		
JMP	imm	E9		——
JZ	imm	0F	84	——
JNZ	imm	0F	85	——
JS	imm	0F	88	——
JNS	imm	0F	89	——

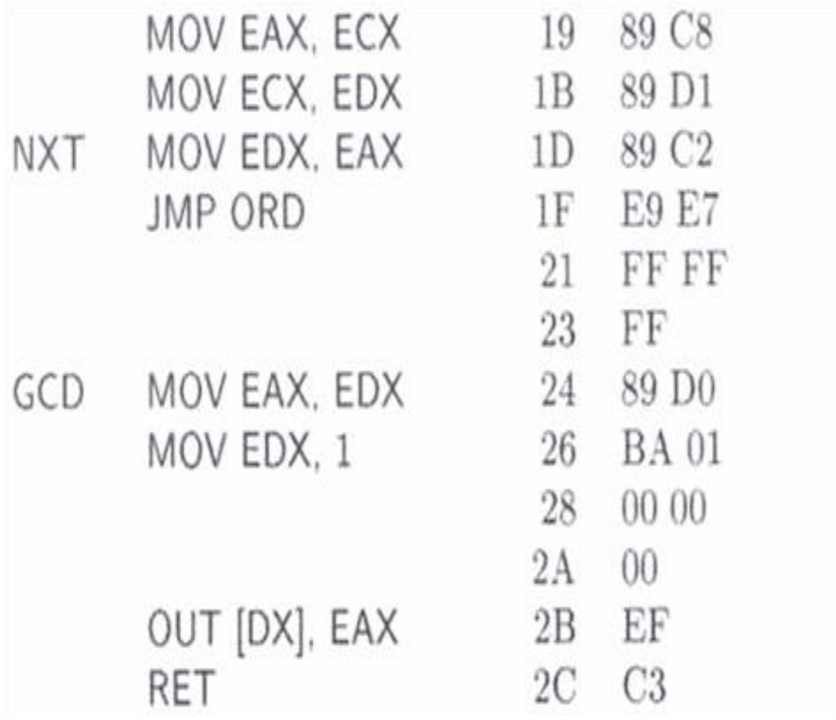
Reg codes

EAX	000
ECX	001
EDX	010
EBX	011
ESP	100
EBP	101
ESI	110
EDI	111

Assembling simple program

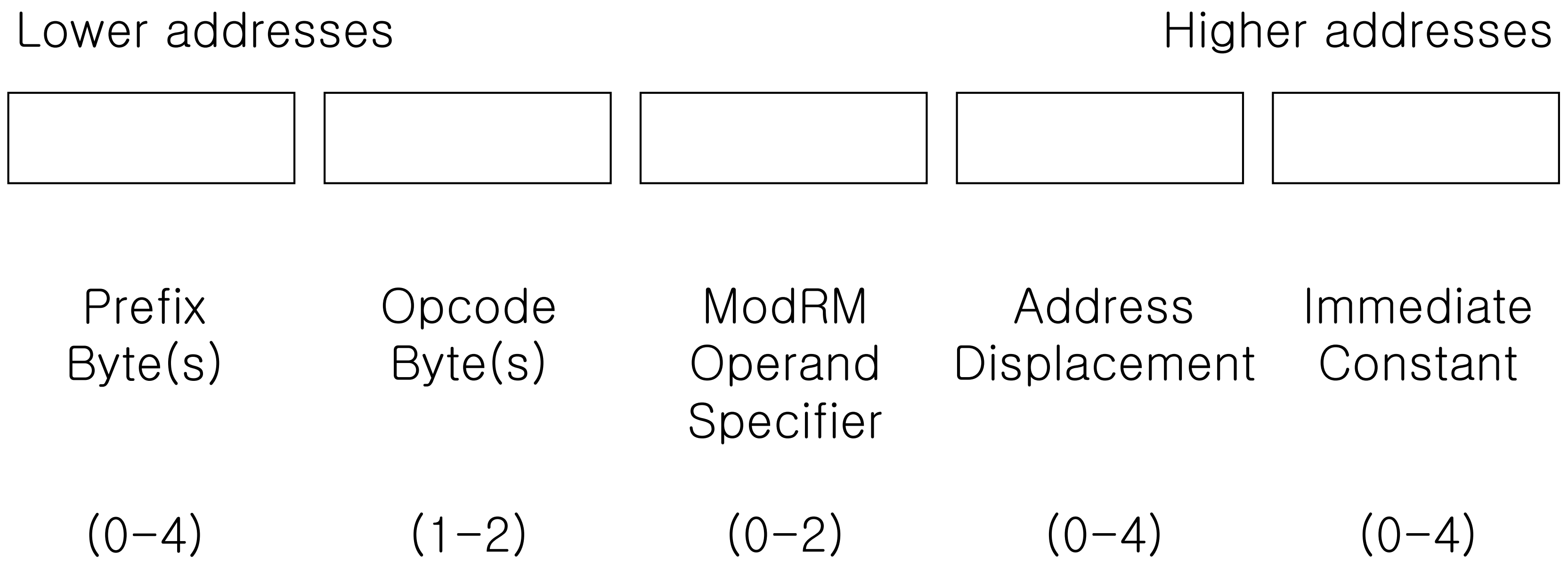


Label	Source Code	Address	Machine Code
	MOV EDX, 0	0	BA 00
		2	00 00
		4	00
	IN EAX, [DX]	5	ED
	MOV ECX, EAX	6	89 C1
	IN EAX, [DX]	8	ED
	MOV EDX, EAX	9	89 C2
ORD	SUB EAX, ECX	B	29 C8
	JZ GCD	D	0F 84
		F	11 00
		11	00 00
	JNS NXT	13	0F 89
		15	04 00
		17	00 00



