DEPARTMENT OF INFORMATION TECHNOLOGY
ANNA UNIVERSITY, MIT CAMPUS

IT7512 – EMBEDDED SYSTEMS LABORATORY (V SEMESTER - 2020 - 2021)

OBJECTIVES:

1. To learn tools relevant to Embedded Systems
2. To explore Embedded C Programs for different embedded processor
3. To write and interpret simple assembly programs that use various features of the processor.

LIST OF EXERCISES:

1. 8051 Assembly Language Experiments (Kit and Simulator) based on:
    a. Data transfer programs
    b. Arithmetic and logical programs
    c. Conversions and sorting
    d. Timers and Interrupts
    e. Serial Communication
    f. I/O interfacing: Traffic Generator, DAC, ADC, Stepper Motor
2. Basic and Interfacing Programs Using Embedded C
3. Real time system programs (Embedded C)
4. KEIL software example programs
5. ARM/Atom based Application Development:
    a. Programs to practice data processing instructions.
    b. Interfacing programs
    c. Program that uses combination of C and ARM/Atom assembly code.
6. Embedded Application Development on Platforms like Bluemix

POINTS TO BE NOTED:

m-modify at certain location.

u-view the content at certain location.

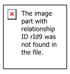
a-make changes at particular location.

g-to execute a program.

ra – used to view contents of registers.

1. STUDY OF 8051 MICRO CONTROLLER

I) THE ARCHITECTURE OF 8051 MICRO CONTROLLER:



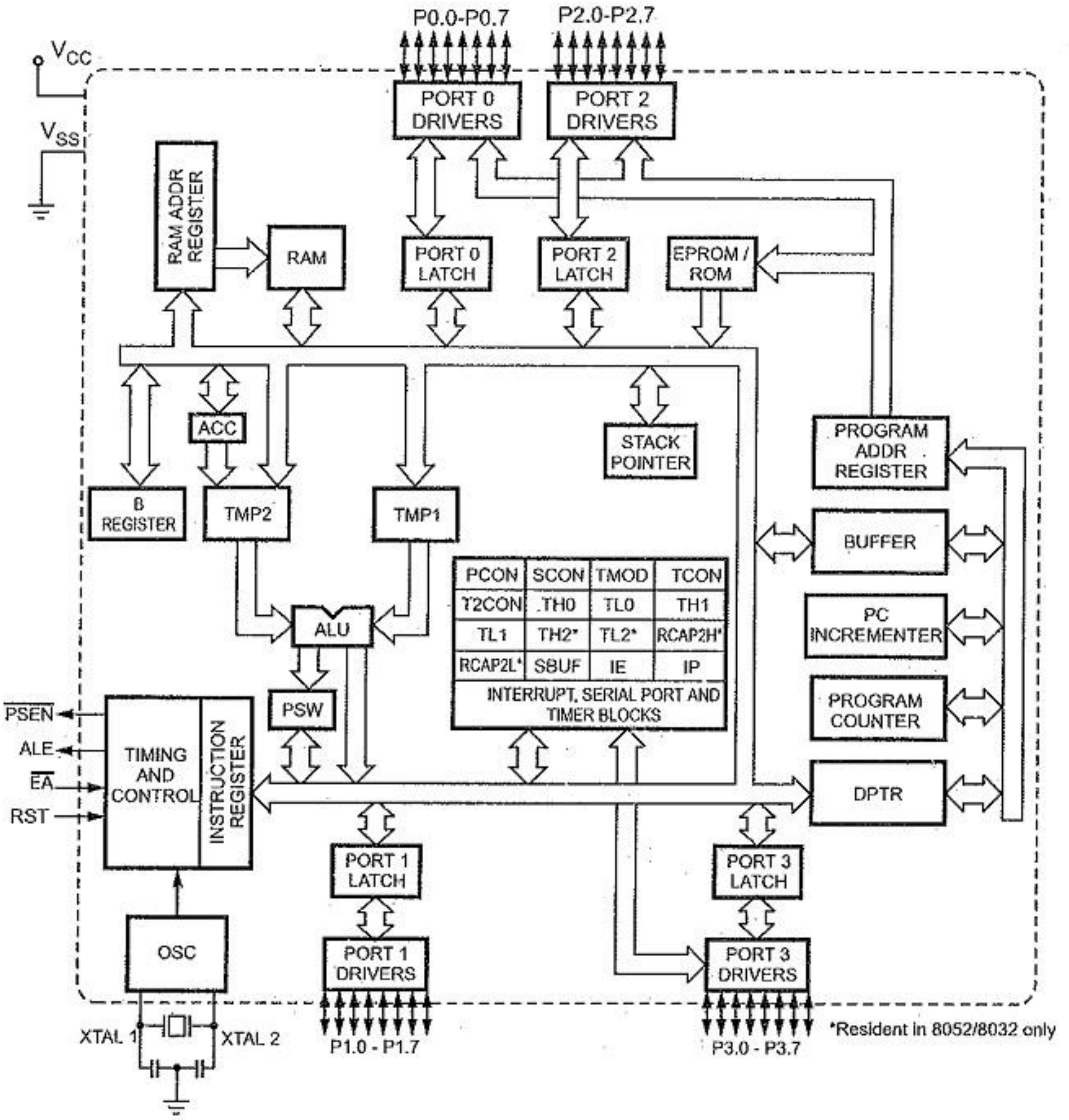
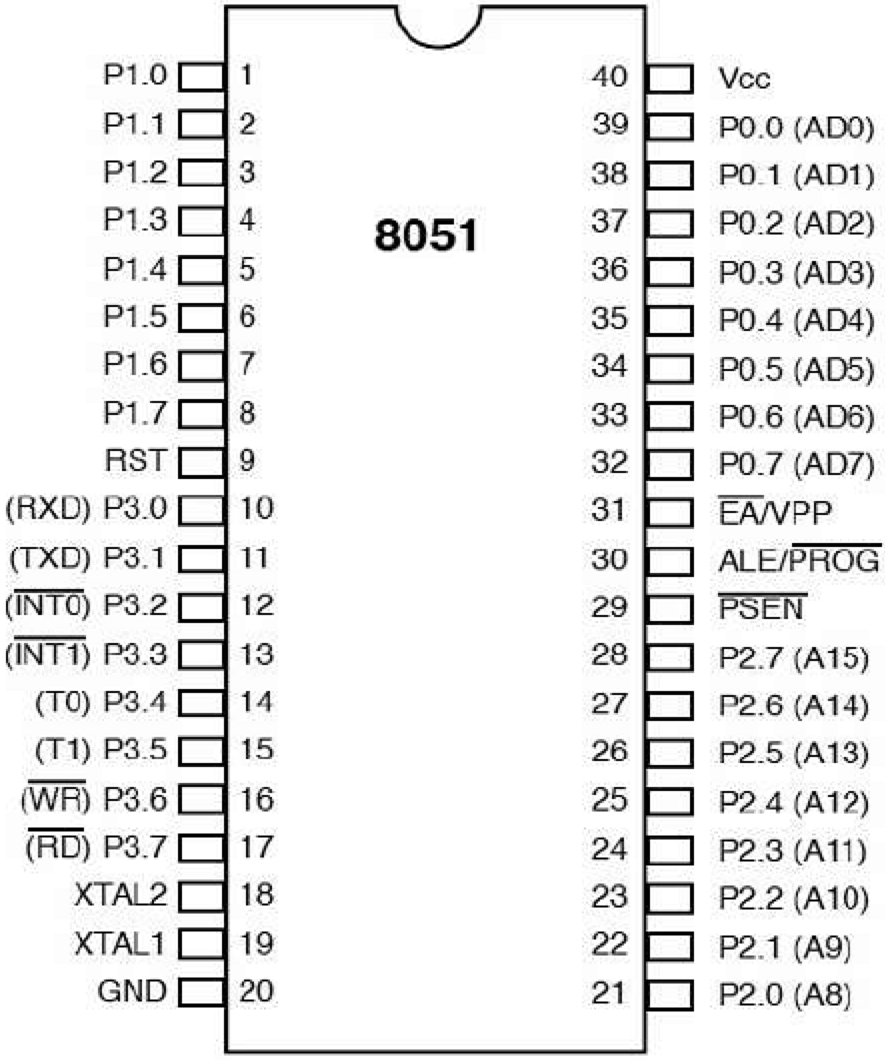


Fig. 12.2 Intel 8051/8031 architecture

Pin diagram of 8051 microcontroller



II) LIST OF COMPONENTS IN THE ARCHITECTURE OF 8051:

An 8051 microcontroller has the following 12 major components:

1. ALU (Arithmetic and Logic Unit)
2. PC (Program Counter)
3. Registers
4. Timers and counters
5. Internal RAM and ROM
6. Four general purpose parallel input/output ports
7. Interrupt control logic with five sources of interrupt
8. Serial data communication
9. PSW (Program Status Word)
10. Data Pointer (DPTR)
11. Stack Pointer (SP)
12. Data and Address bus.

1. ALU

- All arithmetic and logical functions are carried out by the ALU.
- Addition, subtraction with carry, and multiplication come under arithmetic operations. Logical AND, OR and exclusive OR (XOR) come under logical operations.

2. Program Counter (PC)

- A program counter is a 16-bit register and it has no internal address.
- The basic function of program counter is to fetch from memory the address of the next instruction to be executed.
- The PC holds the address of the next instruction residing in memory and when a command is encountered, it produces that instruction.
- This way the PC increments automatically, holding the address of the next instruction.

3. Registers

- Registers are usually known as data storage devices. 8051 microcontroller has 2 registers, namely **Register A** and **Register B**.
- Register A serves as an accumulator while Register B functions as a general purpose register. These registers are used to store the output of mathematical and logical instructions.
- The operations of addition, subtraction, multiplication and division are carried out by Register A.
- Register B is usually unused and comes into picture only when multiplication and division functions are carried out

- Register A also involved in data transfers between the microcontroller and external memory.

4. Internal RAM and ROM

ROM

A code of 4K memory is incorporated as on-chip ROM in 8051. The 8051 ROM is a non-volatile memory meaning that its contents cannot be altered and hence has a similar range of data and program memory, i.e, they can address program memory as well as a 64K separate block of data memory.

RAM

The 8051 microcontroller is composed of 128 bytes of internal RAM.

