The 8051 Architecture

The Microcontroller

- Microcontrollers can be considered as self-contained systems with a processor, memory and I/O ports.
  - In most cases, all that is missing is the software to define the operation of the embedded system.
- Families of microcontrollers
  - Z80 from Zilog
  - MC6811 from Motorola
  - MCS51 from Intel
- Manufacturers of 8051 compatible controllers
  - Intel
  - Atmel
  - Dallas
  - Phillips, etc

The Architectural Needs of a Microcontroller

- Lets consider what architectural features would be needed in a microcontroller.
- What are the expected applications?
  - Sensing the environment (Input)
  - Producing a response (Output)
  - The response may be delayed (Timer/Counter)
  - Prioritized response to I/O devices (Interrupts)
  - Software to control the process (Non-volatile Memory)
  - Temporary data (RAM)

Overview of the 8051

- Originally made by Intel, in 1981, can be considered as a CISC processor
- An 8-bit, single-chip microcontroller optimized for control applications
- 40 pins in a dual in-line package (DIP) layout
- 128 bytes RAM, 4096 bytes (4KB) ROM, 2 timers, 1 serial port, 4 I/O ports
- Harvard architecture
  - The 8051 has separate address spaces for Program Memory, Data Memory.
  - Since each address can refer to two different locations, the specific location is determined by the instruction used.

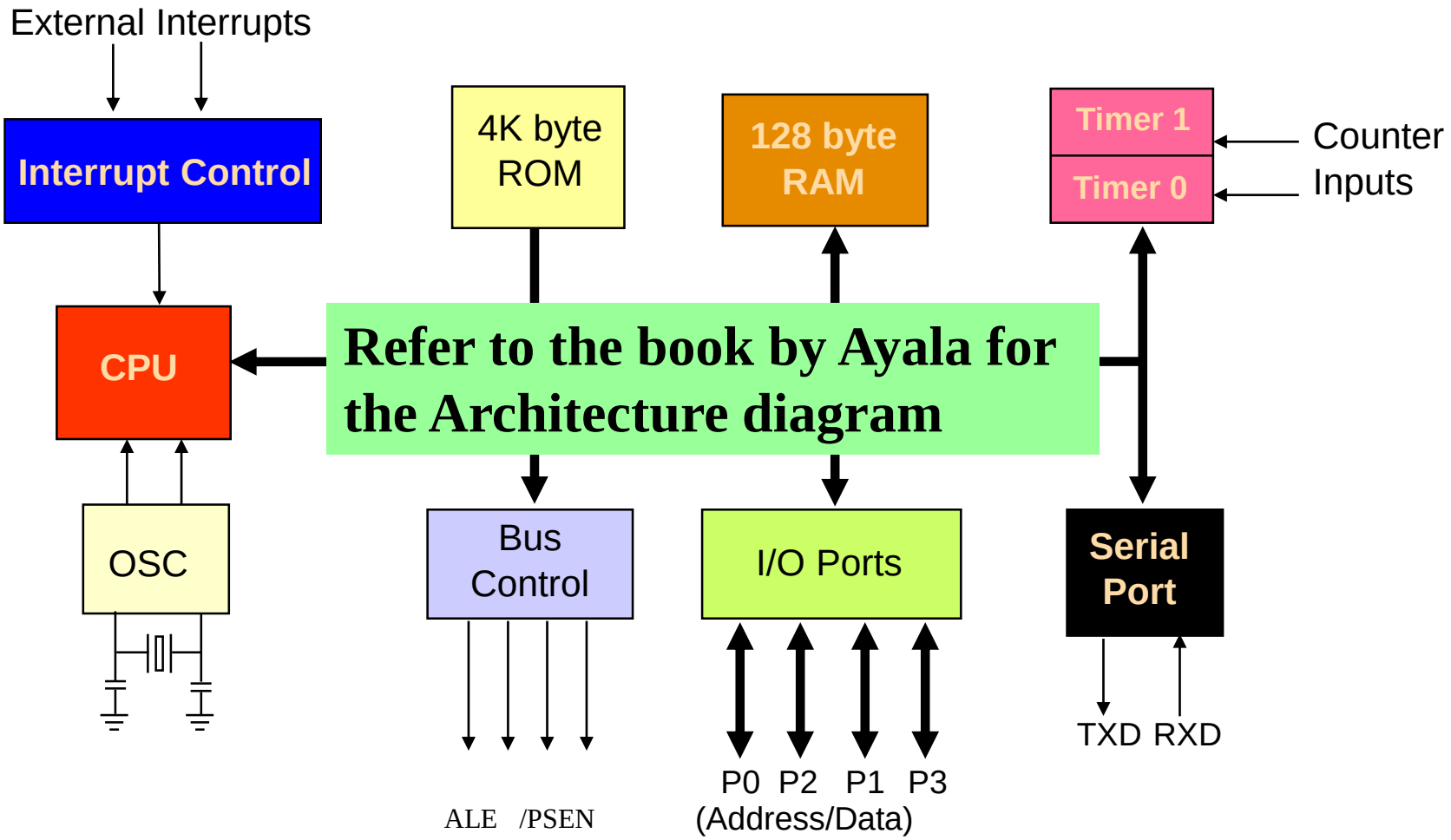
General hardware features

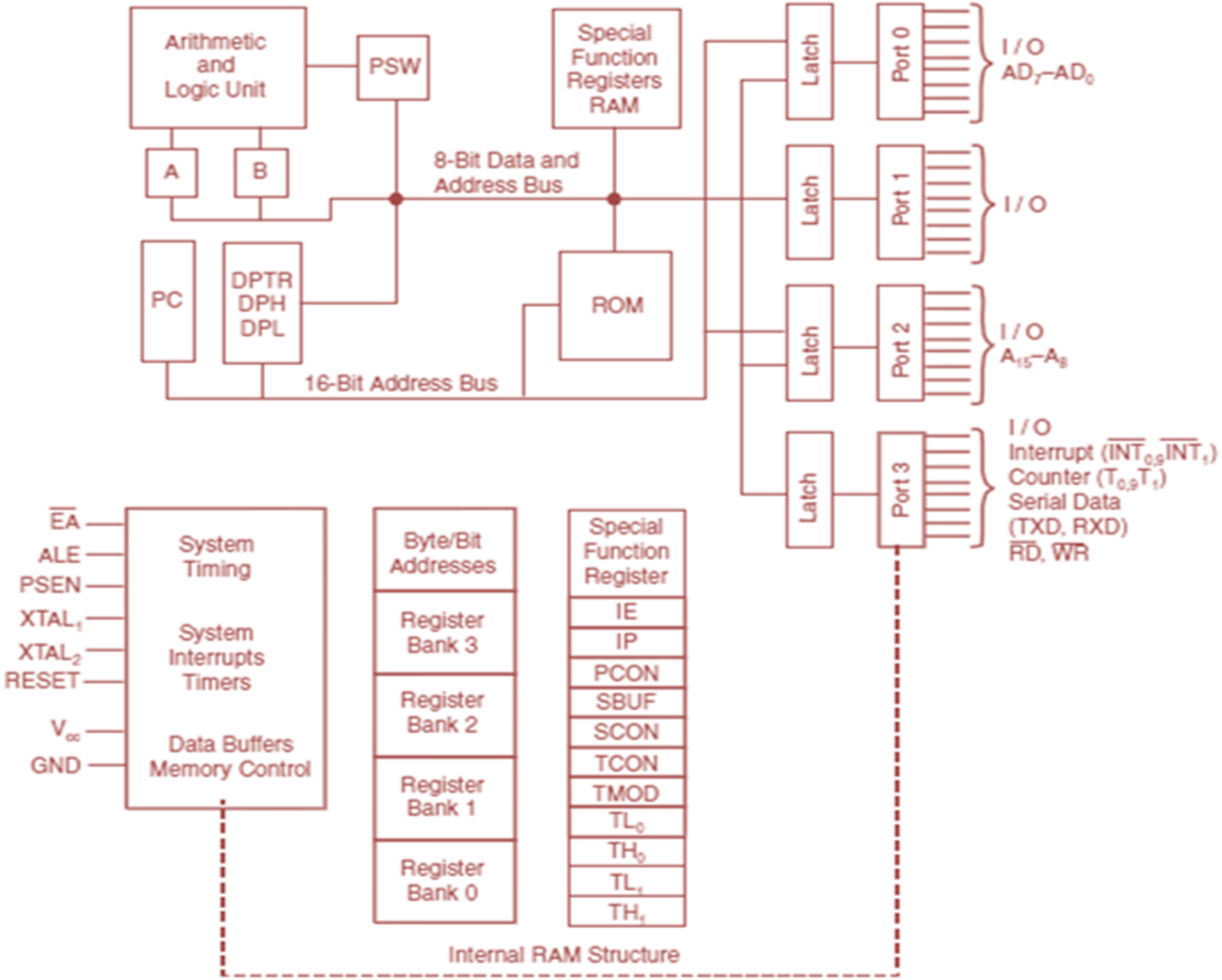
- Built-in 4KB ROM
- 128 bytes internal RAM
  - 4 register banks of 8 bytes each (R0-R7)
  - 16 bytes of bit-addressable area
  - 80 bytes of general purpose memory
- Four 8-bit I/O ports (P0-P3)
- Two 16-bit timers (Timer-0 & Timer-1)
- One serial communication interface
- Five interrupt sources (2 external & 3 internal)
- One oscillator (generates clock signal)

General Operational Features

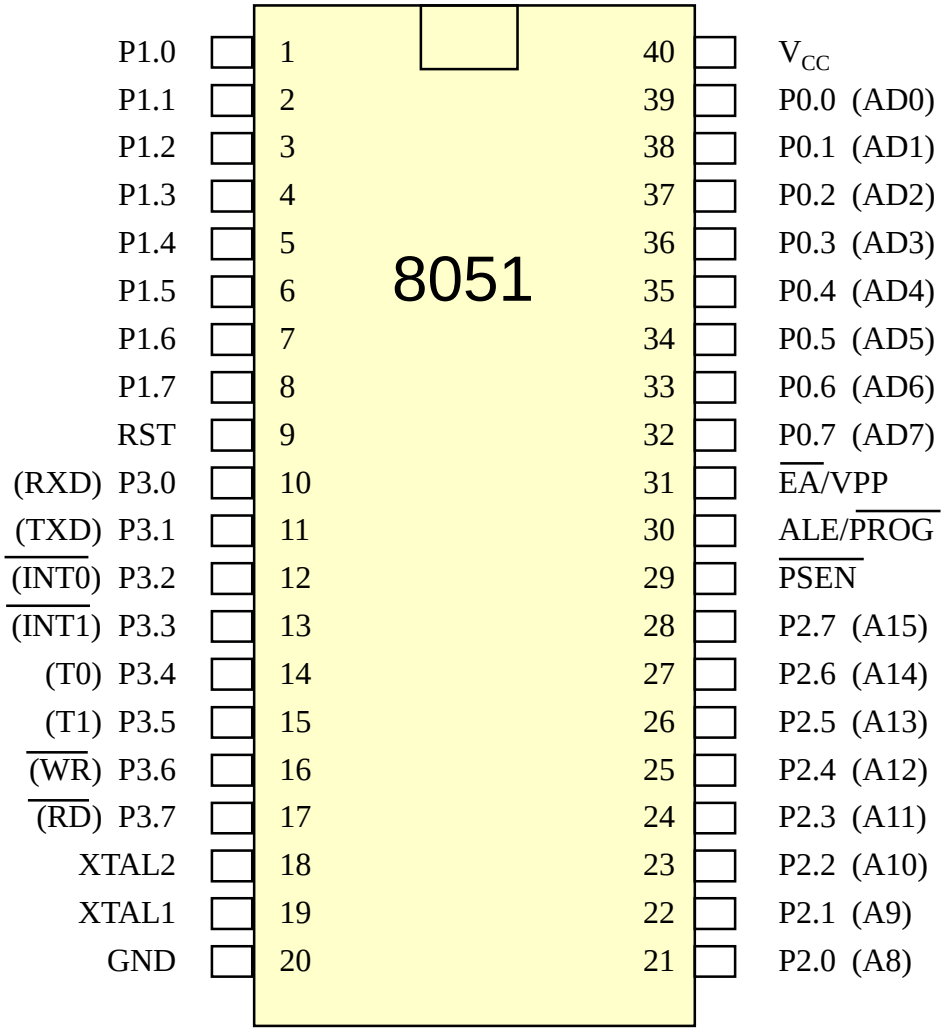
- Memory of 8051 can be increased externally:
  - Increase memory space for codes (programs) up to 64K
  - Increase memory space for data up to 64K
- Boolean instructions work with 1 bit at a time
- Assume clock frequency = 12MHz, it takes about 4 μs (i.e. 4 x 10^{-6}s) to carry out a 8-bit multiplication instruction

