# Week 8 Lab A RISC-V Assembly language programming

## Objectives

* By the end of the session, you will be able to construct their own basic program using RISC-V Assembly Language.
* You will be able to write a program which will add, subtract and store numbers in different registers.
* You will be able to write a program which will output to the console in the software RARS.

## Part A Getting started with RARS/RISC-V assembly language

The tutor will demonstrate how to get started with the RARS simulation environment for RISC-V Assembly language programming, with an example that is different to the one for the exercise.

On Moodle, along with the concepts material, there is a folder named Week 8 concept files that contains the code used in the concepts slides. You can download this code to get started.

In these exercises, you are expected to test your program at each stage and check the results against what you get when you do the calculations by hand or on a calculator.

What is the mnemonic for addition in RISC-V:

Addi

What is meant by an immediate value?

A number

What is the mnemonic for multiplication in RISC-V?

mul

What is the purpose of the first register after the mnemonic?

A destination

### Create an initial program

You should download part19.asm from the Week 8 Resources area in Moodle as a starter code for this task. Alternatively, if you feel you can create this program from scratch you should do so.

* Create an assembly language program to do the following:
  + Store the decimal integer values **5** and **12** in registers **t3** and **t4** respectively.
  + Add the contents of those two registers together and store the result in register **s9**

Assemble and run the program, check the contents of the three registers used.

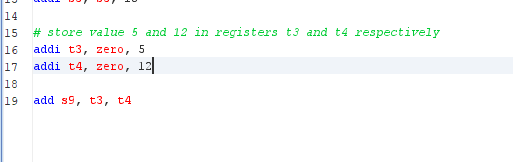
Expected Results:

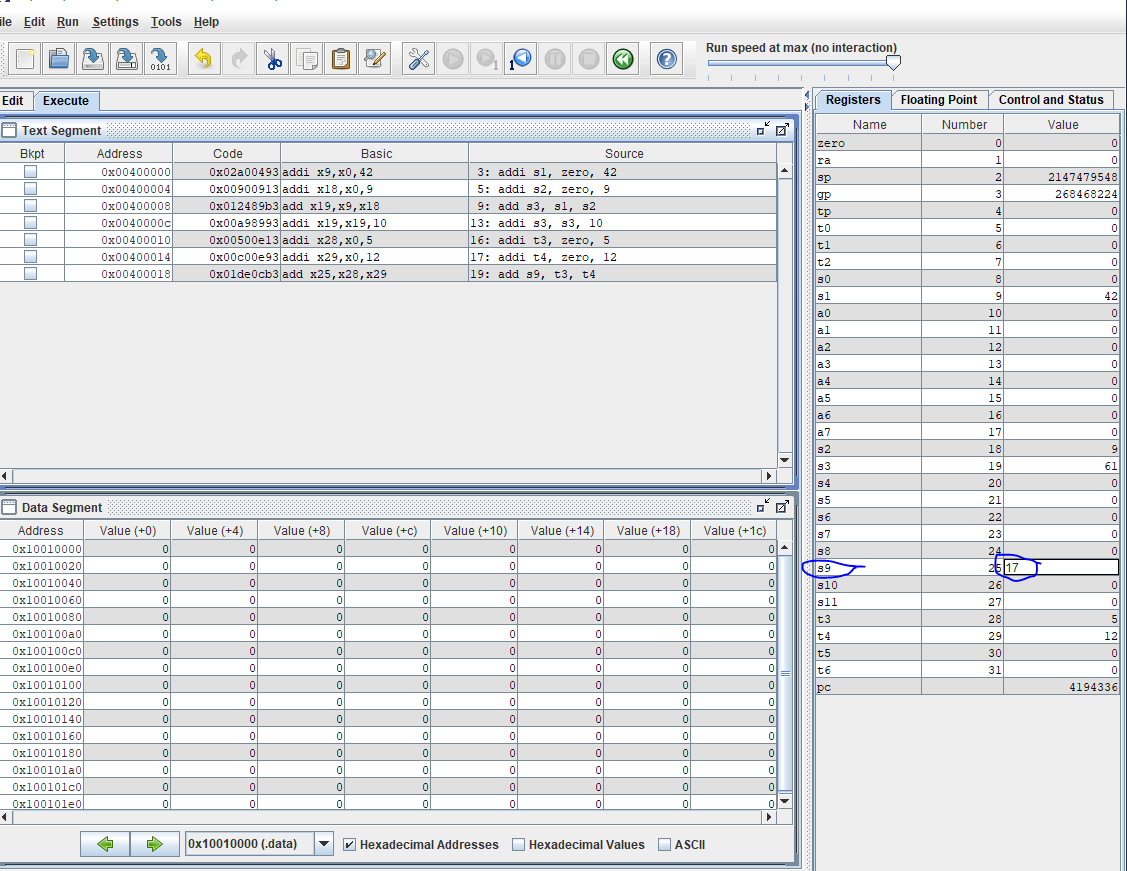
t3: 5

t4: 12

s9: 17

Paste in an image of the data stored inside your registers:





* Amend your program
  + Store a different number (of your choice) in register **t5**.
  + Add the contents of the three **t** registers (**t3**, **t4**, and **t5**) used and store the result in register **s10**.

