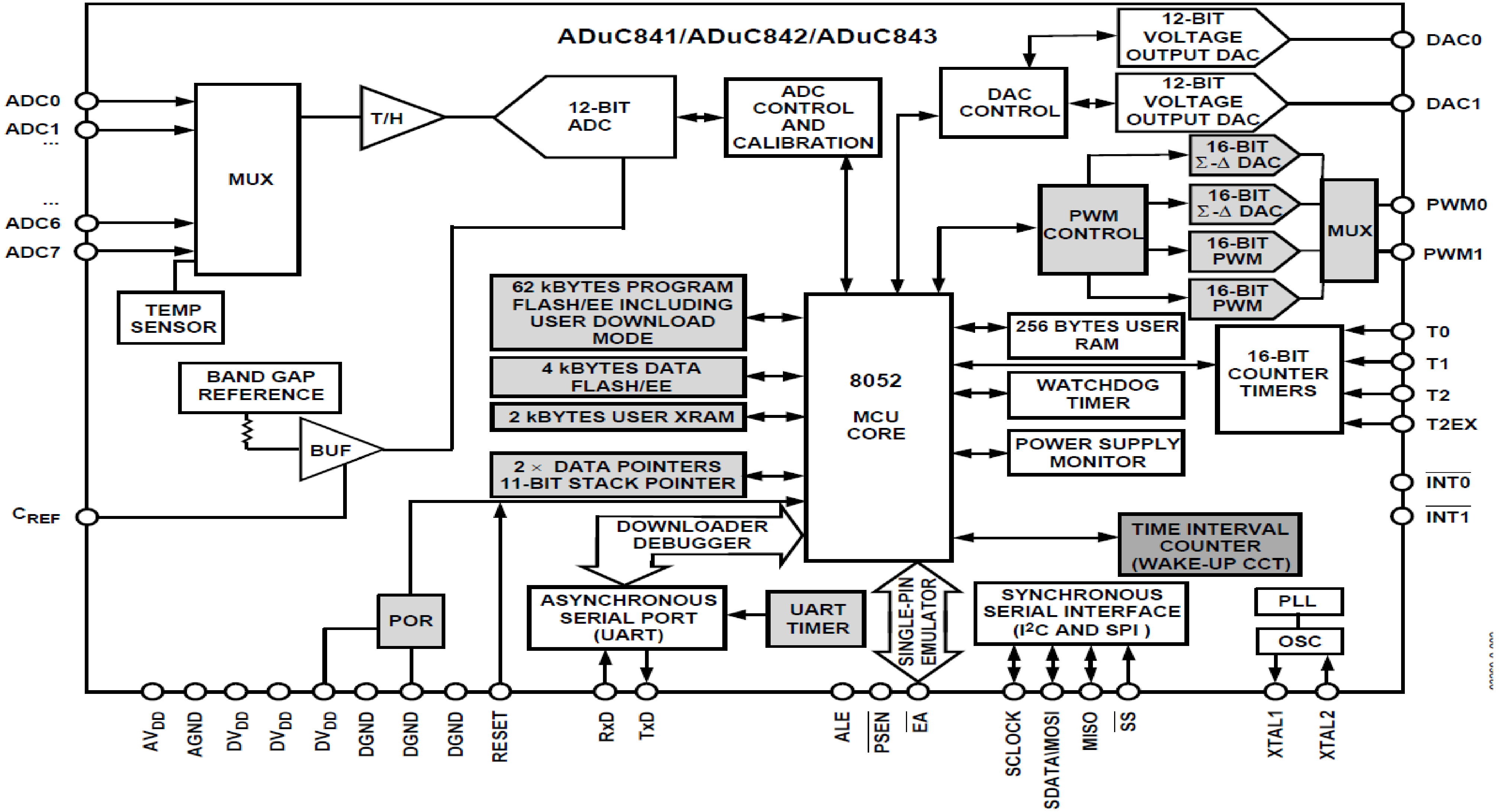
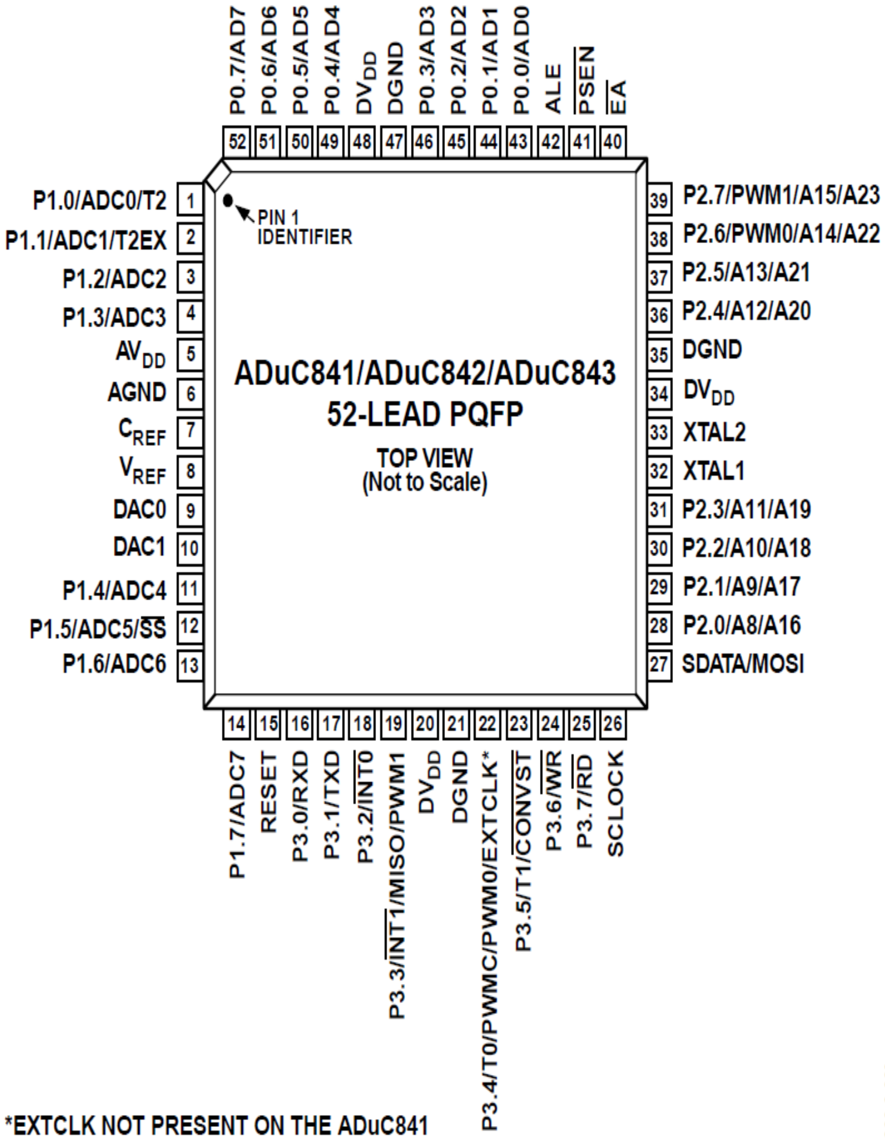


