



Jornadas Sarteco

17 – 19 Junio 2026, Madrid



CREATOR: entorno de desarrollo integrado para la docencia y la investigación en arquitecturas RISC-V

Félix García Carballeira, Diego Camarmas Alonso, Alejandro Calderón Mateos

uc3m | Universidad **Carlos III** de Madrid



Grupo ARCOS

Bloque 1

- ▶ ¿Qué es CREATOR?
 - ▶ El origen de CREATOR
 - ▶ Características de CREATOR
 - ▶ ¿Dónde se usa CREATOR?
 - ▶ Extensiones futuras
- ▶ ¿Cómo utilizar CREATOR? Visión del Estudiante
 - ▶ Edición y compilación de programas
 - ▶ Ejecución y depuración de programas
 - ▶ Convenio de paso de parámetros y uso de pila (Sentinel)
 - ▶ Bibliotecas de funciones
- ▶ ¿Cómo utilizar CREATOR? Visión del Profesor
 - ▶ Soporte para la creación de material didáctico
 - ▶ Soporte para la corrección de prácticas
 - ▶ Soporte para la creación y edición de arquitecturas

Bloque 2

- ▶ Dispositivos y soporte de interrupciones
- ▶ Integración de CREATOR con hardware real
 - ▶ Microcontroladores
 - ▶ Placas SBC
 - ▶ Servicio de laboratorio remoto
 - ▶ Integración con Arduino (CREATino)

Bloque 1

- ▶ **¿Qué es CREATOR?**
 - ▶ El origen de CREATOR
 - ▶ Características de CREATOR
 - ▶ ¿Dónde se usa CREATOR?
 - ▶ Extensiones futuras
- ▶ **¿Cómo utilizar CREATOR? Visión del Estudiante**
 - ▶ Edición y compilación de programas
 - ▶ Ejecución y depuración de programas
 - ▶ Convenio de paso de parámetros y uso de pila (Sentinel)
 - ▶ Bibliotecas de funciones
- ▶ **¿Cómo utilizar CREATOR? Visión del Profesor**
 - ▶ Soporte para la creación de material didáctico
 - ▶ Soporte para la corrección de prácticas
 - ▶ Soporte para la creación y edición de arquitecturas

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, tabletas 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

The screenshot displays the CREATOR web interface, which is an educational integrated development environment for assembly programming. The interface is divided into several sections:

- Header:** Includes the CREATOR logo, navigation links (About, Publications, Evolution, Authors, Contributors, Community), and a GitHub button.
- Left Panel:** Features the CREATOR logo, a description: "An educational integrated development environment for assembly programming, developed by the ARCOS research group at UC3M.", and four buttons: "Try CREATOR", "Documentation", "Usage Statistics", and "Try CREATOR Beta".
- Main Panel:** Shows a code editor with assembly instructions. The instructions are organized into sections: "main", "process_user_input", "copy_loop", and "malicious". Each instruction includes an address, user instruction, and loaded instruction.
- Right Panel:** Displays the "Register Inspector" window, which shows the state of various registers. It is divided into three sections: "Control registers", "Integer registers", and "Floating point registers". Each register is shown with its name and a hexadecimal value.

Address	User Instruction	Loaded Instructions
0x0	main	la a0, attacker_payload auipc a0, 512
0x4		addi a0, a0, 0
0x8	jal ra, process_user_input	jal ra, 12
0xC	li a7, 10	addi a7, x0, 10
0x10	ecall	ecall
0x14	process_user_input	addi sp, sp, -32
0x18	sw ra, 28(sp)	sw ra, 28(sp)
0x1C	mv t0, sp	addi t0, sp, 0
0x20	mv t1, a0	addi t1, a0, 0
0x24	li t2, 8	addi t2, x0, 8
0x28	copy_loop	beqz t2, copy_done
0x2C	lw t3, 0(t1)	lw t3, 0(t1)
0x30	sw t3, 0(t0)	sw t3, 0(t0)
0x34	addi t1, t1, 4	addi t1, t1, 4
0x38	addi t0, t0, 4	addi t0, t0, 4
0x3C	addi t2, t2, -1	addi t2, t2, -1
0x40	j copy_loop	beq zero, zero, -24
0x44	copy_done	lw ra, 28(sp)
0x48	addi sp, sp, 32	addi sp, sp, 32
0x4C	jr ra	jalr x0, 0(ra)
0x50	malicious	li t5, 0xCOFFEE01
0x54		lui t5, 790527
		addi t5, t5, -511

Control registers				
pc	mepc	mcause	mtvec	mstatus
00000008	00000000	00000000	00000000	00000000
mip	mie	mscratch	mtime	mtimecmp
00000000	00000000	00000000	00000002	00000000

Integer registers				
x0 zero	x1 ra	x2 sp	x3 gp	x4 tp
00000000	FFFFFFFF	0FFFFFFC	00000000	00000000
x5 t0	x6 t1	x7 t2	x8 fp s0	x9 s1
00000000	00000000	00000000	00000000	00000000
x10 a0	x11 a1	x12 a2	x13 a3	x14 a4
00200000	00000000	00000000	00000000	00000000
x15 a5	x16 a6	x17 a7	x18 s2	x19 s3
00000000	00000000	00000000	00000000	00000000
x20 s4	x21 s5	x22 s6	x23 s7	x24 r
00000000	00000000	00000000	00000000	00000000
x25 s9	x26 s10	x27 s11	x28 t3	x29 t4
00000000	00000000	00000000	00000000	00000000
x30 t5	x31 t6			
00000000	00000000			

Floating point registers				
f0 ft0	f1 ft1	f2 ft2	f3 ft3	f4 ft4
0000000000000000	0000000000000000	0000000000000000	0000000000000000	0000000000000000
f5 ft5	f6 ft6	f7 ft7	f8 fs0	f9 fs1
0000000000000000	0000000000000000	0000000000000000	0000000000000000	0000000000000000
f10 fa0	f11 fa1	f12 fa2	f13 fa3	f14 fa4
0000000000000000	0000000000000000	0000000000000000	0000000000000000	0000000000000000
f15 fa5	f16 fa6	f17 fa7	f18 fs2	f19 fs3
0000000000000000	0000000000000000	0000000000000000	0000000000000000	0000000000000000
f20 fs4	f21 fs5	f22 fs6	f23 fs7	f24 fs8

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

CREATOR

The screenshot displays the CREATORSIM web interface. The browser address bar shows `creatorsim.github.io/creator/`. The interface includes a navigation menu with 'View', 'Tools', and 'Help', and control buttons for 'Reset', 'Step', 'Run', and 'Stop'. A 'Sentinel' indicator is visible. The main content area is divided into sections for 'Registers', 'Memory', 'Terminal', and 'Statistics'. The 'Registers' section is expanded, showing three categories: Control registers, Integer registers, and Floating point registers. Each register is displayed with its name and a hexadecimal value.

Control registers				
pc	mepc	mcause	mtvec	mstatus
00000000	00000000	00000000	00000000	00000003
mip	mie	mscratch	mtime	mtimecmp
00000000	00008888	00000000	00000000	00000000

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

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

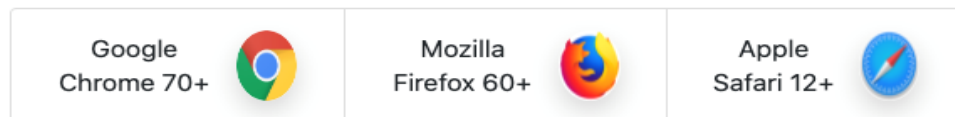
<https://creatorsim.github.io/creator/>



Características

- ▶ Permite describir características y juego de instrucciones de una arquitectura
 - ▶ Actualmente: [RISC-V \(RV32IMFD\)](#), [RISC-V \(RV64IMFD\)](#), MIPS-32, Z80, [RISC-V Sail 32 \(full\)](#), [RISC-V Sail 64 \(full\)](#) y arquitectura a medida
- ▶ Editar/compilar programas en ensamblador del juego de instrucciones elegido
- ▶ Ejecutar/depurar programas en ensamblador en un mismo entorno
- ▶ Ejecutar programas en ensamblador en hardware real
- ▶ Obtener estadísticas sobre los programas ejecutados
- ▶ Ejecución en navegador y en línea de mandatos

Supported Browsers



Disponibilidad

CREATOR

An educational integrated development environment for assembly programming, developed by the ARCOS research group at UC3M.

Try CREATOR

Documentation

Usage Statistics

Try CREATOR Beta

Register Inspector

<https://creatorsim.github.io>

RISC-V

RISC-V Exchange

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.

SUBMIT AN ITEM

CONTACT US

Search Exchange...

Hardware Cores Software Services Learning

CREATOR Simulator

ARCOS

Organization: UC3M
CREATOR: didaCtic and generic assEmbly progrAmming simulaTOR
Software Type: Simulators

SOFTWARE

LEARN MORE

Software
Software Type

- Accelerated Libraries
- Accelerated Libraries, Linux, macOS
- Application Infrastructure
- Application Infrastructure, Simulators
- Bootloaders
- BSD Distro
- C Compilers and Libraries
- C compilers and libraries, Compilers and runtimes for other languages
- Cloud infrastructure

<https://riscv.org/exchange>



Entornos RISC-V soportados

- ▶ Entorno para el juego de instrucciones RV32IMFD y RV64IMFD basado en una especificación propia de Creator
- ▶ Entorno basado en la especificación SAIL (lenguaje para describir ISA)
 - ▶ <https://github.com/riscv/sail-riscv>: especificación formal del juego de instrucciones RISC-V
 - ▶ Juego completo

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: `rdcyle`

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 <i>“tira de caracteres”</i>	Almacena cadena caracteres NO terminada en carácter nulo
.string <i>“tira de caracteres”</i>	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 <i>n</i>	Alinea el siguiente dato en un límite de 2^n

CREATOR (RISC-V)

The screenshot shows the CREATOR application window. The title bar includes the CREATOR logo, a 'Help' button, and a notification: 'Welcome to new CREATOR! For previous version, click here.' The main content area displays a list of processor architectures, each with a logo, a title, a description, and a right-pointing arrow button. The first item, 'RISC-V (RV32IMFD)', is highlighted with a red rectangular box. Below it are 'RISC-V (RV64IMFD)', 'RISC-V Sail 32 - Full Specification', and 'RISC-V Sail 64 - Full Specification'. Further down are 'MIPS-32', 'Z80', and a 'Load Custom Architecture' option with a plus sign button.

RISC-V (RV32IMFD)
RISC-V is a free and open-standard instruction set architecture (ISA) based on RISC principles. This architecture was developed in 2010 at the University of California, Berkeley.
This ISA includes the 32-bit I (integer), M (multiplication and division), F (single-precision floating-point), and D (double-precision floating-point) RISC-V extensions.

RISC-V (RV64IMFD)
RISC-V is a free and open-standard instruction set architecture (ISA) based on RISC principles. This architecture was developed in 2010 at the University of California, Berkeley.
This ISA includes the 64-bit I (integer), M (multiplication and division), F (single-precision floating-point), and D (double-precision floating-point) RISC-V extensions.

RISC-V Sail 32 - Full Specification
RISC-V is a free and open-standard instruction set architecture (ISA) based on RISC principles. This architecture was developed in 2010 at the University of California, Berkeley.
This ISA includes the full 32-bit specification of RISC-V based on Sail.

RISC-V Sail 64 - Full Specification
RISC-V is a free and open-standard instruction set architecture (ISA) based on RISC principles. This architecture was developed in 2010 at the University of California, Berkeley.
This ISA includes the full 64-bit specification of RISC-V based on Sail.

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.

Z80
The Z80 is an 8-bit microprocessor that was designed by Zilog and introduced in 1976. It became very popular in the late 1970s and early 1980s, especially in home computers and embedded systems.

Load Custom Architecture
Import your own architecture definition file (.yaml)

Pantalla inicial

The screenshot shows the CREATOR IDE interface. The top menu bar includes "CREATOR", "View", "Tools", and "Help". Below the menu bar are control buttons for "Reset", "Step", "Run", and "Stop". A "Sentinel" button is also present. The main window is divided into several panes. The left pane contains a sidebar with icons for "#", "Settings", "File", "Share", and "Calculator". The top pane shows "Address", "User Instruction", and "Loaded Instructions". The right pane is titled "Registers" and contains three sections:

- Control registers**: A grid of 10 registers with values: pc (00000000), mepc (00000000), mcause (00000000), mtvec (00000000), mstatus (00000000), mip (00000000), mie (00000000), mscratch (00000000), mtime (00000000), and mtimecmp (00000000).
- Integer registers**: A grid of 32 registers (x0 to x31) with values: x0 | zero (00000000), x1 | ra (FFFFFFFF), x2 | sp (0FFFFFFC), x3 | gp (00000000), x4 | tp (00000000), x5 | t0 (00000000), x6 | t1 (00000000), x7 | t2 (00000000), x8 | fp | s0 (00000000), x9 | s1 (00000000), x10 | a0 (00000000), x11 | a1 (00000000), x12 | a2 (00000000), x13 | a3 (00000000), x14 | a4 (00000000), x15 | a5 (00000000), x16 | a6 (00000000), x17 | a7 (00000000), x18 | s2 (00000000), x19 | s3 (00000000), x20 | s4 (00000000), x21 | s5 (00000000), x22 | s6 (00000000), x23 | s7 (00000000), x24 | s8 (00000000), x25 | s9 (00000000), x26 | s10 (00000000), x27 | s11 (00000000), x28 | t3 (00000000), x29 | t4 (00000000), x30 | t5 (00000000), and x31 | t6 (00000000).
- Floating point registers**: A grid of 32 registers (f0 to f31) with values: f0 | ft0 (0000000000000000), f1 | ft1 (0000000000000000), f2 | ft2 (0000000000000000), f3 | ft3 (0000000000000000), f4 | ft4 (0000000000000000), f5 | ft5 (0000000000000000), f6 | ft6 (0000000000000000), f7 | ft7 (0000000000000000), f8 | fs0 (0000000000000000), f9 | fs1 (0000000000000000), f10 | fa0 (0000000000000000), f11 | fa1 (0000000000000000), f12 | fa2 (0000000000000000), f13 | fa3 (0000000000000000), f14 | fa4 (0000000000000000), f15 | fa5 (0000000000000000), f16 | fa6 (0000000000000000), f17 | fa7 (0000000000000000), f18 | fs2 (0000000000000000), f19 | fs3 (0000000000000000), f20 | fs4 (0000000000000000), f21 | fs5 (0000000000000000), f22 | fs6 (0000000000000000), f23 | fs7 (0000000000000000), f24 | fs8 (0000000000000000), f25 | fs9 (0000000000000000), f26 | fs10 (0000000000000000), f27 | fs11 (0000000000000000), f28 | ft8 (0000000000000000), f29 | ft9 (0000000000000000), f30 | ft10 (0000000000000000), and f31 | ft11 (0000000000000000).

Inspección y edición de la arquitectura

The screenshot shows the CREATOR software interface. The top menu bar includes 'CREATOR', 'View', 'Tools', and 'Help'. Below the menu bar, there are control buttons for 'Reset', 'Step', 'Run', and 'Stop'. The main interface is divided into several sections: 'Address', 'Loaded Instructions', 'Registers', 'Memory', 'Terminal', and 'Statistics'. The 'Registers' section is expanded, showing three categories: Control registers, Integer registers, and Floating point registers. The 'Architecture' option in the left sidebar is highlighted with a red circle. Below the circle, the text 'Ver/Editar arquitectura' is written in red.

Ver/Editar arquitectura

Control registers				
pc	mepc	mcause	mtvec	mstatus
00000000	00000000	00000000	00000000	00000008
mip	mie	mscratch	mtime	mtimecmp
00000000	00000888	00000000	00000000	00000000

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

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

Edición de programas

The screenshot shows the CREATOR software interface. The top menu bar includes 'CREATOR', 'View', 'Tools', and 'Help'. Below the menu bar, there are control buttons for 'Reset', 'Step', 'Run', and 'Stop'. A 'Sentinel' icon and a welcome message are also visible. The main interface is divided into several panels: 'Address' (with a '# Editor' button circled in red), 'Loaded Instructions', 'Registers', 'Memory', 'Terminal', and 'Statistics'. The 'Registers' panel is expanded to show three sections: 'Control registers', 'Integer registers', and 'Floating point registers'. Each register is displayed in a grid format with its name and a hexadecimal value.

Control registers

pc	mepc	mcause	mtvec	mstatus
00000000	00000000	00000000	00000000	00000008
mip	mie	mscratch	mtime	mtimecmp
00000000	00008888	00000000	00000000	00000000

Integer registers

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

Floating point registers

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

Edición de programas

Control de la ejecución

The screenshot shows the CREATOR IDE interface. At the top, the menu bar includes 'CREATOR', 'View', 'Tools', and 'Help'. Below the menu bar, there is a toolbar with several icons. A red circle highlights the execution control buttons: 'Reset', 'Step', 'Run', and 'Stop'. To the right of these buttons is a 'Sentinel' icon and a notification area. Below the toolbar, the main workspace is divided into several panels. On the left, there is a sidebar with icons for '# Editor', 'Simulator', and 'Architecture'. The main area is titled 'Loaded Instructions' and contains a table of registers. The registers are organized into three sections: 'Control registers', 'Integer registers', and 'Floating point registers'. Each register is displayed in a grid format with its name and a hexadecimal value.

Control de la ejecución

Control registers				
pc	mepc	mcause	mtvec	mstatus
00000000	00000000	00000000	00000000	00000008
mip	mie	mscratch	mtime	mtimecmp
00000000	00008888	00000000	00000000	00000000

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

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

Ejemplos de programas en ensamblador

Ejemplos

The screenshot shows the CREATOR IDE interface. A modal window titled "Available examples" is open, displaying a list of 13 assembly examples. The examples are:

- 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
- Example 13: Stack Overflow Vulnerability Demonstration

The background shows the main IDE interface with tabs for Registers, Memory, Terminal, and Statistics. The Registers tab is active, showing control registers like pc, mepc, mcause, mtvec, mstatus, mscratch, mtime, mtimecmp, and general purpose registers x2 through x29, and floating point registers f2 through f29.

Calculadora de números en coma flotante

Calculadora

IEEE 754 Floating Point Converter

32-bit (Float) 64-bit (Double)

IEEE 754 BREAKDOWN

SIGN: 0

EXPONENT (8B): 00000000

MANTISSA (23B): 00000000000000000000000

$value = (-1)^{sign} \times 2^{exponent - 127} \times (1 + mantissa)$

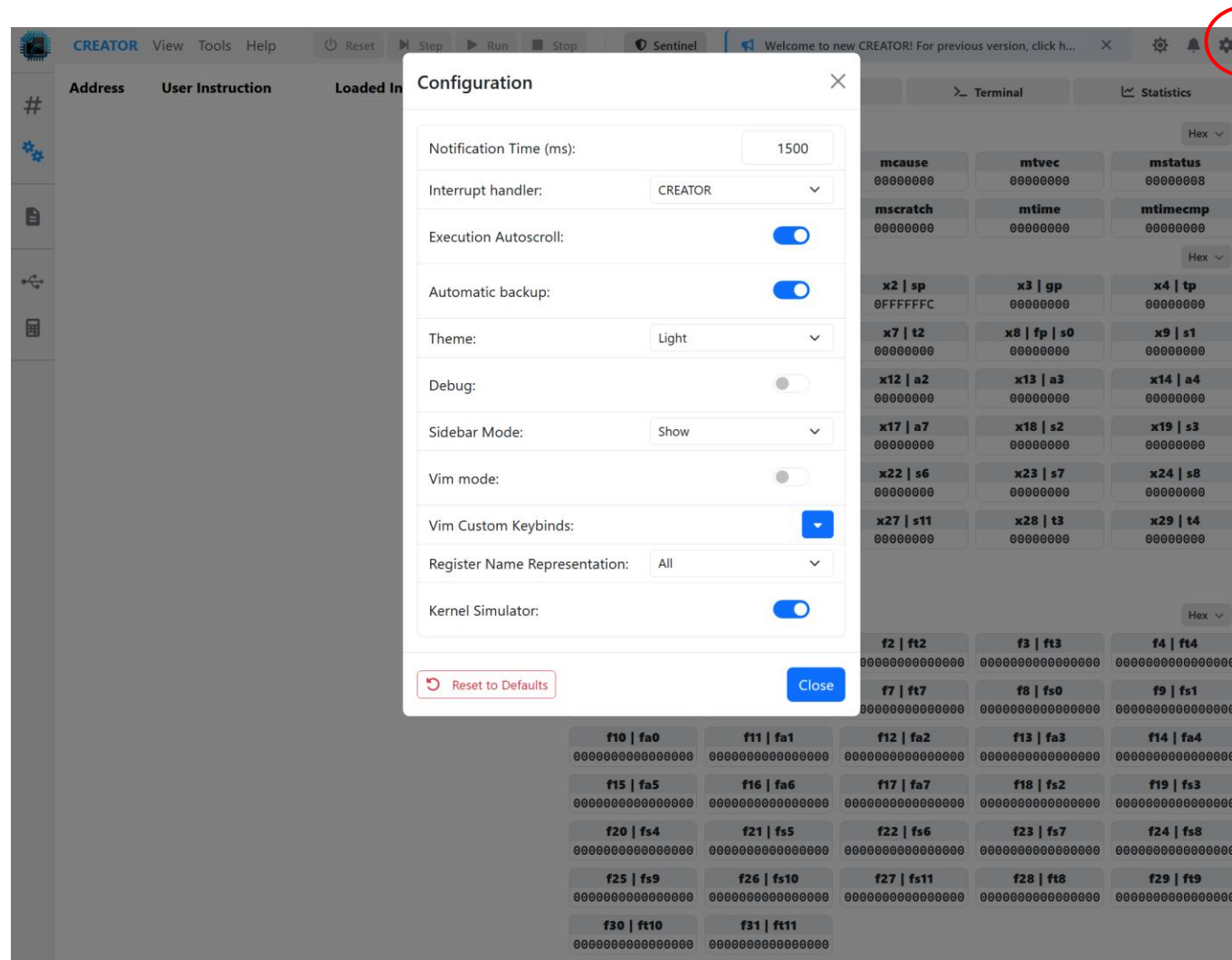
VALUE REPRESENTATIONS

DECIMAL: e.g., 3.14159

HEXADECIMAL: e.g., 0x40490FDB

BINARY: 32-bit binary

Configuración



Configuración

The screenshot shows the CREATOR IDE interface. The top menu bar includes 'CREATOR', 'View', 'Tools', and 'Help'. The 'Help' menu is open, showing options like 'Examples...', 'CREATOR Help', and 'RISC-V (RV32IMFD) Guide', with the latter highlighted by a red circle. The main workspace is divided into several panels: 'Address', 'User Instr', 'Instructions', 'Registers', 'Memory', 'Terminal', and 'Statistics'. The 'Registers' panel is active, displaying a list of registers categorized into Control registers, Integer registers, and Floating point registers. Each register entry shows its name and current value (mostly 00000000).

Control registers				
pc	mepc	mcause	mtvec	mstatus
00000000	00000000	00000000	00000000	00000008
mip	mie	mscratch	mtime	mtimecmp
00000000	00000888	00000000	00000000	00000000

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

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

Address User Instr Examples... Instructions

- Examples...
- CREATOR Help
- RISC-V (RV32IMFD) Guide**

Registers Memory Terminal Statistics

Control registers

pc 00000000	mepc 00000000	mcause 00000000	mtvec 00000000	mstatus 00000008
mip 00000000	mie 00000888	mscratch 00000000	mtime 00000000	mtimecmp 00000000

Integer registers

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

Floating point registers

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

RISC-V Reference Guide (CREATOR Simulator)

Service	Call Code	Arguments	Result
Print_int	1	a0 = integer	
Print_float	2	f0 = float	
Print_double	3	f0 = double	
Print_string	4	a0 = string addr	
Read_int	5		Integer in a0
Read_float	6		Float in f0
Read_double	7		Double in f0
Read_string	8	a0 = string addr a1 = length	String in a0
Shk	9	a0 = length Address in a0	
Exit	10		
Print_char	11	a0 = ASCII code	
Read_char	12		Char in a0

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..a7	Arguments for functions / return value
f0..f11	Temporary (NOT preserved across calls)
fs0..fs11	Saved temporary (preserved across calls)
fa0..fa7	Arguments for functions / return value

Instruction	Operation
li rd, n	rd = n (PseudoInst, n >= 32 bits)
mv rd, rs	rd = rs
lui rd, imm	rd = (imm[12:1]) << 12 (extend the sign)
add rd, rs1, rs2	rd = rs1 + rs2
addi rd, rs1, n	rd = rs1 + n (n >= 12 bits)
sub rd, rs1, rs2	rd = rs1 - rs2
subi rd, rs1, n	rd = rs1 - n
and rd, rs1, rs2	rd = rs1 & rs2
andi rd, rs1, n	rd = rs1 & n (n >= 12 bits)
or rd, rs1, rs2	rd = rs1 rs2
ori rd, rs1, n	rd = rs1 n (n >= 12 bits)
not rd, rs1	rd = ~rs1 (one's complement)
xor rd, rs1, rs2	rd = rs1 ^ rs2
xori rd, rs1, n	rd = rs1 ^ n (n >= 12 bits)
sll rd, rs1, n	rd = rs1 << n (logical, n <= 5 bits)
slli rd, rs1, n	rd = rs1 << n (logical, n <= 5 bits)
srl rd, rs1, n	rd = rs1 >> n (arithmetic, n <= 5 bits)
sri rd, rs1, n	rd = rs1 >> n (arithmetic, n <= 5 bits)
slw rd, rs1, rs2	rd = rs1 >> rs2 (arithmetic)
sli rd, rs1, rs2	rd = rs1 << rs2 (logical)
sri rd, rs1, rs2	rd = rs1 >> rs2 (logical)

Instruction	Operation
beq t0, t1, etiq	Jump to etiq if t0 == t1
bne t0, t1, etiq	Jump to etiq if t0 != t1
blt t0, t1, etiq	Jump to etiq if t0 < t1
bltu t0, t1, etiq	Jump to etiq if t0 < t1 (unsigned)
bgt t0, t1, etiq	Jump to etiq if t0 > t1
bgtu t0, t1, etiq	Jump to etiq if t0 > t1 (unsigned)
ble t0, t1, etiq	Jump to etiq if t0 <= t1
bleu t0, t1, etiq	Jump to etiq if t0 <= t1 (unsigned)
j etiq	PC = PC + etiq

Instruction	Operation
la rd, address	rd = address >> 12 bits
lb rd, mem1	rd = Memory(mem1) load byte
lh rd, mem1	rd = Memory(mem1) load half word
lw rd, mem1	rd = Memory(mem1) load word
lbu rd, mem1	rd = Memory(mem1) load byte (unsigned)
lhu rd, mem1	rd = Memory(mem1) load half word (unsigned)
lwu rd, mem1	rd = Memory(mem1) load word (unsigned)
sb rd, mem1	Memory(mem1) = rd store byte
sh rd, mem1	Memory(mem1) = rd store half word
sw rd, mem1	Memory(mem1) = rd store word

Instruction	Operation
fcvt.w.s rd, fs1	Free single precision (fs1) to integer (rd) with sign
fcvt.w.u rd, fs1	Free single precision (fs1) to integer (rd) without sign
fcvt.s.w rd, rs1	Free integer without sign (rs1) to single precision (rd)
fcvt.d.w rd, rs1	Free integer without sign (rs1) to double precision (rd)
fcvt.w.d rd, fs1	Free double precision (fs1) to integer (rd) with sign
fcvt.w.u rd, fs1	Free double precision (fs1) to integer (rd) without sign
fcvt.d.w rd, rs1	Free integer with sign (rs1) to double precision (rd)
fcvt.d.s rd, fs1	Free integer without sign (rs1) to double precision (rd)
fcvt.s.d rd, fs1	Free double (fs1) to single precision (rd)
fcvt.d.s rd, fs1	Free single (fs1) to double precision (rd)

ARCOS-UC3M



Registros enteros

The screenshot displays the CREATOR IDE interface. The top menu bar includes 'CREATOR', 'View', 'Tools', and 'Help'. Below the menu, there are control buttons for 'Reset', 'Step', 'Run', and 'Stop', along with a 'Sentinel' icon and a welcome message. The main workspace is divided into several panels: 'Address', 'Loaded Instructions', 'Registers', 'Memory', 'Terminal', and 'Statistics'. The 'Registers' panel is active and circled in red. It is divided into three sections: 'Control registers', 'Integer registers', and 'Floating point registers'. The 'Integer registers' section is also circled in red and contains a grid of 32 registers. The values for these registers are as follows:

Register	Value
pc	00000000
mepc	00000000
mcause	00000000
mtvec	00000000
mstatus	00000008
mip	00000000
mie	00000888
mscratch	00000000
mtime	00000000
mtimecmp	00000000
x0 zero	00000000
x1 ra	FFFFFFFF
x2 sp	0FFFFFFC
x3 gp	00000000
x4 tp	00000000
x5 t0	00000000
x6 t1	00000000
x7 t2	00000000
x8 fp s0	00000000
x9 s1	00000000
x10 a0	00000000
x11 a1	00000000
x12 a2	00000000
x13 a3	00000000
x14 a4	00000000
x15 a5	00000000
x16 a6	00000000
x17 a7	00000000
x18 s2	00000000
x19 s3	00000000
x20 s4	00000000
x21 s5	00000000
x22 s6	00000000
x23 s7	00000000
x24 s8	00000000
x25 s9	00000000
x26 s10	00000000
x27 s11	00000000
x28 t3	00000000
x29 t4	00000000
x30 t5	00000000
x31 t6	00000000

Registros enteros

The screenshot shows the CREATOR IDE interface. The 'Registers' tab is selected and circled in red. The 'Integer registers' section is also circled in red, showing a grid of registers with their names and values. The 'Hex' dropdown menu is also circled in red.

Control registers				
pc	mepc	mcause	mtvec	mstatus
00000000	00000000	00000000	00000000	00000008
mip	mie	mscratch	mtime	mtimecmp
00000000	00000888	00000000	00000000	00000000

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

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

Registros en coma flotante

The screenshot shows the CREATOR IDE interface. The 'Registers' tab is selected and circled in red. The floating point registers section is also circled in red, with a 'Hex' dropdown menu highlighted.

Control registers

pc	mepc	mcause	mtvec	mstatus
00000000	00000000	00000000	00000000	00000008
mip	mie	mscratch	mtime	mtimecmp
00000000	00008888	00000000	00000000	00000000

Integer registers

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

Floating point registers

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

Registros en coma flotante

CREATOR View Tools Help Reset Step Run Stop Sentinel Welcome to new CREATOR! For previous version, click h...