	MOV EAX, ECX	19	89 C8
	MOV ECX, EDX	1B	89 D1
NXT	MOV EDX, EAX	1D	89 C2
	JMP ORD	1F	E9 E7
		21	FF FF
		23	FF
GCD	MOV EAX, EDX	24	89 D0
	MOV EDX, 1	26	BA 01
		28	00 00
		2A	00
	OUT [DX], EAX	2B	EF
	RET	2C	C3

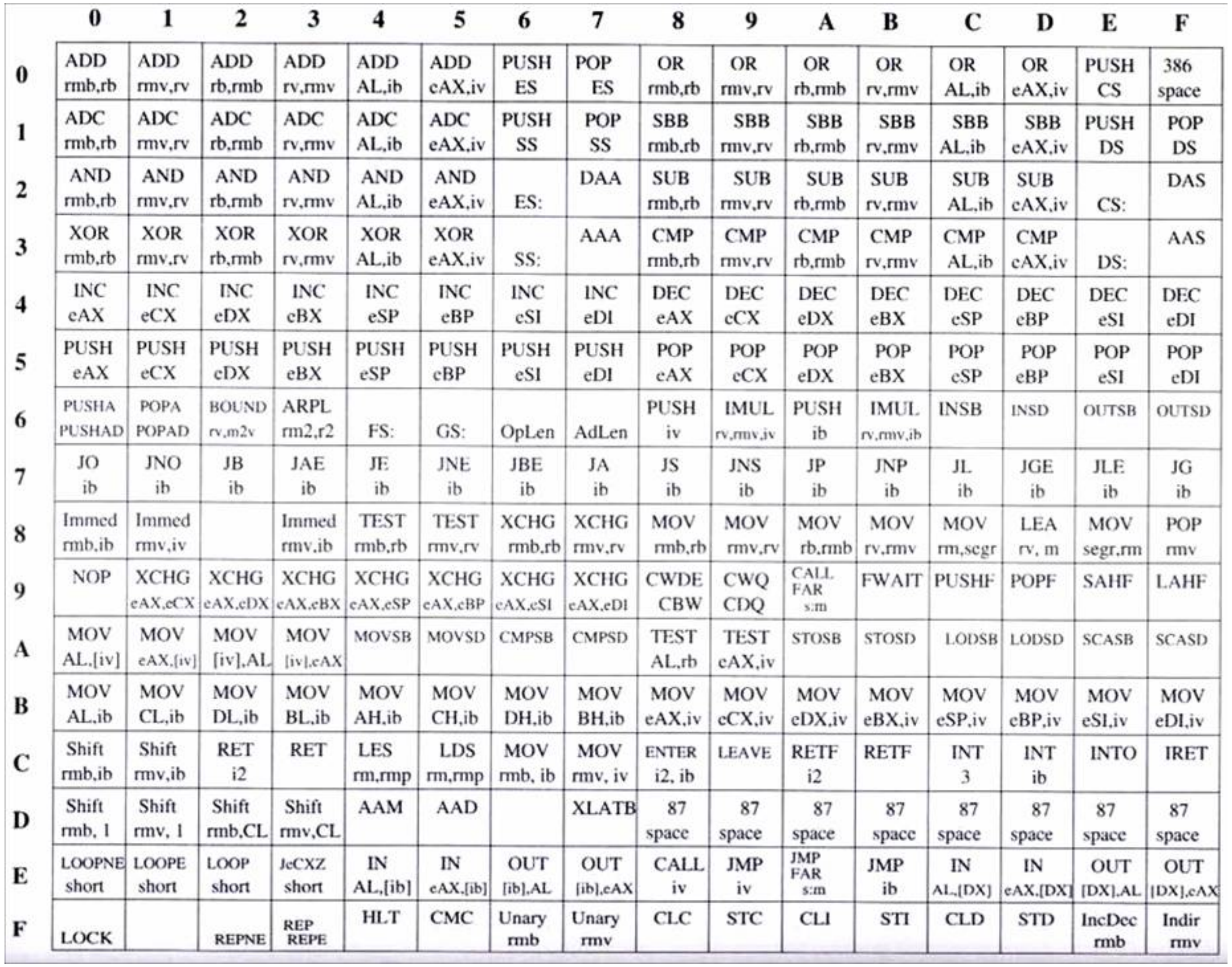
General Machine Instruction Format



Opcode space

- It contains both mnemonic and operand information
- Letter keys for operand
  - **r** register
  - **m** memory
  - **i** immediate
  - **b** byte
  - **2** 2 bytes
  - **v** 32 bits or 16 bits
  - **e** means E in 32 bits and disappears in 16 bits