This is a volatile memory since its contents will be lost if power is switched off.

These 128 bytes of internal RAM are divided into 32 working registers which in turn constitute 4 register banks (Bank 0-Bank 3) with each bank consisting of 8 registers (R0 - R7).

5.PSW (Program Status Word)

Program Status Word or PSW is a hardware register which is a memory location which holds a program's information and also monitors the status of the program this is currently being executed. PSW also has a pointer which points towards the address of the next instruction to be executed. PSW register has 3 fields namely are instruction address field, condition code field and error status field. We can say that PSW is an internal register that keeps track of the computer at every instant.

6.Stack Pointer (SP)

The stack pointer (SP) in 8051 is an 8-bit register. The main purpose of SP is to access the stack. As it has 8-bits it can take values in the range 00 H to FF H. Stack is a special area of data in memory. The SP acts as a pointer for an address that points to the top of the stack.

III) 8051 INSTRUCTION SET:

Arithmatic Operations:

OPCODE	OPERAND	DESCRIPTION	NO. OF BYTES
ADD	A,Rn	Add register to Accumulator	1
ADD	A,direct	Add direct byte to Accumulator	2
ADD	A,@Ri	Add indirect RAM to Accumulator	1
ADD	A,#data	Add immediate data to Accumulator	2
ADDC	A,Rn	Add register to Accumulator with Carry	1
ADDC	A,direct	Add direct byte to Accumulator with Carry	2
ADDC	A,@Ri	Add indirect RAM to Accumulator with Carry	1
ADDC	A,#data	Add immediate data to Acc with Carry	2
SUBB	A,Rn	Subtract Register from Acc with borrow	1
SUBB	A,direct	Subtract direct byte from Acc with borrow	2
SUBB	A,@Ri	Subtract indirect RAM from ACC with borrow	1
SUBB	A,#data	Subtract immediate data from Acc with borrow	2
INC	A	Increment Accumulator	1
INC	Rn	Increment register	1
INC	direct	Increment direct byte	2
INC	@Ri	Increment direct RAM	1
DEC	A	Decrement Accumulator	1
DEC	Rn	Decrement Register	1
DEC	direct	Decrement direct byte	2
DEC	@Ri	Decrement indirect RAM	1
INC	DPTR	Increment Data Pointer	1
MUL	AB	Multiply A & B	1
DIV	AB	Divide A by B	1
DA	A	Decimal Adjust Accumulator	1

LOGICAL OPERATIONS:

ANL	A,Rn	AND Register to Accumulator	1
ANL	A,direct	AND direct byte to Accumulator	2
ANL	A,@Ri	AND indirect RAM to Accumulator	1
ANL	A,#data	AND immediate data to Accumulator	2
ANL	direct,A	AND Accumulator to direct byte	2
ANL	direct,#data	AND immediate data to direct byte	3
ORL	A,Rn	OR register to Accumulator 1	1
ORL	A,	direct OR direct byte to Accumulator	2
ORL	A,@Ri	OR indirect RAM to Accumulator	1
ORL	A,#	data OR immediate data to Accumulator	2
ORL	direct,A	OR Accumulator to direct byte 2	2
ORL	direct,#data	OR immediate data to direct byte	3
XRL	A,Rn	Exclusive-OR register to Accumulator	1
XRL	A,direct	Exclusive-OR direct byte to Accumulator	2
XRL	A,@Ri	Exclusive-OR indirect RAM to Accumulator	1
XRL	A,#data	Exclusive-OR immediate data to Accumulator	2
XRL	direct,A	Exclusive-OR Accumulator to direct byte	2
XRL	direct,#data	Exclusive-OR immediate data to direct byte	3
CLR	A	Clear Accumulator	1
CPL	A	Complement Accumulator	1
RL	A	Rotate Accumulator Left	1
RLC	A	Rotate Accumulator Left through the Carry	1
RR	A	Rotate Accumulator Right	1
RRC	A	Rotate Accumulator Right through the Carry	1
SWAP	A	Swap nibbles within the Accumulator	1

DATA TRANSFER OPERATIONS:

MOV	A,Rn	Move register to Accumulator	1
MOV	A,direct	Move direct byte to Accumulator	2
MOV	A,@Ri	Move indirect RAM to Accumulator	1
MOV	A,#data	Move immediate data to Accumulator	2
MOV	Rn,A	Move Accumulator to register	1
MOV	Rn,direct	Move direct byte to register	2
MOV	Rn,#data	Move immediate data to register	2
MOV	direct,A	Move Accumulator to direct byte	2
MOV	direct,Rn	Move register to direct byte	2

MOV	direct,direct	Move direct byte to direct	3
MOV	direct,@Ri	Move indirect RAM to direct byte	2
MOV	direct,#data	Move immediate data to direct byte	3
MOV	@Ri,A Move	Accumulator to indirect RAM	1
MOV	@Ri,direct	Move direct byte to indirect RAM	2
MOV	@Ri,#data	Move immediate data to indirect RAM	2
MOV	DPTR,#data16	Load Data Pointer with a 16-bit Constant	3
MOVC	A,@A+DPTR	Move Code byte relative to DPTR to Acc	1
MOVC	A,@A+PC	Move Code byte relative to PC to Acc	1
MOVX	A,@Ri	Move External RAM (8-bit address) to Acc	1
MOVX	A,@DPTR	Move External RAM (16-bit address) to Acc	1
MOVX	@Ri,A	Move Acc to External RAM (8-bitaddr)	1
MOVX	@DPTR,A	Move Acc to External RAM (16-bitaddr)	1
PUSH	direct	Push(Write) direct byte onto stack	2
POP	direct	Pop(Read) direct byte from stack	2
XCH	A,Rn	Exchange register with Accumulator	1
XCH	A,direct	Exchange direct byte with Accumulator	2
XCH	A,@Ri	Exchange indirect RAM with Accumulator	1
XCHD	A,@Ri	Exchange low-order Digit indirect RAM with Acc	1

BOOLEAN VARIABLE MANIPULATION:

CLR	C	Clear Carry	1
CLR	bit	Clear direct bit	2
SETB	C	Set Carry	1
SETB	bit	Set direct bit	2
CPL	C	Complement Carry	1
CPL	bit	Complement direct bit	2
ANL	C,bit	AND direct bit to CARRY	2
ANL	C,/bit	AND complement of direct bit to Carry	2
ORL	C,bit	OR direct bit to Carry	2
ORL	C,/bit OR	complement of direct bit to Carry	2
MOV	C,bit	Move direct bit to Carry	2
MOV	bit,C	Move Carry to direct bit	2
JC	rel	Jump if Carry is set	2
JNC	rel	Jump if Carry not set	2
JB	bit,rel	Jump if direct Bit is set	3
JNB	bit,rel	Jump if direct Bit is Not set	3
JBC	bit,rel	Jump if direct Bit is set & clear bit	3

PROGRAM BRANCHING:

ACALL	addr11	Absolute Subroutine Call	2
LCALL	addr16	Long Subroutine Call	3
RET		Return from Subroutine	1
RETI		Return from interrupt	1
AJMP	addr11	Absolute Jump	2
LJMP	addr16	Long Jump	3
SJMP	rel	Short Jump (relative address)	2
JMP	@A+DPTR	Jump indirect relative to the DPTR	1
JZ	rel	Jump if Accumulator is Zero	2
JNZ	rel	Jump if Accumulator is Not Zero	2
CJNE	A,direct,rel	Compare direct byte to Acc and Jump if Not Equal	3
CJNE	A,#data,rel	Compare immediate to Acc and Jump if Not Equal	3
CJNE	Rn,#data,rel	Compare immediate to register and Jump if Not Equal	3
CJNE	@Ri,#data,rel	Compare immediate to indirect and Jump if Not Equal	3
DJNZ	Rn,rel	Decrement register and Jump if Not Zero	2
DJNZ	direct,rel	Decrement direct byte and Jump if Not Zero	3
NOP		No Operation	1

2 8-BIT ARITHMETIC OPERATIONS

AIM:

To perform 8-bit arithmetic addition, subtraction, multiplication and division using 8051 microcontroller.

ALGORITHM:

ADDITION:

- Take 2, 8 bit numbers.
- Add two 8 bit numbers.
- Get the result by executing the commands in the accumulator.

ADDITION:

ADDRESS	OPCODE	MNEUMONICS	DESCRIPTION
8510	79 15	MOV R1,#15	Move the immediate value 15 to R1 register
8512	7A 25	MOV R2,#25	Move the immediate value 25 to R2 register
8514	E9	MOV A,R1	Move the value in register R1 to accumulator
8515	2A	ADD A,R2	Add the value in accumulator and in register r2 and store in accumulator
8516	12 00BB	LCALL 00BB	Exit from program

OUTPUT:

A=3A

SUBTRACTION:

- Take two 8 bit numbers
- Subtract that two 8 bit numbers using subb.
- Get the result by executing the commands in the accumulator

ADDRESS	OPCODE	MNEUMONICS	DESCRIPTION
8530	79 10	MOV R1,#10	Move the immediate value 10 to R1 register
8532	7A 05	MOV R2,#5	Move the immediate value 5 to R2 register
8534	E9	MOV A,R1	Move the value in register R1 to accumulator
8535	9A	SUBB A,R2	Subtract the value in accumulator and in register r2 and store in accumulator
8536	12 00BB	LCALL 00BB	Exit from program

OUTPUT:

0B

MULTIPLICATION:

- Take two 8 bit registers
- Multiply the 2 numbers without using register,fetching values into registers and using datapointer
- Get the result using the three methods .

I) WITHOUT REGISTERS:

ADDRESS	OPCODE	MNEUMONICS	DESCRIPTION
8515	74 11	MOV R1,#11	Move the immediate value to R1 register
8517	75 F0 12	MOV F0,#12	Move the immediate value to B register
851A	A4	MUL AB	Multiply the value in accumulator and b register
851B	12 00 BB	LCALL 00BB	Exit the program

OUTPUT: A=32 B=01

II) WITH REGISTERS:

ADDRESS	OPCODE	MNEUMONICS	DESCRIPTION
8500	79 02	MOV R1,#02	Move the immediate value to R1 register
8502	7A 03	MOV R2,#03	Move the immediate value to R2 register
8504	E9	MOV A,R1	Move the value to accumulator
8505	8A F0	MOV F0,R2	Move the value to b register
8507	A4	MUL AB	Multiply accumulator and b register
8508	FB	MOV R3,A	Move the value from accumulator to R3
8509	12 00	LCALL 00BB	Exit the program

OUTPUT: A=06

III) USING DATAPOINTER:

ADDRESS	OPCODE	MNEUMONICS	DESCRIPTION
8600	74 03	MOV A,#03	Move the immediate value to accumulator
8602	75 F0 02	MOV F0,#02	Move the immediate value to B register
8605	A4	MUL AB	Multiply accumulator and B register
8606	90 85 00	MOV DPTR,#8500	Allocate address 8500 to DPTR
8609	F0	MOVX @DPTR,A	Store the value of accumulator to the DPTR
860A	A3	INC DPTR	Increment the DPTR
860B	E5 F0	MOV A,F0	Move the value of B register to accumulator
860D	F0	MOVX @DPTR,A	Store the value of accumulator to the DPTR
860E	12 00 BB	LCALL 00BB	Exit the program

OUPUT:

8500=06 8501=00

DIVISION:

- Take two 8 bit registers
- Divide the 2 numbers without using register,fetching values into registers and using Datapointer.
- Get the result using the three methods .

IV) WITHOUT REGISTERS:

ADDRESS	OPCODE	MNEUMONICS	DESCRIPTION
8580	74 2A	MOV A,#2A	Move the immediate value to R1 register
8582	75 F0 0A	MOV F0,#0A	Move the immediate value to B register
8585	84	DIV AB	Divide the value in accumulator and b register
8586	12 00 BB	LCALL 00BB	Exit the program

OUTPUT: A=04 B=02

V) WITH REGISTERS:

ADDRESS	OPCODE	MNEUMONICS	DESCRIPTION
8570	79 2A	MOV R1,#2A	Move the immediate value to R1 register
8572	7A 0A	MOV R2,#0A	Move the immediate value to R2 register
8574	E9	MOV A,R1	Move the value to accumulator
8575	8A F0	MOV F0,R2	Move the value to b register
85077	84	DIV AB	Divide accumulator and b register
8578	FB	MOV R3,A	Move the value from accumulator to R3
8579	12 00 BB	LCALL 00BB	Exit the program

OUTPUT: A=04 B=02

VI) USING DATAPOINTER:

ADDRESS	OPCODE	MNEUMONICS	DESCRIPTION
8610	74 2A	MOV A,#2A	Move the immediate value to accumulator
8612	75 F0 0A	MOV F0,#0A	Move the immediate value to B register
8615	84	DIV AB	Divide accumulator and B register
8616	90 85 10	MOV DPTR,#8510	Allocate address 8500 to DPTR
8619	F0	MOVX @DPTR,A	Store the value of accumulator to the DPTR
861A	A3	INC DPTR	Increment the DPTR
861B	E5 F0	MOV A,F0	Move the value of B register to accumulator
861D	F0	MOVX @DPTR,A	Store the value of accumulator to the DPTR
861E	12 00 BB	LCALL 00BB	Exit the program

OUTPUT:

8510=04 8511=02

3. 16 BIT ARITHMETIC OPERATIONS

AIM:

To perform 16 bit arithmetic operations using 8051 micro controller.

ADDITION ALGORITHM:

- Store the hexadecimal numbers in the internal RAM as LSB and MSB.
- Store the operands in internal RAM
- Add the numbers using registers.
- Store the result of LSB and MSB with carry in data pointer.

USING DPTR AND MEMORY LOCATIONS:

ADDRESS	OPCODE	MNEUMONICS	DESCRIPTION
8540	74 40	MOV R1,#40	Move the immediate value to register
8542	E7	MOV A,@R1	Move the value to accumulator
8543	01	INC R1	Increment R1
8544	27	ADD A,@R1	Add the value at R1 with accumulator
8545	90 86 30	MOV DPTR,#8630	Allocate address 8500 to DPTR
8548	F0	MOVX @DPTR,A	Store the value of accumulator to the DPTR
8549	A3	INC DPTR	Increment DPTR
854A	09	INC R1	Increment R1 register
854B	E7	MOV A,@R1	Store the value at R1 to accumulator

854C	09	INC R1	Increment R1
854D	37	ADDC A,@R1	Add with carry the value in R1 and accumulator
854E	F0	MOVX @DPTR,A	Store the value of accumulator to the DPTR
854F	12 00 BB	LCALL 00BB	Exit the program

OUTPUT:

8630=40 8631=60

SUBTRACTION ALGORITHM:

- Store the hexadecimal number into 2 registers as LSB and MSB.
- Subtract the lower older bytes and store it in a register.
- Subtract the higher order bytes and store it in another register.

USING REGISTERS:

ADDRESS	OPCODE	MNEUMONICS	DESCRIPTION
8680	74 20	MOV R1,#20	Move the immediate value to register
8682	7A 10	MOV R2,#10	Move the immediate value to accumulator
8684	E9	MOV A,R1	Move the value in R1 to accumulator
8685	9A	SUBB A.R2	Subtract value in A with value in R2
8686	FD	MOV R5,A	Move the value in accumulator to R5
8687	7B 20	MOV R3,#20	Move the value to R3 register
8689	7C 10	MOV R4,#10	Move the value to R4 register
868B	EB	MOV A,R3	Move the value in R1 to accumulator

868C	9C	SUBB A,R4	Subtract value in A with R4
868D	FE	MOV R6,A	Move the value in A to R6
868E	12 00 BB	LCALL 00BB	Exit the program

OUTPUT:

R5=10 R6=10

MULTIPLICATION ALGORITHM:

- Take two 16 bit hexadecimal numbers and place it into four registers.
- Multiply the two lower bite registers and store it in internal memory addresses.
- Now multiply 2 higher bit registers and store it in internal addresses along with adding the carry from the lower bits.
- Add the result of the multiplication and again store it in internal memory addresses.
- Now the result will be stored in four consecutive internal memory addresses.

ADDRESS	OPCODE	MNEUMONICS	DESCRIPTION
8500	79 3F	MOV R1,#3F	Move the immediate value to register R1
8502	7A 23	MOV R2,#23	Move the immediate value to register R2
8504	7B 11	MOV R3,#11	Move the immediate value to register R3
8506	7C F2	MOV R4,#F2	Move the immediate value to register R4
8508	EB	MOV A,R3	Move the value in register to accumulator
8509	8C F0	MOV F0,R4	Move the value of register to B
850B	A4	MUL AB	Multiply values in A and B register
850C	78 40	MOV R0,#40	Move the memory address to R0

850E	F6	MOV @R0,A	Move the value of A to the memory address in R0
850F	08	INC R0	Increment R1
8510	ES F0	MOV A,F0	Move the value in B to A
8512	F6	MOV @R0,A	Move the value in A to memory address at R0
8513	EC	MOV A,R4	Move the value in R4 to A
8514	89 F0	MOV F0,R1	Move the value in R1 to B register
8516	A4	MUL AB	Multiply the value in A and B register
8517	26	ADD A,@R0	Add the values in A and memory address at R0
8518	F6	MOV @R0,A	Move the value in A to address in R0
8519	08	INC R0	Increment R0
851A	E5 F0	MOV A,F0	Move the value in B reg to A
851C	F6	MOV @R0,A	Move the value in A to address in R0
851D	EA	MOV A,R2	Move the value in R2 to A
851E	8B F0	MOV F0,R3	Move the value in R3 to B reg
8520	A4	MUL AB	Multiply the values in A and B reg
8521	18	DEC R0	Decrement address in R0
8522	26	ADD A,@R0	Add the values in address R0 and A
8523	F6	MOV @R0,A	Move the value in A to address at R0
8524	08	INC R0	Increment address at R0
8525	E5 F0	MOV A,F0	Move the value in B to A
8527	26	ADD A,@R0	Add the value at address in R0 with A
8528	F6	MOV @R0,A	Move the value in A to the address at R0
8529	EA	MOV A,R2	Move the value in R2 to A
852A	89 F0	MOV F0,R1	Move the value in R1to B reg
852C	A4	MUL AB	Multiply the values in A and B reg
852D	26	ADD A,@R0	Add the values in address at R0
852E	F6	MOV @R0,A	Move the value in A to address atR0

852F	08	INC R0	Increment the address at R0
8530	ES F0	MOV A,F0	Add the values in B and A
8532	34 00	ADDC A,#00	Add A with immediate value 0
8534	F6	MOV @R0,A	Move the value in A to address at R0
8535	12 00 BB	LCALL 00BB	Exit the program

OUTPUT:

0040:12

0041:F1

0042:DA

0043:08

RESULT:

Thus all the 16 - bit operation are compiled and executed successfully.

4.MULTI - BYTE ARITHMETIC OPERATIONS

AIM :

To perform multi - byte arithmetic operations using 8051 micro - controller.

ADDTION ALGORITHM:

- For adding four byte number. We are counter, initialized as 04.
- Store the first operands values in internal address 40 and the other operands in the internal address as 50.
- Add the two values and store them back in internal adder 40.
- Decrement the counter and perform the operations again until the counter becomes zero.

ADDRESS	OPCODE	MNEUMONICS	DESCRIPTION
8600	78 40	MOV R0,#40	Move memory address 40 to R0
8602	7A 50	MOV R1,#50	Move the memory address 50 to R1
8606	7B 04	MOV R3,#04	Move the counter values to register R3
8608	E6	MOV A,@R0	Move the values at address R0 and move to A
8609	37	ADDC A,@R1	Add the values at address R1 and A
860A	F6	MOV @R0,A	Move the value at A to address R0
860B	08	INC R0	Increment address R0
860C	09	INC R1	Increment address R1
860D	DB F9	DJNZ R3,8608	If the counter is not zero loop back to address 8608
860F	12 00 BB	LCALL 00BB	Exit the program

INPUT:

R0->0040 0041 0042 0043

4 2 1 3

R1->0050 0051 0052 0053

3 3 5 4

OUTPUT:

R0->0040 0041 0042 0043

7 5 6 7

SUBTRACTION ALGORITHM:

- Inilialze the counter as 4 for subtracting 4 - byte numbers.
- Store the first operand values in internal memory starting from address 40.
- Store the second operand values in internal memory starting from address 50.
- Subtract the two values and store it in internal memory address 40.
- Decrement the counter and perform the operation again until,counter becomes zero.

