Port Structure of 8051 microcontroller

Program Status Word

- Register set select
- Status bits

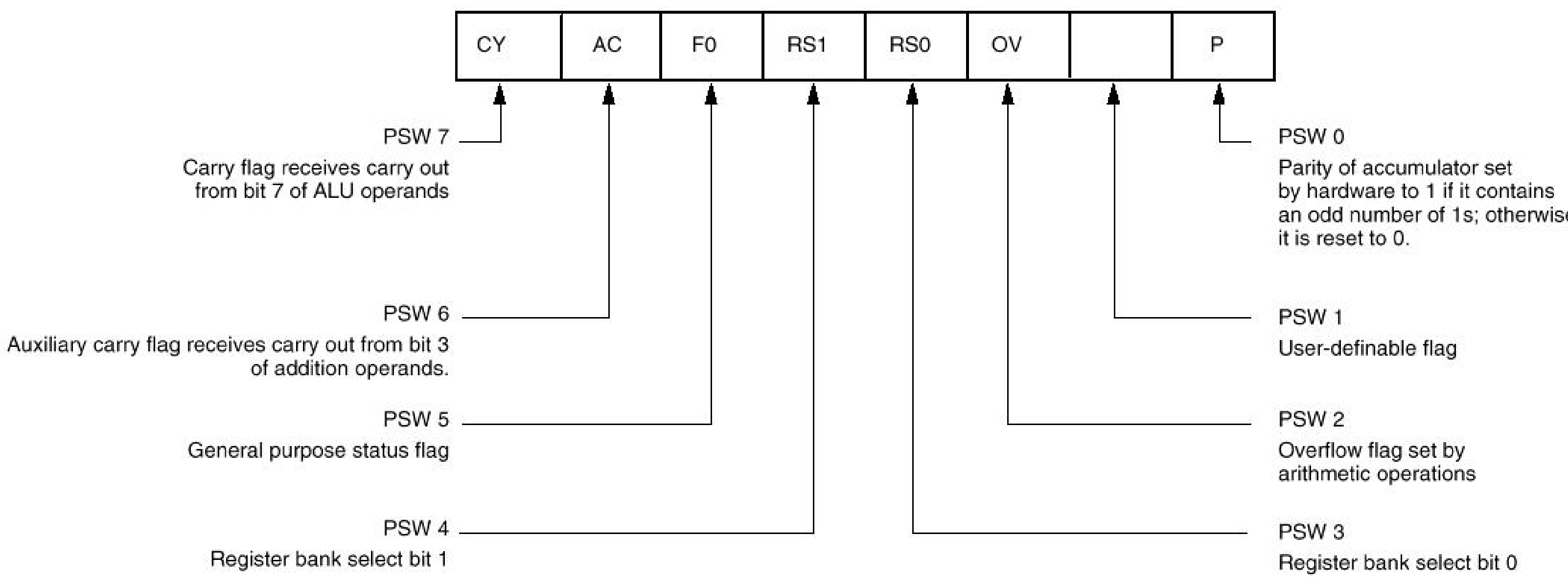
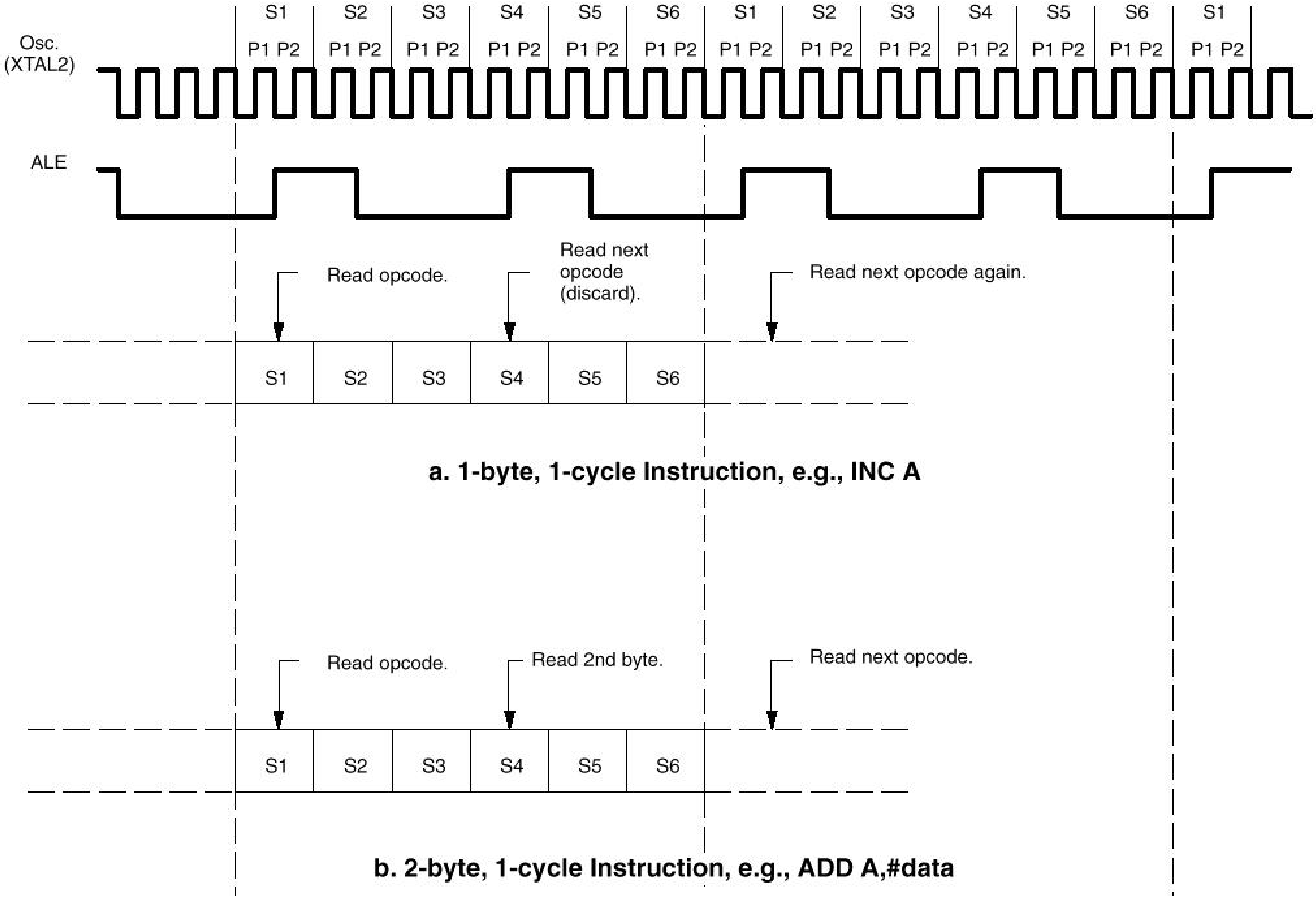


Figure 10. PSW (Program Status Word) Register in 80C51 Devices

Instruction Timing

- One “machine cycle” = 6 states (S1 - S6)
- One state = 2 clock cycles
  - One “machine cycle” = 12 clock cycles
- Instructions take 1 - 4 cycles
  - e.g. 1 cycle instructions: ADD, MOV, SETB, NOP
  - e.g. 2 cycle instructions: JMP, JZ
  - 4 cycle instructions: MUL, DIV

Instruction Timing



a. 1-byte, 1-cycle Instruction, e.g., INC A

b. 2-byte, 1-cycle Instruction, e.g., ADD A,#data

Pins of I/O Port

- The 8051 has four I/O ports

  Port 0 (pins 32-39) : P0 (P0.0 ~ P0.7)

  Port 1 (pins 1-8) : P1 (P1.0 ~ P1.7)

  Port 2 (pins 21-28) : P2 (P2.0 ~ P2.7)

  Port 3 (pins 10-17) : P3 (P3.0 ~ P3.7)

  Each port has 8 pins.