Overview of the entire set of x86 machine code



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	ADD rmb,rb	ADD rmv,rv	ADD rb,rmb	ADD rv,rmv	ADD AL,ib	ADD eAX,iv	PUSH ES	POP ES	OR rmb,rb	OR rmv,rv	OR rb,rmb	OR rv,rmv	OR AL,ib	OR eAX,iv	PUSH CS	386 space
1	ADC rmb,rb	ADC rmv,rv	ADC rb,rmb	ADC rv,rmv	ADC AL,ib	ADC eAX,iv	PUSH SS	POP SS	SBB rmb,rb	SBB rmv,rv	SBB rb,rmb	SBB rv,rmv	SBB AL,ib	SBB eAX,iv	PUSH DS	POP DS
2	AND rmb,rb	AND rmv,rv	AND rb,rmb	AND rv,rmv	AND AL,ib	AND eAX,iv	ES:	DAA	SUB rmb,rb	SUB rmv,rv	SUB rb,rmb	SUB rv,rmv	SUB AL,ib	SUB eAX,iv	CS:	DAS
3	XOR rmb,rb	XOR rmv,rv	XOR rb,rmb	XOR rv,rmv	XOR AL,ib	XOR eAX,iv	SS:	AAA	CMP rmb,rb	CMP rmv,rv	CMP rb,rmb	CMP rv,rmv	CMP AL,ib	CMP eAX,iv	DS:	AAS
4	INC eAX	INC eCX	INC eDX	INC eBX	INC eSP	INC eBP	INC eSI	INC eDI	DEC eAX	DEC eCX	DEC eDX	DEC eBX	DEC eSP	DEC eBP	DEC eSI	DEC eDI
5	PUSH eAX	PUSH eCX	PUSH eDX	PUSH eBX	PUSH eSP	PUSH eBP	PUSH eSI	PUSH eDI	POP eAX	POP eCX	POP eDX	POP eBX	POP eSP	POP eBP	POP eSI	POP eDI
6	PUSHA PUSHAD	POPA POPAD	BOUND rv,m2v	ARPL rm2,r2	FS:	GS:	OpLen	AdLen	PUSH iv	IMUL rv,rmv,iv	PUSH ib	IMUL rv,rmv,ib	INSB	INSD	OUTSB	OUTSD
7	JO ib	JNO ib	JB ib	JAE ib	JE ib	JNE ib	JBE ib	JA ib	JS ib	JNS ib	JP ib	JNP ib	JL ib	JGE ib	JLE ib	JG ib
8	Immed rmb,ib	Immed rmv,iv		Immed rmv,ib	TEST rmb,rb	TEST rmv,rv	XCHG rmb,rb	XCHG rmv,rv	MOV rmb,rb	MOV rmv,rv	MOV rb,rmb	MOV rv,rmv	MOV rm,segr	LEA rv, m	MOV segr,rm	POP rmv
9	NOP	XCHG eAX,eCX	XCHG eAX,eDX	XCHG eAX,eBX	XCHG eAX,eSP	XCHG eAX,eBP	XCHG eAX,eSI	XCHG eAX,eDI	CWDE CBW	CWQ CDQ	CALL FAR s:m	FWAIT	PUSHF	POPF	SAHF	LAHF
A	MOV AL,[iv]	MOV eAX,[iv]	MOV [iv],AL	MOV [iv],eAX	MOVSB	MOVSD	CMPSB	CMPSD	TEST AL,rb	TEST eAX,iv	STOSB	STOSD	LODSB	LODSD	SCASB	SCASD
B	MOV AL,ib	MOV CL,ib	MOV DL,ib	MOV BL,ib	MOV AH,ib	MOV CH,ib	MOV DH,ib	MOV BH,ib	MOV eAX,iv	MOV eCX,iv	MOV eDX,iv	MOV eBX,iv	MOV eSP,iv	MOV eBP,iv	MOV eSI,iv	MOV eDI,iv
C	Shift rmb,ib	Shift rmv,ib	RET i2	RET	LES rm,rmp	LDS rm,rmp	MOV rmb, ib	MOV rmv, iv	ENTER i2, ib	LEAVE	RETF i2	RETF	INT 3	INT ib	INTO	IRET
D	Shift rmb, 1	Shift rmv, 1	Shift rmb,CL	Shift rmv,CL	AAM	AAD		XLATB	87 space	87 space	87 space	87 space	87 space	87 space	87 space	87 space
E	LOOPNE short	LOOPE short	LOOP short	JeCXZ short	IN AL,[ib]	IN eAX,[ib]	OUT [ib],AL	OUT [ib],eAX	CALL iv	JMP iv	JMP FAR s:m	JMP ib	IN AL,[DX]	IN eAX,[DX]	OUT [DX],AL	OUT [DX],eAX
F	LOCK		REPNE	REP REPE	HLT	CMC	Unary rmb	Unary rmv	CLC	STC	CLI	STI	CLD	STD	IncDec rmb	Indir rmv

Example

- Opcode byte 03 : ADD rv, rmv
  - rv indicates that the first operand must be a 16- or a 32-bit register
  - rmv means that the second operand must be a 16- or 32-bit register or memory operand
  - ADD EDI, EAX
  - ADD EDI, [1234H]

