CHIRANJEEVI REDDY INSTITUTE OF ENGINEERING & TECHNOLOGY

(Approved by AICTE,New Delhi & Affiliated to JNTU,ANANTAPUR)
Susheela Nagar, Bellary Road, ANANTAPUR.

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING.

MICROPROCESSORS AND DSP LAB (9A04708)

(IV B.Tech I Semester)

LAB-MANUAL

Head of the Department

S.Raghavendra Swami
M-Tech.,
Assistant professor in ECE.

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY ANANTAPUR

Electronics and Communication Engineering

(9A04708) MICRO PROCESSORS & DSP LAB

B.Tech IV-I Sem. (E.C.E.)

I. Microprocessor 8086 & Microcontroller 8051:

(Any four from 1 – 6, and 7, 8 are compulsory)

1. Arithmetic operation – Multi byte Addition and Subtraction, Multiplication and Division – Signed and unsigned Arithmetic operation, ASCII – arithmetic operation.
2. Logic operations – Shift and rotate – Converting packed BCD to unpacked BCD, BCD to ASCII conversion.
3. By using string operation and Instruction prefix: Move Block, Reverse string, Sorting, Inserting, Deleting, Length of the string, String comparison.
4. Reading and Writing on a parallel port.
5. Timer in different modes.
6. Serial communication implementation.
7. 8259 – Interrupt Controller: Generate an interrupt using 8259 timer.
8. 8279 – Keyboard Display: Write a small program to display a string of characters.

II. DSP Processor: (Any six of the following)

1. To study the architecture of DSP chips – TMS 320C 5X/6X Instructions.
2. To verify linear convolution.
3. To verify the circular convolution.
4. N-point FFT algorithm.
5. MATLAB program to find frequency response of analog LP/HP filters.
6. To compute power density spectrum of a sequence.

Additional Experiments:

1. To Implement IIR Low pass filter
2. To Implement IIR High pass filter.

Design Experiments:

1.To design FIR filter (LP/HP) using rectangular window technique
2. To design FIR filter (LP/HP) using triangular window technique
3. Using Kaiser window design FIR filter (LP/HP)

Equipment required for Laboratories:

1. 8086 µP Kits
2. 8051 Micro Controller kits
3. Interfaces/peripheral subsystems
   i) 8259 PIC
   ii) 8279-KB/Display
   iii) 8255 PPI
   iv) 8251 USART
4. ADC Interface
5. DAC Interface
6. Traffic Controller Interface
7. Elevator Interface

TABLE OF CONTENTS

Microprocessors Programming (8086)-Trainer Kit		
S.NO	**NAME OF THE EXERCISE**	
1	ALP Program to initialize the register with immediate data	
2	ALP Program to move the data from memory locations to registers	
3	ALP Program to move a block of 10 bytes	
4	ALP Program to interchange two words	
5	ALP Program to	
	A) Addition of two 8-bit numbers	
	B) Addition of two 16-bit numbers	
	C) Addition of two 32-bit numbers	
6	ALP Program to	
	A) Subtraction of two 8-bit numbers	
	B) Subtraction of two 16-bit numbers	
7	ALP Program to	
	A) Multiplication of two 8-bit numbers	
	B) Multiplication of two 16-bit numbers	
8	ALP Program to	
	A) Division of 16 - bit number by 8 - bit number	
	B) Division of 32 - bit number by 16 - bit number	
9	A) ALP Program for sum of given ‘n’ numbers	
	B) ALP Program for average of given ‘n’ numbers	
	C) Program for sum of squares of ‘N’ numbers	
10	A) ALP Program to sort given 8 numbers in descending order	
	B) ALP Program to sort given 8 numbers in asescending order	
11	ALP Program to	
	A) Hexadecimal to Decimal Number	
	B) Decimal to Hexadecimal Number	
12	ALP Program to compute the logical 1’s in a word	
13	ALP Program to convert binary to ASCII	

MICROPROCESSOR 8086 – MASM PROGRAMS		
S.NO	**NAME OF THE EXERCISE**	
1	A) Addition of two 8-bit numbers	
	B) Subtraction of two 8-bit numbers	
2	A) Addition of two 16-bit numbers	
	B) Subtraction of two 16-bit numbers	
3	A) Multiplication of two 8 -bit numbers	
	B) Multiplication of two 16 -bit numbers	
4	A) Division of 16-bit number by an 8 -bit number	
	B) Division of 32-bit number by a 16-bit number	
5	A) Signed Multiplication	
	B) Signed Division	
6	A) ASCII Arithmetic Operation – Addition	
	B) ASCII Arithmetic Operation – Subtraction	
	C) ASCII Arithmetic Operation – Multiplication	
	D) ASCII Arithmetic Operation – Division	
7	MASM Program for converting Packed BCD to Unpacked BCD	
8	MASM Program for converting BCD to ASCII	
9	A) Decimal adjustment after addition	
	B) Decimal adjustment after subtraction	
10	A) Block move	
	B) STRING REVERSING: for any Known Character Length	
	C) Length of the string	
	D) String comparison	
	E) String insertion	
	F) String deletion	
	G) Display the string	

MICROPROCESSOR(8086) - INTERFACING PROGRAMS		
1	Generation of positive RAMP wave using DAC Generation of triangular wave using DAC	
2	Generation of Negative RAMP using DAC	
3	Generation of Square Wave	
4	Generation of triangular wave using DAC	
5	Generation of SINE wave using DAC	
6	Stepper motor control using 8086	
7	Traffic light control	
8051 Microcontrollers		
1	Reading and writing on a parallel port of 8051	
2	8051 Timers	
3	Bit and byte operations by using 8051	
4	Understanding three memory areas of 00-ff	
5	2's compliment	
6	8-bit subtraction	
7	8-bit division	
8	Binary to bcd conversion	
9	Sum of n-number	
10	To find smallest number	
11	To find biggest number	
12	BCD TO ASCII coversion	
13	Generation of 00 to 0f numbers	

INTRODICTION TO ALS-SDA-86 MEL TRAINER KIT

OVERVIEWS OF TRAINER

The lNTELs 8086 CPU are very popular and powerful third generation microprocessor. This processor has 16 bit internal architecture, and can transfer 16 bits at a time. Because of its powerfu architecture and flexibility in operation, this processor is suitable for a wide spectrum of micro – computer applications

SDA trainer is a <u>System Design Aid</u> for the Intel 8086 microprocessors. IT is a highly versatile and powerful system designed to assist students and engineers in learning about the architecture and programming of these processors and designing systems& interfacing around them. This trainer kit is configured for maximum mode only. It is housed in an attractive plastic cabinet; with the power supply is connected externally.

The salient features of the system are as follow:

* 8086 CPU operating at 5 MHz in MAX mode.
* Provision for on board 8087 coprocessor
* Provision for 256 KB of EPROM and 256 KB of ram onboard
* Battery backup facility for RAM.
* 48 programmable I/) lens using two 8355's
* Three 16 bit timers using 8253A
*. Priority interrupt controller (PIC) for eight input using 8259A
* Computer compatible keyboard* Display is 16X2 line LCD
* Designed and engineered to integrate user's 8259A
* Computer compatible keyboard
* Display is 16X2 line LCD
* Designed and engineered to integrate user's application specific interface conveniently at a minimum cost.
* Powerful and user friendly keyboard/ serial monitor, support in development of application programs.
* Software support for development of programs on computer, the RS 232C interface cable connecting to computer from the kit facilitates transfer of files between the trainee kit& computer for development and debugging purposes
* High quali9ty reliable PCB with maximum details provided for the user.

SPECIFICATIONS

CPU: Intel 8086 operating at 5MHz in MAX mode; provision for onboard 8087 Co-processor

MEMORY: Total 256KB of memory is provided in the kit

EPROM:2JEDEC compatible sockets for EPROM;Onboard EPROM capacity is 256KB (27C010X2)
128KB of EPROM containing keyboard/serial monitor will be supplied.
RAM: 2 JDEC compatible sockets are provided for RAM.
64KB of ram will be 62256X2 will be supplied.
PARALLEL I/O48 I/O lines using two 8255s

SERIAL I/O: One RS 232C compatible interface using USART 8251A,with hardware selectable baudrate using one channel of 8253 timer with MAX 232C IC

TIMER: Three 16 bit counter/ timers using 8253A counter 1 is used for serial I/O baudrate generation

PIC: Programmable interrupt controller using 8259A provides interrupt vectors for 8 jumper selectable internal/ external sources

KEYBOARD/ DISPLAY :

KEYBOARD: Computer keyboard can be hocked on to the trainer.

DISPLAY: LCD 2x16display:

INTERRUPTS:

NMI: Provision for connecting NMI to a key switch.
INTR: Programmable interrupt controller using 8259A provides interrupt vectors for 8 jumper selectable/external source. On board interrupt source include: 8251A
TXRDY and RXRDY, 8255 and 8087.

INTERFACE BUS SIGNALS

CPU BUS: All address, data and center lines ate TTL compatible and are terminated in berg strip header

PARALLER I/O: All signals are TTL compatible and terminated in berg strip headers for PPI expansion. It’s compatible with all of our experimental interface modules.

SERIAL I/O: Serial port signals are terminated in standard 9 pin D type connecter.

MONITOR SOFTWARE: 128 KB of serial/ keyboard monitor with powerful commands to enter verify and debug user programs, including on board assembler &disassemble commands.

COMPUTER INTERFACE This can be interfaced to host computer system through the main serial port; The driver program or communication between computer and trainer allows the computer to be used as a simple dumb terminal and also facilitates uploading

POWER REQUIREMENTS: +5V DC, with 2.5 Amps current rating(max).

OPERATING CONFIGURATION: Two different modes of operation of trainer are possible. They are:

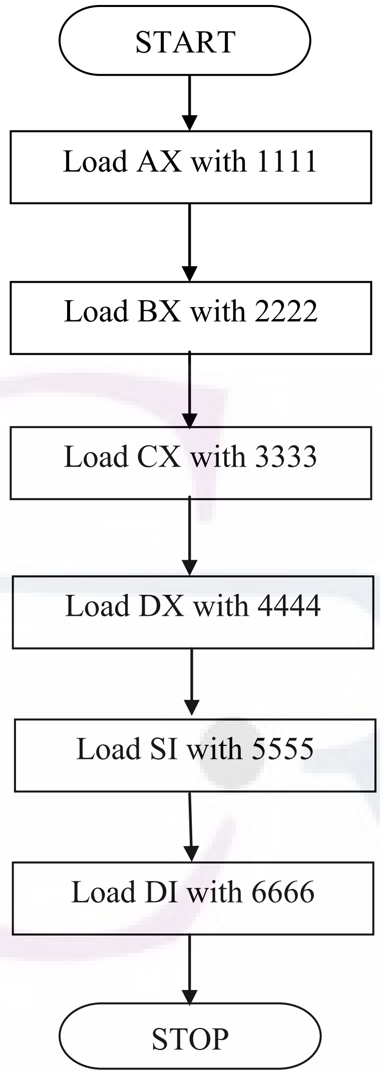
i) serial operation
ii) Keyboard operation

The first configuration requires a computer system with an RS -232C port, can be used as the controlling device as shown below.

RS 232C **CABLE**

◄ **Computer** ———————— ► **Trainer**

When a Computer system is interfaced to trainer, the driver program must be resident in the computer system.
The second mode of operation id achieved through onboard KEYBOARD/DISPAY.
In this mode the trainer kit interacts with the user through a computer keyboard and 16x2 LCD display. This configuration eliminates the need for a computer and offers a convenient way for using the trainer as a stand –alone system.



PROG 1 : Flow chart to move Immediate Data

PROG 1	**ALP Program to initialize the register with immediate data**	

AIM: program initializes the registers with immediate data

APPARATUS:

ALS-86 µp kit.
Key board
Power supply

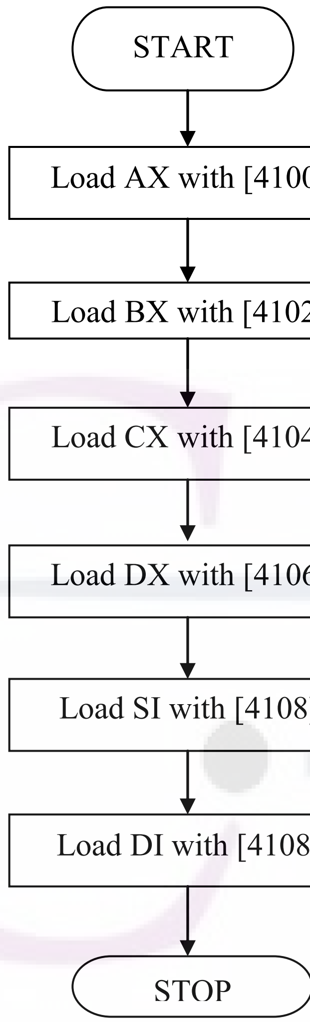
ALGORITHM:

1. Start.
2. Load content in AX Register
3. Load content in BX Register
4. Load content in CX Register
5. Load content in DX Register
6. Load content in SI Register
7. Load content in DI Register
8. End program

SAMPLE OUTPUT:-

AX = 1111	BX=2222,
CX=3333	DX=4444,
SI=5555	DI=6666.

RESULT: Thus the initialization of the register with immediate data has been executed successfully and the result is verified.



PROG 2 : Flow chart for Data movement from memory to registers

PROG 2	**ALP Program to move the data from memory locations to registers**	

AIM:
program initializes the registers with immediate data

APPARATUS:

ALS-86 µp kit.
Key board
Power supply

ALGORITHM:

1. Start.
2. Load content in AX Register from memory location 4100
3. Load content in BX Register from memory location 4102
4. Load content in CX Register from memory location 4104
5. Load content in DX Register from memory location 4106
6. Load content in SI Register from memory location 4108
7. Load content in DI Register from memory location 410A
8. End program

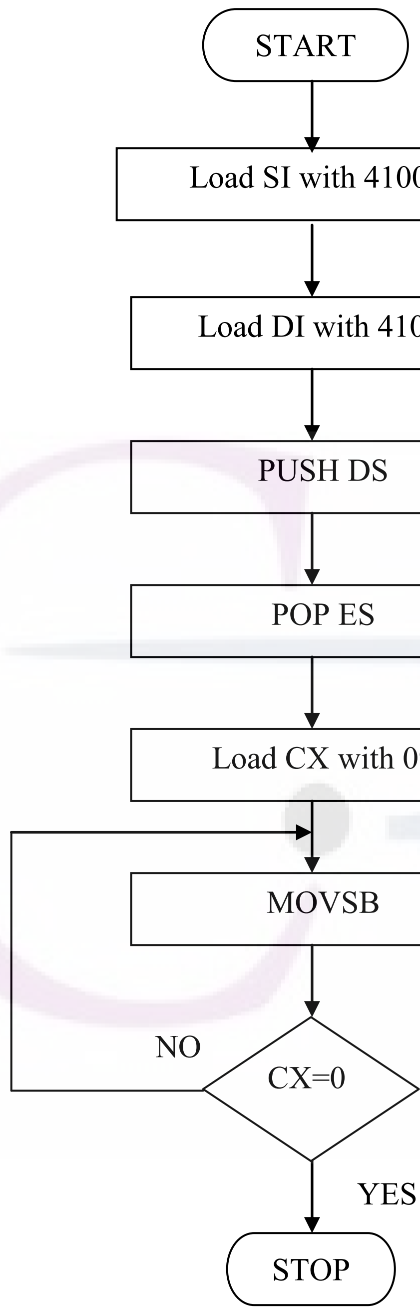
INPUT :-

4100 : 1111
4102 : 2222
4104 : 3333
4106 : 4444
4108 : 5555
410A: 6666

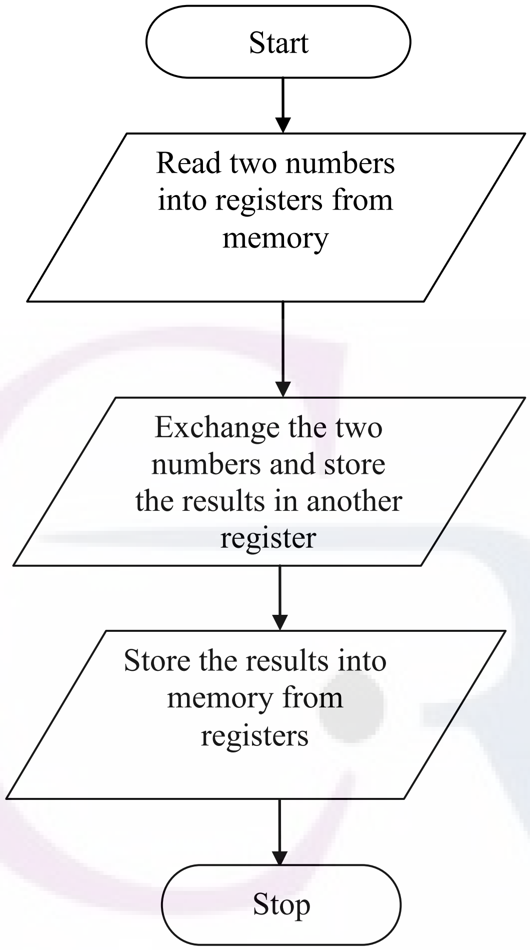
OUTPUT:-

AX = 1111,
BX= 2222,
CX= 3333,
DX=4444,
SI=5555,
DI= 6666.

RESULT: Thus the date is moved from memory to the registers has been executed successfully and the result is verified.


START
Load SI with 4100
Load DI with 410A
PUSH DS
POP ES
Load CX with 09
MOVSB
NO
CX=0
YES
STOP

Prog 3 : Flow chart to Move a block of ten bytes



PROG 3	**ALP Program to move a block of 10 bytes**	

AIM:

program to move a block of memory from one memory location to other

APPARATUS:

ALS-86 µp kit.
Key board
Power supply

ALGORITHM:

1. Start.
2. Load the source Index with starting memory address of content
3. Load the destination Index where to store the memory content
4. Initialize the counter **CX** with value 10
5. Push the contents on to the stack and pop the contents from the stack and store in the memory location pointed by the DI
6. Repeat the step 5 for 10 times until CX becomes 0.
7. End of program

INPUT :-4100 – 4109 : 1,2,3,4,5,6,7,8,9,A

OUTPUT:- 410A - 4114 : 1,2,3,4,5,6,7,8,9,A

RESULT: Thus the program to move a block of memory from one memory location to other has been executed successfully and the result is verified.

