Due Dec 17 by 11:59pm **Points** 100 **Submitting** a file upload
Available Nov 20 at 12am - Dec 17 at 11:59pm

This assignment was locked Dec 17 at 11:59pm.

. .oject 5 - Prime Tester

General Information

Purpose Project Objectives Possible Points Feedback Disclaimers

This project provides students an understanding of the SIC assembly language. In this project, students will create and implement a prime tester program using the SIC assembly language.

Students are not allowed to use the SIC/XE assembly language for this project.

This project partially satisfies **Course Objective 1**. After completing this project, students will be able to:

1. Create and implement an efficient solution using the SIC assembly language
2. Streamline their assembly program to use the fewest bytes of memory possible
3. Efficiently use memory to store and manipulate values
4. Utilize system devices to transfer their program's output to other applications running in the computer
5. Produce the correct output based on flags provided in the input

This assignment is worth a maximum of 100 points.

Students can expect comments on their Project 5 submission within 5 days after the project deadline.

The hyperlinks provided throughout this project provide access to third-party resources that students will use to learn and support the material discussed within this project. The professor provides these links "**as is**"; therefore, the professor does not nor cannot guarantee or endorse the information found at the linked sites beyond the scope of this course.

Furthermore, the professor accepts no responsibility or liability for the content maintained at the linked sites. Students should report non-working links, as well as other issues with linked material, to the professor.

A thorough discussion on advanced topics related to this project are beyond the scope of this course. The professor has taken great care to ensure students have access to all the material needed to complete this project successfully. Although advanced topics related to this project are not covered, students are encouraged to explore the advanced topics related to this project, if time permits; however, failing to do so will not prevent students from completing the project successfully.

Directions & Important Notes

Directions | **Important Notes** | Required Tools | Academic Integrity

- Download the **Project 5 files (https://canvas.unf.edu/courses/93573/files/16170032?wrap=1)** **(https://canvas.unf.edu/courses/93573/files/16170032/download?download_frd=1)** , which you will need to complete the project
  - Watch the **Project 5 Introduction** **(https://youtu.be/pLYKhXCHA80)** video
  - Watch the **SIC Simulator Introduction** **(https://youtu.be/fZ9bE-HB150)** video
- Review the requirements listed in the **Project Requirements** section
- Create and implement a solution that meets all of the stated requirements
  - Students must create their solution using SIC assembly language
  - Students are not allowed to use the SIC/XE assembly language
- Test your project thoroughly using the provided **SIC Simulator**
  - The professor will use the **SIC Simulator** while grading your project
- Upload the **prime.sic** file to Canvas before the posted due date (refer to the Course Schedule)

Students should create a solution using the C language prior to attempting to implementing the solution in the SIC assembly language. Doing so will help students understand the concepts that are important for completing Project 5 successfully.

Students will be required to use one or more of the following tools to earn a passing grade on the project. Each of the tools listed below can be downloaded for free or already exists in the indicated operating system.

- C Development Environment
  - You are free to use your preferred C IDE and other tools
- Plain text editor
  Note: For creating your **prime.sic** file.

- Windows
  - Notepad++
- Mac OS
  - TextWrangler
  - TextEdit (plain-text mode)
- Linux (terminal)
  - pico
  - vi

- SIC Simulator
  Note: Required to test your **prime.sic** file.
  - The SIC Simulator is provided in the **Project 5 files (https://canvas.unf.edu/courses/93573/files/16170032?wrap=1)** **(https://canvas.unf.edu/courses/93573/files/16170032/download?download_frd=1)**

Students are strongly encouraged to adhere to **UNF's Academic Integrity (https://www.unf.edu/catalog/policies/academic_integrity/)** policy while working on the projects for this course. Work that is too similar to another student's work (current or former) may receive 0 points.

Unless authorized to do so by the professor, students should avoid directly copying code from

- Websites like **Stack Overflow (https://stackoverflow.com/)**
- Repositories like **GitHub (https://github.com/)**
- Fellow students (including current or former COP3404 students)

While students are encouraged to work together to complete their projects,

- Share ideas - not code
- Do not provide direct access to files
- Do not share computing devices

Project Requirements

Preliminary Tasks

- Create a solution using your preferred C programming tools prior to attempting the project
  - This will help you understand the concepts that are important for completing Project 5 successfully
- An automatic 10-point (10%) **penalty** will be assessed for very **disorganized** code.
  - Remove the following items from your **prime.sic** file before submitting to Canvas
    - Blank lines
    - Unnecessary comments

Project Information

Introduction

You have been tasked with creating a SIC program to determine whether a given number is prime or not.

