

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

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 (editar, compilar y simular la ejecución de programas)
- ▶ Posibilidad de definir y trabajar con diferentes arquitecturas y lenguajes ensamblador
 - ▶ Características básicas (nº de bits, registros, ...)
 - ▶ Instrucciones
 - ▶ Pseudoinstrucciones
 - ▶ Directivas



Facilidades de Creator desde el punto de vista docente

- ▶ Entender la representación de datos e instrucciones
- ▶ Entender la diferencia entre instrucciones y pseudoinstrucciones
- ▶ Entender el flujo de ejecución de un programa en ensamblador conociendo en todo momento la instrucción en curso y la siguiente (útil en bucles)
- ▶ Entender el convenio de paso de parámetros con alertas cuando no se respeta
- ▶ Entender el concepto de biblioteca de funciones y su uso

CREATOR

didaCtic and geneRic assEmbly progrAmming simulaTOR

Access to CREATOR

The screenshot shows the CREATOR web interface for RISC-V-like assembly programming. The main area displays a table of loaded instructions with columns for Break, Address, Label, User Instruction, and Loaded Instructions. The loaded instructions include addi, jal, li, ecall, and add t1 t2 a1/a2. A green bar highlights the instruction at address 0x34: add a0 t1 zero. To the right, there's a register viewer showing INT Registers, Memory, and Configuration. Below the table is a large button labeled "Execute assembly programs". At the bottom, there are "Clear" and "Edit" buttons.

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:

Signed	Unsigned	IEEE 754	Hexadecimal
<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Name	Alias	All	

Register name representation:

PC	SR	RA	SP	GPR
zero 0	ra 24	sp 208405482	gp 0	
tp 4	10 0	11 -34	12 99	
sd 8	s1 0	a0 23	a1 -77	
st 12	s2 0	a4 0	a5 0	
at 16	a7 0	s2 0	s3 0	
ta 20	s5 0	s6 0	s7 0	
tf 24	s9 0	s10 0	s11 0	
tl 28	t4 208405482	t5 0	t6 0	

Execute assembly programs

Clear Edit

<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
- ▶ 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 paso de parámetros a funciones
- ▶ 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

Juego de instrucciones soportado (RV32IMFD)

98 instrucciones y pseudoinstrucciones

- ▶ 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

The screenshot shows the initial interface of the RISC-V RV32IMFD simulator. At the top, there is a header bar 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 is a logo for "ARCOS" and "uc3m Universidad Carlos III de Madrid Computer Science and Engineering Department". Below the header, there is a toolbar with several buttons: "Architecture" (selected), "# Assembly", "Reset", "Inst.", "Run", "Stop", "Examples", "Calculator", "Configuration", and "Info" (highlighted in yellow). The main area has two tabs: "Break" and "Address". Under the "Break" tab, there are columns for "Label", "User Instruction", and "Loaded Instructions". Under the "Address" tab, there is a column for "Break". To the right of these tabs, there is a section for "Registers" with tabs for "INT Registers" (selected), "FP Registers", "Memory", "Stats", and "Energy (CLK Cycles)". Below the register tabs, there are two sub-sections: "Register value representation" and "Register name representation". The "Register value representation" section contains buttons for "Signed", "Unsig.", "IEEE 754", and "Hex" (selected). The "Register name representation" section contains buttons for "Name", "Alias", and "All" (selected). Below these sections, there is a grid of 32x4 boxes representing registers. Each box contains a register name and its value in hex. The registers are arranged in four rows of eight. The first row contains: zero | x0 00000000, ra | x1 FFFFFFFF, sp | x2 0FFFFFFC, gp | x3 00000000. The second row contains: tp | x4 00000000, t0 | x5 00000000, t1 | x6 00000000, t2 | x7 00000000. The third row contains: fp | s0 | x8 00000000, s1 | x9 00000000, a0 | x10 00000000, a1 | x11 00000000. The fourth row contains: a2 | x12 00000000, a3 | x13 00000000, a4 | x14 00000000, a5 | x15 00000000. The fifth row contains: a6 | x16 00000000, a7 | x17 00000000, s2 | x18 00000000, s3 | x19 00000000. The sixth row contains: s4 | x20 00000000, s5 | x21 00000000, s6 | x22 00000000, s7 | x23 00000000. The seventh row contains: s8 | x24 00000000, s9 | x25 00000000, s10 | x26 00000000, s11 | x27 00000000. The eighth row contains: t3 | x28 00000000, t4 | x29 00000000, t5 | x30 00000000, t6 | x31 00000000. At the bottom of the interface, there are icons for a monitor, a keyboard, and a search bar. To the right of the search bar, there are buttons for "Clear" and "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)

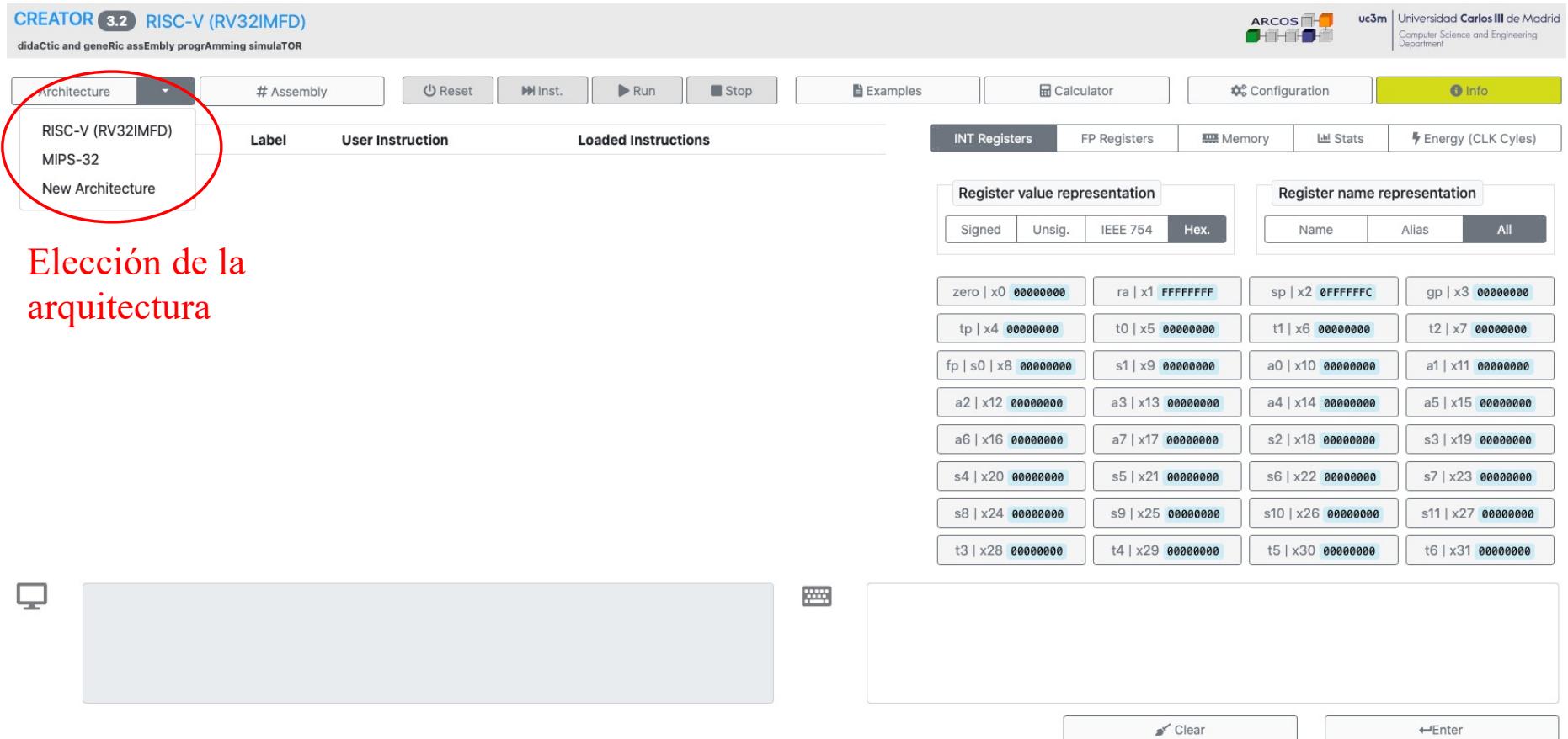
Register value representation Register name representation

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



A screenshot of the ARCOS Creator 3.2 software interface. The top navigation bar includes tabs for Architecture, # Assembly, Reset, Inst., Run, Stop, Examples, Calculator, Configuration, and Info. The Configuration tab is highlighted. Below the tabs, there's a dropdown menu labeled "Architecture" with three options: "RISC-V (RV32IMFD)", "MIPS-32", and "New Architecture". The "New Architecture" option is circled in red. The main workspace shows tables for "User Instruction" and "Loaded Instructions". On the right, there are sections for "Register value representation" and "Register name representation", each with tabs for Signed, Unsig., IEEE 754, and Hex. The register value representation section displays 32 registers (x0 to x31) with their binary values. At the bottom, there are "Clear" and "Enter" buttons. The bottom left corner features the ARCOS logo.

Edición de programas

The screenshot shows the 'CREATOR 3.2 RISC-V (RV32IMFD)' interface. At the top, there is a navigation bar with tabs: Architecture, # Assembly (circled in red), Reset, Run, Stop, Examples, Calculator, Configuration, and Info. Below the navigation bar, there are tabs for 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, each containing a register name and its value in hex. At the bottom, there are icons for monitor and keyboard, and buttons for Clear and Enter.

CREATOR 3.2 RISC-V (RV32IMFD)

Assembly

Break Address Label User instruction Loaded Instructions

INT Registers FP Registers Memory Stats Energy (CLK Cycles)

Register value representation

Register name representation

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



Control de la ejecución

The screenshot shows the CREATOR 3.2 RISC-V (RV32IMFD) simulation interface. A red oval highlights the top navigation bar, which includes buttons for Reset, Inst., Run, and Stop. Below this is a table with columns for Break, Address, Label, User Instruction, and Loaded Instructions. To the right is a section for INT Registers, showing 32-bit register values for zero, ra, sp, tp, t0, t1, fp, s0, a0, a1, a2, a3, a4, a5, a6, a7, s2, s3, s4, s5, s6, s7, s8, s9, t3, t4, t5, and t6. At the bottom are icons for monitor and keyboard, and buttons for Clear and Enter.

