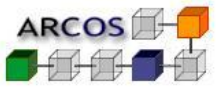
Jornadas Sarteco

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

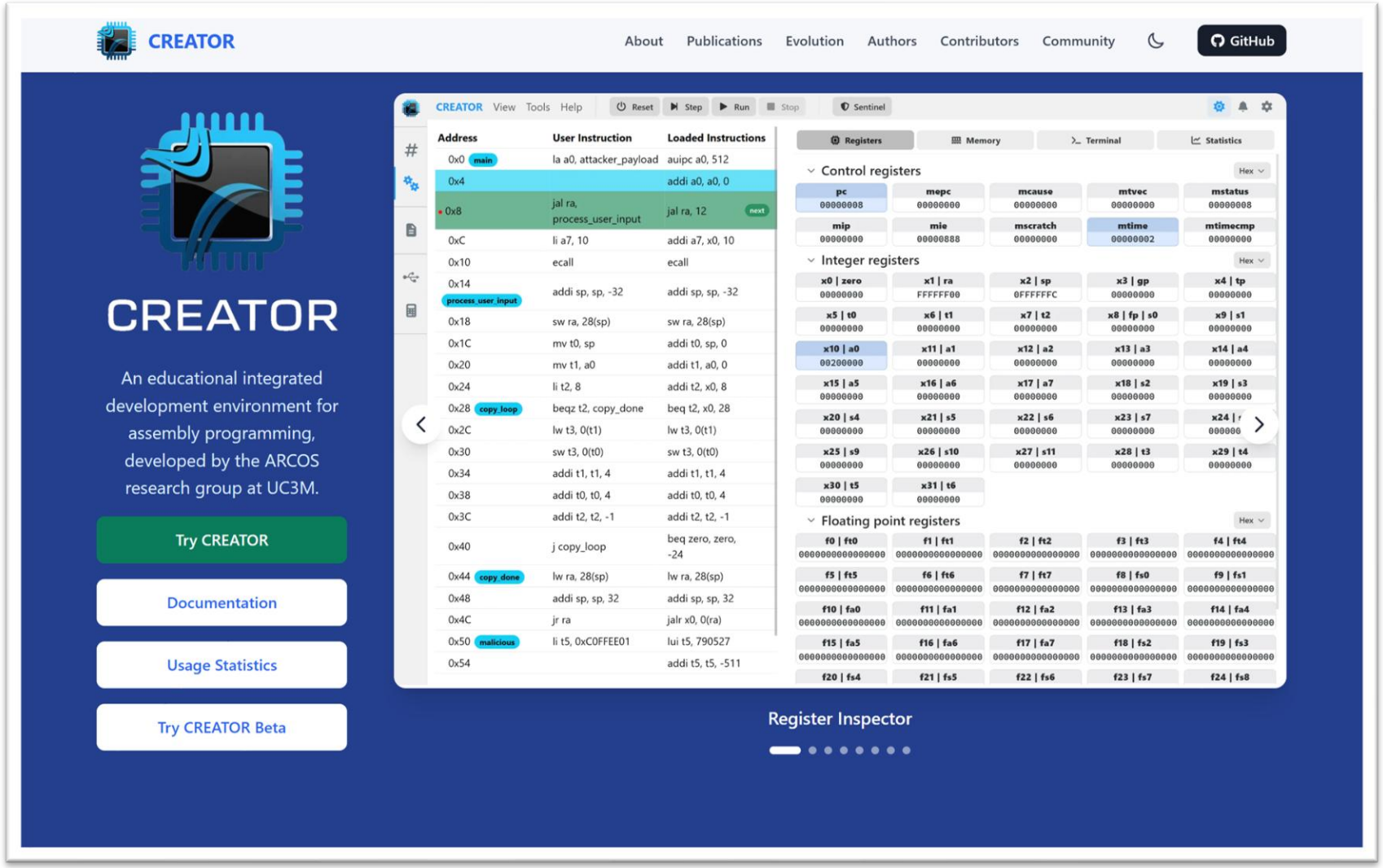
Motivación de CREATOR
didaCtic and geneRic assEmbly prograMming simulaTOR

- 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

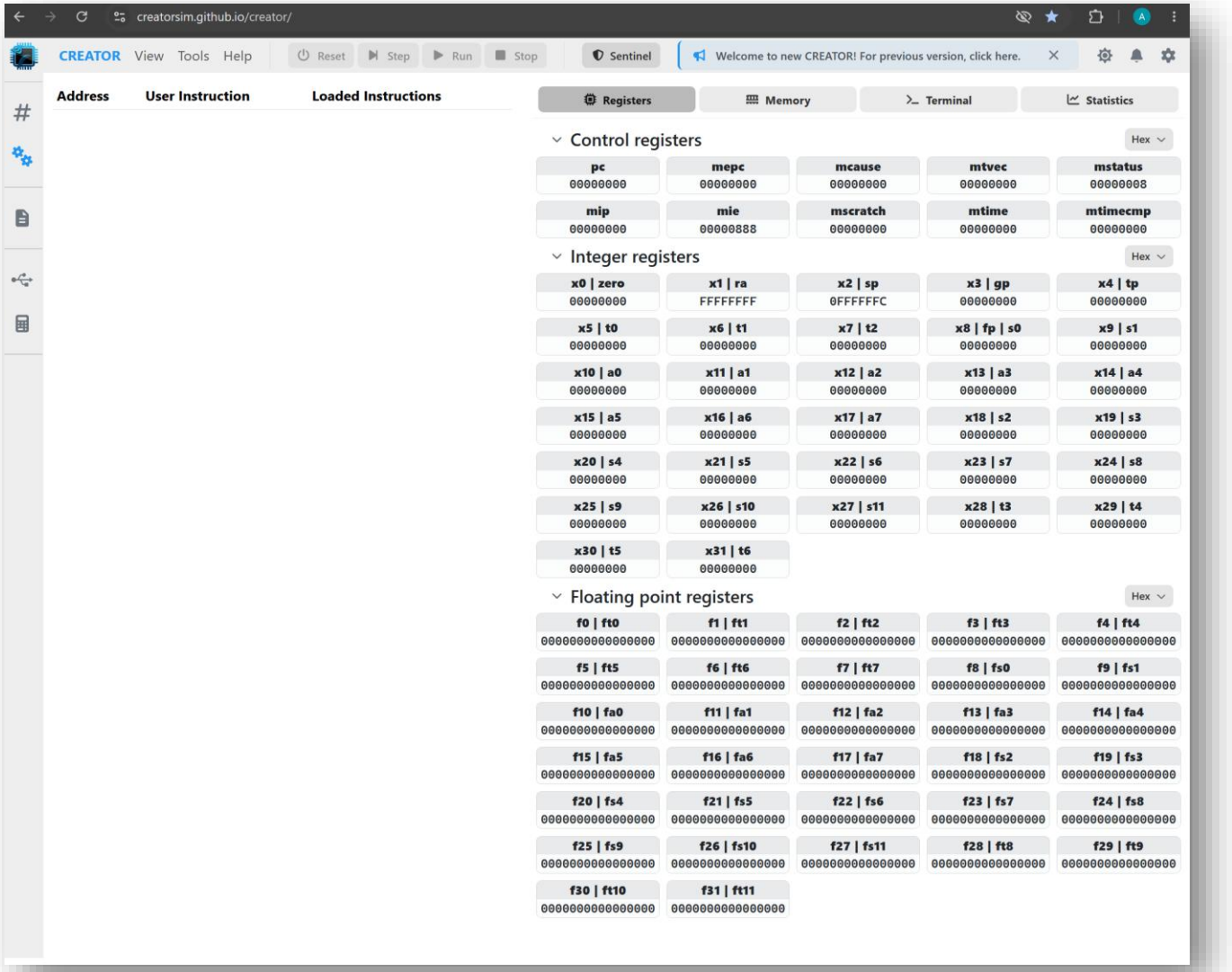
- Facilidades para entender:
  - La representación de datos e instrucciones
  - La diferencia entre instrucciones y pseudoinstrucciones
  - La carga de un programa en memoria
  - El flujo de ejecución de un programa en ensamblador conociendo en todo momento la instrucción en curso y la siguiente (útil 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



https://creatorsim.github.io/

CREATOR

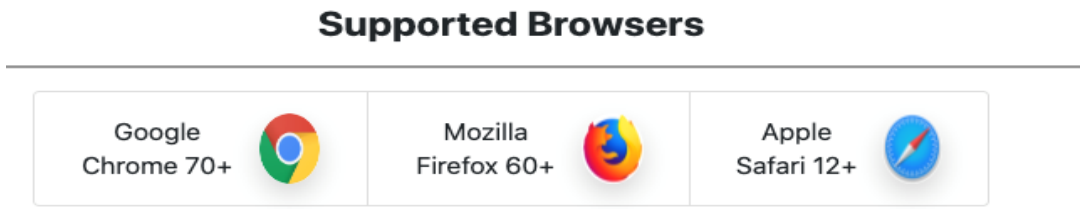


https://creatorsim.github.io/creator/

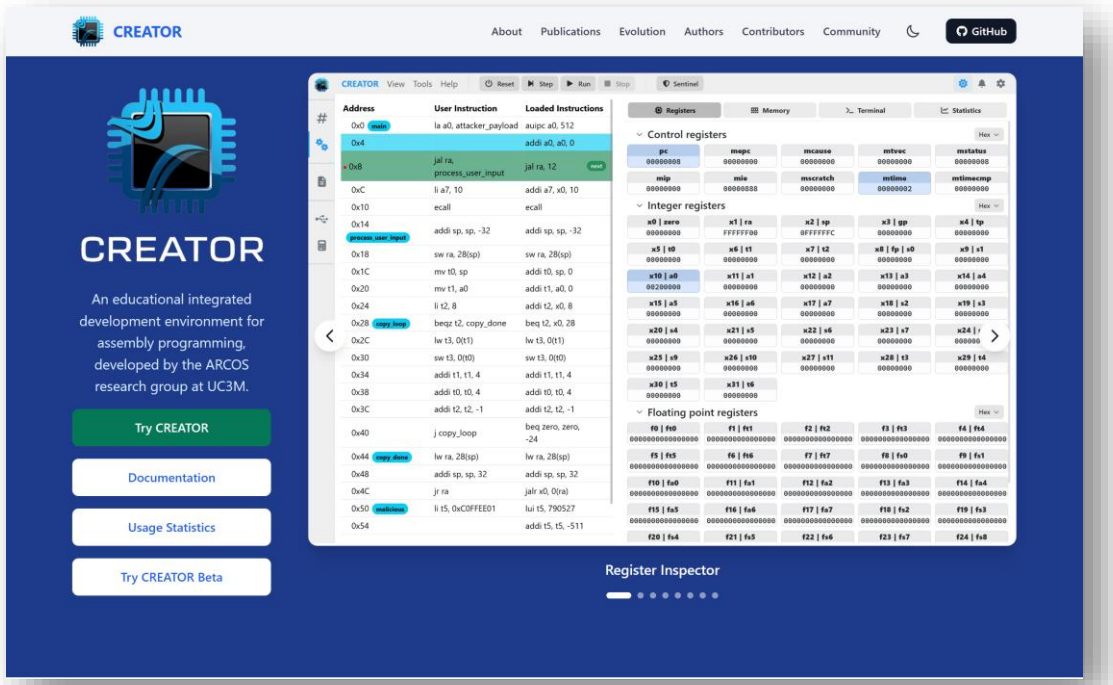
ARCOS

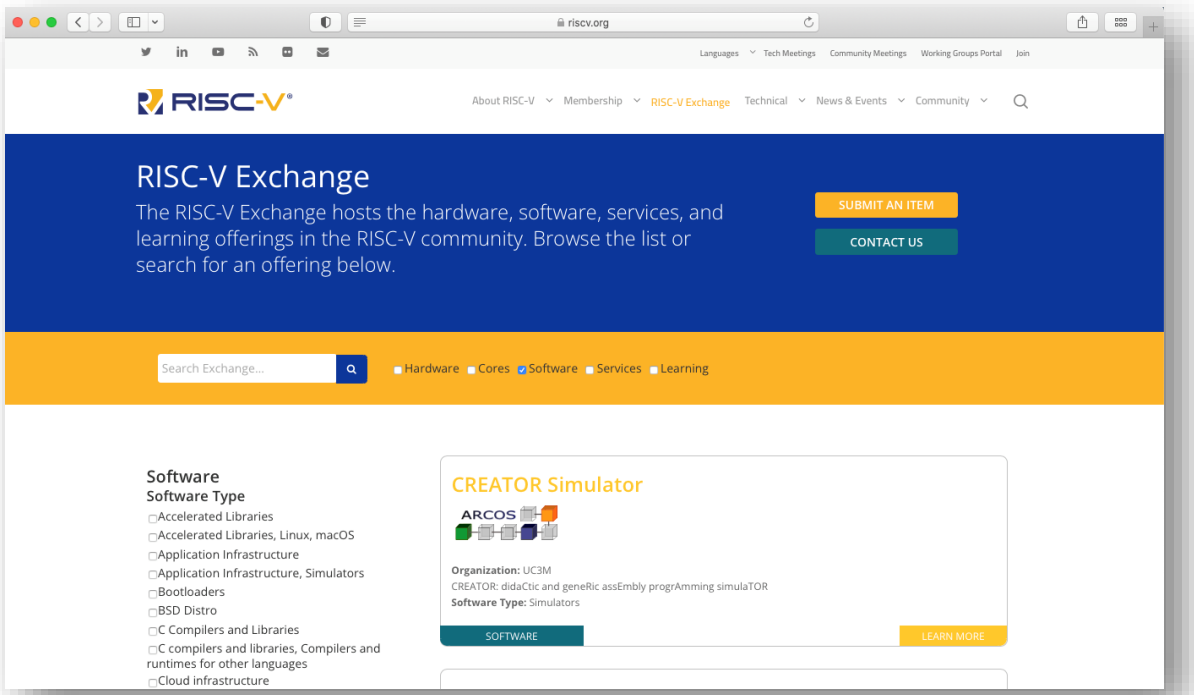
Características

- 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



Disponibilidad





https://creatorsim.github.io

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

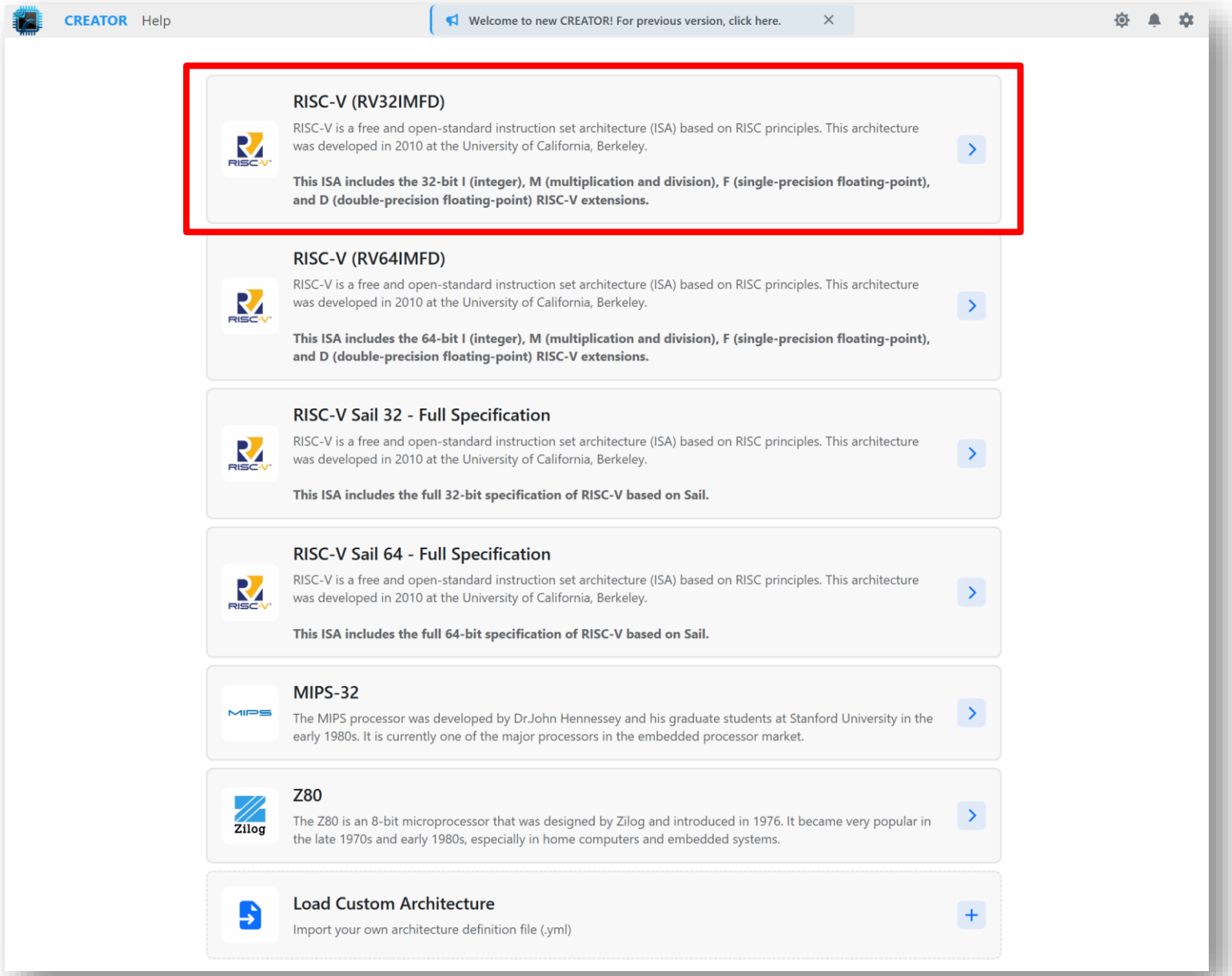
Llamadas al sistema

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

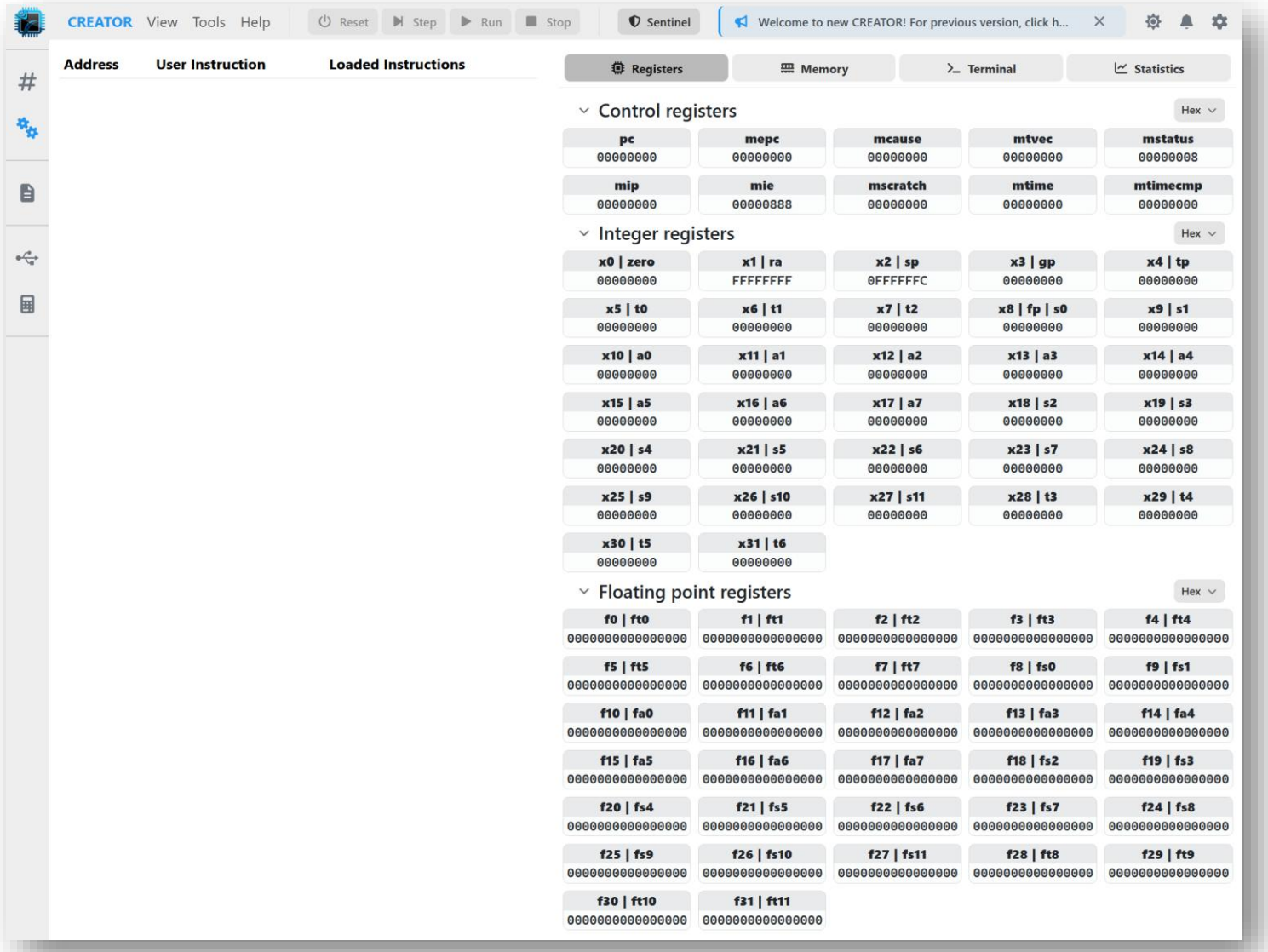
Directivas soportadas

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

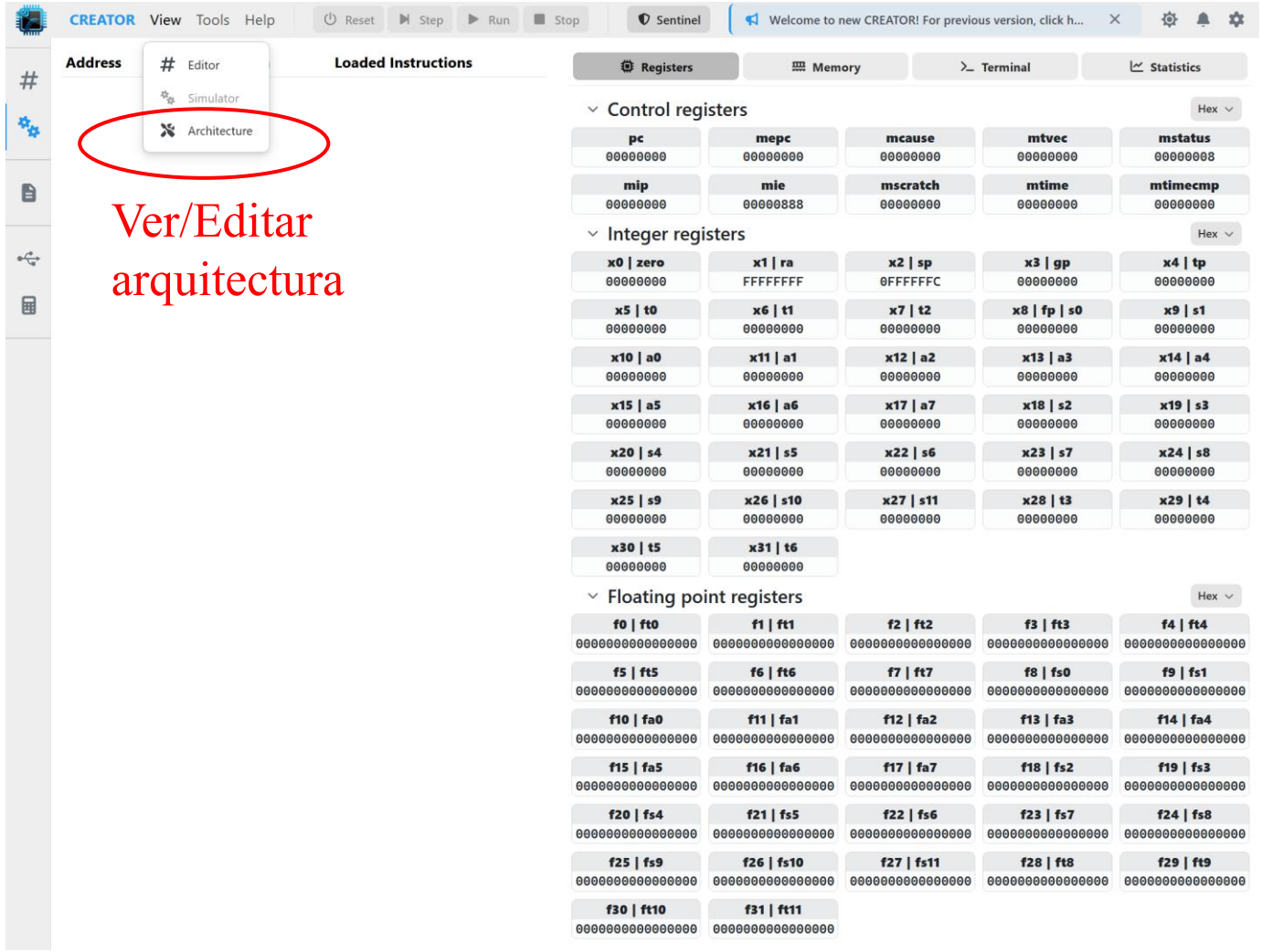
CREATOR (RISC-V)



