Pre-LabT: Training Lab

Lab Objectives

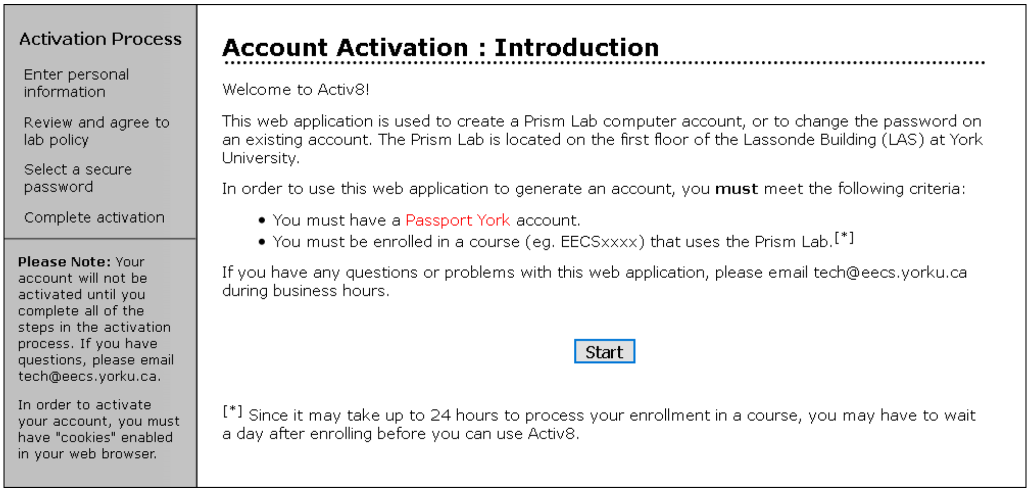
This lab introduces you to the RVS RISC-V simulator and to basic assembly programming. By the end of this lab, you should be able to:

1. Connecting to the EECS computing facilities
2. Use RVS to compile and run assembly programs.
3. Write a small straight line assembly program.
4. How to submit through eClass (Moodle)

Part 1: Connecting to the EECS servers

Open the course Moodle site in your browser and go to the **Labs** section. Follow the instructions for remote access to the EECS computing facilities.

1. Make sure that your EECS account is active and ready to use



Please Note: Your account will not be activated until you complete all of the steps in the activation process. If you have questions, please email tech@eecs.yorku.ca.

In order to activate your account, you must have "cookies" enabled in your web browser.

Account Activation : Introduction

Welcome to Activ8!

This web application is used to create a Prism Lab computer account, or to change the password on an existing account. The Prism Lab is located on the first floor of the Lassonde Building (LAS) at York University.

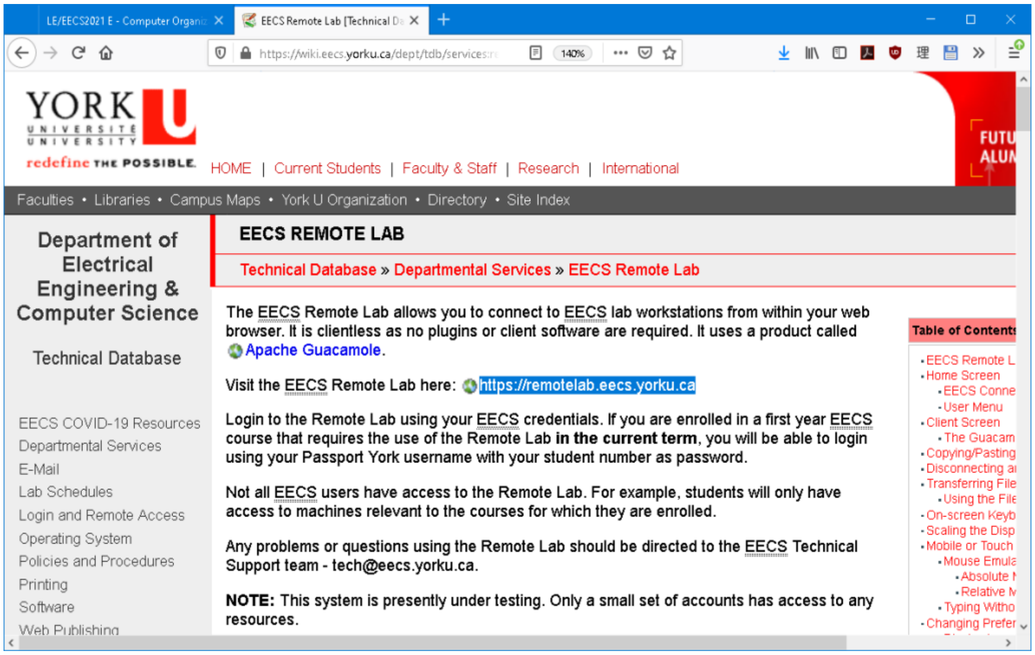
In order to use this web application to generate an account, you **must** meet the following criteria:

- You must have a Passport York account.
- You must be enrolled in a course (eg. EECSxxxx) that uses the Prism Lab.[*]

If you have any questions or problems with this web application, please email tech@eecs.yorku.ca during business hours.

[*] Since it may take up to 24 hours to process your enrollment in a course, you may have to wait a day after enrolling before you can use Activ8.

2. Read the REMOTE LAB (EECS) technical instructions



EECS REMOTE LAB

Technical Database » Departmental Services » EECS Remote Lab

The EECS Remote Lab allows you to connect to EECS lab workstations from within your web browser. It is clientless as no plugins or client software are required. It uses a product called Apache Guacamole.

Visit the EECS Remote Lab here: https://remotelab.eecs.yorku.ca

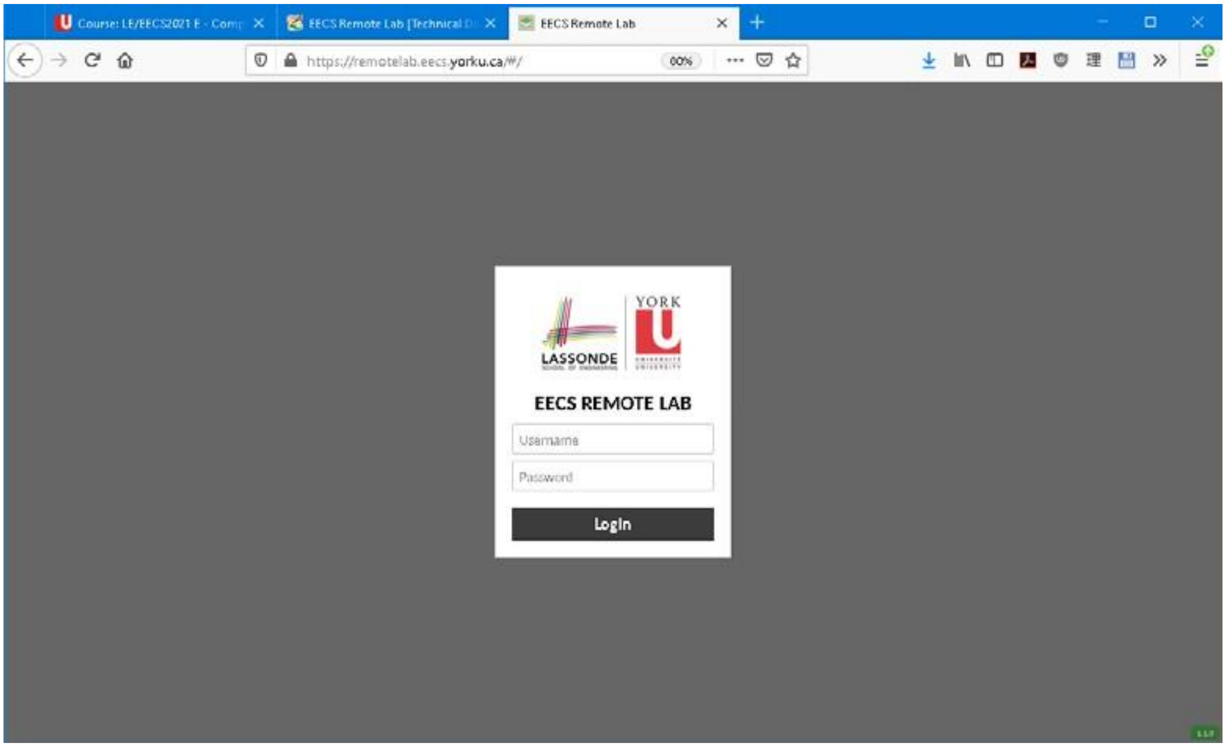
Login to the Remote Lab using your EECS credentials. If you are enrolled in a first year EECS course that requires the use of the Remote Lab **in the current term**, you will be able to login using your Passport York username with your student number as password.

