BECE204L

Digital Assignment -2

MPMC

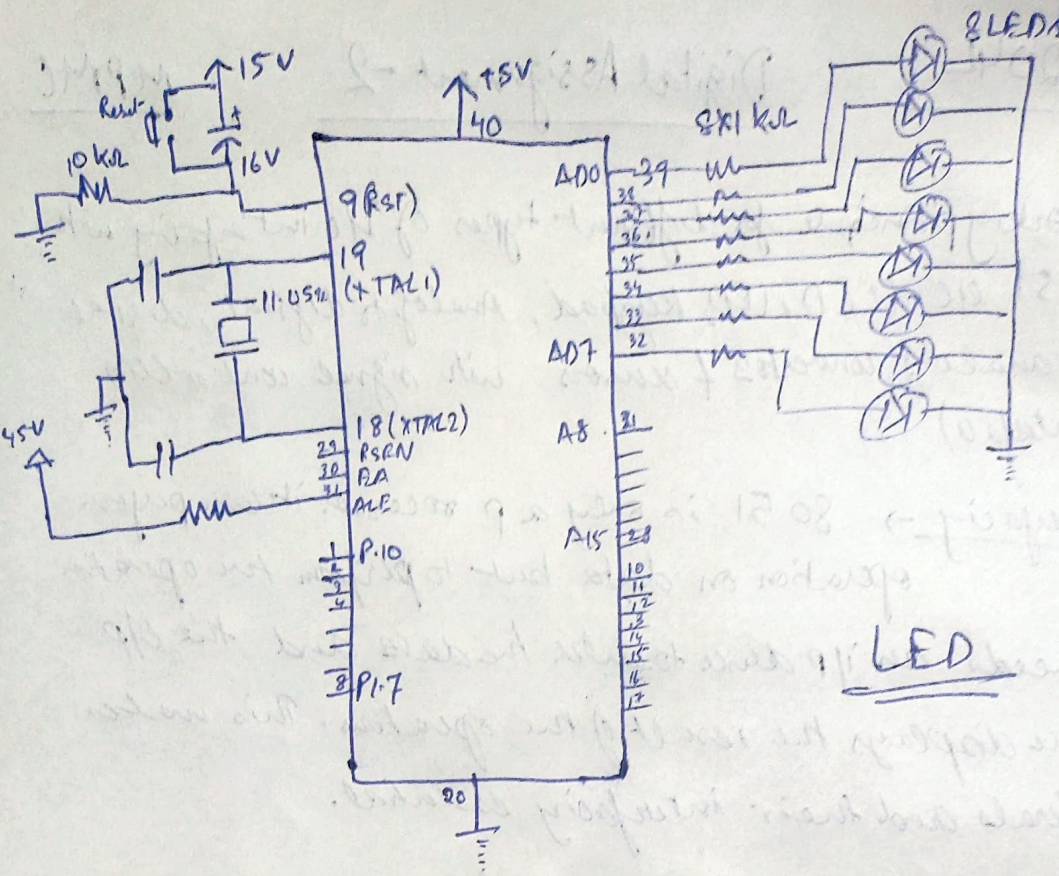
① Working principle for different types of I/O interfacing with 8051 e.g. (LCD, LED, keypad, Analog to digital, digital to analog converters & sensors with signal controlling interface)

Interfacing → 8051 is only a processor it can perform operation on data but to perform the operation it needs an i/p device to enter the data and the o/p device displays the result of the operations. This makes periferals and their interfacing essential.

① LED interfacing → used in mainly →

- indicating the output
- indicators to check the validity of results at diff. stages

They have a very simple principle of operation, its ON & OFF state expresses meaningful information about a device. The common available LED's have a 1.7V drop that means when we apply >1.7, the diode conducts (10mA req. for full int. LED can be interfaced either in common Anode or common cathode configuration. Common Anode is preffered i.e common cathode consumes more power.



