INSTRUCTION SET OF 8086 µP (Module- II)

By:-
Prof. Kamakshya Prasad Nayak
Department of CSE,
GCEK, Bhawanipatna

Types of Instruction Sets

- Data Copy/Transfer Instructions
- Arithmetic & Logical Instructions
- Branch Instructions
- Loop Instructions
- Machine Control Instructions
- Flag Manipulation Instructions
- Shift & Rotate Instructions
- String Instructions

Data Copy/Transfer Instructions

- MOV
- PUSH
- POP
- XCHG
- IN
- OUT
- XLAT
- LEA
- LDS/LES
- LAHF
- SAHF
- PUSHF
- POPF

Data Copy/Transfer Instructions

- **MOV:** Transfers data from one register/memory location to another register/memory location.

- **Restriction:- Direct loading of segment registers with immediate data is not permitted.**

MOV Instruction contd…

If we want to load DS with 5000H then:

- MOV DS, 5000 H (Not permitted)
- MOV AX,5000H

  MOV DS,AX

These 2 instructions used together is the correct procedure to load DS with 5000H.

MOV Instruction contd…

- Format of MOV instruction with other addressing modes is as follows:

- MOV AX,5000H (Immediate)
- MOV AX,BX (Register)
- MOV AX, [SI] (Indirect)
- MOV AX, [2000H] (Direct)
- MOV AX, 50H[BX] (Based relative, 50H displacement)

Data Copy/Transfer Instructions

- **PUSH:** It pushes the contents of the specified register/memory location on to the stack.
- The SP(Stack Pointer) is decremented by 2, after each execution of instruction.
- The addressing in the stack segment is from higher address to lower address i.e. as the stack grows, the address value keeps on decreasing.
- The higher byte is first pushed on to the stack and then the lower byte.

PUSH Instruction contd…

- The sequence of operation for the instruction **PUSH AX** is as follows:-
- Current stack top is already occupied so decrement SP(Stack Pointer) by one and store AH into the address pointed to by SP.
- Further, decrement SP by one and store AL into the location pointed to by SP.
- **Examples of instructions are:-**

➢ PUSH AX

➢ PUSH DS

➢ PUSH [5000H] (Content of 5000H and 5001H in DS are pushed on to the stack)

Data Copy/Transfer Instructions

- POP: Loads the specified register/memory location from the address formed by the current stack segment and SP.
- The SP is incremented by 2.
- The sequence of operations is as follows:-

- Contents of stack top memory location are stored in AL and SP is incremented by one.
- Content of new memory location pointed to by SP is copied to AH and SP is again incremented by one.

POP Instruction contd…

- **Examples of instructions are:-**

- POP AX
- POP DS
- POP [5000H]

- Effectively SP is incremented by 2 and points to the next stack top.

PUSH & POP contd…

- Let AX = 5522H
- => AH=55H and AL=22H
- **When PUSH AX is executed**, AH=55H is first pushed on to the stack and then AL=22H is pushed.
- Similarly, **when POP AX is encountered**, first 22H(AL) is deleted from stack and then 55H(AH).

Data Copy/Transfer Instructions

- XCHG: (Exchange) This instruction exchanges the contents of the source and destination operands which may be registers or memory locations.
- **Restrictions:-**

- Exchange between 2 memory locations is not permitted.

- Immediate data is also not allowed in these instructions.

XCHG instruction contd…

- **Examples of instructions are:-**
- XCHG [5000H], AX (Exchange between memory location [5000H] and register AX)
- XCHG BX, AX (Exchanges data between BX and AX)

Data Copy/Transfer Instructions

- IN : (Input the Port) Used for reading input port
- Address of the input port may be specified directly or indirectly.
- 8-bit or 16-bit port address is allowed.
- DX is the only register which is used to carry port address.
- AL and AX can be 8-bit and 16-bit destination for this instruction.

IN instruction contd…

- **Examples of instructions are:-**
- IN AL, 03H (Reads data from 8-bit port whose address is 03H and stores it in AL)
- IN AX, DX (Reads data from 16-bit port whose implicit address is in DX and stores it in AX)
- MOV DX, 0800H (16-bit address is taken in DX)

  IN AX,DX (Read the content of the port in AX)

Data Copy/Transfer Instructions

- OUT: (Output to the Port) Used for writing to an output port.
- Address of output port may be specified in instruction directly or implicitly in DX.
- Contents of AX or AL are transferred to a directly or indirectly addressed port.
- Data to odd addressed port – transferred on lines D8-D15 and to even addressed port on D0-D7.
- AL and AX are allowed source operands
- 16-bit port address must be in DX.

OUT instruction contd…

- **Examples of instructions are:-**

- OUT 03H,AL (Sends data available in AL to a port whose address is AH)
- OUT DX, AX (Sends data available in AX to a port whose address is specified implicitly in DX)
- MOV DX, 0300H (16-bit port address taken in DX)
- OUT DX, AX (Write the content of AX to a port whose address is in DX)

Data Copy/Transfer Instructions

- XLAT: (Translate) Used for finding codes in case of code conversion problems using look up table technique.
- Example:- In case of a hexadecimal keypad interfaced with 8086, the code of the pressed key is returned in AL. For displaying the number corresponding to the pressed key on the 7-segment display device, it is required that the 7-segment code corresponding to the pressed key is found out and sent to the display port. This translation is done by XLAT instruction.

