Instruction set

- An extensive set of instruction is are provided to carry out various computational task.
- According to the operation carried out by the computers, the instructions are classified into 3 categories.

1. Data Transfer Instruction
2. Data Manipulation "
   a. Arithmetic Instruction
   b. Logical Instruction
   c. Shift Instruction.
3. Program control Instruction.

Data Transfer Instructions : When data is moved from the source to the destination, source copy remains intact.

Name	Mnemonics	
Load	LD	
Store	ST	
Move	MOV	
Exchange	XCH	DE ⇆ HL (8085, 8086)
Input	IN	
output	OUT	
Push	PUSH	
Pop	POP	

IN : This instruction is used to take input from input device. (8085, 8086)

In #portno, reg.
IN #08H, R1

This instruction takes an input into processor register R1, from the device whose identification no is 08H. H indicates hexadecimal no.

OUT : This instruction is used to send output from the processor register to an output device.

OUT reg, #portno
OUT R1, #09H

This instruction sends an output from processor register R1, to an output device whose identification no is 09H.

Data Manipulation Instructions

Arithmetic

Name	Mnemonics
Increment	INC
Decrement	DEC
Add	ADD
Subtract	SUB
Multiply	MUL
Divide	DIV
Add with carry	ADDC
Subtract with borrow	SUBB.

Logical & Bit Manipulation Instructions

Name Mnemonics

1. AND AND
It is used to reset some specific bit position in a register keeping all other bits intact (unchanged)

Ex. AND #F7H, R1

2. OR OR
It is used to set some specific bit position in a registers, keeping all other bits intact (unchanged)

Example OR #04, R1

3. XOR XOR
(Exclusive OR)
It is used to clear the contents of a register

XOR R1, R1

4. Set carry - SETC
It is used to set the carry flag
CF ← 1

5. Clear Carry — CLRC
It is used to Reset the carry flag
CF ← 0

6. Complement carry CMC
It is used to complement the carry Flag.
$CF \leftarrow \overline{CF}$

7. Enable Interrupt EI
It is used to set the Interrupt Flag.
IF = 1, Interrupt activated.
IF = 0, Interrupt deactivated.

INTR is an Interrupt request signal to the Processor.
When IF = 1, then only processor recognizes an interrupt request and responds to it, else ignores the request coming on line INTR.

(Explain INTR PIN and ISR)

8. DI → Disable Interrupt.
It is used to reset the interrupt flag.
IF ← 0
When IF = 0, then the Processor ignores the Interrupt request. DI instruction is used to mask (disable) the INTR signal.

9. Clear CLR

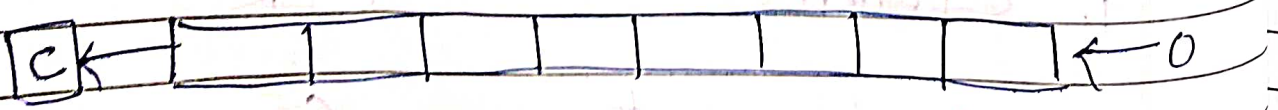
10. complement COM.

Shift Instructions

- Shift n Rotate Instructions:

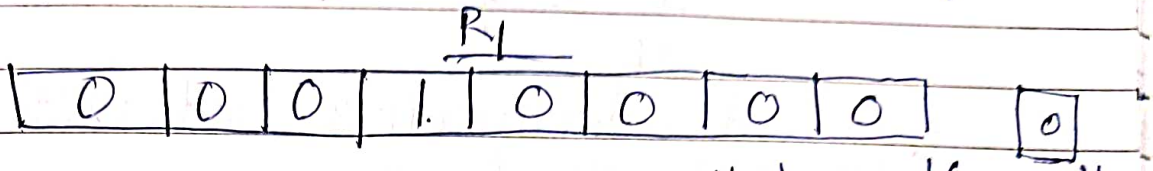
Name	Mnemonics
Logical shift right	SHR
Logecal shift Left	SHL
Arithmetic shift right	SHRA
Arithmetic shift Left	SHLA
Rotate right	ROR
Rotate Left	ROL
Rotate right through carry	RORC
Rotate left through carry	ROLC

- SHL: Logical left shift for unsigned numbers.
Provide a means for shifting blocks of bits within a register or memory.



