第05次組語實習課

學生：林培瑋

2023 Advanced Mixed-Operation System (AMOS) Lab.

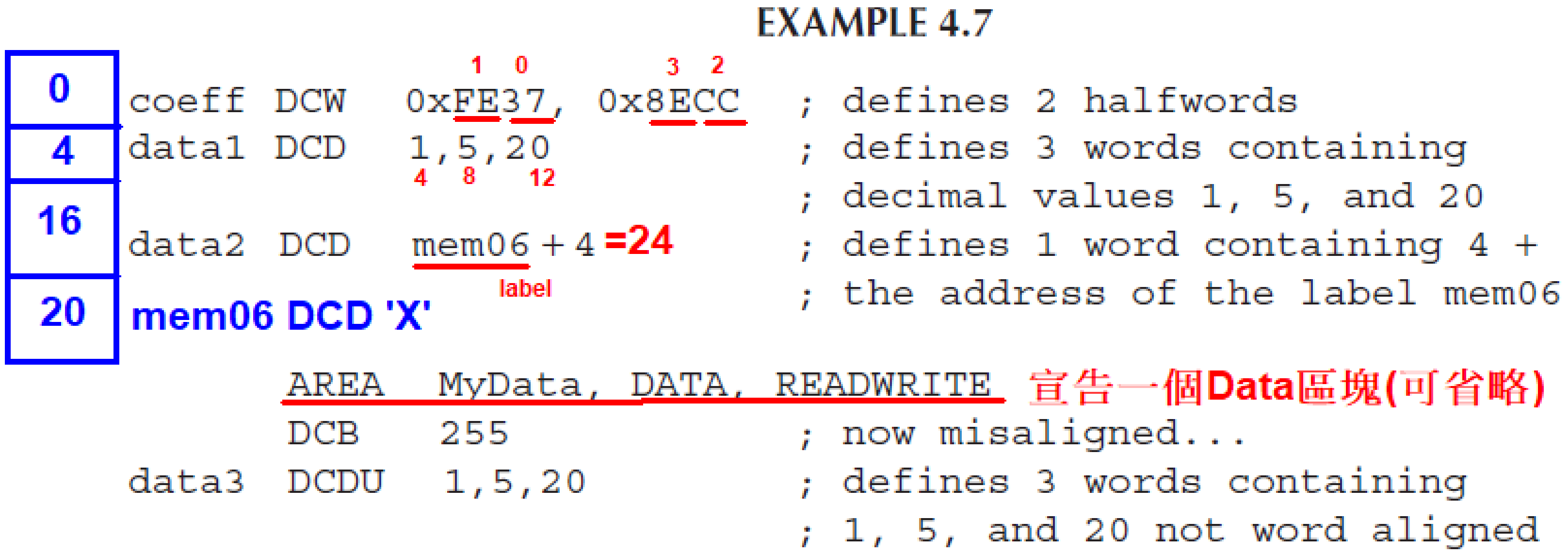
目錄

1017正課複習

2023 Advanced Mixed-Operation System (AMOS) Lab.

p.69 EXAMPLE 4.7

->假設數字表示所在記憶體位置



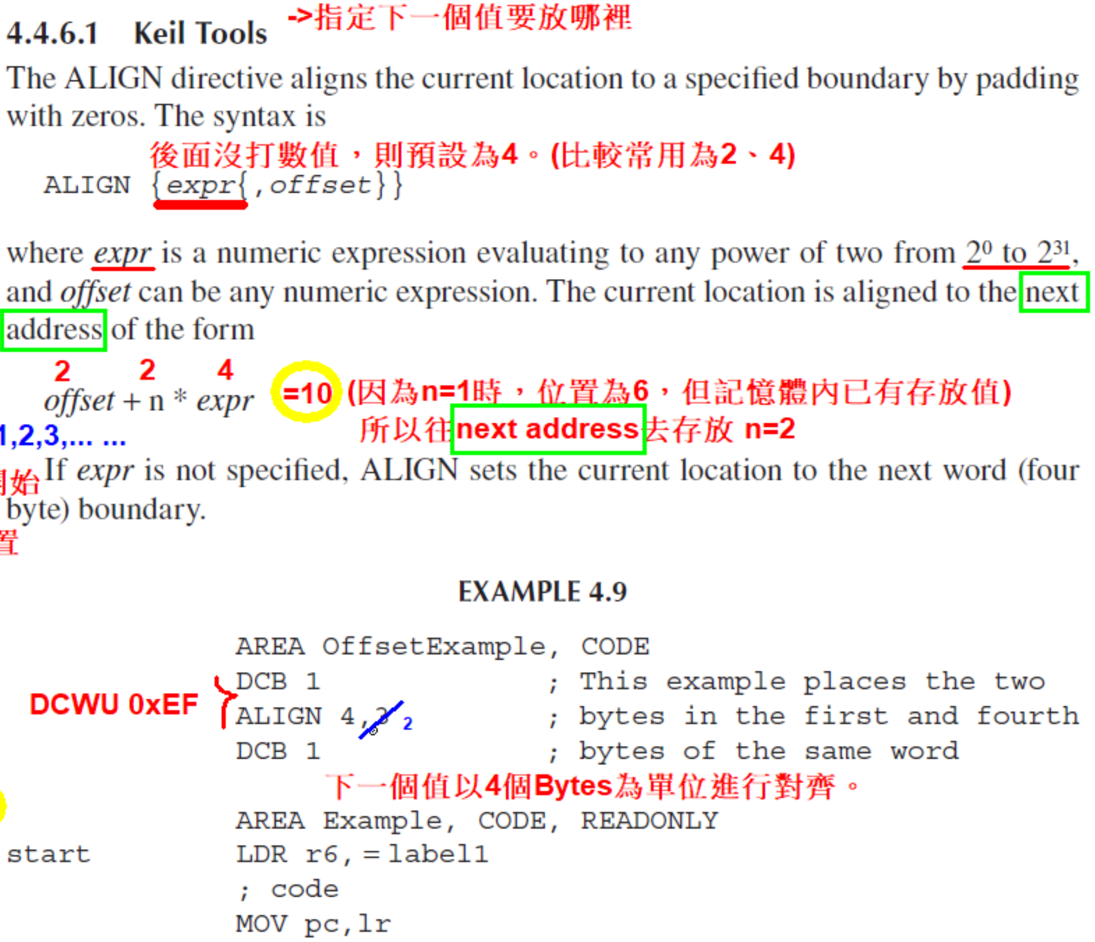
EXAMPLE 4.7

```
coeff  DCW    0xFE37, 0x8ECC   ; defines 2 halfwords
data1  DCD    1,5,20           ; defines 3 words containing
                               ; decimal values 1, 5, and 20
data2  DCD    mem06 + 4        ; defines 1 word containing 4 +
                               ; the address of the label mem06
mem06 DCD 'X'
       AREA   MyData, DATA, READWRITE
       DCB    255              ; now misaligned...
data3  DCDU   1,5,20           ; defines 3 words containing
                               ; 1, 5, and 20 not word aligned
```

0
4
16
20
1
0
3
2
4
8
12
=24
label

宣告一個Data區塊(可省略)

p.70 ALIGN directive



4.4.6.1 Keil Tools

->指定下一個值要放哪裡

The ALIGN directive aligns the current location to a specified boundary by padding with zeros. The syntax is

後面沒打數值，則預設為4。(比較常用為2、4)

```
ALIGN {expr{,offset}}
```

where *expr* is a numeric expression evaluating to any power of two from 2^0 to 2^{31}, and *offset* can be any numeric expression. The current location is aligned to the next address of the form

$$\overset{2}{offset} + \overset{2}{n} * \overset{4}{expr}$$

=10 (因為n=1時，位置為6，但記憶體內已有存放值)
所以往next address去存放 n=2

n=1,2,3,... ...

offset從0開始

If *expr* is not specified, ALIGN sets the current location to the next word (four byte) boundary.

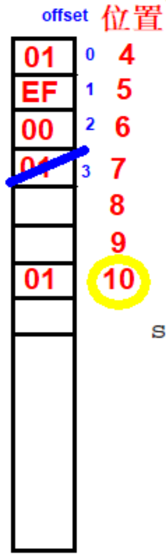
EXAMPLE 4.9

```
              AREA OffsetExample, CODE
              DCB 1                    ; This example places the two
              ALIGN 4,~~3~~ 2          ; bytes in the first and fourth
              DCB 1                    ; bytes of the same word
```

DCWU 0xEF

下一個值以4個Bytes為單位進行對齊。

```
              AREA Example, CODE, READONLY
start         LDR r6, = label1
              ; code
              MOV pc,lr
```