XLAT Instruction contd…

- The code for the example given in the previous slide is given as:-

- MOV AX, SEG TABLE (Address of the segment containing look up table)
- MOV DS, AX (Address is transferred to DS)
- MOV AL, CODE (Code of the pressed key is transferred to AL)
- MOV BX, OFFSET TABLE (Offset of the code look-up table in BX)
- XLAT (Find the equivalent code and store in AL)

Data Copy/Transfer Instructions

- LEA: Load Effective Address

- Loads the effective address formed by the destination operand into the specified source register.

- Example:-

➢ LEA BX, ADR (Offset of label ADR will be transferred to BX)

➢ LEA SI, ADR[BX] (Offset of label ADR will be added to content of BX to form effective address and is loaded in SI)

Data Copy/Transfer Instructions

- LDS/LES : Load pointer to DS or ES

- Loads the DS or ES register and the destination register with the content of the memory location specified as source in the instruction.

- Example:-

LDS BX, 5000H (Content of memory location 5000H is stored in BX as well as DS)

LES BX, 5000H (Content of memory location 5000H is stored in BX as well as ES)

Data Copy/Transfer Instructions

- LAHF: Load AH from lower byte of Flag Register
- SAHF: Store AH to lower byte of Flag Register.

- If a bit in AH=1, corresponding position in the Flag Register is set, otherwise it is reset.

- PUSHF: Push the Flag Register value to the stack. First the upper byte is pushed and then the lower byte)
- POPF: Pop the Flag Register from the stack

* **PUSHF and POPF work like the PUSH and POP operations.**

Arithmetic Instructions

Arithmetic Instructions

- ADD, ADC
- INC
- DEC
- SUB, SBB
- CMP
- AAA, AAS,AAM, AAD
- DAA, DAS
- NEG
- MUL, IMUL
- CBW
- CWD
- DIV, IDIV

Arithmetic Instructions

- ADD: Adds immediate data or contents of memory location or register to another memory location or register.
- All flags are affected during its execution.
- **Restrictions:-**

- **Both source and destination operands cannot be memory locations.**
- **2 segment registers cannot be added.**

ADD Instruction contd…

- Examples:-

- ADD AX, 0100H (Immediate)
- ADD AX,BX (Register)
- ADD AX, [SI] (Register Indirect)
- ADD AX, [5000H] (Direct)
- ADD [5000H], 0100H (Immediate)
- ADD 0100H (Destination AX, implicit)

Arithmetic Instructions

- ADC: Add with Carry
- Same as ADD but adds the carry to the result.
- All flags are affected.
- Examples:-
- ADC 0100H (Immediate, AX implicit)
- ADC AX, BX (Register)
- ADC AX, [SI] (Register Indirect)
- ADC AX, [5000H] (Direct)
- ADC [5000H, 0100H (Immediate)

Arithmetic Instructions

- INC: Increment
- Increases the content of specified register or memory location by 1.
- All condition code flags affected except Carry Flag (CF)
- Immediate data cannot be operand of this instruction.
- Examples:-

INC AX (Register)

INC [BX] (Register Indirect)

INC [5000H] (Direct)

Arithmetic Instructions

- DEC: Decrement

- Subtracts 1 from the contents of specified register or memory location.
- All condition code flags affected except Carry Flag (CF)
- Immediate data cannot be operand of this instruction.
- Examples:-
- DEC AX (Register)
- DEC [5000H] (Direct)

Arithmetic Instructions

- SUB: Subtract

- Subtraction between source and destination operand
- Source may be register or memory location or immediate data and destination may be register or memory location.
- Condition code flags affected by this instruction.
- **Restrictions:-**

Both source and destination cannot be memory operands.

Destination cannot be immediate data.

SUB Instruction contd…

- Examples:-

- SUB AX, 0100H [Immediate, destination is AX]
- SUB AX, BX [Register]
- SUB AX , [5000H] [Direct]
- SUB [5000H], 0100 [Immediate]

Arithmetic Instructions

- SBB : Subtract with Borrow

- Subtracts the source and carry flag together from the destination.
- Carry (Borrow) Flag should be set for this instruction.
- Condition Code flags are affected.
- Examples:-

➢ SUB AX, 0100H [Immediate, destination is AX]

➢ SUB AX, BX [Register]

➢ SUB AX , [5000H] [Direct]

➢ SUB [5000H], 0100 [Immediate]

Arithmetic Instructions

- CMP: Compare

- Compares the source operand(register or immediate data or memory location) with the destination (register or memory location)
- Flags are affected.
- Source = Destination (Zero Flag=1)
- Source > Destination (Carry Flag = 1)
- Source < Destination (Carry Flag = 0)

CMP Instruction contd…

- Examples:-

- CMP BX, 0100H [Immediate]
- CMP AX, 0100H [Immediate]
- CMP [5000H], 0100H [Direct]
- CMP BX, [SI] [Register Indirect]
- CMP BX,CX [Register]

Arithmetic Instructions

- NEG: Negate

- Forms the 2's complement of the specified destination
- Subtracts the contents of the destination from zero and the result is stored in the destination(register or memory location)
- Overflow Flag =1 implies the operation could not be completed successfully.
- Affects all condition code flags.

