# CREATOR: a tool for teaching assembly programming with RISC-V

Felix Garcia-Carballeira[1]*, Alejandro Calderon[1],
Diego Camarmas-Alonso[1] and Elias Del-Pozo-Puñal[1]

[1]Computer Science and Engineering Departament Universidad Carlos III de Madrid. Spain

**Abstract**

*This paper introduces CREATOR, a simulator specially designed for teaching assembly programming. CREATOR allows to define different instruction sets and to edit, and execute assembly programs. This simulator includes more than 100 instructions of the RV32IMFD specification. CREATOR runs directly in web browsers without needing a server and is adapted to run on different devices (desktops, tablets, and smartphones). Unlike other simulators, it includes error detection in the parameter passing convention with alerts if the convention is violated, it helps the creation of teaching materials with the possibility of obtaining a URL that allows the execution of the simulator with a given program, and it offers capabilities to extend the instruction set with new instructions and pseudo-instructions that allows the student to experiment with further instructions.*

## Introduction

This paper presents the features offered by the CREATOR simulator for teaching RISC-V assembly programming [1]. CREATOR (didaCtic and geneRic assEmbly prograMming simulaTOR) is a didactic tool for teaching assembly programming (`https://creatorsim.github.io/creator/`). It has the following main features:

- Didactic simulator specially designed for students and teachers.
- Multiplatform; It runs on a web browser without a server (desktops, tablets, and smartphones).
- An integrated environment for editing, compiling and executing assembly programs.
- Possibility to define and work with different architectures and assembly languages.

Although there are several simulators for teaching use [2], [3], [4], and [5], CREATOR differs from other simulators in offering very visual functionalities that help students understand how data is stored in memory and how to use the parameter passing convention in subroutine calls properly.

From a teaching point of view, CREATOR provides functionalities to help students to understand:

- The representation of data and instructions in memory.
- The difference between instructions and pseudo-instructions.
- How a program is loaded into memory.
- The execution flow of an assembly program, knowing at all times the current and next instruction (useful in loops and branch instructions).
- The parameter passing convention and stack usage with alerts when it is not respected.
- The concept of function library and how it is used.

## Instruction set supported

CREATOR offers the RV32IMFD specification instruction set with more than 100 instructions and pseudo-instructions:

- Data transfer: `li`, `mv`, `lui`.
- Arithmetic and logical over integer registers: `addi`, `add`, `and`, ...
- Arithmetic on floating point numbers (single and double): `fadd.s`, `fmul.d`, ...
- Jump instructions (integer): `beq`, `bne`, ...
- Comparison instructions (integer and floating point): `slt`, `feq.s`, ...
- Transfer instructions between integer and floating-point registers: `fmv.w.x`.
- Function calls and system calls: `jal`, `jr`, `ecall`.
- Memory access (integer and floating point): `lb`, `lw`, `flw`, `fsd`,....
- Conversion operations (integer and floating point): `fcvt.w.s`, ...
- Others: Floating point classification: `fclass.s`, `fclass.d`; Cycle counter: `rdcyle`

## Simulator features

The simulator allows the editing and compilation of programs written in RISC-V assembler in an integrated environment running on the web. Figure 1 shows the view of a program loaded in memory and the values stored in the registers. Figure 2 shows the view of

*Corresponding author: `felix.garcia@uc3m.es`

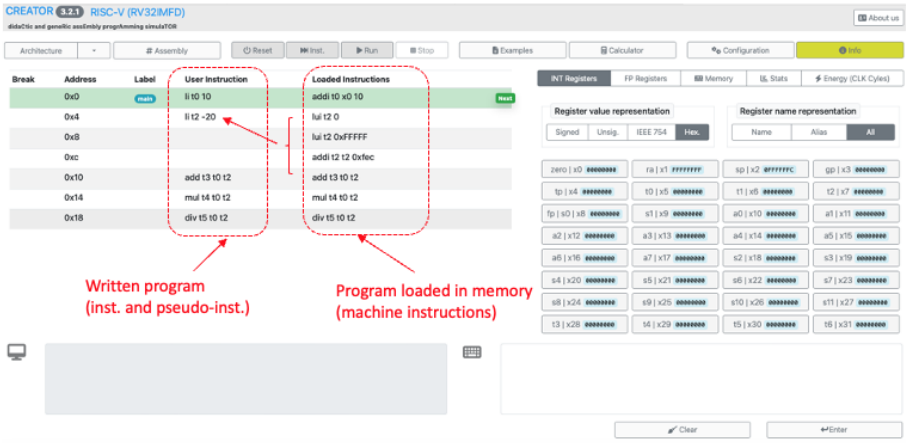


**Figure 1:** *CREATOR: main screen with loaded program in memory*

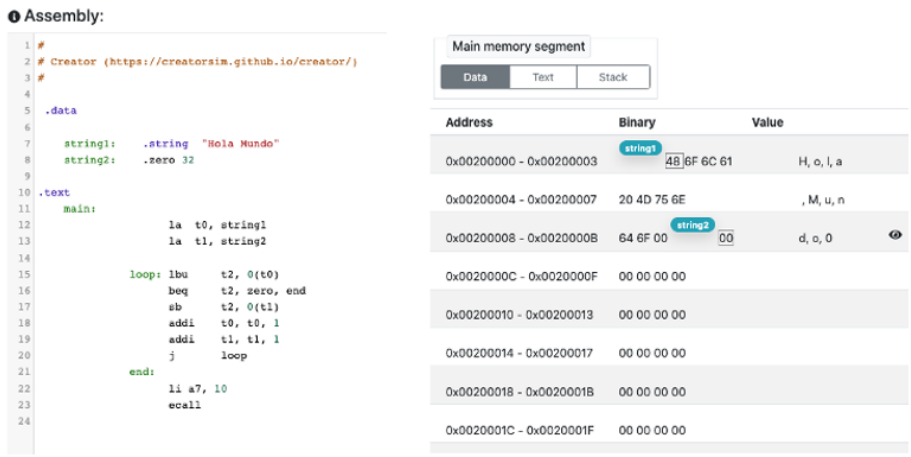


**Figure 2:** *CREATOR: data segment view details panel*

the data segment, with the values stored in memory, and how the tags defined in this data segment are associated with a memory address.

The simulator also includes the following features:

- Display of data in registers and memory in different formats (hexadecimal, signed decimal, unsigned, and floating point).
- A keyboard and screen for data input/output and display.
- Error detection at runtime (unaligned data, stack pointer with the address of data or text segment, writing in the text segment, ...).
- Error detection in the parameter passing convention with alerts if the convention is violated (stack not correctly freed, improper use of registers in functions, e.g. not saving on the stack the `s` registers used in a function).
- Creation and loading of function libraries, allowing students to understand the need to follow a convention in passing parameters to functions.
- Use of the browser cache as a backup mechanism for retrieving work done by students.
- Support for checking laboratory practices from the command-line with scripting: the simulator can be executed from the command-line.
- Support for creating teaching materials with the possibility of obtaining a URL that allows the execution of the simulator with a given program.
- Ability to extend the instruction set with new instructions and pseudo-instructions, allowing the student to experiment with further instructions.

## References

[1] *The CREATOR RISC-V Simulator*. accessed March 17, 2023. 2023. URL: `creatorsim.github.io/creator/`.

[2] *RISC-V Simulator*. accessed March 17, 2023. 2023. URL: `https://ascslab.org/research/briscv/simulator/simulator.html`.

[3] *Venus RISC-V Simulator*. accessed March 17, 2023. 2023. URL: `https://thaumicmekanism.github.io/venus/`.

[4] *Vulcan RISC-V Simulator for Education*. accessed March 17, 2023. 2023. URL: https://vmmc2.github.io/vulcan/#/editor.

[5] *RARS RISC-V Assembler and Runtime Simulator*. accessed March 17, 2023. 2023. URL: https://github.com/TheThirdOne/rars.