

微介實驗一

內部RAM與Flash空間定址與存取

日期：9月24日

報告者：許宸華

Outline

- 實驗內容
- 學習重點
- 組合語言格式與指令
- 資料記憶體(RAM)
- 六種定址法與程式碼
- 程式記憶體(Flash)
- Keil C51實際操作

Outline

- 實驗內容
- 學習重點
- 組合語言格式與指令
- 資料記憶體(RAM)
- 六種定址法與程式碼
- 程式記憶體(Flash)
- Keil C51實際操作

實驗內容

- 利用8051的六種定址法變更A暫存器的內容
- 使用Keil C51 debug mode觀察8051內部暫存器的變化
- 本週基礎題課程為純軟體模擬，不用接電路

Outline

- 實驗內容
- 學習重點
- 組合語言格式與指令
- 資料記憶體(RAM)
- 六種定址法與程式碼
- 程式記憶體(Flash)
- Keil C51實際操作

學習重點

- 8051 內部記憶體 (Flash與RAM) 配置
- 8051 暫存器庫使用方式
- 8051 的各種定址法，使用的指令包含了MOV、MOVC、SETB、CLR

Outline

- 實驗內容
- 學習重點
- 組合語言格式與指令
- 資料記憶體(RAM)
- 六種定址法與程式碼
- 程式記憶體(Flash)
- Keil C51實際操作

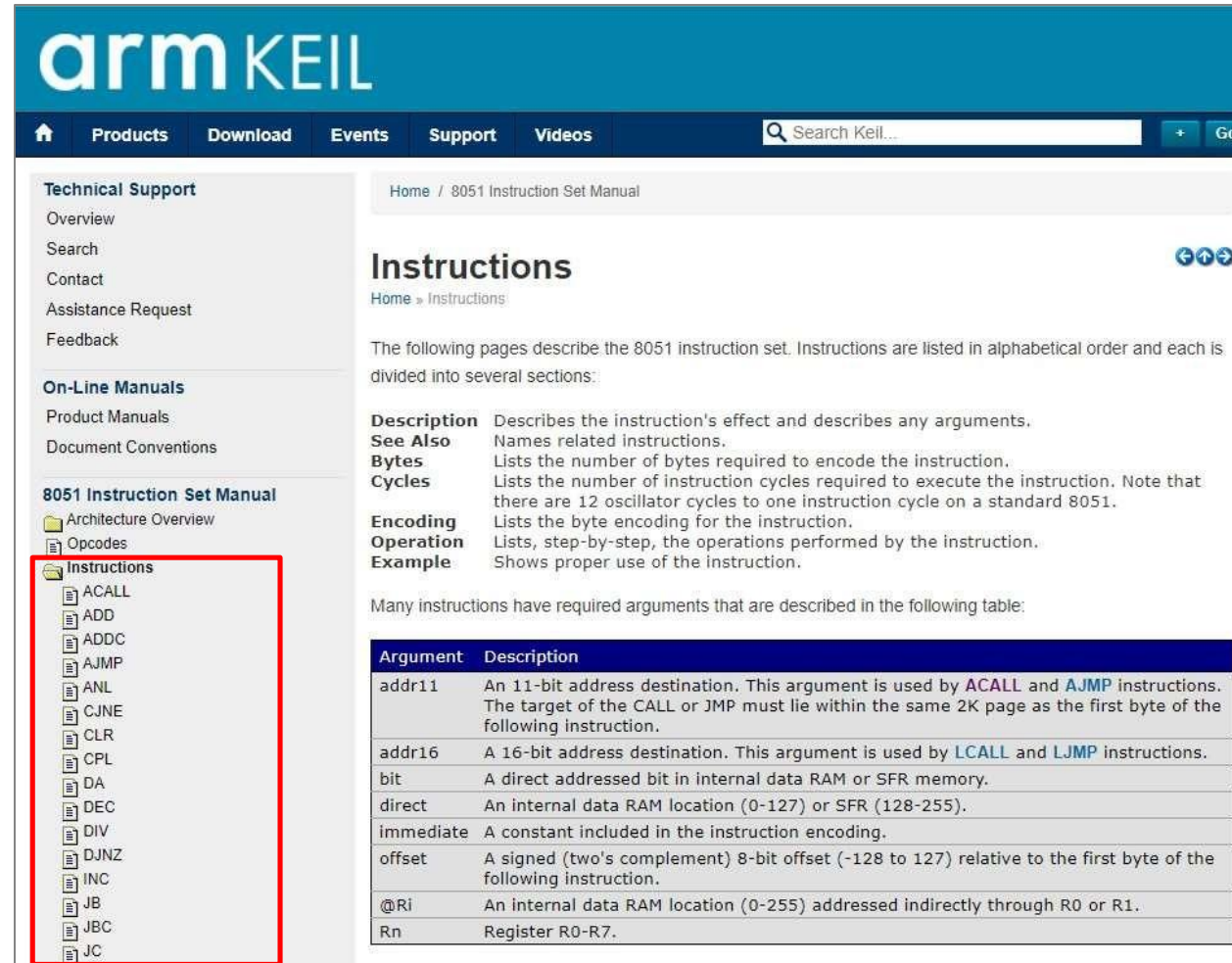
組合語言格式

- 程式經由組譯器組譯後，運算碼及運算元會產生機械碼
- 標籤相當於程式記憶體的位置

	Label 標籤	Mnemonic 運算碼	Operand 運算元	Comment 註解
1		<i>ORG</i>	<i>0</i>	
2	<i>LOOP:</i>	<i>MOV</i>	<i>P1, A</i>	
3		<i>CALL</i>	<i>DELAY</i>	<i>;delay 0.5 sec</i>
4		<i>RL</i>	<i>A</i>	
5		<i>JMP</i>	<i>LOOP</i>	
6	<i>DELAY:</i>	<i>...</i>		

指令Instruction

- 組合語言指令集查詢 **8051 Instruction Set Manual – Keil**



arm KEIL

Home / 8051 Instruction Set Manual

Instructions

The following pages describe the 8051 instruction set. Instructions are listed in alphabetical order and each is divided into several sections:

- Description** Describes the instruction's effect and describes any arguments.
- See Also** Names related instructions.
- Bytes** Lists the number of bytes required to encode the instruction.
- Cycles** Lists the number of instruction cycles required to execute the instruction. Note that there are 12 oscillator cycles to one instruction cycle on a standard 8051.
- Encoding** Lists the byte encoding for the instruction.
- Operation** Lists, step-by-step, the operations performed by the instruction.
- Example** Shows proper use of the instruction.

