第12次組語實習課

學生：林培瑋

2023 Advanced Mixed-Operation System (AMOS) Lab.

Tamkang University
Department of Electrical and Computer Engineering
No.151, Yingzhuan Rd., Tamsui Dist., New Taipei City 25137, Taiwan (R.O.C.)

目錄

- 1212正課複習.....3
- 第4次隨堂考.....10
- 1219正課複習.....16

1212正課複習

2023 Advanced Mixed-Operation System (AMOS) Lab.

Tamkang University
Department of Electrical and Computer Engineering
No.151, Yingzhuan Rd., Tamsui Dist., New Taipei City 25137, Taiwan (R.O.C.)

p.109 Subroutine Example

```
                AREA Example, CODE
                ENTRY                       ; mark first instruction
                BL      func1               ; call first subroutine
                BL      func2               ; call second subroutine
stop            B       stop                ; terminate the program
func1           LDR     r0, =42             ; => MOV r0, #42
                LDR     r1, =0x12345678     ; => LDR r1, [PC, #N]
                                            ; where N=offset to literal pool 1
                LDR     r2, =0xFFFFFFFF     ; => MVN r2, #0
                BX      lr                  ; return from subroutine
                LTORG                       ; literal pool 1 has 0x12345678
func2           LDR     r3, =0x12345678     ; => LDR r3, [PC, #N]-12
                                            ; N=offset back to literal pool 1
                ;LDR    r4, =0x87654321     ; if this is uncommented, it fails.
                                            ; Literal pool 2 is out of reach!
                BX      lr                  ; return from subroutine
                BigTable
                SPACE 4200                  ; clears 4200 bytes of memory,
                                            ; starting here
                END                         ; literal pool 2 empty
```

(一共有32 bits可以使用)

4

N的最大值為12-bit(4096)，因為要預留指令、暫存器編碼後放置的空間。

copy 1's complement of 0 to r2

全名：Move Negative

(此處代表1's complement)

全名：Literal Pool Origin

(不會放重複的數值進Literal Pool)

在此處加入LTORG可解決

在此處新增4200 bytes記憶體空間。

或是把SPACE 4200註解掉，
或是把4200數值改小。

0x87654321

若沒有加註解，會編譯失敗，
因為距離太遠。(N>4096)

```
warning: A1581W: Added 2 bytes of padding at address 0x56
```

```
Literal pool too distant, use LTORG to assemble it within 4KB
```

p.109 Subroutine Example – side effect(副作用)

- side effect：使用者無法得知副函式是否有修改到暫存器資料。

假設r0 = 21 ；r1 = 45 ；r2 = 54

```
                AREA Example, CODE
                ENTRY                   ; mark first instruction
                BL      func1           ; call first subroutine
                BL      func2           ; call second subroutine
stop            B       stop            ; terminate the program
func1           LDR     r0, =42         ; => MOV r0, #42
                LDR     r1, =0x12345678 ; => LDR r1, [PC, #N]
                                        ; where N=offset to literal pool 1
                LDR     r2, =0xFFFFFFFF ; => MVN r2, #0
                BX      lr              ; return from subroutine
                LTORG                   ; literal pool 1 has 0x12345678
func2           LDR     r3, =0x12345678 ; => LDR r3, [PC, #N]
                                        ; N=offset back to literal pool 1
                ;LDR    r4, =0x87654321 ; if this is uncommented, it fails.
                                        ; Literal pool 2 is out of reach!
                BX      lr              ; return from subroutine
BigTable
                SPACE 4200              ; clears 4200 bytes of memory,
                                        ; starting here
                END                     ; literal pool 2 empty
```