PROG 4	**ALP Program to interchange two words**	

AIM:

Write a program to interchange two words

APPARATUS:

ALS-86 µp kit.
Key board
Power supply

ALGORITHM:

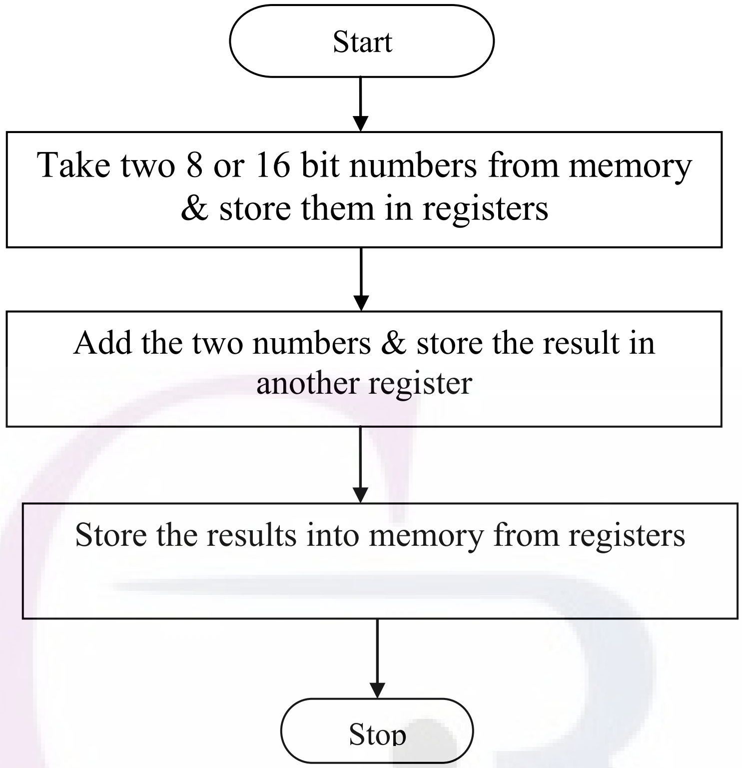
1. Move pointer 1 SI to 4100 location
2. Move Pointer 2 DI to 4102 location
3. Move DI contents to AX
4. Move SI contents to BX
5. Exchange AX with BX
6. Now store them back and Move the content
7. End of program

INPUT :- 4100 : 4101 - AAAA
4102 : 4103- BBBB

OUTPUT:- 4100 : 4101 - BBBB
4102 : 4103 – AAAA

RESULT: Thus the program to interchange two words has been executed successfully and the result is verified.

Flow chart to add 2- 8 or 16 bit numbers



PROG 5 (A)	**Addition of Two 8-Bit numbers**	

AIM:
To perform the addition of two 8 bit numbers using 8086

APPARATUS:

ALS-86 µp kit.
Key board
Power supply

ALGORITHM:

1. Initialize the segment address & clear the carry flag
2. Initialize the destination address
3. Move the first 8-bit number to AL register
4. Move the second 8-bit number to BL register
5. Add the contents of AL and BL and store result in AL
6. End of program

INPUT :- 4100 : 45

4101 : 55

OUTPUT:- 4102 : 9A

RESULT: Thus the addition of two 8-bit numbers has been executed successfully using 8086 microprocessor and the result is verified.

PROG 5 (B)	**Addition of Two 16-Bit numbers**	

AIM:

To perform the addition of two 16- bit numbers using 8086

APPARATUS:

ALS-86 µp kit.
Key board
Power supply

ALGORITHM:

1. start
2. Initialize the segment address & clear the carry flag
3. Initialize the destination address
4. Move the first 8-bit number to AX register
5. Move the second 8-bit number to BX register
6. Add the contents of AX and BX and store result in AX
7. End of program

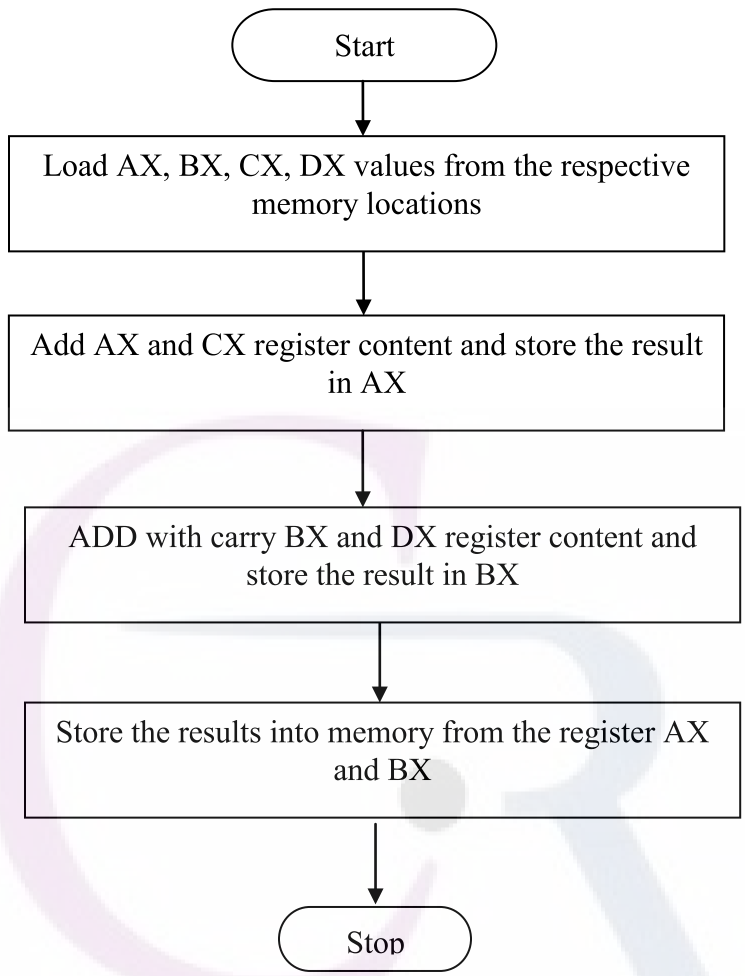
INPUT :- 4100 : ABCD

4102 : 1111

OUTPUT:- 4104 : BCDE

RESULT: Thus the addition of two 16-bit numbers has been executed successfully using 8086 microprocessor and the result is verified

Flow chart to add 2 32-bit numbers



PROG 5 (C)	**Addition of Two 32-Bit numbers**	

AIM:

To perform the addition of two 32 bit numbers using 8086

APPARATUS:

ALS-86 µp kit.
Key board
Power supply

ALGORITHM:

1. start
2. Initialize the segment address & clear the carry flag
3. Initialize the destination address
4. Load AX, BX, CX, DX values from respective memory locations
5. Add AX and CX register content and store the result in AX
6. ADD with carry BX and DX register content and store the result in BX
7. Store the results into memory from registers AX and BX
8. End of program

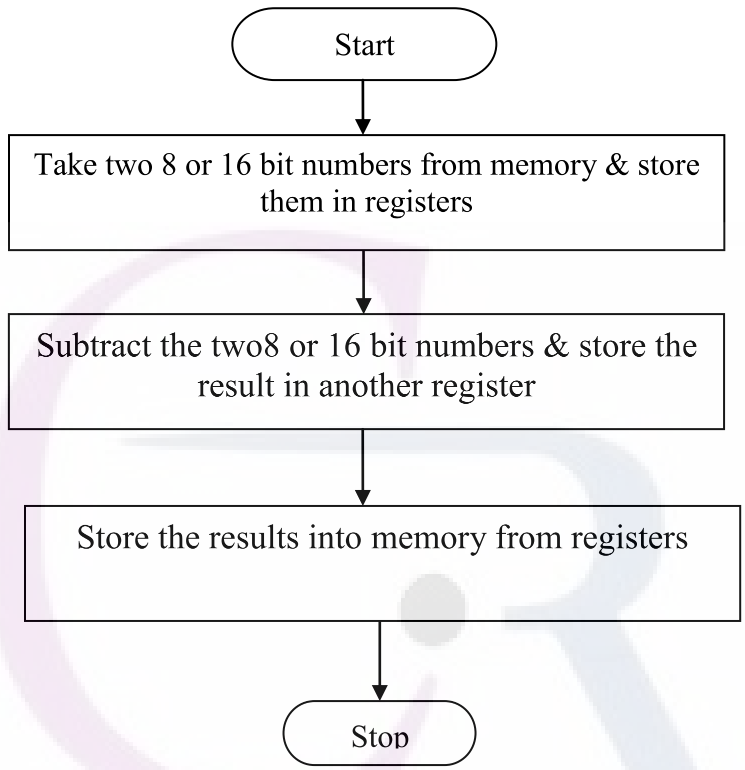
INPUT :- 4100 : 11 11 22 22

4104 : 33 33 44 44

OUTPUT:- 4108 : 44 44 66 66

RESULT: Thus the addition of two 32-bit numbers has been executed successfully using 8086 microprocessor and the result is verified

Flow chart to subtract two 8 and 16 bit numbers



PROG 6 (A)	**Subtraction of Two 8-Bit numbers**	

AIM:

To perform the subtraction of two 8 bit numbers using 8086

APPARATUS:

ALS-86 µp kit.
Key board
Power supply

ALGORITHM:

1. start
2. Initialize the segment address & clear the carry flag
3. Initialize the destination address
4. Move the first 8-bit number to AL register
5. Move the second 8-bit number to BL register
6. Subtract the contents of AL and BL and store result in AL
7. Store the result into memory from registers
8. End of program

INPUT :- 4100 : 45

4101 : 55

OUTPUT:- 4102 : F0

RESULT: Thus the Subtraction of two 8-bit numbers has been executed successfully using 8086 MP and the result is verified.

PROG 6 (B)	**Subtraction of Two 16-Bit numbers**	

AIM:

To perform the subtraction of two 16- bit numbers using 8086

APPARATUS:

ALS-86 µp kit.
Key board
Power supply

ALGORITHM:

1. start
2. Initialize the segment address & clear the carry flag
3. Initialize the destination address
4. Move the first 16-bit number to AX register
5. Move the second 16-bit number to BX register
6. Subtract the contents of AX and BX and store result in AX
7. Store the result into memory from registers
8. End of program

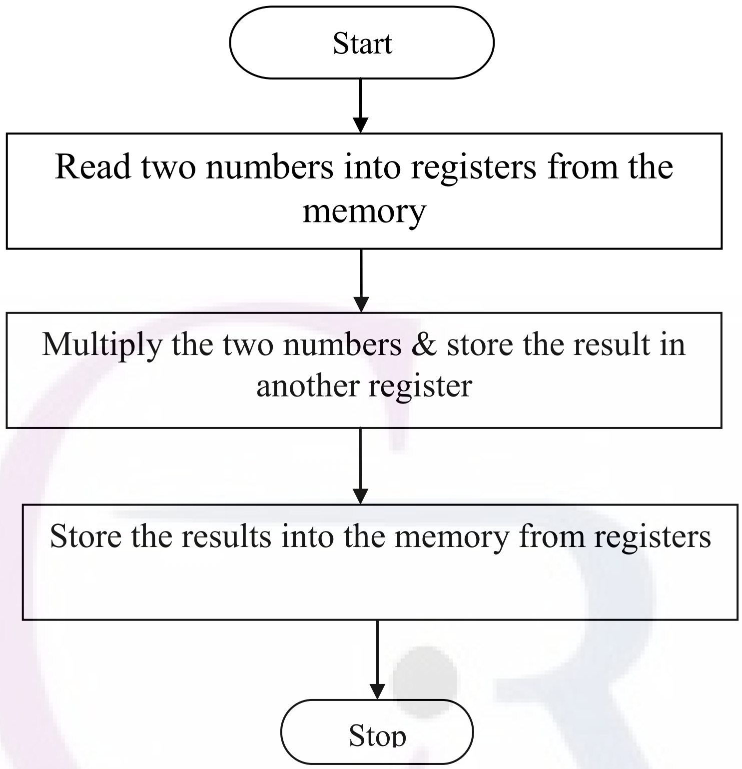
INPUT :- 4100 : 4433

4102 : 2211

OUTPUT:- 4104 : 2222

RESULT: Thus the Subtraction of two 16 -bit numbers has been executed successfully using 8086 MP and the result is verified

Flow chart to multiply two bytes numbers



PROG 7 (A)	**Multiplication of Two 8-Bit numbers or Byte**	

AIM:

To perform the multiplication of two 8- bit numbers using 8086

APPARATUS:

ALS-86 µp kit.
Key board
Power supply

ALGORITHM:

1. start
2. Initialize the segment address
3. Initialize the destination address
4. Read two numbers into registers from memory
5. Multiply the two numbers & store the result in another register
6. Store the result into memory from registers
7. End of program

INPUT :- 4100 : 33

4101 : 22

OUTPUT:- 4102 : C606

RESULT: Thus the multiplication of two 8 -bit numbers has been executed successfully using 8086 MP and the result is verified

PROG 7(B)	**Multiplication of Two 16-Bit numbers or word**	

AIM:

To perform the multiplication of two 16- bit numbers or word using 8086 microprocessor

APPARATUS:

ALS-86 µp kit.
Key board
Power supply

ALGORITHM:

1. Initialize the segment address
2. Initialize the destination address
3. Take two 16 bit numbers from memory & store them in registers
4. Multiply the two numbers and store the result in another register
5. Store the result into memory from registers
6. End of program

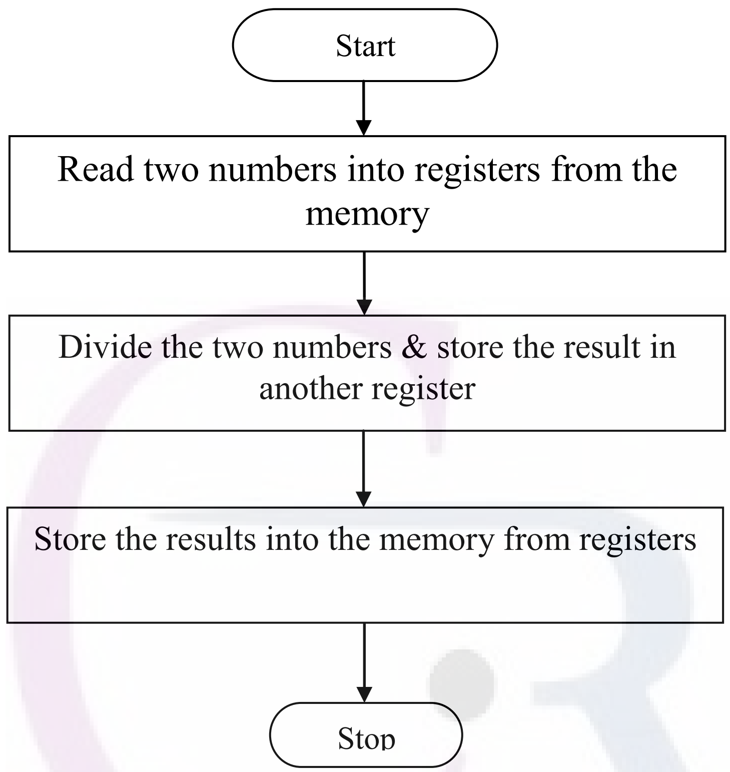
INPUT :- 4100 : 3333

4102 : 2222

OUTPUT:- 4104 : 92C6
4106 : 0603

RESULT: Thus the multiplication of two 8 -bit numbers has been executed successfully using 8086 MP and the result is verified

Flow chart to divide 16 bit by 8 bit number



PROG 8(A)	**Division of 16 bit by 8 bit number**	

AIM:

To perform the division of 16 bit by 8 bit number using 8086 microprocessor

APPARATUS:

ALS-86 µp kit.
Key board
Power supply

ALGORITHM:

1. start
2. Initialize the segment address
3. Initialize the destination address
4. Load two numbers from memory to registers
5. Divide the two numbers &store the result in a register
6. Store the result into memory from registers
7. End of program

INPUT :- 4100 : 2489

4102 : 0002

OUTPUT:- 4103 : 1244

4105 : 0001

RESULT: Thus the division of 16-bit number by 8 bit number has been executed successfully using 8086 MP and the result is verified

PROG 8(B)	**Division of two 32 bit by 16 bit number**	

AIM:

To perform the division of 32- bit number by 16 –Bit number using 8086 microprocessor

APPARATUS:

ALS-86 µp kit.
Key board
Power supply

ALGORITHM:

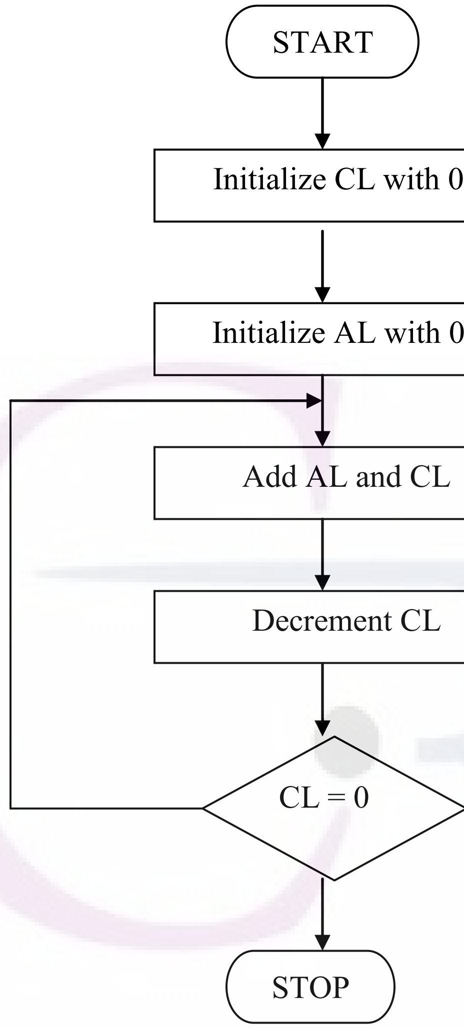
1. start
2. Initialize the segment address
3. Initialize the destination address
4. Load two numbers from memory to registers
5. Divide the two numbers &store the result in a register
6. Store the result into memory from registers
7. End of program

