



KỸ THUẬT VI XỬ LÝ

Giảng viên: **Nguyễn Thanh Ngọc**

GIỚI THIỆU HỌC PHẦN

❖ TỔNG THỜI LƯỢNG: 03 TÍN CHỈ

- Thời lượng: 35 tiết
- Lý thuyết: 25 tiết
- Thực hành/thí nghiệm: 10 tiết

❖ ĐÁNH GIÁ KẾT QUẢ HỌC TẬP:

- Điểm chuyên cần:
 - Đi học đầy đủ, đúng giờ
 - Tham gia xây dựng bài
 - Ý thức khi tham gia học tập
- Hình thức kiểm tra giữa kỳ: Thi viết
- Hình thức thi kết thúc học phần: Trắc nghiệm
- Hoàn thành tối thiểu 3/5 bài thực hành mới được thi

TÀI LIỆU THAM KHẢO

- [1] **Cấu trúc và lập trình họ vi điều khiển 8051**- Nguyễn Tăng Cường – NXB Khoa học & Kỹ thuật.
- [2] **Vi điều khiển** – Trần Viết Thắng, Phạm Hùng Kim Khánh – Trường đại học kỹ thuật công nghệ
- [3] **Vi xử lý** - Hồ Trung Mỹ - NXB ĐHQG, 2003 (TLTK chính).
- [4] **Họ vi điều khiển 8051** - Tống Văn On, Hoàng Đức Hải - NXB LĐXH, 2001.
- [5] **Vi Xử Lý trong Đo Lường và Điều Khiển** - Ngô Diên Tập - NXB Khoa Học & Kỹ Thuật, 2000.



NỘI DUNG MÔN HỌC

Chương 1: Tổng quan về bộ vi xử lý và hệ vi xử lý.

Chương 2: Vi điều khiển 8051

Chương 3: Bộ đếm/ định thời và truyền thông nối tiếp trong 8051

Chương 4: Lập trình ngắt trong 8051

Chương 5: Bộ vi xử lý ARM

Thực hành: - Mô phỏng mạch điện tử số bằng phần mềm Proteus
- Thực hiện tại phòng máy



Chương 1.

TỔNG QUAN VỀ BỘ VXL VÀ HỆ VXL



MỤC TIÊU

- Hiểu được cấu trúc và nguyên lý hoạt động của Hệ vi xử lý
- Hiểu được cấu trúc và nguyên lý hoạt động của Bộ vi xử lý

NỘI DUNG

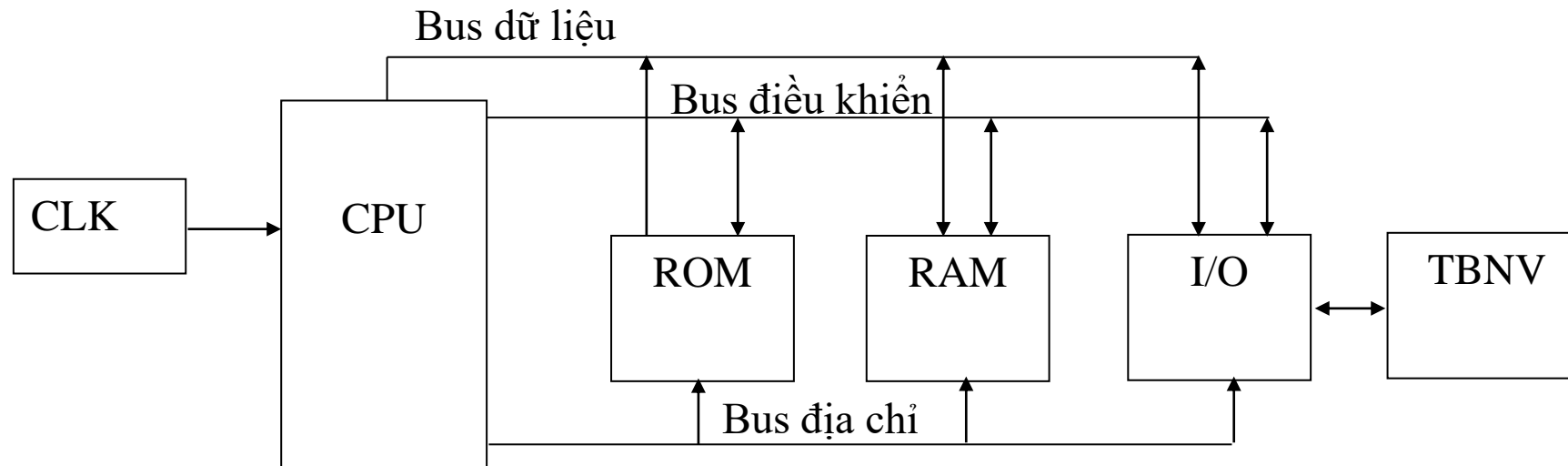
1.1. Hệ vi xử lý

1.2. Bộ vi xử lý

1.3. Các bước thiết kế hệ vi xử lý

1.1. Hệ vi xử lý (1)

1.1.1. Cấu trúc phần cứng



Hình 1.1 . Sơ đồ khối của hệ vi xử lý

1.1. Hệ vi xử lý (2)

1.1.2. Phần mềm

- ❖ Phần mềm là chương trình do người lập trình tạo ra để giải quyết nhiệm vụ bài toán đặt ra khi thiết kế hệ thống. Các chương trình này được đặt trong bộ nhớ ROM và RAM khi hệ thống hoạt động.
- ❖ Trong máy tính, phần mềm được phân lớp.
 - Các lớp trong là các chương trình quản lý, điều khiển, các dữ liệu cố định... làm nhiệm vụ trung gian giữa người và máy được gọi là hệ điều hành (HĐH). Các chương trình của HĐH thường khá lớn và được nạp một phần trong ROM, phần còn lại được chứa trong bộ nhớ ngoài và được nạp vào RAM mỗi khi khởi động máy.
 - Các chương trình do người lập trình viết ở lớp ngoài để giải quyết một bài toán nào đó, khi thực hiện sẽ được HĐH nạp vào trong RAM ở vị trí theo sự phân bố của HĐH. Khi phần cứng thực hiện xong chương trình đó thì bài toán được giải.

1.2. Hệ vi xử lý (3)

1.1.3. Hoạt động của hệ vi xử lý

- ❖ ***Quá trình nhận lệnh:*** CPU nhận lệnh từ bộ nhớ trung tâm. Để nhận 1 byte lệnh, từ giá trị của thanh ghi địa chỉ lệnh, CPU tạo ra các bit địa chỉ đưa ra Bus địa chỉ xác định byte lệnh cần lấy vào. Sau đó CPU đưa các tín hiệu điều khiển đọc byte lệnh về CPU. Khi đó nội dung của các thanh ghi địa chỉ lệnh tự động tăng 1 đơn vị để chuẩn bị cho việc nhận byte lệnh tiếp theo.
- ❖ ***Quá trình giải mã lệnh:*** Byte đầu tiên của lệnh là byte mã lệnh. CPU giải mã byte mã lệnh này để xác định chức năng của lệnh. Nếu lệnh không cần toán hạng chuyển sang chu kỳ thực hiện lệnh. Nếu lệnh cần toán hạng, CPU xác định toán hạng cho lệnh.
- ❖ ***Quá trình thực hiện lệnh.***
 - Nếu là lệnh dừng, thì CPU dừng quá trình làm việc cho tới khi Reset
 - Nếu không phải thì CPU thực hiện lệnh và gửi kết quả. Trong quá trình thực hiện lệnh, nếu là lệnh thao tác ngay trong bộ VXL thì CPU sẽ thực hiện lệnh, còn nếu là lệnh cần thao tác với bên ngoài thì CPU gửi tín hiệu địa chỉ và điều khiển để thực hiện.
 - Sau đó quá trình lại quay lại lệnh tiếp theo. Các quá trình nhận lệnh, giải mã lệnh và thực hiện lệnh có thể tiến hành tuần tự hay song song tùy theo cấu trúc từng bộ VXL

1.2. Bộ vi xử lý (1)

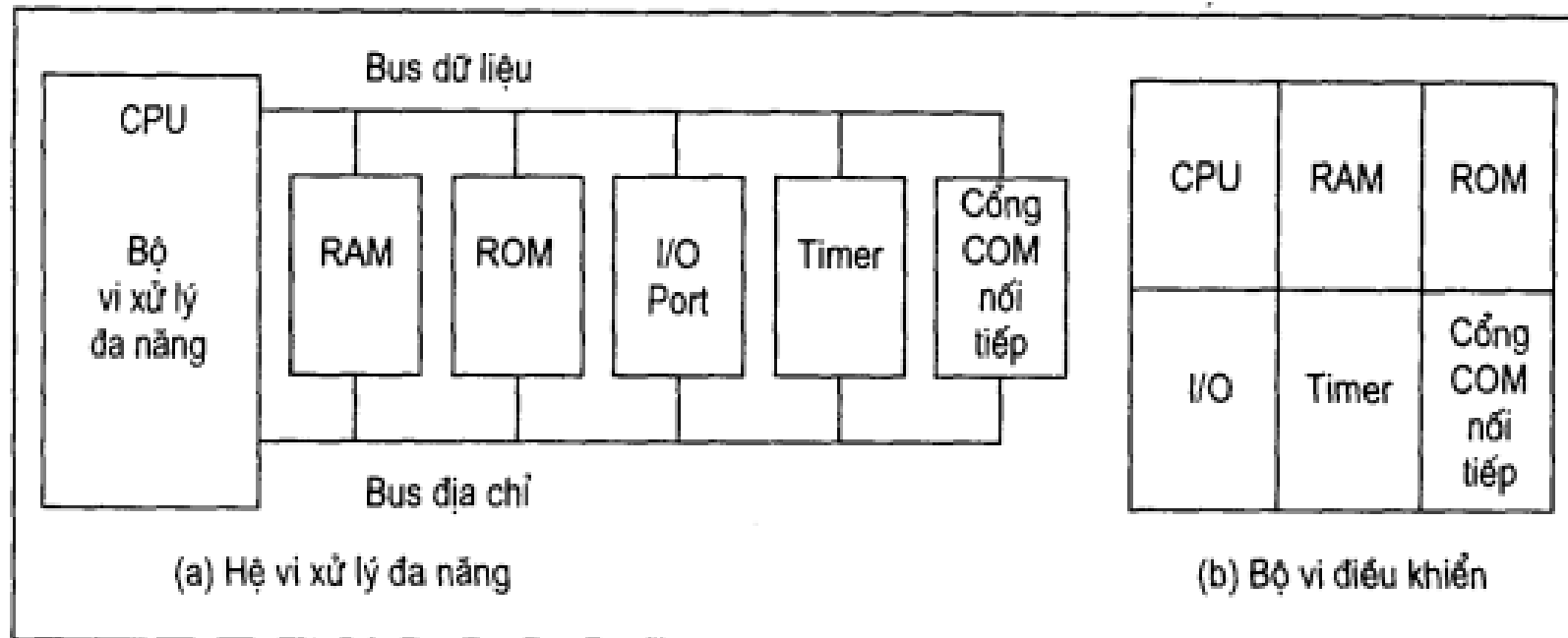
1.2.1. Khái niệm bộ VXL

- Là một vi mạch logic chế tạo theo công nghệ LSI (Large scale integrated) cho đến VLSI (very large scale integrated)
- Hoạt động dưới sự điều khiển của chương trình đặt trong bộ nhớ.
- Bộ VXL là hạt nhân của hệ VXL, nắm quyền kiểm soát toàn bộ các thành phần có trong hệ thống.
- Thực hiện các phép toán số học, logic, đưa ra các quyết định và thông tin với thế giới bên ngoài qua các cổng vào/ra dưới sự điều khiển của chương trình cài đặt trong bộ nhớ.

1.1. Bộ vi xử lý (2)

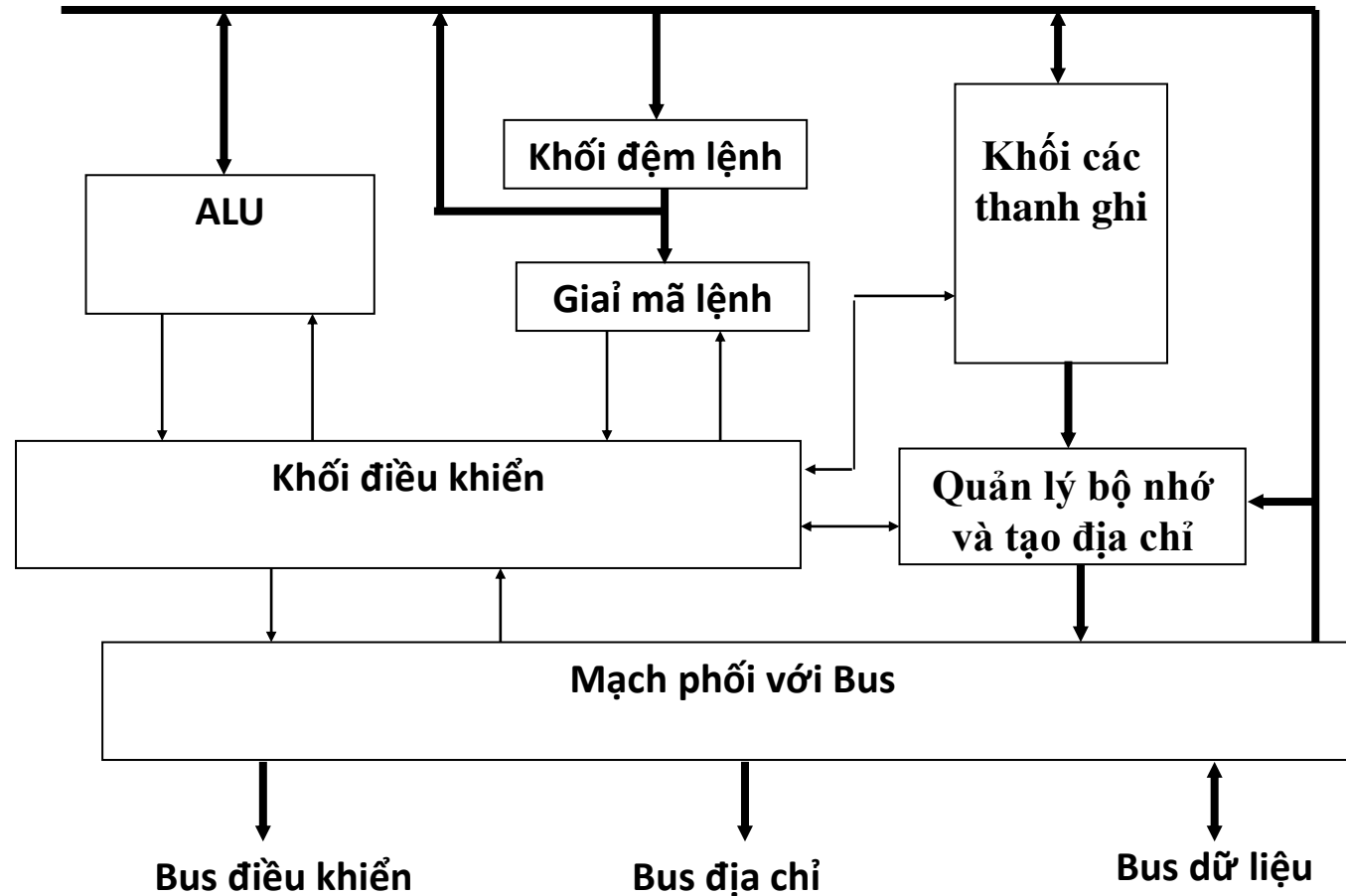
1.1.2. Phân loại bộ vi xử lý

- *VXL đa năng*: bao gồm tất cả các phần tử tính toán trên một chip (trừ bộ nhớ và các cổng vào ra – 8086/8088, 6800)
- *Vi điều khiển*: bao gồm cả bộ nhớ và các cổng vào ra



1.1. Bộ vi xử lý (3)

1.1.3. Cấu trúc chung của bộ vi xử lý



Hình 1.2. Cấu trúc chung của bộ VXL

1.3. Các bước thiết kế một hệ vi xử lý

- Phân tích chức năng nhiệm vụ của hệ vi xử lý cần thiết kế.
- Tổ chức phần cứng cho hệ vi xử lý cần thiết kế.
- Xây dựng phần mềm cho hệ vi xử lý cần thiết kế.
- Nạp chương trình cho hệ vi xử lý là hiệu chỉnh hệ thống.



Chương 2.

VI ĐIỀU KHIỂN 8051

MỤC TIÊU

- Hiểu được sơ đồ và chức năng chân tín hiệu của VĐK 8051, từ đó có thể ghép nối VĐK 8051 với các thành phần phần cứng bên ngoài.
- Hiểu được cấu trúc bên trong, tập lệnh của VĐK 8051, từ đó có thể xây dựng chương trình bằng tập lệnh của 8051

NỘI DUNG

2.1. Tổng quan về 8051

2.2. Sơ đồ chân tín hiệu của 8051

2.3. Tổ chức bộ nhớ của 8051

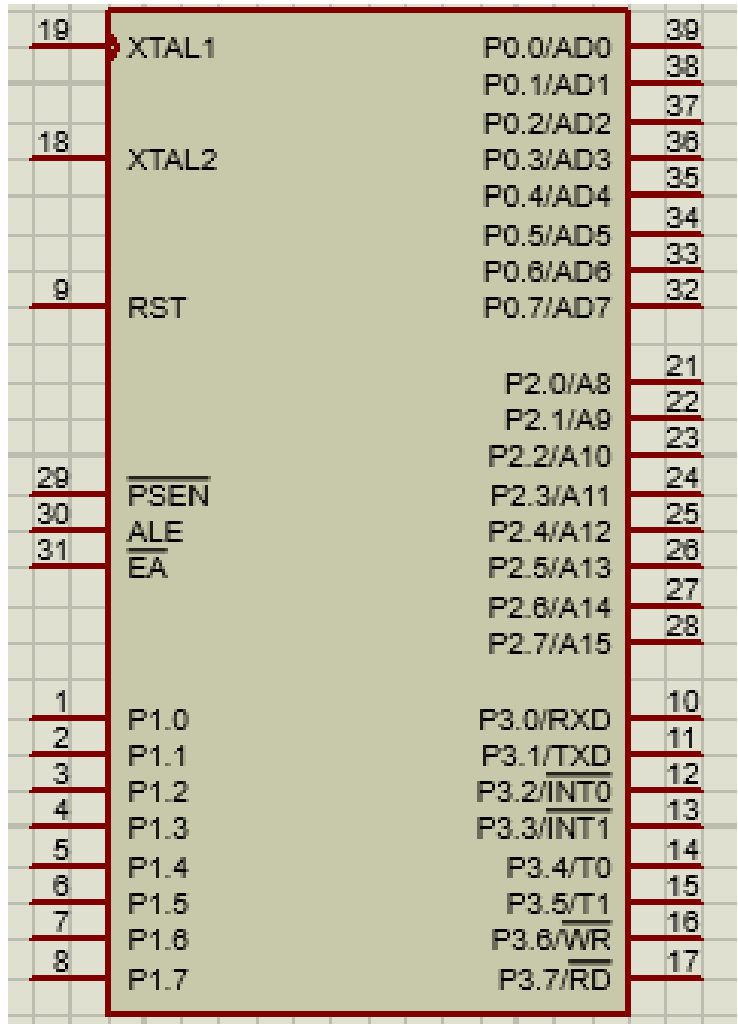
2.4. Tập lệnh của 8051

2.5. Khung chương trình hợp ngữ cho 8051

2.1. Tổng quan về 8051

- Đóng trong hộp 40 chân
- 4KB ROM
- 128 byte RAM
- 4 cổng xuất nhập dữ liệu 8 bit
- 1 cổng nối tiếp
- 2 bộ Timer 16 bit
- 6 nguồn ngắt (2 ngắt ngoài)
- Không gian nhớ chương trình ngoài 64KB
- Không gian nhớ dữ liệu ngoài 64KB
- 210 vị trí được định địa chỉ bit
- Tần số xung nhịp (11.0592 MHz, 12 MHz, 16MHz, 24 MHz)

2.2. Sơ đồ, chức năng chân tín hiệu 8051 (1)

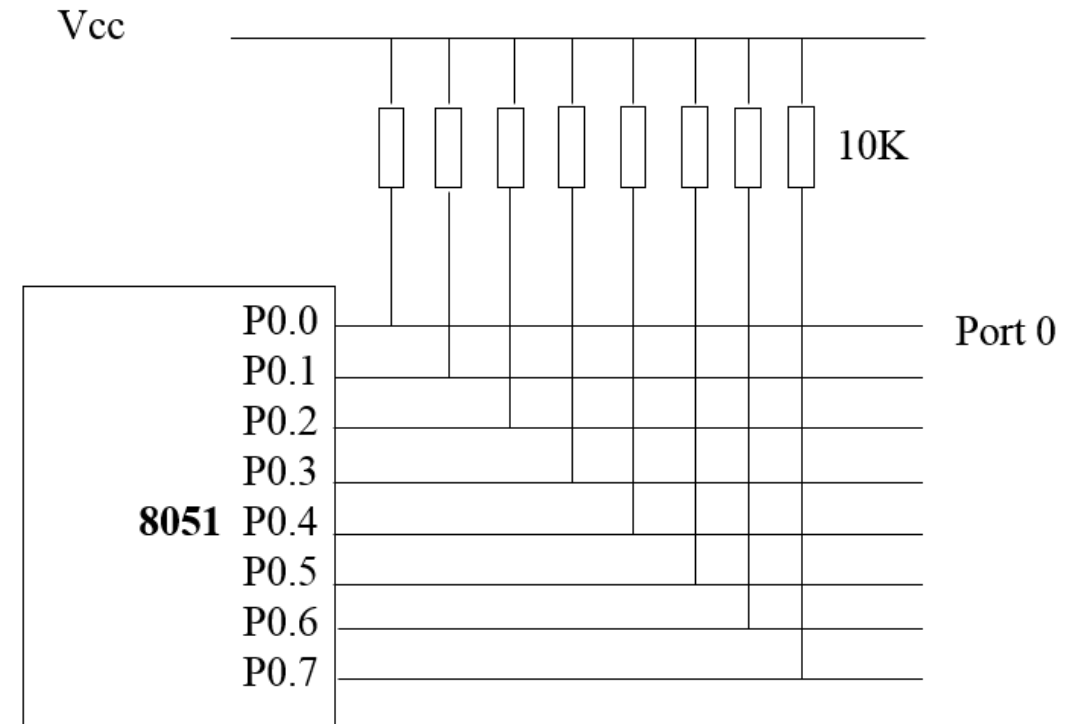


Hình 2.1. Sơ đồ chân của 8051

2.2. Sơ đồ, chức năng chân tín hiệu 8051 (2)

- **Port 0 (từ chân 32 đến 39), có 02 chức năng:**
 - Trong các thiết kế tối thiểu, các chân này có chức năng xuất/nhập dữ liệu;
 - Là byte địa chỉ thấp cho các thiết kế có bộ nhớ ngoài.

