Devang Patel Institute of Advance Technology and Research (A Constitute Institute of CHARUSAT)
Certificate
This is to certify that
Mr./Mrs. Achyut Gopulbhai buldha
of
ID. No. 23DC.3005 has satisfactorily completed
his/her term work in mpco - cs E 202 for
the ending in _No\ _20 24/20 25
2
GUR GO
Date: 15 10 24 Sign. of Faculty Head of Department

A.Y.: 2024-25

CSE202: Microprocessor and Computer Organization

Charotar University of Science and Technology [CHARUSAT] Devang Patel Institute of Advance Technology and Research [DEPSTAR] Department of Computer Science & Engineering

Practical List

	Subject code	:	CSE202	Semester		3	Academic Year	:	2024-25		
Γ	Subject name	:	Microprocessor and Computer Organization								

Sr. No.		Aim Implement a circuit in Logisim to display a given binary number in decimal on a											
1.	seven segmen	display				2	1						
2.	number.				nd Subtraction of sign	2	1						
3.			mon bus system to inte fultiplexer and ii. Deco			2	2						
4.	Implement a ci	and Logic units.	4	2									
5.	Implement a common bus system with ALU, 8 registers and 1 memory unit with necessary control signals.												
6.	(Basics of assembly level programming) Perform following operations on 8-bit data												
	Addition and Logical left shift Rotate left with carry												
	Subtraction or Logical right shift Rotate left without carry												
	Multiplication	xor	Arithmetic left shift	Rotate righ	nt with carry								
	Division	not	Arithmetic right shift	Rotate righ	nt without carry								
7.		y. Perfor	embly level programmi rm addition of all even		om array and save	2	3						
8.		er the gi			d print the appropriate	2	3						
9.	(Procedure in Assembly Level Language) Write an assembly code to evaluate the answer of below given series and store the answer in the ANS variable. Program should have only one procedure to compute the factorial of a number. Series: 1! -2+3!-4+5!-6+7!-8												
10.	Write a program that performs multiplication using the booth algorithm.												
11.	Write a program	n to con	vert a given number sy	ystem to and	ther number system.	4	1						
12.	Write an assen	ıbly leve	el code for a given C p	rogram.		2	3						
Prep	ared By:	aurang I	Patel		Date: 24/06/24								

Date: 1/7/2024

EXPERIMENT NO. 1

AIM: Implement a circuit in logisim to display given decimal number in binary on to seven segment display.

TASK: i. display 0 to 9 on a seven segment display with binary number as input ii. display 0 to 99 using two seven segment displays with binary number as input

TABLES OF CALCULATIONS:

1) 0 to 9:

A	В	C	D	a	b	с	d	е	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1
1	0	1	0	1	1	0	1	1	1	1
1	0	1	1	1	1	1	1	0	1	1
1	1	0	0	1	1	1	1	0	1	1
1	1	0	1	1	0	1	1	0	1	1
1	1	1	0	1	0	1	1	1	1	1
1	1	1	1	1	1	1	1	0	1	1

2) 0 to 99

A	В	С	D	E	F	G	a	b	С	d	е	f	g	h
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0	0	1
0	0	0	0	0	1	0	0	0	0	0	0	0	1	1
0	0	0	0	0	1	1	0	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	0	0	0	0	1	0	0
0	0	0	0	1	0	1	0	0	0	0	0	1	0	1
0	0	0	0	1	1	0	0	0	0	0	0	1	1	0
0	0	0	0	1	1	1	0	0	0	0	0	1	1	1
0	0	0	1	0	0	0	0	0	0	0	1	0	0	0
0	0	0	1	0	0	1	0	0	0	0	1	0	0	1
0	0	0	1	0	1	0	0	0	0	1	0	0	0	0
0	0	0	1	0	1	1	0	0	0	1	0	0	0	1
0	0	0	1	1	0	0	0	0	0	1	0	0	1	1
0	0	0	1	1	0	1	0	0	0	1	0	0	1	1
0	0	0	1	1	1	0	0	0	0	1	0	1	0	0
0	0	0	1	1	1	1	0	0	0	1	0	1	0	1
0	0	1	0	0	0	0	0	0	0	1	0	1	1	0
0	0	1	0	0	0	1	0	0	0	1	0	1	1	1
0	0	1	0	0	1	0	0	0	0	1	1	0	0	0
0	0	1	0	0	1	1	0	0	0	1	1	0	0	1
0	0	1	0	1	0	0	0	0	1	0	0	0	0	0
0	0	1	0	1	0	1	0	0	1	0	0	0	0	1
0	0	1	0	1	1	0	0	0	1	0	0	0	1	1
0	0	1	0	1	1	1	0	0	1	0	0	0	1	1
0	0	1	1	0	0	0	0	0	1	0	0	1	0	0
0	0	1	1	0	0	1	0	0	1	0	0	1	0	1
0	0	1	1	0	1	0	0	0	1	0	0	1	1	0
0	0	1	1	0	1	1	0	0	1	0	0	1	1	1
0	0	1	1	1	0	0	0	0	1	0	1	0	0	0
0	0	1	1	1	0	1	0	0	1	0	1	0	0	1
0	0	1	1	1	1	0	0	0	1	1	0	0	0	0
0	0	1	1	1	1	1	0	0	1	1	0	0	0	1

CSF-I	DEPST	ΔR
COL-I		ΔII

0	1	0	0	0	0	0	0	0	1	1	0	0	1	1
0	1	0	0	0	0	1	0	0	1	1	0	0	1	1
0	1	0	0	0	1	0	0	0	1	1	0	1	0	0
0	1	0	0	0	1	1	0	0	1	1	0	1	0	1
0	1	0	0	1	0	0	0	0	1	1	0	1	1	0
0	1	0	0	1	0	1	0	0	1	1	0	1	1	1
0	1	0	0	1	1	0	0	0	1	1	1	0	0	0
0	1	0	0	1	1	1	0	0	1	1	1	0	0	1
0	1	0	1	0	0	0	0	1	0	0	0	0	0	0
0	1	0	1	0	0	1	0	1	0	0	0	0	0	1
0	1	0	1	0	1	0	0	1	0	0	0	0	1	1
0	1	0	1	0	1	1	0	1	0	0	0	0	1	1
0	1	0	1	1	0	0	0	1	0	0	0	1	0	0
0	1	0	1	1	0	1	0	1	0	0	0	1	0	1
0	1	0	1	1	1	0	0	1	0	0	0	1	1	0
0	1	0	1	1	1	1	0	1	0	0	0	1	1	1
0	1	1	0	0	0	0	0	1	0	0	1	0	0	0
0	1	1	0	0	0	1	0	1	0	0	1	0	0	1
0	1	1	0	0	1	0	0	1	0	1	0	0	0	0
0	1	1	0	0	1	1	0	1	0	1	0	0	0	1
0	1	1	0	1	0	0	0	1	0	1	0	0	1	1
0	1	1	0	1	0	1	0	1	0	1	0	0	1	1
0	1	1	0	1	1	0	0	1	0	1	0	1	0	0
0	1	1	0	1	1	1	0	1	0	1	0	1	0	1
0	1	1	1	0	0	0	0	1	0	1	0	1	1	0
0	1	1	1	0	0	1	0	1	0	1	0	1	1	1
0	1	1	1	0	1	0	0	1	0	1	1	0	0	0
0	1	1	1	0	1	1	0	1	0	1	1	0	0	1
0	1	1	1	1	0	0	0	1	1	0	0	0	0	0
0	1	1	1	1	0	1	0	1	1	0	0	0	0	1
0	1	1	1	1	1	0	0	1	1	0	0	0	1	1
0	1	1	1	1	1	1	0	1	1	0	0	0	1	1
1	0	0	0	0	0	0	0	1	1	0	0	1	0	0

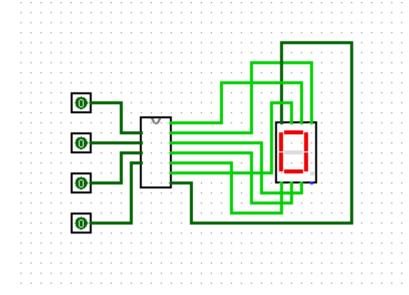
CSE-D	EPS											CSE		Micro	proce	ssor and Computer Organization
	1	0	0	0	0	0	1	0	1	1	0	0	1	0	1	
	1	0	0	0	0	1	0	0	1	1	0	0	1	1	0	
	1	0	0	0	0	1	1	0	1	1	0	0	1	1	1	
	1	0	0	0	1	0	0	0	1	1	0	1	0	0	0	
	1	0	0	0	1	0	1	0	1	1	0	1	0	0	1	
	1	0	0	0	1	1	0	0	1	1	1	0	0	0	0	
	1	0	0	0	1	1	1	0	1	1	1	0	0	0	1	
	1	0	0	1	0	0	0	0	1	1	1	0	0	1	1	
	1	0	0	1	0	0	1	0	1	1	1	0	0	1	1	
	1	0	0	1	0	1	0	0	1	1	1	0	1	0	0	
	1	0	0	1	0	1	1	0	1	1	1	0	1	0	1	
	1	0	0	1	1	0	0	0	1	1	1	0	1	1	0	
	1	0	0	1	1	0	1	0	1	1	1	0	1	1	1	
	1	0	0	1	1	1	0	0	1	1	1	1	0	0	0	
	1	0	0	1	1	1	1	0	1	1	1	1	0	0	1	
	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	
	1	0	1	0	0	0	1	1	0	0	0	0	0	0	1	
	1	0	1	0	0	1	0	1	0	0	0	0	0	1	1	
	1	0	1	0	0	1	1	1	0	0	0	0	0	1	1	
	1	0	1	0	1	0	0	1	0	0	0	0	1	0	0	
	1	0	1	0	1	0	1	1	0	0	0	0	1	0	1	
	1	0	1	0	1	1	0	1	0	0	0	0	1	1	0	
	1	0	1	0	1	1	1	1	0	0	0	0	1	1	1	
	1	0	1	1	0	0	0	1	0	0	0	1	0	0	0	
	1	0	1	1	0	0	1	1	0	0	0	1	0	0	1	
	1	0	1	1	0	1	0	1	0	0	1	0	0	0	0	
	1	0	1	1	0	1	1	1	0	0	1	0	0	0	1	
	1	0	1	1	1	0	0	1	0	0	1	0	0	1	1	
	1	0	1	1	1	0	1	1	0	0	1	0	0	1	1	
	1	0	1	1	1	1	0	1	0	Ō	1	0	1	0	0	
	1	0	1	1	1	1	1	1	0	0	1	0	1	0	1	
	1	1	0	0	0	0	0	1	Ō	Ō	1	0	1	1	0	
	1	1	0	Ō	Ō	Ō	1	1	0	Ō	1	0	1	1	1	
	1	1	0	0	0	1	0	1	0	0	1	1	0	0	0	
	1	1	0	0	0	1	1		0	0	1	1	0			
	1	1	U	U	U	1	1	1	U	U	1	1	U	0	1	

CSE-DEPSTAR

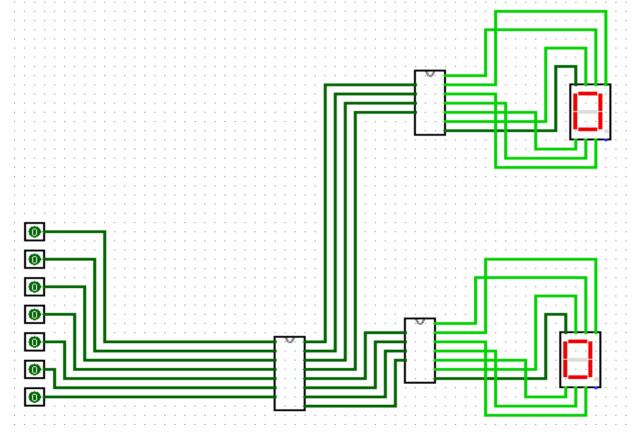
CSE202-Microprocessor and Computer Organization

CIRCUITS:

0 to 9:



0 to 99:

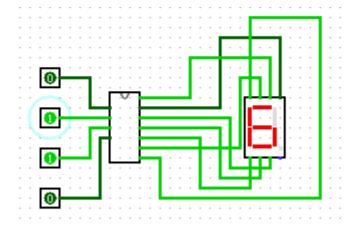


Page No.7

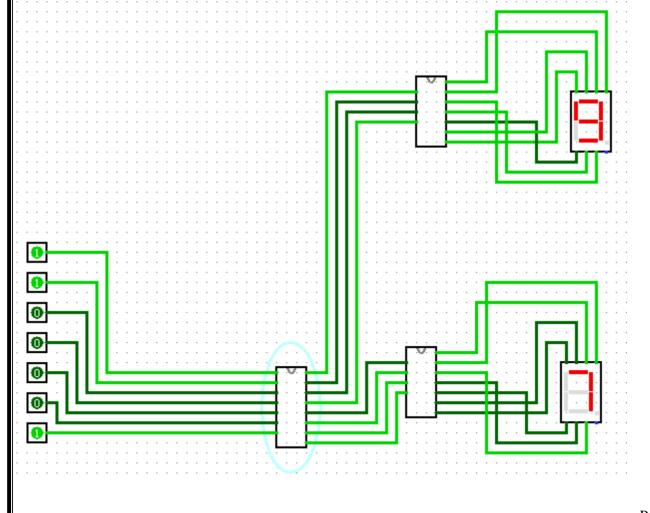
23DCS005

OUTPUTS:

0 to 9:



0 to 99:



Page No.8

23DCS005

COL	T .	CI	200		
() H	-1)	H.H	Σ	IAK	

CONCLUSION:

From this experiment we can conclude that:

- i. We can design a seven segment display with binary number as input.
- ii. We can also design two seven segment displays with binary number as input.
- iii. And we can create 0 to 9 and 0 to 99 counter based on seven segment application.

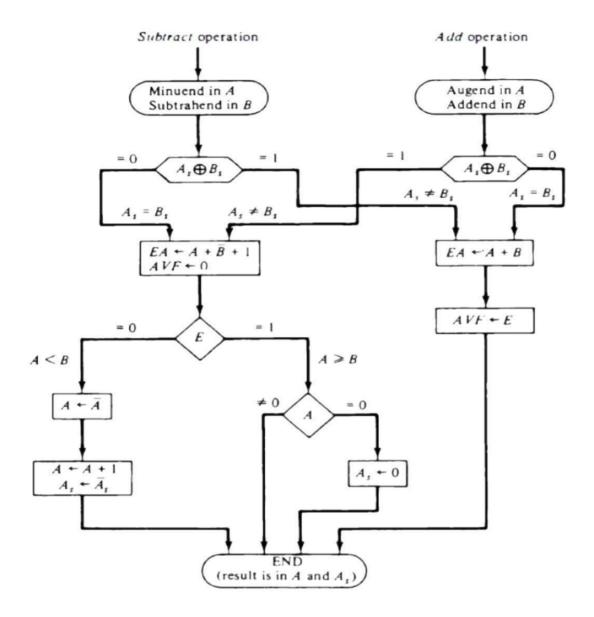
Page No.9

Date:8/7/2024

EXPERIMENT NO. 2

AIM: Implement a circuit in Logisim which perform Addition and Subtraction of sign number.