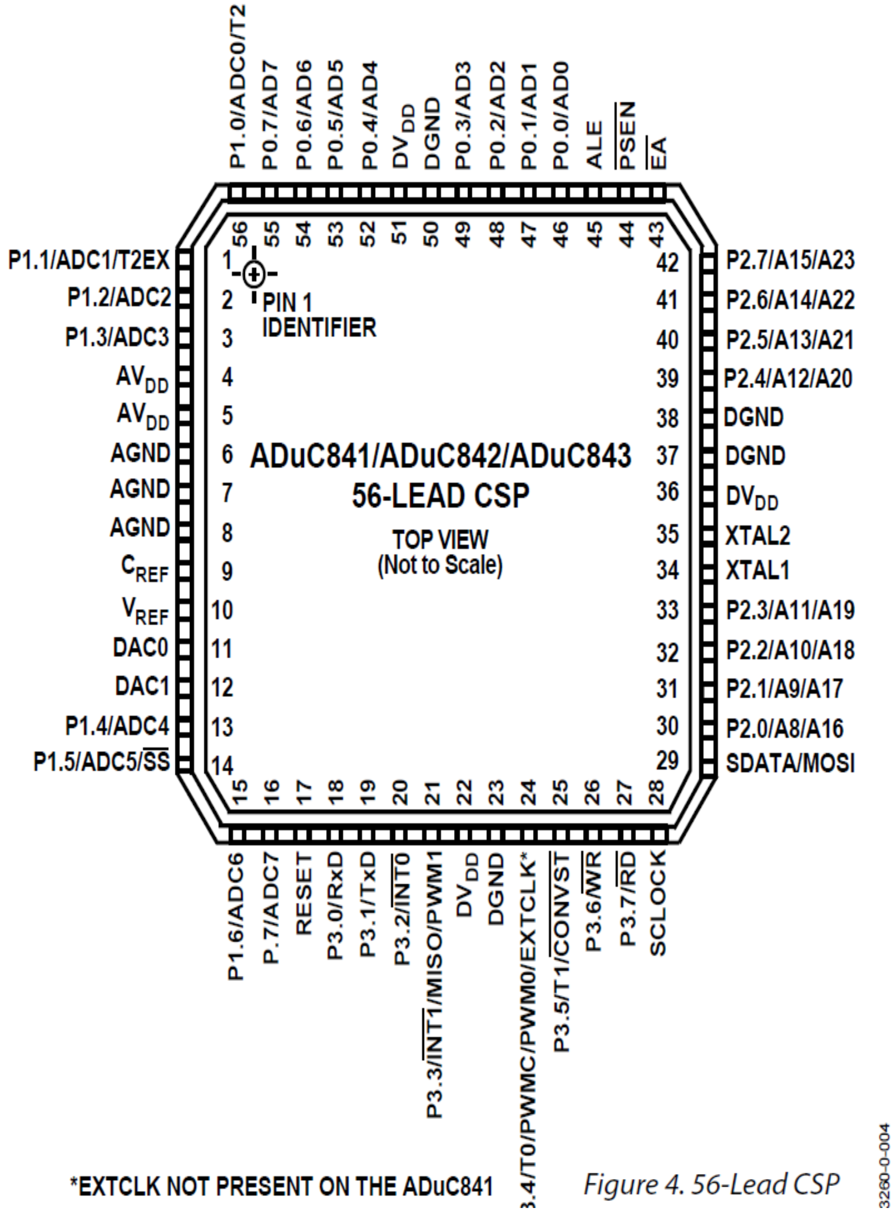
ADuC842 iç yapısı



*EXTCLK NOT PRESENT ON THE ADuC841

03260-0-003

Figure 3. 52-Lead PQPF

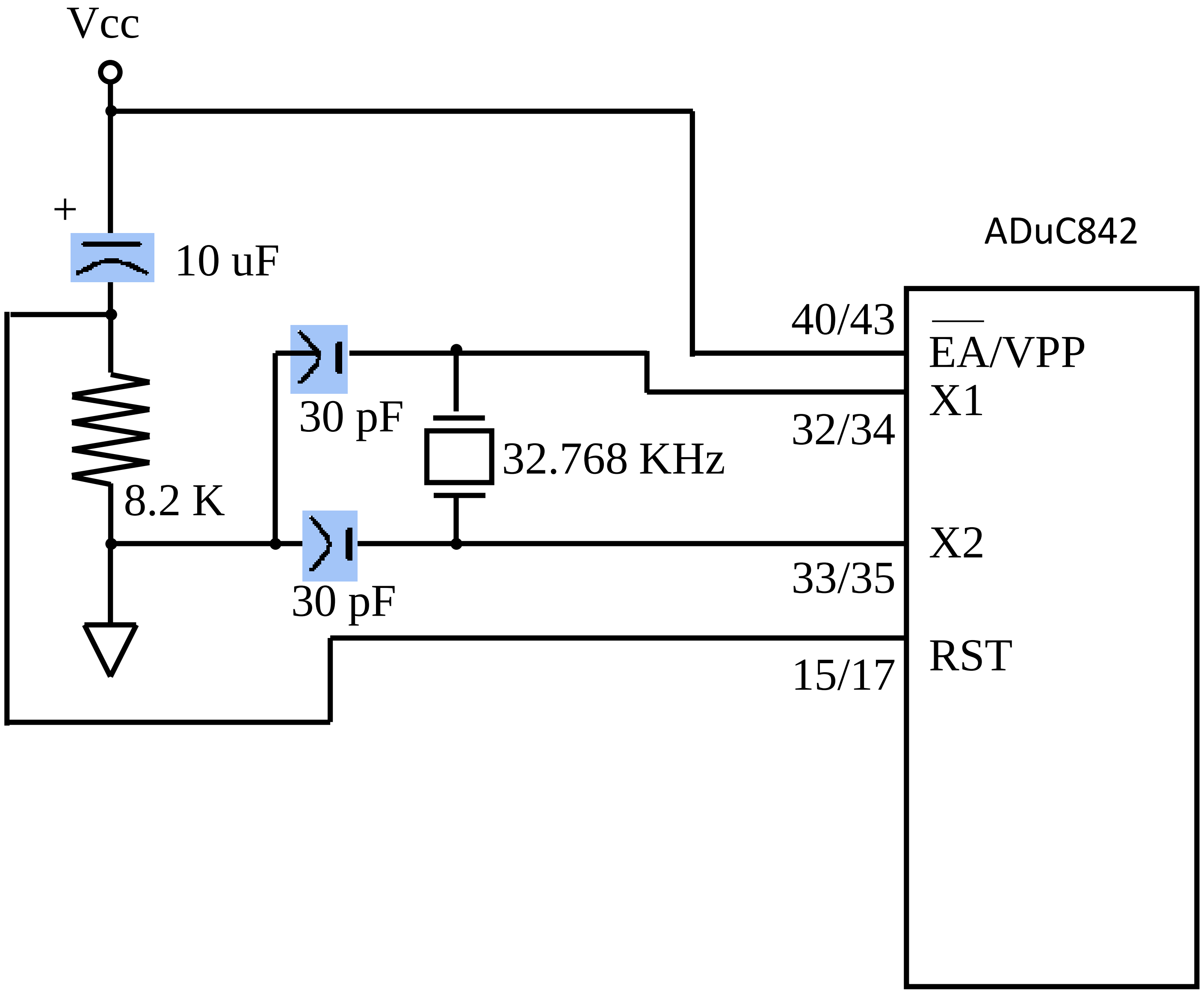


*EXTCLK NOT PRESENT ON THE ADuC841

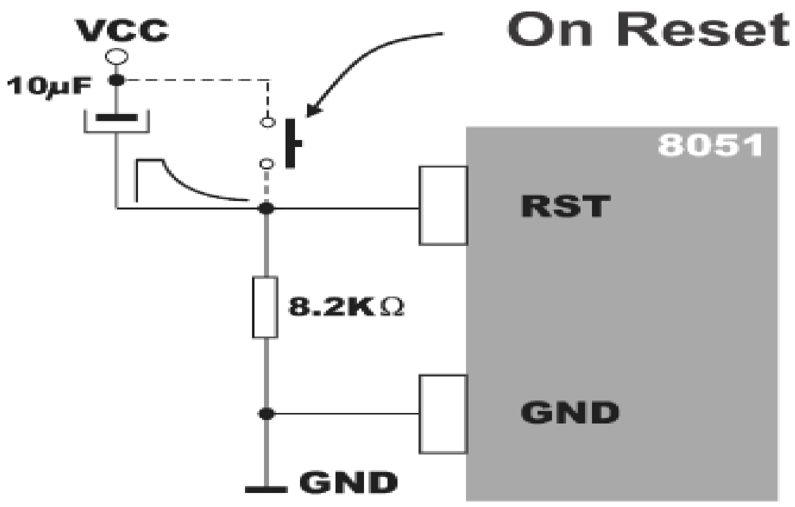
Figure 4. 56-Lead CSP

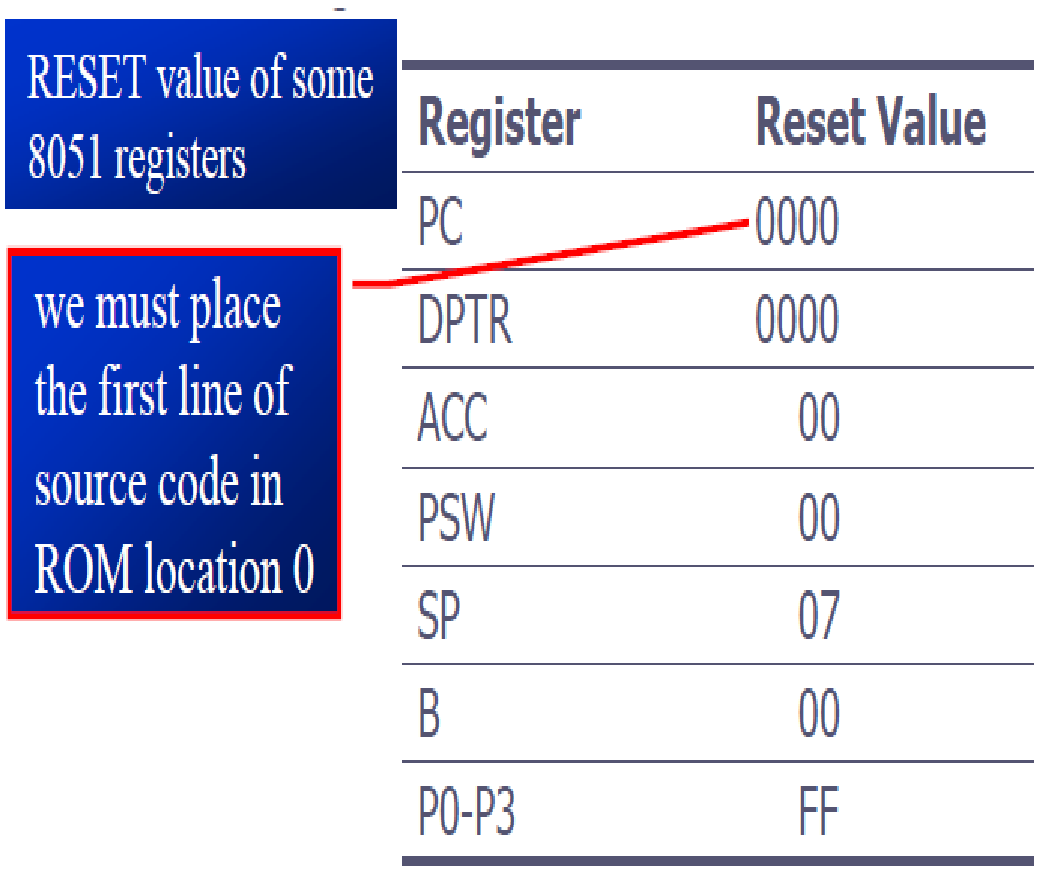
3260-0-004

Power-On RESET Devresi



Tipik Bir Reset Devresi ve Bazı Registerlerin Reset Sonrası Değerleri





RESET value of some 8051 registers

we must place the first line of source code in ROM location 0

Register	Reset Value
PC	0000
DPTR	0000
ACC	00
PSW	00
SP	07
B	00
P0-P3	FF

AduC 842 Programlama Modeli

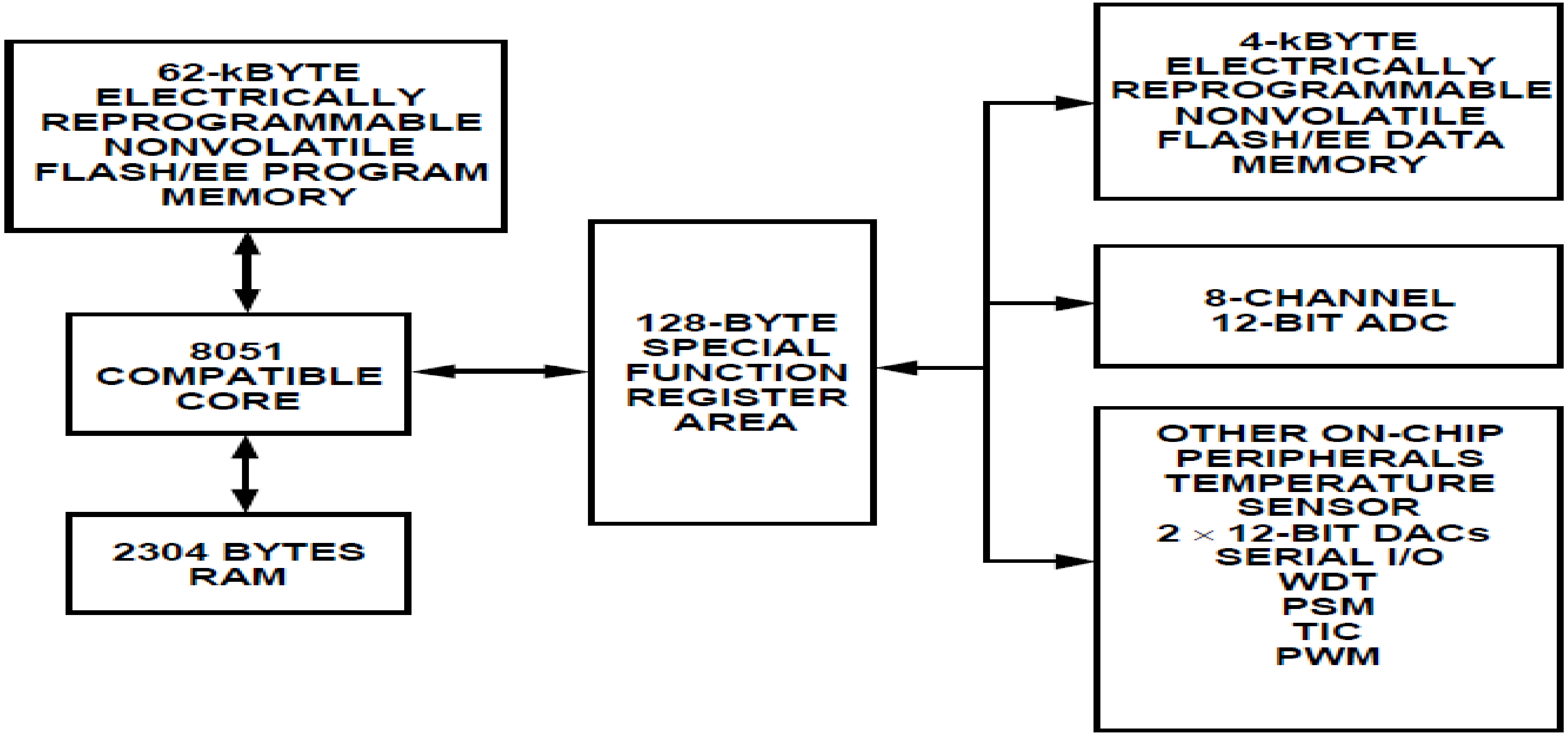
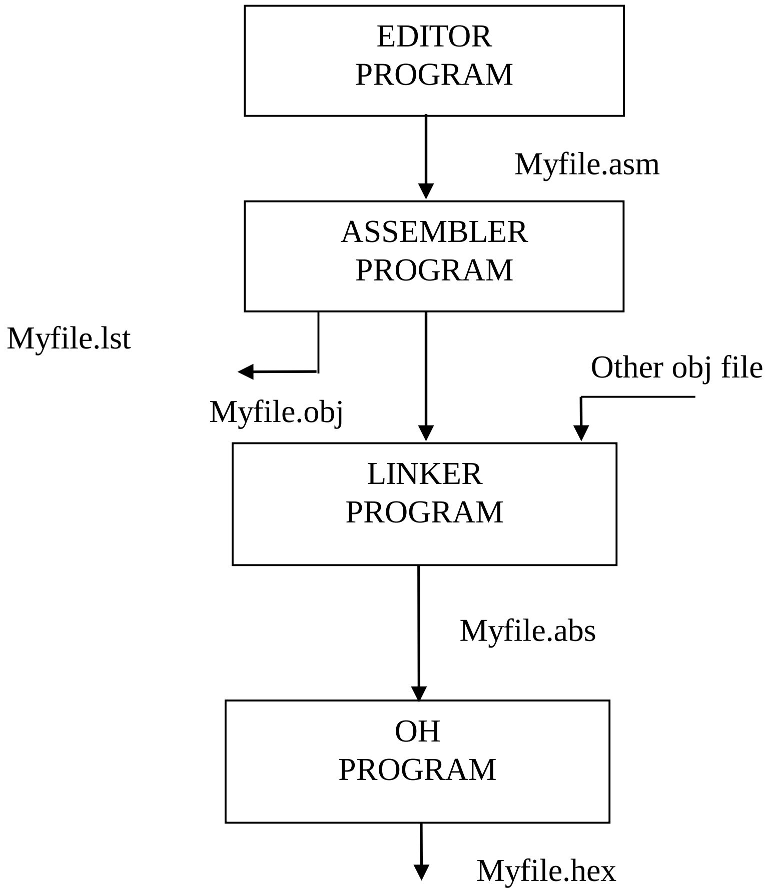

62-kBYTE ELECTRICALLY REPROGRAMMABLE NONVOLATILE FLASH/EE PROGRAM MEMORY
8051 COMPATIBLE CORE
2304 BYTES RAM
128-BYTE SPECIAL FUNCTION REGISTER AREA
4-kBYTE ELECTRICALLY REPROGRAMMABLE NONVOLATILE FLASH/EE DATA MEMORY
8-CHANNEL 12-BIT ADC
OTHER ON-CHIP PERIPHERALS
TEMPERATURE SENSOR
2 × 12-BIT DACs
SERIAL I/O
WDT
PSM
TIC
PWM