mov r3, r0
mov r4, r1
mov r5, r2
此方法不好，因為還是存在暫存器

store r0, r1, r2 into memory

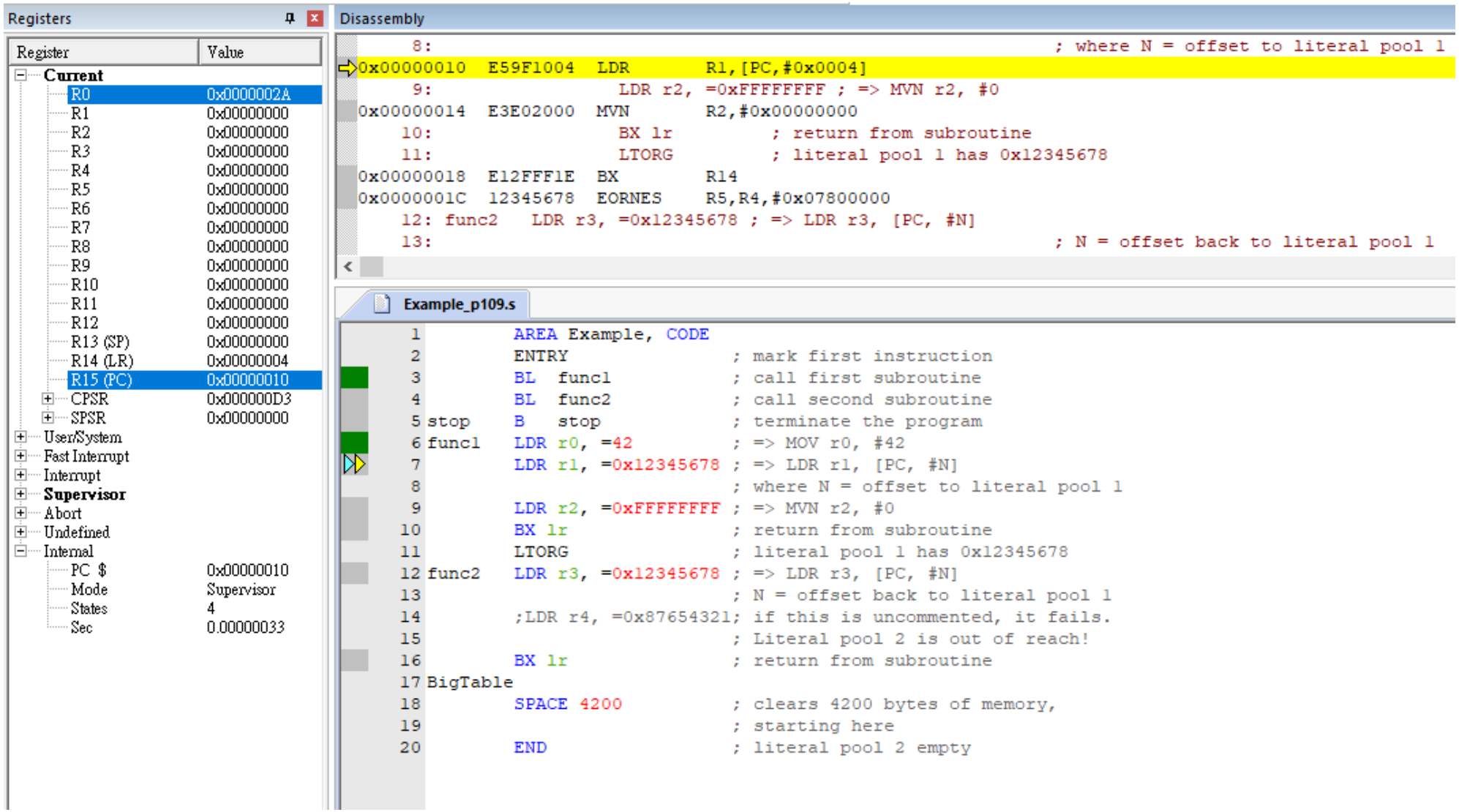
restore r0, r1, r2 into memory

解決方法

STR r0, []
STR r1, []
STR, r2, []
(缺點：太繁瑣)

將暫存器資料先存到記憶體

p.109 Subroutine Example – Keil Tool



p.109 Subroutine Example – Keil Tool

Disassembly

```
0x00000014  E3E02000  MVN       R2,#0x00000000
    10:                 BX lr         ; return from subroutine
    11:                 LTORG         ; literal pool 1 has 0x12345678
0x00000018  E12FFF1E  BX        R14
0x0000001C  12345678  EORNES    R5,R4,#0x07800000
    12: func2   LDR r3, =0x12345678 ; => LDR r3, [PC, #N]
    13:                                                   ; N = offset back to literal pool 1
    14:                 ;LDR r4, =0x87654321; if this is uncommented, it fails.
    15:                                                   ; Literal pool 2 is out of reach!
0x00000020  E51F300C  LDR       R3,[PC,#-0x000C]
    16:                 BX lr         ; return from subroutine
0x00000024  E12FFF1E  BX        R14
0x00000028  00000000  ANDEQ     R0,R0,R0
0x0000002C  00000000  ANDEQ     R0,R0,R0
0x00000030  00000000  ANDEQ     R0,R0,R0
0x00000034  00000000  ANDEQ     R0,R0,R0
0x00000038  00000000  ANDEQ     R0,R0,R0
0x0000003C  00000000  ANDEQ     R0,R0,R0
```

Example_p109.s

```
1         AREA Example, CODE
2         ENTRY                ; mark first instruction
3         BL  func1            ; call first subroutine
4         BL  func2            ; call second subroutine
5 stop    B   stop             ; terminate the program
6 func1   LDR r0, =42          ; => MOV r0, #42
7         LDR r1, =0x12345678  ; => LDR r1, [PC, #N]
8                              ; where N = offset to literal pool 1
9         LDR r2, =0xFFFFFFFF  ; => MVN r2, #0
10        BX lr                ; return from subroutine
11        LTORG                ; literal pool 1 has 0x12345678
12 func2  LDR r3, =0x12345678  ; => LDR r3, [PC, #N]
13                             ; N = offset back to literal pool 1
14        ;LDR r4, =0x87654321; if this is uncommented, it fails.
15                             ; Literal pool 2 is out of reach!
16        BX lr                ; return from subroutine
17 BigTable
18        SPACE 4200           ; clears 4200 bytes of memory,
19                             ; starting here
20        END                  ; literal pool 2 empty
```

p.84 Table 5.2

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

p.278 LDM/STM

or decremented, as shown in Figure 13.2. In the following sections, we'll examine stacks and the other addressing mode suffixes that are easier to use for stack operations.

