

Cơ bản về lập trình hợp ngữ (tiếp)

TS Nguyễn Hồng Quang



Electrical Engineering

1

Giới thiệu lệnh cơ bản

- 5.5 Truy cập bộ nhớ
- 5.6 Vùng nhớ đặc biệt
- 5.7 Lệnh nhảy và vòng lặp



Electrical Engineering

2

Lệnh thường gặp 8051

- Mã máy trong 8051 thì có giá trị trong khoảng 00h-FFh ngoại trừ A5h
- Mã lệnh được phân chia theo nhóm như sau
 - Lệnh chuyển dữ liệu (e.g. MOV, MOVX, PUSH, POP, XCH)
 - Lệnh toán học (e.g. INC, ADDC, DEC, SUBB, MUL, DIV)
 - Lệnh logic (e.g. CLR, SETB, ANL, RRC, ORL, XRL)
 - Lệnh điều khiển (e.g. AJMP, LJMP, JMP, ACALL, LCALL, RET, DJNZ, JNB)
- Phương pháp truy cập địa chỉ: trực tiếp, gián tiếp, thanh ghi, trung gian



5.5 Chế độ truy cập bộ nhớ

- **5.5.1 Immediate Addressing** MOV A,#20h
- **5.5.2 Direct Addressing** MOV A,30h
- **5.5.3 Indirect Addressing** MOV A,@R0
- **5.5.4 External Direct** MOVX A,@DPTR
- **5.5.5 Code Indirect** MOVC A,@A+DPTR



5.5.1 Địa chỉ tức thời

- Gán giá trị
- MOV A,#20h
- Cần có ký hiệu ‘ #’, trước giá trị
- Lệnh thực hiện nhanh, nhưng không mềm dẻo



5.5.1 Ví dụ

MOV	A, # 25H	; Nạp giá trị 25H vào thanh ghi A
MOV	R4, #62	; Nạp giá trị 62 thập phân vào R4
MOV	B, #40H	; Nạp giá trị 40 H vào thanh ghi B
MOV	DPTR, #4521H	; Nạp 4512H vào con trỏ dữ liệu DPTR

MOV	DPTR, #2550H
MOV	A, #50H
MOV	DPH, #25H



5.5.2 Direct Addressing

- Địa chỉ trực tiếp
- **MOV A,30h**
- Nạp vào A, giá trị trong ô địa chỉ 30 H
- Chỉ dùng trong 128 byte RAM trong



5.5.2 Ví dụ

MOV	R0, 40H	; Lưu nội dung của ngăn nhớ 40H của RAM vào R0
MOV	56H, A	; Lưu nội dung thanh ghi A vào ngăn nhớ 56H của RAM
MOV	R4, 7FH	; Chuyển nội dung ngăn nhớ 7FH của RAM vào R4
MOV	A, 4	; Hai lệnh này giống nhau đều sao nội dung thanh ghi R4 vào A
MOV	A, R4	
MOV	A, 7	; Hai lệnh này đều như nhau là sao nội dung R7 vào thanh ghi A
MOV	A, R7	



5.5.3 Indirect Addressing

- Địa chỉ gián tiếp
- **MOV A,@R0**
- Phân tích giá trị R0, xem thanh ghi R0 trỏ địa chỉ nào, copy giá trị trong ô địa chỉ đó vào A
- Ví dụ R0 = 40h, trong ô 40 h chứa 1A, do vậy A sẽ chứa 1 A
- Dùng trong bộ nhớ RAM trong, không truy cập được các thanh ghi đặc biệt



