National Sun Yat-Sen University
ASSEMBLY LANGUAGE AND MICROCOMPUTER
Midterm Exam
9:20-11:20PM April 17 2009

Name: _____

Note: Although there are total 110 points for this exam, the maximum score you can get is 100 points.

1. For the 16-bit MU0 instruction set shown in the right table, ***(16 pts)***

(a) Find out the number of cycles it takes to execute the following code until reaching the ***STP*** instruction. (Assume the code is placed in the memory starting from address #0, and the values stored in memory address #6 and #7 are 10 and 1, respectively.

```
LDA #6;
SUB #7;
JNE #1
STP
```

Instruction	Opcode	Effect
LDA S	0000	ACC := mem_{16}[S]
STO S	0001	mem_{16}[S] := ACC
ADD S	0010	ACC := ACC + mem_{16}[S]
SUB S	0011	ACC := ACC - mem_{16}[S]
JMP S	0100	PC := S
JGE S	0101	if ACC >= 0 PC := S
JNE S	0110	if ACC !=0 PC := S
STP	0111	stop

(b) Show the details of the binary contents for the first eight addresses of the memory.

2. Specify a single ARM instruction which implements the following equation: ***(12 pts)***

(a) r0 = r2-8*r1 (b) mem[r1+4]=r2;

(c) $r1 = r6_{[7:0]}$; (d) r1 = r2 * 17

3. Suppose r1=0xF0000001, r2=0xF0000000 and C=1, N=0, Z=0, V=0, find out the resulting **r1** value of the following instructions. You should also provide the resulting conditional code value (**C N Z**). ***(20 pts)***

(a) ADDC r1, r1, r2 (b) SBCS r1, r1, r2

(c) CMP r1, r2 (d) EOR r1, r2, r2.

4. Suppose r0=0x80010, r1=0x8001c, r2=6, r3=7, r4=8, and the following table shows some part of the memory contents. ***(16 pts)***

Memory address	Data
0x00080020	0x00000005
0x0008001c	0x00000004
0x00080018	0x00000003
0x00080014	0x00000002
0x00080010	0x00000001
0x0008000c	0x00000000

(a) Find out the value of r0 and r1 after the execution of the following instruction.

(i) LDR r0, [r1, #4] ! ***(4 pts)***

(ii) LDR r0, [r1], #4 ***(4 pts)***

(b) Find out the value of r0 and the memory content after the execution of the following instructions: ***(5 pts)***

```
STMIA   r0!, {r2, r4, r3}
```

(c) Following the ***STMIA*** instruction of (b), write down an ARM instruction to restore the value from memory to registers r2, r3, and r4. ***(3 pts)***

5. Explain the following two popular addressing modes: (1) Immediate (2) Base plus offset. You should provide an ARM instruction as an example for each mode. ***(8 pts)***

6. Answer for the following short questions: ***(16 pts)***
   (a) List two ARM exception types. Discuss how to invoke each of two listed exceptions. ***(8 pts)***
   (b) Which two of the 16 registers in ARM are used to be the program counter and link register. ***(4 pts)***
   (c) Write down an ARM instruction for the blank box in the following code such that this code can provide the function of "jump table". ***(4 pts)***

```
            BL      JUMPTAB
            ..
JUMPTAB     ADR     r1, SUBTAB
            CMP     r0, #SUBMAX
            [                ]
            B       ERROR
SUBTAB      DCD     SUB0
            DCD     SUB1
            DCD     SUB2
```

7. For the following ARM codes in the memory with a little-endian ordering, find out the value of ***r0***, ***r1***, ***r2***, and ***r3*** after the program executes the instruction at ***pc=0x1014***. ***(12 pts)***

Memory address (in hexadecimal format)		Instruction
1000		ADR r1, TEXT
1004		LDRB r0, [r1], #1
1008		LDR r2, VALUE
100C		MOV r14,r1
1010		BL SUB1
1014	SUB1	ADD r3, r14, #2
.....	.....	.....
3000	VALUE	DCD &87654321
3004	TEXT	DCD &12345678

8. Modify the following program to output ***r1*** in binary format. For the value loaded into ***r1*** in the example program, you should get 00010010001101000101011001111000. ***(10 pts)***

```
        AREA    Hex_Out,CODE,READONLY
SWI_WriteC      EQU     &0          ; output character in r0
SWI_Exit        EQU     &11         ; finish program
        ENTRY                       ; code entry point
        LDR     r1, VALUE           ; get value to print
        BL      HexOut              ; call hexadecimal outpu
        SWI     SWI_Exit            ; finish
VALUE   DCD     &12345678           ; test value
HexOut  MOV     r2, #8              ; nibble count = 8
LOOP    MOV     r0, r1, LSR #28     ; get top nibble
        CMP     r0, #9              ; 0-9 or A-F?
        ADDGT   r0, r0, #"A"-10     ; ASCII alphabetic
        ADDLE   r0, r0, #"0"        ; ASCII numeric
        SWI     SWI_WriteC          ; print character
        MOV     r1, r1, LSL #4      ; shift left one nibble
        SUBS    r2, r2, #1          ; decrement nibble count
        BNE     LOOP                ; if more do next nibble
        MOV     pc, r14             ; return
        END
```