Hệ nhúng
(Embedded Systems)

IT4210

Đỗ Công Thuần

Khoa Kỹ thuật máy tính, Trường CNTT&TT

Đại học Bách khoa Hà Nội

Email: thuandc@soict.hust.edu.vn

ONE LOVE. ONE FUTURE.

Giới thiệu môn học

- Tên học phần: **Hệ nhúng**
- Mã học phần: **IT4210 (3-0-1-6)**
- Thời lượng:
  - 16.5 buổi lý thuyết (3 tiết/buổi)
  - 3 buổi thực hành (5 tiết/buổi)
- Yêu cầu kiến thức nền tảng:
  - Kiến trúc máy tính
  - Vi xử lý
  - Lập trình C

Mục tiêu môn học

- Nắm được kiến trúc tổng quan, đặc điểm và hoạt động của một hệ nhúng
- Biết thiết kế hệ nhúng cơ bản (nguyên lý thiết kế mạch, …)
- Nắm được kiến trúc vi điều khiển (Intel, ARM)
- Lập trình vi điều khiển từ cơ bản đến nâng cao với các dòng vi điều khiển phổ biến
- Lập trình với hệ điều hành nhúng

Đánh giá học phần

1. Đánh giá quá trình: **40%**
   - Bài tập về nhà
   - Chuyên cần
   - Các bài thực hành, nhóm **4 SV/nhóm**
2. Đánh giá cuối kỳ: **60%**
   - Làm project cuối kỳ, nhóm **4 SV/nhóm**
   - Yêu cầu sinh viên tự chọn nhóm và đăng kí đề tài.
     Chú ý: *danh sách đề tài sẽ được cập nhật sau!*

Tài liệu tham khảo

- Textbook/Lecture notes:
  - Peter Marwedel, ***Embedded System Design: Embedded Systems Foundations of Cyber-Physical Systems, and the Internet of Things***, Spinger, 4th edition, 2021.
  - Edward A. Lee and Sanjit A. Seshia, ***Introduction to Embedded Systems: A Cyber-Physical Systems Approach***, MIT Press, 2nd edition, 2017.
  - Tammy Noergaard, ***Embedded Systems Architecture: A Comprehensive Guide for Engineers and Programmers***, Elsevier, 2nd edition, 2013.
  - Han-Way Huang, Leo Chartrand, ***Microcontroller: An Introduction to Software & Hardware Interfacing***, Cengage Learning, 2004.
  - Lectures in Embedded Systems from *Univ. of Cincinnati (EECE 6017C)*, *Univ. of California, Berkeley (EECS 149)*, *Univ. of Pennsylvania (ESE 350)*, *Univ. of Kansas (EECS388).*
  - *…*
- Manuals/Handbooks/Internet
  - Atmel, Microchip, Texas Instruments, Keil…
  - Keil ASM51
  - Arduino IDE
  - …

Nội dung học phần

- Chương 1: Giới thiệu về Hệ nhúng
- Chương 2: Thiết kế phần cứng Hệ nhúng
- Chương 3: Lập trình với 8051
- Chương 4: Ghép nối ngoại vi với 8051
- Chương 5: Arduino
- Chương 6: Ghép nối nối tiếp
- Chương 7: Ghép nối với thế giới thực
- Chương 8: Kiến trúc ARM
- Chương 9: RTOS và FreeRTOS

Nội dung học phần

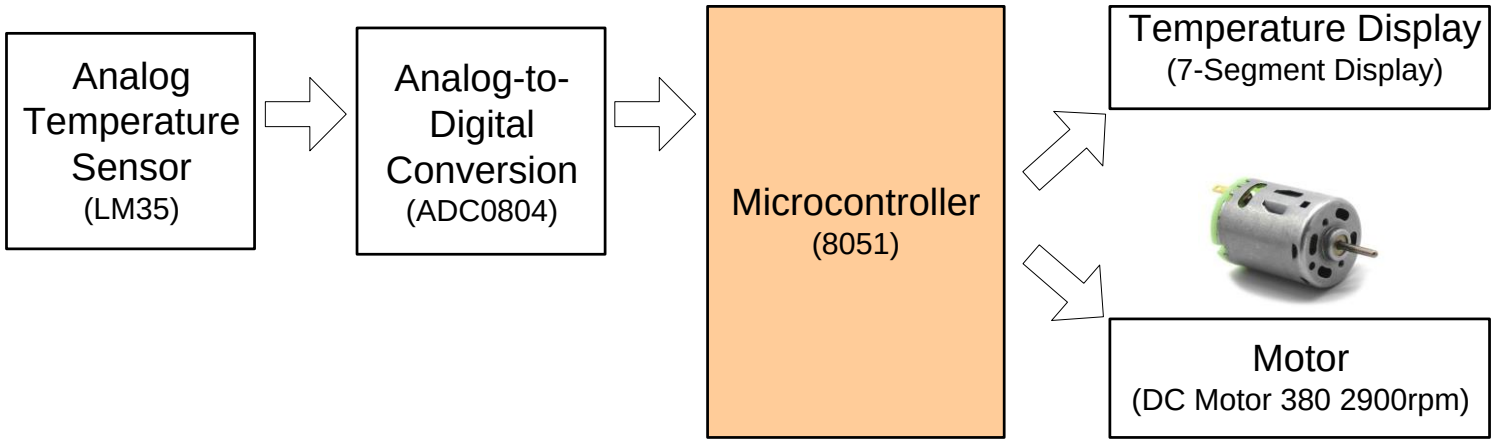
- Chương 1: Giới thiệu về Hệ nhúng
- Chương 2: Thiết kế phần cứng Hệ nhúng
- Chương 3: Lập trình với 8051
- Chương 4: Ghép nối ngoại vi với 8051
- Chương 5: Arduino
- Chương 6: Ghép nối nối tiếp
- Chương 7: Ghép nối với thế giới thực
- Chương 8: Kiến trúc ARM
- Chương 9: RTOS và FreeRTOS

Chương 3

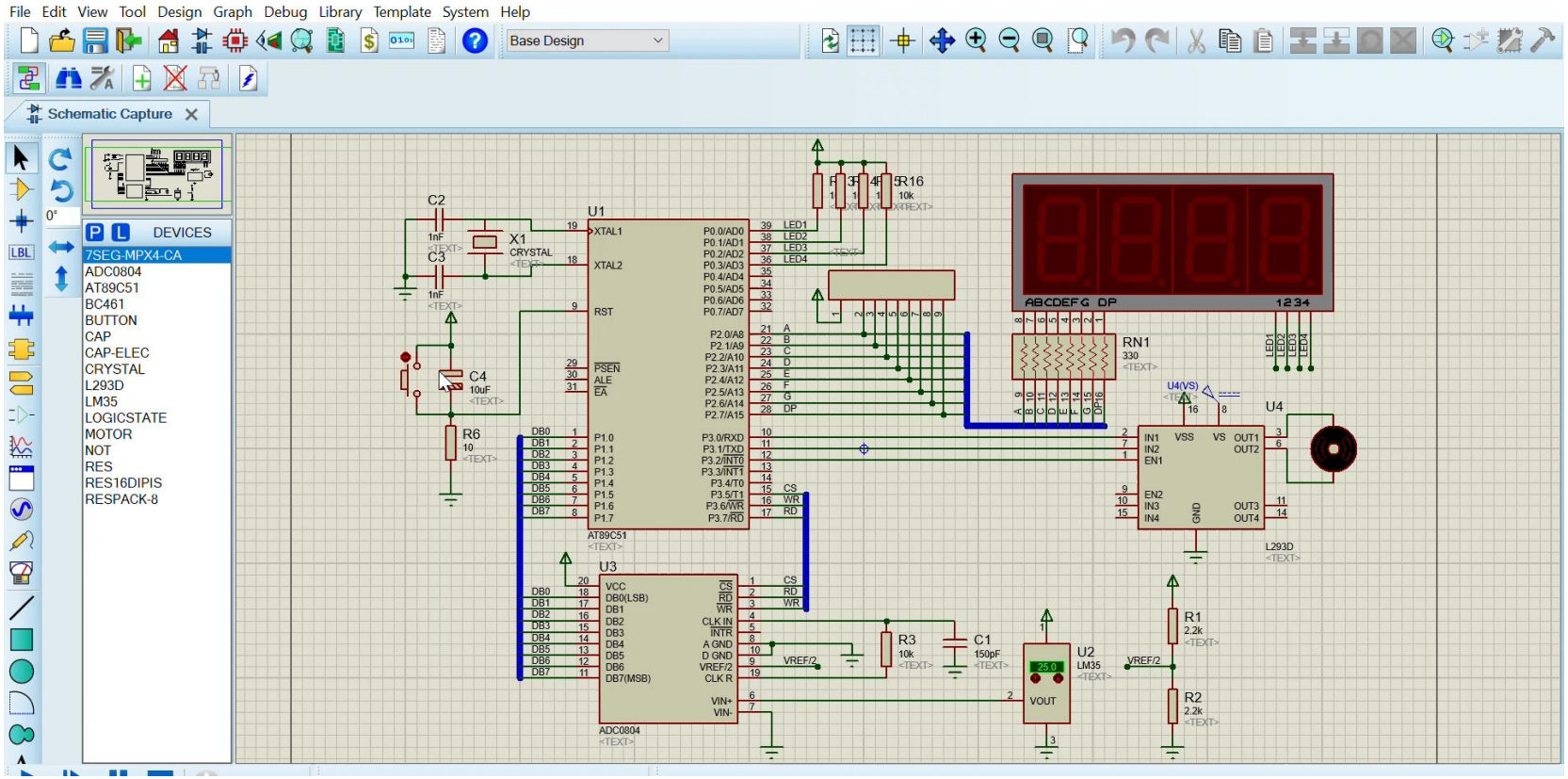
Lập trình với 8051

Minh họa một hệ nhúng

- Hệ thống điều khiển động cơ chạy/dừng tùy thuộc vào nhiệt độ môi trường.