Address # Editor Simulator Architecture

Loaded Instructions Registers Memory Terminal Statistics

Control registers Hex

pc	mepc	mcause	mtvec	mstatus
00000000	00000000	00000000	00000000	00000008
mip	mie	mscratch	mtime	mtimecmp
00000000	00008888	00000000	00000000	00000000

Integer registers Hex

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

Floating point registers Hex

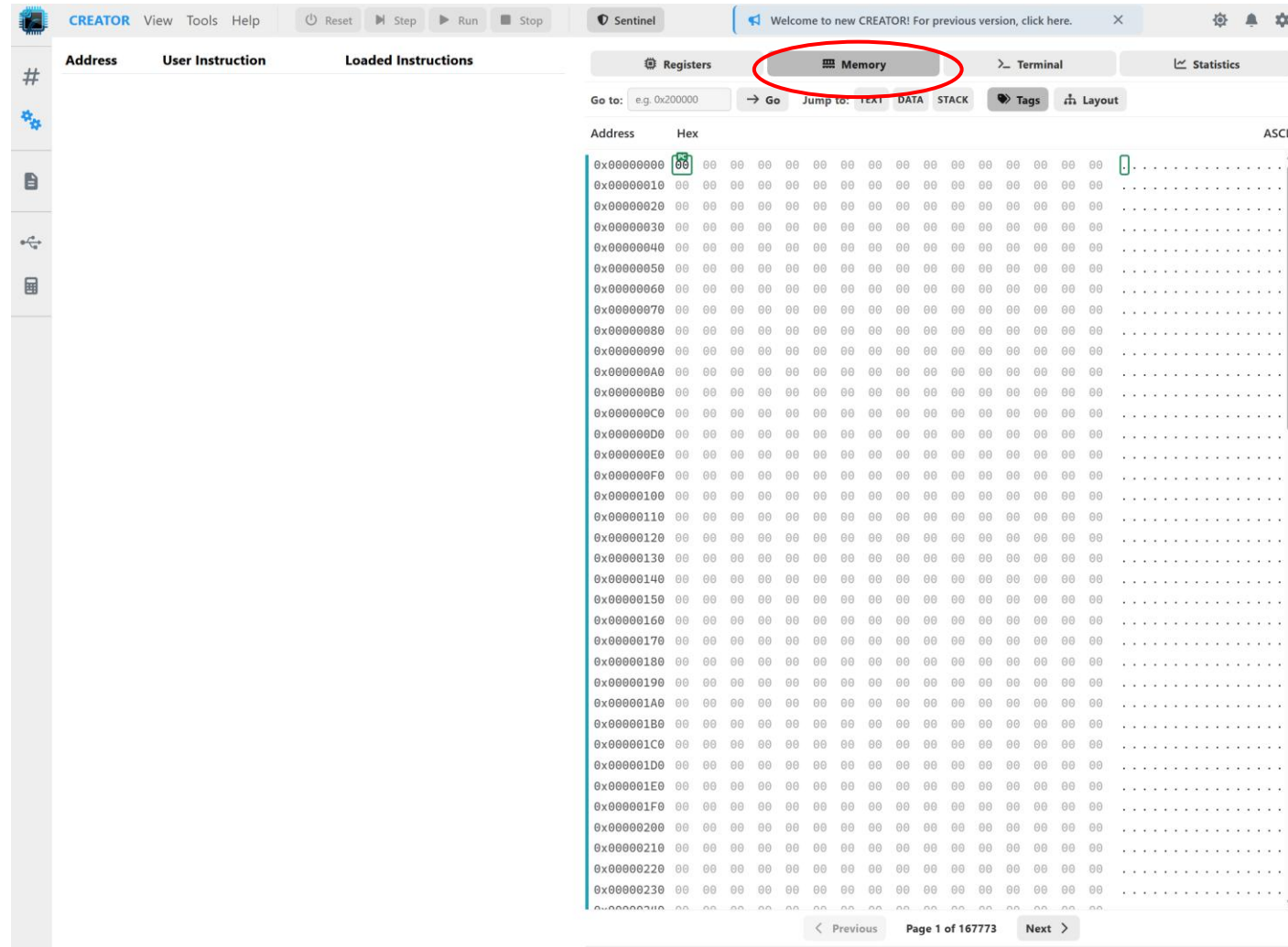
f0 ft0	f1 ft1	f2 ft2	f3 ft3	f4 ft4
0000000000000000	0000000000000000	0000000000000000	0000000000000000	0000000000000000
f5 fa5	f6 fa6	f7 fa7	f8 fa8	f9 fa9
0000000000000000	0000000000000000	0000000000000000	0000000000000000	0000000000000000
f10 fa0	f11 fa1	f12 fa2	f13 fa3	f14 fa4
0000000000000000	0000000000000000	0000000000000000	0000000000000000	0000000000000000
f15 fa5	f16 fa6	f17 fa7	f18 fa8	f19 fa9
0000000000000000	0000000000000000	0000000000000000	0000000000000000	0000000000000000
f20 fs4	f21 fs5	f22 fs6	f23 fs7	f24 fs8
0000000000000000	0000000000000000	0000000000000000	0000000000000000	0000000000000000
f25 fs9	f26 fs10	f27 fs11	f28 fs12	f29 fs13
0000000000000000	0000000000000000	0000000000000000	0000000000000000	0000000000000000
f30 ft10	f31 ft11			
0000000000000000	0000000000000000			

Space view for f15 | fa5

Hexadecimal	000000003F800000
Binary	00111111100000000000000000000000
IEEE 754 (32 bits)	1
IEEE 754 (64 bits)	5.263544247e-315

Enter new value Update

Contenido de la memoria



The screenshot shows the CREATOR IDE interface with the Memory view selected. The Memory view displays a memory dump with the following columns: Address, Hex, and ASCII. The memory dump shows a sequence of 00 values in hex, which correspond to a period (.) in ASCII. The memory dump is scrollable, and the current page is 1 of 16773.

Address	Hex	ASCII
0x00000000	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000010	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000020	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000030	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000040	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000050	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000060	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000070	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000080	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000090	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x000000A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x000000B0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x000000C0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x000000D0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x000000E0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x000000F0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000100	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000110	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000120	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000130	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000140	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000150	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000160	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000170	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000180	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000190	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x000001A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x000001B0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x000001C0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x000001D0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x000001E0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x000001F0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000200	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000210	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000220	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000230	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Pantalla

The screenshot displays the CREATOR IDE interface. At the top, the title bar shows 'CREATOR' and a notification: 'Welcome to new CREATOR! For previous version, click here.' The main workspace is divided into two panes. The left pane contains a table of assembly instructions, and the right pane shows a terminal window.

Address	User Instruction	Loaded Instructions
0x0 main	la t0, max	auipc t0, 512
0x4		addi t0, t0, 0
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, 16
0x18	add a0, a0, t1	add a0, a0, t1
0x1C	addi t1, t1, 1	addi t1, t1, 1
0x20	beq zero, zero, while	beq zero, zero, -12
0x24 end_while	li a7, 1	addi a7, x0, 1
0x28	ecall	ecall
0x2C	li a7, 10	addi a7, x0, 10
0x30	ecall	ecall

The terminal window on the right is currently empty, with a cursor at the top left. The 'Terminal' tab is highlighted with a red circle in the original image.

Estadísticas de ejecución

The screenshot displays the CREATOR IDE interface. The top menu bar includes 'CREATOR', 'View', 'Tools', 'Help', 'Reset', 'Finished', 'Stop', 'Sentinel', and a welcome message. The main window is divided into three panes: a table of instructions, a summary of execution metrics, and a detailed instruction statistics bar chart.

Address	User Instruction	Loaded Instructions
0x0 main	la t0, max	auipc t0, 512
0x4		addi t0, t0, 0
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, 16
0x18	add a0, a0, t1	add a0, a0, t1
0x1C	addi t1, t1, 1	addi t1, t1, 1
0x20	beq zero, zero, while	beq zero, zero, -12
0x24 end_while	li a7, 1	addi a7, x0, 1
0x28	ecall	ecall
0x2C	li a7, 10	addi a7, x0, 10
0x30	ecall	ecall

Summary Metrics:

- TOTAL INSTRUCTIONS: 50
- TOTAL CYCLES: 50
- CPI: 1.00
- IPC: 1.00

INSTRUCTION STATISTICS:

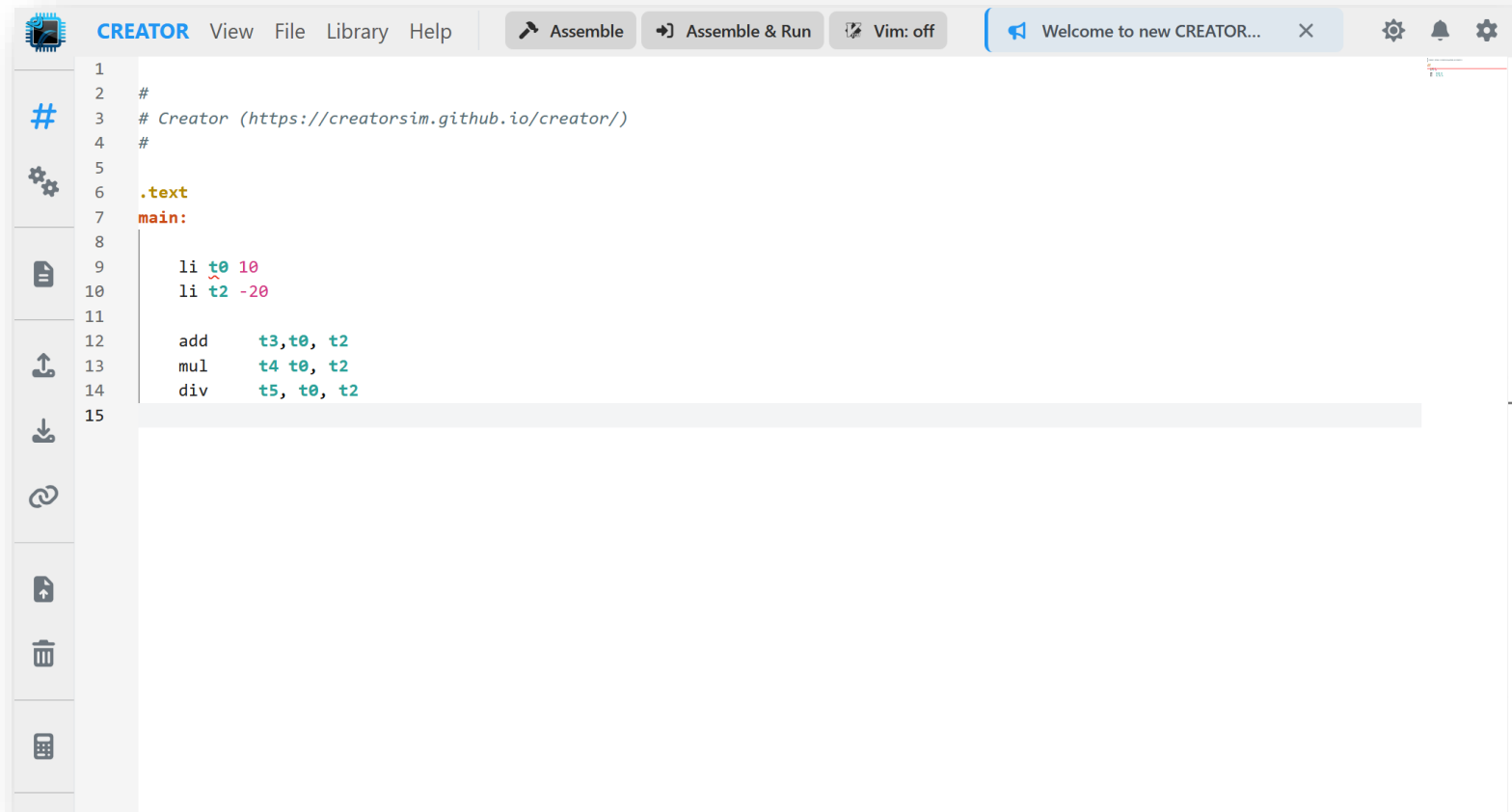
Category	Count	Cycles
Arithmetic integer	26	
Arithmetic floating point	0	
Comparison	0	
Conditional bifurcation	21	
Control	0	
Function call	0	
I/O	0	
Logic	0	
Memory access	1	
Syscall	2	
Transfer between registers	0	
Unconditional bifurcation	0	
Other	0	

Bloque 1

- ▶ ¿Qué es CREATOR?
 - ▶ El origen de CREATOR
 - ▶ Características de CREATOR
 - ▶ ¿Dónde se usa CREATOR?
 - ▶ Extensiones futuras
- ▶ **¿Cómo utilizar CREATOR? Visión del Estudiante**
 - ▶ Edición y compilación de programas
 - ▶ Ejecución y depuración de programas
 - ▶ Convenio de paso de parámetros y uso de pila (Sentinel)
 - ▶ Bibliotecas de funciones
- ▶ ¿Cómo utilizar CREATOR? Visión del Profesor
 - ▶ Soporte para la creación de material didáctico
 - ▶ Soporte para la corrección de prácticas
 - ▶ Soporte para la creación y edición de arquitecturas

Edición de programas en ensamblador

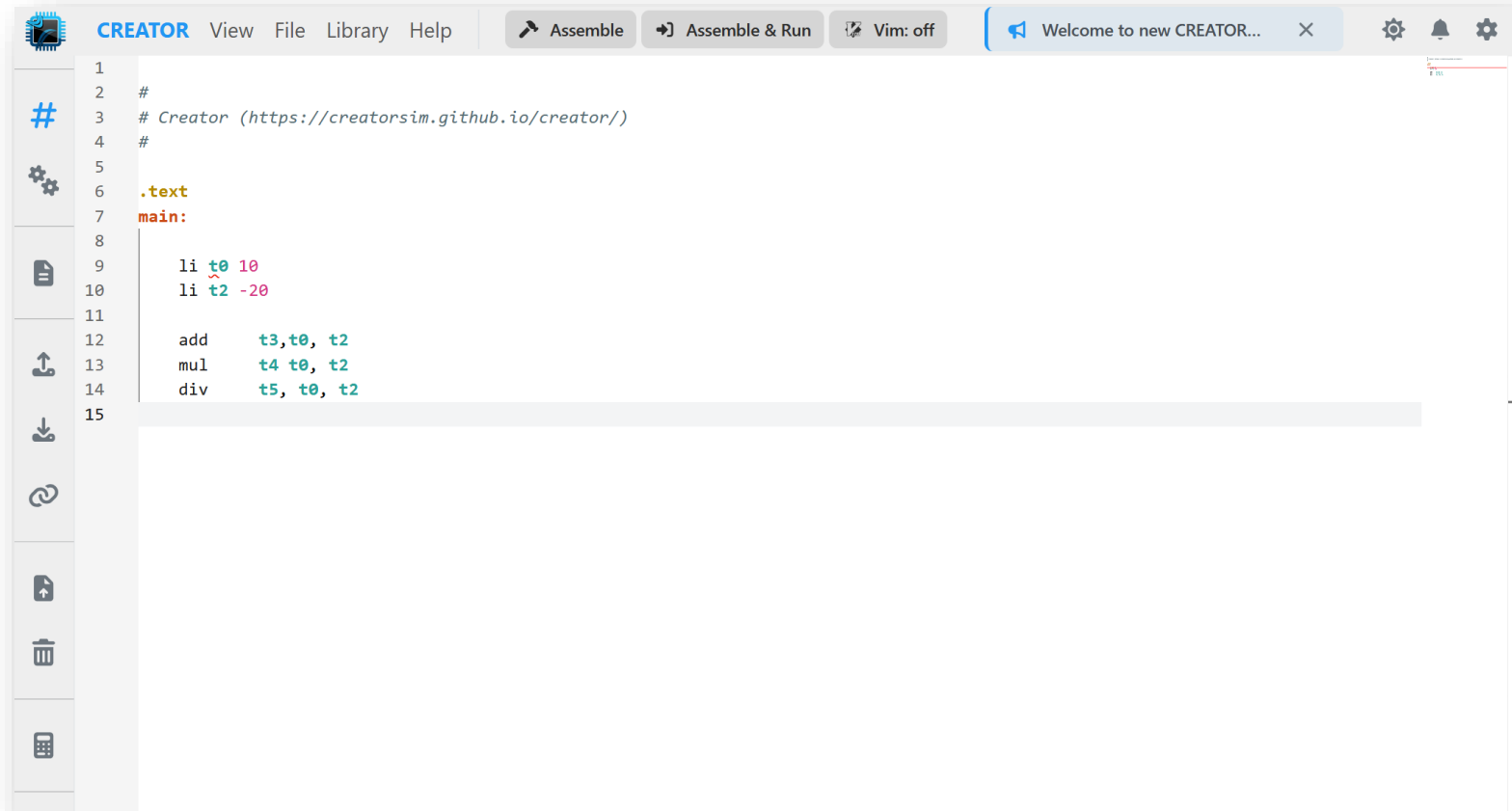
ejemplo



The screenshot shows the CREATOR IDE interface. The main window displays assembly code with line numbers 1 through 15. The code includes comments, a section header, and several instructions. The interface includes a menu bar (View, File, Library, Help), a toolbar (Assemble, Assemble & Run, Vim: off), and a status bar (Welcome to new CREATOR...). A vertical toolbar on the left contains icons for various actions like search, settings, and navigation.

```
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
```

Compilación



The screenshot shows the CREATOR IDE interface. The top menu bar includes 'View', 'File', 'Library', and 'Help'. Below the menu, there are buttons for 'Assemble', 'Assemble & Run', and 'Vim: off'. A tab titled 'Welcome to new CREATOR...' is open. The main editor area displays assembly code with line numbers 1 through 15. The code is as follows:

```
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 View File Library Help Assemble Assemble & Run Vim: off Welcome to new CREATOR...

```
1
2 #
3 # Creator
4 #
5
6 .text
7 main:
8
9     li t0 10
10    li t2 -
11
12    add
13    mul
14    div
15
```

Assembly Code Error

[E09] Error: Incorrect instruction syntax
[assembly:9:8]

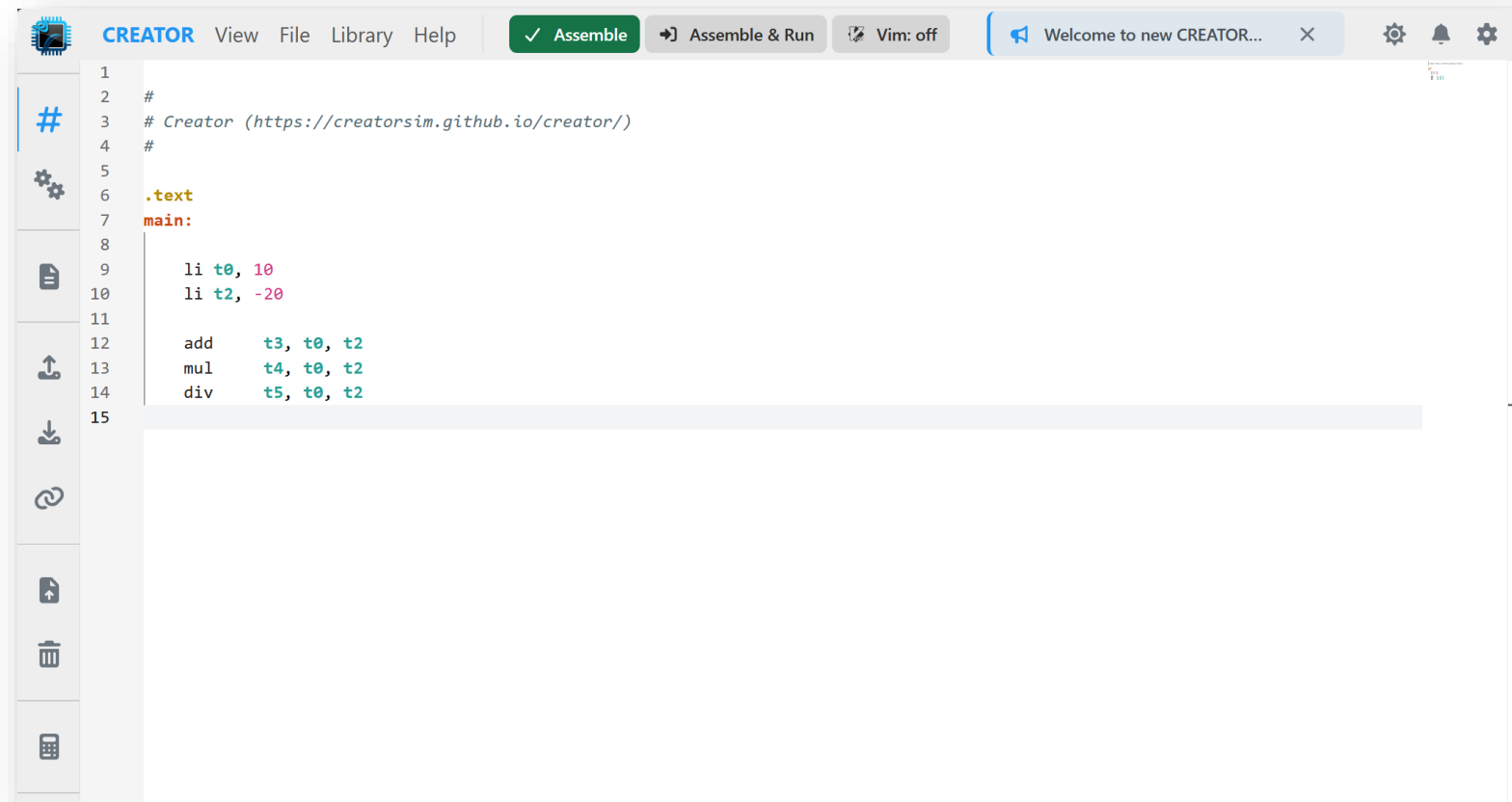
```
9     li t0 10
           |
           | Incorrect syntax
```

Note: Allowed formats:
opcode rd, val

The syntax `opcode rd, val` failed with the following reason:
Error: found integer (10) but expected `,` or binary operator
[assembly:9:11]

```
9     li t0 10
           |
           | While parsing this expression
           |
           | Unexpected input
```

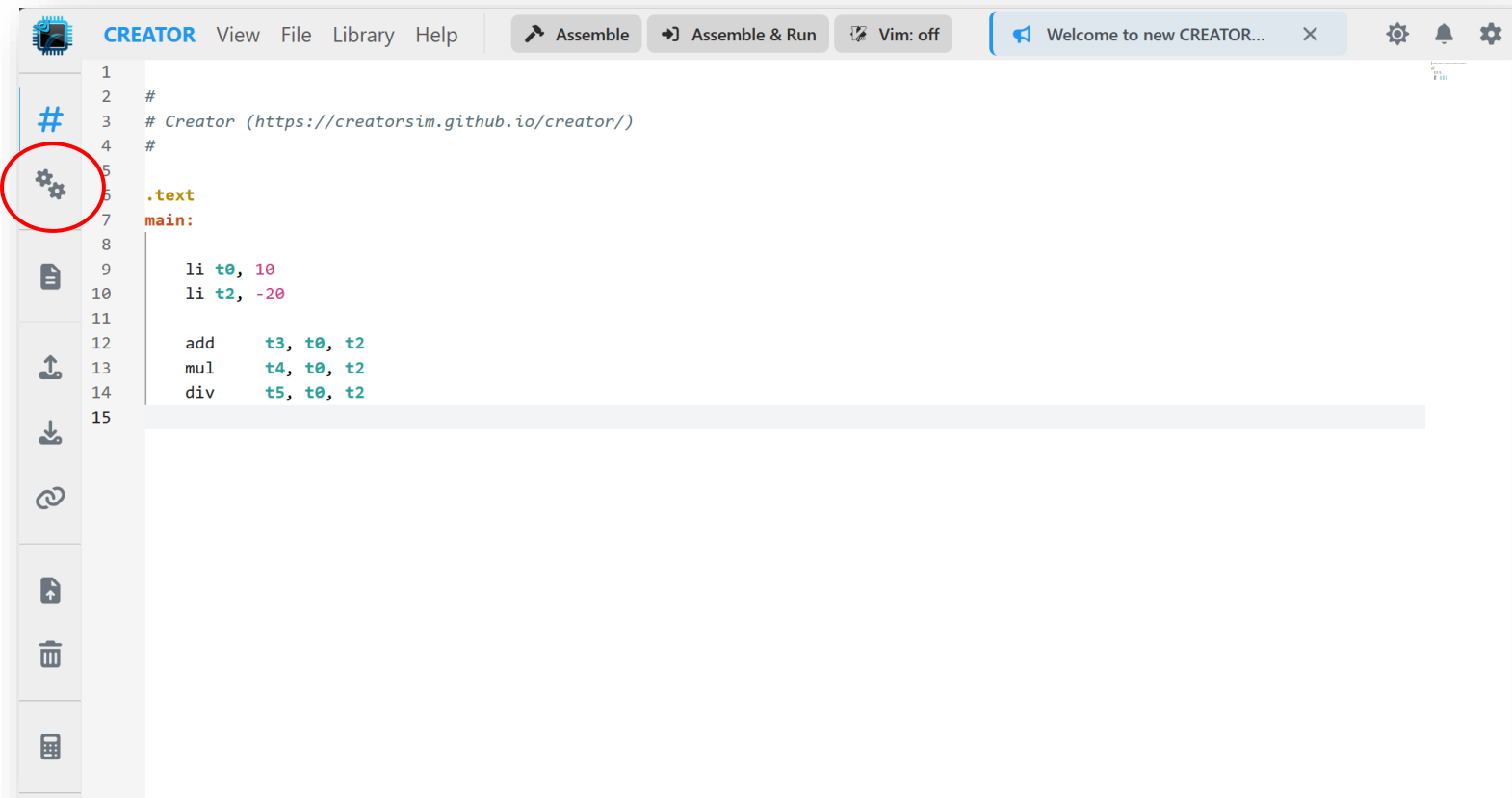
Compilación sin error



The screenshot shows the CREATOR IDE interface. The top menu bar includes 'CREATOR', 'View', 'File', 'Library', and 'Help'. Below the menu, there are buttons for 'Assemble' (with a green checkmark), 'Assemble & Run', and 'Vim: off'. A notification bar on the right says 'Welcome to new CREATOR...'. The main editor area displays assembly code on a line-by-line basis:

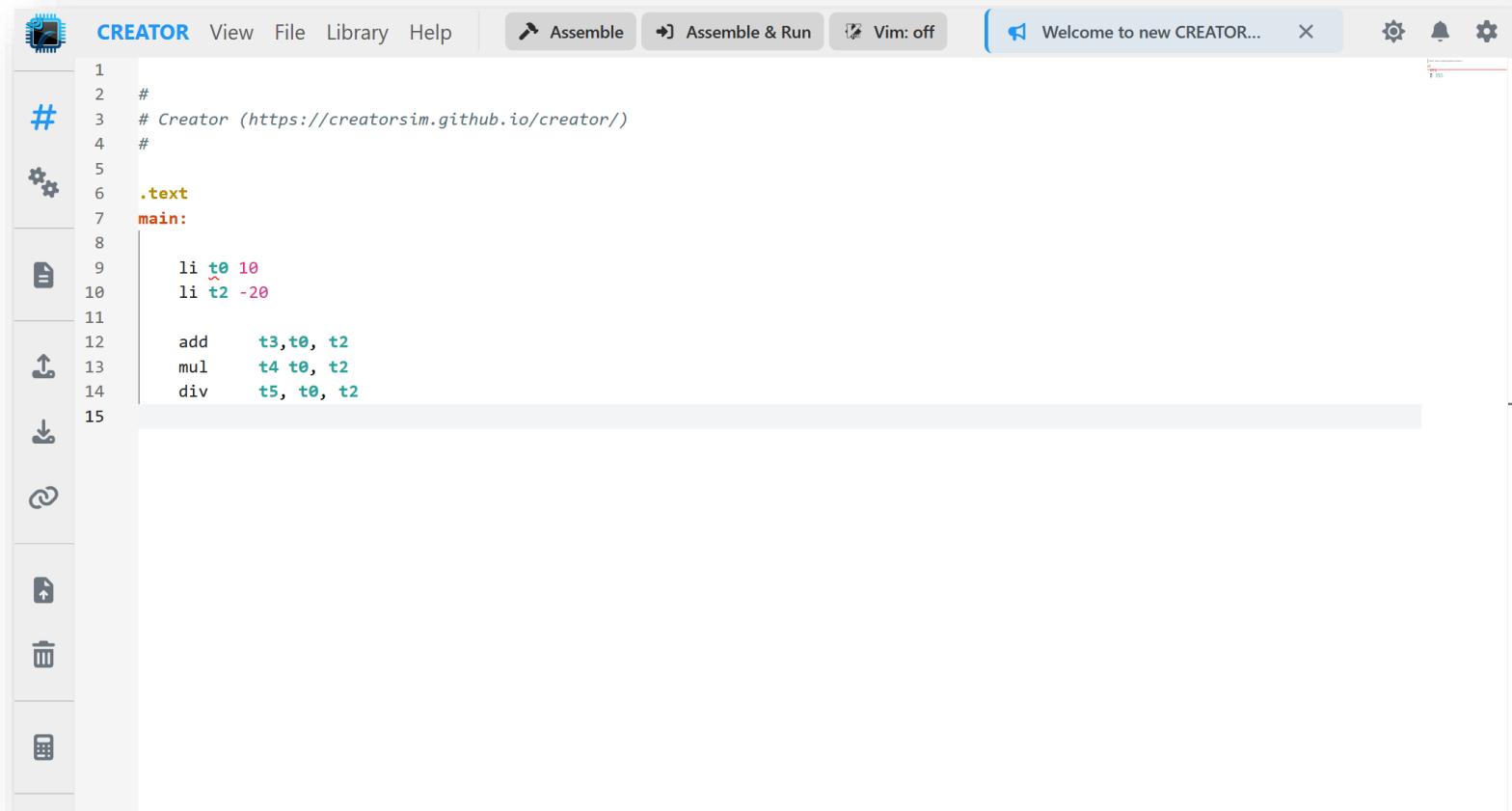
```
1  
2 #  
3 # Creator (https://creatorsim.github.io/creator/)  
4 #  
5  
6 .text  
7 main:  
8  
9     li t0, 10  
10    li t2, -20  
11  
12    add t3, t0, t2  
13    mul t4, t0, t2  
14    div t5, t0, t2  
15
```

Paso al simulador



Modificar el programa para imprimir el valor de t5

ejemplo



```
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

The screenshot displays the CREATOR simulator interface. The top menu includes 'View', 'Tools', and 'Help'. Below the menu are control buttons: 'Reset', 'Step', 'Run', and 'Stop'. A 'Sentinel' icon and a window title 'Welcome to new CREAT...' are also visible.

The main window is divided into two primary sections. On the left, a table lists instructions with their addresses, user-written instructions, and loaded machine instructions. On the right, a 'Registers' panel shows the state of various control and integer registers.

Address	User Instruction	Loaded Instructions
0x0	li t0, 10	addi t0, x0, 10
0x4	li t2, -20	addi t2, x0, -20
0x8	add t3, t0, t2	add t3, t0, t2
0xC	mul t4, t0, t2	mul t4, t0, t2
0x10	div t5, t0, t2	div t5, t0, t2

Control registers			
pc 00000000	mepc 00000000	mcause 00000000	mtvec 00000000
mstatus 00000008	mip 00000000	mie 00000888	mscratch 00000000
mtime 00000000	mtimecmp 00000000		

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

Programa escrito
(inst. y pseudoinst.)

Programa en memoria
(instrucciones máquina)

Flujo de ejecución

CREATOR View Tools Help

Reset Step Run Stop Sentinel

Address	User Instruction	Loaded Instructions
0x0 main	li t0, 10	addi t0, x0, 10
0x4	li t2, -20	addi t2, x0, -20
0x8	add t3, t0, t2	add t3, t0, t2
0xC	mul t4, t0, t2	mul t4, t0, t2 next
0x10	div t5, t0, t2	div t5, t0, t2

Registers Memory

Control registers

pc 0000000C	mepc 00000000
mstatus 00000008	mip 00000000
mtime 00000003	mtimecmp 00000000

Integer registers

x0 zero 00000000	x1 ra FFFFFFFF
x4 tp 00000000	x5 t0 0000000A
x8 fp s0 00000000	x9 s1 00000000

Puntos de ruptura

The screenshot shows the CREATOR IDE interface. At the top, there is a menu bar with 'View', 'Tools', and 'Help'. Below the menu bar are control buttons: 'Reset', 'Step', 'Run', 'Stop', 'Sentinel', and a volume icon. The main area is divided into two panels. The left panel shows a list of instructions with columns for 'Address', 'User Instruction', and 'Loaded Instructions'. The right panel shows the 'Registers' section, which is expanded to show 'Control registers' and 'Integer registers'.