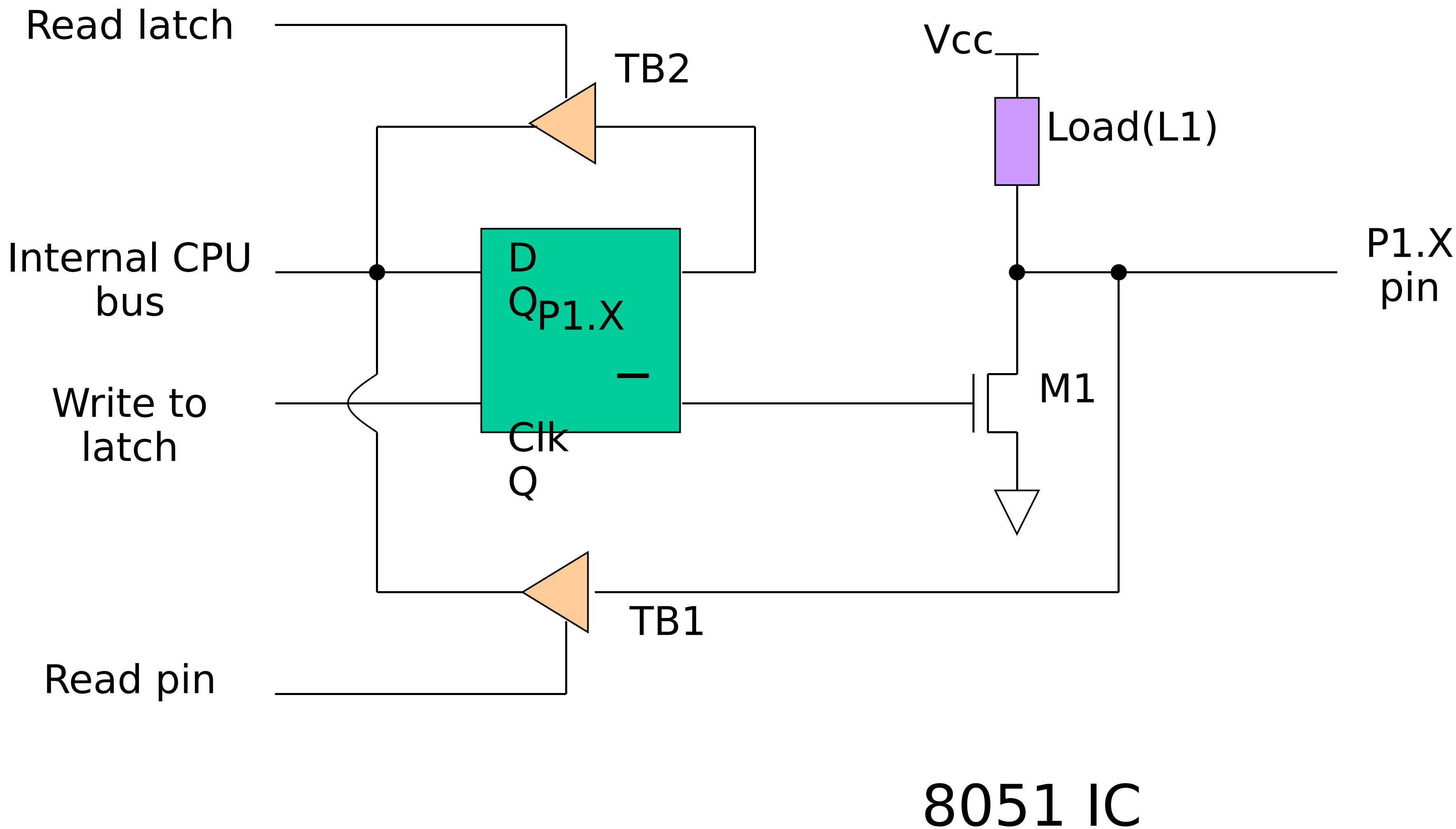
  - Named P0.X (X=0,1,...,7) , P1.X, P2.X, P3.X
  - Ex : P0.0 is the bit 0 (LSB) of P0
  - Ex : P0.7 is the bit 7 (MSB) of P0
  - These 8 bits form a byte.
- Each port can be used as input or output (bi-direction).

I/O Port Programming

Port 1 (pins 1-8)

- Port 1 is denoted by P1.
  - P1.0 ~ P1.7
- We use P1 as examples to show the operations on ports.
  - P1 as an output port (i.e., write CPU data to the external pin)
  - P1 as an input port (i.e., read pin data into CPU bus)

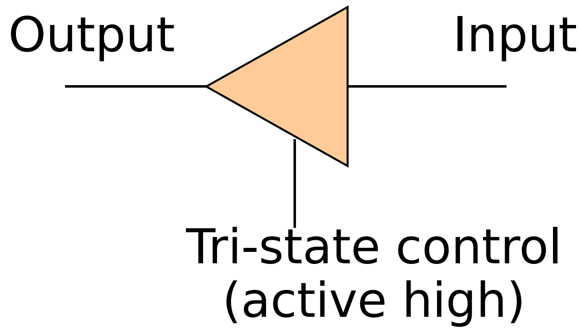
A Pin of Port 1

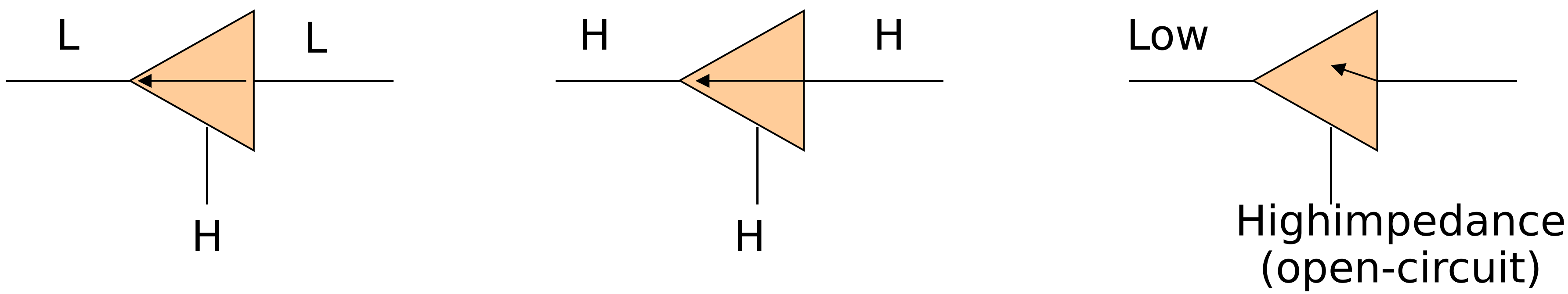


Hardware Structure of I/O Pin

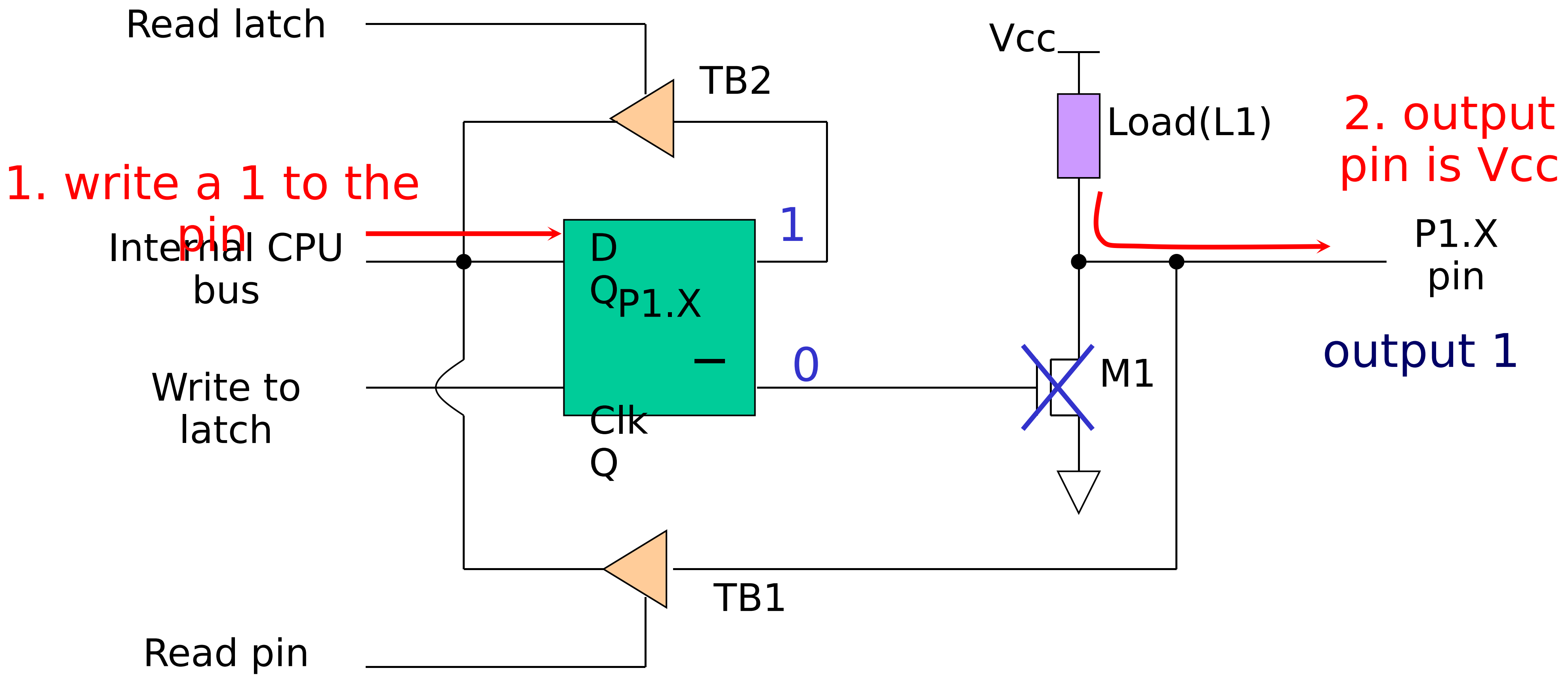
- Each pin of I/O ports
  - Internal CPU bus : communicate with CPU
  - D latch store the value of this pin
    - D latch is controlled by “Write to latch”
      - Write to latch = 1 : write data into the D latch
  - 2 Tri-state buffer :
    - TB1: controlled by “Read pin”
      - Read pin = 1 : really read the data present at the pin
    - TB2: controlled by “Read latch”
      - Read latch = 1 : read value from internal latch
  - A transistor M1 gate
    - Gate=0: open
    - Gate=1: close

