第03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.)

第1次隨堂考講解

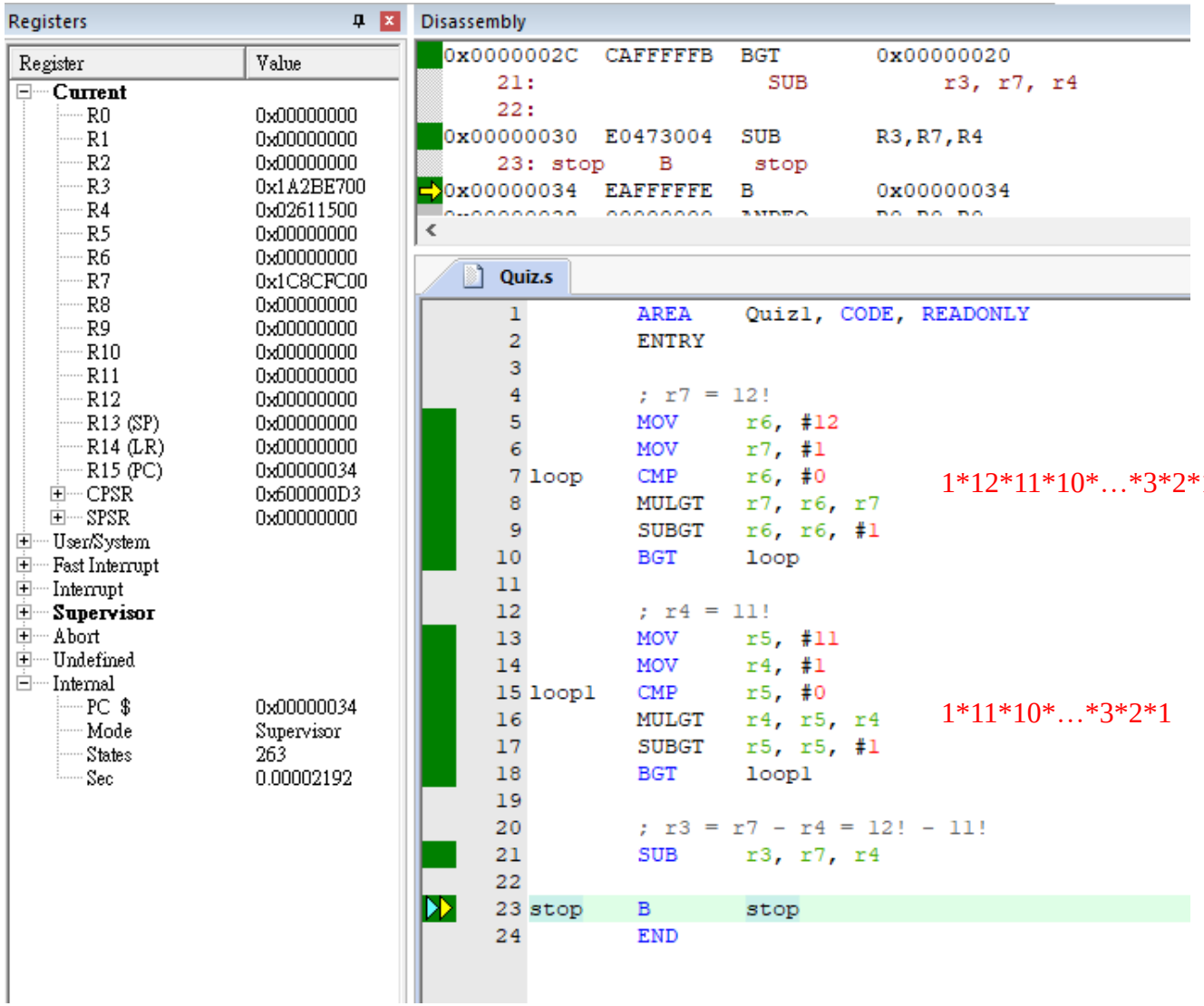
2023 Advanced Mixed-Operation System (AMOS) Lab.

