uc3m | Universidad **Carlos III** de Madrid

Grupo ARCOS

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

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

Félix García Carballeira

felix.garcia@uc3m.es

Material

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

Motivación de Creator
didaCtic and geneRic assEmbly progrAmming simulaTOR

- Simulador didáctico para la enseñanza de la programación en ensamblador
  - Centrado en los estudiantes y profesores
- Multiplataforma
  - Ejecución en web sin servidor (sobremesa, tablets y móviles)
- Entorno integrado (edición, compilación y simulación de programas)
- Posibilidad de definir y trabajar con diferentes arquitecturas y lenguajes ensamblador
  - Características básicas (n° de bits, registros, …)
  - Instrucciones
  - Pseudoinstrucciones
  - Directivas

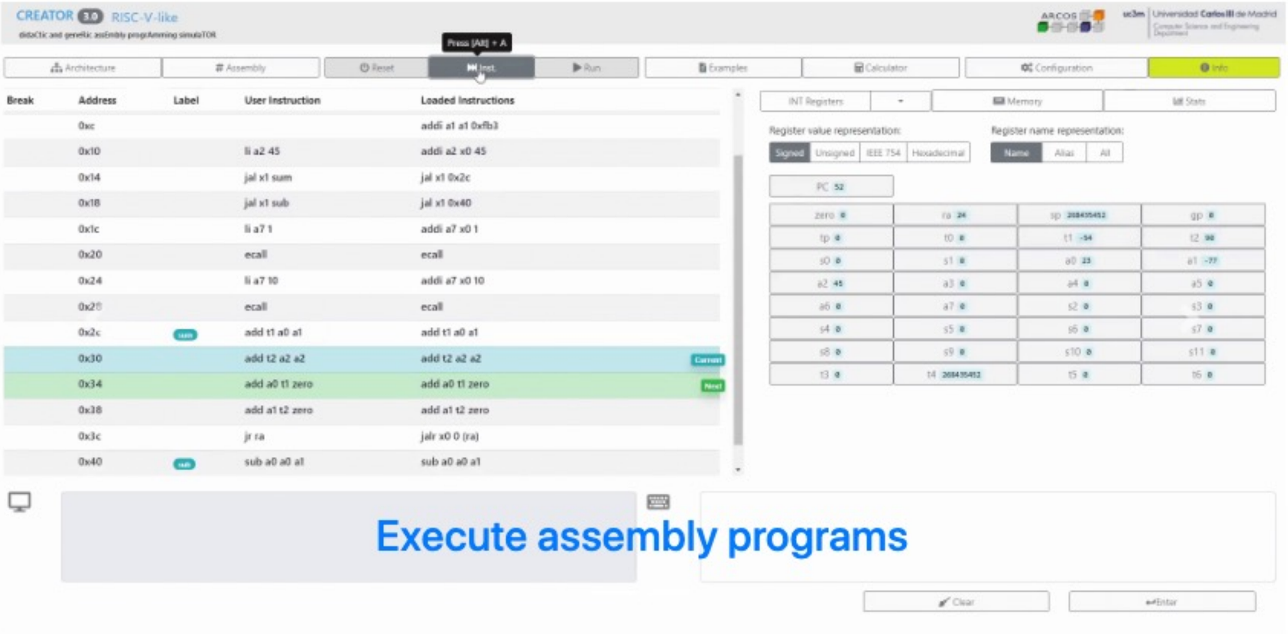
Creator desde el punto de vista docente

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

CREATOR

didaCtic and geneRic assEmbly progrAmming simulaTOR

Access to CREATOR



Break	Address	Label	User Instruction	Loaded Instructions
	0xc			addi a1 a1 0xfb3
	0x10		li a2 45	addi a2 x0 45
	0x14		jal x1 sum	jal x1 0x2c
	0x18		jal x1 sub	jal x1 0x40
	0x1c		li a7 1	addi a7 x0 1
	0x20		ecall	ecall
	0x24		li a7 10	addi a7 x0 10
	0x28		ecall	ecall
	0x2c	sum	add t1 a0 a1	add t1 a0 a1
	0x30		add t2 a2 a2	add t2 a2 a2 Current
	0x34		add a0 t1 zero	add a0 t1 zero Next
	0x38		add a1 t2 zero	add a1 t2 zero
	0x3c		jr ra	jalr x0 0 (ra)
	0x40	sub	sub a0 a0 a1	sub a0 a0 a1

Register value representation: [✓ Signed] Unsigned IEEE 754 Hexadecimal

Register name representation: [✓ Name] Alias All

PC 52

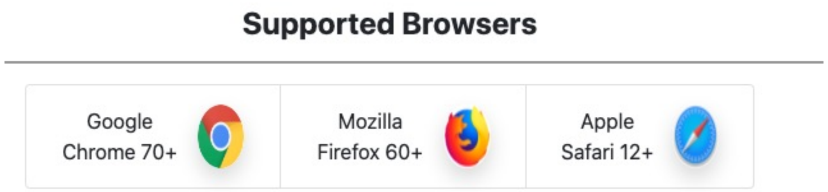
zero 0	ra 24	sp 268435452	gp 0
tp 0	t0 0	t1 -54	t2 90
s0 0	s1 0	a0 23	a1 -77
a2 45	a3 0	a4 0	a5 0
a6 0	a7 0	s2 0	s3 0
s4 0	s5 0	s6 0	s7 0
s8 0	s9 0	s10 0	s11 0
t3 0	t4 268435452	t5 0	t6 0

Execute assembly programs

https://creatorsim.github.io/

Características

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



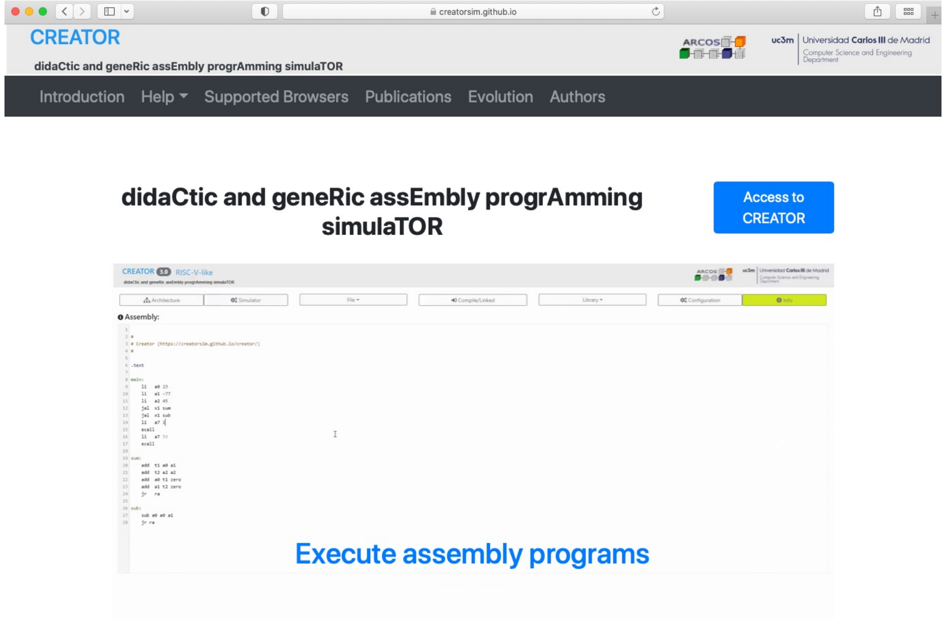
Supported Browsers

Google Chrome 70+	Mozilla Firefox 60+	Apple Safari 12+

Contenido: empleo de CREATOR con RISC-V

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

Disponibilidad



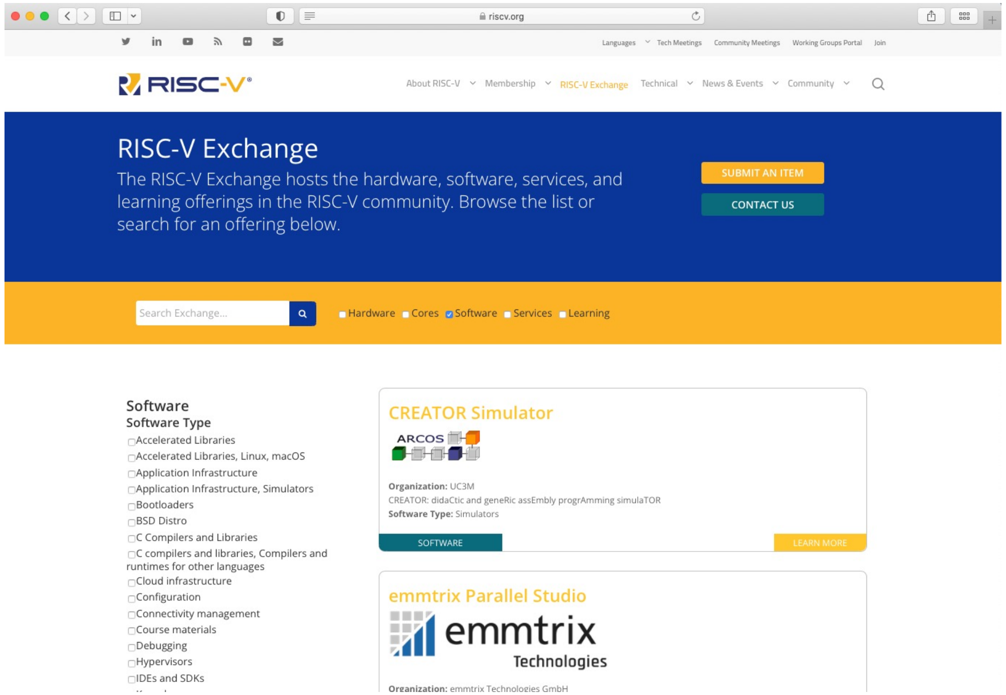
didaCtic and geneRic assEmbly progrAmming simulaTOR

didaCtic and geneRic assEmbly progrAmming simulaTOR

Assembly:

```

#
# Creator (https://creatorsim.github.io/creator/)
#

.text

main:
    li   a0 23
    li   a1 -77
    li   a2 45
    jal  x1 sum
    jal  x1 sub
    li   a7 1
    ecall
    li   a7 10
    ecall

sum:
    add  t1 a0 a1
    add  t2 a2 a2
    add  a0 t1 zero
    add  a1 t2 zero
    jr   ra

sub:
    sub a0 a0 a1
    jr ra
```

Execute assembly programs



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.

□ Hardware □ Cores ☑ Software □ Services □ Learning

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
- □ Configuration
- □ Connectivity management
- □ Course materials
- □ Debugging
- □ Hypervisors
- □ IDEs and SDKs

CREATOR Simulator

Organization: UC3M

CREATOR: didaCtic and geneRic assEmbly progrAmming simulaTOR

Software Type: Simulators

emmtrix Parallel Studio

Organization: emmtrix Technologies GmbH

https://creatorsim.github.io

https://riscv.org/exchange

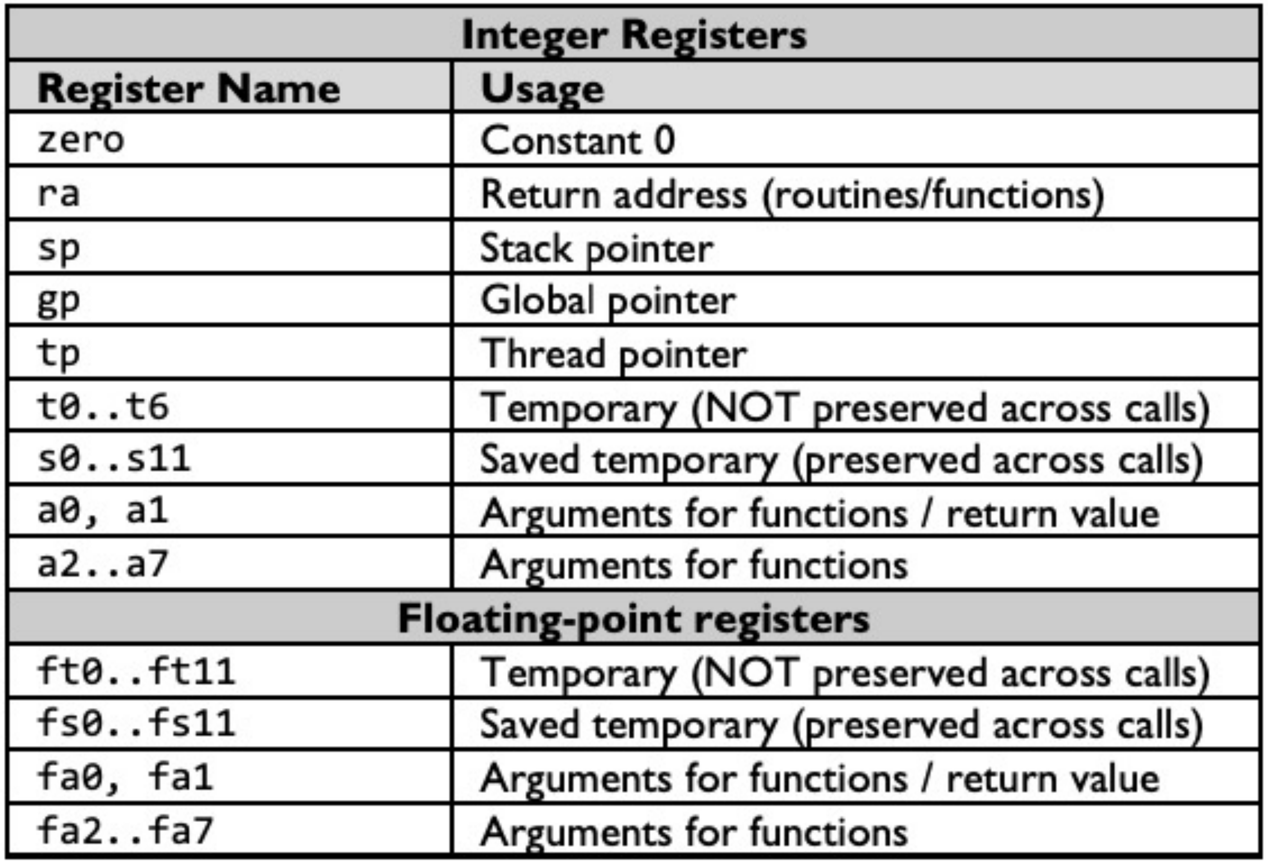
Juego de instrucciones soportado (RV32IMFD)

98 instrucciones y pseudoinstrucciones

Guía de referencia

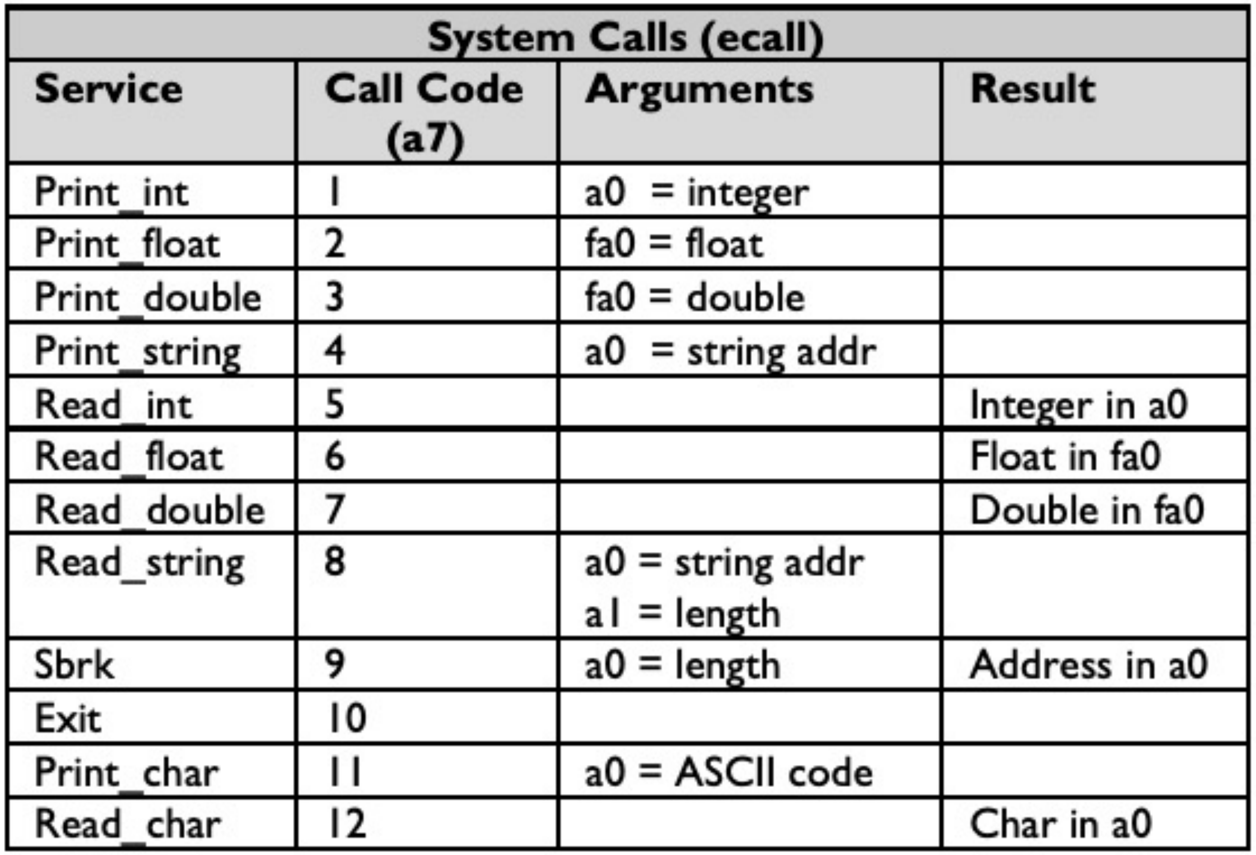
- Transferencia de datos: `li, mv, lui`
- Aritméticas y lógicas sobre registros de enteros: `addi, add, and`, …
- Aritméticas sobre números en coma flotante (float y double): `fadd.s, fmul.d`, …
- Instrucciones de salto (registros enteros): `beq, bne`, …
- Instrucciones de comparación (enteros y coma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

