80x86 - Instruction Set

ADDRESSING MODES

Types of Instruction

- Data Transfer Instructions
- Arithmetic Instructions
- Logical Instructions
- Branch and Program control Instructions

MOV Instruction

MOV destination , source

Addressing Modes

- The Processor executes an instruction – it performs the specified function on data
- Data- called operands
- May be a part of the instruction
- May reside in one of the internal registers of the µp
- May be stored at an address in memory

CISC architecture allows operation from memory
intel 80x86 are cisc

Addressing Modes

- Register Addressing if data is present in register alone
- Immediate Addressing if data is specified as part of instruction
- *Direct Addressing* address given as part of instruction
- *Register Indirect Addressing* address given stored in register
- *Base-plus-index addressing*

memory

- *Register relative addressing*
- *Base relative –plus-indexed addressing*
- *Scaled Indexed Addressing*

Addressing Modes

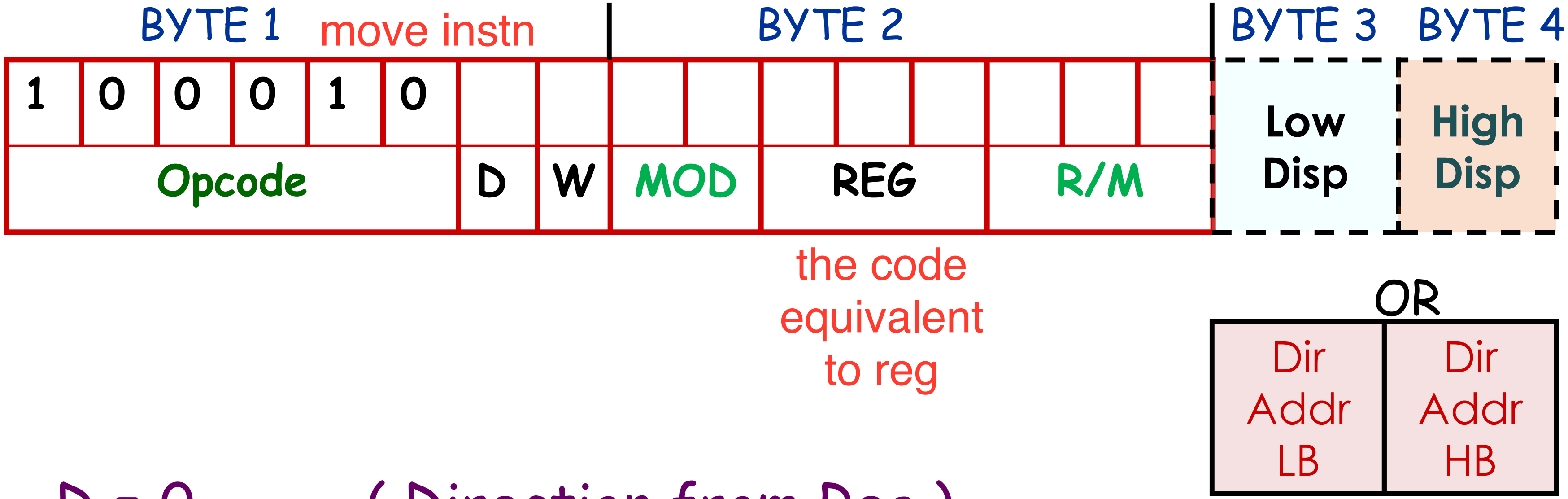
- Register Addressing
  - MOV AX,BX
- Immediate Addressing
  - MOV AX,1420$_H$
- *Direct Addressing*
  - *MOV AX,[2340$_H$]*
- *Register Indirect Addressing*
  - *MOV AX,[BX]*

16 bit addressing format supported by only 8086 and 80286(also supported by 80386 and 80546).
32 bit is supported purely by 80386 and 80546

Addressing Modes

- *Base-plus-index addressing*
  - *MOV AX,[BX+SI]*
- *Register relative addressing*
  - *MOV AX,10[BX]*
- *Base relative –plus-indexed addressing*
  - *MOV AX,[BX+SI+10]*
- *Scaled Indexed Addressing* supported only 80386 ownwards

used only in 8086 and 80286 (16 bit instruction format)



BYTE 1			BYTE 2			BYTE 3	BYTE 4
1 0 0 0 1 0						Low Disp	High Disp
Opcode	D	W	MOD	REG	R/M		

move instn

the code equivalent to reg

OR

Dir Addr LB	Dir Addr HB

D = 0 (Direction from Reg)
= 1 (Direction to Reg)

W = 0 (Data - byte) 8 bit data move
= 1 (Data - word) 16 bit data moved

MOD + R/M - Addressing Modes

later we’ll see what combination of mod+r/m mean

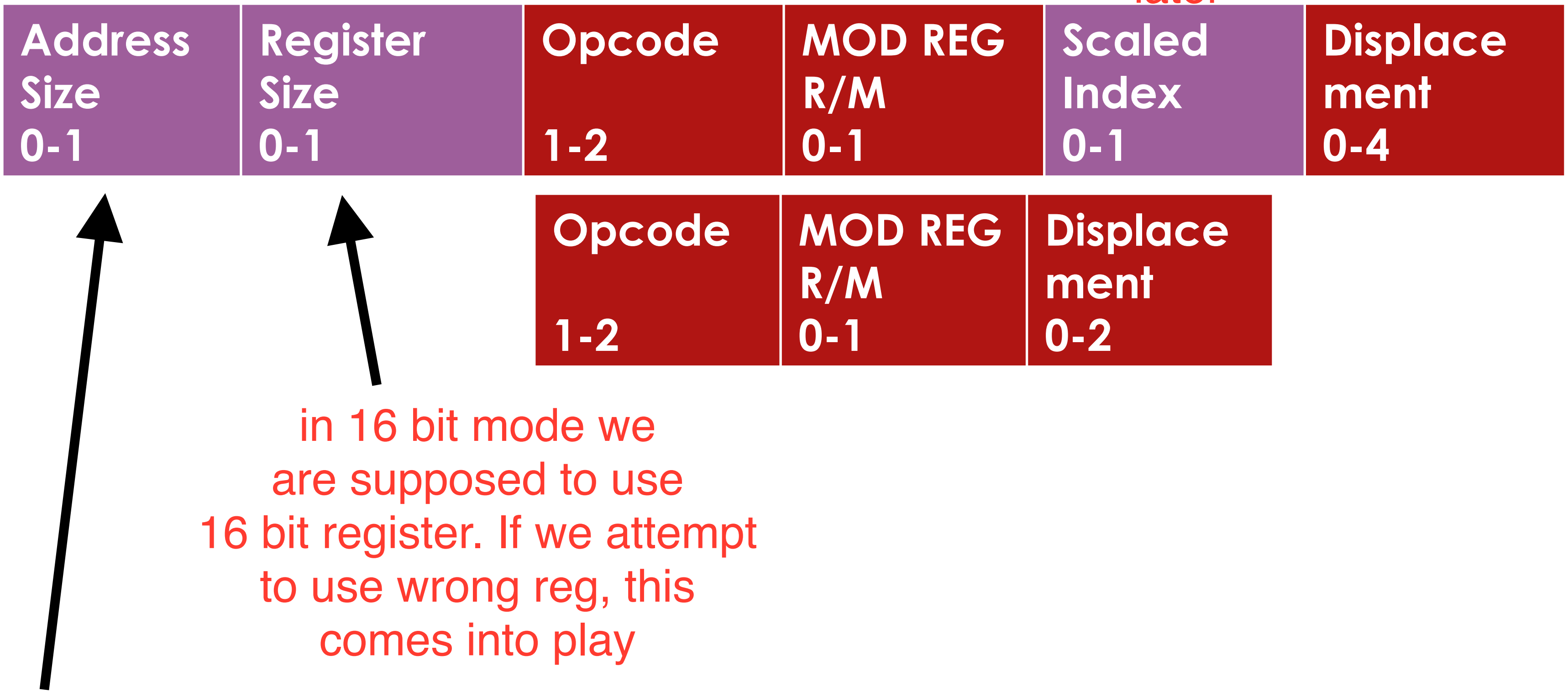
80386

- 16-bit mode of operation - DOS
- 32-bit mode of operation

downward compatibility

out of context- special purpose register:
These are registers other than AX,BX,CX,DX,BP,DI,SI
IP,SP,FLAGS register used internally by EU.
CS,DS,SS,ES - part of bus interface unit

