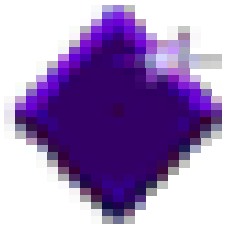
MICROCONTROLLER ENGINEERING



Kỹ thuật Vi điều khiển

Th.S LÊ XỨNG và Th.S NGUYỄN BÁ HỘI

Trường Đại học Bách khoa, ĐHĐN

Lexung59@yahoo.com hoinb@ud.edu.vn

Sách tham khảo

- **I. Scott Mackenzie, *Họ Vi điều khiển 8051*, Dịch: Tống Văn On và Hoàng Đức Hải, 2001**
- **Ngô Diên Tập, *Kỹ thuật ghép nối máy tính*, 2000**
- **Ngô Diên Tập, *Đo lường và điều khiển bằng máy tính*, 1998**
- **Đỗ Xuân Tiến, *Kỹ thuật Vi xử lý và lập trình Assembly cho hệ vi xử lý*, 2003**
- **Sencer Yeralan, *The 8051 Cookbook for Assembly and C with Experiments in Mechatronics amd Robotics*, 2000**
- **Sencer Yeralan, *Programming and Interfacing the 8051 Microcontroller*, 1995**

Lí thuyết

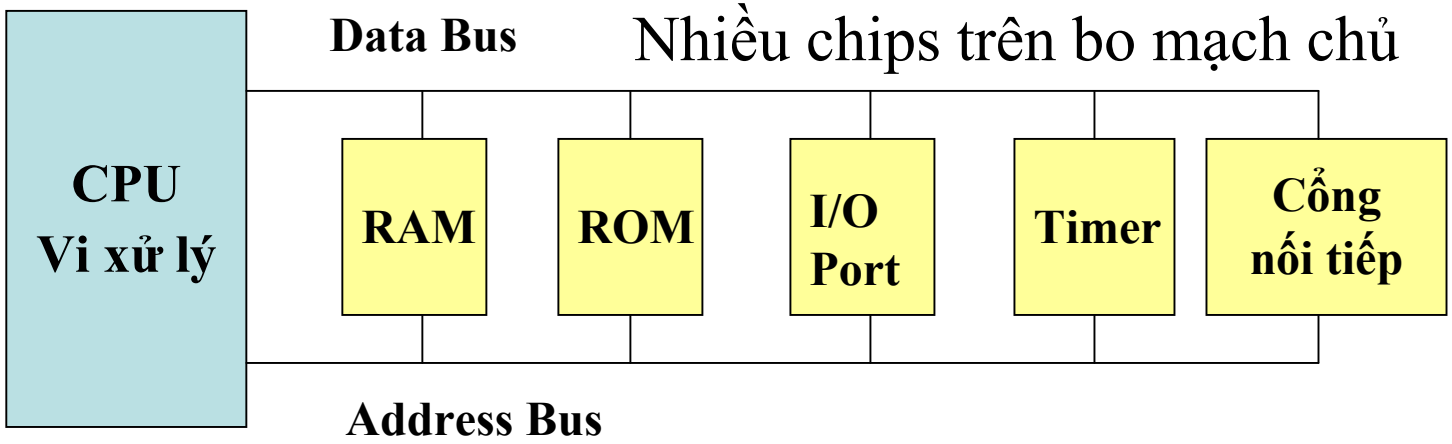
1. Giới thiệu
2. Sơ đồ khối và chân
3. Tổ chức bộ nhớ
4. Các thanh ghi chức năng đặc biệt
5. Dao động và hoạt động reset
6. Tập lệnh
7. Các mode định địa chỉ
8. Lập trình IO
9. Tạo trễ
10. Lập trình Timer/Counter
11. Lập trình giao tiếp nối tiếp
12. Lập trình ngắt
13. Lập trình hợp ngữ

Thực hành

1. LED 1
2. LED 2
3. Hiển thị với LED 7 đoạn
4. Keypad
5. Đo thời gian
6. Tạo sóng vuông
7. Đồng hồ số
8. Giao tiếp PC, viết C code
9. ADC, cảm biến & xuất LED 7 đoạn
10. Motor bước
11. PWM

Vi xử lý

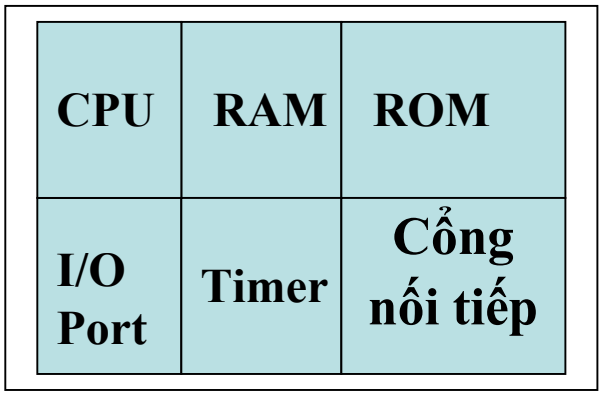
- CPU cho các máy tính
- Không có RAM, ROM, I/O trên CPU chip
- Vd: Intel's x86, Motorola's 680x0



Hệ thống vi xử lý

Vi điều khiển

- Là máy tính mini
- Có RAM, ROM, I/O ports trên CPU chip
- Vd: Motorola's 6811, Intel's 8051, Zilog's Z80, & PIC 16X



tất cả bên trong 1 chip

Vi điều khiển

Vi xử lý & Vi điều khiển

Vi xử lý

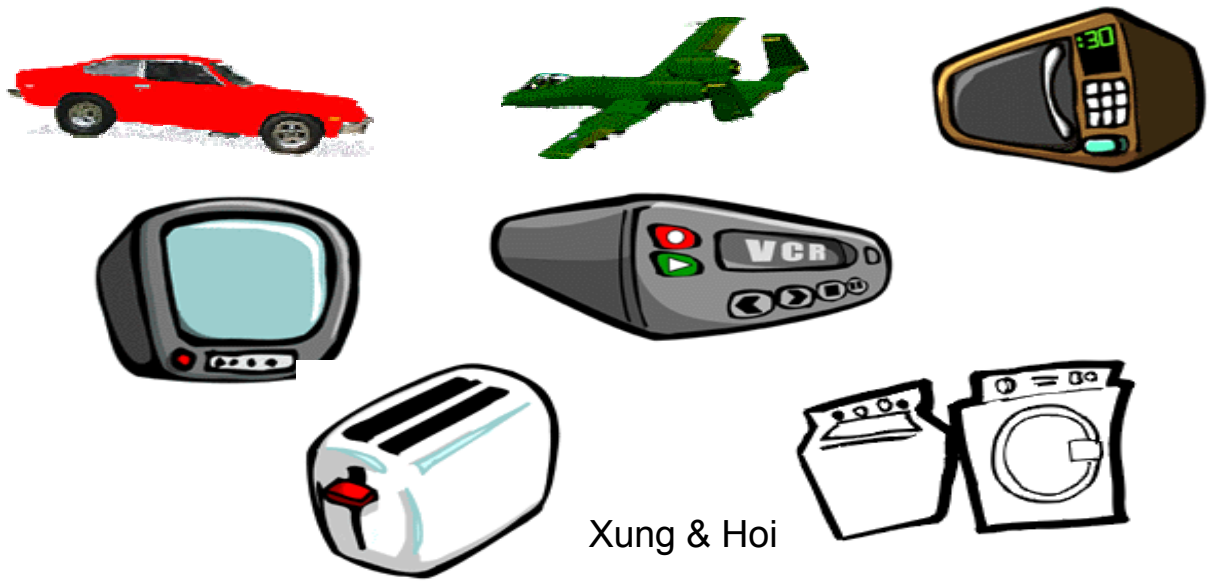
- CPU chip riêng biệt. RAM, ROM, I/O, Timer bên ngoài
- Lượng ROM, RAM, I/O Ports tùy ý
- Giá thành cao
- Đa năng
- Đa mục đích

Vi điều khiển

- CPU, RAM, ROM, I/O & Timer nằm trên cùng 1 chip
- Cố định lượng ROM, RAM, I/O Ports trên chip
- Thích hợp cho các ứng dụng:
  - ✓giá cả thấp
  - ✓năng lượng tiêu thụ thấp
  - ✓không gian hạn chế
- Đơn mục đích

Hệ thống nhúng (Embedded System)

- Bộ xử lý được nhúng (embedded) vào một ứng dụng cụ thể
- Một sản phẩm nhúng chỉ sử dụng VXL,VĐK, FPGA, DSP... để thực thi 1 công việc duy nhất
- Chỉ có một phần mềm ứng dụng & thông thường được nộp trong ROM
- vd：printer, keyboard, video game player...

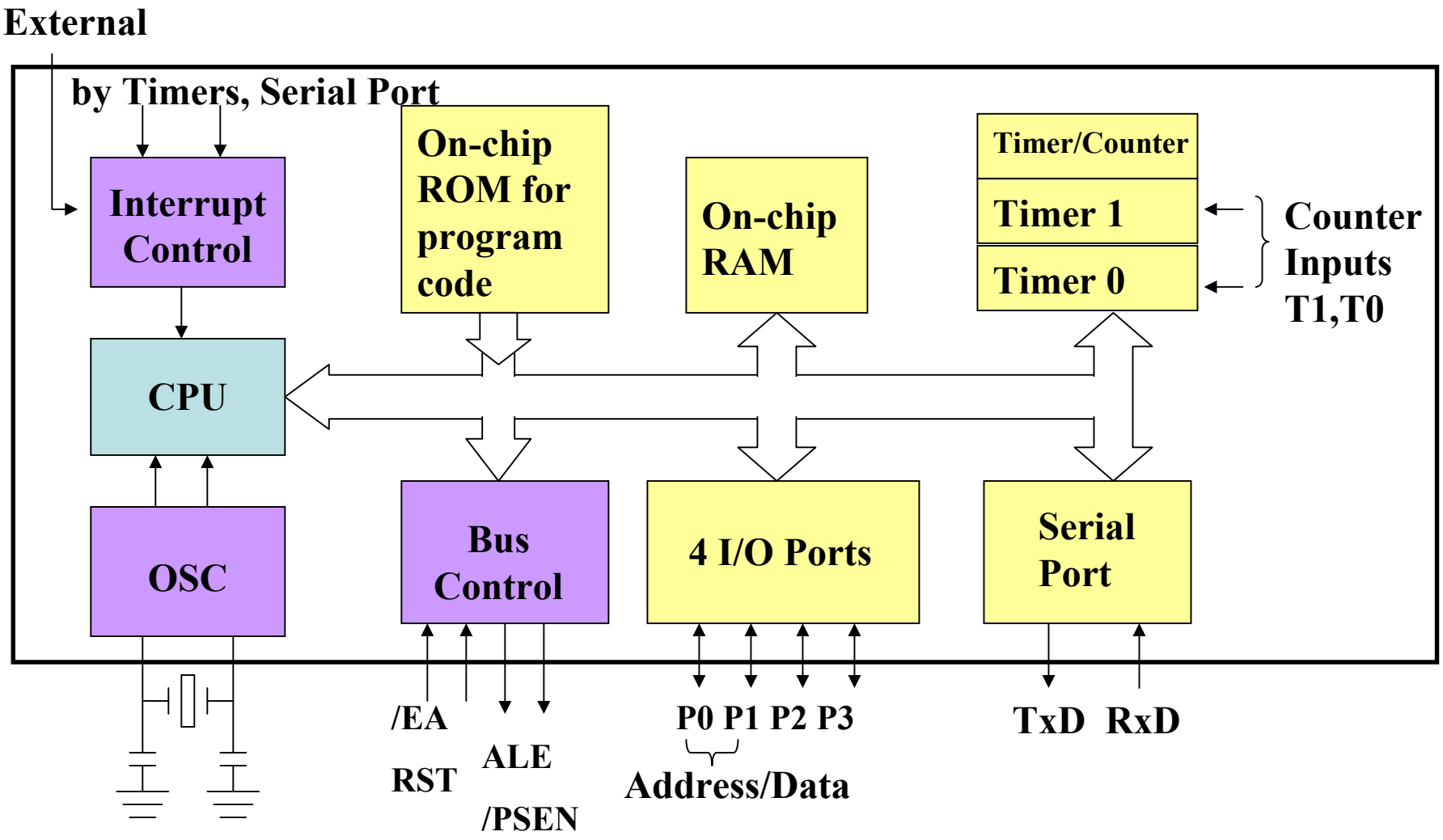


3 tiêu chí chọn vi điều khiển

1. Đáp ứng yêu cầu về nhiệm vụ và giá thành thi công
   - Tốc độ, lượng bộ nhớ, cổng I/O, timers, kích cỡ, đóng gói, năng lượng tiêu thụ
   - Dễ nâng cấp
   - Giá thành
2. Các công cụ phát triển phần mềm
   - Assember, bộ sửa lỗi, trình dịch C, mô phỏng, hỗ trợ kỹ thuật
3. Thị trường cung cấp sản phẩm tin cậy

1. Giới thiệu
2. Sơ đồ khối và chân (block and pin diagrams)
3. Tổ chức bộ nhớ
4. Các thanh ghi chức năng đặc biệt (SFR)
5. Dao động và hoạt động reset
6. Tập lệnh
7. Các mode định địa chỉ
8. Lập trình IO
9. Tạo trễ
10. Lập trình Timer/Counter
11. Lập trình giao tiếp nối tiếp
12. Lập trình ngắt
13. Lập trình hợp ngữ

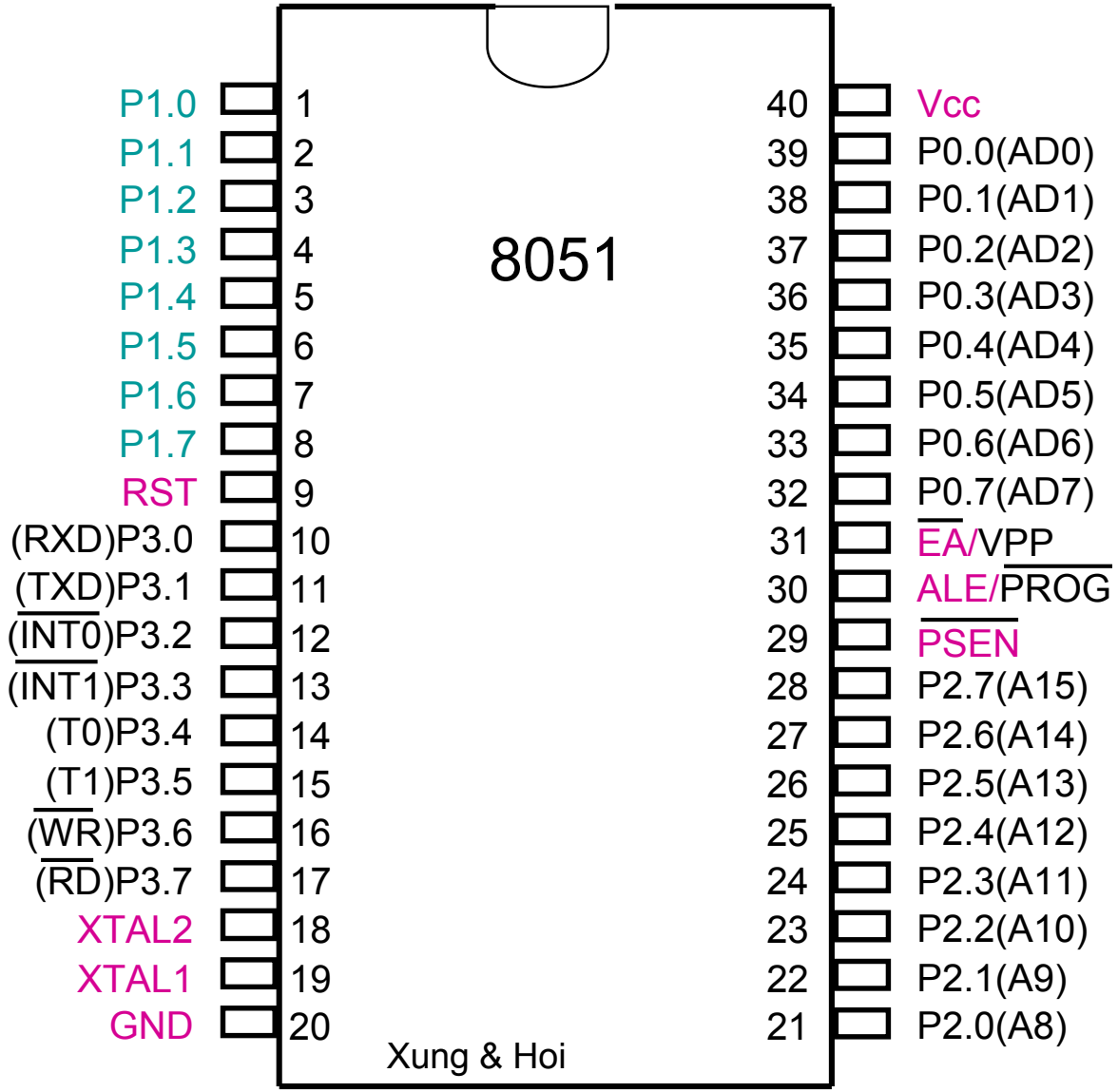
2-1. Sơ đồ khối


External
by Timers, Serial Port
Interrupt Control
On-chip ROM for program code
On-chip RAM
Timer/Counter
Timer 1
Timer 0
Counter Inputs T1,T0
CPU
OSC
Bus Control
4 I/O Ports
Serial Port
/EA
RST
ALE
/PSEN
P0 P1 P2 P3
Address/Data
TxD RxD

2-2. So sánh các thành viên họ 8051

Specification	8031	8051	8751	8951	8052	8752
Code Mem on chip (bytes)	0K	4K ROM	4K EPROM	4K Flash	8K ROM	8K EPROM
Data Mem on chip (bytes)	128	128	128	128	256	256
Timers	2	2	2	2	3	3
I/O pins	32	32	32	32	32	32
Serial Port	1	1	1	1	1	1
Interrupt sources	5	5	5	5	6	6

2-3. Sơ đồ chân


P1.0 1
P1.1 2
P1.2 3
P1.3 4
P1.4 5
P1.5 6
P1.6 7
P1.7 8
RST 9
(RXD)P3.0 10
(TXD)P3.1 11
(INT0)P3.2 12
(INT1)P3.3 13
(T0)P3.4 14
(T1)P3.5 15
(WR)P3.6 16
(RD)P3.7 17
XTAL2 18
XTAL1 19
GND 20
8051
40 Vcc
39 P0.0(AD0)
38 P0.1(AD1)
37 P0.2(AD2)
36 P0.3(AD3)
35 P0.4(AD4)
34 P0.5(AD5)
33 P0.6(AD6)
32 P0.7(AD7)
31 EA/VPP
30 ALE/PROG
29 PSEN
28 P2.7(A15)
27 P2.6(A14)
26 P2.5(A13)
25 P2.4(A12)
24 P2.3(A11)
23 P2.2(A10)
22 P2.1(A9)
21 P2.0(A8)

Các chân 8051 (phần 1)

- Vcc (chân 40):
  - ✓Cung cấp nguồn cho chip
  - ✓+5V
- GND (chân 20): đất
- XTAL1 & XTAL2 (chân 19, 18)
  - ✓2 chân cung cấp xung clock ngoài
  - ✓Cách 1: dao động dùng thạch anh
  - ✓Cách 2: dao động từ nguồn xung clock TTL bên ngoài
  - ✓Quan hệ giữa chu kỳ máy và XTAL

Các chân 8051 (phần 2)

- RST (chân 9): reset
  - ✓input & kích hoạt mức cao
    - ✓Để đảm bảo hoạt động reset xảy ra, xung kích khởi phải kéo dài ít nhất là 2 chu kỳ máy
  - ✓Giá trị các thanh ghi chịu tác động bởi hoạt động reset, xem bảng trong phần 5 bài giảng
  - ✓Mạch reset có chống rung

Các chân 8051 (phần 3)

- /EA (chân 31): External Access
  - ✓/EA nối mass chỉ định rằng code lưu trên bộ nhớ ngoài
  - ✓/PSEN & ALE dùng cho ROM ngoài
  - ✓Với 8051, 8031, 8032 thì /EA nối Vcc
  - ✓"/": chỉ định tác động mức thấp
- /PSEN (chân 29): Program Store Enable
  - ✓Output, cho phép truy xuất bộ nhớ chương trình ngoài
  - ✓Nối tới chân /OE của ROM/EPROM
  - ✓Khi thực thi chương trình ở ROM nội, /EA được gán mức 1, (khi đó /PSEN được tự động giữ ở mức 1)

Các chân 8051 (phần 4)

- ALE (pin 30)：Address Latch Enable
  - Là chân output cho phép chốt địa chỉ để giải đa hợp (de-multiplexing) bus dữ liệu và bus địa chỉ
  - ALE xuất tín hiệu để chốt địa chỉ (byte thấp địa chỉ 16-bit) vào 1 thanh ghi ngoài trong suốt nửa đầu của chu kỳ bộ nhớ (memory cycle). Trong nửa chu kỳ bộ nhớ còn lại, P0 sẽ xuất/nhập dữ liệu
  - ALE có $f=1/6f_{clock}$
  - Có 1 ngoại lệ: trong thời gian thực thi lệnh MOVX, một xung ALE bị bỏ qua
- Cổng I/O: P0, P1, P2, & P3. Mỗi cổng: 8 chân.

Các cổng I/O

- 4 cổng I/O

Port 0 (chân 32-39) : P0 (P0.0～P0.7)

Port 1 (chân 1-8) : P1 (P1.0～P1.7)

Port 2 (chân 21-28) : P2 (P2.0～P2.7)

Port 3 (chân 10-17) : P3 (P3.0～P3.7)

Mỗi cổng có 8 chân

Đánh tên P0.X (X=0,1,...,7), P1.X, P2.X, P3.X

Ex : P0.0 là bit 0 (LSB) của P0

Ex : P0.7 là bit 7 (MSB) của P0

8 bits này cấu thành 1 byte

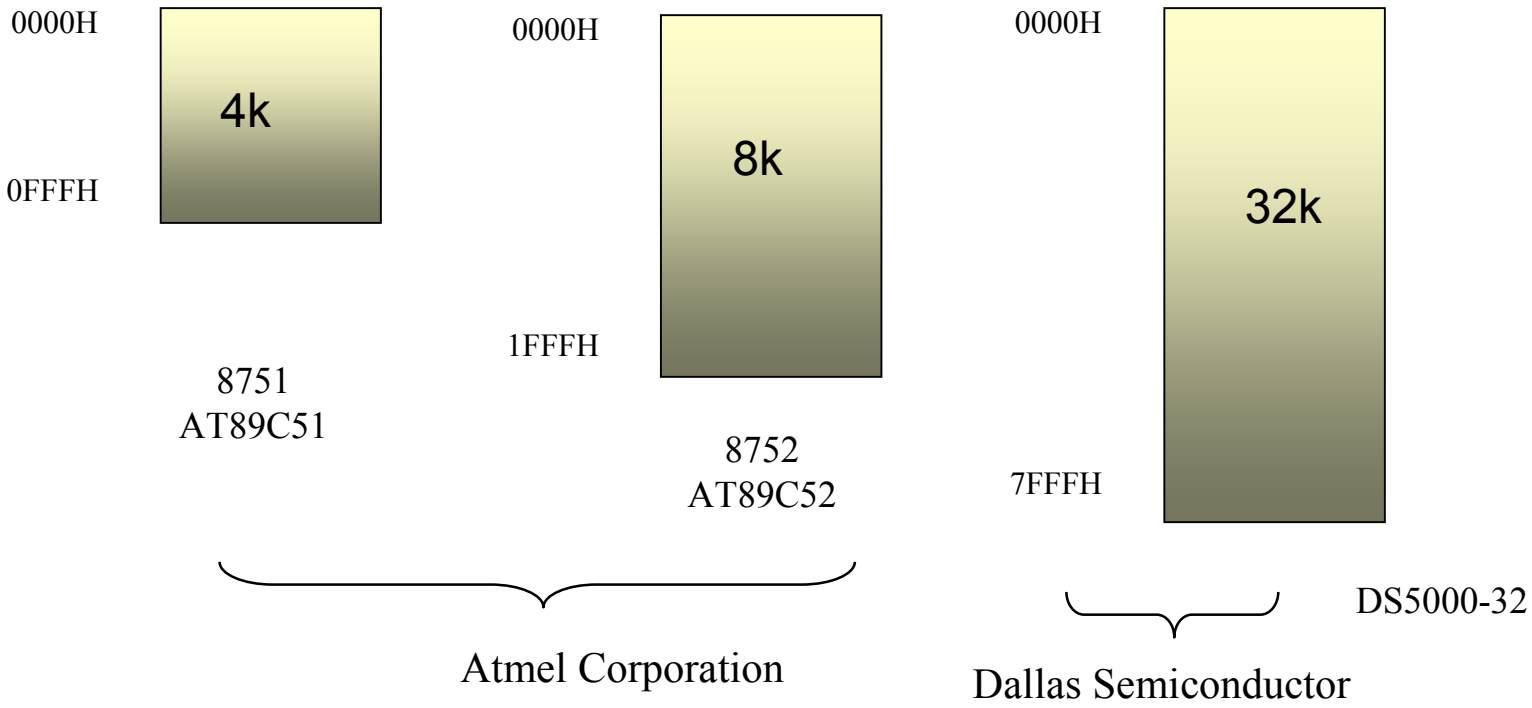
Mỗi cổng có thể được dùng như input hay output

1. Giới thiệu
2. Sơ đồ khối và chân
3. Tổ chức bộ nhớ (Memory Organization)
4. Các thanh ghi chức năng đặc biệt
5. Dao động và hoạt động reset
6. Tập lệnh
7. Các mode định địa chỉ
8. Lập trình IO
9. Tạo trễ
10. Lập trình Timer/Counter
11. Lập trình giao tiếp nối tiếp
12. Lập trình ngắt
13. Lập trình hợp ngữ

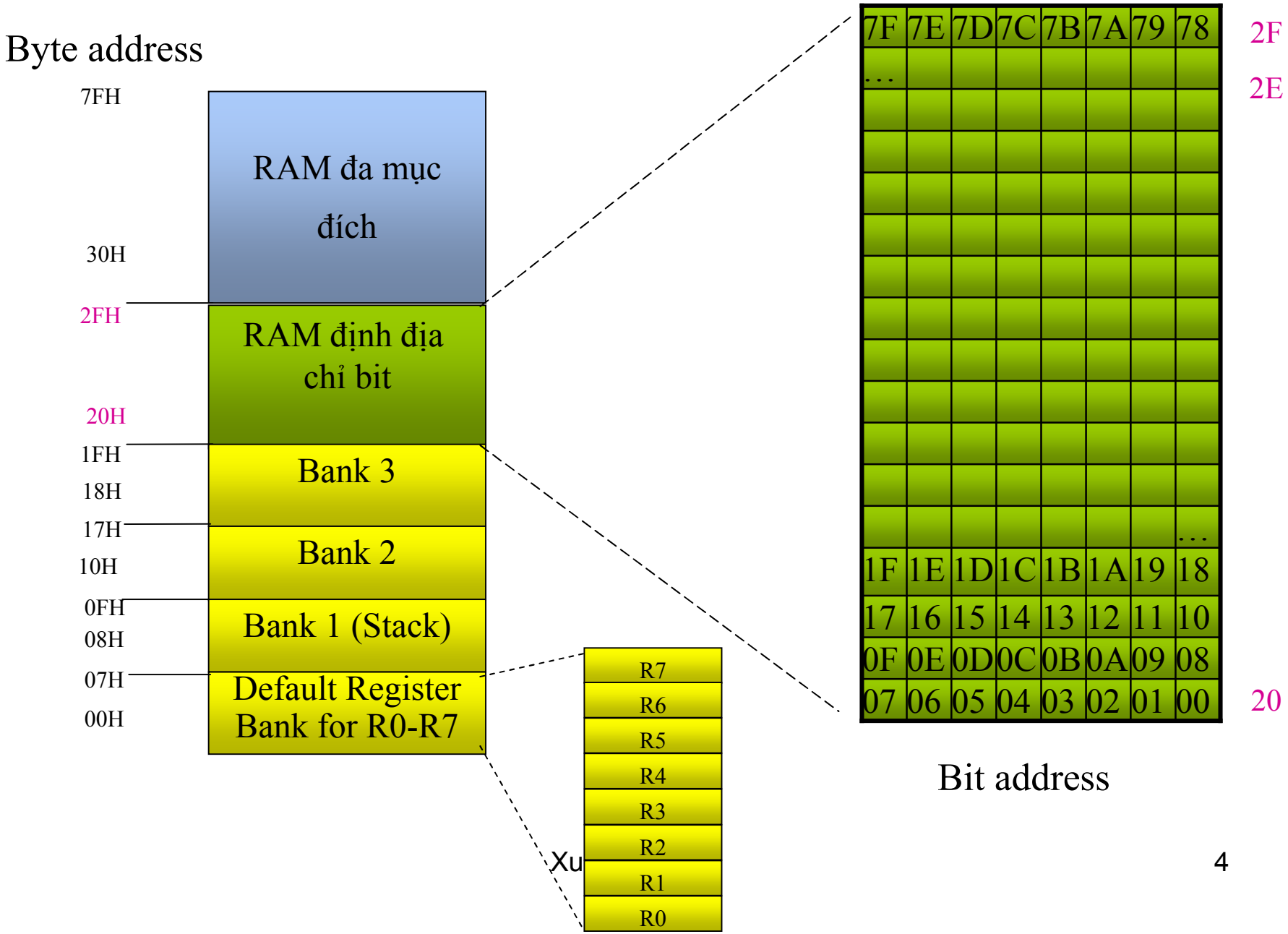
Hai đặc tính cần lưu ý

1. Các thanh ghi và các I/O port được định địa chỉ theo kiểu ánh xạ bộ nhớ (memory mapped) & do đó được truy xuất như 1 vị trí nhớ trong bộ nhớ
2. Stack là trên RAM nội thay vì trên RAM ngoài như đối với các bộ VXL