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

題目&計算過程

- 將A, B, C, D, E, F, G, H, I, J, K, L 十二人排成一列，求L不排在末位的組合數。
  - 全部組合數-L排末位組合數
  - 12! - 11! = 439084800 = 0x1A2BE700

Keil Tool


Registers
Register
Value
Current
R0 0x00000000
R1 0x00000000
R2 0x00000000
R3 0x1A2BE700
R4 0x02611500
R5 0x00000000
R6 0x00000000
R7 0x1C8CFC00
R8 0x00000000
R9 0x00000000
R10 0x00000000
R11 0x00000000
R12 0x00000000
R13 (SP) 0x00000000
R14 (LR) 0x00000000
R15 (PC) 0x00000034
CPSR 0x600000D3
SPSR 0x00000000
User/System
Fast Interrupt
Interrupt
Supervisor
Abort
Undefined
Internal
PC $ 0x00000034
Mode Supervisor
States 263
Sec 0.00002192
Disassembly
0x0000002C CAFFFFFB BGT 0x00000020
21: SUB r3, r7, r4
22:
0x00000030 E0473004 SUB R3,R7,R4
23: stop B stop
0x00000034 EAFFFFFE B 0x00000034
Quiz.s
1 AREA Quiz1, CODE, READONLY
2 ENTRY
3
4 ; r7 = 12!
5 MOV r6, #12
6 MOV r7, #1
7 loop CMP r6, #0
8 MULGT r7, r6, r7
9 SUBGT r6, r6, #1
10 BGT loop
11
12 ; r4 = 11!
13 MOV r5, #11
14 MOV r4, #1
15 loop1 CMP r5, #0
16 MULGT r4, r5, r4
17 SUBGT r5, r5, #1
18 BGT loop1
19
20 ; r3 = r7 - r4 = 12! - 11!
21 SUB r3, r7, r4
22
23 stop B stop
24 END
1*12*11*10*…*3*2*1
1*11*10*…*3*2*1

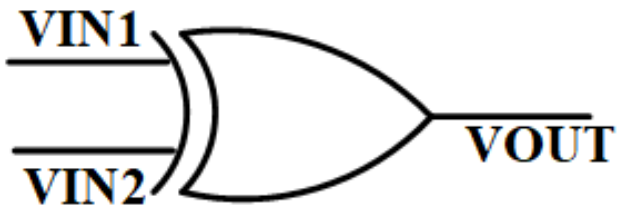
Program 3：Swapping Register Contents

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.)

XOR Gate 特性

- 兩輸入值**相同為0**，**不同為1**。
- 其中一輸入固定為0，則輸出為另一輸入的值。
- 其中一輸入固定為1，則輸出為另一輸入的反向。
- 具有**交換律**、**結合律**。



$$p \oplus q = q \oplus p$$

$$p \oplus (q \oplus r) = (p \oplus q) \oplus r$$

$$p \oplus 0 = p$$

$$p \oplus p = 0$$

$$p \oplus q \oplus q = p \oplus 0 = p$$

VIN1	VIN2	VOUT
0	0	0
0	1	1
1	0	1
1	1	0

Program 3：Swapping Register Contents(1/2)

subtraction, and the branch; otherwise, do not execute any of these instructions."

交換暫存器數值

3.5 PROGRAM 3: SWAPPING REGISTER CONTENTS

This next program is actually a useful way to shuffle data around, and a good exercise in Boolean arithmetic. A fast way to swap the contents of two registers without using an intermediate storage location (such as memory or another register) is to use the exclusive OR operator. Suppose two values A and B are to be exchanged. The following algorithm could be used:

$$A = A \oplus B$$
$$B = A \oplus B$$
$$A = A \oplus B$$

The ARM7TDMI code below implements this algorithm using the Keil assembler, where the values of A = 0xF631024C and B = 0x17539ABD are stored in registers r0 and r1, respectively.

