

微算機原理及應用

單元三：8051的組合語言程式設計

授課老師：林淵翔 老師

大綱

- 組合語言的格式(Assembly format)
- 各種定址法(Addressing mode)
- 指令集(Instruction set)
- 組譯過程與機器碼(Assembly and Machine code)
- 資料型態與虛指令(Data type and Directives)

大綱

- 暫存器Bank與旗標、堆疊(Register banks, Flags and Stack)
- 機器週期與時間延遲計算(Machine cycles and Time delay)
- 範例介紹
- 參考文獻 (References)

單元三
8051的組合語言程式設計
PART A

3.1 組合語言格式

- Label Mnemonic Operand Comment
- 標籤 助憶碼(運算碼) 運算元 註解
- EX:

— Loop:	MOV	P1,A	;delay some time
—	CALL	Delay	
—	RL	A	
—	JMP	Loop	
— Delay: ..			

3.2 MOV 指令(Instruction)

- MOV destination, source ;copy source to dest.
 - MOV A, #55H ; load 55H to A
 - MOV R0, A ; copy content of A to R0

3.3 定址模式(Addressing mode)

- 直接定址法(Direct addressing)
 - MOV A, 10H ;A=55H
- 間接定址法(Indirect addressing)
 - MOV A, @R0 ;If R0=00H then A=12H
- 暫存器定址法(Register instructions)
 - MOV A, R0 ;If R0=12H then A=12H
- 暫存器特定定址法(Register-specific instructions)
 - INC A ;If A=00H then A=01H
- 立即定址法(Immediate constants)
 - MOV A, #100 ;A=64H
- 索引定址法(Indexed addressing)
 - MOVC A, @A+DPTR ;存取程式記憶體資料

On-chip Memory

FFH	88H
↑	
10H	55H
↑	
01H	33H
00H	12H
Address	Data

單元三
8051的組合語言程式設計
PART B

3.4 指令集(Instruction set)

- 算術運算指令(Arithmetic operations)
- 邏輯運算指令(Logical operations)
- 資料搬移指令(Data transfer)
- 布林運算指令(Boolean variable manipulation)
- 程式跳躍指令(Program branching)

3.4.1 算術運算指令

Mnemonic		Description	Byte	Oscillator Period
ARITHMETIC OPERATIONS				
ADD	A,R _n	Add register to Accumulator	1	12
ADD	A,direct	Add direct byte to Accumulator	2	12
ADD	A,@R _i	Add indirect RAM to Accumulator	1	12
ADD	A,#data	Add immediate data to Accumulator	2	12
ADDC	A,R _n	Add register to Accumulator with Carry	1	12
ADDC	A,direct	Add direct byte to Accumulator with Carry	2	12
ADDC	A,@R _i	Add indirect RAM to Accumulator with Carry	1	12
ADDC	A,#data	Add immediate data to Acc with Carry	2	12
SUBB	A,R _n	Subtract Register from Acc with borrow	1	12
SUBB	A,direct	Subtract direct byte from Acc with borrow	2	12
SUBB	A,@R _i	Subtract indirect RAM from ACC with borrow	1	12
SUBB	A,#data	Subtract immediate data from Acc with borrow	2	12

參考資料來源：
 ATMEL 8051
 Microcontrollers
 Hardware Manual

3.4.1 算術運算指令(continued)

INC	A	Increment Accumulator	1	12
INC	R _n	Increment register	1	12
INC	direct	Increment direct byte	2	12
INC	@R _i	Increment direct RAM	1	12
DEC	A	Decrement Accumulator	1	12
DEC	R _n	Decrement Register	1	12
DEC	direct	Decrement direct byte	2	12
DEC	@R _i	Decrement indirect RAM	1	12
INC	DPTR	Increment Data Pointer	1	24
MUL	AB	Multiply A & B	1	48
DIV	AB	Divide A by B	1	48
DA	A	Decimal Adjust Accumulator	1	12

3.4.2 邏輯運算指令

Mnemonic		Description	Byte	Oscillator Period
LOGICAL OPERATIONS				
ANL	A,R _n	AND Register to Accumulator	1	12
ANL	A,direct	AND direct byte to Accumulator	2	12
ANL	A,@R _i	AND indirect RAM to Accumulator	1	12
ANL	A,#data	AND immediate data to Accumulator	2	12
ANL	direct,A	AND Accumulator to direct byte	2	12
ANL	direct,#data	AND immediate data to direct byte	3	24
ORL	A,R _n	OR register to Accumulator	1	12
ORL	A,direct	OR direct byte to Accumulator	2	12
ORL	A,@R _i	OR indirect RAM to Accumulator	1	12
ORL	A,#data	OR immediate data to Accumulator	2	12
ORL	direct,A	OR Accumulator to direct byte	2	12
ORL	direct,#data	OR immediate data to direct byte	3	24

3.4.2 邏輯運算指令(continued)

XRL	A,R _n	Exclusive-OR register to Accumulator	1	12
XRL	A,direct	Exclusive-OR direct byte to Accumulator	2	12
XRL	A,@R _i	Exclusive-OR indirect RAM to Accumulator	1	12
XRL	A,#data	Exclusive-OR immediate data to Accumulator	2	12
XRL	direct,A	Exclusive-OR Accumulator to direct byte	2	12
XRL	direct,#data	Exclusive-OR immediate data to direct byte	3	24
CLR	A	Clear Accumulator	1	12
CPL	A	Complement Accumulator	1	12
RL	A	Rotate Accumulator Left	1	12
RLC	A	Rotate Accumulator Left through the Carry	1	12
RR	A	Rotate Accumulator Right	1	12
RRC	A	Rotate Accumulator Right through the Carry	1	12
SWAP	A	Swap nibbles within the Accumulator	1	12

