Instruction Set of 8086

Instruction Set of 8086

- An instruction is a binary pattern designed inside a microprocessor to perform a specific function.
- The entire group of instructions that a microprocessor supports is called **Instruction Set**.
- 8086 has more than **20,000** instructions.

Classification of Instruction Set

- Data Transfer Instructions
- Arithmetic Instructions
- Logical Instructions
- Control Transfer Instructions
- String Manipulation Instructions
- Processor Control Instructions

Instruction Format

- The size of 8086 instruction is one to six bytes depending upon the addressing modes used for instructions.
- The general Instruction format that most of the **instructions of the 8086 microprocessor** follow is:

OpCode (6 bits)	D (1 bit)	W (2 bits)	MOD (2 bits)	REG (3 bits)	R/M (3 bits)	Lower order bits of displacement	Higher order bits of displacement

Format Contd…

- The Opcode stands for Operation Code.
- Every Instruction has a unique 6-bit opcode.
- For example, the opcode for **MOV** is 100010.
  - **D** stands for direction
    If **D=0**, then the direction is from the register
    If **D=1**, then the direction is to the register
  - **W** stands for word
    If **W=0**, then only a byte is being transferred, i.e. 8 bits
    If **W=1**, them a whole word is being transferred, i.e. 16 bits

- The 2 bit **mod** field defines the method of addressing the operand specified by the r/m field.

Code for mod field	Name of the mode
00	Memory mode with no displacement
01	Memory mode with 8-bit signed displacement
10	Memory mode with 16-bit signed displacement
11	Register mode

The 3 bit **reg** filed is used to indicate the source or destination of the operand along with the **d** field

Code for reg field	Name of the register represented by the code when w=0 or 1	
	W=0	W=1
000	AL	AX
001	CL	CX
010	DL	DX
011	BL	BX
100	AH	SP
101	CH	BP
110	DH	SI
111	AH	DI

Code for r/m field	Effective address calculation when mod 00/01/10		
	Mod=00	Mod=01	Mod=10
000	[BX] + [SI]	[BX] + [SI] + d8	[BX] + [SI] + d16
001	[BX] + [DI]	[BX] + [DI] + d8	[BX] + [DI] + d16
010	[BP] + [SI]	[BP] + [SI] + d8	[BP] + [SI] + d16
100	[SI]	[SI] + d8	[SI] + d16
101	[DI]	[DI] + d8	[DI] + d16
110	d16 (direct)	[BP] + d8	[BP] + d16
111	[BX]	[BX] + d8	[BX] + d16

- The low order displacement and high order displacement are optional and the instruction format contains them only if there exists any displacement in the instruction.
- If the displacement is of 8 bits, then only the cell of low order displacement infilled and if the displacement is of 16 bits, then both the cells od low order and high order are filled, with the exact bits that the displacement number represents.

Types of Instruction formats

1. One byte instruction: Implied or register mode
2. Two-byte instruction: Register to/from memory/register with no displacement
3. Three-byte instructions: register to/from memory with 8-bit displacement
4. Four-byte instructions: register to/from memory with 16-bit displacement
5. Five-byte instructions: immediate 8-bit data to memory with 16-bit displacement
6. Six-byte instructions: immediate 16-bit data to memory with 16-bit displacement

Data Transfer Instructions

- These instructions are used to transfer data from source to destination.
- The operand can be a constant, memory location, register or I/O port address.

Data Transfer Instructions

- **MOV Des, Src:**
  - Src operand can be register, memory location or immediate operand.
  - Des can be register or memory operand.
  - Both Src and Des cannot be memory location at the same time.
  - E.g.:
    - MOV CX, 037A H
    - MOV AL, BL
    - MOV BX, [0301 H]

Data Transfer Instructions

- **PUSH Operand:**
  - It pushes the operand into top of stack.
  - E.g.: PUSH BX
- **POP Des:**
  - It pops the operand from top of stack to Des.
  - Des can be a general purpose register, segment register (except CS) or memory location.
  - E.g.: POP AX

Data Transfer Instructions

- **XCHG Des, Src:**
  - This instruction exchanges Src with Des.
  - It cannot exchange two memory locations directly.
  - E.g.: XCHG DX, AX

Data Transfer Instructions

- **IN Accumulator, Port Address:**
  - It transfers the operand from specified port to accumulator register.
  - E.g.: IN AX, 0028 H
- **OUT Port Address, Accumulator:**
  - It transfers the operand from accumulator to specified port.
  - E.g.: OUT 0028 H, AX

Data Transfer Instructions

- **LEA Register, Src:**
  - It loads a 16-bit register with the offset address of the data specified by the Src.
  - E.g.: LEA BX, [DI]
    - This instruction loads the contents of DI (offset) into the BX register.