Các chân của cổng P0 có dạng cực máng hở, nên phải được nối với một điện trở kéo 10KΩ bên ngoài, Hình 2.2



Hình 2.2. Mắc điện trở kéo cổng P0

2.2. Sơ đồ, chức năng chân tín hiệu 8051 (3)

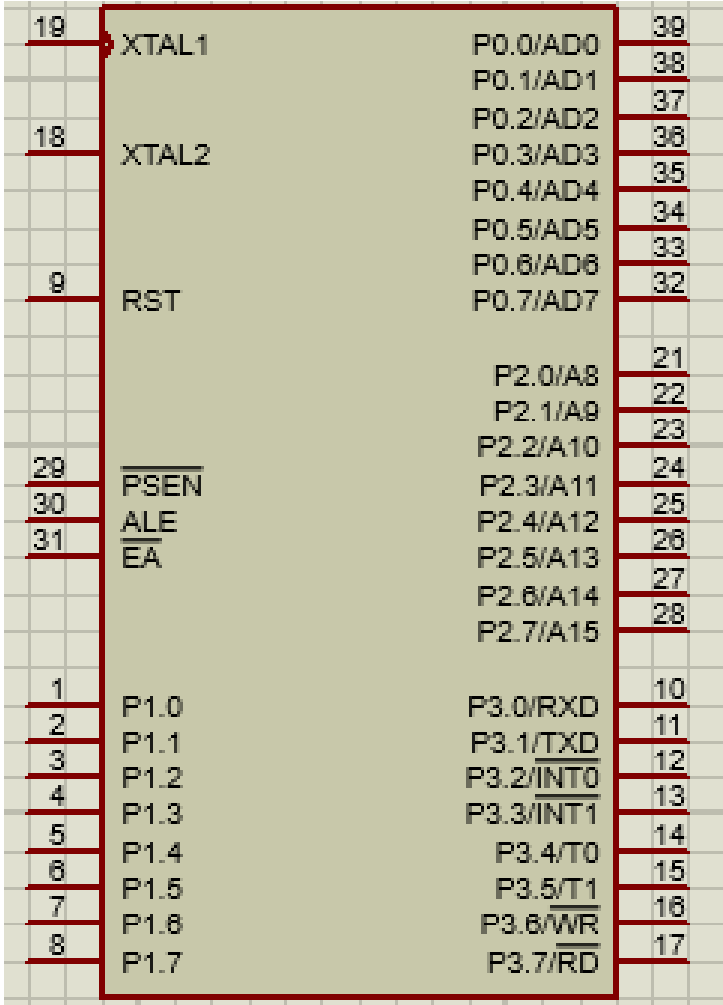
- **Port 1 (từ chân 1 đến 8)**, có 01 chức năng duy nhất là trao đổi dữ liệu với thiết bị bên ngoài khi có yêu cầu
- **Port 2 (từ chân 21 đến 28)**, có 02 chức năng:
 - Trao đổi dữ liệu với thiết bị bên ngoài
 - Là byte cao địa chỉ cho các thiết kế có bộ nhớ ngoài
- **Port 3 (từ chân 10 đến 17)**, có 02 chức năng:
 - Trao đổi dữ liệu với thiết bị bên ngoài
 - Khi không hoạt động trao đổi dữ liệu, mỗi chân của Port 3 thực hiện chức năng riêng (Bảng 2.1)

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

2.2. Sơ đồ, chức năng chân tín hiệu 8051 (4)

Bảng 2.1 Chức năng của các chân Port 3

Bit	Tên	Địa chỉ bit	Chức năng
P3.0	RxD	B0H	Chân nhận dl nối tiếp
P3.1	TxD	B1H	Chân phát dl nối tiếp
P3.2	$\overline{\text{INT0}}$	B2H	Ngõ vào ngắt ngoài 0
P3.3	$\overline{\text{INT1}}$	B3H	Ngõ vào ngắt ngoài 1
P3.4	T0	B4H	Ngõ vào của bộ định thời/đếm 0
P3.5	T1	B5H	Ngõ vào của bộ định thời/đếm 1
P3.6	$\overline{\text{WR}}$	B6H	Điều khiển ghi bộ nhớ dl ngoài
P3.7	$\overline{\text{RD}}$	B7H	Điều khiển đọc bộ nhớ dữ liệu ngoài



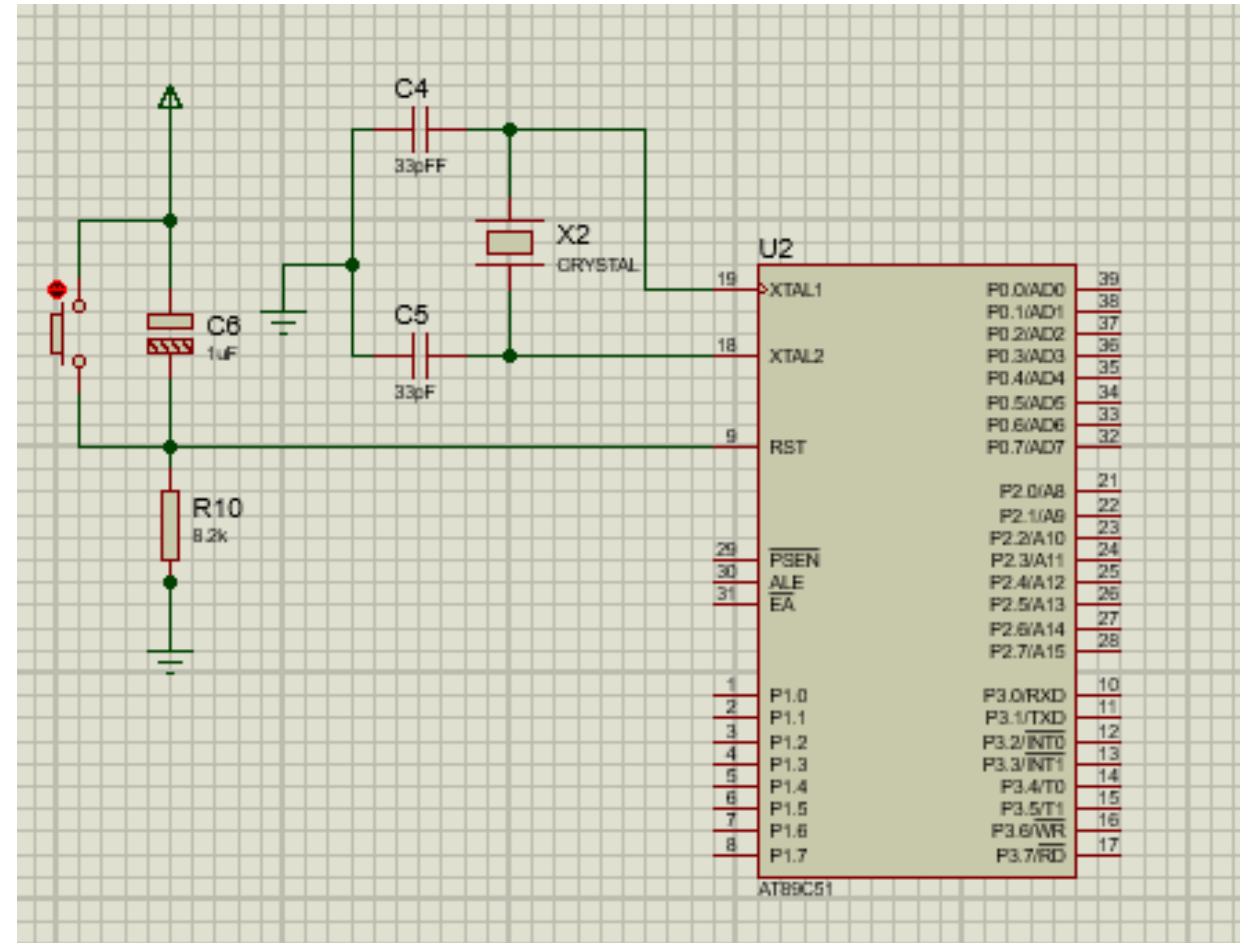
2.2. Sơ đồ, chức năng chân tín hiệu 8051 (5)

- $\overline{\text{PSEN}}$ (program store enable)- chân 29: tín hiệu điều khiển cho phép truy xuất bộ nhớ chương trình ngoài. Thường nối với chân $\overline{\text{OE}}$ của ROM để cho phép đọc các byte lệnh. Trong suốt thời gian tìm – nạp lệnh chân này ở mức logic 0. Khi thực thi chương trình trong ROM nội, chân này không tích cực (logic 1)
- ALE (address latch enable): khi Port 0 được sử dụng làm bus địa chỉ.
- $\overline{\text{EA}}$ -chân 31: nếu ở mức logic 1, 8051 thực thi chương trình trong ROM nội. Nếu ở mức logic 0, thực thi chương trình từ bộ nhớ ngoài.

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

2.2. Sơ đồ, chức năng chân tín hiệu 8051 (6)

- RST (Reset): bình thường ở mức logic thấp. Khi có mức logic cao vi điều khiển khởi động lại, khi đó mọi giá trị trên các thanh ghi bị xóa, bộ đếm chương trình $PC=0$ (tức là CPU nhận lệnh đầu tiên tại địa chỉ $0000h$ – địa chỉ khởi động của 8051)
- Chân XTAL1, XTAL2: mạch tạo dao động ngoài cho 8051. Hình 2.3



Hình 2.3. Mạch tạo dao động, Reset cho 8051

2.3. Tổ chức bộ nhớ 8051

2.3.1. Tổ chức bộ nhớ trong

2.3.2. Tổ chức bộ nhớ ngoài

2.3.3. Ví dụ tổ chức bộ nhớ ngoài

2.3.1. Tổ chức bộ nhớ trong (1)

Hình 2.8. trình bày vùng nhớ RAM trong có 128 byte được gán địa chỉ từ 00H đến 7FH.

-Từ 00H đến 1FH (32 byte) là các bảng thanh ghi và ngăn xếp.

-Từ 20H đến 2FH (16 byte) được định địa chỉ bit. Vùng nhớ này có thể truy nhập theo byte, hoạt bit tùy theo lệnh. Ví dụ:

MOV A, 2Ch ; đọc cả byte

ORL A, #10000000B; set bit cao nhất

MOV 2CH, A; ghi lại cả byte

Ba lệnh trên tương đương lệnh “SETB 67H”

-Từ 30H đến 7FH (80 byte) được sử dụng để lưu thông tin khi đọc và ghi, hay được gọi là bảng nháp. 80 byte này thường được sử dụng để lưu dữ liệu và tham số.

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

Hình 2.8. Tổ chức bộ nhớ RAM trong

2.3.1. Tổ chức bộ nhớ trong (2)

❖ Bảng thanh ghi

- Bảng 0 được mặc định khi sử dụng, để chuyển sang các bảng thanh ghi khác bằng cách sử dụng bit D3, D4 (PSW.3, PSW.4) của thanh ghi PSW, Bảng 2.2. Các bit này có thể được lập, xóa theo lệnh SETB, CLR. Ví dụ, “SETB PSW.3”
- Lưu ý: mặc định các bit này = 0.

Ví dụ 2.1:

SETB PSW.3 ; PSW.3=1

;SETB PSW.4 ; PSW.4=1

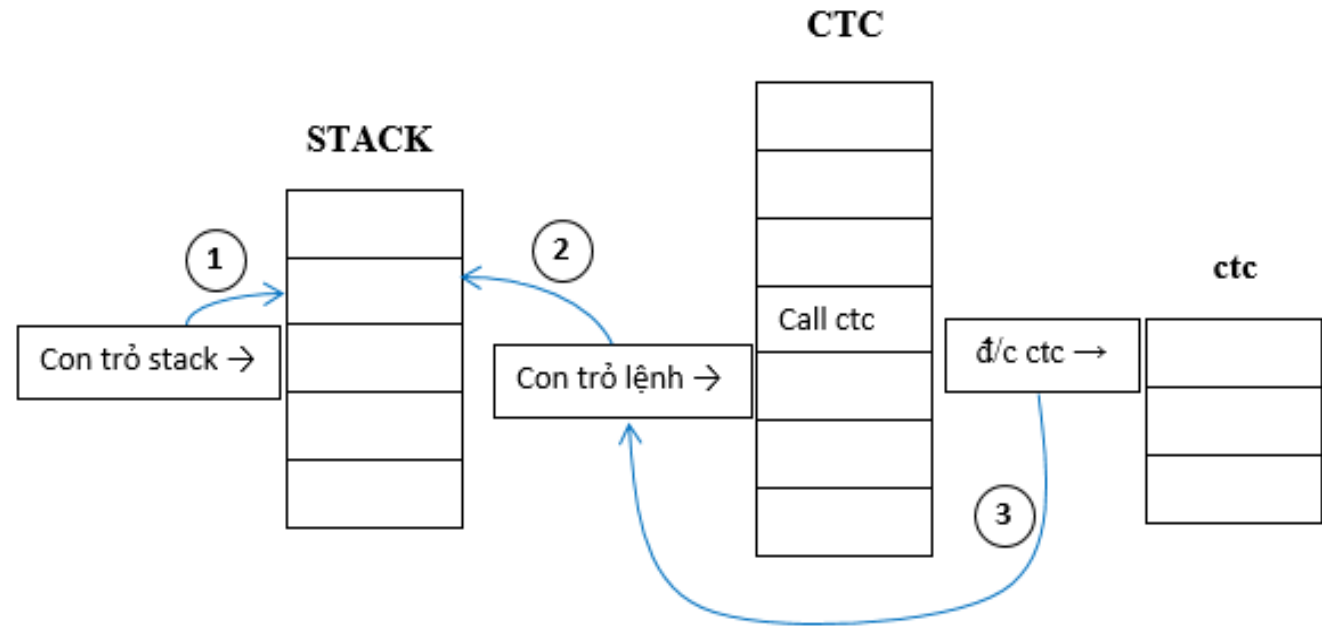
MOV R0, #99; nạp giá trị 99H vào R0

Bảng 2.2. Chọn bảng thanh ghi

	PSW.4	PSW.3
Bảng 0	0	0
Bảng 1	0	1
Bảng 2	1	0
Bảng 3	1	1

2.3.1. Tổ chức bộ nhớ trong (3)

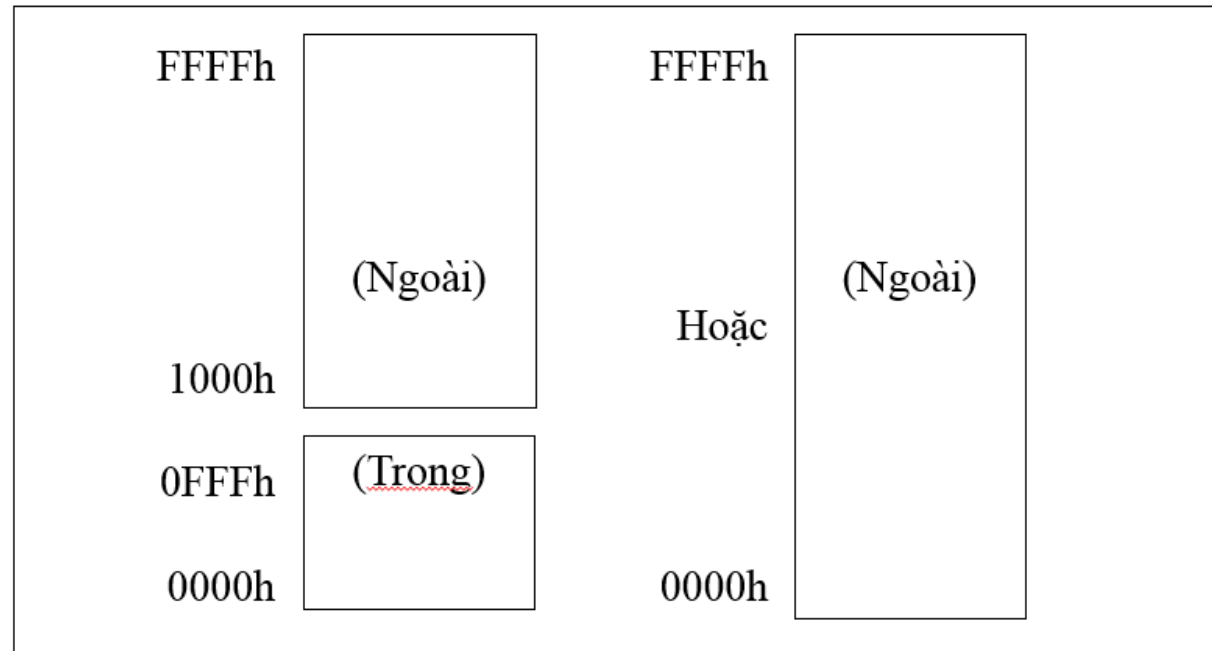
❖ Ngăn xếp (stack)



Ngăn xếp là một vùng nhớ RAM được sử dụng để lưu thông tin (có thể là dữ liệu hoặc địa chỉ) tạm thời. SP (stack pointer) độ dài 8 bit, được sử dụng để trỏ tới đỉnh ngăn xếp. Khi cất dữ liệu vào stack SP được tăng thêm 1.

- Khi bật nguồn, hoặc reset SP=07H, nghĩa là ngăn nhớ có địa chỉ 07H là ngăn nhớ đầu tiên được dùng làm ngăn xếp. Khi đó dung lượng tối đa của ngăn xếp là 24 byte (từ 08H đến 1FH). Nếu chương trình cần nhiều hơn 24 byte cho vùng ngăn xếp thì có thể đổi SP trỏ tới vùng từ 30H đến 7Fh bằng lệnh “MOV SP, #XX”.
- Cần lưu ý rằng các vùng nhớ được định địa chỉ bit không được sử dụng cho vùng ngăn xếp.

2.3.2. Tổ chức bộ nhớ ngoài (1)

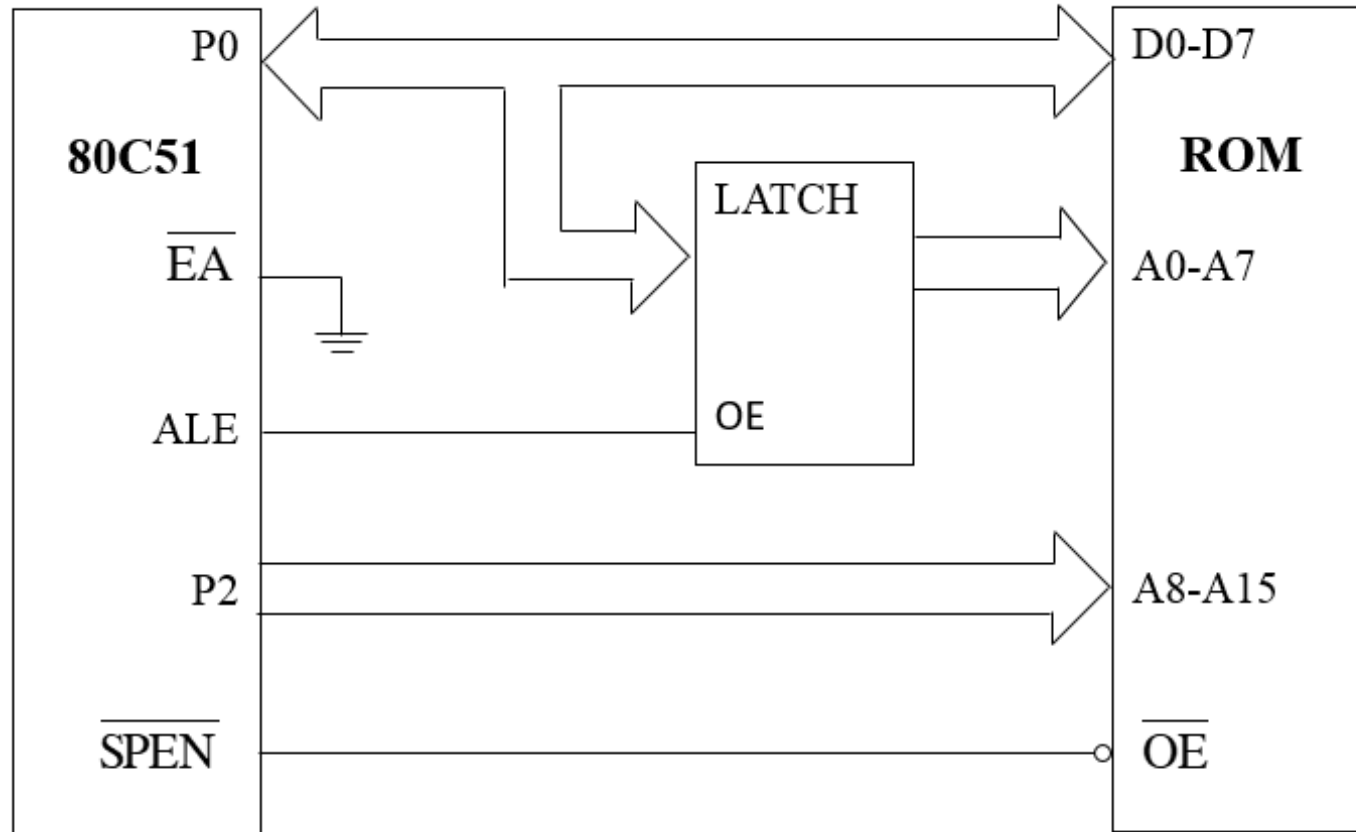


Hình 2.4. Không gian bộ nhớ ROM

8051 có không gian bộ nhớ riêng cho chương trình và dữ liệu. Bộ nhớ chương trình và dữ liệu đặt bên trong chip, tuy nhiên có thể mở rộng bộ nhớ chương trình và dữ liệu bằng các chip nhớ bên ngoài với dung lượng tối đa 64K cho mỗi bộ nhớ. **Không gian bộ nhớ ROM**, Hình 2.4