3-1. Không gian bộ nhớ ROM nội



3-2. Không gian bộ nhớ RAM nội



3-2-1. Vùng RAM đa mục đích

- Truy xuất tự do 80 bytes này theo kiểu định địa chỉ trực tiếp hay gián tiếp

- MOV A, 5FH

  Hoặc:

  MOV R0, #5FH
  MOV A, @R0

3-2-2. Vùng RAM định địa chỉ bit

- Truy xuất các bit riêng rẽ là 1 đặc trưng của VĐK. Các bit có thể được set, xóa, AND, OR … chỉ bằng 1 lệnh so với 1 chuỗi lệnh của VXL
- Các port cũng được định địa chỉ bit
- Vd: để set bit 7FH bằng 1, ta viết:

VĐK:

```
SETB 7FH
```

VXL:

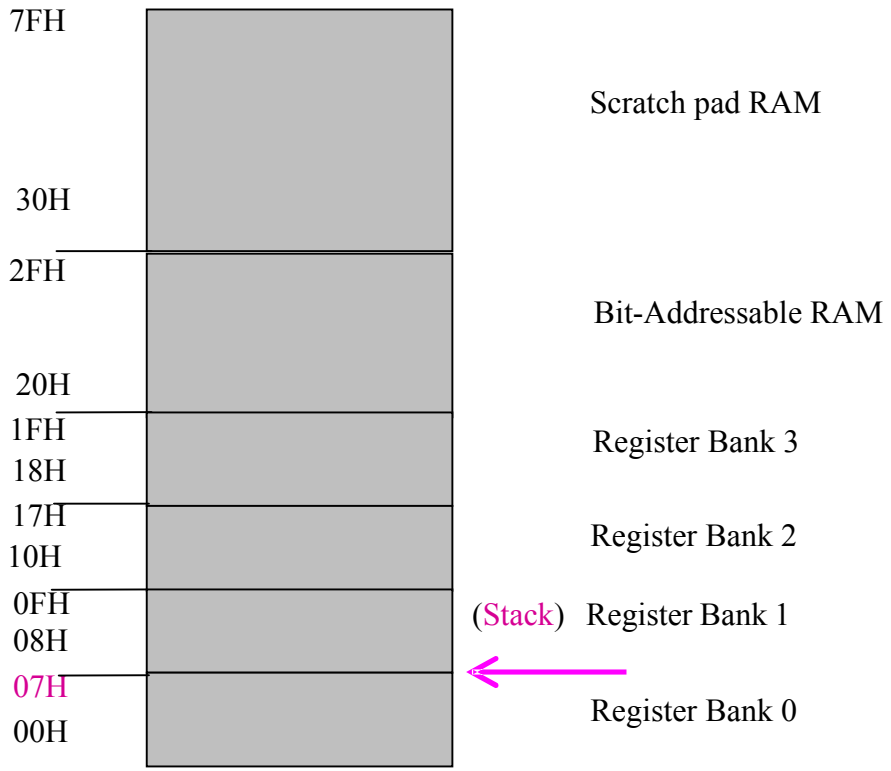
```
MOV A,2FH          ; đọc cả byte
ORL A,#10000000B   ; set bit
MOV 2FH,A          ; ghi trở lại cả byte
```

3-2-3. Các dãy thanh ghi (register banks)

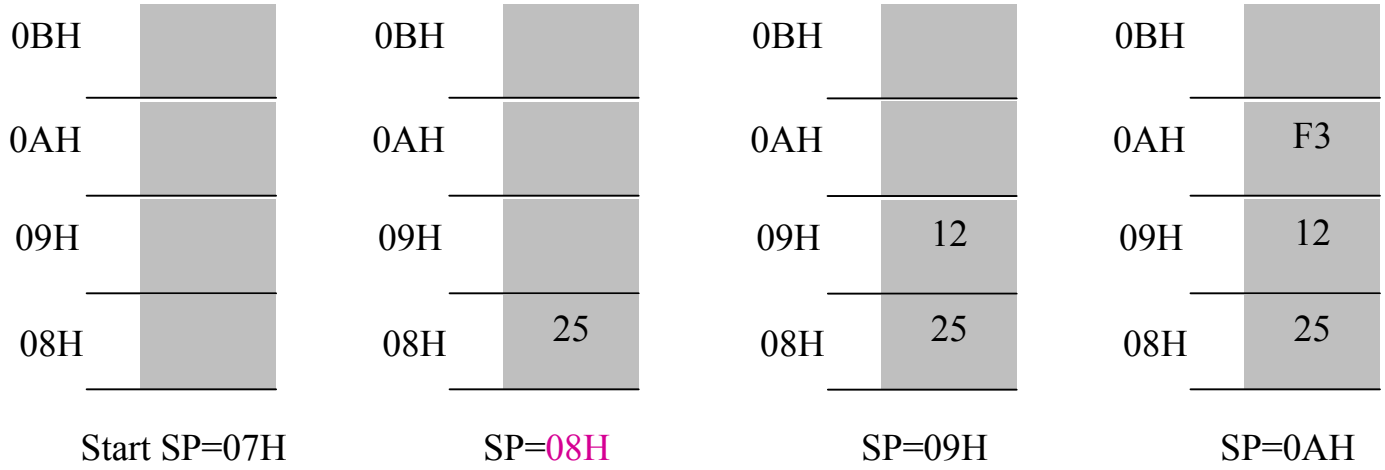
- Các lệnh dùng thanh ghi là những lệnh ngắn & thực hiện nhanh hơn
  - MOV A,R5 ; 1 byte
  - MOV A,05H ; 2 bytes
- Các dữ liệu thường dùng nên chứa ở các thanh ghi
- Ý tưởng các dãy thanh ghi cho phép *chuyển đổi ngữ cảnh* nhanh và hiệu quả ở các module độc lập nhau của phần mềm

3-2-4. Ngăn xếp

- Thanh ghi dùng truy cập ngăn xếp gọi là **SP** (stack pointer)
- **SP** là thanh ghi 8 bit: giá trị từ 00 → FFH.
- Khi được cấp nguồn hay sau khi reset, **SP**=07H



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



1. Giới thiệu
2. Sơ đồ khối và chân
3. Tổ chức bộ nhớ
4. Các thanh ghi chức năng đặc biệt (SFR)
5. Dao động và hoạt động reset
6. Tập lệnh
7. Các mode định địa chỉ
8. Lập trình IO
9. Tạo trễ
10. Lập trình Timer/Counter
11. Lập trình giao tiếp nối tiếp
12. Lập trình ngắt
13. Lập trình hợp ngữ

SFRs

80	P0	SP	DPL	DPH				PCON
88	TCON	TMOD	TL0	TL1	TH0	TH1		
90	P1							
98	SCON	SBUF						
A0	P2							
A8	IE							
B0	P3							
B8	IP							
C0								
C8								
D0	PSW							
D8								
E0	ACC							
E8								
F0	B							
F8								

Blue background are I/O port SFRs
Yellow background are control SFRs
Green background are other SFRs

Xem thêm Hình 2.6 trang 26

CPU registers:

- ACC : Accumulator.
- B : B register.
- PSW : Program Status Word.
- SP : Stack Pointer.
- DPTR : Data Pointer (DPH, DPL).

Interrupt control:

-IE : Interrupt Enable.
-IP : Interrupt Priority.

I/O Ports:

- P0 : Port 0.
- P1 : Port 1.
- P2 : Port 2.
- P3 : Port 3.

TImers:

- TMOD : Timer mode.
- TCON : Timer control.
- TH0 : Timer 0 high byte.
- TL0 : Timer 0 low byte.
- TH1 : Timer 1 high byte.
- TL1 : Timer 1 low byte.

Serial I/O:

- SCON : Serial port control.
- SBUF : Serial data registers.

Other:

- PCON : Power control & misc.

4-1. Thanh ghi PSW

- Thanh ghi PSW (bit addressable)

C	AC	F0	RS1	RS0	OV	--	P

Cờ nhớ	**PSW.7**	**C**
Cờ nhớ phụ	**PSW.6**	**AC**
Available to user for general purpose	**PSW.5**	**--**
Bit chọn dãy thanh ghi 1	**PSW.4**	**RS1**
Bit chọn dãy thanh ghi 0	**PSW.3**	**RS0**
Cờ tràn	**PSW.2**	**OV**
Dự trữ - *User define bit*	**PSW.1**	**--**
Cờ chẵn lẻ	**PSW.0**	**P**

RS1	RS0	Register Bank	Byte Address of R0-R7
0	0	0	00H-07H
0	1	1	08H-0FH
1	0	2	10H-17H
1	1	3	18H-1FH

D7	D6	D5	D4	D3	D2	D1	D0

Địa chỉ bit

- C (carry flag) - Cờ nhớ
  - Được set bằng 1 nếu có số nhớ từ phép cộng bit 7 hay có số mượn mang đến bit 7
    - MOV A,#FFH
    - ADD A,#1
  - Còn được dùng như 1 thanh ghi 1 bit đối với các lệnh logic thao tác trên các bit
    - ANL C,25H
- AC (auxiliary carry) – Cờ nhớ phụ
  - Được set bằng 1 nếu có số nhớ từ bit 3 sang bit 4
- RS1, RS0 - Các bit chọn dãy (bank) thanh ghi
  - Dùng để xác định dãy thanh ghi tích cực
  - Chúng được xóa khi reset

SETB RS1 ≡ SETB 0D4H	
SETB RS0	MOV A,R7

- OV (overflow flag) - Cờ tràn
  - Set bằng 1 sau phép toán cộng hoặc trừ nếu có xuất hiện 1 tràn số học. Khi các số có dấu được cộng hoặc trừ, phần mềm có thể kiểm tra bit tràn OV để xác định KQ có nằm trong tầm hay không
  - Với các số có dấu, KQ nhỏ hơn -128 hoặc lớn hơn +127 sẽ set cờ OV = 1. Với các số không dấu, OV=1 khi KQ vượt quá 255
  - VD:
    - 0F thập phân 15
    - +7F +127
    - ____ _____
    - 8E 142
    - 8EH biểu diễn -114 không đúng với KQ mong muốn là 142 nên OV = 1

- P (parity) - Cờ chẵn lẻ
  - Kiểm tra chẵn lẻ cho thanh chứa A
  - Số các bit 1 trong thanh chứa A cộng với bit P luôn luôn chẵn
    - MOV A,#10101101B
    - →P=1
  - Bit chẵn lẻ được sử dụng kết hợp với các chương trình xuất/nhập nối tiếp trước khi truyền dữ liệu hoặc để kiểm tra chẵn lẻ sau khi nhận dữ liệu

VD1: MOV A,#88H ADD A,#93H	VD2: MOV A,#9CH ADD A,#64H
VD3: MOV A,#38H ADD A,#2FH	VD4: MOV A,#FFH ADD A,#1

VD:

MOV A,#88H

ADD A,#93H

```
  88          10001000
 +93         +10010011
 ---         ----------
 11B         00011011
```

C=1 AC=0 P=0

VD:

MOV A,#9CH

ADD A,#64H

```
  9C          10011100
 +64         +01100100
 ---         ----------
 100         00000000
```

C=1 AC=1 P=0

VD:

MOV A,#38H

ADD A,#2FH

```
  38          00111000
 +2F         +00101111
 ---         ----------
 67          01100111
```

C=0 AC=1 P=1

VD:

MOV A,#FFH

ADD A,#1

A=00H; C=1; AC=1;

Những lệnh ảnh hưởng đến các bit cờ

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		

X có thể là 1 hoặc 0

4-2. Thanh ghi B

- B được dùng với thanh chứa A trong các phép toán nhân, chia
- MUL A,B ; nhân 2 số 8-bit không dấu chứa trong A & B, KQ 16-bit chứa vào cặp thanh ghi B:A (B chứa byte cao)
- DIV AB ; chia A bởi B, thương số cất trong A, dư cất trong B
- B còn được xử lý như thanh ghi nháp
- B được định địa chỉ bit

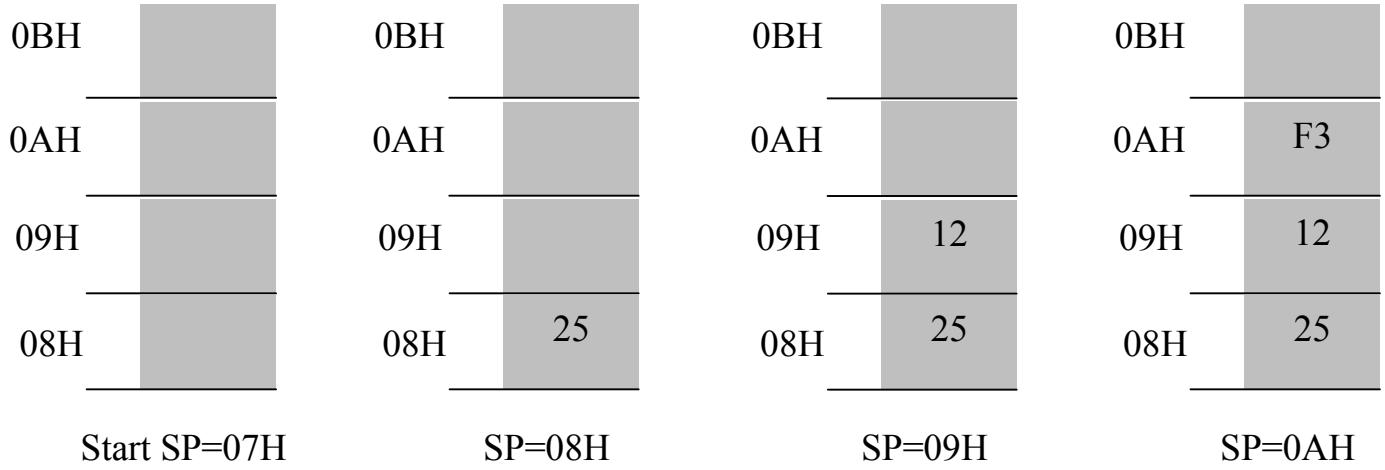
F7	F6	F5	F4	F3	F2	F1	F0

4-3. Con trỏ ngăn xếp (SP)

- SP chứa địa chỉ của dữ liệu hiện đang ở đỉnh của stack
- Các lệnh liên quan đến stack bao gồm lệnh cất dữ liệu vào stack (làm tăng SP trước khi ghi dữ liệu) và lệnh lấy dữ liệu khỏi stack (giảm SP)
- Muốn stack bắt đầu ở 60H:
  - MOV SP,#5FH
  - Thì vùng stack sẽ là 32 byte trên 8051 vì địa chỉ cao nhất của RAM nội là 7FH
  - 5FH được dùng vì SP tăng lên 60H trước khi thao tác cất vào stack đầu tiên được thực thi

- Nếu không khởi động SP, nội dung mặc định là 07H (để duy trì sự tương thích với 8048) → thao tác cất vào stack đầu tiên sẽ lưu dữ liệu vào vị trí nhớ có địa chỉ 08H
- Trong trường hợp này, nếu phần mềm ứng dụng không khởi động SP, dãy thanh ghi 1 (và có lẽ 2, 3) sẽ không còn hợp lệ vì chúng được sử dụng làm stack
- PUSH & POP cất dữ liệu vào stack và lấy dữ liệu từ stack
- ACALL, LCALL, RET, RETI cất và phục hồi bộ đếm chương trình PC

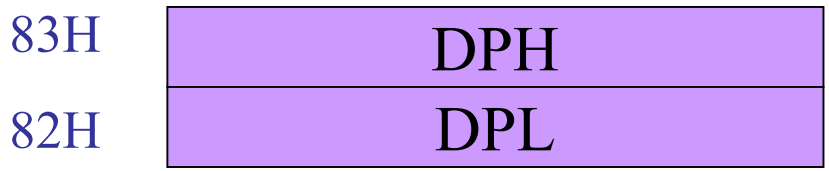
VD:
```
MOV  R6,#25H
MOV  R1,#12H
MOV  R4,#0F3H
PUSH 6
PUSH 1
PUSH 4
```



4-4. Con trỏ dữ liệu (DPTR)

- DPTR được dùng để truy xuất bộ nhớ chương trình ngoài hoặc bộ nhớ dữ liệu ngoài
- VD:

```
MOV     A,#55H
MOV     DPTR,#1000H
MOVX    @DPTR,A
```



4-5. Các thanh ghi I/O port: P0, P1, P2, P3

- Tất cả port đều được định địa chỉ bit
- VD: (điều khiển motor nối bit P1.0)

```
SETB    P1.0
CLR     P1.0
```

- Đoạn chương trình kiểm tra trạng thái BUSY của thiết bị:

```
WAIT:   JB P1.5, WAIT
```

4-6. Các thanh ghi định thời

- 8051 có 2 bộ đếm/định thời 16-bit để định các khoảng thời gian hoặc đếm các sự kiện
- Hoạt động của bộ định thời được thiết lập bởi:
  - TMOD (Timer Mode Register)
  - TCON (Timer Control Register)

4-7. Các thanh ghi của port nối tiếp

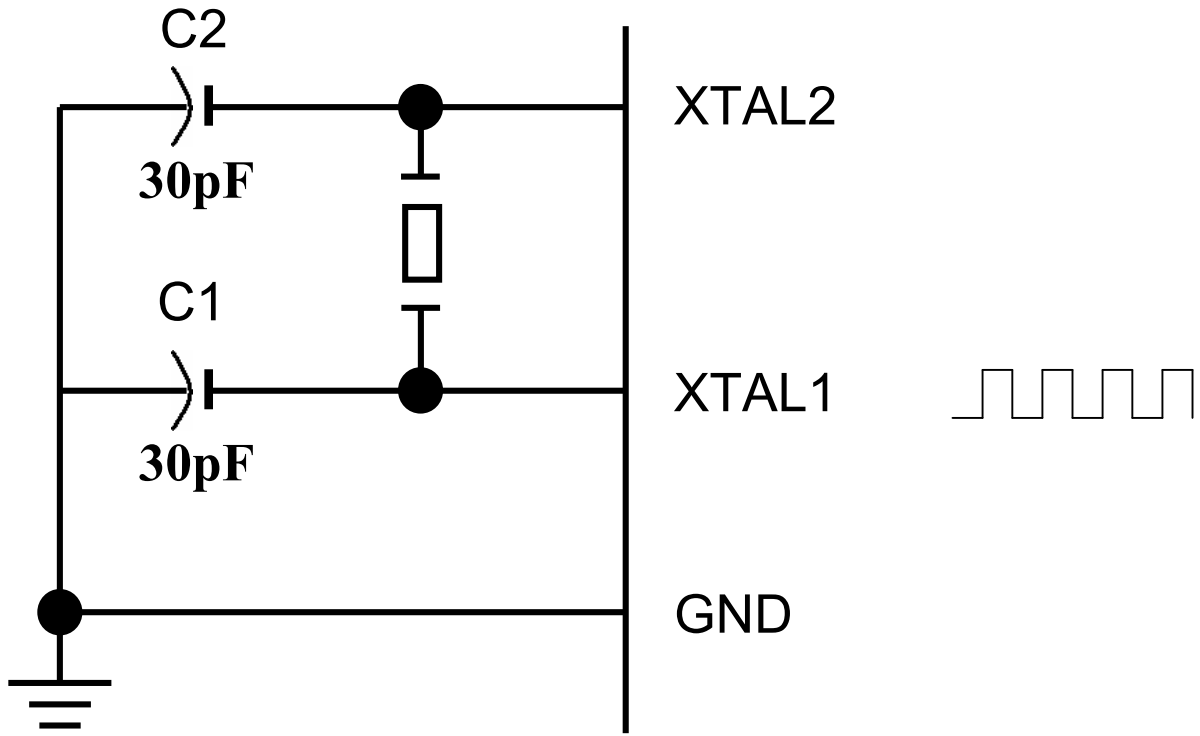
- 8051 có 1 port nối tiếp để truyền thông với các thiết bị nối tiếp
- SBUF (Serial Data Buffer): lưu trữ dữ liệu truyền và nhận
- SCON (Serial Port Control Register): chọn chế độ hoạt động

4-8. Các thanh ghi ngắt

- IE (interrupt enable)
- IP (interrupt priority)

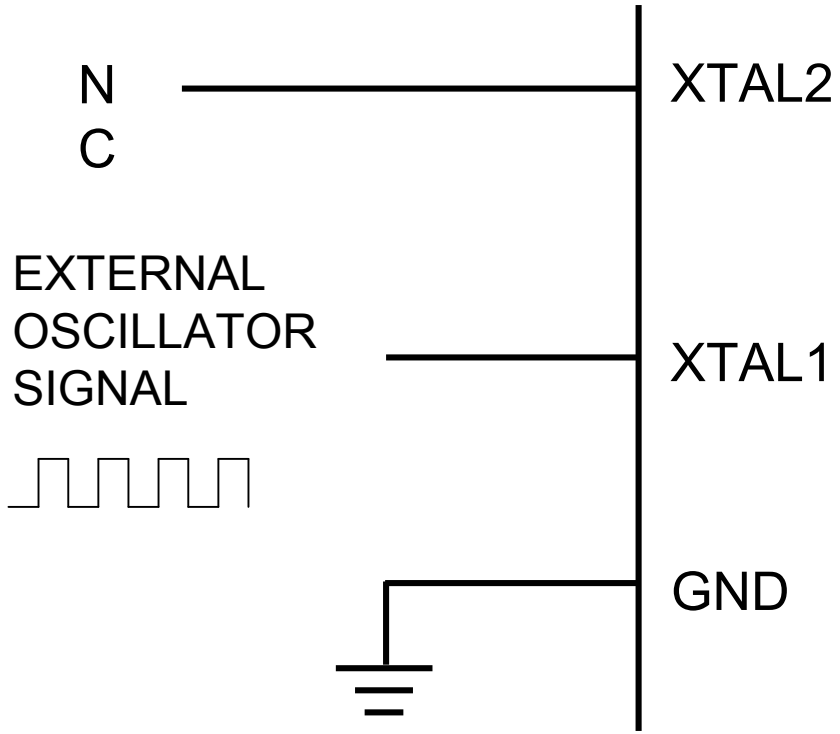
1. Giới thiệu
2. Sơ đồ khối và chân
3. Tổ chức bộ nhớ
4. Các thanh ghi chức năng đặc biệt
5. Dao động và hoạt động reset (oscillator and reset)
6. Tập lệnh
7. Các mode định địa chỉ
8. Lập trình IO
9. Tạo trễ
10. Lập trình Timer/Counter
11. Lập trình giao tiếp nối tiếp
12. Lập trình ngắt
13. Lập trình hợp ngữ

5-1. Dao động dùng thạch anh



5-2. Dao động từ nguồn xung clock ngoài

- Dùng bộ dao động TTL
- Chú ý XTAL2: NC



5-3. Dao động & chu kỳ máy

☛Tìm chu kỳ máy cho

(a) XTAL = 11.0592 MHz

(b) XTAL = 16 MHz.

Giải đáp:

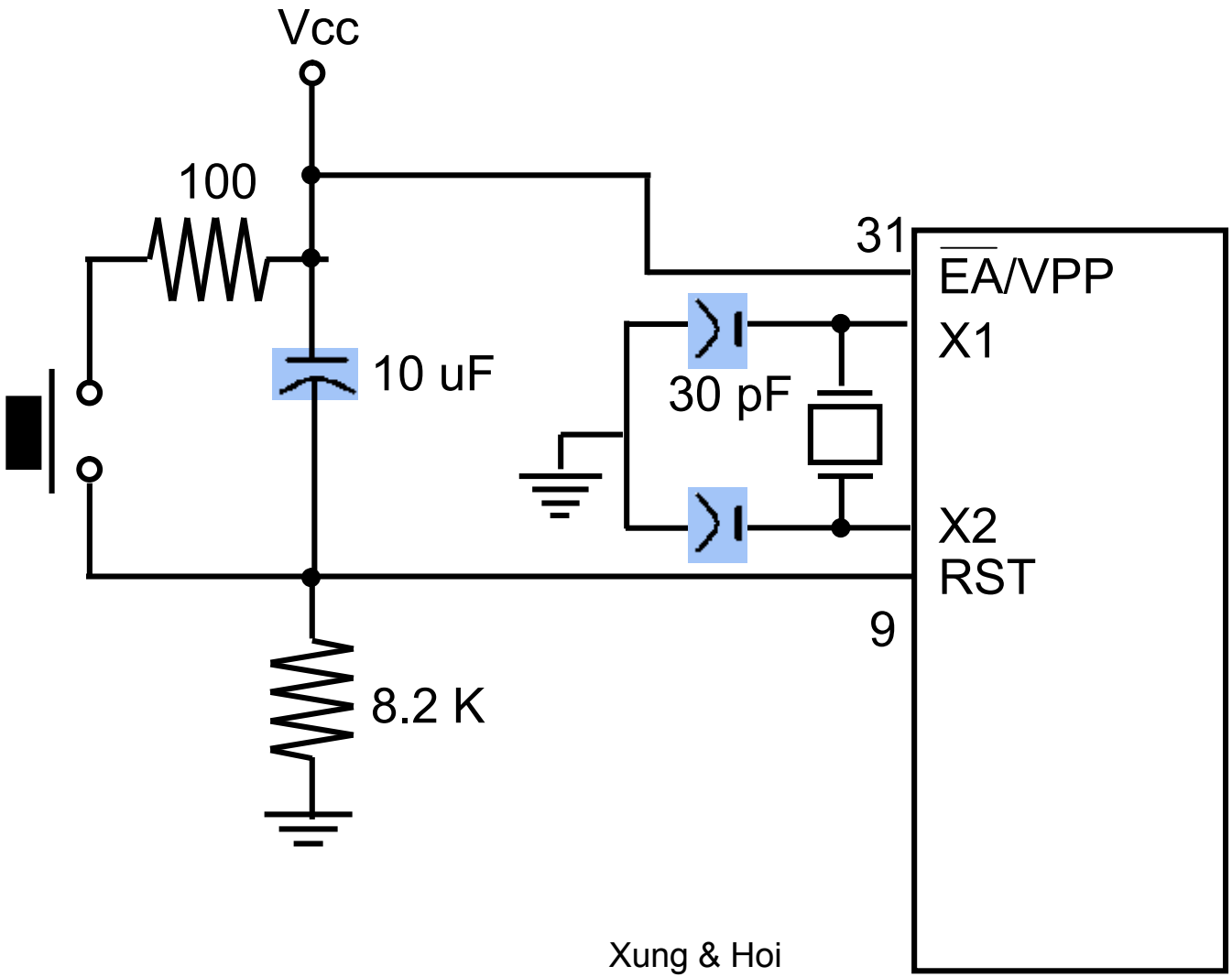
(a) 11.0592 MHz / 12 = 921.6 kHz;

Chu kỳ máy (machine cycle) = 1 / 921.6 kHz = 1.085 μs

(b) 16 MHz / 12 = 1.333 MHz;

Chu kỳ máy (machine cycle) = 1 / 1.333 MHz = 0.75 μs

5-4. Mạch Reset có chống rung



5-5. Giá trị Reset của các thanh ghi 8051

Thanh ghi	*Giá trị Reset*
PC	**0000H**
A	**00H**
B	**00H**
PSW	**00H**
SP	**07H**
DPTR	**0000H**
P0, P1, P2 ,P3	**FFH**

RAM nôi không bị ảnh hưởng

1. Giới thiệu
2. Sơ đồ khối và chân
3. Tổ chức bộ nhớ
4. Các thanh ghi chức năng đặc biệt
5. Dao động và hoạt động reset
6. Tập lệnh (Instruction Set)
7. Các mode định địa chỉ
8. Lập trình IO
9. Tạo trễ
10. Lập trình Timer/Counter
11. Lập trình giao tiếp nối tiếp
12. Lập trình ngắt
13. Lập trình hợp ngữ

MOV dest, source ; dest = source

```
MOV     A, #72H      ;A=72H
MOV     R4, #62H     ;R4=62H
MOV     A, R4        ;A=62H
MOV     B, 7EH
MOV     P1, A   ;mov A to port 1
```

```
MOV     DPTR, #7634H
MOV     DPL, #34H
MOV     DPH, #76H
```

~~MOV DPTR, A~~

~~MOV Rm, Rn~~

Chú ý 1:

MOV A,#72H ≠ MOV A,72H

8086		**8051**	
MOV	AL,72H	MOV	A,#72H
MOV	BX,72H		
MOV	AL,[BX]	MOV	A,72H

Chú ý 2:

MOV A,R3 ≡ MOV A,3

ADD A, Source ;A=A+Source

```
ADD   A,#6      ;A=A+6
ADD   A,R6      ;A=A+R6
ADD   A,6       ;A=A+[6]  or  A=A+R6
```

ADDC A, source ;A=A+source+CY

```
SETB  C         ;CY=1
ADDC  A,R5      ;A=A+R5+1
```

SUBB A, source ;A=A-source-CY

```
SETB  C         ;CY=1
SUBB  A,R5      ;A=A-R5-1
```

SETB bit ; bit=1
CLR bit ; bit=0

```
SETB  C          ; CY=1
SETB  P0.0       ;bit 0 from port 0 =1
SETB  P3.7       ;bit 7 from port 3 =1
SETB  ACC.2      ;bit 2 from Accumulator =1
SETB  05   ;set bit địa chỉ 05h (tại RAM có địa chỉ byte 20h)
```

Chú ý:

Lệnh xóa bit CLR tương tự lệnh set bit SETB

```
   CLR      C      ;CY=0
```

Song tồn tại 1 ngoại lệ câu lệnh sau đây chỉ cho CLR

```
   CLR      A      ;A=0
```

DEC byte ;byte=byte-1

INC byte ;byte=byte+1

```
        INC     R7
        DEC     A
        DEC     40H         ; [40]=[40]-1
```

CPL A ;1's complement

```
        MOV     A,#55H      ;A=01010101 B
L01:    CPL     A
        MOV     P1,A
        ACALL   DELAY
        SJMP    L01
```

ANL - ORL – XRL dest, source

vd:

```
    MOV         R5,#89H
    ANL         R5,#08H
```

RR A

RL A

RRC A

RLC A

NOP & RET & RETI

Giống như các lệnh của 8086

```
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
```

Các lệnh nhảy có điều kiện:

JZ rel	Jump if A=0
JNZ rel	Jump if A≠0
DJNZ byte, rel	Decrement & jump if byte≠0
CJNE dest, source, rel	Jump if dest ≠ source
JC rel	Jump if CY=1
JNC rel	Jump if CY=0
JB bit, rel	Jump if bit=1
JNB bit, rel	Jump if bit=0
JBC bit, rel	Jump if bit=1 & clear bit

- DJNZ byte, rel-addr

  Viết chương trình xóa thanh ghi A, sau đó cộng 3 vào A mười lần?