Tri-state Buffer



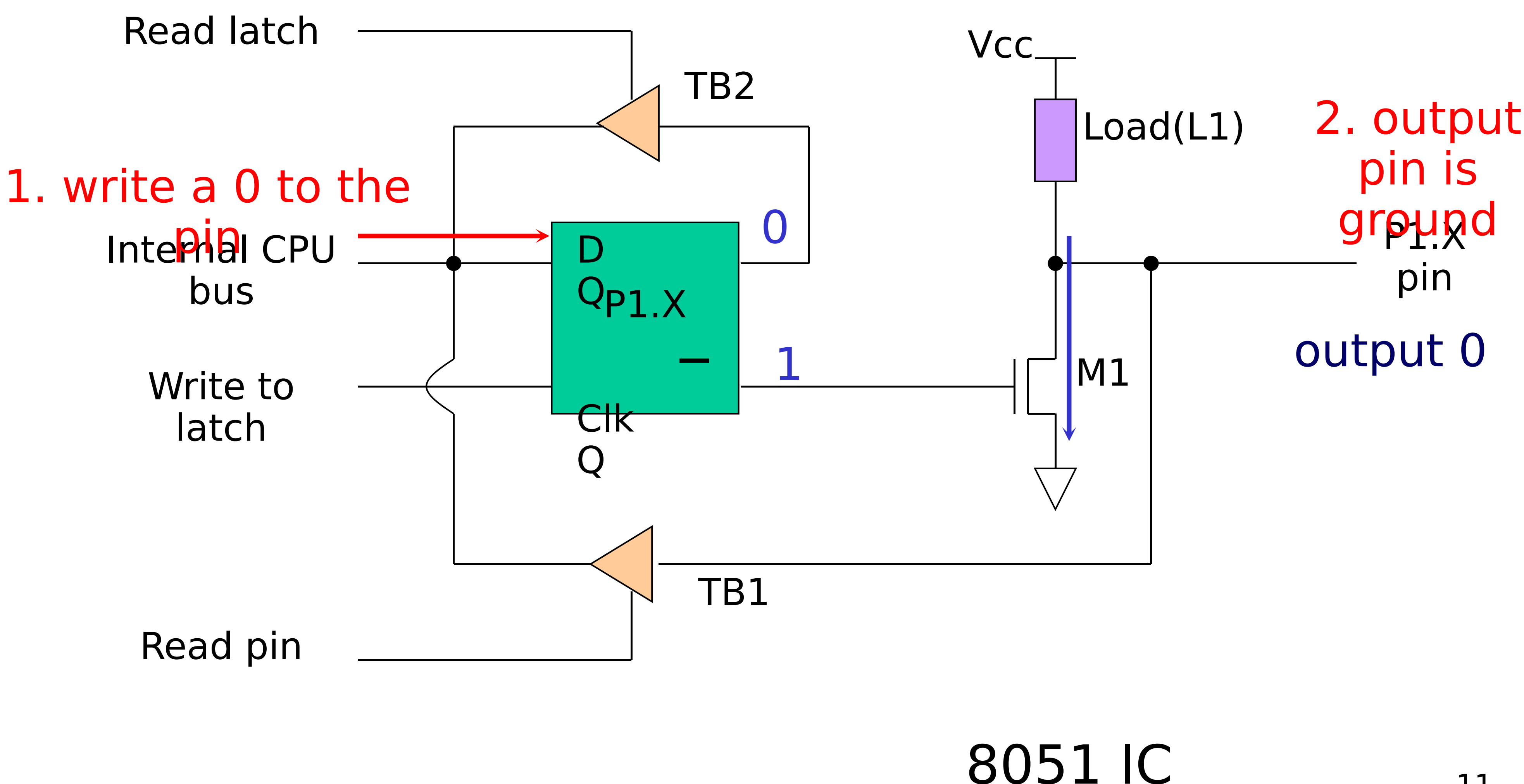


Writing “1” to Output Pin P1.X

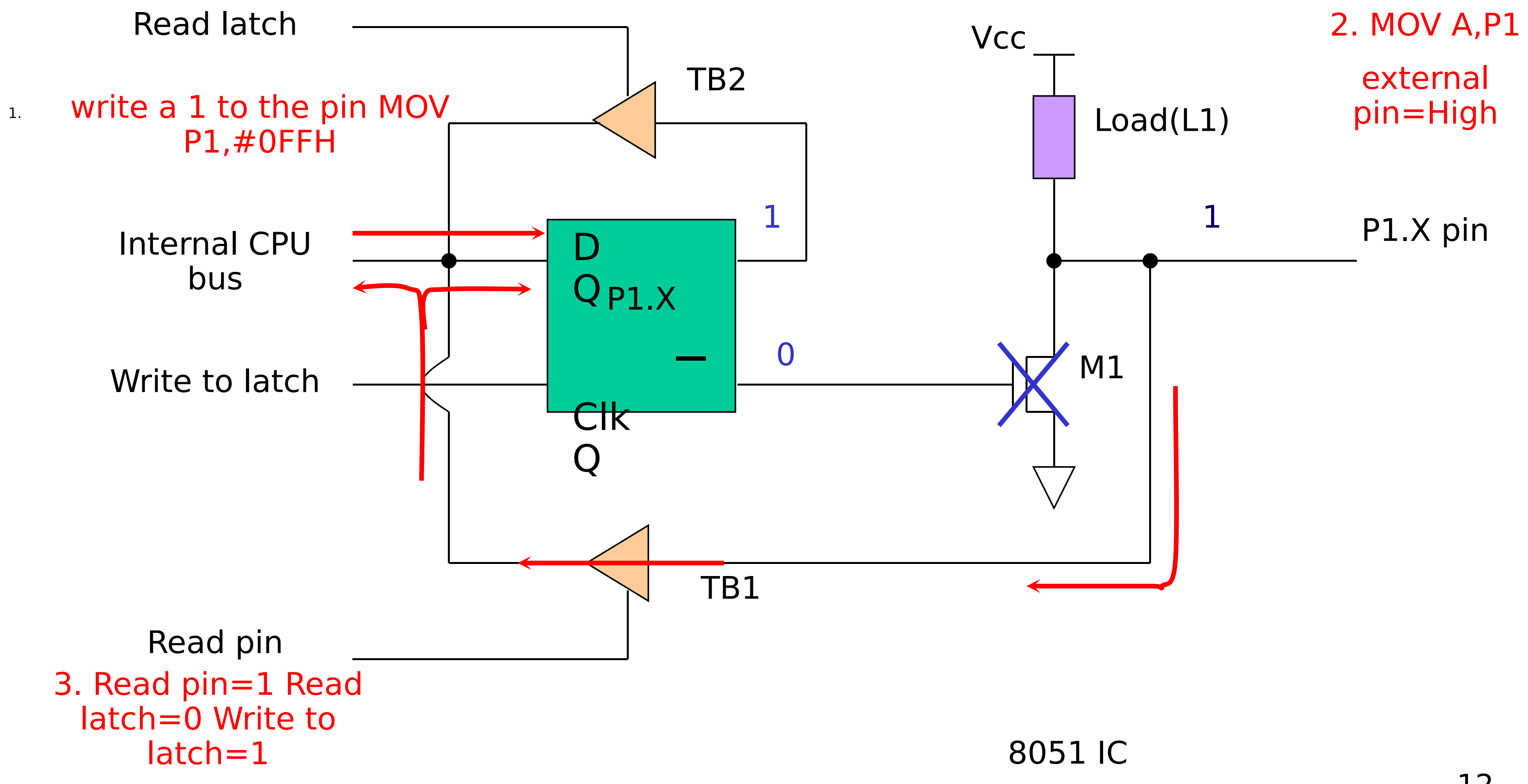


8051 IC

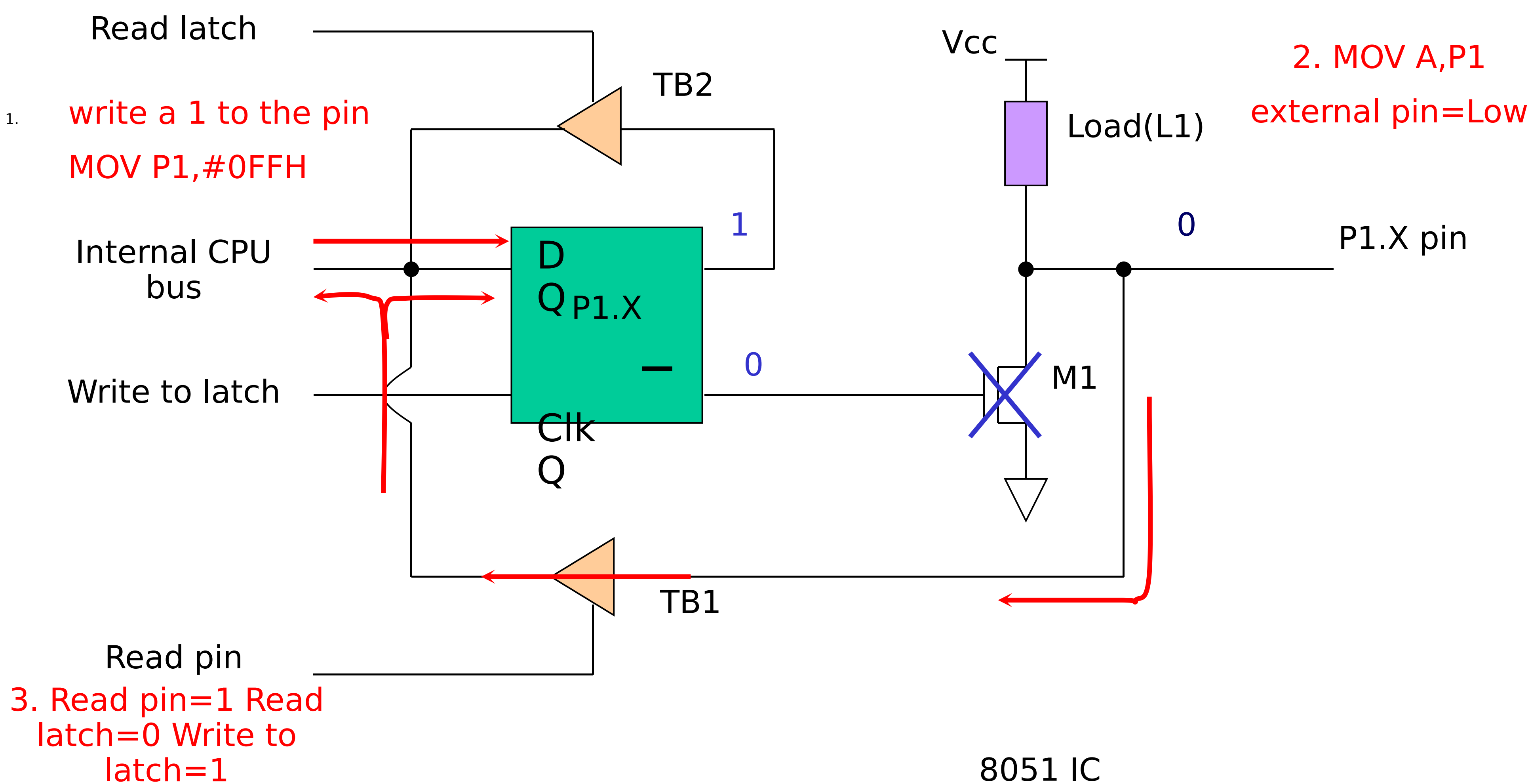
Writing “0” to Output Pin P1.X



Reading “High” at Input Pin



Reading “Low” at Input Pin



3. Read pin=1 Read latch=0 Write to latch=1

Port 1 as Input (Read from Port)

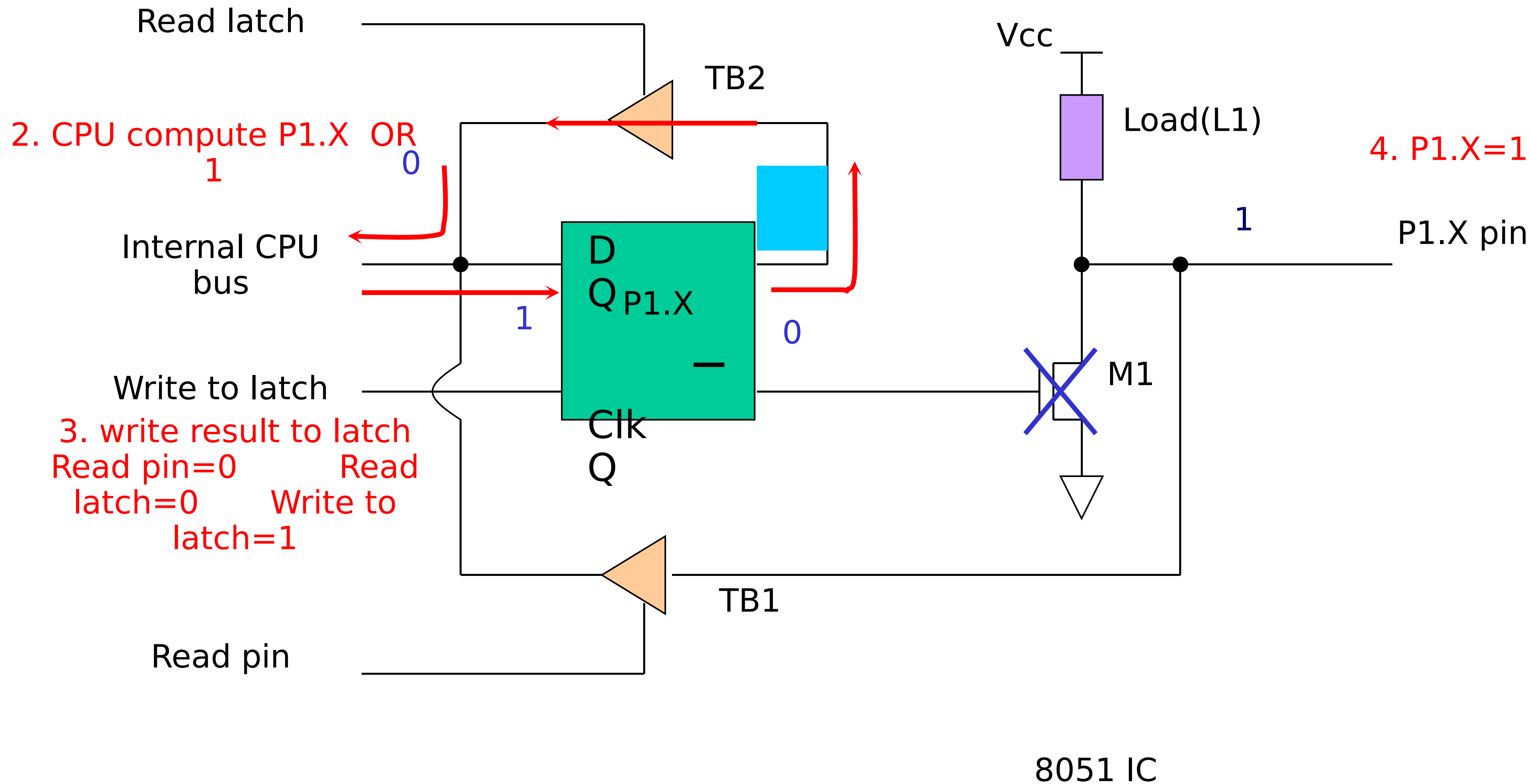
- In order to make P1 an input, the port must be programmed by writing 1 to all the bit.

```
        MOV     A,#0FFH     ;A=11111111B
        MOV     P1,A        ;make P1 an input port
BACK:   MOV     A,P1        ;get data from P0
        MOV     P2,A        ;send data to P2
        SJMP    BACK
```

To be an input port, P0, P1, P2 and P3 have similar methods.