3.4.3 資料搬移指令

Mnemonic		Description	Byte	Oscillator Period
DATA TRANSFER				
MOV	A,R _n	Move register to Accumulator	1	12
MOV	A,direct	Move direct byte to Accumulator	2	12
MOV	A,@R _i	Move indirect RAM to Accumulator	1	12
MOV	A,#data	Move immediate data to Accumulator	2	12
MOV	R _n ,A	Move Accumulator to register	1	12
MOV	R _n ,direct	Move direct byte to register	2	24
MOV	R _n ,#data	Move immediate data to register	2	12
MOV	direct,A	Move Accumulator to direct byte	2	12
MOV	direct,R _n	Move register to direct byte	2	24
MOV	direct,direct	Move direct byte to direct	3	24
MOV	direct,@R _i	Move indirect RAM to direct byte	2	24

3.4.3資料搬移指令(continued)

Mnemonic		Description	Byte	Oscillator Period
MOV	direct,#data	Move immediate data to direct byte	3	24
MOV	@R _i ,A	Move Accumulator to indirect RAM	1	12
MOV	@R _i ,direct	Move direct byte to indirect RAM	2	24
MOV	@R _i ,#data	Move immediate data to indirect RAM	2	12
MOV	DPTR,#data16	Load Data Pointer with a 16-bit constant	3	24
MOVC	A,@A+DPTR	Move Code byte relative to DPTR to Acc	1	24
MOVC	A,@A+PC	Move Code byte relative to PC to Acc	1	24
MOVX	A,@R _i	Move External RAM (8-bit addr) to Acc	1	24
MOVX	A,@DPTR	Move External RAM (16-bit addr) to Acc	1	24
MOVX	@R _i ,A	Move Acc to External RAM (8-bit addr)	1	24
MOVX	@DPTR,A	Move Acc to External RAM (16-bit addr)	1	24
PUSH	direct	Push direct byte onto stack	2	24
POP	direct	Pop direct byte from stack	2	24
XCH	A,R _n	Exchange register with Accumulator	1	12
XCH	A,direct	Exchange direct byte with Accumulator	2	12
XCH	A,@R _i	Exchange indirect RAM with Accumulator	1	12
XCHD	A,@R _i	Exchange low-order Digit indirect RAM with Acc	1	12

3.4.4 布林運算指令

Mnemonic		Description	Byte	Oscillator Period
BOOLEAN VARIABLE MANIPULATION				
CLR	C	Clear Carry	1	12
CLR	bit	Clear direct bit	2	12
SETB	C	Set Carry	1	12
SETB	bit	Set direct bit	2	12
CPL	C	Complement Carry	1	12
CPL	bit	Complement direct bit	2	12
ANL	C,bit	AND direct bit to CARRY	2	24
ANL	C,/bit	AND complement of direct bit to Carry	2	24
ORL	C,bit	OR direct bit to Carry	2	24
ORL	C,/bit	OR complement of direct bit to Carry	2	24
MOV	C,bit	Move direct bit to Carry	2	12
MOV	bit,C	Move Carry to direct bit	2	24
JC	rel	Jump if Carry is set	2	24
JNC	rel	Jump if Carry not set	2	24
JB	bit,rel	Jump if direct Bit is set	3	24
JNB	bit,rel	Jump if direct Bit is Not set	3	24
JBC	bit,rel	Jump if direct Bit is set & clear bit	3	24

3.4.5 程式跳躍指令

Mnemonic		Description	Byte	Oscillator Period
PROGRAM BRANCHING				
ACALL	addr11	Absolute Subroutine Call	2	24
LCALL	addr16	Long Subroutine Call	3	24
RET		Return from Subroutine	1	24
RETI		Return from interrupt	1	24
AJMP	addr11	Absolute Jump	2	24
LJMP	addr16	Long Jump	3	24
SJMP	rel	Short Jump (relative addr)	2	24
JMP	@A+DPTR	Jump indirect relative to the DPTR	1	24
JZ	rel	Jump if Accumulator is Zero	2	24
JNZ	rel	Jump if Accumulator is Not Zero	2	24
CJNE	A,direct,rel	Compare direct byte to Acc and Jump if Not Equal	3	24
CJNE	A,#data,rel	Compare immediate to Acc and Jump if Not Equal	3	24
CJNE	R _n ,#data,rel	Compare immediate to register and Jump if Not Equal	3	24
CJNE	@R _i ,#data,rel	Compare immediate to indirect and Jump if Not Equal	3	24
DJNZ	R _n ,rel	Decrement register and Jump if Not Zero	2	24
DJNZ	direct,rel	Decrement direct byte and Jump if Not Zero	3	24
NOP		No Operation	1	12