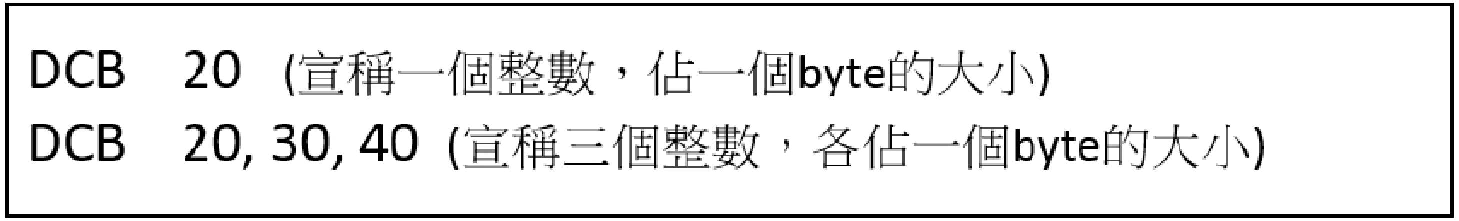
->先指定對齊方式(expr)，然後決定存放位置。

補充說明：DCB

DCB

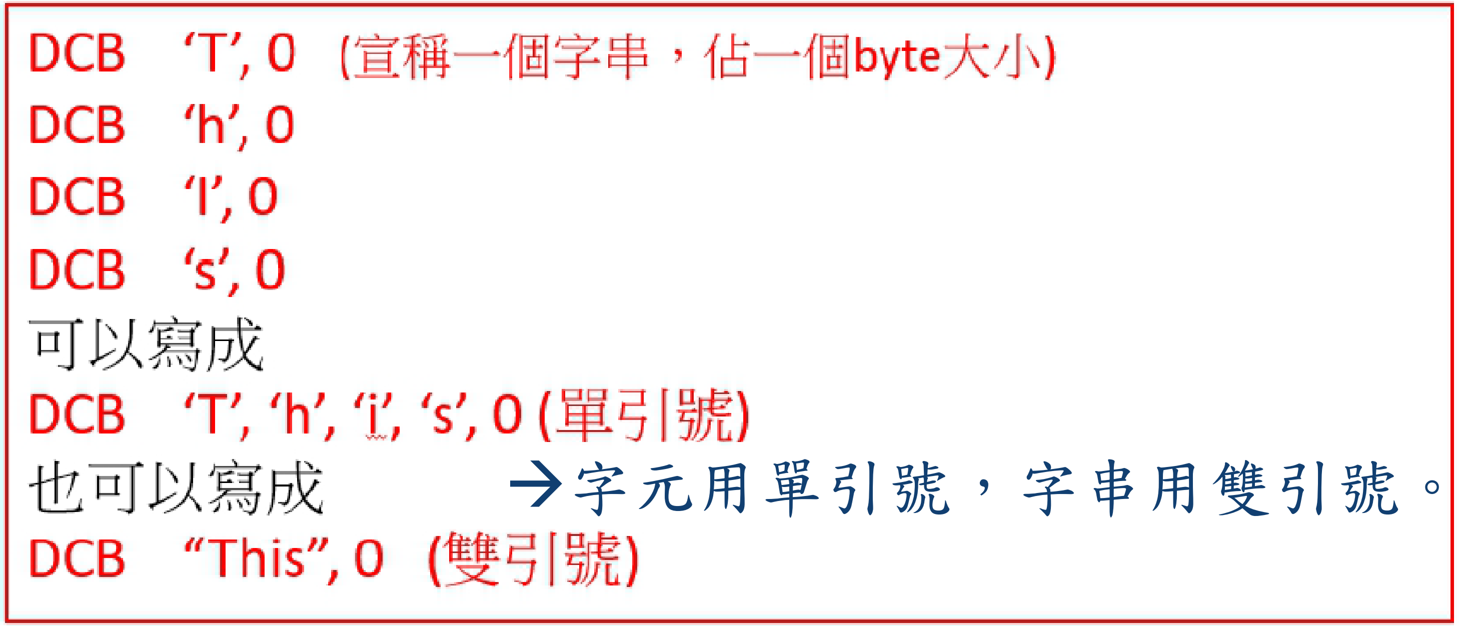
表示大小(Byte)

範例:



DCB　20　(宣稱一個整數，佔一個byte的大小)

DCB　20, 30, 40　(宣稱三個整數，各佔一個byte的大小)



DCB　‘T’, 0　(宣稱一個字串，佔一個byte大小)

DCB　‘h’, 0

DCB　‘I’, 0

DCB　‘s’, 0

可以寫成

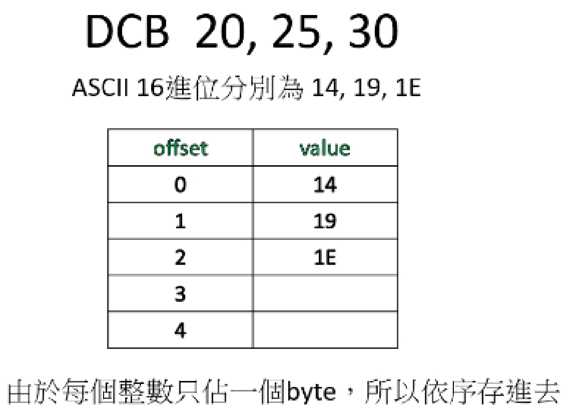
DCB　‘T’, ‘h’, ‘i’, ‘s’, 0 (單引號)

也可以寫成

→字元用單引號，字串用雙引號。

DCB　“This”, 0　(雙引號)

DCB、DCW、DCD圖表說明(1/4)

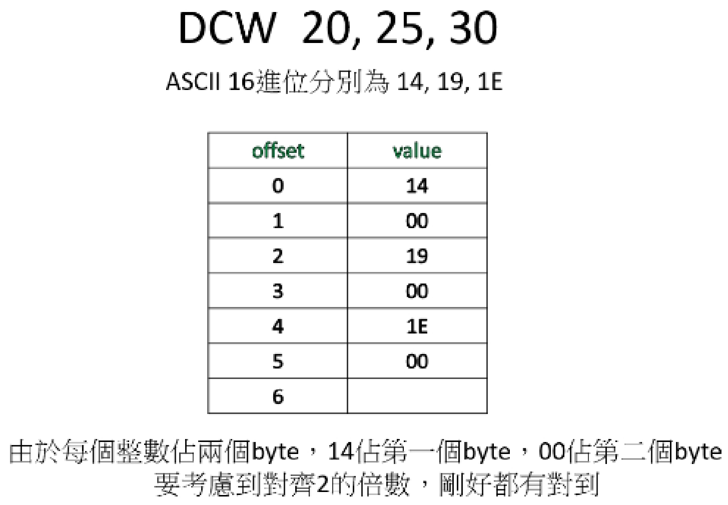


DCB 20, 25, 30

ASCII 16進位分別為 14, 19, 1E

offset	value
0	14
1	19
2	1E
3	
4	

由於每個整數只佔一個byte，所以依序存進去



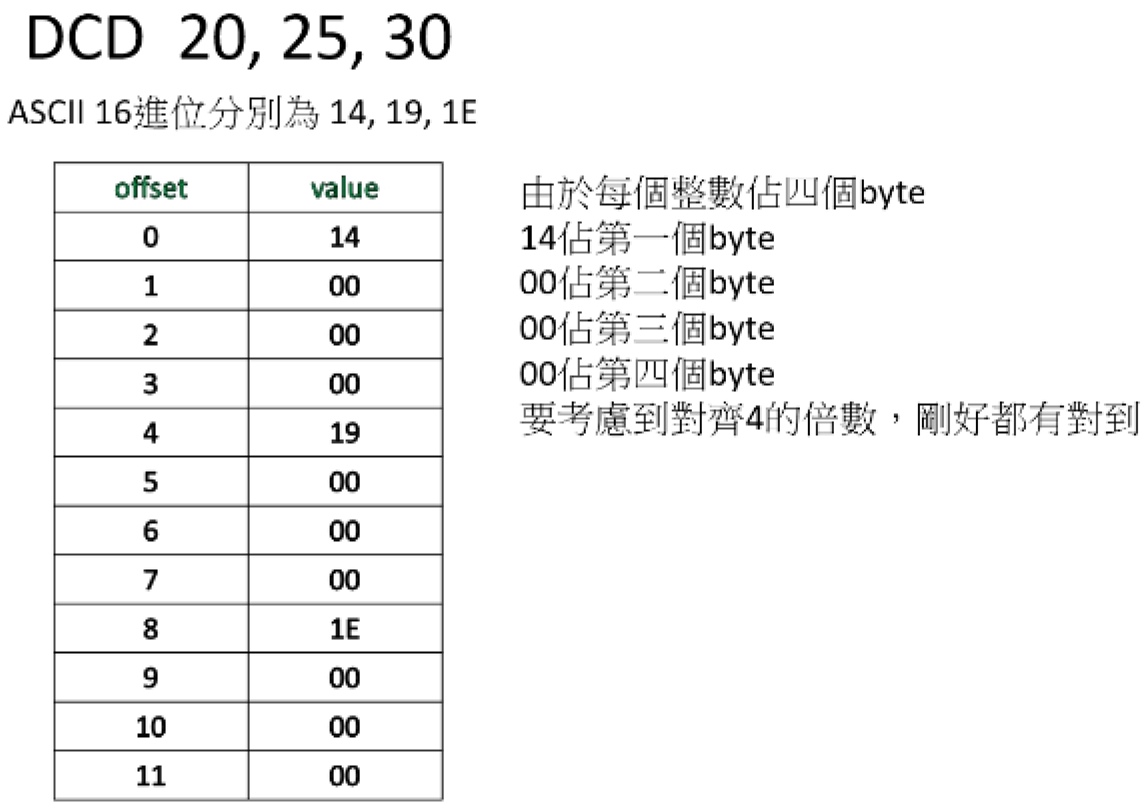
DCW 20, 25, 30

ASCII 16進位分別為 14, 19, 1E

offset	value
0	14
1	00
2	19
3	00
4	1E
5	00
6	

由於每個整數佔兩個byte，14佔第一個byte，00佔第二個byte
要考慮到對齊2的倍數，剛好都有對到

DCB、DCW、DCD圖表說明(2/4)