ALGORITHM:



CSE-DEPSTAR CSE202-Microprocessor and Computer Organization **CIRCUITS:** A B **OUTPUTS:** 1) -(A+B) 0-10-0 00-10 0 E-E-E-E 0.0.0.0 0.0.0.0

Page No.11

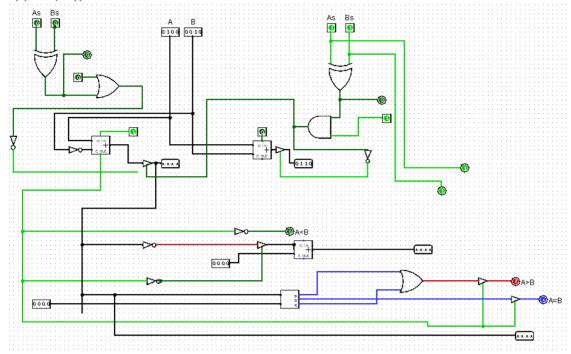
0.0.1.0

23DCS005

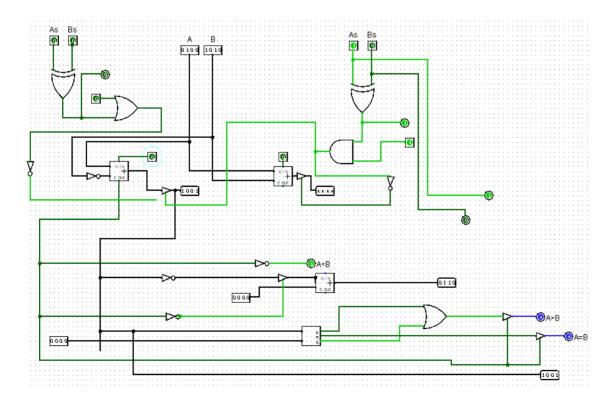
CSE-DEPSTAR

CSE202-Microprocessor and Computer Organization

2)(-A+(-B))=-A-B

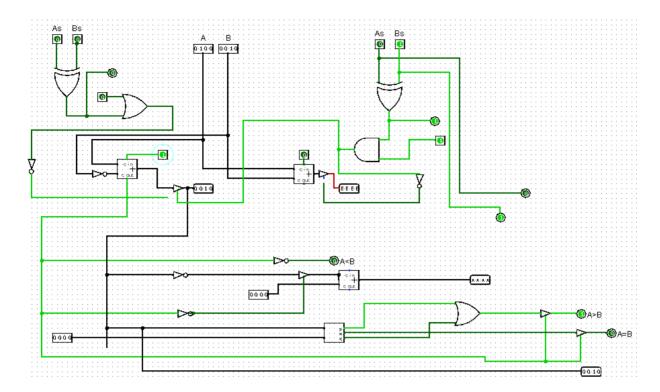


3)(-A+B)

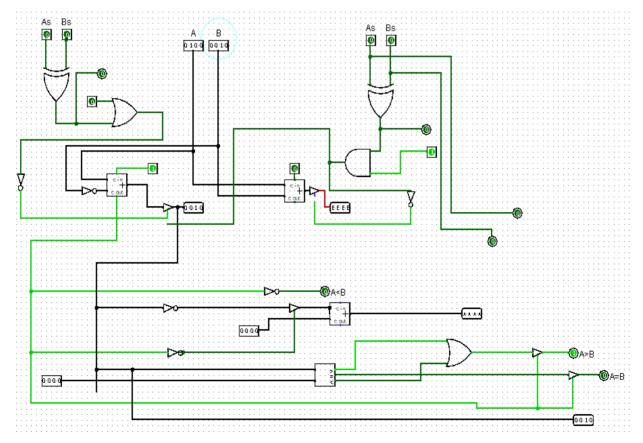


CSE-DEPSTAR $4)(\mathbf{A} - (-\mathbf{B})) = \mathbf{A} + \mathbf{B}$

CSE202-Microprocessor and Computer Organization



5) (**A-B**)

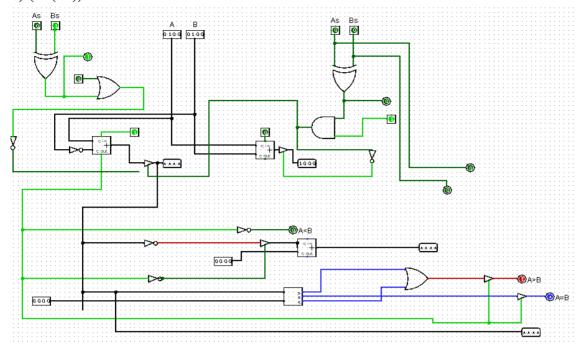


Page No.13

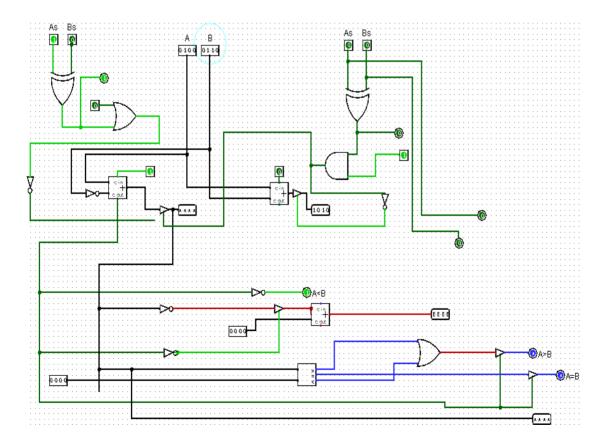
CSE-DEPSTAR

CSE202-Microprocessor and Computer Organization

6) (A-(-B))



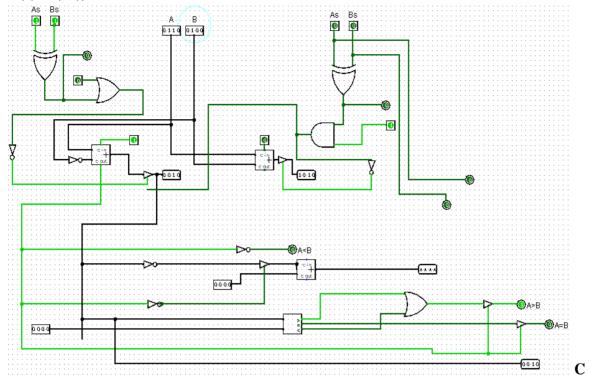
7)(-A-B)



CSE-DEPSTAR

CSE202-Microprocessor and Computer Organization

8 (-A-(-B))



CONCLUSION:

In Logisim, implementing a circuit for the addition and subtraction of signed numbers demonstrates fundamental concepts of digital arithmetic and signed number representation. The circuit effectively handles two's complement arithmetic, ensuring correct results for both positive and negative operands. This practical exercise enhances understanding of binary operations, logic gates, and circuit design principles, reinforcing the core concepts of digital electronics and computer architecture.

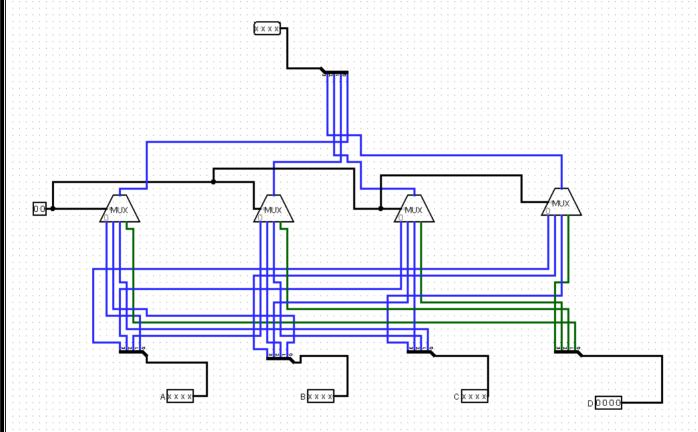
Date: 15/7/2024

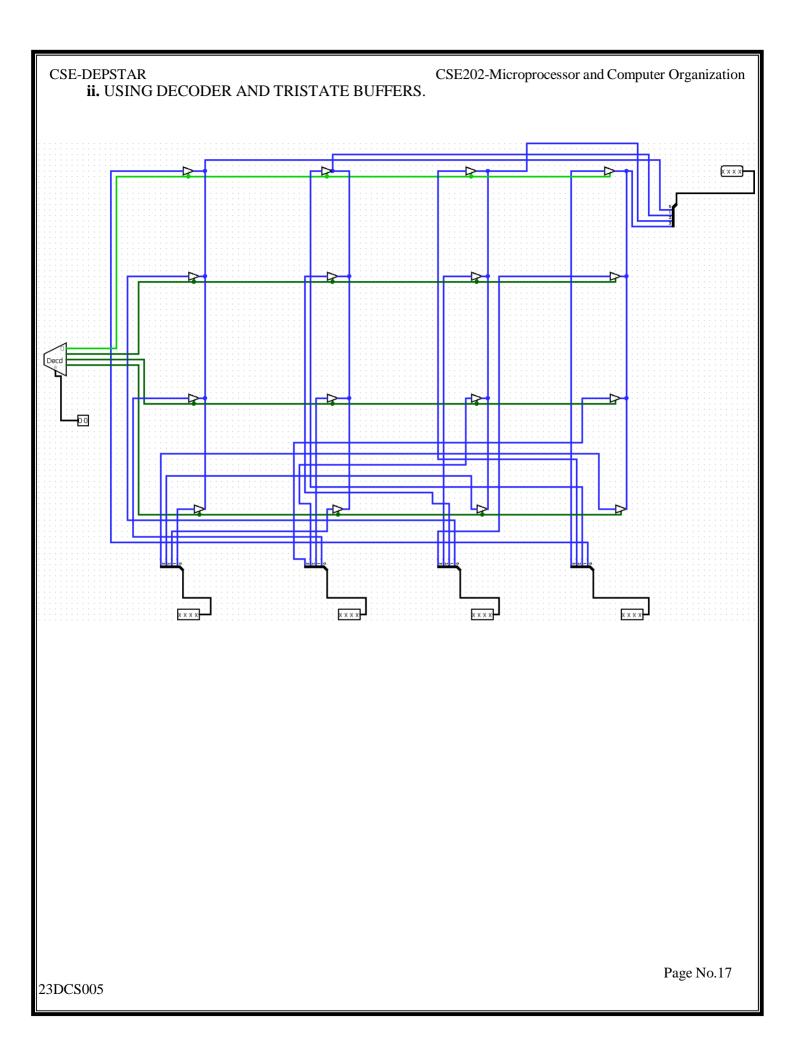
EXPERIMENT NO. 3

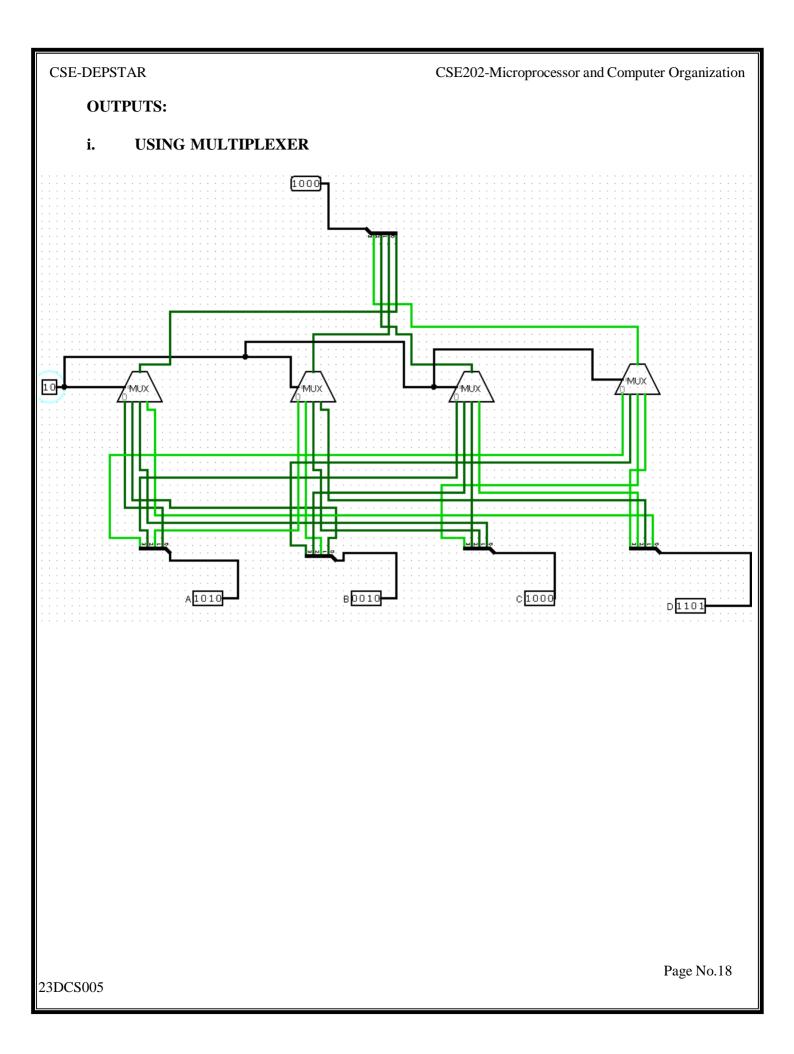
AIM: Implement a 4-bit common bus system to interface four 4-bit registers with a common bus using i. Multiplexer and ii. Decoder and tristate buffers.

CIRCUITS:

i. USING MULTIPLEXER



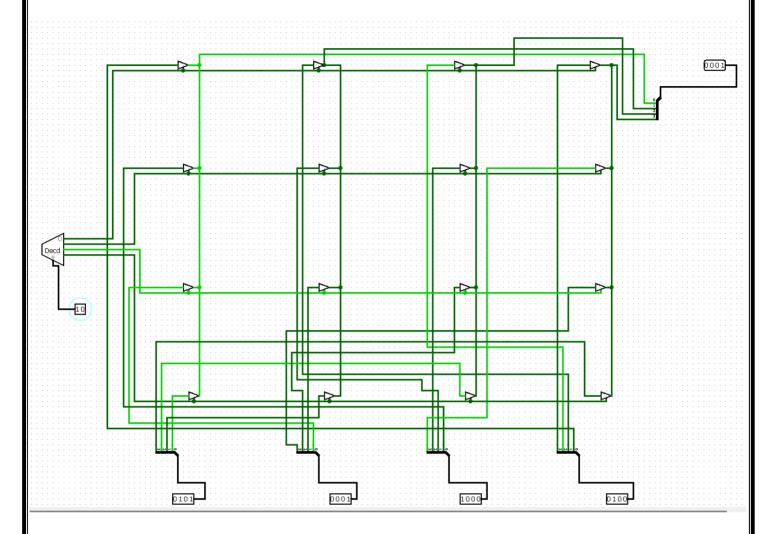




CSE-DEPSTAR

CSE202-Microprocessor and Computer Organization

ii. USING DECODER AND TRISTATE BUFFERS.



