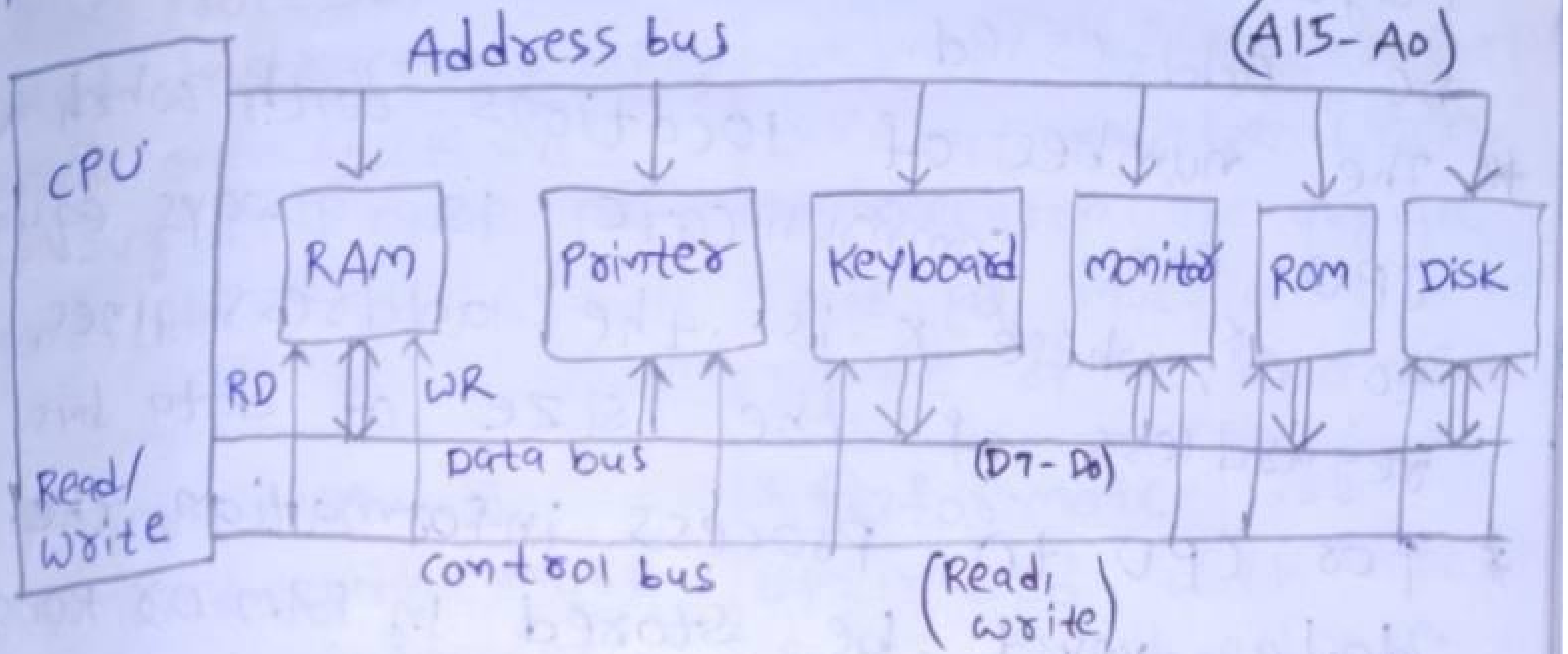
Assignment-1

Harshavardhan Alimi

18EC10021

1. A microcomputer is an electronic device with a microprocessor as its central processing unit.

Microcomputer Architecture



Address bus:-

- For a device (memory or I/O) to be recognised by CPU, it must be assigned an address
  - The address assigned to a given device must be unique.
  - The CPU puts the address on address bus, & the decoding circuitry finds the device
  - it is a unidirectional outgoing bus from CPU.

Data bus:-

- The CPU either gets data from device or sends data to it.

Control bus:-

- Provides read or write signals to the device to indicate if the CPU is asking for information or sending it information.

* The more data buses available, the better the CPU performs.
* Processing power of a computer is related to the size of its buses.
* The more address buses available, the larger the number of devices that can be addressed.
* The number of locations with which a CPU can communicate is always equal to 2^x, where x is the address lines, regardless of the size of data bus.
* For CPU to process information, the data must be stored in RAM or ROM which are refered as primary memory.
* RAM provides information that is fixed and permanent.
* RAM stores information that is not permanent and can change with time.
* The CPU gets information to be processed first from RAM (or ROM) and if it is not there, then seeks it from a mass storage device, called secondary memory & transfers information to RAM.

