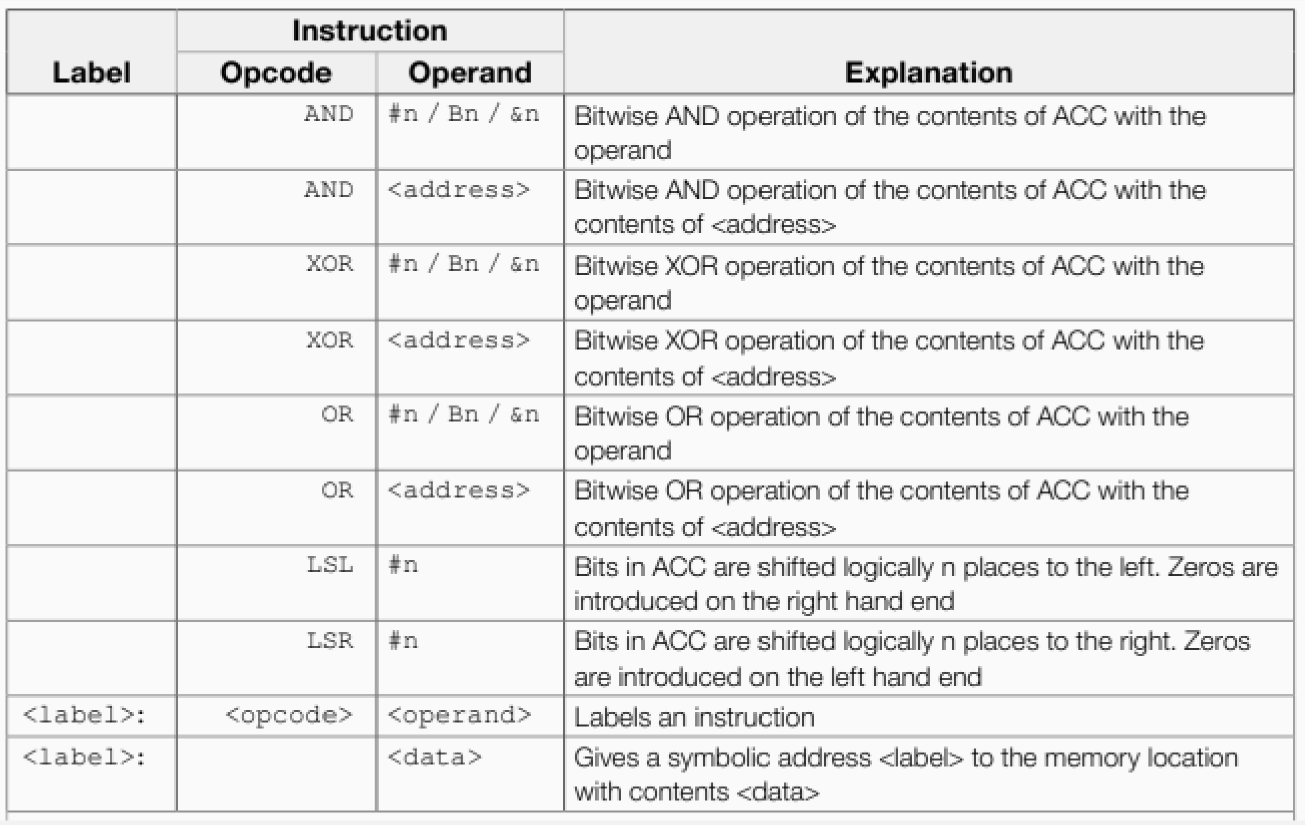
4.3 Bit Manipulation

Objective:

- Show understanding of & perform binary shifts Logical, arithmetic & cyclic Left shift, right shift.