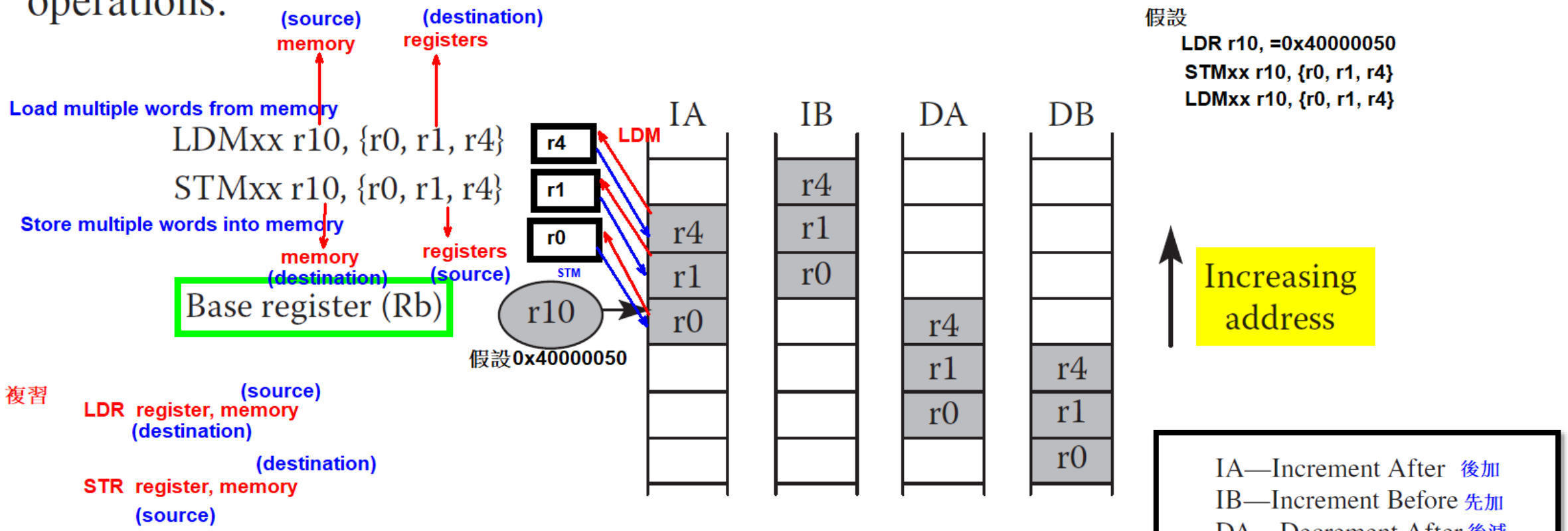


FIGURE 13.2 LDM/STM operations.

第4次隨堂考

2023 Advanced Mixed-Operation System (AMOS) Lab.

Tamkang University
Department of Electrical and Computer Engineering
No.151, Yingzhuan Rd., Tamsui Dist., New Taipei City 25137, Taiwan (R.O.C.)