Scanned by TapScanner

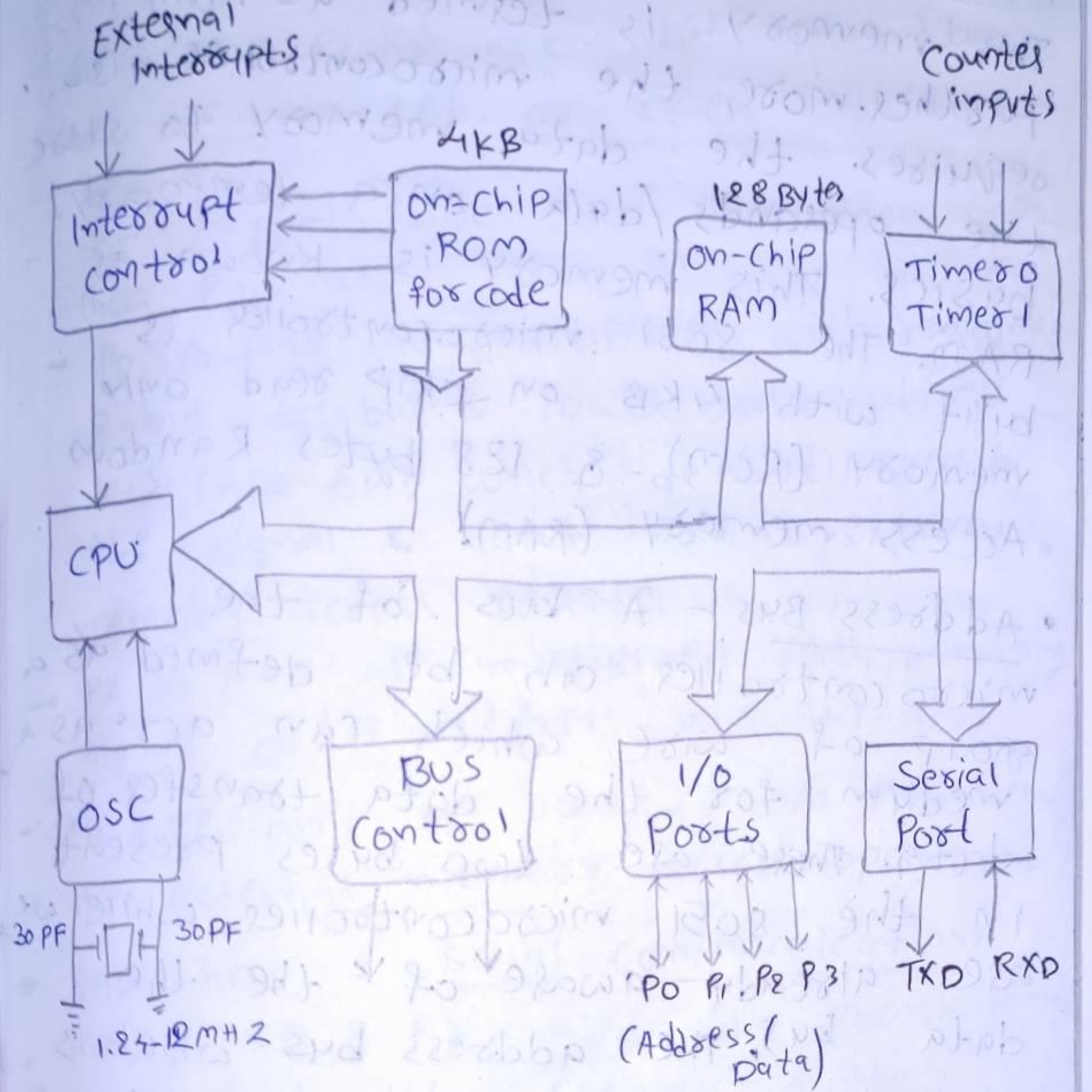
2.

RISC	CISC
1. RISC stands for Reduced instruction set computer.	1. CISC stands for complex instruction set computer.
2. RISC Processors have simple instructions taking about one clock cycle. The avg clk cycle per instruction is 1.5	2. CISC Processor have complex instructions that take up multiple clocks for execution. The avg clk cycle per instruction is in range of 2 and 15.
3. Performance is optimised with more focus on Software.	3. Performatice is optimised with more focus on hardware.
4. It has no memory unit and uses a sep hardware to implement instructions.	4. It has memory unit to implement complex instructions.
5. It has a hard-wired unit of programming.	5. It has a microprogramming unit.
6. The instruction set is reduced i.e. it has only a few instructions in the instruction set. many of these instructions are very Primitive.	6. The instruction set has a variety of diff instructions that can be used for complex operations.
7. The instruction set has a variety of diff instructions that can be used for complex operations.	7. CISC has many diff addressing modes & can thus be used to represent higher-level programming language statements more efficiently.

RISC	CISC
8. Complex addressing modes are synthesized using software	8. CISC already supports complex addressing modes.
9. Multiple register sets are present	9. Only has a single register set.
10. RISC processors are highly pipelined.	10. They are normally not pipelined or less pipelined.
11. The complexity of RISC lies with the compiler that executes the program.	11. The complexity lies in the microprogram
12. Execution time is very less	12. Execution time is very high.
13. code expansion can be a problem.	13. code expansion is not a problem.
14. Decoding of instructions is simple.	14. Decoding of instructions is complex.
15. It does not require external memory for calculations	15. It requires external memory for calculations.

Scanned by TapScanner

3) Internal Block-diagram of 89S51 microcontroller.



• <u>8-bit microcontroller</u>:- The 8951 micro-controller is an 8-bit micrcontroller This signifies that the width of the data-bus is 8-bits. The data bus is utilized to carry data from specific operations. consequently, the CPU can process 8-bits of data at one time.

• Memory :- A microcontroller needs program memory to store program/instructions to perform defined tasks. This memory is termed as ROM. Further-more, the microcontroller also requires the data memory to store the operands/data on a temporary basics. This memory is known as RAM. The 8051 microcontroller is built with 4KB on chip read only memory (ROM) & 128 bytes Random Access memory (RAM)

• Address Bus :- A bus of the microcontroller can be defined as a group of wire which can act as a medium for the data transfer of data. There are two buses present in the 8051 microcontroller. while we are already aware of the the data bus, The address bus which is used to address memory locations, is 16-bit wide, further more, the address bus can also be used to transfer data from CPU to memory. Hence, for obvious reasons the address bus is unidirectional.

Central processing unit :- (CPU)

* it is the heart of microcontroller that mainly comprises of an Arithmetic logic unit (ALU) and a control unit (CU) and other important components. The CPU is the primary device in communicating with peripheral devices like memory, input & output.

Arithmetic logic unit, as the name suggests, performs the Arithmetical & logical operations. CU or control unit is responsible for timing of the communication process between the CPU and its peripherals.

Program memory :-

The instructions of the CPU are stored in the program memory. it is usually implemented as Read only memory or ROM, where the program ~~memory~~ written in to it will be retained even when the power is down or the system is reset.

Modern program memory modules are generally made up of EEPROM (Electrically erasable programmable read-only memory), which is a type of non-volatile memory.

In this type of memory, the data

can be erased and reprogrammed using special programming signals.

when the microcontroller is powered on or manually reset, the processor executes a set of instructions from a pre-defined memory location (address) in the program memory.