Figure 26. Programming Model

Assembler dilinin işlenmesi aşamaları



EDITOR PROGRAM

Myfile.asm

ASSEMBLER PROGRAM

Myfile.lst

Myfile.obj

Other obj file

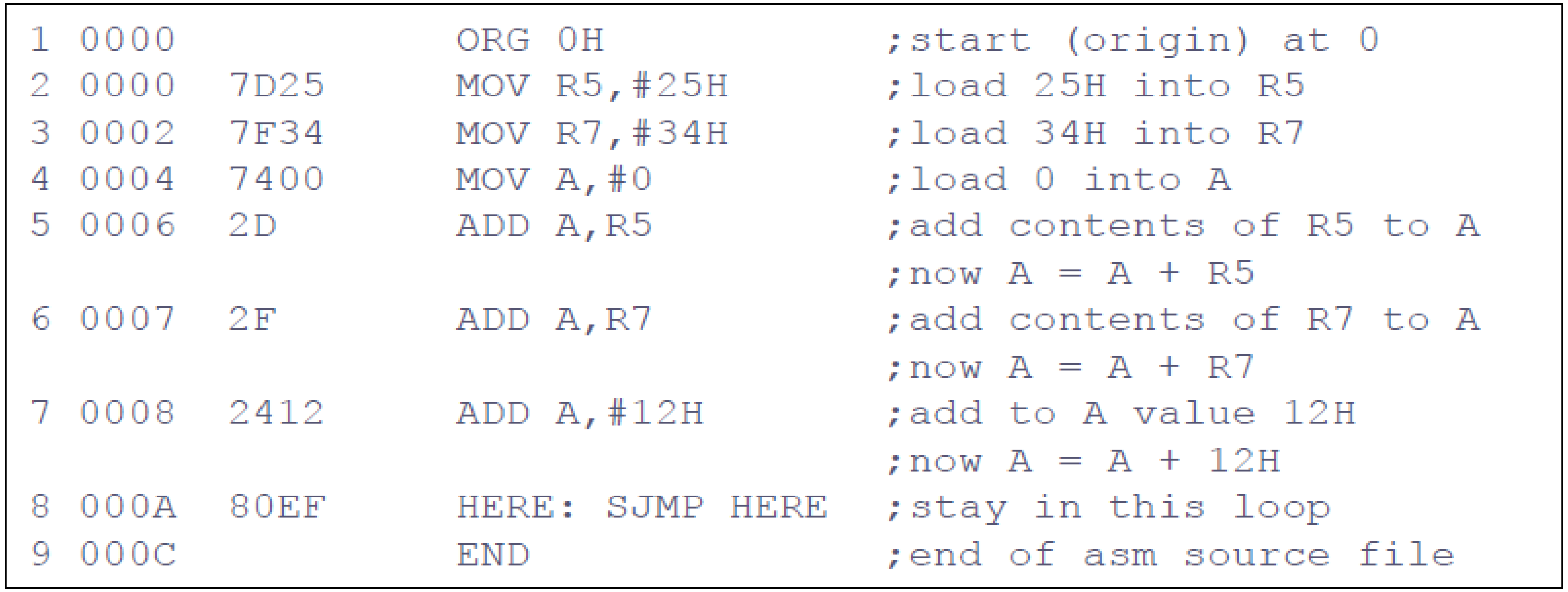
LINKER PROGRAM

Myfile.abs

OH PROGRAM

Myfile.hex

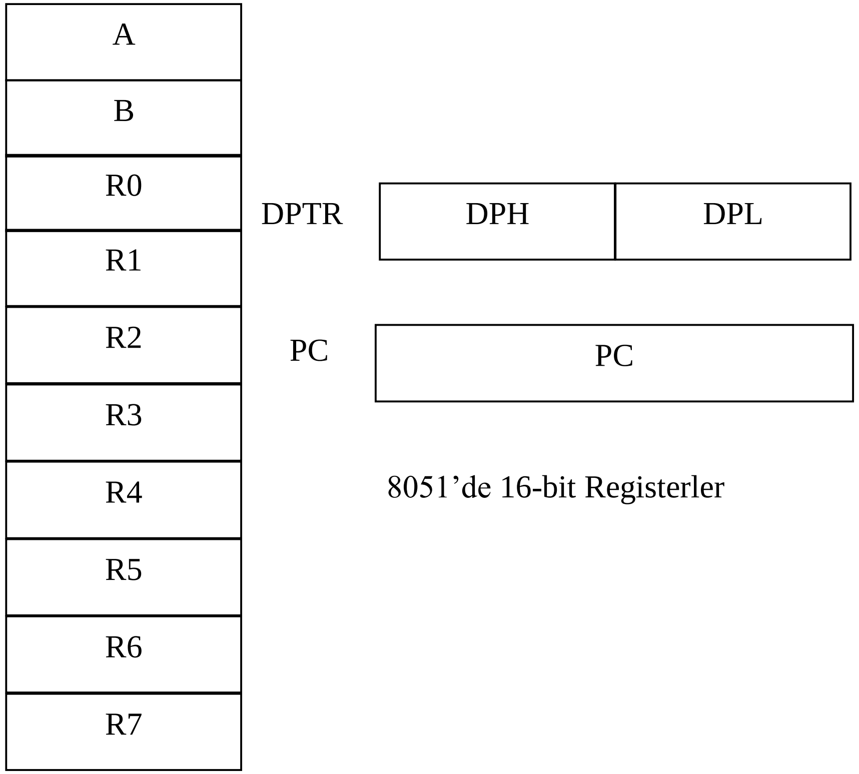
- Examine the list file and how the code is placed in ROM



```
1  0000              ORG 0H             ;start (origin) at 0
2  0000   7D25       MOV R5,#25H        ;load 25H into R5
3  0002   7F34       MOV R7,#34H        ;load 34H into R7
4  0004   7400       MOV A,#0           ;load 0 into A
5  0006   2D         ADD A,R5           ;add contents of R5 to A
                                        ;now A = A + R5
6  0007   2F         ADD A,R7           ;add contents of R7 to A
                                        ;now A = A + R7
7  0008   2412       ADD A,#12H         ;add to A value 12H
                                        ;now A = A + 12H
8  000A   80EF       HERE: SJMP HERE    ;stay in this loop
9  000C              END                ;end of asm source file
```

ROM Address	Machine Language	Assembly Language
0000	7D25	MOV R5, #25H
0002	7F34	MOV R7, #34H
0004	7400	MOV A, #0
0006	2D	ADD A, R5
0007	2F	ADD A, R7
0008	2412	ADD A, #12H
000A	80EF	HERE: SJMP HERE

Bazı Önemli Kaydediciler (Registers)



8051’de 16-bit Registerler

A: Accumulator
B: Used specially in MUL/DIV
R0-R7: GPRs

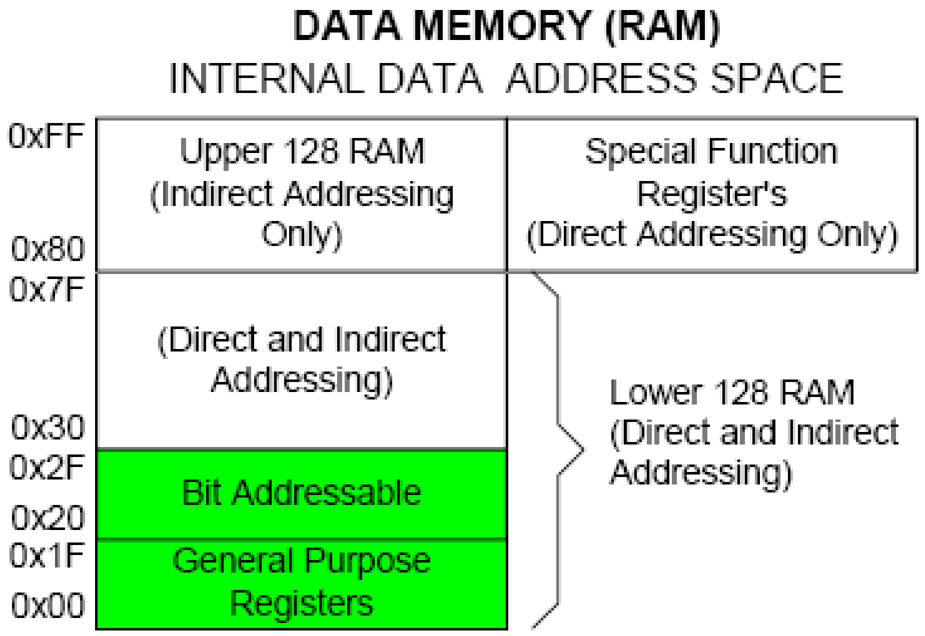
PSW

Bit No.	MSB 7	6	5	4	3	2	1	LSB 0	
$D0_H$	CY	AC	F0	RS1	RS0	OV	F1	P	PSW

<table>
<tr><th>Bit</th><th>Function</th></tr>
<tr><td>CY</td><td>Carry Flag
Used by arithmetic and conditional branch instruction.</td></tr>
<tr><td>AC</td><td>Auxiliary Carry Flag
Used by instructions which execute BCD operations.</td></tr>
<tr><td>F0</td><td>General Purpose Flag</td></tr>
<tr><td>RS1
RS0</td><td>Register Bank select control bits
These bits are used to select one of the four register banks.
<table>
<tr><th>RS1</th><th>RS0</th><th>Function</th></tr>
<tr><td>0</td><td>0</td><td>Registerbank 0 at data address 00_H - 07_H selected</td></tr>
<tr><td>0</td><td>1</td><td>Registerbank 1 at data address 08_H - $0F_H$ selected</td></tr>
<tr><td>1</td><td>0</td><td>Registerbank 2 at data address 10_H - 17_H selected</td></tr>
<tr><td>1</td><td>1</td><td>Registerbank 3 at data address 18_H - $1F_H$ selected</td></tr>
</table></td></tr>
<tr><td>OV</td><td>Overflow Flag
Used by arithmetic instruction.</td></tr>
<tr><td>F1</td><td>General Purpose Flag</td></tr>
<tr><td>P</td><td>Parity Flag
Always set/cleared by hardware to indicate an odd/even number of “one” bits in the accumulator.</td></tr>
</table>

Internal Data Memory

- Internal Data Memory 3 bölüme ayrılmıştır
    - Lower 128
    - Upper 128
    - Special function register (SFR)
- 80H to FFH adres aralığı Upper 128 ve SFRs bölgeleri tarafından ortak kullanıldığından dolayı 384 byte’lık fiziksel Ram hafıza bloğu oluşmuş olur.
- Uygun komutlarla her bir hafıza bloğuna ulaşılabilinir.



DATA MEMORY (RAM)

INTERNAL DATA ADDRESS SPACE

0xFF

0x80

Upper 128 RAM (Indirect Addressing Only)

Special Function Register's (Direct Addressing Only)

0x7F

0x30

(Direct and Indirect Addressing)

0x2F

0x20

Bit Addressable

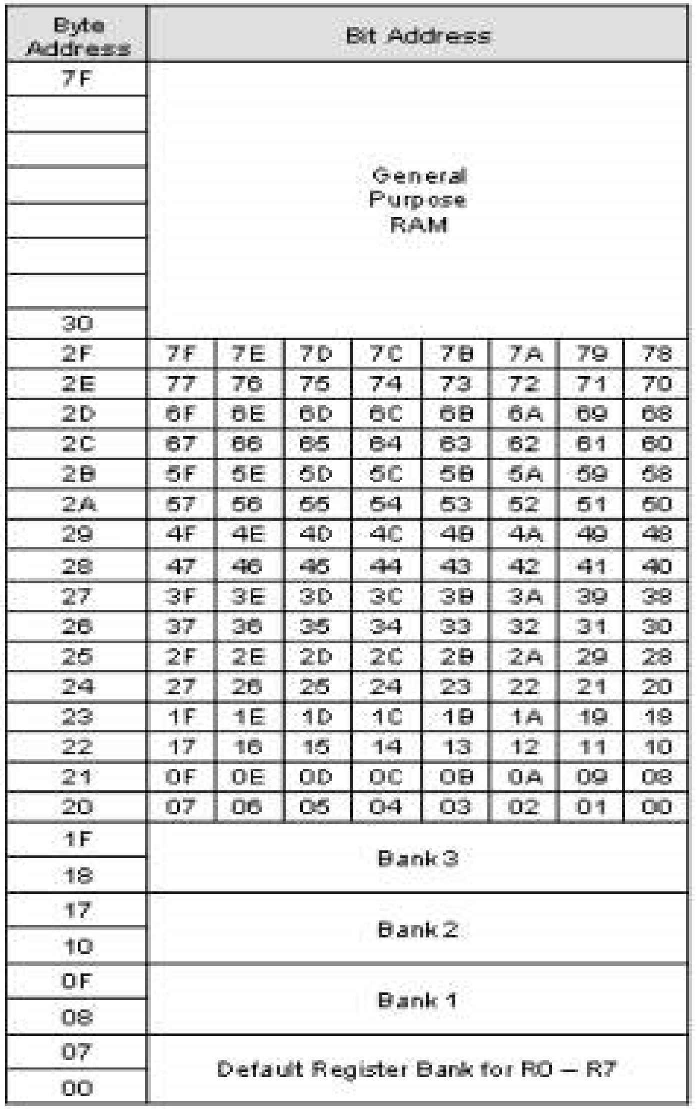
0x1F

0x00

General Purpose Registers

Lower 128 RAM (Direct and Indirect Addressing)