Arithmetic Instructions

- MUL: Unsigned Multiplication Byte or Word

- Multiplies the unsigned byte or word (in general purpose registers or memory location) by the contents of AL.
- Most significant word of the result is stored in DX and the least significant one in AX.
- All flags affected.
- **Restrictions:-**

  Immediate operand is not allowed.

  Most significant byte or word of the result is 0 implies both CF and OF are set.

MUL Instruction contd…

- Examples:-

- MUL BH [(AX) <- (AL) x (BH)]
- MUL CX [(DX) (AX) <- (AX) x (CX)]
- MUL WORD PTR [SI] [(DX) (AX) <- (AX) x ([SI])]

*RHS in Red represents meaning of the instructions.

Arithmetic Instructions

- IMUL: Signed Multiplication

- **Multiplies signed byte in AL with signed byte in source operand or signed word in AX with signed word in source operand**
- Source can be general purpose register, memory operand, index register or base register.
- In case of 32bit results, higher order word is stored in DX and lower order word in AX.
- AF, PF, SF, ZF undefined in this case.
- CF and OF are set when AH and DX contain parts of the result.
- AL and AX are implicit operands.

IMUL Instruction contd…

- Examples:-

- IMUL BH
- IMUL CX
- IMUL [SI]
- Restrictions:-

  Immediate data cannot be a source in this case.

Arithmetic Instructions

- CBW: Convert Signed Byte to Word
- Converts the signed byte to signed word
- Does not affect any word.
- CWD: Convert Word to Double Word
- Operation is done before signed division
- Does not affect any flag.

Arithmetic Instructions

- DIV: Unsigned Division

- **Divides and unsigned word or double word by a 16-bit or 8-bit operand.**
- Dividend in AX for 16-bit operation and divisor may be specified by any of the addressing modes except immediate.
- Quotient will be in AL, remainder in AH.
- In case of double word, higher word is in DX and lower word in AX, divisor mentioned as in previous case, quotient will be in AX and remainder in DX.
- Does not affect any flag
- Interrupts generated if result is too big to fit in the destination. (Divide by zero interrupt)

Arithmetic Instructions

- IDIV : Signed Division

- **Same operation as DIV but performs on signed operands.**
- Results are also stored as in DIV case and results are signed numbers.
- Flags are undefined after IDIV instruction.
- Interrupts are generated if the result is too big to fit in the designated registers. (Divide by zero interrupt)

Logical Instructions

- AND (Logical AND)
- OR (Logical OR)
- NOT (Logical Invert)
- XOR (Logical Exclusive OR)
- TEST (Logical Compare Instruction)
- SHL/SAL (Shift Logical/ Arithmetic Left)
- SHR (Shift Logical Right)
- SAR (Shift Arithmetic Right)
- ROR (Rotate Right without carry)
- ROL (Rotate Left without carry)
- RCR (Rotate Right through carry)
- RCL (Rotate Left through carry)

Logical Instructions

- AND : Logical AND

- Bit by bit logical AND between source and destination
- Examples:-
- AND AX, 0008H
- AND AX,BX
- AND AX, [5000H]
- AND [5000H], DX

Logical Instructions

- OR : Logical OR

- Bit by bit logical OR between source and destination
- Examples:-
- OR AX, 0098H
- OR AX,BX
- OR AX, [5000H]
- OR [5000H], 0008H

Logical Instructions

- NOT : Logical Invert

- Complements the contents of the register or memory location, bit by bit.
- Examples:-
- NOT AX
- NOT [5000H]

Logical Instructions

- XOR : Logical Exclusive OR

- Examples:-
- XOR AX, 0098H
- XOR AX, BX
- XOR AX, [5000H]