Not all EECS users have access to the Remote Lab. For example, students will only have access to machines relevant to the courses for which they are enrolled.

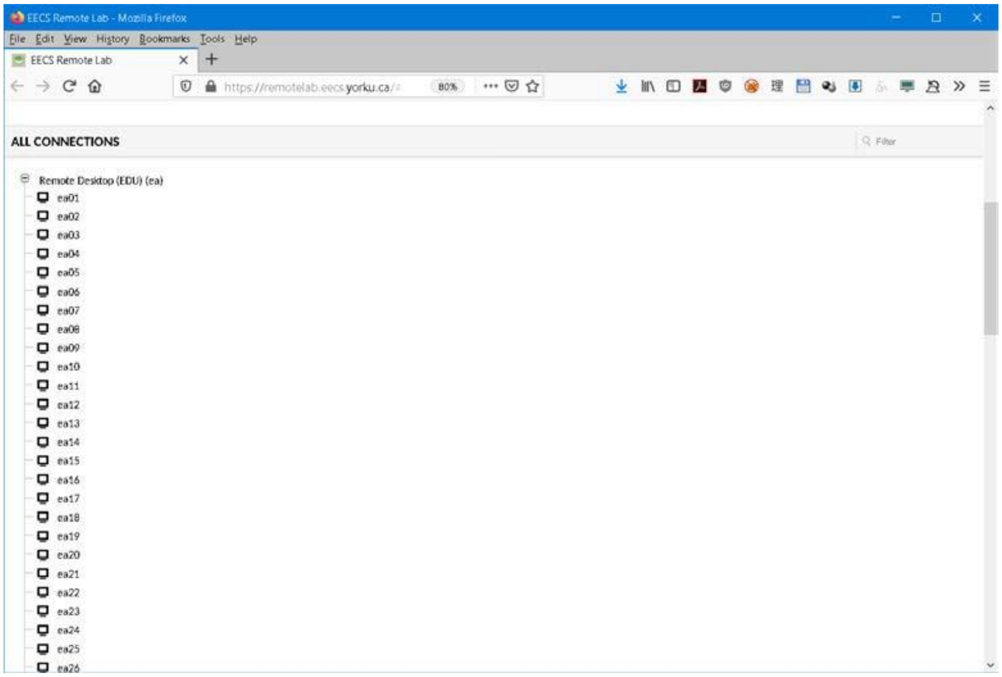
Any problems or questions using the Remote Lab should be directed to the EECS Technical Support team - tech@eecs.yorku.ca.

NOTE: This system is presently under testing. Only a small set of accounts has access to any resources.

3. Follow the provided link to visit the EECS Remote Lab


Course: LE/EECS2021 E - Comp
EECS Remote Lab [Technical D
EECS Remote Lab
https://remotelab.eecs.yorku.ca/#/
60%
LASSONDE
YORK UNIVERSITY
EECS REMOTE LAB
Username
Password
Login

Enter your EECS credentials from step 1 in the EECS REMOTE LAB login window as shown above. Any problems or questions with respect to the Remote Lab access should be directed to the EECS Technical Support team - tech@eecs.yorku.ca. In the following window, click on the **Remote Desktop (EDU) (ea) link** to see the list of the available workstations.

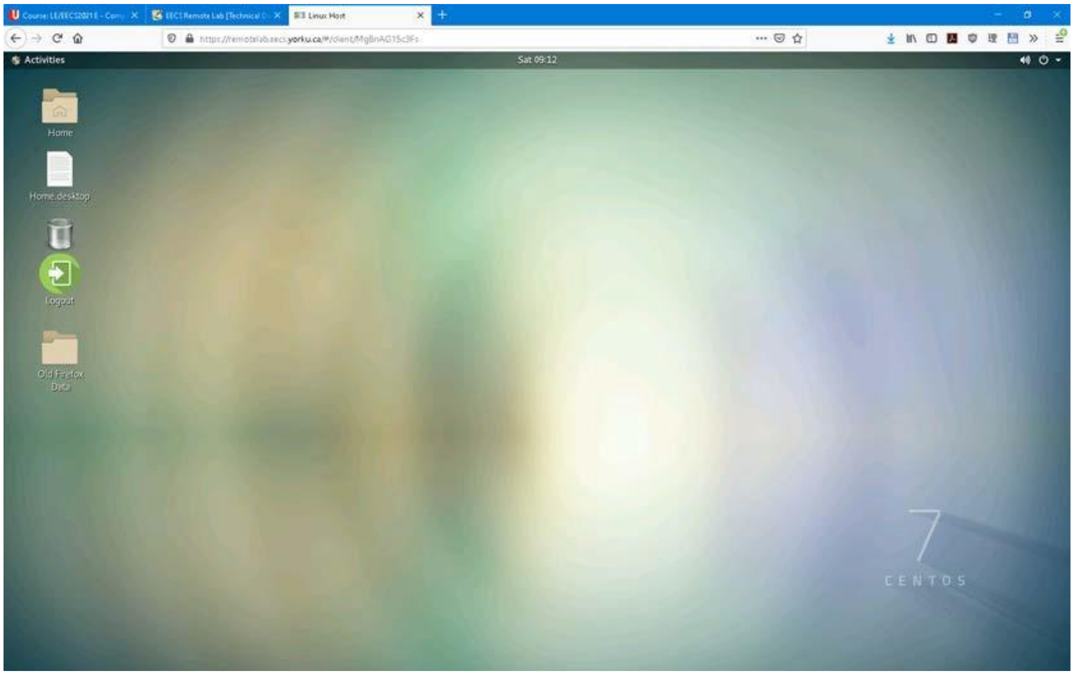

EECS Remote Lab - Mozilla Firefox
File Edit View History Bookmarks Tools Help
EECS Remote Lab
https://remotelab.eecs.yorku.ca/#
80%
Filter

ALL CONNECTIONS

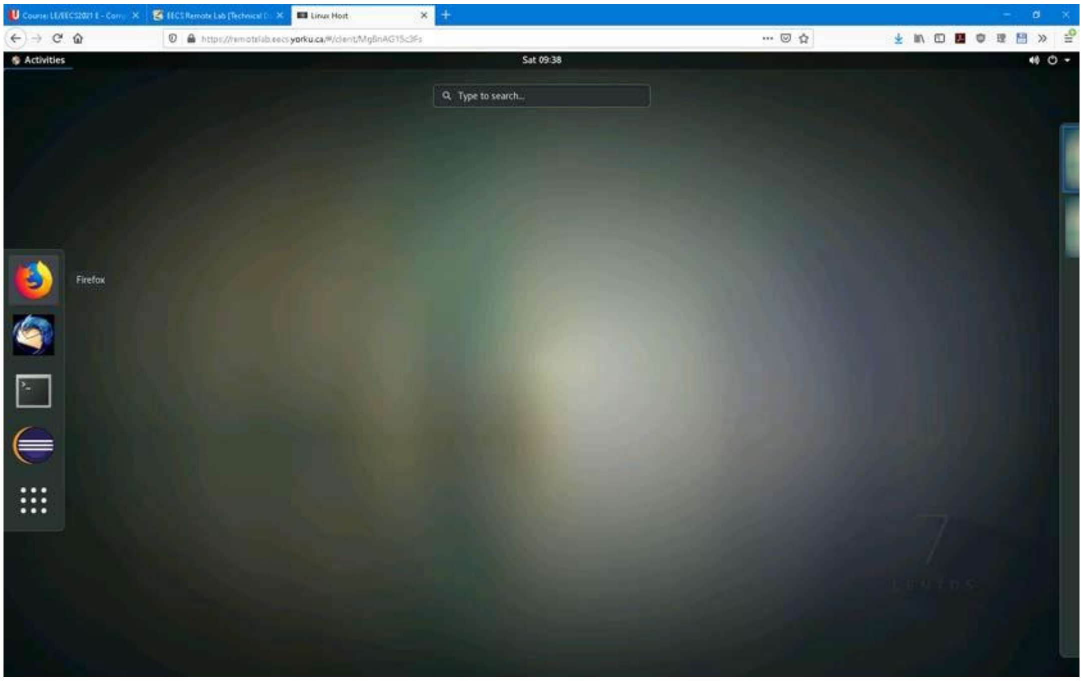
Remote Desktop (EDU) (ea)

- ea01
- ea02
- ea03
- ea04
- ea05
- ea06
- ea07
- ea08
- ea09
- ea10
- ea11
- ea12
- ea13
- ea14
- ea15
- ea16
- ea17
- ea18
- ea19
- ea20
- ea21
- ea22
- ea23
- ea24
- ea25
- ea26

Select a free workstation and click on it to connect. The Linux Desktop shows in your browser.