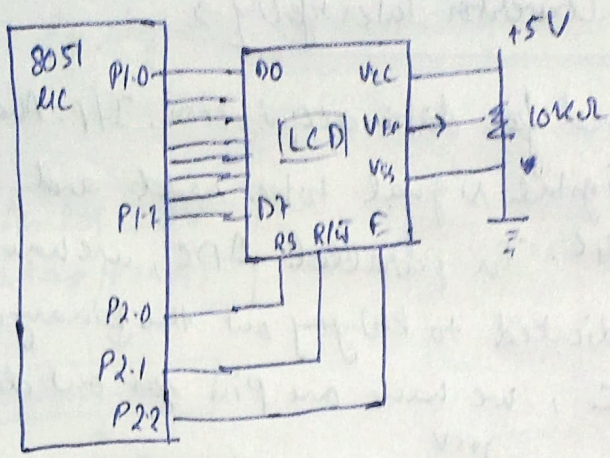
LED

LCD Interfacing → LCD is becoming popular and replacing LEDs due to its declining prices, It can display number, characters and graphics. It has a refreshing controller which relieves the task of refreshing the LCD.

LCD command codes →

1 - clear display screen
2 - return home
4 - Decrement cursor
6 - Increment cursor
5 - shift display rt.
7 - shift display left.
8 - Display off, cursor off
A - display off, cursor on
C - display ON, cursor OFF
E - display OFF, cursor blinking
F - display ON, cursor blinking
10 - shift cursor blinking
14 - shift cursor rt ; 18 - shift entire display left.

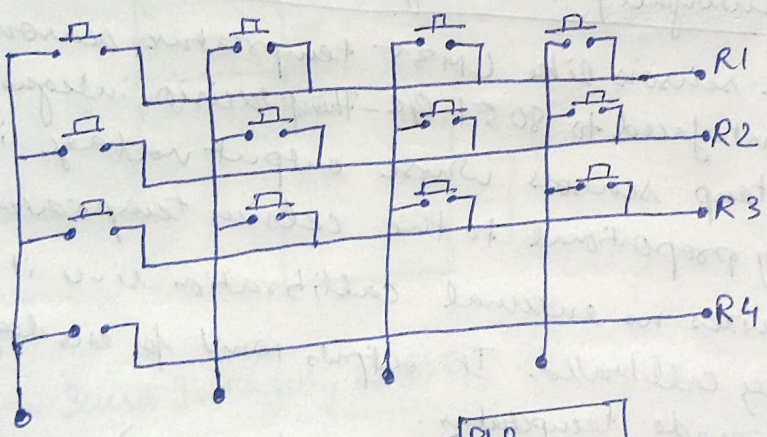
; 1C - shift entire display rt.
; C0 - cursor to second line
; 80 - cursor to 1st line
; 38 - 2 lines and 5×7 matrix.

