CMPT 295

Junchen Li (Richard)

301385486

2020/2/6

Assignment 3

Objectives:

* IEEE floating point number addition and rounding
* Memory addressing modes
* Assembly instructions
* Reading object code (machine level instruction) expressed in hexadecimal and understanding how these instructions are stored in memory
* Writing a C program that corresponds to given assembly program

Submission:

* Submit your document called **Assignment\_3.pdf**, which must include your answers to all of the questions in Assignment 3.
  + Add your full name and student number at the top of the first page of your document **Assignment\_3.pdf**.
* When creating your assignment, first include the question itself and its number then include your answer, keeping the questions in its original numerical order.
* **If you hand-write your answers (as opposed to using a computer application to write them):** When putting your assignment together, do not take photos (no .jpg) of your assignment sheets! Scan them instead! Better quality -> easier to read -> easier to mark!
* Submit your assignment electronically on CourSys

Due:

* Thursday, Feb. 6 at 3pm
* Late assignments will receive a grade of 0, but they will be marked in order to provide the student with feedback.

Marking scheme:

This assignment will be marked as follows:

* + Questions 1, 2 and 5 will be marked for correctness.
  + Questions 3 and 4 will be marked for completeness, i.e., you get marks for completing (answering) the question, but it is up to you to verify the correctness of your answer by looking up the solutions when they are posted.

The amount of marks for each question is indicated as part of the question.

A solution will be posted after the due date.

1. [3 marks] IEEE floating point number addition and rounding

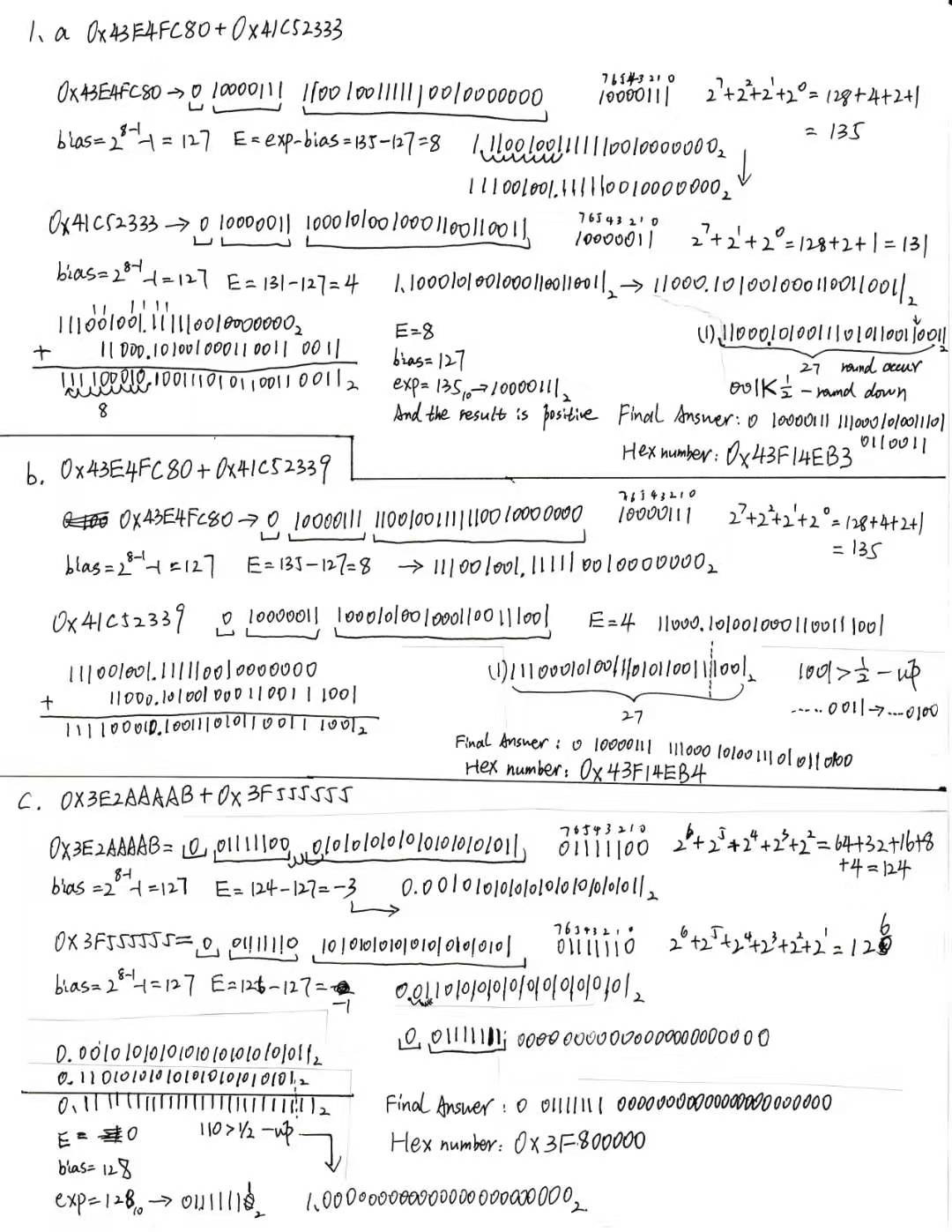
When adding real numbers expressed in scientific notation (base 10), we must first transform them such that they have the same exponent. For example, 3.1416 + 1.0 x 103 must be transformed to 3.1416 + 1000.0. Once the numbers are expressed with the same exponent, we need to align their decimal point, then we can add them (1003.1416 = 1.0031416 x 103).

