ECE 5362 Homework 3
Due 10:20am Sept. 18 (Carmen PDF Submission)

1. (20 points, 10 pts each) For the following program, what is the content of R2 (in hexadecimal) at the end of execution: a) if this is executed on a byte-addressable, big-endian machine with a word size of 4 bytes (32 bits)? b) if it is executed on a byte-addressable, little-endian machine with a word size of 4 bytes (32 bits)? The instruction format is "Move destination, source". Use XX for bytes with unknown value. Note that you do not need to know the value of DATA for this problem. Assume this machine allows you to write/read memory words across their boundaries.

```
Move R3, #DATA
Move 2(R3), #$FFEEDDCC
Move (R3)+, #$1A2B3C
Load  R2, -1(R3)
```

2. (40 points) Given the following 68000 program, answer the two questions below.

```
            MOVEA.L     #NUMBERS, A0
            MOVE.L      SIZE, D0
            MOVEA.L     A0, A1
            ADDA.L      D0, A1
LOOP1:      ADDA.L      #−1, A1
            CMPA.L      A0, A1
            BLS         END
            MOVE.B      (A1), D1
            MOVEA.L     A1, A2
            ADDA.L      #−1, A2
LOOP2:      MOVE.B      (A2), D2
            CMP.B       D2, D1
            BGE         SKIP
            MOVE.B      D2, (A1)
            MOVE.B      D1, (A2)
            MOVE.B      D2, D1
SKIP:       ADDA.L      #−1, A2
            CMPA.L      A0, A2
            BHS         LOOP2
            JMP         LOOP1
END:        next instruction
```

   a) (10 pts) What does this program do? Note "CMP.B" compares a byte and "CMPA.L" compares a long word of address, but both will NOT change the contents of the registers. For example, "CMP.B A, B" performs subtraction [B]-[A] and sets the flag Z or N to 1 if the result is zero or negative. BGE (branch on greater than or equal) compares **signed** numbers and branches when N=0. BHS (branch on higher than or same) is similar to BGE but does **unsigned** comparison. BLS (branch on lower than or same) is also **unsigned** comparison and branches when N=1 or Z=1. Read the 68000 Instruction Set handout to know more about how those signed/unsigned comparisons and branches should work.

   b) (30 pts in total) Assume that NUMBERS = $A00000 is the starting memory address of three consecutive 16-bit data words: 1234, 5362, -666 (all in decimal). Memory word location SIZE contains the number of 6. What are the contents (in hexadecimal) of three memory words (starting from NUMBERS) after the execution of the program (3 pts for each byte)? What are the contents of A1, A2, D1, and D2 (3pt each)? Please assume 2's complement for negative numbers. Also note that BGE compares **signed** numbers. For example, 0x06 is greater than 0xFE if they are interpreted as signed numbers. If you only know the content of a certain byte in a register, just say which byte has what content.

3. (40 points, 10 pts each) Given the below program of the RISC machine with a 32-bit word size, what are the stack contents and the contents of the stack pointer (SP), in hexadecimal, immediately after each of the following instruction of the program is executed? Assume that [SP] = 0xFF00 at Level 1, before the execution of the calling program begins. Assume the contents of R5, R6, and R7 are initially 0x1A, 0x2D, and 0x3F, respectively. Memory word location SIZE contains the number of 8. ONE = 0xA000 is the starting memory address of two consecutive 32-bit data words: 0x5362ECEF and 0x66150914. TWO = 0xB000 is the starting memory address of two consecutive 32-bit data words: 0x5342ECEF and 0x66160918. Note for the stack contents, also show those contents (if any) that have been previously popped out of the stack. Instruction *LoadByte R6, LOC* is used to load a memory byte at LOC to the least significant (rightmost) byte of register R6. *LoadByte* loads a byte from the memory into the rightmost eight bit positions of a 32-bit processor register and clears the remaining higher-order bits to zero. Assume we do not use stack for storing return address (i.e., PC).
   a) The first Store instruction in the calling program.
   b) The second Store instruction in the subroutine.
   c) The last Store instruction in the subroutine.
   d) The last Add instruction in the subroutine.

```
                Load            R2, SIZE
                Move            R3, #ONE
                Move            R4, #TWO
                Subtract        SP, SP, #4
                Store           R0, (SP)
                Call            SUBROUT
                next instruction

SUBROUT:        Subtract        SP, SP, #12
                Store           R5, 8(SP)
                Store           R6, 4(SP)
                Store           R7, (SP)
                Load            R5, 12(SP)
                Load            R8, (R3)
LOOP:           LoadByte        R6, (R3)
                LoadByte        R7, (R4)
                Branch if [R6]=[R7] SKIP
                Add             R5, R5, #1
                LoadByte        R8, (R3)
SKIP:           Add             R3, R3, #1
                Add             R4, R4, #1
                Subtract        R2, R2, #1
                Branch if [R2]>[R0] LOOP
```

```
Store           R8, 16(SP)
Store           R5, 12(SP)
Load            R7, (SP)
Load            R6, 4(SP)
Load            R5, 8(SP)
Add             SP, SP, #12
Return
```