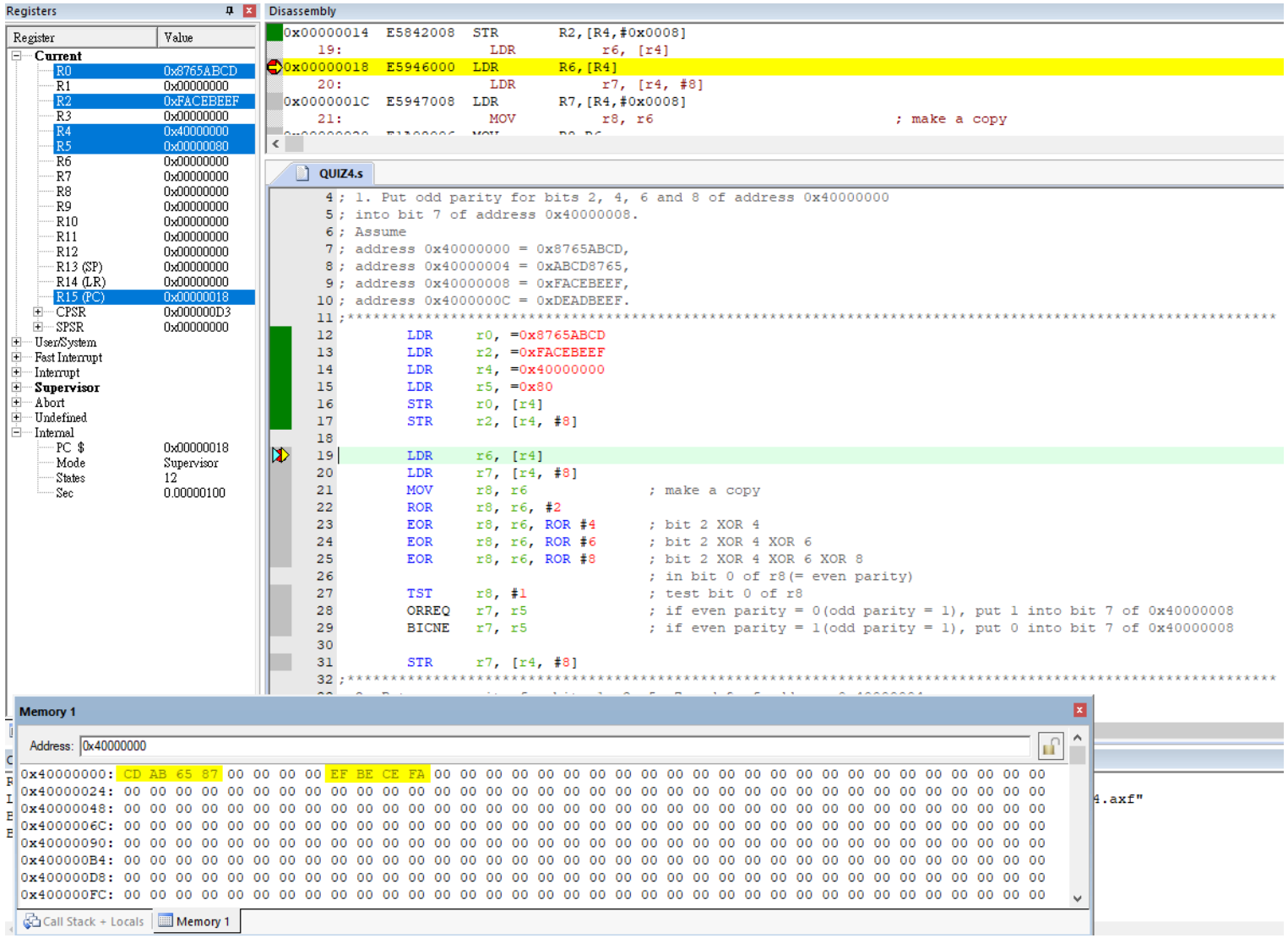
評分標準

→共有2題，一題50分。
(每一題配分如下表所示)

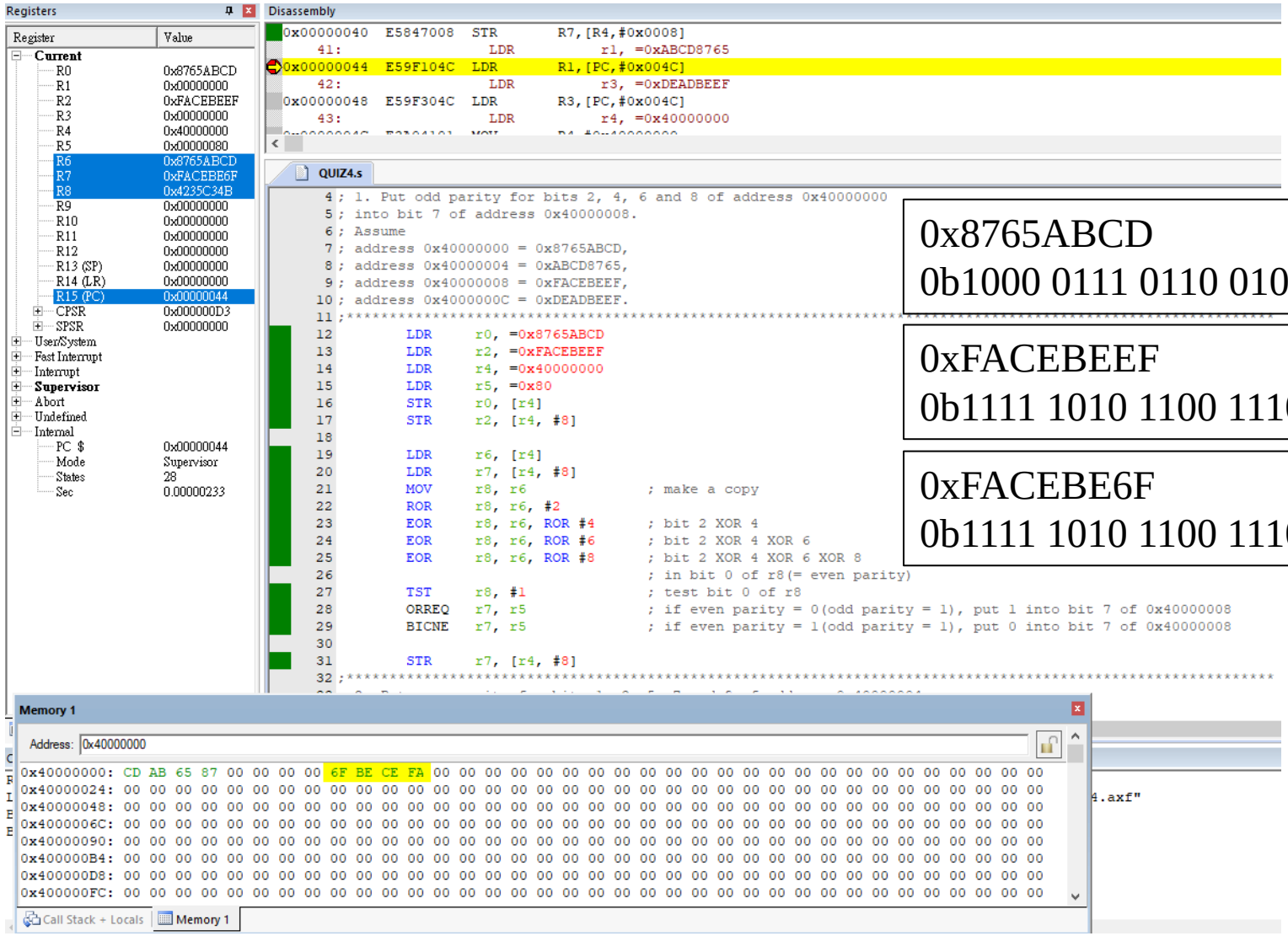
0	未繳交、交白券、**程式碼裡沒學號姓名**
10	基本分(只交程式碼，沒進入Debugger介面)
20	有進入Debugger介面，程式碼與題目要求的差很多
30	程式碼有小錯誤，導致輸出結果數值不正確
40	輸出結果數值正確，但未將輸出結果存回記憶體
45	未初始化
50	完全正確

