Computer Organization And Architecture Lab Manual

Subject Code: CSE181402 Class: IV Semester(CSE)

Prepared By Mr. Biswajit Sarma Assistant Professor



Department of Computer Science & Engineering JORHAT ENGINEERING COLLEGE JORHAT : 785007, ASSAM

Vision of the Department

To become a prominent department of Computer Science and Engineering for producing quality human resources to meet the needs of the industry and society

Mission of the Department

1: To impart quality education through well-designed curriculum and academic facilities to meet the computing needs of the industry and society

2: To inculcate the spirit of creativity, team work, innovation, entrepreneurship and professional ethics among the students

3: To facilitate effective interactions to foster networking with alumni, industries, institutions of learning and research and other stake-holders

4: To promote research and continuous learning in the field of Computer Science and Engineering

OBJECTIVE: This lab complements the Computer Organization and Architecture course. Students will gain practical experience with designing and implementing concepts of micro processor systems using 8085 micro processor such as simple arithmetic operations, loop, pointer, counter, interrupt, interfacing.

Program Outcomes

PO1	Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
PO2	Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
PO3	Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4	Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.	
PO5	Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.	
PO6	The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal,health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.	
PO7	Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.	
PO8	Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.	
PO9	Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.	
PO10	O Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.	
PO11	Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.	
PO12		

Program Specific Outcomes

PSO1	1 Gain ability to employ modern computer languages, environments and platforms in creating innovative career paths	
PSO2	Achieve an ability to implement, test and maintain computer based system that fulfils the desired needs	

COMPUTER ORGANIZATION AND ARCHITECTURE LAB SYLLABUS (Practical Hours: 04, Credits: 00)

Implement the following programs using 8085 Simulator(gnusim8085 and 8085Compiler.jar) and 8085 Kits.

Exp. No.	List of Experiments	Page No.
1.	Write a program using 8085Microprocessor for Decimal, Hexadecimal addition and subtraction of two Numbers.	5-9
2.	Write a program using 8085 Microprocessor for addition and subtraction of two BCD numbers.	10-11
3.	To perform multiplication and division of two 8 bit numbers using 8085.	12-15
4.	To find the largest and smallest number in an array of data using 8085 instruction set.	16-18
5.	To write a program to arrange an array of data in ascending and descending order.	19-20
6.	Write an simple Interrupt service routine to understand interrupt.	21-26
7.	Interfacing a program to initiate 8251 and to check transmissionand reception of character	27-28

Experiment No:1

AIM: Write a program using 8085Microprocessor for Decimal, Hexadecimal addition and subtraction of two Numbers.

a) Write a program to add two hexadecimal & decimal numbers.

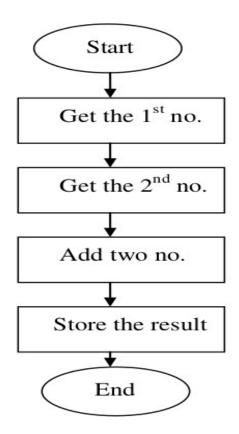
DESCRIPTION/ALGORITHM:-

Hexadecimal Addition : The program takes the content of 2009, adds it to 200B & stores the result back at 200C.

Steps:

- 1. Initialize HL Reg. pair with address where the first number is lying.
- 2. Store the number in accumulator.
- 3. Get the second number.
- 4. Add the two numbers and store the result in 200B.
- 5. Go back to Monitor

Let: (2009 H) = 80 H (200B H) = 15 H Result = 80 H + 15 H = 95 H (2009 H) A A B (200B H) A A + B A A - B (200C H) FLOWCHART : -



PROGRAM:-

 LXI H, 2009;
 Point 1 st no.

 MOV A, M;
 Load the acc.

 INX H;
 Adv Pointer

 ADD M;
 ADD 2 nd NO.

 INX H;
 Adv Pointer

 MOV M, A;
 Store Result

 RST 5;
 Foundation of the store

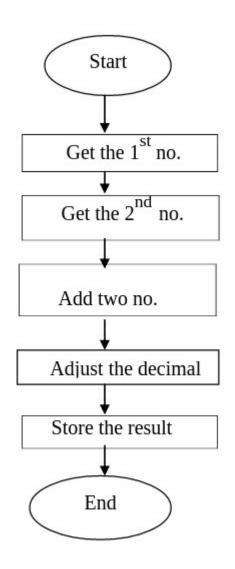
ila Dacat Accomplar Dabur	Hala	
ile Reset Assembler Debug		
	🗱 📎 🔅 🧶 🛑 🛑 🖕	
Image: Segisters Flag A OB S 0 BC OO OO Z 0 DE OO OO AC 0 HL O7 DB P 0 PSW OO OO C 0 PC 42 OC C 0 Decimal - Hex Convertion Hex 0 0 Decimal - Hex Conc C 0 Image: To Hex Image: To Dec Image: To Dec Image: To Dec Image: Update Port Value Image: To Dec Image: To Dec Image: To Dec Image: Update Port Value Image: To Dec Image: To Dec Image: To Dec Image: Update Port Value Image: To Dec Image: To Dec Image: To Dec Image: To Dec Image: Update Port Value Image: To Dec Image	Load me at 1 2 3 4 jmp start 5 5 6 ; data 7 8 9 ; code 10 start: nop 11 LXI H, 2009 12 MOV A, M 13 INX H 14 ADD M 15 INX H 16 MOV M, A 17 RST 5 18 19 hlt Line No A	2009 5 2010 6 2011 11 2012 0 2013 0 2014 0 2015 0 2016 0 2017 0 2018 0

