Mohammad Iqbal  
CSC 210  
10/15/2024

**Technical Report: Rational Number Arithmetic Using Macros in MIPS Assembly**

**Overview**

This MIPS assembly program implements basic rational number arithmetic (addition, subtraction, multiplication, and division) using macros. The program prompts the user to input two rational numbers, selects the desired operation, and prints the result in the form of a rational number (numerator and denominator).

The code leverages the power of **macros** in assembly to enhance modularity and readability. Macros encapsulate common functionality like reading integers, printing strings and integers, and handling rational number arithmetic operations, allowing reuse throughout the program.

**Key Concepts and Components**

**1. Data Segment**

The data segment stores all the static strings and messages used for user interaction:

A close-up of a computer screen

Description automatically generated

These strings are printed during runtime using the macros defined in the code. For instance, prompt\_rational1 prompts the user to input the first rational number, and error\_zero\_den is displayed if a zero denominator is detected.

**2. Macro Definitions**

Macros are used extensively to improve code reuse. Some key macros defined in the code are:

* **print\_str (%str)**: This macro prints the string passed as a parameter to the console using a system call. The string is loaded into $a0, and the system call for printing a string (li $v0, 4) is invoked.

A close-up of a computer code

Description automatically generated

* **print\_newline**: Uses print\_str to print a newline character.

A black text on a white background

Description automatically generated

* **print\_int (%int)**: This macro prints the integer value from a specified register. It moves the integer value into $a0, then invokes a syscall to print an integer (li $v0, 1).

A close-up of a computer code

Description automatically generated

* **print\_rational\_num (%numerator, %denominator)**: This macro prints a rational number in the format of a numerator followed by a denominator, using the previously defined macros for printing integers and newlines.

A close-up of a computer code

Description automatically generated

* **read\_int**: This macro reads an integer from the user by invoking the appropriate system call (li $v0, 5).

A close-up of a white background

Description automatically generated

* **store\_int (%reg)**: Stores the value in $v0 (after read\_int) into the specified register.

A close up of text

Description automatically generated

* **done**: This macro is used to terminate the program gracefully by invoking the syscall to exit (li $v0, 10).

A close-up of a white background

Description automatically generated

These macros ensure that common operations such as input/output and arithmetic are handled concisely and reused throughout the code.

**3. Main Program Flow**

The main section of the program is responsible for orchestrating the input from the user and executing the desired rational number operation based on user input.

* 1. **Input Handling**:
     + The program starts by prompting the user for two rational numbers, each consisting of a numerator and a denominator (e.g., p1/q1 and p2/q2).
     + The user is then asked to select one of four operations: addition, subtraction, multiplication, or division.

Example prompt for operation selection:

A close-up of a white background

Description automatically generated

* 1. **Operation Selection**: Depending on the user's selection, the program jumps to the respective operation handler:
     + - **Addition** ((p1/q1) + (p2/q2)):

A black text on a white background

Description automatically generated

* + - * **Subtraction** ((p1/q1) - (p2/q2)):

A black text on a white background

Description automatically generated

* + - * **Multiplication** ((p1/q1) \times (p2/q2)):

A mathematical equation with black text

Description automatically generated

* + - * **Division** ((p1/q1) / (p2/q2)):

A mathematical equation with black text

Description automatically generated

Example branch for the addition operation:

A close-up of a number

Description automatically generated

* 1. **Arithmetic Operations**: Each arithmetic operation involves simple multiplication and addition/subtraction. For instance, in addition:

A group of black text

Description automatically generated

* 1. **Result Display**: After the operation, the result is printed in the form of a rational number using the print\_rational\_num macro:



* 1. **Denominator Check**: If a zero denominator is detected at any point (e.g., during division), the program jumps to an error handler that prints an error message:

A group of black text

Description automatically generated

4. **Arithmetic Logic**

Each operation is implemented through basic integer multiplication and addition/subtraction. The use of registers is as follows:

* $t0: p1 (numerator of the first rational number)
* $t1: q1 (denominator of the first rational number)
* $t2: p2 (numerator of the second rational number)
* $t3: q2 (denominator of the second rational number)
* $t4: Holds the user’s operation choice
* $t8: Stores the computed numerator
* $t9: Stores the computed denominator

**Conclusion**

This MIPS assembly program efficiently handles rational number arithmetic using macros, which improves modularity and code reuse. The program’s flow revolves around taking user input, performing the selected operation, and printing the results, all while managing errors like zero denominators. The use of macros for repetitive tasks simplifies the program, making it more readable and maintainable.