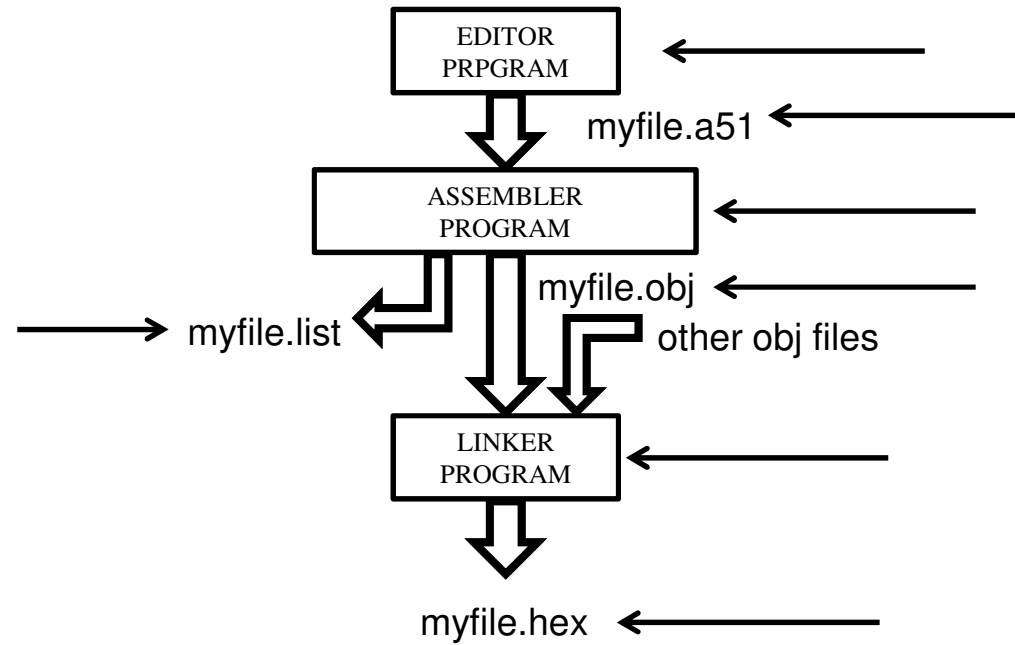
單元三
8051的組合語言程式設計
PART C

3.5 組合語言程式範例

- .asm or .a51 file

ORG 0H		;start (origin) at 0
MOV R5,#25H		;load 25H into R5
MOV R7,#34H		;load 34H into R7
MOV A,#0		;load 0 into A
ADD A,R5		;add contents of R5 to A
		;now A = A + R5
ADD A,R7		;add contents of R7 to A
		;now A = A + R7
ADD A,#12H		;add to A value 12H
		;now A = A + 12H
HERE: SJMP HERE		;stay in this loop
END		;end of asm source file

3.6 程式產生步驟



3.6.1 List file

ROM位址 機器碼

1	0000		ORG 0H	;start (origin) at 0
2	0000	7D25	MOV R5,#25H	;load 25H into R5
3	0002	7F34	MOV R7,#34H	;load 34H into R7
4	0004	7400	MOV A,#0	;load 0 into A
5	0006	2D	ADD A,R5	;add contents of R5 to A ;now A = A + R5
6	0007	2F	ADD A,R7	;add contents of R7 to A ;now A = A + R7
7	0008	2412	ADD A,#12H	;add to A value 12H ;now A = A + 12H
8	000A	80FE	HERE: SJMP HERE	;stay in this loop
9	000C		END	;end of asm source file

3.6.2 ROM contents

ROM Address	Machine Language	Assembly Language
0000	7D25	MOV R5,#25H
0002	7F34	MOV R7,#34H
0004	7400	MOV A,#0
0006	2D	ADD A,R5
0007	2F	ADD A,R7
0008	2412	ADD A,#12H
000A	80FE	HERE: SJMP HERE

ROM Address	Code
0000	7D
0001	25
0002	7F
0003	34
0004	74
0005	00
0006	2D
0007	2F
0008	24
0009	12
000A	80
000B	FE

3.7 8051假指令和資料型態

- ORG (origin)
 - 用來設定起始位址
- DB (define bytes)
 - 用來定義8位元資料

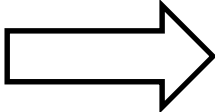
		ORG	100H	
0100H →	DATA1:	DB	28	;DECIMAL(1C in hex)
0101H →	DATA2:	DB	00110100B	;Binary(34 in hex)
0102H →	DATA3:	DB	37H	
		ORG	110H	
0110H →	DATA4:	DB	"2591"	;ASCII numbers
		ORG	118H	
0118H →	DATA6:	DB	"My name is Jenny"	;ASCII characters

3.7 8051資料型態和假指令(continued)

- EQU (equate)
 - 用來定義一常數，但不會佔記憶體空間。

–

Count EQU 25
.....
MOV A, #Count



MOV A, #25

- END directive
 - 讓組譯器知道這是程式結束的地方。

單元三
8051的組合語言程式設計
PART D

3.8 Program status word (PSW) 暫存器

(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 an odd/even number of "one" bits in the accumulator, i.e., even parity.			