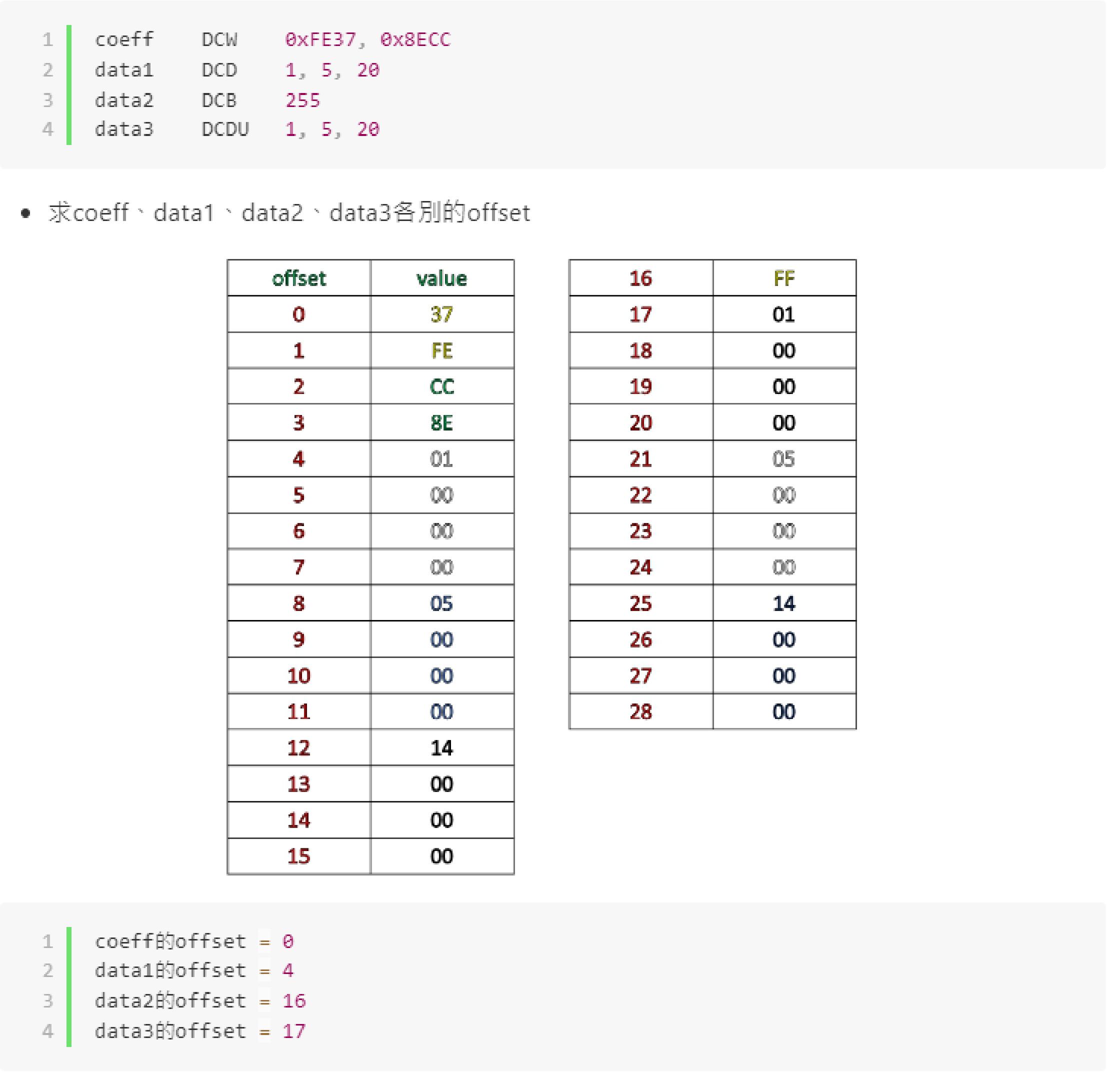
DCD 20, 25, 30

ASCII 16進位分別為 14, 19, 1E

offset	value
0	14
1	00
2	00
3	00
4	19
5	00
6	00
7	00
8	1E
9	00
10	00
11	00

由於每個整數佔四個byte
14佔第一個byte
00佔第二個byte
00佔第三個byte
00佔第四個byte
要考慮到對齊4的倍數，剛好都有對到

DCB、DCW、DCD圖表說明(3/4)



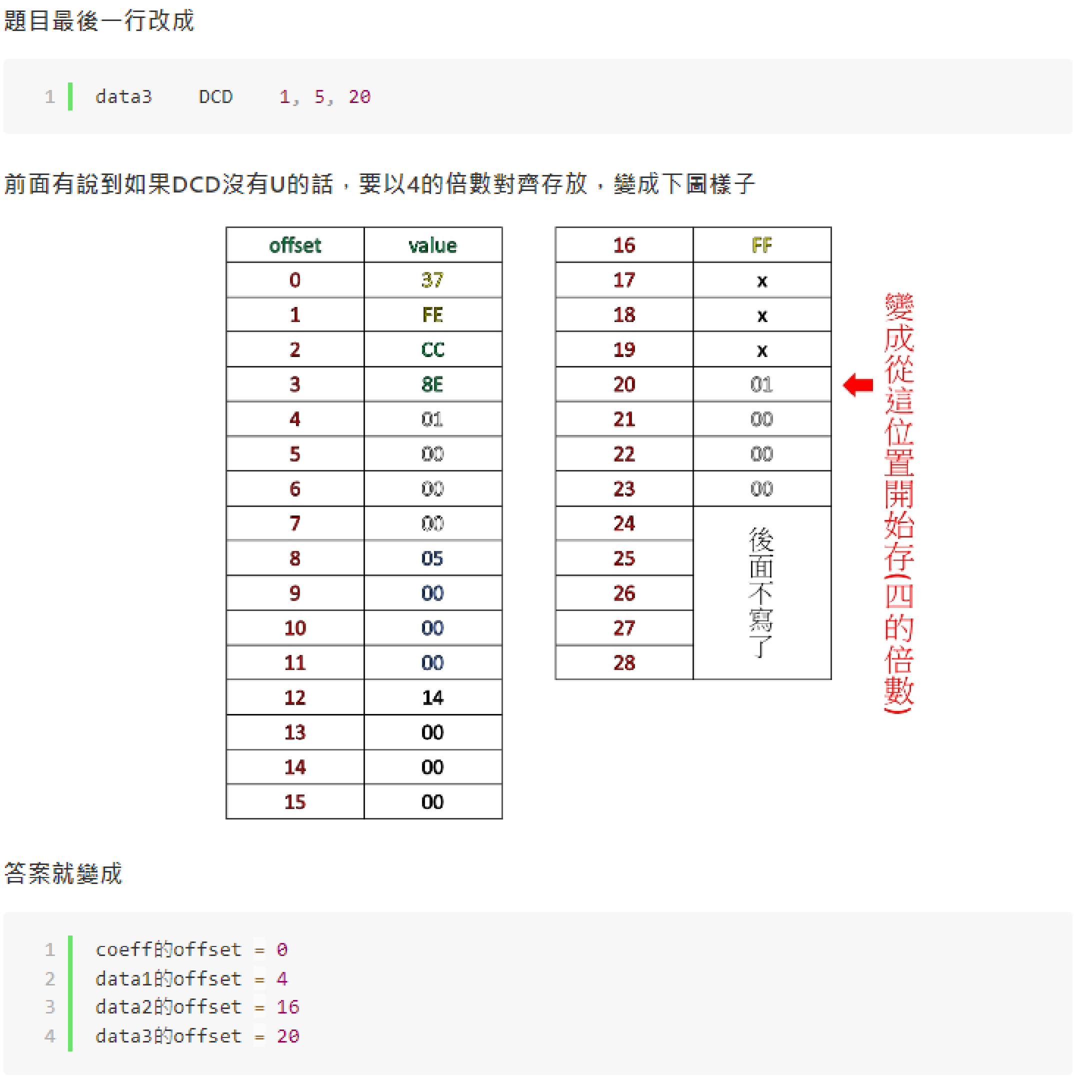
```
coeff     DCW    0xFE37, 0x8ECC
data1     DCD    1, 5, 20
data2     DCB    255
data3     DCDU   1, 5, 20
```

- 求coeff、data1、data2、data3各別的offset

offset	value
0	37
1	FE
2	CC
3	8E
4	01
5	00
6	00
7	00
8	05
9	00
10	00
11	00
12	14
13	00
14	00
15	00
16	FF
17	01
18	00
19	00
20	00
21	05
22	00
23	00
24	00
25	14
26	00
27	00
28	00

```
coeff的offset = 0
data1的offset = 4
data2的offset = 16
data3的offset = 17
```

DCB、DCW、DCD圖表說明(4/4)



題目最後一行改成

```
data3    DCD    1, 5, 20
```

前面有說到如果DCD沒有U的話，要以4的倍數對齊存放，變成下圖樣子

offset	value
0	37
1	FE
2	CC
3	8E
4	01
5	00
6	00
7	00
8	05
9	00
10	00
11	00
12	14
13	00
14	00
15	00