2.3.2. Tổ chức bộ nhớ ngoài (2)

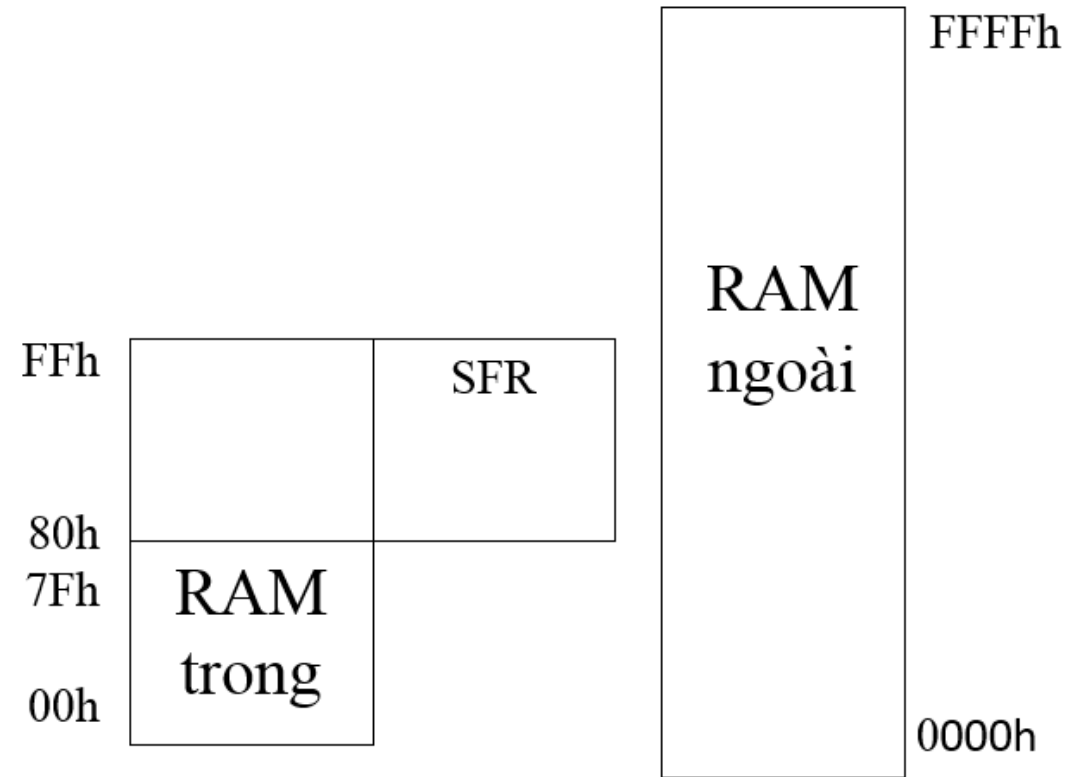
❖ Truy xuất bộ nhớ ROM ngoài, Hình 2.5



Hình 2.5. Truy xuất bộ nhớ ROM ngoài

2.3.2. Tổ chức bộ nhớ ngoài (3)

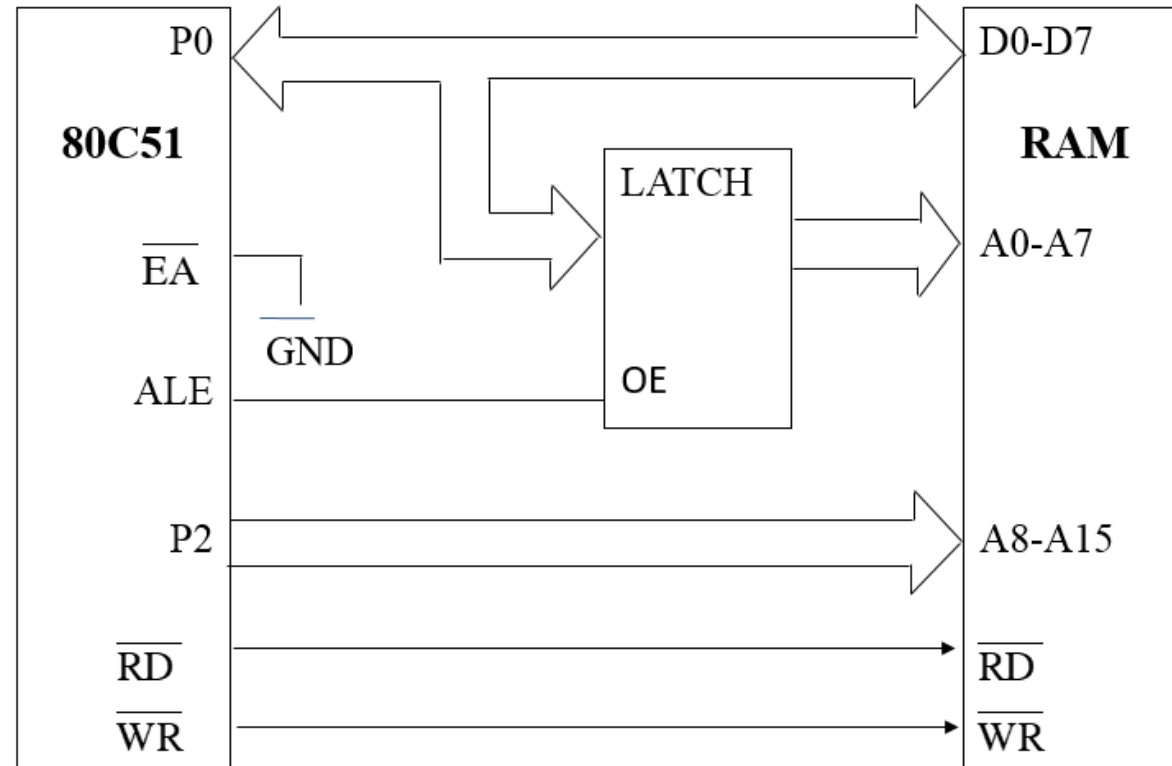
❖ Không gian bộ nhớ RAM, Hình 2.6



Hình 2.6. Không gian bộ nhớ RAM

2.3.2. Tổ chức bộ nhớ ngoài (4)

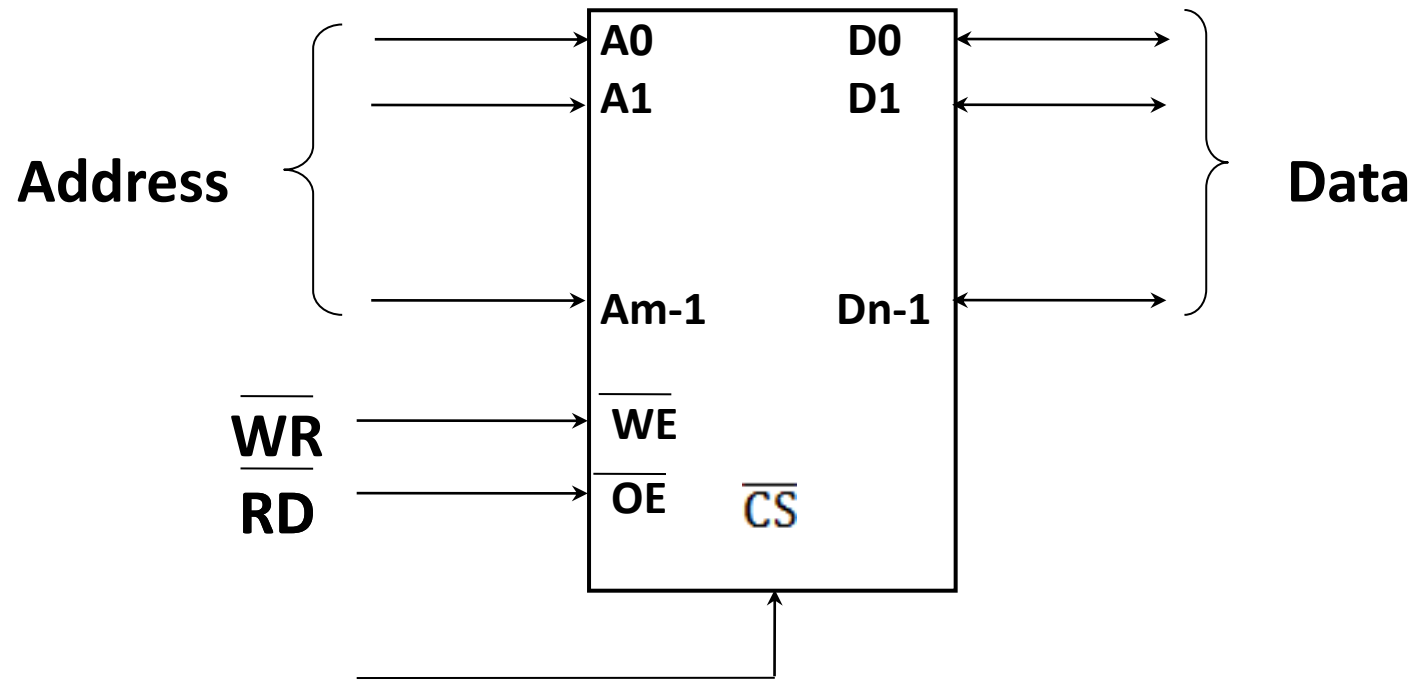
❖ Truy xuất bộ nhớ RAM ngoài, Hình 2.7



Hình 2.7. Truy xuất bộ nhớ RAM ngoài

2.3.2. Tổ chức bộ nhớ ngoài (5)

❖ Cấu trúc chung của IC nhớ



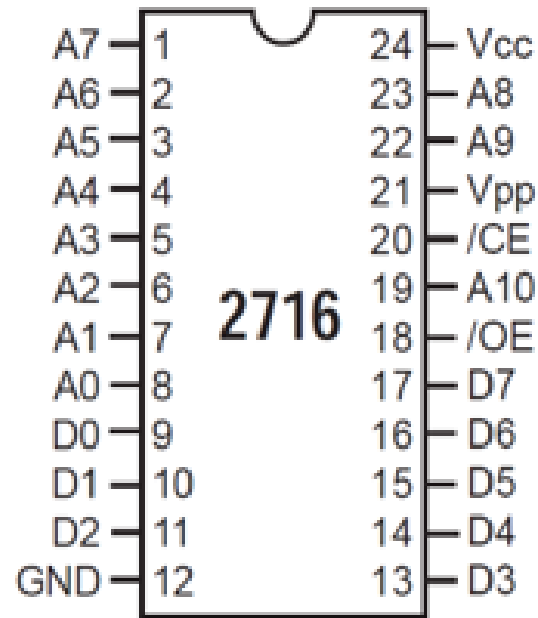
Hình 2.8. Cấu trúc chung của 1 IC nhớ

2.3.2. Tổ chức bộ nhớ ngoài (6)

❖ Một số IC nhớ ROM

Bảng 2.3. Họ EPROM 27x

Số hiệu Chip	Dung lượng
2708	1K*8
2716	2K*8
2732	4K*8
2764	8K*8
27128	16K*8
27256	32K*8
27512	64K*8



- $T_{ac} = 450\text{ns}$
- V_{pp} : Read 5V, Program 25V
- V_{cc} : Power 5V
- GND: Ground
- A0-A10: Address Inputs
- D0-D7: Data Outputs
- \overline{CS} : Chip enable
- \overline{OE} : Output enable

Hình 2.9. Sơ đồ EPROM 2716

2.3.2. Tổ chức bộ nhớ ngoài (7)

❖ Một số IC nhớ RAM

Bảng 2.4. Một số IC nhớ RAM

Số hiệu Chip	Dung lượng
HT 6116	2K*8
CY 6264	8K*8
CY 62256	32K*8
HM 62864	64K*8

- CS: Chip Selet Input
- WR: Output Enabe Input
- I/O1-I/O8: DataInputs/Outputs
- A0-A14: Address Inputs
- OE: Output enable



Hình 2.10. Sơ đồ SRAM 62256

- T_{ac} từ 55-70ns
- VCC: Power (5V)
- VSS: Grond

2.3.3. Ví dụ tổ chức bộ nhớ ngoài – ROM (1)

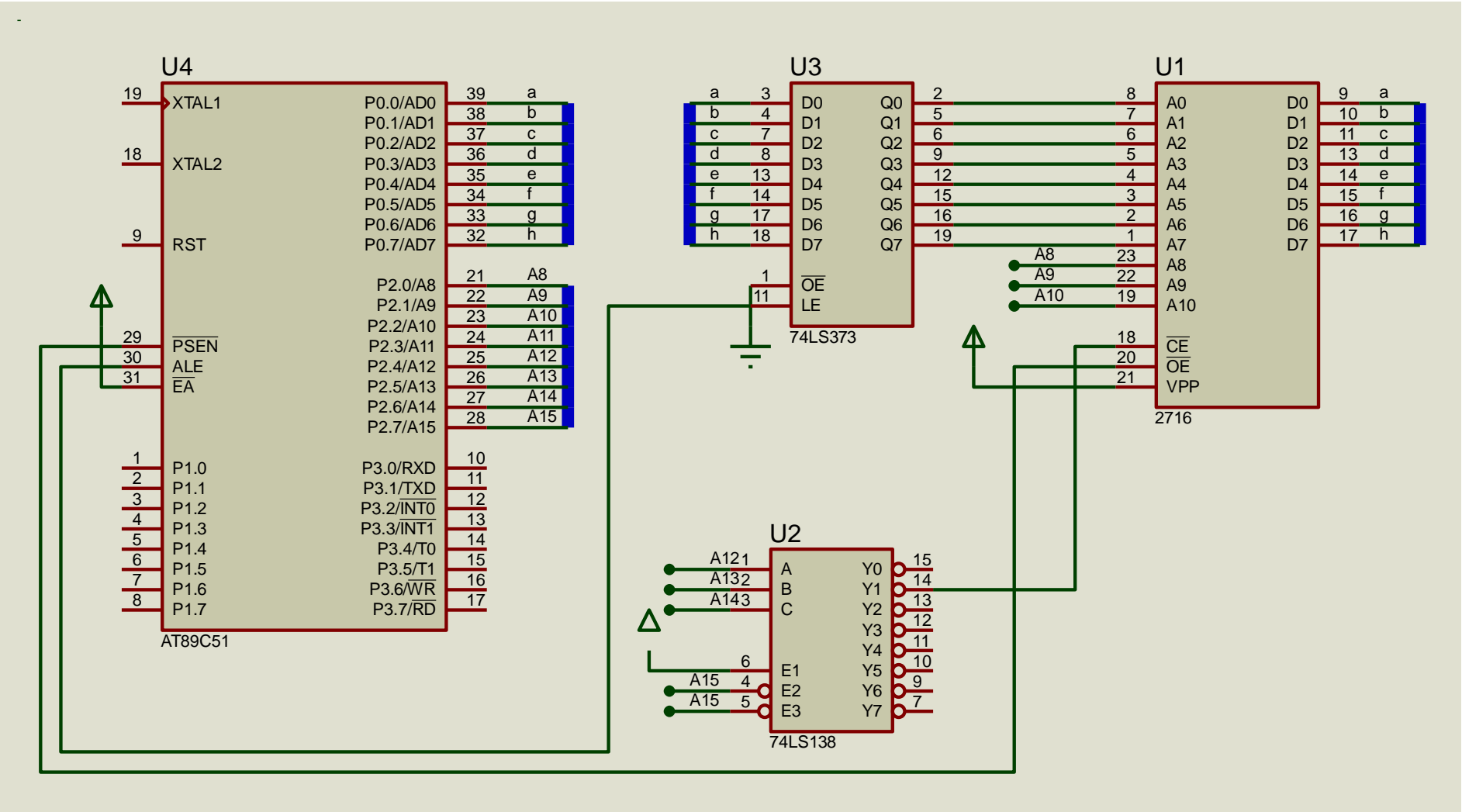
Ví dụ 2.2: Ghép nối 8051 với ROM ngoài dung lượng 2KB bắt đầu từ địa chỉ 1000h

- Bản đồ địa chỉ:

A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1
						
0	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1

2.3.3. Ví dụ tổ chức bộ nhớ ngoài – ROM (2)

- Sơ đồ mạch:



2.3.3. Ví dụ tổ chức bộ nhớ ngoài – RAM (1)

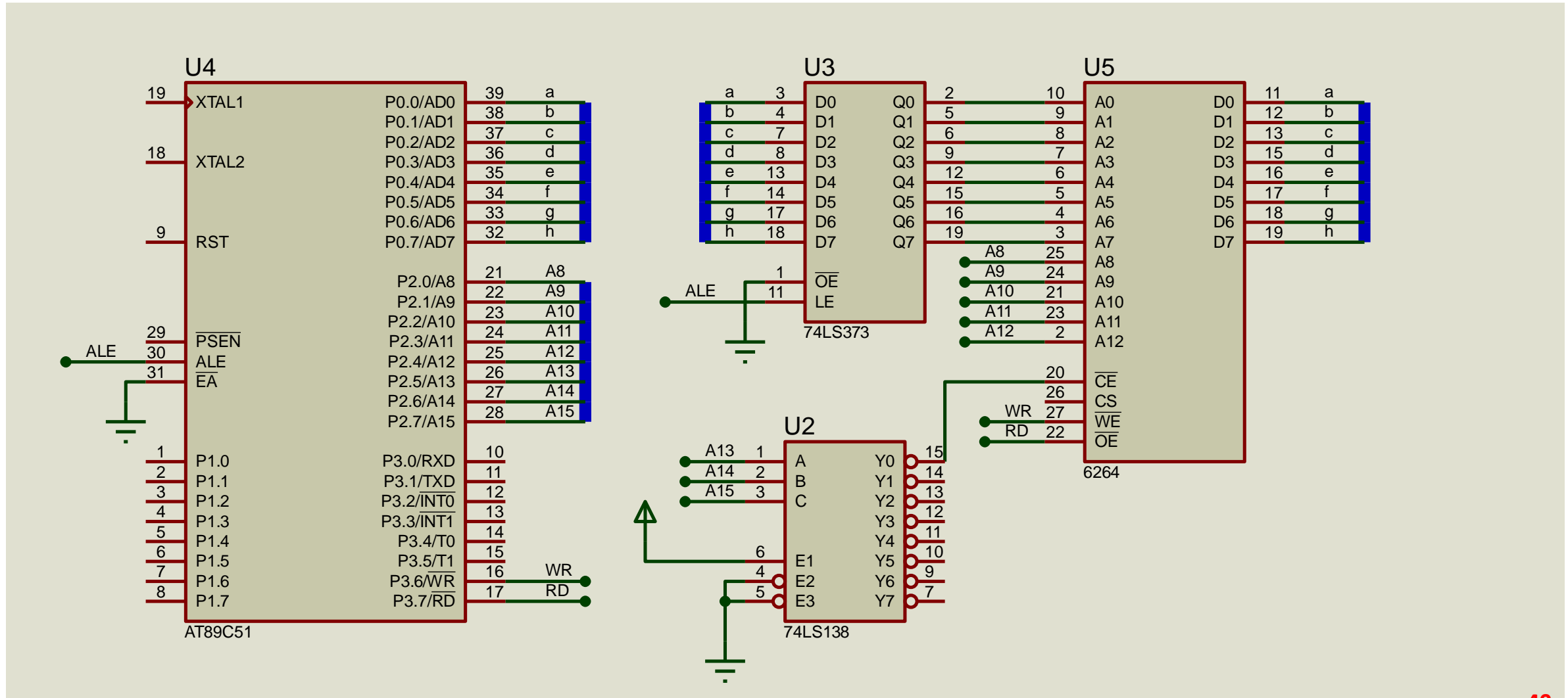
Ví dụ 2.3: ghép nối 8051 với RAM ngoài dung lượng 8KB bắt đầu từ địa chỉ 0000h

- Bản đồ địa chỉ:

A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
						
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1

2.3.3. Ví dụ tổ chức bộ nhớ ngoài – RAM (2)

- Sơ đồ mạch:



2.4. Tập lệnh của 8051

2.4.1. Các thanh ghi chức năng đặc biệt của 8051

2.4.2. Các chế độ địa chỉ

2.4.3. Tập lệnh

2.4.1. Các thanh ghi chức năng đặc biệt của 8051 (1)

❖ Thanh ghi trạng thái chương trình PSW

PSW (Program status word), hay còn gọi là thanh ghi cờ 8 bit, nhưng chỉ có 6 bit được sử dụng. Hai bit cờ chưa dùng là các cờ mà người dùng có thể định nghĩa được. Bốn trong số các cờ là cờ điều kiện, được thiết lập do kết quả thực hiện một lệnh, đó là CY, AC, P và OV, các bit có chức năng được tóm tắt trong Bảng 2.5.

Địa chỉ byte		Địa chỉ bit								
FF										
F0	E7	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	D1	D0		PSW
B8	--	--	--	BC	BB	BA	B9	B8		IP
B0	B7	B6	B5	B4	B3	B2	B1	B0		F3
A8	AF	--	--	AC	AB	AA	A9	A8		IE
A0	A7	A6	A5	A4	A3	A2	A1	A0		P2
99	Không định địa chỉ bit									SBUF
98	9F	9E	9D	9C	9B	9A	99	98		SCON
90	97	96	95	94	93	92	91	90		P1
8D	Không định địa chỉ bit									TH1
8C	Không định địa chỉ bit									TH0
8B	Không định địa chỉ bit									TL1
8A	Không định địa chỉ bit									TL0
89	Không định địa chỉ bit									TMOD
88	8F	8E	8D	8C	8B	8A	89	88		TCON
87	Không định địa chỉ bit									PCON
83	Không định địa chỉ bit									DPH
82	Không định địa chỉ bit									DPL
81	Không định địa chỉ bit									SP
80	87	86	85	84	83	82	81	80		P0
Các thanh ghi chức năng đặc biệt										

2.4.1. Các thanh ghi chức năng đặc biệt của 8051 (2)

Bảng 2.5 Thanh ghi trạng thái chương trình PSW

Bit	Ký hiệu	Địa chỉ	Mô tả bit
PSW.7	CY	D7H	Cờ nhớ chính
PSW.6	AC	D6H	Cờ nhớ phụ
PSW.5	--	D5H	Dành cho người dung sử dụng mục đích chung
PSW.4	RS1	D4H	Chọn các băng thanh ghi
PSW.3	RS0	D3H	
			00 – Băng 0
			01 – băng 1
			10 – Băng 2
			11 – Băng 3
PSW.2	OV	D2H	Cờ tràn
PSW.1	--	D1H	Dành cho người dung định nghĩa
PSW.0	P	D0H	Cờ kiểm tra chẵn lẻ

2.4.1. Các thanh ghi chức năng đặc biệt của 8051 (3)

- ❖ **Thanh chứa A:** Có địa chỉ E0, được sử dụng nhiều nhất, dùng để chứa toán hạng, và lưu kết quả trong các phép toán số học và logic.
- ❖ **Thanh ghi B:** Thanh ghi B có địa chỉ F0h được dùng chung với thanh ghi A trong các phép toán nhân, chia.

Ví dụ 2.4:

MUL AB ; thực hiện nhân không dấu nội dung của A và B,
 ; kết quả 16 bit đặt vào BA (B chứa byte cao,
 ; A chứa byte thấp)

DIV AB ; Chia A cho B, A chứa thương số, B chứa số dư

2.4.1. Các thanh ghi chức năng đặc biệt của 8051 (4)

- ❖ **Con trỏ ngăn xếp:** Thanh ghi con trỏ ngăn xếp (SP) là thanh ghi 8 bit, chứa địa chỉ của đỉnh ngăn xếp. $SP=SP+1$ khi cất dữ liệu vào ngăn xếp; $SP=SP-1$ khi lấy dữ liệu ra từ ngăn xếp.
- ❖ **Con trỏ dữ liệu DPTR:** DPTR (data pointer) được sử dụng để truy xuất bộ nhớ chương trình, hoặc dữ liệu ngoài. DPTR là thanh ghi 16 bit, thanh ghi này được chia là 02 phần: DPL (byte thấp - địa chỉ 82H), DPH (byte cao – địa chỉ 83H)

Ví dụ 2.5:

```
MOV  A, #55H
```

```
MOV  DPTR, #1000H
```

```
MOV  @DPTR, A
```

Đoạn chương trình nạp giá trị 55H vào ô nhớ ngoài có địa chỉ 1000H

2.4.1. Các thanh ghi chức năng đặc biệt của 8051 (5)

❖ Các thanh ghi định thời/bộ đếm

8051 có 02 định thời/bộ đếm (timer/counter) 16 bit để định khoảng thời gian hoặc đếm các sự kiện.

- Bộ định thời 0: TL0 – byte thấp, địa chỉ 8AH; TH0 – byte cao, địa chỉ 8CH
- Bộ định thời 1: TL1 – byte thấp, địa chỉ 8BH; TH1 – byte cao, địa chỉ 8DH

Hoạt động của bộ định thời được thiết lập bởi thanh ghi chế độ định thời TMOD (time mode register) và thanh ghi điều khiển định thời TCON (timer control register). Các thanh ghi này sẽ thảo luận ở phần sau.

2.4.1. Các thanh ghi chức năng đặc biệt của 8051 (6)

❖ **Các thanh ghi cổng nối tiếp:** 8051 có một cổng nối tiếp để truyền thông với các thiết bị nối tiếp.

- Thanh ghi cấu hình cổng nối tiếp SCON
- Thanh ghi SBUF (serial data buffer) được sử dụng để lưu dữ liệu truyền đi, hoặc nhận về
(hoạt động chi tiết được thảo luận ở phần sau)

❖ **Thanh ghi ngắt**

8051 có một cấu trúc ngắt với 2 mức ưu tiên và 5 nguyên nhân ngắt (5 source, 2 priority level interrupt structure). Các ngắt bị vô hiệu hóa sau khi reset hệ thống, sau đó được cho phép bằng cách ghi vào thanh ghi cho phép ngắt IE (interrupt enable register).