Control de la ejecución

Break	Address	Label	User Instruction	Loaded Instructions

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



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)

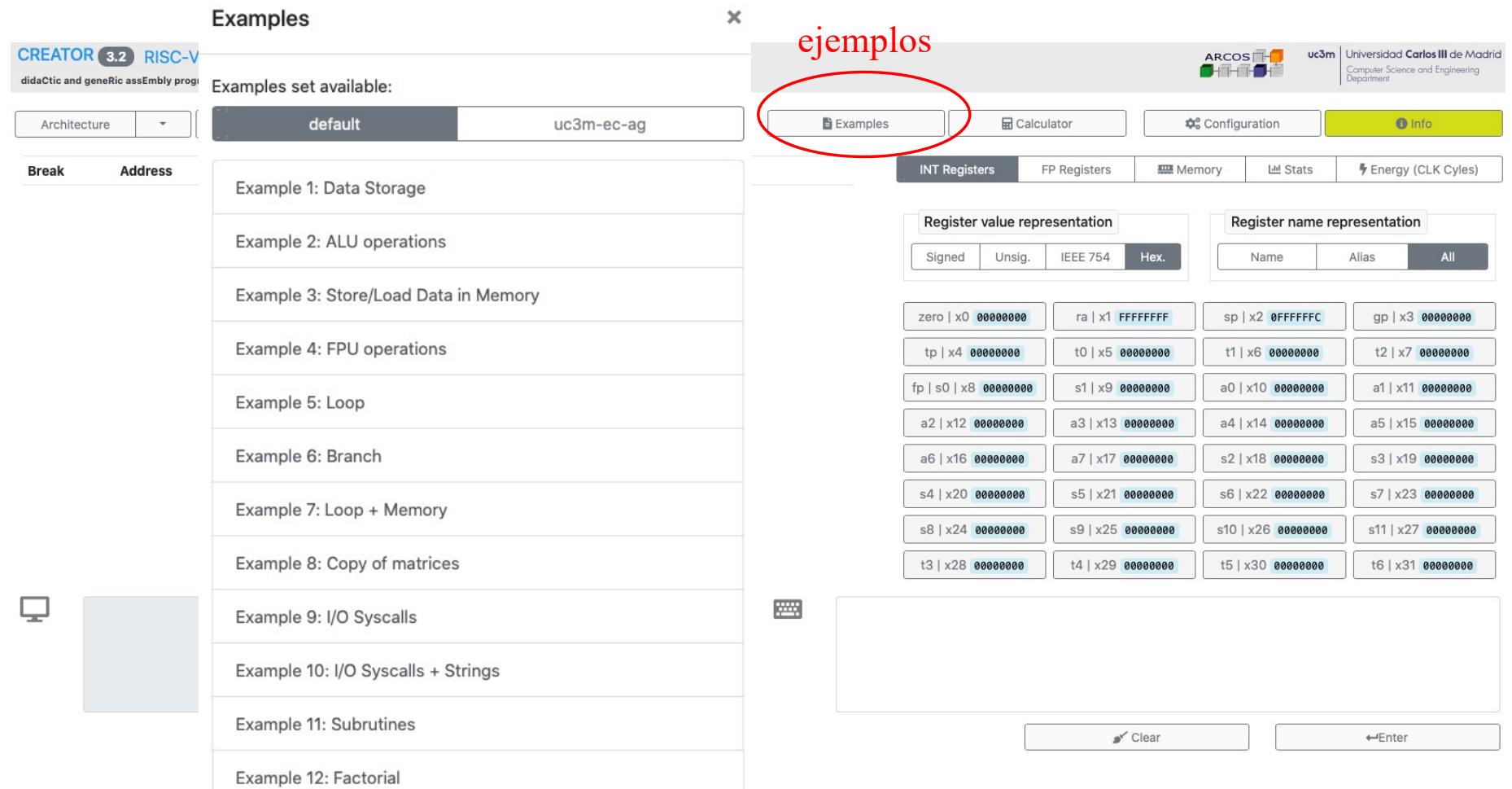
Register value representation Register name representation

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 progrAming 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)

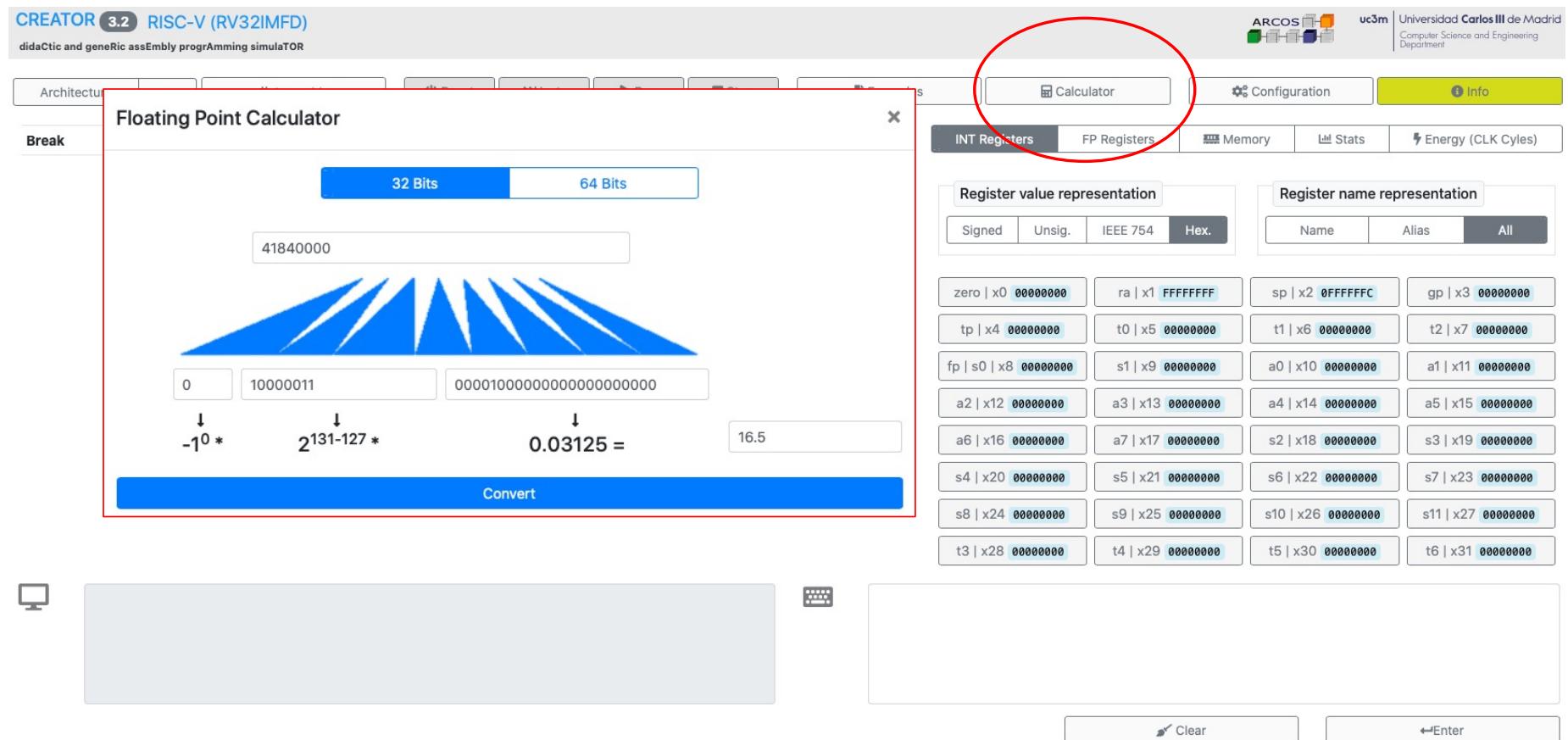
INT Registers FP Registers Memory Stats Energy (CLK Cycles)

Register value representation Register name representation

Signed Unsigned 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



Configuración

The screenshot shows the configuration interface for the RISC-V (RV32IMFD) simulator. It includes the following sections:

- Execution Speed:** A slider with a blue dot at the center.
- Execution Autoscroll:** A toggle switch that is turned on.
- Notification Time:** A slider with a blue dot at the center.
- Instruction Help Size:** A slider with a blue dot at the center.
- Dark Mode:** A toggle switch that is turned off.
- Debug:** A toggle switch that is turned on.

At the bottom right of the configuration window, there are two buttons: "Clear" and "Enter".

The screenshot shows the ARCOS memory dump tool interface. It displays a grid of registers and their values. The columns are labeled:

- Register Name
- Value Representation
- Register Name Representation

	Register Name	Value Representation	Register Name Representation
ra	Unsig.	IEEE 754	Hex.
sp	00000000	ra x1 FFFFFFFF	sp x2 0FFFFFFC
t0	00000000	t0 x5 00000000	t1 x6 00000000
s1	00000000	s1 x9 00000000	a0 x10 00000000
a3	00000000	a3 x13 00000000	a4 x14 00000000
a7	00000000	a7 x17 00000000	s2 x18 00000000
s5	00000000	s5 x21 00000000	s3 x19 00000000
s9	00000000	s9 x25 00000000	s6 x22 00000000
t4	00000000	t4 x29 00000000	s10 x26 00000000
t5	00000000	t5 x30 00000000	s11 x27 00000000
t6	00000000	t6 x31 00000000	

At the top of the memory dump window, there is a red circle around the "Configuration" tab. The tabs include: Registers, FP Registers, Memory, Stats, and Energy (CLK Cycles). The "Registers" tab is currently selected.