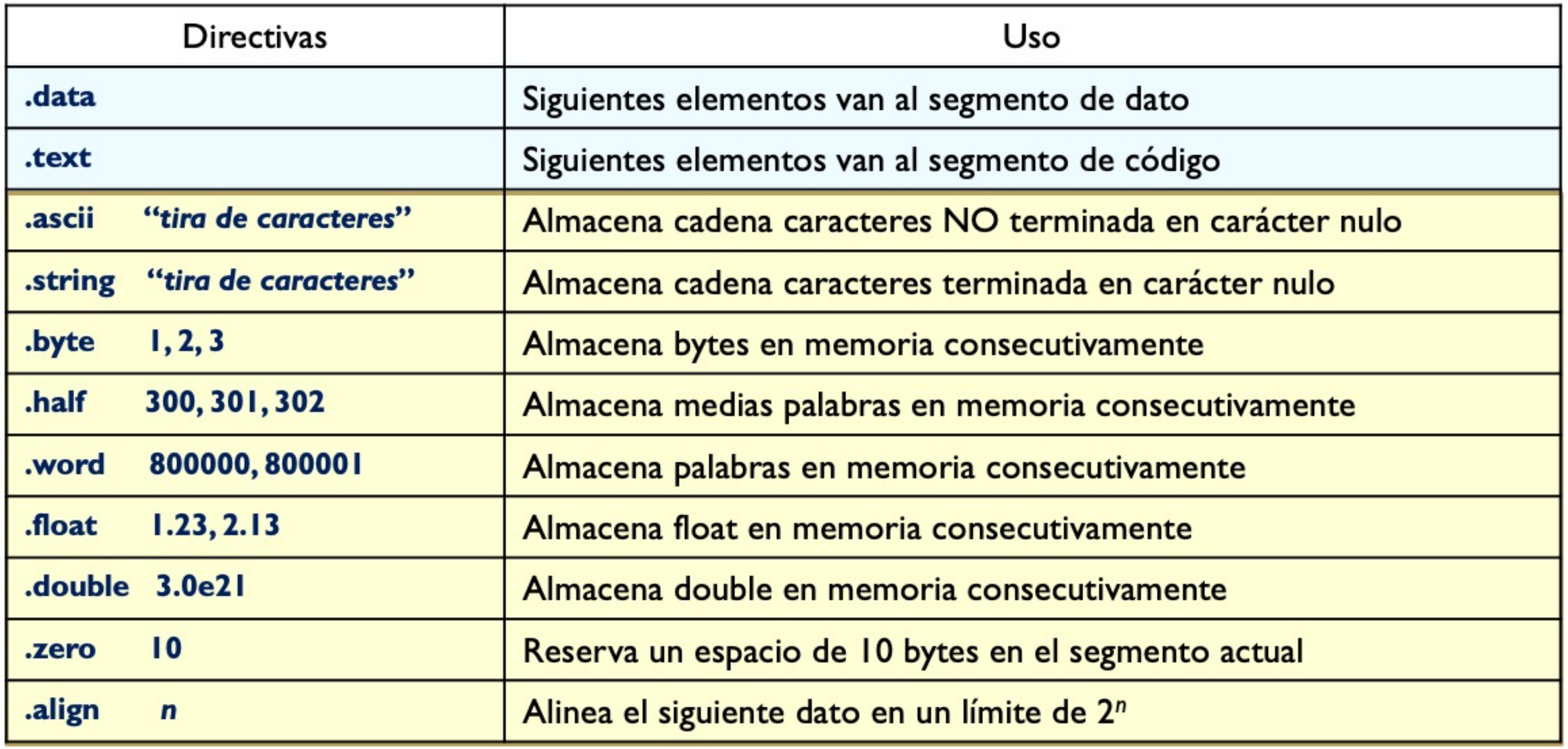


System Calls (ecall)			
Service	**Call Code (a7)**	**Arguments**	**Result**
Print_int	1	a0 = integer	
Print_float	2	fa0 = float	
Print_double	3	fa0 = double	
Print_string	4	a0 = string addr	
Read_int	5		Integer in a0
Read_float	6		Float in fa0
Read_double	7		Double in fa0

Read_string	8	a0 = string addr
		a1 = length	

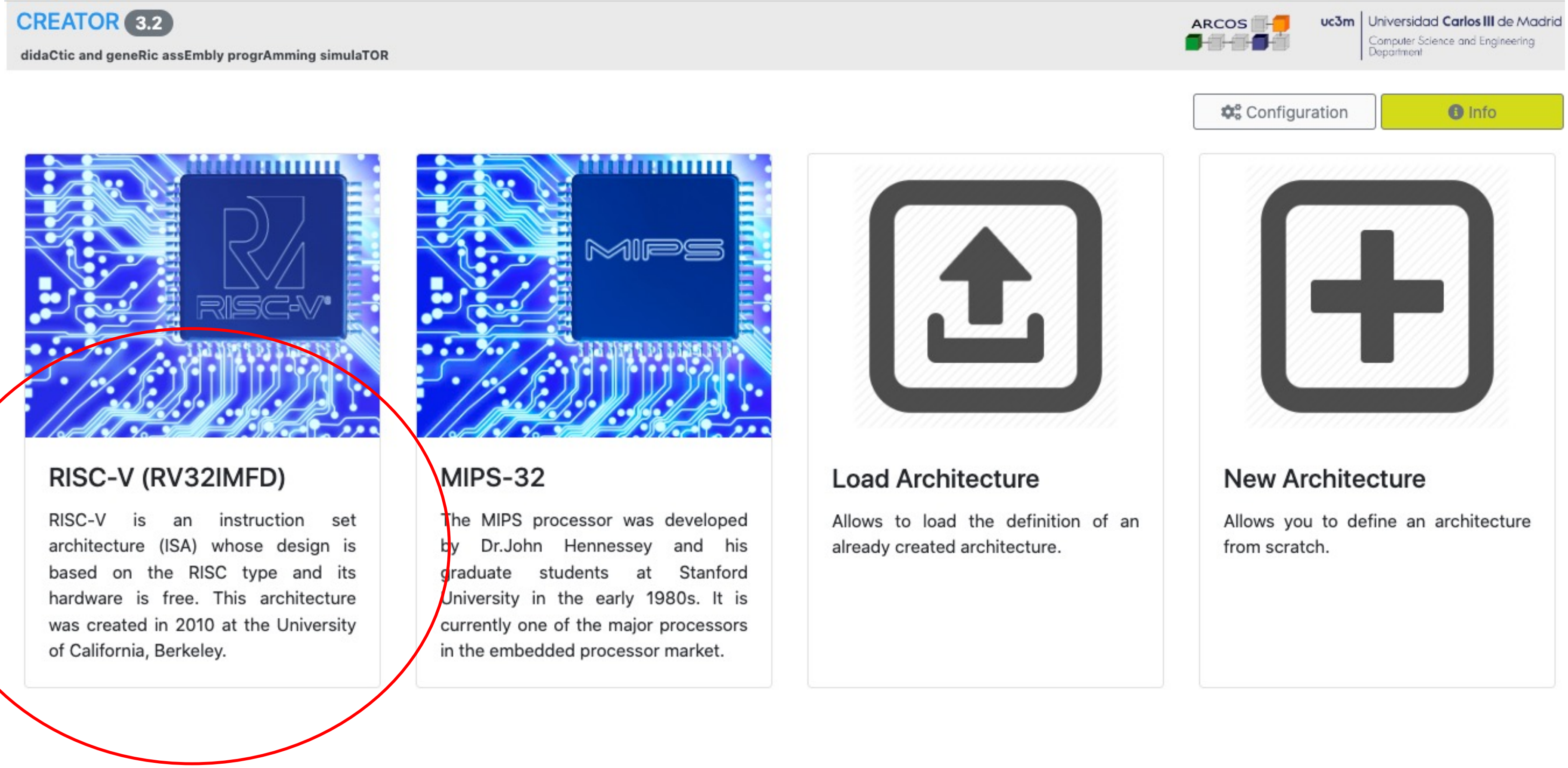
Sbrk	9	a0 = length	Address in a0
Exit	10		
Print_char	11	a0 = ASCII code	
Read_char	12		Char in a0

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)



CREATOR 3.2

didaCtic and geneRic assEmbly progrAmming simulaTOR

ARCOS
uc3m Universidad Carlos III de Madrid
Computer Science and Engineering Department
Configuration
Info
RISC-V
MIPS

RISC-V (RV32IMFD)

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

MIPS-32

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

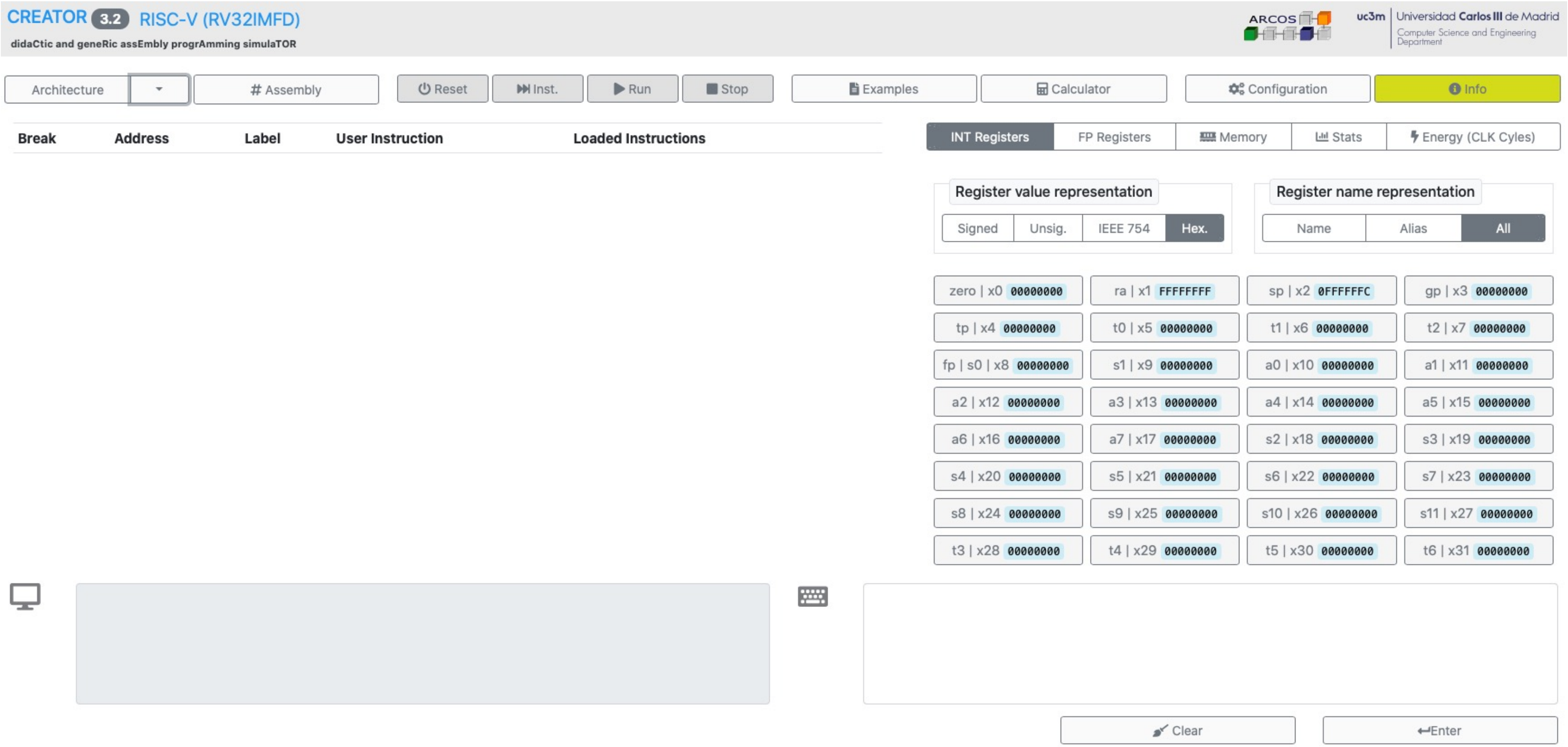
Load Architecture

Allows to load the definition of an already created architecture.

New Architecture

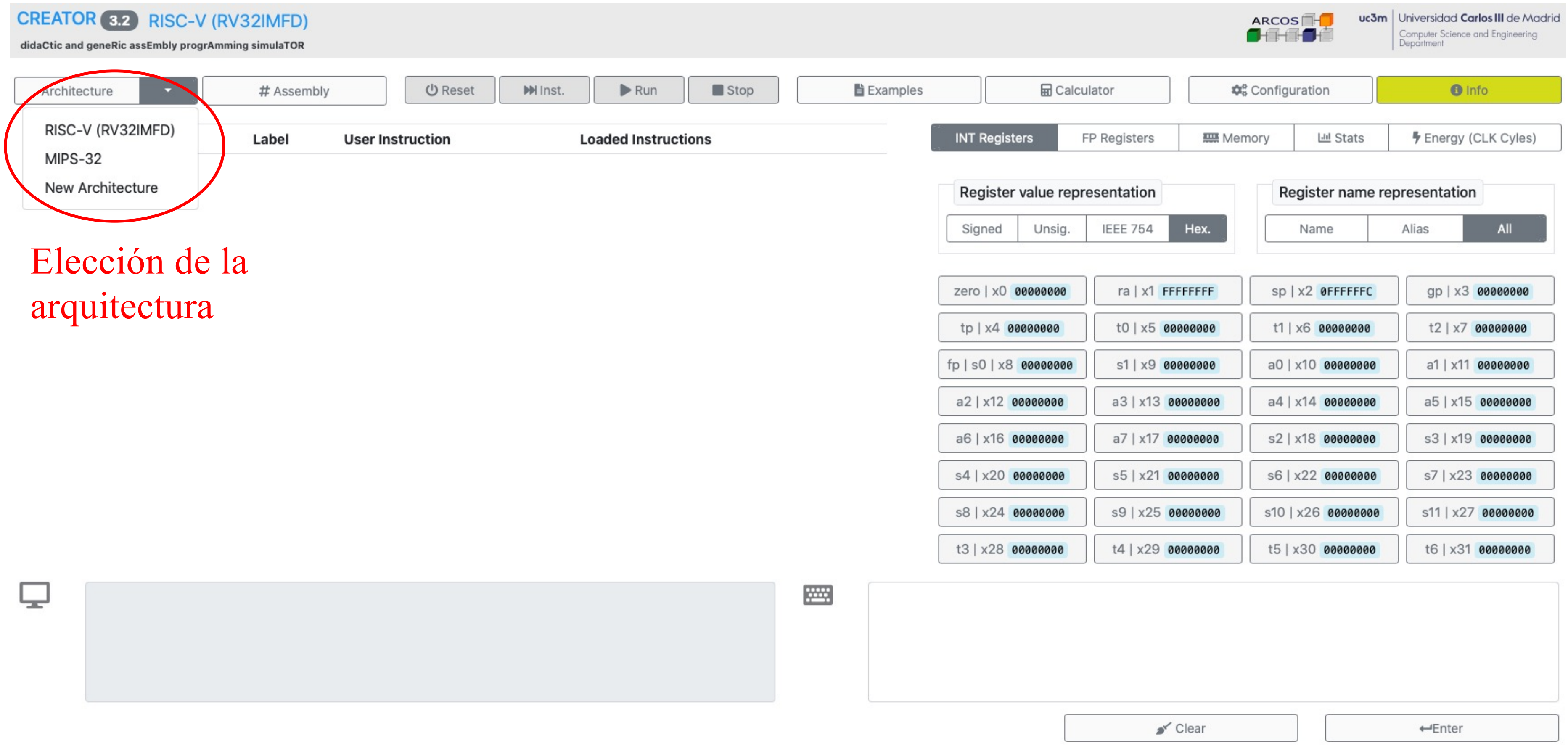
Allows you to define an architecture from scratch.

Pantalla inicial



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

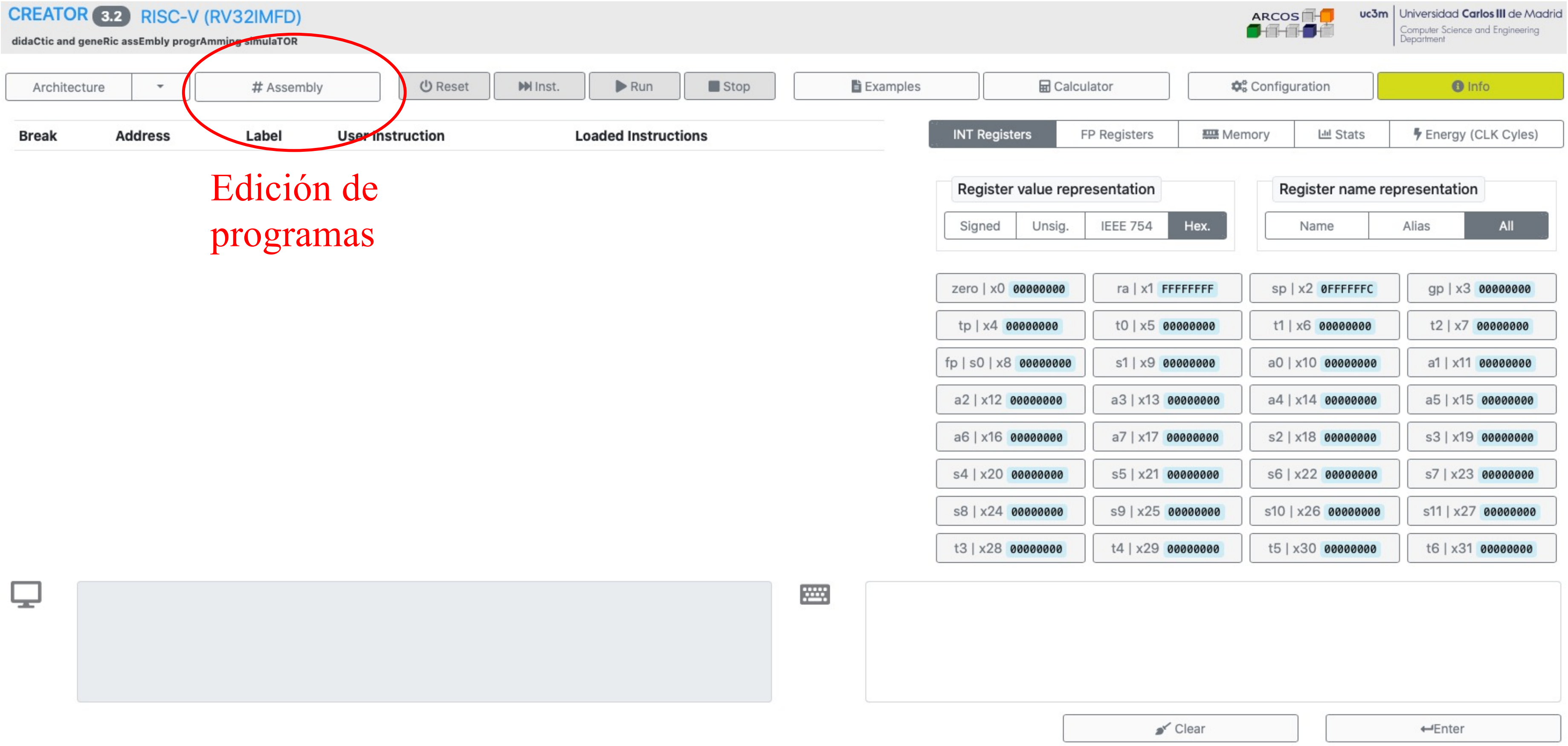
Elección de la arquitectura



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

Elección de la arquitectura

Edición de programas



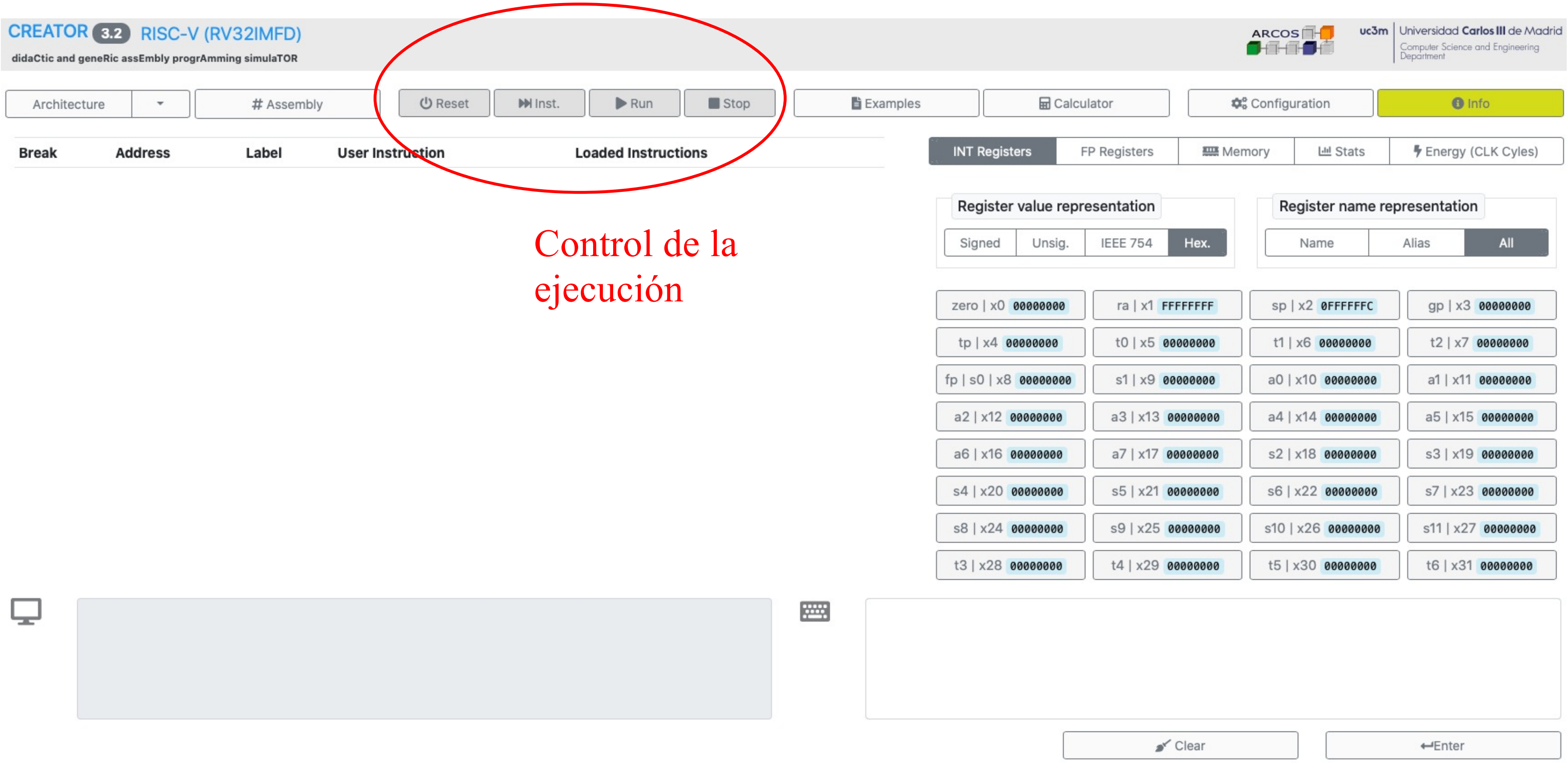
Register value representation: Signed | Unsig. | IEEE 754 | [✓ Hex.]