Pantalla inicial

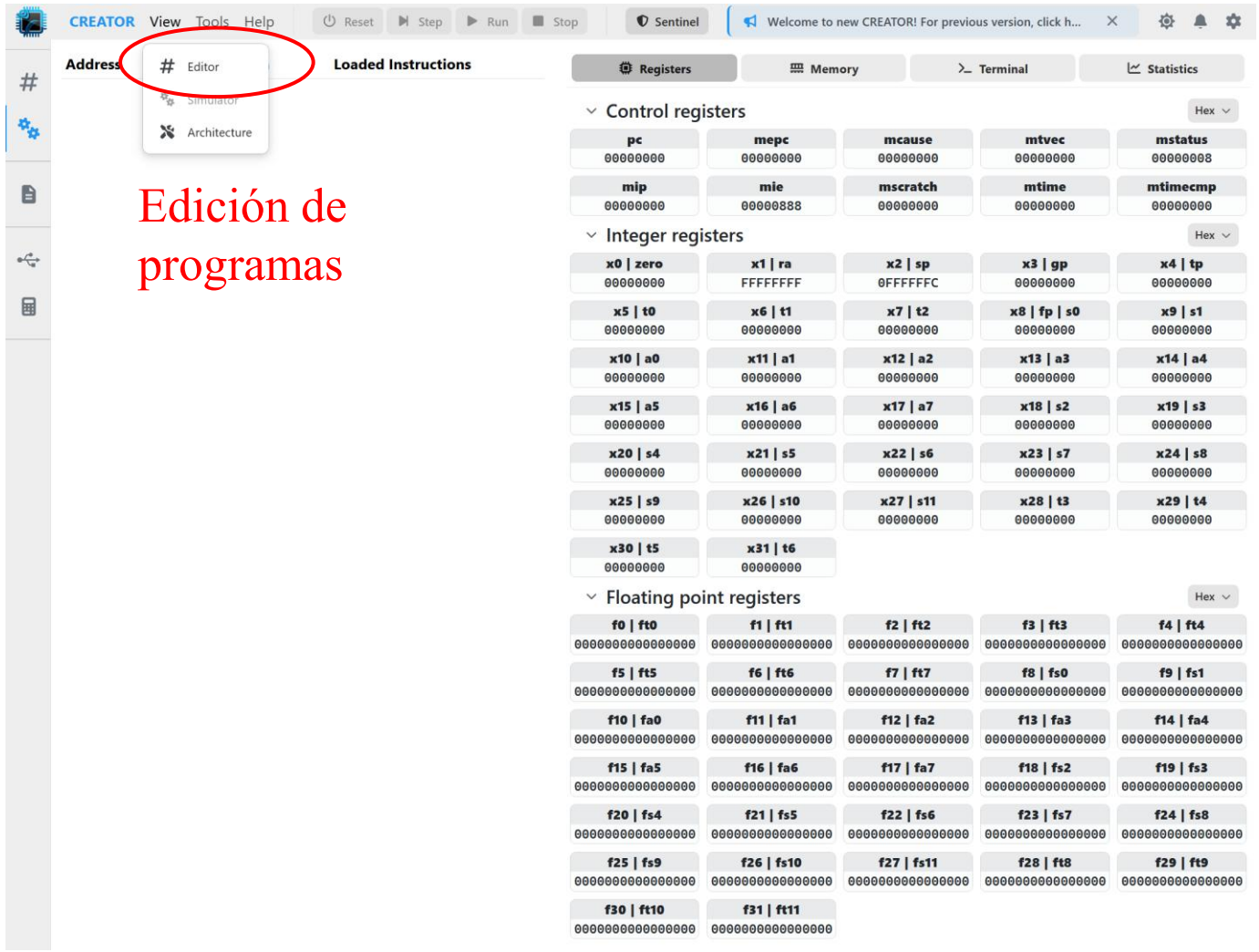


Inspección y edición de la arquitectura



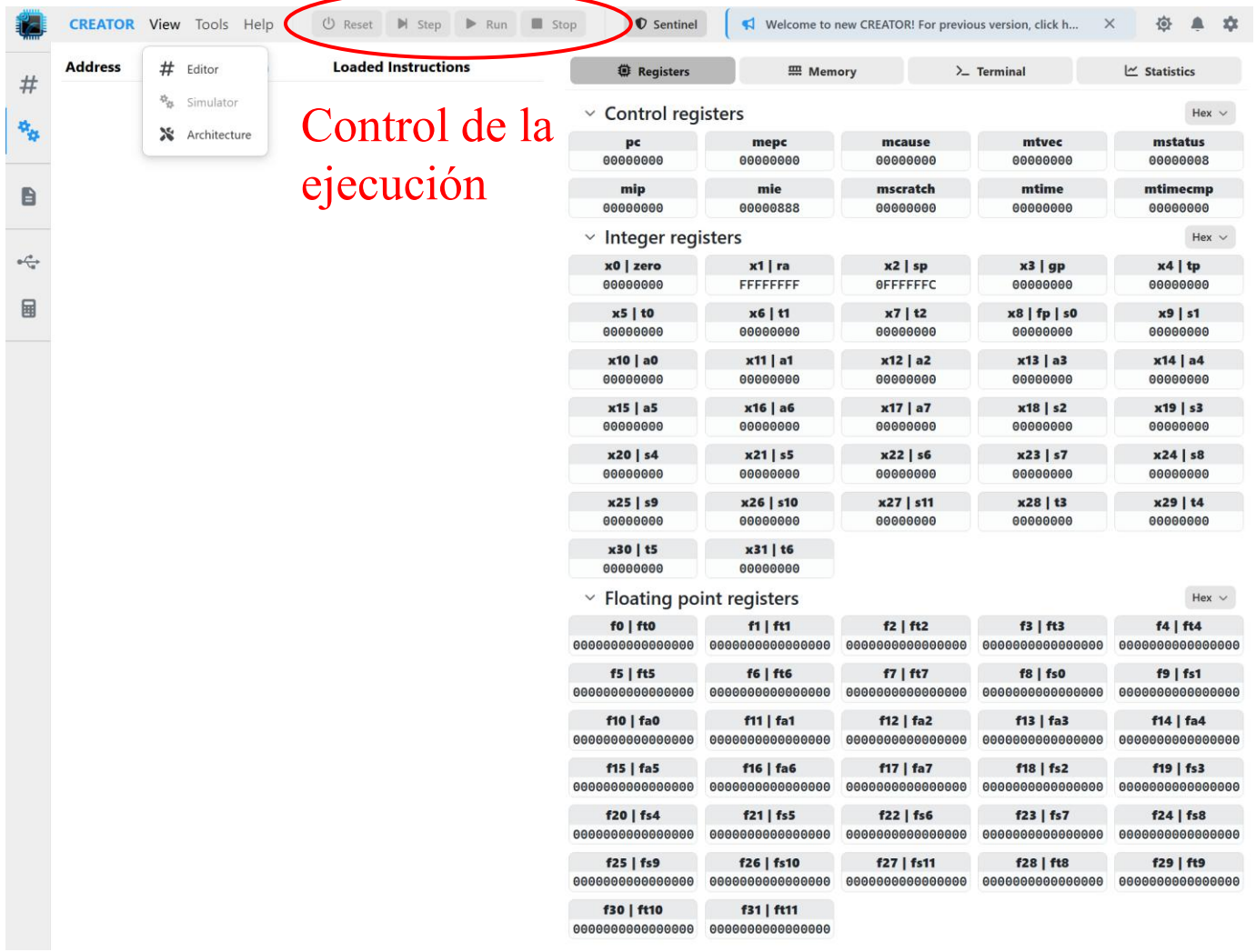
Ver/Editar arquitectura

Edición de programas



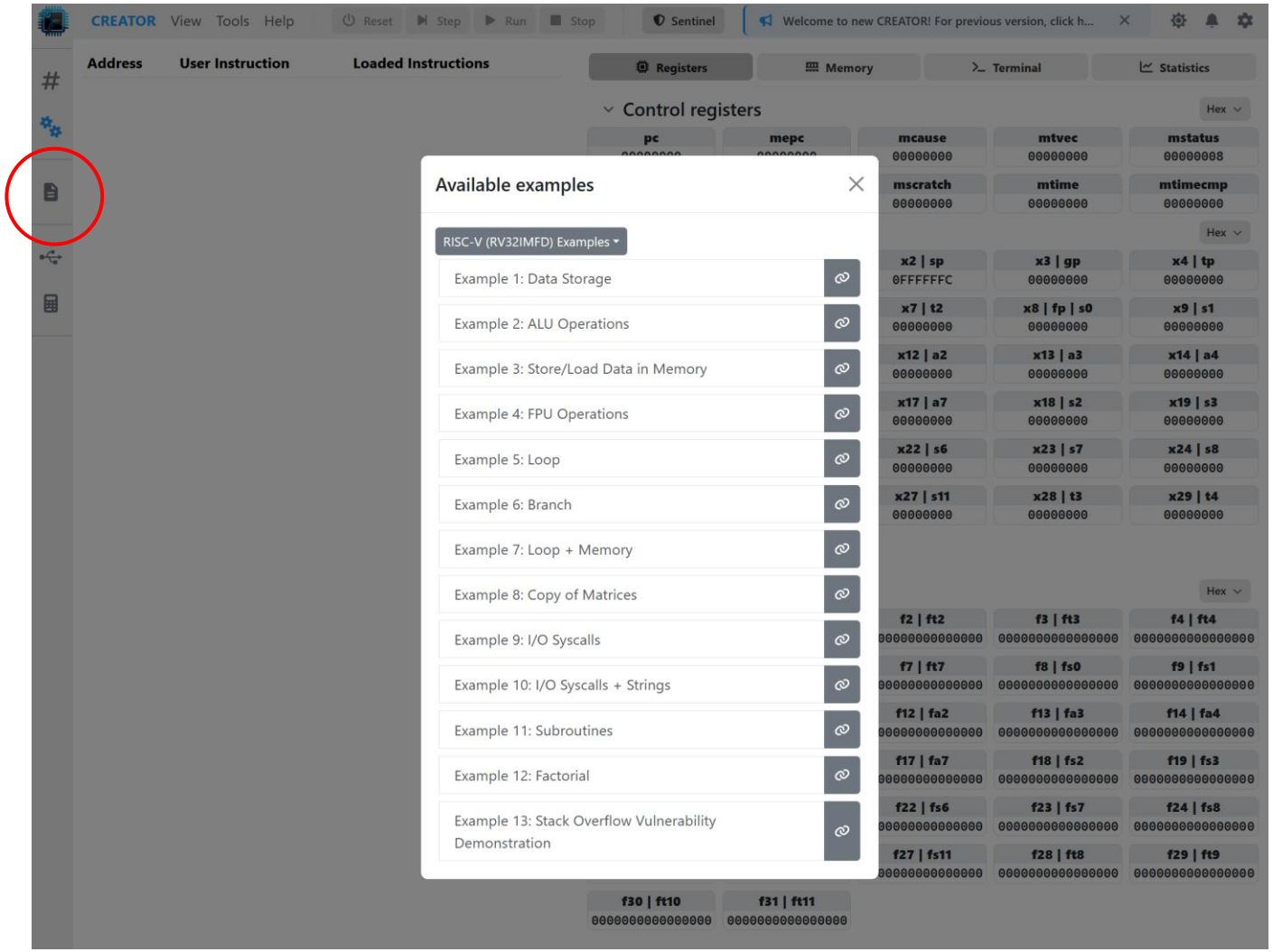
Edición de programas

Control de la ejecución



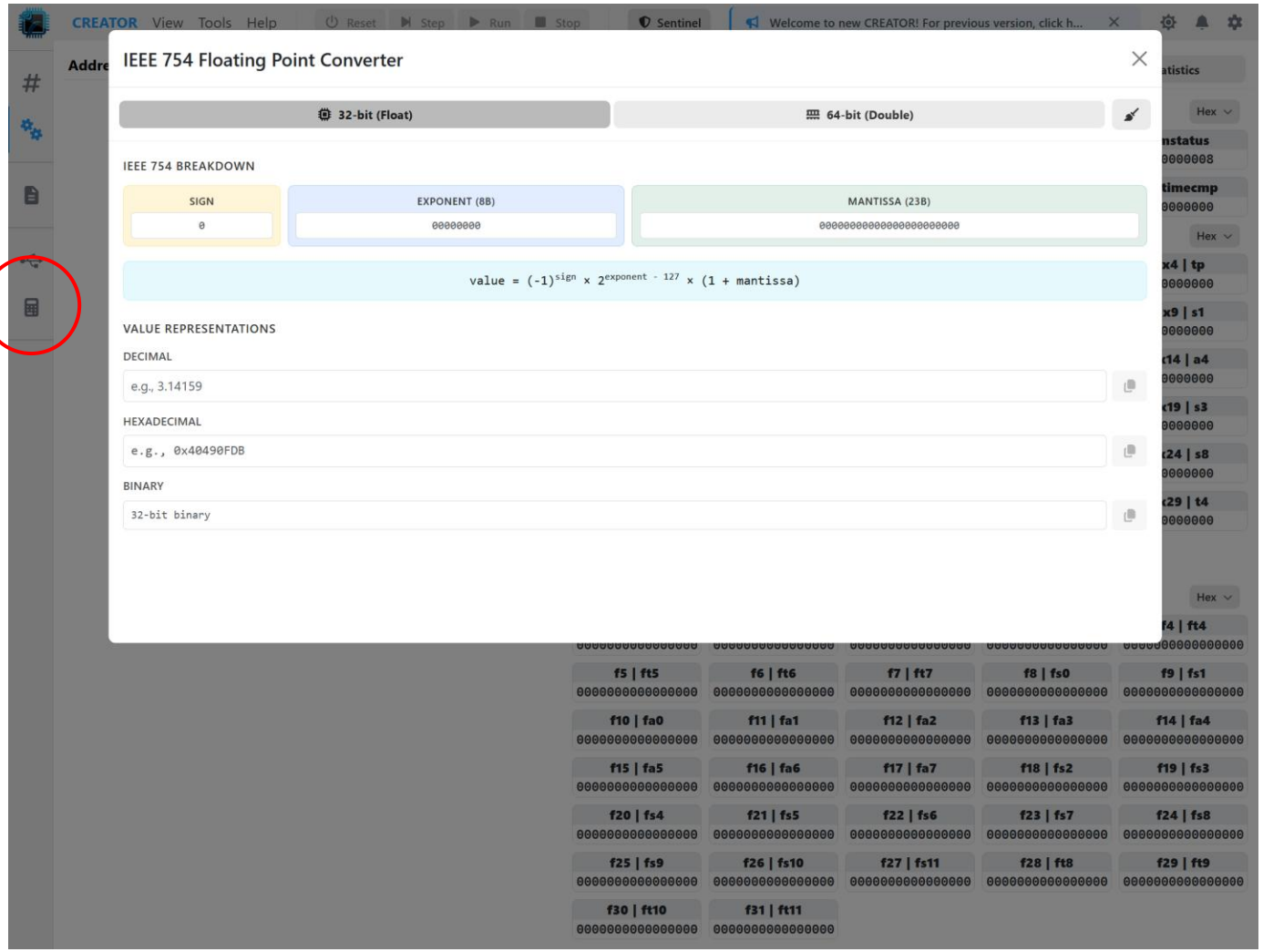
Ejemplos de programas en ensamblador

Ejemplos

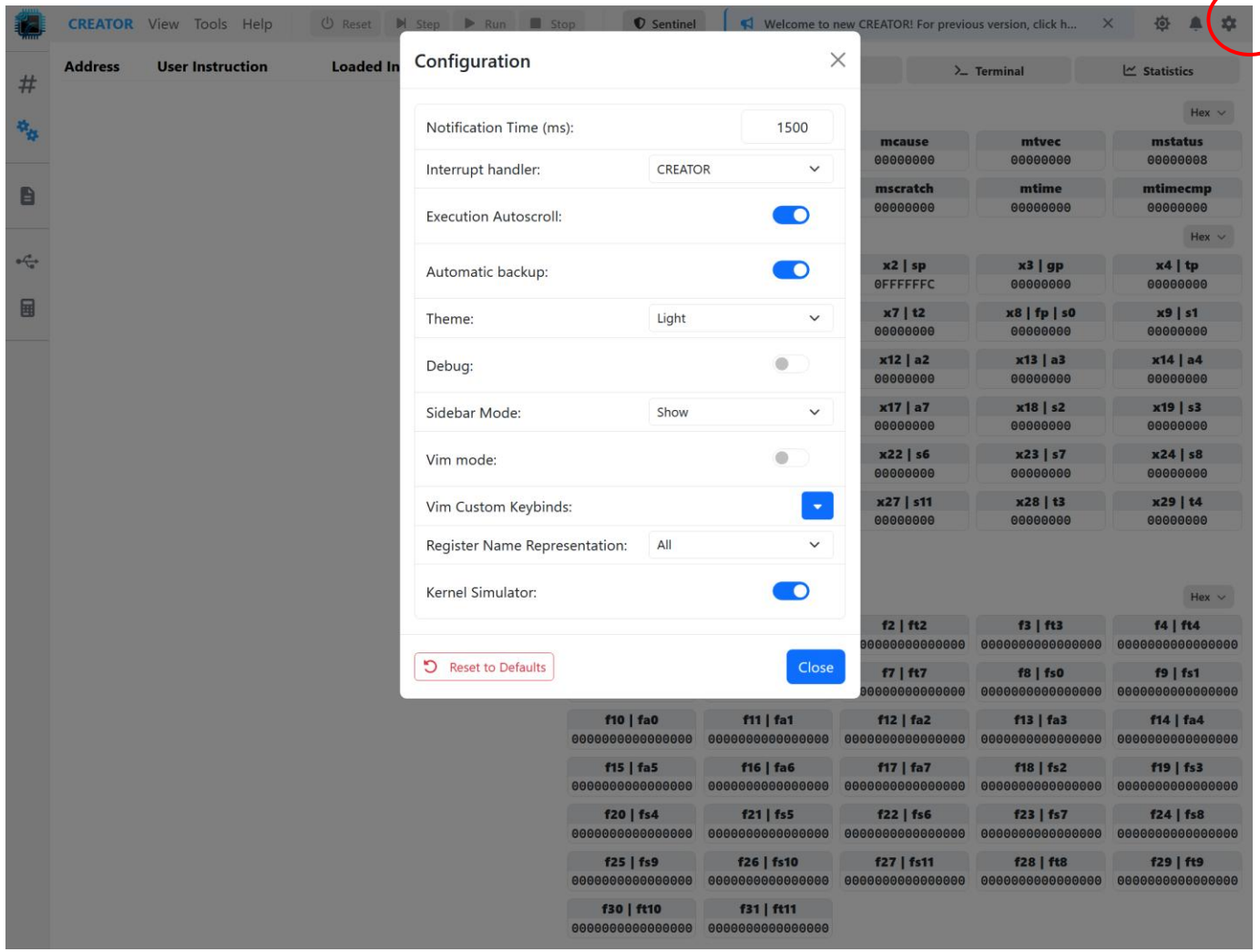


Calculadora de números en coma flotante

Calculadora

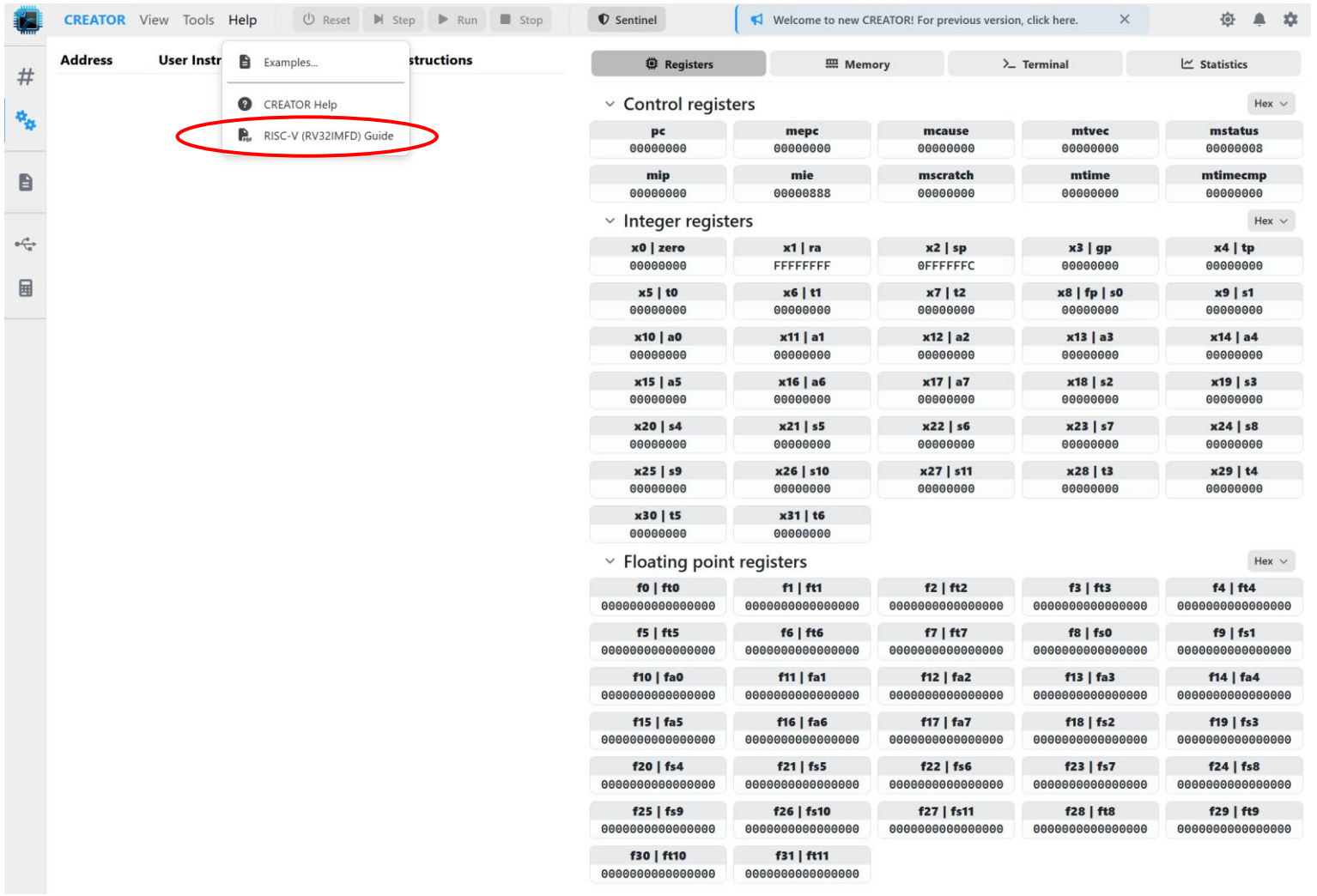


Configuración

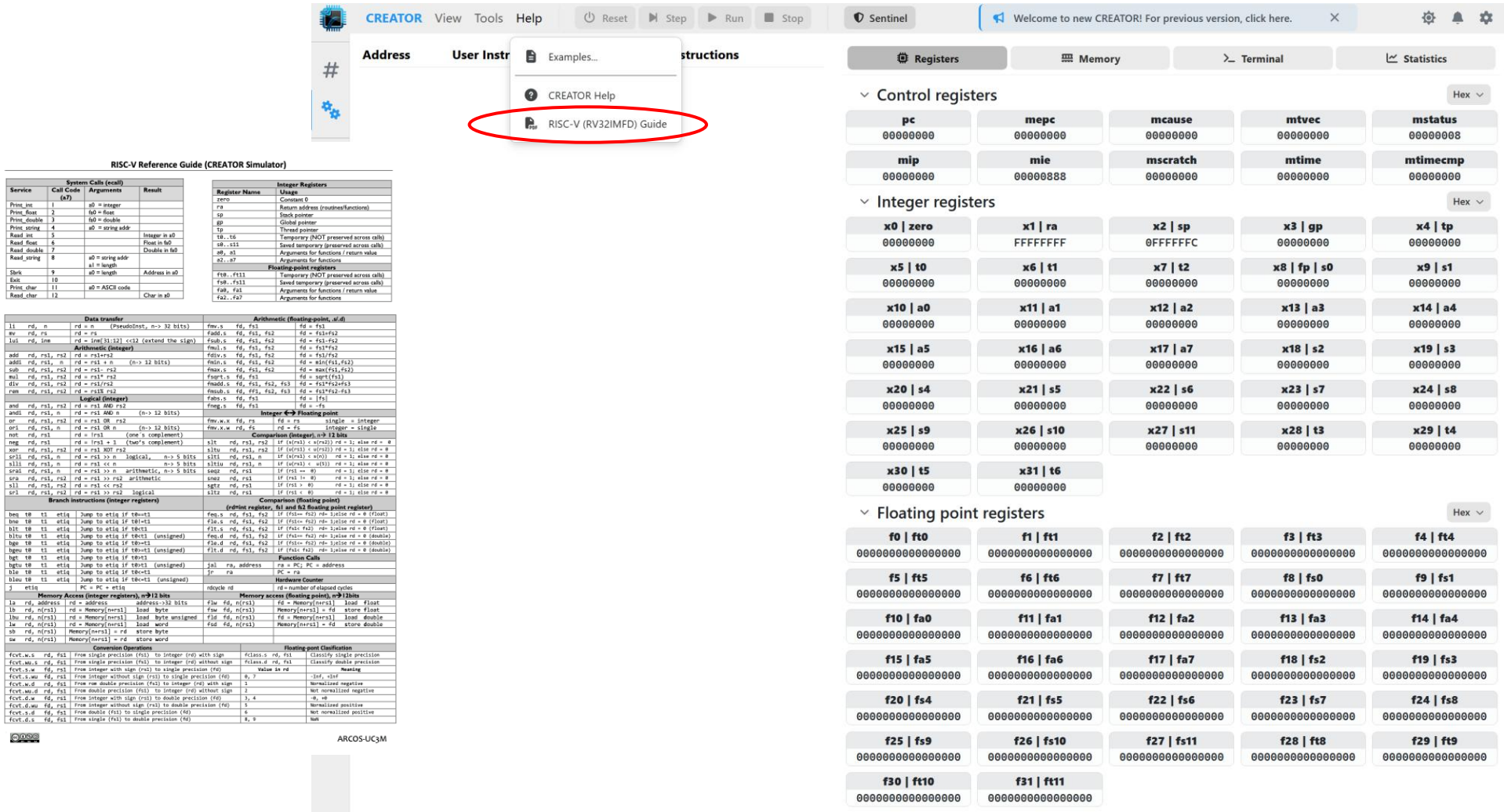


Configuración

Ayuda

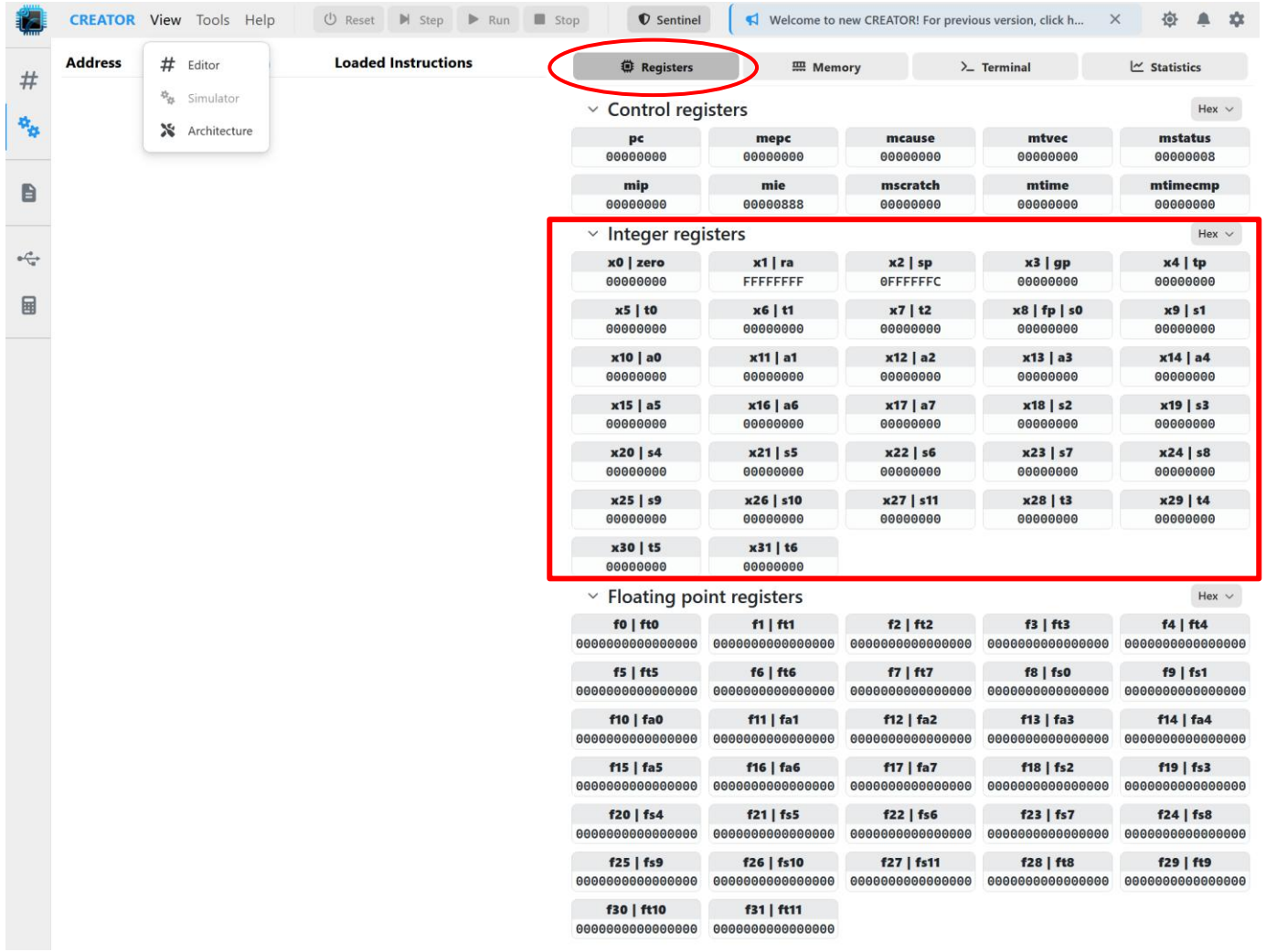

CREATOR
View
Tools
Help
Reset
Step
Run
Stop
Sentinel
Welcome to new CREATOR! For previous version, click here.
Address
User Instr
Examples...
CREATOR Help
RISC-V (RV32IMFD) Guide
structions
Registers
Memory
Terminal
Statistics
Control registers
Hex
pc 00000000
mepc 00000000
mcause 00000000
mtvec 00000000
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
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

Ayuda

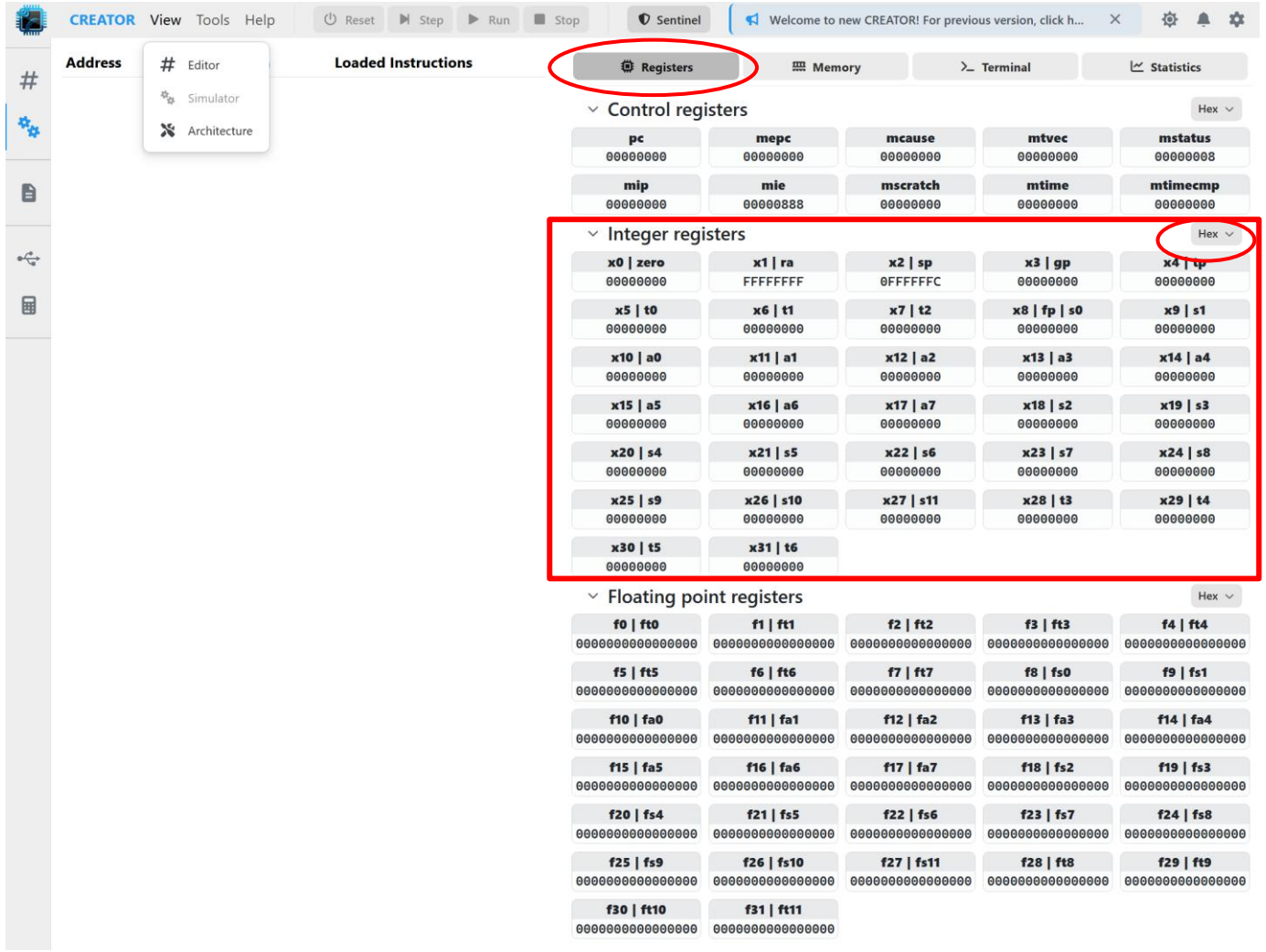

CREATOR View Tools Help
Reset Step Run Stop
Sentinel
Welcome to new CREATOR! For previous version, click here.
Address
User Instr
Examples...
CREATOR Help
RISC-V (RV32IMFD) Guide
Registers Memory Terminal Statistics
Control registers
Hex
pc 00000000
mepc 00000000
mcause 00000000
mtvec 00000000
mstatus 00000008
mip 00000000
mie 00000888
mscratch 00000000
mtime 00000000
mtimecmp 00000000
Integer registers
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

RISC-V Reference Guide (CREATOR Simulator)
System Calls (ecall)
Integer Registers
Data transfer
Arithmetic (integer)
Logical (integer)
Branch instructions (integer registers)
Arithmetic (floating-point, s/.d)
Integer ↔ Floating point
Comparison (integer), n↔ 12 bits
Comparison (floating point)
Function Calls
Hardware Counter
Memory Access (integer registers), n↔12 bits
Memory access (floating point), n↔12bits
Conversion Operations
Floating-point Clasification
ARCOS-UC3M

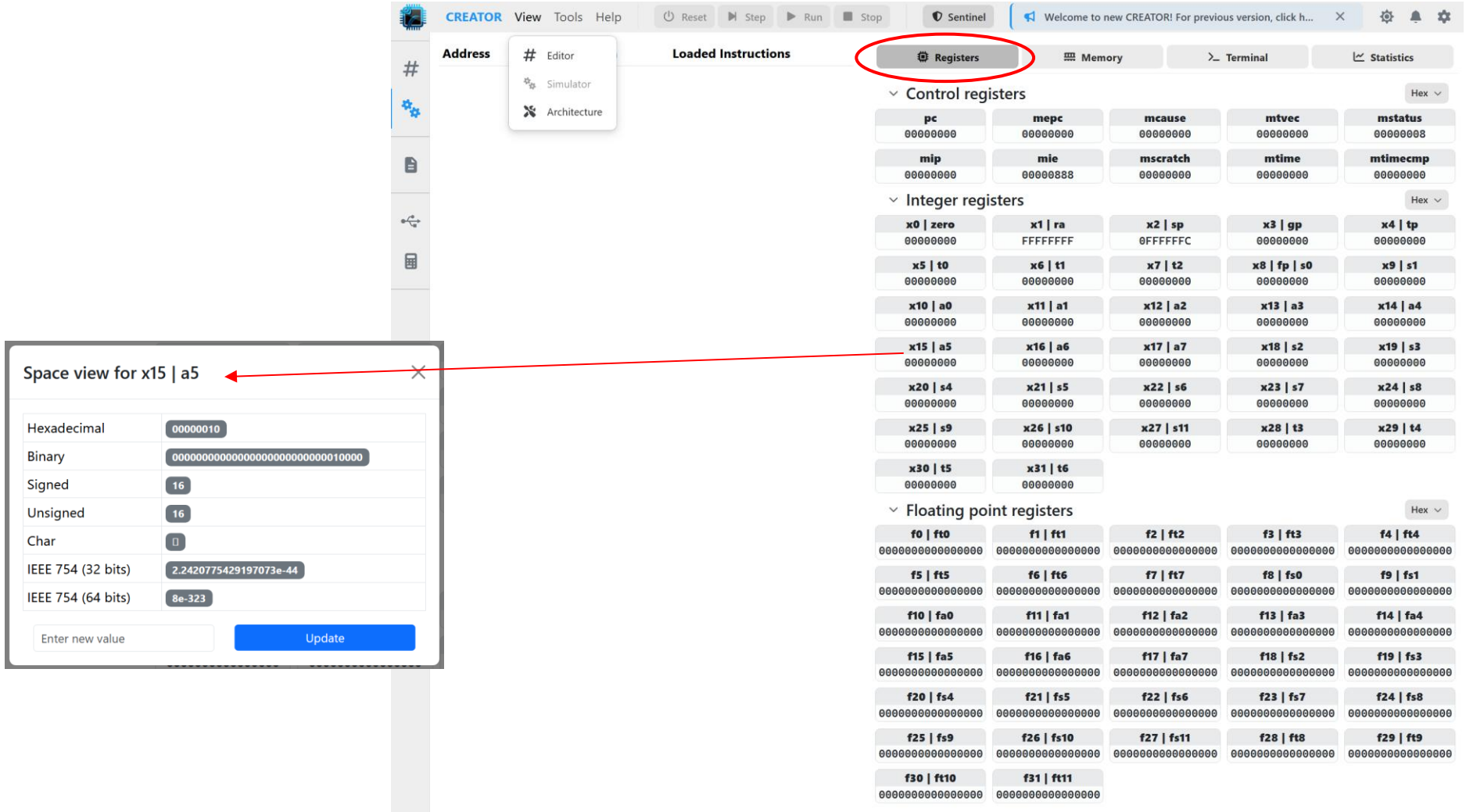
Registros enteros



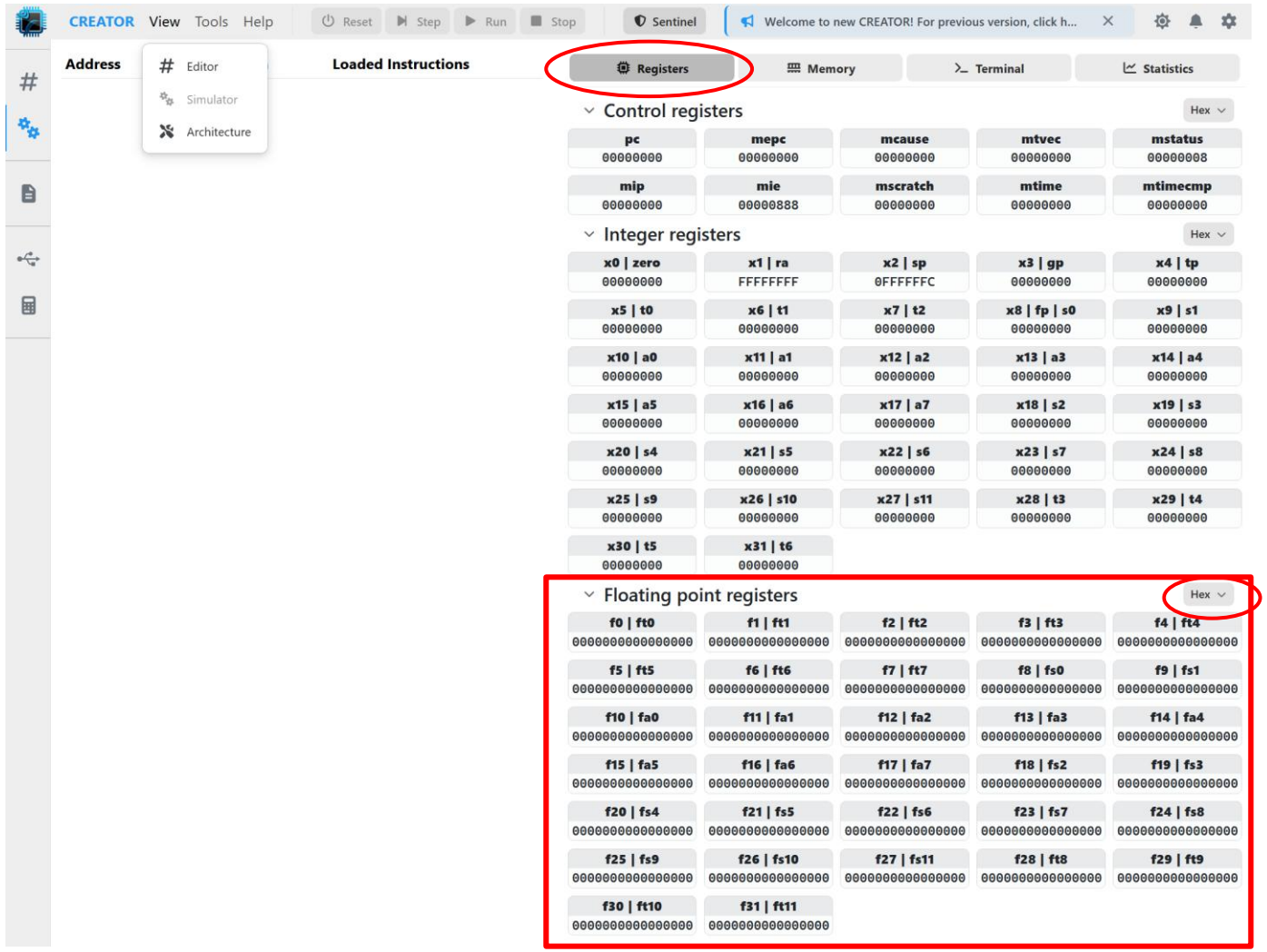
Registros enteros



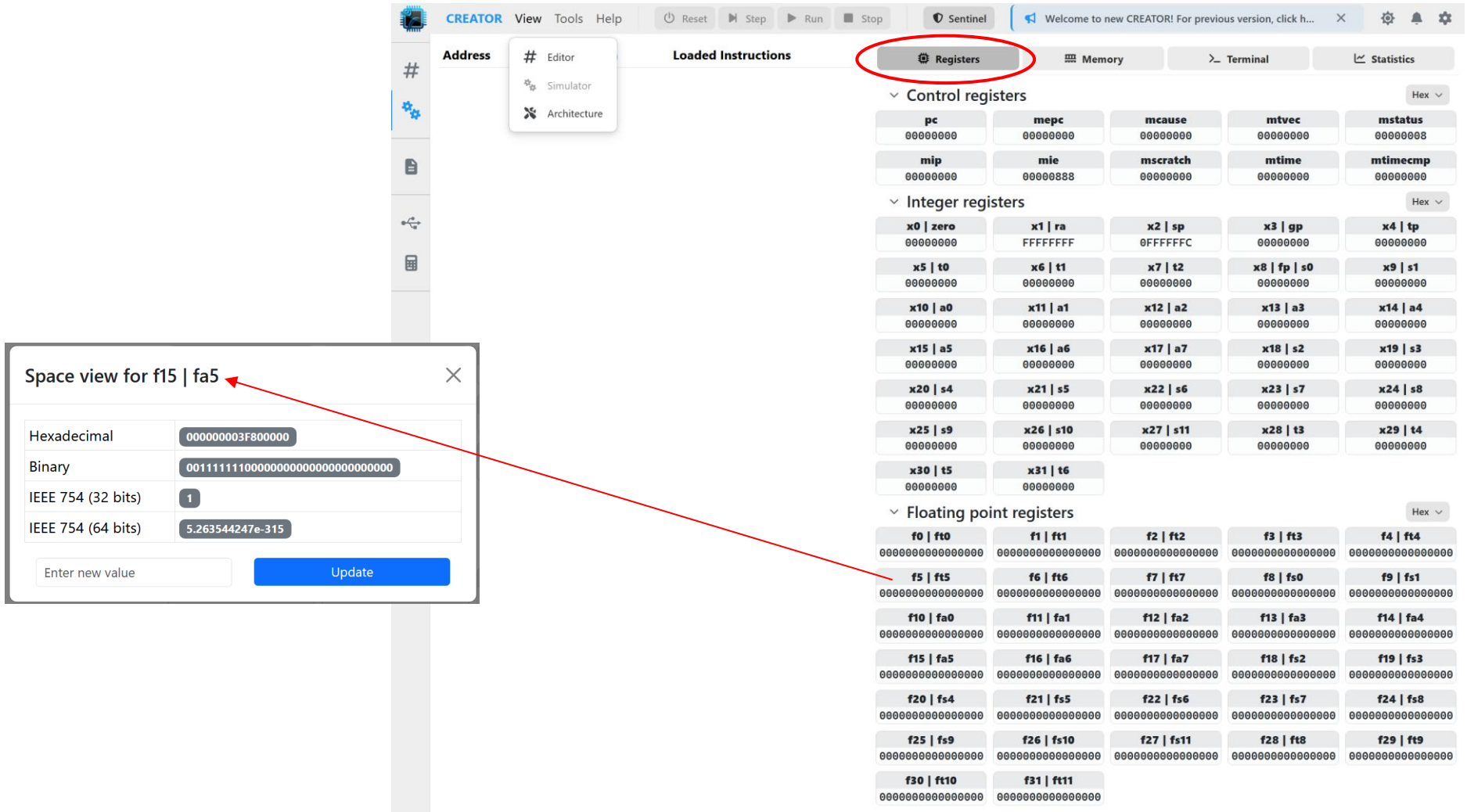
Registros enteros



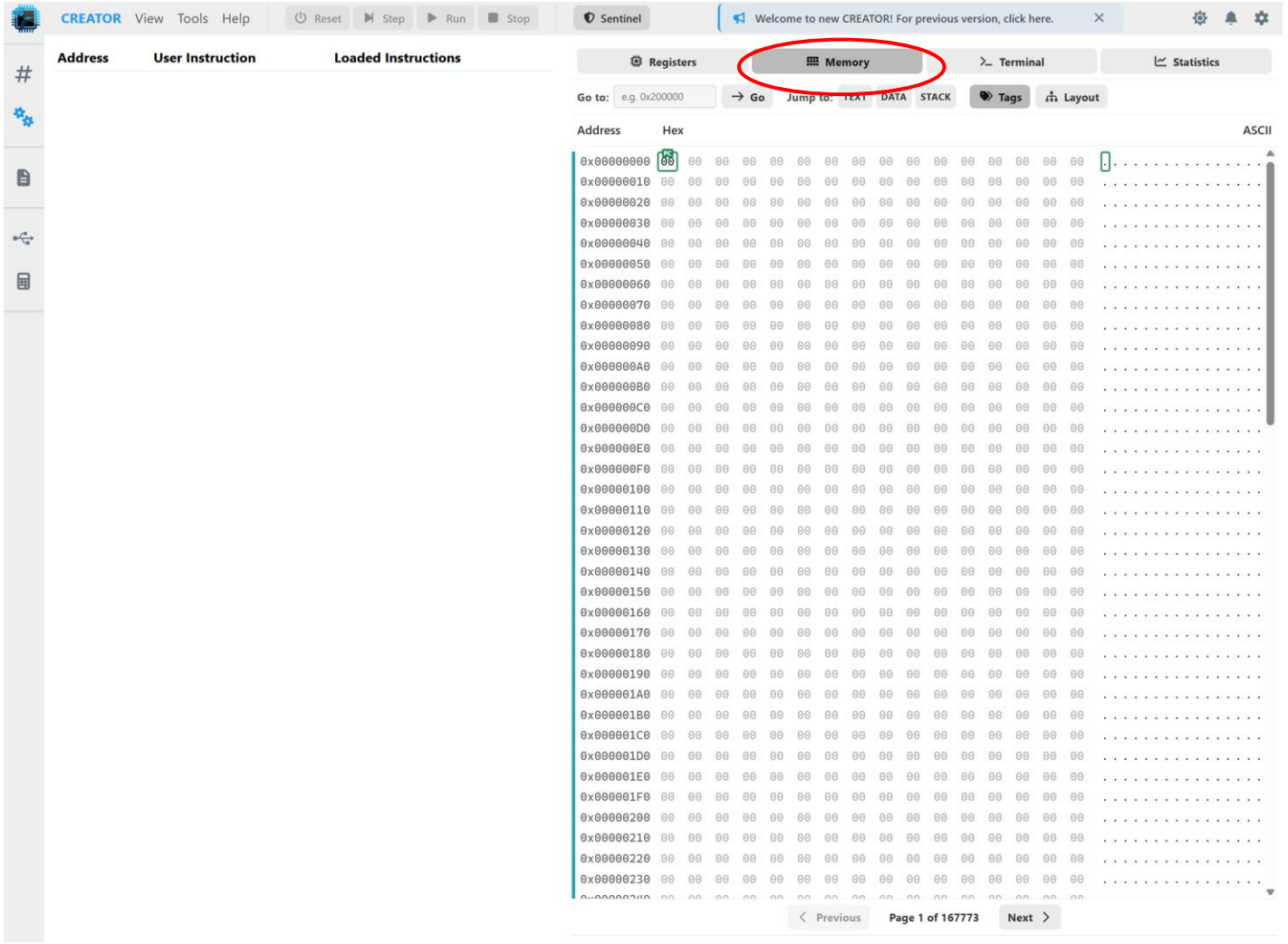
Registros en coma flotante



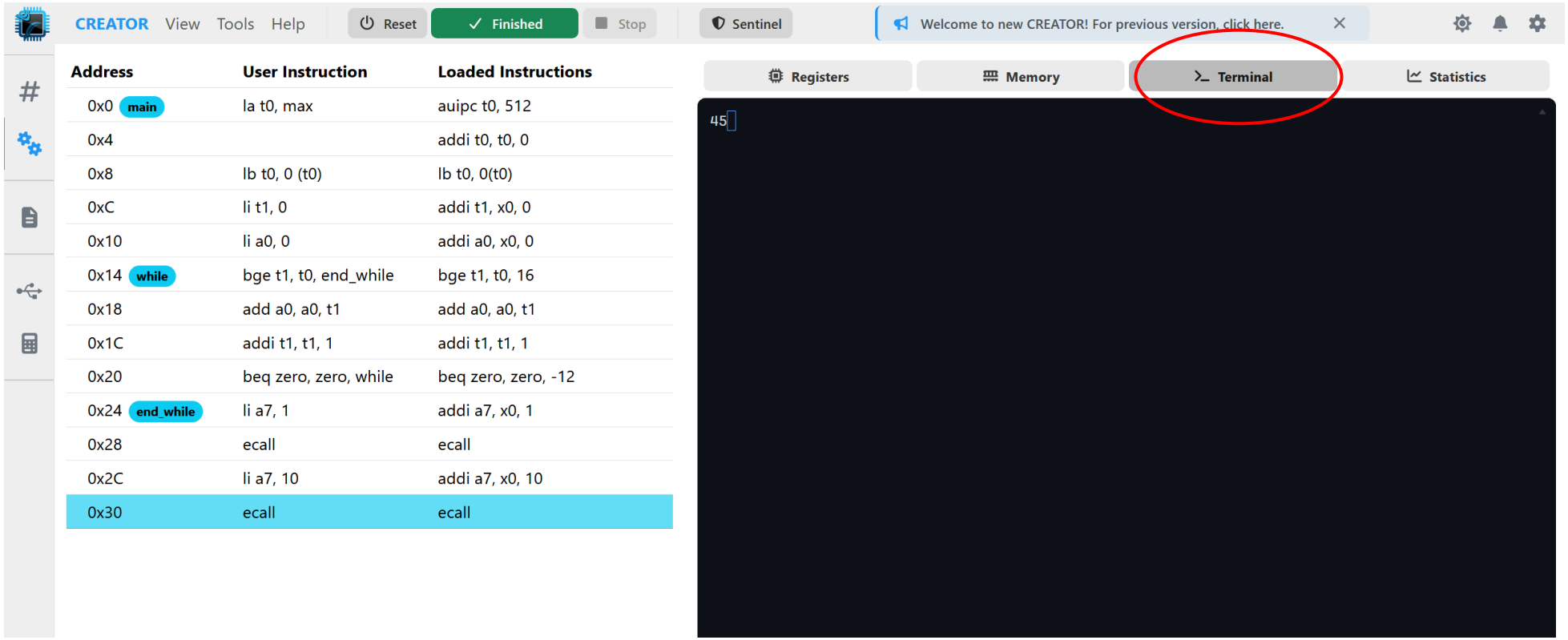
Registros en coma flotante



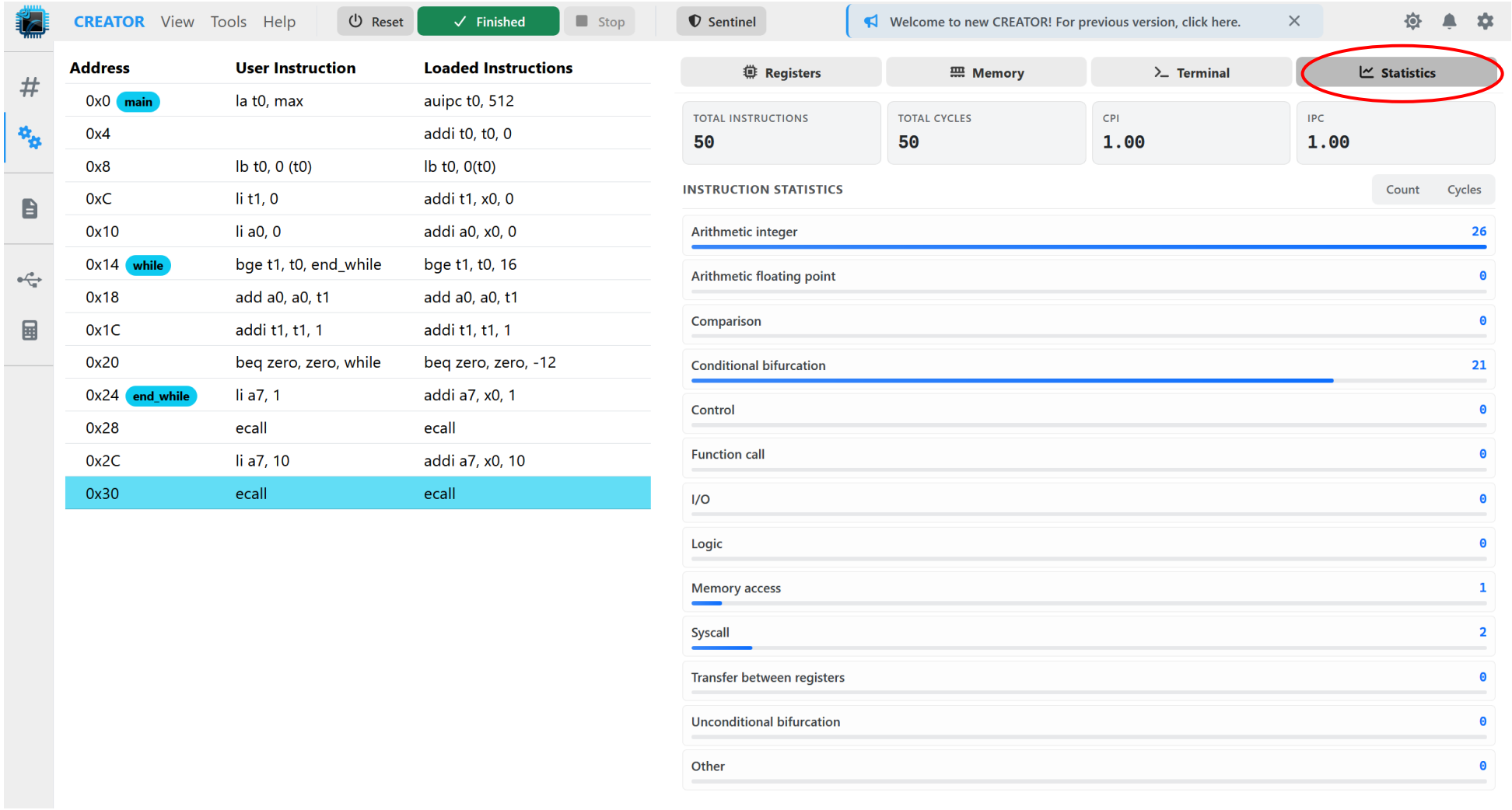
Contenido de la memoria



Pantalla


CREATOR View Tools Help
Reset
Finished
Stop
Sentinel
Welcome to new CREATOR! For previous version, click here.
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
Registers
Memory
Terminal
Statistics
45

Estadísticas de ejecución

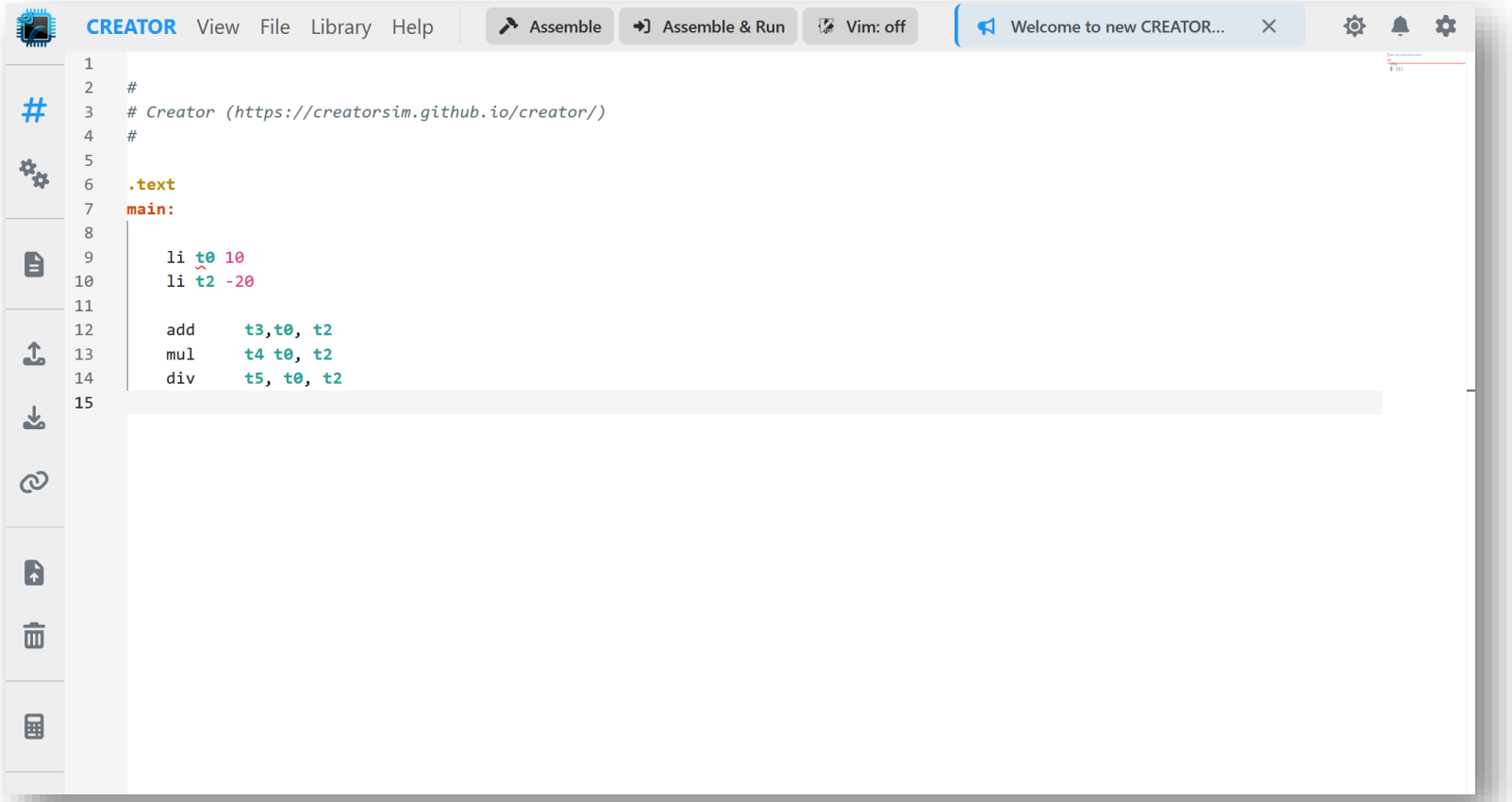


Bloque 1

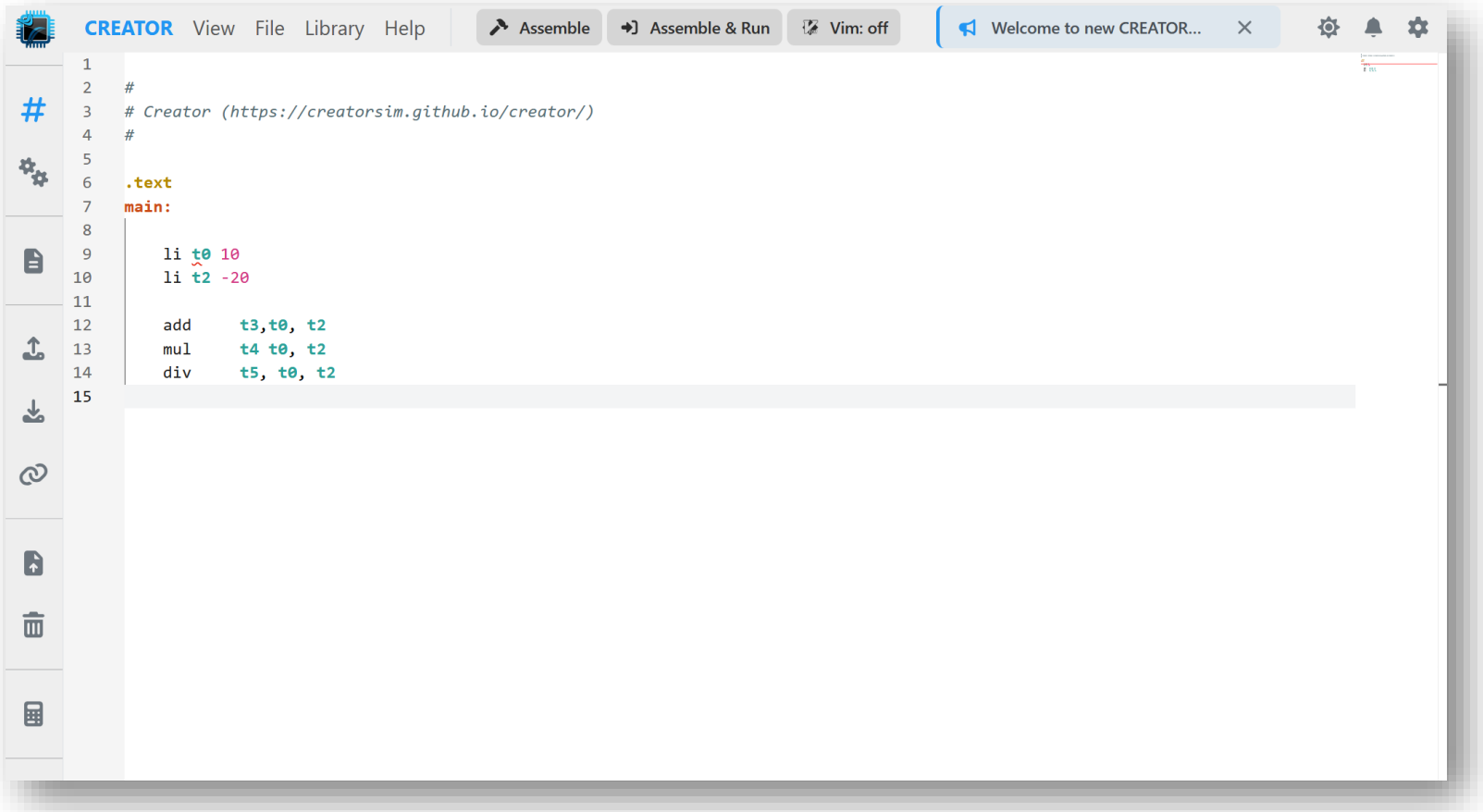
- ¿Qué es CREATOR?
  - El origen de CREATOR
  - Características de CREATOR
  - ¿Dónde se usa CREATOR?
  - Extensiones futuras
- **¿Cómo utilizar CREATOR? Visión del Estudiante**
  - **Edición y compilación de programas**
  - **Ejecución y depuración de programas**
  - **Convenio de paso de parámetros 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