Note:

The contents of (RS1, RS0) enable the working register banks as follows:

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

3.9 會影響旗標變化的指令

Instruction	Flag			Instruction	Flag		
	C	OV	AC		C	OV	AC
ADD	X	X	X	CLR C	0		
ADDC	X	X	X	CPL C	X		
SUBB	X	X	X	ANL C,bit	X		
MUL	0	X		ANL C,/bit	X		
DIV	0	X		ORL C,bit	X		
DA	X			ORL C,/bit	X		
RRC	X			MOV C,bit	X		
RLC	X			CJNE	X		
SETB C	1						

Note: X can be 0 or 1

3.9.1 旗標變化範例

請問 CY, AC, 和 P 旗標在用下面指令做完 9CH 加 84H 後的狀態。

Solution:

```
MOV    A,#9CH
ADD    A,#84H
```

```
    9C
  + 84
  ----
   120
```

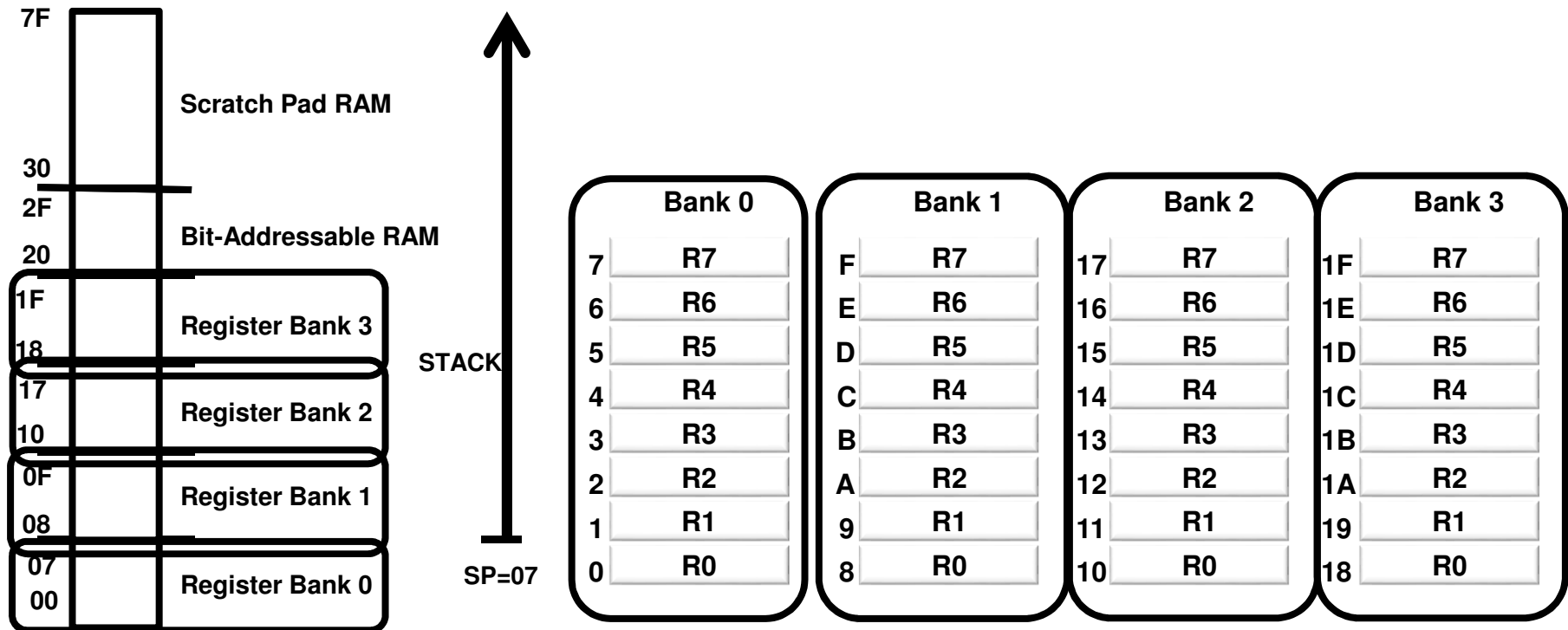
```
    CY  AC
    ↗  ↖
    10011100
  + 10000100
  ----
   00100000
  (D7.....D0)
```