Demo Video


File Edit View Tool Design Graph Debug Library Template System Help
Base Design
Schematic Capture
DEVICES
7SEG-MPX4-CA
ADC0804
AT89C51
BC461
BUTTON
CAP
CAP-ELEC
CRYSTAL
L293D
LM35
LOGICSTATE
MOTOR
NOT
RES
RES16DIPIS
RESPACK-8
U1
AT89C51
U3
ADC0804
U2
LM35
U4
L293D
RN1
X1
CRYSTAL
C1
C2
C3
C4
R1
R2
R3
R6
R16

Nội dung

- Giới thiệu
- Tập lệnh
- Lập trình hợp ngữ

Giới thiệu

Đơn vị xử lý thông tin trong các hệ thống nhúng:

- **ASIC** (*Application-Specific Integrated Circuit*)
- **Reconfigurable Logic** (e.g.: FPGA)
- **Microprocessor/Microcontroller**
  - Intel (**8051**, 8051XA, 8096, x86, …)
  - Atmel AVR (ATmega8, ATmega328, Atmel 251 …)
  - PIC (PIC1x, PIC2x, PIC3x …)
  - ARM (**ARM Cortex-M**, Cortex-A, Cortex-R …)
  - MIPS (MIPS II, MIPS III, …)
  - …

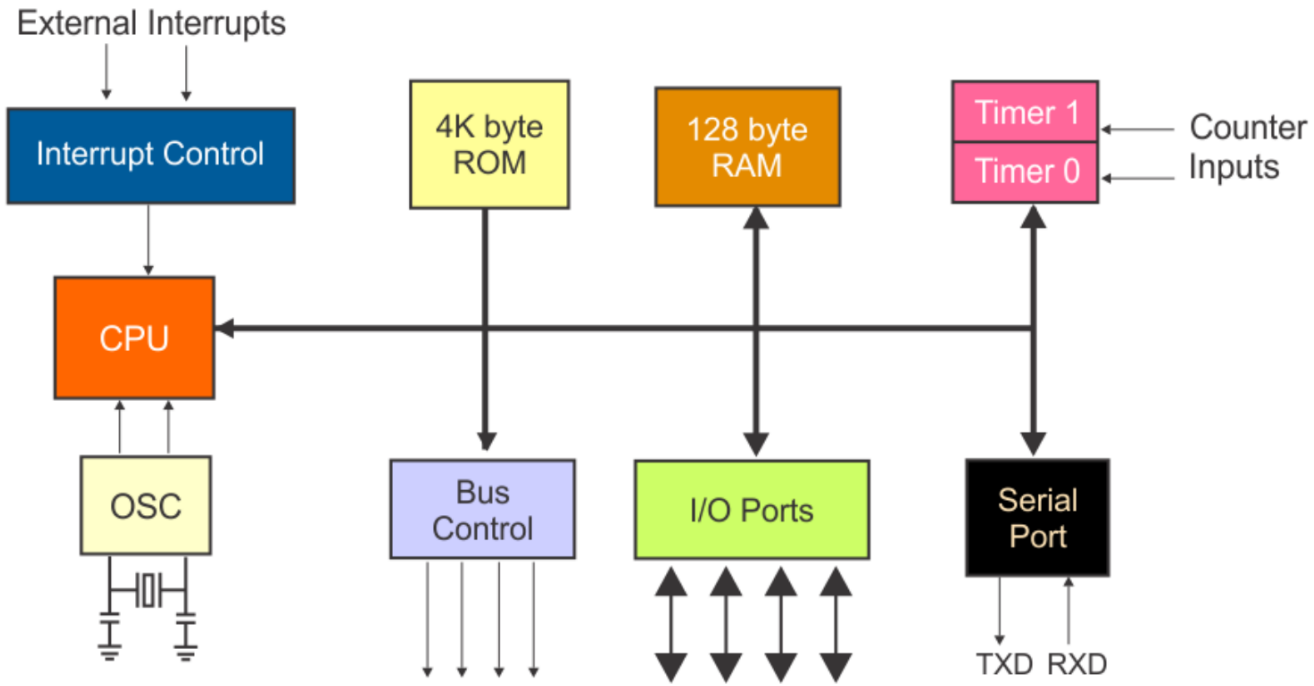
Vi điều khiển 8051

- Vi điều khiển 8051 được Intel giới thiệu đầu tiên năm 1980 với tên MCS-51.
- Rất phổ biến những năm 1980s – 1990s.
- Intel cho phép các hãng khác phát triển vi điều khiển dựa trên lõi của 8051.

  → Rất nhiều phiên bản vi điều khiển dẫn xuất từ 8051 đã được chế tạo và còn phổ biến tới ngày nay (vd: IP cores).

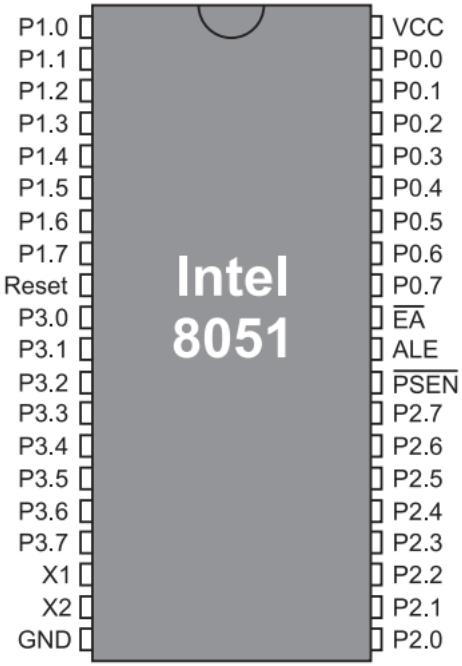
- Một số vendor tiêu biểu: Atmel, SiliconLab, TI …