Address	User Instruction	Loaded Instructions
0x0 main	li t0, 10	addi t0, x0, 10
0x4	li t2, -20	addi t2, x0, -20
0x8	add t3, t0, t2	add t3, t0, t2
• 0xC	mul t4, t0, t2	mul t4, t0, t2 next
0x10	div t5, t0, t2	div t5, t0, t2

Registers

Control registers

pc 0000000C	mepc 00000000
mstatus 00000008	mip 00000000
mtime 00000003	mtimecmp 00000000

Integer registers

x0 zero 00000000	x1 ra FFFFFFFF
x4 tp 00000000	x5 t0 0000000A
x8 fp s0 00000000	x9 s1 00000000

Segmento de datos

[ejemplo](#)

```
CREATOR View File Library Help Assemble Assemble & Run Vim: off
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

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
```

Registers Memory Terminal Statistics

Go to: e.g. 0x200000 → Go Jump to: TEXT DATA STACK

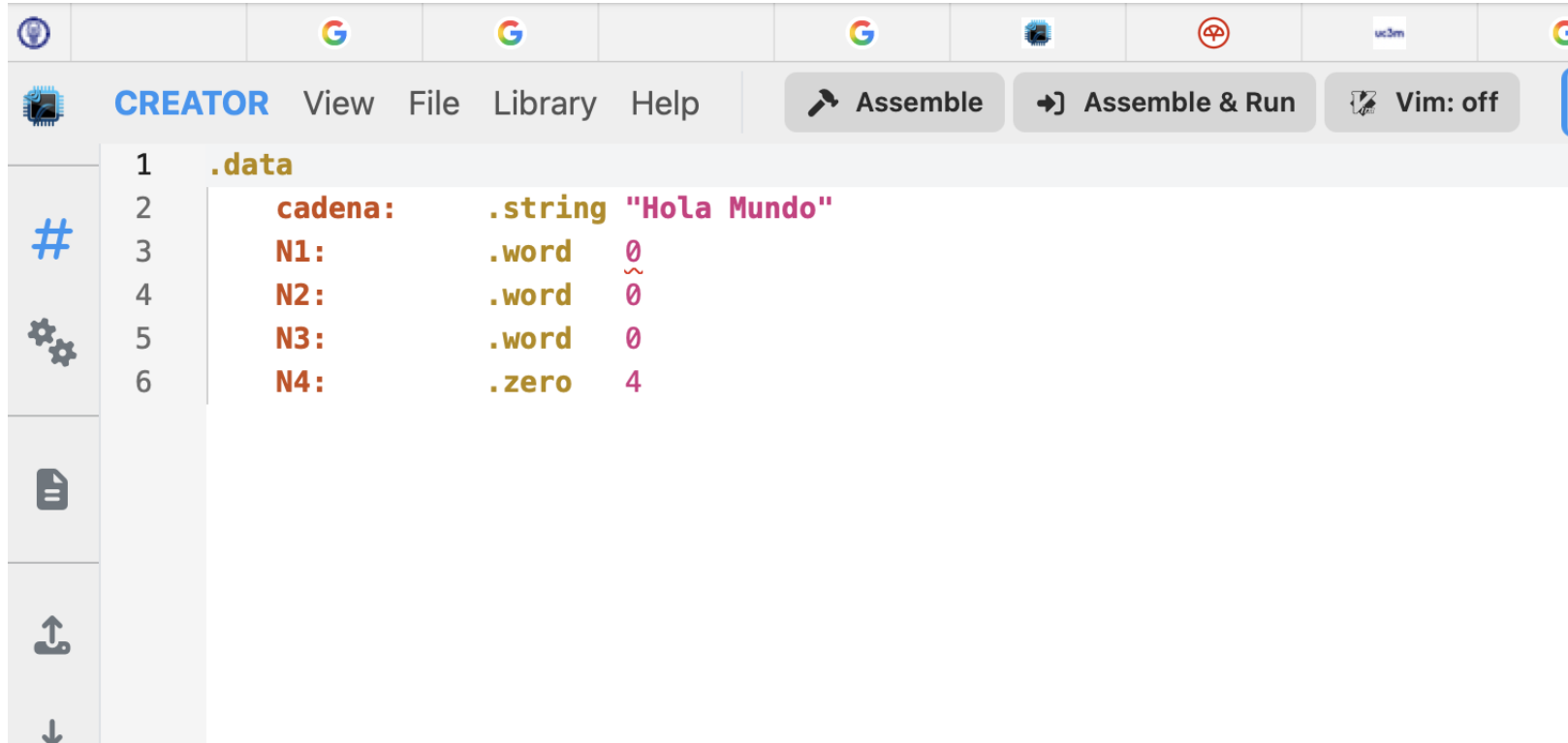
Hex Tags Layout

Address	Hex
	cadena A N
0x00200000	48 6F 6C 61 20 6D 75 6E 64 6F 00 01 00 00 00 40
	C F D
0x00200010	61 00 00 00 C1 48 00 00 C0 31 00 00 00 00 00 00
	v1
0x00200020	00 00 00 01 00 00 00 02 00 00 00 03 00 00 00 04
	v2
0x00200030	00 00 00 05 00 00 00 00 00 00 00 00 00 00 00 00
0x00200040	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00200050	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00200060	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00



Detección de datos no alineados

[ejemplo](#)

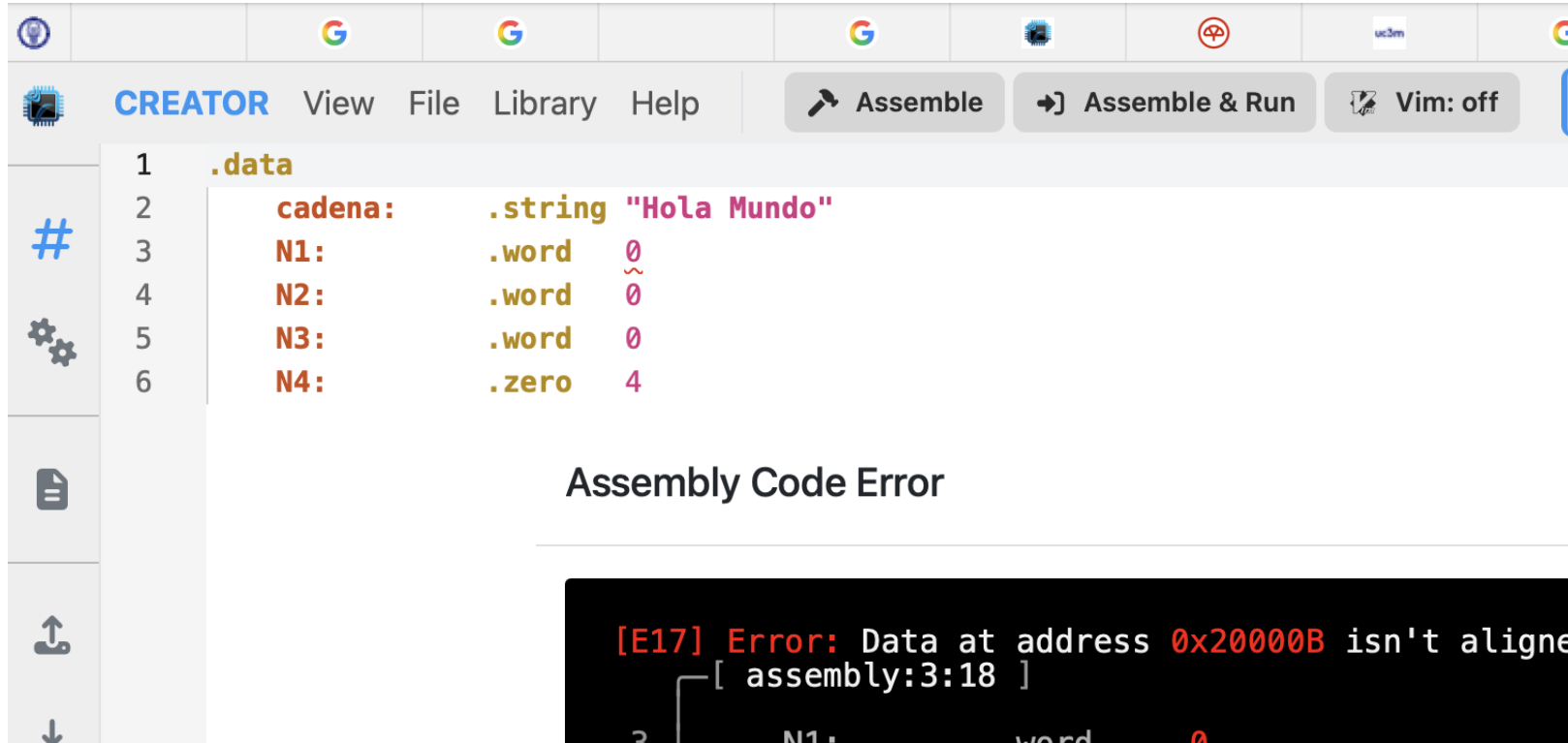


The screenshot shows the CREATOR IDE interface. The menu bar includes 'CREATOR', 'View', 'File', 'Library', and 'Help'. Below the menu bar are three buttons: 'Assemble', 'Assemble & Run', and 'Vim: off'. The main editor area displays assembly code with line numbers 1 through 6. The code defines a data section with a string and three words, followed by a zero field.

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

Detección de datos no alineados

ejemplo



The screenshot shows the CREATOR IDE interface. The main editor displays the following assembly code:

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

Below the code, an "Assembly Code Error" dialog box is open, displaying the following error message:

```
[E17] Error: Data at address 0x20000B isn't aligned to size 4 nor word size 4
[ assembly:3:18 ]
3 N1: .word 0
└─ This value isn't aligned
```

Segmento de datos corregido

[ejemplo](#)



```
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
8
9  .text
10
11 main:
12 | jr ra
```

Ejemplo. Ejecutar el siguiente programa

ejemplo

```
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, 0(t1)
12
13     li t0, 2
14     la t1, N2
15     sw t0, 0(t1)
16
17     li t0, -3
18     la t1, N3
19     sw t0, 0(t1)
20
21     li t0, 4
22     la t1, N4
23     sw t0, 0(t1)
```

Ejemplo: antes de la ejecución

Address	User Instruction	Loaded Instructions
0x0 main	li t0, 1	addi t0, x0, 1 next
0x4	la t1, N1	auipc t1, 512
0x8		addi t1, t1, -4
0xC	sw t0, 0(t1)	sw t0, 0(t1)
0x10	li t0, 2	addi t0, x0, 2
0x14	la t1, N2	auipc t1, 512
0x18		addi t1, t1, -16
0x1C	sw t0, 0(t1)	sw t0, 0(t1)
0x20	li t0, -3	addi t0, x0, -3
0x24	la t1, N3	auipc t1, 512
0x28		addi t1, t1, -28
0x2C	sw t0, 0(t1)	sw t0, 0(t1)
0x30	li t0, 4	addi t0, x0, 4
0x34	la t1, N4	auipc t1, 512
0x38		addi t1, t1, -40
0x3C	sw t0, 0(t1)	sw t0, 0(t1)

Registers
Memory
Terminal
Statistics

Go to:
→ Go
Jump to: TEXT DATA STACK
Tags
Layout

Address	Hex	ASCII
	<div style="display: flex; justify-content: space-around; font-size: small;"> N1 N2 N3 N4 </div>	
0x00200000	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00200010	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00200020	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00200030	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00200040	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00200050	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00200060	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00200070	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00200080	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00200090	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x002000A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x002000B0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x002000C0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x002000D0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x002000E0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x002000F0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00200100	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Ejemplo: después de la ejecución

Address	User Instruction	Loaded Instructions
0x0 main	li t0, 1	addi t0, x0, 1
0x4	la t1, N1	auipc t1, 512
0x8		addi t1, t1, -4
0xC	sw t0, 0(t1)	sw t0, 0(t1)
0x10	li t0, 2	addi t0, x0, 2
0x14	la t1, N2	auipc t1, 512
0x18		addi t1, t1, -16
0x1C	sw t0, 0(t1)	sw t0, 0(t1)
0x20	li t0, -3	addi t0, x0, -3
0x24	la t1, N3	auipc t1, 512
0x28		addi t1, t1, -28
0x2C	sw t0, 0(t1)	sw t0, 0(t1)
0x30	li t0, 4	addi t0, x0, 4
0x34	la t1, N4	auipc t1, 512
0x38		addi t1, t1, -40
0x3C	sw t0, 0(t1)	sw t0, 0(t1)

The screenshot shows a debugger interface with tabs for Registers, Memory, Terminal, and Statistics. The Memory tab is active, displaying a memory dump. The 'Go to' field contains 'e.g. 0x200000'. Below the tabs, there are buttons for 'Go', 'Jump to: TEXT DATA STACK', 'Tags', and 'Layout'. The memory dump table has columns for 'Address', 'Hex', and 'ASCII'. The first row of the memory dump is highlighted with a red circle. Above the first row, there are four colored boxes labeled N1 (red), N2 (blue), N3 (teal), and N4 (yellow). The first row of the memory dump shows the address 0x00200000 and the hex value 00 00 00 01 00 00 00 02 FF FF FF FD 00 00 00 04. The ASCII column for this row shows a series of dots.

Address	Hex	ASCII
0x00200000	00 00 00 01 00 00 00 02 FF FF FF FD 00 00 00 04
0x00200010	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00200020	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00200030	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00200040	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00200050	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00200060	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00200070	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00200080	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00200090	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x002000A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x002000B0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x002000C0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x002000D0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x002000E0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x002000F0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00200100	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00200110	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Visualización del segmento de texto

Instrucciones y pseudoinstrucciones

ejemplo

Address	User Instruction	Loaded Instructions
0x0	main li t0, 1	addi t0, x0, 1
0x4	la t1, N1	auipc t1, 512
0x8		addi t1, t1, -4
0xC	sw t0, 0(t1)	sw t0, 0(t1)
0x10	li t0, 2	addi t0, x0, 2
0x14	la t1, N2	auipc t1, 512
0x18		addi t1, t1, -16
0x1C	sw t0, 0(t1)	sw t0, 0(t1)
0x20	li t0, -3	addi t0, x0, -3
0x24	la t1, N3	auipc t1, 512
0x28		addi t1, t1, -28
0x2C	sw t0, 0(t1)	sw t0, 0(t1)
0x30	li t0, 4	addi t0, x0, 4
0x34	la t1, N4	auipc t1, 512
0x38		addi t1, t1, -40
0x3C	sw t0, 0(t1)	sw t0, 0(t1)
0x40	jr ra	jalr x0, 0(ra)

Registers

Memory

Terminal

Go to: → Go

Jump to: TEXT DATA STACK

Address	Hex
0x00000000	addi t0, x0, 1 10 02 93 00 20 03 17 FF C3 03 13 00 53 20 23
0x00000010	addi t0, x0, 2 00 20 02 93 00 20 03 17 FF 03 03 13 00 53 20 23
0x00000020	addi t0, x0, ... FF D0 02 93 00 20 03 17 FE 43 03 13 00 53 20 23
0x00000030	addi t0, x0, 4 00 40 02 93 00 20 03 17 FD 83 03 13 00 53 20 23
0x00000040	jalr x0, 0(ra) 00 00 80 67 00 00 00 00 00 00 00 00 00 00 00
0x00000050	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000060	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000070	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000080	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000090	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x000000A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x000000B0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x000000C0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x000000D0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

← Previous Page 1 of 167773 Next →

Programa escrito
(inst. y pseudoinst.)

Programa en memoria
(instrucciones máquina)



Calcular la longitud de una cadena

ejemplo

```
CREATOR View File Library Help Assemble Assemble & Run
1 #
2 # Creator (https://creatorsim.github.io/creator/)
3 #
4
5 .data
6     cadena: .string "Hola Mundo"
7
8
9 .text
10 main:
11
12
```

Visualización del segmento de texto

Flujo de ejecución

ejemplo

Address	User Instruction	Loaded Instructions
0x0 main	li t0, 1	addi t0, x0, 1
0x4	la t1, N1	auipc t1, 512
0x8		addi t1, t1, -4
0xC	sw t0, 0(t1)	sw t0, 0(t1)
0x10	li t0, 2	addi t0, x0, 2
0x14	la t1, N2	auipc t1, 512 next
0x18		addi t1, t1, -16
0x1C	sw t0, 0(t1)	sw t0, 0(t1)
0x20	li t0, -3	addi t0, x0, -3
0x24	la t1, N3	auipc t1, 512
0x28		addi t1, t1, -28
0x2C	sw t0, 0(t1)	sw t0, 0(t1)
0x30	li t0, 4	addi t0, x0, 4
0x34	la t1, N4	auipc t1, 512
0x38		addi t1, t1, -40
0x3C	sw t0, 0(t1)	sw t0, 0(t1)

Registers
Memory
>_ Terminal

Go to: → Go Jump to: TEXT DATA STACK Tag

Address	Hex
0x00000000	addi t0, x0, 1 auipc t1, 512 addi t1, t1, ... sw t0, 0(t1)
0x00000010	addi t0, x0, 2 auipc t1, 512 addi t1, t1, ... sw t0, 0(t1)
0x00000020	addi t0, x0, ... auipc t1, 512 addi t1, t1, ... sw t0, 0(t1)
0x00000030	addi t0, x0, 4 auipc t1, 512 addi t1, t1, ... sw t0, 0(t1)
0x00000040	jalr x0, 0(ra)
0x00000050	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000060	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000070	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000080	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000090	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x000000A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x000000B0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x000000C0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x000000D0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Visualización del segmento de texto

Varias funciones

[ejemplo](#)

```
CREATOR View File Library Help Assemble Assemble & Run Vim: off
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
16
17 jr ra
```

Visualización del segmento de texto

Varias funciones

CREATOR View Tools Help Reset Step Run Stop Sentinel Welcome to new CREATOR! For previous version

Address	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, -24 next
0x1C	jal ra, f2	jal ra, -20
0x20	jal ra, f3	jal ra, -16
0x24	jr ra	jalr x0, 0(ra)

Registers Memory Terminal

Go to: → Go Jump to: TEXT DATA STACK Ta

Address	Hex
0x00000000	addi a0, x0, 1 jalr x0, 0(ra) addi a0, x0, 2 jalr x0, 0(ra) 00 10 05 13 00 00 80 67 00 20 05 13 00 00 80 67
0x00000010	addi a0, x0, 3 jalr x0, 0(ra) jal ra, -24 jal ra, -20 00 30 05 13 00 00 80 67 FE 9F F0 EF FE DF F0 EF
0x00000020	jal ra, -16 jalr x0, 0(ra) FF 1F F0 EF 00 00 80 67
0x00000030	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000040	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000050	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000060	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000070	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000080	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000090	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x000000A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x000000B0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x000000C0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x000000D0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x000000E0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x000000F0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Llamadas al sistema

ejemplo

```
CREATOR View File Library Help Assemble Assemble & Run Vim: off
1 #
2 # Creator (https://creatorsim.github.io/creator/)
3 #
4
5 .data
6     str_text: .string "Insert a number: "
7     str_result: .string "Result = "
8
9 .text
10 main:
11     # print "Insert a number: "
12     la a0, str_text
13     li a7, 4
14     ecall
15
16     # read int
17     li a7, 5
18     ecall
19
20     mv t0, a0
21
22     # print "Insert a number: "
23     la a0, str_text
24     li a7, 4
25     ecall
26
27     # read int
28     li a7, 5
29     ecall
30     mv t1, a0
31
32     # print "Result = "
33     la a0, str_result
34     li a7, 4
35     ecall
36
37     add a0, t0, t1
38     li a7, 1
39     ecall
```

Llamadas al sistema

CREATOR View Tools Help [Reset] [Step] [Run] [Stop] Sentinel [Welcome to new CREATOR! For previous version, click ...] [Settings] [Notifications]

Address	User Instruction	Loaded Instructions
0x0 main	la a0, str_text	auipc a0, 512
0x4		addi a0, a0, 0
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 next
0x1C	la a0, str_text	auipc a0, 512
0x20		addi a0, a0, -28
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
0x38	la a0, str_result	auipc a0, 512
0x3C		addi a0, a0, -38
0x40	li a7, 4	addi a7, x0, 4

Registers Memory **Terminal** Statistics

```
Insert a number: 
```

Llamadas al sistema

The screenshot displays the CREATOR IDE interface. The top menu bar includes 'CREATOR', 'View', 'Tools', and 'Help'. Below the menu, there are control buttons for 'Reset', 'Finished', 'Stop', and 'Sentinel'. A notification bar at the top right says 'Welcome to new CREATOR! For previous version, click ...'. The main workspace is divided into two panes. The left pane shows a table of assembly instructions with columns for 'Address', 'User Instruction', and 'Loaded Instructions'. The right pane is a terminal window showing the execution output.

Address	User Instruction	Loaded Instructions
0x0 main	la a0, str_text	auipc a0, 512
0x4		addi a0, a0, 0
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, 512
0x20		addi a0, a0, -28
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
0x38	la a0, str_result	auipc a0, 512
0x3C		addi a0, a0, -38
0x40	li a7, 4	addi a7, x0, 4
0x44	ecall	ecall
0x48	add a0, t0, t1	add a0, t0, t1
0x4C	li a7, 1	addi a7, x0, 1
0x50	ecall	ecall
0x54	li a7, 10	addi a7, x0, 10
0x58	ecall	ecall
0x5C	jr ra	jalr x0, 0(ra) next

```
Insert a number: 5
Insert a number: 6
Result = 11
```

Ejemplo: ejecutar el siguiente programa

ejemplo

```
CREATOR View File Library Help Assemble Assen
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:
16         lbu   t2, 0(t0)
17         beq   t2, zero, end
18         sb    t2, 0(t1)
19         addi  t0, t0, 1
20         addi  t1, t1, 1
21         j     loop
22
23     end:
24         li   a7, 10
25         ecall
26
27         jr   ra
```

CREATOR View Tools Help Reset Step Run Stop Sentinel

Address	User Instruction	Loaded Instructions
0x0 main	la t0, string1	auipc t0, 512 next
0x4		addi t0, t0, 0
0x8	la t1, string2	auipc t1, 512
0xC		addi t1, t1, 3
0x10 loop	lbu t2, 0(t0)	lbu t2, 0(t0)
0x14	beq t2, zero, end	beq t2, zero, 20
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	beq zero, zero, -20
0x28 end	li a7, 10	addi a7, x0, 10
0x2C	ecall	ecall
0x30	jr ra	jalr x0, 0(ra)

Ejecución del programa. Bucles

CREATOR View Tools Help Reset Step Run Stop Sentinel

Address	User Instruction	Loaded Instructions
0x0 main	la t0, string1	auipc t0, 512
0x4		addi t0, t0, 0
0x8	la t1, string2	auipc t1, 512
0xC		addi t1, t1, 3
0x10 loop	lbu t2, 0(t0)	lbu t2, 0(t0) next
0x14	beq t2, zero, end	beq t2, zero, 20
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	beq zero, zero, -20
0x28 end	li a7, 10	addi a7, x0, 10
0x2C	ecall	ecall
0x30	jr ra	jalr x0, 0(ra)

Ejemplo de llamadas a funciones anidadas

ejemplo

- Comprobar el crecimiento de la pila

The screenshot shows the CREATOR IDE interface. On the left, a table displays assembly instructions with their addresses, user-written instructions, and loaded instructions. The instruction at address 0x3C, `la a0, str2`, is highlighted in blue. Below it, the instruction at address 0x40, `addi a0, a0, -57`, is highlighted in green and has a 'next' button. On the right, the 'Memory' tab is active, showing a memory dump from address 0x0FFFFFF0 to 0x0FFFFFFE. The stack grows downwards, with the current instruction at 0x0FFFFFF0 containing the value 00 00 00 28. A stack frame table below the memory dump shows the current frame for 'Callee: f2' and its caller 'f1', which is called from 'main'. The 'x1,ra' register is shown to contain the address 94 FF FF FF.

Address	User Instruction	Loaded Instructions
0x14	ecall	ecall
0x18	li a0, '\n'	addi a0, x0, 10
0x1C	li a7, 11	addi a7, x0, 11
0x20	ecall	ecall
0x24	jal ra, f2	jal ra, 16
0x28	lw ra, 0(sp)	lw ra, 0(sp)
0x2C	addi sp, sp, 4	addi sp, sp, 4
0x30	jr ra	jalr x0, 0(ra)
0x34 f2	addi sp, sp, -4	addi sp, sp, -4
0x38	sw ra, 0(sp)	sw ra, 0(sp)
0x3C	la a0, str2	auipc a0, 512
0x40	addi a0, a0, -57	addi a0, a0, -57
0x44	li a7, 4	addi a7, x0, 4
0x48	ecall	ecall
0x4C	li a0, '\n'	addi a0, x0, 10
0x50	li a7, 11	addi a7, x0, 11
0x54	ecall	ecall
0x58	jal ra, f3	jal ra, 16
0x5C	lw ra, 0(sp)	lw ra, 0(sp)
0x60	addi sp, sp, 4	addi sp, sp, 4
0x64	jr ra	jalr x0, 0(ra)

Address	Hex
0x0FFFFFF0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0FFFFFF1	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0FFFFFF2	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0FFFFFF3	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0FFFFFF4	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0FFFFFF5	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0FFFFFF6	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0FFFFFF7	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0FFFFFF8	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0FFFFFF9	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0FFFFFFA	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0FFFFFFB	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0FFFFFFC	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0FFFFFFD	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0FFFFFFE	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Callee: f2	Caller: f1	main
x1,ra	x1,ra	x1,ra
00 00 00 28	00 00 00 94	FF FF FF FF 00 00 00 00

Crecimiento de la pila en factorial

ejemplo

The screenshot displays the CREATOR IDE interface. The top menu bar includes 'CREATOR', 'View', 'Tools', 'Help', 'Reset', 'Step', 'Run', 'Stop', and 'Sentinel'. A notification bar at the top right says 'Welcome to new CREATOR! For previous version, click ... X'.

The main window is divided into three panes:

- Assembly Table:** A table with columns 'Address', 'User Instruction', and 'Loaded Instructions'. The current instruction at address 0x38 is highlighted in green: 'li x5, 2' with 'addi x5, x0, 2' loaded. A 'next' button is visible to the right of this row.
- Registers:** A pane showing register values. The 'x10, a0' register is highlighted with a value of 05.
- Memory:** A pane showing a memory dump starting at address 0x0FFFFFF0. The dump shows several zero-filled bytes. A call stack is visible below the memory dump, showing 'Caller: factorial' and 'main'.

Address	User Instruction	Loaded Instructions
0xC	jal x1, factorial	jal x1, 28
0x10	li a7, 1	addi a7, x0, 1
0x14	ecall	ecall
0x18	lw ra, 0(sp)	lw ra, 0(sp)
0x1C	addi sp, sp, 4	addi sp, sp, 4
0x20	li a7, 10	addi a7, x0, 10
0x24	ecall	ecall
0x28	addi sp, sp, -12	addi sp, sp, -12
0x2C	sw ra, 8(sp)	sw ra, 8(sp)
0x30	sw fp, 4(sp)	sw fp, 4(sp)
0x34	addi fp, sp, 4	addi fp, sp, 4
0x38	li x5, 2	addi x5, x0, 2
0x3C	bge a0, t0, b_else	bge a0, t0, 12
0x40	li a0, 1	addi a0, x0, 1
0x44	beq x0, x0, b_efs	beq x0, x0, 24
0x48	sw a0, -4(fp)	sw a0, -4(fp)
0x4C	addi a0, a0, -1	addi a0, a0, -1
0x50	jal x1, factorial	jal x1, -40
0x54	lw t1, -4(fp)	lw t1, -4(fp)
0x58	mul a0, a0, t1	mul a0, a0, t1
0x5C	lw ra, 8(sp)	lw ra, 8(sp)

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	
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 porqué 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

ejemplo

```
CREATOR View File Library Help Assemble Assemble
1 #
2 # Creator (https://creatorsim.github.io/creator/)
3 #
4 .data
5
6 .text
7     #if (a0 > a1) return a0 else return 01
8     max:      addi    sp, sp, -8
9             sw      s0, 0(sp)
10            mv      s0, a0
11            mv      s1, a1
12            bge     s0, s1, bigger
13            mv      a0, s1
14            j       end
15            bigger: mv      a0, s0
16            end:    jr      ra
17
18
19
20     main:     li      a0, 8
21            li      a1, 7
22            jal     ra, max
23
24            li      a7, 1
25            ecall
26
27            jr      ra
```

Detección de errores en el convenio de paso de parámetros

The screenshot shows the CREATOR IDE interface. At the top, there are navigation buttons: View, Tools, Help, Reset, Step, Run, and Stop. A Sentinel icon indicates 50 errors. The main area is divided into three columns: Address, User Instruction, and Loaded Instructions. The current instruction at address 0x24 is highlighted in green and labeled 'main'. To the right, there are sections for Registers, Control registers (pc, mip), Integer registers (x0|zero, x5|t0, x10|a0, x15|a5, x20|s4, x25|s9, x30|t5), and Floating point registers (f0|ft0).

Address	User Instruction	Loaded Instructions
0x0	addi sp, sp, -8	addi sp, sp, -8
0x4	sw s0, 0(sp)	sw s0, 0(sp)
0x8	mv s0, a0	addi s0, a0, 0
0xC	mv s1, a1	addi s1, a1, 0
0x10	bge s0, s1, bigger	bge s0, s1, 12
0x14	mv a0, s1	addi a0, s1, 0
0x18	j end	beq zero, zero, 8
0x1C	mv a0, s0	addi a0, s0, 0
0x20	jr ra	jalr x0, 0(ra)
0x24	li a0, 8	addi a0, x0, 8
0x28	li a1, 7	addi a1, x0, 7
0x2C	jal ra, max	jal ra, -44
0x30	li a7, 1	addi a7, x0, 1
0x34	ecall	ecall
0x38	jr ra	jalr x0, 0(ra)

The screenshot shows the Calling Convention error log. It displays 50 errors and 50 violations. The log contains five entries, each with a timestamp of 12:53:28 and a count of 1. Each entry is titled 'unknown' and contains the message 'EMPTY CALLSTACK' and 'Calling convention violation: Cannot leave function: call stack is empty'.

Calling Convention 50 errors 50 violations

- unknown 12:53:28 1
EMPTY CALLSTACK
Calling convention violation
Cannot leave function: call stack is empty
- unknown 12:53:28 1
EMPTY CALLSTACK
Calling convention violation
Cannot leave function: call stack is empty
- unknown 12:53:28 1
EMPTY CALLSTACK
Calling convention violation
Cannot leave function: call stack is empty
- unknown 12:53:28 1
EMPTY CALLSTACK
Calling convention violation
Cannot leave function: call stack is empty
- unknown 12:53:28 1
EMPTY CALLSTACK
Calling convention violation
Cannot leave function: call stack is empty

¿Cómo corregirlo?

Solución

ejemplo

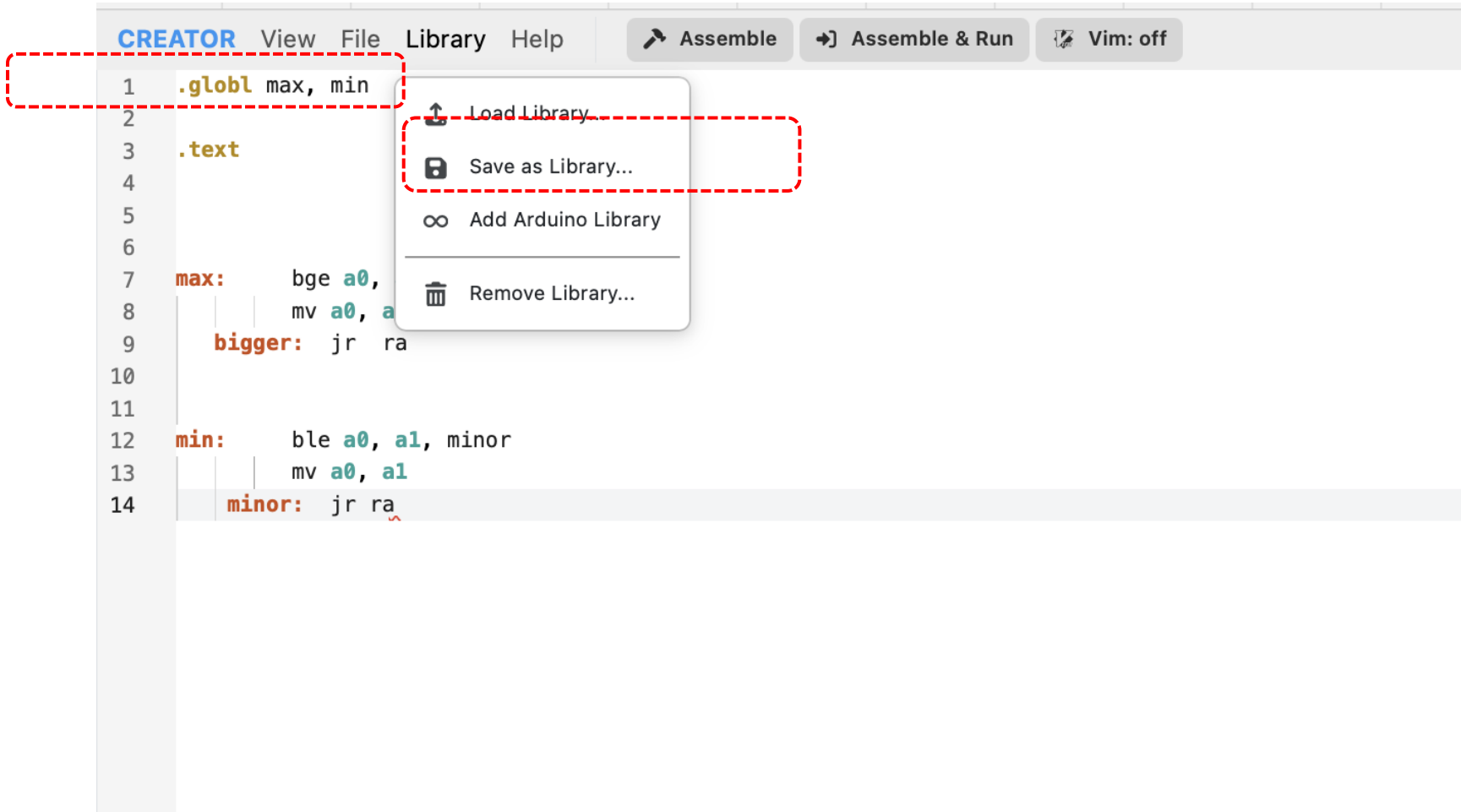
```
CREATOR View File Library Help Assemble Assemble
1 #
2 # Creator (https://creatorsim.github.io/creator/)
3 #
4 .data
5
6 .text
7     #if (a0 > a1) return a0 else return 01
8     max:      addi    sp, sp, -8
9              sw     s0, 0(sp)
10             mv     s0, a0
11             mv     s1, a1
12             bge    s0, s1, bigger
13             mv     a0, s1
14             j      end
15     bigger:  mv     a0, s0
16     end:     jr     ra
17
18
19
20     main:    li     a0, 8
21             li     a1, 7
22             jal   ra, max
23
24             li     a7, 1
25             ecall
26
27             jr    ra
```

Creación de librerías

[ejemplo](#)

```
CREATOR View File Library Help  Assemble Assemble & Run Vim: off
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
```

Creación de librerías



The screenshot shows the CREATOR IDE interface. The menu bar includes 'CREATOR', 'View', 'File', 'Library', and 'Help'. Below the menu bar are three buttons: 'Assemble', 'Assemble & Run', and 'Vim: off'. The main editor area displays assembly code with line numbers 1 through 14. A context menu is open over the code, listing four options: 'Load Library', 'Save as Library...', 'Add Arduino Library', and 'Remove Library...'. Red dashed boxes highlight the first line of code and the 'Save as Library...' option in the menu.