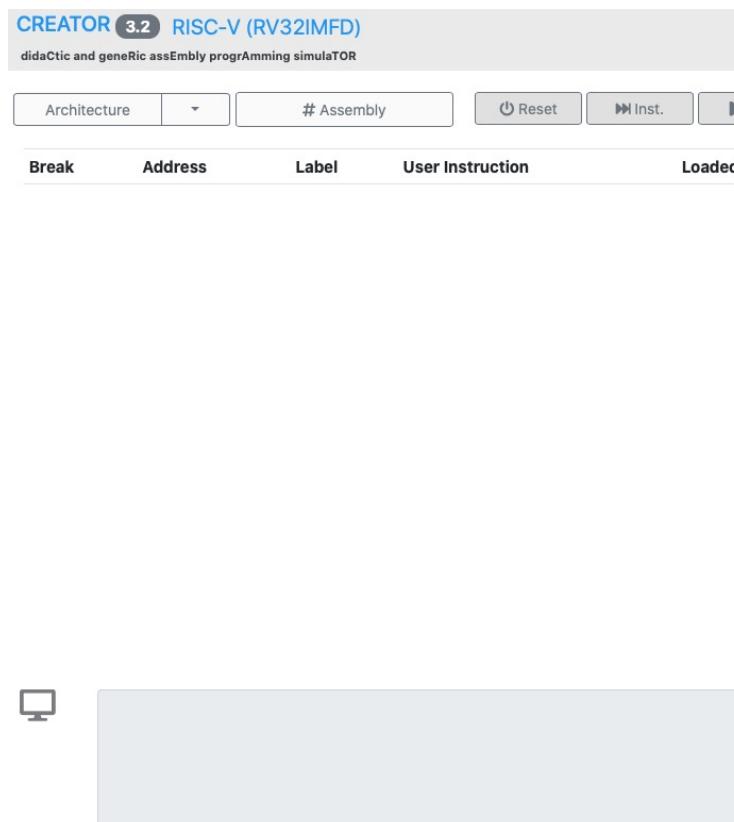


Configuración

The screenshot shows the CREATOR 3.2 RISC-V (RV32IMFD) simulation 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, uc3m, and Universidad Carlos III de Madrid Computer Science and Engineering Department. Below the header, there is a toolbar with various buttons: "Architecture", "# Assembly", "Reset", "Inst.", "Run", "Stop", "Examples", "Calculator", "Configuration", and "Info". The "Info" button is highlighted with a red oval. Below the toolbar, there is a table with columns: "Break", "Address", "Label", "User Instruction", and "Loaded Instructions". To the right of the table, there are tabs for "INT Registers", "FP Registers", "Memory", "Stats", and "Energy (CLK Cyles)". The "INT Registers" tab is selected. Below the tabs, there are two sections: "Register value representation" and "Register name representation". The "Register value representation" section has buttons for "Signed", "Unsig.", "IEEE 754", and "Hex.". The "Register name representation" section has buttons for "Name", "Alias", and "All". Below these sections, there is a grid of 32 registers, each with a name and a hex value. The registers are arranged in four rows of eight. The first row contains: zero | x0 0000000, ra | x1 FFFFFFFF, sp | x2 0FFFFFC, gp | x3 0000000. The second row contains: tp | x4 0000000, t0 | x5 0000000, t1 | x6 0000000, t2 | x7 0000000. The third row contains: fp | s0 | x8 0000000, s1 | x9 0000000, a0 | x10 0000000, a1 | x11 0000000. The fourth row contains: a2 | x12 0000000, a3 | x13 0000000, a4 | x14 0000000, a5 | x15 0000000. The fifth row contains: a6 | x16 0000000, a7 | x17 0000000, s2 | x18 0000000, s3 | x19 0000000. The sixth row contains: s4 | x20 0000000, s5 | x21 0000000, s6 | x22 0000000, s7 | x23 0000000. The seventh row contains: s8 | x24 0000000, s9 | x25 0000000, s10 | x26 0000000, s11 | x27 0000000. The eighth row contains: t3 | x28 0000000, t4 | x29 0000000, t5 | x30 0000000, t6 | x31 0000000. At the bottom of the interface, there are icons for monitor and keyboard, and two buttons: "Clear" and "Enter".



Configuración



The 'Instruction Help' window has a search bar at the top labeled 'Search instruction'. Below it is a section titled 'RISC-V (RV32IMFD) Guide' containing a list of instructions:

- 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 control panel includes:

- ARCOS logo and uc3m logo.
- Universidad Carlos III de Madrid Computer Science and Engineering Department.
- Buttons for 'Configuration' and 'Info' (highlighted with a red circle).
- Buttons for 'Memory', 'Stats', and 'Energy (CLK Cycles)'.
- A 'Register name representation' section with tabs for 'Name', 'Alias', and 'All'. It lists register pairs:
 - 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' and 'Enter' buttons.



Configuración

CREATOR 3.2 RISC-V (RV32IMFD)

didaCtic and geneRic assEmby progrAmming simulaTOR

Architecture	# Assembly	<input type="button" value="Reset"/>	<input type="button" value="Inst."/> Loaded																																																																																										
Break	Address	Label	User Instruction																																																																																										
 RISC-V Reference Guide (CREATOR Simulator)																																																																																													
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">Item</th> <th style="width: 10%;">Calls (call)</th> <th style="width: 10%;">de</th> <th style="width: 10%;">Arguments</th> <th style="width: 10%;">Result</th> </tr> </thead> <tbody> <tr> <td>a0 = integer</td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>lb = float</td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>lb = double</td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>a0 = string add</td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td> Integer in a0</td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td> Float in a0</td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td> Double in a0</td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>a0 = string addr</td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td> a1 = length</td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td> a2 = Address in a0</td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>a0 = ASCII code</td> <td></td> <td></td> <td></td> <td>Char in a0</td> </tr> </tbody> </table>	Item	Calls (call)	de	Arguments	Result	a0 = integer					lb = float					lb = double					a0 = string add					Integer in a0					Float in a0					Double in a0					a0 = string addr					a1 = length					a2 = Address in a0					a0 = ASCII code				Char in a0	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">Register Name</th> <th style="width: 10%;">Usage</th> </tr> </thead> <tbody> <tr> <td>zero</td> <td>Constant</td> </tr> <tr> <td>ra</td> <td>Return address (functions/functions)</td> </tr> <tr> <td>sp</td> <td>Stack pointer</td> </tr> <tr> <td>gp</td> <td>Global pointer</td> </tr> <tr> <td>tp</td> <td>Thread pointer</td> </tr> <tr> <td>ta</td> <td>Temporary (preserved across calls)</td> </tr> <tr> <td>tb..tss</td> <td>Temporary (preserved across calls)</td> </tr> <tr> <td>ta..tsz</td> <td>Arguments for functions / return values</td> </tr> <tr> <td>ta..tz</td> <td>Arguments for functions</td> </tr> <tr> <td colspan="2" style="text-align: center;">Floating-point registers</td> </tr> <tr> <td>fa0..fa11</td> <td>Saved temporary (preserved across calls)</td> </tr> <tr> <td>fb0..fb11</td> <td>Saved temporary (preserved across calls)</td> </tr> <tr> <td>fa0..fa1</td> <td>Arguments for functions / return values</td> </tr> <tr> <td>fa2..fa2</td> <td>Arguments for functions</td> </tr> </tbody> </table>			Register Name	Usage	zero	Constant	ra	Return address (functions/functions)	sp	Stack pointer	gp	Global pointer	tp	Thread pointer	ta	Temporary (preserved across calls)	tb..tss	Temporary (preserved across calls)	ta..tsz	Arguments for functions / return values	ta..tz	Arguments for functions	Floating-point registers		fa0..fa11	Saved temporary (preserved across calls)	fb0..fb11	Saved temporary (preserved across calls)	fa0..fa1	Arguments for functions / return values	fa2..fa2	Arguments for functions
Item	Calls (call)	de	Arguments	Result																																																																																									
a0 = integer																																																																																													
lb = float																																																																																													
lb = double																																																																																													
a0 = string add																																																																																													
Integer in a0																																																																																													
Float in a0																																																																																													
Double in a0																																																																																													
a0 = string addr																																																																																													
a1 = length																																																																																													
a2 = Address in a0																																																																																													
a0 = ASCII code				Char in a0																																																																																									
Register Name	Usage																																																																																												
zero	Constant																																																																																												
ra	Return address (functions/functions)																																																																																												
sp	Stack pointer																																																																																												
gp	Global pointer																																																																																												
tp	Thread pointer																																																																																												
ta	Temporary (preserved across calls)																																																																																												
tb..tss	Temporary (preserved across calls)																																																																																												
ta..tsz	Arguments for functions / return values																																																																																												
ta..tz	Arguments for functions																																																																																												
Floating-point registers																																																																																													
fa0..fa11	Saved temporary (preserved across calls)																																																																																												
fb0..fb11	Saved temporary (preserved across calls)																																																																																												
fa0..fa1	Arguments for functions / return values																																																																																												
fa2..fa2	Arguments for functions																																																																																												
Data transfer																																																																																													
Arithmetic (integer)																																																																																													
Logical (integer)																																																																																													
Arithmetic (floating point)																																																																																													
Comparisons (integer)																																																																																													
Comparisons (floating point)																																																																																													
Instructions (integer register)																																																																																													
Instructions (floating point register)																																																																																													
Instructions (hardware counter)																																																																																													
Memory access (floating point), (float)																																																																																													
Conversions (integer register)																																																																																													
Conversions (floating point register)																																																																																													
Conversions (hardware counter)																																																																																													
Conversions (double precision)																																																																																													
Floating point Classification																																																																																													

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

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

Configuration Info

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 registers, each with a name and a hex value. The 'Register name representation' section lists the same 32 registers by name. At the bottom, there are 'Clear' and 'Enter' buttons.

Register	Value
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 'CREATOR 3.2 RISC-V (RV32IMFD)' simulation interface. At the top, there are buttons for Architecture, # Assembly, Reset, Run, Stop, Examples, Calculator, Configuration, and Info. The Configuration button is highlighted in yellow. Below the buttons, there are 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. To the right is the 'Register name representation' section with buttons for Name, Alias, and All. Below these sections is a grid of 32x4 boxes, each containing a register name and its hex value. The registers are grouped by type: zero, ra, sp, gp; tp, t0, t1, t2; fp, s1, a0, a1; a2, a3, a4, a5; a6, a7, s2, s3; s4, s5, s6, s7; s8, s9, s10, s11; t3, t4, t5, t6.

Register	Value (Hex)
zero	00000000
ra	FFFFFFFFFF
sp	0FFFFFFC
gp	00000000
tp	00000000
t0	00000010
t1	00000000
t2	00000000
fp	00000000
s1	00000000
a0	00000000
a1	00000000
a2	00000000
a3	00000000
a4	00000000
a5	00000000
a6	00000000
a7	00000000
s2	00000000
s3	00000000
s4	00000000
s5	00000000
s6	00000000
s7	00000000
s8	00000000
s9	00000000
s10	00000000
s11	00000000
t3	00000000
t4	00000000
t5	00000000
t6	00000000



Registros enteros

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

Hex.	00000010
Binary	000000000000000000000000000010000
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.

At the top of the interface, there are tabs for Architecture, # Assembly, Reset, Run, Stop, Examples, Calculator, Configuration, and Info. On the right, there are tabs for INT Registers, FP Registers, Memory, Stats, and Energy (CLK Cycles). Below these tabs is a grid of register values:

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 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 of the interface are "Clear" and "Enter" buttons.



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 (highlighted), 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. The Hex tab is highlighted in yellow. The Register name representation section has tabs for Name, Alias, and All. The All tab is highlighted in yellow. The main area displays a grid of floating-point registers. Each register is labeled with its name (e.g., ft0 | f0, ft1 | f1, etc.) and its hex value (e.g., 0000000000000000). The first four columns of registers are highlighted in blue.

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



Registros en coma flotante

CREATOR 3.2 RISC-V (RV32IMFD)

didaCtic and geneRic assEmbly progrAmming simulaTOR

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

Break	Address	Label	User Instruction	Loaded Instructions
				ft4 f4 0000000041840000
				ft5 f5 0000000000000000
				ft6 f6 0000000000000000
				ft7 f7 0000000000000000
				ft8 f8 0000000000000000
				ft9 f9 0000000000000000
				ft10 f10 0000000000000000
				ft11 f11 0000000000000000
				ft12 f12 0000000000000000
				ft13 f13 0000000000000000
				ft14 f14 0000000000000000
				ft15 f15 0000000000000000
				ft16 f16 0000000000000000
				ft17 f17 0000000000000000
				ft18 f18 0000000000000000
				ft19 f19 0000000000000000
				ft20 f20 0000000000000000
				ft21 f21 0000000000000000
				ft22 f22 0000000000000000
				ft23 f23 0000000000000000
				ft24 f24 0000000000000000
				ft25 f25 0000000000000000
				ft26 f26 0000000000000000
				ft27 f27 0000000000000000
				ft28 f28 0000000000000000
				ft29 f29 0000000000000000
				ft30 f30 0000000000000000
				ft31 f31 0000000000000000

Register value representation

Signed Unsigned IEEE 754 Hex.