pseudo instruction

Lord Register

Exclusive OR

```
        AREA Prog3, CODE, READONLY
        ENTRY
        LDR   r0, =0xF631024C   ; load some data
        LDR   r1, =0x17539ABD   ; load some data
        EOR   r0, r0, r1        ; r0 XOR r1
        EOR   r1, r0, r1        ; r1 XOR r0
        EOR   r0, r0, r1        ; r0 XOR r1
stop    B     stop              ; stop program
        END
```

MOV範圍限制：0~255 (1 byte)

Program 3：Swapping Register Contents(2/2)

subtraction, and the branch; otherwise, do not execute any of these instructions."

3.5 PROGRAM 3: SWAPPING REGISTER CONTENTS

This next program is actually a useful way to shuffle data around, and a good exercise in Boolean arithmetic. A fast way to swap the contents of two registers without using an intermediate storage location (such as memory or another register) is to use the exclusive OR operator. Suppose two values A and B are to be exchanged. The following algorithm could be used:

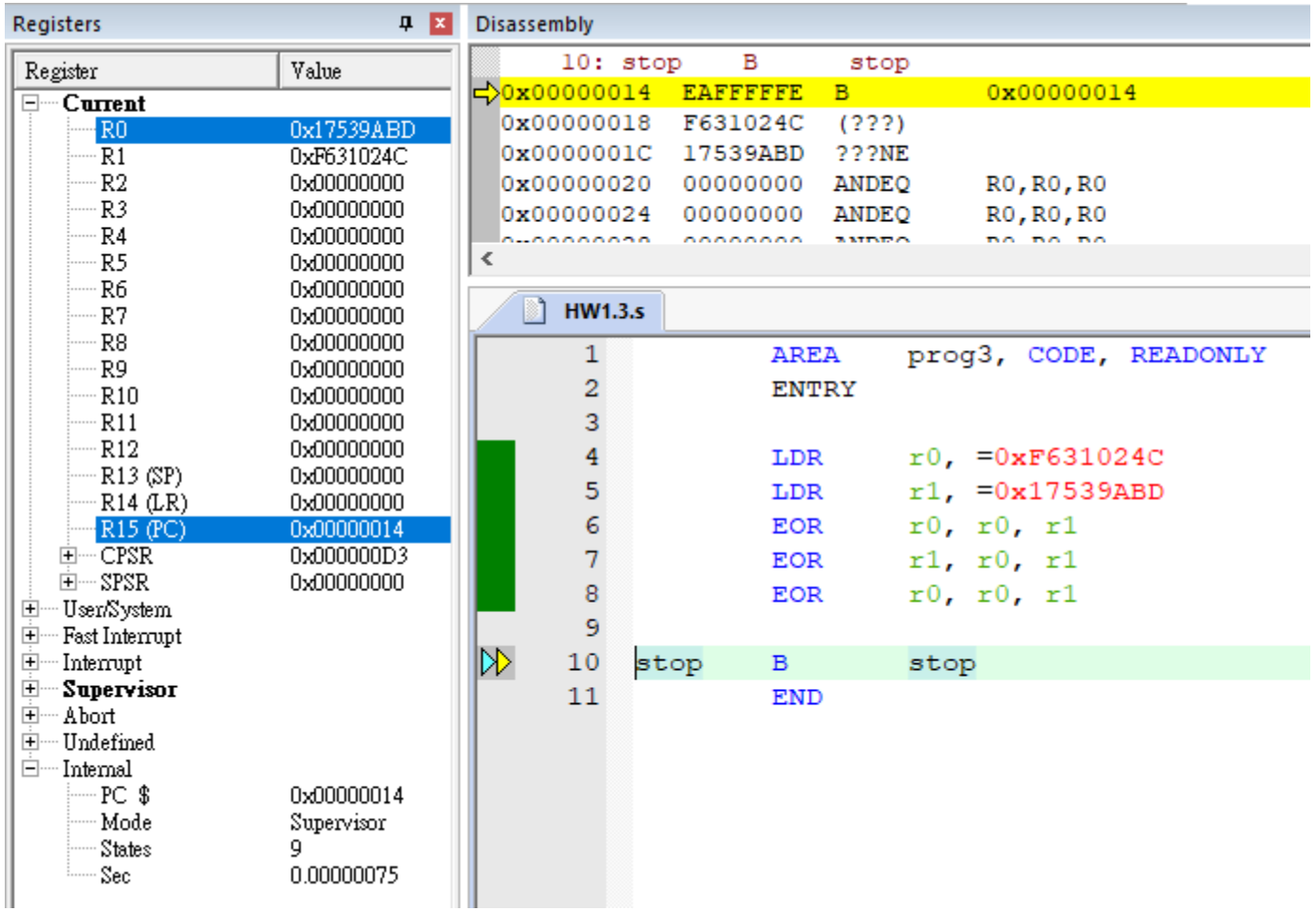
$$A = A \oplus B$$
$$B = A \oplus B$$
$$A = A \oplus B$$