INPUT :- 4100 : 2849
4102 : 0000
4104 : 0002

OUTPUT:- AX : 1244
DX : 0001

RESULT: Thus the division of 32-bit number by 16- bit number has been executed successfully using 8086 MP and the result is verified

Sum of given N numbers



PROG 9(A)	**Program for sum of given ‘n’ numbers**	

AIM:

To perform the sum of given numbers(1,2,3,4,and 5) using 8086 microprocessor

APPARATUS:

ALS-86 µp kit.
Key board
Power supply

ALGORITHM:

1. start
2. Initialize the CL register with 05
3. Initialize the register A to 00
4. Add register contents of A and C
5. Decrement the count value if count is not equal to zero jump to Loop1(L1)
6. Repeat step 5 up to count becomes 0
7. End of program

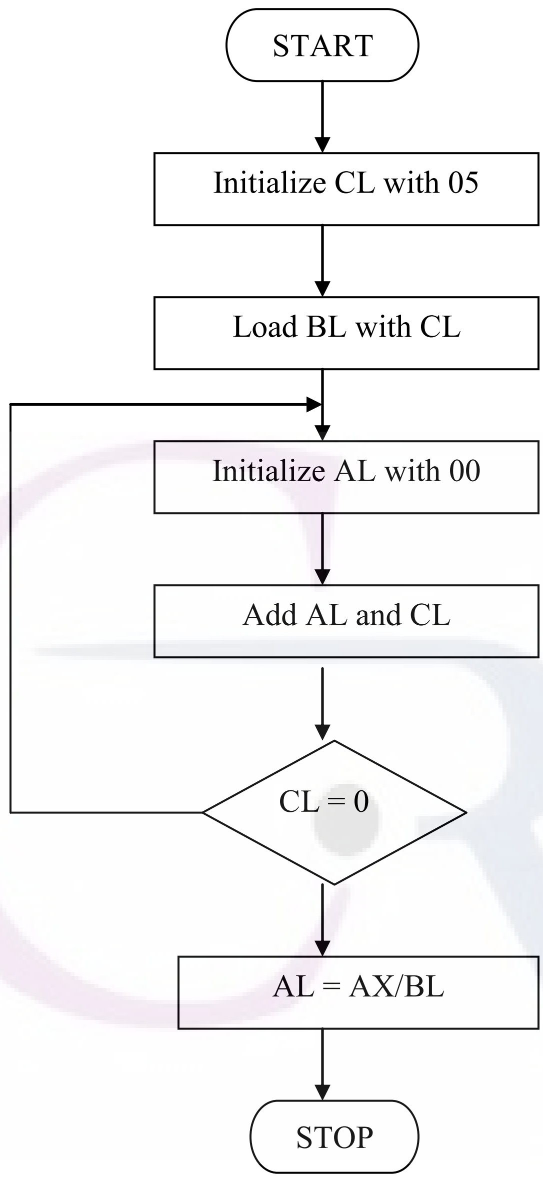
INPUT :-

5+4+3+2+1

OUTPUT:- AL: 0F

RESULT: Thus the sum of given N numbers(1,2,3,4,and 5) has been executed successfully using 8086 MP and the result is verified

Average of given N numbers



PROG 9(B)	**Program for average of given 'n' numbers**	

AIM:

To perform the average of given numbers(1,2,3,4,and 5) using 8086 microprocessor

APPARATUS:

ALS-86 µp kit.
Key board
Power supply

ALGORITHM:

1. start
2. Initialize the CL and BL register with 05
3. Initialize the register A to 00
4. Add register contents of A and C
5. Decrement the count value if count is not equal to zero jump to Loop1(L1)
6. Repeat step 5 up to count becomes 0
7. Divide the content of A with 05.
8. End of program

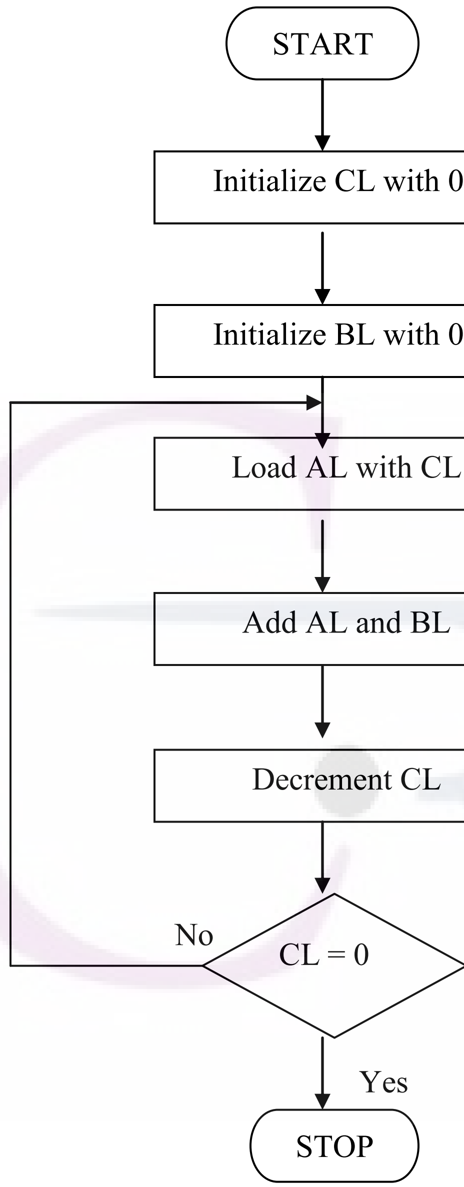
INPUT :-

(5+4+3+2+1) / 05

OUTPUT:- AL: 03

RESULT: Thus the average of given N numbers(1,2,3,4,and 5) has been executed successfully using 8086 MP and the result is verified

Sum of squares of N numbers



PROG 9(C)	**Program for sum of squares of 'N' numbers**	

AIM:

To perform the sum of squares of 'N' numbers (1,2,3,4,and 5) using 8086 microprocessor

APPARATUS:

ALS-86 µp kit.
Key board
Power supply

ALGORITHM:

1. start
2. Initialize the CL and load the same into AL
3. Initialize the register BL to 00
4. Multiply CL with AL
5. Add AL and BL store result in BL
6. Decrement the count value if count is not equal to zero jump to Loop1(L1)
7. Repeat step 6 up to count becomes 0
8. End of program

INPUT :-

$5^2+4^2+3^2+2^2+1$
25+16+9+4+1

OUTPUT:- AL: 37(Hex) 55(Dec)

RESULT: Thus the sum of squares of given N numbers(1,2,3,4,and 5) has been executed successfully using 8086 MP and the result is verified

PROG 10(A)	**Program to Sort Given 8 bit Number in descending order**	

AIM:

To perform a program to sort given 8 number in descending order starting from address 4100

APPARATUS:

ALS-86 µp kit.
Key board
Power supply

ALGORITHM:

1. start
2. Initialize the segment address
3. Initialize the destination address
4. Counter initializes to 8
5. BX to point input stream
6. Get the byte AL
7. Increment the pointer
8. Compare AL with next byte
9. If AL>=[BL] jump out
10. Else exchange
11. Adjacent byte
12. If exchange is mode flag it
13. Loop till the end
14. Now see if a pass is exchange free

End of program

INPUT :-

4100 : 02
4101 : 04
4102 : 06
4103 : 03
4104 : 01
4105 : 05
4106 : 08
4107 : 07

OUTPUT:-

4100 : 08
4101 : 07
4102 : 06
4103 : 05
4104 : 04
4105 : 03
4106 : 02
4107 : 0

RESULT:

By using 8086 microprocessor trainer the given 8 bytes of data is sorted in the descending order

PROG10 (B)	**Program to Sort Given 8 bit Number in ascending order**	

AIM:

To perform a program to sort given 8 number in ascending order starting from address 4100

APPARATUS:

ALS-86 µp kit.
Key board
Power supply

ALGORITHM:

1. start
2. Initialize the segment address
3. Initialize the destination address
4. Get the data byte into AL
5. Get the 10 digit to lower
6. Nibble of AL, most with units
7. Multiply by 10's digit by 10
8. Add unit to it
9. End of program

INPUT :-

4100 : 02
4101 : 04
4102 : 06
4103 : 03
4104 : 01
4105 : 05
4106 : 08
4107 : 07

OUTPUT:-

4100 : 01
4101 : 02
4102 : 03
4103 : 04
4104 : 05
4105 : 06
4106 : 07
4107 : 08

RESULT:

By using 8086 microprocessor trainer the given 8 bytes of data is sorted in the ascending order

PROG 11(A)	**Program to convert given Hex decimal number to decimal number**	

AIM:

Program to convert given Hexa decimal number to decimal number

APPARATUS:

ALS-86 µp kit.
Key board
Power supply

ALGORITHM:

1. start
2. Initialize the segment address
3. Initialize the destination address
4. Get the hex decimal byte to AL
5. Set AH to 0
6. Divide the number by 100
7. Now 100's are in AL move it to CH
8. Get the remainder to AL
9. Set AH to 0
10. Now divide the number by 10
11. Now 10's are in AL
12. Units are in AH
13. Position is in AH to lower nibble and it with F0
14. Now AL will contain 10's and units & get 100's to AH from CH
15. Now the result in AH,AL
16. End of program

INPUT :- 4100 : 52 (Hex Equivalent)

OUTPUT:- AX: 143 (Decimal Number)

RESULT:

Thus conversion of Hex number to decimal number performed by 8086 MP

PROG 11 (B)	**Program to convert decimal number to Hexa decimal number**	

AIM:
Program to convert given decimal number to hexa decimal number

APPARATUS:

ALS-86 µp kit.
Key board
Power supply

ALGORITHM:

10. start
11. Initialize the segment address
12. Initialize the destination address
13. Get the data byte into AL
14. Get the 10 digit to lower
15. Nibble of AL, most with units
16. Multiply by 10's digit by 10
17. Add unit to it
18. End of program

INPUT :- 4100 : 52 (Dec Number)

OUTPUT:- **AL** : 34 (Hex Equivalent)

RESULT:
Thus conversion of decimal number to Hex number performed by 8086 microprocessor

PROG 12	**Program to compute the logical 1's in a word**	

AIM:

Program to compute the logical 1's in a wordusing 8086 assembly language

APPARATUS:

ALS-86 µp kit.
Key board
Power supply

ALGORITHM:

1. start
2. Initialize the segment address
3. Initialize the destination address
4. Get the data words to AX
5. Initializes count location to 0
6. Set counter to 16
7. Shift AX to check last bit
8. If it not 1 jump out and if it is 1 increment counter
9. Repeat for all 16 bits
10. End of the program

Input : 4100 : 32 00

Output : 4102 : 03

RESULT:

Thus number of one's are counted in a given word using 8086 microprocessor program

PROG 13	**Program to convert binary to ASCII**	8086 Kit program

AIM: Program to convert given binary number to ASCII using 8086 microprocessor

APPARATUS:

ALS-86 µp kit.
Key board
Power supply

ALGORITHM:

1. start
2. Get the AL value from 4100 memory location load the same in AH.
3. Mask with higher nibble
4. Call convert to ASCII routine
5. Load AL from AH and initialize the counter with 4
6. Rotate AL for 4 times(getting higher nibble)
7. Mask with higher nibble
8. Call convert to ASCII routine
9. End of the program

Input : 4100 : 3A

Output : AL : 33 AH : 41

RESULT:

The given binary number is converted into ASCII value using 8086 microprocessor.

PROG 1A	1. (A) ADDITION OF TWO 8-BIT NUMBERS	

AIM: To write an assembly language program to perform addition of two 8-bit signed and unsigned numbers

RESOURCES REQUIERED:

- ✓ Personal Computer
- ✓ MASM/TASM Software Installed

Flags Affected: AF, CF, OF, PF, SF, ZF

Calculations:

SYMBOL	ADDRESS	VALUES1	VALUES2
X	0000	12	FE
Y	0001	56	03

X = 12H 0001 0010
Y = 56H 0101 0110

X+Y 0110 1000 = 68H CF=0

X = #FEH 1111 1110
Y = #03H 0000 0011

X+Y 0000 0001 = 01H CF=1, AF=1

Result :

Thus the program for addition of two bytes has been executed successfully by using MASM & result is verified

PROG 1B	**SUBTRACTION OF TWO 8-BIT NUMBERS**	MASM

AIM: To write an assembly language program to perform subtraction of two 8-bit signed and unsigned numbers

RESOURCES REQUIERED:

- ✓ Personal Computer
- ✓ MASM/TASM Software Installed

Flags Affected: AF, CF, OF, PF, SF, ZF

Calculations:

SYMBOL	ADDRESS	VALUES1	VALUES2
X	0000	7D	02
Y	0001	56	03

X = #7DH 0111 1101
Y = 56H 0101 0110

X-Y 0010 0111 = 27H CF=0

X = #02H 0000 0010
Y = #03H 0000 0011

X-Y 1111 1111 = #FFH CF=1, AF=1, PF=1, SF=1

Result :

Thus the program for addition of two bytes has been executed successfully by using masm & result is verified

PROG 2A	ADDITION OF TWO 16-BIT NUMBERS	

AIM: To write an assembly language program to perform addition of two 16-bit signed and unsigned numbers

RESOURCES REQUIERED:

- ✓ Personal Computer
- ✓ MASM/TASM Software Installed

FLAGS which may affect: AF, CF, OF, PF, SF, ZF

CALCULATIONS:

SYMBOL	ADDRESS	VALUES1	VALUES2
X	0000	1234	FDFE
Y	0002	5678	1234
RES	0004		

X = 1234H 0001 0010 0011 0100
Y = 5678H 0101 0110 0111 1000
--- -

X+Y= 68AC 110100010101100

Result :
Thus the program for addition of two words has been executed successfully by using MASM & result is verified

PROG 2B	SUBTRACTION OF TWO 16-BIT NUMBERS	

AIM: To write an assembly language program to perform subtraction of two 16-bit signed and unsigned numbers

RESOURCES REQUIERED:

- ✓ Personal Computer
- ✓ MASM/TASM Software Installed

FLAGS which may affect: AF, CF, OF, PF, SF, ZF

Calculations:

SYMBOL	ADDRESS	VALUES1	VALUES2
X	0000	3322	1ABC
Y	0002	2211	8E12
RES	0004		

X = 3322 11001100100010
Y = 2211 10001000010001

X+Y= 1111 1000100010001

Result

Thus the program for subtraction of two words has been executed successfully by using MASM & result is verified.

PROG 3A	MULTIPLICATION OF TWO 8 -BIT NUMBERS	MASM

AIM: To write an assembly language program to perform multiplication of two 8-bit unsigned numbers

RESOURCES REQUIERED:
- ✓ Personal Computer
- ✓ MASM/TASM Software Installed

FLAGS AFFECTED: OF, CF
CALCULATIONS:

SYMBOL	ADDRESS	VALUES1	VALUES2
X	0000	12	FF
Y	0001	34	AA
RES	0002	03A8	A956

Result

Thus the program for multiplication of two 8-bit program executed successfully by using TASM & result is verified.

PROG 3B	MULTIPLICATION OF TWO 16 -BIT NUMBERS	MASM

AIM: To write an assembly language program to perform multiplication of two 16-bit unsigned numbers

RESOURCES REQUIERED:
- ✓ Personal Computer
- ✓ MASM/TASM Software Installed

FLAGS AFFECTED: OF, CF
CALCULATIONS:

SYMBOL	ADDRESS	VALUES1	VALUES2
X	0000	1234	1111
Y	0002	5678	1111
RESLW	0004	0060	4321
RESHW	0006	0626	0123

Result :

Thus the program for multiplication of two 16-bit program executed successfully by using TASM & result is verified.

PROG 4A	**DIVISION OF 16-BIT Number by an 8 -BIT NUMBER**	MASM

AIM: To write an assembly language program to perform division of 16-bit unsigned number by 8-bit unsigned number

RESOURCES REQUIERED:
- ✓ Personal Computer
- ✓ MASM/TASM Software Installed

FLAGS AFFECTED: IF

CALCULATIONS:

SYMBOL	ADDRESS	VALUES1	VALUES2
X	0000	014D	1200
Y	0002	34	55
Q	0003		
R	0004		

Result:

Thus the 16 bit by 8 bit division of program executed successfully by using MASM & result is verified.

PROG 4B	DIVISION OF 32-BIT Number by a 16-BIT NUMBER	MASM

AIM: To write an assembly language program to perform division of 32-bit unsigned number by 16-bit unsigned number.

RESOURCES REQUIERED:

- ✓ Personal Computer
- ✓ MASM/TASM Software Installed

FLAGS AFFECTED: IF

CALCULATIONS:

SYMBOL	ADDRESS	VALUES1	VALUES2
DLW	0000	0032	0BCD
DHW	0002	02AB	00AB
Divisr	0004	2012	ABCD
Q	0006		
R	0008		

Result :

Thus the 32 bit by 16 bit division of program executed successfully by using MASM & result is verified.

PROG 5A	**Signed Multiplication**	MASM

AIM: To write an assembly language program to perform multiplication of two 16-bit signed numbers

RESOURCES REQUIERED:

- ✓ Personal Computer
- ✓ MASM/TASM Software Installed

FLAGS AFFECTED: OF,CF

CALCULATIONS:

SYMBOL	ADDRESS	VALUES1	VALUES2
X	0000		
Y	0002		
RESLW	0004		
RESHW	0006		

Case 1:Two Positive Numbers
X = 7593, Y = 6845
Case 2: **one positive number & one negative number**
X = 8A6D, Y = 6845
Case 3: **two negative numbers**
X = 8A6D, Y = 97BB

Result: thus the multiplication of two 16-bit signed numbers performed using MASM & results verified

PROG 5B	Signed Division	MASM

AIM: To write an assembly language program to perform Division of 16-bit signed number by 8-Bit Signed Number

RESOURCES REQUIERED:
- ✓ Personal Computer
- ✓ MASM/TASM Software Installed

FLAGS AFFECTED: IF

CALCULATIONS:

SYMBOL	ADDRESS	VALUES1	VALUES2
X	0000		
Y	0002		
Q	0003		
R	0004		

Case 1:Two Positive Numbers
X = 26F8, Y = 56
Case 2: **one negative number & one positive number**
X = D908, Y = 56
Case 3: **one positive number & one negative number** X = 26F8, Y = AA

Result: Thus the division 16-bit by 8-bit signed numbers are performed using MASM & results verified

PROG 6A	ASCII ADDITION	MASM

AIM: To write an ALP to perform the addition of two ASCII bytes.