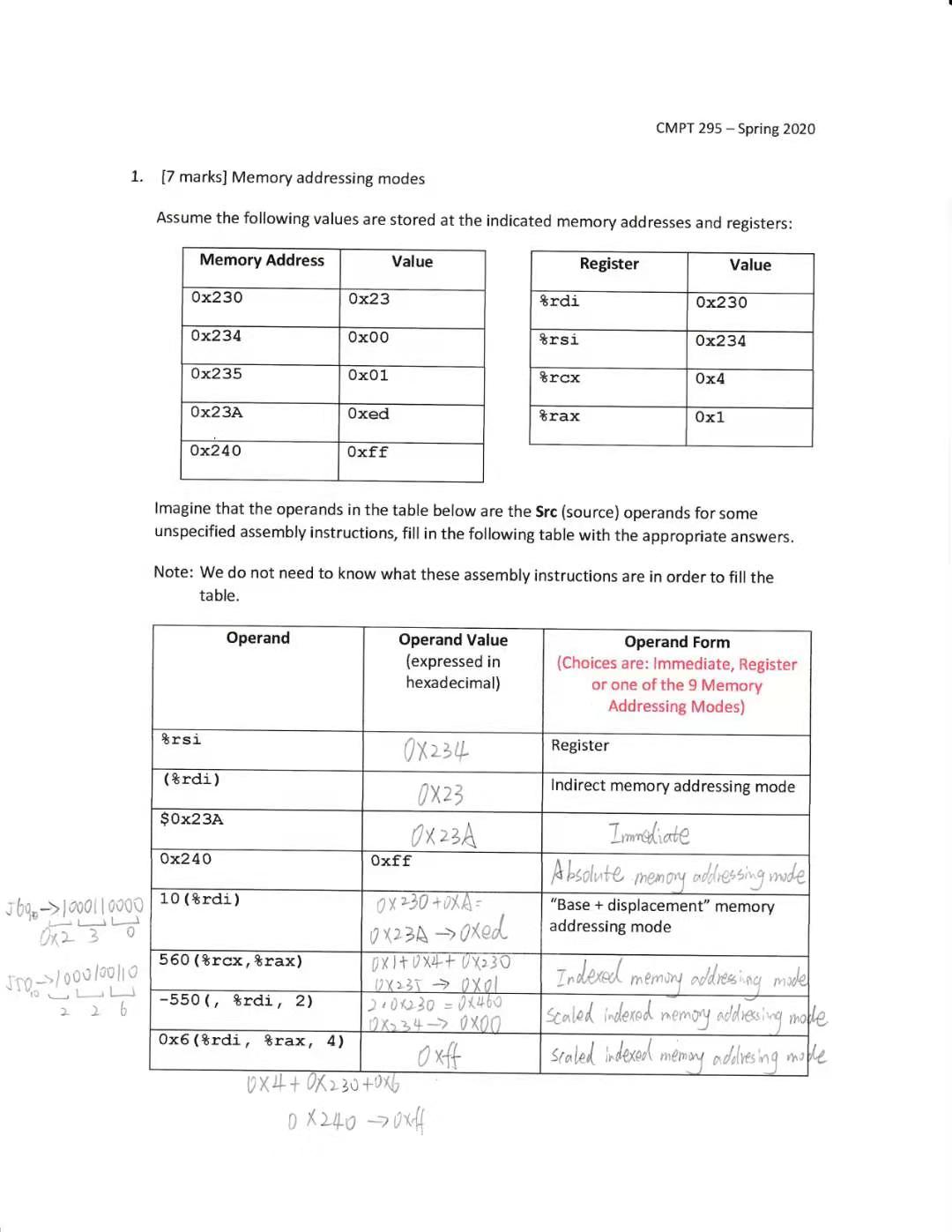
The same is true when adding IEEE floating point numbers, except that the base we are working with is 2.

