ECE 366: Project 2 | Progress Report 2 Due: 04/05/2025

Team Members:
Shahed Alhanbali (670025395)
Cindy Jurado (653416500)
Qudsia Sultana (657855210)

Link to GitHub Repository: https://github.com/shahedalhanbali/Project-2-366

Contribution Breakdown:

- Qudsia Sultana: Built the control flow for even/odd detection and added register tracking for testing purposes.
- Shahed Alhanbali: Helped write and test the repeated subtraction logic for computing m % 2 and confirmed correct register values.
- Cindy Jurado: Debugged loop logic, verified memory outputs, and contributed to writing and formatting this report.

Summary: In this progress report, our team implemented the function `Odd(m)` in MIPS Assembly using division by repeated subtraction, as shown in Figure 3 of the assignment. This method avoids using the MIPS `DIV` instruction, which was prohibited. Instead, the program subtracts 2 repeatedly from `m` to compute `m % 2`.

(b) Design a MIPS program that will implement Odd(m) of Figure 2. Please use division by repeated subtraction to implement m%2 (reads m modulo 2 and computes the remainder of the division m/2). Usage of MIPS DIV instruction will yield a zero (0) point. Use the function of Figure 3 to implement the division by subtraction. [Points: 20]

```
# odd.asm
# MIPS program to check if a number m is odd using repeated subtraction
# Assumes m is stored in memory. Result stored at $t5

.data

m:      .word 5   # change the number based on what we want to test

result: .word 0



.text

.globl main
```

```
main:
   # Load m into $t0
   lw $t0, m           # $t0 = m
   li $t1, 2           # $t1 = divisor (2)

   # Set up loop variables
   move $t2, $t0       # $t2 = copy of m
   li $t3, 0           # $t3 = quotient

division_loop:
   blt $t2, $t1, done_division  # if x < y, break
   sub $t2, $t2, $t1           # x = x - y
   addi $t3, $t3, 1            # increment quotient
   j division_loop

done_division:
   # $t2 now holds remainder (m % 2)
   # If remainder == 0, m is even → store 0
   # If remainder == 1, m is odd → store 1

   li $t4, 0
   beq $t2, $zero, store_even   # If remainder == 0 → even
   li $t4, 1                    # remainder != 0 → odd

store_even:
   sw $t4, result              # store result (1 if odd, 0 if even)
   move $t5, $t4
```

How to Run the Program:

1. Open the file `odd.asm` in MARS MIPS Simulator.

2. Assemble and run the program.

3. Modify the value of `m` in `.data` to test different cases.

4. After execution, check:

   - Or the value in register `$t5`

   - `1` means `m` is odd, `0` means `m` is even

Sample Inputs/Outputs:

m = 5 → result = 1

m = 10 → result = 0

m = 13 → result = 1

m = 4 → result = 0