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

Uso de librerías

ejemplo

```
CREATOR View File Library Help Assemble Assemble & Run Vim: off
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



Uso de librerías.

ejemplo

```
CREATOR View File Library Help Assemble Assemble & Run Vim: off
1 #
2 # Creator (https://creatorsim.github.io/creator/)
3 #
4
5 .text
6
7
8
9
10
11
12
13
14
15 jal ra, max
16
17
18
19
20
21 ecall
22
23 li a0, '\n'
24 li a7, 11
25 ecall
```

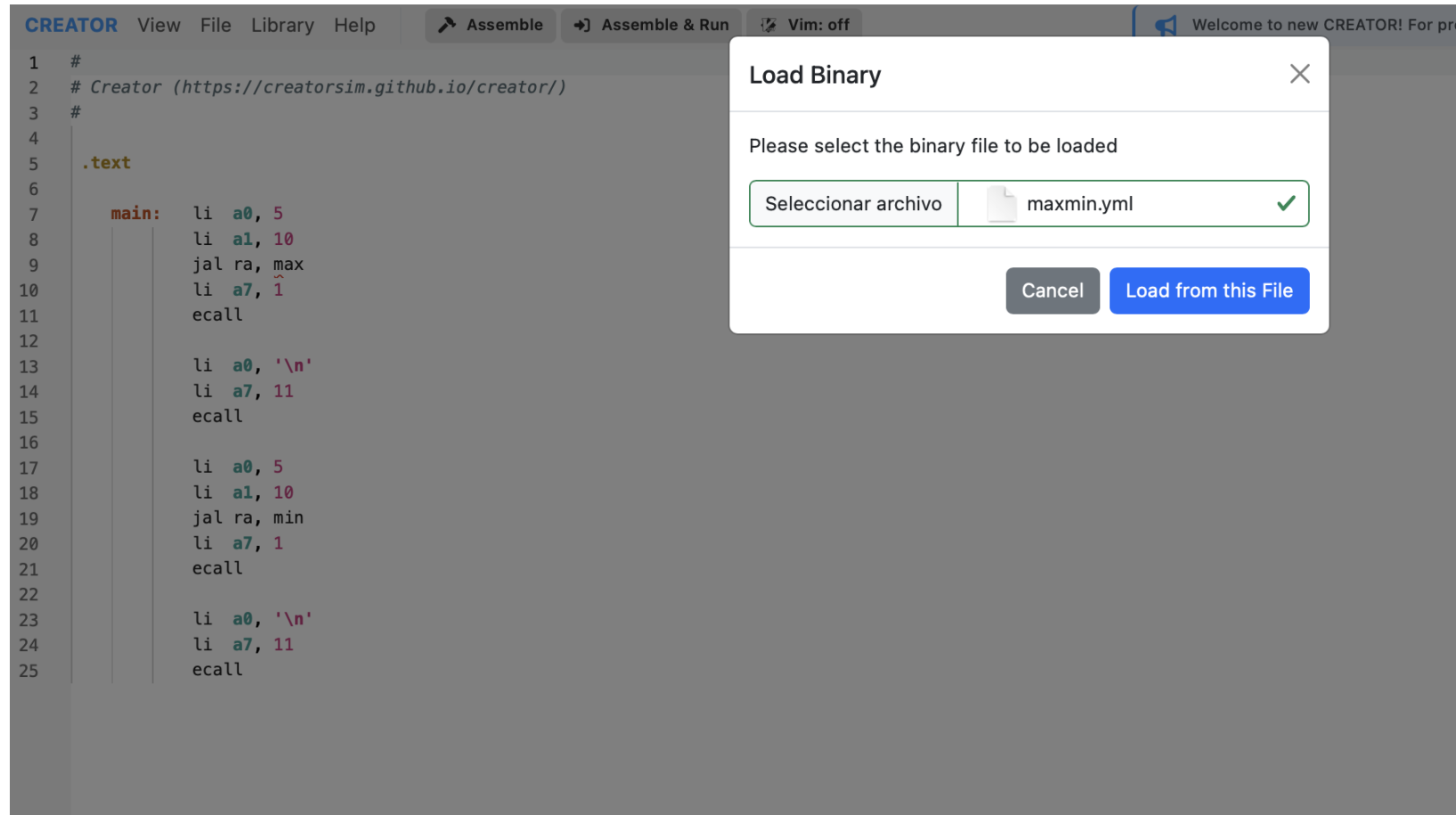
Assembly Code Error [X]

```
[E03] Error: Label max isn't defined
[ assembly:9:21 ]
9 jal ra, max
Unknown label
```

```
jal ra, 1
ecall

li a0, '\n'
li a7, 11
ecall
```

Librería cargada



The image shows a screenshot of the CREATOR IDE interface. The background displays assembly code in a text editor. The code is as follows:

```
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
```

Overlaid on the IDE is a "Load Binary" dialog box. The dialog has a title bar with a close button (X). The main text inside the dialog reads "Please select the binary file to be loaded". Below this text is a file selection field containing the text "maxmin.yml" with a green checkmark to its right. To the left of the file name is a button labeled "Seleccionar archivo". At the bottom of the dialog are two buttons: "Cancel" and "Load from this File".

Librería cargada

```
CREATOR View File Library Help Assemble Assemble & Run Vim: off Library (2) Welcome to new CREATOR! For prev
```

```
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
```

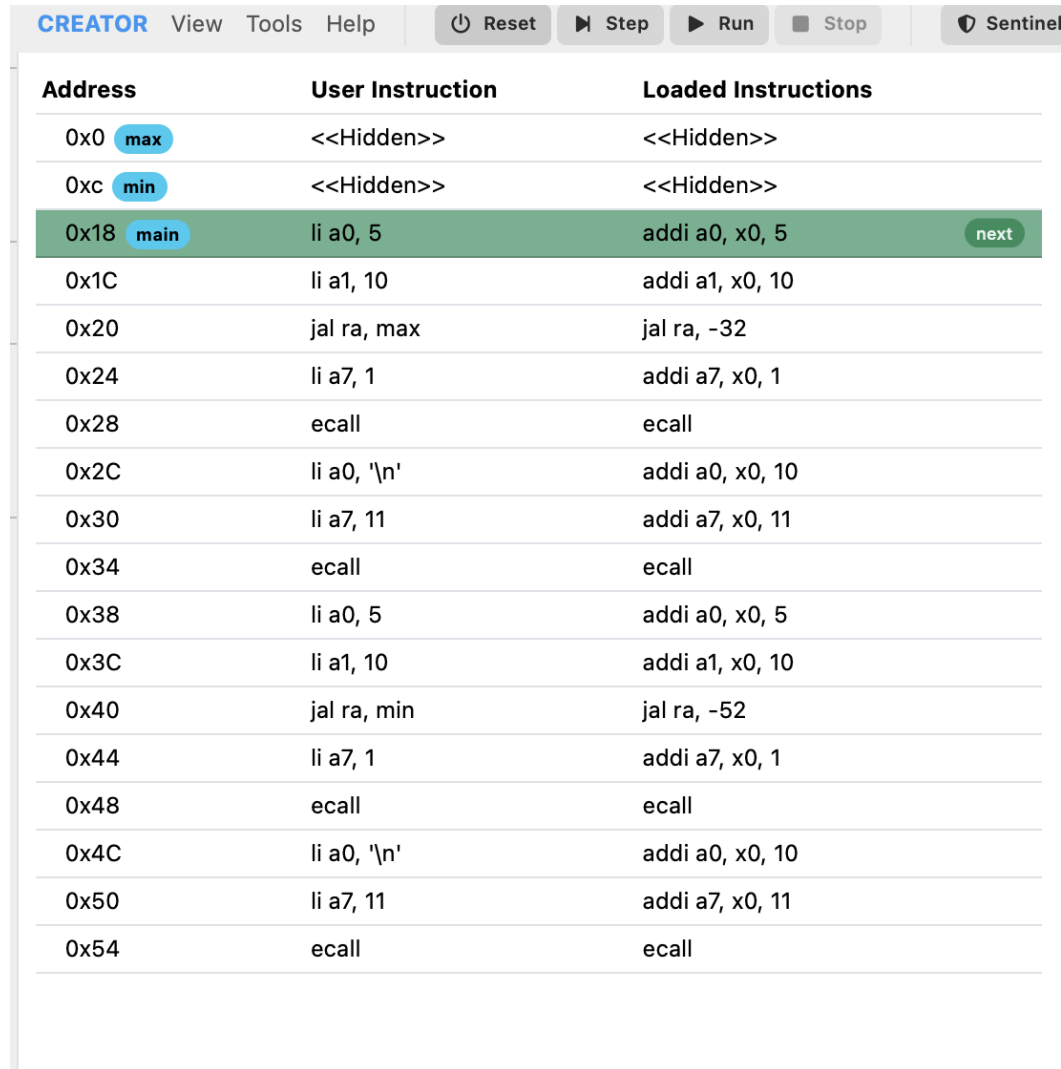
Library Tags

`max()` @0x00000000

`min()` @0x0000000c

Close

Uso de librerías: ejecutar y corregir el problema que aparece en la ejecución



CREATOR View Tools Help [Reset] [Step] [Run] [Stop] [Sentinel]

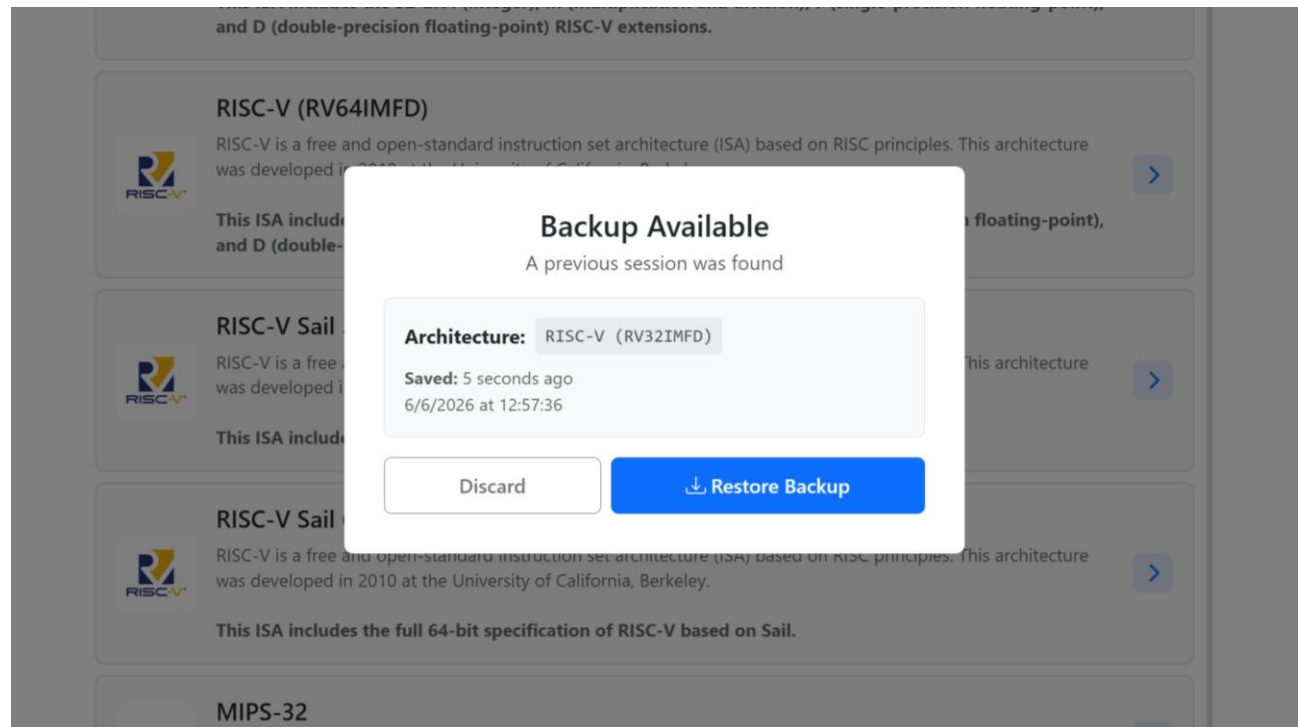
Address	User Instruction	Loaded Instructions
0x0 max	<<Hidden>>	<<Hidden>>
0xc min	<<Hidden>>	<<Hidden>>
0x18 main	li a0, 5	addi a0, x0, 5 next
0x1C	li a1, 10	addi a1, x0, 10
0x20	jal ra, max	jal ra, -32
0x24	li a7, 1	addi a7, x0, 1
0x28	ecall	ecall
0x2C	li a0, '\n'	addi a0, x0, 10
0x30	li a7, 11	addi a7, x0, 11
0x34	ecall	ecall
0x38	li a0, 5	addi a0, x0, 5
0x3C	li a1, 10	addi a1, x0, 10
0x40	jal ra, min	jal ra, -52
0x44	li a7, 1	addi a7, x0, 1
0x48	ecall	ecall
0x4C	li a0, '\n'	addi a0, x0, 10
0x50	li a7, 11	addi a7, x0, 11
0x54	ecall	ecall

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



Bloque 1

- ▶ ¿Qué es CREATOR?
 - ▶ El origen de CREATOR
 - ▶ Características de CREATOR
 - ▶ ¿Dónde se usa CREATOR?
 - ▶ Extensiones futuras
- ▶ ¿Cómo utilizar CREATOR? Visión del Estudiante
 - ▶ Edición y compilación de programas
 - ▶ Ejecución y depuración de programas
 - ▶ Convenio de paso de parámetros y uso de pila (Sentinel)
 - ▶ Bibliotecas de funciones
- ▶ **¿Cómo utilizar CREATOR? Visión del Profesor**
 - ▶ Soporte para la creación de material didáctico
 - ▶ Soporte para la corrección de prácticas
 - ▶ Soporte para la creación y edición de arquitecturas

Soporte a la corrección de prácticas: interfaz en línea de comandos

- ▶ Ejecución en línea de comandos
 - ▶ <https://creatorsim.github.io/creator-wiki/cli/overview.html>
- ▶ Prerrequisitos: linux, node.js, npm, bun, deno, cargo y rust

```
curl -fsSL https://bun.sh/install | bash
curl -fsSL https://deno.land/install.sh | sh
curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh
```

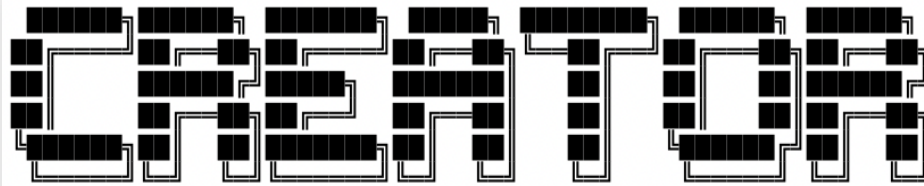
Soporte a la corrección de prácticas: interfaz en línea de comandos

► Pasos:

```
git clone --recurse-submodules https://github.com/creatorsim/creator.git
cd creator
bun install
bun dev:wasm
bun build:cli
./creator-cli -help
```

Ejecución

- ▶ `./creator-cli --architecture architecture/RISCV/RV32IMFD.yml --assembly examples/RISCV-32/example1.s`



Type 'help' for available commands.

[CREATOR> list

B	Address	Label	Loaded Instruction	User Instruction
>	0x0	main	auipc a0, 512	la a0, byte
	0x4		addi a0, a0, 0	
	0x8		lb a0, 0(a0)	lb a0, 0(a0)
	0xC		addi a7, x0, 1	li a7, 1
	0x10		ecall	ecall
	0x14		auipc a0, 512	la a0, half
	0x18		addi a0, a0, -18	
	0x1C		lh a0, 0(a0)	lh a0, 0(a0)
	0x20		addi a7, x0, 1	li a7, 1
	0x24		ecall	ecall
	0x28		addi a7, x0, 10	li a7, 10
	0x2C		ecall	ecall

CREATOR> █

Comandos

Available commands:

```
step
unstep
run [n]
silent [n]
break [addr]
reg list
reg <type>
reg <name>
mem <address> [count]
list
hexview <address> [count] [bytesPerLine]
reset
snapshot [filename]
restore <filename>
stack
clear
help
alias
about
quit
```

- Execute one instruction
- Undo last instruction
- Run n instructions or until program completes
- Run silently
- Set/unset breakpoint at address or list all
- List available register types
- Display registers of type
- Display specific register
- Display memory (count in bytes)
- Show all loaded instructions
- Hex viewer
- Reset program to initial state
- Save a complete snapshot of current state
- Restore a previously saved snapshot
- Display call stack hierarchy and frame info
- Clear the console screen
- Show this help message
- Show available aliases
- Show information about the simulator
- Quit the simulator

Visualización de registros

```
CREATOR> reg int
```

```
Integer registers:
```

```
x0(zero) : 0x00000000 x1(ra) : 0xffffffff x2(sp) : 0x0ffffffc x3(gp) : 0x00000000
x4(tp) : 0x00000000 x5(t0) : 0x00000000 x6(t1) : 0x00000000 x7(t2) : 0x00000000
x8(fp,s0) : 0x00000000 x9(s1) : 0x00000000 x10(a0) : 0x00000000 x11(a1) : 0x00000000
x12(a2) : 0x00000000 x13(a3) : 0x00000000 x14(a4) : 0x00000000 x15(a5) : 0x00000000
x16(a6) : 0x00000000 x17(a7) : 0x00000000 x18(s2) : 0x00000000 x19(s3) : 0x00000000
x20(s4) : 0x00000000 x21(s5) : 0x00000000 x22(s6) : 0x00000000 x23(s7) : 0x00000000
x24(s8) : 0x00000000 x25(s9) : 0x00000000 x26(s10) : 0x00000000 x27(s11) : 0x00000000
x28(t3) : 0x00000000 x29(t4) : 0x00000000 x30(t5) : 0x00000000 x31(t6) : 0x00000000
```

```
type help for available commands.
```

```
CREATOR> reg int decimal
```

```
Integer registers:
```

```
x0(zero) : 0 x1(ra) : 4294967295 x2(sp) : 268435452 x3(gp) : 0
x4(tp) : 0 x5(t0) : 0 x6(t1) : 0 x7(t2) : 0
x8(fp,s0) : 0 x9(s1) : 0 x10(a0) : 12 x11(a1) : 0
x12(a2) : 0 x13(a3) : 0 x14(a4) : 0 x15(a5) : 0
x16(a6) : 0 x17(a7) : 0 x18(s2) : 0 x19(s3) : 0
x20(s4) : 0 x21(s5) : 0 x22(s6) : 0 x23(s7) : 0
x24(s8) : 0 x25(s9) : 0 x26(s10) : 0 x27(s11) : 0
x28(t3) : 0 x29(t4) : 0 x30(t5) : 0 x31(t6) : 0
```

Visualización de la memoria

```
[CREATOR> hexview 0x0 200
      00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f | ASCII
-----|-----
0x0000: 00 20 05 17 00 05 05 13 00 05 05 03 00 10 08 93 | . . . . .
0x0010: 00 00 00 73 00 20 05 17 fe e5 05 13 00 05 15 03 | . . . S . . . . .
0x0020: 00 10 08 93 00 00 00 73 00 a0 08 93 00 00 00 73 | . . . . . S . . . . . S
0x0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | . . . . .
0x0040: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | . . . . .
0x0050: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | . . . . .
0x0060: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | . . . . .
0x0070: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | . . . . .
0x0080: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | . . . . .
0x0090: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | . . . . .
0x00a0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | . . . . .
0x00b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | . . . . .
0x00c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | . . . . .
```

Carga de una librería

► Ejemplo:

```
creator-cli -a riscv32.yml -l stdlib.yml -s program.s
```

Validación de un programa

- ▶ Documentación:

<https://creatorsim.github.io/creator-wiki/teaching-resources/validator.html>

- ▶ Ejemplo:

```
./creator-cli --architecture architecture/RISCV/RV32IMFD.yml \  
  --assembly examples/RISCV-32/factorial.s \  
  --validate output.yml
```

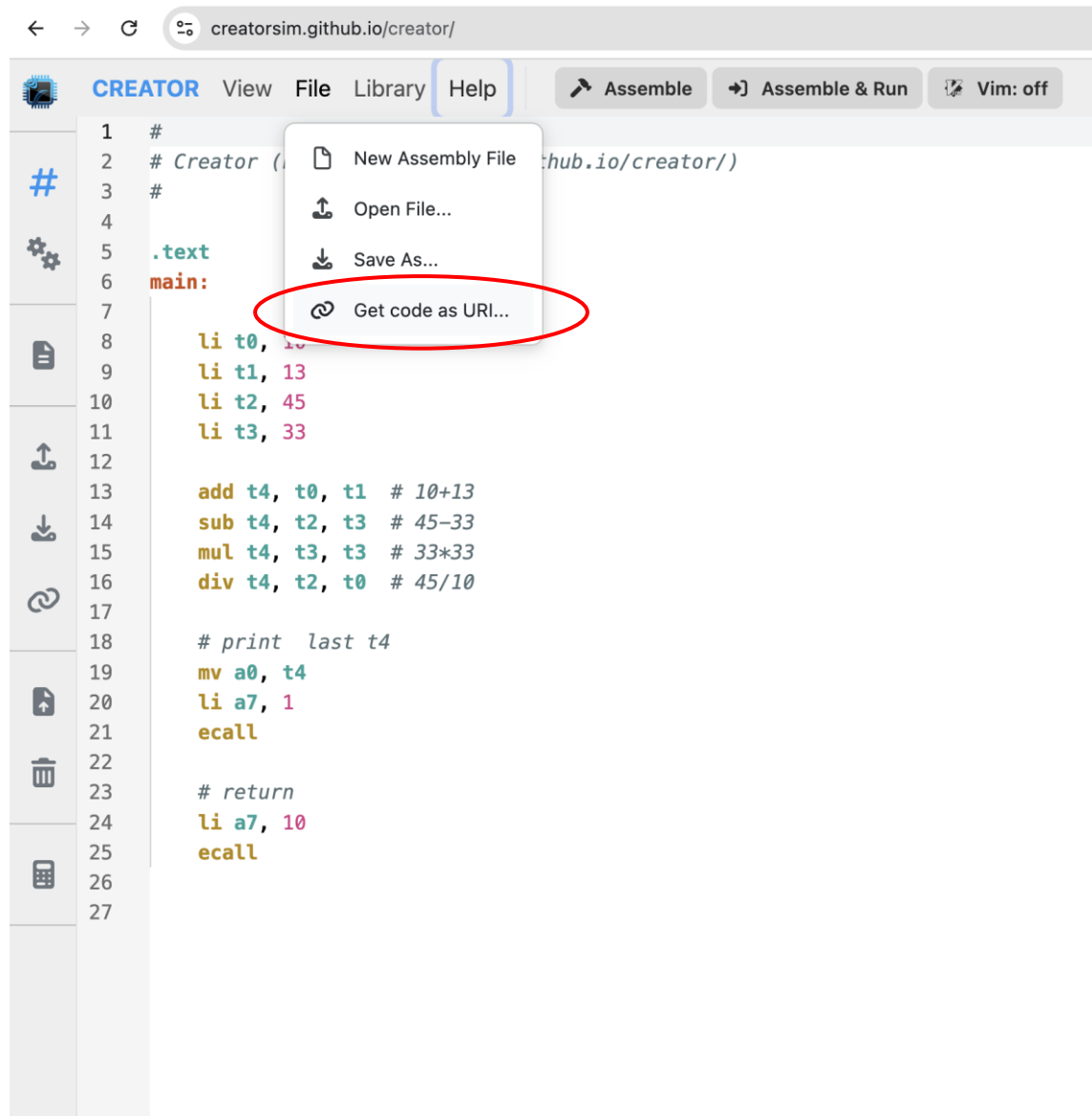
- ▶ “output.yml” comprueba la salida:

```
maxCycles: 100  
sentinel: true  
state:  
  registers:  
    a1: 120  
  memory:  
    "0x200000": 120  
  display: "120"
```

Ejemplo

- ▶ Modificar el factorial para que deje el resultado en el registro a l
- ▶ Modificar el fichero de validación para comprobar el resultado de $\text{factorial}(8) = 40320$ y verificar que el resultado correcto se encuentra en a l

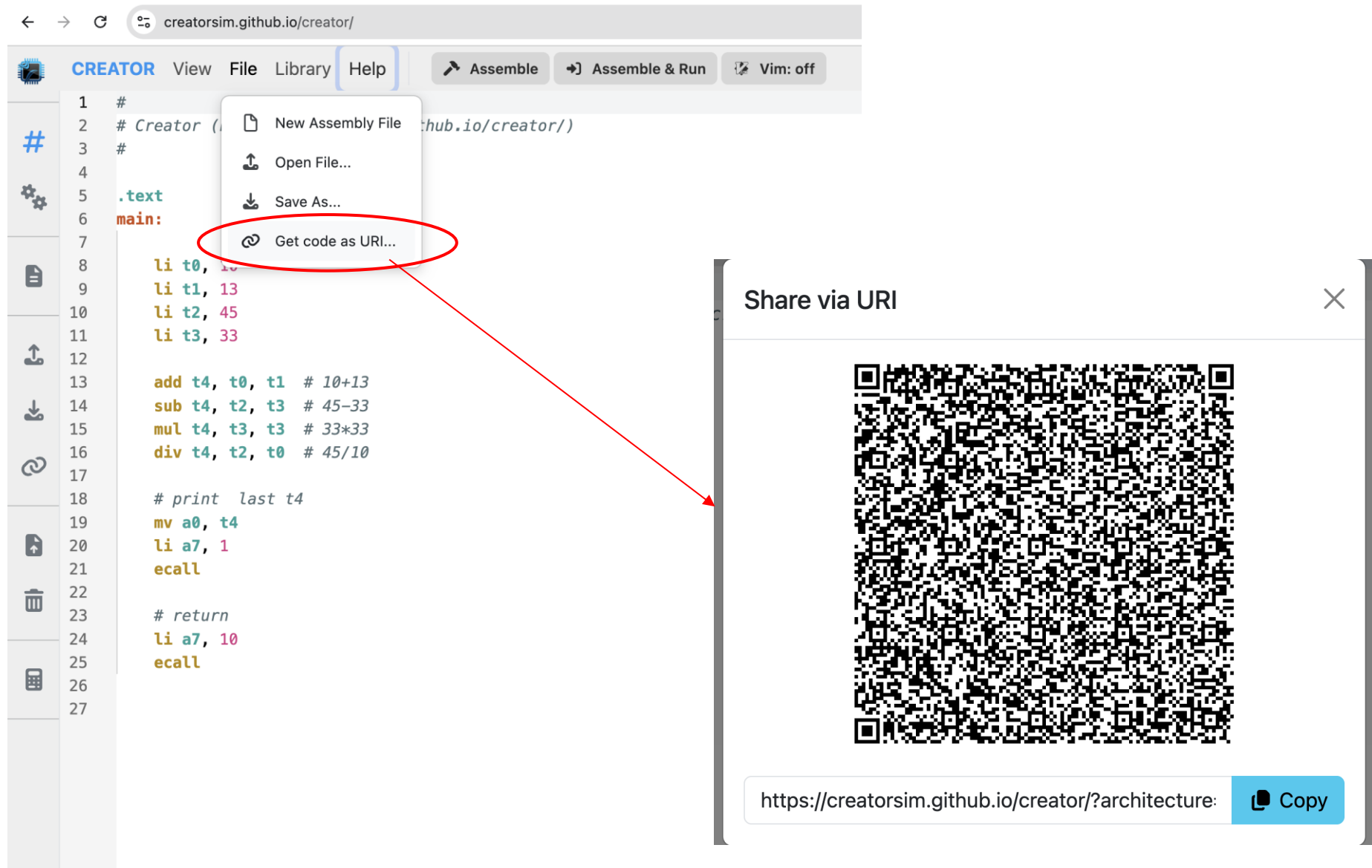
Ayuda a la creación de materiales docentes



The screenshot shows the CREATOR web IDE interface. The browser address bar displays `creatorsim.github.io/creator/`. The interface includes a menu bar with 'View', 'File', 'Library', and 'Help'. Below the menu bar are buttons for 'Assemble', 'Assemble & Run', and 'Vim: off'. The main editor area shows assembly code with line numbers 1 through 27. A context menu is open over the code, with the option 'Get code as URI...' circled in red. The code in the editor is as follows:

```
1 #
2 # Creator (
3 #
4
5 .text
6 main:
7
8 li t0, 10
9 li t1, 13
10 li t2, 45
11 li t3, 33
12
13 add t4, t0, t1 # 10+13
14 sub t4, t2, t3 # 45-33
15 mul t4, t3, t3 # 33*33
16 div t4, t2, t0 # 45/10
17
18 # print last t4
19 mv a0, t4
20 li a7, 1
21 ecall
22
23 # return
24 li a7, 10
25 ecall
26
27
```

Ayuda a la creación de materiales docentes

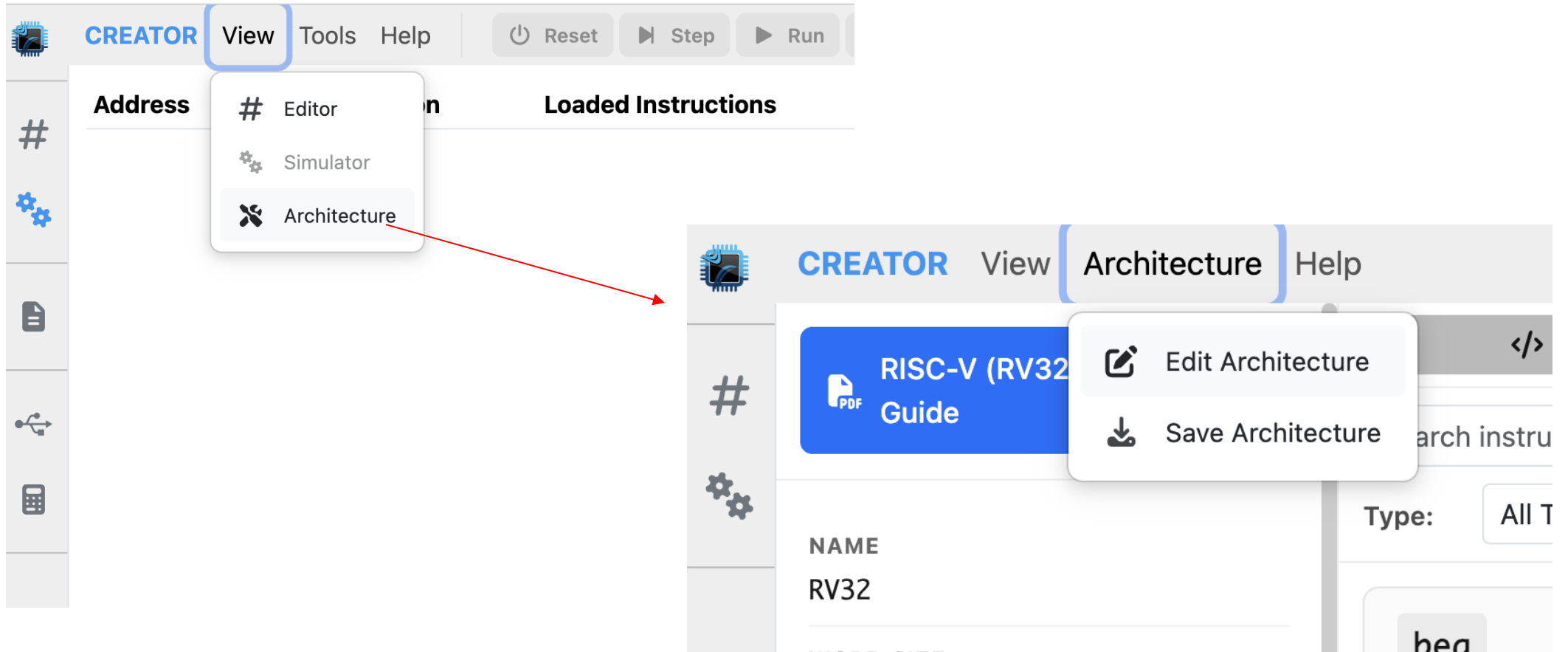


The image shows a screenshot of the CREATOR web interface. The browser address bar displays `creatorsim.github.io/creator/`. The interface includes a menu bar with 'View', 'File', 'Library', and 'Help'. The 'File' menu is open, showing options: 'New Assembly File', 'Open File...', 'Save As...', and 'Get code as URI...'. The 'Get code as URI...' option is circled in red. A red arrow points from this option to a 'Share via URI' dialog box. The dialog box contains a large QR code and a text input field with the URL `https://creatorsim.github.io/creator/?architecture:` and a 'Copy' button.