Reading the Latch

1. Read pin=0 Read latch=1 Write to latch=0 (Assume P1.X=0 initially)



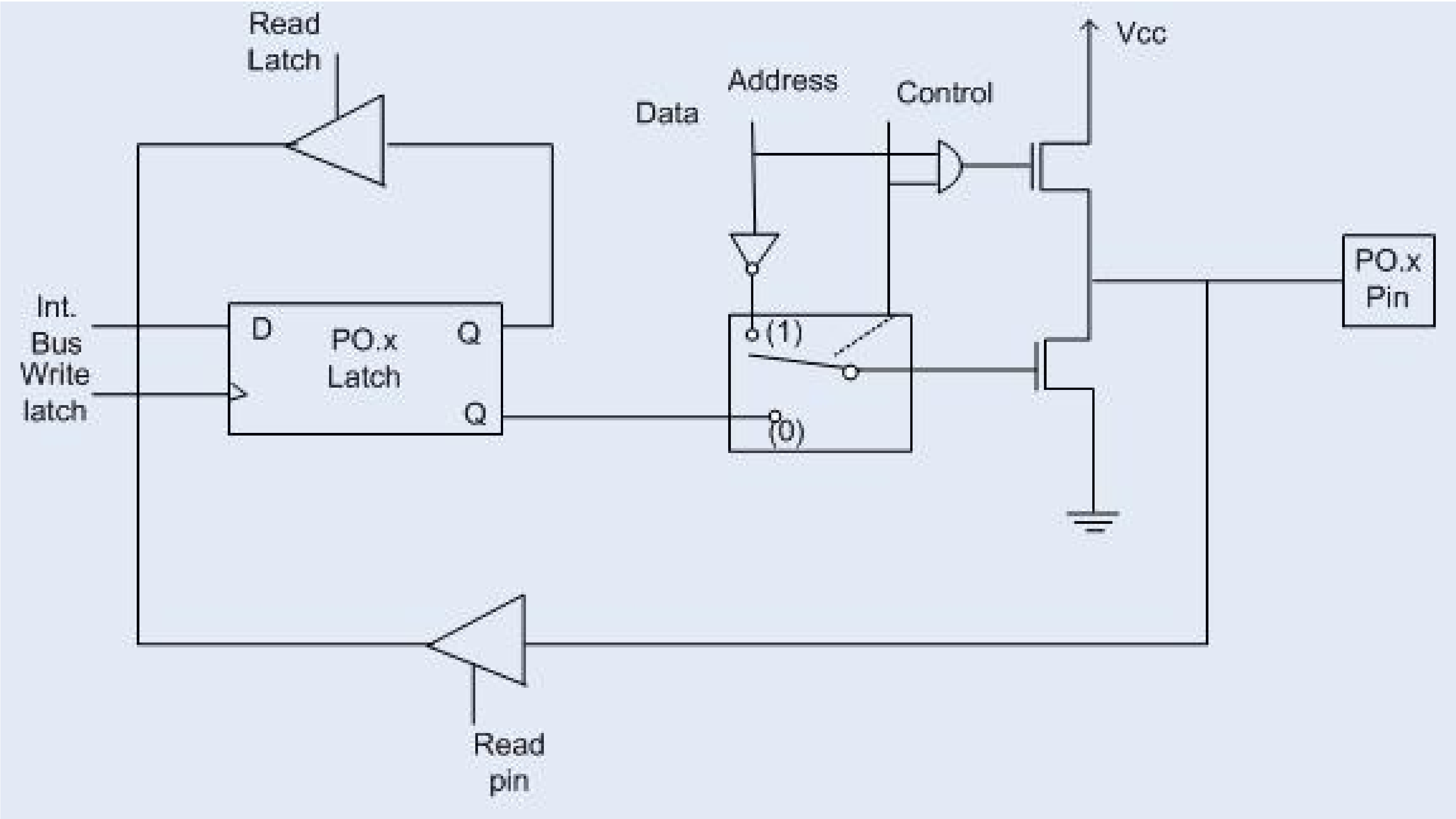
Read-modify-write Feature

- Read-modify-write Instructions
- This features combines 3 actions in a single instruction :

  1. CPU reads the latch of the port
  2. CPU perform the operation
  3. Modifying the latch
  4. Writing to the pin

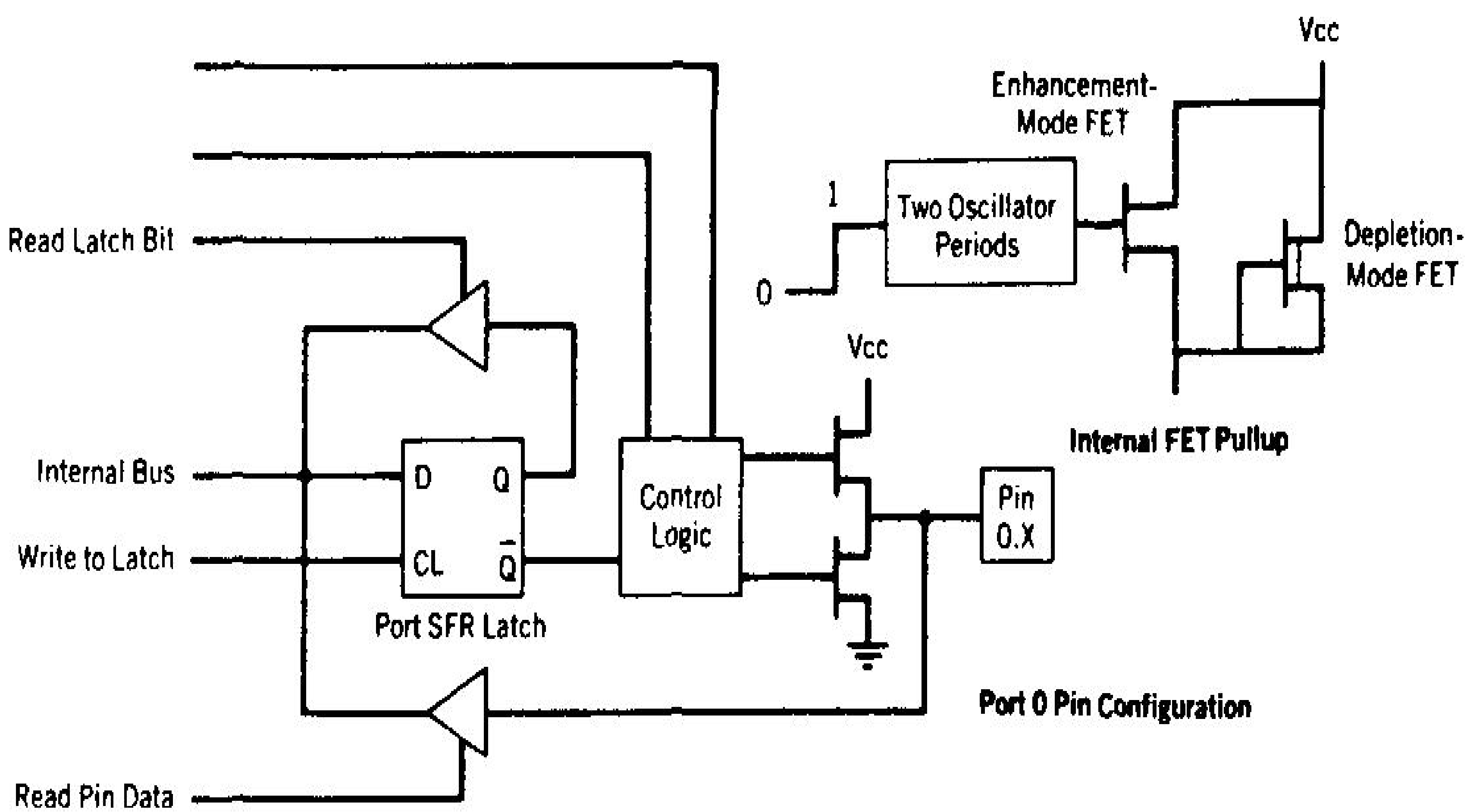
  Note that 8 pins of P1 work independently.

Port - 0

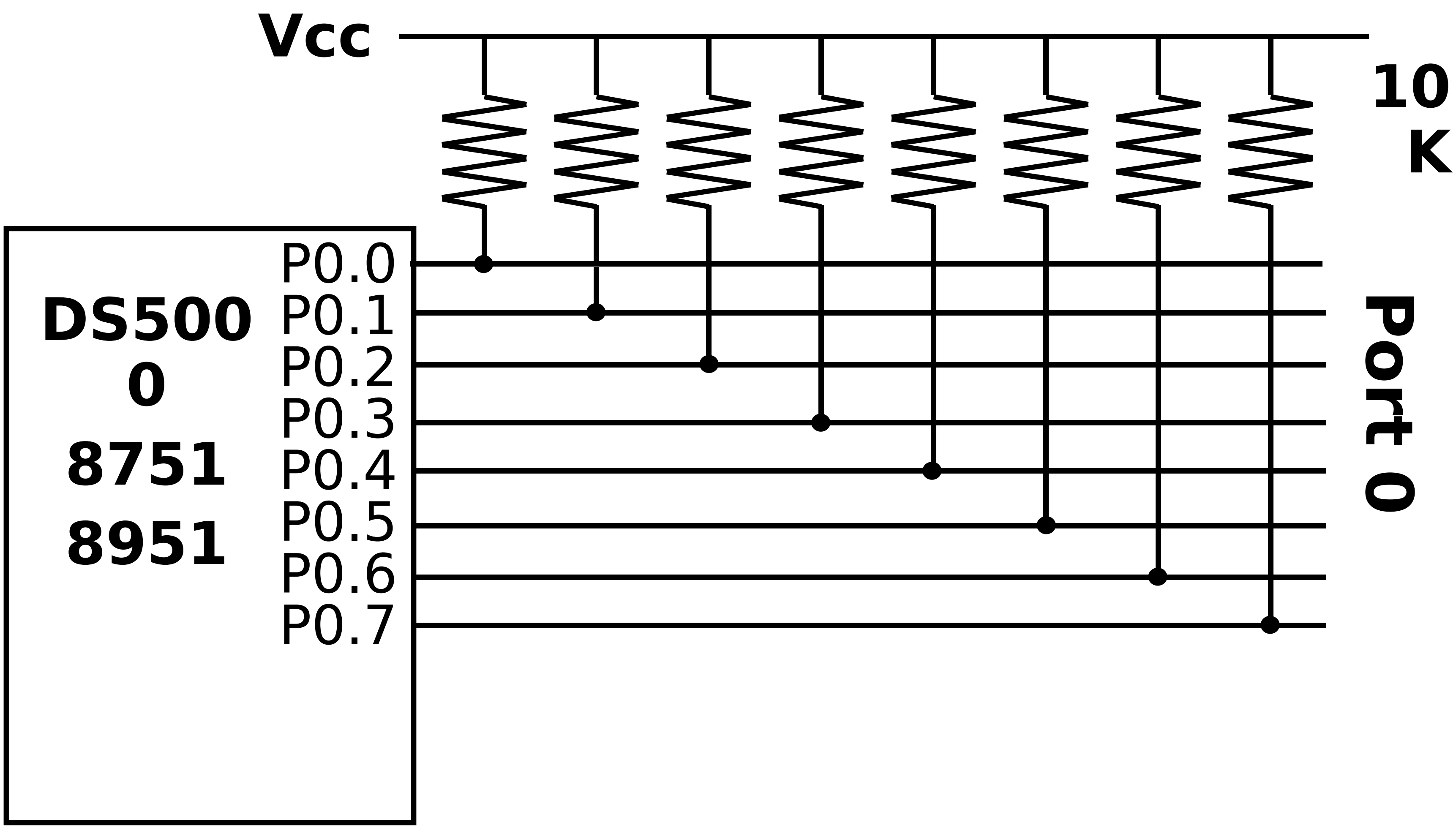


Read Latch
Vcc
Address
Control
Data
Int. Bus
Write latch
D
PO.x Latch
Q
Q
(1)
(0)
PO.x Pin
Read pin