The type of information conveyed by the first byte

- One-byte suffices to completely specify an instruction.
  - F4 -> HLT
- First byte determines which operation is to be performed, but it does not specify the operands. Second (and third) byte(s) often carries the operand information (ModRM bytes)
  - 89 -> MOV rmv, rv

The type of information conveyed by the first byte

- First byte specify merely a general type of operation, and the ModRM byte is used to complete the specification of the operation as well as carry operand information
  - C1 -> Shift rmv, ib
  - It is just a pointer into Table 5.3
- It is just a pointer to Table 5.4
  - 0F

The type of information conveyed by the first byte

- If the first byte is in the range D8-DF, then the command is a floating point command
- There are also prefix bytes which modify the subsequent commands.
  - F0 -> LOCK

The ModRM byte

- Consist of three parts

Mod		Reg			R/M		
7	6	5	4	3	2	1	0

  - Bits 7 and 6 are the Mod bits. When the mod bits are 11, the R/M bits designate a register. Otherwise the R/M bits are used for memory coding
  - Bits 5, 4 and 3 are the Reg bits and are often designated using a slash notation, /r.
  - Bits 2, 1 and 0 are the R/M bits, and are used to specify or help specify a memory location except when the Mod bits are 11.

rv, rmv coding

- ADD EDI, EAX
  - Two Mod bits must be 11
  - R bits for EDI is 111
  - R/M bits for EAX is 000
  - Code is 1111 1000 = F8H

Mod		Reg			R/M		
1	1	1	1	1	0	0	0

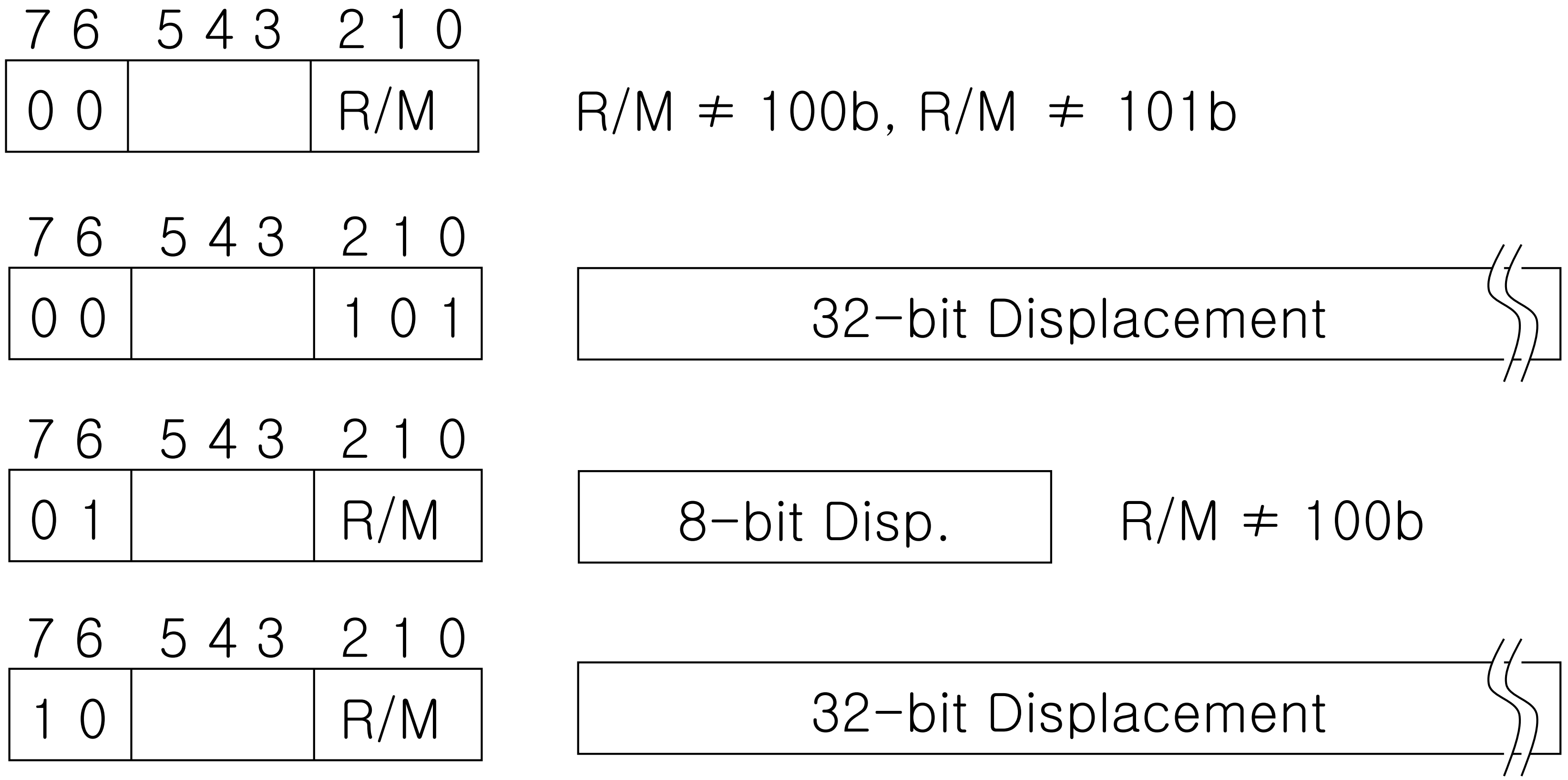
  - Combination with code 03
  - -> 03 F8

rv, rmv coding

- Mod bits usage
  - 11
    - ADD EDI, EAX
  - 00 (actual memory operand)
    - ADD EDI, [EAX]
  - 01 and 10 (immediate displacement either ib or iv)
    - ADD EDI, [EAX + 5] -> 03 78 05
    - ADD EDI, [EAX + 87654321H] -> 03 B8 21 43 65 87