```
1 #
2 # Creator (
3 #
4
5 .text
6 main:
7
8 li t0, 10
9 li t1, 13
10 li t2, 45
11 li t3, 33
12
13 add t4, t0, t1 # 10+13
14 sub t4, t2, t3 # 45-33
15 mul t4, t3, t3 # 33*33
16 div t4, t2, t0 # 45/10
17
18 # print last t4
19 mv a0, t4
20 li a7, 1
21 ecall
22
23 # return
24 li a7, 10
25 ecall
26
27
```

Capacidades para extender el juego de instrucciones y crear nuevas arquitecturas

- ▶ Arquitectura definida en formato YAML



Edición de la arquitectura

Edit Architecture



```
1 version: 2.0.0
2
3 # ASCII text generated w/ https://patorjk.com/software/taag/#p=display&h=1&f=Standard
4 # instruction help text extracted from https://msyksphinz-self.github.io/riscv-isadoc/html/index.html
5
6 config:
7   name: RV32
8   word_size: 32
9   byte_size: 8
10  description: RISC-V is a free and open standard ISA (Instruction Set Architecture) based on RISC (Reduced Instruction
11  Set Computer) principles. It was created in 2010 at the University of California, Berkeley.
12  endianness: big_endian
13  memory_alignment: true
14  main_function: main
15  passing_convention: true
16  sensitive_register_name: true
17  comment_prefix: "#"
18  start_address: 0x0
19  pc_offset: -4
20  plugin: riscv
21  assemblers:
22    - name: CreatorAssembler
23      description: "Default CREATOR assembler"
24
25  extensions:
26    I:
27      description: "RV32I Base Instruction Set"
28      template: Base
29
30  M:
```

Cancel

Save

Ejercicio propuesto

- ▶ Añadir una nueva pseudoinstrucción:

- ▶ `psdli rd1, rd2, inm` \rightarrow `rd1 = inm` y `rd2 = inm`

- ▶ Añadir una nueva instrucción:

- ▶ `ldli rd1, rd2, inm` \rightarrow `rd1 = inm` y `rd2 = inm`

- ▶ Código de operación: `IIIIIIII`

- ▶ Pasos:

- ▶ Descargar el archivo YAML con la arquitectura
 - ▶ Editar y entender el archivo con la arquitectura
 - ▶ Modificar la arquitectura incorporando `psdli` y `ldli`
 - ▶ Cargar la nueva arquitectura

Carga de una nueva arquitectura

- RISC-V (RV32IMFD)**
RISC-V is a free and open-standard instruction set architecture (ISA) based on RISC principles. This architecture was developed in 2010 at the University of California, Berkeley.
This ISA includes the 32-bit I (integer), M (multiplication and division), F (single-precision floating-point), and D (double-precision floating-point) RISC-V extensions.
- RISC-V (RV64IMFD)**
RISC-V is a free and open-standard instruction set architecture (ISA) based on RISC principles. This architecture was developed in 2010 at the University of California, Berkeley.
This ISA includes the 64-bit I (integer), M (multiplication and division), F (single-precision floating-point), and D (double-precision floating-point) RISC-V extensions.
- RISC-V Sail 32 - Full Specification**
RISC-V is a free and open-standard instruction set architecture (ISA) based on RISC principles. This architecture was developed in 2010 at the University of California, Berkeley.
This ISA includes the full 32-bit specification of RISC-V based on Sail.
- RISC-V Sail 64 - Full Specification**
RISC-V is a free and open-standard instruction set architecture (ISA) based on RISC principles. This architecture was developed in 2010 at the University of California, Berkeley.
This ISA includes the full 64-bit specification of RISC-V based on Sail.
- 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.
- Z80**
The Z80 is an 8-bit microprocessor that was designed by Zilog and introduced in 1976. It became very popular in the late 1970s and early 1980s, especially in home computers and embedded systems.
- Load Custom Architecture**
Import your own architecture definition file (.yaml)

Load Custom Architecture

Architecture Name

Description

Architecture File

Seleccionar archivo Ningún archivo seleccionado

Cancel OK

Nueva instrucción

CREATOR View Architecture Help

Welcome to new CREATOR! For previous version, click here.

#

RV32

WORD SIZE 32

BYTE SIZE 8

DESCRIPTION
RISC-V is a free and open standard ISA (Instruction Set Architecture) based on RISC (Reduced Instruction Set Computer) principles. It was created in 2010 at the University of California, Berkeley.

ENDIANNESS
Big Endian

MEMORY ALIGNMENT
Enabled

MAIN FUNCTION
main

PASSING CONVENTION
Enabled

SENSITIVE REGISTER NAME
Enabled

COMMENT PREFIX
#

START ADDRESS
0

PC OFFSET
-4

riscv

</> Instructions Pseudoinstructions Assembler Directives Registers

Search instructions... 130 / 130

Type: All Types

bltu WORDS: 1 CYCLES: 1

SYNTAX
bltu rs1, rs2, imm

DESCRIPTION
Take the branch if registers rs1 is less than rs2, using unsigned comparison.

31 0
i... imm [3/4] rs2 rs1 funct3 imm [4/4] i... opcode
1 6 5 5 3 4 1 7
110 1100011

bne WORDS: 1 CYCLES: 1

SYNTAX
bne rs1, rs2, imm

DESCRIPTION
Take the branch if registers rs1 and rs2 are not equal.

31 0
i... imm [3/4] rs2 rs1 funct3 imm [4/4] i... opcode
1 6 5 5 3 4 1 7
001 1100011

addi WORDS: 1 CYCLES: 1

SYNTAX
addi rd, rs1, imm

DESCRIPTION
Adds the sign-extended 12-bit immediate to register rs1. Arithmetic overflow is ignored and the result is simply the low XLEN bits of the result.

31 0
imm rs1 funct3 rd opcode
12 5 3 5 7
000 0010011

idli WORDS: 1 CYCLES: 1

SYNTAX
idli rd, rs1, imm

DESCRIPTION
Adds the sign-extended 12-bit immediate to register rs1. Arithmetic overflow is ignored and the result is simply the low XLEN bits of the result.

31 0
imm rs1 funct3 rd opcode
12 5 3 5 7
000 1111111

Nueva pseudoinstrucción

The screenshot shows the CREATOR software interface. The main window is titled "Welcome to new CREATOR! For previous version, click here." and has tabs for "Instructions", "Pseudoinstructions", "Assembler Directives", and "Registers". The "Pseudoinstructions" tab is active, showing a search bar and a list of instructions. The "psdli" instruction is highlighted with a red oval. The interface also includes a sidebar with properties for RV32, such as "WORD SIZE 32", "BYTE SIZE 8", "DESCRIPTION", "ENDIANNESS Big Endian", "MEMORY ALIGNMENT Enabled", "MAIN FUNCTION main", "PASSING CONVENTION Enabled", "SENSITIVE REGISTER NAME Enabled", and "COMMENT PREFIX #".

NAME	WORD SIZE	BYTE SIZE	DESCRIPTION	ENDIANNESS	MEMORY ALIGNMENT	MAIN FUNCTION	PASSING CONVENTION	SENSITIVE REGISTER NAME	COMMENT PREFIX
RV32	32	8	RISC-V is a free and open standard ISA (Instruction Set Architecture) based on RISC (Reduced Instruction Set Computer) principles. It was created in 2010 at the University of California, Berkeley.	Big Endian	Enabled	main	Enabled	Enabled	#

Instruction	Assembly Code	C-like Pseudocode
li	li rd, val	<pre>no_ret_op{ tmp = Field.2.(31,0).int; tmp_low = tmp & 0x00000FFF; tmp_low -= tmp_low > 0x7FF ? 0x1000 : 0; tmp_hi = (tmp - tmp_low) >>> 12; }; if (tmp_hi == 0) { addi rd, x0, op{tmp_low}; } else { lui rd, op{tmp_hi}; addi rd, rd, op{tmp_low}; };</pre>
psdli	psdli rd, rs, val	<pre>no_ret_op{ tmp = Field.3.(31,0).int; tmp_low = tmp & 0x00000FFF; tmp_low -= tmp_low > 0x7FF ? 0x1000 : 0; tmp_hi = (tmp - tmp_low) >>> 12; }; if (tmp_hi == 0) { addi rd, x0, op{tmp_low}; addi rs, x0, op{tmp_low}; } else { lui rd, op{tmp_hi}; addi rd, rd, op{tmp_low}; lui rs, op{tmp_hi}; addi rs, rs, op{tmp_low}; };</pre>

Programa con el nuevo juego de instrucciones

```
CREATOR View File Library Help Assemble Assemble & Run Vim: off
1 .text
2
3 main:
4
5     psdli    t1, t2, 5
6     idli     a1, a2, 10
7
8     jr ra
```

CREATOR View Tools Help Reset Step Run Stop Sent

Address	User Instruction	Loaded Instructions
0x0 main	psdli t1, t2, 5	addi t1, x0, 5 next
0x4		addi t2, x0, 5
0x8	idli a1, a2, 10	idli a1, a2, 10
0xC	jr ra	jalr x0, 0(ra)

Bloque 2

- ▶ **Dispositivos y soporte de interrupciones**
- ▶ Integración de CREATOR con hardware real
 - ▶ Microcontroladores
 - ▶ Placas SBC
 - ▶ Servicio de laboratorio remoto
 - ▶ Integración con Arduino (CREATino)

Dispositivos en CREATOR

- ▶ **Características principales:**
 - ▶ Están mapeados en memoria
 - ▶ Usan rangos de direcciones específicos
 - ▶ Tienen registros de control y estado
 - ▶ Proporcionan búferes de datos
 - ▶ Se pueden activar o desactivar
 - ▶ Mantienen su propio estado
- ▶ **Dispositivos incluidos:**
 - ▶ Dispositivo de consola (pantalla + teclado) y dispositivo de sistema
 - ▶ Gestionan llamadas al sistema (`ecall/syscall`)

(1/2) Consola

- ▶ **Mapa de memoria:**
 - ▶ 0xF0000000: registro de control
 - ▶ 0xF0000004: registro de estado
 - ▶ 0xF0000008-0xF000000F: datos (8 bytes)

- ▶ **Llamadas al sistema:**
 - ▶ 1: imprimir entero
 - ▶ 2: imprimir decimal 32-bits
 - ▶ 3: imprimir decimal 64-bits
 - ▶ 4: imprimir cadena de caracteres
 - ▶ 5: leer entero
 - ▶ 6: leer decimal 32-bits
 - ▶ 7: leer decimal 64-bits
 - ▶ 8: leer cadena de caracteres
 - ▶ 11: imprimir carácter
 - ▶ 12: leer carácter

(1/2) Consola

- ▶ **Mapa de memoria:**
 - ▶ 0xF0000000: registro de control
 - ▶ 0xF0000004: registro de estado
 - ▶ 0xF0000008-0xF000000F: datos (8 bytes)
- ▶ **Llamadas al sistema:**
 - ▶ 1: imprimir entero
 - ▶ 2: imprimir decimal 32-bits
 - ▶ 3: imprimir decimal 64-bits
 - ▶ 4: imprimir cadena de caracteres
 - ▶ 5: leer entero
 - ▶ 6: leer decimal 32-bits
 - ▶ 7: leer decimal 64-bits
 - ▶ 8: leer cadena de caracteres
 - ▶ 11: imprimir carácter
 - ▶ 12: leer carácter

```
.data
    string: .string "Hello string!"

.text
main:
    # t0 <- console data addr
    la t0, 0xF0000008
    # t1 <- console ctrl addr
    la t1, 0xF0000000

    # write 105
    li t2, 105
    sw t2, 0(t0) # [console data] <- 105
    li t2, 1
    sw t2, 0(t1) # [console ctrl] <- write integer

    # write '\n'
    li t2, '\n'
    sw t2, 0(t0) # [console data] <- '\n'
    li t2, 11
    sw t2, 0(t1) # [console ctrl] <- write char

    # write string
    la t2, string
    sw t2, 0(t0) # [console data] <- starting address
    li t2, 4
    sw t2, 0(t1) # [console ctrl] <- write string

    jr ra
```

(2/2) Dispositivo de sistema

- ▶ Mapa de memoria:
 - ▶ 0xF0000010: registro de control
 - ▶ 0xF0000014: registro de estado
 - ▶ 0xF0000018-0xF000001F: datos (8 bytes)
- ▶ Llamadas al sistema:
 - ▶ 9: sbrk (pedir memoria del *heap*)
 - ▶ 10: exit (terminar ejecución)

(2/2) Dispositivo de sistema

- ▶ Mapa de memoria:
 - ▶ 0xF0000010: registro de control
 - ▶ 0xF0000014: registro de estado
 - ▶ 0xF0000018-0xF000001F: datos (8 bytes)
- ▶ Llamadas al sistema:
 - ▶ 9: sbrk (pedir memoria del *heap*)
 - ▶ 10: exit (terminar ejecución)

```
.text
main:
    # t0 <- console status addr
    la t0, 0xF0000014
    # t1 <- console ctrl addr
    la t1, 0xF0000010

    # exit
    li t2, 10
    sw t2, 0(t1) # [ctrl] <- exit

    # exit
    li a7, 10
    ecall

    # return
    jr ra
```

Ciclo de vida del dispositivo

- ▶ **Inicialización**
 - ▶ Registro en el mapa de dispositivos de CREATOR
- ▶ **Comprobación de dirección**
 - ▶ En cada acceso a memoria se comprueba si es una dirección de un dispositivo
- ▶ **Invocación del manejador**
 - ▶ En cada ciclo se “despierta” para procesar peticiones pendientes
- ▶ **Reinicio**
 - ▶ Se puede reiniciar los dispositivos a su estado inicial

Modos de ejecución en CREATOR

- ▶ **Modos incluidos:**
 - ▶ Modo usuario/a (no privilegiado)
 - ▶ Modo kernel (privilegiado)
- ▶ **Características principales:**
 - ▶ Posibilidad de definir instrucciones en modo privilegiado (`privileged: true`)
 - ▶ Antes de ejecutar una instrucción CREATOR comprueba
 - Si en modo usuario/a y instrucción privilegiada -> excepción violación privilegio

Modos de ejecución en CREATOR

- ▶ **Modos incluidos:**
 - ▶ Modo usuario/a (no privilegiado)
 - ▶ Modo kernel (privilegiado)
- ▶ **Transiciones:**
 - ▶ Usuario a kernel (*trap* por interrupción, excepción o llamada al sistema)
 - ▶ Kernel a usuario/a (retorno de *trap*)

Usuario a kernel (interrupción, excepción o llamada al sistema)

interrupts:

handlers:

...

custom: |

// Disable interrupts

CAPI.INTERRUPTS.globalDisable() ;

// Switch to kernel mode

CAPI.INTERRUPTS.setKernelMode() ;

// get PC (pointing to the next instruction)

registers.mepc = CAPI.REG.read("pc") ;

// jump to associated handler assembly subroutine

if (registers.mtvec & 1n) { // vectored mode

 registers.pc = (registers.mtvec >> 2n) + 4 * (registers.mcause & (2 ** 32 - 1)) ;

} else { // direct mode

 registers.pc = (registers.mtvec >> 2n) ;

}

Kernel a usuario (retorno de *trap*)

```
- name: "mret"  
privileged: true  
definition: |  
    // restore PC  
    registers.pc = registers.mepc ;  
  
    // Switch to kernel mode  
    CAPI.INTERRUPTS.setUserMode() ;  
  
    // Enable interrupts  
    CAPI.INTERRUPTS.globalClear();  
    CAPI.INTERRUPTS.globalEnable() ;  
  
    // Some pending clean-up for UI  
    CAPI.INTERRUPTS.clearHighlight() ;  
    CAPI.STACK.endFrame() ;
```

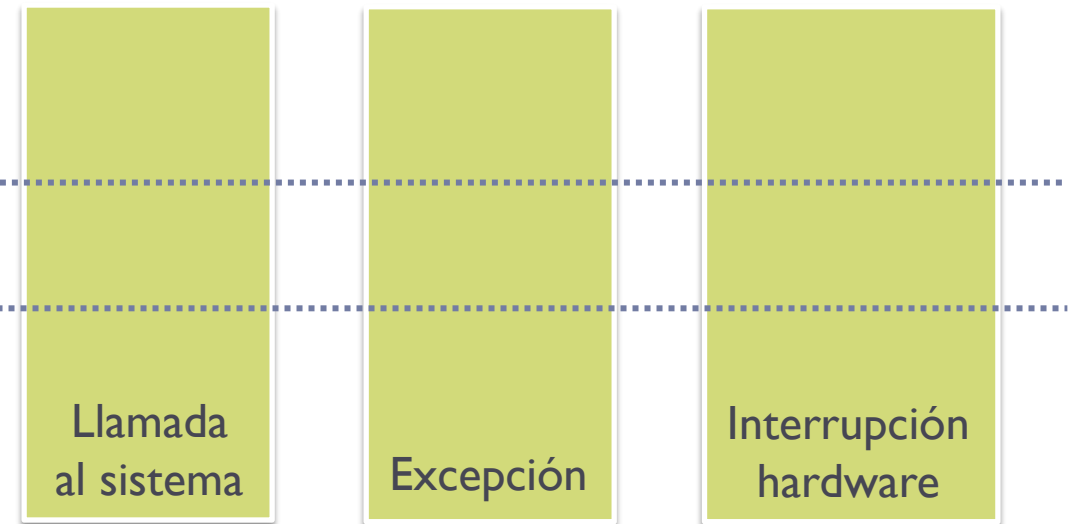
Gestión de eventos en CREATOR

- ▶ Modos incluidos:

- ▶ Modo usuario/a (no privilegiado)
- ▶ Modo kernel (privilegiado)

- ▶ Transiciones:

- ▶ Usuario a kernel (*trap*)
- ▶ Kernel a usuario/a (retorno de *trap*)



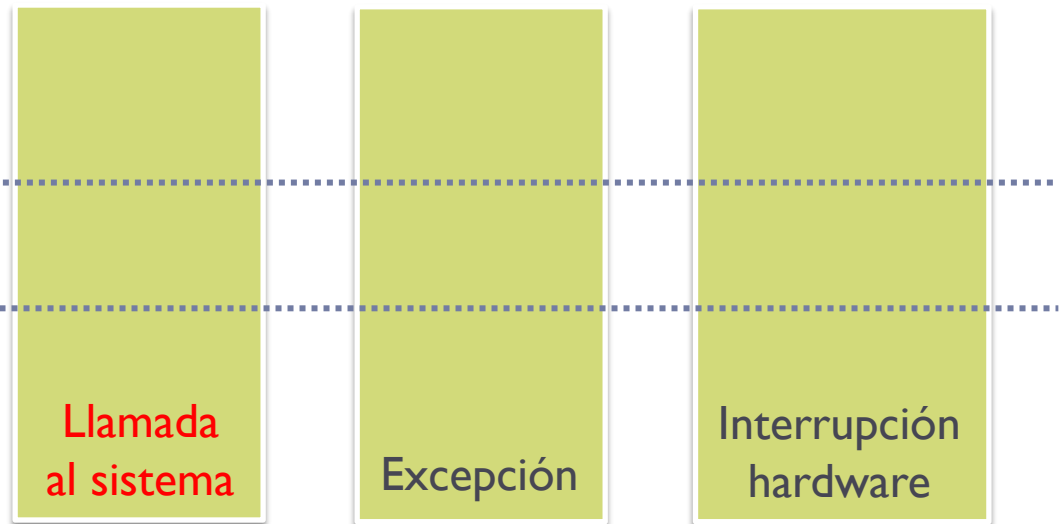
Gestión de eventos en CREATOR

- ▶ Modos incluidos:

- ▶ Modo usuario/a (no privilegiado)
- ▶ Modo kernel (privilegiado)

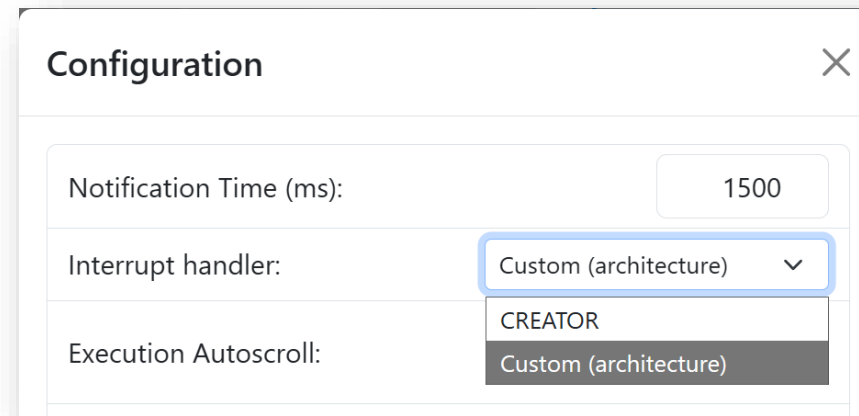
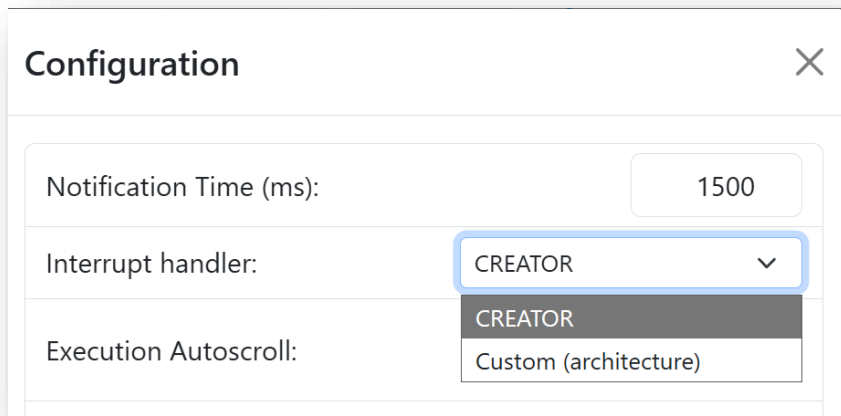
- ▶ Transiciones:

- ▶ Usuario a kernel (*trap*)
- ▶ Kernel a usuario/a (retorno de *trap*)



Soporte para llamadas al sistema

- ▶ **Manejador por defecto en CREATOR**
 - ▶ Usado para las llamadas al sistema (ecall/syscall) sin *trap*
- ▶ **Manejador definido por la arquitectura**
 - ▶ Precisa definir el manejador en el programa ensamblador



Manejador por defecto en CREATOR

```
{
  "name": "ecall",
  "type": "Syscall",

  # ...

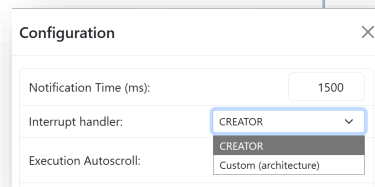
  "definition": "switch(a7) {
    case 1: capi_print_int('a0');
                break;
    case 2: capi_print_float('fa0');
                break;
    # ...
    case 12: capi_read_char('a0');
              break;
  }",
  "help": "",
},
# ...
```

- ▶ Antes de añadir interrupciones a CREATOR, la definición de la función **ecall** de RISC-V ejecutaba la llamada al sistema deseada en función del valor del registro a7.
- ▶ Se analizó cómo añadir soporte de interrupciones a CREATOR y que **ecall** usara dicho soporte:
 - ▶ No debería de obligar a visibilizar un *kernel* con la implementación de manejadores
 - ▶ No se quería incluir un *kernel* de forma silenciosa que pudiera tener interacciones no deseadas
 - ▶ No debería forzar a tener dos arquitecturas: una con interrupciones y otra sin ellas

Manejador por defecto en CREATOR

```
# ...
instructions:
  base:
    # ...
    - name: ecall
      template: standard
      fields:
        - field: opcode
          value: "0x02"
      definition: CAPI.INTERRUPTS.create(InterruptType.EnvironmentCall);

# ...
interrupts:
  check: |
  is_enabled: |
  enable: |
  disable: |
  create: |
  clear: |
  global_clear: |
  handlers:
    # ...
    creator_ecall: |
      switch (registers.A) {
        case 1n:
          CAPI.SYSCALL.print(registers.B, 'int32');
          break;
        # ...
      }
    CAPI.INTERRUPTS.clearHighlight();
    CAPI.STACK.endFrame();
```

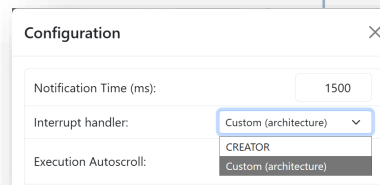


- ▶ Antes de añadir interrupciones a CREATOR, la definición de la función **ecall** de RISC-V ejecutaba la llamada al sistema deseada en función del valor del registro a7.
- ▶ Se analizó cómo añadir soporte de interrupciones a CREATOR y que **ecall** usara dicho soporte:
 - ▶ No debería de obligar a visibilizar un *kernel* con la implementación de manejadores
 - ▶ No se quería incluir un *kernel* de forma silenciosa que pudiera tener interacciones no deseadas
 - ▶ No debería forzar a tener dos arquitecturas: una con interrupciones y otra sin ellas
- ▶ Aproximación usada:
 - ▶ Tener dos controladores de interrupciones.
 - ▶ Uno resuelve **ecall** casi como antes (no *kernel*)
 - ▶ El otro llama a una subrutina de tratamiento (*rti*)

Manejador definido por la arquitectura

```
# ...
instructions:
base:
# ...
- name: ecall
template: standard
fields:
- field: opcode
value: "0x02"
definition: CAPI.INTERRUPTS.create(InterruptType.EnvironmentCall);
```

```
# ...
interrupts:
check: |
# ...
handlers:
custom: |
CAPI.INTERRUPTS.globalDisable();
CAPI.INTERRUPTS.setKernelMode();
registers.mepc = CAPI.REG.read("pc");
if (registers.mtvec & 1n) { // vectored mode
registers.pc = (registers.mtvec >> 2n) + 4*(registers.mcause & (2**32 - 1));
} else { // direct mode
registers.pc = registers.mtvec >> 2n;
}
```

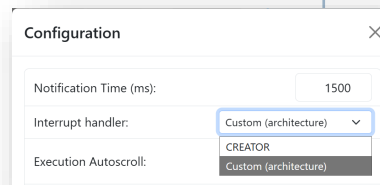


- ▶ Antes de añadir interrupciones a CREATOR, la definición de la función **ecall** de RISC-V ejecutaba la llamada al sistema deseada en función del valor del registro a7.
- ▶ Se analizó cómo añadir soporte de interrupciones a CREATOR y que **ecall** usara dicho soporte:
 - ▶ No debería de obligar a visibilizar un *kernel* con la implementación de manejadores
 - ▶ No se quería incluir un *kernel* de forma silenciosa que pudiera tener interacciones no deseadas
 - ▶ No debería forzar a tener dos arquitecturas: una con interrupciones y otra sin ellas
- ▶ Aproximación usada:
 - ▶ Tener dos controladores de interrupciones.
 - ▶ Uno resuelve **ecall** casi como antes (no *kernel*)
 - ▶ El otro llama a una subrutina de tratamiento (rti)

Manejador definido por la arquitectura

```
# ...
instructions:
base:
- name: mret
  template: standard
  fields:
  - field: opcode
    value: "0x01"
  definition: |
    // restore PC and switch to kernel mode
    registers.pc = registers.mepc ;
    CAPI.INTERRUPTS.setUserMode() ;
    // Enable interrupts
    CAPI.INTERRUPTS.globalClear();
    CAPI.INTERRUPTS.globalEnable() ;
    // Some pending clean-up for UI
    CAPI.INTERRUPTS.clearHighlight() ;
    CAPI.STACK.endFrame() ;

# ...
interrupts:
check: |
# ...
handlers:
custom: |
  CAPI.INTERRUPTS.globalDisable();
  CAPI.INTERRUPTS.setKernelMode();
  registers.mepc = CAPI.REG.read("pc");
  if (registers.mtvec & 1n) { // vectored mode
    registers.pc = (registers.mtvec >> 2n) + 4*(registers.mcause & (2**32 - 1));
  } else { // direct mode
    registers.pc = registers.mtvec >> 2n;
  }
}
```

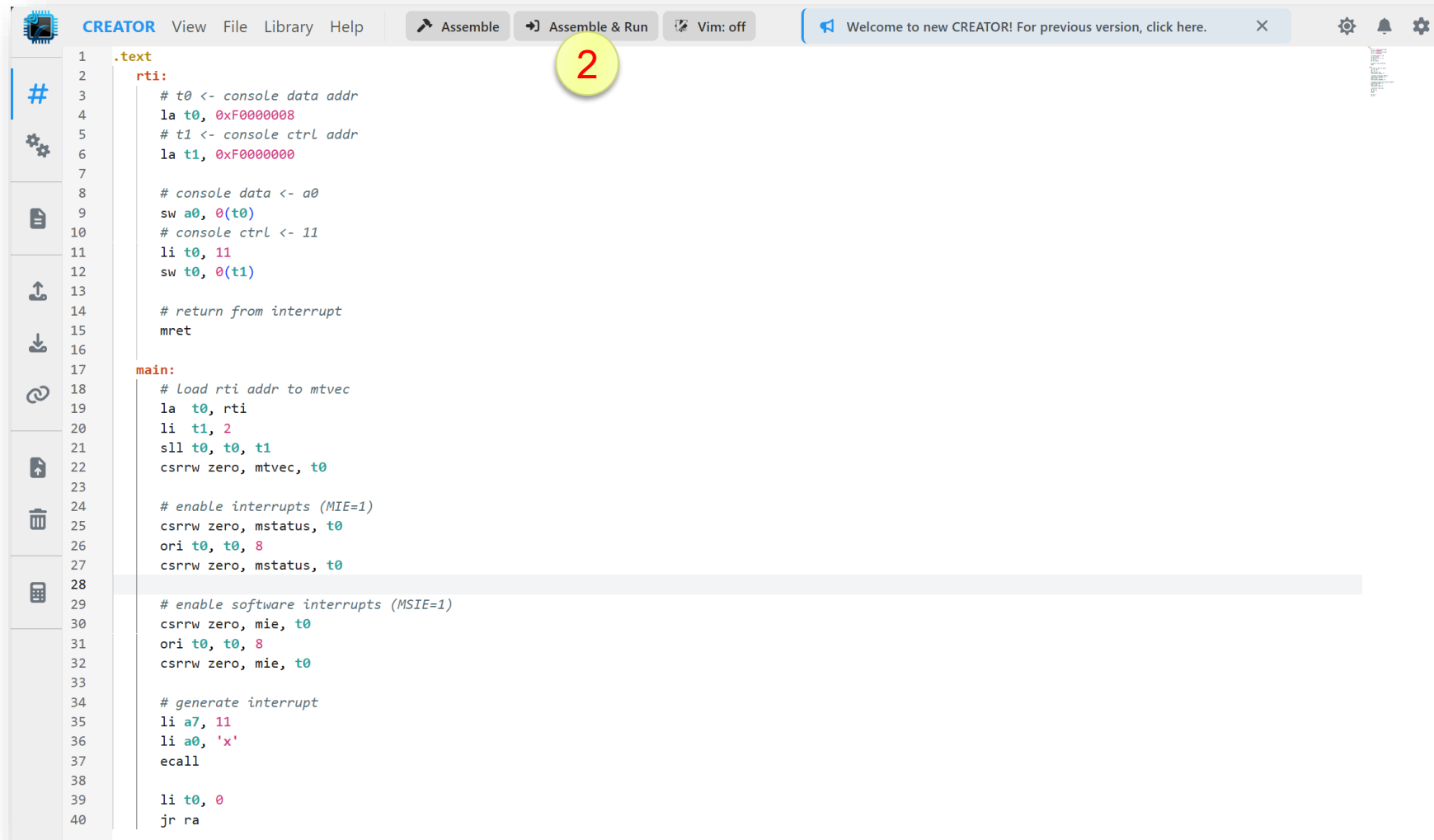


- ▶ Antes de añadir interrupciones a CREATOR, la definición de la función **ecall** de RISC-V ejecutaba la llamada al sistema deseada en función del valor del registro a7.
- ▶ Se analizó cómo añadir soporte de interrupciones a CREATOR y que **ecall** usara dicho soporte:
 - ▶ No debería de obligar a visibilizar un *kernel* con la implementación de manejadores
 - ▶ No se quería incluir un *kernel* de forma silenciosa que pudiera tener interacciones no deseadas
 - ▶ No debería forzar a tener dos arquitecturas: una con interrupciones y otra sin ellas
- ▶ Aproximación usada:
 - ▶ Tener dos controladores de interrupciones.
 - ▶ Uno resuelve **ecall** casi como antes (no *kernel*)
 - ▶ El otro llama a una subrutina de tratamiento (rti)

Soporte para llamadas: ejemplo

Ejemplo

1