(Hoạt động chi tiết được thảo luận ở phần sau)

❖ **Thanh ghi điều khiển nguồn PCON** (power control register) cho phép tăng tốc độ của cổng nối tiếp

2.4.2. Các chế độ địa chỉ (1)

❖ Chế độ địa chỉ tức thời

- Toán hạng nguồn là hằng số
- Trước dữ liệu tức thời cần có dấu #

Ví dụ 2.6: MOV A, #25; Nạp giá trị 25 và thanh ghi A

❖ Chế độ định địa chỉ thanh ghi

- Sử dụng thanh ghi để lưu trữ dữ liệu cần thao tác
- Thanh ghi nguồn và đích phải phù hợp về kích thước
- Không được chuyển dữ liệu giữa các thanh ghi Rn vào Rm

Ví dụ 2.7:

MOV A, R0 ; Chuyển dữ liệu trong R0 vào A

MOV DPTR, A ; không thực hiện được, vì không cùng kích thước

MOV R1, R2 ; không hợp lệ

2.4.2. Các chế độ địa chỉ (2)

❖ Chế độ địa chỉ trực tiếp

- Toán hạng (nguồn hoặc đích) là địa chỉ của ô nhớ
- Trước địa chỉ ô nhớ không có dấu #

Ví dụ 2.8:

MOV A, 35h ; DL trong ô nhớ có đ/c 35h được chuyển vào thanh ghi A

MOV 56h, A ; chuyển nd thanh ghi A vào ô nhớ có địa 56h

❖ Chế độ định địa chỉ gián tiếp thanh ghi

- Địa chỉ ô nhớ chứa dữ liệu được chứa trong thanh ghi R0 hoặc R1
- Trước thanh ghi R0 hoặc R1 phải chèn thêm ký tự “@” để biểu thị cho chế độ địa chỉ này

Ví dụ 2.9:

MOV A, @R0 ;chuyển dl trong ô nhớ có đ/c trong R0 vào A

MOV @R1, A ; chuyển nd A vào ô nhớ có đ/c trong R1

Bài tập

Câu 1: 0E0H trong câu lệnh JB 0E0H, LP là chỉ:

- A. Thanh ghi tích lũy A
- B. Bit cao nhất của thanh ghi tích lũy A
- C. Bit thấp nhất của thanh ghi tích lũy A
- D. 1 địa chỉ của ô nhớ

Bài tập

- Câu 2: Trước khi thực hiện lệnh `MOV R0, #20H` biết $(R_0)=30H$, $(20H)=38H$, sau khi thực hiện lệnh $(R_0) =$
 - A. 00H
 - B. 20H
 - C. 30H
 - D. 38H

Bài tập

- Câu 3: Trước khi thực hiện lệnh MOV R₀, 20H biết (R₀)=30H, (20H)=38H, sau khi thực hiện lệnh (R₀) =
 - A. 00H
 - B. 20H
 - C. 30H
 - D. 38H

Câu 4: Thực hiện 3 dòng lệnh dưới đây, thì nội dung của ô nhớ 30H là:

MOV R₁, #30H

MOV 40H, #0EH

MOV @R₁, 40H

A. 40H

B. 0EH

C. 30H

D. FFH

2.4.3. Tập lệnh

- ❖ Cấu trúc chung một lệnh
- ❖ Lệnh chuyển dữ liệu
- ❖ Lệnh số học
- ❖ Lệnh logic
- ❖ Xử lý bit
- ❖ Lệnh rẽ nhánh

❖ Cấu trúc chung một lệnh

[label:] <mã lệnh> [toán hạng 1][,toán hạng 2],[toán hạng 3] [;chú giải]

- <mã lệnh> bắt buộc phải có
- Các trường khác có thể có hoặc không (tùy từng lệnh)
- Các trường cách nhau bởi dấu “cách” hoặc “tab”
- Label phải kết thúc bởi dấu “:”, có độ dài tối đa 31 ký tự gồm các chữ cái, số, dấu “?”, “_”, không được trùng với mã lệnh, bắt đầu bằng các chữ cái.
- Chú giải bắt đầu bằng “;”

❖ **Lệnh chuyển dữ liệu (1)**

• **MOV**

Cú pháp: MOV đích, nguồn ; thực hiện chuyển nguồn vào đích

- Các giá trị có thể nạp trực tiếp vào bất kỳ thanh ghi nào A, B, R0-R7, giá trị phải có dấu “#” ở trước
- Giá trị bắt đầu là chữ cái phải thêm số 0 đằng trước.
- Đích, nguồn phải có cùng độ dài.

Ví dụ 2.10:

MOV A, #12H

MOV R1, #15H

MOV B, #0BH

MOV R2, #123H ; không hợp lệ

MOV A, 35h ; chuyển nội dung ô nhớ có đ/c 35h vào thanh ghi A

MOV @R1, A ; chuyển nội dung thanh ghi A vào ô nhớ có đ/c trong R1

❖ **Lệnh chuyển dữ liệu (2)**

• **MOVC**

Cú pháp:

MOVC A, @A + DPTR ; chuyển vào A một byte từ bộ nhớ chương trình tại vị trí
; cách con trỏ gốc DPTR một khoảng là nội dung của A

MOVC A, @A + PC ; chuyển vào A một byte từ bộ nhớ chương trình tại vị trí
; cách vị trí lệnh hiện hành một khoảng là nội dung của A

• **MOVX**

Cú pháp:

MOVX A, @Ri

MOVX A, @DPTR

MOVX @Ri, A

MOVX @DPTR, A

❖ **Lệnh chuyển dữ liệu (3)**

- **PUSH**

Cú pháp: PUSH trực tiếp

PUSH ACC

PUSH R1

- **POP**

Cú pháp: POP trực tiếp

POP R1

POP ACC

❖ Lệnh chuyển dữ liệu (4)

- **XCH**

Cú pháp: XCH A, nguồn ; đổi nội dung thanh ghi A với nguồn (nguồn là thanh ghi (Ri) hay ô nhớ)

Ví dụ 2.11:

XCH A, R7

XCH A, 60H

XCH A, @R0

- **XCHD**

Cú pháp: XCHD A, @Ri ; hoán đổi 4 bit thấp của A với 4 bit thấp của ô nhớ được trỏ bởi Ri

Ví dụ 2.12:

MOV A, #0F3h ; A = 1111 0011

MOV R1, #40h ; R1 = 40h

MOV @R1, #5Bh ; ô nhớ = 0101 1011

XCHD A, @R1 ; A = 1111 1011; ô nhớ địa chỉ 40h: 0101 0011

❖ Lệnh số học (1)

- **ADD:** cộng các số không nhớ

Cú pháp: ADD A, nguồn ; $A = A + \text{nguồn}$

➤ **Lưu ý:**

- Toán hạng đích luôn là thanh ghi A, toán hạng nguồn có thể là dữ liệu, thanh ghi hay ô nhớ
- Ảnh hưởng tới các cờ: AF, CF, PF

Ví dụ 2.13:

MOV A, #0F5h ; A= 1111 0101B

ADD A, #0Bh ; A= 1111 0101B

; + 0000 1011B

; = 0000 0000B, CF=1, AF=1, PF=1

Ví dụ 2.14:

MOV R0, #40h

MOV A, #5

ADD A, @R0

❖ Lệnh số học (2)

- **ADDC**: cộng các số có nhớ

Cú pháp: ADDC A, nguồn ; $A = A + \text{nguồn} + CF$

➤ **Lưu ý:**

- Toán hạng đích luôn là thanh ghi A, toán hạng nguồn có thể là dữ liệu, thanh ghi hay ô nhớ
- Ảnh hưởng tới các cờ: AF, CF, PF

Ví dụ 2.15: cộng 2 số 16 bit 3CE7H + 3B8DH

MOV	A, #0E7H	; nạp 0E7H (byte thấp) vào A
ADD	A, #8DH	; $A = 0E7H + 8DH = 00H$, CF=1
MOV	R6, A	; nạp A (byte thấp của tổng) vào R6
MOV	A, 3CH	; nạp 3CH (byte cao) vào A
ADDC	A, 3BH	; $A = 3CH + 3BH + CF = 78H$
MOV	R7, A	; nạp A (byte cao của tổng) vào R6

❖ Lệnh số học (3)

- **DA**: hiệu chỉnh nội dung thanh ghi A sau khi cộng các số ở mã BCD
 - Nếu giá trị 4 bit thấp >9 hoặc $AF=1$, cộng 06H vào A
 - Nếu giá trị 4 bit cao >9 hoặc $CF=1$, cộng 60H vào A

Ví dụ 2.16:

```
MOV A, #29H ; A=0010 1001
ADD A, #18H ; A=0010 1001
               ; + 0001 1000
               ; = 0100 0001=41H, AF=1, CF=0
DA           ; A=A+06H=41h+06H=47H
```

Ví dụ 2.17:

```
MOV A, #35H ; A=0011 0101
ADD A, #28H ; A=0011 0101
               ; + 0010 1000
               ; = 0101 1101=5DH, AF=0, 4 bit thấp = 0Dh=13>9
DA           ; A=A+06H=5DH+06H=63H
```

Ví dụ 2.18:

```
MOV A, #35H ; A=0011 0101
ADD A, #78H ; A=0011 0101
               ; + 0111 1000
               ; = 1010 1101=0ADH, 4 bit cao = 0A=10>9 4 bit thấp = 0Dh=13>9
DA           ; A=A+66H=0ADH+66H=13H, CF=1
```

❖ Lệnh số học (4)

- **SUBB**: trừ các số

Cú pháp: SUBB A, nguồn ; A=A-nguồn-CF

Quá trình thực hiện:

- Lấy **bù 2** của toán hạng nguồn
- Cộng nội dung của A với giá trị bù 2 vừa xác định (*kết quả ở dạng bù 2*)
- Đảo cờ nhớ (CF=0 số dương; CF=1 số âm)

Ví dụ 2.19:

```
MOV    A, #3Fh ; A      = 0011 1111
MOV    R3, #23h ; R3     = 0010 0011
SUBB   A, R3    ; A      = 0011 1111
        ; +      1101 1101 (bù 2 của 23h)
        ; =      0001 1100 (bù 2), CF=1
        ; =      CF=0 (đảo cờ nhớ), số dương
        ; kết quả A=0001 1100 = 1Ch
```

Ví dụ 2.20:

```
MOV    A, #15h ; A      = 0001 0101
MOV    R3, #27h ; R3     = 0010 0111
SUBB   A, R3    ; A      = 0001 0101
        ; +      1101 1001 (bù 2 của 27h)
        ; =      1110 1110 (bù 2), CF=0
        ; =      CF=1 (đảo cờ nhớ), số âm
        ; kết quả A=0001 0010 = -12h
```

❖ Lệnh số học (5)

- **MUL**: nhân 2 số 8 bit

Cú pháp: MUL AB ; A*B kết quả 16 bit, B chứa byte cao, A chứa byte thấp

Ví dụ 2.21:

MOV A, #25H

MOV B, #65H

MUL AB ; 25H*56H=0E99h, B=0EH, A=99H

Ví dụ 2.22:

MOV A, #95

MOV B, #10

DIV AB ; A=09 (thương số), B=05 (số dư)

❖ Lệnh logic

ANL đích, nguồn ; đích = đích AND nguồn

ORL đích, nguồn ; đích = đích OR nguồn

XRL A, nguồn ; đích = A XOR nguồn (nguồn: thanh ghi, ô nhớ, giá trị)

CPL A ; lấy bù nội dung của A

Ví dụ 2.23:

MOV A, #55h ; A = 0101 0101B

CPL A ; A = 1010 1010B

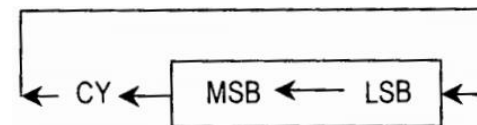
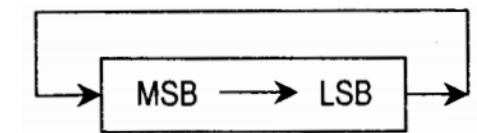
RR A ; quay phải A đi 01 bit

RL A ; quay trái A đi 01 bit

RRC A ; quay phải A 01 bit quan cờ CF

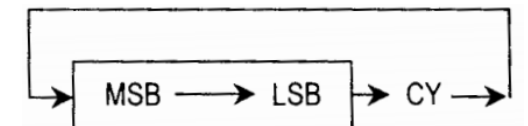
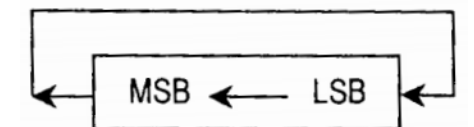
RLC A ; quay trái A 01 bit quan cờ CF

SWAP A ; hoán đổi nội dung thanh ghi A



Trước D7 - D4 D3 - D0

Sau D3 - D0 D3 - D0



❖ Lệnh xử lý bit (1)

Lệnh	Chức năng
SETB bit	Thiết lập bit (bit = 1)
CLR bit	Xóa bit về 0 (bit = 0)
CPL bit	Bù bit (bit = NOT bit)
JB bit, label	Nhảy tới label nếu bit = 1
JNB bit, label	Nhảy tới label nếu bit = 0
JBC bit, label	Nhảy tới label nếu bit = 1 và sau đó xóa bit

❖ Lệnh xử lý bit (2)

Ví dụ 2.24: tạo dãy xung vuông tại P1.0

```
HERE:  SETB    P1.0    ; thiết lập P1.0 lên 1
        ACALL   DELAY
        CLR     P1.0
        SJMP    HERE
```

Hoặc có thể viết

```
HERE:  CPL     P1.0    ; lấy bù bit 0 của P1.0
        ACALL   DELAY
        SJMP    HERE
```

Ví dụ 2.25: Nhận dữ liệu từ cổng P1 và kiểm tra bit P1.0. Nếu $P1.0 = 0$ gửi nội dung thanh ghi R0 ra cổng P2, nếu $P1.0 = 1$ gửi nội dung thanh ghi R1 ra cổng P2

```
        MOV     A, P1
        JNB     ACC.0, NEXT1
        MOV     P2, R1
        SJMP    NEXT2
NEXT1:
        MOV     P2, R0
NEXT2:
```

Phần mềm



Phần cứng



❖ Lệnh rẽ nhánh (1)

Lệnh	Ý nghĩa
JZ label	Nhảy tới <i>label</i> nếu $A = 0$
JNZ label	Nhảy tới <i>label</i> nếu $A \neq 0$
DJNZ Rn, label	Nhảy tới <i>label</i> nếu $Rn \neq 0$, $Rn = Rn - 1$
DJNZ direct, label	Nhảy tới <i>label</i> nếu nội dung ô nhớ có địa chỉ direct $\neq 0$, [direct]=[direct]-1
CJNE A, direct, label	Nhảy tới <i>label</i> nếu nội dung của A \neq [direct], CF=0 nếu $A \geq$ [direct], CF=1 nếu $A <$ [direct]
CJNE Reg, #data, label (Reg: A, Rn)	Nhảy tới <i>label</i> nếu nội dung của Reg \neq dulieu, CF=0 nếu $Reg \geq$ dulieu, CF=1 nếu $Reg <$ dulieu
JC label	Nhảy tới <i>label</i> nếu CF=1
JNC label	Nhảy tới <i>label</i> nếu CF \neq 0

❖ Lệnh rẽ nhánh (2)

Lệnh	Ý nghĩa
SJMP label	Nhảy không điều kiện tới label (trong không gian 128 byte)
LJMP label	Nhảy không điều kiện tới label (trong không gian 64 KB)
ACALL ctc	Gọi chương trình con trong phạm vi 2KB
LCALL ctc	Gọi chương trình con trong phạm vi 64KB
RET	Kết thúc chương trình con

2.5. Khung chương trình hợp ngữ 8051 (1)

❖ Một số chỉ dẫn

- **ORG** (origin): bắt đầu chương trình, Ví dụ:

ORG 0000h

- **EQU** (Equate): được sử dụng định nghĩa một hằng số, Ví dụ:

count EQU 25

.....

MOV R3, count

- **BIT**: được sử dụng định nghĩa một bit, Ví dụ:

WR BIT P2.2

.....

SETB WR

- **END**: chỉ báo kết thúc trình hợp dịch

2.5. Khung chương trình hợp ngữ 8051 (1)

- **Khung chương trình**

ORG 0000h

- khai báo hằng (nếu có)
- LJMP địa_chỉ_ctrình_chính

;-----

ORG địa_chỉ_ngắt (nếu chương trình sử dụng ngắt)

- các câu lệnh

RETI

;-----

ORG địa_chỉ_ctrình_chính

MAIN:

- ;bắt đầu chương trình chính
- các câu lệnh

RET

;-----

Ct_con1:

.....

Ret

CT_con2:



.....

Ret



END

MỘT SỐ VÍ DỤ LẬP TRÌNH HỢP NGỮ 8051



Ví dụ 2.26: Ghép nối 8051 với LED đơn thông qua cổng P2, lập trình sáng tuần tự các LED

- Phần mềm: 
- Phần cứng: 

Ví dụ 2.27: Sử dụng cổng P2 của 8051 ghép nối với 01 LED 7 đoạn, lập trình hiển thị liên tục các số từ 0-9

- Phần mềm: 
- Phần cứng: 

Ví dụ 2.28: Sử dụng cổng P2 của 8051 ghép nối với 02 LED 7 đoạn, lập trình hiển thị liên tục các số từ 00-99.

- Phần mềm: 
- Phần cứng: 



Chương 3.

BỘ ĐẾM/ĐỊNH THỜI VÀ TRUYỀN TIN NỐI TIẾP TRONG 8051

MỤC TIÊU

- Hiểu được nguyên lý làm việc của bộ đếm/bộ định thời, lập trình bộ đếm/bộ định thời.
- Hiểu được nguyên lý truyền thông nối tiếp UART, lập trình truyền thông nối tiếp qua UART

NỘI DUNG

3.1. Nguyên lý hoạt động cơ bản của bộ định thời/bộ đếm

3.2. Các thanh ghi dùng cho bộ định thời/bộ đếm

3.3. Các chế độ định thời

3.4. Lập trình cho bộ định thời/bộ đếm

3.5. Truyền thông nối tiếp trong 8051

3.1. Nguyên lý hoạt động cơ bản của bộ định thời/bộ đếm

Một bộ định thời là một chuỗi các Flip-Flop, với mỗi Flip-Flop là mạch chia 2. Chuỗi này nhận một tín hiệu ngõ vào làm nguồn xung *clock*. Xung *clock* đặt vào Flip-Flop thứ nhất, Flip-Flop chia đôi tần số xung *clock*. Ngõ ra của Flip-Flop thứ nhất là nguồn xung *clock* cho Flip-Flop thứ 2, nguồn xung này cũng được chia cho 2,... Vì mỗi một tầng kế tiếp nhau đều chia cho 2 nên một bộ định thời có n tầng sẽ chia tần số xung *clock* ở ngõ vào của bộ định thời này cho 2^n

Ngõ ra của tầng cuối cùng làm xung *clock* cho một Flip-Flop báo tràn bộ định thời (còn tràn – OF). OF được kiểm tra bằng phần mềm, hoặc tạo ra một ngắt. Giá trị nhị phân trong các Flip-Flop của bộ định thời là số đếm của các xung *clock* từ khi bộ định thời bắt đầu đếm.

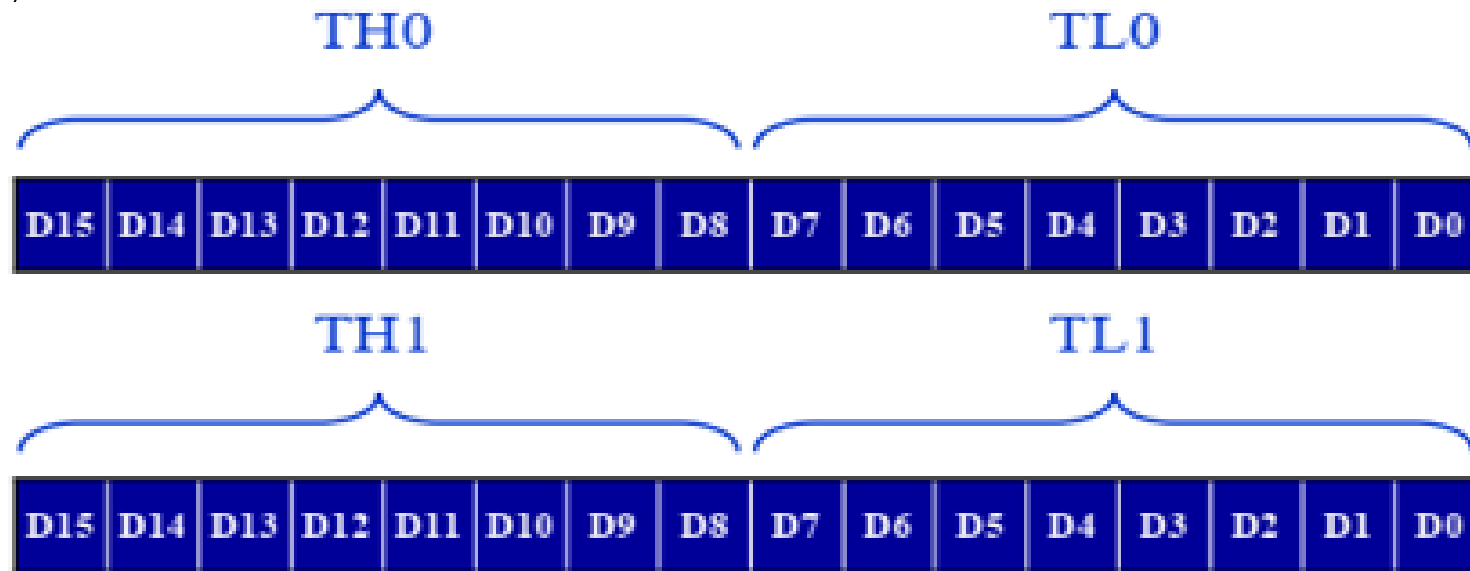
8051 có 02 bộ định thời/bộ đếm (Timer/Counter 0, Timer/Counter 1). Chúng được sử dụng bộ đếm để đếm các sự kiện hoặc làm bộ định thời để tạo trễ thời gian.

3.2. Các thanh ghi dùng cho bộ định thời/bộ đếm - **Thanh ghi chứa giá trị đếm**

Mỗi bộ định thời/bộ đếm trong 8051 có 1 thanh ghi 16 bit chứa giá trị đếm, được truy cập (đọc/ghi) dưới 2 thanh ghi 8 bit độc lập gọi là byte thấp (TLx), byte cao (THx), Hình 3.1. Các thanh ghi này được truy cập như các thanh ghi khác (A, B, R0, R1,...). Ví dụ:

```
MOV TL0, #4Fh
```

```
MOV R5, TH0
```



Hình 3.1. Thanh ghi của bộ định thời/bộ đếm

3.2. Các thanh ghi dùng cho bộ định thời/bộ đếm –**TMOD (1)**

Cả hai bộ đếm/định thời đều dùng chung một thanh ghi chế độ định thời (TMOD - timer mode register) để thiết lập các chế độ làm việc khác nhau. TMOD là thanh ghi 8 bit, 4 bit thấp sử dụng để thiết lập chế độ cho bộ định thời/bộ đếm 0; 4 bit cao được sử dụng để thiết lập chế độ hoạt động cho bộ định thời/bộ đếm 1, Hình 3.2

MSB				LSB			
GATE	C/T	M1	M0	GATE	C/T	M1	M0
Timer1				Timer0			

Hình 3.2. Thanh ghi TMOD

- **C/T: bit chọn chức năng đếm hoặc định thời**
 - C/T =0 định thời, hoạt động dựa vào xung clock nội bên trong chip. Tần số clock của bộ định thời = 1/12 của tần số thạch anh.
 - C/T = 1 đếm sự kiện. bộ định thời/bộ đếm hoạt động dựa vào xung clock cấp từ bên ngoài đưa vào trên chân T0 (với bộ định thời/bộ đếm 0), chân T1 (với bộ định thời/bộ đếm 1).

3.2. Các thanh ghi dùng cho bộ định thời/bộ đếm –**TMOD** (2)

MSB				LSB			
GATE	C/T	M1	M0	GATE	C/T	M1	M0
Timer1				Timer0			

Hình 3.2. Thanh ghi TMOD

GATE: Điều khiển khởi động/dừng hoạt động của bộ định thời/bộ đếm

- GATE = 0. Khởi động bộ định thời/bộ đếm bằng phần mềm bằng cách set bit TRx lên 1. (TR0 cho bộ đếm/bộ định thời 0, TR1 cho bộ định thời/bộ đếm 1). bộ định thời/bộ đếm dừng hoạt động nếu bit TRx được xóa về 0.
- GATE = 1. Khởi động bộ định thời/bộ đếm bằng phần cứng (đề cập trong nội dung về ngắt)

3.2. Các thanh ghi dùng cho bộ định thời/bộ đếm –**TMOD (3)**

MSB				LSB			
GATE	C/T	M1	M0	GATE	C/T	M1	M0
Timer1				Timer0			

Hình 3.2. Thanh ghi TMOD

- **M1, M0:** thiết lập chế độ hoạt động cho các bộ định thời/bộ đếm, Bảng 3.