32- bit instruction Format



later

Address Size 0-1	Register Size 0-1	Opcode 1-2	MOD REG R/M 0-1	Scaled Index 0-1	Displacement 0-4

Opcode 1-2	MOD REG R/M 0-1	Displacement 0-2

in 16 bit mode we
are supposed to use
16 bit register. If we attempt
to use wrong reg, this
comes into play

Address override- when we are in 16 bit mode of oper
and trying to use 32 bit or vice versa

32-bit addressing modes

- First two bytes are over-riding prefix
  - need not be used always
  - 1st modifies size of address
    - 16-bit mode – 32 bit addressing mode 67_H
    - 32-bit mode – 16 bit address mode 67_H
  - 2nd modifies size of register
    - 16-bit mode – 32 bit register 66_H
    - 32-bit mode – 16 bit register 66_H

EAX/AX/AL	000
EBX/BX/BL	011
ECX/CX/CL	001
EDX/DX/DL	010
ESP/SP/AH	100
EBP/BP/CH	101
ESI/SI/DH	110
EDI/DI/BH	111

REG

32 bit EAX
16 bit AX
8 bit AL
similar for every row

AX - accumulator
BX - base index register
(holds pointer value mostly)
CX - count register(string/loop)
DX - data register

SP - stack pointer
BP - base pointer
SI source Index pointer
DI destination index pointer

multipurpose
register

MOD	00	01	10	11	
R/M				W = 0	W = 1
000	[BX] + [SI]	[BX]+[SI] +d8	[BX]+[SI]+d16	AL	AX
001	[BX] + [DI]	[BX]+[DI]+d8	[BX]+[DI]+d16	CL	CX
010	[BP] + [SI]	[BP]+[SI]+d8	[BP]+[SI]+d16	DL	DX
011	[BP] + [DI]	[BP]+[DI]+d8	[BP]+[DI]+d16	BL	BX
100	[SI]	[SI]+d8	[SI]+d16	AH	SP
101	[DI]	[DI]+d8	[DI]+d16	CH	BP
110	d16	[BP] + d8	[BP] + d16	DH	SI
111	[BX]	[BX]+d8	[BX]+d16	BH	DI

Instruction Template

MOD R/M	00	01	10	11	
				W = 0	W = 1
000	EAX	EAX+d8	EAX+d32	AL	EAX
001	ECX	ECX+d8	ECX+d32	CL	ECX
010	EDX	EDX+d8	EDX+d32	DL	EDX
011	EBX	EBX+d8	EBX+d32	BL	EBX
100	Scaled Index	Scaled Index +d8	Scaled Index +d32	AH	ESP
101	d32	EBP+d8	EBP+d32	CH	EBP
110	ESI	ESI+d8	ESI+d32	DH	ESI
111	EDI	EDI+d8	EDI+d32	BH	EDI

Instruction Template

80x86 - Instruction Set

ADDRESSING MODE – REGISTER, IMMEDIATE

Register Addressing

- MOV AX,BX both AX and BX are 16 bits we cannot move 16 bit to 8 bit
- (AX) ← (BX)
- BX = $194A_H$
- AX = $194A_H$

after the move, AX contains value in B X
though BX also contains that value after

MOV AX,BX

1	0	0	0	1	0	0	1	1	1	0	1	1	0	0	0

89D8

1	0	0	0	1	0	1	1	1	1	0	0	0	0	1	1

8BC3

Immediate Addressing

- MOV AH,$4C_H$
- (AH) ← 0100 1100
- AX = 9844_H
- AX = $4C44_H$

MOV AH,$4C_H$ - 1011 W REG

1	0	1	1	0	1	0	0

B4 4C

Immediate Addressing

- MOV CX,AD4C$_H$
- (CX) ← 1010 1101 0100 1100
- CX = 9844$_H$
- CX = AD4C$_H$

MOV CX,AD4C$_H$ - 1011 W REG

1	0	1	1	1	0	0	1

B9 4C AD

Little Endian

lower byte specified before higher byte AD4C becomes 4CAD

Little Vs. Big Endian - 56 27

00000	27	00000	56
00001	56	00001	27

0000 and 0001 represent memory location

all intel go by little endian

Little Vs. Big Endian - A0 49 56 27

00000	27
00001	56
00002	49
00003	A0

00000	A0
00001	49
00002	56
00003	27

80x86 - Instruction Set

ADDRESSING MODE – DIRECT ADDRESSING

data present in memory

Direct Addressing

- MOV AX,[1234_H] within brackets offset
- (AX) ← DS:1234 whenever we are fetching data from memory by default we are using data segment register
- DS = 2000_H
- Address = 20000 + 1234 = 21234
- 21234_H 74
- 21235_H 82
- AX = 82 74_H