RESOURCES REQUIERED:

- ✓ Personal Computer
- ✓ MASM/TASM Software Installed

FLAGS AFFECTED: AF, CF

CALCULATIONS:

SYMBOL	ADDRESS	VALUES1	VALUES2
CHAR1	0000	35h	
CHAR2	0001	37h	
RES(AX)	0002	0102h	

RES = 0102 (AX) unpacked BCD of 12

Result :
Thus the program for AAA has been executed successfully by using MASM & result is verified.

PROG 6B	ASCII SUBTRACTION	MASM

AIM: To write an ALP to perform the subtraction of two ASCII bytes.

RESOURCES REQUIERED:
- ✓ Personal Computer
- ✓ MASM/TASM Software Installed

FLAGS AFFECTED: AF,CF

CALCULATIONS:

SYMBOL	ADDRESS	VALUES1	VALUES2
CHAR1	0000	‘37h’	
CHAR2	0001	‘35h’	
RES	0002	02	
	0003	00	

‘37’ – ‘35’ = 02
RES= 0002H

Result

Thus the program for AAS has been executed successfully by using MASM & result is verified.

PROG 6C	ASCII MULTIPLICATION	MASM

AIM: To write an ALP to perform the multiplication of two ASCII bytes.

RESOURCES REQUIERED:

- ✓ Personal Computer
- ✓ MASM/TASM Software Installed

FLAGS AFFECTED: PF,SF,ZF

CALCULATIONS:

SYMBOL	ADDRESS	VALUES1	VALUES2
NUM1	0000	09	
NUM2	0001	05	
RES	0002	05	
	0003	04	

09 * 05 =
RES = 0405 (AX) unpacked BCD of 45

Result:

Thus the program for AAM has been executed successfully by using MASM & result is verified

PROG 6D	**ASCII DIVISION**	MASM

AIM: To write an ALP to perform the division of two ASCII numbers.

RESOURCES REQUIERED:
- ✓ Personal Computer
- ✓ MASM/TASM Software Installed

FLAGS AFFECTED: PF,SF,ZF

CALCULATIONS:

SYMBOL	ADDRESS	VALUES1	VALUES2
DIVIDEND	0000	0607	
DIVISOR	0002	09	
RESQ	0003	07	
RESR	0004	04	

INPUT:DIVIDEND = 0607H unpacked BCD of 67
DIVISOR = 09H
OUTPUT: RESQ = 07 (AL)
RESR = 04 (AH)

Result :
Thus the program for AAD has been executed successfully by using TASM & result is verified.

PROG 7	PACKED BCD TO UNPACKED BCD	MASM

AIM: To write an assembly language program to perform PACKED BCD into UNPACKED BCD conversion.

RESOURCES REQUIERED:

- ✓ Personal Computer
- ✓ MASM/TASM Software Installed

FLAGS AFFECTED: PF

CALCULATIONS:

SYMBOL	ADDRESS	VALUES1	VALUES2
BCD	0000	49	
UBCD1	0002		
UBCD2	0003		

INPUT: BCD = 49

OUTPUT: UBCD1 = 09 UBCD2 = 04

Result :

Thus packed BCD converted into unpacked BCD using MASM & results verified

PROG 8	BCD TO ASCII CONVERSION	MASM

AIM: To write an assembly language program to perform BCD to ASCII conversion.

RESOURCES REQUIERED:

- ✓ Personal Computer
- ✓ MASM/TASM Software Installed

FLAGS AFFECTED: PF

CALCULATIONS:

SYMBOL	ADDRESS	VALUES1	VALUES2
BCD	0000	49	
ASCII1	0002		
ASCII2	0003		

INPUT: BCD = 49

OUTPUT: ASCII1= 39 ASCII2= 34

Result :
Thus the given BCD number converted into ASCII using MASM & results verified.

PROG 9A	DECIMAL ADJUSTMENT AFTER ADDITION	MASM

AIM: To write an assembly language program to perform DECIMAL adjust after addition of two 8-bit signed and unsigned numbers

RESOURCES REQUIERED:

- ✓ Personal Computer
- ✓ MASM/TASM Software Installed

Program:

Result :

Thus the program for DAA has been executed successfully by using MASM & result is verified.

PROG 9B	DECIMAL ADJUSTMENT AFTER SUBTRACTION	MASM

AIM: To write an assembly language program to perform DECIMAL adjust after subtraction of two 8-bit signed and unsigned numbers

RESOURCES REQUIERED:

- ✓ Personal Computer
- ✓ MASM/TASM Software Installed

Program:

Result :

Thus the program for DAS has been executed successfully by using MASM & result is verified.

PROG 10 A	BLOCK MOVE	MASM

AIM: To write an assembly language program to move the block of data from a specified source location to the specified destination location.

RESOURCES REQUIERED:

- ✓ Personal Computer
- ✓ MASM/TASM Software Installed

REGISTERS USED: AX, DS, ES, SI, DI

FLAGS AFFECTED: No flags are affected

RESULT:

INPUT: STR (DS:2000H) = 04H,F9H,BCH,98H,40H

OUTPUT: STR1(ES:3000H) = 04H,F9H,BCH,98H,40H

Results:

Thus the block of data moved from a specified source location to the specified destination location using MASM 7 results verified.

PROG 10 B	STRING REVERSING: for any Known Character Length	MASM

AIM: To write an assembly language program to reverse the given string.

RESOURCES REQUIERED:

- ✓ Personal Computer
- ✓ MASM/TASM Software Installed

REGISTERS USED: AX, DS, SI, BX

FLAGS AFFECTED: ZF, PF

RESULT:

INPUT: STR (DS:0000H) = 01H,#02H,#03H,04H

OUTPUT: STR (DS:0000H) = 04H,#03H,#02H,01H

Result :

Thus the given string is reversed using MASM & results verified.

PROG 10 C	LENGTH OF THE STRING	MASM

AIM: To write an assembly language program to find the length of the given string.

RESOURCES REQUIERED:

- ✓ Personal Computer
- ✓ MASM/TASM Software Installed

REGISTERS USED: AX,DS,SI,CL

FLAGS AFFECTED: ZF,PF,SF,AF,CF

RESULT:

INPUT: STR (DS:0000H) =
01H, #03H,08H,09H,05H,07H,#02H

OUTPUT: LENGTH = 07H (CL)

Result :

Thus the length of the given string is identified using MASM software & results are verfified.

PROG 10 D	STRING COMPARISION	MASM

AIM: To write an assembly language program to compare the given strings.

RESOURCES REQUIERED:

- ✓ Personal Computer
- ✓ MASM/TASM Software Installed

REGISTERS USED: AX, DS, SI, DI, CL

FLAGS AFFECTED: ZF,CF

RESULT:

INPUT: SRC (DS:0000H) =
04H,05H,07H,08H
DST (ES:0000H) =
04H,06H,07H,09H

OUTPUT:
I): IF SRC = DST THEN ZF = 1 &
II): IF SRC ≠ DST THEN ZF = 0

Result :
The given two strings are compared using MASM software and results are verified.

PROG 10 E	STRING INSERTION	MASM

AIM: To write an Assembly Language Program for inserting one string into the other.

RESOURCES REQUIERED:

- ✓ Personal Computer
- ✓ MASM/TASM Software Installed

FLAGS AFFECTED: No flags are affected

OUTPUT: new string

Result:
Thus sting insertion program executed sucessfully using MASM software and results are verfied.

PROG 10 F	STRING DELETION	MASM

AIM: To write an Assembly Language Program for deleting a string in specified index range

RESOURCES REQUIERED:

- ✓ Personal Computer
- ✓ MASM/TASM Software Installed

FLAGS AFFECTED: No flags are affected

RESULT:

INPUT: computer

OUTPUT: couter

.

Results :

thus the string deletion program executed successfully using MASM software and results verified.

PROG 10 G	DISPLAY THE STRING	MASM

AIM: To write an assembly language program to display the given string.

RESOURCES REQUIERED:

- ✓ Personal Computer
- ✓ MASM/TASM Software Installed

REGISTERS USED: AX, DS, DX

FLAGS AFFECTED: No flags are affected

output : WELCOME TO MICROPROCESSORS LAB

Result:
Thus the program to display the given string executed successfully using MASM & results verified.

PROG 1	GENERATION OF POSITIVE RAMP SIGNAL	8086 Interfacing

AIM: To write an assembly language program to generate a Positive ramp signal using 8086 microprocessor.

RESOURCES REQUIERED: 8086 Microprocessor trainer kit,
Dual DAC kit,
CRO.

RESULT: Positive ramp signal is generated using 8086 trainer kit.

PROG 2	GENERATION OF NEGATIVE RAMP SIGNAL	8086 Interfacing

AIM: To write an assembly language program to generate a negative ramp signal using 8086 trainer kit.

RESOURCES REQUIERED: 8086 Microprocessor trainer kit,
Dual DAC kit,
CRO.

RESULT: Negative ramp signal is generated using 8086 trainer kit.

PROG 3	SQUARE WAVE GENERATION	8086 Interfacing

AIM: To write an assembly language program to generate a Square wave using 8086 trainer kit.

RESOURCES REQUIERED: 8086 Microprocessor trainer kit,
Dual DAC kit,
CRO.

RESULT: Square wave is generated using 8086 trainer kit.

PROG 4	GENERATION OF TRIANGULAR WAVE	8086 Interfacing

AIM: To write an assembly language program to generate a triangular wave using 8086 trainer kit.

RESOURCES REQUIERED: 8086 Microprocessor trainer kit,
Dual DAC kit,
CRO.

.

RESULT: Triangular wave is generated using 8086 trainer kit.

PROG 5	GENERATION OF SINE WAVE	8086 Interfacing

AIM: To write an assembly language program to generate a sine wave using 8086 trainer kit.

RESOURCES REQUIERED: 8086 Microprocessor trainer kit,
Dual DAC kit,
CRO.

FORMULA USED

128+128 SIN X   Where X=0 T0 360 with Step 5

LOOK UP TABLE

4000	80
4001	8B
4002	96
4003	A1
4004	AB
4005	B6
4006	C0
4007	C9
4008	D2
4009	DA
400A	E2
400B	E8
400C	EF
400D	F4
400E	F8
400F	FB
4010	FE
4011	FF
4012	FF
4026	74
4027	6A
4028	5F
4029	54
402A	4A
402B	40
402C	36
402D	2E
402E	25
402F	1E
4030	17
4031	11
4032	0C
4033	08
4034	04
4035	02
4036	00
4037	00

4013	FF
4014	FF
4015	FE
4016	FB
4017	F8
4018	F4
4019	EF
401A	E8
401B	E2
401C	DA
401D	D2
401E	C9
401F	C0
4020	B6
4021	AB
4022	A1
4023	96
4024	8B
4025	80
4038	00
4039	02
403A	04
403B	08
403C	0C
403D	11
403E	17
403F	1E
4040	25
4041	2E
4042	36
4043	40
4044	4A
4045	54
4046	5F
4047	6A
4048	74
4049	80

RESULT: sine wave is generated using 8086 trainer kit.

PROG 6	STEPPER MOTOR CONTROL	8086 Interfacing

AIM: To Interface Stepper Motor to 8086 using 8255 and write Assembly Language Program to rotate Stepper Motor in Anticlockwise direction in full stepping.

RESOURCES REQUIRED: 8086 trainer kit**,**
8225 microprocessor Interface,
Stepper Motor,
Power Supply.

RESULT: Stepper motor is interfaced with 8086 using 8255 and operation is verified.

PROG 7	TRAFFIC LIGHT CONTROL	8086 Interfacing

AIM: To develop Traffic light Control system using 8086

RESOURCES REQUIRED: Microprocessor trainer kit,
Traffic light controller kit,
power supply,
data cable etc

```
DEC BX
JNZ D1
INT #03H
```

RESULT: Traffic light Control system is developed using 8086.

PROG 1	READING AND WRITING ON A PARALLEL PORT OF 8051	8051 Microcontrollers

AIM:

To write an assembly language program to perform the operations like reading and writing the data on a parallel port of 8051 microcontroller.

RESOURCES REQUIRED: 8051 Micro Controller kit.
Key Board
Adapter.

RESULT: Thus the operations like reading and writing the data on a parallel port performed using 8051 microcontroller.

PROG 2	8051 TIMERS	8051 Microcontrollers

AIM: To perform the 8051 TIMER functionality using 8051 microcontroller

RESOURCES REQUIERED: 8051 microcontroller kit
Keyboard
Power supply

RESULT: Thus 8051 timer functionality is verified using 8051 microcontroller.

PROG 3	BIT AND BYTE OPERATIONS BY USING 8051	8051 Microcontrollers

AIM: To write an assembly language program to perform the BIT and BYTE operations like set, reset and swap by using 8051 microcontroller.

RESOURCES REQUIERED: 8051 microcontroller kit
Keyboard
Power supply

RESULT: Thus 8051 timer functionality is verified using 8051 microcontroller.

PROG 4	**Understanding three memory areas of 00-FF**	**8051 Microcontrollers**

AIM: To Understanding three memory areas of 00-FF using 8051 microcontroller

RESOURCES REQUIERED: 8051 microcontroller kit
Keyboard
Power supply

BIT ADDRESSABLE AREA FROM 0-2F

ORG 9000H

```
          SETB 00H
          SETB 01H
          SETB 02H
          JB 50H, DIS_B5
          MOV R6,#00H
          LJMP END2
DIS_BS : MOV R6.#B5H
END2    : LCALL 003
```

RESULT: Thus 8051 timer functionality is verified using 8051 microcontroller.

PROG 5	2'S COMPLIMENT	8051 Microcontrollers

AIM: To perform the 2's compliment operation by using 8051 micro controller

RESOURCES REQUIERED: 8051 microcontroller kit
Keyboard
Power supply

RESULT: 2's compliment operation is done

PROG 6	**8-BIT SUBTRACTION**	**8051 Microcontrollers**

AIM: To perform subtraction of two 8-bit numbers stored at some location. Store the result in next location and in next location store 00H or 01H for positive and negative numbers respectively

RESOURCES REQUIERED: 8051 microcontroller kit
Keyboard
Power supply

RESULT:

8-bit subtraction is done by 8051 kit

PROG 7	8-BIT DIVISION	8051 Microcontrollers

AIM: To perform the division operation of two 8-bit numbers using 8051 microcontroller

RESOURCES REQUIERED: 8051 microcontroller kit
Keyboard
Power supply

RESULT: Division of two 8-bit numbers is done by using 8051 microcontroller kit

PROG 8	BINARY TO BCD CONVERSION	8051 Microcontrollers

AIM: To convert a Binary number into a BCD number using 8051 micro controller

RESOURCES REQUIERED: 8051 microcontroller kit
Keyboard
Power supply

RESULT: Binary to BCD conversion of given number is done by 8051 kit

PROG 9	SUM 0F N-NUMBER	8051 Microcontrollers

AIM: To perform sum of N natural numbers by using 8051 microcontroller

RESOURCES REQUIERED: 8051 microcontroller kit
Keyboard
Power supply

RESULT:
Sum of first four natural numbers is done with 8051 kit.

PROG 10	SMALLEST NUMBER	8051 Microcontrollers

AIM: To perform an operation of finding smallest number among given data by using 8051 micro controller

RESOURCES REQUIERED: 8051 microcontroller kit
Keyboard
Power supply

RESULT: Among the given data smallest number found using 8051 kit

PROG 11	BIGGEST NUMBER	**8051 Microcontrollers**

AIM: To perform an operation of finding biggest number among given data by using 8051 micro controller

RESOURCES REQUIERED: 8051 microcontroller kit
Keyboard
Power supply

RESULT: Among the given data biggest number found using 8051 kit

PROG 12	BCD TO ASCII COVERSION	8051 Microcontrollers

AIM: To convert BCD number into ASCII by using 8051 micro controller

RESOURCES REQUIERED: 8051 microcontroller kit
Keyboard
Power supply

RESULT:
The given number is converted into ASCII using 8051 microcontroller kit.

PROG 13	GENERATION OF 00 TO 0F NUMBERS	8051 Microcontrollers

AIM: To generate a series of numbers from 00h to 0Fh by using 8051 micro controller

RESOURCES REQUIERED: 8051 microcontroller kit
Keyboard
Power supply

RESULT: Numbers from 00 to 0F are generated using 8051 microcontroller kit.

DSP PROCESSORS

EXP :1	ARCHITECTURE OF C6713 DSP PROCESSOR	

AIM: To study the architecture of C6713 DSP Processor

EQUIPMENTS:

PC with 6713 CC studio 3.0 software
DSP C6713 Kit.
USB connecting cable.
Power Cord.

Theory:

A signal can be defined as a function that conveys information, generally about the state or behavior of a physical system. There are two basic types of signals viz Analog (continuous time signals which are defined along a continuum of times) and Digital (discrete-time).

Remarkably, under reasonable constraints, a continuous time signal can be adequately represented by samples, obtaining discrete time signals. Thus digital signal processing is an ideal choice for anyone who needs the performance advantage of digital manipulation along with today's analog reality.

Hence a processor which is designed to perform the special operations(digital manipulations) on the digital signal within very less time can be called as a Digital signal processor. The difference between a DSP processor, conventional microprocessor and a microcontroller are listed below.