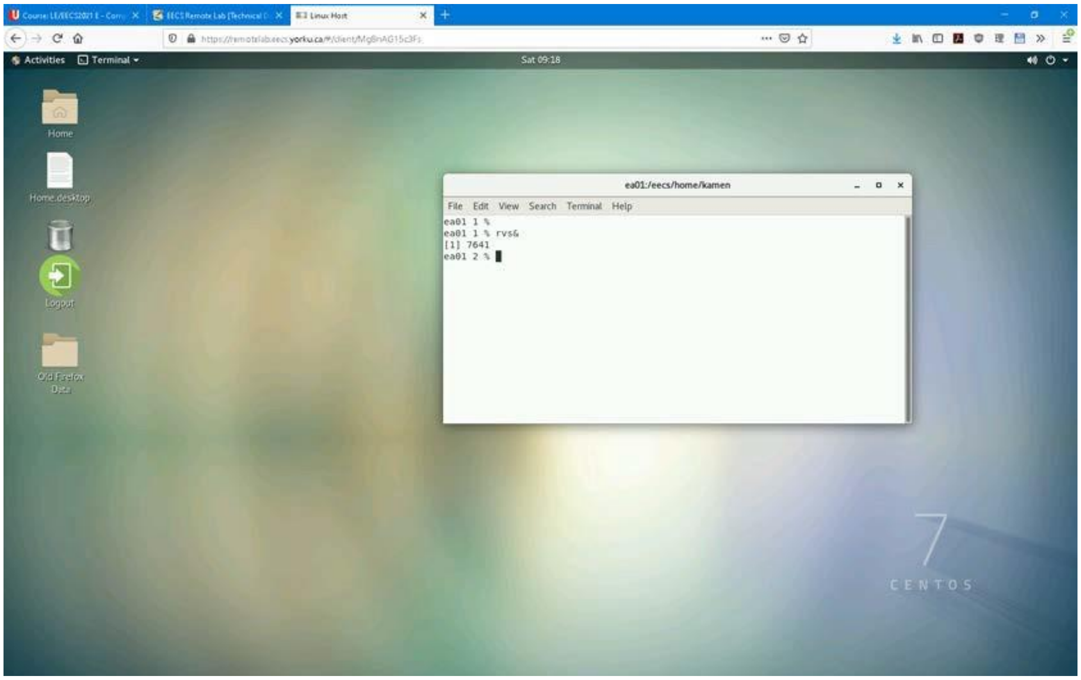

Course: LE/EECS2021 E - Comp
EECS Remote Lab (Technical D
Linux Host
https://remotelab.eecs.yorku.ca/#/client/MgBnAG15c3Fs
Activities
Sat 09:12
Home
Home.desktop
Logout
Old Firefox Data
7
CENTOS

You will use the above desktop to do all the course work. To start with, let's run Firefox. Click on the Activities menu in the top-left corner.



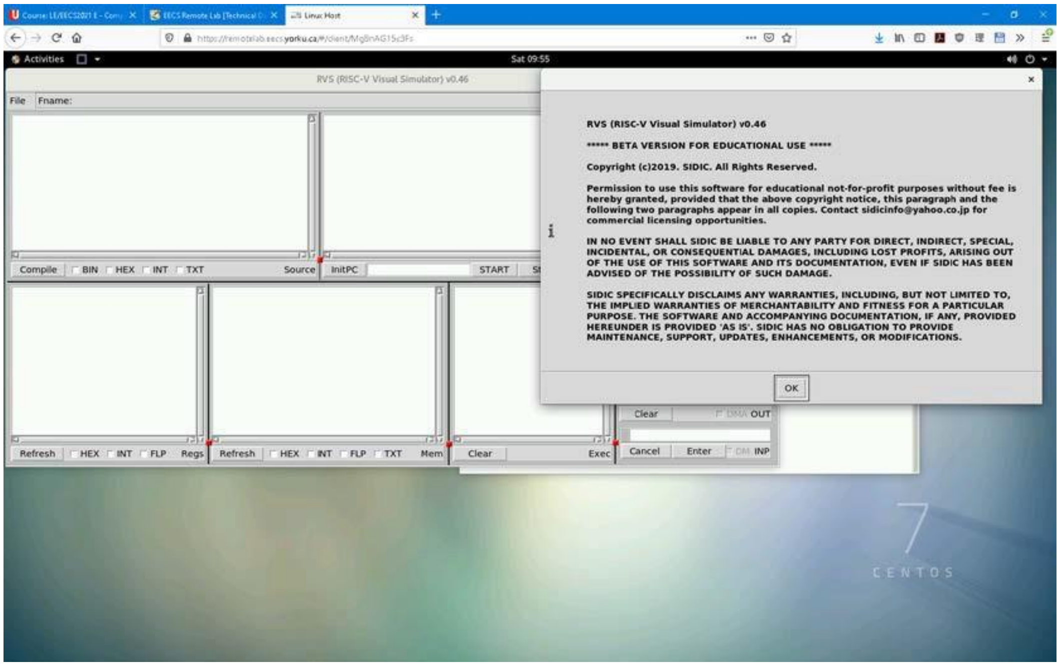
A column of icons is shown on the left- click on the top icon to start Firefox. Now you can open the course Moodle site in Firefox and follow the instructions and the links provided there.

You will also use the Linux Desktop to run the **RISC-V Visual Simulator (RVS)**. First clear the desktop by minimizing the Firefox window. Then **Right-click** on the desktop to show the menu and select **Open Terminal** in it. The Linux Desktop page with the terminal window should look as follows.



```
ea01 1 %
ea01 1 % rvs&
[1] 7641
ea01 2 %
```

In the Terminal window, type `rvs&` to start the RISC-V Visual Simulator.



RVS (RISC-V Visual Simulator) v0.46

***** BETA VERSION FOR EDUCATIONAL USE *****

Copyright (c)2019. SIDIC. All Rights Reserved.

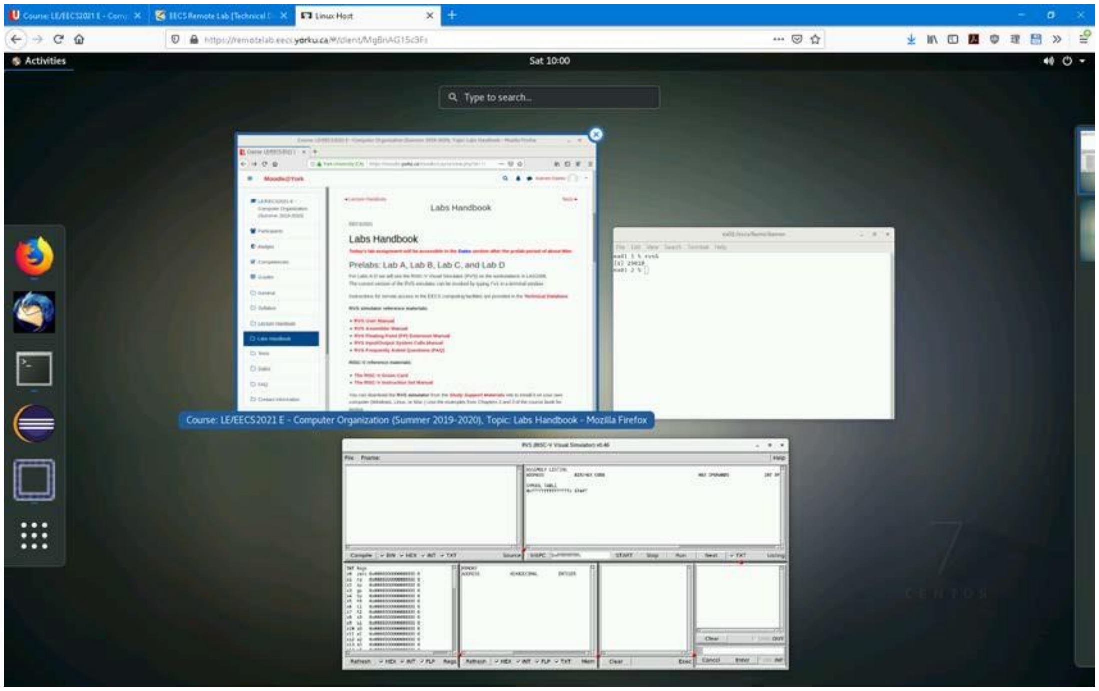
Permission to use this software for educational not-for-profit purposes without fee is hereby granted, provided that the above copyright notice, this paragraph and the following two paragraphs appear in all copies. Contact sidicinfo@yahoo.co.jp for commercial licensing opportunities.