Register name representation

Name Alias All

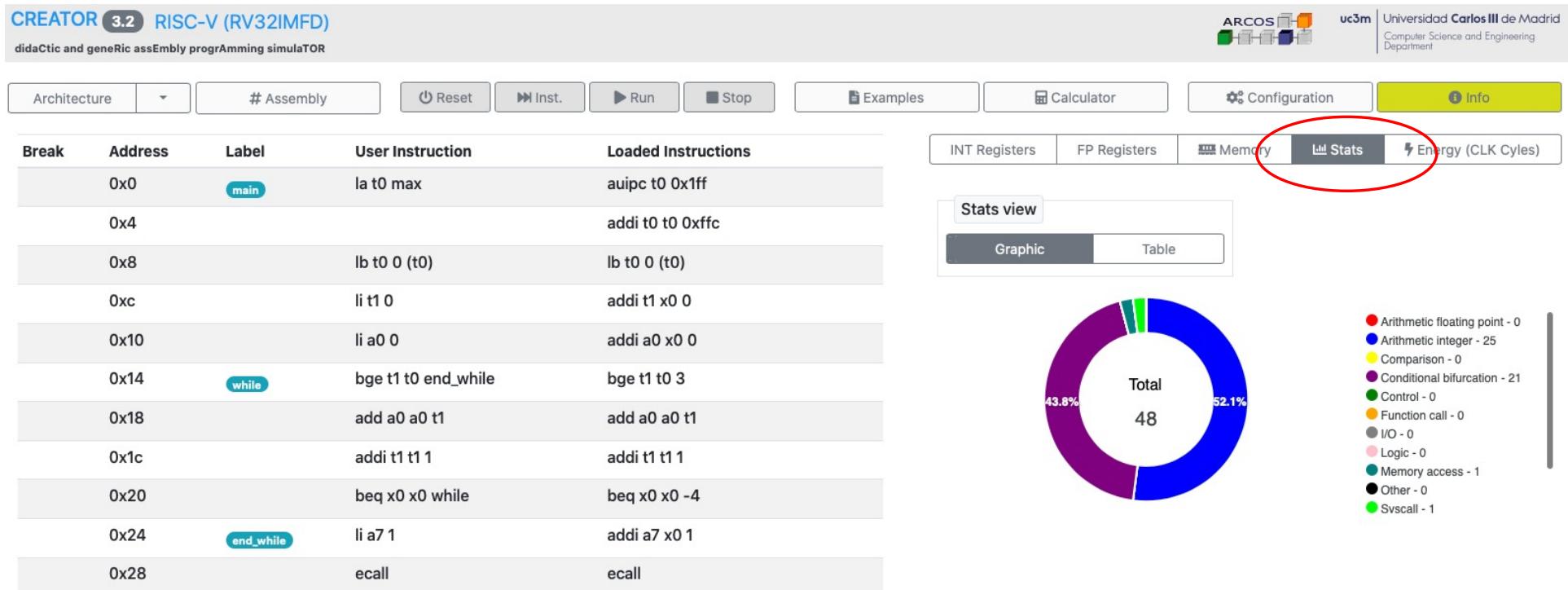


Contenido de la memoria

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 toolbars for Architecture, Assembly, Reset, Run, Stop, Examples, Calculator, Configuration, and Info. A navigation bar includes Break, Address, Label, User Instruction, and Loaded Instructions. The main area displays the 'Main memory segment' with tabs for Data, Text, and Stack. A table below shows columns for Address, Binary, and Value. At the bottom are icons for monitor, keyboard, and two buttons 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)' simulation interface. The top navigation bar includes tabs for Architecture, # Assembly, Reset, Run, Stop, Examples, Calculator, Configuration, and Info. The main workspace is divided into several sections: 'Break' and 'Address' columns, 'Label' and 'User Instruction' columns, and a 'Loaded instructions' section. Below these are tabs for INT Registers, FP Registers, Memory (selected), Stats, and Energy (CLK Cycles). A 'Main memory segment' section shows Data, Text, and Stack tabs. A table below has columns for Address, Binary, and Value. At the bottom 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' (selected), '# Assembly', 'Reset', 'Inst.', 'Run', 'Stop', 'Examples', 'Calculator', 'Configuration', and 'Info'. Below the navigation bar, there are columns for 'Break', 'Address', 'Label', 'User Instruction', and 'Loaded instructions'. On the right side, there are tabs for 'INT Registers', 'FP Registers', 'Memory' (which is selected), 'Stats', and 'Energy (CLK Cycles)'. A 'Main memory segment' section is visible, containing tabs for 'Data' (selected), 'Text', and 'Stack'. Below these sections is 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 above it. This input field is circled in red. Below the input field are buttons for 'Clear' and 'Enter'. The bottom left corner features the 'ARCOS' logo, and the bottom right corner has the number '31'.



Edición de programas en ensamblador

CREATOR 3.2 RISC-V (RV32IMFD)

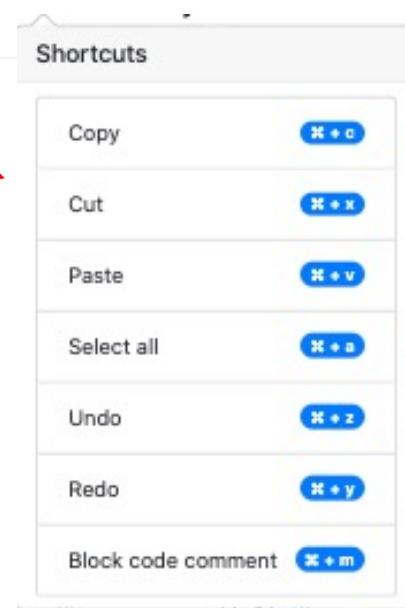
didaCtic and geneRic assEmbly program simulaTOR

Architecture Simulator ➔ Compile/Linked

Assembly:

```
1 #
2 # Creator (https://creatorsim.github.io/creator/)
3 #
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

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

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
0x14			mul t4 t0 t2	mul t4 t0 t2
0x18			div t5 t0 t2	div t5 t0 t2

INT Registers FP Registers Memory Stats Energy (CLK Cycles)

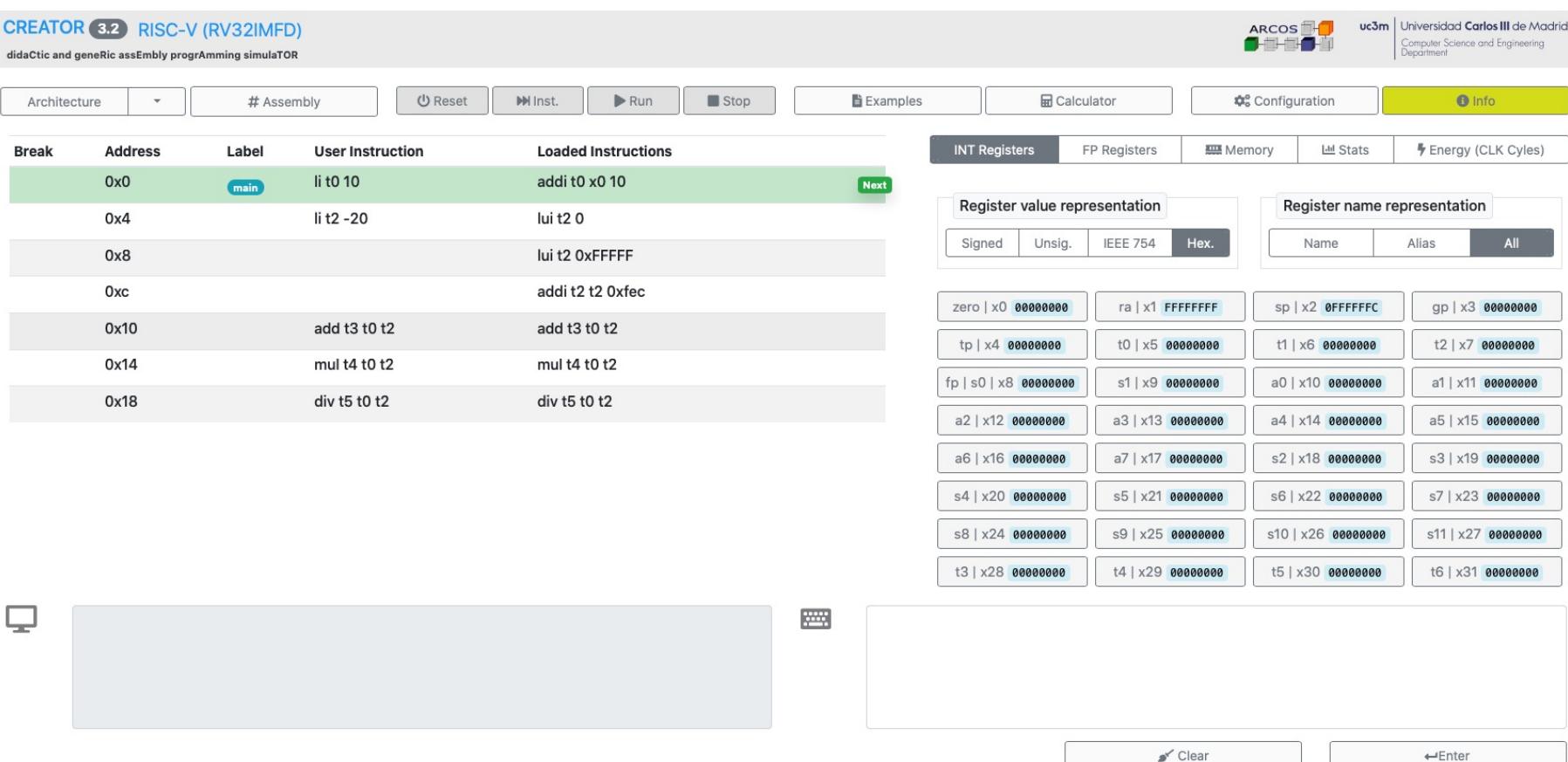
Register value representation Register name representation

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

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

➊ 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
17
18
19
```

Visualización de datos en memoria

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
17
18
19
```

Main memory segment			
Data	Text	Stack	
Address	Binary	Value	
0x00200000 - 0x00200003	48 6F 6C 61	H, o, l, a	
0x00200004 - 0x00200007	20 6D 75 6E	, m, u, n	
0x00200008 - 0x0020000B	64 6F 00 01	d, o, 0, 1	
0x0020000C - 0x0020000F	00 00 00 40	64	
0x00200010 - 0x00200013	61 00 00 00	97	
0x00200014 - 0x00200017	C1 48 00 00	-12.5	
0x00200018 - 0x0020001B	C0 28 00 00	-12.5	
0x0020001C - 0x0020001F	00 00 00 00		
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	

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
17
18
19
```

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	v2 5	
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		