Register name representation: Name | Alias | [✓ All]

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

Edición de programas

Control de la ejecución



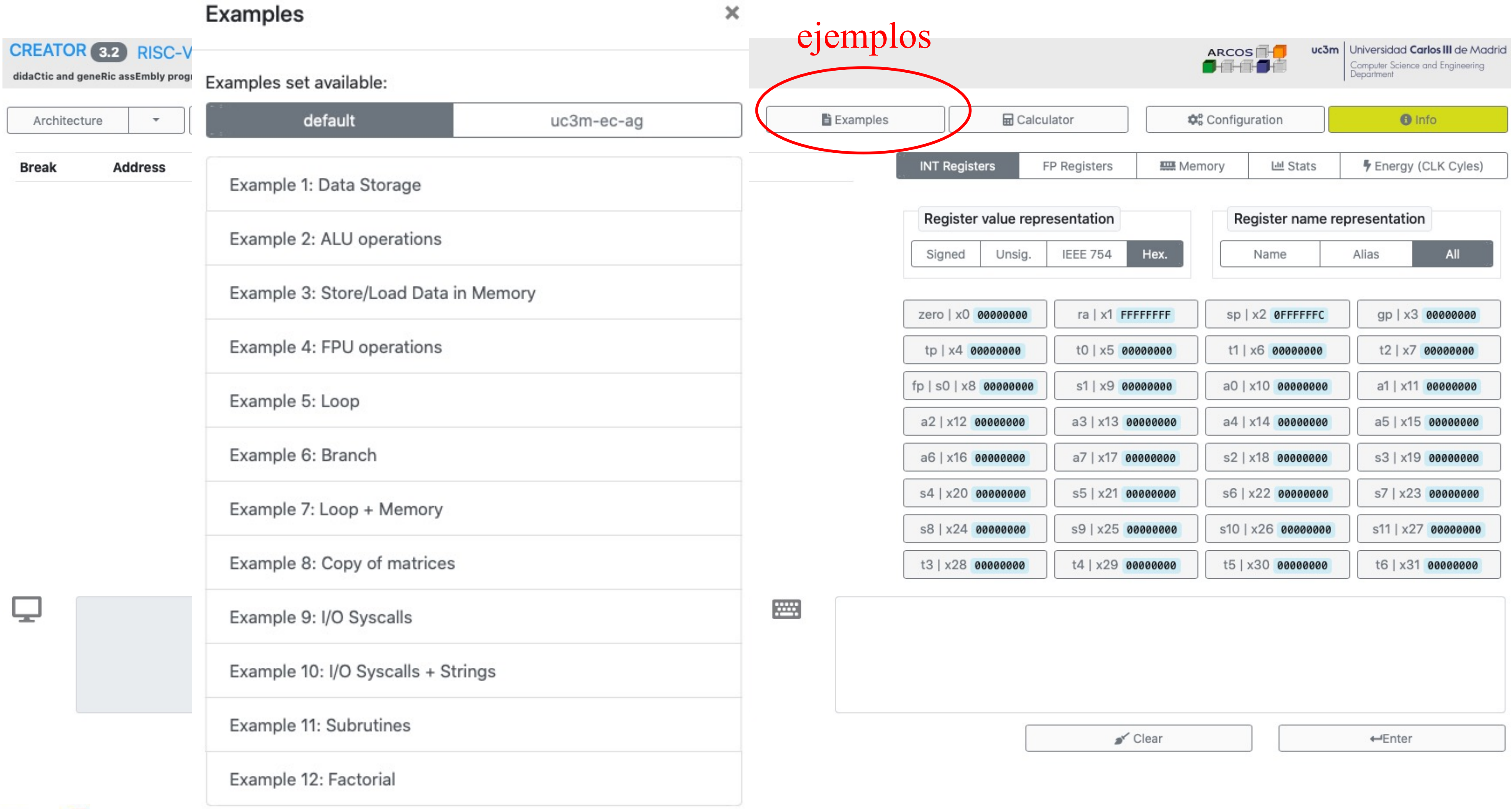
Control de la ejecución

Register value representation: Signed | Unsig. | IEEE 754 | [✓ Hex.]

Register name representation: Name | Alias | [✓ All]

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

Ejemplos de programas en ensamblador



ejemplos

Examples

Examples set available:

default | uc3m-ec-ag

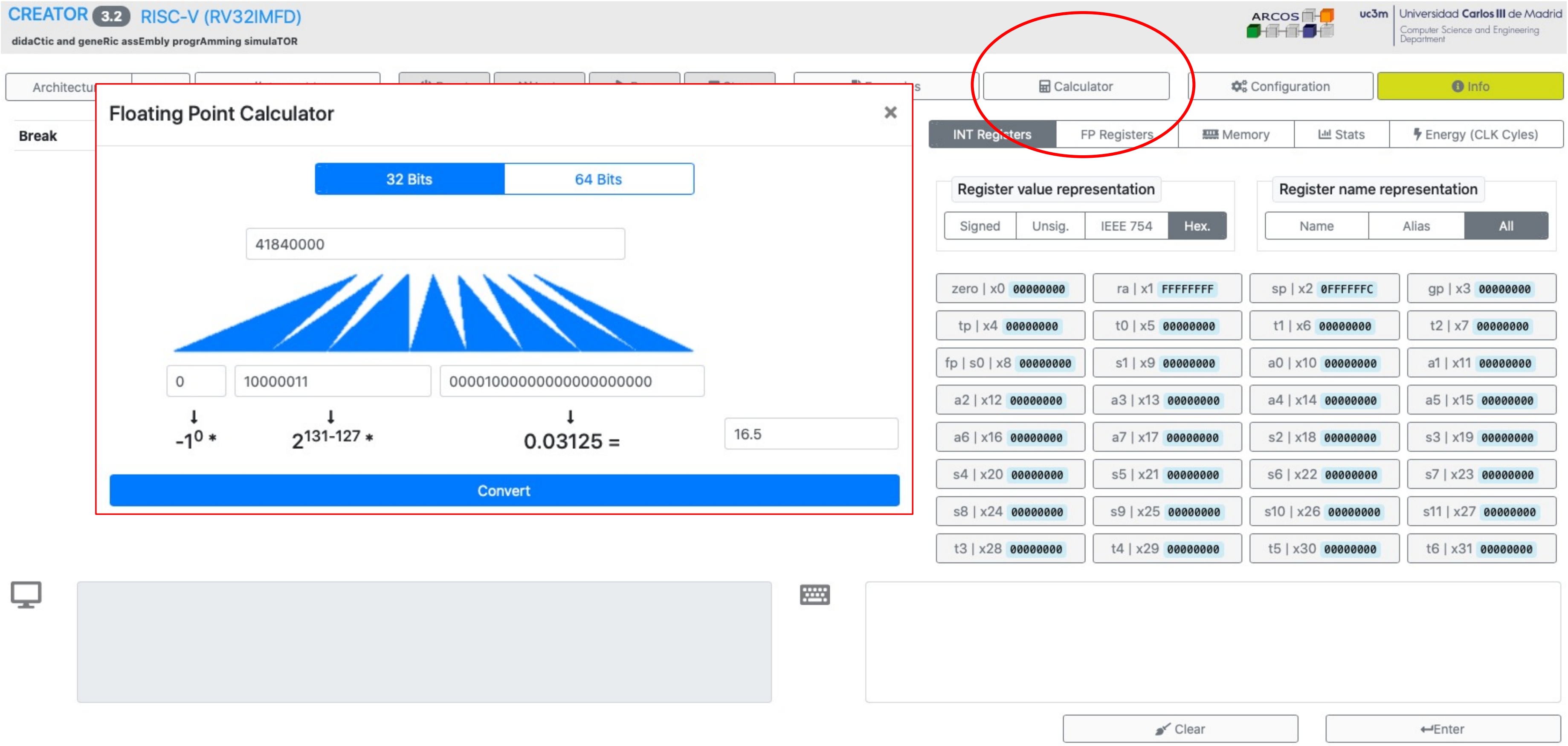
- Example 1: Data Storage
- Example 2: ALU operations
- Example 3: Store/Load Data in Memory
- Example 4: FPU operations
- Example 5: Loop
- Example 6: Branch
- Example 7: Loop + Memory
- Example 8: Copy of matrices
- Example 9: I/O Syscalls
- Example 10: I/O Syscalls + Strings
- Example 11: Subrutines
- Example 12: Factorial

Register value representation: Signed | Unsig. | IEEE 754 | **Hex.**

Register name representation: Name | Alias | **All**

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

Calculadora de números en coma flotante


CREATOR 3.2 RISC-V (RV32IMFD)
didaCtic and geneRic assEmbly progrAmming simulaTOR
ARCOS
uc3m Universidad Carlos III de Madrid
Computer Science and Engineering Department
Architectu
Calculator
Configuration
Info
Break
INT Registers
FP Registers
Memory
Stats
Energy (CLK Cyles)
Clear
↵Enter