CY = 1 因為 **D7 bit** 有進位。

AC = 1 因為 **D3 bit** 有進位到 **D4 bit**。

P = 1 因為 **A** 有奇數個(**odd**) **1s**。

3.10 8051 Register banks 和 STACK



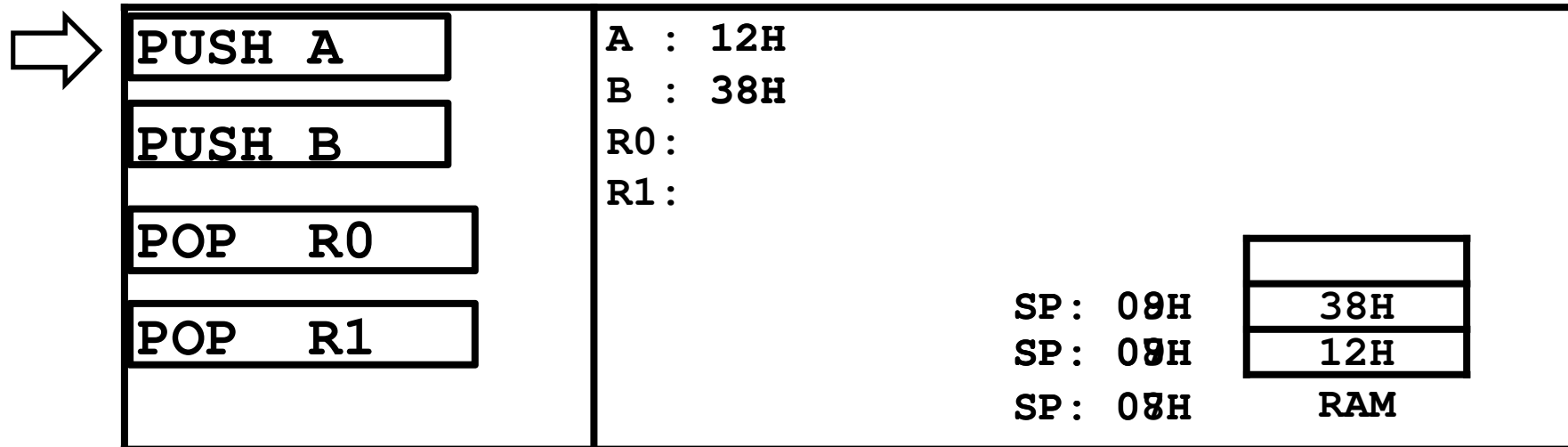
3.11 8051的堆疊

- 堆疊(stack)是 RAM 的一部分，可以讓CPU用來暫存一些資訊。
- 依後進先出 (LIFO, Last In First Out) 的原理運作。
- 用來存取堆疊的暫存器為 SP (stack pointer)暫存器。

3.11 8051的堆疊

- 當 8051 開機時(powered up) , SP暫存器的值為 07 。
- 當 push 資料到堆疊時, SP會加 1 。
- 每次 pop, 堆疊最上面的那個byte會被複製到指令所指定的暫存器且SP會減 1 。

3.11 8051的堆疊(動畫)



單元三
8051的組合語言程式設計
PART E

3.12 迴圈與跳躍指令

- 在8051, "DJNZ reg, label" 可以用來做迴圈(loop)。

Example: 先清除A, 然後 $A=A+5$ 做 10 次。

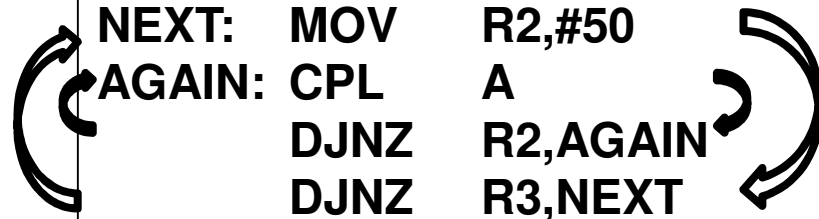
			R2=10	R2=9	R2=8	R2=0
	ORG	100H				
	MOV	A,#0	←			;A = 0, clear ACC
	MOV	R2,#10	←			;load, counter R2 = 10
↻	AGAIN:	ADD	A,#05	←		;add 05 to add
		DJNZ	R2,AGAIN	←		;repeat until R2 = 0(10 times)
		MOV	R5,A	←		;save A in R5

3.12 迴圈與跳躍指令(continued)

- 巢狀迴圈
 - 迴圈需超過256次時

Example: A = 55H, 然後取 A 的補數 500 次。

	MOV	A,#55H		;A = 55H
	MOV	R3,#10		;R3 = 10, the outer loop count
	MOV	R2,#50		;R2 = 50, the inner loop count
NEXT:	MOV	R2,#50		;R2 = 50, the inner loop count
AGAIN:	CPL	A		;complement A register
	DJNZ	R2,AGAIN		;repeat it 50 times(inner loop)
	DJNZ	R3,NEXT		



3.12 迴圈與跳躍指令(continued)

- 其他跳躍指令

指令	動作
JZ	Jump if A = 0
JNZ	Jump if A ≠ 0
DJNZ	Decrement and jump if register ≠ 0
CJNE A, data	Jump if A ≠ data
CJNE reg, #data	Jump if byte ≠ #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

3.13 8051的機器週期

- CPU需要花一些 **clock cycles** 去執行一個指令(instruction).
- 在 8051 , 這些clock cycles 叫做**機器週期** (*machine cycles*).
- 在**original 8051**, 一個 machine cycle 為 12 個 oscillator periods.
- 因此 , 8051的 1 個 machine cycle = crystal frequency/12 的倒數.

有三種不同的crystal頻率在8051-based系統 , 請找出每一種crystal頻率下的機器週期(Machine cycle)。

(a) 11.0592MHz (b) 16MHz (c) 20MHz

解答：

(a) $11.0592 \text{ MHz} / 12 = 921.6 \text{ kHz}$; machine cycle = $1/921.6 \text{ kHz} = 1.085\mu\text{s}$ ←

(b) $16 \text{ MHz} / 12 = 1.333 \text{ MHz}$; machine cycle(MC) = $1/1.333 \text{ MHz} = 0.75\mu\text{s}$ ←

(c) $20 \text{ MHz} / 12 = 1.66\text{MHz}$; MC = $1/ 1.66\text{MHz} = 0.6\mu\text{s}$ ←

3.14 時間延遲(Time Delay)

如果crystal frequency 是 11.0592 MHz，請計算Time delay的大小？

```
AGAIN:  MOV    A,#66H           ;load A with 66H
        MOV    P1,A            ;issue value in reg A to port
        ACALL  DELAY           ;time delay
        CPL    A               ;complement reg A
        SJMP   AGAIN           ;keep doing this indefinitely
```

;---- Time delay

```
DEALY:  MOV    R2,#200          ;load R2 with 200
HERE:   DJNZ   R2,HERE          ;stay here until R2 become 0
        RET                    ;return to caller
```