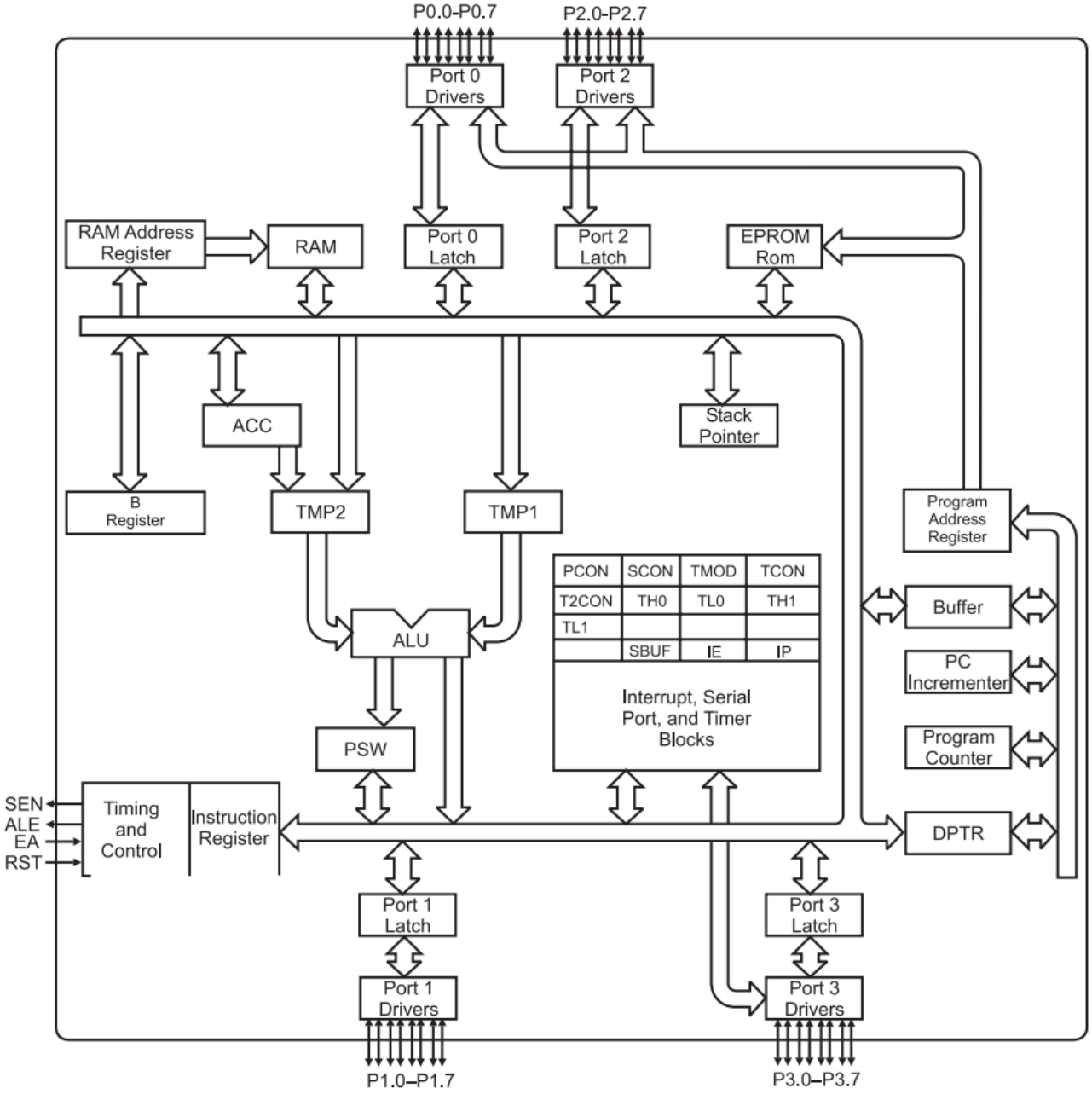
Sơ đồ khối



8-bit Processor, Harvard Architecture
Operating Frequency: 24MHz max
128 byte RAM
4KB ROM (int.) – 64KB (ext.)

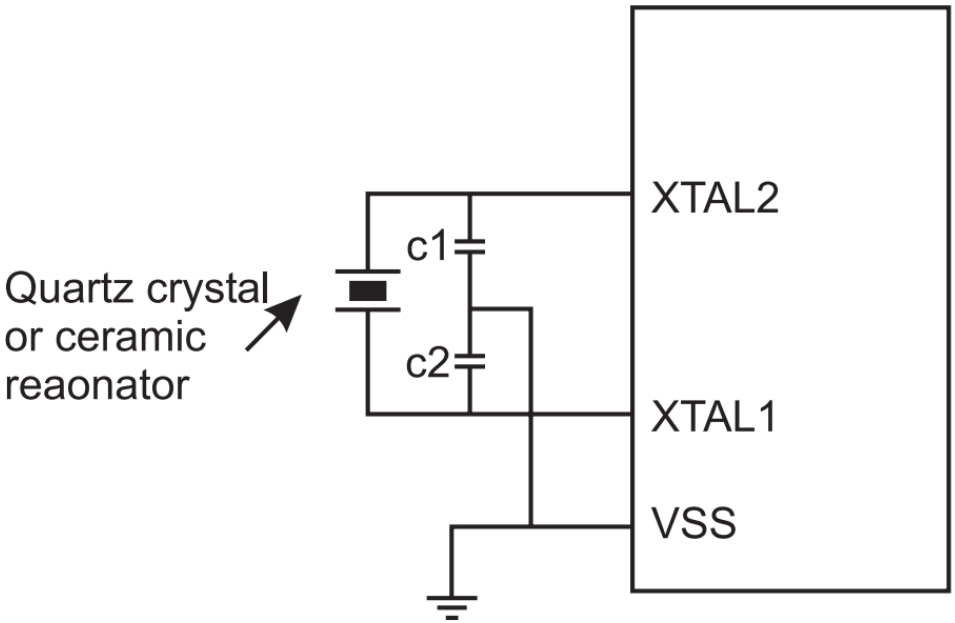
4 8-bit Ports (I/O Ports): P0, P1, P2, P3
1 Serial Port (UART)
2 External Interrupts (INT0 & INT1)
2 Timers/Counters (Timer 0 & Timer 1)


P1.0
P1.1
P1.2
P1.3
P1.4
P1.5
P1.6
P1.7
Reset
P3.0
P3.1
P3.2
P3.3
P3.4
P3.5
P3.6
P3.7
X1
X2
GND
VCC
P0.0
P0.1
P0.2
P0.3
P0.4
P0.5
P0.6
P0.7
EA
ALE
PSEN
P2.7
P2.6
P2.5
P2.4
P2.3
P2.2
P2.1
P2.0
Intel
8051


P0.0-P0.7
P2.0-P2.7
Port 0 Drivers
Port 2 Drivers
RAM Address Register
RAM
Port 0 Latch
Port 2 Latch
EPROM Rom
ACC
Stack Pointer
B Register
TMP2
TMP1
Program Address Register
PCON
SCON
TMOD
TCON
T2CON
TH0
TL0
TH1
TL1
SBUF
IE
IP
Buffer
ALU
PC Incrementer
Interrupt, Serial Port, and Timer Blocks
Program Counter
PSW
SEN
ALE
EA
RST
Timing and Control
Instruction Register
DPTR
Port 1 Latch
Port 3 Latch
Port 1 Drivers
Port 3 Drivers
P1.0–P1.7
P3.0–P3.7

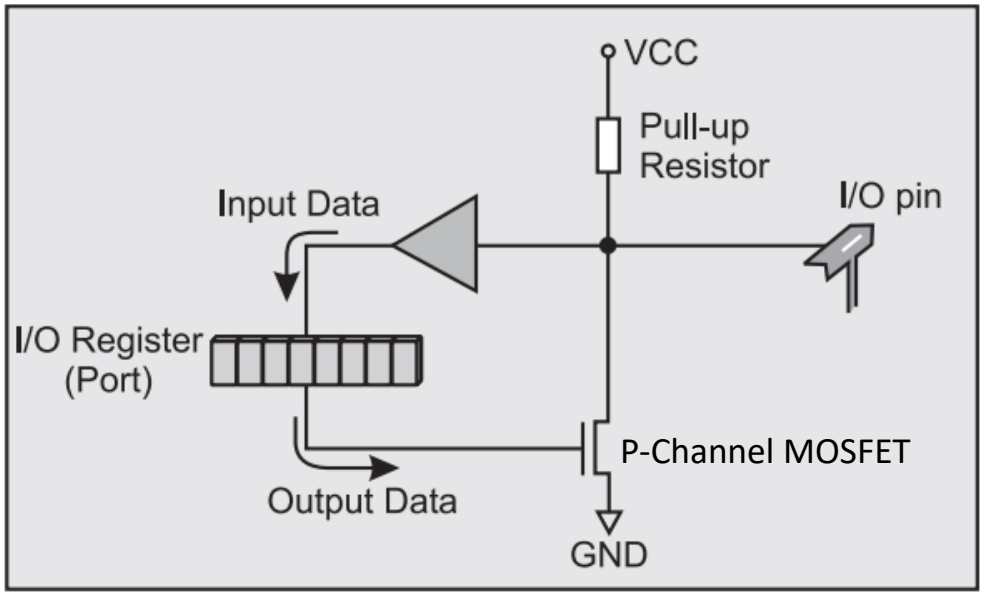
Oscillator

- 8051 cần sử dụng mạch tạo dao động ngoài.
  - Tần số hoạt động thường khoảng 12 MHz (max 40 Mhz)
  - **1 machine cycle = 12 clock cycles**



I/O Ports

- 8051 có 4 I/O 8-bit ports
- Mỗi port/pin có thể cấu hình output (= 0) hoặc input (= 1).