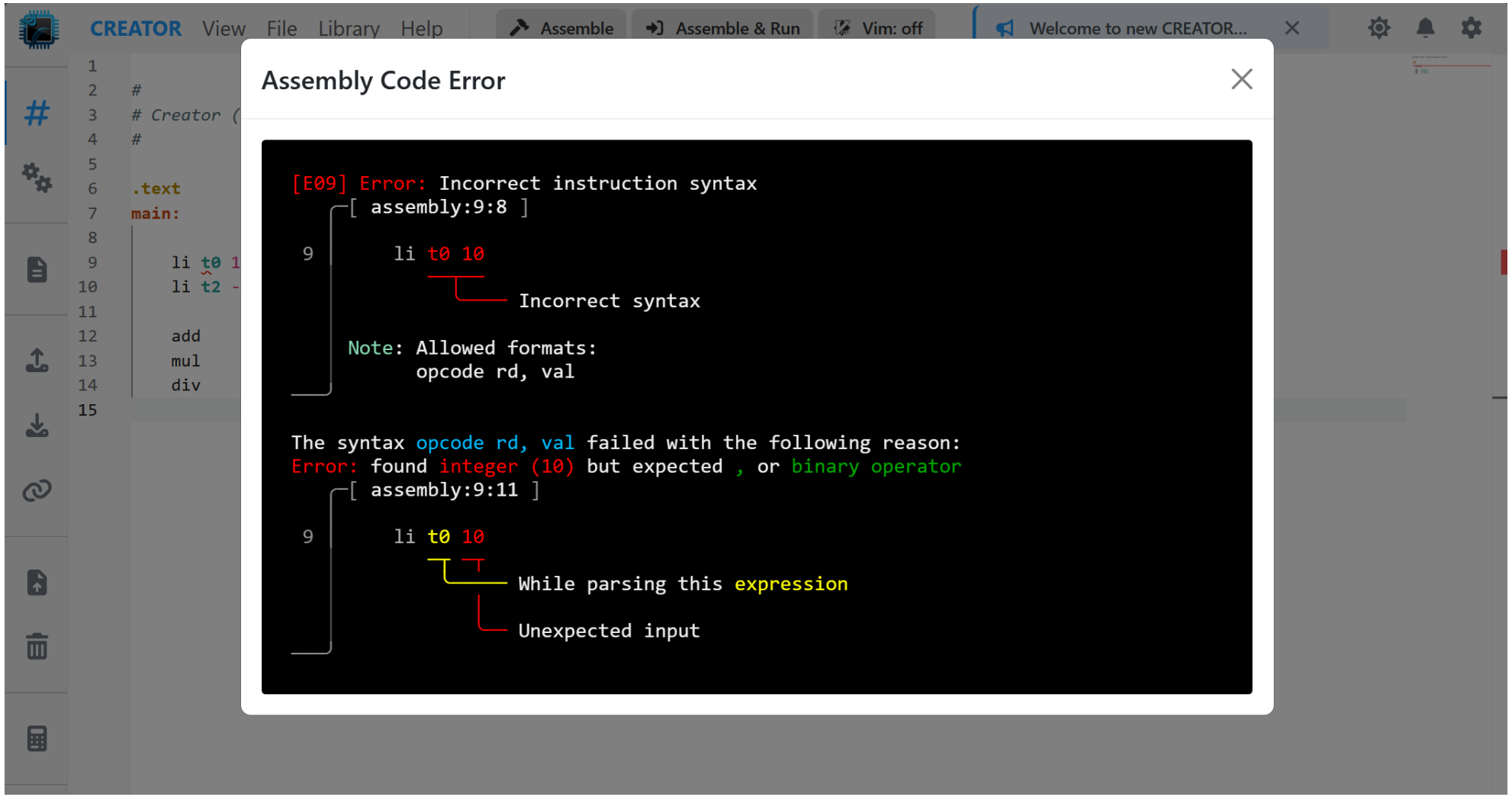
ejemplo



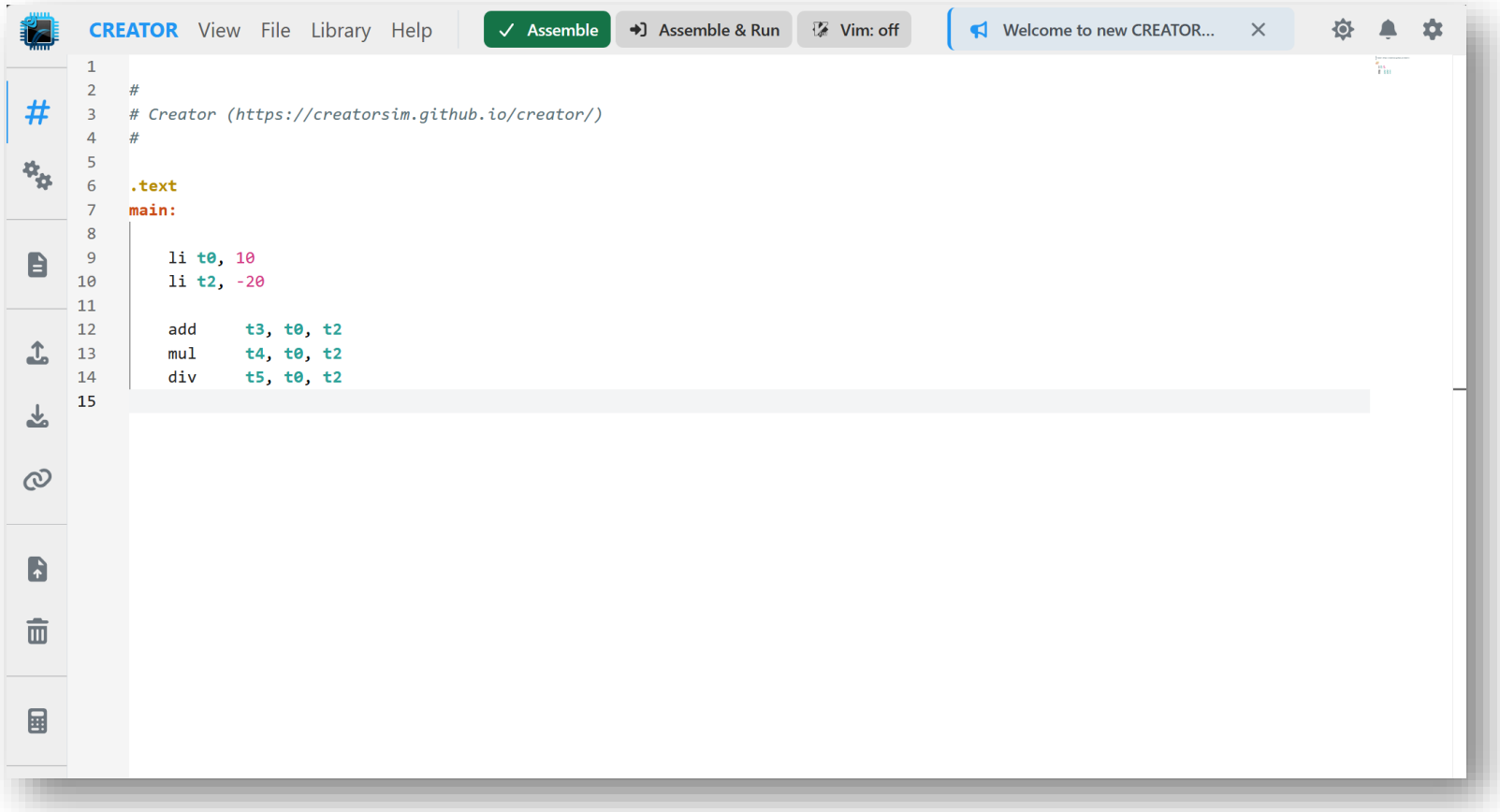
Compilación



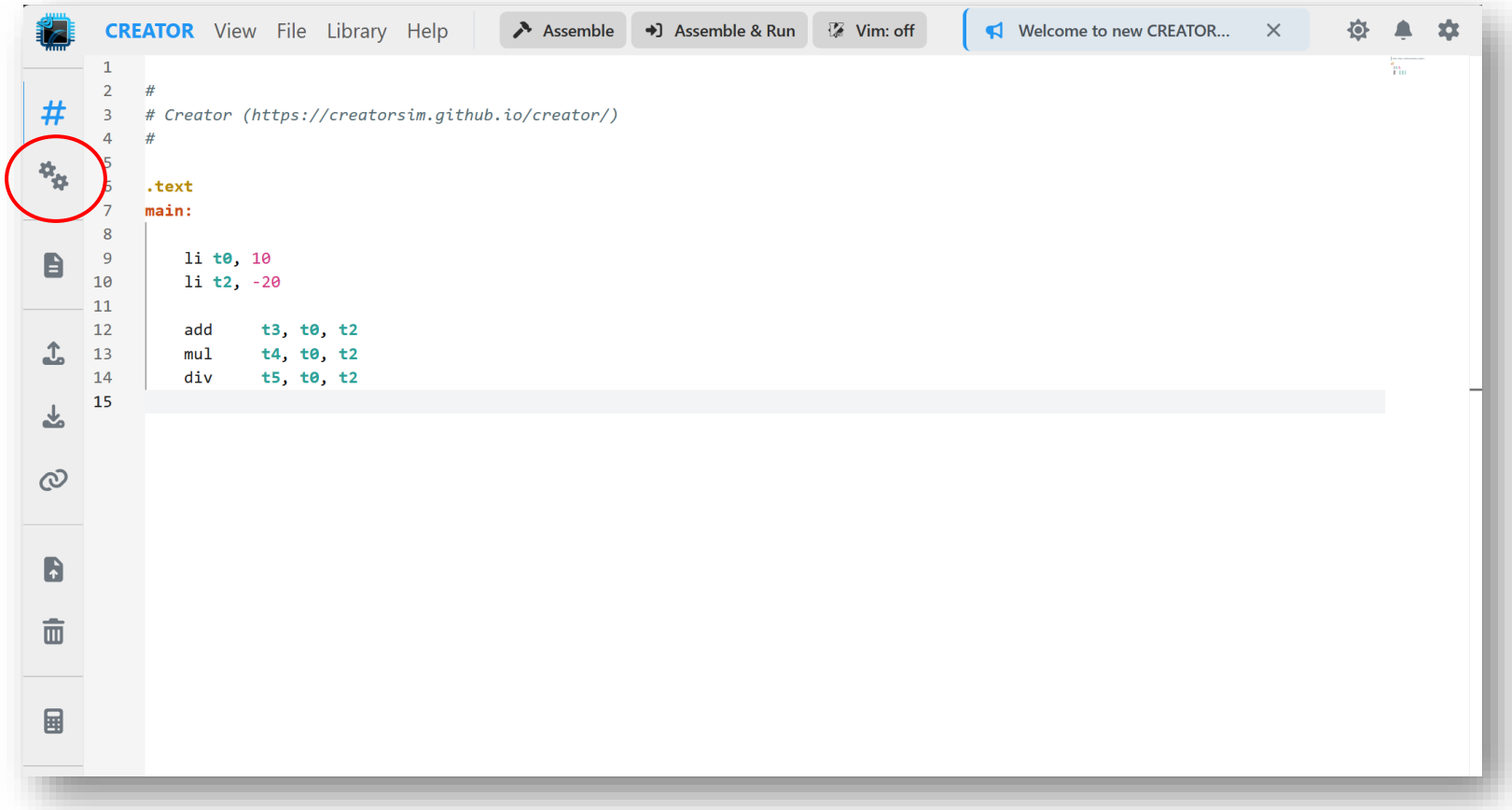
Error de compilación



Compilación sin error

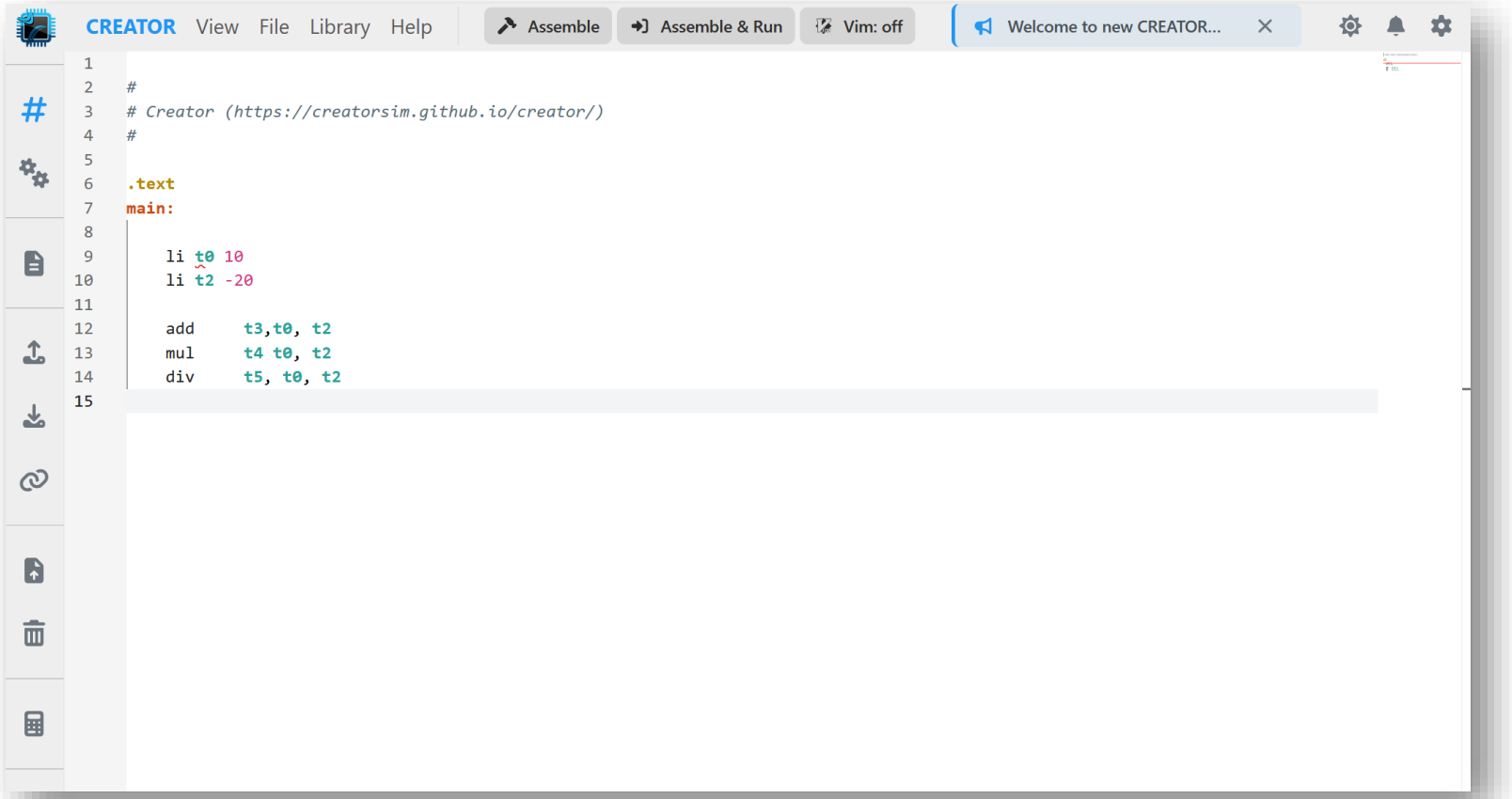


Paso al simulador



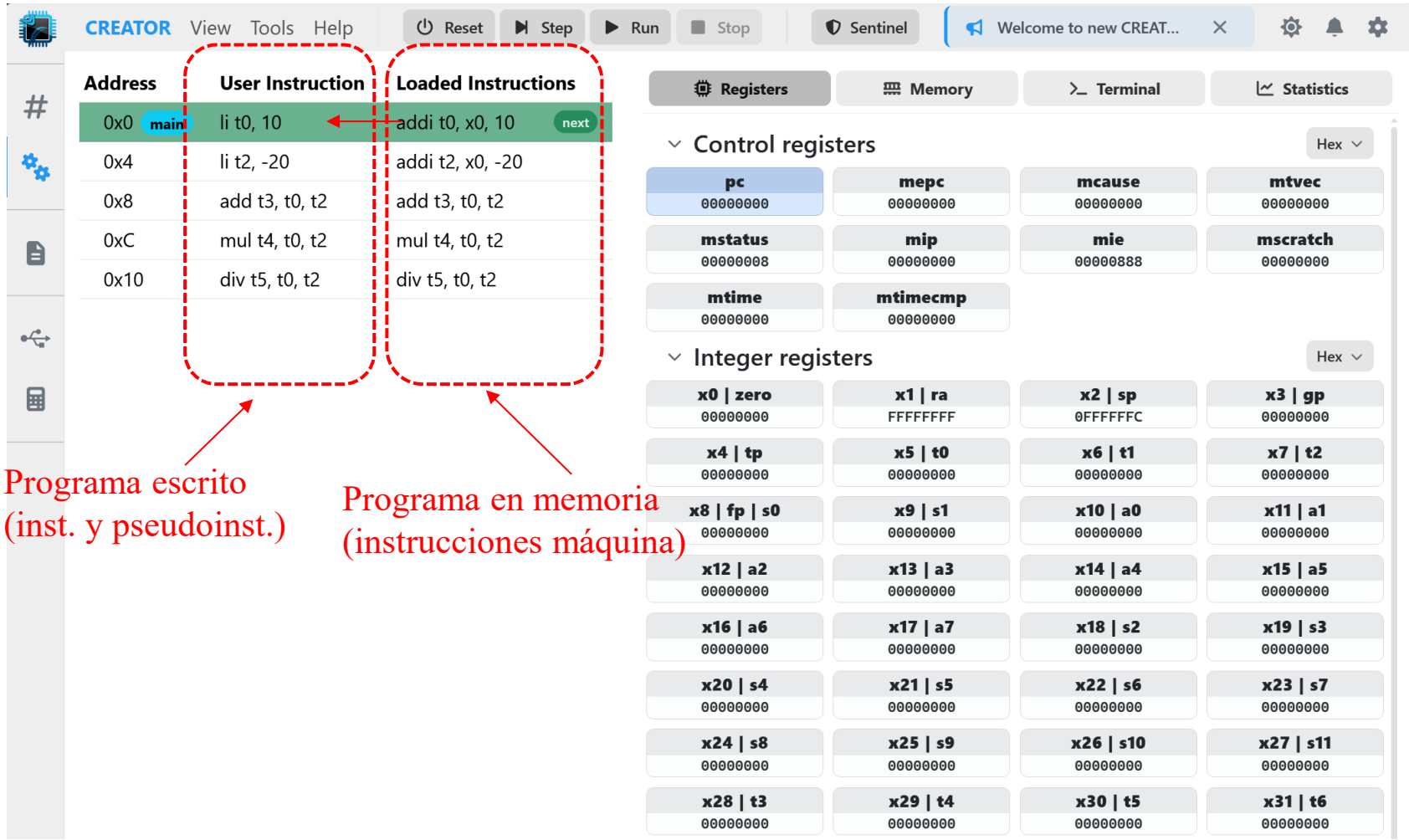
Modificar el programa para imprimir el valor de t5

ejemplo

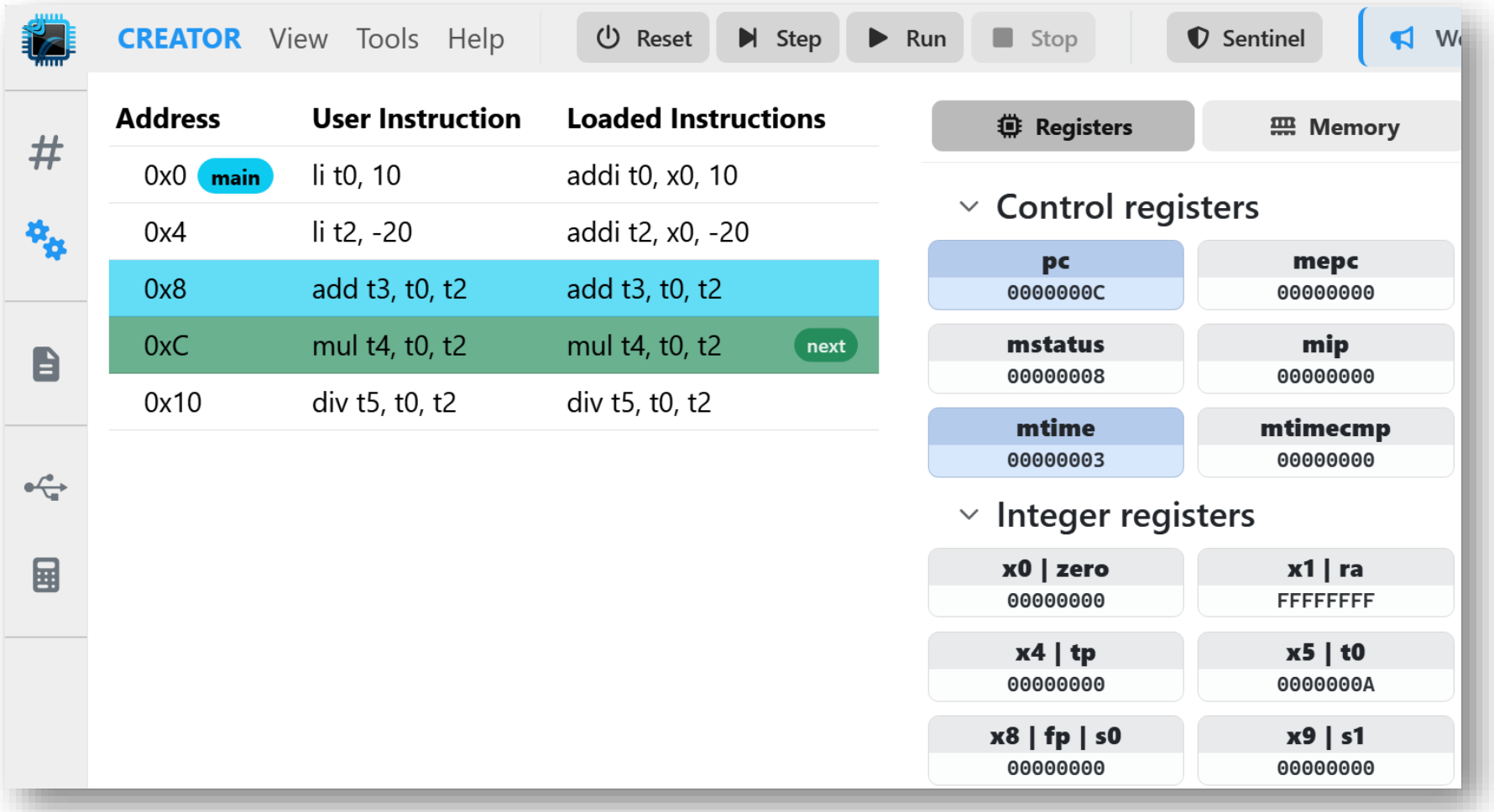


Simulador

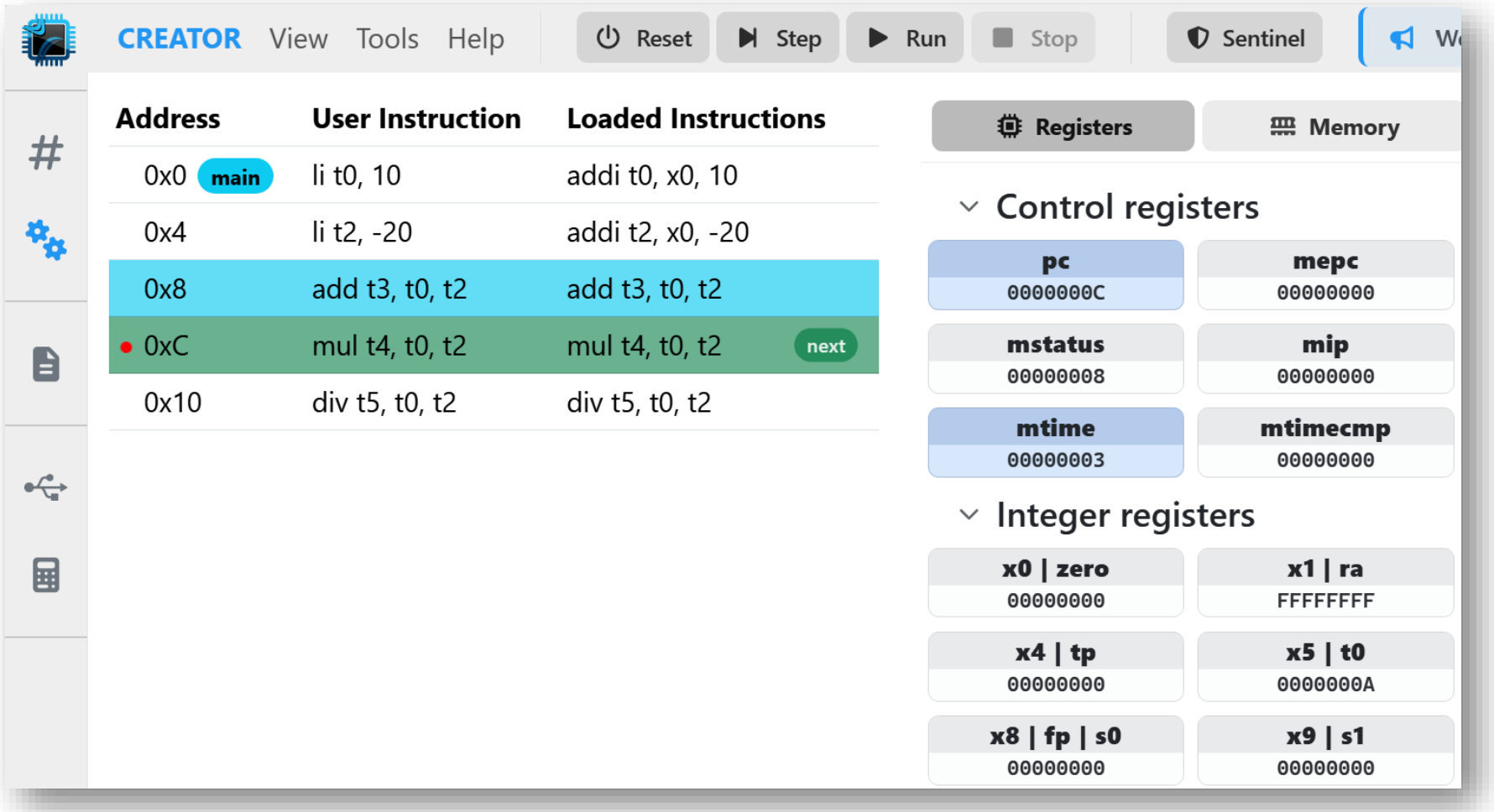
ejemplo



Flujo de ejecución



Puntos de ruptura



Segmento de datos

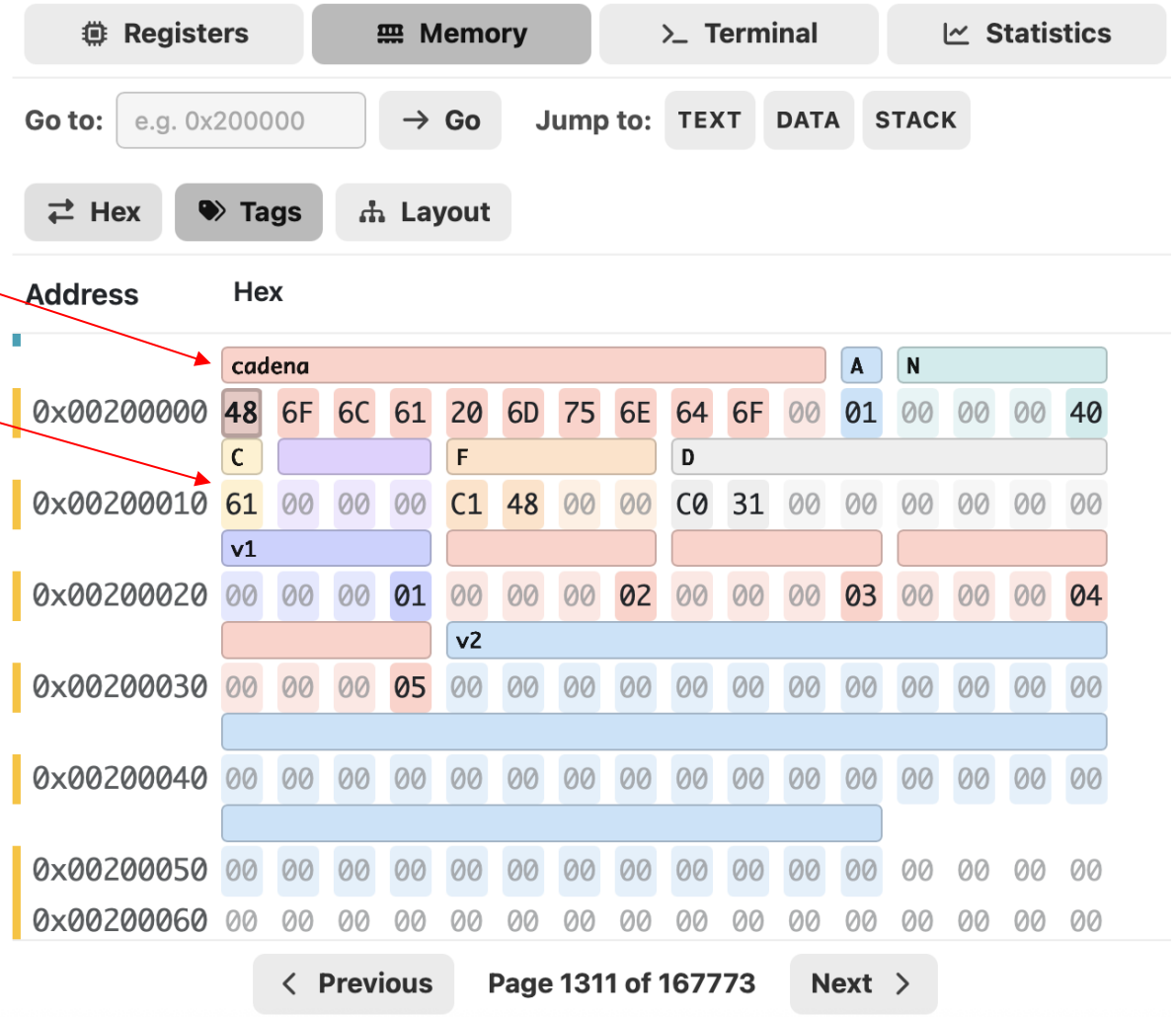
ejemplo

Visualización de datos en memoria

ejemplo

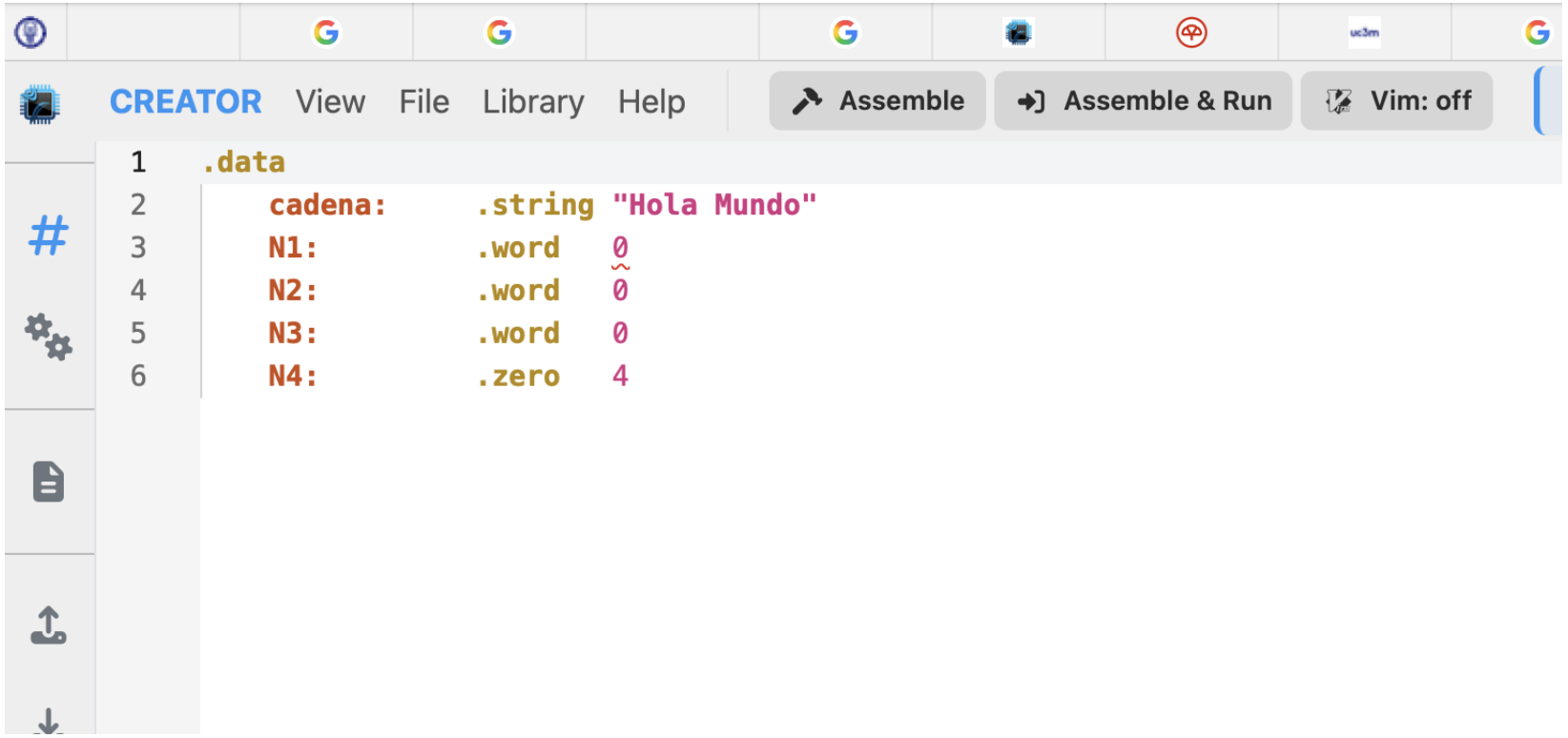
Assembly:

```
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
```



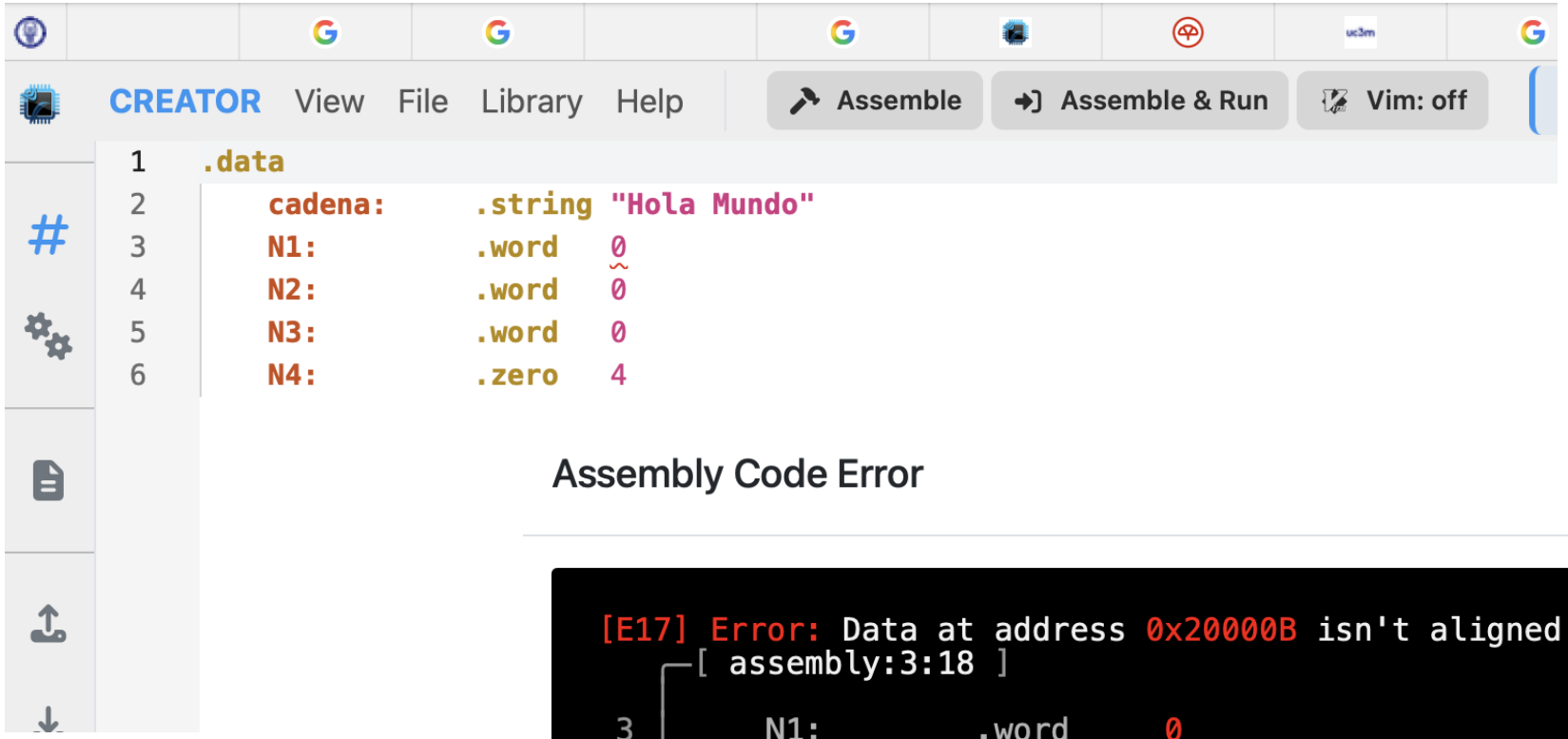
Detección de datos no alineados

ejemplo



Detección de datos no alineados

ejemplo

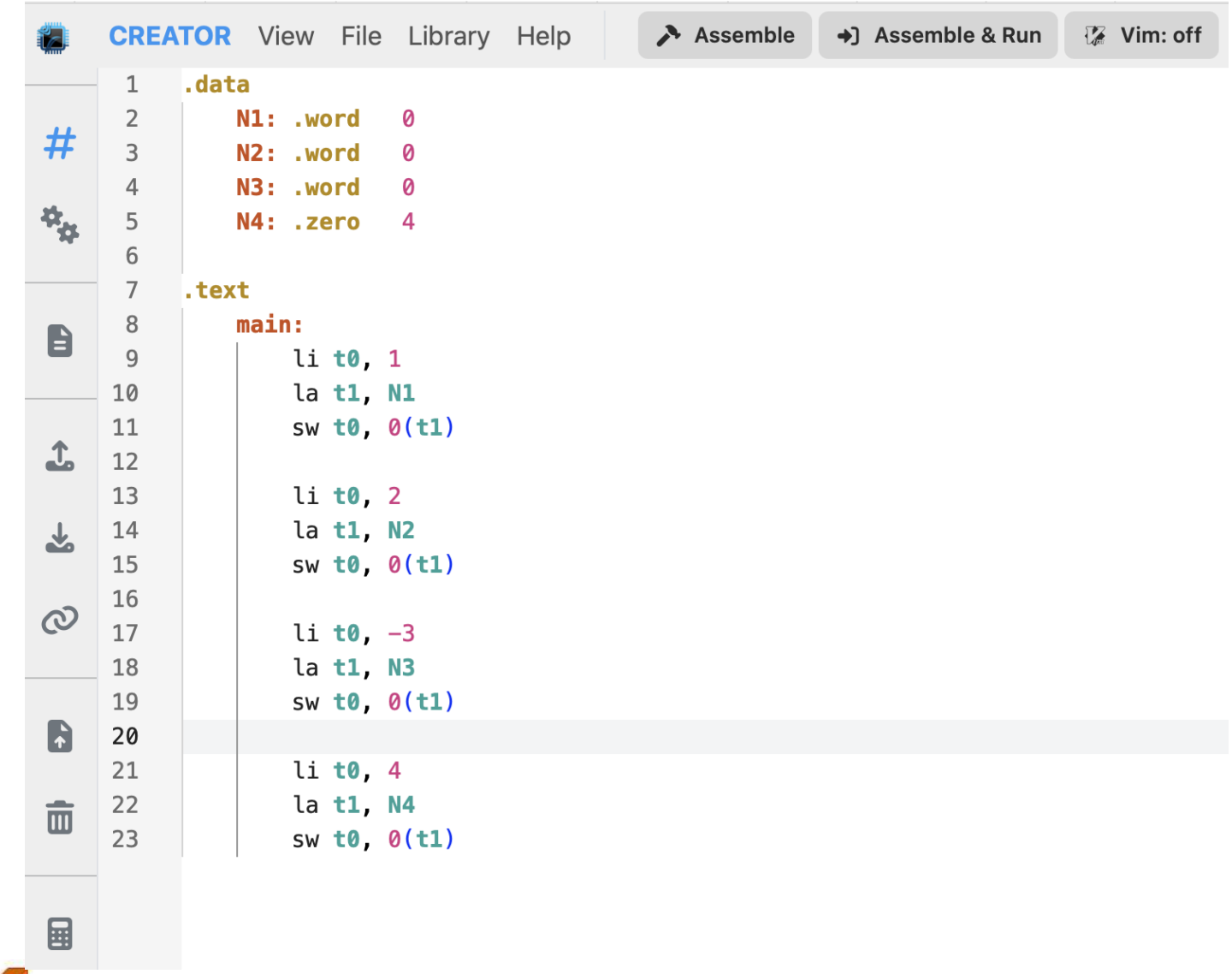


Segmento de datos corregido

ejemplo

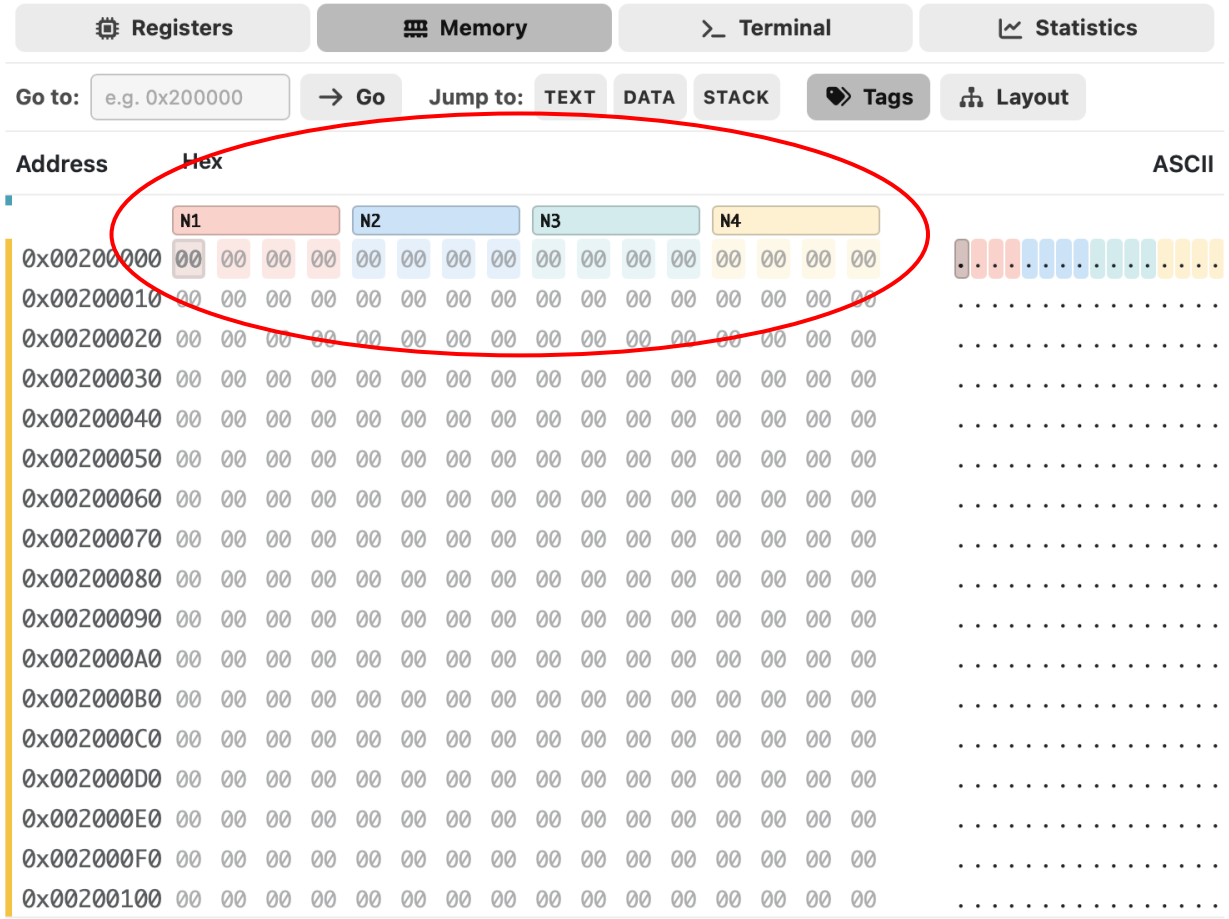
Ejemplo. Ejecutar el siguiente programa

ejemplo



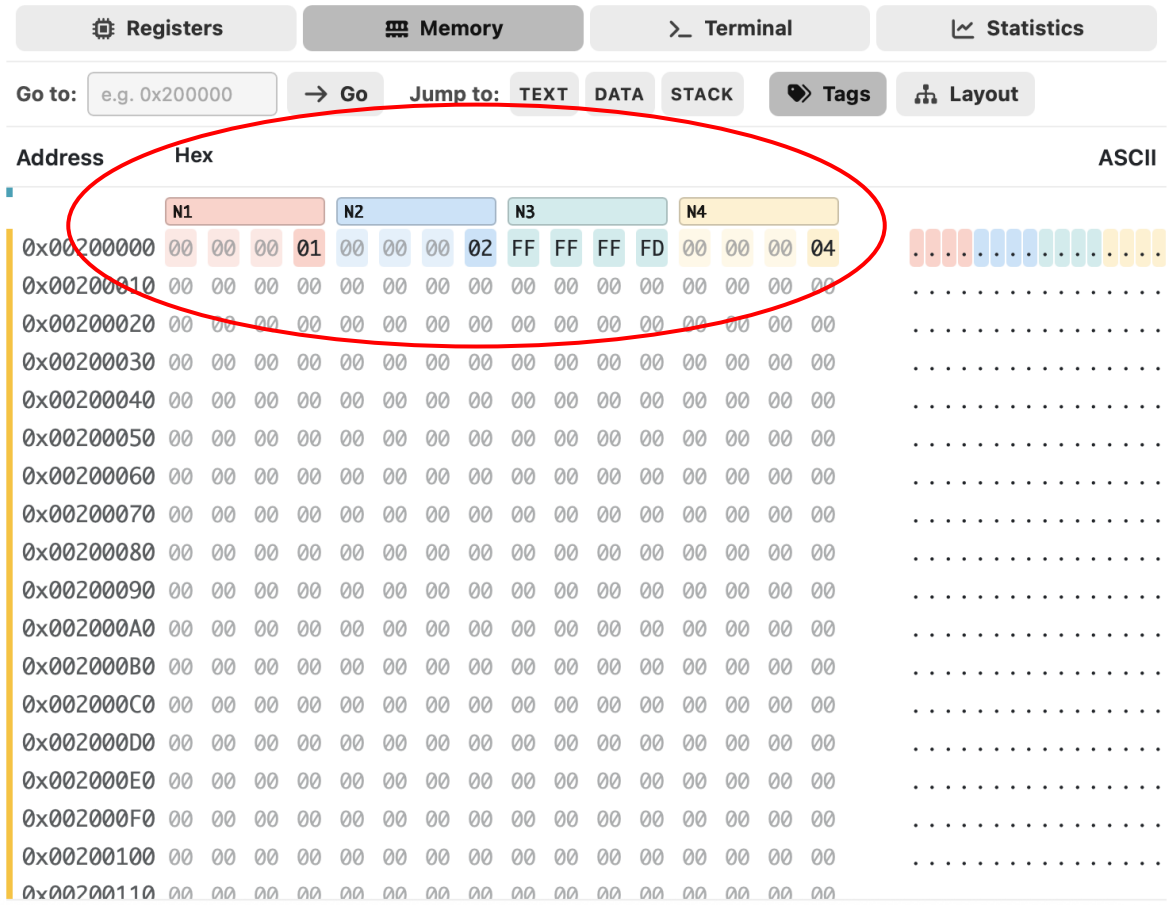
Ejemplo: antes de la ejecución

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)