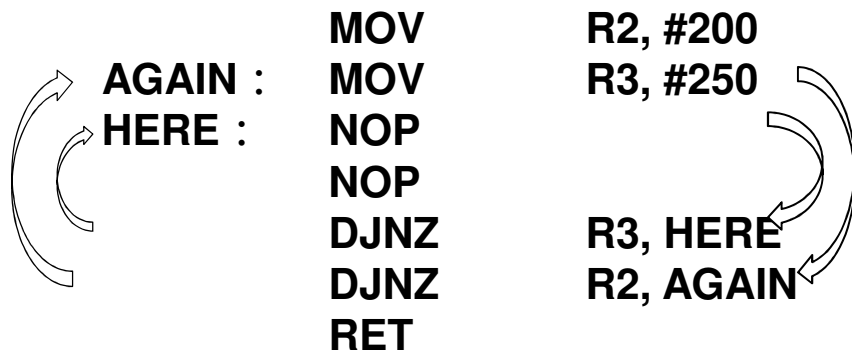
			Machine Cycle
DEALY:	MOV	R2,#200	1
HERE:	DJNZ	R2,HERE	2
	RET		2

Time Delay = $[(200 \times 2) + 1 + 2] \times 1.085 \mu\text{s} = 437.255 \mu\text{s}$.

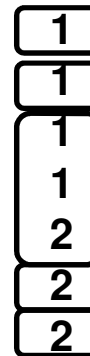
3.14 時間延遲 (continued)

如果1個machine cycle為1.085μs，請問以下程序的DELAY時間(time delay)為？

DELAY :



Machine cycle



HERE loop = $[(1+1+2) \times 250] \times 1.085\mu\text{s} = 1085\mu\text{s}$.

AGAIN loop = $[(2+1) \times 200] \times 1.085\mu\text{s} + 1085\mu\text{s} \times 200 = 217.651\text{ms}$

DELAY total = $217.651\text{ms} + [(2+1) \times 1.085\mu\text{s}] = 217654.255\text{ us}$

單元三
8051的組合語言程式設計
PART F

3.15 直接與間接定址法

- 請寫一程式將資料88H複製到RAM記憶體的40H~44H這五個記憶體位置，請使用以下的方法：
 - (a)直接定址法(Direct addressing mode)
 - (b)暫存器間接定址法(register indirect addressing mode)，無迴圈。
 - (c)暫存器間接定址法(register indirect addressing mode)，用迴圈。

(a)	MOV	A,#088H	←	;load A with value 88H
	MOV	40H,A	←	;copy A to RAM location 40H
	MOV	41H,A	←	;copy A to RAM location 41H
	MOV	42H,A	←	;copy A to RAM location 42H
	MOV	43H,A	←	;copy A to RAM location 43H
	MOV	44H,A	←	;copy A to RAM location 44H

3.15 直接與間接定址法(continued)

(b)

MOV	A,#088H	←	;load A with value 88H
MOV	R0,#40H	←	;load the pointer. <u>R0 = 40H</u>
MOV	@R0,A	←	;copy A to RAM location R0 points to
INC	R0	←	;increment pointer. Now <u>R0 = 41H</u>
MOV	@R0,A	←	;copy A to RAM location R0 points to
INC	R0	←	;increment pointer. Now <u>R0 = 42H</u>
MOV	@R0,A	←	;copy A to RAM location R0 points to
INC	R0	←	;increment pointer. Now <u>R0 = 43H</u>
MOV	@R0,A	←	;copy A to RAM location R0 points to
INC	R0	←	;increment pointer. Now <u>R0 = 44H</u>
MOV	@R0,A	←	

3.15 直接與間接定址法(continued)

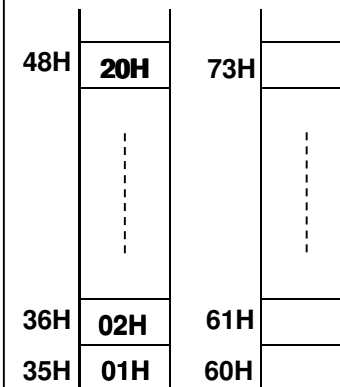
(c)

	MOV	A,#088H	←	;A = 88H
	MOV	R0,#40H	←	;load the pointer. R0 = 40H, RAM address
	MOV	R2,#05	←	;load counter, R2 = 5
AGAIN:	MOV	@R0,A	←←	;copy 55H to RAM location R0 points to
	INC	R0	←←	;increment R0 pointer
	DJNZ	R2,AGAIN	←←	;loop until counter = zero

3.15 直接與間接定址法(continued)