Decimal Addition:

Steps:

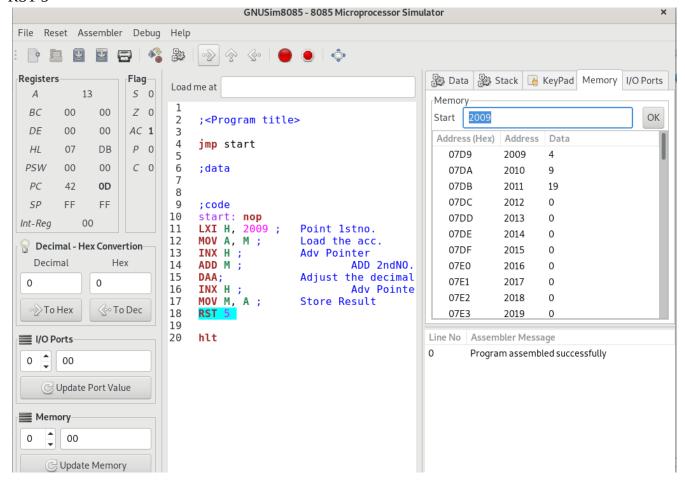
- 1. Initialize HL Reg. pair with address where the first number is lying.
- 2. Store the number in accumulator.
- 3. Get the second number.
- 4. Add the two numbers and store the result in 200B.
- 5. Go back to Monitor

FLOWCHART:-



PROGRAM:-

LXI H, 2009 ;	Point 1stno.
MOV A, M ;	Load the acc.
INX H;	Adv Pointer
ADD M;	ADD 2ndNO.
DAA;	Adjust the decimal
INX H;	Adv Pointer
MOV M, A;	Store Result
RST 5	



Question: Write a program using 8085Microprocessor for Decimal, Hexadecimal subtraction of two Numbers.

Experiment No:2

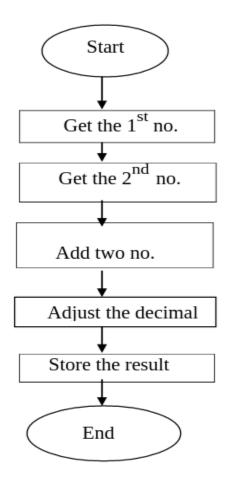
AIM: Write a program using 8085 Microprocessor for addition and subtraction of two BCD numbers.

DESCRIPTION/ALGORITHM:-

Steps:1. Initialize HL Reg. pair with address where the firstnumber is lying.

- 2. Store the number in accumulator.
- 3. Get the second number.
- 4. Add the two numbers and store the result in 200B.
- 5. Go back to Monitor

FLOWCHART:-



PROGRAM:-

LXI H, 2009 ;	Point 1stno.
MOV A, M ;	Load the acc.
INX H;	Adv Pointer
ADD M;	Addition IIND NO.
DAA;	Adjust the decimal
INX H;	Adv Pointer
MOV M, A;	Store Result
RST 5	
	GNUSim8085 - 8085 Microprocessor Simulator
File Reset Assembler Debug	g Help
: 🕒 🖪 🗳 🗗 🖓	🟅 🐉 🐵 🔗 🗇 😑 😐 💠
Registers	Load me at 🐉 Data 🐉 Stack 🐻 KeyPad Memory I/O Port
A 1D 5 0	1 Memory
BC 00 00 Z 0 DE 00 00 AC 1	2 ; <program title=""> Start 2009 OK</program>
HL 07 DB P 1	3 4 jmp start Address (Hex) Address Data
PSW 00 00 C 0	5 07D9 2009 14 6 ;data 07DA 2010 9
PC 42 OD	7 07DB 2011 29
SP FF FF	8 9 ; code 07DC 2012 0
Int-Reg 00	10 start: nop 07DD 2013 0 11 LXI H, 2009 ; Point 1stno. 07DF 2014 0
S Decimal - Hex Convertion	12 MOV A, M ; Load the acc.
Decimal Hex	13 INX H ; Adv Pointer 07DF 2015 0 14 ADD M ; Addition IIND NO. 07E0 2016 0
0 0	15 DAA; Adjust the decimal 07F1 2017 0
	16INX H ;Adv PointerOver 1000000000000000000000000000000000000
To Hex Or To Dec	18 RST 5 19 hlt
I/O Ports	Line No Assembler Message
0 🔹 00	0 Program assembled successfully
G Update Port Value	
Memory	
C Update Memory	

Question: Write a program using 8085 Microprocessor for subtraction of two BCD numbers.