Many instructions have required arguments that are described in the following table:

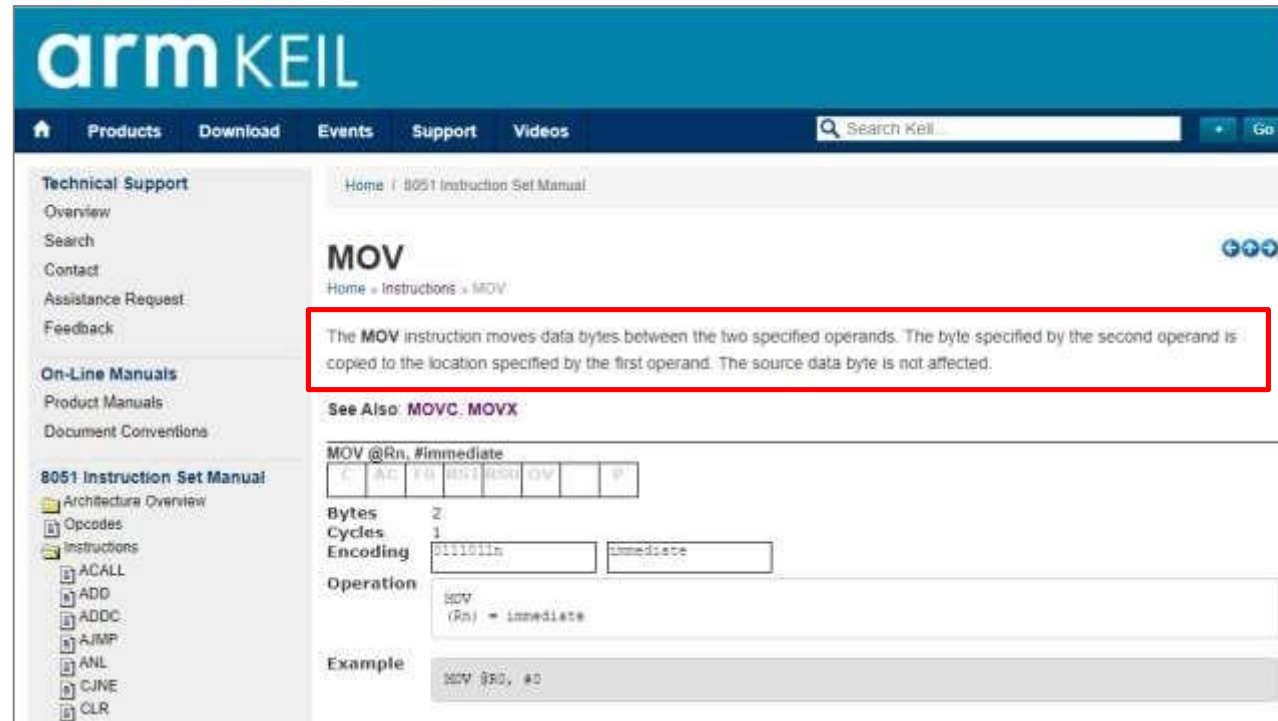
Argument	Description
addr11	An 11-bit address destination. This argument is used by ACALL and AJMP instructions. The target of the CALL or JMP must lie within the same 2K page as the first byte of the following instruction.
addr16	A 16-bit address destination. This argument is used by LCALL and LJMP instructions.
bit	A direct addressed bit in internal data RAM or SFR memory.
direct	An internal data RAM location (0-127) or SFR (128-255).
immediate	A constant included in the instruction encoding.
offset	A signed (two's complement) 8-bit offset (-128 to 127) relative to the first byte of the following instruction.
@Ri	An internal data RAM location (0-255) addressed indirectly through R0 or R1.
Rn	Register R0-R7.

指令Instruction-MOV

- **MOV** destination, source ;copy source to destination

範例一： **MOV** A, #55H ;A暫存器存入55H的數值

範例二： **MOV** R0, A ;複製A暫存器的內容到R0



The screenshot shows the ARM KEIL website's technical support page for the MOV instruction. The page title is "MOV" and it includes a breadcrumb trail "Home » Instructions » MOV". A red box highlights the following text: "The **MOV** instruction moves data bytes between the two specified operands. The byte specified by the second operand is copied to the location specified by the first operand. The source data byte is not affected." Below this, there is a section "See Also: MOV, MOVX" and a table showing the instruction format: MOV @Rn, #Immediate. The table lists the instruction's bytes, cycles, encoding, and operation. The operation is shown as MOV (Rn) = Immediate. An example is provided: MOV R0, #0.

The **MOV** instruction moves **data bytes** between the two specified operands.

The byte specified by the second operand is copied to the location specified by the first operand. The source data byte is not affected.