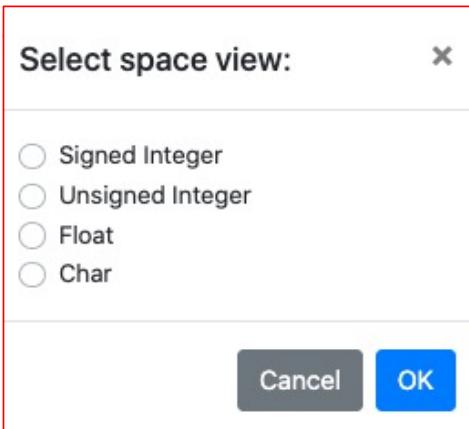


Visualización de datos en memoria

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
17
18
19
```

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
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

The screenshot shows the RISC-V CREATOR 3.2 interface. On the left, there's an assembly code editor with the following content:

```
1 .data
2     cadena:    .string "Hola Mundo"
3     N1:        .word 0
4     N2:        .word 0
5     N3:        .word 0
6     N4:        .zero 0
7
8
```

A modal window titled "Assembly Code Error" is open, highlighting a portion of the code in red. The highlighted area contains:

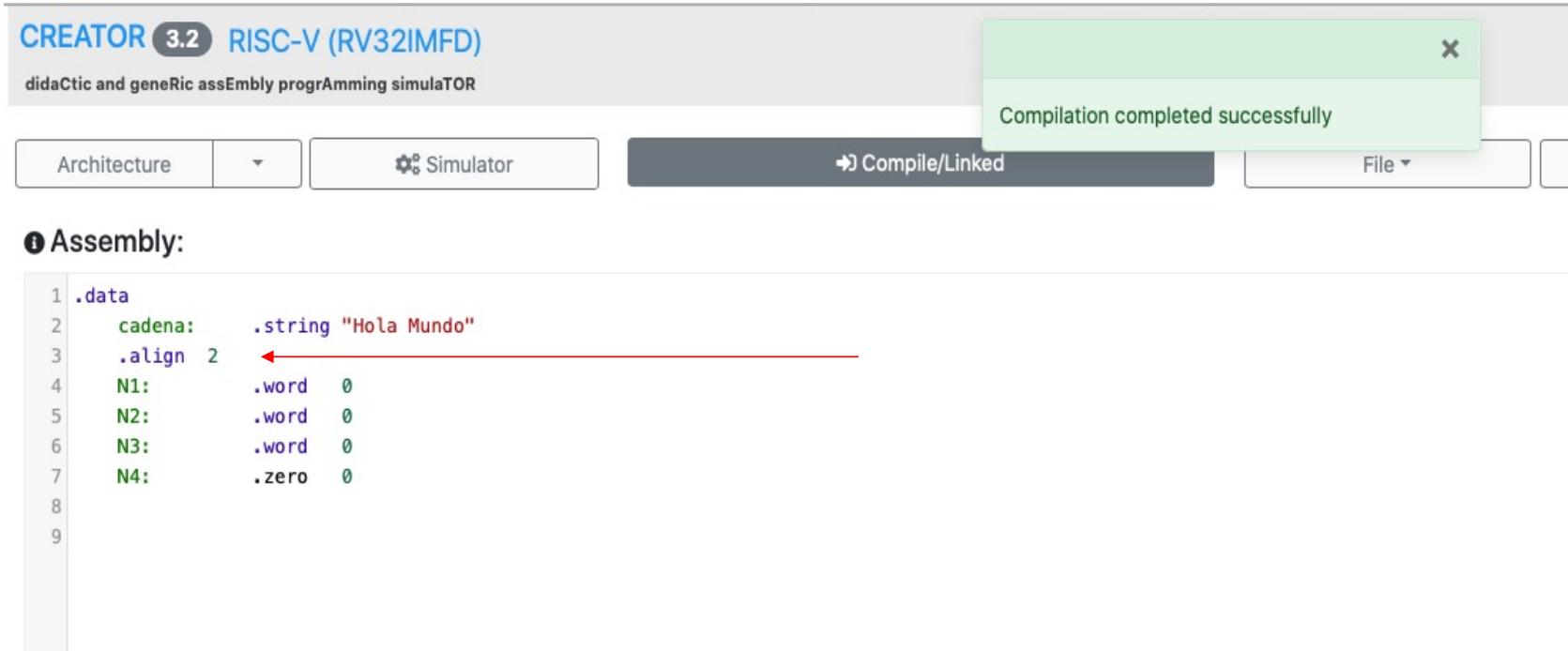
```
        ...
2         cadena: .string "Hola Mundo"
*         3         N1: .word 0
        4         N2: .word 0
        ...

```

The error message in the modal states: "The data must be aligned".

Segmento de datos corregido

ejemplo



The screenshot shows the CREATOR 3.2 RISC-V (RV32IMFD) software 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". In the assembly editor, there is a red arrow pointing to the ".align 2" instruction at line 3. The assembly code is as follows:

```
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   0
8
9
```

Ejemplo. Ejecutar el siguiente programa

ejemplo

CREATOR 3.2 RISC-V (RV32IMFD)

didactic and generic assembly programming simulator

Architecture ▾ Simulator ➔ Compile/Linked

❶ Assembly:

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

Ejemplo: antes de la ejecución

CREATOR 3.2 RISC-V (RV32IMFD)

didaCtic and geneRic assEmby progrAmming simulaTOR

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

Break	Address	Label	User Instruction	Loaded Instructions
0x0		main	li t0 1	addi t0 x0 1
0x4			sw t0 N1 (zero)	sw t0 0x200000 (zero)
0x8			li t0 2	addi t0 x0 2
0xc			sw t0 N2 (zero)	sw t0 0x200004 (zero)
0x10			li t0 3	addi t0 x0 3
0x14			sw t0 N3 (zero)	sw t0 0x200008 (zero)
0x18			li t0 4	addi t0 x0 4
0x1c			sw t0 N4 (zero)	sw t0 0x20000c (zero)

INT Registers	FP Registers	Memory	Stats	Energy (CLK Cycles)
Main memory segment				
Data	Text	Stack		
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

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

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

Break	Address	Label	User Instruction	Loaded Instructions
0x0		main	li t0 1	addi t0 x0 1
0x4			sw t0 N1 (zero)	sw t0 0x200000 (zero)
0x8			li t0 2	addi t0 x0 2
0xc			sw t0 N2 (zero)	sw t0 0x200004 (zero)
0x10			li t0 3	addi t0 x0 3
0x14			sw t0 N3 (zero)	sw t0 0x200008 (zero)
0x18			li t0 4	addi t0 x0 4
0x1c			sw t0 N4 (zero)	sw t0 0x20000c (zero)

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

INT Registers FP Registers Memory Stats Energy (CLK Cycles)

Main memory segment

Data	Text	Stack
0x00200000 - 0x00200003	N1 00 00 00 01	1
0x00200004 - 0x00200007	N2 00 00 00 02	2
0x00200008 - 0x0020000B	N3 00 00 00 03	3
0x0020000C - 0x0020000F	N4 00 00 00 04	4

A red oval highlights the value 4 at address 0x0020000C.



Visualización del segmento de texto

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	
0x0		main	li t0 1		
0x4			sw t0 N1 (zero)		
0x8			li t0 2		
0xc			sw t0 N2 (zero)		
0x10			li t0 3		
0x14			sw t0 N3 (zero)		
0x18			li t0 4		
0x1c			sw t0 N4 (zero)		

User Instruction (Left): The assembly code written by the user, showing instructions like li t0 1, sw t0 N1 (zero), etc.

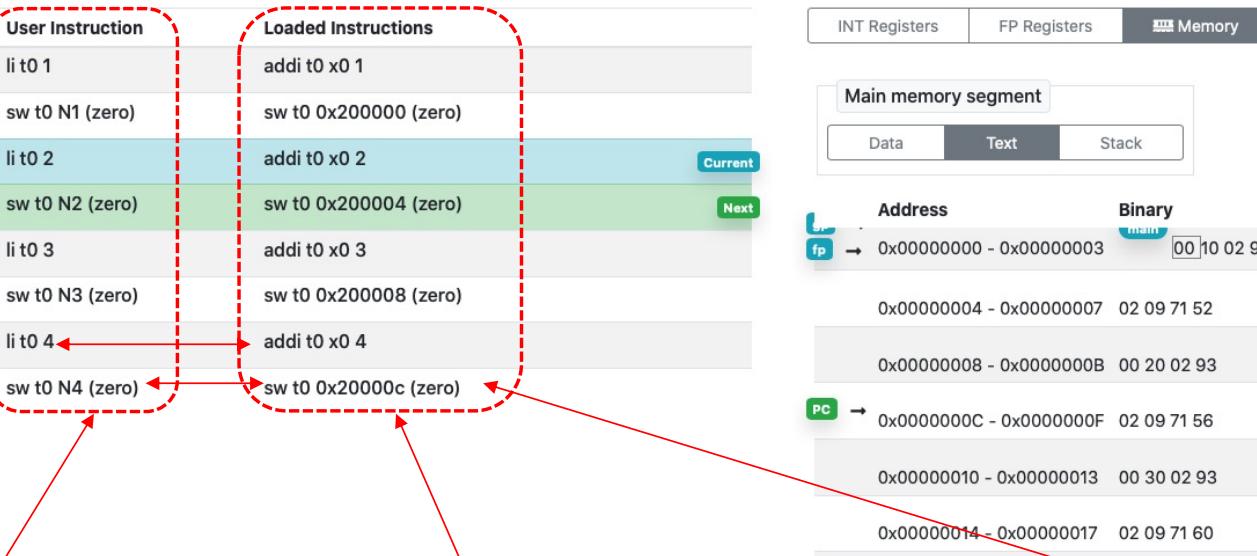
Loaded Instructions (Right): The actual machine instructions loaded into memory, showing addi t0 x0 1, sw t0 0x200000 (zero), etc.

Main memory segment (Top right): A table showing the memory layout. It has columns for Address, Binary, and Value. The Value column shows the machine code for each instruction. Red arrows point from the User Instruction table to the corresponding entries in the Main memory segment table.

fp	Address	Binary	Value
→ 0x00000000 - 0x00000003	00 10 02 93	addi t0 x0 1	
0x00000004 - 0x00000007	02 09 71 52	0	
0x00000008 - 0x0000000B	00 20 02 93		
PC	Address	Binary	Value
→ 0x0000000C - 0x0000000F	02 09 71 56	0	
0x00000010 - 0x00000013	00 30 02 93		
0x00000014 - 0x00000017	02 09 71 60	0	
0x00000018 - 0x0000001B	00 40 02 93		
0x0000001C - 0x0000001F	02 09 71 64	sw t0 0x20000c (zero)	
0x00000020 - 0x00000023	00 50 20 23		

Programa escrito (inst. y pseudoinst.)

Programa en memoria (instrucciones máquina)



Visualización del segmento de texto

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

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		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		Next			
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	f1 00 10 05 13	addi a0 x0 1
0x00000004 - 0x00000007	00 00 80 67	jalr x0 0 (ra)
0x00000008 - 0x0000000B	f2 00 20 05 13	addi a0 x0 2
0x0000000C - 0x0000000F	00 00 80 67	jalr x0 0 (ra)
0x00000010 - 0x00000013	f3 00 30 05 13	addi a0 x0 3
0x00000014 - 0x00000017	00 00 80 67	jalr x0 0 (ra)
0x00000018 - 0x0000001B	main 00 00 00 EF	jal ra 0x0



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 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 00200000	a1 x11 00000000
a2 x12 0000000	a3 x13 0000000	a4 x14 00000000	a5 x15 00000000
a6 x16 0000000	a7 x17 00000005	s2 x18 00000000	s3 x19 00000000
s4 x20 0000000	s5 x21 00000000	s6 x22 00000000	s7 x23 00000000
s8 x24 0000000	s9 x25 00000000	s10 x26 00000000	s11 x27 00000000
t3 x28 0000000	t4 x29 00000000	t5 x30 00000000	t6 x31 00000000

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

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 0xffffc					
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	
Address	Binary	Value	
0x00000004 - 0x00000007	FF C2 82 83	addi t0 t0 0xffffc	
0x00000008 - 0x0000000B	00 1F F3 17	auipc t1 0x1ff	
0x0000000C - 0x0000000F	FF F3 03 13	addi t1 t1 0xffff	
PC → 0x00000010 - 0x00000013	loop 00 02 C3 83	lbu t2 0 (t0)	①
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		



Llamadas a funciones

- ▶ Convenio simplificado
 - ▶ La pila no necesita estar alineada a 8 bytes
- ▶ Alerta si se viola 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	
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 de llamadas a funciones anidadas

ejemplo

- ▶ Comprobar el crecimiento de la pila

Main memory segment			
	Data	Text	Stack
Address	Binary	Value	
sp → 0xFFFFFFF0 - 0xFFFFFFF3	00 00 00 28	40	
0xFFFFFFF4 - 0xFFFFFFF7	00 00 00 94	148	
0xFFFFFFF8 - 0xFFFFFFFB	FF FF FF FF	4294967295	
0xFFFFFFFc - 0xFFFFFFFF	00 00 00 00	00	

Stack memory keys:

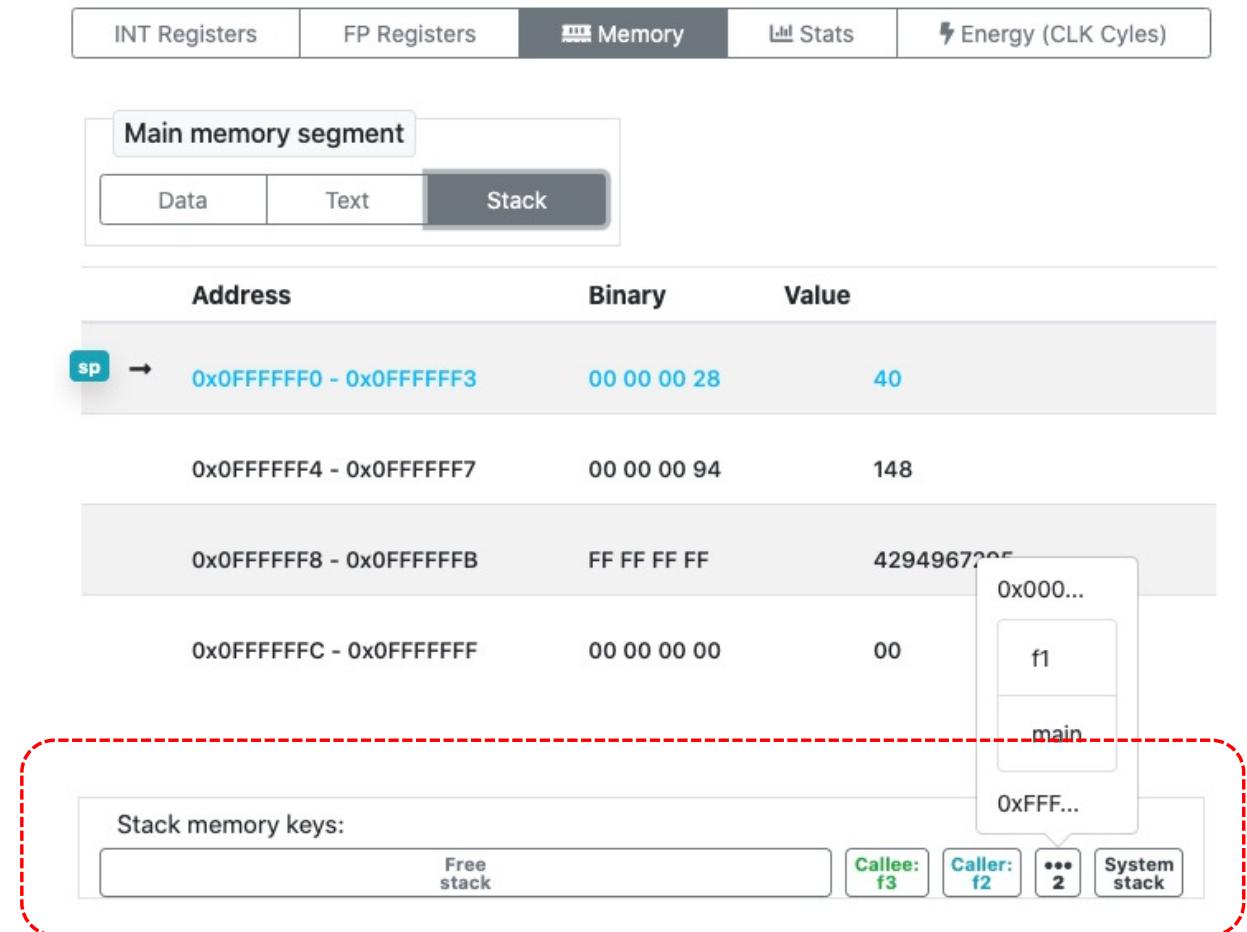
- Free stack
- Callee: f2
- Caller: f1
- ... 1
- System stack

Ejemplo de llamadas a funciones anidadas

ejemplo

- ▶ Comprobar el crecimiento de la pila

No funciona bien ?



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

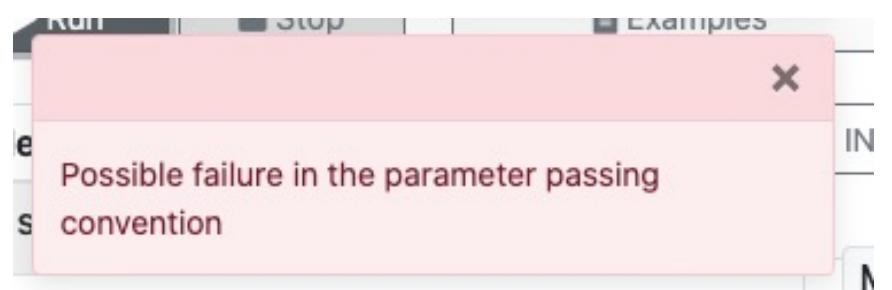
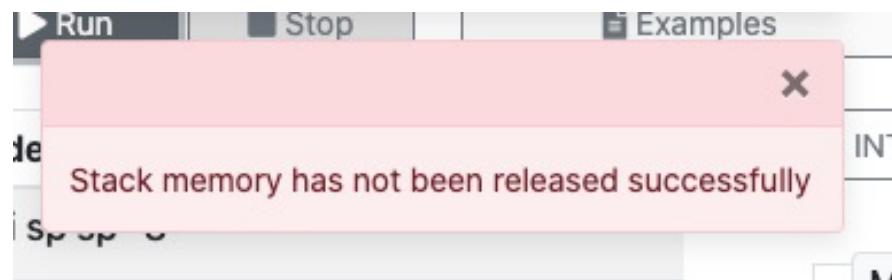
```
.data
    .text
max:      addi    sp, sp -8
          sw     s0, 0(sp)
          mv     s0, a0
          mv     s1, a1
          bge   s0, s1, bigger
          mv     a0, s1
          jr     ra
bigger:   mv     a0, s0
          jr     ra

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
.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
```

Creación de librerías

ejemplo

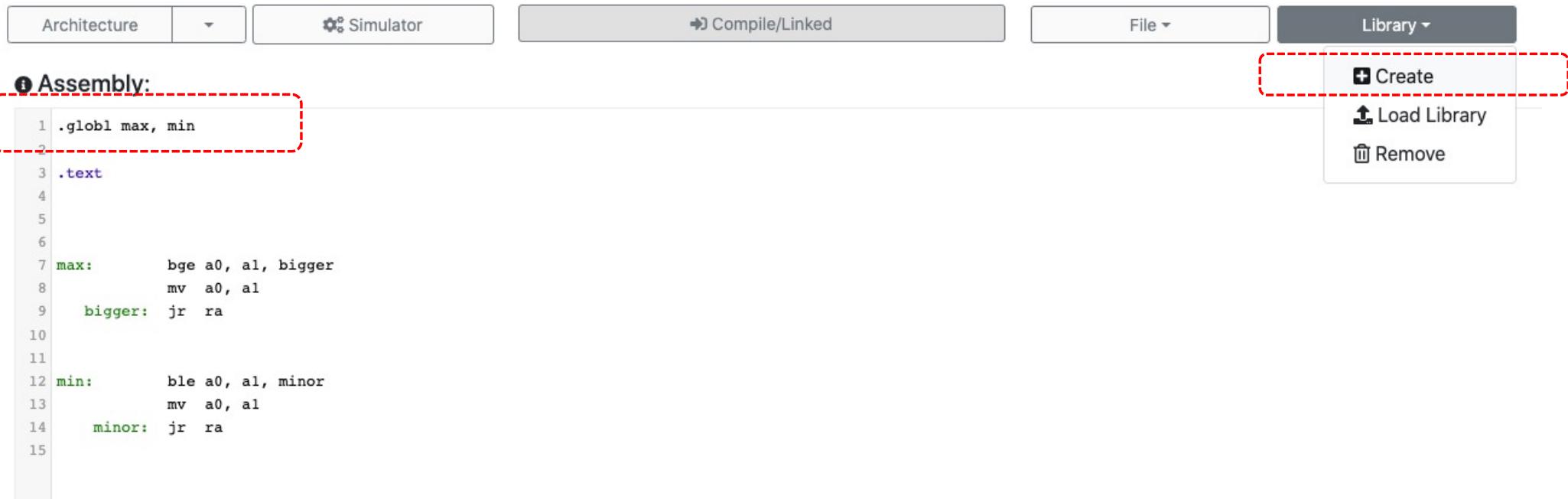


➊ Assembly:

```
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 assembly editor interface. The top navigation bar includes tabs for 'Architecture' (with a dropdown), 'Simulator' (with a CPU icon), 'Compile/Linked' (selected, with a right-pointing arrow icon), 'File' (with a dropdown), and 'Library' (selected, with a dropdown). A red dashed box highlights the 'Assembly' tab. Below it, the assembly code for 'max' and 'min' functions is displayed:

```
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
```

A red dashed box also highlights the 'Library' dropdown menu on the right, which contains three options: '+ Create', 'Load Library', and 'Remove'.

