CSE2312-002, 003 (Spring 2020)
Homework #4

Notes:

- All numbers are in base-10 unless otherwise noted.
- If part of a problem is not solvable, explain why in the answer area.
- Print out the form and handwrite your answers in the spaces below.
- Place the hw4.s file and the scanned answers to problems 1-6 in a single zip file with name lastname_hw4.zip, where lastname is your last name as listed in MyMav.
- Submit the single zip file to Canvas before 11:59:00pm on April 25, 2020.
- Make sure that the code follows the procedure call standards for ARM architecture (see IHI0042F section 5.1), with emphasis on this requirement: “A subroutine must preserve the contents of the registers r4-r8, r10, r11 and SP (and r9 in PCS variants that designate r9 as v6).” (in other words, push and pop R4-11 if you need to use them, as shown in the vector.s examples in class)

1. Encode the following numbers as single-precision floating point numbers:

a. -4096

s = ___, e = _____, m = _____

32b hex value = _____

b. 0.0625

s = ___, e = _____, m = _____

32b hex value = _____

c. 0.33...

s = ___, e = _____, m = _____

32b hex value = _____

d. 256 + 1/256

s = ___, e = _____, m = _____

32b hex value = _____

e. 1024.0001

s = ___, e = _____, m = _____

32b hex value = _____

2. Assume float x = 2097152.

a. Calculate the smallest positive number that can be added to x that will not be lost in the mantissa.

b. In general, what is the ratio of the large to the smallest single-precision floating point number that can be added together without a loss of accuracy?

3. For the following code, calculate the number of instruction cycles required to execute the following code, using the simplified pipeline timing rules in class, including the time to call this function with BL bro8 and the time to return from the function with BX LR. You can assume that the pipeline is full before the BL bro8 instruction is executed.

```
bro32:

    MOV R1, R0

    MOV R0, #0

    MOV R2, #0x80000000

    MOV R3, #0x00000001

bro32_loop:

    TST R1, R2

    ORRNE R0, R0, R3

    MOVS R2, R2, LSR #1

    MOV R3, R3, LSL #1

    BNE bro32_loop

    BX LR
```

Clock cycles: _____

If the clock rate is 4 GHz, what is the execution time in nanoseconds? _____

4. Assume SP = 0x20001038 before the following instructions are executed:

```
Address     Instruction
10000000:       BL fn
          fn:
10001000:       MOV R0, #1024
10001004:       MOV R1, #0x12345678
10001008:       MOV R2, #1400
1000100C:       MOV R3, #0x8000
10001010:       PUSH {R0, R1, R2, R3, LR}
          loop:
10001014:       B loop
```

After this program enters the endless loop:

What is the value of the SP? _____

Assuming the processor uses big-endian convention, what is the value of the following memory locations (place X in the blank if there is not enough information):

Address	8-bit Data
0x2000103A	_____
0x20001039	_____
0x20001038	_____
0x20001037	_____
0x20001036	_____
0x20001035	_____
0x20001034	_____
0x20001033	_____
0x20001032	_____
0x20001031	_____
0x20001030	_____
0x2000102F	_____
0x2000102E	_____
0x2000102D	_____
0x2000102C	_____
0x2000102B	_____
0x2000102A	_____
0x20001029	_____
0x20001028	_____
0x20001027	_____
0x20001026	_____
0x20001025	_____
0x20001024	_____

5. Explain the concept of memory virtualization, including the concept of paging and fragmentation. Also explain the role of virtualization in memory protection between running processes (“programs”).

6. Explain the concept of cache, including the principle of locality. Explain how this can speed up memory accesses.

7. Write assembly functions that implement the following C functions:

a. void diffF32(float z[], const float x[], const float y[], uint32_t count)
   // compute the difference of each element in the arrays x and y containing
   // count entries; z[i] = x[i] – y[i], for i = 0, 1, … count-1
   // If x = {10, 20, 30}, y = {5, 10, 5}, and count = 3, then z = {5, 10, 25}

b. double prodF32_64(const float x[], uint32_t count)
   // returns the product of the elements in an array (x) containing count entries
   // if x = {1, 2, 3} and count = 3, then the function returns 6

c. float dotpF32(const float x[], const float y[], uint32_t count)
   // returns the dot product of two arrays (x and y) containing count entries

d. double minF64(const double x[], uint32_t count)
   // returns the minimum value in the array (x) containing count entries
   // if x = {-1.1, 20, -3} and count = 3, then the function returns -1.1

   Write the solution of all of these functions in a single file hw4.s with the functions being callable from a C program. You do not need to send the .c file used to test these functions.