The 8051 Block Diagram

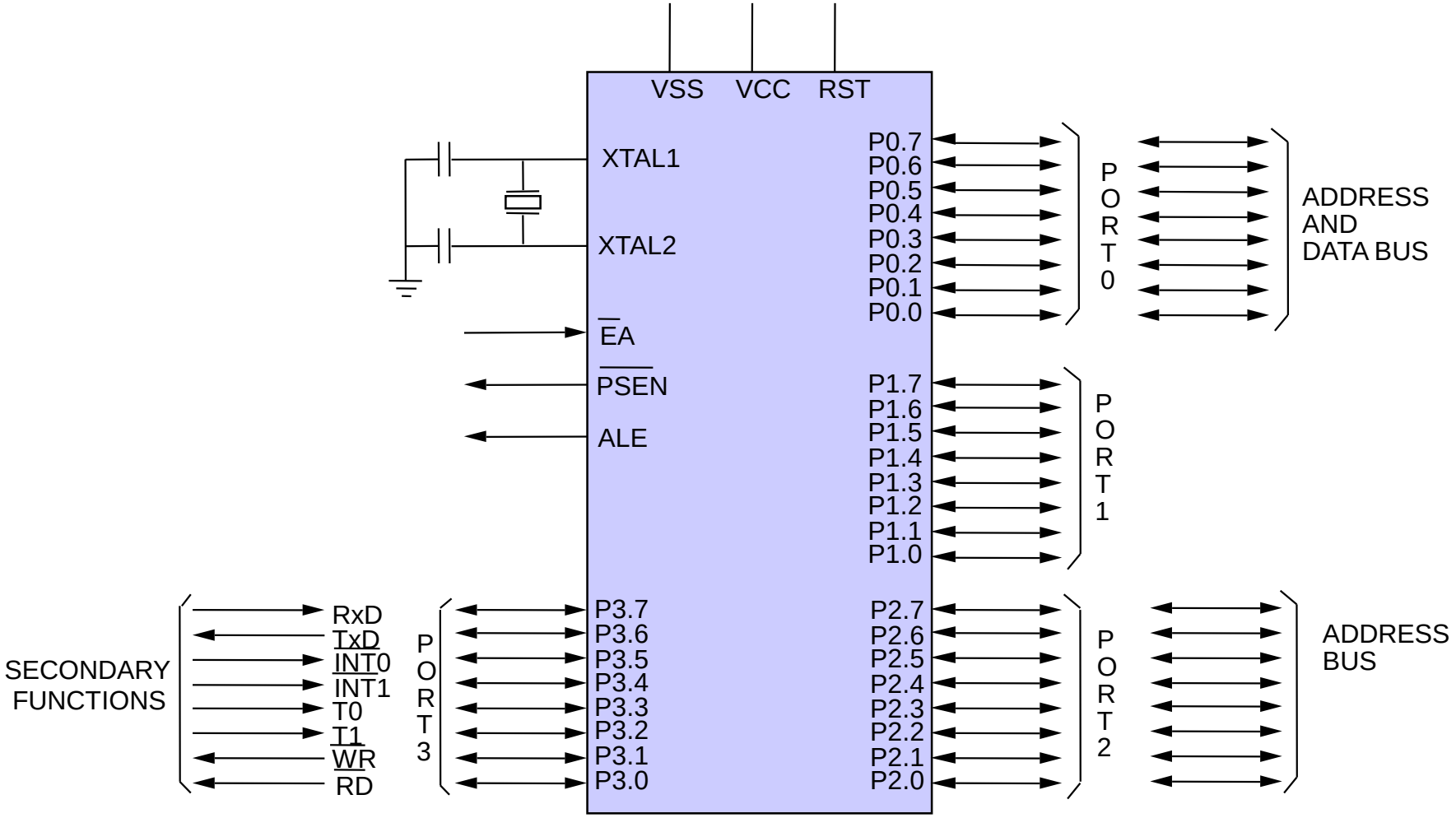



Arithmetic and Logic Unit
PSW
A
B
PC
DPTR DPH DPL
8-Bit Data and Address Bus
16-Bit Address Bus
Special Function Registers RAM
ROM
Latch
Port 0
I / O
AD7–AD0
Port 1
I / O
Port 2
I / O
A15–A8
Port 3
I / O
Interrupt (INT0, INT1)
Counter (T0, T1)
Serial Data
(TXD, RXD)
RD, WR
EA
ALE
PSEN
XTAL1
XTAL2
RESET
Vcc
GND
System Timing
System Interrupts Timers
Data Buffers Memory Control
Byte/Bit Addresses
Register Bank 3
Register Bank 2
Register Bank 1
Register Bank 0
Special Function Register
IE
IP
PCON
SBUF
SCON
TCON
TMOD
TL0
TH0
TL1
TH1
Internal RAM Structure

The 8051 Pin Assignments



The 8051 Logic Symbol



Hardware Description

1. Timing diagram and T states
2. Program counter (PC)
3. Data pointer (DPTR)
4. Accumulator (“A”) register
5. B register
6. Flags
7. Program status word (PSW)
8. Internal memory (ROM, RAM, additional memory)
9. Stack & stack pointer (SP)
10. Special function register (SFR)

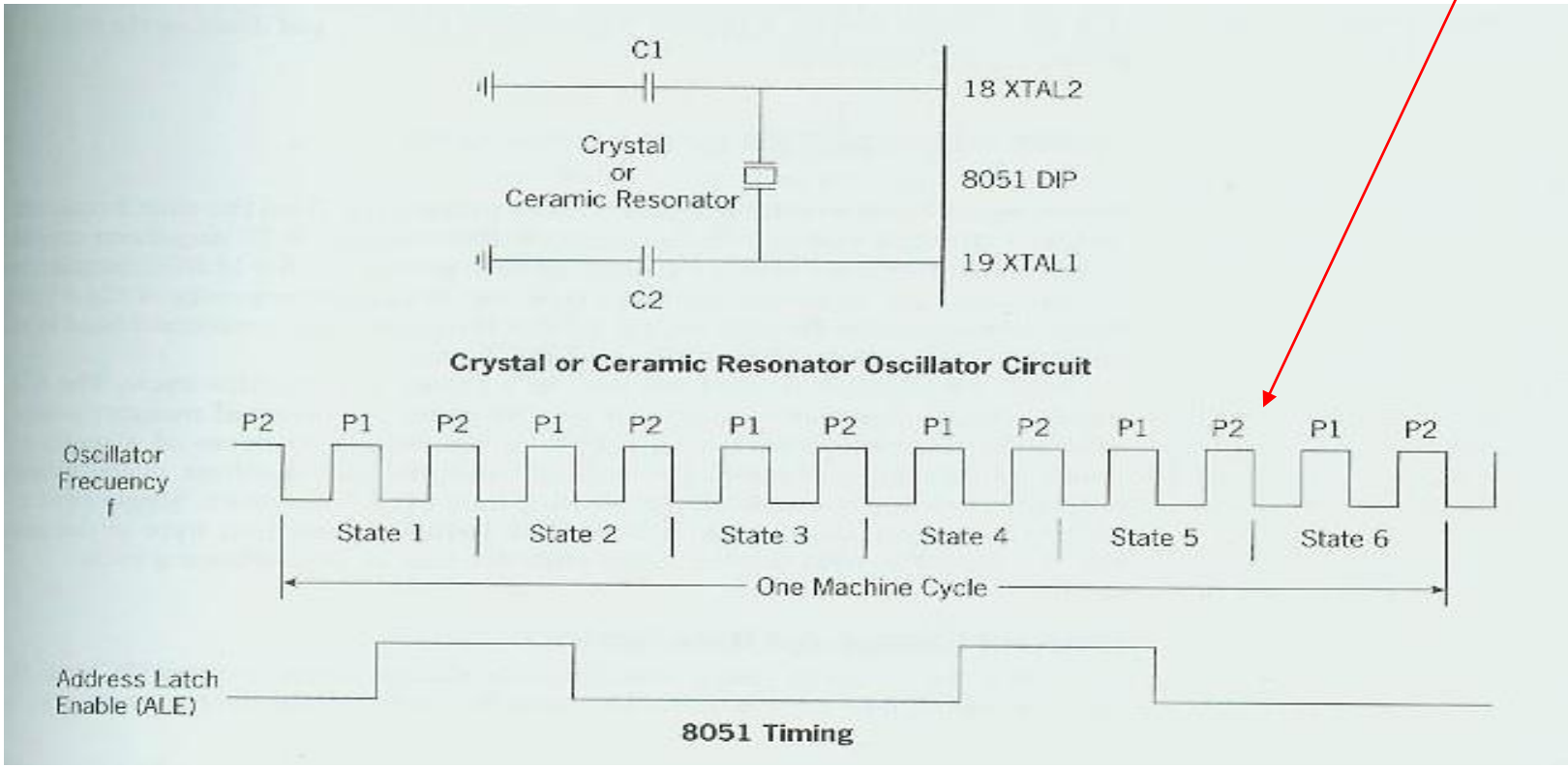
few new terms

- Timing diagram: representation of a set of signals in time domain.
- Machine cycle: the steps that are taken to accomplish operations such as memory read/write.
- An IC can have one or more MC
- A T-state is the time required to perform the smallest operation by the processor
- sequence of a few smallest operations will accomplish the execution of an instruction
- A few T-states are required to finish the execution of an instruction

Timing diagram and T states

- Oscillator circuit is the heart of 8051
- Produces clock pulses
- Synchronize all 8051's internal operations

A single machine cycle consists of 12 crystal pulses !



Machine Cycle

- Machine cycle is the basic repetitive process that the CPU performs once it is powered on. A machine cycle consists of a fixed number of clock cycles.
- The 8051 family needs 12 clock cycles for one machine cycle.
- The CPU takes one or more machine cycles to execute an instruction.
- The number of machine cycles of the 8051 instructions are ranging from 1 to 4.

Examples on machine cycle time

Find the elapse time of the machine cycle for:

(a) XTAL = 11.0592 MHz

(b) XTAL = 16 MHz

(c) XTAL = 20 MHz

Solution:

(a) 11.0592 MHz / 12 = 921.6 kHz

Machine cycle = 1 / 921.6 kHz = 1.085 μs

(b) 16 MHz / 12 = 1.333 MHz

Machine cycle = 1 / 1.333 MHz = 0.75 μs

(c) 20 MHz / 12 = 1.667 MHz

Machine cycle = 1 / 1.667 MHz = 0.60 μs

Example

- How much time will be taken by 8051 to execute MOV instruction, given that it takes 2 cycles to execute? Assume 16 MHz crystal frequency.
- Using 16 MHz, one machine cycle will be 0.75 micro seconds.
- Therefore, 1.5 microseconds will be taken for execution of MOV instruction.
- instruction’s information about its size and number of cycles to execute will be mentioned in data sheet of the processor.

Program Counter (PC)

- ***PC*** is a 16-bit register
- ***PC*** is the only register that does not have an internal address
- Holds the address of the memory location to fetch the program instruction
- Program ROM may be on the chip at addresses 0000H to 0FFFH (4Kbytes), external to the chip for addresses that exceed 0FFFH
- Program ROM may be totally external for all addresses from 0000H to FFFFH
- ***PC*** is automatically incremented (+1) after every instruction byte is fetched

Data Pointer (DPTR)

- ***DPTR*** is a 16-bit register
- ***DPTR*** is made up of two 8-bit registers: ***DPH*** and ***DPL***
- ***DPTR*** holds the memory addresses for internal and external code access and external data access

(eg. MOVC A,@A+DPTR MOVX A,@DPTR MOVX @DPTR,A)

- ***DPTR*** is under the control of program instructions and can be specified by its 16-bit name, or by each individual byte name, ***DPH*** and ***DPL***
- ***DPTR*** does not have a single internal address; ***DPH*** and ***DPL*** are each assigned an address (83H and 82H)

Accumulator (Register A) 0xE0

- Most versatile CPU register and is used for many operations, including addition, integer multiplication and division, and Boolean bit manipulations
- register ***A*** is also used for all data transfer between the 8051 and any external memory

Register B 0xF0

- register ***B*** is used with the register ***A*** for multiplication and division operations

  (eg. MUL AB DIV AB)

- No other special function other than as a location where data may be stored

Flags

- Flags are ***1-bit registers*** provided to store the results of certain program instructions
- Other instructions can test the condition of the flags and make decisions based on the flag states
- Flags are grouped inside the ***program status word (PSW)*** and the ***power control (PCON)*** registers for convenient addressing
- **Math flags**: indicate the outcomes of math operations (***CY, AC, OV, P***)
- **User flags**: general-purpose flags that may be used by the programmer to record some event in the program (***F0, GF0, GF1***).
    - user defines the function of this flag. User can set, test and clear this flag using instructions.