Bảng 3.1. Các chế độ định thời của 8051.

M1	M0	Chế độ	Chế độ hoạt động
0	0	0	Bộ định thời 13 bit gồm 8 bit là bộ định thời/ bộ đếm 5 bit đặt trước
0	1	1	Bộ định thời 16 bit (không có đặt trước)
1	0	2	Bộ định thời 8 bit tự nạp lại
1	1	3	Chế độ bộ định thời chia tách

3.2. Các thanh ghi dùng cho bộ định thời/bộ đếm – TCON

7	6	5	4	3	2	1	0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

Thanh ghi điều khiển bộ định thời/bộ đếm (TCON) chứa các bit điều khiển và trạng thái của bộ định thời/bộ đếm 0 và 1. Bốn bit cao (TCON.4 đến TCON.7) được sử dụng để điều khiển khởi động/dừng hoặc báo các tràn đối bộ định thời/bộ đếm, *Bảng 3.2*. Bốn bit thấp (TCON.0 đến TCON.3) không dùng cho bộ định thời/bộ đếm (*đề cập tới trong nội dung ngắt*)

Bảng 3.2. Bốn bit cao thanh ghi TCON

Bit	Ký hiệu	Địa chỉ bit	Mô tả
TCON.7	TF1	8Fh	Báo tràn cho bộ định thời/bộ đếm 1. Khi giá trị trong thi ghi chứa giá trị đếm TL1, TH1 bị tràn, thì bit TF1 sẽ được bật lên 1.
TCON.6	TR1	8Eh	Được sử dụng để khởi động hay dừng bộ định thời/bộ đếm 1. TR1 =1: khởi động. TR1 = 0: dừng
TCON.5	TF0	8Dh	Báo tràn cho bộ định thời/bộ đếm 0
TCON.4	TR0	8Ch	Được sử dụng để khởi động hay dừng bộ định thời/bộ đếm 0.

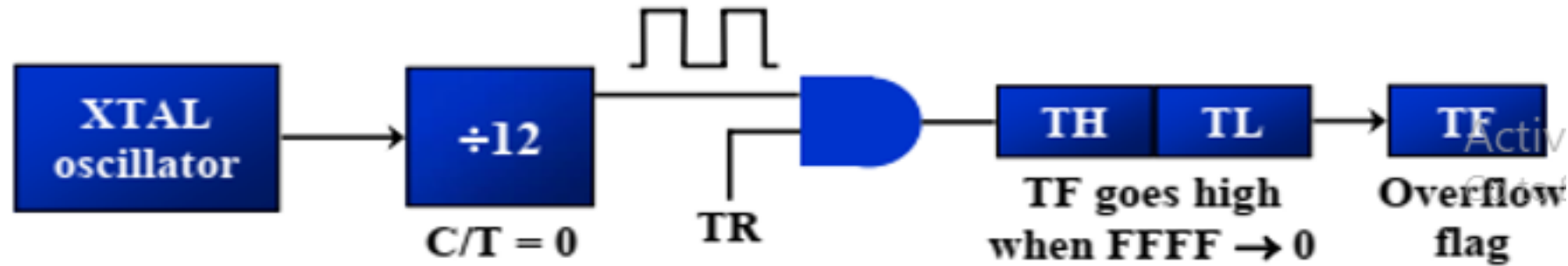
3.3. Các chế độ định thời

Có 4 chế độ định thời (Bảng 3.1). Chúng ta chỉ tập trung vào các chế độ được sử dụng rộng rãi là chế độ 1 và 2.

Bảng 3.1. Các chế độ định thời của 8051.

M1	M0	Chế độ	Chế độ hoạt động
0	0	0	Bộ định thời 13 bit gồm 8 bit là bộ định thời/ bộ đếm 5 bit đặt trước
0	1	1	Bộ định thời 16 bit (không có đặt trước)
1	0	2	Bộ định thời 8 bit tự nạp lại
1	1	3	Chế độ bộ định thời chia tách

3.4. Lập trình cho bộ định thời/bộ đếm - **Lập trình định thời Chế độ 1 (1)**



Bộ định thời 16 bit, các giá trị trong khoảng từ 0000h đến FFFFh có thể được nạp vào TL và TH của bộ định thời

- 1) Sau khi TL và TH được nạp, thực hiện khởi động bộ định thời bằng lệnh “SETB TR0” cho bộ Timer 0 và “SETB TR1” cho Timer 1
- 2) Sau khi được khởi động bắt đầu đếm tăng, sau khi đạt tới giá trị FFFFh bộ định thời sẽ quay về 0000h và lập cờ TF (Timer Flage) =1. Khi cờ bộ định thời được thiết lập, để dừng bộ định thời bằng phần mềm sử dụng lệnh “CLR TR0” cho Timer 0, “CLR TR1” cho Timer 1.
- 3) Bộ định thời sau khi đạt giá trị giới hạn thì thực hiện quay vòng về 0. Để lặp lại quá trình đếm của bộ định thời, các thanh ghi TL, TH phải được nạp lại giá trị ban đầu và TF cần được xóa về 0.

3.4. Lập trình cho bộ định thời/bộ đếm - **Lập trình định thời Chế độ 1 (2)**

❖ Các bước lập trình

- 1) Nạp giá trị cho thanh ghi TMOD xác định bộ Timer 0 hay Timer 1 và chế độ nào được chọn.
- 2) Nạp giá trị đếm ban đầu cho các thanh ghi TL, TH
- 3) Khởi động bộ định thời
- 4) Kiểm tra trạng thái bật của bộ định thời TF bằng lệnh “JNB TFX, đích”. Thoát khỏi vòng lặp khi TF được bật lên cao
- 5) Dừng bộ định thời
- 6) Xóa cờ TF cho vòng kế tiếp.
- 7) Quay trở lại bước 2 để nạp lại TL, TH

3.4. Lập trình cho bộ định thời/bộ đếm - **Lập trình định thời Chế độ 1 (3)**

Ví dụ 3.1: Xác định tần số clock và chu kỳ của bộ định thời/bộ đếm của 8051 với các tần số thạch anh: 12MHz, 16MHz, 11,0592MHz



- $\frac{1}{12} \times 12\text{MHz} = 1\text{MHz}$ và $T = \frac{1}{1\text{MHz}} = 1\mu\text{s}$

- $\frac{1}{12} \times 16\text{MHz} = 1,333\text{MHz}$ và $T = \frac{1}{1,333\text{MHz}} = 0,75\mu\text{s}$

- $\frac{1}{12} \times 11,0592\text{MHz} = 0,9216\text{MHz}$ và $T = \frac{1}{0,9216\text{MHz}} = 1,085\mu\text{s}$

3.4. Lập trình cho bộ định thời/bộ đếm - **Lập trình định thời Chế độ 1 (4)**

Ví dụ 3.2. Tính thời gian trễ trong đoạn chương trình sau, giả thiết tần số của 8051 XTAL = 12MHz

```
MOV    TMOD, #01      ; chọn Timer 0 chế độ 1 (16 bit)
MOV    TL0, #0F2h
MOV    TH0, #0FFh
DELAY:
SETB   TR0             ; khởi động Timer 0
AGAIN: JNB   TF0, AGAIN ; kiểm tra cờ Timer 0
CLR    TR0             ; dừng Timer 0
CLR    TF0             ; xóa cờ Timer 0
RET
```

Giải:

- Theo **Ví dụ 3.1**, chu kỳ xung đồng hồ của Timer 0 là $1\mu s$. Như vậy bộ Timer 0 đếm tăng sau mỗi $1\mu s$, ta có **Độ trễ** = **Số đếm** * $1\mu s$.
- **Số đếm** = $(FFFFh - FFF2h) + 1 = 0Dh + 1 = 14$ (cộng thêm 1 vì, cần thêm một nhịp đồng hồ để thực hiện quay vòng từ FFFFh về 0 và bật cờ TF. Do vậy **Độ trễ** = $14 * 1\mu s = 14\mu s$

3.4. Lập trình cho bộ định thời/bộ đếm - **Lập trình định thời Chế độ 1 (5)**

Ví dụ 3.3. Tạo trễ có thời gian 1ms cho 8051 có XTAL = 12Hz

- **Xác định nội dung của TL, TH:**

Thời gian trễ = 1ms = 1000 μ S = (Số đếm + 1) * 1 μ s = ((65535 – THTL)+1) * 1 μ s

$$\rightarrow 1000 = 65536 - THTL$$

$$\rightarrow THTL = 64536 = FC18h$$

$$\rightarrow TH = FCh, TL = 18h$$

- **Chương trình:**

DELAY:

MOV TMOD, #01h ; chọn Timer 0, chế độ 1

MOV TH0, #0FCh

MOV TL0, #018H

SETB TR0 ; khởi động Timer 0

HERE: JNB TF0, HERE ; kiểm tra cờ Timer 0

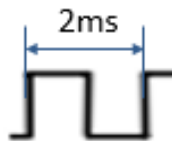
CLR TR0 ; xóa TR0 (dừng Timer 0)

CLR TF0 ; Xóa TF0

RET



3.4. Lập trình cho bộ định thời/bộ đếm - **Lập trình định thời Chế độ 1 (6)**

Ví dụ 3.4. Tạo xung vuông T = 2 ms trên chân P2.0 của 8051 (XTAL=12 MHz), sử dụng Timer 0 để tạo trễ thời gian.

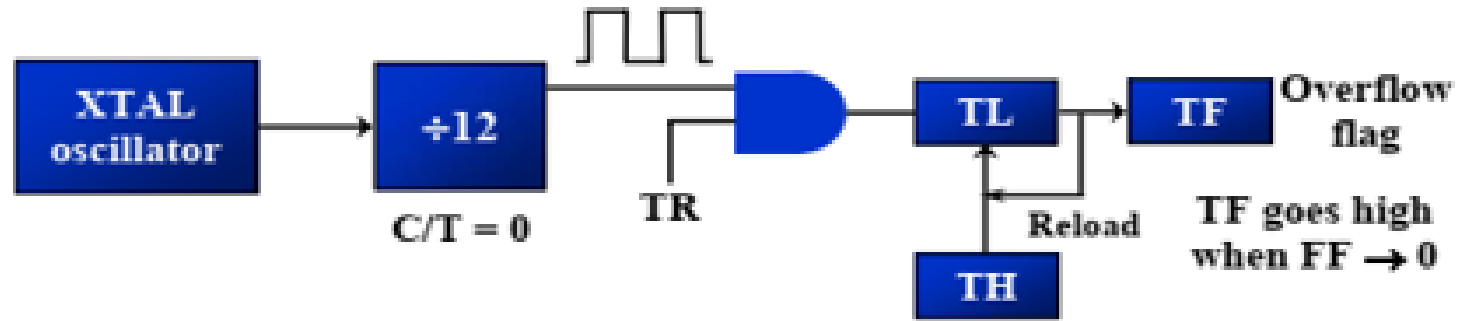


Theo **Ví dụ 3.3**, để tạo trễ 1ms sử dụng 8051 có XTAL = 12 MHz, nội dung TH = FCh, TL = 18h.

Chương trình:

	MOV	TMOD, #01h	; chọn Timer 0, chế độ 1		
HERE:	MOV	TH0, #0FCh			
	MOV	TL0, #18h			
	CPL	P2.0		Phần mềm	
	ACALL	DELAY			
	SJMP	HERE	; nạp lại TH, TL		
DELAY:				Phần cứng	
	SETB	TR0	; khởi động Timer 0		
AGAIN:	JNB	TF0, AGAIN	; kiểm tra cờ Timer 0		
	CLR	TR0	; xóa TR0 (dừng Timer 0)		
	CLR	TF0	; Xóa TF0		
	RET				

3.4. Lập trình cho bộ định thời/bộ đếm - **Lập trình định thời Chế độ 2 (1)**



- 1) Bộ định thời 8 bit, nạp các giá trị từ 00h đến FFh vào thanh ghi TH của bộ định thời.
- 2) Sau khi TH được nạp, 8051 sao nội dung của TH vào TL và bộ định thời được khởi động bằng lệnh “SETB TR0” đối với Timer 0, “SETB TR1” đối với Timer 1.
- 3) Sau khi được khởi động bộ định thời bắt đầu đếm tăng nội dung TL cho tới khi đạt giá trị giới hạn FFh. Khi quay vòng từ FFh về 00h cờ TF0 (Timer 0) hoặc TF1 (Timer 1) được lập bằng 1.
- 4) Khi TF = 1, TL được tự động nạp lại với giá trị ban đầu đang được lưu ở TH. Để lặp lại quá trình chỉ cần xóa cờ TF và bộ định thời tự làm việc mà không cần nạp lại giá trị ban đầu (chế độ tự nạp – khác với chế độ 1 phải nạp lại các thanh ghi TH và TL)

3.4. Lập trình cho bộ định thời/bộ đếm - **Lập trình định thời Chế độ 2 (2)**

❖ **Các bước lập trình**

- 1) Nạp giá trị cho thanh ghi TMOD xác định bộ Timer 0 hay Timer 1 và chế độ nào được chọn.
- 2) Nạp giá trị đếm ban đầu cho các thanh ghi TH
- 3) Khởi động bộ định thời
- 4) Kiểm tra trạng thái bật của bộ định thời TF bằng lệnh “JNB TFx, đích”. Thoát khỏi vòng lặp khi TF được bật lên cao
- 5) Xóa cờ TF cho vòng kế tiếp.
- 6) Quay trở lại bước 4 vì chế độ 2 tự nạp lại

3.4. Lập trình cho bộ định thời/bộ đếm - **Lập trình định thời Chế độ 2 (3)**

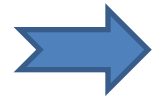
Ví dụ 3.5. Xác định tần số sóng vuông được tạo ra trên chân P2.0 trong đoạn chương trình sau, giả thiết tần số của 8051 XTAL = 12MHz

```
MOV    TMOD, #20h    ; chọn Timer 1 chế độ 2 (8 bit, tự nạp lại)
MOV    TH1, #5        ; TH1 = 5
SETB   TR1           ; khởi động Timer 1
AGAIN: JNB   TF1, AGAIN ; kiểm tra cờ Timer 1
CPL    P2.0
CLR    TF1           ; xóa cờ Timer 1
SJMP   AGAIN
```

Giải:

- Thời gian trễ (nửa chu kỳ) = $(256 - 5) * 1\mu s = 251 \mu s$.
- Cả chu kỳ xung $T = 251 * 2 = 502 \mu s$
- Tần số = $1/T = 1,992kHz$

Phần mềm



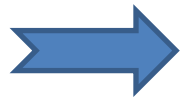
Phần cứng



3.4. Lập trình cho bộ định thời/bộ đếm - **Lập trình định thời Chế độ 2 (4)**

Bài tập 3.1. Giả thiết nội dung của TH = 08Bh, TL = 3Eh của 8051 làm việc ở tần số 12 MHz. Tính thời gian trễ (sử dụng Timer 0 làm việc ở chế độ 1), từ đó viết đoạn chương trình tạo ra một dãy xung vuông tại chân P2.0 có chu kỳ bằng 4 lần thời gian trễ đã tính.

Phần mềm



Phần cứng



Bài tập 3.2. Tính thời gian trễ lớn nhất sử dụng Timer của 8051 làm việc ở tần số 12MHz (sử dụng Timer 1 làm việc ở chế độ 2), từ đó viết đoạn chương trình tạo ra dãy xung vuông tại chân P2.0 có chu kỳ bằng 10 lần thời gian trễ đã tính.

Phần mềm



Phần cứng



3.4. Lập trình cho bộ định thời/bộ đếm - **Lập trình bộ đếm (1)**

- Khi Bộ định thời/bộ đếm thực hiện chức năng định thời sử dụng nguồn xung thạch anh của 8051, còn thực hiện chức năng bộ đếm thì sử dụng nguồn xung lấy từ bên ngoài của 8051 để tăng nội dung các thanh ghi TH, TL.
- Bit C/T của TMOD có nhiệm vụ xác định nguồn xung. Nếu C/T=0 bộ định thời nhận xung đồng hồ từ bộ dao động thạch anh của 8051, nếu C/T=1 bộ đếm nhận xung từ bên ngoài 8051.
- C/T=1 bộ đếm thực hiện đếm tăng khi có xung đồng hồ đưa tới chân P3.4 (T0 – đầu vào của Timer 0) hoặc P3.5(T1 – đầu vào của Timer 1)

3.4. Lập trình cho bộ định thời/bộ đếm - **Lập trình bộ đếm (2)**

Ví dụ 3.6. Giả sử xung đồng hồ được cấp tới chân T1. Viết chương trình cho bộ đếm 1 ở chế độ 2 để đếm các xung và hiển thị trạng thái của số đếm TL1 trên cổng P2

MOV	TMOD, #01100000B	; chọn bộ đếm 1, chế độ 2,	
		; C/T=1 xung ngoài	
MOV	TH1, #0	; xóa TH1	
SETB	P3.5	; lấy đầu vào T1	
AGAIN: SETB	TR1	; khởi động bộ đếm	Phần mềm →
BACK: MOV	A, TL1	; lấy bản sao số đếm TL1	
MOV	P2, A	; đưa TL1 hiển thị ra cổng P2	Phần cứng →
JNB	TF1, BACK	; duy trì đếm nếu TF = 0	
CLR	TR1	; dung bộ đếm	
CLR	TF1	; xóa cờ TF1	
SJMP	AGAIN	; tiếp tục thực hiện	

➤ **Lưu ý:** các cổng được thiết lập là cổng ra mỗi khi 8051 được cấp nguồn, do vậy để P3.5 là đầu vào phải chuyển nó lên mức cao bằng lệnh “SETB P3.5

3.5. Truyền thông nối tiếp - **Mở đầu**

- 8051 có một cổng nối tiếp, chức năng cơ bản là thực hiện chuyển đổi dữ liệu song song sang nối tiếp khi phát và chuyển đổi nối tiếp sang song song khi thu.
- Các mạch phần cứng bên ngoài truy xuất cổng nối tiếp thông qua chân TxD (P3.1) – phát dữ liệu, chân RxD (P3.0) – thu dữ liệu.
- Cổng nối tiếp của 8051 hoạt động song công (full duplex), tức là có khả năng thu và phát đồng thời.
- Để truy xuất cổng nối tiếp 8051 sử dụng hai thanh ghi: SBUF (đệm dữ liệu) và SCON (cấu hình cổng nối tiếp)

3.5. Truyền thông nối tiếp - Tốc độ baud của 8051 (1)

- Tốc độ truyền tin nối tiếp được tính bằng bit/giây (*bps*). Một thuật ngữ khác cũng thường được sử dụng là *baud*. Tuy nhiên, khái niệm *bps* và *baud* không hoàn toàn giống nhau. *Baud* là đơn vị đo dùng cho modem và được định nghĩa là số lần thay đổi tín hiệu trong một giây. Đối với modem, mỗi lần thay đổi tín hiệu có thể truyền được nhiều bit dữ liệu. Còn đối với đường truyền thì tốc độ *baud* và *bps* là một. Do vậy, ở đây sẽ không phân biệt hai thuật ngữ trên.
- Với 8051 thu/phát dữ liệu nối tiếp theo nhiều tốc độ khác nhau. Việc thay đổi tốc độ được lập trình qua bộ định thời Timer 1 và được lập trình về mode 2, chế độ thanh ghi 8 bit tự động nạp lại.

3.5. Truyền thông nối tiếp - Tốc độ baud của 8051 (2)

XTAL (MHz)	XTAL (Hz)	BAUD	$TH1=256 - ((XTAL / (12*32)) / Baud)$	HEX	Thập phân
11.0592	11059200	9600	253	FDh	-3
		4800	250	FAh	-6
		2400	244	F4h	-12
		1200	232	E8h	-24
12	12000000	9600	252.7447917		
		4800	249.4895833		
		2400	242.9791667		
		1200	229.9583333		

3.5. Truyền thông nối tiếp - Tốc độ baud của 8051 (3)

❖ Bảng tốc độ baud

*Bảng 3.3. Giá trị thanh ghi TH1 của Timer1 với các tốc độ baud khác nhau
(với XTAL = 11.0592)*

Tốc độ Baud	TH1 (thập phân)	TH1 (số hex)
9600	-3	ED
4800	-6	FA
2400	-12	F4
1200	-24	E8

➤ Để nhân đôi tốc độ Baud:

MOV A, PCON ; nạp thanh ghi PCON vào ACC

SETB ACC.7 ; đặt D7 = 1

MOV PCON, A ; đặt SMOD (của PCON) = 1 nhân đôi tốc độ baud

3.5. Truyền thông nối tiếp - Các thanh ghi cổng nối tiếp (1)

❖ **Thanh ghi SBUF:** là thanh ghi 8 bit. Dữ liệu được đặt vào SBUF trước khi truyền, và lưu dữ liệu nhận được. SBUF được truy cập như các thanh ghi khác.