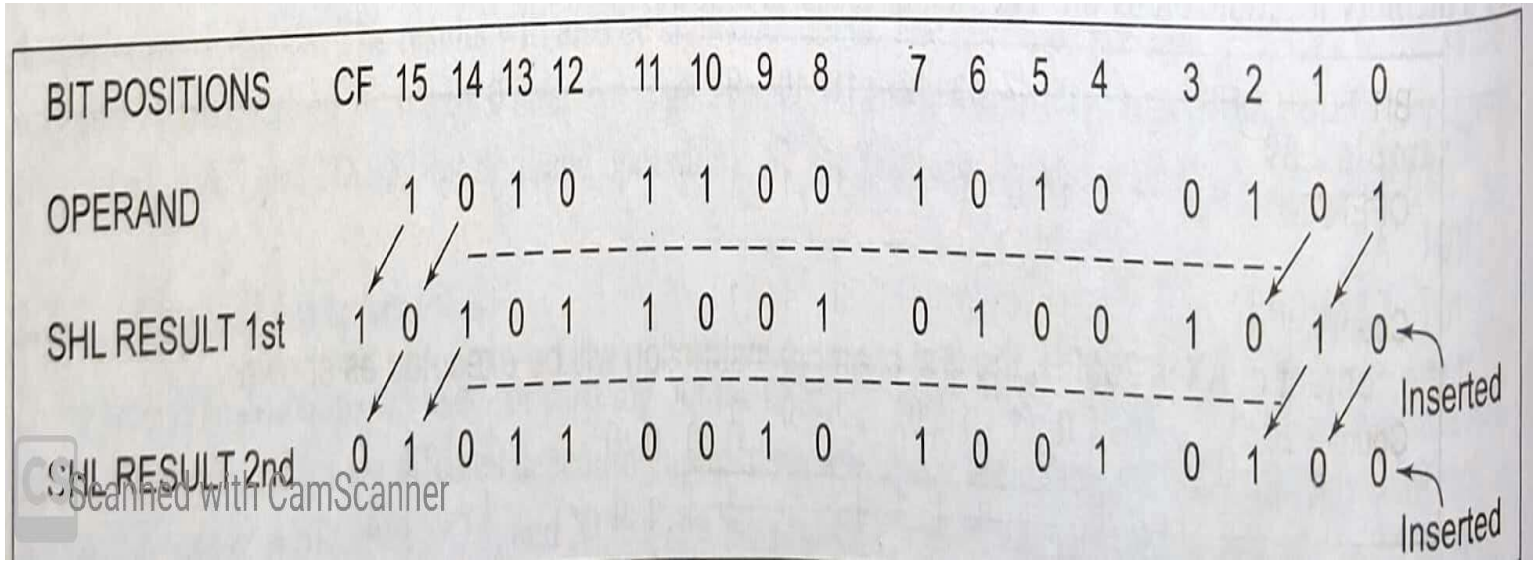
Logical Instructions

- TEST : Logical Compare Instruction

- Performs bit by bit logical AND between 2 operands but result is not available for use.
- Flags like OF,CF, SF, ZF and PF are affected.
- Example:-
- TEST AX,BX
- TEST [0500H]
- TEST [BX] [DI], CX

