國立中山大學 101 學年度第一學期資工系組合語言與微處理機期末考試題

學號：　　　　　　　　姓名：

一、選擇與是非題（每題 3 分，15 分）

(　　) 1. 關於 APCS （ARM Procedure Call Standard）何者錯誤？ ①four argument registers (r0~r3) which pass values into the function ②r0~r3 are callee-saved register variables ③five (r4~r8) register variables which the function must return with unchanged values ④ r14: link register。

(　　) 2. 關於 Thumb instruction 何者錯誤？①most instructions are executed conditionally ②many data processing instructions use a 2-address format ③less regular than ARM instruction format ④Thumb code uses 40% more instructions than the ARM code。

(　　) 3. FPA10 coprocessor 的 register bank 中有 8 個 double precision floating-point registers。

(　　) 4. ARM code 使用 BX 指令呼叫 Thumb subroutine，而 Thumb subroutine 使用 BLX r14 返回 ARM code。

(　　) 5. 若 memory 為 32-bit，Thumb code 執行較快；若 memory 為 16-bit，ARM code 執行較快。

二、問答題（88 分）

1. Write the decimal number 2013 in IEEE 754 single precision floating-point format as shown in Fig. 1. (6%)

2. Please explain the meaning of the following ARM instructions. (10%)

```
(1) MRS     r0, CPSR                  (2) LDC      p6, C5, [r0]
    ORR     r0, r0, #&20000000            CDPEQ    p6, 6, C1, C5, C7, 4
    MSR     CPSR_f, r0                    MCR      p6, 3, r2, C1, C2
```

3. Please write the binary encodings of the following ARM instructions by Fig. 2 to Fig. 5 and the following coding table of shift operation. (12%)

(1) ADDLT r3, r6, r9, LSL #2　　(2) LDR r3, [r5 , #4] !　　(3) STRNE r9, [r1, r7, LSR r2]

00	LSL	01	LSR	10	ASR	11	ROR

4. Find the instruction coding of Thumb instruction “BL target” in the following program according to the Thumb branch instruction binary encoding shown in Fig. 6. (8%)

```
pc=1024             BL target
pc=1028             MOV r1, r2
pc=1030             SUB r3, r1
                    • • •
pc=9274   target    ADD r1, r2        • • •
```

5. The instruction coding of Thumb data processing instructions is shown in Fig. 7.

(1) Check if the following Thumb instruction syntax is correct. If not, you should also explain why. (9%)
(a) SUB r8, r1, #32　　(b) ADD r1, r2, r3, LSR #2　　(c) SUBEQ SP, SP, #58

(2) Write the equivalent 32-bit ARM instruction for the following Thumb instruction: (9%)
(a) ADC r0, r6　　(b) ASR r1, r3, #3　　(c) PUSH {r2, r5}

6. Translate the following C programs into the ARM assembly language programs with higher performance using the following register allocation: i→r0, x→r1, y→r2, sum→r3. (8%, 8%, 10%)

```
(1) for (i=0; i<100; i++) {      (2) while ( x > y ) {          (3) switch (i) {
        a[i] = 0;                        sum = sum + x;                1: statements 1; break;
    }                                    x = x – y;                    2: statements 2; break;
                                     }                                    • • •
                                                                       N: statements N; break;
                                                                       default:   ;
                                                                   }
```

7. Suppose one ARM-based embedded system uses total of 2G-bytes ROM and 2G-bytes of SRAM memory. Please explain the behavior of the simple ARM memory system control logic shown in Fig. 8(b). (8%)

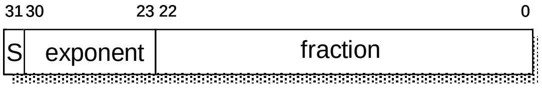


Fig. 1

1. H=0 LR:=PC+(sign-extended offset shifted left 12 places);
2. H=1 PC:=LR+(offset shifted left 1 place); LR:=oldPC+3.

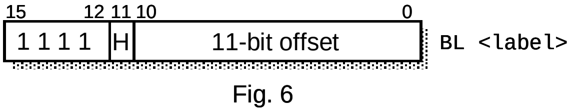
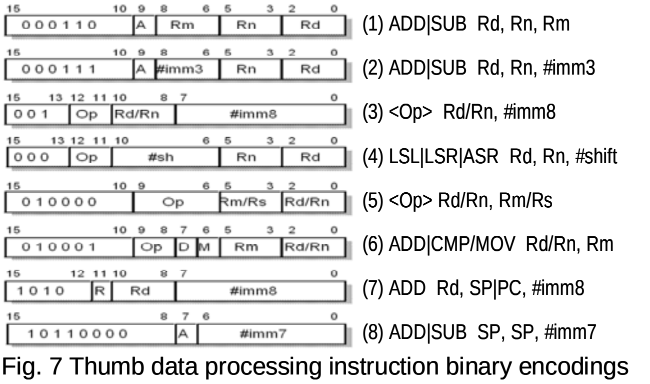


Fig. 6



15 10 9 8 6 5 3 2 0
0 0 0 1 1 0 | A | Rm | Rn | Rd
(1) ADD|SUB Rd, Rn, Rm
15 10 9 8 6 5 3 2 0
0 0 0 1 1 1 | A | #imm3 | Rn | Rd
(2) ADD|SUB Rd, Rn, #imm3
15 13 12 11 10 8 7 0
0 0 1 | Op | Rd/Rn | #imm8
(3) <Op> Rd/Rn, #imm8
15 13 12 11 10 6 5 3 2 0
0 0 0 | Op | #sh | Rn | Rd
(4) LSL|LSR|ASR Rd, Rn, #shift
15 10 9 6 5 3 2 0
0 1 0 0 0 0 | Op | Rm/Rs | Rd/Rn
(5) <Op> Rd/Rn, Rm/Rs
15 10 9 8 7 6 5 3 2 0
0 1 0 0 0 1 | Op | D | M | Rm | Rd/Rn
(6) ADD|CMP/MOV Rd/Rn, Rm
15 12 11 10 8 7 0
1 0 1 0 | R | Rd | #imm8
(7) ADD Rd, SP|PC, #imm8
15 8 7 6 0
1 0 1 1 0 0 0 0 | A | #imm7
(8) ADD|SUB SP, SP, #imm7

Fig. 7 Thumb data processing instruction binary encodings

Opcode [31:28]	Mnemonic extension	Interpretation
0000	EQ	Equal / equals zero
0001	NE	Not equal
1000	HI	Unsigned higher
1001	LS	Unsigned lower or same
1010	GE	Signed greater than or equal
1011	LT	Signed less than
1100	GT	Signed greater than
1101	LE	Signed less than or equal
1110	AL	Always
1111	NV	Never (do not use!)

Fig. 2 ARM condition codes

Opcode [24:21]	Mnemonic	Meaning
0010	SUB	Subtract
0011	RSB	Reverse subtract
0100	ADD	Add
0101	ADC	Add with carry
0110	SBC	Subtract with carry
0111	RSC	Reverse subtract with carry
1000	TST	Test
1001	TEQ	Test equivalence
1010	CMP	Compare
1101	MOV	Move

Fig. 3 ARM data processing instructions

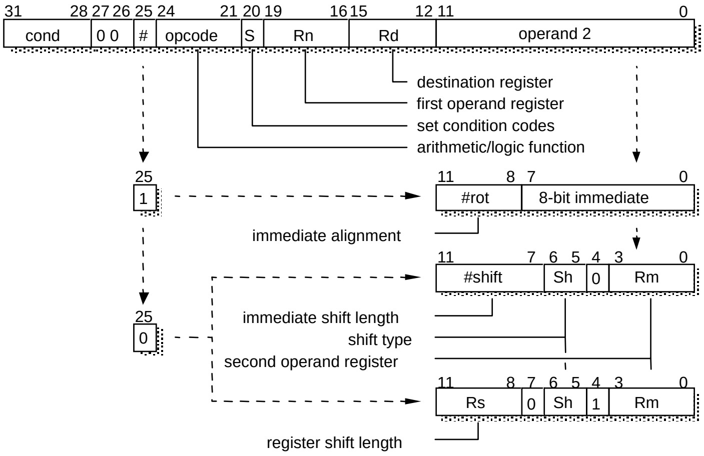


Fig. 4 Data processing instruction binary encoding

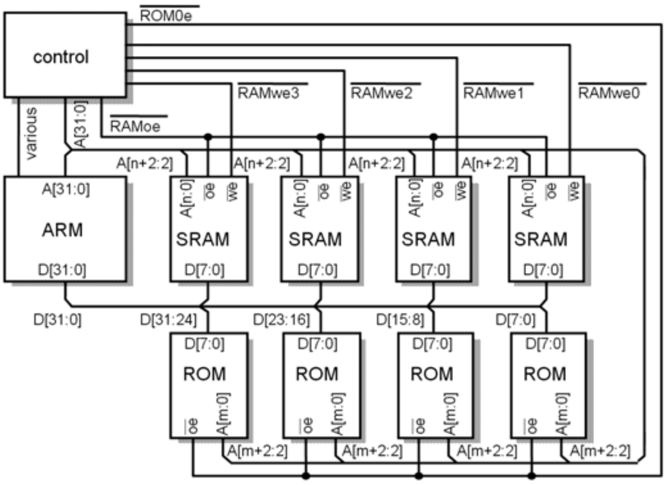
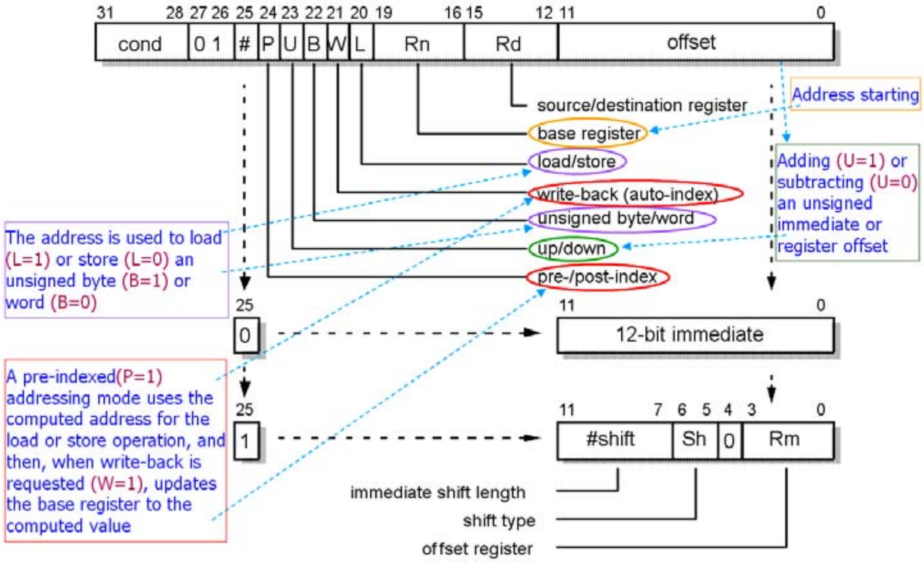


Fig. 8 (a)



Address starting

Adding (U=1) or subtracting (U=0) an unsigned immediate or register offset

The address is used to load (L=1) or store (L=0) an unsigned byte (B=1) or word (B=0)

A pre-indexed(P=1) addressing mode uses the computed address for the load or store operation, and then, when write-back is requested (W=1), updates the base register to the computed value

Fig. 5 data transfer instruction binary encoding

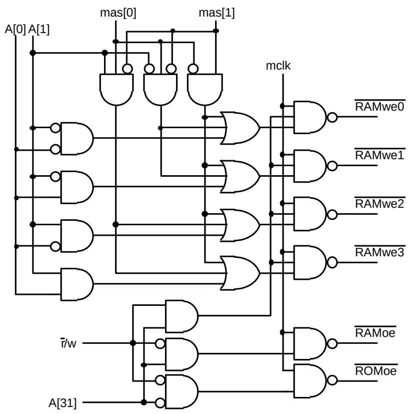


Fig. 8 (b)

Fig. 8 Simple ARM memory system control logic