Project 1: **Calculator**

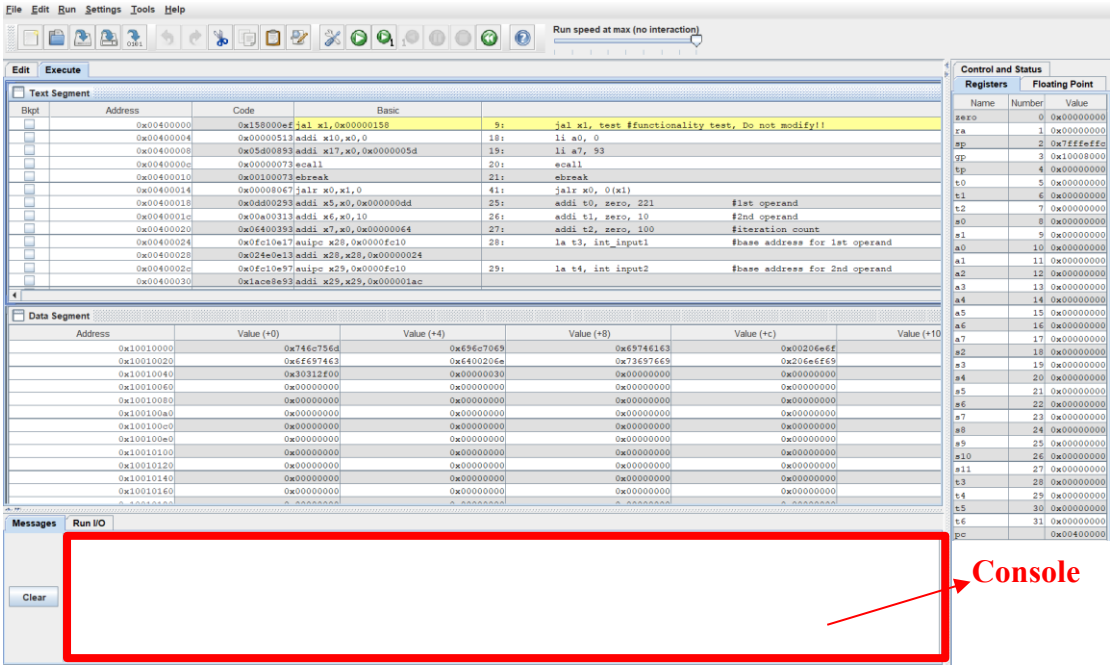
1. Description

The goal of this project is to write a calculator program with RISC-V assembly language. This program will perform the **32-bit** addition, subtraction, multiplication, and division with unsigned numbers. You will use a RISC-V emulator called **RARS** for this project.

- **RARS** (https://github.com/TheThirdOne/rars)
- The **RARS** tutorial is available in the LMS

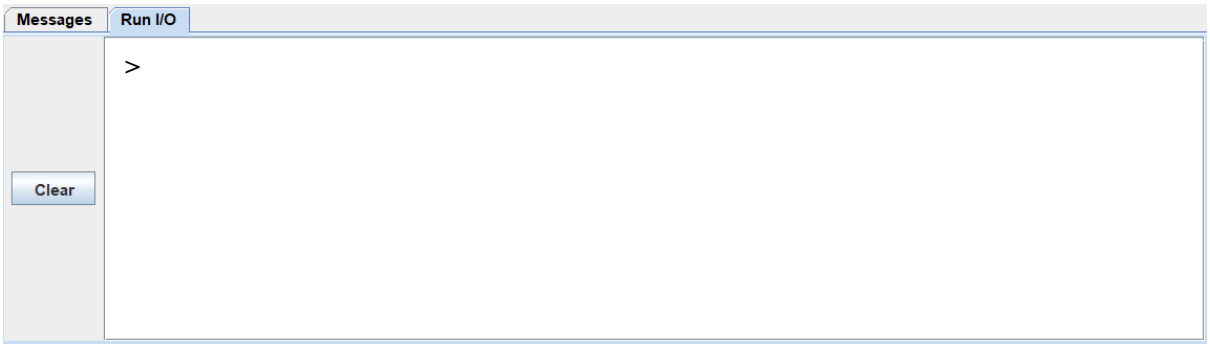
2. Requirements

1) Write a RISC-V assembly language program that performs the **32-bit** addition, subtraction, multiplication, and division.
2) Input and result of the computation are 32-bit values.
3) **You cannot use "mul", "mulh", "mulhu", "mulhsu", "div", "divu", "rem", "remu", and "sub" instructions.**
4) **You have to implement the algorithms that we have studied in Chapter 3 for the multiplication and division (see figures at the slide 3-8 and 3-15).**
5) Your program should be able to read a string from the console of the RARS emulator and print the computation result to the console.