Lower 128—Register Banks and RAM



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								

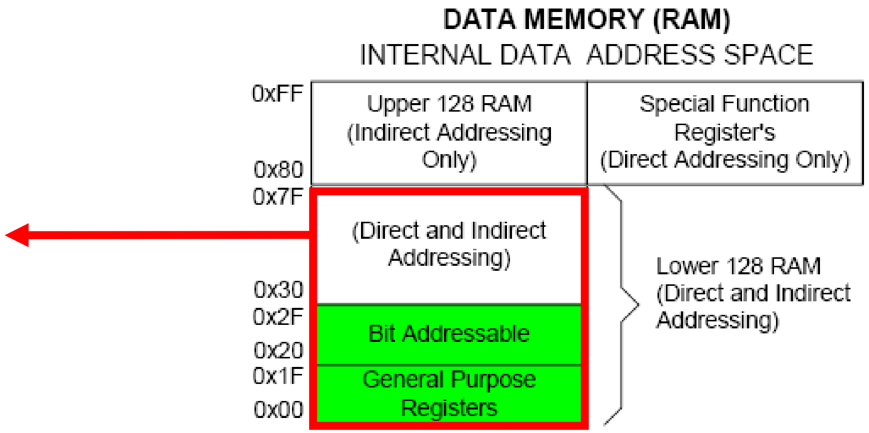
Bit Addressable

Bit-adreslenebilir Alan (16 bytes)

Genel Amaçlı RAM (80 bytes)

Register Banks

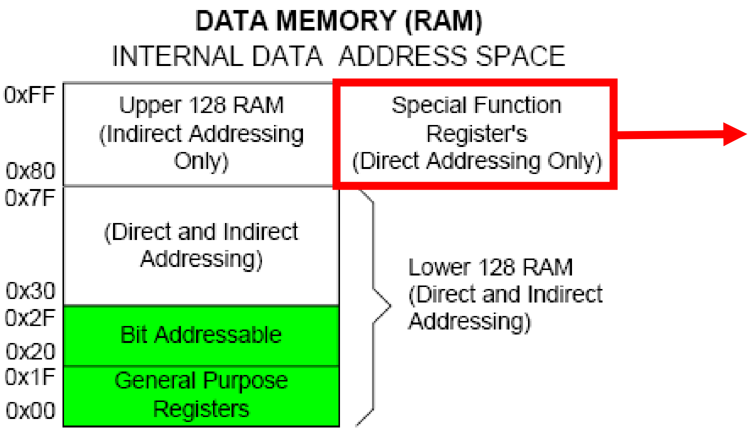
(4 Bank her biri 8 byte)

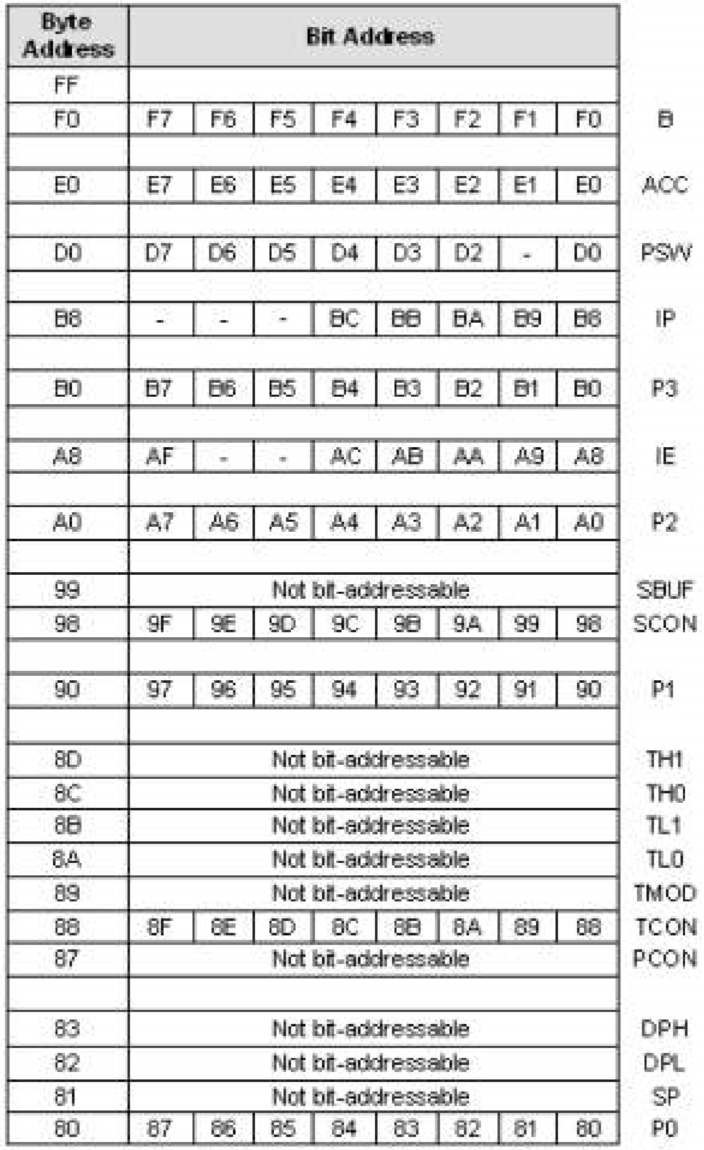


DATA MEMORY (RAM)
INTERNAL DATA ADDRESS SPACE
0xFF
0x80
0x7F
0x30
0x2F
0x20
0x1F
0x00
Upper 128 RAM (Indirect Addressing Only)
Special Function Register's (Direct Addressing Only)
(Direct and Indirect Addressing)
Bit Addressable
General Purpose Registers
Lower 128 RAM (Direct and Indirect Addressing)

Special Function Registers (SFRs) - Özel Fonksiyon Kaydedicileri

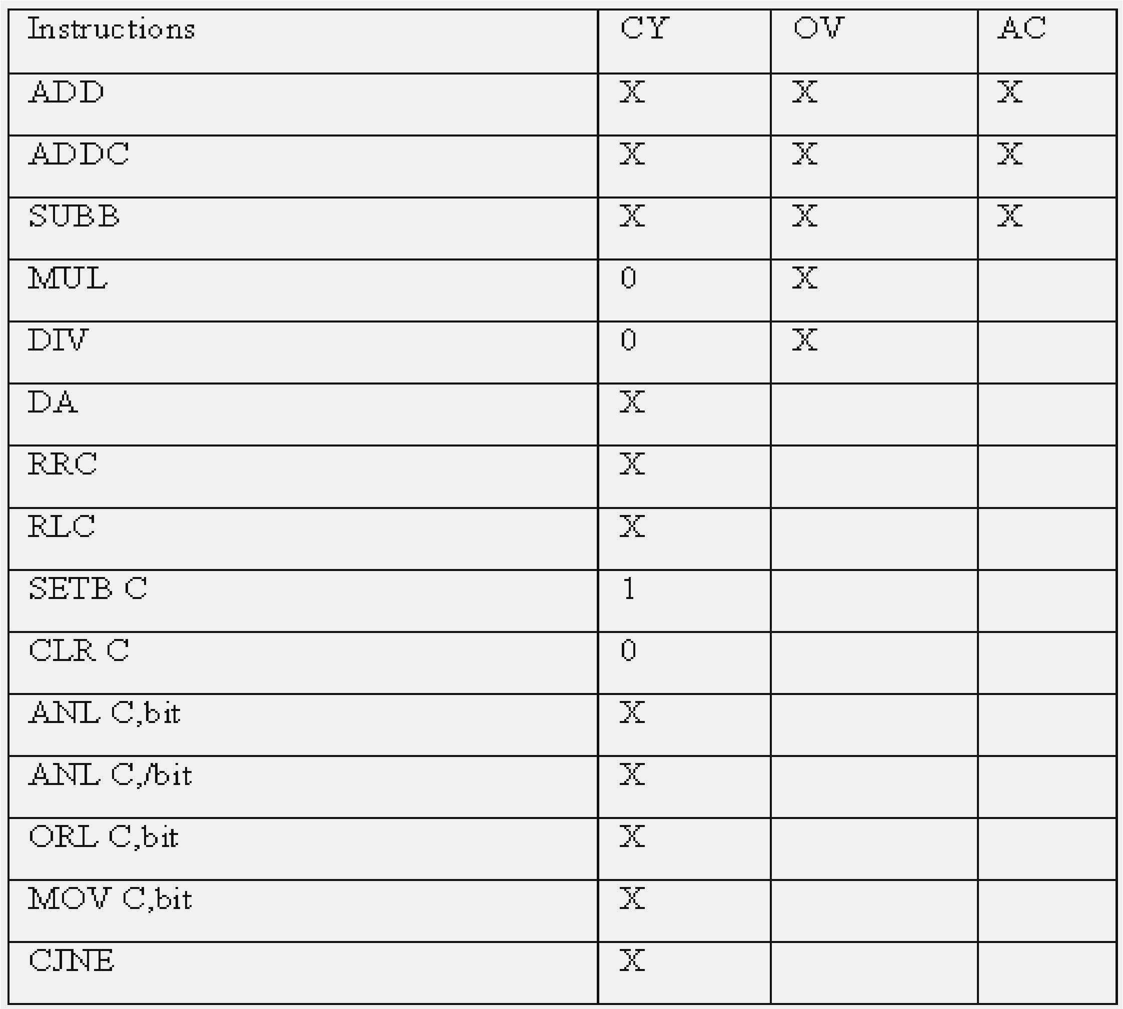
- SFR kaydediciler mikro denetleyici ve çevre birimler arasındaki kontrol işlemlerini ve data değişimini düzenler
- Özel Fonksiyon Registerleri 0H veya 8H ile biter ve bit-adreslenebilir *byte'lara sahiptirler*
- Bazı registerler bit-adreslenemez. Bunlar; stack pointer (SP) ve data pointer (DPTR) registerleridir.


DATA MEMORY (RAM)
INTERNAL DATA ADDRESS SPACE
0xFF
0x80
0x7F
0x30
0x2F
0x20
0x1F
0x00
Upper 128 RAM (Indirect Addressing Only)
Special Function Register's (Direct Addressing Only)
(Direct and Indirect Addressing)
Bit Addressable
General Purpose Registers
Lower 128 RAM (Direct and Indirect Addressing)



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	9E	9D	9C	9B	9A	99	98	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

Bazı Komutların Bayrak Bitlere (Flag Bits) Etkileri



Instructions	CY	OV	AC
ADD	X	X	X
ADDC	X	X	X
SUBB	X	X	X
MUL	0	X	
DIV	0	X	
DA	X		
RRC	X		
RLC	X		
SETB C	1		
CLR C	0		
ANL C,bit	X		
ANL C,/bit	X		
ORL C,bit	X		
MOV C,bit	X		
CJNE	X		

ADD Komutunun Bayrak Bitlere Etkisi

```
MOV      A,#88H
ADD      A,#93H

    88              10001000
   +93             +10010011
    ----             --------------
    11B            00011011
CY=1     AC=0      P=0
```

```
MOV      A,#38H
ADD      A,#2FH

   38              00111000
  +2F             +00101111
   ----             --------------
   67              01100111
CY=0     AC=1      P=1
```

```
MOV      A,#9CH
ADD      A,#64H

    9C              10011100
   +64             +01100100
    ----             --------------
    100            00000000
CY=1     AC=1      P=0
```

Addressing Modes

- Immediate
- Register
- Direct
- Register Indirect
- Indexed

Immediate Addressing Mode

```
MOV     A,#65H
MOV     A,#’A’
MOV     R6,#65H
MOV     DPTR,#2343H
MOV     P1,#65H
```

Example :

```
  Num          EQU     30
  …
  MOV          R0,Num
  MOV          DPTR,#data1
  …
  ORG          100H
  data1:       db      “Example”
```

Register Addressing Mode

MOV Rn, A ;n=0,..,7

ADD A, Rn

MOV DPL, R6

~~MOV DPTR, A~~

~~MOV Rm, Rn~~

Direct Addressing Mode

Although the entire of 128 bytes of RAM can be accessed using direct addressing mode, it is most often used to access RAM loc. 30 – 7FH.

```
MOV     R0, 40H
MOV     56H, A
MOV     A, 4            ; ≡ MOV A, R4
MOV     6, 2            ; copy R2 to R6
                        ; MOV  R6,R2 is invalid !
```

SFR register and their address

```
MOV     0E0H, #66H      ; ≡ MOV A,#66H
MOV     0F0H, R2        ; ≡ MOV B, R2
MOV     80H,A           ; ≡ MOV P1,A
```

Register Indirect Addressing Mode

- In this mode, register is used as a pointer to the data.

```
MOV                 A,@Ri       ; move content of RAM loc.Where address is held by Ri into A
                                ( i=0 or 1 )
MOV                 @R1,B
```

In other word, the content of register R0 or R1 is sources or target in MOV, ADD and SUBB insructions.

