ANNAMALAI UNIVERSITY

FACULTY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF ELECTRICAL ENGINEERING

B.E (EEE)

V - SEMESTER

MICROPROCESSOR AND MICROCONTROLLER LAB

Name……

Reg. No……..

Semester……Batch …….

DEPARTMENT OF ELECTRICAL ENGINEERING

VISION

To develop the Department into a "Centre of Excellence" with a perspective to provide quality education and skill-based training with state-of-the-art technologies to the students, thereby enabling them to become achievers and contributors to the industry, society and nation together with a sense of commitment to the profession.

MISSION

M1: To impart quality education in tune with emerging technological developments in the field of Electrical and Electronics Engineering.

M2: To provide practical hands-on-training with a view to understand the theoretical concepts and latest technological developments.

M3: To produce employable and self-employable graduates.

M4: To nurture the personality traits among the students in different dimensions emphasizing the ethical values and to address the diversified societal needs of the Nation

M5: To create futuristic ambience with the state-of-the-art facilities for pursuing research.

EECP 507 MICROPROCESSOR AND MICROCONTROLLER LAB				
EXP. NO	**DATE**	**EXPERIMENT NAME**	**MARKS**	**REMARKS**
1		Study of 8085 Microprocessor a) Finding out the largest and smallest number b) Sorting an array		
2		Study of 8051 Microcontroller a) Arithmetic Operations b) Code Conversion		
3		Study of 8097 Microcontroller a)Arithmetic Operations b)Logical Operations		
4		Study of Programmable Peripheral Interface 8255		
5		Serial Data Communication using USART 8251 and Timer 8253		
6		Seven Segment LED Display using 8051 Microcontroller		
7		Stepper Motor Control using 8051 Microcontroller		
8		Study of Keyboard Display Interface 8279 using 8051 Microcontroller		
9		Applications of 8097 Microcontroller a) DAC b) ADC c) PWM Generation		
10		Serial Data Communication Between Two 8051 Kits		

EX. No: **DATE:**

STUDY OF 8085 MICROPROCESSOR

Aim

To study about the 8085 microprocessor and to execute the following programs on 8085 microprocessor kit.

(i) To find the largest and smallest number in an array.
(ii) To arrange the numbers in ascending and descending order.

Theory

Intel 8085 is an 8-bit microprocessor. It is a 40 pin IC package. It uses a single +5v power supply. Its clock speed is about 3 MHz.

8085 System Bus

A typical microprocessor communicates with memory and other devices (input and output) using three busses: Address Bus, Data Bus and Control Bus.

Arithmetic Logic Unit (ALU)

ALU performs the actual numerical and logic operation such as ‘add’, ‘subtract’, ‘AND’, ‘OR’ etc.

Registers

The 8085 microprocessor includes six registers, one accumulator, and one flag register. In addition, it has two 16-bit registers, the stack pointer and the program counter.

Instruction Register/Decoder

Latest instruction is sent to Instruction register from memory, prior to execution. So it is the temporary store for the current instruction of a program. Decoder, then decodes or interprets the instruction.

Timing and control unit

This unit synchronizes all the microprocessor operations with the clock and generates the control signals necessary for communication between the microprocessor and peripherals.

Interrupt control

8085 microprocessor has 5 interrupts used to interrupt a program execution. They are

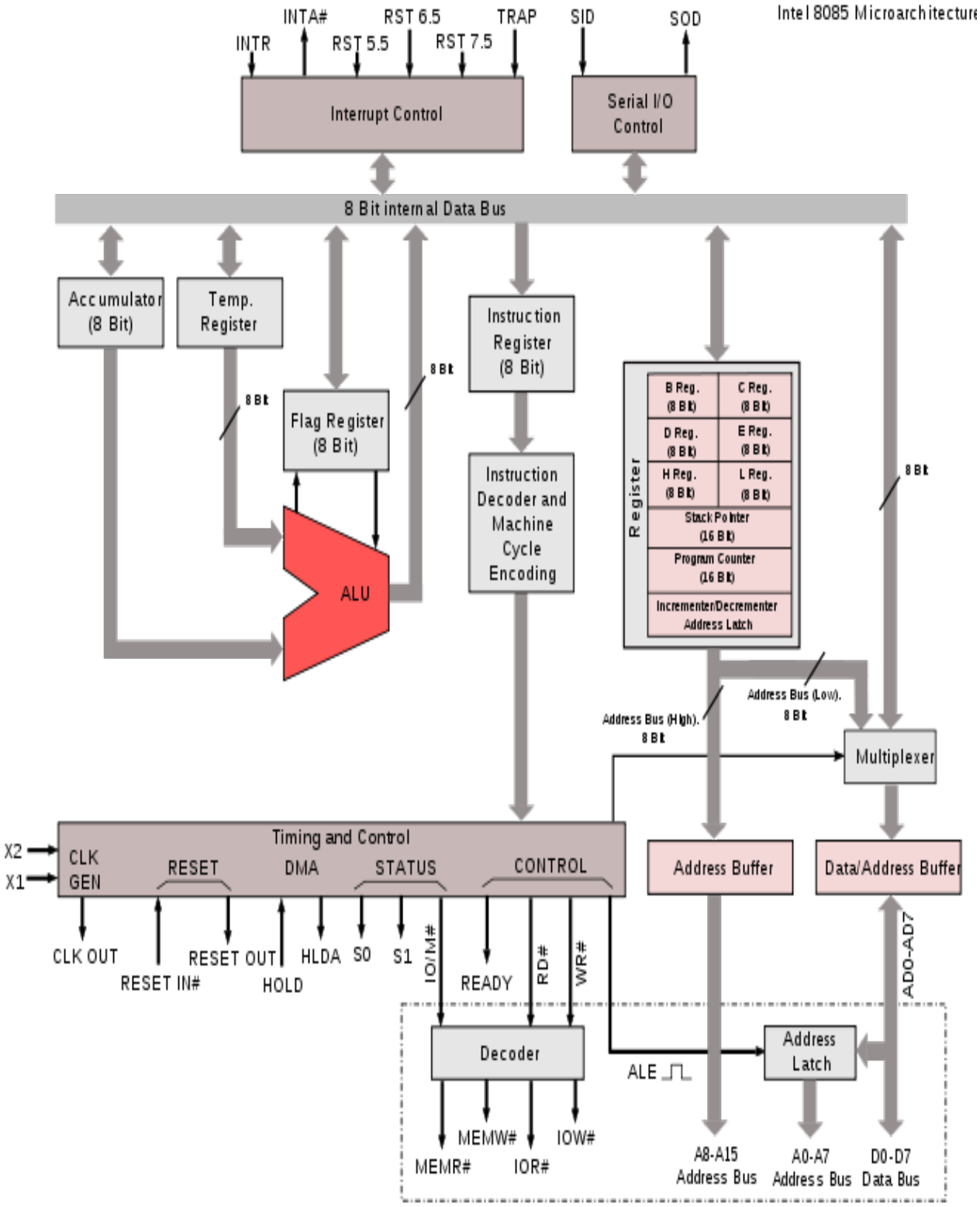
INTR, RST5.5, RST6.5, RST7.5, TRAP. TRAP is a non maskable interrupt and other interrupts are maskable.

Serial I/O control

8085 microprocessor has 2 signals SID and SOD to implement the serial transmission.

(i) SID-Serial Input Data line SOD-Serial Output Data line

Architecture details of 8085 Microprocessor



Architecture of 8051 Microcontroller

Program 1

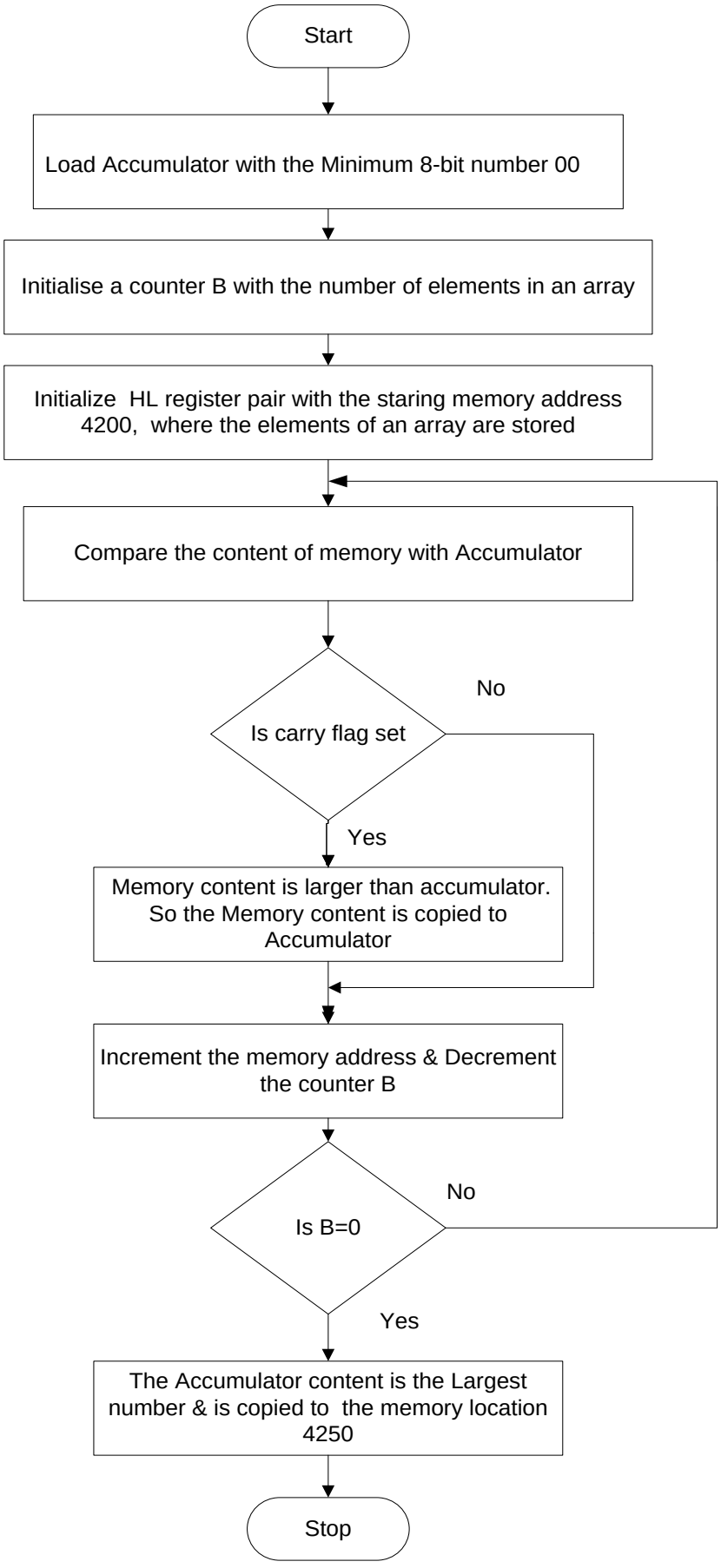
To find the largest number in an array.

ADDRESS	LABEL	OPCODE	MNEMONICS	COMMENTS
4100			MVI A,00	Move the Accumulator with the smallest 8 bit number 00
4102			MVI B,05	Initialize the B register as counter with the array size
4104			LXI H, 4200	Load HL register pair with the memory address 4200
4107	LOOP1		CMP M	Compare the content of memory with Accumulator
4108			JNC LOOP	Jump on no carry to the memory address specified by the label LOOP
410B			MOV A,M	Move the bigger number available in memory to Accumulator
410C	LOOP		INX H	Increment HL register pair
410D			DCR B	Decrement B register
410E			JNZ LOOP1	If the B register is not equal to zero, then jump to the memory address specified by the label LOOP1
4111			STA 4250	Store the Accumulator content (i.e. the largest number) in the memory address 4250
4114			HLT	Halt

Data **Result**

4200		4250	
4201			
4202			
4203			
4204			

Flowchart- To find the largest number in an array.



Program 2

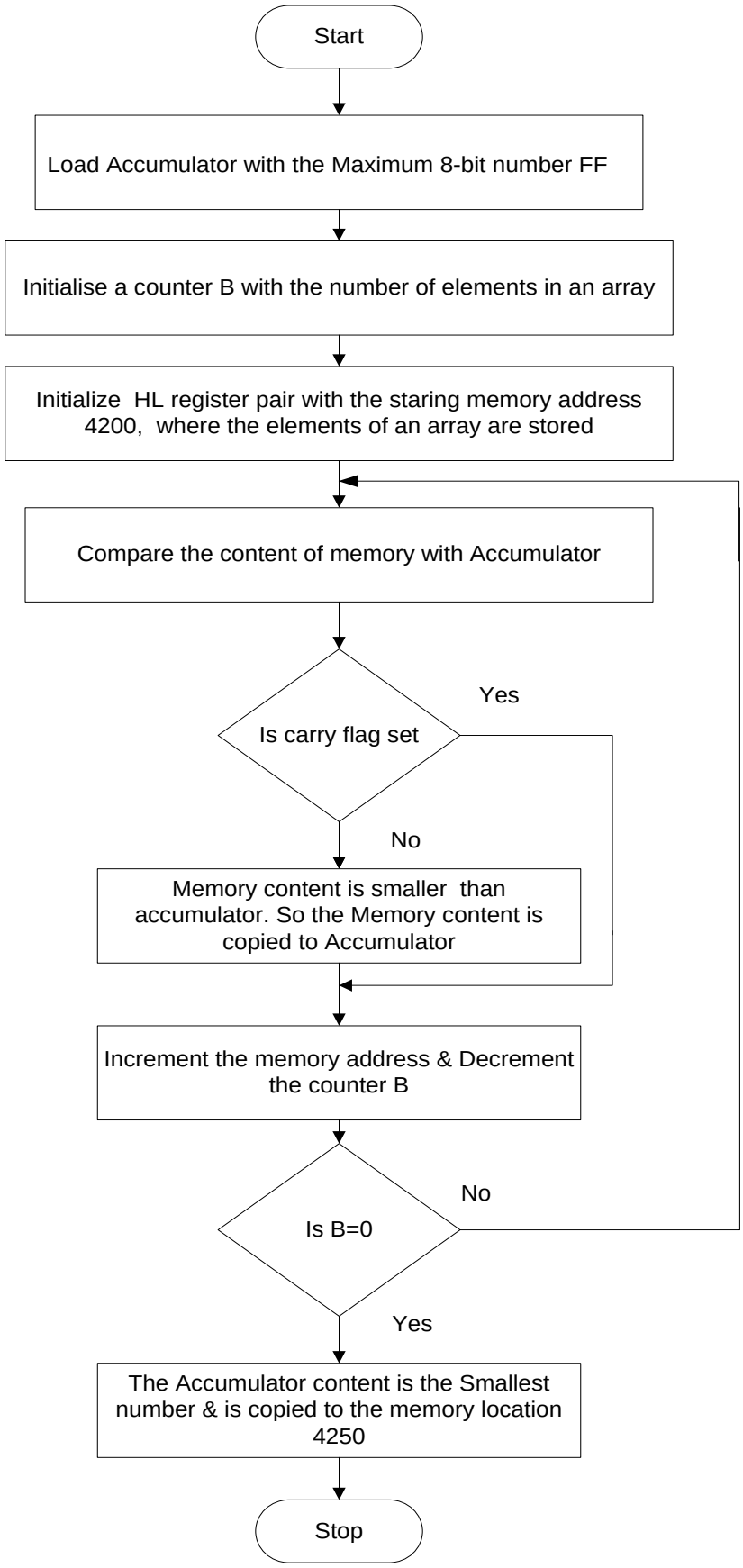
To find the smallest number in an array.

ADDRESS	LABEL	OPCODE	MNEMONICS	COMMENTS
4100			MVI A,FF	Move the Accumulator with the largest 8 bit number FF
4102			MVI B,05	Initialize the B register as counter with the array size
4104			LXI H, 4200	Load HL register pair with the memory address 4200
4107	LOOP1		CMP M	Compare the content of memory with Accumulator
4108			JC LOOP	Jump on carry to the memory address specified by the label LOOP
410B			MOV A,M	Move the smaller number available in memory to Accumulator
410C	LOOP		INX H	Increment HL register pair
410D			DCR B	Decrement B register
410E			JNZ LOOP1	If the B register is not equal to zero, then jump to the memory address specified by the label LOOP1
4111			STA 4250	Store the Accumulator content (i.e. the smallest number) in the memory address 4250
4114			HLT	Halt

Data **Result**

4200		4250	
4201			
4202			
4203			
4204			

Flowchart-To find the smallest number in an array

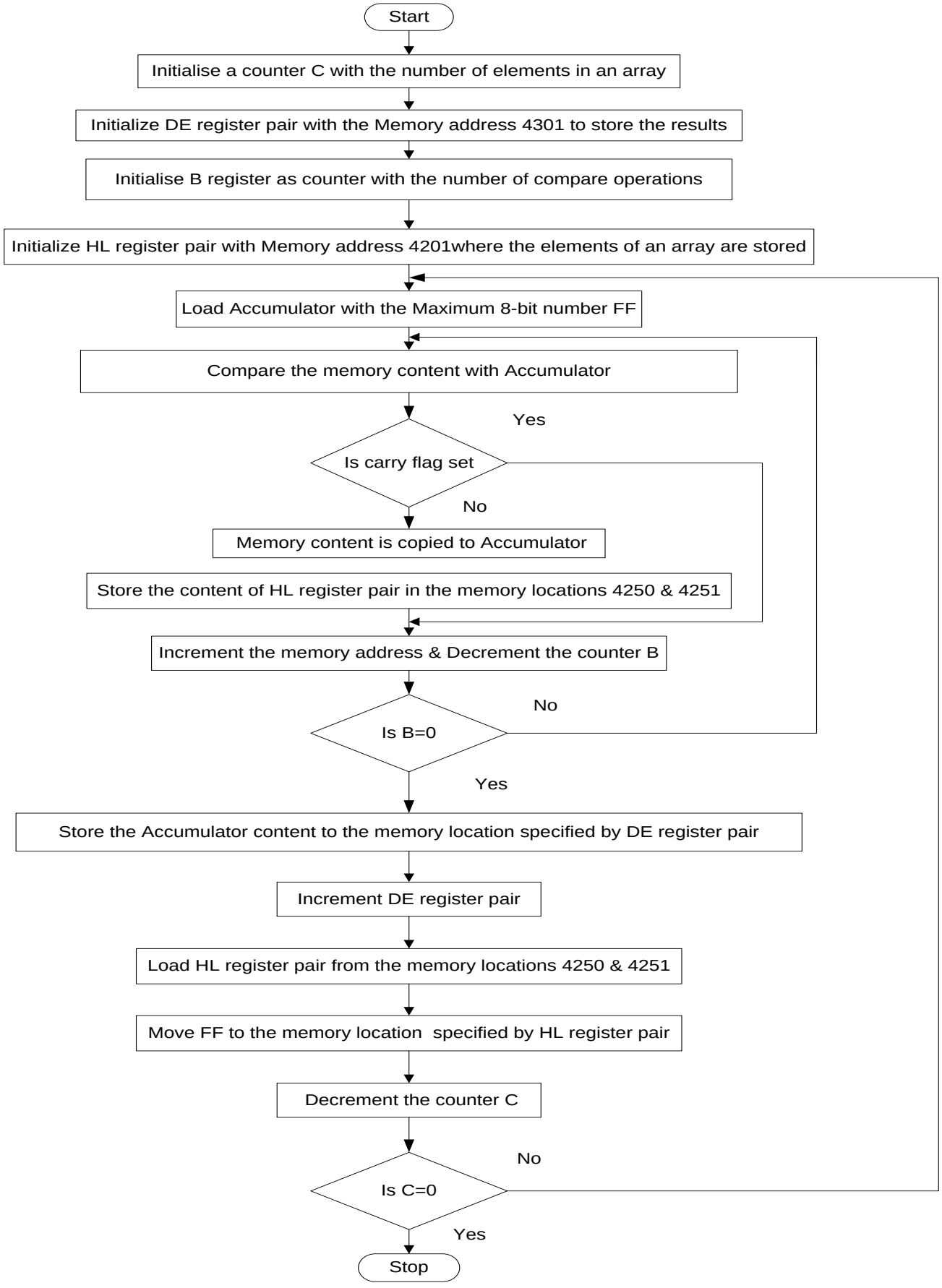


Program 3

To sort an array of numbers in ascending order.

ADDRESS	LABEL	OPCODE	MNEMONICS	COMMENTS
4100			MVI C,05	Initialize the C register as counter with the array size
4102			LXI D, 4301	Load DE register pair with the memory address 4301
4105	LOOP2		MVI B,05	Initialize the B register as counter with the number of compare operations
4107			LXI H, 4201	Load HL register pair with the memory address 4201
410A			MVI A,FF	Move the Accumulator with the largest 8 bit number FF
410C	LOOP1		CMP M	Compare the content of memory with Accumulator
410D			JC LOOP	Jump on carry to the memory address specified by the label LOOP
4110			MOV A,M	Move the smaller number available in memory to Accumulator
4111			SHLD 4250	Store HL register pair in 4250 & 4251
4114	LOOP		INX H	Increment HL register pair
4115			DCR B	Decrement B register
4116			JNZ LOOP1	If the B register is not equal to zero, then jump to the memory address specified by the label LOOP1
4119			STAX D	Store the Accumulator content (i.e. the smallest number) in the memory address 4250
411A			INX D	Increment HL register pair
411B			LHLD 4250	Load HL register pair from 4250 & 4251

Flowchart -To sort an array of numbers in ascending order.



411E			MVI M, FF	Move FF to memory location where the smallest number is located
4120			DCR C	Decrement C register
4121			JNZ LOOP2	If the C register is not equal to zero, then jump to the memory address specified by the label LOOP2
4124			HLT	Halt

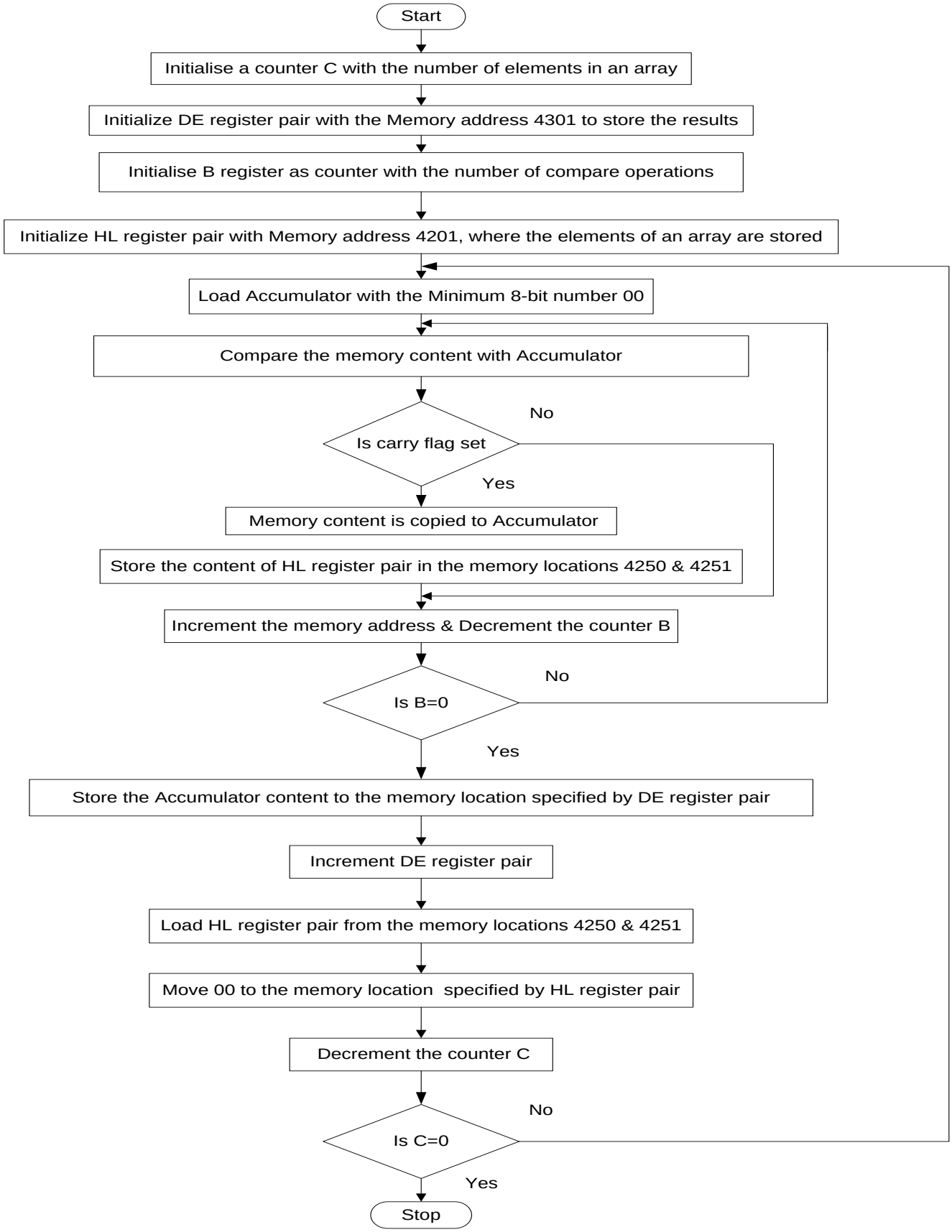
Data

4201	
4202	
4203	
4204	
4205	

Result

4301	
4302	
4303	
4304	
4305	

Flowchart- Sorting an array of numbers in descending order.



Program 4

To sort an array of numbers in descending order.

ADDRESS	LABEL	OPCODE	MNEMONICS	COMMENTS
4100			MVI C,05	Initialize the C register as counter with the array size
4102			LXI D, 4301	Load DE register pair with the memory address 4301
4105	LOOP2:		MVI B,05	Initialize the B register as counter with the number of compare operations
4107			LXI H, 4201	Load HL register pair with the memory address 4201
410A			MVI A,00	Move the Accumulator with the smallest 8 bit number 00
410C	LOOP1:		CMP M	Compare the content of memory with Accumulator
410D			JNC LOOP	Jump on no carry to the memory address specified by the label LOOP
4110			MOV A,M	Move the smaller number available in memory to Accumulator
4111			SHLD 4250	Store HL register pair in 4250 & 4251
4114	LOOP:		INX H	Increment HL register pair
4115			DCR B	Decrement B register
4116			JNZ LOOP1	If the B register is not equal to zero, then jump to the memory address specified by the label LOOP1
4119			STAX D	Store the Accumulator content (i.e. the smallest number) in the memory address 4250
411A			INX D	Increment HL register pair
411B			LHLD 4250	Load HL register pair from 4250 & 4251

411E			MVI M, 00	Move 00 to memory location where the largest number is located
4120			DCR C	Decrement C register
4121			JNZ LOOP2	If the C register is not equal to zero, then jump to the memory address specified by the label LOOP2
4124			HLT	Halt

Data

4201	
4202	
4203	
4204	
4205	

Result

4301	
4302	
4303	
4304	
4305	

Result

The above programs were executed and their results are verified using 8085 microprocessor.

EX.NO: **DATE:**

STUDY OF MICROCONTROLLER-8051

AIM:

To perform various arithmetic operations using 8051 microcontroller.

THEORY:

It is 8-bit microcontroller, means MC 8051 can Read, Write and Process 8 bit data. This is mostly used microcontroller in the robotics, home appliances like mp3 player, washing machines, electronic iron and industries. Mostly used blocks in the architecture of 8051 are as follows:

MC 8051 has 128 byte Random Access memory for data storage. Random access memory is non volatile memory. During execution for storing the data the RAM is used. RAM consists of the register banks, stack for temporary data storage. It also consists of some special function register (SFR) which are used for some specific purpose like timer, input output ports etc. Normally microcontroller has 256 byte RAM in which 128 byte is used for user space which is normally Register banks and stack. But other 128 byte RAM which consists of SFRs.

In 8051, 4KB read only memory (ROM) is available for program storage. This is used for permanent data storage. This is volatile memory; the data saved in this memory does not disappear after power failure.

Timers and Counters

Timer means which can give the delay of particular time between some events. For example on or off the lights after every 2 sec. This delay can be provided through some assembly program but in microcontroller two hardware pins are available for delay generation.

• In MC8051, two timer pins are available T0 and T1, by these timers we can give the delay of particular time if we use these in timer mode and can count external pulses at these pins if we use these pins in counter mode.

• 16 bits timers are available. Means we can generate delay between 0000H to FFFFH.

• Two special function registers are available.

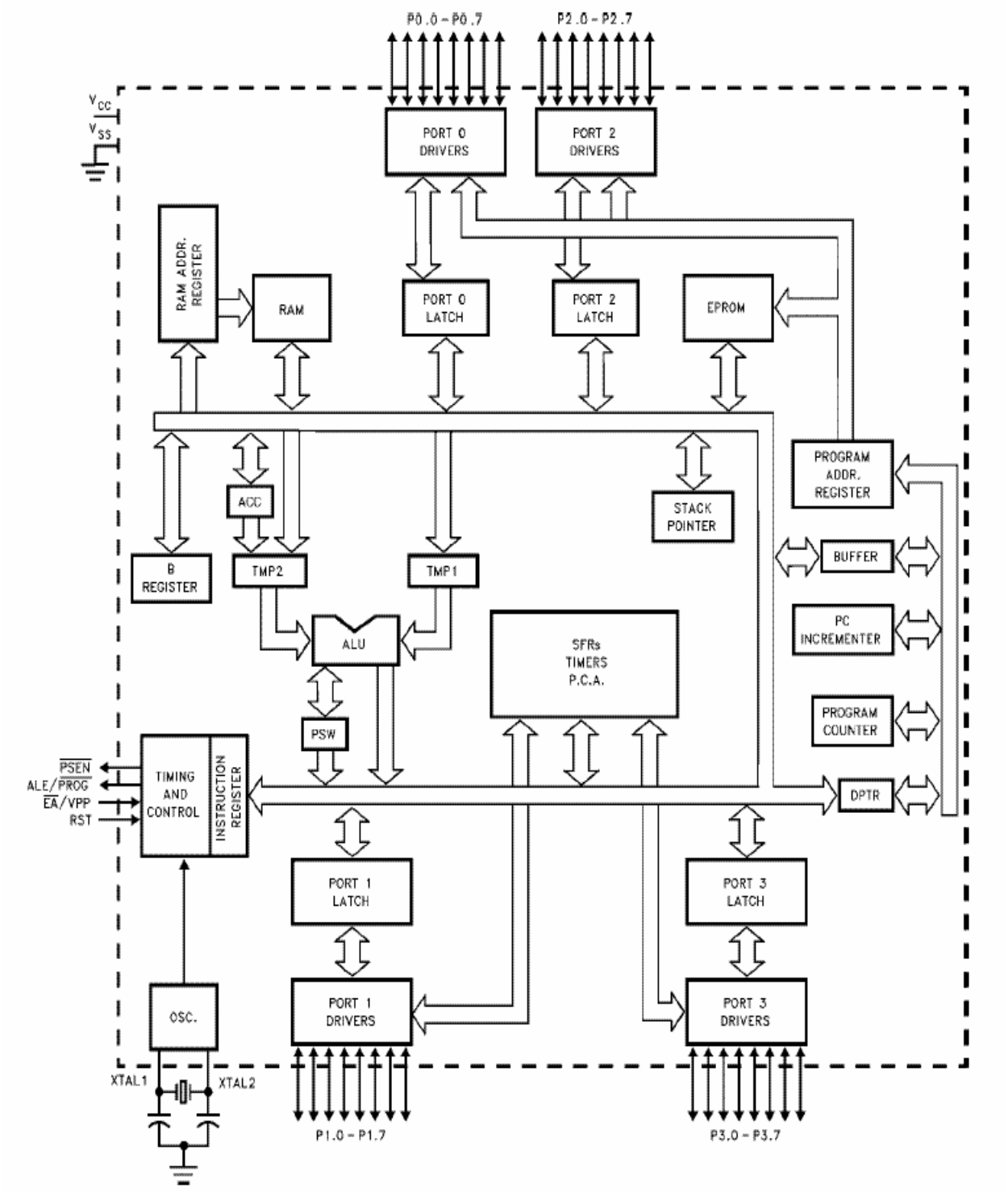
• TMOD, TCON registers are used for controlling timer operation.

Serial Port

• There are two pins available for serial communication TXD and RXD. Normally TXD is used for transmitting serial data which is in SBUF register, RXD is used for receiving the serial data. SCON register is used for controlling the operation.

Oscillator

• It is used for providing the clock to MC8051 which decides the speed or baud rate of MC. The use of crystal which frequency vary from 4MHz to 30 MHz. There are four input output ports available P0, P1, P2, P3.


P0.0 – P0.7
P2.0 – P2.7
VCC
VSS
PORT 0 DRIVERS
PORT 2 DRIVERS
RAM ADDR. REGISTER
RAM
PORT 0 LATCH
PORT 2 LATCH
EPROM
PROGRAM ADDR. REGISTER
ACC
STACK POINTER
BUFFER
B REGISTER
TMP2
TMP1
PC INCREMENTER
ALU
SFRs TIMERS P.C.A.
PSW
PROGRAM COUNTER
PSEN
ALE/PROG
EA/VPP
RST
TIMING AND CONTROL
INSTRUCTION REGISTER
DPTR
PORT 1 LATCH
PORT 3 LATCH
OSC.
PORT 1 DRIVERS
PORT 3 DRIVERS
XTAL1
XTAL2
P1.0 – P1.7
P3.0 – P3.7

Interrupts

• Interrupts are defined as requests because they can be refused (masked) if they are not used, that is when an interrupt is acknowledged. A special set of events or routines are followed to handle the interrupts. These special routines are known as interrupt handler or interrupt service routines (ISR). These are located at a special location in memory. INT0 and INT1 are the pins for external interrupts.

8051 Flag Bits and PSW Register

→ Used to indicate the Arithmetic condition of ACC.

→ Flag register in 8051 is called as program status word (PSW). This special function register PSW is also bit addressable and 8 bit wide means each bit can be set or reset independently.

There are four flags in 8051

• Parity flag • overflow flag • Auxiliary carry • carry flag

→ Four Register Banks

• RS1(PSW0.4) RS0(PSW0.3)

Register Bank Select

0 0 Bank 0

0 1 Bank 1

1 0 Bank 2

1 1 Bank 3

• F0 → user definable bit

Stack is last in first out (LIFO)

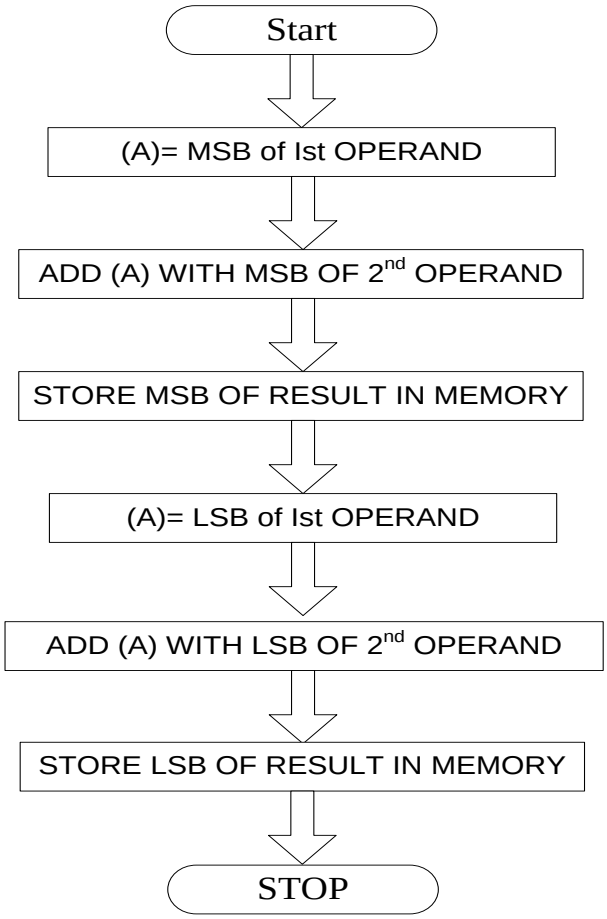
DPTR

It is a 16 bit register, it is divided into two parts DPH and DPL. DPH for Higher order 8 bits, DPL for lower order 8 bits. DPTR, DPH, DPL these all are SFRs in 8051.

PROCEDURE:

1. Enter Opcode and data in trainer.
2. Execute the program.
3. Change the data and see that correct results are obtained.

16 Bit Addition



A) ARITHMETIC OPERATIONS

PROGRAM 1: 16 Bit Addition

OBJECTIVE:

To perform 16 bit addition of two 16-bit data using immediate addressing mode and store the result in memory.

THEORY:

As there is only one 16-bit register in 8051. 16-bit addition is performed by using ADDC instruction twice, i.e. addition LSD first and MSD next. This program adds the 16bit data 1234 with 5678 and store the result in 4150 and 4151 using immediate addressing.

Program1:16 Bit Addition

ADDRESS	LABEL	OPCODE	MNEMONICS	COMMENTS
8100			CLR C	Clear cy flag
8101			MOV A,#DATA1	Move Data 1 to A reg
8103			ADD A, #DATA2	Add Data 2 with A Reg .
8105			MOVDPTR,#8150	Move 8150 to DPTR
8108			MOVX@DPTR, A	Move LSB Result from accumulator to DPTR
8109			INC DPTR	Increment DPTR
810A			MOV A,# DATA M1	Move MSB Data 1 to A reg
810C			ADDC,# DATA M2	Add MSB Data 2 with A Reg and CY flag.
810E			MOVX@DPTR,A	Move MSB Result from accumulator to DPTR
810F	HERE		SJMP HERE	Short Jump

DATA:

8102: 34, 8104:78, 810B:12, 810D: 56

RESULT:

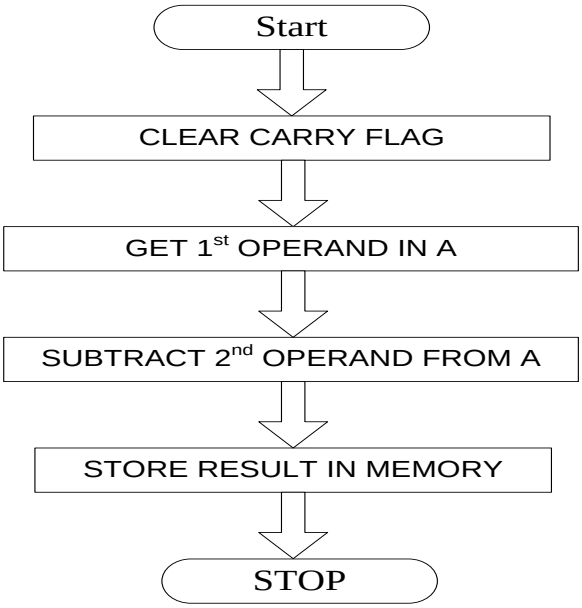
8150: AC; 8151:68

PROGRAM 2: 8 BIT SUBTRACTION

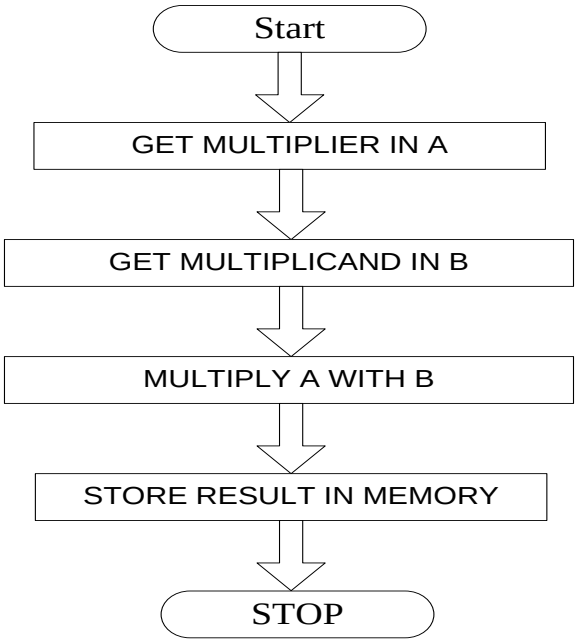
OBJECTIVE:

To perform subtraction of two 8 bit data using immediate addressing mode and store the result in memory.

8 BIT SUBTRACTION



8 BIT MULTIPLICATION



THEORY:

Using the accumulator, subtraction is performed and results are stored. Immediate addressing is employed. The SUBB instruction write the result in the ACC.

Program 2: 8 Bit Subtraction

ADDRESS	LABEL	OPCODE	MNEMONICS	COMMENTS
8100			CLR C	Clear cy flag
8101			MOV A,#DATA1	Move Data 1 to A reg
8103			SUBB A,#DATA2	SUB Data 2 with A Reg.
8105			MOV DPTR, #8500	Move 8500 to DPTR
8108			MOVX @DPTR, A	Move Result from accumulator to DPTR
8109	HERE		SJMP HERE	Short Jump

DATA:

8102: 20, 8104:10

RESULT:

8500: 10

PROGRAM 3: 8 BIT MULTIPLICATION

OBJECTIVE:

To obtain the product of two 8bit data using immediate addressing and store the result in memory.

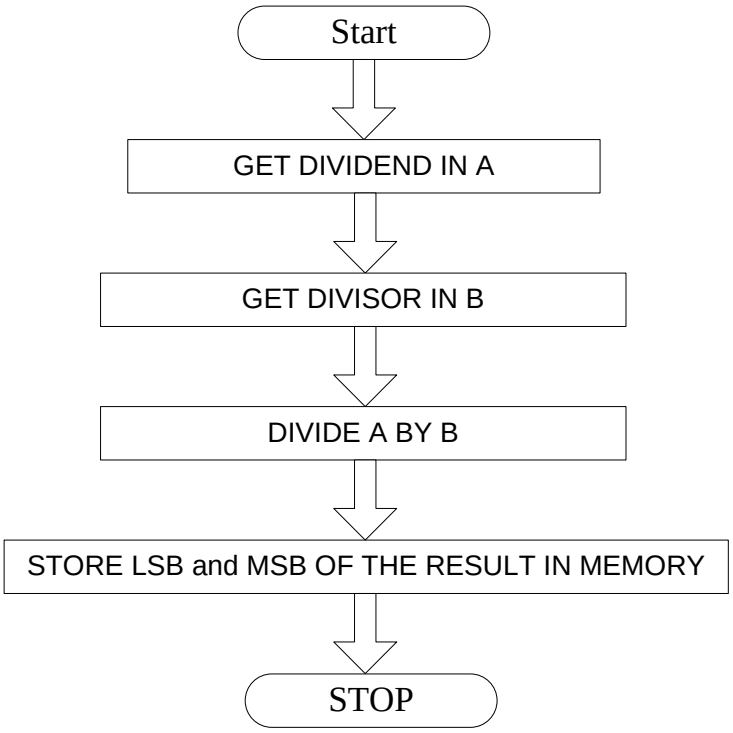
THEORY:

The 8051 microcontroller has MUL instruction unlike many other microprocessors. The MUL instruction multiplies the unsigned integers in A and B. The low order byte of the product is left in A and higher byte in B. If the product is greater than 255, the overflow flag is set, otherwise it is cleared. Carry flag is always cleared.

Program3: 8 Bit Multiplication

ADDRESS	LABEL	OPCODE	MNEMONICS	COMMENTS
8100			MOV A,#DATA1	Move Data 1 to A reg
8102			MOV B,#DATA2	Move Data 2 to B reg
8105			MUL AB	MUL B Reg. with A Reg.
8106			MOV DPTR, #8500	Move 8500 to DPTR

8 BIT DIVISION



8109			MOVX @DPTR,A	Move Result from accumulator to DPTR
810A			INC DPTR	Increment DPTR
810B			MOV A,B	Move B Reg. Result to A Reg.
810D			MOVX @DPTR, A	Move Result from accumulator to DPTR
810E	HERE		SJMP HERE	Short Jump

DATA:
8101: 0A, 8105:07

RESULT:
8500: 46; 8501:00

PROGRAM 4: 8 BIT DIVISION

OBJECTIVE:

To divide a 8 bit number by another 8bit number and to store the quotient and remainder in memory.

THEORY:

The 8051 microcontroller has DIV instruction unlike any other 8bit microprocessors DIV instruction divides the unsigned 8bit integer in A by that in B. The ACC receives the integer part of the quotient and B receives the integer remainder. The carry and overflow flags will be cleared.

Program 4: 8 Bit Division

ADDRESS	LABEL	OPCODE	MNEMONICS	COMMENTS
8100			MOV A,#DATA1	Move Data 1 to A reg
8102			MOV B,#DATA2	Move Data 2 to B reg
8105			DIV AB	DIV A Reg. by B Reg.
8106			MOV DPTR, #8500	Move 8500 to DPTR
8109			MOVX @DPTR,A	Move Result from accumulator to DPTR
810A			INC DPTR	Increment DPTR
810B			MOV A,B	Move B Reg. Result to A Reg.
810D			MOVX @DPTR, A	Move Result from accumulator to DPTR
810E	HERE		SJMP HERE	Short Jump

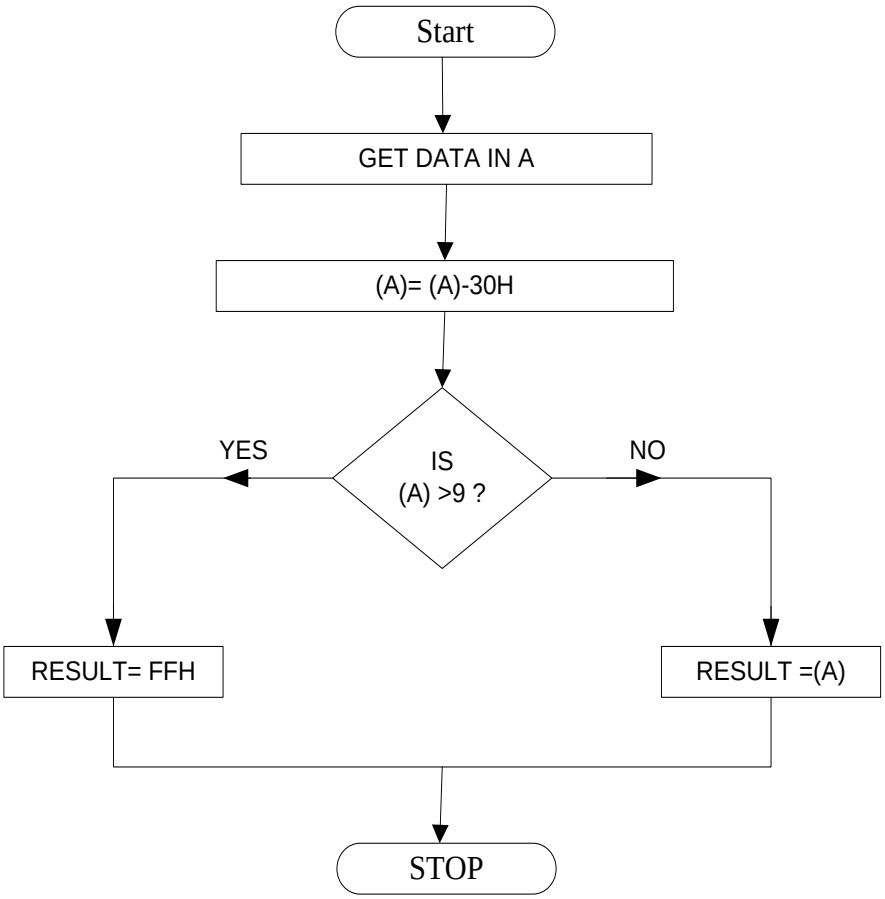
DATA:
8101: 65, 8104:0A

RESULT:
8500: 0C (Quotient); 8501:05(Remainder)

RESULT:

Thus the various arithmetic operations were performed and verified using 8051 Microcontroller.

ASCII TO DECIMAL CONVERSION



B) CODE CONVERSION USING 8051 MICROCONTROLLER

AIM:

To write and execute a few code conversion programs using 8051 microcontroller.

THEORY:

Code conversion is very essential in micro computing because many peripherals that are to be dealt which may provide data in ASCII, BCD or various special codes. Each one of the special codes which is different from the binary concept of 8051 must be converted first to binary, for the microprocessor to process it. After processing, the binary form must again be converted back to the special code of that peripheral, before transferring it to the peripheral.

PROGRAM 1: ASCII TO DECIMAL CONVERSION

OBJECTIVE: To convert the ASCII number in the accumulator to its equivalent decimal number.

THEORY: Conversion of an ASCII number t decimal is very simple because all the decimal numbers form a sequence in ASCII. Any ASCII number can be converted to decimal just by subtracting 30 from it.

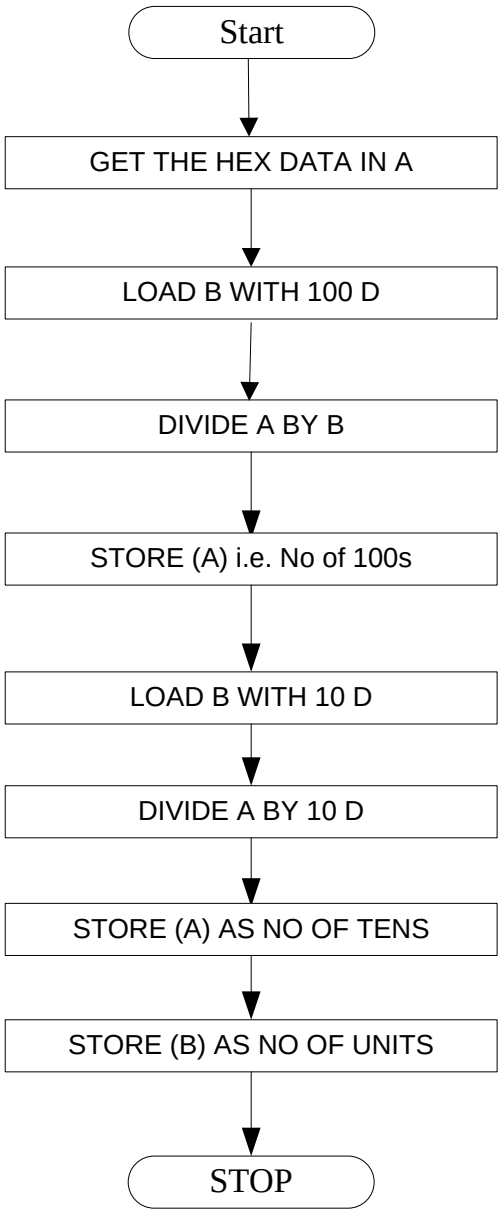
Program 1: ASCII TO DECIMAL CONVERSION

ADDRESS	LABEL	OPCODE	MNEMONICS	COMMENTS
8100			MOV DPTR, #8500	Move 8500 to DPTR
8103			MOV A, #DATA	Move data to A reg.
8105			CLR C	Clear CY flag
8106			SUBB A, #30	Sub 30 from A reg.
8108			CLR C	Clear CY flag
8109			SUBB A,#0A	Sub 0A from A reg. with borrow
810B			JC STR	Jump carry to STR
810D			MOV A,#0FF	Move FF to A reg.
810F			SJMP L1	SJMP to L1
8111	STR:		ADD A, #0A	Add 0A with A reg.
8113	L1:		MOVX @DPTR,A	Move Result from accumulator to DPTR
8114	HLT:		SJMP HLT	Short Jump

Data: 35
Result [8500]:

Data: 3B
Result [8500]:

HEX TO DECIMAL CONVERSION



PROGRAM 2: HEX TO DECIMAL CONVERSION

OBJECTIVE: To obtain the decimal equivalent of an 8 bit hex number stored in memory.

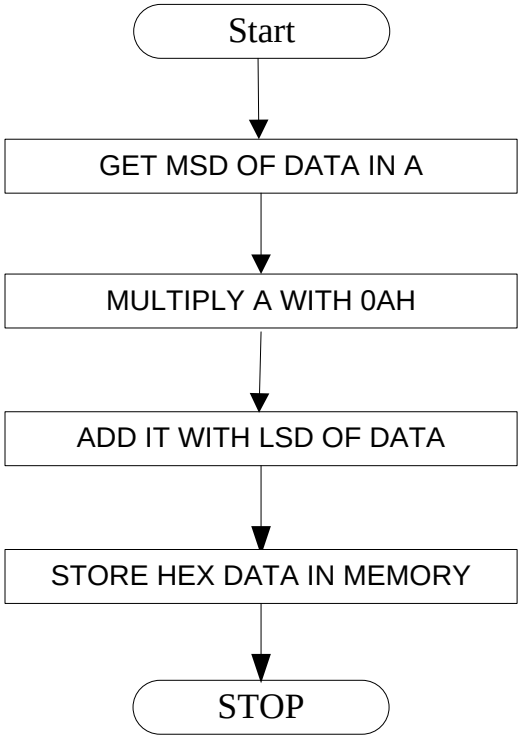
THEORY: The Hex number is converted to its equivalent decimal number. The Hex number to be converted is brought to the accumulator and is divided by 100 D to find the number of hundreds in it. DIV instruction of 8051 is used in this program. The remainder is divided by 10 D to count the number of tens in it. Finally the remainder obtained from the above division gives the number of units in the given Hex number. The result is stored in memory in the unpacked form.

Program 2: HEX TO DECIMAL CONVERSION

ADDRESS	LABEL	OPCODE	MNEMONICS	COMMENTS
8100			MOV DPTR, #8500	Move 8500 to DPTR
8103			MOVX A,@DPTR	Move data to A reg.
8104			MOV B,#64	Move 64 to B reg.
8107			DIV AB	Div A by B
8108			MOV DPTR, #8501	Move 8501 to DPTR
810B			MOVX @DPTR,A	Move data to A reg
810C			MOV AB	Move B reg. to A reg.
810E			MOV B, #0A	Move 0A to B reg.
8111			DIV AB	Div A by B
8112			INC DPTR	Increment DPTR
8113			MOVX @DPTR,A	Move data to A reg
8114			INC DPTR	Increment DPTR
8115			MOV A,B	Move B reg. to A reg
4118			MOVX @DPTR,A	Move Result from accumulator to DPTR
4119	HLT:		SJMP HLT	Short Jump

Data: [8500]: 0F
Result [8501]:
[8502]:

DECIMAL TO HEX CONVERSION



PROGRAM 3: DECIMAL TO HEX CONVERSION

OBJECTIVE: To convert BCD digits in memory to the equivalent hex number.

THEORY: Considering that out f the two unpacked BCD digits at 8500 and 8501, the digit at 8500 is the MSD, the logic is to multiply this by 0A (10D) and then add the LSD at 8501 to the product.

Program 3: DECIMALTO HEX CONVERSION

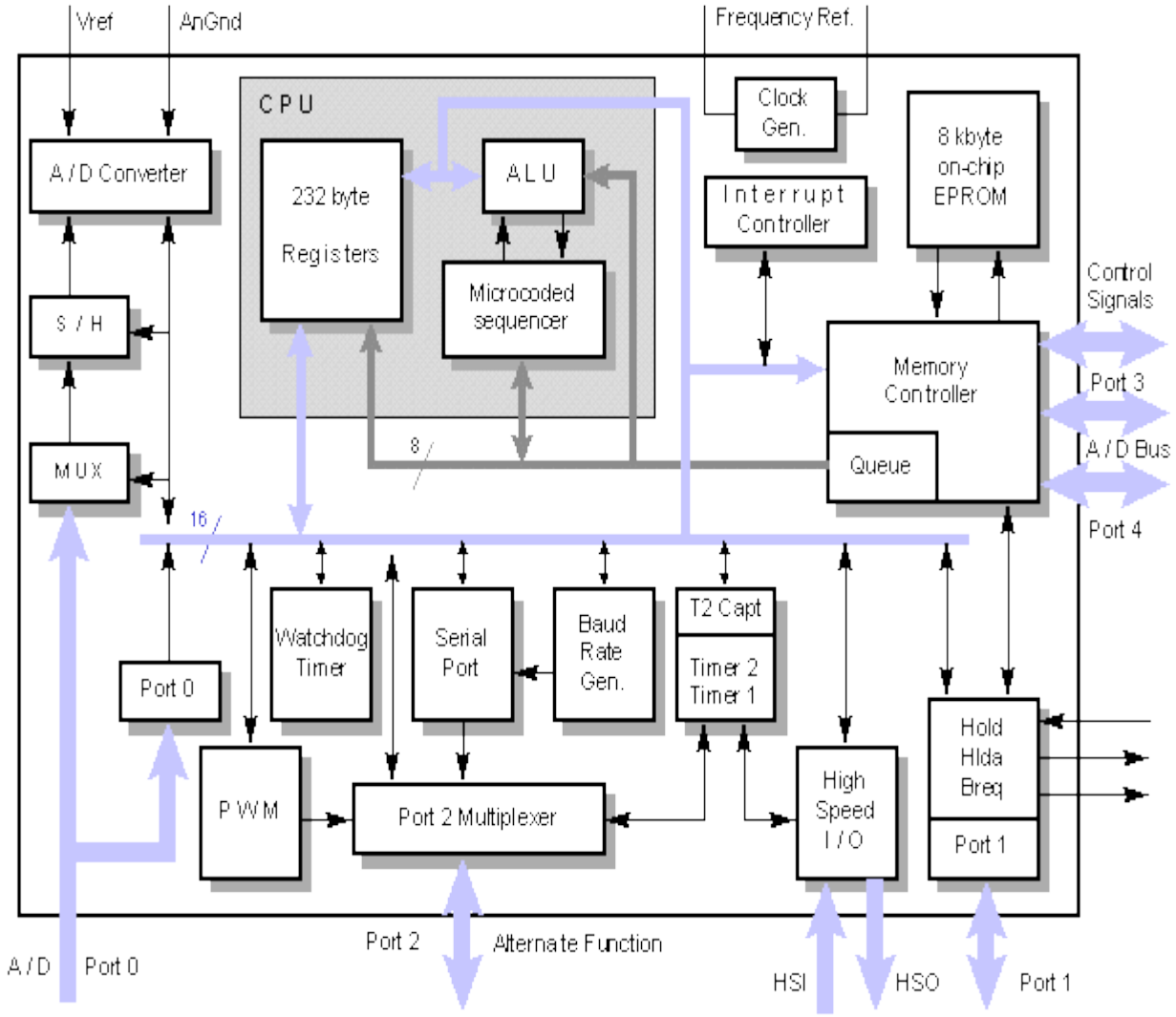
ADDRESS	LABEL	OPCODE	MNEMONICS	COMMENTS
8100			MOV DPTR, #8500	Move 8500 to DPTR
8103			MOVX A,@DPTR	Move data to A reg.
8104			MOV B,#0AH	Move 0A to B reg.
8107			MUL AB	MUL A with B reg.
8108			MOV B,A	Move A reg. B reg.
810A			INC DPTR	Increment DPTR
810B			MOVX A,@DPTR	Move DPTR content to A reg.
810C			ADD A,B	Add A reg. and B reg. content
810E			INC DPTR	Increment DPTR
810F			MOVX @DPTR,A	Move Result from accumulator to DPTR
8110	HLT:		SJMP HLT	Short Jump

Data [8500]:03
[8501]:06

Result [8502]:

RESULT:

Thus the above programs are loaded and the results are verified.



Architecture of 8097 Microcontroller

EX.NO: **DATE:**

STUDY OF 8097 MICROCONTROLLER

AIM

To perform various arithmetic and logical operations using 8097 microcontroller.

THEORY

Micro controllers are designed as a single chip, which typically includes a microprocessor, 64 bytes of Read/write memory, 8 Kbytes of ROM and several signal lines to connect I/O s. These are complete microcomputers on a chip also known as microcontroller. The features of 8097 are

1. A 8097 is a 16 bit microcontroller.
2. The 8097 is designed to use in applications which require high speed calculations and fast I/O operations.
3. The high speed I/O section of an 8097 includes a 16 bit timer, a 16 bit counter, a 4 input programmable edge detector, 4 software timers and a 6 output programmable event generator.
4. The 8097 has 8 multiplexed input analog to digital converter with 10 bit resolution. It can fully run under interrupt control.
5. Its programmable PWM output signals can be used as control signals to drive a motor and for any other applications.
6. Its several port has several modes of operation with programmable baud rates.
7. It supports register to register architecture which increases processing speed.
8. It has 100 instructions which can operate on bit, byte, word, double words.
9. It consists of a complete set of 16-bit arithmetic instructions including multiply and divide instructions.
10. Logical and arithmetic instructions are available for both byte and word operations.
11. The bit operations are possible and these can be performed on any bit in the register file or in the special function register.

The major components of the CPU on the 8097 are register file and the register ALU(RALU). The CPU register file has 256 bytes of memory. Out of 256 locations the first 24 memory locations 00H to 17H are the special function registers (SFRs). The SFRs are used to control the on chip I/O section. The upper 16 bytes of RAM is called power down RAM because these locations receive their power from the V_{PD} pin in the power down mode. RALU contains a 17 bit ALU, the program status word (PSW), the program counter (PC), a loop counter and three temporary registers.

The 8097 has two 16 bit timers. Timer 1 is used to synchronize events to real time, while timer 2 can be clocked externally and synchronizes events to external occurrences.

The high speed input unit (HIS), can be used to record the time at which an event occurs with respect to timer 1. The high speed output unit (HSO) is used to trigger events at specific times with minimal CPU overhead. These events include: starting an A to D conversion, resetting timer 2 setting 4 software flags and switching up to 6 output lines. The HSO can be

programmed to generate interrupts at preset timers. The A/D converter on the 8097 provides 8-input channels with a 10-bit digital output. The channels are multiplexed. Successive approximation technique is used for conversion.

There are five 8 bit I/O ports on 8097. Port 0 is an input only port which shares its pins with the analog inputs to the A/D converter. Port 1 is a quasi-bidirectional I/O port. Port 2 is a multifunctional port. Port 3 and port 4 pins have two functions. They are either bidirectional ports with open-drain outputs or system bus pins which the memory controller uses when it is accessing external memory.

The watchdog timer provides as a means of graceful recovery from a software upset. Once the watchdog is initialized if the software fails to reset the watchdog at least every 64K state times, a hardware reset will be initialized and the system will restart.

The addressable memory space on the 8097 consists of 64Kbytes. Locations 1FFEH and 1FFFH are reserved for ports 3 and 4 respectively. The 9 interrupt vectors are stored in locations 2000H through 2011H. 2012H through 207FH are reserved for Intel's factory test code. Resetting the 8097 causes instructions to be fetched starting from location 2080H. This location was chosen to allow a system to have up to 8K of RAM continuous with the register file. External memory is addressed through lines AD0- AD15 which forms a 16-bit multiplexed data bus. These lines share pins with I/O ports 3 and 4.

Eight interrupt sources are available on the 8097. Software trap, Extint, serial port, software timers, HSL0, high speed outputs, HIS data available, A/D conversion complete and timer overflow.

Program 1

Addition of Two 16-Bit Numbers

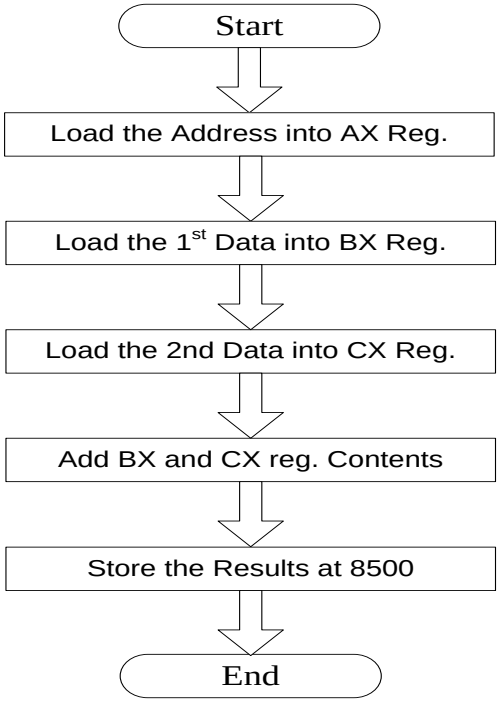
Objective: To add two16 bit hex numbers and store the result at 8500.

Data Reg 1: 0003-------0000 0000 0000 0011

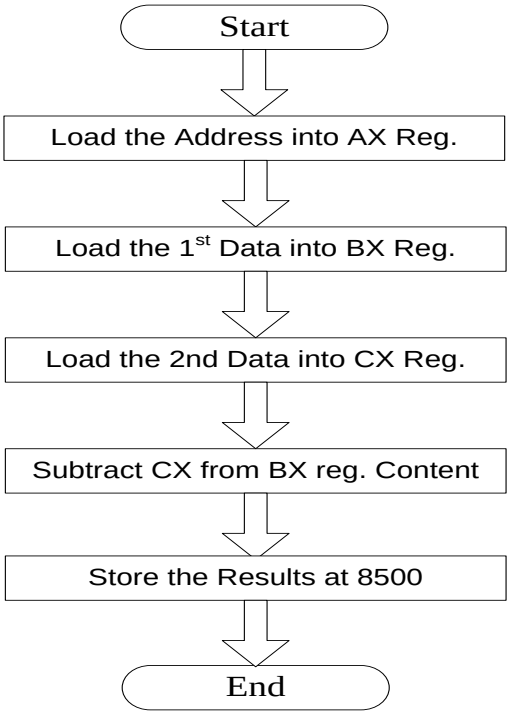
Reg 2: 0007-------0000 0000 0000 0111

Result (8500): 0000 0000 0000 1010 (000A)

Addition of Two 16-Bit Numbers



Subtraction of Two 16-Bit Numbers



Address	Opcodes	Mnemonics	Comments
8100		LD AX #8500H	Load AX with address 8500
8104		LD BX #0003H	Load BX with data 0003
8108		LD CX #0007H	Load CX with data 0007
810C		ADD BX,CX	ADD BX and CX contents and store the result in BX
810F		ST BX, [AX]+	Store the result from BX to AX+
8112		SJMP HERE	Short Jump

Program 2

Subtraction of Two 16-Bit Numbers

Objective: To subtract two16 bit hex numbers and store the result at 8500.

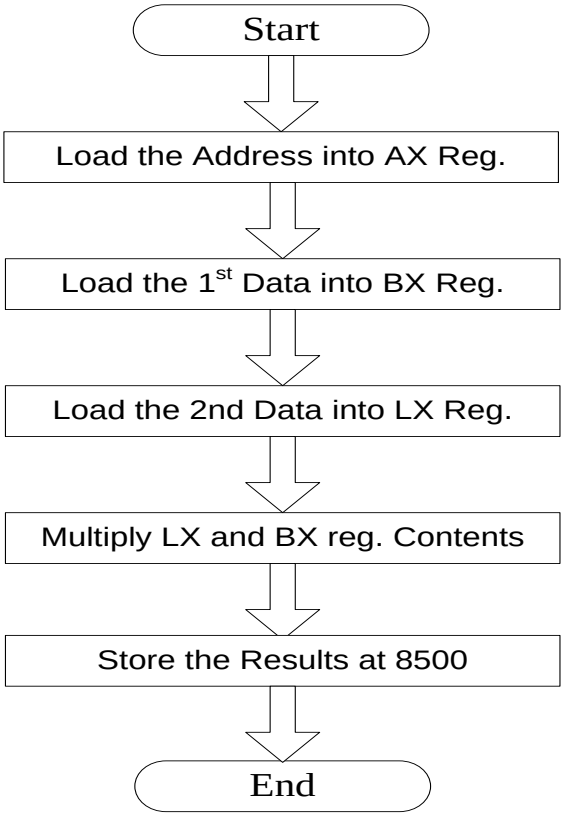
Data Reg 1: 0007-------0000 0000 0000 0111

Reg 1: 0003-------0000 0000 0000 0011

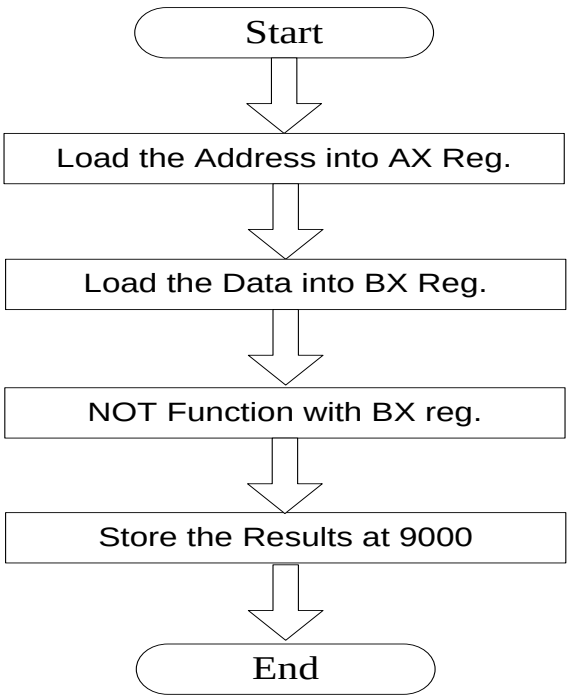
Result 8500 0000 0000 0000 0100 (0004)

Address	Opcodes	Mnemonics	Comments
8100		LD AX #8500H	Load AX with address 8500
8104		LD BX #0007H	Load BX with data 0007
8108		LD CX #0003H	Load CX with data 0003
810C		SUB BX,CX	SUB BX and CX contents and store the result in BX
810F		ST BX, [AX]+	Store the result from BX to AX+
8112		SJMP HERE	Short Jump

Multiplication of Two 16-Bit Numbers



One's Complement of a 16-Bit Number



Program 3

Multiplication of Two 16-Bit Numbers

Objective: To multiply two16 bit hex numbers and store the result at 8500.

Data Reg 1: 0007-------0000 0000 0000 0111

Reg 1: 0003-------0000 0000 0000 0011

Result (8500): 0000 0000 0001 0101 (0015)

Address	**Opcodes**	**Mnemonics**	**Comments**
8100		LD AX #8500H	Load AX with address 8500
8104		LD BX #0007H	Load BX with data 0007
8108		LD CX #0003H	Load CX with data 0003
810C		MUL CX,BX	MUL CX and BX contents and store the result in CX
810F		ST CL, [AX]+	Store the result from CX to AX+
8112		ST CH, [AX]+	
8115		SJMP HERE	Short Jump

Program 4

One's Complement of a 16-Bit Number

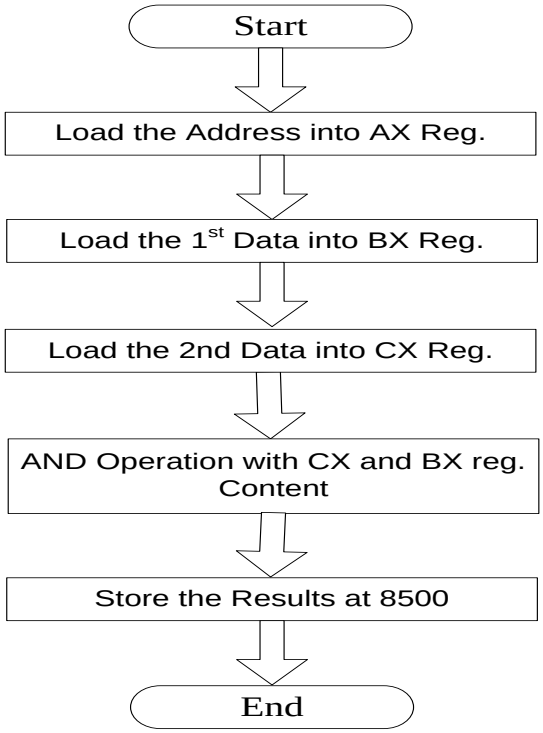
Objective: To find the one's Complement of the 16 bit data in on chip register and store the result at 9000.

Data Reg BX: 1234-------0001 0010 0011 0100

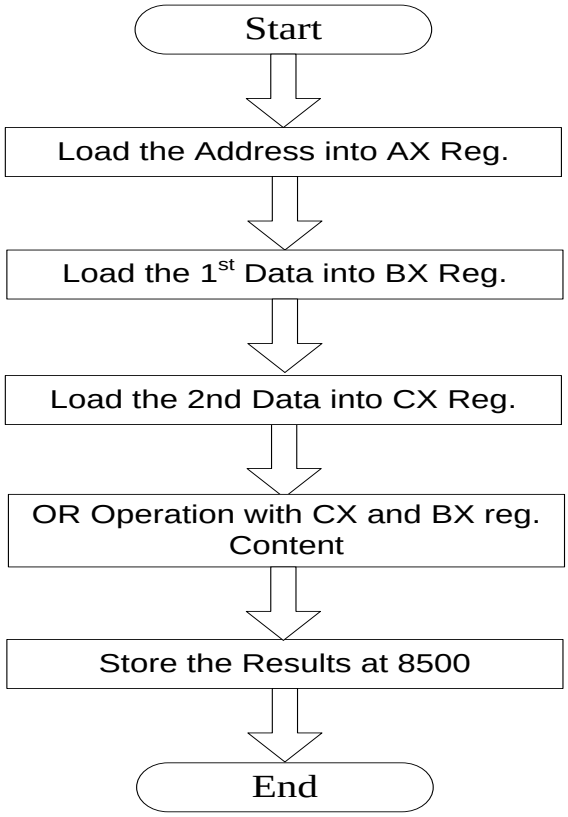
Result (9000): 1110 1101 1100 1011 (EDCB)

Address	**Opcodes**	**Mnemonics**	**Comments**
8100		LD AX #9000H	Load AX with address 9000
8104		LD BX #1234H	Load BX with data 1234
8108		NOT BX	NOT the BX content
810A		ST BX [AX]	Store the result from BX to AX
810D		SJMP 820D	Short Jump

AND Operation of Two 16-Bit Numbers



OR Operation of Two 16-Bit Numbers



Program 5

AND Operation of Two 16-Bit Numbers

Objective: To discuss the usage of AND instruction

Data Reg 1: 0003-------0000 0000 0000 0011

Reg 2: 0007-------0000 0000 0000 0111

Result (8500): 0000 0000 0000 0011 (0003)

Address	Opcodes	Mnemonics	Comments
8100		LD AX #8500H	Load AX with address 8500
8104		LD BX #0003H	Load BX with data 0003
8108		LD CX #0007H	Load CX with data 0007
810C		AND DX,CX,BX	AND BX and CX contents and store the result in DX
8110		ST DX, [AX]+	Store the result from DX to AX+
8113		SJMP HERE	Short Jump

Program 6

OR Operation of Two 16-Bit Numbers

Objective: To discuss the usage of OR instruction

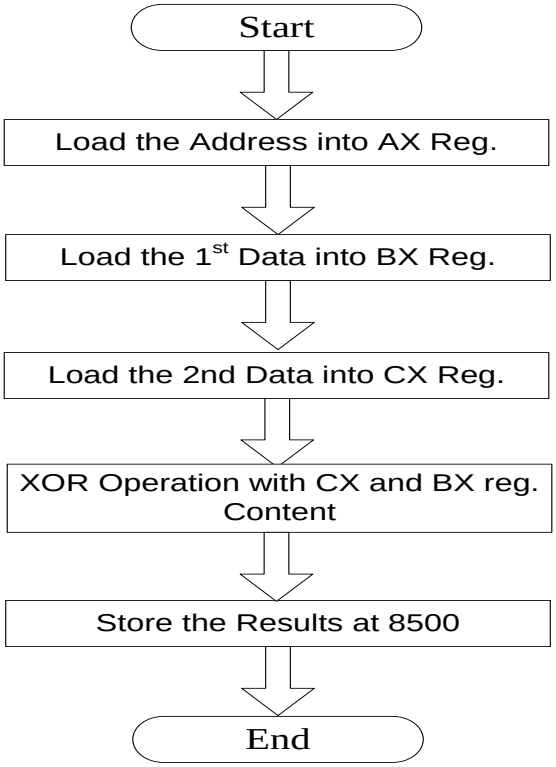
Data Reg 1: 0003-------0000 0000 0000 0011

Reg 2: 0007-------0000 0000 0000 0111

Result (8500): 0000 0000 0000 0111 (0007)

Address	Opcodes	Mnemonics	Comments
8100		LD AX #8500H	Load AX with address 8500
8104		LD BX #0003H	Load BX with data 0003
8108		LDCX #0007H	Load CX with data 0007
810C		OR CX,BX	OR BX and CX contents and store the result in CX
8110		ST CX, [AX]+	Store the result from CX to AX+
8113		SJMP HERE	Short Jump

XOR Operation of Two 16-Bit Numbers



Program 7

XOR Operation of Two 16-Bit Numbers

Objective: To discuss the usage of XOR instruction

Data Reg 1: 0003-------0000 0000 0000 0011

Reg 2: 0007-------0000 0000 0000 0111

Result (8500): 0000 0000 0000 0100 (0004)

Address	**Opcodes**	**Mnemonics**	**Comments**
8100		LD AX #8500H	Load AX with address 8500
8104		LD BX#0003H	Load BX with data 0003
8108		LD CX #0007H	Load CX with data 0007
810C		XOR CX,BX	XOR BX and CX contents and store the result in CX
8110		ST CX, [AX]+	Store the result from CX to AX+
8113		SJMP HERE	Short Jump

RESULT:

Thus the various arithmetic and logical operations were performed and verified using 8097 microcontroller.

EX.NO: **DATE:**

STUDY OF PROGRAMMABLE PERIPHERAL INTERFACE-8255

AIM

To study the Input/ Output and BSR mode operations of 8255 interfaced with 8051 microcontroller.

THEORY

PPI is abbreviation for Programmable Peripheral Interface. It is an I/O port chip used for interfacing I/O devices with microprocessor. It is a very commonly used peripheral chip. It contains three 8 bit ports; Port A, Port B & Port C. Port C is composed of two independent 4-bit ports : PC7-4 (PC Upper) and PC3-0 (PC Lower). The three ports of 8255 are divided into two groups, Group A and Group B. Group A contains Port A (PA0 – PA7) and Port C higher order lines (PC4 – PC7). Group B contains Port B (PB0 – PB7) and Port C lower order lines (PC0 – PC3). Group A can be configured in three modes: mode-0, mode-1 & mode-2. Whereas Group B can configured in two modes: mode- 0 and mode-1.

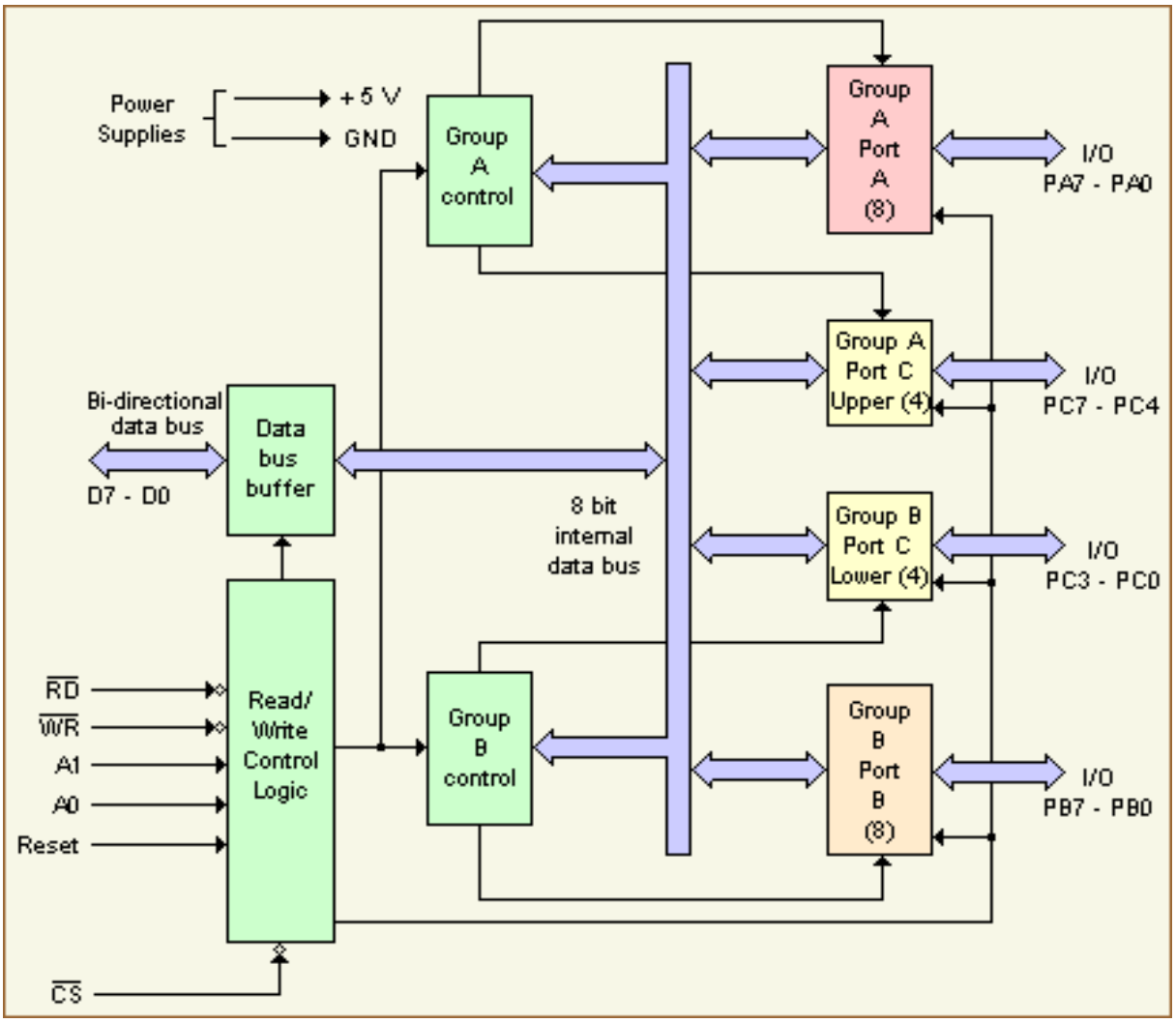
The ports are configured as input or output by the command word. The command word contains information such as mode, bit set, etc. i.e. the command word decides whether a Port is input Port or output Port and the mode of data transfer through a port. The command word is written into the control register of 8255. The control word of 8255 can only be written into, no read operation is allowed.

There are 3 modes of operation for the ports of 8255. Mode 0, Mode 1, and Mode 2.

Read/Write control logic

The function of this block is to manage all the internal and external transfer of both data and control or status word. The control pins with which the CPU communicates to 8255 are the RESET, CS, RD, WR, A0, A1, D0 – D7.Its basic operations are as given in the following table.

RD	-	Active low read
WR	-	Active low write
CS	-	Active low chip select
A0, A1	-	Port address select
D0 – D7	-	Bidirectional data bus



Functional Block Diagram of 8255

Port address

The addresses for the four registers in the 8255 are:

Register	Address
Control register	C6
Port A	C0
Port B	C2
Port C	C4

Programming the 8255

There are 2 control words in 8255

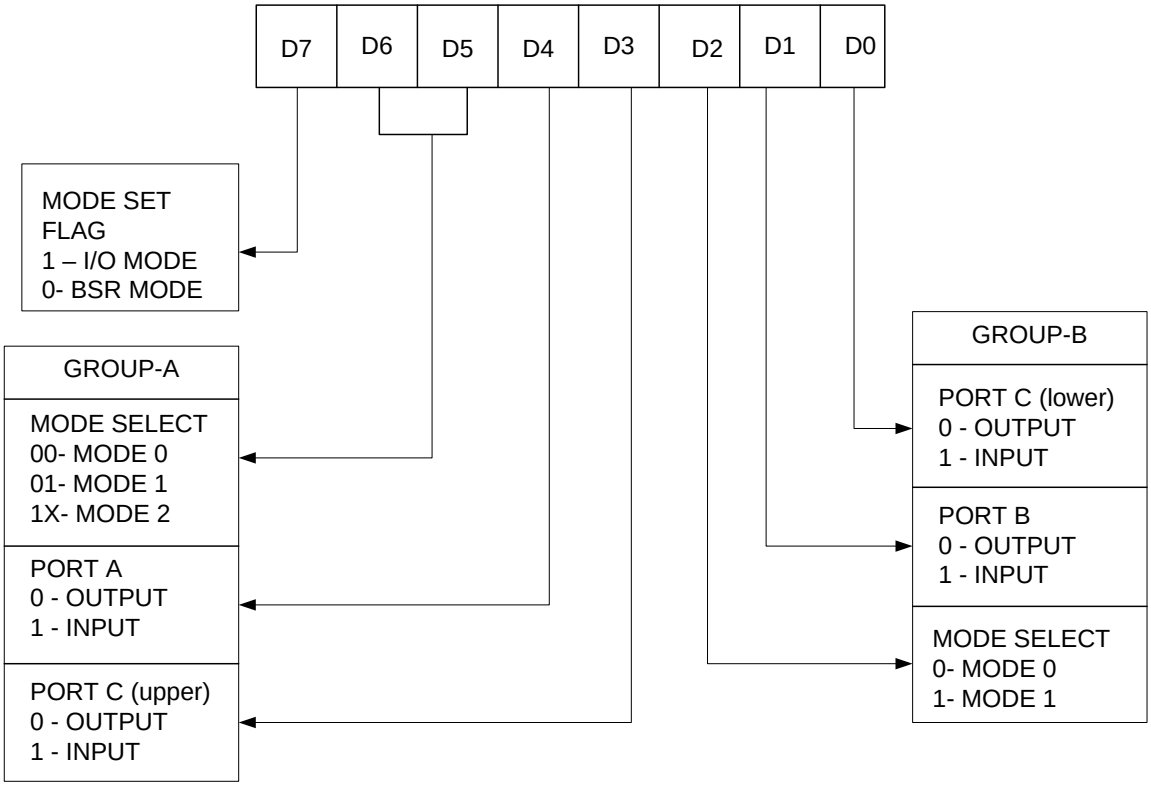
1. Mode Definition (MD) Control word and
2. Bit Set / Reset (BSR) Control Word

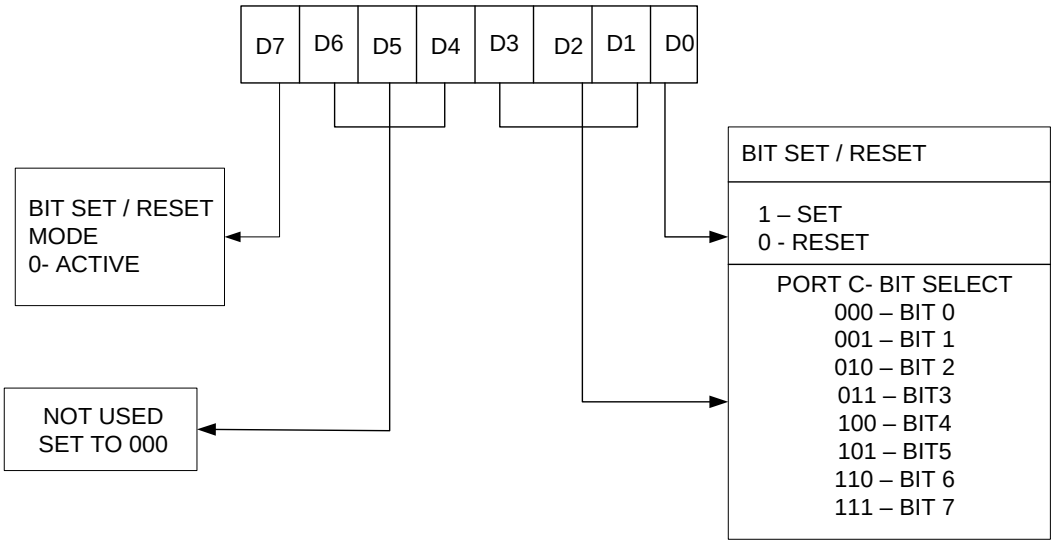
Program 1

To initialize port A as input port in mode0 & to input the data, set by the SPDT switches through port A

ADDRESS	LABEL	OPCODE	MNEMONICS	COMMENTS
4100			MOV DPTR, # FFC6 H	FFC6-Address of the control register
4103			MOV A, # 90 H	Move the control word to the accumulator
4105			MOVX @ DPTR, A	Output the control word to the control register
4106			MOV DPTR, # FFC0 H	FFC0- Address of port A. The data set by the SPDT switch is moved to accumulator
4109			MOVX A, @ DPTR	
410A			MOV DPTR, # 4500 H	The accumulator value is stored to the memory location 4500
410D			MOVX @ DPTR, A	
410E	HERE		SJMP HERE	

Mode Definition (MD) Control word





program 2

- To initialize port A as input port & port B as an output port in mode 0.
- To input the data, set by the SPDT switches through port A & to output the same data by the LEDs through port B.

ADDRESS	LABEL	OPCODE	MNEMONICS	COMMENTS
4100			MOV DPTR, # FFC6 H	FFC6-Address of the control register
4103			MOV A, # 90 H	Move the control word to the accumulator
4105			MOVX @ DPTR, A	Output the control word to the control register
4106			MOV DPTR, # FFC0 H	FFC0- Address of port A. The data set by the SPDT switch is moved to accumulator
4109			MOVX A, @ DPTR	
410A			MOV DPTR, # FFC2 H	FFC2- Address of port B. The accumulator value is send through port B to glow the LEDs.
410D			MOVX @ DPTR, A	
410E	HERE		SJMP HERE	

Program 3

- To initialize port C as an output port in mode 0.
- To output the data through port C.

ADDRESS	LABEL	OPCODE	MNEMONICS	COMMENTS
4100			MOV DPTR, # FFC6 H	FFC6-Address of the control register
4103			MOV A, # 90 H	Move the control word to the accumulator
4105			MOVX @ DPTR, A	Output the control word to the control register
4106			MOV DPTR, # FFC4 H	FFC4-Address of the port C
4109			MOV A, # 80	Move the accumulator with data
410B			MOVX @ DPTR, A	To output the data through port C
410C	HERE		SJMP HERE	

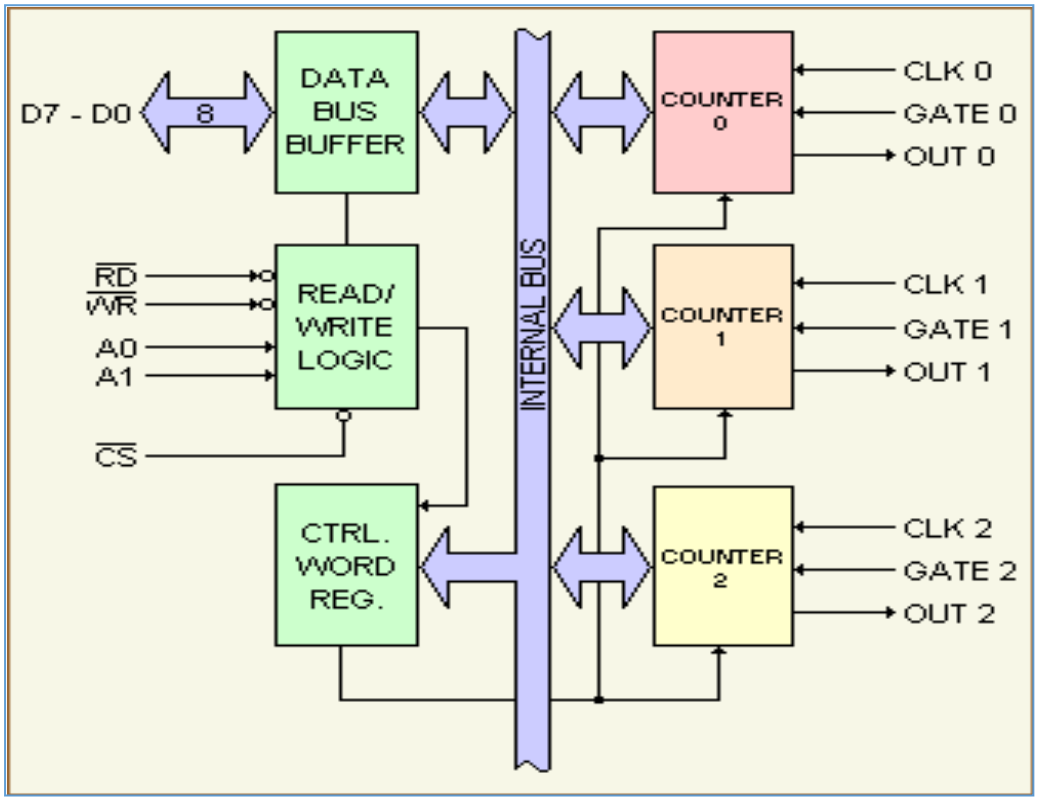
Program 4

- To initialize port C as an input port in mode 0.

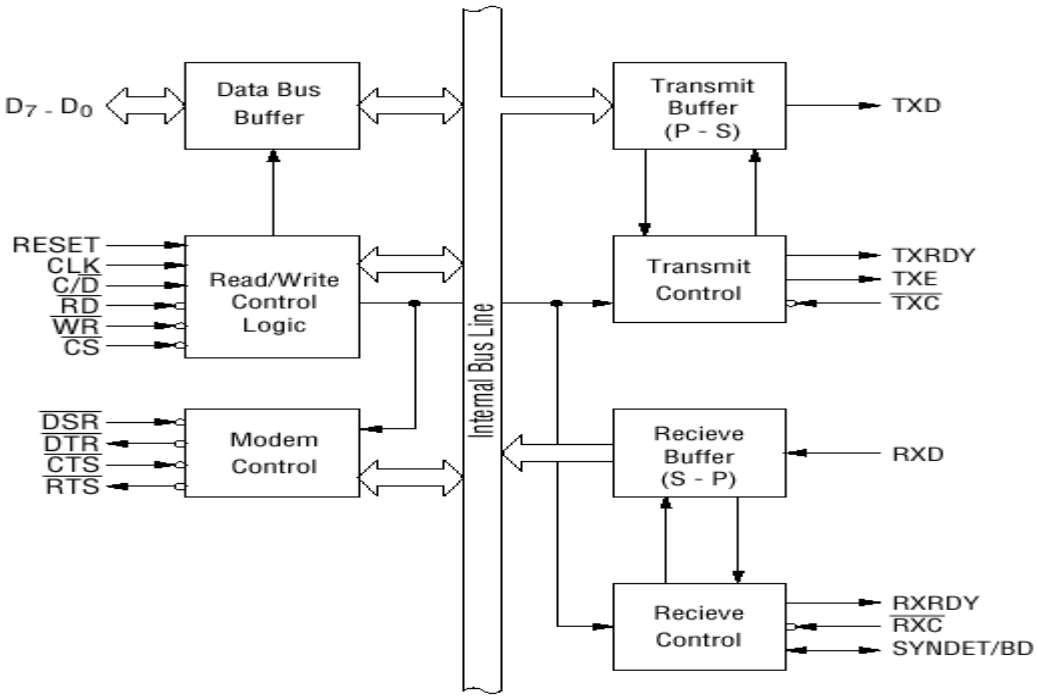
ADDRESS	LABEL	OPCODE	MNEMONICS	COMMENTS
4100			MOV DPTR, # FFC6 H	FFC6-Address of the control register
4103			MOV A, # 99 H	Move the control word to the accumulator
4105			MOVX @ DPTR, A	Output the control word to the control register
4106			MOV DPTR, # FFC4 H	FFC4-Address of the port C
4109			MOV A, @ DPTR	Move the data from port C to accumulator
410A			MOV DPTR, # 4500 H	Move the accumulator content to the memory location 4500
410D			MOVX @ DPTR, A	
410E	HERE		SJMP HERE	

Result

The operation of 8255 in Input / Output mode has been studied and the results were verified.



Block Diagram of 8253-Programmable Interval Timer



Block Diagram of 8251-Programmable Communication Interface

EX.NO: DATE:

SINGLE CHARACTER TRANSMISSION AND RECEPTION USING 8051 MICROCONTROLLER

AIM:

To initiate 8253 and 8251 and to check the transmission and reception of a character.

The Programmable Interval Timer/Counter – 8253:

The 8253 programmable Interval timer consists of three independent 16-bit programmable counters (timers). Each counter is capable of counting in binary or binary coded decimal. The maximum allowable frequency to any counter is 10MHz. This device is useful whenever the microprocessor must control real-time events. The timer in a personal computer is an 8253. To operate a counter a 16-bit count is loaded in its register and on command, it begins to decrement the count until it reaches 0. At the end of the count it generates a pulse, which interrupts the processor. The count can count either in binary or BCD.

Data bus buffer- It is a communication path between the timer and the microprocessor. The buffer is 8-bit and bidirectional. It is connected to the data bus of the microprocessor.

Read / write logic controls the reading and the writing of the counter registers.

Control word register, specifies the counter to be used and either a Read or a write operation.

A1	A0	Selection
0	0	Counter 0
0	1	Counter 1
1	0	Counter 2
1	1	Control Register

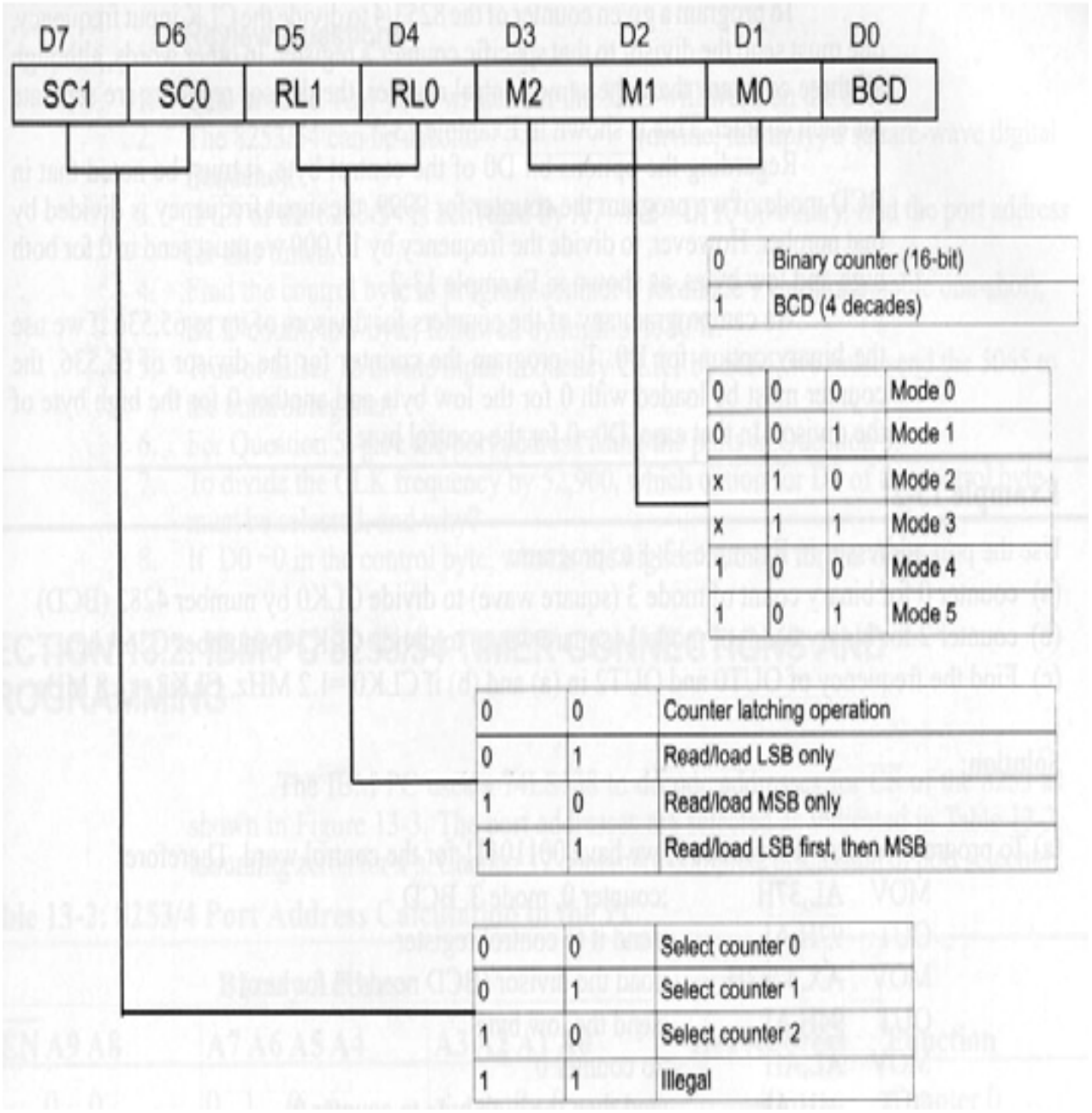
Each counter in the block diagram has 3 logical lines connected to it. Two of these lines, clock and gate, are inputs. The third, labeled OUT is an output.
Clock - clock input for the counter. The counter is 16 bits. It provides the basic operating frequency to the timer. The maximum clock frequency is 1 / 380 nanoseconds or 2.6megahertz.
Out – this is the output of the timer.
Gate – controls the timer. It's used either to enable or disable the counter.

The 8253 can operate independently in any one of the 6 modes. They are

Mode 0 Interrupt on terminal count

Mode 1 Programmable one shot

Mode 2 Rate Generator



8253-Control Word Format

Mode 3 Square wave rate Generator

Mode 4 Software triggered strobe

Mode 5 Hardware triggered strobe

Programmable Communication Interface-8251 (USART)

The 8251 is used as a peripheral device for serial communication and is programmed by the CPU to operate using virtually any serial data transmission technique. The USART accepts data characters from the CPU in parallel format and then converts them into a continuous serial data stream for transmission. Simultaneously, it can receive serial data streams and convert them into parallel data characters for the CPU. The CPU can read the status of USART at any time. These include data transmission errors and control signals. Prior to starting data transmission or reception, the 8251 must be loaded with a set of control words generated by the CPU. These control signals define the complete functional definition of the 8251 and must immediately follow a RESET operation. Control words should be written into the control register of 8251.

Command Instruction Format:

This format defines a status word that is used to control the actual operation of 8251. All control words written into 8251 after the mode instruction will load the command instruction. The command instructions can be written into 8251 at any time in the data block during the operation of the 8251. To return to the mode instruction format, the master reset bit in the command instruction word can be set to initiate an internal reset operation which automatically places the 8251 back into the mode instruction format. Command instructions must follow the mode instructions or sync characters.

There are two types of control word.

1. Mode instruction (setting of function)

2. Command (setting of operation)

1) Mode Instruction

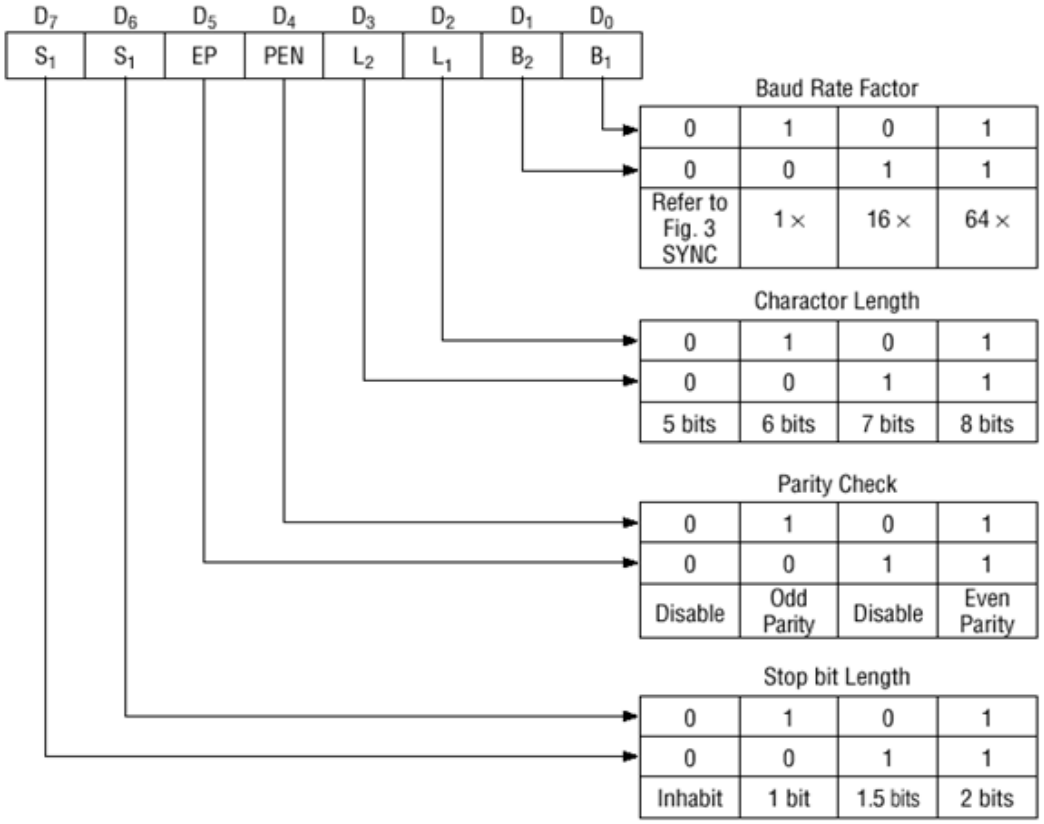
Mode instruction is used for setting the function of the 8251. Mode instruction will be in "wait for write" at either internal reset or external reset. That is, the writing of a control word after resetting will be recognized as a "mode instruction."

Items set by mode instruction are as follows:

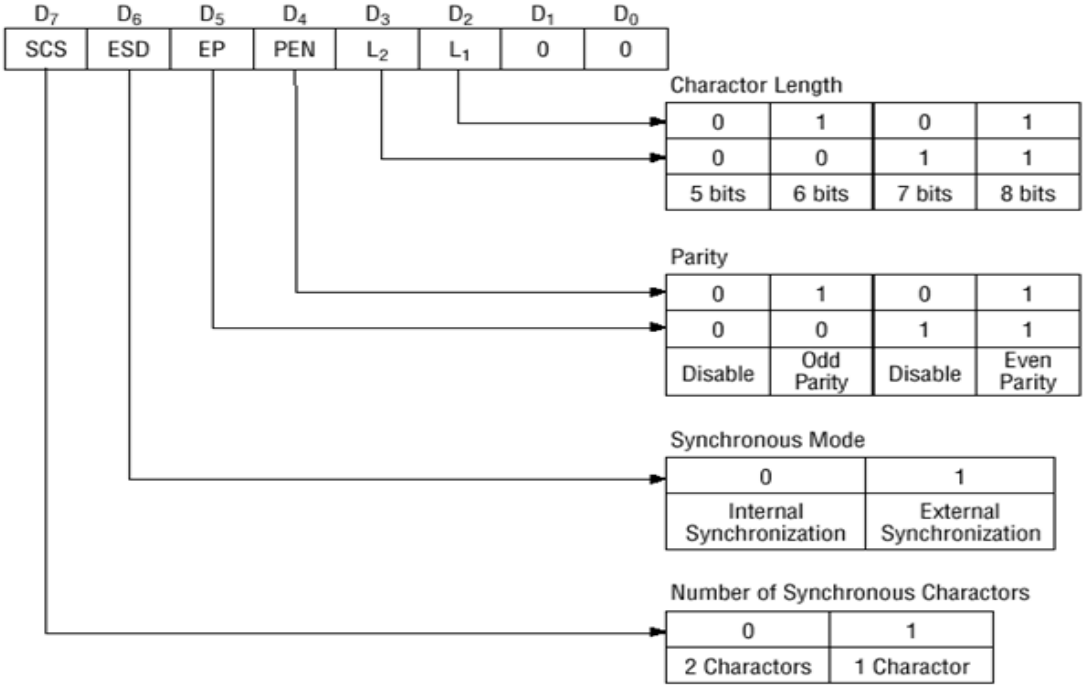
• Synchronous/asynchronous mode

• Stop bit length (asynchronous mode)

• Character length



8251-Asynchronous Mode Instruction Format



8251-synchronous Mode Instruction Format

• Parity bit

• Baud rate factor (asynchronous mode)

• Internal/external synchronization (synchronous mode)

• Number of synchronous characters (Synchronous mode)

The bit configuration of mode instruction is shown in Figures 4 and 5. In the case of synchronous mode, it is necessary to write one-or two byte sync characters. If sync characters were written, a function will be set because the writing of sync characters constitutes part of mode instruction.

2) Command

Command is used for setting the operation of the 8251. It is possible to write a command whenever necessary after writing a mode instruction and sync characters. Items to be set by command are as follows:

• Transmit Enable/Disable

• Receive Enable/Disable

• DTR, RTS Output of data.

• Resetting of error flag.

• Sending to break characters

• Internal resetting

• Hunt mode (synchronous mode)

3) Status Word

It is possible to see the internal status of the 8251 by reading a status word. The bit configuration of status word is shown in Fig. 7.

The following table gives the addresses of the peripheral.

8253 Control Register-------FFCE	8253 Channel 0 -------FFC8
8253 Channel 1 -------FFCA	8253 Channel 2 -------FFCC
8251 Control Register-------FFC2	8251 Data Register -------FFC0

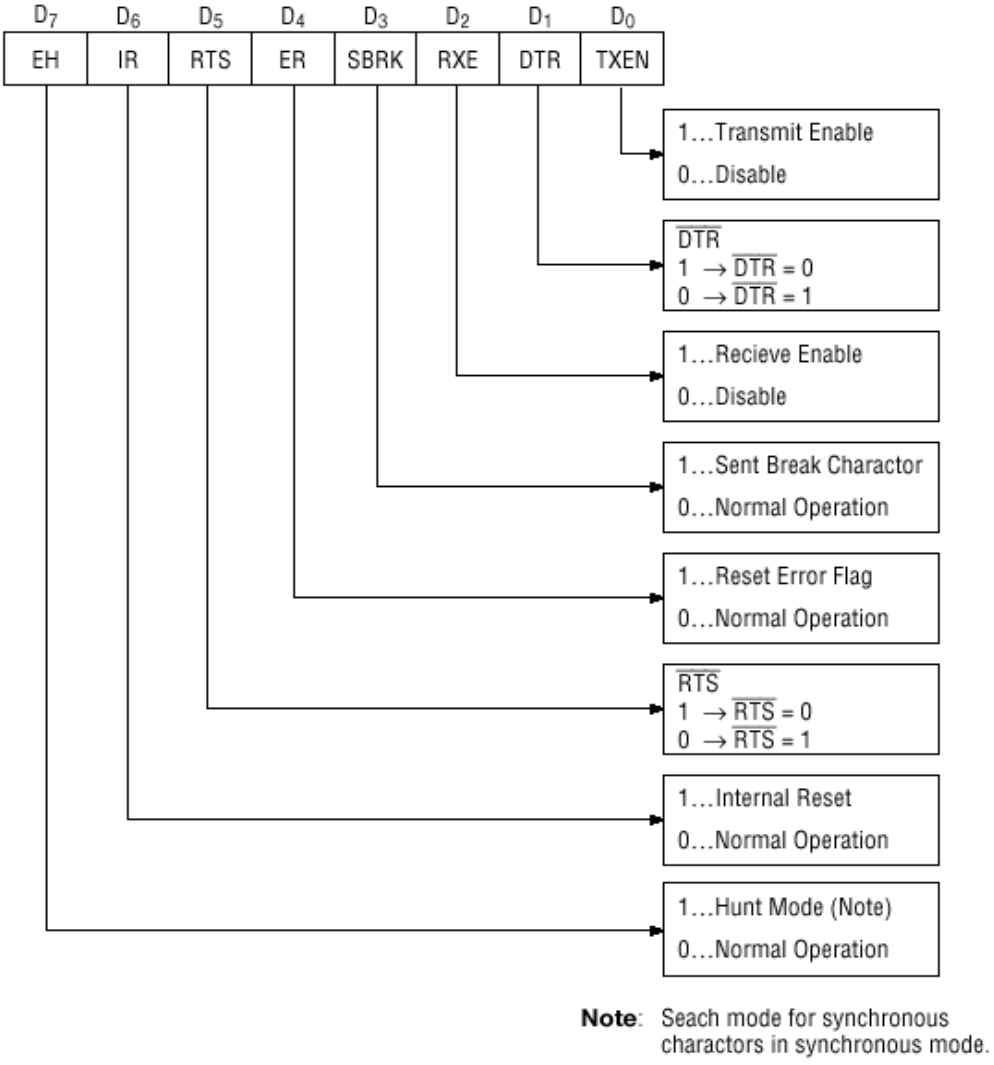


Fig. 6: Bit Configuration of Command

Program-To Check Transmission and Reception of a Character

ADDRESS	LABEL	OPCODE	MNEMONICS	COMMENTS
8100			MOV A,#36	Channel 0 in mode 3
8102			MOV DPTR, #FFCE	8253-Control Reg. address
8105			MOVX @DPTR,A	
8106			MOV A,#0A	to get an output frequency of 150KHZ at ch-0, so as to get a baud rate of 150KHZ at ch-0.
8108			MOV DPTR,#FFC8	8253-Control 0 address
810B			MOVX@DPTR,A	
810C			MOV A,#00	
810E			MOVX @DPTR,A	
810F			MOV A,#4E	8251-Mode Command Word
8111			MOV DPTR,#FFC2	8251-Control Reg. address
8114			MOVX@DPTR,A	
8115			MOV A,#37	8251-Control Word
8117			MOVX @DPTR,A	
8118			MOV A,#41	Data
811A			MOV DPTR,#FFC0	8253-Data Reg. address
811D			MOVX@DPTR,A	
811E			MOVX A,@DPTR	
811F			MOV DPTR,#8200	
8122			MOVX@DPTR,A	
8123	HERE:		SJMP HERE	

RESULT:

The program receives the character 41H and stores it at location 8200H.

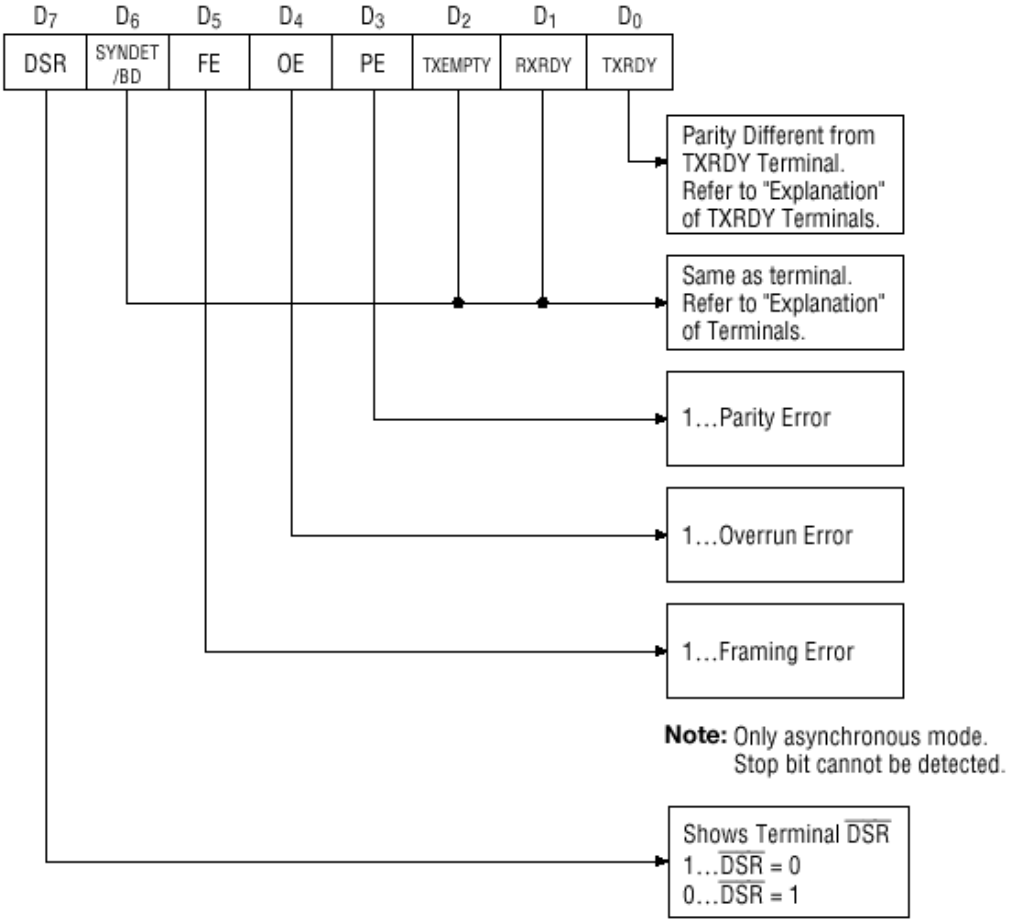
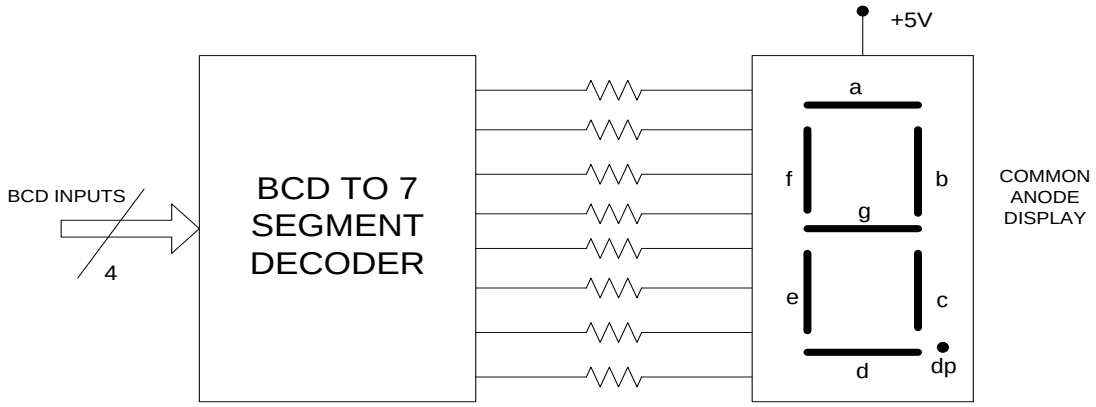
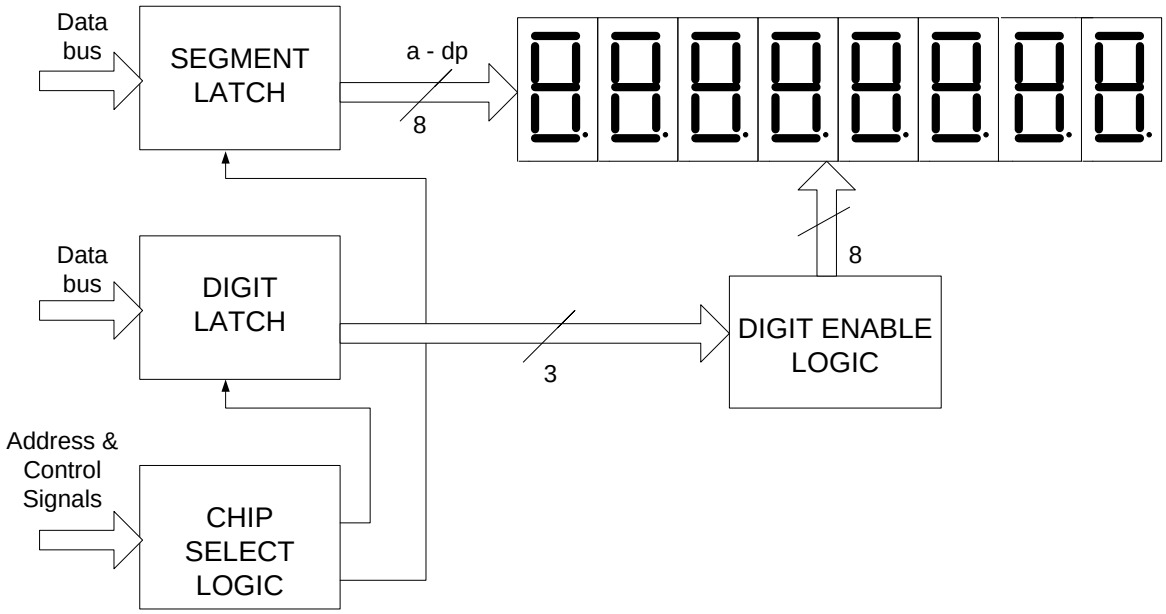


Fig. 7: Bit Configuration of Status Word


+5V
BCD INPUTS
4
BCD TO 7
SEGMENT
DECODER
a
f
b
g
e
c
d
dp
COMMON
ANODE
DISPLAY


Data
bus
SEGMENT
LATCH
a - dp
8
Data
bus
DIGIT
LATCH
3
DIGIT ENABLE
LOGIC
8
Address &
Control
Signals
CHIP
SELECT
LOGIC

EX.NO: **DATE:**

SEVEN SEGMENT LED DISPLAY USING 8051 MICROCONTROLLER

AIM

To develop programs for displaying fixed message and rolling message using 8051 Microcontroller.

THEORY

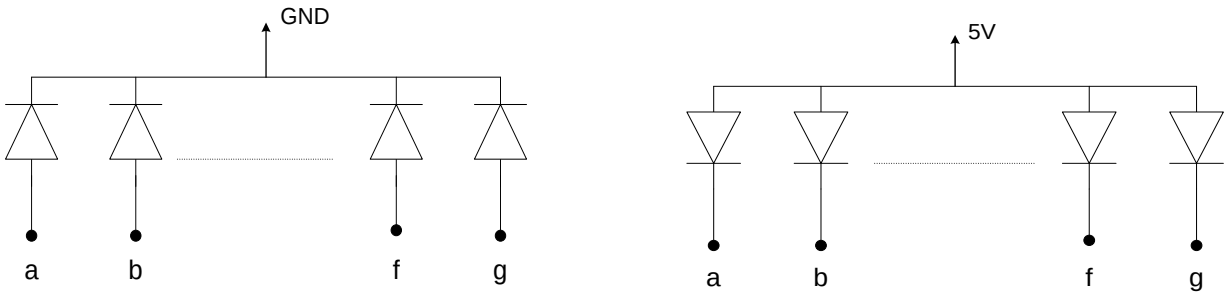
The seven segment display is the most commonly used, least expensive and the easiest to interface with the microprocessor. There are two types of seven segment displays

They are

(i) Common cathode display

(ii) Common anode display

Common cathode display **Common anode display**



For a common anode display, a low is applied to a segment to turn it on. For a Common cathode display a high is applied to a segment to turn it on. In this board Common cathode display is used.

Static Displays

A seven segment display is driven by a BCD to seven segment driver / decoder. When the BCD input is sent to the inputs of the BCD-to-seven segment decoder, it outputs low on the segments required to display the number represented by the BCD code. This circuit is referred to as static display.

Disadvantages of Static Displays

1. Power consumption is very high when more number of seven segment displays are used. The current required by the decoders and LED displays might be several times the current required by the rest of the circuitry in the instrument used.

2. Separate decoders are required for each seven segment display.

Multiplexed Displays

To overcome the disadvantage of static displays, multiplexed displays are used. The trick of multiplexing displays is that the segment information is sent out to all of the digits on a common bus. But only one display is turned ON at a time, at which the information is displayed. So it is possible to display 8 information at different digits by simultaneously switching the data as well as the digit-location in a sequential order.

Program 1

To display a fixed message "HELLO"

VBMB-016 is used to display 8 digits of information using a simple software routine. The algorithm used in the software is as follows

- Select the most significant digit by issuing a suitable control word to the digit select port.
- Out the Hex code corresponding to the first digit information to the data port.
- Introduce a delay for atleast 2 msec.
- Select the second digit.
- Out second digit Hex code.
- Introduce delay for 2ms.
- Likewise repeat for all digits.
- After displaying the 8^{th} digit, repeat from the first digit.

ADDRESS	LABEL	OPCODE	MNEMONICS	COMMENTS
4100	START		MOV DPTR, # 4200	Move the data pointer with the memory address of the message to be displayed
4103			MOV R2, DPL	
4105	CON		MOV R3, DPH	
4107			MOV R0, # 07	
4109			MOV R7, # 08	Initialize no. of digits to scan
410B	L1		MOV A, R0	Select digit position
410C			MOV DPTR, # FFC0	

410F			MOVX @ DPTR, A	
4110			MOV DPL, R2	Out the Hex code corresponding to the digit information to the data port
4112			MOV DPH, R3	
4114			MOVX A, @ DPTR	
4115			MOV DPTR, # FFC8	
4118			MOVX @ DPTR, A	
4119			LCALL DELAY	Call the delay subroutine program
411C			INC R2	Select the next digit by incrementing the data pointer
411D			DEC R0	Check if 8 digits are displayed, if not repeat from first digit
411E			DJNZ R7, L1	
4120			JMP START	After displaying the 8th digit, repeat from the first digit
4122	DELAY		MOV R4, # 02	Delay program for 2 msec delay
4124	L3		MOV R5, # FF	
4126	L2		DJNZ R5, L2	
4128			DJNZ R4, L3	
412A			RET	

Character	D7	D6	D5	D4	D3	D2	D1	D0	Hex	Address
H	0	1	1	1	0	1	1	0	76	4200
E	0	1	1	1	1	0	0	1	79	4201
L	0	0	1	1	1	0	0	0	38	4202
L	0	0	1	1	1	0	0	0	38	4203
O	0	0	1	1	1	1	1	1	3F	4204

Data - 0 to 7

Character	D7	D6	D5	D4	D3	D2	D1	D0	Hex	Address
0	0	0	1	1	1	1	1	1	3F	4200
1	0	0	0	0	0	1	1	0	06	4201
2	0	1	0	1	1	0	1	1	5B	4202
3	0	1	0	0	1	1	1	1	4F	4203
4	0	1	1	0	0	1	1	0	66	4204
5	0	1	1	0	1	1	0	1	6D	4205
6	0	1	1	1	1	1	0	1	7D	4206
7	0	0	0	0	0	1	1	1	07	4207
Blank	0	0	0	0	0	0	0	0	00	4208
Blank	0	0	0	0	0	0	0	0	00	4209
Blank	0	0	0	0	0	0	0	0	00	420A
Blank	0	0	0	0	0	0	0	0	00	420B
Blank	0	0	0	0	0	0	0	0	00	420C
Blank	0	0	0	0	0	0	0	0	00	420D
Blank	0	0	0	0	0	0	0	0	00	420E
0	0	0	1	1	1	1	1	1	3F	420F
1	0	0	0	0	0	1	1	0	06	4210
2	0	1	0	1	1	0	1	1	5B	4211
3	0	1	0	0	1	1	1	1	4F	4212
4	0	1	1	0	0	1	1	0	66	4213
5	0	1	1	0	1	1	0	1	6D	4214
6	0	1	1	1	1	1	0	1	7D	4215
7	0	0	0	0	0	1	1	1	07	4216

Program 2

To display a Rolling message “0 to 7”

- This program displays a block of information (integers 0-7) in a rolling mode.

- The data are stored from memory location 4200. The contents of memory locations from 4200H must be as follows. First block from 4200-4207 is displayed, then 4201-4208 is displayed, likewise the message is left shifted and displayed.

ADDRESS	LABEL	OPCODE	MNEMONICS	COMMENTS
4100	REP		MOV R1, 0F	Delay for rolling
4102			MOV R4, 00	
4104	CON		MOV DPTR, # 4200	Move the data pointer with the memory address of the message to be displayed
4107			MOV R2, DPL	
4109			MOV R3, DPH	
410B			MOV A, R4	
410C			ADD A, R2	
410D			MOV R2, A	
410E			MOV R0, # 07	
4110			MOV R7, # 08	Initialize no. of digits to scan
4112	L5		DEC R7	
4113			MOV A, R0	Select digit position
4114			MOV DPTR, # FFC0	
4117			MOVX @ DPTR, A	
4118			MOV DPH, R3	Out the Hex code corresponding to the digit information to the data port
411A			MOV DPL, R2	
411C			MOVX A, @ DPTR	
411D			MOV DPTR, # FFC8	

4120			MOVX @ DPTR, A	
4121			LCALL DELAY	Call the delay subroutine program
4124			INC R2	Select the next digit by incrementing the data pointer
4125			DEC R0	Check if 8 digits are displayed, if not repeat from first digit
4126			CJNE R7, # 00, L5	
4129			DJNZ R1, CON	
412B			MOV A, R2	
412C			MOV R6, # 07	
412E			SUBB A, R6	
412F			MOV R2, A	
4130			CJNE R2, # 0F, RE	Check for the completion of rolling process of 16 characters
4133			JMP REP	
4135	RE		INC R4	
4136			JMP CON	Repeat from the first digit
4138	DELAY		PUSH DPL	Delay program for 2 msec delay
413A			PUSH DPH	
413C			MOV DPL, # 02	
413F	L3		MOV DPH, # FF	
4142	L2		DJNZ DPH, L2	
4145			DJNZ DPL, L3	
4148			POP DPH	
414A			POP DPL	
414C			RET	

Result

The programs for displaying fixed and rolling messages using 8051 microcontroller has been developed and the results were verified.

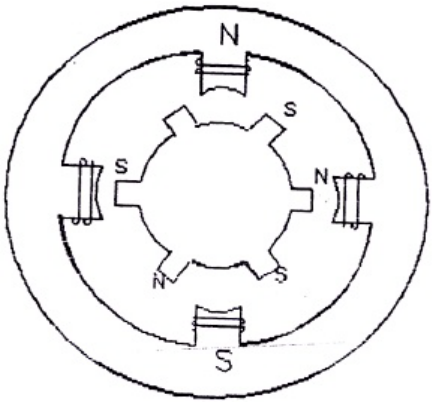


Fig. 1 Stepper motor-Four pole structure

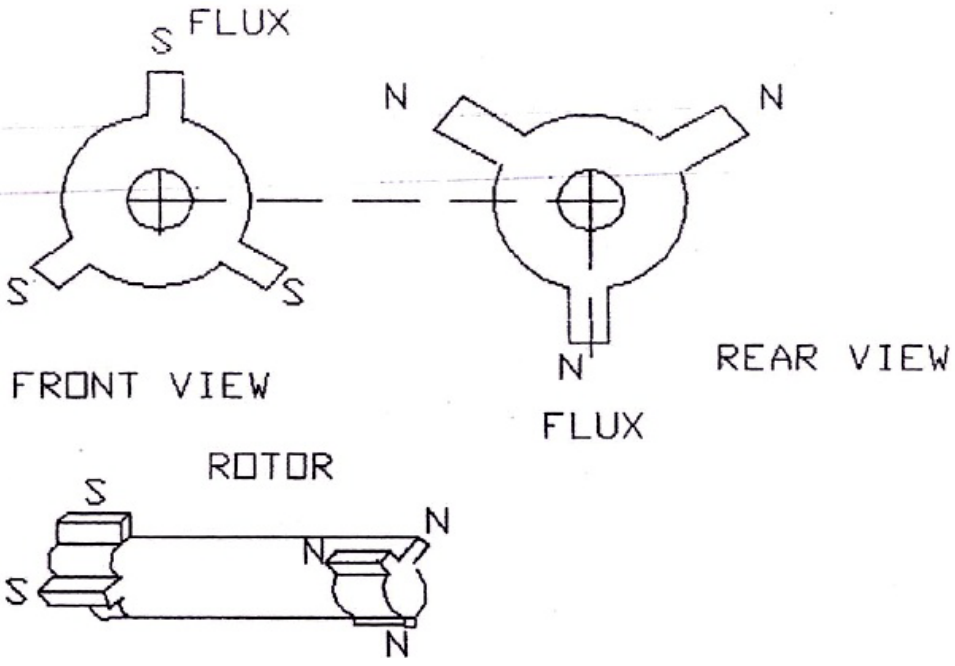


Fig. 2 Stepper motor- Cross sectional view

EX.No: **DATE:**

STEPPER MOTOR CONTROL USING 8051 MICROCONTROLLER

AIM

To control stepper motor in Clockwise and Anticlockwise direction using 8051 microcontroller.

THEORY

STEPPER MOTOR SPECIFICATION

Supply voltage	:	12V
No. of stacks	:	2
Wt	:	2Kg
Total no. of rotor teeth	:	100

Stepper motor

A Stepper motor is a device which converts digits pulses to precise angular movements. The rotory motion occurs in a stepwise manner from one equilibrium position to the next. Stepper motors are widely used in simple position control systems in the open and closed loop mode. It is used for variety of applications such as computer peripherals (printers, disk driver etc.) and in the areas of process control machine tools, medicine, numerically controlled machines and robotics.

A Stepper motor could be either of the reluctance type or of the permanent magnet type (PM). A PM stepper motor consists of multi-phase stator and two part permanent magnet rotor. The VR stepper motor has un-magnetised rotor. PM stepper motor is the most commonly used type. The basic two phase stepper motor consists of two pairs of stator poles. Each of the four poles has its own winding. The excitation of any one winding generates a north pole (N), a south pole(S) gets induced at the diametrically opposite side.

As shown in the Fig.1, the four pole structure is continuous with the stator frame and the magnetic field passes through the cylindrical stator annular ring. The rotor magnetic system has two end faces. The left face is permanently magnetised as South Pole and the right face as north Pole. The South pole structure and the North Pole structure possess similar pole faces.

The North Pole structure is twisted with respect to the South Pole structure, so that South Pole comes precisely between two north poles. The North Pole structure is offset with respect to the South Pole structure by one pole pitch. The cross sectional view is shown in Fig. 2. In an arrangement where there are four stator poles and three pairs of rotor poles, there exists 12 possible stable positions in which a south pole of the rotor can lock with a north pole of the stator.

There are three different schemes available for "stepping" a stepper motor. They are:

(a)Wave scheme

(b) 2-phase scheme and (c) Half stepping or mixed scheme

Table for 2-phase switching scheme

STEP	Clockwise									Anticlockwise								
					A1	A2	B1	B2						A1	A2	B1	B2	
	D7	D6	D5	D4	D3	D2	D1	D0	Firing angle	D7	D6	D5	D4	D3	D2	D1	D0	Firing Angle
1	x	x	x	x	1	0	0	1	09	x	x	x	x	1	0	1	0	0A
2	x	x	x	x	0	1	0	1	05	x	x	x	x	0	1	1	0	06
3	x	x	x	x	0	1	1	0	06	x	x	x	x	0	1	0	1	05
4	x	x	x	x	1	0	1	0	0A	x	x	x	x	1	0	0	1	09

2-phase scheme

In this scheme, any two adjacent stator windings are energised. There are two magnetic fields active in quadrature and none of the rotor pole faces can be in direct alignment with the stator poles. A partial but symmetric alignment of the rotor poles is of course possible. Typical equilibrium conditions of the rotor when the windings on two successive stator poles are excited is illustrated in Fig.3. In step (a), A1 and B1 are energised. The pole-face S1 tries to align itself with the axis of A1 (N) and the pole face S2 with B1(N). The north pole N3 of the rotor finds itself in the neutral zone between A1(N) and B1(N). S1 and S2 of the rotor, position themselves symmetrically with respect to the two stator north pole. Next, when B1 and A2 are energised, S2 tends to align with B1(N) and S3 with A2(N). Of course, again under equilibrium conditions, only partial alignment is possible and N1 finds itself in the neutral region, midway between B1(N) and A2(N) [Step (b)]. In step (c), A2 and B2 are on. S3 and S1 tend to align with A2(N) and B2(N), respectively, with N2 in the neutral zone. Step (d) illustrates the case when A1 and B2 are on. The switching sequence for the 2-phase scheme is given below

The stepping action is caused by sequential switching of the power supply to 2 phases of motor having double winding with centre taps. The angle of each step for the motor can be calculated as

$$\text{Step angle in degrees} = \frac{360^0}{N_s * N_r}$$

Where,

N_s is the No. of the stacks.

N_r is the No. of rotor teeth.

$$\text{Step angle in degrees} = \frac{360^0}{2*100} = 1.8^0$$

Therefore,

No. of steps / revolutions = $\frac{360^0}{1.8}$= 200 steps

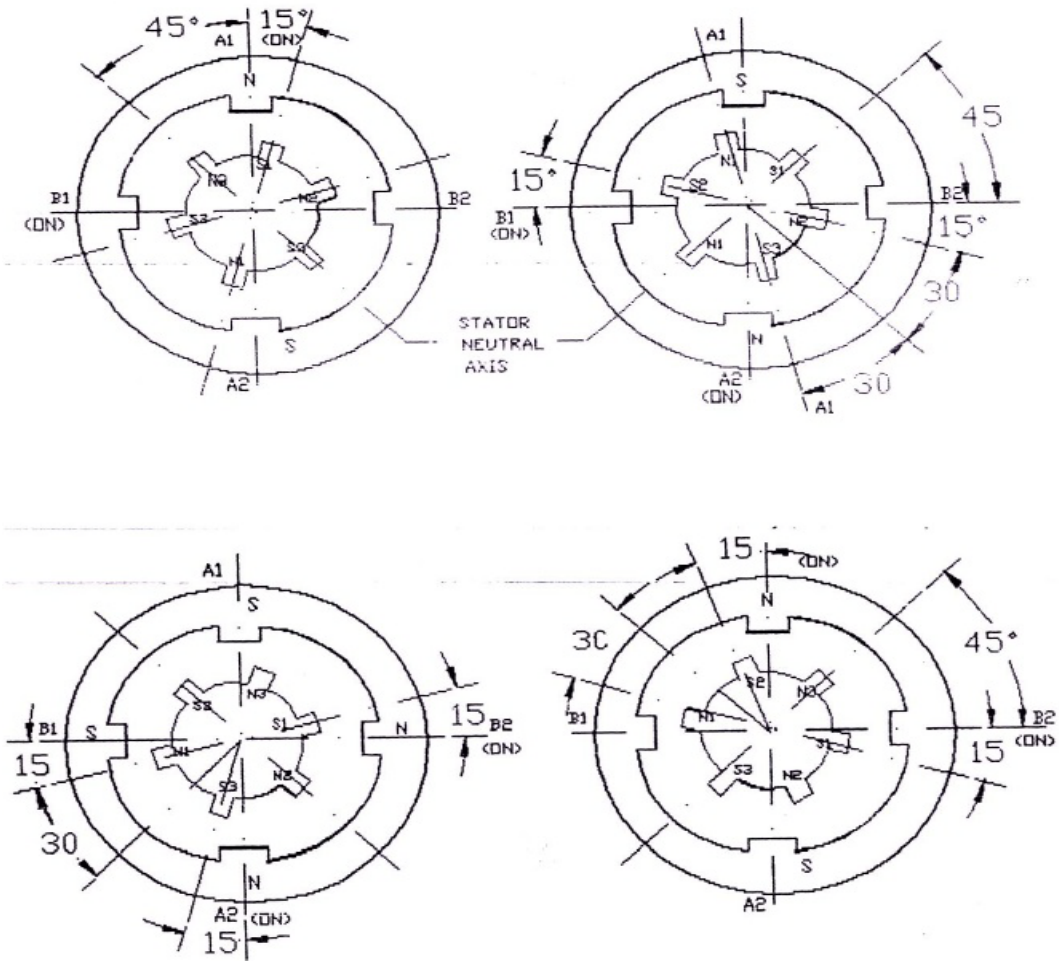
The firing pulses corresponding to data 0A, 06, 05, & 09 cause 4 steps movement. i.e., in each sequence it produces 4 steps.

Therefore,

$$\text{No. of sequence requiring to produce one revolution} = \frac{\text{No. of steps / revolution}}{\text{No. of step / sequence}}$$

$$= \frac{200}{4} = 50 \text{ d} = 32 \text{ h}$$

step (a) **step (b)**

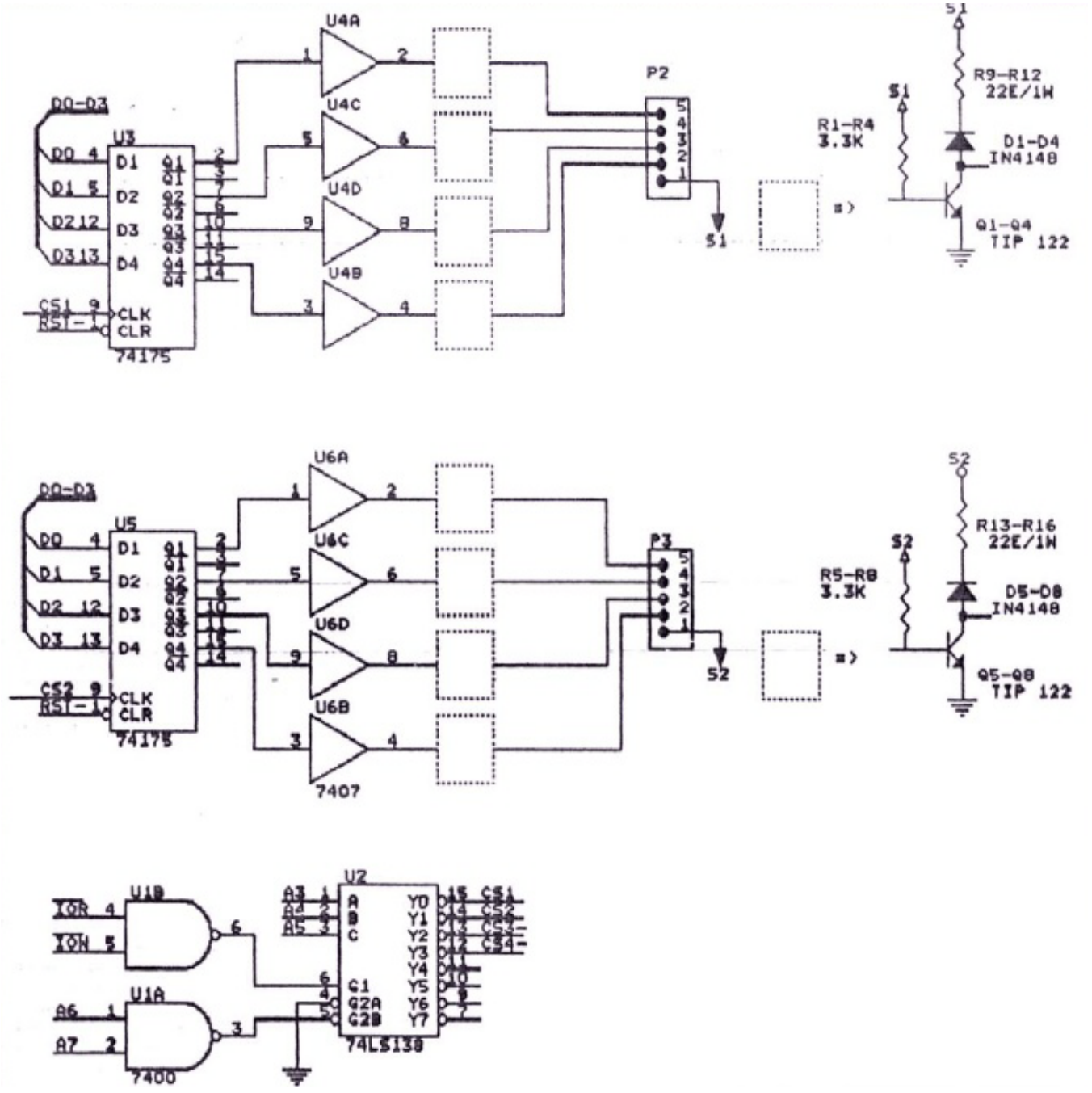


step (c) **step (d)**

Fig. 3 2-phase drive scheme

Program

ADDRESS	LABEL	OPCODE	MNEMONICS	COMMENTS
4100	START		MOV R4, # 32 H	Initialize a counter with the No. of sequence required to produce one revolution
4102	L2		MOV DPTR, # FD	Move DPTR with the memory address having clockwise sequence
4105			LCALL L1	Call the loop L1 for 4-step sequence
4108			DJNZ R4, L2	Check for the completion of one revolution, if not completed go to loop L2
410A			LCALL DELAY	Delay between forward and reverse rotation
410D			MOV R4, # 32 H	Initialize a counter with the No. of sequence required to produce one revolution
410F	L3		MOV DPTR, # RV	Move DPTR with the memory address having Anticlockwise sequence
4112			LCALL L1	Call the loop L1 for 4-step sequence
4115			DJNZ R4, L3	Check for the completion of one revolution, if not completed go to loop L3
4117			LCALL DELAY	Delay between forward and reverse rotation
411A			SJMP START	Start from beginning
411C	L1		MOV R0, # 04 H	
411E	LOOP		MOVX A, @ DPTR	
411F			PUSH 83 H	
4121			PUSH 82 H	
4123			MOV DPTR, # FFC0 H	
4126			MOV R2, # 04 H	



Stepper Motor Interface

4128	L7		MOV R1, # 05 H	
412A	L6		MOV R3, # FF H	
412C	L4		DJNZ R3, L4	
412E			DJNZ R1, L6	
4130			DJNZ R2, L7	
4132			MOVX @ DPTR, A	
4133			POP 82 H	
4135			POP 83 H	
4137			INC DPTR	
4138			DJNZ R0, LOOP	
413A			RET	
413B	DELAY		MOV R5, # 01 H	Delay program
413D	L9		MOV R2, # 05 H	
413F	L8		DJNZ R2, L8	
4141			DJNZ R5, L9	
4143			RET	
4200	FD	09,05,06,0A		Data for forward and reverse rotation
4204	RV	0A,06,05,09		

Result

The control of stepper motor using 8051 Microcontroller has been studied by changing the direction of rotation and speed.

EX.NO: **DATE:**

KEYBOARD DISPLAY INTERFACE 8279 USING 8051 MICROCONTROLLER

AIM

To study the performance of keyboard display interface 8279 and to execute the following programs.

1. To display character ‘A’
2. To display a Rolling message ‘HELP US’
3. To read a key
3. To accept a key and display it

THEORY

The 8279 is a Hardware approach to interface a matrix keyboard & a multiplexed display. The keyboard segment can be connected to a 64 contact key matrix. Keyboard entries are debounced and stored in the internal FIFO memory; an interrupt signal is generated with each entry. The display segment can provide a 16 character scanned display interface with LED’S. This segment has 16*8 R/W memory (RAM), which can be used to read/write information in right entry or left entry format.

The four major sections of 8279 are

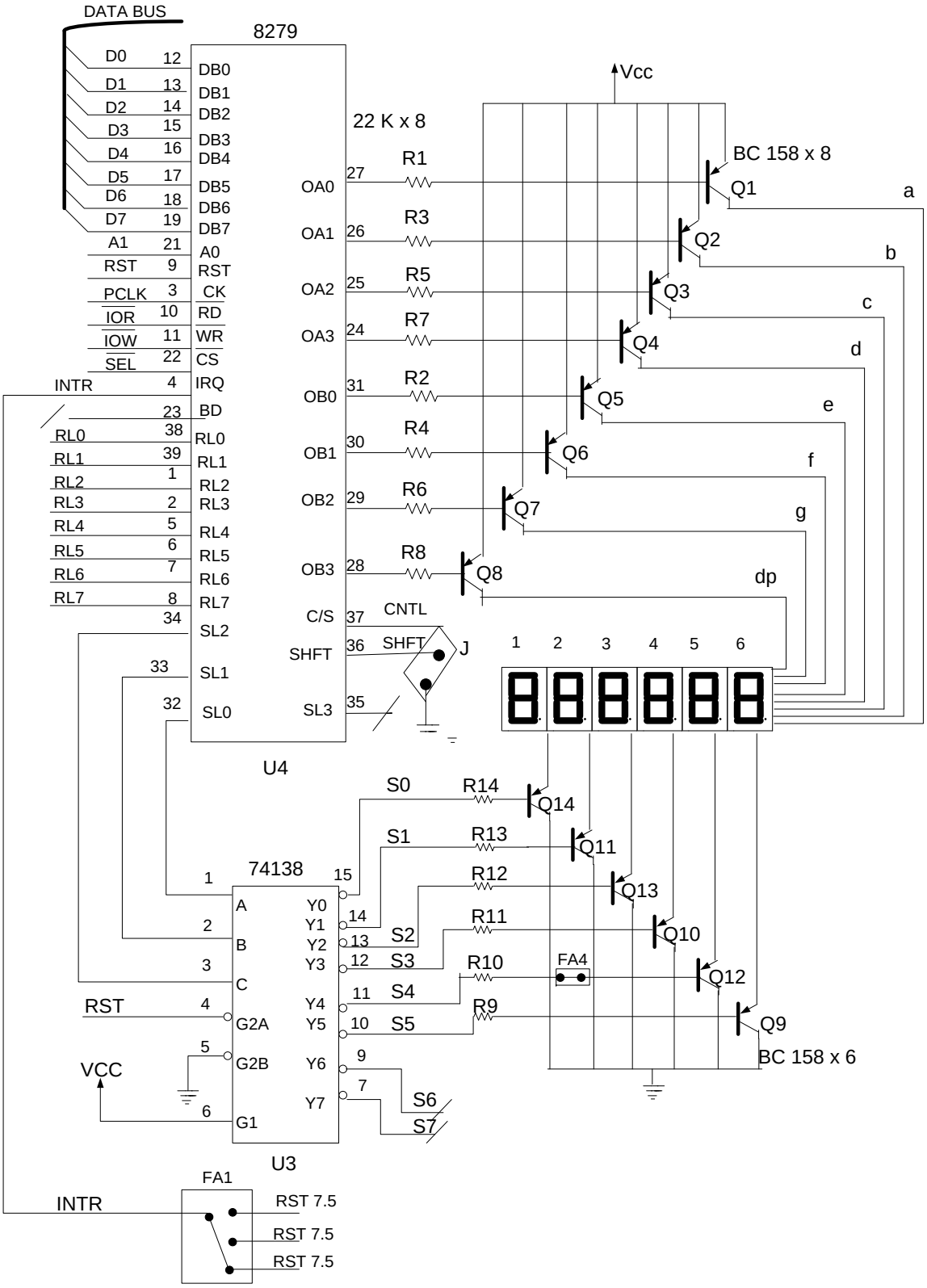
1. Keyboard
2. Scan
3. Display
4. Microprocessor interface

KEYBOARD SECTION

The keyboard section consists of 16 keys and the keys are automatically debounced. The keyboard can be operated in 2 modes.

1. Two-key lockout
2. N-key roll over

In the two-key lock out mode, if two keys are pressed simultaneously, only the first key is recognised. In the N-key rollover mode, simultaneous keys are recognised and their codes are stored in the internal buffer.



KEYBOARD DISPLAY INTERFACE

SCAN SECTION

This section has 4 scan lines SL0-SL3. The 4 scan lines are decoded using 4 to 16 decoder to generate 16 lines for scanning. These lines can be connected to the rows of a matrix keyboard & digit drivers of multiplexed display.

DISPLAY SECTION

This section has 8 output lines divided into 2 groups A0-A3 & B0-B3. They can be either as 8 lines or 2 groups of 4 lines in conjunction with scan lines for multiplexed display. The display can be blanked using BD lines. This section includes 16*8 display RAM. The microprocessor can read from or write into any of these registers.

MICROPROCESSOR INTERFACE SECTION

This includes 8 bidirectional data lines DB0-DB7, one IRQ and 6 lines for interfacing. The IRQ line goes high whenever data entries are stored in FIFO. This signal is used to interrupt the microprocessor to indicate availability of data.

Display mode setup

The command word for keyboard and display mode is,

0	0	0	D	D	K	K	K

DD – Display Mode

0 0 8- bit character display – left entry

0 1 16 - bit character display – left entry

1 0 8 - bit character display – right entry

1 1 16 - bit character display – right entry

KKK – Keyboard Mode

0 0 0 Encoded scan keyboard – 2 key lockout

0 0 1 Encoded scan keyboard – 2 key lockout

0 1 0 Encoded scan keyboard – N key rollover

0 1 1 Decoded scan keyboard – N key rollover

1 0 0 Encoded scan sensor matrix

1 0 1 Decoded scan sensor matrix

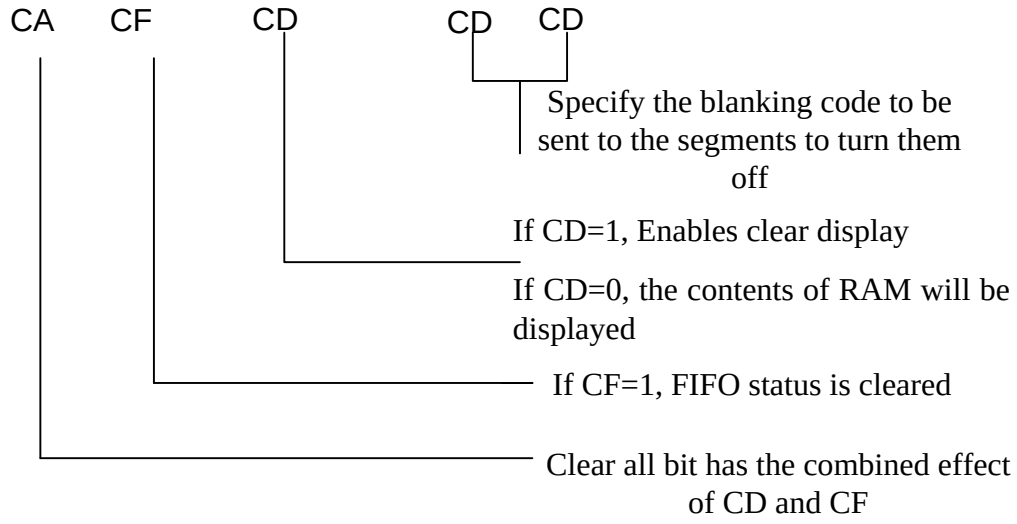
1 1 0 Strobed input, Encoded display scan

1 1 1 Strobed input, Decoded display scan

Clear Display

The command word for clear display is,

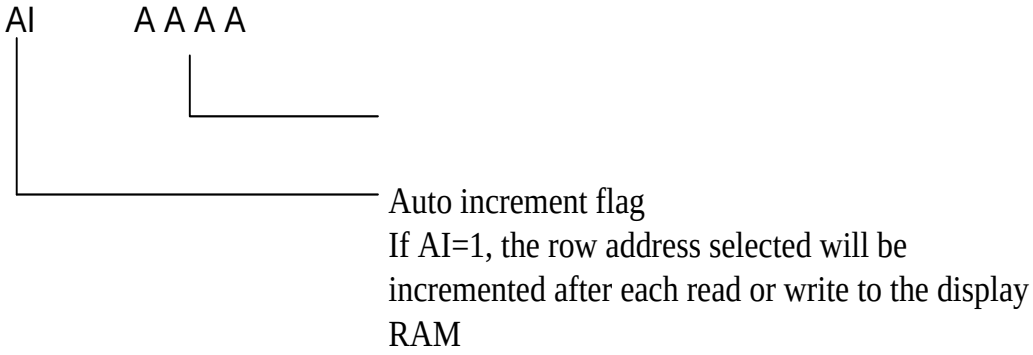
1	1	0	CD	CD	CD	CF	CA



Write display RAM

The command word for write display RAM is,

1	0	0	AI	A	A	A	A



KEYBOARD DISPLAY INTERFACE

Program 1

To display character 'A'

To initialize 8279 and to display the character 'A' in the first digit of the display.

ADDRESS	LABEL	OPCODE	MNEMONICS	COMMENTS
4100			MOV DPTR, # FFC2	FFC2- Address of control/status register
4103			MOV A, # 00	Set mode and display
4105			MOVX @DPTR, A	
4106			MOV A, # CC	Clear display
4108			MOVX @DPTR, A	
4109			MOV A, # 90	Write display
410B			MOVX @DPTR, A	
410C			MOV DPTR, # FFC0	FFC0- Address of data register
410F			MOV A, # 88	Display the character 'A'
4111			MOVX @DPTR, A	
4112			MOV R0, # 05	Blank the rest of the display
4114			MOV A, # FF	
4116	LOOP		MOVX @DPTR, A	
4117			DJNZ R0, LOOP	
4119	HERE		SJMP HERE	

Program 2

To display a Rolling message 'HELP US'

ADDRESS	LABEL	OPCODE	MNEMONICS	COMMENTS
4100	START		MOV DPTR, # FFC2	FFC2- Address of control/status register
			MOV R0, # 00	To initialize look-up table at 4400
			MOV R1, # 44	
4103			MOV A, # 10	Set mode and display
4105			MOVX @DPTR, A	
4106			MOV A, # CC	Clear display
4108			MOVX @DPTR, A	
4109			MOV A, # 90	Write display
410B			MOVX @DPTR, A	
	LOOP		MOV DPH, R1	
			MOV DPL, R0	
			MOVX A, @ DPTR	
410C			MOV DPTR, # FFC0	FFC0- Address of data register
4111			MOVX @DPTR, A	
			LCALL DELAY	
			INC R0	
			CJNE R0, # 0F, LOOP	
4119			SJMP START	

Look up table –HELP US

Memory Address	Hex code	Display Character
4400	FF	Blank
4401	FF	Blank
4402	FF	Blank
4403	FF	Blank
4404	FF	Blank
4405	FF	Blank
4406	FF	Blank
4407	FF	Blank
4408	98	H
4409	68	E
440A	7C	L
440B	C8	P
440C	FF	Blank
440D	1C	U
440E	29	S
440F	FF	Blank

Delay Program

ADDRESS	LABEL	OPCODE	MNEMONICS	COMMENTS
4500			MOV R4, #A0	
4502	LOOP2		MOV R5, # FF	
4504	LOOP1		NOP	
4505			DJNZ R5, LOOP1	
4507			DJNZ R4, LOOP2	
4509			RET	

Program 3

To read a key

ADDRESS	LABEL	OPCODE	MNEMONICS	COMMENTS
4100			MOV DPTR, # FFC2	FFC2- Address of control/status register
4103	WAIT		MOV A, @ DPTR	Check for a key closure- To identify key closure last 3 bits of status word is continuously monitored.
4104			ANL A, # 07	
4106			JZ WAIT	
4108			MOV A, # 40	To read the FIFO RAM register
410A			MOVX @DPTR, A	
410B			MOV DPTR, # FFC0	FFC0- Address of data register
410E			MOV A, @DPTR	
410F			MOV DPTR, # 4200	
4112			MOVX @DPTR, A	Key code is stored in the memory location 4200
4113	HERE		SJMP HERE	

Look up table

Memory Address	Hex code	Display Character
4200	0C	0
4201	9F	1
4202	4A	2
4203	0B	3
4204	99	4
4205	29	5
4206	28	6
4207	8F	7
4208	08	8
4209	09	9
420A	88	A
420B	38	B
420C	6C	C
420D	1A	D
420E	68	E
420F	E8	F

Program 4

To accept a key and display it

To initialize 8279 and to accept and display a key pressed.

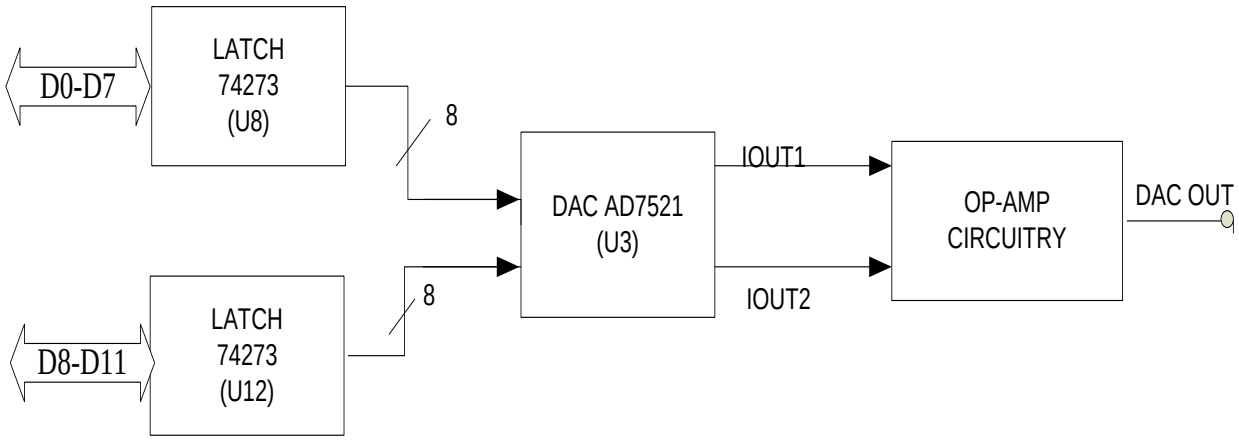
ADDRESS	LABEL	OPCODE	MNEMONICS	COMMENTS
4100			MOV R1, # 42	Set the pointer to 4200
4102			MOV R0, # 08	
4104			MOV DPTR, # FFC2	FFC2- Address of control/status register
4107			MOV A, # 00	Set mode and display
4109			MOVX @DPTR, A	
410A			MOV A, # CC	Clear display
410C			MOVX @DPTR, A	
410D			MOV A, # 90	Write display
410F			MOVX @DPTR, A	
4110			MOV A, # FF	
4112			MOV DPTR, # FFC0	FFC0- Address of data register
4115	LOOP		MOVX @DPTR, A	
4116			DJNZ R0, LOOP	
4118	HERE		MOV DPTR, # FFC2	
411B	WAIT		MOV A, @ DPTR	Check for a key closure- To identify key closure last 3 bits of status word is continuously monitored.
411C			ANL A, # 07	
411E			JZ WAIT	
4120			MOV A, # 40	To read the FIFO RAM register
4122			MOVX @DPTR, A	
4123			MOV DPTR, # FFC0	
4126			MOV A, @DPTR	
4127			ANL A, # 0F	Get the key code pressed

4129			MOV DPL, A	
412B			MOV DPH, R1	
412D			MOVX A, @DPTR	Code from look up table
412E			MOV DPTR, # FFC0	
4131			MOVX @DPTR, A	Display the key pressed
4132			SJMP HERE	Jump to loop with the label HERE

Result

Thus various programs has been executed with 8279 interfaced with 8051 microcontroller.

Interfacing DAC to 8097 Microcontroller



EX.NO: **DATE:**

APPLICATIONS OF 8097 MICROCONTROLLER

A) DIGITAL TO ANALOG CONVERTER

AIM

To generate saw tooth, triangular and square waveform using 8097 microcontroller, DAC AD 7521.

APPARATUS REQUIRED

1. Microcontroller(8097) kit
2. CRO
3. Probes

THEORY

The 12 data lines (D0-D11) are connected to the DAC through an 8 bit and 4 bit latches. The current outputs are passed onto an Op-amp circuit which gives the voltage output. This voltage output is differential and a maximum of 20v peak to peak can be obtained by changing the load resistor value of the Op-amp.

PROCEDURE

1. Connect the microcontroller 8097 analog I/O bus pin 10 to the CRO channel.
2. Switch on the power supply and enter the program.
3. Execute the programs and trace the waveforms.

Program 1-Saw tooth waveform

Address	Opcode	Label	Mnemonics	Comments
8100		START	LDAX, #0FF10H	Load DAC port-A add in AX reg.
8104			LD BX,#0000	Load BX reg with 0000
8108		LOOP	ST BX, [AX]	Out BX through PORT-A
810B			ADD BX, #0001	Add 0001 to BX reg
810F			CMP BX,#07FFH	Compare BX with 07FF
8113			JNE LOOP	Not equal ,Out BX value, till the condition satisfied
8115			SJMP START	Short Jump

Program 2-Triangular waveform

Address	Opcode	Label	Mnemonics	Comments
8100		START	LDAX, #0FF10H	Load DAC port-A add in AX reg.
8104			LD BX,#0000	Load BX reg with 0000
8108		LOOP1	ST BX, [AX]	Out BX through PORT-A
810B			ADD BX, #0001	Add 0001 to BX reg
810F			CMP BX,#01FFH	Compare BX with 01FF
8113			JNE LOOP1	Check the condition
8115		LOOP2	ST BX,[AX]	Out BX through PORT-A
8118			SUB BX,#0001	Subtract 0001 from BX reg.
811C			CMP BX, #0000H	Compare BX with 0000
8120			JNE LOOP2	Check the condition
8122			SJMP START	Short Jump

Program 3-Square waveform

Address	Opcode	Label	Mnemonics	Comments
8100			LDAX, #0FF10H	Load DAC port-A add in AX reg.
8104		START	LD BX,#0000	Load BX reg with 0000
8108			ST BX, [AX]	Out BX through PORT-A
810B			SCALL DLY	Short call Delay
810D			LD BX,#0FFFH	Load Bx reg with 0FFF
8111			ST BX, [AX]	Out BX through PORT-A
8114			SCALL DLY	Short call Delay
8116			SJMP START	Start Jump
8118		DLY	LD CL,#FF	Delay Program
811B		LOOP	SUB CL,#01	
811E			CMP CL,#00	
8121			JNE LOOP	
8123			RET	

B) ANALOG TO DIGITAL CONVERTER

AIM:

To convert the given analog input to digital output.

APPARATUS REQUIRED:

1. Microcontroller(8097) kit
2. Potential Divider
3. Digital Mutimeter
4. 5v DC source

THEORY:

The 8097 has eight Analog to Digital converter channels. The special features of the on-chip ADC are:

1. 10- bit successive approximation type ADC
2. 22μsec conversion time
3. 0 to 5v unipolar input to ADC
4. Built in sample and Hold
5. ADC sampling through the internal timer of 80196.

Analog to digital conversion can be initiated in any one of the ADC channels. The channel can be writing onto the ADC command register. The AD conversion begins by writing onto the command register or by means of trigger from the high speed output unit. The command register is at on- chip memory address 02. The format of ADC command register is:

7	6	5	4	3	2	1	0
X	X	X	X	GO	-----	CH#	-----

CH# selects which of the eight analog input channels is to be converted to digital form.

GO indicates when the conversion is to be initiated. When GO=1 means start now. GO=0 means conversion is to be initiated by HSO unit at the specified time.

Results of the ADC are read from the ADC result register at locations 02 and 03. Note that the ADC status bit may not be set until 8 state times after the GO command, so it is necessary to wait 8 state times before testing it.

PROCEDURE:

1. Connect a voltage source to the ADC channel through the potential divider arrangement.
2. Enter the program.
3. Verify the results for an different analog input.

Analog to Digital Converter:

ANALOG (SET VALUE)	DIGITAL	ANALOG (CALCULATED VALUE)	DIGITAL(CALCULATED VALUE)
4.9			
3.7			
2.9			
1.78			
0.02			

Specimen Calculation:

Program- Analog to Digital Converter

Address	Opcode	Label	Mnemonics	Comments
8100			ldb ad_command, #0AH	Command word select the ch. Selection and initialize the AD conversion
8103			Nop	Wait for 8 state times for the
8104			Nop	ADC status to set
8105		Check	Jbs ad_result_lo,3,Check	check the ADC result reg.bit 3
8108			ldb al, ad_result_lo	If bit3 =0, wait
810B			ldb ah, ad_result_hi	If bit 3=1, ADC in progress
810E			shr ax, #6	shift ax reg contents with 6 times
8111			and ax, #3ffH	AND operation with 3ff
8115			ld bx, #9000H	load the address to BX reg.
8119			st ax,[bx]	Store the result in BX
811C		Jump	sjmp Jump	Short Jump

Pulse width Modulator:

Duty Cycle	Decimal Value	Hex. Value
10%		
25%		
50%		
75%		
90%		

Specimen Calculation:

C) PULSE WIDTH MODULATOR

AIM:

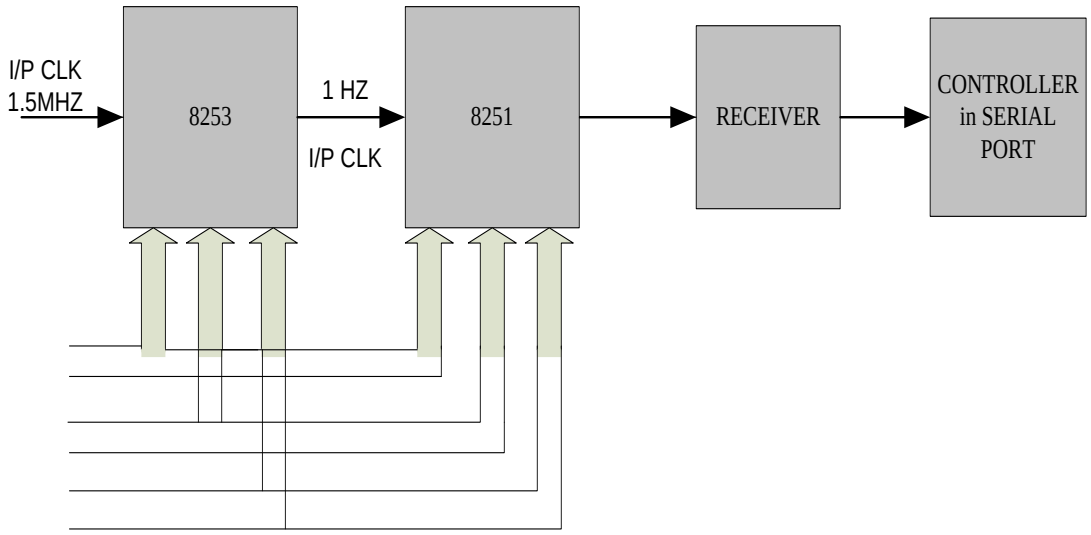
To vary the duty cycle of pulse width modulator control registers and trace the waveform.