Outline

- 實驗內容
- 學習重點
- 組合語言格式與指令
- 資料記憶體(RAM)
- 六種定址法與程式碼
- 程式記憶體(Flash)
- Keil C51實際操作

8051 memory map

- 此表為8051內部RAM的memory map
- 主要可以拆分為4個區域
 - 可位元定址區
Bit Addressable Area
 - 一般資料存放區/堆疊區
Scratch Pad Area
 - 特殊功能暫存器區
Special Function Register (SFR)
 - 暫存器庫區
Register Bank

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

...									
F0	F7	F6	F5	F4	F3	F2	F1	F0	B
...									
E0	E7	E6	E5	E4	E3	E2	E1	E0	ACC
...									
D0	D7	D6	D5	D4	D3	D2	D1	D0	PSW
...									
B8	BF	BE	BD	BC	BB	BA	B9	B8	IP
...									
B0	B7	B6	B5	B4	B3	B2	B1	B0	P3
...									
A8	AF	AE	AD	AC	AB	AA	A9	A8	IE
...									
A0	A7	A6	A5	A4	A3	A2	A1	A0	P2
...									
99									SBUF
98	9F	9E	9D	9C	9B	9A	99	98	SCON
...									
90	97	96	95	94	93	92	91	90	P1
...									
8D									TH1
8C									TH0
8B									TL1
8A									TL0
89									TMOD
88	8F	8E	8D	8C	8B	8A	89	88	TCON
87									PCON
...									
83									DPH
82									DPL
81									SP
80	87	86	85	84	83	82	81	80	P0

可位元定址區

Bit Addressable Area

- 位於8051內部RAM的20H~2FH位址
- 可以使用SETB、CLR指令控制某個位址byte中的一個bit數值
- 在Keil C51中，可以利用以下幾種指令修改此記憶體的值
- *MOV 20H, #01H*
- *SETB 00H*
- *SETB 20H.0*

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

一般資料存放區/ 堆疊區

Scratch Pad Area

- 位於8051內部RAM的30H~7FH位址
- 僅可一次修改一整個byte的值
- 在Keil C51中，可以利用以下幾種指令修改此記憶體的值
- *MOV 30H, #01H*

附註：在設計程式時，經常會使用SP暫存器搭配PUSH與POP指令利用此區記憶體，詳細方法將在第5章做介紹。

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

14

特殊功能暫存器區

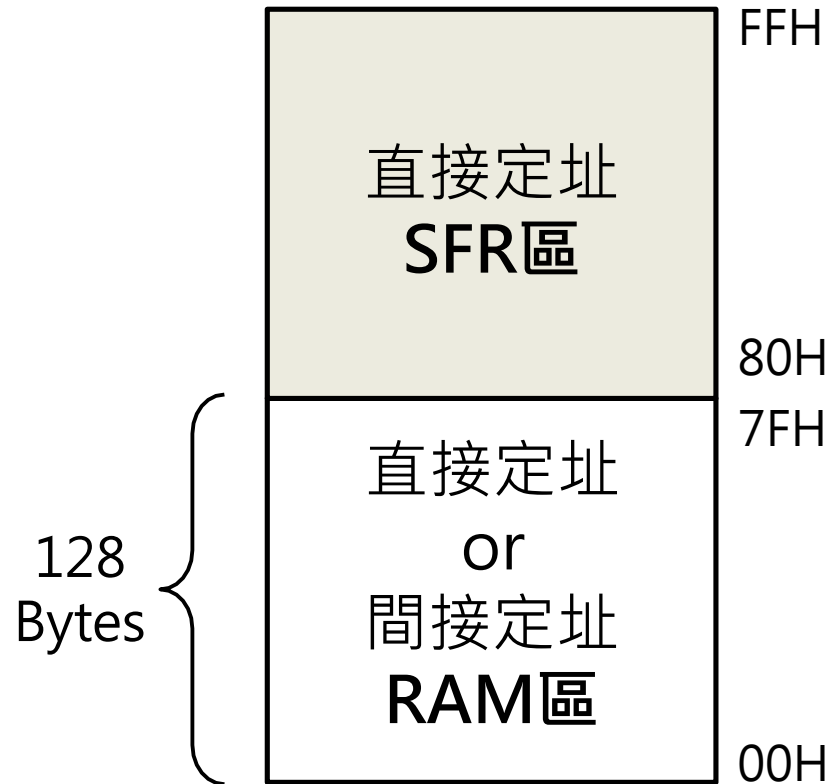
Special Function Register (SFR)

- 位於8051內部RAM的80H~FFH位址
- 此處有許多會影響8051行為模式的暫存器
- 僅可使用直接定址法讀取與修改此位址的值
- 在Keil C51中，可以利用以下幾種指令修改此記憶體的值
- *MOV A, #01H*
- *MOV 0E0H, #01H*
- *SETB 0E0H*
- *SETB 0E0H.0*

