DAS: Decimal Adjust After Subtraction: This instruction converts the results of subtraction of two packed BCD numbers to a valid BCD number. The subtraction has to be in AL only. If the lower nibble of AL is greater than 9, this instruction will subtract 06 from lower nibble of AL. If the result of subtractions sets the carry flag or if upper nibble is greater than 9, it subtracts 60H from AL. This instruction modifier the AF, CF, PF and ZF flags. The OF is undefined after DAS instruction.
The examples are as follows:

Ex: i. AL=75 BH=46

```
SUB AL,BH    ;    AL←2F=(AL)-(BH)
             ;    AF=1
DAS          ;    AL←29 (as F>9,F-6=9)
```

ii. AL=38 CH=61

```
SUB AL, CH   ;    AL←D7 CF=1(borrow)
DAS          ;    AL←77(as D>9, D-6=7)
             ;    CF=1(borrow)
```

NEG: Negate: The negate instruction forms 2's complement of the specified destination in the instruction. For obtaining 2's complement, it subtracts the contents of destination from zero. The result is stored back in the destination operand which may be a register or a memory location. If OF is set, it indicates that the operation could not be completed successfully. This instruction affects all the condition code flags.

CBW: Convert signed Byte to Word: This instruction converts a signed byte to a signed word. In other words, it copies the sign bit of a byte to be converted to all the bits in the higher byte of the result word. The byte to be converted must be in AL. The result will be in AX. It does not affect any flag.

CWD: Convert Signed Word to double Word: This instruction copies the sign bit of AX to all the bits of DX register. This operation is to be done before signed division. It does not affect any other flag.

3. Logical Instructions:

AND	OR	NOT	XOR	TEST

These byte of instructions are used for carrying out the bit by bit shift, rotate or basic logical operations. All the conditional code flags are affected depending upon the result. Basic logical operations available with 8086 instruction set an AND, OR, NOT and XOR.

AND: Logical AND: This instruction bit by bit ANDs the source operand that may be an immediate, a register, or a memory location to the destination operand that may be a register or a memory location. The result is stored in the destination operand. At least one of the operand should be a register or a memory operand. Both the operands cannot be memory locations or immediate operand.

The examples of this instruction are as follows:

Syntax: i. AND mem/reg1, mem/reg2
[mem/reg1]←[mem/reg1]∧[mem/reg2]
Ex: AND BL, CH

ii. AND mem,data
[mem]←[mem] ∧ data
Ex: AND start,05H

iii. AND reg,data
[reg]←[reg] ∧ data

Ex: AND AL, FOH

iv. AND A,data
[A]←[A] ∧ data
A:AL/AX
Ex: AND AX,1021H

OR: Logical OR: The OR instruction carries out the OR operation in the same way as described in case of the AND operation. The limitations on source and destination operands are also the same as in case of AND operation.

Syntax: i. OR mem/reg1, mem/reg2
[mem/reg1]←[mem/reg1] ∨ [mem/reg2]
Ex: OR BL, CH

ii. OR mem,data
[mem←[mem] ∨ data
Ex: OR start, 05H

iii. OR Start,05H
[reg]←[reg] ∨ data
Ex: OR AL, FOH

iv. OR A, data
[A]←[A] ∨ data
Ex: OR AL, 1021H
A: AL/AX.

NOT: Logical Invert: The NOT instruction complements (invents) the contents of an operand register or a memory location bit by bit.

Syntax: i. NOT reg
[reg] ←[reg]'
Ex: NOT AX

ii. NOT mem
[mem]←[mem]'
Ex: NOT [SI]

XOR: Logical Exclusive OR: The XOR operation is again carried out in a similar way to the AND and OR operation. The constraints on high output, when the 2 input bits are dissimilar. Otherwise, the output is zero.

Syntax: i. XOR mem/reg1, mem/reg2
[mem/reg1]←[mem/reg1] ⊕ [mem/reg2]
Ex: XOR BL, CH

ii. XOR mem,data
[mem] ← [mem] ⊕ data
Ex: XOR start, 05H

iii. XOR reg, data
[reg]← [reg] ⊕ data
Ex: XOR AL, FOH

iv. XOR A, data
[A]← [A] ⊕ data

A: AL/AX

Ex: XOR AX, 1021H

CMP: Compare: This instruction compares the source operand, which may be a register or an immediate data or a memory location, with a destination operand that may be a register or a memory location. For comparison, it subtracts the source operand from the destination operand but does not store the result anywhere. The flags are affected depending on the result of subtraction. If both the operands are equal, zero flag is set. If the source operand is greater than the destination operand, carry flag is set or else, carry flag is reset.

Syntax: i. CMP mem/reg1, mem/reg2
[mem/reg1] – [mem/reg2]
Ex: CMP CX, BX

ii. CMP mem/reg, data
[mem/reg] – data
Ex: CMP CH, 03H

iii. CMP A, data
[A]- data
A: AL/AX
Ex: CMP AX, 1301H

TEST: Logical Compare Instruction: The TEST instruction performs a bit by bit logical AND operation on the two operands. Each bit of the result is then set to 1, if the corresponding bits of both operands are1, else the result bit is rest to 0. The result of this and operation is not available for further use, but flags are affected. The affected flags are OF, CF, ZF and PF. The operands may be register, memory or immediate data.

Syntax: i. TEST mem/reg1, mem/reg2
[mem/reg1] $\wedge$ [mem/reg2]
Ex: Test CX,BX
ii. TEST mem/reg, data
[mem/reg] $\wedge$ data
Ex: TEST CH, 03H

iii. TEST A, data
[A] $\wedge$ data
A: AL/AX
Ex: TEST AX, 1301H

4.Shift Instructions:

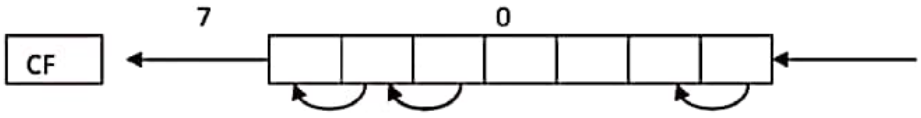
SHL/SAL	SHR	SAR

SHL/SAL: Shift Logical/ Arithmetic Left: These instructions shift the operand word or byte bit by bit to the left and insert zeros in the newly introduced least significant bits. In case of all the SHIFT and ROTATE instructions, the count is either 1 or specified by register CL. The operand may reside in a register or memory location but cannot be immediate data. All flags are affected depending on the result.

Ex:

BIT POSITIONS:	CF	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OPERAND:		1	0	1	0	1	1	0	0	1	0	1	0	0	1	0	1
SHL	1		0	1	0	1	1	0	0	1	0	1	0	0	1	0	1	0
RESULT1st																	
SHL	0	1	0	1	1	0	0	1	0	1	0	0	1	0	1	0	0
RESULT 2nd																	

Syntax: i. SAL mem/reg,1
Shift arithmetic left once



ii. SAL mem/reg, CL

Shift arithmetic left a byte or word by shift count in CL register.

iii. SHL mem/reg,1
Shift Logical Left
Ex: SHL BL, 01H

iv. SHL mem/reg, CL
Shift Logical Left once a byte or word in mem/reg.

SHR: Shift Logical Right: This instruction performs bit-wise right shifts on the operand word or byte that may reside in a register or a memory location, by the specified count in the instruction and inserts zeros in the shifted positions. The result is stored in the destination operand. This instruction shifts the operand through carry flag.

Ex:

BIT POSITIONS:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	CF
OPERAND	: 1	0	1	0	1	1	0	0	1	0	1	0	0	1	0	1	
Count=1		0	1	0	1	0	1	1	0	0	1	0	1	0	0	1	0	1
Count=2	0	0	1	0	1	0	1	1	0	0	1	0	1	0	0	1	0	

SAR: Shift Arithmetic Right: This instruction performs right shifts on the operand word or byte, that may be a register or a memory location by the specified count in the instruction and inserts the most significant bit of the operand the newly inserted positions. The result is stored in the destination operand. All the condition code flags are affected. This shift operation shifts the operand through carry flag.

Ex:

BIT POSITIONS:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	CF
OPERAND:		1	0	1	0	1	1	0	0	1	0	1	0	0	1	0	1
Count=1		1	1	0	1	0	1	1	0	0	1	0	1	0	0	1	0	1
inserted MSB=1																	
Count=2	1	1	1	0	1	0	1	1	0	0	1	0	1	0	0	1	0	
	inserted MSB=1																

Immediate operand is not allowed in any of the shift instructions.

Syntax: i. SAR mem/reg,1
ii. SAR mem/reg, CL