One-Byte Address Modes – One-byte ModRM

- "Base reg + displacement" form ☜ Data addressing mode 참조



Examples

- mod = 00,   ADD EAX, [EDX]

	mod reg R/M
opcode	00 000 010

- mod = 01,   SUB EDI, [EDI+127]

	mod reg R/M	
opcode	01 111 111	01111111

- mod = 10,   ADD ESI, [EDI+12345678h]

	mod reg R/M	32-bit displacement			
opcode	10 110 111	01111000	01010110	00110100	00010010

- mod = 00 & RM = 101,   MOV ESI, [12345678h]

	mod reg R/M	32-bit displacement			
opcode	00 110 101	01111000	01010110	00110100	00010010

Examples

- mod = 01 & RM = 101,   MOV ESI, [EBP]

	mod reg R/M	
opcode	01 110 101	00000000

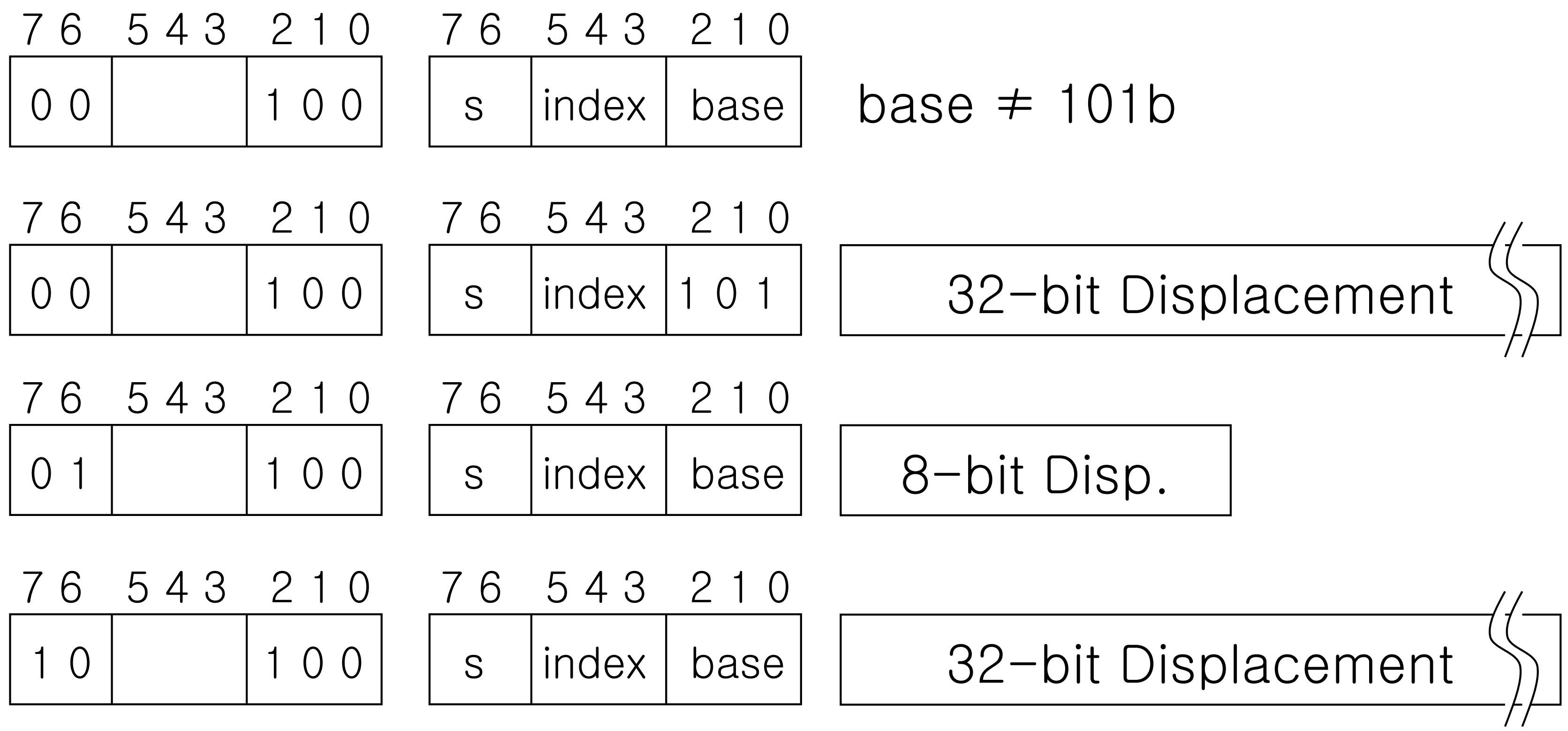
- mod = 10,   MOV [12345678h], 12345678h

	mod reg R/M		
opcode	00 000 101	32-bit disp.	32-bit immed.

