CSC 230 - SUMMER 2017
INTRODUCTION TO COMPUTER ARCHITECTURE
ASSIGNMENT 1
UNIVERSITY OF VICTORIA

Due: Sunday, May 28th, 2017 before 11:55pm. **Late assignments will not be accepted.**

1 Overview

For this assignment, you will implement two short programs in AVR assembly. Both programs will require the use of memory access instructions, conditional branches and arithmetic with carry. You should start early to ensure that you can ask for help if you get stuck.

Although the AVR instruction set has a full slate of addition and subtraction instructions, there are very few multiplication instructions and no instructions for integer division. The objective of this assignment is to implement multiplication and division in AVR assembly to allow 16-bit numbers to be multiplied or divided (with both the quotient and remainder retained). Although there are several variants of the MUL instruction available, you will likely find it easier to write your solution using only addition and subtraction (since the MUL instructions have a peculiar format that does not map well onto this task).

Let A and B be 16-bit unsigned integers. The product AB requires at most 32 bits. The quotient A/B and remainder $A\%B$ each require at most 16 bits. For this assignment, your task is to write two AVR assembly programs:

- A program to compute the product AB and store it in memory as a 32-bit little-endian unsigned integer.
- A program to compute the quotient A/B and remainder $A\%B$ and store each value in memory as a 16-bit little-endian unsigned integer.

2 Code Templates

Since the only environment available for running AVR assembly (besides the AVR boards that we haven't used yet) is the simulator, it is not really possible to write a complete program in the traditional sense, because there is no facility for input and output. Instead, two code templates have been posted on conneX: `mul32.asm` and `divmod16.asm`. In each template, the 16-bit input operands A and B are pre-loaded into two pairs of registers (R17:R16 for A and R19:R18 for B). Add your implementation of each algorithm to the indicated place in each template. You may change the loaded operand values for testing purposes, but do not move or change any other aspects of the `ldi` instructions. When your code has computed the result for each task, store the result in data memory (using the variables indicated in each code template). If you do not use the code templates provided to implement your solution, it will not be possible to mark your code and you will receive a mark of zero.

3 Evaluation

Submit your completed `mul32.asm` and `divmod16.asm` files electronically through the Assignments tab on conneX. Do not submit any other files.

The assignment will be marked out of 12 marks and is worth 6% of your final grade.

The marks are distributed among the components of the assignment as follows. If either file does not correctly assemble **as submitted**, you will receive no marks for that part of the assignment (since it will be impossible to test your code). Make sure your code assembles exactly as submitted (even if an error is caused by a minor error like a typo, you will receive a zero if the file does not assemble).

Marks	Component
1	The `mul32.asm` code gives the correct result when both operands are less than 16. Note that to be "correct", the result must be stored in the required memory locations (`OUT3:OUT0`) Solutions which store the result anywhere else (or leave it in registers) will not receive any marks.
2	The `mul32.asm` code gives the correct result when the product AB is at most $2^{16}-1$.
1	The `mul32.asm` code gives the correct result when one operand is less than 256 (and the other operand is unconstrained).
2	The `mul32.asm` code gives the correct result for all valid operand pairs.
2	The `divmod16.asm` code gives the correct quotient value (in memory locations `DIV1:DIV0`) when the denominator (B) is less than 256.
2	The `divmod16.asm` code gives the correct remainder value (in memory locations `MOD1:MOD0`) when the denominator (B) is less than 256.
2	The `divmod16.asm` code gives the correct quotient and remainder values for all valid operand pairs.

You are permitted to delete and resubmit your assignment as many times as you want before the due date, but no submissions or resubmissions will be accepted after the due date has passed. You will receive a mark of zero if you have not officially submitted your assignment (and received a confirmation email) before the due date.

The two files you submit are expected to work correctly in the simulator provided by the AVRStudio4 environment in ECS 249. Only the files that you submit through conneX will be marked. The best way to make sure your submission is correct is to download it from conneX after submitting and test it. You are not permitted to revise your submission after the due date, and late submissions will not be accepted, so you should ensure that you have submitted the correct version of your code before the due date. conneX will allow you to change your submission before the due date if you notice a mistake. After submitting your assignment, conneX will automatically send you a confirmation email. **If you do not receive such an email, you did not submit the assignment.** If you have problems with the submission process, send an email to the instructor **before** the due date.