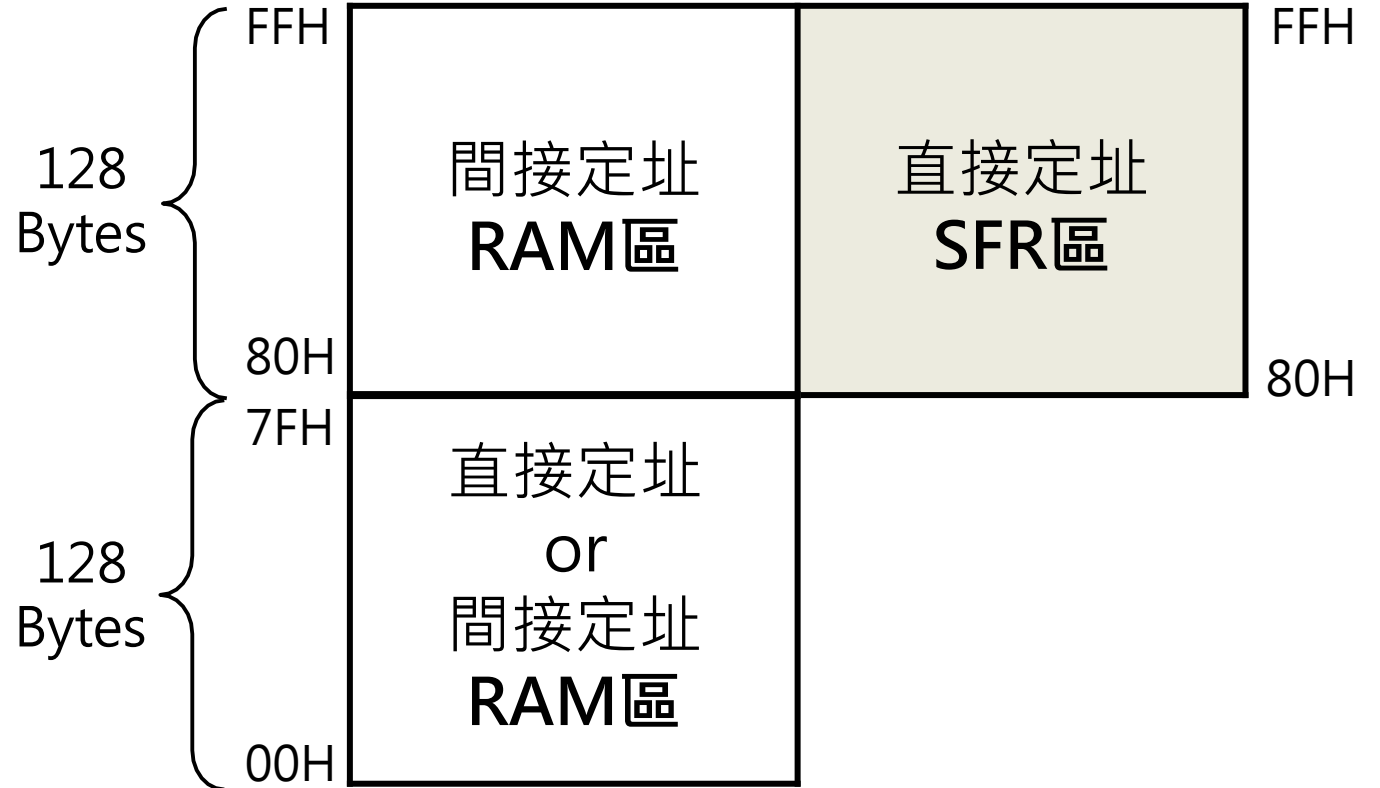
附註：在Keil C51的16進制中，如果要定的位址或數值最高位數 $\geq A$ ，則要在前方加上0。

...									
F0	F7	F6	F5	F4	F3	F2	F1	F0	B
...									
E0	E7	E6	E5	E4	E3	E2	E1	E0	ACC
...									
D0	D7	D6	D5	D4	D3	D2	D1	D0	PSW
...									
B8	BF	BE	BD	BC	BB	BA	B9	B8	IP
...									
B0	B7	B6	B5	B4	B3	B2	B1	B0	P3
...									
A8	AF	AE	AD	AC	AB	AA	A9	A8	IE
...									
A0	A7	A6	A5	A4	A3	A2	A1	A0	P2
...									
99									SBUF
98	9F	9E	9D	9C	9B	9A	99	98	SCON
...									
90	97	96	95	94	93	92	91	90	P1
...									
8D									TH1
8C									TH0
8B									TL1
8A									TL0
89									TMOD
88	8F	8E	8D	8C	8B	8A	89	88	TCON
87									PCON
...									
83									DPH
82									DPL
81									SP
80	87	86	85	84	83	82	81	80	P0

附註：8052內部RAM 80H~FFH



8051內部資料記憶體



8052內部資料記憶體

附註：8052內部RAM 80H~FFH

- 8052有256bytes的內部RAM
 - 直接定址與間接定址定的80H~FFH是不同的記憶體。
 - 要使用內部80H~FFH(128~255)位址的RAM時，必須使用間接定址法來存取。
- 8051只有128bytes的內部RAM
 - 無法用間接定址法操控內部80H~FFH這些位址。

暫存器庫區Register Bank

- 位於8051內部RAM的00H~1FH位址
- 總共有四組Register Bank
- 每組Register Bank皆包含了R0、R1...R7等8個暫存器
- 暫存器庫可由SFR中Program Status Word (PSW) 暫存器中的RS1和RS0做切換。

RS1	RS0	暫存器庫	位址
0	0	RB0	00H~07H
0	1	RB1	08H~0FH
1	0	RB2	10H~17H
1	1	RB3	18H~1FH

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

暫存器庫區Register Bank

若想要使用Register Bank 1，可以用以下幾種指令：

1. *SETB* *RS0*
 CLR *RS1*
2. *SETB* *0D3H*
 CLR *0D4H*
3. *SETB* *0D0H.3*
 CLR *0D0H.4*
4. *MOV* *PSW, #08H*
5. *MOV* *0D0H, #08H*
- (不使用到其他暫存器的話) (不使用到其他暫存器的話)

	7	6	5	4	3	2	1	0
PSW	CY	AC	F0	RS1	RS0	OV		P

附註：關於PSW暫存器將在第4章有更詳細的介紹

...									
F0	F7	F6	F5	F4	F3	F2	F1	F0	B
...									
E0	E7	E6	E5	E4	E3	E2	E1	E0	ACC
...									
D0	D7	D6	D5	D4	D3	D2	D1	D0	PSW
...									
B8	BF	BE	BD	BC	BB	BA	B9	B8	IP
...									
B0	B7	B6	B5	B4	B3	B2	B1	B0	P3
...									
A8	AF	AE	AD	AC	AB	AA	A9	A8	IE
...									
A0	A7	A6	A5	A4	A3	A2	A1	A0	P2
...									
99									SBUF
98	9F	9E	9D	9C	9B	9A	99	98	SCON
...									
90	97	96	95	94	93	92	91	90	P1
...									
8D									TH1
8C									TH0
8B									TL1
8A									TL0
89									TMOD
88	8F	8E	8D	8C	8B	8A	89	88	TCON
87									PCON
...									
83									DPH
82									DPL
81									SP
80	87	86	85	84	83	82	81	80	P0