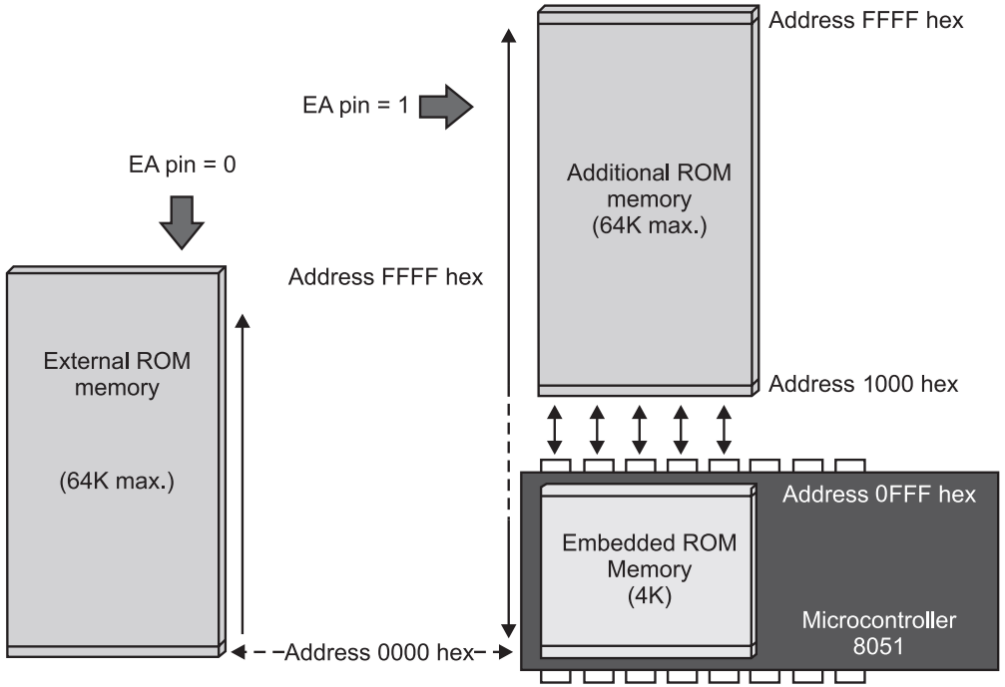
Tổ chức bộ nhớ

- **Bộ nhớ chương trình** (*Program Memory*): chứa chương trình điều khiển (*firmwares*).
- **Bộ nhớ dữ liệu** (*Data Memory*): chứa dữ liệu và các thanh ghi SFRs phục vụ hoạt động của CPU.
- Bộ nhớ chương trình và Bộ nhớ dữ liệu có không gian địa chỉ riêng biệt.

Bộ nhớ chương trình (Program Memory)

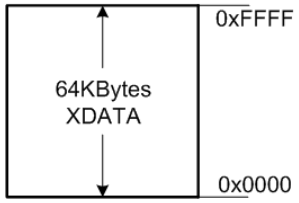
- Bus địa chỉ 16-bit → có thể quản lý tới 64KB
- Hỗ trợ ghép nối bộ nhớ ngoài qua chân EA (*External Access*), kết hợp với chân ALE (*Address Latch Enable*) + chân PSEN (*Program Store Enable*)

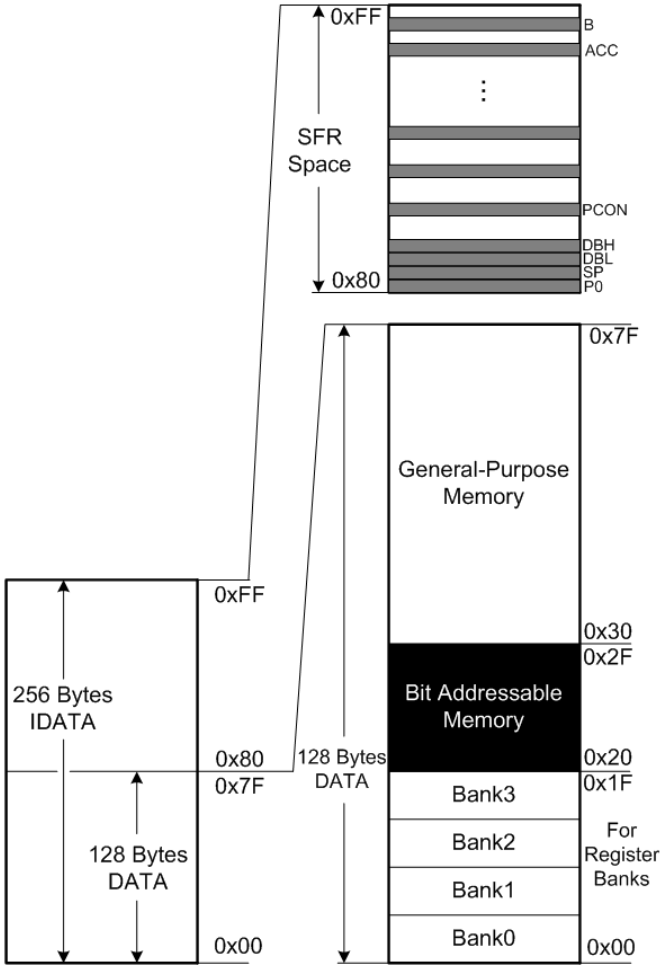
- **EA = 0:**
  - Dùng ext. ROM, bỏ qua int. ROM
- **EA = 1:**
  - Dùng int. ROM rồi đến ext. ROM



Bộ nhớ dữ liệu (Data Memory)

- **128 bytes (00h – 7Fh)**
  - 32 bytes (00h – 1Fh) dùng cho Register Bank (Bank 0 – 3) và Stack (Bank 1)
  - 16 bytes (20h – 2Fh) dùng cho *bit-addressable Read/Write memory*
  - 80 bytes (30h – 7Fh) dùng cho *general-purpose memory* (*scrachpad area*)
- **128 bytes (80h – FFh) dùng cho SFRs** (*Special Function Registers*)
- **Có thể mở rộng đến 64KB RAM**





Tập thanh ghi (Register File)

- **R0-R7**: các thanh ghi đa năng, được ánh xạ vào 1 trong 4 Register Banks (Bank 0 – 3)
- Các thanh ghi đặc biệt:
  - **A** (*accumulator):* thanh ghi đa năng
  - **B** (*auxiliary accumulator*): thanh ghi đa năng, dùng cho lệnh MUL và DIV
  - **DPTR** (*data pointer*): DPTR = DPH + DPL
  - **PC** (*program counter*): địa chỉ của lệnh tiếp theo
  - **PSW** (*program status word*): trạng thái của CPU
  - **SFRs** (*special function registers*): dùng cho các chức năng điều khiển ngoại vi

Data Pointer Register

- Không phải là thanh ghi vật lý
- Để chứa dữ liệu (*code*, *data*) hoặc kết quả tính toán trung gian
- 16-bit DPTR = 8-bit DPH + 8-bit DPL
- 16 bits được dùng để định địa chỉ bộ nhớ (*memory addressing*)
  - Ví dụ: ghi giá trị 46h vào địa chỉ 2500h

```
MOV A, #46h
MOV DPTR, #2500h ; DPTR ← Address
MOVX @DPTR, A    ; write content to external RAM at
                 ; address 2500h
```

Program Counter

- Thanh ghi 16-bit
- Luôn chứa địa chỉ của câu lệnh tiếp theo sẽ được thực thi
  - Khi 8051 bị reset, PC = 0000h.
  - Với 2-byte instruction, PC = PC + 2
  - Với 3-byte instruction, PC = PC + 3

Program Status Word

(MSB)							(LSB)
CY	AC	F0	RS1	RS0	OV	-	P

Symbol	Position	Name and Significance
CY	PSW.7	Carry flag
AC	PSW.6	Auxiliary Carry flag. (For BCD operations.)
F0	PSW.5	Flag 0 (Available to the user for general purposes.)
RS1	PSW.4	Register bank Select control bits 1 & 0. Set/cleared by software to determine working register bank (see Note).
RS0	PSW.3	
OV	PSW.2	Overflow flag.
-	PSW.1	(reserved)
P	PSW.0	Parity flag. Set/cleared by hardware each instruction cycle to indicate and odd/even number of "one" bits in the accumulator, i.e., even parity.

(0.0)-Bank 0(00H-07H)
(0.1)-Bank 1(08H-0FH)
(1.0)-Bank 2(10H-17H)
(1.1)-Bank 3(18H-1FH)

→ **PSW chứa các bit phản ánh trạng thái của CPU**

Special Function Registers (SFRs)

ACC – Accumulator
B – Aux. accumulator
PWS – uC state register
DPH, DPL – 16bit data pointer
SP – Instruction & Stack pointers
P0,P1,P2,P4 – I/O ports data registers
IE – Interrupt mask
IP,IPH – Interrupt priority registers
PCON – Power-save control register
TCON,TMOD – Timer control registers
TL0,TH0 – 16-bit timer register
AUXR – Aux. control bits

0F8H									0FFH
0F0H	B 00000000								0F7H
0E8H									0EFH
0E0H	ACC 00000000								0E7H
0D8H									0DFH
0D0H	PSW 00000000								0D7H
0C8H									0CFH
0C0H									0C7H
0B8H	IP XX000000								0BFH
0B0H	P3 11111111								0B7H
0A8H	IE 0X000000								0AFH
0A0H	P2 11111111		AUXR1 XXXXXXX0				WDTRST XXXXXXXX		0A7H
98H	SCON 00000000	SBUF XXXXXXXX							9FH
90H	P1 11111111								97H
88H	TCON 00000000	TMOD 00000000	TL0 00000000	TL1 00000000	TH0 00000000	TH1 00000000	AUXR XXX00XX0		8FH
80H	P0 11111111	SP 00000111	DP0L 00000000	DP0H 00000000	DP1L 00000000	DP1H 00000000		PCON 0XXX0000	87H