ADDRESS	OPCODE	MNEUMONICS	DESCRIPTION
8600	78 40	MOV R0,#40	Move the memory address 40 to R0
8602	7A 50	MOV R1,#50	Move the memory address 50 to R1
8606	7B 04	MOV R3,#04	Move the counter to R3
8608	E6	MOV A,@R0	Move the value at R0 to A
8609	97	SUBB A,@R1	Subtract the value in A with the value in address R1
860A	F6	MOV @R0,A	Move the value A to address at R0
860B	08	INC R0	Increment R0

860C	09	INC R1	Increment R1
860D	DB F9	DJZN R3,8608	Decrement R3,if not zero the loop back to 8608
860F	12 00 BB	LCALL 00BB	Exit the program

INPUT:

R0->0040 0041 0042 0043

7 5 6 7

R1->0050 0051 0052 0053

3 3 5 4

OUTPUT:

R0->0040 0041 0042 0043

4 2 1 3

MULTIPLICATION ALGORITHM:

- Take a 16 - bit multiplexer and and store in memory address 40.
- Take a 8 - bit multiplexer and store in memory address 50.
- Multiply 16 - bit and 8 - bit values and store in memory address 60.
- Store the carry in the accumulator.

ADDRESS	OPCODE	MNEUMONICS	DESCRIPTION
8510	7A 02	MOV R2,#02	Move the counter value 2 to R2

8512	78 40	MOV R0,#40	Move the memory address 40 to R0
8514	AD 50	MOV R5,50	Move the memory address 50 to R5
8516	79 60	MOV R1,#60	Move the memory address 60 to R1 for storing result
8518	C3	CLR C	Clear carry
8519	7C 00	MOV R4,#00	Clear the register R4
851B	E6	MOV A,@R0	Move the value of R0 to A
851C	8D F0	MOV F0,R5	Move the value R5 to B register
851E	A4	MUL AB	Multiply value in A and B register
851F	3C	ADDC A,R4	Add value in A and R4
8520	F7	MOV @R1,A	Move the A value to address stored in R1
8521	AC F0	MOV R4,F0	Move the value in B register to R4
8523	08	INC R0	Increment R0
8524	09	INC R1	Increment R1
8525	DA F4	DJNZ R2,851B	Decrement R2 and if not zero jump to 851B
8527	E5 F0	MOV A,F0	Move values of B register to A
8529	34 00	ADDC A,#00	Add A with zero and store in A
852B	F7	MOV @R1,A	Move value in A to memory address R1
852C	12 00 BB	LCALL 00BB	Exit the program

INPUT:

40:34

41:12

50:12

OUTPUT:

A:01

60:A8

61:47

62:01

RESULT:

Thus all the multi - byte arithmetic operations are compiled and executed successfully.

5.SORTING : ASCENDING & DESECENDING ORDER

AIM:

To sort the given elements in the ascending and desecending order.

SORTING IN ASCENDING ORDER:

ALGORITHM:

- Array of element are stored in the internal memory.
- Sort the given element in the ascending order and store it in the same memory.
- Display the result.

ADDRESS	OPCODE	MNEUMONICS	DESCRIPTION
8500	7C 04	MOV R4,#04	Move the immediate value to register R4
8502	7B 04	AGAIN:MOV R3,#04	Move the immediate value to register R3
8504	78 40	MOV R0,#40	Move the immediate value to register R0
8506	CB	CLR C	Clear the carry
8507	E6	UP:MOV A,@R0	Move the value in register to accumulator
8508	F9	MOV R1,A	Move the value of register to accumulator
8509	08	INC R0	Increment register R0
850A	E6	MOV A,@R0	Move the register value in the accumulator
850B	99	SUBB A,R1	Subtract R1 from A
850C	50 06	JNC SKIP	Jump to SKIP if no carry
850E	E6	MOV A,@R0	Move the register value to the accumulator

850F	18	DEC @R0	Decrement R0
8510	F6	MOV @R0,A	Move the accumulator to register
8511	E9	MOV A,R1	Move the register to accumulator
8512	08	INC R0	Increment R0
8513	F6	MOV @R0,A	Move the accumulator to the register R0
8514	DB F1	SKIP:DJNZ R3,UP	GO TO UP,if R3>0
8516	DC EB	DJNZ R4,AGAIN	GO TO AGAIN,if R4>0
8518	12 00 BB	LCALL 00BB	Terminate the program

INPUT:

40:6

41:2

42:3

43:8

OUTPUT:

40:2

41:3

42:6

43:8

SORTING IN DESCENDING ORDER:

ALGORITHM:

- Array of element are stored in the internal memory.
- Sort the given element in the descending order and store it in the same memory.
- Display the result.

ADDRESS	OPCODE	MNEUMONICS	DESCRIPTION

8500	7C 04	MOV R4,#04	Move the immediate value to register R4
8502	7B 04	AGAIN:MOV R3,#04	Move the immediate value to register R3
8504	78 40	MOV R0,#40	Move the immediate value to register R0
8506	CB	CLR C	Clear the carry
8507	E6	UP:MOV A,@R0	Move the value in register to accumulator
8508	F9	MOV R1,A	Move the value of register to accumulator
8509	08	INC R0	Increment register R0
850A	E6	MOV A,@R0	Move the register value in the accumulator
850B	99	SUBB A,R1	Subtract R1 from A
850C	50 66	JC SKIP	Jump to SKIP if carry is needed
850E	E6	MOV A,@R0	Move the register value to the accumulator
850F	18	DEC @R0	Decrement R0
8510	F6	MOV @R0,A	Move the accumulator to register
8511	E9	MOV A,R1	Move the register to accumulator
8512	08	INC R0	Increment R0
8513	F6	MOV @R0,A	Move the accumulator to the register R0
8514	DB F1	SKIP:DJNZ R3,UP	GO TO UP,if R3>0
8516	DC EA	DJNZ R4,AGAIN	GO TO AGAIN,if R4>0
8518	12 00 BB	LCALL 00BB	Terminate the program

INPUT:

40:6

41:2

42:3

43:8

OUTPUT:

40:8

41:6

42:3

43:2

RESULT:

Thus the sorting is performed successfully.

6. a.CODE CONVERTER

AIM:

To write the assembly code to convert

- binary to gray code
- Gray to binary code
- Hexadecimal to BCD
- BCD to hexadecimal

BINARY TO GRAY CODE:

ALGORITHM:

- Get the 8-bit binary value.
- Convert binary to gray code.
- Display the result.

ADDRESS	OPCODE	MNEUMONICS	DESCRIPTION
8500	C3	CLR C	Clear the carry
8501	74 A2	MOV A,#A2	Move the immediate value to accumulator
8503	F8	MOV R0,A	Move the accumulator value to register
8504	13	RRC A	Right rotates with carry
8505	68	XRL A,R0	Ex-or accumulator with the register
8506	12 00 BB	LCALL 00BB	Terminate the program

INPUT:

A2

OUTPUT:

R0:F3

GRAY TO BINARY:

ALGORITHM:

- Get the 8-bit binary value.
- Convert gray to binary code
- Display the result.

ADDRESS	OPCODE	MNEUMONICS	DESCRIPTION
8510	74 0B	MOV A,#0B	Move the gray code 0B to accumulator
8512	75 0C 07	MOV 0C,#07	Initialize the counter value as 07
8515	F5 0B	MOV 0B,A	Move the value in A to B
8517	54 B0	ANL A,#80	AND the value of A and 80
8519	1B	RRC A	Rigth rotate the A with carry
851A	54 7F	ANL A,#7F	To extract the MSB bit,add value of A and 7F
851C	65 0B	XRL A,0B	XOR the value A and B
851E	05 0C F8	DJNZ 0C,8519	Decrement the C,if not zero jump to 8519
8521	12 00 BB	LCALL 00BB	Terminate the program

INPUT:

0B

OUTPUT:

A:0D

HEXADECIMAL TO BCD:

ALGORITHM:

- Hexadecimal number is stored in accumulator
- Divide the accumulator value by OA(that is 10).
- Higher bits are stored in R1.
- Lower bits are stored in Re
- View the result.

ADDRESS	OPCODE	MNEUMONICS	DESCRIPTION
8500	7A 2A	MOV A,#2A	Move the 2A value to accumulator A
8502	75 F0 0A	MOV F0,#0A	Move 0A value to register F0
8505	84	DIV AB	Divide A with B
8506	F9	MOV R1,A	Move the accumulator A to R1 register
8507	AA F0	MOV R2,F0	Move F0 register to R2 register
8509	12 00 BB	LCALL 00BB	Terminate the program

INPUT:

2A

OUTPUT:

R2:02

R1:04

BCD TO HEXADECIMAL:

ALGORITHM:

- Get the 8 bit BCD value.
- convert to hexadecimal value
- Display the result.

ADDRE	OPCOD	MNEUMONICS	DESCRIPTION

SS	E		
8600	78 05	MOV R0,#05	Move 5 value to the register R0
8602	79 05	MOV R1,#05	Move 5 value to the register R1
860F	75 F0 0A	MOV F0,#0A	Move the 0A value to the F0 register
8607	E8	MOV A,R0	Move R0 value to accumulator A
8608	A4	MUL AB	Multiply A with B
8609	FB	MOV R3,A	Move accumulator A to r3
860A	29	ADD A,R1	Add A with R1
860B	FB	MOV R3,A	Move accumulator A to R3
860C	12 00 BB	LCALL 00BB	Terminate the program

INPUT:

R0:05 R1:05

OUTPUT:

R3:37

RESULT:

Thus the code converter code are successfully executed.

6.b. HEXADECIMAL TO ASCII

AIM:

To write the assembly code to convert hexadecimal to ASCII character.

HEXADECIMAL TO ASCII:

ALGORITHM:

- Move OE value to accumulator and A to register.
- Check whether the number is in range 0-9 or not.

- When the number is in that range then hexadecimal digit is numeric and simply add 50H with it to get ASCII value.
- When the number is not in the range 0-9 ,the number is in the range A-F,so for that are,convert the number to 37H onward.
- If value is numeric,result will be negative and carry is set,then add 30H to get ASCII value.
- If result is positive or 0,add 37H with the result.

ADDRESS	OPCODE	MNEUMONICS	DESCRIPTION
9000	7C 0E	MOV A,#0E	Move the value 0E to accumulator
9002	FA	MOV R2,A	Move the accumulator value to the register
9003	C3	CLR C	Clear the carry
9004	94 0A	SUBB A,#0A	Subtract accumulator with 10
9006	40 05	JC 900D	Jump to 900D if carry is there
9008	FA	MOV A,R2	Move the R2 to accumulator
9009	24 37	ADD A,#37	Add accumulator value with 37
900B	80 03	SJMP 9010	Jump to 9010
900D	EA	MOV A,R2	Move the register value to accumulator
900E	24 30	ADD A,#30	Add A with 30
9010	F9	MOV R1,A	Move the accumulator to register
9011	12 00 BB	LCALL 00 BB	Terminate the program

INPUT:

0E

OUTPUT:

A:45

RESULT:

Thus the hexadecimal code is converted to ASCII and external successfully.

7.ASSMEBLY CODE IN KEIL SOFTWARE

AIM:

To perform the following operation using assembly language in keil ide.

- Sum of n number
- Addition of two arrays
- Multiplication of 2 2-type number.

SUM OF N NUMBERS:

ALGORITHM:

- Take 5 number & store in internal memory.
- Add it in accumulator.
- Store the result in register.
- View the result.

ASSEMBLY CODE:

```
        ORG 000H
        MOV A,#00H
        MOV R0,#40H
        MOV R1,#05
LABEL:  ADD A,@R0
        INC R0
        DJNZ R1,LABEL
        MOV R2,A
        END
```

INPUT:

I40:01

I41:02

I42:03

I43:04

I44:05

OUTPUT:

R2:0F

ADDITION OF TWO ARRAYS:

ALGORITHM:

- Take two array and store it in internal memory.
- Add the two array and store it in the internal memory.
- View the result.

ASSEMBLY CODE:

```
        ORG 000H
        MOV RO,#50H
        MOV R1,#60H
        MOV R2,#05
LABEL: MOV A,@R0
        ADD A,@R0
        MOV @R0,A
        INC R0
        INC R1
        DJNZ R2,LABEL
        END
```

INPUT:

I50:01 I51:02 I52:03 I53:04 I54:05

I60:02 I61:02 I62:03 I63:04 I64:05

OUTPUT:

I50:02 I51:04 I52:06 I53:08 I54:0

MULTIPLICATION OF 2-16 BIT NUMBER:

ALGORITHM:

- Take two 16 bit number.
- Store the 1st 16 bit number in R1,R3 with MSB and LSB respectively.
- Store the 2nd 16 bit number in R2,R4 with MSB and LSB respectively.
- Multiply these two number.
- Store the result in the internal memory
- View the results.

ASSEMBLY CODE:

```
ORG 000H
MOV R1,#3fH
MOV R2,#23H
MOV R3,#11h
MOV R4,#2fH
MOV A,R3
MOV B,R4
MUL AB
MOV R0,#40H
MOV @R0,A
INC R0
MOV @R0,B
```

```
MOV A,R4
MOV B,R1
MUL AB
ADD A,@R0
MOV @R0,A
INC R0
MOV @R0,B
MOV A,R2
MOV B,R3
MUL AB
DEC R0
ADD A,@R0
MOV @R0,A
INC R0
MOV A,B
ADD ,@R0
MOV @R0,A
MOV A,R2
MOV B,R1
MUL AB
ADD A,@R0
MOV @R0,A
MOV A,B
ADDC A,#00H
INC R0
MOV @R0,A
END
```

OUTPUT:

I40:1F I41:E7 I42:AA I43:08

VALUE:08AAE71F.

RESULT:

Simple operation using assembly language in KEIL IDE is executed successfully.

8.TOGGLING ASSEMBLY/CODE IN KEIL SOFTWARE

AIM:

To perform toggling in both assembly & code in KEIL software by following methods

- Toggling the LED without interrupt in assembly code.
- Toggling the LED with interrupt in assembly code.
- Toggling the LED without interrupt in C code.
- Toggling the LED with interrupt in C code.

A) TOGGLING THE LED WITHOUT INTERRUPT IN ASSEMBLY CODE:

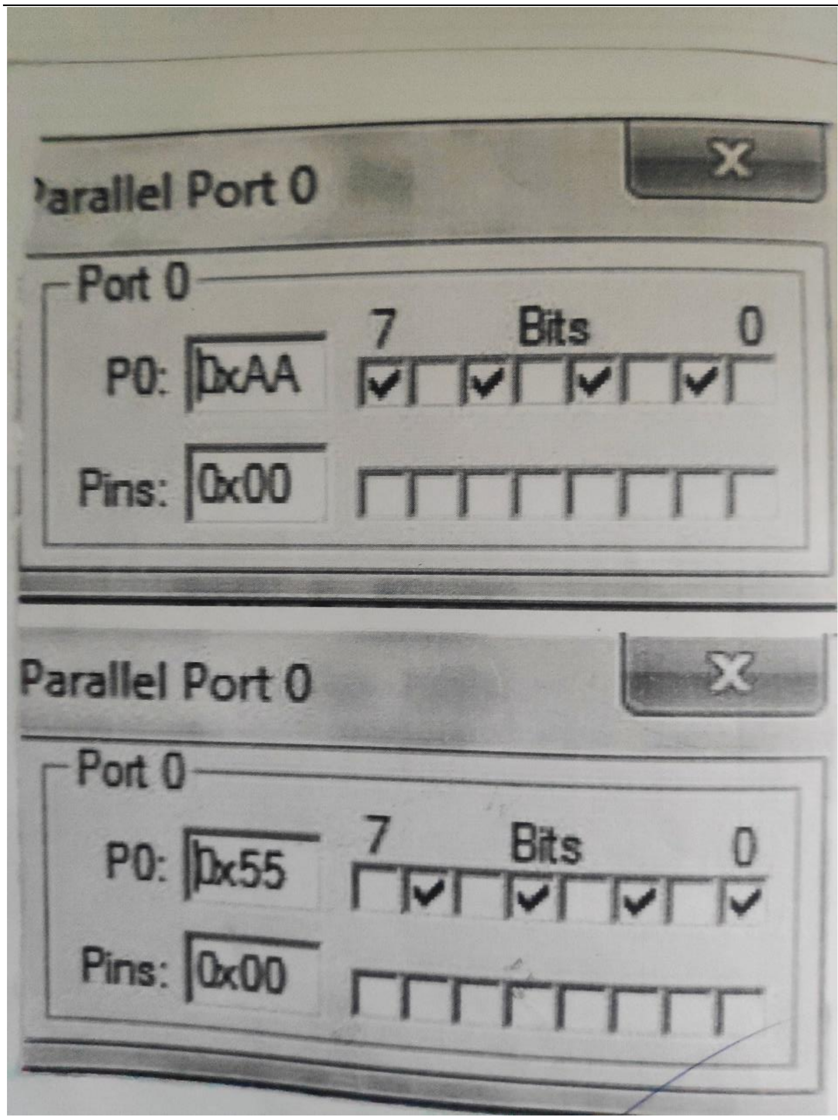
ALGORITHM:

- Assign the port,with some value.
- Assign the accumulator with some other value.
- Move the A value to the port 1.
- Toggle the port bit using ACALL DELAY.

ASSEMBLY CODE:

```
        ORG 000H
          LOOP:MOV P1,#01H
                ACALL DELAY
                MOV A,#55H
                CPL A
                MOV P1,A
                ACALL DELAY
                SJMP LOOP
         DELAY:MOV R0,#20H
                MOV R1,#21H
         ORDER:DJNZ R0,ORDER
          INNER:DJNZ R1,INNER
                 RET
                 END
```

OUTPUT:



B) TOGGLING THE LED WITH INTERRUPT IN ASSEMBLY CODE:

ALGORITHM:

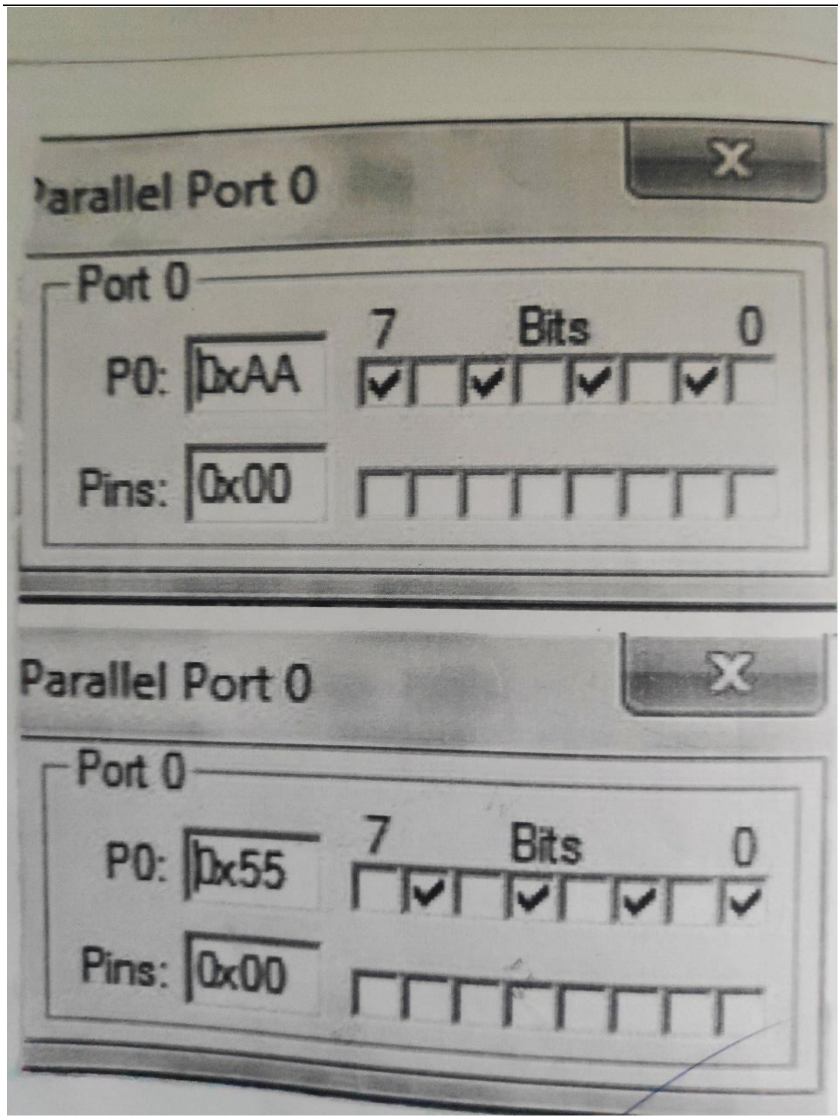
- Assign the port,with some value.
- Assign the accumulator with some other value.
- Complement the accumulator & store it in port 1.
- delay created using times.

ASSEMBLY CODE:

```
ORG 000H
        MOV A,#55H
```

```
AGAIN:CPL A
        MOV P1,A
        ACALL DELAY
        SJMP AGAIN
DELAY:MOV TIMED,#10H
        MOV TH1,#5fH
        MOV TL1,#4fH
        CLR TF1
        SET B TR1
MONITOR:JNB Tf1,MONITOR
        CLR TR1
        CLR Tf1
        RET
        END
```

OUTPUT:



C) TOGGLING THE LED WITHOUT INTERRUPT IN C CODE:

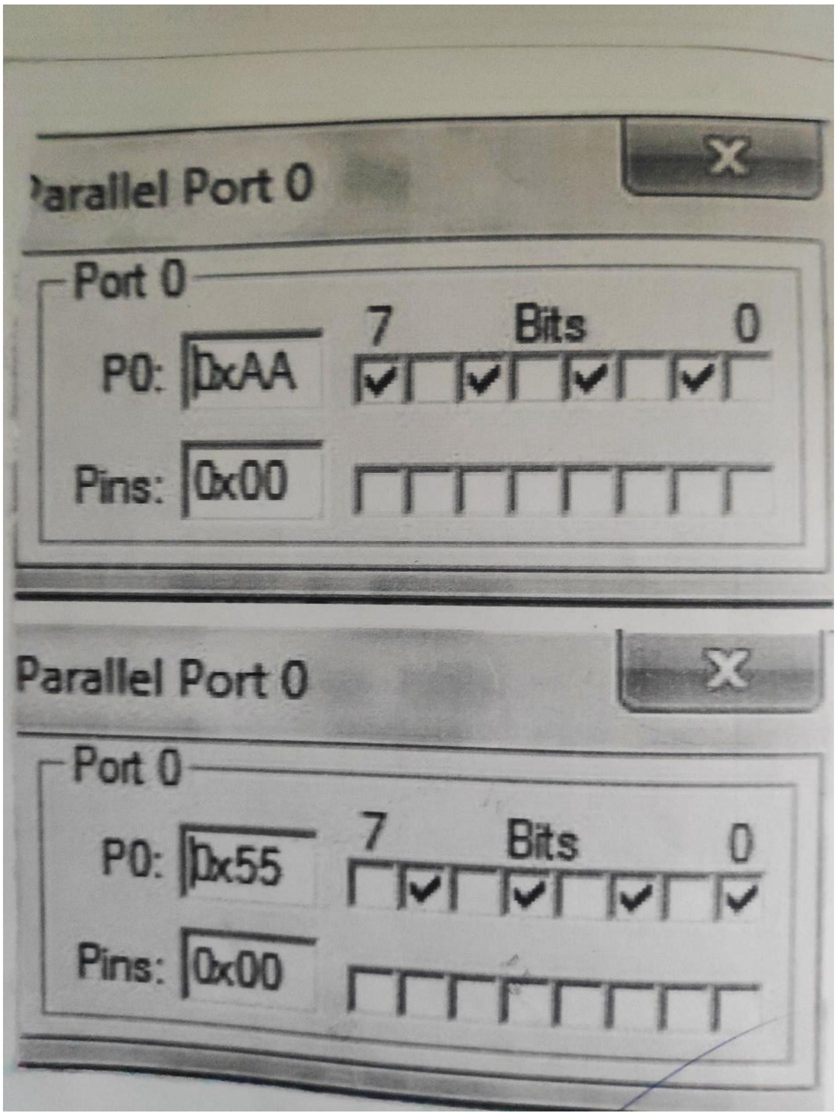
ALGORITHM:

- Assign the port,with some value.
- Assign the accumulator with some other value.
- Toggle the bit of port 1,by calling delay function and complementary accumulator.

C CODE:

```
#include<reg51.h>
Void delay()
{
Int I,j;
For(i=0;I<20;I++);
For(j=0;j<20;j++);
}
Void main()
{
P1=0x00;
While(1)
{
P1=0x55;
Delay();
P1=0xaa;
Delay();
}
}
```

OUTPUT:



D) TOGGLING THE LED WITH INTERRUPT IN C CODE:

ALGORITHM:

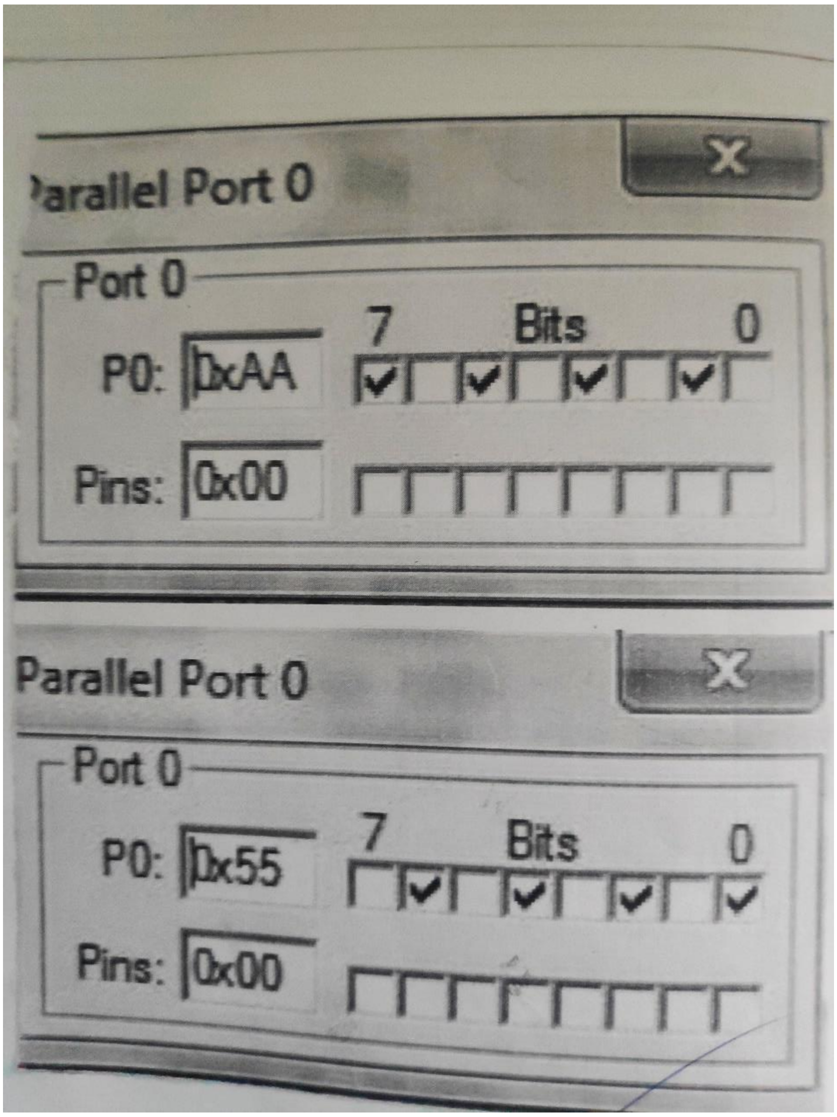
- Assign a value to accumulator.
- Complement the accumulator and store it in port1.
- Delay created using timers.

C CODE:

```
#include<reg51.h>
Void delay()
{
TMOD=0x10;
TM1=0x5f;
TML1=0x4f;
TF1=0;
While(TF!=0)
{
TR1=0;
TF1=0;
}
}
Void main()
{
P1=0x00;
While(1)
{
P1=0xaa;
Delay();
P1=0x55;
Delay();
}
```

}

OUTPUT:



RESULT:

Toggling the LED using assembly & C code is successfully executed.

9.SERIAL DATA TRANSFER AND RECIEVER IN KEIL SOFTWARE

AIM:

To write a program to transfer and receive data in serial port in KEIL software.

A)SERIAL DATA TRANSFER IN C:

ALGORITHM:

- Configure time/auto reload node.
- Load TH1 with value for 4800 baud rate load oxFD.
- Load SCON with serial mode and control bits ie:0x50(mode 1 erable reception)
- Start timer 1
- Load transmitting data in SBUF register.
- Wait until load data is completely transmitted by TI flag.
- When TI flag is set,clear it and repeat from step 5 to transmit more data.

C CODE:

```
#include<reg51.h>
Void main()
{
TMOD=0x20;
TM1=0xFA;
SCON=0x50;
TR1=1;
While(1)
{
```

```
TI=0;
SBUF='A';
While(TI==0);
}
}
```

OUTPUT:

UART

Hello Hello

B)SERIAL DATA RECEIVER IN C:

ALGORITHM:

- Configure time/auto reload node.

- Load TH1 with value for 4800 baud rate load oxFD.
- Load SCON with serial mode and control bits ie:0x50(mode 1 erable reception)
- Start timer 1
- Wait till the data is received.RI will be set once the data is received in SBUF register.
- Clear the receiver flag(RI)for next cycle.
- Copy/read the received data from SBUF register.

C CODE:

```
#include<reg51.h>
Void main()
{
Unsigned char m;
TMOD=0x20;
TM1=0xFA;
SCON=0x50;
TR1=1;
While(1)
{
RI=0;
While(RI==0);
m=SBUF;
}
}
```

OUTPUT:

Command:

Sin=’x’

Sin=’Y’

UART

Hello Hello

RESULT:

Serial data transfer and receiver in KEIL software is executed successfully.

C)ASSEMBLY CODE FOR SERIAL DATA TRANFER :

SOURCE CODE:

```
ORG OOH
MOV TMOD,#20H
MOV TH1,#6
MOV SCON,#50H
```

```
        SETB TR1
  AGAIN:MOV STAY,'A'
   HERE:JNB TI,HERE
        CLR TI
        SJMP AGAIN
        END
```

OUTPUT:

UART
Hello Hello

10.STUDY OF STEPPER MOTOR

AIM:

To study the working of a stepper motor.

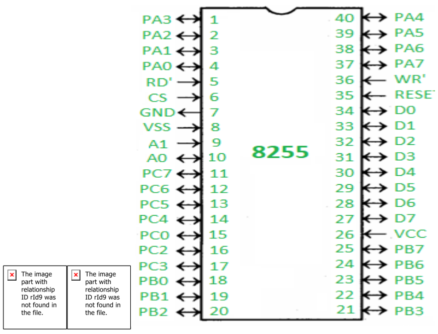
STEPPER MOTOR:

A stepper motor is a brush less DC motor that divides a full rotation into a number of equal steps.

STEPPER MOTOR INTERFACE CARD:

- Stepper motor interface card controls 4 phase stepper motor of torque ranging from 2kgkm to 4kgkm.
- For phase stepper motor is controlled through four input called A,A,B,B and applying the pulses in specific sequence to rotate either in clockwise or in anti clockwise direction.
- The 4 phase pulses of specific sequence may be generated by discrete logic or by microcontroller circuit.
- Simplest way is by using peripheral interface chip 8255.since two phases A and B are complement of A and B it is efficient if two wave forms are generated using simple inverts.
- PAO PA1 and PA2 controls stepper motor 2PA2 controls apply to motor.

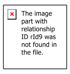
PIN DIAGRAM:

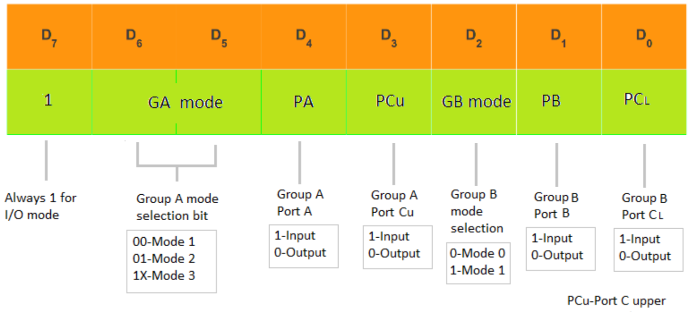


PORT ADDRESS OF 8255:

8255 ADDRESS		8051	
		BOTTOM	**TOP**
Control work register		6043H	6003H
Port A address		6040H	6000H
Port B address		6041H	6001H
Port C address		6042H	6002H

8255 CONTROL WORD FORMAT (I/O MODEL):





8255 Control Word For I/O mode

STEP SEQUENCE FOR 1 STEP CLOCKWISE ROTATION:

STEP 1	**1**	**0**	**1**	**= 05**
STEP 2	**1**	**1**	**1**	**= 07**
STEP 3	**1**	**1**	**0**	**= 06**
STEP 4	**1**	**0**	**1**	**= 04**

STEP SEQUENCE FOR 2 STEP ANTICLOCKWISE ROTATION:

STEP 1	**1**	**0**	**0**	**= 04**
STEP 2	**1**	**1**	**0**	**= 06**
STEP 3	**1**	**1**	**1**	**= 07**
STEP 4	**1**	**0**	**1**	**= 05**

SPECIFICATION:

STEPPER MOTOR CONTROLLER CARD:

- No.of.stepper motor control:2
- Interface to microcontroller kid:through 26 pin connector.
- Step rate:programmable.

- No.of.steps:programmable.

STEPPER MOTOR:

- Type::hybrid permanent magnet rotor.
- Phase::far.
- Step angle::18 degree per step.
- Switching sequence ::4 step.
- Operating temperature:0 to 50 degree celsius.

STEPPER MOTOR - CLOCKWISE DIRECTION:

AIM:

To write code in assembly language to rotate stepper motor in clockwise direction.

ALGORITHM:

- Move control word register add to DPTR location.
- Move A to DPTR location.
- Move port A address of 8255 to DPTR location.
- Move 1,2,3,4 step.sequence into A for clockwise direction.
- Return to code.

OPCODE	ADDRESS	LABEL	MNEUMONICS	DESCRIPTION
7A FF	9100	DELAY	MOV R2,#FF	Delay routine
7B 0A	9102	DLY 1	MOV R3,#OA	Delay routine
DB FE	9104	DLY 2	DJNZ R3,DLY 2	Delay routine
DA FA	9106		DJNZ R2,DLY 1	Delay routine
22	9108		RET	Delay routine

90 60 08	9000		MOV DPTR,#6003	Control port of 8051.
74 80	9003		MOV A,#80H	Move 80 value to A.
F0	9005		MOVX @DPTR,A	Move a value to accumulator.
90 60 00	9006		MOV DPTR,#6000	Move 6000 to DPTR.
74 04	9009	LOOP	MOV A,#04H	First step sequence.
F0	900B		MOVX @DPTR,A	Move A to DPTR.
12 91 00	900C		LCALL DELAY	Call the delay routine.
74 06	900F		MOV A,#06H	Second step sequence.
F0	9011		MOVX @DPTR,A	Move A to DPTR.
12 91 00	9012		LCALL DELAY	Call the delay routine.
74 07	9015		MOV A,#07H	Third step sequence.
F0	9017		MOVX @DPTR,A	Move A to DPTR.
12 91 00	9018		LCALL DELAY	Call delay routine.
74 05	901B		MOV A,#05H	Fourth step sequence.
F0	901D		MOVX @DPTR,A	Move A to DPTR.
12 91 00	901E		LCALL DELAY	Call delay routine.
02 09 09	9021		SJUMP LOOP	

STEPPER MOTOR - ANTICLOCKWISE DIRECTION:

AIM:

To write code in assembly language to rotate stepper motor in anticlockwise direction.

ALGORITHM:

- Set the control port of 8255.
- Set all the port and bits as output.
- Move each step sequence for anticlockwise rotation into port A.
- Call the delay routine for each step sequence.
- Long jump to first sequence of rotation.

OPCODE	ADDRESS	LABEL	MNEUMONICS	DESCRIPTION
7A FF	9100	DELAY	MOV R2,#FF	Delay routine
7B 0A	9102	DLY 1	MOV R3,#OA	Delay routine
DB FE	9104	DLY 2	DJNZ R3,DLY 2	Delay routine
DA FA	9106		DJNZ R2,DLY 1	Delay routine
22	9108		RET	Delay routine
90 60 08	8500		MOV DPTR,#6003	Control port of 8051.
74 80	8503		MOV A,#80H	Move 80 value to A.
F0	8505		MOVX @DPTR,A	Move a value to accumulator.
90 60 00	8506		MOV DPTR,#6000	Move 6000 to DPTR.
79 20	8509	LOOP	MOVR1,#20	
7A 20	850B		MOV R2,#20	
74 05	850D		MOV A,#05H	First step sequence.

F0	850F		MOV @DPTR,A	Move A to DPTR.
12 86 00	8510		LCALL DELAY	Call the delay routine.
74 07	8513		MOV A,#07H	Second step sequence.
F0	8515		MOVX @DPTR,A	Move A to DPTR.
12 86 00	8516		LCALL DELAY	Call the delay routine.
74 06	8519		MOV A,#06H	Third step sequence.
F0	851B		MOVX @DPTR,A	Move A to DPTR.
12 86 00	851C		LCALL DELAY	Call delay routine.
74 04	851F		MOV A,#04H	Fourth step sequence.
F0	8521		MOVX @DPTR,A	Move A to DPTR.
12 86 00	8522		LCALL DELAY	Call delay routine.
D9 E6	8525		DJNZ R1,850D	
74 04	8527		MOV A,#06	
F0	8529		MOVX @DPTR,A	
12 86 00	852A		LCALL DELAY	
02 09 09	852F		SJMP LOOP	

STEPPER MOTOR - BOTH THE DIRECTION:

AIM:

To write code in assembly language to rotate stepper motor in both the clockwise and anticlockwise direction.

ALGORITHM:

- Set the control port of 8255.
- Set all the port and bits as output.
- Move each step sequence for clockwise and anticlockwise rotation in the specific angle.
- Call the delay routine for each step sequence.

OPCODE	ADDRESS	LABEL	MNEMONICS
F0	852F		MOVX @DPTR,A
12 86 00	8530		ACALL DELAY
74 07	8533		MOV A,#07
F0	8535		MOVX @DPTR,A
12 86 00	8536		ACALL DELAY
74 05	8539		MOV A,#05
F0	853B		MOVX @DPTR,A
12 86 00	853C		ACALL DELAY
DA E6	853F		DJNZ R2,8527
02 85 09	854		LJMP LOOP
7B EF	8600	DELAY	MOV R3,#0A
7C 0A	8602	DLY 1	MOV R4,#0A
DC FE	8604	DLY 2	DJNZ R4,DLY 2
DB FA	8606		DJNZ R3,DLY 1

22	8608		RET

OPCODE	ADDRESS	LABEL	MNEUMONICS	DESCRIPTION
90 60 08	9000		MOV DPTR,#6003	Control port of 8051.
74 80	9003		MOV A,#80	Move 80 value to A.
F0	9005		MOVX @DPTR,A	Move a value to accumulator.
90 60 00	9006		MOV DPTR,#6000	Move 6000 to DPTR.
74 04	9009	LOOP	MOV A,#05H	First step sequence.
F0	900B		MOVX @DPTR,A	Move A to DPTR.
12 91 00	900C		LCALL DELAY	Call the delay routine.
74 06	900F		MOV A,#07H	Second step sequence.
F0	9011		MOVX @DPTR,A	Move A to DPTR.
12 91 00	9012		LCALL DELAY	Call the delay routine.
74 07	9015		MOV A,#08H	Third step sequence.
F0	9017		MOVX @DPTR,A	Move A to DPTR.
12 91 00	9018		LCALL DELAY	Call delay routine.
74 05	901B		MOV A,#04H	Fourth step sequence.

F0	901D		MOVX @DPTR,A	Move A to DPTR.
12 91 00	901E		LCALL DELAY	Call delay routine.
02 09 09	9021		SJUMP LOOP	

RESULT:

Thus the rotation of stepper motor in all specific angle is successfully executed.

11.DAC INTERFACE

AIM:

To write the assembly code to generate the following waves using DAC interface:

- Square wave
- Triangular wave
- Saw tooth wave.

SQUARE WAVE GENERATION:

ADDRESS	OPCODE	MNEUMONICS	DESCRIPTION
9000	74 80	MOV A,#80	All the ports are output.
9002	90 60 03	MOV DPTR,#6003	Control port of 8255
9005	F0	MOVX @DPTR,A	All the bits are

			output
9006	90 60 01	MOV DPTR,#6001	Port B address
LOOP:9009	74 00	MOV A,#00	0 volt of wave
900B	F0	MOVX @DPTR,A	Move to port B
900C	12 91 00	LCALL 9100	Call delay
900F	74 33	MOV A,#33	2 volts of wave
9011	F0	MOV @DPTR,A	Move to port B
9012	12 91 00	LCALL 9100	Call delay
9015	80 F2	SJMP 9007	Short jump back to 0 volt wave
9100	7B FF	MOV R3,#FF	Move FF to R3
HERE:9102	DB FE	DJNZ R3,9102	Decrement R3
9104	22	RET	Return back to main routine.

TRIANGULAR WAVE GENERATION:

ADDRESS	OPCODE	MNEUMONICS	DESCRIPTION
8500	74 80	MOV A,#80	All the ports are output.
8502	90 60 03	MOV DPTR,#6003	Control port of 8255
8505	F0	MOVX @DPTR,A	All the bits are output
8506	90 60 01	MOV DPTR,#6001	Port B address
LOOP:8509	74 00	MOV A,#00	0 volt of wave
850B	F0	MOVX @DPTR,A	Move to port B
850C	24 01	ADD A,#01	Add 1 volt to A

850E	84 FF AA	CJNE A,#FF,850B	Compare jump if not equal to 850B
AGAIN:8511	94 01	SUBB A,#01	Subtract 1 volt
8513	F0	MOVX @DPTR,A	Move to port B
8517	B4 00 FA	CJNE A,#00,8511	Compare jump if not equal to 8511
8517	80 F0	SJMP 8509	Short jump back

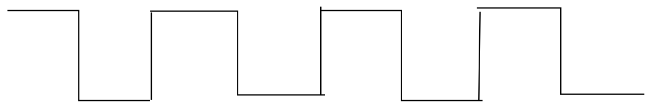
SAWTOOTH WAVE GENERATION:

ADDRESS	OPCODE	MNEUMONICS	DESCRIPTION
8600	74 80	MOV A,#80	All the ports are output.
8602	90 60 03	MOV DPTR,#6003	Control port of 8255
8605	F0	MOVX @DPTR,A	All the bits are output
8606	90 60 01	MOV DPTR,#6001	Port B address
LOOP:8609	74 00	MOV A,#00	0 volt of wave
AGAIN:860B	F0	MOVX @DPTR,A	Move to port B
860C	24 01	ADD A,#01	Add 1 volt to A
860E	84 FF AA	LCALL 8700	Call delay
8611	94 01	CJNE A,#FF,8608	Compare and jump not equal to 860B
8614	F0	MOV A,#00	Move 0 to A
8614	B4 00 FA	MOVX @DPTR,A	Move to port B
8617	80 F0	LCALL 8700	Call delay
861A		SJMP 8609	Infinite loop

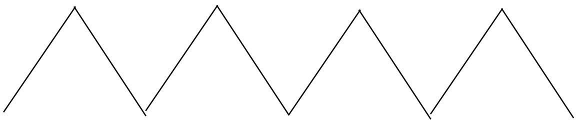
8700		MOV R3,#10	Move 10 to R3
8702		DJNZ R3,8702	Decrement R3 & jump to 8702
8704	22	RET	return

OUTPUT:

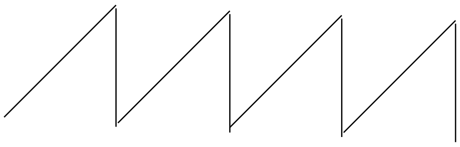
SQUARE WAVE GENERATION:



TRIANGULAR WAVE GENERATION:



SAWTOOTH WAVE:



RESULT:

Thus the following waves are generated using DAC interface.

12.LCD INTERFACE WITH 8051

AIM:

To with an assembly code to display alphabets in two words using LCD interface with 8051.

ASSEMBLY CODE:

ADDRESS	OPCODE	LABEL	MNEUMONICS	DESCRIPTION
8500	12 90 00		LCALL INIT	
8503	90 60 00		MOV DPTR,#6000	Port A
8506	74 01		MOV A,#01H	
8508	F0		MOVX @DPTR,A	
8509	12 91 00		LCALL LCDENA	
850C	90 60 00		MOV DPTR,#6000	
850F	74 85		MOV A,#85H	Move cursor to middle of 1st line
8511	F0		MOVX @DPTR,A	
8512	12 91 00		LCALL LCDENA	
8515	D1 00		ACALL PRINT	Print the name
8517	90 60 00		MOV DPTR,#6000	Port A
851A	74 C4		MOV A,#C4	
851C	F0		MOVX @DPTR,A	Cursor to goto 2nd line
851D	12 91 00		LCALL LCDENA	

8520	90 60 00		MOV DPTR,#6000	
8523	D1 0C		ACALL PRINT	
8525	12 00 BB		LCALL 00BB	
8600	76 08	PRINT	MOV R2,#04H	
8602	78 41		MOV R0,#41	
8604	E6		MOV A,@R0	
8605	F0		MOVX @DPTR,A	
8606	12 93 00		LCALL LCDEN ADATA	
8609	08		INC R0	
860A	DA 04		DJNZ R2,LOOP	
860C	7A C4	PRINT 1	MOV R2,#C4H	
860E	78 51		MOV R0,#51H	
8610	E6	LOOP 1	MOV A,@R0	
8611	F0		MOVX @DPTR,A	
8612	12 93 00		LCALL LCDENADATA	
8615	08		INC R0	
8616	DA F8		DJNZ R2,LOOP	
8618	22		RET	
9000	90 60 03	INIT	MOV DPTR,6008	Control word register to 8255
9003	74 80		MOV A,#80	
9005	F0		MOVX @DPTR,A	

9006	90 60 00		MOV DPTR,#6000	Port A data
9009	74 34		MOV A,#38	Function set to enable display on / off
900B	F0		MOVX @DPTR,A	
900C	12 91 00		LCALL LCDENA	Function set
900F	74 08		MOV A,#08H	
9011	F0		MOVX @DPTR,A	
9012	12 91 00		LCALL LCDENA	
9015	74 0F		MOV A,#0FH	Display on/off
9017	F0		MOVX 2DPTR,A	
9018	12 91 00		LCALL LCDENA	
901B	74 06		MOV A,#06H	Configure entry mode
901D	F0		MOV @DPTR,A	
901E	12 911 00		LCALL LCDENA	
9021	22		RET	
9100	90 60 02	LCDENA	MOV DPTR,#6002	Port c
9103	74 02		MOV A,#02H	LCDENA
9105	F0		MOVX @DPTR,A	
9106	74 00		MOV A,#00H	LCD display
9108	F0		MOVX @DPTR,A	
9109	12 92 00		LCALL	

			DELAY	
910C	90 60 00		MOV DPTR,#6000	Port A
910F	22		RET	
9300	90 60 02	LCDENADATA	MOV DPTR,#6002	PORT C
9303	74 03		MOV A,#03H	Enable LCD
9305	F0		MOV @DPTR,A	Command register
9306	74 00		MOV A,#00H	Display the data
9308	F0		MOVX @DPTR,A	
9309	12 92 00		LCALL DELAY	Call the delay
930C	96 60 00		MOV DPTR,#600H	Port A
930F	0F 22		RET	
9200	70 FF	DELAY	MOV R5,#FFH	Delay routine.
9202	DD FE	DLY 1	DJNZ R5,DLY1	
9204	22		RET	

OUTPUT:

ABCD

XYZ

RESULT:

Thus the alphabets and number are displayed in the LCD with 8051 successfully.

AIM:

To write an assembly code to display alphabets and number using LCD interfacing.

ASSEMBLY CODE:

ADDRESS	LABEL	OPCODE	MNEUMONICS	DESCRIPTION
8900	MAIN	12 89 42	LCALL INIT	
8903		90 60 00	MOV DPTR,#6000	PORT A
8906		74 01	MOV A,#01	
8908		F0	MOVX @DPTR,A	MOVE A TO DPTR
8909		12 91 00	LCALL LCDENA	
890C		79 00	MOV R1,#00	PORT B
890E		74 41	MOV A,#41	CHAR ‘A’
8910		F0	MOVX @DPTR,A	MOVE VALUE AT A TO DPTR
8911		12 89 69	LCALL LCDENA	
8913		74 42	MOV A,#42	CHAR ‘B’
8916		F0	MOVX @DPTR,A	MOVE VALUE AT A TO DPTR
8917		12 89 69	LCALL LCDENA	
891A		74 43	MOV A,#43	CHAR ’C’
891C		F0	MOVX @DPTR,A	MOVE VALUE AT A TO DPTR

891D		12 89 69	LCALL LCDENA	
8920		12 86 42	LCALL INIT	
8923		90 60 00	MOV DPTR,#6000	PORT B
8926		74 0C	MOV A,#0C	GOTO 2ND LINE
8928		F0	MOVX @DPTR,A	A TO DPTR
8929		12 89 79	LCALL LCDENADATA	
892C		90 60 00	MOV DPTR,#6000	PORT A
892F		74 32	MOV A,#31	NUM’1’
8931		F0	MOVX @DPTR,A	A TO DPTR
8932		12 89 69	LCALL LCDENA	
8935		74 32	MOV A,#32	NUM’2’
8937		F0	MOVX @DPTR,A	A TO DPTR
8935		12 89 69	LCALL LCDENA	
893B		74 33	MOV A,#33	NUM’3’
893D		F0	MOVX @DPTR,A	A TO DPTR
893E		12 89 69	LCALL LCDENA	
841		12 90 60	LCALL 00BB	

OUTPUT:

ABC123

RESULT:

Thus the alphabets and number are displayed in the LCD with 8051 successfully.