IN NO EVENT SHALL SIDIC BE LIABLE TO ANY PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, INCLUDING LOST PROFITS, ARISING OUT OF THE USE OF THIS SOFTWARE AND ITS DOCUMENTATION, EVEN IF SIDIC HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

SIDIC SPECIFICALLY DISCLAIMS ANY WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE SOFTWARE AND ACCOMPANYING DOCUMENTATION, IF ANY, PROVIDED HEREUNDER IS PROVIDED 'AS IS'. SIDIC HAS NO OBLIGATION TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.

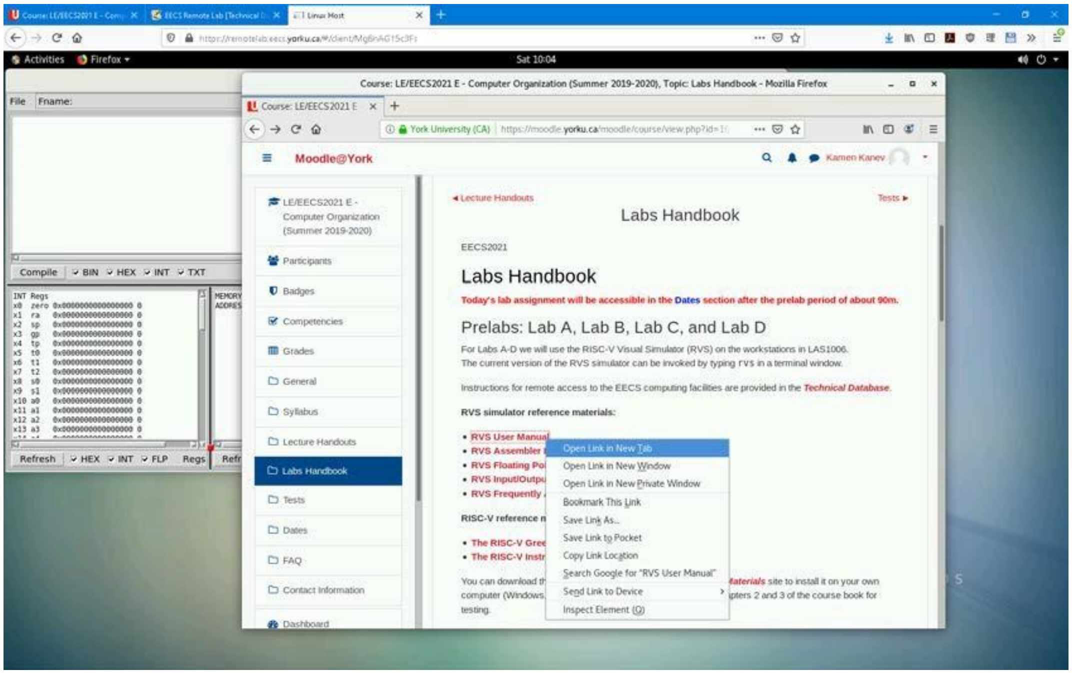
Read the notice and press OK to close it. The rvs is ready to use.

Now we want to bring back the Firefox window so that we can see the RVS reference materials. Click on the Activities menu and select the Firefox screen.



Activities
Sat 10:00
Type to search...
Labs Handbook
Course: LE/EECS2021 E - Computer Organization (Summer 2019-2020), Topic: Labs Handbook - Mozilla Firefox
RVS (RISC-V Visual Simulator) v0.46

Right-click on the RVS User Manual link to open the manual in a New Tab



Course: LE/EECS2021 E - Computer Organization (Summer 2019-2020), Topic: Labs Handbook - Mozilla Firefox

◄ Lecture Handouts

Tests ►

Labs Handbook

EECS2021

Labs Handbook

Today's lab assignment will be accessible in the Dates section after the prelab period of about 90m.

Prelabs: Lab A, Lab B, Lab C, and Lab D

For Labs A-D we will use the RISC-V Visual Simulator (RVS) on the workstations in LAS1006.
The current version of the RVS simulator can be invoked by typing `rvs` in a terminal window.

Instructions for remote access to the EECS computing facilities are provided in the *Technical Database*.

RVS simulator reference materials:

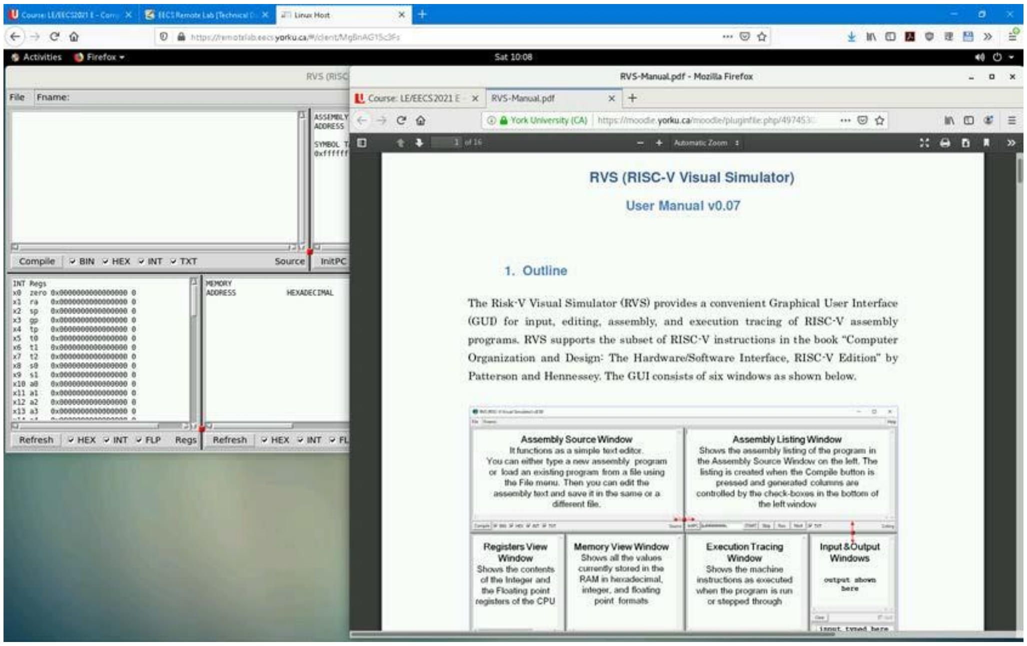
- **RVS User Manual**
- **RVS Assembler**
- **RVS Floating Poi**
- **RVS Input/Outpu**
- **RVS Frequently**

RISC-V reference m

- **The RISC-V Gree**
- **The RISC-V Instr**

You can download th ... *Materials* site to install it on your own computer (Windows, ... pters 2 and 3 of the course book for testing.

Arrange the RVS and the Firefox windows in a convenient way so that you can easily switch between them.



RVS-Manual.pdf - Mozilla Firefox

RVS (RISC-V Visual Simulator)

User Manual v0.07

1. Outline

The Risk-V Visual Simulator (RVS) provides a convenient Graphical User Interface (GUI) for input, editing, assembly, and execution tracing of RISC-V assembly programs. RVS supports the subset of RISC-V instructions in the book "Computer Organization and Design: The Hardware/Software Interface, RISC-V Edition" by Patterson and Hennessey. The GUI consists of six windows as shown below.

Assembly Source Window
It functions as a simple text editor. You can either type a new assembly program or load an existing program from a file using the File menu. Then you can edit the assembly text and save it in the same or a different file.

