Hiệu đính từ slide của thầy Hồ Trung Mỹ
Bộ môn Điện tử - DH BK TPHCM

CHƯƠNG 3

HỌ VI ĐIỀU KHIỂN 8051

Nội dung

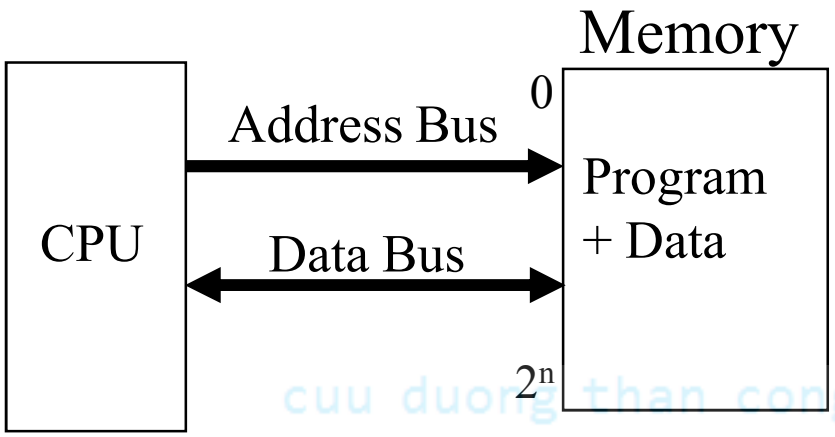
3.1 Giới thiệu họ vi điều khiển 8051

3.2 Kiến trúc phần cứng 8051

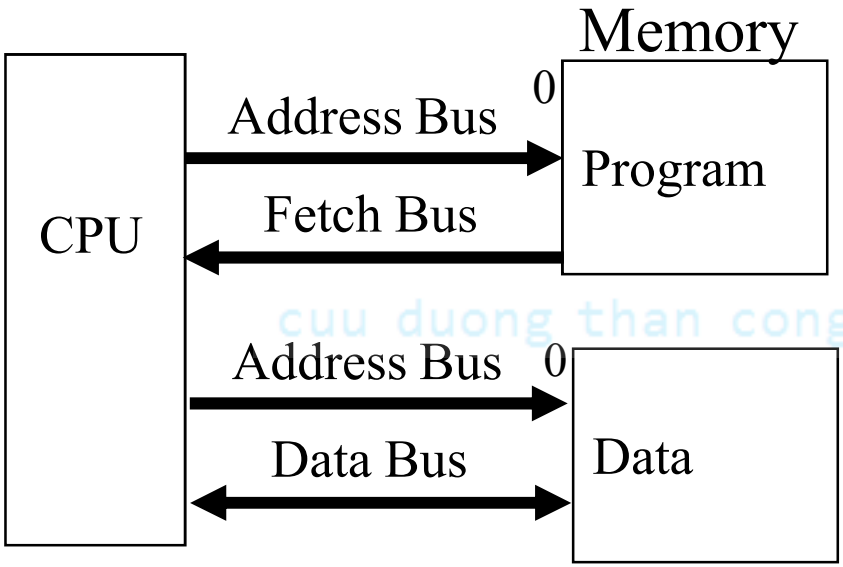
3.3 Các phương pháp định địa chỉ

3.1 Giới thiệu họ vi điều khiển 8051

Các kiến trúc vi điều khiển



Von Neumann Architecture



Harvard Architecture

Họ VĐK 8051

- 8051 là vi điều khiển đầu tiên của họ vi điều khiển MCS51 được Intel sản xuất vào năm 1980. Họ MCS51 là họ 8-bit có khả năng định địa chỉ 64KB bộ nhớ chương trình và 64KB bộ nhớ dữ liệu.

Bảng 3.1 So sánh các vi điều khiển trong họ 8051.

Có ROM trong	Có EPROM trong	Không có ROM trong	ROM (bytes)	RAM (bytes)	Các cổng I/O 8 bit	Các mạch định thì/ Bộ đếm 16 bit	UART
80C51	87C51	80C31	4K	128	4	2	có
80C52	87C52	80C32	8K	256	4	3	có

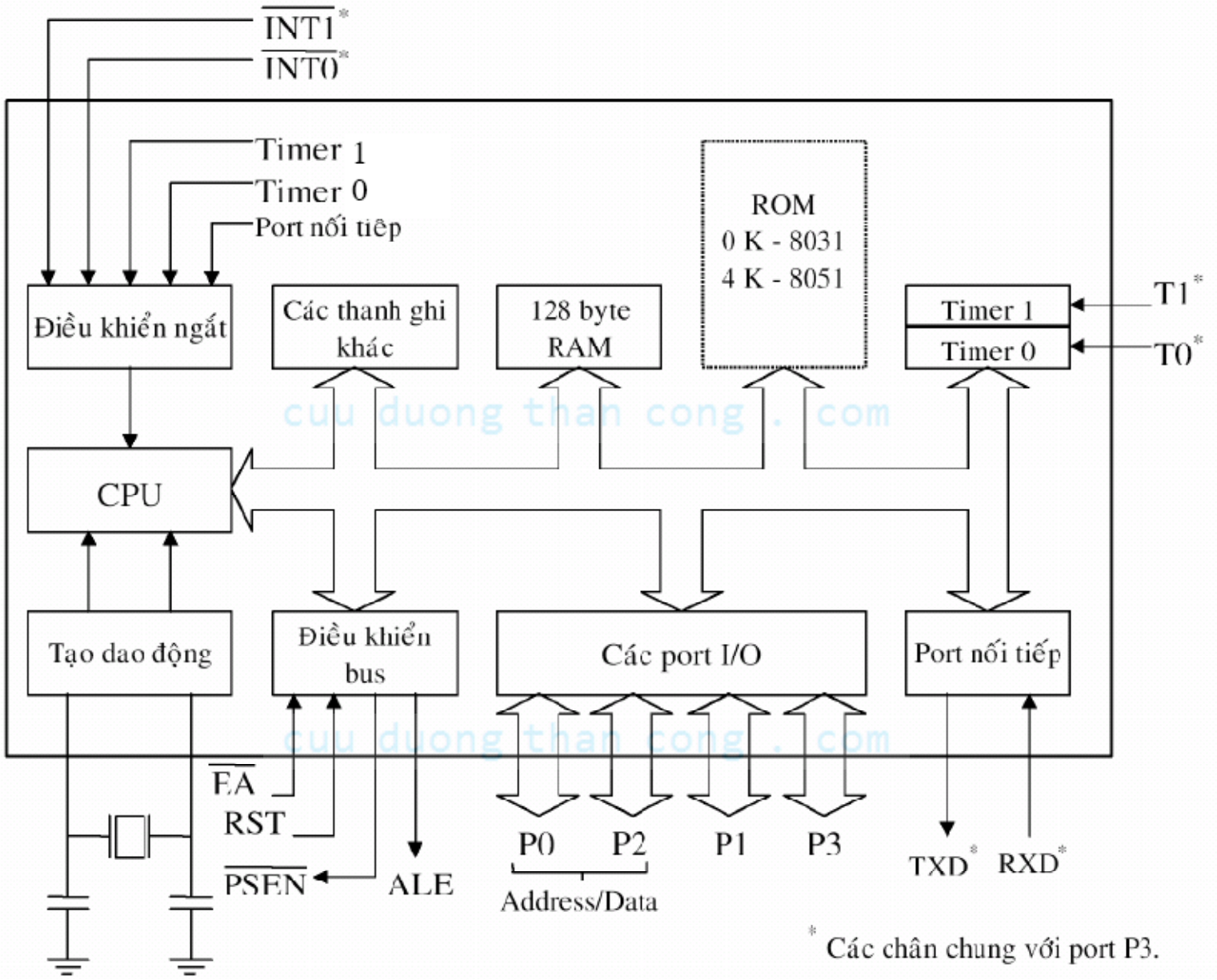
Chú ý: – Loạt 80C3X không có ROM/EPROM trong chip.
– Loạt 80C5X có từ 2KB đến 8KB ROM/EPROM trong chip.
– Loạt 89XX có bộ nhớ chương trình bên trong là "Flash EPROM".
– Loạt 80CX1 có 128 byte RAM nội.
– Loạt 80CX2 có 256 byte RAM nội.
– Về công nghệ chế tạo: loạt 8XXXX với công nghệ NMOS, loạt 8XCXX với công nghệ CMOS.

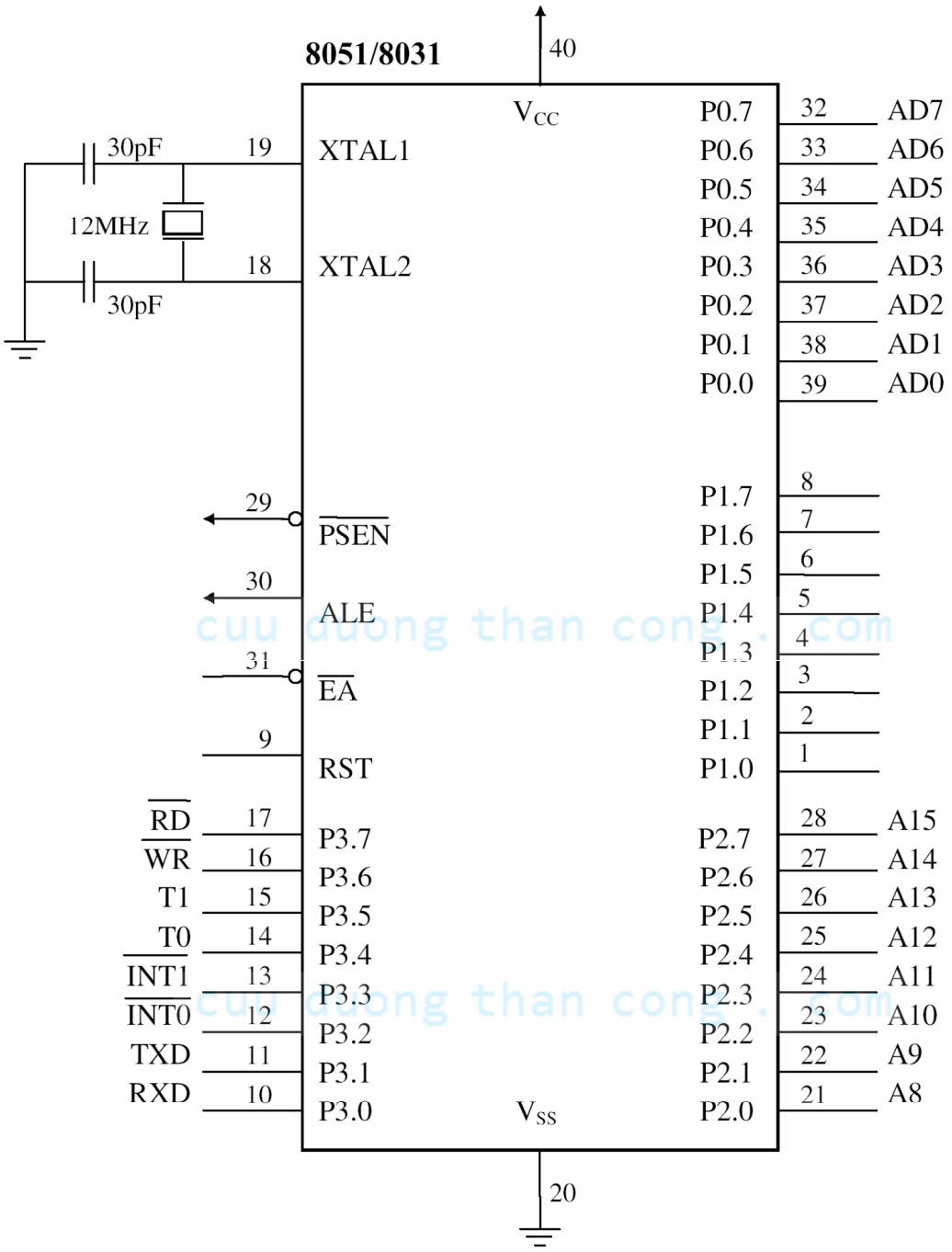
Comparison of MCS-51 ICs

Part Number	On-Chip Code Memory	On-Chip Data Memory	Timers
8051	4K ROM	128 bytes	2
8031	0K	128 bytes	2
8751	4K EPROM	128 bytes	2
8052	8K ROM	256 bytes	3
8032	0K	256 bytes	3
8752	8K EPROM	256 bytes	3

3.2 Kiến trúc phần cứng 8051

Sơ đồ khối 8051/8031





Hình 3.2 Sơ đồ chân 8051

Ý nghĩa các chân trên MCU 8051

- **Port 0 (Cổng 0)**

  Port 0 là một port hai chức năng trên các chân 32–39. Trong các thiết kế cỡ nhỏ (không dùng bộ nhớ mở rộng) nó có chức năng như các đường I/O. Đối với các thiết kế lớn với bộ nhớ mở rộng, nó được dồn kênh giữa bus địa chỉ và bus dữ liệu.

- **Port 1 (Cổng 1)**

  Port 1 là cổng dành riêng cho nhập/xuất trên các chân 1–8. Các chân được ký hiệu P1.0, P1.1, P1.2, ... có thể dùng cho giao tiếp với các thiết bị ngoài nếu cần. Port 1 không có chức năng khác, vì vậy chúng chỉ được dùng cho giao tiếp với các thiết bị ngoài.

- **Port 2 (Cổng 2)**

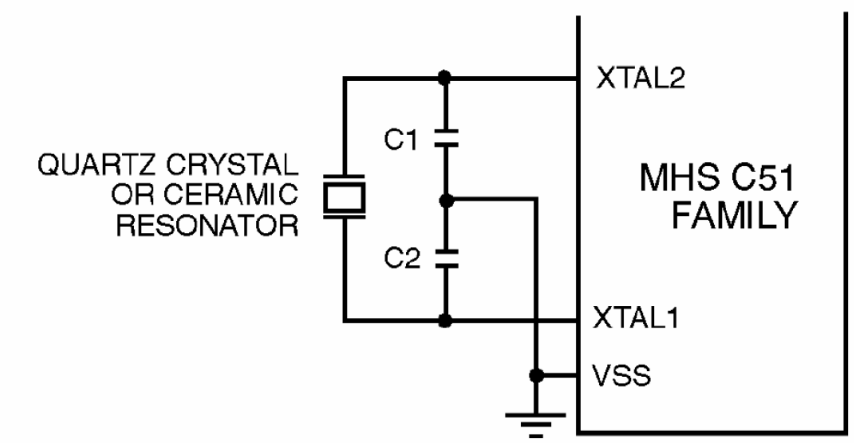
  Port 2 là một cổng công dụng kép trên các chân 21–28 được dùng như các đường xuất nhập hoặc là byte cao của bus địa chỉ đối với các thiết kế dùng bộ nhớ mở rộng.

- **Port 3 (Cổng 3)**

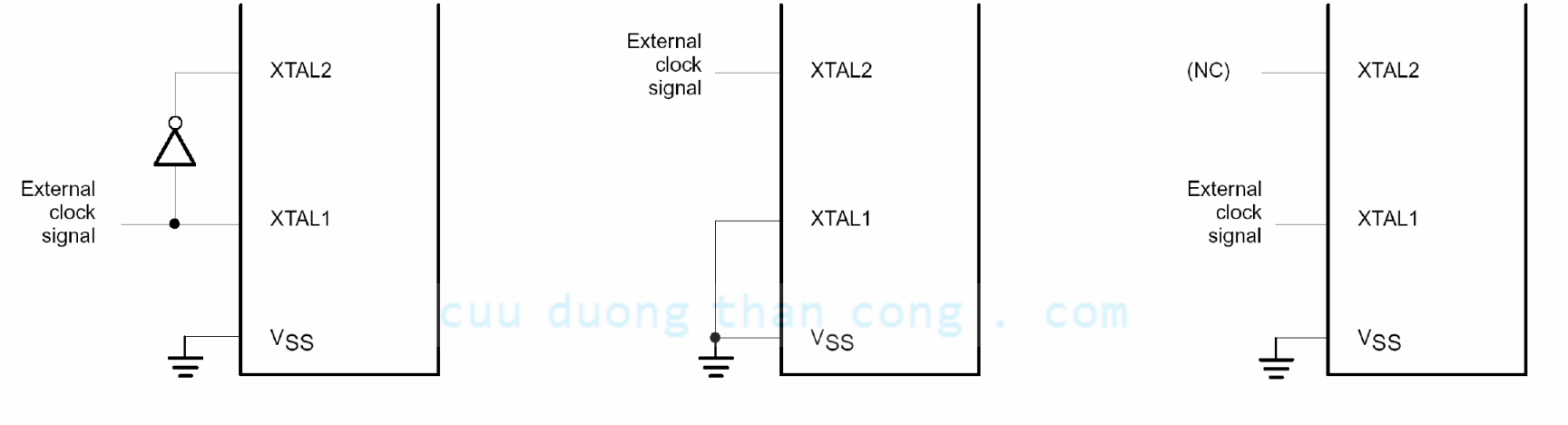
  Port 3 cũng là một cổng công dụng kép trên các chân 10–17. Các chân của port này có nhiều chức năng, các công dụng chuyển đổi có liên hệ với các đặc tính đặc biệt của 8051/8031

Các chức năng chuyển đổi ở Port 3

Bit	Tên	Địa chỉ bit	Chức năng chuyển đổi
P3.0	RXD	B0H	Dữ liệu nhận cho port nối tiếp.
P3.1	TXD	B1H	Dữ liệu phát cho port nối tiếp.
P3.2	$\overline{\text{INT0}}$	B2H	Ngắt 0 bên ngoài.
P3.3	$\overline{\text{INT1}}$	B3H	Ngắt 1 bên ngoài.
P3.4	T0	B4H	Ngõ vào của timer/counter 0.
P3.5	T1	B5H	Ngõ vào của timer/counter 1.
P3.6	$\overline{\text{WR}}$	B6H	Xung ghi bộ nhớ dữ liệu ngoài.
P3.7	$\overline{\text{RD}}$	B7H	Xung đọc bộ nhớ dữ liệu ngoài.



Hình 3.3 Dao động trên chip (C1=C2= 30pF ± 10pF với thạch anh [crystal] và C1=C2= 30pF ± 10pF với bộ cộng hưởng gốm [ceramic resonator])

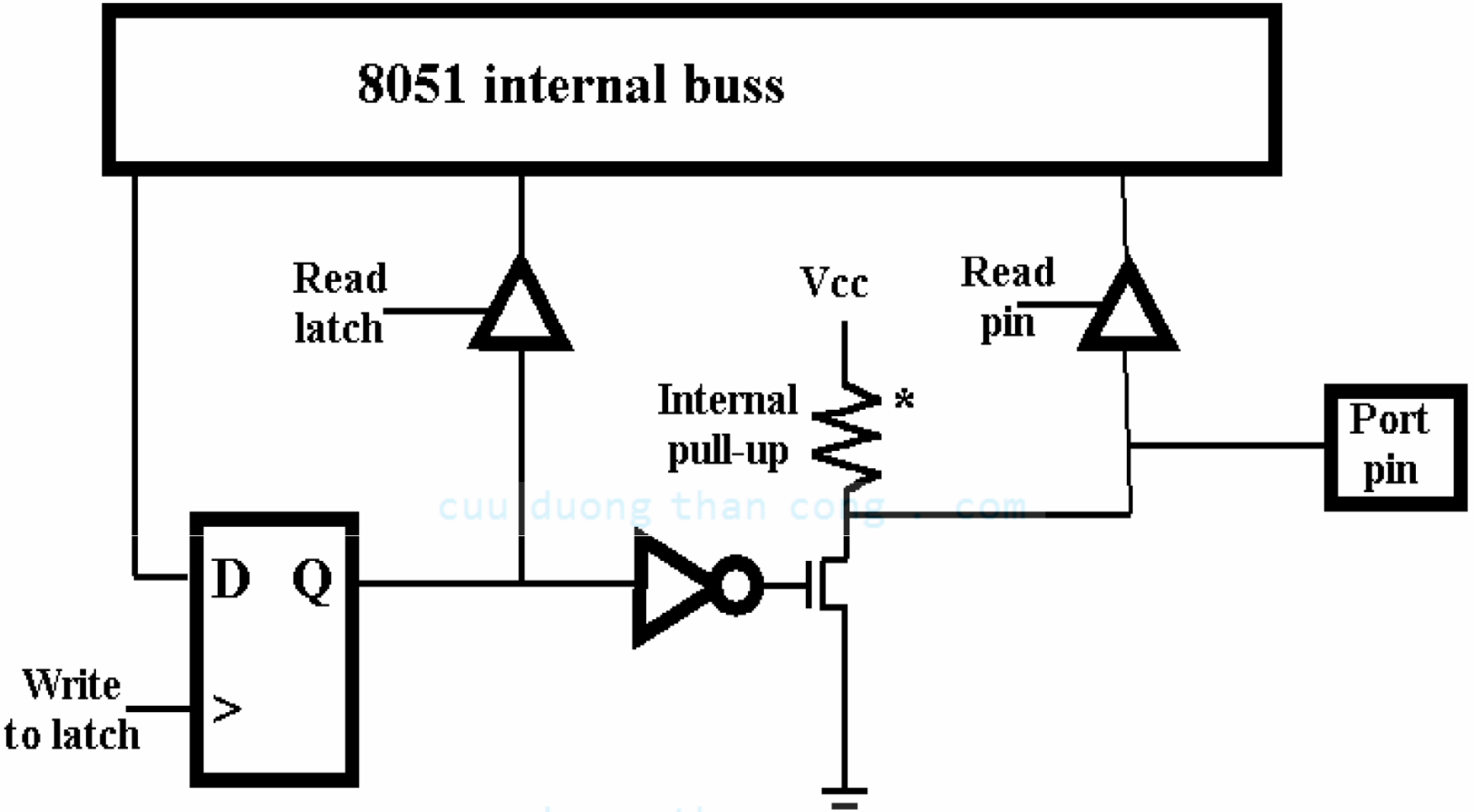


a) 8051 loại NMOS hoặc CMOS. b) chỉ cho loại NMOS. c) chỉ cho loại CMOS.