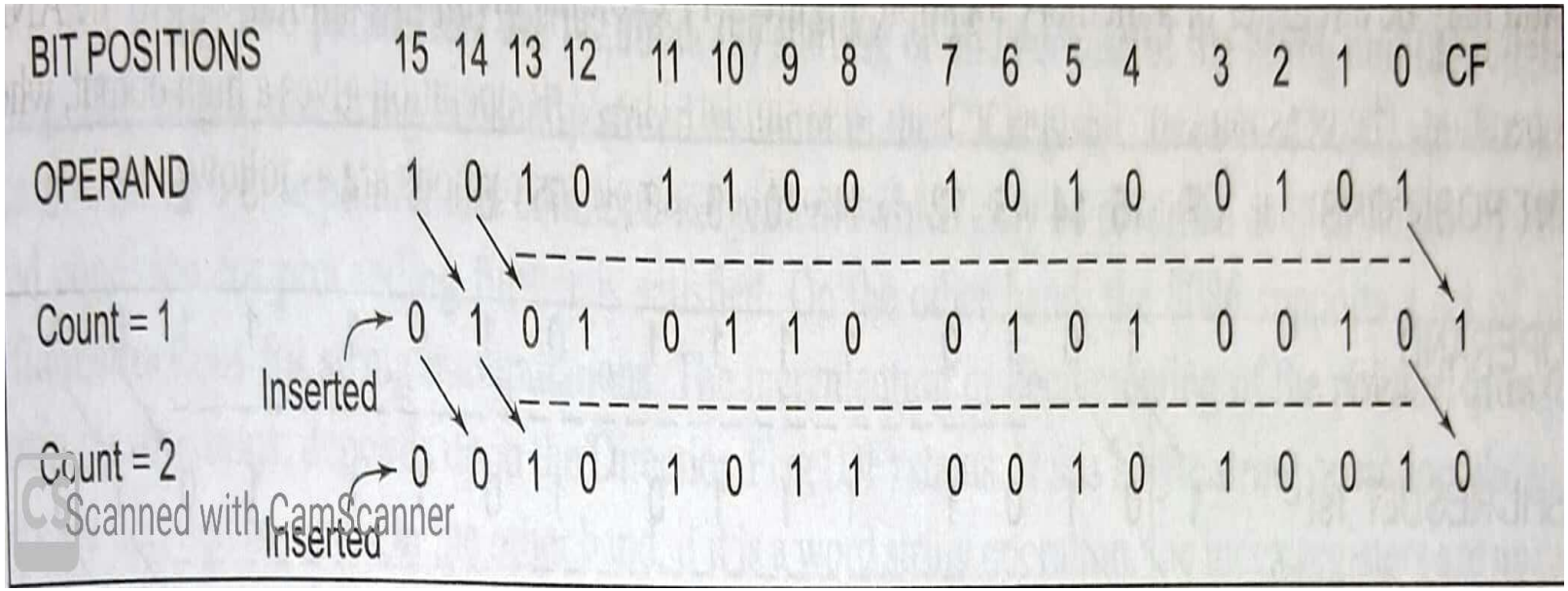
Logical Instructions

- SHL/SAL : Shift Logical/Arithmetic Left



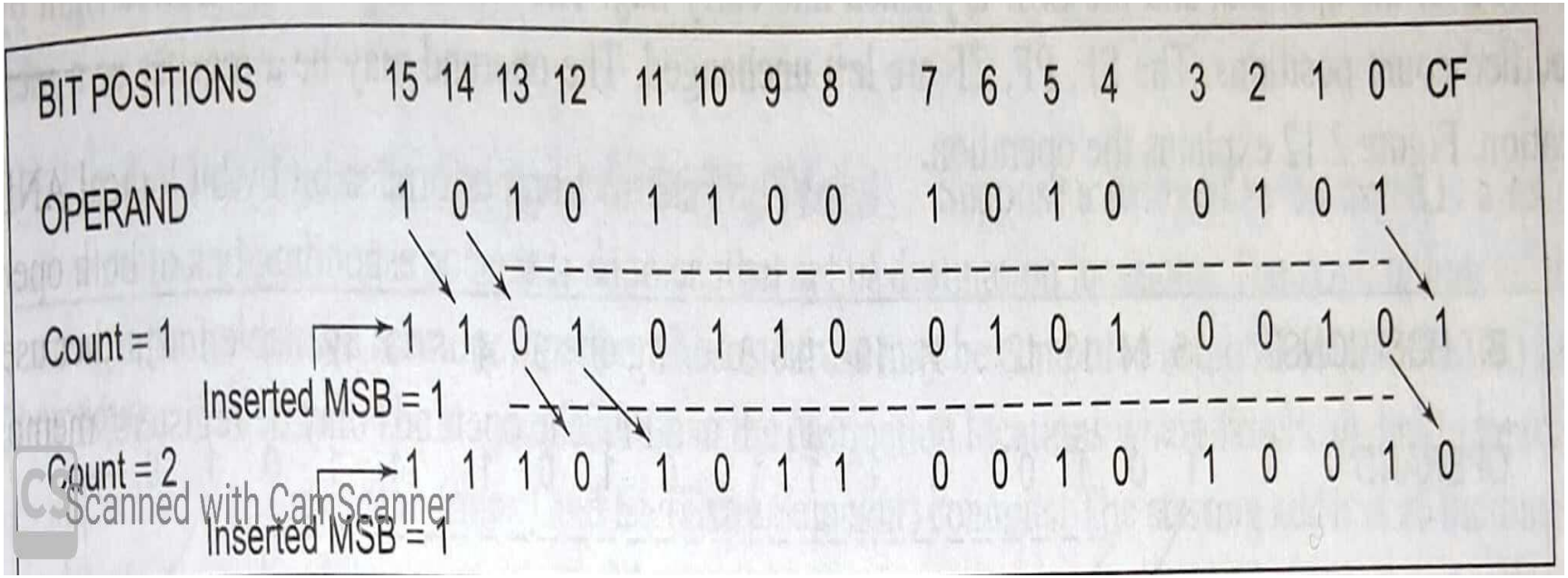
Logical Instructions

- SHR : Shift Logical Right



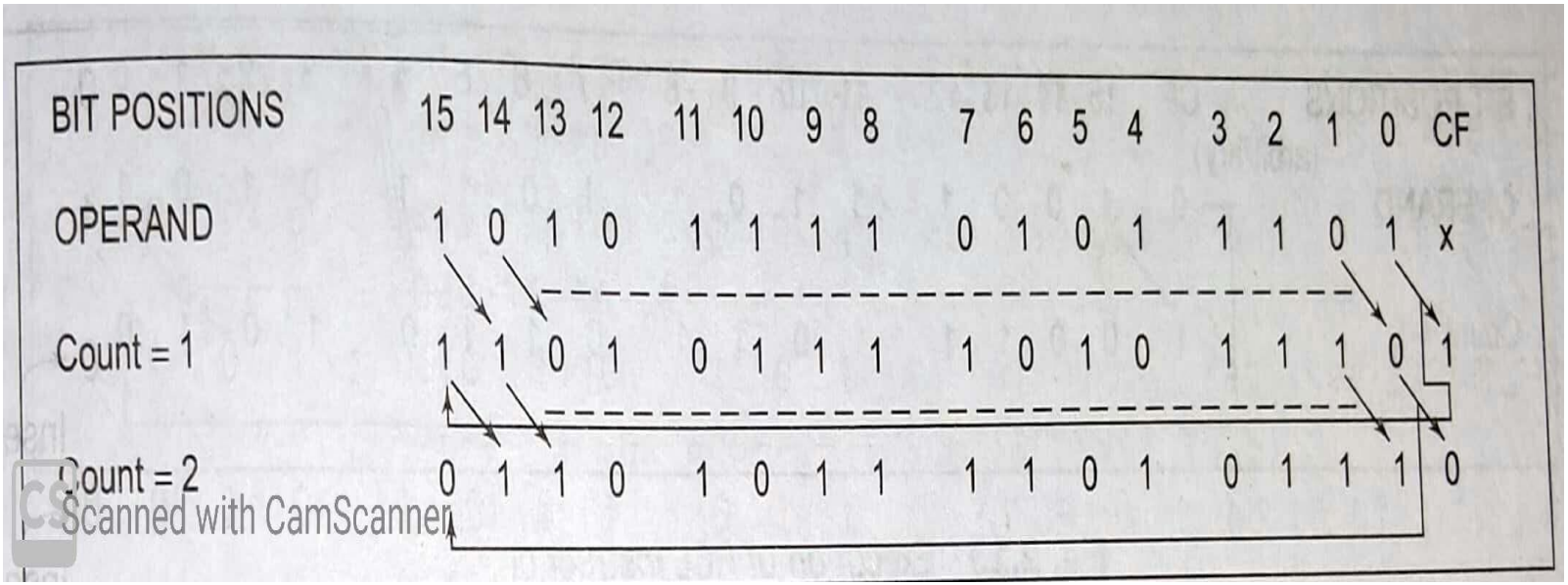
Logical Instructions

- SAR : Shift Arithmetic Right



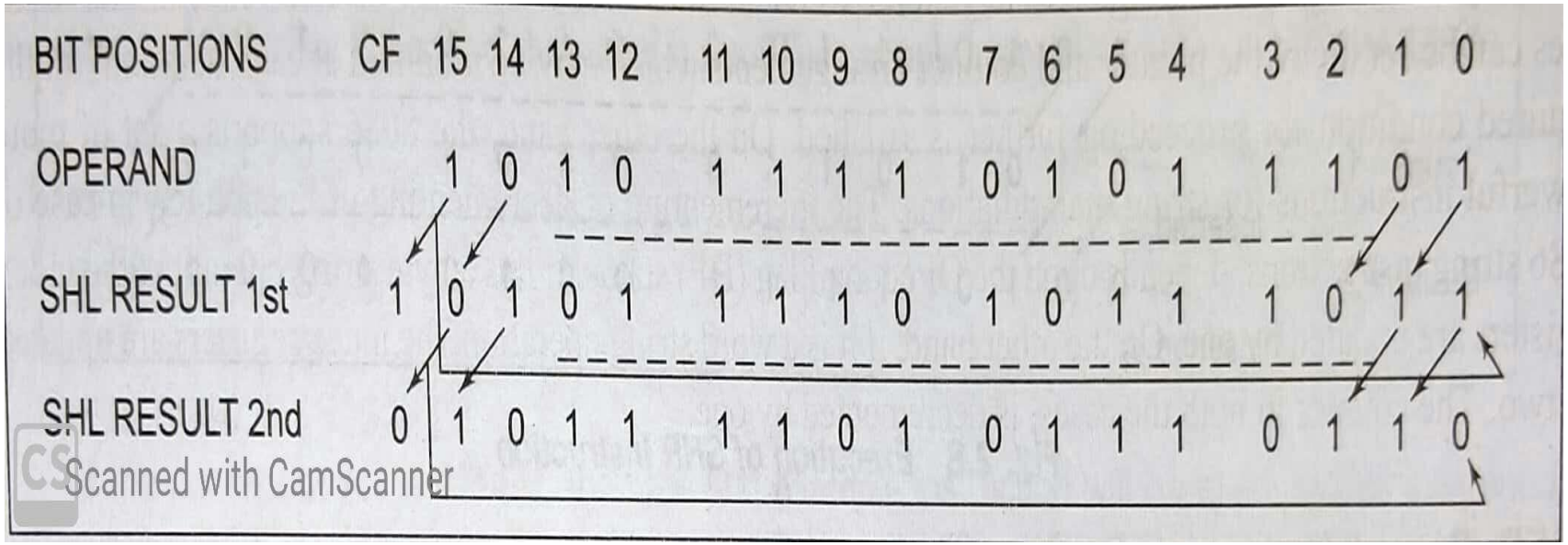
Logical Instructions

- ROR : Rotate Right without carry



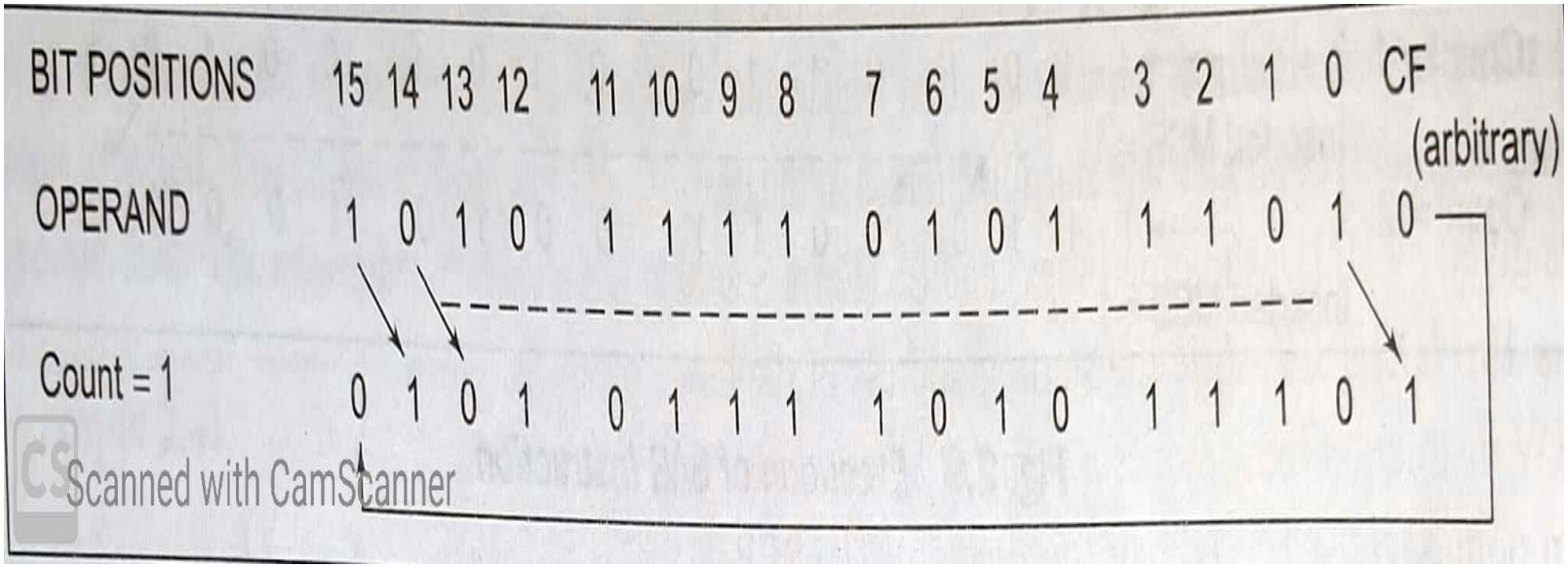
Logical Instructions

- ROL : Rotate left without carry



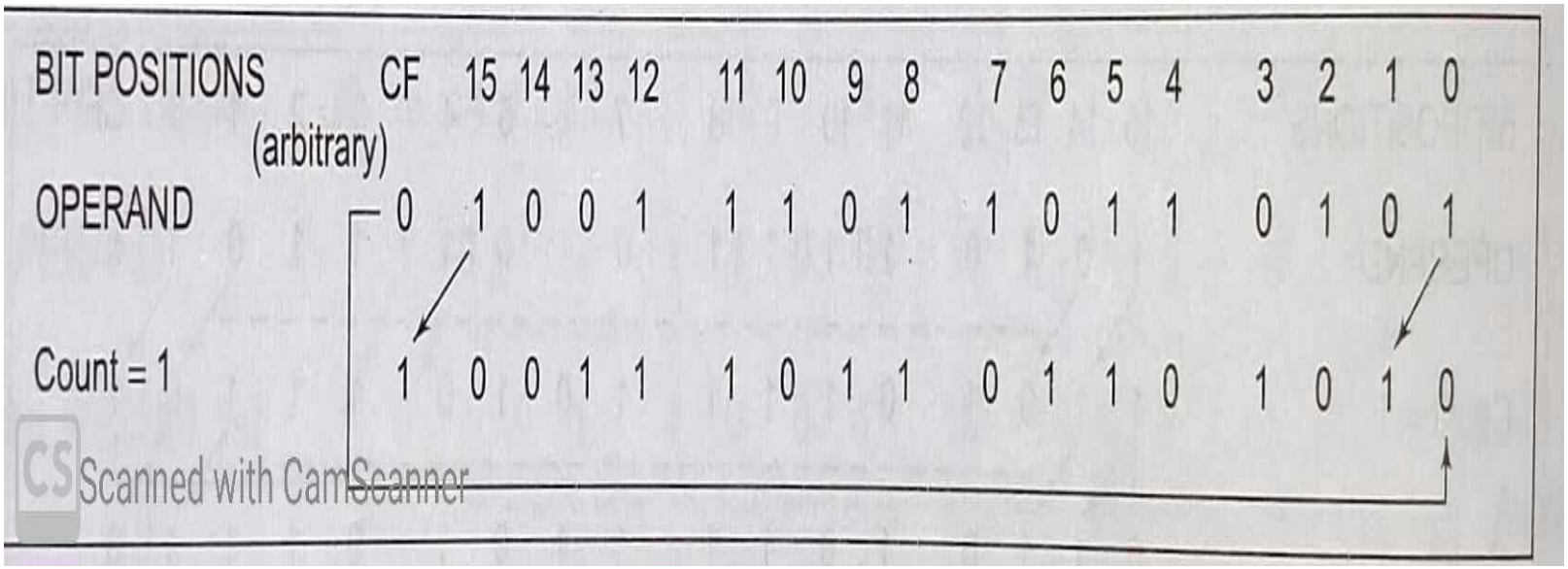
Logical Instructions

- RCR : Rotate Right through carry



Logical Instructions