Floating Point Calculator

32 Bits | 64 Bits

41840000

0	10000011	00001000000000000000000

$-1^{0} * \quad 2^{131-127} * \quad 0.03125 =$ 16.5

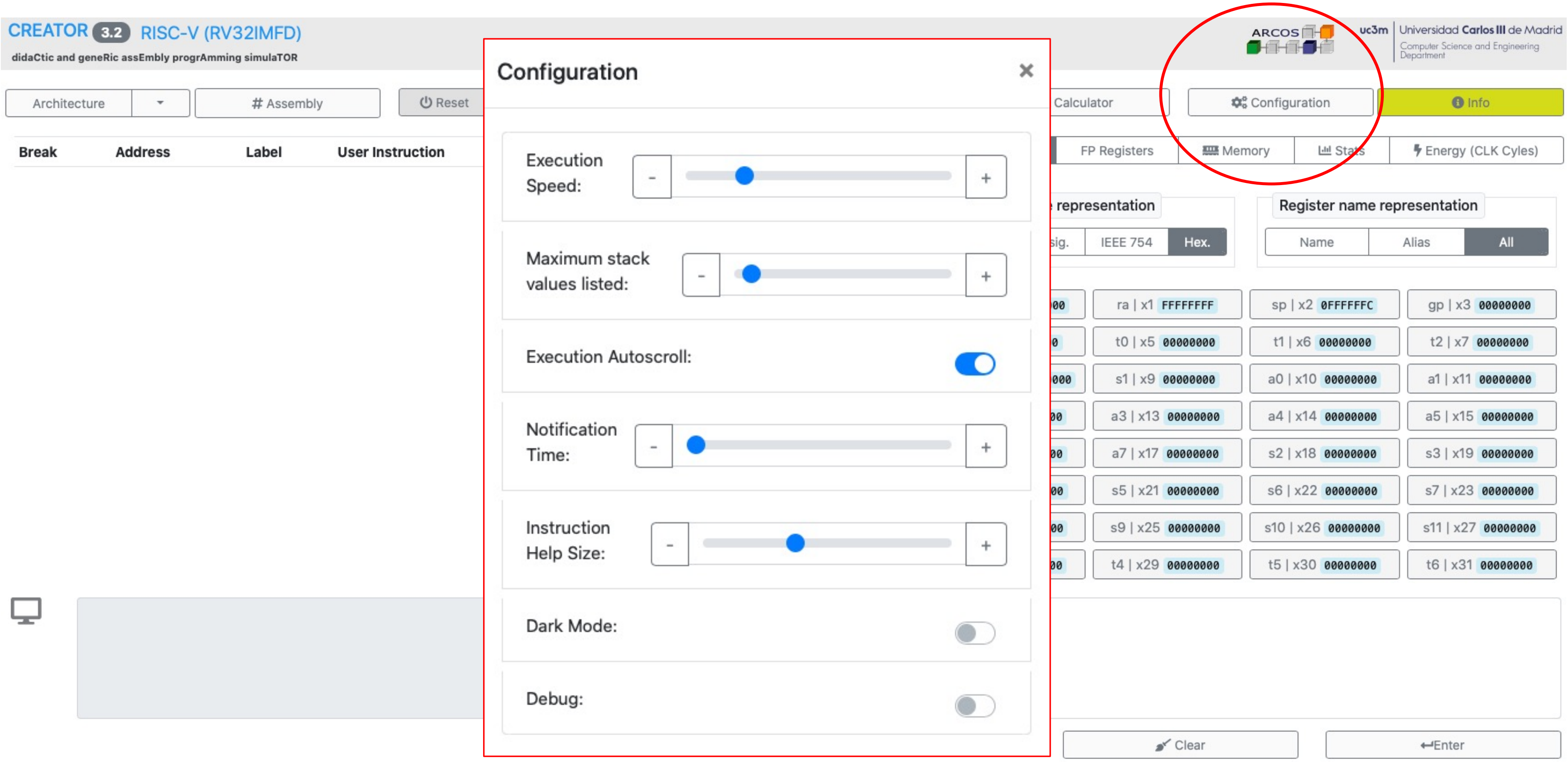
Convert

Register value representation: Signed | Unsig. | IEEE 754 | **Hex.**

Register name representation: Name | Alias | **All**

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

Configuración



Configuration ×

Execution Speed: - +

Maximum stack values listed: - +

Execution Autoscroll: ●

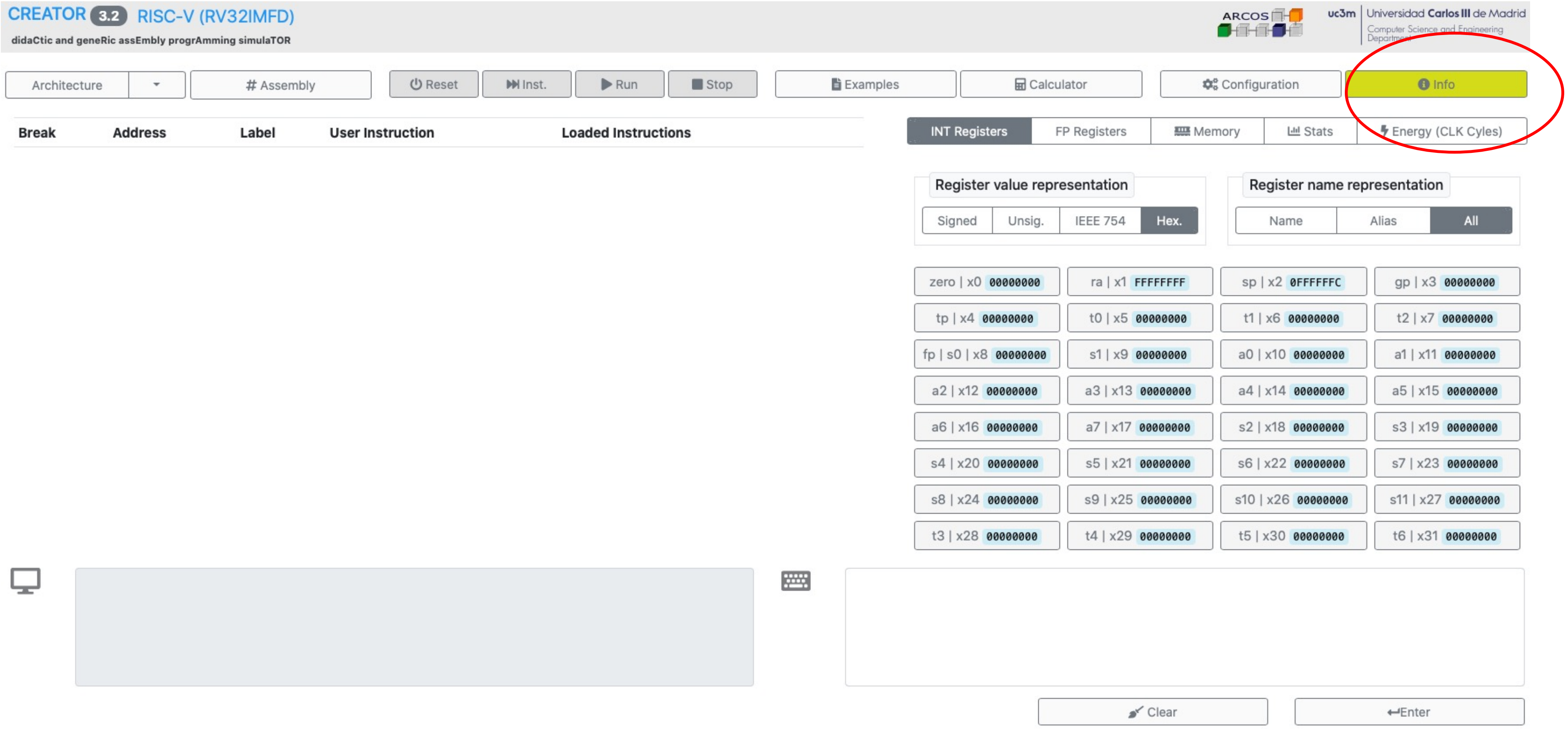
Notification Time: - +

Instruction Help Size: - +

Dark Mode: ○

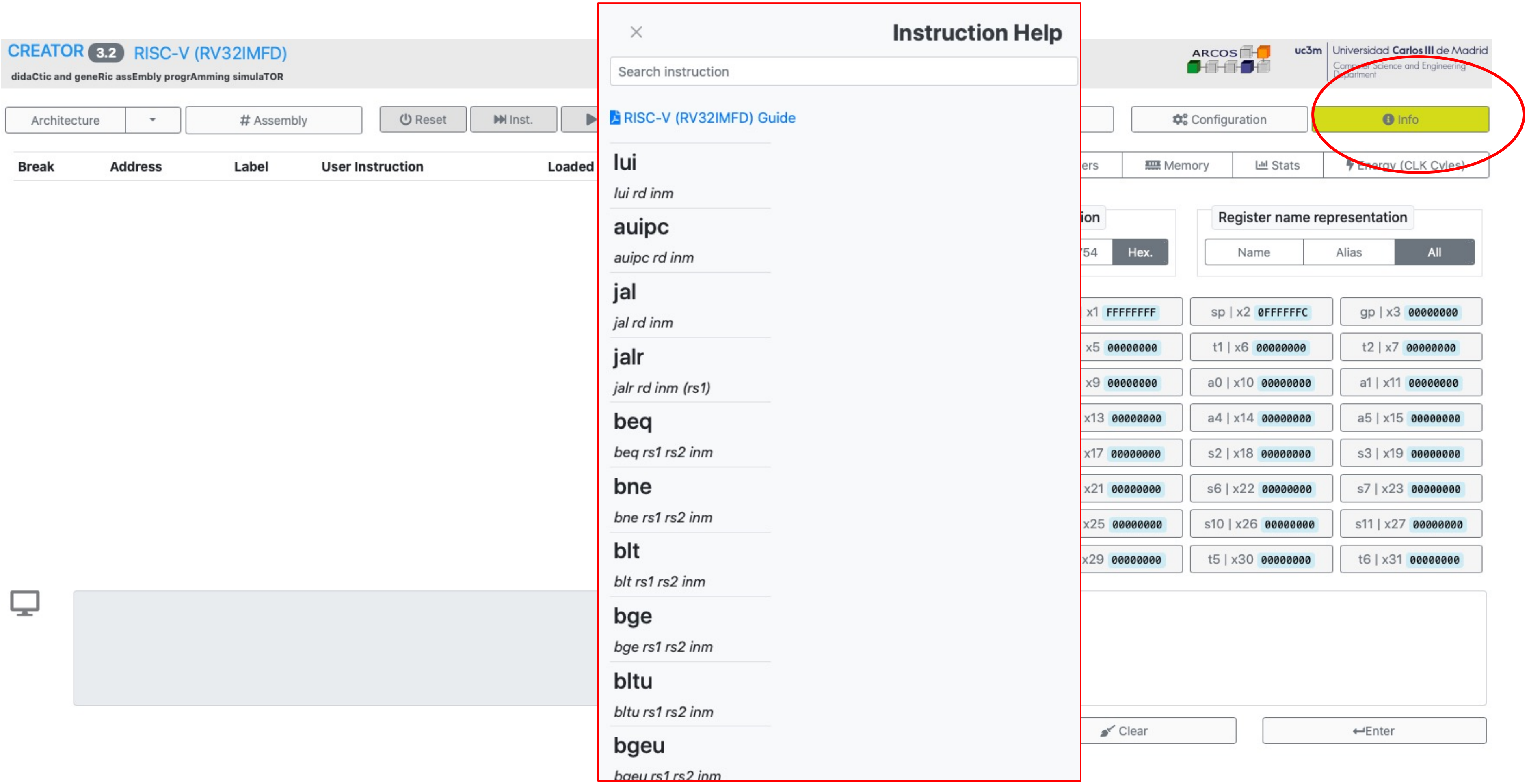
Debug: ○

Configuración



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

Configuración



CREATOR 3.2 RISC-V (RV32IMFD)
didaCtic and geneRic assEmbly progrAmming simulaTOR
ARCOS
uc3m Universidad Carlos III de Madrid
Computer Science and Engineering Department
Architecture
# Assembly
Reset
Inst.
Configuration
Info
Break
Address
Label
User Instruction
Loaded
ers
Memory
Stats
Energy (CLK Cycles)
ion
754
Hex.
Register name representation
Name
Alias
All
Clear
Enter

Instruction Help

Search instruction

RISC-V (RV32IMFD) Guide

lui
lui rd inm

auipc
auipc rd inm

jal
jal rd inm

jalr
jalr rd inm (rs1)

beq
beq rs1 rs2 inm

bne
bne rs1 rs2 inm

blt
blt rs1 rs2 inm

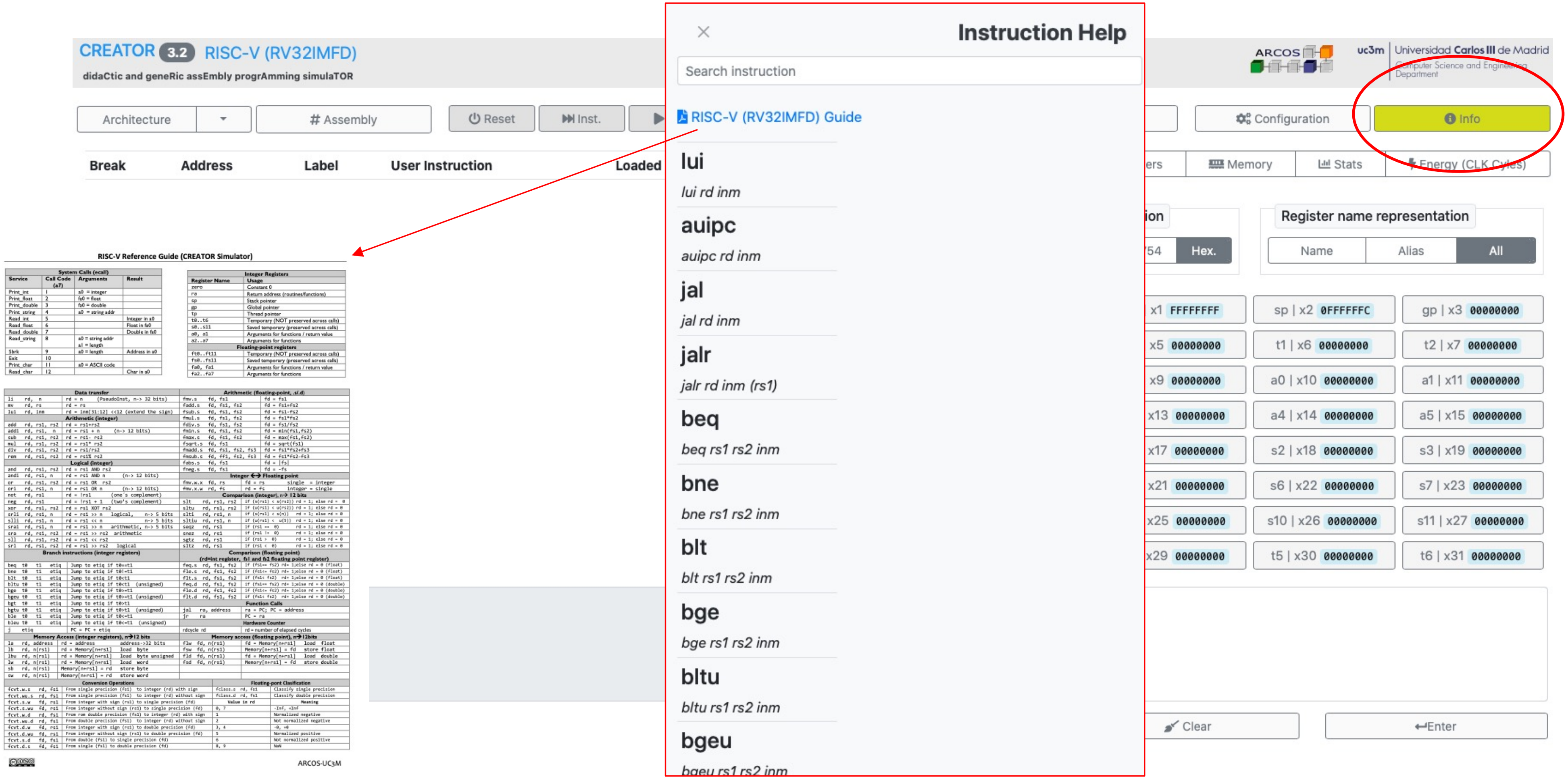
bge
bge rs1 rs2 inm

bltu
bltu rs1 rs2 inm

bgeu
bgeu rs1 rs2 inm

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

Configuración



CREATOR 3.2 RISC-V (RV32IMFD)

didaCtic and geneRic assEmbly progrAmming simulaTOR

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

Instruction Help

Search instruction

RISC-V (RV32IMFD) Guide

lui
lui rd inm

auipc
auipc rd inm

jal
jal rd inm

jalr
jalr rd inm (rs1)

beq
beq rs1 rs2 inm

bne
bne rs1 rs2 inm

blt
blt rs1 rs2 inm

bge
bge rs1 rs2 inm

bltu
bltu rs1 rs2 inm

bgeu
bgeu rs1 rs2 inm

RISC-V Reference Guide (CREATOR Simulator)

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

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

Data transfer	
li rd, n	rd = n (PseudoInst, n-> 32 bits)
mv rd, rs	rd = rs
lui rd, inm	rd = inm[31:12] <<12 (extend the sign)
Arithmetic (integer)	
add rd, rs1, rs2	rd = rs1+rs2
addi rd, rs1, n	rd = rs1 + n (n-> 12 bits)
sub rd, rs1, rs2	rd = rs1- rs2
mul rd, rs1, rs2	rd = rs1* rs2
div rd, rs1, rs2	rd = rs1/rs2
rem rd, rs1, rs2	rd = rs1% rs2
Logical (integer)	
and rd, rs1, rs2	rd = rs1 AND rs2
andi rd, rs1, n	rd = rs1 AND n (n-> 12 bits)
or rd, rs1, rs2	rd = rs1 OR rs2
ori rd, rs1, n	rd = rs1 OR n (n-> 12 bits)
not rd, rs1	rd = !rs1 (one´s complement)
neg rd, rs1	rd = !rs1 + 1 (two's complement)
xor rd, rs1, rs2	rd = rs1 XOT rs2
srli rd, rs1, n	rd = rs1 >> n logical, n-> 5 bits
slli rd, rs1, n	rd = rs1 << n n-> 5 bits
srai rd, rs1, n	rd = rs1 >> n arithmetic, n-> 5 bits
sra rd, rs1, rs2	rd = rs1 >> rs2 arithmetic
sll rd, rs1, rs2	rd = rs1 << rs2
srl rd, rs1, rs2	rd = rs1 >> rs2 logical
Branch instructions (integer registers)	
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)
bge t0 t1 etiq	Jump to etiq if t0>=t1
bgeu 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
Memory Access (integer registers), n→12 bits	
la rd, address	rd = address address->32 bits
lb rd, n(rs1)	rd = Memory[n+rs1] load byte
lbu rd, n(rs1)	rd = Memory[n+rs1] load byte unsigned
lw rd, n(rs1)	rd = Memory[n+rs1] load word
sb rd, n(rs1)	Memory[n+rs1] = rd store byte
sw rd, n(rs1)	Memory[n+rs1] = rd store word

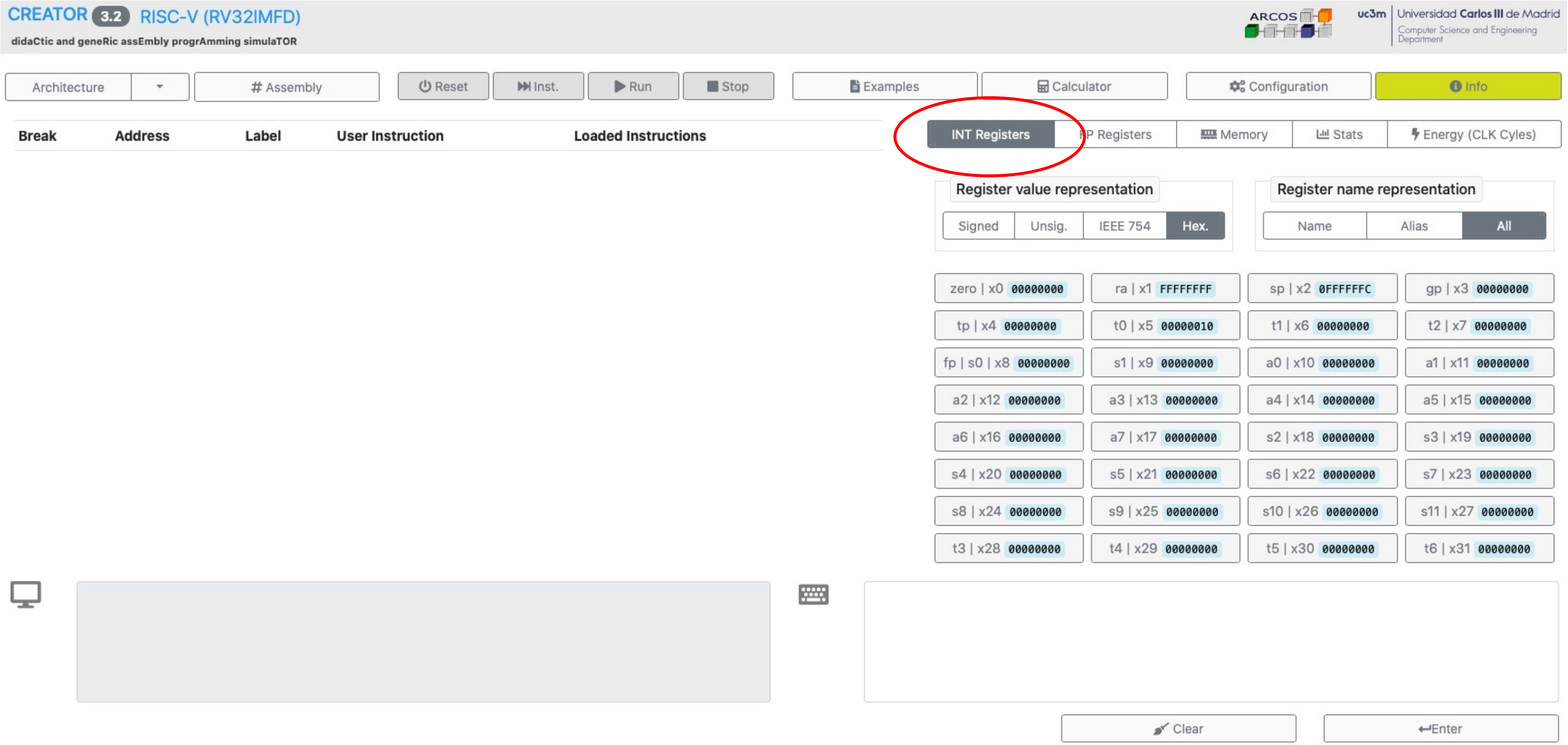
Arithmetic (floating-point, .s/.d)	
fmv.s fd, fs1	fd = fs1
fadd.s fd, fs1, fs2	fd = fs1+fs2
fsub.s fd, fs1, fs2	fd = fs1-fs2
fmul.s fd, fs1, fs2	fd = fs1*fs2
fdiv.s fd, fs1, fs2	fd = fs1/fs2
fmin.s fd, fs1, fs2	fd = min(fs1,fs2)
fmax.s fd, fs1, fs2	fd = max(fs1,fs2)
fsqrt.s fd, fs1	fd = sqrt(fs1)
fmadd.s fd, fs1, fs2, fs3	fd = fs1*fs2+fs3
fmsub.s fd, ff1, fs2, fs3	fd = fs1*fs2-fs3
fabs.s fd, fs1	fd = |fs|
fneg.s fd, fs1	fd = -fs
Integer ↔ Floating point	
fmv.w.x fd, rs	fd = rs single = integer
fmv.x.w rd, fs	rd = fs integer = single
Comparison (integer), n→ 12 bits	
slt rd, rs1, rs2	if (s(rs1) < s(rs2)) rd = 1; else rd = 0
sltu rd, rs1, rs2	if (u(rs1) < u(rs2)) rd = 1; else rd = 0
slti rd, rs1, n	if (s(rs1) < s(n)) rd = 1; else rd = 0
sltiu rd, rs1, n	if (u(rs1) < u(5)) rd = 1; else rd = 0
seqz rd, rs1	if (rs1 == 0) rd = 1; else rd = 0
snez rd, rs1	if (rs1 != 0) rd = 1; else rd = 0
sgtz rd, rs1	if (rs1 > 0) rd = 1; else rd = 0
sltz rd, rs1	if (rs1 < 0) rd = 1; else rd = 0
Comparison (floating point) (rd=int register, fs1 and fs2 floating point register)	
feq.s rd, fs1, fs2	if (fs1== fs2) rd= 1;else rd = 0 (float)
fle.s rd, fs1, fs2	if (fs1<= fs2) rd= 1;else rd = 0 (float)
flt.s rd, fs1, fs2	if (fs1< fs2) rd= 1;else rd = 0 (float)
feq.d rd, fs1, fs2	if (fs1== fs2) rd= 1;else rd = 0 (double)
fle.d rd, fs1, fs2	if (fs1<= fs2) rd= 1;else rd = 0 (double)
flt.d rd, fs1, fs2	if (fs1< fs2) rd= 1;else rd = 0 (double)
Function Calls	
jal ra, address	ra = PC; PC = address
jr ra	PC = ra
Hardware Counter	
rdcycle rd	rd = number of elapsed cycles
Memory access (floating point), n→12bits	
flw fd, n(rs1)	fd = Memory[n+rs1] load float
fsw fd, n(rs1)	Memory[n+rs1] = fd store float
fld fd, n(rs1)	fd = Memory[n+rs1] load double
fsd fd, n(rs1)	Memory[n+rs1] = fd store double

Conversion Operations	
fcvt.w.s rd, fs1	From single precision (fs1) to integer (rd) with sign
fcvt.wu.s rd, fs1	From single precision (fs1) to integer (rd) without sign
fcvt.s.w fd, rs1	From integer with sign (rs1) to single precision (fd)
fcvt.s.wu fd, rs1	From integer without sign (rs1) to single precision (fd)
fcvt.w.d rd, fs1	From rom double precision (fs1) to integer (rd) with sign
fcvt.wu.d rd, fs1	From double precision (fs1) to integer (rd) without sign
fcvt.d.w fd, rs1	From integer with sign (rs1) to double precision (fd)
fcvt.d.wu fd, rs1	From integer without sign (rs1) to double precision (fd)
fcvt.s.d fd, fs1	From double (fs1) to single precision (fd)
fcvt.d.s fd, fs1	From single (fs1) to double precision (fd)

Floating-pont Clasification	
fclass.s rd, fs1	Classify single precision
fclass.d rd, fs1	Classify double precision
Value in rd	**Meaning**
0, 7	-Inf, +Inf
1	Normalized negative
2	Not normalized negative
3, 4	-0, +0
5	Normalized positive
6	Not normalized positive
8, 9	NaN