Microprocessor or General Purpose Processor such as Intel xx86 or Motorola 680xx family

Contains - only CPU
-No RAM
-No ROM
-No I/O ports
-No Timer

Microcontroller such as 8051 family

Contains - CPU
- RAM
- ROM
-I/O ports
- Timer &
- Interrupt circuitry

Some Micro Controllers also contain A/D, D/A and Flash Memory

DSP Processors such as Texas instruments and Analog Devices

Contains - CPU
- RAM
- ROM
- I/O ports
- Timer

Optimized for – fast arithmetic
- Extended precision
- Dual operand fetch
- Zero overhead loop
- Circular buffering

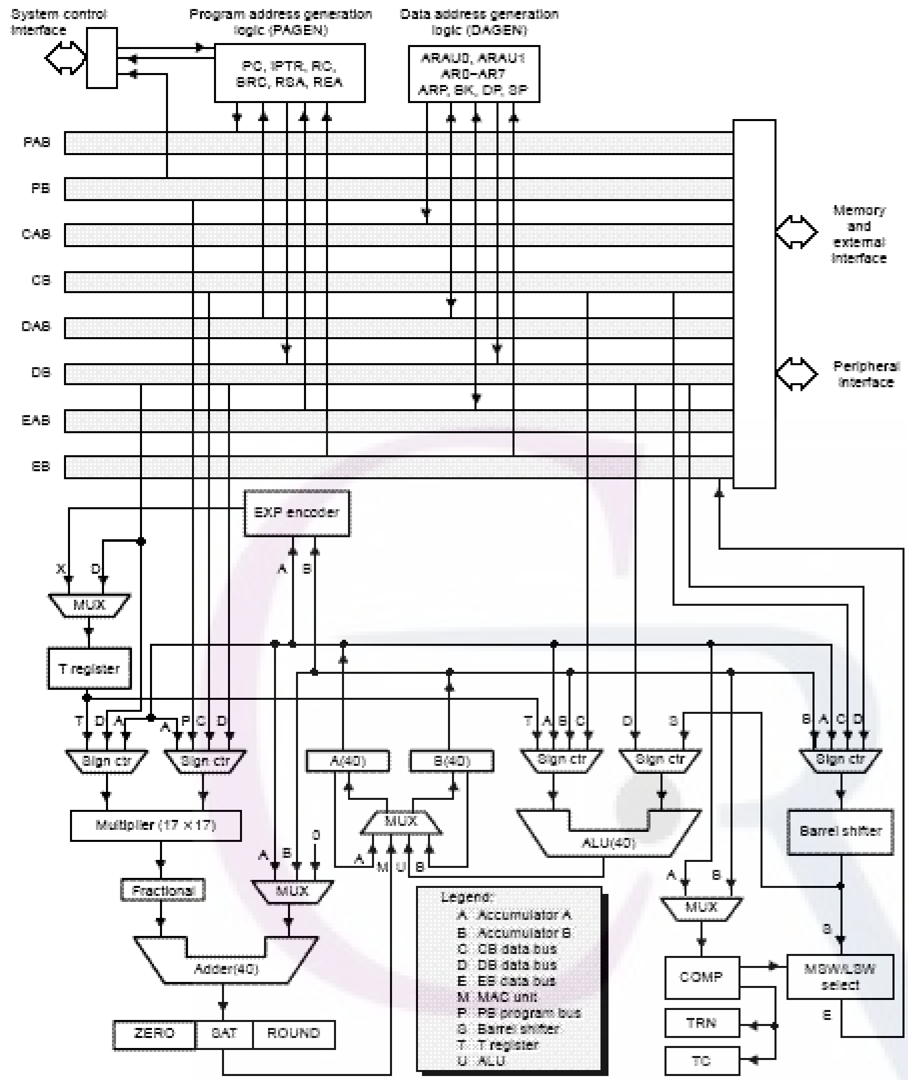
The basic features of a DSP Processor are

Feature	Use
Fast-Multiply accumulate	Most DSP algorithms, including filtering, transforms, etc. are multiplication- intensive
Multiple – access memory architecture	Many data-intensive DSP operations require reading a program instruction and multiple data items during each instruction cycle for best performance
Specialized addressing modes	Efficient handling of data arrays and first-in, first-out buffers in memory
Specialized program control	Efficient control of loops for many iterative DSP algorithms. Fast interrupt handling for frequent I/O operations.
On-chip peripherals and I/O interfaces	On-chip peripherals like A/D converters allow for small low cost system designs. Similarly I/O interfaces tailored for common peripherals allow clean interfaces to off-chip I/O devices.

ARCHITECTURE OF 6713 DSP PROCESSOR

This chapter provides an overview of the architectural structure of the TMS320C67xx DSP, which comprises the central processing unit (CPU), memory, and on-chip peripherals. The C67xE DSPs use an advanced modified Harvard architecture that maximizes processing power with eight buses. Separate program and data spaces allow simultaneous access to program instructions and data, providing a high degree of parallelism. For example, three reads and one write can be performed in a single cycle. Instructions with parallel store and application-specific instructions fully utilize this architecture. In addition, data can be transferred between data and program spaces. Such Parallelism supports a powerful set of arithmetic, logic, and bit-manipulation operations that can all be performed in a single machine cycle. Also, the C67xx DSP includes the control mechanisms to manage interrupts, repeated operations, and function calling.

Fig :BLOCK DIAGRAM OF TMS 320VC 6713



Bus Structure

The C67xx □DSP architecture is built around eight major 16-bit buses (four program/data buses and four address buses):

_ The program bus (PB) carries the instruction code and immediate operands from program memory.

_ Three data buses (CB, DB, and EB) interconnect to various elements, such as the CPU, data address generation logic, program address generation logic, on-chip peripherals, and data memory.

_ The CB and DB carry the operands that are read from data memory.

_ The EB carries the data to be written to memory.

_ Four address buses (PAB, CAB, DAB, and EAB) carry the addresses needed for instruction execution.

The C67xx DSP can generate up to two data-memory addresses per cycle using the two auxiliary register arithmetic units (ARAU0 and ARAU1). The PB can carry data operands stored in program space (for instance, a coefficient table) to the multiplier and adder for multiply/accumulate operations or to a destination in data space for data move instructions (MVPD and READA). This capability, in conjunction with the feature of dual-operand read, supports the execution of single-cycle, 3-operand instructions such as the FIRS instruction. The C67xx DSP also has an on-chip bidirectional bus for accessing on-chip peripherals. This bus is connected to DB and EB through the bus exchanger in the CPU interface. Accesses that use this bus can require two or more cycles for reads and writes, depending on the peripheral's structure.

Central Processing Unit (CPU)

The CPU is common to all C67xE devices. The C67x CPU contains:

_ 40-bit arithmetic logic unit (ALU)
_ Two 40-bit accumulators
_ Barrel shifter
_ 17 × 17-bit multiplier
_ 40-bit adder
_ Compare, select, and store unit (CSSU)
_ Data address generation unit
_ Program address generation unit

Arithmetic Logic Unit (ALU)

The C67x DSP performs 2s-complement arithmetic with a 40-bit arithmetic logic unit (ALU) and two 40-bit accumulators (accumulators A and B). The ALU can also perform Boolean operations. The ALU uses these inputs:

_ 16-bit immediate value
_ 16-bit word from data memory
_ 16-bit value in the temporary register, T
_ Two 16-bit words from data memory
_ 32-bit word from data memory
_ 40-bit word from either accumulator

The ALU can also function as two 16-bit ALUs and perform two 16-bit operations simultaneously.

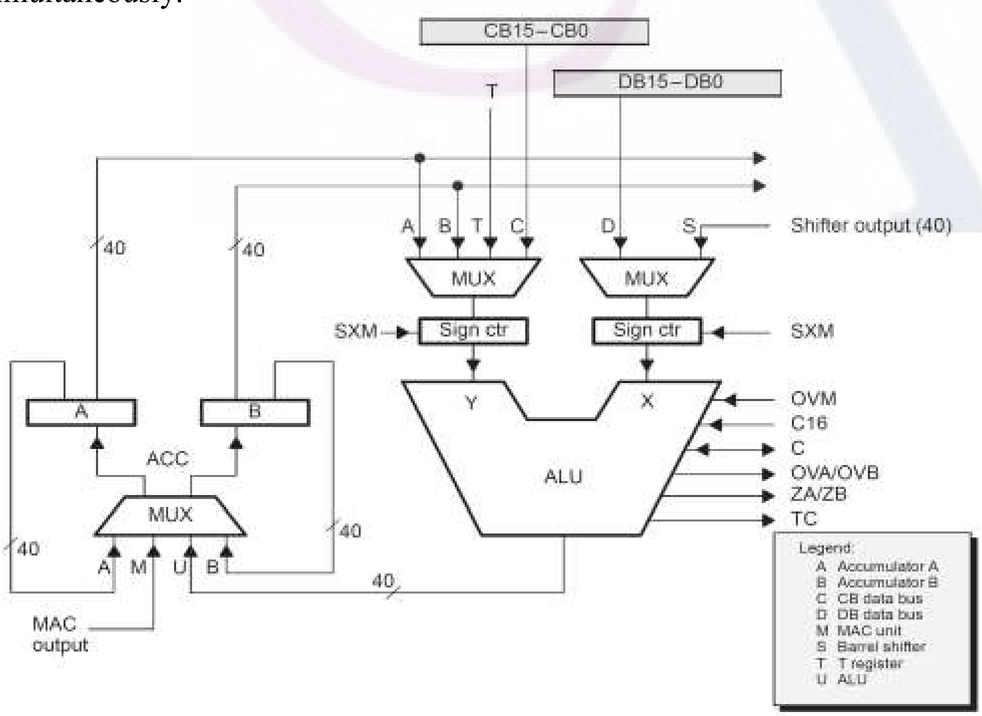


Fig :ALU UNIT

Accumulators

Accumulators A and B store the output from the ALU or the multiplier/adder block. They can also provide a second input to the ALU; accumulator A can be an input to the multiplier/adder. Each accumulator is divided into three parts:

_ Guard bits (bits 39–32)
_ High-order word (bits 31–16)
_ Low-order word (bits 15–0)

Instructions are provided for storing the guard bits, for storing the high- and the low-order accumulator words in data memory, and for transferring 32-bit accumulator words in or out of data memory. Also, either of the accumulators can be used as temporary storage for the other.

Barrel Shifter

The C67x DSP barrel shifter has a 40-bit input connected to the accumulators or to data memory (using CB or DB), and a 40-bit output connected to the ALU or to data memory (using EB). The barrel shifter can produce a left shift of 0 to 31 bits and a right shift of 0 to 16 bits on the input data. The shift requirements are defined in the shift count field of the instruction, the shift count field (ASM) of status register ST1, or in temporary register T (when it is designated as a shift count register).The barrel shifter and the exponent encoder normalize the values in an accumulator in a single cycle. The LSBs of the output are filled with 0s, and the MSBs can be either zero filled or sign extended, depending on the state of the sign-extension mode bit (SXM) in ST1. Additional shift capabilities enable the processor to perform numerical scaling, bit extraction, extended arithmetic, and overflow prevention operations.

Multiplier/Adder Unit

The multiplier/adder unit performs 17 _ 17-bit 2s-complement multiplication with a 40-bit addition in a single instruction cycle. The multiplier/adder block consists of several elements: a multiplier, an adder, signed/unsigned input control logic, fractional control logic, a zero detector, a rounder (2s complement), overflow/saturation logic, and a 16-bit temporary storage register (T). The multiplier has two inputs: one input is selected from T, a data-memory operand, or accumulator A; the other is selected from program memory, data memory, accumulator A, or an immediate value. The fast, on-chip multiplier allows the C54x DSP to perform operations efficiently such as convolution, correlation, and filtering. In addition, the multiplier and ALU together execute multiply/accumulate (MAC) computations and ALU operations in parallel in a single instruction cycle. This function is used in determining the Euclidian distance and in implementing symmetrical and LMS filters, which are required for complex DSP algorithms.

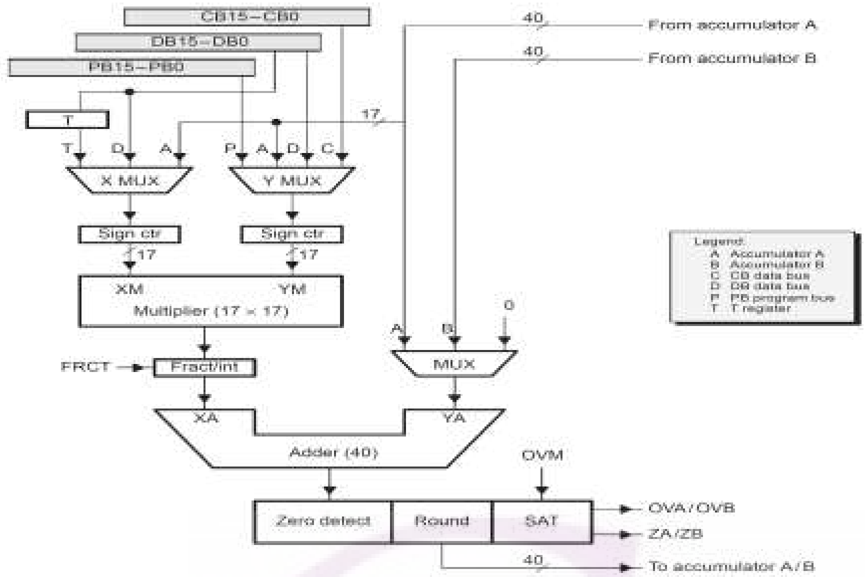


Fig :MULTIPLIER/ADDER UNIT

These are the some of the important parts of the processor and you are instructed to go through the detailed architecture once which helps you in developing the optimized code for the required application.

Result:

The architecture of C6713 DSP Processor is studied.

EXP :2	LINEAR CONVOLUTION USING “C”	

AIM: To determine the Linear Convolution of the one-dimensional signal by writing ‘c’ program and using DSP Kit.

EQUIPMENTS:

PC with 6713 CC studio 3.0 software
DSP C6713 Kit.
USB connecting cable.
Power Cord.

THEORY

Convolution is a formal mathematical operation, just as multiplication, addition, and integration. Addition takes two numbers and produces a third number, while convolution takes two signals and produces a third signal. Convolution is used in the mathematics of many fields, such as probability and statistics. In linear systems, convolution is used to describe the relationship between three signals of interest: the input signal, the impulse response, and the output signal.

$$y(n) = \sum_{k=0}^{N-1} x_1(k)x_2(n-k) \quad 0 < n < N-1 \qquad (1)$$

In this equation, x1(k), x2 (n-k) and y(n) represent the input to and output from the system at time n. Here we could see that one of the input is shifted in time by a value everytime it is multiplied with the other input signal. Linear Convolution is quite often used as a method of implementing filters of various types.

PROCEDURE:

1. *Open Code Composer Studio and make sure DSP kit is Switched ON*
2. *Start a new Project using ’new project’ → from project menu icon and save it.*
3. *Write a program for generating sinewave in ‘c’ language and save it with ‘.c’ extension.*
4. *Add runtime support library file ‘rts6700.lib’ to the “library” icon of the project*
5. *Add linear command file ‘Hello.cmd’ to the source icon of the project.*
6. *Compile the program using icon in task bar and make corrections if any errors occur.*
7. *Build in and Build on the project; using icons in the task bar.*
8. *Go to ‘debug’ and double click on ‘connect’ option.*
9. *Go to the file and click on load program and then open the file we find extension ‘.out’ project file.*
10. *Go to debug and run the program.*
11. *To view graphically, select view → graph → time/frequency.*

ALGORITHM

Step 1 Declare three buffers namely Input buffer, Temporary Buffer, Output Buffer.
Step 2 Get the input from the CODEC, store it in Input buffer and transfer it to the first location of the Temporary buffer.
Step 3 Make the Temporary buffer to point to the last location.
Step 4 Multiply the temporary buffer with the coefficients in the data memory and accumulate it with the previous output.
Step 5 Store the output in the output buffer.
Step 6 Repeat the steps from 2 to 5.

Result:
enter value for m**4**
enter value for n**4**
Enter values for i/p
1 2 3 4
Enter Values for n
1 2 3 4

The Value of output y[0]=**1**
The Value of output y[1]=**4**
The Value of output y[2]=**10**
The Value of output y[3]=**20**
The Value of output y[4]=**25**
The Value of output y[5]=**24**
The Value of output y[6]=**16**

Result:

Hence Linear Convolution of the one-dimensional signal is determined by writing 'c' program and using DSP Kit.

EXP :3	CIRCULAR CONVOLUTION USING "C"	

AIM: To determine the Circular Convolution of the one-dimensional signal by writing 'c' program and using DSP Kit.

EQUIPMENTS:

PC with 6713 CC studio 3.0 software
DSP C6713 Kit.
USB connecting cable.
Power Cord.

THEORY

Circular convolution is another way of finding the convolution sum of two input signals. It resembles the linear convolution, except that the sample values of one of the input signals is folded and right shifted before the convolution sum is found. Also note that circular convolution could also be found by taking the DFT of the two input signals and finding the product of the two frequency domain signals. The Inverse DFT of the product would give the output of the signal in the time domain which is the circular convolution output. The two input signals could have been of varying sample lengths. But we take the DFT of higher point, which ever signals levels to. For eg. If one of the signal is of length 256 and the other spans 51 samples, then we could only take 256 point DFT. So the output of IDFT would be containing 256 samples instead of 306 samples, which follows N1+N2 – 1 where N1 & N2 are the lengths 256 and 51 respectively of the two inputs. Thus the output which should have been 306 samples long is fitted into 256 samples. The 256 points end up being a distorted version of the correct signal. This process is called circular convolution.

PROCEDURE:

1. *Open Code Composer Studio and make sure DSP kit is Switched ON*
2. *Start a new Project using 'new project' → from project menu icon and save it.*
3. *Write a program for generating sinewave in 'c' language and save it with '.c' extension.*
4. *Add runtime support library file 'rts6700.lib' to the "library" icon of the project*
5. *Add linear command file 'Hello.cmd' to the source icon of the project.*
6. *Compile the program using icon in task bar and make corrections if any errors occur.*
7. *Build in and Build on the project; using icons in the task bar.*
8. *Go to 'debug' and double click on 'connect' option.*
9. *Go to the file and click on load program and then open the file we find extension '.out' project file.*
10. *Go to debug and run the program.*
11. *To view graphically, select view → graph → time/frequency.*

OUTPUT:-

Enter the first sequence

5

6

7

Enter the second sequence

7

8

5

4

OUTPUT ;- the circular convolution is

94 110 122 106

Result:

```
The Circular Convolution of the one-dimensional signal by writing 'c'
program and using DSP Kit.
```

EXP :4	FFT USING "C"	

AIM: To determine the FFT of the one-dimensional signal by writing 'c' program and using DSP Kit.