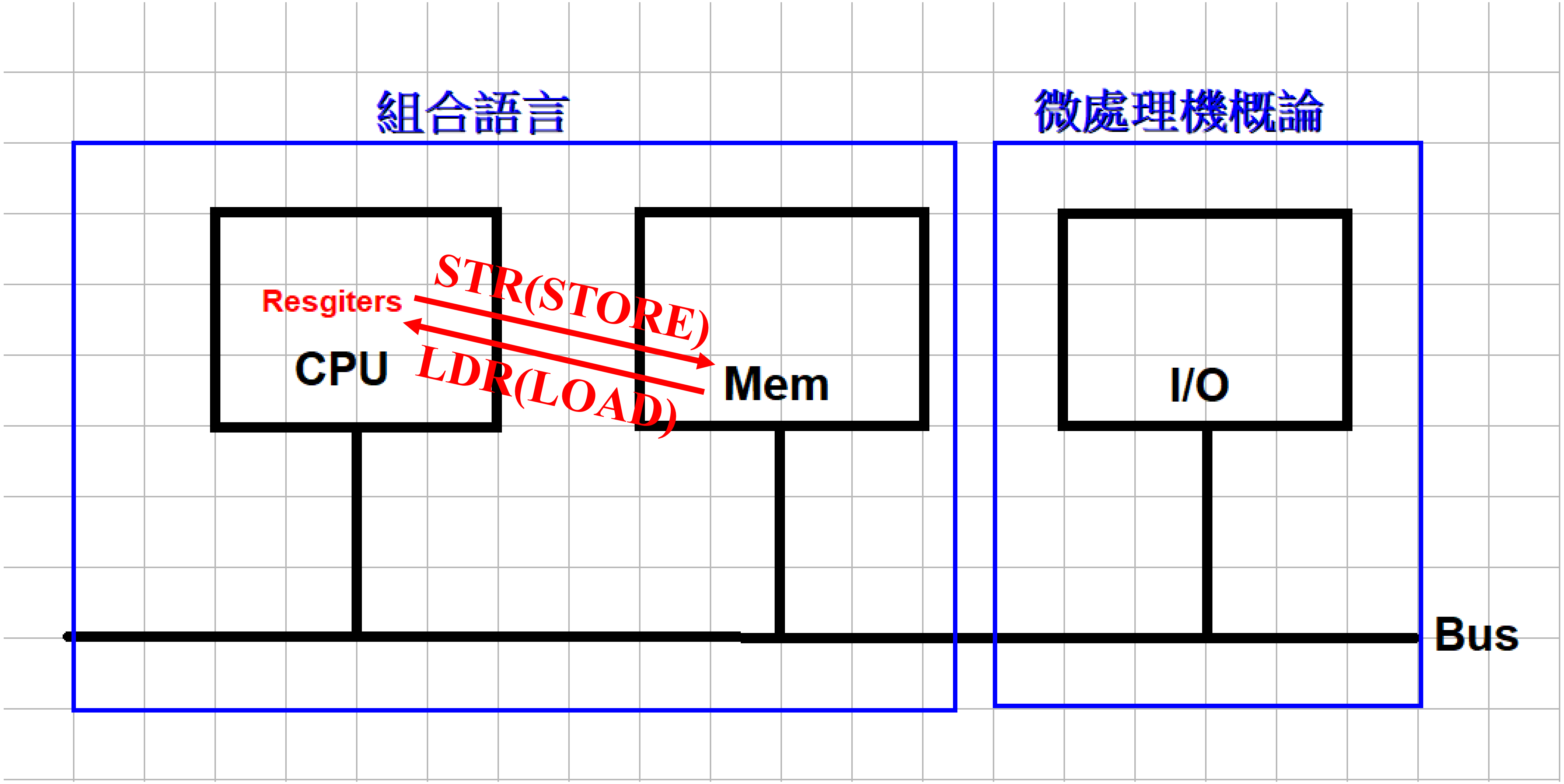
16	FF
17	x
18	x
19	x
20	01
21	00
22	00
23	00
24	後面不寫了
25	
26	
27	
28	

變成從這位置開始存(四的倍數)

答案就變成

```
coeff的offset = 0
data1的offset = 4
data2的offset = 16
data3的offset = 20
```

p.84 Loads, Stores, and Addressing



p.84 Loads, Stores, and Addressing

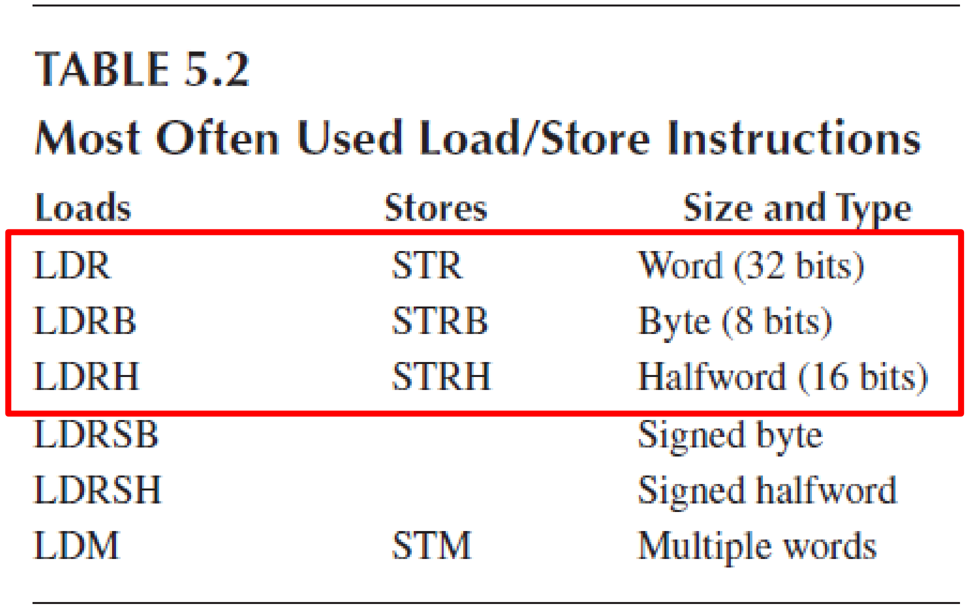


TABLE 5.2
Most Often Used Load/Store Instructions

Loads	Stores	Size and Type
LDR	STR	Word (32 bits)
LDRB	STRB	Byte (8 bits)
LDRH	STRH	Halfword (16 bits)
LDRSB		Signed byte
LDRSH		Signed halfword
LDM	STM	Multiple words

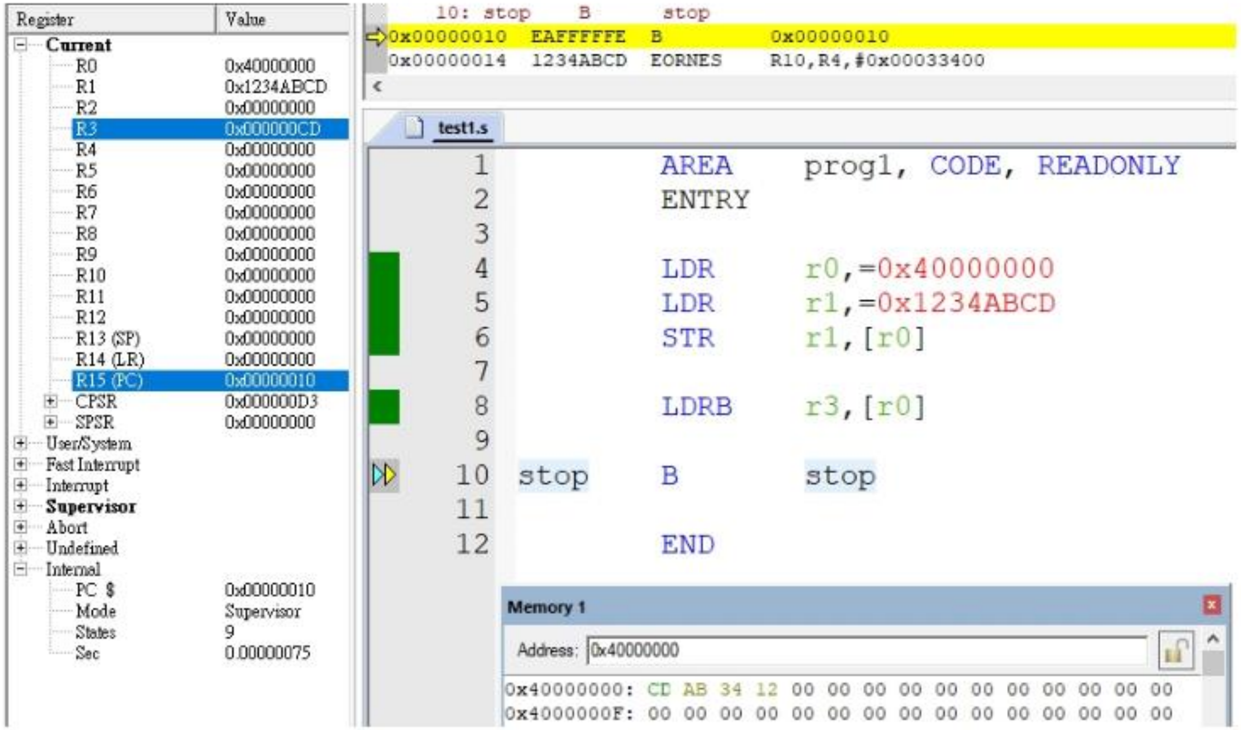
LDRB
都跟LDR同理，不過copy的值的size為一個Byte(8bits)。

