

XIII Seminario de Invierno CAPAP-H, Almería, 1, 2 y 3 de febrero de 2023

CREATOR como herramienta docente para la enseñanza de la programación en ensamblador con RISC V

Félix García Carballeira

felix.garcia@uc3m.es

Material

<https://github.com/fgcarbal/Creator/>



Motivación de Creator didaCtic and geneRic assEmbly progrAmming simulaTOR

- ▶ Simulador didáctico para la enseñanza de la programación en ensamblador
 - ▶ Centrado en los estudiantes y profesores
- ▶ Multiplataforma
 - ▶ Ejecución en web sin servidor (sobremesa, tablets y móviles)
- ▶ Entorno integrado (edición, compilación y simulación de programas)
- ▶ Posibilidad de definir y trabajar con diferentes arquitecturas y lenguajes ensamblador
 - ▶ Características básicas (nº de bits, registros, ...)
 - ▶ Instrucciones
 - ▶ Pseudoinstrucciones
 - ▶ Directivas



Creator desde el punto de vista docente

- ▶ Facilidades para entender:
 - ▶ La representación de datos e instrucciones
 - ▶ La diferencia entre instrucciones y pseudoinstrucciones
 - ▶ La carga de un programa en memoria
 - ▶ El flujo de ejecución de un programa en ensamblador conociendo en todo momento la instrucción en curso y la siguiente (útil en bucles)
 - ▶ El convenio de paso de parámetros y uso de pila con alertas cuando no se respeta
 - ▶ El concepto de biblioteca de funciones y su uso

CREATOR

didaCtic and geneRic assEmbly progrAmming simulaTOR

Access to CREATOR

The screenshot shows the CREATOR interface for a RISC-V-like processor. The main window displays a table of loaded instructions with columns for Break, Address, Label, User Instruction, and Loaded Instructions. The loaded instructions include addi, jal, li, and ecall. The current instruction at address 0x34 is add a0 t1 zero. To the right, there is a register file view showing PC, SR, and various general-purpose registers (ra, sp, gp, etc.) with their current values. Below the table, a large button says "Execute assembly programs".

Break	Address	Label	User Instruction	Loaded Instructions
	0xc			addi a1 a1 0xfs3
	0x10	li a2 45		addi a2 x0 45
	0x14	jal x1 sum		jal x1 0x2c
	0x18	jal x1 sub		jal x1 0x40
	0x1c	li a7 1		addi a7 x0 1
	0x20	ecall		ecall
	0x24	li a7 10		addi a7 x0 10
	0x28	ecall		ecall
	0x2c	add t1 a0 a1		add t1 a0 a1
	0x30	add t2 a2 a2		add t2 a2 a2
	0x34	add a0 t1 zero		add a0 t1 zero
	0x38	add a1 t2 zero		add a1 t2 zero
	0x3c	jr ra		jalr x0 0 (ra)
	0x40	sub a0 a0 a1		sub a0 a0 a1

Register value representation: Register name representation:

Signed	Unsigned	IEEE 754	Hexadecimal	Name	Alias	All
zero 0	ra 24	sp 208405482	gp 0			
tp 0	10 0	11 -34	12 99			
10 0	s1 0	a0 23	a1 -77			
12 48	a2 0	a4 0	a5 0			
a6 0	a7 0	s2 0	s3 0			
s4 0	s5 0	s6 0	s7 0			
s8 0	s9 0	s10 0	s11 0			
13 0	14 208405482	15 0	16 0			

Execute assembly programs

Clear andStart

<https://creatorsim.github.io/>



Características

- ▶ Permite describir las características de una arquitectura y su juego de instrucciones
 - ▶ Actualmente: MIPS-32, [RISC-V \(RV32IMFD\)](#)
- ▶ Editar y compilar programas en ensamblador del juego de instrucciones elegido
- ▶ Ejecutar/depurar programas en ensamblador en un mismo entorno
- ▶ Obtener estadísticas sobre los programas ejecutados
- ▶ Ejecución en navegador

Supported Browsers



Contenido: empleo de CREATOR con RISC-V

- ▶ Juego de instrucciones soportado
- ▶ Visión del estudiante:
 - ▶ Características del entorno
 - ▶ Edición y compilación de programas
 - ▶ Ejecución y depuración de programas
 - ▶ Bibliotecas de funciones
 - ▶ Facilidades para entender el empleo de funciones y uso de pila
- ▶ Visión del profesor:
 - ▶ Soporte a la corrección de prácticas
 - ▶ Soporte a la creación de material didáctico
 - ▶ Capacidades para extender el juego de instrucciones y crear nuevas arquitecturas

Disponibilidad

The screenshot shows the CREATOR simulator interface. At the top, there's a navigation bar with links for Introduction, Help, Supported Browsers, Publications, Evolution, and Authors. The main title is "didaCtic and geneRic assEmbly progrAmming simulaTOR". Below the title is a blue button labeled "Access to CREATOR". The central area contains a terminal-like window displaying assembly code. The code includes instructions like `addi \$t0, \$t1, 10` and `add \$t2, \$t3, \$t4`. A status bar at the bottom indicates "Execute assembly programs".

<https://creatorsim.github.io>

The screenshot shows the RISC-V Exchange website. At the top, there's a navigation bar with links for About RISC-V, Membership, RISC-V Exchange, Technical, News & Events, Community, and a search bar. The main title is "RISC-V Exchange". Below the title, it says "The RISC-V Exchange hosts the hardware, software, services, and learning offerings in the RISC-V community. Browse the list or search for an offering below." There are two buttons: "SUBMIT AN ITEM" and "CONTACT US". Below this is a search bar with the placeholder "Search Exchange..." and a filter section with checkboxes for Hardware, Cores, Software, Services, and Learning. On the right, there are two cards: one for "CREATOR Simulator" and another for "emmtrix Parallel Studio".

<https://riscv.org/exchange>



Juego de instrucciones soportado (RV32IMFD)

98 instrucciones y pseudoinstrucciones

[Guía de referencia](#)

- ▶ Transferencia de datos: `li`, `mv`, `lui`
- ▶ Aritméticas y lógicas sobre registros de enteros: `addi`, `add`, `and`, ...
- ▶ Aritméticas sobre números en coma flotante (float y double): `fadd.s`, `fmul.d`, ...
- ▶ Instrucciones de salto (registros enteros): `beq`, `bne`, ...
- ▶ Instrucciones de comparación (enteros y coma flotante): `slt`, `feq.s`, ...
- ▶ Instrucciones de transferencia entre registros enteros y coma flotante: `fmv.w.x`,
- ▶ Llamadas a funciones y llamadas al sistema: `jal`, `jr`, `ecall`
- ▶ Acceso a memoria (enteros y coma flotante): `lb`, `lw`, `flw`, `fsd`, ...
- ▶ Operaciones de conversión (enteros y coma flotante): `fcvt.w.s`, ...
- ▶ Otras:
 - ▶ Clasificación de coma flotante: `fclass.s`, `fclass.d`
 - ▶ Contador de ciclos: `rdcycle`



Registros

Integer Registers	
Register Name	Usage
zero	Constant 0
ra	Return address (routines/functions)
sp	Stack pointer
gp	Global pointer
tp	Thread pointer
t0..t6	Temporary (NOT preserved across calls)
s0..s11	Saved temporary (preserved across calls)
a0, a1	Arguments for functions / return value
a2..a7	Arguments for functions
Floating-point registers	
ft0..ft11	Temporary (NOT preserved across calls)
fs0..fs11	Saved temporary (preserved across calls)
fa0, fa1	Arguments for functions / return value
fa2..fa7	Arguments for functions

Llamadas al sistema

System Calls (ecall)			
Service	Call Code (a7)	Arguments	Result
Print_int	1	a0 = integer	
Print_float	2	fa0 = float	
Print_double	3	fa0 = double	
Print_string	4	a0 = string addr	
Read_int	5		Integer in a0
Read_float	6		Float in fa0
Read_double	7		Double in fa0
Read_string	8	a0 = string addr a1 = length	
Sbrk	9	a0 = length	Address in a0
Exit	10		
Print_char	11	a0 = ASCII code	
Read_char	12		Char in a0

Directivas soportadas

Directivas	Uso
.data	Siguientes elementos van al segmento de dato
.text	Siguientes elementos van al segmento de código
.ascii "tira de caracteres"	Almacena cadena caracteres NO terminada en carácter nulo
.string "tira de caracteres"	Almacena cadena caracteres terminada en carácter nulo
.byte 1, 2, 3	Almacena bytes en memoria consecutivamente
.half 300, 301, 302	Almacena medias palabras en memoria consecutivamente
.word 800000, 800001	Almacena palabras en memoria consecutivamente
.float 1.23, 2.13	Almacena float en memoria consecutivamente
.double 3.0e21	Almacena double en memoria consecutivamente
.zero 10	Reserva un espacio de 10 bytes en el segmento actual
.align n	Alinea el siguiente dato en un límite de 2^n

CREATOR (RISC-V)

CREATOR 3.2
didaCtic and geneRic assEmbly progrAmming simulaTOR

ARCOS uc3m Universidad Carlos III de Madrid
Computer Science and Engineering Department

Configuration Info

RISC-V (RV32IMFD)

RISC-V is an instruction set architecture (ISA) whose design is based on the RISC type and its hardware is free. This architecture was created in 2010 at the University of California, Berkeley.

MIPS-32

The MIPS processor was developed by Dr.John Hennessey and his graduate students at Stanford University in the early 1980s. It is currently one of the major processors in the embedded processor market.

Load Architecture

Allows to load the definition of an already created architecture.

New Architecture

Allows you to define an architecture from scratch.

Pantalla inicial

CREATOR 3.2 RISC-V (RV32IMFD)
didactic and generic assembly programming simulator

ARCOS uc3m Universidad Carlos III de Madrid
Computer Science and Engineering Department

Architecture # Assembly Reset Inst. Run Stop Examples Calculator Configuration Info

Break Address Label User Instruction Loaded Instructions

INT Registers FP Registers Memory Stats Energy (CLK Cycles)

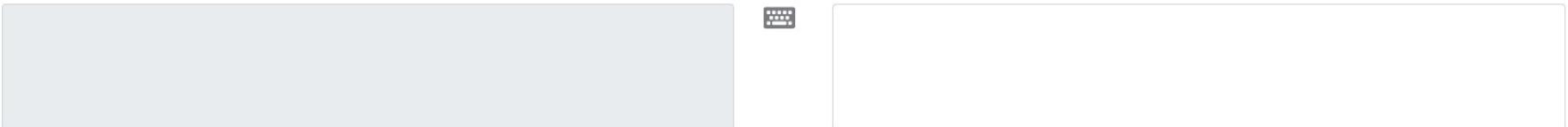
Register value representation Register name representation

Signed Unsig. IEEE 754 Hex.

Name Alias All

zero x0 00000000	ra x1 FFFFFFFF	sp x2 0FFFFFFC	gp x3 00000000
tp x4 00000000	t0 x5 00000000	t1 x6 00000000	t2 x7 00000000
fp s0 x8 00000000	s1 x9 00000000	a0 x10 00000000	a1 x11 00000000
a2 x12 00000000	a3 x13 00000000	a4 x14 00000000	a5 x15 00000000
a6 x16 00000000	a7 x17 00000000	s2 x18 00000000	s3 x19 00000000
s4 x20 00000000	s5 x21 00000000	s6 x22 00000000	s7 x23 00000000
s8 x24 00000000	s9 x25 00000000	s10 x26 00000000	s11 x27 00000000
t3 x28 00000000	t4 x29 00000000	t5 x30 00000000	t6 x31 00000000

Clear Enter



Elección de la arquitectura

CREATOR 3.2 RISC-V (RV32IMFD)
didaCtic and generic assembly programming simulator

ARCOS uc3m Universidad Carlos III de Madrid
Computer Science and Engineering Department

Architecture # Assembly Reset Inst. Run Stop Examples Calculator Configuration Info

RISC-V (RV32IMFD)
MIPS-32
New Architecture

Label User Instruction Loaded Instructions

INT Registers FP Registers Memory Stats Energy (CLK Cycles)

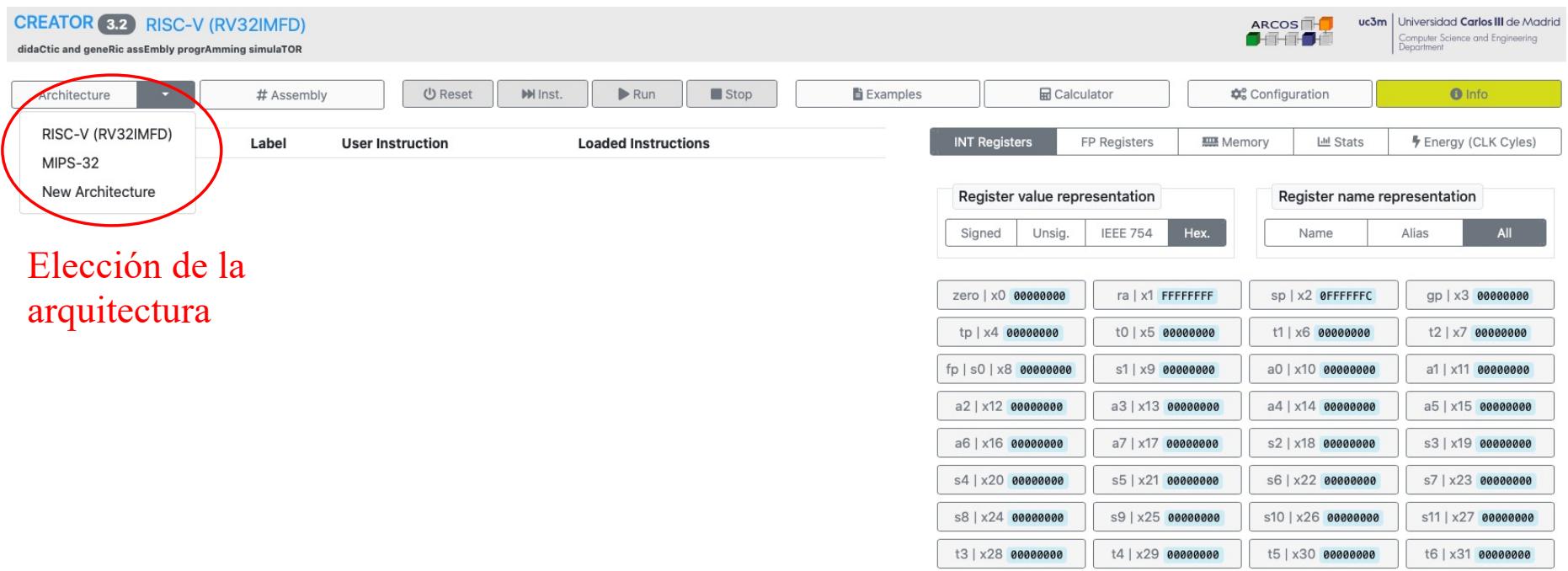
Register value representation Register name representation

Signed Unsig. IEEE 754 Hex.

Name Alias All

zero x0 00000000	ra x1 FFFFFFFF	sp x2 0FFFFFFF	gp x3 00000000
tp x4 00000000	t0 x5 00000000	t1 x6 00000000	t2 x7 00000000
fp s0 x8 00000000	s1 x9 00000000	a0 x10 00000000	a1 x11 00000000
a2 x12 00000000	a3 x13 00000000	a4 x14 00000000	a5 x15 00000000
a6 x16 00000000	a7 x17 00000000	s2 x18 00000000	s3 x19 00000000
s4 x20 00000000	s5 x21 00000000	s6 x22 00000000	s7 x23 00000000
s8 x24 00000000	s9 x25 00000000	s10 x26 00000000	s11 x27 00000000
t3 x28 00000000	t4 x29 00000000	t5 x30 00000000	t6 x31 00000000

Clear Enter



The screenshot shows the ARCOS Creator 3.2 software interface. At the top, there's a navigation bar with tabs for Architecture, # Assembly, Reset, Inst., Run, Stop, Examples, Calculator, Configuration, and Info. Below the navigation bar, there's a dropdown menu labeled "Architecture" which is currently set to "RISC-V (RV32IMFD)". This dropdown is circled in red. There are also options for "MIPS-32" and "New Architecture". To the right of the dropdown, there are tabs for INT Registers, FP Registers, Memory, Stats, and Energy (CLK Cycles). Further down, there are two sections: "Register value representation" and "Register name representation". The "Register value representation" section has tabs for Signed, Unsig., IEEE 754, and Hex. The "Register name representation" section has tabs for Name, Alias, and All. Below these sections is a large table displaying register values. At the bottom of the interface, there are icons for monitor, keyboard, and mouse, followed by "Clear" and "Enter" buttons.



Edición de programas

The screenshot shows the 'CREATOR 3.2 RISC-V (RV32IMFD)' interface. At the top, there's a navigation bar with tabs: Architecture, # Assembly (circled in red), Reset, Run, Stop, Examples, Calculator, Configuration, and Info. Below the navigation bar is a toolbar with Break, Address, Label, User instruction, and Loaded Instructions. On the right, there are tabs for INT Registers, FP Registers, Memory, Stats, and Energy (CLK Cycles). The main area contains two sections: 'Register value representation' and 'Register name representation'. The 'Register value representation' section has buttons for Signed, Unsigned, IEEE 754, and Hex. The 'Register name representation' section has buttons for Name, Alias, and All. Below these sections is a grid of 32x4 boxes showing register values. At the bottom, there are icons for monitor and keyboard, and buttons for Clear and Enter.

CREATOR 3.2 RISC-V (RV32IMFD)

didaCtic and geneRic assEmbly progrAmming simulaTOR

ARCOS uc3m Universidad Carlos III de Madrid
Computer Science and Engineering Department

Architecture # Assembly Reset Run Stop Examples Calculator Configuration Info

Break Address Label User instruction Loaded Instructions

INT Registers FP Registers Memory Stats Energy (CLK Cycles)

Register value representation Register name representation

Signed Unsigned IEEE 754 Hex

Name Alias All

zero x0 0000000	ra x1 FFFFFFFF	sp x2 0FFFFFFC	gp x3 0000000
tp x4 0000000	t0 x5 0000000	t1 x6 0000000	t2 x7 0000000
fp s0 x8 0000000	s1 x9 0000000	a0 x10 0000000	a1 x11 0000000
a2 x12 0000000	a3 x13 0000000	a4 x14 0000000	a5 x15 0000000
a6 x16 0000000	a7 x17 0000000	s2 x18 0000000	s3 x19 0000000
s4 x20 0000000	s5 x21 0000000	s6 x22 0000000	s7 x23 0000000
s8 x24 0000000	s9 x25 0000000	s10 x26 0000000	s11 x27 0000000
t3 x28 0000000	t4 x29 0000000	t5 x30 0000000	t6 x31 0000000

Edición de programas

Clear Enter



Control de la ejecución

The screenshot shows the 'CREATOR 3.2 RISC-V (RV32IMFD)' simulation interface. A red circle highlights the control buttons at the top: 'Reset' (with a power icon), 'Inst.' (with a play icon), 'Run' (with a right-pointing arrow icon), and 'Stop' (with a square icon). Below these buttons is a toolbar with links to 'Examples', 'Calculator', 'Configuration', and 'Info'. The main workspace has tabs for 'Break', 'Address', 'Label', 'User Instruction', and 'Loaded Instructions'. The 'User Instruction' tab is active. To the right, there's a section for 'INT Registers' and 'FP Registers', followed by memory and statistics. A large red text overlay 'Control de la ejecución' is centered over the workspace.