→補繳分數 = 原始分數*0.9

第一題-初始化


Registers
Register
Value
Current
R0 0x8765ABCD
R1 0x00000000
R2 0xFACEBEEF
R3 0x00000000
R4 0x40000000
R5 0x00000080
R6 0x00000000
R7 0x00000000
R8 0x00000000
R9 0x00000000
R10 0x00000000
R11 0x00000000
R12 0x00000000
R13 (SP) 0x00000000
R14 (LR) 0x00000000
R15 (PC) 0x00000018
CPSR 0x000000D3
SPSR 0x00000000
User/System
Fast Interrupt
Interrupt
Supervisor
Abort
Undefined
Internal
PC $ 0x00000018
Mode Supervisor
States 12
Sec 0.00000100
Disassembly
0x00000014 E5842008 STR R2,[R4,#0x0008]
19: LDR r6, [r4]
0x00000018 E5946000 LDR R6,[R4]
20: LDR r7, [r4, #8]
0x0000001C E5947008 LDR R7,[R4,#0x0008]
21: MOV r8, r6 ; make a copy
QUIZ4.s
4 ; 1. Put odd parity for bits 2, 4, 6 and 8 of address 0x40000000
5 ; into bit 7 of address 0x40000008.
6 ; Assume
7 ; address 0x40000000 = 0x8765ABCD,
8 ; address 0x40000004 = 0xABCD8765,
9 ; address 0x40000008 = 0xFACEBEEF,
10 ; address 0x4000000C = 0xDEADBEEF.
12 LDR r0, =0x8765ABCD
13 LDR r2, =0xFACEBEEF
14 LDR r4, =0x40000000
15 LDR r5, =0x80
16 STR r0, [r4]
17 STR r2, [r4, #8]
19 LDR r6, [r4]
20 LDR r7, [r4, #8]
21 MOV r8, r6 ; make a copy
22 ROR r8, r6, #2
23 EOR r8, r6, ROR #4 ; bit 2 XOR 4
24 EOR r8, r6, ROR #6 ; bit 2 XOR 4 XOR 6
25 EOR r8, r6, ROR #8 ; bit 2 XOR 4 XOR 6 XOR 8
26 ; in bit 0 of r8(= even parity)
27 TST r8, #1 ; test bit 0 of r8
28 ORREQ r7, r5 ; if even parity = 0(odd parity = 1), put 1 into bit 7 of 0x40000008
29 BICNE r7, r5 ; if even parity = 1(odd parity = 1), put 0 into bit 7 of 0x40000008
31 STR r7, [r4, #8]
Memory 1
Address: 0x40000000
0x40000000: CD AB 65 87 00 00 00 00 EF BE CE FA 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000024: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000048: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x4000006C: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000090: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x400000B4: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x400000D8: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x400000FC: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Call Stack + Locals
Memory 1