LDRH
copy的值的size為一個Halfword(16bits)。

LDR|STR{<size>}{<cond>}   <Rd>, <addressing_mode>

LDRB指令範例

LDRB



Register	Value
Current	
R0	0x40000000
R1	0x1234ABCD
R2	0x00000000
R3	0x000000CD
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000010
CPSR	0x000000D3
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	
Supervisor	
Abort	
Undefined	
Internal	
PC $	0x00000010
Mode	Supervisor
States	9
Sec	0.00000075

```
    10: stop     B        stop
0x00000010  EAFFFFFE  B         0x00000010
0x00000014  1234ABCD  EORNES    R10,R4,#0x00033400
```

test1.s

```
            AREA    prog1, CODE, READONLY
            ENTRY

            LDR     r0,=0x40000000
            LDR     r1,=0x1234ABCD
            STR     r1,[r0]

            LDRB    r3,[r0]

stop        B       stop

            END
```

Memory 1

Address: 0x40000000

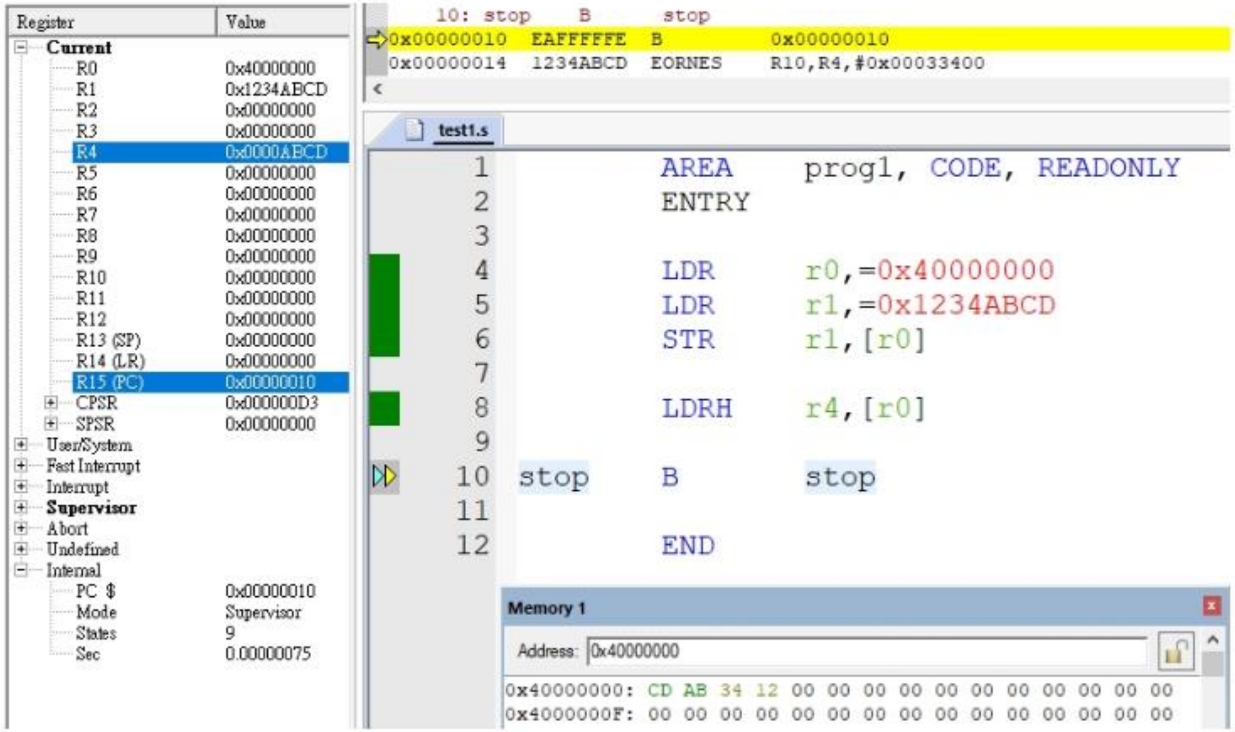
```
0x40000000: CD AB 34 12 00 00 00 00 00 00 00 00 00 00 00
0x4000000F: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

LDRB程式說明

```
LDR     r0,=0x40000000       ;將0x40000000的值存入r0
LDR     r1,=0x1234ABCD       ;將0x1234ABCD的值存入r1
STR     r1,[r0]              ;將r1的值存入r0記憶體中
LDRB    r3,[r0]              ;將r0記憶體位置內的值copy一個byte的size到r3
```

LDRH指令範例

LDRH



Register	Value
Current	
R0	0x40000000
R1	0x1234ABCD
R2	0x00000000
R3	0x00000000
R4	0x0000ABCD
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000010
CPSR	0x000000D3
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	
Supervisor	
Abort	
Undefined	
Internal	
PC $	0x00000010
Mode	Supervisor
States	9
Sec	0.00000075

```
    10: stop    B       stop
0x00000010  EAFFFFFE  B         0x00000010
0x00000014  1234ABCD  EORNES    R10,R4,#0x00033400
```

test1.s

```
          AREA    prog1, CODE, READONLY
          ENTRY

          LDR     r0,=0x40000000
          LDR     r1,=0x1234ABCD
          STR     r1,[r0]

          LDRH    r4,[r0]

stop      B       stop

          END
```

Memory 1

Address: 0x40000000

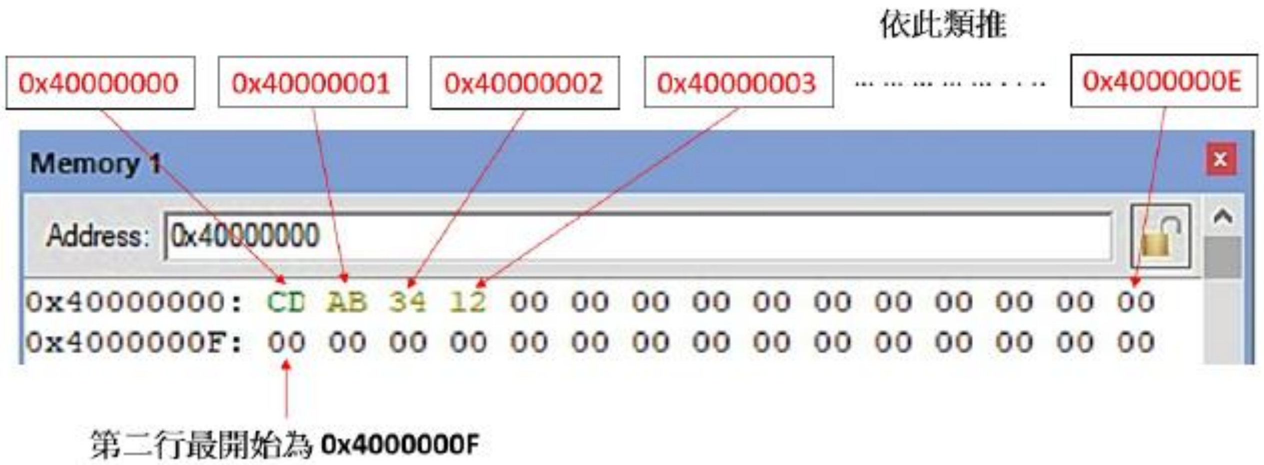
```
0x40000000: CD AB 34 12 00 00 00 00 00 00 00 00 00 00 00 00
0x4000000F: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

LDRH程式說明

```
LDR     r0,=0x40000000      ;將0x40000000的值存入r0
LDR     r1,=0x1234ABCD      ;將0x1234ABCD的值存入r1
STR     r1,[r0]             ;將r1的值存入r0記憶體中
LDRH    r4,[r0]             ;將r0記憶體位置內的值copy一個Halfwodr的size到r4
```

p.85 EXAMPLE 5.1

- STR值到記憶體後的狀態，不知道這樣會不會更好理解



依此類推

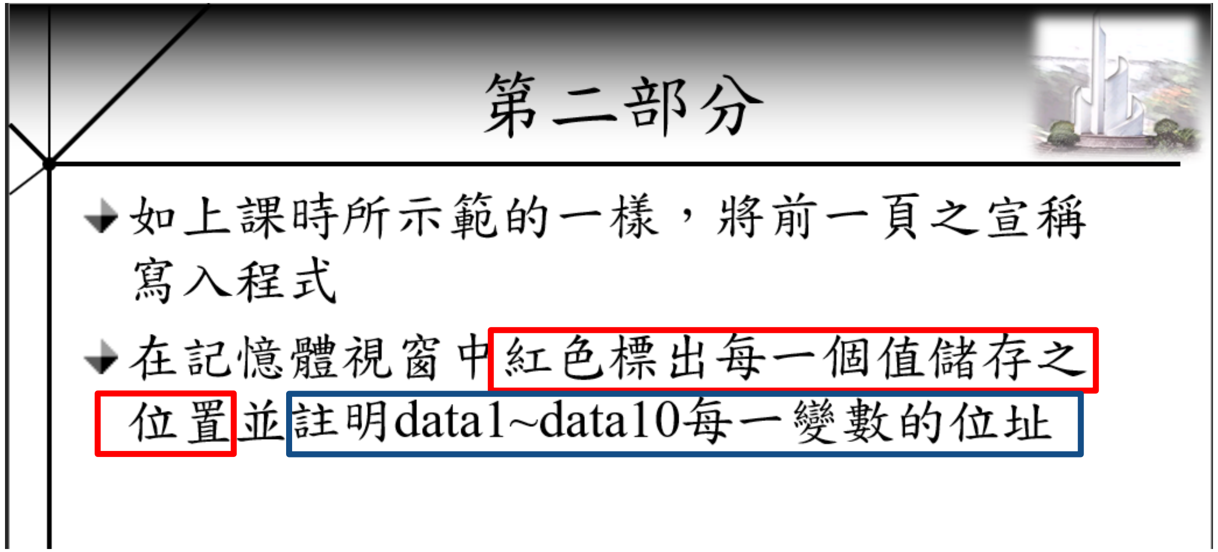
0x40000000 0x40000001 0x40000002 0x40000003 ………………….. 0x4000000E