Ejemplo: después de la ejecución

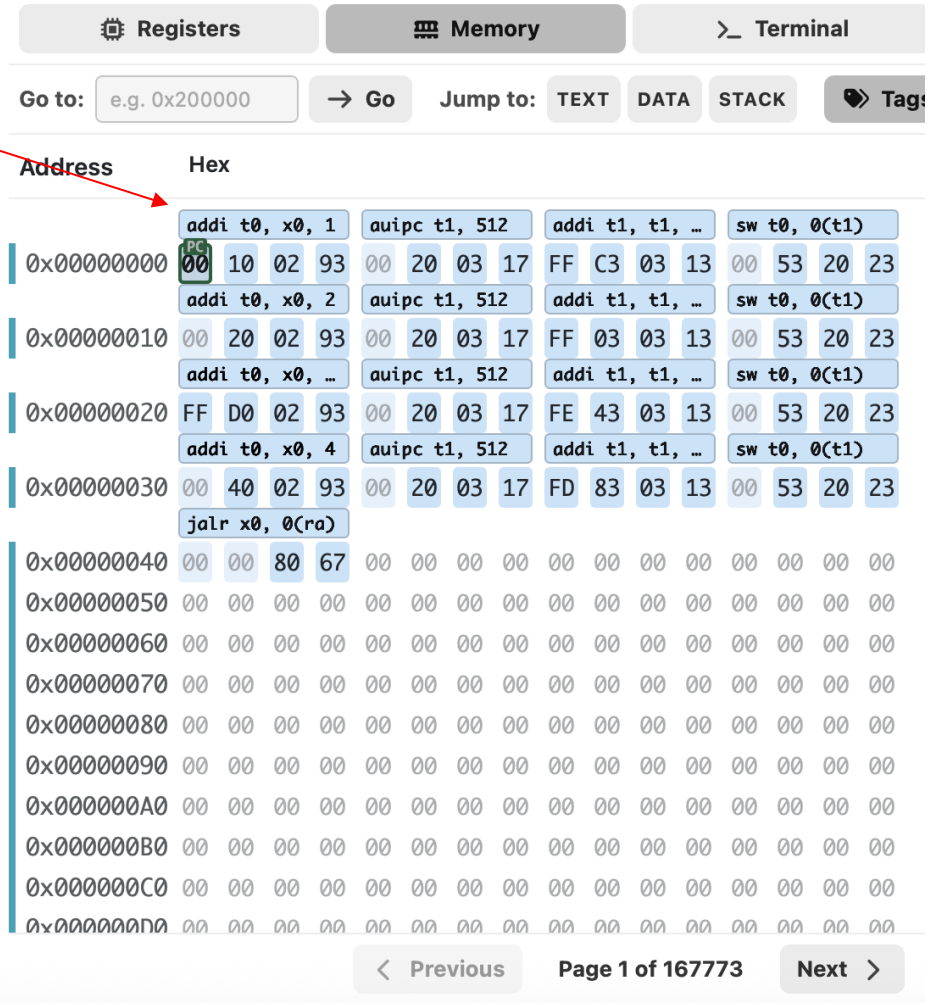
Address	User Instruction	Loaded Instructions
0x0 main	li t0, 1	addi t0, x0, 1
0x4	la t1, N1	auipc t1, 512
0x8		addi t1, t1, -4
0xC	sw t0, 0(t1)	sw t0, 0(t1)
0x10	li t0, 2	addi t0, x0, 2
0x14	la t1, N2	auipc t1, 512
0x18		addi t1, t1, -16
0x1C	sw t0, 0(t1)	sw t0, 0(t1)
0x20	li t0, -3	addi t0, x0, -3
0x24	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)



Visualización del segmento de texto
Instrucciones y pseudoinstrucciones

ejemplo

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)
0x40	jr ra	jalr x0, 0(ra)

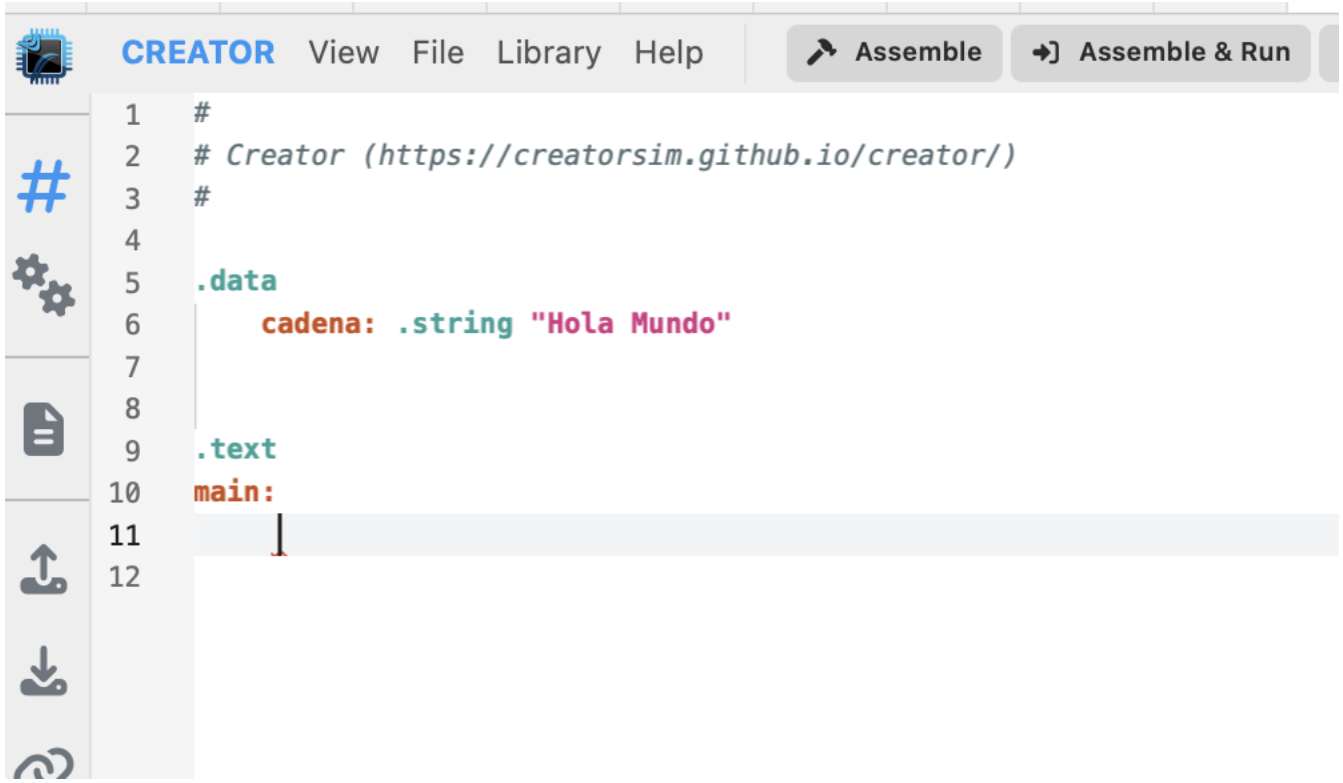


Programa escrito
(inst. y pseudoinst.)

Programa en memoria
(instrucciones máquina)

Calcular la longitud de una cadena

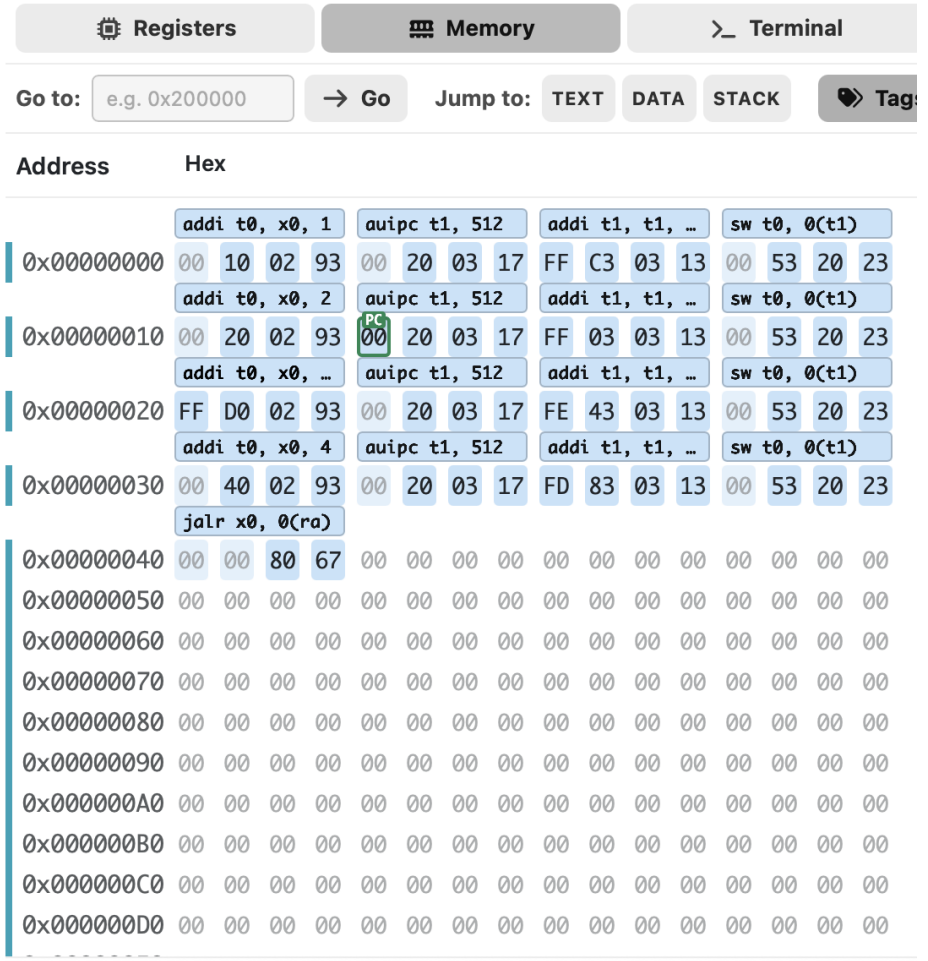
ejemplo



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)



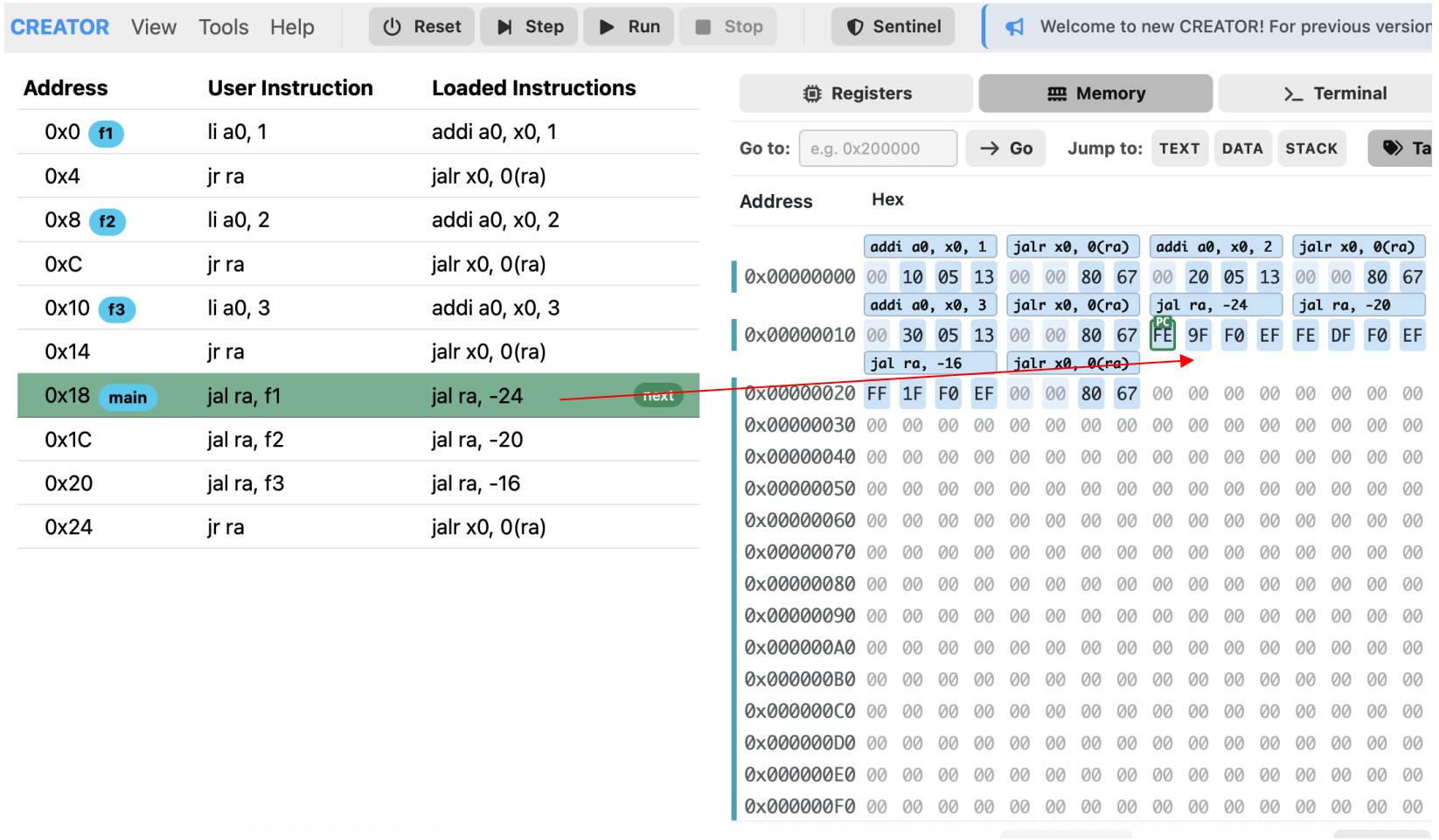
Visualización del segmento de texto
Varias funciones

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



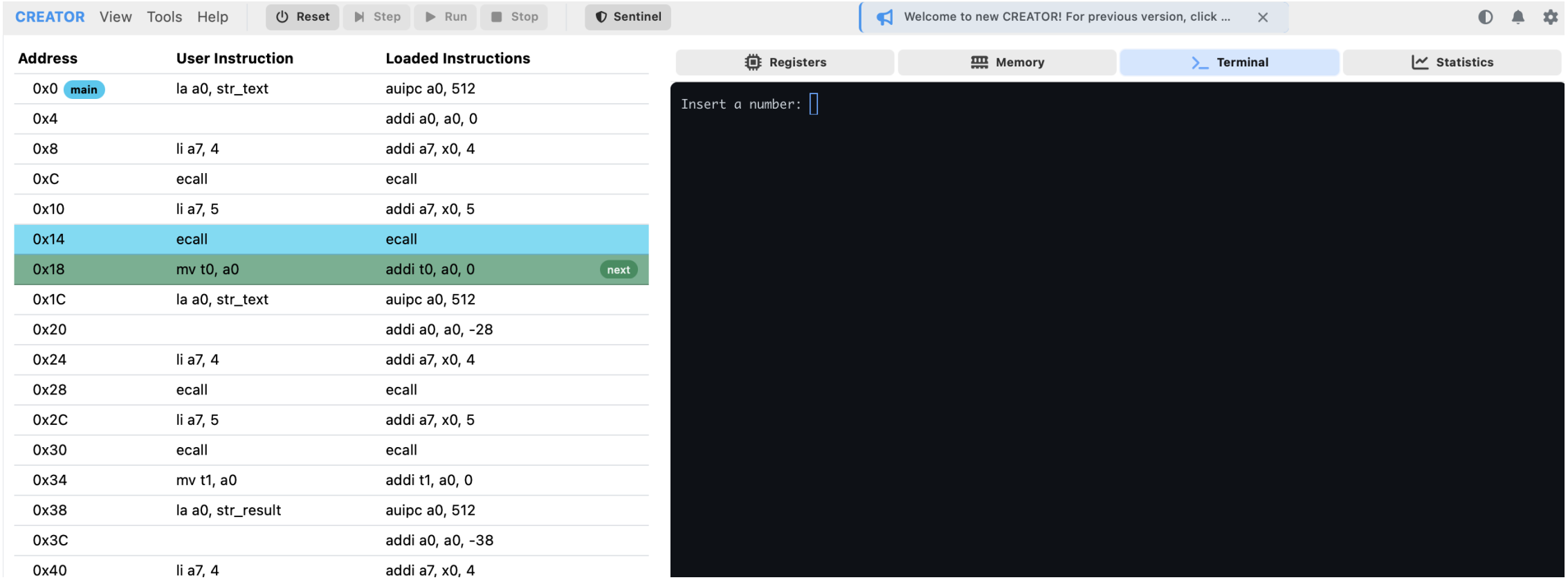
Llamadas al sistema

ejemplo

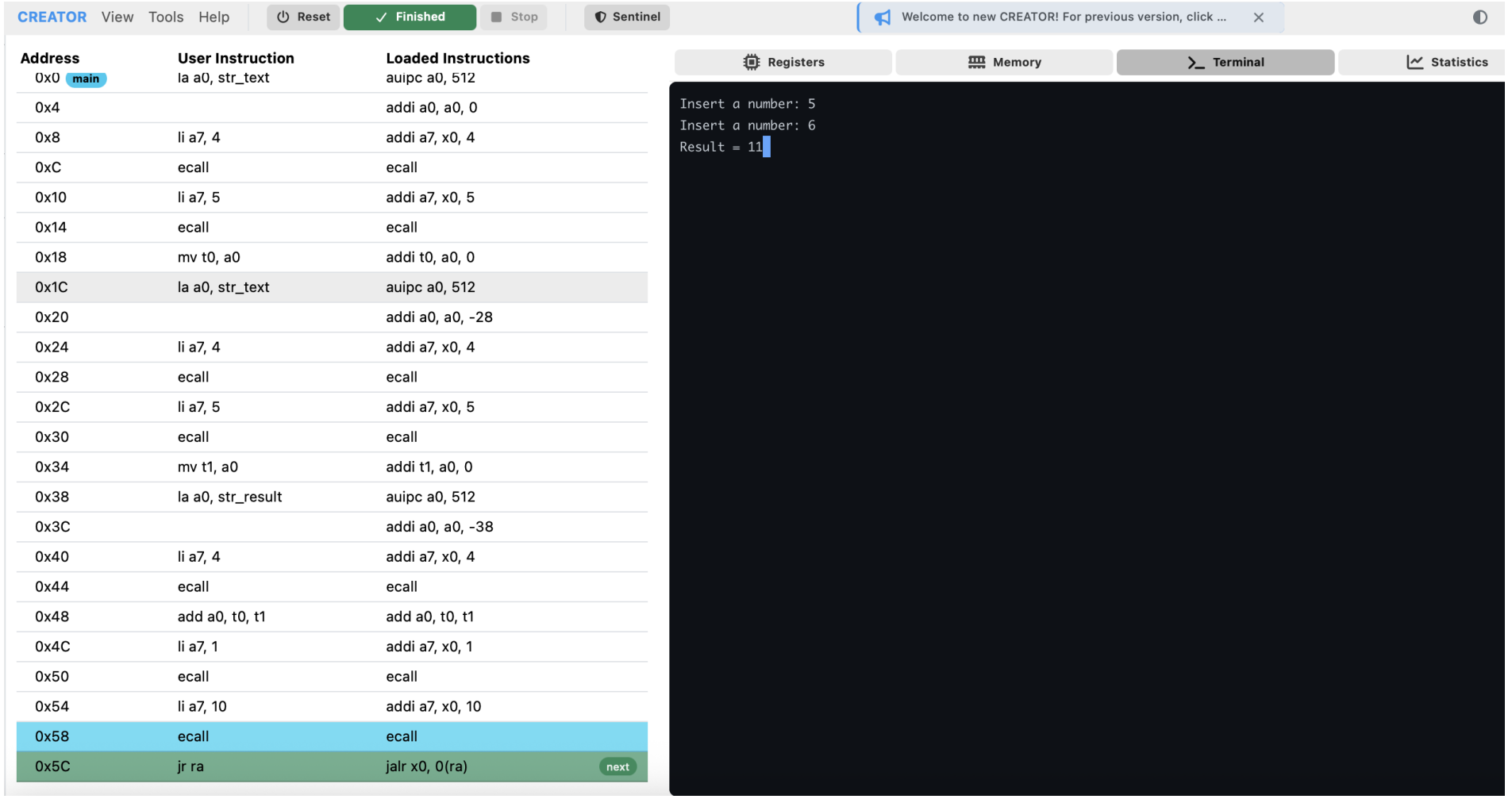
CREATOR View File Library Help Assemble Assemble & Run Vim: off

```
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
```

Llamadas al sistema



Llamadas al sistema



Ejemplo: ejecutar el siguiente programa

ejemplo

CREATOR View File Library Help Assemble Assem

```
1  #
2  # Creator (https://creatorsim.github.io/creator/)
3  #
4
5  .data
6
7       string1:    .string  "Hola Mundo"
8       string2:    .zero 32
9
10 .text
11      main:
12                          la    t0, string1
13                          la    t1, string2
14
15              loop:       lbu   t2, 0(t0)
16                          beq   t2, zero, end
17                          sb          t2, 0(t1)
18                          addi  t0, t0, 1
19                          addi  t1, t1, 1
20                          j     loop
21               end:
22                          li a7, 10
23                          ecall
24
25                          jr ra
26
27
```

CREATOR View Tools Help Reset Step Run Stop Sentinel

^ ⌥ C

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

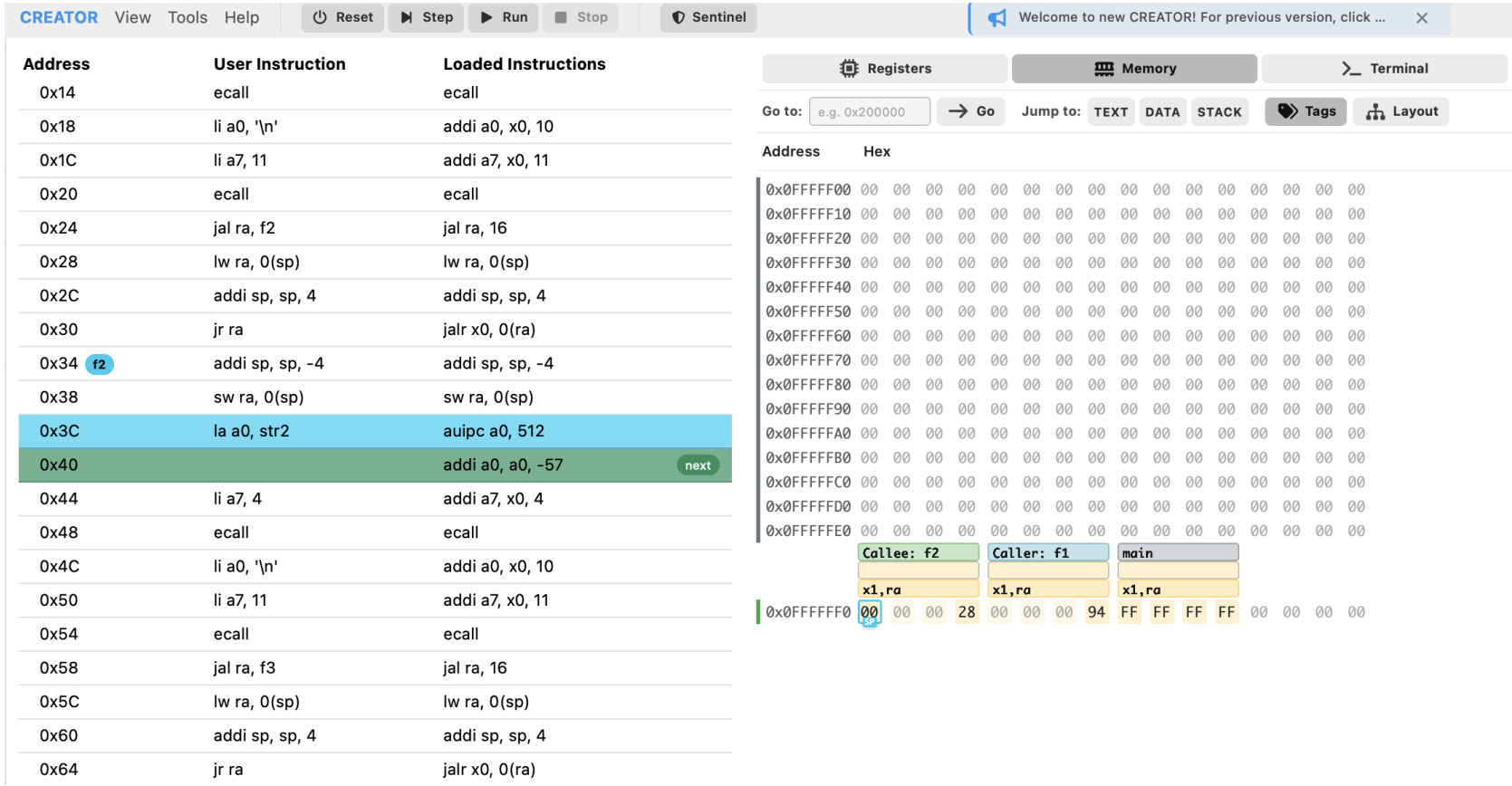
CREATOR View Tools Help | Reset | Step | Run | Stop | Sentinel

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

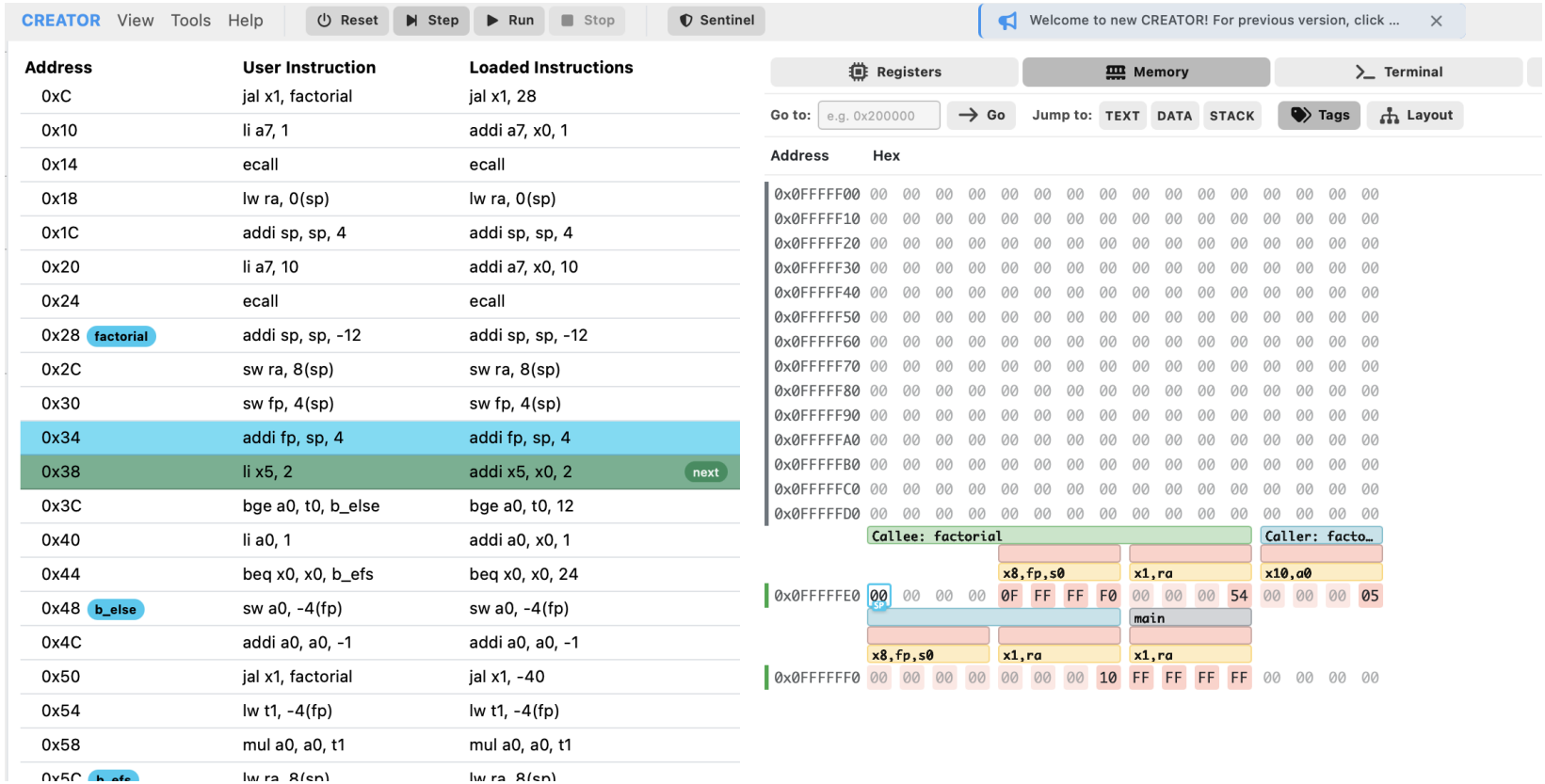
ejemplo

- Comprobar el crecimiento de la pila



Crecimiento de la pila en factorial

ejemplo



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
```

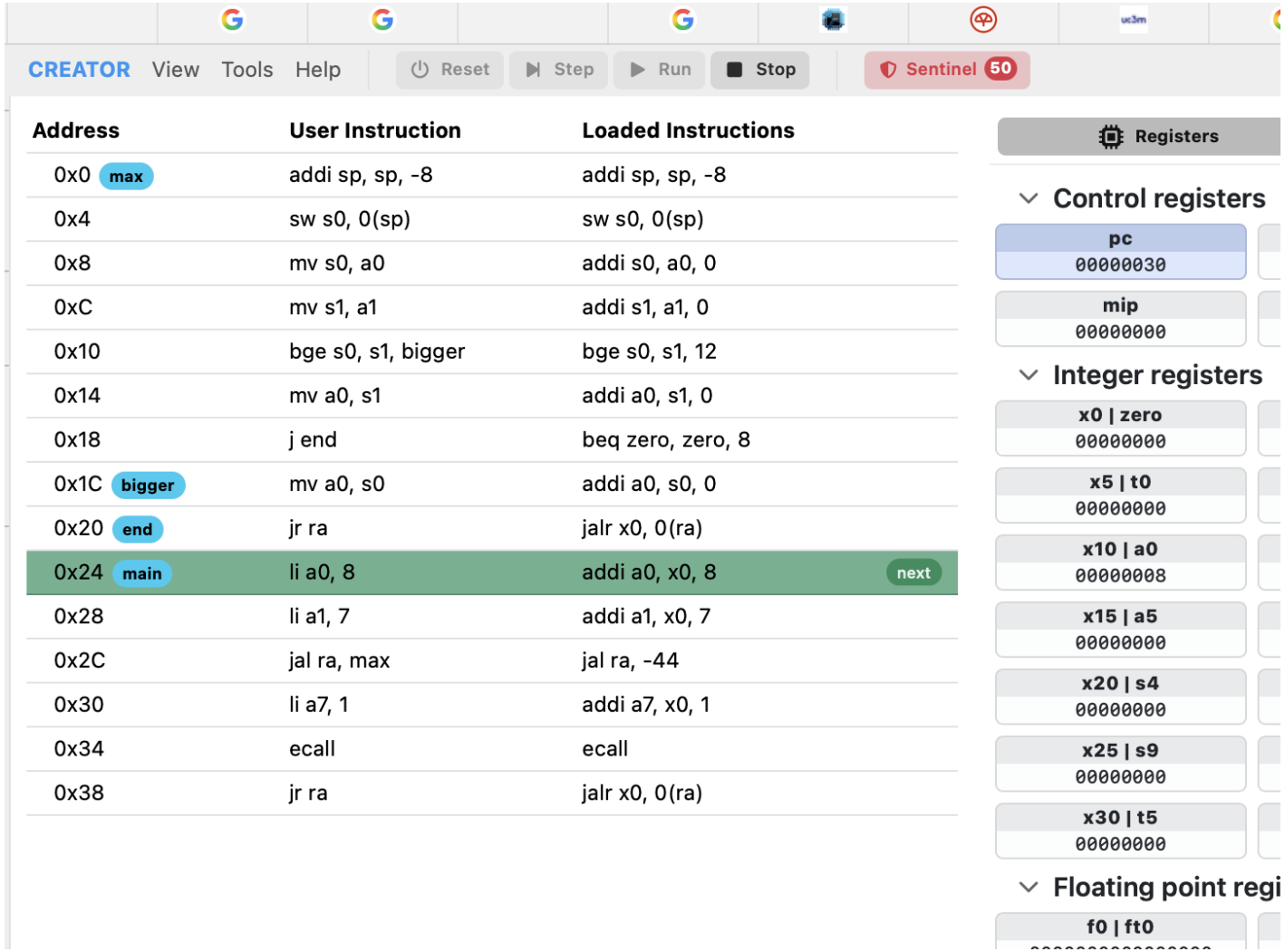
Detección de errores en el convenio de paso de parámetros

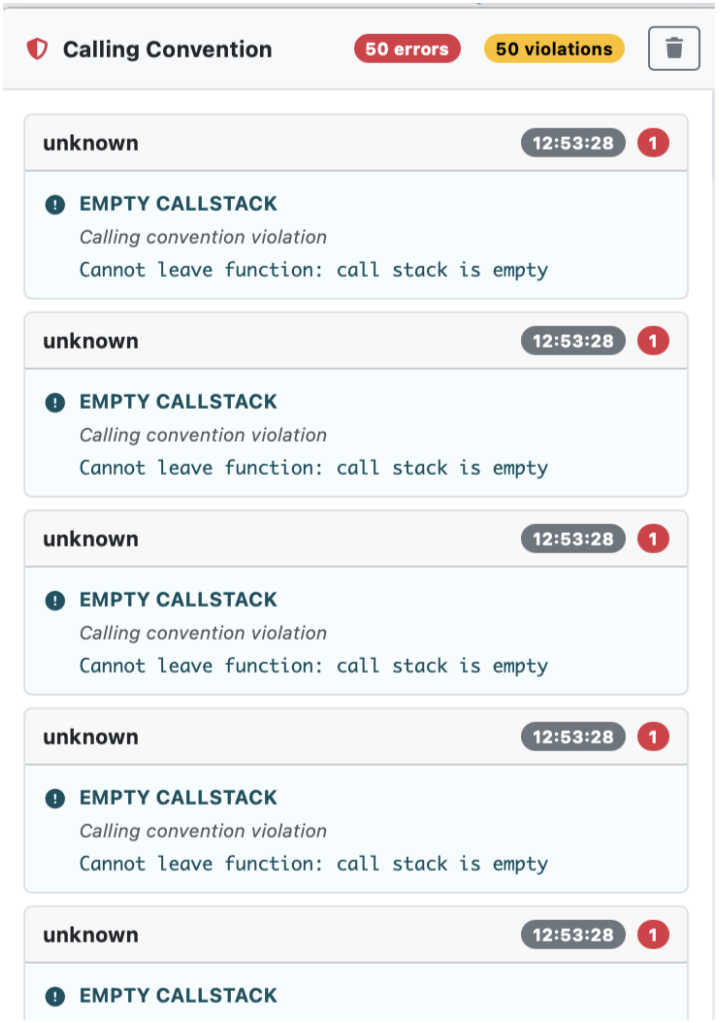
CREATOR View File Library Help Assemble Assemble