Ví dụ 3.7:

MOV SBUF, #“D” ; nạp 44h (mã ASCII của “D”) vào SBUF

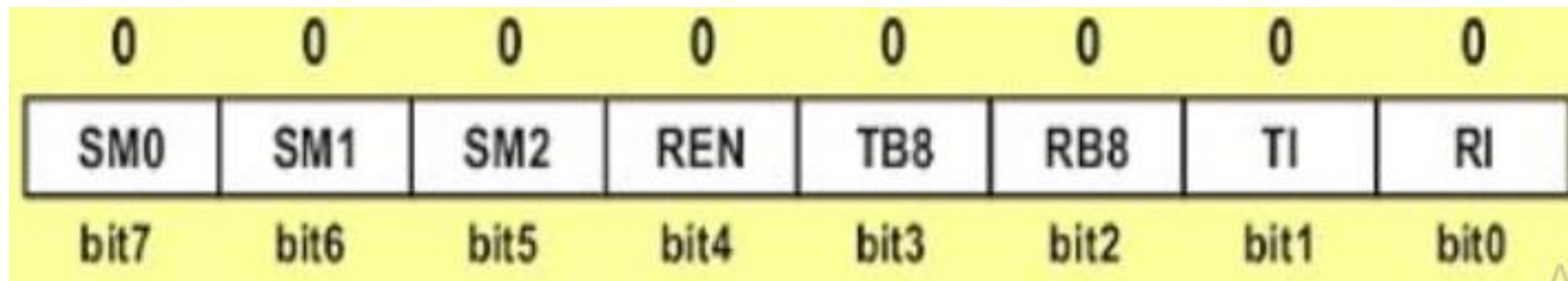
MOV SBUF, A ; chuyển nội của của A vào SBUF

MOV A, SBUF ; chuyển nội dung SBUF vào A

- Khi byte dữ liệu được ghi vào SBUF, byte dữ liệu được định khung với bit Start và Stop và được truyền nối tiếp qua TxD.
- Khi các bit được nhận nối tiếp từ RxD thì 8051 mở khung, tức là loại các bit Start, Stop để lấy ra byte dữ liệu và đặt vào SBUF

3.5. Truyền thông nối tiếp - Các thanh ghi cổng nối tiếp (2)

❖ **Thanh ghi điều khiển nối tiếp SCON:** là thanh ghi 8 bit được lập trình bit khởi động Start, bit dừng Stop và các bit dữ liệu khi định khung dữ liệu, Hình 3.3



Hình 3.3. Thanh ghi điều khiển nối tiếp SCON

SM0, SM1: được sử dụng để xác định chế độ khung dữ liệu bằng cách xác định số bit của một ký tự và các bit Start, Stop. Các tổ hợp của chúng Bảng 3.4

3.5. Truyền thông nối tiếp - Các thanh ghi cổng nối tiếp (3)

❖ Thanh ghi điều khiển nối tiếp SCON:

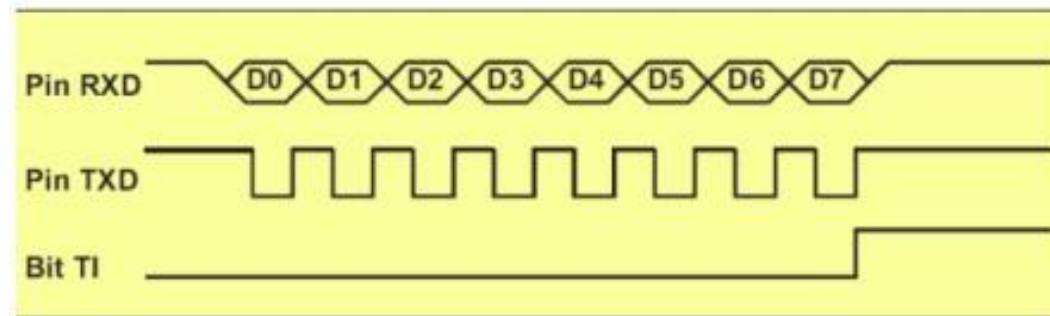
Bảng 3.5. Các chế độ cổng nối tiếp

SM0	SM1	Chế độ	Mô tả	Tốc độ baud
0	0	0	Thanh ghi dịch	Cố định (1/12 tần số clock)
0	1	1	UART 8 bit	Thay đổi (cấu hình qua Timer1)
1	0	2	UART 9 bit	Cố định (1/12 hoặc 1/64 tần số clock)
1	1	3	UART 9 bit	Thay đổi (cấu hình qua Timer1)

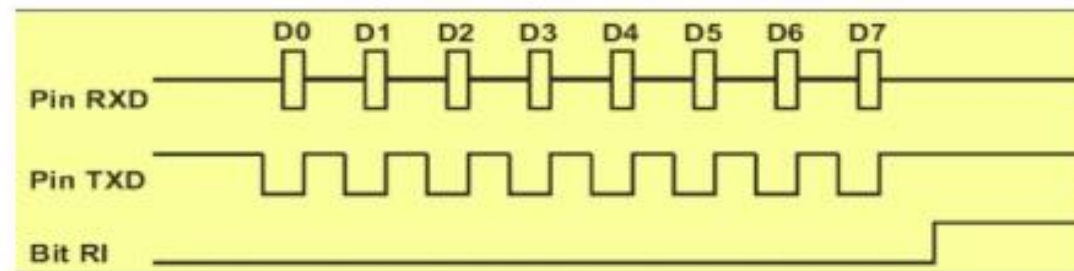
3.5. Truyền thông nối tiếp - Các thanh ghi cổng nối tiếp (4)

❖ Thanh ghi điều khiển nối tiếp SCON:

- **Chế độ 0:** Đây là chế độ thanh ghi dịch 8 bit, không có bit start/stop, ở chế độ này RxD là chân truyền nhận, còn TxD phát xung đồng bộ
 - Quá trình truyền bắt đầu khi ghi giá trị vào SBUF, kết thúc khi $TI = 1$



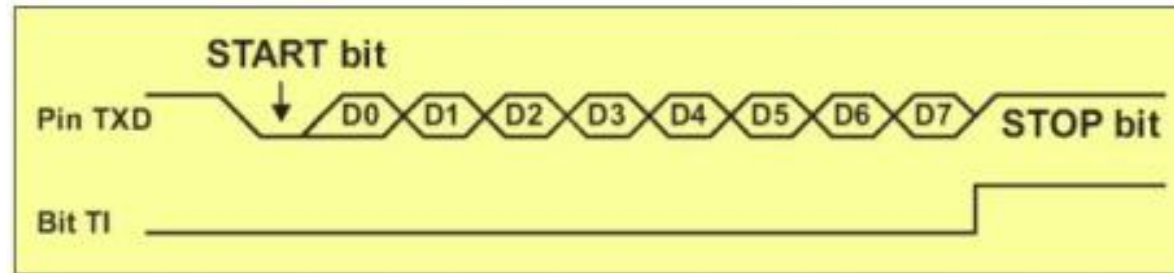
- Quá trình nhận tự động bởi hệ thống và kết thúc khi $RI = 1$



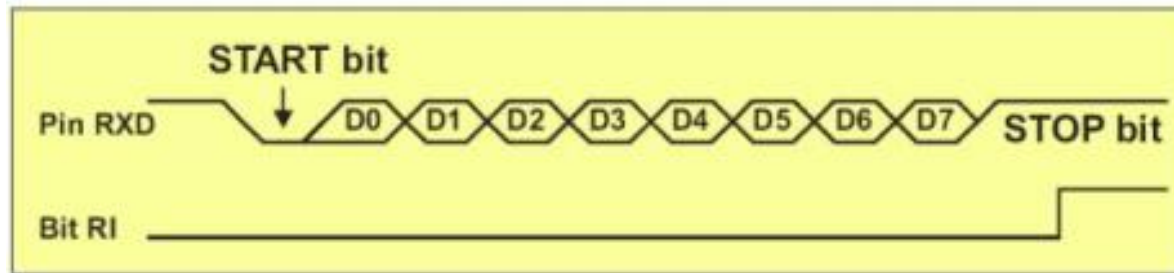
3.5. Truyền thông nối tiếp - Các thanh ghi cổng nối tiếp (5)

❖ Thanh ghi điều khiển nối tiếp SCON:

- **Chế độ 1:** Truyền thông bất đồng bộ với frame truyền 10 bit (1 bit Start, 8 bit dữ liệu, 1 bit Stop). TxD truyền, RxD nhận dữ liệu, tốc độ truyền được cài đặt qua Timer 1
- Quá trình truyền:



- Quá trình nhận:

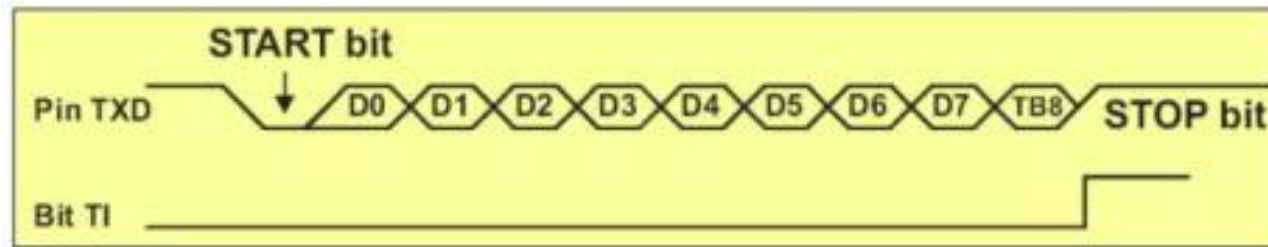


3.5. Truyền thông nối tiếp - Các thanh ghi cổng nối tiếp (6)

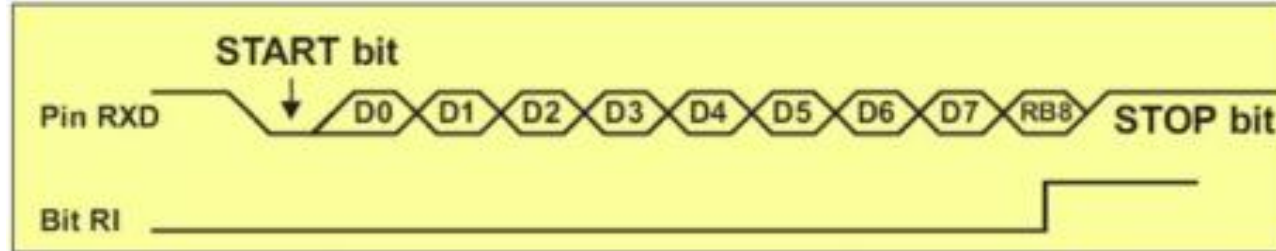
❖ Thanh ghi điều khiển nối tiếp SCON:

- **Chế độ 2:** Truyền thông bất đồng bộ với frame truyền 11 bit: 1 bit Start, 8 bit dữ liệu, 1 bit lập trình được (bên truyền là TB8, nhận là RB8) và 1 bit Stop. TxD thực hiện truyền, RxD nhận dữ liệu, tốc độ truyền cài đặt qua Timer1. Bit thứ 9 thường được dung là bit parity.

- Quá trình truyền:



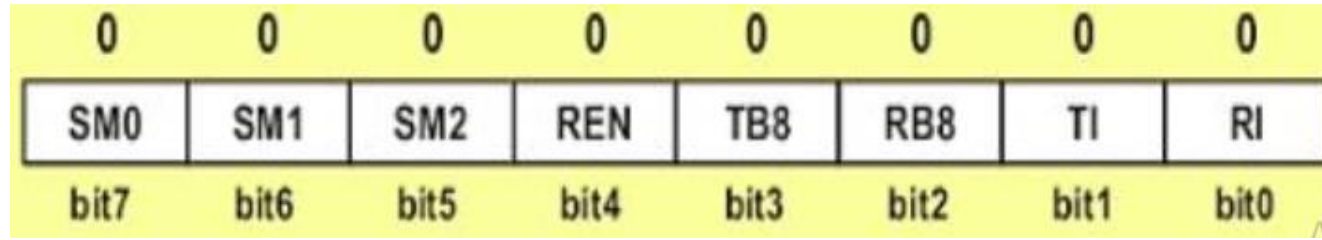
- Quá trình nhận:



- **Chế độ 3:** tương tự như chế độ 2, trừ tốc độ baud

3.5. Truyền thông nối tiếp - Các thanh ghi cổng nối tiếp (7)

- Thanh ghi điều khiển nối tiếp SCON:



Hình 3.3. Thanh ghi điều khiển nối tiếp SCON

SM2: SM2 =0 hệ thống đơn xử lý, SM2=1 hệ thống đa xử lý (truyền thông giữa các bộ vi xử lý)

REN: bit cho phép thu, để thu dữ liệu bit REN phải được set bằng 1, nếu REN bằng 0 chỉ cho phép phát (bộ thu bị khóa)

TB8 và RB8: được dùng cho chế độ nối tiếp 2 và 3.

TI: Cờ ngắt truyền, khi một byte trong SBUF được truyền thành công thì TI = 1. Trước khi truyền byte khác bit này cần phải xóa bằng phần mềm

RI: Cờ ngắt nhận, khi nhận thành công 1 byte vào SBUF thì RI = 1. Sau khi đọc SBUF, RI cần phải xóa bằng phần mềm

3.5. Truyền thông nối tiếp – Lập trình truyền DL nối tiếp chế độ 1 (1)

1. Nạp giá trị 20h vào thanh ghi TMOD báo sử dụng Timer1 ở chế độ 2 (8 bit tự nạp) để thiết lập chế độ baud.
2. Nạp giá trị vào TH1 để thiết lập chế độ baud
3. Nạp giá trị 50h vào SCON báo chế độ nối tiếp 1 để định khung 8 bit dữ liệu, 1 bit start, 1 bit stop.
4. Bật TR1=1 khởi động Timer 1
5. Xóa bit TI bằng lệnh CLR TI
6. Ghi byte ký tự cần truyền vào SBUF
7. Kiểm tra bit cờ TI bằng lệnh “JNB TI, xx” để báo hoàn tất việc chuyển ký tự
8. Trở về bước 5 để truyền ký tự tiếp theo

3.5. Truyền thông nối tiếp – Lập trình truyền DL nối tiếp chế độ 1 (2)

Ví dụ 3.8: truyền nối tiếp liên tục một ký tự “A” với tốc độ 4800 baud.

```
MOV  TMOD, #20h  ; chọn Timer 1, chế độ 2 (tự nạp lại)
MOV  TH1, #-6     ; chọn tốc độ 4800 baud
MOV  SCON, #50h   ; khung uart: 8b+1start+1stop, cho phép thu
SETB TR1          ; khởi động Timer1
AGAIN: MOV  SBUF, #'A' ; truyền ký tự “A”
HERE: JNB   TI, HERE ; chờ đến bit cuối cùng
      CLR  TI       ; xóa bit TI cho ký tự tiếp theo
      SJMP AGAIN    ; tiếp tục gửi lại chữ “A”
```

3.5. Truyền thông nối tiếp – Lập trình nhận DL nối tiếp chế độ 1 (1)

1. Nạp giá trị 20h vào thanh ghi TMOD báo sử dụng Timer1 ở chế độ 2 (8 bit, tự động nạp lại) để thiết lập chế độ baud.
2. Nạp giá trị vào TH1 để thiết lập chế độ baud
3. Nạp giá trị 50h vào SCON báo chế độ nối tiếp 1 để định khung 8 bit dữ liệu, 1 bit start, 1 bit stop.
4. Bật TR1=1 khởi động Timer 1
5. Xóa bit RI bằng lệnh CLR RI
6. Kiểm tra bit cờ RI bằng lệnh JNB RI,xx để toàn bộ ký tự đã nhận được chưa
7. Khi RI được thiết lập thì trong SBUF đã có 1 byte. Các nội dung của nó được lưu.
8. Quay về bước 5 để nhận một ký tự tiếp theo

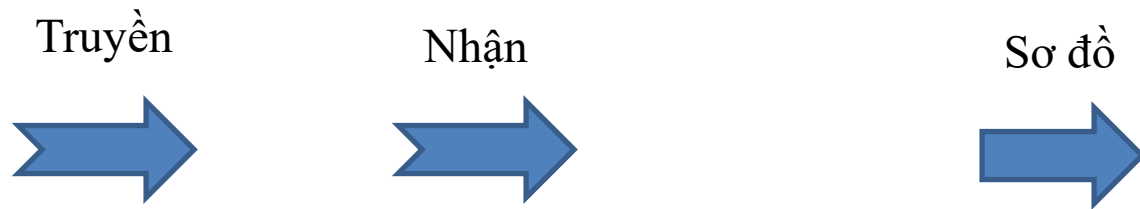
3.5. Truyền thông nối tiếp – Lập trình nhận DL nối tiếp chế độ 1 (2)

Ví dụ 3.9: Nhận các byte dữ liệu nối tiếp và gửi ra cổng P1, tốc độ 4800 baud, 8 bit dữ liệu và 1 bit stop

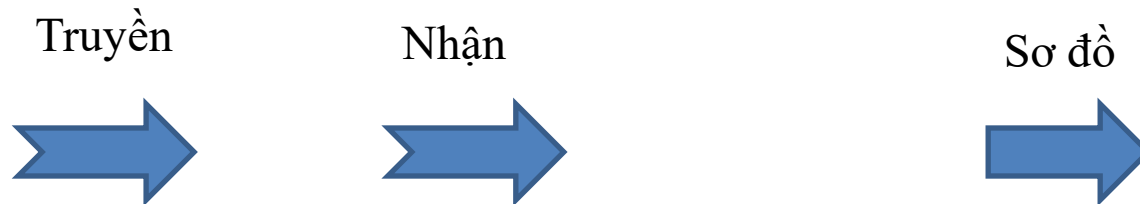
```
MOV  TMOD, #20h  ; chọn Timer 1, chế độ 2 (tự nạp lại)
MOV  TH1, #-6     ; chọn tốc độ 4800 baud
MOV  SCON, #50h   ; khung uart: 8b+1start+1stop, cho phép thu
SETB TR1          ; khởi động Timer1
HERE: JNB  RI, HERE ; chờ đến bit cuối cùng
MOV  A, SBUF      ; lưu ký tự vào thanh ghi A
MOV  P1, A        ; gửi ra cổng P1
CLR  RI           ; sẵn sàng nhận byte kế tiếp
SJMP HERE         ; tiếp tục nhận dữ liệu
```

3.5. Truyền thông nối tiếp – Bài tập

Bài tập 3.3: Thực hiện truyền nối tiếp giữa 02 bộ vi điều khiển chế độ 1, tốc độ UART 9600 baud để truyền nối tiếp (một chiều) các số đếm (tăng) từ 0 tới 9 và hiển thị lên LED 7 đoạn.



Bài tập 3.4: Thực hiện truyền nối tiếp giữa 02 bộ vi điều khiển chế độ 1, tốc độ UART 9600 baud để truyền nối tiếp (hai chiều) các số đếm (một bộ đếm tăng, một bộ đếm giảm) từ 0 đến 9 và hiển thị trên LED 7 đoạn





Chương 4.

LẬP TRÌNH NGẮT TRONG 8051

MỤC TIÊU

- **Hiểu được việc tổ chức ngắt trong 8051**
- **Có khả năng lập trình định thời, truyền thông nối tiếp UART sử dụng ngắt**

NỘI DUNG

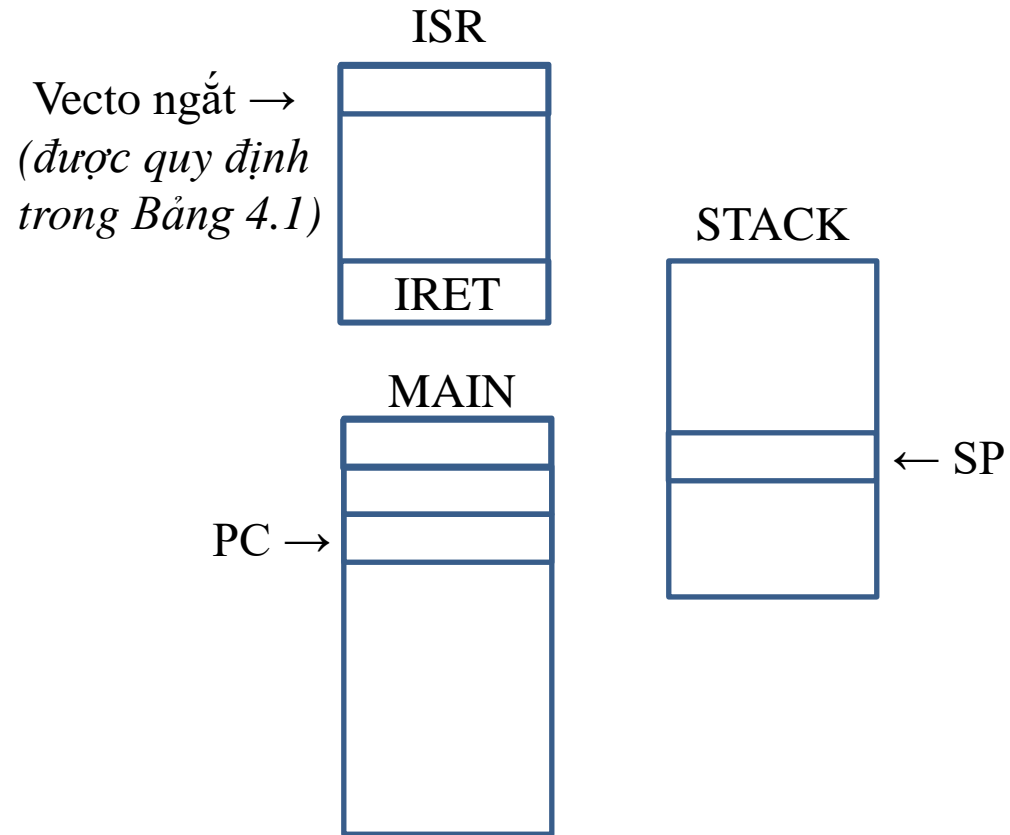
- 4.1. Giới thiệu ngắt**
- 4.2. Tổ chức ngắt của 8051**
- 4.3. Xử lý ngắt**
- 4.4. Lập trình ngắt bộ định thời**
- 4.5. Lập trình ngắt truyền thông nối tiếp**

4.1. Giới thiệu ngắt (1)

- Ngắt (interrupt) là một sự kiện bên trong hoặc bên ngoài làm ngưng một chương trình khác đang thực thi. Các ngắt đóng vai trò quan trọng trong việc thiết kế và thực hiện các ứng dụng của bộ vi điều khiển
- Chương trình xử lý một ngắt được gọi là trình phục vụ ngắt ISR (interrupt service routine) hay trình quản lý ngắt (interrupt handler).
- **Trình tự thực xử lý ngắt:**
 1. Kết thúc lệnh hiện tại và lưu địa chỉ của lệnh kế tiếp (PC) vào Stack
 2. Lưu trạng thái hiện hành của tất cả các ngắt
 3. Nhảy đến một vị trí cố định trong bộ nhớ được gọi là bảng vector ngắt, nơi lưu giữ địa chỉ của một **trình phục vụ ngắt**.
 4. Nhận địa chỉ **ISR** từ bảng vector ngắt và nhảy tới địa chỉ đó và bắt đầu thực hiện trình phục vụ ngắt cho đến lệnh cuối cùng của **ISR** là RETI.
 5. Khi gặp lệnh RETI (kết thúc trình phục vụ ngắt), bộ vi điều khiển quay trở về nơi nó đã bị ngắt bằng cách nạp 02 byte của đỉnh stack vào PC, khi đó bộ VĐK sẽ thực hiện tiếp các lệnh từ địa chỉ đó

4.1. Giới thiệu ngắt (2)

- **Trình tự thực xử lý ngắt:**



- **Khi có ngắt**
 - $PC + 1 \rightarrow [SP]$
 - $SP = SP + 2$
 - $PC \leftarrow \text{Vecto ngắt}$
- **Khi gặp lệnh IRET**
 - $PC \leftarrow [SP]$
 - $SP = SP - 2$

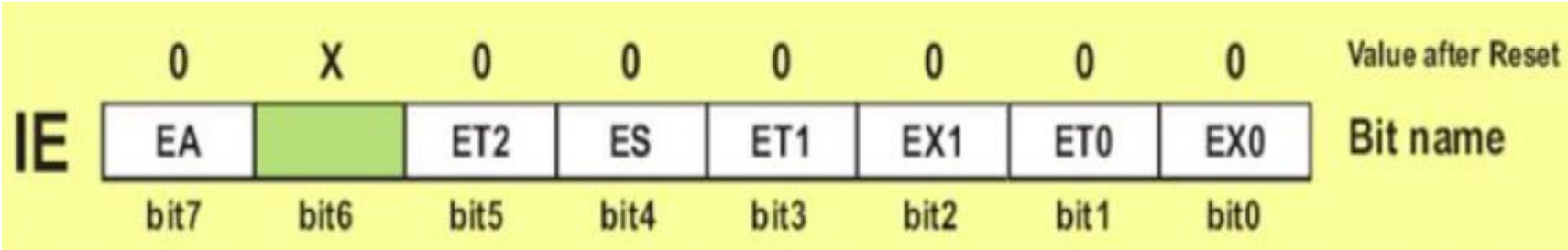
4.2. Tổ chức ngắt của 8051 - Bảng vec tơ ngắt của 8051

Bảng 4.1. Bảng vector ngắt của 8051

Ngắt	Cờ ngắt	Địa chỉ trình phục vụ ngắt	Số thứ tự ngắt
Reset	-	0000h	0
Ngắt cứng ngoài 0	IE0	0003h	1
Ngắt Timer0	TF0	000Bh	2
Ngắt cứng ngoài 1	IE1	0013h	3
Ngắt Timer1	TF1	001Bh	4
Ngắt truyền thông nối tiếp	RI/TI	0023h	5

4.2. Tổ chức ngắt của 8051 - Cho phép ngắt và cấm ngắt (1)

Khi Reset tất cả các ngắt đều bị cấm (bị che). Vì vậy, các ngắt phải được cho phép bằng phần mềm để bộ VĐK có thể đáp ứng được. Việc cho phép hoặc cấm ngắt được thực hiện bởi thanh ghi cho phép ngắt IE (Interrupt Enable) có địa chỉ byte là 0A8h, Hình 4.1



Hình 4.1. Thanh ghi IE

Các bit được mô tả trong Bảng 4.2

4.2. Tổ chức ngắt của 8051 - Cho phép ngắt và cấm ngắt (2)

Bảng 4.2. Mô tả các bit của thanh ghi IE

Bit	Ký hiệu	Mô tả
IE.7	EA	= 0: cấm tất cả các ngắt = 1: từng nguồn ngắt sẽ được mở hoặc cấm bằng cách bật/xóa bit cho phép tương ứng.
IE.6	--	Dự phòng
IE.5	ET2	Cho phép/cấm ngắt Timer 2
IE.4	ES	Cho phép/cấm ngắt truyền thông nối tiếp
IE.3	ET1	Cho phép/cấm ngắt Timer 1
IE.2	EX1	Cho phép/cấm ngắt ngoài 1
IE.1	ET0	Cho phép/cấm ngắt Timer 0
IE.0	EX0	Cho phép/cấm ngắt ngoài 0