Memory 1

Address: 0x40000000

```
0x40000000: CD AB 34 12 00 00 00 00 00 00 00 00 00 00 00
0x4000000F: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

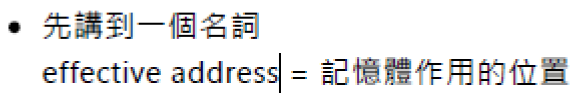
第二行最開始為 0x4000000F



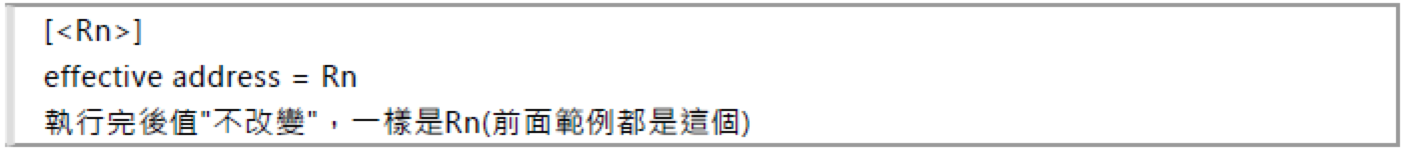
第二部分

- 如上課時所示範的一樣，將前一頁之宣稱寫入程式
- 在記憶體視窗中紅色標出每一個值儲存之位置並註明data1~data10每一變數的位址

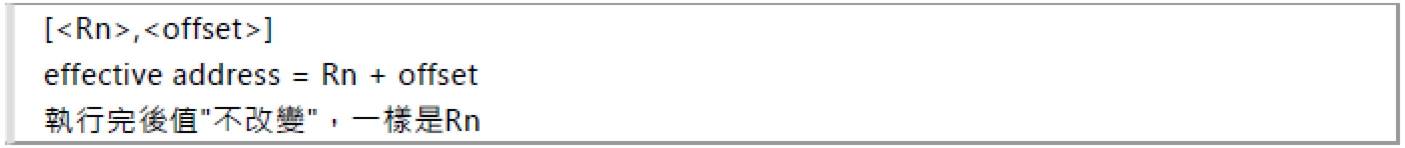
p.88 Effective Address(EA)



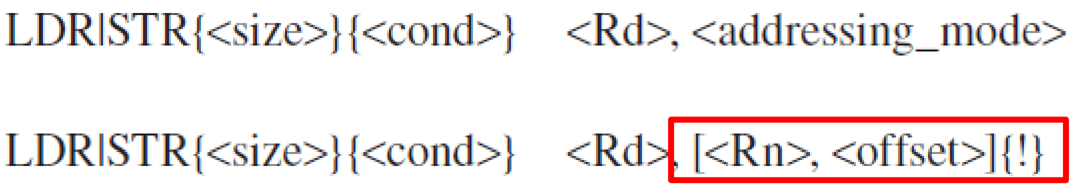
- 先講到一個名詞
  effective address = 記憶體作用的位置



[<Rn>]
effective address = Rn
執行完後值"不改變"，一樣是Rn(前面範例都是這個)



[<Rn>,<offset>]
effective address = Rn + offset
執行完後值"不改變"，一樣是Rn



LDR|STR{<size>}{<cond>}   <Rd>, <addressing_mode>

LDR|STR{<size>}{<cond>}   <Rd>, [<Rn>, <offset>]{!}

EX.

LDR	R0, [R1]	EA = R1
LDR	R0, [R1, #4]	EA = R1 + 4
LDR	R0, [R1, R2]	EA = R1 + R2

第2次隨堂考

2023 Advanced Mixed-Operation System (AMOS) Lab.

評分標準

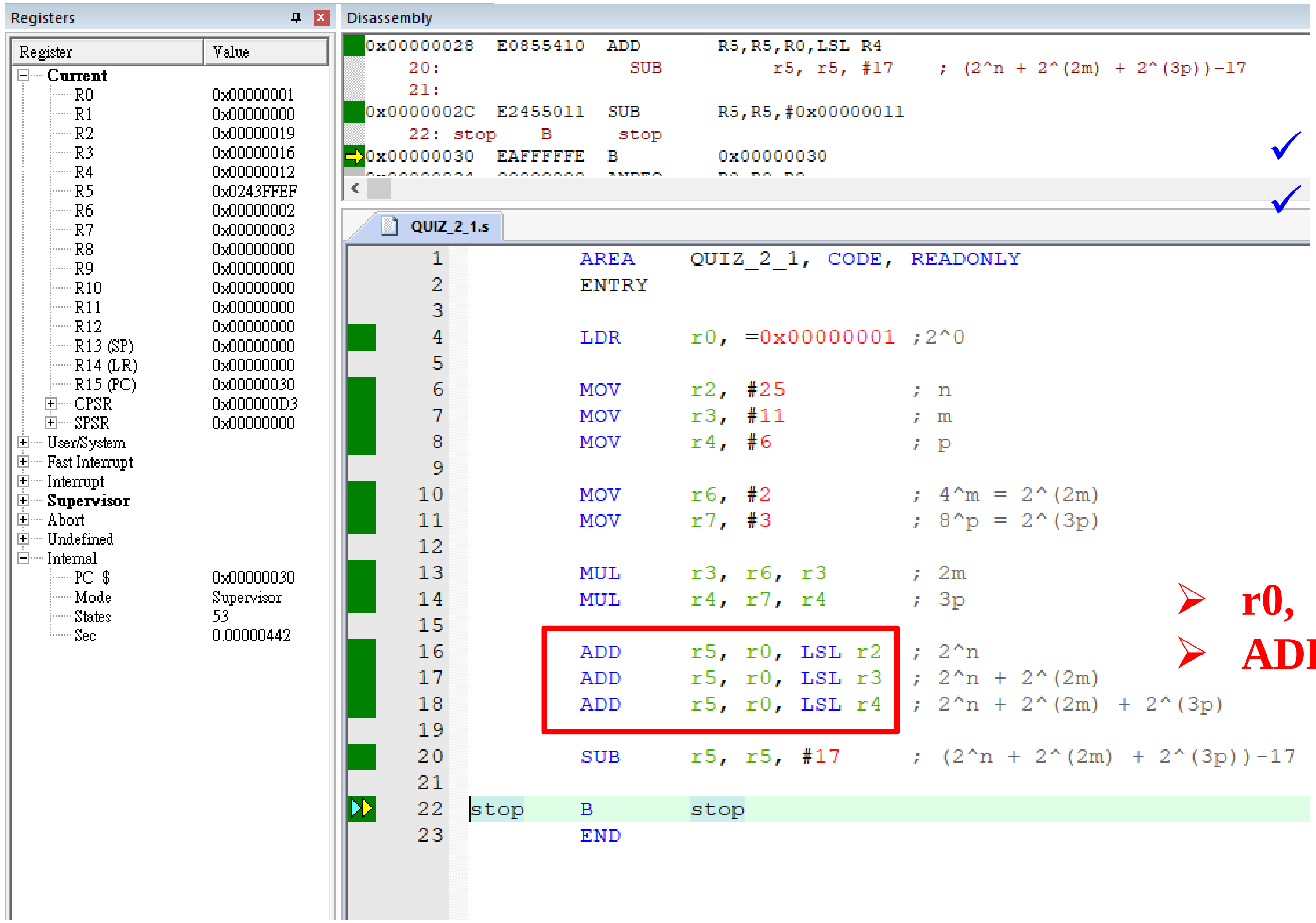
→一共有2題，一題50分。

0	未繳交、交白券、程式碼裡沒學號姓名
10	基本分(只交程式碼，沒進入Debugger介面)
20	有進入Debugger介面，程式碼與題目要求的差很多
30	程式碼有小錯誤，導致輸出結果數值不正確
40	最終結果暫存器與題目不一樣、沒寫出2's Complement程式碼
50	完全正確

第2次隨堂考-第1題

1. Compute j = ($2^n+4^m+8^p$) – 17 and put j in r5 assuming n, m and p are respectively in r2, r3 and r4 initially.

   Assume r2 = 25, r3 = 11 and r4 = 6 initially.



Registers

Register	Value
Current	
R0	0x00000001
R1	0x00000000
R2	0x00000019
R3	0x00000016
R4	0x00000012
R5	0x0243FFEF
R6	0x00000002
R7	0x00000003
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000030
CPSR	0x000000D3
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	
Supervisor	
Abort	
Undefined	
Internal	
PC $	0x00000030
Mode	Supervisor
States	53
Sec	0.00000442

Disassembly

```
0x00000028  E0855410  ADD       R5,R5,R0,LSL R4
    20:                  SUB          r5, r5, #17     ; (2^n + 2^(2m) + 2^(3p))-17
    21:
0x0000002C  E2455011  SUB       R5,R5,#0x00000011
    22: stop    B        stop
0x00000030  EAFFFFFE  B         0x00000030
```

QUIZ_2_1.s