Uso de librerías

ejemplo

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

- Toolbar: Architecture, Simulator, Compile/Linked, File, Library.
- Assembly View: Shows assembly code for a program named "main". The code includes several `jal ra, max` and `jal ra, min` instructions, which are highlighted with red arrows pointing to the "Load Library" option in the context menu.
- Context Menu (Library): A red dashed box highlights the "Load Library" option under the "Library" dropdown.

Assembly:

```
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
```

Llamadas

Librería cargada

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

- Top navigation bar with tabs: Architecture, Simulator, Compile/Linked, File, Library, Configuration, and Info (highlighted).
- Assembly code editor labeled "Assembly":

```
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
```
- Panel on the right labeled "Library tags:" containing two entries:

max
min



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

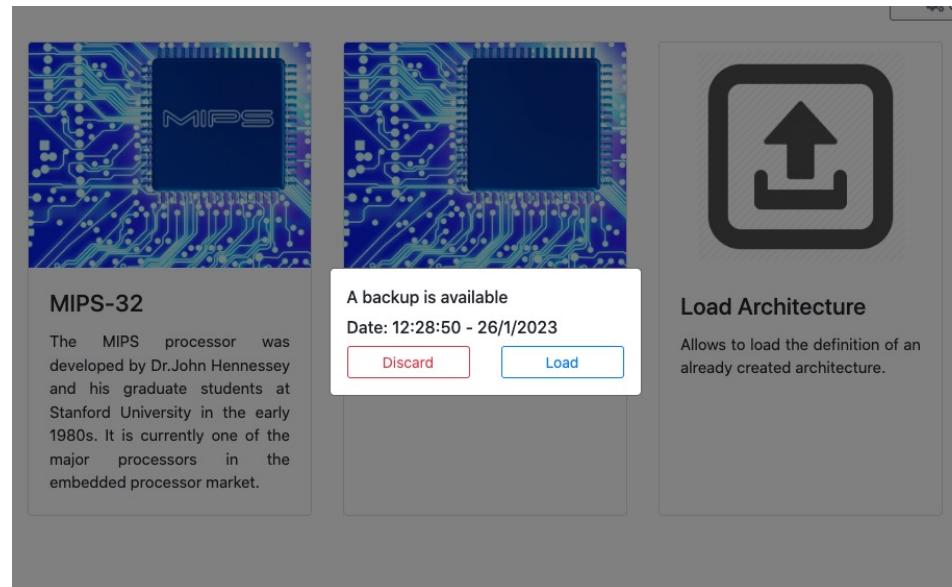
Data Text Stack

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



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 paso de parámetros a funciones
- ▶ 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
```

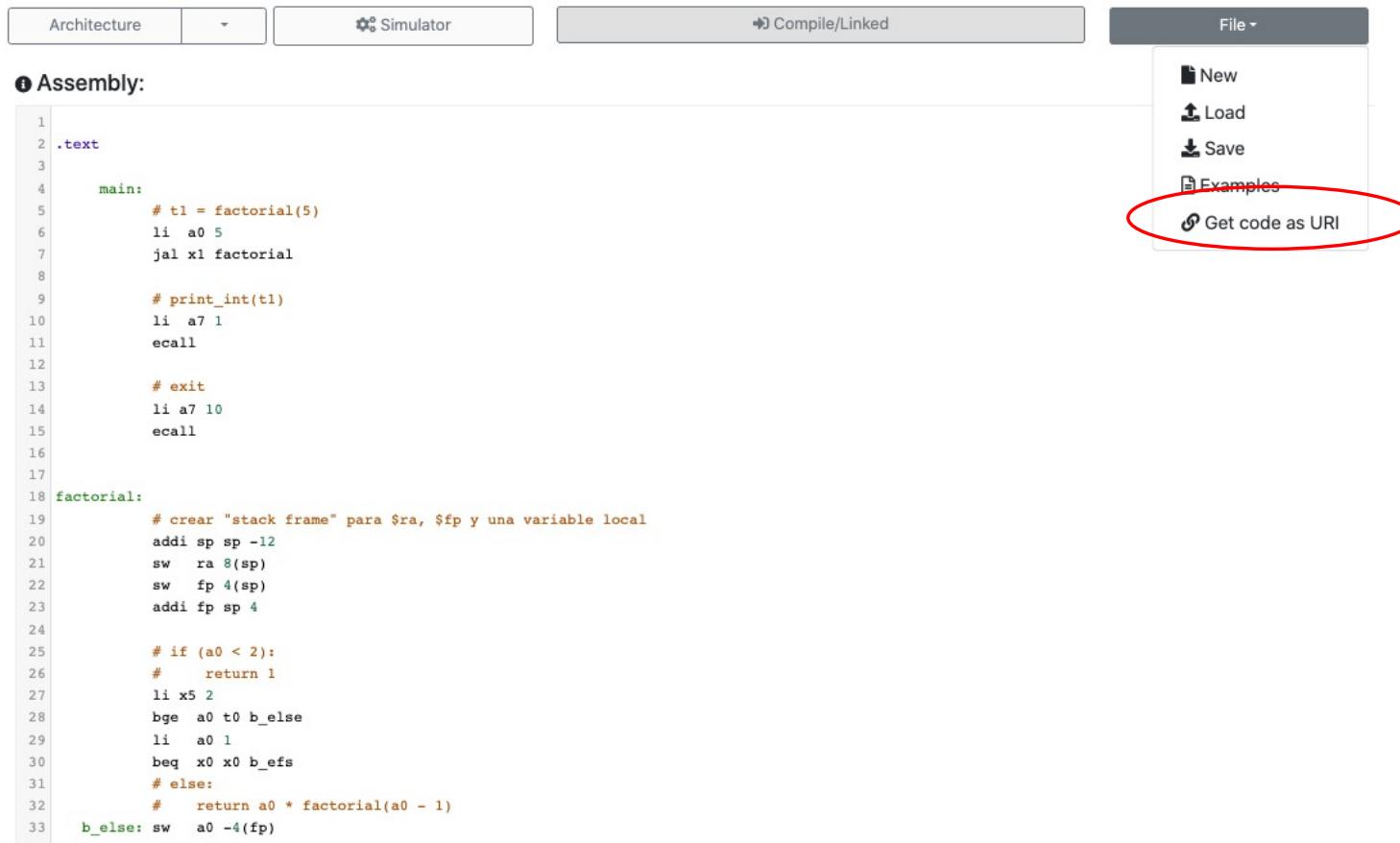
Ejemplo de salida incorrecta

- ▶ ./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



Architecture ▾ Simulator ▾ Compile/Linked ▾ File ▾

Assembly:

```
1 .text
2
3
4     main:
5         # t1 = factorial(5)
6         li a0 5
7         jal xl factorial
8
9         # print_int(t1)
10        li a7 1
11        ecall
12
13        # exit
14        li a7 10
15        ecall
16
17
18 factorial:
19     # crear "stack frame" para $ra, $fp y una variable local
20     addi sp sp -12
21     sw ra 8(sp)
22     sw fp 4(sp)
23     addi fp sp 4
24
25     # if (a0 < 2):
26     #     return 1
27     li x5 2
28     bge a0 t0 b_else
29     li a0 1
30     beq x0 x0 b_efs
31     # else:
32     #     return a0 * factorial(a0 - 1)
33     b_else: sw a0 -4(fp)
```



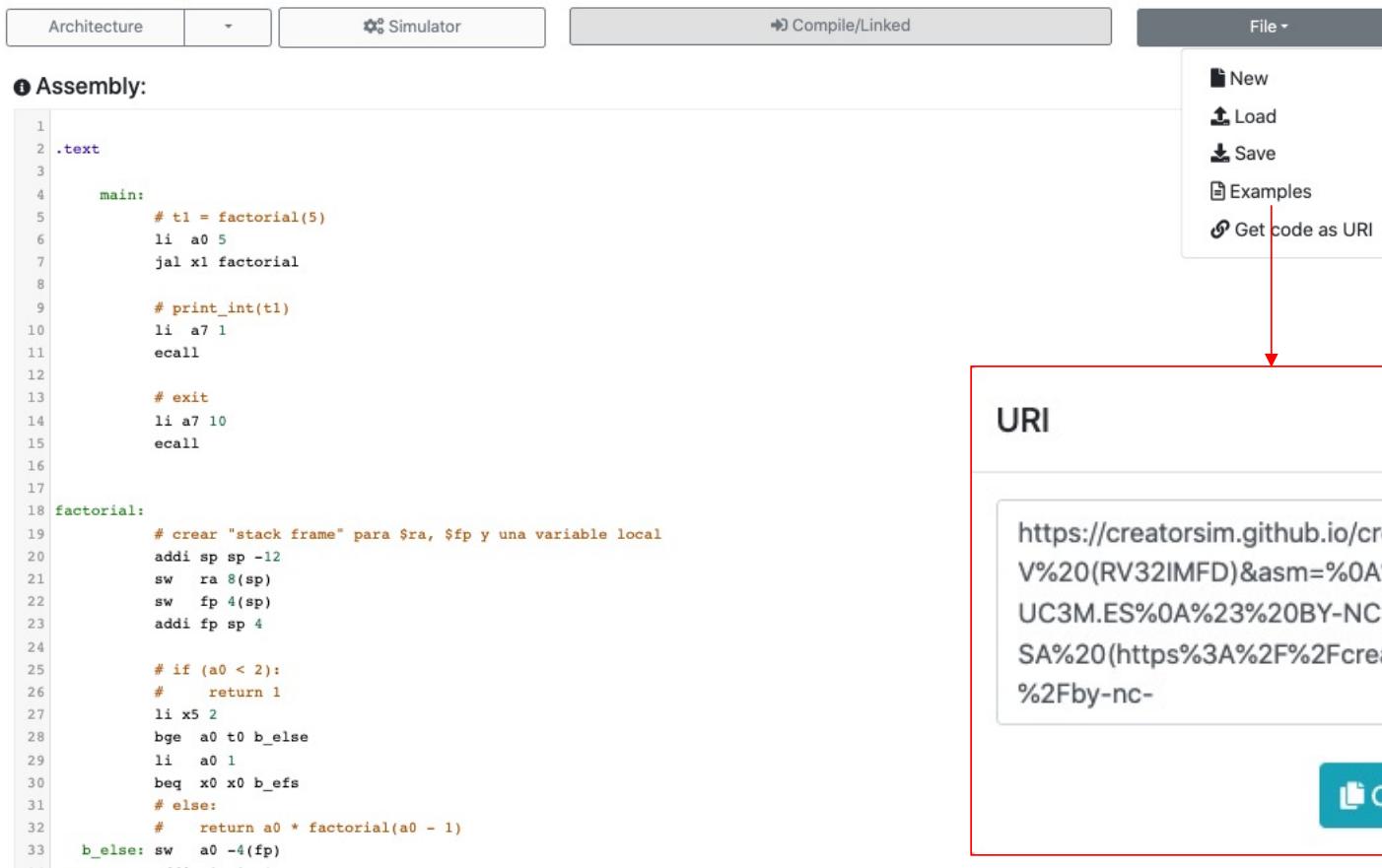
Ayuda a la creación de materiales docentes

The screenshot shows the ARCOS simulator interface. At the top, there are tabs for "Architecture" (selected), "Simulator", "Compile/Linked", and "File". A dropdown menu under "File" includes options: New, Load, Save, Examples, and Get code as URI. A red arrow points from this menu to a modal window titled "URI". The modal contains the URL: [https://creatorsim.github.io/creator/?architecture=RISC-V%20\(RV32IMFD\)&asm=%0A%23%0A%23%20ARCOS.INF. UC3M.ES%0A%23%20BY-NC-SA%20\(https%3A%2F%2Fcreativecommons.org%2Fllicenses%2Fby-nc-](https://creatorsim.github.io/creator/?architecture=RISC-V%20(RV32IMFD)&asm=%0A%23%0A%23%20ARCOS.INF. UC3M.ES%0A%23%20BY-NC-SA%20(https%3A%2F%2Fcreativecommons.org%2Fllicenses%2Fby-nc-). A "Copy" button is at the bottom of the modal.