Two-Byte Address Modes – Two-byte ModRM

- "Base reg + Index reg + scale factor + displacement" form

☜ Data addressing mode 참조



Examples

- MOV EAX, [EBX+ESI*2]

	mod	reg	2nd	s	idx	base
opcode	00	000	100	01	110	011

- MOV EAX, [999999+EDI+EAX*4]

	mod	reg	2nd	s	idx	base	
opcode	10	000	100	10	000	111	32-bit disp.

- MOV EAX, [TABLE+ESI*4]

	mod	reg	2nd	s	idx	base	
opcode	00	000	100	10	110	101	32-bit disp.

- MOV EDI, [ESP+24]

	mod	reg	2nd	s	idx	base	
opcode	01	111	100	00	100	100	00011000

rmv, rv coding

- MOV EDI, EAX
  - Using 8B -> 8B F8
  - Using 89 -> 89 C7
    - EDI goes into the least significant bits
    - EAX is coded in the R bits

Mod		Reg			R/M		
1	1	0	0	0	1	1	1

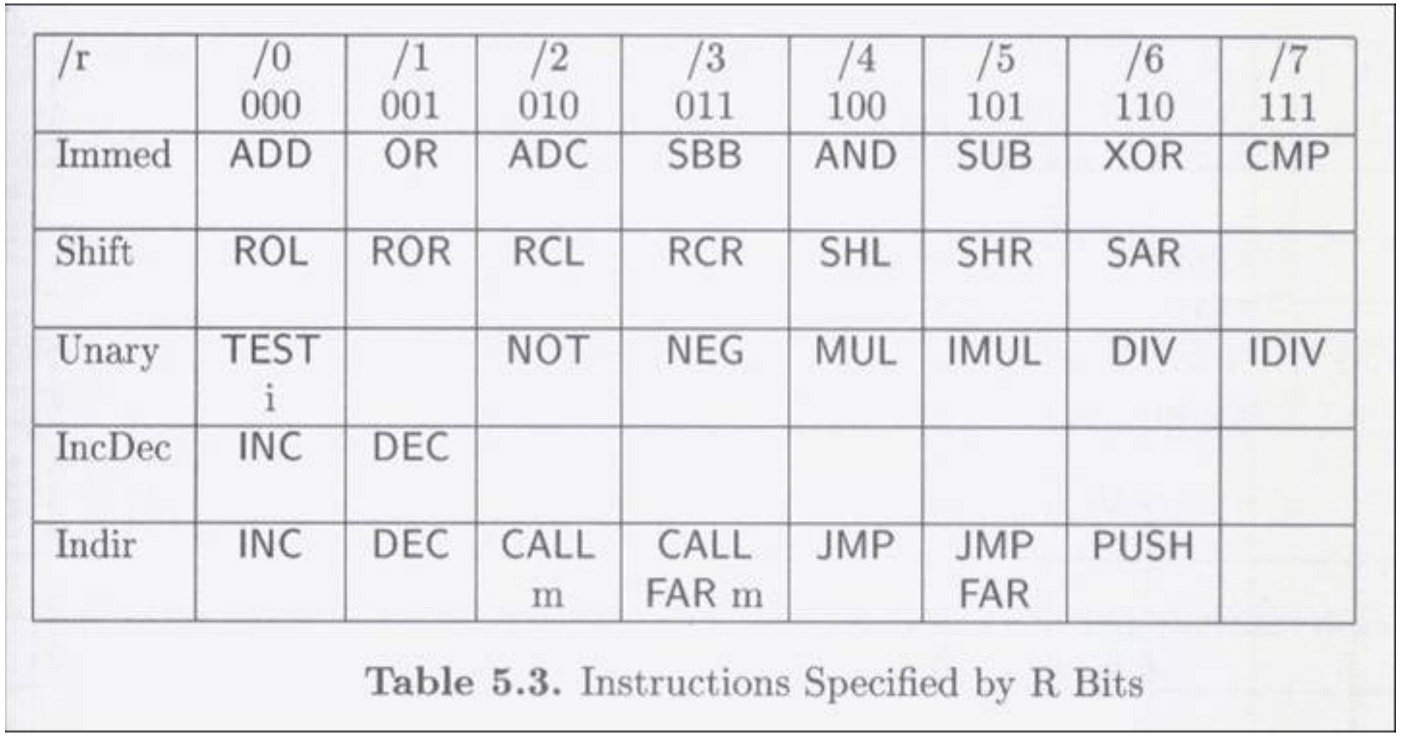
- 1100 0111 = C7H

Nonregister R bits

- The first byte is insufficient to specify an operation, but the R bits of a ModRM byte are used to complete the specification.
- SUB ECX, 32
  - 83 -> Immed rmv, ib
  - Immed has a SUB under /r = /5 = 101
  - Mod = 11
  - M = 001 for ECX
  - 32 = 20H
  - 83 (11 101 001) 20 = 83 E9 20

Nonregister R bits

- T 5.3

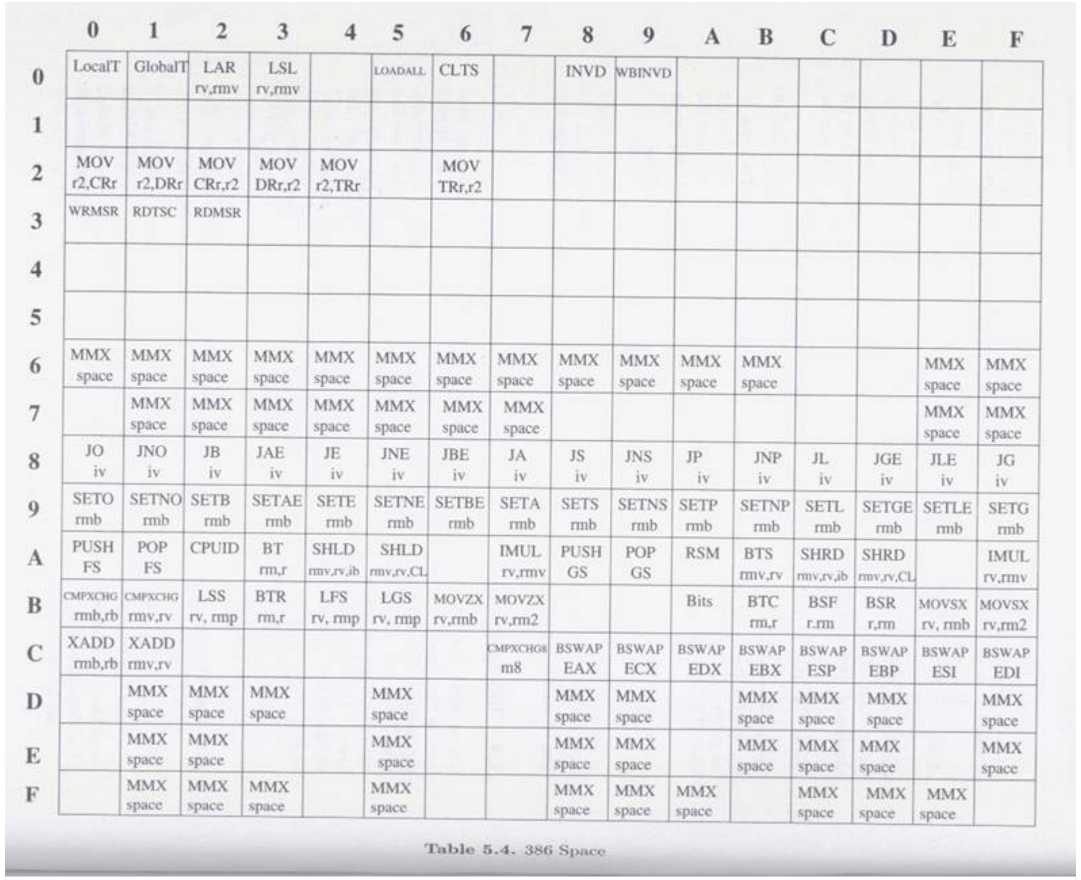


/r	/0 000	/1 001	/2 010	/3 011	/4 100	/5 101	/6 110	/7 111
Immed	ADD	OR	ADC	SBB	AND	SUB	XOR	CMP
Shift	ROL	ROR	RCL	RCR	SHL	SHR	SAR	
Unary	TEST i		NOT	NEG	MUL	IMUL	DIV	IDIV
IncDec	INC	DEC						
Indir	INC	DEC	CALL m	CALL FAR m	JMP	JMP FAR	PUSH	

Table 5.3. Instructions Specified by R Bits

386 space

- Open 386 space followed by 0F, most of them appears first on 386



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	LocalT	GlobalT	LAR rv,rmv	LSL rv,rmv		LOADALL	CLTS		INVD	WBINVD						
1																
2	MOV r2,CRr	MOV r2,DRr	MOV CRr,r2	MOV DRr,r2	MOV r2,TRr		MOV TRr,r2									
3	WRMSR	RDTSC	RDMSR													
4																
5																
6	MMX space	MMX space	MMX space	MMX space	MMX space	MMX space	MMX space	MMX space	MMX space	MMX space	MMX space	MMX space			MMX space	MMX space
7		MMX space	MMX space	MMX space	MMX space	MMX space	MMX space	MMX space							MMX space	MMX space
8	JO iv	JNO iv	JB iv	JAE iv	JE iv	JNE iv	JBE iv	JA iv	JS iv	JNS iv	JP iv	JNP iv	JL iv	JGE iv	JLE iv	JG iv
9	SETO rmb	SETNO rmb	SETB rmb	SETAE rmb	SETE rmb	SETNE rmb	SETBE rmb	SETA rmb	SETS rmb	SETNS rmb	SETP rmb	SETNP rmb	SETL rmb	SETGE rmb	SETLE rmb	SETG rmb
A	PUSH FS	POP FS	CPUID	BT rm,r	SHLD rmv,rv,ib	SHLD rmv,rv,CL		IMUL rv,rmv	PUSH GS	POP GS	RSM	BTS rmv,rv	SHRD rmv,rv,ib	SHRD rmv,rv,CL		IMUL rv,rmv
B	CMPXCHG rmb,rb	CMPXCHG rmv,rv	LSS rv, rmp	BTR rm,r	LFS rv, rmp	LGS rv, rmp	MOVZX rv,rmb	MOVZX rv,rm2			Bits	BTC rm,r	BSF r.rm	BSR r,rm	MOVSX rv, rmb	MOVSX rv,rm2
C	XADD rmb,rb	XADD rmv,rv						CMPXCHG8 m8	BSWAP EAX	BSWAP ECX	BSWAP EDX	BSWAP EBX	BSWAP ESP	BSWAP EBP	BSWAP ESI	BSWAP EDI
D		MMX space	MMX space	MMX space		MMX space			MMX space	MMX space		MMX space	MMX space	MMX space		MMX space
E		MMX space	MMX space			MMX space			MMX space	MMX space		MMX space	MMX space	MMX space		MMX space
F		MMX space	MMX space	MMX space		MMX space			MMX space	MMX space	MMX space		MMX space	MMX space	MMX space	