```
        AREA    QUIZ_2_1, CODE, READONLY
        ENTRY

        LDR     r0, =0x00000001 ;2^0

        MOV     r2, #25         ; n
        MOV     r3, #11         ; m
        MOV     r4, #6          ; p

        MOV     r6, #2          ; 4^m = 2^(2m)
        MOV     r7, #3          ; 8^p = 2^(3p)

        MUL     r3, r6, r3      ; 2m
        MUL     r4, r7, r4      ; 3p

        ADD     r5, r0, LSL r2  ; 2^n
        ADD     r5, r0, LSL r3  ; 2^n + 2^(2m)
        ADD     r5, r0, LSL r4  ; 2^n + 2^(2m) + 2^(3p)

        SUB     r5, r5, #17     ; (2^n + 2^(2m) + 2^(3p))-17

stop    B       stop
        END
```

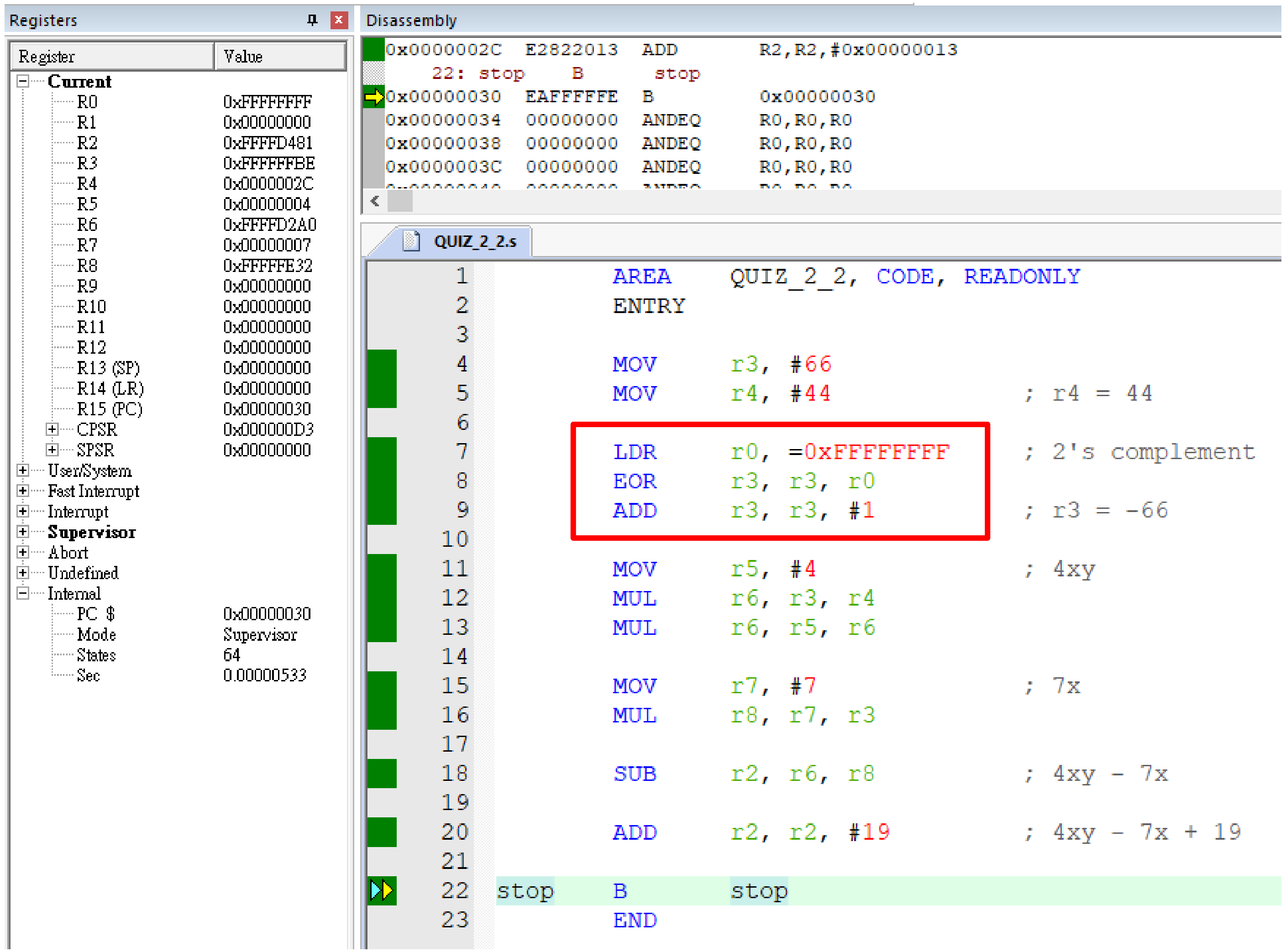
✓ **STEP 1. 了解二進制 0 bit~ 31 bit 代表2^0~ 2^{31}。**

✓ **STEP 2. 了解LSL是以bit為單位進行移動。**

- **r0, LSL r2 → r0左移r2值，運算完後不改變r0值。**
- **ADD r5, r5, r0 → ADD r5, r0**

第2次隨堂考-第2題

2. Write a program to get the result of 4xy - 7x + 19 (Assuming r3 = x = -66 and r4 = y = 44 and r2 =result).(須以程式碼來運算 2's complement)



Registers

Register	Value
Current	
R0	0xFFFFFFFF
R1	0x00000000
R2	0xFFFFD481
R3	0xFFFFFFBE
R4	0x0000002C
R5	0x00000004
R6	0xFFFFD2A0
R7	0x00000007
R8	0xFFFFFE32
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000030
CPSR	0x000000D3
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	
Supervisor	
Abort	
Undefined	
Internal	
PC $	0x00000030
Mode	Supervisor
States	64
Sec	0.00000533

Disassembly

```
0x0000002C  E2822013  ADD       R2,R2,#0x00000013
    22: stop     B       stop
0x00000030  EAFFFFFE  B         0x00000030
0x00000034  00000000  ANDEQ     R0,R0,R0
0x00000038  00000000  ANDEQ     R0,R0,R0
0x0000003C  00000000  ANDEQ     R0,R0,R0
```

QUIZ_2_2.s

```
        AREA    QUIZ_2_2, CODE, READONLY
        ENTRY

        MOV     r3, #66
        MOV     r4, #44                 ; r4 = 44

        LDR     r0, =0xFFFFFFFF         ; 2's complement
        EOR     r3, r3, r0
        ADD     r3, r3, #1              ; r3 = -66

        MOV     r5, #4                  ; 4xy
        MUL     r6, r3, r4
        MUL     r6, r5, r6