Outline

- 實驗內容
- 學習重點
- 組合語言格式與指令
- 資料記憶體(RAM)
- 六種定址法與程式碼
- 程式記憶體(Flash)
- Keil C51實際操作

8051六種定址法

- 直接定址法(Direct addressing)
MOV A, direct
- 間接定址法(Indirect addressing)
MOV A, @Ri
- 暫存器定址法(Register addressing)
MOV A, Rn
- 立即定址法(Immediate addressing)
MOV A, #immediate
- 索引定址法(Index addressing)
MOVC A, @A+DPTR
- 位元定址法(Bit addressing)
SETB bit

位元定址法 (Bit addressing)

- 運算元為「可位元定址」的位址
- 在可位元定址區以及部份的SFR中，可以利用SETB或是CLR指令來存取某個byte位址中一個bit的數值

```
1.  ORG 0                                ; code start from 0
2.  MOV 20H, #00H                        ; address 20H = 00H
3.  SETB 20H.1                          ; address 20H bit 1 = 1
4.  MOV A, 20H                          ; A = ?
5.  SETB 03H                            ; bit address 03H = 1
6.  MOV A, 20H                          ; A = ?
7.  SJMP $                              ; infinite loop
8.  END
```

20	07	06	05	04	03	02	01	00
-----------	----	----	----	----	----	----	----	----

立即定址法 (Immediate addressing)

- 運算元為「常數資料」，以「#」號作為前置符號
- 直接存取指定的數值

```
1.  ORG 0                                ; code start from 0
2.  MOV R0, #30H                          ; R0 = 30H
3. MOV R1, #10H 4.                        ; R1 = 10H
   MOV 30H, #55H                          ; address 30H = 55H
5.  MOV 31H, #24H                        ; address 31H = 24H
6.  MOV A, #31H                          ; A = ?
7.  SJMP $                               ; infinite loop
8.  END
```

資料記憶體RAM

Address	Data
FFH	?
...	?
31H	24H
30H	55H
...	?
01H	10H
00H	30H

Register bank 0 R1

Register bank 0 R0

常數資料形式

二進位

- 00000000B~11111111B
- 數字後加「B」做為標示

十進位

- 0~255
- 十進位數字後可加上「D」，或不加任何標示。

十六進位

- 00H~FFH
- 數字後加「H」做為標示，或字首加上「0x」做為前綴。

直接定址法 (Direct addressing)

- 運算元為「位址」，以16進位數值表示
- 直接存取某個位址內的數值

```
1.  ORG 0                                ; code start from 0
2.  MOV R0, #30H                          ; R0 = 30H
3.  MOV R1, #10H                          ; R1 = 10H
4.  MOV 30H, #55H                        ; address 30H = 55H
5.  MOV 31H, #24H                        ; address 31H = 24H
6.  MOV A, 30H                          ; A = ?
7.  SJMP $                               ; infinite loop
8.  END
```

資料記憶體RAM

Address	Data
FFH	?
...	?
31H	24H
30H	55H
...	?
01H	10H
00H	30H

Register bank 0 R1

Register bank 0 R0

間接定址法 (Indirect addressing)

- 運算元為「暫存器」，以「@」作為前置符號
- 將存入R0或R1暫存器的數值當作位址，並且根據這個位址存取資料
- 使用上概念類似C語言的指標

```
1.  ORG 0                                ; code start from 0
2.  MOV R0, #30H                           ; R0 = 30H
3.  MOV R1, #10H                           ; R1 = 10H
4.  MOV 30H, #55H                         ; address 30H = 55H
5.  MOV 31H, #24H                         ; address 31H = 24H
6.  MOV A, @R0                             ; A = ?
7.  SJMP $                                ; infinite loop
8.  END
```

附註：僅有R0與R1可以使用間接定址法

資料記憶體RAM

Address	Data
FFH	?
...	?
31H	24H
30H	55H
...	?
01H	10H
00H	30H

Register bank 0 R1

Register bank 0 R0

暫存器定址法 (Register addressing)

- 運算元為「暫存器」，以R0~R7表示
- 直接存取某個暫存器內的數值

```
1.  ORG 0                                ; code start from 0
2.  MOV R0, #30H                          ; R0 = 30H
3.  MOV R1, #10H                          ; R1 = 10H
4.  MOV 30H, #55H                        ; address 30H = 55H
5.  MOV 31H, #24H                        ; address 31H = 24H
6.  MOV A, R0                            ; A = ?
7.  SJMP $                               ; infinite loop
8.  END
```

資料記憶體RAM

Address	Data
FFH	?
...	?
31H	24H
30H	55H
...	?
01H	10H
00H	30H

Register bank 0 R1

Register bank 0 R0

索引定址法 (Index addressing)

- 運算元同時為「索引暫存器+基底暫存器」，以「@」作為前置符號
- 使用MOVC指令存取程式記憶體的资料
- 運算元所在的位址= 基底暫存器的數值+ 索引暫存器的數值，並且根據這個位址存取資料
- 概念類似C語言中的陣列

基底暫存器：DPTR or PC
索引暫存器：A

```
1.      ORG 0                                ; code start from 0
2.      MOV DPTR, #TABLE                    ; DPTR = 'TABLE' label address
3.      MOV A, #01H                         ; A = 01H
4.      MOVC A, @A+DPTR                     ; A = ?
5.      C:0008H SJMP $                       ; infinite loop
6.      TABLE: DB 01H → C:0008H ; save 01H at 'TABLE' address
7.      DB 02H → C:0009H ; save 02H at 'TABLE' label +1 address
8.      END
```

Outline

- 實驗內容
- 學習重點
- 組合語言格式與指令
- 資料記憶體(RAM)
- 六種定址法與程式碼
- 程式記憶體(Flash)
- Keil C51實際操作

程式碼在Flash的位置

- 根據在Keil C51中下的ORG指令，程式碼會從那個位置開始寫入，接著按照順序與長度填入機械碼
- 查詢8051指令集，可以得知每個指令的長度、消耗時間
- 可以用Keil C51的 Disassembly Window或是Memory Window觀察程式碼最後會被燒錄進8051 Flash的位址與機械碼

