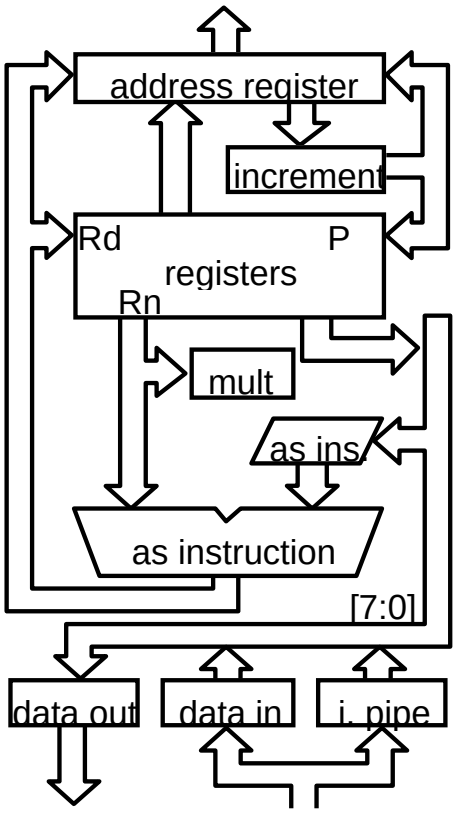
National Sun Yat-Sen University
ASSEMBLY LANGUAGE AND MICROCOMPUTER
Final Exam
2010/06/25 9:20-11:30AM

Name: _____

Note: Although there are more than 100 points for this exam, the maximum score you can get is 100 points.

1. Refer to the following 3-stage (fetch, decode, execute) ARM7 pipeline data path. ***(16 pts)***
   (a) Find out the number of cycles it will takes to run the ARM instruction ***LDR r0, [r1], #4*** at the execution stages. ***(4 pts)***
   (b) Show the datapath activity at each cycle. ***(6 pts)***
   (c) Fill the following immediate field of the instruction used to return from an IRQ. You have to explain the reason. ***(6 pts)***

   SUBS pc, r14,_____



2. Show how the variable ***data*** is organized in the little-endian memory without packing. ***(10 pts)***

```
struct T1 {float a[2], short b};
struct T2 { T1 c , char d2[5] } data;
```

3. Answer the following short questions: ***(9 pts)***
   (a) Explain what "**caller-saved**" register variables mean. ***(3 pts)***
   (b) Explain what the "**stack**" is and list three different usages of stack to support the high-level programming. ***(6 pts)***

4. For the following simple assembly code: ***(15 pts)***
   (a) Explain the function of this code. ***(5 pts)***
   (b) Describe the main drawback of this realization. Rewrite and improve this code to solve the drawback. ***(10 pts)***

```
        BL      TAB
TAB     CMP     r0, #0
        BEQ     SUB0
        CMP     r0, #1
        BEQ     SUB1
        ……..
        CMP     r0, #9
        BEQ     SUB9
```

5. The instruction coding of Thumb data processing instructions is shown in the following figure. ***(21 pts)***

(a) Check if the following Thumb instruction syntax is correct. If not, you should also explain why. ***(12 pts)***

(1) SUB r3, r8, r2

(2) ADDNE r2, r3, #1

(3) CMP r13, r9

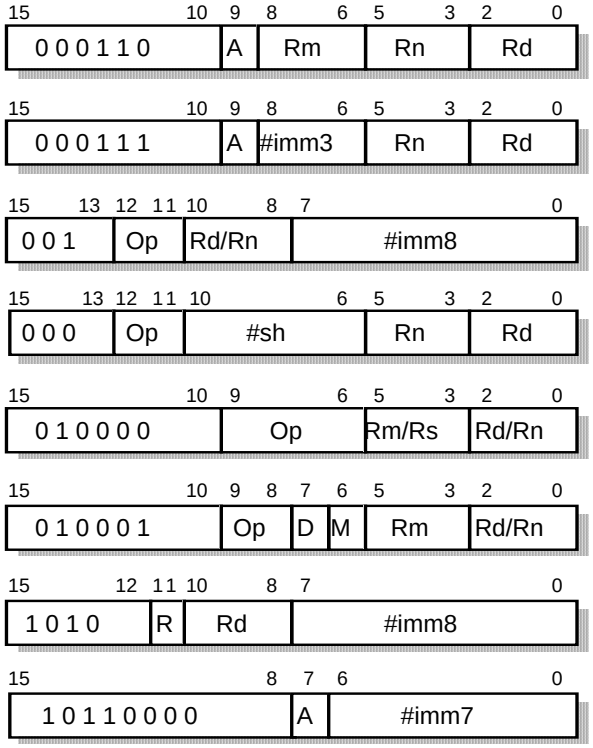
(4) ADD r6, r14, 29

(b) Write the equivalent 32-bit ARM instruction for the following Thumb instruction: ***(9 pts)***

(1) POP r4, r0, pc

(2) SUB r3, #52

(3) ASR r2, r3, #3



(1) `ADD|SUB Rd,Rn,Rm`

(2) `ADD|SUB Rd,Rn,#imm3`

(3) `<Op> R d/Rn ,#imm8`

(4) `LSL|LSR|ASR Rd,Rn,#shift`

(5) `<Op> Rd/Rn,Rm/Rs`

(6) `ADD|CMP|MOV Rd/Rn,Rm`

(7) `ADD Rd,SP|PC,#imm8`

(8) `ADD|SUB SP,SP,#imm7`

6. For the thumb instruction, ***(10 pts)***

(c) Fill a correct instruction in the empty box in the program shown in the right such that it can call the subroutine written in Thumb instruction correctly. ***(5 pts)***

(d) Modify the following thumb code such that it can perform the division correctly. ***(5 pts)***

```
                CODE 32
                [          ]
                BLX     r0

                CODE 16
Thumb           ADD     r1, #1
                BX      lr
```

```
                MOV r3, #0
Loop            SUB     r0, r1
                ADD     r3, #1
                BGE     Loop
                SUB     r3, #1
                ADD     r0, r1, r2
```

7. Find out the 32-bit instruction coding for the following ARM instructions based on the given coding information. ***(12 pts)***

   (a) ADCEQS r7, r1, r0, LSL #2

   (b) RSB r1, r2, r4

   (c) SUBLT r1, r3, #23

Coding table of ARM condition codes

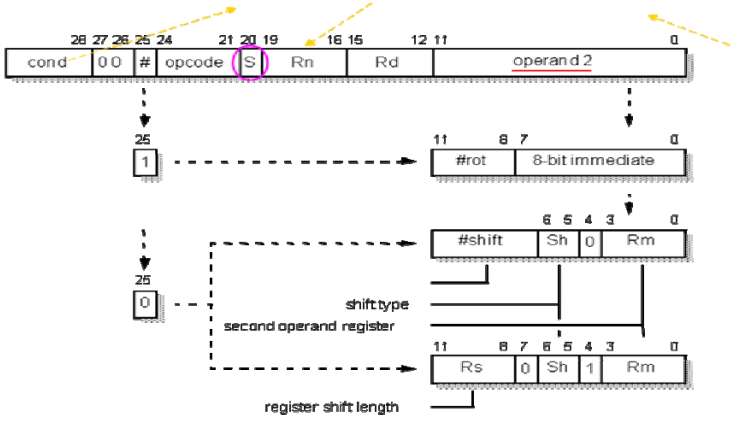
0000	EQ	0100	MI	1000	HI	1100	GT
0001	NE	0101	PL	1001	LS	1101	LE
0010	CS/HS	0110	VS	1010	GE	1110	AL
0011	CC/LO	0111	VC	1011	LT	1111	NV

Coding table of Opcode

0000	AND	0100	ADD	1000	TST	1100	ORR
0001	EOR	0101	ADC	1001	TEQ	1101	MOV
0010	SUB	0110	SBC	1010	CMP	1110	BIC
0011	RSB	0111	RSC	1011	CMN	1111	MVB

Coding table of Shift Operation

00	LSL	01	LSR	10	ASR	11	ROR



8. Figure 7(b) shows the partial assembly code corresponding to the original C code shown in Fig. 7(a). Complete the assembly code by filling the ten space regions. ***(15 pts)***

```
#include <stdio.h>

int func1 (int a);
int func2 (int a, int b);

int main()
{       int a, b, c, d, e;
                char data[20];

                a = 6;
                b = 7;
                c = 10;

                d=func1(c);
                e=func2(a, b);

        printf("Result : %s %d %d\n",data, d, e);

        return 0;
}

int func1 (int a)
{
        int c[10], b;

        b = c[2]+a;

        return(b);
}
```

```
func1
$a
.text
    0x00000000:    SUB     r13,r13,#0x28
    0x00000004:    LDR     _____________
    0x00000008:    ADD     r0,r1,r0
    0x0000000c:    ADD     _____________
    0x00000010:    MOV     _____________
main
    0x00000014:    STMFD   _____________
    0x00000018:    SUB     r13,r13,#0x18
    0x0000001c:    MOV     r2,#6
    0x00000020:    MOV     r3,#7
    0x00000024:    MOV     r0,#0xa
    0x00000028:    BL      func1  ; 0x0
    0x0000002c:    MOV     r4,r0
    0x00000030:    MOV     _____________
    0x00000034:    MOV     _____________
    0x00000038:    BL      func2
    0x0000003c:    MOV     _____________
    0x00000040:    MOV     _____________
    0x00000044:    ADD     r1,r13,#4
    0x00000048:    ADD     r0,pc,#0xc ; #0x5c
    0x0000004c:    BL      _printf
    0x00000050:    MOV     _____________
    0x00000054:    ADD     r13,r13,#0x18
    0x00000058:    ____________________
$d
    0x0000005c:    DCD     1970496850
    0x00000060:    DCD     975205484
    0x00000064:    DCD     544417056
    0x00000068:    DCD     622879781
    0x0000006c:    DCD     2660
```