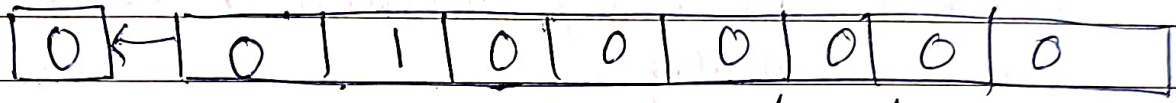
The contents of the operand are shifted left by the number of

bits specified in the source operand of the instruction. The vacated bits are filled with zeros. The shifted bits are passed through the C flag, and then dropped. Left shifting an operand is equivalent to multiplying the operand by $2^{(\text{bit position shifted})}$



value = 16 = 2^4

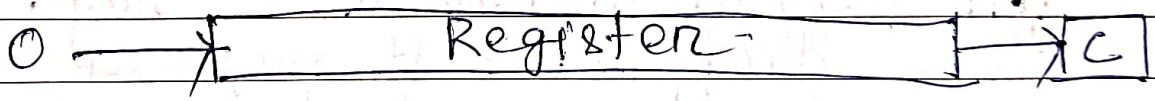
SHL #2, R1



R1 = 2^6 = 64 = 16 × 2^2
= 16 × 4

- SHR: Logical right shift for unsigned numbers.

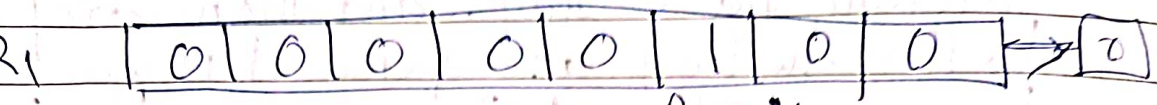
Provide a means for shifting blocks of bits within a register or memory.



The contents of the operand are shifted right by the number of bits specified in the source operand of the instruction. The vacated bits are filled with zeros. The shifted bits are passed through the C flag, and then dropped. Right shifting an operand is equivalent to dividing the operand by $2^{(\text{bit Posin shifted})}$

$R_1 = 16$

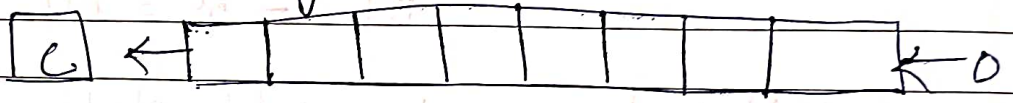
SHR #2, R1



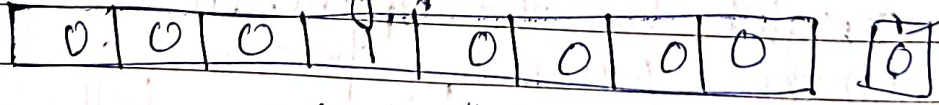
now $R_1 = 4$

$\frac{16}{4} = 4$

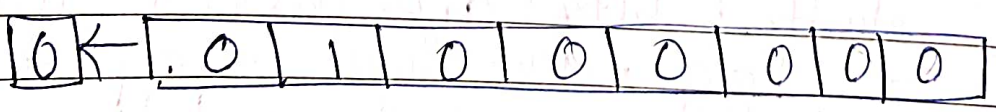
- SHLA : Arithmetic Left shift for signed numbers.
Provide a means for shifting blocks of bits within a register or memory.



The contents of the operand are shifted left by the number of bits specified in the source operand of the Instruction. The vacated bits are filled with zeros. The shifted bits are passed through the C flag, and then dropped. Left shifting an operand is equivalent to multiplying the operand by $2^{(\text{bit positions shifted})}$



SHLA #2, R1



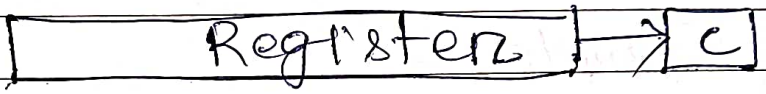
SHLA affect the overflow flag

The overflow flag will be 1, if the sign changes after the shift operation else will be zero.

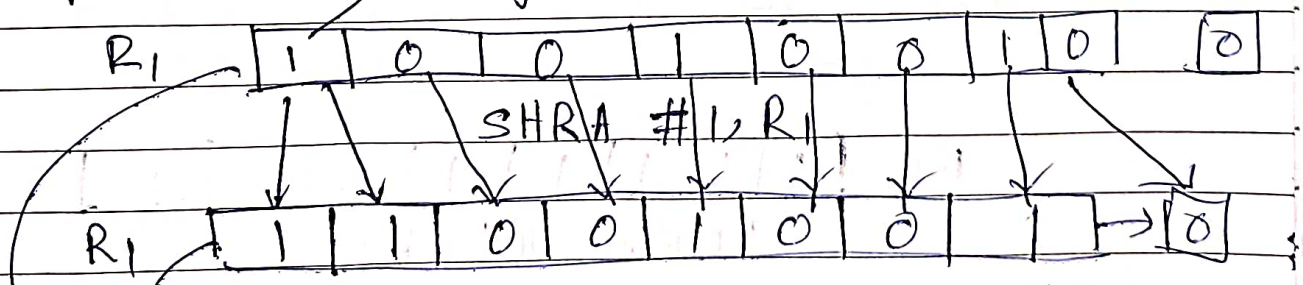
Arithmetic Right shift operation.