The ARM7TDMI code below implements this algorithm using the Keil assembler, where the values of A = 0xF631024C and B = 0x17539ABD are stored in registers r0 and r1, respectively.

```
        AREA Prog3, CODE, READONLY
        ENTRY
        LDR   r0, =0xF631024C   ; load some data
        LDR   r1, =0x17539ABD   ; load some data
        EOR   r0, r0, r1        ; r0 XOR r1
        EOR   r1, r0, r1        ; r1 XOR r0
        EOR   r0, r0, r1        ; r0 XOR r1
stop    B     stop              ; stop program
        END
```

r0 = r0 XOR r1
r1 = r0 XOR r1 XOR r1 → r0 XOR 0
r0 = r0 XOR r1 XOR r0 → r1 XOR 0

Program 3：Swapping Register Contents using Keil Tool



DCB, DCW, DCD

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.)

Frequently Used Directives

TABLE 4.2
Frequently Used Directives

Keil Directive	CCS Directive	Uses
AREA	.sect	Defines a block of code or data
RN	.asg	Can be used to associate a register with a name
EQU	.equ	Equates a symbol to a numeric constant
ENTRY		Declares an entry point to your program
DCB, DCW, DCD	.byte, .half, .word	Allocates memory and specifies initial runtime contents
ALIGN	.align	Aligns data or code to a particular memory boundary
SPACE	.space	Reserves a zeroed block of memory of a particular size
LTORG		Assigns the starting point of a literal pool
END	.end	Designates the end of a source file

Allocating Memory and Specifying Contents

8位元二補數

二進位值	二補數表示	無符號數表示
00000000	0	0
00000001	1	1
...	...	...
01111110	126	126
01111111	127	127
10000000	−128	128
10000001	−127	129
10000010	−126	130
...	...	...
11111110	−2	254
11111111	−1	255

4.4.5 Allocating Memory and Specifying Contents

When writing programs that contain tables or data that must be configured before the program begins, it is necessary to specify exactly what memory looks like. Strings, floating-point constants, and even addresses can be stored in memory as data using various directives.

4.4.5.1 Keil Tools

One of the more common directives, DCB, actually defines the initial runtime contents of memory. The syntax is

unsigned numbers (nonnegatives) (0、positives) (0~255)

signed numbers (negatives、0、positives)

{*label*} DCB *expr*{,*expr*}…

byte

where *expr* is either a numeric expression that evaluates to an integer in the range –128 to 255, or a quoted string, where the characters of the string are stored consecutively in memory. Since the DCB directive affects memory at the byte level, you should use an ALIGN directive afterward if any instructions follow to ensure that the instruction is aligned correctly in memory.