Experiment No:3

AIM: To perform multiplication and division of two 8 bit numbers using 8085. DESCRIPTION/ALGORITHM:-

1) Start the program by loading HL register pair with address of memory location.

2) Move the data to a register (B register).

3) Get the second data and load into Accumulator.

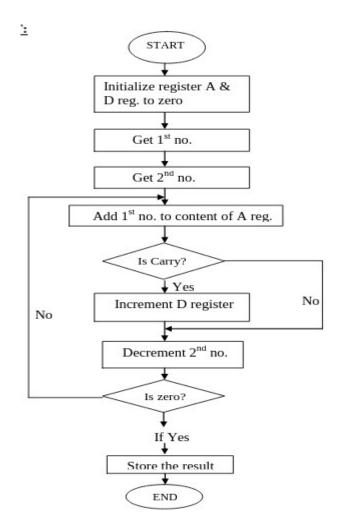
4) Add the two register contents.

5) Check for carry.

6) Increment the value of carry.

7) Check whether repeated addition is over and store the value of product and carryin memory location.

FLOWCHART:



PROGRAM:

MVI D, 00 ; Initialize register D to 00

MVI A, 00; Initialize Accumulator content to 00

LXI H, 4150; HL Points 4150

MOV B, M ; Get the first number in B -register

INX H; HL Points 4151

MOV C, M; Get the second number in C-reg.

LOOP: ADD B ; Add content of A -reg to register B.

JNC NEXT; Jump on no carry to NEXT.

INR D ; Increment content of register D

NEXT: DCR C; Decrement content of register C.

JNZ LOOP; Jump on no zero to address

STA 4152 ; Store the result in Memory

MOV A, D; Get the carry in Accumulator

STA 4153 ; Store the MSB of result in Memory

HLT; Terminate the program.

OBJECTIVE: -Write a program to perform multiplication of two 8 bit numbers using bit rotation method.

DESCRIPTION/ALGORITHM:-

1) Start the program by loading HL register pair with address of memory location.

2) Move the data to a register (Eregister).

3) Get the second data and load into Accumulator.

4) Add the two register contents.

5) Check for carry.

6) Increment the value of carry.

7) Check whether repeated addition is over and store the value of product and carryin memory location.

8) Terminate the program.

EXAMPLE :

Steps	Product	Multiplier	Comments
	B ₇ B ₆ B ₅ B ₄ B ₃ B ₂ B ₁ B ₀	CY B ₃ B ₂ B ₁ B	D
	000000000	0 0 1 0 1	Initial Stage
Step 1	000000000	0 1 0 1 0	Shift left by 1
	0 0 0 0 0 0 0 0	0 1 0 1 0	Don't add since CY=0
Step 2	0 0 0 0 0 0 0 0	1 0 1 0 0	Shift
-	0 0 0 0 1 1 0 0	1 0 1 0 0	Add multiplicand;CY=1
Step 3	0 0 0 1 1 0 0 0	0 1 0 0 0	Shift left by 1
-	0 0 0 1 1 0 0 0	0 1 0 0 0	Don't add since CY=0
Step 4	$0 \ 0 \ 1 \ 1 \ 0 \ 0 \ 0 \ 0$	1 0 0 0 0	Add multiplicand;CY=1

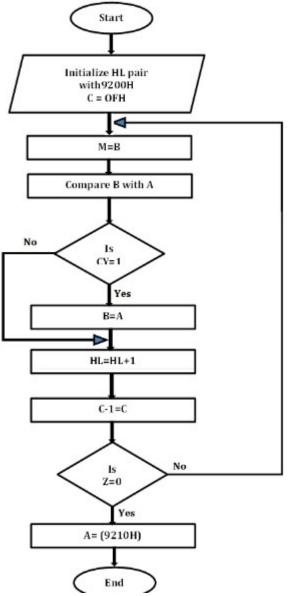
PROGRAM:

LXI H, 2200H ;Initialize the memory pointer MOV E , M;Get multiplicand MVI D, 00H ;Extend to 16 bits INX H;Increment memory pointer MOV A , M;Get Multiplier LXI H , 0000H ;Product = 0 MVI B, 08H;Initialize counter with count 8 LOOP: DAD H;Product = product X 2 RAL JNC XYZ ;Is carry from multiplier 1? DAD D ;Yes, product = product + multiplicand XYZ: DCR B ;Is counter = 0 JNZ LOOP; No, repeat SHLD 2202H ;Store the result HLT

Question: Perform division of two 8 bit numbers using 8085.

Experiment No:4 AIM: To find the largest and smallest number in an array of data using 8085 instruction set.

DESCRIPTION/ALGORITHM:-Write a program to find the largest number in a given array of 6 elements. The array is stored in memory from 9200H onwards. Store the result at the end of the array. **FLOWCHART:-**



PROCEDURE:-To find largest of given no. of a given string we compare all given no. one by one. Suppose given no. is 2, 4, 3, 1, 0 1^{st} we compare 2 & 4 (2 is in register A & 4 is in RegisterB). A < B so put B into (A) & Compare with next number i.e. 3 Here A > B so directly compare 4 with 1 then 0.

Program:

8	
LXI H,9200H;	Point to get array size
MOV C, M;	Get the size of array
INX H;	Point to actual array
MOV B, M;	Load the first number into B
DCR C;	Decrease C
LOOP: INX H;	Point to next location
MOV A, M;	Get the next number from memory to Acc
CMP B;	Compare Acc and B
JC SKIP;	if B >= A,then skip
MOV B, A;	If CY is 1, update B
SKIP: DCR C;	Decrease C
JNZ LOOP;	When count is not 0, go to LOOP
LXI H,9208H;	Point to destination address
MOV M, B;	Store the minimum number
HLT;	Terminate the program

	GNUSim8085 - 8085 Microprocessor Simulator	x
File Reset Assembler Debug	Help	
🕒 🖻 🖺 🖬 🖙 🗳	; 🎥 🗞 송 🕼 🛑 🔘 💠	
Registers Flag A 06 S 0 BC 06 00 Z 1 DE 00 05 AC 0 HL 92 08 P 1 PSW 00 00 C 0 PC 42 1A C 0 Decimal - Hex Convertion Decimal Hex 0 Decimal - Hex Go 0 0 0 Image: To Hex Image: To Dec Image: To Dec Image: To Dec Image: To Hex Image: To Dec Image: To Dec Image: To Dec Image: To Dec Image: To Dec Image: To Dec Image: To Dec Image: To Dec Image: To Dec Image: To Dec Image: To Dec Image: To Dec Image	Load me at 1 2 ; <program title=""> 3 4 jmp start 5 6 ;data 7 8 9 ;code 10 start: nop 11 LXI H,9200H; Point to get array size 12 MOV C, M; Get the size of array 13 INX H ; Point to actual array 14 MOV B, M; Load the first number into B 15 DCR C; Decrease C 16 LOOP: INX H; Point to next location 17 MOV A, M; Get the next number from memory to Acc 18 CMP B; Compare Acc and B 19 JC SKIP; if B ≯ A, then skip 20 MOV B, A; If CY is 1, update B 21 SKIP: DCR C; Decrease C 22 JNZ LOOP; When count is not 0, go to LOOP 23 LXI H,9208H; Point to destination address 24 MOV M, B; Store the minimum number 25 HLT; Terminate the program</program>	Data Stack KeyPad Memory I/O Ports Memory 9200h OK OK Address (Hex) Address Data OK 9200 37376 6 O 9201 37377 1 O 9202 37378 2 O 9203 37379 3 O 9204 37380 4 O 9205 37381 5 O 9208 37384 6 O 9209 37385 0 O 9208 37386 O O

Question: Find the smallest number in an array of data using 8085 instruction set.

Experiment No:5

AIM: To write a program to arrange an array of data in ascending and descending order.

Algorithm

- 1. Initialize HL pair as memory pointer
- 2. Get the count at 5000 into C register
- 3. Copy it in D register (for bubble sort (N-1) times required)
- 4. Get the first value in A register
- 5. Compare it with the value at next location
- 6. If they are out of order, exchange the contents of A register and Memory
- 7. Decrement D register content by 1
- 8. Repeat steps 5 and 7 till the value in D- register become zero
- 9. Decrement C register content by 1
- 10. Repeat steps 3 to 9 till the value in C register becomes zero

Program:

LXI H,5000h	;Set pointer for array
MOV C,M	;Load the Count
DCR C	;Decrement Count
REPEAT: MOV D,C	
LXI H,5001h	
LOOP: MOV A,M	;copy content of memory location to Accumulator
INX H	
CMP M	
JC SKIP	;jump to skip if carry generated
MOV B,M	;copy content of memory location to B - Register
MOV M,A	;copy content of Accumulator to memory location
DCX H	;Decrement content of HL pair of registers
MOV M,B	;copy content of B - Register to memory location
INX H	;Increment content of HL pair of registers

SKIP: DCR D	;Decrement content of Register - D
JNZ LOOP	;jump to loop if not equal to zero
DCR C	;Decrement count
JNZ REPEAT	;jump to repeat if not equal to zero
HLT	;Terminate Program

r.	GNUSim8085 - 8085 Microprocessor Simulator ×
File Reset Assembler Debu	g Help
· · • • • • • •	
Registers S 0 A 02 S 0 BC 01 00 Z 1 DE 00 05 AC 0 HL 50 02 P 1 PSW 00 00 C 0 PC 42 21 SP FF FF Int-Reg 00 0 U U U Decimal Hex 0 0 U U O 0 Se To Dec U U U Image: To Hex Se To Dec U U U	Load me at 1 2 ; <program title=""> 3 4 jmp start 5 ; data 7 ; code 9 ; code 10 start: nop 11 LXI H, 5000h ;Set pointer for array 20 MOV C,M ;Load the Count 13 DCR C ;Decrement Count 14 REPEAT: MOV D,C 15 LXI H, 5001h 16 LOOP: MOV A,M ;copy content of memory locat 17 INX H 18 CMP M 19 JC SKIP ;jump to skip if carry genera 20 MOV B,M ;copy content of memory locat 21 MOV M,A ;copy content of Accumulator 0 Program assembled surcessfully</program>
0 0 0 0 0 0 0 0 0 0 0 0 0 0	21MOV M,A;COPY content of Accumulator0Program assembled successfully22DCX H;Decrement content of HL pair0Program assembled successfully23MOV M,B;copy content of B - Register124INX H;Increment content of HL pair25SKIP: DCR D;Decrement content of Register26JNZ LOOP;jump to loop if not equal tc27DCR C;Decrement count28JNZ REPEAT;jump to repeat if not equal29HLT;Terminate Program3031

Question: write a program to arrange an array of data in descending order.