CONCLUSION:

Implementing a 4-bit common bus system using multiplexers and decoder with tri-state buffers proved effective in interfacing four 4-bit registers. The multiplexer method efficiently selects and routes data between registers and the bus, offering simplicity in control logic. Conversely, the decoder and tri-state buffers method provided straightforward enable/disable functionality for each register's connection to the bus, ensuring data integrity and minimal bus contention. Both approaches demonstrate practical solutions for interfacing multiple registers with a common bus in digital systems design.

Date: 02/08/2024

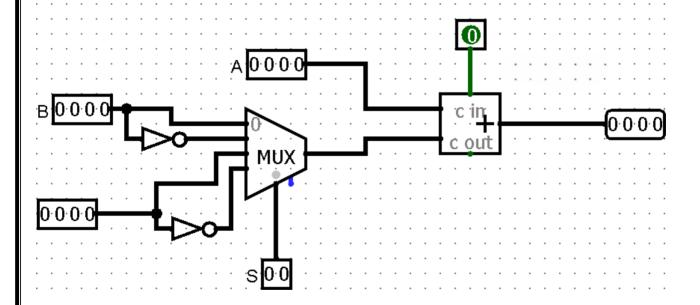
EXPERIMENT NO. 4

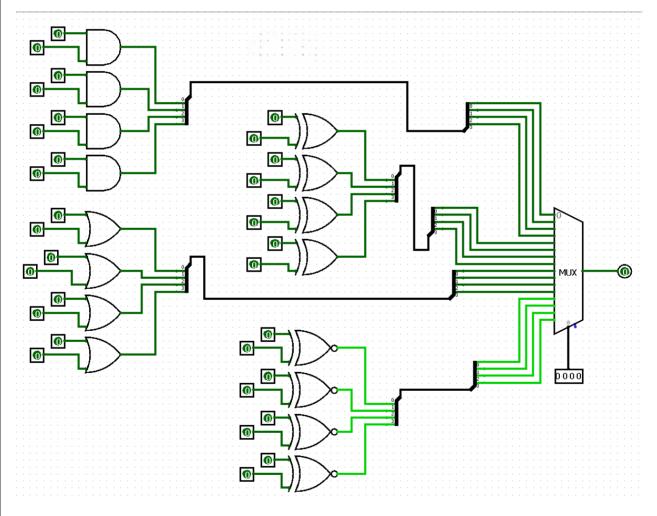
AIM: Implement arithmetic and logic unit circuits in Logisim.

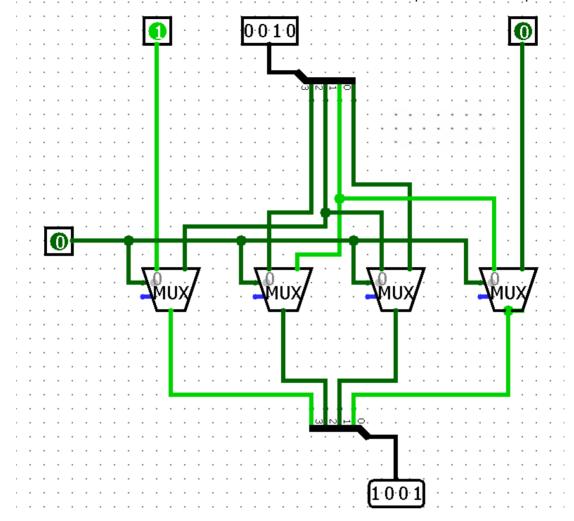
OBJECTIVES:

- i. Implement 1-bit, 2-bit, 4-bit and 8-bit arithmetic unit circuits
- ii. Implement 1-bit, 2-bit, 4-bit and 8-bit logical unit circuits for four logical functions
- iii.Implement 1-bit and 2-bit logical unit circuits for sixteen logical functions
- iv. Implement 2-bit, 4-bit and 8-bit bidirectional shifter
- v. Implement 1-bit, 2-bit, 4-bit and 8-bit ALU

CIRCUITS:

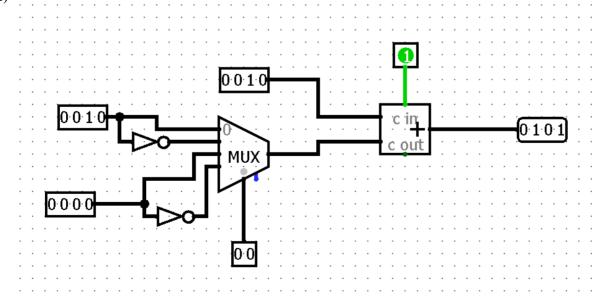


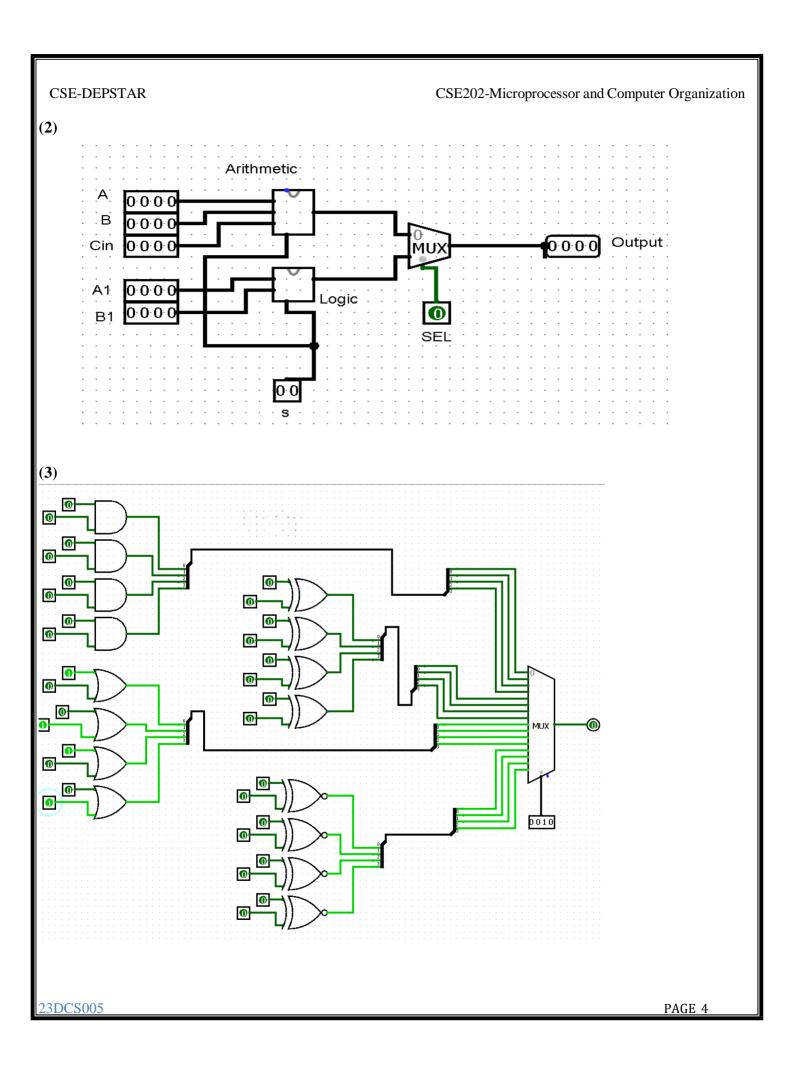


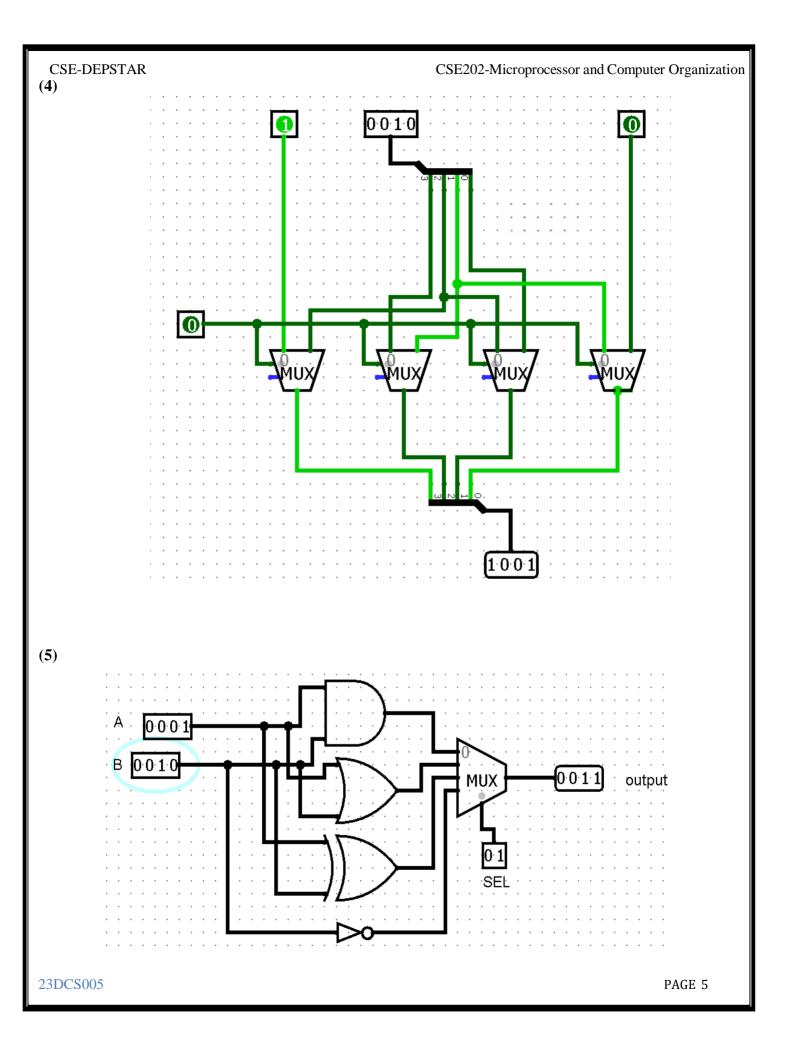


OUTPUTS:

(1)







CONCLUSION:

Implementing arithmetic and logic unit circuits in Logisim involves designing and simulating digital circuits to perform arithmetic and logical operations, enhancingunderstanding of fundamental computer architecture principles.

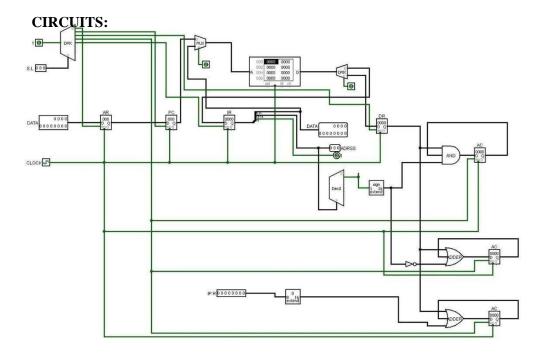
Date: 13/08/23

EXPERIMENT NO. 5

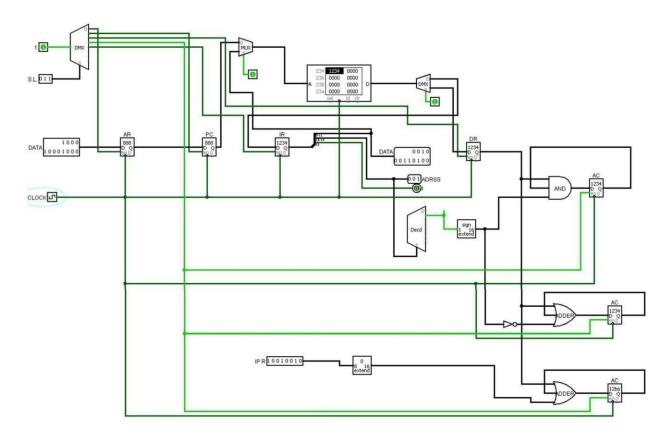
AIM: Implement a common bus system with ALU, 8 registers and 1 memory unit with necessary control signals.

OBJECTIVES:

- i. Implement 16-bit ALU that is suitable for common bus system
- ii. Implement 16-bit common bus system with ALU, 8 registers and 1 memory unit



CSE-DEPSTAR **OUTPUTS:**



CONCLUSION:

By this practical I learnt about implementing a common bus system that interates an ALU Arithmetic logic units multiple registers and memory Unit. I designed a logical a 16 Bit ALU that can Interact with the common bus system, allowing data transfer between ALU.