Assembly Listing Window
Shows the assembly listing of the program in the Assembly Source Window on the left. The listing is created when the Compile button is pressed and generated columns are controlled by the check-boxes in the bottom of the left window

Registers View Window
Shows the contents of the Integer and the Floating point registers of the CPU

Memory View Window
Shows all the values currently stored in the RAM in hexadecimal, integer, and floating point formats

Execution Tracing Window
Shows the machine instructions as executed when the program is run or stepped through

Input & Output Windows
output shown here
input typed here

When reading the manual, try the examples shown in it by running them in the RVS.

You can copy from the PDF file shown in Firefox (Ctrl-C) and paste into the Source window of the RVS (Ctrl-V). Pay attention to the space and new line character that may need manual re-entry.

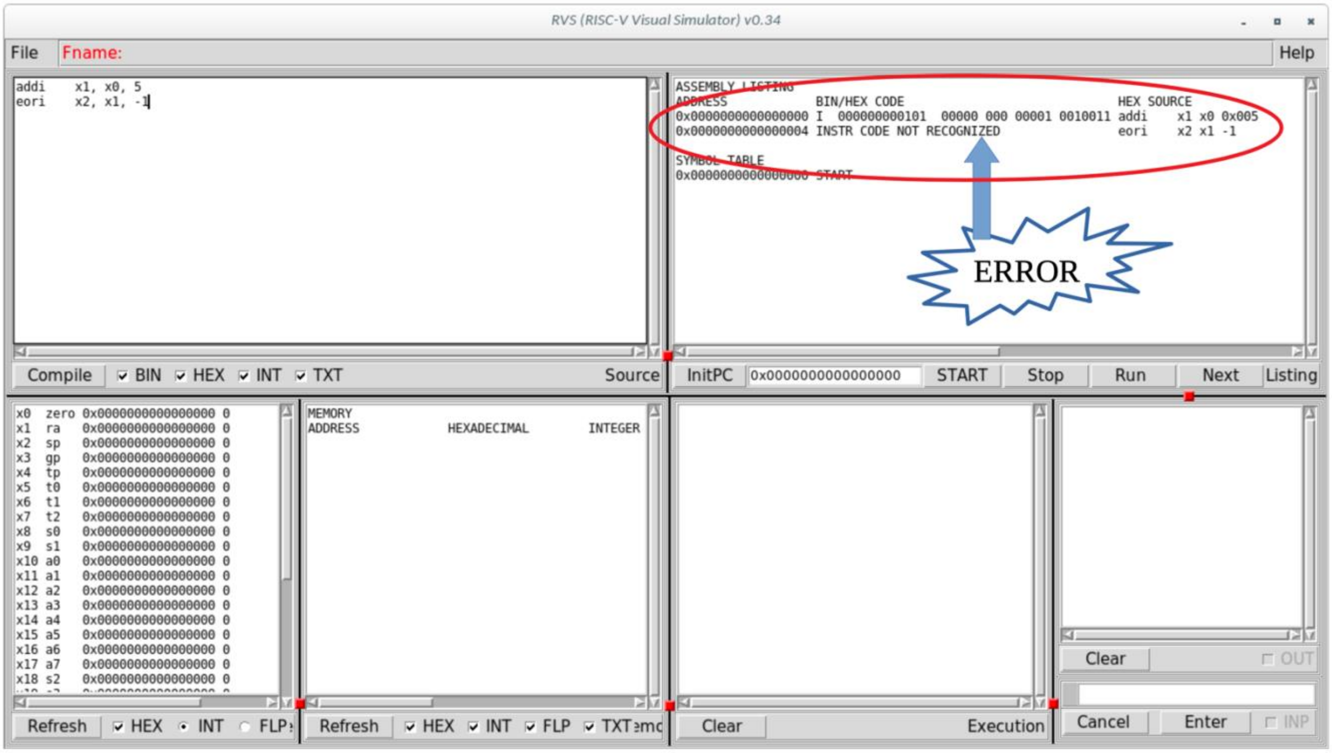
Part 2: Using RISC-V Visual Simulator

Syntax Errors

Before we start, let us try to run a program that contains syntax error.

Run the simulator and enter the following code in the source code window or load a text file containing the code shown below File → Load (the file must be terminated by a newline character, which most editors do anyway).

```
addi    x1, x0, 5
eoi     x2, x1, -1
```



```
addi    x1, x0, 5
eori    x2, x1, -1
```

```
ASSEMBLY LISTING
ADDRESS              BIN/HEX CODE                                 HEX SOURCE
0x0000000000000000 I  000000000101  00000 000 00001 0010011 addi    x1 x0 0x005
0x0000000000000004 INSTR CODE NOT RECOGNIZED                  eori    x2 x1 -1

SYMBOL TABLE
0x0000000000000000 START
```

```
x0  zero 0x0000000000000000 0
x1  ra   0x0000000000000000 0
x2  sp   0x0000000000000000 0
x3  gp   0x0000000000000000 0
x4  tp   0x0000000000000000 0
x5  t0   0x0000000000000000 0
x6  t1   0x0000000000000000 0
x7  t2   0x0000000000000000 0
x8  s0   0x0000000000000000 0
x9  s1   0x0000000000000000 0
x10 a0   0x0000000000000000 0
x11 a1   0x0000000000000000 0
x12 a2   0x0000000000000000 0
x13 a3   0x0000000000000000 0
x14 a4   0x0000000000000000 0
x15 a5   0x0000000000000000 0
x16 a6   0x0000000000000000 0
x17 a7   0x0000000000000000 0
x18 s2   0x0000000000000000 0
```

```
MEMORY
ADDRESS              HEXADECIMAL          INTEGER
```

The first line is addi x1, x0, 5 is a legitimate instruction, the second line is not (eori is not an instruction in RISC-V).

Click on “compile” and check the Assembly listing window (shown within red ellipse). The first line in the window (below “ADDRESS”) shows the memory location 0x0000 ..0 (since we did not specify where to put the code, it starts from location zero) and the representation of the first instruction

```
addi    x1, x0, 5
```

It shows the instruction type (I for Immediate) followed by the binary, hex, and text representation of the instruction.

The following line, is memory location 0x00..04 followed by “INSTR CODE NOT RECOGNIZED”. That is a syntax error, you have to go back and correct that instruction (eoi is xor in the ARM assembly language, easy mistake to do if you are familiar with ARM).

Register manipulation

Launch the simulator

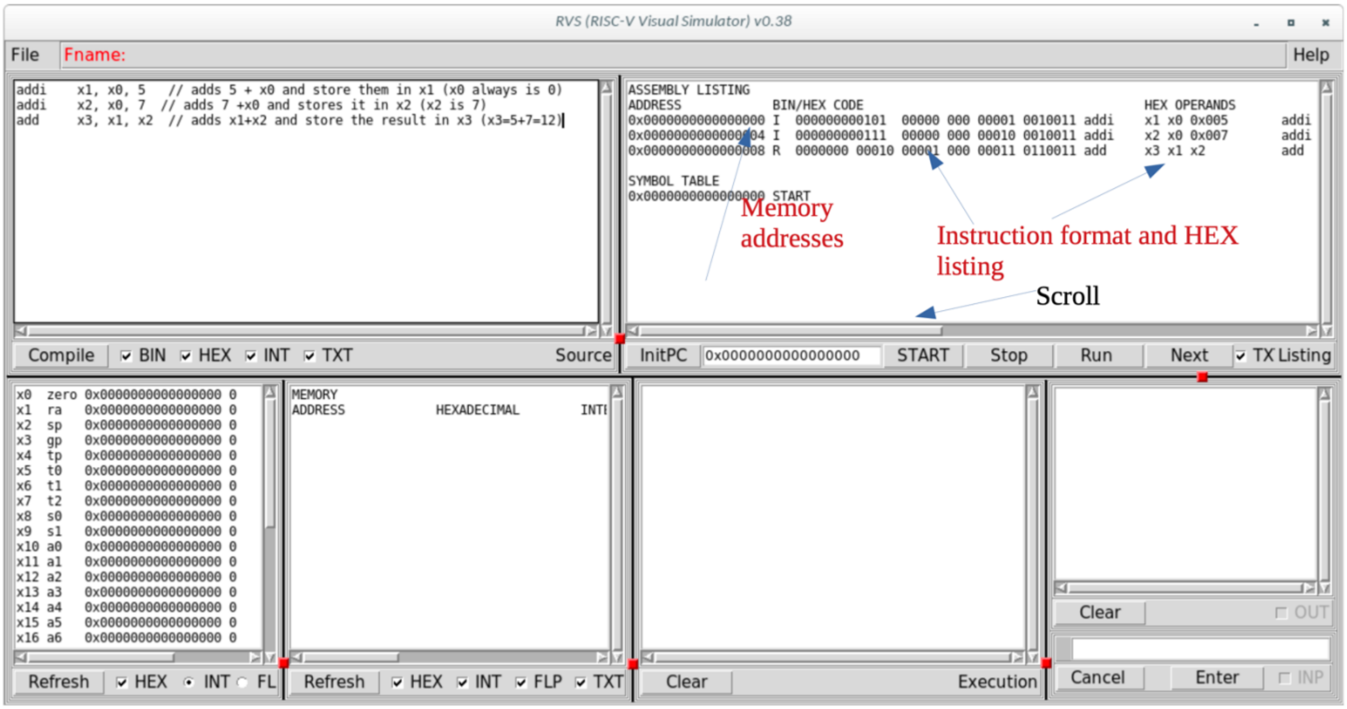
Then follow these steps for your first program/interaction with the simulator.