Label	Instruction		Explanation
	Opcode	Operand	
	AND	#n / Bn / &n	Bitwise AND operation of the contents of ACC with the operand
	AND	<address>	Bitwise AND operation of the contents of ACC with the contents of <address>
	XOR	#n / Bn / &n	Bitwise XOR operation of the contents of ACC with the operand
	XOR	<address>	Bitwise XOR operation of the contents of ACC with the contents of <address>
	OR	#n / Bn / &n	Bitwise OR operation of the contents of ACC with the operand
	OR	<address>	Bitwise OR operation of the contents of ACC with the contents of <address>
	LSL	#n	Bits in ACC are shifted logically n places to the left. Zeros are introduced on the right hand end
	LSR	#n	Bits in ACC are shifted logically n places to the right. Zeros are introduced on the left hand end
<label>:	<opcode>	<operand>	Labels an instruction
<label>:		<data>	Gives a symbolic address <label> to the memory location with contents <data>

Note: ACC denotes Accumulator, IX denotes Index Register, # denotes a denary number, e.g. #123, B denotes a binary number, e.g. B01001010 & denotes a hexadecimal number, e.g. &4

Bit Manipulation to Control Devices

Bit Masking allows **checking**, **setting** and **resetting** individual bits within a binary value. Bitwise operations are **fast** and **simple** operations on binary data where each binary digit is treated **individually**.

Advantage of using bit masking is to speed up processing and require reduced or less processing to perform the task (as require just few binary tricks).

Application of Bit Masking:

- Useful in simple **control systems** where simple systems use individual bits as **flags**. Substantial processing efficiency can be gained.
- Widely used in networking when using **subnet masks**.

Mask: A number that is used with the logical operators AND, OR, XOR & NOT to identify, remove or set a single bit or group of bits in an address or register.

Bitwise Operations include :

- **Check Bits** (find the value of or a bit)
- **Set Bits** (Set the bit/bits to 1)
- **Clear Bits** (Set the bit/bits to 0)
- **Toggle Bits** (Set a bit/bits to the binary opposite)
- **Bitwise AND :** Used to check if the bit has been **set**?
- **Bitwise OR :** Used to set the bit to 1.
- **Bitwise XOR :** Use to **clear** a bit that has been set.

What is Bit Masking?

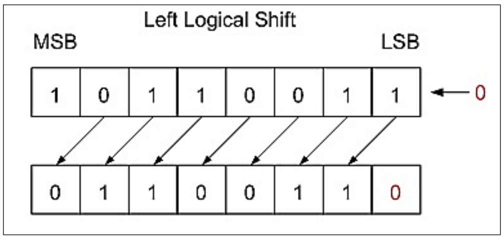
Bit masking is process of getting processer to ignore all bits that we don't want to work on and only process digits we want to process.

Binary Shifts

Binary shift involves **moving bits** stored in a register a **given number** of places within register. **Each bit** within register may be used for a different purpose.

Types of Shift.

- **Logical Shift** – bits shifted out of register are replaced with **zeros**.
  - **Left Logical Shift** of one position moves each bit to left by one. Vacant least significant bit (LSB) is filled with **zero** and most significant bit (MSB) is **discarded**.



  - **Right Logical Shift** of one position moves each bit to the **right** by one. Least significant bit is **discarded** and vacant MSB is filled with **zero.**

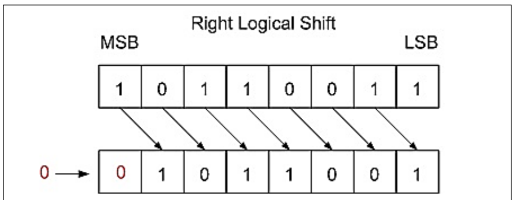
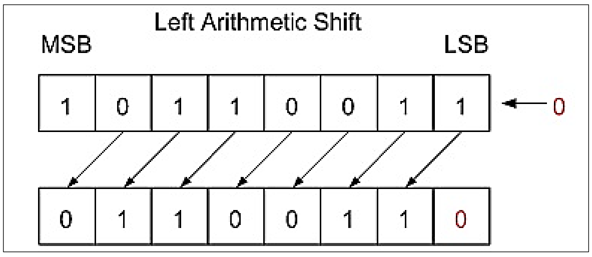
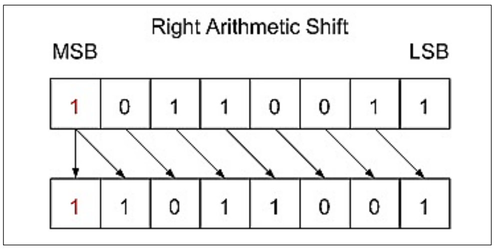


Fig. 1 Logical Shift by one bit

- **Arithmetic Shift** – Sign of number is **preserved**.
  - **Left Arithmetic Shift** of **one** position moves each bit to left by **one**. Vacant least significant bit (LSB) is filled with **zero** and most significant bit (MSB) is **discarded.** It is **identical** to Left Logical Shift.



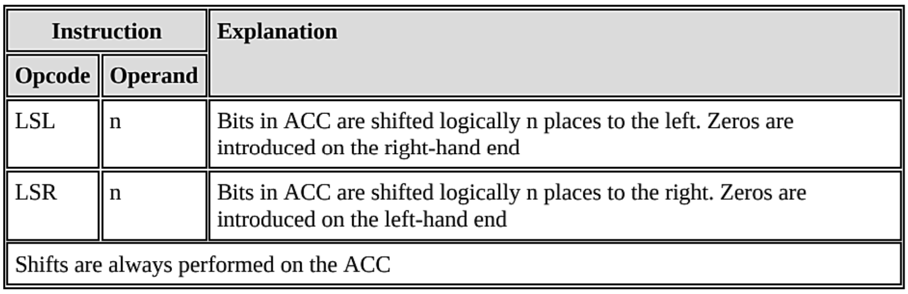
- **Right Arithmetic Shift** of one position moves each bit to right by one. Least significant bit is **discarded** and vacant MSB is filled with value of previous (now shifted one position to right) MSB.



- **Cyclic Shift** – no bits are **lost** during a shift. Bits shifted out of one end of register are introduced at other end of register.

  Example, an 8-bit register containing binary value **101**01111 shifted left cyclically three places would become 01111**101**.
- **Left Shift** – bits are shifted to left; gives direction of shift for logical, arithmetic and cyclic shifts.
- **Right Shift** – bits are shifted to right; gives direction of shift for logical, arithmetic and cyclic shifts.

Logical Shifts in Assembly Language Programming



Instruction		Explanation
Opcode	**Operand**	
LSL	n	Bits in ACC are shifted logically n places to the left. Zeros are introduced on the right-hand end
LSR	n	Bits in ACC are shifted logically n places to the right. Zeros are introduced on the left-hand end
Shifts are always performed on the ACC		

Bit Manipulation used in Monitoring and Control

In monitoring and control, **each bit** in a register can be used as a **flag** and would need to be tested, **set** or **cleared** separately.

Example, control system with eight different sensors would need to record when data from each sensor had been processed.

- **AND** is used to check if bit has been **set**.
- **OR** is used to **set** the bit.
- **XOR** is used to **clear** a bit that has been set.

Setting of All Bits To Zero:

LDD 0034	Loads byte into accumulator from an address 0034.
AND #B00000000	Uses bitwise AND operation of contents of ACC With operand to convert each bit to 0.
STO 0034	Stores altered byte in original address.

Toggling of value for one bit

LDD 0034	Loads byte into accumulator from an address.
XOR #B00000001	Bitwise **XOR operation** of contents of ACC with operand to toggle value of bit stored in **position 0**.
STO 0034	Stores altered byte in the original address.

Setting of a bit to have value 1

LDD 0034	Loads byte into accumulator from an address.
OR #B00000100	Uses bitwise OR operation of contents of ACC with operand to **set flag** represented by bit in position 2. All other bit positions remain unchanged
STO 0034	Stores the altered byte in the original address

Setting all bits to zero except one bit which is of interest

LDD 0034	Loads byte into accumulator from an address.
AND #B00000010	Uses bitwise AND operation of contents of ACC with operand to leave value in **position 1** unchanged but to convert every other bit to 0
STO 0034	Stores the altered byte in the original address.

☺☺☺

Exam Style Question

ESQ: (i) Current contents of the ACC are:

1	0	0	1	0	0	1	1

Show result after execution of following instruction.

XOR B 0 0 0 1 1 1 1 1

(ii) Current contents of ACC are:

1	0	0	1	0	0	1	1

Show the result after the execution of the following instruction.

AND B 1 1 1 1 0 0 0 0

****