EQUIPMENTS:

PC with 6713 CC studio 3.0 software
DSP C6713 Kit.
USB connecting cable.
Power Cord.

THEORY:

The Fast Fourier Transform is useful to map the time-domain sequence into a continuous function of a frequency variable. The FFT of a sequence {x(n)} of length N is given by a complex-valued sequence X(k).

$$X(k) = \sum_{k=0}^{M} x(n) \quad e^{-j2\pi \frac{nk}{n}} ; 0 < k < N-1$$

The above equation is the mathematical representation of the DFT. As the number of computations involved in transforming a N point time domain signal into its corresponding frequency domain signal was found to be N2 complex multiplications, an alternative algorithm involving lesser number of computations is opted.

When the sequence x(n) is divided into 2 sequences and the DFT performed separately, the resulting number of computations would be N2/2

(i.e.)

$$x(k) = \sum_{n=0}^{\frac{N^2}{2-1}} x(2n) \quad W_N^{2nk} + \sum_{n=0}^{\frac{N^2}{2-1}} x(2n+1) \quad W_N^{(2n+1)k}$$

Consider x(2n) be the even sample sequences and x(2n+1) be the odd sample sequence derived form x(n).

$$\sum_{n=0}^{\frac{N^2}{2-1}} x(2n) \quad W_N^{2nk}$$

would result

(N/2)2multiplication's $\sum_{n=0}^{\frac{N^2}{2-1}} x(2n+1) \quad W_N^{(2n+1)k}$

an other $(N/2)^2$ multiplication's finally resulting in $(N/2)^2 + (N/2)^2$

$$= \frac{N^2}{4} + \frac{N^2}{4} = \frac{N^2}{2} \; Computations$$

Further solving Eg. (2)

$$x(k) = \sum_{n=0}^{\frac{N^2}{2-1}} x(2n) \quad W_N^{2nk} + \sum_{n=0}^{\frac{N}{2-1}} x(2n+1) \quad W_N^{(2nk)} \overset{k}{\underset{N}{W}} \quad (9)$$

$$= \sum_{n=0}^{\frac{N}{2-1}} x(2n) \quad W_N^{2nk} + \overset{k}{\underset{N}{W}} \sum_{n=0}^{\frac{N}{2-1}} x(2n+1) \quad W_N^{(2nk)} \quad (10)$$

Dividing the sequence x (2n) into further 2 odd and even sequences would reduce the computations.

$W_N \rightarrow$ is the twiddle factor

$$= e^{\frac{-j2\pi}{n}}$$

$$W_N^{nk} = e^{\left(\frac{-j2\pi}{n}\right)nk}$$

$$W_N^{\left(K+\frac{N}{2}\right)} = W_N \quad W_N^{\left(K+\frac{N}{2}\right)} \qquad (11)$$

$$= e^{\frac{-j2\pi}{n}k} \quad e^{\frac{-j2\pi}{n} \frac{n}{2}}$$

$$= W_N^k \quad e^{\frac{-j2\pi}{n}k}$$

$$= W_N^k \quad (\cos\pi - j\sin\pi)$$

$$= W_N^{\left(K+\frac{N}{2}\right)} = W_N^k(-1)$$

$$= W_N^{\left(K+\frac{N}{2}\right)} = W_N^k \qquad (12)$$

Employing this equation, we deduce

$$x(k) = \sum_{n=0}^{\frac{N^2}{2-1}} x(2n) \quad W_N^{2nk} + \sum_{n=0}^{\frac{N}{2-1}} \ x(2n+1) \quad W_N^{(2nk)} \qquad (13)$$

$$x(k+\frac{N}{2}) = \sum_{n=0}^{\frac{N}{2-1}} x(2n) \quad W_N^{2nk} - W\sum_{N}^{K} \ x(2n+1)^{\frac{N}{2-1}} \quad W_N^{(2nk)} \qquad (14)$$

The time burden created by this large number of computations limits the usefulness of DFT in many applications. Tremendous efforts devoted to develop more efficient ways of computing DFT resulted in the above explained Fast Fourier Transform algorithm. This mathematical shortcut reduces the number of calculations the DFT requires drastically. The above mentioned radix-2 decimation in time FFT is employed for domain transformation.

Dividing the DFT into smaller DFTs is the basis of the FFT. A radix-2 FFT divides the DFT into two smaller DFTs, each of which is divided into smaller DFTs and so on, resulting in a combination of two-point DFTs. The Decimation -In-Time (DIT) FFT divides the input (time) sequence into two groups, one of even samples and the other of odd samples. N/2 point DFT are performed on the these sub-sequences and their outputs are combined to form the N point DFT.

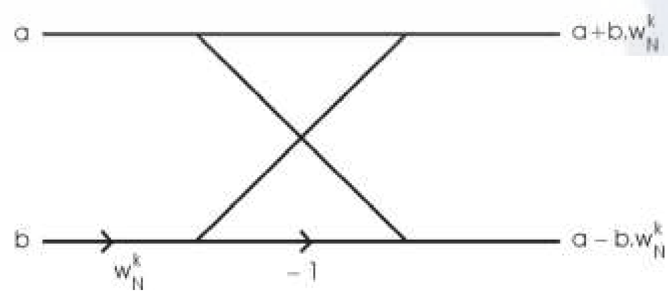
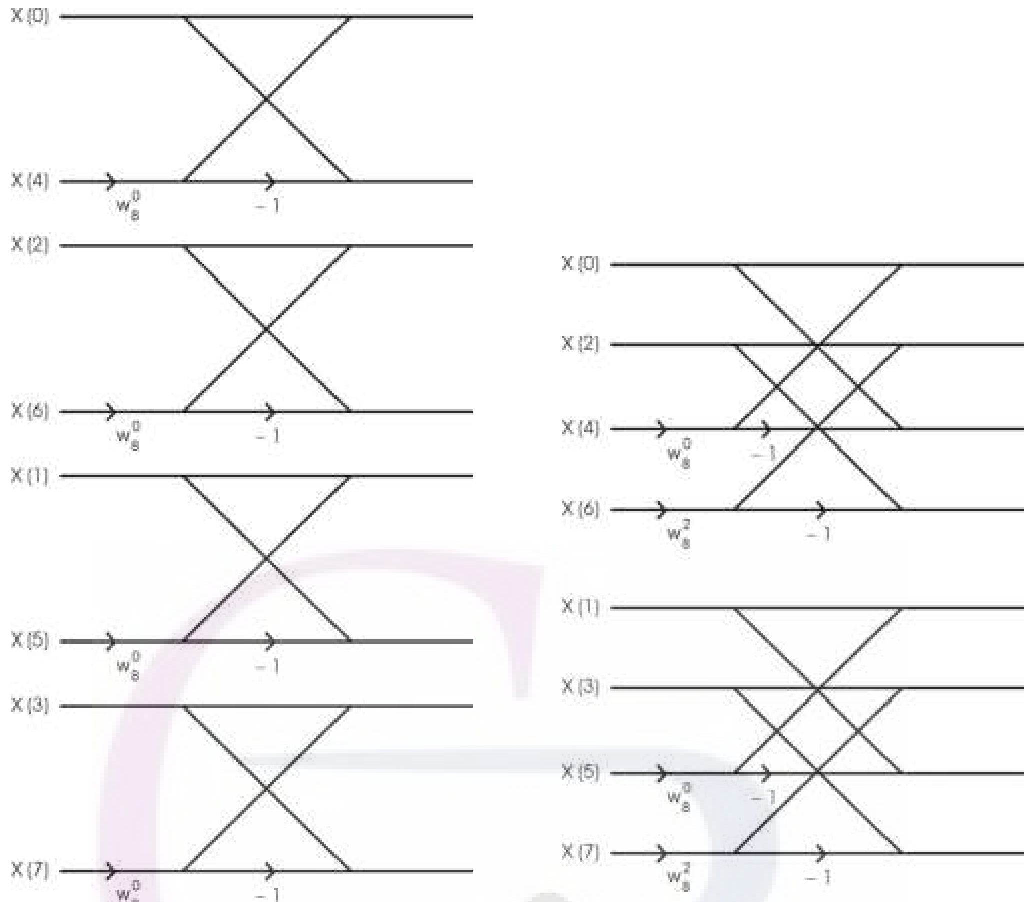


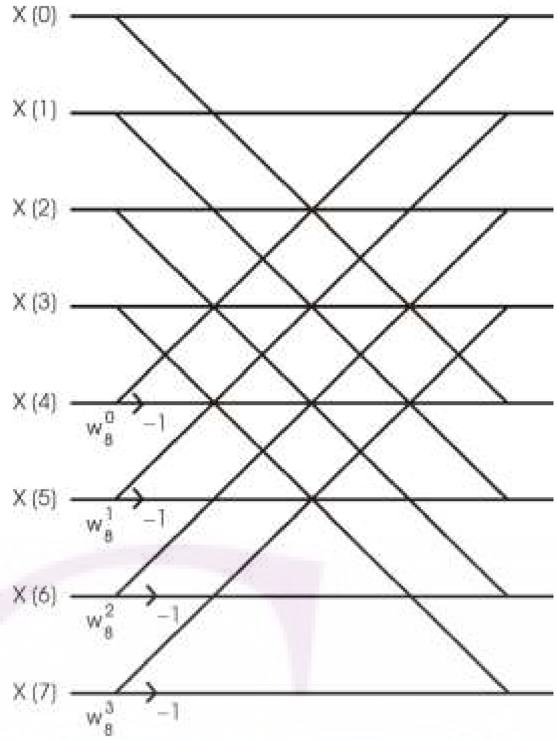
FIG. 3A.1

The above shown mathematical representation forms the basis of N point FFT and is called the **Butterfly Structure**.



STAGE – I

STAGE - II



STAGE – III

FIG. 3A.2 – 8 POINT DIT

ALGORITHM

Step 1 sample the input (N) of any desired frequency. Convert it to fixed-point format and scale the input to avoid overflow during manipulation.

Step 2 Declare four buffers namely real input, real exponent, imaginary exponent and imaginary input.

Step 3 Declare three counters for stage, group and butterfly.

Step 4 Implement the Fast Fourier Transform for the input signal.

Step 5 Store the output (Real and Imaginary) in the output buffer.

Step 6 Decrement the counter of butterfly. Repeat from the Step 4 until the counter reaches zero.

Step 7 If the butterfly counter is zero, modify the exponent value.

Step 8 Repeat from the Step 4 until the group counter reaches zero.

Step 9 If the group counter is zero, multiply the butterfly value by two and divide the group value by two.

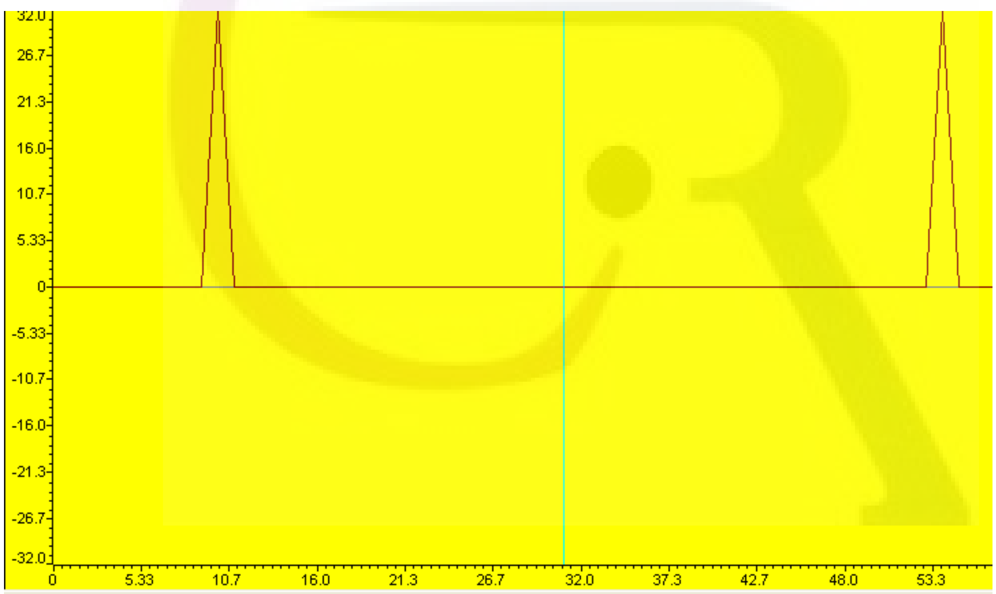
Step 10 Repeat from the Step 4 until the stage counter reaches zero.

Step 11 Transmit the FFT output through **line out** port.

PROCEDURE:

1. *Open Code Composer Studio and make sure DSP kit is Switched ON*
2. *Start a new Project using 'new project' → from project menu icon and save it.*
3. *Write a program for generating sinewave in 'c' language and save it with '.c' extension.*
4. *Add runtime support library file 'rts6700.lib' to the "library" icon of the project*
5. *Add linear command file 'Hello.cmd' to the source icon of the project.*
6. *Compile the program using icon in task bar and make corrections if any errors occur.*
7. *Build in and Build on the project; using icons in the task bar.*
8. *Go to 'debug' and double click on 'connect' option.*
9. *Go to the file and click on load program and then open the file we find extension '.out' project file.*
10. *Go to debug and run the program.*
11. *To view graphically, select view → graph → time/frequency.*

OUTPUT:



DFT or FFT spectrum of sinusoidal signal f= 10 Hz

Result:

The FFT of the one-dimensional signal is determined by writing 'c' program and using DSP Kit.

EXP :05(A)	FREQUENCY RESPONSE OF ANALOG LPF (USING ELLIP TYPE FILTER)	

AIM: To write a matlab program for finding the frequency response of analog LPF using ellip type.

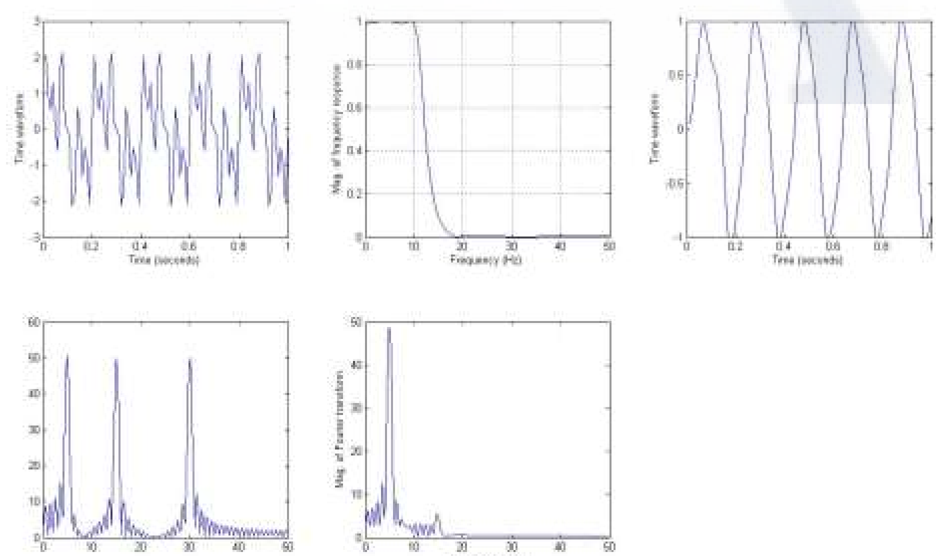
EQUIPMENTS:

PC with Matlab Software

RESULT:

Matlab program for finding the frequency response of analog LPF using ellip type is written.

Waveforms:



EXP :05(A)	FREQUENCY RESPONSE OF ANALOG LPF (USING ELLIP TYPE FILTER)	

AIM: To write a matlab program for finding the frequency response of analog LPF using ellip type.

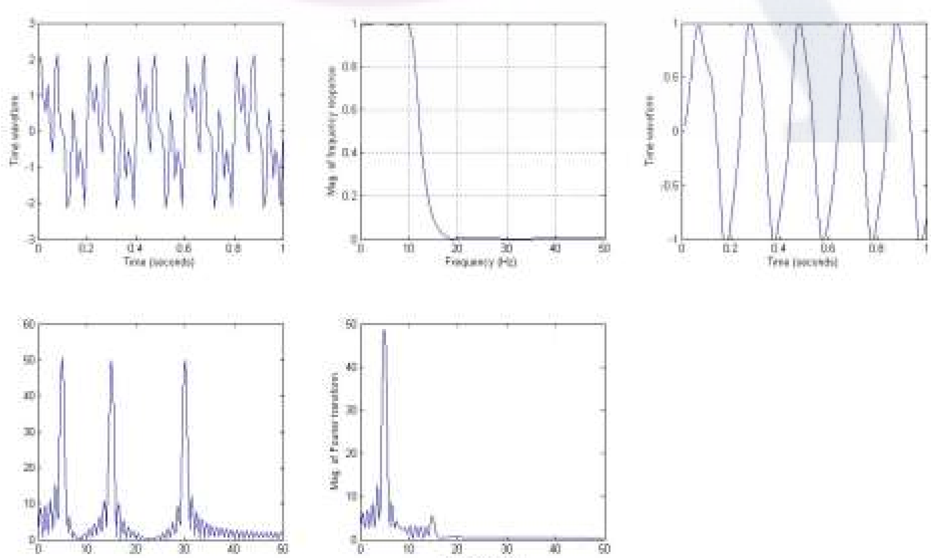
EQUIPMENTS:

PC with Matlab Software

RESULT:

Matlab program for finding the frequency response of analog LPF using ellip type is written.

Waveforms:



EXP :05(B)	FREQUENCY RESPONSE OF ANALOG HPF(USING BUTTERWORTH FILTER)	

AIM: To write a matlab program for finding the frequency response of analog HPF using Butterworth type.