Write the following into the Assembly Source Window (or load a file contains the text below).

```
addi    x1, x0, 5       // adds 5 + x0 and store them in x1 (x0 always is 0)
addi    x2, x0, 7       // adds 7 +x0 and stores it in x2 (x2 is 7)
add     x3, x1, x2      // adds x1+x2 and store the result in x3 (x3=5+7=12)
```

click on compile.

The simulator window will look like this



```
addi    x1, x0, 5   // adds 5 + x0 and store them in x1 (x0 always is 0)
addi    x2, x0, 7  // adds 7 +x0 and stores it in x2 (x2 is 7)
add     x3, x1, x2  // adds x1+x2 and store the result in x3 (x3=5+7=12)
```

```
ASSEMBLY LISTING
ADDRESS            BIN/HEX CODE                                            HEX OPERANDS
0x0000000000000000 I  000000000101  00000 000 00001 0010011 addi    x1 x0 0x005      addi
0x0000000000000004 I  000000000111  00000 000 00010 0010011 addi    x2 x0 0x007      addi
0x0000000000000008 R  0000000 00010 00001 000 00011 0110011 add     x3 x1 x2         add

SYMBOL TABLE
0x0000000000000000 START
```

InitPC 0x0000000000000000

```
x0  zero 0x0000000000000000 0
x1  ra   0x0000000000000000 0
x2  sp   0x0000000000000000 0
x3  gp   0x0000000000000000 0
x4  tp   0x0000000000000000 0
x5  t0   0x0000000000000000 0
x6  t1   0x0000000000000000 0
x7  t2   0x0000000000000000 0
x8  s0   0x0000000000000000 0
x9  s1   0x0000000000000000 0
x10 a0   0x0000000000000000 0
x11 a1   0x0000000000000000 0
x12 a2   0x0000000000000000 0
x13 a3   0x0000000000000000 0
x14 a4   0x0000000000000000 0
x15 a5   0x0000000000000000 0
x16 a6   0x0000000000000000 0
```

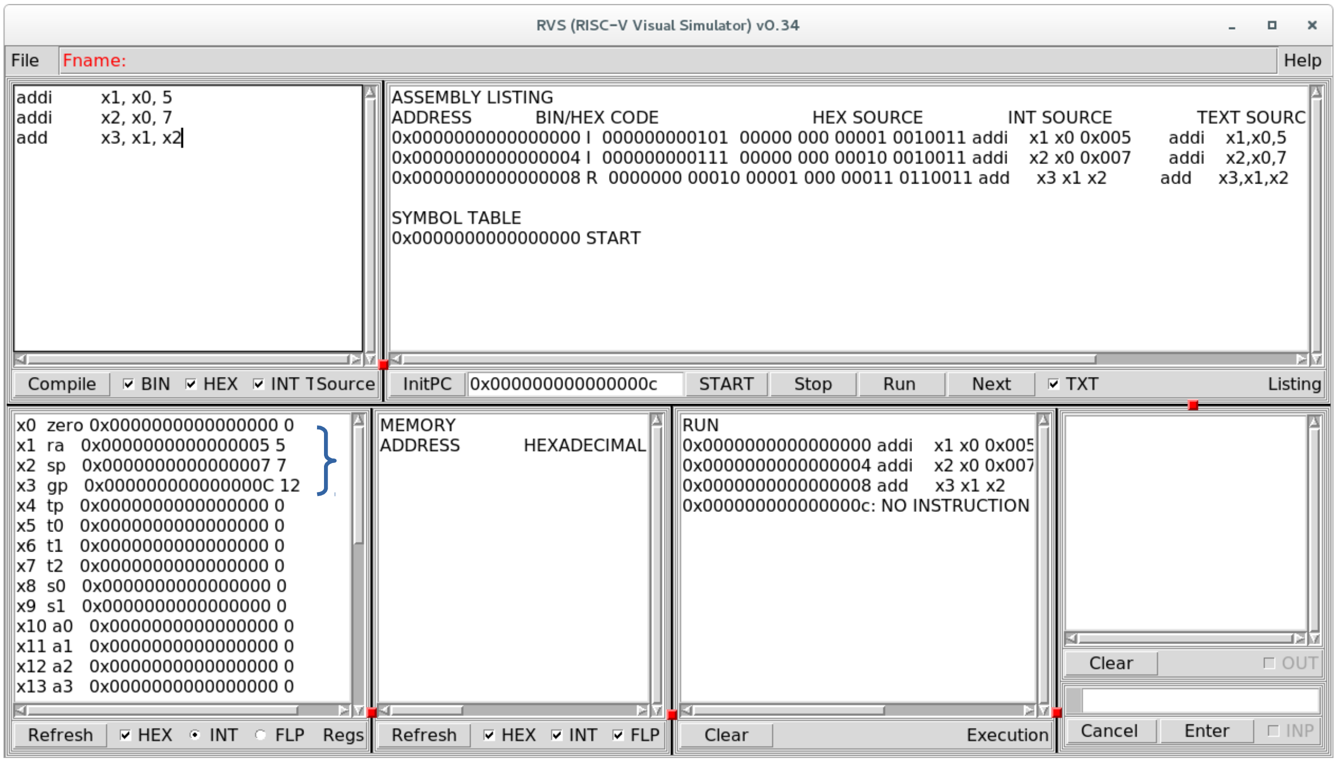
```
MEMORY
ADDRESS             HEXADECIMAL         INTE
```

Note that the register window shows the contents of all the 32 registers in the CPU. The registers are initialized to zero.

In the assembly listing window, you can see the memory addresses where the instructions are stored. Since we did not specify the starting memory address, the default is 0 (we can change this using ORG directive as we will see later). Three instructions are stored in locations 0, 4 and 8 respectively. If you remember from the class, RISC-V instructions are 32-bit each (4 bytes).

Under BIN/HEX CODE you can see the binary representation of the instructions showing the different fields as discussed in the class and the textbook. Before the representation, the instruction type is shown (The first two instructions show I, for immediate, and the third one is R for R-type or Register- Register ALU operation). Following that it shows the different operands for the three instructions in HEX. If you scroll to the right, you will see the operands in decimal.

Now, click on START or RUN, the program runs and the register values changes as shown in the following Figure.



```
addi        x1, x0, 5
addi        x2, x0, 7
add         x3, x1, x2
```

```
ASSEMBLY LISTING
ADDRESS           BIN/HEX CODE                        HEX SOURCE            INT SOURCE           TEXT SOURC
0x0000000000000000 I  000000000101  00000 000 00001 0010011 addi   x1 x0 0x005      addi   x1,x0,5
0x0000000000000004 I  000000000111  00000 000 00010 0010011 addi   x2 x0 0x007      addi   x2,x0,7
0x0000000000000008 R  0000000 00010 00001 000 00011 0110011 add    x3 x1 x2        add    x3,x1,x2

SYMBOL TABLE
0x0000000000000000 START
```

```
x0  zero 0x0000000000000000 0
x1  ra   0x0000000000000005 5
x2  sp   0x0000000000000007 7
x3  gp   0x000000000000000C 12
x4  tp   0x0000000000000000 0
x5  t0   0x0000000000000000 0
x6  t1   0x0000000000000000 0
x7  t2   0x0000000000000000 0
x8  s0   0x0000000000000000 0
x9  s1   0x0000000000000000 0
x10 a0   0x0000000000000000 0
x11 a1   0x0000000000000000 0
x12 a2   0x0000000000000000 0
x13 a3   0x0000000000000000 0
```

```
MEMORY
ADDRESS          HEXADECIMAL
```

```
RUN
0x0000000000000000 addi    x1 x0 0x005
0x0000000000000004 addi    x2 x0 0x007
0x0000000000000008 add     x3 x1 x2
0x000000000000000c: NO INSTRUCTION
```

Only the registers that we modified in the program have changed values (x1=5, x2=7, and x3=12).