Program Status Word (PSW) 0xD0

PSW contains;

- the math flags
- user program flag ***F0***
- the register select bits (RS1, RS0)
  - RS bits identify which of the four general-purpose register banks is currently in use by the program

7	6	5	4	3	2	1	0
CY	AC	F0	RS1	RS0	OV	--	P

Program Status Word (PSW) 0xD0

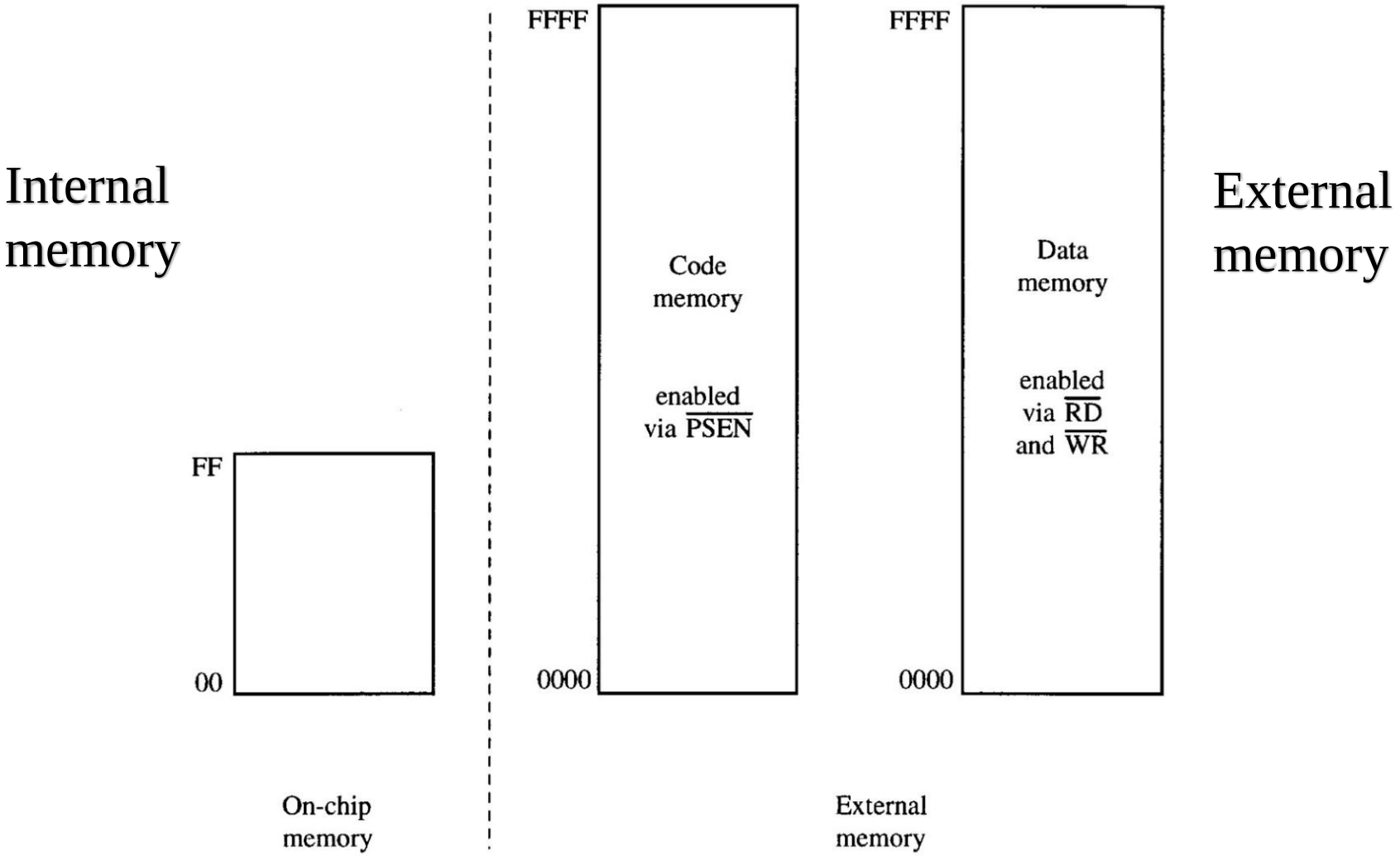
Bit	Symbol	Function
7	CY	Carry Flag; used in arithmetic, JUMP, ROTATE, and BOOLEAN instruction
6	AC	Auxiliary carry flag; used for BCD arithmetic
5	F0	User flag 0
4	RS1	Register bank select bit 1
3	RS0	Register bank select bit 0
2	OV	Overflow flag; used in arithmetic instructions
1	--	Reserved for future use
0	P	Parity flag; shows parity of register A: 1 = Odd Parity

Register values after Reset of 8051

Register	Reset Value
PC	0000
ACC	0000
B	0000
PSW	0000
SP	0007
DPTR	0000

RAM are all zero.

8051 memory organization



Internal RAM

Bit addressable

Byte Address	Bit Address							
7F	General Purpose RAM							
30								
2F	7F	7E	7D	7C	7B	7A	79	78
2E	77	76	75	74	73	72	71	70
2D	6F	6E	6D	6C	6B	6A	69	68
2C	67	66	65	64	63	62	61	60
2B	5F	5E	5D	5C	5B	5A	59	58
2A	57	56	55	54	53	52	51	50
29	4F	4E	4D	4C	4B	4A	49	48
28	47	46	45	44	43	42	41	40
27	3F	3E	3D	3C	3B	3A	39	38
26	37	36	35	34	33	32	31	30
25	2F	2E	2D	2C	2B	2A	29	28
24	27	26	25	24	23	22	21	20
23	1F	1E	1D	1C	1B	1A	19	18
22	17	16	15	14	13	12	11	10
21	0F	0E	0D	0C	0B	0A	09	08
20	07	06	05	04	03	02	01	00
1F	Bank 3							
18								
17	Bank 2							
10								
0F	Bank 1							
08								
07	Default Register Bank for R0 – R7							
00								

Bit Addressable

Byte Address	Bit Address								
FF									
F0	F7	F6	F5	F4	F3	F2	F1	F0	B
E0	E7	E6	E5	E4	E3	E2	E1	E0	ACC
D0	D7	D6	D5	D4	D3	D2	-	D0	PSW
B8	-	-	-	BC	BB	BA	B9	B8	IP
B0	B7	B6	B5	B4	B3	B2	B1	B0	P3
A8	AF	-	-	AC	AB	AA	A9	A8	IE
A0	A7	A6	A5	A4	A3	A2	A1	A0	P2
99	Not bit-addressable								SBUF
98	9F	96	95	94	93	92	91	90	SCON
90	97	96	95	94	93	92	91	90	P1
8D	Not bit-addressable								TH1
8C	Not bit-addressable								TH0
8B	Not bit-addressable								TL1
8A	Not bit-addressable								TL0
89	Not bit-addressable								TMOD
88	8F	8E	8D	8C	8B	8A	89	88	TCON
87	Not bit-addressable								PCON
83	Not bit-addressable								DPH
82	Not bit-addressable								DPL
81	Not bit-addressable								SP
80	87	86	85	84	83	82	81	80	P0

Internal Memory

- A functioning computer must have memory for program code bytes, commonly in ROM, and RAM memory for variable data that can be altered as the program runs
- 8051 has internal RAM (128 bytes) and ROM (4Kbytes)
- 8051 uses the same address but in different memories for code and data
- Internal circuitry access the correct memory based on the nature of the operation in progress

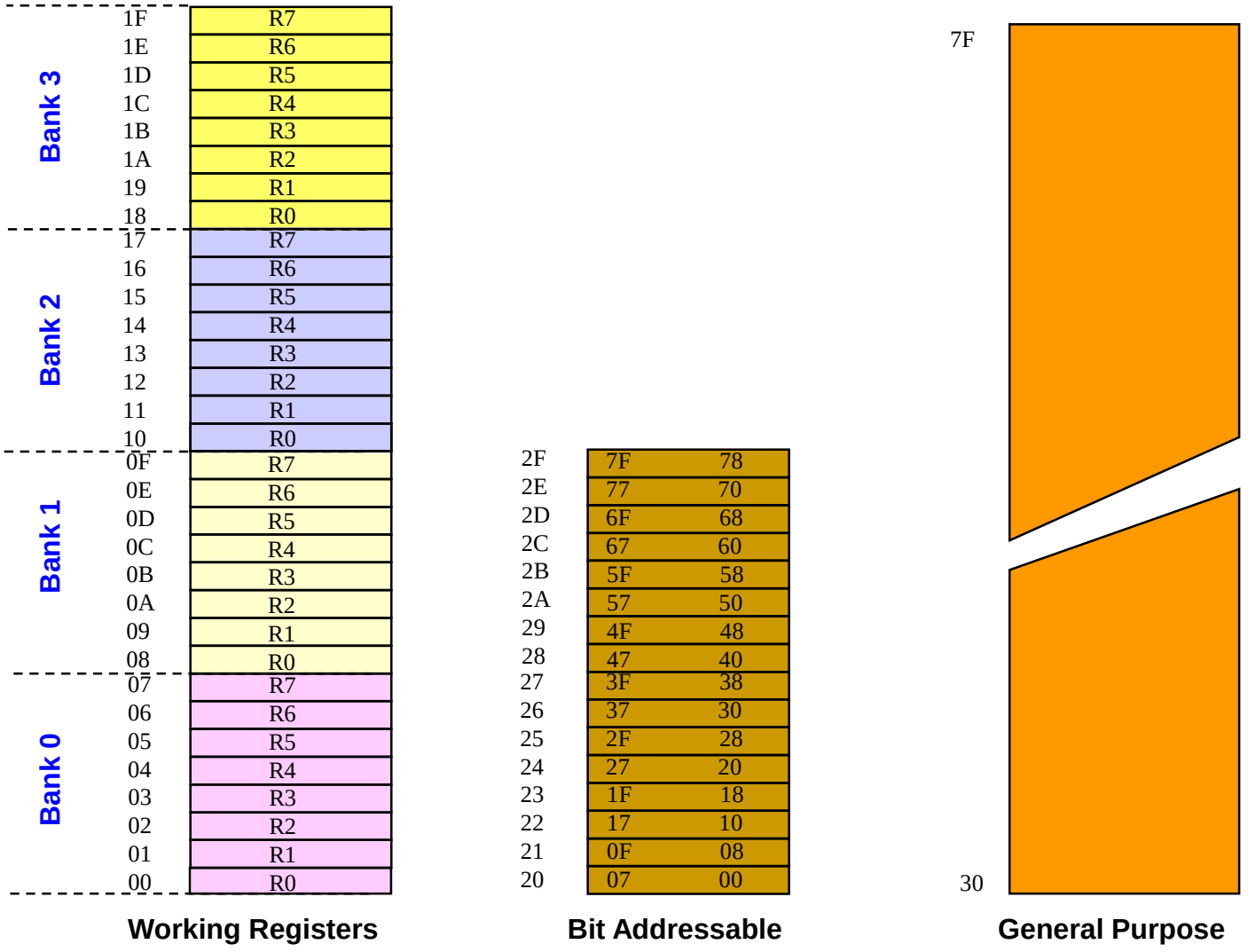
Register Banks

- Four banks of 8 byte-sized registers, **R0** to **R7**
- Addresses are :

18 - 1F	for bank 3
10 - 17	for bank 2
08 - 0F	for bank 1
00 - 07	for bank 0 (default)

- Active bank selected by bits [**RS1**, **RS0**] in PSW.
- supports easy and quick "context switching" between subroutines

8051 Internal RAM Organisation



Program Storage

- After reset, the MCS-51 starts fetching instructions from 0000H.
- This can be either on-chip or external depending on the value of the $\overline{EA}$ input pin.
  - If $\overline{EA}$ is low, then the program memory is external.
  - If $\overline{EA}$ is high, then addresses from 0000H to 0FFFH will refer to on-chip memory and addresses from 1000H up to FFFFH refer to external memory.

Bit addressable feature

- First bit has the same address as the register.
- Examples:
  - CPL 2F.7 is same as CPL 7FH
  - CPL D0.7 is same as CPL D7H
  - Similarly, P1 has address 90H in the SFR, so, P1.7 or address 97H refer to the same bit.