Control de la ejecución

CREATOR 3.2 RISC-V (RV32IMFD)
didaCtic and geneRic assEmbly progrAmming simulaTOR

ARCOS uc3m Universidad Carlos III de Madrid
Computer Science and Engineering Department

Architecture # Assembly Reset Inst. Run Stop Examples Calculator Configuration Info

Break Address Label User Instruction Loaded Instructions

INT Registers FP Registers Memory Stats Energy (CLK Cyles)

Register value representation Register name representation

Signed	Unsig.	IEEE 754	Hex.
zero x0 0000000	ra x1 FFFFFFFF	sp x2 0FFFFFFC	gp x3 0000000
tp x4 0000000	t0 x5 0000000	t1 x6 0000000	t2 x7 0000000
fp s0 x8 0000000	s1 x9 0000000	a0 x10 0000000	a1 x11 0000000
a2 x12 0000000	a3 x13 0000000	a4 x14 0000000	a5 x15 0000000
a6 x16 0000000	a7 x17 0000000	s2 x18 0000000	s3 x19 0000000
s4 x20 0000000	s5 x21 0000000	s6 x22 0000000	s7 x23 0000000
s8 x24 0000000	s9 x25 0000000	s10 x26 0000000	s11 x27 0000000
t3 x28 0000000	t4 x29 0000000	t5 x30 0000000	t6 x31 0000000

Clear Enter



Ejemplos de programas en ensamblador

Examples

CREATOR 3.2 RISC-V
didactic and generic assembly programs

Examples set available:

Architecture Break Address

default uc3m-ec-ag

Example 1: Data Storage

Example 2: ALU operations

Example 3: Store/Load Data in Memory

Example 4: FPU operations

Example 5: Loop

Example 6: Branch

Example 7: Loop + Memory

Example 8: Copy of matrices

Example 9: I/O Syscalls

Example 10: I/O Syscalls + Strings

Example 11: Subroutines

Example 12: Factorial

ejemplos

ARCOS uc3m Universidad Carlos III de Madrid Computer Science and Engineering Department

INT Registers FP Registers Memory Stats Energy (CLK Cycles)

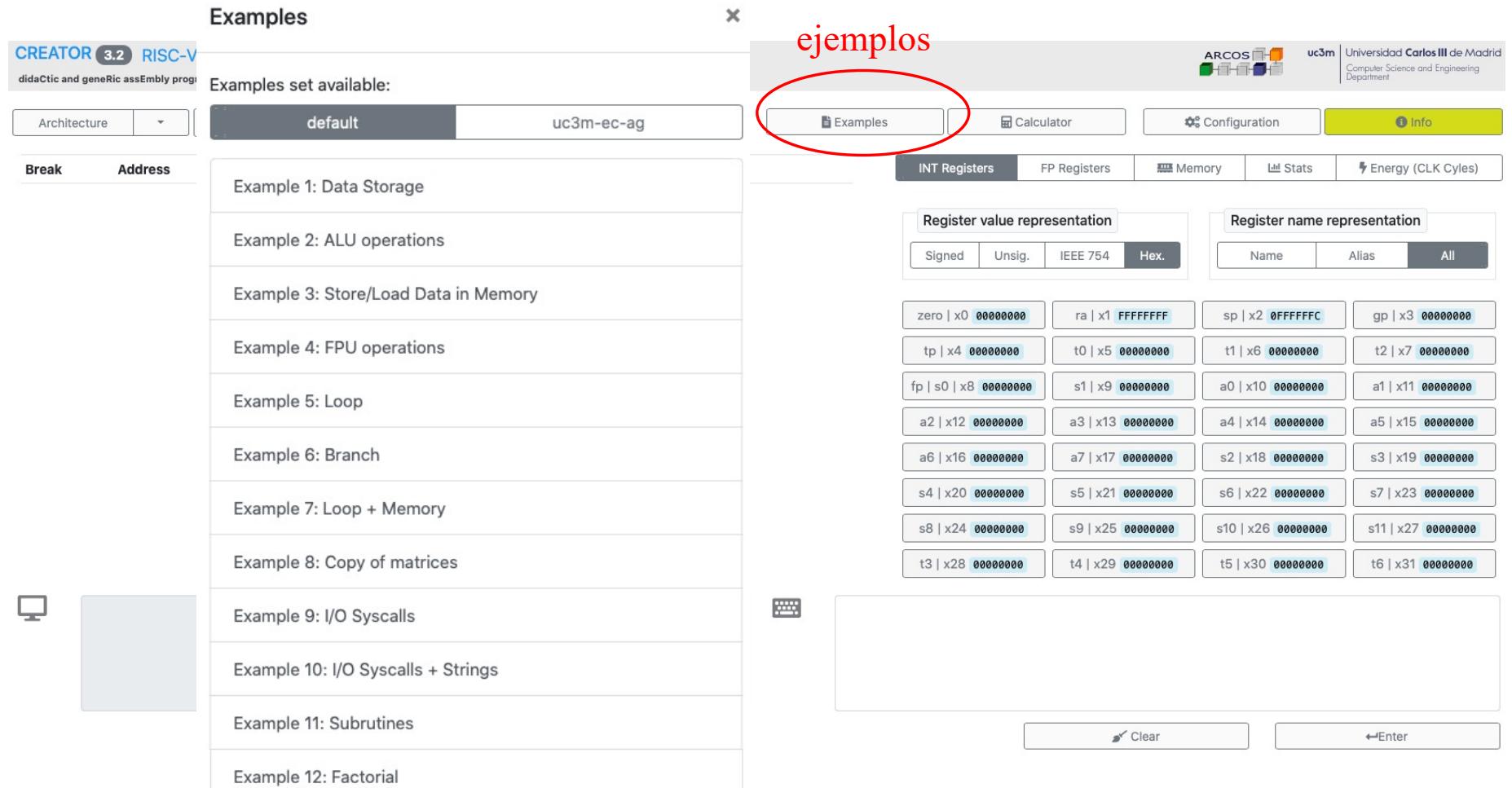
Register value representation Register name representation

Signed Unsigned IEEE 754 Hex.

Name Alias All

zero x0 0000000	ra x1 FFFFFFFF	sp x2 0FFFFFC	gp x3 0000000
tp x4 0000000	t0 x5 0000000	t1 x6 0000000	t2 x7 0000000
fp s0 x8 0000000	s1 x9 0000000	a0 x10 0000000	a1 x11 0000000
a2 x12 0000000	a3 x13 0000000	a4 x14 0000000	a5 x15 0000000
a6 x16 0000000	a7 x17 0000000	s2 x18 0000000	s3 x19 0000000
s4 x20 0000000	s5 x21 0000000	s6 x22 0000000	s7 x23 0000000
s8 x24 0000000	s9 x25 0000000	s10 x26 0000000	s11 x27 0000000
t3 x28 0000000	t4 x29 0000000	t5 x30 0000000	t6 x31 0000000

Clear Enter



Calculadora de números en coma flotante

CREATOR 3.2 RISC-V (RV32IMFD)
didaCtic and geneRic assEmby progrAmming simulaTOR

ARCOS uc3m Universidad Carlos III de Madrid
Computer Science and Engineering Department

Floating Point Calculator

32 Bits 64 Bits

41840000

0 10000011 00001000000000000000000000000000

↓ ↓ ↓

-1⁰ * 2¹³¹⁻¹²⁷ * 0.03125 = 16.5

Convert

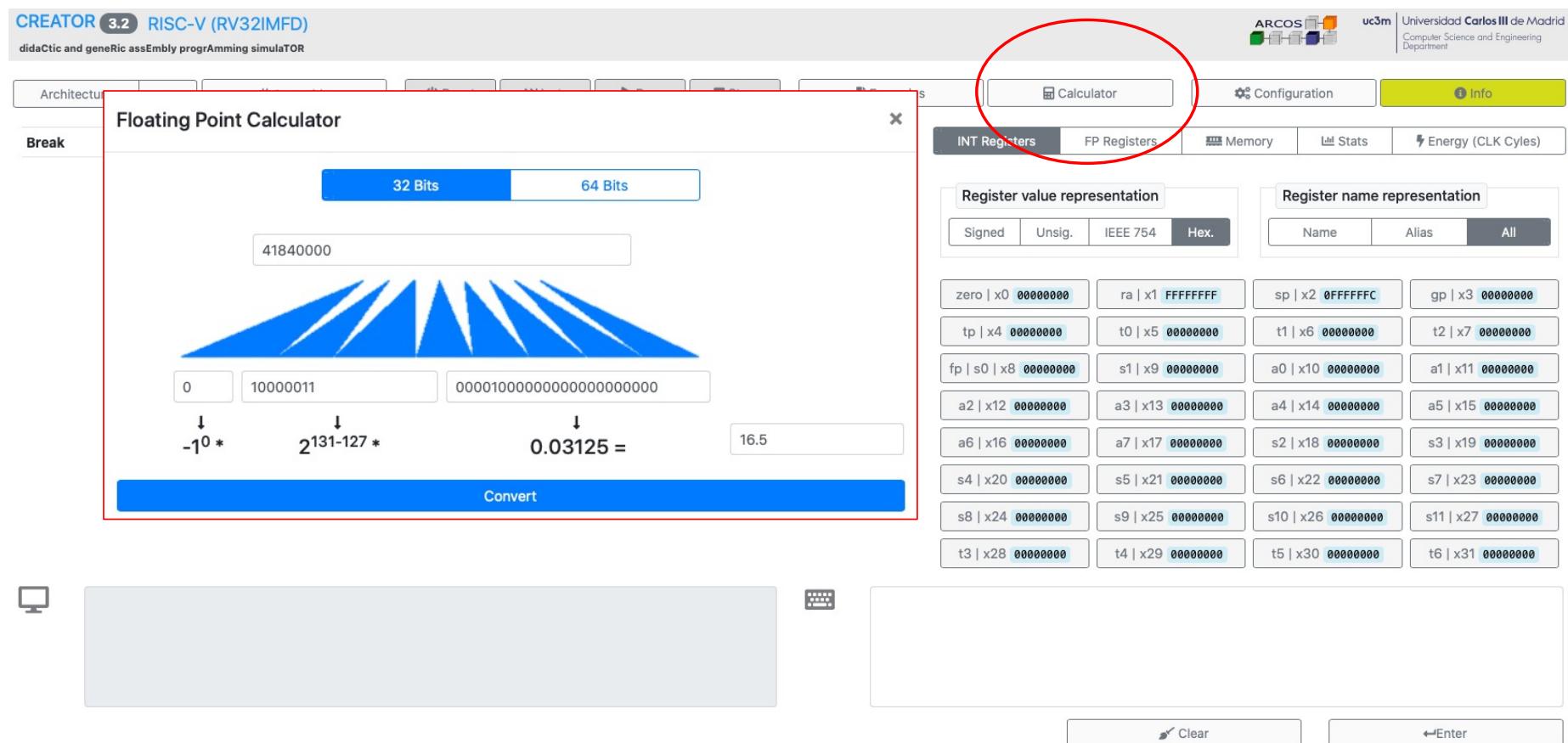
Calculator (highlighted with a red circle)

INT Registers FP Registers Memory Stats Energy (CLK Cycles)

Register value representation Register name representation

Signed	Unsig.	IEEE 754	Hex.
zero x0 00000000	ra x1 FFFFFFFF	sp x2 0FFFFFFC	gp x3 00000000
tp x4 00000000	t0 x5 00000000	t1 x6 00000000	t2 x7 00000000
fp s0 x8 00000000	s1 x9 00000000	a0 x10 00000000	a1 x11 00000000
a2 x12 00000000	a3 x13 00000000	a4 x14 00000000	a5 x15 00000000
a6 x16 00000000	a7 x17 00000000	s2 x18 00000000	s3 x19 00000000
s4 x20 00000000	s5 x21 00000000	s6 x22 00000000	s7 x23 00000000
s8 x24 00000000	s9 x25 00000000	s10 x26 00000000	s11 x27 00000000
t3 x28 00000000	t4 x29 00000000	t5 x30 00000000	t6 x31 00000000

Clear Enter



Configuración

The screenshot shows the configuration interface for the RISC-V (RV32IMFD) simulator. The top navigation bar includes tabs for 'Calculator', 'Configuration' (which is highlighted with a red circle), and 'Info'. The main area features a 'Configuration' dialog box with various settings:

- Execution Speed: A slider set to a medium value.
- Maximum stack values listed: A slider set to a medium value.
- Execution Autoscroll: A toggle switch turned on.
- Notification Time: A slider set to a medium value.
- Instruction Help Size: A slider set to a medium value.
- Dark Mode: A toggle switch turned off.
- Debug: A toggle switch turned off.

Below the configuration dialog, there are sections for register representation and memory dump representation, each with IEEE 754 and Hex options. The memory dump representation section shows a grid of registers with their addresses and values:

sig.	Name	Alias	All
00	ra x1 FFFFFFFF	sp x2 0FFFFFFC	gp x3 00000000
0	t0 x5 00000000	t1 x6 00000000	t2 x7 00000000
000	s1 x9 00000000	a0 x10 00000000	a1 x11 00000000
00	a3 x13 00000000	a4 x14 00000000	a5 x15 00000000
00	a7 x17 00000000	s2 x18 00000000	s3 x19 00000000
00	s5 x21 00000000	s6 x22 00000000	s7 x23 00000000
00	s9 x25 00000000	s10 x26 00000000	s11 x27 00000000
00	t4 x29 00000000	t5 x30 00000000	t6 x31 00000000

At the bottom are 'Clear' and 'Enter' buttons.



Configuración

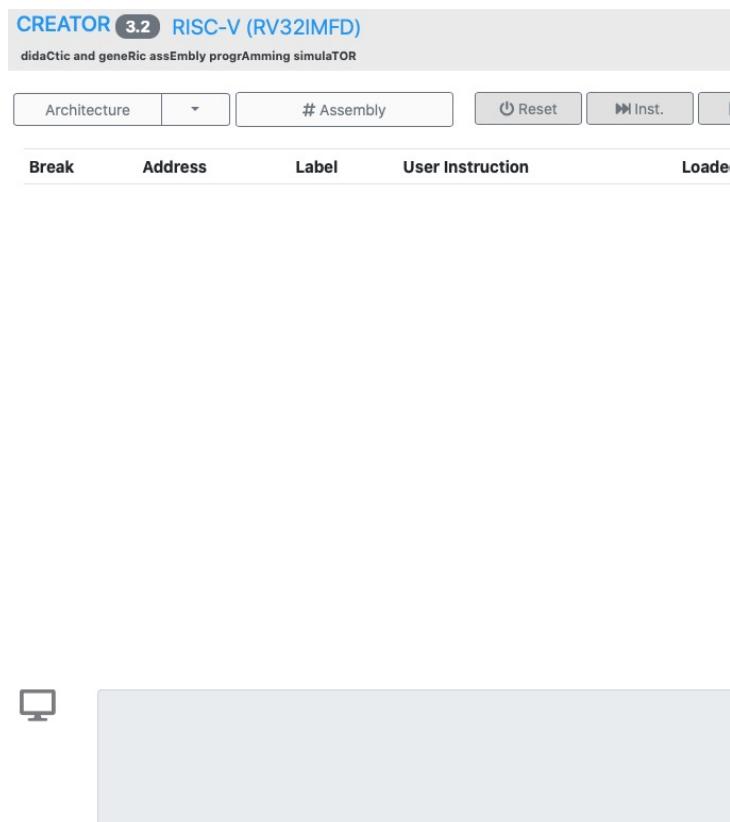
The screenshot shows the 'Info' tab of the RISC-V (RV32IMFD) configuration interface. The interface includes tabs for Architecture, Assembly, Reset, Run, Stop, Examples, Calculator, Configuration, and Info. The Configuration tab is highlighted with a red oval. Below the tabs, there are sections for Break, Address, Label, User Instruction, and Loaded Instructions. A large table displays register values in various formats (Signed, Unsigned, IEEE 754, Hex). The table has four columns: Register name representation (Name, Alias, All) and Register value representation (zero, ra, sp, gp, tp, t0, t1, t2, fp, s0, s1, a0, a1, a2, a3, a4, a5, a6, a7, s2, s3, s4, s5, s6, s7, s8, s9, s10, t3, t4, t5, t6). The first column shows the register name followed by its value in hex. The second column shows the register name followed by its value in hex. The third column shows the register name followed by its value in hex. The fourth column shows the register name followed by its value in hex.

Register name representation	Register value representation
Name	zero x0 0000000
Alias	ra x1 FFFFFFFF
All	sp x2 0FFFFFC
Name	gp x3 0000000
Alias	tp x4 0000000
All	t0 x5 0000000
Name	t1 x6 0000000
Alias	t2 x7 0000000
All	fp s0 x8 0000000
Name	s1 x9 0000000
Alias	a0 x10 0000000
All	a1 x11 0000000
Name	a2 x12 0000000
Alias	a3 x13 0000000
All	a4 x14 0000000
Name	a5 x15 0000000
Alias	a6 x16 0000000
All	a7 x17 0000000
Name	s2 x18 0000000
Alias	s3 x19 0000000
All	s4 x20 0000000
Name	s5 x21 0000000
Alias	s6 x22 0000000
All	s7 x23 0000000
Name	s8 x24 0000000
Alias	s9 x25 0000000
All	s10 x26 0000000
Name	t3 x28 0000000
Alias	t4 x29 0000000
All	t5 x30 0000000
Name	t6 x31 0000000

Below the table are icons for monitor, keyboard, and a large empty area. At the bottom are 'Clear' and 'Enter' buttons.



Configuración



The 'Instruction Help' window has a search bar at the top labeled 'Search instruction' and a section titled 'RISC-V (RV32IMFD) Guide'. It lists several RISC-V instructions with their mnemonics and formats:

- lui
- lui rd imm
- auipc
- auipc rd imm
- jal
- jal rd imm
- jalr
- jalr rd imm (rs1)
- beq
- beq rs1 rs2 imm
- bne
- bne rs1 rs2 imm
- blt
- blt rs1 rs2 imm
- bge
- bge rs1 rs2 imm
- bltu
- bltu rs1 rs2 imm
- bgeu
- bgeu rs1 rs2 imm

The ARCOS interface includes tabs for 'Configuration' and 'Info' (highlighted with a red circle). Below the tabs are buttons for 'Memory', 'Stats', and 'Energy (CLK Cycles)'. On the right, there is a 'Register name representation' section with a table of register values:

54	Hex.	Name	Alias	All
x1	FFFFFFFFFF	sp x2	0FFFFFFC	gp x3 00000000
x5	00000000	t1 x6	00000000	t2 x7 00000000
x9	00000000	a0 x10	00000000	a1 x11 00000000
x13	00000000	a4 x14	00000000	a5 x15 00000000
x17	00000000	s2 x18	00000000	s3 x19 00000000
x21	00000000	s6 x22	00000000	s7 x23 00000000
x25	00000000	s10 x26	00000000	s11 x27 00000000
x29	00000000	t5 x30	00000000	t6 x31 00000000

At the bottom are 'Clear' and 'Enter' buttons.



Configuración

Instruction Help	
<input type="text"/> Search instruction	
 RISC-V (RV32IMFD) Guide	
lui	<i>lui rd imm</i>
auipc	<i>auipc rd imm</i>
jal	<i>jal rd imm</i>
jalr	<i>jalr rd imm (rs1)</i>
beq	<i>beq rs1 rs2 imm</i>
bne	<i>bne rs1 rs2 imm</i>
blt	<i>blt rs1 rs2 imm</i>
bge	<i>bge rs1 rs2 imm</i>
bltu	<i>bltu rs1 rs2 imm</i>
bgeu	

ARCOS uc3m Universidad Carlos III de Madrid
Computer Science and Engineering Department

Configuration Info (Red Circle)

Memory Stats Energy (CLK Cycles)

Register name representation

Name	Alias	All
x1 FFFFFFFF	sp x2 0FFFFFFC	gp x3 00000000
x5 00000000	t1 x6 00000000	t2 x7 00000000
x9 00000000	a0 x10 00000000	a1 x11 00000000
x13 00000000	a4 x14 00000000	a5 x15 00000000
x17 00000000	s2 x18 00000000	s3 x19 00000000
x21 00000000	s6 x22 00000000	s7 x23 00000000
x25 00000000	s10 x26 00000000	s11 x27 00000000
x29 00000000	t5 x30 00000000	t6 x31 00000000

Clear Enter



Registros enteros

The screenshot shows the 'INT Registers' tab of the RISC-V (RV32IMFD) simulation interface. A red circle highlights the 'INT Registers' button in the top navigation bar. Below it, two sections are visible: 'Register value representation' and 'Register name representation'. The 'Register value representation' section contains 32 entries, each showing a register name followed by its hex value. The 'Register name representation' section contains three columns: 'Name', 'Alias', and 'All'. At the bottom, there are 'Clear' and 'Enter' buttons.

