



Electrical and Computer Engineering Department
 University of Maryland
 College Park, MD 20742-3285

Glenn L. Martin Institute of Technology ♦ A. James Clark School of Engineering

Dr. Charles B. Silio, Jr.
 Telephone 301-405-3668
 Fax 301-314-9281
 silio@eng.umd.edu

ENEE 350 Homework Set No. 9

(Due: Class 22, Wed., Jul. 11, 2007)

and

Programming Project 4

(Due: Class 25, Tues., Jul. 17, 2007)

1. Read Appendix B of text by A. Tanenbaum, *Structured Computer Organization*, 5th ed., Prentice-Hall, 2006, and work the following problems from Appendix B:
 - a. Problem B-1.
 - b. Problem B-2.
 - c. Problem B-3.

2. What sign-magnitude decimal values are represented by the following IEEE 754 single precision floating-point words whose contents are shown using hexadecimal shorthand? (Hint: use C-compiler and formatted output to save yourself from doing considerable work.)

- | | |
|-------------|-------------|
| a. B9EBEDFA | c. 40490FDB |
| b. 7F800000 | d. FF83FD03 |

3. Print out and read the handout on floating point representations by C. Silio in course website file www.ece.umd.edu/class/enee350.S2007/Notes/fltnngpt.ps or [fltnngpt.pdf](#).
4. The designers of a particular computer have decided that the computer must be capable of representing single-precision (single-word) floating-point numbers in the range $\pm(10^{-17}$ to $10^{17})$ with a precision of one part in 10^5 . Determine the minimal binary word length which must be chosen for this machine, and indicate the floating-point format you would choose for doing this in order to facilitate the sorting of floating-point numbers. (Assume that $2^{10} = 10^3$ to facilitate decimal to binary conversions.)
5. Recall from your reading of Silio's notes on floating-point representations that the UNIVAC 1100 series computers have 36-bit words and perform 1's complement arithmetic. Suppose UNIVAC 1100 registers A1 and A2 contain the following bit patterns in octal shorthand.

(A1) = 572053777777 (A2) = 206556400000

Viewing the contents of A1 and A2 as single-precision floating-point numbers:

- a. What sign-magnitude decimal number is contained in A1?
 - b. What sign-magnitude decimal number is contained in A2?
6. In a DEC PDP-11 the contents of two consecutive memory words are (in binary):

1011111111010000 0000000000000000

Recall that the single-precision floating-point format for this machine is of the form: 1+8+23(24) bits with a binary normalized mantissa 0.1xxx as in

1	8	23
S _M	BIASED EXPONENT	BINARY NORMALIZED MANTISSA

If this 32-bit pattern is interpreted as a single-precision floating-point number, what sign-magnitude decimal number does it represent?

-continued-

7. The IBM 360/370 series computers use a sign-magnitude hexadecimally normalized, biased-exponent, 32-bit representation for single precision floating-point numbers in the following format:

1	7	24
S _M	BIASED EXPONENT	HEXADECIMALLY NORMALIZED MANTISSA

Write the 8-digit hexadecimal representation of the bit pattern in the 32-bits known to contain the single-precision representation of the following floating-point number shown here in both its decimal and octal forms:

$$-(27\frac{2}{13})_{10} = -(33.1166116611661166...)_{8}$$

8. Consider the following biased exponent (bias = 2⁵), sign-magnitude floating point format for representing binary normalized numbers in single-precision words in a machine with 2's complement fixed-point arithmetic; the mantissa (significand) is a binary normalized fraction, and there are no hidden bits:

1	6	7
S _M	BIASED EXPONENT	BINARY NORMALIZED MANTISSA

Suppose we are given the following two operands represented in this format:

$$X = 1\ 000010\ 1010001 \quad Y = 0\ 000101\ 1100110$$

Show the bit pattern in the single-precision word S that results from the floating add of the contents in X and Y, assuming that the result is truncated to a 7-bit precision fraction.

9. **Programming Project 4 (Due: Class 25, Wed., Apr. 25, 2007):** Consider the following biased exponent (bias = 2⁶), sign-magnitude floating point format for representing binary normalized numbers in 16-bit single-precision words in a machine with 2's complement fixed-point arithmetic; the mantissa (significand) is a binary normalized mixed number with hidden bit similar to IEEE754.

1	7	8
S _M	BIASED EXPONENT	BINARY NORMALIZED SIGNIFICAND

For example, the following two operands represented in this format:

$$A = 1\ 0000010\ 10100011 \quad B = 0\ 0000101\ 11001100$$

where $A = 0x82A3 = -1.10100011 \times 2^{-62}$ and $B = 0x05CC = +1.11001100 \times 2^{-59}$.

- Making use of the MAC-2 instruction repertoire and the `inv(x)` function you wrote and tested in programming assignment 3, write and test a procedure (i.e., a function subprogram) **or(x,y)** that computes the bit-wise logical OR of the n-tuples x and y. The arguments are passed by reference, with address y pushed on the stack first followed by address x pushed on the stack followed by a call to function **or**, which returns the value computed in the ac register (return by value).
- Making use of the MAC-2 instruction repertoire, write a (void function) procedure **ashr(x)** that performs a 1-bit position arithmetic (algebraic) right shift of the contents of memory location x and leaves the result in memory location x, where the address x is passed by reference on the stack.
- Again, making use of the MAC-2 instruction repertoire and whatever other functions (such as the OR function and procedure `ashr(x)` from parts a.) and b.) write and test a procedure (i.e., a function subprogram) **fadd(x,y)** that performs a floating add of single-precision floating point numbers in memory locations x and y and returns the single-precision floating-point format result in the ac register, where all single-precision floating point numbers are represented in the format specified above in Problem 9. Again, the arguments are passed by reference, with address y pushed on the stack first followed by address x pushed on the stack followed by a call to function **fadd**, which returns the value computed in the ac register (return by value).
- Test your **fadd** function using the following main program (**prg4main**):

Repair the following main program, if necessary, to accomplish the desired results as stated in the comments.

```

    /prg4main
    EXTRN  inv
    EXTRN  or
    EXTRN  fadd
x1      0x7D5C
x2      0x7A33
x3      0x0b98
x4      0x02A3
ans1    RES    1
ans2    RES    1
ans3    RES    1
ans4    RES    1
ans5    RES    1
ans6    RES    1
start   loco   4020
        swap
        loco   x1
        push
        call   inv
        stod  x1    /create data x1=0x82A3
        stod  ans1
        loco  ans1
        push
        call  ashr  /make sure ashr is working
        insp  1
        loco  x2
        push
        call  inv
        stod  x2    /create data x2=0x85CC
        call  or
        stod  ans2  /make sure OR is working
        call  fadd
        stod  ans3  /ans3=fadd(x1,x2)
        loco  x3
        stol  0
        call  ashr  /ashr shifts x3 right arthimetically
        call  fadd
        stod  ans4  /ans4=fadd(x1,x3)
        loco  x4
        stol  1
        call  fadd
        stod  ans5  /ans5=fadd(x3,x4)
        loco  x2
        stol  0
        call  fadd
        stod  ans6  /ans6=fadd(x2,x4)
        insp  2
        halt
        END    start

```