

# Microprocessor & Interfacing

## Lecture 15

### 8086 Addressing Modes & Instructions



**PARUL BANSAL**  
**ASST PROFESSOR**  
**ECS DEPARTMENT**  
**DRONACHARYA COLLEGE OF ENGINEERING**

# Contents



- Addressing Modes
- Instructions
  - Data transfer
  - Arithmetic
  - Logical
  - String
  - Program transfer
  - Processor Control

# Addressing Modes



1. Implied Addressing – The data value/data address is implicitly associated with the instruction
2. Register Addressing – The data is specified by referring the register or the register pair in which the data is present
3. Immediate Addressing – The data itself is provided in the instruction
4. Direct Addressing – The instruction operand specifies the memory address where data is located

# Cont..



5. Register indirect addressing – The instruction specifies a register containing an address, where data is located

Based - 8-bit or 16-bit instruction operand is added to the contents of a base register (BX or BP), the resulting value is a pointer to location where data resides

Indexed - 8-bit or 16-bit instruction operand is added to the contents of an index register (SI or DI), the resulting value is a pointer to location where data resides

## Cont..



6. Based Indexed - the contents of a base register (BX or BP) is added to the contents of an index register (SI or DI), the resulting value is a pointer to location where data resides
7. Based Indexed with displacement - 8-bit or 16-bit instruction operand is added to the contents of a base register (BX or BP) and index register (SI or DI), the resulting value is a pointer to location where data resides

# Data Transfer Instructions



General-Purpose	
MOV	Move byte or word
PUSH	Push word onto stack
POP	Pop word off stack
PUSHA	Push registers onto stack
POPA	Pop registers off stack
XCHG	Exchange byte or word
XLAT	Translate byte

# Cont..



<b>Input/Output</b>	
IN	Input byte or word
OUT	Output byte or word
<b>Address Object and Stack Frame</b>	
LEA	Load effective address
LDS	Load pointer using DS
LES	Load pointer using ES
ENTER	Build stack frame
LEAVE	Tear down stack frame
<b>Flag Transfer</b>	
LAHF	Load AH register from flags
SAHF	Store AH register in flags
PUSHF	Push flags from stack
POPF	Pop flags off stack

# Arithmetic Instructions



<b>Addition</b>	
ADD	Add byte or word
ADC	Add byte or word with carry
INC	Increment byte or word by 1
AAA	ASCII adjust for addition
DAA	Decimal adjust for addition
<b>Subtraction</b>	
SUB	Subtract byte or word
SBB	Subtract byte or word with borrow
DEC	Decrement byte or word by 1
NEG	Negate byte or word
CMP	Compare byte or word
AAS	ASCII adjust for subtraction
DAS	Decimal adjust for subtraction



## Cont..



<b>Multiplication</b>	
MUL	Multiply byte or word unsigned
IMUL	Integer multiply byte or word
AAM	ASCII adjust for multiplication
<b>Division</b>	
DIV	Divide byte or word unsigned
IDIV	Integer divide byte or word
AAD	ASCII adjust for division
CBW	Convert byte to word
CWD	Convert word to double-word

# Number Representation



Hex	Bit Pattern	Unsigned Binary	Signed Binary	Unpacked Decimal	Packed Decimal
07	0 0 0 0 0 1 1 1	7	+7	7	7
89	1 0 0 0 1 0 0 1	137	-119	invalid	89
C5	1 1 0 0 0 1 0 1	197	-59	invalid	invalid

# Logical Instructions



Logicals	
NOT	"Not" byte or word
AND	"And" byte or word
OR	"Inclusive or" byte or word
XOR	"Exclusive or" byte or word
TEST	"Test" byte or word
Shifts	
SHL/SAL	Shift logical/arithmetic left byte or word
SHR	Shift logical right byte or word
SAR	Shift arithmetic right byte or word
Rotates	
ROL	Rotate left byte or word
ROR	Rotate right byte or word
RCL	Rotate through carry left byte or word
RCR	Rotate through carry right byte or word

# String Instructions



REP	Repeat
REPE/REPZ	Repeat while equal/zero
REPNE/REPNZ	Repeat while not equal/not zero
MOVSB/MOVSX	Move byte string/word string
MOVS	Move byte or word string
INS	Input byte or word string
OUTS	Output byte or word string
CMPS	Compare byte or word string
SCAS	Scan byte or word string
LODS	Load byte or word string
STOS	Store byte or word string

# Program Transfer Instructions



## Conditional Transfers

JA/JNBE	Jump if above/not below nor equal
JAE/JNB	Jump if above or equal/not below
JB/JNAE	Jump if below/not above nor equal
JBE/JNA	Jump if below or equal/not above
JC	Jump if carry
JE/JZ	Jump if equal/zero
JG/JNLE	Jump if greater/not less nor equal
JGE/JNL	Jump if greater or equal/not less
JL/JNGE	Jump if less/not greater nor equal
JLE/JNG	Jump if less or equal/not greater
JNC	Jump if not carry
JNE/JNZ	Jump if not equal/not zero
JNO	Jump if not overflow
JNP/JPO	Jump if not parity/parity odd
JNS	Jump if not sign
JO	Jump if overflow
JP/JPE	Jump if parity/parity even
JS	Jump if sign

# Cont..



<b>Unconditional Transfers</b>	
CALL	Call procedure
RET	Return from procedure
JMP	Jump
<b>Iteration Control</b>	
LOOP	Loop
LOOPE/LOOPZ	Loop if equal/zero
LOOPNE/LOOPNZ	Loop if not equal/not zero
JCXZ	Jump if register CX=0
<b>Interrupts</b>	
INT	Interrupt
INTO	Interrupt if overflow
BOUND	Interrupt if out of array bounds
IRET	Interrupt return

# Processor Control Instructions



Flag Operations	
STC	Set Carry flag
CLC	Clear Carry flag
CMC	Complement Carry flag
STD	Set Direction flag
CLD	Clear Direction flag
STI	Set Interrupt Enable flag
CLI	Clear Interrupt Enable flag
External Synchronization	
HLT	Halt until interrupt or reset
WAIT	Wait for $\overline{\text{TEST}}$ pin active
ESC	Escape to external processor
LOCK	Lock bus during next instruction
No Operation	
NOP	No operation