第一題

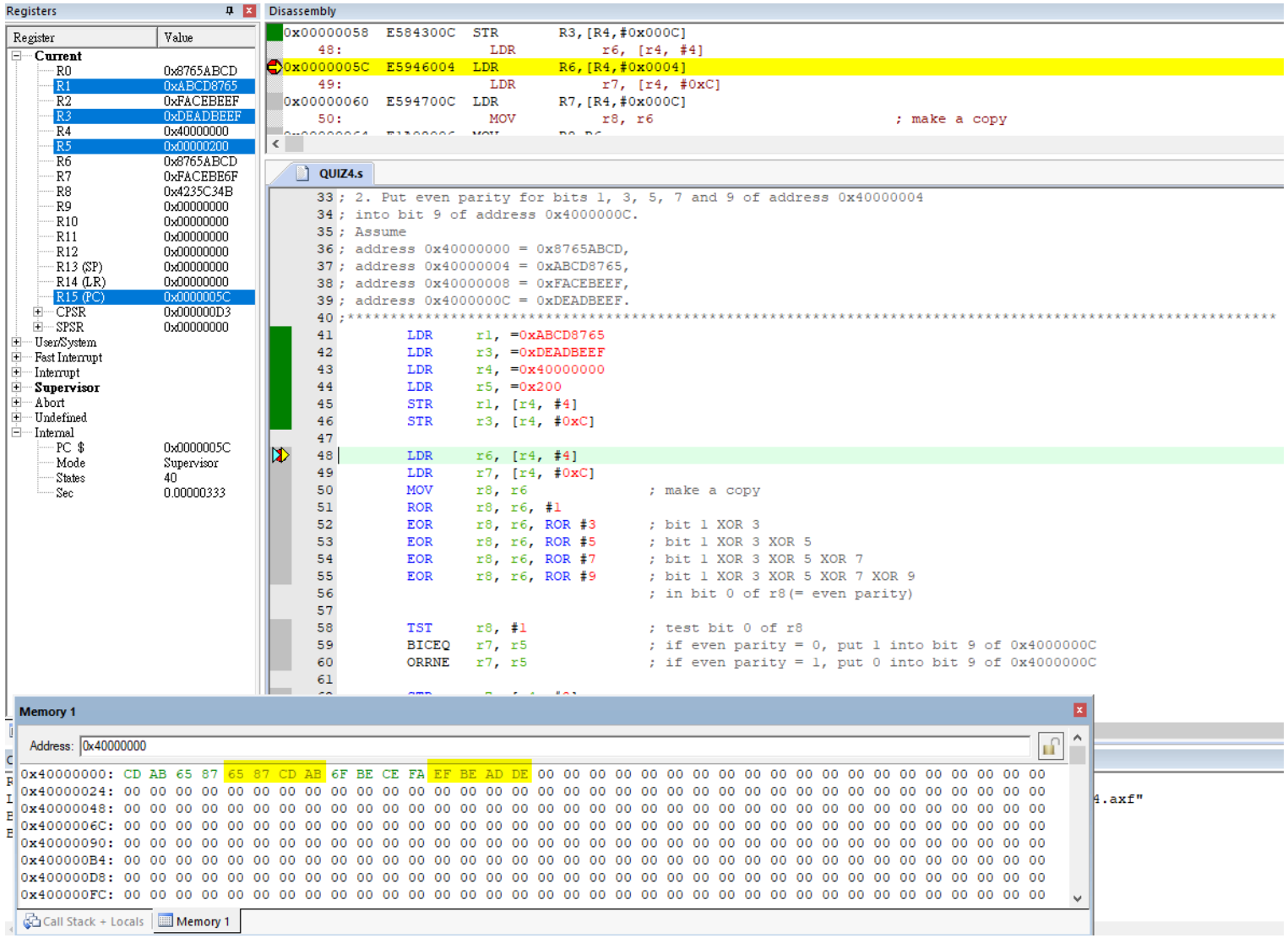


0x8765ABCD
0b1000 0111 0110 0101 1010 1011 1100 1101

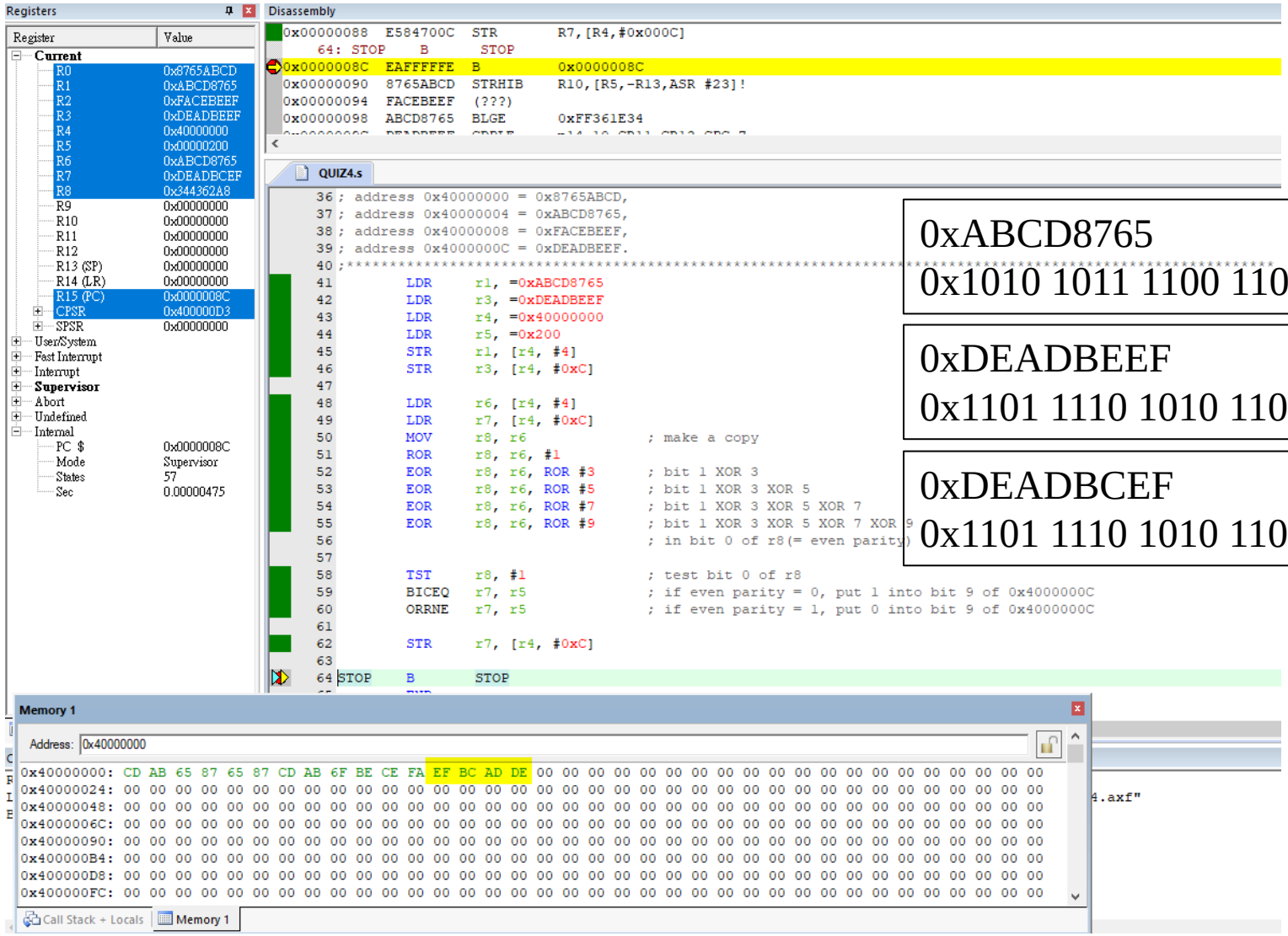
0xFACEBEEF
0b1111 1010 1100 1110 1011 1110 1110 1111