5.5.3 Ví dụ

	MOV A, #55H	; Nạp A giá trị 55H
	MOV 40H, A	; Sao chép A vào ngăn nhớ RAM 40H
	MOV 41H, A	; Sao chép A vào ngăn nhớ RAM 41H
	MOV 42H, A	; Sao chép A vào ngăn nhớ RAM 42H
	MOV 43H, A	; Sao chép A vào ngăn nhớ RAM 43H
	MOV 44H, A	; Sao chép A vào ngăn nhớ RAM 44H
b)	MOV A, # 55H	; Nạp vào A giá trị 55H
	MOV R0, #40H	; Nạp con trỏ R0 = 40 H
	MOV @R0, A	; Sao chép A vào vị trí ngăn nhớ RAM do R0 chỉ đến
	INC R0	; Tăng con trỏ. Bây giờ R0 = 41H
	MOV @R0, A	; Sao chép A vào vị trí ngăn nhớ RAM do R0 chỉ
	INC R0	; Tăng con trỏ. Bây giờ R0 = 42H
	MOV @R0, A	; Sao chép A vào vị trí ngăn nhớ RAM do R0 chỉ
	INC R0	; Tăng con trỏ. Bây giờ R0 = 43H
	MOV @R0, A	; Sao chép A vào vị trí ngăn nhớ RAM do R0 chỉ
	MOV @R0, A	; Tăng con trỏ. Bây giờ R0 = 44H
c)	MOV A, # 55H	; Nạp vào A giá trị 55H
	MOV R0, #40H	; Nạp con trỏ địa chỉ ngăn nhớ RAM R0 = 40H
	MOV R2, #05	; Nạp bộ đếm R2 = 5
AGAIN:	MOV @R0, A	; Sao chép A vào vị trí ngăn nhớ RAM do R0 chỉ đến
	INC R0	; Tăng con trỏ R0
	DJNZ R2, AGAIN	; Lặp lại cho đến khi bộ đếm = 0.



5.5.3 Ưu điểm của chế độ truy cập gián tiếp qua thanh ghi

- Tăng tính mềm dẻo trong truy xuất bộ nhớ
- Ví dụ: hãy viết chương trình để xóa 16 vị trí ngăn nhớ RAM bắt đầu tại địa chỉ 60H.

```
CLR    A           ; Xóa A=0
MOV    R1, #60H    ; Nạp con trỏ. R1= 60H
MOV    R7, #16H    ; Nạp bộ đếm, R7 = 16 (10 H dạng hex)

AGAIN: MOV    @R1, A ; Xóa vị trí ngăn nhớ RAM do R1 chỉ đến
INC    R1          ; Tăng R1
DJNZ   R7, AGAIN   ; Lặp lại cho đến khi bộ đếm = 0
```



5.5.3 Ví dụ tiếp

Hãy viết chương trình để sao chép một khối 10 byte dữ liệu từ vị trí ngăn nhớ RAM bắt đầu từ 35H vào các vị trí ngăn nhớ RAM bắt đầu từ 60H

```
MOV    R0, #35H    ; Con trỏ nguồn
MOV    R1, #60H    ; Con trỏ đích
MOV    R3, #10     ; Bộ đếm
BACK:  MOV    A, @R0 ; Lấy 1byte từ nguồn
        MOV    @R1, A ; Sao chép nó đến đích
        INC    R0     ; Tăng con trỏ nguồn
        INC    R1     ; Tăng con trỏ đích
        DJNZ   R3, BACK ; Lặp lại cho đến khi sao chép hết 10 byte
```



5.5.4 External Direct

- Truy cập bộ nhớ ngoài
- MOVX A,@DPTR
 - Nạp vào A giá trị ô nhớ có địa chỉ bởi DPTR
- MOVX @DPTR,A
 - Nạp vào ô nhớ có địa chỉ bởi DPTR giá trị A



5.5.4 Ví dụ về Index addressing mode

```
ORG 0
mov DPTR, #LUT ; 3000H is the LUT address
mov A, #0FFH
mov P1, A ; program the port P1 to input data
mov A,#00h
back: inc A ; read x
      movc A, @A+DPTR ; get x2 from LUT
      mov P2, A ; output x2 to P2
      sjmp back ; for (1) loop

ORG 3000H
LUT: DB 0, 1, 4, 9, 16, 25, 36, 49, 64, 81
```



5.5.5 Code Indirect

- `MOVC A,@A+DPTR`
- Nạp vào A, giá trị trữ bởi `DPTR +` với A, trong bảng tìm kiếm