最高有效位(sign bit)：0表示正數、1表示負數

➢ 複習2's complements：
0000 1101 ← 13
1111 0010 ← 1's complements
+1
1111 0011 ← -13

Allocating Memory and Specifying Contents

	A	B	C	D	E	F	G	H	I	J	K	L
1	10進制	16進制	字元	10進制	16進制	字元	10進制	16進制	字元	10進制	16進制	字元
2	33	21	!	57	39	9	81	51	Q	105	69	i
3	34	22	"	58	3A	:	82	52	R	106	6A	j
4	35	23	#	59	3B	;	83	53	S	107	6B	k
5	36	24	$	60	3C	<	84	54	T	108	6C	l
6	37	25	%	61	3D	=	85	55	U	109	6D	m
7	38	26	&	62	3E	>	86	56	V	110	6E	n
8	39	27	'	63	3F	?	87	57	W	111	6F	o
9	40	28	(	64	40	@	88	58	X	112	70	p
10	41	29	)	65	41	A	89	59	Y	113	71	q
11	42	2A	*	66	42	B	90	5A	Z	114	72	r
12	43	2B	+	67	43	C	91	5B	[	115	73	s
13	44	2C	,	68	44	D	92	5C	\	116	74	t
14	45	2D	-	69	45	E	93	5D	]	117	75	u
15	46	2E	.	70	46	F	94	5E	^	118	76	v
16	47	2F	/	71	47	G	95	5F	_	119	77	w
17	48	30	0	72	48	H	96	60	`	120	78	x
18	49	31	1	73	49	I	97	61	a	121	79	y
19	50	32	2	74	4A	J	98	62	b	122	7A	z
20	51	33	3	75	4B	K	99	63	c	123	7B	{
21	52	34	4	76	4C	L	100	64	d	124	7C	|
22	53	35	5	77	4D	M	101	65	e	125	7D	}
23	54	36	6	78	4E	N	102	66	f	126	7E	~
24	55	37	7	79	4F	O	103	67	g	127	7F	
25	56	38	8	80	50	P	104	68	h	128	80	

EXAMPLE 4.6

Unlike strings in C, ARM assembler strings are not null-terminated. You can construct a null-terminated string using DCB as follows:

```
C_string DCB "C_string",0
```
→Null character

If this string started at address 0x4000 in memory, it would look like

Address		ASCII equivalent
0x4000	43	C
0x4001	5F	_
0x4002	73	s
0x4003	74	t
0x4004	72	r
0x4005	69	i
0x4006	6E	n
0x4007	67	g
0x4008	00	

Allocating Memory and Specifying Contents

In addition to the directive for allocating memory at the resolution of bytes, there are directives for reserving and defining halfwords and words, with and without alignment. The DCW directive allocates one or more halfwords of memory, aligned on two-byte boundaries (DCWU does the same thing, only without the memory alignment). The syntax for these directives is

{*label*} DCW{U} *expr*{,*expr*}…

➢ Halfword(16 bits/2 bytes)

where *expr* is a numeric expression that evaluates to an integer in the range –32768 to 65535.

Another frequently used directive, DCD, allocates one or more words of memory, aligned on four-byte boundaries (DCDU does the same thing, only without the memory alignment). The syntax for these directives is

{*label*} DCD{U} *expr*{,*expr*}

➢ word(32 bits/4 bytes)

where *expr* is either a numeric expression or a program-relative expression. DCD inserts up to 3 bytes of padding before the first defined word, if necessary, to achieve a 4-byte alignment. If alignment isn't required, then use the DCDU directive.

延用過去的寫法：
- DCW：W表示Word
(過去一個word：16 bits)
- DCD：D表示Double word

DCW vs. DCWU

DCW

halfword(佔2個byte)

在記憶體裡需要以2的倍數對齊存放

DCWU

在記憶體裡不需要以2的倍數對齊存放

DCD vs. DCDU

DCD

Double word(佔4個byte)

在記憶體裡需要以4的倍數對齊存放

DCDU

在記憶體裡不需要以4的倍數對齊存放

Example 4.7

假設題目如下

```
1  coeff    DCW    0xFE37, 0x8ECC
2  data1    DCD    1, 5, 20
3  data2    DCB    255
4  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

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

Q&A

Thanks for your attention !!