Experiment No:6

AIM: Write an simple Interrupt service routine to understand interrupt. For this Experiment Please download 8085Compiler.jar file (A new simulator) and run \$ java -jar 8085Compiler.jar

Instruction	Machine hex code	Interrupt Vector Address
RST 0	C7	0000H
RST 1	CF	0008H
RST 2	D7	0010H
RST 3	DF	0018H
RST 4	E7	0020H
RST 5	EF	0028H
RST 6	F7	0030H
RST 7	FF	0032H

Software interrupts and their vector addresses

Hardware interrupts of 8085

Interrupt	Interrupt vector	Maskable or non-	Edge or level	priority
	address	maskable	triggered	
TRAP	0024H	Non-makable	Level	1
RST 7.5	003CH	Maskable	Rising edge	2
RST 6.5	0034H	Maskable	Level	3
RST 5.5	002CH	Maskable	Level	4
INTR	Decided by hardware	Maskable	Level	5

Procesure: As mention in the above tables all the interrupts have their own interrupt address. Interrupt means we are doing something (Normal code) suddenly one special thing happens (Interrupt happens) and we need to perform the special task immideatly and then return back to the original work. Lets take one example:

Normal code: Register B is initialized to 0 and it is incremnting indefinately.

Interrupt code: When interrupt occurs Register A is assigned a value 05.

program:

EI;	Enabling interrupt
-----	--------------------

mvi b,00; Register B is initialized to 0

loop: inr b; incremening register B

jmp loop; for ever loop

hlt; stop the program

#org 003ch; RST 7.5 interrupt address, we need to write the interrupt service routine in the address 003ch