Data Transfer Instructions

- **LDS Des, Src:**
  - This **instruction** copies a word from two memory locations into the register specified in the **instruction**.
  - It then copies a word from the next two memory locations into the DS register. It is useful for pointing to SI and DS at the start of a string before using a string **instruction**.
  - E.g.: LDS BX, [0301 H]

Data Transfer Instructions

- **LES Des, Src:**
  - This **instruction** copies a word from two memory locations into the register specified in the **instruction**.
  - It then copies a word from the next two memory locations into the ES register.
  - It is useful for pointing to DI and ES at the start of a string before using a string **instruction**.
  - E.g.: LES BX, [0301 H]

Data Transfer Instructions

- **LAHF:**
  - It copies the lower byte of flag register to AH.
- **SAHF:**
  - It copies the contents of AH to lower byte of flag register.
- **PUSHF:**
  - Pushes flag register to top of stack.
- **POPF:**
  - Pops the stack top to flag register.

Arithmetic Instructions

- **ADD Des, Src:**
  - It adds a byte to byte or a word to word.
  - It effects AF, CF, OF, PF, SF, ZF flags.
  - E.g.:
    - ADD AL, 74H
    - ADD DX, AX
    - ADD AX, [BX]

Arithmetic Instructions

- **ADC Des, Src:**
  - It adds the two operands with CF.
  - It effects AF, CF, OF, PF, SF, ZF flags.
  - E.g.:
    - ADC AL, 74H
    - ADC DX, AX
    - ADC AX, [BX]

Arithmetic Instructions

- **SUB Des, Src:**
  - It subtracts a byte from byte or a word from word.
  - It effects AF, CF, OF, PF, SF, ZF flags.
  - For subtraction, CF acts as borrow flag.
  - E.g.:
    - SUB AL, 74H
    - SUB DX, AX
    - SUB AX, [BX]

Arithmetic Instructions

- **SBB Des, Src:**
  - It subtracts the two operands and also the borrow from the result.
  - It effects AF, CF, OF, PF, SF, ZF flags.
  - E.g.:
    - SBB AL, 74H
    - SBB DX, AX
    - SBB AX, [BX]

Arithmetic Instructions

- **INC Src:**
  - It increments the byte or word by one.
  - The operand can be a register or memory location.
  - It effects AF, OF, PF, SF, ZF flags.
  - CF is not effected.
  - E.g.: INC AX

Arithmetic Instructions

- **DEC Src:**
  - It decrements the byte or word by one.
  - The operand can be a register or memory location.
  - It effects AF, OF, PF, SF, ZF flags.
  - CF is not effected.
  - E.g.: DEC AX

Arithmetic Instructions

- **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)

Arithmetic Instructions

- **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.

Arithmetic Instructions

- **NEG Src:**
  - It creates 2's complement of a given number.
  - That means, it changes the sign of a number.

Arithmetic Instructions

- **CMP Des, Src:**
  - It compares two specified bytes or words.
  - The Src and Des can be a constant, register or memory location.
  - Both operands cannot be a memory location at the same time.
  - The comparison is done simply by internally subtracting the source from destination.
  - The value of source and destination does not change, but the flags are modified to indicate the result.

Arithmetic Instructions

- **MUL Src:**
  - It is an unsigned multiplication instruction.
  - It multiplies two bytes to produce a word or two words to produce a double word.
  - AX = AL * Src
  - DX : AX = AX * Src
  - This instruction assumes one of the operand in AL or AX.
  - Src can be a register or memory location.
- **IMUL Src:**
  - It is a signed multiplication instruction.

Arithmetic Instructions

- **DIV Src:**
  - It is an unsigned division instruction.
  - It divides word by byte or double word by word.
  - The operand is stored in AX, divisor is Src and the result is stored as:
    - AH = remainder AL = quotient
- **IDIV Src:**
  - It is a signed division instruction.