Địa chỉ thanh ghi trong RAM

Giá trị khởi tạo của thanh ghi

Biểu diễn dữ liệu

- Các hằng (*literal*)
  - Số nhị phân: 1011b, 1011B, ...
  - Số thập phân: 35, 35d, 35D, ...
  - Số Hexa: 4Ah, 0ABCDh, 0FFFFH, ...
  - Kí tự: "A", 'HELLO', "Bach Khoa", ...
- Tất cả các kiểu dữ liệu trên sau đó đều được trình dịch Assembler dịch ra mã nhị phân.
- Mỗi kí tự được dịch thành mã ASCII tương ứng
  - Chương trình không phân biệt 'A' với 41h hay 65

Kiểu dữ liệu

- Chỉ có một kiểu dữ liệu là kiểu dữ liệu 8 bit
- Để định nghĩa một dữ liệu kiểu byte, cần sử dụng chỉ dẫn **DB**

```
        ORG    500H
DATA1:  DB     28                ;Số thập phân (=1CH)
DATA2:  DB     00110101B         ;Số nhị phân (=35H)
DATA3:  DB     39H               ;Số dạng Hexa
        ORG    510H
DATA4:  DB     "2591"            ;Các số ASCII
        ORG    518H
DATA5:  DB "My name is Binh"     ;Ký tự ASCII
```

Các chỉ dẫn

- **ORG**: Báo địa chỉ bắt đầu
- **EQU**: dùng để định nghĩa hằng số
  - VD: COUNT EQU 25
- **END**: báo kết thúc file mã nguồn

Tập lệnh của 8051

- Số lệnh: khoảng 110 lệnh hợp ngữ.
  - Mô tả một dòng lệnh hợp ngữ:

[nhãn:] [mã thao tác] [các toán hạng] [;chú giải]

Ví dụ: start: MOV A, #25 ; A = 25

- Các phương pháp định địa chỉ (*Addressing Modes*):
  - Định địa chỉ tức thì (*immediate addressing*)
  - Định địa chỉ thanh ghi (*register addressing*)
  - Định địa chỉ trực tiếp (*direct addressing*)
  - Định địa chỉ gián tiếp (*indirect addressing*)
  - Định địa chỉ thanh ghi chỉ số, cơ sở (*index+base register addressing*)

Truy cập bộ nhớ

Định địa chỉ tức thì

- Toán hạng nguồn là hằng số
- Giá trị của hằng số được xác định ngay sau byte *inst. opcode* trong *Program Memory*
- Sử dụng "#" cho "*immediate data*"
- Ví dụ:

```
ADD A, #127 ; cộng giá trị 127 vào giá trị trong A
MOV DPL, #50H ; nạp giá trị 50H vào DPL
MOV DPTR, #425000 ; invalid
```

Định địa chỉ thanh ghi

- Toán hạng là thanh ghi
- Thanh ghi (e.g., A, B, R0-R7 of current bank) chứa dữ liệu cần xử lý
- Thanh ghi được xác định bởi 3 bit trong *inst. opcode*
- Ví dụ:

ADD A, R7 ; cộng giá trị trong R7 với giá trị trong A

MOV R5, A ; nạp giá trị trong A vào R5

MOV DPTR, A; **invalid**

MOV R4, R5; **invalid**

Định địa chỉ trực tiếp

- Toán hạng nguồn là ngăn nhớ có địa chỉ cho trực tiếp trong lệnh
- Ngăn nhớ trong không gian địa chỉ 128B *Internal RAM* (00H-7FH)
- Ví dụ:

  ADD A, 7FH ; cộng giá trị ở vị trí 7FH vào A

  MOV R1, 20H ; nạp giá trị ở vị trí 20H trong RAM vào R1

  MOV A, 4 ; ? **= MOV A, R4**

  MOV R1, E8H ; **invalid**

- 80H-FFH không được sử dụng bởi programmer!

Định địa chỉ gián tiếp

- Toán hạng nguồn là ngăn nhớ có địa chỉ để trong thanh ghi
- **Chỉ sử dụng R0, R1 cho *internal RAM*** (R0 & R1, aka "*data pointer*")
- Ví dụ:

```
ADD A, @R0        ; cộng giá trị ở vị trí trong RAM chứa
                  ; bởi R0
MOV @R0, A        ; gán giá trị của A vào vị trí trong RAM
                  ; chứa bởi R0
MOV @R1, #35H ; gán giá trị 35H vào vị trí trong RAM
                  ; chứa bởi R1
MOVC A, @A + DPTR ; index+base register addressing
```

Example

- In the following program, assume:
  - that the word “USA” is burned into ROM locations starting at 200H, and
  - that the program is burned into ROM locations starting at 0.

How does the program work and where is “USA” stored after the program is run?

```
        ORG 0000H          ;
        MOV DPTR,#200H     ;
        CLR A              ;
        MOVC A,@A+DPTR     ;
        MOV R0,A           ;
        INC DPTR           ;
        CLR A              ;
        MOVC A,@A+DPTR     ;
        MOV R1,A           ;
        INC DPTR           ;
        CLR A              ;
        MOVC A,@A+DPTR     ;
        MOV R2,A           ;
Here:   SJMP HERE          ;

        ORG 200H
MYDATA:DB “USA”
        END                ;
```

Tập lệnh 8051

Lệnh chuyển dữ liệu (*data transfer*)	
Lệnh	**Giải thích**
MOV đích, nguồn	Đích = nguồn (Bộ nhớ trong)
MOVX đích, nguồn	Đích = nguồn (Thao tác bộ nhớ ngoài)
PUSH	Đẩy dữ liệu vào đỉnh ngăn xếp
POP	Lấy dữ liệu từ đỉnh ngăn xếp
XCH	Tráo đổi dữ liệu
XCHD	Tráo đổi dữ liệu (4 bit thấp)

Tập lệnh 8051

Mnemonic	Operation	Addressing Modes				Execution Time @ 12MHz (µs)
		Dir	Ind	Reg	Imm	
MOV A, <src>	A = <src>	X	X	X	X	1
MOV <dest>, A	<dest> = A	X	X	X		1
MOV <dest>, <src>	<dest> = <src>	X	X	X	X	2
MOV DPTR, # data 16	DPTR = 16-bit immediate constant				X	2
PUSH <src>	INC SP: MOV "@SP", <scr>	X				2
POP <dest>	MOV <dest>, "@SP": DEC SP	X				2
XCH A, <byte>	ACC and <byte> Exchange Data	X	X	X		1
XCHD A, @Ri	ACC and @ Ri exchange low nibbles		X			1

Example

MOV A,#55H

MOV A,#data

Bytes: 2

Cycles: 1

Encoding:

0	1	1	1	0	1	0	0	immediate data

Operation: MOV
(A) ← #data

MOV P1, P0

MOV direct,direct

Bytes: 3

Cycles: 2

Encoding:

1	0	0	0	0	1	0	1	dir. addr. (scr)	dir. addr. (dest)

Operation: MOV
(direct) ← (direct)

☞ Read - **Atmel 8051 Microcontrollers Hardware Manual**
- 8051 Instruction Set Manual – Keil (https://www.keil.com/support/man/docs/is51/)

MOV

Thanh ghi đích có thể là **A, B, R0-R7**

```
MOV A,#55H      ; reg.A ← 55h
MOV R6,#12      ; R6←12=0CH
MOV R0,A        ; A=55H, R0=55H
MOV R5,#0F9H    ; load F9H into R5
MOV A,17H       ; load the value of mem
                ; location 17h to A
MOV A,#’4’      ; A = ?
```

Tập lệnh 8051

Lệnh số học (*arithmetic*)	
Lệnh	**Giải thích**
ADD đích, nguồn	Đích = đích + nguồn
ADDC đích, nguồn	Đích = đích + nguồn + carry
SUBB đích, nguồn	Đích = đích – nguồn - carry
INC nguồn	Đích = đích + 1
DEC nguồn	Đích = đích - 1
MUL AB	A*B
DIV AB	A/B

Tập lệnh 8051

Mnemonic	Operation	Addressing Modes				Execution Time in X1 Mode @12 MHz (µs)
		Dir	Ind	Reg	Imm	
ADD A, <byt>e	A = A + <byte>	X	X	X	X	
ADDC A, <byte>	A = A + <byte> + C	X	X	X	X	1
SUBB A, <byte>	A = A – <byte> – C	X	X	X	X	1
INC A	A = A + 1	Accumulator only				1
INC <byte>	<byte> = <byte> + 1	X	X	X		1
INC DPTR	DPTR = DPTR + 1	Data Pointer only				2
DEC A	A = A – 1	Accumulator only				1
DEC <byte>	<byte> = <byte> – 1	X	X	X		1
MUL AB	B:A = B × A	ACC and B only				4
DIV AB	A = Int [A/B] B = Mod [A/B]	ACC and B only				4
DA A	Decimal Adjust	Accumulator only				1

Lệnh số học

- Toán hạng đích bắt buộc là thanh ghi A

```
MOV A,#25H      ;load 25H into A
MOV R2,#34H     ;load 34H into R2
ADD A,R2        ;add R2 to A=A+R2
```

➔ R2, A = ?

```
ADD R0,A        ;không hợp lệ
```

Tập lệnh 8051

Lệnh logic + thao tác bit (*logical+bit manipulation*)	
Lệnh	**Giải thích**
ANL	Lệnh “AND”
ORL	Lệnh “OR”
XRL	Lệnh “XOR”
CLR	Xóa bit
RL, RLC	Lệnh quay trái
RR, RRC	Lệnh quay phải

Tập lệnh 8051

Mnemonic	Operation	Addressing Modes				Execution Time @ 12MHz (µs)
		Dir	Ind	Reg	Imm	
ANL A, <byte>	A = A AND <byte>	X	X	X	X	1
ANL <byte>, A	<byte> = <byte> AND A	X				1
ANL <byte>, # data	<byte> = <byte> AND # data	X				2
ORL A, <byte>	A = A OR <byte>	X	X	X	X	1
ORL <byte>, A	<byte> = <byte> OR A	X				1
ORL <byte>, # data	<byte> = <byte> OR # data	X				2
XRL A, <byte>	A = A XOR <byte>	X	X	X	X	1
XRL <byte>, A	<byte> = <byte> XOR A	X				1
XRL <byte>, # data	<byte> = <byte> XOR # data	X				2

Ví dụ

- Lệnh ANL

  ANL đích, nguồn ; đích=đích AND nguồn

  – Mục đích: che, xóa bit

  – VD: - Xóa 4 bit thấp của thanh ghi A

  ANL A, #0F0h

- Lệnh ORL

  ORL đích, nguồn ; đích=đích OR nguồn

  – Mục đích: thiết lập bit

  – VD: - Thiết lập 4 bit cao của thanh ghi A

  ORL A, #0F0h

Các lệnh logic, lệnh quay

- Lệnh XRL

  XRL đích, nguồn ; đích=đích XOR nguồn

  – Mục đích: - Xóa thanh ghi (XOR với chính nó)
  - Đảo bit (XOR với 1)

  – VD: Xóa thanh ghi A

  XRL A,A

  Đảo các bit của thanh ghi A

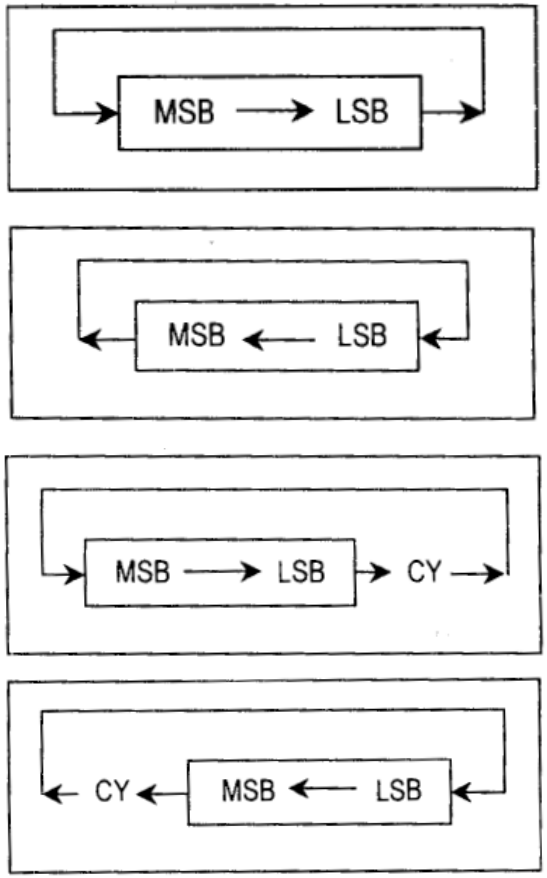
  XRL A,#0FFh

Tập lệnh 8051

CLR A	A = 00H	Accumulator only	1
CLP A	A = NOT A	Accumulator only	1
RL A	Rotate ACC Left 1 bit	Accumulator only	1
RLC A	Rotate Left through Carry	Accumulator only	1
RR A	Rotate ACC Right 1 bit	Accumulator only	1
RRC A	Rotate Right through Carry	Accumulator only	1
SWAP A	Swap Nibbles in A	Accumulator only	1

Các lệnh logic, lệnh quay

- Lệnh quay
  - Lệnh quay phải: RR A
  - Lệnh quay trái: RL A
  - Lệnh quay phải qua cờ nhớ:
    RRC A
  - Lệnh quay trái qua cờ nhớ:
    RLC A



Lệnh thao tác với bit

C = carry flag

Table 1-9. 8051 Boolean Instructions

Mnemonic	Operation	Execution Time @ 12MHz (µs)
ANL C,bit	C = C AND bit	2
ANL C,/bit	C = C AND (NOT bit)	2
ORL C,bit	C = C OR bit	2
ORL C,/bit	C = C OR (NOT bit)	2
MOV C,bit	C = bit	1
MOV bit,C	bit = C	2
CLR C	C = 0	1
CLR bit	bit = 0	1
SETB C	C = 1	1
SETB bit	bit = 1	1
CPL C	C = NOT C	1
CPL bit	bit = NOT bit	1
JC rel	Jump if C = 1	2
JNC rel	Jump if C = 0	2
JB bit,rel	Jump if bit = 1	2
JNB bit,rel	Jump if bit = 0	2
JBC bit,rel	Jump if bit = 1 ; CLR bit	2

Tập lệnh 8051

Lệnh rẽ nhánh (*branching and control transfer*)	
Lệnh	**Giải thích**
ACALL	Gọi chương trình con, địa chỉ 11 bit
LCALL	Gọi chương trình con, địa chỉ 16 bit
RET	Trở về từ chương trình con
JMP	Lệnh nhảy không điều kiện
JZ, JNZ, JB, JNB…	Lệnh nhảy có điều kiện (kiểm tra bit)

Lệnh nhảy, rẽ nhánh

Mnemonic	Operation	Execution Time @ 12MHz (µs)
JMP addr	Jump to addr	2
JMP @A + DPTR	Jump to A + DPTR	2
CALL addr	Call subroutine at addr	2
RET	Return from subroutine	2
RETI	Return from interrupt	2
NOP	No operation	1

Mnemonic	Operation	Addressing Modes				Execution Time @ 12MHz (µs)
		DIR	IND	REG	IMM	
JZ rel	Jump if A = 0	Accumulator only				2
JNZ rel	Jump if A ≠ 0	Accumulator only				2
DJNZ <byte>,rel	Decrement and jump if not zero	X		X		2
CJNZ A,<byte>,rel	Jump if A = <byte>	X			X	2
CJNE <byte>,#data,rel	Jump if <byte> = #data		X	X		2

Các lệnh rẽ nhánh

- **Lệnh nhảy có điều kiện**: JZ, JNZ, DJNZ, JC, JNC, JB, JNB
- **Lệnh nhảy không điều kiện**: SJMP (nhảy ngắn), LJMP (nhảy dài)
- Ví dụ:

```
        MOV     A,R5        ;A=R5
        JNZ     NEXT        ;Nhảy tới NEXT nếu A khác 0
        MOV     R5,#55h
NEXT:
        …
```

SJMP rel

Function: Short Jump

Description: Program control branches unconditionally to the address indicated. The branch destination is computed by adding the signed displacement in the second instruction byte to the PC, after incrementing the PC twice. Therefore, the range of destinations allowed is from 128 bytes preceding this instruction 127 bytes following it.

Example: The label RELADR is assigned to an instruction at program memory location 0123H. The following instruction,

```
SJMP        RELADR
```

assembles into location 0100H. After the instruction is executed, the PC contains the value 0123H.

Note: Under the above conditions the instruction following SJMP is at 102H. Therefore, the displacement byte of the instruction is the relative offset (0123H-0102H) = 21H. Put another way, an SJMP with a displacement of 0FEH is a one-instruction infinite loop.

Bytes: 2

Cycles: 2

Encoding:

1	0	0	0	0	0	0	0	rel. address

Operation: SJMP

(PC) ← (PC) + 2

(PC) ← (PC) + rel

LJMP addr16

Function: Long Jump

Description: LJMP causes an unconditional branch to the indicated address, by loading the high-order and low-order bytes of the PC (respectively) with the second and third instruction bytes. The destination may therefore be anywhere in the full 64K program memory address space. No flags are affected.

Example: The label JMPADR is assigned to the instruction at program memory location 1234H. The instruction,

LJMP JMPADR

at location 0123H will load the program counter with 1234H.