→

```
        ORG 0000H
        MOV   A, #0;
        MOV   R2, #10
AGAIN:  ADD   A, #03
        DJNZ  R2, AGAIN ;repeat until R2=0 (10 lần)
        MOV   P1, A
```

CJNE <dest-byte>, <src-byte>, rel

- So sánh và nhảy nếu không bằng. CJNE so sánh 2 giá trị đầu tiên và rẽ nhánh nếu các giá trị của 2 toán hạng không bằng nhau
- C được set bằng 1 nếu giá trị nguyên không dấu của <dest-byte> nhỏ hơn của <src-byte>. Ngược lại, C = 0
- Cả 2 toán hạng không bị ảnh hưởng sau khi thực thi lệnh

VD:

Viết 1 chương trình so sánh R0, R1:
Nếu R0>R1: gửi 1 ra port 2
Else if R0<R1: gửi 0FFH ra port 2
Else gửi 0 ra port 2

Giải đáp:

```
                    CJNE   R0, 1, NOT_EQ
                    …                          ;R0=R1
NOT_EQ:             JC     R0_NHO_HON
                    …                          ;R0>R1
R0_NHO_HON:         …                          ;R0<R1
```

VD:

```
        MOV   A, 79H
Port 1 là ngõ vào có giá trị thay đổi liên tục.
        ….
WAIT:   CJNE  A,P1,WAIT
        …
```

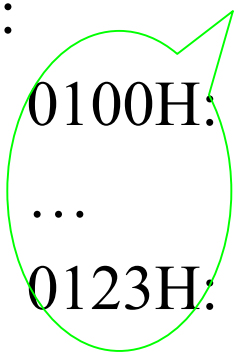
tác dụng gì?

Trả lời: Chương trình lặp lại tại điểm này cho tới khi nhận giá trị 79H tại đầu vào P1.

- **SJMP rel-addr**

  Là lệnh nhảy không điều kiện. Là lệnh nhảy ngắn (2-byte). Byte đầu tiên là opcode, byte thứ 2 là địa chỉ tương đối của đích. Địa chỉ tương đối trong khoảng 00→FFH nhưng chia thành 2 hướng: tới, lui (forward, backward) nên tầm nhảy cho phép là -128 đến +127 bytes trước lệnh và sau lệnh.

- VD:



PC

0100H: SJMP RelativeAddress

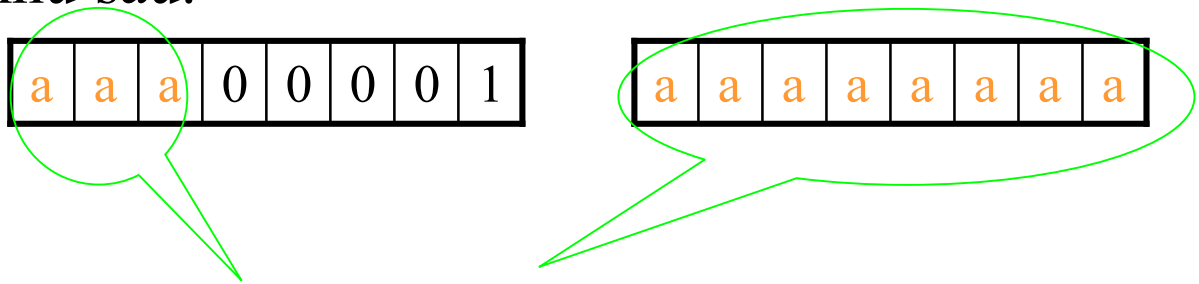
…

0123H: RelativeAddress

→ Byte độ dời của lệnh: 0123H-0102H=21H là độ dời tương đối

AJMP addr11

Là lệnh nhảy không điều kiện. Nhảy đến địa chỉ tuyệt đối. Đích nhảy đến phải ở trong vùng 2K của bộ nhớ chương trình với byte đầu tiên của lệnh theo sau lệnh AJMP. Mã đối tượng như sau:



A10-A8 & A7-A0 của địa chỉ đích

LJMP addr16

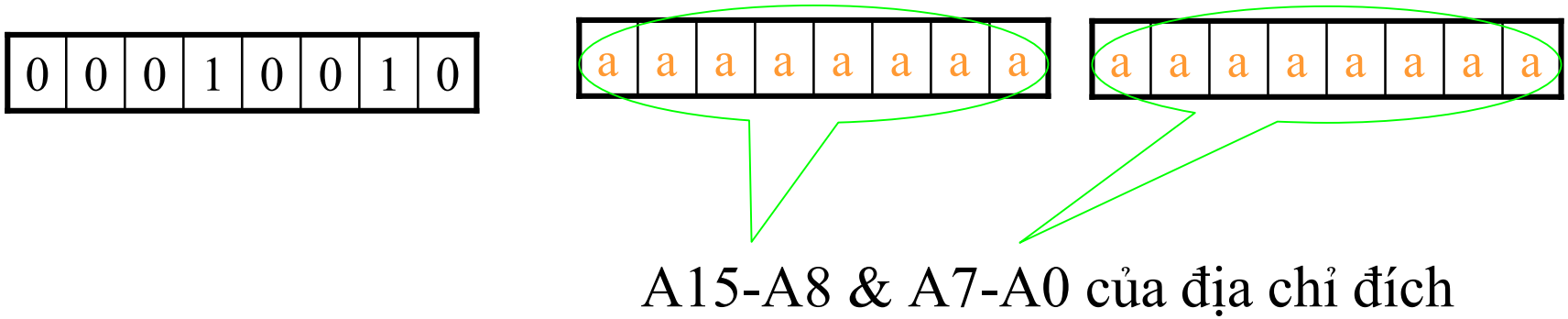
Là lệnh nhảy không điều kiện (3 bytes). Byte đầu tiên là opcode, byte 2 & 3 chứa địa chỉ đích 16-bit. Tầm nhảy từ 0000→FFFFH

CALL

1 lệnh chuyển điều khiển khác là CALL, dùng để gọi 1 chương trình con (subroutine).

LCALL

Là lệnh 3 bytes. LCALL dùng để gọi chương trình con nằm bất kỳ đâu trong khoảng 64K byte không gian địa chỉ của 8051.



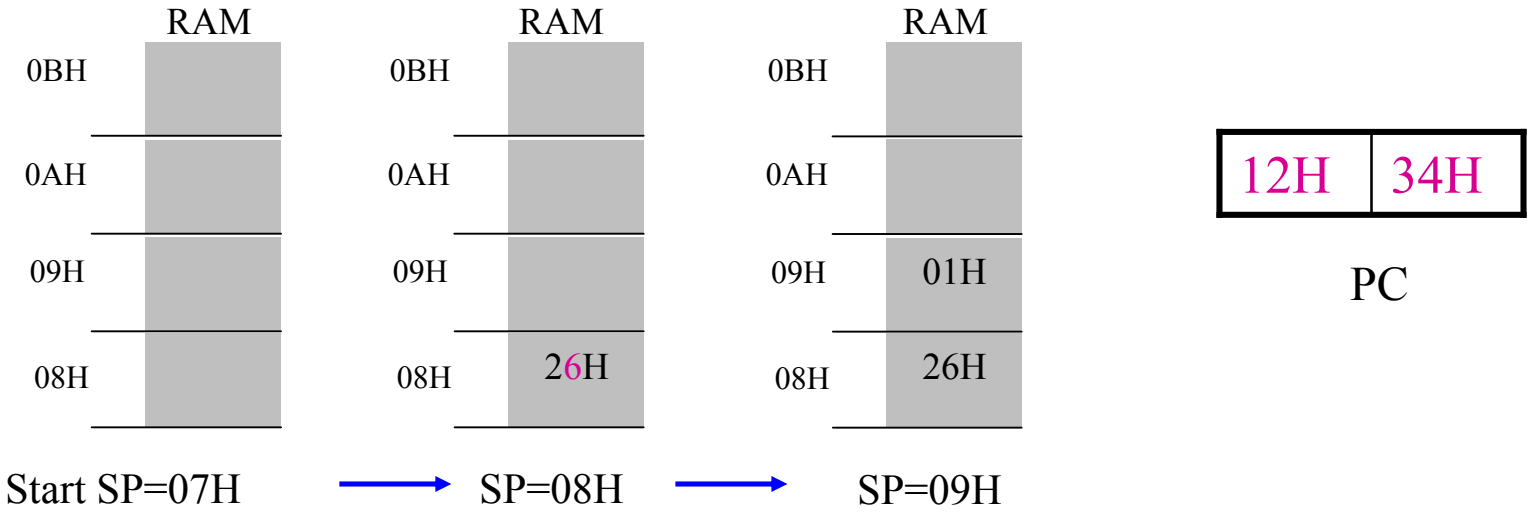
VD:

Ban đầu không khởi động stack pointer (SP).

Nhãn Subroutine đặt tại vị trí 1234H trong bộ nhớ chương trình.

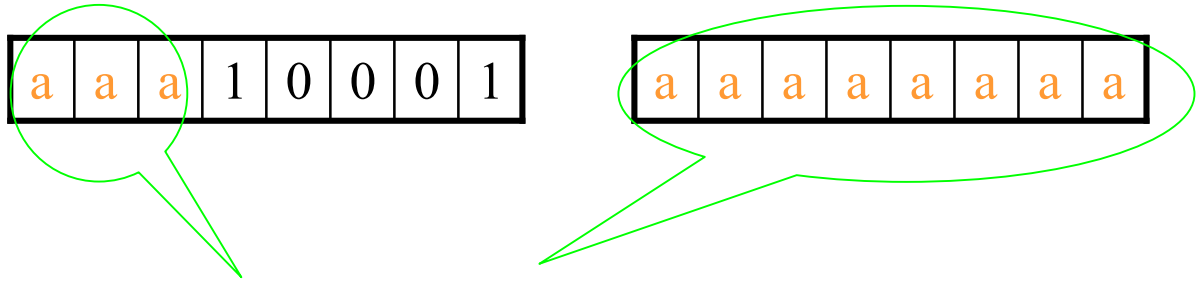
Tại 0123H thực hiện: LCALL Subroutine

Mô tả hoạt động sau khi thực hiện lệnh trên?



ACALL

Là lệnh 2-byte. Chương trình con phải ở trong phạm vi 2K của bộ nhớ chương trình so với byte đầu tiên của lệnh theo sau ACALL.



1. Giới thiệu
2. Sơ đồ khối và chân
3. Tổ chức bộ nhớ
4. Các thanh ghi chức năng đặc biệt (SFR)
5. Dao động và hoạt động reset
6. Tập lệnh
7. Các mode định địa chỉ (addressing modes)
8. Lập trình IO
9. Tạo trễ
10. Lập trình Timer/Counter
11. Lập trình giao tiếp nối tiếp
12. Lập trình ngắt
13. Lập trình hợp ngữ

- Dữ liệu chứa ở đâu khi thực thi 1 lệnh? → Nhiều lời giải tương ứng các kiểu định địa chỉ khác nhau

- Các kiểu định địa chỉ cho phép xác định nguồn và đích của dữ liệu theo nhiều cách khác nhau tùy tình huống lập trình.
  1. Tức thời
  2. Thanh ghi
  3. Trực tiếp
  4. Gián tiếp
  5. Chỉ số

7-1. Định địa chỉ tức thời

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

VD :

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

7-2. Định địa chỉ thanh ghi

```
MOV     Rn, A       ;n=0,..,7
ADD     A, Rn
MOV     DPL, R6
```

```
MOV DPTR, A
MOV Rm, Rn
```

7-3. Định địa chỉ trực tiếp

•Dùng truy xuất các biến nhớ hoặc các thanh ghi trên chip
•Mặc dầu có thể truy cập cả 128 bytes RAM nội sử dụng kiểu định địa chỉ trực tiếp, song thông thường ta chỉ dùng cho vùng RAM nội đa mục đích (có địa chỉ từ 30 – 7FH)

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

Thanh ghi chức năng đặc biệt & địa chỉ:

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

7-4. Định địa chỉ gián tiếp

- Làm thế nào nhận biết 1 biến khi địa chỉ biến đã được xác định, tính toán hoặc sửa đổi trong khi 1 chương trình đang chạy?
- Khi quản lý các vị trí nhớ liên tiếp. Các điểm nhập được định chỉ số trong các bảng chứa trong RAM (các dãy số hay các chuỗi ký tự)

→ Giải pháp là kiểu định địa chỉ gián tiếp

- Khi này, thanh ghi được sử dụng như 1 con trỏ (pointer) đến dữ liệu
- MOV A,@Ri ; copy dữ liệu trỏ bởi Ri vào A (i=0 hay 1)
- MOV @R1,B
- Nói cách khác, nội dung của các thanh ghi R0 hay R1 có thể là nguồn hoặc đích trong các lệnh MOV, ADD & SUBB

Bài toán 7.1. Copy bytes trong RAM nội

☛ ***Viết chương trình copy 10 bytes từ vùng RAM có địa chỉ bắt đầu là 37H tới vùng RAM có địa chỉ bắt đầu là 59H***

Giải đáp:

```
        MOV   R0,#37h        ; Con trỏ nguồn
        MOV   R1,#59h        ; Con trỏ đích
        MOV   R2,#10         ; Bộ đếm
L1:     MOV   A,@R0
        MOV   @R1,A
        INC   R0
        INC   R1
        DJNZ  R2,L1
```

Bài toán 7.2. Xóa RAM nội

☛ ***Viết chương trình xóa RAM nội từ 60H → 7FH***

Trả lời:

```
                MOV   R0,#60H
LOOP:           MOV   @R0,#0
                INC   R0
                CJNE  R0,#80H,LOOP
```

7-5. Định địa chỉ chỉ số & truy cập ROM nội

- Được sử dụng khi truy cập các thành phần dữ liệu của ***bảng nhảy*** hoặc ***bảng tìm kiếm***

  MOVC A, @A+DPTR

  A = nội dung tại địa chỉ A+DPTR trong ROM

Chú ý:

- Các thành phần dữ liệu được lưu trong không gian bộ nhớ chương trình ROM của 8051, nên sử dụng MOVC thay vì MOV. "C": code.

Thanh ghi nền		*Offset*		*Địa chỉ tác động*
PC (or DPTR)	+	A	=	PC (or DPTR)

Bài toán 7.3. Copy bytes ROM → RAM

- *VD: Giả sử không gian bộ nhớ ROM bắt đầu tại địa chỉ 250H chứa "ROBOCON", viết chương trình truyền các bytes trên vào vùng nhớ RAM bắt đầu tại địa chỉ 40H*

Giải đáp:

```
        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      “ROBOCON”,0

        END
```

Bài toán 7.4. Đọc x & xuất x^2

- ***Viết chương trình đọc giá trị x từ port 1 và xuất giá trị x^2 ra port 2 ?***

Giải đáp:

```
        ORG     0
        MOV     DPTR, #TAB1
        MOV     A, #0FFH      ; chú ý!
        MOV     P1, A
L01:
        MOV     A, P1
        MOVC    A, @A+DPTR
        MOV     P2, A
        SJMP    L01
;-----------------------------------------------
        ORG     300H
TAB1:   DB      0,1,4,9,16,25,36,49,64,81

        END
```

- MOVC A,@A+PC ;hoạt động tương tự, ngoại trừ ở đây, bộ đếm chương trình PC được dùng để chứa địa chỉ nền và bảng được truy xuất nhờ vào chương trình con. Số của *điểm nhập (entry-number)* yêu cầu cho vào thanh chứa A, sau đó chương trình con được gọi. Bảng phải được định nghĩa ngay sau lệnh RET trong chương trình.

```
                    MOV   A, entry-number
                    CALL  TIM-KIEM
                    …
                    …
TIM-KIEM:           INC   A
                    MOVC        A, @A+PC
                    RET
TABLE:    DB        data1,data2,data3,…
```

Homework

Viết chương trình cho 2 ví dụ trên dùng
MOVC A, @A+PC
thay vì
MOVC A, @A+DPTR?

1. Giới thiệu
2. Sơ đồ khối và chân
3. Tổ chức bộ nhớ
4. Các thanh ghi chức năng đặc biệt
5. Dao động và hoạt động reset
6. Tập lệnh
7. Các mode định địa chỉ
8. Lập trình IO (IO Port Programming)
9. Tạo trễ
10. Lập trình Timer/Counter
11. Lập trình giao tiếp nối tiếp
12. Lập trình ngắt
13. Lập trình hợp ngữ

8-1. Cổng 1 (Port 1)

Port 1 (chân 1- 8)

- Port 1 được ký hiệu P1

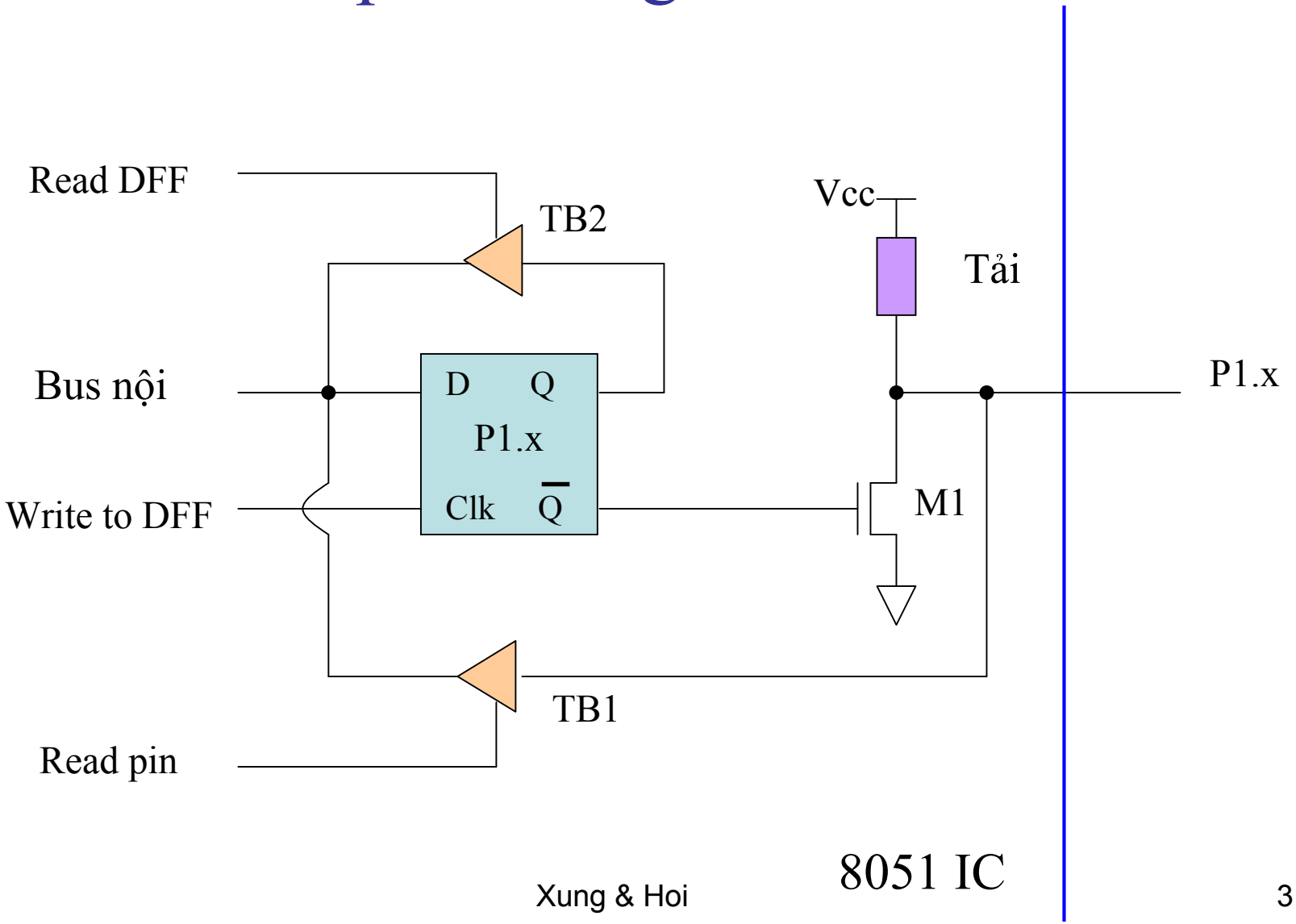
  → Các chân: P1.0 - P1.7

- Sử dụng P1 trong các ví dụ sau đây để chỉ ra hoạt động của chúng

  → P1 là cổng ra – output (ghi dữ liệu CPU ra các chân bên ngoài)

  → P1 là cổng vào – input (đọc dữ liệu từ các chân bên ngoài vào CPU bus)

8-2. Cấu trúc phần cứng của P1.x



a. Bus nội: giao tiếp với CPU

b. Bộ chốt dữ liệu DFF: lưu trữ giá trị của chân. Khi "Write to DFF" = 1: ghi dữ liệu vào DFF

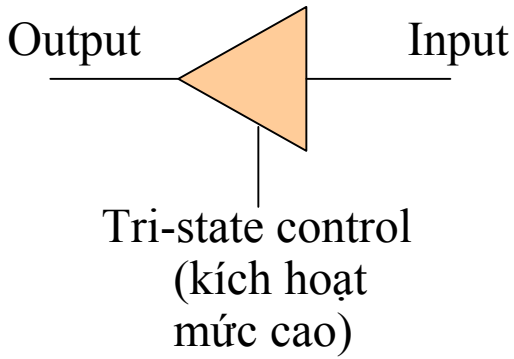
c. Hai bộ đệm 3 trạng thái (tri-state buffers):

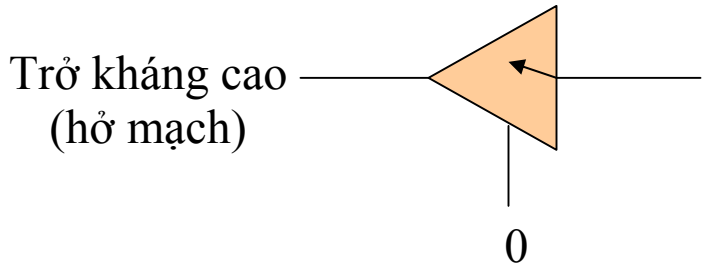
- TB1: điều khiển bởi "Read pin". Khi "Read pin" = 1: đọc giá trị tại chân ngoài

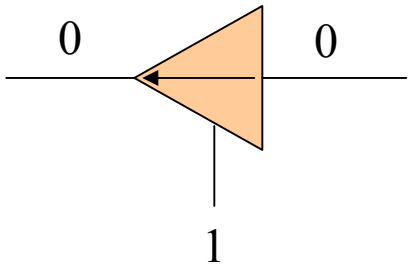
- TB2: điều khiển bởi "Read DFF". Khi "Read DFF" = 1: đọc giá trị từ DFF nội

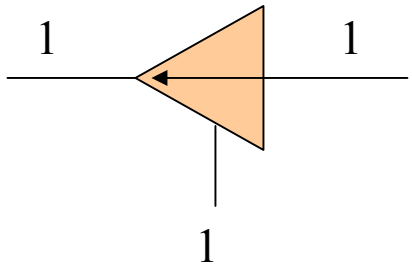
d. Transistor M1

Bộ đệm 3 trạng thái (Tri-state Buffer)