```
1 .text
2 rti:
3 # t0 <- console data addr
4 la t0, 0xF0000008
5 # t1 <- console ctrl addr
6 la t1, 0xF0000000
7
8 # console data <- a0
9 sw a0, 0(t0)
10 # console ctrl <- 11
11 li t0, 11
12 sw t0, 0(t1)
13
14 # return from interrupt
15 mret
16
17 main:
18 # Load rti addr to mtvec
19 la t0, rti
20 li t1, 2
21 sll t0, t0, t1
22 csrrw zero, mtvec, t0
23
24 # enable interrupts (MIE=1)
25 csrrw zero, mstatus, t0
26 ori t0, t0, 8
27 csrrw zero, mstatus, t0
28
29 # enable software interrupts (MSIE=1)
30 csrrw zero, mie, t0
31 ori t0, t0, 8
32 csrrw zero, mie, t0
33
34 # generate interrupt
35 li a7, 11
36 li a0, 'x'
37 ecall
38
39 li t0, 0
40 jr ra
```

Soporte para llamadas: ejemplo

Ejemplo

The screenshot shows the CREATOR IDE interface. On the left, a table lists assembly instructions with their addresses, user instructions, and loaded instructions. The instruction at address 0x20 is highlighted in green and labeled 'main'. The 'Registers' tab is selected, showing a grid of registers categorized into Control, Integer, and Floating point registers. A yellow circle with the number '3' is placed over the 'Registers' tab label.

Address	User Instruction	Loaded Instructions
0x0	la t0, 0xF0000008	auipc t0, -65536
0x4		addi t0, t0, 8
0x8	la t1, 0xF0000000	auipc t1, -65536
0xC		addi t1, t1, -8
0x10	sw a0, 0(t0)	sw a0, 0(t0)
0x14	li t0, 11	addi t0, x0, 11
0x18	sw t0, 0(t1)	sw t0, 0(t1)
0x1C	mret	mret
0x20	la t0, rti	auipc t0, 0
0x24		addi t0, t0, -32
0x28	li t1, 2	addi t1, x0, 2
0x2C	sll t0, t0, t1	sll t0, t0, t1
0x30	csrrw zero, mtvec, t0	csrrw zero, mtvec, t0
0x34	csrrw zero, mstatus, t0	csrrw zero, mstatus, t0
0x38	ori t0, t0, 8	ori t0, t0, 8
0x3C	csrrw zero, mstatus, t0	csrrw zero, mstatus, t0
0x40	csrrw zero, mie, t0	csrrw zero, mie, t0
0x44	ori t0, t0, 8	ori t0, t0, 8
0x48	csrrw zero, mie, t0	csrrw zero, mie, t0
0x4C	li a7, 11	addi a7, x0, 11
0x50	li a0, 'x'	addi a0, x0, 120
0x54	ecall	ecall
0x58	li t0, 0	addi t0, x0, 0
0x5C	jr ra	jlr x0, 0(ra)

Registers

Control registers

pc 00000020	mepc 00000000	mcause 00000000	mtvec 00000000	mstatus 00000008
mip 00000000	mie 00000000	mscratch 00000000	mtime 00000000	mtimecmp 00000000

Integer registers

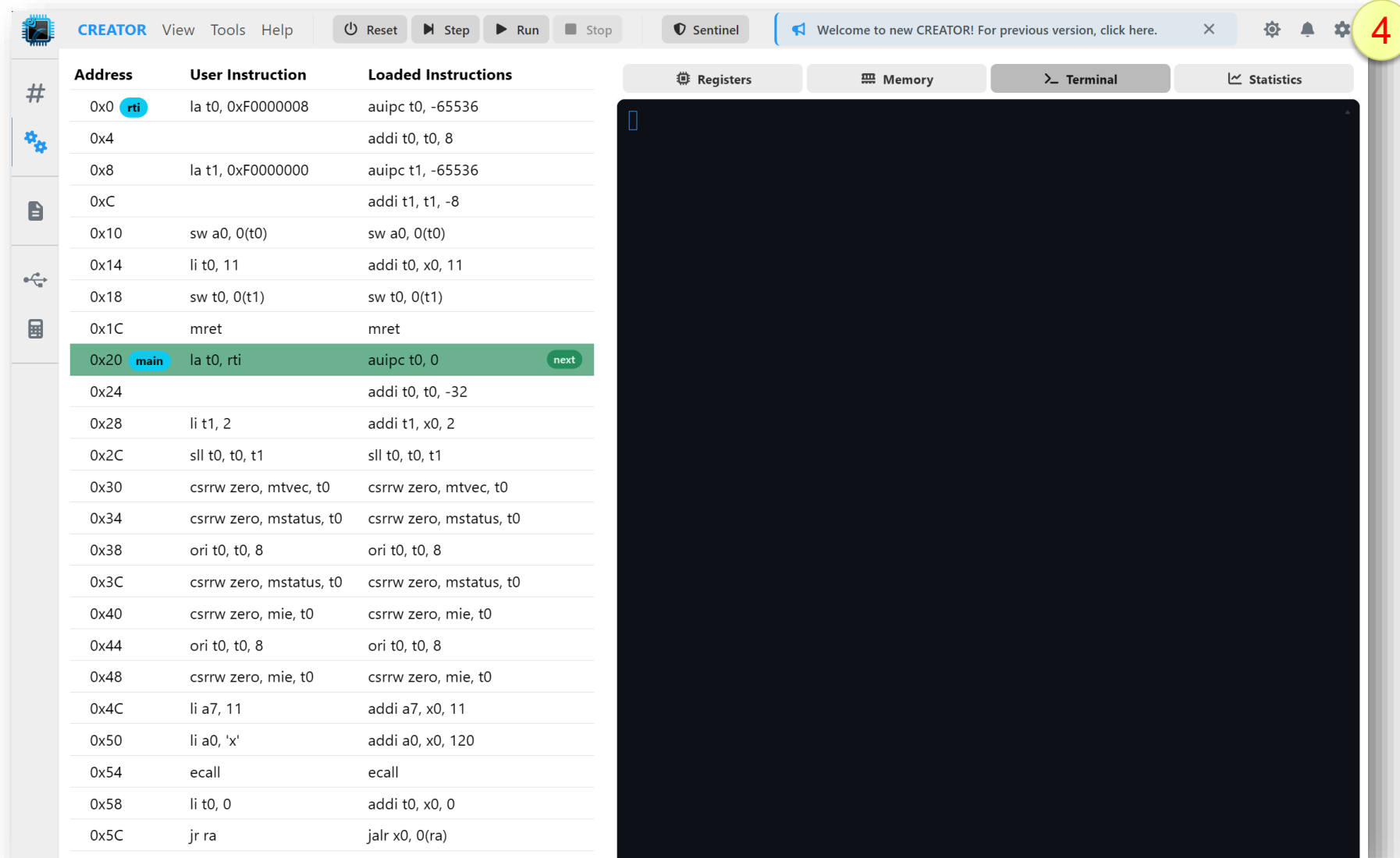
x0 zero 00000000	x1 ra FFFFFFFF	x2 sp 0FFFFFFC	x3 gp 00000000	x4 tp 00000000
x5 t0 00000000	x6 t1 00000000	x7 t2 00000000	x8 fp s0 00000000	x9 s1 00000000
x10 a0 00000000	x11 a1 00000000	x12 a2 00000000	x13 a3 00000000	x14 a4 00000000
x15 a5 00000000	x16 a6 00000000	x17 a7 00000000	x18 s2 00000000	x19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

Floating point registers

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

Soporte para llamadas: ejemplo

Ejemplo



The screenshot shows the CREATOR IDE interface. The top menu bar includes 'CREATOR', 'View', 'Tools', and 'Help'. Below the menu are control buttons for 'Reset', 'Step', 'Run', and 'Stop', along with a 'Sentinel' icon and a welcome message. The main workspace is divided into several panes: a list of instructions, 'Registers', 'Memory', 'Terminal', and 'Statistics'. The instruction list has columns for 'Address', 'User Instruction', and 'Loaded Instructions'. The instruction at address 0x20 is highlighted in green and labeled 'main', with a 'next' button to its right. The terminal window is currently empty.

Address	User Instruction	Loaded Instructions
0x0	la t0, 0xF0000008	auipc t0, -65536
0x4		addi t0, t0, 8
0x8	la t1, 0xF0000000	auipc t1, -65536
0xC		addi t1, t1, -8
0x10	sw a0, 0(t0)	sw a0, 0(t0)
0x14	li t0, 11	addi t0, x0, 11
0x18	sw t0, 0(t1)	sw t0, 0(t1)
0x1C	mret	mret
0x20	la t0, rti	auipc t0, 0
0x24		addi t0, t0, -32
0x28	li t1, 2	addi t1, x0, 2
0x2C	sll t0, t0, t1	sll t0, t0, t1
0x30	csrrw zero, mtvec, t0	csrrw zero, mtvec, t0
0x34	csrrw zero, mstatus, t0	csrrw zero, mstatus, t0
0x38	ori t0, t0, 8	ori t0, t0, 8
0x3C	csrrw zero, mstatus, t0	csrrw zero, mstatus, t0
0x40	csrrw zero, mie, t0	csrrw zero, mie, t0
0x44	ori t0, t0, 8	ori t0, t0, 8
0x48	csrrw zero, mie, t0	csrrw zero, mie, t0
0x4C	li a7, 11	addi a7, x0, 11
0x50	li a0, 'x'	addi a0, x0, 120
0x54	ecall	ecall
0x58	li t0, 0	addi t0, x0, 0
0x5C	jr ra	jalr x0, 0(ra)

Soporte para llamadas: ejemplo

Ejemplo

#	Address	User Instruction	Loaded Ins
	0x0	la t0, 0xF0000008	auipc t0, -6
	0x4		addi t0, t0, 8
	0x8	la t1, 0xF0000000	auipc t1, -6
	0xC		addi t1, t1, -
	0x10	sw a0, 0(t0)	sw a0, 0(t0)
	0x14	li t0, 11	addi t0, x0,
	0x18	sw t0, 0(t1)	sw t0, 0(t1)
	0x1C	mret	mret
	0x20	la t0, rti	auipc t0, 0
	0x24		addi t0, t0, -
	0x28	li t1, 2	addi t1, x0,
	0x2C	sll t0, t0, t1	sll t0, t0, t1
	0x30	csrwr zero, mtvec, t0	csrwr zero, t
	0x34	csrwr zero, mstatus, t0	csrwr zero, t
	0x38	ori t0, t0, 8	ori t0, t0, 8
	0x3C	csrwr zero, mstatus, t0	csrwr zero, t
	0x40	csrwr zero, mie, t0	csrwr zero, t
	0x44	ori t0, t0, 8	ori t0, t0, 8
	0x48	csrwr zero, mie, t0	csrwr zero, t
	0x4C	li a7, 11	addi a7, x0,
	0x50	li a0, 'x'	addi a0, x0, 120
	0x54	ecall	ecall
	0x58	li t0, 0	addi t0, x0, 0
	0x5C	jr ra	jalr x0, 0(ra)

Soporte para llamadas: ejemplo

Ejemplo

6

The screenshot shows the CREATOR IDE interface. The top menu bar includes 'View', 'Tools', and 'Help'. Below the menu, there are buttons for 'Reset', 'Finished', and 'Stop'. A yellow circle with the number '6' is positioned above the 'Finished' button. The main window is divided into several panes. On the left, there is a sidebar with icons for '#', 'Settings', 'File', 'Share', and 'Table'. The central pane displays assembly code with columns for 'Address', 'User Instruction', and 'Loaded Instructions'. The 'main' label is highlighted in blue at address 0x20. On the right, there are tabs for 'Registers', 'Memory', 'Terminal', and 'Statistics'. The 'Terminal' tab is active, showing a black terminal window with a red circle around the 'x' icon in the top-left corner.

Address	User Instruction	Loaded Instructions
0x0	la t0, 0xF0000008	auipc t0, -65536
0x4		addi t0, t0, 8
0x8	la t1, 0xF0000000	auipc t1, -65536
0xC		addi t1, t1, -8
0x10	sw a0, 0(t0)	sw a0, 0(t0)
0x14	li t0, 11	addi t0, x0, 11
0x18	sw t0, 0(t1)	sw t0, 0(t1)
0x1C	mret	mret
0x20	la t0, rti	auipc t0, 0
0x24		addi t0, t0, -32
0x28	li t1, 2	addi t1, x0, 2
0x2C	sll t0, t0, t1	sll t0, t0, t1
0x30	csrrw zero, mtvec, t0	csrrw zero, mtvec, t0
0x34	csrrw zero, mstatus, t0	csrrw zero, mstatus, t0
0x38	ori t0, t0, 8	ori t0, t0, 8
0x3C	csrrw zero, mstatus, t0	csrrw zero, mstatus, t0
0x40	csrrw zero, mie, t0	csrrw zero, mie, t0
0x44	ori t0, t0, 8	ori t0, t0, 8
0x48	csrrw zero, mie, t0	csrrw zero, mie, t0
0x4C	li a7, 11	addi a7, x0, 11
0x50	li a0, 'x'	addi a0, x0, 120
0x54	ecall	ecall
0x58	li t0, 0	addi t0, x0, 0
0x5C	jr ra	jalr x0, 0(ra)

Soporte para llamadas: ejemplo

Ejemplo

7

The screenshot shows the CREATOR IDE interface. The top menu bar includes 'CREATOR', 'View', 'Tools', and 'Help'. Below the menu are buttons for 'Reset', 'Step', 'Run', and 'Stop'. A 'Sentinel' icon is also present. A notification bar says 'Welcome to new CREATOR! For previous version, click here.' On the right side of the IDE, there are icons for settings, a bell, and a gear. A yellow circle with the number '8' is overlaid on the top right corner of the IDE window.

The main window is divided into two panes. The left pane displays assembly code with columns for 'Address', 'User Instruction', and 'Loaded Instructions'. The right pane is a terminal window, currently empty.

Address	User Instruction	Loaded Instructions
0x0	la t0, 0xF0000008	auipc t0, -65536
0x4		addi t0, t0, 8
0x8	la t1, 0xF0000000	auipc t1, -65536
0xC		addi t1, t1, -8
0x10	sw a0, 0(t0)	sw a0, 0(t0)
0x14	li t0, 11	addi t0, x0, 11
0x18	sw t0, 0(t1)	sw t0, 0(t1)
0x1C	mret	mret
0x20	la t0, rti	auipc t0, 0
0x24		addi t0, t0, -32
0x28	li t1, 2	addi t1, x0, 2
0x2C	sll t0, t0, t1	sll t0, t0, t1
0x30	csrrw zero, mtvec, t0	csrrw zero, mtvec, t0
0x34	csrrw zero, mstatus, t0	csrrw zero, mstatus, t0
0x38	ori t0, t0, 8	ori t0, t0, 8
0x3C	csrrw zero, mstatus, t0	csrrw zero, mstatus, t0
0x40	csrrw zero, mie, t0	csrrw zero, mie, t0
0x44	ori t0, t0, 8	ori t0, t0, 8
0x48	csrrw zero, mie, t0	csrrw zero, mie, t0
0x4C	li a7, 11	addi a7, x0, 11
0x50	li a0, 'x'	addi a0, x0, 120
0x54	ecall	ecall
0x58	li t0, 0	addi t0, x0, 0
0x5C	jr ra	jalr x0, 0(ra)

8

Soporte para llamadas: ejemplo

Ejemplo

#	Address	User Instruction	Loaded Instruction
	0x0	la t0, 0xF0000008	auipc t0, -65536
	0x4		addi t0, t0, 8
	0x8	la t1, 0xF0000000	auipc t1, -65536
	0xC		addi t1, t1, 8
	0x10	sw a0, 0(t0)	sw a0, 0(t0)
	0x14	li t0, 11	addi t0, x0, 11
	0x18	sw t0, 0(t1)	sw t0, 0(t1)
	0x1C	mret	mret
	0x20	la t0, rti	auipc t0, 0
	0x24		addi t0, t0, 8
	0x28	li t1, 2	addi t1, x0, 2
	0x2C	sll t0, t0, t1	sll t0, t0, t1
	0x30	csrrw zero, mtvec, t0	csrrw zero, mtvec, t0
	0x34	csrrw zero, mstatus, t0	csrrw zero, mstatus, t0
	0x38	ori t0, t0, 8	ori t0, t0, 8
	0x3C	csrrw zero, mstatus, t0	csrrw zero, mstatus, t0
	0x40	csrrw zero, mie, t0	csrrw zero, mie, t0
	0x44	ori t0, t0, 8	ori t0, t0, 8
	0x48	csrrw zero, mie, t0	csrrw zero, mie, t0
	0x4C	li a7, 11	addi a7, x0, 11
	0x50	li a0, 'x'	addi a0, x0, 120
	0x54	ecall	ecall
	0x58	li t0, 0	addi t0, x0, 0
	0x5C	jr ra	jalr x0, 0(ra)

Soporte para llamadas: ejemplo

Ejemplo

10

The screenshot shows the CREATOR IDE interface. The main window displays assembly code with columns for Address, User Instruction, and Loaded Instructions. The instruction at address 0x18 is highlighted in blue, and the instruction at address 0x1C is highlighted in green with a 'next' button. The instruction at address 0x54 is highlighted in yellow with a 'next' button. A red circle highlights the 'x' icon in the top-left corner of the terminal window, which is currently empty.

Address	User Instruction	Loaded Instructions
0x0	la t0, 0xF0000008	auipc t0, -65536
0x4		addi t0, t0, 8
0x8	la t1, 0xF0000000	auipc t1, -65536
0xC		addi t1, t1, -8
0x10	sw a0, 0(t0)	sw a0, 0(t0)
0x14	li t0, 11	addi t0, x0, 11
0x18	sw t0, 0(t1)	sw t0, 0(t1)
0x1C	mret	mret
0x20	la t0, rti	auipc t0, 0
0x24		addi t0, t0, -32
0x28	li t1, 2	addi t1, x0, 2
0x2C	sll t0, t0, t1	sll t0, t0, t1
0x30	crrw zero, mtvec, t0	crrw zero, mtvec, t0
0x34	crrw zero, mstatus, t0	crrw zero, mstatus, t0
0x38	ori t0, t0, 8	ori t0, t0, 8
0x3C	crrw zero, mstatus, t0	crrw zero, mstatus, t0
0x40	crrw zero, mie, t0	crrw zero, mie, t0
0x44	ori t0, t0, 8	ori t0, t0, 8
0x48	crrw zero, mie, t0	crrw zero, mie, t0
0x4C	li a7, 11	addi a7, x0, 11
0x50	li a0, 'x'	addi a0, x0, 120
0x54	ecall	ecall
0x58	li t0, 0	addi t0, x0, 0
0x5C	jr ra	jalr x0, 0(ra)

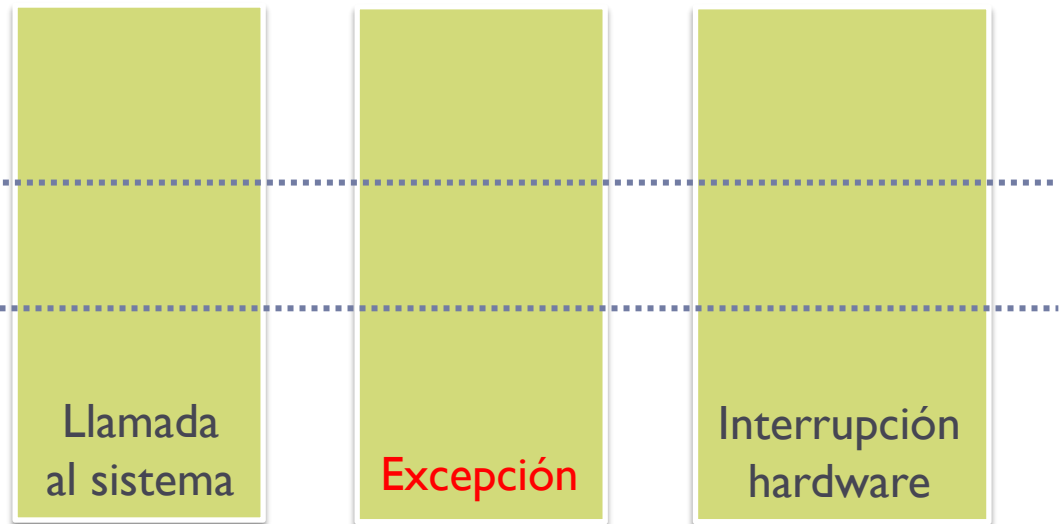
Modos de ejecución en CREATOR

- ▶ Modos incluidos:

- ▶ Modo usuario/a (no privilegiado)
- ▶ Modo kernel (privilegiado)

- ▶ Transiciones:

- ▶ Usuario a kernel (*trap*)
- ▶ Kernel a usuario/a (retorno de *trap*)



Soporte para excepciones

- ▶ Gestionado desde la instrucción tras una validación
 - ▶ Usado para excepciones aritméticas (0/0) y de protección (dirección no válida).

```
Edit Architecture

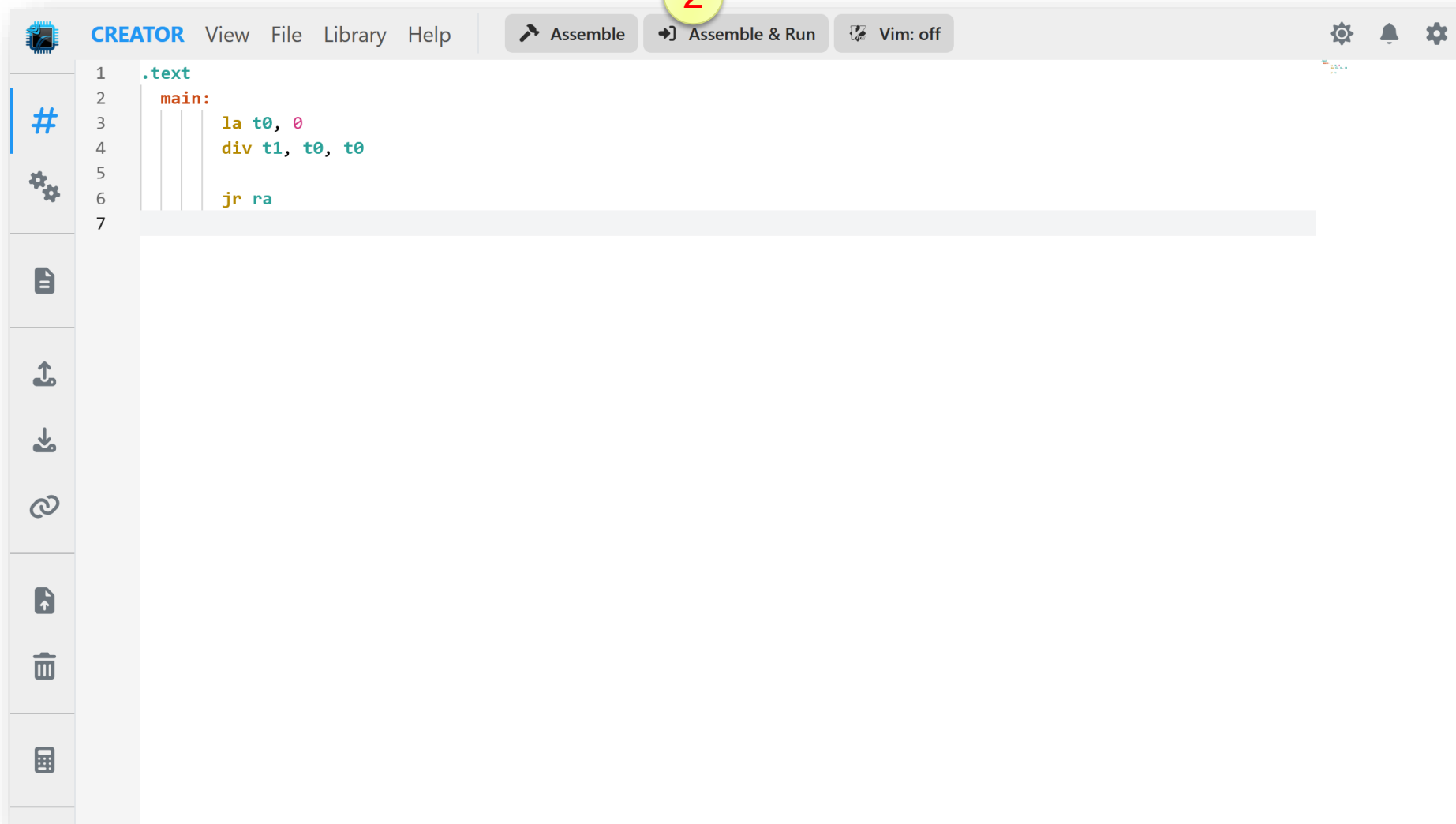
2084
2085 # R-type instructions
2086 - name: div
2087   template: R
2088   fields:
2089     - field: opcode
2090       | value: "0110011"
2091     - field: funct7
2092       | value: "0000001"
2093     - field: funct3
2094       | value: "100"
2095   definition: |
2096     if (registers[rs2] != 0) {
2097       registers[rd] = BigInt.asIntN(
2098         32,
2099         (BigInt.asIntN(32, BigInt(registers[rs1])) / BigInt.asIntN(32, BigInt(registers[rs2])))
2100       );
2101     }
2102     else CAPI.VALIDATION.raise('Division by zero not allowed');
2103 help: Perform an XLEN bits by XLEN bits signed integer division of rs1 by
2104 rs2, rounding towards zero.
2105
```

Soporte para excepciones: ejemplo

Ejemplo

1

2



The screenshot shows the CREATOR IDE interface. The menu bar includes 'View', 'File', 'Library', and 'Help'. The toolbar contains 'Assemble', 'Assemble & Run', and 'Vim: off'. The editor displays assembly code for a .text segment:

```
1 .text
2   main:
3     |   |   |   |   |
4     |   |   |   |   |   la t0, 0
5     |   |   |   |   |   div t1, t0, t0
6     |   |   |   |   |   jr  ra
7
```

Soporte para excepciones: ejemplo

Ejemplo

3

The screenshot shows the CREATOR IDE interface. The top bar includes 'CREATOR View Tools Help', 'Reset', 'Finished', 'Stop', and 'Sentinel'. The main window is divided into several panes:

- Instruction List:** A table with columns 'Address', 'User Instruction', and 'Loaded Instructions'. The instruction at address 0xC, 'jalr x0, 0(ra)', is highlighted in blue.
- Registers:** A section with tabs for 'Registers', 'Memory', 'Terminal', and 'Statistics'. It is expanded to show 'Control registers' and 'Integer registers'.
 - Control registers:** Includes 'pc' (FFFFF7FE), 'mepc' (00000000), 'mcause' (00000000), 'mtvec' (00000000), 'mstatus' (00000008), 'mip' (00000000), 'mie' (00008888), 'mscratch' (00000000), 'mtime' (00000004), and 'mtimecmp' (00000000).
 - Integer registers:** A grid of registers from x0 to x31, each with its name and value. For example, x0 | zero is 00000000, x1 | ra is FFFFFFFF, and x2 | sp is 0FFFFFFC.
- Floating point registers:** A section for floating point registers, partially obscured by an error dialog.

A red error dialog box is overlaid on the floating point registers, containing the text: 'There has been an exception. Error description: Division by zero not allowed'.

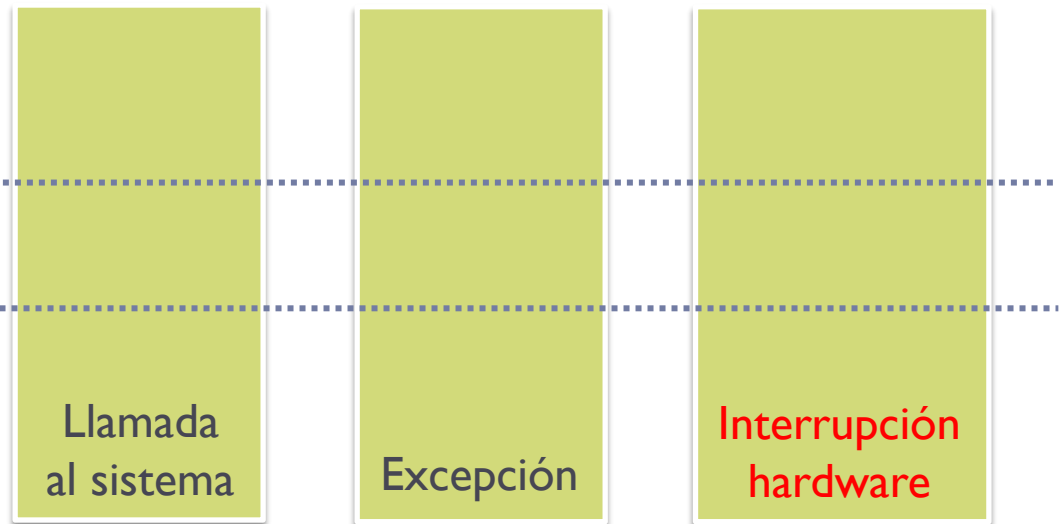
Modos de ejecución en CREATOR

- ▶ **Modos incluidos:**

- ▶ Modo usuario/a (no privilegiado)
- ▶ Modo kernel (privilegiado)

- ▶ **Transiciones:**

- ▶ Usuario a kernel (*trap*)
- ▶ Kernel a usuario/a (retorno de *trap*)



Soporte para interrupciones

- ▶ Manejador definido por la arquitectura
 - ▶ Precisa definir el manejador en el programa ensamblador
 - ▶ Precisa un dispositivo que genere interrupciones

```
Edit Architecture

4157
4158 timer:
4159     tick_cycles: 1
4160     advance: |
4161         registers.mtime = (registers.mtime + 1n) % (2n**32n - 1n);
4162     handler: |
4163         if (registers.mtime === registers.mtimecmp) {
4164             CAPI.INTERRUPTS.create(InterruptType.Timer);
4165         }
4166     is_enabled: |
4167         return !! (registers.mie & 2n ** 7n); // MTIE
4168     enable: |
4169         registers.mie |= 2n ** 7n; // MTIE = 1
4170     disable: |
4171         registers.mie &= ~(2n ** 7n); // MTIE = 0
4172
```

Soporte para interrupciones: ejemplo

Ejemplo

1

2