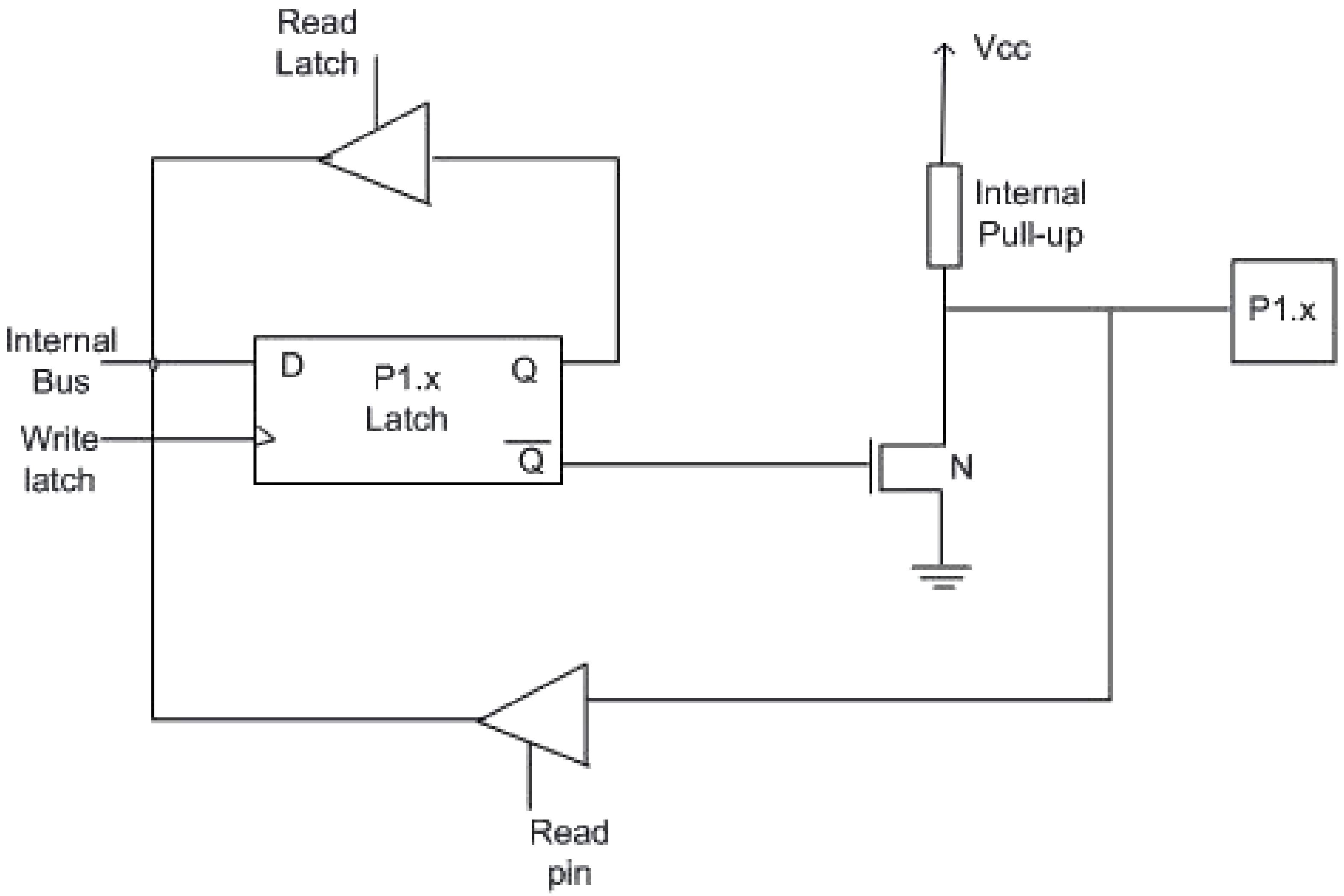
Port0


Vcc
Enhancement-
Mode FET
Two Oscillator
Periods
1
0
Read Latch Bit
Depletion-
Mode FET
Vcc
Internal FET Pullup
Internal Bus
D
Q
Control
Logic
Pin
0.X
Write to Latch
CL
$\bar{Q}$
Port SFR Latch
Port 0 Pin Configuration
Read Pin Data