Date:

EXPERIMENT NO. 6

AIM: Perform following operations with basic assembly language programming:

addition	and	logical left shift	rotate left with carry			
subtraction	or	logical right shift	rotate left without carry			
multiplication	xor	arithmetic left shift	rotate right with carry			
division	not	arithmetic right shift	rotate right without carry			

Objective 1:

Perform addition of two 32-bit numbers i. 1xxx1B64h and ii. 9135F13Ah and store result at memory location 30020h. (consider xxx is last three digits of your enrolment number)

Code:

org 100h

mov ax,1B64h

mov bx,0F13Ah

add ax,bx

mov cx,3000h

mov ds,cx

mov di,0020h

mov [di],ax

mov ax,1005h

mov bx,9135h

adc ax,bx

inc di

inc di

mov [di],ax

ret

		0 = = 0 = 11				
Screenshot	(of the	Value	Operation	(Before	(After
Instruction	in	program	in	executed by	execution of	execution of
memory			Instruc	instruction	instruction)	instruction)
-			tion		Content of to be	Content of
			Pointe		affected	affected
			r		Registers/Mem	Registers/Mem
					ory locations	ory locations
					and flags	and flags
			0100	Store lower bit		
				of number A		
	Instruction	Instruction in	Instruction in program	Instruction in program in Instruction Pointe r	Instruction in program in Instruction instruction Pointe r O100 Store lower bit	Instruction in program memory Instruction Pointe r O100 Store lower bit Executed by execution of instruction Content of to be affected Registers/Mem ory locations and flags

CSE202 Mi

	CSE202-Micr	roprocessor and Com	puter Organization
07100: B8 184 ₹ 07101: 64 100 d 07102: 1B 027 ←	in Ax	AX 00 00	AX 18 64

			CSE202-Micro	oprocessor and Com	puter Organization
mov bx,0F13A h	07103: BB 187 ק 07104: 3A 058 : 07105: F1 241 ±	0103	Store lower bit of number B in Bx	BX 00 00	BX F1 3A
add ax,bx	07106: 03 003 ♥ 07107: C3 195 ├	0106	Add content of Ax and Bx	AX 1B 64	AX OC 9E
				CF S ▼	
mov cx,3000h	07108: B9 185 07109: 00 000 NULL 0710A: 30 048 0	0108	Store segment address in Cx	CX 00 1F	CX 30 00
mov ds,cx	0710B: 8E 142 A 0710C: D9 217	010B	copy content of cx into ds	DS 0700	DS 3000
mov di,0020h	0710D: BF 191 7 0710E: 20 032 SPA 0710F: 00 000 NULL	010D	Store offset address in di	DI 0000	DI 0020
mov [di],ax	07110: 89 137 ë 07111: 05 005 ☆	0110	Store content of Ax At 30020h	30020: 00 000 NULL 30021: 00 000 NULL	30020: 9E 158 R 30021: 0C 012 P
mov ax,1122h	07112: B8 184 ₹ 07113: 22 034 " 07114: 11 017 ◀	0112	Store higher bit of number A in Ax	∆X ØC 9E	AX 11 22
mov bx,9135h	07115: ВВ 187 п 07116: 35 053 5 07117: 91 145 æ	0115	Store higher bit of number B in Bx	BX F1 3A	BX 91 35
adc ax,bx	07118: 13 019 !! 07119: C3 195 }	0118	Add Content Of Ax and Bx	AX C1 22	AX A1 7C
			With Carry	CF 1 ▼	CF S ▼
inc di	0711A: 47 071 G	011A	Increment di	DI 0020	DI 0021
inc di	0711B: 47 071 G	011B	Increment di	DI 0021	DI 6622
mov [di],ax	0711C: 89 137 ë 0711D: 05 005 ♠	011C	Store content of Ax At 30022h	30022: 00 000 NULL 30023: 00 000 NULL	30022: 78 120 x 30023: A1 161 1

Objective 2:

Perform Subtraction of two 32-bit numbers i. 1xxx1B64h and ii. 9135F13Ah and store result at memory location 30040h. (consider xxx is last three digits of your enrolment number)

Code: org 100h mov ax,1B64h mov bx,0F13Ah sub ax,bx mov cx,3000h mov ds,cx mov di,0040h mov [di],ax mov ax,1005h mov bx,9135h sbb ax,bx inc di inc di mov [di],ax ret

	Consombat of the	V al	On and 4:	(Dofore	(A C4 ore
Instructi	Screenshot of the		Operation	(Before	(After
on in	Instruction in program	in	executed by		
Assembl	memory	Instruc	instruction	instruction)	instruction)
y		tion		Content of to be	Content of
Languag		Pointe		affected	affected
e		r		Registers/Mem	Registers/Mem
				ory locations	ory locations
				and flags	and flags
mov	07100: B8 184 a	0100	Store lower bit	H L	_H_L
ax,1B64h	07101: 64 100 d		of number Ain	AX 00 00	AX 1B 64
W.2,22012	07102: 1B 027 ←		Ax	700	, ,
mov	07103: BB 187 п	0103	Store lower bit	BX 00 00	BV F4 00
bx,0F13A	07104: 3A 058 :	0100	of number B in		BX F1 3A
h	07105: F1 241 ±		Bx		
	07106: 2B 043 +	0106	Sub content of	H L	HL
sub ax,bx	07107: C3 195 H	0100		AX 1B 64	
	0.10.1		Ax and Bx	AX 1-0 1-1	AX 2A 2A
				CF 6 ▼	05 4 -
				CI D	CF 1 <u>▼</u>
mov	07108: B9 185 {	0108	Store segment	CX 00 1F	CX 30 00
cx,3000h	07109: 00 000 NULL 0710A: 30 048 0		address in Cx	CX OO II	CX OU OU
mov ds,cx	0710B: 8E 142 A	010B		07.00	0000
mov us,cx	0710C: D9 217	OIOD	copy content of	DS 0700	DS 3000
	0710D: BF 191 7	010D	cx into ds		
mov	0710E: 40 064 C	010D	Store offset	DI 0000	DI 0040
di,0040h	0710F: 00 000 NULL		address in di		
mov	07110: 89 137 ë	0110	Store content	30040: 00 000 NULL 30041: 00 000 NULL	30040: 2A 042 * 30041: 2A 042 *
[di],ax	07111: 05 005 ♠		of Ax At		
			30040h		
mov	07112: B8 184 q	0112	Store higher bit	registers	HLL
ax,1122h	07113: 22 034 ° 07114: 11 017 ◀	~ -	of number A in		AX 11 22
ungi 12211	87117- 11 817 V		Ax	AX 2A 2A	
mov	07115: BB 187 a	0115		DV F4 06	64 65
mov	07116: 35 053 5	0113	Store higher bit		BX 91 35
bx,9135h	07117: 91 145 æ		of number B in		
		0440	Bx		
sbb ax,bx	07118: 1B 027 ← 07119: C3 195 }	0118	Sub Content Of		H L
	9/11/- 03 1/3 F		Ax and Bx	AX C1 22	AX 7F 18
			With Barrow	CF 1 ▼	05
					CF 1 <u>▼</u>
	<u> </u>		l		

inc di	0711A: 47 071 G	011A	Incriment di	DI	0040	DI	0041
inc di	0711B: 47 071 G	011B	incriment di	DI	0041	DI	0042
mov [di],ax	0711C: 89 137 ë 0711D: 05 005 ❖	011C	Store content of Ax At 30122h	30042: 00 30043: 00	9 000 NULL	30042: 0 0 30043: 7 1	

Objective 3:

Perform Multiplication of two 16-bit numbers i. Exxxh and ii. A2B3h and store result at memory location 30060h. (consider xxx is last three digits of your enrolment number)

Code:

org 100h

mov ax,0E005h

mov dx,0A2B3h

mul dx

mov cx,3000h

mov ds,cx

mov di,0060h

mov [di],ax

inc di

inc di

mov [di],dx

ret

	DI-DIEI EXECUTION.				
Instructi	Screenshot of the	Value	Operation	(Before	(After
on in	Instruction in program	in	executed by	execution of	execution of
Assembl	memory	Instruc	instruction	instruction)	instruction)
y	-	tion		Content of to be	Content of
Languag		Pointe		affected	affected
e		r		Registers/Mem	Registers/Mem
		_		ory locations	O
				and flags	and flags
mov ax,0E122	07100: B8 184 q 07101: 22 034 " 07102: E1 225 B	0100	Store lower bit of number A	AX 00 00	AX E1 22
h			in Ax		
Mov dx,0A2B3 h	07103: BA 186 07104: B3 179 07105: A2 162 6	0103	Store lower bit of number B in Dx	D/1	DX A2 B3
mul dx	07106: F7 247 ≈ 07107: E2 226 Γ	0106	Mul content of Ax and Bx Store lower Bit in Ax and higher bit in Dx	AX C1 22	AX 10 F2 DX 8E 89 CF 1 ▼
mov cx,3000h	07108: B9 185 07109: 00 000 NULL 0710A: 30 048 0	0108	Store segment address in Cx	CX 00 17	CX 30 00

mov ds,cx	0710B: 8E 142 ä 0710C: D9 217 j	010B	copy content of cx into ds	DS 0700	DS 3000
mov di,0060h	0710D: BF 191 7 0710E: 60 096 0710F: 00 000 NULL	010D	Store offset address in di	DI 6868	DI 8969
mov [di],ax	07110: 89 137 ë 07111: 05 005 ♠	0110	Store content of Ax At 30060h		
inc di	07112: 47 071 G	0112	Incriment di	DI 0060	DI 8861
inc di	07113: 47 071 G	0113	incriment di	DI 0061	DI 0062
mov [di],ax	07114: 89 137 ë 07115: 15 021 §	0114	Store content of Dx At 30062h	===== === =============================	30062: 86 134 & 30063: 8E 142 ä

Objective 4:

Perform Division on Exxxh by 0777h and store result at memory location 30070h. (consider xxx is last three digits of your enrolment number)

Code:

org 100h

mov ax,0E005h

mov bx,0777h

div bx

mov cx,3000h

mov ds,cx

mov di,0070h

mov [di],ax

ret

Instructi	Screenshot of the	Value	Operation	(Before	(After
on in	Instruction in program	in	executed by	execution of	execution of
Assembl	memory	Instruc	instruction	instruction)	instruction)
y	·	tion		Content of to be	Content of
Languag		Pointe		affected	affected
l e		r		Registers/Mem	Registers/Mem
				ory locations	0
				and flags	and flags
mov	07100: B8 184 3	0100	Store number	registers H L	H L
ax,0E122	07101: 22 034 ° 07102: E1 225 β		A	AX 00 00	AX E1 22
h			in Ax	AV 100 100	. j==
Mov	07103: BB 187 T	0103	Store lower bit	BX 00 00	BX 07 77
dx,0777h	07104: 77 119 W		of number B in		bx or rr
,	07105: 07 007 BEEP		Bx		
div bx	07106: F7 247 ≈	0106	Div content of	_H_L	H L
	07107: Е2 226 Г		Ax and Bx	AX C1 22	AX 00 1E
			Store in Ax	,	
		I .			

mov cx,3000h	07108: B9 185 { 07109: 00 000 NULL 0710A: 30 048 0	0108	Store segment address in Cx	CX 00 13	CX 30 00
mov ds,cx	0710B: 8E 142 A 0710C: D9 217	010B	copy content of cx into ds	DS 0700	DS 3000
mov di,0070h	0710D: BF 191 7 0710E: 70 112 p 0710F: 00 000 NULL	010D	Store offset address in di	DI 9999	DI 0070
mov [di],ax	07110: 89 137 ë 07111: 05 005 ♠	0110	Store content of Ax At 30070h	30070: 00 000 NULL 30071: 00 000 NULL	30070: 1E 030 ▲ 30071: 00 000 NULL

Objective 5:

Perform ANDing on Cxxxh by 00FFh and store result at memory location 30080h. (consider xxx is last three digits of your enrolment number)

Code:

org 100h

mov ax,0C005h

mov bx,00FFh

and ax,bx

mov cx,3000h

mov ds,cx

mov di,0080h

mov [di],ax

ret

<u> </u>	DI-SIEI EXECUTION.				
Instructi	Screenshot of the	Value	Operation	(Before	(After
on in	Instruction in program	in	executed by	execution of	execution of
Assembl	memory	Instruc	instruction	instruction)	instruction)
y		tion		Content of to be	Content of
Languag		Pointe		affected	affected
e		r		Registers/Mem	Registers/Mem
				ory locations	ory locations
				and flags	and flags
mov	07100: B8 184 7 07101: 22 034 "	0100	Store number	registers — H L	_ H _ L
ax,0C122	07102: C1 193 ±		A	AX 00 00	AX C1 22
h			in Ax	W/ 00 00	
Mov	07103: BB 187 7 07104: FF 255 RES	0103	Store number	BX 00 00	BX 00 FF
dx,00FFh	07104: FF 255 RES 07105: 00 000 NULL		B in Bx		
and ax,bx	07106: 23 035 #	0106	and content of	_H_L	_H_L
	07107: C3 195 }		Ax and Bx	AX C1 22	AX C1 22
			Store in Ax	, ,	, ,
mov	07108: B9 185	0108	Store segment	CX 66 13	CX 30 00
cx,3000h	07109: 00 000 NULL 0710A: 30 048 0		address in Cx		
mov ds,cx	0710B: 8E 142 A	010B	copy content of	DS 0700	DS 3000
	0710C: D9 217 J		cx into ds	55 51 55	00 0000
L			l		

mov di,0080h	0710D: BF 191 7 0710E: 80 128 C 0710F: 00 000 NULL	010D	Store offset address in di	DI 9999	DI 0080
mov	07110: 89 137 ë	0110	Store contentof	30080: 00 000 NULL	30080: 42 066 B
[di],ax	07111: 05 005 ☆		Ax At 30080h	30081: 00 000 NULL	30081: 00 000 NULL

Objective 6:

Perform ORing on Cxxxh by 00FFh and store result at memory location 300A0h. (consider xxx is last three digits of your enrolment number)

Code:

org 100h

mov ax,0C005h

mov bx,00FFh

or ax,bx

mov cx,3000h

mov ds,cx

mov di,00A0h

mov [di],ax

ret

5111	-B1-SIEF EXECUTION:				
Instructi	Screenshot of the		Operation	(Before	(After
on in	Instruction in program	in	executed by	execution of	execution of
Assembl	memory	Instruc	instruction	instruction)	instruction)
y		tion		Content of to be	Content of
Languag		Pointe		affected	affected
e		r		Registers/Mem	Registers/Mem
				ory locations	ory locations
				and flags	and flags
mov	07100: B8 184 7 07101: 22 034 "	0100	Store number	H_L	H L
ax,0C122	07101: 22 034 " 07102: C1 193 1		A	AX 00 00	AX C1 22
h			in Ax		
Mov	07103: BB 187 7	0103	Store number	BX 00 00	BX 00 FF
dx,00FFh	07104: FF 255 RES 07105: 00 000 NULL		B in Bx		אלן ייט
or ax,bx	07106: 0B 011 &	0106	or content of Ax	HL	HLL
	07107: C3 195 }		and Bx Store in	AX C1 22	AX CO FF
			Ax	,	
mov	07108: B9 185	0108	Store segment	CX 00 13	CX 30 00
cx,3000h	07109: 00 000 NULL 0710A: 30 048 0		address in Cx		
mov ds,cx	0710B: 8E 142 ä	010B	copy content of	DS 0700	DS 3000
	0710C: D9 217 J		cx into ds		
mov	0710D: BF 191 1	010D	Store offset	DI 0000	DI 00A0
di,00A0h	0710E: 80 128 ¢ 0710F: 00 000 NULL		address in di		51 55116
		0110		300AO: OO OOO NIIT.L	300A0: FF 255 RES
mov	07110: 89 137 ë 07111: 05 005 ♠	0110	Store contentof	300A0: 00 000 NULL 300A1: 00 000 NULL	300A1: CO 192 L
[di],ax	57111 · 65 665 2		Ax At 300A0h		

Objective 7:

Perform XORing on Cxxxh by 00FFh and store result at memory location 300B0h. (consider xxx is last three digits of your enrolment number)

Code:

org 100h

mov ax,0C005h

mov bx,00FFh

xor ax,bx

mov cx,3000h

mov ds,cx

mov di,00B0h

mov [di],ax

ret

STEP-BY-STEP EXECUTION:

T4	Companded of the	Value	Oneration	(Dofore	(A Ctore
Instructi	Screenshot of the	_	Operation	(Before	(After
on in	Instruction in program		executed by		
Assembl	memory	Instruc	instruction	instruction)	instruction)
y		tion		Content of to be	Content of
Languag		Pointe		affected	affected
e		r		Registers/Mem	Registers/Mem
				ory locations	\cup
				and flags	and flags
mov	07100: B8 184 7	0100	Store number	H_L	H_L
ax,0C122	07101: 22 034 °	J100	A	AX 00 00	AX C1 22
h	17102. GI 193 -		in Ax		
Mov	07103: BB 187 a	0103	Store number	ny 00 00	00 55
	07104: FF 255 RES	0103	B in Bx	BX 00 00	BX 00 FF
dx,00FFh	07105: 00 000 NULL		B In BX		
xor ax,bx	07106: OB 011 &	0106	xor content of		HL
	07107: C3 195 }		Ax and Bx	AX C1 22	AX CO B9
			Store in Ax	,	 -
					 -
mov	07108: B9 185	0108	Store segment	CX 66 13	CX 30 00
cx,3000h	07109: 00 000 NULL 0710A: 30 048 0		address in Cx	Gr. 55 15	
mov ds,cx	0710B: 8E 142 A	010B	copy content of	DS 0700	2000
mov us,cx	0710C: D9 217	OIOD	cx into ds	DS 0700	DS 3000
200 0 2 1	0710D: BF 191 7	010D		DI 0000	
mov	0710E: 80 128 C	กากก		DI 0000	DI 00B 0
di,00B0h	0710F: 00 000 NULL		address in di		 -
mov	07110: 89 137 ë	0110	Store content	300B0: 00 000 NULL 300B1: 00 000 NULL	300B0: BD 189 L 300B1: CO 192 L
[di],ax	07111: 05 005 🛧	-	of Ax At	SOUDI. OO OOO HOLL	200DI: OB 172 -
[],			300B0h		 -
				l	

Objective 8:

Perform NOT operation on Cxxxh and store result at memory location 300C0h. (consider xxx is last three digits of your enrolment number) Code:

org 100h mov ax,0C005h not ax mov cx,3000h mov ds,cx mov di,00C0h mov [di],ax ret

STEP-BY-STEP EXECUTION:

Instructi	Screenshot of the	Value	Operation	(Before	(After
on in	Instruction in program	in	executed by	execution of	execution of
Assembl	memory	Instruc	instruction	instruction)	instruction)
y		tion		Content of to be	Content of
Languag		Pointe		affected	affected
e		r		Registers/Mem	Registers/Mem
				ory locations	· ·
				and flags	and flags
mov	07100: B8 184 7 07101: 22 034 "	0100	Store number	H_L	H_L
ax,0C122	07102: C1 193 ±		A	AX 00 00	AX C1 22
h			in Ax		,
not ax	07103: F7 247 ≈ 07104: D0 208 ¹¹	0103	not content of	HL	AX 3F B9
	07104: DO 208 H		Ax Store in Ax	AX C1 22	AV 101 102
mov	07105: B9 185 {	0105	Store segment	CX 00 10	CX 30 00
cx,3000h	07106: 00 000 NULL	0105	address in Cx	CX 00 10	CX 30 00
	07107: 30 048 0 07108: 8E 142 ä	0100			
mov ds,cx	07108: 8E 142 H	0108	copy content of	DS 0700	DS 3000
	01107- 07 21	0404	cx into ds		
mov	0710A: BF 191 0710B: CO 192	010A	Store offset	DI 0000	DI 0000
di,00C0h	0710C: 00 000 NULL		address in di		
mov	07110: 89 137 ë	0110	Store contentof	300C0: 00 000 NULL 300C1: 00 000 NULL	300C0: BD 189 H 300C1: 3F 063 ?
[di],ax	07111: 05 005 🛧		Ax At 300C0h		
			1	i .	

Objective 9:

Perform logical left shift on Cxxxh and store result at memory location 300D0h. (consider xxx is last three digits of your enrolment number)

Code:

org 100h

mov ax,0C005h

shl ax ,1

mov cx,3000h

mov ds,cx

mov di,00D0h

mov [di],ax

ret

STEP-BY-STEP EXECUTION:

Instructi on in Assembl y Languag e	Screenshot of the Instruction in program memory		Operation executed by instruction	(Before execution of instruction) Content of to be affected Registers/Mem ory locations and flags	instruction) Content of affected Registers/Mem
mov	07100: B8 184 q 07101: 22 034 "	0100	Store number	H L	H L
ax,0C122 h	07102: C1 193 ¹		A in Ax	AX 00 00	AX C1 22
shl ax,1	07103: D1 209 τ 07104: E0 224 α	0103	Shift Left content of Ax Store in Ax		AX 80 8C
mov cx,3000h	07105: B9 185 07106: 00 000 NULL 07107: 30 048 0	0105	Store segment address in Cx	CX 00 10	CX 30 00
mov ds,cx	07108: 8E 142 ä 07109: D9 217 J	0108	copy content of cx into ds	DS 0700	DS 3000
mov di,00D0h	0710A: BF 191 1 0710B: C0 192 1 0710C: 00 000 NULL	010A	Store offset address in di	DI 6999	DI 9909
mov [di],ax	07110: 89 137 ë 07111: 05 005 ♠	0110	Store content of Ax At 300D0h	SOUDE: OF SOU HOLL	300D0: 84 132 ä 300D1: 80 128 C

Objective 10:

Perform logical right shift on Cxxxh and store result at memory location 300D0h. (consider xxx is last three digits of your enrolment number)

Code:

org 100h

mov ax,0C005h

shr ax,1

mov cx,3000h

mov ds,cx

mov di,00D0h

mov [di],ax

ret

Instructi	Screenshot	of	the	Value	Operation		(Before	(After
on in	Instruction	in pro	ogram	in	executed	by	execution of	execution of
Assembl	memory			Instruc	instruction		instruction)	instruction)
y				tion			Content of to be	Content of
Languag				Pointe			affected	affected
e				r			Registers/Mem	Registers/Mem
							ory locations	ory locations
							and flags	and flags

			CDL202 WHE	oprocessor and con	
mov ax,0C122	07100: B8 184 7 07101: 22 034 " 07102: C1 193 1	0100	Store number	AX 00 00	registers H L AX C1 22
h			in Ax	AV 100 100	
shr ax,1	07103: D1 209 ∓ 07104: E8 232 ፬	0103	Shift Right content of Ax Store in Ax	AX C1 22	AX 68 23
mov cx,3000h	07105: B9 185 07106: 00 000 NULL 07107: 30 048 0	0105	Store segment address in Cx	CX 00 10	CX 30 00
mov ds,cx	07108: 8E 142 A 07109: D9 217	0108	copy content of cx into ds	DS 0700	DS 3000
mov di,00D0h	0710A: BF 191 0710B: C0 192 0710C: 00 000 NULL	010A	Store offset address in di	DI 9999	DI 890 9
mov [di],ax	07110: 89 137 ë 07111: 05 005 ♠	0110	Store content of Ax At 300D0h	300D0: 00 000 NULL 300D1: 00 000 NULL	300D0: 21 033 ! 300D1: 60 096

Objective 11:

Perform arithmetic left shift on Cxxxh and store result at memory location 300E0h. (consider xxx is last three digits of your enrolment number)

Code:

org 100h

mov ax,0C005h

sal ax,1

mov cx,3000h

mov ds,cx

mov di,00E0h

mov [di],ax

ret

	-DI-SIEI EXECUTION.				
Instructi	Screenshot of the	Value	Operation	(Before	(After
on in	Instruction in program	in	executed by	execution of	execution of
Assembl	memory	Instruc	instruction	instruction)	instruction)
y	_	tion		Content of to be	Content of
Languag		Pointe		affected	affected
e		r		Registers/Mem	Registers/Mem
				ory locations	ory locations
				and flags	and flags
mov ax,0C122	07100: B8 184 7 07101: 22 034 " 07102: C1 193 1	0100	Store number A	AX 00 00	AX C1 22
h			in Ax		
sal ax,1	07103: D1 209 - 07104: E0 224 α	0103	Arithmetic Shift Left	AX C1 22	AX 80 8C
			content of Ax		
	00.05		Store in Ax		
mov cx,3000h	07105: B9 185 07106: 00 000 NULL 07107: 30 048 0	0105	Store segment address in Cx	CX 99 19	CX 30 00

mov ds,cx	07108: 8E 142 ä 07109: D9 217 J	0108	copy content of cx into ds	DS 0700	DS 3000
mov di,00E0h	0710A: BF 191 1 0710B: CO 192 1 0710C: 00 000 NULL	010A	Store offset address in di	DI 6000	DI 00E0
mov [di],ax	07110: 89 137 ë 07111: 05 005 ☆	0110	Store contentof Ax At 300E0h	300E0: 00 000 NULL 300E1: 00 000 NULL	300E0: 84 132 ä 300E1: 80 128 Ç

Objective 12:

Perform arithmetic right shift on Cxxxh and store result at memory location 300F0h. (consider xxx is last three digits of your enrolment number)

Code:

org 100h

mov ax,0C005h

sar ax,1

mov cx,3000h

mov ds,cx

mov di,00F0h

mov [di],ax

ret

	DI-BIEL EXECUTION.				
Instructi	Screenshot of the	Value	Operation	(Before	(After
on in	Instruction in program	in	executed by	execution of	execution of
Assembl	memory	Instruc	instruction	instruction)	instruction)
y		tion		Content of to be	Content of
Languag		Pointe		affected	affected
e		r		Registers/Mem	Registers/Mem
				ory locations	ory locations
				and flags	and flags
mov	07100: B8 184 7 07101: 46 070 F	0100	Store number	H L	_H_L
ax,0C122	07102: CO 192		A	AX 00 00	AX C1 22
h			in Ax	, , , ,	, ,
sar ax,1	07103: D1 209 =	0103	Arithmetic	_H_L	H_L_
	07104: F8 248 °		Shift Right	AX C1 22	AX E 0 23
			content of Ax	, ,	
			Store in Ax		
mov	07105: B9 185	0105	Store segment	CX 00 10	CX 30 00
cx,3000h	07106: 00 000 NULL 07107: 30 048 0		address in Cx	, , , ,	
mov ds,cx	07108: 8E 142 A	0108	copy content of	DS 0700	DS 3000
	07109: D9 217 J		cx into ds	55 01 00	03 0000
mov	0710A: BF 191 7	010A	Store offset	DI 0000	DI 00F0
di,00F0h	0710B: C0 192 L		address in di	51 0000	51 001 0
	0710C: 00 000 NULL 07110: 89 137 ë	011D		300F0: 00 000 NULL	300F0: 21 033 !
mov	07111: 05 005 4	011D	Store content		300F1: EØ 224 a
[di],ax	0,111, 00 000 1		of Ax At		
			300F0h		

Objective 13:

Perform rotate left shift with carry on Cxxxh and store result at memory location 30100h. (consider xxx is last three digits of your enrolment number)

Code:

org 100h

mov ax,0C005h

rcl ax,1

mov cx,3000h

mov ds,cx

mov di,0100h

mov [di],ax

ret

STEP-BY-STEP EXECUTION:

51121	DI-SIEI EXECUTION.				
Instructi		Value	Operation	(Before	(After
on in	Instruction in program	in	executed by	execution of	execution of
Assembl	memory	Instruc	instruction	instruction)	instruction)
y		tion		Content of to be	
Languag		Pointe		affected	affected
e		r		Registers/Mem	Registers/Mem
				ory locations	· ·
				and flags	and flags
mov	07100: B8 184 q 07101: 46 070 F	0100	Store number	H L	HL
ax,0C122	07102: CO 192		A	AX 00 00	AX C1 22
h			in Ax		
rcl ax,1	07103: D1 209 T	0103	Rotate Left		AX 80 8C
	07104: DO 208 ¹¹		Shift with	AX C1 22	AX 80 8C
			carry content	, ,	
			of Ax Store in		
			Ax		
mov	07105: B9 185 07106: 00 000 NULL	0105	Store segment	CX 00 10	CX 30 00
cx,3000h	07100: 00 000 NULL 07107: 30 048 0		address in Cx		
mov ds,cx	07108: 8E 142 ä	0108	copy content of	DS 0700	DS 3000
	07109: D9 217 J		cx into ds	23 01 00	03 0000
mov	0710A: BF 191 7	010A	Store offset	DI 0000	DI 0100
di,0100h	0710B: C0 192 L		address in di	5. 5556	0100
·	0710C: 00 000 NULL 07110: 89 137 ë	0110		30100: 00 000 NULL 30101: 00 000 NULL	30100: 84 132 ä
mov	07111: 05 005 4	0110	Store content		30101: 80 128 C
[di],ax	0,111, 00 000 1		of Ax At		
			30100h		

Objective 14:

Perform rotate left shift without carry on Cxxxh and store result at memory location 30110h. (consider xxx is last three digits of your enrolment number)

Code:

org 100h

mov ax,0C005h

rol ax,1

mov cx,3000h

mov ds,cx mov di,0110h mov [di],ax ret

STEP-BY-STEP EXECUTION:

Instructi on in Assembl y Languag e	Screenshot of the Instruction in program memory	in Instruc tion Pointe r	Operation executed by instruction	(Before execution of instruction) Content of to be affected Registers/Mem ory locations and flags	instruction) Content of affected Registers/Mem
mov ax,0C122 h	07100: B8 184 7 07101: 46 070 F 07102: C0 192 L	0100	Store number A in Ax	AX 00 00	AX C1 22
rol ax,1	07103: D1 209 T 07104: C0 192 T	0103	Rotate Left Shift without carry content of Ax Store in Ax	AX C1 22	AX 80 8D
mov cx,3000h	07105: B9 185 07106: 00 000 NULL 07107: 30 048 0	0105	Store segment address in Cx	CX 00 1F	CX 30 00
mov ds,cx	07109: D9 217 J	0108	copy content of cx into ds	DS 0700	DS 3000
mov di,0110h	0710A: BF 191 1 0710B: C0 192 1 0710C: 00 000 NULL	010A	Store offset address in di	DI 9999	DI 0110
mov [di],ax	07110: 89 137 ë 07111: 05 005 ☆	0110	Store content of Ax At 30110h	000000000000000000000000000000000000000	30110: 85 133 à 30111: 80 128 C

Objective 15:

Perform rotate right shift with carry on Cxxxh and store result at memory location 30120h. (consider xxx is last three digits of your enrolment number) Code:

org 100h

mov ax,0C005h

rcr ax,1

mov cx,3000h

mov ds,cx

mov di,0120h

mov [di],ax

ret

Instructi	Screenshot of the	Value	Operation	(Before	(After
on in	Instruction in program	in	executed by	execution of	execution of
Assembl	memory	Instruc	instruction	instruction)	instruction)
y	-	tion		Content of to be	Content of
Languag		Pointe		affected	affected
e		r		Registers/Mem	Registers/Mem
				ory locations	ory locations
				and flags	and flags
mov	07100: B8 184 q 07101: 46 070 F	0100	Store number	H L	_H_L
ax,0C122	07102: CO 192 L		A	AX 00 00	AX C1 22
h			in Ax	, , ,	, , ,
rcr ax,1	07103: D1 209 =	0103	Rotate Right	_ H_ L	AX 60 23
	07104: D8 216 +		Shift with	AX C1 22	A/ 100 120
			carry content		
			of Ax Store in		
			Ax		
mov	07105: B9 185 { 07106: 00 000 NULL	0105	Store segment	CX 00 10	CX 30 00
cx,3000h	07107: 30 048 0		address in Cx		, , , , , , , , , , , , , , , , , , , ,
mov ds,cx		0108	copy content of	DS 0700	DS 3000
,	07109: D9 217 J		cx into ds	55 01 00	03 0000
mov	0710A: BF 191 7	010A	Store offset	DI 0000	DI 0400
di,0120h	0710B: C0 192 L		address in di	D. 0000	DI 0120
	0710C: 00 000 NULL 07110: 89 137 ë	0110		30120: 00 000 NULL	30120: 21 033 !
mov	07111: 05 005 \$	0110	Store content	30121: 00 000 NULL	30120: 21 033 ! 30121: 60 096
[di],ax	00 000 1		of Ax At		
			30120h		

Objective 16:

Perform rotate right shift without carry on Cxxxh and store result at memory location 30130h. (consider xxx is last three digits of your enrolment number)

Code:

org 100h

mov ax,0C005h

ror ax,1

mov cx,3000h

mov ds,cx

mov di,0130h

mov [di],ax

ret

Instructi	Screenshot	0	f the	Value	Operation		(Before	(After
on in	Instruction	in	program	in	executed	by	execution of	execution of
Assembl	memory			Instruc	instruction		instruction)	instruction)
y				tion			Content of to be	Content of
Languag				Pointe			affected	affected
e				r			Registers/Mem	Registers/Mem
							ory locations	ory locations
							and flags	and flags

mov	07100: B8 184 q 07101: 46 070 F	0100	Store number	ΔX 00 00	H L C1 22
ax,0C122	07102: CO 192 L		A	AX 00 00	AX C1 22
h			in Ax		
ror ax,1	07103: D1 209 =	0103	Rotate Right	ı H L	H L
Í	07104: C8 200 ^L		Shift without	AX C1 22	AX 60 23
	ARTAE KA TAE II		carry content	w 10.1 12.2	
			•		
			of Ax Store in		
			Ax		
mov	07105: B9 185	0105	Store segment	CX 00 10	CX 30 00
cx,3000h	07106: 00 000 NULL 07107: 30 048 0		address in Cx		ox jou ju
mov ds,cx		0108	copy content of	DS 0700	DS 3000
	07109: D9 217 J		cx into ds		
mov	0710A: BF 191 7	010A	Store offset	DI 0000	DI 0130
di,0130h	0710B: CO 192 L		address in di	2.	0100
41,015011	0710C: 00 000 NULL		uuui (55 III ui		
mov	07110: 89 137 ë	0110	Store contentof	30130: 00 000 NULL 30131: 00 000 NULL	30130: 21 033 ! 30131: 60 096
[di],ax	07111: 05 005 ♠		Ax At 30130h		
L 3/***					

CONCLUSION:

In conclusion, this experiment successfully demonstrated various assembly language operations, including arithmetic, logical, shift, and rotate functions on 32 -bit numbers. Each operation's results were accurately stored in specific memory locations, showcasing efficient handling of data and manipulation of registers using instructions like ADD, SUB, MUL, DIV, and logical shifts.

Date:19-09-24

EXPERIMENT NO. 7

AIM: Create an array. Perform addition of all even numbers from array and save answer in one variable.

ALGORITHM:

Step:1 Set Data Segment:

Load the value 2000h into the AX register.

Set the DS register to AX, initializing the data segment to 2000h.

Step:2 Initialize Index and Counter:

Set SI to 50h, pointing to the starting address of the array in memory (DS:50h).

Set CL to 5h, initializing the loop counter to 5, which represents the number of elements in

Step:3 Initialize Sum:

the

Set AL to 0, initializing the accumulator to store the sum of even numbers.

Step:4 Loop through the Array:

Repeat the following steps 5 times (controlled by CL).

Step:5 Inside the Loop:

Load the byte at the address [SI] (the current element in the array) into BL.

Use the test bl, 01h instruction to check if the least significant bit (LSB) is 0 (indicating an even number).

If the number is odd (LSB is 1), skip the addition step by jumping to L1.

If the number is even, add it to the accumulator (AL) using add al, bl.

Increment SI to point to the next element in the array.

Use the loop instruction to decrease CL and repeat the loop until CL becomes 0.

Step:6 End the Program:

When all 5 elements have been processed, return control to the caller using the ret instruction.

org 100h

mov ax,2000h mov ds,ax mov si,50h mov cl,5h

mov al,0h

L2:

mov bl,[si] test bl,01h JNZ L1

add al,bl

L1: inc si

loop L2

ret

Instruction	Screenshot	Value in	Operation	(Before execution	(After execution
				f	
in	of the	Instructio	executed by	instruction) Content of	of instruction)
Assembly	Instruction	n Pointer	instruction	to be affected	Content of
Language	in program memory			Registers/Memory locations and flags	affected Registers/Memor y locations and flags
mov ax,2000h	07100: B8 184 7 07101: 00 000 NULL 07102: 20 032 SPA	0100	mov ax,2000h loads the value in ax	AX 00 00	AX 20 00
mov ds,ax	07103: 8E 142 ä 07104: D8 216 ‡	IP 6163	Mov da,ax loads the value from ax,ds	DS 0700	DS 2000
mov si,50h	07105: BE 190 ± 07106: 50 080 P 07107: 00 000 NULL	01 05	Mov si,50h set the si to 50h	SI 0000	SI 0050
mov cl,5h	07108: B1 177 ₪ 07109: 05 005 ☆	01 08	Mov cl,5h set the cl to the 5h	CX 00 19	CX 00 05

mov al,0h	0710A: BO 176 N 0710B: 00 000 NUL	IP	01 0A	Mov al,0h set the al to the 0h	AX	H 20	00	AX	H 20	00
mov bl,[si]	0710C: 8A 138 è 0710D: 1C 028 ∟	IP	01 OC	Mov bl,[si] set the bl to the si	BX	00	00	BX	00	04
test bl,01h	0710E: F6 246 ÷ 0710F: C3 195 } 07110: 01 001 ©	IP	01 0E	The instruction `test bl, 01h` checks if the least significant bit of the `BL` register is 1 or 0.	BX	00	94	BX	00	94
JNZ L1	07111: 75 117 u 07112: 02 002 9	IP	0111	The instruction 'JNZ L1' means "Jump to the label 'L1' if the result of the previous operation is not zero						
add al,bl	07113: 02 002 0 07114: C3 195	IP	0113	Add al,bl means that al + bl						
inc si	07115: 46 070	IP	0115	Increase Si by 1	SI	005	0	SI	005	1
LOOP L2	07116: E2 226 [07117: F4 244	IP	0116	LOOP traversal for the cl times here 5						
inc si	07115: 46 070 F	IP	0115	2nd traversal of the loop	SI	005	1	SI	0052	2
add al,bl	07113: 02 002 8 07114: C3 195 }	IP	0113		AX	H 20	04	AX	20	96
Inc si	07115: 46 070 F	IP	0115	3 rd traversal of the loop	SI	0052	!	SI	0053	
Inc si	07115: 46 070 F	IP	0115	4th traversal of the loop	SI	0052		SI	0054	
Inc si	07115: 46 070 F	IP	0115	5th traversal of the loop	SI	0054		SI	0055	

CONCLUSION:

The given assembly code sums all the even numbers from a memory array of 5 elements. It checks each number using a bitwise test to determine if it's even, and adds even values to the `AL` register. The loop iterates over each element in the array, and the sum of all even numbers is stored in the `AL` register when the program completes. This demonstrates basic use of looping, conditional checks, and arithmetic in 8086 assembly language.

Page No:

Date:

EXPERIMENT NO. 8

AIM: Find out whether the given string is palindrome or not and print appropriate message. Don't use procedure.

ALGORITHM:

Input the String:

Load the string into memory.

Initialize Pointers:

• Set two pointers: one at the beginning of the string and the other at the end of the string.

Compare Characters:

- Compare the character at the start with the character at the end.
- If the characters match, move the start pointer forward and the end pointer backward. Repeat Until Middle:
- Repeat the comparison until the two pointers meet in the middle of the string (or cross each other).

Palindrome Check:

- If all characters match, the string is a palindrome.
- If any character does not match, the string is not a palindrome.

Output the Result:

• Print whether the string is a palindrome or not.

CSE-DEPSTAR

1. Palindrome:

CODE:

```
no e prac
org Iøøh
n1:
s db abcba'
s size
dB øDh.øAh.'$'
start:
; printin actual s tring...
offset s:
int 21b
lea d i. s
add s i. s —size
dec si; point to
nou cx.s_size
Je IS —palindrome
shr c x. 1; divide
nov
nou b 1. t sil
cnp a 1. bl
last char'
•sin le char
is palindrome!
Sne not_palindeone
dec si
loop next—char
s _pa indi•one
nov
nou dx.offset
int 21b
.jnp stop
no t —pa 1 in drone:
nov ah.9
nou dx offset
int 21iI
; wait for any
ret
nsgl db "this
nsg2 db "this
nsgl
nsg2
key preg:
is palidrone!$"
is not pal i drone' $ "
```

Instruction in Assembly	Screenshot of the Instruction in	Value Instruc	in	Operation executed by	(Before execution of			(Afte		of
Language	program	Pointe		instruction	instruction)			ition uction		
	memory				Content of to be			Cont		of
					affected Registers/Memo			affec Regis		Mρ
					ry locations and			_		
					flags			and f		
jmp start	07100: EB 235 6 07101: 08 008 BACK	IP [0100	Direct jump at start		-			-	
mov ah,9	0710A: B4 180 { 0710B: 09 009 TAB	IP	01 0A	Store the value in ah	AX	00	00	AX	89	00
mov dx,offset s;	0710C: BA 186 0710D: 02 002 5 0710E: 01 001 5	J.	01 0C	Offset address of string s store in dx.	DX	00	00	DX	01	02
Int 21h	0710F: CD 205 = 07110: 21 033 !	IP [01 0F	Int 21h use for see the output on screen		_		abo	ba	
lea di,s	07111: BF 191 7 07112: 02 002 5 07113: 01 001 ©	IP	0111	Load effective Address of string s store in di.	DI	88	00	DI	01	02
mov si,di	07114: 8B 139 ï 07115: F7 247 ≈	IP [0114	Copy the data of di and store in si.	SI	06	100	SI	91	02
add si,s_size	07116: 83 131 â 07117: C6 198 ⊧ 07118: 05 005 ♠	IP	0116	add the size of a string in si.	SI	61	02	SI	61	07
dec si	07119: 4E 078 N	IP [8119	Decrement in si.	SI	01	07	SI	61	86
mov cx,s_size	0711A: B9 185 0711B: 05 005 & 0711C: 00 000 NULI	IP	011A	The size of string is store in cx.	CX	00	6F	CX	00	05
cmp cx,1	0711D: 83 131 â 0711E: F9 249 · 0711F: 01 001 ©	IP [011D	Compare the contents of general purpose register with 1.	CX	00	05	CX	00	05
je is_palindrome	07120: 74 116 t 07121: 0E 014 N	IP [0120	Jump if equals the previous comparison		-			-	
shr cx,1	07122: D1 209 π 07123: E9 233 θ		0122	Right shift of the value of cx by 1.	CX	00	05	c×	00	02
mov al,[di]	07124: 8A 138 è 07125: 05 005 ☆		0124	The data of [di] is stored in al.	AX	09	24	$A \times$	09	61
mov bl,[si]	07126: 8A 138 è 07127: 1C 028 ∟	IP [0126	The data of [si] is stored in bl.	BX	00	99	BX	00	61
cmp al,bl	07128: 3A 058: 07129: C3 195	IP [0128	Compare the	$\mathbb{A}\!\!\times\!$	09	61	$A\!\!\times\!$	69	61
				contents of al and bl.	BX	00	61	В×	00	61

CSE202-Microprocessor and Computer Organization **CSE-DEPSTAR** 0712A: 75 117 u 0712B: 0D 013 CRET Jump if not equals ine IΡ 012A not_palindrome to each other. inc di Increment the data 0712C: 47 071 G 0103 IΡ DΙ 012C DΙ 0102 of di. dec si Decrement the 0712D: 4E 078 N IΡ SI 0106 01 05 012D SI data of si. Execute the loop next char IΡ 012E next_char loop. The data of [di] is mov al,[di] ΙP 0124 99 69 62 61 ΑX ΔX stored in al. The data of [si] is mov bl,[si] IΡ 0126 00 62 BΧ 00 61 BXstored in bl. cmp al,bl Compare the IΡ 0128 09 62 09 62 ΑX ΑX contents of al and BΧ 00 62 BΧ 00 62 bl 0712A: 75 117 0712B: 0D 013 Jump if not equals ine IΡ 012A not_palindrome to each other. Increment the data inc di DΙ 0712C: 47 071 G 0103 IΡ 012C DΙ 0104 of di. Decrement the dec si 0712D: 4E 078 N IΡ 012D SI 0105 SI 0104 data of si. loop next_char Execute the IΡ 012E next_char loop. mov ah.9 Value store in ah. IΡ 0130 99 62 09 62 ΑX ΑX Offset address of mov dx,offset IΡ 0132 01 DΧ 01 02 DΧ 45 msg1 is store in dx msg1 abcba this is palidrome! Output show on int 21h IΡ 0135 screen. Direct jump at imp stop IΡ 0137 stop 07140: B4 180 -07141: 00 000 NULL 0 store in ah mov ah,0 IΡ 0140 00 24 24 ΑX 99 ΑX register. F41C0: FF 255 RES F41C1: FF 255 RES abcba this is palidrome! int 16h Output show on IΡ 01C0 screen.

CSE-DEPSTAR

2. Not Palindrome:

```
CODE:
e prac
1ØØh
J n Start
01
s db '23CSØ24'
s size
db ØDh.ØAh.'$'
start:
; printin actual string...
nou dx.offset s:
ine 2ih
lea dies
nov s i.di
add s i. s —size
dec si; point to last char!
nov cx.s_size
cnp c x. 1
; sin Char
shr c x. 1; divide by 2 V
next—char:
nou al. Idi)
ne not palindrone
nC di
dec si
loop next _ehae
is —pa 1 ind rone
nou ax. offset nagi
int 2ih
n ot -pa indrone:
nou dx.offset
ine 2ih
stop:
;uait for any
nov ah.Ø
int
ret
nsg2 db
"this
"this
g2
key preg:
is not pal idronet$...
```

CSE-DEPSTAR **STEP-BY-STEP EXECUTION:**

Instruction in Assembly Language	Screenshot of the Instruction in program memory	Instruction ex Pointer in		instruction	Conto affect Regis	ntion of action) ent of to be ted atters/Memo	(After execution of instruction) Content of affected Registers/Me mory locations and flags			
jmp start	07100: EB 235 6 07101: 0A 010 NEWL	IP	91 99	Direct jump at start		-		-		
mov ah,9	0710C: B4 180 0710D: 09 009 TAB	IP	91 9C	Store the value in ah	AX	00 00	AX	89 00		
mov dx,offset s;	0710E: BA 186 0710F: 02 002 ♥ 07110: 01 001 ⊕	IP	01 0E	Offset address of string s store in dx.	DX	00 00	DX	01 02		
Int 21h	07111: CD 205 = 07112: 21 033 !	IP	8111	Int 21h use for see the output on screen		-	230	OCS005		
lea di,s	07113: BF 191 7 07114: 02 002 © 07115: 01 001 ©	IP	0113	Load effective Address of string s store in di.	DI	9999	DI	0102		
mov si,di	07116: 8B 139 ï 07117: F7 247 ≈	IP	0116	Copy the data of di and store in si.	SI	0000	SI	01 02		
add si,s_size	07118: 83 131 â 07119: C6 198 } 0711A: 07 007 BEEP	IP	0118	add the size of a string in si.	SI	01 02	SI	01 09		
dec si	0711B: 4E 078 N	IP	011B	Decrement in si.	SI	01 09	SI	01 08		
mov cx,s_size	0711C: B9 185 0711D: 07 007 BEEP 0711E: 00 000 NULL	IP	011C	The size of string is store in cx.	CX	00 71	$^{\circ}$	00 07		
cmp cx,1	0711F: 83 131 â 07120: F9 249 · 07121: 01 001 ©	IP	011F	Compare the contents of general purpose register with 1.	CX	00 07	CX	00 05		
je is_palindrome	07122: 74 116 t 07123: 0E 014 Л	IP	0122	Jump if equals the previous comparison		-		-		
shr cx,1	07124: D1 209 τ 07125: E9 233 θ	IP	0124	Right shift of the value of cx by 1.	CX	00 07	$\mathbb{C}\!\!\times\!$	00 03		
mov al,[di]	07126: 8A 138 è 07127: 05 005 ♠	IP	0126	The data of [di] is stored in al.	$A \times$	09 24	AX	09 32		
mov bl,[si]	07128: 8A 138 è 07129: 1C 028 ⊏	IP	0128	The data of [si] is stored in bl.	BX	00 00	BX	00 34		
cmp al,bl	0712A: 3A 058: 0712B: C3 195	IP	012A	Compare the contents of al and bl.	AX BX		AX BX	09 32 00 34		
jne not_palindrome	0712C: 75 117 u 0712D: 0D 013 CRET	IP	012C	Jump if not equals to each other.		=		-		

CSE-DEPSTAR	CSE202-Microprocessor and Computer Organization									
mov ah,9	0713B: B4 180 0713C: 09 009 TAB	IP	013B	Value store in ah.	AX 09 32	AX 89 32				
mov dx,offset msg2	0713D: BA 186 0713E: 5A 090 Z 0713F: 01 001 ©	IP	013D	Offset address of msg1 is store in dx	DX 01 02	DX 01 5A				
int 21h	07140: CD 205 = 07141: 21 033 !	IP	0140	Output show on screen.	-	23CSO24 this is not palidrome!				
jmp stop	07137: EB 235 δ 07138: 07 007 BEEP	IP	0137	Direct jump at stop	-	-				
mov ah,0	07142: B4 180 07143: 00 000 NULL	IP	0142	0 store in ah register.	AX 09 24	AX				
int 16h	F41C0: FF 255 RES F41C1: FF 255 RES	IP	0100	Output show on screen.	-	23CSD24 this is not palidrome!				

3. Single Character:

```
CODE:
o Pg I guh
startt
; print in S
int 21b
lea d i.'
add
dee; po int to char'
je is_palindrone •sin le char
shr cx.l :divide Ky 2!
next _char:
nov
nov b I.
cop
dec
loop next—char
is pal in drone'
nou ax. offset nsgt
int 21 h
*top
not _pa 1 indrone:
ax. of f set
ine 21b
top:
; "ait for any
n sg2
key pres:
int 16b
nsg2 db
"t his
Is
pal
```

Instruction in Assembly Language	Screenshot of the Instruction in program memory	Point	uction	instruction	affect Regis	tion oction ent of ed ters/		affect	ition uction ent ted sters/l	of Me
jmp start	07100: EB 235 δ 07101: 0A 010 NEWL	IP	0100	Direct jump at start		-			-	
mov ah,9	07106: B4 180 07107: 09 009 TAB	IP	01 06	Store the value in ah	AX	00	99	AX	09	00
mov dx,offset s;	07108: BA 186 07109: 02 002 € 0710A: 01 001 ⊕	IP	01 08	Offset address of string s store in dx.	DX	99	00	DX	01	02
Int 21h	0710B: CD 205 = 0710C: 21 033 !	IP	01 0B	Int 21h use for see the output on screen		-		a		
lea di,s	0710D: BF 191 7 0710E: 02 002 8 0710F: 01 001 ©	IP	01 0D	Load effective Address of string s store in di.	DI	6	000	DI	61	02
mov si,di	07110: 8B 139 ï 07111: F7 247 ≈	IP	0110	Copy the data of di and store in si.	SI	0	000	SI	01	02
add si,s_size	07112: 83 131 â 07113: C6 198 ⊧ 07114: 01 001 ⊕	IP	0112	add the size of a string in si.	SI	61	02	SI	61	03
dec si	07115: 4E 078 N	IP	0115	Decrement in si.	SI	01	03	SI	01	02
mov cx,s_size	07116: B9 185 07117: 01 001 © 07118: 00 000 NULL	IP	0116	The size of string is store in cx.	cx	00	6B	CX	00	01
cmp cx,1	07119: 83 131 â 0711A: F9 249 · 0711B: 01 001 ©	IP	8119	Compare the contents of general purpose register with 1.	CX	00	01	CX	00	91
je is_palindrome	0711C: 74 116 t 0711D: 0E 014 ∏	IP	011C	Jump if equals the previous comparison		-			-	
mov ah,9	0712C: B4 180 + 0712D: 09 009 TAB	IP	012C	Value store in ah.	AX	09	24	AX	09	24
mov dx,offset msg2	0712E: BA 186 0712F: 41 065 A 07130: 01 001 ©	IP	012E	Offset address of msg1 is store in dx	DX	01	02	DX	01	41
int 21h	07131: CD 205 = 07132: 21 033 !	IP	0131	Output show on screen.		=		a this i	is pali	drome!
jmp stop	07133: EB 235 6 07134: 07 007 BEEP	IP	0133	Direct jump at stop		-			-	
mov ah,0	0713C: B4 180 + 0713D: 00 000 NULL	IP	013C	0 store in ah register.	AX	09	24	ΑX	00	24
int 16h	0713E: CD 205 = 0713F: 16 022 =	IP	013E	Output show on screen.		-		a this	is pal:	idrome!

CONCLUSION:

The code checks if the string "abcba" is a palindrome by comparing characters from both ends inward. If all characters match, it prints a message stating that the string is a palindrome. If a mismatch is found, it prints a message indicating the string is not a palindrome.

Date:

EXPERIMENT NO. 9

AIM: Write an assembly code to evaluate the answer of below given series and store the answer in the ANS variable. Program should have only one procedure to compute the factorial of a number. Series: 1! -2+3!-4+5!-6+7!-8

ALGORITHM:

Setup Data and Initialize Variables:

- Initialize data segment (ds) using the mov instruction.
- Set up cx to start at 0001h (11 in decimal).

Loop through Numbers:

• Start a loop (next) that iterates over numbers from cx = 11 down to 1.

Check if Even or Odd:

- Use the division instruction (div) to divide the current number by 2.
- If the remainder (ah) is 1, the number is odd, so call the factorial_fun function.
- If the number is even, skip the factorial calculation.

Factorial Function:

- If the number is odd, the factorial_fun procedure calculates the factorial of the number stored in cx:
 - Start with 1 and multiply it by cx, decrementing cx by 1 each time, until cx reaches 1.
 - Add the result to the answer (ans).

Update the Result:

- For even numbers, subtract the number from the ans.
- For odd numbers, the factorial result is added to the ans.

Repeat the Process:

- After each iteration, decrement cx and repeat the loop until cx becomes zero.
 End Program:
- Once the loop is complete, the program halts (hlt).

CODE:

mode pract ica

, 23CS024

. data

I dup O

ans dw

1 dup 2

two db

tcx dw

I dup 2

code

CSE-DEPSTAR

@dat a

mov ax,

mov ds,ax

0008h

mov cx,

OOOlh

mov ax,

next:

mov dx,cx

mov ax, dx

div two

cmp ah,01h

j ne for_even

mov tcx, cx

call factorial_fun

mov cx,tcx

for_even:

sub ans, dx

I oop next

hlt

factorial fun proc

mov ax,OOj1h

lb:

mul cx

I oop lb

add ans, ax

ret

factorial_fun endp

Instructio	Screenshot of the	Value in	Operation	(Before execution	(After execution		
n in	Instruction in	Instruction	executed by	of instruction)	of instruction)		
Assembly	program memory	Pointer	instruction	Content of to be	Content of		
Language				affected	affected		
				Registers/Memor	Registers/Memo		
				y locations and	ry locations and		
				flags	flags		
mov	07110: B8 184 ₹ 07111: 10 016 ►	IP 0000	Segment address	AX 00 00	AX 07 10		
ax,@data	07112: 07 007 BEEP		of data is move		A.		
			into the ax.				

CSE-DEPSTAI				CSE202-Micro	process	or an	d Compu	ter Orga	anizat	ion
mov ds,ax	07113: 8E 142 Ä 07114: D8 216 ‡	IP	0003	The value of ax is store into data segment.	DS	07	700	DS	67	710
mov cx,0008h	07115: B9 185 07116: 08 008 BACK 07117: 00 000 NULL	IP	0005	0008h value store in cx.	CX	00	46	CX	00	08
mov ax,0001h	07118: B8 184 7 07119: 01 001 © 0711A: 00 000 NULL	IP	0008	0001h value store in ax.	AX	07	10	AX	00	01
mov dx,cx	0711B: 8B 139 ï 0711C: D1 209 T	IP	000B	The value of cx is store in dx.	DX	00	00	DX	00	08
mov ax,dx	0711D: 8B 139 Ï 0711E: C2 194 T	IP	000D	The value of dx is store in ax.	AX	00	01	AX	00	08
div two	0711F: F6 246 ÷ 07120: 36 054 6 07121: 02 002 © 07122: 00 000 NULL	IP	000F	Division of ax and two(0002h).	AX	00	08	AX	00	04
cmp ah,01h	07123: 80 128 C 07124: FC 252 07125: 01 001 ©	IP	0013	Higher bytes of ax compare with 01h.	AX	99	94	It is n	ot sar	ne data.
jne for_even	07126: 75 117 u 07127: 0B 011 &	IP	8816	Jump if previous statement is false.		-		Direct jump at for_even		
sub ans,dx	07133: 29 041) 07134: 16 022 — 07135: 00 000 NULL 07136: 00 000 NULL	IP	0023	Substraction of ans and dx and it's store in ans.			000 NULL 000 NULL			248 ° 255 RES
loop next	07137: E2 226 Г 07138: E2 226 Г	IP	0027	Repeat next loop.		-			-	
mov dx,cx	0711B: 8B 139 ï 0711C: D1 209 T	IP	000B	The value of cx is store in dx.	DX	00	02	DX	00	01
mov ax,dx	0711D: 8B 139 ï 0711E: C2 194 _T	IP	000D	The value of dx is store in ax.	AX	00	01	$A \times$	00	01
div two	0711F: F6 246 ÷ 07120: 36 054 6 07121: 02 002 © 07122: 00 000 NULL	IP	000F	Division of ax and two(0002h).	$A \times$	00	01	AX	91	00
cmp ah,01h	07123: 80 128 C 07124: FC 252 0 07125: 01 001 ©	IP	9913	Higher bytes of ax compare with 01h.	AX	01	99	It is s	ame d	lata.
jne for_even	07126: 75 117 u 07127: 0B 011 &	IP	0016	Jump if previous statement is false.	954.90	-	-		-	
mov tcx,cx	07128: 89 137 ë 07129: 0E 014 ∏ 0712A: 03 003 ♥ 0712B: 00 000 NULL	IP	0018	The value of cx store in tcx.	07103	: 03 E	000 NULL	07103 07104	1: 01 1: 00	001 ⊖ 000 NULL
call factorial_fun	0712C: E8 232 ፬ 0712D: 0B 011 ♂ 0712E: 00 000 NULL	IP	001C	Call factorial function.					-	_
mov ax,0001h	0713A: B8 184 7 0713B: 01 001 © 0713C: 00 000 NULL	IP	002A	0001h store in ax.	AX	01	00	$\mathbb{A} \times$	00	01
mul cx	0713D: F7 247 ≈ 0713E: E1 225 β	IP	002D	Multiplication of ax and cx. it's store in ax.	AX	00	91	AX	00	01
loop lb	0713F: E2 226 Г 07140: FC 252 "	IP	002F	Decrease the value of cx and Repeat loop lb if	CX	00	01	CX	00	99

CSE-DEPSTAR CSE202-Microprocessor and Computer Organization

add ans,ax	07141: 01 001 © 07142: 06 006 ± 07143: 00 000 NULL 07144: 00 000 NULL	IP	0031	cx is non-zero. Addition of ans and ax.value is store in ans.	07100: 1A 026 → 07101: 14 020 ¶	07100: 1B 027 + 07101: 14 020 ¶
ret	-	IP	0035	-	-	0710:0000 141B
mov cx,tcx	0712F: 8B 139 ï 07130: 0E 014 ∏ 07131: 03 003 ♥ 07132: 00 000 NULL	IP	001F	The value of tcx is store in cx.	CX 00 00	CX 00 01
sub ans,dx	07133: 29 041 > 07134: 16 022 = 07135: 00 000 NULL 07136: 00 000 NULL	IP	0023	Substraction of ans and dx and it's store in ans.	07100: 1B 027 ← 07101: 14 020 ¶	07100: 1B 027 ← 07101: 14 020 ¶
loop next	07137: E2 226 Г 07138: E2 226 Г	IP	0027	Repeat next loop.if cx is non-zero.	-	-
hlt	07139: F4 244 Γ	IP	0029	Stop the code	-	the emulator is halted.

CONCLUSION:

This algorithm loops through numbers from 11 down to 1, checking if each number is even or odd. For odd numbers, it calculates the factorial and adds it to the answer (ans). For even numbers, it subtracts the number from ans. The result is stored in ans, which is updated after each iteration.

Date:

EXPERIMENT NO. 10

AIM: Write a program which perform multiplication using booth algorithm

ALGORITHM:

Input Multiplicand and Multiplier:

- Take input for the number of bits (bits), the multiplicand (m), and the multiplier (r).
- Check if the input numbers can be represented within the given number of bits.