SHRA: Arithmetic right shift for signed numbers.

Provide a means for shifting block of bits within a register or memory.



The contents of the operand are shifted right by the number of bits specified in the source operand of the instruction. The vacated bits are filled by the previous sign bit. The shifted bits are passed through the C-flag and then dropped. Right shifting an operand is equivalent to dividing the operand by 2 (bit position shifted) -ve



2's complement = 1 1 1 0 1 1 0

$= -110$

2's complement = 1 0 1 1 0 1 1

$= -55$

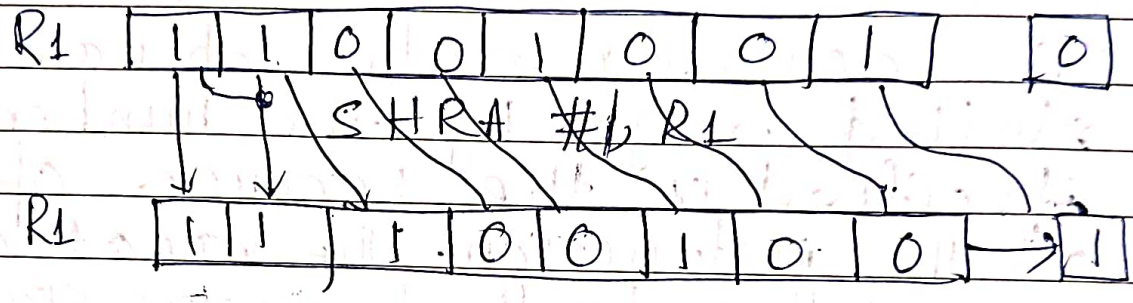
$-110 / 2^1 = -55$

$2^1 \longrightarrow$ 1 bit position is shifted.

Numerical

Arithmetic Right shift instruction

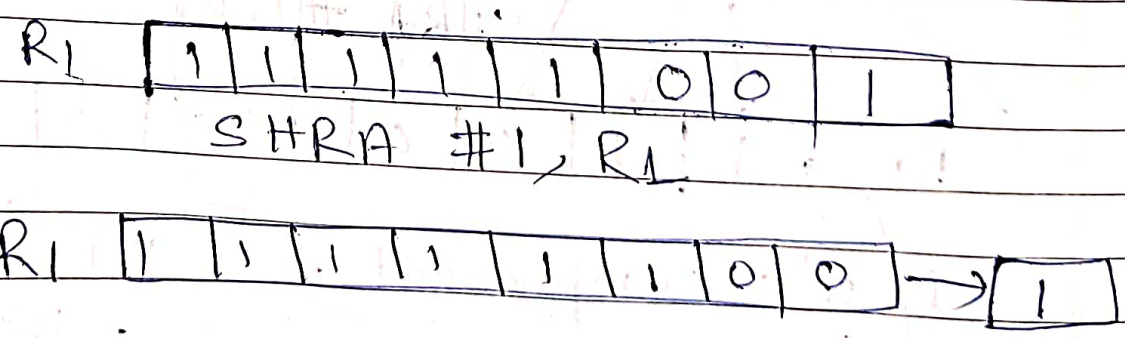
Example -1



Here, R_1 containing -55 before the shift operation, and after the SHRA, it containing - 28

$-55 / 2^1 = -27.5 = -28$

Example - 2



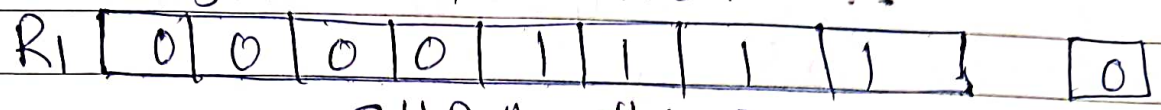
Here R_1 contains -7 before the shift operation, and after the

SHRA, it containg -4

$\frac{-7}{2} = -3.5 = -4$

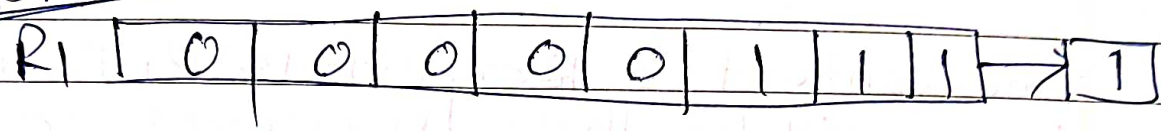
Example-3.