        MOV     r7, #7                  ; 7x
        MUL     r8, r7, r3

        SUB     r2, r6, r8              ; 4xy - 7x

        ADD     r2, r2, #19             ; 4xy - 7x + 19

stop    B       stop
        END
```

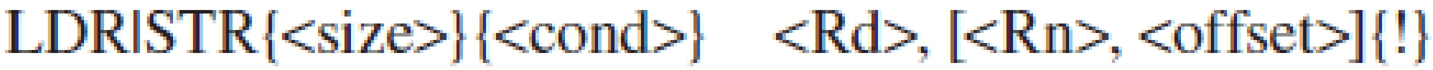
✓ **STEP 1.了解EOR是bit to bit指令。**

✓ **STEP 2.任何數與1 XOR會反向。**

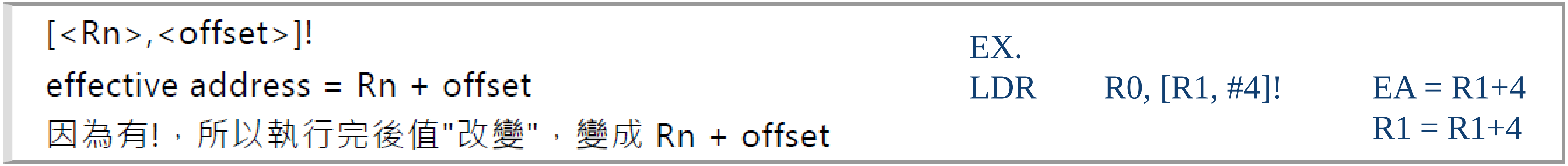
1024正課複習

2023 Advanced Mixed-Operation System (AMOS) Lab.

p.88 p.89 註記記憶體位置的語法



LDR|STR{<size>}{<cond>}   <Rd>, [<Rn>, <offset>]{!}



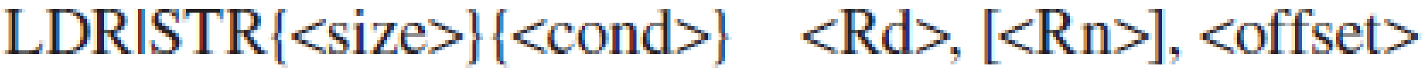
[<Rn>,<offset>]!

effective address = Rn + offset

因為有!，所以執行完後值"改變"，變成 Rn + offset

EX.

LDR	R0, [R1, #4]!	EA = R1+4
		R1 = R1+4



LDR|STR{<size>}{<cond>}   <Rd>, [<Rn>], <offset>

[<Rn>], <offset>

Effective Address = Rn

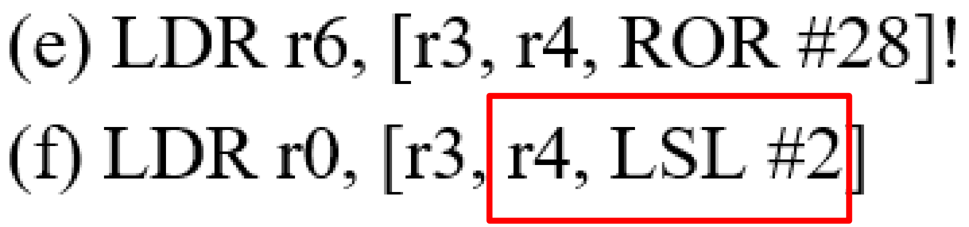
執行完後Rn變為Rn + offset

EX.

LDR	R0, [R1], #4	EA = R1
		R1 = R1+4

LDR	R0, [R1], R2	EA = R1
		R1 = R1+R2

作業二 第三部分



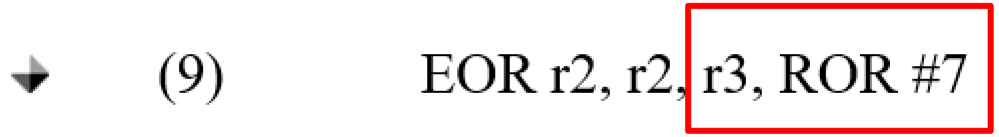
(e) LDR r6, [r3, r4, ROR #28]!

(f) LDR r0, [r3, $\boxed{\text{r4, LSL \#2}}$]

LSL r4, r4, #2

→但這裡的r4不會改變

EA = base + offset(number
register
instruction)



(9) EOR r2, r2, $\boxed{\text{r3, ROR \#7}}$

ROR r3, r3, #7

→但這裡的r3不會改變

Destination, Source1, Source2

→Source2也可以是**instruction**

→運算完後不改變Source2內暫存器的值

p.126 p.127 BIC指令

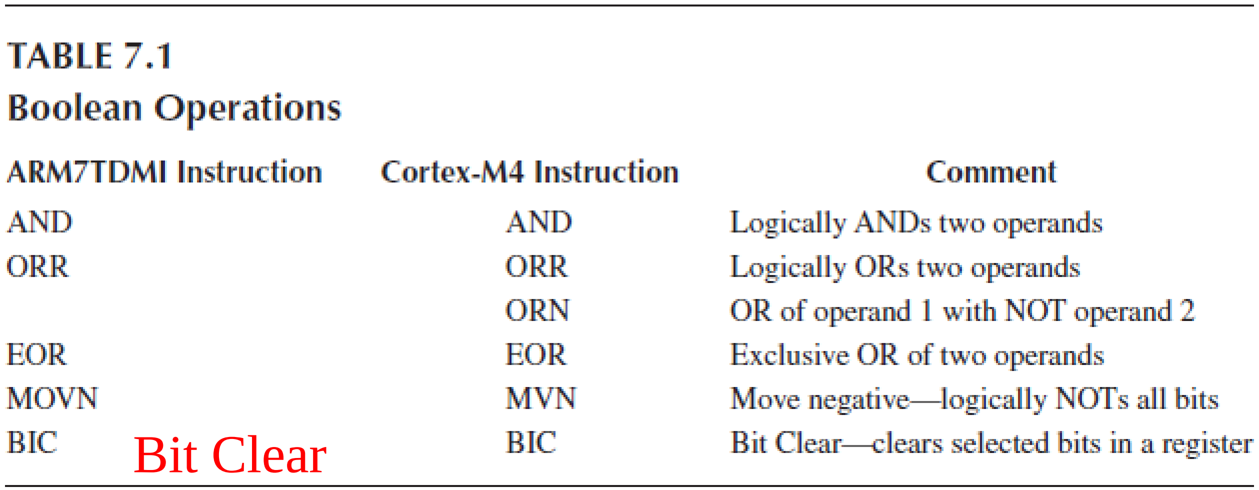


TABLE 7.1
Boolean Operations

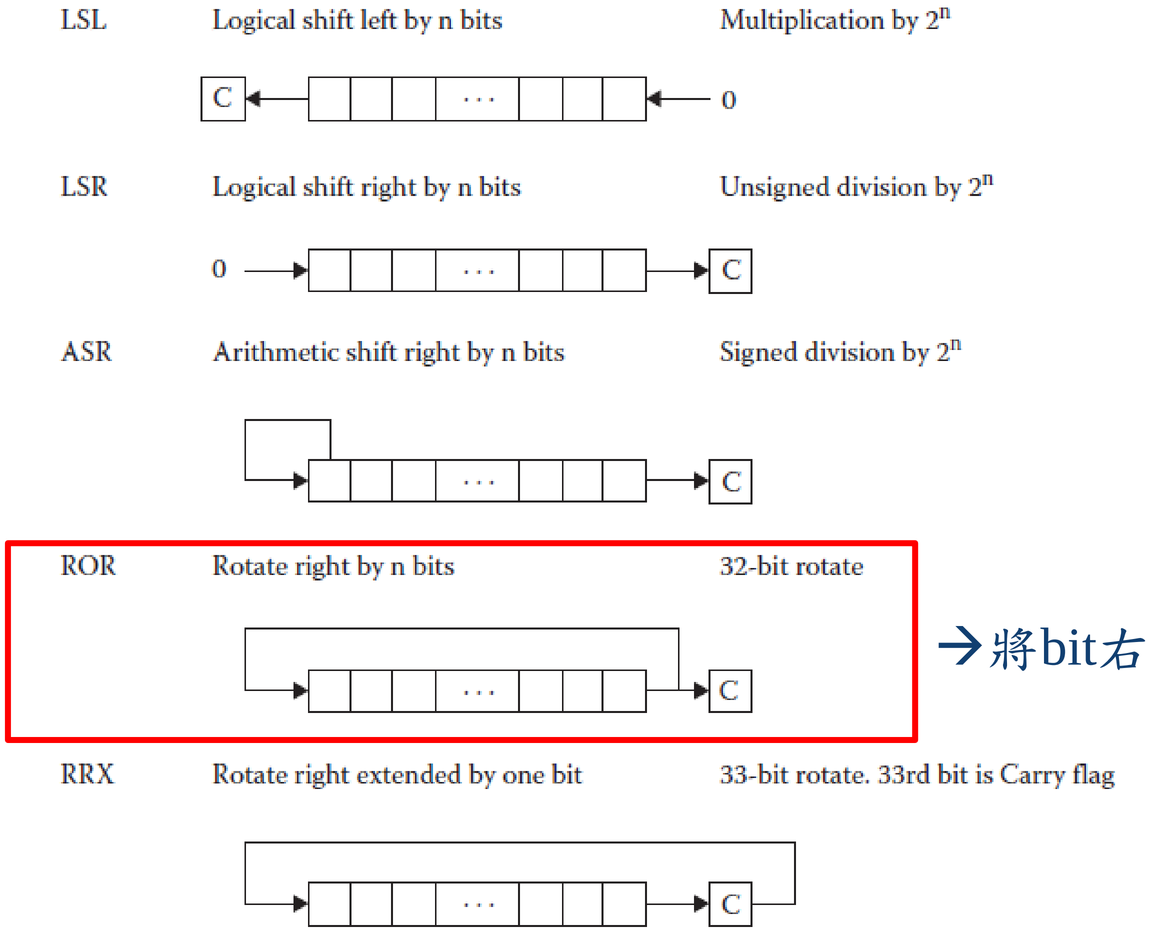
ARM7TDMI Instruction	Cortex-M4 Instruction	Comment
AND	AND	Logically ANDs two operands
ORR	ORR	Logically ORs two operands
	ORN	OR of operand 1 with NOT operand 2
EOR	EOR	Exclusive OR of two operands
MOVN	MVN	Move negative—logically NOTs all bits
BIC	BIC	Bit Clear—clears selected bits in a register

Bit Clear

```
BIC r2, r3, #0xFF000000
```

→用1指定哪些bits要清為0

p.128 p129. ROR指令

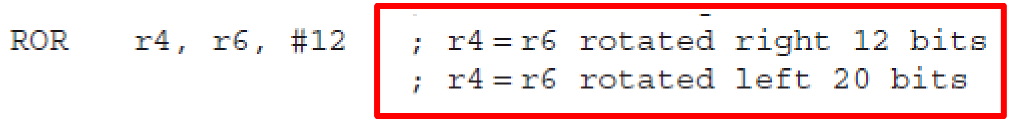


LSL	Logical shift left by n bits	Multiplication by 2^n
LSR	Logical shift right by n bits	Unsigned division by 2^n
ASR	Arithmetic shift right by n bits	Signed division by 2^n
ROR	Rotate right by n bits	32-bit rotate
RRX	Rotate right extended by one bit	33-bit rotate. 33rd bit is Carry flag

FIGURE 7.5 Shifts and rotates.

→將bit右移後，多出來的bit依序存回高位元

r6 右移12bits = r6左移20bits



```
ROR    r4, r6, #12   ; r4 = r6 rotated right 12 bits
                     ; r4 = r6 rotated left 20 bits
```

作業二 第三部分

AND	0	1
0	0	0
1	0	1

→any bit AND 0 = 0 →**clear**

→any bit AND 1 = any bit →**unchanged**

ORR	0	1
0	0	1
1	1	1

→any bit ORR 0 = any bit →**unchanged**

→any bit ORR 1 = 1→**set**

EOR	0	1
0	0	1
1	1	0

→any bit EOR 0 = any bit →**unchanged**

→any bit EOR 1 = invert the bit

Q&A

Thanks for your attention !!