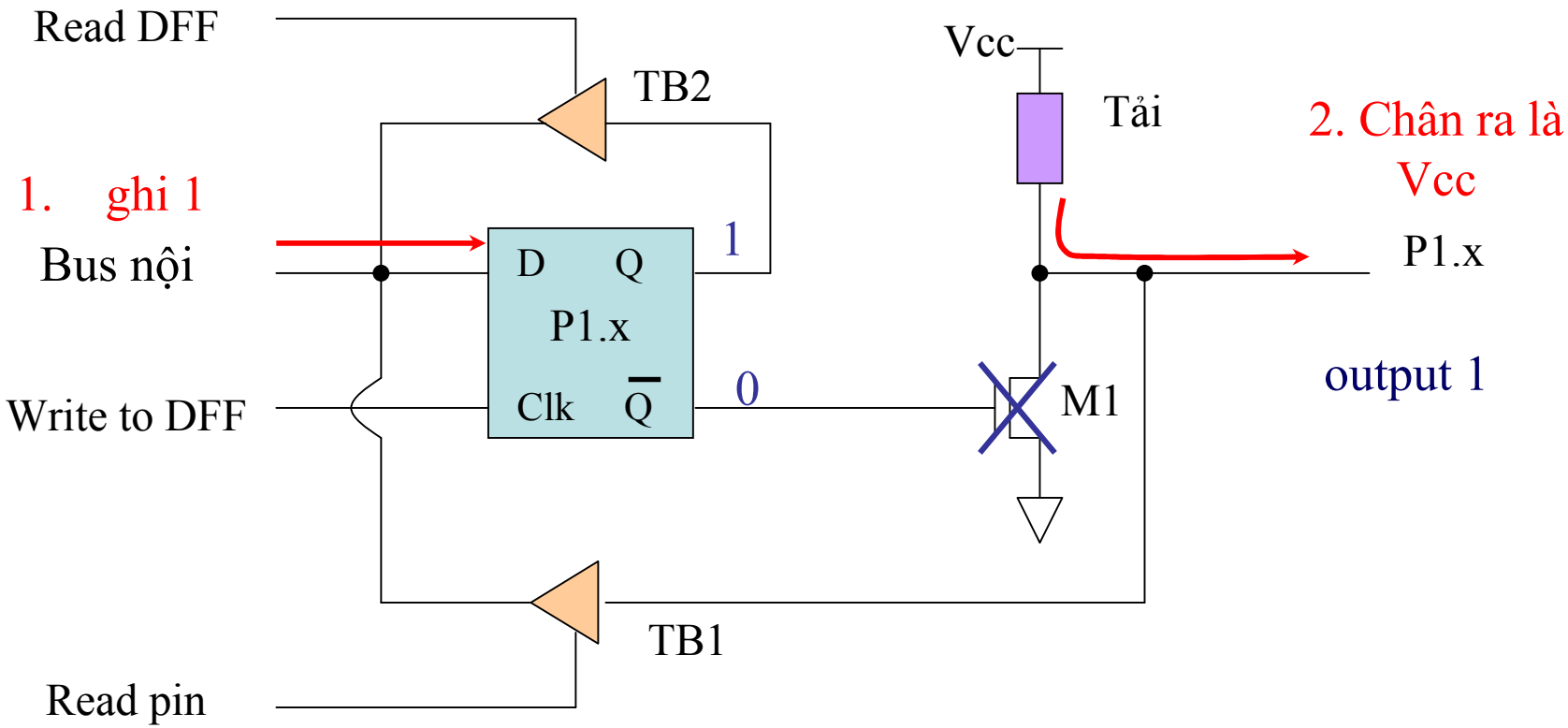


8-3. Ghi ra cổng output

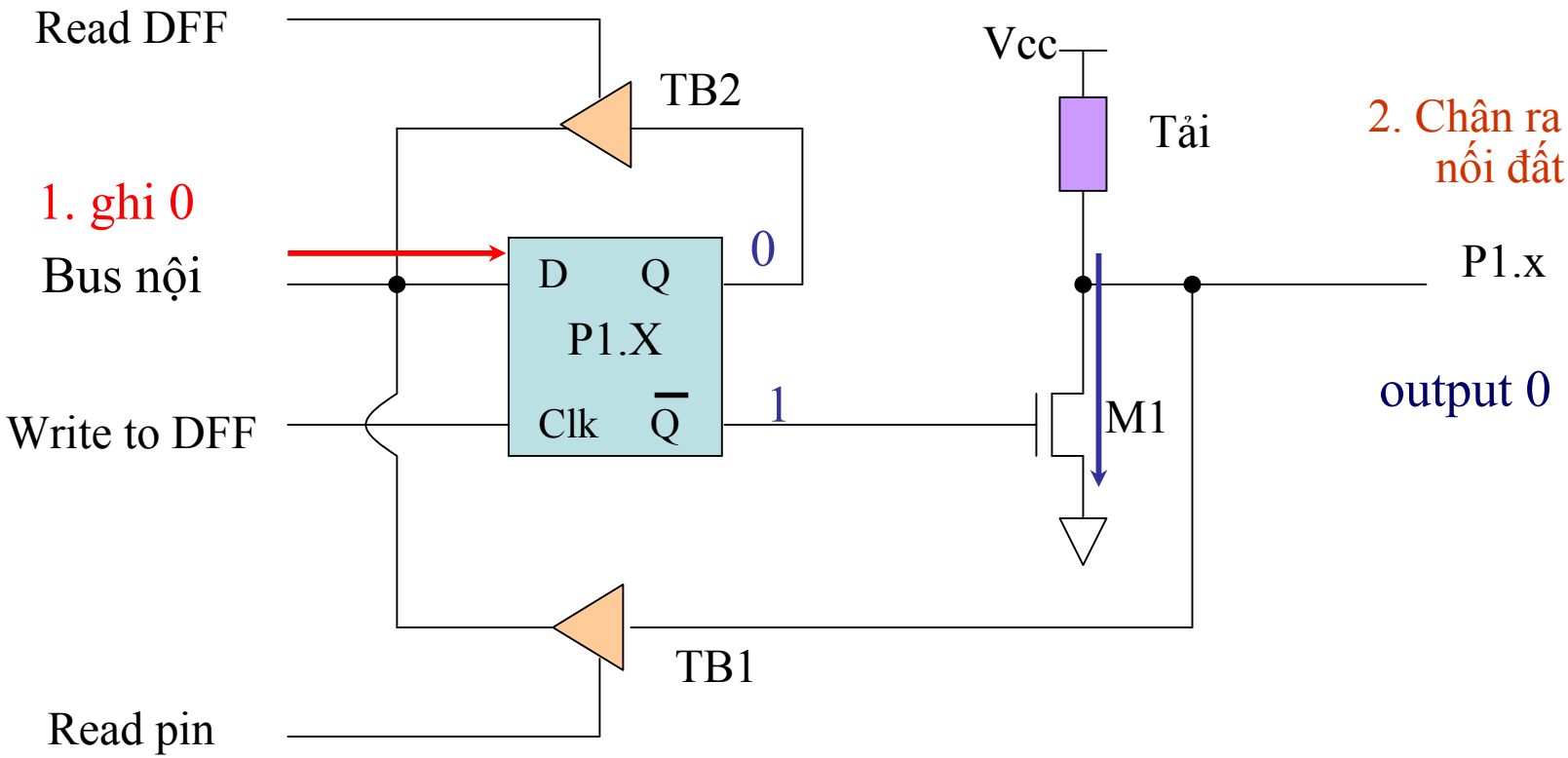
Vd:

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

8-3-1. Ghi “1” ra chân output P1.x



8-3-2. Ghi “0” ra chân output P1.x



8-4. Đọc từ chân input & bộ chốt

Khi đọc chân, có hai khả năng sau:

- Đọc trạng thái của chân input (bên ngoài)

```
MOV   A,Px
JNB   P2.1,Label
JB    P2.1,Label
```

- Đọc dữ liệu bộ chốt của chân output (bên trong)

```
ANL   P1,A
ORL   P1,A
INC   P1
```

Đọc-Sửa đổi-Ghi

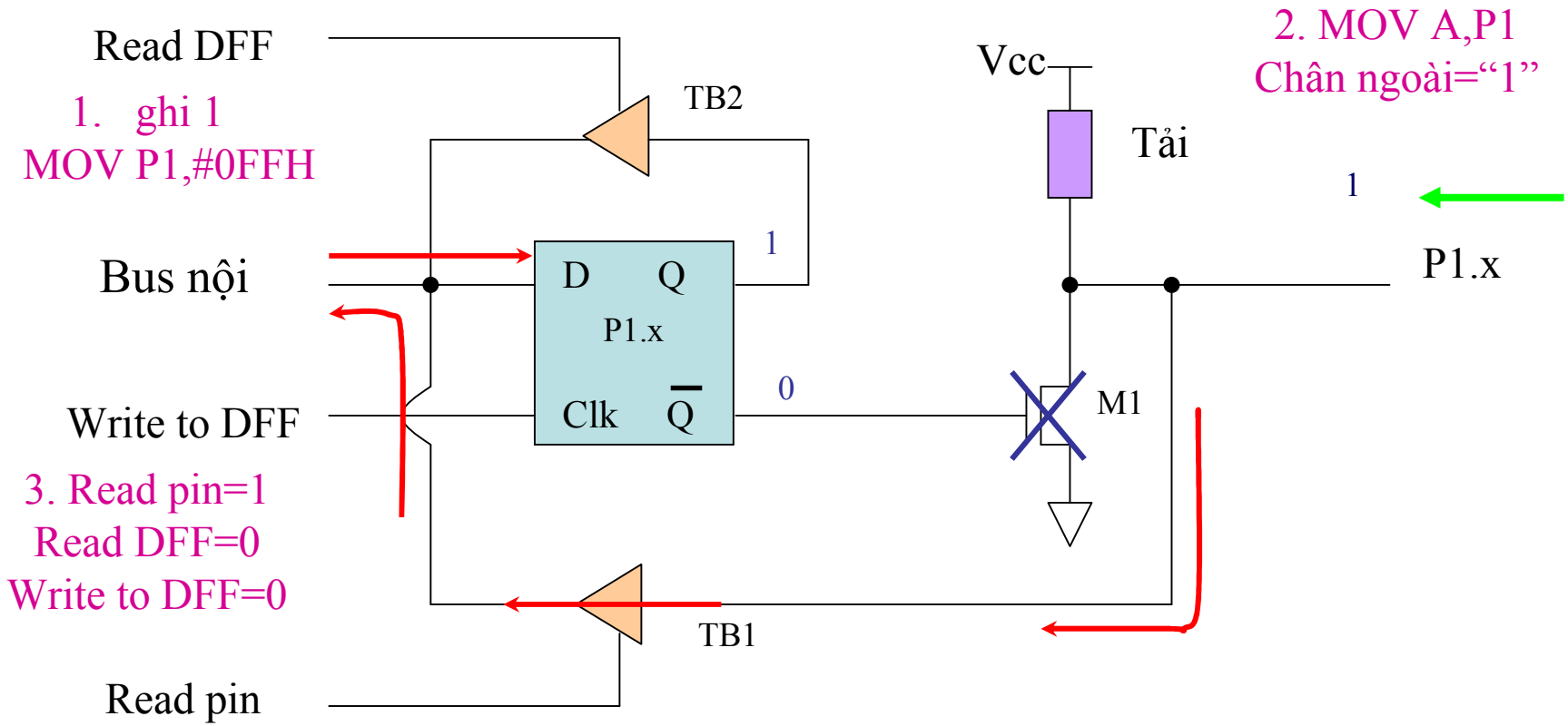
8-4-1. Đọc từ chân input

Để P1 là input, P1 phải được lập trình bằng cách ghi "1" vào tất cả các bit của P1

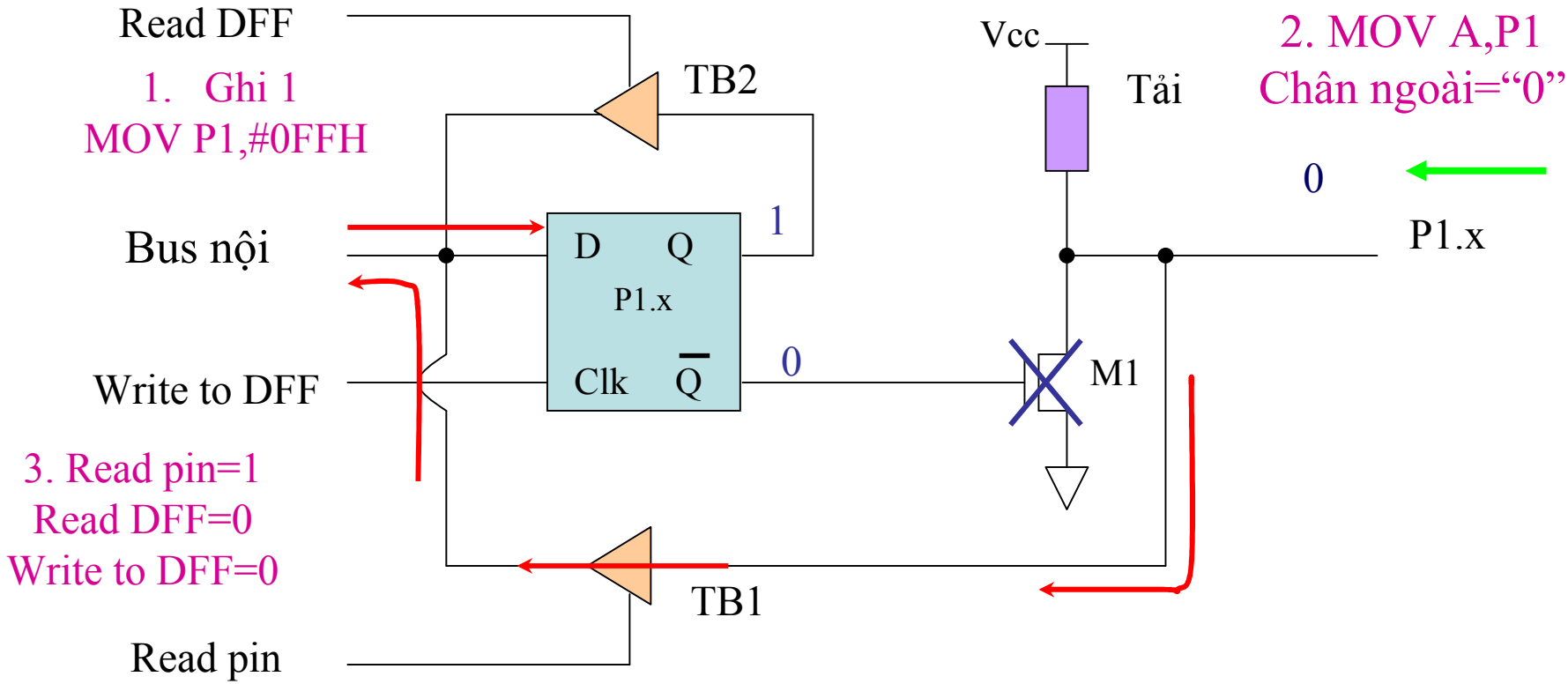
```
        MOV  P1,#0FFH     ; P1=11111111B
                          ; P1 là input
BACK:   MOV  A,P1
        MOV  P2,A
        SJMP BACK
```

☛ tương tự cho P0, P2, P3

Đọc “1” tại chân input



Đọc “0” tại chân input



Các lệnh đọc chân input

Lệnh	*Ví dụ*	*Mô tả*
MOV A,PX	**MOV A,P2**	Đọc P2 vào A
JNB PX.Y,..	**JNB P2.1,TARGET**	Nhảy nếu P2.1 = 0
JB PX.Y,..	**JB P1.3,TARGET**	Nhảy nếu P1.3 = 1
MOV C,PX.Y	**MOV C,P2.4**	Copy trạng thái chân P2.4 vào CY

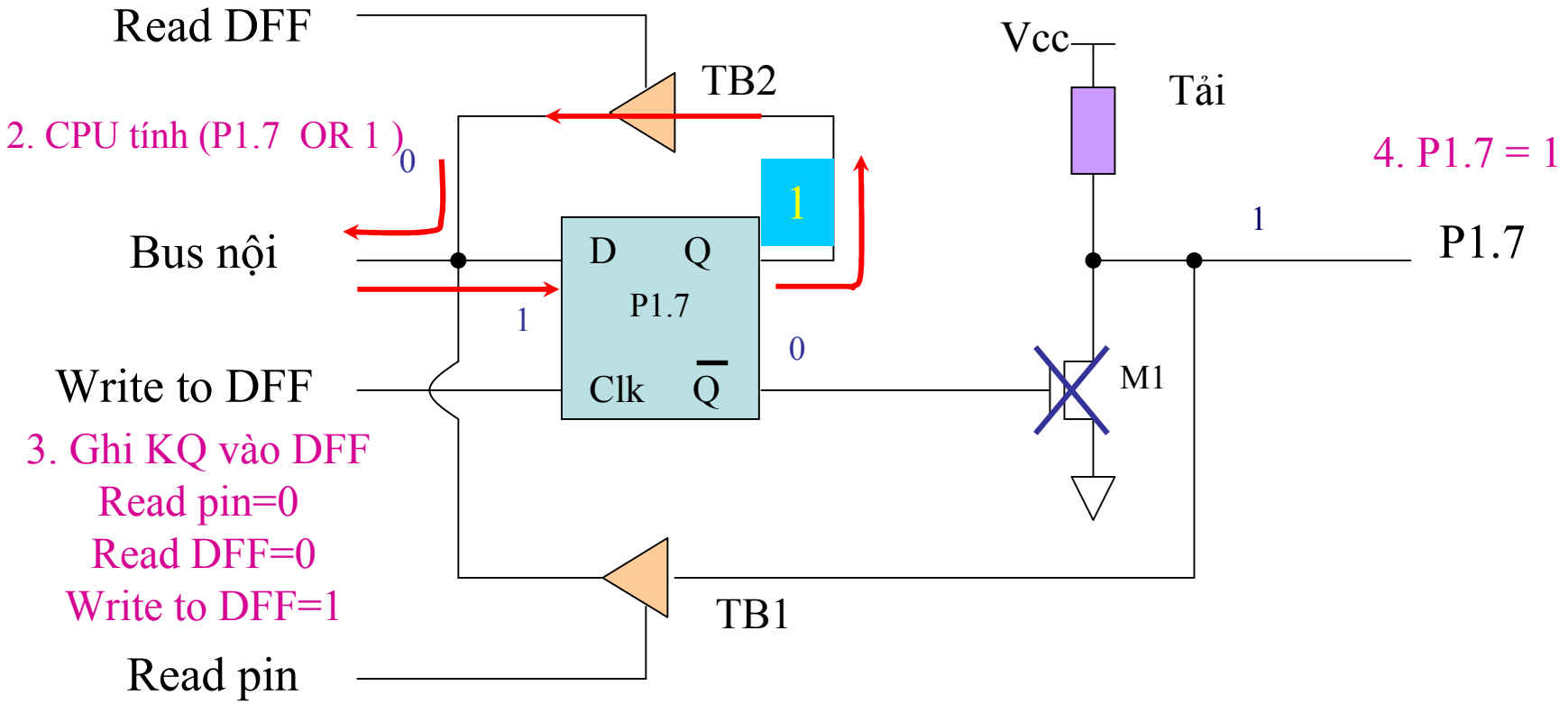
8-4-2. Đọc chân ouput tức đọc bộ chốt

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

- "Read DFF" kích hoạt TB2 và chuyển dữ liệu từ Q của DFF vào CPU → đọc được P1.7 = 0
- CPU thực hiện OR dữ liệu này với bit 1 → được 1
- D của DFF bị thay đổi thành 1
- Ghi KQ ra chân → P1.7 = 1

Đọc bộ chốt

1. Read pin = 0
Read DFF = 1
Write to DFF = 0 (ban đầu P1.7=0)



Đọc-Sửa đổi-Ghi

Đặc điểm này bao gồm 3 hành động trong 1 lệnh đơn:

1. CPU đọc bộ chốt
2. CPU thực hiện tính toán → sửa đổi bộ chốt
3. Ghi ra chân

Chú ý: 8 chân của Port làm việc độc lập nhau

Các lệnh có đặc điểm Đọc-Sửa đổi-Ghi

Lệnh	Ví dụ
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

Tóm lại

- 1 chân là output thì có thể ghi dữ liệu trực tiếp ra chân
- 1 chân là output thì đọc trạng thái của chân nghĩa là đọc bộ chốt tương ứng của chân

- 1 chân là input thì set nó lên "1" trước khi thao tác
- 1 chân là input thì đọc trạng thái trực tiếp từ chân

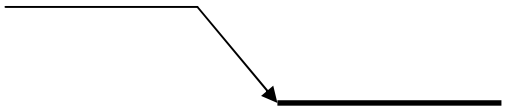
Câu hỏi gợi nhớ

- Cách ghi dữ liệu ra chân?
- Cách đọc dữ liệu từ chân?
  - ✓Đọc giá trị từ chân bên ngoài
    - •Tại sao phải set chân trước khi tiến hành đọc?
  - ✓Đọc giá trị từ bộ chốt
    - •Lệnh như thế nào gọi là có tính chất Đọc-Sửa đổi-Ghi?

Bài toán 1: thao tác bit

Bài toán:

1. Theo dõi bit P1.2 cho đến khi nhận được “1”
2. Khi nhận được “1”, ghi 45H ra P0
3. & gửi xung “High-to-Low” ra chân P2.3



Giải đáp:

```
        SETB  P1.2          ;P1.2 là input
        MOV   A,#45H        ;A=45H
AGAIN:  JNB   P1.2,AGAIN    ;lặp lại đến khi P1.2=1
        MOV   P0,A          ;xuất A ra P0
        SETB  P2.3          ;P2.3 = "High"
        CLR   P2.3          ;P2.3 = "Low" for H-to-L
```

Trong đó :

1. JNB: jump if no bit (jump if P1.2 = 0)
2. Xung "H-to-L" được tạo bởi chuỗi lệnh SETB & CLR

Bài toán 2: Ưu điểm khi sử dụng câu lệnh có đặc điểm **Đọc-Sửa đổi-Ghi**

☛**Có ba cách để nháy các bit của Port 1 liên tục**

- **Cách 1: Gửi dữ liệu tới P1 qua thanh chứa A**

```
BACK:   MOV     A,#55H      ;A=01010101B
        MOV     P1,A
        ACALL   DELAY
        MOV     A,#0AAH     ;A=10101010B
        MOV     P1,A
        ACALL   DELAY
        SJMP    BACK
```

- **Cách 2:** **Xuất dữ liệu trực tiếp**

```
BACK:   MOV     P1,#55H     ;P1=01010101B
        ACALL   DELAY
        MOV     P1,#0AAH    ;P1=10101010B
        ACALL   DELAY
        SJMP    BACK
```

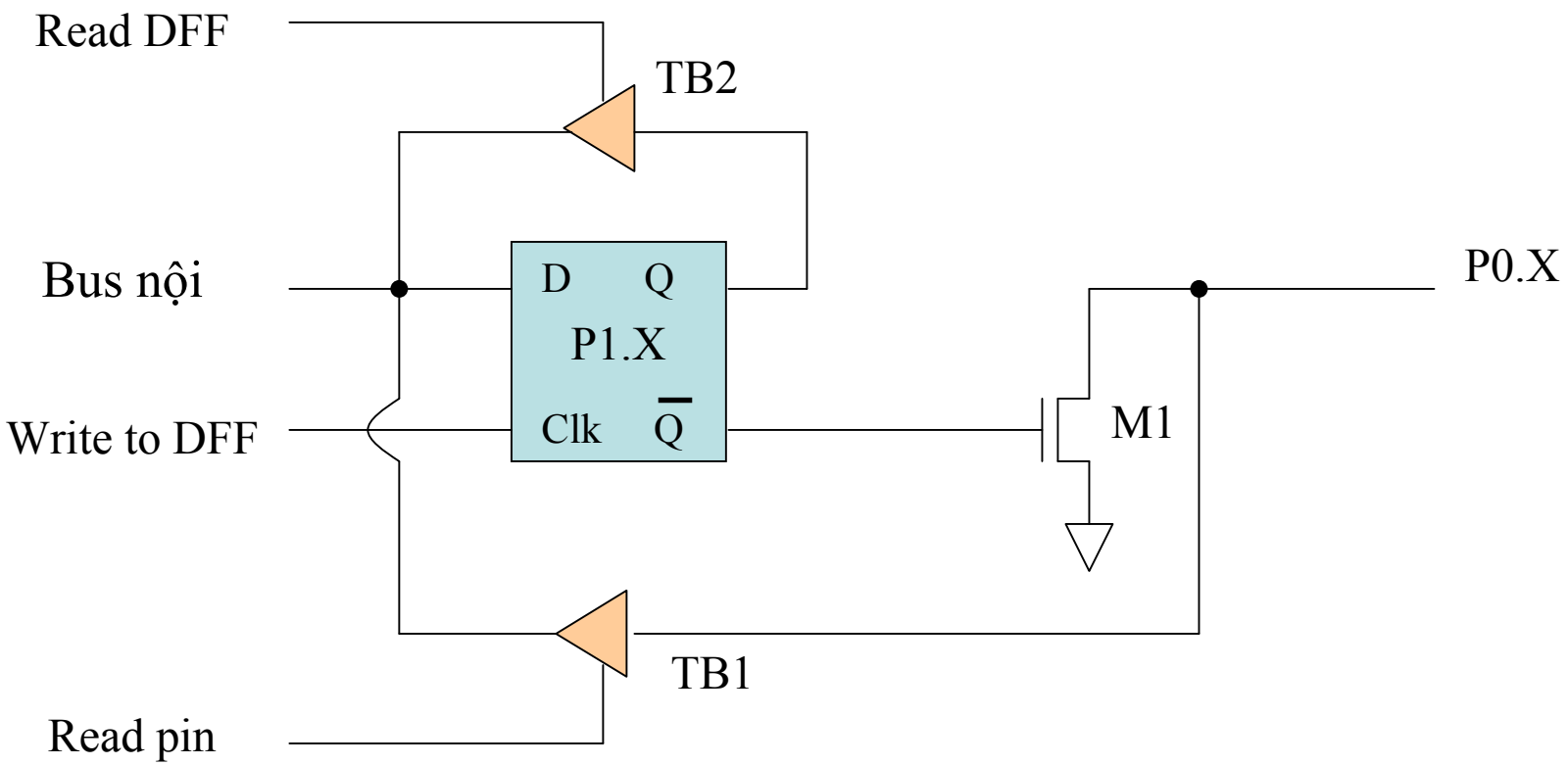
- **Cách 3:** **Dùng lệnh với đặc điểm Đọc-Sửa đổi-Ghi**

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

8-5. Các cổng (port) còn lại

- P1, P2, P3 có điện trở tải nội
- P0 không có điện trở tải nội
- Phía người lập trình: không có sự khác biệt nào giữa các cổng
- Tất cả các cổng là output và ở mức logic cao khi Reset

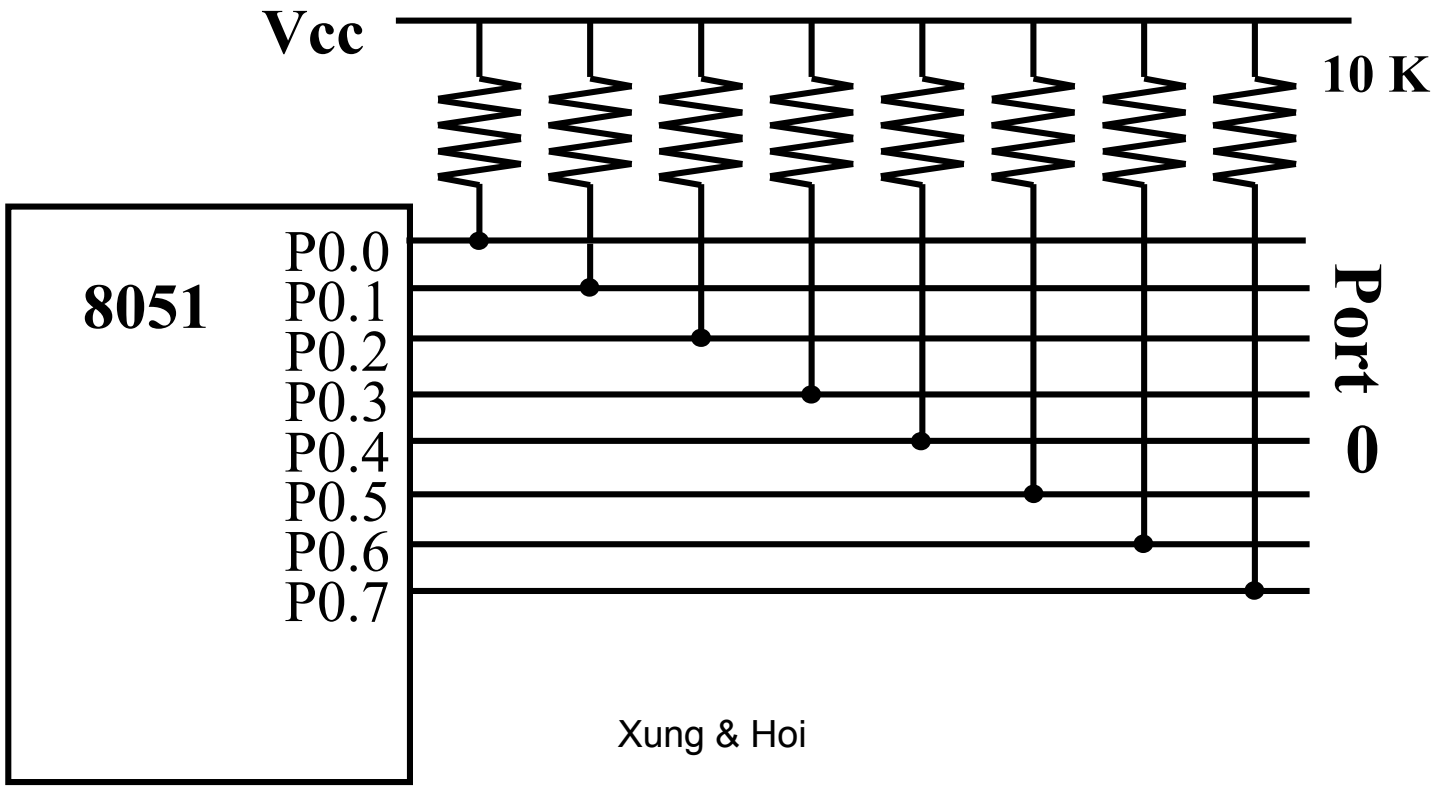
Cấu trúc phần cứng P0.x



Port 0

Khi P0 được dùng để xuất hay nhập dữ liệu, phải có các điện trở kéo lên bên ngoài (10K hoặc 4,7K)

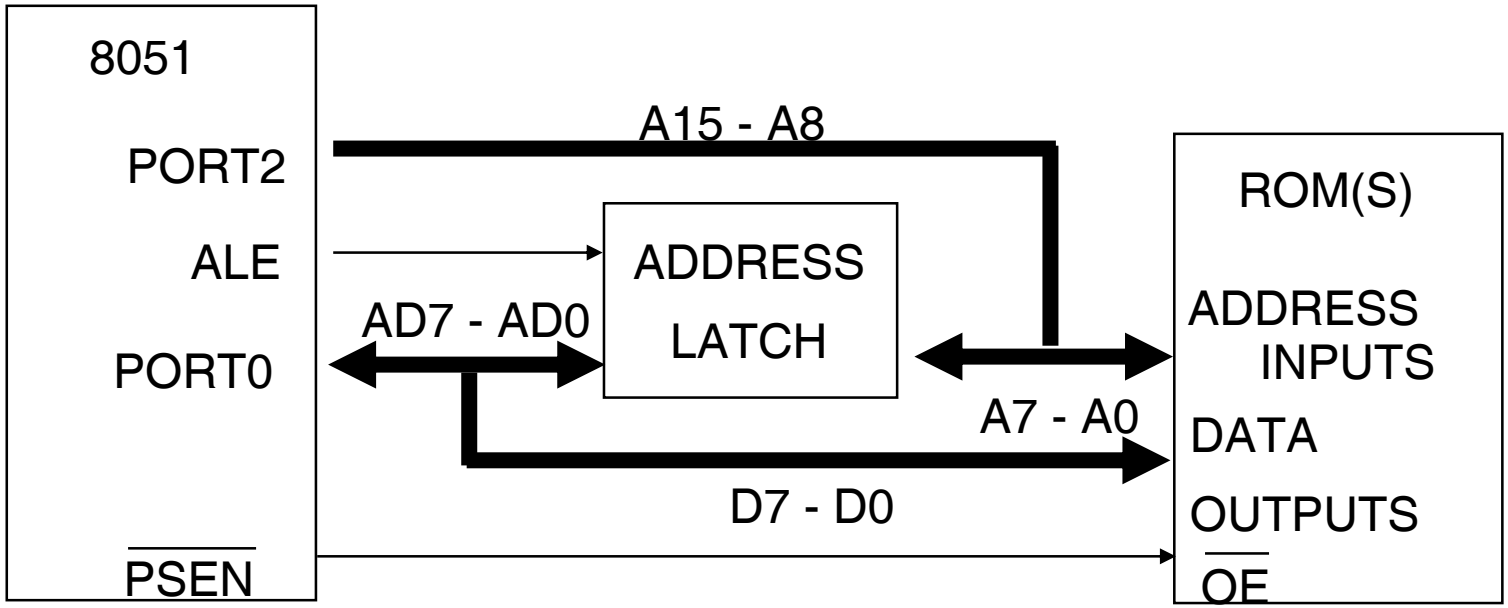
Và với các điện trở kéo lên bên ngoài này, khi reset hay khi bật nguồn, P0 mặc định là output.