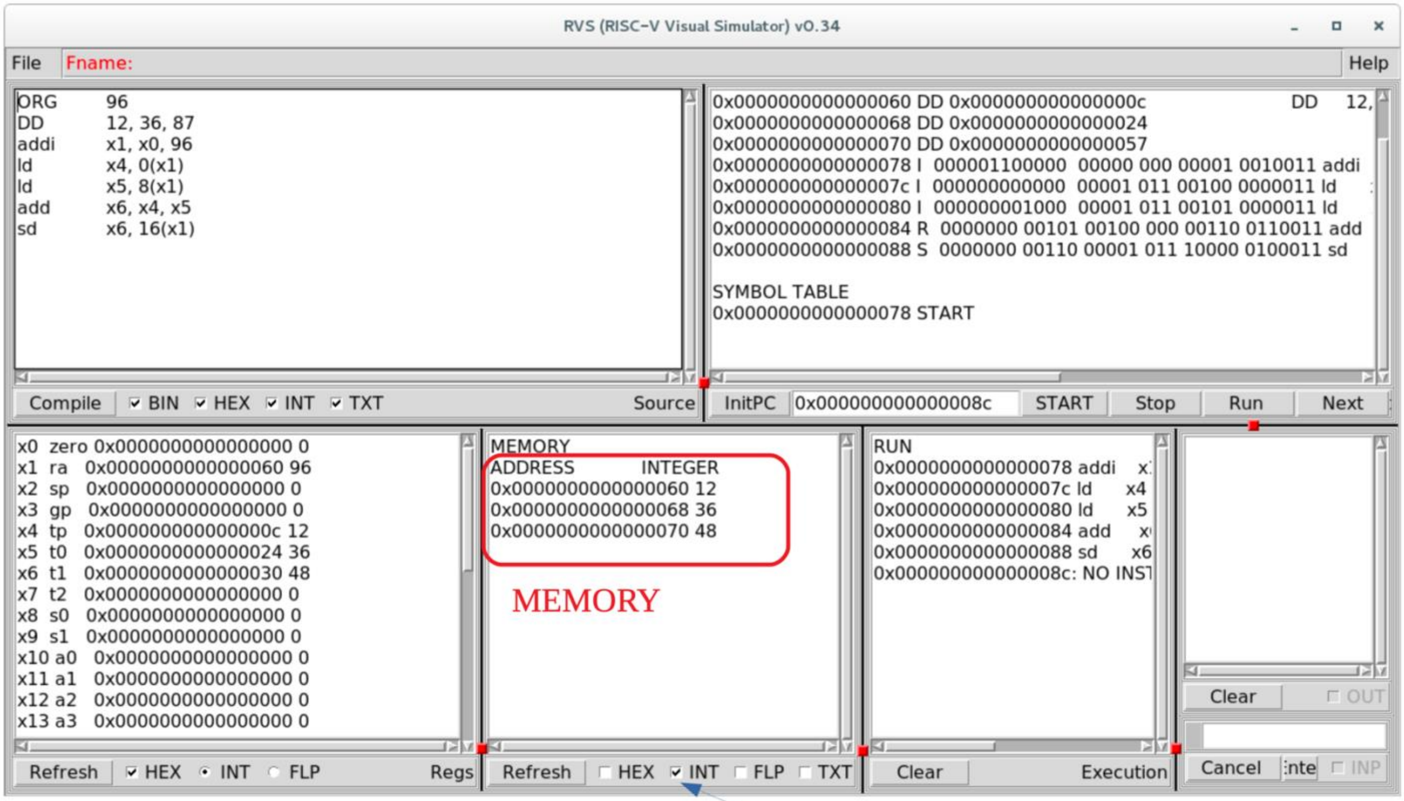
Memory manipulation

Launch the simulator

Enter the following code (or put it in a file and open it in the simulator)

```
ORG     96
DD      12, 36, 87
addi    x1, x0, 96
ld      x4, 0(x1)
ld      x5, 8(x1)
add     x6, x4, x5
sd      x6, 16(x1)
```

Compile and run the program. The window should look like this



RVS (RISC-V Visual Simulator) v0.34

File | Fname: | Help

```
ORG     96
DD      12, 36, 87
addi    x1, x0, 96
ld      x4, 0(x1)
ld      x5, 8(x1)
add     x6, x4, x5
sd      x6, 16(x1)
```

```
0x0000000000000060 DD 0x000000000000000c                     DD    12,
0x0000000000000068 DD 0x0000000000000024
0x0000000000000070 DD 0x0000000000000057
0x0000000000000078 I  000001100000  00000 000 00001 0010011 addi
0x000000000000007c I  000000000000  00001 011 00100 0000011 ld
0x0000000000000080 I  000000001000  00001 011 00101 0000011 ld
0x0000000000000084 R  0000000 00101 00100 000 00110 0110011 add
0x0000000000000088 S  0000000 00110 00001 011 10000 0100011 sd

SYMBOL TABLE
0x0000000000000078 START
```

Compile ☑ BIN ☑ HEX ☑ INT ☑ TXT Source

InitPC 0x000000000000008c START Stop Run Next

```
x0  zero 0x0000000000000000 0
x1  ra   0x0000000000000060 96
x2  sp   0x0000000000000000 0
x3  gp   0x0000000000000000 0
x4  tp   0x000000000000000c 12
x5  t0   0x0000000000000024 36
x6  t1   0x0000000000000030 48
x7  t2   0x0000000000000000 0
x8  s0   0x0000000000000000 0
x9  s1   0x0000000000000000 0
x10 a0   0x0000000000000000 0
x11 a1   0x0000000000000000 0
x12 a2   0x0000000000000000 0
x13 a3   0x0000000000000000 0
```

Refresh ☑ HEX ● INT ○ FLP Regs

```
MEMORY
ADDRESS            INTEGER
0x0000000000000060 12
0x0000000000000068 36
0x0000000000000070 48
```

MEMORY

Refresh □ HEX ☑ INT □ FLP □ TXT

```
RUN
0x0000000000000078 addi    x
0x000000000000007c ld      x4
0x0000000000000080 ld      x5
0x0000000000000084 add     x
0x0000000000000088 sd      x6
0x000000000000008c: NO INST
```

Clear Execution

Show the value in decimal

Show the value in decimal

Let us us take a close look at the simulator.

1. The first line

   ORG 96

   States that you start at location 96 (hex = 0x60). That basically means the memory assignment will start at location 96. So, the next line data or instruction will be in location 96.

2. The second statement

```
DD      12, 36, 87
```

Define three double words 12, 36, and 87 and stores them in locations 96, 96+8=104, 104+8=112 (in hex 0x60, 0x68, 0x70).

Now, look at the memory contents window, it shows the contents of the memory.

```
0x0000000000000060          12
```

In memory location 0x60 (decimal 96) the contents is the integer value 12. The next two lines shown the two integers 36 and 87 in memory location 0x68 and 0x70 (decimal 104 and 112). The above figure does not show the value 87, since it has changed in the program as we will see soon.

The values is shown in decimal since we clicked the INT box, if you click on HEX, the value will be shown both in decimal and hex

3. The next line is

```
addi    x1, x0, 96
```

It adds the contents of x0 (by definition 0) to the integer 96 and stores the result in x1. The register window shows the content of x1 to be 96. sine the value of x1 is 96, we can say that x1 is pointing to the memory where we started storing data (96 or 0X60).

4. The following line is

```
ld      x4, 0(x1)
```

It loads a doubleword from location 0+contents of X1 REG(x1) that is 96, to register x4. The instruction loads the contents of memory location 96 (0x60) into register x4. From statement 2 above, we stored 12 in location 96. Now x4 contains 12.

5. The following line is

```
ld      x5, 8(x1)
```

it loads from location 8+REG(x1), i.e. 8+96=104 (in hex 0x68) into register x5. If you look at the memory window, in location 0x96 there is 36. now x5 contains 36 (check the register window to be sure it contains 36, if not, you made a mistake. Find out your mistake and correct it).

