LAB 3

AIM: Further programming.To understand the concept of transfer of control within a program and its use.

The flexibility and versatility of computers rests on the ability to transfer program control to an instruction that is not in sequential order.
If some instructions are to be executed repeatedly then it is necessary to jump to the first instruction once those instructions have been executed and so on.Obviously the address of the first instruction is not sequential to the last one.and hence a branch must occur.

A branch maybe **conditional** or **non-conditional.**

Non-Conditional Branch: In this case program control is transferred to another location without any pre-conditions.This is done by loading the address of the next desired instruction into the program-counter.
eg.

JMP 20FE ;Transfers control of program to location 20FE.

JMP START: :Transfers control to symbolic address START.

ADDRESS	LABEL:	INSTRUCTION
....	....	....
2010	START:	....
....	....	....
....	....	JMP 20FE
....	....	....
....	....	....
20FE	....	....
....	....	....
....	....	JMP START:
....	....	....

**** A label is a symbolic address.

Conditional-Branch:
This transfer takes place only if a certain condition is satisfied else the program continues sequentially

There exists a flag-register in the microprocessor. This register is an indication of the state of the microprocessor.You can use this register for conditional branching.This results in the flag register being examined to see if the branch should be executed.

Flag-Register

S-Sign bit. Z-Zero. AC-Auxillary Carry. P-Parity. C-Carry

Some instructions which use the flag-register are:

OP-CODE	CONDITION	FLAG
JNZ	Not Zero	Z=0
JZ	Zero	Z=1
JNC	No Carry	C=0
JC	Carry	C=1
JPO	Parity Odd	P=0
JPE	Parity Even	P=1
JP	Plus	S=0
JM	Minus	S=1

***Be careful when using the flag register.Some instructions cause the flags to change! See programming manual of MAT 385 Vol 4.

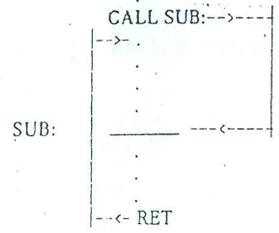
SUBROUTINES:
If it is necessary to execute a certain part of the code several times over at different locations in the program then subroutines should be used.This saves the task of writing the same code repeatedly at different locations and thus saves on memory.It also allows different sections of the code to be developed separately and eases debugging.Finally the different subroutines maybe collected together to form a good structured program.This is by far the most important of the branch control instructions.

In order for the sub-routine to be available to the program.it must be possible to transfer control from any desired point in the program to the beginning of the subroutine and once the subroutine is finished to return control to the next sequential instruction in the main program
The two instructions for inputting a subroutine are:

CALL XXXX ;where XXXX is a 4-bit hex address of the start of the subroutine.

and RET

The structure of the routine is as shown below.



Subroutines can also be nested.ie. A subroutine within a subroutine.
*** Since the program has to know where to return to after executing the subroutine it stores the value of the address next to the CALL instruction on the "stack".On encountering the RET instruction it loads the address into the program counter and thus returns to the instruction following the CALL.

EXCERCISES

EXPT 1:
Use single-step execution to see the action of the unconditional jump instruction below.

ADDRESS	INSTRUCTION
2800	JMP 2810
.	.
.	.
.	.
2810	JMP 2800

What do you think is changing?

EXPT 2:
Write a routine that inputs values 01.....10 into the memory locations 8020 to 8029.

EXPT 3:
Using subst-mem key fill locations 2000-2004 with five arbitrary values.Now write a program that compares the values in the five locations and puts the largest of them in register C.

Hint: Use Compare instructions.

***For queries look at Assignments 7-8 of MAT 385 Manual VOL1.

LAB 4

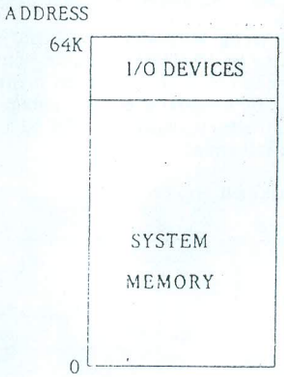
AIM: To realise input/output on the 8085 microprocessor -the most important feature from the users viewpoint.

A program resident in memory may require data input to process,and the user more often than not wishes to see the result of such an operation.Hence the need to be able to input and output data

Input/Output maybe **analog** or **digital** .For analog input/output the input has to be converted to digital form before it can be processed by the microprocessor(a digital device)and then the digital output has to be converted to analog form.No such problems exist for digital data.