Example:

Write a program to copy a block of 10 bytes from RAM location sterting at 37h to RAM location starting at 59h.

Solution:

```
   MOV   R0,37h            ; source pointer
   MOV   R1,59h            ; dest pointer
   MOV   R2,10             ; counter
L1: MOV   A,@R0
   MOV   @R1,A
   INC   R0
   INC   R1
   DJNZ  R2,L1
```

Indexed Addressing Mode And On-Chip ROM Access

- This mode is widely used in accessing data elements of look-up table entries located in the program (code) space ROM at the 8051

```
MOVC        A,@A+DPTR
```

A= content of address A +DPTR from ROM

Note:

Because the data elements are stored in the program (code) space ROM of the 8051, it uses the instruction MOVC instead of MOV. The "C" means code.

- Example:

  Assuming that ROM space starting at 250h contains “Hello.”, write a program to transfer the bytes into RAM locations starting at 40h.

Solution:

```
          ORG                 0
          MOV       DPTR,#MYDATA
          MOV       R0,#40H
L1:                 CLR                 A
          MOVC      A,@A+DPTR
          JZ                  L2
          MOV                 @R0,A
          INC                 DPTR
          INC                 R0
          SJMP                L1
L2:                 SJMP                L2
;-----------------------------------
          ORG                 250H
          MYDATA: DB          “Hello”,0

          END
```

<u>**Notice the NULL character ,0, as end of string and how we use the JZ instruction to detect that.**</u>

- Example:

  Write a program to get the x value from P1 and send x^2 to P2, continuously .

Solution:

```
        ORG             0                   ;code segment
        MOV     DPTR, #TAB1     ;moving data segment to data pointer
        MOV     A,#0FFH         ;configuring P1 as input port
        MOV     P1,A
L01:
        MOV     A,P1            ;reading value from P1
        MOVC    A,@A+DPTR
        MOV             P2,A
        SJMP            L01
;------------------------------------------------
        ORG             300H                ;data segment
TAB1:   DB      0,1,4,9,16,25,36,49,64,81

        END
```

External Memory Addressing

- MOVX A, @R1 ; A ← [R1] (in external memory)
- MOVX A, @DPTR
- MOVX @DPTR, A

MUL & DIV

- **MUL AB** **;**B|A = A*B

```
MOV     A,#25H
MOV     B,#65H
MUL     AB          ;25H*65H=0E99
                    ;B=0EH, A=99H
```

- **DIV AB** **;**A = A/B, B = A mod B

```
MOV     A,#25
MOV     B,#10
DIV AB          ;A=2, B=5
```

Example

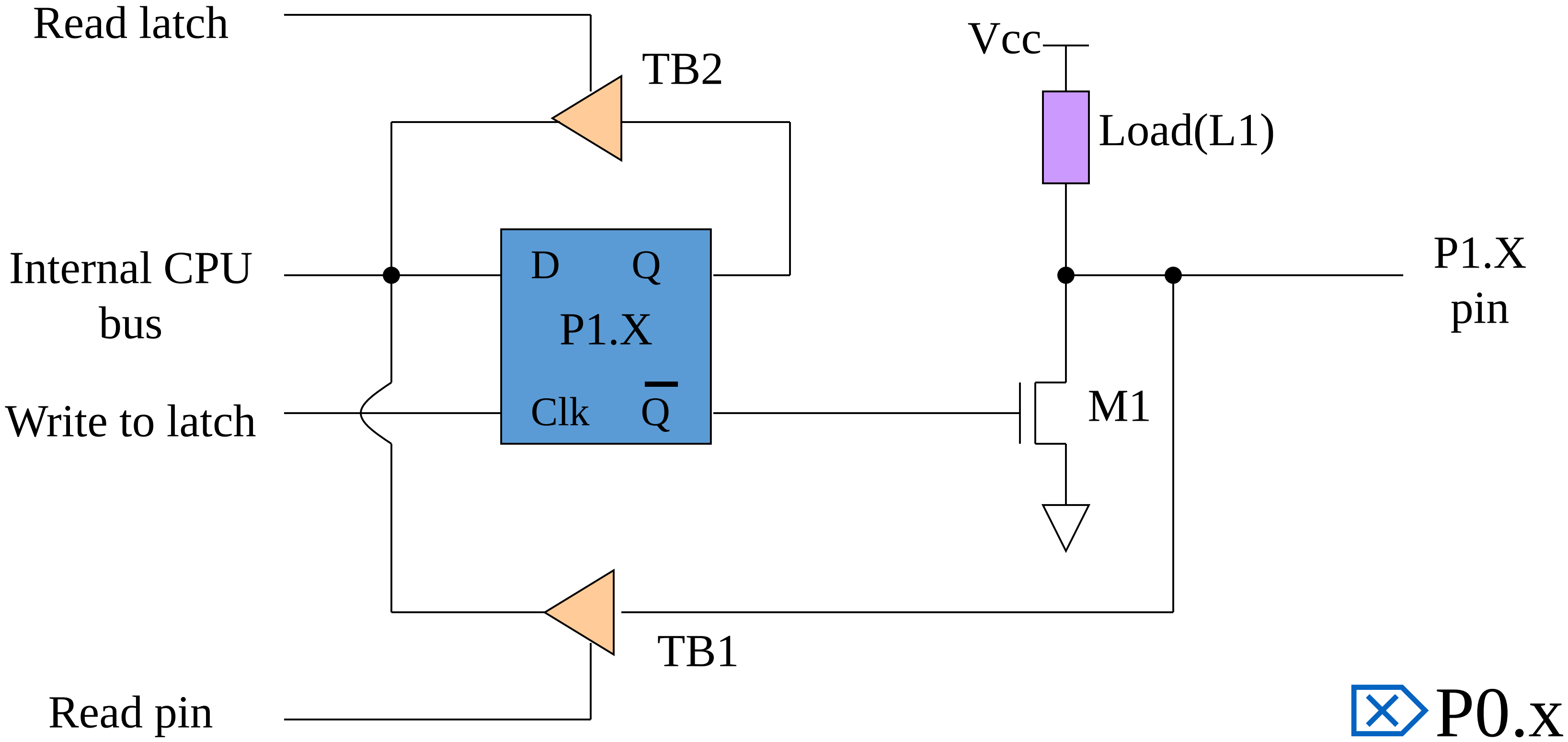
	A	B	R5	R7	Address	Data
ORG 0H						
VAL1 EQU 05H						
MOV R5,#25H						
LOOP: MOV R7,#VAL1						
MOV A,#0						
ADD A,R5						
ADD A,#12H						
RRC A						
DJNZ A, LOOP						
SETB ACC.3						
CLR A						
CJNE A, #0, LOOP						
HERE: SJMP HERE						
END						

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



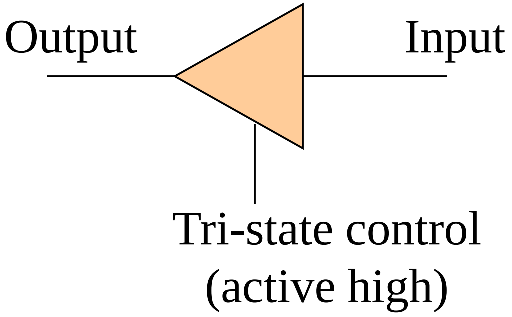
⌧P0.x

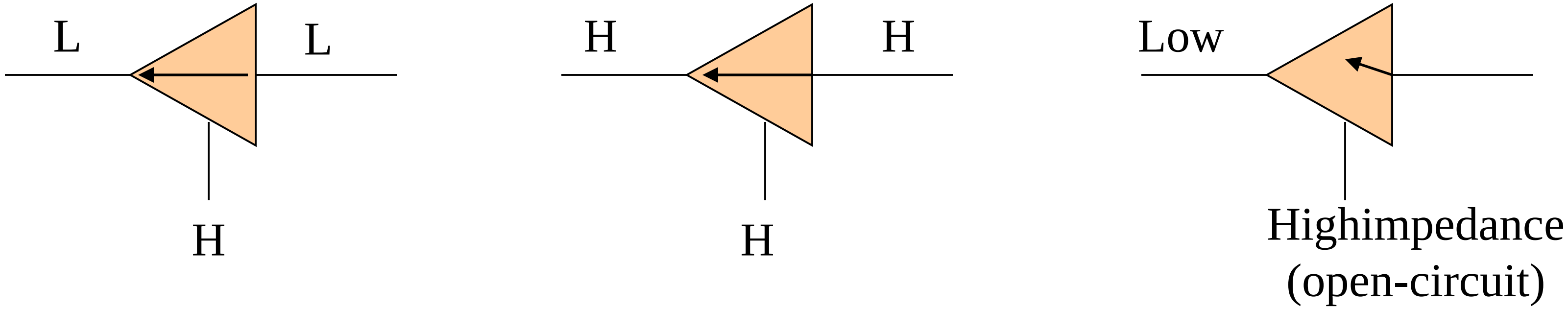
8051 IC

Hardware Structure of I/O Pin

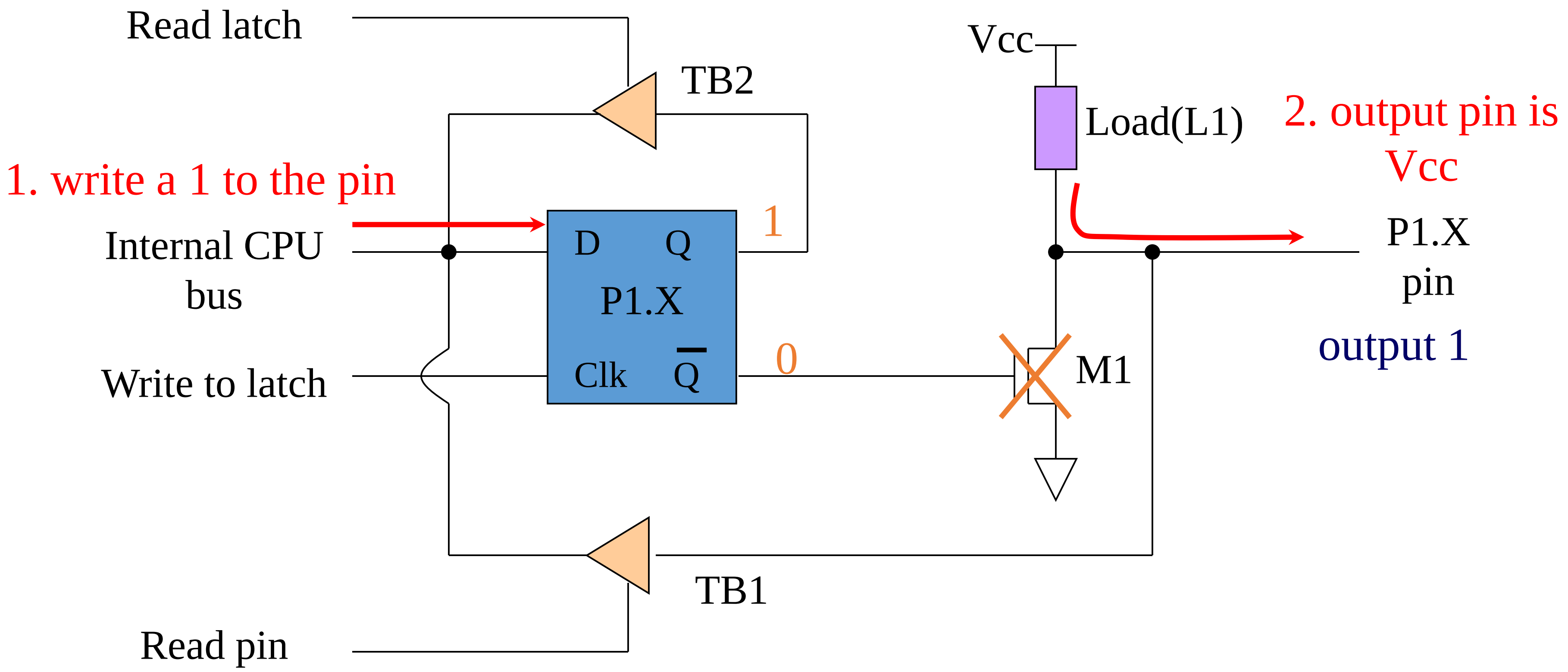
- Each pin of I/O ports
  - Internal CPU bus：communicate with CPU
  - A D latch store the value of this pin
    - D latch is controlled by “Write to latch”
      - Write to latch＝1：write data into the D latch
  - 2 Tri-state buffer：
    - TB1: controlled by “Read pin”
      - Read pin＝1：really read the 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