5.5.5 Ví dụ

```
; (a) Phương pháp này sử dụng một bộ đếm
      ORG      000
      MOV      DPTR, # MYDATA          ; Nạp con trỏ ROM
      MOV      R0, #40H                 ; Nạp con trỏ RAM
      MOV      R2, #7                   ; Nạp bộ đếm
BACK:  CLR     A                         ; Xóa thanh ghi A
      MOVC     A, @A + DPTR             ; Chuyển dữ liệu từ không gian mã
      MOV      R0, A                    ; Cất nó vào ngăn nhớ RAM
      INC      DPTR                     ; Tăng con trỏ ROM
      INC      R0                       ; Tăng con trỏ RAM
      DJNZ     R2, BACK                 ; Lặp lại cho đến khi bộ đếm = 0
HERE:  SJMP     HERE
;----- không gian mã của ROM trên chip dùng để cất dữ liệu ORG 250H

MYDATA: DB      "AMER1CA"
      END
```



5.6 (Vùng nhớ đặt biệt) SFR

- Bộ nhớ RAM trong khoảng 80H – FFH (128byte)
- Chỉ có 21 thanh ghi hợp lệ
- Thực hiện các chức năng phục vụ riêng cho 8051
- Làm việc như làm việc với bộ nhớ RAM bình thường



5.6.4 Các lệnh sử dụng tới PSW

Instruction	CY	OV	AC
ADD	X	X	X
ADDC	X	X	X
SUBB	X	X	X
MUL	0	X	
DIV	0	X	
DA	X		
RPC	X		
PLC	X		
SETB C	1		
CLR C	0		
CPL C	X		
ANL C, bit	X		
ANL C, /bit	X		
ORL C, bit	X		
ORL C, /bit	X		
MOV C, bit	X		
CJNE	X		



5.6.4 Ví dụ: Lệnh ADD và PSW

38	0011 1000	
+ 2F	0010 1111	
-----	-----	
67	0110 0111	
-----	-----	
-		
CY = 0	MOV A, #38H	
AC = 1	ADD A, #2FH	; Sau khi cộng A = 67H, CY = 0
P = 1		



Electrical Engineering

19

5.6.4 Ví dụ PSW tiếp

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

Cờ CY = 1 vì có nhớ qua bit D7
 Cờ AC = 1 vì có nhớ từ D3 sang D4
 Cờ P = 0 vì thanh ghi A không có bit 1 nào (chẵn)

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

Cờ CY = 1 vì có nhớ từ bit D7
 Cờ AC = 0 vì không có nhớ từ D3 sang D4
 Cờ P = 0 vì số bit 1 trong A là 4 (chẵn)






Electrical Engineering

20

5.6.8 Bộ nhớ SFR của 89c52

80	P0	SP	DPL	DPH				PCON	87
90	ICON	IMOD	TL0	TL1	TH0	TH1			9F
98	P1								9F
98	SCON	SBUF							9F
A0	P2								A7
A0	IE								AF
B0	P3								B7
B0	IP								B7
C0									C7
C0	IECON		RCAP2L	RCAP2H	IL2	IH2			CF
D0	PSW								D7
D0									D7
E0	ACC								E7
E0									E7
F0	R								F7
F0									F7

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



5.7 Lệnh nhảy và vòng lặp

Lệnh: DJNZ reg, Label

Cho phép vòng lặp nhiều lần, với thanh ghi R0-R7

```

;This program adds value 3 to the ACC ten times
MOV A, #0 ;A=0, clear ACC
MOV R2, #10 ;load counter R2=10
AGAIN: ADD A, #03 ;add 03 to ACC
DJNZ R2, AGAIN ;repeat until R2=0, 10 times
MOV R5, A ;save A in R5
    
```



5.7.1 Lệnh nhảy có điều kiện

- JZ label ; nhảy nếu A=0
- JNZ label; nhảy nếu A!=0

```

MOV  A,R0      ;A=R0
JZ   OVER      ;jump if A = 0
MOV  A,R1      ;A=R1
JZ   OVER      ;jump if A = 0
... \
OVER:
MOV  A,R5      ;copy R5 to A
JNZ  NEXT      ;jump if A is not zero
MOV  R5,#55H
NEXT:  ...