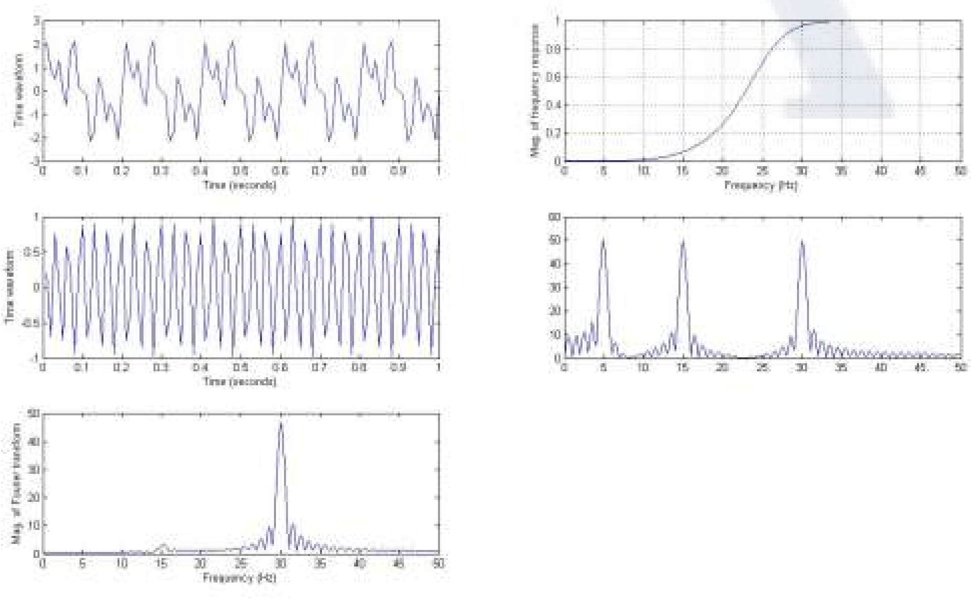
EQUIPMENTS:

PC with Matlab Software

RESULT:

Matlab program for finding the frequency response of analog HPF using Butterworth type is written.

Waveforms:



EXP :5(c)	FREQUENCY RESPONSE OF ANALOG BPF(USING CHEBY1 FILTER)	

AIM: To write Matlab program for finding the frequency response of analog BPF using cheby-1 type.

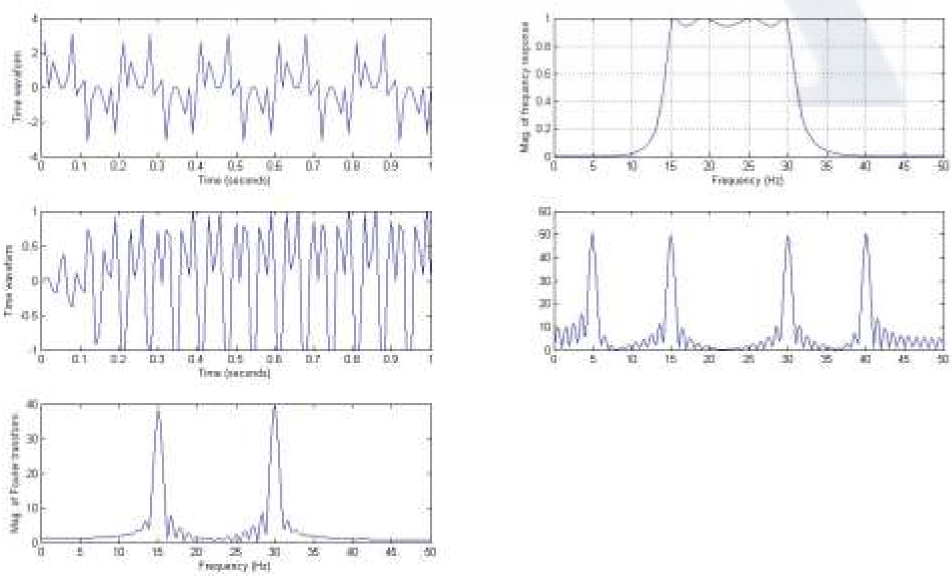
EQUIPMENTS:

PC with Matlab Software

RESULT:

Matlab program for finding the frequency response of analog BPF using cheby-1 type is written.

Waveforms:



EXP :05(D)	FREQUENCY RESPONSE OF ANALOG BSF(USING CHEBY2 FILTER)	

AIM: To write Matlab program for finding the frequency response of analog BSF using cheby-2 type.

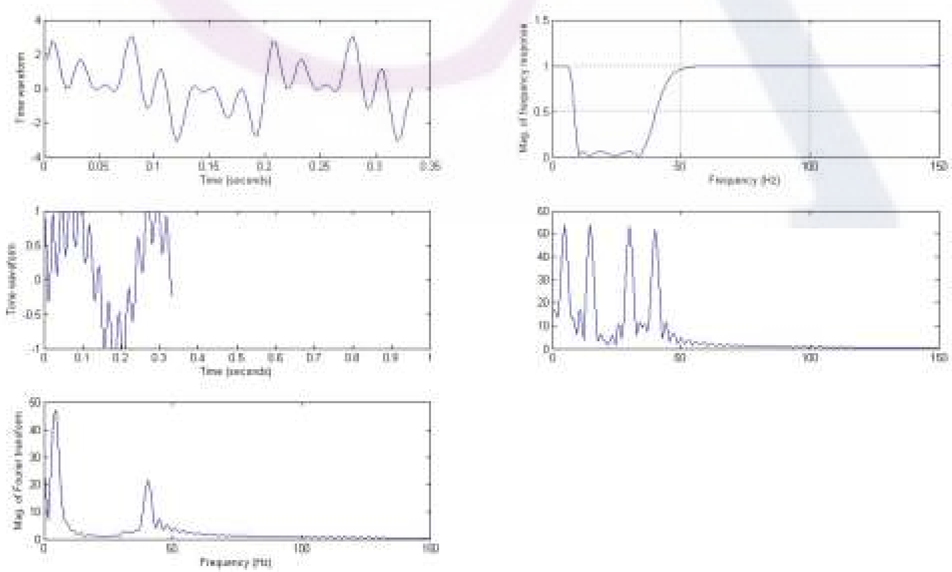
EQUIPMENTS:

PC with Matlab Software

RESULT:

Matlab program for finding the frequency response of analog BSF using cheby-2 type is written.

Waveforms:



EXP :6	POWER DENSITY SPECTRUM OF AM WAVE	

AIM: To write a matlab program power density spectrum of the sequence in matlab.

EQUIPMENTS:

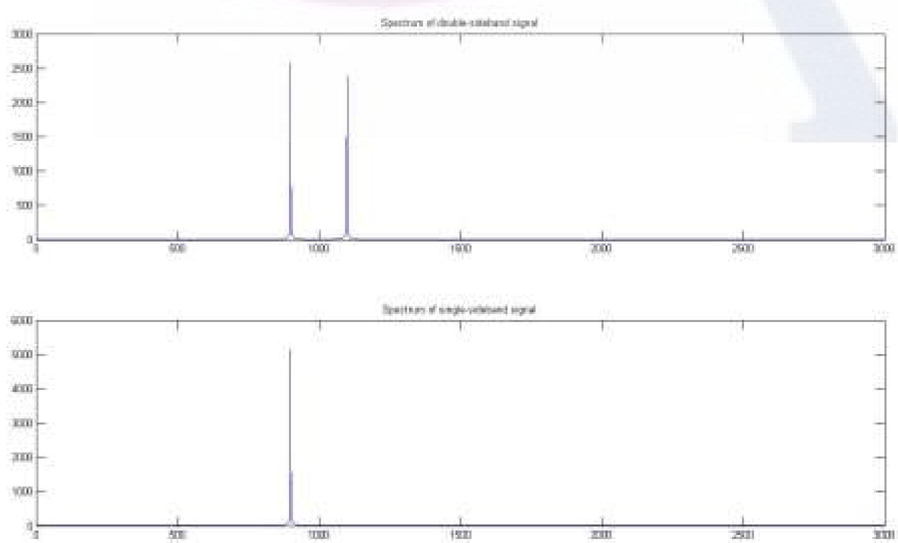
PC with Matlab Software

RESULT:

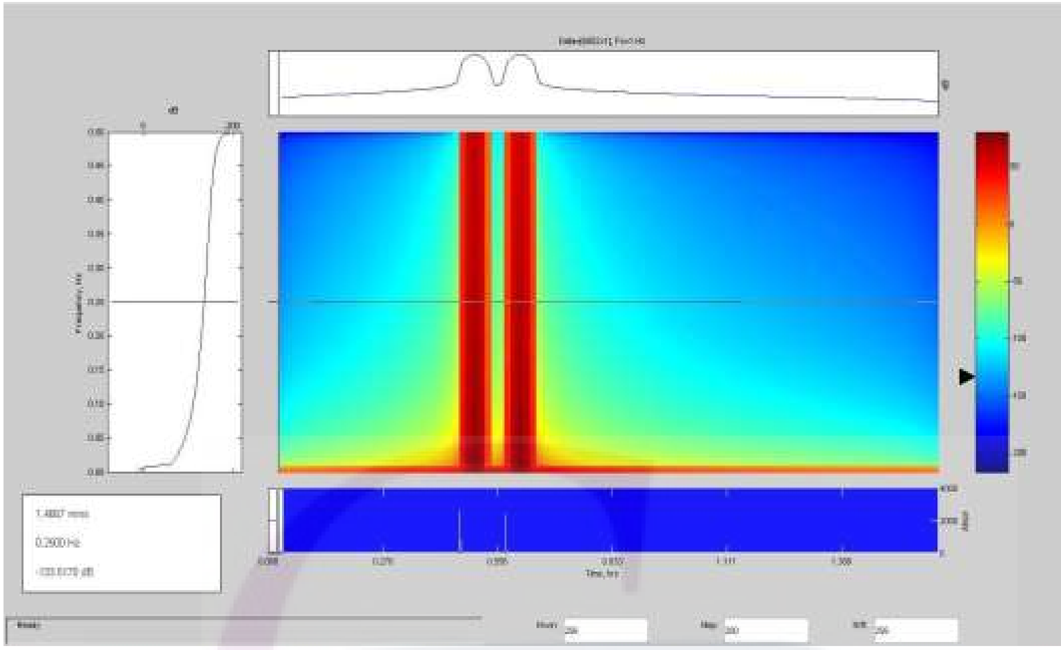
By using matlab programme, power density spectrum is found.

Waveforms:

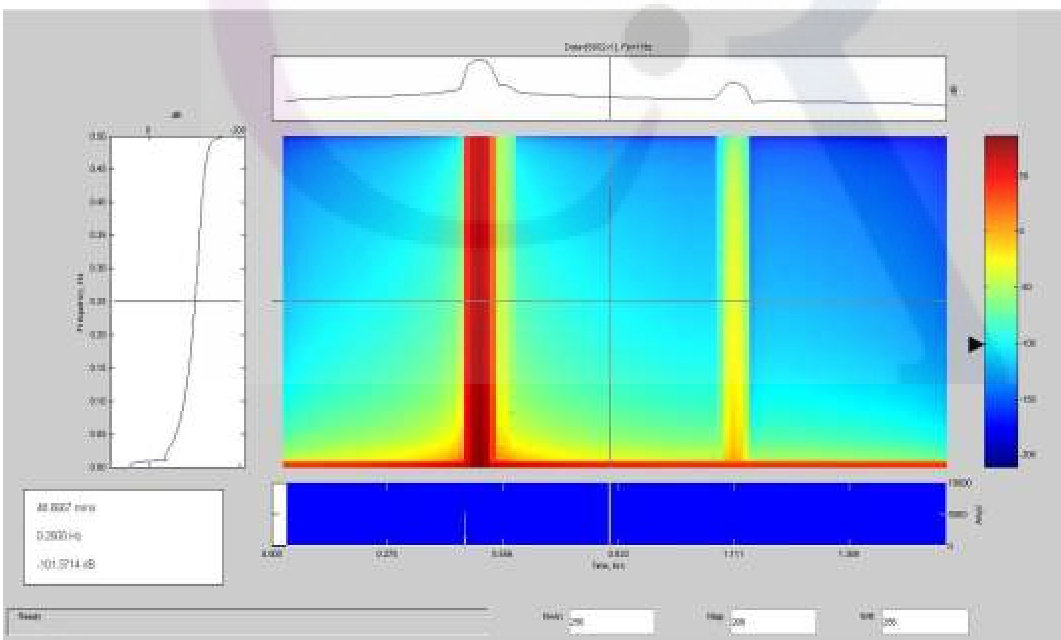
Absolute value of FFT of Zdouble & Zsingle Vs Frequency:



Specgramdemo output for zdouble:



Specgramdemo output for zsingle:



FIR FILTERS (Design Experiments)

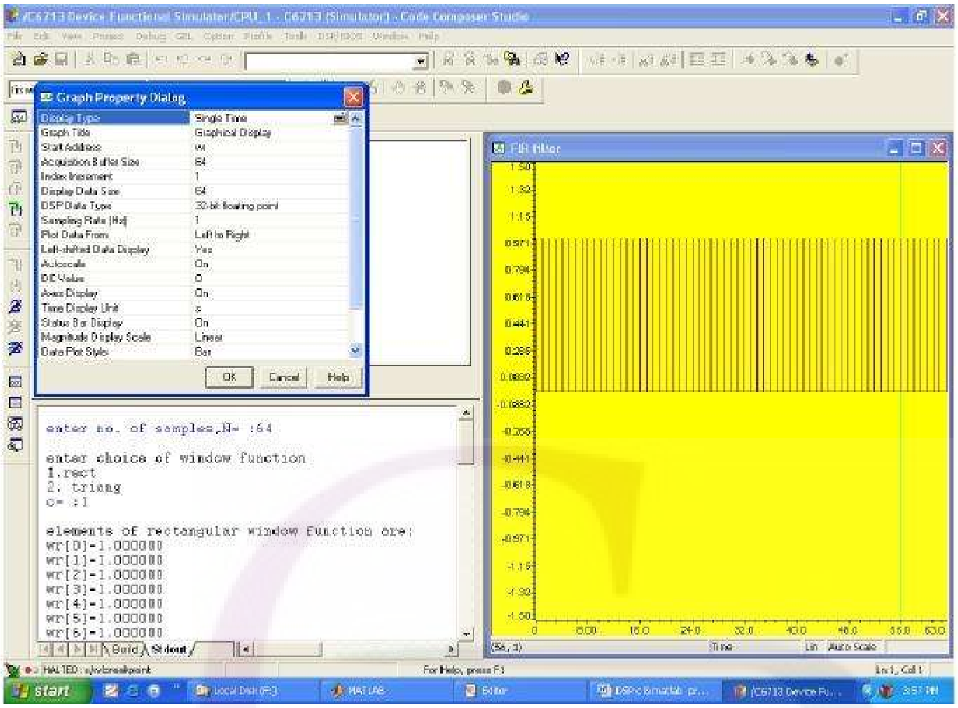
AIM
To verify FIR filters.

EQUIPMENTS:

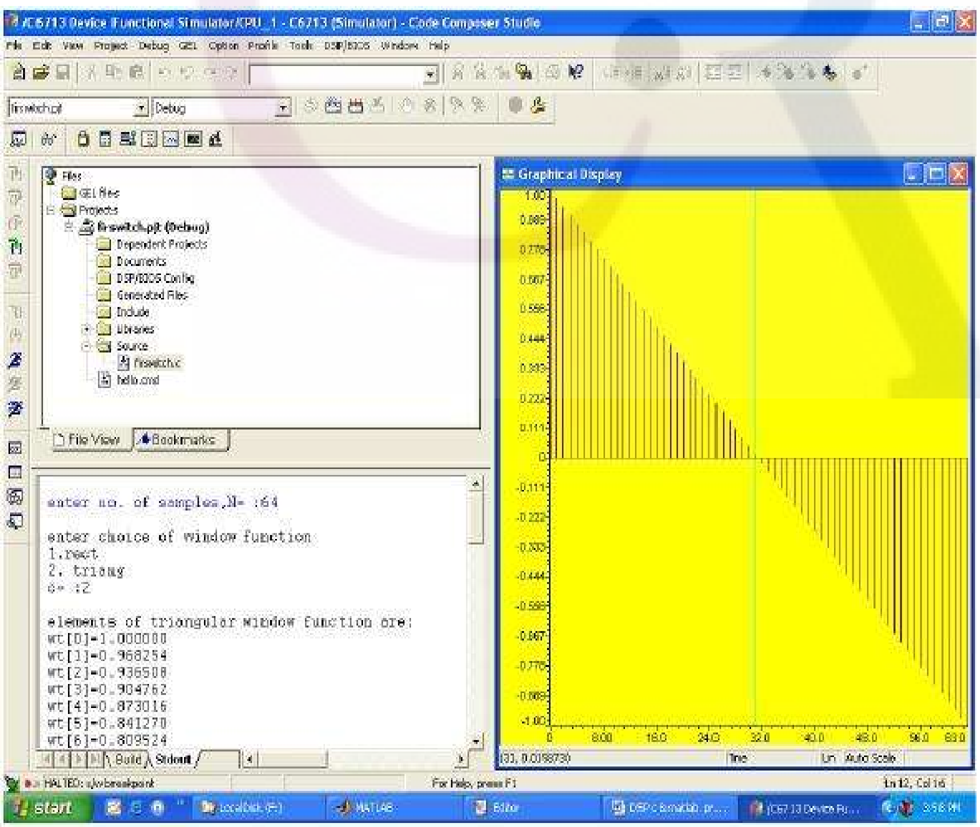
Operating System	– Windows XP
Constructor	**-** Simulator
Software	- CCStudio 3 & MATLAB 7.5

THEORY:
A Finite Impulse Response (FIR) filter is a discrete linear time-invariant system whose output is based on the weighted summation of a finite number of past inputs. The coefficients are generated by using FDS (Filter Design Software or Digital filter design package). FIR – filter is a finite impulse response filter. Order of the filter should be specified. Infinite response is truncated to get finite impulse response. placing a window of finite length does this. Types of windows available are Rectangular, Barlett, Hamming, Hanning, Blackmann window etc. This FIR filter is an all zero filter.