ARCOS-UC3M

Registros enteros

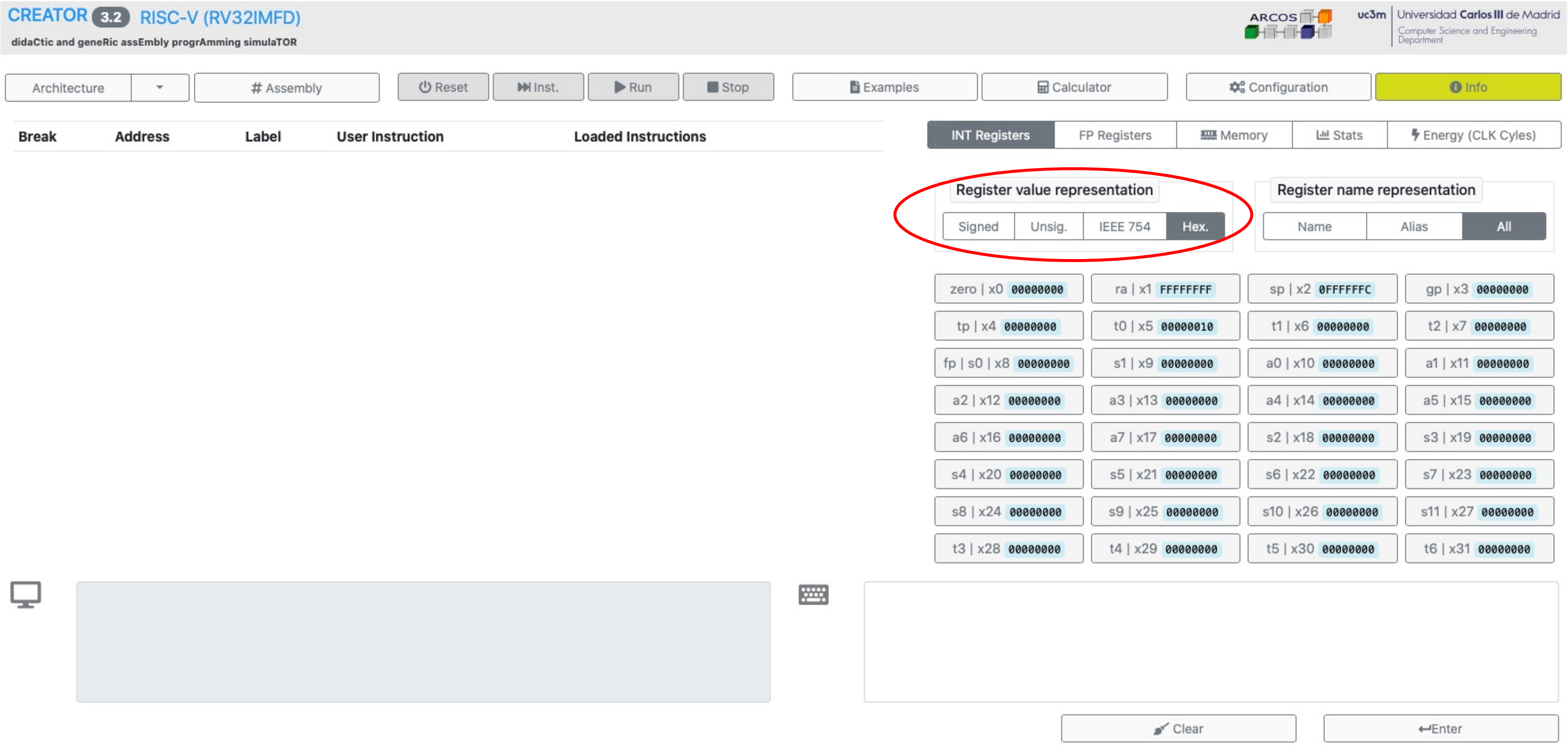


Register value representation: Signed | Unsig. | IEEE 754 | [✓ Hex.]

Register name representation: Name | Alias | [✓ All]

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

Registros enteros


CREATOR 3.2 RISC-V (RV32IMFD)
didaCtic and geneRic assEmbly progrAmming simulaTOR
ARCOS
uc3m Universidad Carlos III de Madrid
Computer Science and Engineering Department
Architecture
# Assembly
Reset
Inst.
Run
Stop
Examples
Calculator
Configuration
Info
INT Registers
FP Registers
Memory
Stats
Energy (CLK Cyles)
Clear
Enter

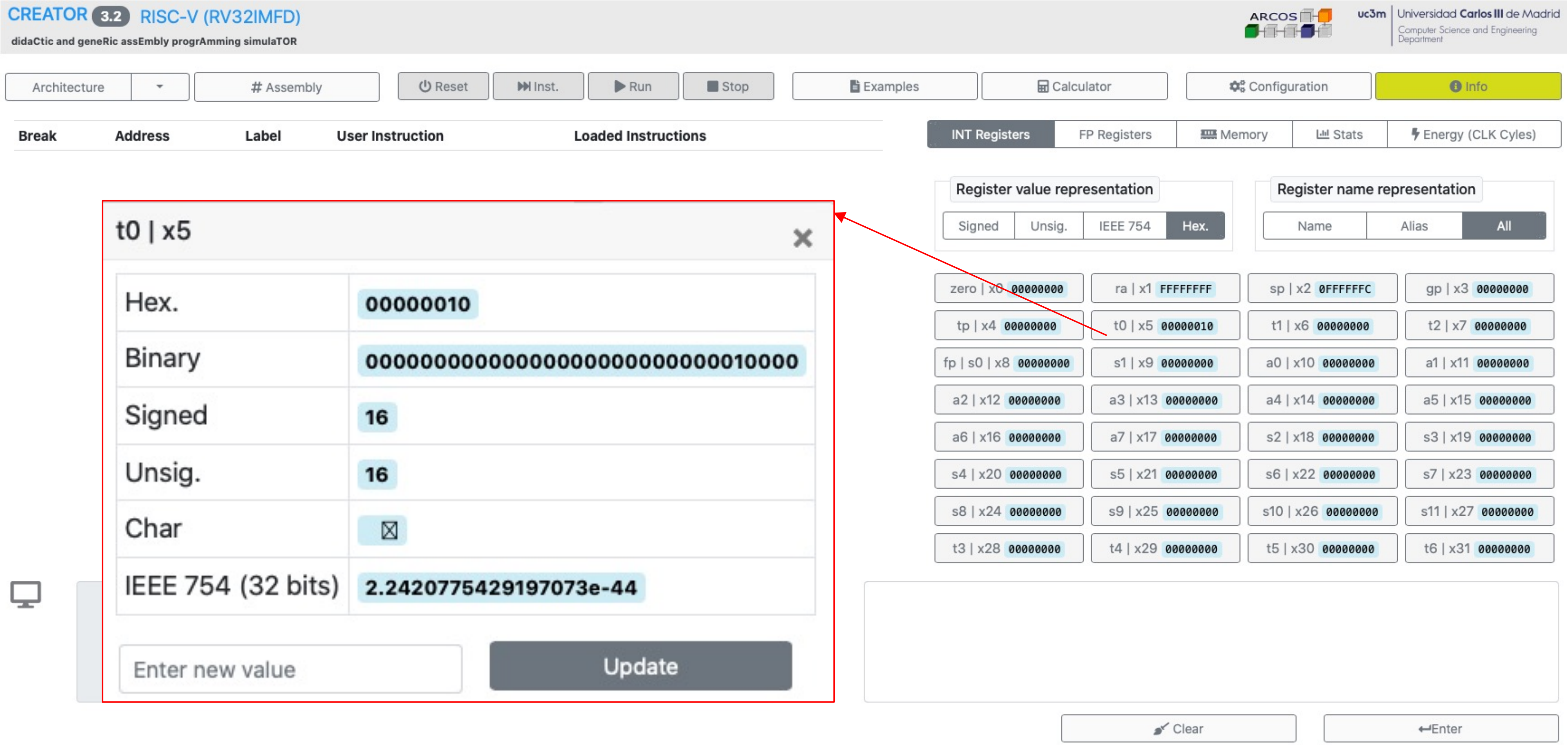
Break	Address	Label	User Instruction	Loaded Instructions

Register value representation: Signed Unsig. IEEE 754 [✓ Hex.]

Register name representation: Name Alias [✓ All]

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

Registros enteros

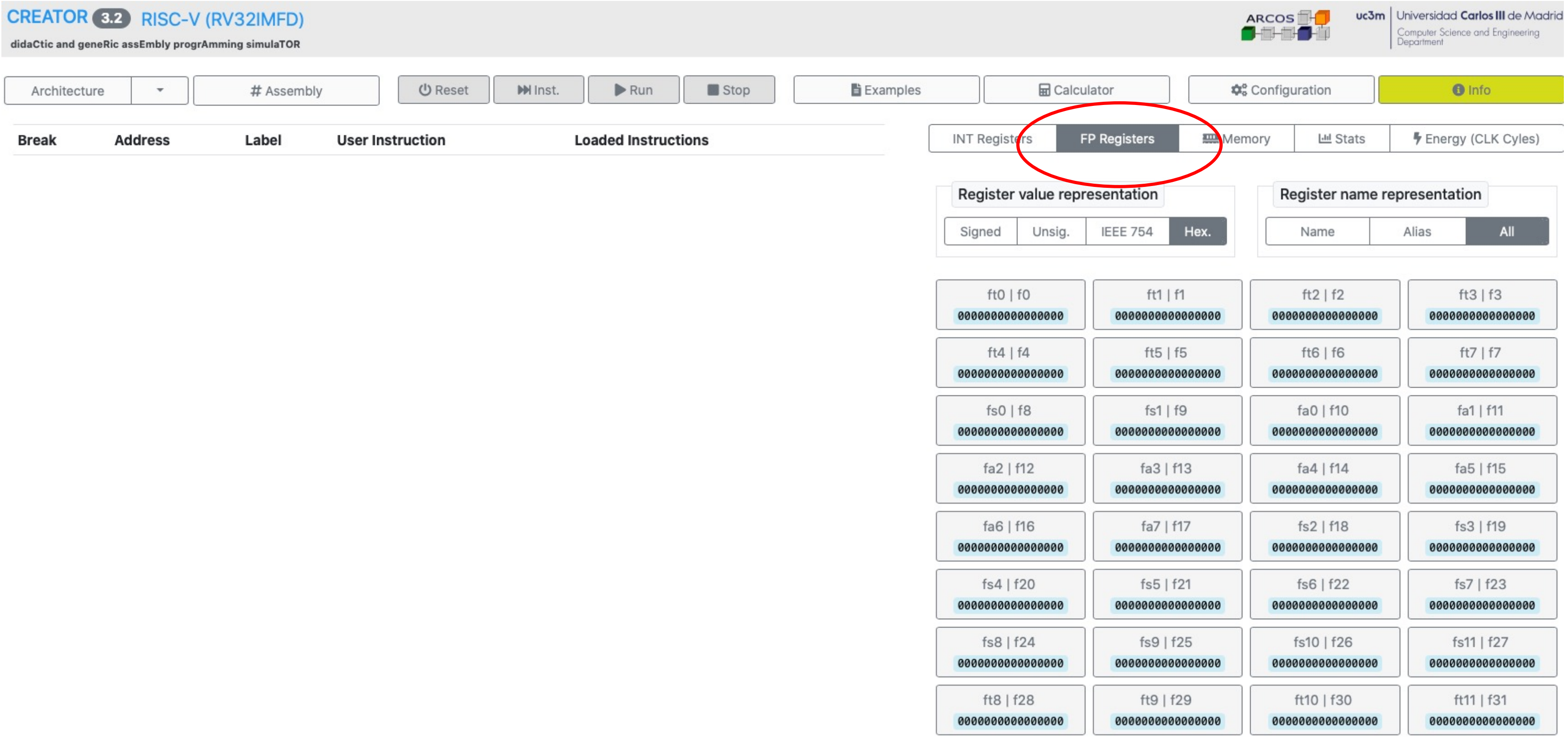


t0 | x5

Hex.	00000010
Binary	00000000000000000000000000010000
Signed	16
Unsig.	16
Char	☒
IEEE 754 (32 bits)	2.2420775429197073e-44

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

Registros en coma flotante

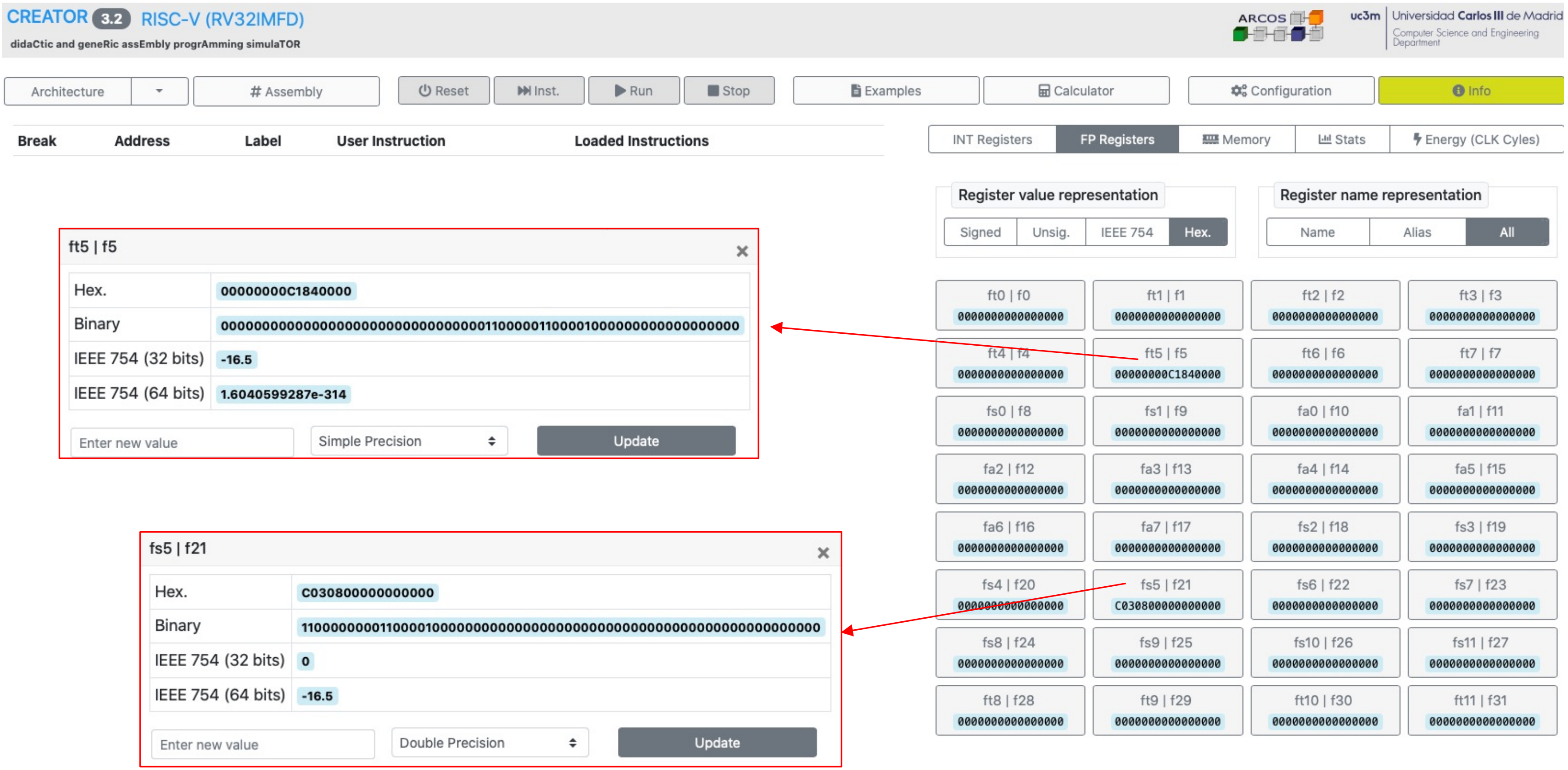


Register value representation: Signed | Unsig. | IEEE 754 | **Hex.**

Register name representation: Name | Alias | **All**

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

Registros en coma flotante



CREATOR 3.2 RISC-V (RV32IMFD)

didaCtic and geneRic assEmbly progrAmming simulaTOR

Register value representation: Signed | Unsig. | IEEE 754 | **Hex.**

Register name representation: Name | Alias | **All**

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

ft5 | f5

Hex.	00000000C1840000
Binary	0000000000000000000000000000000011000001100001000000000000000000
IEEE 754 (32 bits)	-16.5
IEEE 754 (64 bits)	1.6040599287e-314

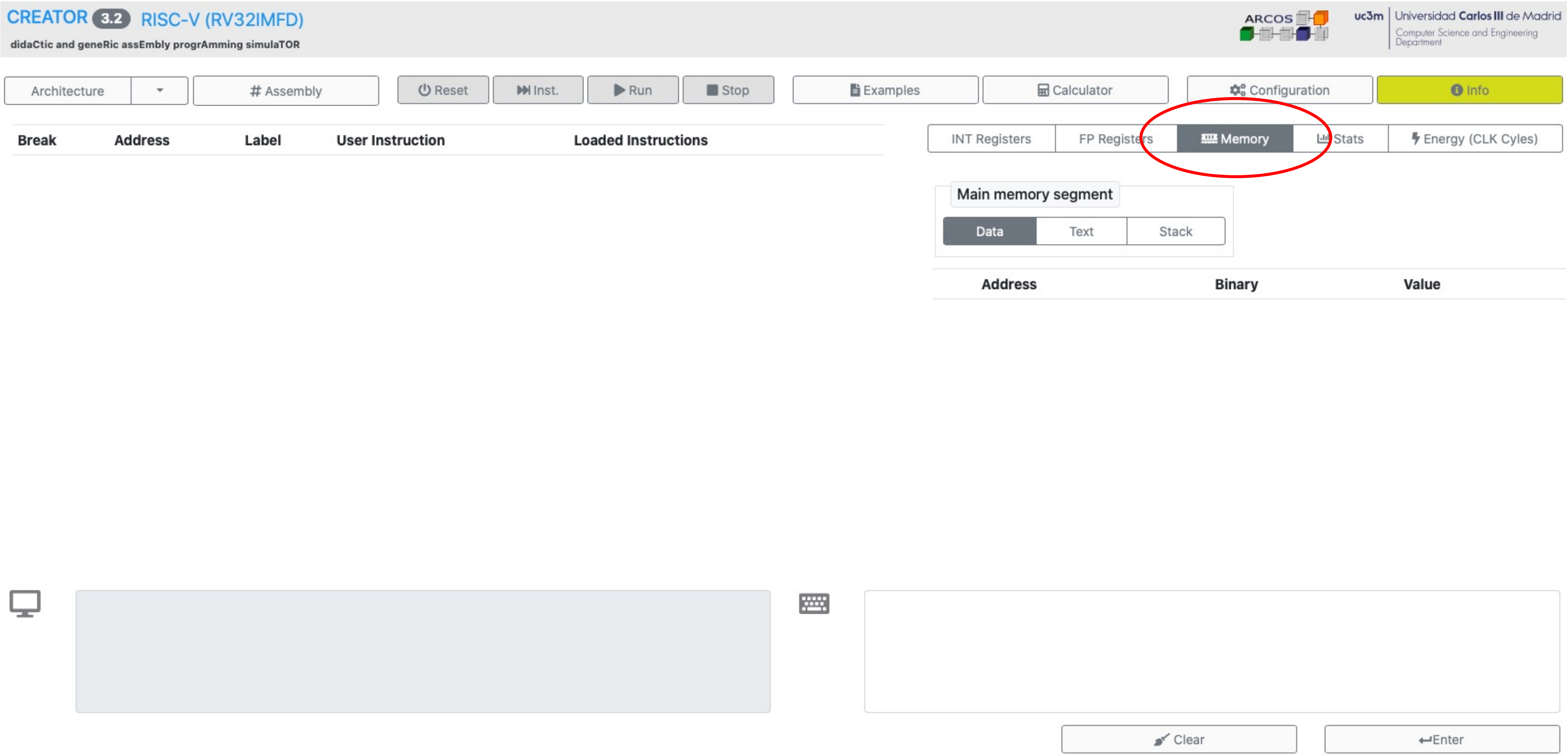
Enter new value | Simple Precision | Update