- 從位址35H開始的RAM位置搬一塊20 bytes的資料到位址60H開始的RAM位置。

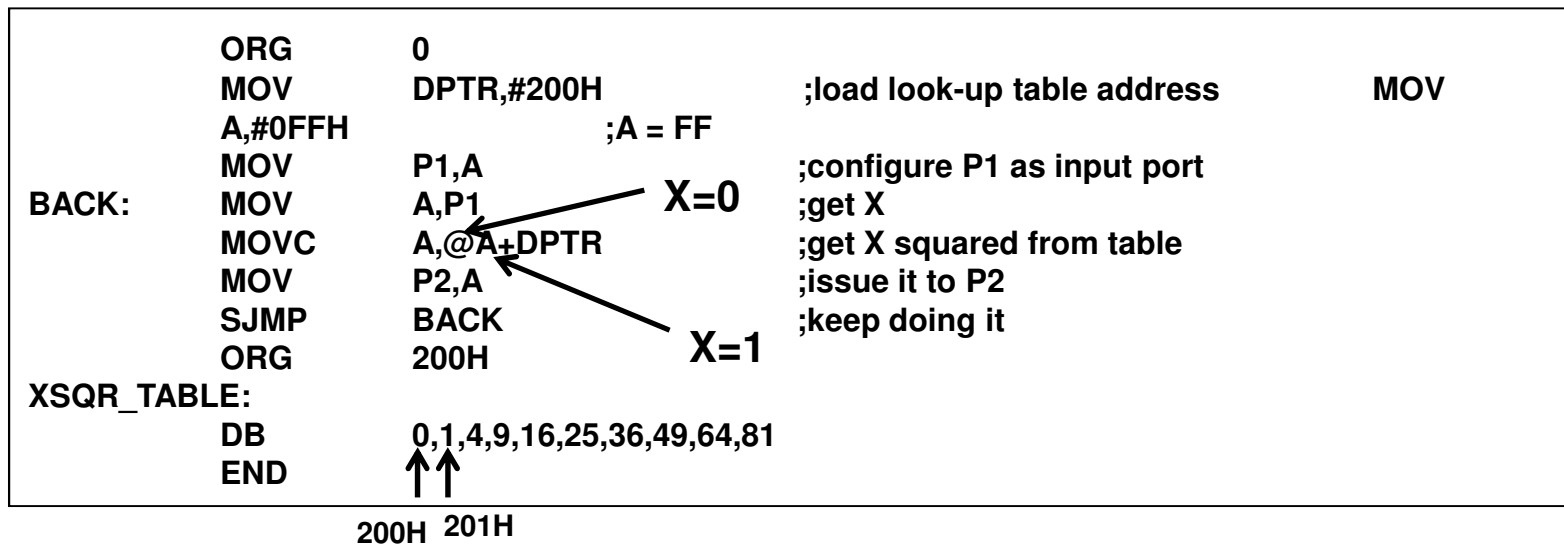
→MOV	R0,#35H	;source pointer
→MOV	R1,#60H	;destination pointer
→MOV	R3,#20	;counter
BACK: →MOV	A,@R0	;get a byte from source
→MOV	@R1,A	;copy it to destination
→INC	R0	;increment source pointer
→INC	R1	;increment destination pointer
→DJNZ	R3,BACK	;Keep doing it for all ten bytes



A = 00H

3.16 查表法(Lookup Table)

- 連續從P1輸入取得x，計算 x^2 後由P2送出。




單元三
8051的組合語言程式設計
PART G

3.17 數學運算指令

- ADD
 - Signed (有號數) 8-bit operands


MOV	A,#+96	;A 0110 0000 (A = 60H)
MOV	R1,#+71	;R1 = 0100 0111 (R1 = 47H)
ADD	A,R1	;A = A7H = -89 decimal, INVALID!

		
+96	0110 0000	
+ +71	+ 0100 0111	
+ 167	1010 0111	and OV = 1

3.17 數學運算指令(continued)

- ADD
 - Signed 8-bit operands

MOV	A,#-128	;A = 1000 0000 (A = 80H)
MOV	R5,#-2	;R5 = 1111 1110 (R5 = FEH)
ADD	A,R5	;A = 0111 1110 (A = 7EH =+126, invalid)

			
	-128	1000 0000	
+ 	-2	+1111 1110	
	-130	0111 1110	and OV = 1

3.18 邏輯運算指令

- AND

```
MOV  A,#31H      ;A = 31H
ANL  A,#0FH      ;A = A AND 0FH (now A = 01)
```

31H	0011 0001	
0FH	0000 1111	
<u>01H</u>	<u>0000 0001</u>	31H AND 0FH = 01H

3.18 邏輯運算指令(continued)

- OR

```
MOV  A,#01H
ORL  A,#30H
```

01H	0000 0001	
30H	0011 0000	
<hr/>	<hr/>	
31H	0011 0001	01H OR 30H = 31H

3.18 邏輯運算指令(continued)

- XOR

```
MOV  A,#55H
XRL  A,#78H
```

```
55H  0101 0101
```

```
78H  0111 1000
```

```
2DH  0010 1101  55H XOR 78H = 2DH
```

3.18 邏輯運算指令(continued)

- 找 84H 的 2's 補數。

```
MOV  A,#84H
CPL  A           ;1's comp.
ADD  A,#1       ;2's comp.
```

```
84H = 1000  0100
1's = 0111  1011
      -----
           +1
      0111  1100 = 7CH
```

3.19 比較指令

- CJNE

```
    MOV    A,#44H
    CJNE   A,#99H,NEXT
    ...
NEXT : ...
    ...
```

(a) It jumps when 44H and 99H are not equal.

(b) A = 44H, its original value before the comparison.

單元三
8051的組合語言程式設計
PART H

3.20 旋轉指令

- RR



- RL



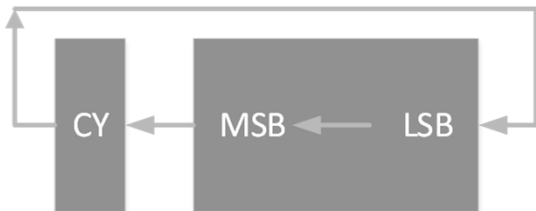
RR	A	;rotate right A	
MOV	A,#36H	;A = 0011 0110	←
RR	A	;A = 0001 1011	←
RR	A	;A = 1000 1101	←
RR	A	;A = 1100 0110	←
RR	A	;A = 0110 0011	
RL	A	;rotate left A	
MOV	A,#72H	;A = 0111 0010	←
RL	A	;A = 1110 0100	←
RL	A	;A = 1100 1001	←