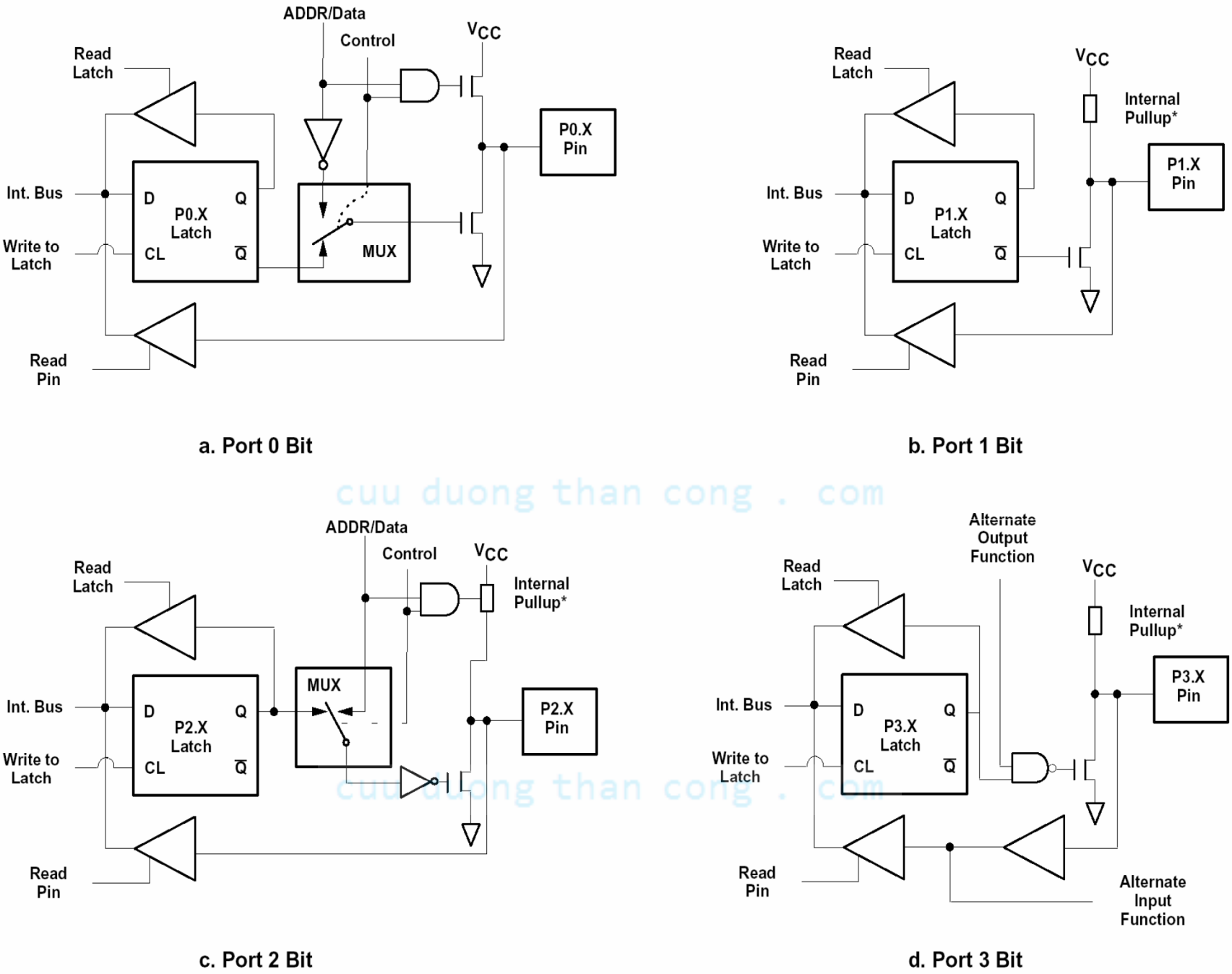
Hình 3.4 Sử dụng xung nhịp từ bên ngoài.

Cấu trúc cổng I/O

- Khả năng lái là 4 tải TTL loại LS (Low Power Schottky) với các cổng P1, P2, và P3; và 8 tải TTL loại LS với cổng P0.
- Chú ý là điện trở kéo lên bên trong không có trong Port 0 (ngoại trừ lúc làm việc như bus dữ liệu / địa chỉ bên ngoài). Điện trở kéo lên có thể được sử dụng với P0 tùy theo đặc tính vào của thiết bị mà nó lái.

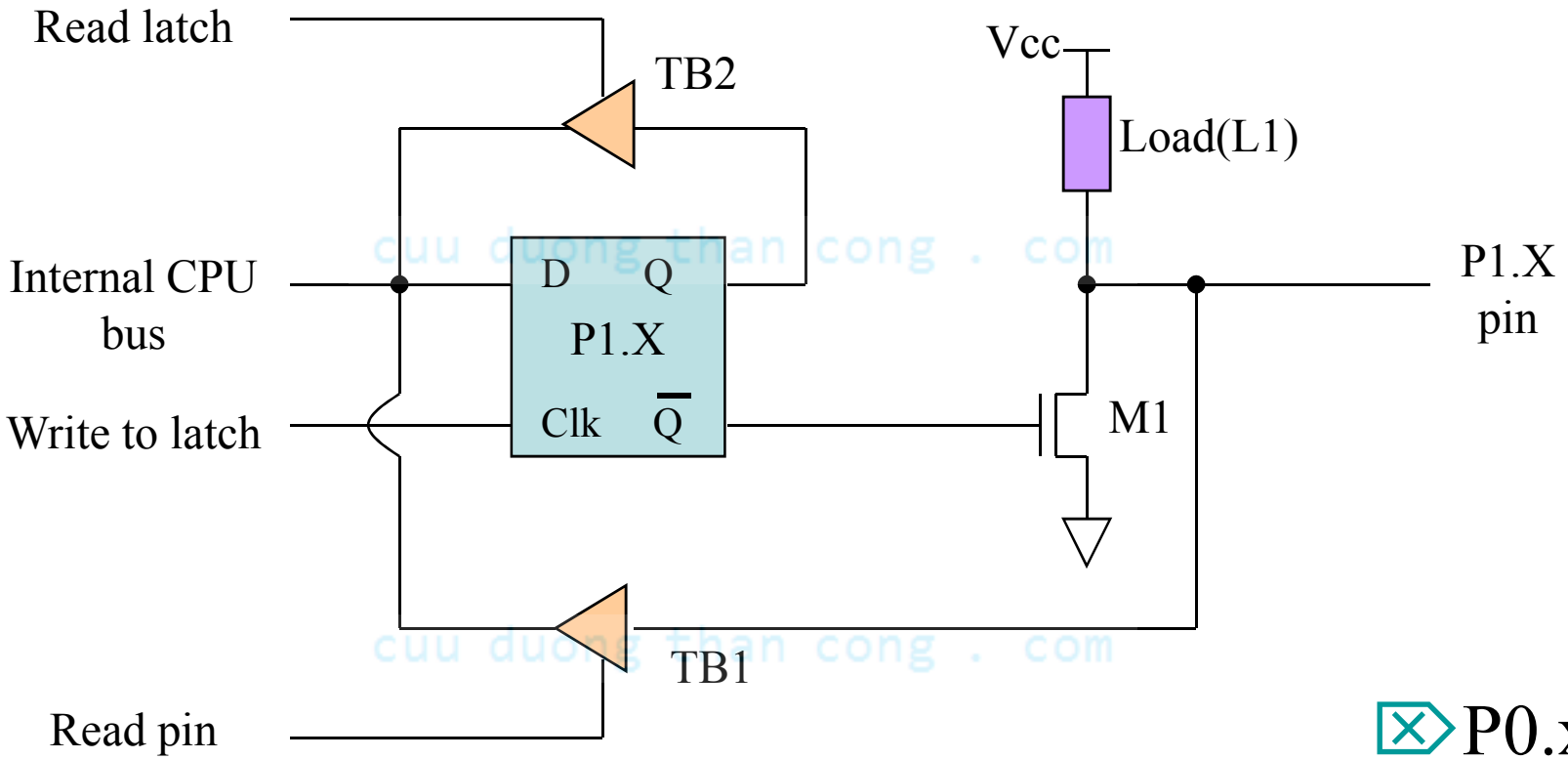


Hình 3.5 Mạch tóm tắt các cổng I/O.

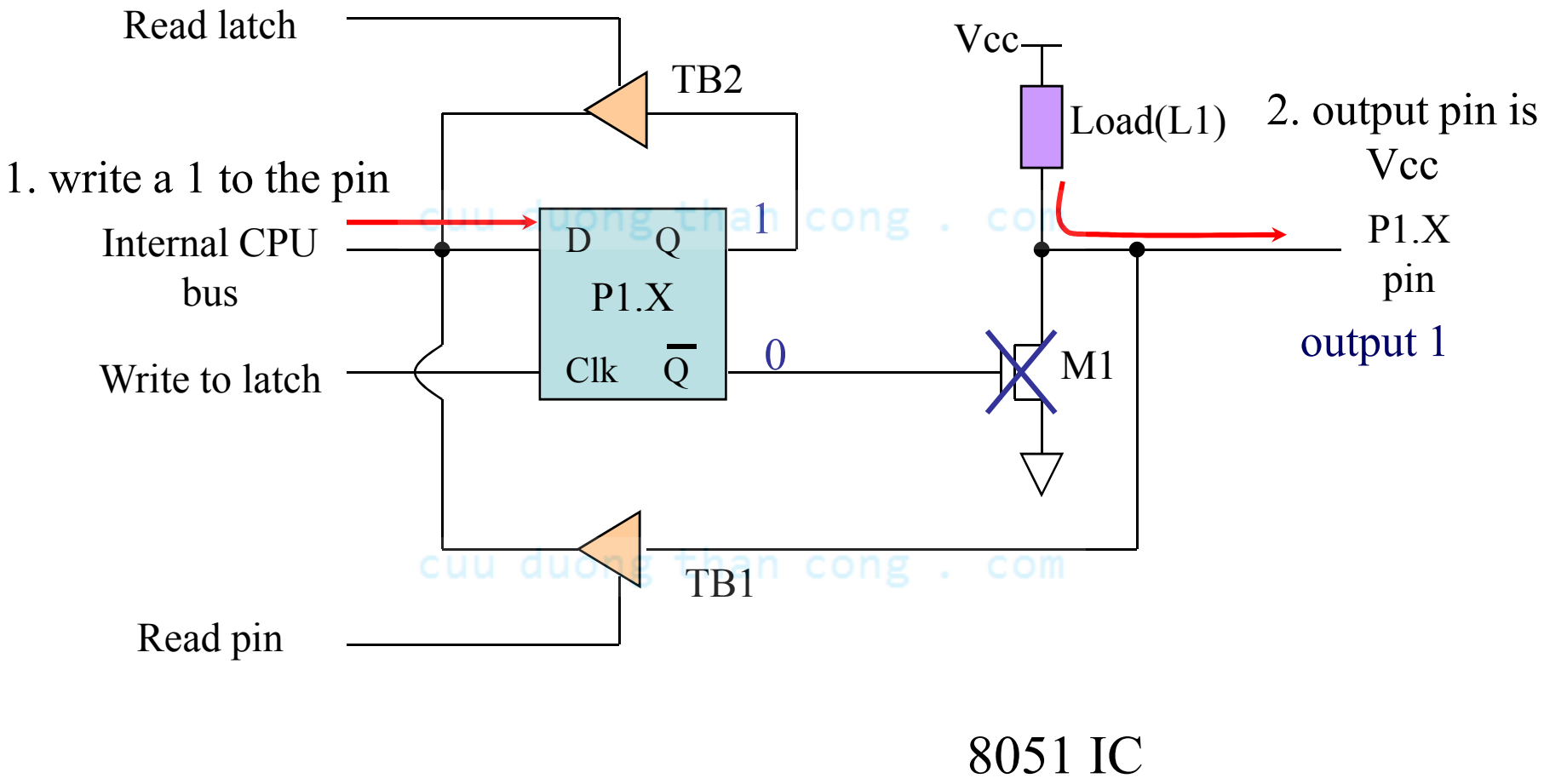


Hình 3.6 Các mạch chốt cổng và các bộ đệm I/O.