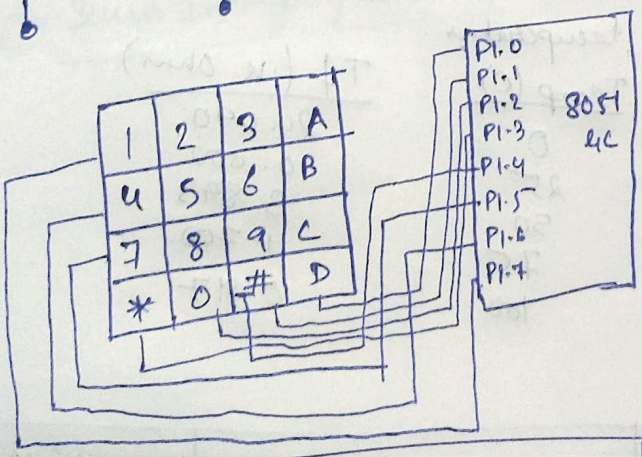


LCD interfacing

Keypad Interfacing (4x4) Matrix :→

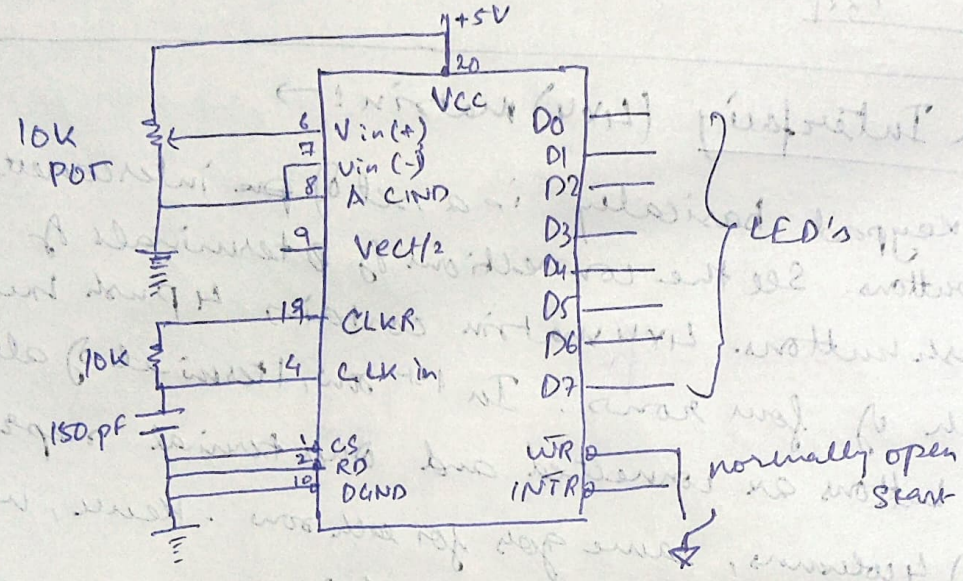
Matrix keypad basically is a set of interconnected push buttons. See the connections of terminals of the push buttons. 4x4 matrix contains 4 push buttons in each of four rows. In 1st row, 1 terminal of all 4 push buttons are connected and other terminal represents each of 4 columns, same goes for each row. Hence, we get 8 terminals to connect to µC.





Analog to Digital Converter Interfacing →

These are widely used for data acquisition. I/P = Analog signal and convert to digital signal to be read and processed by the µC. In parallel ADC, we have 8 more pins dedicated to bringing out the binary data but in serial ADC, we have one pin for out data.