Before execution.



SHRA #1, R1

After



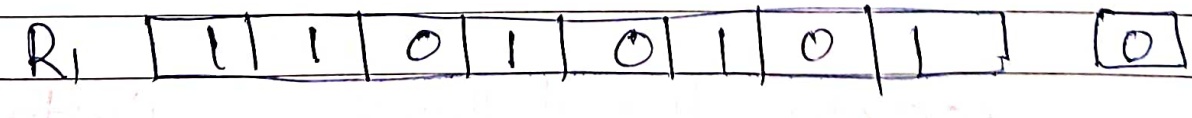
Here, R_1 containg +15 before the shift operation, and after the SHRA, it containg +7.

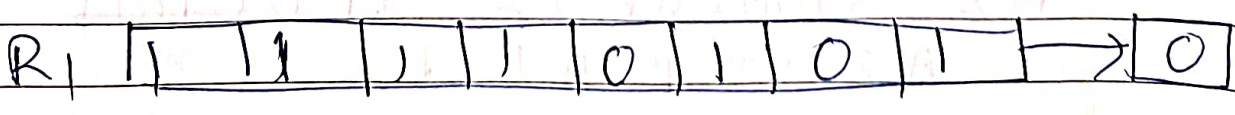
$\frac{15}{2} = 7.5 = 7$

Example-4

The content of Register R_1 is 11010101. what will be the decimal value after execution of AshiftR #2, R_1. [Assume the number is represented in 2's complement format]

Ans





Here, R_1 containg -43 before the shift operation, and After the AshiftR 2 times it containg -11.

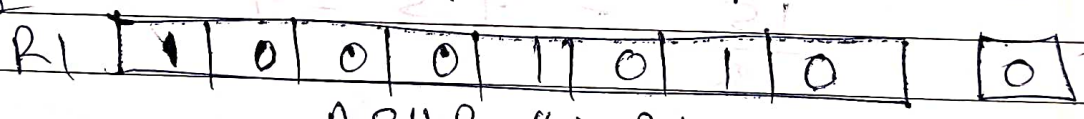
$$\frac{-43}{2^2} = -(10.7) = -11$$

Example-5

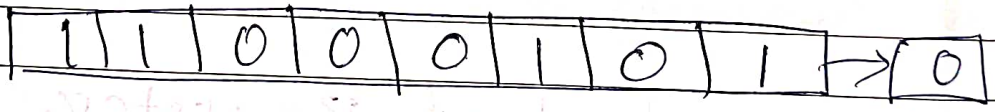
The content of Register R_1 is 1000010. What will be the decimal value in R_1 after the execution of ASHR #1, R_1

(Assume the number are written in 2's complement form)

Ans



ASHR #1, R_1



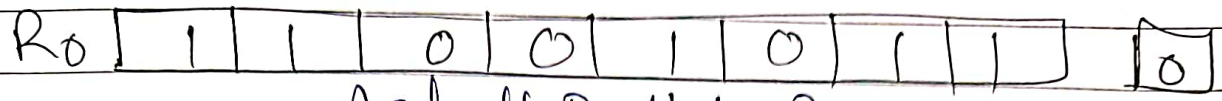
Here R_1 containg -118 before the shift operation, and after the AshiftR 1 time it containg

$$\frac{-118}{2^1} = -59$$

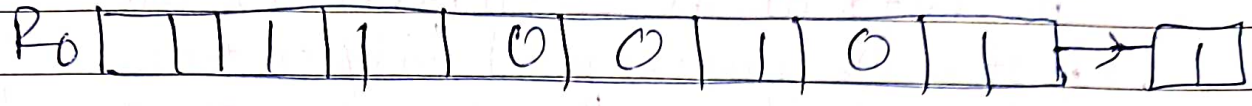
Example-6

Execute the following instruction where R_0 is 8 bits and its content is 11001011

(i) AshiftR #1, R_0



Ashift R #1, R0



Here, R0 contains -53 before the shift operation, and After the Ashift R 1 time, it contains

$$\frac{-53}{2^1} = -26.5 = -27$$

Numerical 5

The content of register R_1 is 1011001100110011. write the instruction for performing the following operations.

(i) Clear the LSB of R_1 to 0.