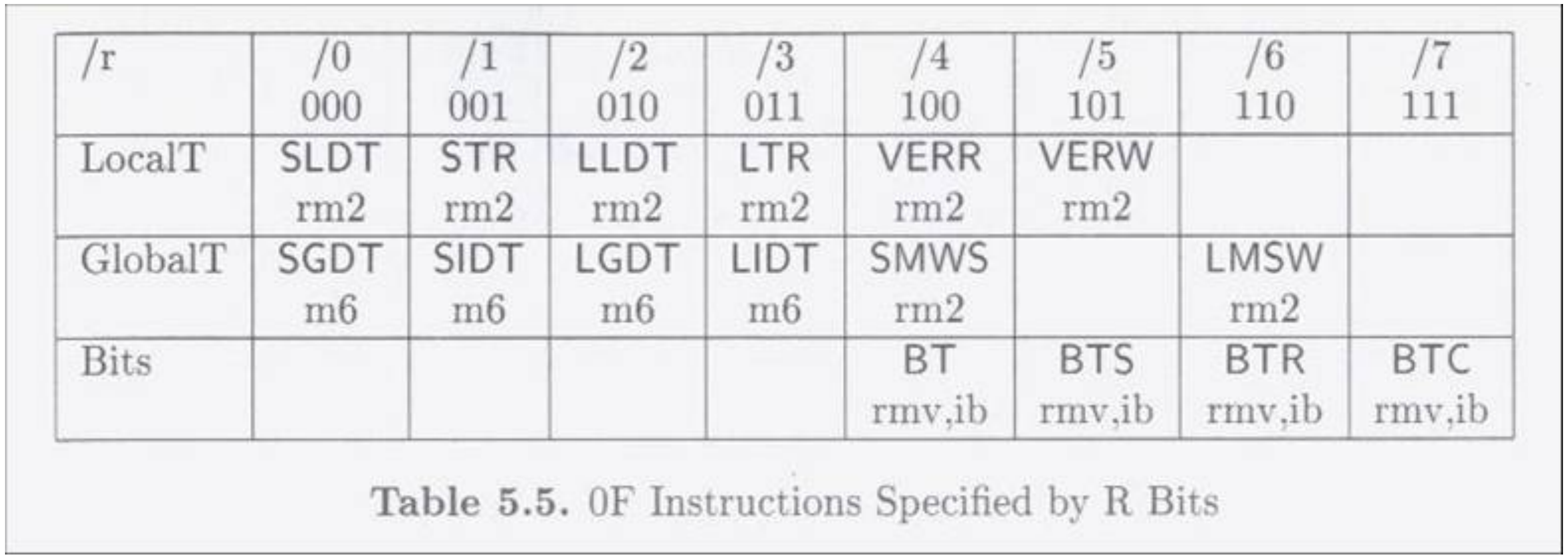
Table 5.4. 386 Space

386 space

- Sometimes one entry specifies exactly one instruction. For example, 0F CA is BSWAP EDX
- Sometimes a ModRM byte is needed to convey operand information. XADD instruction whose first two bytes are 0F C1
- Sometimes a ModRM byte is used to specify an operation. When 0F 01, we must check R bits of the ModRM (Table 5.5)
- MMX instructions

386 space

- Table 5.5



/r	/0 000	/1 001	/2 010	/3 011	/4 100	/5 101	/6 110	/7 111
LocalT	SLDT rm2	STR rm2	LLDT rm2	LTR rm2	VERR rm2	VERW rm2		
GlobalT	SGDT m6	SIDT m6	LGDT m6	LIDT m6	SMWS rm2		LMSW rm2	
Bits					BT rmv,ib	BTS rmv,ib	BTR rmv,ib	BTC rmv,ib

Table 5.5. 0F Instructions Specified by R Bits

32-bit vs. 16-bit code

- Code example

	32-bit	16-bit
MOV ECX, EAX	89 C1	66 89 C1
MOV CX, AX	66 89 C1	89 C1

- Code for 8-bit registers

```
AL  000        AH  100
CL  001        CH  101
DL  010        DH  110
BL  011        BH  111
```

  - MOV AL, CL -> 8A C1