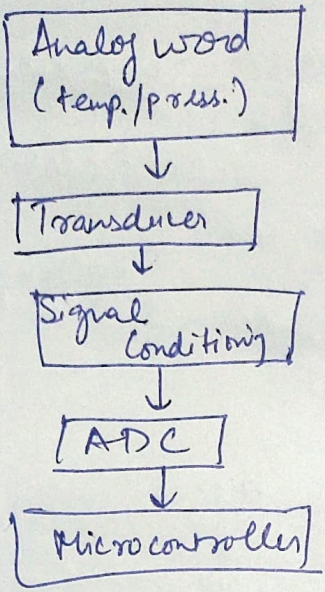


Sensor Interfacing and signal conditioning →

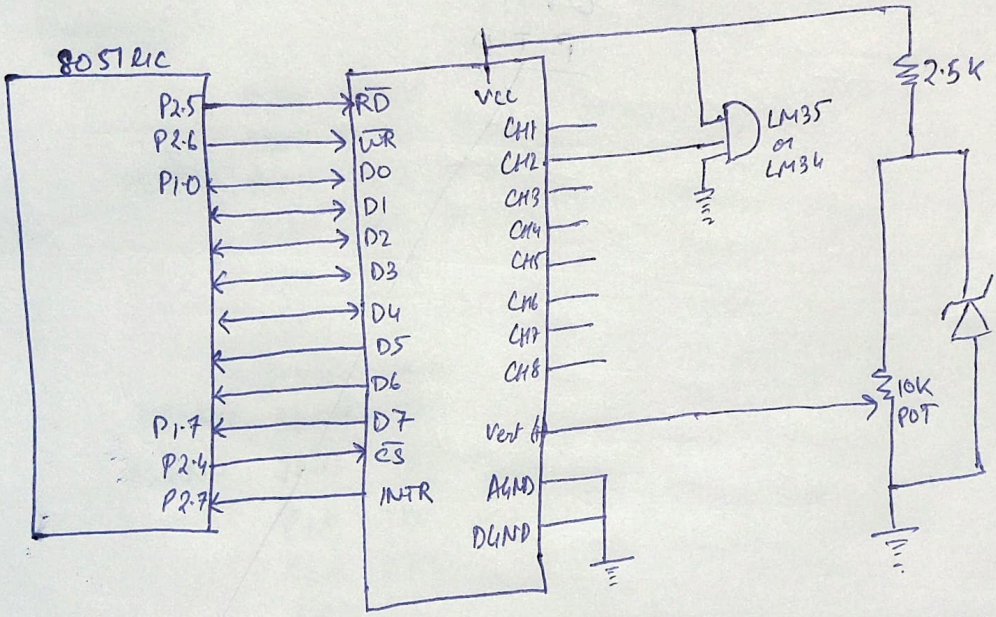
We have sensors like LM35 temprature sensors which can be interfaced to 8051 µC. These are precision integrated circuit temp sensors whose output voltage is linearly proportional to the celsius temprature. It requires no external callibration since it is internally callibrated. It outputs 10mV for each degree of centigrade temperature.

Temp (C)	Rt (K ohms)
0	24.490
25	10.000
50	3.893
75	1.700
100	0.817

Process of signal conditioning and interfacing with 8051 is done as follows:



Temp (C)	Vin (mV)	Vout (D7-D0)
0	0	0000 0000
1	10	0000 0001
2	20	0000 0010
3	30	0000 0011
10	100	0000 1010
30	300	0001 1110



Sensor Interfacing in 8051 µC.

Question 2) Write an 8051 ALP using timer 0 mode 1 to generate square wave on one of the pins of P1. The freq. of the wave to be generated →

Calculation → 21BCB0069

$= 2 \times 1 \times 11 \times 12 \times 11 \times 6 \times 9 = 157 KHZ$

$T = 1/f = 6.369 \mu s$

for ½ of a wave $= 3.18 s$

1MC = 1.085 μs

$count = \frac{3.18}{1.085} = 2.93 \simeq 3$ count

$$\begin{array}{r} FFFF \\ -\ 3 \\ \hline FFFC + 1 = FF \mid FD \end{array}$$

↓ THO ↓ TH1 value.

Program →

```
        ORG 0000H
        MOV TMOD, #01H
HERE:   MOV TL0, #0FDH
        MOV TH0, #0FFH
        CPL P1.0
        ACALL DELAY
        SJMP HERE
DELAY:  SETB TR0
AGAIN:  JNB TF0, AGAIN
        CLR TR0
        CLR TF0
        RET
        END
```

Question 3)

Write an 8051 ALP to transfer Reg no. & name as Data, serially at Baud rate 9600.

```
        ORG   0000H
H_1:    MOV   DPTR, #DATA1
        MOV   TMOD, #20H
        MOV   TH1, #-3
        MOV   SCON, #50H
        SETB  TR1
        MOV   R1, #18H
AGAIN:  CLR   A
        MOVC  A, @A+DPTR
        MOV   SBUF, A
HERE:   JNB   TI, HERE
        CLR   TI
        INC   DPTR
        DJNZ  R1, AGAIN
        SJMP  H_1
DATA1:  DB "21BCB0069 VISHAL AGRAWAL "
        END
```

Question 4) Write an 8051 ALP to transfer uni-y interrupts,

(a) Recieve starting letter of your name as input serially and send it to P0,

(b) Make timer 0 generate a square wave of 5kHz freq. on P3.2.