程式碼在Flash的位置

ORG及END皆為虛擬指令。
ORG 0告訴 assembler 將
程式碼從程式記憶體的
0000H開始填入。

1. **ORG 0**
2. **MOV R0, #30H**
3. **MOV A, R0**
4. **END**

MOV R0, #data

指令hex code為78，需
要2 Bytes的空間，data
為30。

Hex Code	Number of Bytes	Mnemonic	Operands
66	1	XRL	A,@R0
67	1	XRL	A,@R1
68	1	XRL	A,R0
69	1	XRL	A,R1
6A	1	XRL	A,R2
6B	1	XRL	A,R3
6C	1	XRL	A,R4
6D	1	XRL	A,R5
6E	1	XRL	A,R6
6F	1	XRL	A,R7
70	2	JNZ	code addr
71	2	ACALL	code addr
72	2	ORL	C,bit addr
73	1	JMP	@A+DPTR
74	2	MOV	A,#data
75	3	MOV	data addr,#data
76	2	MOV	@R0,#data
77	2	MOV	@R1,#data
78	2	MOV	R0,#data

程式記憶體Flash

Address	Data
0000H	78
0001H	30
0002H	
0003H	
0004H	
0005H	

因此0000H被assembler
填入指令78，0001H被填
入data 30。

程式碼在Flash的位置

1. *ORG 0*
2. *MOV R0, #30H*
3. *MOV A, R0*
4. *END*

MOV A, R0
指令hex code為E8，需要
1 Byte的空間。

Hex Code	Number of Bytes	Mnemonic	Operands
E6	1	MOV	A,@R0
E7	1	MOV	A,@R1
E8	1	MOV	A,R0
E9	1	MOV	A,R1
EA	1	MOV	A,R2
EB	1	MOV	A,R3
EC	1	MOV	A,R4
ED	1	MOV	A,R5
EE	1	MOV	A,R6
EF	1	MOV	A,R7
F0	1	MOVX	@DPTR,A
F1	2	ACALL	code addr
F2	1	MOVX	@R0,A
F3	1	MOVX	@R1,A

程式記憶體Flash

Address	Data
0000H	78
0001H	30
0002H	E8
0003H	
0004H	
0005H	

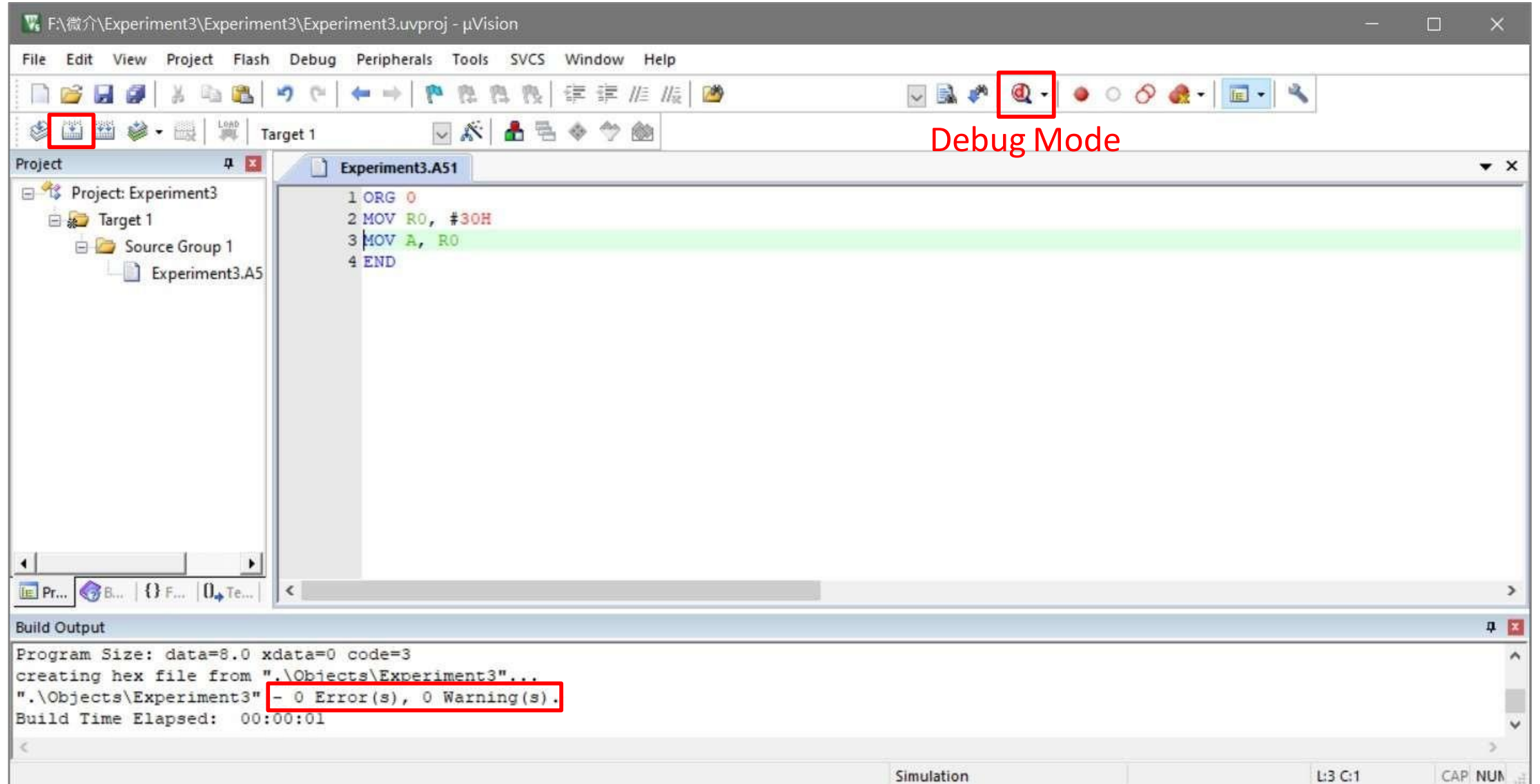
Assembler在接下來的
0002H位址填入 E8H。

Outline

- 實驗內容
- 學習重點
- 組合語言格式與指令
- 資料記憶體(RAM)
- 六種定址法與程式碼
- 程式記憶體(Flash)
- Keil C51實際操作

Keil C51 – Debug Mode

編譯



Keil C51 – 單步執行

F:\微介\Experiment3\Experiment3\Experiment3.uvproj - μVision

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help

Registers

Register	Value
r0	0x30
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
sp	0x07
PC	0x0002
auxr1	0x00
dptr	0x0000
states	1