```
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
```

ejemplo

Detección de errores en el convenio de paso de parámetros





¿Cómo corregirlo?

Solución

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
```

ejemplo

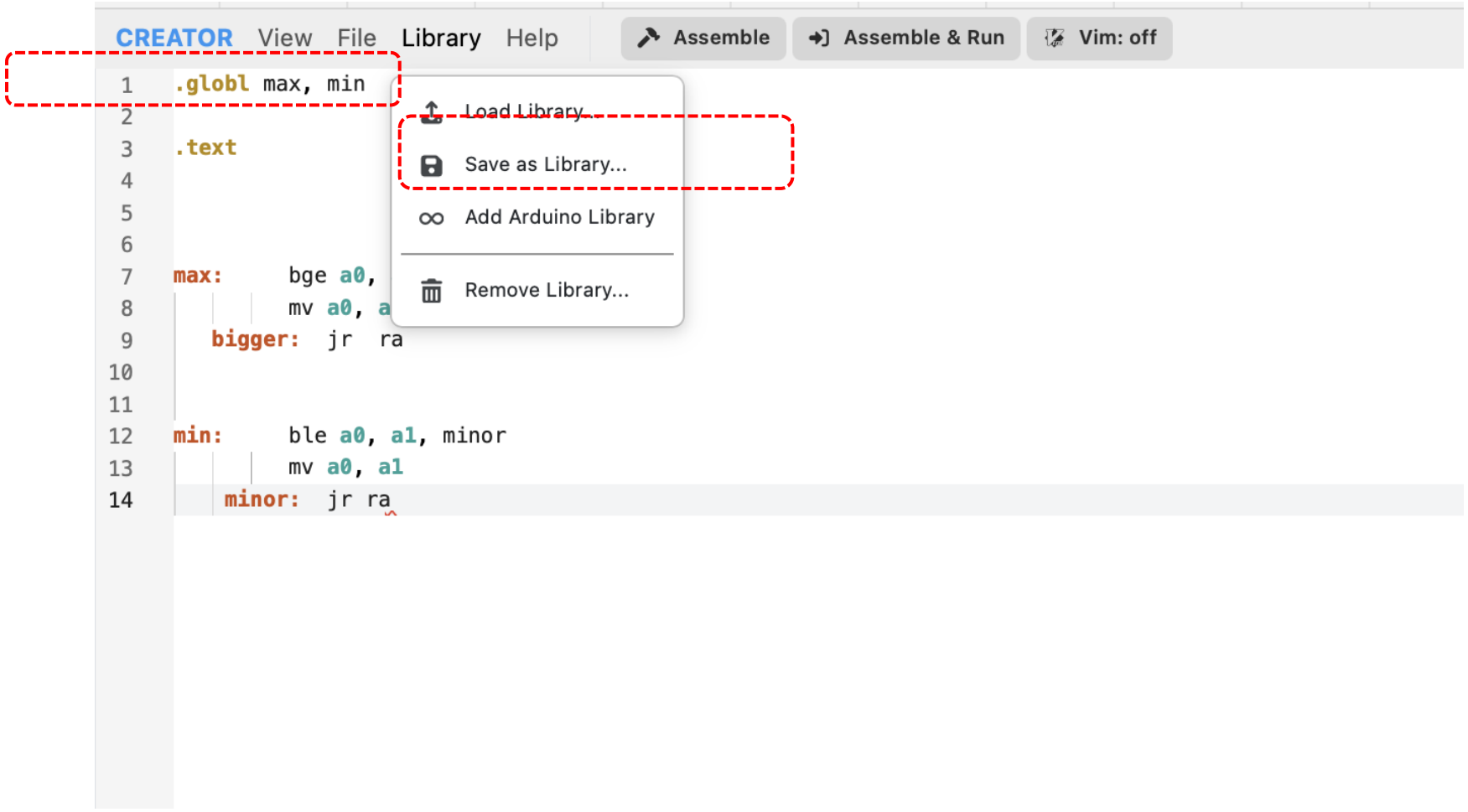
Creación de librerías

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



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

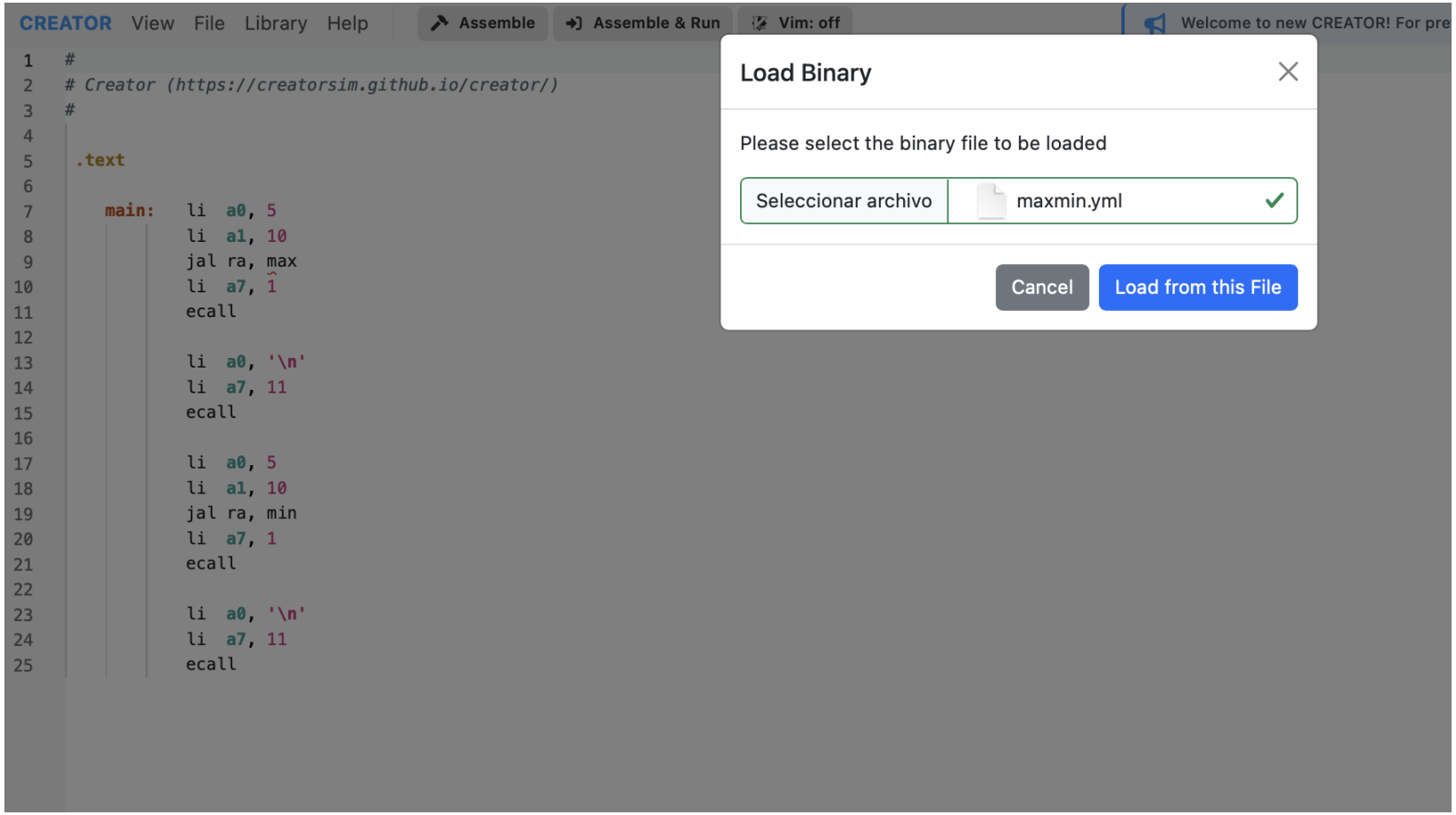
CREATOR View File Library Help | Assemble | Assemble & Run | Vim: off

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

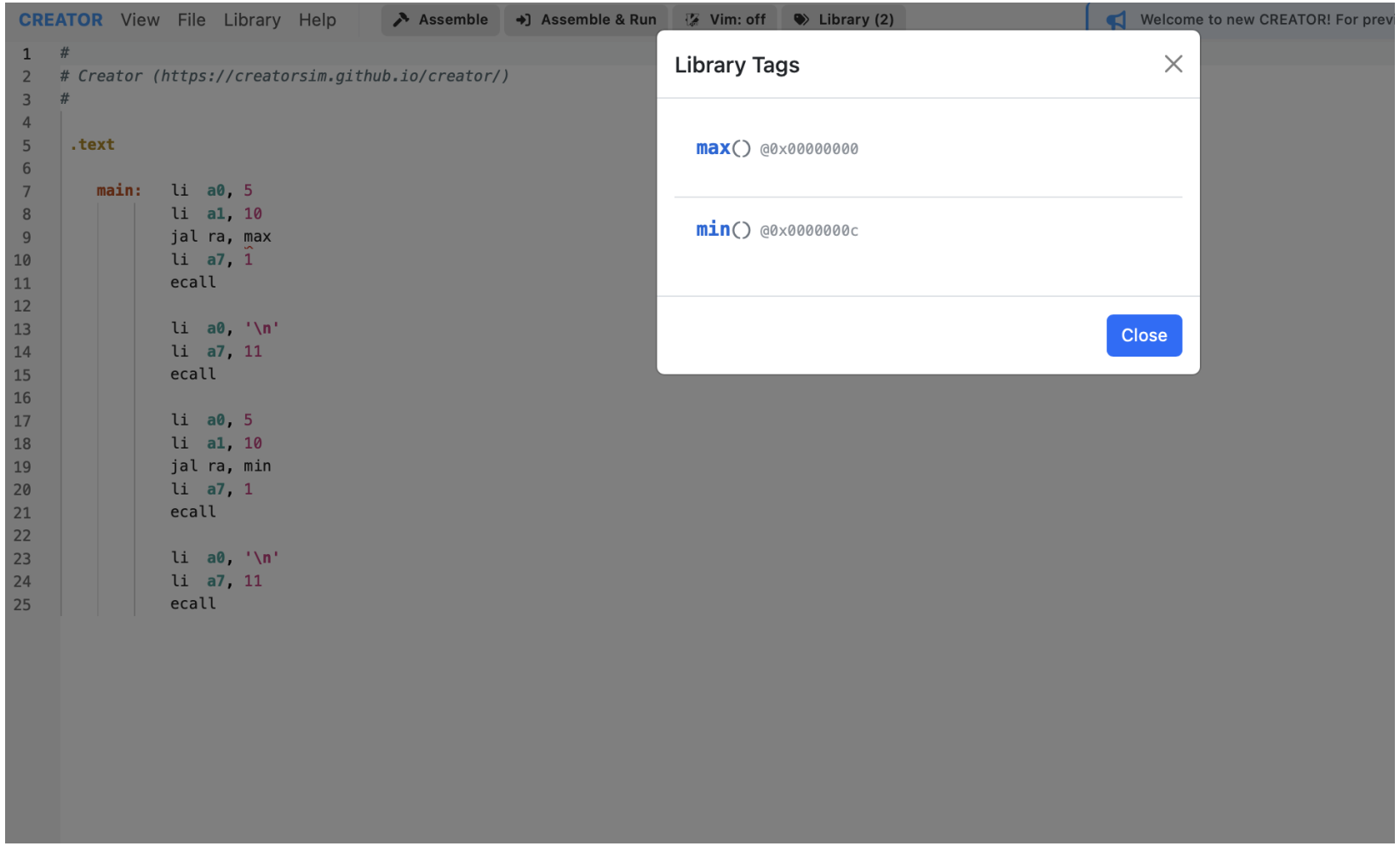
Assembly Code Error

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

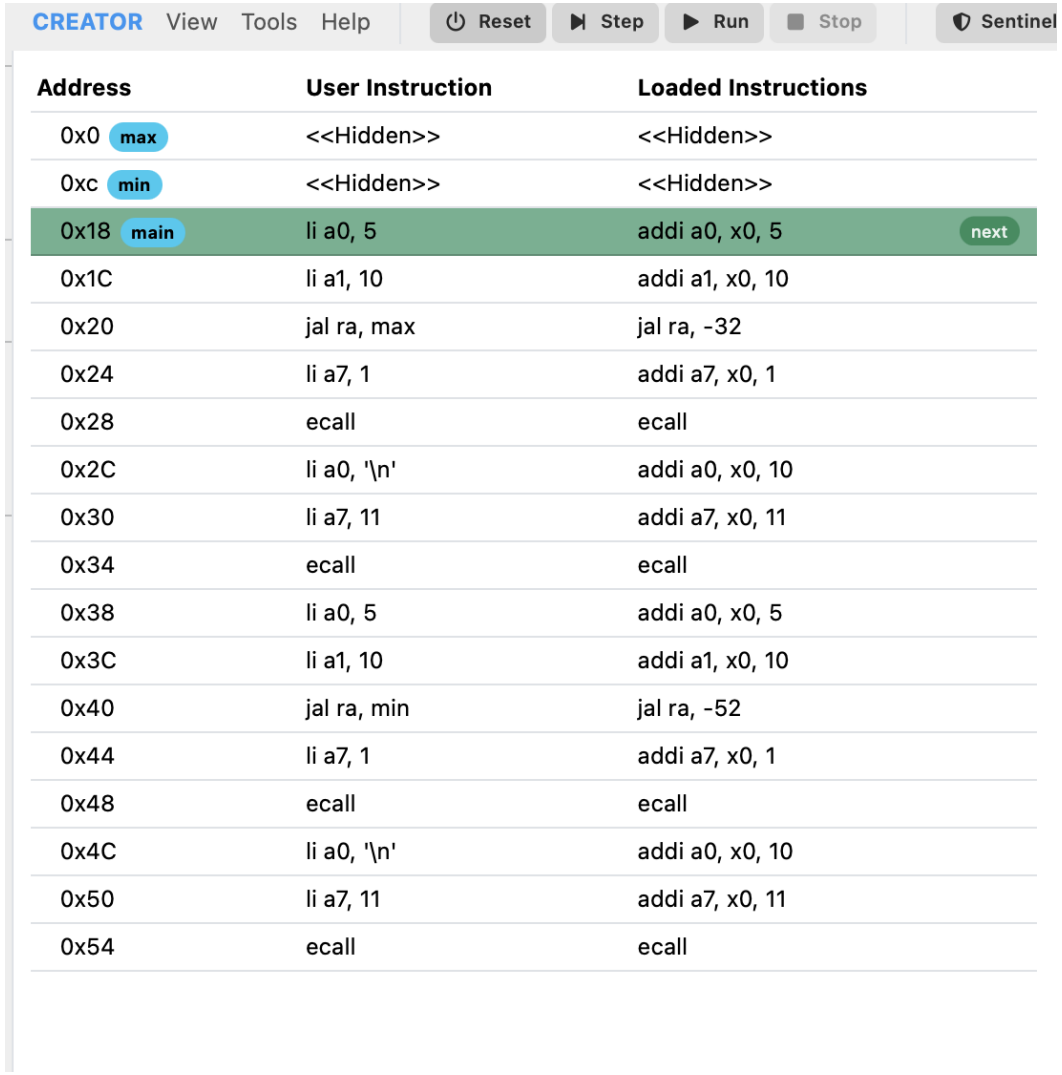
Librería cargada



Librería cargada



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



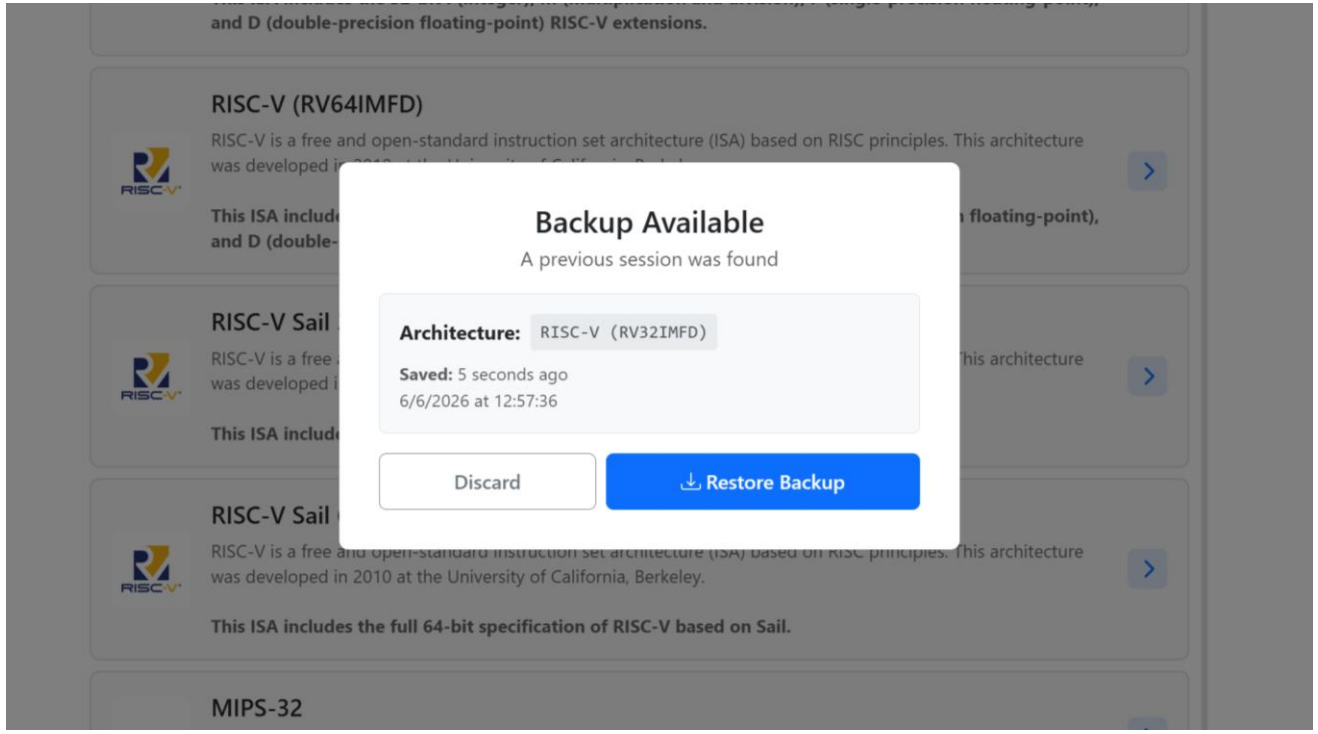
Address	User Instruction	Loaded Instructions
0x0 max	<<Hidden>>	<<Hidden>>
0xc min	<<Hidden>>	<<Hidden>>
0x18 main	li a0, 5	addi a0, x0, 5 next
0x1C	li a1, 10	addi a1, x0, 10
0x20	jal ra, max	jal ra, -32
0x24	li a7, 1	addi a7, x0, 1
0x28	ecall	ecall
0x2C	li a0, '\n'	addi a0, x0, 10
0x30	li a7, 11	addi 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
```

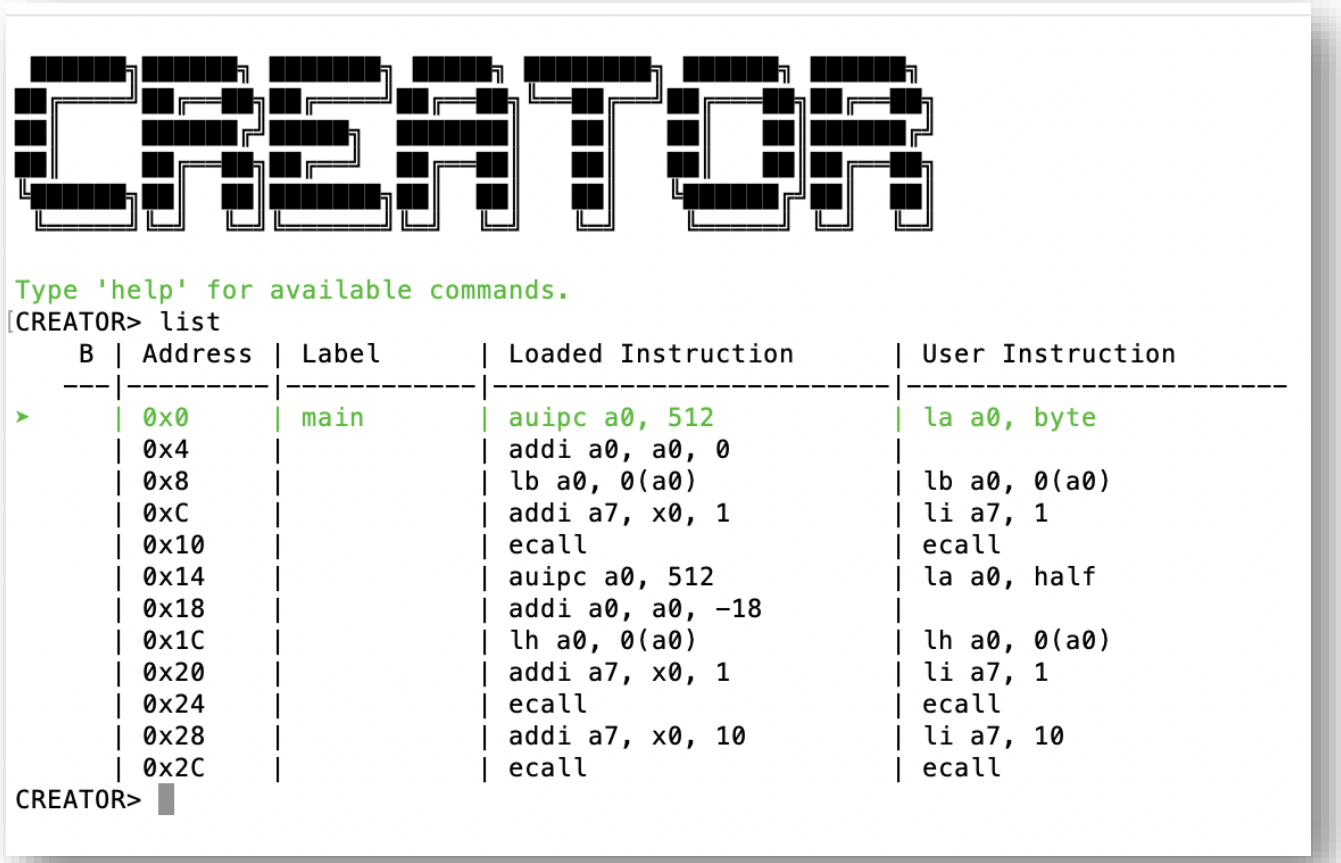
Soporte a la corrección de prácticas: interfaz en línea de comandos

- 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



Comandos

```
Available commands:
  step                                    - Execute one instruction
  unstep                                  - Undo last instruction
  run [n]                                 - Run n instructions or until program completes
  silent [n]                              - Run silently
  break [addr]                            - Set/unset breakpoint at address or list all
  reg list                                - List available register types
  reg <type>                              - Display registers of type
  reg <name>                              - Display specific register
  mem <address> [count]                   - Display memory (count in bytes)
  list                                    - Show all loaded instructions
  hexview <address> [count] [bytesPerLine]  - Hex viewer
  reset                                   - Reset program to initial state
  snapshot [filename]                     - Save a complete snapshot of current state
  restore <filename>                      - Restore a previously saved snapshot
  stack                                   - Display call stack hierarchy and frame info
  clear                                   - Clear the console screen
  help                                    - Show this help message
  alias                                   - Show available aliases
  about                                   - Show information about the simulator
  quit                                    - 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
```

```
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                         | ........
```

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
```

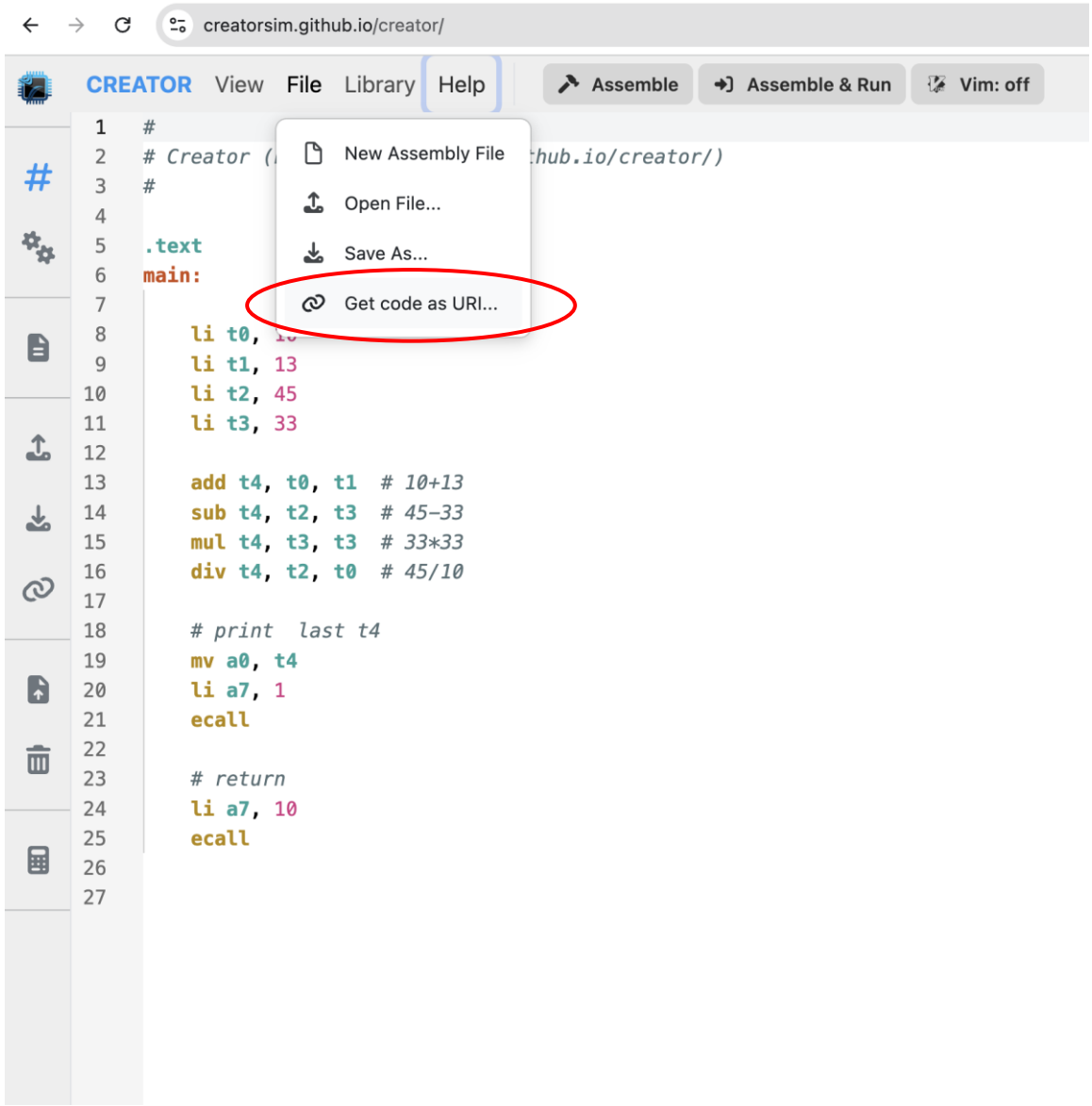
- "output.yml" comprueba la salida:

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

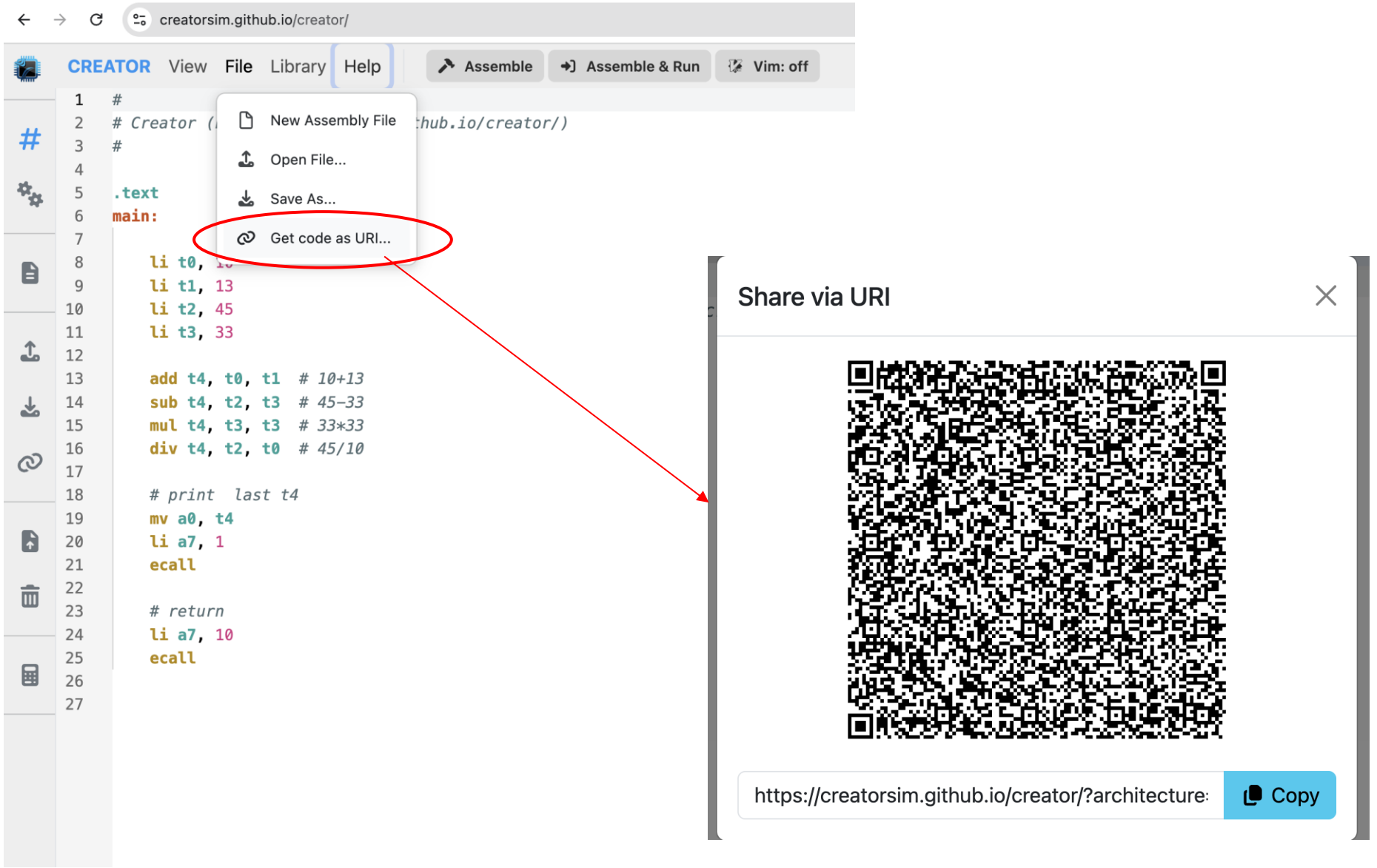
Ejemplo

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

Ayuda a la creación de materiales docentes

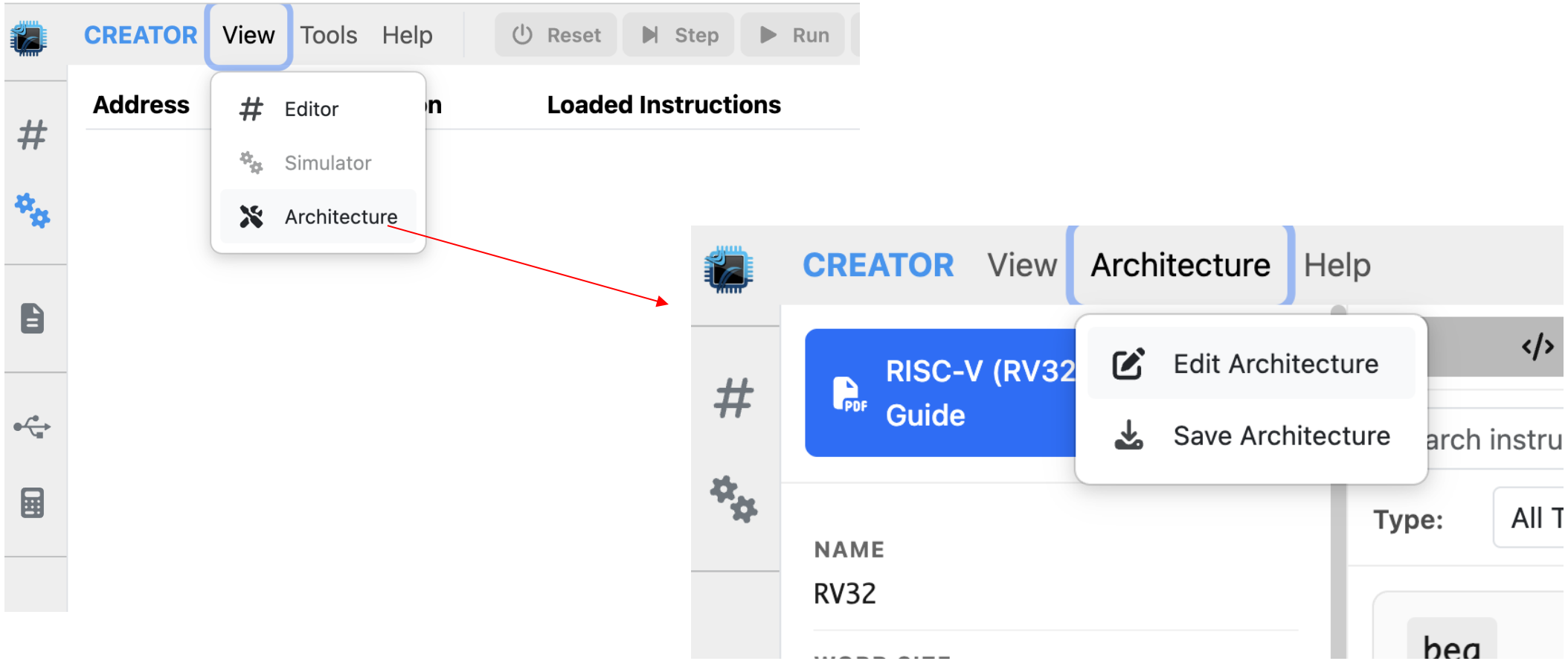


Ayuda a la creación de materiales docentes



Capacidades para extender el juego de instrucciones y crear nuevas arquitecturas

- Arquitectura definida en formato YAML



Edición de la arquitectura

Edit Architecture

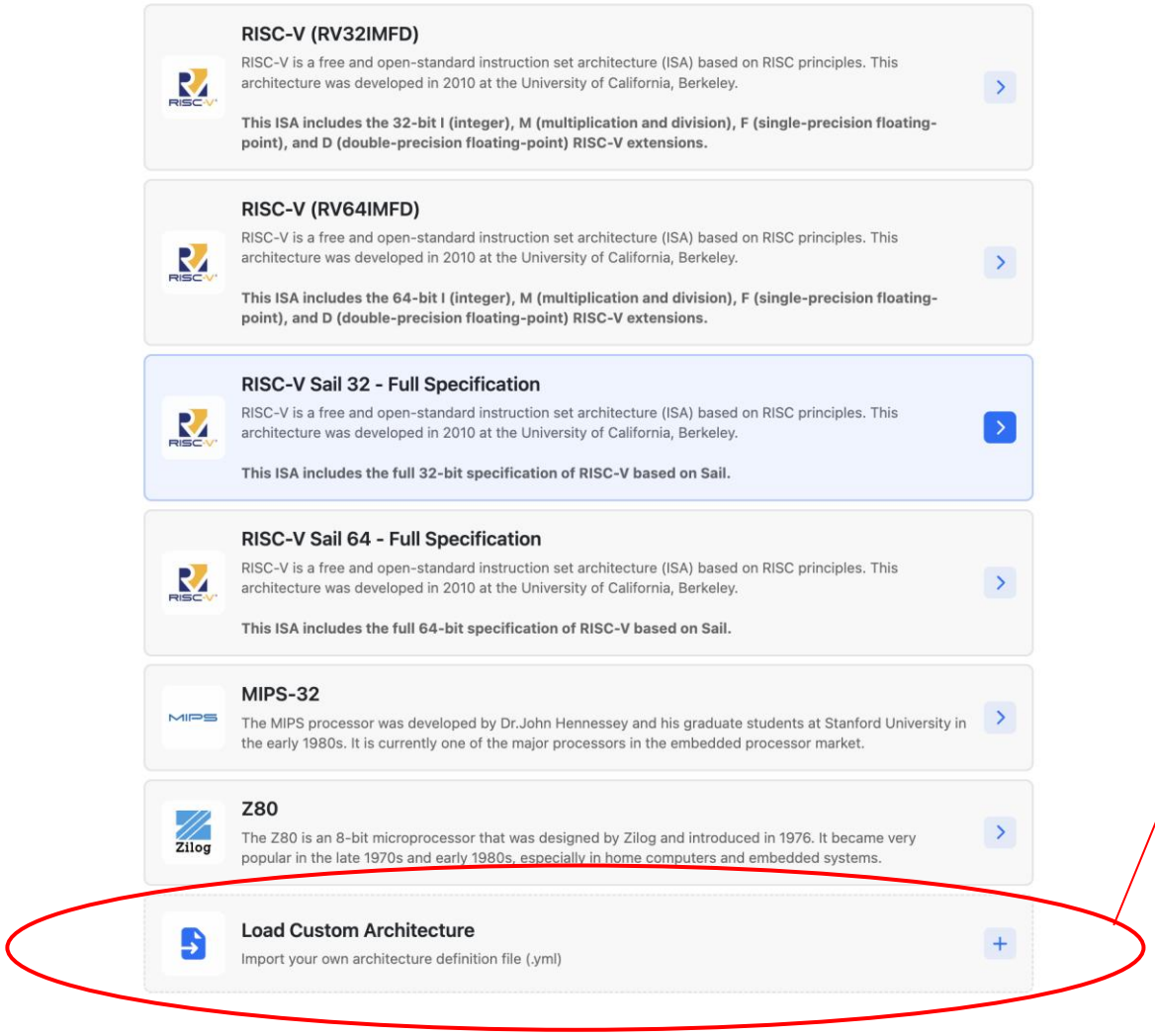
```yaml
version: 2.0.0

# ASCII text generated w/ https://patorjk.com/software/taag/#p=display&h=1&f=Standard
# instruction help text extracted from https://msyksphinz-self.github.io/riscv-isadoc/html/index.html

config:
  name: 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: true
  main_function: main
  passing_convention: true
  sensitive_register_name: true
  comment_prefix: "#"
  start_address: 0x0
  pc_offset: -4
  plugin: riscv
  assemblers:
    - name: CreatorAssembler
      description: "Default CREATOR assembler"

extensions:
  I:
    description: "RV32I Base Instruction Set"
    template: Base
  M:
```

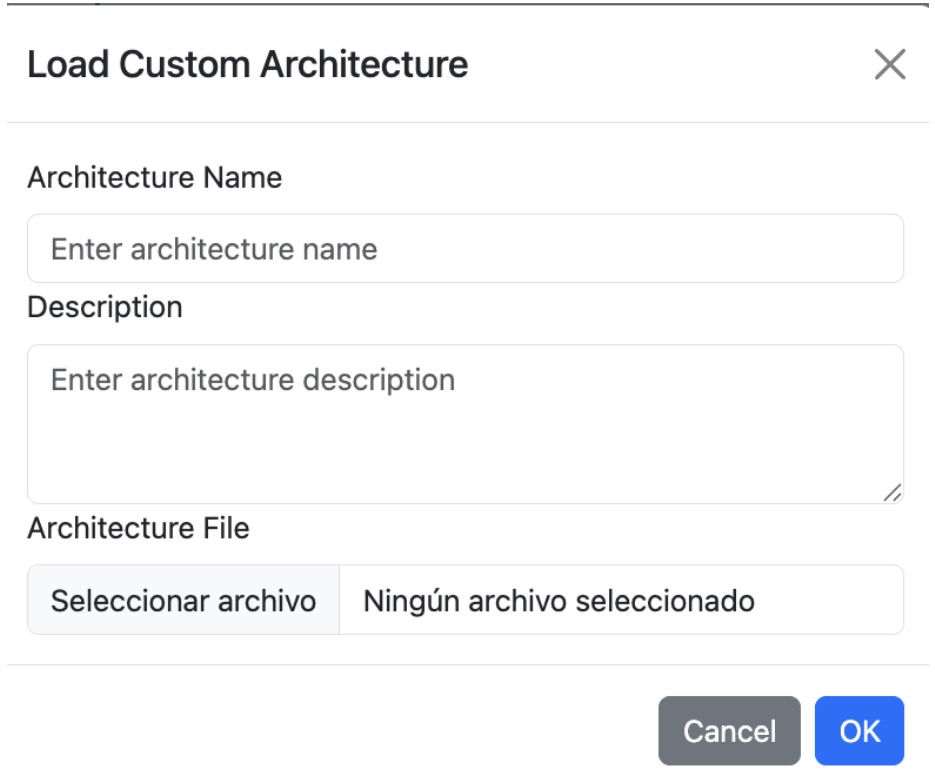
Cancel Save

Ejercicio propuesto

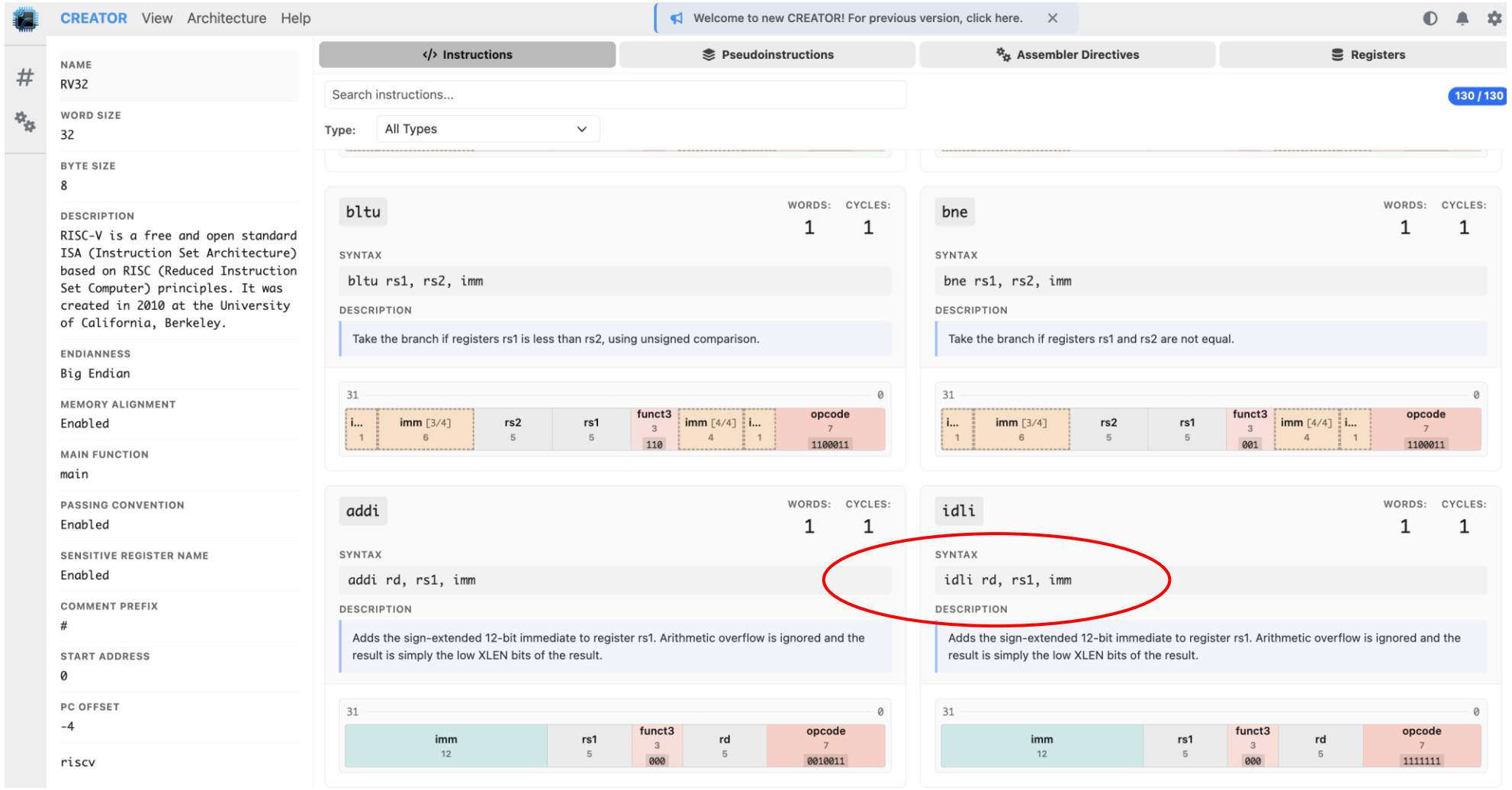
- Añadir una nueva pseudoinstrucción:
  - psdli rd1, rd2, inm → rd1 = inm y rd2 = inm
- Añadir una nueva instrucción:
  - ldli rd1, rd2, inm → rd1 = inm y rd2 = inm
    - Código de operación: 11111111
- Pasos:
  - Descargar el archivo YAML con la arquitectura
  - Editar y entender el archivo con la arquitectura
  - Modificar la arquitectura incorporando psdli y ldli
  - Cargar la nueva arquitectura