Assume XTAL - 11.0592, set the baud rate at 4800.

```
        ORG   0000H
        LJMP  MAIN

        ORG   000BH
        CPL   P3.2
        MOV   TL0, #0A4H
        MOV   TH0, #0FFH
        RETI
```

Calculation →

$f = 5kHz$,

$\frac{1}{f} = T = 200\,\mu s$

for 1/2 cycle, $100\,\mu s$

$count = \frac{100}{1.085} = 92$

$$\begin{array}{r} FFFF \\ -\ 5C \\ \hline FFA3 + 1 = \end{array} \begin{array}{c|c} TH0 & TL0 \\ FF & A4 \end{array}$$

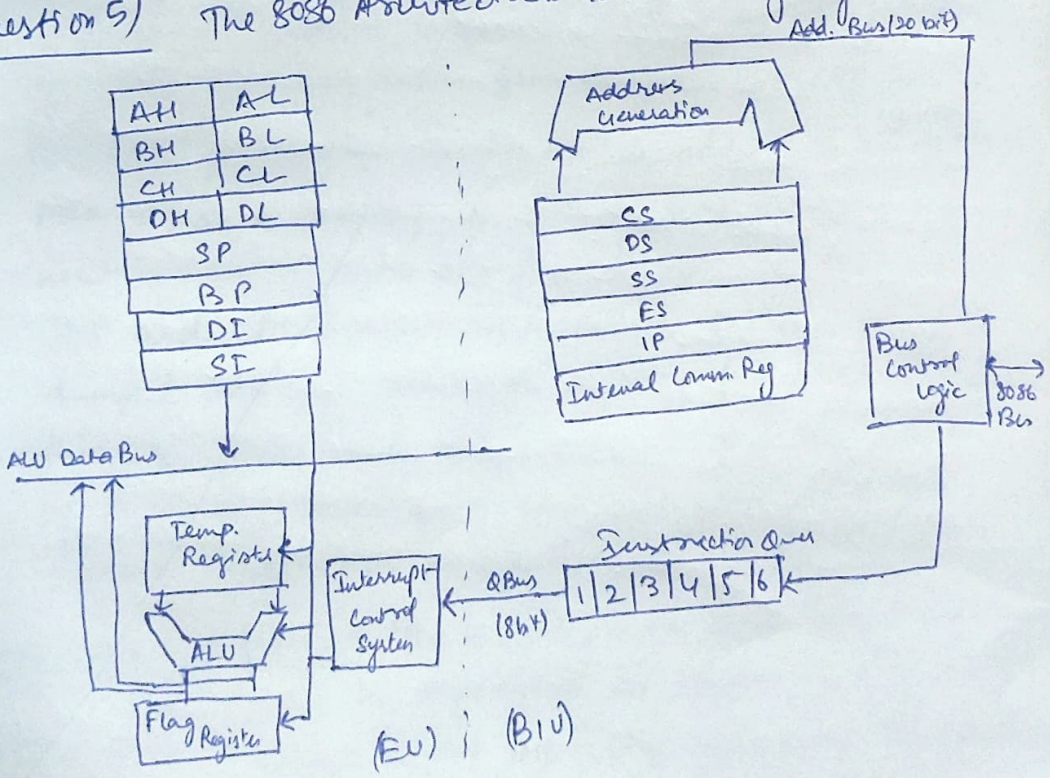
```
        ORG 0023H
        LJMP SERIAL

        ORG 0030H
MAIN:   MOV TMOD, #00100001B
        MOV TL0, #00H
        MOV TH0, #0FFH.
        MOV TH1, #-6
        MOV SCON, #50H
        MOV IE, #10010010B
        SETB TR0
        SETB TR1
HERE:   SJMP HERE

SERIAL: JB T1, TRANSMIT
        MOV A, SBUF
        MOV P0, A
        CLR RI
        RETI
TRANSMIT: CLR TI
        RETI

        END
```

Question 5) The 8086 Architecture and memory segmentation.



Execution (Unit) EU → This executes instructions that have already been fetched by the BIU. The EU tells the BIU where to fetch instructions or data from, decodes instructions, and executes instructions. The main components are General purpose registers, the ALU, special purpose registers, the IR's and instruction decoder and the flag (status register).