8-6. Vai trò kép của P0

- Khi nối 8051 tới bộ nhớ ngoài thì nó sử dụng các port để gửi địa chỉ và đọc các lệnh
  - 8051 có khả năng truy xuất 64K bytes bộ nhớ ngoài
  - Địa chỉ 16-bit: P0 cung cấp các đường địa chỉ A0-A7, P2 cung cấp A8-A15
  - Đồng thời, P0 cung cấp các đường dữ liệu D0-D7
- Khi P0 được sử dụng đa hợp địa chỉ/dữ liệu, nó được kết nối tới 74LS373 để chốt địa chỉ
  - Khi này không cần các điện trở kéo lên bên ngoài

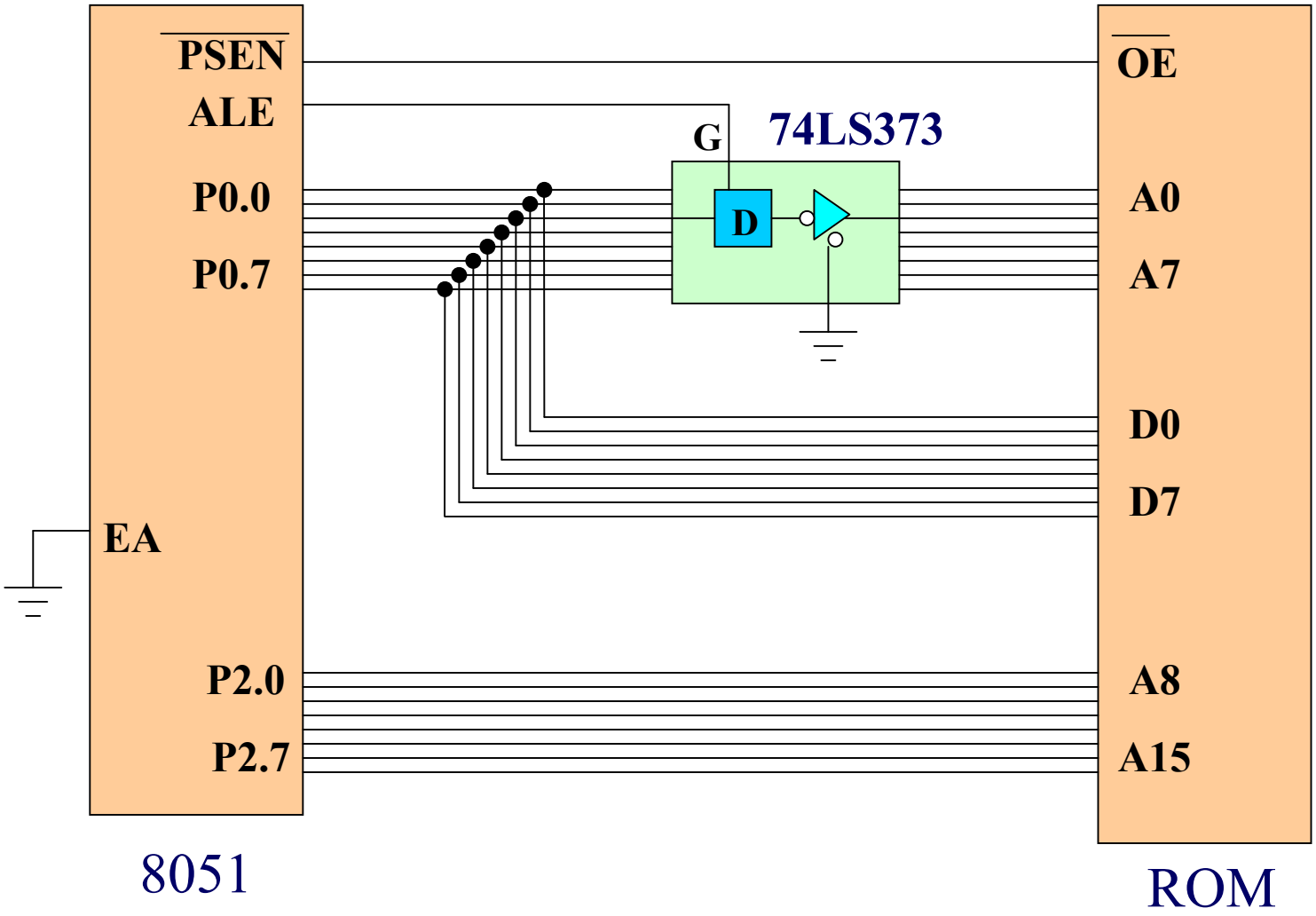
8-6-1. Bộ chốt 74LS373

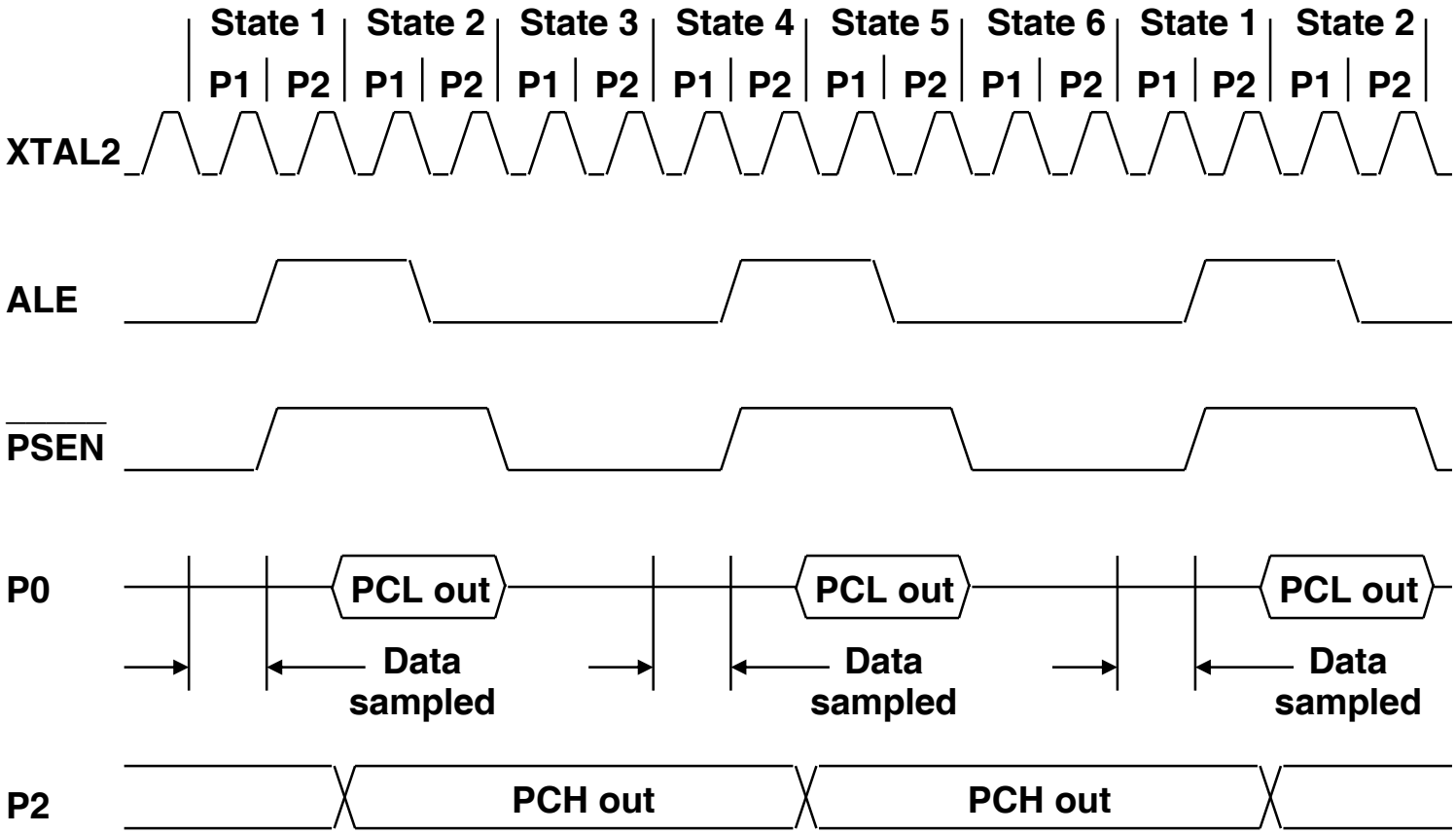


Chân ALE được dùng để giải đa hợp (de-multiplexing) địa chỉ và dữ liệu bằng cách nối tới chân G của bộ chốt 74LS373

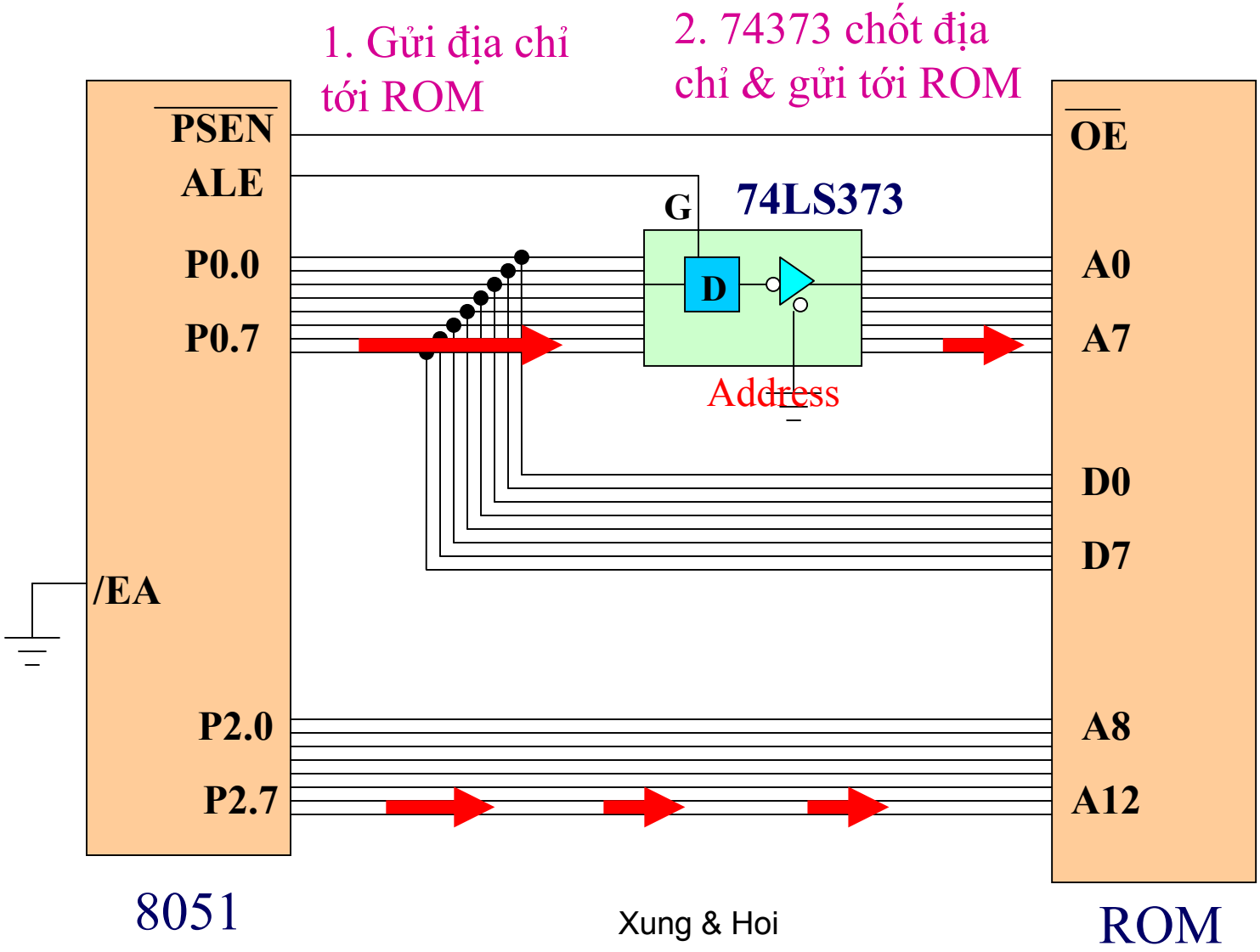
Khi ALE=0, P0 cung cấp dữ liệu D0-D7

Khi ALE=1, P0 cung cấp địa chỉ A0-A7

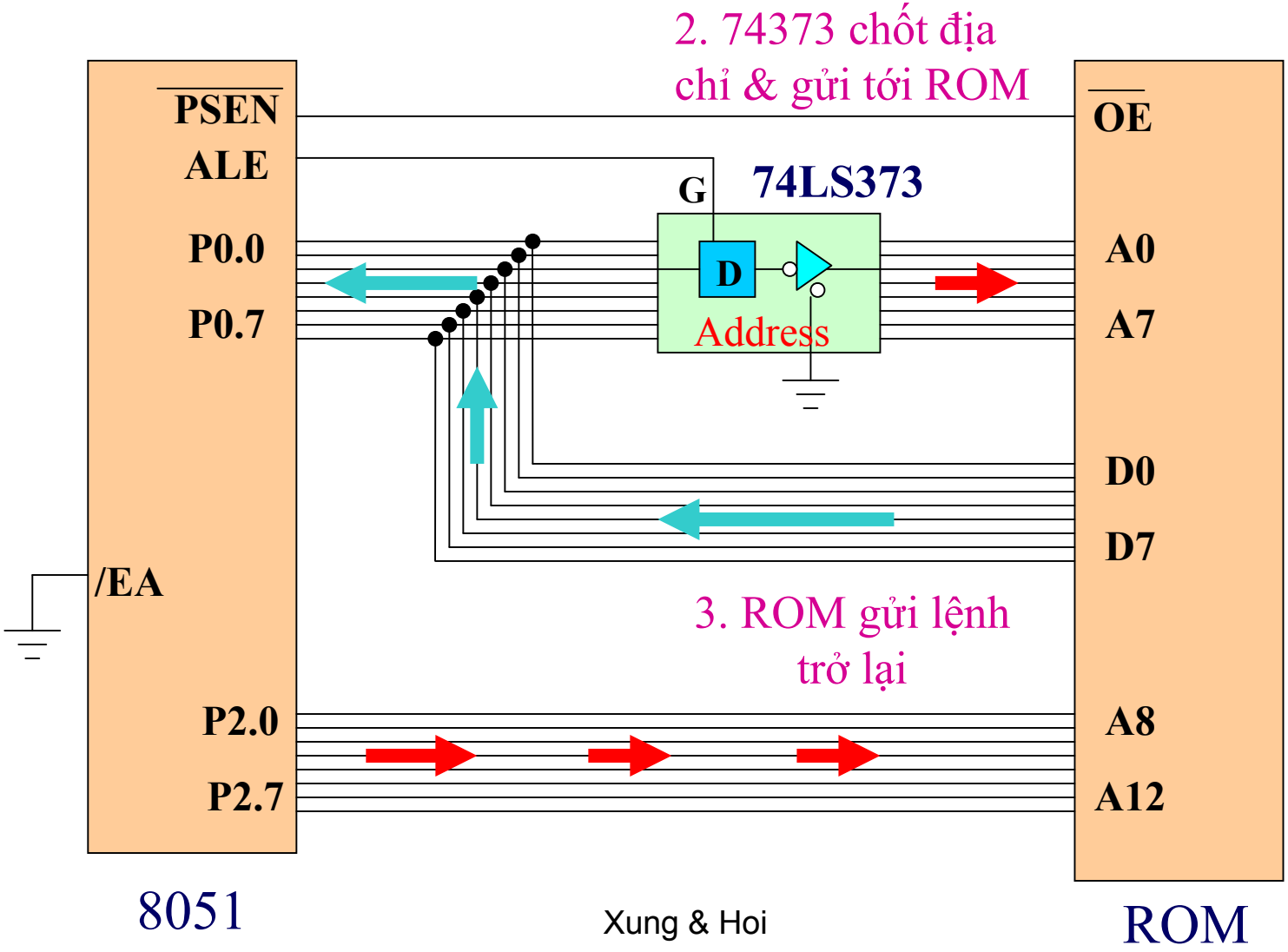

PSEN
ALE
P0.0
P0.7
EA
P2.0
P2.7
8051
G
74LS373
D
OE
A0
A7
D0
D7
A8
A15
ROM


State 1
State 2
State 3
State 4
State 5
State 6
State 1
State 2
P1
P2
XTAL2
ALE
PSEN
P0
PCL out
PCL out
PCL out
Data sampled
Data sampled
Data sampled
P2
PCH out
PCH out

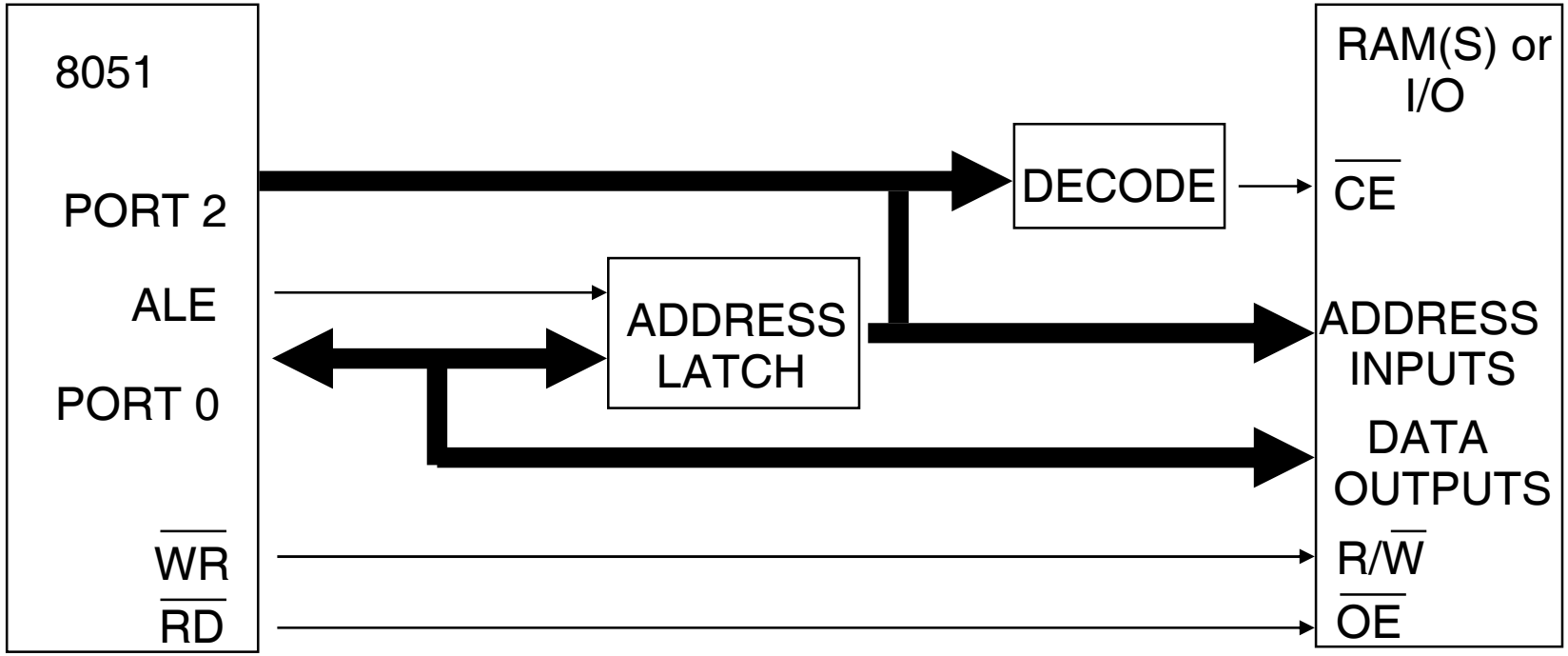
8-6-2. Đọc ROM ngoài (1/2)



Đọc ROM ngoài (2/2)



8-6-3. Đọc ghi RAM ngoài

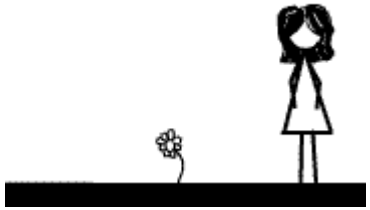


Không gian tối đa 64KB. Truy xuất địa chỉ gián tiếp qua 2 thang ghi R0 và R1, con trỏ dữ liệu.

8-7. Port 3

- Port 3 không cần các điện trở kéo lên bên ngoài
- Mặc dầu Port 3 cũng được cấu hình như ouput khi reset nhưng nó thường dùng cho các chức năng riêng nêu dưới đây

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



1. Giới thiệu
2. Sơ đồ khối và chân
3. Tổ chức bộ nhớ
4. Các thanh ghi chức năng đặc biệt (SFR)
5. Dao động và hoạt động reset
6. Tập lệnh
7. Các mode định địa chỉ (addressing modes)
8. Lập trình IO
9. Tạo trễ
10. Lập trình Timer/Counter
11. Lập trình giao tiếp nối tiếp
12. Lập trình ngắt
13. Lập trình hợp ngữ

Chu kỳ máy - Machine Cycle (MC)

- Chy kỳ máy (machine cycle) là đơn vị đo thời gian thực thi các lệnh
- Tần số dao động của thạch anh cho họ 8051 có thể từ 4MHz đến 30 MHz, tùy thuộc nhà sản xuất. Song thông thường, loại 11.0592 MHz được dùng nhằm làm cho các hệ thống dựa trên 8051 tương thích với cổng nối tiếp của IBM PC.
- Với họ 8051, một chu kỳ máy kéo dài 12 chu kỳ dao động

Bài toán 9.1. Tính thời gian thực thi 1 lệnh

- ***Cho 1 hệ 8051 có dao động thạch anh 11.0592 MHz. Tìm thời gian thực thi cho các câu lệnh sau:***

a. MOV R3,#79H
b. LJMP
c. MUL AB
d. DEC R5
e. SJMP
f. NOP

Giải đáp:

a	1	1*1.085us
b	2	2*1.085us
c	4	4*1.085us
d	1	1*1.085us
e	2	2*1.085us
f	1	1*1.085us

Bài toán 9.2. Tính thời gian trễ đoạn code

- *Tìm thời gian trễ cho chương trình con say đây, (với tần số dao động thạch anh là 11.0592 MHz)*

```
DELAY:   MOV   R3,#250          ;      1 MC
HERE:    NOP                    ;      1 MC
         NOP                    ;      1 MC
         NOP                    ;      1 MC
         NOP                    ;      1 MC
         DJNZ  R3,HERE          ;      2 MC
         RET                    ;      1 MC
```

Answer: [250x(1+1+1+1+2)+2]x1.085 us=1629.67 us

Bài toán 9.3. Nháy LED

- *Chương trình lần lượt thay đổi trạng thái sáng/tối của LED hay lần lượt thay đổi trạng thái đóng/mở của rơle:* *(với KIT 8951)*

```
        ORG     0000
        MOV     A,#01H
L1:     RL      A
        MOV     P2,A
        LCALL   DELAY_500ms
        LCALL   DELAY_500ms
        SJMP    L1
DELAY_500ms: …
```

```
DELAY_500ms:
        PUSH 00H
        PUSH 01H

        MOV   R1, #250
D1:     MOV   R0, #250
        DJNZ  R0, $
        MOV   R0, #250
        DJNZ  R0, $
        DJNZ  R1, D1
        MOV   R1, #250
D2:     MOV   R0, #250
        DJNZ  R0, $
        MOV   R0, #250
        DJNZ  R0, $
        DJNZ  R1, D2

        POP   01H
        POP   00H
        RET

        END
```

1. Giới thiệu
2. Sơ đồ khối và chân
3. Tổ chức bộ nhớ
4. Các thanh ghi chức năng đặc biệt (SFR)
5. Dao động và hoạt động reset
6. Tập lệnh
7. Các mode định địa chỉ
8. Lập trình IO
9. Tạo trễ
10. Lập trình Timer/Counter
11. Lập trình giao tiếp nối tiếp
12. Lập trình ngắt
13. Lập trình hợp ngữ

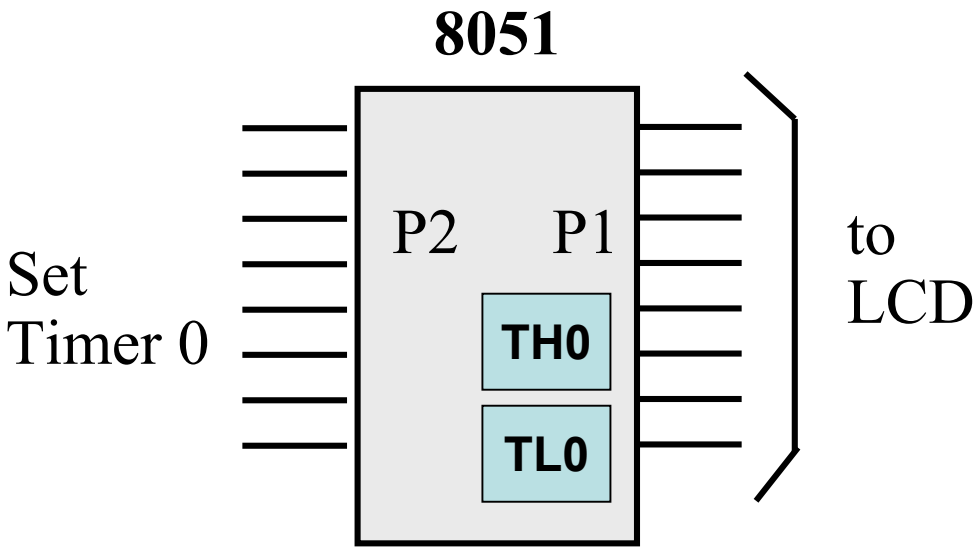
10-1. Giới thiệu

- 8051 có 2 timers/counters: timer/counter 0 & timer/counter 1. Chúng có thể được dùng như:

1. **Bộ định thời (Timer)** dùng như 1 bộ tạo trễ
   - Nguồn xung clock chính là dao động thạch anh bên trong
2. **Bộ đếm sự kiện (Event Counter)**
   - Đầu vào từ chân bên ngoài để đếm số sự kiện
   - Có thể dùng đếm số người đi qua cổng, số vòng quay của bánh xe, hay bất kể các sự kiện mà chuyển được sang dạng xung
3. **Tạo tốc độ baud (baud rate)** cho port nối tiếp của 8051

