Introduction to Computer Systems
Homework 1 – 64bit Arithmetic

2022 Spring, CSE3030

Sogang University

Making a 64-bit Calculator Using two 32-bit

1. Introduction

This assignment aims to become more familiar with the binary representation of integers and understand what happens during arithmetic operations between two integers.

Our goal is to make a 64-bit arithmetic calculator.

Making a 64-bit Calculator Using two 32-bit

- 2. Problem specification

  - 2.1 Overview

    Write two C functions named add64(), sub64() which receive two 32-bit integers (int64.hp, int64.lp) and compute the addition and subtraction of those integers, respectively.

    We give the skeleton code of the 64-bit arithmetic program.
    And also provide function getBit, setBit and printBinaryRepresentation.

    Initially, The program accepts two 64-bit integers (in hex) as accepting 32-bit int four times.
    After that, Program will calculate.

Making a 64-bit Calculator Using two 32-bit

- 2. Problem specification

  - 2.2 Restrictions

    Do not use other C libraries (Library is provided in skeleton code).

    Use the provided struct int64.

Making a 64-bit Calculator Using two 32-bit

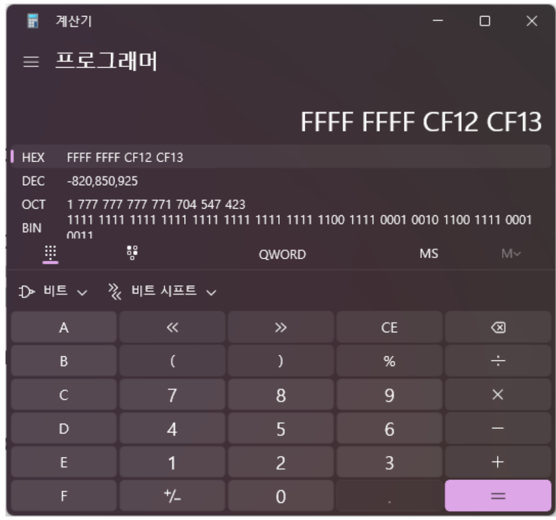
- 2. Problem specification

  - 2.3 What do you have to do?

    1. Roughly describe all functions.
    ex) The getBit function accepts hexadecimal operands in bits.
    ex) The setBit function is …

    2. Complete add64, complement64, and sub64 functions
    hint) The implementation order would be add64, complement64, and sub64.
    hint) add64 : See full-adder and bitwise operation of lecture material 3 (float)
    hint) complement64 : Think about 2's complement.
    hint) sub64 : Use your complete add64 and complement64 function.

    3. Check your calculated results.
    hint) Check your results by referring to the Windows default calculator or
    the screenshots on the next page.

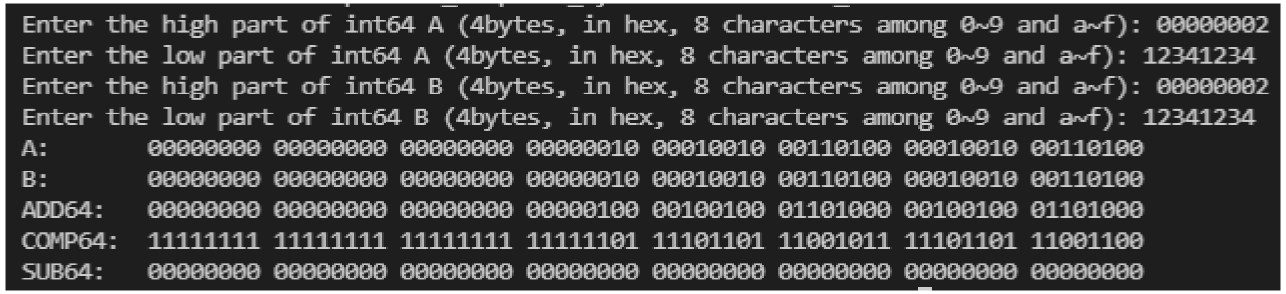


```
FFFF FFFF CF12 CF13
HEX  FFFF FFFF CF12 CF13
DEC  -820,850,925
OCT  1 777 777 777 771 704 547 423
BIN  1111 1111 1111 1111 1111 1111 1111 1111 1100 1111 0001 0010 1100 1111 0001
     0011
```

Making a 64-bit Calculator Using two 32-bit

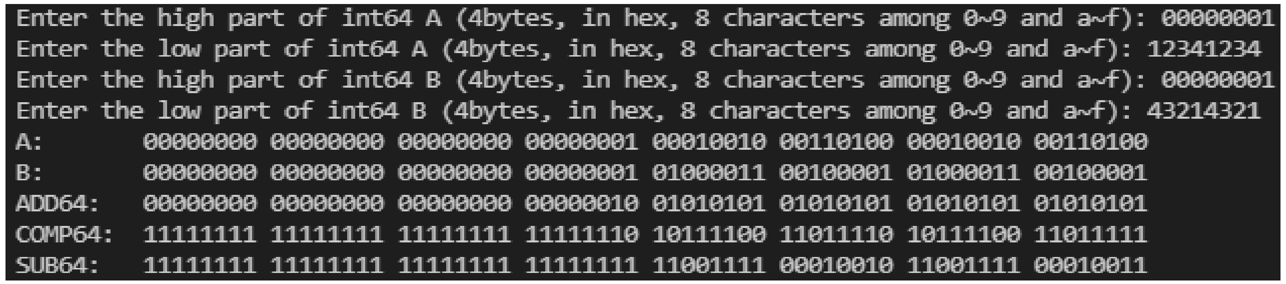
- 3. Example

  Example input for this program is :