Bus Interface Unit (BIU) → BIU sends out addresses, fetches instruction from memory, reads data from ports and memory and writes data to ports and memory. It handles all transfers of data and addresses on the buses for the execution unit.

Pipelining Architecture → While the EU is decoding an instruction or executing an instruction, which does not require the use of the buses, the BIU fetches up to six instruction bytes for the following instructions. The BIU stores the pre-fetched bytes in a first-in-first-out register set called a queue. When the EU is ready for its next instruction from the queue in the BIU. This is much faster than sending out an address to the system memory and waiting for memory to send back the next instruction byte or bytes. Except in the case of JMP and CALL instructions, where the queue must be dumped and then reloaded starting from a new address, this pre-fetch and queue scheme greatly speeds up processing. Fetching the next instruction while the current instruction executes is called pipelining.

Memory Segmentation → The memory in an 8086 based system is organized as segmented memory. The CPU 8086 is able to access 1MB of physical memory. The complete

1MB of memory can be divided into 16 segments, each of 64 KB size and is addressed by one of the segment registers. The 16 bit contents of the segment registers actually point to the starting location of a particular segment. The address of the segments may be assigned as 0000H to F000H respectively. To address a specific memory location within a segment, we need an offset address. The offset address values are from 0000H to FFFFH so that the physical address range from 00000H to FFFFFH.

<u>Advantages</u> → ↑

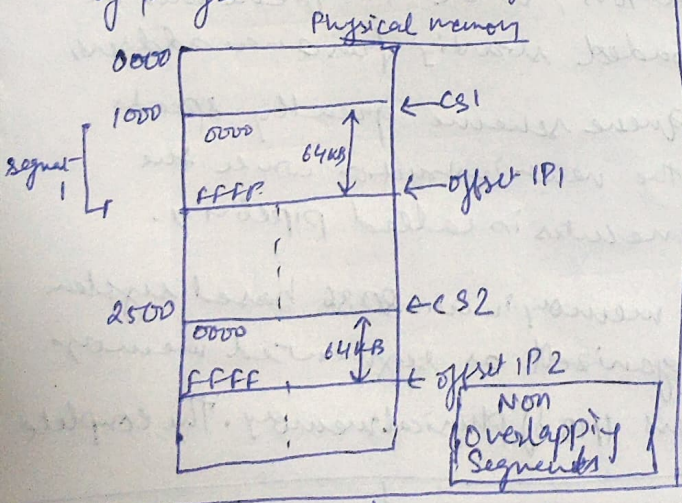
① Allows the memory capacity to be 1MB although the actual addresses to be handled are of 16 bit size.

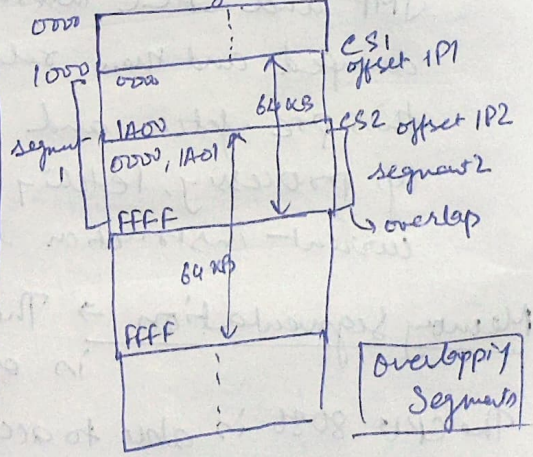
② Allows the placing of code, data and stack portions of the same program in different parts (segments) of memory, for data and code protection.

③ Permits a program and/or its data to be put into different areas of memory each time the program is executed, i.e. provision for relocation is done.

<u>Overlapping and non-overlapping memory segments</u> →

In the overlapping area locations physical address = CS1 + IP1 = CS2 + IP2, where '+' indicates the procedure of physical address formation.