```



5.7.1 Lệnh nhảy có điều kiện

- JNC label ;nhảy nếu CY=0
- Ví dụ tính tổng 79H, F5H, E2H và lưu kết quả vào R0 (low byte) và R5 (high byte).

```

MOV  A,#0      ;A=0
MOV  R5,A      ;clear R5
ADD  A,#79H    ;A=0+79H=79H
; JNC N_1      ;if CY=0, add next number
; INC R5       ;if CY=1, increment R5
N_1: ADD  A,#0F5H ;A=79+F5=6E and CY=1
      JNC  N_2   ;jump if CY=0
      INC  R5    ;if CY=1, increment R5 (R5=1)
N_2: ADD  A,#0E2H ;A=6E+E2=50 and CY=1
      JNC  OVER  ;jump if CY=0
      INC  R5    ;if CY=1, increment 5
OVER: MOV  R0,A  ;now R0=50H, and R5=02

```



5.7.1 Các lệnh nhảy trong 8051

Instructions	Actions
JZ	Jump if A = 0
JNZ	Jump if A \neq 0
DJNZ	Decrement and Jump if A \neq 0
CJNE A,byte	Jump if A \neq byte
CJNE reg,#data	Jump if byte \neq #data
JC	Jump if CY = 1
JNC	Jump if CY = 0
JB	Jump if bit = 1
JNB	Jump if bit = 0
JBC	Jump if bit = 1 and clear bit

- Các lệnh nhảy có điều kiện đều trong ± 128 bytes giới hạn



5.7.2 Các lệnh nhảy khác

- SJMP, ± 128 bytes giới hạn
- AJMP, 2 kbytes block giới hạn
- LJMP – 3 bytes
- Việc tính địa chỉ lệnh nhảy được tính bằng hiệu địa chỉ thực trừ đi địa chỉ con trỏ PC ngay sau lệnh



5.7.2 Ví dụ về lệnh nhảy

Line	PC	Opcode	Mnemonic Operand
01	0000		ORG 0000
02	0000	7800	MOV R0, #0
03	0002	7455	MOV A, #55H
04	0004	6003	JZ NEXT
05	0006	08	INC R0
06	0007	04	AGAIN: INC A
07	0008	04	INC A
08	0009	2477	NEXT: ADD A, #77H
09	000B	5005	JNC OVER
10	000D	E4	CLR A
11	000E	F8	MOV R0, A
12	000F	F9	MOV R1, A
13	0010	FA	MOV R2, A
14	0011	FB	MOV R3, A
15	0012	2B	OVER: ADD A, R3
16	0013	50F2	JNC AGAIN
17	0015	80FE	HERE: SJMP HERE
18	0017		END



Electrical Engineering

27

5.7.2 Lệnh gọi chương trình

- LCALL, ACALL
 - Gọi chương trình con theo tên
 - Lưu trữ PC vào stack
- RET
 - Kết thúc chương trình con
 - Lấy giá trị PC từ stack



Electrical Engineering

28

5.7.2 Ví dụ

```
ORG 0
BACK: MOV A,#55H ;load A with 55H
      MOV P1,A   ;send 55H to port 1
      LCALL DELAY ;time delay
      MOV A,#0AAH ;load A with AA (in hex)
      MOV P1,A   ;send AAH to port 1
      LCALL DELAY
      SJMP BACK  ;keep doing this indefinitely
      ...
      END        ;end of asm file
```

A rewritten program which is more efficiently

```
ORG 0
MOV A,#55H ;load A with 55H
BACK: MOV P1,A ;send 55H to port 1
      ACALL DELAY ;time delay
      CPL A ;complement reg A
      SJMP BACK ;keep doing this indefinitely
      ...
      END ;end of asm file
```