Register	Value (Hex)
zero x0	00000000
ra x1	FFFFFFFFFF
sp x2	0FFFFFFC
gp x3	00000000
tp x4	00000000
t0 x5	00000010
t1 x6	00000000
t2 x7	00000000
fp s0 x8	00000000
s1 x9	00000000
a0 x10	00000000
a1 x11	00000000
a2 x12	00000000
a3 x13	00000000
a4 x14	00000000
a5 x15	00000000
a6 x16	00000000
a7 x17	00000000
s2 x18	00000000
s3 x19	00000000
s4 x20	00000000
s5 x21	00000000
s6 x22	00000000
s7 x23	00000000
s8 x24	00000000
s9 x25	00000000
s10 x26	00000000
s11 x27	00000000
t3 x28	00000000
t4 x29	00000000
t5 x30	00000000
t6 x31	00000000



Registros enteros

The screenshot shows the ARCOS RISC-V (RV32IMFD) simulation interface. At the top, there are tabs for Architecture, # Assembly, Reset, Run, Stop, Examples, Calculator, Configuration, and Info. The Configuration tab is highlighted.

In the center, there's a table for Loaded Instructions with columns for Break, Address, Label, User Instruction, and Loaded Instructions. Below this is a navigation bar with tabs for INT Registers, FP Registers, Memory, Stats, and Energy (CLK Cycles). The INT Registers tab is selected.

A red circle highlights the "Register value representation" section, which includes buttons for Signed, Unsigned, IEEE 754, and Hex. The Hex button is selected. To the right is the "Register name representation" section with tabs for Name, Alias, and All. The All tab is selected.

The main area displays a grid of 32-bit register values in hex format:

zero x0 00000000	ra x1 FFFFFFFF	sp x2 0FFFFFFC	gp x3 00000000
tp x4 00000000	t0 x5 00000010	t1 x6 00000000	t2 x7 00000000
fp s0 x8 00000000	s1 x9 00000000	a0 x10 00000000	a1 x11 00000000
a2 x12 00000000	a3 x13 00000000	a4 x14 00000000	a5 x15 00000000
a6 x16 00000000	a7 x17 00000000	s2 x18 00000000	s3 x19 00000000
s4 x20 00000000	s5 x21 00000000	s6 x22 00000000	s7 x23 00000000
s8 x24 00000000	s9 x25 00000000	s10 x26 00000000	s11 x27 00000000
t3 x28 00000000	t4 x29 00000000	t5 x30 00000000	t6 x31 00000000

At the bottom, there are icons for monitor and keyboard, and buttons for Clear and Enter.



Registros enteros

The screenshot shows the CREATOR 3.2 RISC-V (RV32IMFD) simulator interface. A red box highlights the register value representation for register t0 | x5. The highlighted area contains the following table:

	t0 x5
Hex.	00000010
Binary	00000000000000000000000000000000100000
Signed	16
Unsig.	16
Char	☒
IEEE 754 (32 bits)	2.2420775429197073e-44

Below this table is an "Enter new value" input field and a "Update" button. A red arrow points from the "Update" button to the "Hex." value in the table.

The interface also includes tabs for INT Registers, FP Registers, Memory, Stats, and Energy (CLK Cycles). The INT Registers tab is selected. The Register value representation section shows various registers with their current values in hex, binary, signed, unsigned, and IEEE 754 formats. The Register name representation section shows the names of the registers.

Registros en coma flotante

The screenshot shows the CREATOR 3.2 RISC-V (RV32IMFD) simulation interface. At the top, there are tabs for Architecture, # Assembly, Reset, Inst., Run, Stop, Examples, Calculator, Configuration, and Info. The Info tab is highlighted in yellow. Below the tabs, there are columns for Break, Address, Label, User Instruction, and Loaded Instructions. A red circle highlights the FP Registers tab in the navigation bar, which is also highlighted in yellow. To the right of the navigation bar are buttons for INT Registers, FP Registers, Memory, Stats, and Energy (CLK Cycles). Below the navigation bar, there are two sections: Register value representation and Register name representation. The Register value representation section has tabs for Signed, Unsigned, IEEE 754, and Hex, with Hex selected. The Register name representation section has tabs for Name, Alias, and All, with All selected. The main area displays a grid of floating-point registers (ft0-f11, fs0-f23, fa0-f31) with their corresponding names and hex values.

Register	Value (Hex)
ft0 f0	0000000000000000
ft1 f1	0000000000000000
ft2 f2	0000000000000000
ft3 f3	0000000000000000
ft4 f4	0000000000000000
ft5 f5	0000000000000000
ft6 f6	0000000000000000
ft7 f7	0000000000000000
fs0 f8	0000000000000000
fs1 f9	0000000000000000
fa0 f10	0000000000000000
fa1 f11	0000000000000000
fa2 f12	0000000000000000
fa3 f13	0000000000000000
fa4 f14	0000000000000000
fa5 f15	0000000000000000
fa6 f16	0000000000000000
fa7 f17	0000000000000000
fs2 f18	0000000000000000
fs3 f19	0000000000000000
fs4 f20	0000000000000000
fs5 f21	0000000000000000
fs6 f22	0000000000000000
fs7 f23	0000000000000000
fs8 f24	0000000000000000
fs9 f25	0000000000000000
fs10 f26	0000000000000000
fs11 f27	0000000000000000
ft8 f28	0000000000000000
ft9 f29	0000000000000000
ft10 f30	0000000000000000
ft11 f31	0000000000000000



Registros en coma flotante

CREATOR 3.2 RISC-V (RV32IMFD)
didaCtic and geneRic assEmby progrAMming simulaTOR

ARCOS | uc3m | Universidad Carlos III de Madrid
Computer Science and Engineering Department

Architecture # Assembly Reset Inst. Run Stop Examples Calculator Configuration Info

Break Address Label User Instruction Loaded Instructions INT Registers FP Registers Memory Stats Energy (CLK Cyles)

ft5 | f5

Hex.	00000000C1840000
Binary	00000000000000000000000000000000110000110000100000000000000000
IEEE 754 (32 bits)	-16.5
IEEE 754 (64 bits)	1.6040599287e-314

Enter new value Simple Precision Update

ft5 | f5 00000000C1840000

fs5 | f21

Hex.	C030800000000000
Binary	1100000000110000100
IEEE 754 (32 bits)	0
IEEE 754 (64 bits)	-16.5

Enter new value Double Precision Update

Register value representation Register name representation

Signed Unsigned IEEE 754 Hex. Name Alias All

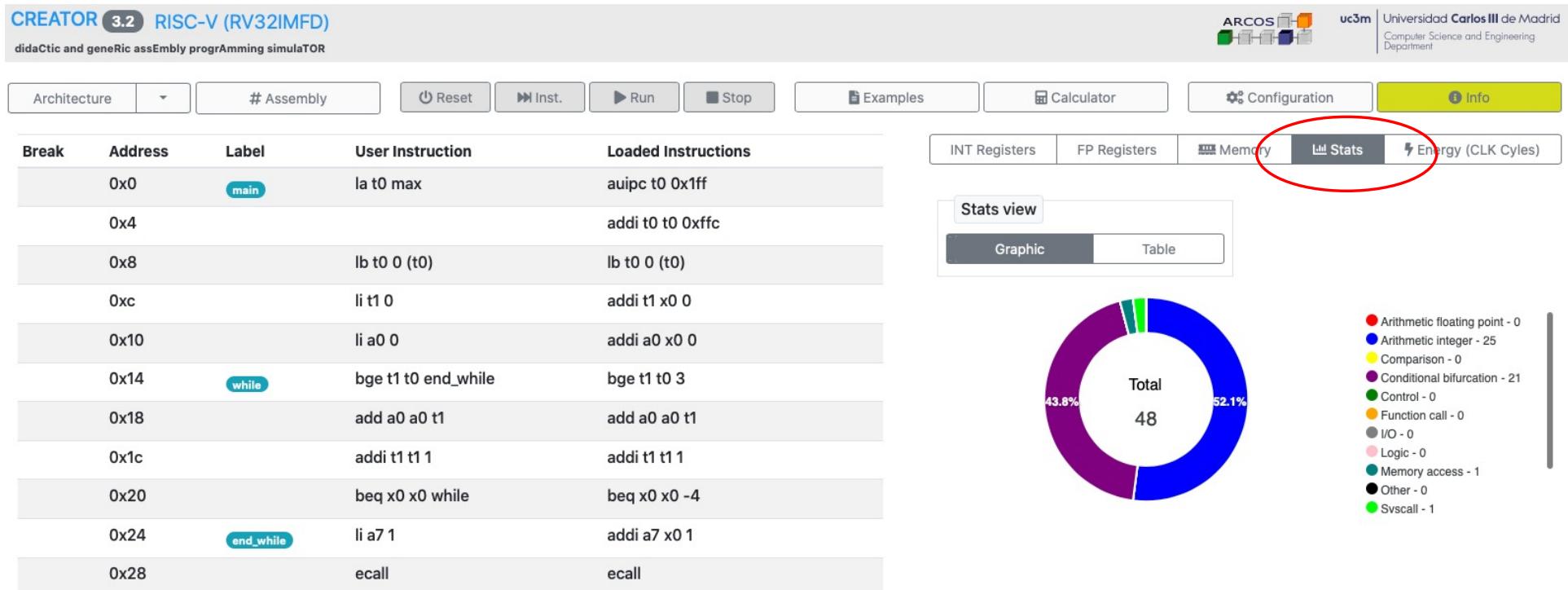
ft0 f0 0000000000000000	ft1 f1 0000000000000000	ft2 f2 0000000000000000	ft3 f3 0000000000000000
ft4 f4 0000000000000000	ft5 f5 00000000C1840000	ft6 f6 0000000000000000	ft7 f7 0000000000000000
fs0 f8 0000000000000000	fs1 f9 0000000000000000	fa0 f10 0000000000000000	fa1 f11 0000000000000000
fa2 f12 0000000000000000	fa3 f13 0000000000000000	fa4 f14 0000000000000000	fa5 f15 0000000000000000
fa6 f16 0000000000000000	fa7 f17 0000000000000000	fs2 f18 0000000000000000	fs3 f19 0000000000000000
fs4 f20 0000000000000000	fs5 f21 C030800000000000	fs6 f22 0000000000000000	fs7 f23 0000000000000000
fs8 f24 0000000000000000	fs9 f25 0000000000000000	fs10 f26 0000000000000000	fs11 f27 0000000000000000
ft8 f28 0000000000000000	ft9 f29 0000000000000000	ft10 f30 0000000000000000	ft11 f31 0000000000000000

Contenido de la memoria

The screenshot shows the 'MEMORY' tab of the RISC-V (RV32IMFD) simulation interface. The interface includes a header with 'CREATOR 3.2 RISC-V (RV32IMFD)', 'didaCtic and geneRic assEmbly progrAmming simulaTOR', and logos for 'ARCOS' and 'uc3m Universidad Carlos III de Madrid'. Below the header are buttons for 'Architecture', '# Assembly', 'Reset', 'Inst.', 'Run', 'Stop', 'Examples', 'Calculator', 'Configuration', and 'Info'. A navigation bar below the buttons includes 'Break', 'Address', 'Label', 'User Instruction', and 'Loaded instructions'. The 'MEMORY' tab is highlighted with a red circle. Below the tabs are sections for 'INT Registers', 'FP Registers', 'Memory', 'Stats', and 'Energy (CLK Cycles)'. The 'Memory' section is titled 'Main memory segment' and contains tabs for 'Data', 'Text', and 'Stack'. A table below shows columns for 'Address', 'Binary', and 'Value'. At the bottom, there are icons for a monitor, keyboard, and two empty input fields labeled 'Clear' and 'Enter'.



Estadísticas de ejecución



Ciclos ejecutados

CREATOR 3.2 RISC-V (RV32IMFD)
didaCtic and geneRic assEmbyl progrAmming simulaTOR

ARCOS uc3m Universidad Carlos III de Madrid
Computer Science and Engineering Department

Architecture	# Assembly	Reset	Inst.	Run	Stop	Examples	Calculator	Configuration	Info
Break	Address	Label	User Instruction	Loaded Instructions					
0x0		main	la t0 max	auipc t0 0x1ff					
0x4				addi t0 t0 0xfffc					
0x8			lb t0 0 (t0)	lb t0 0 (t0)					
0xc			li t1 0	addi t1 x0 0					
0x10			li a0 0	addi a0 x0 0					
0x14		while	bge t1 t0 end_while	bge t1 t0 3					
0x18			add a0 a0 t1	add a0 a0 t1					
0x1c			addi t1 t1 1	addi t1 t1 1					
0x20			beq x0 x0 while	beq x0 x0 -4					
0x24		end_while	li a7 1	addi a7 x0 1					
0x28			ecall	ecall					

INT Registers FP Registers Memory Stats Energy (CLK Cycles)

CLK Cycles view
Graphic Table Total CLK Cycles: 48

Instruction Type	CLK Cycles
Arithmetic floating point	0
Arithmetic integer	25
Comparison	0
Conditional bifurcation	21
Control	0
Function call	0
I/O	0
Logic	0
Memory access	0
Other	0
Syscall	0
Transfer between registers	0
Unconditional bifurcation	0



Pantalla

The screenshot shows the 'CREATOR 3.2 RISC-V (RV32IMFD)' interface. At the top, there's a navigation bar with tabs for Architecture, # Assembly, Reset, Run, Stop, Examples, Calculator, Configuration, and Info. The Info tab is highlighted in yellow. Below the navigation bar, there are columns for Break, Address, Label, User Instruction, and Loaded Instructions. To the right, there are tabs for INT Registers, FP Registers, Memory (which is selected), Stats, and Energy (CLK Cycles). A large central area is labeled 'Main memory segment' and contains tabs for Data, Text, and Stack. Below this is a table with columns for Address, Binary, and Value. At the bottom, there are 'Clear' and 'Enter' buttons.



Teclado

The screenshot shows the 'CREATOR 3.2 RISC-V (RV32IMFD)' simulation interface. At the top, there's a navigation bar with tabs for Architecture, # Assembly, Reset, Run, Stop, Examples, Calculator, Configuration, and Info. Below the navigation bar, there are columns for Break, Address, Label, User Instruction, and Loaded Instructions. To the right, there are tabs for INT Registers, FP Registers, Memory (which is selected), Stats, and Energy (CLK Cycles). A main memory segment is shown with Data, Text, and Stack sections. Below the memory, there's a table with columns for Address, Binary, and Value. At the bottom left, there's a monitor icon and a large input field with a keyboard icon. A red oval highlights this input field. At the bottom right, there are buttons for Clear and Enter.

CREATOR 3.2 RISC-V (RV32IMFD)

didaCtic and geneRic assEmbly progrAmming simulaTOR

ARCOS uc3m Universidad Carlos III de Madrid
Computer Science and Engineering Department

Architecture # Assembly Reset Run Stop Examples Calculator Configuration Info

Break Address Label User Instruction Loaded Instructions

INT Registers FP Registers Memory Stats Energy (CLK Cycles)

Main memory segment

Data Text Stack

Address	Binary	Value
---------	--------	-------

Clear Enter

ARCOS

33

Edición de programas en ensamblador

CREATOR 3.2 RISC-V (RV32IMFD)

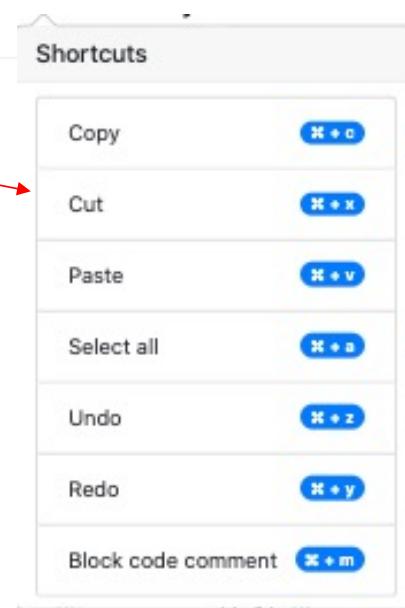
didaCtic and geneRic assEmbly progrAMming simulaTOR

Architecture ▾ Simulator ➔ Compile/Linked

Assembly:

```
1 #
2 #
3 # Creator (https://creatorsim.github.io/creator/)
4 #
5 .text
6 main:
7
8     li t0 10
9     li t2 -20
10
11    add    t3,t0, t2
12    mul    t4 t0, t2
13    div    t5, t0, t2
14
15
```

ejemplo



Compilación

The screenshot shows the CREATOR 3.2 RISC-V (RV32IMFD) interface. The title bar reads "CREATOR 3.2 RISC-V (RV32IMFD)" and "didaCtic and geneRic assEmbly progrAMming simulaTOR". The menu bar includes "Architecture", "Simulator", "Compile/Linked", and "File". A green notification box in the top right corner says "Compilation completed successfully". The main area displays assembly code:

```
1
2 #
3 # Creator (https://creatorsim.github.io/creator/)
4 #
5
6 .text
7 main:
8
9     li t0 10
10    li t2 -20
11
12   add    t3,t0, t2
13   mul    t4 t0, t2
14   div    t5, t0, t2
15
```

Error de compilación

CREATOR 3.2 RISC-V (RV32IMFD)
didactic and generic assembly programming simulator

Architecture Simulator

Assembly:

```
1
2 #
3 # Creator (https://creatorsim.github.io/creator/)
4 #
5
6 .text
7 main:
8
9   li t0 10
10  li t2 -20
11
12  add   t3,t0, t2
13  mul   t4 t0, t2
14  div   ti, t0, t2
15
```

Assembly Code Error

Code fragment:

```
...
13      mul t4 t0, t2
*     14      div ti, t0, t2
15
...
```

Error description:

Register 'ti' not found



Paso al simulador

CREATOR 3.2 RISC-V (RV32IMFD)
didactic and generic assembly programming simulator

Architecture ▾ Simulator ▾ Compile/Linked File ▾

❶ Assembly:

```
1
2 #
3 # Creator (https://creatorsim.github.io/creator/)
4 #
5
6 .text
7 main:
8
9     li t0 10
10    li t2 -20
11
12    add    t3,t0, t2
13    mul    t4 t0, t2
14    div    t5, t0, t2
15
```



Simulador

ejemplo

CREATOR 3.2 RISC-V (RV32IMFD)
didaCtic and generic assembly programming simulator

ARCOS uc3m Universidad Carlos III de Madrid
Computer Science and Engineering Department

Break	Address	Label	User Instruction
0x0		main	li t0 10
0x4			li t2 -20
0x8			
0xc			
0x10			add t3 t0 t2
0x14			mul t4 t0 t2
0x18			div t5 t0 t2

Assembly # Reset ▶ Inst. ▶ Run Stop Examples Calculator Configuration Info

User Instruction Loaded Instructions Next

Programa escrito (inst. y pseudoinst.)