Data Memory :-

Data memory is a microcontroller is responsible for storing values of variables, temporary data, intermediate results and other data for proper operation of the program.

Data memory is often called as RAM (Random access memory), which is a type of volatile memory. It is generally organised as registers & includes both special function registers (SFR's) & user accessible memory Locations.

• Input / output ports :-

- 8951 has four input / output port P0, P1, P2, P3
- Each port is 8 bit wide & their SFR (P0, P1, P2, P3) are bit accessable. i.e., we can set or reset individual bit.
- Some ports have dual functionality on their pins as,
  - P0 I/O pins are multiplexed with remaining 8-bit databus & lower order address bus (AD0-AD7) which demultiplexed by ALE signal & latch used in external memory acess operatn.
  - P2 I/O pins are multiplexed with remaining higher order address bus (A8-A15)
- P0 & P2 can't be used as I/O pins in the external memory access operation.

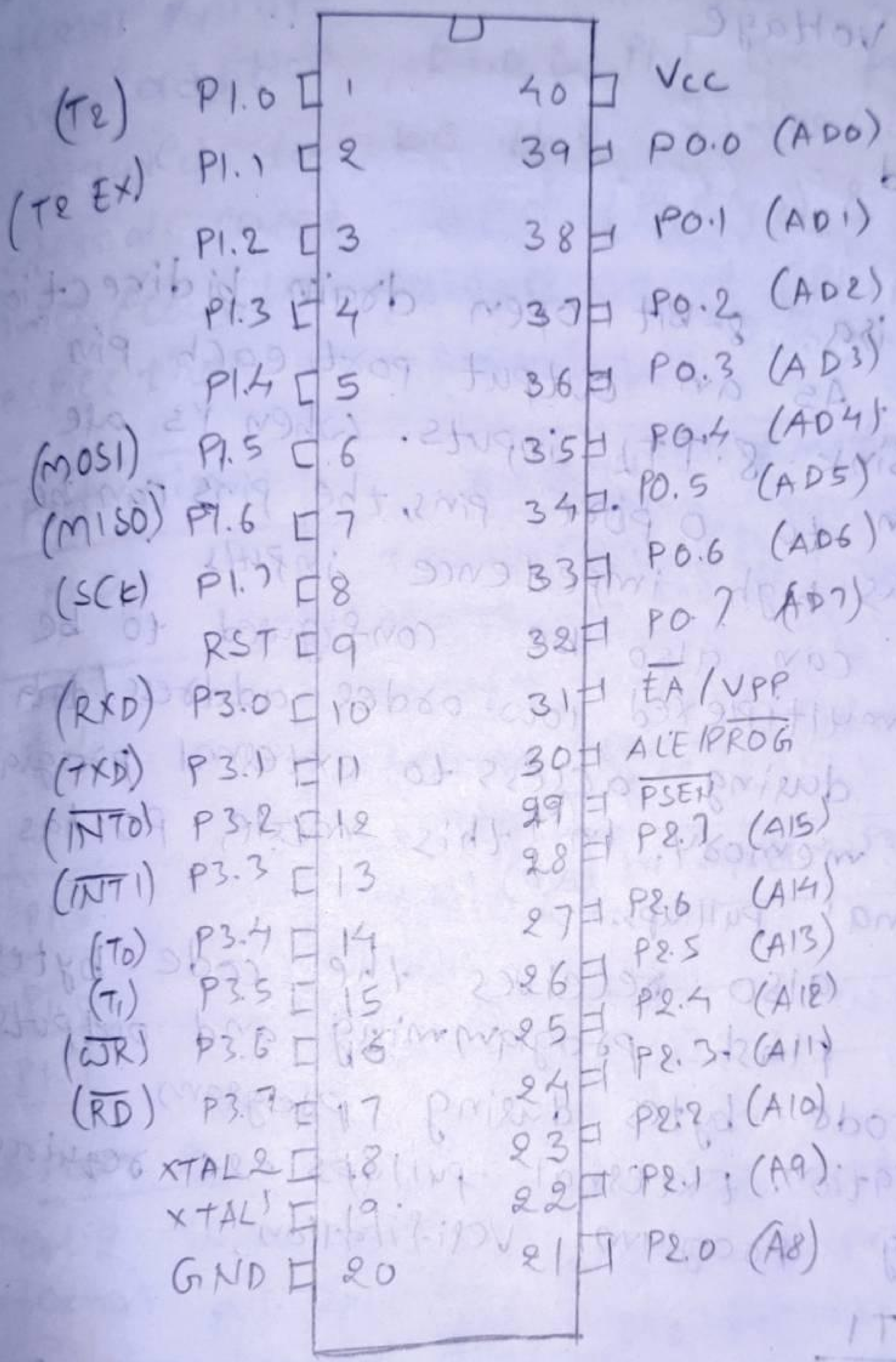
• 8951 has 2 serial communication pins TXD & RXD used for transmitting & received data serially via the SBUF register, SCON SFR used to control serial operation.

• Oscillator :-
- it is used to provide a clock to the 89S51 which decides the speed & baud rate, we use crystals of freq varying from 4 to 30 MHz. Normally we use 11.0592 MHZ which is required for 9600 baud rate in serial communication

- Interrupts :-
- interrupts are requested by internal or external peripherals which are masked while unused.
- Interrupt handler routines are called after each interrupts event occurs.
- These routines are called an interrupt service routine and are located in special memory loc.
- INT0 & INT1 pins used to accept external interrupts.

- Timers & counters :-
- 8951 has 2 timer pins T0 & T1.
- By these timers, we can generate a delay of a particular time in Timer mode.
- We can count external pulses are available as T0 (TH0 & TL0) & T1 (TH1 & TL1)
  i.e., Higher 8-bits in TH0/TH1
  Lower 8-bits in TL0/TL1
- TMOD & TCON registers are used to select mode & control the timer operation.

4). Pin configuration of 89S52 microcontroller.