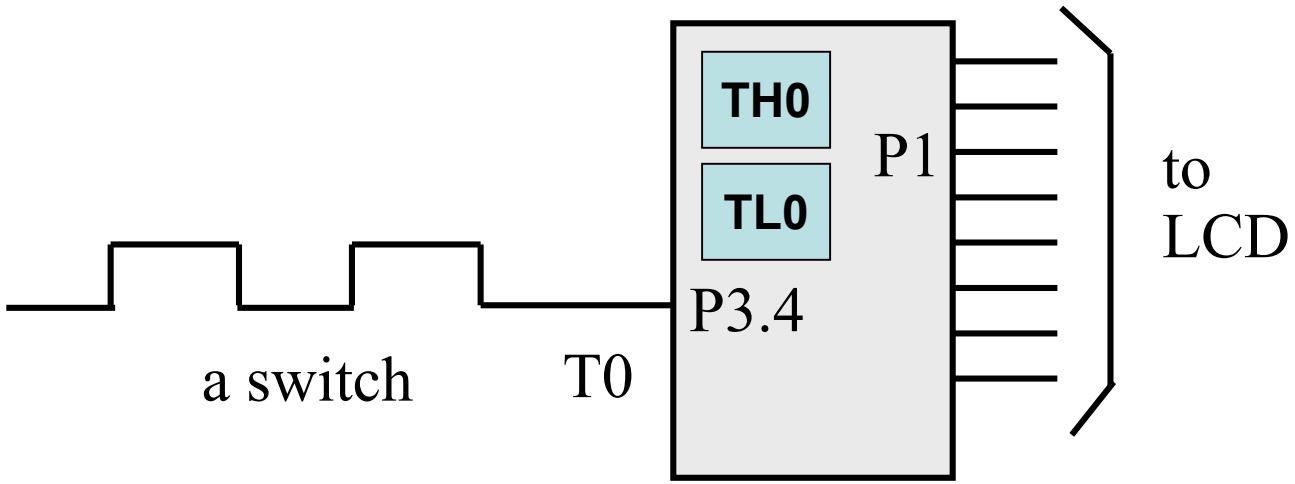
Timer

- Khởi tạo giá trị ban đầu cho các thanh ghi
- Kích hoạt Timer, sau đó 8051 tính lên
- Ngõ vào là từ clock nội (machine cycle)
- Khi các thanh ghi bằng 0 thì 8051 sẽ set cờ tràn



Counter

- Đếm số sự kiện:
  - Chỉ ra số sự kiện trên các thanh ghi
  - Counter 0: Ngõ vào từ chân bên ngoài T0 (P3.4)
  - Counter 1: Ngõ vào từ chân bên ngoài T1 (P3.5)



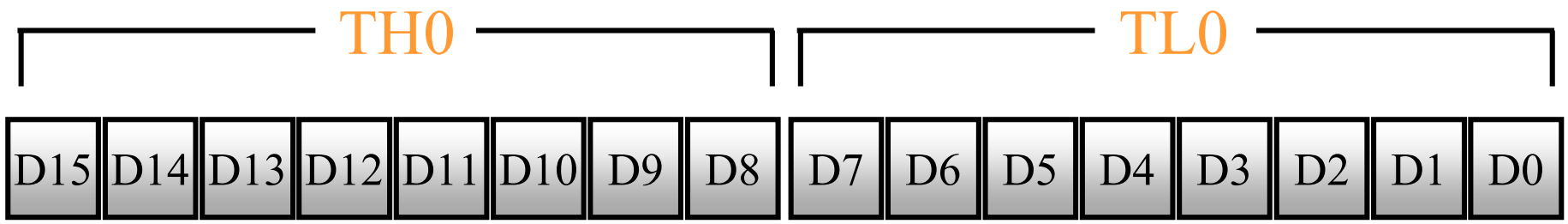
Các thanh ghi dùng truy xuất Timer/Counter

- TH0, TL0, TH1, TL1
- TMOD thanh ghi chế độ định thời (Timer mode register)
- TCON thanh ghi điều khiển định thời (Timer control register)
- 8052 với 3 timers/counters sẽ có thêm các thanh ghi T2CON (Timer 2 control register), TH2 and TL2

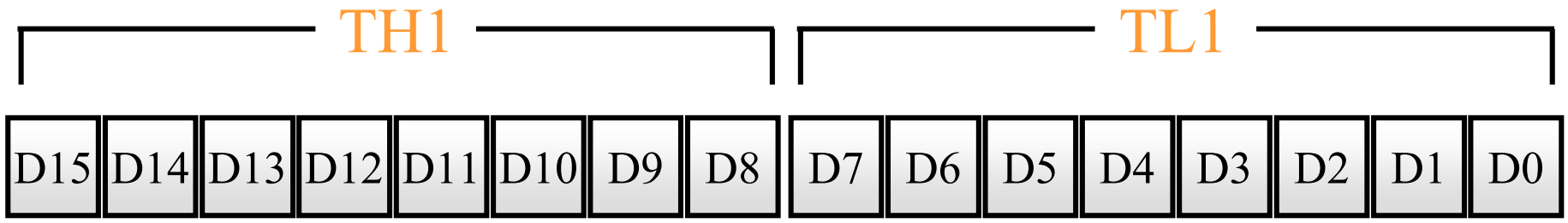
SFR Name	Description	SFR Address
TH0	Timer 0 High Byte	8Ch
TL0	Timer 0 Low Byte	8Ah
TH1	Timer 1 High Byte	8Dh
TL1	Timer 1 Low Byte	8Bh
TCON	Timer Control	88h
TMOD	Timer Mode	89h

TH0, TL0, TH1, TL1 (not bit addressable)

- Cả timer 0 & timer 1 đều có độ rộng 16 bits
  - Các thanh ghi này lưu trữ
    - Giá trị tạo thời gian trễ (time delay) (nếu là timer)
    - Số sự kiện (number of events) (nếu là counter)
  - Timer 0: **TH0** & **TL0**
    - Timer 0 high byte, timer 0 low byte
  - Timer 1: **TH1** & **TL1**
    - Timer 1 high byte, timer 1 low byte
  - Mỗi bộ định thời 16-bit có thể được truy cập như 2 thanh ghi 8-bit tách biệt



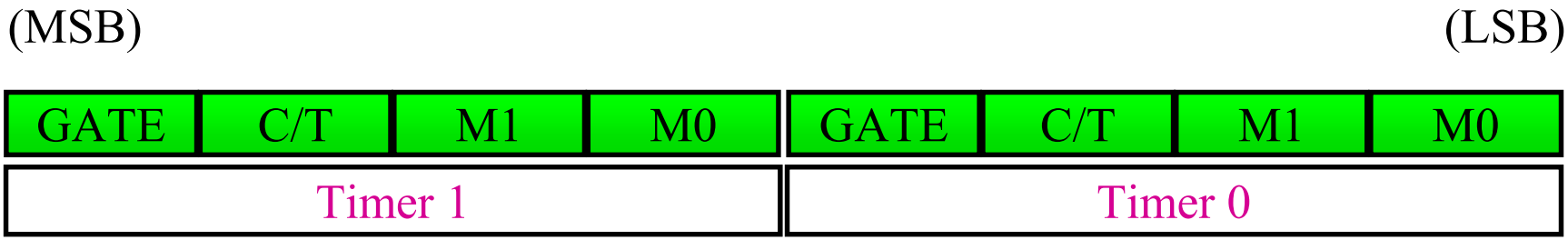
Timer 0



Timer 1

10-2. Thanh ghi chế độ định thời TMOD

- Timer mode register: **TMOD** (not bit addressable)
  - Thanh ghi 8-bit
  - Thiết lập chế độ hoạt động cho các bộ định thời:
    - 4 bits thấp dành cho Timer 0 (Set to 0000 if not used)
    - 4 bits cao dành cho Timer 1 (Set to 0000 if not used)



(MSB) (LSB)

GATE	C/T	M1	M0	GATE	C/T	M1	M0
Timer 1				Timer 0			

GATE

Bit điều khiển cổng. Khi set lên 1, timer chỉ hoạt động khi chân /INTx ở mức cao và TRx = 1. Khi xóa, timer hoạt động khi TRx = 1

C/T

Bit chọn chức năng đếm hoặc định thời: 1: đếm; 0: định thời

M1

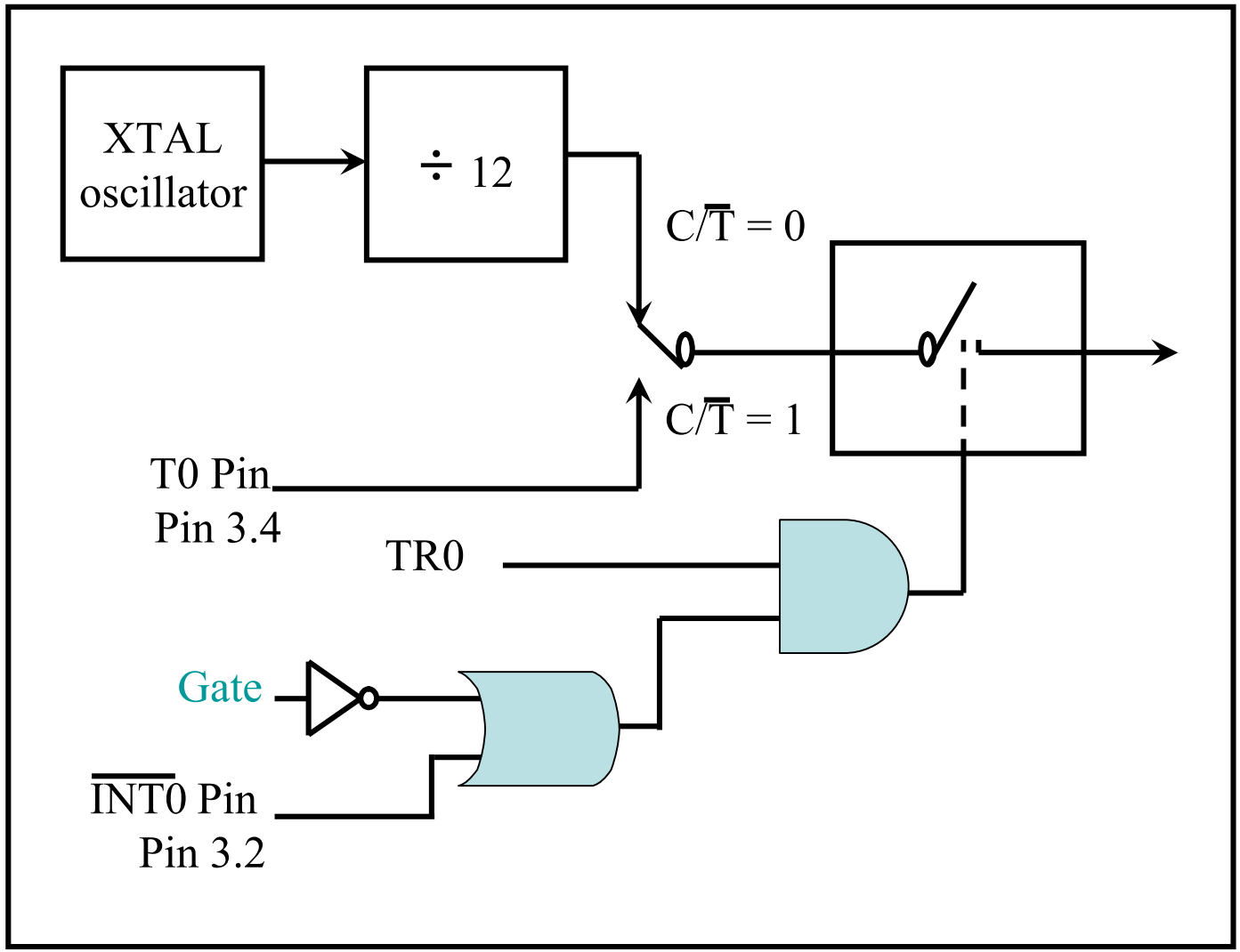
Chọn mode - bit 1

M0

Chọn mode - bit 0

GATE

- GATE=0
  - Điều khiển bên trong (Internal control)
  - Bật hay tắt timer bằng phần mềm
  - Timer được cho phép khi TR được set
- GATE=1
  - Điều khiển bên ngoài (External control)
  - Bật hay tắt timer bằng phần mềm & một nguồn bên ngoài (external source)
  - Timer được cho phép khi chân /INT ở mức cao & TR được set

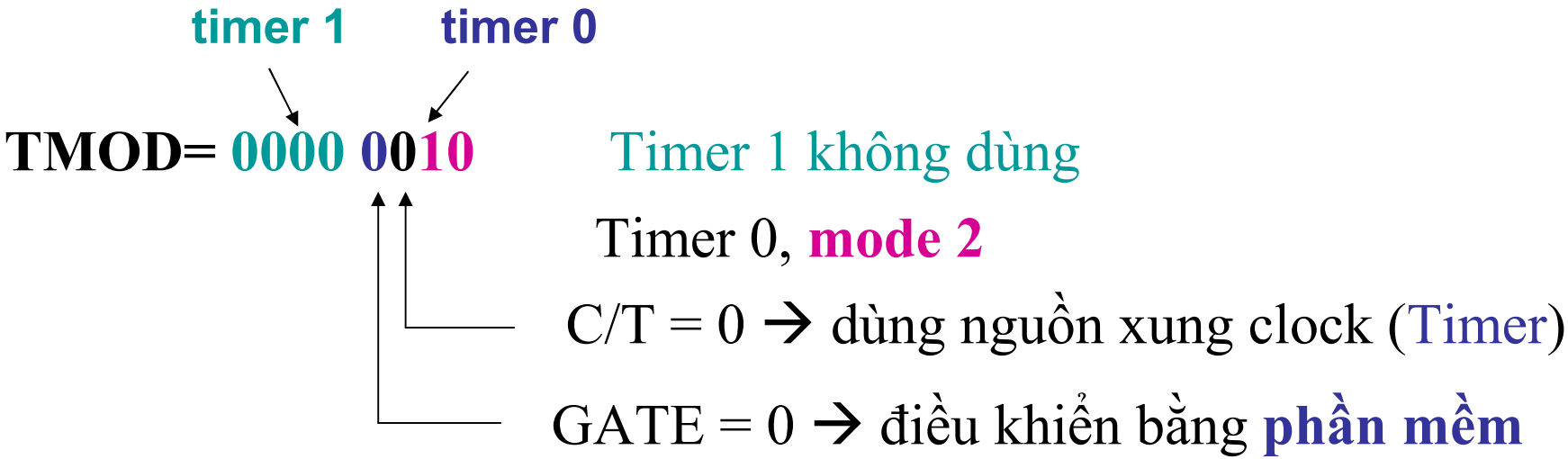

XTAL
oscillator
÷ 12
C/T = 0
C/T = 1
T0 Pin
Pin 3.4
TR0
Gate
INT0 Pin
Pin 3.2

M1, M0

M1	M0	Mode	Hoạt động
0	0	0	Chế độ định thời **13-bit** 8-bit THx + 5-bit TLx (x= 0 or 1)
0	1	1	Chế độ định thời **16-bit** 8-bit THx + 8-bit TLx
1	0	2	Chế độ tự động nạp lại **8-bit** (auto reload mode) 8-bit auto reload timer/counter; THx lưu trữ giá trị nạp lại cho TLx mỗi khi tràn (overflow)
1	1	3	Chế độ định thời chia sẻ (Split timer mode)

VD: Tìm giá trị TMOD nếu muốn lập trình Timer 0 làm việc ở mode 2 ? Dùng 8051 XTAL cho xung clock, & dùng lệnh để bật hay tắt timer

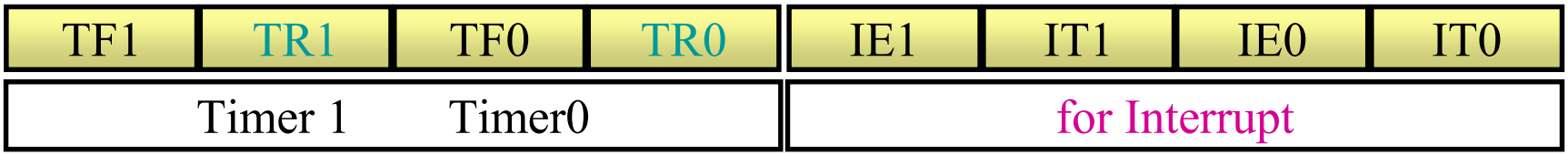
Giải đáp:



10-3. Thanh ghi điều khiển định thời TCON

- Timer control register: **TCON**
  - ½ byte cao cho timer/counter, ½ byte thấp cho interrupts
- **TR** (run control bit)
  - TR0 for Timer/counter 0; TR1 for Timer/counter 1.
  - TR được set hay xóa bởi phần mềm để bật/tắt timer/counter
    - TR=0: off (stop)
    - TR=1: on (start)

(MSB) (LSB)



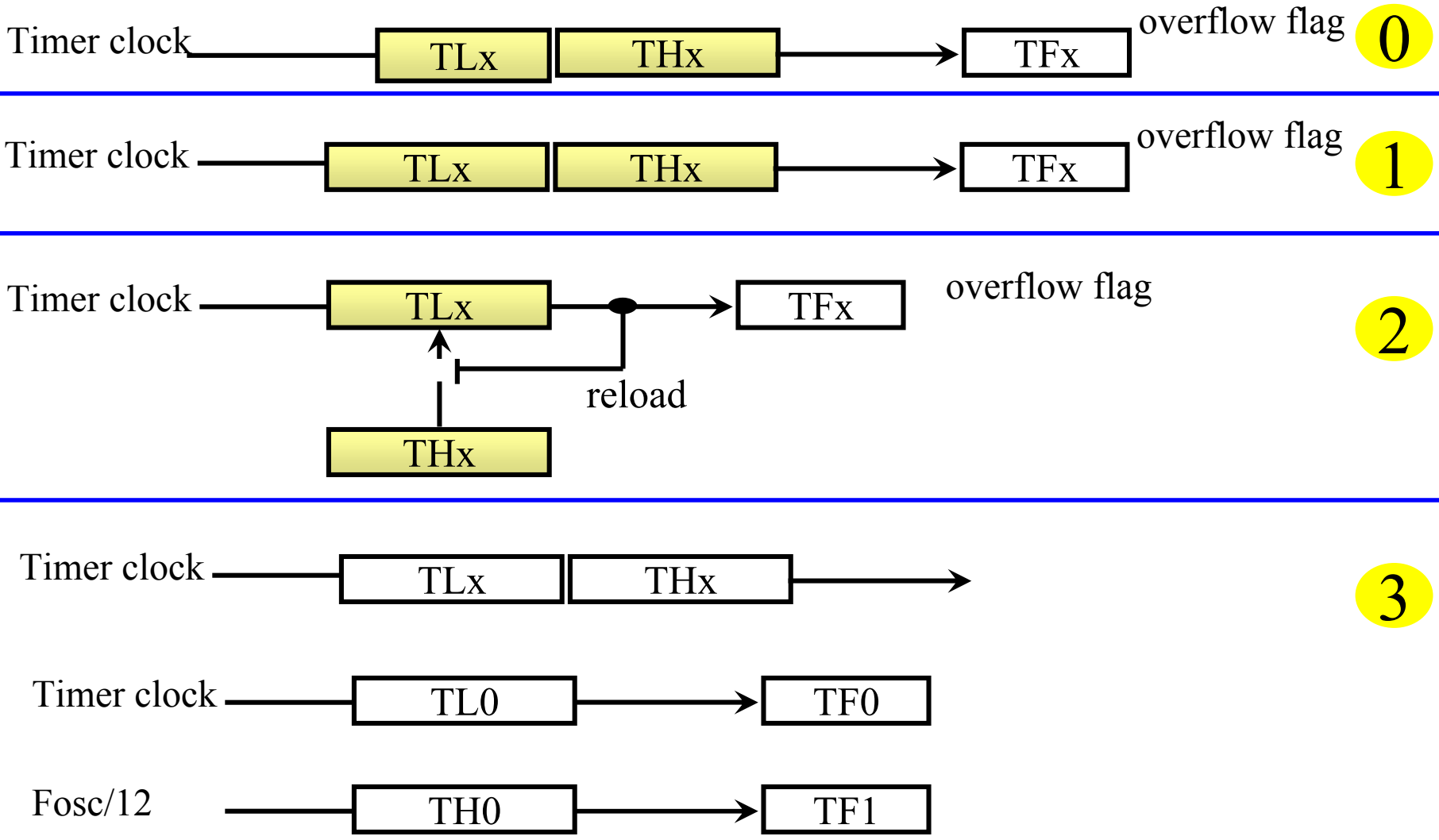
- **TF** (timer flag, control flag)
  - Cờ tràn của bộ định thời
  - TF0 cho timer/counter 0; TF1 cho timer/counter 1
  - Khởi đầu, TF=0 & được set bởi phần cứng khi có tràn – tức TH-TL chuyển đếm về 0000 từ FFFFH
    - Nếu cho phép ngắt, thì TF=1 sẽ kích khởi ISR
  - Được xóa bởi phần mềm (hoặc bởi phần cứng khi bộ xi xử lý trỏ đến trình phục vụ ngắt)

(MSB) (LSB)

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
Timer 1		Timer0		for Interrupt			

10-4. Các chế độ định thời

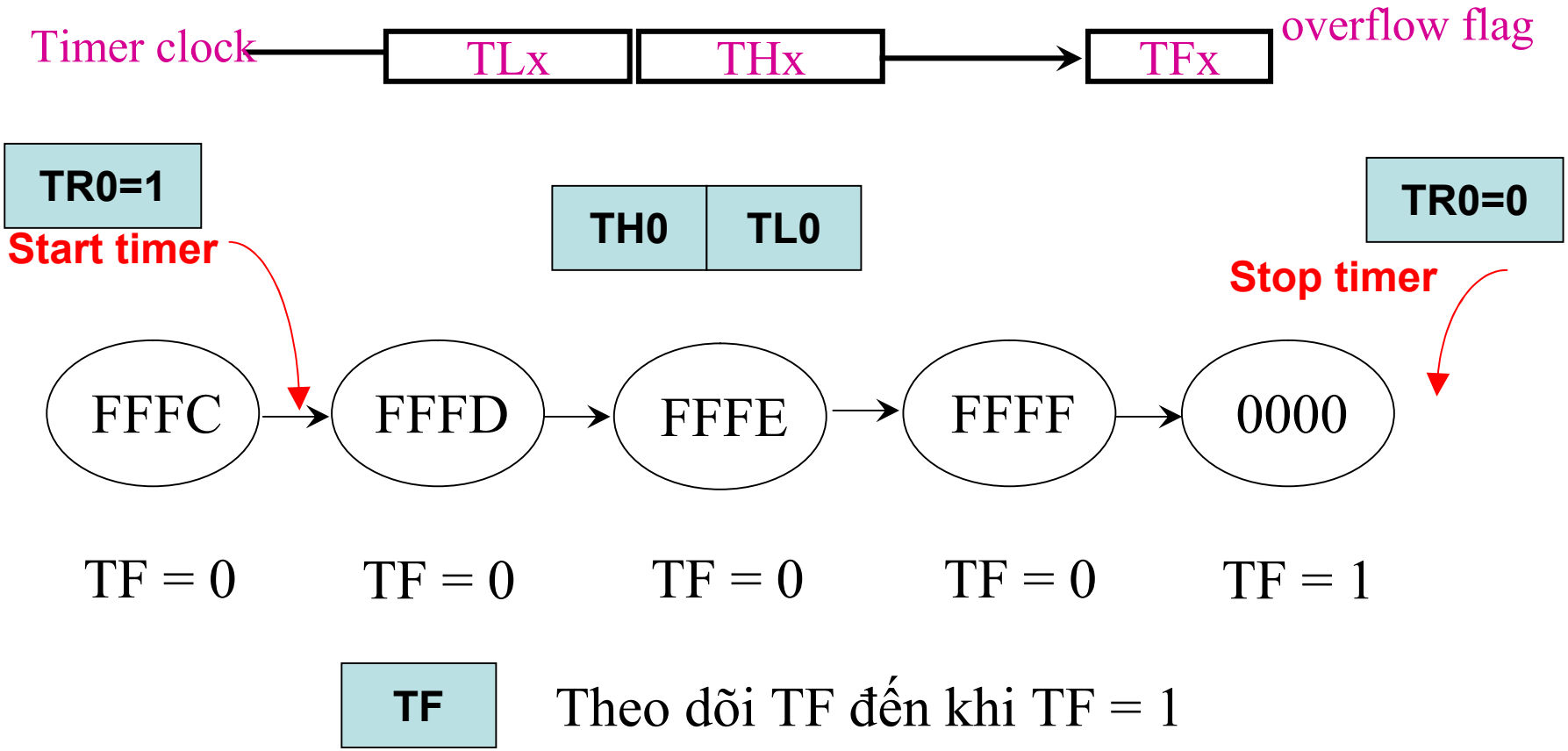
mode



M1	M0	Mode	Hoạt động
0	0	0	Chế độ định thời **13-bit** 8-bit THx + 5-bit TLx (x= 0 or 1)
0	1	1	Chế độ định thời **16-bit** 8-bit THx + 8-bit TLx
1	0	2	Chế độ tự động nạp lại **8-bit** (auto reload mode) 8-bit auto reload timer/counter; THx lưu trữ giá trị nạp lại cho TLx mỗi khi tràn (overflow)
1	1	3	Chế độ định thời chia sẻ (Split timer mode)

10-4-1. Chế độ định thời 16-bit (mode 1)

1. Chọn timer 0 làm việc ở mode 1 (định thời 16-bit)
   - `MOV TMOD,#01H`
2. Khởi tạo giá trị ban đầu nạp cho TH0 & TL0.
   - `MOV TH0,#FFH`
   - `MOV TL0,#FCH`
3. Thoạt đầu nên xóa cờ TF0 về 0
   - `CLR TF0`
4. Bật Timer 0
   - `SETB TR0`

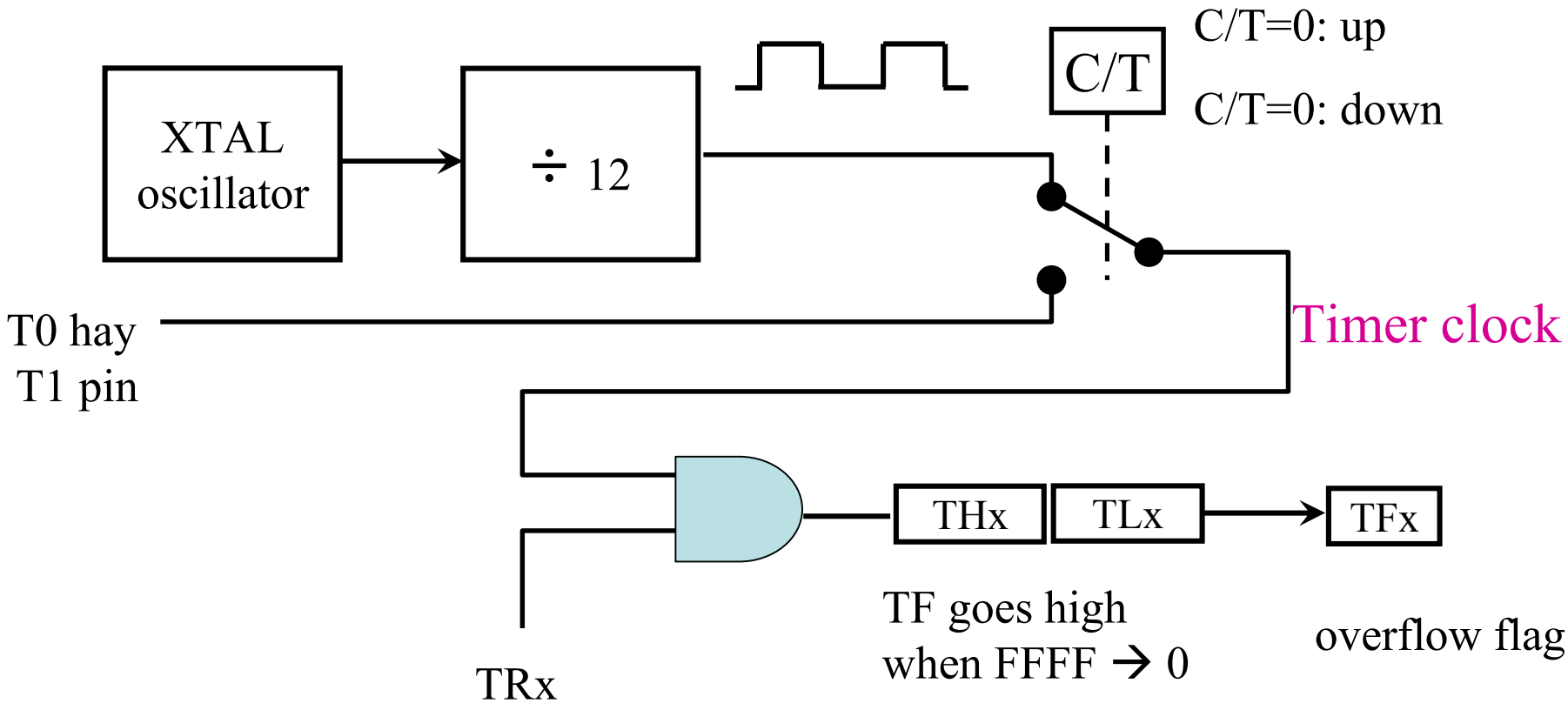


5. Khi có xung clock đến, 8051 bắt đầu đếm lên bằng cách tăng giá trị trong các thanh ghi TH0-TL0

TH0-TL0= FFFCH,FFFDH,FFFEH,FFFFH,0000H

6. Khi TH0-TL0 có sự chuyển số đếm từ **FFFFH → 0000** thì 8051 set TF0=1
   - **TH0-TL0 = FFFEH, FFFFH, 0000H (khi này TF0=1)**
7. Theo dõi cờ tràn (TF)
   - `AGAIN: JNB TF0, AGAIN`
8. Xóa TR0 để tắt timer 0
   - `CLR TR0`
9. Xóa cớ tràn cho vòng tiếp theo
   - `CLR TF0`