Perform the following IEEE floating point number additions following the algorithm described above, i.e., first, transform the IEEE floating point numbers (expressed as hexadecimal numbers) such that they have the same exponents, align their binary points and add them. Express their sum as an IEEE floating point number, then express this IEEE floating point number as a hexadecimal number. Show your work and clearly show the result of rounding, if rounding occurs.

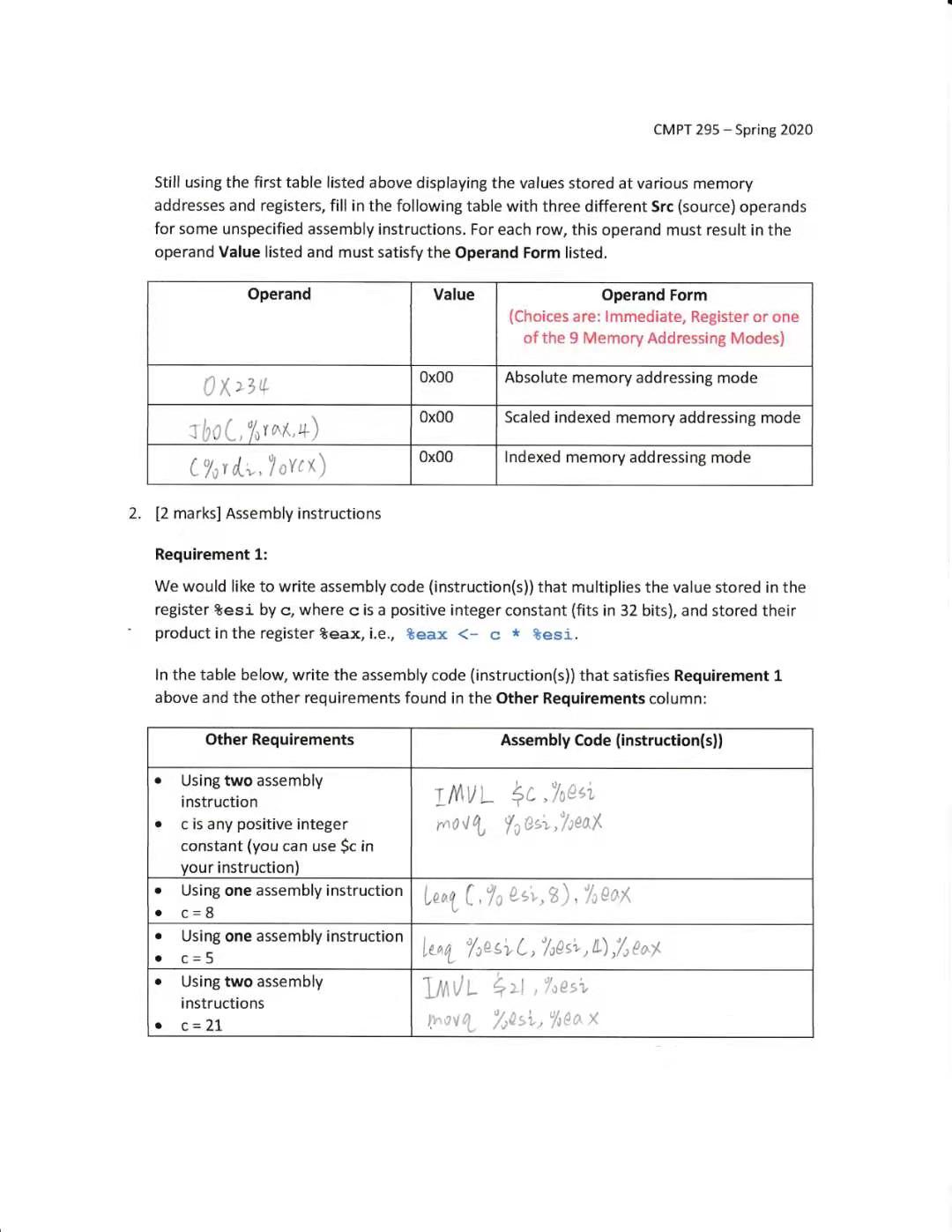
* 1. 0x43E4FC80 + 0x41C52333
  2. 0x43E4FC80 + 0x41C52339
  3. 0x3E2AAAAB + 0x3F555555

where 0.16666667 approximates 0x3E2AAAAB  
 and 0.83333333 approximates 0x3F555555