40-Lead PDIP

Pin Description :-

VCC
supply voltage

GND
GRound

PORTO
PORTO isa 8-bit open drain bidirectiona lb port. As an output port, each pin can sink 8.TTL inputs. when 1's are written to 0 port pins, the pins can be used as high impedence inputs
PORT 0 can also be configured to be the multiplexed low-order address/data bus during access to external program data memory. In this mode, P0 has internal pullups.
PORTO also receives the code bytes during Flash programming and outputs the code bytes during program verification. External pullups are required during program verification.

~~P0~~ PORT1
PORT1 is a 8-bit bidirectional I/O Port with internal pullups. The port1 output buffers can sink/source 4 TTL inputs when 1's are written to port1 pins, they are pulled high by the internal pull-ups and can be used as inputs.

Scanned by TapScanner

As inputs, Port 1 pins that are externally being used as ~~in~~ pulled low will source current (I_{IL}) because of internal pull ups.

In addition, P1.0 & P1.1 can be configured to be the timer/counter 2 external count input (P1.0/T2) & the timer/counter trigger input (P1.1/T2 Ex), respectively as shown in following table.

port pin	Alternate functions.
P1.0	T2 (external count input to Timer/counter 2) clock-out.
P1.1	T2Ex (Timer/counter 2 capture/reload trigger & direction control).
P1.5	MOSI (used in system-programming)
P1.6	MISO " "
P1.7	SCK " "

~~P2~~ PORT 2

Port 2 is a Bidirectional I/O port with internal pull-ups. The Port 2 output buffers can sink/source four TTL inputs. When 1's are written to port 2 pins, they are pulled high by internal pullups & can be used as inputs. As inputs, Port 2 pins that are extremely being pulled low will source current (I_{IL}) because of internal pull-ups.

Scanned by TapScanner

Port 2 emits the high-order address byte during fetches from external Program memory & during accesses to external data memory that use 16-bi addresses (MOVX @ DPTR). In this applicatⁿ, Port 2 uses strong internal pullups when emitting 1's. During accesses to external data memory that use 8-bit addresses (MOVX @ R1), Port 2 emits the contents of the P2 Specia function Register.

Port 2 also receives the high-orde address bits & some control signals during Flash programming & verificatⁿ.

<u>Port-3:-</u>

Port 3 is a 8-bit bi-directional I/O Port with internal pull-ups. The Port3 output buffers can sink (source 4 TTL inputs. When 1's are written to Port3 pins, they are pulled high by the intern pullups & can be used as inputs. As inputs, Port 3 pins that are externally being pulled low will source current (I_{IL}) because of pullups

Ports 3 also serves the functions of various special features of 8952 as shown in following table.

Port Pin	Alternate function
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	$\overline{INT0}$ (external interrupt 0)
P3.3	$\overline{INT1}$ (external interrupt 1)
P3.4	T0 (timer 0 external input)
P3.5	T1 (timer 1 external input)
P3.6	$\overline{WR}$ (external data memory write strobe)
P3.7	$\overline{RD}$ (external data memory Read strobe)

RST:
Reset input, A high on this pin for 2 machine cycles while the oscillator is running resets the device. This pin drives High for 96 oscillator periods after the Watchdog times out. The DISRTO bit in SFR AUXR (address 8EH) can be used to disable this feature. In the default state of bit DISRTO, the RESET HIGH out feature is enabled.

.ALE / $\overline{PROG}$:-
Address latch enable (ALE) is an output pulse for Latching the low byte of address during accesses to external memory. This Pin is also the program pulse input ($\overline{PROG}$) during Flash Programming.

In normal operation, ALE is emitted at a constant rate of 1/6th oscillator frequency & may be used for external timing or clocking purposes. Note, however that one ALE pulse is skipped during each access to external data memory.

If desired, ALE operation can be disabled by setting bit 0 of SFR location 8EH. with the bit set, ALE is active only during a movx or movc instruct^n. otherwise the pin is weakly pulled high. setting the ALE-disable bit has no effect if the microcontroller is in external execution mode.

$\overline{\text{PSEN}}$:-

program strobe enable is the read strobe to external program memory.

when 8952 is executing code from external program memory, $\overline{\text{PSEN}}$ is activated twice each machine cycle, except that 2 $\overline{\text{PSEN}}$ activations are skipped during each access to external data memory.

$\overline{EA}$/VPP :-

·External Access enable ($\overline{EA}$) must be strapped to GND in order to enable the device to fetch code from external program memory locations starting at 0000H up to FFFFH.

·However, that if lock bit 1 is programmed, $\overline{EA}$ will be internally latched on reset.

$\overline{EA}$ should be strapped to Vcc for internal program executions.

This pin also receives the 12-volt programming enable voltage (VPP) during Flash programming.

XTAL1 :-

Input to the inverting oscillator amplifier & input to the internal clock operating circuit.

XTAL2 :-

Output from the inverting oscillator amplifier.

5) let, for transmitting 8051

$f_{XTAL} = 7.3728$ MHZ .

$\therefore f_{mclk} = f_{XTAL}/12 = 614.4$ KHZ

$f_{TCLK} = f_{mclk}/32 = 19200$ HZ

let, for ~~trar~~ recieving 8051

$f_{XTAL} = 16.5888$ MHZ

$\therefore f_{mclk} = f_{XTAL}/12 = 1.3824$ KHZ

$f_{TCLK} = f_{mclk}/32 = 43200$ HZ

.let the baud rate = 4800 bps

$\therefore$ For transmitting 8051

$$TH1 = 256 - \frac{19200}{4800} = 252 \text{ (or -4)}$$

$\therefore$ For receving 8051

$$TH1 = 256 - \frac{43200}{4800} = 247 \text{ (or -9)}$$

~~As~~ Assembly Language for transmitting 8051

```
ORG 0000H
START: MOV TMOD,#20H       ; Timer1, mode2 (auto-reload)
       MOV TH1,#252        ; For baud rate-4800
       MOV SCON,#50H       ; mode 1 (8-bit, 1 stop) REN
                             enable

       MOV DPTR,#4DDDH     ; load pointer for message
       SETB TR1            ; start Timer 1
                           ;
NEXT:  CLR A               ; get the character.
       MOVC A,@A+DPTR
```

```
        JZ OVER          ; if last Character getard
        ACALL SEND       ; call transfer
        INC DPTR .       ; next one
        SJMP NEXT        ; stay in loop
OVER:   SJMP OVER        ; stay here when finishes
SEND:   MOV SBUF, A      ; Load the data
WAIT :  JNB TI, WAIT     ; stay here until last
                                            bit gon
        CLR TI           ; get ready for next chy
        RET              ; return to caller
                         ; END assembly
        END
```