MOV AX,[1234_H]

1	0	0	0	1	0	1	1	0	0	0	0	0	1	1	0

8B06 34 12

Direct Addressing

- MOV [D600$_H$],BX
- DS = 2000$_H$
- Address = 20000 + D600 = 2D600
- BX = 8A 17$_H$
- 2D600$_H$ 17
- 2D601$_H$ 8A

MOV   [$D600_H$],BX

1	0	0	0	1	0	0	1	0	0	0	1	1	1	1	0

891E 00 D6

MOV [D600$_H$],BH

1	0	0	0	1	0	0	0	0	0	1	1	1	1	1	0

883E 00 D6

Direct Addressing

- MOV EAX,[1234_H]
- (EAX) ← DS:1234
- DS = 2000_H
- Address = 20000 + 1234 = 21234
- 21234_H 74
- 21235_H 82
- 21236_H A3
- 21237_H 45
- EAX = 45 A3 82 74_H

MOV EAX,[1234_H]

1	0	0	0	1	0	1	1	0	0	0	0	0	1	0	1

66 8B05 34 12

80x86 - Instruction Set

ADDRESSING MODE – REGISTER INDIRECT ADDRESSING

Register Indirect Addressing

- MOV AX,[BX]
- BX = 1234_H
- (AX) ← DS:1234
- DS = 2000_H
- Address = 20000 + 1234 = 21234
- 21234_H 74
- 21235_H 82
- AX = 82 74_H

MOV AX,[BX]

1	0	0	0	1	0	1	1	0	0	0	0	0	1	1	1

8B07

Register Indirect Addressing

- MOV [SI],BH
- SI = $D600_H$
- DS = 2000_H
- Address = 20000 + D600 = 2D600
- BX = 8A 17_H
- $2D600_H$ 8A

MOV [SI],BH

1	0	0	0	1	0	0	0	0	0	1	1	1	1	0	0

883C

Register Indirect Addressing

- MOV EAX,[BX]
- BX = 1234_H
- (EAX) ← DS:1234
- DS = 2000_H
- Address = 20000 + 1234 = 21234
- 21234_H 74
- 21235_H 82
- 21236_H A3
- 21237_H 45
- EAX = 45 A3 82 74_H

MOV EAX,[BX]

1	0	0	0	1	0	1	1	0	0	0	0	0	1	1	1

8B07

16-bit mode - 66 8B07

32-bit mode - 67 8B07

Register Indirect Addressing

- MOV EAX,[ECX]
- ECX = 0000 1234$_H$
- (EAX) ← DS:1234
- DS = 2000$_H$
- Address = 20000 + 1234 = 21234
- 21234$_H$ 74
- 21235$_H$ 82
- 21236$_H$ A3
- 21237$_H$ 45
- EAX = 45 A3 82 74$_H$

MOV EAX,[ECX]

1	0	0	0	1	0	1	1	0	0	0	0	0	0	0	1

8B01

32-bit mode - 8B01

16-bit mode - 67 66 8B01

80x86 - Instruction Set

ADDRESSING MODE – REGISTER RELATIVE ADDRESSING

Register Relative Addressing

- MOV AX, 34[BX]
- MOV AX,[BX+34]
- BX = 1200_H
- (AX) ← DS:1200+34
- DS = 2000_H
- Address = 20000 + 1200 + 34 = 21234
- 21234_H 74
- 21235_H 82
- AX = 82 74_H

MOV AX,[BX+34]

1	0	0	0	1	0	1	1	0	1	0	0	0	1	1	1

8B47 34

Register Relative Addressing

- MOV [SI+600],BH
- SI = $D000_H$
- DS = 2000_H
- Address = 20000 + D000+600 = 2D600
- BX = 8A 17_H
- $2D600_H$ 8A

MOV [SI+600],BH

1	0	0	0	1	0	0	0	1	0	1	1	1	1	0	0

88BC 00 06

Register Relative Addressing

- MOV EAX,[ECX + 234_H]
- ECX = 0000 1000_H
- (EAX) ← DS:1234
- DS = 2000_H
- Address = 20000 + 1000 + 234 = 21234
- 21234_H 74
- 21235_H 82
- 21236_H A3
- 21237_H 45
- EAX = 45 A3 82 74_H

MOV EAX,[ECX+234_H]

1	0	0	0	1	0	1	1	1	0	0	0	0	0	0	1

8B81 34 02 followed by four zeroes not shown here.