Programa en memoria (instrucciones máquinas)

INT Registers		FP Registers		Memory		Stats		Energy (CLK Cycles)	
Register value representation		Register name representation							
Signed	Unsig.	IEEE 754	Hex.	Name	Alias	All			
zero x0 00000000	ra x1 FFFFFFFF	sp x2 0FFFFFFC	gp x3 00000000						
tp x4 00000000	t0 x5 00000000	t1 x6 00000000	t2 x7 00000000						
fp s0 x8 00000000	s1 x9 00000000	a0 x10 00000000	a1 x11 00000000						
a2 x12 00000000	a3 x13 00000000	a4 x14 00000000	a5 x15 00000000						
a6 x16 00000000	a7 x17 00000000	s2 x18 00000000	s3 x19 00000000						
s4 x20 00000000	s5 x21 00000000	s6 x22 00000000	s7 x23 00000000						
s8 x24 00000000	s9 x25 00000000	s10 x26 00000000	s11 x27 00000000						
t3 x28 00000000	t4 x29 00000000	t5 x30 00000000	t6 x31 00000000						

Monitor Keyboard

Clear Enter



Flujo de ejecución

CREATOR 3.2 RISC-V (RV32IMFD)
didactic and generic assembly programming simulator

Architecture	# Assembly	Reset	Inst.	Run	Stop	Examples
Break	Address	Label	User Instruction	Loaded Instructions		
0x0		main	li t0 10	addi t0 x0 10		
0x4			li t2 -20	lui t2 0		
0x8				lui t2 0xFFFFF	Current	
0xc				addi t2 t2 0xfc		Next
0x10			add t3 t0 t2	add t3 t0 t2		
0x14			mul t4 t0 t2	mul t4 t0 t2		
0x18			div t5 t0 t2	div t5 t0 t2		



Puntos de ruptura

CREATOR 3.2 RISC-V (RV32IMFD)
didactic and generic assembly programming simulator

Architecture	# Assembly	Reset	Inst.	Run	Stop	Examples
Break	Address	Label	User Instruction	Loaded Instructions		
	0x0	main	li t0 10	addi t0 x0 10		
	0x4		li t2 -20	lui t2 0		
	0x8			lui t2 0xFFFFF		
	0xc			addi t2 t2 0xfc		
	0x10		add t3 t0 t2	add t3 t0 t2	Current	
STOP	0x14		mul t4 t0 t2	mul t4 t0 t2		Next
	0x18		div t5 t0 t2	div t5 t0 t2		

Segmento de datos

ejemplo



Assembly:

```
1 .data
2
3     cadena: .string "Hola mundo"
4     A:      .byte 1
5     .align 2
6     N:      .word 64
7     C:      .byte 'a'
8     .align 2
9     F:      .float -12.5
10    D:      .double -12.5
11
12    # int v1[5]={1,2,3,4,5}
13    v1:      .word 1, 2, 3, 4, 5
14
15    #int v2[10]
16    v2:      .zero 40
```



Visualización de datos en memoria

ejemplo

1 Assembly:

```
1 .data
2
3     cadena: .string "Hola mundo"
4     A:     .byte 1
5     .align 2
6     N:     .word 64
7     C:     .byte 'a'
8     .align 2
9     F:     .float -12.5
10    D:     .double -12.5
11
12    # int v1[5]={1,2,3,4,5}
13    v1:     .word 1, 2, 3, 4, 5
14
15    #int v2[10]
16    v2:     .zero 40
```

Main memory segment			
Data	Text	Stack	
Address	Binary	Value	
0x00200000 - 0x00200003	48 6F 6C 61	H, o, l, a	①
0x00200004 - 0x00200007	20 6D 75 6E	, m, u, n	②
0x00200008 - 0x0020000B	64 6F 00 01	d, o, 0, 1	③
0x0020000C - 0x0020000F	00 00 00 40	64	④
0x00200010 - 0x00200013	61 00 00 00	97	⑤
0x00200014 - 0x00200017	C1 48 00 00	-12.5	⑥
0x00200018 - 0x0020001B	C0 28 00 00	-12.5	⑦
0x0020001C - 0x0020001F	00 00 00 00		
0x00200020 - 0x00200023	00 00 00 01	1	⑧
0x00200024 - 0x00200027	00 00 00 02	2	
0x00200028 - 0x0020002B	00 00 00 03	3	
0x0020002C - 0x0020002F	00 00 00 04	4	



Visualización de datos en memoria

1 Assembly:

```
1 .data  
2  
3     cadena: .string "Hola mundo"  
4     A:     .byte 1  
5     .align 2  
6     N:     .word 64  
7     C:     .byte 'a'  
8     .align 2  
9     F:     .float -12.5  
10    D:    .double -12.5  
11  
12    # int v1[5]={1,2,3,4,5}  
13    v1:    .word 1, 2, 3, 4, 5  
14  
15    #int v2[10]  
16    v2:    .zero 40
```

Main memory segment			
Data	Text	Stack	
Address	Binary	Value	
0x0020001C - 0x0020001F	00 00 00 00		
0x00200020 - 0x00200023	00 00 00 01	v1 1	
0x00200024 - 0x00200027	00 00 00 02	2	
0x00200028 - 0x0020002B	00 00 00 03	3	
0x0020002C - 0x0020002F	00 00 00 04	4	
0x00200030 - 0x00200033	00 00 00 05	5	
0x00200034 - 0x00200037	00 00 00 00	v2 0	
0x00200038 - 0x0020003B	00 00 00 00		
0x0020003C - 0x0020003F	00 00 00 00		
0x00200040 - 0x00200043	00 00 00 00		
0x00200044 - 0x00200047	00 00 00 00		
0x00200048 - 0x0020004B	00 00 00 00		

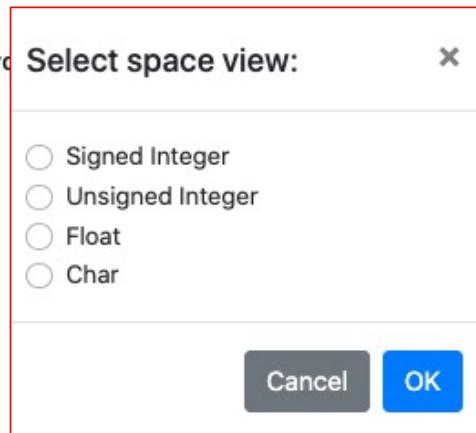


Visualización de datos en memoria

1 Assembly:

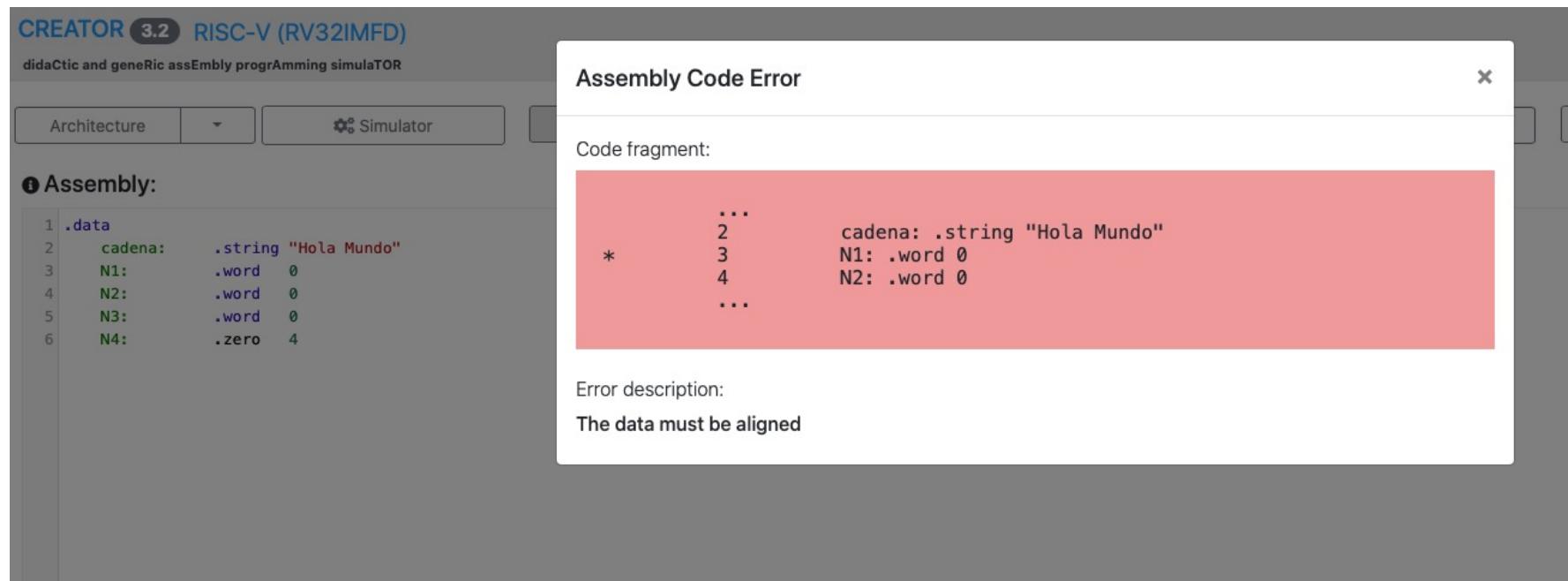
```
1 .data
2
3     cadena: .string "Hola mundo"
4     A:     .byte 1
5     .align 2
6     N:     .word 64
7     C:     .byte 'a'
8     .align 2
9     F:     .float -12.5
10    D:    .double -12.5
11
12    # int v1[5]={1,2,3,4,5}
13    v1:    .word 1, 2, 3, 4, 5
14
15    #int v2[10]
16    v2:    .zero
```

Main memory segment			
Data	Text	Stack	
Address	Binary	Value	
0x0020001C - 0x0020001F	00 00 00 00		v1
0x00200020 - 0x00200023	00 00 00 01	1	
0x00200024 - 0x00200027	00 00 00 02	2	
0x00200028 - 0x0020002B	00 00 00 03	3	
0x0020002C - 0x0020002F	00 00 00 04	4	
0x00200030 - 0x00200033	00 00 00 05	5	v2
0x00200034 - 0x00200037	00 00 00 00		
0x00200038 - 0x0020003B	00 00 00 00		
0x0020003C - 0x0020003F	00 00 00 00		
0x00200040 - 0x00200043	00 00 00 00		
0x00200044 - 0x00200047	00 00 00 00		
0x00200048 - 0x0020004B	00 00 00 00		



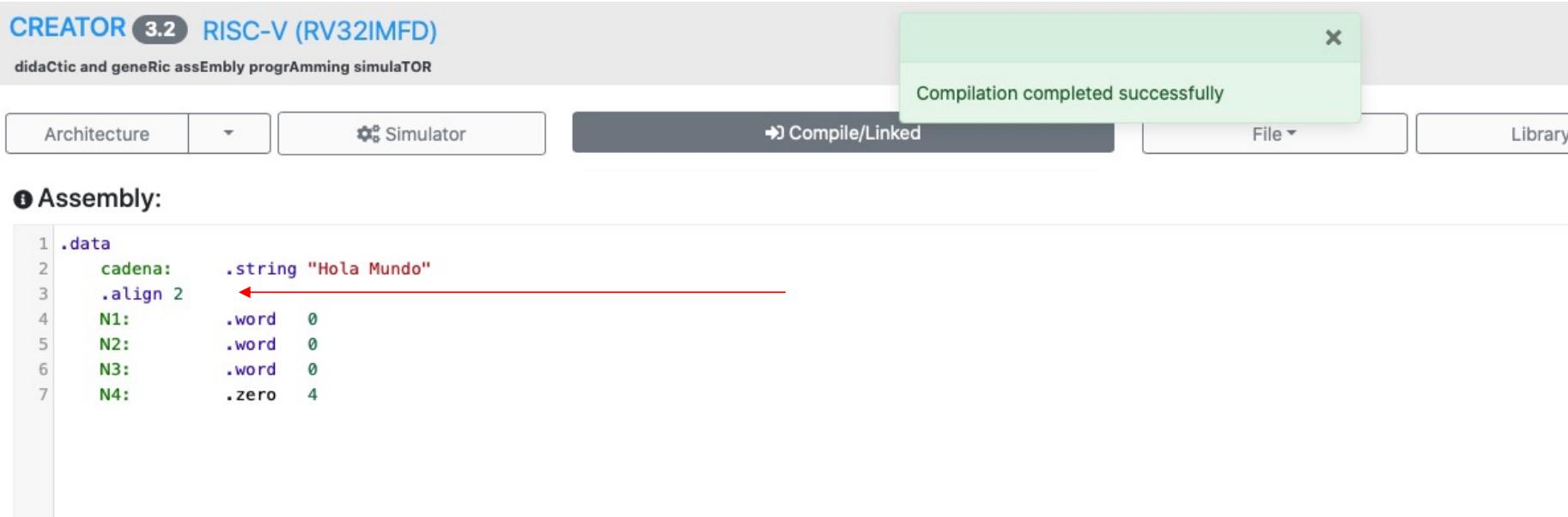
Detección de datos no alineados

ejemplo



Segmento de datos corregido

ejemplo



The screenshot shows the CREATOR 3.2 RISC-V (RV32IMFD) tool interface. The title bar reads "CREATOR 3.2 RISC-V (RV32IMFD)" and "didaCtic and geneRic assEmbly progrAmming simulaTOR". The menu bar includes "Architecture", "Simulator", "Compile/Linked" (which is highlighted in dark grey), "File", and "Library". A green notification box in the top right corner says "Compilation completed successfully". In the assembly editor, there is a comment "● Assembly:" followed by the following code:

```
1 .data
2     cadena:    .string "Hola Mundo"
3     .align 2 ←
4     N1:        .word  0
5     N2:        .word  0
6     N3:        .word  0
7     N4:        .zero   4
```

A red arrow points to the ".align 2" instruction at line 3.



Ejemplo. Ejecutar el siguiente programa

ejemplo

● Assembly:

```
1 .data
2     N1: .word  0
3     N2: .word  0
4     N3: .word  0
5     N4: .zero  4
6
7 .text
8     main:
9         li t0, 1
10        la t1, N1
11        sw t0, @t1)
12
13        li t0, 2
14        la t1, N2
15        sw t0, @t1)
16
17        li t0, -3
18        la t1, N3
19        sw t0, @t1)
20
21        li t0, 4
22        la t1, N4
23        sw t0, @t1)
```



Ejemplo: antes de la ejecución

Architecture ▾ # Assembly ⏪ Reset ⏴ Inst. ▶ Run ⏹ Stop Examples Calculator Configuration Info

Break	Address	Label	User Instruction	Loaded Instructions
0x0		main	li t0 1	addi t0 x0 1
0x4			la t1 N1	auipc t1 0x1ff
0x8				addi t1 t1 0xff8
0xc			sw t0 0 (t1)	sw t0 0 (t1)
0x10			li t0 2	addi t0 x0 2
0x14			la t1 N2	auipc t1 0x1ff
0x18				addi t1 t1 0xfc
0x1c			sw t0 0 (t1)	sw t0 0 (t1)
0x20			li t0 -3	lui t0 0
0x24				lui t0 0xffff
0x28				addi t0 t0 0ffd
0x2c			la t1 N3	auipc t1 0x1ff
0x30				addi t1 t1 0xfd8

INT Registers FP Registers Memory Stats Energy (CLK Cycles)

Main memory segment

Address	Binary	Value
0x00200000 - 0x00200003	N1 00 00 00 00	0
0x00200004 - 0x00200007	N2 00 00 00 00	0
0x00200008 - 0x0020000B	N3 00 00 00 00	0
0x0020000C - 0x0020000F	N4 00 00 00 00	0

Ejemplo: después de la ejecución

Architecture ▾ # Assembly ⏪ Reset ► Inst. ▶ Run ■ Stop Examples Calculator Configuration Info

Break	Address	Label	User Instruction	Loaded Instructions
0x0		main	li t0 1	addi t0 x0 1
0x4			la t1 N1	auipc t1 0x1ff
0x8				addi t1 t1 0xff8
0xc			sw t0 0 (t1)	sw t0 0 (t1)
0x10			li t0 2	addi t0 x0 2
0x14			la t1 N2	auipc t1 0x1ff
0x18				addi t1 t1 0xfc
0x1c			sw t0 0 (t1)	sw t0 0 (t1)
0x20			li t0 -3	lui t0 0
0x24				lui t0 0xffff
0x28				addi t0 t0 0ffd
0x2c			la t1 N3	auipc t1 0x1ff
0x30				addi t1 t1 0xfd8

INT Registers FP Registers Memory Stats Energy (CLK Cycles)

Main memory segment

Address	Binary	Value
0x00200000 - 0x00200003	N1 00 00 00 01	1
0x00200004 - 0x00200007	N2 00 00 00 02	2
0x00200008 - 0x0020000B	N3 FF FF FF FD	4294967293
0x0020000C - 0x0020000F	N4 00 00 00 04	

Ejemplo: después de la ejecución

Architecture ▾ # Assembly ⏪ Reset ► Inst. ▶ Run ■ Stop Examples Calculator Configuration Info

Break	Address	Label	User Instruction	Loaded Instructions
0x0		main	li t0 1	addi t0 x0 1
0x4			la t1 N1	auipc t1 0x1ff
0x8				addi t1 t1 0xff8
0xc			sw t0 0 (t1)	sw t0 0 (t1)
0x10			li t0 2	addi t0 x0 2
0x14			la t1 N2	auipc t1 0x1ff
0x18				addi t1 t1 0xfc
0x1c			sw t0 0 (t1)	sw t0 0 (t1)
0x20			li t0 -3	lui t0 0
0x24				lui t0 0xffff
0x28				addi t0 t0 0ffd
0x2c			la t1 N3	auipc t1 0x1ff
0x30				addi t1 t1 0xfd8

INT Registers FP Registers Memory Stats Energy (CLK Cycles)

Main memory segment

Data	Text	Stack
N1	00 00 00 01	1
N2	00 00 00 02	2
N3	FF FF FF FD	4294967293
N4	00 00 00 04	



Ejemplo: después de la ejecución

Screenshot of a debugger interface showing assembly code, registers, memory, and a floating-point status window.