4.2. Tổ chức ngắt của 8051 - Cho phép ngắt và cấm ngắt (3)

Ví dụ 4.1. Viết các lệnh thực hiện

- a) Cho phép ngắt nối tiếp, ngắt Timer 0 và ngắt cứng ngoài 1
- b) Cấm (che) ngắt Timer 0
- c) Cấm tất cả mọi ngắt chỉ bằng một lệnh duy nhất

Giải

a) MOV IE, #10010110B

; lệnh trên tương đương đoạn chương trình

SETB IE.7 ; cho phép tất cả các ngắt

SETB IE.4 ; cho phép ngắt nối tiếp

SETB IE.1 ; cho phép ngắt Timer 1

SETB IE.2 ; cho phép ngắt cứng ngoài 1

b) CLR IE.1 ; xóa (che) ngắt Timer 0

c) CLR IE.7 ; cấm tất cả các ngắt

4.2. Tổ chức ngắt của 8051 - Mức ưu tiên ngắt của 8051 (1)

❖ Mức ưu tiên ngắt khi Reset, Bảng 4.3

Bảng 4.3. Bảng mức ưu tiên ngắt của 8051

Mức ưu tiên từ cao xuống thấp	
Ngắt ngoài 0	INT0
Ngắt bộ định thời 0	TF0
Ngắt ngoài 1	INT1
Ngắt bộ định thời 1	TF1
Ngắt truyền thông nối tiếp	RI, TI

❖ Thiết lập mức ưu tiên ngắt sử dụng thanh ghi IP

- Có thể thay đổi trình tự ưu tiên cho ở Bảng 4.3 bằng cách gán mức ưu tiên cao hơn cho bất kỳ ngắt nào. Điều này được thực hiện bằng cách lập trình cho thanh ghi mức ưu tiên ngắt IP (Interrupt Priority).
- Nếu nhiều bit trong thanh ghi IP cùng được đặt lên cao, khi đó chúng được phục vụ theo trình tự trong Bảng 4.3

4.2. Tổ chức ngắt của 8051 - Mức ưu tiên ngắt của 8051 (2)

❖ Thiết lập mức ưu tiên ngắt sử dụng thanh ghi IP



Hình 4.3. Thanh ghi IP

--	IP.7	Dự trữ
--	IP.6	Dự trữ
PT2	IP.5	Bit ưu tiên ngắt Timer2 (dùng cho 8052)
PS	IP.4	Bit ưu tiên ngắt cổng nối tiếp
PT1	IP.3	Bit ưu tiên ngắt Timer1
PX1	IP.2	Bit ưu tiên ngắt ngoài 1
PT0	IP.1	Bit ưu tiên ngắt Timer0
PX0	IP.0	Bit ưu tiên ngắt ngoài 0

4.2. Tổ chức ngắt của 8051 - Mức ưu tiên ngắt của 8051 (3)

❖ Thiết lập mức ưu tiên ngắt sử dụng thanh ghi IP, ví dụ:

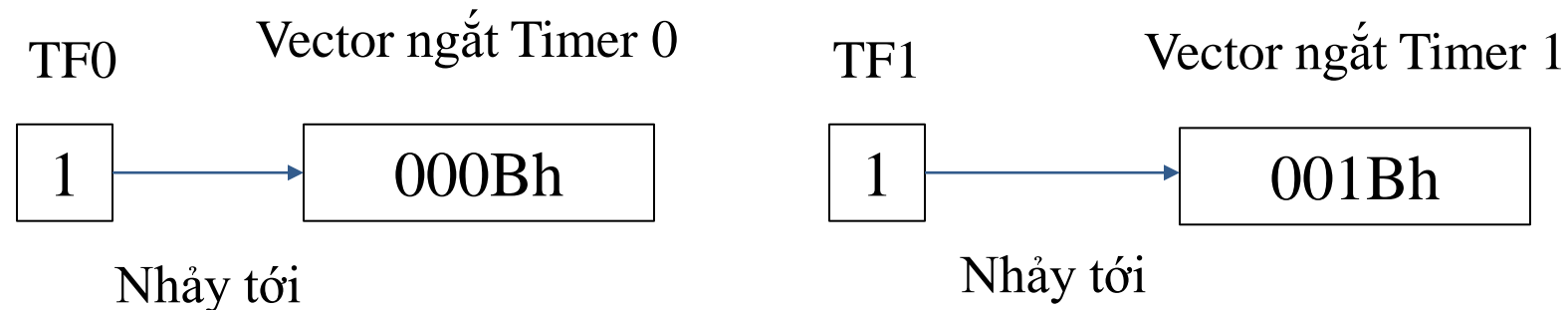
- a) Lập trình thanh ghi IP để gán mức ưu tiên cao nhất cho ngắt INT1
- b) Phân tích điều gì xảy ra khi INT0, INT1 và TF0 được kích hoạt cùng một lúc. Giả thiết tất cả các ngắt đều có tách động ngắt theo sườn

Giải:

- a) `MOV IP, #00000100B` ; đặt IP.2=1 để INT1 có mức ưu tiên cao hơn
; tương đương lệnh “`SETB IP.2`”
- b) INT1 được gán mức ưu tiên cao hơn, do vậy khi INT0, INT1 và TF0 cùng kích hoạt thì thứ tự được đáp ứng: INT1, INT0, TF0

4.3. Lập trình ngắt bộ định thời - Các bước lập trình ngắt bộ định thời

- Lập bit IE.7 (EA) cho phép các ngắt
- Lập bit cho phép ngắt bộ định thời IE.1 (ET0) đối với Timer 0, IE.3 (ET1) đối với Timer 1
- Khi bộ định thời quay trở về 0, cờ TF được lập bằng 1, bộ VĐK sẽ ngừng chương trình chính đang thực hiện và nhảy tới địa chỉ 000Bh (Vector ngắt Timer 0), 001Bh (Vector ngắt Timer 1) để thực hiện trình điều khiển ngắt (ISR)



4.3. Lập trình ngắt bộ định thời – Một số ví dụ (1)

Ví dụ 4.2. Tạo xung vuông $T = 2\text{ ms}$ trên chân P2.0 của 8051 (XTAL=12 MHz), sử dụng ngắt Timer 0. Nội dung TH = FCh, TL = 18h (xem **Ví dụ 3.3**)

```

                ORG      0000h
                LJMP     MAIN
                ORG      000Bh          ; vector ngat cua Timer 0
                CPL      P2.0          ; lay bù
                MOV      TH0, #0FCh
                MOV      TL0, #18h
                SETB     TR0
                RETI
MAIN:           ORG      0030h          ; dia chi chuong trình chính
                MOV      TMOD, #01h    ; chon Timer 0, che do 1
                MOV      TH0, #0FCh
                MOV      TL0, #18h
                MOV      IE, #10000010B ; cho phép ngat Timer 0
                SETB     TR0            ; khoi dong Timer 0
                SJMP     $
                END
```

Phần mềm



Phần cứng

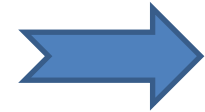


4.3. Lập trình ngắt bộ định thời – Một số ví dụ (2)

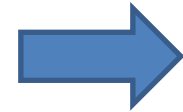
Ví dụ 4.3. Viết chương trình nhận liên tục dữ liệu 8 bit từ cổng P0 và gửi ra cổng P1. Trong thời gian nhận và gửi dữ liệu tạo một sóng vuông có chu kỳ $502\mu\text{s}$. Sử dụng ngắt Timer 0, chế độ 2; tần số của 8051 là $\text{XTAL} = 12\text{MHz}$

```
ORG    0000h
LJMP   MAIN
; --- chương trình ISR cho Timer0 để tạo sóng vuông
ORG    000Bh          ; vector ngắt của Timer 0
CPL    P2.0           ; lấy bù
SETB   TR0
RETI
; --- chương trình chính
ORG    0030h          ; địa chỉ chương trình chính
MAIN:  MOV    TMOD, #02h      ; chọn Timer 0, chế độ 2 tự nạp lại
        MOV    P0, #0FFh     ; P0 làm cổng vào
        MOV    TH0, #5
        MOV    IE, #10000010B ; cho phép ngắt Timer 0
        SETB   TR0           ; khởi động Timer 0
BACK:  MOV    A, P0          ; nhận dữ liệu từ P0
        MOV    P1, A
        SJMP   BACK
END
```

Phần mềm



Phần cứng

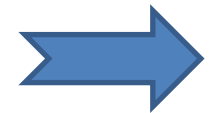


4.3. Lập trình ngắt bộ định thời – Một số ví dụ (3)

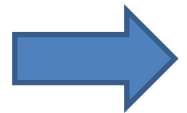
Ví dụ 4.4. Thực hiện lại **ví dụ 4.3**, nhưng sử dụng thêm không gian chương trình để chứa trình điều khiển ngắt ISR

```
ORG    0000h
LJMP   MAIN
; --- chương trình ISR cho Timer0 để tạo sóng vuông
ORG    000Bh                ; vector ngắt của Timer 0
LJMP   ISR_T1
; --- chương trình chính
ORG    0030h                ; địa chỉ chương trình chính
MAIN:  MOV    TMOD, #02h     ; chọn Timer 0, chế độ 2 tải lại
        MOV    P0, #0FFh    ; P0 làm cổng vào
        MOV    TH0, #5
        MOV    IE, #10000010B ; cho phép ngắt Timer 0
        SETB   TR0          ; khởi động Timer 0
BACK:  MOV    A, P0          ; nhận dữ liệu từ P0
        MOV    P1, A
        SJMP   BACK
ISR_T1: CPL    P2.0          ; lấy bù
        SETB   TR0
        RETI
END
```

Phần mềm



Phần cứng



4.3. Lập trình ngắt bộ định thời – Một số lưu ý

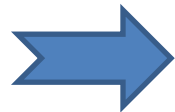
1. Trình ứng dụng không được sử dụng không gian bộ nhớ dành cho bảng vector ngắt (trong trường hợp hệ thống sử dụng ngắt). Do vậy, mã chương trình được đặt từ địa chỉ 30h. Lệnh LJMP lái bộ điều khiển tránh khỏi bảng vector ngắt.
2. Nếu trình phục vụ ngắt (ISR) có kích thước nhỏ đặt vừa trong không gian nhớ dành cho ngắt trong bảng vector ngắt thì bắt đầu từ địa chỉ tương ứng (ví dụ 4.3). Nếu kích thước ISR lớn hơn không gian nhớ dành cho ngắt thì sử dụng thêm vùng chương trình (ví dụ 4.4).
3. Mỗi khi bộ Timer quay về 0 thì cờ TFX được bật lên, bộ VĐK sẽ nhảy tới vector ngắt tương ứng để thực hiện trình phục ngắt (ISR)
4. Trong trình phục vụ ngắt không cần thực hiện lệnh “CLR TFX” trước lệnh RETI vì 8051 sẽ xóa cờ TF khi nhảy tới bảng vector ngắt.

4.3. Lập trình ngắt bộ định thời – Bài tập

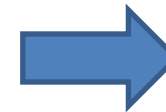
Bài tập 4.1. Viết chương trình nhận liên tục dữ liệu 8 bit từ cổng P0 và gửi ra cổng P1. Trong thời gian nhận và gửi dữ liệu tạo một sóng vuông với mức cao kéo dài $1000\mu\text{s}$, mức thấp dài $500\mu\text{s}$. Sử dụng ngắt Timer 1, chế độ; tần số của 8051 là $\text{XTAL} = 12\text{MHz}$

- Trễ $1000\mu\text{s} \rightarrow$ nội dung TH1 FCh; TL1=18h (xem ví dụ 3.3)
- Trễ $500\mu\text{s} \rightarrow$ nội dung TH1=?, TL1=?

Phần mềm

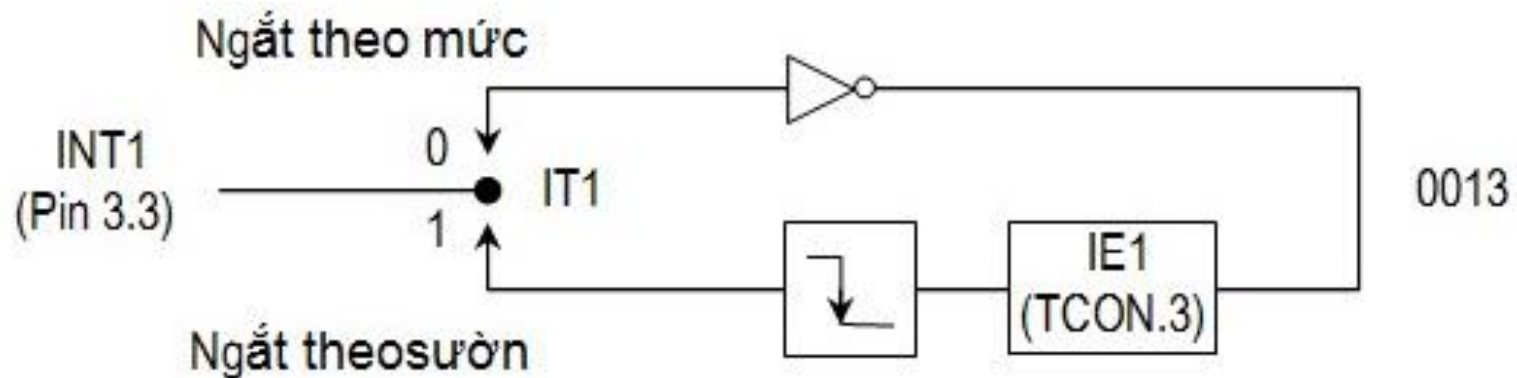
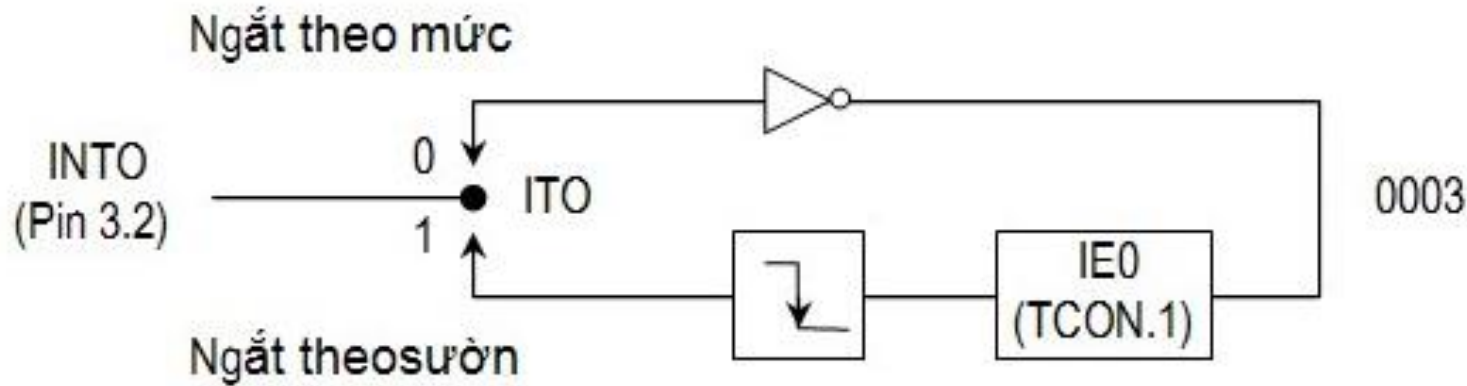


Phần cứng



4.4. Lập trình ngắt cứng ngoài

8051 có 02 ngắt cứng ngoài INT0 (P3.2), INT1 (P3.3). Vector ngắt tương ứng là 0003h và 0013h. Có 02 cách kích hoạt ngắt cứng ngoài là theo mức hoặc theo sườn.



4.4. Lập trình ngắt cứng ngoài - Kích hoạt ngắt theo mức (1)

Là chế độ mặc định khi khởi động hoặc Reset. Ở chế độ này, các chân INT0 và INT1 bình thường ở mức cao, khi có tín hiệu mức thấp cấp tới tín hiệu được kích hoạt. Khi đó bộ VĐK sẽ dừng chương trình chính và chuyển sang thực hiện ISR có vector ngắt tương ứng.

Trước khi thực hiện lệnh RETI mức thấp tại chân INTx phải chuyển sang mức cao, nếu không sẽ tạo ra một ngắt khác. Tức là, nếu vẫn duy ở mức thấp khi ISR kết thúc 8051 sẽ hiểu là có một ngắt mới và nhảy đến bảng vector ngắt để thực hiện ISR

4.4. Lập trình ngắt cứng ngoài - Kích hoạt ngắt theo mức (2)

Ví dụ 4.5. Giả sử chân INT được nối đến công tắc bình thường ở mức cao. Mỗi khi chân này xuống thấp thì bật đèn LED. Đèn LED được nối đến chân P2.0 và bình thường ở chế độ tắt. Nếu đèn được bật thì cần phải sáng vài phần giây. Khi công tắc được ấn xuống, đèn LED sáng liên tục

```
ORG      0000h
LJMP     MAIN
; --- chương trình ISR cho ngắt ngoại INT1 để bật LED
ORG      0013h                ; vector ngắt của Timer 0
CLR      P2.0
MOV      R3, #255
BACK:    DJNZ R3, BACK
SETB     P2.0
RETI
; --- chương trình chính
ORG      0030h                ; địa chỉ chương trình chính
MAIN:    MOV IE, #10000100B    ; cho phép ngắt INT1
SJMP     $
END
```

Phần mềm



Phần cứng



4.4. Lập trình ngắt cứng ngoài - Kích hoạt ngắt theo sườn (1)

Để đồ cách kích hoạt ngắt theo sườn cần phải lập trình cho các bit của thanh ghi TCON (thanh ghi điều khiển bộ định thời/bộ đếm), Hình 4.2

7	6	5	4	3	2	1	0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

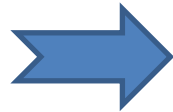
Hình 4.2. TCON – Thành ghi điều khiển bộ định thời/bộ đếm

- Bốn bit từ TCON.4 đến TCON.7 (đã tìm hiểu trong mục 3.2)
- **TCON.3 (IE1):** cờ ngắt INT1 (kích hoạt ngắt theo sườn), được CPU thiết lập khi phát hiện có sườn xuống ngắt ngoài và được CPU xóa khi ngắt được xử lý (*lưu ý: không sử dụng đối với kích hoạt ngắt theo mức*)
- **TCON.2 (IT1):** điều khiển ngắt 1, được thiết lập/xóa bằng phần mềm để xác định kiểu kích hoạt ngắt theo sườn xuống, hay mức thấp
- **TCON.1 (IE0):** tương tự như IE1, nhưng cho ngắt INT0
- **TCON.0 (IT0):** tương tự như IT0, nhưng cho ngắt INT0

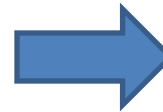
4.4. Lập trình ngắt cứng ngoài - Kích hoạt ngắt theo sườn (2)

Ví dụ 4.6. Giả thiết chân P3.3 (INT1) được nối với một máy tạo xung. Viết chương trình trong đó sườn xuống của xung sẽ chuyển chân P2.0 lên cao, chân này được nối tới đèn LED. Như vậy đèn LED được bật và tắt cùng tần số với xung cấp tới chân INT1.

Phần mềm



Phần cứng



- **Lưu ý:** Khi các ISR kết thúc, nghĩa là khi thực hiện lệnh `RETI`, các bit `TCON.1` và `TCON.3` được xóa để báo rằng phục vụ ngắt đã được hoàn tất và 8051 sẵn sàng đáp ứng ngắt khác trên chân đó. Để có thể nhận được ngắt khác từ tín hiệu trên đó phải trở lại mức cao và sau đó xuống thấp để tạo nên một ngắt tác động theo sườn xuống.

4.5. Lập trình ngắt truyền thông nối tiếp - Cờ RI, TI và ngắt

- Cờ ngắt phát TI (Transfer interrupt) được bật lên khi bit cuối cùng của khung dữ liệu-bit Stop được phát đi báo rằng thanh ghi SBUF sẵn sàng phát byte kế tiếp.
- Cờ ngắt thu RI (Receive Interrupt) được bật lên 1 khi toàn bộ khung dữ liệu kể cả bit Stop đã được nhận. Nói cách khác, thanh ghi SBUF đang lưu một byte, thì cờ RI bật lên báo rằng byte dữ liệu thu được cất đi trước khi bị mất (bị ghi đè) bởi dữ liệu tiếp theo sẽ được nhận.
- 8051 chỉ có một ngắt dành riêng cho truyền thông nối tiếp. Ngắt này được dùng cho cả phát và thu dữ liệu. Nếu bit ngắt trong thanh ghi IE (IE.4) được phép khi RI và TI bật lên, thì 8051 nhận được ngắt và nhảy đến địa chỉ trình phục vụ ngắt dành cho truyền thông nối tiếp có vector ngắt 0023h trong bảng vector ngắt. Lúc đó, cần kiểm tra cờ TI và RI để xem cờ nào gây ngắt để có đáp ứng phù hợp.

4.5. Lập trình ngắt truyền thông nối tiếp – Một số ví dụ (1)

Ví dụ 4.7. Viết chương trình, trong đó 8051 đọc dữ liệu từ cổng P1 và gửi liên tục tới cổng P2 đồng thời đưa một bản sao dữ liệu tới cổng COM nối tiếp để thực hiện truyền nối tiếp. Giả thiết tần số XTAL là 11.0592MHz và tốc độ truyền 9600 baud.

Giải

```
ORG    0000h
LJMP   MAIN
;--- vector ngắt truyền thông nối tiếp
ORG    0023h
LJMP   SERIAL          ; nhảy đến ISR nối tiếp
;--- chương trình chính
ORG    0030h
MAIN:  MOV    P1, #0FFh      ; đặt P1 làm cổng vào
        MOV    TMOD, #20h    ; Timer 1, chế độ 2 tải lại
        MOV    TH1, #0FDh    ; chọn tốc độ baud = 9600
        MOV    SCON, #50h    ; khung 8 bit, 1 stop, cho phép (REN)
        MOV    IE, #10010000h ; cho phép ngắt nối tiếp
        SETB   TR1           ; khởi động Timer 1
```

4.5. Lập trình ngắt truyền thông nối tiếp – Một số ví dụ (2)

Ví dụ 4.7. (tiếp)

```
BACK:  MOV    A, P1                ; nhan du lieu tu cong P1
        MOV    SBUF, A            ; gui sao toi SBUF
        MOV    P2, A              ; gui ra cong P2
        SJMP   BACK               ; lap lai vong lap
; ----trình phục vụ ngắt truyền thông nối tiếp
        ORG    100h
SERIAL: JB     TI, TRANS           ; nhay neu co TI cao
        MOV    A, SBUF            ; neu khong thi nhan du lieu
        CLR    RI                 ; xoa co RI
        RETI                      ; tro ve chuong trinh chinh
TRANS:  CLR    TI                 ; xoa co TI
        RETI                      ; tro ve chuong trinh chinh
        END
```

- Trong chương trình này, cần lưu ý đến cờ TI và RI. Tại thời điểm một byte được ghi vào SBUF thì byte này được định khung và truyền nối tiếp. Khi bit cuối cùng (stop) được truyền đi cờ TI được bật cao và gây ra ngắt nối tiếp. Trong trình phục vụ ngắt nối tiếp, cần kiểm tra cả cờ TI và RI vì cả hai để có thể gọi ngắt chỉ có một ngắt chung cho cả phát và thu dữ liệu.

