EE 3381: Microprocessors and Embedded Systems

Spring 2012 Second Exam (April 24, 2012)

Instructions: Read carefully before beginning.

- The time for this exam is 1 hour and 20 minutes.
- You may only refer to the ARM Instruction Set Quick Reference Card. You may NOT use your textbooks, lecture notes, lab assignments, lab solutions, a computer, or phone.
- During the exam, you may not contact Professor Camp, even for clarifications: Interpret problems as best as you can and explain your interpretations/assumptions.
- Write your answers on the exam pages provided, using front and back if needed. There is an additional page at the end of the exam for reference that may be used for scratch work, if necessary.
- This exam is covered by the SMU Honor Code. Please write and sign the following pledge: On my honor, I have neither given nor received any unauthorized aid on this examination.
- Good luck!

PLEDGE:

Signature: ______________________________

Name (Printed): ______________________________

Problem	Score	Total Possible
1		10
2		10
3		6
4		14
5		20
6		20
7		20
Total		100

1. (10 pts) Translate the following conditions into a single ARM instruction (a condition flag state table is on the last page for your reference):

   a. Multiply registers r7 and r12, putting the results in register r3 only if C is set and Z is clear.

   b. Compare registers r6 and r8 only if Z is clear.

2. (10 pts) Assume that memory and registers r0 through r3 appear as follows:

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

   Describe the memory and register contents after executing the instruction

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

3. (6 pts) What mode or modes does the processor have to be in to move the contents of the SPSR to the CPSR?

4. (14 pts) Read the following code and describe what it does. Do not simply interpret line for line, but rather come up with the big picture of the operation the code performs.

```
       AREA   Prob4, CODE, READONLY
num    EQU    32
       ENTRY

start  LDR    r0, =Values
       LDR    r1, =Temp
       MOV    r2, #num
sub1   LDR    r3, [r0], #4
       STR    r3, [r1], #4
       SUBS   r2, r2, #1
       BNE    sub1

       LDR    r0, =Values
       MOV    r2, #num
       SUB    r1, r1, #4

sub2   LDR    r3, [r1], #-4
       STR    r3, [r0], #4
       SUBS   r2, r2, #1
       BNE    sub2

Done   B      Done

       AREA   Block, DATA, READWRITE
Values DCD    1, 2, 4, 5, 6, 7, -8, 11, 23, 5
       DCD    4, 5, 6, 2, 3, 4, 5, 13, 45, 7
       DCD    3, 4
Temp   DCD    0, 0, 0, 0, 0, 0, 0, 0, 0, 0
       DCD    0, 0, 0, 0, 0, 0, 0, 0, 0, 0
       DCD    0, 0

       END
```

What does this code do?

5. (20 pts) Write the division routine below as a subroutine that uses full descending stacks. Pass the subroutine arguments using the stack, and pass the parameters via the stack in such a way where 142 is being divided by 7. For full credit, you may not use FD in the STM/LDM commands, but rather: IA, IB, DA, or DB.

   The following is the original code, to be changed to passing on the stack.

```
          AREA    Division, CODE, READONLY
____________________________________  ; EQU for initial location of stack
Rcnt      RN      0                   ; assign r0 to Rcnt
Ra        RN      1                   ; assign r1 to Ra (dividend)
Rb        RN      2                   ; assign r2 to Rb (divisor)
Rc        RN      3                   ; assign r3 to Rc (result)
Rd        RN      4                   ; assign r4 to Rd (remainder)
          ENTRY
____________________________________  ; Set up stack pointer
____________________________________  ; Set value for dividend (Ra)
____________________________________  ; Set value for divisor (Rb)
____________________________________  ; Put both on stack
____________________________________  ; Call Division subroutine
____________________________________  ; Pop result/remainder into Rc/Rd
____________________________________  ; End program after subr. call

Division   _________________________  ; Save off scratch/link registers
____________________________________  ; Load dividend from stack
____________________________________  ; Load divisor from stack
          MOV     Rcnt, #1            ; Bit to control the division
Div1      CMP     Rb, #0x80000000     ; move Rb until greater than Ra
          CMPCC   Rb, Ra
          MOVCC   Rb, Rb, LSL #1
          MOVCC   Rcnt, Rcnt, LSL #1
          BCC     Div1
          MOV     Rc, #0
Div2      CMP     Ra, Rb              ; Test for possible subtraction
          SUBCS   Ra, Ra, Rb          ; Subtract if OK
          ADDCS   Rc, Rc, Rcnt        ; Put relevant bit into result
          MOVS    Rcnt, Rcnt, LSR #1  ; Shift control bit
          MOVNE   Rb, Rb, LSR #1      ; Halve unless finished
          BNE     Div2                ; Result into Rc, remainder in Ra
_____________________________________ ; Store result into second to last word of stack
_____________________________________ ; Store remainder into last word of stack
_____________________________________ ; Restore scratch registers and PC
          END
```

Fill in the blanks above to complete the code.