Assembly View:

Break	Address	Label	User Instruction	Loaded Instructions
0x0		main	li t0 1	addi t0 x0 1
0x4			la t1 N1	auipc t1 0x1ff
0x8				addi t1 t1 0xff8
0xc			sw t0 0 (t1)	sw t0 0 (t1)
0x10			li t0 2	addi t0 x0 2
0x14			la t1 N2	auipc t1 0x1ff
0x18				addi t1 t1 0xfc
0x1c			sw t0 0 (t1)	sw t0 0 (t1)
0x20			li t0 -3	lui t0 0
0x24				lui t0 0xFFFF
0x28				addi t0 t0 0ffd
0x2c			la t1 N3	auipc t1 0x1ff
0x30				addi t1 t1 0xfd8
0x34			sw t0 0 (t1)	sw t0 0 (t1)

Registers View:

- INT Registers
- FP Registers
- Memory** (selected)
- Stats
- Energy (CLK Cycles)

Main memory segment:

Address	Binary	Value
0x00200000 - 0x00200003	N1 00 00 00 01	1
0x00200004 - 0x00200007	N2 00 00 00 02	2
0x00200008 - 0x0020000B	N3 FF FF FF FD	-3
0x0020000C - 0x0020000F	N4 00 00 00 04	4

Floating-point Status:

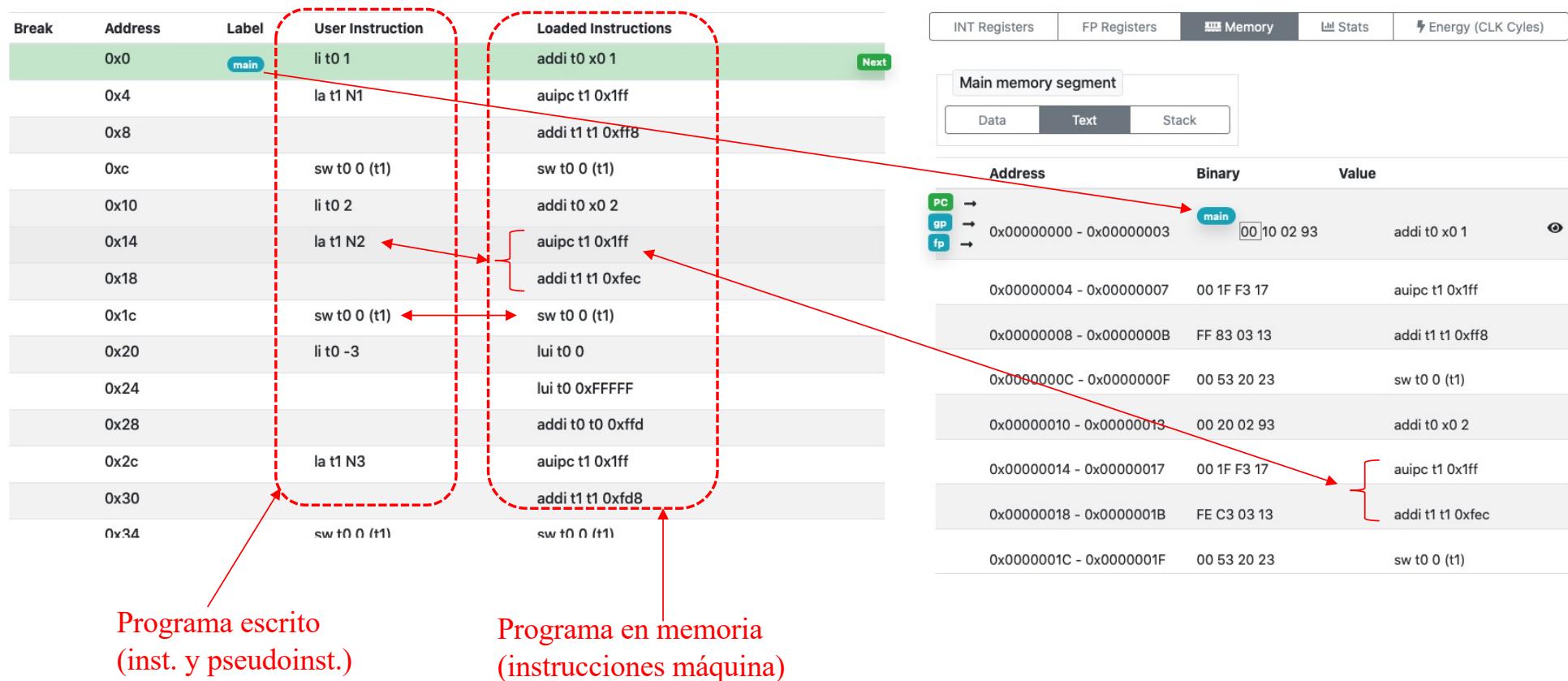
Select space view: Signed Integer Unsigned Integer Float Char

Cancel OK



Visualización del segmento de texto Instrucciones y pseudoinstrucciones

ejemplo



Visualización del segmento de texto

Flujo de ejecución

ejemplo

Break	Address	Label	User Instruction	Loaded Instructions
0x0		main	li t0 1	addi t0 x0 1
0x4			la t1 N1	auipc t1 0x1ff
0x8				addi t1 t1 0xff8
0xc			sw t0 0 (t1)	sw t0 0 (t1)
0x10			li t0 2	addi t0 x0 2
0x14			la t1 N2	auipc t1 0x1ff
0x18				addi t1 t1 0xfc
0x1c			sw t0 0 (t1)	sw t0 0 (t1)
0x20			li t0 -3	lui t0 0
0x24				lui t0 0xFFFF
0x28				addi t0 t0 0xfd
0x2c			la t1 N3	auipc t1 0x1ff
0x30				addi t1 t1 0xfd8
0x34			sw t0 0 (t1)	sw t0 0 (t1)

INT Registers	FP Registers	Memory	Stats	Energy (CLK Cycles)
Main memory segment				
Data	Text	Stack		
Address Binary Value				
0x00000008 - 0x0000000B	FF 83 03 13	addi t1 t1 0xff8		
0x0000000C - 0x0000000F	00 53 20 23	sw t0 0 (t1)		
0x00000010 - 0x00000013	00 20 02 93	addi t0 x0 2		
0x00000014 - 0x00000017	00 1F F3 17	auipc t1 0x1ff		
PC → 0x00000018 - 0x0000001B	FE C3 03 13	addi t1 t1 0xfc		
0x0000001C - 0x0000001F	00 53 20 23	sw t0 0 (t1)		
0x00000020 - 0x00000023	00 00 02 B7	lui t0 0		
0x00000024 - 0x00000027	FF FF F2 B7	lui t0 0xFFFF		
0x00000028 - 0x0000002B	FF D2 82 93	addi t0 t0 0xfd		



Visualización del segmento de texto

Varias funciones

ejemplo

CREATOR 3.2 RISC-V (RV32IMFD)
didactic and generic assembly programming simulator

Architecture ▾ Simulator ➔ Compile/Linked []

➊ Assembly:

```
1 .text
2     f1:           li  a0, 1
3                 jr  ra
4
5     f2:           li  a0, 2
6                 jr  ra
7
8     f3:           li  a0, 3
9                 jr  ra
10
11
12    main:
13        jal   ra, f1
14        jal   ra, f2
15        jal   ra, f3
```



Visualización del segmento de texto

Varias funciones

Architecture # Assembly ⏪ Reset ► Inst. ▶ Run ■ Stop Examples Calculator Configuration Info

Break	Address	Label	User Instruction	Loaded Instructions	Next
0x0		f1	li a0 1	addi a0 x0 1	
0x4			jr ra	jalr x0 0 (ra)	
0x8		f2	li a0 2	addi a0 x0 2	
0xc			jr ra	jalr x0 0 (ra)	
0x10		f3	li a0 3	addi a0 x0 3	
0x14			jr ra	jalr x0 0 (ra)	
0x18		main	jal ra f1	jal ra 0x0	Current
0x1c			jal ra f2	jal ra 0x8	
0x20			jal ra f3	jal ra 0x10	

INT Registers	FP Registers	Memory	Stats	Energy (CLK Cycles)
Main memory segment				
Data	Text	Stack		
Address	Binary	Value		
PC → 0x00000000 - 0x00000003	00 10 05 13	f1 addi a0 x0 1	①	
0x00000004 - 0x00000007	00 00 80 67	jalr x0 0 (ra)	②	
0x00000008 - 0x0000000B	00 20 05 13	f2 addi a0 x0 2	③	
0x0000000C - 0x0000000F	00 00 80 67	jalr x0 0 (ra)	④	
0x00000010 - 0x00000013	00 30 05 13	f3 addi a0 x0 3	⑤	
0x00000014 - 0x00000017	00 00 80 67	jalr x0 0 (ra)	⑥	
0x00000018 - 0x0000001B	00 00 00 EF	main jal ra 0x0	⑦	
0x0000001C - 0x0000001F	00 00 80 EF	jal ra 0x8	⑧	



Llamadas al sistema

ejemplo

CREATOR 3.2 RISC-V (RV32IMFD)

didactic and generic assembly programming simulator

Architecture

Simulator

Compile/Linked

File

Assembly:

```
4
5 .data
6     str_text: .string "Insert a number: "
7     str_result: .string "Result = "
8
9
10
11 .text
12 main:
13     # print "Insert a number: "
14     la a0 str_text
15     li a7 4
16     ecall
17
18     # read int
19     li a7 5
20     ecall
21
22     mv t0, a0
23
24     # print "Insert a number: "
25     la a0 str_text
26     li a7 4
27     ecall
28
29     # read int
30     li a7 5
31     ecall
32     mv t1, a0
33
34     # print "Result = "
35     la a0 str_result
36     li a7 4
37     ecall
```



Llamadas al sistema

Architecture # Assembly Reset Inst. Run Stop Examples Calculator Configuration Info

Break	Address	Label	User Instruction	Loaded Instructions
0x0		main	la a0 str_text	auipc a0 0x1ff
0x4				addi a0 a0 0xffc
0x8			li a7 4	addi a7 x0 4
0xc			ecall	ecall
0x10			li a7 5	addi a7 x0 5
0x14			ecall	ecall
0x18			mv t0 a0	addi t0 a0 0
0x1c			la a0 str_text	auipc a0 0x1ff
0x20				addi a0 a0 0xfe0
0x24			li a7 4	addi a7 x0 4
0x28			ecall	ecall
0x2c			li a7 5	addi a7 x0 5
0x30			ecall	ecall
0x34			mv t1 a0	addi t1 a0 0

INT Registers FP Registers Memory Stats Energy (CLK Cycles)

Register value representation Register name representation

Signed	Unsig.	IEEE 754	Hex.
Name Alias All			
zero x0 0000000	ra x1 FFFFFFFF	sp x2 0FFFFFC	gp x3 0000000
tp x4 0000000	t0 x5 0000000	t1 x6 0000000	t2 x7 0000000
fp s0 x8 0000000	s1 x9 0000000	a0 x10 0020000	a1 x11 0000000
a2 x12 0000000	a3 x13 0000000	a4 x14 0000000	a5 x15 0000000
a6 x16 0000000	a7 x17 0000005	s2 x18 0000000	s3 x19 0000000
s4 x20 0000000	s5 x21 0000000	s6 x22 0000000	s7 x23 0000000
s8 x24 0000000	s9 x25 0000000	s10 x26 0000000	s11 x27 0000000
t3 x28 0000000	t4 x29 0000000	t5 x30 0000000	t6 x31 0000000

Insert a number: Clear Enter



Ejemplo: ejecutar el siguiente programa

ejemplo

Assembly:

```
1 #
2 # Creator (https://creatorsim.github.io/creator/)
3 #
4
5 .data
6
7     string1:    .string "Hola Mundo"
8     string2:    .zero 32
9
10 .text
11     main:
12         la t0, string1
13         la t1, string2
14
15     loop: lbu   t2, 0(t0)
16         beq   t2, zero, end
17         sb    t2, 0(t1)
18         addi  t0, t0, 1
19         addi  t1, t1, 1
20         j     loop
21     end:
22         li a7, 10
23         ecall
24
```

Main memory segment		
Data	Text	Stack
0x00200000 - 0x00200003	48 6F 6C 61	H, o, l, a
0x00200004 - 0x00200007	20 4D 75 6E	, M, u, n
0x00200008 - 0x0020000B	64 6F 00 00	d, o, 0
0x0020000C - 0x0020000F	00 00 00 00	
0x00200010 - 0x00200013	00 00 00 00	
0x00200014 - 0x00200017	00 00 00 00	
0x00200018 - 0x0020001B	00 00 00 00	
0x0020001C - 0x0020001F	00 00 00 00	

Ejecución del programa

Architecture	# Assembly	Reset	Inst.	Run	Stop	Examples	Calculator	Configuration	Info
Break	Address	Label	User Instruction	Loaded Instructions					
0x0		main	la t0 string1	auipc t0 0x1ff					
0x4				addi t0 t0 0xfffc					
0x8			la t1 string2	auipc t1 0x1ff					
0xc				addi t1 t1 0xffff					
0x10		loop	lbu t2 0 (t0)	lbu t2 0 (t0)	Current				
0x14			beq t2 zero end	beq t2 zero 4	Next				
0x18			sb t2 0 (t1)	sb t2 0 (t1)					
0x1c			addi t0 t0 1	addi t0 t0 1					
0x20			addi t1 t1 1	addi t1 t1 1					
0x24			j loop	jal x0 0x10					
0x28		end	li a7 10	addi a7 x0 10					
0x2c			ecall	ecall					

INT Registers	FP Registers	Memory	Stats	Energy (CLK Cycles)
Main memory segment				
Data	Text	Stack		
Address	Binary	Value		
0x00000004 - 0x00000007	FF C2 02 93	auipc t0 0x1ff		
0x00000008 - 0x0000000B	00 1F F3 17	auipc t1 0x1ff		
0x0000000C - 0x0000000F	FF F3 03 13	addi t1 t1 0xffff		
0x00000010 - 0x00000013	loop 00 02 C3 83	lbu t2 0 (t0)	①	
PC → 0x00000014 - 0x00000017	00 03 82 63	beq t2 zero 4		
0x00000018 - 0x0000001B	00 73 00 23	sb t2 0 (t1)		
0x0000001C - 0x0000001F	00 12 82 93	addi t0 t0 1		
0x00000020 - 0x00000023	00 13 03 13	addi t1 t1 1		
0x00000024 - 0x00000027	00 01 00 6F	jal x0 0x10		
	end			



Ejecución del programa. Bucles

Architecture ▾ # Assembly ⏹ Reset ⏷ Inst. ▶ Run ⏻ Stop Examples Calculator Configuration Info

Break	Address	Label	User Instruction	Loaded Instructions
0x0		main	la t0 string1	auipc t0 0x1ff
0x4				addi t0 t0 0xffff
0x8		la t1 string2		auipc t1 0x1ff
0xc				addi t1 t1 0xffff
0x10		loop	lbu t2 0 (t0)	lbu t2 0 (t0) Next
0x14			beq t2 zero end	beq t2 zero 4
0x18			sb t2 0 (t1)	sb t2 0 (t1)
0x1c			addi t0 t0 1	addi t0 t0 1
0x20			addi t1 t1 1	addi t1 t1 1
0x24			j loop	jal x0 0x10 Current
0x28		end	li a7 10	addi a7 x0 10
0x2c			ecall	ecall

INT Registers FP Registers Memory Stats Energy (CLK Cycles)

Main memory segment

Data	Text	Stack
0x00000004 - 0x00000007 FF C2 82 83		
0x00000008 - 0x0000000B 00 1F F3 17		auipc t1 0x1ff
0x0000000C - 0x0000000F FF F3 03 13		addi t1 t1 0xffff
0x00000010 - 0x00000013 00 02 C3 83	loop	lbu t2 0 (t0) ②
0x00000014 - 0x00000017 00 03 82 63		beq t2 zero 4
0x00000018 - 0x0000001B 00 73 00 23		sb t2 0 (t1)
0x0000001C - 0x0000001F 00 12 82 93		addi t0 t0 1
0x00000020 - 0x00000023 00 13 03 13		addi t1 t1 1
0x00000024 - 0x00000027 00 01 00 6F		jal x0 0x10
	end	



Ejemplo de llamadas a funciones anidadas

ejemplo

- ▶ Comprobar el crecimiento de la pila

INT Registers	FP Registers	Memory	Stats	Energy (CLK Cycles)															
Main memory segment																			
	Data	Text	Stack																
<table><thead><tr><th>Address</th><th>Binary</th><th>Value</th></tr></thead><tbody><tr><td>0x0FFFFFF0 - 0x0FFFFFF3</td><td>00 00 00 00</td><td>undefined</td></tr><tr><td>sp → 0x0FFFFFF4 - 0x0FFFFFF7</td><td>00 00 00 00</td><td>undefined</td></tr><tr><td>0x0FFFFFF8 - 0x0FFFFFFB</td><td>FF FF FF FF</td><td>4294967295</td></tr><tr><td>0x0FFFFFFC - 0x0FFFFFFF</td><td>00 00 00 00</td><td>00</td></tr></tbody></table>					Address	Binary	Value	0x0FFFFFF0 - 0x0FFFFFF3	00 00 00 00	undefined	sp → 0x0FFFFFF4 - 0x0FFFFFF7	00 00 00 00	undefined	0x0FFFFFF8 - 0x0FFFFFFB	FF FF FF FF	4294967295	0x0FFFFFFC - 0x0FFFFFFF	00 00 00 00	00
Address	Binary	Value																	
0x0FFFFFF0 - 0x0FFFFFF3	00 00 00 00	undefined																	
sp → 0x0FFFFFF4 - 0x0FFFFFF7	00 00 00 00	undefined																	
0x0FFFFFF8 - 0x0FFFFFFB	FF FF FF FF	4294967295																	
0x0FFFFFFC - 0x0FFFFFFF	00 00 00 00	00																	
<table><thead><tr><th>Stack memory areas:</th><th>Free stack</th><th>Callee: f1</th><th>Caller: main</th><th>System stack</th></tr></thead><tbody><tr><td></td><td></td><td></td><td></td><td></td></tr></tbody></table>					Stack memory areas:	Free stack	Callee: f1	Caller: main	System stack										
Stack memory areas:	Free stack	Callee: f1	Caller: main	System stack															



Ejemplo de llamadas a funciones anidadas

ejemplo

- ▶ Comprobar el crecimiento de la pila

Main memory segment		INT Registers	FP Registers	Memory	Stats	Energy (CLK Cycles)
		Data	Text	Stack		
Address	Binary	Value				
sp → 0xFFFFFFF0 - 0xFFFFFFF3	00 00 00 28	40				
0xFFFFFFF0	0x000...	f3	F7	00 00 00 94	148	
0xFFFFFFF1		f2	FB	FF FF FF FF	4294967295	
0xFFFFFFF2		f1	FF	00 00 00 00	00	
		main				
0xFFFF...						

Stack memory areas: ⓘ

Free stack

Callee: f3 Caller: f2 ... 2 System stack

Llamadas a funciones

- ▶ Convenio simplificado
 - ▶ La pila no necesita estar alineada a 8 bytes
- ▶ Alerta si se incumple el convenio de paso de parámetros y uso de pila

Integer Registers	
Register Name	Usage
zero	Constant 0
ra	Return address (routines/functions)
sp	Stack pointer
gp	Global pointer
tp	Thread pointer
t0..t6	Temporary (NOT preserved across calls)
s0..s11	Saved temporary (preserved across calls)
a0, a1	Arguments for functions / return value
a2..a7	Arguments for functions

Floating-point registers	
Register Name	Usage
ft0..ft11	Temporary (NOT preserved across calls)
fs0..fs11	Saved temporary (preserved across calls)
fa0, fa1	Arguments for functions / return value
fa2..fa7	Arguments for functions

Detección de errores en el convenio de paso de parámetros

- ▶ Corregir los fallos que aparecen en el ejemplo por un uso incorrecto del convenio de paso de parámetros y uso de pila
- ▶ Modelo simplificado que se utiliza actualmente
 - ▶ La pila no tiene porque estar alineada a 8

Integer Registers	
Register Name	Usage
zero	Constant 0
ra	Return address (routines/functions)
sp	Stack pointer
gp	Global pointer
tp	Thread pointer
t0..t6	Temporary (NOT preserved across calls)
s0..s11	Saved temporary (preserved across calls)
a0, a1	Arguments for functions / return value
a2..a7	Arguments for functions
Floating-point registers	
ft0..ft11	Temporary (NOT preserved across calls)
fs0..fs11	Saved temporary (preserved across calls)
fa0, fa1	Arguments for functions / return value
fa2..fa7	Arguments for functions

ejemplo

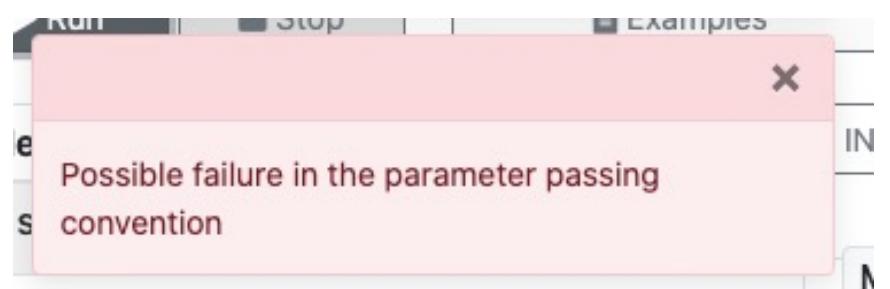
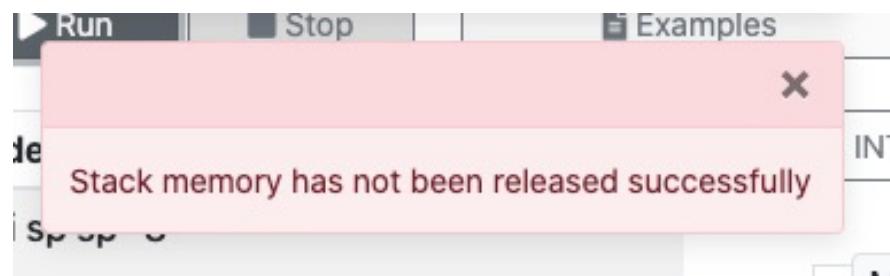
```
.data
.text
max:    addi    sp, sp -8
        sw      s0, 0(sp)
        mv      s0, a0
        mv      s1, a1
        bge   s0, s1, bigger
        mv      a0, s1
        jr      ra
bigger: mv      a0, s0
        jr      ra

main:   li      a0, 8
        li      a1, 7
        jal     ra, max
        li      a7, 1
        ecall
        li a7, 10
        ecall
```



