DEPARTMENT OF COMPUTER SCIENCE
COLORADOSCHOOLOFMINES EARTH • ENERGY • ENVIRONMENT

CSCI 341: Computer Organization
HW 9: Multiplication

When writing code, comments are important! Note the rubric specifies that about half your lines should be commented - this is not atypical for assembly.

We strongly recommend you write the pseudocode first (it is a required element in the submitted README), and if needed, rewrite it using only the constructs if-goto and goto, to make the leap from abstraction to RISC-V smaller. Have your pseudocode handy if you come to office hours for assistance with this.

In this assignment you will be implementing the *optimized* hardware multiplication algorithm in RISC-V. If you need a refresher on how this algorithm works, look at the slides presented in class. The multiplication algorithm must be implemented as a procedure called "mult", with the following signature:

1. Name: mult
2. Arguments:
   a1: First Number
   a2: Second Number
3. Return Value:
   a0: First number * second number

This program will be written in a similar manner to the previous two, with the mult procedure declared globally. This assignments grading script will be used when grading to call your procedure. To test your code, both the file with your procedure and the grading script need to be open in RARS, with no other files open. Under the settings menu in RARS check the "Assemble all files currently open" and "Initialize Program Counter to global main if defined" options.

Requirements/Limits

Implement the multiplication algorithm as described in Figure 3.5 in the textbook (i.e., the optimized version in the lecture slides) using shifts, addition, subtraction, and other basic operations to complete the algorithms. You are NOT allowed to use any form of RISC-V multiplication or division instructions (such as mul or div) in your solution.

Your function takes parameters as specified above and returns a value; it does not read anything from the user or print anything to the console.

The parameter values are taken as 32 bit unsigned values and our perfect user will only enter values that result in a 32-bit result (all upper 32 bits are 0). That is, you can assume all numeric inputs and results will be valid 32-bit unsigned integers.

Notes/Hints

We are asking you to assume the values are unsigned and making your calculations; however values are read in and printed out as signed so you may see/use negative values when inputs or results are over 2^32-1.

Multiplication does calculate a 64-bit value spread across two registers; you only need to return the low word in a0, since we guarantee we will only input values with results that fit in 32 bits.

Functions must be used properly (as demonstrated in class and the textbook) and use the algorithm requested in this prompt to receive full points. Remember, a function cannot assume values (0 or otherwise) in any registers other than those passed in as arguments (a2 and a3 in this case).

We expect your algorithm to function correctly for all unsigned 32 bit values, with multiplication results fitting in 32 bits.

For multiplication, the final product will be in the lower half of the 64-bit Product register, since we are guaranteeing it will fit in 32 bits.

The Product register is a pair of 32-bit registers. If you choose (s3 s4) for your product register, when shifting it right one bit, s3's LSB should become MSB of s4. Note, s3 is the upper half and s4 the lower half of the 64-bit product register.

Example

An example run through of the program is shown below along with several test cases.

```
Enter a number: 9
Enter a number: 4
Mult Result: 36

-- program is finished running (0) --
```

First Number	Second Number	Result
536870911	4	2147483644

77	192	14784

As always, assume a perfect user (see notes above).

What to submit

What to turn in:

1. A zipped file titled lastnameFirstname containing:
    a. Your README, with the file name lastnameFirstname_README.txt
    b. Your code, with the file name lastnameFirstname_ASSEMBLY.s **Do not submit the file you downloaded to test your code.**

In your README, include the following elements:

1. Include YOUR NAME and the names of all people who helped/collaborated as per the syllabus and CS@Mines collaboration policy. [This is an individual assignment]
2. Pseudocode (in C) for your implementation.
3. Describe the challenges you encountered and how you surmounted them.
4. What did you like about the assignment?
5. How long did you spend working on this assignment?
6. A description of any features you added for extra credit (if any)

Grading

Section	Points
Procedures Implemented Properly: Caller: *jal ra, LABEL* Callee: *jalr zero, ra,* 0 No use of mul or div (note any solution using only mul will result in a 0) This is a big part of this grade: DOUBLE CHECK IT IS DONE CORRECTLY	10 points
Public Test Cases Pass	2 points each, 4 points total
Private Test Cases Pass	2 points each, 20 points total
Adequate comment coverage (50%)	5 pts
No pseudo instructions	6 pts, -1 for each up to -6 pts

README has required elements and effort is demonstrated	2 pts
Code submission has ".s" file extension Grader script is NOT submitted	3 pts
Total	50 pts