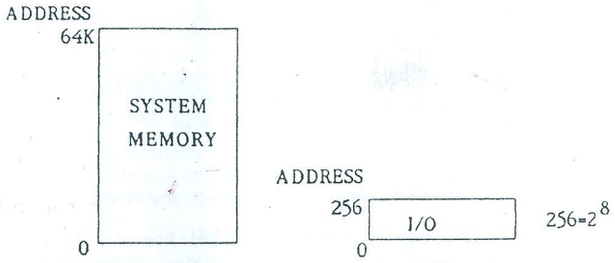
Input/Output may be implemented in one of two ways :-

Memory Mapped IO:
Here a part of the memory is used for input/output.Hence,each input/output operation is similar to a memory-access and all the instructions used for memory-access can be used for input/output. However this ties-up part of the systems memory (since it is set aside for I/O).



Programmed I/O
In this case input/output is achieved by special instructions executed by the processor("IN and OUT" for the 8085).
The processor outputs a request for I/O devices on the bus,indicating to them that the address on the address-bus is for an I/O device.In this way the system does not tie-up memory,and a larger space is available for programs.However this prevents us using the memory instructions.

The system memory is then as shown below.



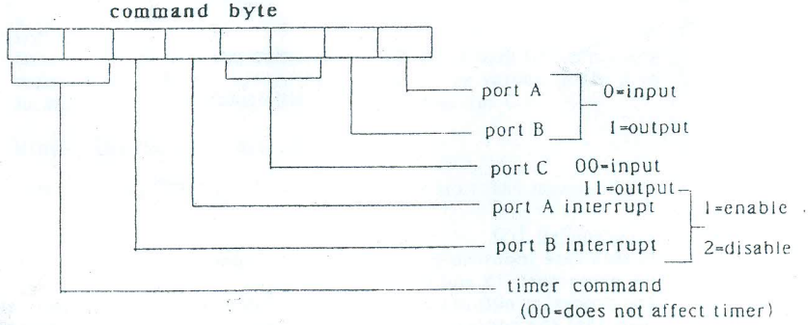
The 8085 uses the latter option -programmed I/O,although it is possible to configure it for memory-mapped I/O.

Port Initialisation:
I/O on the MAT is organised as several 8-bit ports.A maximum of 256 ports is possible (we don't need that many!).This assignment is concerned mainly with **two** of them.

PORT 21 : 8 Input switches on the front panel of the MAT.

PORT 22 : 8 "Led's" acting as outputs on the front panel.

These ports are **programmable** as "Input" or "Output" ports and are incorporated into an IO/TIMER memory chip(Intel 8155).
To configure these ports in a desired manner(ie. which shall be input and which shall be output),a 8-bit command is sent to the command register (addr 20)which resides in the 8155.Each of the 8-bits has a particular significance.



The command is transferred to 8155 using the "OUT" instruction as demonstrated below.

```
MVI  A,0E     ;0E= 0000 1110
OUT  20       ;Send to command register.
```

This sets port A to be 8-inputs,port B as 8-outputs,port C to be an output port and doesn't affect the timer.

Can you see how this happens?

Some common addresses:

ADDRESS(hex)	USE
20	command reg
21	port A data
22	port B data
23	port C Control

TIPS :
- The accumulator is always involved in IN and OUT instruction.
- To be able to single-step in an input/output program save the accumulator in location 20FF,since the monitor routine changes the value of the command byte when single-step is used.

EXCERCISES

EXPT 1:
Do exercise 9.1 of Assignment 9 in MAT 385 manual VOL 1.

EXPT2:
Write a program that reads the input from the switches , logically "OR's" it with 1010 1010(or any other value)and outputs the result to the LED's..Write it such that you can continuously change data using the switches.

EXPT 3:
Write a program that expects two 1-byte numbers from the switches,adds them and displays the result on the LED's.Write it such that you can continuously input data.
Extend the program to include a section so that what is shown on the LED's is also shown in Data -field of the display.

Hint: Use a monitor sub-routine UPDDT(address 036E) and take care to save the status of your processor before entering the sub-routine.

* * * For queries consult Assignment 9 of MAT 385.VOL 1.