```
CREATOR View File Library Help Assemble Assemble & Run Vim: off Welcome to new CREATOR! For previous version, click here. X
1 .text
2 rti:
3 # [t0, t1] <- console [data, ctrl.] addr
4 la t0, 0xF0000008
5 la t1, 0xF0000000
6
7 # print('x', 11)
8 li a0, 'x'
9 sw a0, 0(t0)
10 li t0, 11
11 sw t0, 0(t1)
12
13 # return from interrupt
14 mret
15
16 main:
17 # Load rti addr to mtvec
18 la t0, rti
19 li t1, 2
20 sll t0, t0, t1
21 csrrw zero, mtvec, t0
22
23 # enable interrupts (mstatus=1000b)
24 csrrw zero, mstatus, t0
25 ori t0, t0, 8
26 csrrw zero, mstatus, t0
27
28 # set timer (mtimecmp=15)
29 li t0, 15
30 csrrw zero, mtimecmp, t0
31
32 # enable timer (MIE=2^7)
33 csrrw zero, mie, t0
34 ori t0, t0, 128
35 csrrw zero, mie, t0
36
37 # wait for interrupt...
38 li t5, 0
39 li t6, 10
40 l1: beq t5, t6, e1
41 addi t5, t5, 1
42 beq x0, x0, l1
43 e1: # end
44 jr ra
45
```

Soporte para interrupciones: ejemplo

Ejemplo

The screenshot shows the CREATOR IDE interface. A 'Configuration' dialog box is open, displaying various settings. The 'Interrupt handler' dropdown menu is highlighted with a yellow circle containing the number 4. The 'Notification Time (ms)' is set to 1500. Other settings include 'Execution Autoscroll' (checked), 'Automatic backup' (checked), 'Theme' (Light), 'Debug' (unchecked), 'Sidebar Mode' (Show), 'Vim mode' (unchecked), 'Vim Custom Keybinds' (dropdown), 'Register Name Representation' (All), and 'Kernel Simulator' (checked). The background shows the assembly code editor with instructions like 'la t0, 0xF0000008' and 'auipc t0, -65'. A yellow circle with the number 3 is located in the top right corner of the IDE window.

Soporte para interrupciones: ejemplo

Ejemplo

5

The screenshot displays the CREATOR IDE interface. On the left, a list of assembly instructions is shown with their addresses and loaded instructions. The instruction at address 0x68 is highlighted in yellow. On the right, the Register window shows the values of various registers, including control registers, integer registers, and floating point registers.

Address	User Instruction	Loaded Instructions
0x0	rti la t0, 0xF0000008	auipc t0, -65536
0x4		addi t0, t0, 8
0x8	la t1, 0xF0000000	auipc t1, -65536
0xC		addi t1, t1, -8
0x10	li a0, 'x'	addi a0, x0, 120
0x14	sw a0, 0(t0)	sw a0, 0(t0)
0x18	li t0, 11	addi t0, x0, 11
0x1C	sw t0, 0(t1)	sw t0, 0(t1)
0x20	mret	mret
0x24	main la t0, rti	auipc t0, 0
0x28		addi t0, t0, -36
0x2C	li t1, 2	addi t1, x0, 2
0x30	sll t0, t0, t1	sll t0, t0, t1
0x34	csrwr zero, mtvec, t0	csrwr zero, mtvec, t0
0x38	csrwr zero, mstatus, t0	csrwr zero, mstatus, t0
0x3C	ori t0, t0, 8	ori t0, t0, 8
0x40	csrwr zero, mstatus, t0	csrwr zero, mstatus, t0
0x44	li t0, 15	addi t0, x0, 15
0x48	csrwr zero, mtimecmp, t0	csrwr zero, mtimecmp, t0
0x4C	csrwr zero, mie, t0	csrwr zero, mie, t0
0x50	ori t0, t0, 128	ori t0, t0, 128
0x54	csrwr zero, mie, t0	csrwr zero, mie, t0
0x58	li t5, 0	addi t5, x0, 0
0x5C	li t6, 10	addi t6, x0, 10
0x60	li beq t5, t6, e1	beq t5, t6, 12
0x64	addi t5, t5, 1	addi t5, t5, 1
0x68	beq x0, x0, l1	beq x0, x0, -8
0x6C	e1 jr ra	jalr x0, 0(ra)

Control registers				
pc	mepc	mcause	mtvec	mstatus
00000000	00000060	80000007	00000000	00000008
mip	mie	mscratch	mtime	mtimecmp
00000080	0000008F	00000000	00000010	0000000F

Integer registers				
x0 zero	x1 ra	x2 sp	x3 gp	x4 tp
00000000	FFFFFFFF	0FFFFFFC	00000000	00000000
x5 t0	x6 t1	x7 t2	x8 fp s0	x9 s1
0000008F	00000002	00000000	00000000	00000000
x10 a0	x11 a1	x12 a2	x13 a3	x14 a4
00000000	00000000	00000000	00000000	00000000
x15 a5	x16 a6	x17 a7	x18 s2	x19 s3
00000000	00000000	00000000	00000000	00000000
x20 s4	x21 s5	x22 s6	x23 s7	x24 s8
00000000	00000000	00000000	00000000	00000000
x25 s9	x26 s10	x27 s11	x28 t3	x29 t4
00000000	00000000	00000000	00000000	00000000
x30 t5	x31 t6			
00000001	0000000A			

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

Soporte para interrupciones: ejemplo

Ejemplo

The screenshot shows the CREATOR IDE interface. The assembly code is displayed in a table with columns for Address, User Instruction, and Loaded Instructions. The instruction at address 0x1C is highlighted in blue, and the instruction at address 0x68 is highlighted in yellow. The terminal window is open and empty.

Address	User Instruction	Loaded Instructions
0x0	la t0, 0xF0000008	auipc t0, -65536
0x4		addi t0, t0, 8
0x8	la t1, 0xF0000000	auipc t1, -65536
0xC		addi t1, t1, -8
0x10	li a0, 'x'	addi a0, x0, 120
0x14	sw a0, 0(t0)	sw a0, 0(t0)
0x18	li t0, 11	addi t0, x0, 11
0x1C	sw t0, 0(t1)	sw t0, 0(t1)
0x20	mret	mret
0x24	la t0, rti	auipc t0, 0
0x28		addi t0, t0, -36
0x2C	li t1, 2	addi t1, x0, 2
0x30	sll t0, t0, t1	sll t0, t0, t1
0x34	csrrw zero, mtvec, t0	csrrw zero, mtvec, t0
0x38	csrrw zero, mstatus, t0	csrrw zero, mstatus, t0
0x3C	ori t0, t0, 8	ori t0, t0, 8
0x40	csrrw zero, mstatus, t0	csrrw zero, mstatus, t0
0x44	li t0, 15	addi t0, x0, 15
0x48	csrrw zero, mtimecmp, t0	csrrw zero, mtimecmp, t0
0x4C	csrrw zero, mie, t0	csrrw zero, mie, t0
0x50	ori t0, t0, 128	ori t0, t0, 128
0x54	csrrw zero, mie, t0	csrrw zero, mie, t0
0x58	li t5, 0	addi t5, x0, 0
0x5C	li t6, 10	addi t6, x0, 10
0x60	beq t5, t6, e1	beq t5, t6, 12
0x64	addi t5, t5, 1	addi t5, t5, 1
0x68	beq x0, x0, I1	beq x0, x0, -8
0x6C	jr ra	jalr x0, 0(ra)

Ejercicio

```
rti:
    # [t0, t1] <- [data addr, ctrl. addr]
    la t0, 0xF0000008
    la t1, 0xF0000000

    # console print('x', 11)
    li a0, 'x'
    sw a0, 0(t0)
    li t0, 11
    sw t0, 0(t1)

    # TODO: programar siguiente interrupción

    # return from interrupt
    mret
```

- ▶ De un “disparo” (*one-shot*)
- ▶ Si se precisa lanzar interrupción de forma periódica entonces al final del tratamiento de interrupción hay que programar cuándo será la siguiente.

Posible solución

Ejemplo

```
rti:
    # [t0, t1] <- [data addr, ctrl. addr]
    la t0, 0xF0000008
    la t1, 0xF0000000

    # console print('x', 11)
    li a0, 'x'
    sw a0, 0(t0)
    li t0, 11
    sw t0, 0(t1)

    # programar siguiente interrupción
    csrrw zero, mtimecmp, t0
    addi t0, t0, 35
    csrrw zero, mtimecmp, t0

    # return from interrupt
    mret
```

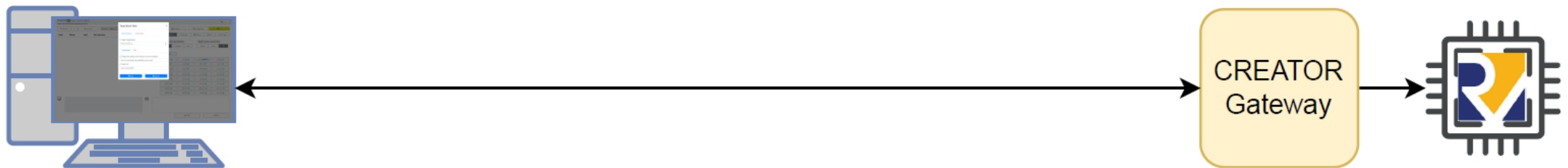
- ▶ De un “disparo” (*one-shot*)
- ▶ Si se precisa lanzar interrupción de forma periódica entonces al final del tratamiento de interrupción hay que programar cuándo será la siguiente.
- ▶ Incrementar registro `mtimecmp` al número de ciclos futuro.

Bloque 2

- ▶ Dispositivos y soporte de interrupciones
- ▶ **Integración de CREATOR con hardware real**
 - ▶ Microcontroladores
 - ▶ Placas SBC
 - ▶ Servicio de laboratorio remoto
 - ▶ Integración con Arduino (CREATino)

Integración de CREATOR con hardware real

- ▶ CREATOR como entorno de desarrollo intermedio
 - ▶ Integra lenguaje ensamblador y *hardware real*
 - ▶ Permite a los estudiantes ver las implicaciones de su código cuando ejecuta sobre *hardware real*



Integración de CREATOR con hardware real

▶ Cuadro de diálogo

- ▶ Permite al usuario cargar y ejecutar el programa en el dispositivo *hardware* desde CREATOR
- ▶ Permite definir los parámetros necesarios para realizar estas acciones
 - ▶ Modelo del dispositivo *hardware*
 - ▶ Puerto de conexión
 - ▶ URL del servicio web

Target Board Flash

Local Device Remote Device

Device type:
 Espressif SBC

Target board:
ESP32-C3 (RISC-V)

Target port (please verify the port on your computer):
rfc2217://host.docker.internal:4000?ign_set_control

(3) Flash URL:
http://localhost:8080

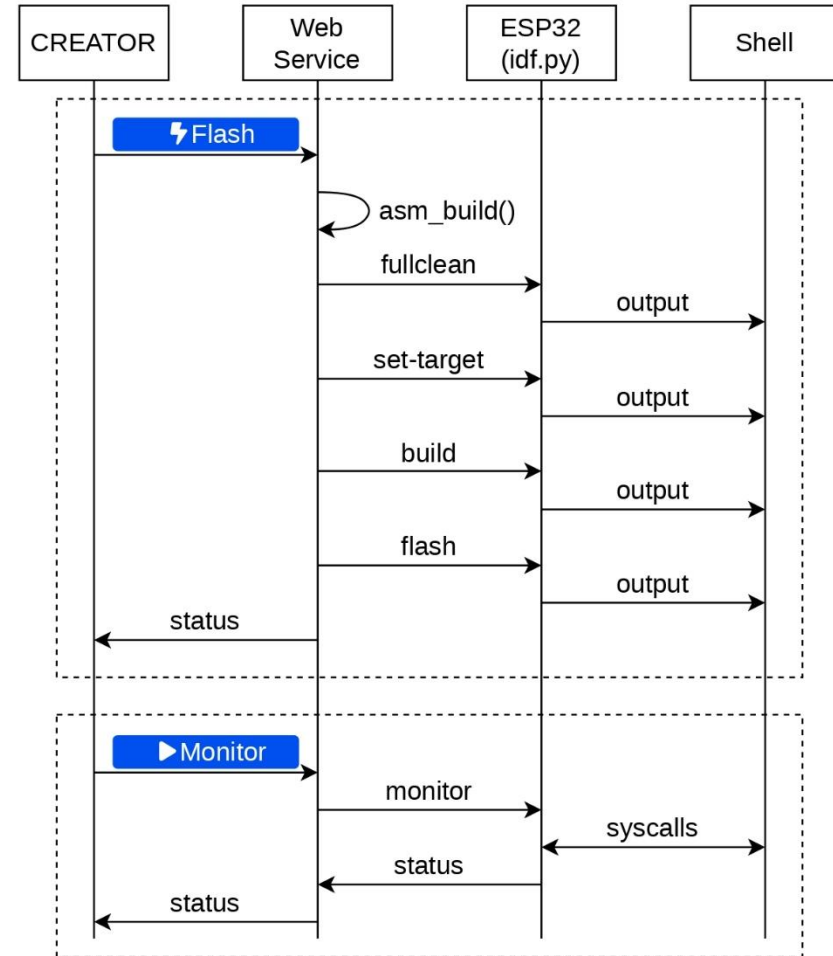
Flash Monitor
Clean Debug
Erase Flash Stop

For instructions on how to use this feature, please refer to the [CREATOR Gateway Documentation](#).

Integración de CREATOR con hardware real

▶ Servicio web

- ▶ *Framework* Flask de Python3
- ▶ Intermediario entre la interfaz web y los drivers del dispositivo *hardware*
- ▶ Operaciones:
 - ▶ *Flash*
 - ▶ *Monitor*
 - ▶ *Clear*
 - ▶ *Debug*
- ▶ Contenedor *Docker*



Microcontroladores

- ▶ Arquitectura RISC-V
- ▶ Espressif ESP32
 - ▶ ESP32 C3
 - ▶ ESP32 C6
 - ▶ ESP32 H2
- ▶ Ventajas
 - ▶ Menor coste económico
 - ▶ Mayor familiaridad
- ▶ Desventajas
 - ▶ Menor potencia de cómputo
 - ▶ Juego de instrucciones reducido

Target Board Flash

Local Device Remote Device

Device type:
 Espressif SBC

Target board:
ESP32-C3 (RISC-V)

Target port (please verify the port on your computer):
rfc2217://host.docker.internal:4000?ign_set_control

(3) Flash URL:
http://localhost:8080

Flash Monitor
Clean Debug
Erase Flash Stop

For instructions on how to use this feature, please refer to the [CREATOR Gateway Documentation](#).

Placas SBC

- ▶ Arquitectura RISC-V
- ▶ Placas SBC:
 - ▶ OrangePi RV2
 - ▶ Nezha D1-H 64 bit RISC-V
- ▶ Ventajas
 - ▶ Mayor potencia de cómputo
 - ▶ Juego de instrucciones completo
- ▶ Desventajas
 - ▶ Mayor coste económico
 - ▶ Mayor complejidad de configuración

Target Board Flash

Local Device

Remote Device

Device type:

Espressif SBC

Target board:

SBC (RISC-V)

Target user: (please verify your board user)

ubuntu@10.0.0.1

Target location: (please verify the route exists in your board)

~/creator

(3) Flash URL:

http://localhost:8080

Flash

Monitor

Clean

Debug

Erase Flash

Stop

For instructions on how to use this feature, please refer to the [CREATOR Gateway Documentation](#).

Ejercicio guiado – Microcontrolador ESP32

Despliegue Linux

1. Instalar *Docker Engine*
2. Descargar el contenedor *Docker* del Gateway de CREATOR
<https://hub.docker.com/repository/docker/creatorsim/creator-gateway-esp32>
3. Conectar el microcontrolador al ordenador (evitar utilizar HUBs de USBs)
4. Verificar el puerto donde se ha conectado. Habitualmente se encuentra en el directorio `/dev/`. Por defecto:
 1. Linux: `/dev/ttyUSB*`
5. Ejecutar el contenedor en la terminal:

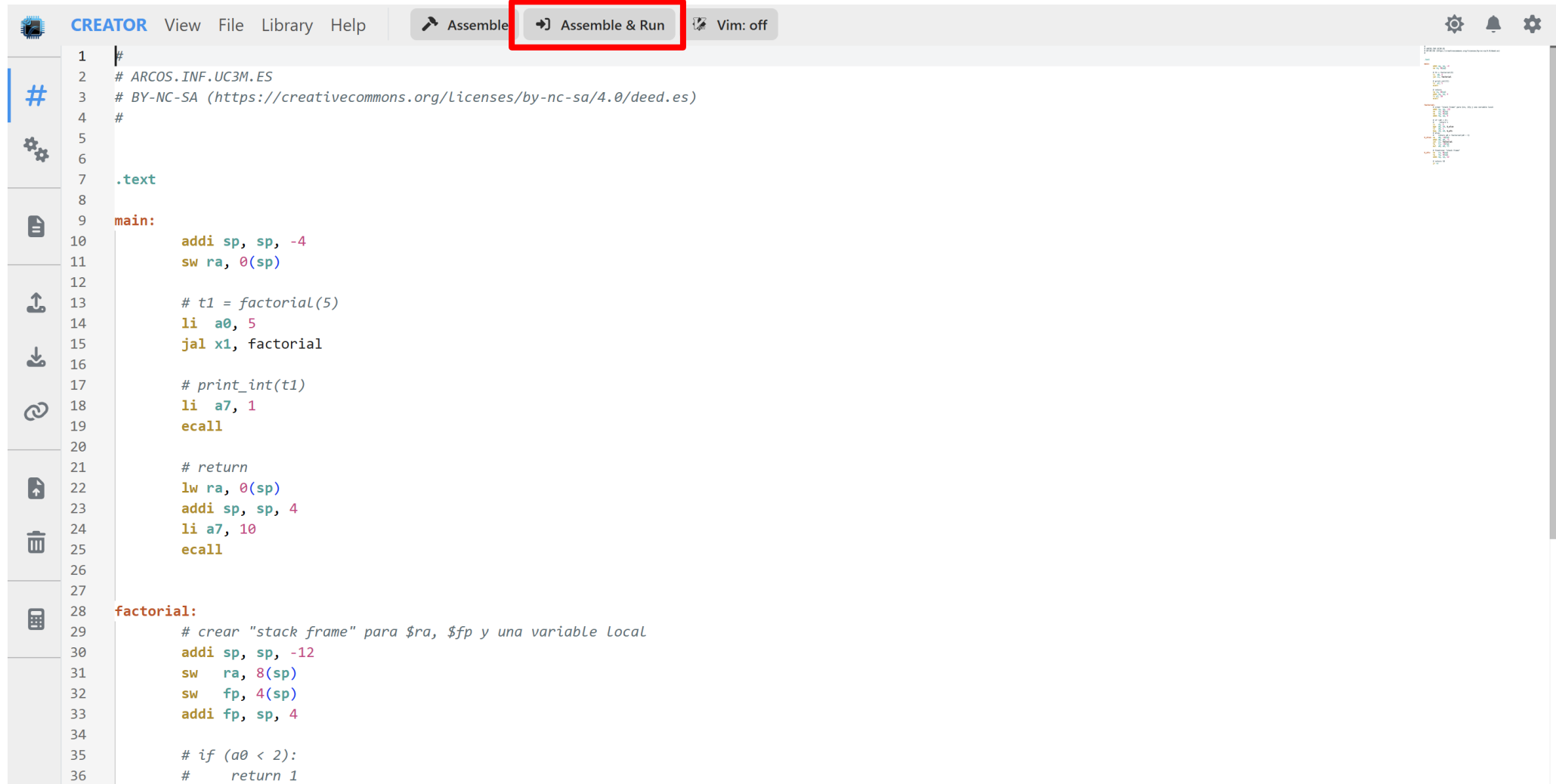
```
docker run --rm --name creator-gateway-esp32 -it --init --device=<puerto> \  
--add-host=host.docker.internal:host-gateway \  
-p 8080:8080 -p 5000:5000 creatorsim/creator-gateway-esp32
```

Ejercicio guiado – Microcontrolador ESP32

Despliegue Windows

1. Instalar *Docker Desktop*
2. Descargar el contenedor *Docker* del Gateway de CREATOR
<https://hub.docker.com/repository/docker/creatorsim/creator-gateway-esp32>
3. Conectar el microcontrolador al ordenador (evitar utilizar HUBs de USBs)
4. Verificar el puerto donde se ha conectado. Administrador de dispositivos. Por defecto:
 1. Windows: COM3
5. Configurar el puerto serie remoto:
 1. Descargar la herramienta esptool: <https://github.com/espressif/esptool/releases>
 2. Ejecutar en el CMD la herramienta: `esp_rfc2217_server -v -p 4000 <puerto COMx>`
6. Ejecutar el contenedor en el CMD:
`docker run --rm --name creator-gateway-esp32 -it --init -p 8080:8080 -p 5000:5000 \`
`--add-host=host.docker.internal:host-gateway creatorsim/creator-gateway-esp32:latest`

Ejercicio guiado – Microcontrolador ESP32



```
1 |#
2 |# ARCOS.INF.UC3M.ES
3 |# BY-NC-SA (https://creativecommons.org/licenses/by-nc-sa/4.0/deed.es)
4 |#
5 |
6 |
7 |.text
8 |
9 |main:
10 |    addi sp, sp, -4
11 |    sw ra, 0(sp)
12 |
13 |    # t1 = factorial(5)
14 |    li a0, 5
15 |    jal x1, factorial
16 |
17 |    # print_int(t1)
18 |    li a7, 1
19 |    ecall
20 |
21 |    # return
22 |    lw ra, 0(sp)
23 |    addi sp, sp, 4
24 |    li a7, 10
25 |    ecall
26 |
27 |
28 |factorial:
29 |    # crear "stack frame" para $ra, $fp y una variable local
30 |    addi sp, sp, -12
31 |    sw ra, 8(sp)
32 |    sw fp, 4(sp)
33 |    addi fp, sp, 4
34 |
35 |    # if (a0 < 2):
36 |    #     return 1
```

Ejercicio guiado – Microcontrolador ESP32

CREATOR View Tools Help [Reset] [Step] [Run] [Stop] Sentinel [Settings] [Notifications] [Settings]

Address	User Instruction	Loaded Instructions
0x0 main	addi sp, sp, -4	addi sp, sp, -4 next
0x4	sw ra, 0(sp)	sw ra, 0(sp)
0x8	li a0, 5	addi a0, x0, 5
0xC	jal x1, factorial	jal x1, 28
0x10	li a7, 1	addi a7, x0, 1
0x14	ecall	ecall
0x18	lw ra, 0(sp)	lw ra, 0(sp)
0x1C	addi sp, sp, 4	addi sp, sp, 4
0x20	li a7, 10	addi a7, x0, 10
0x24	ecall	ecall
0x28 factorial	addi sp, sp, -12	addi sp, sp, -12
0x2C	sw ra, 8(sp)	sw ra, 8(sp)
0x30	sw fp, 4(sp)	sw fp, 4(sp)
0x34	addi fp, sp, 4	addi fp, sp, 4
0x38	li x5, 2	addi x5, x0, 2
0x3C	bge a0, t0, b_else	bge a0, t0, 12
0x40	li a0, 1	addi a0, x0, 1

Registers Memory Terminal Statistics

Control registers Hex ▾

pc 00000000	mepc 00000000	mcause 00000000	mtvec 00000000	mstatus 00000008
mip 00000000	mie 00000888	mscratch 00000000	mtime 00000000	mtimecmp 00000000

Integer registers Hex ▾

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

Floating point registers Hex ▾

Ejercicio guiado – Microcontrolador ESP32

1. Cargar el programa ensamblador en el microcontrolador:
 1. Abrir el cuadro de diálogo de la funcionalidad Flash:
 1. View → Simulator → Tools → Flash
 2. Complimentar el formulario web:
 1. Target Board Flash: **Local Device**
 2. Device type: **Espressif**
 3. Target Board: **ESP32 C6 (RISC-V)**
 4. Target port: **<puerto donde está conectado el microcontrolador>**
 5. Flash URL: <http://localhost:8080>
 3. Acción *Flash*

Ejercicio guiado – Microcontrolador ESP32

The screenshot shows the CREATOR IDE interface. In the foreground, a 'Target Board Flash' dialog box is open. The dialog has a title bar with a close button. It contains the following elements:

- Two tabs: 'Local Device' and 'Remote Device'.
- 'Device type:' section with radio buttons for 'Espressif' (selected) and 'SBC'.
- 'Target board:' dropdown menu showing 'ESP32-C6 (RISC-V)'.
- 'Target port (please verify the port on your computer):' text input field containing 'rfc2217://host.docker.internal:4000?ign_set_control'.
- 'Flash URL:' text input field containing 'http://localhost:8080'.
- A grid of action buttons: 'Flash' (blue, highlighted with a red box), 'Monitor' (blue), 'Clean' (red), 'Debug' (blue), 'Erase Flash' (red), and 'Stop' (blue).
- Footer text: 'For instructions on how to use this feature, please refer to the [CREATOR Gateway Documentation](#).'

The background shows an assembly code window with columns for 'Address', 'User Instruction', and 'Loaded In'. The code includes instructions like 'addi sp, sp, -4', 'sw ra, 0(sp)', 'li a0, 5', 'jal x1, factorial', 'ecall', 'lw ra, 0(sp)', 'addi sp, sp, 4', 'li a7, 10', 'ecall', 'addi sp, sp, -12', 'sw ra, 8(sp)', 'sw fp, 4(sp)', 'addi fp, sp, 4', 'li x5, 2', 'bge a0, t0, b_else', 'li a0, 1', 'beq x0, x0, b_efs', and 'sw a0, -4(fp)'. To the right, a memory dump window shows various registers and memory locations with their hexadecimal values, such as 'mcause: 00000000', 'mtvec: 00000000', 'mstatus: 00000003', etc.

Ejercicio guiado – Microcontrolador ESP32

```
creator-gateway-esp32-1 | esptool.py v4.8.1
creator-gateway-esp32-1 | Serial port rfc2217://host.docker.internal:4000?ign_set_control
creator-gateway-esp32-1 | Connecting...
creator-gateway-esp32-1 | Device PID identification is only supported on COM and /dev/ serial ports.
creator-gateway-esp32-1 | .
creator-gateway-esp32-1 | Chip is ESP32-C6FH4 (QFN32) (revision v0.2)
creator-gateway-esp32-1 | Features: WiFi 6, BT 5, IEEE802.15.4
creator-gateway-esp32-1 | Crystal is 40MHz
creator-gateway-esp32-1 | MAC: fc:01:2c:ff:fe:f9:17:bc
creator-gateway-esp32-1 | BASE MAC: fc:01:2c:f9:17:bc
creator-gateway-esp32-1 | MAC_EXT: ff:fe
creator-gateway-esp32-1 | Uploading stub...
creator-gateway-esp32-1 | Running stub...
creator-gateway-esp32-1 | Stub running...
creator-gateway-esp32-1 | Changing baud rate to 460800
creator-gateway-esp32-1 | Changed.
creator-gateway-esp32-1 | Configuring flash size...
creator-gateway-esp32-1 | Flash will be erased from 0x00000000 to 0x00005fff...
creator-gateway-esp32-1 | Flash will be erased from 0x00010000 to 0x00039fff...
creator-gateway-esp32-1 | Flash will be erased from 0x00008000 to 0x00008fff...
creator-gateway-esp32-1 | SHA digest in image updated
creator-gateway-esp32-1 | Compressed 22256 bytes to 13315...
creator-gateway-esp32-1 | Writing at 0x00000000... (100 %)
creator-gateway-esp32-1 | Wrote 22256 bytes (13315 compressed) at 0x00000000 in 0.9 seconds (effective 194.4 kbit/s)...
creator-gateway-esp32-1 | Hash of data verified.
creator-gateway-esp32-1 | Compressed 172016 bytes to 87445...
creator-gateway-esp32-1 | Writing at 0x0003726c... (100 %)
creator-gateway-esp32-1 | Wrote 172016 bytes (87445 compressed) at 0x00010000 in 4.1 seconds (effective 335.5 kbit/s)...
creator-gateway-esp32-1 | Hash of data verified.
creator-gateway-esp32-1 | Compressed 3072 bytes to 103...
creator-gateway-esp32-1 | Writing at 0x00008000... (100 %)
creator-gateway-esp32-1 | Wrote 3072 bytes (103 compressed) at 0x00008000 in 0.6 seconds (effective 39.5 kbit/s)...
creator-gateway-esp32-1 | Hash of data verified.
creator-gateway-esp32-1 | Leaving...
creator-gateway-esp32-1 | Hard resetting via RTS pin...
creator-gateway-esp32-1 | Done
creator-gateway-esp32-1 | 2026-06-09 14:28:44,416 - INFO - 172.17.0.1 - - [09/Jun/2026 14:28:44] "POST /flash HTTP/1.1" 200 -
```

Ejercicio guiado – Microcontrolador ESP32

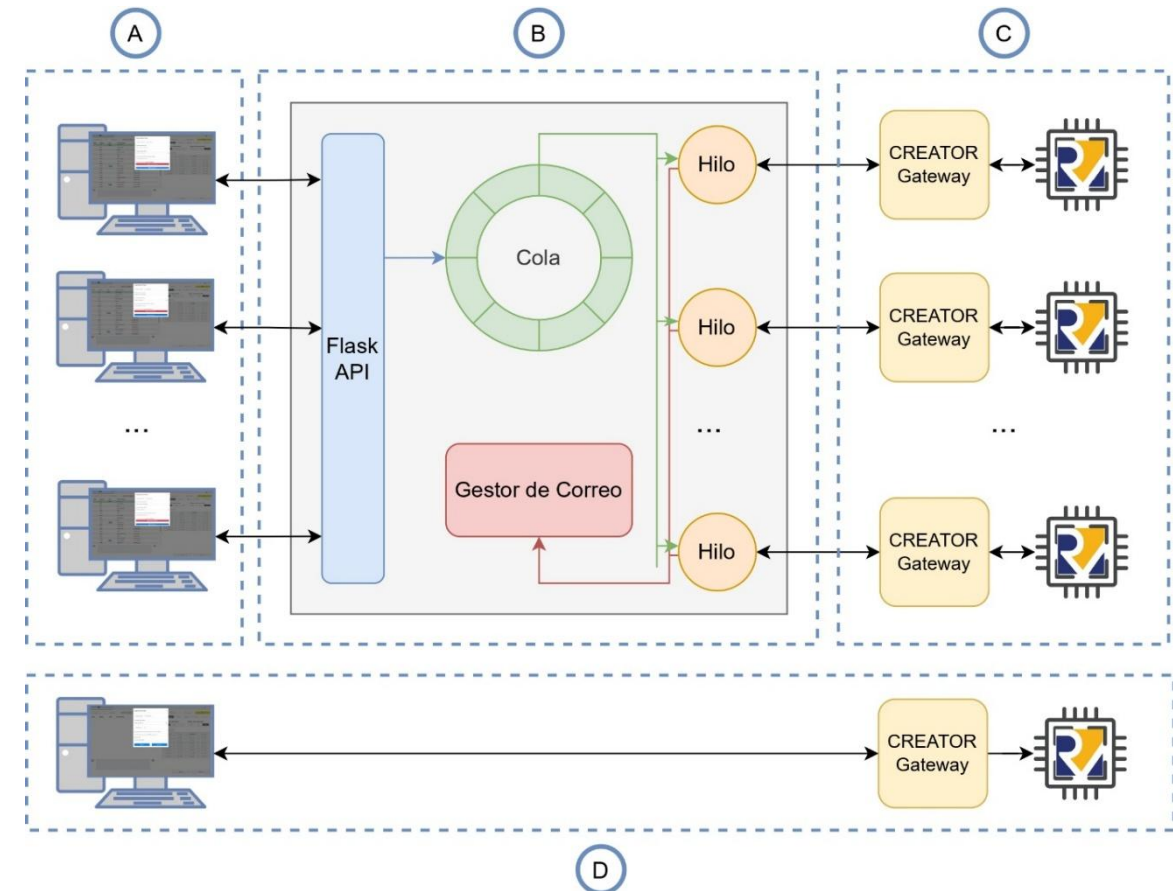
I. Ejecutar el programa ensamblador en el microcontrolador:

I. Acción *Monitor*