  - Internal RAM

Stack and Stack Pointer (SP)

- ***SP*** is a 8-bit register used to hold an internal RAM address that is called the "***top of the stack***"
- ***Stack*** refers to an area of internal RAM that is used in conjunction with certain opcodes to store and retrieve data quickly
- ***SP*** holds the internal RAM address where the last byte of data was stored by a stack operation
- When data is to be placed on the stack, the ***SP*** increments before storing data on the stack so that the stack ***grows up*** as data is stored
- As data is retrieved from the stack, the byte is read from the stack, and then the ***SP*** decrements to point to the next available byte of stored data
- ***SP*** = 07H after reset

Special Function Registers (SFR)

- 8051 has 21 ***SFR***s which occupy the addresses from 80H to FFH (128bytes)
- Not all of the addresses from 80H to FFH are used for ***SFR***s
- Attempt to use the “empty” addresses may get unpredictable result
- Internal RAM

Internal ROM

- Internal ROM occupies the code address space from 0000H to 0FFFH (Size = 4K byte)
- Program addresses higher than 0FFFH will automatically fetch code bytes from external program memory
- Code bytes can also be fetched exclusively from an external memory by connecting the external access pin ($\overline{EA}$) to ground

Some Important Pins

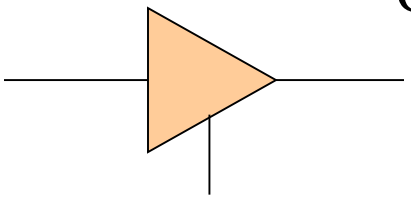
- VCC (pin 40 - provides supply voltage of +5V)
- GND (pin 20)
- XTAL1 & XTAL2 (pins 19 & 18 - to crystal and then caps)
- RST (pin 9- reset)
- EA (pin 31 - external access)
- PSEN (pin 29 - program store enable)
- ALE (pin 30 - address latch enable)
- Ports 0-3

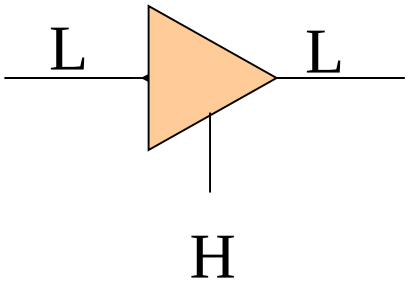
I/O Ports (P0 - P3)

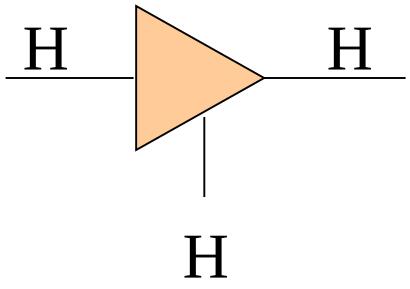
One of the most useful features of the 8051 is that it consists of 4 I/O ports (P0 - P3)

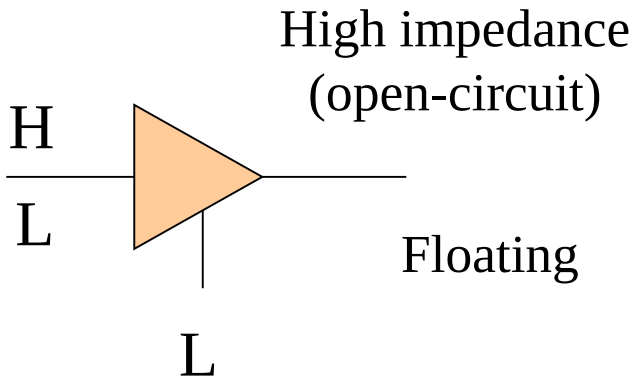
- All ports can be configured either as input or output port
- All ports have multiple functions (except P1)
- All ports are bit addressable
- On **RESET** all the ports are configured as ***output***
- When a port's pin need to be used as an ***input***, a "**1**" ***must be*** written to the corresponding latch by the program to configure it as input (eg. MOV P1, #0FFH)

Tri-state Buffer







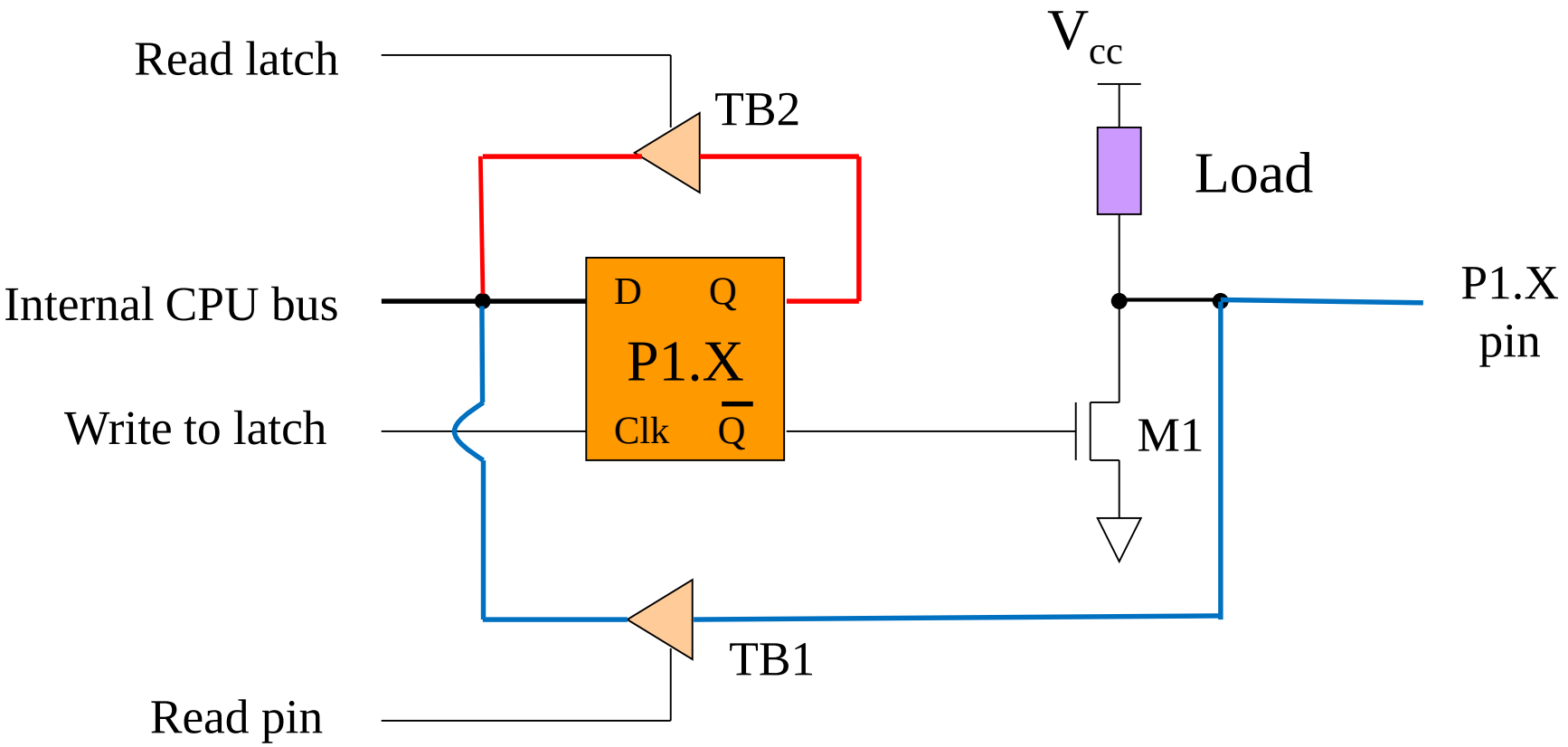


Hardware Structure of I/O Pin

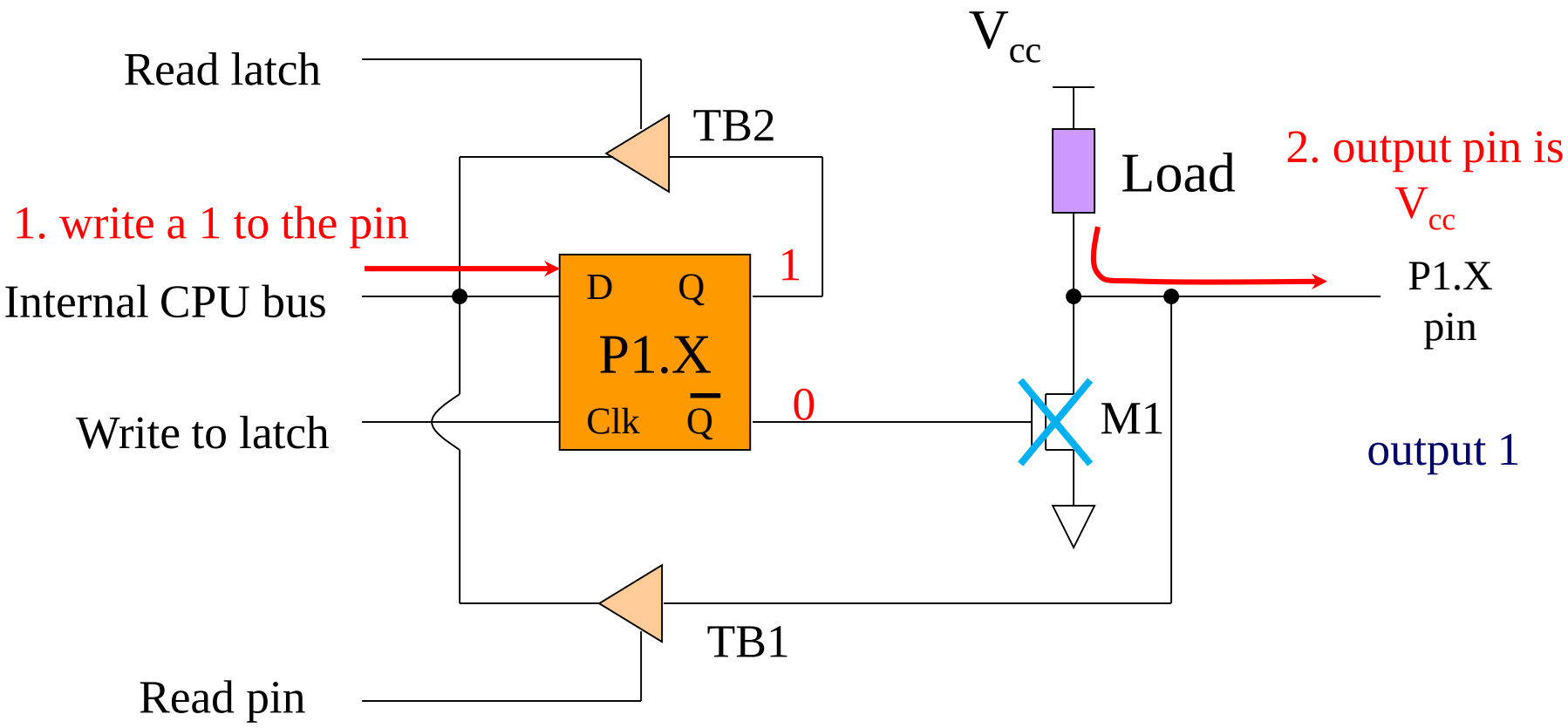
Each pin of I/O ports:

- Internal CPU bus：communicate with CPU
- D latch to store the value (0 or 1) of the pin
  - D latch is controlled by “Write to latch” signal
- Two Tri-state buffer:
  - TB1: controlled by “Read pin”
    - Read pin＝1：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

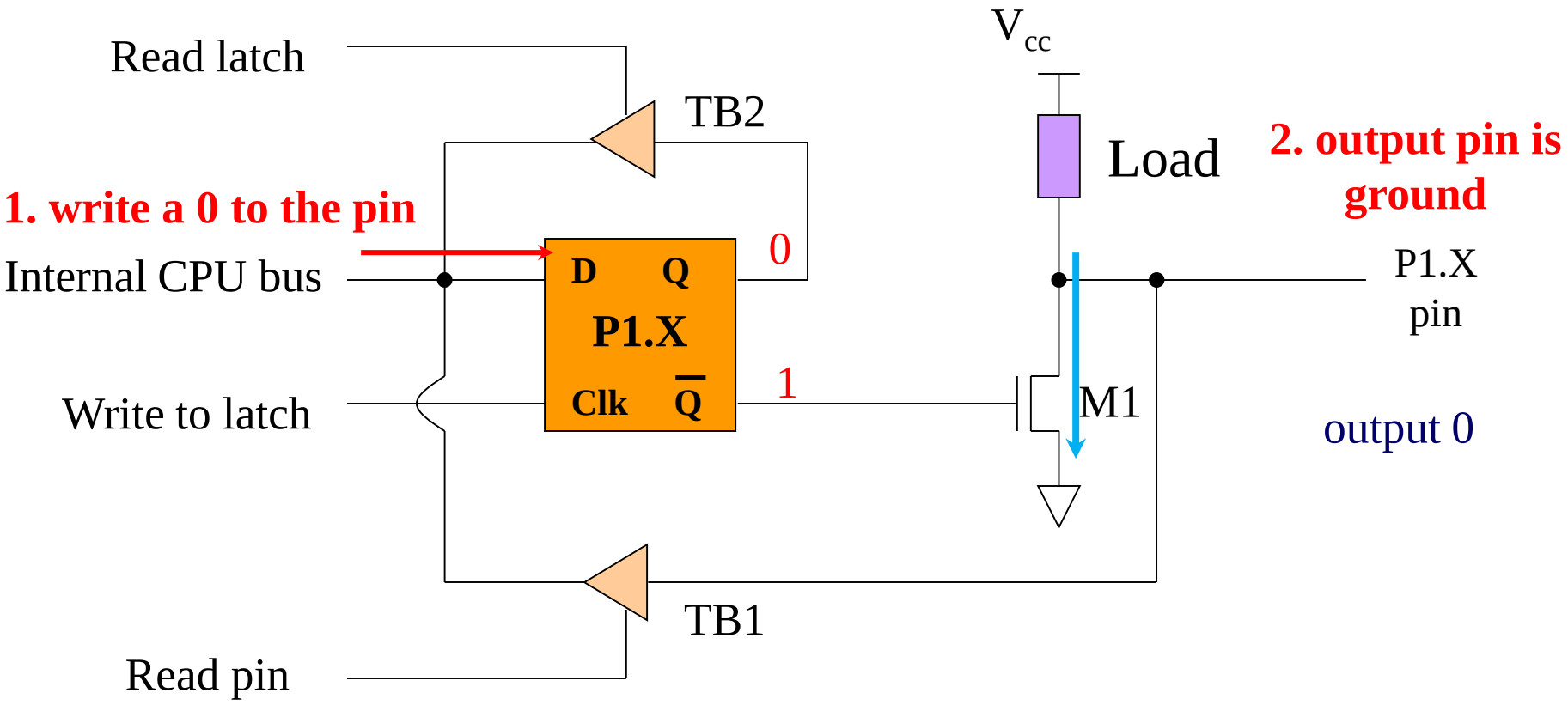
Pin of Port 1


Read latch
TB2
Vcc
Load
Internal CPU bus
D
Q
P1.X
P1.X pin
Write to latch
Clk
Q
M1
TB1
Read pin

Writing “1” to Output Pin P1.X



Writing “0” to Output Pin P1.X



Port as Output（Writing to port）

- Send data to Port 1：

```
        MOV     A , #55H
BACK:   MOV     P1 , A
        ACALL   DELAY
        CPL   A
        SJMP  BACK
```

  - Let P1 toggle.
  - You can write to P1 directly.

