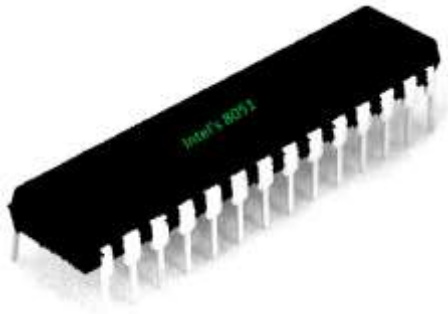
SNS COLLEGE OF ENGINEERING

(Autonomous)

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

19EC401 – MICROPROCESSORS AND MICROCONTROLLERS



8086 Instruction

INSTRUCTION SET BASICS

- **Instruction:-** An instruction is a **binary pattern** designed inside a microprocessor to perform a **specific function**.

- **Opcode:-** It stands for operational code. It specifies the **type of operation to be performed by CPU**. It is the first field in the machine language instruction format.
- E.g. 08 is the opcode for instruction “MOV X,Y”.

- **Operand:-** We can also say it as **data on which operation should act**. operands may be register values or memory values. The CPU executes the instructions using information present in this field. It may be 8-bit data or 16-bit data.

INSTRUCTION SET BASICS

- **Assembler:-** it converts the instruction into sequence of binary bits, so that this bits can be read by the processor.

- **Mnemonics:-** these are the **symbolic codes** for either instructions or commands to perform a particular function.
- **E.g. MOV, ADD, SUB etc.**

TYPES OF INSTRUCTION SET OF 8086 MICROPROCESSOR

(1). Data Copy/Transfer instructions.
(2).Arithmetic & Logical instructions.
(3) Branch instructions.
(4) Loop instructions.
(5) Machine Control instructions.
(6) Flag Manipulation instructions.
(7) Shift & Rotate instructions.
(8) String instructions.

DATA COPY/TRANSFER INSTRUCTIONS

MOV destination,source

- There will be transfer of data from source to destination.
- Eg
- (1)MOV CX,037A H;
- (2)MOV AL,BL;
- (3)MOV BX,[0301 H];

BEFORE EXECUTION

AX	2000H

MOV BX ,AX

AFTER EXECUTION

BX	2000H

BEFORE EXECUTION

AH		AL	
BH		BL	
CH		CL	
DH		DL	

40

MOV CL,M

AFTER EXECUTION

AH		AL	
BH		BL	
CH		CL	40
DH		DL	

40

STACK POINTER

- It is a 16-bit register, contains the address of the data item currently on top of the stack.
- Stack operation includes pushing (providing) data on to the stack and popping (taking)data from the stack.
- **Pushing** operation decrements stack pointer and **Popping** operation increments stack pointer. i.e. there is a last in first out **(LIFO)** operation.

PUSH SOURCE

- This instruction pushes the contents of specified source on to the stack.
- Ex: PUSH AX;
- In this stack pointer is decremented by 2.
- The higher byte data is pushed first (SP-1).
- Then lower byte data is pushed (SP-2).

POP DESTINATION

- This instruction pops (takes) the contents of specified destination.
- Ex: POP AX;
- In this stack pointer is incremented by 2.
- The lower byte data is popped first (SP+1).
- Then higher byte data is popped (SP+2).

XCHG Destination,source

- This instruction exchanges contents of Source with destination.
- It cannot exchange two memory locations directly.
- The contents of AL are exchanged with BL.
- The contents of AH are exchanged with BH.

- E.g.

(1).XCHG BX,AX ;

(2).XCHG [5000H],AX ;

IN AL/AX, 8-bit/16-bit port address

- It reads from the specified port address.
- It copies data to accumulator from a port with 8- bit or 16-bit address.
- DX is the only register is allowed to carry port address.
- Eg.

(1) IN AL, 80H;

OUT 8-bit/16-bit port address,AL/AX

- It writes to the specified port address.
- It copies contents of accumulator to the port with 8-bit or 16-bit address.
- DX is the only register is allowed to carry port address.
- Eg.

(1) OUT 80H,AL;

XLAT

- Also known as translate instruction.
- It is used to find out codes in case of code conversion.
- i.e.it translates code of the key pressed to the corresponding 7-segment code.
- After execution this instruction contents of AL register always gets replaced.
- E.g.XLAT;

LEA 16-bit register (source),address (dest.)

- **LEA** Also known as **L**oad **E**ffective **A**ddress (**LEA**).
- It loads effective address formed by the destination into the source register.

- Eg.

(1) LEA BX,Address;

LDS 16-bit register (source), address (dest.);
LES 16-bit register (source),address (dest.);

- **LDS** Also known as **L**oad **D**ata **S**egment (**LDS**).
- **LES** Also known as **L**oad **E**xtra **S**egment (**LES**).
- It loads the contents of DS (Data Segment) or ES (Extra Segment) & contents of the destination to the contents of source register.

- E.g.

(1) LDS BX,5000H;

(2) LES BX,5000H;

L A H F:- This instruction loads the A H register from the contents of lower byte of the flag register.

- This command is used to observe the status of the all conditional flags of flag register.

E.g. L A H F;

DATA TRANSFER INSTRUCTIONS summary

Mnemonic	Meaning	Format	Operation
MOV	Move	Mov D,S	(S) → (D)
XCHG	Exchange	XCHG D,S	(S) (D)
LEA	Load Effective Address	LEA Reg16,EA	EA → (Reg16)
PUSH	pushes the operand into top of stack.	PUSH BX	
POP	pops the operand from top of stack to Des.	POP BX	
IN	transfers the operand from specified port to accumulator register.	IN AX,0028	
OUT	transfers the operand from accumulator to specified port.	OUT 0028,BX	

ARITHMETIC INSTRUCTIONS

- These instructions perform the operations like:

- Addition,
- Subtraction,
- Increment,
- Decrement.

A D D destination,source

- This instruction adds the contents of source operand with the contents of destination operand.
- The source may be immediate data, memory location or register.
- The destination may be memory location or register.
- The result is stored in destination operand.
- A X is the default destination register.
- E.g.(1).A D D A X,2020H ;

  (2).A D D A X,B X;.

BEFORE EXECUTION

AH	10	AL	10

ADD AX ,2020H

AFTER EXECUTION

AH	30	AL	30

1010
+2020
3030

BEFORE EXECUTION

AH	10	AL	10
BH	20	BL	20

ADD AX,BX

AFTER EXECUTION

AH	30	AL	30
BH	20	BL	20

A D C destination, source

- This instruction adds the contents of source operand with the contents of destination operand with carry flag bit.
- The source may be immediate data, memory location or register.
- The destination may be memory location or register.
- The result is stored in destination operand.
- A X is the default destination register.

- E.g.(1).A D C A X ,2020H ;

  (2).A D C A X ,BX ;

I N C source

- This instruction increases the contents of source operand by 1.
- The source may be memory location or register.
- The source can not be immediate data.
- The result is stored in the same place.

- E.g.(1).IN C A X ;

  (2).IN C [5000H];.

BEFORE EXECUTION

AH	10	AL	10

INC AX

AFTER EXECUTION

AH	10	AL	11

BEFORE EXECUTION

5000H	1010

INC [5000H]

AFTER EXECUTION

5000H	1011

D E C source

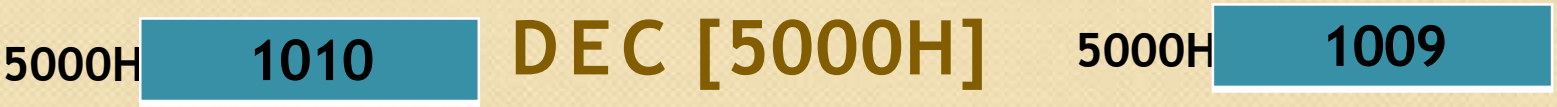
- This instruction decreases the contents of source operand by 1.
- The source may be memory location or register.
- The source can not be immediate data.
- The result is stored in the same place.
- E.g.(1).D E C A X ;
- (2).D E C [5000H];

BEFORE EXECUTION

AFTER EXECUTION

BEFORE EXECUTION

AFTER EXECUTION



- **AAA (ASCII Adjust after Addition):**
  - The data entered from the terminal is in ASCII format.
  - In ASCII, 0 – 9 are represented by 30H – 39H.
  - This instruction allows us to add the ASCII codes.
  - This instruction does not have any operand.
- **Other ASCII Instructions:**
  - **AAS** (ASCII Adjust after Subtraction)
  - **AAM** (ASCII Adjust after Multiplication)
  - **AAD** (ASCII Adjust Before Division)

- **DAA (Decimal Adjust after Addition)**
  - It is used to make sure that the result of adding two BCD numbers is adjusted to be a correct BCD number.
  - It only works on AL register.
- **DAS (Decimal Adjust after Subtraction)**
  - It is used to make sure that the result of subtracting two BCD numbers is adjusted to be a correct BCD number.
  - It only works on AL register.

MUL operand

- Unsigned Multiplication.
- Operand contents are positively signed.
- Operand may be general purpose register or memory location.
- If operand is of 8-bit then multiply it with contents of AL.
- If operand is of 16-bit then multiply it with contents of AX.
- Result is stored in accumulator (AX).
- **E.g. MUL BH**

IMUL operand

- Signed Multiplication.
- Operand contents are negatively signed.
- Operand may be general purpose register, memory location or index register.
- If operand is of 8-bit then multiply it with contents of AL.
- If operand is of 16-bit then multiply it with contents of AX.
- Result is stored in accumulator (AX).
- E.g. IMUL BH

DIV operand

- Unsigned Division.
- Operand may be register or memory.
- Operand contents are positively signed.
- Operand may be general purpose register or memory location.
- AL=AX /Operand (8-bit/16-bit) & AH =Remainder.
- E.g. **DIV BL**

IDIV operand

- Signed Division.
- Operand may be register or memory.
- Operand contents are negatively signed.
- Operand may be general purpose register or memory location.
- AL=AX /Operand (8-bit/16-bit) & AH =Remainder.

Ex: IDIV BL

ARITHMETIC INSTRUCTIONS

Mnemonic	Meaning	Format	Operation
SUB	Subtract	SUB D,S	(D) - (S) → (D) Borrow → (CF)
SBB	Subtract with borrow	SBB D,S	(D) - (S) - (CF) → (D)
DEC	Decrement by one	DEC D	(D) - 1 → (D)
NEG	Negate	NEG D	
DAS	Decimal adjust for subtraction	DAS	Convert the result in AL to packed decimal format
AAS	ASCII adjust for subtraction	AAS	(AL) difference (AH) dec by 1 if borrow
ADD	Addition	ADD D,S	(S)+(D) → (D) carry → (CF)
ADC	Add with carry	ADC D,S	(S)+(D)+(CF) → carry → (CF) (D)
INC	Increment by one	INC D	(D)+1 → (D)
AAA	ASCII adjust for addition	AAA	If the sum is >9, AH is incremented by 1
DAA	Decimal adjust for addition	DAA	Adjust AL for decimal Packed BCD

LOGICAL (or) Bit Manipulation Instructions

- AND
  - Especially used in clearing certain bits (masking) xxxx xxxx **AND** 0000 1111 = 0000 xxxx (clear the first four bits)
    - Examples: AND BL, 0FH

- OR
  - Used in setting certain bits

    xxxx xxxx **OR** 0000 1111 = xxxx 1111 (Set the upper four bits)

- XOR
  - Used in Inverting bits

    xxxx xxxx XOR 0000 1111 = xxxxx'x'x'x'

    -

SHL Instruction

The SHL (shift left) instruction performs a logical left shift on the destination operand, filling the lowest bit with 0.

SHR Instruction

The SHR (shift right) instruction performs a logical right shift on the destination operand.The highest bit position is filled with a zero.

SAR Instruction

SAR (shift arithmetic right) performs a right arithmetic shift on the destination operand

Branching Instructions (or) Program Execution Transfer Instructions

- These instructions cause change in the sequence of the execution of instruction.
- This change can be through a condition or sometimes unconditional.
- The conditions are represented by flags.
- **JMP Des:**
  - This instruction is used for unconditional jump from one place to another.

- **CALLDes:**
  - This instruction is used to call a subroutine or function or procedure.
  - The address of next instruction after CALL is saved onto stack.
- **RET:**
  - It returns the control from procedure to calling program.
  - Every CALL instruction should have a RET.

Conditional Jump Table

Mnemonic	Meaning
JA	Jump if Above
JAE	Jump if Above or Equal
JB	Jump if Below
JBE	Jump if Below or Equal
JC	Jump if Carry
JE	Jump if Equal
JNC	Jump if Not Carry
JNE	Jump if Not Equal
JNZ	Jump if Not Zero
JPE	Jump if Parity Even
JPO	Jump if Parity Odd
JZ	Jump if Zero

- **Loop Des:**
  - This is a looping instruction.
  - The number of times looping is required is placed in the CX register.
  - With each iteration, the contents of CX are decremented.
  - ZF is checked whether to loop again or not.

STRING INSTRUCTION

- String in assembly language is just a sequentially stored bytes or words.
- There are very strong set of string instructions in 8086.
- By using these string instructions, the size of the program is considerably reduced.

CMPS Des, Src:

- It compares the string bytes or words.

SCAS String:

- It scans a string
- It compares the String with byte in AL or with word in AX.

- **MOVS / MOVSB / MOVSW:**
  - It causes moving of byte or word from one string to another.
  - In this instruction, the source string is in Data Segment and destination string is in Extra Segment.
  - SI and DI store the offset values for source and destination index.

- **REP (Repeat):**
  - This is an instruction prefix.
  - It causes the repetition of the instruction until CX becomes zero.
  - E.g : REP MOVSB STR1, STR2
    - It copies byte by byte contents.
    - REP repeats the operation MOVSB until CX becomes zero.

Processor Control Instructions

- These instructions control the processor itself.
- 8086 allows to control certain control flags that:
  - causes the processing in a certain direction
  - processor synchronization if more than one microprocessor attached.

- **STC**
  - It sets the carry flag to 1.
- **CLC**
  - It clears the carry flag to 0.
- **CMC**
  - It complements the carry flag

STD:

- It sets the direction flag to 1.
- If it is set, string bytes are accessed from higher memory address to lower memory address.

CLD:

- It clears the direction flag to 0.
- If it is reset, the string bytes are accessed from lower memory address to higher memory address.

- **HLT** instruction – HALT processing
  - The HLT instruction will cause the 8086 to stop fetching and executing instructions.
- **NOP** instruction
  - this instruction simply takes up three clock cycles and does no processing.
- **LOCK** instruction
  - this is a prefix to an instruction. This prefix makes sure that during execution of the instruction, control of system bus is not taken by other microprocessor.
- **WAIT** instruction
  - this instruction takes 8086 to an idle condition. The CPU will not do any processing during this.

Bit Manipulation Instructions(Logical Instructions)

Mnemonic	Meaning	Format	Operation
AND	Logical AND	AND D,S	(S) · (D) → (D)
OR	Logical Inclusive OR	OR D,S	(S)+(D) → (D)
XOR	Logical Exclusive OR	XOR D,S	(S) ⊕ (D) → (D)
NOT	LOGICAL NOT	NOT D	(D) → (D)

Shift & Rotate Instructions

Mnemonic	Meaning	Format
SAL/SHL	Shift arithmetic Left/ Shift Logical left	SAL/SHL D, Count
SHR	Shift logical right	SHR D, Count
SAR	Shift arithmetic right	SAR D, Count

Mnemonic	Meaning	Format
ROL	Rotate Left	ROL D,Count
ROR	Rotate Right	ROR D,Count
RCL	Rotate Left through Carry	RCL D,Count
RCR	Rotate right through Carry	RCR D,Count

BRANCHING OR PROGRAM EXECUTION TRANSFER INSTRUCTIONS

- **CALL** - call a subroutine
- **RET** - returns the control from procedure to calling program
- **JMP Des** – Unconditional Jump
- **Jxx Des** – conditional Jump (ex: JC 8000)
- **Loop Des**

STRING INSTRUCTIONS

- **CMPS Des, Src -** compares the string bytes
- **SCAS String -** scans a string
- **MOVS / MOVSB / MOVSW -** moving of byte or word
- **REP (Repeat) -** repetition of the instruction

PROCESSOR CONTROL INSTRUCTIONS

- **STC** – set the carry flag (CF=1)
- **CLC** – clear the carry flag (CF=0)
- **STD** – set the direction flag (DF=1)
- **CLD** – clear the direction flag (DF=0)
- **HLT** – stop fetching & execution
- **NOP** – no operation(no processing)
- **LOCK** - control of system bus is not taken by other µP
- **WAIT** - CPU will not do any processing
- **ESC** - µP does NOP or access a data from memory for coprocessor


ASSESSMENT

THANK YOU