Arithmetic Instructions

- **CBW (Convert Byte to Word):**
  - This instruction converts byte in AL to word in AX.
  - The conversion is done by extending the sign bit of AL throughout AH.
- **CWD (Convert Word to Double Word):**
  - This instruction converts word in AX to double word in DX : AX.
  - The conversion is done by extending the sign bit of AX throughout DX.

Logical Instructions

- These instructions are used at the bit level.
- These instructions can be used for:
  - Testing a zero bit
  - Set or reset a bit
  - Shift bits across registers

Logical Instructions

- **NOT Src:**
  - It complements each bit of Src to produce 1's complement of the specified operand.
  - The operand can be a register or memory location.

Logical Instructions

- **AND Des, Src:**
  - It performs AND operation of Des and Src.
  - Src can be immediate number, register or memory location.
  - Des can be register or memory location.
  - Both operands cannot be memory locations at the same time.
  - CF and OF become zero after the operation.
  - PF, SF and ZF are updated.

Logical Instructions

- **OR Des, Src:**
  - It performs OR operation of Des and Src.
  - Src can be immediate number, register or memory location.
  - Des can be register or memory location.
  - Both operands cannot be memory locations at the same time.
  - CF and OF become zero after the operation.
  - PF, SF and ZF are updated.

Logical Instructions

- **XOR Des, Src:**
  - It performs XOR operation of Des and Src.
  - Src can be immediate number, register or memory location.
  - Des can be register or memory location.
  - Both operands cannot be memory locations at the same time.
  - CF and OF become zero after the operation.
  - PF, SF and ZF are updated.

Logical Instructions

- **SHL Des, Count:**
  - It shift bits of byte or word left, by count.
  - It puts zero(s) in LSBs.
  - MSB is shifted into carry flag.
  - If the number of bits desired to be shifted is 1, then the immediate number 1 can be written in Count.
  - However, if the number of bits to be shifted is more than 1, then the count is put in CL register.

Logical Instructions

- **SHR Des, Count:**
  - It shift bits of byte or word right, by count.
  - It puts zero(s) in MSBs.
  - LSB is shifted into carry flag.
  - If the number of bits desired to be shifted is 1, then the immediate number 1 can be written in Count.
  - However, if the number of bits to be shifted is more than 1, then the count is put in CL register.

Logical Instructions

- **ROL Des, Count:**
  - It rotates bits of byte or word left, by count.
  - MSB is transferred to LSB and also to CF.
  - If the number of bits desired to be shifted is 1, then the immediate number 1 can be written in Count.
  - However, if the number of bits to be shifted is more than 1, then the count is put in CL register.

Logical Instructions

- **ROR Des, Count:**
  - It rotates bits of byte or word right, by count.
  - LSB is transferred to MSB and also to CF.
  - If the number of bits desired to be shifted is 1, then the immediate number 1 can be written in Count.
  - However, if the number of bits to be shifted is more than 1, then the count is put in CL register.

Control 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.

Control Transfer Instructions

- **CALL Des:**
  - 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.

Control Transfer Instructions

- **JMP Des:**
  - This instruction is used for unconditional jump from one place to another.

- **Jxx Des (Conditional Jump):**
  - All the conditional jumps follow some conditional statements or any instruction that affects the flag.

Conditional Jump Table

Mnemonic	Meaning	Jump Condition
JA	Jump if Above	CF = 0 and ZF = 0
JAE	Jump if Above or Equal	CF = 0
JB	Jump if Below	CF = 1
JBE	Jump if Below or Equal	CF = 1 or ZF = 1
JC	Jump if Carry	CF = 1
JE	Jump if Equal	ZF = 1
JNC	Jump if Not Carry	CF = 0
JNE	Jump if Not Equal	ZF = 0
JNZ	Jump if Not Zero	ZF = 0
JPE	Jump if Parity Even	PF = 1
JPO	Jump if Parity Odd	PF = 0
JZ	Jump if Zero	ZF = 1

Control Transfer Instructions

- **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 Manipulation Instructions

- 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.

String Manipulation Instructions

- **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.

String Manipulation Instructions

- **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.

String Manipulation Instructions

- **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.

Processor Control Instructions

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

Processor Control Instructions

- **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.