6. The following line is

```
add     x6, x4, x5
```

Adds REG(x4), that is 12, to REG(x5), that is 36, and the result is stored in x6. If you check the contents of register x6 in the register memory, you find it 48.

7. The following line is

```
sd      x6, 16(x1)
```

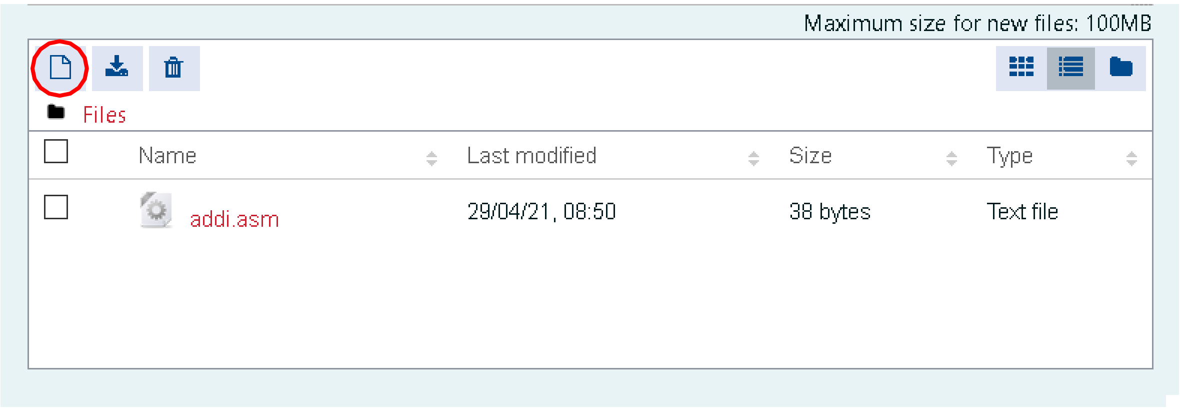
Stores the contents of x6 (that is 48) in memory location pointed to by the contents of x1 + 16 (REG(x1)+16 = 96+16 = 112, in hex 0x70). Now check location 0x70 (or 112 decimal) in the memory window it is 48.

Part 2: Submitting your work

Go back to the **Labs** section in Moodle and read the **LAB REPORTS** part. Following the instructions click on the **LabT Report** link then start the quiz which will include a link for the Lab book for the lab and a possible Quiz.

Upload the completed the file part of the lab work.

Locate the files in the appropriate folder in Remote Lab then drag and drop it to the blue arrow in the file drop area. You should see the following screen:

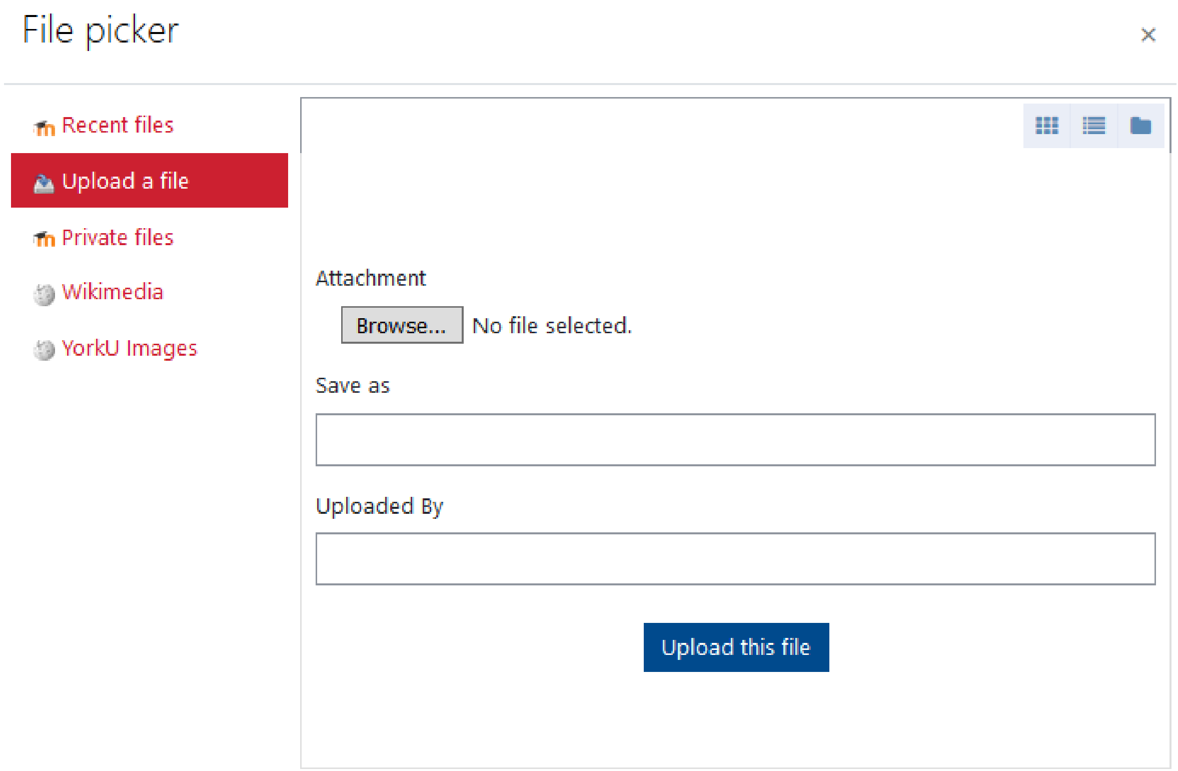


Maximum size for new files: 100MB

Files

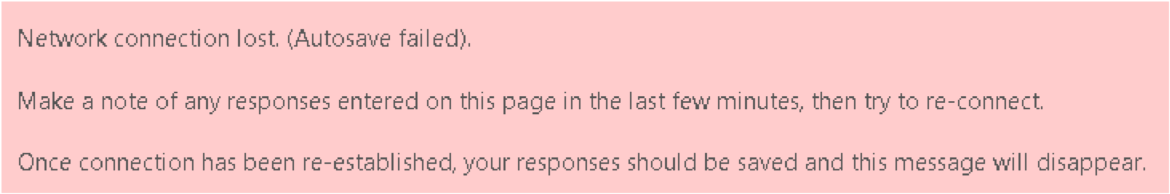
□	Name	Last modified	Size	Type
□	addi.asm	29/04/21, 08:50	38 bytes	Text file

Alternatively you can click on the **Add...** icon in the top left corner of the file drop window (encircled in red) and use the **File Picker** interface shown below.



With **Upload a file** selected, you can click on the **Browse...** button and select your file using the graphical interface shown below.

The submission is automatic and happens every few minutes but problems may arise if your network connection is unstable as indicated by the following message:



Network connection lost. (Autosave failed).

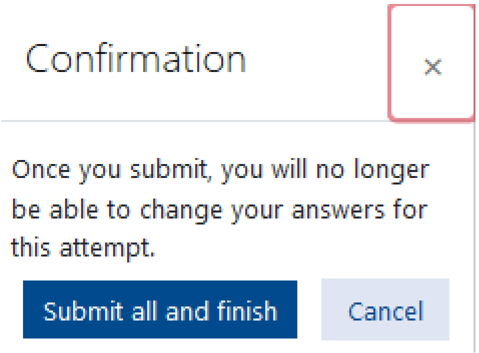
Make a note of any responses entered on this page in the last few minutes, then try to re-connect.

Once connection has been re-established, your responses should be saved and this message will disappear.

To avoid such problems you can initiate an immediate submit of the current content by clicking on the **Finish attempt ...** button. Then click on the **Return to attempt** button to get back to the previous screen and continue with the preparation of your lab report.

With this, the explanation of the submit procedure is complete. Please follow this procedure whenever you are instructed in the lab book to `submit` a file.

At the end of each lab you should submit your completed Lab Report by clicking on the **Finish attempt ...** button and then on the **Submit all and finish ...** button. The following small confirmation window will show up:



Confirmation

Once you submit, you will no longer be able to change your answers for this attempt.

Clicking on the **Submit all and finish** button will finalize your lab report submission. Right afterwards you will be given a chance to review your final submission. Check the content and the attachments of your lab report and click on the **Finish review** link to complete your review.

Important: While multiple submissions are possible, your most recently submitted report for each lab will be the one that is eligible for consideration. Therefore you must do your final lab report submission before the submission deadline for each of the labs.