Note that the add instruction can only add two registers together, so you will need more than one add instruction in your program.

Expected Results:

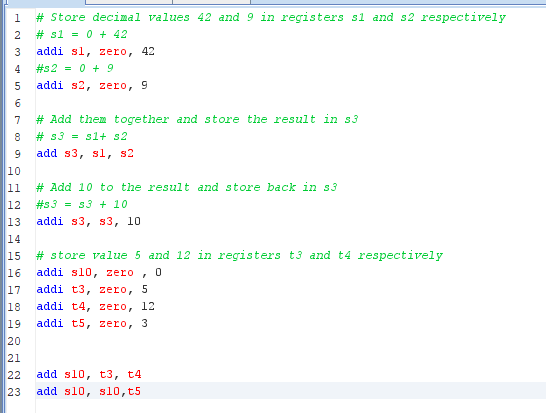
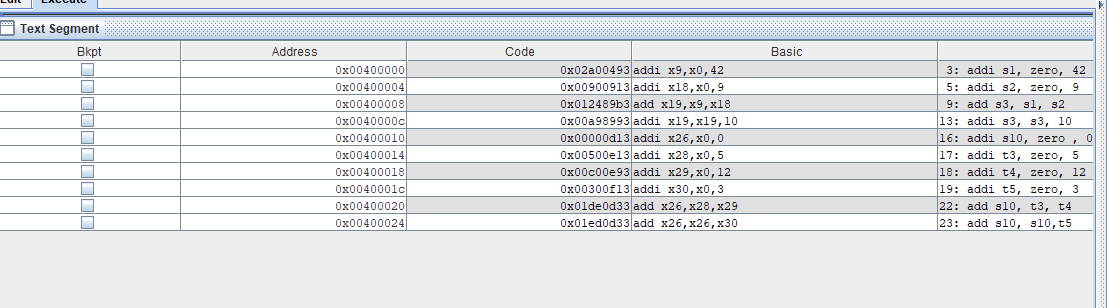
t3: 5

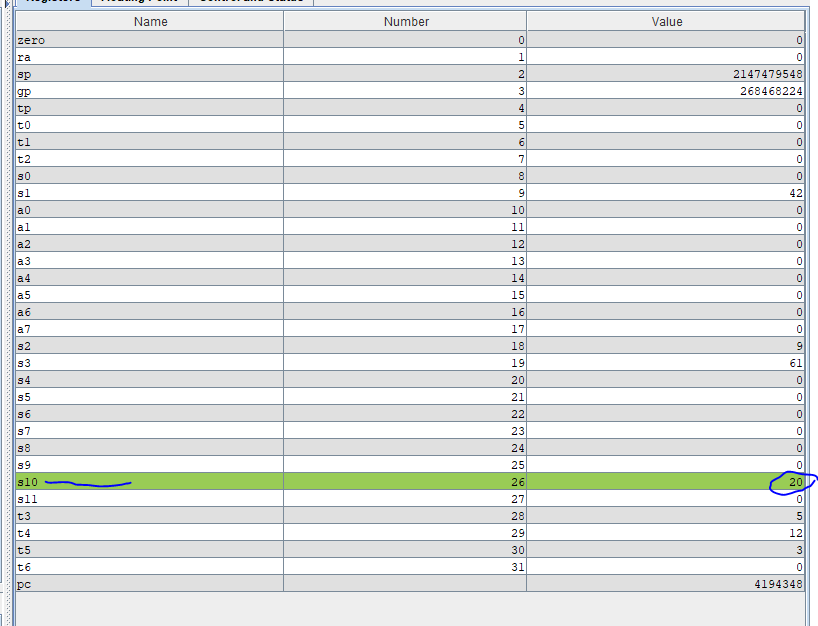
t4: 12

t5: 3

s10: 20

Paste an image of your code:



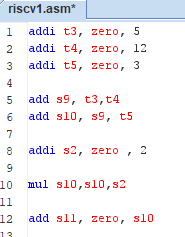
* Amend your program:
  + Multiply the value stored in the register **s10** by 2
  + Save this new value in the same register (**s10**)
  + Copy this value to the register **s11** so that both **s10** and **s11** have the same value

Expected Result:

S10: 40

S11: 40

Paste a screenshot of **both** your code and the values stored in the registers (you will need two screenshots):



* Create a new copy of your program and amend it:
  + Multiply the values from the three **t** registers together (instead of adding)
  + Store the result in a different **t** register of your choice.
  + Remove the lines of code that stores values inside **s10** and **s11**

Expected Result (please replace the ‘?’ with the register you have used):

t?:

Paste a screenshot of the data stored inside the different registers:

## Extensions:

* Write a code completely from scratch:
  + Store the values 10 and 5 in registers **t3** and **t4** respectively.
  + Create instructions which add, subtract, multiply and divide **t3** and **t4** and store the results of these arithmetic operations in 4 different registers

Note that when you subtract, you must add a minus number (e.g. add -2 is the same as subtract 2)

Expected Results (state the registers you have stored the values in):

+:

-:

÷:

×:

Paste a screenshot of your code and the values stored inside the registers: