CMPE-250
Assembly and Embedded Programming
Spring 2021
Laboratory Exercise Two
Basic Arithmetic Operations

This exercise investigates arithmetic in an Arm Cortex-M0+ assembly language program. The objective of this exercise is familiarization with basic arithmetic operations using the Cortex-M0+ (movement of data into registers, use of constant values, and basic Cortex-M0+ arithmetic instructions). An assembly language program is written to compute an arithmetic expression, and it is simulated in the Keil MDK-ARM IDE.

Prelab Work

For the expression below from the program specification, calculate and write the results (in 32-bit hexadecimal representation) of *each* arithmetic operation in evaluating the expression from left to right, observing standard algebraic order of operations. In lab, these intermediate and final results will be used to verify the correct contents of the Cortex-M0+ registers, which the Keil simulator shows only in 32-bit hexadecimal.

$$45 + 6 + ([-13] \div 4) - (7 \times 3) - 65 + 33$$

Program Specification

Write a properly commented and properly formatted Cortex-M0+ assembly language program to compute the result of the expression below.

$$45 + 6 + ([-13] \div 4) - (7 \times 3) - 65 + 33$$

In addition to computing the correct arithmetic result, the program must meet the following requirements.

- The expression, as written above without any simplification, must be evaluated from left to right, observing standard algebraic order of operations.
- The only arithmetic instructions that may be used are addition and subtraction.
- Multiplication and division operations must be implemented with a combination of shift and addition instructions.
- Any constants needed must be moved into registers as values before using them. (Exception: for shifts to implement multiply or divide, the shift amount may be an immediate value in the instruction, as directed below.)
- The values in the expression must be declared or obtained as follows.
  - Constants 45, 6, 13, 7, 65, and 33: immediate values in move instructions
  - Shift amounts for ÷ 4 and × 2: assembler EQUates as DIV4 and MULT2, respectively, for the number of bits to shift

Procedure

1. Create a new directory (folder) and Keil MDK-ARM project for this exercise.
2. Write a properly commented and properly formatted Cortex-M0+ assembly language program according to the preceding specification.
3. Assemble and build the program with the Keil ARM assembler to create a listing file and a map file.
4. Debug the program using the simulator, and verify that the results match your prelab calculations.
5. Capture the final simulation results. (To generate a bitmap copy of the simulator window, press *Alt-Prt Scr*. Then open Microsoft Word and paste.)
6. Demonstrate step 4 for your lab instructor.
7. Submit your source file (.s), listing file (.lst), and report to the respective myCourses assignments for this lab exercise.

Baseline Metrics

The metrics in the table below can be used to compare your solution to a baseline solution. They are provided solely for your information and personal curiosity about the efficiency of your solution. There are no grading criteria associated with how your code compares with them.

Code	Instructions/LOC	Code Size (bytes)
Assembly (program code*)	15	30
Assembly (total read only)	—	280
Assembly (total image)	—	536
C (program code)	17	34
C (total read only)	—	524
C (total image)	—	528

*Not including instructions from provided program template.

Report

Write a report consisting of the following sections. Your writing should follow the rules of professional technical writing and should meet the specifications in "Laboratory and Report Guidelines" on myCourses.

- Abstract
- Procedure
- Results
  - ☞ The screen capture from step 5 should be part of your report's results section. (Remember to introduce and explain it in text rather than merely pasting it into the report.)
- Conclusion

Submission

myCourses Assignments
Separate assignment for each item below • Report (including cover sheet) • Source code (.s) • Listing (.lst)

As specified in "Laboratory and Report Guidelines," late submissions are penalized per day late, where "day late" is calculated by rounding submitted time to the nearest greater or equal integer number of days late, (i.e., ceiling function).

Note: To receive grading credit for this lab exercise you must earn points for *both* the demonstration *and* the report.