mvi a,05; Register A is set as 05

ret

Edit Tools Settings Simulation Subroutine View Load Sample Program Help Editor Assembler Registers Memory Devices * Address Labe Mnemonics Hexcode Bytes M-Cycles T-States • * Address Labe Mnemonics Hexcode Bytes M-Cycles T-States • * Modif MVI B,00 06 2 2 7 • Register B 000 0							8085	Sim	ulator								
Assembler Address Label Mnemonics Hexcode Bytes M-Cycles T-States ◊ 0001 MVI B,00 06 2 2 7 ◊ 0001 MVI B,00 06 2 2 7 ◊ 0001 MVI B,00 06 2 2 7 ◊ 0003 LOOP NR B 04 1 1 4 ◊ 0003 LOOP NR B 04 1 1 4 ◊ 0003 LOOP NR B 04 1 1 4 ◊ 0004 jMP LOOP C3 3 10 00 0 <td>ile Edit T</td> <td>ools Se</td> <td>ttings Simu</td> <td>lation S</td> <td>ubrout</td> <td>ine Viev</td> <td>v Load</td> <td>San</td> <td>nple Program Help</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td>	ile Edit T	ools Se	ttings Simu	lation S	ubrout	ine Viev	v Load	San	nple Program Help								
* Address Label Mnemonics Hexcode Bytes M-Cycles T-States ✓ 0000 EI FB 1 1 4 A ✓ 0001 MVI B,00 06 2 2 7 B 00 0	Editor	Assemble	er						Registers Memory	Devices							
✓ 0000 EI FB 1 1 4 Accumulator 000 0	Assem	bler							Registers :								
Ý 0001 MVI B,00 06 2 2 7 F 0002 00 0		Label								Value		_					-
0002 000 00 0<													_				-
✓ 0003 LOOP INR B 04 1 1 4 ✓ 0004 JMP LOOP C3 3 10 0005 00 0			MVI B,00		2	2	7					-	-	-			-
✓ 0004 JMP LOOP C3 3 10 0005 03 - - - 0 <td></td> <td>-</td> <td></td> <td>-</td>															-		-
0005 03 00 00 00 0		LOOP										_	_				-
0006 00 Register L 00 0			JMP LOOP		3	3	10					_	-				-
V 0007 HLT 76 1 2 5 003C MVI A,05 3E 2 2 7 003D 05 -													_				-
V 003C MVI A,05 3E 2 2 7 003D 05 05 05 05 06 06 0 </td <td></td> <td>_</td> <td>_</td> <td></td> <td></td> <td></td> <td>-</td>												_	_				-
003D 05 1 3 10 ✓ 003E RET C9 1 3 10 Flag Resister 00 0									Memory(M)	00	0 0	0	0	0	0	0	0
V 003E RET C9 1 3 10 Flag Resister 00 0			MVI A,05		2	2	7										
Image Resister 00 0									Resister	Value	S Z	*	AC	*	Ρ	*	CY
Stack Pointer(SP) 0000 Memory Pointer (HL) 0000 Program Status Word(PSW) 0000 Program Counter(PC) 0000 Clock Cycle Counter 0 Instruction Counter 0 Start From → 0000 0 0 0 0 For SIM instruction SOD SID INTR TRAP R7.5 R6.5 R5.5 0 0 0 0 Start From → 0000 Step By Step For SIM instruction SOD SDE * R7.5 MSE M M M 0 0 0 For RIM instruction SID 17.5 16.5 15.5 IE M M M 0	√ 003E		RET	C9	1	3	10		Flag Resister	00	0 0	0	0	0	0	0	0
Stack Pointer(SP) 0000 Memory Pointer (HL) 0000 Program Status Word(PSW) 0000 Program Counter(PC) 0000 Clock Cycle Counter 0 Instruction Counter 0 Start From → 0000 0 0 0 0 For SIM instruction SOD SID INTR TRAP R7.5 R6.5 R5.5 0 0 0 0 Start From → 0000 Step By Step For SIM instruction SOD SDE * R7.5 MSE M M M 0 0 0 For RIM instruction SID 17.5 16.5 15.5 IE M M M 0												_	_				
Memory Pointer (HL) 0000 Program Status Word(PSW) 0000 Program Counter(PC) 0000 Clock Cycle Counter 0 Instruction Counter 0 Start From → 0000 0 0 0 0 0 For SIM instruction SOD SID INTR TRAP R7.5 R6.5 R5.5 0 0 0 0 0 Bun all At a Time Step By Step For SIM instruction SOD SDE * R7.5 MSE M M M 0 </td <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>Type</td> <td></td> <td></td> <td></td> <td></td> <td>Valu</td> <td>e</td> <td></td> <td></td>									Type					Valu	e		
Program Status Word(PSW) 0000 Program Counter(PC) 0000 Clock Cycle Counter 0 Image: Start From → 0000 Run all At a Time Step By Step For RIM instruction SID INTR TRAP R7.5 R6.5 R5.5 0 0 0 0 0 0 0 Run all At a Time Step By Step For RIM instruction SID I7.5 I6.5 I5.5 IE M.									Stack Pointer(SP)					000	0		
Program Counter(PC) 0000 Clock Cycle Counter 0 Instruction Counter 0 Start From → 0000 Bun all At a Time Step By Step For RIM instruction SID 17.5 16.5 15.5 1E Main M M 0 0 0 0 No. Converter Tool : Hexadecimal Decimal Binary									Memory Pointer (HL)					000	0		
Program Counter(PC) 0000 Clock Cycle Counter 0 Instruction Counter 0 Start From → 0000 Bun all At a Time Step By Step For RIM instruction SID 17.5 16.5 15.5 1E Main M M 0 0 0 0 No. Converter Tool : Hexadecimal Decimal Binary										(PSW)				000	0		
Image: Simulate instruction Counter instruction instructins instructins instructins instruction instruction instruction in									Program Counter(PC)					000	0		
Simulate SOD SID INTR TRAP R7.5 R6.5 R5.5 Start From → 0000 0 0 0 0 0 0 0 0 Run all At a Time Step By Step Step By Step For RIM instruction SID I7.5 I6.5 I5.5 IE M <td></td> <td>0</td> <td></td> <td></td> <td></td>														0			
Simulate Start From → 0000 Run all At a Time Step By Step For RIM instruction SiD IT.5 IG.5 IS.5 IE M M M 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0								T	Instruction Counter					0			
Simulate Start From → 0000 Run all At a Time Step By Step For RIM instruction SiD IT.5 IG.5 IS.5 IE M M M 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0																	
Simulate Start From → 0000 Run all At a Time Step By Step For RIM instruction SiD IT.5 IG.5 IS.5 IE M M M 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0											D	07 5		P.6	5	DS	5 5
Start From → 0000 000 Run all At a Time Step By Step For SIM instruction SOD SDE * R7.5 MSE M M M 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 For RIM instruction SID 17.5 16.5 15.5 1E M M M 0 0 0 0 0 0 0 0 0 0 0 0 0 0 No. Converter Tool : Hexadecimal	📄 Simul	ate											,				
Run all At a Time Step By Step For SIM instruction SOD SDE * R7.5 MSE M M M M M Brown all At a Time Step By Step For RIM instruction SDD SDE * R7.5 MSE M M M 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	Start From	n → 000	0							0 0				0			0
Run all At a Time Step By Step For RIM instruction SID 17.5 16.5 15.5 1E M.M M 0 No. Converter Tool : Hexadecimal		000							For SIM instruction	50D 5D	- *	D7	E MC	E M		4	NA.
Run all At a Time Step By Step For RIM instruction SID 17.5 16.5 15.5 1E No. Converter Tool : Hexadecimal									i or sur instruction		_						
For RIM instruction SID 17.5 16.5 15.5 1E M M 0			- T!		C 1 - - - - - - - - - -	D C				0 0	0	U		ו	0	0	0
Sile File File <th< td=""><td><u> </u></td><td>un all At</td><td>alime</td><td></td><td>s<u>t</u>ep</td><td>By Step</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></th<>	<u> </u>	un all At	alime		s <u>t</u> ep	By Step											
No. Converter Tool : Hexadecimal Decimal Binary									For RIM instruction	SID 17.5	5 16.5	15.5	5 IE	M	N	1	M
Hexadecimal Decimal Binary								- 1		0 0	0	0	0)	0	0	0
Hexadecimal Decimal Binary									No. Converter Tool :							_	
											nal			B	inary	/	
										Deen		0				,	0
												0			_		