4.5. Lập trình ngắt truyền thông nối tiếp – Một số ví dụ (3)

Ví dụ 4.8. Viết chương trình cho VĐK 8051 thực hiện nhận dữ liệu từ cổng P1 và gửi liên tục tới cổng P2, trong khi đó dữ liệu nhận được từ cổng nối tiếp gửi ra cổng P0. Giả thiết tần số XTAL là 11.0592MHz và tốc độ truyền 9600 baud.

Giải

```
ORG    0000h
LJMP   MAIN
;--- vector ngắt truyền thông nối tiếp
ORG    0023h
LJMP   SERIAL          ; nhảy đến ISR nối tiếp
;--- chương trình chính
ORG    0030h
MAIN:  MOV    P1, #0FFh      ; đặt P1 làm cổng vào
        MOV    TMOD, #20h    ; Timer 1, chế độ 2 tự nạp lại
        MOV    TH1, #0FDh    ; chọn tốc độ baud = 9600
        MOV    SCON, #50h     ; khung 8 bit, 1 stop, cho phép (REN)
        MOV    IE, #10010000h ; cho phép ngắt nối tiếp
        SETB   TR1            ; khởi động Timer 1
```

4.5. Lập trình ngắt truyền thông nối tiếp – Một số ví dụ (4)

Ví dụ 4.8. (tiếp)

```
BACK:  MOV    A, P1          ; nhan du lieu tu cong P1
        MOV    P2, A         ; gui ra cong P2
        SJMP   BACK         ; lap lai vong lap
; ----trinh phuc vu ngat truyen thong noi tiep
        ORG    100h
SERIAL: JB     TI, TRANS     ; nhay neu co TI cao
        MOV    A, SBUF       ; neu khong thi nhan du lieu
        MOV    P0, A         ; gui du lieu nhan duoc ra cong P0
        CLR    RI           ; xoa co RI
        RETI                ; tro ve chuong trinh chinh
TRANS: CLR    TI            ; xoa co TI
        RETI                ; tro ve chuong trinh chinh
END
```



Chương 5. VI VI XỬ LÝ ARM



MỤC TIÊU

- Hiểu được kiến trúc tổng quan vi xử lý ARM
- Hiểu được cấu trúc chung của tập lệnh ARM, mô hình giao tiếp của ARM
- Đặc điểm các dòng ARM

NỘI DUNG

5.1. Tổng quan về vi xử lý ARM

5.2. Kiến trúc cơ bản của vi xử lý ARM

5.3. Cấu trúc load – store

5.4. Cấu trúc tập lệnh

5.5. Mô hình giao tiếp trong vi xử lý ARM

5.6. Đặc điểm của các dòng vi xử lý ARM

5.1. Tổng quan về vi xử lý ARM

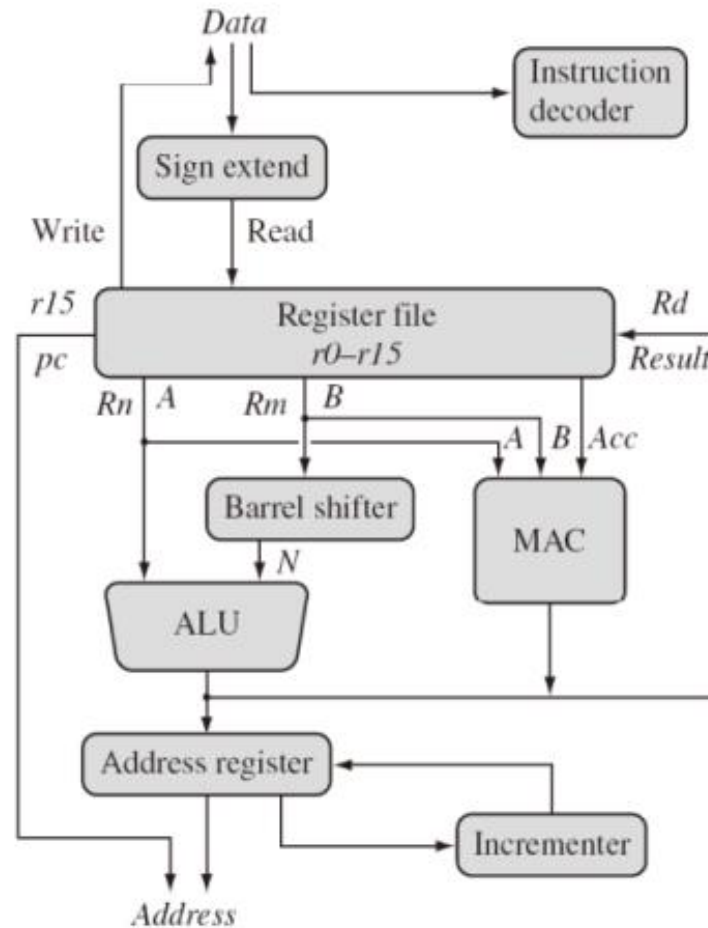
ARM là một trong những họ CPU dựa trên kiến trúc RISC (máy tính tập lệnh giản thiểu) được phát triển bởi Advanced RISC Machines (ARM).

ARM là một loại cấu trúc vi xử lý RISC 32 bit và 64 bit. RISC được thiết kế để thực hiện một số lượng nhỏ hơn các loại hướng dẫn máy tính để chúng có thể hoạt động ở tốc độ cao hơn, thực hiện thêm hàng triệu phép tính mỗi giây (MIPS). Bằng cách loại bỏ các phép tính không cần thiết và tối ưu hóa các lộ trình.

Kiến trúc ARM được được thiết kế chuyên dụng cho các ứng dụng nhúng. Do đó, hiện thực hóa chip ARM được thiết kế để cho các ứng dụng nhỏ, nhưng có hiệu năng cao, tiêu thụ ít năng lượng

5.2. Kiến trúc cơ bản của vi xử lý ARM (1)

Kiến trúc cơ bản của vi xử lý ARM, Hình 5.1



Hình 5.1. Kiến trúc ARM

5.2. Kiến trúc cơ bản của vi xử lý ARM (2)

ARM được thiết kế theo kiến trúc RISC, do vậy nó chứa các kiến trúc RISC chung:

- Các thanh ghi đồng dạng
- Kiến trúc dạng Load-Store. Các địa chỉ Load/Store chỉ được xác định từ nội dung thanh ghi và các chỉ lệnh
- Các kiểu đánh địa chỉ đơn giản
- Các chỉ lệnh có độ dài cố định và đồng dạng, do đó đơn giản hóa việc giải mã các câu lệnh
- Thiết kế tối giản số chu kỳ xung nhịp cho một chỉ lệnh

5.2. Kiến trúc cơ bản của vi xử lý ARM (3)

Ngoài ra, kiến trúc ARM có thể cung cấp:

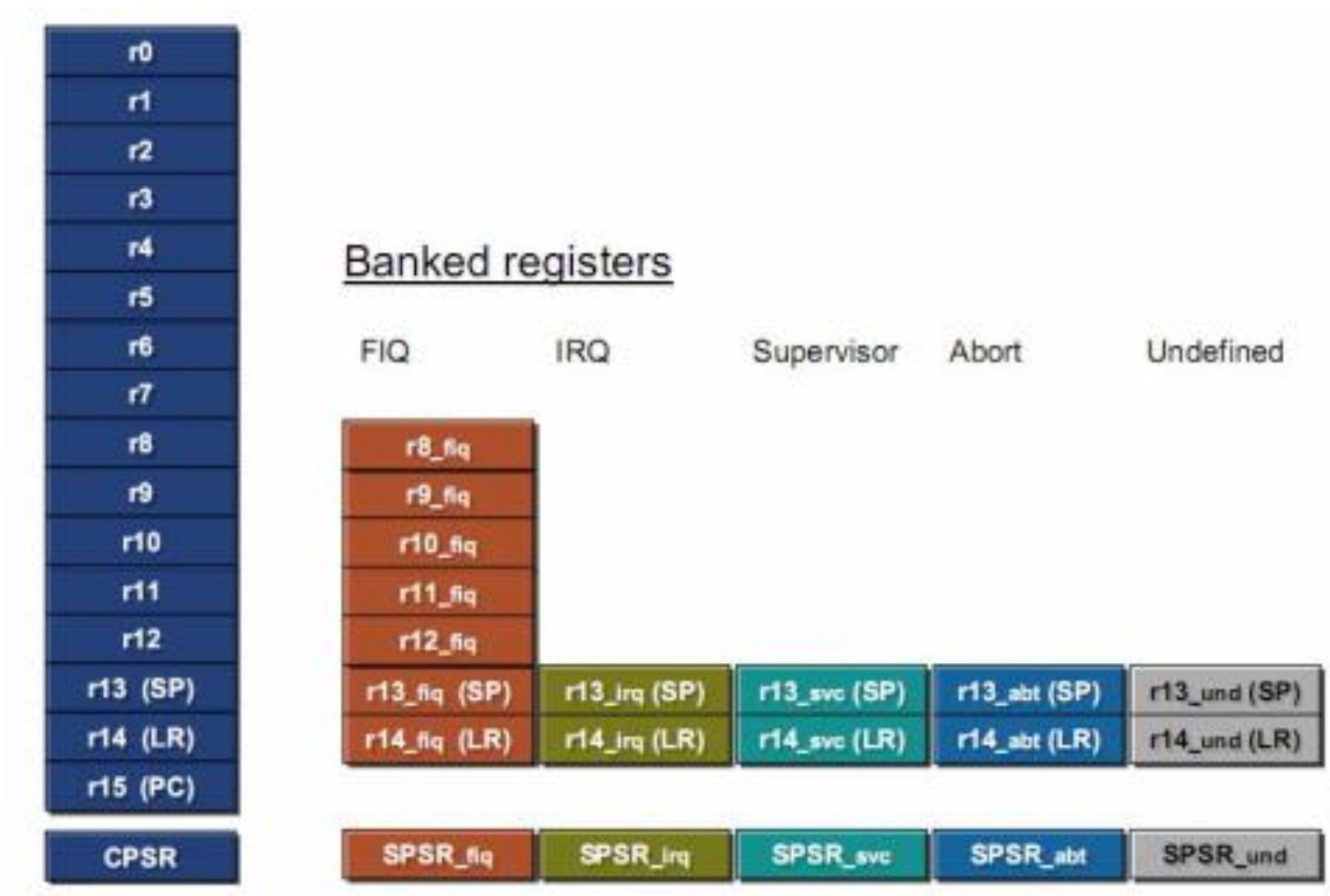
- Điều khiển cả khối ALU và bộ dịch chuyển (shifter) trong các lệnh xử lý dữ liệu để tối ưu hóa việc sử dụng ALU và bộ dịch chuyển.
- Các chế độ địa chỉ tự tăng hoặc tự giảm để tối ưu hóa các lệnh vòng lặp.
- Các lệnh Load/Store để tối đa dữ liệu truyền qua.

Nhờ các tối ưu trên nền kiến trúc RISC căn bản, ARM có thể đạt được một sự cân bằng giữa hiệu năng cao, kích thước mã nguồn ít, công suất tiêu thụ thấp.

5.2. Kiến trúc cơ bản của vi xử lý ARM (4)

Các thanh ghi

Tổ chức các thanh ghi của ARM được thực hiện trên hình 5.2



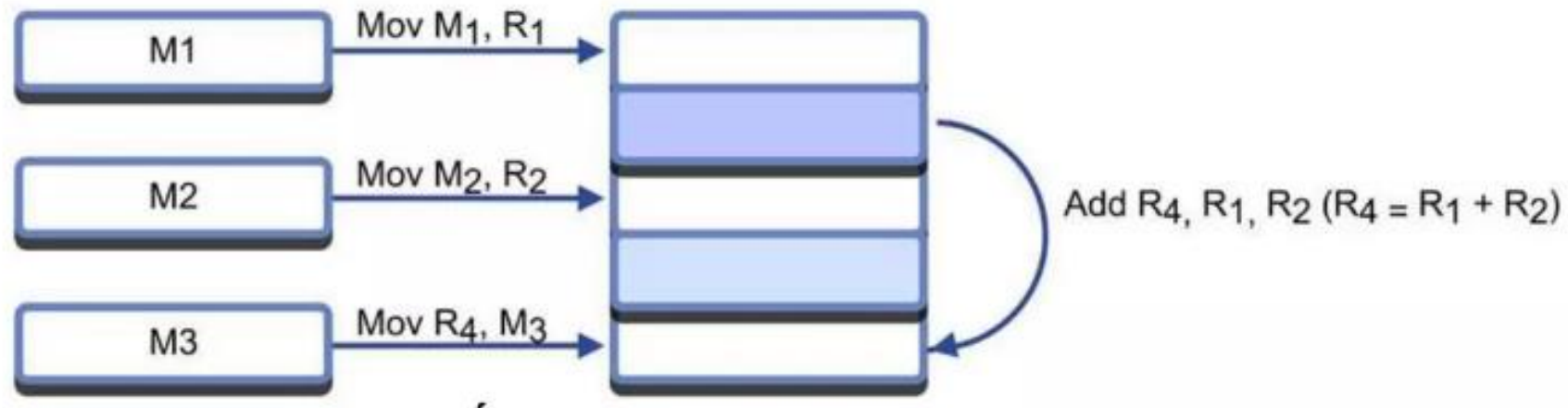
Hình 5.2. Tổ chức thanh ghi của ARM

5.2. Kiến trúc cơ bản của vi xử lý ARM (5)

Các thanh ghi

- ARM có 37 thanh ghi, trong đó 31 thanh ghi đa dụng. Tuy nhiên tại một thời điểm chỉ có 16 thanh ghi đa dụng và 2 thanh ghi trạng thái hiển thị. Các thanh ghi khác ở dạng ẩn, chỉ hiển thị ở một số chế độ hoạt động riêng.
- Các thanh ghi đa dụng có thể dùng để lưu dữ liệu hoặc địa chỉ. Các thanh ghi này được đánh dấu bằng ký hiệu r. Tất cả các thanh ghi đều 32 bit. Trong các thanh ghi đa dụng có 3 thanh ghi được dùng cho các chức năng hoặc nhiệm vụ đặt biệt : r13, r14, r15
 - R13 được dùng làm stack pointer (SP)
 - R14 là thanh ghi kết nối (LR) chứa địa chỉ quay lại của chương trình khi chương trình chạy một hàm con
 - R15 là bộ đếm chương trình (PC) và chứa địa chỉ của lệnh tiếp theo
- Hai thanh ghi trạng thái: thanh ghi TT chương trình hiện tại (CPSR) để giám sát các trạng thái hoạt động hiện tại; thanh ghi TT chương trình lưu (SPSR) để lưu trữ giá trị của CPSR khi có một ngoại lệ xảy ra.

5.3. Cấu trúc Load - Store



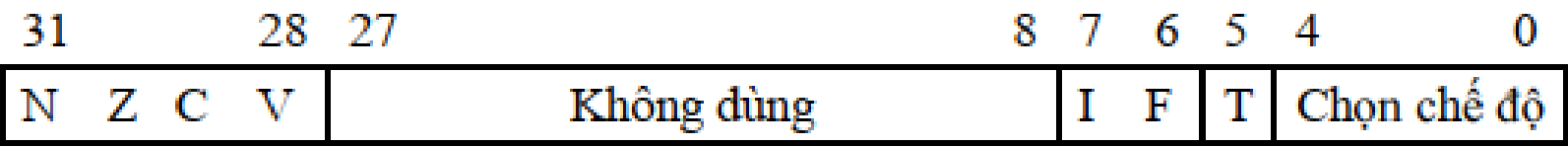
Hình 5.3. Cấu trúc Load - Store

ARM là bộ vi xử lý dựa trên kiến trúc RISC, do đó hỗ trợ kiến trúc nạp và lưu trữ (Load – Store Architecture). Để thực hiện lệnh xử lý dữ liệu, các toán hạng phải được nạp vào một tập thanh ghi trung tâm, các phép toán dữ liệu phải được thực hiện trên các thanh ghi này và kết quả sau đó được lưu lại trong bộ nhớ.

5.4. Cấu trúc tập lệnh ARM (1)

ARM cung cấp khả năng thực hiện một cách có điều kiện hầu hết các lệnh dựa trên tổ hợp trạng thái của các cờ điều kiện trong thanh ghi CPSR.

Thanh ghi CPSR cho biết trạng thái của chương trình hiện tại và được mô tả trong Hình 5.4



Hình 5.4. Vị trí các bit trên thanh ghi CPSR

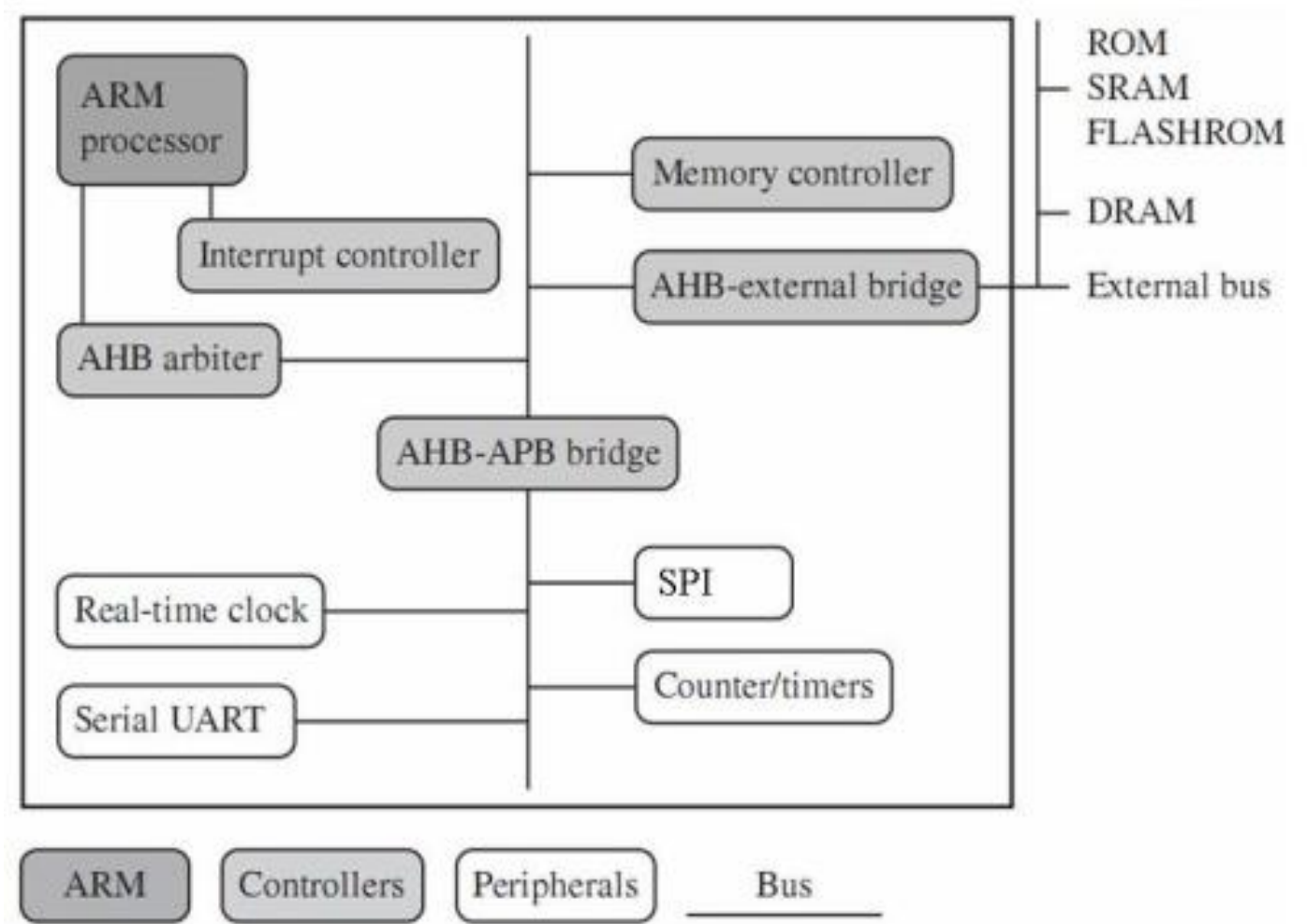
5.4. Cấu trúc tập lệnh ARM (2)

Tất cả lệnh của ARM đều là 32 bit:

- Có cấu trúc dạng load-store.
- Cấu trúc lệnh định dạng ba địa chỉ (nghĩa là địa chỉ của hai toán hạng nguồn và toán hạng đích đều là các địa chỉ riêng biệt).
- Mỗi một lệnh thực thi một điều kiện.
- Có cả lệnh load-store nhiều thanh ghi đồng thời.
- Có khả năng dịch bit kết hợp với thực thi lệnh ALU trong chỉ một chu kỳ máy.
- Chế độ Thumb code: là một chế độ đặc biệt của ARM dùng để tăng mật độ mã bằng cách nén lệnh 32 bit thành 16 bit. Một phần cứng đặc biệt sẽ giải nén lệnh Thumb 16 bit thành lệnh 32 bit

5.5. Mô hình giao tiếp trong ARM

Vi điều khiển ARM được thực thi trên hệ thống kiến trúc các bus truyền dữ liệu đa chức năng của vi điều khiển. Bao gồm bộ vi xử lý ARM kết nối qua hệ thống bus truyền dữ liệu hiệu suất cao để đồng bộ nhanh với SRAM, các bus giao tiếp ngoài, và cầu nối tới các bus truyền ngoại vi công suất thấp, được mô tả trong Hình 5.5.



Hình 5.5. Mô hình giao tiếp trong ARM

5.5. Mô hình giao tiếp trong ARM

Các khối chức năng trong vi điều khiển ARM bao gồm:

- Bộ xử lý ARM;
- Bộ điều khiển ngắt;
- Bộ phân xử bus truyền hiệu suất cao (AHB-Advanced High-performance Bus);
- Bộ điều khiển bộ nhớ;
- SRAM;
- EPROM hoặc Flash;
- DRAM;
- Cầu nối AHB – APB (Advanced Peripheral Bus: Bus truyền ngoại vi tối ưu) - Cầu nối ngoài AHB;
- Bộ đếm/định thời;
- Khối SPI (Serial Peripheral Interface): Khối giao tiếp các thiết bị ngoại vi nối tiếp;
- Khối Serial UART (Serial Universal Asynchronous Receiver/Transmitter): Khối giao tiếp nối tiếp truyền/thu không đồng bộ đa năng.

5.6. Đặc điểm các dòng vi xử lý ARM

- **Cortex-A:** Là nhân chip được sử dụng nhiều nhất trên các dòng chip Android, Cortex-A sở hữu công cụ chính là SMID (tập lệnh đa dữ liệu) có khả năng thực hiện các tác vụ như truy cập bộ nhớ và xử lý dữ liệu song song trên một tập hợp vector.
- **Cortex-R:** Đem đến nhiều giải pháp tính toán hiệu suất cải tiến dành cho những hệ thống nhúng đòi hỏi độ bảo mật cao và đảm bảo tính sẵn sàng cùng khả năng đáp ứng thời gian thực.
- **Cortex-M:** Là nhân chip thường được ứng dụng trong ngành công nghiệp xe hơi, với đặc trưng là kích cỡ nhỏ và không chiếm nhiều không gian trong cấu trúc hệ thống linh kiện ô tô.
- **Ethos-N:** Nổi danh trong ngành công nghiệp AI và được ứng dụng rất nhiều do khả năng tương thích với các tác vụ máy học.
- **Ethos-U:** Được xem là “bản mini” rút gọn từ nhân Ethos-N, Ethos-U sẽ vận hành tương tự như một bộ đồng xử lý Co-processor.
- **Neoverse:** Có tuổi đời còn khá ngắn, ra đời từ tháng 10/2018, Neoverse được biết đến như nhân xử lý chuyên phục vụ các hệ thống máy chủ và trung tâm dữ liệu.
- **SecurCore:** Đúng như tên gọi, SecurCore đặt yếu tố bảo mật lên hàng đầu và là lõi chip được ARM Holdings, Ltd thiết kế dành cho các thẻ thông minh, rất phổ biến trong ngành ngân hàng, thanh toán.

The End