COMP 3050-201/202 Computer Architecture

Homework #5 Spring, 2018

• This assignment is due no later than midnight (11:59:59 PM) **March 28/29**.

• All submissions must be made electronically using the submit command as described below.

- **File 1:** A short **write-up** that **first** specifies what you think your **degree of success** with a project is (**from 0% to 100%**), followed by a brief discussion of your approach to the project along with a **detailed description** of any problems that you were **not** able to resolve for this project. **Failure to specifically provide this information will result in a 0 grade** on your assignment. If you do **not disclose** problems in your write-up and problems are detected when your program is tested, you will receive a grade of 0. **Make sure that you include your email address in your write-up so that the corrector can email you your grade.**
- **File(s) 2(a, b, c, …)**: Your **complete source code** …. For this assignment you should have a single masm (.asm) source file for your work.
- **File 3:** We don't need a makefile here, but you should briefly describe the steps needed to build and run this problem in a simple text file.
- **File 4:** A file that includes your **resulting output** run(s) from your project. This is a simple text file that shows your output, but make sure that you **annotate it** so that it is self-descriptive and that all detailed output is well identified.

• The files described above should be the only files placed in one of your subdirectories, and this subdirectory should be the target of your submit command, this time specifying "hw5". See the on-line file **Assignment_Submit_Details.pdf** for specific directions.

- The **AddInput** function will return (in the **AC**) a value of **0 f it succeeds**, **and a value of -1 if it overflows.**

- The function will get its operands by reading **two <u>positive only</u>** base 10 integers from the input buffer (address 4092) and will convert the input from the arriving series of ASCII characters read from the receiver into 16 bit, 2's complement binary integers.

- The function must then add the two binary numbers together, encode the result into a set of numeric ASCII bytes and print the result to the output buffer (address 4094) using polled IO as discussed in class (a NL and CR character must be printed immediately after the numeric result is printed). It must also store the computed sum as a binary value into the location pointed to by the single argument passed.

- You must provide output results for the five sets of numbers shown below

    - `235 + 0`
    - `16341 + 957`
    - `23786 + 12400`
    - `12 + 23`
    - `1 + 1343`

- Each input number is expected to be on a separate line with a new line (enter key) pressed after each complete number is typed.

- The input values must be in base 10 and output values printed must be in base 10. (You will need the microcode that you wrote in assignment 3, since the **MULT** and **DIV** instructions will be needed to do the conversions required in this assignment, and the **RSHIFT** instruction is useful for shifting string bytes into position to send to the transmitter.)

- If the numbers you add overflow, do not store a result in the location pointed to by the function argument, but do print out an ascii message that states an overflow condition.

- Indicate **clearly** in your short write-up, how much of this assignment you feel you were able to do correctly, and what, if anything, you were not able to do. **Failure to specifically provide this information will result in a 0 grade** on your assignment. If you do not disclose problems in your write up and problems are detected when your program is tested, you will receive a grade of 0.

Since your output will be printed on the screen you do not need to show results with the debugger as we have done previously, but you will need to execute a halt command to stop your program which will show one debugger frame. The output for one successful and one unsuccessful (overflow occurred) run should look something like what's shown below:

Addition Succeeds

```
-bash-3.00$./mic1 promfile.dat IO_adder.obj 0 2048
Read in 107 micro instructions
Read in 120 machine instructions
Starting PC is : 0000000000000000 base 10: 0
Starting SP is : 0000100000000000 base 10: 2048

PLEASE ENTER AN INTEGER BETWEEN 1 AND 32767
208
PLEASE ENTER AN INTEGER BETWEEN 1 AND 32767
5075
THE SUM OF THESE INTEGERS IS:
5283

ProgramCounter : 0000000000101101 base 10: 45
Accumulator : 0000000000000000 base 10: 0
InstructionReg : 1111111100000000 base 10: 65280
TempInstr : 1000000000000000 base 10: 32768
StackPointer : 0000100000000000 base 10: 2048
ARegister : 1111111111110101 base 10: 65525
BRegister : 0000000000000000 base 10: 0
CRegister : 0000000000000000 base 10: 0
DRegister : 0000000000000000 base 10: 0
ERegister : 0000000000000000 base 10: 0
FRegister : 0000000000000000 base 10: 0
Total cycles : 184781

Type decimal address to view memory, q to quit or c to continue: q
MIC-1 emulator finishing, goodbye
```

Addition Overflows

```
-bash-3.00$ ./mic1 promfile.dat adder.obj 0 2048
Read in 107 micro instructions
Read in 120 machine instructions
Starting PC is : 0000000000000000 base 10: 0
Starting SP is : 0000100000000000 base 10: 2048

PLEASE ENTER AN INTEGER BETWEEN 1 AND 32767
31000
PLEASE ENTER AN INTEGER BETWEEN 1 AND 32767
10000
THE SUM OF THESE INTEGERS IS:
OVERFLOW, NO SUM POSSIBLE

ProgramCounter : 0000000000101101 base 10: 45
Accumulator : 1111111111111111 base 10: 65536
InstructionReg : 1111111100000000 base 10: 65280
TempInstr : 1000000000000000 base 10: 32768
StackPointer : 0000100000000000 base 10: 2048
ARegister : 1111111111110101 base 10: 65525
BRegister : 0000000000000000 base 10: 0
CRegister : 0000000000000000 base 10: 0
DRegister : 0000000000000000 base 10: 0
ERegister : 0000000000000000 base 10: 0
FRegister : 0000000000000000 base 10: 0
Total cycles : 184781

Type decimal address to view memory, q to quit or c to continue: q

MIC-1 emulator finishing, goodbye
```