Assembly Language for receives

```
ORG 0000H
START: MOV TMOD, #20H ; Timer1, mode 2 (auto reload)
       MOV TH1, #247   ; 4800 baudrate
       MOV SCON, #50H  ; mode 1 (8-bit, 1stop). REN
                                            enable)
       SETB TR1        ; start Timer1
RECV : JNB RI, RECV    ; stay here until Last bit
                                            receive
       MOV A, SBUF     ; trasfer data to accumulator
       CLR RI          ; get ready for nxt char
       SJMP RECV       ; stay in loop
                       ; END assembly
       END
```

6)

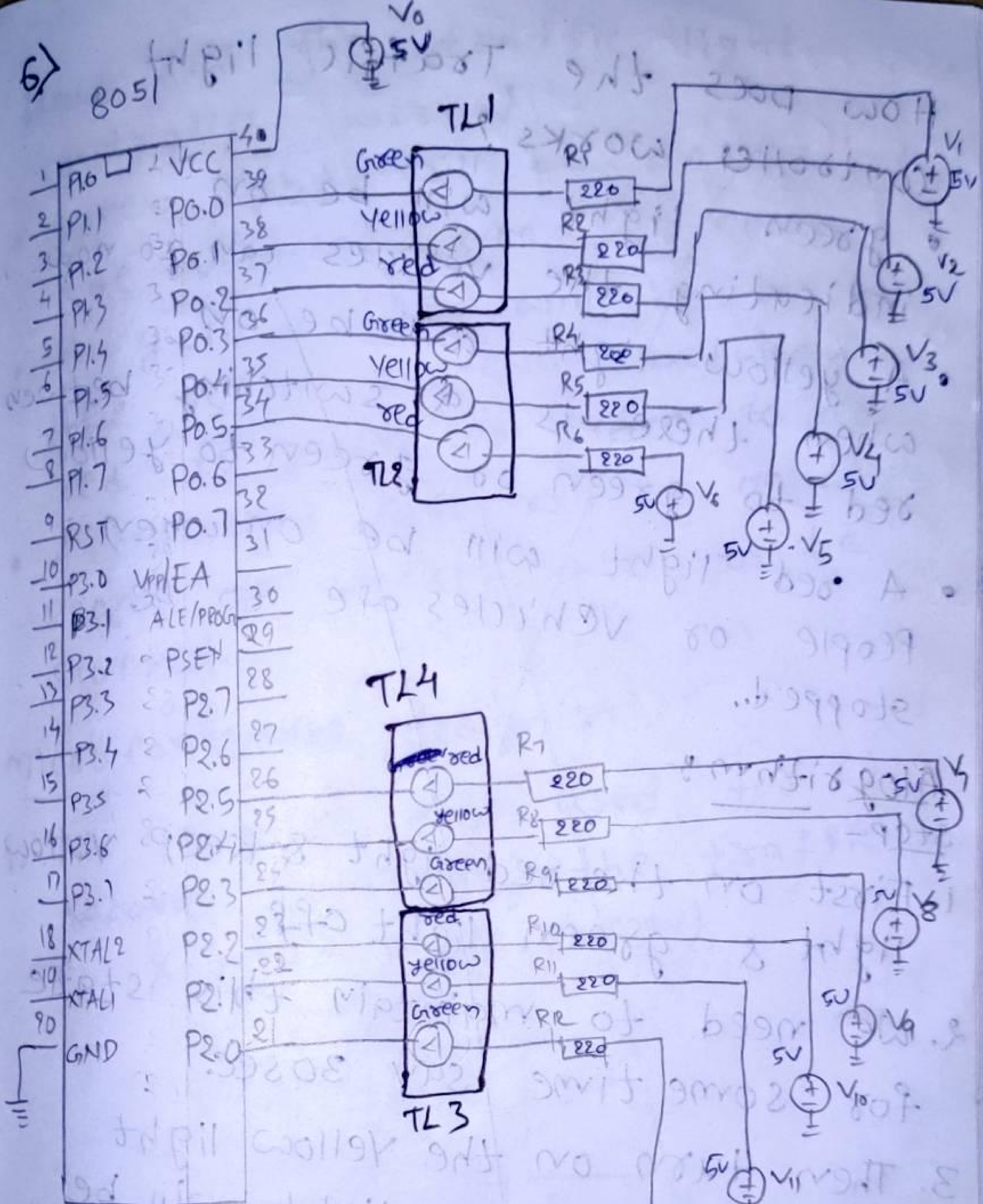


Fig:- circuit diagram.

TLi :- Traffic light light for Lane i

Green :- Green LED

Red :- Red LED

Yellow :- yellow LED

□ :- Resistor

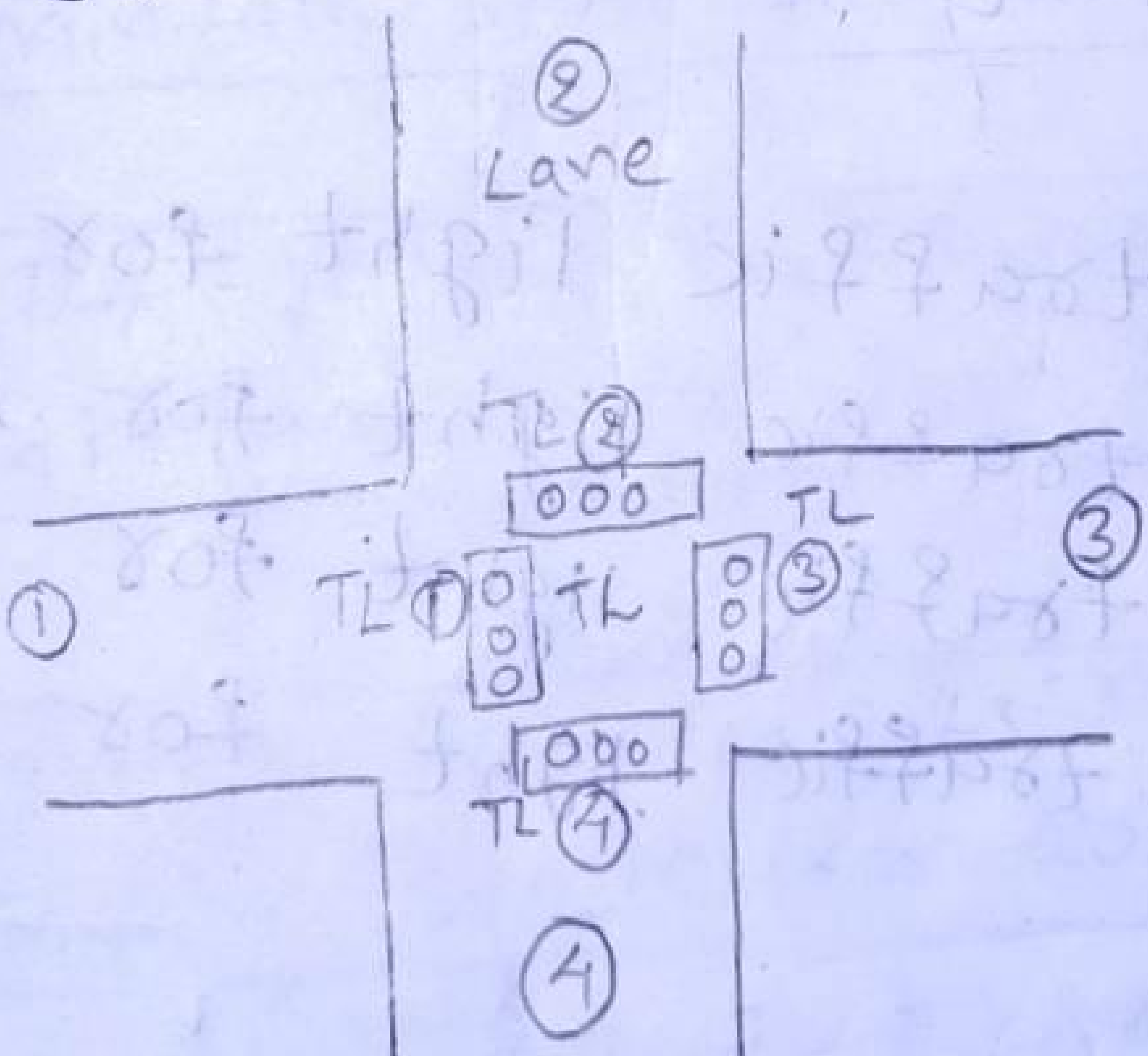
4-way Traffic light system.

How does the traffic light controller works?

- green light will be on indicating the vehicles can go now
- A yellow light will be on when there is a switching between red to green or green to red.
- A red light light will be on when people or vehicles are to be stopped.

micro controller design :-

1. This microller ~~A~~ is used for ~~XXXane~~ controlling the traffic in X-junction (i.e., 4 way)



2. This microcontroller is designed in such a way that it will allow the vehicles to go ~~fro~~many one of the lane and stopping the

Scanned by TapScanner

vehicles in other 3 Lanes

•> Here the stop period is 64 sec

•> Ready to go/stop (yellow light) Period in TL1 is 2 sec & 2 sec.

•> To go (green light) period in TL1 is 20 sec.

Above 3 time periods is applicable for every lane.

→ i.e., For ex:- in Lane 1 it will be green for first 20 sec & then become ready to stop for 2 sec and gets stopped for 64 sec, after then ready to go for 2 sec and after then repeats.