Disassembly

```
2: MOV R0, #30H
C:0x0000 7830 MOV R0,#0x30
3: MOV A, R0
C:0x0002 E8 MOV A,R0
C:0x0003 00 NOP
C:0x0004 00 NOP
C:0x0005 00 NOP
```

Experiment3.A51

```
1 ORG 0
2 MOV R0, #30H
3 MOV A, R0
4 END
```

Command

Running with Code Size Limit: 2K
Load "F:\\微介\\Experiment3\\Experiment3\\Objects\\Experiment3"

ASM ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAcc

F:\微介\Experiment3\Experiment3\Experiment3.uvproj - μVision

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help

Registers

Register	Value
r0	0x30
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
a	0x30
b	0x00
sp	0x07
PC	0x0003
auxr1	0x00
dptr	0x0000
states	2

Disassembly

```
2: MOV R0, #30H
C:0x0000 7830 MOV R0,#0x30
3: MOV A, R0
C:0x0002 E8 MOV A,R0
C:0x0003 00 NOP
C:0x0004 00 NOP
C:0x0005 00 NOP
```

Experiment3.A51

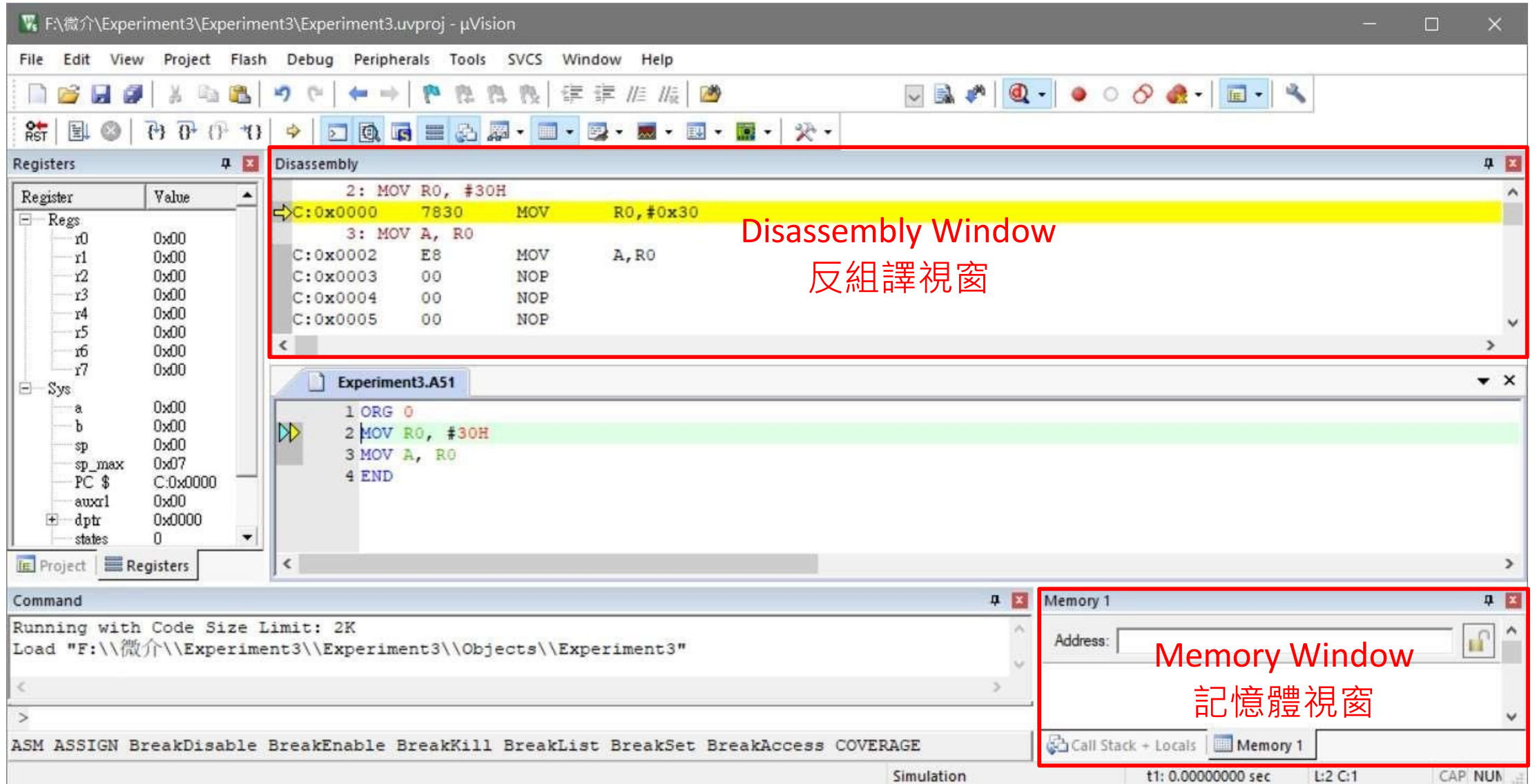
```
1 ORG 0
2 MOV R0, #30H
3 MOV A, R0
4 END
```