fs5 | f21

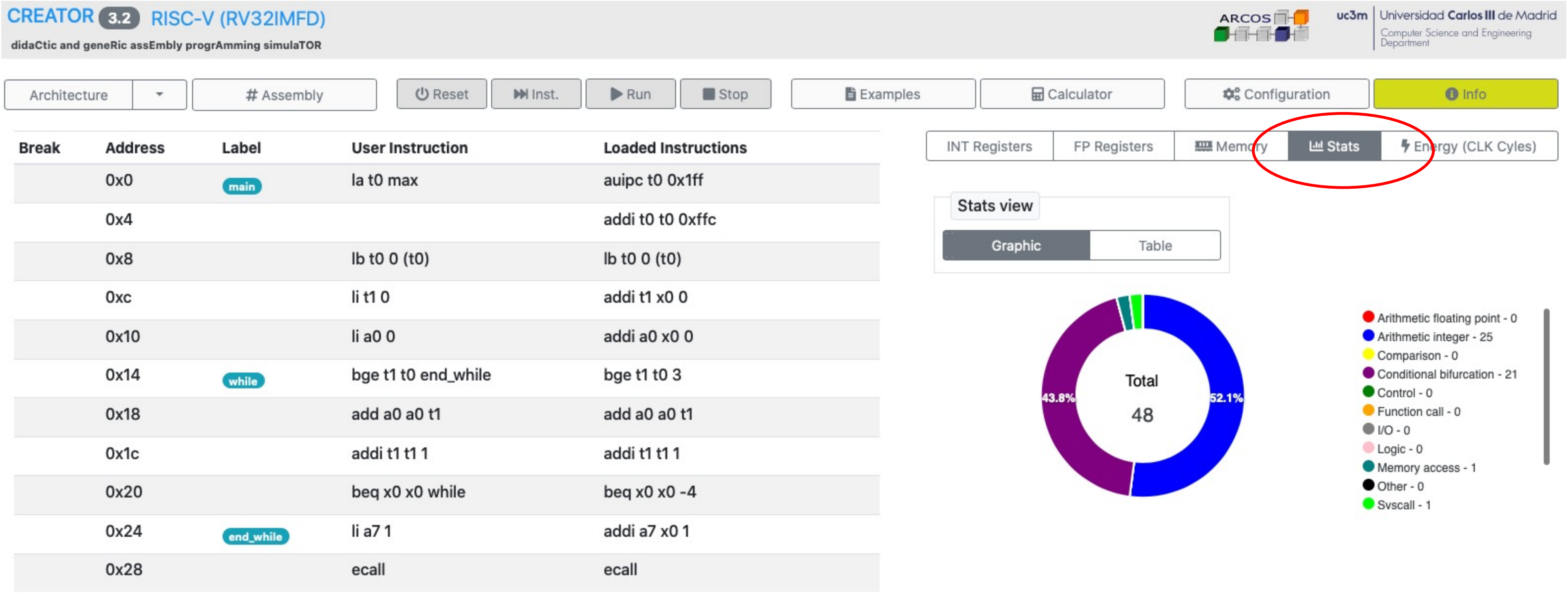
Hex.	C030800000000000
Binary	1100000000110000100000000000000000000000000000000000000000000000
IEEE 754 (32 bits)	0
IEEE 754 (64 bits)	-16.5

Enter new value | Double Precision | Update

Contenido de la memoria



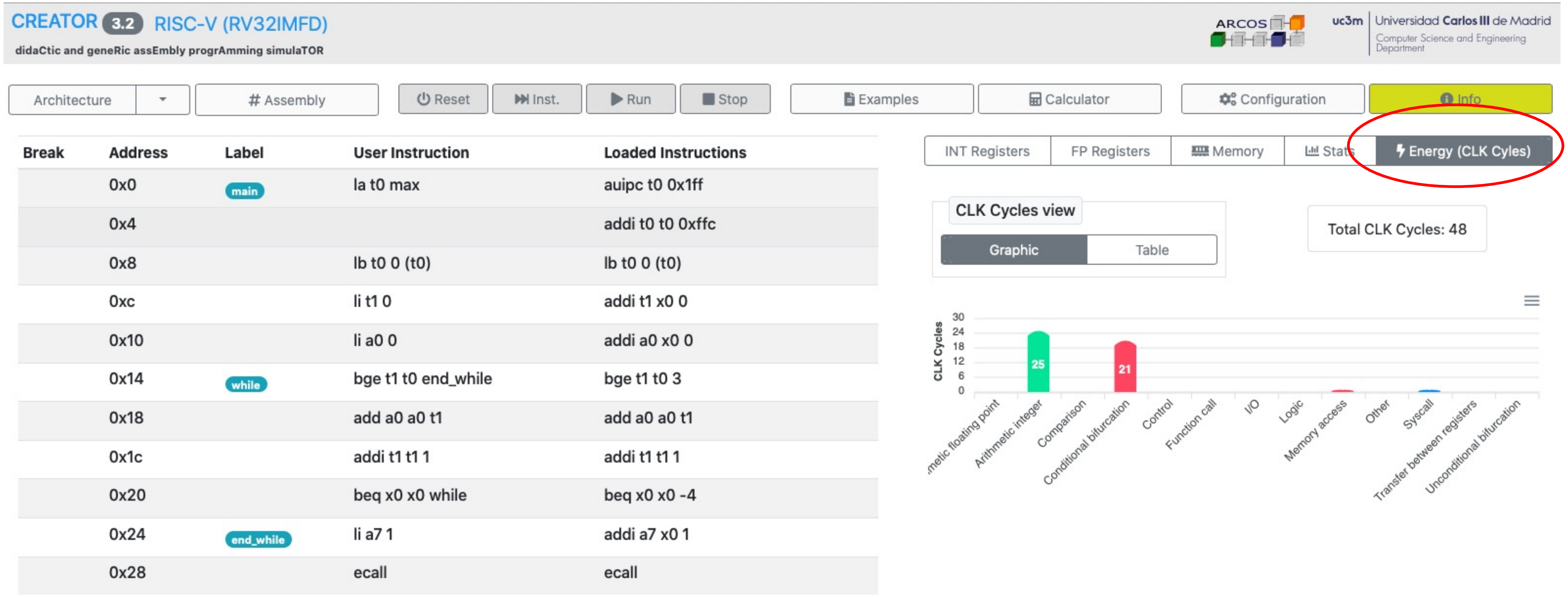
Estadísticas de ejecución



Break	Address	Label	User Instruction	Loaded Instructions
	0x0	main	la t0 max	auipc t0 0x1ff
	0x4			addi t0 t0 0xffc
	0x8		lb t0 0 (t0)	lb t0 0 (t0)
	0xc		li t1 0	addi t1 x0 0
	0x10		li a0 0	addi a0 x0 0
	0x14	while	bge t1 t0 end_while	bge t1 t0 3
	0x18		add a0 a0 t1	add a0 a0 t1
	0x1c		addi t1 t1 1	addi t1 t1 1
	0x20		beq x0 x0 while	beq x0 x0 -4
	0x24	end_while	li a7 1	addi a7 x0 1
	0x28		ecall	ecall

- ● Arithmetic floating point - 0
- ● Arithmetic integer - 25
- ● Comparison - 0
- ● Conditional bifurcation - 21
- ● Control - 0
- ● Function call - 0
- ● I/O - 0
- ● Logic - 0
- ● Memory access - 1
- ● Other - 0
- ● Syscall - 1

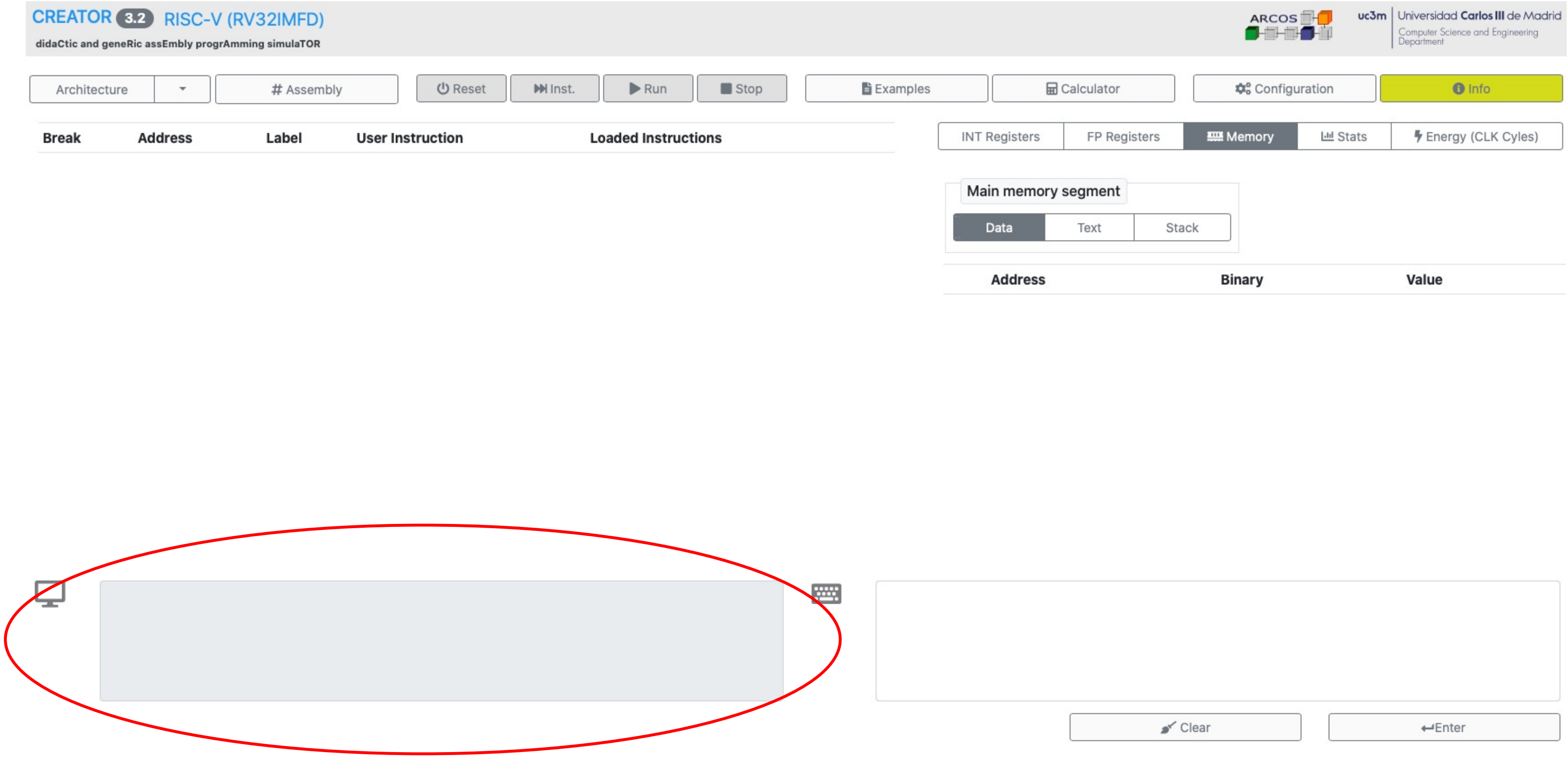
Ciclos ejecutados



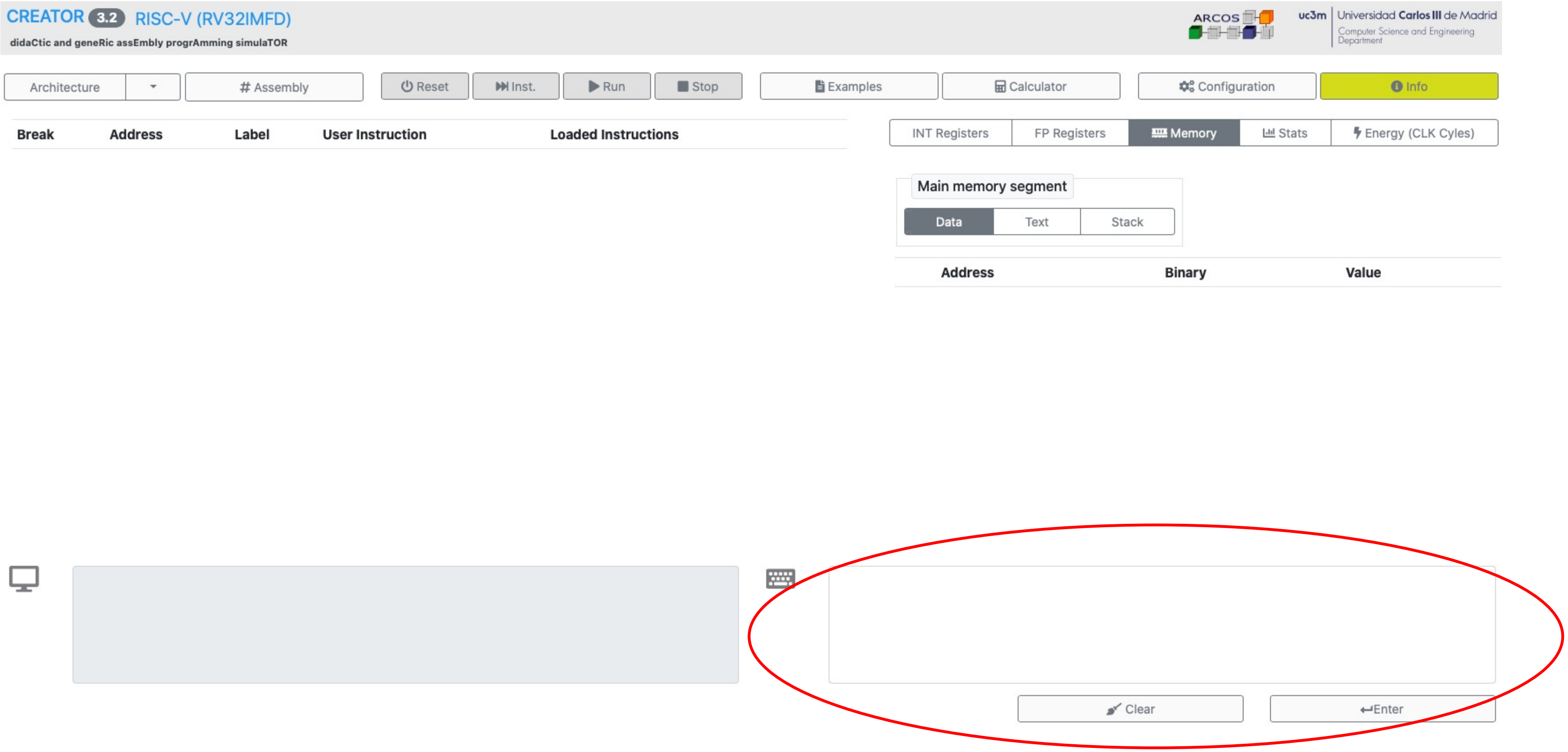
Break	Address	Label	User Instruction	Loaded Instructions
	0x0	main	la t0 max	auipc t0 0x1ff
	0x4			addi t0 t0 0xffc
	0x8		lb t0 0 (t0)	lb t0 0 (t0)
	0xc		li t1 0	addi t1 x0 0
	0x10		li a0 0	addi a0 x0 0
	0x14	while	bge t1 t0 end_while	bge t1 t0 3
	0x18		add a0 a0 t1	add a0 a0 t1
	0x1c		addi t1 t1 1	addi t1 t1 1
	0x20		beq x0 x0 while	beq x0 x0 -4
	0x24	end_while	li a7 1	addi a7 x0 1
	0x28		ecall	ecall

Total CLK Cycles: 48

Pantalla


CREATOR 3.2 RISC-V (RV32IMFD)
didaCtic and geneRic assEmbly progrAmming simulaTOR
ARCOS
uc3m Universidad Carlos III de Madrid
Computer Science and Engineering Department
Architecture
# Assembly
Reset
Inst.
Run
Stop
Examples
Calculator
Configuration
Info
Break
Address
Label
User Instruction
Loaded Instructions
INT Registers
FP Registers
Memory
Stats
Energy (CLK Cyles)
Main memory segment
Data
Text
Stack
Address
Binary
Value
Clear
Enter

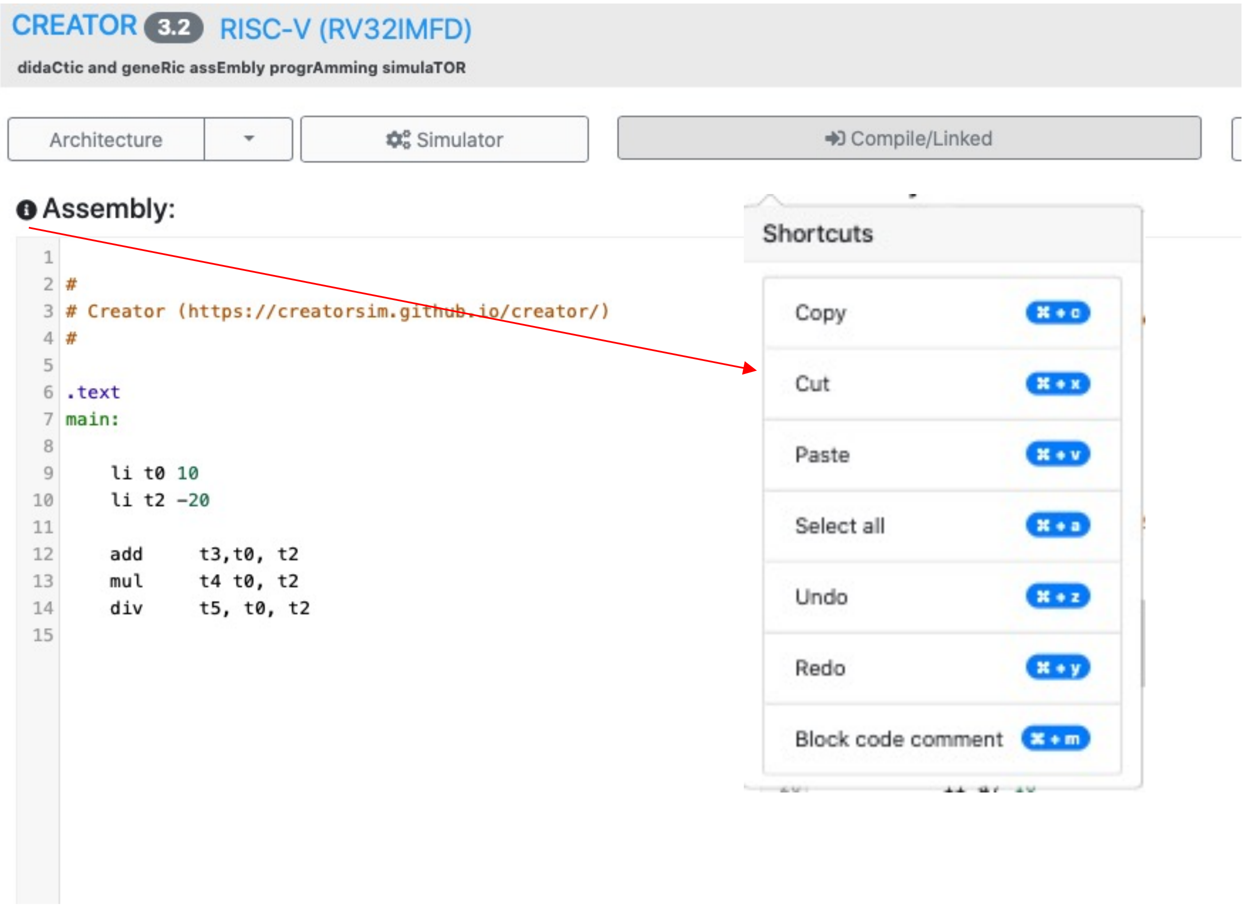
Teclado



CREATOR 3.2 RISC-V (RV32IMFD)
didaCtic and geneRic assEmbly progrAmming simulaTOR
ARCOS
uc3m Universidad Carlos III de Madrid
Computer Science and Engineering Department
Architecture
# Assembly
Reset
Inst.
Run
Stop
Examples
Calculator
Configuration
Info
Break
Address
Label
User Instruction
Loaded Instructions
INT Registers
FP Registers
Memory
Stats
Energy (CLK Cyles)
Main memory segment
Data
Text
Stack
Address
Binary
Value
Clear
↵Enter

