CSE 320 **Fall 2018**

Computer Project #9

Assignment Overview

This assignment develops familiarity with subprograms in assembly language. You will develop a set of ARM assembly language functions which implement common mathematical operations.

It is worth 40 points (4% of course grade) and must be completed no later than 11:59 PM on Thursday, 11/8.

Assignment Deliverables

The deliverables for this assignment are the following files:

`proj09.makefile` – the makefile which produces **`proj09`**
`proj09.support.s` – the source code for your support module
`proj09.driver.c` – the source code for your driver module

Be sure to use the specified file names and to submit them for grading via the CSE handin system.

Assignment Specifications

1. You will develop the ARM assembly language functions listed below:

```
int negate( int N );
int absolute( int N );

int add( int A, int B );
int sub( int A, int B );
int mul( int A, int B );
int divide( int A, int B );

int power( int N, int I );
int factorial( int N );
```

Those eight functions (and any "helper" functions which you develop) will constitute a module named "proj09.support.s". The functions in that module will not call any C library functions.

Function "negate" will return the negation of N.

Function "absolute" will return the absolute value of N.

Function "add" will return the sum of A and B.

Function "sub" will return the value of B subtracted from A.

Function "mul" will return the product of A and B.

Function "divide" will return the quotient of A divided by B.

Function "power" will return N raised to the Ith power.

Function "factorial" will return N!.

All functions will return the value 0x80000000 for error conditions.

2. You will develop a driver module to test your implementation of the support module. The driver module will consist of function "main" and any additional helper functions which you choose to implement. All output will be appropriately labeled.

Your driver module may not be written as an interactive program, where the user supplies input in response to prompts. Instead, your test cases will be included in the source code as literal constants.

Assignment Notes

1. The functions in your support module must be hand-written ARM assembly language functions (you may not submit compiler-generated assembly language functions).

2. Your program will be translated and linked using "gcc". For example, the following commands could be used to translate and link your program, then load and execute it:

```
<prompt> gcc -march=native -c proj09.support.s
<prompt> gcc -Wall -c proj09.driver.c
<prompt> gcc proj09.support.o proj09.driver.o -o proj09
<prompt> proj09
```

The option "-march=native" allows the assembler to use "sdiv" instructions.

3. In order to interface ARM assembly language functions with C functions, you must follow certain conventions about register usage.

The calling function will place up to four parameters in registers R0 through R3 (with the first argument in register R0).

The called function must save and restore registers R4 through R11 if it uses any of those registers (the calling function assumes that registers R4 through R11 are not altered by calling another function).

The called function place its return value in register R0 before returning to the calling function.

Registers R12, R13, R14 and R15 are used by the system and their contents must not be modified by your functions.