Carga de una nueva arquitectura

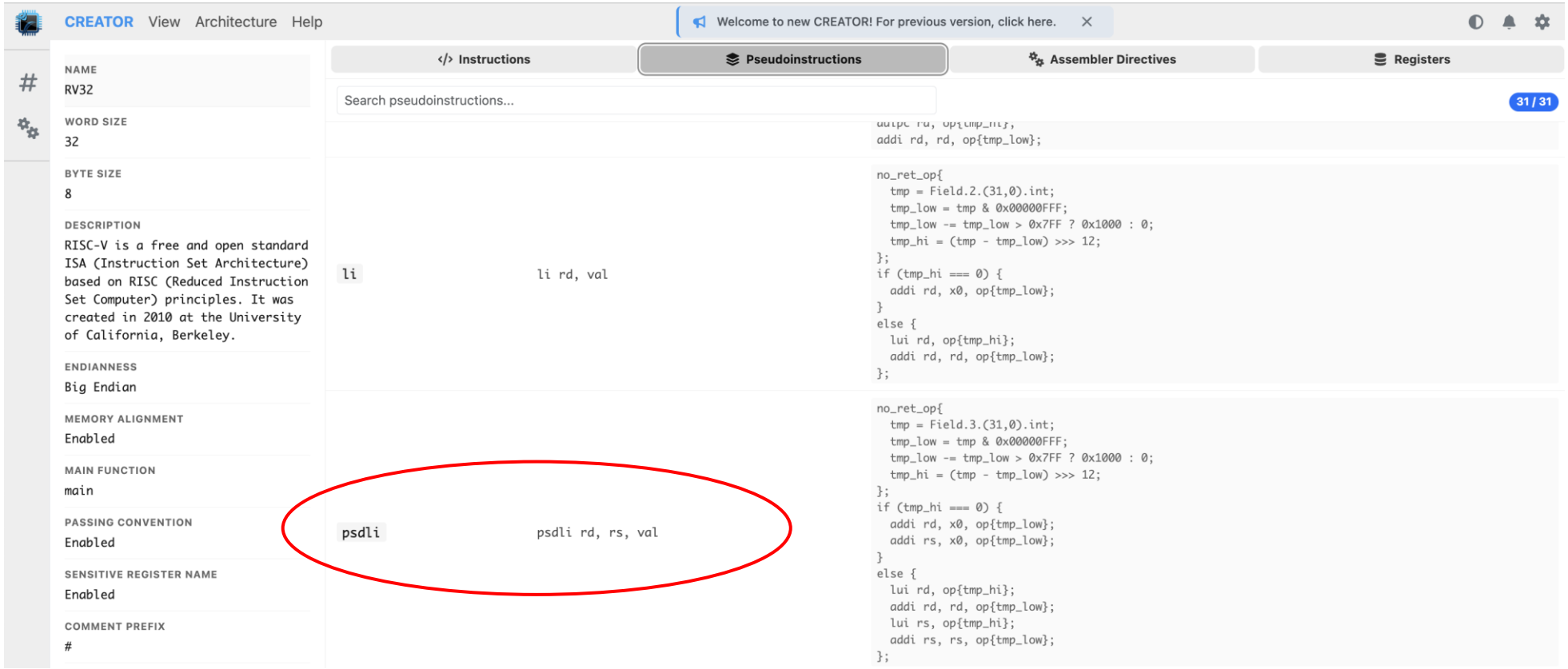




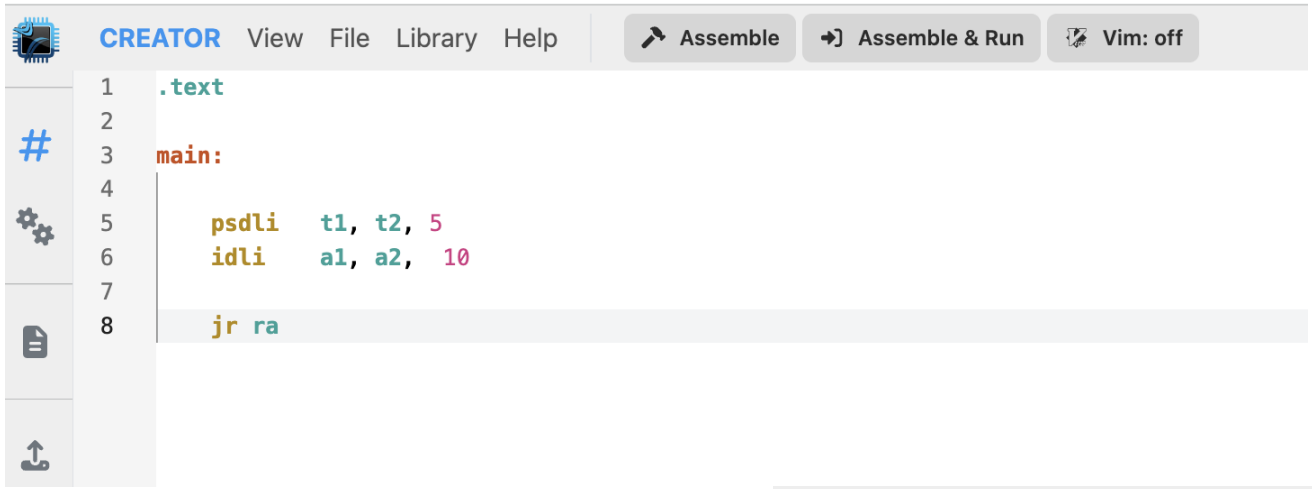
Nueva instrucción

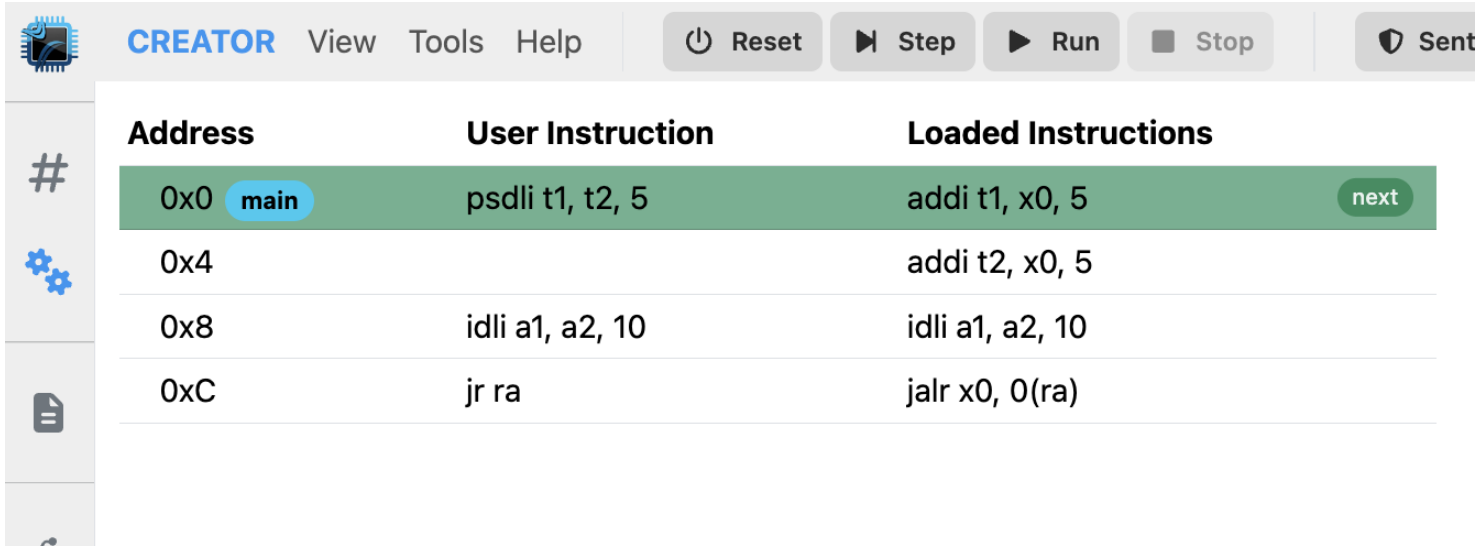


Nueva pseudoinstrucción



Programa con el nuevo juego de instrucciones





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`)

https://creatorsim.github.io/creator-wiki/development/devices.html

(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

https://creatorsim.github.io/creator-wiki/development/devices.html

(1/2) Consola

Ejemplo

- 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
```

https://creatorsim.github.io/creator-wiki/development/devices.html

(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)

https://creatorsim.github.io/creator-wiki/development/devices.html

(2/2) Dispositivo de sistema

Ejemplo

- 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
```

https://creatorsim.github.io/creator-wiki/development/devices.html

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

https://creatorsim.github.io/creator-wiki/development/devices.html

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

https://creatorsim.github.io/creator-wiki/development/privileged.html

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*)

https://creatorsim.github.io/creator-wiki/development/privileged.html

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

WiP

```
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 subrutine
      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) ;
      }
```

https://github.com/creatorsim/creator/blob/master/architecture/RISCV/RV32IMFD.yml#L4053

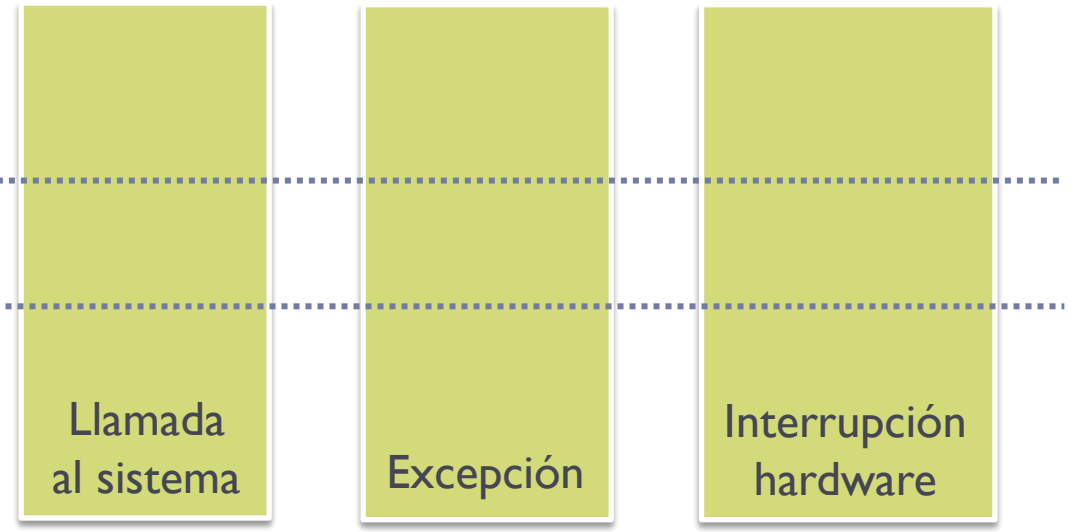
Kernel a usuario (retorno de *trap*)

```
- 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() ;
```

https://github.com/creatorsim/creator/blob/master/architecture/RISCV/RV32IMFD.yml#L3588

Gestión de eventos en CREATOR

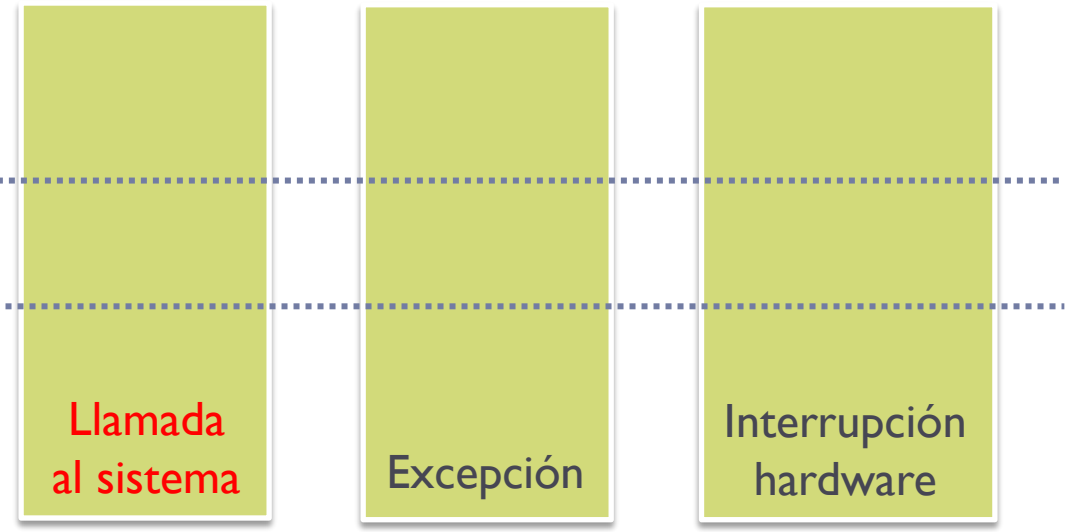
- Modos incluidos:
  - Modo usuario/a (no privilegiado)
  - Modo kernel (privilegiado)
- Transiciones:
  - Usuario a kernel (*trap*)
  - Kernel a usuario/a (retorno de *trap*)



https://creatorsim.github.io/creator-wiki/development/privileged.html

Gestión de eventos en CREATOR

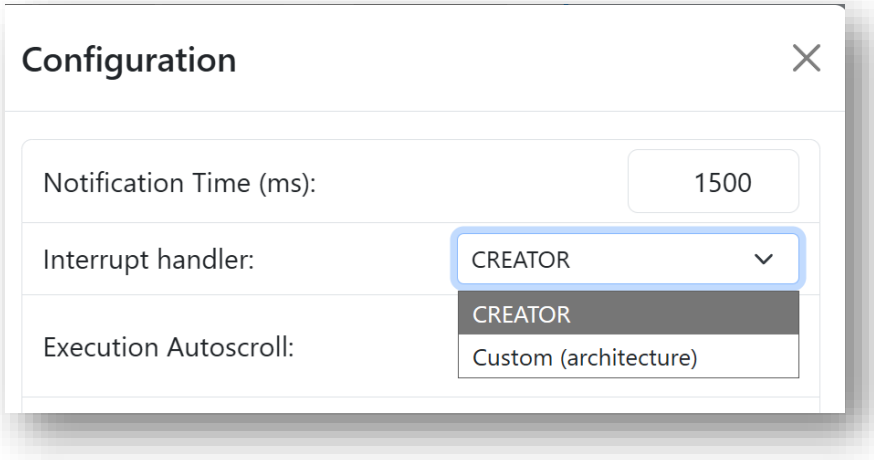
- Modos incluidos:
  - Modo usuario/a (no privilegiado)
  - Modo kernel (privilegiado)
- Transiciones:
  - Usuario a kernel (*trap*)
  - Kernel a usuario/a (retorno de *trap*)

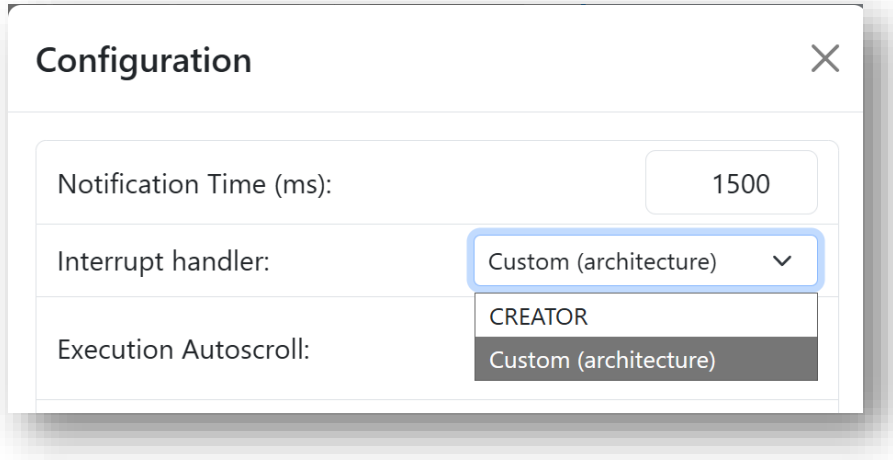


https://creatorsim.github.io/creator-wiki/development/privileged.html

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





https://creatorsim.github.io/creator-wiki/web/execution.html

Manejador por defecto en CREATOR

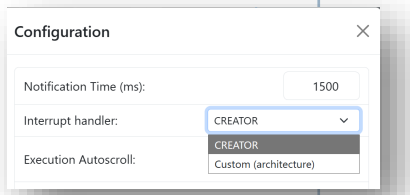
```
{
    "name": "ecall",
    "type": "Syscall",

    # …

    "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

https://github.com/creatorsim/creator-v4.0/blob/main/architecture/RISC_V_RV32IMFD.json#L1614C1-L1641C7

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();
```



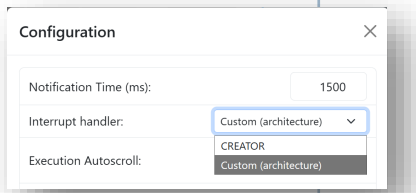
- Antes de añadir interrupciones a CREATOR, la definición de la función `ecall` de RISC-V ejecutaba la llamada al sistema deseada en función del valor del registro a7.
- Se analizó cómo añadir soporte de interrupciones a CREATOR y que `ecall` usara dicho soporte:
  - No debería de obligar a visibilizar 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)

https://creatorsim.github.io/creator-wiki/web/execution.html

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;
          }
```

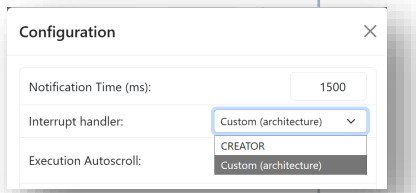


- Antes de añadir interrupciones a CREATOR, la definición de la función `ecall` de RISC-V ejecutaba la llamada al sistema deseada en función del valor del registro a7.
- Se analizó cómo añadir soporte de interrupciones a CREATOR y que `ecall` usara dicho soporte:
  - No debería de obligar a visibilizar 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)

https://creatorsim.github.io/creator-wiki/web/execution.html

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;
        }
```



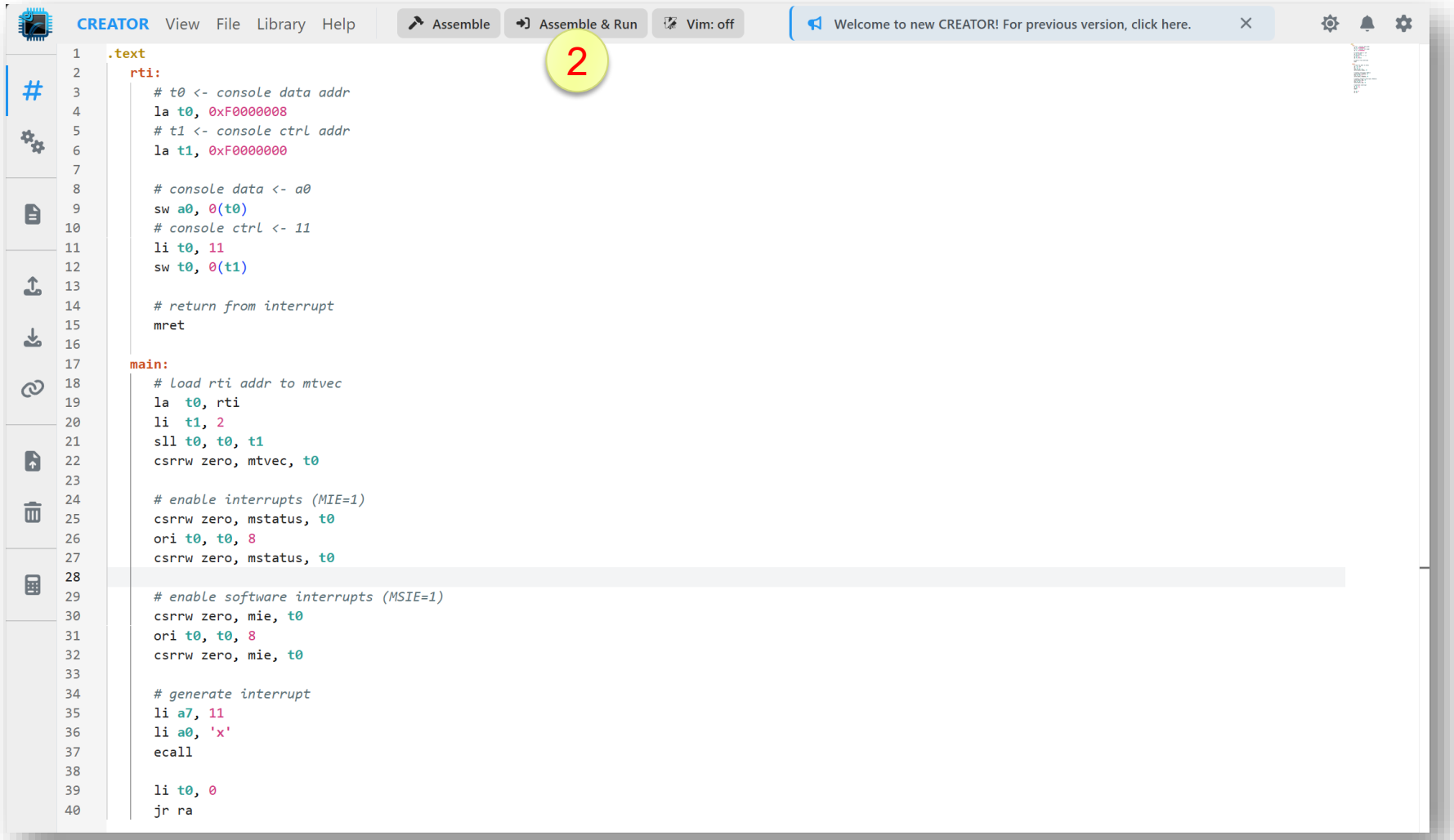
- Antes de añadir interrupciones a CREATOR, la definición de la función `ecall` de RISC-V ejecutaba la llamada al sistema deseada en función del valor del registro a7.
- Se analizó cómo añadir soporte de interrupciones a CREATOR y que `ecall` usara dicho soporte:
  - No debería de obligar a visibilizar 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)

https://creatorsim.github.io/creator-wiki/web/execution.html

Soporte para llamadas: ejemplo

Ejemplo

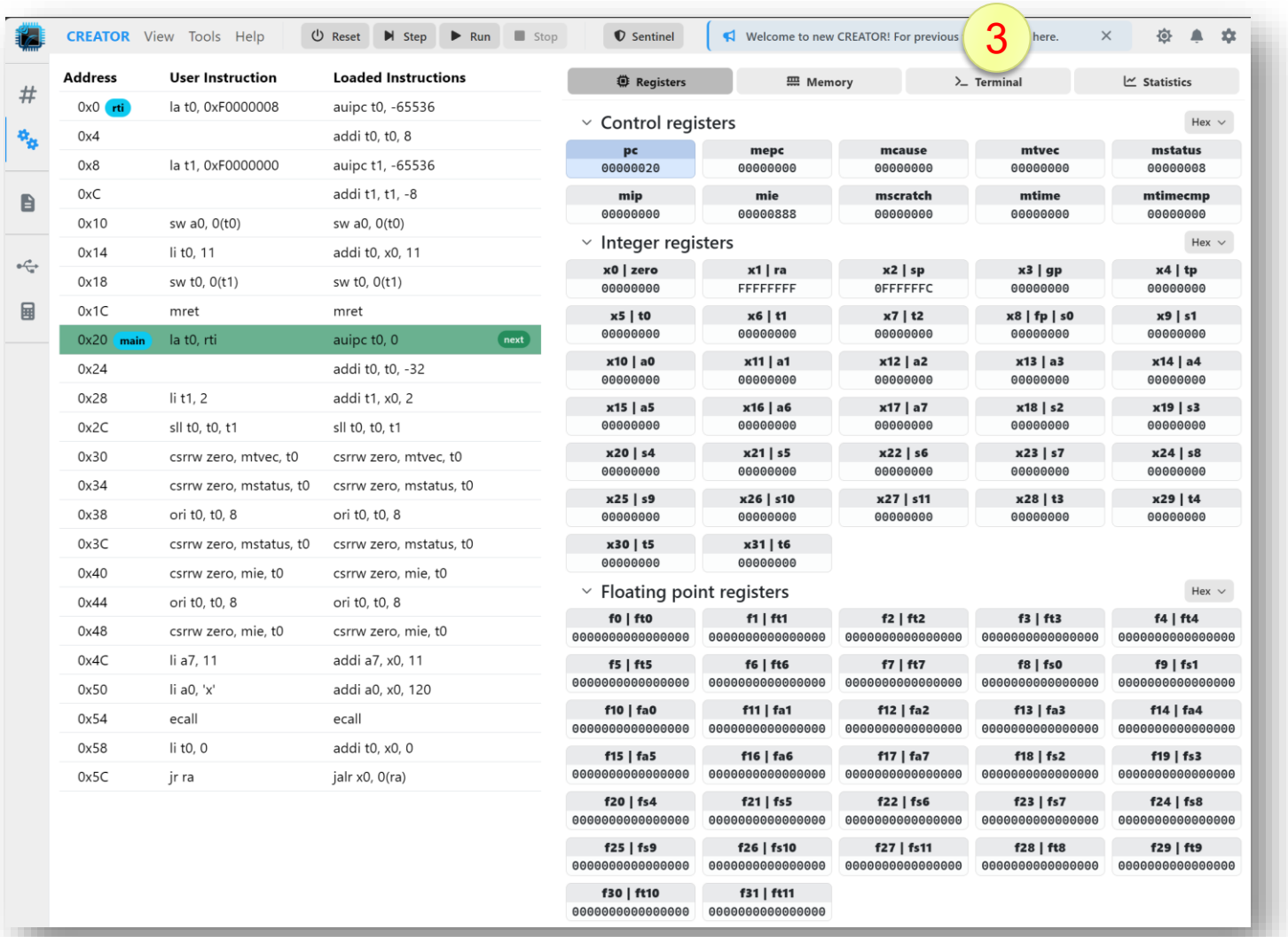
1




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

    # console data <- a0
    sw a0, 0(t0)
    # console ctrl <- 11
    li t0, 11
    sw t0, 0(t1)

    # return from interrupt
    mret

  main:
    # load rti addr to mtvec
    la  t0, rti
    li  t1, 2
    sll t0, t0, t1
    csrrw zero, mtvec, t0

    # enable interrupts (MIE=1)
    csrrw zero, mstatus, t0
    ori t0, t0, 8
    csrrw zero, mstatus, t0

    # enable software interrupts (MSIE=1)
    csrrw zero, mie, t0
    ori t0, t0, 8
    csrrw zero, mie, t0

    # generate interrupt
    li a7, 11
    li a0, 'x'
    ecall

    li t0, 0
    jr ra
```

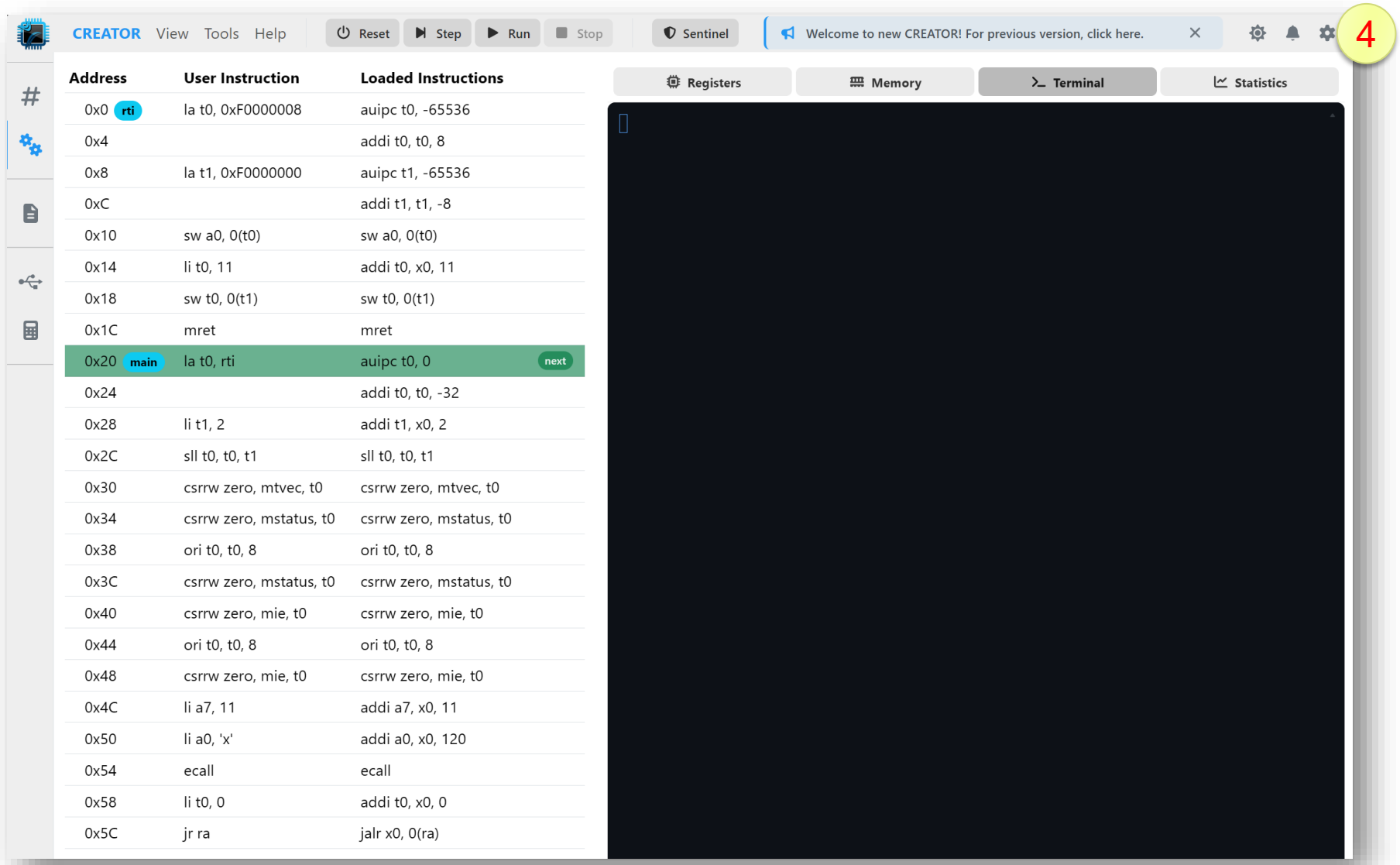
Soporte para llamadas: ejemplo

Ejemplo



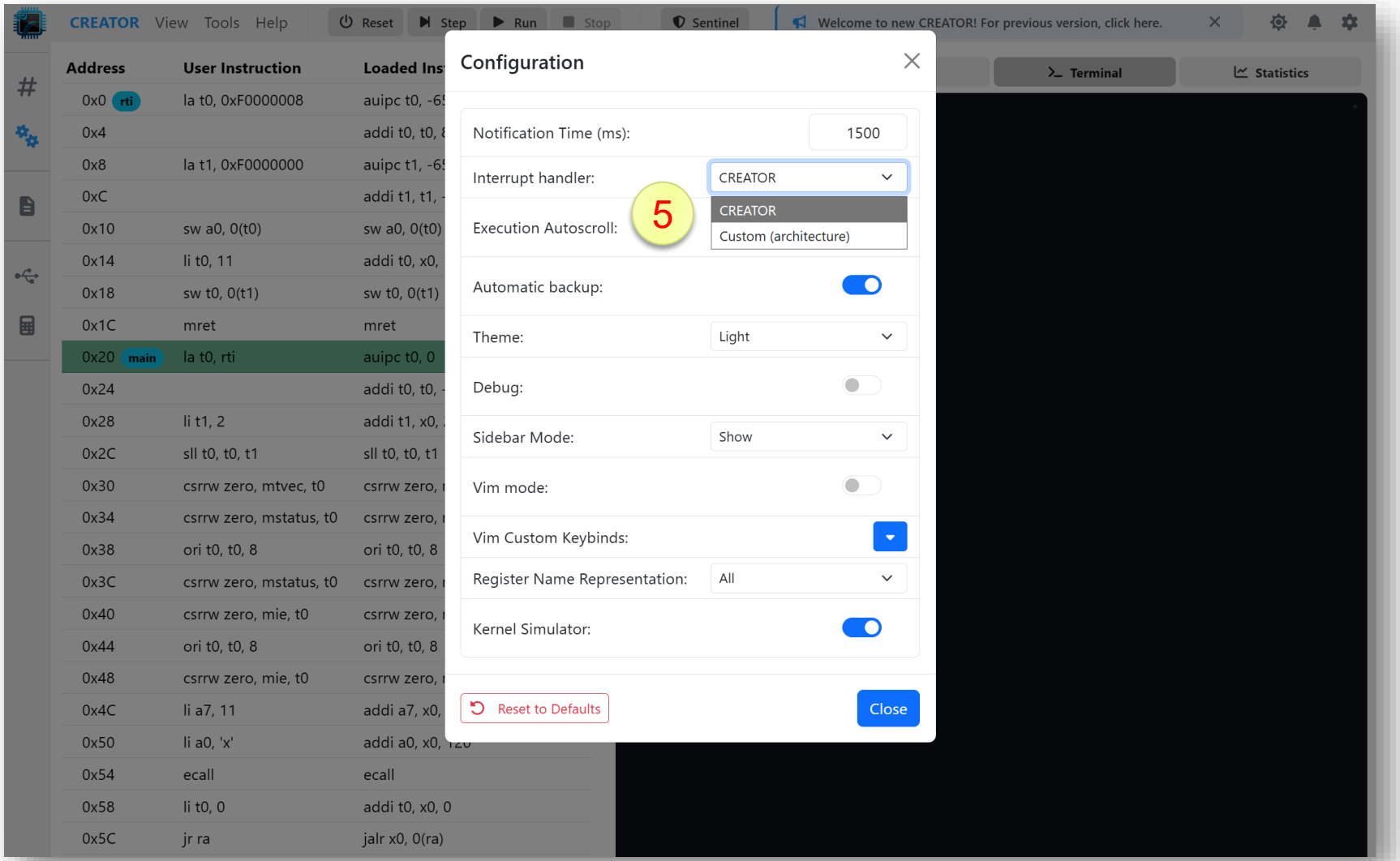
Soporte para llamadas: ejemplo

Ejemplo



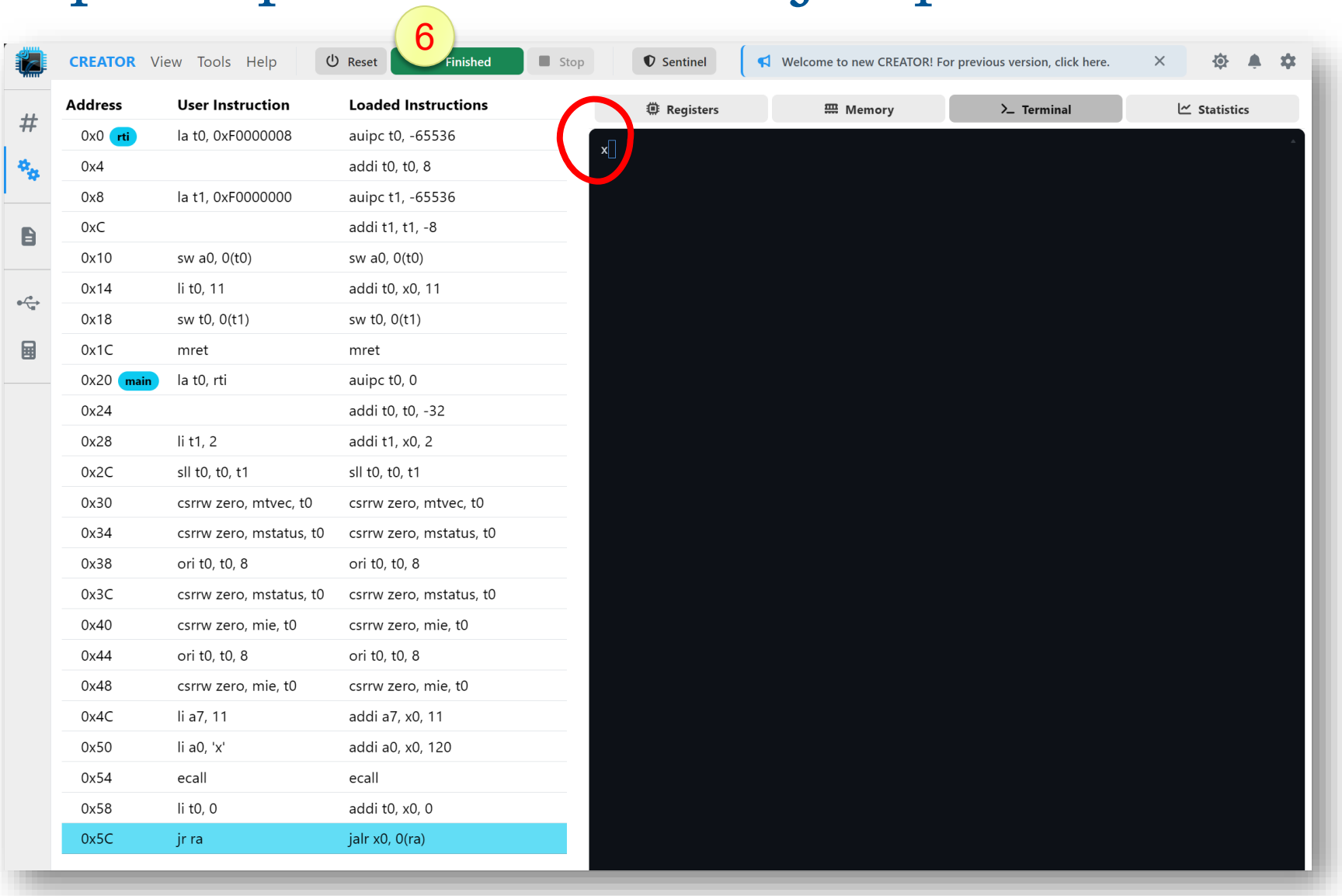
Soporte para llamadas: ejemplo

Ejemplo



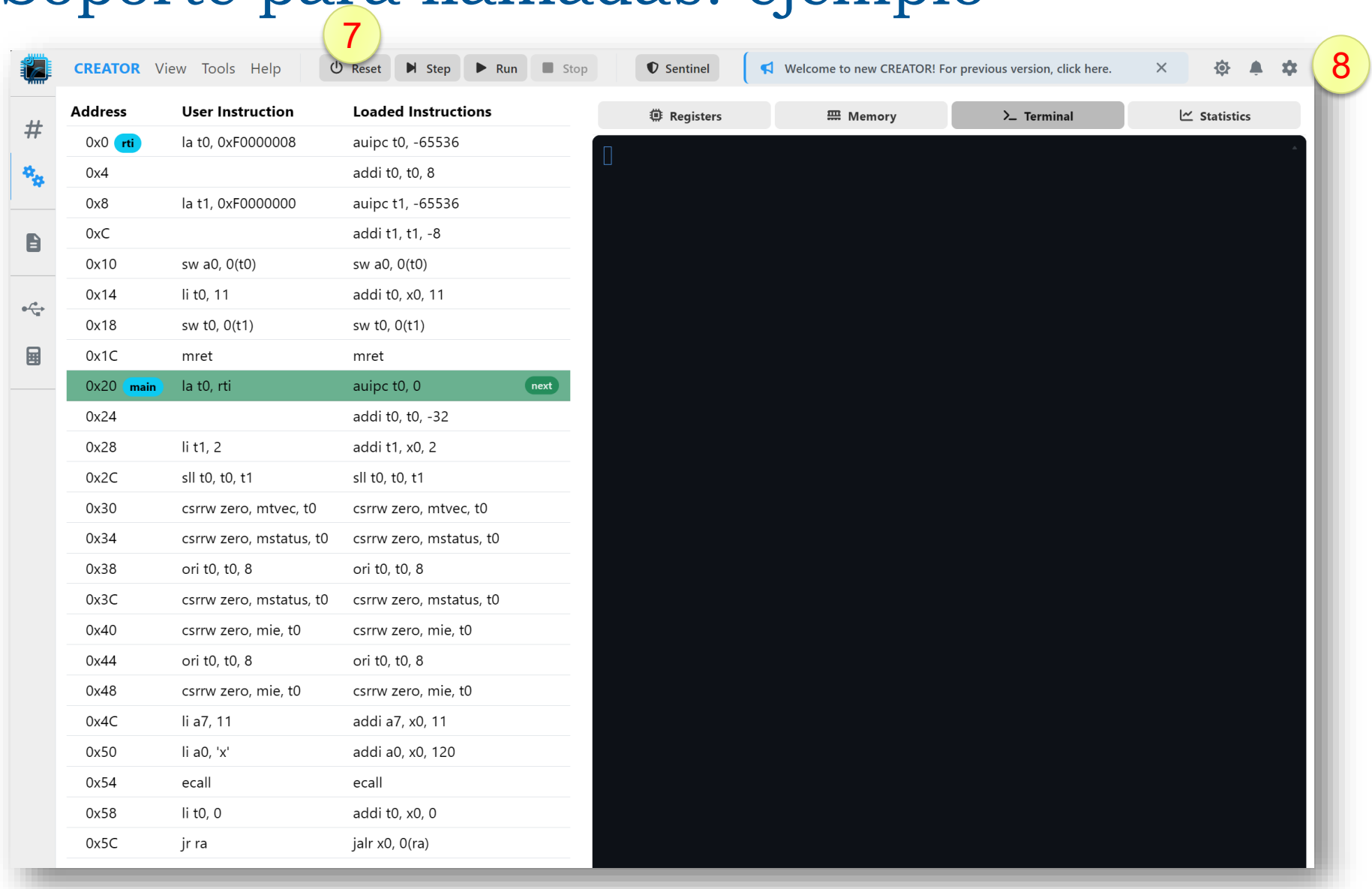
Soporte para llamadas: ejemplo

Ejemplo



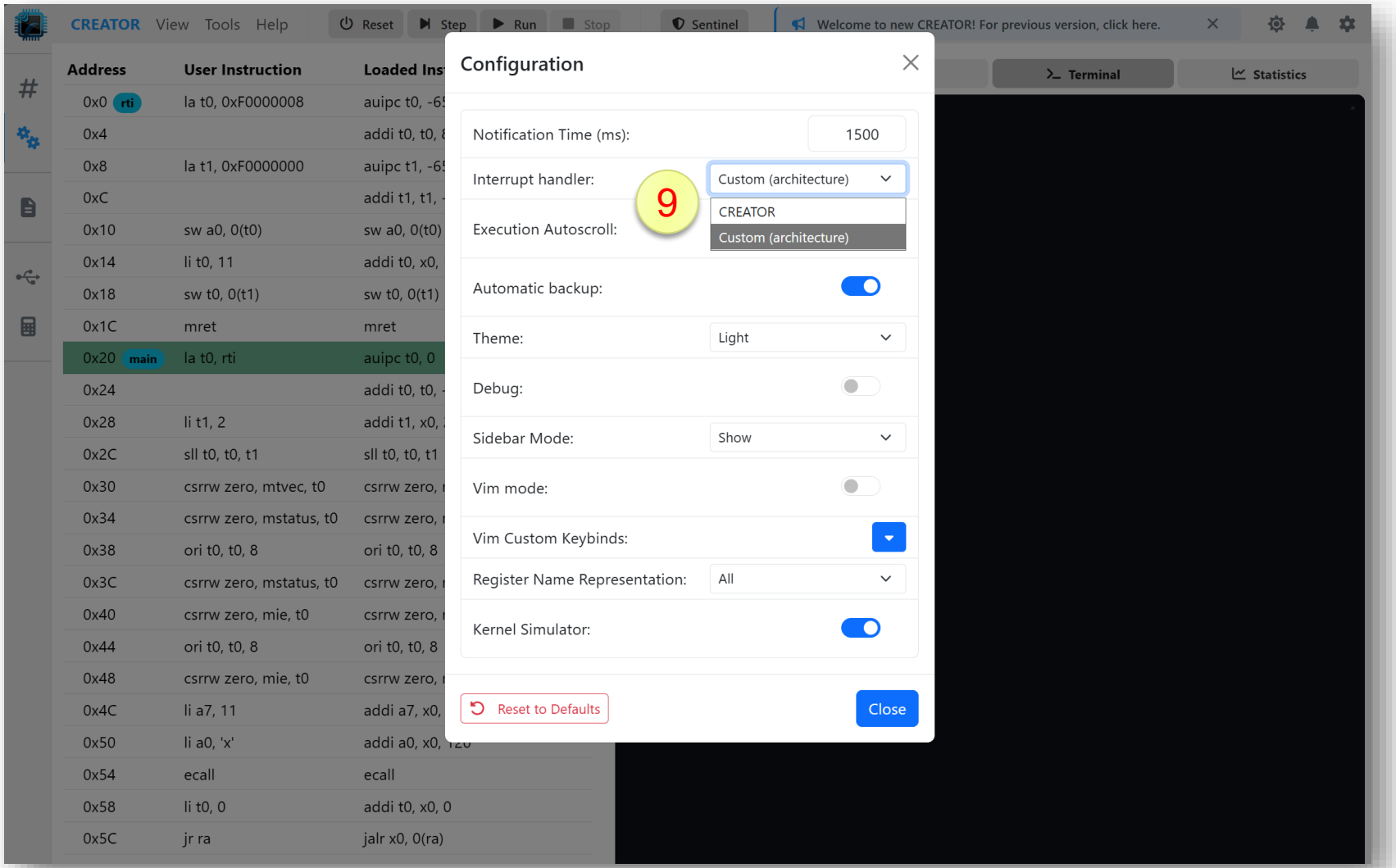
Soporte para llamadas: ejemplo

Ejemplo



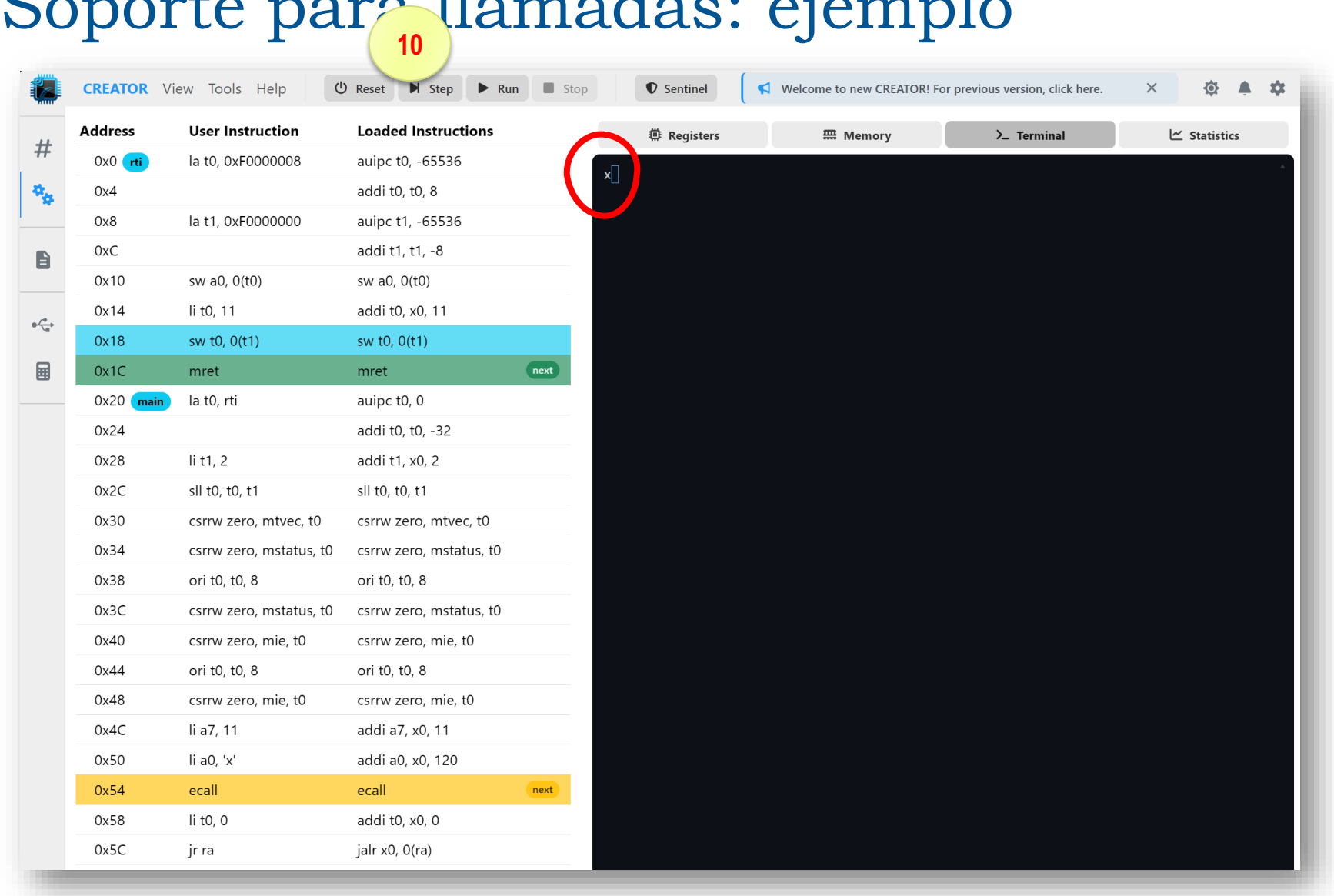
Soporte para llamadas: ejemplo

Ejemplo



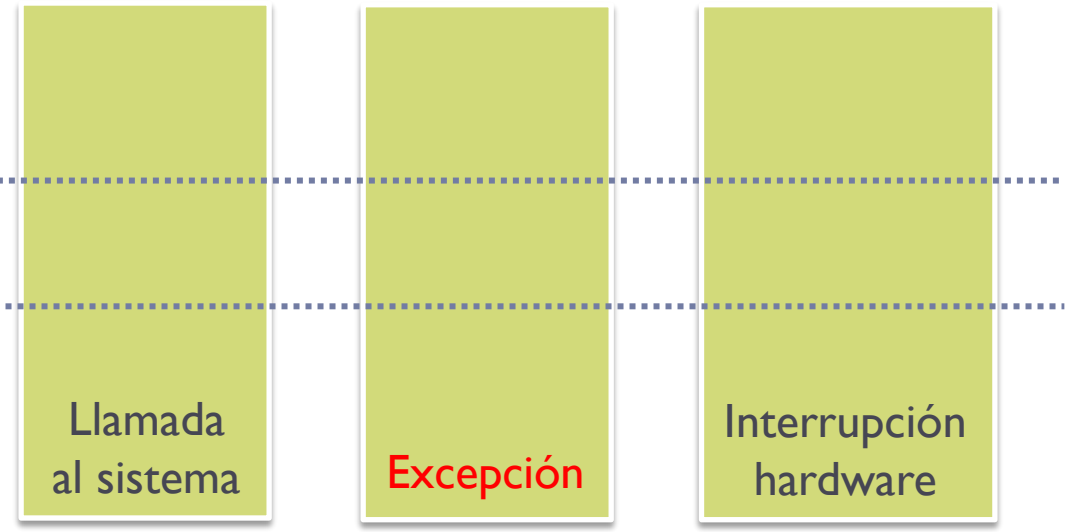
Soporte para llamadas: ejemplo

Ejemplo



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*)



https://creatorsim.github.io/creator-wiki/development/privileged.html

Soporte para excepciones

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

Edit Architecture

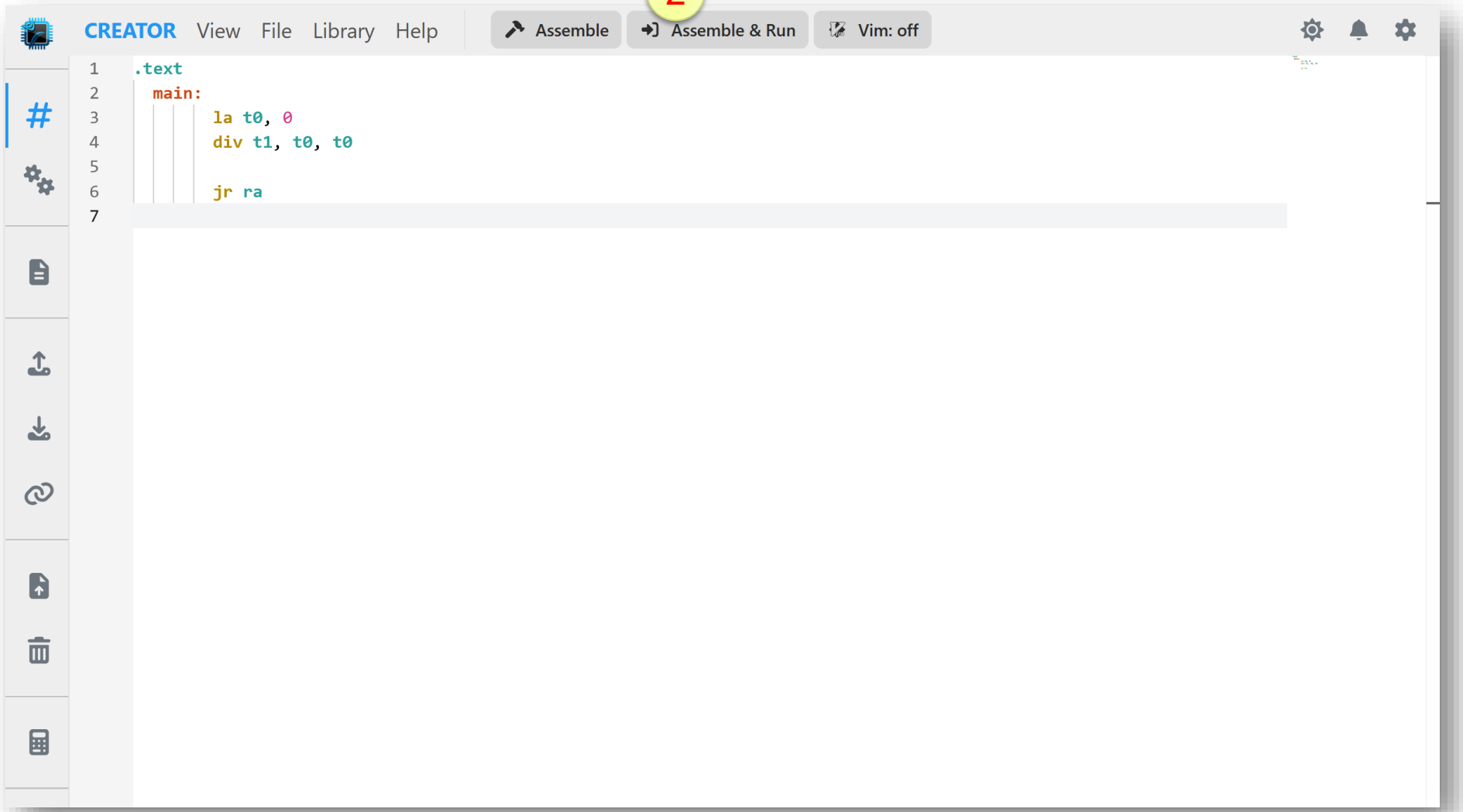
```
2084
2085  # R-type instructions
2086  - name: div
2087    template: R
2088    fields:
2089      - field: opcode
2090        value: "0110011"
2091      - field: funct7
2092        value: "0000001"
2093      - field: funct3
2094        value: "100"
2095    definition: |
2096      if (registers[rs2] != 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