Normal code is loaded in the address 0000 and Inerrupt code loaded in 003c

Eile Edit Tools Settings Simulation Subroutine View Load Sample Program Help Editor Assembler Registers Memory Devices * Address Label Mnemonics Hexcode Bytes M-Cycles T-States	0 1 0 0
Address Label Mnemonics Hexcode Bytes M-Cycles T-States Register Value 7 6 5 4 3 2 1 V 0000 EI FB 1 4 Accumulator 00 0	0 1 0 0
* Address Label Mnemonics Hexcode Bytes M-Cycles T-States ✓ 0000 EI FB 1 4 Accumulator 00 0	0 1 0 0
* Address Label Mnemonics Hexcode Bytes M-Cycles T-States ✓ 0000 EI FB 1 4 ▲ ✓ 0001 MVI B,00 06 2 2 7 ■ 0002 00 0<	0 1 0 0
V 0000 EI FB 1 1 4 Accumulator 000 0	0 1 0 0
✓ 0001 MVI B,00 06 2 2 7 ≣ 0002 00 00 0 <td>1 0 0</td>	1 0 0
0002 00 00 0 00 0 </td <td>0</td>	0
✓ 0003 LOOP INR B 04 1 1 4 ✓ 0004 JMP LOOP C3 3 10	0
V 0004 JMP LOOP C3 3 3 10 Register E 000 0 0 0 0 0 0 0 0 0 0 0 0	_
	0
0005 00 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0
0006 00 00 00 00 00 00 00 00 00 00 00 00	0
V 0007 HLT 76 1 2 5 Memory(M) FB 1 1 1 1 1 0 1	1
√ 003C MVIA,05 3E 2 2 7	
003D 05 Resister Value S Z * AC * P *	CY
V 003E RET C9 1 3 10 Flag Resister 04 0 0 0 0 1 0	0
Type Value	
Stack Pointer(SP) 0000	
Memory Pointer (HL) 0000	
Program Status Word(PSW) 0004	
Program Counter(PC) 0004	
Clock Cycle Counter 127	
Instruction Counter 19	
instruction counter 19	
SOD SID INTR TRAP R7.5 R6.5 R	5.5
	0
Simulate Sod SDE * R7.5 MSE M M	M
	0
Start From $\rightarrow 0000$	
<u>Backward</u> <u>Stop</u> Forward <u>0 0 0 1 0 0</u>	0
No. Converter Tool :	
Hexadecimal Decimal Binary	
	0

Created by : Jubin Mitra

When we start the program Register B is incremented and reaches 09 value. Still R7.5 is set to 0.

e Edit T	ools Se	ttings Simu	lation S	ubrouti	ine Viev	8085 v Load		ulator ple Program Help							
	Assemble							Registers Memor	y Devices	1					
📄 Assem	bler							Registers :							
* Address ✓ 0000 ✓ 0001 0002 ✓ 0003 ✓ 0004 0005 0006 ✓ 0007 ✓ 003C	Loop	Mnemonics EI MVI B,00 INR B JMP LOOP HLT MVI A,05	Hexcode FB 006 00 04 C3 03 03 00 76 3E	Bytes 1 2 1 3 3 1 2	M-Cycles 1 2 1 3 2 2 2	T-States 4 7 4 10 5 7		Register Accumulator Register B Register C Register D Register E Register H Register H Register L Memory(M)	09 00 00 00 00 00	7 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1	5 0 0 0 0 0 0 0 0 1	4 3 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1	0 0 0 0 0 0 0 0	1 0 0 0 0 0 0 0 1	0 0 1 0 0 0 0 1 1
003D √ 003E 		RET	05	1	3	10	•	Resister Flag Resister Type Stack Pointer(SP) Memory Pointer (HL Program Status Wor Program Counter(PC Clock Cycle Counter Instruction Counter) d(PSW))	S Z 0 0	*	00	1 00 00 04 04 7	*	CY 0
Simul Start Fror Bac			ŝtop		Forwa	rd		SOD SID 0 0 For SIM instruction	0 0	*	0	MSE 0	0		5.5 0 M 0 M
			- •					No. Converter Tool Hexadecimal O	: Decim	al	0		Binai	ry	0

Now we set the value of R7.5 to 1, enabling RST7.5.