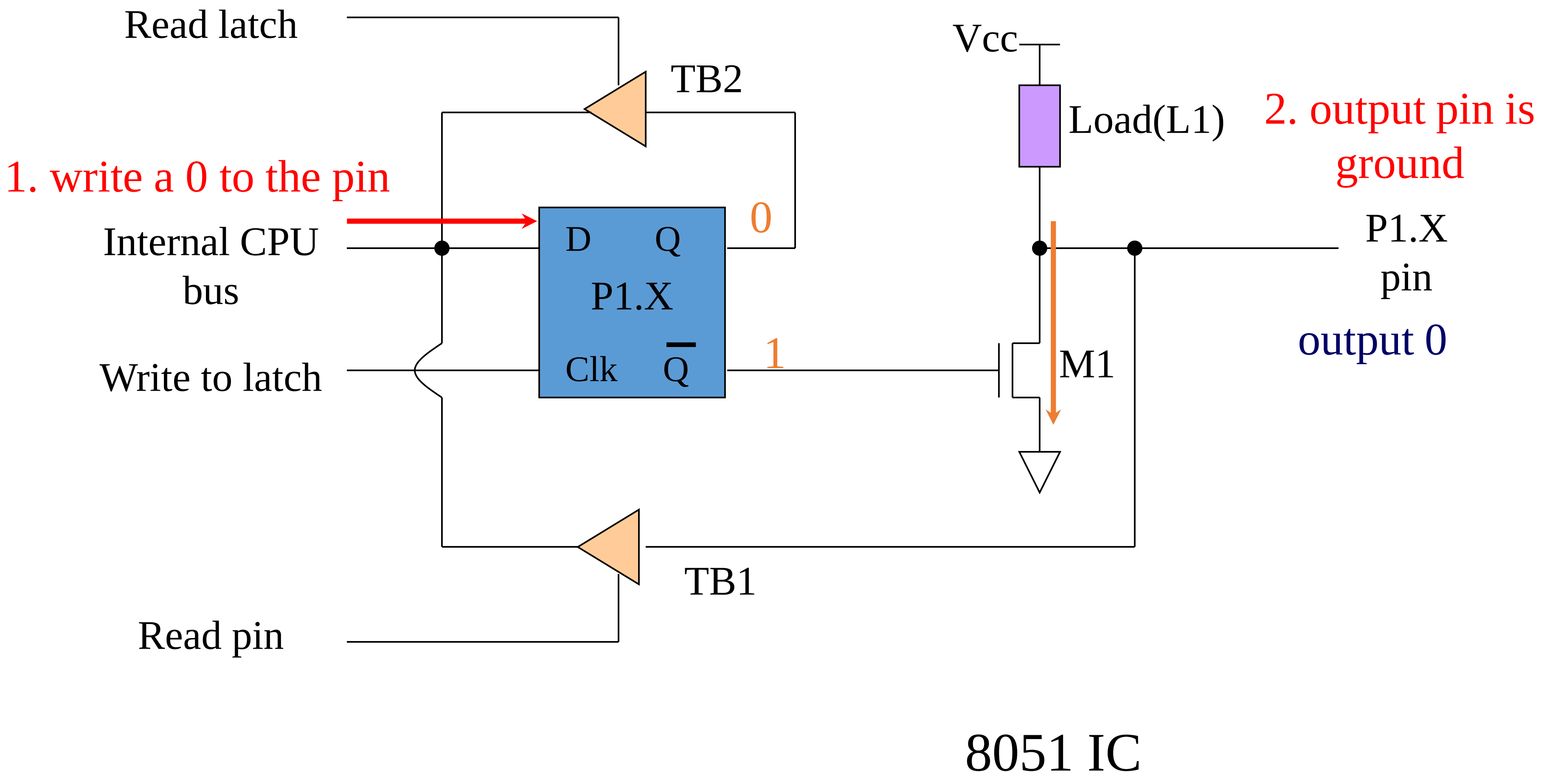


Writing “1” to Output Pin P1.X



8051 IC

Writing “0” to Output Pin P1.X



Port 1 as Output（Write to a Port）

- Send data to Port 1：

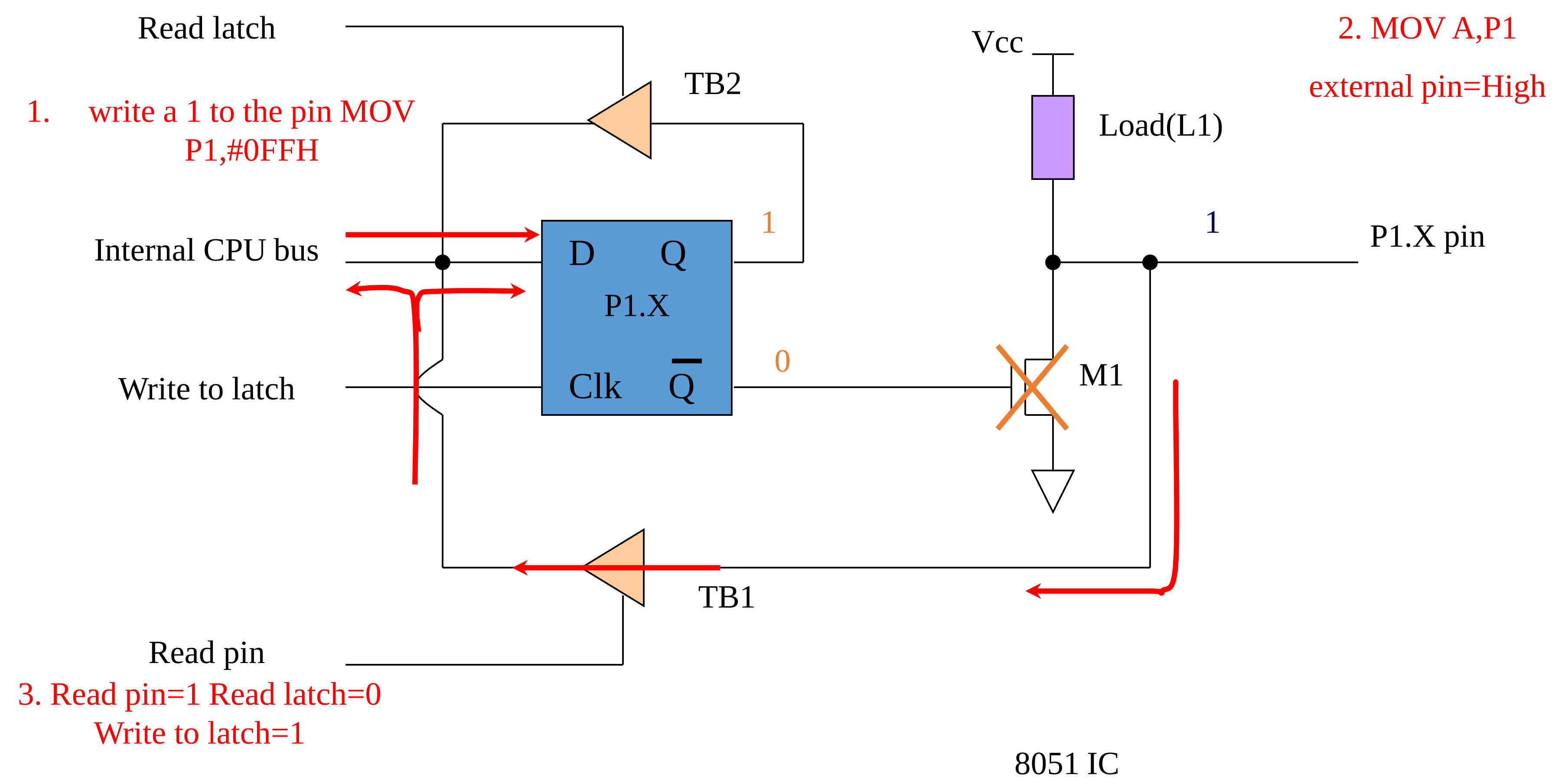
```
          MOV     A,#55H
BACK:     MOV     P1,A
         ACALL    DELAY
         CPL  A
         SJMP  BACK
```

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

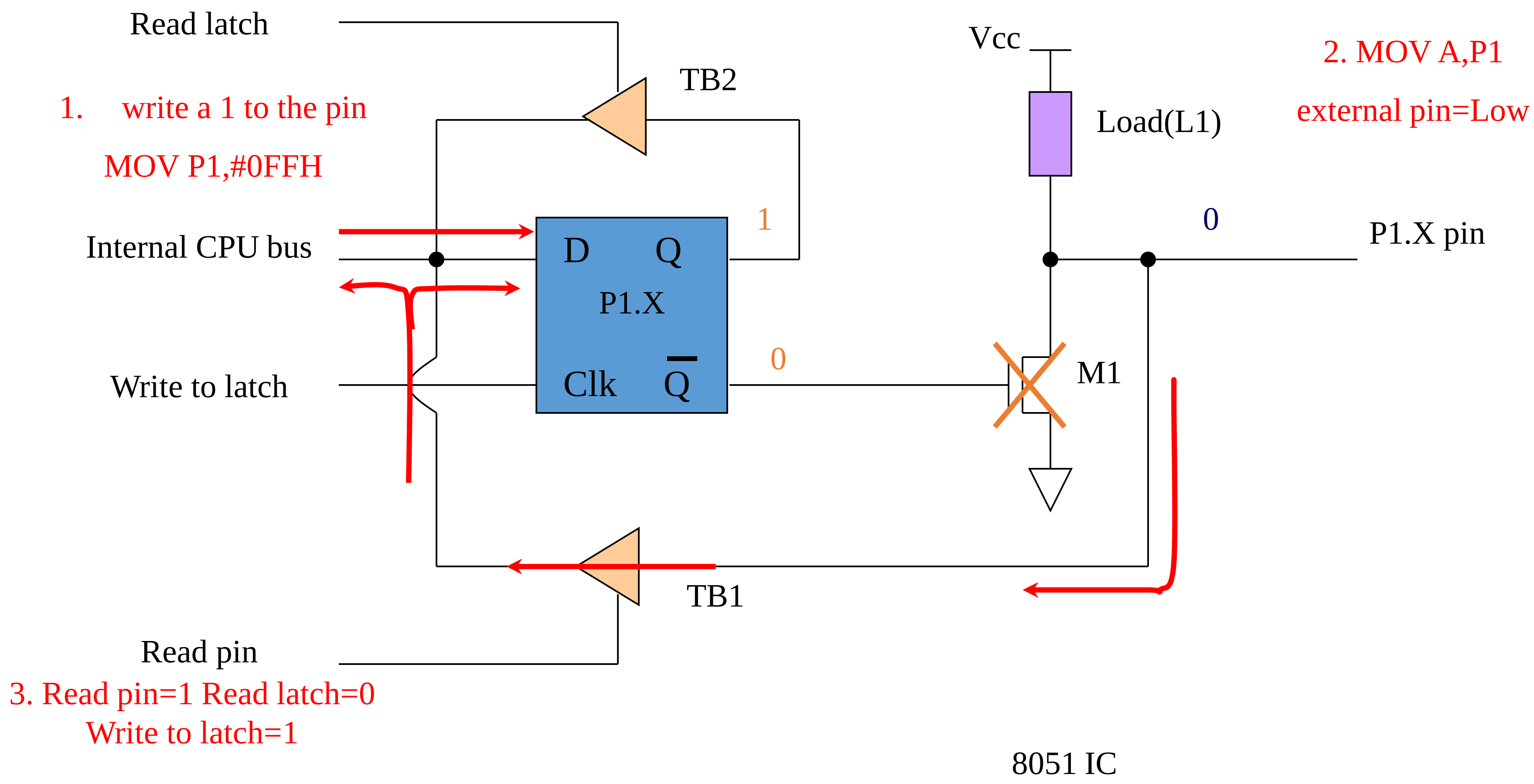
Reading Input v.s. Port Latch

- 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
    - Figures C-11, C-12
  - Read the *internal latch* of the output port.
    - ANL   P1, A   ; P1 ← P1 AND A
    - ORL   P1, A   ; P1 ← P1 OR A
    - INC   P1   ; increase P1
    - Figure C-17
    - Table C-6 Read-Modify-Write Instruction (or Table 8-5)
- See Section 8.3

Reading "High" at Input Pin



Reading “Low” at Input Pin



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.

