CSC201, SECTION 002, Fall 2000: Homework Assignment #2

DUE DATE

Monday, October 2, at the start of class.

INSTRUCTIONS FOR PREPARATION

- Neat, in order, answers easy to find.
- Staple the pages together at the upper left corner.
- Fold lengthwise. On the outside write the course number, the assignment number, the date, and your name.

Thanks for your help.

PROBLEMS

1. (3) Convert the following decimal numbers into hexadecimal and binary: (example: 17 decimal = 11h, or 10001b)
    1. 20
    2. 201
    3. 2021

2. (3) Convert the following hexadecimal or binary numbers into decimal:
    1. 0Ah
    2. 2Fh
    3. 1001001001b

3. (5) Using 5 bits for each, show the unsigned, signed-magnitude, and twos-complement representations for:
    1. 0

2. +16
3. +3
4. -5
5. -16

Just say "can't be done" for cases that can't be represented.

4. (3)
    1. What SASM macro is used to extend the precision of a twos-complement 8-bit number to 32 bits?
    2. If the unsigned binary number 1010 is extended to 8 bits (with zero fill), what is the 8-bit result in binary, and what does this represent, in decimal?
    3. If the signed twos-complement number 1010 is extended to 8 bits (with sign extension), what is the 8-bit result in binary, and what does this represent in decimal?

5. (2)
    1. Convert the binary fraction .101 into decimal.
    2. Convert the decimal fraction .205 into binary.

6. (3) Normalize the following floating point numbers:

Sign of mantissa	Exponent (in decimal)	Mantissa (in binary)
1	+8	111.001
0	-2	0.00011101
0	-100	1100110.0

7. (2) What do the following floating point numbers represent in decimal?

Sign of mantissa	Exponent (in decimal)	Mantissa (in binary)
0	-4	1.0111
1	+6	1.0010

8. (2)
    1. In decimal, what is the maximum positive value for the exponent in the single-precision floating-point standard, and the minium negative value?
    2. In decimal, what is the maximum positive value for the mantissa in the single-precision floating-point standard, and the minimum non-zero positive value?

9. (2)
    1. Write the decimal value -10.5 in binry FPS representation (but represent the exponent in decimal).
    2. Write the decimal value +.008 in binary FPS representation (but represent the exponent in decimal).

10. (5) For each of the following, assume op1 = 10100011b and op2 = 11000101b. Ignore the fact that SASM logical and arithmetic operations work on doublewords instead of byte-length operands. What will be the new value of op1 after executing the instruction:
    1. lnot op1
    2. lneg op1
    3. land op1, op2
    4. lor op1, op2
    5. lxor op1, op2

11. (3)
    1. Write the SASM code to clear bits 3-0 (least significant 4 bits) of a doubleword operand labeled "op1".
    2. Write the SASM code to set bits 7-4 of operand "op1".
    3. Write the SASM code to insert the value '0111' into bits 11-8 of operand "op1".

12. (4) Ignore the fact that shifting is defined for doublewords, rather than bytes, in SASM.
    1. If the value of op1 = 10000101b, what will be the new binary value of op1 after executing the instruction "rlsh op1"?
    2. If the value of op1 = 10000101b, what will be the new binary value of op1 after executing the instruction "rash op1"?
    3. If the value of op1 = 10000101b, what will be the new binary value of op1 after executing the instruction "rrot op1"?
    4. If the value of op1 = 10000101b, what will be the new binary value of op1 after executing the instruction "llsh op1"?

13. (3)
    1. Suppose op1 is a twos-complement (signed) binary number with value 10000100b. What is this value in decimal, and (also in decimal) what is the new value of op1 after executing "rash op1"?
    2. Suppose op1 is a twos-complement (signed) binary number with value 00001101b. What is this value in decimal, and (also in decimal) what is the new value of op1 after executing "rash op1"?
    3. Suppose op1 is an unsigned binary number with value 10000100b. What is

this value in decimal, and (also in decimal) what is the new value of op1 after executing "llsh op1"?