6. (20 pts) Write a SWI handler that accepts the number 0xACE. When the handler sees this value, it should reverse the nibbles in register r7. The SWI exception handler should examine the actual SWI instruction in memory to determine its number. Be sure to set up a stack pointer in supervisor mode before handling the exception.

   Format for the code:

   - Any EQU statements to initialize RAM, mode, or I/F bits
   - Enter supervisor mode
   - Initialize stack pointer in supervisor mode
   - In handler, store off relevant registers and SPSR to stack
   - Get SWI number and check to see if it equals 0xACE
   - If so, call the subroutine to reverse nibbles
   - Implement reverse nibbles
   - When subroutine completes, it returns to handler main body
   - Restore SPSR and registers

7. (20 pts) The following code will configure the UART and put one byte into the UART for transmitting. Rewrite the following code (by changing it only where necessary) for using full descending stacks. To receive full credit, do not use FD with the stores and loads, but rather the appropriate increment before (IB), increment after (IA), decrement before (DB), or decrement after (DA). Also, change the UART transmission to 6 bits, with even parity, and two stop bits.

```
           AREA     UARTDm, CODE, READONLY
PINSEL0    EQU      0xE002C000          ; Controls the function of the pins
U0START    EQU      0xE000C000          ; Start of UART0 registers
LCR0       EQU      0xC                 ; Line control register for UART0
LSR0       EQU      0x14                ; Line status register for UART0
RAMSTART   EQU      0x40000000          ; Start of onboard RAM for 2104
           ENTRY

start      LDR      sp, =RAMSTART       ; Set up stack pointer
           BL       UARTConfig          ; Initialize/configure UART0
           LDR      r1, =CharData       ; Starting address of characters

Loop       LDRB     r0, [r1], #1        ; Load character, increment address
           CMP      r0, #0              ; Null terminated?
           BLNE     Transmit            ; Send character to UART
           BNE      Loop                ; Continue if not a '0'
Done       B        Done                ; Otherwise, we're done

; Subroutine UARTConfig
; Configures the I/O pins first and sets up the UART control register.
; The parameters are currently set to 8 bits, no parity, and 1 stop bit.
UARTConfig STMIA    sp!, {r5, r6, lr}
           LDR      r5, =PINSEL0        ; Base address of register
           LDR      r6, [r5]            ; Get contents
           BIC      r6, r6, #0xF        ; Clear out lower nibble
           ORR      r6, r6, #0x5        ; Sets P0.0 to Tx0 and P0.1 to Rx0
           STR      r6, [r5]            ; Read/Modify/Write back to register
           LDR      r5, =U0START
           MOV      r6, #0x83           ; Set 8 bits, no parity, 1 stop bit
           STRB     r6, [r5, #LCR0]     ; Write control byte to LCR
           MOV      r6, #0x61           ; 9600 baud @15MHz VPB clock
           STRB     r6, [r5]            ; Store control byte
           MOV      r6, #3              ; Set DLAB = 0
           STRB     r6, [r5, #LCR0]     ; Tx and Rx buffers set up
           LDMDB    sp!, {r5,r6,pc}
```

```
; Subroutine Transmit
; This routine puts one byte into the UART for transmitting.
Transmit   STMIA    sp!, {r5, r6, lr}
           LDR      r5, =U0START
wait       LDRB     r6, [r5, #LSR0]     ; Get status of buffer
           CMP      r6, #0x20           ; Buffer empty?
           BEQ      wait                ; Spin until buffer's empty
           STRB     r0, [r5]
           LDMDB    sp!, {r5, r6, pc}

CharData   DCB      "Watson. Come quickly!", 0
           END
```

Opcode [31:28]	Mnemonic extension	Meaning	Condition flag state
0000	EQ	Equal	Z set
0001	NE	Not equal	Z clear
0010	CS/HS	Carry set/unsigned higher or same	C set
0011	CC/LO	Carry clear/unsigned lower	C clear
0100	MI	Minus/negative	N set
0101	PL	Plus/positive or zero	N clear
0110	VS	Overflow	V set
0111	VC	No overflow	V clear
1000	HI	Unsigned higher	C set and Z clear
1001	LS	Unsigned lower or same	C clear or Z set
1010	GE	Signed greater than or equal	N set and V set, or N clear and V clear (N == V)
1011	LT	Signed less than	N set and V clear, or N clear and V set (N != V)
1100	GT	Signed greater than	Z clear, and either N set and V set, or N clear and V clear (Z == 0,N == V)
1101	LE	Signed less than or equal	Z set, or N set and V clear, or N clear and V set (Z == 1 or N != V)
1110	AL	Always (unconditional)	-
1111	-	See *Condition code 0b1111*	-