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

Reading Latch

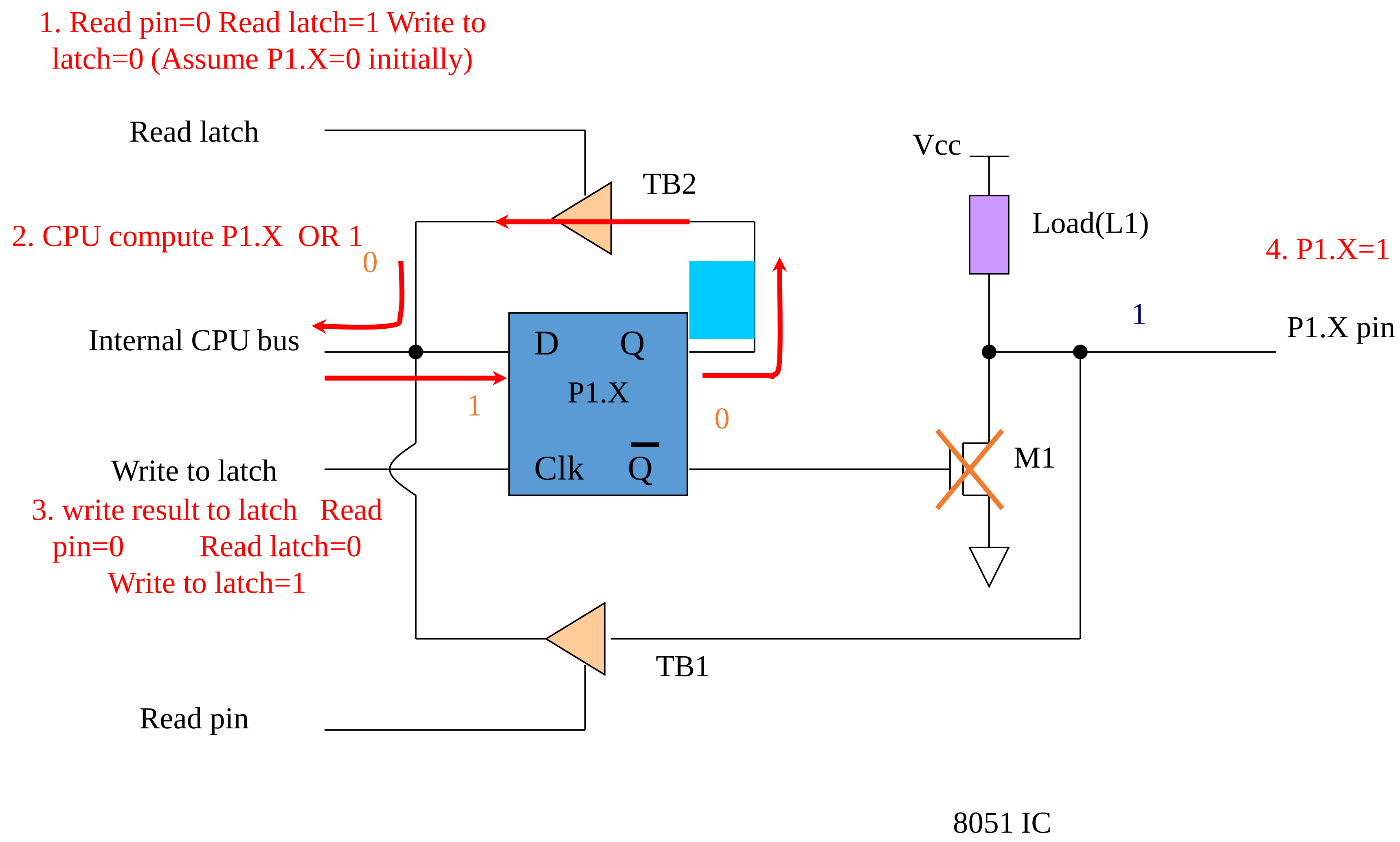
- Exclusive-or the Port 1：

  MOV P1,#55H ;P1=01010101

  ORL P1,#0F0H ;P1=11110101

  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.

Reading the Latch



Port 1 as Input（Read from latch）

- Exclusive-or the Port 1：

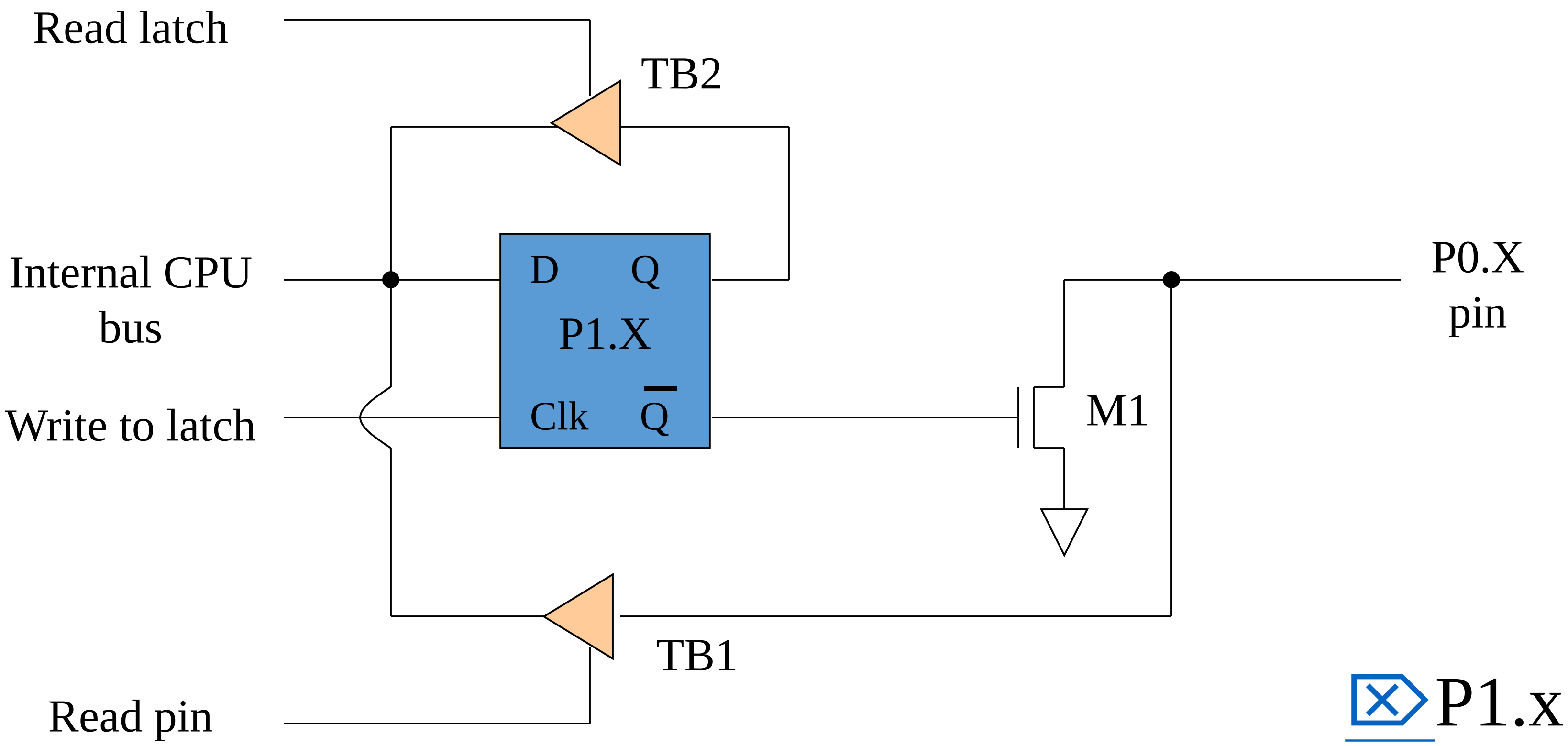
```
         MOV   P1,#55H   ;P1=01010101
AGAIN:   XOR   P1,#0FFH  ;complement
         ACALL DELAY
         SJMP  AGAIN
```

  - Note that the XOR of 55H and FFH gives AAH.
  - XOR of AAH and FFH gives 55H.
  - The instruction read the data in the latch (not from the pin).
  - The instruction result will put into the latch and the pin.

Other Pins

- P1, P2, and P3 have internal pull-up resisters.
  - P1, P2, and P3 are not open drain.
- P0 has no internal pull-up resistors and does not connects to Vcc inside the 8051.
  - P0 is open drain.
  - Compare the figures of P1.X and P0.X.
- However, for a programmer, it is the same to program P0, P1, P2 and P3.
- All the ports upon RESET are configured as output.

A Pin of Port 0

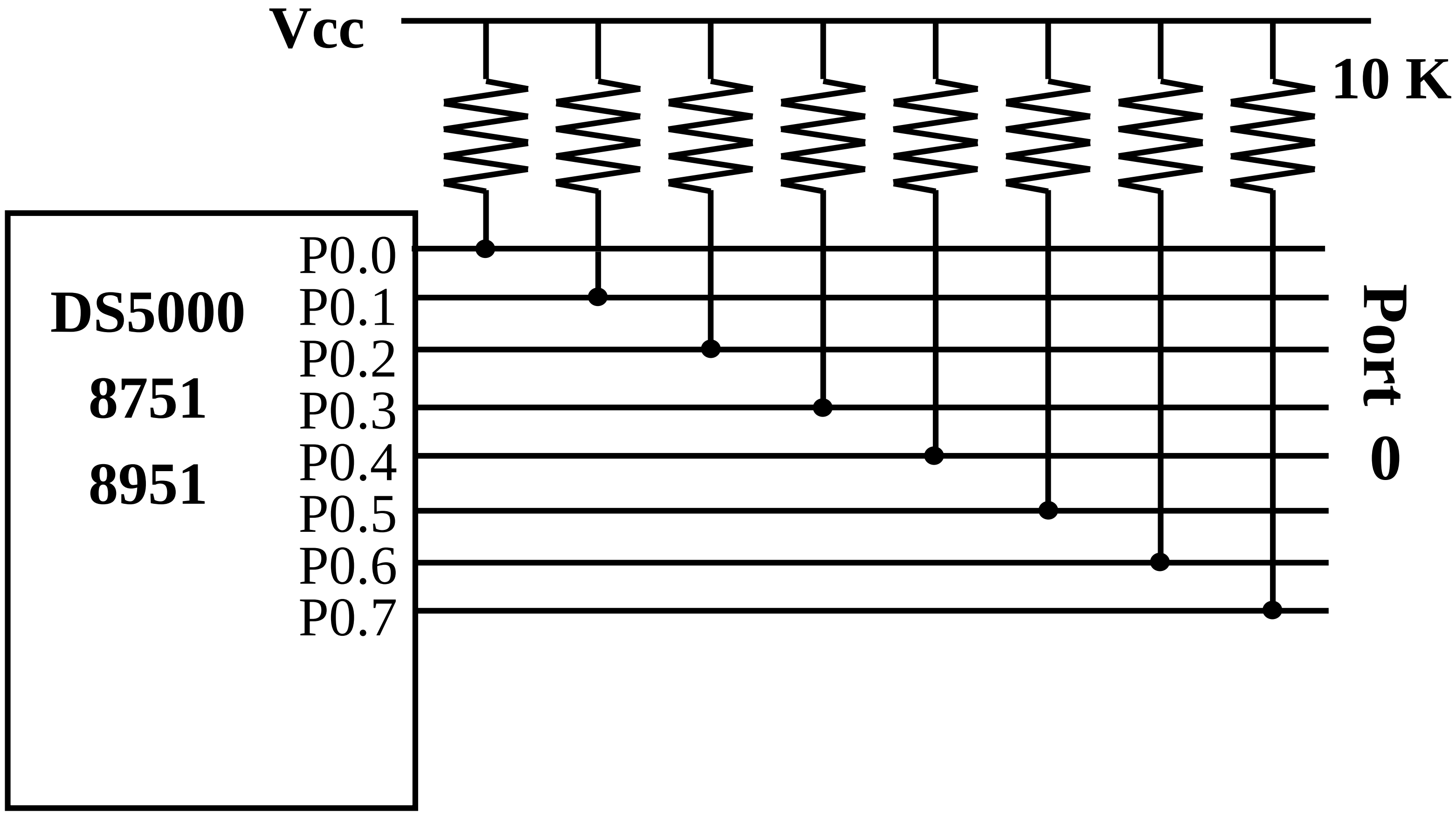


8051 IC

Port 0（pins 32-39）

- P0 is an open drain.
  - Open drain is a term used for MOS chips in the same way that open collector is used for TTL chips. ⌦
- When P0 is used for simple data I/O we must connect it to external pull-up resistors.
  - Each pin of P0 must be connected externally to a 10K ohm pull-up resistor.
  - With external pull-up resistors connected upon reset, port 0 is configured as an output port.