Nguồn xung clock cung cấp cho bộ định thời



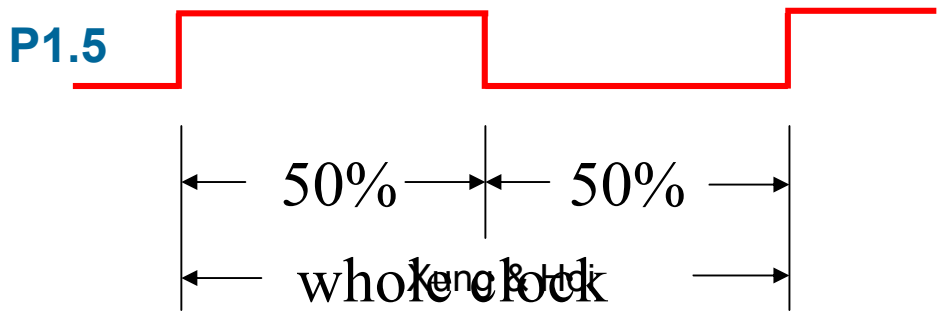
Mode 1

Bài toán 10.1. Sóng vuông

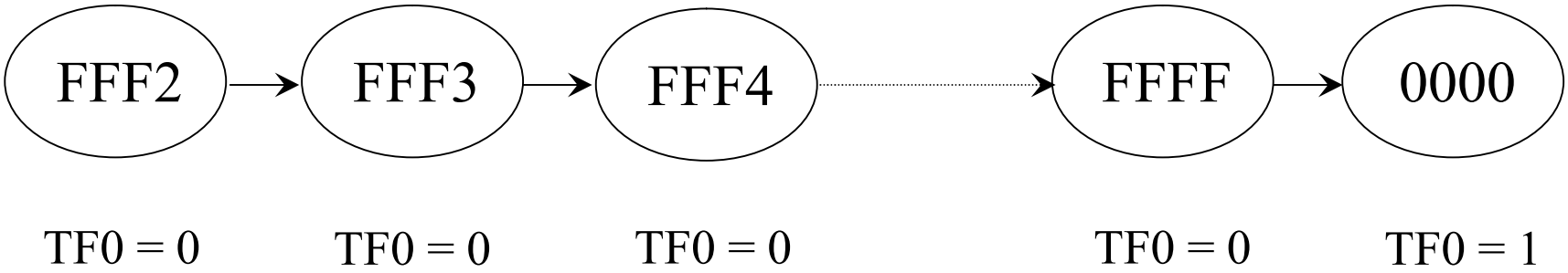
☛Viết chương trình tạo sóng vuông có thời gian mức cao và thấp bằng nhau trên chân P1.5. Dùng Timer 0 tạo trễ mode 1

;each loop is a half clock

```
        MOV   TMOD, #01    ;Timer 0,mode 1(16-bit)
HERE:   MOV   TL0, #0F2H   ;Giá trị Timer0 = FFF2H
        MOV   TH0, #0FFH
        CPL   P1.5
        ACALL        DELAY
        SJMP         HERE
```



```
;tạo trễ dùng timer 0
DELAY:
        SETB  TR0          ;start the timer 0
        JNB   TF0, $
        CLR   TR0          ;stop timer 0
        CLR   TF0          ;clear timer 0 flag
        RET
```



Vài điểm cần chú ý trong VD trên:

1. TMOD = **`0000 0001`** được thực thi
2. **`FFF2H`** chuyển vào TH0 – TL0
3. Set và xóa P1.5 trong thời gian mức cao và thấp của xung
4. CTC DELAY dùng Timer được gọi
5. Trong CTC DELAY, bật timer 0 dùng lệnh "**`SETB TR0`**"
6. Timer 0 đếm lên mỗi khi có xung clock (cung cấp bởi dao động thạch anh)

   Timer 0 đếm qua các giá trị: FFF3, FFF4, FFF5, FFF6, FFF7, FFF8, FFF9, FFFA, FFFB, FFFC, FFFFD, FFFE, FFFFH, 0000H → **`TF0 = 1`** và khi này lệnh JNB bị bỏ qua
7. Tắt Timer 0 dùng lệnh "**`CLR TR0`**". CTC DELAY kết thúc, tiến trình được lặp lại.

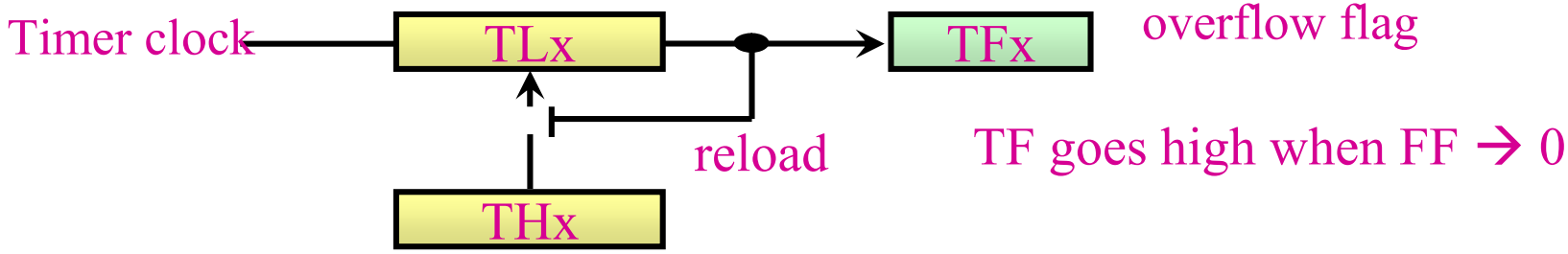
   Chú ý, làm việc với mode 1, nên để lặp lại tiến trình, ta phải nhập lại giá trị cho các thanh ghi TH-TL & bật lại Timer.

10-4-2. Chế độ định thời 13-bit (mode 0)

- Mode 0 tương tự mode 1, ngoại trừ nó là **13-bit** timer thay vì 16-bit.
  - 8-bit TH0 + 5-bit TL0
- Bộ đếm vì thế có giá trị từ 0000 đến 1FFF trong các thanh ghi định thời TH0-TL0
  - 2^{13}-1= 2000H-1=1FFFH
- Ta cũng gán các giá trị ban đầu vào TH0-TL0 để thực hiện đếm lên
- Khi Timer đạt đến 1FFFH thì nó chuyển về 0000, và TF0 được set

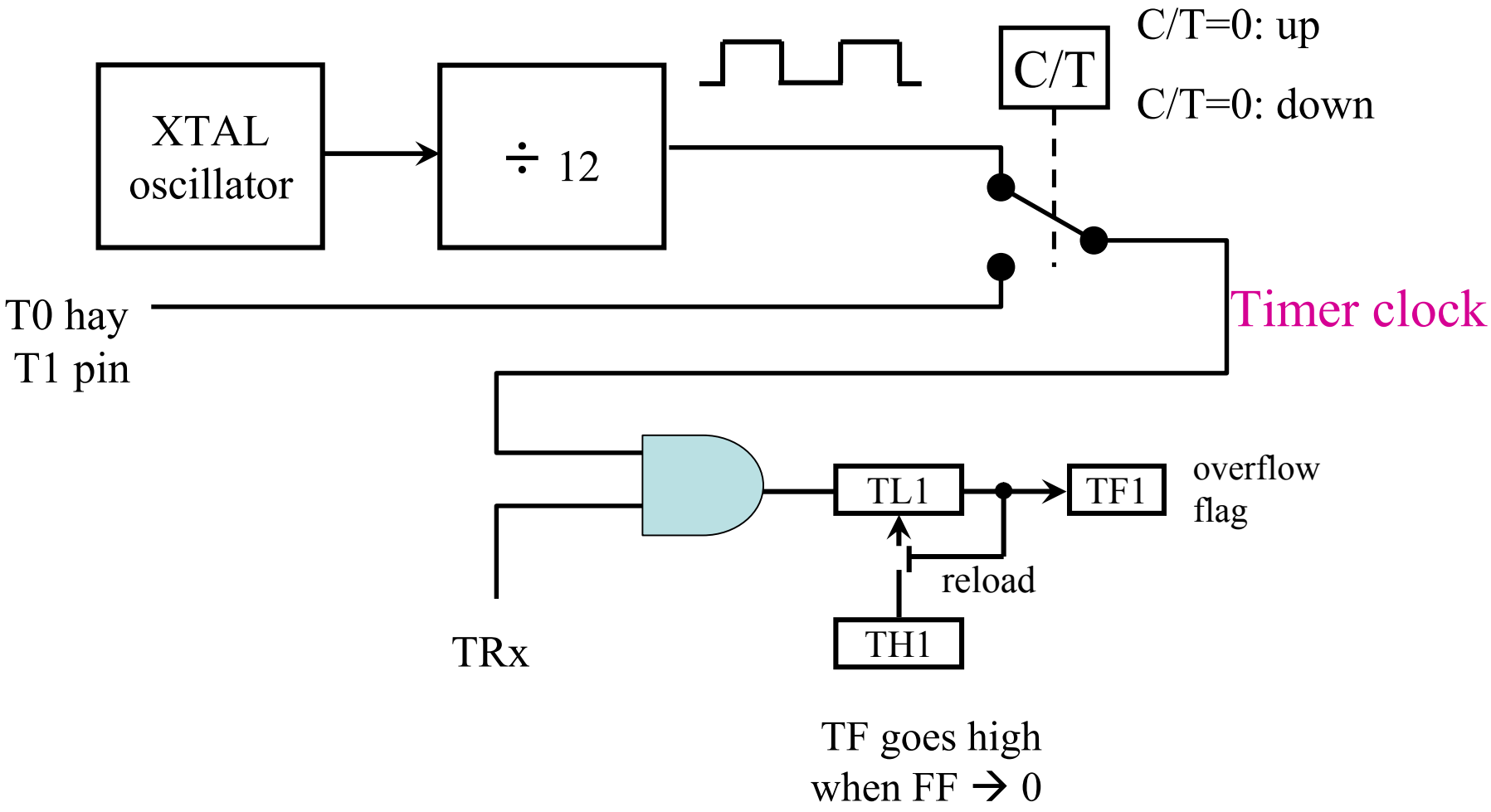
10-4-3. Chế độ tự nạp lại 8-bit (mode 2)

- Bộ định thời 8-bit
  - Cho phép các giá trị từ 00 đến FFH lưu trong TH0
- Tự động nạp lại (Auto-reloading)
- TL0 được tăng lên khi TR0=1
- VD, tạo thời gian trễ với 200 chu kỳ máy (MC) trên timer 0.



1. Chọn mode 2 timer 0
   - `MOV TMOD,#02H`
2. Khởi tạo giá trị đầu trong TH0
   - `MOV TH0,#38H`
3. Xóa cờ TF0=0.
   - `CLR TF0`
4. Sau khi TH0 được gán, 8051 copy giá trị này vào TL0
   - **TL0=TH0=38H**
5. Bật Timer
   - `SETB TR0`

6. 8051 đếm lên TL0
   - **TL0= 38H, 39H, 3AH,....**
7. Khi TL0 từ FFH → 00 thì 8051 set TF0=1. Bên cạnh đó, TL0 được tự động nạp lại giá trị lưu giữ trong TH0
   - **TL0= FEH, FFH, 00H (lúc này TF0=1)**
   - 8051 tự động nạp lại **TL0=TH0=38H.**
   - Quay lại bước 6
8. CLR TF0
9. CLR TR0


XTAL
oscillator
÷ 12
C/T
C/T=0: up
C/T=0: down
T0 hay
T1 pin
Timer clock
TL1
TF1
overflow
flag
reload
TH1
TRx
TF goes high
when FF → 0

Mode 2

10-5. Tính toán trễ dùng Timer

Áp dụng đ/v chế độ định thời 16-bit (mode 1)

(a) Dạng hexa

(FFFF – YYXX + 1) * 1.085 µs

Trong đó YYXX là các giá trị ban đầu của TH, TL tương ứng

(b) Dạng thập phân

Chuyển các giá trị YYXX của TH, TL sang dạng thập phân NNNNN, → ta có

(65536 – NNNNN) * 1.085 µs

XTAL = 11.0592 MHz

Bài toán 10.2. Sóng vuông, tìm f

- Chương trình sau tạo sóng vuông trên chân P1.5 liên tục dùng timer 1 tạo trễ mode 1. Tìm tần số?

  (Không bao gồm overhead gây bởi các lệnh trong vòng lặp)

```
        MOV   TMOD,#10H ;timer 1, mode 1
AGAIN:MOV   TL1,#34H   ;timer value=7634H
      MOV   TH1,#76H
      SETB  TR1        ;start
BACK: JNB   TF1,BACK
      CLR   TR1        ;stop
      CPL   P1.5       ;next half clock
      CLR   TF1        ;clear timer flag 1
      SJMP  AGAIN      ;reload timer1
```

Giải đáp:

Với mode 1, các thanh ghi định thời TH, TL phải được khởi động lại sau mỗi lần tràn.

FFFFH – 7634H + 1 = 89CCH = 35276 clock count

½ chu kỳ = 35276 x 1.085 μs = 38.274 ms

Chu kỳ = 2 x 38.274 ms = 76.548 ms

Tần số = 1/ 76.548 ms = 13.064 Hz.

Trong các tính toán trên, không bao gồm overhead gây bởi các lệnh trong vòng lặp

Bài toán 10.3.

- ☛Tìm KQ các bài toán trên trong trường hợp có tính đến overhead?
- ☛Giải các bài toán trên với chế độ định thời tự nạp lại 8-bit (mode 2)?
- ☛Với phạm vi tần số nào thì không thể dùng chế độ tự nạp lại 8-bit (mode 2), giải thích?

10-6. Tìm giá trị các thanh ghi định thời

- Giả định biết trước thời gian trễ, XTAL = 11.0592 MHz
- Làm sao tính toán các giá trị cần gán cho TH, TL?

1. Chia thời gian trễ cho **1.085 μs.**
2. Thực hiện **65536 –n**, với **n** (decimal) từ bước 1
3. Chuyển KQ trong bước 2 sang hex **yyxx**
4. Set **TH = yy** và **TL = xx**.

Bài toán 10.4. Tạo sóng vuông theo f cho sẵn

☛XTAL = 11.0592 MHz, viết chương trình tạo sóng vuông 50 Hz trên chân P2.3

Giải đáp:

(a) T = 1 / 50 Hz = 20 ms.

(b) Thời gian mức cao = Thời gian mức thấp = 10 ms.

(c) 10 ms / 1.085 μs = 9216

65536 – 9216 = 56320 in decimal = DC00H in hex.

(d) TL1 = 00H and TH1 = DCH.

```
        MOV   TMOD,#10H   ;timer 1, mode 1
AGAIN:  MOV   TL1,#00     ;Timer value = DC00H
        MOV   TH1,#0DCH
        SETB  TR1         ;start
BACK:   JNB   TF1,BACK
        CLR   TR1         ;stop
        CPL   P2.3
        CLR   TF1         ;clear timer flag 1
        SJMP  AGAIN       ;reload timer since
                          ;mode 1 is not
                          ;auto-reload
```

Tạo thời gian trễ lớn

- Độ lớn t_{delay} phụ thuộc 2 thông số:
  - Tần số dao động thạch anh
  - Giá trị chứa trong các thanh ghi định thời TH & TL
- Thời gian trễ là lớn nhất khi TH=TL=0.

Nếu vẫn chưa đủ?

Bài toán 10.5. Tính delay

☛Khảo sát BT sau và tìm thời gian trễ? *(Không tính overhead)*

```
        MOV   TMOD,#10H
        MOV   R3,#200
AGAIN:  MOV   TL1,#08H
        MOV   TH1,#01H
        SETB  TR1
BACK:   JNB   TF1,BACK
        CLR   TR1
        CLR   TF1
        DJNZ  R3,AGAIN
```

Giải đáp:

TH – TL = 0108H = 264 (decimal)

65536 – 264 = 65272.

Trễ do timer = 65272 × 1.085 μs = 70.820 ms

Tổng thời gian trễ = 200 × 70.820 ms = 14.164024 seconds

Bài toán 10.6. Tính tần số xung vuông

☛Tìm tần số xung vuông trên chân P1.0 ?

```
        MOV   TMOD,#2H    ;Timer 0,mode 2
        MOV   TH0,#0
AGAIN:  MOV   R5,#250     ;count 250 times
        ACALL DELAY
        CPL   P1.0
        SJMP  AGAIN
DELAY:  SETB  TR0         ;start
BACK:   JNB   TF0,BACK
        CLR   TR0         ;stop
        CLR   TF0         ;clear TF
        DJNZ  R5,DELAY    ;timer 2: auto-reload
        RET
```

Giải đáp: T = 2 (250 x 256 x 1.085 μs) = 138.88 ms → f = 72 Hz.

Bài toán 10.7. Tìm giá trị gán cho TH

☛Giả sử đang lập trình cho Timer ở mode 2, tìm giá trị hex gán cho thanh ghi TH trong các trường hợp sau:

(a) `MOV TH1,#-200`

(b) `MOV TH0,#-60`

(c) `MOV TH1,#-3`

(d) `MOV TH1,#-12`

(e) `MOV TH0,#-48`

- Giải đáp: Vài 8051 assemblers cho phép cách thức sau:
- -200 = -C8H → 2's complement of –200 = 100H – C8H = 38H

Decimal value)	*2's complement (TH*
-200 = - C8H	**38H**
- 60 = - 3CH	**C4H**
- 3	**FDH**
- 12	**F4H**
- 48	**D0H**

Bài toán 10.8. Tìm f

(a) Tìm tần số sóng vuông với đoạn code sau

(b) Thời gian mức cao và thấp

```
MOV         TMOD,#2H          ;Timer 0,mode 2
            MOV   TH0,#-150   ;Count=150
AGAIN:SETB        P1.3
      ACALL       DELAY
      ACALL       DELAY
      CLR         P1.3
      ACALL       DEALY
      SJMP        AGAIN
DELAY:SETB  TR0               ;start
BACK: JNB   TF0,BACK
      CLR   TR0               ;stop
      CLR   TF0               ;clear TF
      RET
```

high period

low period

Giải đáp:

"`MOV TH0,#-150`" dùng 150 clocks.

CTC DELAY = 150 × 1.085 µs = 162 µs.

Thời gian mức cao gấp 2 lần thời gian mức thấp (66% duty cycle).

Chu kỳ = Thời gian mức cao + thời gian mức thấp

= 325.5 µs + 162.25 µs = 488.25 µs

Tần số = 2.048 kHz.

10-7. Bộ đếm

- Các bộ định thời (timers) cũng có thể dùng như những bộ đếm (counters) để đếm ***sự kiện*** xảy ra bên ngoài 8051
- Khi đó, xung từ ngoài sẽ làm tăng giá trị các thanh ghi TH, TL
- Khi C/T=1, bộ đếm sẽ đếm lên khi có xung xuất hiện từ:
  - T0: timer 0 input (Pin 14, P3.4)
  - T1: timer 1 input (Pin 15, P3.5)

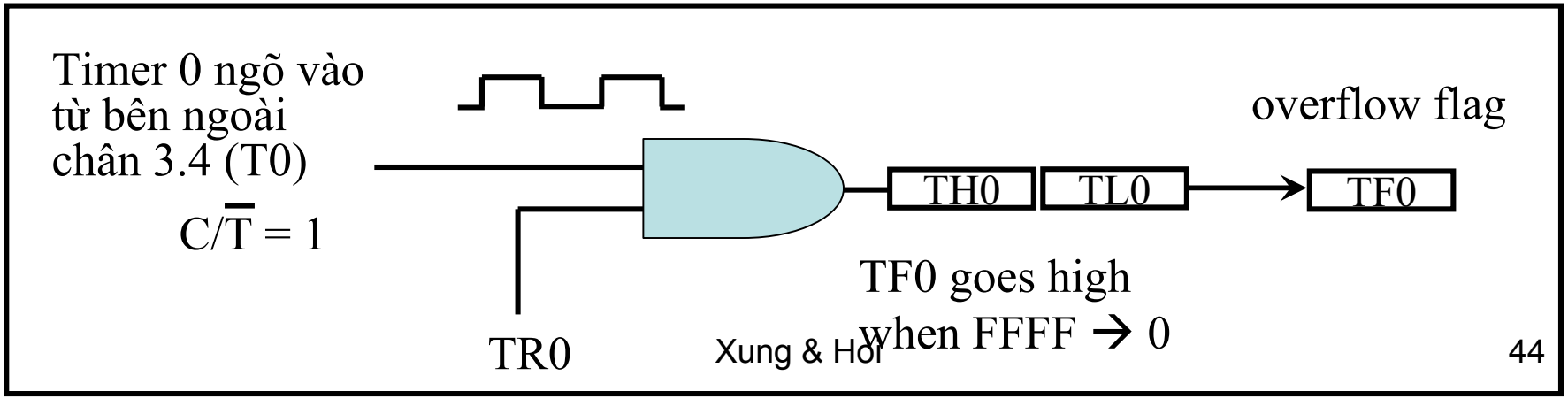
Pin	*Port Pin*	*Function*	*Description*
14	**P3.4**	**T0**	**Timer/Counter 0 external input**
15	**P3.5**	**T1**	**Timer/Counter 1 external input**

(MSB) (LSB)

GATE	C/T=1	M1	M0	GATE	C/T=1	M1	M0
Timer 1				Timer 0			

10-7-1. Chế độ đếm 16-bit (mode 1)

- **16-bit** counter (TH0 and TL0)
- Giá trị trong các thanh ghi TH0-TL0 tăng khi: TR0 được set lên 1 và một xung bên ngoài (T0) xuất hiện
- Khi bộ đếm (TH0-TL0) đạt tới giá trị lớn nhất là FFFFH, nó được chuyển trạng thái về 0000, và TF0 được set lên 1
- Bằng cách nạp giá trị ban đầu cho TH0-TL0, theo dõi TF0=1 để nhận biết 1 tình huống nào đó (vd: 100 người đã đến).

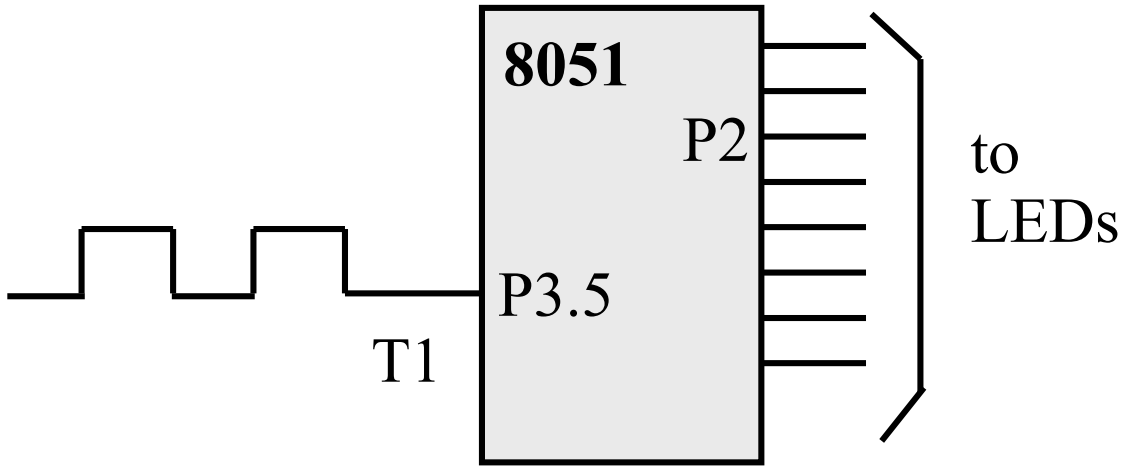


10-7-2. Chế độ đếm tự nạp lại 8-bit (mode 2)

- 8-bit **counter**.
  - Chỉ cho phép các giá trị từ 00 đến FFH nạp vào TH0
- Tự động nạp lại (Auto-reloading)
  Giá trị trong TL0 sẽ tăng nếu TR0=1 và 1 xung bên ngoài xuất hiện

Bài toán 10.9. Đếm xung & xuất port

Giả định 1 xung clock được đưa vào chân T1, viết chương trình bộ đếm 1 làm việc ở mode 2 để đếm xung & hiển thị giá trị của TL1 ra P2, khi Counter tràn thì kết thúc?

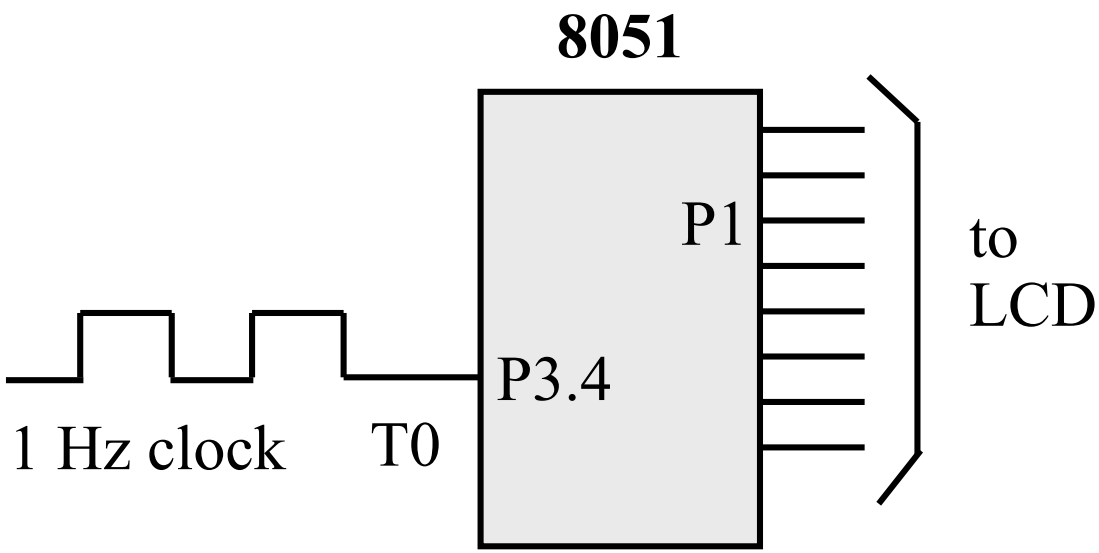


Ans:

```
                MOV  TMOD, #01100000B  ;mode 2, counter 1
                MOV  TH1, #0
                SETB P3.5
                SETB TR1               ;start
BACK:           MOV  A,TL1
                MOV  P2,A              ;display in P2
                JNB  TF1,BACK          ;overflow
                CLR  TR1               ;stop
                CLR  TF1               ;make TF=0
```

Bài toán 10.10. Đếm xung & Hiển thị LCD

- Giả sử 1 xung tần số 1Hz được cấp vào chân P3.4. Viết chương trình hiển thị counter 0 trên LCD. Khởi tạo giá trị ban đầu cho thanh ghi TH0 là -60.



```
        ACALL LCD_SET_UP        ;initialize the LCD
        MOV  TMOD,#00000110B    ;Counter 0,mode2
        MOV  TH0,#-60
        SETB P3.4               ;make T0 as input
AGAIN:
        SETB TR0                ;starts the counter
BACK:
        MOV  A,TL0              ;every 60 events
        ACALL CONV              ;convert in R2,R3,R4
        JNB  TF0,BACK           ;loop if TF0=0
        CLR  TR0                ;stop
        CLR  TF0
        SJMP AGAIN
```

;converting 8-bit binary to ASCII

```
CONV: MOV   B,#10      ;divide by 10
      DIV   AB
      MOV   R2,B       ;save low digit
      MOV   B,#10      ;divide by 10 once more
      DIV   AB
      ORL   A,#30H     ;make it ASCII
      MOV   R4,A
      MOV   A,B
      ORL   A,#30H
      MOV   R3,A
      MOV   A,R2
      ORL   A,#30H
      MOV   R2,A       ;ACALL LCD_DISPLAY here
      RET
```

Bài toán 10.11. Đồng hồ số

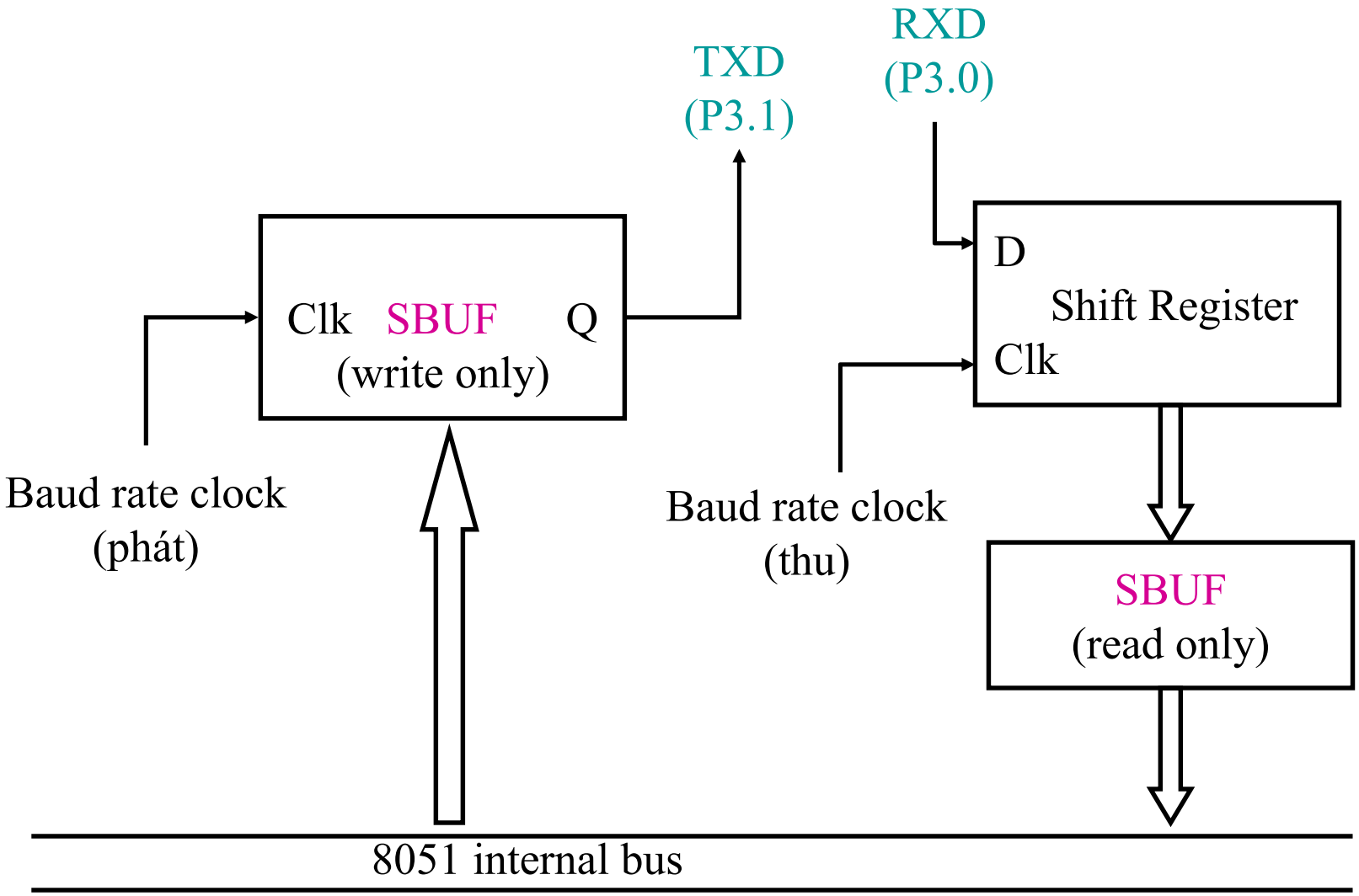
- Cải tiến ví dụ trên thực hiện 1 đồng hồ số đơn giản, chưa cần các nút hiệu chỉnh giờ, phút ?