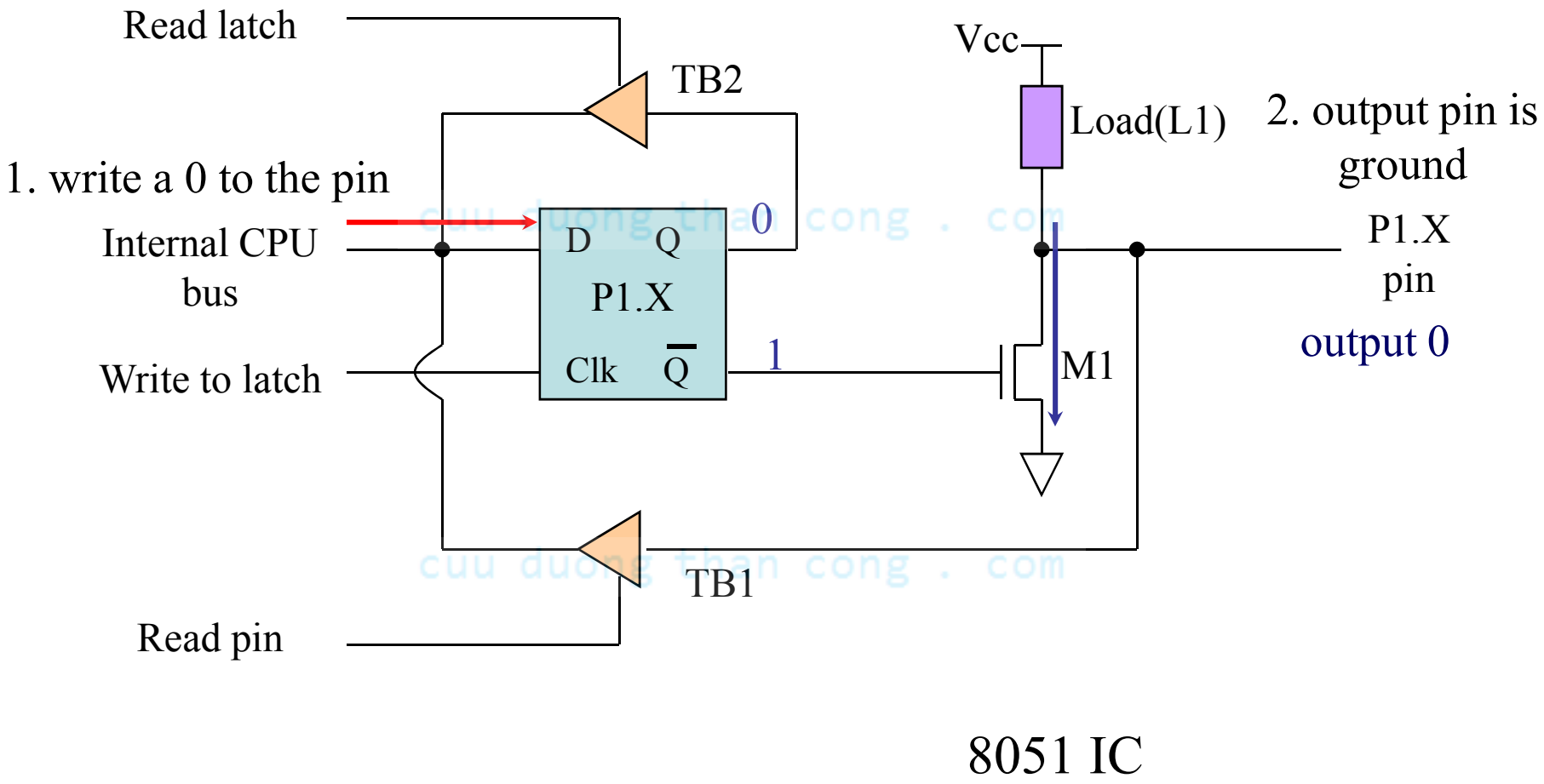
A Pin of Port 1



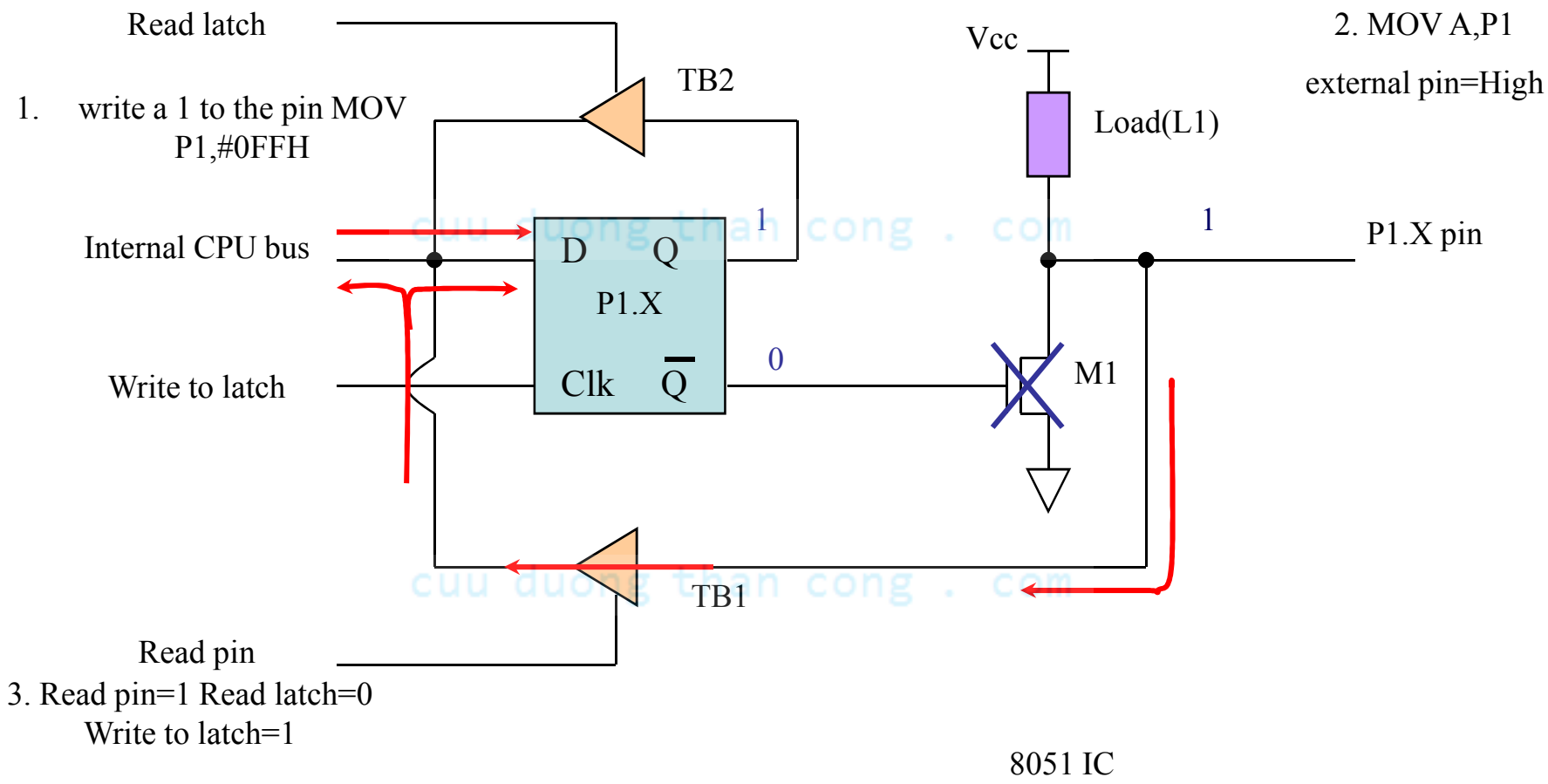
Writing “1” to Output Pin P1.X



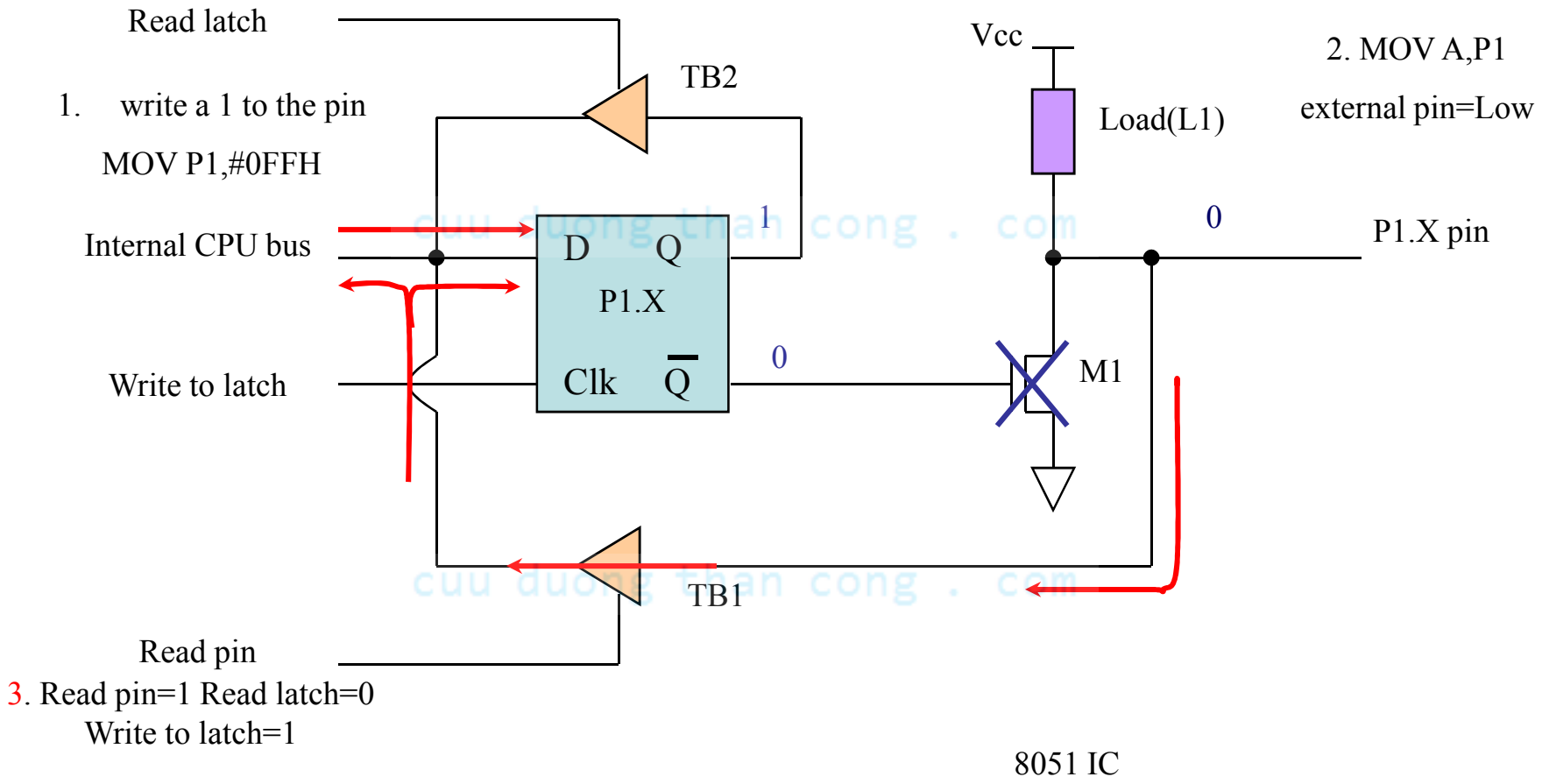
Writing “0” to Output Pin P1.X



Reading “High” at Input Pin



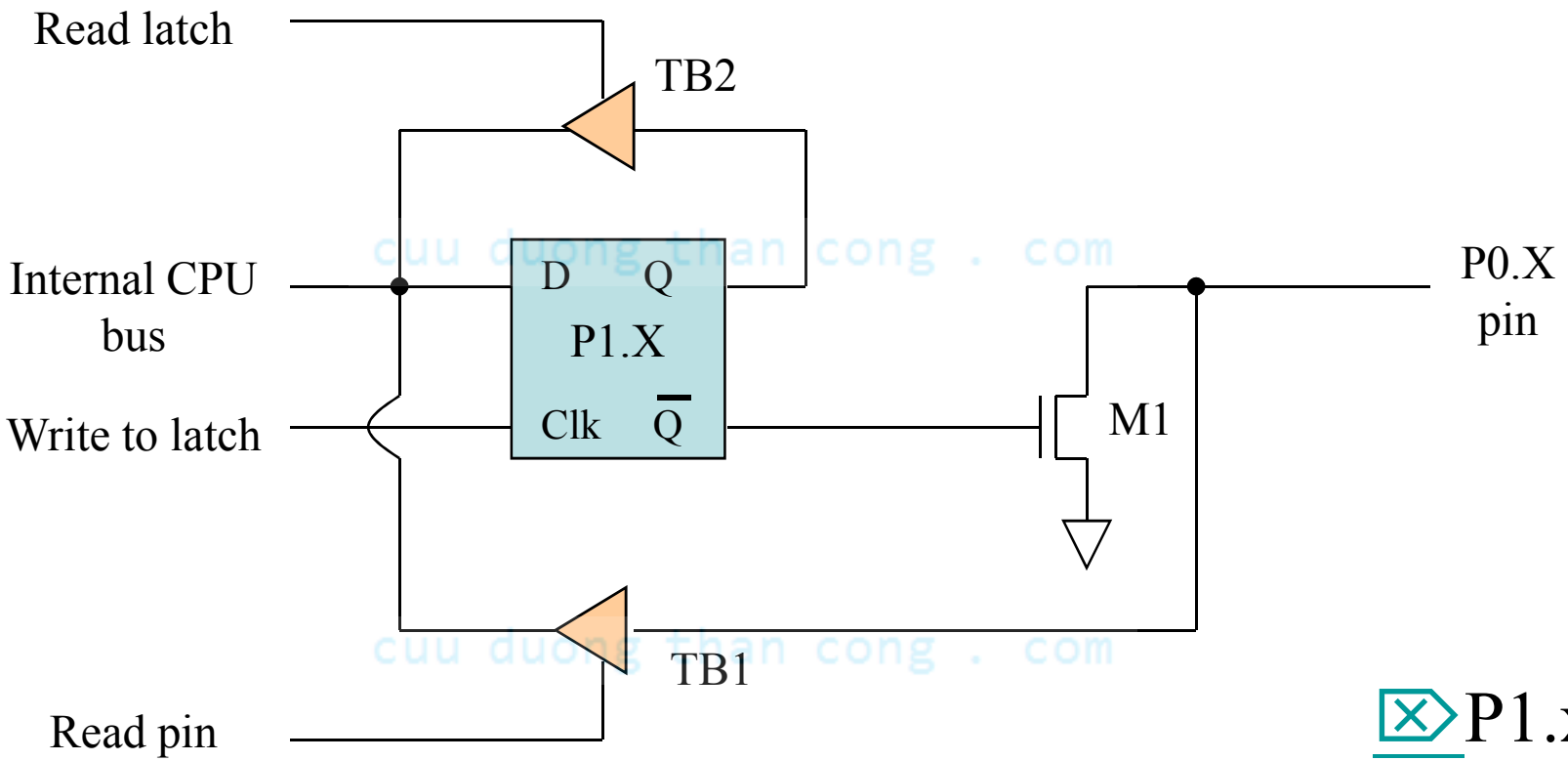
Reading “Low” at Input Pin



Other Pins

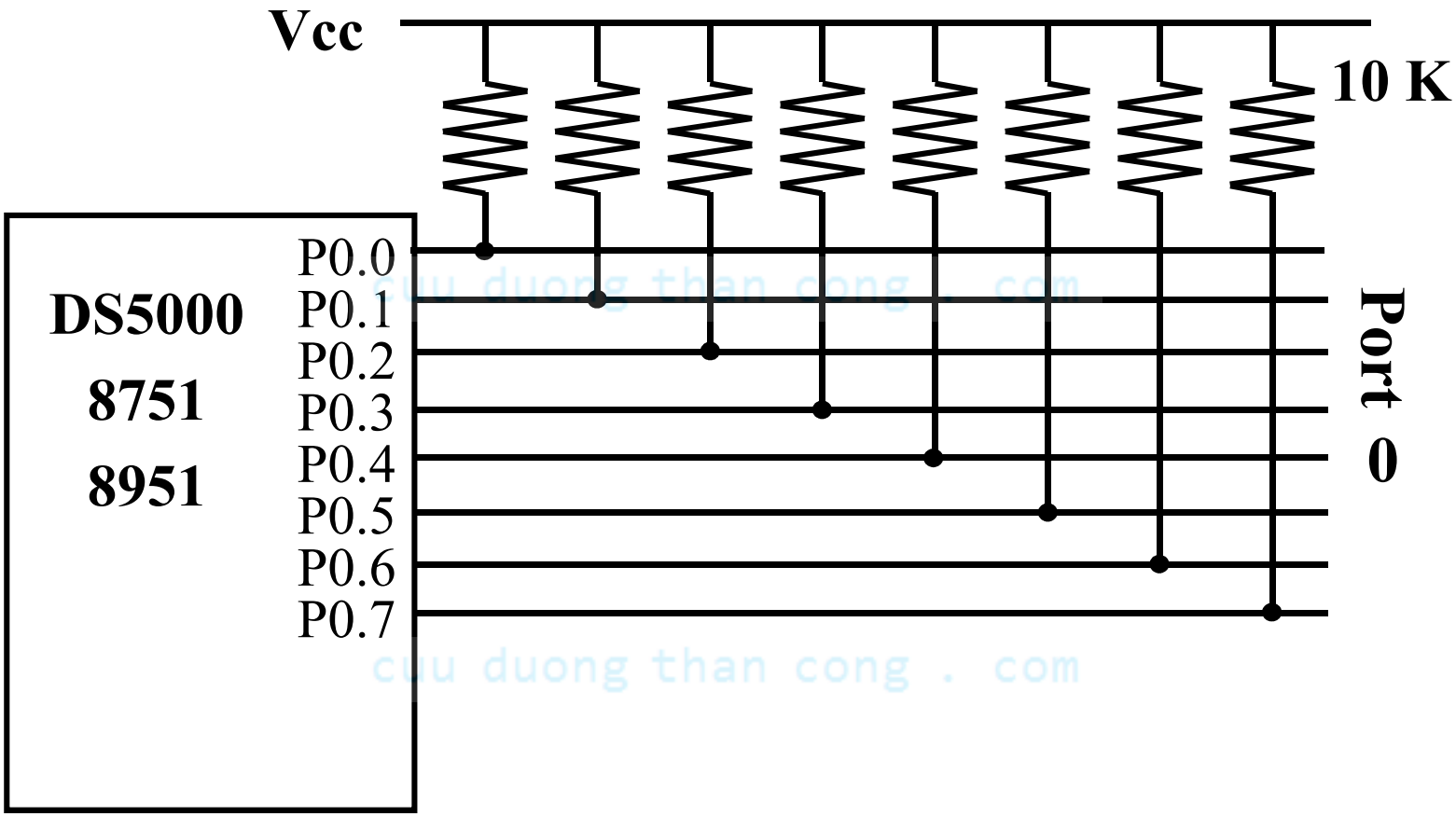
- P1, P2, and P3 have internal pull-up resisters.
  - P1, P2, and P3 are not open drain.
- P0 has no internal pull-up resistors and does not connects to Vcc inside the 8051.
  - P0 is open drain.
  - Compare the figures of P1.X and P0.X. 
- However, for a programmer, it is the same to program P0, P1,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

A Pin of Port 0



8051 IC

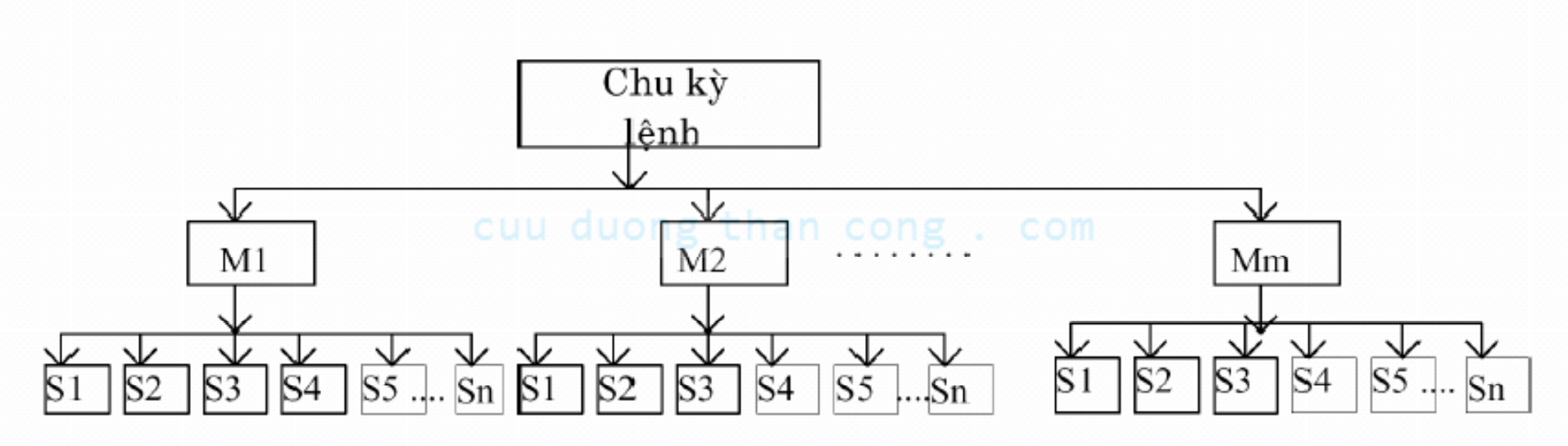
Port 0 with Pull-Up Resistors



Port 3 Alternate Functions

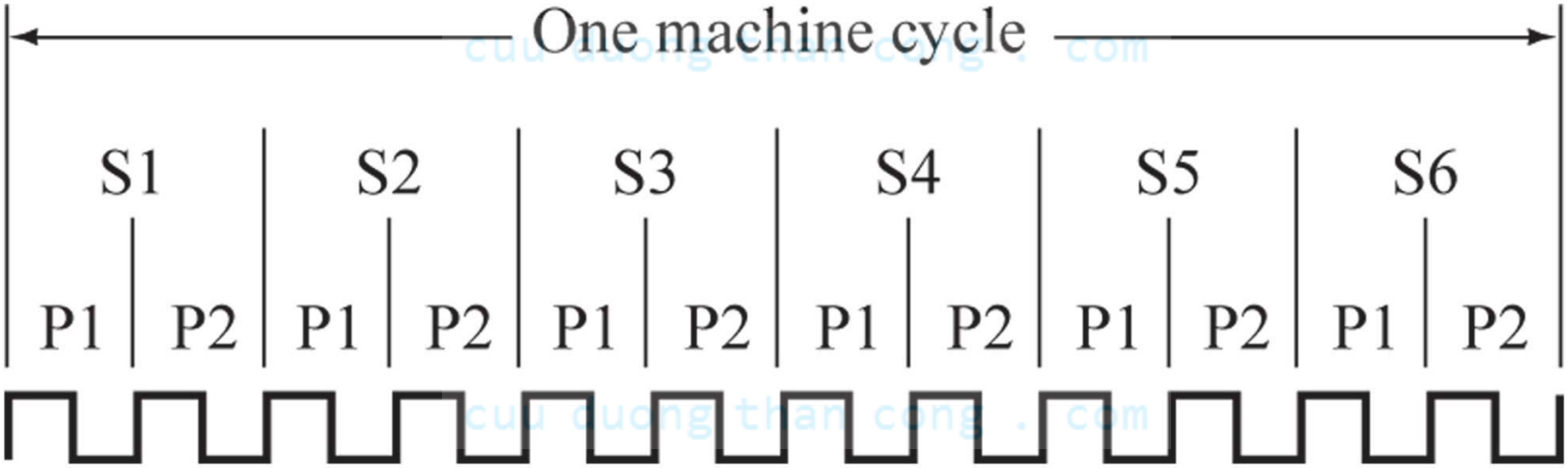
P3 Bit	Function	Pin
P3.0	RxD	10
P3.1	TxD	11
P3.2	$\overline{\text{INT0}}$	12
P3.3	$\overline{\text{INT1}}$	13
P3.4	T0	14
P3.5	T1	15
P3.6	$\overline{\text{WR}}$	16
P3.7	$\overline{\text{RD}}$	17

Chu kỳ lệnh, chu kỳ máy và trạng thái

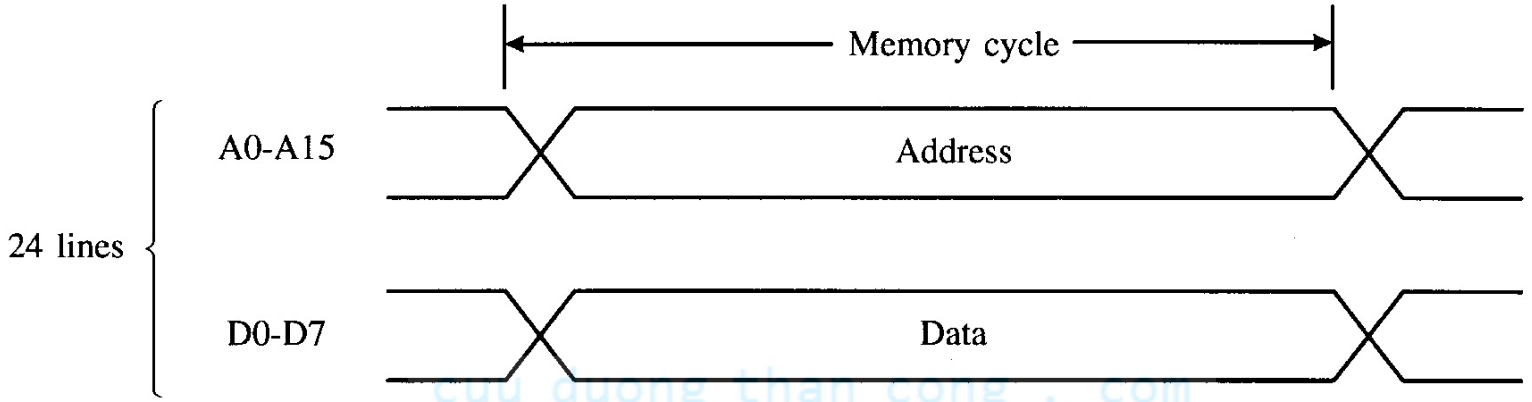


Hình 3.7 Chia một chu kỳ lệnh thành các chu kỳ máy và các trạng thái. (Các chu kỳ máy Mi và tạng thái Si được vẽ trong ... là nhiệm ý).