RESULTS:

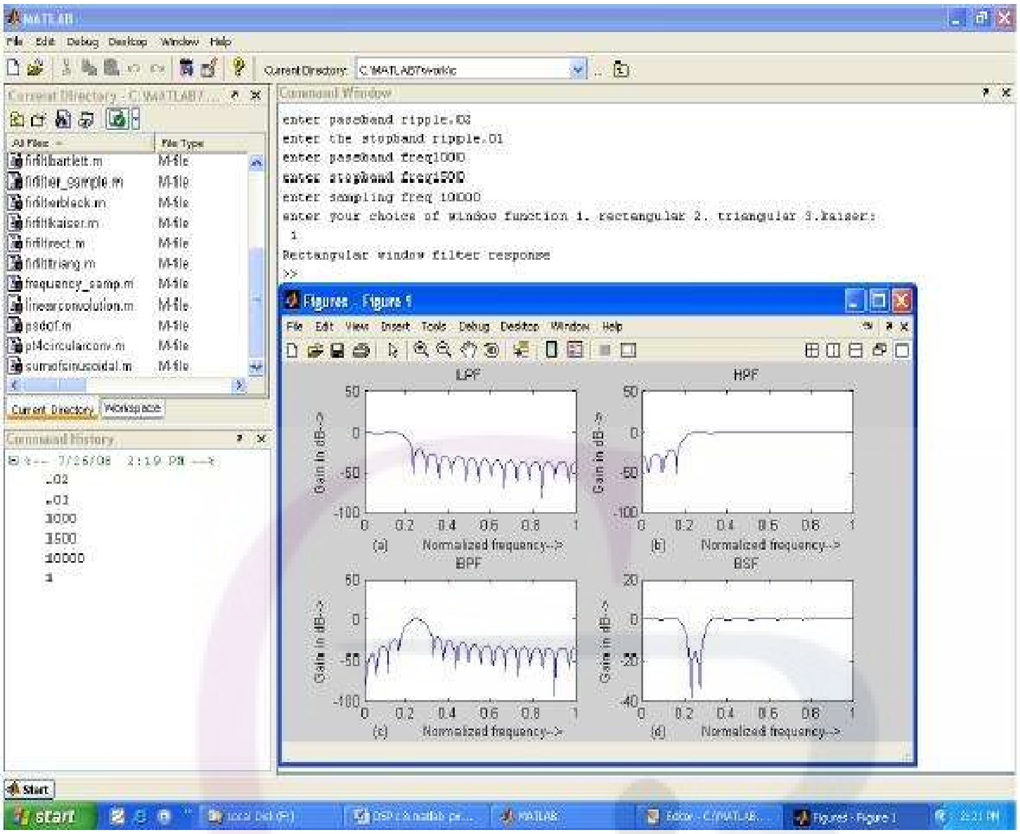


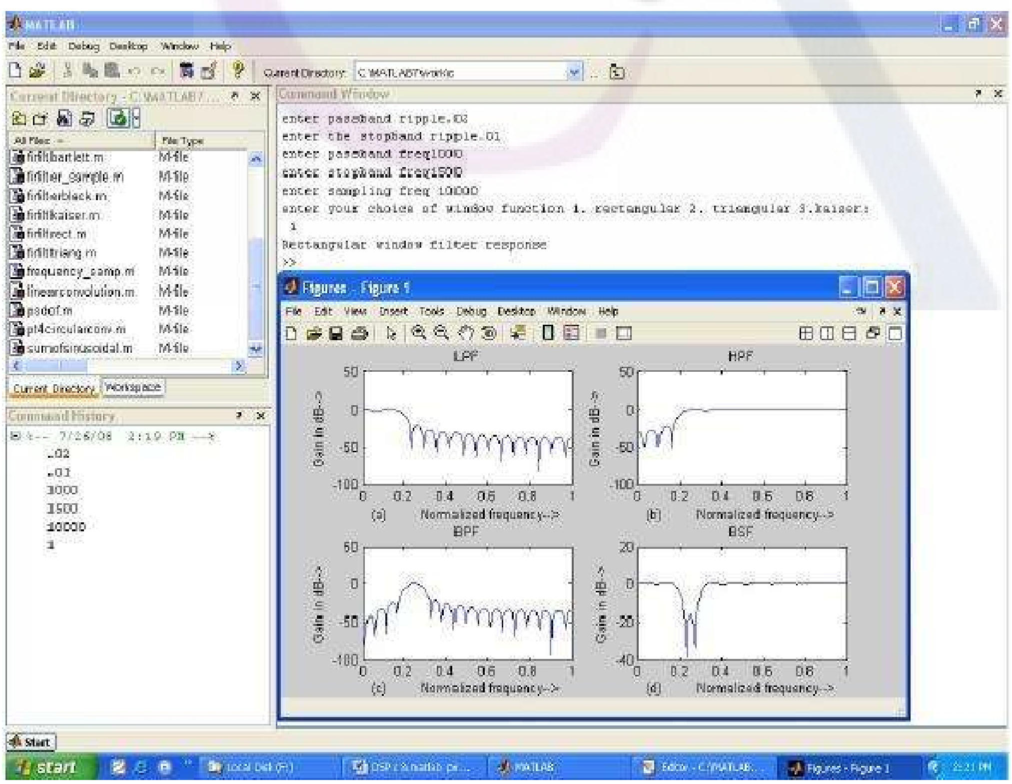
/C6713 Device Functional Simulator/CPU_1 - C6713 (Simulator) - Code Composer Studio
Graph Property Dialog
FIR filter
enter no. of samples,N= :64
enter choice of window function
1.rect
2. triang
c= :1
elements of rectangular window function are:
wr[0]=1.000000
wr[1]=1.000000
wr[2]=1.000000
wr[3]=1.000000
wr[4]=1.000000
wr[5]=1.000000
wr[6]=1.000000



/C6713 Device Functional Simulator/CPU_1 - C6713 (Simulator) - Code Composer Studio
Graphical Display
enter no. of samples,N= :64
enter choice of window function
1.rect
2. triang
c= :2
elements of triangular window function are:
wt[0]=1.000000
wt[1]=0.968254
wt[2]=0.936508
wt[3]=0.904762
wt[4]=0.873016
wt[5]=0.841270
wt[6]=0.809524

RESULTS:


enter passband ripple.02
enter the stopband ripple.01
enter passband freq1000
enter stopband freq1500
enter sampling freq 10000
enter your choice of window function 1. rectangular 2. triangular 3.kaiser:
1
Rectangular window filter response
LPF
HPF
BPF
BSF
Normalized frequency-->
Gain in dB-->


enter passband ripple.02
enter the stopband ripple.01
enter passband freq1000
enter stopband freq1500
enter sampling freq 10000
enter your choice of window function 1. rectangular 2. triangular 3.kaiser:
1
Rectangular window filter response
LPF
HPF
BPF
BSF
Normalized frequency-->
Gain in dB-->

IIR filters(Additional Experiments)

AIM
To design and implement IIR (LPF/HPF)filters.

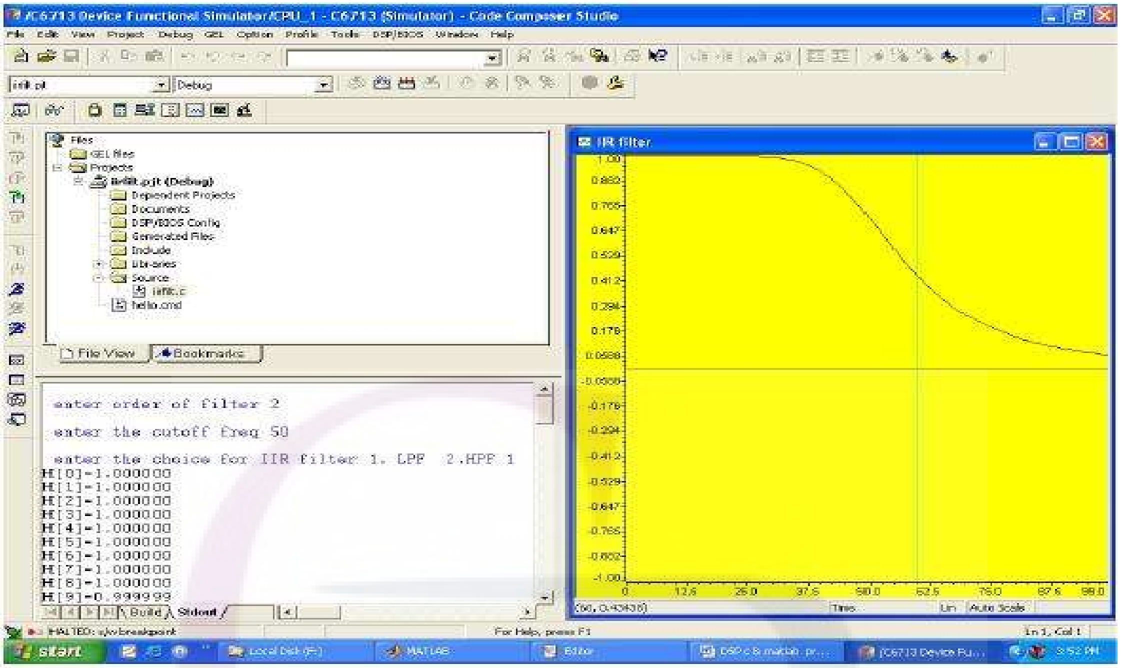
EQUIPMENTS:

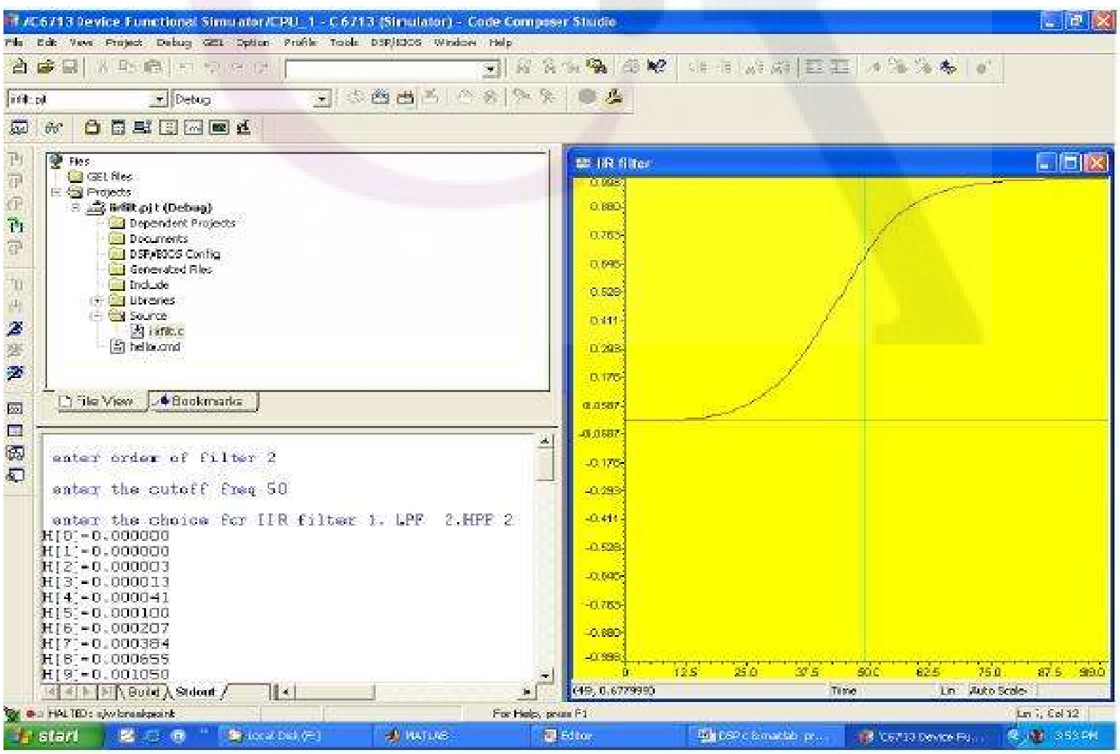
Operating System	– Windows XP
Constructor	- Simulator
Software	- CCStudio 3 & MATLAB 7.5

THEORY:

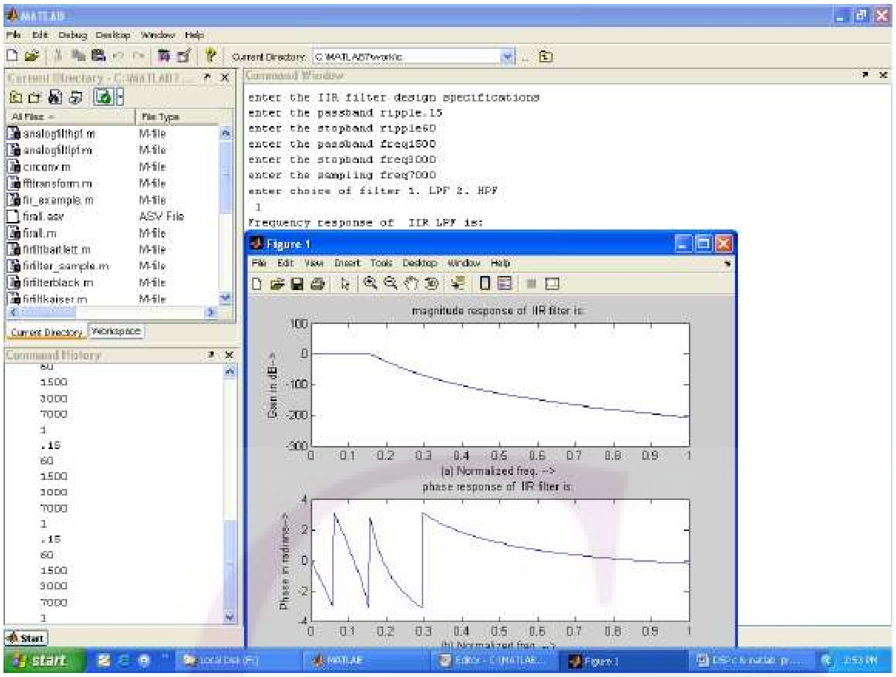
The IIR filter can realize both the poles and zeroes of a system because it has a rational transfer function, described by polynomials in z in both the numerator and the denominator
These filter coefficients are generated using FDS
(Filter Design software or Digital Filter design package).
IIR filters can be expanded as infinite impulse response filters. In designing IIR filters, cutoff frequencies of the filters should be mentioned. The order of the filter can be estimated using butter worth polynomial. That's why the filters are named as butter worth filters. Filter coefficients can be found and the response can be plotted.

RESULTS:

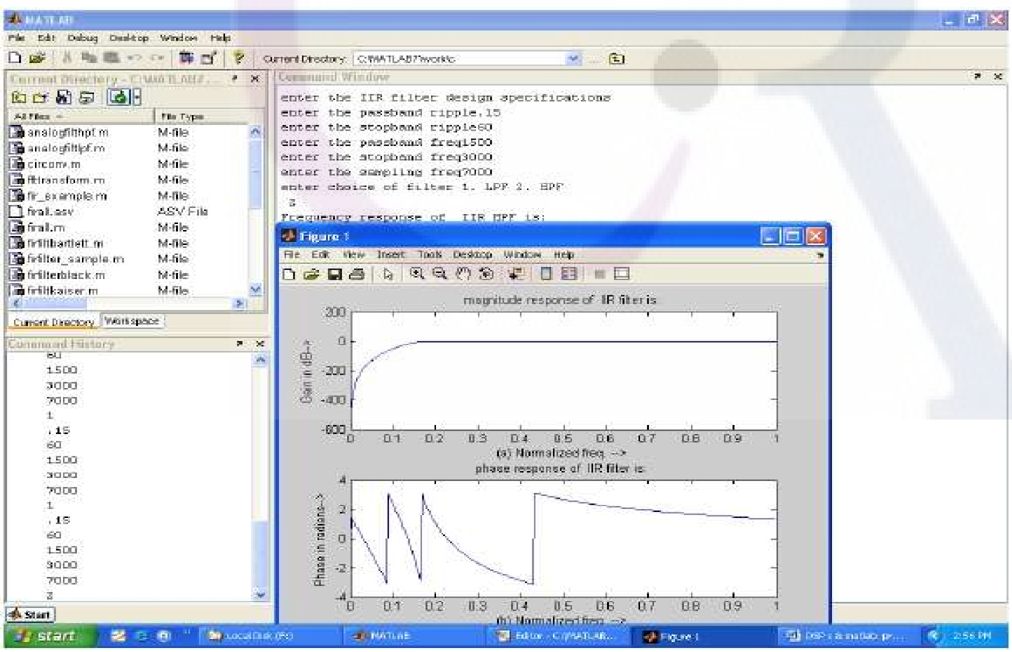

/C6713 Device Functional Simulator/CPU_1 - C6713 (Simulator) - Code Composer Studio
IIR filter
enter order of filter 2
enter the cutoff freq 50
enter the choice for IIR filter 1. LPF 2.HPF 1
H[0]=1.000000
H[1]=1.000000
H[2]=1.000000
H[3]=1.000000
H[4]=1.000000
H[5]=1.000000
H[6]=1.000000
H[7]=1.000000
H[8]=1.000000
H[9]=0.999999


/C6713 Device Functional Simulator/CPU_1 - C6713 (Simulator) - Code Composer Studio
IIR filter
enter order of filter 2
enter the cutoff freq 50
enter the choice for IIR filter 1. LPF 2.HPF 2
H[0]=0.000000
H[1]=0.000000
H[2]=0.000003
H[3]=0.000013
H[4]=0.000041
H[5]=0.000100
H[6]=0.000207
H[7]=0.000384
H[8]=0.000655
H[9]=0.001050

RESULTS:



enter the IIR filter design specifications
enter the passband ripple.15
enter the stopband ripple60
enter the passband freq1500
enter the stopband freq3000
enter the sampling freq7000
enter choice of filter 1. LPF 2. HPF
1
Frequency response of IIR LPF is:
magnitude response of IIR filter is:
Gain in dB-->
(a) Normalized freq. -->
phase response of IIR filter is:
Phase in radians-->



enter the IIR filter design specifications
enter the passband ripple.15
enter the stopband ripple60
enter the passband freq1500
enter the stopband freq3000
enter the sampling freq7000
enter choice of filter 1. LPF 2. HPF
2
Frequency response of IIR HPF is:
magnitude response of IIR filter is:
Gain in dB-->
(a) Normalized freq. -->
phase response of IIR filter is:
Phase in radians-->