						8085	Sim	ulator										>
<u>F</u> ile Edit	Tools Se	ettings Simu	lation S	ubrouti	ine View	/ Load	San	ple Progra	ım Help									
Editor	Assembl	ler						Register	rs Memory	/ Devices								
Asse	mbler							📄 Regis	sters :									
		1								1								
* Addre	ss Label	Mnemonics	Hexcode	Bytes	M-Cycles				Register	Value		6	5	4	3	2	1	0
√ 0000		EI	FB	1	1	4		Accum		05		-	0		0	1	0	1
√ 0001		MVI B,00	06	2	2	7		Registe		09		-	0	-	1	0	0	1
0002			00					Registe		00		_	0		0	0	0	0
√ 0003	LOOP	INR B	04	1	1	4		Registe		00		-	0	-	0	0	0	0
√ 0004		JMP LOOP	C3	3	3	10		Registe		00	-		0	-	0	0	0	0
0005			03					Registe		00		-	0		0	0	0	0
0006			00					Registe		00)	0	-	0	0	0	0
√ 0007		HLT	76	1	2	5		Memory	y(M)	FB	1	L	1	1	1	0	1	1
√ 003C		MVI A,05	3E	2	2	7						_					_	
003D			05						Resister	Value	S	Ζİ	*	AC	*	P	*	CY
√ 003E		RET	C9	1	3	10		Flag Re	sister	04	0 ()	0	0	0	1	0	0
									Туре						alu			
									ointer(SP)						00			
									y Pointer (HL)						00			
									n Status Word						504			
									n Counter(PC))					03	E		
									ycle Counter						44			
							-	Instruc	tion Counter					2	1			
														_		_		
								SOD	SID	INTR TRA	٩P	R7	7.5	F	۱ 6 .5	,	R5	.5
								0	0	0 ()		1		0			0
🗂 Sim	ulate							For SIM	l instruction	SOD SD	F *	R	7.5	MSE	M	. N	1	М
Start Fr		00										_	0	0		0	0	0
	00	00																
							1	For RIM	1 instruction	SID 17.	5 16 9		5 5	IE	М.	N	1	М
			_		_					0	_		0	0		0	0	0
B	ackward	5	<u>S</u> top		For <u>w</u> a	rd				0	. 0		0	0		0	0	0
								No. Co	nverter Tool	:								
								Но	xadecimal	Deci	mal				Bi	nary	,	
								0	Autechnal	Deci	nai		0		DI	nar y		0
				_										_				-
Created b	v : lubin l	Mitra																

When we set the value of R7.5 automatically control jumps to location 003ch and register A value set to as 05.

Experiment No:7 AIM: Interfacing a program to initiate 8251 and to check transmissionand reception of character

DESCRIPTION/ALGORITHM:-

Steps:1.Intitialize timer IC 2.Move the mode command word to A 3. Output it to port address C2 4. Moce the command instruction word to A reg. 5.Output it to port address C2 6.Move the data to be transferred to A 7.Output it to port address C0 8.Reset the system 9.Get data from input port C0 10.Store the value in memory 11.Reset the system **PROGRAM:** MVI A,36H Out CEH MVI A,0AH Out C8 HLXI H,4200H MVI A,4EH Out C2H MVIA, 37H Out C2H MVI A, 42H Out C0H RST 1 #ORG 4200H In C0H STA 4500H RST 1

Assembler						Interfacing device	
Assembler							
Address Labe	Mnemonics	Hexcode	Bytes	M-Cycles	T-States	I/O Port Editor	
0000	MVI A,36	3E	2	2	7	0 1 2 3 4 5 6 7 8 9 A B C D E F	
0001		36				0 1 2 3 4 5 6 7 8 9 A B C D E F	
0002	OUT CE	D3	2	3	10		
0003		CE					
0004	MVI A,0A	3E	2	2	7	20 00 00 00 00 00 00 00 00 00 00 00 00 0	
0005		0A				30 00 00 00 00 00 00 00 00 00 00 00 00 0	
0006	OUT C8	D3	2	3	10		
0007		C8				50 00 00 00 00 00 00 00 00 00 00 00 00 0	
0008	LXI H,4200	21	3	3	10	60 00 00 00 00 00 00 00 00 00 00 00 00 0	
0009		00				70 00 00 00 00 00 00 00 00 00 00 00 00 0	
000A		42				80 00 00 00 00 00 00 00 00 00 00 00 00 0	
000B	MVI A,4E	3E	2	2	7	90 00 00 00 00 00 00 00 00 00 00 00 00 0	
000C		4E				A0 00 00 00 00 00 00 00 00 00 00 00 00 0	
000D	OUT C2	D3	2	3	10	B0 00 00 00 00 00 00 00 00 00 00 00 00 0	
000E		C2				C0 42 00 37 00 00 00 00 00 0A 00 00 00 00 36 00	
000F	MVI A,37	3E	2	2	7	D0 00 00 00 00 00 00 00 00 00 00 00 00 0	
0010		37				E0 00 00 00 00 00 00 00 00 00 00 00 00 0	
0011	OUT C2	D3	2	3	10	F0 00<	
0012		C2					
0012	MULLA 4D	25			7		
] Simulate tart From → [<u>B</u> ackward	0000			Forwa	rd		

Have a look at the I/O port addresses C000h,C002H,C008H,C00EH.