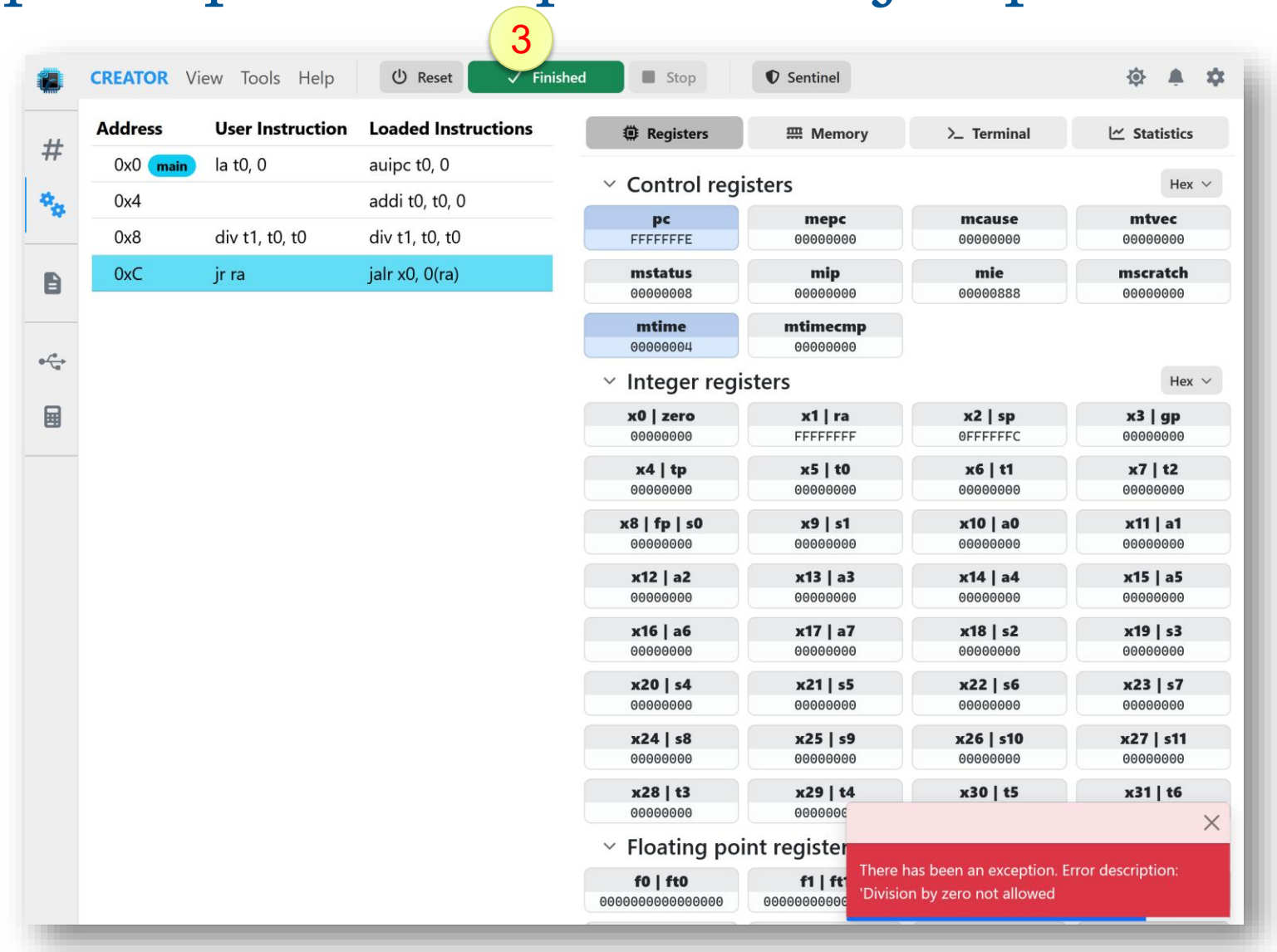
1

2



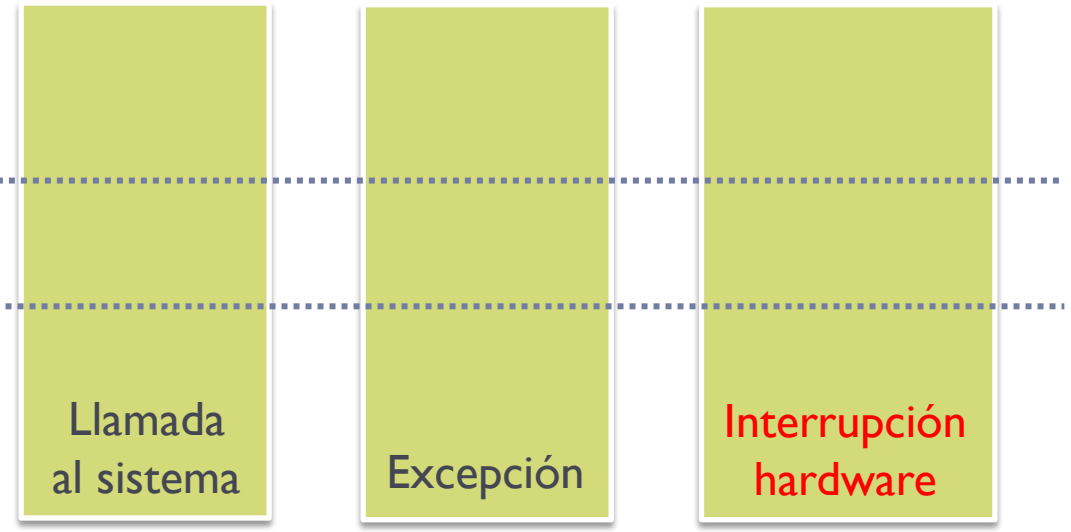
Soporte para excepciones: ejemplo

Ejemplo



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*)



https://creatorsim.github.io/creator-wiki/development/privileged.html

Soporte para interrupciones

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

Edit Architecture

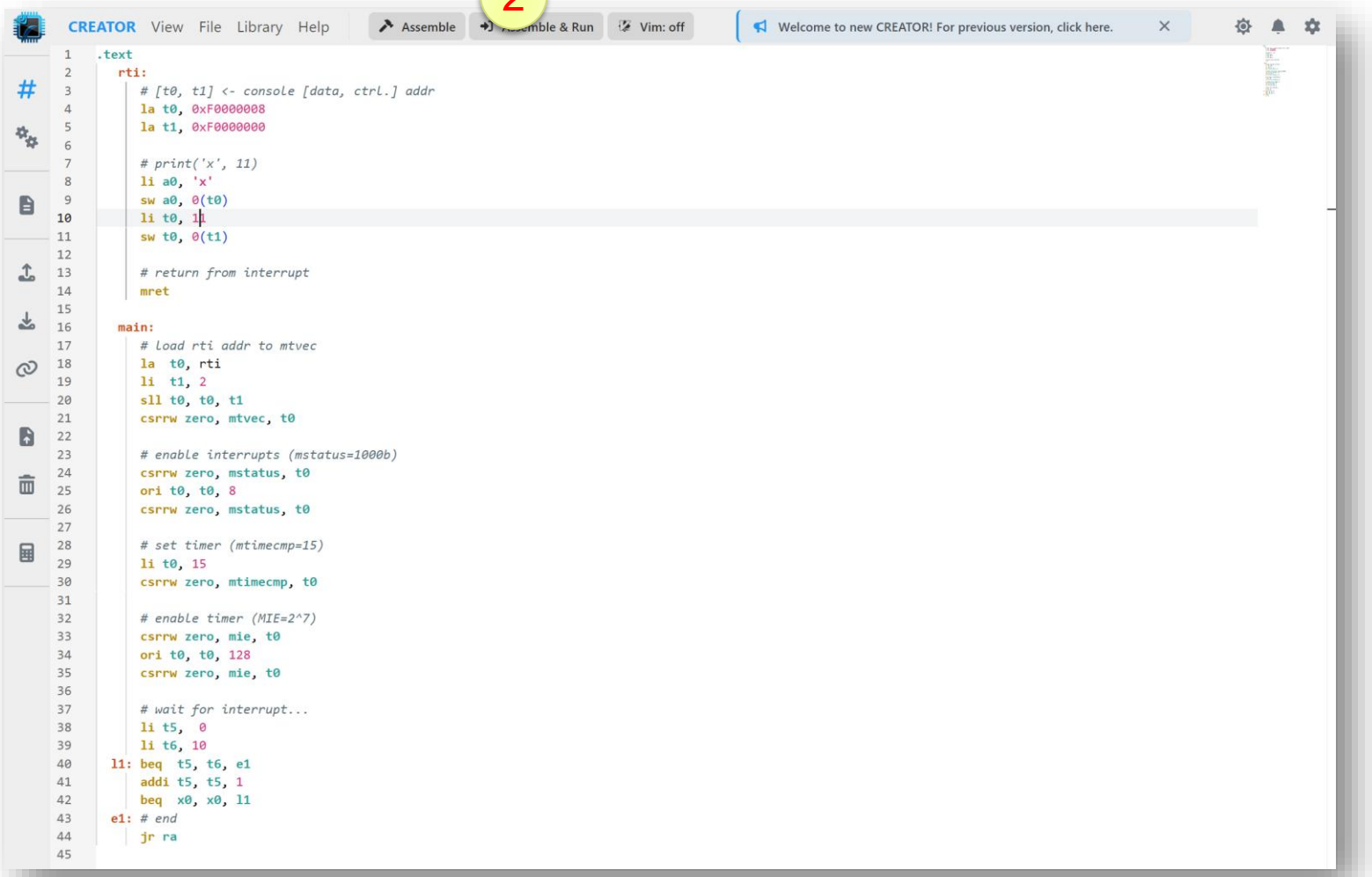
```
4157
4158  timer:
4159    tick_cycles: 1
4160    advance: |
4161      registers.mtime = (registers.mtime + 1n) % (2n**32n - 1n);
4162    handler: |
4163      if (registers.mtime === registers.mtimecmp) {
4164        CAPI.INTERRUPTS.create(InterruptType.Timer);
4165      }
4166    is_enabled: |
4167      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

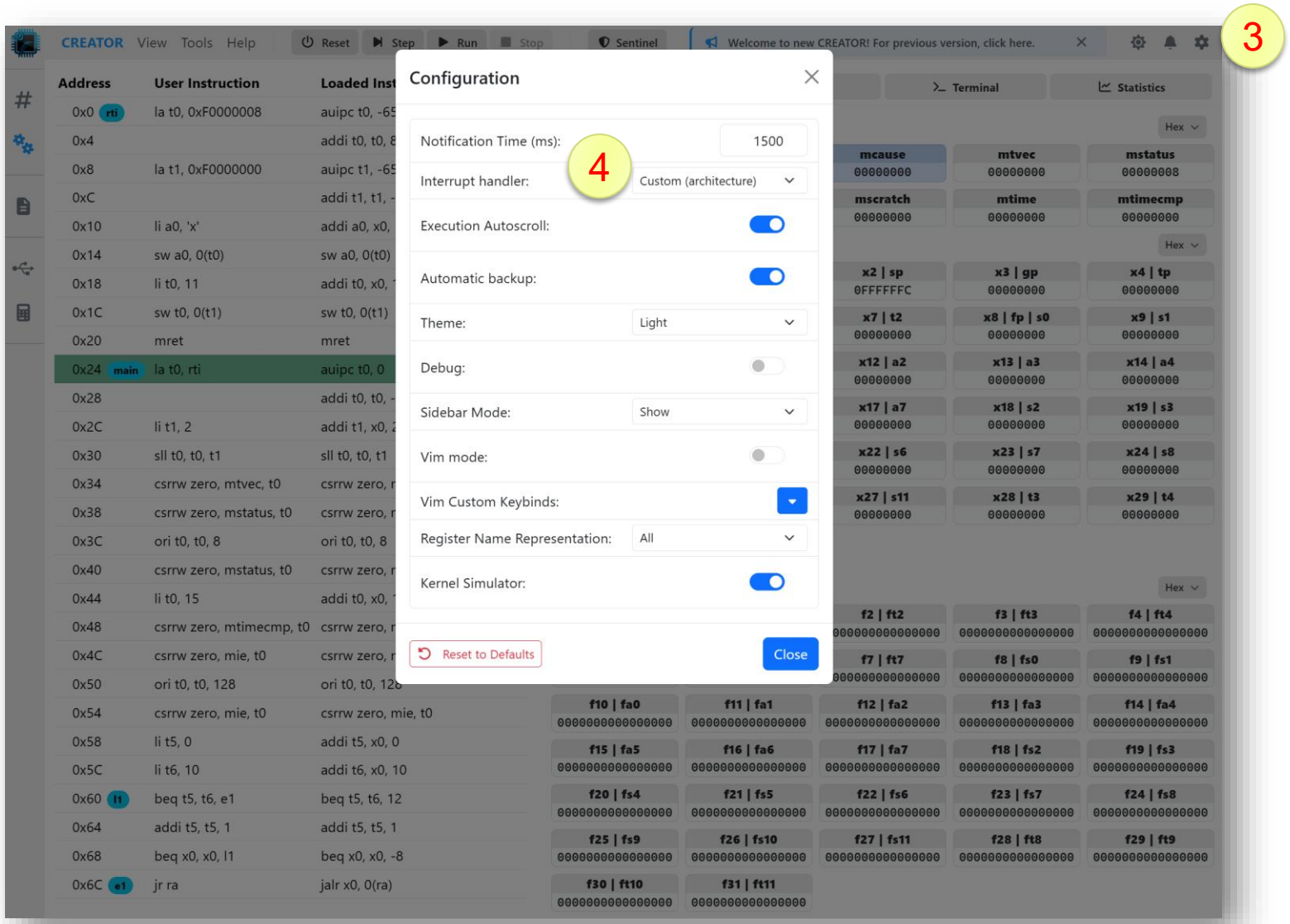
1

2



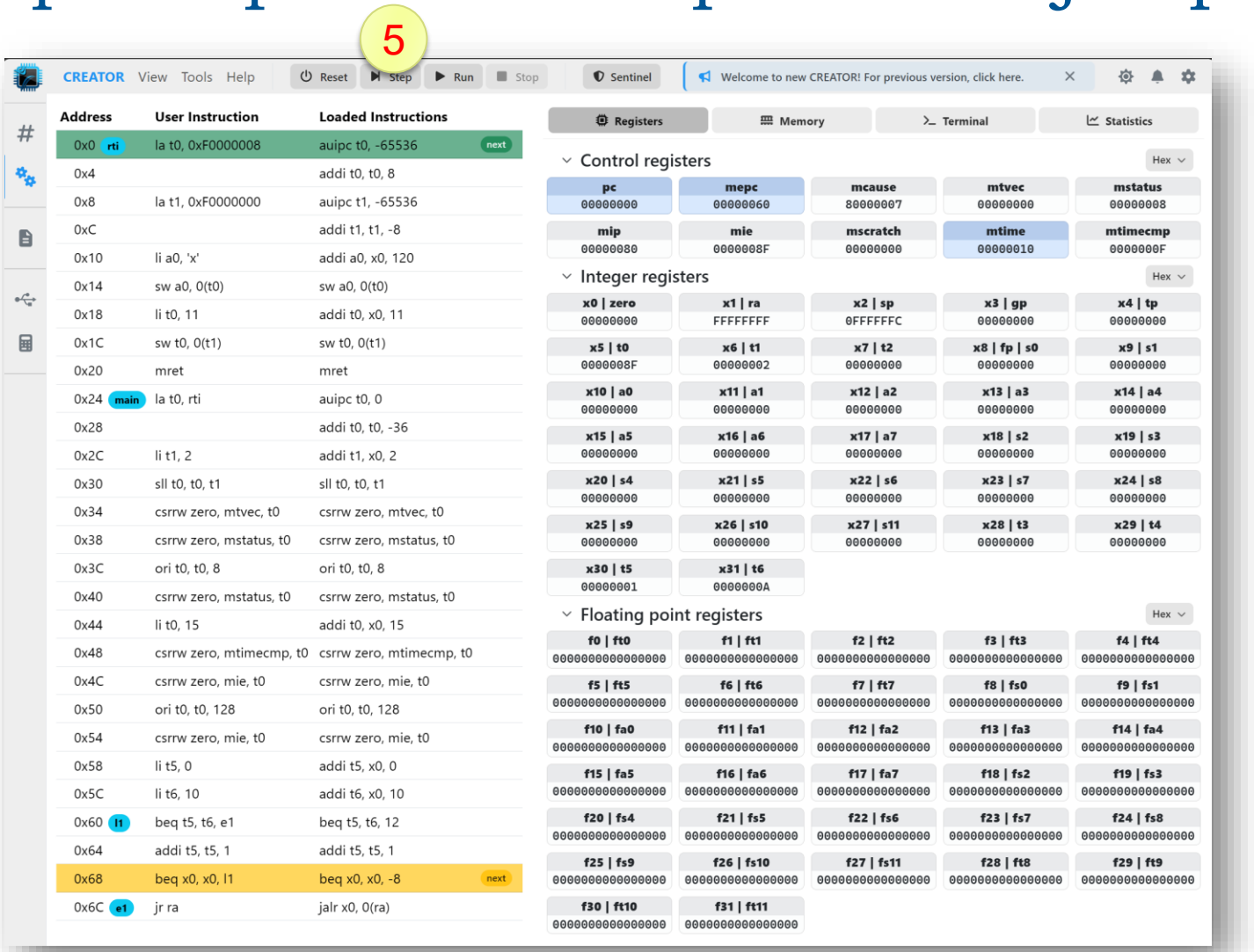
Soporte para interrupciones: ejemplo

Ejemplo



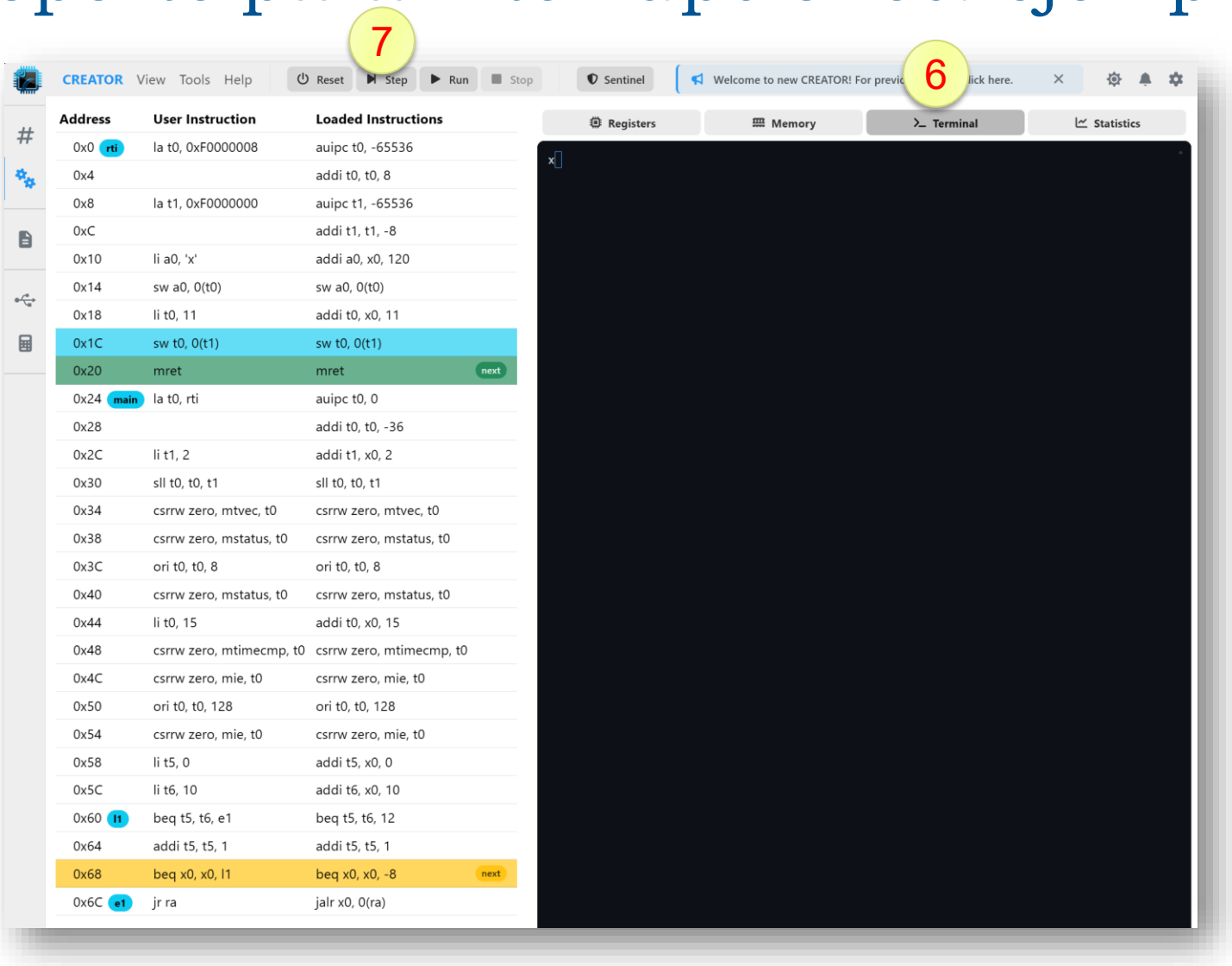
Soporte para interrupciones: ejemplo

Ejemplo



Soporte para interrupciones: ejemplo

Ejemplo



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
```

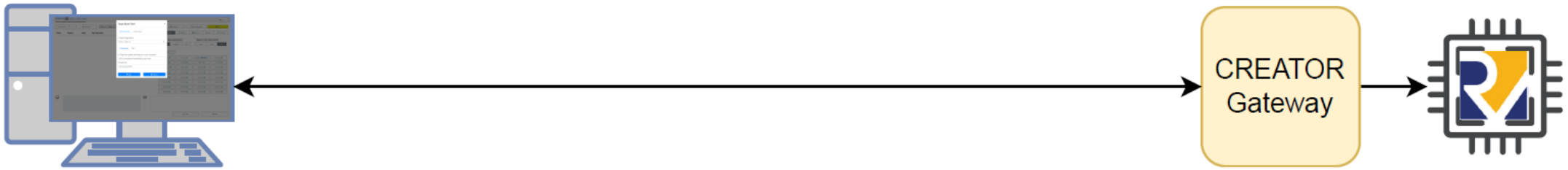
- De un "disparo" (*one-shot*)
- Si se precisa lanzar interrupción de forma periódica entonces al final del tratamiento de interrupción hay que programar cuándo será la siguiente.
- Incrementar registro `mtimecmp` al número de ciclos futuro.

Bloque 2

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

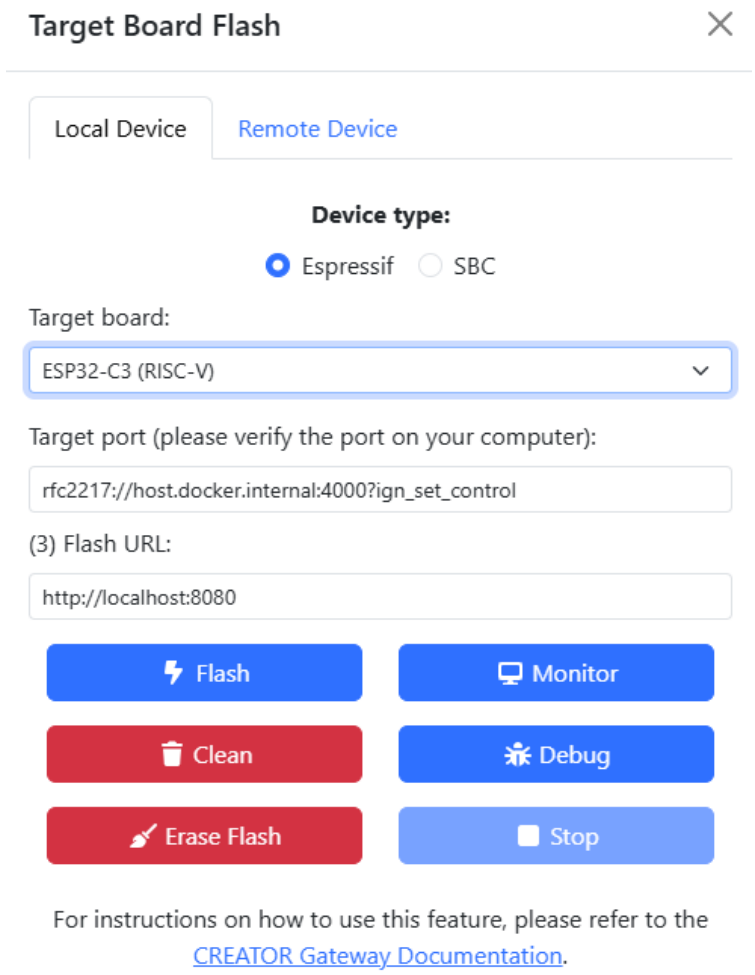
Integración de CREATOR con hardware real

- 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



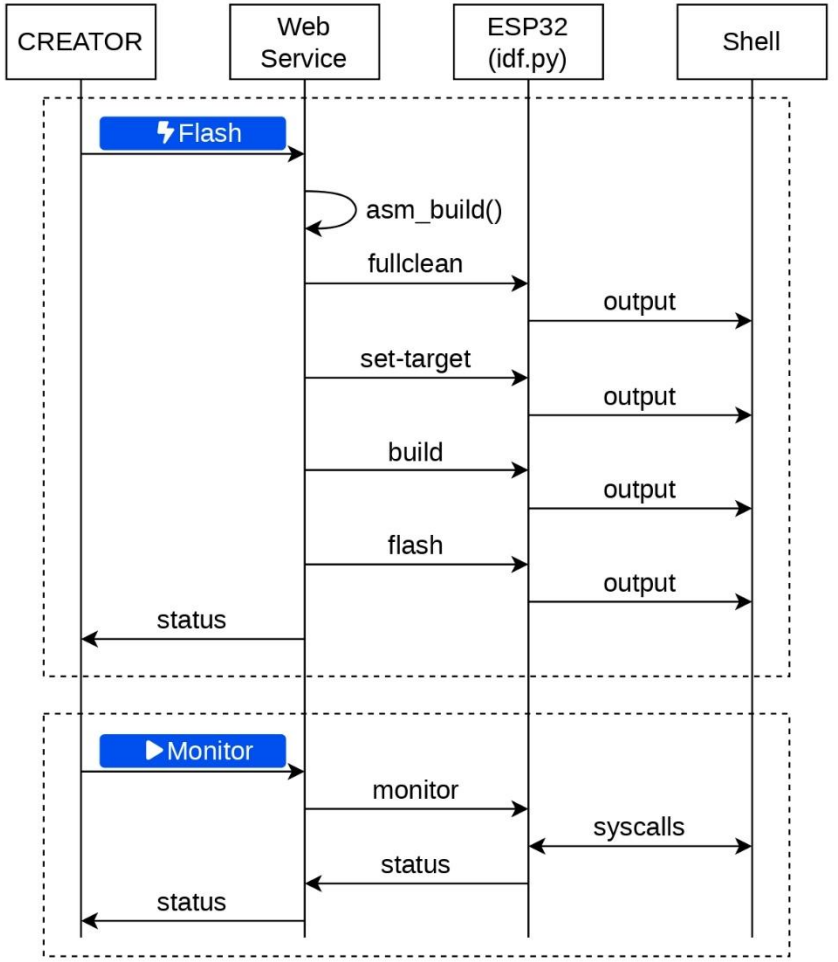
Integración de CREATOR con hardware real

- 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



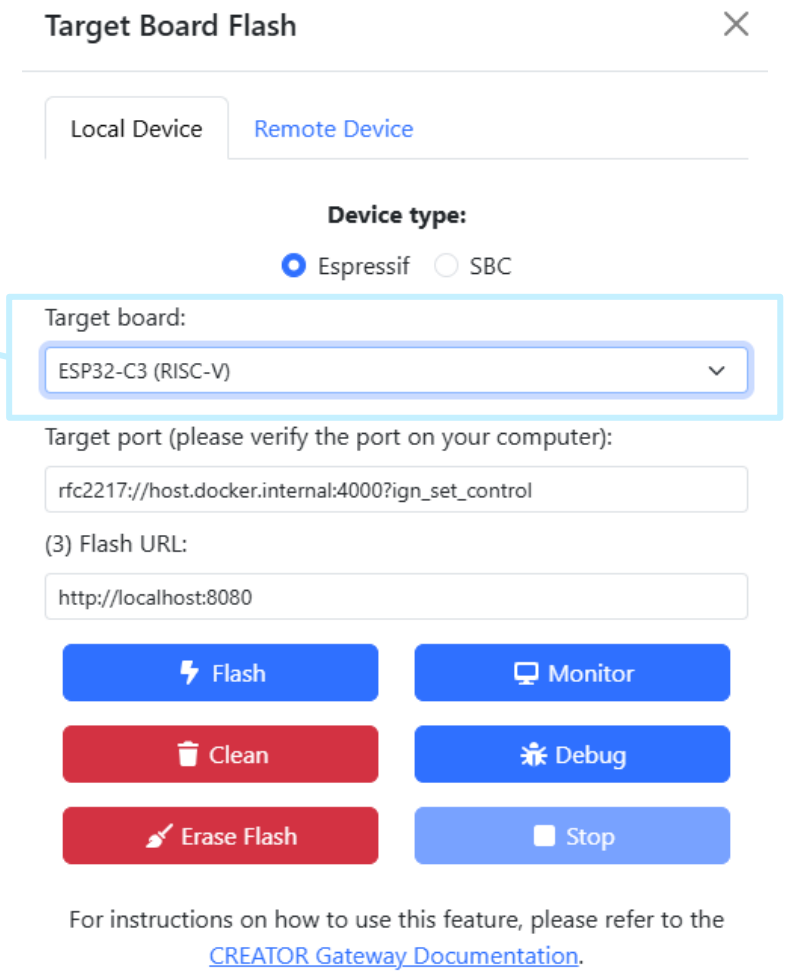
Integración de CREATOR con hardware real

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



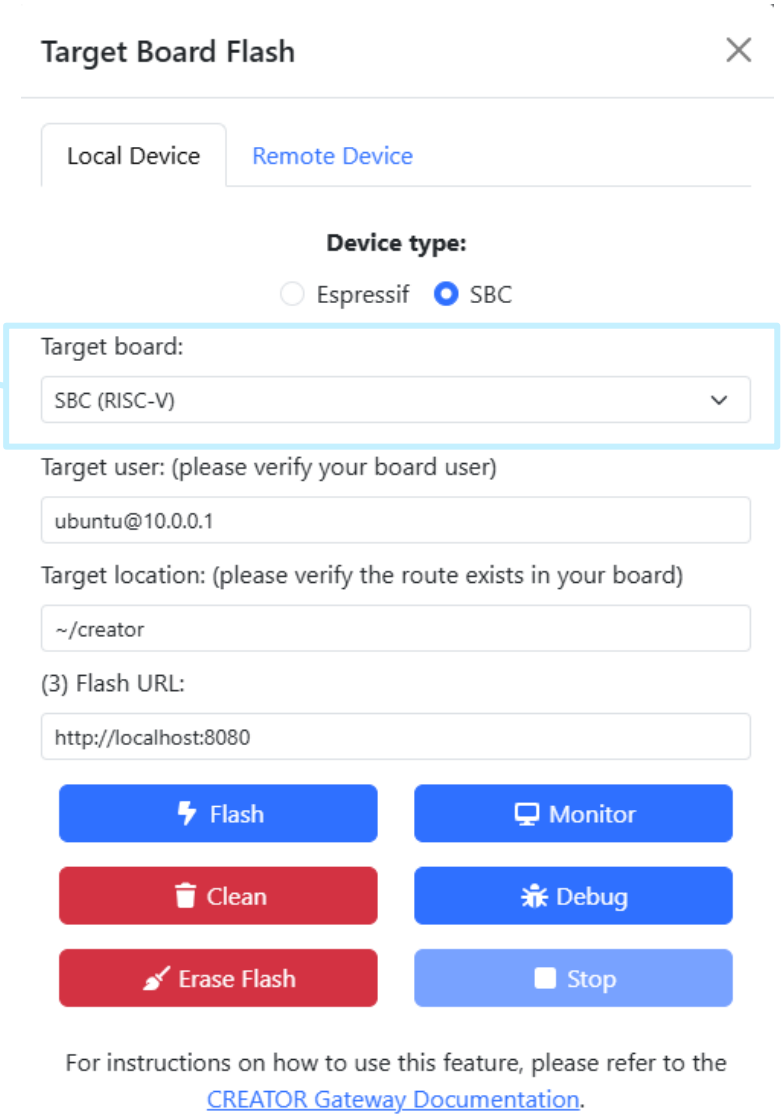
Microcontroladores

- 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



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



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
```

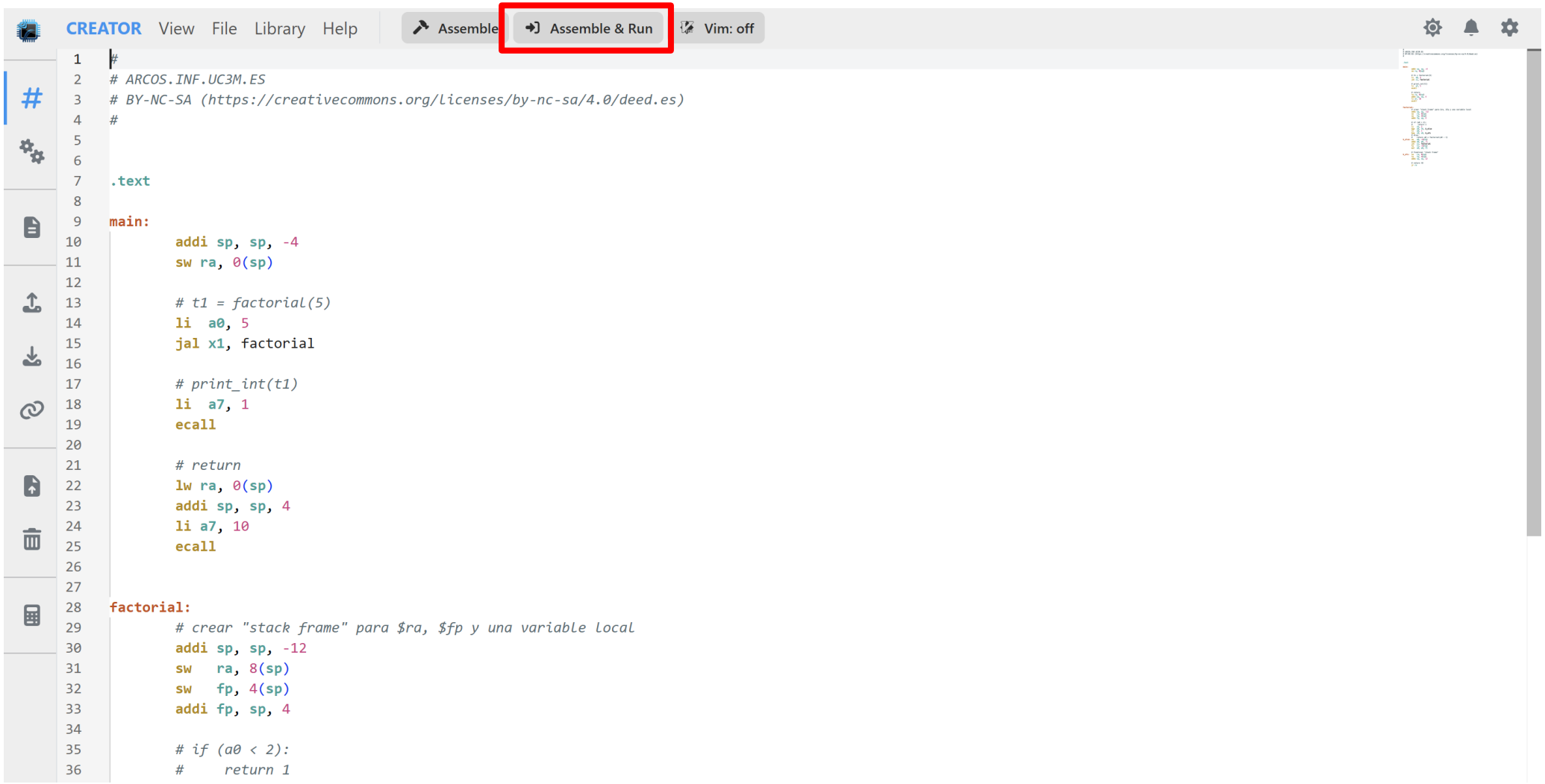
Ejercicio guiado – Microcontrolador ESP32
Despliegue Windows

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
```