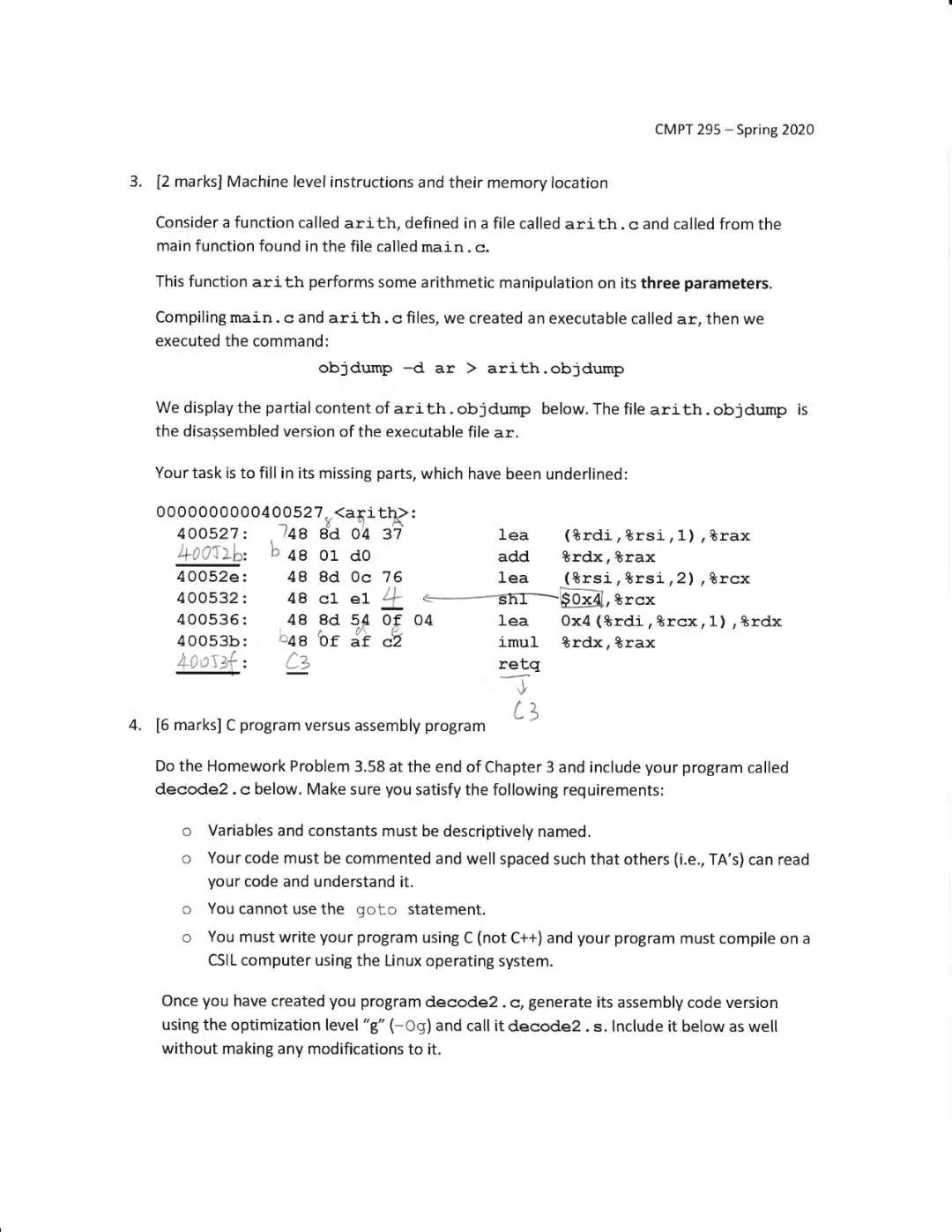




2.



3.



5.

4.

#include <stdio.h>

// x->%rdi y->%rsi z->%rdx

long decode2 (long x,long y,long z)

{

y=y-z; // subq y<-y-z

x=x\*y; // imulq x<-x\*y

long temp1=y; //movq %rax<-y

long temp2=temp1<<63; //salq y<-$63 left shift same as SHL

long temp3=temp2>>63; // sarq y<-$63 right shift

long ans=temp3^x; // xorq %rax<-^x Exclusive-or

return ans; //return

}