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

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

Contribution Breakdown:

- Shahed Alhanbali:

  - Helped write and test the Fibonacci function (Part a)

  - Implemented the repeated subtraction logic for `Odd(m)` (Part b)

  - Integrated Fibonacci and Odd-check logic in Part (c)

  - Tested large input values and debugged overflow issues

  - Verified memory and register results across all parts

- Cindy Jurado:

  - Contributed to debugging and verifying the Fibonacci loop logic

  - Documented run instructions and formatted test outputs for all parts

  - Confirmed accuracy of results stored in memory and registers

  - Helped validate edge cases and overflow handling

  - Assisted with formatting and final README/report editing

- Qudsia Sultana:

  - Built control flow logic for all parts and tested corner cases

  - Implemented the Fibonacci loop structure in Part (a) and reused it in Part (c)

  - Connected Fibonacci output to Odd-check in Part (c)

  - Ensured `$t5` and memory values were consistent for output verification

  - Wrote the README and How-To sections

(a) Design a MIPS program that will implement Fibonacci(n) of Figure 1. [Points : 50]

```
# fibonacci.asm
# Computes the n-th Fibonacci number (Fibonacci(n))
# Input: $a0 = n (set manually below)
# Output: $v0 = Fibonacci(n)

.text

.globl main

main:

   ################################################

   # SET THE VALUE OF n TO TEST HERE:

   addi $a0, $zero, 7      # <--- Change 7 to any value you want to test

   ################################################



   # If n <= 1, return n

   addi $t6, $zero, 1

   ble $a0, $t6, base_case



   # Initialize a = 0, b = 1, counter = n

   addi $t0, $zero, 0      # a = 0

   addi $t1, $zero, 1      # b = 1

   add $t2, $a0, $zero     # counter = n



fib_loop:

   subi $t2, $t2, 1        # counter--

   add $t3, $t1, $zero     # temp = b

   add $t1, $t0, $t1       # b = a + b

   add $t0, $t3, $zero     # a = temp

   addi $t6, $zero, 1
```

```
    bgt $t2, $t6, fib_loop  # loop while counter > 1

    add $v0, $t1, $zero     # result = b
    j done

base_case:
    add $v0, $a0, $zero     # result = n (when n <= 1)

done:
    nop                     # End of program
```

How to Run the Program:

- Open the file fibonacci.asm in MARS MIPS Simulator.
- Load the desired value of n into register $a0.
- Assemble (first build by hitting the gear icon) and run the program (green play button).
- The computed Fibonacci number will be stored in register $v0.

Sample Inputs/Outputs:

Input (n)	Fibonacci(n) (decimal)
0	0
1	1
5	5
7	13
10	55

(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

# Checks if a number m is odd using repeated subtraction

# Input: Set m in the line below

# Output: $t5 = 1 if odd, 0 if even


.data

   result: .word 0


.text

.globl main

main:

   ################################################

   # SET VALUE OF m TO TEST HERE:

   addi $t0, $zero, 6      # <--- Change # to test another number

   ################################################


   addi $t1, $zero, 2      # divisor = 2

   add $t2, $t0, $zero     # copy m into $t2

   addi $t3, $zero, 0      # quotient = 0


div_loop:

   blt $t2, $t1, div_done

   sub $t2, $t2, $t1       # m = m - 2

   addi $t3, $t3, 1
```

```
    j div_loop
div_done:
    addi $t4, $zero, 0      # assume even by default
    bne $t2, $zero, is_odd  # if remainder != 0, it's odd
    j store

is_odd:
    addi $t4, $zero, 1

store:
    sw $t4, result
    add $t5, $t4, $zero     # result also in $t5
```

How to Run the Program:

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

2. Assemble and run the program (click the gear icon and click the green play button)

3. After execution, check:

   - 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

(c) Design a MIPS Program IsFibonacciOdd(n) using the programs from Part (a) and Part (b). [Points : 30]

```
# is_fibonacci_odd.asm

# Computes Fibonacci(n), checks if it's odd, and stores result

# Input: Set `n` in the line marked below

# Output: 1 (if Fibonacci(n) is odd), 0 (if even), stored in $t5 and memory


.data

   result: .word 0      # 1 if odd, 0 if even


.text

.globl main

main:

   ################################################

   # SET THE VALUE OF n TO TEST HERE:

   addi $a0, $zero, 12       # <--- Change this value to test another n

   ################################################


   # --- Fibonacci(n) ---

   addi $t6, $zero, 1

   ble $a0, $t6, fib_base


   addi $t0, $zero, 0       # a = 0

   addi $t1, $zero, 1       # b = 1

   add $t2, $a0, $zero      # counter = n
```

```
fib_loop:
    subi $t2, $t2, 1         # counter--
    add $t3, $t1, $zero      # temp = b
    add $t1, $t0, $t1        # b = a + b
    add $t0, $t3, $zero      # a = temp
    addi $t6, $zero, 1
    bgt $t2, $t6, fib_loop

    add $t6, $t1, $zero      # t6 = Fibonacci(n)
    j check_odd

fib_base:
    add $t6, $a0, $zero      # t6 = n (when n <= 1)

# --- Check if Fibonacci(n) is odd ---
check_odd:
    addi $t1, $zero, 2       # divisor = 2
    add $t2, $t6, $zero      # copy of Fibonacci(n)
    addi $t3, $zero, 0       # quotient = 0

mod_loop:
    blt $t2, $t1, mod_done
    sub $t2, $t2, $t1
    addi $t3, $t3, 1
    j mod_loop
```

```
mod_done:
   addi $t4, $zero, 0       # assume even
   bne $t2, $zero, is_odd
   j store

is_odd:
   addi $t4, $zero, 1       # remainder != 0 → odd

store:
   sw $t4, result
   add $t5, $t4, $zero      # also in register $t5 for easy check
```

How to Run the Program:

1. Open the file is_fibonacci_odd.asm in the MARS MIPS Simulator.
2. Assemble the code.
3. Run the code (click the gear icon and click the green play button)
4. After running:
    - Check register $t5 for the result.
        - 1 → Fibonacci(n) is odd
        - 0 → Fibonacci(n) is even

Sample Inputs/Outputs:

Input (n)	Fibonacci(n)	Output (result)

3	2	0 (even)
5	5	1 (odd)
6	8	0 (even)
7	13	1 (odd)