i.e., total time cycle is <u>88 sec</u>

TL_1 :- traffic light for Lane 1

TL_2 :- traffic light for Lane 2

TL_3 :- traffic light for Lane 3

TL_4 :- traffic light for Lane 4

Scanned by TapScanner

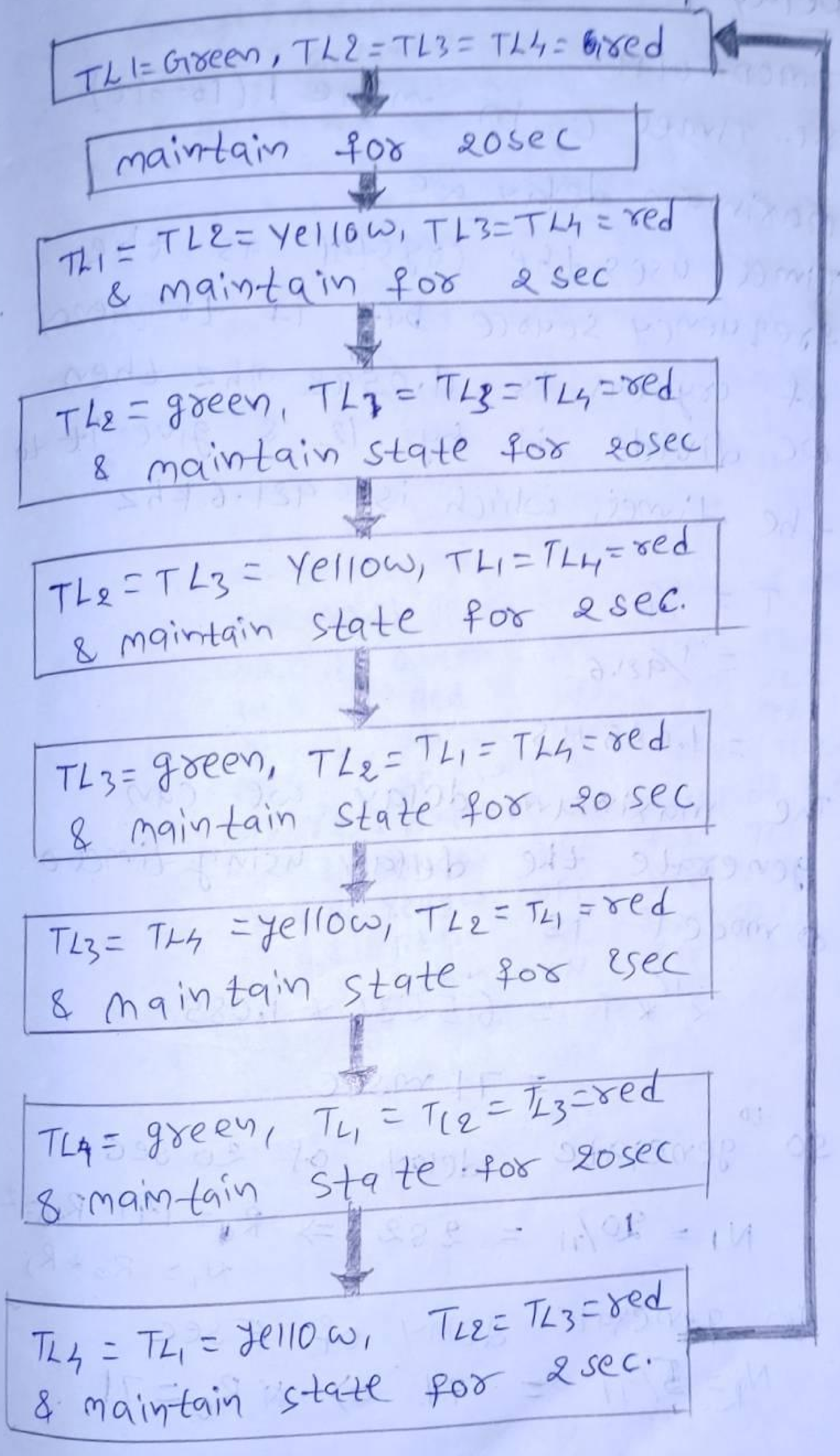
Time diagram for each traffic light and shows which of 3LED's will be ON in that time

Lane / ses	0-20	20-22	22-42	42-44	44-64	64-66	66-86	86-88
1 TL1	G	Y	R	R	R	R	R	Y
2 TL2	R	Y	G	Y	R	R	R	R
3 TL3	R	R	R	Y	G	Y	R	R
4 TL4	R	R	R	R	R	Y	G	Y

Port configuration for TLi

	Green LED	Yellow LED	RED LED
TL1	P0.0	P0.1	P0.2
TL2	P0.3	P0.4	P0.5
TL3	P2.0	P2.1	P2.2
TL4	P2.3	P2.4	P2.5

Flow-chart.



TL1 = Green, TL2 = TL3 = TL4 = red

maintain for 20sec

TL1 = TL2 = yellow, TL3 = TL4 = red & maintain for 2 sec

TL2 = green, TL1 = TL3 = TL4 = red & maintain state for 20sec

TL2 = TL3 = yellow, TL1 = TL4 = red & maintain state for 2 sec.

TL3 = green, TL2 = TL1 = TL4 = red & maintain state for 20 sec

TL3 = TL4 = yellow, TL2 = TL1 = red & maintain state for 2sec

TL4 = green, TL1 = TL2 = TL3 = red & maintain state for 20sec

TL4 = TL1 = yellow, TL2 = TL3 = red & maintain state for 2 sec.

Delay Part design :

TMOD = 01H

i.e. Timer 0 in mode 1 (16-bit)

~~maximum delay we~~

Timer uses the crystal as the frequency source but if frequency of crystal is 11.0592 mhz then we divide ~~it~~ by 12 & give it to the timer, which is 921.6 khz