```
creator-gateway-esp32-1 | W (243) spi_flash: Detected size(4096k) larger than the size in the binary image header(2048k). Using the size in the binary image header.
creator-gateway-esp32-1 | I (257) sleep: Configure to isolate all GPIO pins in sleep state
creator-gateway-esp32-1 | I (263) sleep: Enable automatic switching of GPIO sleep configuration
creator-gateway-esp32-1 | I (270) coexist: coex firmware version: cbb41d7
creator-gateway-esp32-1 | I (285) coexist: coexist rom version 5b8dcfa
creator-gateway-esp32-1 | I (285) main_task: Started on CPU0
creator-gateway-esp32-1 | I (285) main_task: Calling app_main()
creator-gateway-esp32-1 | Started program...
creator-gateway-esp32-1 | -----
creator-gateway-esp32-1 | 120
creator-gateway-esp32-1 | Finished program: 16518 cycles
creator-gateway-esp32-1 | -----
creator-gateway-esp32-1 | I (300) main_task: Returned from app_main()
creator-gateway-esp32-1 | 2026-06-09 14:30:20,180 - INFO - 172.17.0.1 - - [09/Jun/2026 14:30:20] "OPTIONS /stopmonitor HTTP/1.1" 200 -
creator-gateway-esp32-1 | 2026-06-09 14:30:20,196 - INFO - 172.17.0.1 - - [09/Jun/2026 14:30:20] "POST /stopmonitor HTTP/1.1" 200 -
creator-gateway-esp32-1 | 2026-06-09 14:30:20,248 - INFO - 172.17.0.1 - - [09/Jun/2026 14:30:20] "POST /monitor HTTP/1.1" 200 -
```

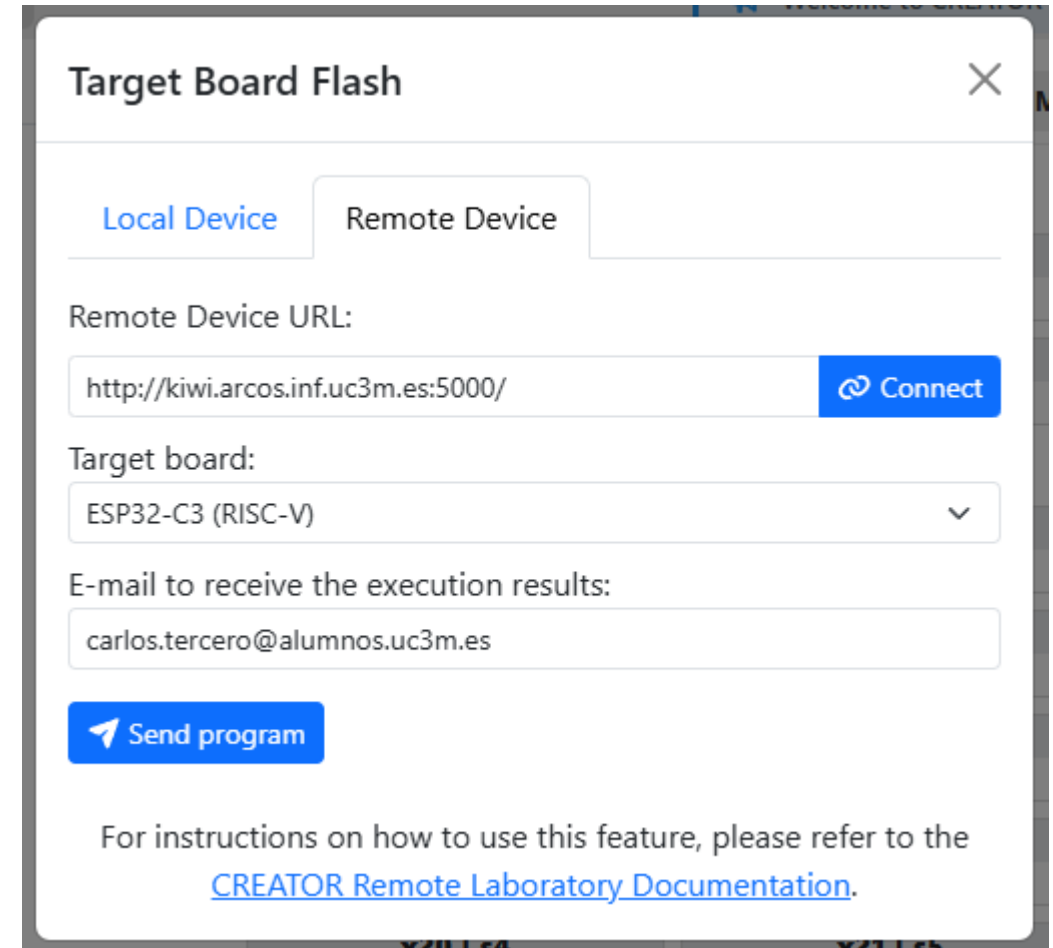
Servicio de laboratorio remoto

- ▶ *Framework* Flask de Python3
- ▶ Evita tener que comprar el dispositivo *hardware* y configurar el entorno
- ▶ Permite ejecutar los programas ensamblador sobre diferentes dispositivos *hardware*
- ▶ Múltiples usuarios trabajan sobre los mismos dispositivos *hardware*
 - ▶ Sistema de colas
 - ▶ Envío de los resultados por correo electrónico
- ▶ Contenedor *Docker*



Servicio de laboratorio remoto

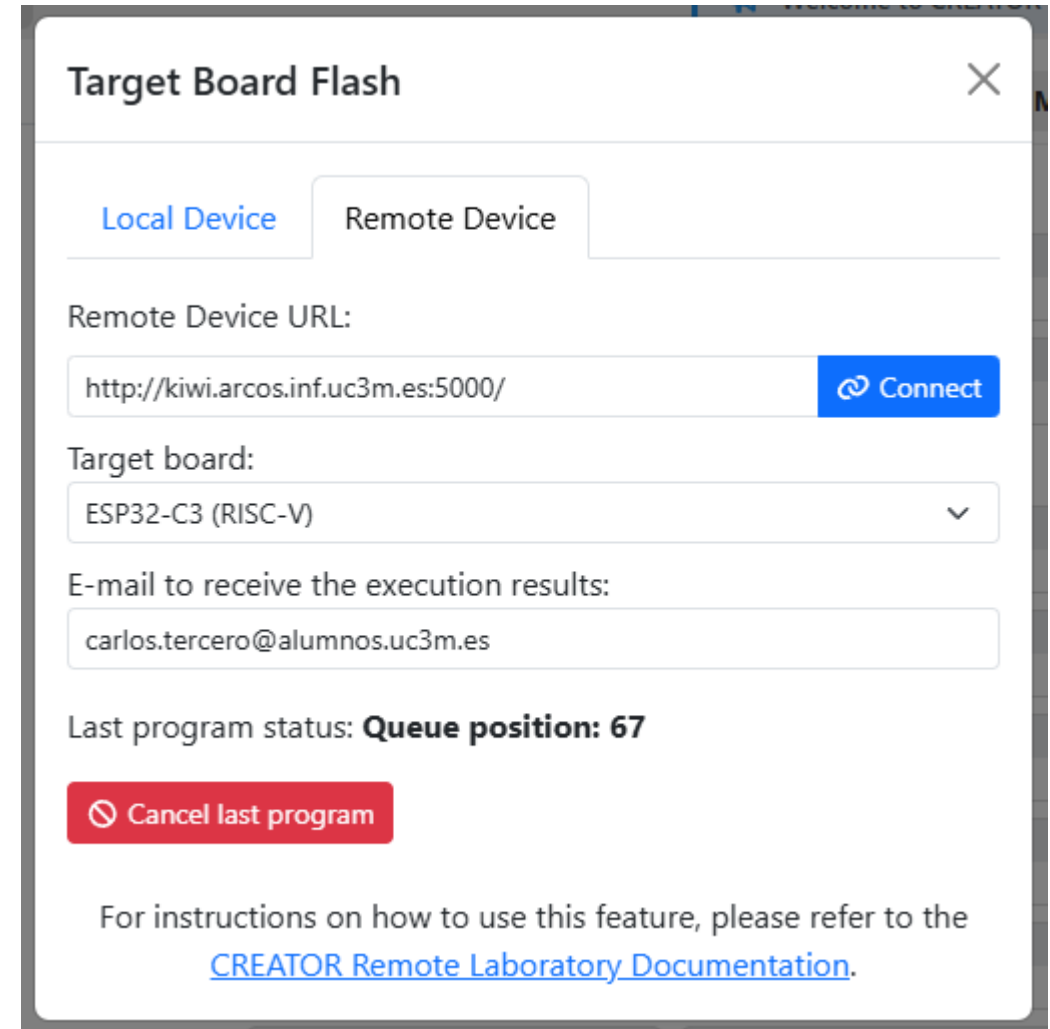
- ▶ *Framework* Flask de Python3
- ▶ Evita tener que comprar el dispositivo *hardware* y configurar el entorno
- ▶ Permite ejecutar los programas ensamblador sobre diferentes dispositivos *hardware*
- ▶ Múltiples usuarios trabajan sobre los mismos dispositivos *hardware*
 - ▶ Sistema de colas
 - ▶ Envío de los resultados por correo electrónico
- ▶ Contenedor *Docker*



The screenshot shows a web interface titled "Target Board Flash" with a close button (X) in the top right corner. It features two tabs: "Local Device" (selected) and "Remote Device". Under the "Remote Device" tab, there is a "Remote Device URL:" field containing "http://kiwi.arcos.inf.uc3m.es:5000/" and a blue "Connect" button with a refresh icon. Below that is a "Target board:" dropdown menu currently set to "ESP32-C3 (RISC-V)". An "E-mail to receive the execution results:" field contains "carlos.tercero@alumnos.uc3m.es". A blue "Send program" button with a paper plane icon is positioned below the email field. At the bottom, there is a note: "For instructions on how to use this feature, please refer to the [CREATOR Remote Laboratory Documentation](#)."

Servicio de laboratorio remoto

- ▶ *Framework* Flask de Python3
- ▶ Evita tener que comprar el dispositivo *hardware* y configurar el entorno
- ▶ Permite ejecutar los programas ensamblador sobre diferentes dispositivos *hardware*
- ▶ Múltiples usuarios trabajan sobre los mismos dispositivos *hardware*
 - ▶ Sistema de colas
 - ▶ Envío de los resultados por correo electrónico
- ▶ Contenedor *Docker*



Target Board Flash

Local Device Remote Device

Remote Device URL:

Target board:


E-mail to receive the execution results:

Last program status: **Queue position: 67**

For instructions on how to use this feature, please refer to the [CREATOR Remote Laboratory Documentation](#).

Servicio de laboratorio remoto

- ▶ *Framework* Flask de Python3
- ▶ Evita tener que comprar el dispositivo *hardware* y configurar el entorno
- ▶ Permite ejecutar los programas ensamblador sobre diferentes dispositivos *hardware*
- ▶ Múltiples usuarios trabajan sobre los mismos dispositivos *hardware*
 - ▶ Sistema de colas
 - ▶ Envío de los resultados por correo electrónico
- ▶ Contenedor *Docker*

[CREATOR] Remote device results  
Externo > Recibidos x



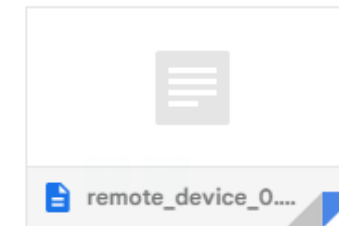
creator.arcos.inf.uc3m.es@gm...  11:41 (hace 2 minutos)   
para mí ▾

Remote device ID=0 has been successfully completed, the execution results are attached.

Sincerely,
CREATOR Team

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

1 archivo adjunto • Analizado por Gmail 



 Responder

 Reenviar

Servicio de laboratorio remoto

► Despliegue del servicio de laboratorio:

1. Descargar la imagen Docker
<https://hub.docker.com/r/creatorsim/creator-remote-lab>

2. Crear la siguiente estructura de ficheros

```
creator-remote-lab/  
├─ compose.yaml  
├─ results/  
├─ config/  
| └─ deployment.json
```

3. Crear el fichero *compose.yaml*

```
services:  
  creator-remote-lab:  
    image: creatorsim/creator-remote-lab:latest  
    volumes:  
      - ./config:/app/config  
      - ./results:/app/results  
    ports:  
      - "5000:5000"  
    environment:  
      - EMAIL=test@inf.uc3m.es # google mail address  
      - PASSW=abcdefghijklmnop # google app password  
  
  target-0:  
    image: creatorsim/creator-gateway-esp32:latest  
    devices:  
      - /dev/ttyUSB0  
    stdin_open: true  
    tty: true  
  
  target-1:  
    image: creatorsim/creator-gateway-esp32:latest  
    devices:  
      - /dev/ttyUSB1  
    stdin_open: true  
    tty: true
```

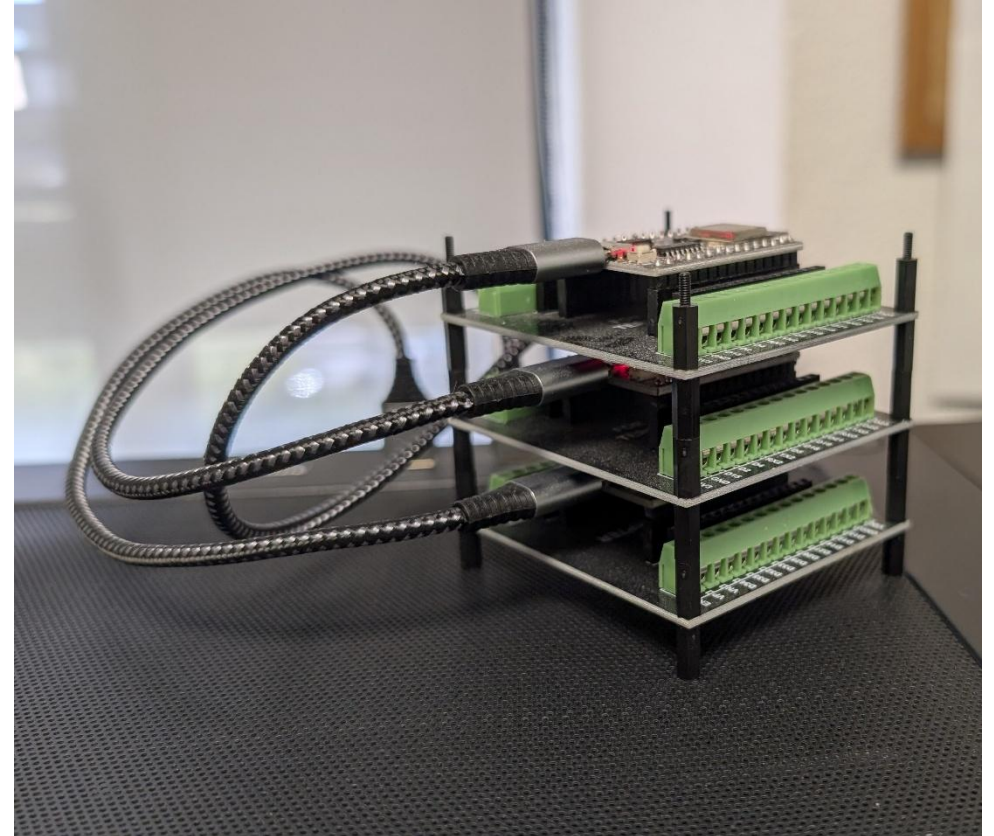
Servicio de laboratorio remoto

► Despliegue del servicio de laboratorio:

4. Crear el fichero *config/deployment.json*

```
{
  "target-0": {
    "target_board": "esp32c3",
    "target_port": "/dev/ttyUSB0",
    "target_url": "http://target-0:8080"
  },
  "target-1": {
    "target_board": "esp32c3",
    "target_port": "/dev/ttyUSB1",
    "target_url": "http://target-1:8080"
  }
}
```

5. Desplegar con *Docker Compose* docker compose up



<http://kiwi.arcos.inf.uc3m.es:8080/creator>

<http://kiwi.arcos.inf.uc3m.es:5000>

Integración con Arduino (CREATino)

- ▶ Biblioteca nativa en CREATOR
- ▶ Permite la simulación de los puertos GPIO del hardware en la interfaz web de CREATOR
- ▶ Permite utilizar los puertos GPIO de los dispositivos *hardware* en los programas ensamblador

The screenshot displays the CREATOR IDE interface. The main window shows assembly code with columns for Address, User Instruction, and Loaded Instructions. The instruction at address 0x194, `jal ra, digitalWrite`, is highlighted in green and labeled 'next'. A 'loop' label is visible at address 0x184. The Arduino Function Tracer on the right shows a sequence of function calls: `initArduino()`, `pinMode(7, 3)`, `digitalWrite(7, 1)`, `delay(1000)`, and `digitalWrite(7, 0)`. The Hardware View on the far right shows a board selected as 'ESP32-C6 DevKit' and a list of GPIO pins (GPIO0 to GPIO23) with their current status (all 0). A physical ESP32-C6 DevKit board is shown in the center of the hardware view.

Address	User Instruction	Loaded Instructions
0x174	<code>jal ra, pinMode</code>	<code>jal ra, -364</code>
0x178	<code>lw ra, 0(sp)</code>	<code>lw ra, 0(sp)</code>
0x17C	<code>addi sp, sp, 4</code>	<code>addi sp, sp, 4</code>
0x180	<code>jr ra</code>	<code>jalr x0, 0(ra)</code>
0x184	<code>li a0, 7</code>	<code>addi a0, x0, 7</code>
0x188	<code>li a1, 0x1</code>	<code>addi a1, x0, 1</code>
0x18C	<code>addi sp, sp, -4</code>	<code>addi sp, sp, -4</code>
0x190	<code>sw ra, 0(sp)</code>	<code>sw ra, 0(sp)</code>
0x194	<code>jal ra, digitalWrite</code>	<code>jal ra, -392</code>
0x198	<code>lw ra, 0(sp)</code>	<code>lw ra, 0(sp)</code>
0x19C	<code>addi sp, sp, 4</code>	<code>addi sp, sp, 4</code>
0x1A0	<code>la a0, time</code>	<code>auipc a0, 512</code>
0x1A4		<code>addi a0, a0, -416</code>
0x1A8	<code>lw a0, 0(a0)</code>	<code>lw a0, 0(a0)</code>
0x1AC	<code>addi sp, sp, -4</code>	<code>addi sp, sp, -4</code>
0x1B0	<code>sw ra, 0(sp)</code>	<code>sw ra, 0(sp)</code>
0x1B4	<code>jal ra, delay</code>	<code>jal ra, -256</code>
0x1B8	<code>lw ra, 0(sp)</code>	<code>lw ra, 0(sp)</code>
0x1BC	<code>addi sp, sp, 4</code>	<code>addi sp, sp, 4</code>
0x1C0	<code>li a0, 7</code>	<code>addi a0, x0, 7</code>
0x1C4	<code>li a1, 0x0</code>	<code>addi a1, x0, 0</code>
0x1C8	<code>addi sp, sp, -4</code>	<code>addi sp, sp, -4</code>

Ejercicio guiado – CREATino

The screenshot shows the CREATino IDE interface. The menu bar includes 'CREATOR', 'View', 'File', 'Library', and 'Help'. The 'Library' menu is open, showing options: 'Load Library...', 'Save as Library...', 'Add Arduino Library', and 'Remove Library...'. The 'Add Arduino Library' option is highlighted with a red box and a circled '2'. The 'Assemble & Run' button in the toolbar is also highlighted with a red box and a circled '3'. The main editor displays assembly code for a program named 'Creatino Example'. The code includes sections for '.data', '.text', 'main', 'setup', and 'loop'. The 'loop' section contains instructions for digital writing, delay, and LED control.

```
1 # Creatino Example
2 .data
3     time:
4         .word 1000
5 .text
6
7 main:
8     addi sp, sp, -4
9     sw ra, 0(sp)
10    jal ra, initArduino
11    jal ra, setup
12    lw ra, 0(sp)
13    addi sp, sp, 4
14    j loop
15
16 setup:
17     nop
18
19 loop:
20     #digitalWrite(LED_BUILTIN, HIGH);
21     li a0, 8
22     li a1, 0x1
23     addi sp, sp, -4
24     sw ra, 0(sp)
25     jal ra, digitalWrite
26     lw ra, 0(sp)
27     addi sp, sp, 4
28     #delay(1000);
29     la a0, time
30     lw a0, 0(a0)
31     addi sp, sp, -4
32     sw ra, 0(sp)
33     jal ra, delay
34     lw ra, 0(sp)
35     addi sp, sp, 4
36     #digitalWrite(LED_BUILTIN, LOW);
37     li a0, 8
```

Ejercicio guiado – CREATino

The screenshot displays the CREATino IDE interface. At the top, there is a menu bar with 'CREATOR View Tools Help' and a toolbar with 'Reset', 'Step', 'Run', 'Stop', and 'Sentinel' buttons. On the right side of the toolbar, there are settings, notification, and help icons. A red box highlights the 'Arduino' button in the top right corner.

The main interface is divided into three panels:

- Assembly Code Panel (Left):** A table with columns 'Address', 'User Instruction', and 'Loaded Instructions'. The current instruction at address 0x164 is highlighted in green and labeled 'loop'.

Address	User Instruction	Loaded Instructions
0xf0	serial_write	<<Hidden>>
0xf4	serial_printf	<<Hidden>>
0x144	main	addi sp, sp, -4
0x148		sw ra, 0(sp)
0x14C		jal ra, initArduino
0x150		jal ra, setup
0x154		lw ra, 0(sp)
0x158		addi sp, sp, 4
0x15C		j loop
0x160	setup	nop
0x164	loop	li a0,8
0x168		li a1, 0x1
0x16C		addi sp, sp, -4
0x170		sw ra, 0(sp)
0x174		jal ra, digitalWrite
0x178		lw ra, 0(sp)
0x17C		addi sp, sp, 4
0x180		la a0, time
0x184		addi a0, a0, -384
- Arduino Function Tracer Panel (Middle):** Shows a list of function calls: `> initArduino()`, `> digitalWrite(8, 1)`, `> delay(1000)`, `> digitalWrite(8, 0)`, `> delay(1000)`, `> digitalWrite(8, 1)`, `> delay(1000)`, `> digitalWrite(8, 0)`, and `> delay(1000)`. A cursor is visible at the end of the last line.
- Hardware View Panel (Right):** Displays a schematic of an ESP32-C6 DevKit. The board is labeled 'ESP32-C6'. A dropdown menu shows 'BOARD SELECTED: ESP32-C6 DevKit'. Below the board, there are 16 GPIO pins, each with a blue button and a value of 0: GPIO4, GPIO5, GPIO6, GPIO7, GPIO0, GPIO1, GPIO8, GPIO10, GPIO11, GPIO2, GPIO3, GPIO15, GPIO23, GPIO22, GPIO21, GPIO20, GPIO19, GPIO18, GPIO9, GPIO17, GPIO9, GPIO13, and GPIO12.

Ejercicio guiado – CREATino

1. Cargar el programa ensamblador en el microcontrolador:
 1. Abrir el cuadro de diálogo de la funcionalidad Flash:
 1. View → Simulator → Tools → Flash
 2. Complimentar el formulario web:
 1. Target Board Flash: **Local Device**
 2. Device type: **Espressif**
 3. Target Board: **ESP32 C6 (RISC-V)**
 4. Target port: **<puerto donde está conectado el microcontrolador>**
 5. Flash URL: <http://localhost:8080>
 3. Acción *Flash*

Ejercicio guiado – CREATino

The screenshot shows the CREATOR IDE interface. In the background, there is a table of assembly instructions and a hardware view of an ESP32-C6 board with various GPIO pins labeled. A modal dialog titled "Target Board Flash" is centered on the screen. The dialog has two tabs: "Local Device" and "Remote Device". Under "Device type", "Espressif" is selected. The "Target board:" dropdown is set to "ESP32-C6 (RISC-V)". The "Target port (please verify the port on your computer):" field contains "rfc2217://host.docker.internal:4000?ign_set_control". The "Flash URL:" field contains "http://localhost:8080". A red box highlights the "Flash" button. Other buttons include "Monitor", "Clean", "Erase Flash", "Debug", and "Stop". At the bottom of the dialog, there is a link to "CREATOR Gateway Documentation".

#	Address	User Instruction	Loaded
	0x194	jal ra, delay	jal ra, -2
	0x198	lw ra, 0(sp)	lw ra, 0(s
	0x19C	addi sp, sp, 4	addi sp, :
	0x1A0	li a0,8	addi a0,
	0x1A4	li a1, 0x0	addi a1,
	0x1A8	addi sp, sp, -4	addi sp,
	0x1AC	sw ra, 0(sp)	sw ra, 0(s
	0x1B0	jal ra, digitalWrite	jal ra, -4
	0x1B4	lw ra, 0(sp)	lw ra, 0(s
	0x1B8	addi sp, sp, 4	addi sp,
	0x1BC	la a0, time	auipc a0,
	0x1C0		addi a0,
	0x1C4	lw a0, 0(a0)	lw a0, 0(s
	0x1C8	addi sp, sp, -4	addi sp,
	0x1CC	sw ra, 0(sp)	sw ra, 0(s
	0x1D0	jal ra, delay	jal ra, -2
	0x1D4	lw ra, 0(sp)	lw ra, 0(s
	0x1D8	addi sp, sp, 4	addi sp,

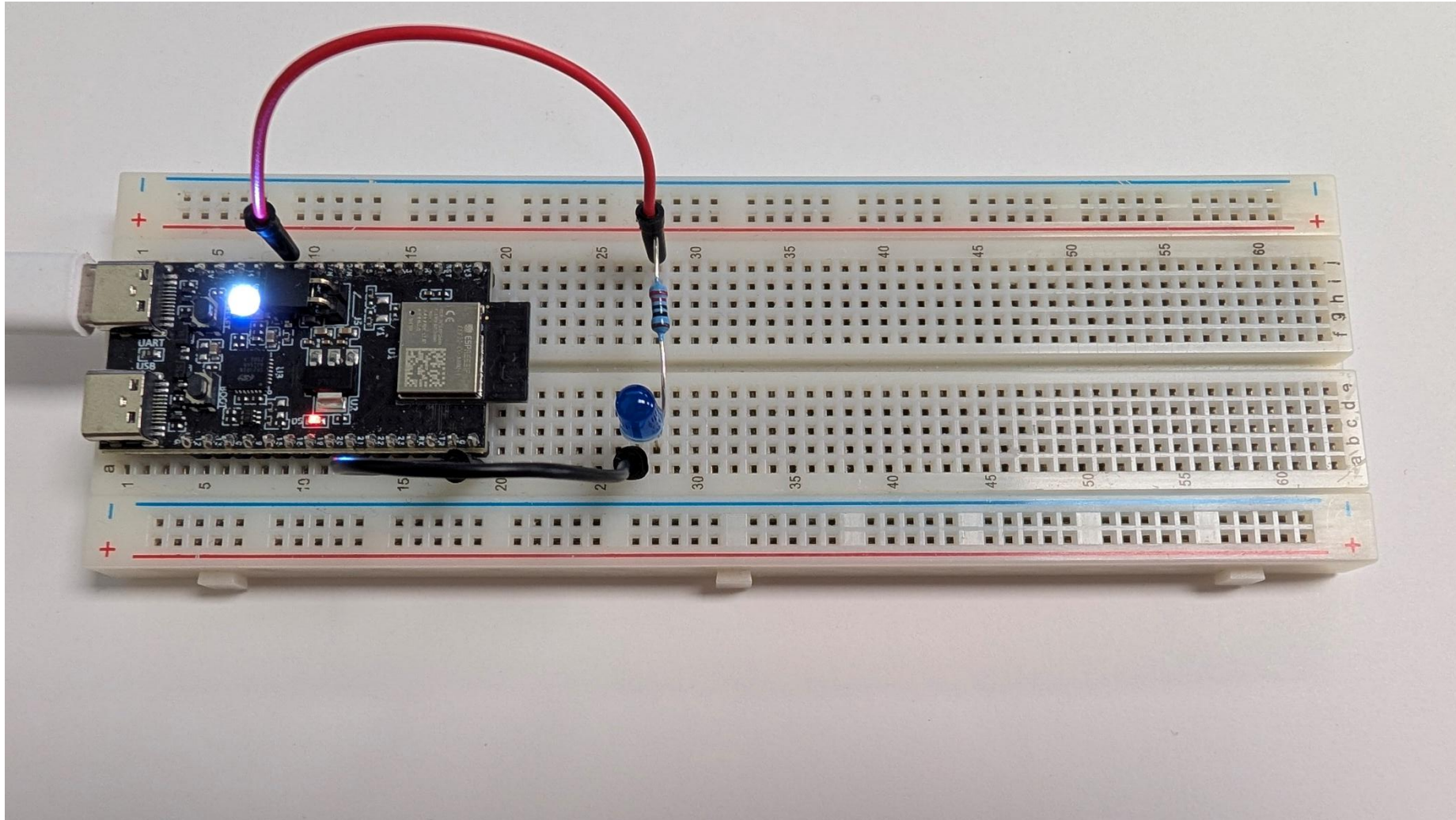
Hardware View
BOARD SELECTED: ESP32-C6 DevKit

GPIO4 0, GPIO5 0, GPIO6 0, GPIO7 0, GPIO8 0, GPIO10 0, GPIO11 0, GPIO1 0, GPIO8 0, GPIO10 0, GPIO11 0, GPIO2 0, GPIO3 0, GPIO15 0, GPIO23 0, GPIO22 0, GPIO21 0, GPIO20 0, GPIO19 0, GPIO18 0, GPIO9 0, GPIO17 0, GPIO9 0, GPIO13 0, GPIO12 0

Ejercicio guiado – CREATino

```
-hard_reset write_flash --flash_mode dio --flash_freq 80m --flash_size 2MB 0x0 bootloader/bootloader.bin 0x10000 hello_world.bin 0x8000 partition_table/partition-table.bin
creator-gateway-esp32-1 | esptool.py v4.8.1
creator-gateway-esp32-1 | Serial port rfc2217://host.docker.internal:4000?ign_set_control
creator-gateway-esp32-1 | Connecting...
creator-gateway-esp32-1 | Device PID identification is only supported on COM and /dev/ serial ports.
creator-gateway-esp32-1 | .
creator-gateway-esp32-1 | Chip is ESP32-C6FH4 (QFN32) (revision v0.2)
creator-gateway-esp32-1 | Features: WiFi 6, BT 5, IEEE802.15.4
creator-gateway-esp32-1 | Crystal is 40MHz
creator-gateway-esp32-1 | MAC: fc:01:2c:ff:fe:f9:17:bc
creator-gateway-esp32-1 | BASE MAC: fc:01:2c:f9:17:bc
creator-gateway-esp32-1 | MAC_EXT: ff:fe
creator-gateway-esp32-1 | Uploading stub...
creator-gateway-esp32-1 | Running stub...
creator-gateway-esp32-1 | Stub running...
creator-gateway-esp32-1 | Changing baud rate to 460800
creator-gateway-esp32-1 | Changed.
creator-gateway-esp32-1 | Configuring flash size...
creator-gateway-esp32-1 | Flash will be erased from 0x00000000 to 0x00005fff...
creator-gateway-esp32-1 | Flash will be erased from 0x00010000 to 0x00067fff...
creator-gateway-esp32-1 | Flash will be erased from 0x00008000 to 0x00008fff...
creator-gateway-esp32-1 | SHA digest in image updated
creator-gateway-esp32-1 | Compressed 22256 bytes to 13315...
Writing at 0x00000000... (100 %)
creator-gateway-esp32-1 | Wrote 22256 bytes (13315 compressed) at 0x00000000 in 0.9 seconds (effective 195.5 kbit/s)...
creator-gateway-esp32-1 | Hash of data verified.
creator-gateway-esp32-1 | Compressed 360160 bytes to 147934...
Writing at 0x00067ad8... (100 %)
creator-gateway-esp32-1 | Wrote 360160 bytes (147934 compressed) at 0x00010000 in 6.7 seconds (effective 432.4 kbit/s)...
creator-gateway-esp32-1 | Hash of data verified.
creator-gateway-esp32-1 | Compressed 3072 bytes to 103...
Writing at 0x00008000... (100 %)
creator-gateway-esp32-1 | Wrote 3072 bytes (103 compressed) at 0x00008000 in 0.6 seconds (effective 39.8 kbit/s)...
creator-gateway-esp32-1 | Hash of data verified.
creator-gateway-esp32-1 | Leaving...
creator-gateway-esp32-1 | Hard resetting via RTS pin...
creator-gateway-esp32-1 | Done
creator-gateway-esp32-1 | 2026-06-09 14:38:47,448 - INFO - 172.17.0.1 - - [09/Jun/2026 14:38:47] "POST /flash HTTP/1.1" 200 -
```

Ejercicio guiado – CREATino



Ejercicio guiado – CREATino

- ▶ **Ejercicio 1: Desarrollo de un semáforo con el LED integrado**
 - ▶ Hay que utilizar `rgbLedWrite(pin, red, green, blue)`
 - ▶ El amarillo se forma con la mezcla de rojo y verde
- ▶ **Ejercicio 2: Desarrollo de un semáforo utilizando 3 LEDs**
 - ▶ Recuerde utilizar una resistencia de 220 por cada LED



Jornadas Sarteco

17 – 19 Junio 2026, Madrid



CREATOR: entorno de desarrollo integrado para la docencia y la investigación en arquitecturas RISC-V

Félix García Carballeira, Diego Camarmas Alonso, Alejandro Calderón Mateos

uc3m | Universidad **Carlos III** de Madrid



Grupo ARCOS