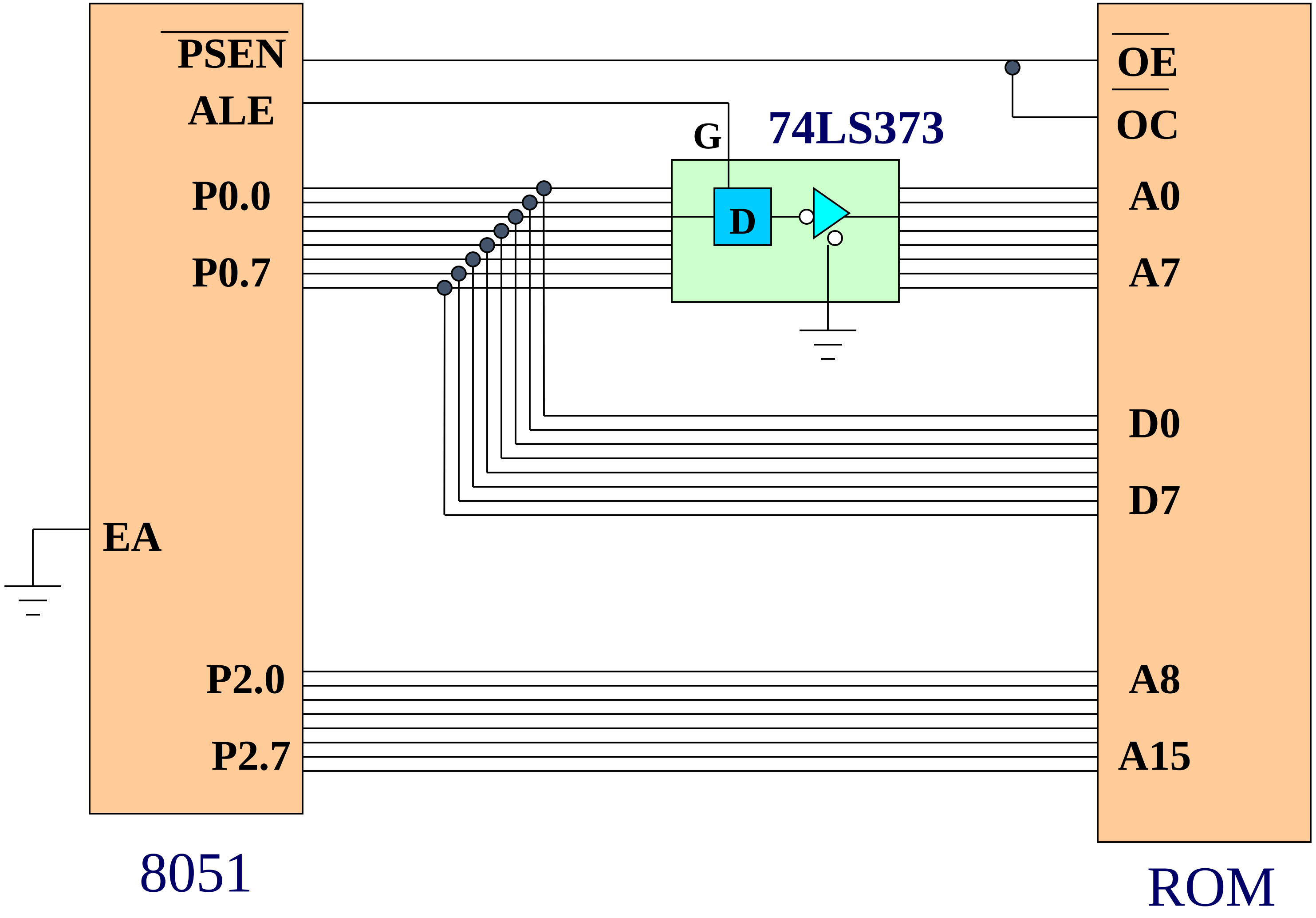
Port 0 with Pull-Up Resistors



Dual Role of Port 0

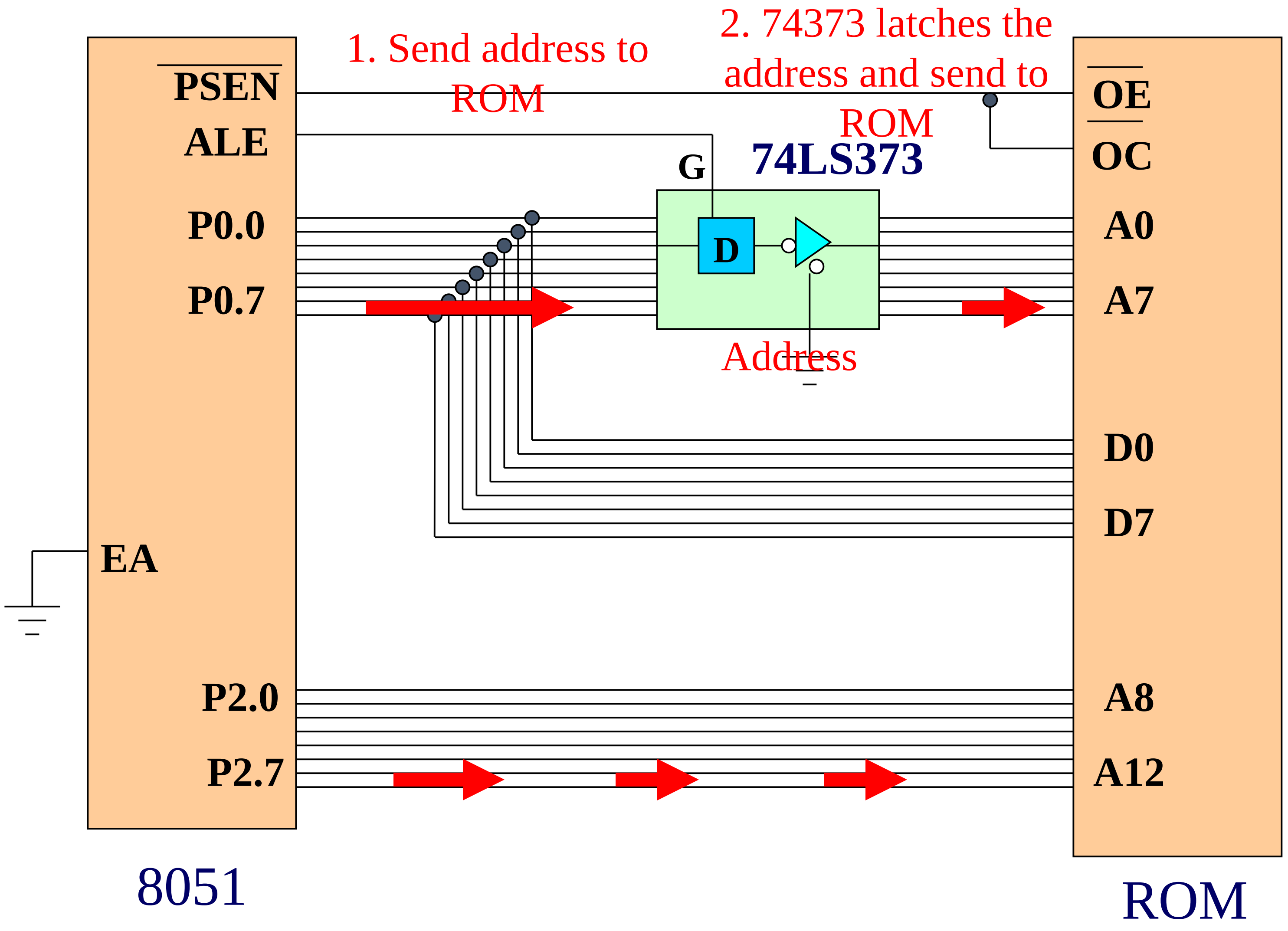
- When connecting an 8051/8031 to an external memory, the 8051 uses ports to send addresses and read instructions.
  - 8031 is capable of accessing 64K bytes of external memory.
  - 16-bit address：P0 provides both address A0-A7, P2 provides address A8-A15.
  - Also, P0 provides data lines D0-D7.
- When P0 is used for address/data multiplexing, it is connected to the 74LS373 to latch the address.
  - There is no need for external pull-up resistors as shown in Chapter 14.

74LS373



PSEN
ALE
P0.0
P0.7
EA
P2.0
P2.7
8051
G
74LS373
D
OE
OC
A0
A7
D0
D7
A8
A15
ROM

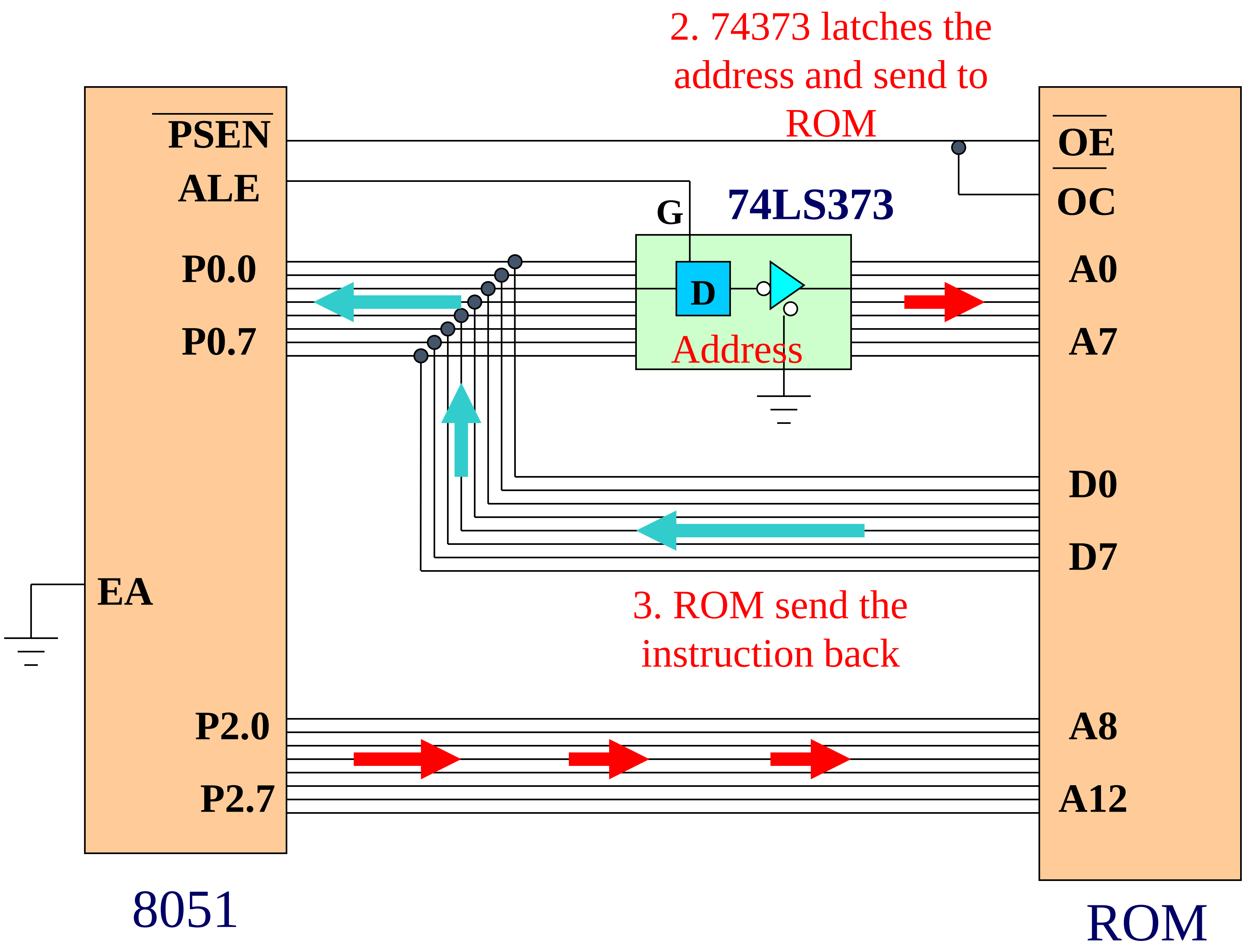
Reading ROM (1/2)



1. Send address to ROM

2. 74373 latches the address and send to ROM

Reading ROM (2/2)



2. 74373 latches the address and send to ROM

3. ROM send the instruction back

ALE Pin

- The ALE pin is used for de-multiplexing the address and data by connecting to the G pin of the 74LS373 latch.
  - When ALE=0, P0 provides data D0-D7.
  - When ALE=1, P0 provides address A0-A7.
  - The reason is to allow P0 to multiplex address and data.

Port 2（pins 21-28）

- Port 2 does not need any pull-up resistors since it already has pull-up resistors internally.
- In an 8031-based system, P2 are used to provide address A8-A15.

Port 3（pins 10-17）

- Port 3 does not need any pull-up resistors since it already has pull-up resistors internally.
- Although port 3 is configured as an output port upon reset, this is not the way it is most commonly used.
- Port 3 has the additional function of providing signals.
  - Serial communications signal：RxD, TxD（Chapter 10）
  - External interrupt：/INT0, /INT1（Chapter 11）
  - Timer/counter：T0, T1（Chapter 9）
  - External memory accesses in 8031-based system：/WR, /RD（Chapter 14）

Port 3 Alternate Functions

P3 Bit	**Function**	**Pin**
P3.0	RxD	10
P3.1	TxD	11
P3.2	$\overline{\text{INT0}}$	12
P3.3	$\overline{\text{INT1}}$	13
P3.4	T0	14
P3.5	T1	15
P3.6	$\overline{\text{WR}}$	16
P3.7	$\overline{\text{RD}}$	17

Generating Delays

- You can generate short delays using a register and incrementing or decrementing its value
- Example:

```
         mov r1, #0ah
   loop: djnz r1, loop
```

- How much delay is that?
  - Djnz is a 2-byte instruction it takes two machine cycles
  - One machine cycle is 1/12 of the system clock period
  - For a 12 MHz system clock that is:
  - Machine cycle = 12/12 = 1 MHz
  - Machine period = 1/(1 MHz) = 10^(-6) s = 1 µs
  - Loop time = 10*2*1 µs = 20 µs

Generating longer delays

- Each register is 8 bits long, so it can increment 256 times before overflowing
- For larger delays, or when interrupts are required 8051 uses two timers

```
DELAY:
        MOV     R5,#0FFH
DLY2:
        MOV     R7,#0FFH
 DLY1:
        MOV     R6,#0FFH
        DJNZ    R6,$
        DJNZ    R7,DLY1
        DJNZ    R5,DLY2
        RET
```