Computer Science 304
Computer Organization
Spring 2025
Assignment 9

Due: 11:59 p.m., Friday, 5/2/2025

For this project, you will write programs in C and Y86 assembly that compute various math functions.

For the C program, you are only allowed to use addition, subtraction, and bitwise conjunction (and). Your solution should consist of one source file with the following functions as listed below:

`int neg (int num)`
returns the negation of **`num`**

`int abs (int num)`
checks **`num`** to see if it is non-negative; if not, gets positive value by calling **`neg`**; returns abs value of **`num`**

`int mult (int op1, int op2)`
computes and returns the product of **`op1`** and **`op2`** by repeatedly adding **`op1`** to itself **`op2`** times (this function should be called with **`op1 >= op2`**, if possible)

`int rfact (int n)`
computes and returns the factorial value of **`n`** through recursion (see IA32 **`rfact`** in slides)

```
main
   get first num from array

   while num != 0
      print num in decimal and hex

      call neg to get negated value of num and print in decimal and hex

      call abs to get absolute value of num; store in num2

      call rfact to get factorial of num2 and print in decimal and hex

      print hex row of 1's to separate from next num

      get next num from array
```

The Y86 assembly code should execute similarly to the C program; however, the printed results should be stored in an array declared at the end of the assembly code called **`output`**. As the program runs, results should be stored in successive locations. These values will be printed in lowercase hex at the end of execution and can be compared to those generated by the C program.

The input for both programs should be stored in a variable array called **`input`** (for C) and an array called **`input`** near the end of the Y86 assembly code. The last entry in this array should be set to 0 so that the program knows when to stop. The stack should be the last item listed in the assembly with a suggested starting location of 0x1000.

All the above C functions should be written as formal subroutines in Y86 using the stack for calls and returns, along with parameters and temporary value storage. Keep in mind that in Y86, you only have operations for add, subtract, bitwise and, and bitwise xor; though here, you should only need add and subtract. Be sure to comment both your C and Y86 code for clarity, including a header in each file. Additionally, comment nearly every line of your Y86 code, use mnemonic variable names and labels in C and Y86, respectively, and format all code with blank lines, indentation, and consistent, neat spacing. Finally, you are expected to follow good programming practices.

For your submission, hardcode the values in the **`input`** array to 5, -8, -12. The TA may change these to test other values for grading.

The C program should be compiled and executed with the commands:

```
gcc rfun.c -o rfun
rfun
```

The Y86 program should be assembled and executed with the commands:

```
yas rfun.ys
yis rfun.yo
```

You should submit the following files on Blackboard:

```
rfun.c
rfun.ys
```