Detección de errores en el convenio de paso de parámetros

▶ Fallos detectados:



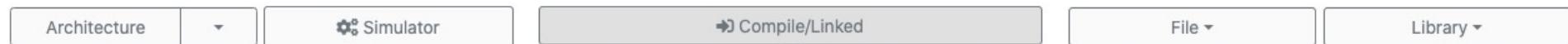
ejemplo

```
.data
.max:      addi    sp, sp -8
           sw     s0, 0(sp)
           mv     s0, a0
           mv     s1, a1
           bge   s0, s1, bigger
           mv     a0, s1
           jr     ra
.bigger:   mv     a0, s0
           jr     ra

.text
main:      li      a0, 8
           li      a1, 7
           jal    ra, max
           li      a7, 1
           ecall
           li      a7, 10
           ecall
```

Creación de librerías

ejemplo



➊ Assembly:

```
1 .globl max, min
2
3 .text
4
5
6
7 max:      bge a0, a1, bigger
8         mv  a0, a1
9     bigger: jr  ra
10
11
12 min:      ble a0, a1, minor
13         mv  a0, a1|
14     minor: jr  ra
15
```

Creación de librerías

The screenshot shows the ARCOS software interface. At the top, there are several tabs: 'Architecture' (with a dropdown arrow), 'Simulator' (with a CPU icon), 'Compile/Linked' (highlighted in grey), 'File' (with a dropdown arrow), and 'Library' (with a dropdown arrow). The 'Library' tab has a red dashed box around its dropdown menu, which contains three items: '+ Create', 'Load Library', and 'Remove'. The main area is titled 'Assembly:' with a small info icon. A red dashed box highlights the first few lines of assembly code:

```
1 .globl max, min
2
3 .text
4
5
6
7 max:      bge a0, al, bigger
8         mv  a0, al
9         bigger: jr  ra
10
11
12 min:      ble a0, al, minor
13         mv  a0, al
14         minor: jr  ra
15
```

Uso de librerías

ejemplo

The screenshot shows the ARCOS simulator interface. At the top, there are tabs for 'Architecture' (selected), 'Simulator', 'Compile/Linked' (disabled), 'File', and 'Library'. A red dashed box highlights the 'Library' dropdown menu, which includes options: '+ Create', 'Load Library' (with an upward arrow icon), and 'Remove' (with a trash bin icon). On the left, under the 'Assembly' tab, is the assembly code:

```
1 #  
2 # Creator (https://creatorsim.github.io/creator/)  
3 #  
4  
5 .text  
6  
7     main:    li  a0, 5  
8         li  a1, 10  
9         jal ra, max  
10        li  a7, 1  
11        ecall  
12  
13        li  a0, '\n'  
14        li  a7, 11  
15        ecall  
16  
17        li  a0, 5  
18        li  a1, 10  
19        jal ra, min  
20        li  a7, 1  
21        ecall  
22  
23        li  a0, '\n'  
24        li  a7, 11  
25        ecall
```

Two red arrows point from the text 'Llamadas' (Calls) to the 'jal ra, max' and 'jal ra, min' instructions in the assembly code.

Llamadas



Librería cargada

The screenshot shows the ARCOS tool interface with the following components:

- Top navigation bar: Architecture (dropdown), Simulator, Compile/Linked, File (dropdown), Library (dropdown), Configuration, Info.
- Assembly view: A code editor titled "Assembly" containing assembly language code. The code defines a "main" function that performs two operations: calculating the maximum of two numbers (using a "max" library function) and calculating the minimum of two numbers (using a "min" library function). Both operations involve loading values into registers a0 and a1, calling the respective library functions, and then printing the result to the console.
- Library tags panel: A sidebar titled "Library tags:" containing two entries: "max" and "min". The "max" entry has a blue highlighted status, while "min" is unhighlighted. This panel is circled with a red dotted line.

```
1 #
2 # Creator (https://creatorsim.github.io/creator/)
3 #
4
5 .text
6
7     main: li a0, 5
8         li a1, 10
9         jal ra, max
10        li a7, 1
11        ecall
12
13        li a0, '\n'
14        li a7, 11
15        ecall
16
17        li a0, 5
18        li a1, 10
19        jal ra, min
20        li a7, 1
21        ecall
22
23        li a0, '\n'
24        li a7, 11
25        ecall
26
27
28
29
```

Uso de librerías

Architecture ▾ # Assembly ⏪ Reset ⏴ Inst. ⏴ Run ⏴ Stop Examples Calculator Configuration Info

Break	Address	Label	User Instruction	Loaded Instructions
0x0		max	<<Hidden>>	<<Hidden>>
0xc		min	<<Hidden>>	<<Hidden>>
0x18		main	li a0 5	addi a0 x0 5 Next
0x1c			li a1 10	addi a1 x0 10
0x20			jal ra max	jal ra 0x0
0x24			li a7 1	addi a7 x0 1
0x28			ecall	ecall
0x2c			li a0 10	addi a0 x0 10
0x30			li a7 11	addi a7 x0 11
0x34			ecall	ecall

INT Registers FP Registers Memory Stats Energy (CLK Cycles)

Main memory segment

Data Text Stack

Address	Binary	Value
gp → 0x00000000 - 0x00000003	max 01 00 B5 50	0
0x00000004 - 0x00000007	00 05 85 13	
0x00000008 - 0x0000000B	00 00 80 67	***** 0
0x0000000C - 0x0000000F	min 01 00 A5 D0	0

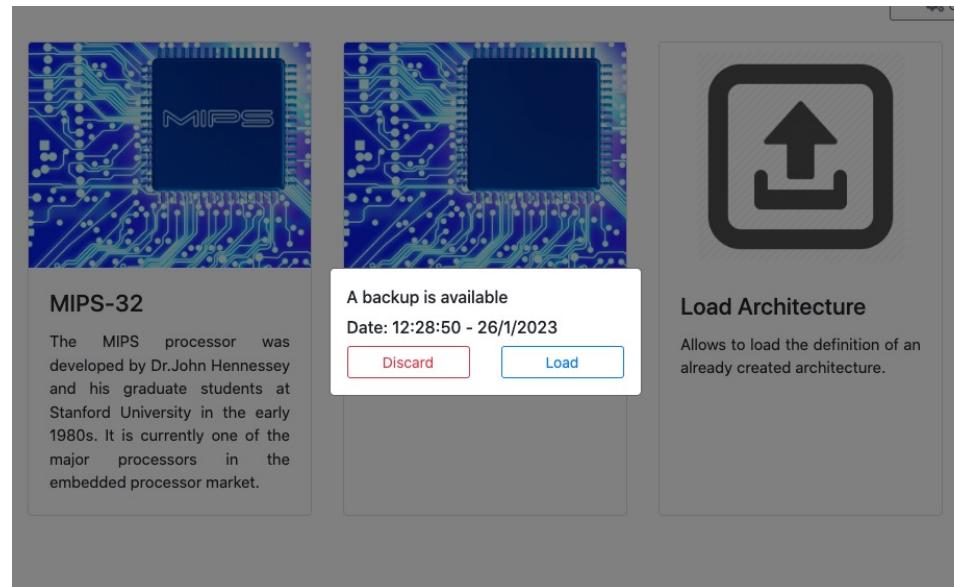


Errores en tiempo de ejecución

- ▶ Programa sin función main
- ▶ Puntero de pila (sp) apunta al segmento de datos o de texto
- ▶ Escritura en el segmento de texto
- ▶ Acceso a una posición de memoria no alineada

Caché del navegador para recuperación de errores

- ▶ El programa que se está editando se guarda en la caché del navegador cada vez que se compila
- ▶ Si el navegador falla se puede recuperar el programa al volverlo a cargar



Contenido (RISC-V)

- ▶ Juego de instrucciones soportado
- ▶ Visión del estudiante:
 - ▶ Características del entorno
 - ▶ Edición y compilación de programas
 - ▶ Ejecución y depuración de programas
 - ▶ Bibliotecas de funciones
 - ▶ Facilidades para entender el empleo de funciones y uso de pila
- ▶ Visión del profesor:
 - ▶ Soporte a la corrección de prácticas
 - ▶ Soporte a la creación de material didáctico
 - ▶ Capacidades para extender el juego de instrucciones y crear nuevas arquitecturas

Soporte a la corrección de prácticas

- ▶ Ejecución en línea de comandos
- ▶ Prerrequisitos:
 - ▶ Linux, node.js y npm
- ▶ Pasos:
 - ▶ Descargar el repositorio:
 - ▶ `git clone https://github.com/creatorsim/creator.git`
 - ▶ `cd creator`
 - ▶ Instalar los paquetes:
 - ▶ `npm install terser jshint colors yargs readline-sync`



Compilación y ejecución de un programa

▶ `./creator.sh -h`

```
CREATOR
-----
version: 3.2
website: https://creatorsim.github.io/

Usage: creator.sh -a <file name> -s <file name>
Usage: creator.sh -h

Options:
  --version      Show version number          [boolean]
  -a, --architecture Architecture file        [string] [required] [default: ""]
  -s, --assembly   Assembly file              [string] [required] [default: ""]
  -d, --directory  Assemblies directory       [string] [default: ""]
  -l, --library    Assembly library file     [string] [default: ""]
  -r, --result     Result file to compare with [string] [default: ""]
  --describe      Help on element            [string] [default: ""]
  --maxins        Maximum number of instructions to be executed
                  [string] [default: "1000000"]
  -o, --output     Define output format       [string] [default: "normal"]
  --color          Colored output             [boolean] [default: false]
  -h, --help        Show help                 [boolean]

Examples:
  ./creator.sh To show examples.
```



Ejecución de un programa

▶ `./creator.sh -a architecture/RISC_V_RV32IMFD.json -s ./factorial.s`

```
CREATOR
-----
version: 3.2
website: https://creatorsim.github.io/

[./factorial.s]
120
[Architecture] Architecture 'architecture/RISC_V_RV32IMFD.json' loaded successfully.
[Library] Without library
[Compile] Code './factorial.s' compiled successfully.
[Execute] Executed successfully.
[FinalState] cr[PC]:0x18; ir[ra,x1]:0x8; ir[t0,x5]:0x2; ir[t1,x6]:0x5; ir[a0,x10]:0x78; ir[a7,x17]:0xa; keyboard[0x0]:'';
display[0x0]:'120';
```

Ejecución de un programa y comprobación de resultados

- ▶ Podemos comparar la salida con un resultado de referencia
- ▶ Ejemplo de resultado de referencia para comparar solo la salida:

referencia.txt

```
display[0x0]: '120';
```

- ▶ Ejecución y comparación con salida de referencia:

```
▶ ./creator.sh -a architecture/RISC_V_RV32IMFD.json -s ./factorial.s  
-r referencia.txt
```



Ejemplo de salida correcta

- ▶ ./creator.sh -a architecture/ RISC_V_RV32IMFD.json -s factorial.s -r referencia.txt

```
CREATOR
-----
version: 3.2
website: https://creatorsim.github.io/

[./factorial.s]
120
[Architecture] Architecture 'architecture/RISC_V_RV32IMFD.json' loaded successfully.
[Library] Without library
[Compile] Code './factorial.s' compiled successfully.
[Execute] Executed successfully.
[State] Equals
```

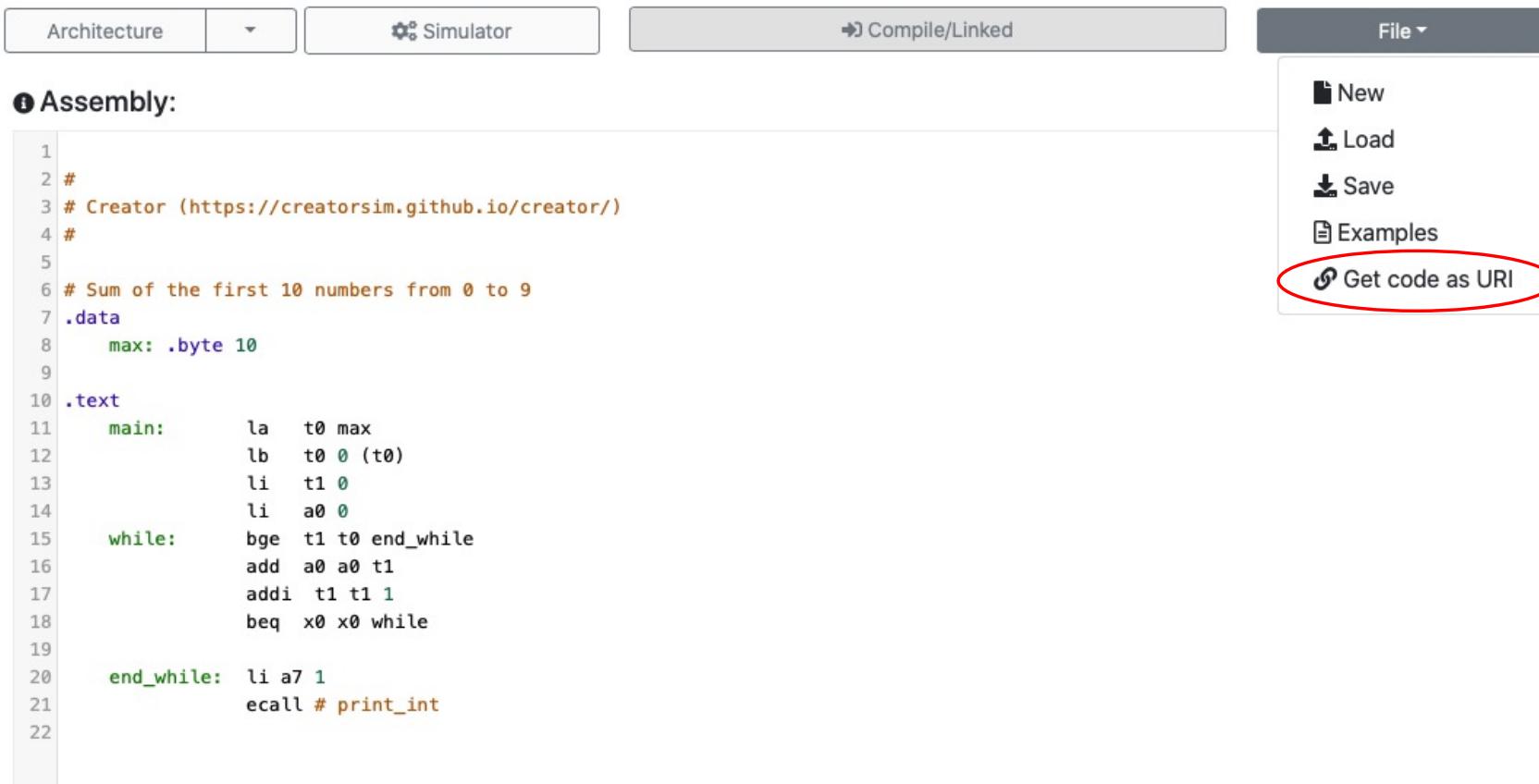
Ejemplo de salida incorrecta

- ▶ ./creator.sh -a architecture/ RISC_V_RV32IMFD.json -s factorial.s -r referencia.txt

```
CREATOR
-----
version: 3.2
website: https://creatorsim.github.io/

[./factorial.s]
808
[Architecture] Architecture 'architecture/RISC_V_RV32IMFD.json' loaded successfully.
[Library] Without library
[Compile] Code './factorial.s' compiled successfully.
[Execute] Executed successfully.
[State] Different: display[0x0]='120' is ='808'.
```

Ayuda a la creación de materiales docentes



The screenshot shows the ARCOS assembly editor interface. At the top, there are tabs for "Architecture" (selected), "Simulator", "Compile/Linked" (disabled), and "File". The "File" menu is open, showing options: New, Load, Save, Examples, and Get code as URI, with the last option circled in red.

Assembly:

```
1
2 #
3 # Creator (https://creatorsim.github.io/creator/)
4 #
5
6 # Sum of the first 10 numbers from 0 to 9
7 .data
8     max: .byte 10
9
10 .text
11     main:    la    t0 max
12         lb    t0 0 (t0)
13         li    t1 0
14         li    a0 0
15     while:   bge   t1 t0 end_while
16         add   a0 a0 t1
17         addi  t1 t1 1
18         beq   x0 x0 while
19
20     end_while: li a7 1
21         ecall # print_int
22
```

Ayuda a la creación de materiales docentes

The screenshot shows a debugger interface with the following components:

- Top navigation bar: Architecture (dropdown), Simulator, Compile/Linked, File (dropdown).
- Assembly code area:

```
1 #
2 # Creator (https://creatorsim.github.io/creator/)
3 #
4 #
5 # Sum of the first 10 numbers from 0 to 9
6 .data
7     max: .byte 10
8
9 .text
10    main:    la    t0 max
11        lb    t0 0 (t0)
12        li    t1 0
13        li    a0 0
14    while:   bge   t1 t0 end_while
15        add   a0 a0 t1
16        addi  t1 t1 1
17        beq   x0 x0 while
18
19    end_while: li    a7 1
20        ecall # print_int
21
22
```
- File menu dropdown open, showing options: New, Load, Save, Examples, Get code as URI.
- A modal dialog titled "URI" is displayed, containing the URL:

```
https://creatorsim.github.io/creator/?architecture=RISC-V%20(RV32IMFD)&asm=%0A%23%0A%23%20Creator%20https%3A%2F%2Fcreatorsim.github.io%2Fcreator%2F)%0A%23%0A%0A%23%20Sum%20of%20the%20first%2010%20numbers%20from%200%20to%209%0A.data%0A%09max
```
- A red arrow points from the "Get code as URI" menu item to the "URI" dialog.
- A blue button labeled "enlace" is located to the right of the "URI" dialog.