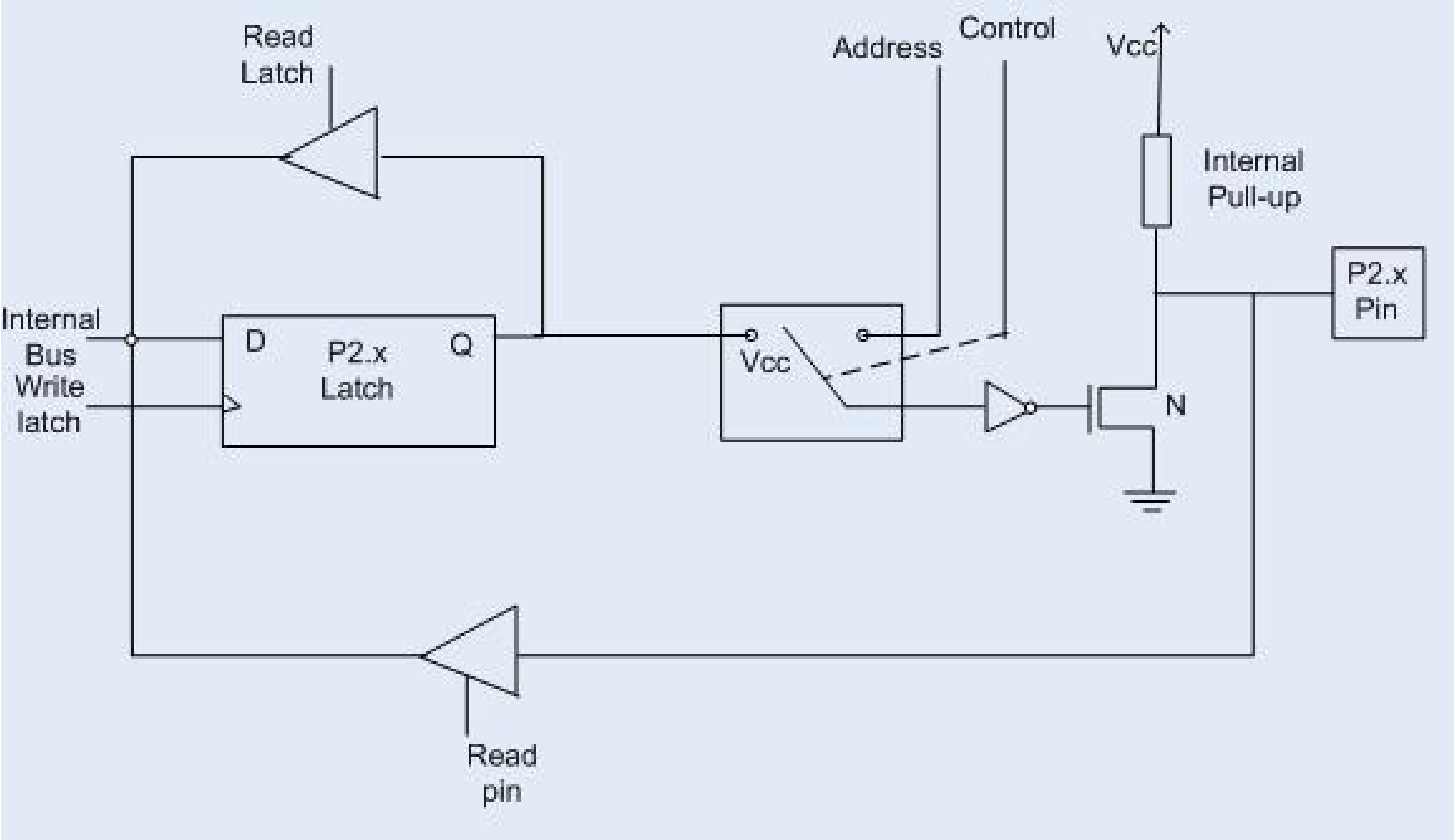
Port 0 with Pull-Up Resistors



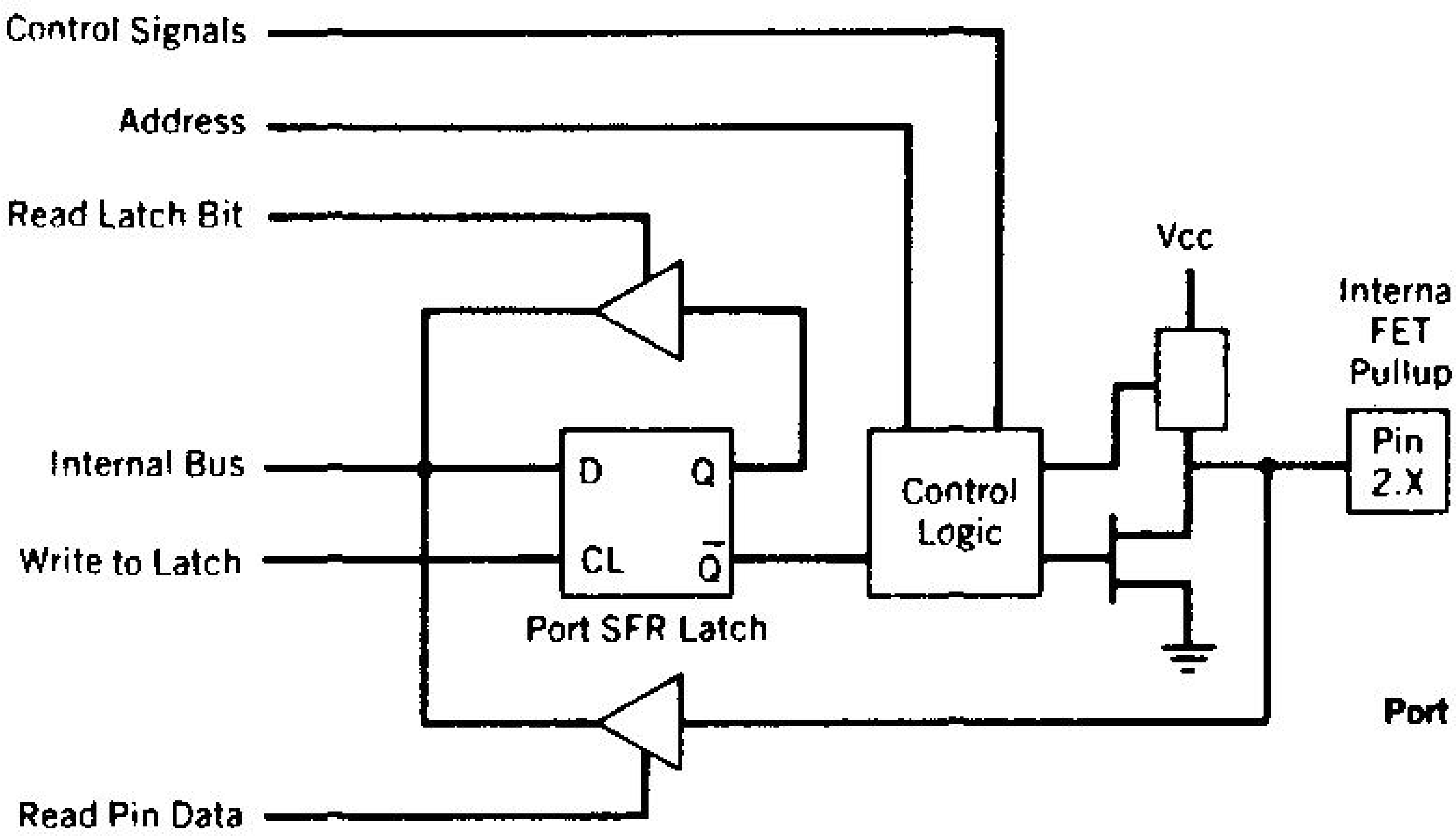
Port - 1


Read
Latch
Vcc
Internal
Pull-up
P1.x
Internal
Bus
D
P1.x
Latch
Q
Write
latch
$\overline{Q}$
N
Read
pin

Port - 2

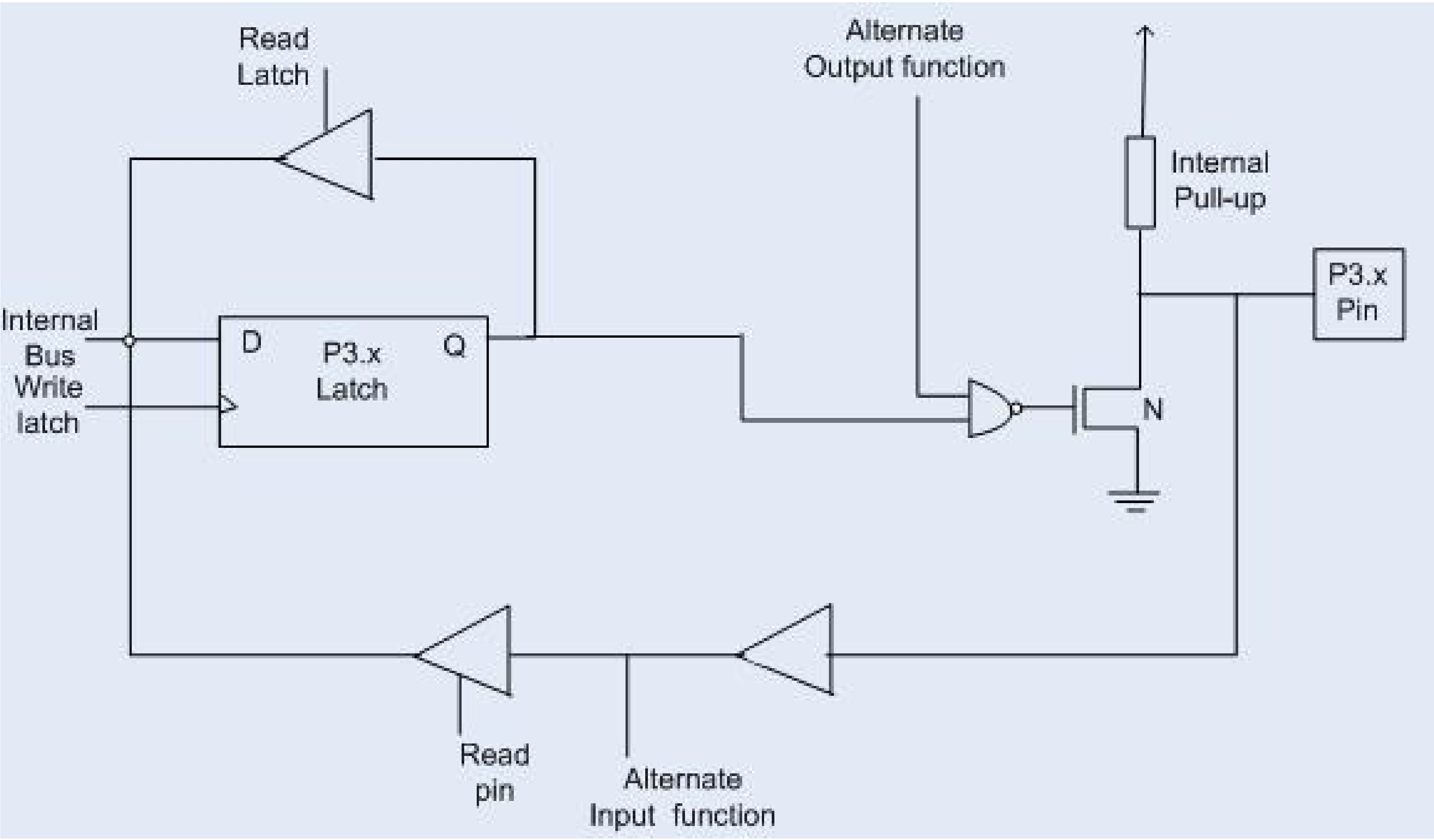

Read
Latch
Address
Control
Vcc
Internal
Pull-up
P2.x
Pin
Internal
Bus
Write
latch
D
P2.x
Latch
Q
Vcc
N
Read
pin