14. (2)
    1. What is the result, in binary, of multiplying the unsigned binary numbers 1001b and 0111b?
    2. What are the results (quotient and remainder), in binary, of dividing the unsigned binary number 100011b by 000110b?

15. (3) Do addition of the following 4-bit unsigned operands, show the 4-bit sum, and indicate when overflow occurs.
    1. 0110b + 0101b
    2. 0011b + 0100b
    3. 1010b + 0110b

16. (4) Do the indicated operation (addition or subtraction) of the following 4-bit twos-complement (signed) operands, show the 4-bit result, and indicate when overflow occurs.
    1. 0110b + 0101b
    2. 0101b - 0110b
    3. 1001b - 0010b
    4. 1110b + 1111b

17. (3) Show the normalized floating point representation (decimal exponent, 4-bit mantissa (including the bit to the left of the binary point)) of the results of doing the following floating point operations:

First operand (sign, exponent in decimal, mantissa)	operation	Second operand (sign, exponent in decimal, mantissa)
0, +5, 1.110	+	0, +2, 1.010
1, +5, 1.110	*	0, +3, 1.010
0, +5, 1.110	/	0, +3, 1.010

PROGRAMMING

1. (30 points) Write a SASM program (named "hw2a.asm") that reads in character strings and determines if each is a syntactically valid decimal real number. The

syntax rules for real numbers are as follows:

1. A real number has two parts: a mantissa, which is mandatory, and an exponent, which is optional.
2. The mantissa has 4 parts: a sign, an integer part, a decimal point, and a fractional part.
    1. The sign, which is optional, must be either + or -
    2. One or both of the integer part and the fractional part must be present.
    3. The decimal point separates the integer and fractional parts. If there is only an integer part, there must not be a decimal point. If there is only a fractional part, there must be a decimal point.
    4. The integer part and the fractional part have the same syntax. Each consists of one or more occurrences of the decimal digits 0--9.
3. The exponent consists of 3 parts: the exponent indicator, the sign, and an integer part.
    1. The exponent indicator, which is required, must be the letter E or the letter e.
    2. The sign, which is optional, must be either + or -.
    3. The integer part, which is required, consists of one or more occurences of the decimal digits 0--9.

Your program should expect each string to be on a line terminated with LF/CR. Every string will consist entirely of letters, digits, and punctuation marks; there will not be any embedded spaces.

Here is an example of running the program. The input to the program is shown indented for clarity. In the real input, there will not be any indentation.

```
        123E+10
Valid number
        1.2.3E1
Syntax error
        .E
Syntax error
        +-1E3
Syntax error
        0E.3
Syntax error
        .123e-10
Valid number
        -1.23E-3
Valid number
        .351
Valid number
        abc
Syntax error
        +-.
```

```
Syntax error
        -.23E-.23
Syntax error
        1E+15
Valid number
        0
Valid number
        0.0E0
Valid number
```

2. (20) Write a program (named "hw2b.asm") which will read a string representing a single unsigned decimal integer, and print the hex (8 hex digit) and binary (32 binary digit) representations, in ASCII. An example of running the program is the following (input indented for clarity, not actually indented when being tested):

```
        266
= 0000010Ah, 00000000000000000000000100001010b
```

WARNING! Keep all your program files strictly in your EOS locker space, with no access given to other users. Do not put your files on the local machine, such as in temp space! If you do, you are inviting someone else to use your code, since files on the local PC file system are not protected.

Use submit to turn in your 2 programs, one at a time. Be sure your programs include a header comment indicating the assignment, date, and your name. Include a reasonable number of comments in your code as well.

One piece of advice. You may have difficulties with both the logic of the program, and writing the program in SASM. If that is the case, you will find it easier to first write the program in a high-level language, such as C, and make sure you get the logic right. Then manually translate your high-level code into SASM, making sure you get the assembly language programming right.

GRADING

This homework is graded on a scale of 100 points; the points for each problem are shown above. The homework score will be weighted to contribute 5% of your course grade.

Problems will be graded according to the following interpretation:

- All right = full credit
- halfway to mostly right = half credit

- Not much or none right = no credit