Question 6) Interface 4x4 hex keyboards and write a program to read the key closure and display hex code for key pressed on 7segment display.

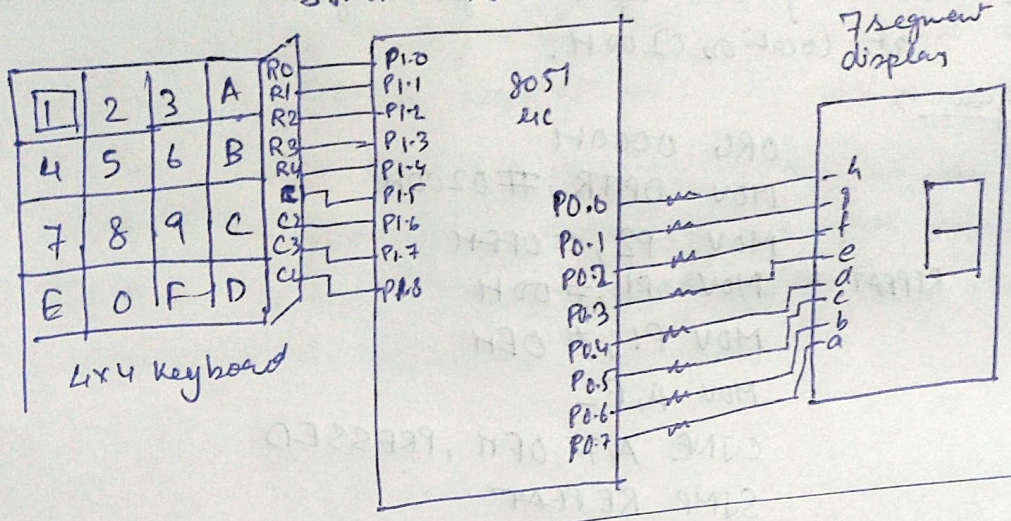
Sol^n) Assuming look up table for 7segment code is located at location 0200H.

Program →

```
        ORG 0000H
        MOV DPTR, #0200H
        MOV P2, #0FFH
REPEAT: MOV P1, #00H
        MOV P1, #0EH
        MOV A, P2
        CJNE A, #0FH, PRESSED
        SJMP REPEAT
PRESSED: ACALL DELAY
        MOV R0, #00H
        MOV P1, #0EH
        MOV A, P2
        CJNE A, #0FH, CHECK
        MOV R0, #04H
        MOV P1, #0DH
        MOV A, P2
        CJNE A, #0FH, CHECK
        MOV R0, #08H
        MOV P1, #0DH
        MOV A, P2
        CJNE A, #0FH, CHECK
        MOV R0, #0CH
        MOV P1, #0DH
        MOV A, P2
        CJNE A, #0FH, CHECK
CHECK:  RRC A
        JNC NOT
        INC R0
        SJMP DETECT
```

~~NOT:~~

```
GOT : MOV A, R0
      MOVC A, @A+DPTR
      MOV P0, A
      ACALL DELAY
      SJMP REPEAT.
```



Ques7) Draw the ARM processor architecture and explain its working principle.