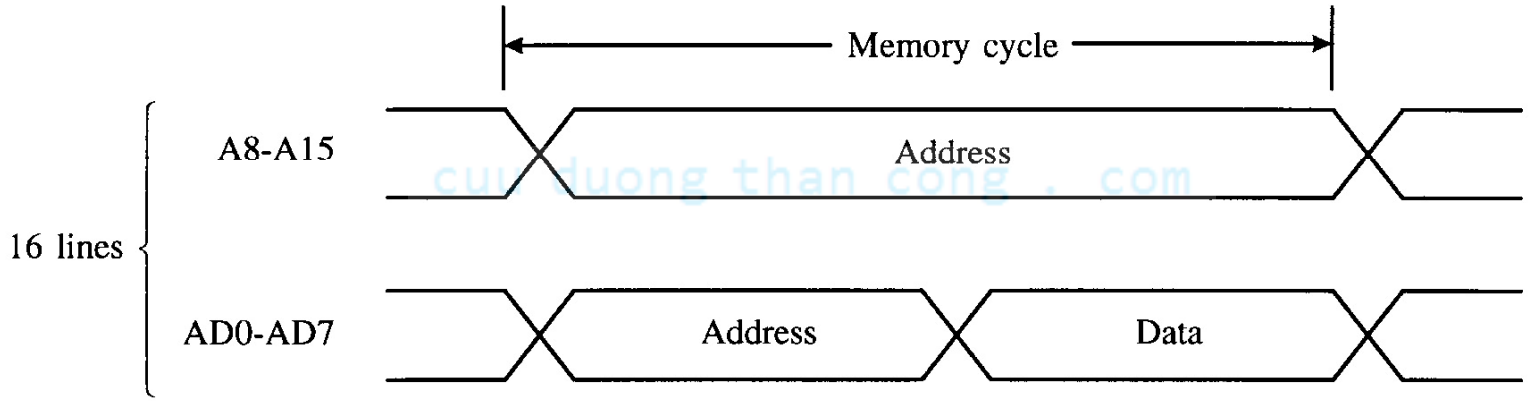
12MHz internal clock
6 machine cycles



Dồn kênh bus địa chỉ (byte thấp) và bus dữ liệu

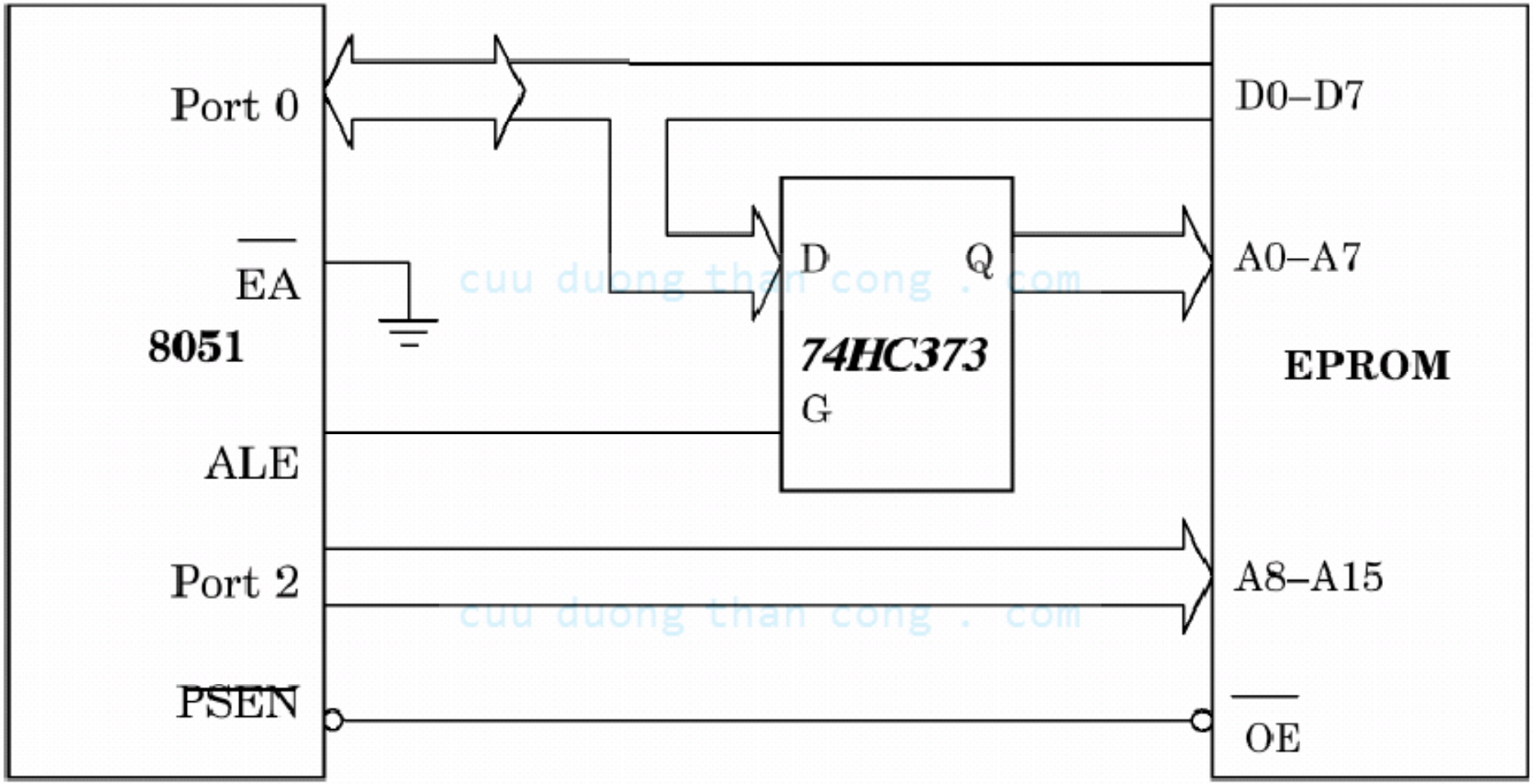


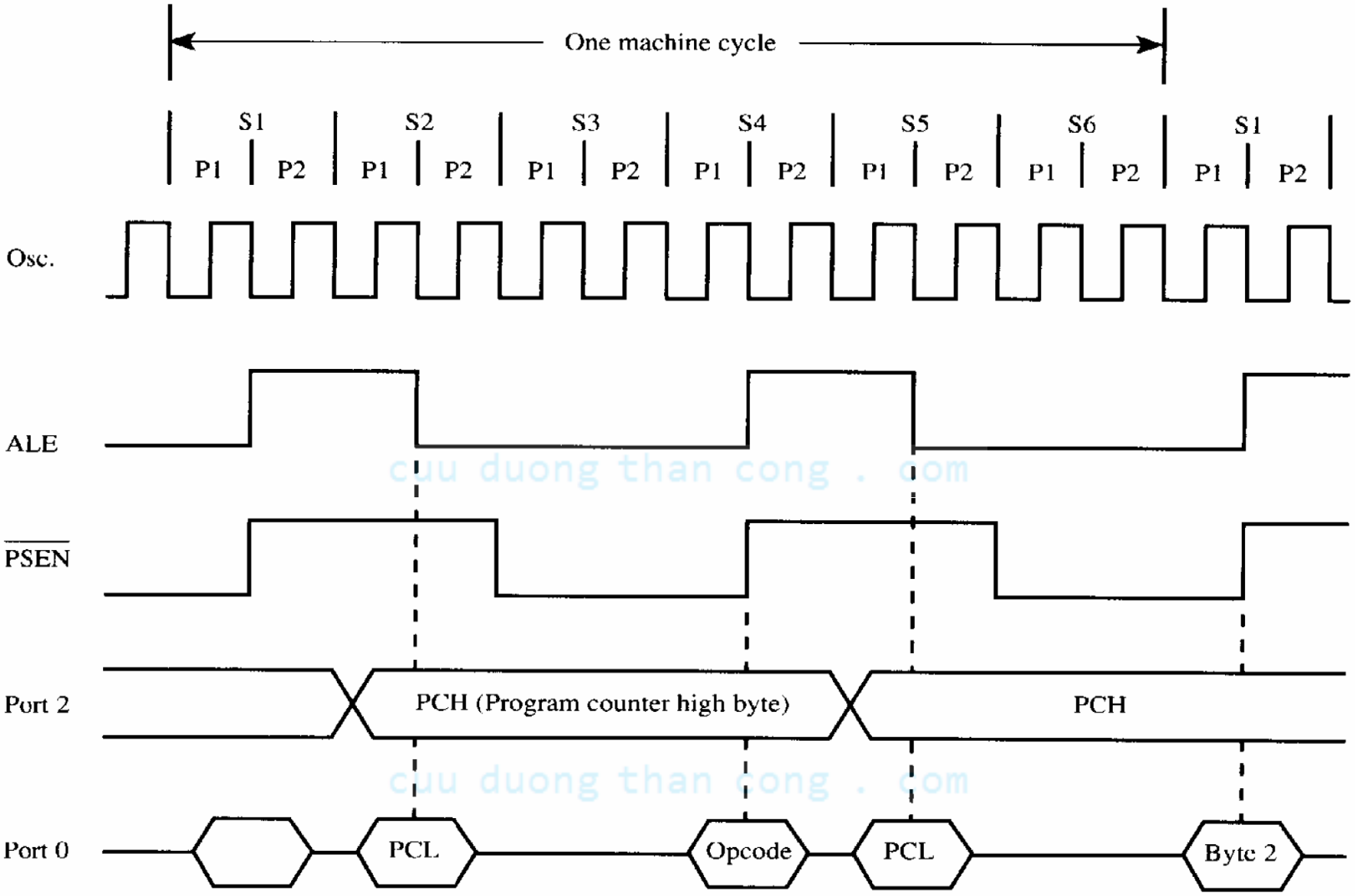
(a) Nonmultiplexed (24 lines)



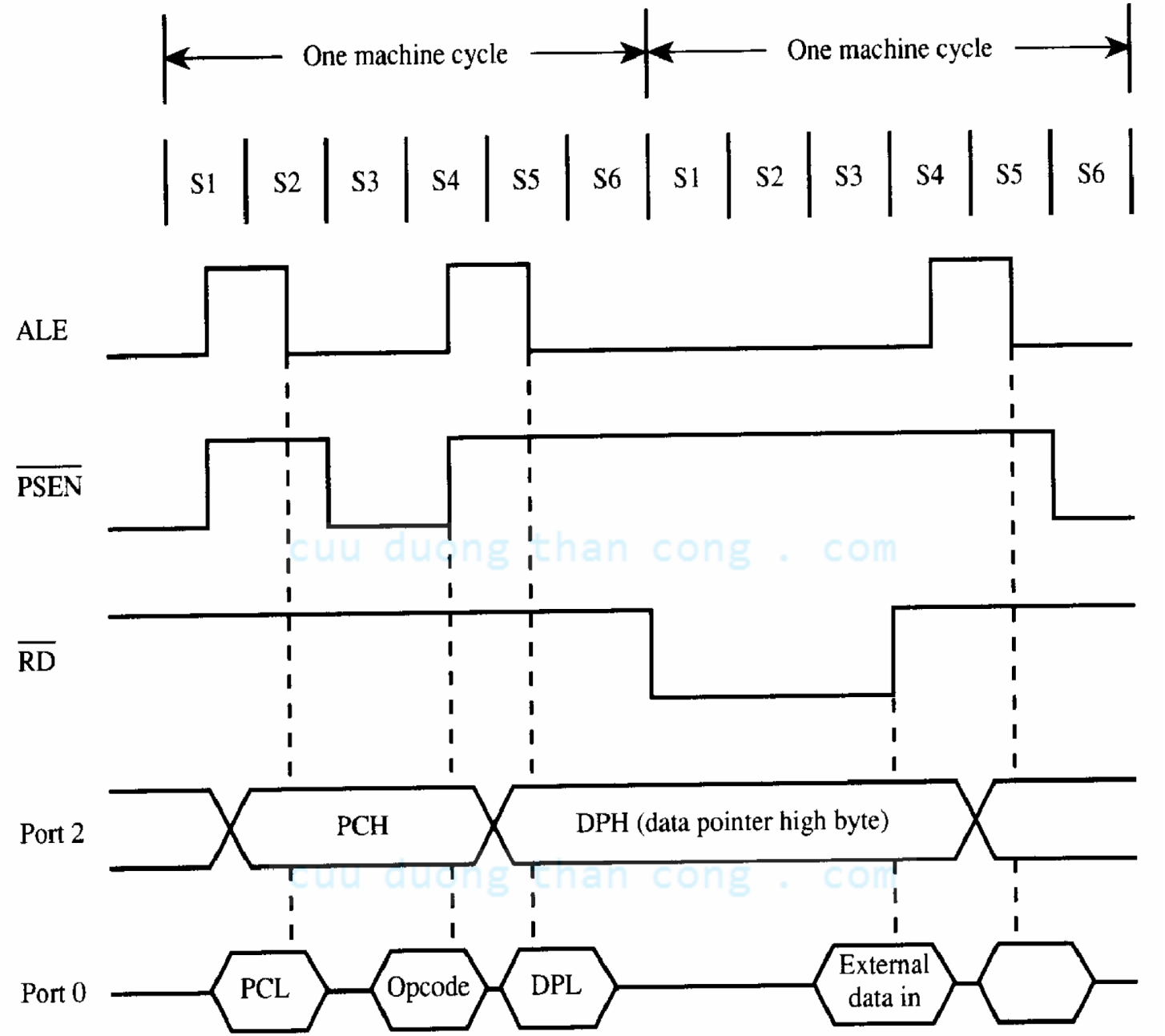
(b) Multiplexed (16 lines)

Truy cập bộ nhớ chương trình bên ngoài

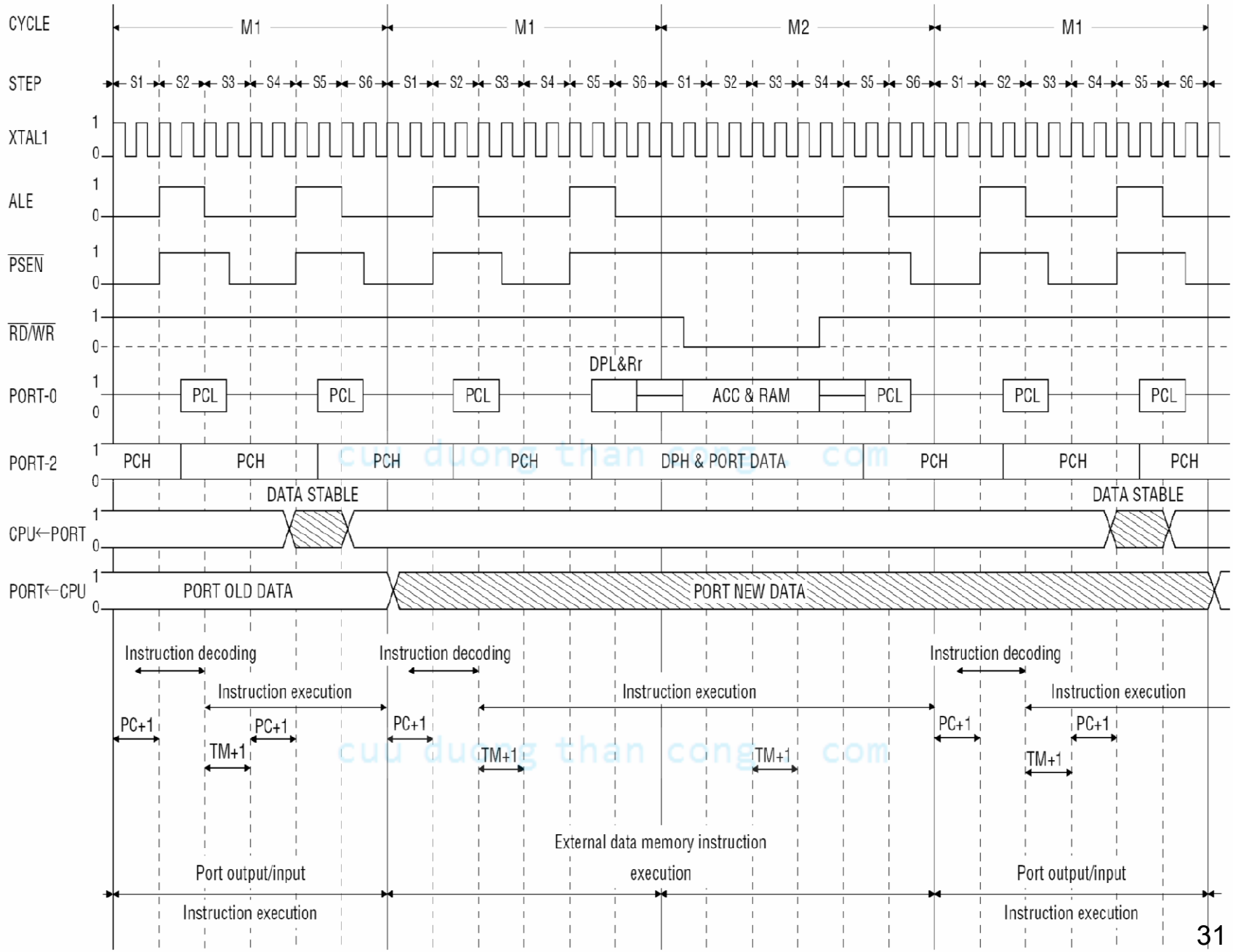




Hình 3.8 Giản đồ định thì đọc bộ nhớ chương trình bên ngoài.

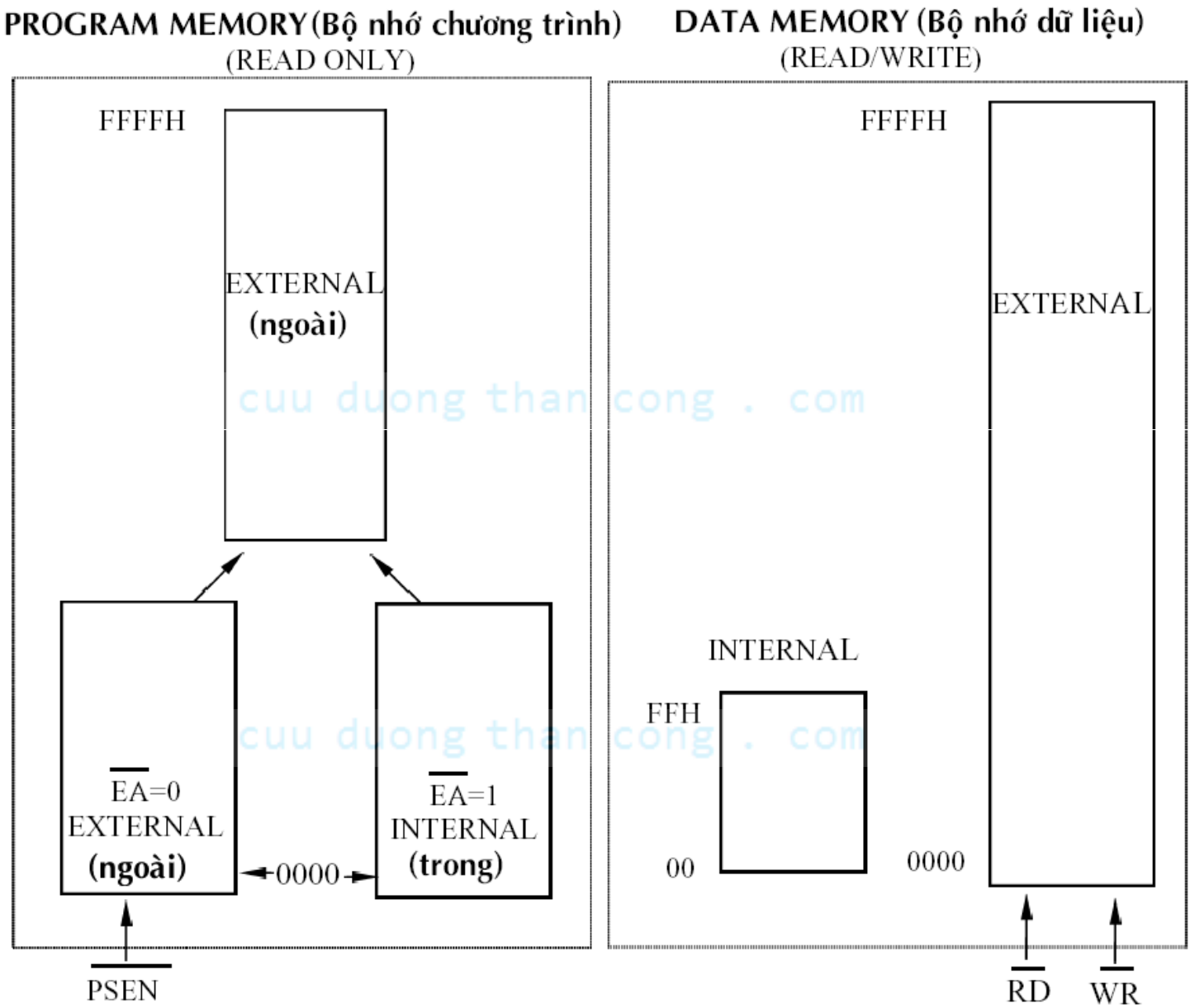


Hình 3.9 Định thì cho lệnh MOVX.



Hình 3.10 Tóm tắt các chu kỳ máy trong họ 8051.

Cấu trúc bộ nhớ 8051



Tóm tắt bộ nhớ dữ liệu trên chip

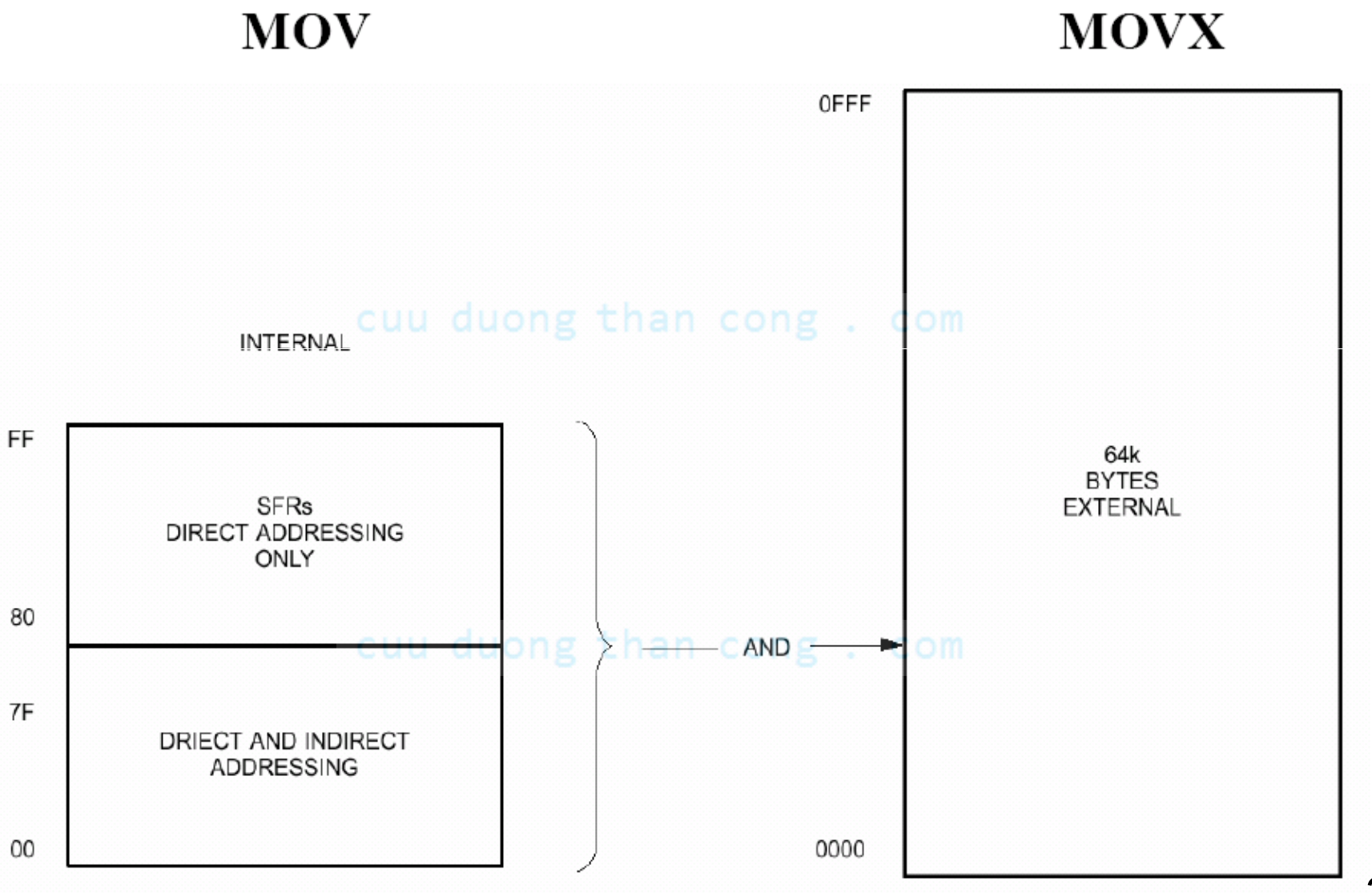
Địa chỉ byte	Địa chỉ bit							
7F	RAM đa dụng							
30								
2F	7F	7E	7D	7C	7B	7A	79	78
2E	77	76	75	74	73	72	71	70
2D	6F	6E	6D	6C	6B	6A	69	68
2C	67	66	65	64	63	62	61	60
2B	5F	5E	5D	5C	5B	5A	59	58
2A	57	56	55	54	53	52	51	50
29	4F	4E	4D	4C	4B	4A	49	48
28	47	46	45	44	43	42	41	40
27	3F	3E	3D	3C	3B	3A	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	07	00
1F	Bank 3							
18								
17	Bank 2							
10								
0F	Bank 1							
08								
07	Bank thanh ghi 0 (mặc định cho R0-R7)							
00								

RAM

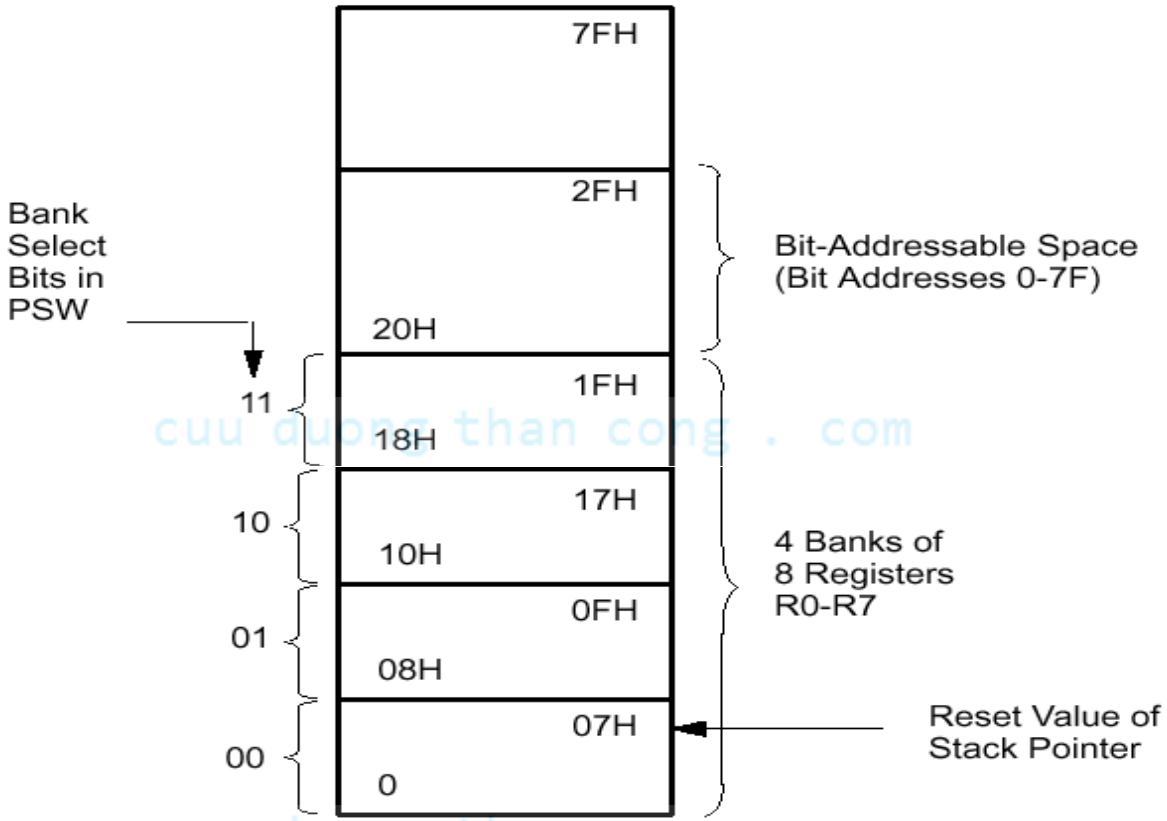
Địa chỉ byte	Địa chỉ bit								
FF									
F0	F7	F6	F5	F4	F3	F2	F1	F0	B
E0	E7	E6	E5	E4	E3	E2	E1	E0	ACC
D0	D7	D6	D5	D4	D3	D2	–	D0	PSW
B8	–	–	–	BC	BB	BA	B9	B8	IP
B0	B7	B6	B5	B4	B3	B2	B1	B0	P3
A8	AF	–	–	AC	AB	AA	A9	A8	IE
A0	A7	A6	A5	A4	A3	A2	A1	A0	P2
99	không được địa chỉ hóa bit								SBUF
98	9F	9E	9D	9C	9B	9A	99	98	SCON
90	97	96	95	94	93	92	91	90	P1
8D	không được địa chỉ hóa bit								TH1
8C	không được địa chỉ hóa bit								TH0
8B	không được địa chỉ hóa bit								TL1
8A	không được địa chỉ hóa bit								TL0
89	không được địa chỉ hóa bit								TMOD
88	8F	8E	8D	8C	8B	8A	89	88	TCON
87	không được địa chỉ hóa bit								PCON
83	không được địa chỉ hóa bit								DPH
82	không được địa chỉ hóa bit								DPL
81	không được địa chỉ hóa bit								SP
80	87	86	85	84	83	82	81	80	P0

CÁC THANH GHI CHỨC NĂNG ĐẶC BIỆT

Bộ nhớ dữ liệu 8051



Lower 128 Bytes of Internal RAM



20H-2FH: 128 Bit-addressable bits occupying bit address 00H-7FH.

30H-7FH: General purpose RAM (can be accessed through direct or indirect addressing)

Upper 128 Bytes of Internal RAM

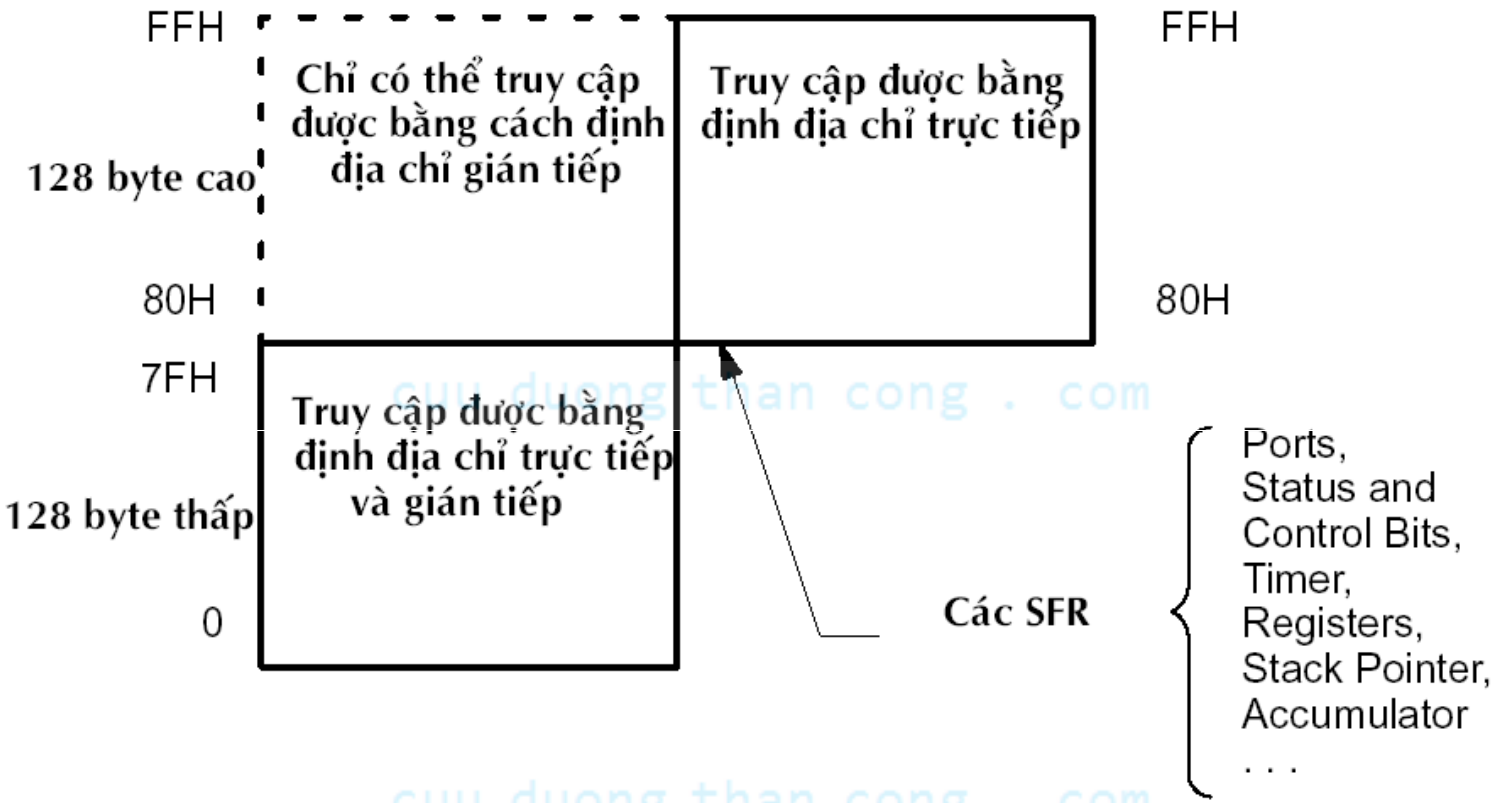
No bit-addressable spaces

Available as stack space in devices with 256 bytes RAM

Not implemented in 8051

Available only in 8052. Can be accessed by indirect addressing only (via @R0 or @R1). Can be used as stack area by setting SP to FFH.

Vùng nhớ 8032/8052



MOV Ri, #5FH (I = 0 or 1)
MOV A, @Ri

MOV A, 5FH
MOV A, FFH

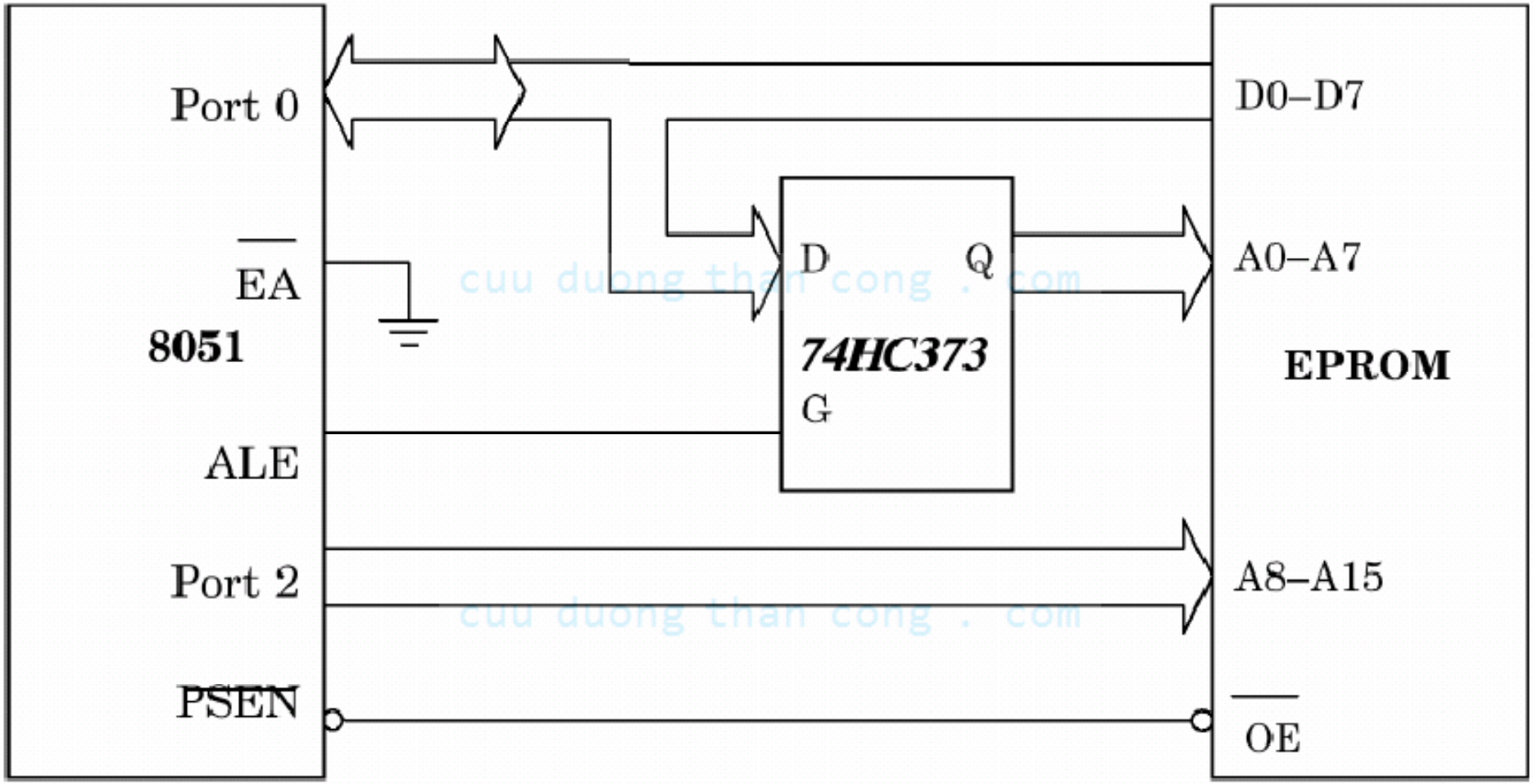
Tóm tắt thanh ghi PSW

Bit	Ký hiệu	Địa chỉ bit	Mô tả bit
PSW.7	CY	D7H	Cờ nhớ (Carry flag).
PSW.6	AC	D6H	Cờ nhớ phụ (Auxiliary carry flag).
PSW.5	F0	D5H	Cờ 0.
PSW.4	RS1	D4H	Chọn băng thanh ghi, bit 1.
PSW.3	RS0	D3H	Chọn băng thanh ghi, bit 0. 00 = bank 0; địa chỉ 00H–07H 01 = bank 1; địa chỉ 08H–0FH 10 = bank 2; địa chỉ 10H–17H 11 = bank 3; địa chỉ 18H–1FH
PSW.2	OV	D2H	Cờ báo tràn (Overflow flag).
PSW.1	–	D1H	Dự trữ.
PSW.0	P	D0H	Cờ kiểm tra chẵn (Even parity flag).

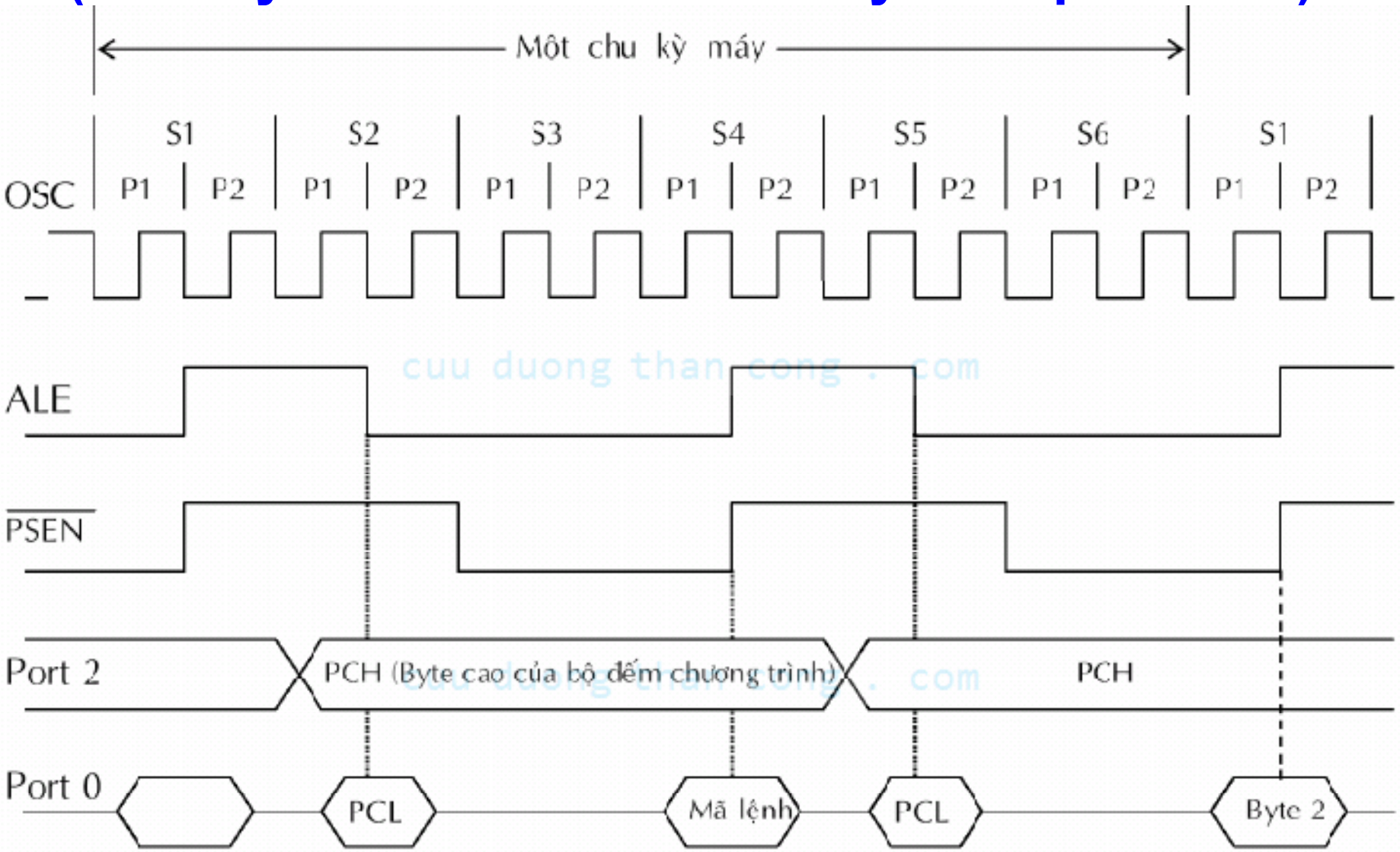
Tóm tắt thanh ghi PCON

Bit	Ký hiệu	Mô tả bit
7	SMOD	Bit tăng gấp đôi tốc độ baud; khi được đặt lên 1 thì tốc độ baud được tăng gấp đôi trong các chế độ cổng nối tiếp 1, 2, hoặc 3.
6	–	Không được định nghĩa.
5	–	Không được định nghĩa.
4	–	Không được định nghĩa.
3	GF1	Cờ đa dụng (General Purpose Flag), bit 1.
2	GF0	Cờ đa dụng, bit 0.
1	PD	Tắt nguồn (power down); được đặt lên 1 để kích hoạt chế độ tắt nguồn; chỉ thoát khi bị xóa về 0.
0	IDL	chế độ nghỉ; được đặt lên 1 để kích hoạt chế độ nghỉ; chỉ thoát khi có ngắt hoặt reset hệ thống.

Truy cập bộ nhớ chương trình bên ngoài

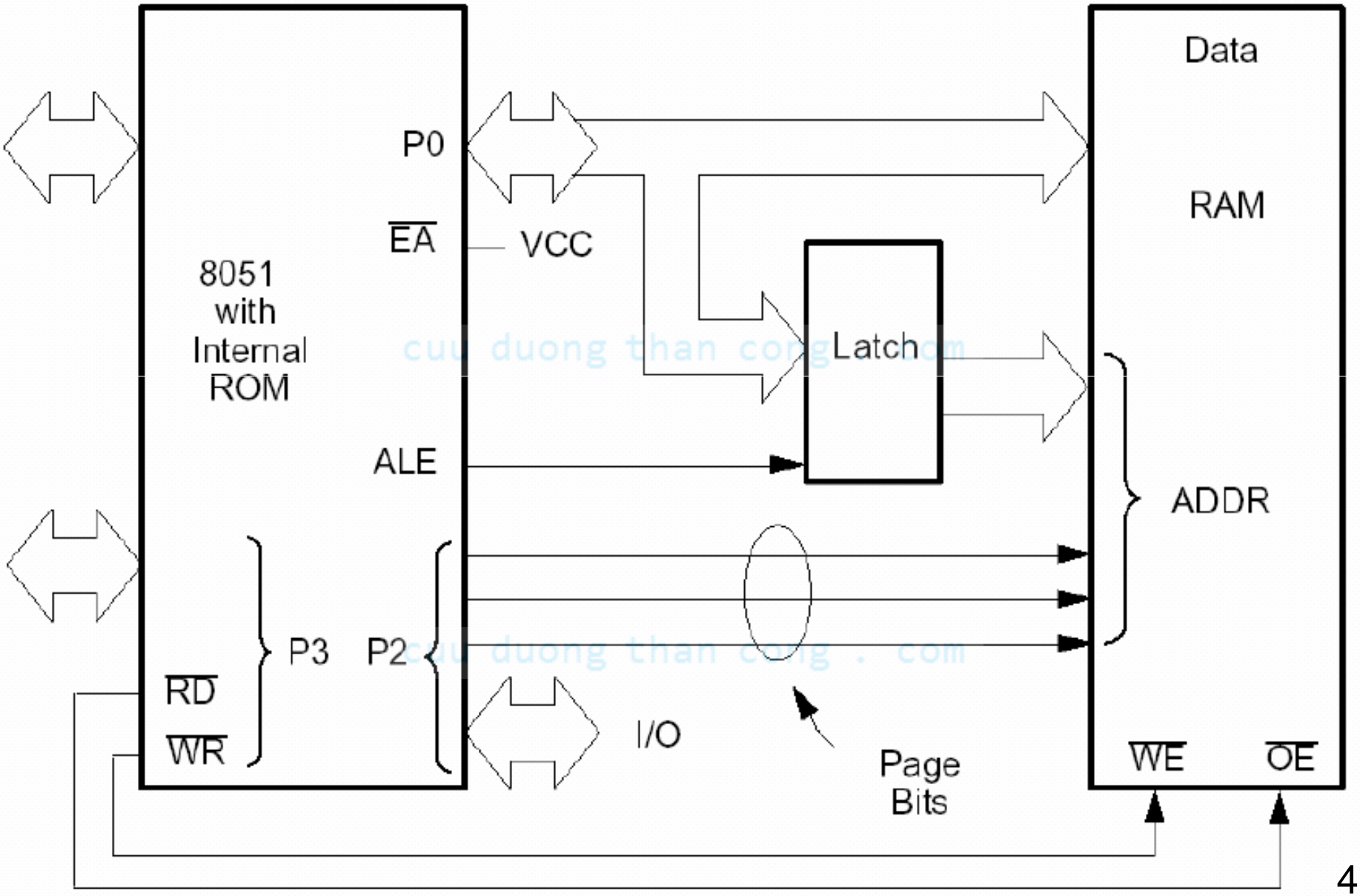


Định thì đọc bộ nhớ chương trình bên ngoài (PCH byte của PC và PCL là byte thấp của PC)

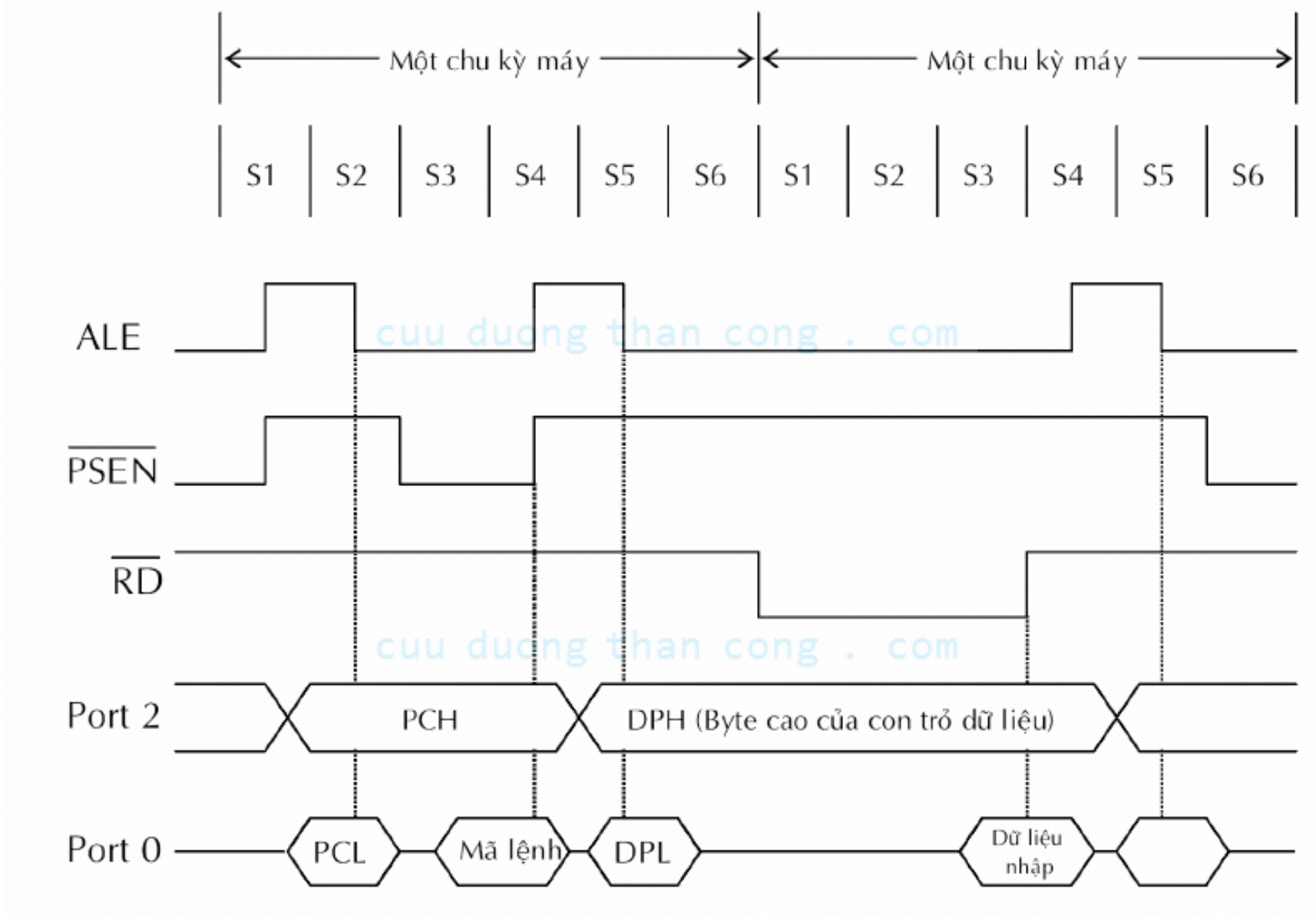


/PSEN ở mức thấp trong thời gian lấy lệnh

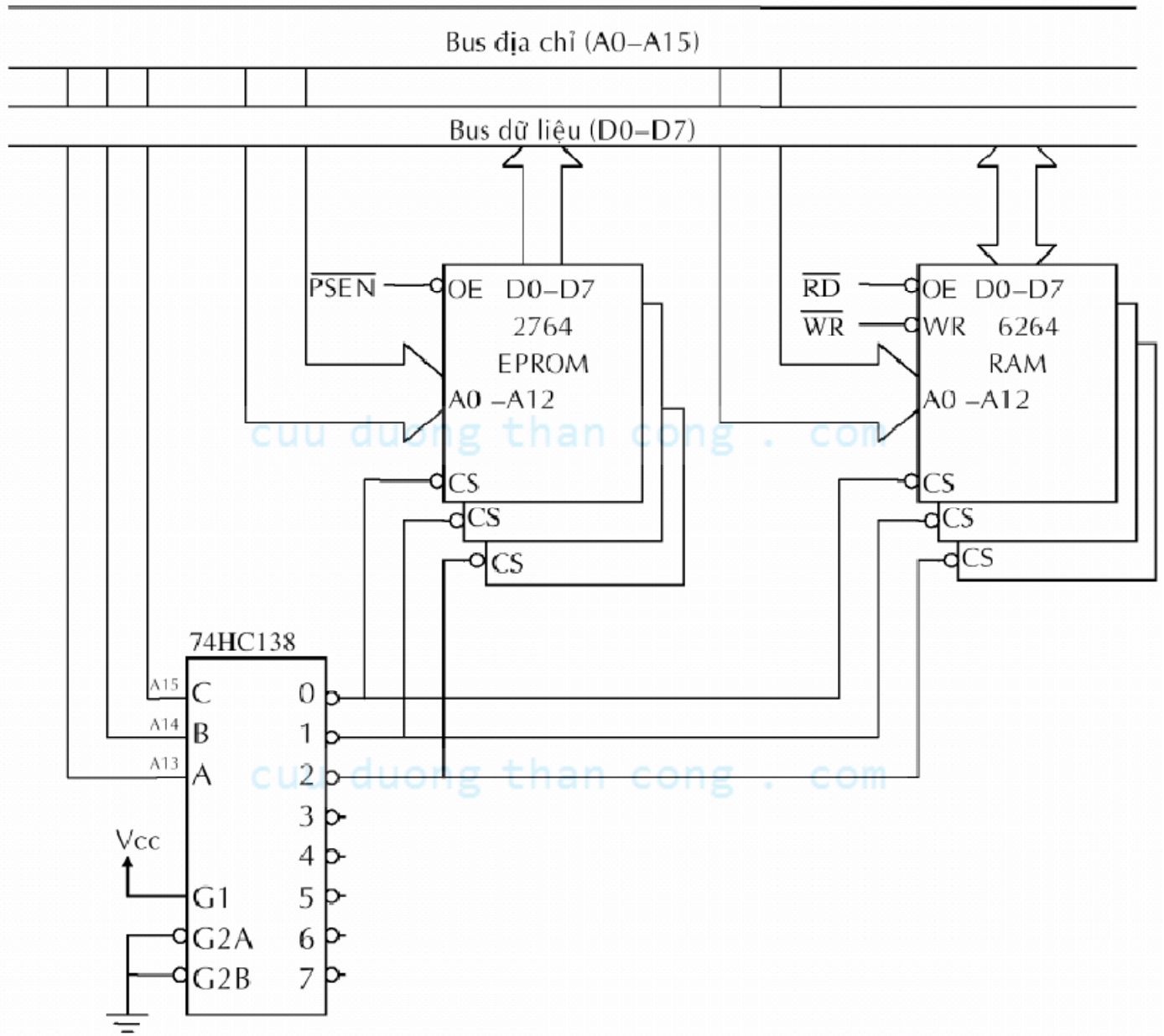
Truy cập bộ nhớ dữ liệu bên ngoài



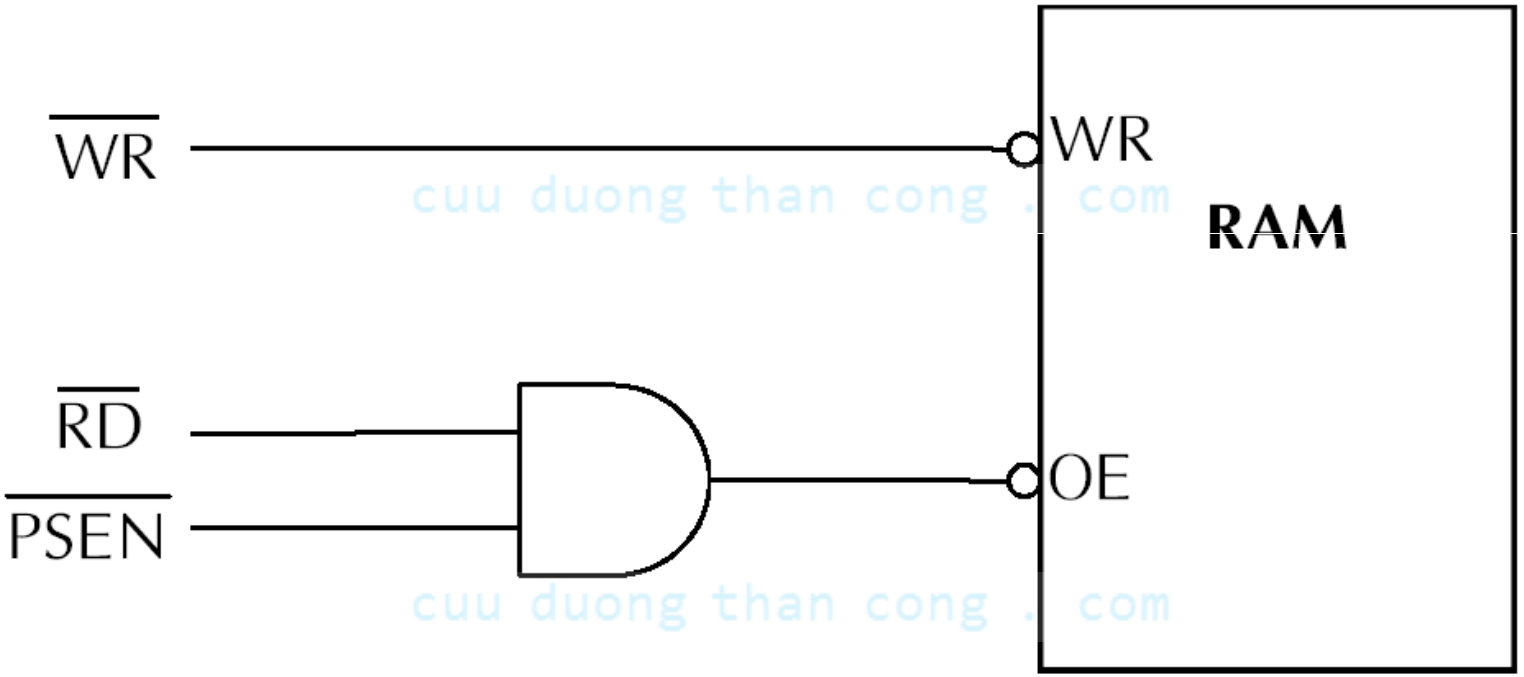
Giản đồ định thì cho lệnh MOVX



Mạch giải mã địa chỉ các EPROM 8KB và RAM 8KB với hệ 8051

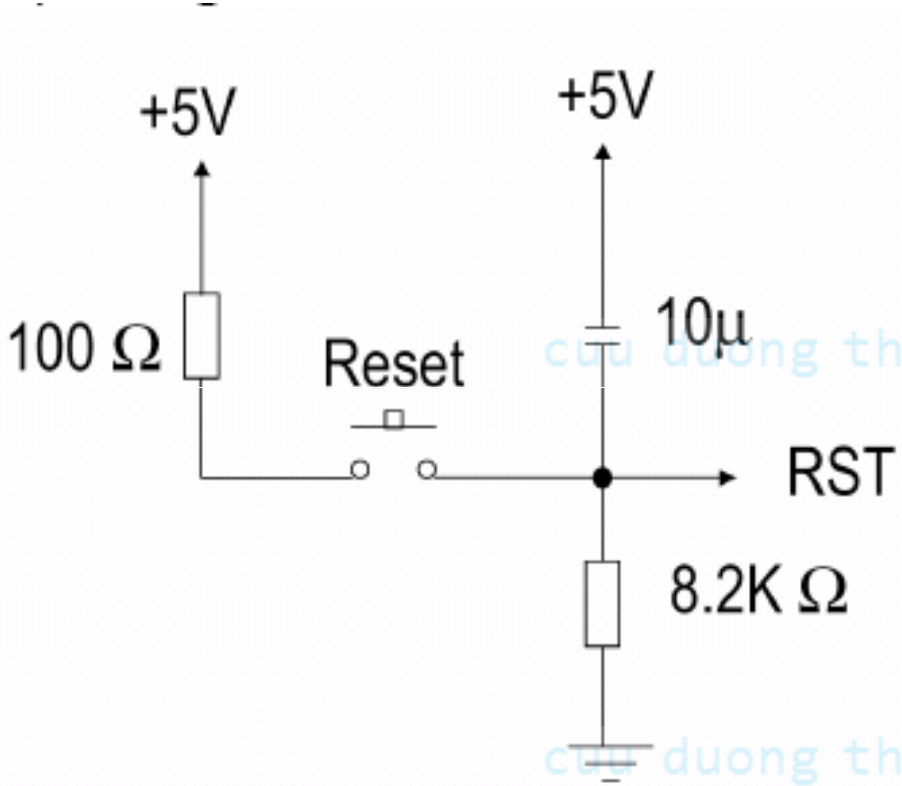


Phủ lấp vùng nhớ dữ liệu và chương trình bên ngoài

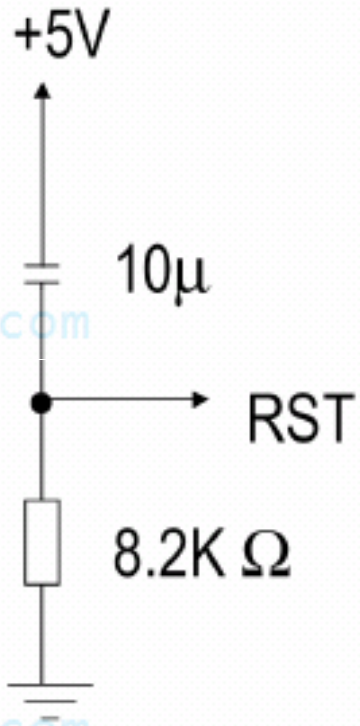


Hoạt động reset

Mạch reset hệ thống



a) Reset bằng tay



b) Reset khi mở nguồn cấp điện

Các giá trị thanh ghi sau khi reset hệ thống

Thanh ghi	Nội dung
Bộ đếm chương trình PC	0000H
Thanh ghi tích lũy ACC	00H
Thanh ghi B	00H
PSW	00H
SP	07H
DPTR	0000H
Ports 0–3	FFH
IP (8031/8051)	XXX00000B
IP (8032/8052)	XX000000B
IE (8031/8051)	0XX00000B
IE (8032/8052)	0X000000B
Các thanh ghi Timer	00H
SCON	00H
SBUF	00H
PCON (HMOS)	0XXXXXXXB
PCON (CMOS)	0XXX0000B

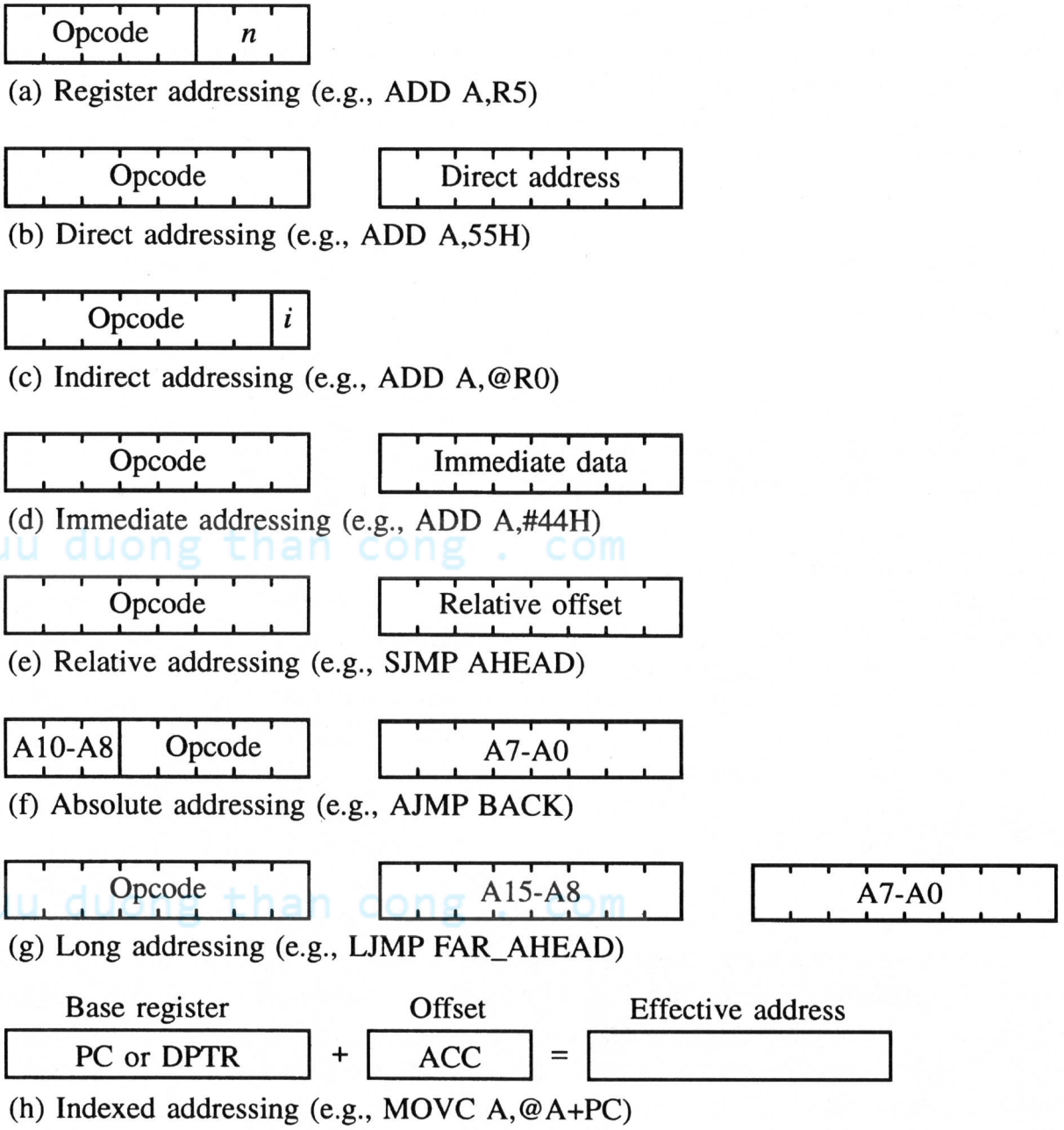
3.3 CÁC PHƯƠNG PHÁP ĐỊNH ĐỊA CHỈ

Các cách định vị địa chỉ

- Be the way to access data
- 8051 has different addressing mode:
  - Immediate (constant data)
  - Register (register data)
  - Direct (RAM data)
  - Register indirect (RAM data)
  - Indexed (ROM data)

  - relative addressing
  - Absolute addressing
  - Long addressing

8051 Instruction Format



Addressing Modes

1) Immediate Mode – specify data by its value

```
mov A, #0           ;put 0 in the accumulator
                    ;A = 00000000
mov R4, #11h        ;put 11hex in the R4 register
                    ;R4 = 00010001
mov B, #11          ;put 11 decimal in b register
                    ;B = 00001011
mov DPTR,#7521h     ;put 7521 hex in DPTR
                    ;DPTR = 0111010100100001
```