(ii) Set the MSB of R_1 to 1.

Note LSB = Least significant byte.
MSB = Most significant byte.

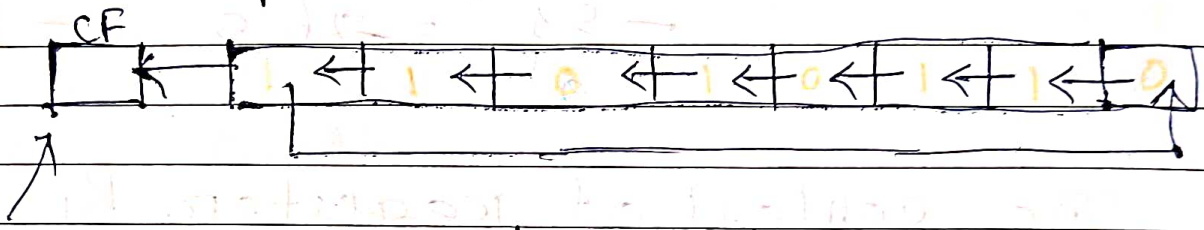
Ans (i) AND # FF00H, R1

(ii) OR # FF00H, R1

Rotate Instructions.

Rotate Left Instruction (Rotate L) ROL

The bits of the destination are rotated left. The number of bits rotated is determined by the source operand. The bits rotated out of the most significant bit of the operand go to both the carry bit and the least significant bit of the operand.



Qs. The content of register R1 is 11010110. What will be the decimal value after execution of RotateL #2, R1. [Assume the number is represented in 2's complement format]

Ans- So after the RotateL #2, R1, the content of R1 will become.

R1 = 10101101, (After the 1st rotation)

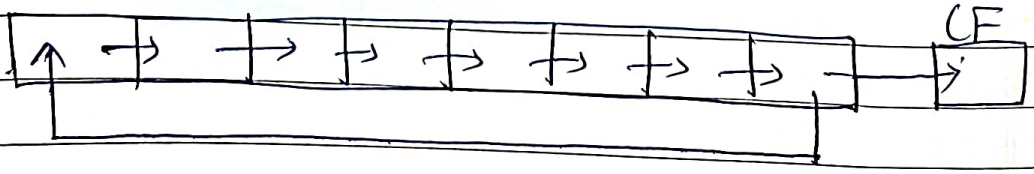
R1 = 01011011. (After the 2nd rotation)

Here, MSB is 0, Hence Result is +ve

So the [R1] in decimal = $(01011011)_2$

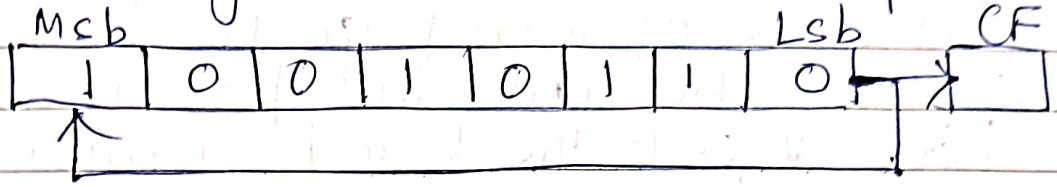
= 91

Rotate Right Instruction (RotateR)



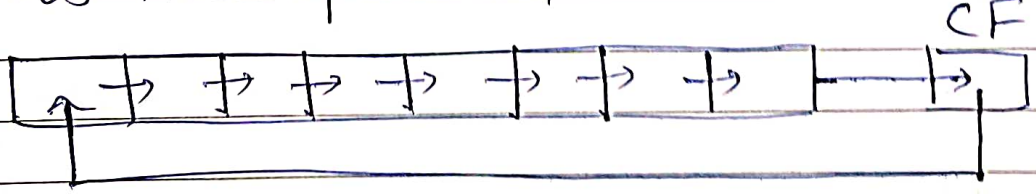
The bits of the destination are rotated right. The number of bits rotated is determined by the source operand. The bits rotated out of the least significant bit of the operand go to both carry bit and the most significant bit of the operand.

Ex



Rotate Right through carry Instruction (RotateRC) RORC

The bits of the destination are rotated right. The number of bits rotated is determined by the source operand. The bits rotated out of the least significant bit of the operand go to the carry bit and the previous Carry bit goes to the most significant bit of the operand.



Rotate Left through Carry Instruction (RotateLC) RCLC

The bits of the destination are rotated left. The number of bits rotated is determined by the source operand. The bits rotated out of the most significant bit of the operand go to both the carry bit and the previous carry bit goes to the least significant bit of the operand.