Prime Tester Program Requirements and Limitations

- Prime Tester program input (number to test)
  - Written to the **first word** of SIC memory
  - Range of 0 to 999
- Prime Tester program process
  - Define the constants and variables needed
    Note: Constants and variables can exist anywhere in the program, as needed.
    - Constant: **BYTE** or **WORD**
    - Variable: **RESB** or **RESW**
  - Define a **custom subroutine** to exit the program
    - Place the value 0 in the **Linkage** register (i.e., LDL   ZERO)
    - Perform the **RSUB** command
  - Define a **custom subroutine** to calculate the **quotient** and **remainder**
    - Use a variable to store the **quotient**
    - Use a variable to store the **remainder**
  - Define a **custom subroutine** to display the **digits** of a number
    - Handle up to 3 digits
    - Do not display **preceding** 0s
      - The number **2** should display as 2 rather than 002
      - The number **42** should display as 42 rather than 042
      - The number **402** should display as 402
  - Read input value from the **first word** of SIC memory
    - **Validate** the input value is **between** 0 and 999
      - Only test if the input value is **greater** than **999**
    - If input value is **out of range**, (i.e., **greater** than 999)
      - Write ERROR: INPUT OUT OF RANGE! to the **output device**
      - Exit the program
  - Write the **input value** digits followed by a space character to the **output device**
    - Example: 73
    - You are not required to **test** if the device is ready
    - Use SIC Simulator device 42
  - Determine if the input value is **less than** 2
    - If the input value is less than 2,
      - Write IS NOT PRIME: LESS THAN 2 to the **output device**
        - You are not required to **test** if the device is ready
        - Use SIC Simulator device 42

            - Exit the program
    - Determine if the input value is **even** and **not equal** to 2
        - If the input value is even and not equal to 2,
            - Write IS NOT PRIME: EVEN NUMBER, BUT NOT 2 to the **output device**
                - You are not required to **test** if the device is ready
                - Use SIC Simulator device 42
            - Exit the program
    - **Loop** through all the odd numbers **between** 3 and **one-half** of the input value
        - During each **iteration** of the loop,
            - Determine if the **current** odd number **divides evenly** into the input value
                - Call the **custom subroutine** to calculate the **quotient** and **remainder**
                    - The **quotient** is stored the HI variable
                    - The **remainder** is stored the LO variable
            - If the current odd number divides evenly into the input value
                - Write IS NOT PRIME: EVENLY DIVISIBLE BY X to the **output device**
                    - X is the **current** odd number
                    - You are not required to **test** if the device is ready
                    - Use SIC Simulator device 42
                - Exit the program
            - Otherwise, prepare for the **next** odd number
    - If the program completes the loop,
        - Write IS PRIME to the **output device**
            - You are not required to **test** if the device is ready
            - Use SIC Simulator device 42
        - Exit the program
- Prime Tester program limitations
    - The submitted program cannot be **larger** than 600 bytes
        - A larger program will be accepted with a penalty
    - The program must **load** itself in the lowest memory address possible
        - Do not store **instructions** or **data** in the first word of SIC memory

Extra Credit

You can earn 25 points of extra credit by submitting your **prime.sic** SIC assembly language program to Canvas by 11:59pm on **Sunday**, December 10th.

Important Notes

- Your **prime.sic** SIC assembly language program must earn at least **75 points** to be eligible for extra credit
- You **cannot** resubmit your **prime.sic** file for any reason after the **December 10th** to remain eligible for extra credit

Important Material

The following material will be helpful in completing and testing your **prime.sic** program. Most of these documents are provided in the **Project 5 files (https://canvas.unf.edu/courses/93573/files/16170032?wrap=1) (https://canvas.unf.edu/courses/93573/files/16170032/download?download_frd=1)** download.

- Example Program.sic
  - A documented example SIC assembly program to illustrate basic functionality of the SIC assembly language
- SIC Instruction Set
  - Provides a thorough discussion of the SIC instruction set and registers
- Prime Datasheet
  - Provides a list of all the prime numbers between 0 and 999
- **Prime Number Calculator (http://www.math.com/students/calculators/source/prime-number.htm)**
  - This website may be helpful in thoroughly testing your **prime.sic** program

Examples

The following examples demonstrate the output your **prime.sic** program should produce given the input value.

Note: Your prime.sic program may be tested with values not listed in these examples.

```
Input (HEX) Output
-----       ------
000000      0 IS NOT PRIME: LESS THAN 2
000001      1 IS NOT PRIME: LESS THAN 2
000002      2 IS PRIME
000003      3 IS PRIME
000004      4 IS NOT PRIME: EVEN NUMBER, BUT NOT 2
000005      5 IS PRIME
000006      6 IS NOT PRIME: EVEN NUMBER, BUT NOT 2
000007      7 IS PRIME
000008      8 IS NOT PRIME: EVEN NUMBER, BUT NOT 2
000009      9 IS NOT PRIME: EVENLY DIVISIBLE BY 3
00000A      10 IS NOT PRIME: EVEN NUMBER, BUT NOT 2
0003E8      ERROR: INPUT OUT OF RANGE!                 // Input value is 1,000
```