Sol:) ARM: Advanced RISC machine

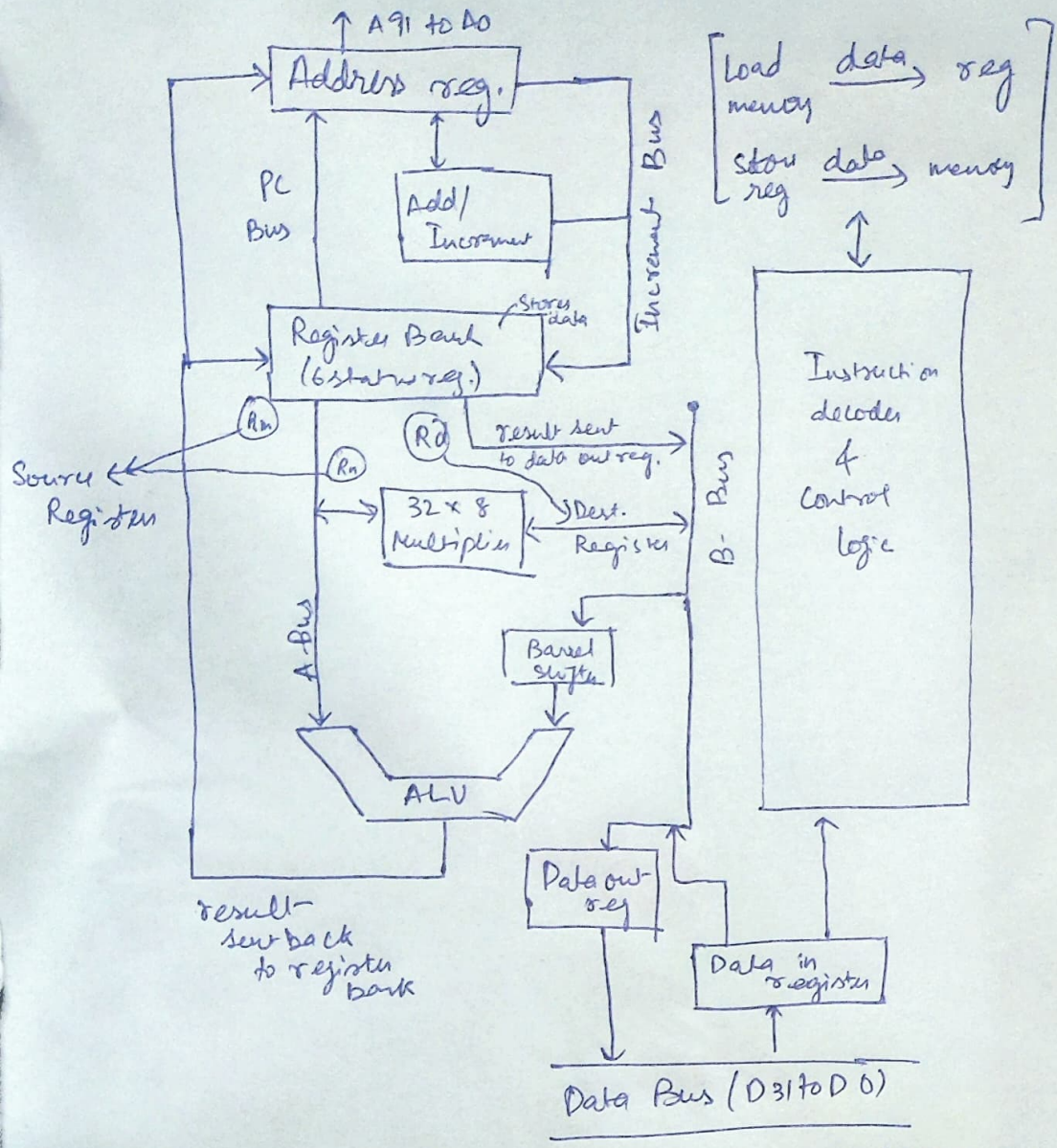
This is ~~one~~ used at many places and is most-licensed processor in the world. It is based on Berkeley RISC machine, ARM architecture is a fixed length instruction (32-bit), supports the concept of pipelining and has a load & store architecture type.

A byte has 8 bits in ARM type processors, word means 32 bits and half word means 16 bits.

Mainly ARMs implement 2 types of instruction set architecture
- 32 bit ARM instruction set
- 16-bit THUMB instruction set.

It is said to be a load/store architecture because every operation (fetching data, decoding or executing) is done with the help of registers (32-bit) only.

ARM architecture and data flow →



- Data items are placed in register file, no data processing instructions directly manipulate data in memory.
- Instructions typically use 2 source registers and single result or destination register.
- A Barrel shifter on the data path can pre-process data before it enters ALU.
- Increment / Decrement logic can update register content for sequential access independent of ALU.
- In load/store architecture, the addressing mode for every operand is fixed.