Reading from ports

When reading ports, there are two possibilities

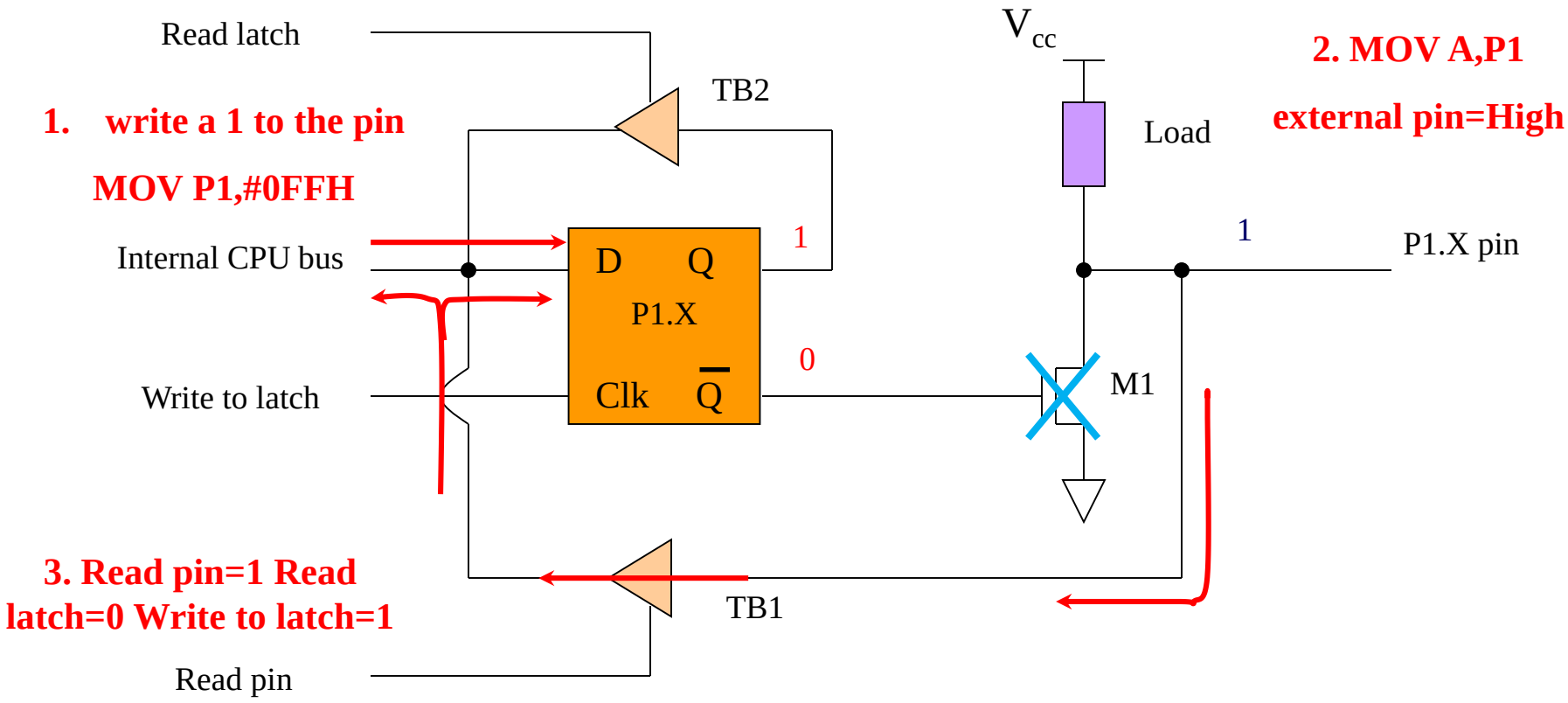
Read the status of the *input pin*. (from *external pin value*)

```
MOV  A, PX
JNB  P2.1, TARGET   ; jump if P2.1
                    ; is not set
JB   P2.1, TARGET   ; jump if P2.1
                    ; is set
```

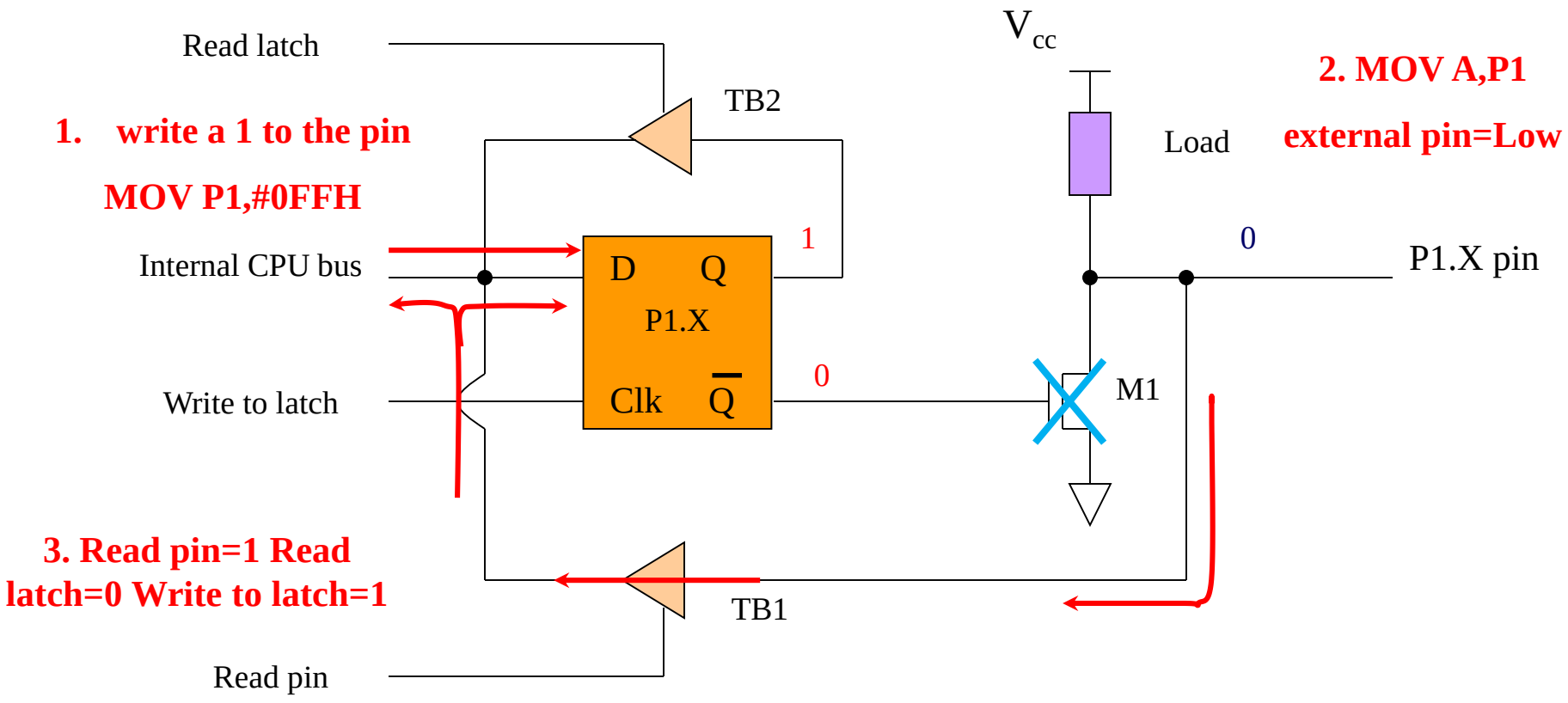
Read the *internal latch* of the *output port*: Read-Modify-Write.

```
ANL  P1, A      ; P1 ← P1 AND A
ORL  P1, A      ; P1 ← P1 OR A
INC  P1         ; increase P1
```

Reading “High” at Input Pin



Reading “Low” at Input Pin



Port as Input（Reading from Port）

- To make P1 as input port, configure it by writing 1 to all the 8 bits.

```
        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.

Instructions For Reading an Input Port

Following are instructions for reading external pins of ports:

Mnemonics	Examples	Description
MOV A,PX	**MOV A,P2**	Bring into A the data at P2 pins
JNB PX.Y,..	**JNB P2.1,TARGET**	Jump if pin P2.1 is low
JB PX.Y,..	**JB P1.3,TARGET**	Jump if pin P1.3 is high
MOV C,PX.Y	**MOV C,P2.4**	Copy status of pin P2.4 to CY

Read-Modify-Write Feature

- A method used to access the 8051 ports
- Combining all 3 actions in a single instructions :
  - Read the data at the port
  - Modify (do operation on) the data at the port
  - Write the results to the port

```
        MOV     P1, #55H
AGAIN:  XRL     P1, #0FFH
        ACALL   DELAY
        SJMP    AGAIN
```

Read-Modify-Write Instructions

Mnemonics	Example
ANL	ANL P1,A
ORL	ORL P1,A
XRL	XRL P1,A
JBC PX.Y, TARGET	JBC P1.1, TARGET
CPL	CPL P1.2
INC	INC P1
DEC	DEC P1
DJNZ PX, TARGET	DJNZ P1,TARGET
MOV PX.Y,C	MOV P1.2,C
CLR PX.Y	CLR P1.3
SETB PX.Y	SETB P1.4

Read-write-modifying port's latch

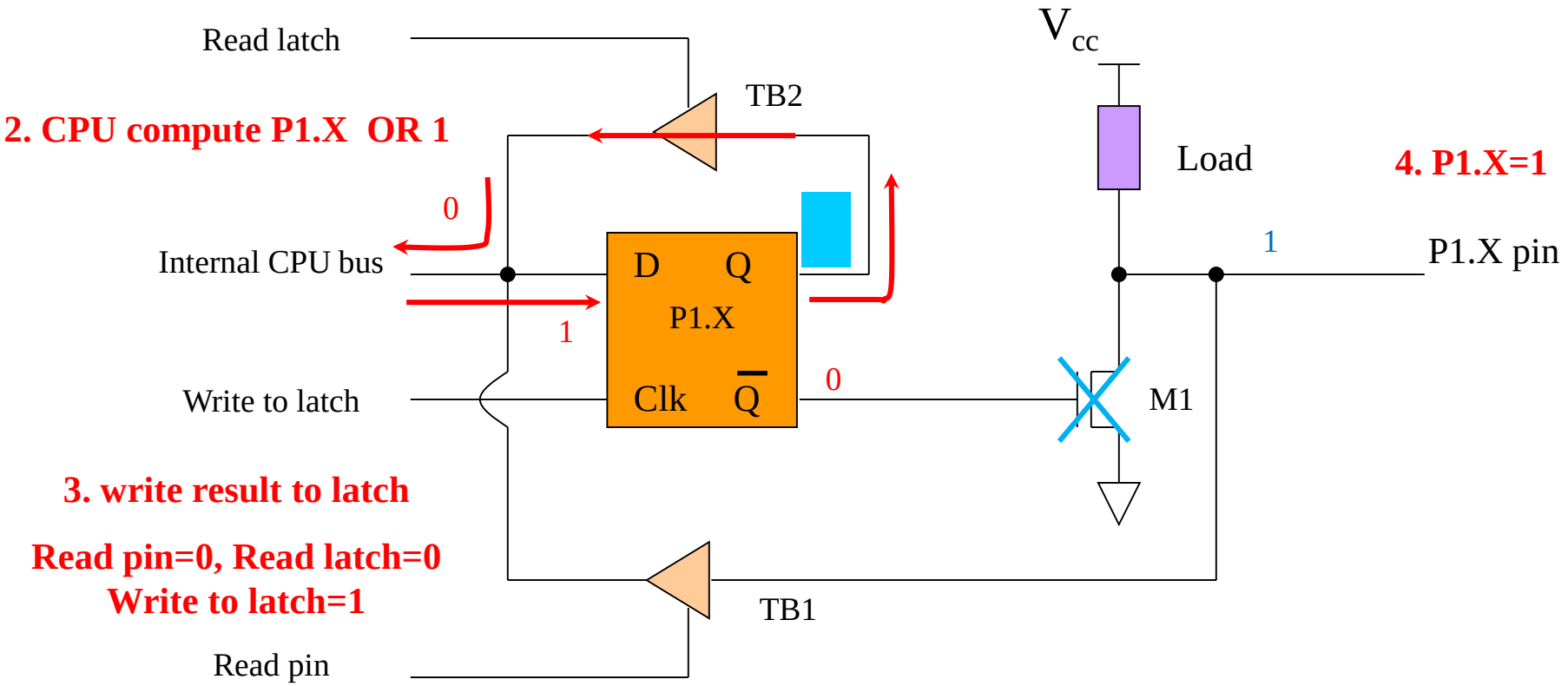
1. The read latch activates TB2 and bring the data from the Q latch into CPU.
   - Read P1.0=0
2. CPU performs an operation.
   - This data is ORed with bit 1 of register A. Get 1.
3. The latch is modified.
   - D latch of P1.0 has value 1.
4. The result is written to the external pin.
   - External pin (pin 1: P1.0) has value 1.

OR Port 1：