```
1 .text
2
3
4     main:
5         # t1 = factorial(5)
6         li a0 5
7         jal x1 factorial
8
9         # print_int(t1)
10        li a7 1
11        ecall
12
13        # exit
14        li a7 10
15        ecall
16
17
18 factorial:
19     # crear "stack frame" para $ra, $fp y una variable local
20     addi sp sp -12
21     sw ra 8(sp)
22     sw fp 4(sp)
23     addi fp sp 4
24
25     # if (a0 < 2):
26     #     return 1
27     li x5 2
28     bge a0 t0 b_else
29     li a0 1
30     beq x0 x0 b_efs
31     # else:
32     #     return a0 * factorial(a0 - 1)
33     b_else: sw a0 -4(fp)
```



Ayuda a la creación de materiales docentes



The screenshot shows the ARCOS simulator interface. At the top, there are tabs for 'Architecture' (with a dropdown), 'Simulator' (with a gear icon), 'Compile/Linked' (with a double arrow icon), and 'File' (with a dropdown). The 'File' dropdown is open, showing options: 'New', 'Load', 'Save', 'Examples', and 'Get code as URI'. A red arrow points from the 'Get code as URI' option down to a modal window titled 'URI'. The modal contains a URL: [https://creatorsim.github.io/creator/?architecture=RISC-V%20\(RV32IMFD\)&asm=%0A%23%0A%23%20ARCOS.INF. UC3M.ES%0A%23%20BY-NC-SA%20\(https%3A%2F%2Fcreativecommons.org%2Fllicenses%2Fby-nc-](https://creatorsim.github.io/creator/?architecture=RISC-V%20(RV32IMFD)&asm=%0A%23%0A%23%20ARCOS.INF. UC3M.ES%0A%23%20BY-NC-SA%20(https%3A%2F%2Fcreativecommons.org%2Fllicenses%2Fby-nc-). A red box highlights this URL. A red arrow also points from the 'Get code as URI' option in the menu to the URL in the modal. The modal has a 'Copy' button at the bottom right.

ejemplo

```
1 .text
2
3
4     main:
5         # t1 = factorial(5)
6         li a0 5
7         jal x1 factorial
8
9         # print_int(t1)
10        li a7 1
11        ecall
12
13        # exit
14        li a7 10
15        ecall
16
17
18 factorial:
19     # crear "stack frame" para $ra, $fp y una variable local
20     addi sp sp -12
21     sw ra 8(sp)
22     sw fp 4(sp)
23     addi fp sp 4
24
25     # if (a0 < 2):
26     #     return 1
27     li x5 2
28     bge a0 t0 b_else
29     li a0 1
30     beq x0 x0 b_efs
31     # else:
32     #     return a0 * factorial(a0 - 1)
33     b_else: sw a0 -4(fp)
```



Añadir nuevas instrucciones

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

Architecture ▾ Simulator ➔ Compile/Linked File ▾

● Assembly:

```
1
```

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>	rs2_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:

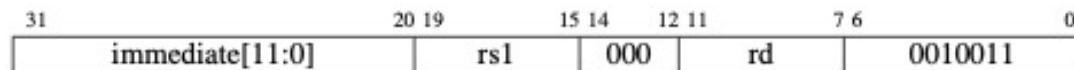
Enter Subroutine Exit Subroutine

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

Cancel [Save](#)



Ejemplo de definición de una instrucción (addi)



Edit addi

código de operación

	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 ✓

« < Principal Fields Syntax Definition Help > »

Cancel Save

Ejemplo de definición de una instrucción (addi)

Edit addi ×

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 sub-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: ✓

Number of Words: ✓

Number of fields: ✓

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 blitz rs1 offset ×

Pseudoinstruction Definition:

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

Pseudoinstruction Definition

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

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

Creación de una nueva pseudoinstrucción

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

Architecture general information:

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

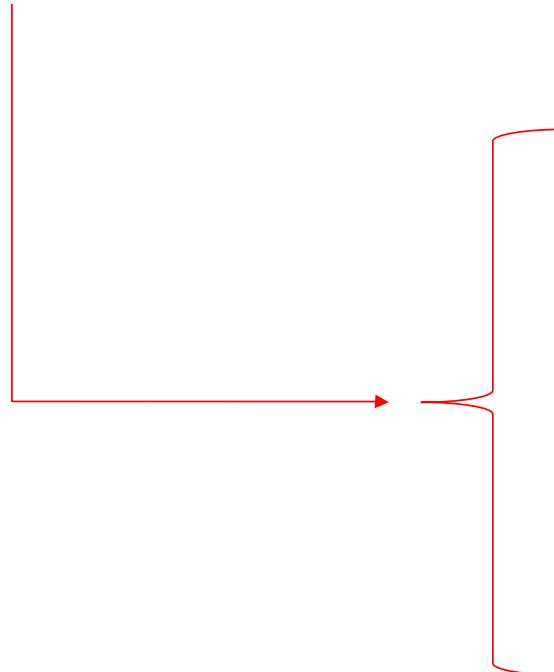
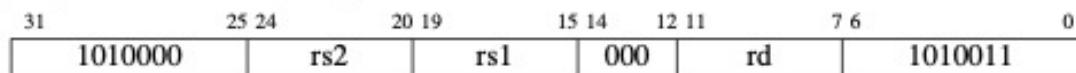
$$f[rd] = f[rs1] \times f[rs2] + f[rs3]$$

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	

Nueva instrucción: fmadd.s



New Instruction

	Name:	Type	Break	Start Bit	End Bit
Field 0	fmadd.s	co		6 ✓	0 ✓ 1000011 ✓
Field 1	rd ✓	SFP-Reg ✓ ↴		11 ✓	7 ✓
Field 2	rs1 ✓	SFP-Reg ✓ ↴		19 ✓	15 ✓
Field 3	rs2 ✓	SFP-Reg ✓ ↴		24 ✓	20 ✓
Field 4	rs3 ✓	SFP-Reg ✓ ↴		31 ✓	30 ✓

« < Principal Fields Syntax Definition Help > »

Cancel Save



Nueva instrucción: fmadd.s

Edit fmadd.s X

Assembly Definition:

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

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

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

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)

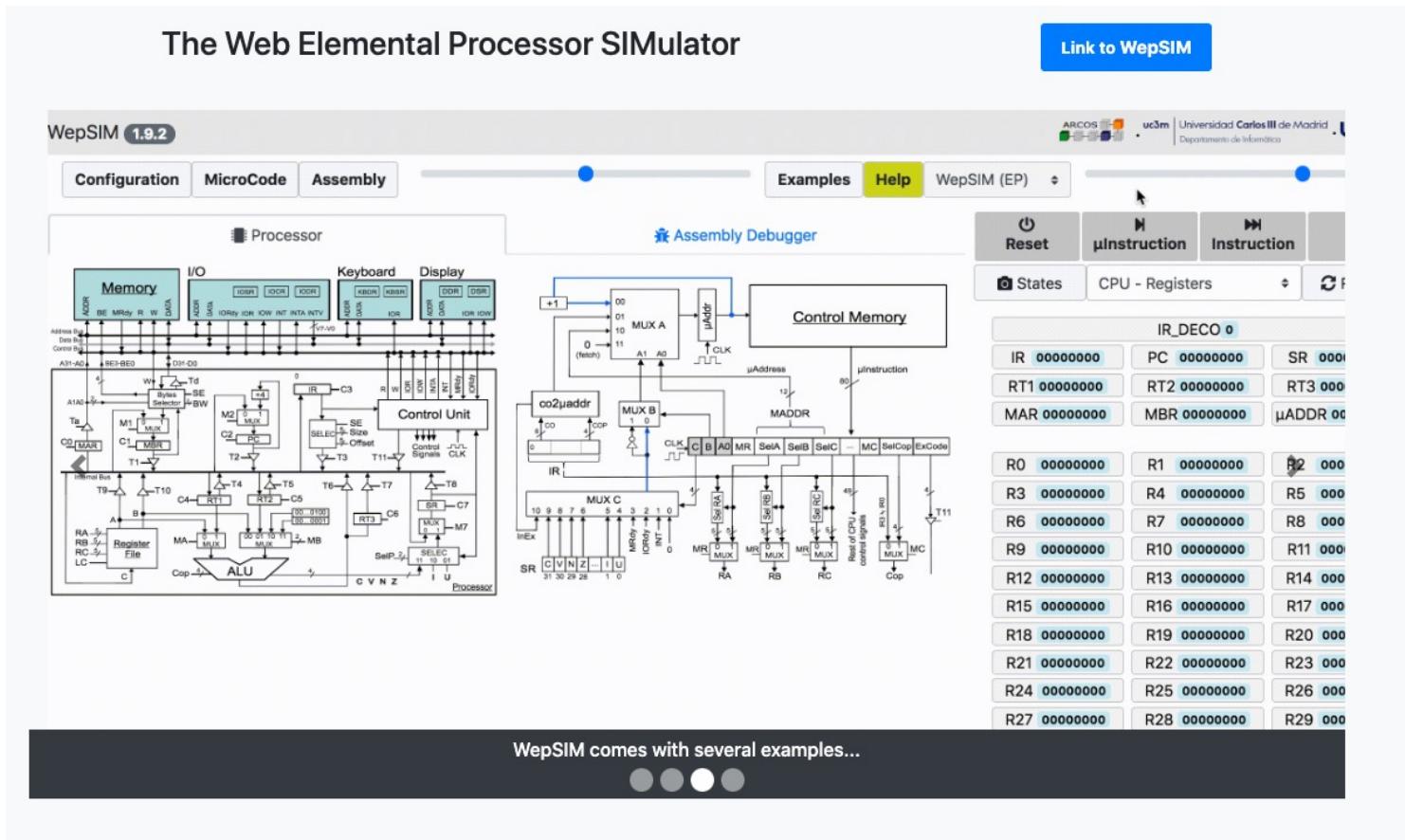
- ▶ Definición:

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

Extensiones futuras

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

Otro simulador: WepSim



WepSim