- RCL : Rotate Left through carry



String Manipulation Instructions

- Series of data bytes or words in consecutive memory locations are called a byte strings or word strings.
- For referring to a string, two things are required:-

  - Starting/end address of string

  - Length of the string (stored as count in CX register)
- DF (Direction Flag) is used while reading strings.
- Index registers updated by 1 for byte string instruction and by 2 for word string operation.

String Manipulation Instructions

- REP
- MOVSB/MOVSW
- CMPS
- SCAS
- LODS
- STOS

String Manipulation Instructions

- REP (Repeat Instruction Prefix)

- Used as prefix to other instructions
- Instruction after REP repeats until CX is zero.
- Variants of REP :-

  - REPE/REPZ (Repeat while equal or zero)

  - REPNE/REPNZ (Repeat while not equal/not zero)

String Manipulation Instructions

- MOVSB/ MOVSW : Move string byte or word

- **Example:-**

```
MOV AX, 5000H [Source segment address is 5000H]
MOV DS,AX [Load it to DS]
MOV AX,6000H [Dest. segment address is 6000H]
MOV ES,AX [Load it to ES]
MOV CX, 0FFH [Move string length to CX]
MOV SI, 1000H [Source index 1000H to SI]
MOV DI, 2000H [Dest. Index 2000H to DI]
CLD [Clear DF, set suto increment mode]
REP MOVSB [Move 0FFH string bytes from src. to dest.]
```