- Việc sử dụng lệnh "`JNB TF0,target`" để giám sát cờ TF0 là sự hoang phí thời gian vô cùng lớn
  - Giải pháp là dùng ngắt, đề cập trong phần 12 của bài giảng
  - Với ngắt, ta có thể thực thi nhiều việc
  - Khi cờ TF được set, nó sẽ thông báo cho chúng ta

1. Giới thiệu
2. Sơ đồ khối và chân
3. Tổ chức bộ nhớ
4. Các thanh ghi chức năng đặc biệt (SFR)
5. Dao động và hoạt động reset
6. Tập lệnh
7. Các mode định địa chỉ (addressing modes)
8. Lập trình IO
9. Tạo trễ
10. Lập trình Timer/Counter
11. Lập trình giao tiếp nối tiếp (serial comm programming)
12. Lập trình ngắt (interrupt programming)
13. Lập trình hợp ngữ

11-1. Giới thiệu

- Port nối tiếp hoạt động song công (full duplex), nghĩa là có khả năng thu và phát đồng thời
- Sử dụng 2 thanh ghi chức năng đặc biệt SBUF (địa chỉ byte là 99H) & SCON (địa chỉ byte là 98H) để truy xuất port nối tiếp
- Việc ghi lên SBUF sẽ nạp dữ liệu để phát, và việc đọc SBUF sẽ truy xuất dữ liệu đã nhận được → thực ra có 2 SBUF riêng rẽ
- SCON chứa các bit trạng thái và điều khiển, thanh này được định địa chỉ bit
- Tần số hoạt động của port nối tiếp hay còn gọi là tốc độ baud (baud rate) có thể cố định hoặc thay đổi


TXD
(P3.1)
RXD
(P3.0)
Clk SBUF Q
(write only)
D
Shift Register
Clk
Baud rate clock
(phát)
Baud rate clock
(thu)
SBUF
(read only)
8051 internal bus

11-2. Thanh khi điều khiển port nối tiếp

SCON

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

SM1, SM0: chọn chế độ của port nối tiếp

SM2: cho phép truyền thông đa xử lý

REN: cho phép thu, phải được set để nhận các ký tự

TB8: bit thứ 9 được phát ở chế độ 2 & 3

RB8: bit thứ 9 nhận được

TI: cờ ngắt phát, được set ngay sau khi kết thúc việc phát 1 ký tự; được xóa bởi phần mềm

RI: cờ ngắt thu, được set ngay sau khi kết thúc việc thu 1 ký tự; được xóa bởi phần mềm

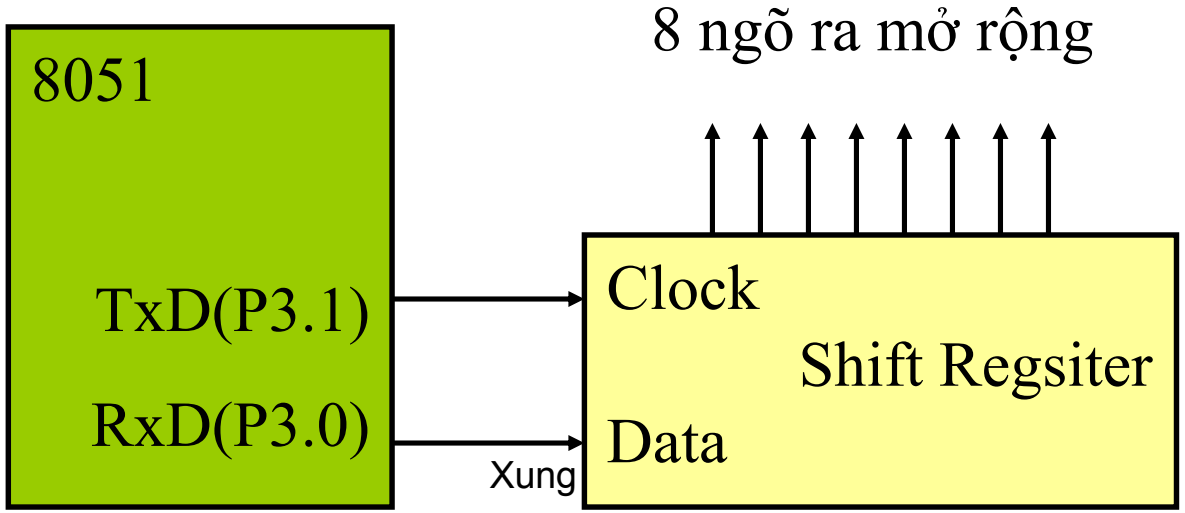
11-3. Các chế độ hoạt động

SM0	SM1	Mode	Mô tả	Tốc độ baud
0	0	0	Thanh ghi dịch	Cố định ($f_{OSC}/12$)
0	1	1	UART 8-bit	Thay đổi (thiết lập bởi bộ định thời)
1	0	2	UART 9-bit	Cố định ($f_{OSC}/12$ hoặc $f_{OSC}/64$)
1	1	3	UART 9-bit	Thay đổi (thiết lập bởi bộ định thời)

11-3-1. Thanh ghi dịch 8-bit (mode 0)

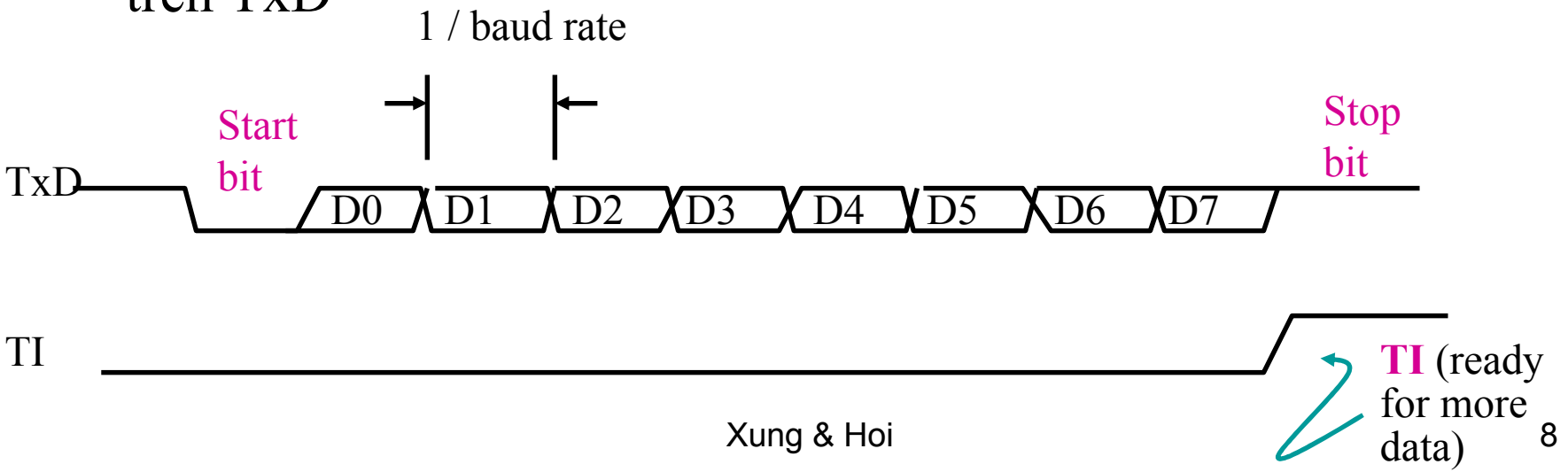
- Khi phát và thu dữ liệu 8-bit, bit LSB được phát hoặc thu trước tiên
- Tốc độ baud cố định = $f_{OSC}/12$
- Chân RxD dùng cho cả việc thu phát dữ liệu trong khi TxD dùng làm chân xuất clock dịch bit
- Phát:
  - Ghi vào SBUF
  - Dữ liệu được dịch ra ngoài trên chân RxD (P3.0)
  - Xung clock dịch bit được gửi ra trên chân TxD (P3.1)
  - Mỗi bit hợp lệ truyền đi trên RxD trong 1 chu kỳ máy

- Thu:
  - Chỉ được khởi động khi REN = 1 & RI = 0, nghĩa là phải set REN = 1 ở thời điểm bắt đầu chương trình & xóa RI để bắt đầu công việc thu dữ liệu
  - Dữ liệu được dịch vào chân RxD bởi xung clock dịch bit (tác động sườn lên)
- 1 Ứng dụng của mode 0:
  - Mở rộng thêm các ngõ ra cho 8051



11-3-2. UART 8-bit có tốc độ baud thay đổi (mode 1)

- UART là bộ thu phát dữ liệu nối tiếp với mỗi ký tự dữ liệu được đứng trước bởi 1 bit start và đứng sau bởi 1 bit stop. Thỉnh thoảng 1 bit parity được chèn vào.
- Hoạt động chủ yếu của UART là biến dữ liệu phát từ // thành nt và biến dữ liệu thu từ nt thành //
- Cờ ngắt phát TI được set bằng 1 ngay khi bit stop xuất hiện trên TxD



- Việc nhận được khởi động bởi 1 chuyển trạng thái từ 1 xuống 0 trên đường RxD (bit start)
- Bit start sau đó được bỏ qua & 8 bit dữ liệu sau đó được nhận tuần tự vào thanh ghi dịch bit của port nối tiếp. Khi cả 8 bit được nhận, ta có:
  - Bit thứ 9 (bit stop) → RB8 của SCON
  - 8 bit dữ liệu được nạp vào SBUF
  - Cờ ngắt thu RI được set
- Note: Các điều trên chỉ xảy ra nếu trước đó cờ RI = 0

11-4. Khởi động và truy xuất các thanh ghi

- Cho phép nhận: REN trong SCON phải được set bởi phần mềm để cho phép nhận ký tự

```
SETB    REN
```

Hoặc

```
MOV     SCON, #xxx1xxxxB
```

- Thêm vào bit chẵn lẻ: bit P trong PSW được set hoặc xóa ở mỗi chu kỳ máy để thiết lập việc kiểm tra chẵn cho 8 bit chứa trong A

```
MOV     C, P        ; đưa bit kiểm tra chẵn vào TB8
MOV     TB8, C      ; bit này trở thành bit thứ 9
MOV     SBUF, A     ; di chuyển 8 bit dữ liệu vào SBUF
```

- Các cờ ngắt: RI và TI được set bằng phần cứng nhưng phải xóa bằng phần mềm

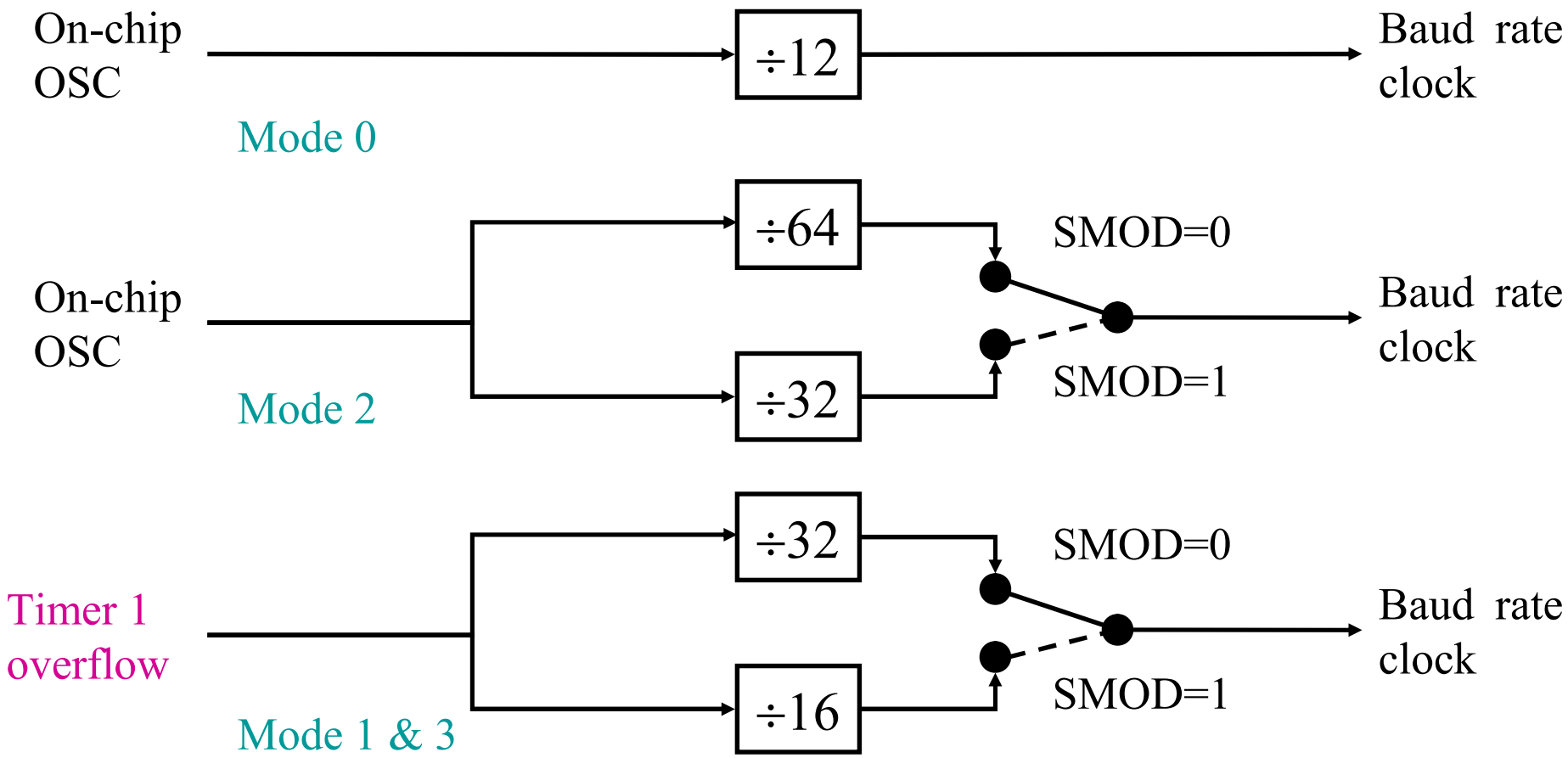
CTC nhận 1 ký tự qua port nối tiếp như sau:

```
INCHAR:   JNB   RI, $
          CLR   RI
          MOV   A, SBUF
          RET
```

CTC phát 1 ký tự qua port nt:

```
OUTCHAR:  JNB   TI, $
          CLR   TI
          MOV   SBUF, A
          RET
```

11-5. Tốc độ baud của port nối tiếp



Dùng timer 1 làm xung clock tốc độ baud

- Thông thường khởi động thanh ghi TMOD ở chế độ tự động nạp lại 8-bit (mode 2) & đặt giá trị nạp lại thích hợp vào thanh ghi TH1 để có tốc độ tràn đúng, từ đó tạo ra tốc độ baud
- MOV TMOD, #0010xxxxB ;for timer 1
- Baud rate = Timer 1 overflow / 32 hay /16 tùy thuộc giá trị bit SMOD

Bài toán 11.1

- Tính toán các giá trị nạp lại cho thanh ghi TH1 đối với các tốc độ baud 9600, 4800, 2400, 1200 (XTAL = 11,0592MHz)?

Tốc độ baud	Tần số thạch anh (MHz)	SMOD	Giá trị nạp cho TH1	Tốc độ baud thực tế	Sai số
9600	12	1	-7 (F9H)	8929	~7%
1200	12	0	-26 (E6H)	1202	0,16%
19200	11,0592	1	-3 (FDH)	19200	0%
9600	11,0592	0	-3 (FDH)	9600	0%
4800	11,0592	0	-6 (FAH)	4800	0%
2400	11,0592	0	-12 (F4H)	2400	0%
1200	11,0592	0	-24 (E8H)	1200	0%

Tóm lại:

Nếu PCON.7 = 0

- TH1 = 256 - ((Crystal / 384) / Baud)

Nếu PCON.7 =1

- TH1 = 256 - ((Crystal / 192) / Baud)

Làm vd với baud rate 19200 để chứng tỏ vai trò bit SMOD

11.059Mhz crystal:

- TH1 = 256 - ((Crystal / 384) / Baud)
  TH1 = 256 - ((11059200 / 384) / 19200)
  TH1 = 256 - ((28799) / 19200)
  TH1 = 256 - 1.5 = 254.5

- Nếu set 254 ta đạt được 14,400 baud còn với 255 ta đạt được 28,800 baud
- Set PCON.7 (SMOD). Ta có:

  TH1 = 256 - ((Crystal / 192) / Baud)
  TH1 = 256 - ((11059200 / 192) / 19200)
  TH1 = 256 - ((57699) / 19200)
  TH1 = 256 - 3 = 253

Kết luận với 19,200 baud (11.0592MHz crystal) ta phải:

1. Cấu hình Port nối tiếp mode 1 or 3
2. Timer 1 hoạt động mode 2 (8-bit auto-reload)
3. Set TH1 bằng 253
4. Set PCON.7 (SMOD)

Các bài toán

- ☛*BT11.1.* Khởi động port nt: Viết chuỗi lệnh khởi động port nối tiếp hoạt động như 1 UART 8-bit với tốc độ baud 2400. Dùng timer 1 để cung cấp xung clock tốc độ baud?
- ☛*BT11.2.* CTC xuất ký tự: Viết CTC mang tên OUTCHAR phát mã ASCII 7-bit chứa trong A ra port nối tiếp với bit kiểm tra chẵn là bit thứ 8. Việc trở về từ CTC phải không làm thay đổi nội dung thanh chứa A?
- ☛*BT11.3.* CTC thu ký tự: Viết CTC mang tên INCHAR để thu 1 ký tự từ port nối tiếp & trả về mã ASCII 7-bit trong A. Sử dụng kiểm tra chẵn trong bit thứ 8 thu được và set cờ nhớ bằng 1 nếu có lỗi chẵn lẻ?

MAX 232

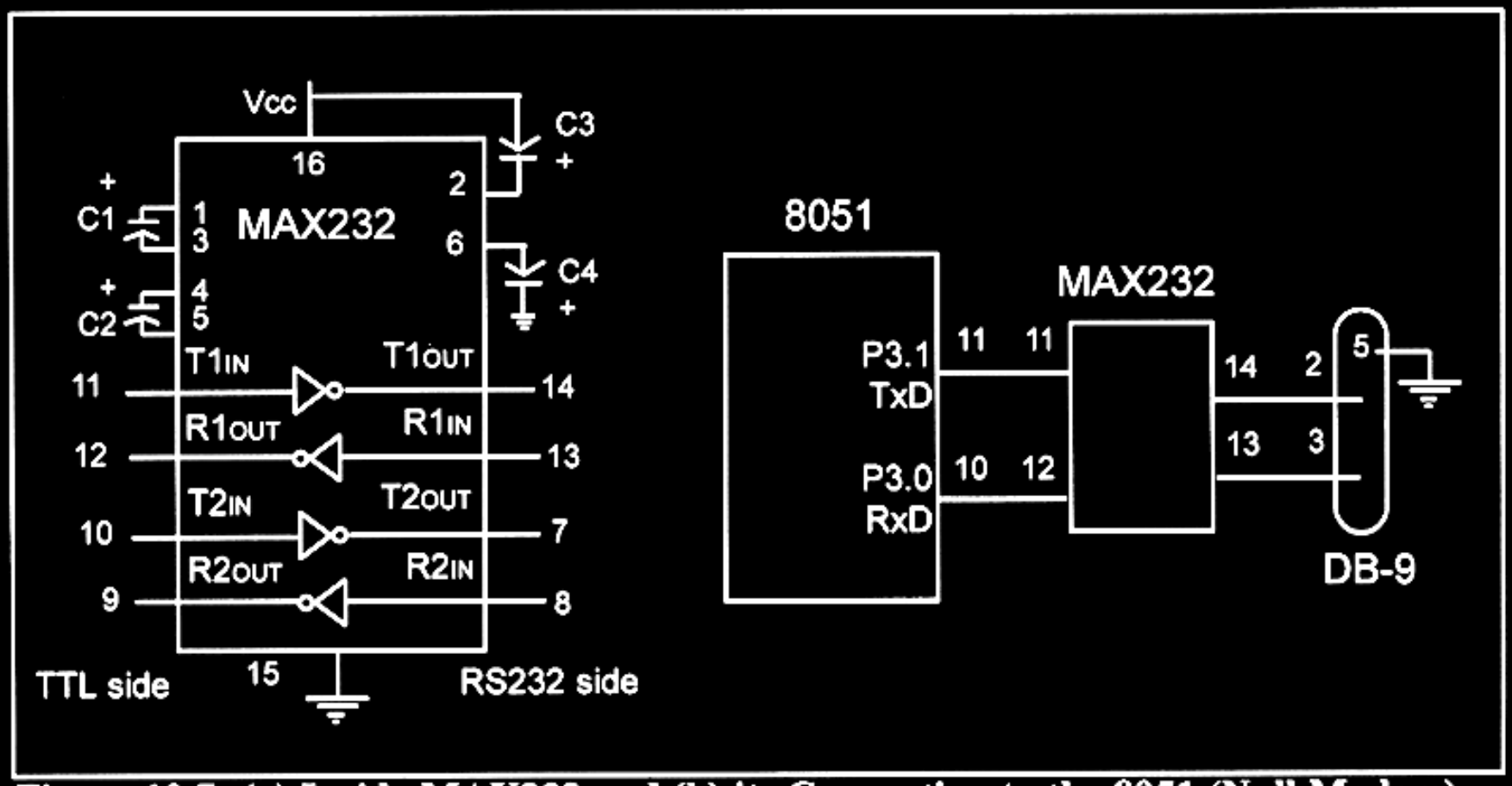
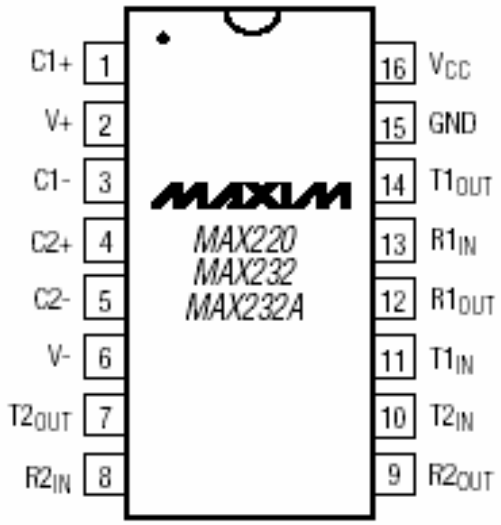
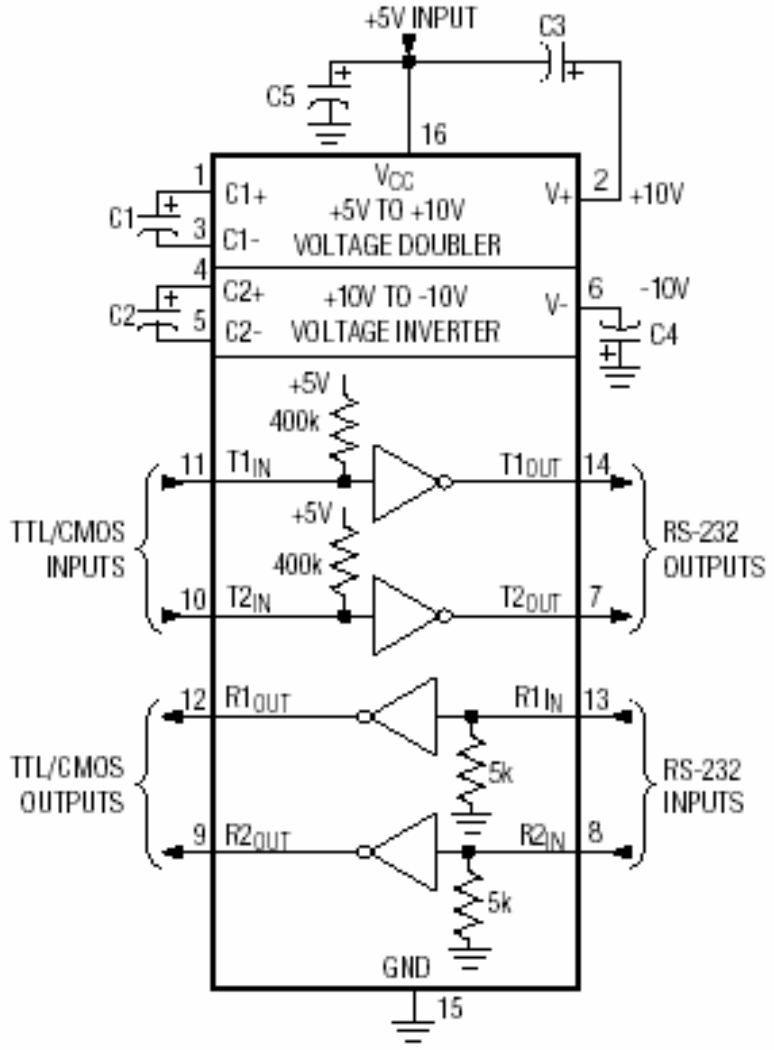


Figure 10.7: (a) Inside MAX232 and (b) its Connection to the 8051 (Null Modem)

TOP VIEW



DIP/SO

CAPACITANCE (µF)					
DEVICE	C1	C2	C3	C4	C5
MAX220	4.7	4.7	10	10	4.7
MAX232	1.0	1.0	1.0	1.0	1.0
MAX232A	0.1	0.1	0.1	0.1	0.1