0xFACEBE6F
0b1111 1010 1100 1110 1011 1110 0110 1111

第二題-初始化


Registers
Register
Value
Current
R0 0x8765ABCD
R1 0xABCD8765
R2 0xFACEBEEF
R3 0xDEADBEEF
R4 0x40000000
R5 0x00000200
R6 0x8765ABCD
R7 0xFACEBE6F
R8 0x4235C34B
R9 0x00000000
R10 0x00000000
R11 0x00000000
R12 0x00000000
R13 (SP) 0x00000000
R14 (LR) 0x00000000
R15 (PC) 0x0000005C
CPSR 0x000000D3
SPSR 0x00000000
User/System
Fast Interrupt
Interrupt
Supervisor
Abort
Undefined
Internal
PC $ 0x0000005C
Mode Supervisor
States 40
Sec 0.00000333
Disassembly
0x00000058 E584300C STR R3,[R4,#0x000C]
48: LDR r6, [r4, #4]
0x0000005C E5946004 LDR R6,[R4,#0x0004]
49: LDR r7, [r4, #0xC]
0x00000060 E594700C LDR R7,[R4,#0x000C]
50: MOV r8, r6 ; make a copy
QUIZ4.s
33 ; 2. Put even parity for bits 1, 3, 5, 7 and 9 of address 0x40000004
34 ; into bit 9 of address 0x4000000C.
35 ; Assume
36 ; address 0x40000000 = 0x8765ABCD,
37 ; address 0x40000004 = 0xABCD8765,
38 ; address 0x40000008 = 0xFACEBEEF,
39 ; address 0x4000000C = 0xDEADBEEF.
40 ;*****
41 LDR r1, =0xABCD8765
42 LDR r3, =0xDEADBEEF
43 LDR r4, =0x40000000
44 LDR r5, =0x200
45 STR r1, [r4, #4]
46 STR r3, [r4, #0xC]
47
48 LDR r6, [r4, #4]
49 LDR r7, [r4, #0xC]
50 MOV r8, r6 ; make a copy
51 ROR r8, r6, #1
52 EOR r8, r6, ROR #3 ; bit 1 XOR 3
53 EOR r8, r6, ROR #5 ; bit 1 XOR 3 XOR 5
54 EOR r8, r6, ROR #7 ; bit 1 XOR 3 XOR 5 XOR 7
55 EOR r8, r6, ROR #9 ; bit 1 XOR 3 XOR 5 XOR 7 XOR 9
56 ; in bit 0 of r8(= even parity)
57
58 TST r8, #1 ; test bit 0 of r8
59 BICEQ r7, r5 ; if even parity = 0, put 1 into bit 9 of 0x4000000C
60 ORRNE r7, r5 ; if even parity = 1, put 0 into bit 9 of 0x4000000C
61
Memory 1
Address: 0x40000000
0x40000000: CD AB 65 87 65 87 CD AB 6F BE CE FA EF BE AD DE 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000024: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000048: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x4000006C: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000090: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x400000B4: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x400000D8: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x400000FC: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Call Stack + Locals
Memory 1
4.axf"

第二題



0xABCD8765
0x1010 1011 1100 1101 1000 0111 0110 0101

0xDEADBEEF
0x1101 1110 1010 1101 1011 1110 1110 1111

0xDEADBCEF
0x1101 1110 1010 1101 1011 1100 1110 1111

1219正課複習

2023 Advanced Mixed-Operation System (AMOS) Lab.

Tamkang University
Department of Electrical and Computer Engineering
No.151, Yingzhuan Rd., Tamsui Dist., New Taipei City 25137, Taiwan (R.O.C.)

p.278 LDM/STM

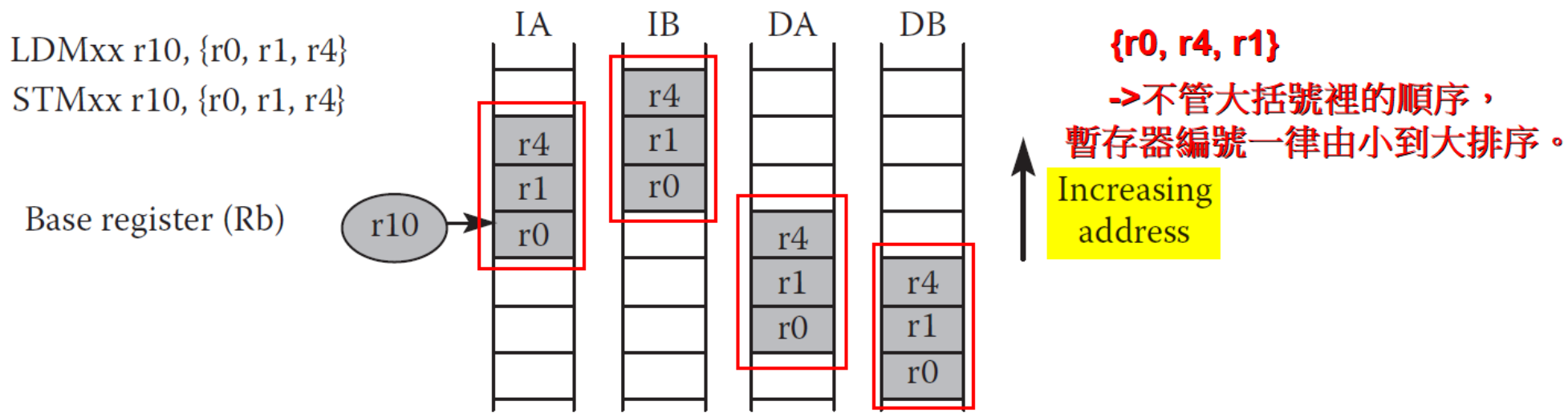


FIGURE 13.2 LDM/STM operations.

where the options are identical to those for the LDM instruction. The syntax for the Cortex-M3/M4 is

STM <address-mode> {<cond>} <Rn> {!}, <reg-list>

有加!，Rn值會改變。

EX：
STMIA r10, {r0, r1, r4}
r10 = 0x40000000
STMIA r10!, {r0, r1, r4}
r10 = 0x4000000C(有加!)

p.109 Subroutine

r14(LR)
r15(PC)

r13(Stack Point, SP)
LDR r10, =0x40000030

(PUSH)
STMIA r10!, {r0, r1, r2}
STMIA r10, {r0, r1, r2}

(POP)
LDMDB r10!, {r0, r1, r2}
LDMIA r10, {r0, r1, r2}

```
                AREA Example, CODE
                ENTRY                       ; mark first instruction
                BL      func1               ; call first subroutine
                BL      func2               ; call second subroutine
stop            B       stop                ; terminate the program
func1           LDR     r0, =42             ; => MOV r0, #42
                LDR     r1, =0x12345678     ; => LDR r1, [PC, #N]
                                            ; where N=offset to literal pool 1
                LDR     r2, =0xFFFFFFFF     ; => MVN r2, #0
                BX      lr                  ; return from subroutine
                LTORG                       ; literal pool 1 has 0x12345678
func2           LDR     r3, =0x12345678     ; => LDR r3, [PC, #N]
                                            ; N=offset back to literal pool 1
                ;LDR    r4, =0x87654321     ; if this is uncommented, it fails.
                                            ; Literal pool 2 is out of reach!
                BX      lr                  ; return from subroutine
BigTable
                SPACE 4200                  ; clears 4200 bytes of memory,
                                            ; starting here
                END                         ; literal pool 2 empty
```

p.281 TABLE13.1

TABLE 13.1
Stack-Oriented Suffixes

Stack Type	PUSH	POP
Full descending	STMFD (STMDB)	LDMFD (LDMIA)
Full ascending	STMFA (STMIB)	LDMFA (LDMDA)
Empty descending	STMED (STMDA)	LDMED (LDMIB)
Empty ascending	STMEA (STMIA)	LDMEA (LDMDB)

小技巧：
D是減、I是加；
B是先、A是後。

EX：
STMIB(先加)
初始記憶體位置(Rb)
先加，再存。

Descending or ascending—The stack grows downward, starting with a high address and progressing to a lower one (a descending stack), or upward, starting from a low address and progressing to a higher address (an ascending stack).
Full or empty—The stack pointer can either point to the last item in the stack (a full stack), or the next free space on the stack (an empty stack).

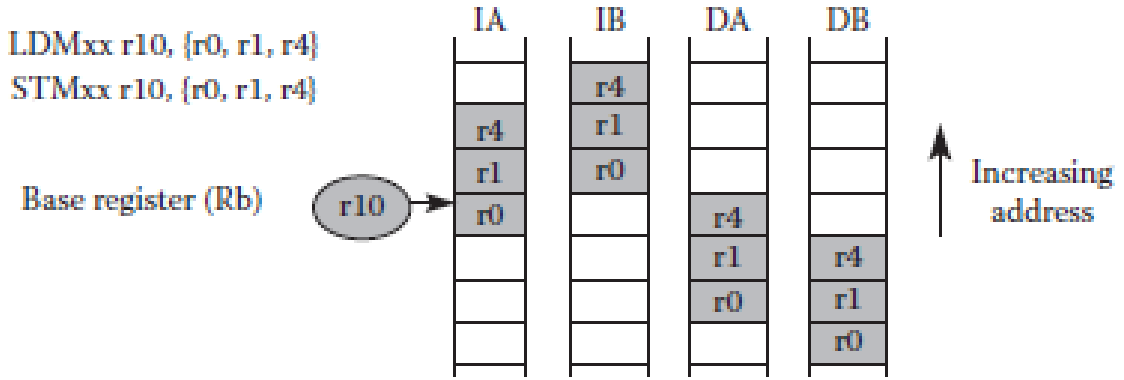


FIGURE 13.2 LDM/STM operations.

p.292 EXERCISES

13.6 EXERCISES

1. What's wrong with the following ARM7TDMI instructions?
   a. `STMIA r5!, {r5, r4, r9}` Source、Destination暫存器不能用一樣的
   b. `LDMDA r2, {}` 大括弧裡面不能為空
   c. `STMDB r15!, {r0-r3, r4, lr}` PC值隨著程式碼執行時而改變，且PC所指向的位置也不能給使用者自由存取。
2. On the ARM7TDMI, if register r6 holds the address 0x8000 and you executed the instruction

   `STMIA r6, {r7, r4, r0, lr}` 編號由小到大 (後加)

   what address now holds the value in register r0? Register r4? Register r7? The Link Register?

   0x8000 0x8004 0x8008 0x800C

p.293 EXERCISES

3. Assume that memory and ARM7TDMI registers r0 through r3 appear as follows:

Address			Register
0x8010	0x00000001	0x13	r0
0x800C	0xFEEDDEAF	0xFFFFFFFF	r1
0x8008	0x00008888	0xEEEEEEEE	r2
0x8004	0x12340000	0x8000	r3
0x8000	0xBABE0000		

Describe the memory and register contents after executing the instruction

```
LDMIA  r3!, {r0, r1, r2}
```

r0 = 0xBABE0000
r1 = 0x12340000
r2 = 0x00008888
r3 = 0x800C

Q&A

Thanks for your attention !!