```
MOV   P1, #55H          ;P1=01010101
ORL   P1, #0F0H         ;P1=11110101
```

Read-write-modifying port's latch

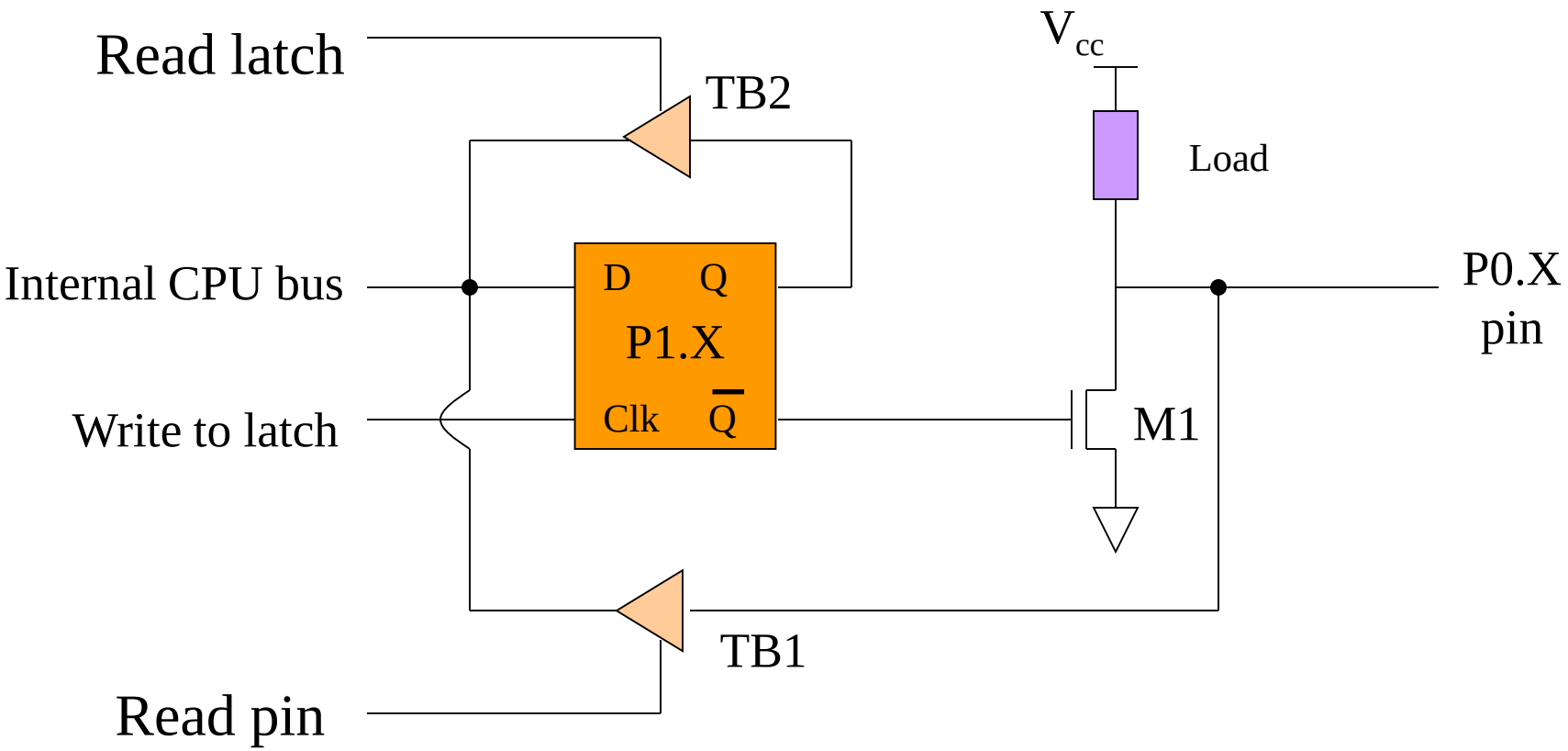
1. Read pin=0 Read latch=1 Write to latch=0 (Assume P1.X=0 initially)



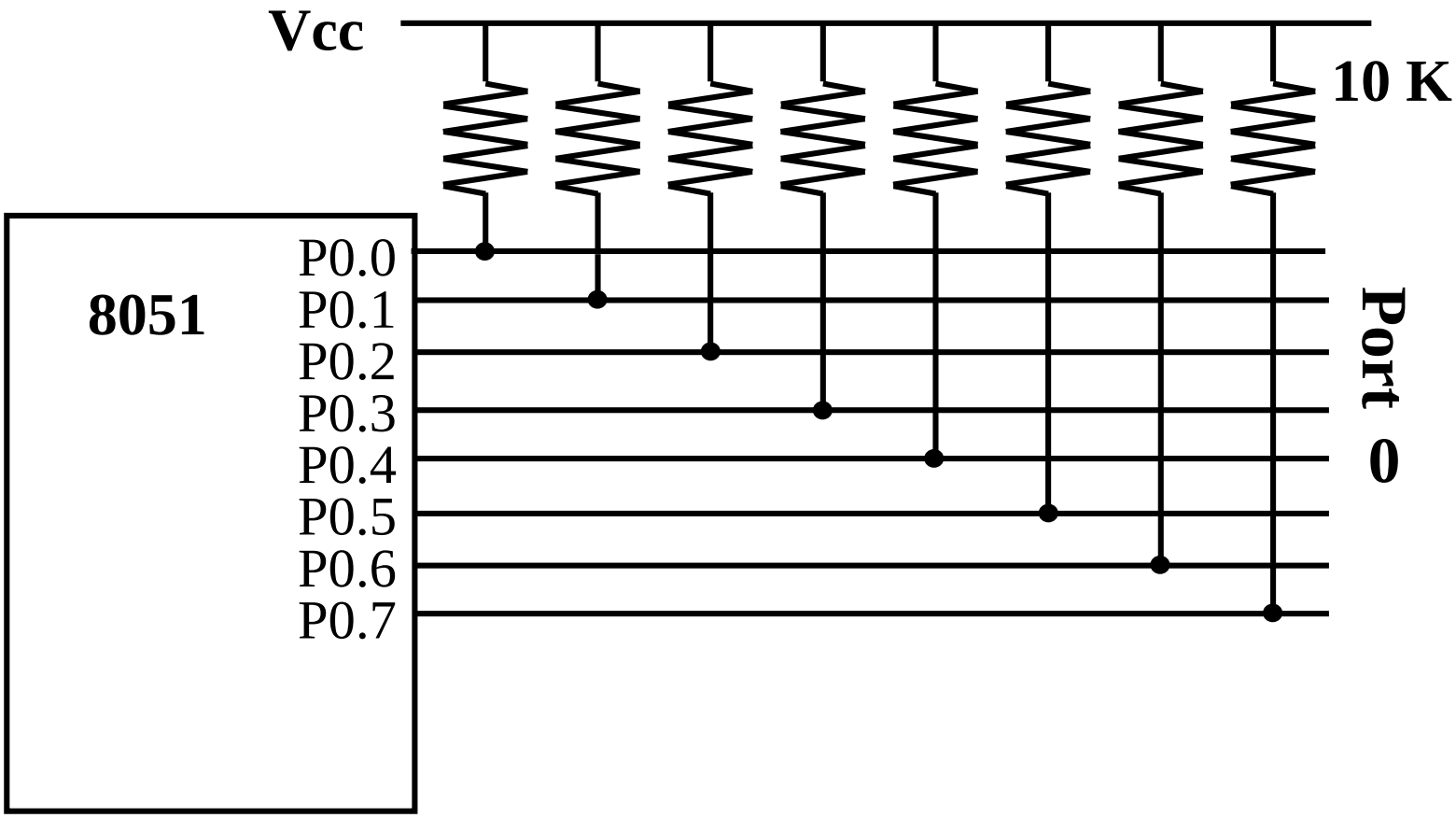
Port 0

- Occupies a total of 8 pins (Pins 32-39)
- Can be used for :
  - Input only
  - Output only
  - Input and output at the same time (i.e. some pins for input and the others for output)
- Can be used to handle both address and data
- Need pull-up resistors

Pin of Port 0


Read latch
TB2
V_{cc}
Load
Internal CPU bus
D
Q
P1.X
Clk
$\overline{Q}$
P0.X
pin
Write to latch
M1
TB1
Read pin

Port 0 with Pull-Up Resistors



Port 0 as an Output Port

The following code will continuously send out to port 0 the alternating values 55H and AAH

```
        MOV     A, #55H
BACK:   MOV     P0, A
        ACALL   DELAY
        CPL     A
        SJMP    BACK
```

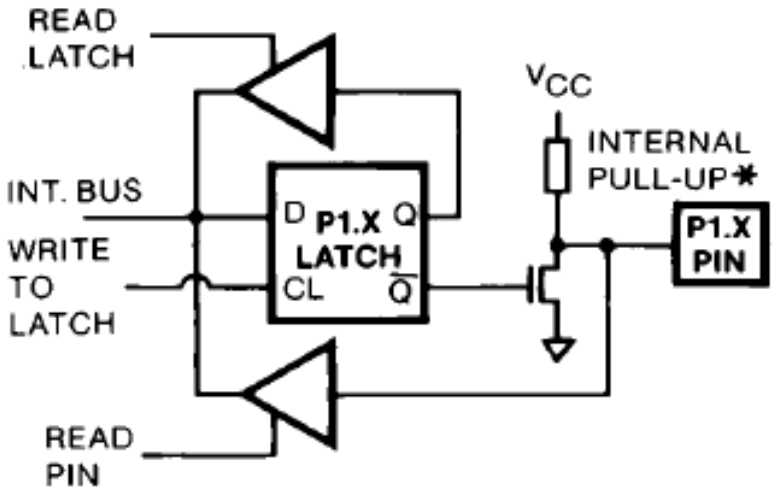
Port 0 as an Input Port

In the following code, port 0 is configured first as an input port by writing 1s to it; then data that is received from port 0 is sent to P1

```
        MOV     A, #0FFH
        MOV     P0, A
BACK:   MOV     A, P0
        MOV     P1, A
        SJMP    BACK
```

Port 1

- Occupies a total of 8 pins (Pins 1-8)
- Can be used as input or output
- Does not need any pull-up resistors
- Upon reset, port 1 is configured as an output port
- No alternative functions



Port 1 as an Output Port

The following code will continuously send out to port 1 the alternating values 55H and AAH

```
        MOV     A, #55H
BACK:   MOV     P1, A
        ACALL   DELAY
        CPL     A
        SJMP    BACK
```

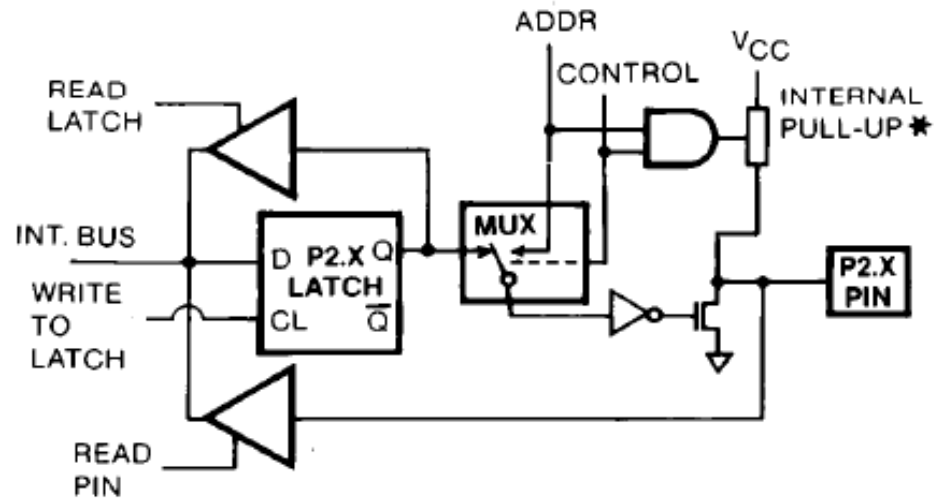
Port 1 as an Input Port

In the following code, port 1 is configured first as an input port by writing 1s to it, and then data is received from that port and saved in R7, R6, and R5

```
MOV     A, #0FFH
MOV     P1, A
MOV     A, P1
MOV     R7, A
ACALL   DELAY
MOV     A, P1
MOV     R6, A
ACALL   DELAY
MOV     A, P1
MOV     R5, A
```

Port 2

- Occupies a total of 8 pins (Pins 21-28)
- Similar function as Port 1
- Can be used as input or output
- Does not need any pull-up resistors
- Upon reset, port 2 is configured as an output port



Port 2 as an Output Port

The following code will continuously send out to port 2 the alternating values 55H and AAH

```
        MOV     A, #55H
BACK:   MOV     P2, A
        ACALL   DELAY
        CPL     A
        SJMP    BACK
```

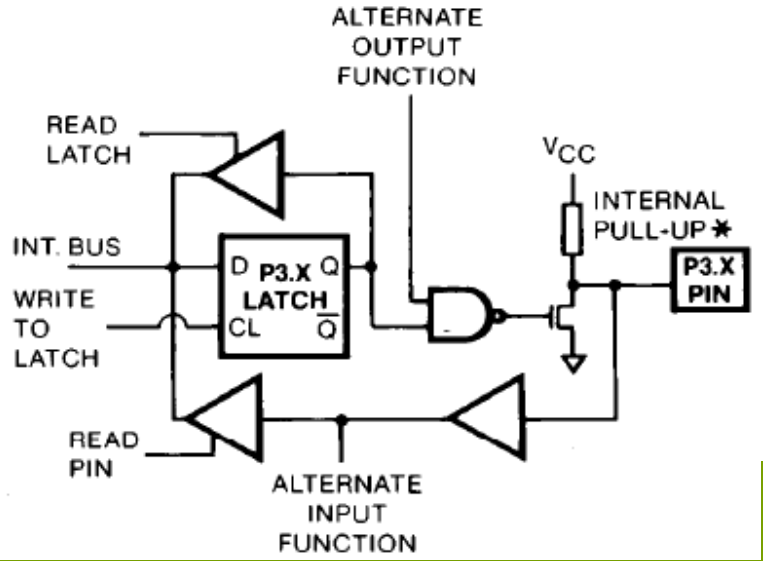
Port 2 as an Input Port

In the following code, port 2 is configured first as an input port by writing 1s to it, and then data is received from that port and sent to P1

```
        MOV     A, #0FFH
        MOV     P2, A
BACK:   MOV     A, P2
        MOV     P1, A
        SJMP    BACK
```

Port 3

- Occupies a total of 8 pins (Pins 10-17)
- Similar function as Port 1 and Port 2
- Can be used as input or output
- Does not need any pull-up resistors
- Upon reset, port 3 is configured as an output port
- Pins can be individually programmable for other uses
- Most commonly used to provide some important signals (e.g. interrupts)



Port 3 Alternate Functions

P3 Bit	Function	Pin
P3.0	RxD	Serial communication
P3.1	TxD	
P3.2	$\overline{\text{INT0}}$	External interrupt signals
P3.3	$\overline{\text{INT1}}$	
P3.4	T0	Counter pulse signal inputs
P3.5	T1	
P3.6	$\overline{\text{WR}}$	Memory control signals
P3.7	$\overline{\text{RD}}$	

Single-bit Addressability of Ports

- One of the most powerful features of the 8051
- Access only one or several bits of the port instead of the entire 8 bits