3.20 旋轉指令(continued)

- RRC



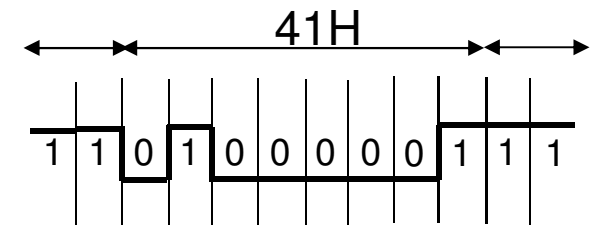
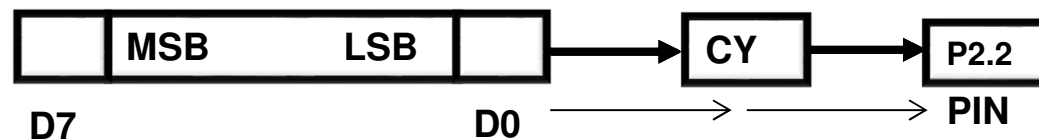
- RLC



RRC	A	;rotate right through carry	
CLR	C	;make CY = 0	
MOV	A,#26H	;A = 0010 0110	←
RRC	A	;A = 0001 0011 CY = 0	←
RRC	A	;A = 0000 1001 CY = 1	←
RRC	A	;A = 1000 0100 CY = 1	←
RLC	A	;rotate left through carry	
SETB	C	;make CY = 1	←
MOV	A,#15H	;A = 0001 0101	←
RLC	A	;A = 0010 1011 CY = 0	←
RLC	A	;A = 0101 0110 CY = 0	←
RLC	A	;A = 1010 1100 CY = 0	←
RLC	A	;A = 0101 1000 CY = 1	←

3.20 旋轉指令(continued)

- 將數值 41H 串列地一次一bit (one bit at a time) 由 P2.2 腳送出. 由LSB 開始送，並在開始和結束時各放兩個High。

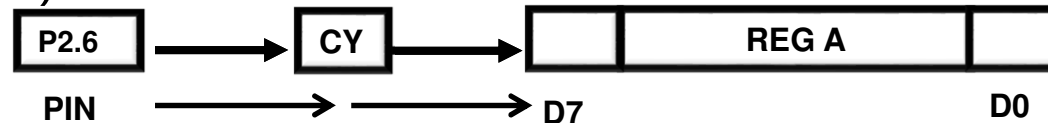


```

MOV     A,#41H  <-----
SETB    P2.2   <-----      ;high
SETB    P2.2   <-----      ;high
MOV     R4,#8   <-----
HERE:   RRC     A    <-----
MOV     P2.2,C  <-----      ;send the carry bit to P2.2
DJNZ    R4,HERE <-----
SETB    P2.2   <-----      ;high
SETB    P2.2   <-----      ;high
    
```

3.20 旋轉指令(continued)

- 經由 P2.6 接收—8個位元的串列資料並將結果放到R1。(從LSB開始接收)



```
MOV    R4,#8      ←  
HERE:  MOV    C,P2.6  ← ;bring in bit  
       RRC     A      ←  
       DJNZ   R4,HERE ←  
       MOV    R1,A    ← ;save it
```

3.21 Checksum

- 我們有4 bytes的資料 : 25H, 62H, 3FH, and 52H.
 - (a)計算 checksum byte,
 - (b)做 checksum 檢查確定資料完整性, and
 - (c)如果第二個byte 62H 變成 22H , checksum 如何偵測錯誤。

(a) 計算checksum byte.

$$\begin{array}{r} 25H \\ + 62H \\ + 3FH \\ + 52H \\ \hline 118H \end{array}$$

checksum byte
= ~18H+1 = E8H

(b) 執行checksum檢查

$$\begin{array}{r} 25H \\ + 62H \\ + 3FH \\ + 52H \\ + E8H \\ \hline 200H \end{array}$$

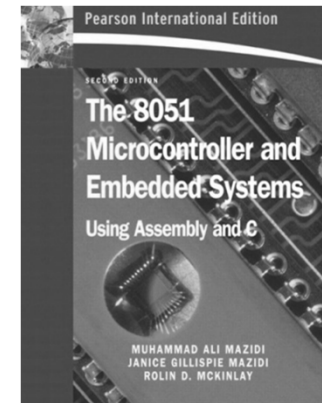
3.21 Checksum (continued)

(c) 如果第二個byte 62H 變成 22H，checksum 如何偵測錯誤。

$$\begin{array}{r} 25\text{H} \\ +22\text{H} \\ +3\text{FH} \\ +52\text{H} \\ +\text{E8H} \\ \hline 1\text{C0H} \end{array}$$

3.22 參考文獻

- ATMEL AT89S51 datasheet (doc2487.pdf)
- ATMEL 8051 Microcontrollers Hardware Manual (doc4316.pdf)
- ATMEL 8051 Microcontroller Instruction Set (doc0509.pdf)
- The 8051 Microcontroller and Embedded Systems Using Assembly and C, Second Edition, by Muhammad Ali Mazidi, Janice Gillispie Mazidi, Rolin D. McKinlay.



3.23 複習題

- 8051的組合語言格式?
- 8051有哪幾種定址法?
- 8051有哪些指令?
- 甚麼是機器週期? 如何計算時間延遲?
- 甚麼是Checksum byte?