Addressing Modes

1) Immediate Mode – continue

```
        MOV DPTR,#7521h
        MOV DPL,#21H
        MOV DPH, #75
```

```
        COUNT EQU 30
         ~
         ~
        mov R4, #COUNT
```

```
        MOV DPTR,#MYDATA
        ~
        ~
        ORG 200H
MYDATA:DB "HELLO"
```

Notes of Immediate Addressing

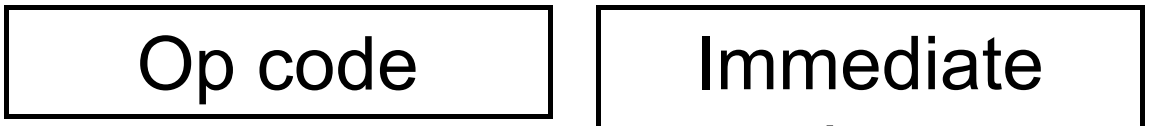
- Add ***"#"*** before any immediate data
- Only the source operand can be immediate
- Add ***"h"*** after a base-16 number, ***"b"*** after a base-2 number; otherwise assumed ***base-10***
- Use ' ' to enclose any character
- Precede all ***base-16*** numbers that begin with A-F by a ***"0"***

~~MOV A,#ABh~~

MOV A,#0ABH

8051 Instruction Format

- immediate addressing




```
add a,#3dh        ;machine code=243d
```

- Direct addressing


```
mov r3,0E8h       ;machine code=ABE8
```

Addressing Modes

2) Direct Mode – play with R0-R7 by direct address

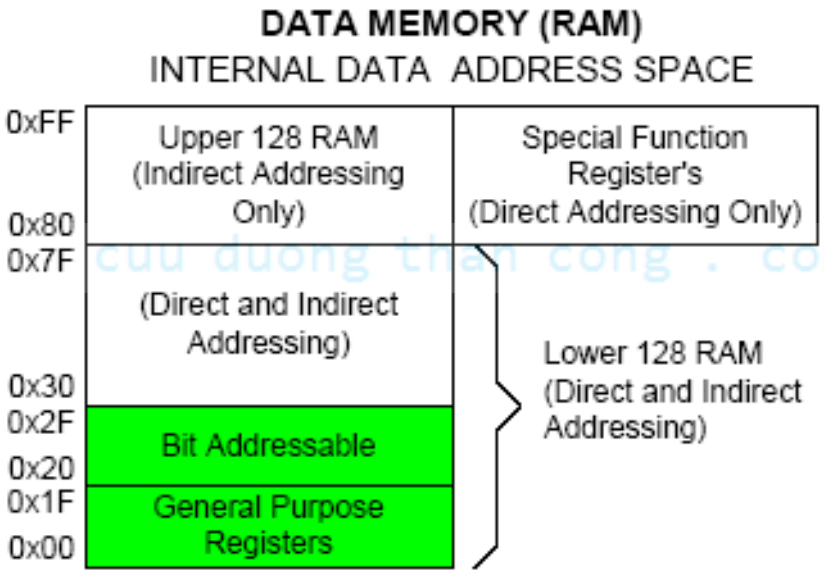
```
MOV A,4   ≡   MOV A,R4

MOV A,7   ≡   MOV A,R7

MOV 7,6   ≡   MOV R7,R6

MOV R2,#5     ;Put 5 in R2
MOV R2,5      ;Put content of RAM at 5 in R2
```

Addressing Modes

2) Direct Mode – specify data by its 8-bit address

Usually for 30h-7Fh of RAM

```
Mov a, 70h          ; copy contents of RAM at 70h to a
Mov R0,40h        ; copy contents of RAM at 40h to a
Mov 56h,a         ; put contents of a at 56h
Mov 0D0h,a            ; put contents of a into PSW
```



Examples of Direct Addressing

Instruction	Operation
MOV 80h, A or MOV P0, A	Copy contents of register A to location 80h (Port 0 latch)
MOV A, 80h or MOV A, P0	Copy contents of location 80h (Port 0 pins) to register A
ABC EQU 80h MOV A, ABC	; equate ; Port 0 to A
MOV R0, 12h	Copy contents from RAM location 12h to register R0
MOV 0A8h, 77h or MOV IE, 77h	Copy contents from RAM location 77h to IE register of SFRs

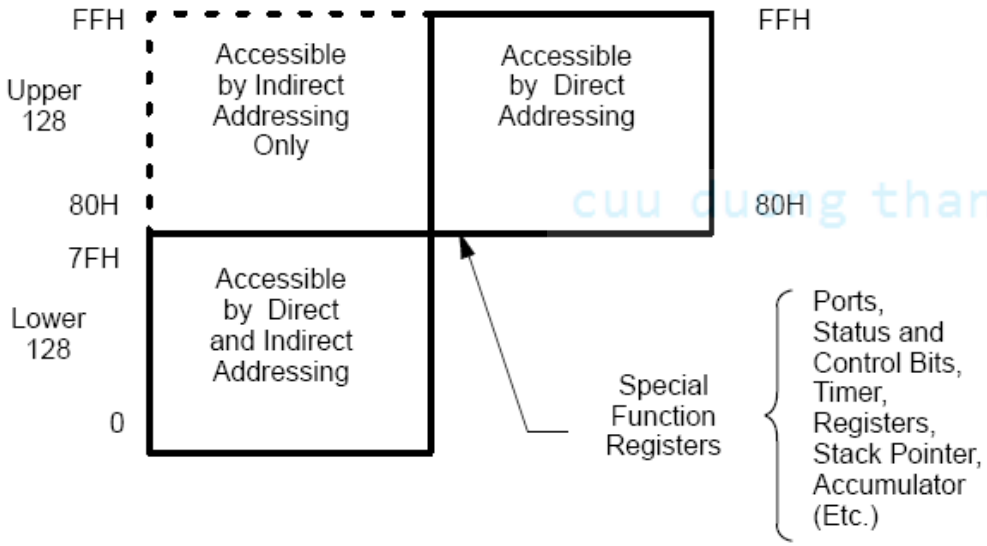
MOV direct,direct

Note: No "#" sign in the instruction

Examples of Direct Addressing

```
MOV A, 2        ; copy location 02 (R2) to A
MOV B, 2        ; copy location 02 (R2) to B
MOV 7, 2        ; copy location 02 to 07 (R2 to R7)
                ; since "MOV R7, R2" is invalid
```

MOV DIRECT, DIRECT



Stack and Direct Addressing Mode

Only direct addressing mode is allowed for pushing onto the stack

PUSH	direct
POP	direct

- PUSH A (Invalid)
- PUSH 0E0h (Valid)
- PUSH R3 (Invalid)
- PUSH 03 (Valid)
- POP R4 (Invalid)
- POP 04 (Valid)

Example

Show the code to push R5, R6, and A onto the stack and then pop them back into R2, R3, and B, where register B = register A, R2 = R6, and R3 = R5.

```
PUSH  05          ; push R5 onto stack
PUSH  06          ; push R6 onto stack
PUSH  0E0h        ; push register A onto stack
POP   0F0h        ; pop top of stack into register B
                  ; now register B = register A
POP   02          ; pop top of stack into R2
                  ; now R2 = R6
POP   03          ; pop top of stack into R3
                  ; now R3 = R5
```

Notes of Direct Addressing

- The address value is limited to one byte, 00 – FFh (128-byte RAM and SFR)
- Using MOV to move data **from itself to itself** can lead to unpredictable results → **error** Eg. MOV A, A
- MOV data **to** a port changes the port latch
- MOV data **from** port gets data from port pins

Addressing Modes

3) Register Addressing – either source or destination is one of CPU register

```
MOV R0,A
MOV A,R7
ADD A,R4
ADD A,R7
MOV DPTR,#25F5H
MOV R5,DPL
MOV R1,DPH
```

Note that MOV R4,R7 is incorrect

8051 Instruction Format

- Register addressing


```
070D E8      mov a,r0          ;E8 = 1110 1000
070E E9      mov a,r1          ;E9 = 1110 1001
070F EA      mov a,r2          ;EA = 1110 1010
0710 ED      mov a,r5          ;ED = 1110 1101
0711 EF      mov a,r7          ;Ef = 1110 1111
0712 2F      add a,r7
0713 F8      mov r0,a
0714 F9      mov r1,a
0715 FA      mov r2,a
0716 FD      mov r5,a
0717 FD      mov r5,a
```

Notes of Register Addressing

The most efficient addressing mode:

- No need to do memory access
- Instructions are much shorter
- Result: speed (hence efficiency) increased
- We can move data between ***Acc*** and ***Rn*** (n = 0 to 7) but movement of data between ***Rn*** registers is not allowed

  e.g. MOV R4, R7 ⟸ (**Invalid**)


```
;Use the following :
MOV   A, R7
MOV   R4,A
MOV   R4,07H      ; this is direct addressing mode
```

Review Questions

1. Can the programmer of a microcontroller make up new addressing modes?
2. Show the instruction to load 1000 0000 (binary) into R3.
3. Why is the following invalid? "**MOV R2, DPTR**"
4. True or false. DPTR is a 16-bit register that is also accessible in low-byte and high-byte formats.
5. Is the PC (program counter) also available in low-byte and high-byte formats?

Addressing Modes

4) Register Indirect – the address of the source or destination is specified in registers

Uses registers R0 or R1 for 8-bit address:

```
mov psw, #0          ; use register bank 0
mov r0, #3Ch
mov @r0, #3          ; M[3Ch] ← 3
```

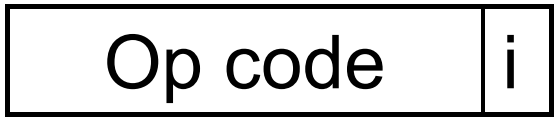
Uses DPTR register for 16-bit addresses:

```
mov dptr, #9000h  ; dptr ← 9000h
movx a, @dptr             ; a ← M[9000h]
```

Note that 9000h is an address in external memory

8051 Instruction Format

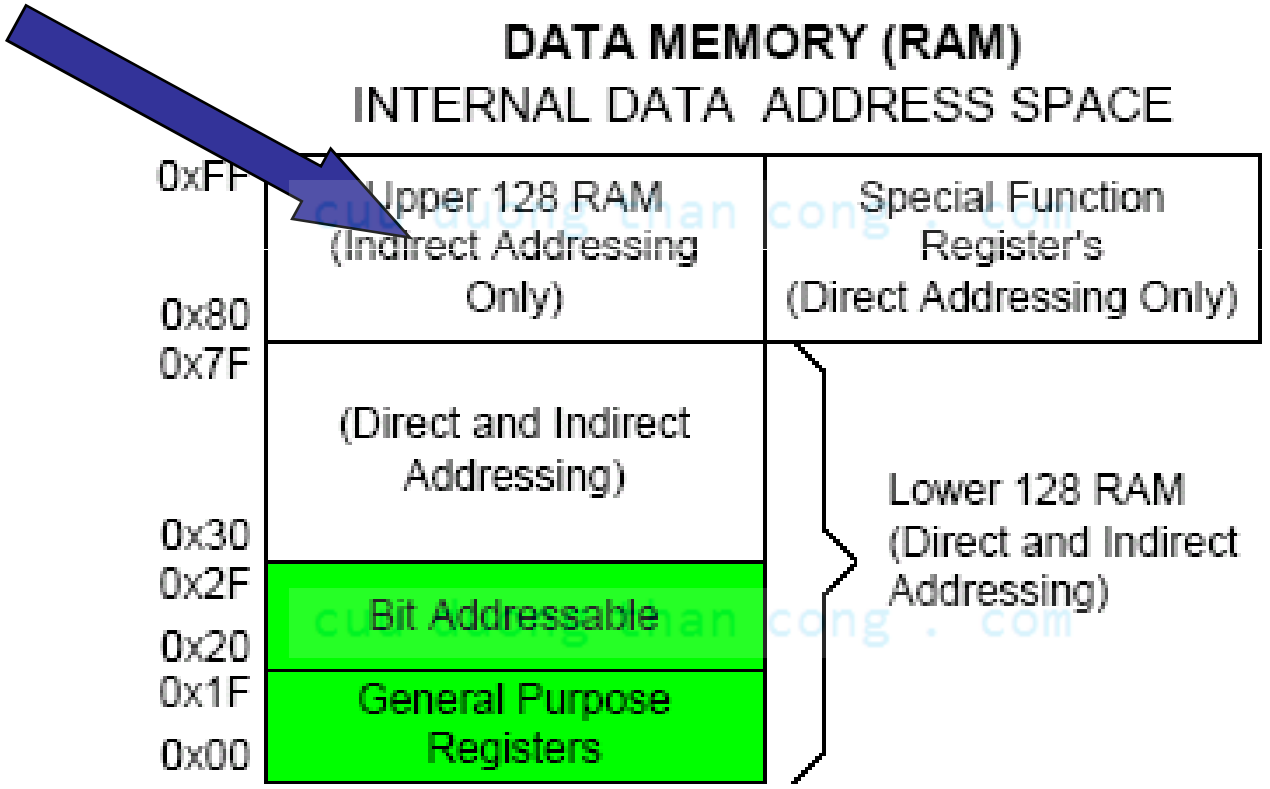
- Register indirect addressing



mov a, @Ri ; i = 0 or 1

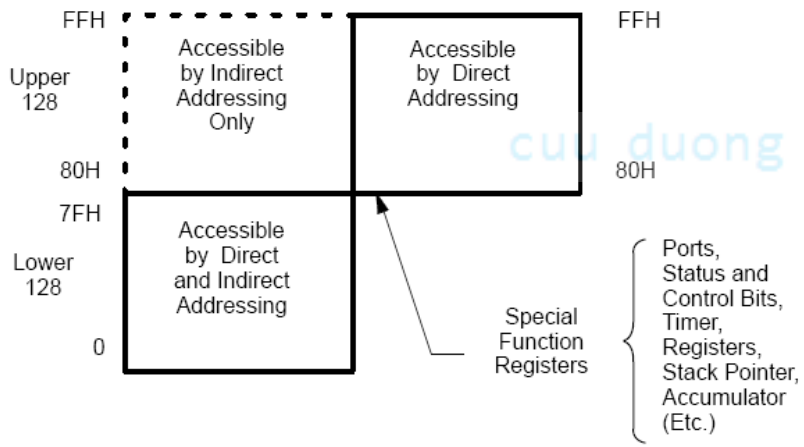
```
070D E7      mov  a,@r1
070D 93      movc a,@a+dptr
070E 83      movc a,@a+pc
070F E0      movx a,@dptr
0710 F0      movx @dptr,a
0711 F2      movx @r0,a
0712 E3      movx a,@r1
```

Use Register Indirect to access upper RAM block (+8052)

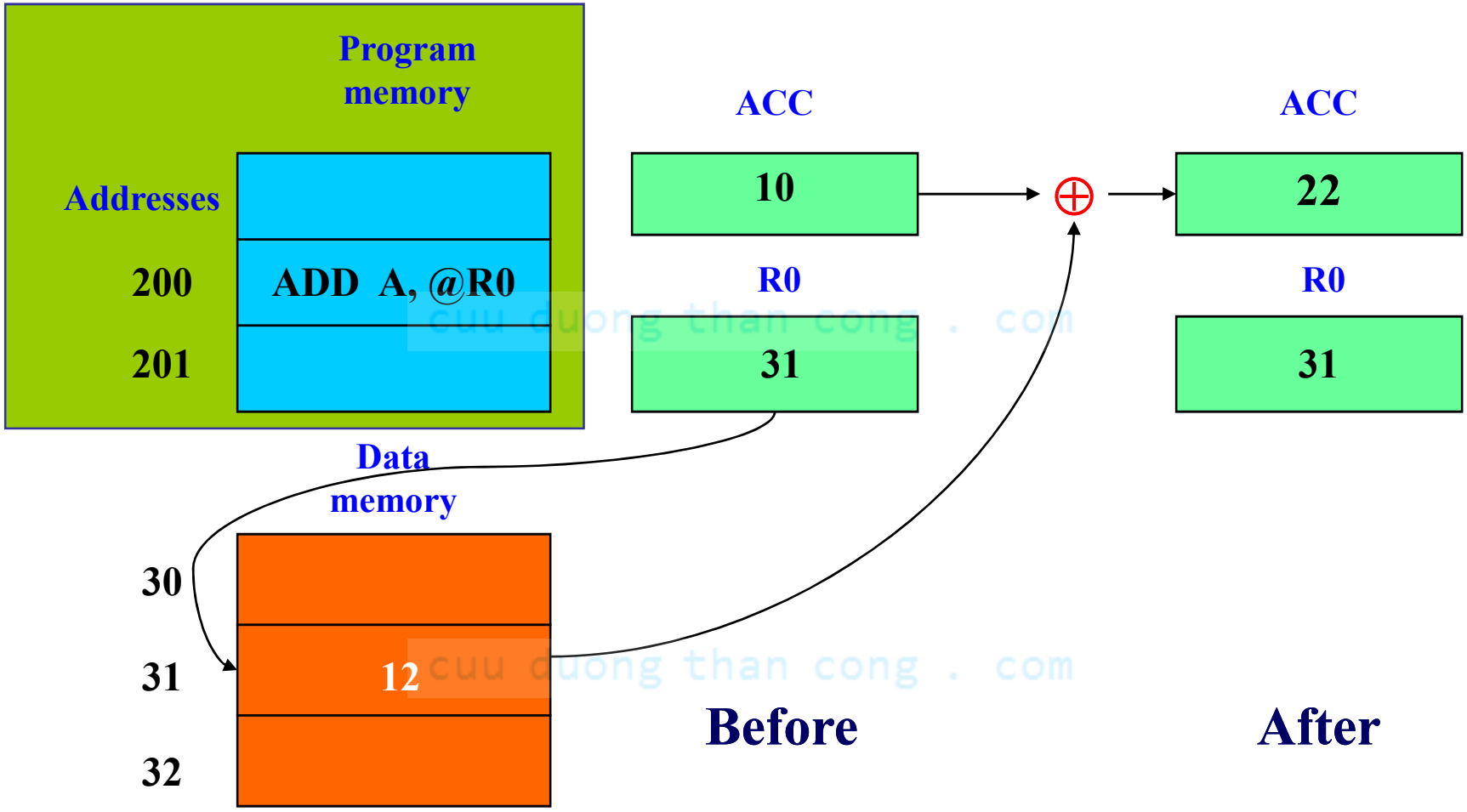


Register Indirect Addressing

- Use a register to hold the address of the operand; i.e. using a register as a ***pointer***
- ***Only R0 and R1*** can be used when data is inside the CPU (address ranges from 00h – 7Fh) eg. MOV A, @R1
- ***R0 ,R1*** and ***DPTR*** can be used when addressing external memory locations eg. MOVX A,@R1 MOVX A,@DPTR
- Must put a **"@"** sign before the register name



Register Indirect Addressing (eg. ADD A,@R0)



8051 Internal data memory

Examples of Indirect Addressing

Instruction	Operation
MOV @R1, A	Copy the data in A to the address pointed to by the contents of R1
MOV A, @R0	Copy the contents of the address pointed to by register R0 to the A register
MOV @R1, #35h	Copy the number 35h to the address pointed to by register R1
MOV @R0, 80h or MOV @R0, P0	Copy the contents of the port 0 pins to the address pointed to by register R0.
MOVX A, @R0	Copy the contents of the **external data address** pointed to by register R0 to the A register
MOVX A, @DPTR	Copy the contents of the **external data address** pointed to by register DPTR to the A register

MOV @Ri,#data where i = 0 or 1

Example

Write a program segment to copy the value 55h into RAM memory locations 40h to 44h using:

(a) Direct addressing mode;

(b) register indirect addressing mode without a loop; and

(c) with a loop

Solution to Example (a)

Direct addressing mode

```
MOV   A, #55h      ; load A with value 55h
MOV   40h, A ; copy A to RAM location 40h
MOV   41h, A ; copy A to RAM location 41h
MOV   42h, A ; copy A to RAM location 42h
MOV   43h, A ; copy A to RAM location 43h
MOV   44h, A ; copy A to RAM location 44h
```

Solution to Example (b)

register indirect addressing mode without a loop

```
MOV   A, #55h      ; load A with value 55h
MOV   R0, #40h     ; load the pointer. R0 = 40h
MOV   @R0, A       ; copy A to RAM location R0 points to
INC    R0          ; increment pointer. Now R0 = 41h
MOV   @R0, A       ; copy A to RAM location R0 points to
INC    R0          ; increment pointer. Now R0 = 42h
MOV   @R0, A       ; copy A to RAM location R0 points to
INC    R0          ; increment pointer. Now R0 = 43h
MOV   @R0, A       ; copy A to RAM location R0 points to
INC    R0          ; increment pointer. Now R0 = 44h
MOV   @R0, A       ; copy A to RAM location R0 points to
```

Solution to Example (c)

Loop method

```
        MOV   A, #55h        ; A = 55h
        MOV   R0, #40h       ; load pointer. R0 = 40h, RAM add.
        MOV   R2, #05        ; load counter, R2 = 5
AGAIN:
        MOV   @R0, A         ; copy 55A to RAM location R0 points to
        INC    R0            ; increment R0 pointer
        DJNZ  R2, AGAIN      ; loop until counter = zero
```

```
        MOV     R2, #05h      ; example
LP:
        ;-------------------------------
        ; do 5 times inside the loop
        ;-------------------------------
        DJNZ  R2, LP     ; R2 as counter
```

"DJNZ" : decrement and jump if Not Zero

DJNZ direct, relative

DJNZ Rn, relative where n = 0,1,,,7

Example (looping)

Write a program segment to clear 15 RAM locations starting at RAM address 60h.

```
        CLR   A              ; A = 0
        MOV   R1, #60h       ; load pointer. R1 = 60h
        MOV   R7, #15        ; load counter, R7 = 15 (0F in HEX)
AGAIN:  MOV   @R1, A         ; clear RAM location R1 points to
        INC    R1            ; increment R1 pointer
        DJNZ   R7, AGAIN     ; loop until counter = zero
```



```
; clear one ram location at address 60h
CLR    A
MOV    R1,#60h
MOV    @R1,A
```

Setup a loop using DJNZ and register R7 as counter

Example (block transfer)

Write a program segment to copy a block of 10 bytes of data from RAM locations starting at 35h to RAM locations starting at 60h.

```
        MOV   R0, #35h    ; source pointer
        MOV   R1, #60h    ; destination pointer
        MOV   R3, #10     ; counter
BACK:
        MOV   A, @R0      ; get a byte from source
        MOV   @R1, A      ; copy it to destination
        INC   R0          ; increment source pointer
        INC   R1          ; increment destination pointer
        DJNZ  R3, BACK    ; keep doing it for all ten bytes
```

Notes of Indirect Addressing

- Using pointer in the program enables handling ***dynamic data structures*** ➡ an advantage
- Dynamic data: the data value is not fixed
- In this mode, we can defer the calculation of the address of data and the determination of the amount of memory to allocate at (program) runtime (eg. MOV A, @R0)

Register or direct addressing (eg. MOV A, 30H) cannot be used , since they require operand addresses to be known at assemble-time.