$T = 1/F$

$= 1/921.6$

$= 1.085 \mu s$

The maximum delay we can generate the delay using timer 0 mode 1 is

$2^{16} * T = 65536 \times 1.085$

$= 71 \text{ msec}$

so to generate delay of 20 sec

$N_1 = 20/71 = 282 \Rightarrow R_1 = 141 \quad R_0 = 2$

$N_1 = R_0 * R_1$

to generate delay of 5 sec

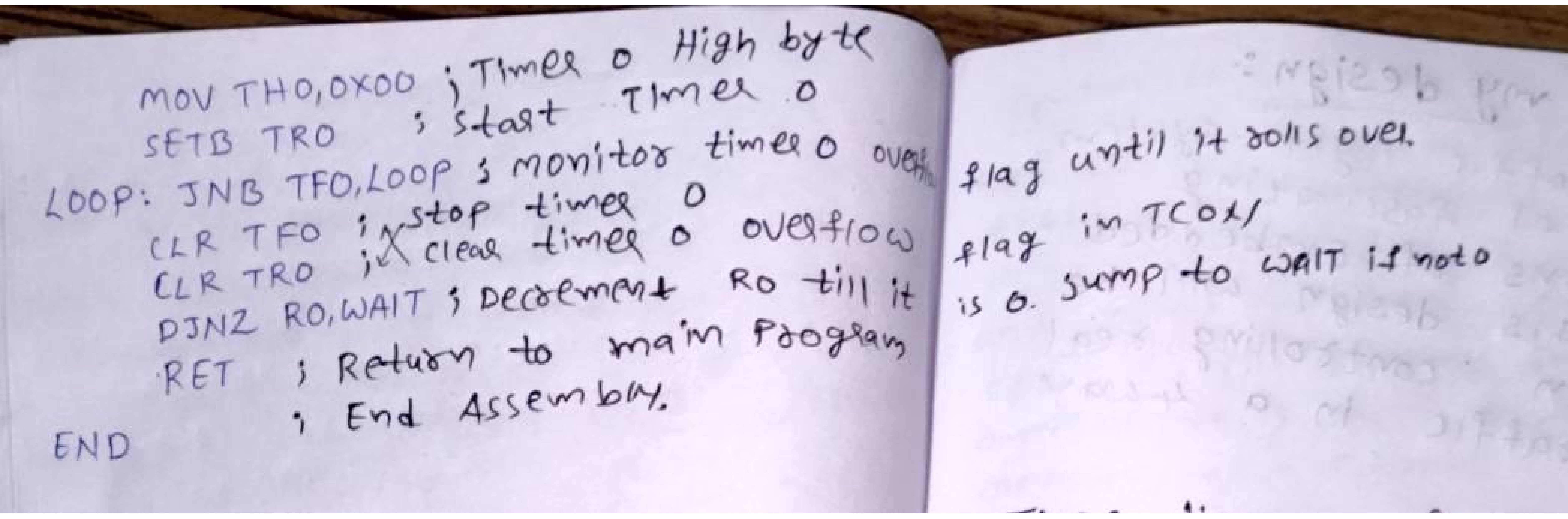
$N_2 = 5/71 = 71 \Rightarrow R_0 = 71$

Scanned by TapScanner

Assembly Code:

```
ORG 0000H ; Assembly starts from 0000H
LJMP START ; Go/jump to main code
ORG 0200H ; start location for main code
START: CLR P0.0 ; ~~red on~~ & green on For TL1
    SETB P0.1 ; Yellow OFF for TL1
    SETB P0.2 ; Red OFF for TL1
    SETB P0.3 ; Green OFF for TL2
    SETB P0.4 ; yellow OFF for TL2
    CLR P0.5 ; REd ON for TL2
    SETB P2.0 ; Green OFF for TL3
    SETB P2.1 ; yellow OFF for TL3
    CLR P2.2 ; Red ON for TL3
    SETB P2.3 ; Green OFF for TL4
    SETB P2.4 ; yellow OFF for TL4
    CLR P2.5 ; Red ON for TL4
    LCALL DELAY20 ; generate a delay of 20sec
    SETB P0.0 ; Green OFF for TL1
    SETB P0.5 ; Red OFF for TL2
    CLR P0.1 ; yellow ON for TL1
    CLR P0.4 ; yellow ON for TL2
    LCALL DELAY2 ; generate a delay of 2sec
    SETB P0.1 ; yellow OFF for TL1
    SETB P0.4 ; Yellow OFF for TL2
    CLR P0.2 ; RED ON for TL1
    CLR P0.3 ; Green ON for TL2
    LCALL DELAY20 ; delay of 20 sec
    SETB P0.3 ; Green off for TL2
    SETB P2.2 ; Red off for TL3
    CLR P0.4 ; yellow ON for TL2
    CLR P2.1 ; Yellow ON for TL3
    LCALL DELAY2 ; delay of 2 sec
    SETB P0.4 ; yellow OFF for TL2
    SETB P2.1 ; yellow OFF for TL3
    CLR P0.5 ; REd ON for TL2
    CLR P2.0 ; Green ON for TL3
    LCALL DELAY20 ; delay of 20 sec
```

```
        SETB P2.0 ; Green OFF for TL3
        SETB P2.5 ; Red OFF for TL3
        CLR  P2.1 ; yellow ON for TL3
        CLR  P2.4 ; yellow ON for TL4
        LCALL DELAY2 ; delay of 2sec
        SETB P2.1 ; yellow OFF for TL3
        SETB P2.4 ; Yellow OFF for TL4
        CLR  P2.2 ; RED ON for TL3
        CLR  P2.3 ; Green ON for TL4
        LCALL DELAY20 ; delay of 20sec
        SETB P2.3 ; Green off for TL4
        SETB P0.2 ; Red OFF for TL1
        CLR  P2.4 ; yellow ON for TL4
        CLR  P0.1 ; yellow ON for TL1
        LCALL DELAY2 ; delay of 2 sec
        SJMP START ; again repeat same PROCESS code
ORG 1200H ; start Location for DELAY20
DELAY20: MOV TMOD,#01 ; Timer 0, mode1
         MOV R1,#2 ; Load register R1 with 2
WAIT1: MOV R0,#141 ; Load register R0 with 141
WAIT2: MOV TL0,0X00 ; Timer 0 Low byte count
       MOV TH0,0X00 ; Timer 0 High byte count (overall value=0X0000)
       SETB TR0 ; start Timer 0 (TR0 is run control bit D4 in TCON SFR)
CHECK: JNB TF0,CHECK ; monitor Timer 0 overflow flag until it rolls over (TF0 is bit5 in TCON)
       CLR TR0 ; stop timer 0
       CLR TF0 ; clear Timer 0 overflow flag in TCON
       DJNZ R0,WAIT2 ; Decrement R0 till it is 0. Jump to WAIT2 if not 0.
       DJNZ R1,WAIT1 ; Decrement R1 till it is 0. Jump to WAIT1 if not0.
       RET ; Return to main Program
ORG 1700H ; start location for DELAY2
DELAY2: MOV TMOD,#01 ; Timer 0, mode1
        MOV R0,#71 ; Load register R0 with 71
WAIT: MOV TL0,0X00 ; Timer 0 Low byte
```



```
      MOV TH0,0X00 ; Timer 0 High byte
      SETB TR0     ; start Timer 0
LOOP: JNB TF0,LOOP ; monitor timer 0 overflow flag until it rolls over.
      CLR TF0      ; stop timer 0
      CLR TR0      ; clear timer 0 overflow flag in TCON/
      DJNZ R0,WAIT ; Decrement R0 till it is 0. sump to WAIT if not 0
      RET          ; Return to main Program
                   ; End Assembly.
END
```

utility of my design :-

* 4-way Traffic light system was one of fascinating applications of Embedded systems. This design will be helpful in controlling real life traffic in a 4-way junction.