```
BACK:   CPL     P1.2
        ACALL   DELAY
        SJMP    BACK
```

Example

Write a program to perform the following:

(a) Keep monitoring the P1.2 bit until it becomes high;
(b) When P1.2 becomes high, write value 45H to port 0; and
(c) Send a high-to-low (H-to-L) pulse to P2.3

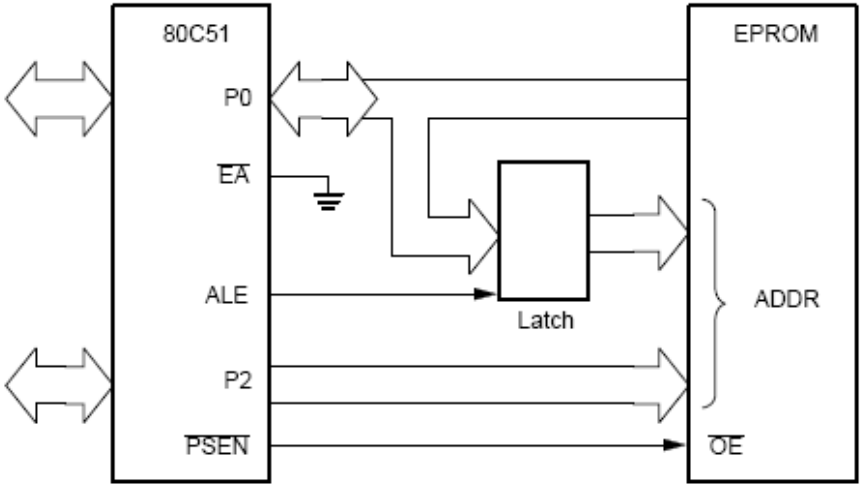
```
                SETB    P1.2            ; config pin P1.2 as input
                MOV     A, #45H
AGAIN:
                 JNB    P1.2, AGAIN     ; wait until P1.2=1
; now P1.2 = 1
                MOV     P0, A
                SETB    P2.3
                CLR     P2.3
```

Accessing the External Memory

- Port 0 acts as a multiplexed address/data bus. Sends the low byte of the address value on port 0 via a latch.
  - The signal ALE operates as to allow an external latch to store the low byte while the multiplexed bus is made ready to receive the byte from the external memory.
  - Port 0 then switches function and becomes the data bus receiving the byte from memory.
- Port 2 sends the high byte of the address value directly to the external memory.

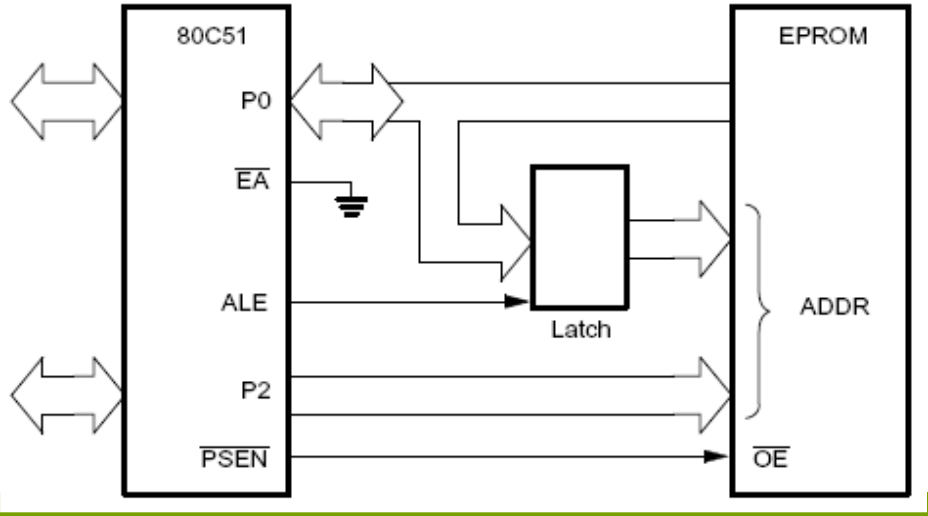
Role of Port 0 in accessing external memory

- When connecting 8051 to an external memory, port 0 provides both address and data (AD0 – AD7)
  - When ALE = 0, it provides data D0 – D7
  - When ALE = 1, it provides address A0 – A7
- ALE is used for demultiplexing address and data with the help of a 74LS373 latch

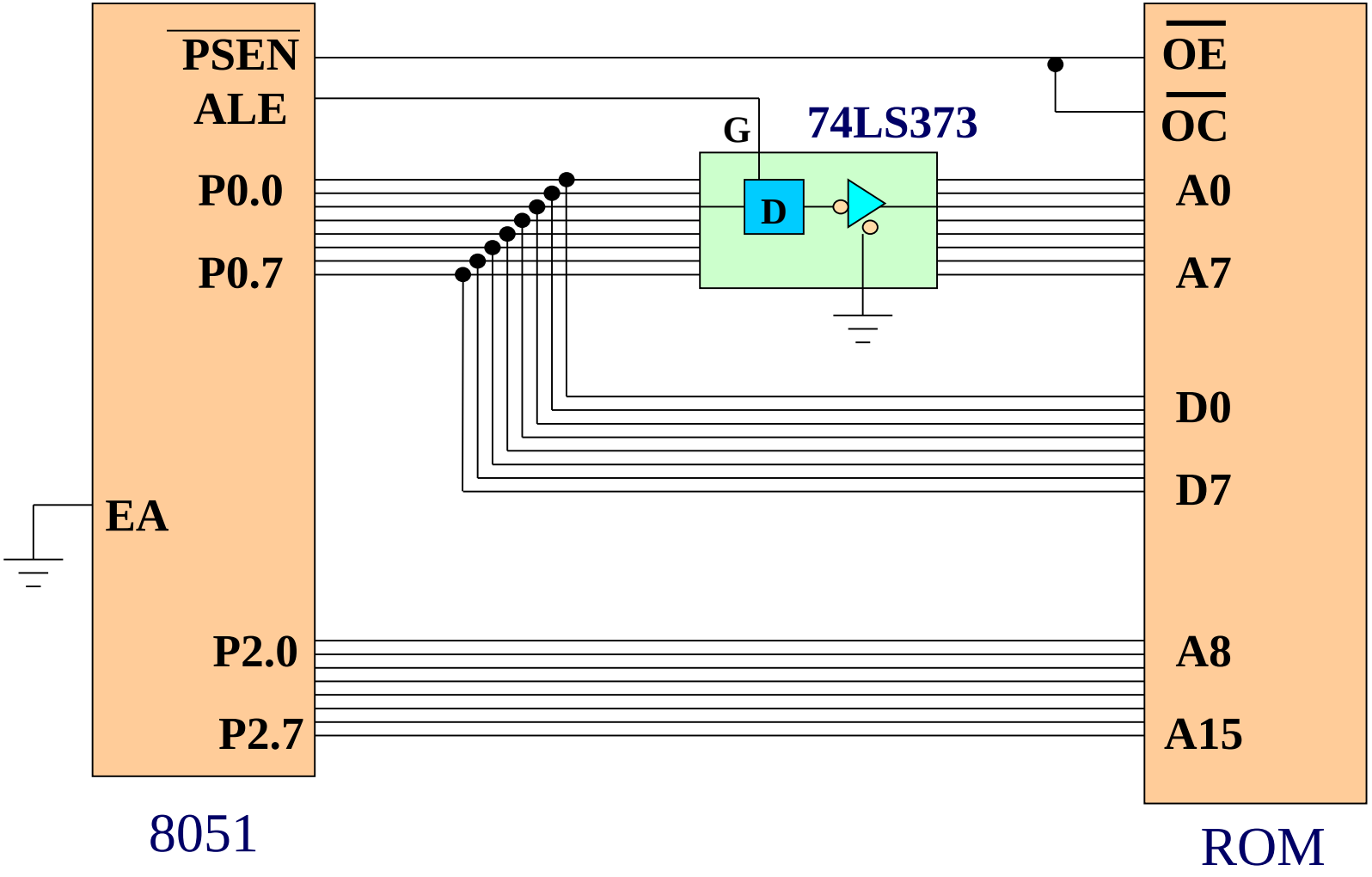


Role of Port 2 in accessing external memory

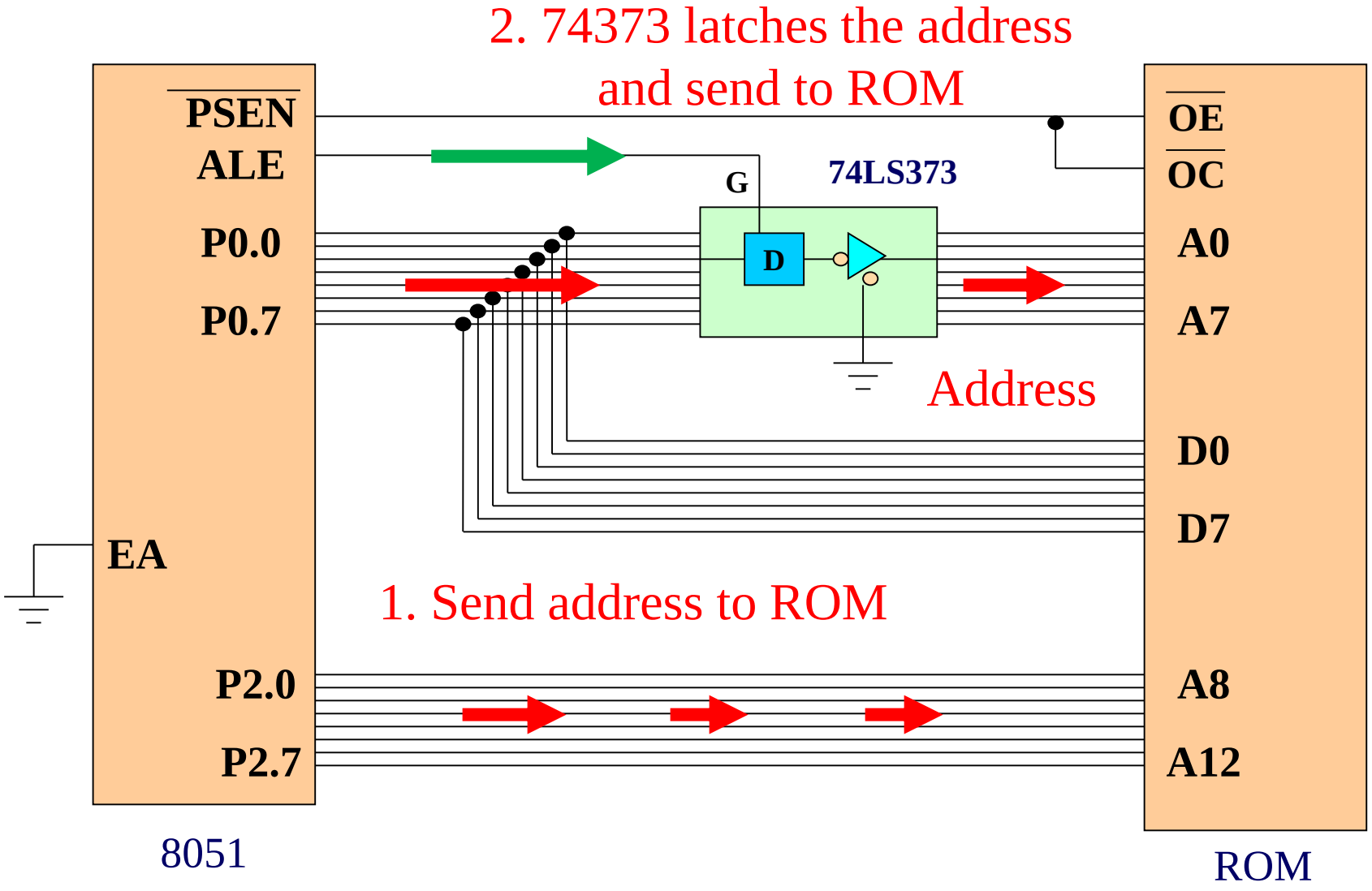
- When connecting an 8051 to an external memory, port 2 provides the higher byte of address (A8 – A15)
- It is used along with P0 to provide complete 16-bit address
- When P2 is used for the upper 8 bits of the 16-bit address, it cannot be used for I/O



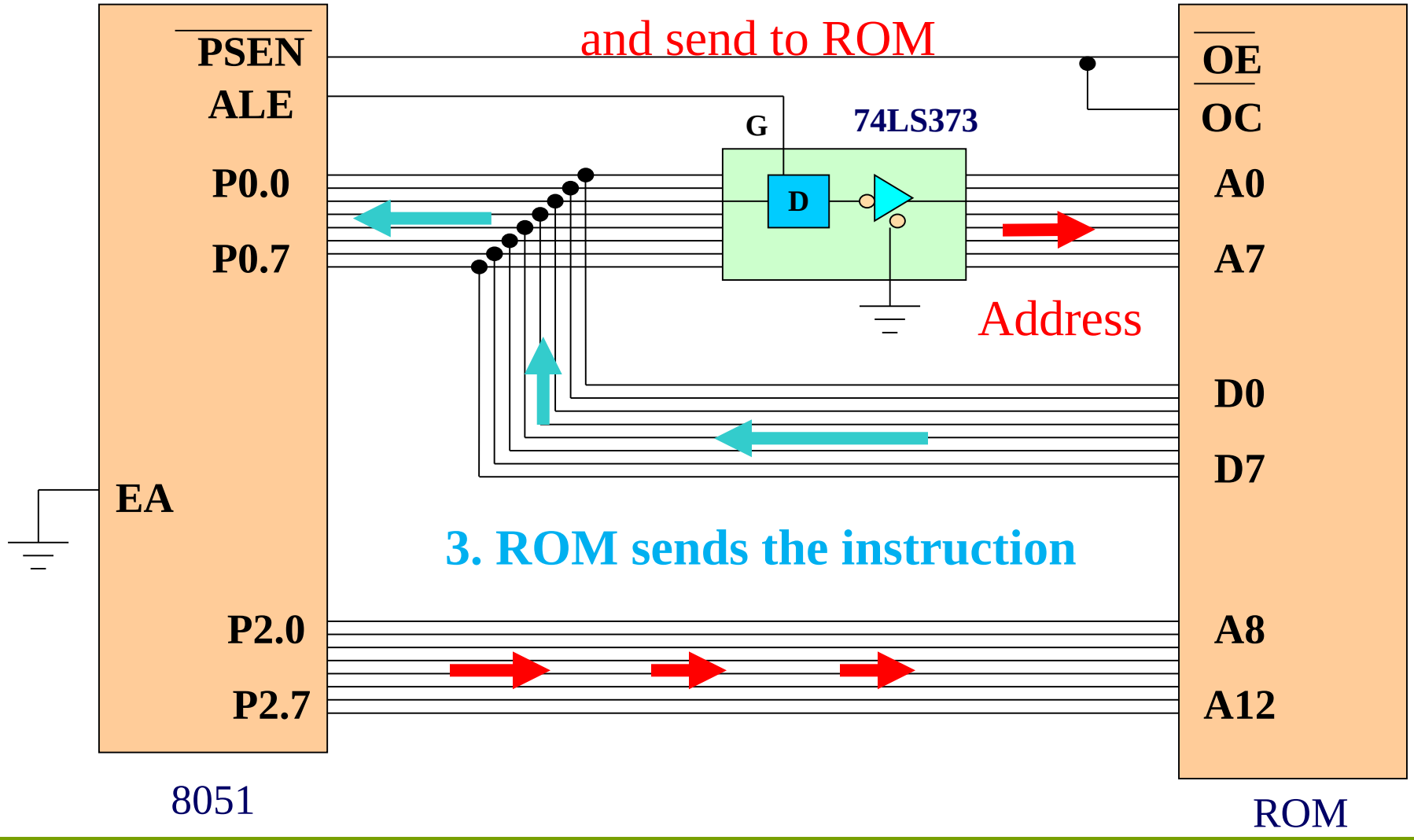
Demultiplexing address and data



Reading ROM (1/2)



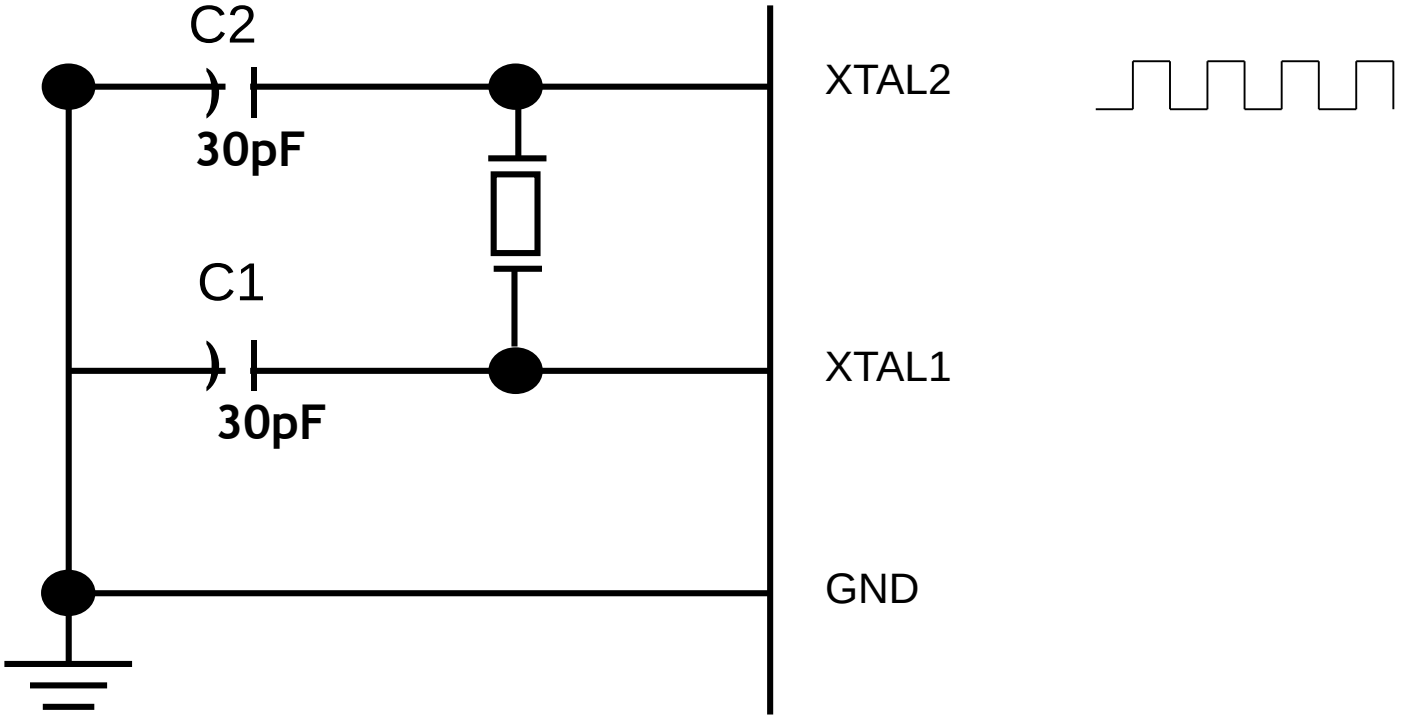
Reading ROM (2/2)



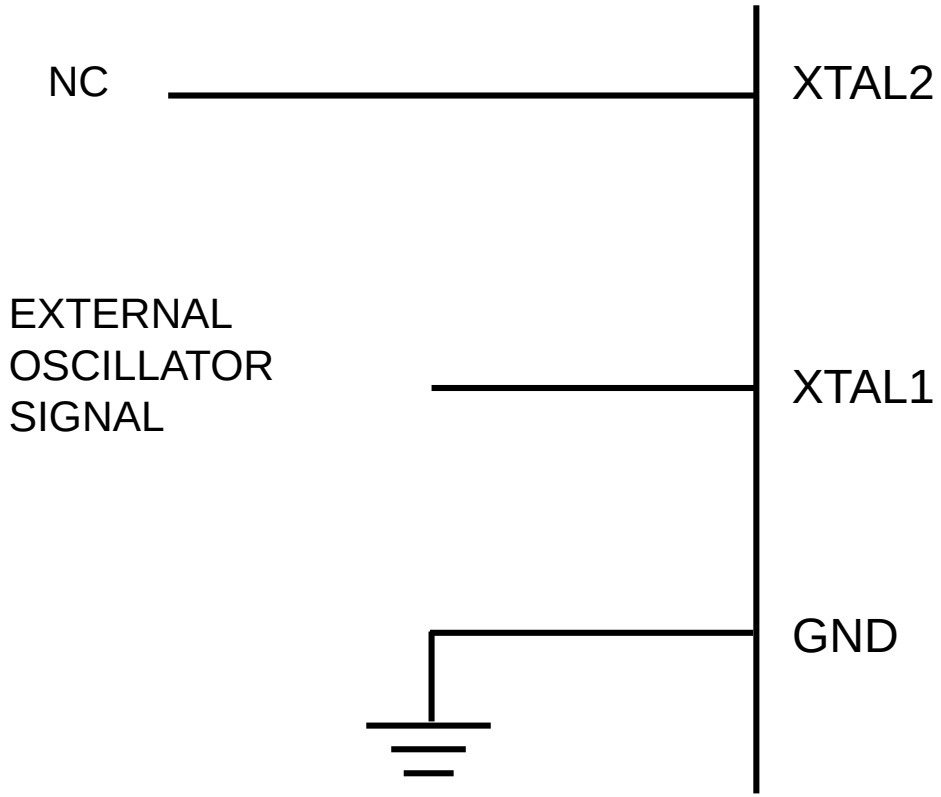
Few pins of 8051

- RST
  - Active high input signal
  - Should be high for at least 2 machine cycles
- XTAL-1 and XTAL-2
  - Can connect external crystal
  - Can use external oscillator signal source
    - XTAL-1 is connected to the clock signal
    - XTAL-2 is not connected

Clock signal generation using crystal



Clock signal generation using external oscillator signal



Time taken to execute an instruction

INSTRUCTION	MACHINE CYCLE
MOV R2,#55H	1
DEC R2	1
DJNZ R2,target	2
LJMP	2
SJMP	2
NOP	1
MUL AB	4
RET	2

Considering 11.0592 MHz crystal frequency

Finding the delay duration

```
MOV A,#55H
AGAIN: MOV P1,A
ACALL DELAY
CPL A
SJMP AGAIN
;----------------Time Delay
DELAY: MOV R3,#255
HERE: DJNZ R3,HERE
RET
```

Body of the loop
entry and exit instructions

2

How to increase this delay duration?

1. Using NOP instruction
2. Using loop inside loop

Getting more delay duration using NOP

DELAY: MOV R2,#255	M/C = 1
HERE: NOP	M/C = 1
NOP	M/C = 1
NOP	M/C = 1
NOP	M/C = 1
DJNZ R2,HERE	M/C = 2
RET	M/C = 2

(255 * (4+2) + 1 + 2) * 1.085 usec
= 1663.3 usec

Delay generation by using loop inside a loop

DELAY: MOV R2,#200	M/C = 1
AGAIN: MOV R3,#250	M/C = 1
HERE: NOP	M/C = 1
NOP	M/C = 1
DJNZ R3,HERE	M/C = 2
DJNZ R2,AGAIN	M/C = 2
RET	M/C = 2

{ 200 * { 1 + [250 * (2+2)] + 2 } + (1 + 2) } * 1.085 usec = 217.654 msec