Addressing Modes

5) Register Indexed Mode – source or destination address is the sum of the base address and the accumulator (Index)

- Base address can be DPTR or PC

```
mov dptr, #4000h
mov a, #5
movc a, @a + dptr  ;a ← M[4005]
```

Addressing Modes

5) Register Indexed Mode continue

- Base address can be DPTR or PC


```
                ORG 1000h
          1000  mov a, #5
PC  ⟶     1002  movc a, @a + PC    ;a ← M[1008]
          1003  Nop
```


- Lookup Table
- MOVC only can read internal code memory

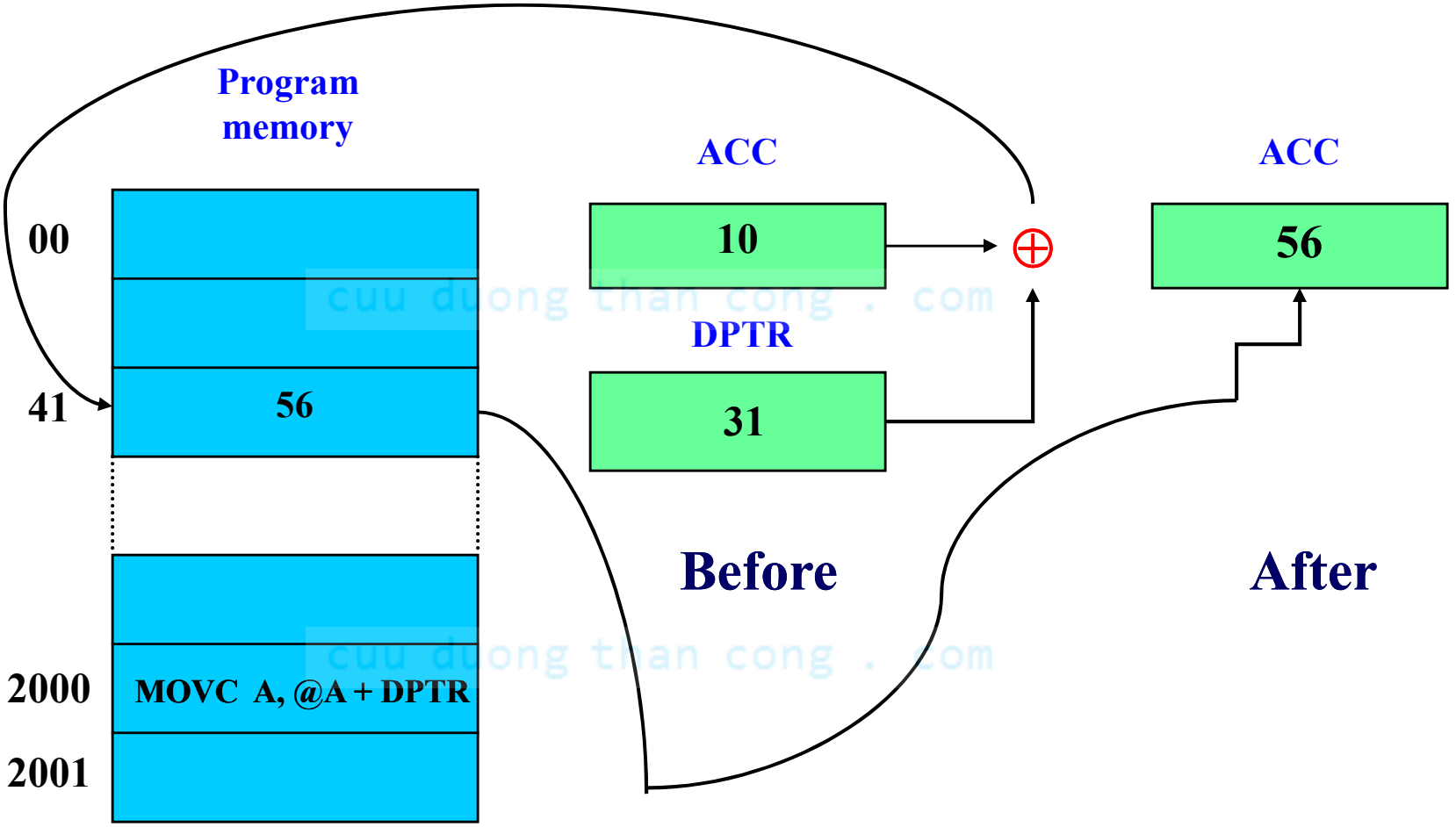
Indexed Addressing

```
MOVC   A, @A+DPTR
MOVC   A, @A+PC
JMP    @A+DPTR
```

- Using a ***base register*** (starting point) and an ***offset*** (how much to parse through) to form the effective address for a JMP or MOVC instruction
- Used to parse through an array of items or a look-up table
- Usually, the **DPTR** is the base register and the "**A**" is the offset
- **A** increases/decreases to parse through the list

Indexed Addressing

Example: MOVC A,@A+DPTR



Examples of Indexed Addressing

Instruction	Operation
MOVC A, @A + DPTR	Copy the code byte, found at the ROM address formed by adding register A and the DPTR register, to A
MOVC A, @A + PC	Copy the code byte, found at the ROM address formed by adding A and the PC, to A
JMP @A + DPTR	Jump to the address formed by adding A to the DPTR, this is an unconditional jump and will always be done.

Example (look-up table)

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

```
        ORG    0
        MOV    DPTR, #300h  ; load look-up table address
        MOV    A, #0FFh     ; A = FF
        MOV    P1, A        ; configure P1 as input port
BACK:   MOV    A, P1        ; get X
        MOV    A, @A+DPTR   ; get X square from table
        MOV    P2, A        ; issue it to P2
        SJMP   BACK         ; keep doing it

        ORG    300h
TABLE:  DB     0, 1, 4, 9, 16, 25, 36, 49, 64, 81
        END
```

Review Questions

1. The instruction “MOV A, 40h” uses _____ addressing mode. Why?
2. What address is assigned to register R2 of bank 0?
3. What address is assigned to register R2 of bank 2?
4. What address is assigned to register A?
5. Which registers are allowed to be used for register indirect addressing mode if the data is in on-chip RAM?

Access to Accumulator

- A register can be accessed by direct and register mode

- This 3 instruction has same function with different code

```
0703 E500          mov a,00h
0705 8500E0        mov acc,00h
0708 8500E0        mov 0e0h,00h
```

- Also this 3 instruction

```
070B E9            mov a,r1
070C 89E0          mov acc,r1
070E 89E0          mov 0e0h,r1
```

Access to SFRs

- B – always direct mode - except in MUL & DIV

```
0703 8500F0          mov b,00h
0706 8500F0          mov 0f0h,00h

0709 8CF0            mov b,r4
070B 8CF0            mov 0f0h,r4
```

- P0~P3 – are direct address

```
0704 F580            mov p0,a
0706 F580            mov 80h,a
0708 859080          mov p0,p1
```

- Also other SFRs (pcon, tmod, psw,….)

SFRs Address

All SFRs such as

(ACC, B, PCON, TMOD, PSW, P0~P3, …)

are accessible by name and direct address

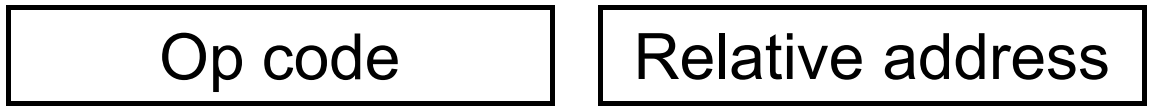
But

both of them

Must be coded as direct address

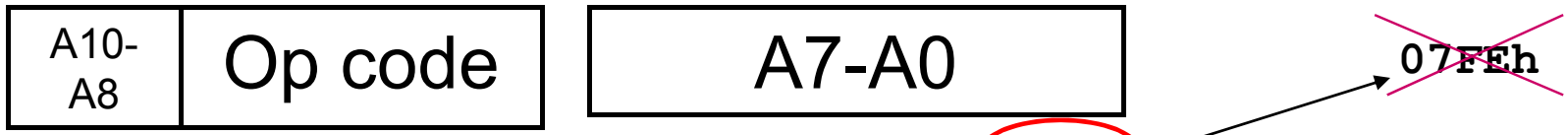
8051 Instruction Format

- 6) relative addressing




```
here:  sjmp  here     ;machine code=80FE (FE=-2)
Range = (-128 ~ 127)
```


- 7) Absolute addressing (limited in 2k current mem block)

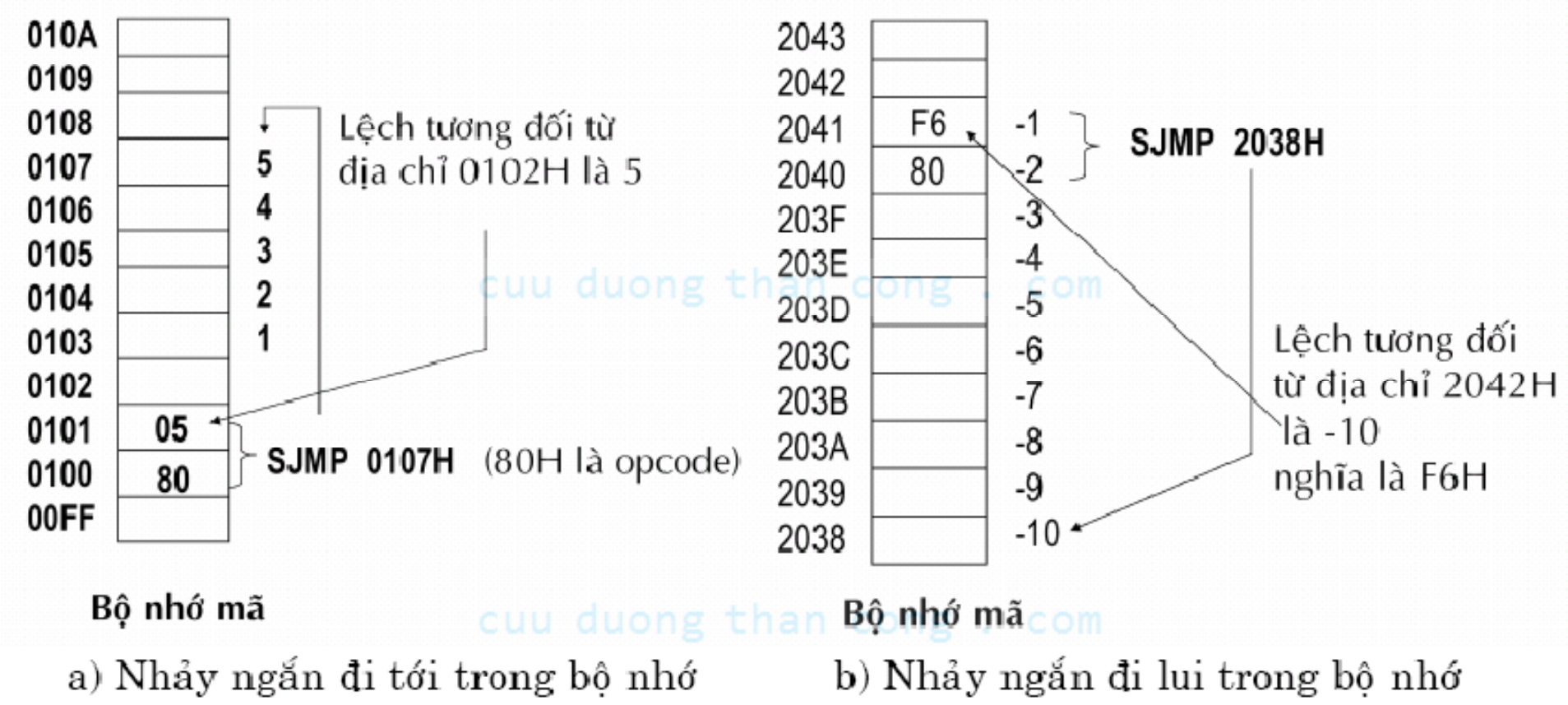



```
0700
0700 E106
0702 00
0703 00
0704 00
0705 00
```

```
1            org 0700h
2            ajmp next    ;next=706h
3            nop
4            nop
5            nop
6            nop
7      next:
8            end
```

07FEh

Tính toán offset với định địa chỉ tương đối



a) Nhảy ngắn đi tới trong bộ nhớ

b) Nhảy ngắn đi lui trong bộ nhớ

Offset tương đối = byte thấp của (địa chỉ nhảy đến – địa chỉ lệnh kế)

Relative Addressing

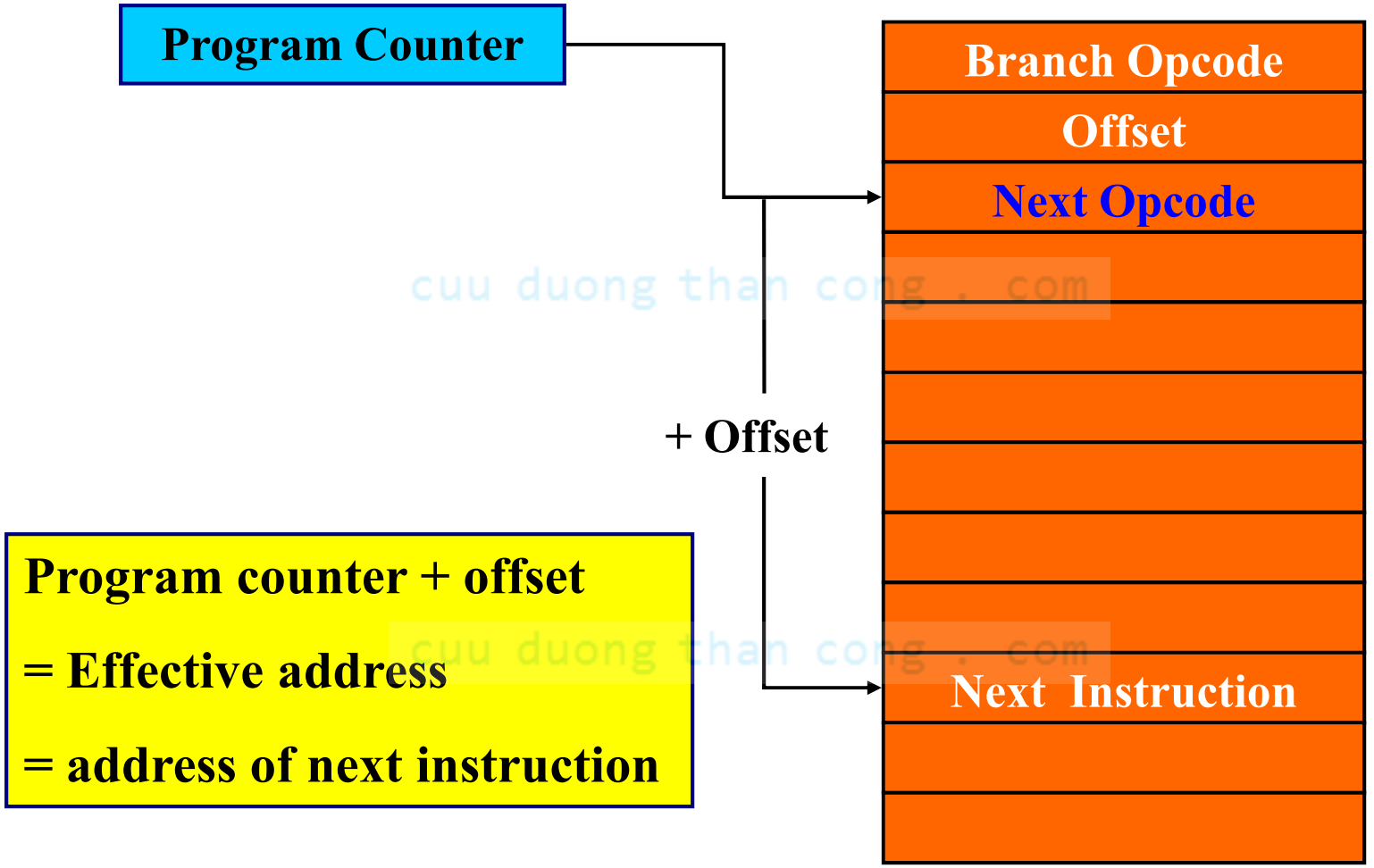
```
SJMP  relative
DJNZ  direct, relative
DJNZ  Rn, relative, where n=0,1,,,7
```

- Used in jump (“JMP”) instructions
- Relative address: an 8-bit value (-128 to +127)
- You may treat relative address as an offset
- Labels indicate the JMP destinations (i.e. where to stop)
- Assembler finds out the relative address using the label

Relative Addressing

- The relative address is added to the PC
- The sum is the address of the next instruction to be executed
- As a result, program skips to the desired line right away instead of going through each line one by one
- Labels indicate the JMP destinations (i.e. where to stop).

Relative Addressing



Examples of Relative Addressing

Instruction	Operation
SJMP NXT	Jump to relative address with the label 'NXT'; this is an unconditional jump and is always taken.
DJNZ R1, DWN	Decrement register R1 by 1 and jump to the relative address specified by the label 'DWN' if the result of R1 is not zero.

Code Address	Op Codes			Mnemonics	Operands	Remarks
0030	74	89		MOV	A, #89h	; load A with data 89h
0032	00			NOP		; endless loop
0033	80	FD		SJMP	0032h	
0035						

Absolute Addressing

```
ACALL  address11
AJMP   address11
```

- Only used with the instructions ***ACALL*** and ***AJMP***
- Similar to indexed addressing mode
- The largest "jump" that can be made is 2K

$2^{11} = 2048 = 2K$

The subroutine called must therefore start within the same 2K Block of the program memory as the first byte of the instruction following ACALL.

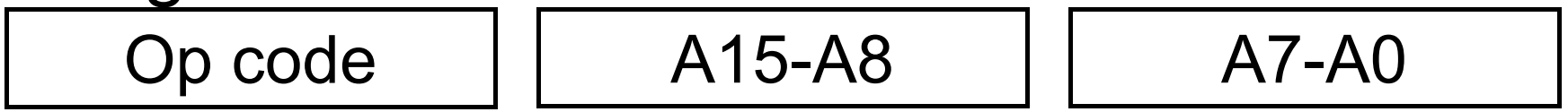
Absolute Addressing

ACALL address11
AJMP address11

```
            ORG     00H                 ; reset location
            LJMP    START               ; 3 bytes instruction

            ORG     3FFEH
START:      ACALL   FORWARD             ; 2 bytes instruction
; now code address at 4000H
            LJMP    TEST

            ORG     47FFH               ; 0100011111111111B
FORWARD:
            RET

            ORG     5800H               ; 0101100000000000B
BACKWARD:
            RET

            ORG     5FFDH
TEST:       ACALL   BACKWARD            ; 2 bytes instruction
; now code address at 5FFFH
            SJMP    $
            END
```

8051 Instruction Format

- Long distance address



- **Range = (0000h ~ FFFFh)**


```
0700                    1           org 0700h
0700 020707             2           ajmp next    ;next=0707h
0703 00                 3           nop
0704 00                 4           nop
0705 00                 5           nop
0706 00                 6           nop
                        7     next:
                        8           end
```

Bài tập 1

Cho biết mã máy và các cách định địa chỉ của các lệnh sau:

1. MOVX @DPTR, A
2. SETB P3.1
3. ADD A, R3
4. MOV A, #0FBH
5. MOV A, @R1
6. MOV 41H, A

Bài tập 2

1. Viết các lệnh thực hiện cất giá trị FFH vào RAM dữ liệu bên ngoài ở địa chỉ 19A3H.
2. Sau đoạn chương trình này, cho biết các địa chỉ bit có nội dung là 1:
   a) MOV 25h, #13h
   b) MOV R0, #22h
   c) MOV @R0, 25h

Bài tập 3

1. Cho biết mã máy sau biểu diễn lệnh/tác vụ gì?

   75H, 8AH, E7H

2. Giả sử lệnh

   AJMP LABEL

   trong bộ nhớ mã ở địa chỉ 1600H và 1601H, và nhãn LABEL ứng với lệnh ở địa chỉ 1723H

   a) Cho biết mã máy của lệnh này?

   b) Lệnh này có hợp lệ không khi LABEL ứng với lệnh ở địa chỉ 1A23H? Giải thích