Capacidades para extender el juego de instrucciones y crear nuevas arquitecturas

Architecture Info	Memory Layout	Register File	Instructions	Pseudoinstructions	Directives
Architecture general information:					
Field	Value	Actions			
Name	RISC-V (RV32IMFD)				
Bits	32	<button>Edit</button>	<button>Reset</button>		
Data Format	Big Endian	<button>Edit</button>	<button>Reset</button>		
Memory Alignment	Enabled	<button>Edit</button>	<button>Reset</button>		
Main Function	main	<button>Edit</button>	<button>Reset</button>		
Passing Convention	Enabled	<button>Edit</button>	<button>Reset</button>		
Sensitive Register Name	Enabled	<button>Edit</button>	<button>Reset</button>		



Ejemplo de definición de una instrucción (addi)

CREATOR 3.2 RISC-V (RV32IMFD)
didactic and generic assembly programming simulator

ARCOS uc3m Universidad Carlos III de Madrid
Computer Science and Engineering Department

Assembly Simulator Save Configuration Info

Architecture Info Memory Layout Register File Instructions Pseudoinstructions Directives

Instruction set:

+ New instruction ⚡ Reset Instructions

Name	Co	Extended CO	Nwords	Instruction syntax	Properties	Power Consumption	Fields	Definition	Actions
addi	0010011	000	1	addi rd rs1 imm addi,INT-Reg,INT-Reg,inm-signed		1	<button>View Fields</button>	rsz_name); rd = rs1 + inm;	<button>Edit</button> <button>Delete</button>

Ejemplo de definición de una instrucción (addi)

Edit addi ×

Name: ✓

Type: ✓ ÷

Number of Words: ✓

CLK Cycles: ✓

Number of fields: (Including co and cop) ✓

Properties:

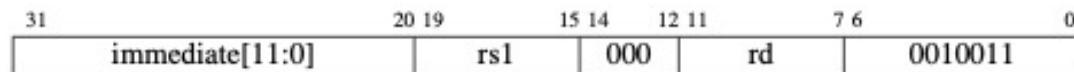
Enter Subroutine Exit Subroutine

[Principal](#) [Fields](#) [Syntax](#) [Definition](#) [Help](#) > »

Cancel Save



Ejemplo de definición de una instrucción (addi)



Edit addi ×

	Name:	Type	Break	Start Bit	End Bit	
Field 0	addi	co		6 ✓	0 ✓	
Field 1	inm ✓	inm-sign ✓	<input type="checkbox"/>	31 ✓	20 ✓	
Field 2	rs1 ✓	INT-Reg ✓		19 ✓	15 ✓	
Field 3	rd ✓	INT-Reg ✓		11 ✓	7 ✓	
Field 4	cop ✓	cop ✓		14 ✓	12 ✓	000 ✓

código de operación

[«](#) [<](#) [Principal](#) [Fields](#) [Syntax](#) [Definition](#) [Help](#) [»](#) [»](#)

[Cancel](#) [Save](#)

Ejemplo de definición de una instrucción (addi)

Edit addi X

Instruction Syntax Definition:

`F0 F3 F2 F1` ✓

Detailed Syntax:

`addi,INT-Reg,INT-Reg,inm-signed`

Instruction Syntax:

`addi rd rs1 inm`

« < Principal Fields Syntax Definition Help › »

Cancel Save

Ejemplo de definición de una instrucción (addi)

Edit addi ×

Assembly Definition:

```
rd = rs1 + inm;
```

✓

[«](#) [‹](#) [Principal](#) [Fields](#) [Syntax](#) **Definition** [Help](#) [›](#) [»](#)

[Cancel](#) [Save](#)

Ejemplo de definición de una instrucción (addi)

Edit addi

x

Assembly help:

Example: reg1=reg2+reg3

« < Principal Fields Syntax Definition Help

Cancel Save



Creación de una nueva pseudoinstrucción

- ▶ Ejemplo:
 - ▶ bltz rs1, offset if (rs1 < 0) PC = PC + offset
 - ▶ Se expande a: blt rs1, zero, offset

The screenshot shows the ARCOS architecture configuration interface. On the left, there is a sidebar with a dropdown menu showing 'Architecture' and three options: 'RISC-V (RV32IMFD)', 'MIPS-32', and 'New Architecture'. A red arrow points from the 'RISC-V (RV32IMFD)' option to the main content area. The main content area has tabs at the top: '# Assembly', 'Simulator', 'Save', and 'Configuration'. Below these tabs, there are several other tabs: 'Architecture Info' (which is selected), 'Memory Layout', 'Register File', 'Instructions', 'Pseudoinstructions', and 'Directives'. The 'Architecture general information:' section contains a table with the following data:

Field	Value	Actions
Name	RISC-V (RV32IMFD)	<input type="button" value="Edit"/> <input type="button" value="Reset"/>
Bits	32	<input type="button" value="Edit"/> <input type="button" value="Reset"/>
Data Format	Big Endian	<input type="button" value="Edit"/> <input type="button" value="Reset"/>
Memory Alignment	Enabled	<input type="button" value="Edit"/> <input type="button" value="Reset"/>
Main Function	main	<input type="button" value="Edit"/> <input type="button" value="Reset"/>
Passing Convention	Enabled	<input type="button" value="Edit"/> <input type="button" value="Reset"/>
Sensitive Register Name	Enabled	<input type="button" value="Edit"/> <input type="button" value="Reset"/>



Creación de una nueva pseudoinstrucción

New Pseudoinstruction X

Name:
blitz rs1 offset ✓

Number of Words:
1 ✓

Number of fields:
2 ✓

[Principal](#) [Fields](#) [Syntax](#) [Definition](#) [Help](#) > »

[Cancel](#) [Save](#)

Creación de una nueva pseudoinstrucción

New Pseudoinstruction ×

	Name:	Type
Field 0	rs1	✓ INT-Reg ✓ ▾
Field 1	offset	✓ inm-signed ✓ ▾

[«](#) [‹](#) [Principal](#) **Fields** [Syntax](#) [Definition](#) [Help](#) [›](#) [»](#)

[Cancel](#) [Save](#)

Creación de una nueva pseudoinstrucción

New Pseudoinstruction

X

Pseudoinstruction Syntax Definition:

bltz F0 F1



Detailed Syntax:

bltz,INT-Reg,inm-signed

Pseudoinstruction Syntax:

bltz rs1 offset

« < Principal Fields Syntax Definition Help > »

Cancel

Save

Creación de una nueva pseudoinstrucción

Edit bltz rs1 offset ×

Pseudoinstruction Definition:

```
blt rs1, zero, offset;
```

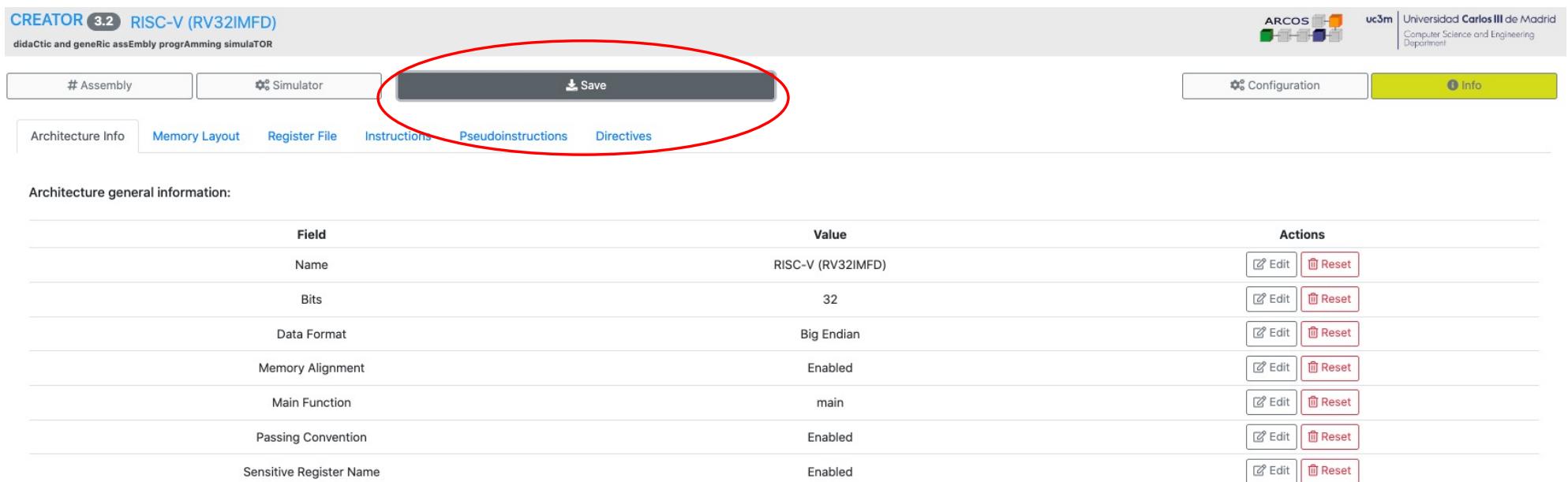
Pseudoinstruction Definition

✓

[«](#) [‹](#) [Principal](#) [Fields](#) [Syntax](#) **Definition** [Help](#) [›](#) [»](#)

[Cancel](#) [Save](#)

Creación de una nueva pseudoinstrucción



The screenshot shows the CREATOR 3.2 RISC-V (RV32IMFD) interface. At the top, there is a header with the title "CREATOR 3.2 RISC-V (RV32IMFD)" and a subtitle "didaCtic and geneRic assEmbly progrAMming simulaTOR". On the right side of the header, there are logos for "ARCOS" and "uc3m Universidad Carlos III de Madrid Computer Science and Engineering Department". Below the header, there is a navigation bar with tabs: "# Assembly", "Simulator", "Save" (which is highlighted with a red oval), "Configuration", and "Info". Underneath the tabs, there is a sub-navigation bar with tabs: "Architecture Info", "Memory Layout", "Register File", "Instructions", "Pseudoinstructions" (which is also highlighted with a red oval), and "Directives". The main content area is titled "Architecture general information:" and contains a table with the following data:

Field	Value	Actions
Name	RISC-V (RV32IMFD)	<input type="button" value="Edit"/> <input type="button" value="Reset"/>
Bits	32	<input type="button" value="Edit"/> <input type="button" value="Reset"/>
Data Format	Big Endian	<input type="button" value="Edit"/> <input type="button" value="Reset"/>
Memory Alignment	Enabled	<input type="button" value="Edit"/> <input type="button" value="Reset"/>
Main Function	main	<input type="button" value="Edit"/> <input type="button" value="Reset"/>
Passing Convention	Enabled	<input type="button" value="Edit"/> <input type="button" value="Reset"/>
Sensitive Register Name	Enabled	<input type="button" value="Edit"/> <input type="button" value="Reset"/>



Creación de una nueva instrucción

- ▶ Ejemplo: fmadd.s rd, rs1, rs2, rs3

fmadd.s rd, rs1, rs2, rs3

$$f[rd] = f[rs1] \times f[rs2] + f[rs3]$$

Floating-point Fused Multiply-Add, Single-Precision. Tipo R4, RV32F y RV64F.

Multiplica los números de punto flotante de precisión simple en $f[rs1]$ y $f[rs2]$, suma el producto sin redondear al número de punto flotante de precisión simple en $f[rs3]$, y escribe el resultado redondeado de precisión simple en $f[rd]$.

31	27 26 25 24	20 19	15 14	12 11	7 6	0
rs3	00	rs2	rs1	rm	rd	1000011

Creación de una nueva instrucción

CREATOR 3.2 RISC-V (RV32IMFD)
didaCtic and geneRic assEmbly progrAmming simulaTOR

Assembly Simulator Save Configuration Info

Architecture Info Memory Layout Register File Instructions Pseudoinstructions Directives

Instruction set:

Name	Co	Extended CO	Nwords	Instruction syntax	Properties	Power Consumption	Fields	Definition	Actions
jalr	1100111	000	1	jalr rd imm (rs1) jalr,INT-Reg,imm-signed,(INT-Reg)	exit_subroutine	1	View Fields	PC = (rs1+imm)&~1; capi_callconv_end(); capi_drawstack_end(PC);	Edit Delete
beq	1100011	000	1	beq rs1 rs2 imm beq,INT-Reg,INT-Reg,offset_words		1	View Fields	if (rs1 === rs2) PC = PC + ((imm << 16) >> 14);	Edit Delete
bne	1100011	001	1	bne rs1 rs2 imm bne,INT-Reg,INT-Reg,offset_words		1	View Fields	if (rs1 != rs2) PC = PC + ((imm << 16) >> 14);	Edit Delete
blt	1100011	100	1	blt rs1 rs2 imm blt,INT-Reg,INT-Reg,offset_words		1	View Fields	if (capi_uint2int(rs1) < capi_uint2int(rs2)) PC = PC + ((imm << 16) >> 14);	Edit Delete
bge	1100011	101	1	bge rs1 rs2 imm bge,INT-Reg,INT-Reg,offset_words		1	View Fields	if (capi_uint2int(rs1) >= capi_uint2int(rs2)) PC = PC + ((imm << 16) >> 14);	Edit Delete



Nueva instrucción: fmadd.s

New Instruction

Name:

fmadd.s

Type:

Arithmetic floating point

Number of Words:

1

CLK Cycles:

4

Number of fields: (Including co and cop)

6

Properties:

Enter Subroutine Exit Subroutine

Principal Fields Syntax Definition Help > »

Cancel Save

fmadd.s rd, rs1, rs2, rs3

$f[rd] = f[rs1] \times f[rs2] + f[rs3]$

Floating-point Fused Multiply-Add, Single-Precision. Tipo R4, RV32F y RV64F.

Multiplica los números de punto flotante de precisión simple en f[rs1] y f[rs2], suma el producto sin redondear al número de punto flotante de precisión simple en f[rs3], y escribe el resultado redondeado de precisión simple en f[rd].

31	27 26 25 24	20 19	15 14	12 11	7 6	0
rs3	00	rs2	rs1	rm	rd	1000011



Nueva instrucción: fmadd.s

The diagram illustrates the bit layout of the fmadd.s instruction and its configuration in a software interface.

Bit Layout:

31	rs3	27 26 25 24	20 19	15 14	12 11	7 6	0
	00	rs2	rs1	rm	rd	1000011	

Software Interface - Edit fmadd.s:

Name:	Type	Break	Start Bit	End Bit
Field 0	fmadd.s	co	6	0 100011 ✓
Field 1	rd	SFP-Reg ✓	11	7 ✓
Field 2	rs1	SFP-Reg ✓	19	15 ✓
Field 3	rs2	SFP-Reg ✓	24	20 ✓
Field 4	co	cop ✓	26	25 00 ✓
Field 5	rs3	SFP-Reg ✓	31	27 ✓

Navigation and Buttons:

« < Principal Fields Syntax Definition Help > »

Cancel Save

Nueva instrucción: fmadd.s

31	27 26	25 24	20 19	15 14	12 11	7 6	0
rs3	00	rs2	rs1	rm	rd	1000011	

F5 F3 F2 F1 F0

Edit fmadd.s



Instruction Syntax Definition:

F0 F1 F2 F3 F5



Detailed Syntax:

fmadd.s,SFP-Reg,SFP-Reg,SFP-Reg,SFP-Reg

Instruction Syntax:

fmadd.s rd rs1 rs2 rs3

« < Principal Fields Syntax Definition Help > »

Cancel

Save

Nueva instrucción: fmadd.s

New Instruction



Assembly Definition:

```
rd=rs1*rs2+rs3;
```



Instruction Definition

« < Principal Fields Syntax **Definition** Help > »

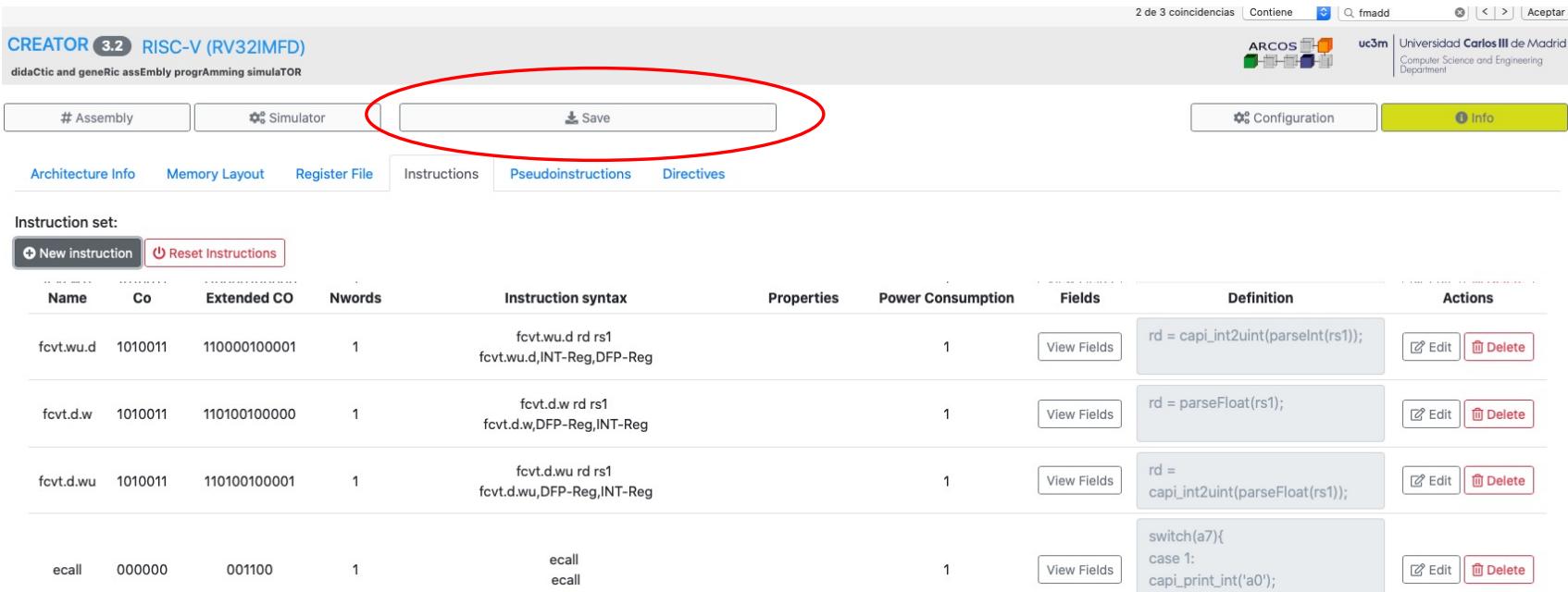
fmadd.s rd, rs1, rs2, rs3 $f[rd] = f[rs1] \times f[rs2] + f[rs3]$
Floating-point Fused Multiply-Add, Single-Precision. Tipo R4, RV32F y RV64F.
Multiplica los números de punto flotante de precisión simple en f[rs1] y f[rs2], suma el producto sin redondear al número de punto flotante de precisión simple en f[rs3], y escribe el resultado redondeado de precisión simple en f[rd].

31	27 26	25 24	20 19	15 14	12 11	7 6	0
rs3	00	rs2	rs1	rm	rd	1000011	

Cancel

Save

Creación de una nueva instrucción

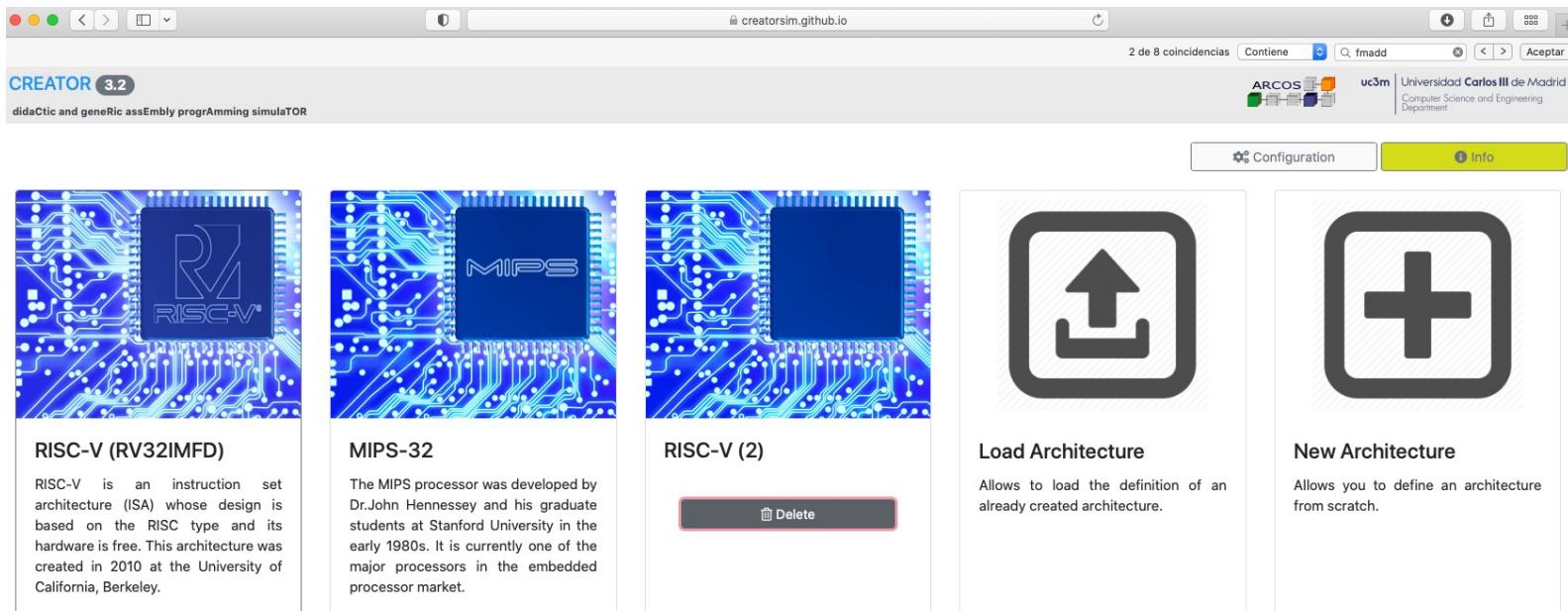


The screenshot shows the 'Instructions' tab of the RISC-V Instruction Creator tool. A red oval highlights the 'Save' button in the top navigation bar. The interface includes tabs for Architecture Info, Memory Layout, Register File, Instructions (selected), Pseudoinstructions, and Directives. Below the tabs, there's a section for the instruction set with buttons for New instruction and Reset Instructions. The main table lists four instructions: fcvt.wu.d, fcvt.d.w, fcvt.d.wu, and ecall, each with its properties, power consumption, fields, definition, and actions (Edit, Delete).