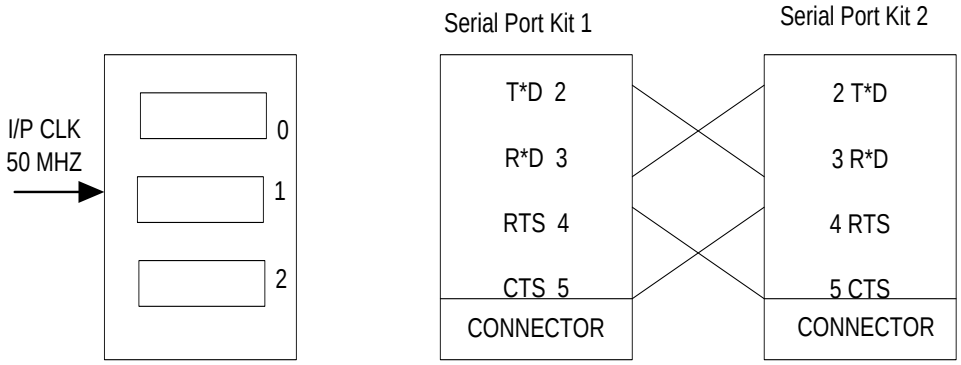
The 80196 features one pulse width modulated output channel. The PWM output waveform is a variable duty cycle, which represents every 64 microseconds. Changes in duty cycle are made by writing to the PWM control register. The PWM control register lies in the on chip RAM address 17H. The PWM output is terminated at the pin 12 of the connector analog I/O P10. The PWM output shares a pin P2.5, so that these two features cannot be used at the same time. The pin 2.5 can function as PWM or a standard port pin. The option can be selected by setting bit 0 of the I/O control register (called IOC1).

Address	Opcode	Label	Mnemonics	Comments
8100			ldb al,ioc1	ioc1 content to al reg.
8103			orb al,#01	OR oper. al with 01
8106			stb al,ioc1	store al content to ioc1
8109			ldb pwm control word,7F	Duty cycle value loaded into PWM reg.
810C			sjmp 810C	Short Jump

RESULTS:

The above programs are loaded and the results are verified.


I/P CLK
1.5MHZ
8253
1 HZ
I/P CLK
8251
RECEIVER
CONTROLLER
in SERIAL
PORT


I/P CLK
50 MHZ
0
1
2
Serial Port Kit 1
T*D 2
R*D 3
RTS 4
CTS 5
CONNECTOR
Serial Port Kit 2
2 T*D
3 R*D
4 RTS
5 CTS
CONNECTOR

EX.NO: **DATE:**

SERIAL DATA TRANSMISSION- KIT TO KIT TRANSFER USING 8051 MICROCONTROLLER

AIM

To transmit the data serially from one microprocessor to another microprocessor using 8051 microcontroller kit.

DESCRIPTION OF HARDWARE

The microprocessor kit transmitter/receiver data serially through serial port via USART 8251. The device used for transmitter is MC 1488 and for receiver MC 1489. The kit can either transmit or receive data kit which are linked to RS232 cable. The input clock frequency of 8251 is equal to product of rate factor and baudrate. The frequency of 8253 is divided by the count value in the count register.

INITIALIZATION OF 8253

Calculation of counter value

Given baud rate = 9600

Baudrate factor =16

Input clock frequency=9600*16

This is also the output frequency from channel 0 for8253. Count to be closed in the registor= (i/p clk)/(o/p clk).

TRANSMITTER PROGRAM

ADDRESS	LABEL	OPCODE	MNEMONICS	COMMENTS
4100		75 89 20	MOV TMOD, #20H	
4103		75 8D A0	MOV TH1, #0A0H	
4106		75 8B 00	MOV TL1, #00H	
4109		75 88 40	MOV TCON, #40H	
410C		75 98 58	MOV SCON, #58H	
410F		90 45 00	MOV DPTR,#4500H	
4112	RELOAD	7D 05	MOV R5,#05H	
4114	REPEAT	E0	MOVX A,@DPTR	
4115		F5 99	MOV SBUF,A	
4117	CHECK	30 99 FD	JNB SCON.1,CHECK	
411A		C2 99	CLR SCON.1	
411C		A3	INC DPTR	
411D		B4 3F F2	CJNE A,#3FH,RELOAD	
4120		DD F2	DJNZ R5, REPEAT	
4122		E4	CLAR A	
4123		12 00 20	LCALL 0020H	

INPUT:

4500 - 00
4501 - 11
4502 - 22
4503 - 33
4504 - 44

RECEIVER PROGRAM

ADDRESS	LABEL	OPCODE	MNEMONICS	COMMENTS
4100		75 89 20	MOV TMOD, #20H	
4103		75 8D A0	MOV TH1, #0A0H	
4106		75 8B 00	MOV TL1, #00H	
4109		75 88 40	MOV TCON, #40H	
410C		75 98 58	MOV SCON, #58H	
410F		90 45 00	MOV DPTR,#4500H	
4112	RELOAD	7D 05	MOV R5,#05H	
4114	CHECK	30 98 FD	JNB SCON.0,CHECK	
4115		C2 98	CLR SCON.0	
4117		E5 99	MOV A,SBUF	
411A		F0	MOVX@DPTR,A	
411C		A3	INC DPTR	
411D		B4 3F F2	CJNE A,#3FH,RELOAD	
4120		DD F2	DJNZ R5, CHECK	
4122		E4	CLAR A	
4123		12 00 20	LCALL 0020H	

OUTPUT:

4500 - 00
4501 - 11
4502 - 22
4503 - 33
4504 - 44

RESULT:

The data is transmitted serially from one kit to another kit.

OPCODE SHEETS

1) 8085 MICROPROCESSOR

S.N0.	Mnemonics, Operand	Opcode	S. No.	Mnemonics, Operand	Opcode	S. No.	Mnemonics, Operand	Opcode
1.	ACI Data	CE	55.	DCR C	0D	109.	MOV B, D	42
2.	ADC A	8F	56.	DCR D	15	110.	MOV B, E	43
3.	ADC B	88	57.	DCR E	1D	111.	MOV B, H	44
4.	ADC C	89	58.	DCR H	25	112.	MOV B, L	45
5.	ADC D	8A	59.	DCR L	2D	113.	MOV B, M	46
6.	ADC E	8B	60.	DCR M	35	114.	MOV C, A	4F
7.	ADC H	8C	61.	DCX B	0B	115.	MOV C, B	48
8.	ADC L	8D	62.	DCX D	1B	116.	MOV C, C	49
9.	ADC M	8E	63.	DCX H	2B	117.	MOV C, D	4A
10.	ADD A	87	64.	DCX SP	3B	118.	MOV C, E	4B
11.	ADD B	80	65.	DI	F3	119.	MOV C, H	4C
12.	ADD C	81	66.	EI	FB	120.	MOV C, L	4D
13.	ADD D	82	67.	HLT	76	121.	MOV C, M	4E
14.	ADD E	83	68.	IN Port-address	DB	122.	MOV D,A	57
15.	ADD H	84	69.	INR A	3C	123.	MOV D, B	50
16.	ADD L	85	70.	INR B	04	124.	MOV D, C	51
17.	ADD M	86	71.	INR C	0C	125.	MOV D, D	52
18.	ADI Data	C6	72.	INR D	14	126.	MOV D, E	53
19.	ANA A	A7	73.	INR E	1C	127.	MOV D, H	54
20.	ANA B	A0	74.	INR H	24	128.	MOV D, L	55
21.	ANA C	A1	75.	INR L	2C	129.	MOV D, M	56
22.	ANA D	A2	76.	INR M	34	130.	MOV E, A	5F
23.	ANA E	A3	77.	INX B	03	131.	MOV E, B	58
24.	ANA H	A4	78.	INX D	13	132.	MOV E, C	59
25.	ANA L	A5	79.	INX H	23	133.	MOV E, D	5A
26.	ANA M	A6	80.	INX SP	33	134.	MOV E, E	5B
27.	ANI Data	E6	81.	JC Label	DA	135.	MOV E, H	5C
28.	CALL Label	CD	82.	JM Label	FA	136.	MOV E, L	5D
29.	CC Label	DC	83.	JMP Label	C3	137.	MOV E, M	5E
30.	CM Label	FC	84.	JNC Label	D2	138.	MOV H, A	67
31.	CMA	2F	85.	JNZ Label	C2	139.	MOV H, B	60
32.	CMC	3F	86.	JP Label	F2	140.	MOV H, C	61
33.	CMP A	BF	87.	JPE Label	EA	141.	MOV H, D	62
34.	CMP B	B8	88.	JPO Label	E2	142.	MOV H, E	63
35.	CMP C	B9	89.	JZ Label	CA	143.	MOV H, H	64
36.	CMP D	BA	90.	LDA Address	3A	144.	MOV H, L	65
37.	CMP E	BB	91.	LDAX B	0A	145.	MOV H, M	66
38.	CMP H	BC	92.	LDAX D	1A	146.	MOV L, A	6F
39.	CMP L	BD	93.	LHLD Address	2A	147.	MOV L, B	68
40.	CMP M	BD	94.	LXI B	01	148.	MOV L, C	69
41.	CNC Label	D4	95.	LXI D	11	149.	MOV L, D	6A
42.	CNZ Label	C4	96.	LXI H	21	150.	MOV L, E	6B
43.	CP Label	F4	97.	LXI SP	31	151.	MOV L, H	6C
44.	CPE Label	EC	98.	MOV A, A	7F	152.	MOV L, L	6D
45.	CPI Data	FE	99.	MOV A, B	78	153.	MOV L, M	6E
46.	CPO Label	E4	100.	MOV A, C	79	154.	MOV M,A	77
47.	CZ Label	CC	101.	MOV A, D	7A	155.	MOV M, B	70
48.	DAA	27	102.	MOV A, E	7B	156.	MOV M, C	71
49.	DAD B	09	103.	MOV A, H	7C	157.	MOV M, D	72
50.	DAD D	19	104.	MOV A, L	7D	158.	MOV M, E	73
51.	DAD H	29	105.	MOV A, M	7E	159.	MOV M, H	74
52.	DAD SP	39	106.	MOV B, A	47	160.	MOV M, L	75
53.	DCR A	3D	107.	MOV B, B	40	161.	MVI A, Data	3E

54.	DCR B	05	108.	MOV B, C	41	162.	MVI B, Data	06
S. No.	**Mnemonics, Operand**	**Opcode**	**S. No.**	**Mnemonics, Operand**	**Opcode**	**S. No.**	**Mnemonics, Operand**	**Opcode**
163.	MVI C, Data	0E	192.	RET	C9	219.	SBI Data	DE
164.	MVI D, Data	16	193.	RIM	20	220.	SHLD Address	22
165.	MVI E, Data	1E	194.	RLC	07	221.	SIM	30
166.	MVI H, Data	26	195.	RM	F8	222.	SPHL	F9
167.	MVI L, Data	2E	196.	RNC	D0	223.	STA Address	32
168.	MVI M, Data	36	197.	RNZ	C0	224.	STAX B	02
169.	NOP	00	198.	RP	F0	225.	STAX D	12
170.	ORA A	B7	199.	RPE	E8	226.	STC	37
171.	ORA B	B0	200.	RPO	E0	227.	SUB A	97
172.	ORA C	B1	209.	RST 7	FF	228.	SUB B	90
173.	ORA D	B2	210.	RZ	C8	229.	SUB C	91
174.	ORA E	B3	201.	RRC	0F	230.	SUB D	92
175.	ORA H	B4	202.	RST 0	C7	231.	SUB E	93
176.	ORA L	B5	203.	RST 1	CF	232.	SUB H	94
177.	ORA M	B6	204.	RST 2	D7	233.	SUB L	95
178.	ORI Data	F6	205.	RST 3	DF	234.	SUB M	96
179.	OUT Port-Address	D3	206.	RST 4	E7	235.	SUI Data	D6
180.	PCHL	E9	207.	RST 5	EF	236.	XCHG	EB
181.	POP B	C1	208.	RST 6	F7	237.	XRA A	AF
182.	POP D	D1	209.	RST 7	FF	238.	XRA B	A8
183.	POP H	E1	210.	RZ	C8	239.	XRA C	A9
184.	POP PSW	F1	211.	SBB A	9F	240.	XRA D	AA
185.	PUSH B	C5	212.	SBB B	98	241.	XRA E	AB
186.	PUSH D	D5	213.	SBB C	99	242.	XRA H	AC
187.	PUSH H	E5	214.	SBB D	9A	243.	XRA L	AD
188.	PUSH PSW	F5	215.	SBB E	9B	244.	XRA M	AE
189.	RAL	17	216.	SBB H	9C	245.	XRI Data	EE
190.	RAR	1F	217.	SBB L	9D	246.	XTHL	E3
191.	RC	D8	218.	SBB M	9E			

2) 8097 MICROCONTROLLER

MNEMONICS	OPCODE
ADD	44
AND	40
CMP	89
JBS	3B
JNE	D7
LD	A1
NOP	FD
NOT	02
OR	80
RET	F0
SCALL	28
SHR	08
SJMP	27
ST	C2
SUB	69
XOR	84
ORB	91

LDB B1[IMMEDIATE ADDRESSING]

LDB B0[DIRECT ADDRESSING]

STB C6[INDIRECT ADDRESSING]

STB C6[DIRECT ADDRESSING]

AD_ COMMAND 02

AD_RESULT_LO 02

AD_RESULT_HI 03

AX equ 50h

BX equ 52h

BL equ 52h

BH equ 53h

CL equ 53h

CH equ 55h

LX equ 58h

LL equ 58h

LH equ 5ah

3) 8051 MICROCONTROLLER

Hex Code	Number of Bytes	Mnemonic	Operands
00	1	NOP	
01	2	AJMP	code addr
02	3	LJMP	code addr
03	1	RR	A
04	1	INC	A
05	2	INC	data addr
06	1	INC	@R0
07	1	INC	@R1
08	1	INC	R0
09	1	INC	R1
0A	1	INC	R2
0B	1	INC	R3
0C	1	INC	R4
0D	1	INC	R5
0E	1	INC	R6
0F	1	INC	R7
10	3	JBC	bit addr,code addr
11	2	ACALL	code addr
12	3	LCALL	code addr
13	1	RRC	A
14	1	DEC	A
15	2	DEC	data addr
16	1	DEC	@R0
17	1	DEC	@R1
18	1	DEC	R0
19	1	DEC	R1
1A	1	DEC	R2
1B	1	DEC	R3
1C	1	DEC	R4
1D	1	DEC	R5
1E	1	DEC	R6
1F	1	DEC	R7
20	3	JB	bit addr,code addr
21	2	AJMP	code addr
22	1	RET	
23	1	RL	A
24	2	ADD	A,#data
25	2	ADD	A,data addr

Hex Code	Number of Bytes	Mnemonic	Operands
26	1	ADD	A,@R0
27	1	ADD	A,@R1
28	1	ADD	A,R0
29	1	ADD	A,R1
2A	1	ADD	A,R2
2B	1	ADD	A,R3
2C	1	ADD	A,R4
2D	1	ADD	A,R5
2E	1	ADD	A,R6
2F	1	ADD	A,R7
30	3	JNB	bit addr,code addr
31	2	ACALL	code addr
32	1	RETI	
33	1	RLC	A
34	2	ADDC	A,#data
35	2	ADDC	A,data addr
36	1	ADDC	A,@R0
37	1	ADDC	A,@R1
38	1	ADDC	A,R0
39	1	ADDC	A,R1
3A	1	ADDC	A,R2
3B	1	ADDC	A,R3
3C	1	ADDC	A,R4
3D	1	ADDC	A,R5
3E	1	ADDC	A,R6
3F	1	ADDC	A,R7
40	2	JC	code addr
41	2	AJMP	code addr
42	2	ORL	data addr,A
43	3	ORL	data addr,#data
44	2	ORL	A,#data
45	2	ORL	A,data addr
46	1	ORL	A,@R0
47	1	ORL	A,@R1
48	1	ORL	A,R0
49	1	ORL	A,R1
4A	1	ORL	A,R2

Hex Code	Number of Bytes	Mnemonic	Operands
4B	1	ORL	A,R3
4C	1	ORL	A,R4
4D	1	ORL	A,R5
4E	1	ORL	A,R6
4F	1	ORL	A,R7
50	2	JNC	code addr
51	2	ACALL	code addr
52	2	ANL	data addr,A
53	3	ANL	data addr,#data
54	2	ANL	A,#data
55	2	ANL	A,data addr
56	1	ANL	A,@R0
57	1	ANL	A,@R1
58	1	ANL	A,R0
59	1	ANL	A,R1
5A	1	ANL	A,R2
5B	1	ANL	A,R3
5C	1	ANL	A,R4
5D	1	ANL	A,R5
5E	1	ANL	A,R6
5F	1	ANL	A,R7
60	2	JZ	code addr
61	2	AJMP	code addr
62	2	XRL	data addr,A
63	3	XRL	data addr,#data
64	2	XRL	A,#data
65	2	XRL	A,data addr
66	1	XRL	A,@R0
67	1	XRL	A,@R1
68	1	XRL	A,R0
69	1	XRL	A,R1
6A	1	XRL	A,R2
6B	1	XRL	A,R3
6C	1	XRL	A,R4
6D	1	XRL	A,R5
6E	1	XRL	A,R6
6F	1	XRL	A,R7
70	2	JNZ	code addr

Hex Code	Number of Bytes	Mnemonic	Operands
71	2	ACALL	code addr
72	2	ORL	C,bit addr
73	1	JMP	@A+DPTR
74	2	MOV	A,#data
75	3	MOV	data addr,#data
76	2	MOV	@R0,#data
77	2	MOV	@R1,#data
78	2	MOV	R0,#data
79	2	MOV	R1,#data
7A	2	MOV	R2,#data
7B	2	MOV	R3,#data
7C	2	MOV	R4,#data
7D	2	MOV	R5,#data
7E	2	MOV	R6,#data
7F	2	MOV	R7,#data
80	2	SJMP	code addr
81	2	AJMP	code addr
82	2	ANL	C,bit addr
83	1	MOVC	A,@A+PC
84	1	DIV	AB
85	3	MOV	data addr,data addr
86	2	MOV	data addr,@R0
87	2	MOV	data addr,@R1
88	2	MOV	data addr,R0
89	2	MOV	data addr,R1
8A	2	MOV	data addr,R2
8B	2	MOV	data addr,R3
8C	2	MOV	data addr,R4
8D	2	MOV	data addr,R5
8E	2	MOV	data addr,R6
8F	2	MOV	data addr,R7
90	3	MOV	DPTR,#data
91	2	ACALL	code addr
92	2	MOV	bit addr,C
93	1	MOVC	A,@A+DPTR
94	2	SUBB	A,#data
95	2	SUBB	A,data addr
96	1	SUBB	A,@R0

Hex Code	Number of Bytes	Mnemonic	Operands
97	1	SUBB	A,@R1
98	1	SUBB	A,R0
99	1	SUBB	A,R1
9A	1	SUBB	A,R2
9B	1	SUBB	A,R3
9C	1	SUBB	A,R4
9D	1	SUBB	A,R5
9E	1	SUBB	A,R6
9F	1	SUBB	A,R7
A0	2	ORL	C,/bit addr
A1	2	AJMP	code addr
A2	2	MOV	C,bit addr
A3	1	INC	DPTR
A4	1	MUL	AB
A5		reserved	
A6	2	MOV	@R0,data addr
A7	2	MOV	@R1,data addr
A8	2	MOV	R0,data addr
A9	2	MOV	R1,data addr
AA	2	MOV	R2,data addr
AB	2	MOV	R3,data addr
AC	2	MOV	R4,data addr
AD	2	MOV	R5,data addr
AE	2	MOV	R6,data addr
AF	2	MOV	R7,data addr
B0	2	ANL	C,/bit addr
B1	2	ACALL	code addr
B2	2	CPL	bit addr
B3	1	CPL	C
B4	3	CJNE	A,#data,code addr
B5	3	CJNE	A,data addr,code addr
B6	3	CJNE	@R0,#data,code addr
B7	3	CJNE	@R1,#data,code addr
B8	3	CJNE	R0,#data,code addr
B9	3	CJNE	R1,#data,code addr
BA	3	CJNE	R2,#data,code addr
BB	3	CJNE	R3,#data,code addr
BC	3	CJNE	R4,#data,code addr

Hex Code	Number of Bytes	Mnemonic	Operands
BD	3	CJNE	R5,#data,code addr
BE	3	CJNE	R6,#data,code addr
BF	3	CJNE	R7,#data,code addr
C0	2	PUSH	data addr
C1	2	AJMP	code addr
C2	2	CLR	bit addr
C3	1	CLR	C
C4	1	SWAP	A
C5	2	XCH	A,data addr
C6	1	XCH	A,@R0
C7	1	XCH	A,@R1
C8	1	XCH	A,R0
C9	1	XCH	A,R1
CA	1	XCH	A,R2
CB	1	XCH	A,R3
CC	1	XCH	A,R4
CD	1	XCH	A,R5
CE	1	XCH	A,R6
CF	1	XCH	A,R7
D0	2	POP	data addr
D1	2	ACALL	code addr
D2	2	SETB	bit addr
D3	1	SETB	C
D4	1	DA	A
D5	3	DJNZ	data addr,code addr
D6	1	XCHD	A,@R0
D7	1	XCHD	A,@R1
D8	2	DJNZ	R0,code addr
D9	2	DJNZ	R1,code addr
DA	2	DJNZ	R2,code addr
DB	2	DJNZ	R3,code addr
DC	2	DJNZ	R4,code addr
DD	2	DJNZ	R5,code addr
DE	2	DJNZ	R6,code addr
DF	2	DJNZ	R7,code addr
E0	1	MOVX	A,@DPTR
E1	2	AJMP	code addr
E2	1	MOVX	A,@R0

Hex Code	Number of Bytes	Mnemonic	Operands
E3	1	MOVX	A,@R1
E4	1	CLR	A
E5	2	MOV	A,data addr
E6	1	MOV	A,@R0
E7	1	MOV	A,@R1
E8	1	MOV	A,R0
E9	1	MOV	A,R1
EA	1	MOV	A,R2
EB	1	MOV	A,R3
EC	1	MOV	A,R4
ED	1	MOV	A,R5
EE	1	MOV	A,R6
EF	1	MOV	A,R7
F0	1	MOVX	@DPTR,A
F1	2	ACALL	code addr
F2	1	MOVX	@R0,A
F3	1	MOVX	@R1,A
F4	1	CPL	A
F5	2	MOV	data addr,A
F6	1	MOV	@R0,A
F7	1	MOV	@R1,A
F8	1	MOV	R0,A
F9	1	MOV	R1,A
FA	1	MOV	R2,A
FB	1	MOV	R3,A
FC	1	MOV	R4,A
FD	1	MOV	R5,A
FE	1	MOV	R6,A
FF	1	MOV	R7,A