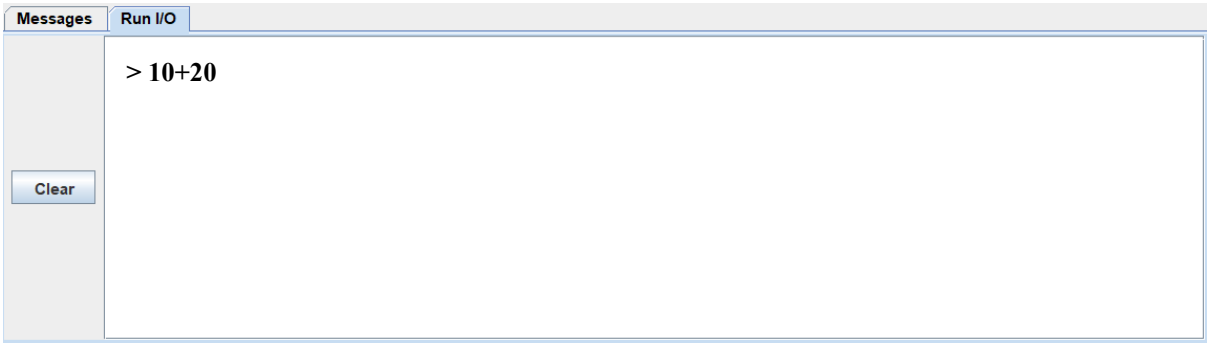


- **Execution example: Test with Console**

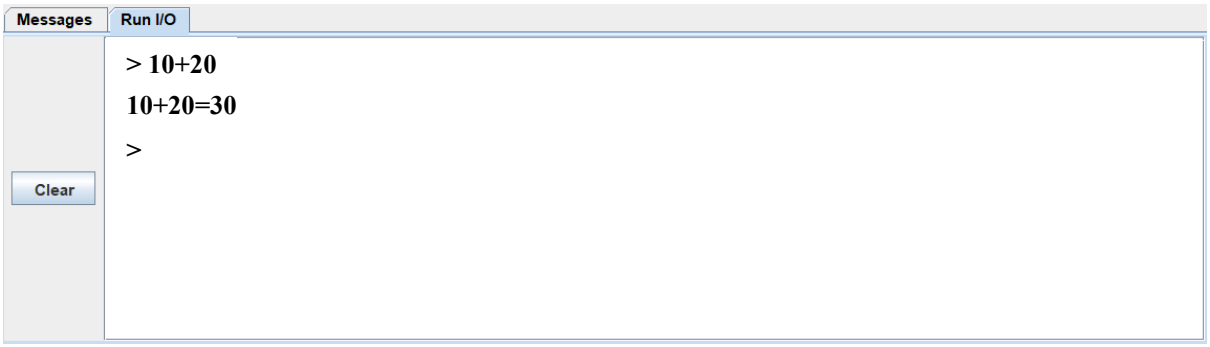
  a. Show prompt in the console.



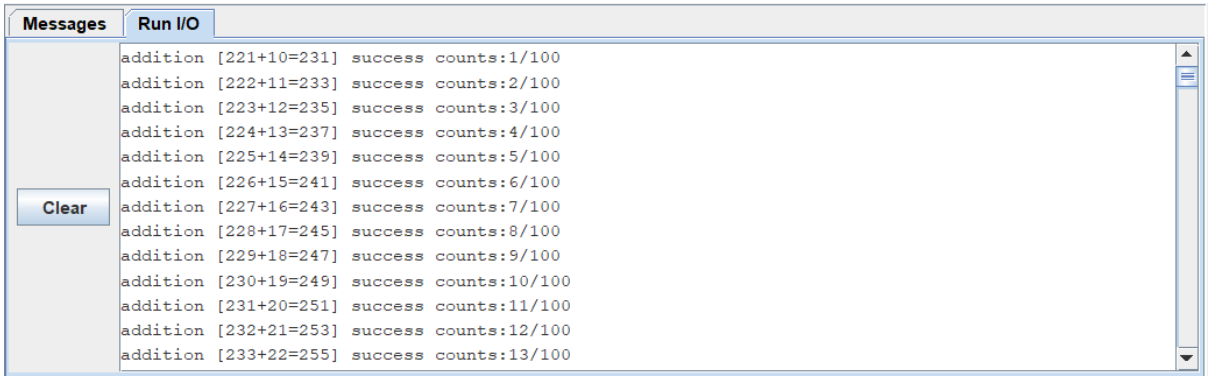
  b. **Type** an equation into the console.