Name	Co	Extended Co	Nwords	Instruction syntax	Properties	Power Consumption	Fields	Definition	Actions
fcvt.wu.d	1010011	110000100001	1	fcvt.wu.d rd rs1 fcvt.wu.d,INT-Reg,DFP-Reg		1	<button>View Fields</button>	rd = capi_int2uint(parseInt(rs1));	<button>Edit</button> <button>Delete</button>
fcvt.d.w	1010011	110100100000	1	fcvt.d.w rd rs1 fcvt.d.w,DFP-Reg,INT-Reg		1	<button>View Fields</button>	rd = parseFloat(rs1);	<button>Edit</button> <button>Delete</button>
fcvt.d.wu	1010011	110100100001	1	fcvt.d.wu rd rs1 fcvt.d.wu,DFP-Reg,INT-Reg		1	<button>View Fields</button>	rd = capi_int2uint(parseFloat(rs1));	<button>Edit</button> <button>Delete</button>
ecall	000000	001100	1	ecall ecall		1	<button>View Fields</button>	switch(a7){ case 1: capi_print_int('a0'); break;	<button>Edit</button> <button>Delete</button>



Nueva arquitectura



Ejemplo

➊ Assembly:

ejemplo

```
1 .data
2
3     a: .float 4
4     b: .float 5.5
5     c: .float 2.3
6
7 .text
8
9 main:
10    li t0, a
11    flw ft1, 0(t0)
12
13    li t0, b
14    flw ft2, 0(t0)
15
16    li t0, c
17    flw ft3, 0(t0)
18
19    fmadd.s fa0, ft1, ft2, ft3
20
21    li a7, 2
22    ecall
```



API para la definición de instrucciones

- ▶ [API de ayuda](#) para definición de instrucciones

- ▶ Instrucción: lw rd inm (rs1)

- ▶ Definición:

```
var addr = capi_int2uint(rs1)+inm;  
rd = capi_mem_read(addr, 'w', rd_name);
```

- ▶ Instrucción: sb rd inm (rs1)

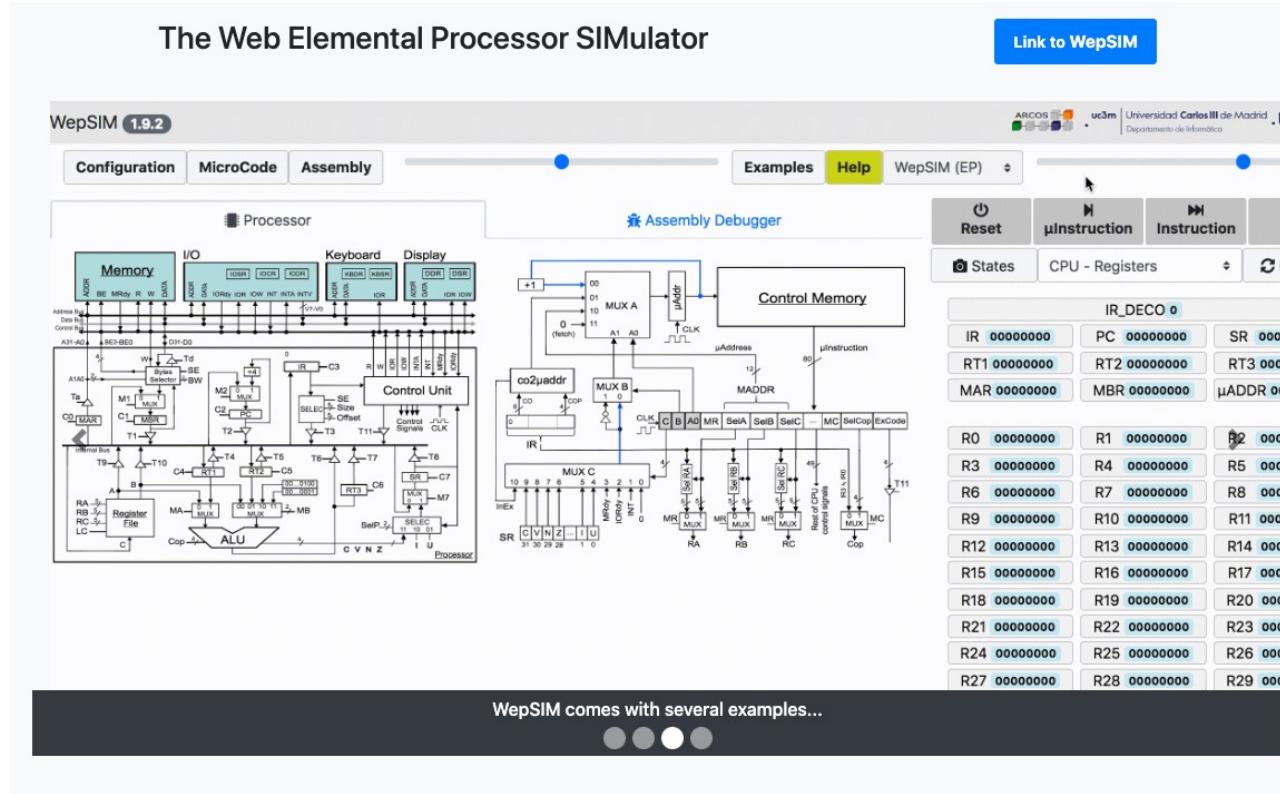
- ▶ Definición:

```
capi_mem_write(rs1+inm, rd, 'b', rs2_name);
```

Extensiones futuras

- ▶ Simulador de caché
- ▶ Registros e instrucciones vectoriales
- ▶ Simulador de pipeline
- ▶

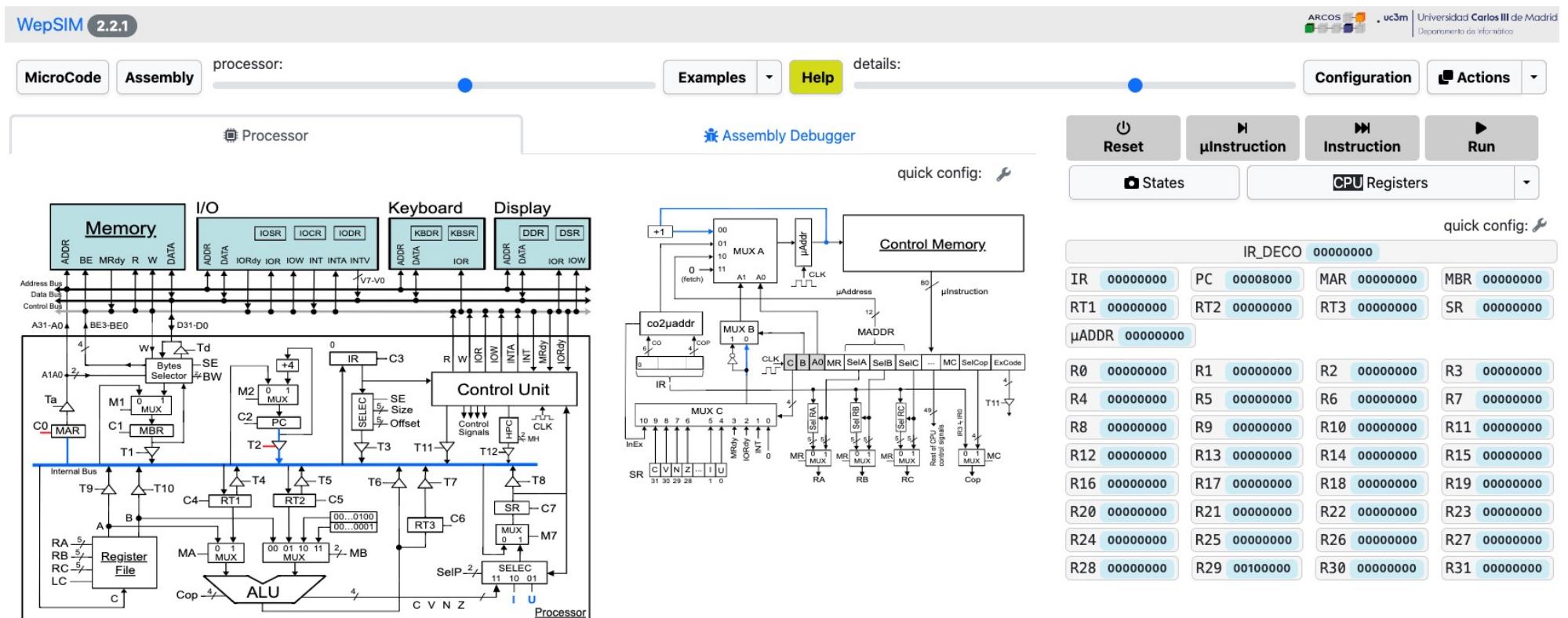
Otro simulador: WepSim



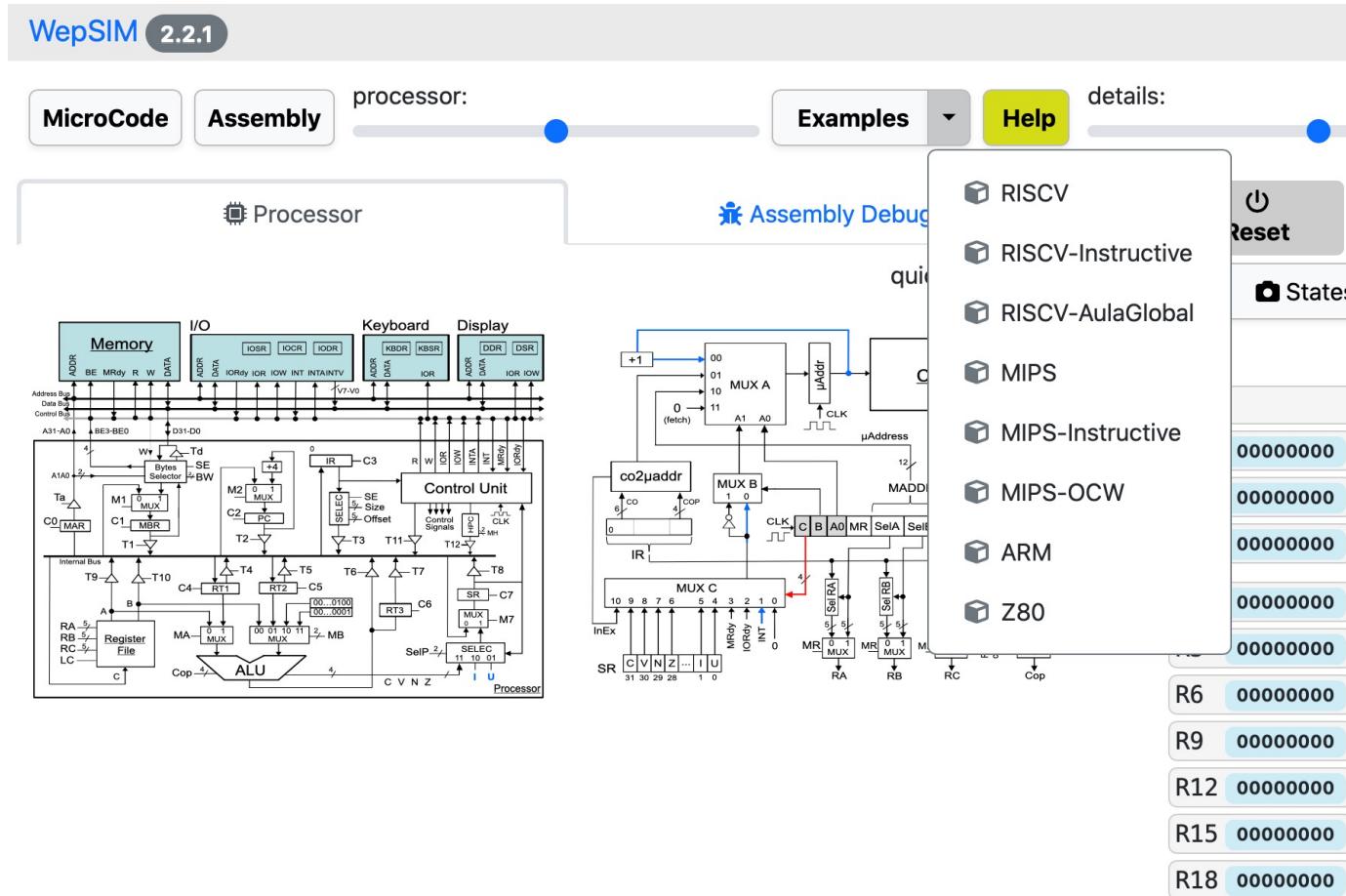
<https://wepsim.github.io>



WepSim



Ejemplos de juegos de instrucciones



Ejecución de un programa (ensamblador)

WepSIM 2.2.1

ARCOS uc3m Universidad Carlos III de Madrid
Departamento de Informática

MicroCode Assembly processor: Examples Help details: Configuration Actions

Processor Assembly Debugger

labels	address	breakpoint	content	assembly	pseudoinstructions
.text					
	0x8004	-	0x59420000	sw a0 0 (sp)	sw a0 0 (sp)
	0x8008	-	0x15600000	lui a1 0x0	li a1 0
	0x800c	-	0x696b0000	addu a1 a1 0x0	
b1	0x8010	-	0x2540000c	beq a0 zero f1	beqz a0 f1
	0x8014	-	0x8d6b5000	add a1 a1 a0	add a1 a1 a0
	0x8018	-	0x654a0fff	addi a0 a0 -1	addi a0 a0 -1
	0x801c	-	0x2400ffff	beq zero zero b1	beq zero zero b1
f1	0x8020	-	0x45420000	lw a0 0 (sp)	lw a0 0 (sp)
	0x8024	-	0x64420004	addi sp sp 4	addi sp sp 4
	0x8028	-	0x20010000	jalr zero ra 0	jr ra
main	0x802c	-	0x64420ffc	addi sp sp -4	addi sp sp -4
	0x8030	-	0x58220000	sw ra 0 (sp)	sw ra 0 (sp)
	0x8034	-	0x15400000	lui a0 0x0	li a0 2
	0x8038	-	0x694a0002	addu a0 a0 0x2	
	0x803c	-	0x1c2ffffc0	jal ra accu	jal ra accu
	0x8040	-	0x44220000	lw ra 0 (sp)	lw ra 0 (sp)
	0x8044	-	0x64420004	addi sp sp 4	addi sp sp 4
	0x8048	-	0x20010000	jalr zero ra 0	jr ra

Reset µInstruction Instruction Run

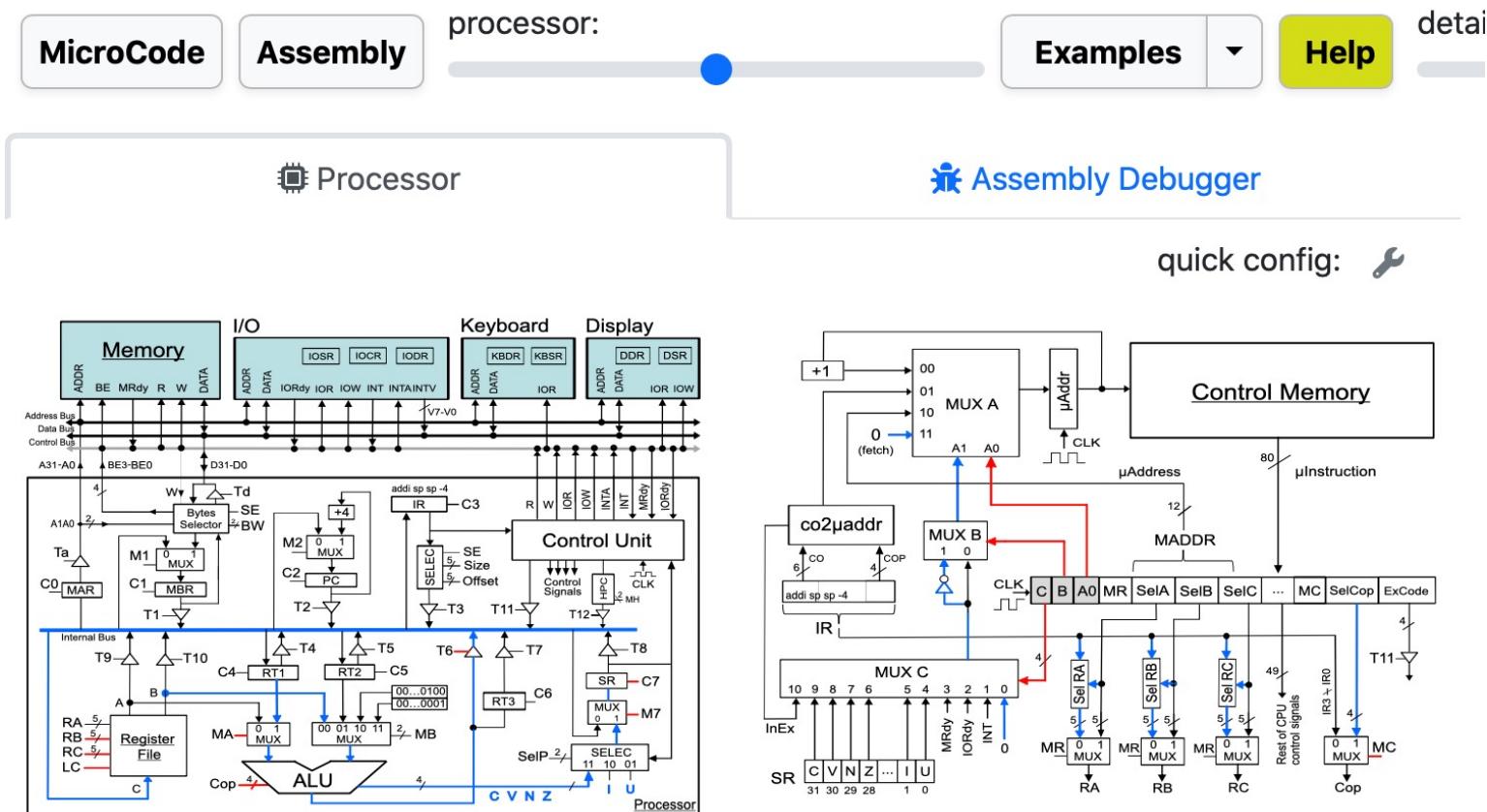
States CPU Registers

quick config: IR_DECO 00000000

IR 00000000	PC 0000802C	MAR 00000000
MBR 00000000	RT1 00000000	RT2 00000000
RT3 00000000	SR 00000000	µADDR 00000000
R0 00000000	R1 00000000	R2 00100000
R3 00000000	R4 00000000	R5 00000000
R6 00000000	R7 00000000	R8 00000000
R9 00000000	R10 00000000	R11 00000000
R12 00000000	R13 00000000	R14 00000000
R15 00000000	R16 00000000	R17 00000000
R18 00000000	R19 00000000	R20 00000000
R21 00000000	R22 00000000	R23 00000000
R24 00000000	R25 00000000	R26 00000000
R27 00000000	R28 00000000	R29 00000000
R30 00000000		R31 00000000



Ejecución del programa (procesador)



XIII Seminario de Invierno CAPAP-H, Almería, 1, 2 y 3 de febrero de 2023

CREATOR como herramienta docente para la enseñanza de la programación en ensamblador con RISC V

Félix García Carballeira

felix.garcia@uc3m.es