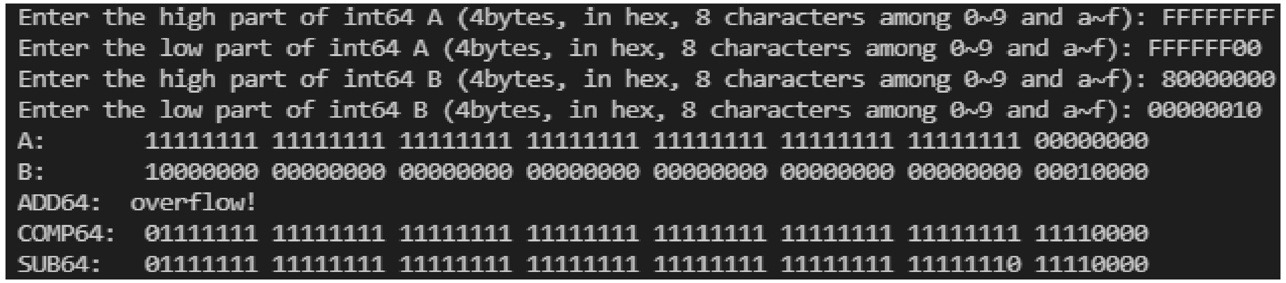
```
Enter the high part of int64 A (4bytes, in hex, 8 characters among 0~9 and a~f): 00000002
Enter the low part of int64 A (4bytes, in hex, 8 characters among 0~9 and a~f): 12341234
Enter the high part of int64 B (4bytes, in hex, 8 characters among 0~9 and a~f): 00000002
Enter the low part of int64 B (4bytes, in hex, 8 characters among 0~9 and a~f): 12341234
A:       00000000 00000000 00000000 00000010 00010010 00110100 00010010 00110100
B:       00000000 00000000 00000000 00000010 00010010 00110100 00010010 00110100
ADD64:   00000000 00000000 00000000 00000100 00100100 01101000 00100100 01101000
COMP64:  11111111 11111111 11111111 11111101 11101101 11001011 11101101 11001100
SUB64:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

1) Input same value



```
Enter the high part of int64 A (4bytes, in hex, 8 characters among 0~9 and a~f): 00000001
Enter the low part of int64 A (4bytes, in hex, 8 characters among 0~9 and a~f): 12341234
Enter the high part of int64 B (4bytes, in hex, 8 characters among 0~9 and a~f): 00000001
Enter the low part of int64 B (4bytes, in hex, 8 characters among 0~9 and a~f): 43214321
A:       00000000 00000000 00000000 00000001 00010010 00110100 00010010 00110100
B:       00000000 00000000 00000000 00000001 01000011 00100001 01000011 00100001
ADD64:   00000000 00000000 00000000 00000010 01010101 01010101 01010101 01010101
COMP64:  11111111 11111111 11111111 11111110 10111100 11011110 10111100 11011111
SUB64:   11111111 11111111 11111111 11111111 11001111 00010010 11001111 00010011
```

2) Input each other value



```
Enter the high part of int64 A (4bytes, in hex, 8 characters among 0~9 and a~f): FFFFFFFF
Enter the low part of int64 A (4bytes, in hex, 8 characters among 0~9 and a~f): FFFFFF00
Enter the high part of int64 B (4bytes, in hex, 8 characters among 0~9 and a~f): 80000000
Enter the low part of int64 B (4bytes, in hex, 8 characters among 0~9 and a~f): 00000010
A:       11111111 11111111 11111111 11111111 11111111 11111111 11111111 00000000
B:       10000000 00000000 00000000 00000000 00000000 00000000 00000000 00010000
ADD64:  overflow!
COMP64:  01111111 11111111 11111111 11111111 11111111 11111111 11111111 11110000
SUB64:   01111111 11111111 11111111 11111111 11111111 11111111 11111110 11110000
```

3) Example of overflow

Making a 64-bit Calculator Using two 32-bit

- 4. Evaluation

Do not submit this assignment.
Do it yourself or collaborate with your team.

We do not grade these assignments and are awarded.

Making a 64-bit Calculator Using two 32-bit

Good Luck

If you have any questions about the assignment,
Send an email to the TA of Class1.

The solution code will be released in about 4-5 weeks.

tyeun7@sogang.ac.kr