Command

Load "F:\\微介\\Experiment3\\Experiment3\\Objects\\Experiment3"
*** error 65: access violation at C:0x0003 : no 'execute/read' permission

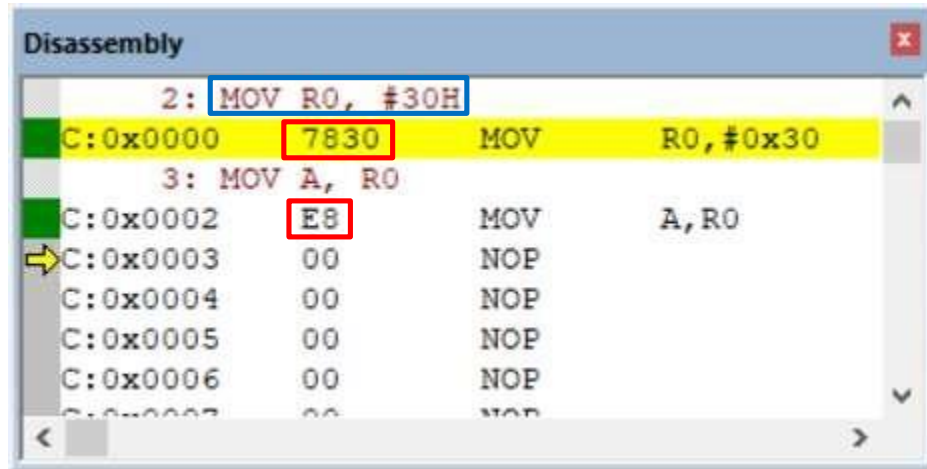
ASM ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAcc

Keil C51

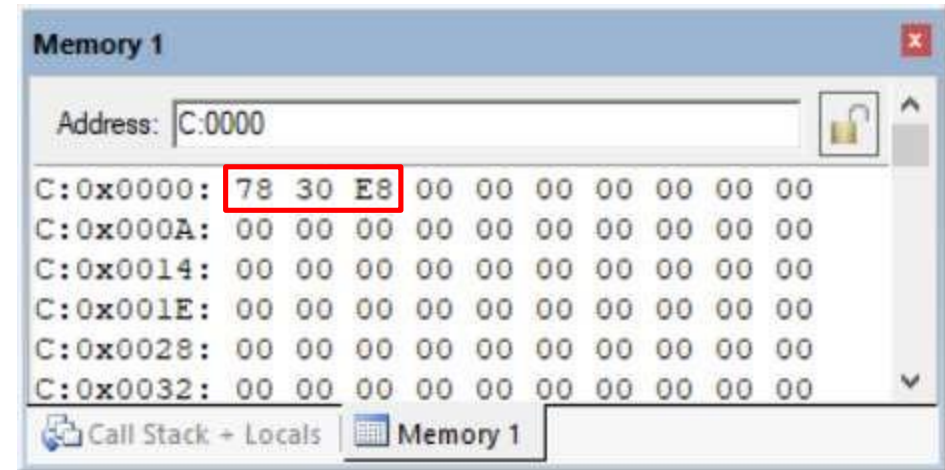


Keil C51

C：程式記憶體

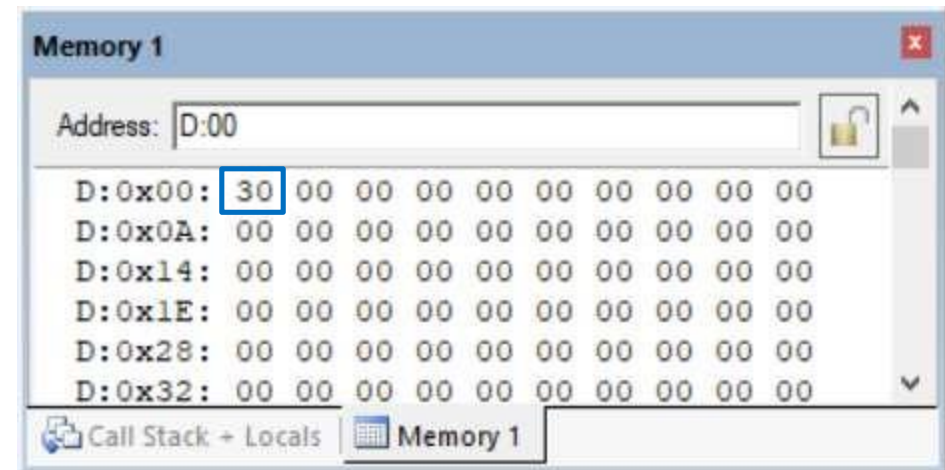


```
Disassembly
2: MOV R0, #30H
C:0x0000 7830 MOV R0, #0x30
3: MOV A, R0
C:0x0002 E8 MOV A, R0
C:0x0003 00 NOP
C:0x0004 00 NOP
C:0x0005 00 NOP
C:0x0006 00 NOP
C:0x0007 00 NOP
```



```
Memory 1
Address: C:0000
C:0x0000: 78 30 E8 00 00 00 00 00 00 00
C:0x000A: 00 00 00 00 00 00 00 00 00 00
C:0x0014: 00 00 00 00 00 00 00 00 00 00
C:0x001E: 00 00 00 00 00 00 00 00 00 00
C:0x0028: 00 00 00 00 00 00 00 00 00 00
C:0x0032: 00 00 00 00 00 00 00 00 00 00
```

D：資料記憶體



```
Memory 1
Address: D:00
D:0x00: 30 00 00 00 00 00 00 00 00 00
D:0x0A: 00 00 00 00 00 00 00 00 00 00
D:0x14: 00 00 00 00 00 00 00 00 00 00
D:0x1E: 00 00 00 00 00 00 00 00 00 00
D:0x28: 00 00 00 00 00 00 00 00 00 00
D:0x32: 00 00 00 00 00 00 00 00 00 00
```

Q & A