Bytes: 3

Cycles: 2

Encoding:

0	0	0	0	0	0	1	0	addr15-addr8	addr7-addr0

Operation: LJMP

(PC) ← $addr_{15-0}$

Lệnh lặp

- Lệnh DJNZ
  - Cú pháp

    DJNZ thanh_ghi, nhãn

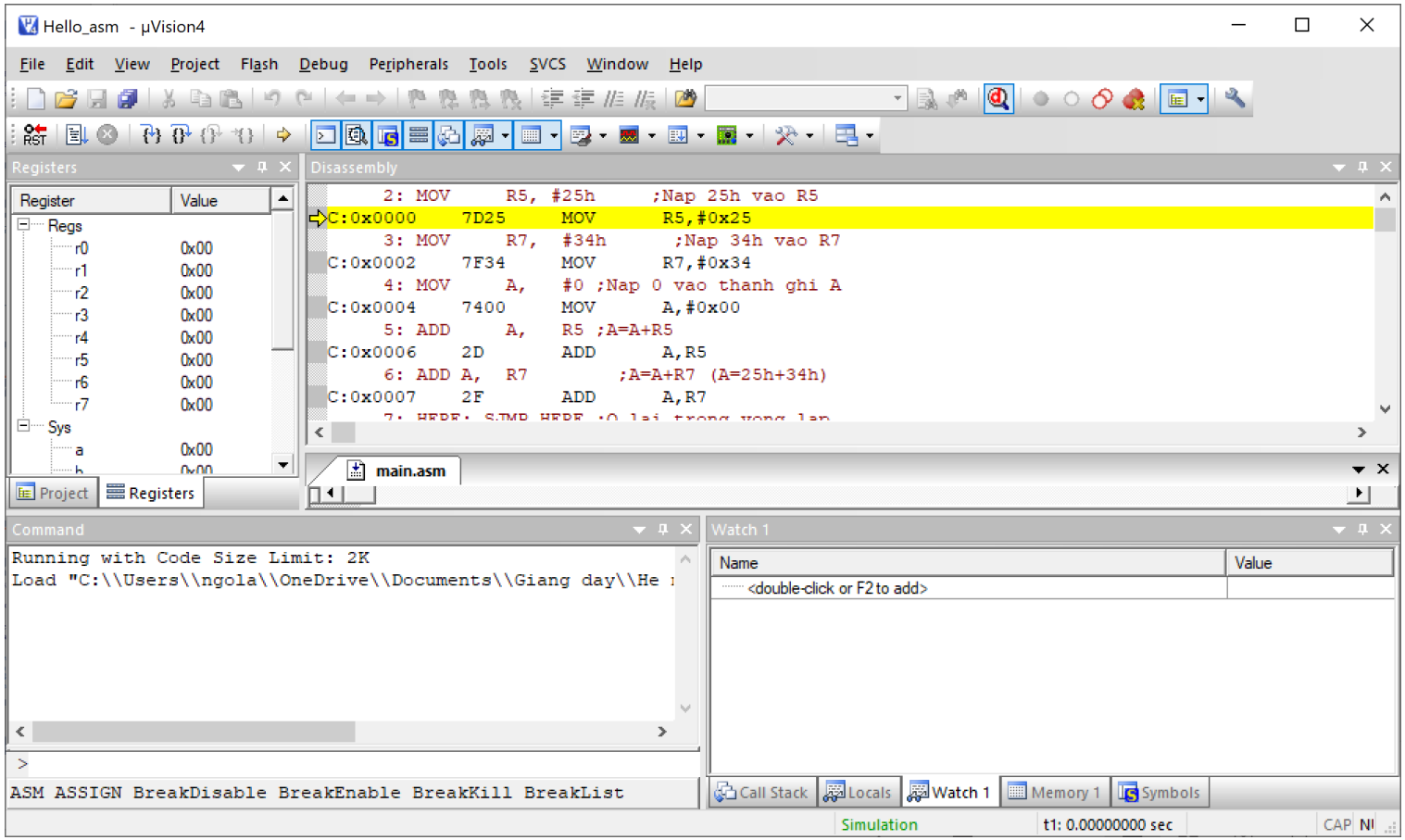
    + Sau mỗi lần nhảy, giá trị thanh ghi bị giảm đi 1
    + Nếu giá trị thanh ghi vẫn khác 0 thì nhảy tới nhãn
- Ví dụ: Xóa thanh ghi A, cộng 3 vào thanh ghi A 10 lần

```
        MOV   A,#0
        MOV   R2,#10
BACK:   ADD   A,#3
        DJNZ  R2,BACK      ;Lặp 10 lần
        MOV   R5,A         ;Cất A vào R5
```

Ví dụ chương trình đầy đủ

```
ORG   000          ;Dia chi bat dau cua chuong trinh
MOV   R5, #25h     ;Nap 25h vao R5
MOV   R7, #34h     ;Nap 34h vao R7
MOV   A, #0        ;Nap 0 vao thanh ghi A
ADD   A, R5        ;A=A+R5
ADD   A, R7        ;A=A+R7 (A=25h+34h)
HERE: SJMP HERE    ;loop forever. WHY?
END
```

Lập trình và mô phỏng với Keil C51



Lập trình cổng vào ra

- 8051 có 4 cổng vào ra GPIO (mỗi cổng 8 bit): P0, P1, P2, P3.
- Sau khi reset, các cổng ở chế độ mặc định là cổng ra (output).
- Để cổng/chân làm việc ở chế độ cổng/chân vào (input) phải ghi bit 1 ra cổng/chân tương ứng.
  - Ví dụ:

```
MOV  P1,#0FF     ; Cổng 1 thành cổng vào
SETB  P1.0       ; Chân P1.0 làm chân vào
MOV  P1,#03      ; Chân P1.0 và P1.1 làm chân vào
                 ; các chân còn lại làm chân ra
```

Xuất dữ liệu ra cổng/chân

- Xuất dữ liệu ra cổng ra:

  MOV tên_cổng, giá trị

  – Ví dụ: MOV P1, #55h

- Xuất dữ liệu ra từng chân:
  – Đưa chân lên mức cao:

  SETB bit

  Ví dụ: SETB P1.0

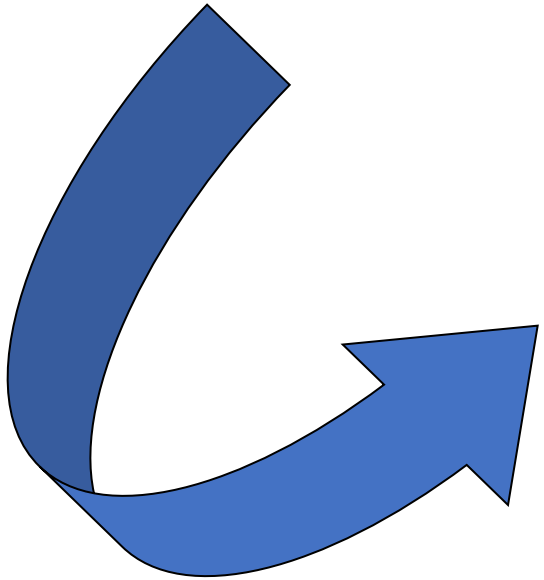
  – Đưa chân xuống mức thấp:

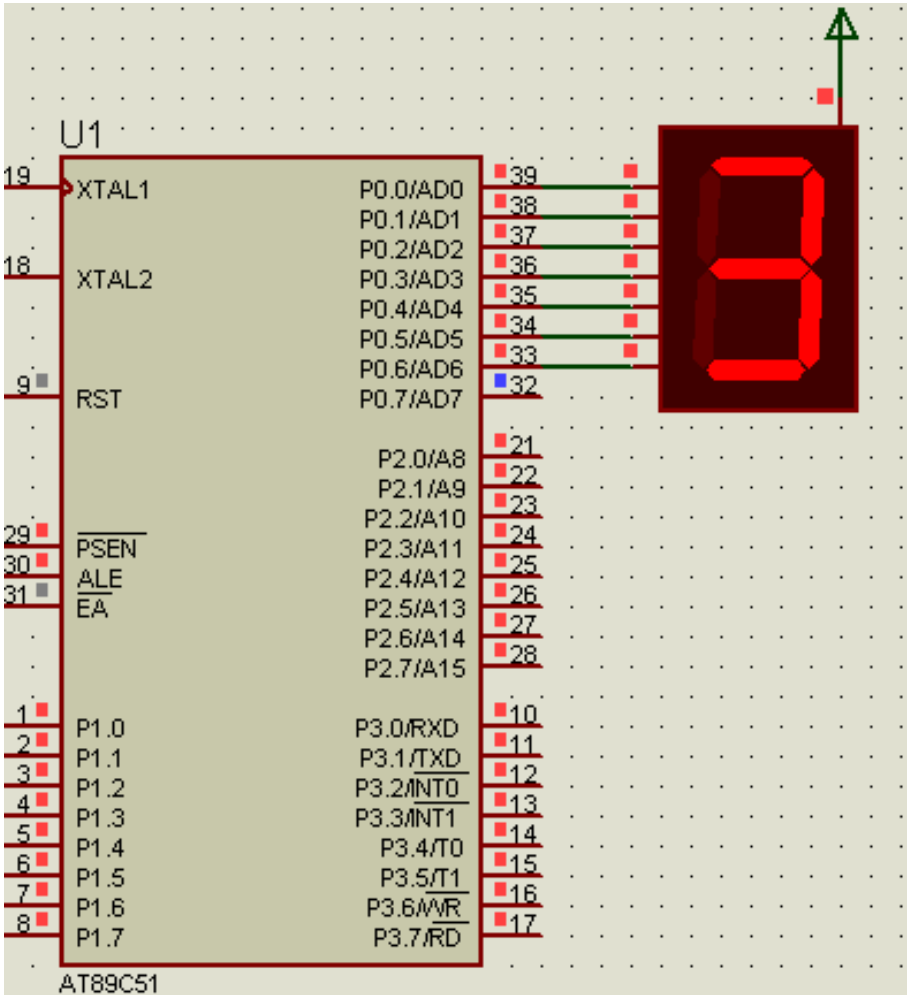
  CLR bit

  Ví dụ: CLR P1.0

Ví dụ xuất dữ liệu ra cổng ra

MOV P0,#30h





Đọc dữ liệu từ cổng vào

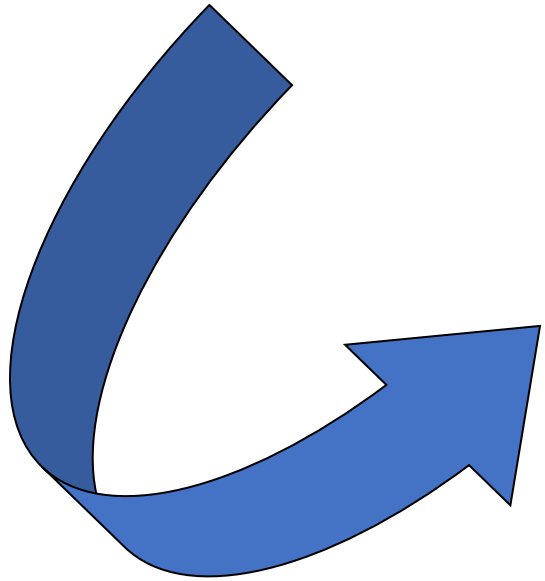
- **Bước 1**: Thiết lập cổng làm việc ở chế độ cổng vào
- **Bước 2**: Đọc dữ liệu trên cổng

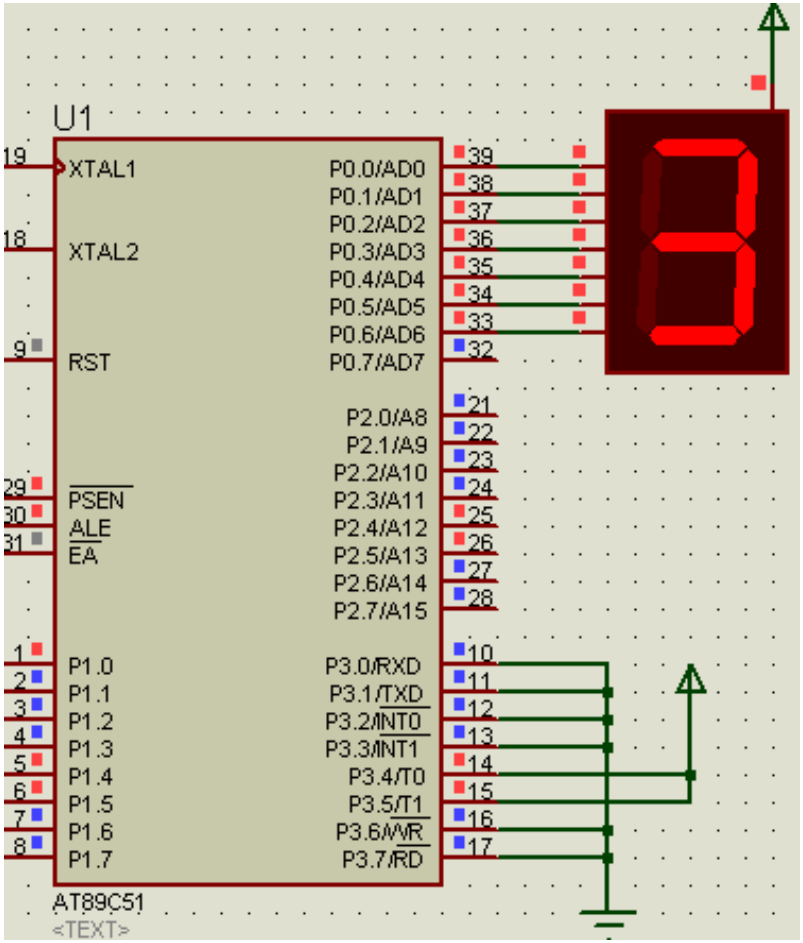
Ví dụ:

```
MOV     P1, #0FFh
MOV     A, P1       ; Đọc giá trị tại
                    ; cổng P1, lưu vào A
MOV     P2, A       ; Xuất ra cổng P2
```

Ví dụ đọc dữ liệu từ cổng vào

```
MOV   P3,#0FFh
MOV   A,P3
MOV   P0,A
MOV   P2,A
```





Chương trình con

- Là đoạn chương trình có thể được gọi lặp lại nhiều lần từ chương trình chính.
- Khai báo

```
routine_name:
        ;body
        ;body
        RET
```

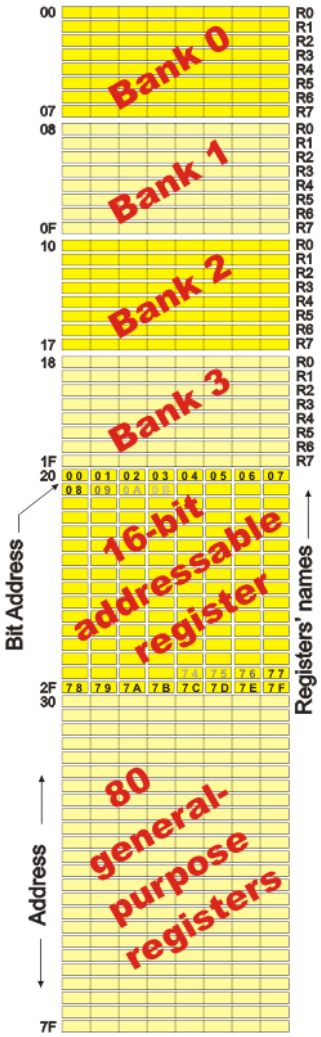
- Sử dụng

```
ACALL routine_name;short call (giới hạn 2k)
LCALL routine_name;long call
```

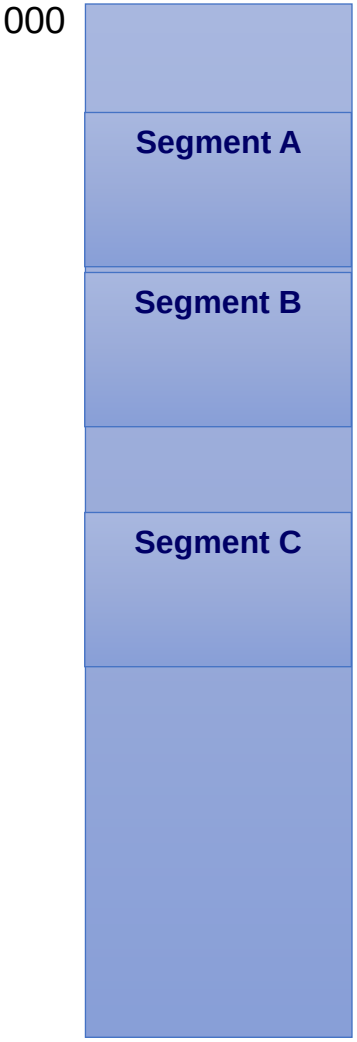
Ví dụ nhấp nháy LED

```
        ORG 000                     ;Địa chỉ bắt đầu của chương trình
AGAIN:
        SETB P1.0                   ;Nhấp nháy LED ở chân P1.0
        ACALL DELAY
        CLR P1.0
        ACALL DELAY
        SJMP AGAIN
DELAY:                  ;Tạo trễ
        MOV R1,#255
LOOP:
        DJNZ R1,LOOP
        RET
        END
```

Tổ chức chương trình



Data memory space



Reset vector, fixed

Relocatable

Relocatable

Relocatable

Program memory space

Ví dụ

```
;Program Structure demo
;Code segments
MAIN_PROG  SEGMENT CODE
SUBROUTINE SEGMENT CODE
;Simple 'data segment'
var1 EQU 0x30

;Reset address
CSEG AT 000
          JMP MAIN
          RSEG MAIN_PROG
MAIN: MOV R5, #25h
          MOV       R7,       #34h
          MOV       A,        #0
          ADD       A,        R5
          ADD A,    R7
          MOV var1, A
          CALL LAP
          RSEG SUBROUTINE
LAP: SJMP LAP
          RET
          END
```