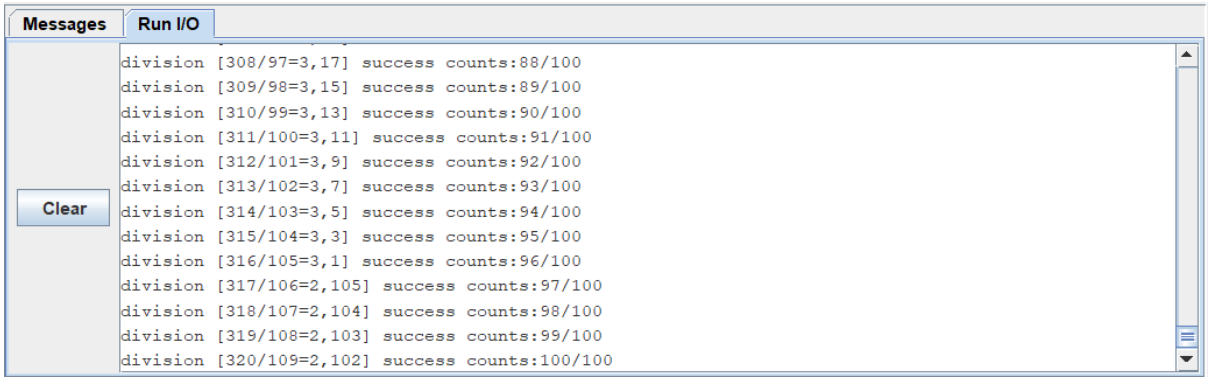
  c. Display a result.



- **Execution example: Automated Test using Loop**



…



3. Procedure

1) **Step1:** Download RARS from the github

2) **Step2:** Download "comp0411proj1.asm" and "common.asm" from the LMS.

3) **Step3:** Edit the comp0411proj1.asm to implement the calculator program.
   - Complete main() procedure
   - Complete calc() procedure
   - You can add your own procedures in the comp0411proj1.asm file.

4) **Step4:** Assemble and test your code
   - main() procedure already includes a basic testing facility. So, you can test the functionality of your code as soon as you complete the calc() procedure.
   - **Your code should meet the following requirements on the argument passing for the calc() procedure.**
     - **x11** should contain the type of the arithmetic operations (0: addition, 1: subtraction, 2: multiplication, 3: division)
     - **x12** should contain the first operand (the dividend for division)
     - **x13** should contain the second operand (the divisor for division)
     - **x10** should be used to return a computation result to caller
     - **x14** should be used to return the remainder of a division operation to caller

5) **Step5:** Submit your assignment to the LMS. You should only submit the following files:
   - comp0411proj1_StudentID.asm (e.g., comp0411p1_201800001.asm)

4. Grading

Your submission will be graded based on the following criteria

1) **Basic functionality test: 70%**
   - **Addition:** 10%
   - **Subtraction:** 10%
   - **Multiplication:** 25%
   - **Division:** 25%

2) **Test with the console: 30%**
   - **Addition:** 5%
   - **Subtraction:** 5%
   - **Multiplication:** 10%
   - **Division:** 10%

Late Day Policy

No late submission is allowed.

Plagiarism

No plagiarism will be tolerated. If the assignment is to be worked on your own, please respect it. If the instructor determines that there are substantial similarities exceeding the likelihood of such an event, he will call the two (or more) students to explain them and possibly to take an immediate test (or assignment, at the discretion of the instructor) to determine the student's abilities related to the offending work.

Appendix

1. How to use system call to read a string from the console and to write a string to the console?

   https://github.com/TheThirdOne/rars/wiki/Environment-Calls

2. Assembler directives

   https://github.com/TheThirdOne/rars/wiki/Assembler-Directives

3. RISC-V calling convention register usage

Register	ABI Name	Description	Saver
x0	zero	Hard-wired zero	—
x1	ra	Return address	Caller
x2	sp	Stack pointer	Callee
x3	gp	Global pointer	—
x4	tp	Thread pointer	—
x5–7	t0–2	Temporaries	Caller
x8	s0/fp	Saved register/frame pointer	Callee
x9	s1	Saved register	Callee
x10–11	a0–1	Function arguments/return values	Caller
x12–17	a2–7	Function arguments	Caller
x18–27	s2–11	Saved registers	Callee
x28–31	t3–6	Temporaries	Caller
f0–7	ft0–7	FP temporaries	Caller
f8–9	fs0–1	FP saved registers	Callee
f10–11	fa0–1	FP arguments/return values	Caller
f12–17	fa2–7	FP arguments	Caller
f18–27	fs2–11	FP saved registers	Callee
f28–31	ft8–11	FP temporaries	Caller