String Manipulation Instructions

- CMPS : Compare string byte or word
- Compares two bytes or words
- If both are equal, ZF is set.
- **Example:-**

```
MOV AX,SEG1 [Seg add. of string1 to AX]
MOV DS,AX [Load it to DS]
MOV AX, SEG2 [Seg. add. Of string2 to AX]
MOV ES,AX [Load it to ES]
MOV SI, OFFSET   STRING1
MOV DI, OFFSET STRING2
MOV CX, 010H [Length of string to CX]
CLD
REPE CMPSW [compare 010H words of string1 and string2, if equal
                then CX=0 and ZF is set, else ZF is reset]
```

String Manipulation Instructions

- SCAS : Scan string byte or word
- Scans for byte or word in AL/AX.
- String is pointed by ES/DI pair.
- DF controls mode of scanning.
- **Example:-**

MOV AX,SEG [seg add of string to AX]
MOV ES,AX [Load it to ES]
MOV DI,OFFSET [string offset is moved to DI]
MOV CX,010H [Length of string to CX]
MOV AX, WORD [WORD to be scanned is in AX]
CLD
REPNE SCASW [scan 010H bytes of string, till a match to WORD is found]

String Manipulation Instructions

- LODS : Load string byte or string word
- Loads the AL/AX by the content pointed by DS:SI pair.
- SI is modified as per DF.
- Variants of this instruction are LODSB/LODSW for byte and word respectively.
- STOS : Store string byte or word
- Stores AL/AX contents to the location pointed by ES:DI pair.

Control Transfer / Branch Instructions

- **UNCONDITIONAL CONTROL TRANSFER:-**

- CALL (Unconditional Call to a subroutine)
- RET (Return from procedure)
- INT N (Interrupt type N)
- INTO (Interrupt on Overflow)
- JMP (Unconditional Jump)
- IRET (Return from ISR)
- LOOP (Loop Unconditionally)

Control Transfer / Branch Instructions

- **CONDITIONAL CONTROL TRANSFER:-**

- JZ/JE (Jump if ZF=1)
- JNZ/JNE (if ZF=0)
- JS (if SF=1)
- JNS (if SF=0)
- JO (of OF=1)
- JNO (if OF=0)
- JP/JPE (if PF=1)
- JNP (if PF=0)

Control Transfer / Branch Instructions

- **CONDITIONAL CONTROL TRANSFER:-**

- JB/JNAE/JC (if CF=1)
- JNB/JAE/JNC (if CF=0)
- JBE/JNA (if CF=1 or ZF=1)
- JNBE/JA (if CF=0 or ZF=0)
- JL/JNGE (if neither SF=1 nor OF=1)
- JNL/JGE (if neither SF=0 nor OF=0)
- JLE/JNC (if ZF=1 or neither SF nor OF is 1)
- JNLE/JE (if ZF=0 or at least one of SF and OF is 1)

Control Transfer / Branch Instructions

- LOOPZ/LOOPE (Loop over a sequence of instructions while ZF=1)
- LOOPNZ/LOOPNE (Loop over a sequence of instructions while ZF=0)

Flag Manipulation and Processor Control Instructions

- **Flag Manipulation Instructions:-**

- CLC (clear carry flag)
- CMC (complement carry flag)
- STC (set carry flag)
- CLD (clear direction flag)
- STD (set direction flag)
- CLI (clear interrupt flag)
- STI (set interrupt flag)

Flag Manipulation and Processor Control Instructions

- **Machine Control Instructions**

- WAIT (Wait for Test Input pin to go low)
- HLT (Halt the processor)
- NOP (No operation)
- ESC (Escape external devices like co-processor)
- LOCK (Bus lock instruction)

Assembler Directives and Operators

- DB (Define Byte)
- DW (Define Word)
- DQ (Define Quadword)
- DT (Define Ten Bytes)
- ASSUME (Assume logical segment name)
- END (End of Program)
- ENDP (End of Procedure)
- ENDS (End of Segment)

Assembler Directives and Operators

- EVEN (Align on even memory address)
- EQU (equate label with value or symbol)
- EXTRN (var/proc defined in some other proc)
- GROUP (group the related segments)
- LABEL (name the current content of the location counter)
- LENGTH (byte length of a label)
- LOCAL (var/const/proc used in the module itself)
- NAME (logical name of a module)

Assembler Directives and Operators

- OFFSET (Offset of a label)
- ORG (Origin of memory allotment)
- PROC (start of a named procedure)
- PTR (declare type of label/var/memory operand)
- PUBLIC (used along with EXTRN)
- SEG (Segment address of a label)
- SEGMENT (Logical segment marking during use)
- SHORT (one byte of displacement)

Assembler Directives and Operators

- TYPE (decide the data type)
- GLOBAL (variable/constant/procedure accessed by other procedures)
- '+' &' –' (For adding/subtracting displacements)
- FAR PTR (Label is not in the same segment)
- NEAR PTR (Label is in the same segment)

References

- Bhurchandi K.M. and Ray A.K., “Advanced Microprocessor and Peripherals”, 3rd edition, McGraw Hill