Port 2



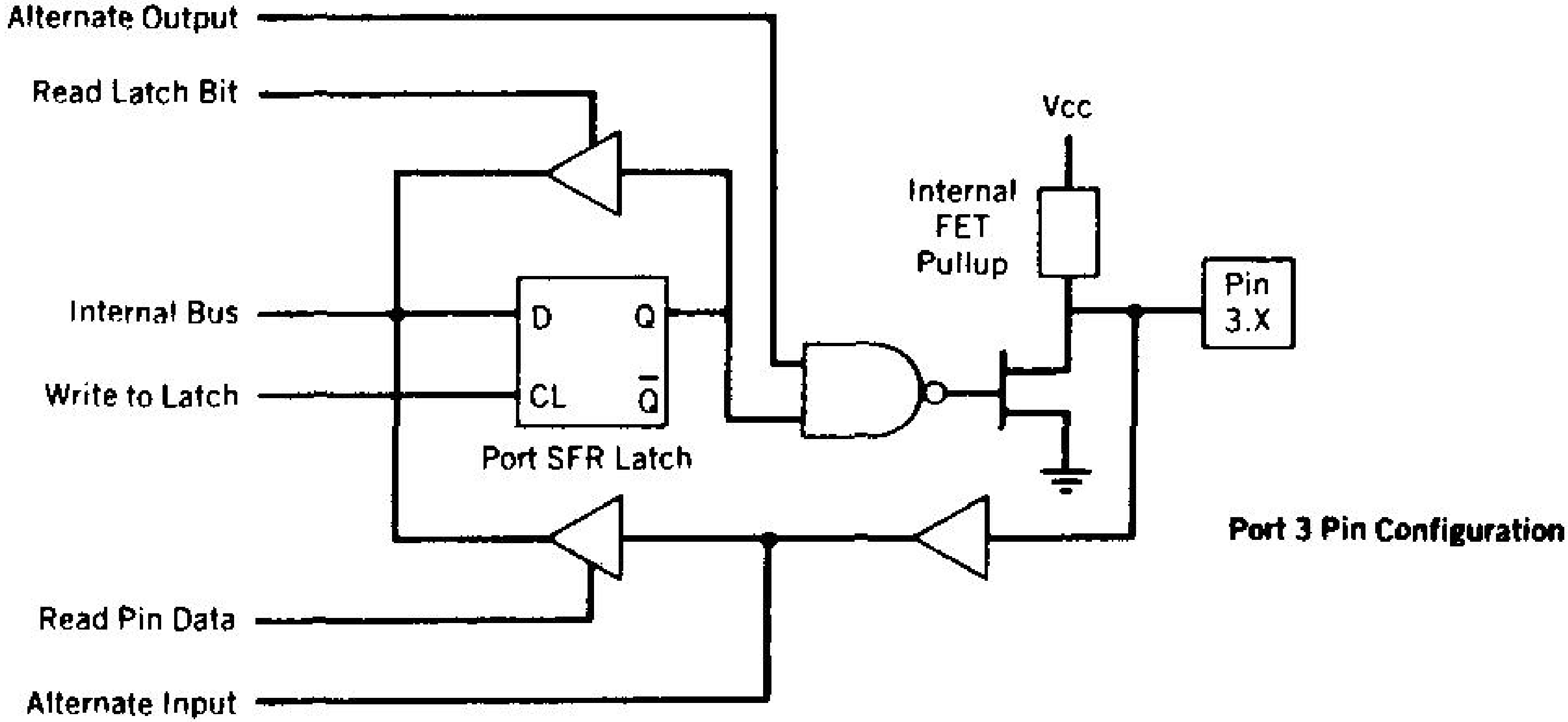
Port 2 Pin Configuration

Port - 3



Read
Latch
Alternate
Output function
Internal
Pull-up
P3.x
Pin
Internal
Bus
Write
latch
D
P3.x
Latch
Q
N
Read
pin
Alternate
Input function

Port-3

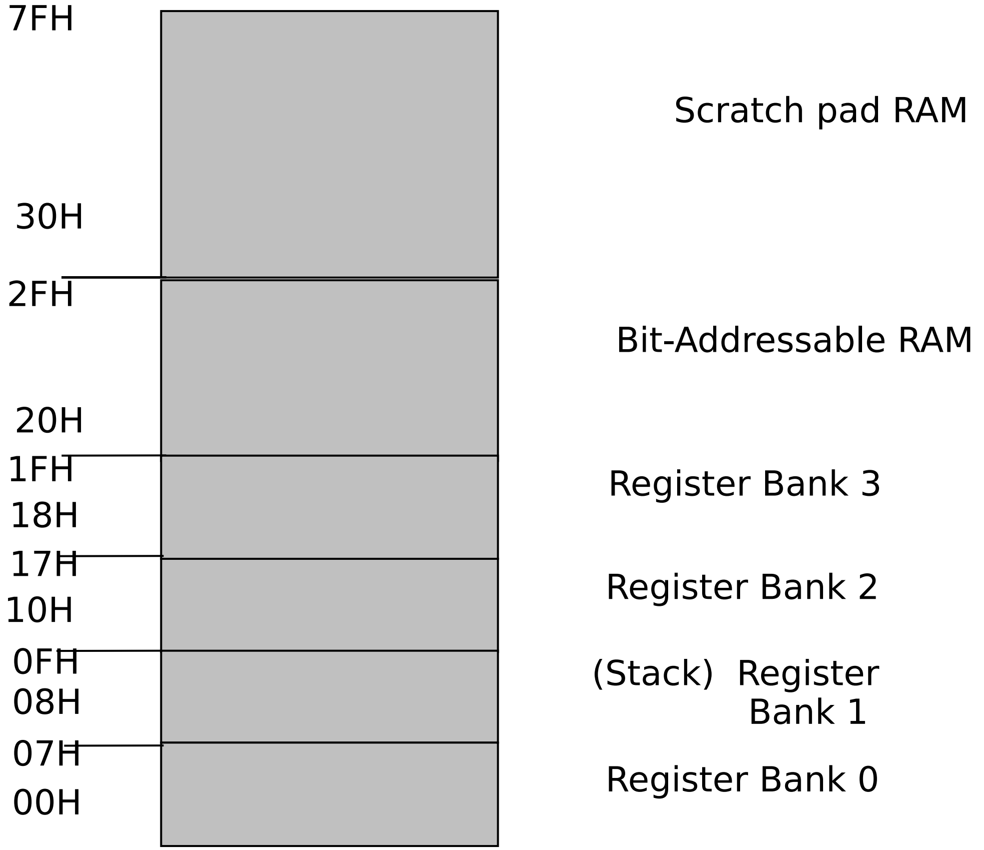


Alternate Output
Read Latch Bit
Vcc
Internal
FET
Pullup
Pin
3.X
Internal Bus
Write to Latch
D
Q
CL
$\overline{Q}$
Port SFR Latch
Port 3 Pin Configuration
Read Pin Data
Alternate Input

Stack in the 8051

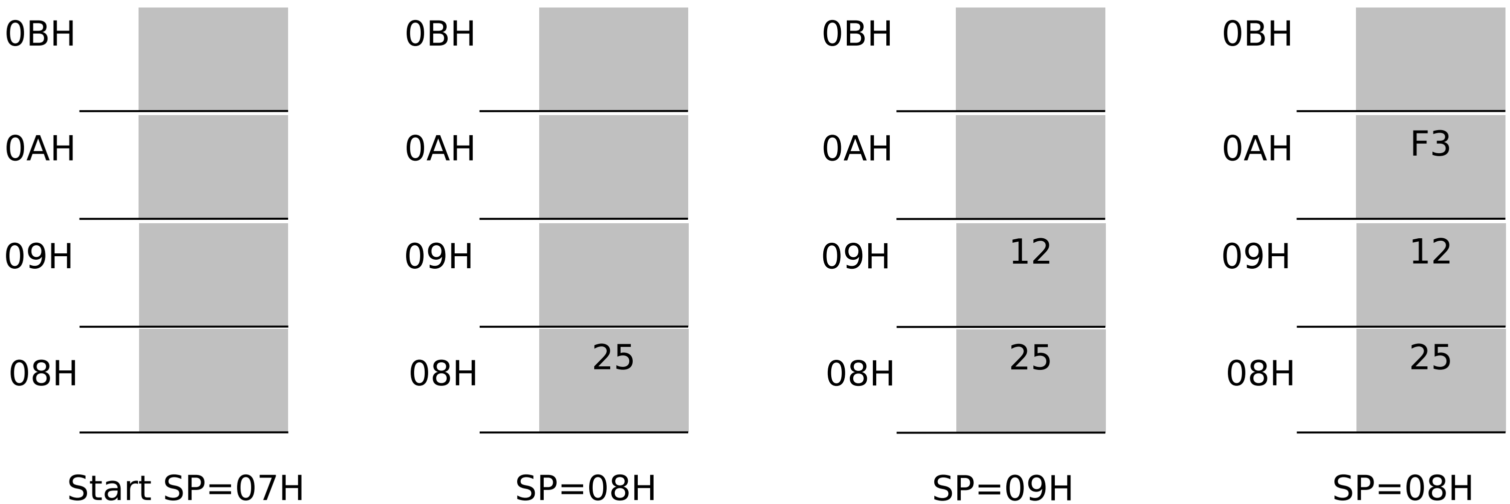
- The register used to access the stack is called **SP** (stack pointer) register.

- The stack pointer in the 8051 is only 8 bits wide, which means that it can take value 00 to FFH. <u>When 8051 powered up, the SP register contains value 07.</u>



Example:

```
MOV   R6,#25H
MOV   R1,#12H
MOV   R4,#0F3H
PUSH  6
PUSH  1
PUSH  4
```



Difference between Microprocessor and Microcontroller

1. Key difference in both of them is presence of external peripheral, where microcontrollers have RAM, ROM, EEPROM embedded in it while we have to use external circuits in case of microprocessors.
2. As all the peripheral of microcontroller are on single chip it is compact while microprocessor is bulky.
3. Microcontrollers are made by using complementary metal oxide semiconductor technology so they are far cheaper than microprocessors. In addition the applications made with microcontrollers are cheaper because they need lesser external components, while the overall cost of systems made with microprocessors are high because of the high number of external components required for such systems.
4. Processing speed of microcontrollers is about 8 MHz to 50 MHz, but in contrary processing speed of general microprocessors is above 1 GHz so it works much faster than microcontrollers.
5. Generally microcontrollers have power saving system, like idle mode or power saving mode so overall it uses less power and also since external components are low overall consumption of power is less. While in microprocessors generally there is no power saving system and also many external components are used with it, so its power consumption is high in comparison with microcontrollers.
6. Microcontrollers are compact so it makes them favorable and efficient system for small products and applications while microprocessors are bulky so they are preferred for larger applications.
7. Tasks performed by microcontrollers are limited and generally less complex. While task performed by microprocessors are software development, Game development, website, documents making etc. which are generally more complex so require more memory and speed so that’s why external ROM, RAM are used with it.
8. Microcontrollers are based on Harvard architecture where program memory and data memory are separate while microprocessors are based on von Neumann model where program and data are stored in same memory module.