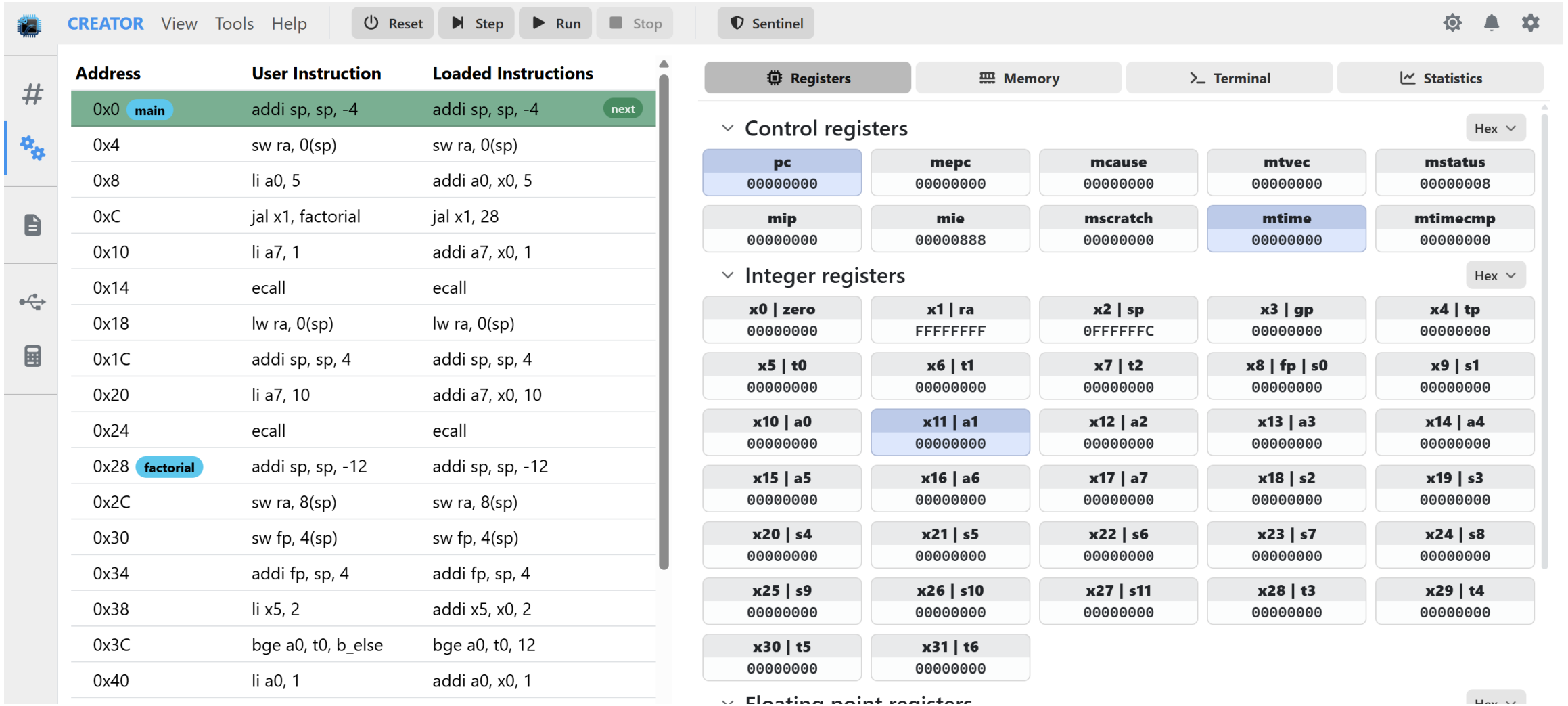
Ejercicio guiado – Microcontrolador ESP32

Programa




```
#
# ARCOS.INF.UC3M.ES
# BY-NC-SA (https://creativecommons.org/licenses/by-nc-sa/4.0/deed.es)
#


.text

main:
        addi sp, sp, -4
        sw ra, 0(sp)

        # t1 = factorial(5)
        li  a0, 5
        jal x1, factorial

        # print_int(t1)
        li  a7, 1
        ecall

        # return
        lw ra, 0(sp)
        addi sp, sp, 4
        li a7, 10
        ecall


factorial:
        # crear "stack frame" para $ra, $fp y una variable local
        addi sp, sp, -12
        sw   ra, 8(sp)
        sw   fp, 4(sp)
        addi fp, sp, 4

        # if (a0 < 2):
        #     return 1
```

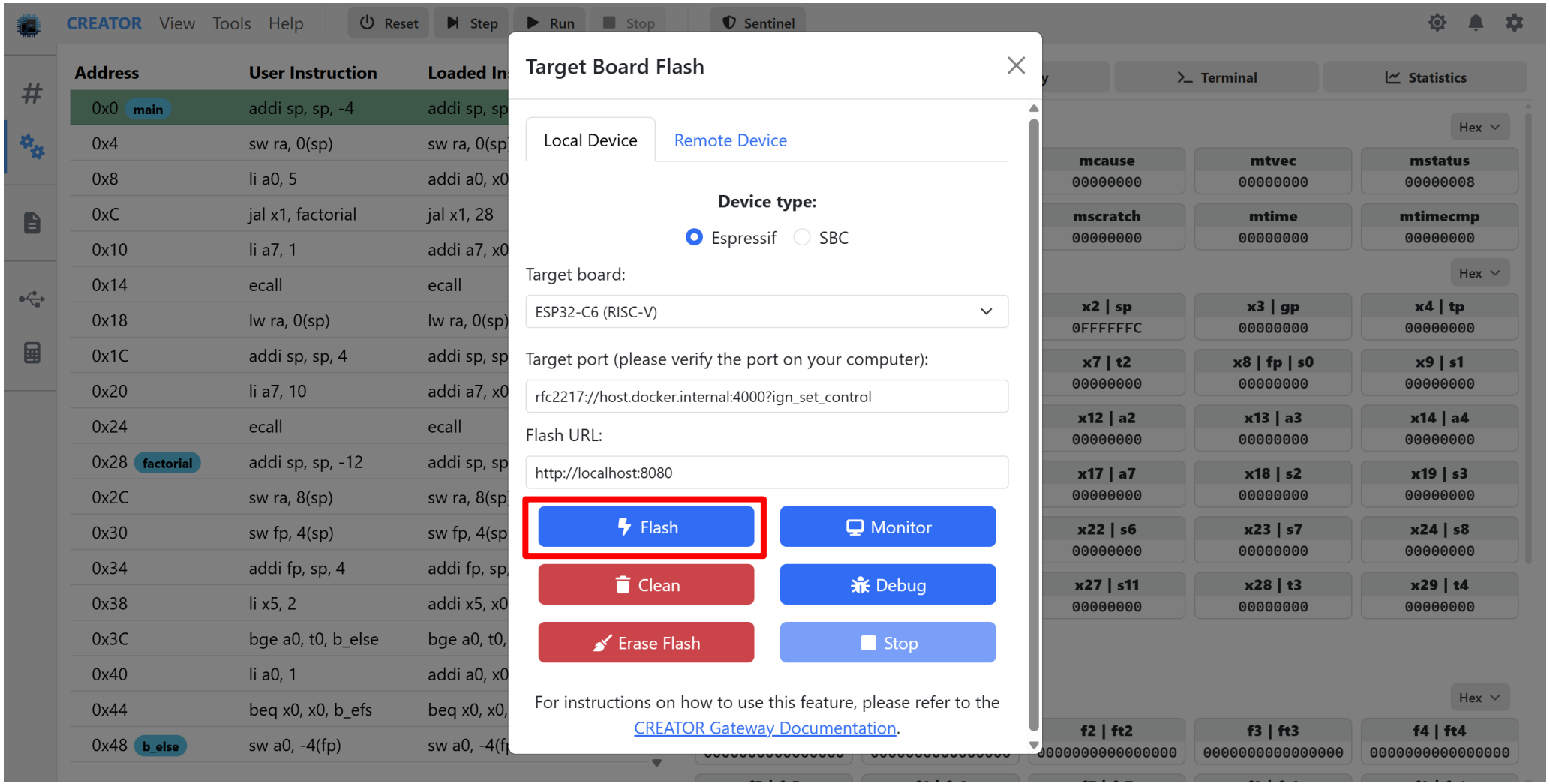
Ejercicio guiado – Microcontrolador ESP32



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. Cumplimentar 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



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...
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...
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...
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  |
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 -
```

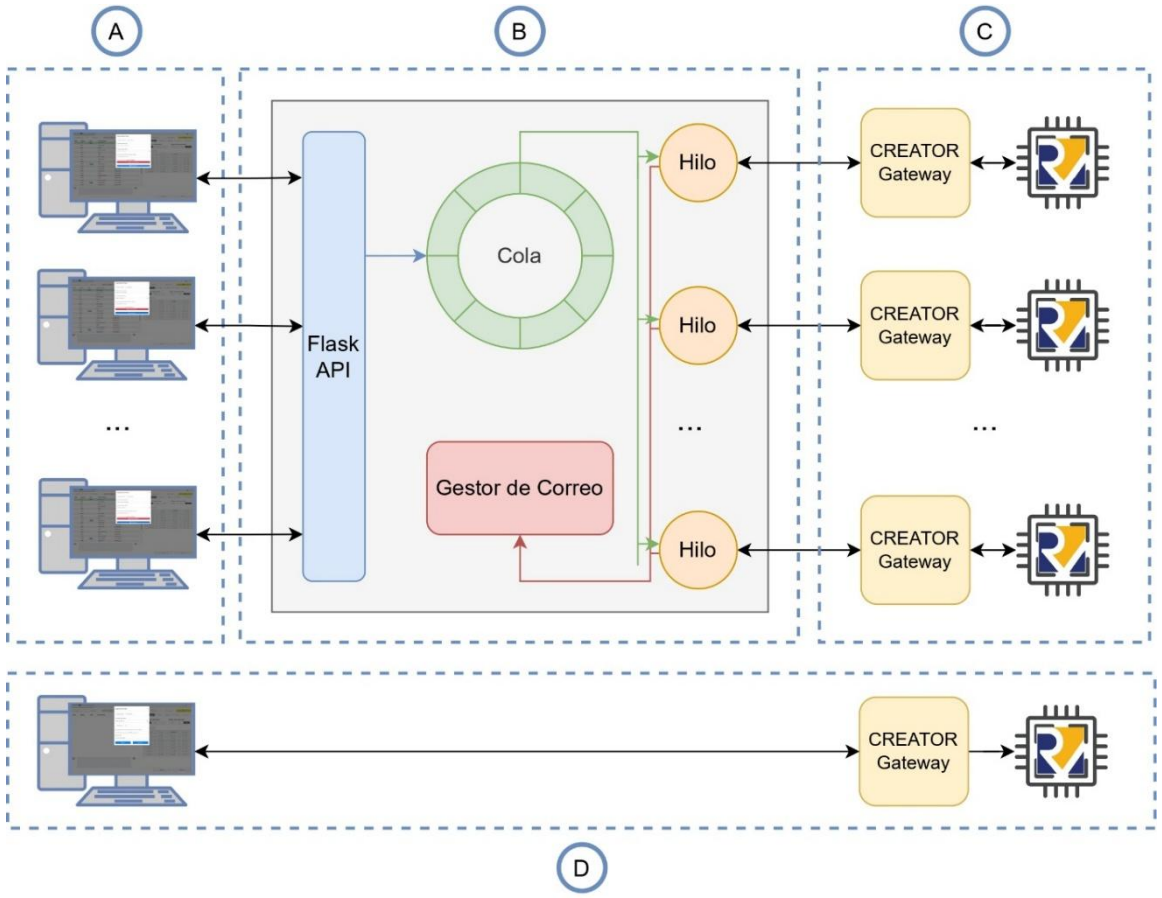
Ejercicio guiado – Microcontrolador ESP32

1. Ejecutar el programa ensamblador en el microcontrolador:
   1. 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 bin
ary 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 -
```

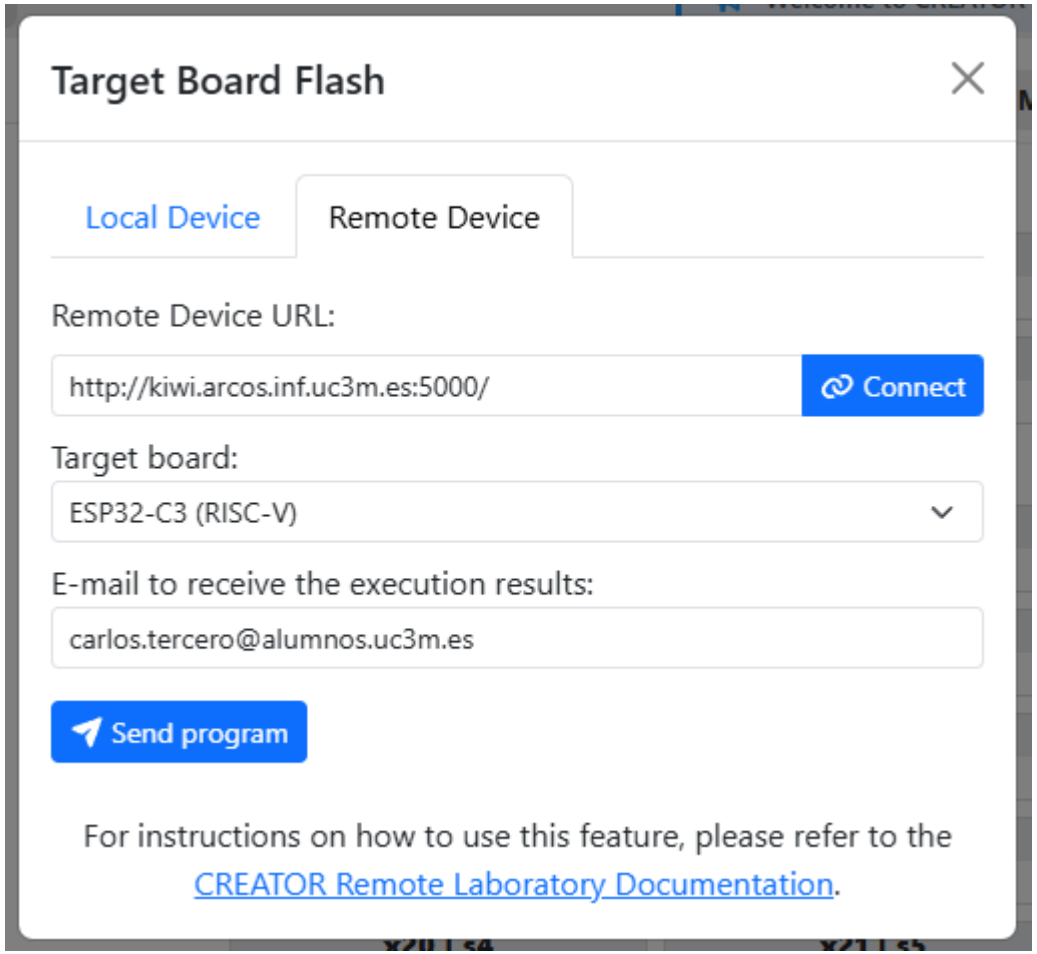
Servicio de laboratorio remoto

- *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*



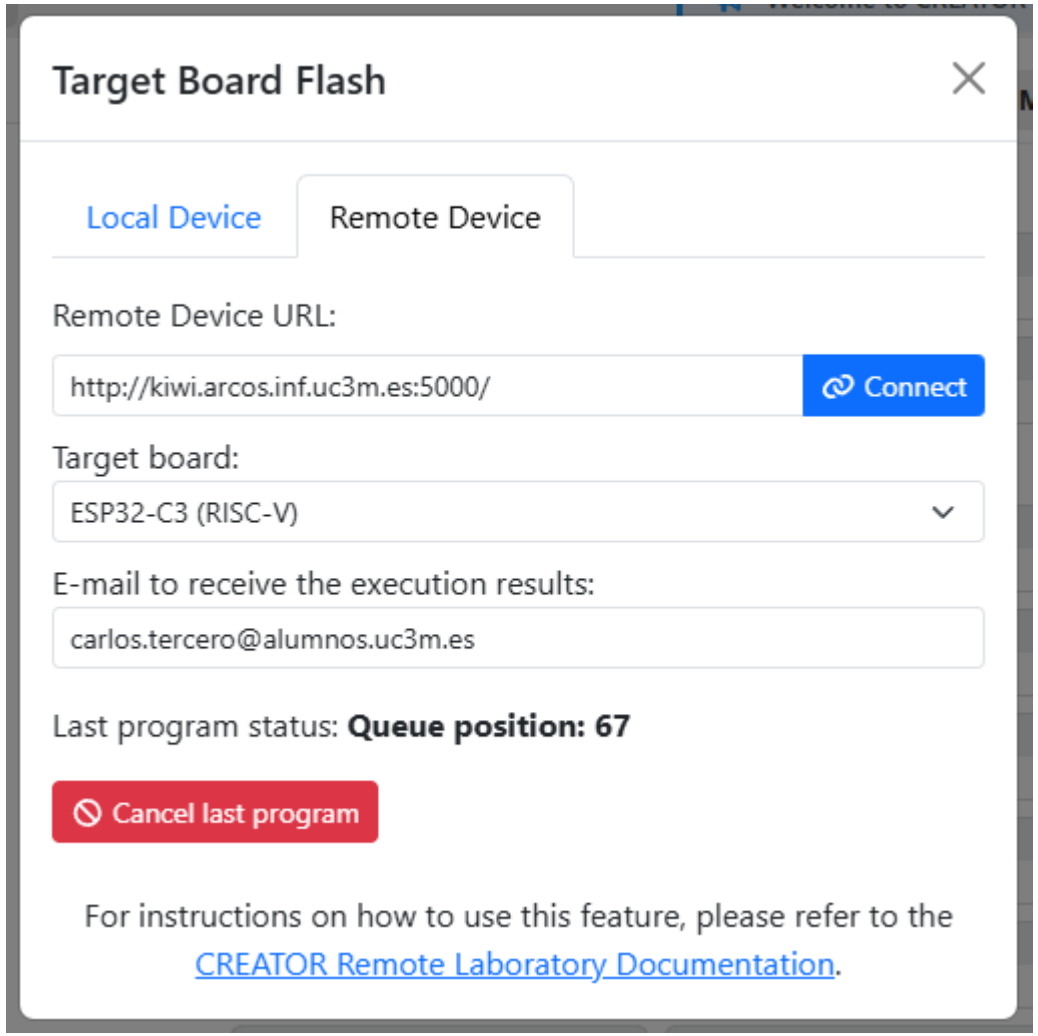
Servicio de laboratorio remoto

- *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*



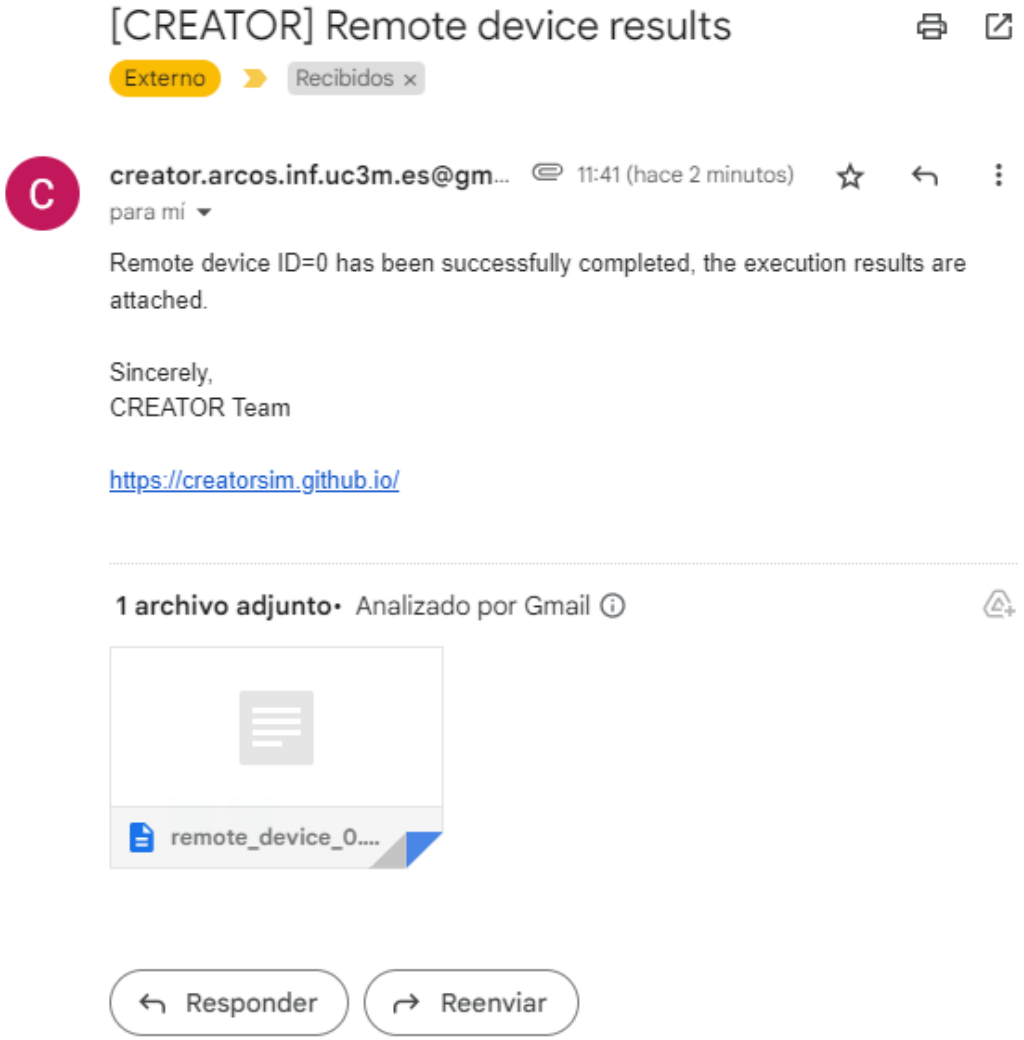
Servicio de laboratorio remoto

- *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*



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*

```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
```

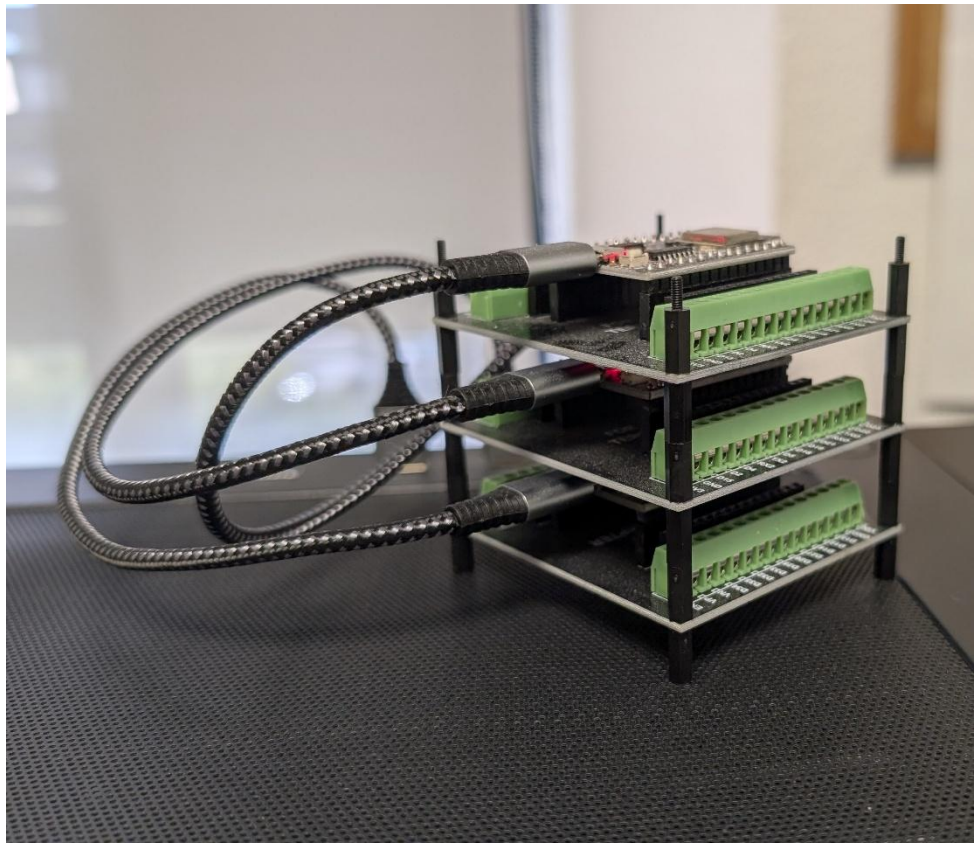
Servicio de laboratorio remoto

- 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

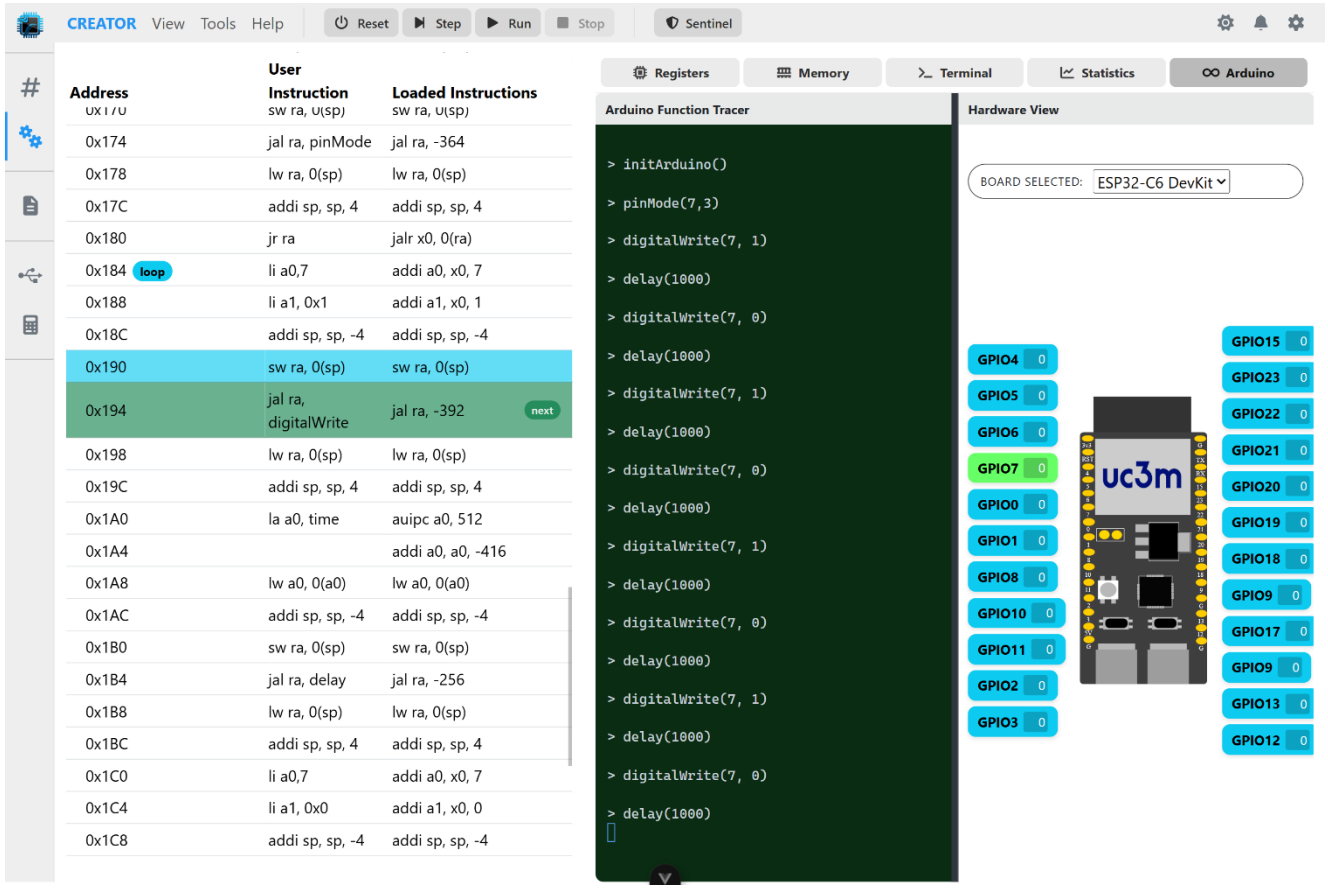


http://kiwi.arcos.inf.uc3m.es:8080/creator

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

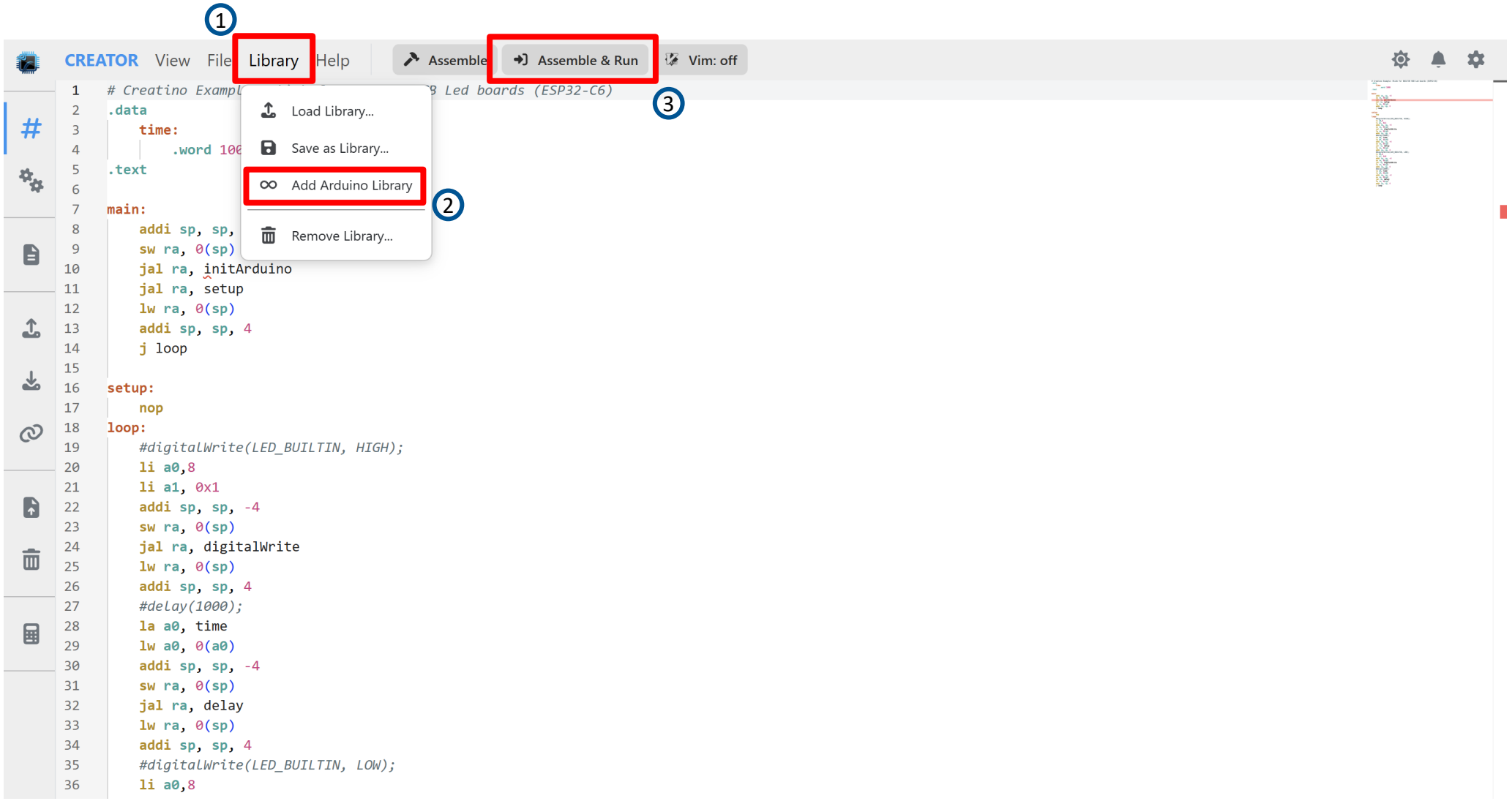
Integración con Arduino (CREATino)

- 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

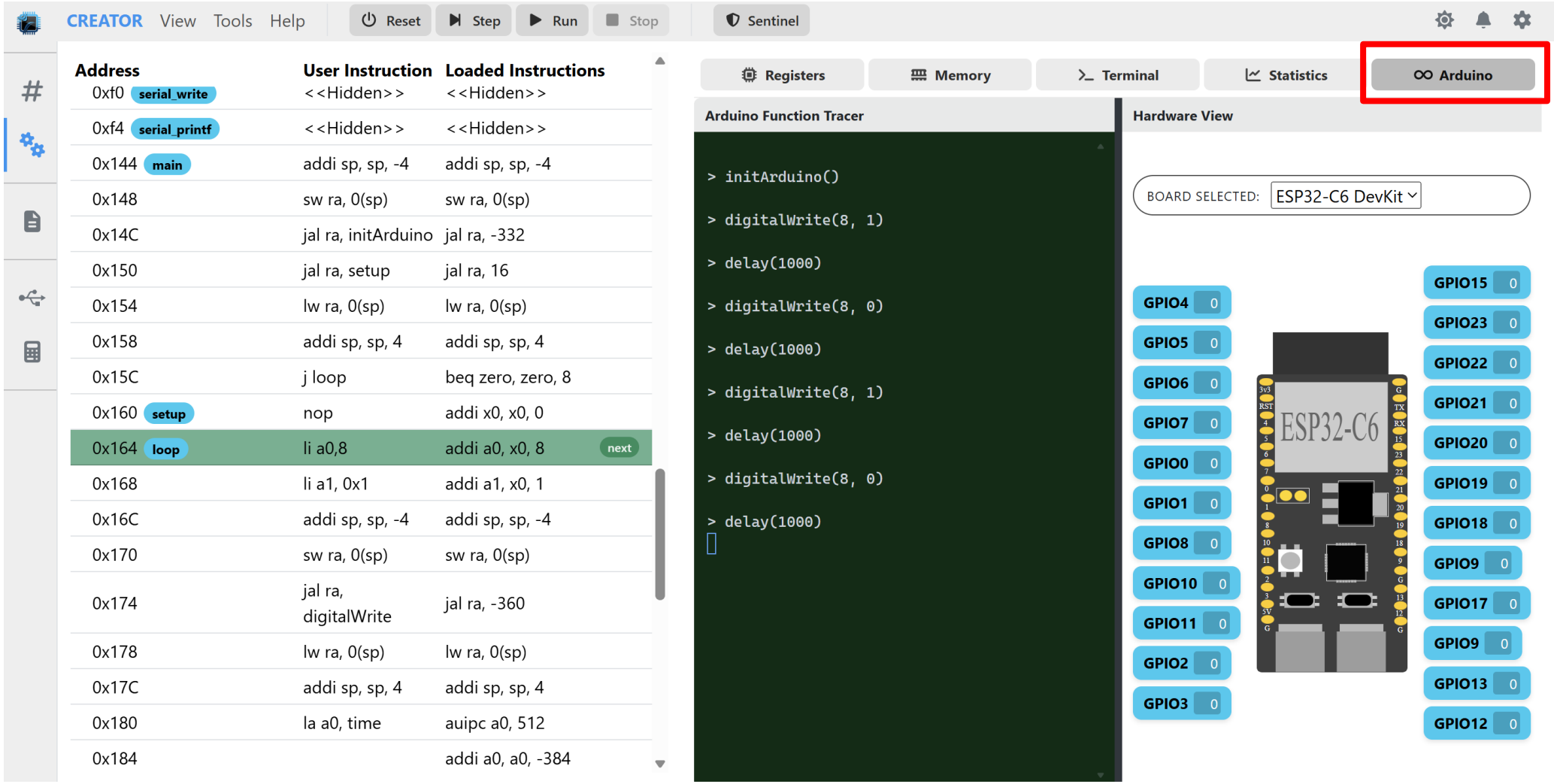


Ejercicio guiado – CREATino

Programa



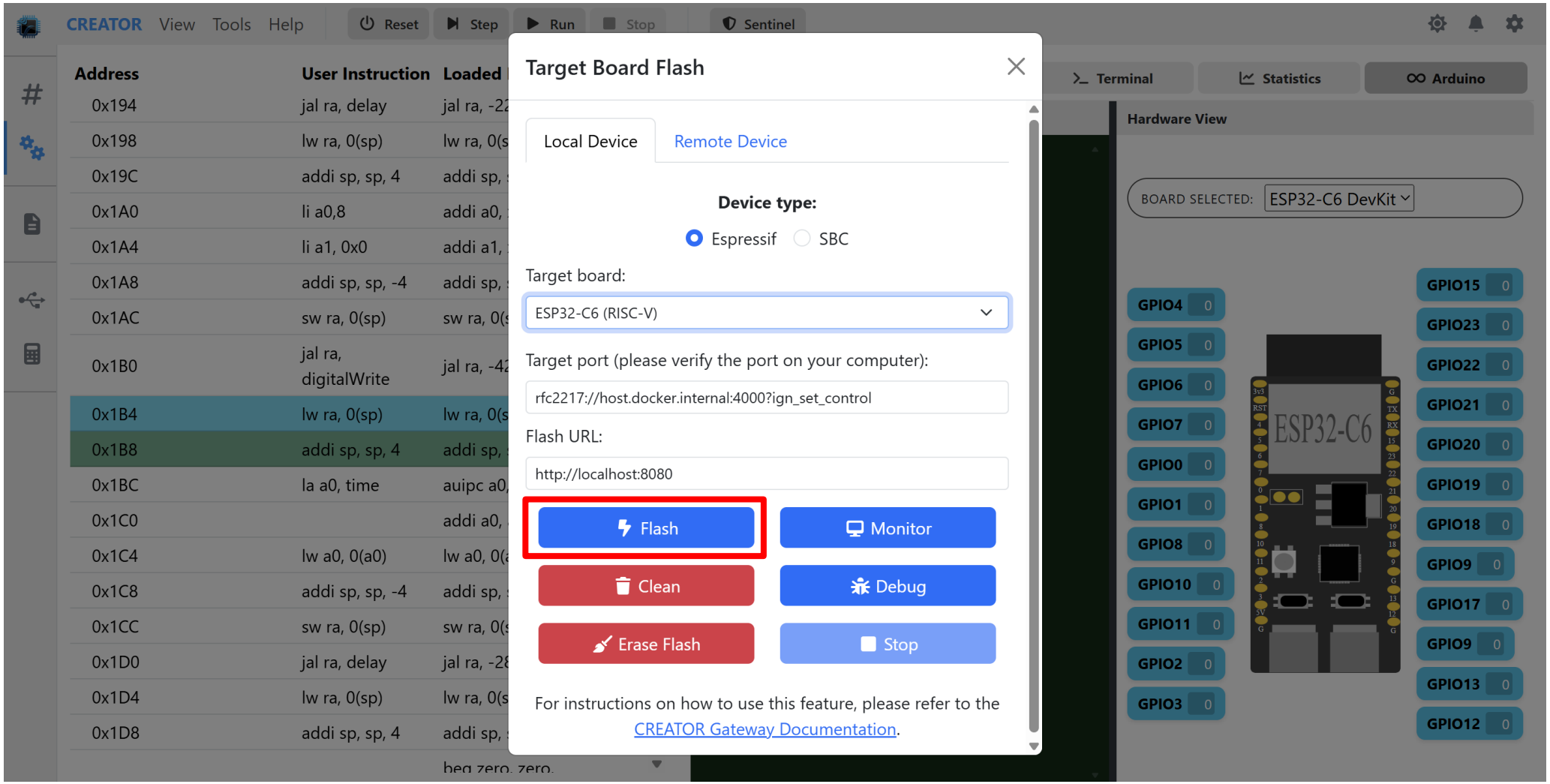
Ejercicio guiado – CREATino



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. Cumplimentar 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*

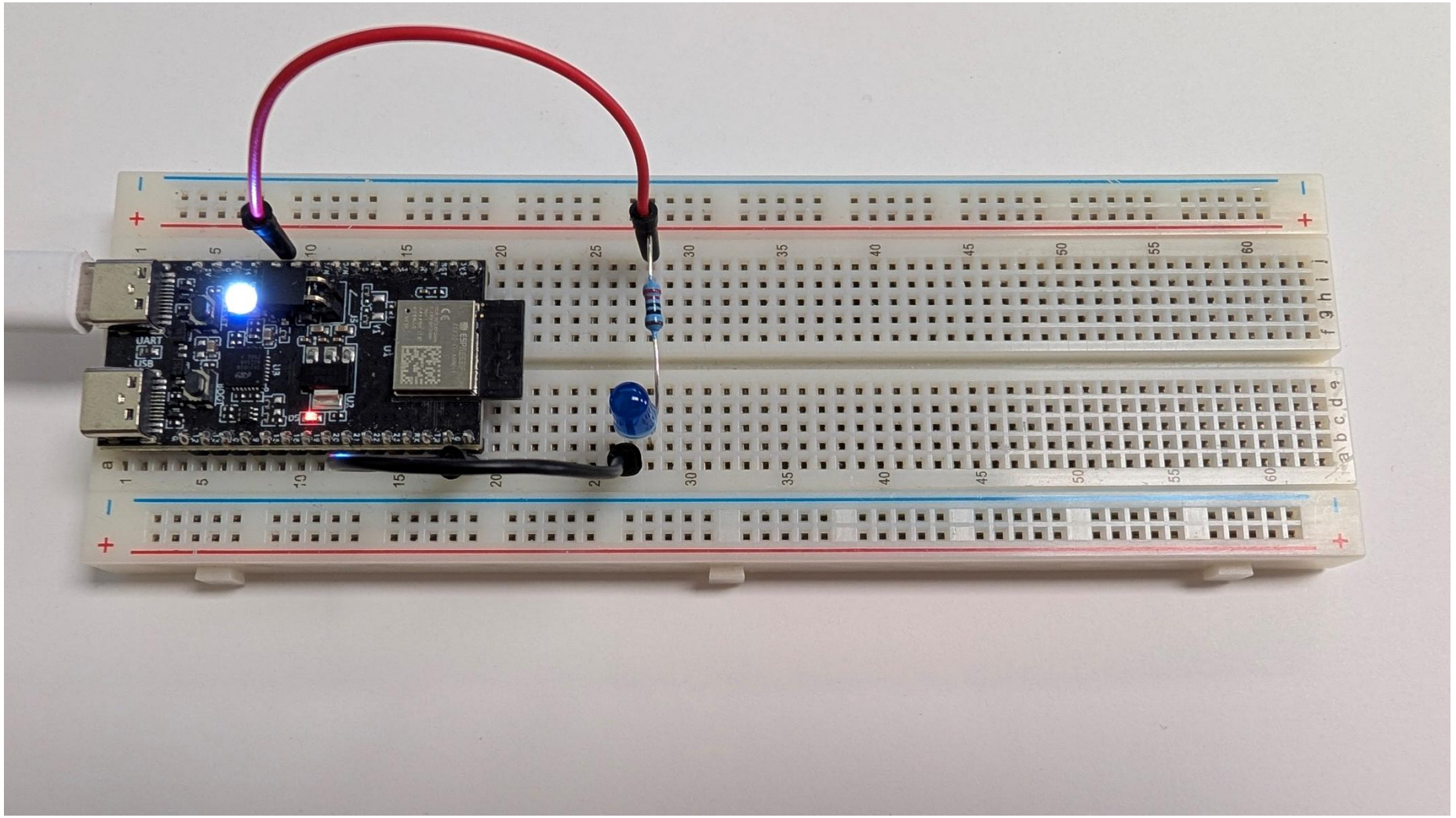
Ejercicio guiado – CREATino



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_ta
ble/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  |
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

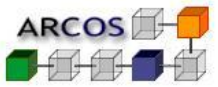
Jornadas Sarteco

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