Convert to Binary:

• Convert the multiplicand (m) and multiplier (r) to their binary representations using the decToBin() function.

Calculate Two's Complement:

• Compute the two's complement of the multiplicand to use for subtraction by inverting the bits and adding 1.

Initialize Registers:

- Initialize A (accumulator) to all zeros.
- Initialize Q1 to zero (used in Booth's algorithm).
- Initialize Q with the binary representation of the multiplier.

Perform Booth's Multiplication:

- Loop for bits iterations:
 - o Check the last bit of Q and Q1.
 - o If Q0 is 1 and Q1 is 0, subtract M (i.e., A = A M).
 - o If O0 is 0 and O1 is 1, add M (i.e., A = A + M).
 - Perform an arithmetic right shift on A, Q, and Q1.

Output Binary Result:

• After all iterations, concatenate A and Q to form the final binary result.

Convert Binary to Decimal:

- If the result is negative, convert the two's complement to a positive number and add a negative sign.
- Print the decimal result.

End Program:

• Output the result in both binary and decimal formats and exit the program.

Page No:

CSE-DEPSTAR CODE:

```
#include <iostream>
#include <string>
#include <cmath>
using namespace std;
// Function to convert decimal to binary
string decToBin(int num, int bits) {
string bin = "";
for (int i = bits - 1; i >= 0; i--) {
bin += (num & (1 << i)) ? '1' : '0'; // Check each bit from left to right
return bin;
// Function to calculate 2's complement of a binary string
string twosComp(string bin) {
// Invert the bits (1's complement)
for (int i = 0; i < bin.size(); i++) {
bin[i] = (bin[i] == '1') ? '0' : '1'; // Flip bits (1 -> 0, 0 -> 1)
// Add 1 to get 2's complement
int carry = 1;
for (int i = bin.size() - 1; i >= 0; i--) 
if (bin[i] == '0' \&\& carry == 1) {
bin[i] = '1';
carry = 0; // No more carry, stop here
else if (bin[i] == '1' && carry == 1) {
bin[i] = '0'; // Still a carry, continue
return bin;
// Function to add two binary numbers
string binAdd(string a, string b) {
string result = "";
int carry = 0;
for (int i = a.size() - 1; i >= 0; i--) {
int bitA = a[i] - '0'; // Convert char to int
int bitB = b[i] - '0';
int sum = bitA + bitB + carry;
result = char((sum \% 2) + '0') + result; // Add the sum bit to result
carry = sum / 2; // Calculate carry for next iteration
```

```
CSE-DEPSTAR
                                                                     CSE202-Microprocessor and Computer Organization
return result;
// Function to perform arithmetic right shift on A and O
void rightShift(string &A, string &Q, char &Q1) {
Q1 = Q.back(); // Store the last bit of Q (Q0)
Q = A.back() + Q.substr(0, Q.size() - 1); // Shift Q by 1 bit
A = A[0] + A.substr(0, A.size() - 1); // Shift A by 1 bit (preserve sign bit)
// Booth's Algorithm implementation
string boothMult(int m, int r, int bits) {
// Convert multiplicand (M) and multiplier (Q) to binary
string M = decToBin(m, bits);
string O = decToBin(r, bits);
string A(bits, '0'); // Initialize A to 0 (same size as M and Q)
char Q1 = '0'; // Initialize Q-1 to 0
// Compute -M using 2's complement
string negM = twosComp(M);
// Print initial values
cout << "Initial values:\n";</pre>
cout << "M = " << M << "\n":
cout << "-M = " << negM << "\n";
cout << "Q = " << Q << "\n";
cout << "A = " << A << "\n";
// Perform the algorithm for 'bits' number of steps
for (int i = 0; i < bits; i++) {
// Check the last bit of Q (Q0) and Q-1
if (Q.back() == '1' && Q1 == '0') {
A = binAdd(A, negM); // A = A - M
ellet = (Q.back() == '0' & Q1 == '1') {
A = binAdd(A, M); // A = A + M
// Right shift A, Q, and Q1
rightShift(A, Q, Q1);
// Print step-by-step results
cout << "\nStep" << i + 1 << ":\n";
cout << "A = " << A << " Q = " << Q << " Q1 = " << Q1 << "\n";
return A + Q; // Return the concatenated result of A and Q
// Function to check if a number can be represented in 'bits' number of bits
bool isInRange(int num, int bits) {
int maxVal = pow(2, bits - 1) - 1; // Max value for signed integer
int minVal = -pow(2, bits - 1); // Min value for signed integer
return (num >= minVal && num <= maxVal);
```

```
CSE-DEPSTAR
                                                                      CSE202-Microprocessor and Computer Organization
int main() {
int bits, multiplicand, multiplier;
// Input number of bits and numbers
cout << "Enter size of binary number (number of bits): ";
cin >> bits:
cout << "Enter multiplicand: ";</pre>
cin >> multiplicand;
cout << "Enter multiplier: ";
cin >> multiplier;
// Check if the input numbers are within range for the given bit size
if (!isInRange(multiplicand, bits) || !isInRange(multiplier, bits)) {
cout << "Error: Numbers are out of range for " << bits << " bits." << endl;
return 1; // Exit if inputs are invalid
// Perform multiplication using Booth's Algorithm
string result = boothMult(multiplicand, multiplier, bits);
// Print binary result
cout << "\nBinary result: " << result << endl;</pre>
// Convert binary result to decimal
int decimalResult = (result[0] == '1') ? -stoi(twosComp(result), nullptr, 2) : stoi(result, nullptr, 2);
cout << "Decimal result: " << decimalResult << endl;</pre>
cout << "Program is created by DHRUV_LOKADIYA_23CS041" << endl;
return 0;
```

CSE-DEPSTAR OUTPUTS:

```
Enter size of binary number (number of bits): 8
Enter multiplicand: 25
Enter multiplier: 30
Initial values:
M = 00011001
-M = 11100111
0 = 00011110
A = 00000000
Step 1:
A = 000000000 Q = 00001111 Q1 = 0
Step 2:
A = 11110011 Q = 10000111 Q1 = 1
Step 3:
A = 11111001 \ Q = 11000011 \ Q1 = 1
Step 4:
A = 111111100 Q = 11100001 Q1 = 1
A = 111111110 Q = 01110000 Q1 = 1
Step 6:
A = 00001011 Q = 10111000 Q1 = 0
Step 7:
A = 00000101 Q = 11011100 Q1 = 0
Step 8:
A = 00000010 Q = 11101110 Q1 = 0
Binary result: 0000001011101110
Decimal result: 750
Program is created by DHRUV LOKADIYA-23CS041
Enter size of binary number (number of bits): 4
Enter multiplicand: 40
Enter multiplier: 41
Error: Numbers are out of range for 4 bits.
```

```
CSE-DEPSTAR
```

```
Enter size of binary number (number of bits): 6
Enter multiplicand: 15
Enter multiplier: 4
Initial values:
M = 001111
-M = 110001
Q = 000100
A = 000000
Step 1:
A = 000000 Q = 000010 Q1 = 0
Step 2:
A = 000000 Q = 000001 Q1 = 0
Step 3:
A = 111000 Q = 100000 Q1 = 1
Step 4:
A = 000011 Q = 110000 Q1 = 0
Step 5:
A = 000001 Q = 111000 Q1 = 0
Step 6:
A = 000000 Q = 111100 Q1 = 0
Binary result: 000000111100
Decimal result: 60
Program is created by DHRUV LOKADIYA-23CS041
```

CONCLUSION:

This code implements Booth's Algorithm for multiplying two integers using binary arithmetic. It efficiently handles signed numbers, converting them to binary, performing bitwise operations, and using two's complement for subtraction. The result is displayed in both binary and decimal formats. It also ensures the inputs fit within the specified bit size.

Date:

EXPERIMENT NO. 11

AIM: Write a program to convert a given number system to other number system.

TASK:

Ask user to choose one form given number system: Any radix

Ask user to enter a number (number can be float) for that number system.

Check that number entered for given number system is correct.

If not give the error

If correct then convert entered number in required radix number system

If you can show the conversion, then it's well and good.

NOTE:

Program can be implemented using any programming language.

You have to submit the program in assignment section of team after wards.

If you turn in Late Marks will be deducted.

SAMPLE CALCULATIONS OF NUMBER CONVERSIONS:

ALGORITHM:

Input Base and Number:

- Prompt the user to input the base of the number (fromBase) and the number in that base (numStr).
- Validate that the base is between 2 and 36.

Validate Number:

- For each character in the string numStr:
 - o If the character is a digit (0-9), check if its value is within the given base.
 - o If the character is a letter (A-Z or a-z), ensure it represents a valid digit for the base (e.g., for base 16, letters should only range from A to F).
 - o If the character is invalid, print an error and exit.

Convert to Decimal:

- Initialize decimalValue to 0.
- For each digit in numStr, convert it to its decimal equivalent using its base and the appropriate power of the base (taking into account both integer and fractional parts if present).
- Update decimalValue as the sum of all digit values times their respective powers of the base

Input Target Base:

- Prompt the user to input the target base (toBase) to which they want to convert the number.
- Validate that the base is between 2 and 36.

Convert Decimal to Target Base:

- Separate the integer and fractional parts of decimalValue.
- Convert the integer part by repeatedly dividing by toBase and storing the

Page No:

CSE-DEPSTAR

remainders (digits), which are later reversed to form the result.

• Convert the fractional part by multiplying it by toBase, extracting the integer part, and repeating this process for a fixed number of digits or until the fractional part becomes zero.

Output the Result:

- Print the converted number in the target base.
 - **Program End:**
- End the program after printing the result and exit gracefully.

CODE:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
// Function prototypes
int isValidNumber(const char *numStr, int base);
double convertToDecimal(const char *numStr, int base);
void convertFromDecimal(double num, int base);
// Main function
int main() {
int fromBase, toBase;
char numStr[50]:
printf("Enter the base of the number system (2-36): ");
scanf("%d", &fromBase);
if (fromBase < 2 \parallel fromBase > 36) {
printf("Invalid base!\n");
return 1;
}
printf("Enter the number: ");
scanf("%s", numStr);
if (!isValidNumber(numStr, fromBase)) {
printf("Invalid number for the given base!\n");
return 1;
}
double decimalValue = convertToDecimal(numStr, fromBase);
printf("Enter the base to convert to (2-36): ");
scanf("%d", &toBase);
if (toBase < 2 \parallel toBase > 36) {
printf("Invalid base!\n");
return 1:
}
```

```
printf("The number %s in base %d is ", numStr, fromBase);
convertFromDecimal(decimalValue, toBase);
printf(" in base %d.\n", toBase);
printf("\n This file prepared by 23CS024-Hardik Hadiya.");
return 0;
// Function to check if the number is valid for the given base
int is ValidNumber(const char *numStr, int base) {
for (int i = 0; i < strlen(numStr); i++) {
char c = numStr[i];
if (c \ge 0' \&\& c \le 9')
if (c - '0' >= base) return 0;
else if (c >= 'A' && c <= 'Z') 
if (c - 'A' + 10) = base) return 0;
} else if (c \ge 'a' \&\& c \le 'z') {
if (c - 'a' + 10 >= base) return 0;
\} else if (c == '.') {
// Allow decimal point in floating numbers
continue;
} else {
return 0:
return 1;
}
// Function to convert a number in a given base to decimal
double convertToDecimal(const char *numStr, int base) {
double decimal Value = 0;
int len = strlen(numStr);
int point = -1;
for (int i = 0; i < len; i++) {
if (numStr[i] == '.') {
point = i;
break;
}
int power = (point == -1) ? len -1 : point -1;
for (int i = 0; i < len; i++) {
char c = numStr[i];
if (c == '.') continue;
int digitValue = (c \ge 0' \&\& c \le 9')? c - 0':
(c \ge 'A' \&\& c \le 'Z') ? c - 'A' + 10 : c - 'a' + 10;
decimalValue += digitValue * pow(base, power--);
}
return decimalValue;
```

```
CSE-DEPSTAR
// Function to convert a decimal number to a given base
void convertFromDecimal(double num, int base) {
char result[50];
int index = 0;
long intPart = (long)num;
double fracPart = num - intPart:
// Convert integer part
while (intPart > 0) {
int digitValue = intPart % base;
result[index++] = (digitValue < 10) ? digitValue + '0' : digitValue + 'A' - 10;
intPart /= base;
result[index] = \0;
// Reverse the integer part of the result
for (int i = 0, j = index - 1; i < j; i++, j--) {
char temp = result[i];
result[i] = result[j];
result[j] = temp;
}
// Add fractional part
if (fracPart > 0) {
strcat(result, ".");
for (int i = 0; i < 10 && fracPart > 0; i++) {
fracPart *= base;
int digitValue = (int)fracPart;
fracPart -= digitValue;
char digitChar = (digitValue < 10) ? digitValue + '0' : digitValue + 'A' - 10;
result[strlen(result) + 1] = '\0';
result[strlen(result)] = digitChar;
}
```

printf("%s", result);

OUTPUTS:

1. Decimal to Binary:

```
Enter the base of the number system (2-36): 10 Enter the number: 24 Enter the base to convert to (2-36): 2 The number 24 in base 10 is 11000 in base 2.

This file prepared by 23CS024-Hardik Hadiya.
```

2. Decimal to Octal:

```
Enter the base of the number system (2-36): 10 Enter the number: 24 Enter the base to convert to (2-36): 8 The number 24 in base 10 is 30 in base 8.

This file prepared by 23CS024-Hardik Hadiya.
```

3. Decimal to Hexadecimal:

```
Enter the base of the number system (2-36): 10 Enter the number: 24 Enter the base to convert to (2-36): 16 The number 24 in base 10 is 18 in base 16.
```

This file prepared by 23CS024-Hardik Hadiya.

4. Binary to Decimal:

```
Enter the base of the number system (2-36): 2
Enter the number: 11000
Enter the base to convert to (2-36): 10
The number 11000 in base 2 is 24 in base 10.
```

This file prepared by 23CS024-Hardik Hadiya.

5. Base-5 to Base-3:

```
Enter the base of the number system (2-36): 5
Enter the number: 24
Enter the base to convert to (2-36): 3
The number 24 in base 5 is 112 in base 3.
This file prepared by 23CSO24-Hardik Hadiya.
```

CSE-DEPSTAR

CSE202-Microprocessor and Computer Organization

6. If ,we select 24 numeric value of Base-3,than give invalid number.

Enter the base of the number system (2-36): 3 Enter the number: 24 Invalid number for the given base!

CONCLUSION:

This C program converts numbers between different bases (from base 2 to base 36). It validates he input number, converts it to a decimal, and then re-converts the decimal to the desired target ase. The program can handle both integer and fractional parts of numbers, ensuring precise conversions across a wide range of bases.