32-bit mode - 8B81 34 02

16-bit mode - 67 66 8B01 34 02

80x86 - Instruction Set

ADDRESSING MODE – BASED INDEXED, BASE – RELATIVE INDEXED

Based plus Indexed Addressing

- MOV AX, [BX+SI]
- BX = 1200_H
- SI = 0034_H
- (AX) ← DS:1200+34
- DS = 2000_H
- Address = 20000 + 1200 + 34 = 21234
- 21234_H 74
- 21235_H 82
- AX = 82 74_H

MOV AX,[BX+SI]

1	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0

8B00

Base Relative plus Indexed Addressing

- MOV [BX+SI+600],BH
- SI = 1000_H
- BX = $C000_H$
- DS = 2000_H
- Address = 20000 + C000+1000+600 = 2D600
- BX = 8A 17_H
- $2D600_H$ 8A

MOV [BX+SI+600],BH

1	0	0	0	1	0	0	0	1	0	1	1	1	0	0	0

88B8 00 06

Based plus Indexed Addressing

- MOV EAX, [BX+SI]
- BX = 1200_H
- SI = 0034_H
- (AX) ← DS:1200+34
- DS = 2000_H
- Address = 20000 + 1200 + 34 = 21234
- 21234_H 74
- 21235_H 82
- 21236_H A3
- 21237_H 45
- EAX = 45 A3 82 74_H

MOV EAX,[BX+SI]

1	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0

66 8B00

MOV EAX,[BX+SI]

1	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0

67 8B00

MOV AX,[BX+SI]

1	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0

67 66 8B00

80x86 - Instruction Set

ADDRESSING MODE –SCALED INDEXED

Scaled Indexed Addressing

- MOV EAX,[EBX+4*ECX]
- EBX = 0000 1230_H
- ECX = 0000 0001_H
- (AX) ← DS:1230+ 4*1
- DS = 2000_H
- Address = 20000 + 1230 + 4*1 = 21234
- 21234_H 74
- 21235_H 82
- 21236_H A3
- 21237_H 45
- EAX = 45 A3 82 74_H

MOV EAX,[EBX+4 * ECX]

1	0	0	0	1	0	1	1	0	0	0	0	0	1	0	0

S	S	In	d	x	B	as	e
1	0	0	0	1	0	1	1

67 66 8B008B

Addressing Modes

- Register Addressing
- Immediate Addressing
- *Direct Addressing*
- *Register Indirect Addressing*
- *Base-plus-index addressing*
- *Register relative addressing*
- *Base relative –plus-indexed addressing*
- *Scaled Indexed Addressing*

80x86 - Instruction Set

SEGMENT OVERRIDES

Default 16 bit segment and offset address combinations

Segment	offset	special purpose
CS	IP	Instruction Address
SS	SP (or) BP	Stack address
DS	BX,DI,SI an 8-bit number 16 - bit number	Data address
ES	DI for string Instructions	String destination address

Addressing Modes

- Register Addressing
- Immediate Addressing
- *Direct Addressing*
- *Register Indirect Addressing*
- *Base-plus-index addressing*
- *Register relative addressing*
- *Base relative –plus-indexed addressing*
- *Scaled Indexed Addressing*

Segment Override

- MOV AX,[BX]
- BX = 1234_H
- (AX) ← DS:1234
- MOV AX, ES:[BX]

Segment Override Prefix

Segment	Prefix Value
ES	26_H
CS	$2E_H$
SS	36_H
DS	$3E_H$
FS	64_H
GS	65_H

MOV AX,ES:[BX]

1	0	0	0	1	0	1	1	0	0	0	0	0	1	1	1

26 8B07

MOV DS, AX

1000110 = 3E(prefix of DS)

always 0

1000 11D0 MOD SEGREG R/M

1000 1110 11 011 000

8ED8

Seg	Code
ES	000
CS	001
SS	010
DS	011
FS	100
GS	101