Introduction

MIPS has a **`mul`** instruction. What if you are not permitted to use it? Writing some multiplication using addition and bit-twiddling MIPS code.

Assignment Goals

Learning Outcomes After completing this group assignment, each student is expected to be able to

- Multiply 2's-compliment integers with a naive count-controlled loop and three (3) registers.
- Count the number of bits in a word that are set.
- Understand doubling and adding can be used to multiply *faster*, using the same three (3) registers.

Procedure

Get out paper. The group will turn in *one* document. Make sure all participating members' names are on the page.
Copy each question before the answer. Since the point is to make your answer sheet a stand-alone study guide, the copy need not be verbatim but must give your answer enough context to evaluate.
Assign Roles. Students should take roles they have not held recently (or, perhaps, ever):

Manager Move discussion forward.
Recorder Writes the report that will be turned in.
Reflector Monitor that everyone gets heard and is caught up. (This is a **group** obligation, really.)
Speaker (Combine w/ **Reflector** if necessary.) Asks facilitator questions; communicates what the team has done.

Answer these questions.

1. Write (in MIPS assembly) the `multiply` function. The function takes two parameters, `x` and `y`, and returns their product in **`$v0`**. You may safely assume the product fits in one word.

2. Write a snippet of MIPS code printing "even" if the value in **`$t7`** is even and "odd" if the value in the register is odd. You should use **`bnez`** (or **`beqz`**) for the if statement.

3. Write a stippet of MIPS code that does not use **`div`** or `mod` that puts half the value in **`$t0`** in register **`$t1`**. Assume the value is non-negative and round **down**.

4. When the value in **`$t1`** in the previous question is *odd*, what do you know about the bit pattern in **`$t0`**?

5. Can you **double** the value in register **`$t5`** *without* adding or subtracting? (Or multiplying.)

6. Write a loop in MIPS code that counts the number of 1 bits in the value in register **`$t0`**.

7. Can you use your loop from above to rewrite the body of `multiply`, using doubling, halving, and adding if necessary to get the answer in no more than 32 times through the loop?