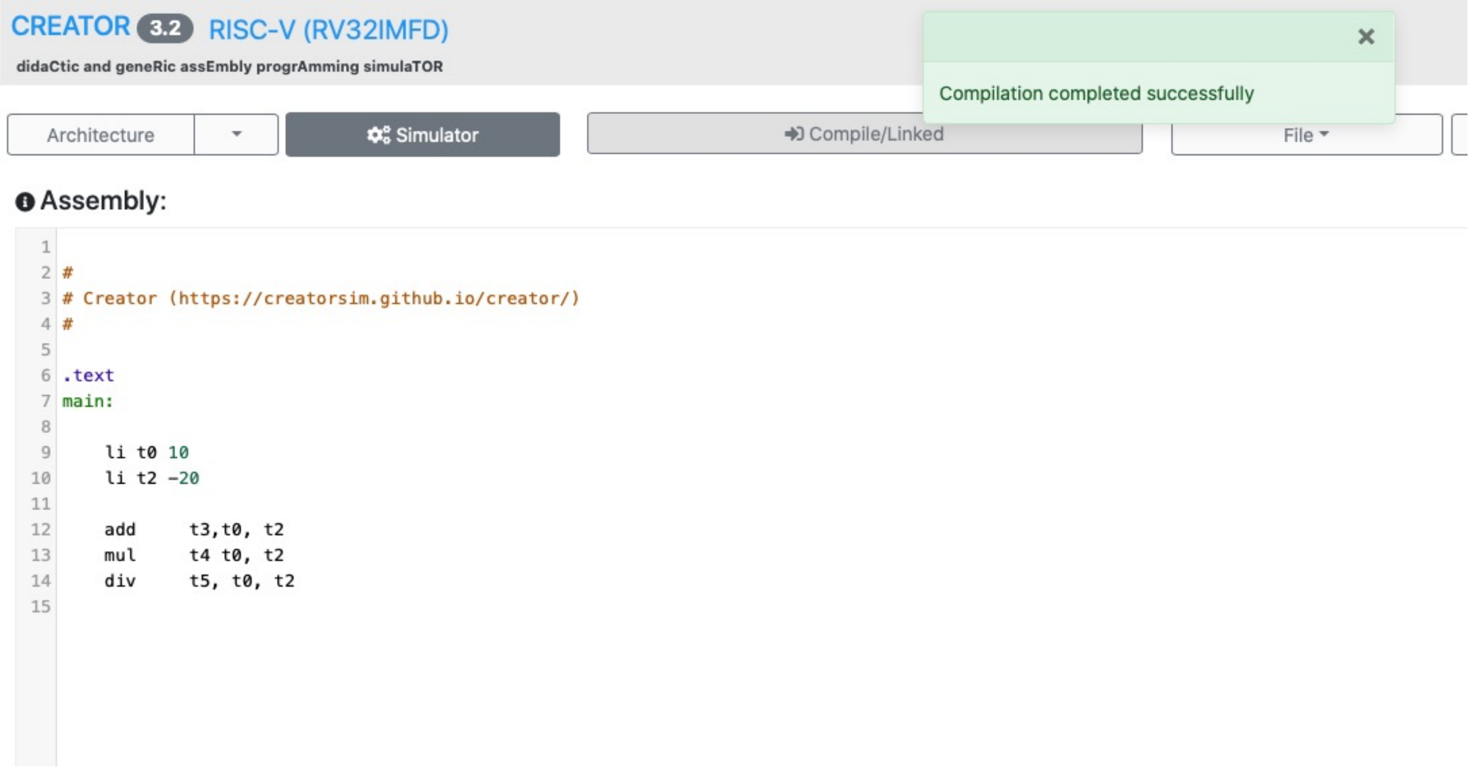
Edición de programas en ensamblador

ejemplo



```

#
# Creator (https://creatorsim.github.io/creator/)
#

.text
main:

    li t0 10
    li t2 -20

    add     t3,t0, t2
    mul     t4 t0, t2
    div     t5, t0, t2

```

Compilación

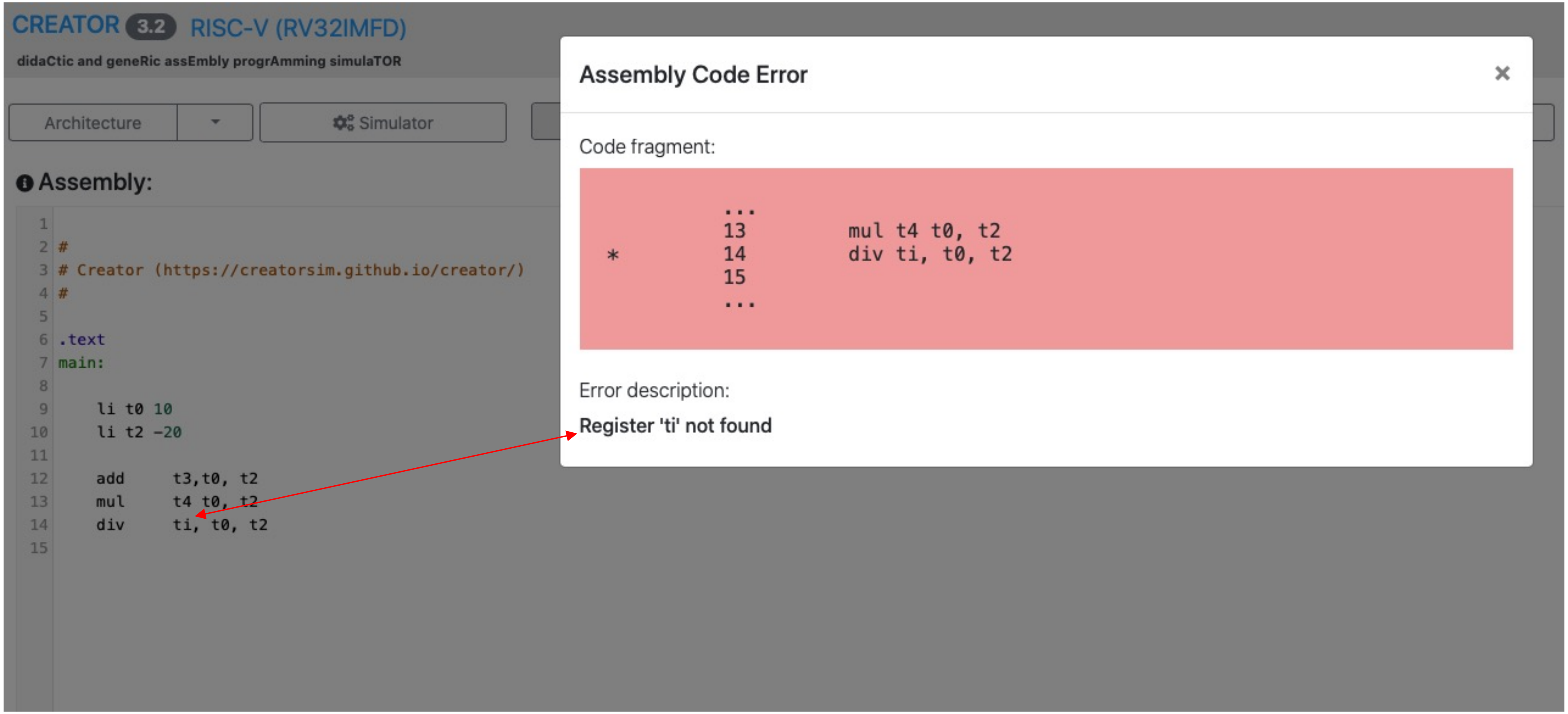


CREATOR 3.2 RISC-V (RV32IMFD)
didaCtic and geneRic assEmbly progrAmming simulaTOR
Architecture
Simulator
Compile/Linked
File
Assembly:

Compilation completed successfully

```

#
# Creator (https://creatorsim.github.io/creator/)
#

.text
main:

    li t0 10
    li t2 -20

    add     t3,t0, t2
    mul     t4 t0, t2
    div     t5, t0, t2

```

Error de compilación



CREATOR 3.2 RISC-V (RV32IMFD)
didaCtic and geneRic assEmbly progrAmming simulaTOR
Architecture
Simulator
Assembly:

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

.text
main:

    li t0 10
    li t2 -20

    add     t3,t0, t2
    mul     t4 t0, t2
    div     ti, t0, t2

```

Assembly Code Error

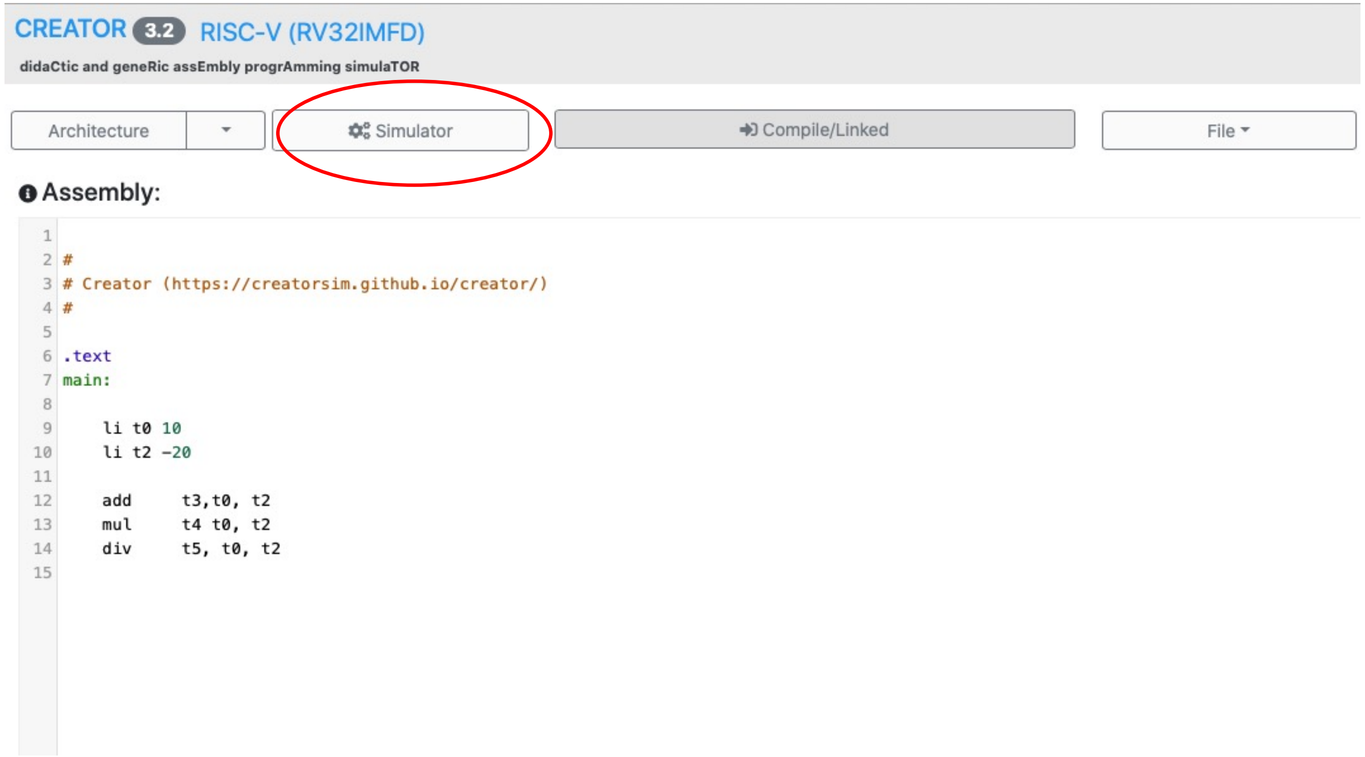
Code fragment:

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

Error description:

Register 'ti' not found

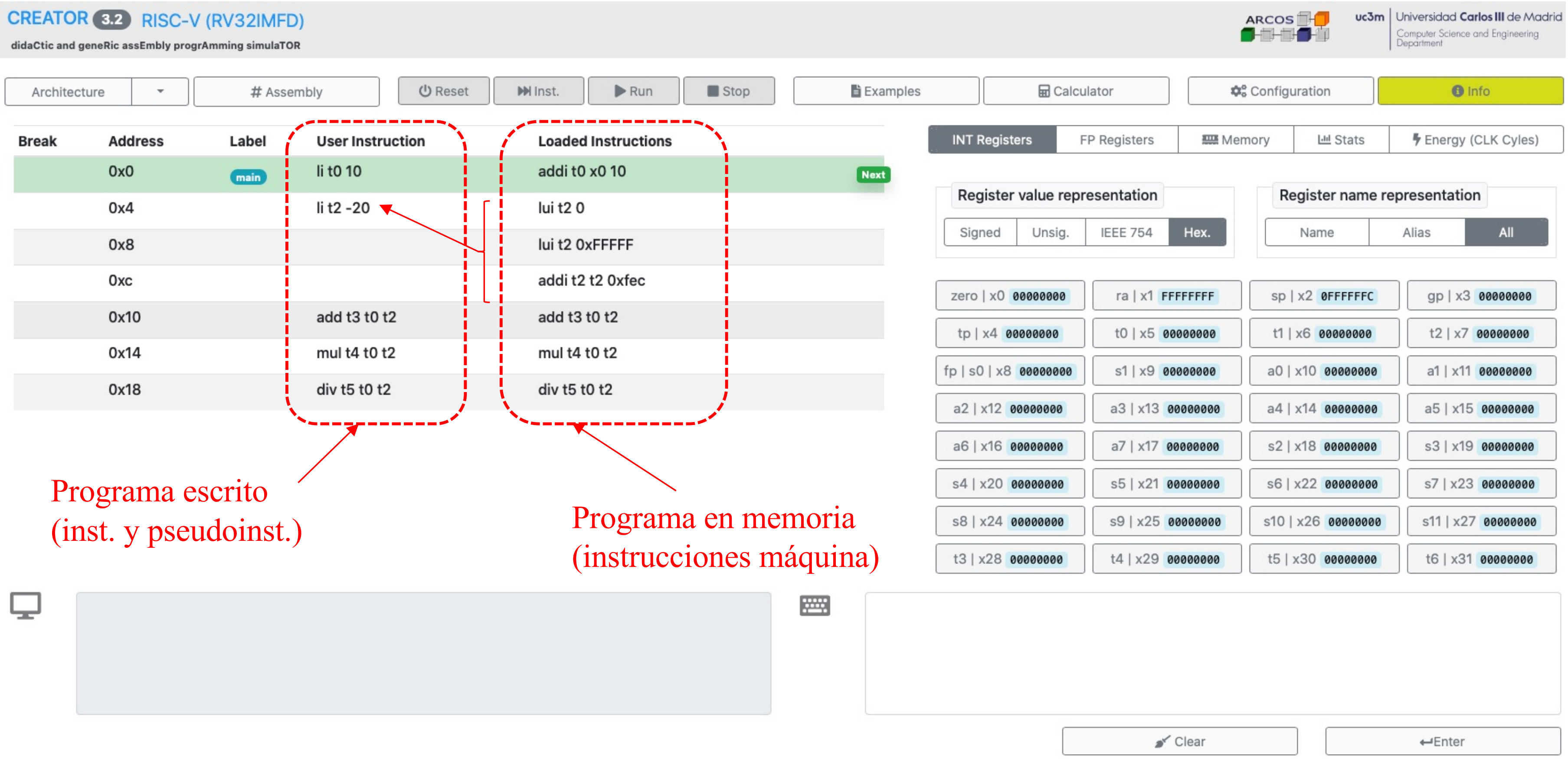
Paso al simulador


CREATOR 3.2 RISC-V (RV32IMFD)
didaCtic and geneRic assEmbly progrAmming simulaTOR
Architecture
Simulator
Compile/Linked
File
Assembly:

```

#
# Creator (https://creatorsim.github.io/creator/)
#

.text
main:

    li t0 10
    li t2 -20

    add     t3,t0, t2
    mul     t4 t0, t2
    div     t5, t0, t2

```

Simulador

ejemplo



CREATOR 3.2 RISC-V (RV32IMFD)
didaCtic and geneRic assEmbly progrAmming simulaTOR
ARCOS
uc3m Universidad Carlos III de Madrid
Computer Science and Engineering Department
Architecture
# Assembly
Reset
Inst.
Run
Stop
Examples
Calculator
Configuration
Info
INT Registers
FP Registers
Memory
Stats
Energy (CLK Cyles)
Next
main
Clear
Enter

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

Register value representation: Signed | Unsig. | IEEE 754 | **Hex.**

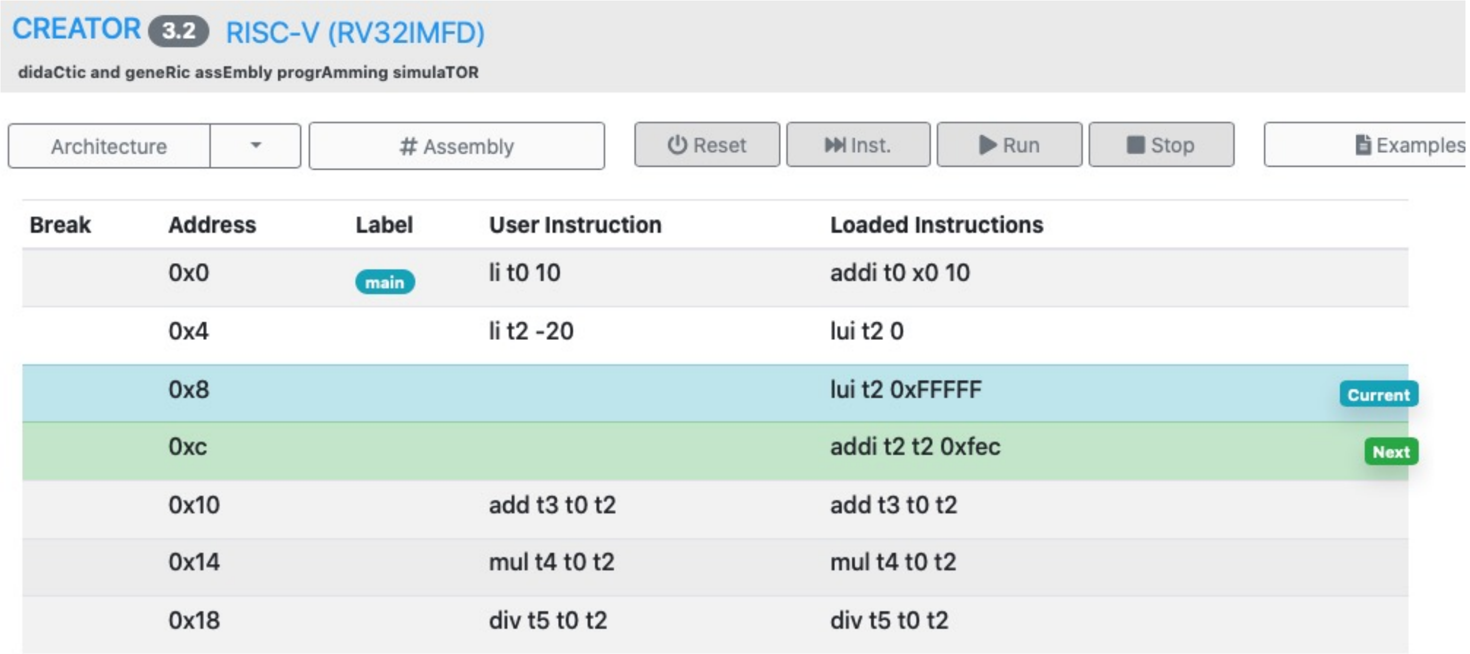
Register name representation: Name | Alias | **All**

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

Programa escrito
(inst. y pseudoinst.)

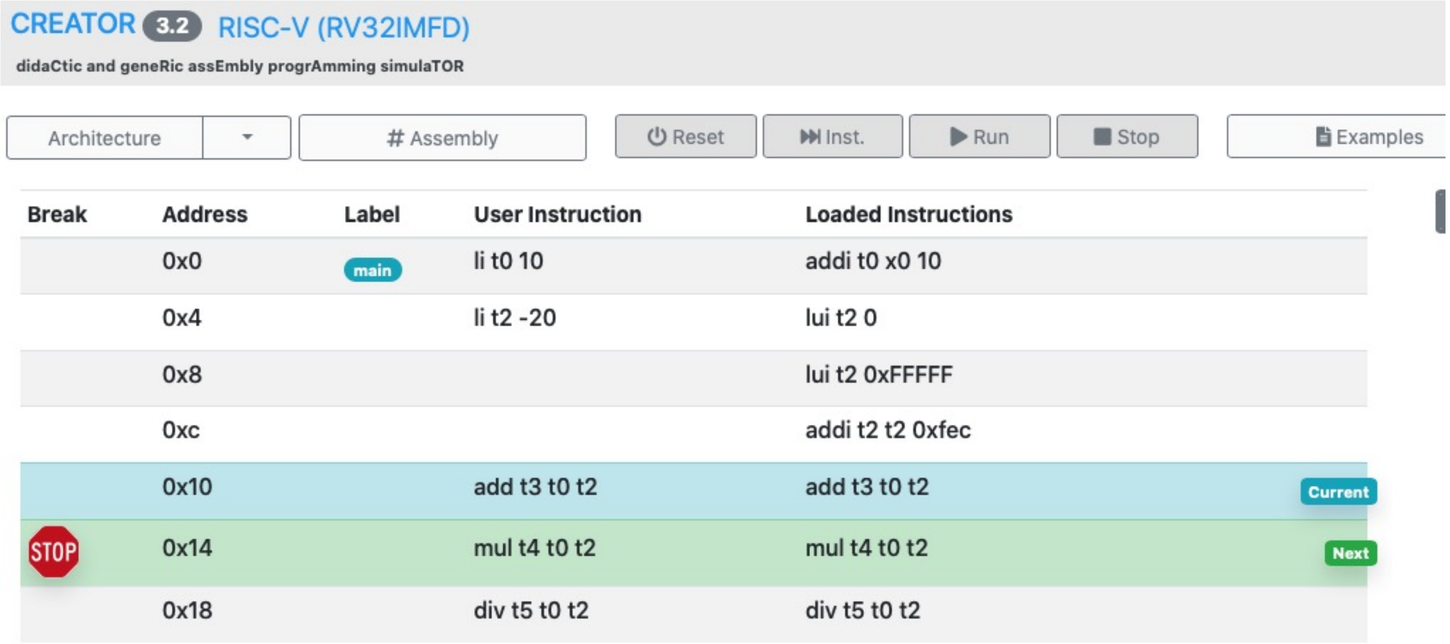
Programa en memoria
(instrucciones máquina)

Flujo de ejecución


CREATOR 3.2 RISC-V (RV32IMFD)
didaCtic and geneRic assEmbly progrAmming simulaTOR
Architecture
# Assembly
Reset
Inst.
Run
Stop
Examples

Break	Address	Label	User Instruction	Loaded Instructions	
	0x0	main	li t0 10	addi t0 x0 10	
	0x4		li t2 -20	lui t2 0	
	0x8			lui t2 0xFFFFF	Current
	0xc			addi t2 t2 0xfec	Next
	0x10		add t3 t0 t2	add t3 t0 t2	
	0x14		mul t4 t0 t2	mul t4 t0 t2	
	0x18		div t5 t0 t2	div t5 t0 t2	

Puntos de ruptura

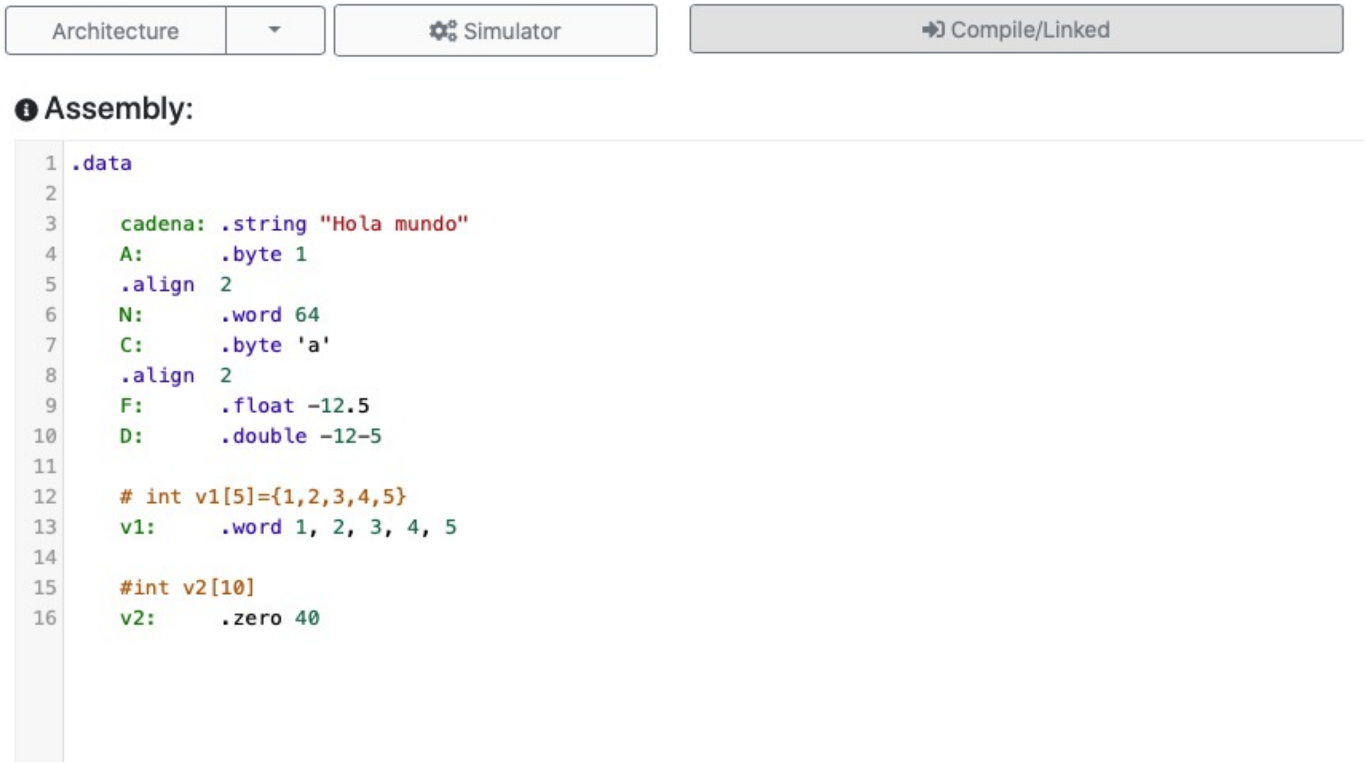


CREATOR 3.2 RISC-V (RV32IMFD)
didaCtic and geneRic assEmbly progrAmming simulaTOR
Architecture
# Assembly
Reset
Inst.
Run
Stop
Examples

Break	Address	Label	User Instruction	Loaded Instructions	
	0x0	main	li t0 10	addi t0 x0 10	
	0x4		li t2 -20	lui t2 0	
	0x8			lui t2 0xFFFFF	
	0xc			addi t2 t2 0xfec	
	0x10		add t3 t0 t2	add t3 t0 t2	Current
STOP	0x14		mul t4 t0 t2	mul t4 t0 t2	Next
	0x18		div t5 t0 t2	div t5 t0 t2	

Segmento de datos

ejemplo

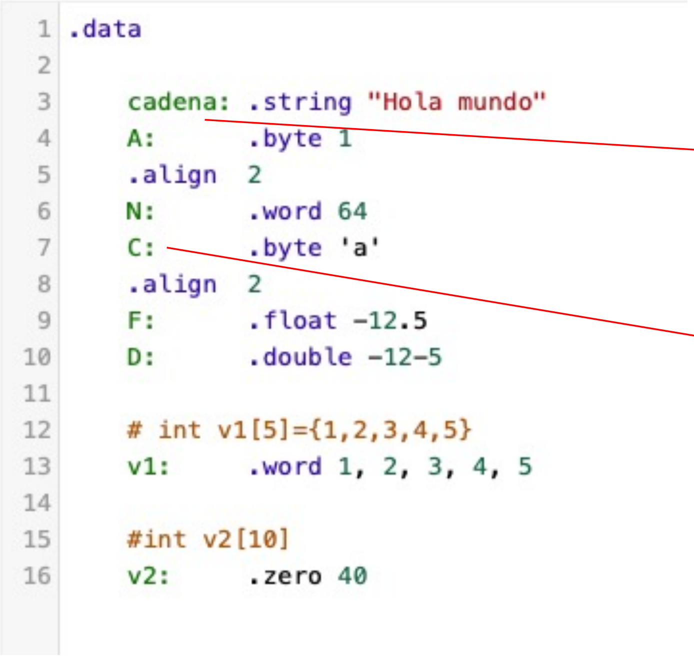


```
.data

    cadena: .string "Hola mundo"
    A:      .byte 1
    .align  2
    N:      .word 64
    C:      .byte 'a'
    .align  2
    F:      .float -12.5
    D:      .double -12-5

    # int v1[5]={1,2,3,4,5}
    v1:     .word 1, 2, 3, 4, 5

    #int v2[10]
    v2:     .zero 40
```

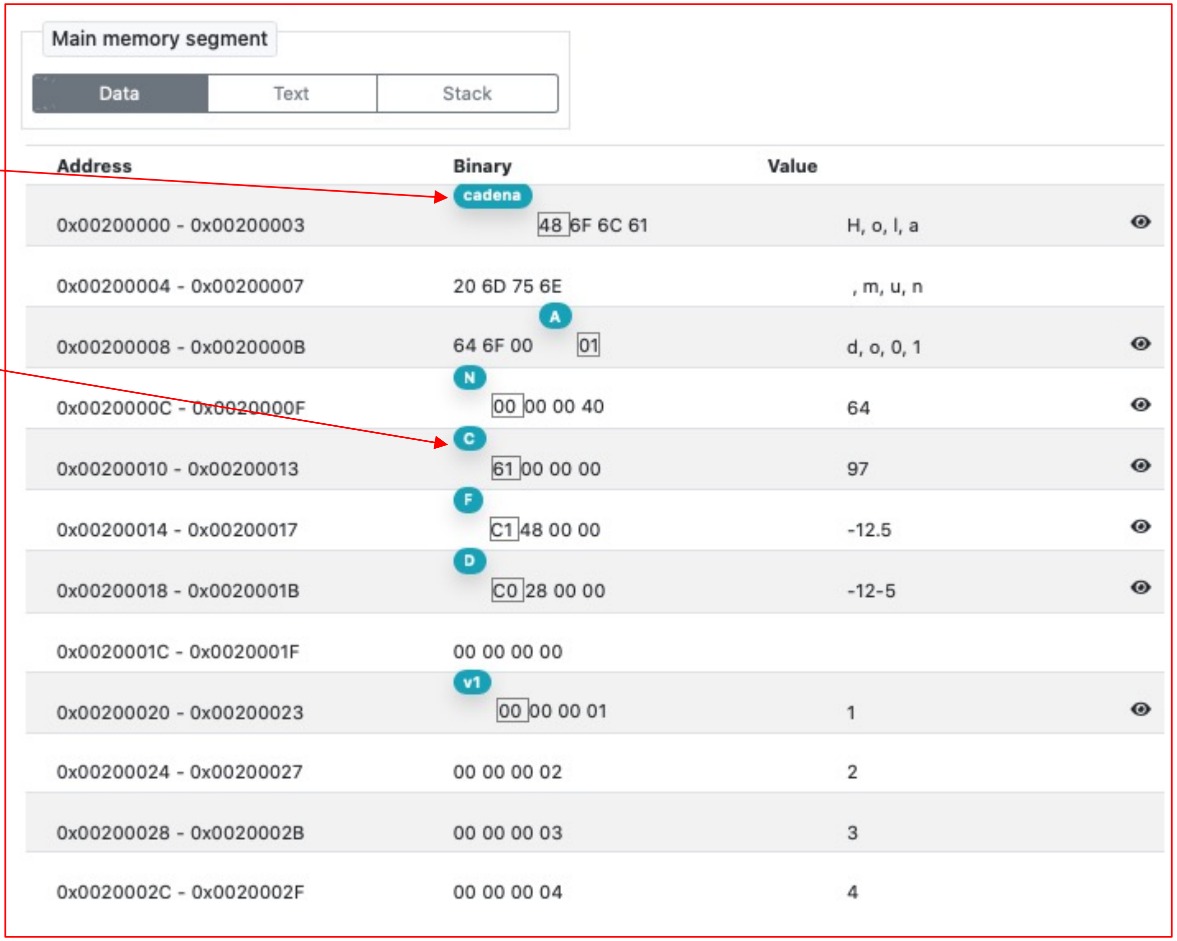
Visualización de datos en memoria

ejemplo

Assembly:

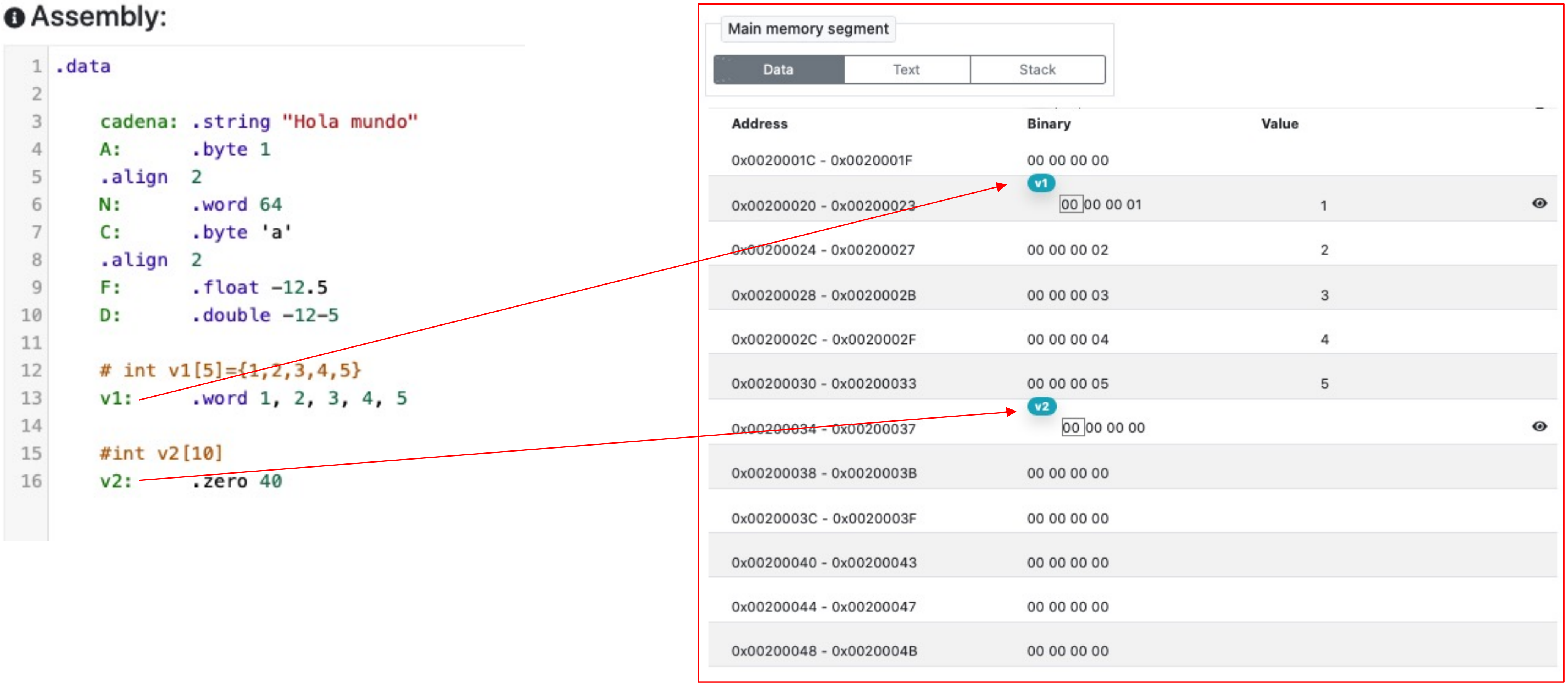


```
.data

    cadena: .string "Hola mundo"
    A:      .byte 1
    .align  2
    N:      .word 64
    C:      .byte 'a'
    .align  2
    F:      .float -12.5
    D:      .double -12-5

    # int v1[5]={1,2,3,4,5}
    v1:     .word 1, 2, 3, 4, 5

    #int v2[10]
    v2:     .zero 40
```



Address	Binary	Value
0x00200000 - 0x00200003	cadena 48 6F 6C 61	H, o, l, a
0x00200004 - 0x00200007	20 6D 75 6E	, m, u, n
0x00200008 - 0x0020000B	A 64 6F 00 01	d, o, 0, 1
0x0020000C - 0x0020000F	N 00 00 00 40	64
0x00200010 - 0x00200013	C 61 00 00 00	97
0x00200014 - 0x00200017	F C1 48 00 00	-12.5
0x00200018 - 0x0020001B	D C0 28 00 00	-12-5
0x0020001C - 0x0020001F	00 00 00 00	
0x00200020 - 0x00200023	v1 00 00 00 01	1
0x00200024 - 0x00200027	00 00 00 02	2
0x00200028 - 0x0020002B	00 00 00 03	3
0x0020002C - 0x0020002F	00 00 00 04	4

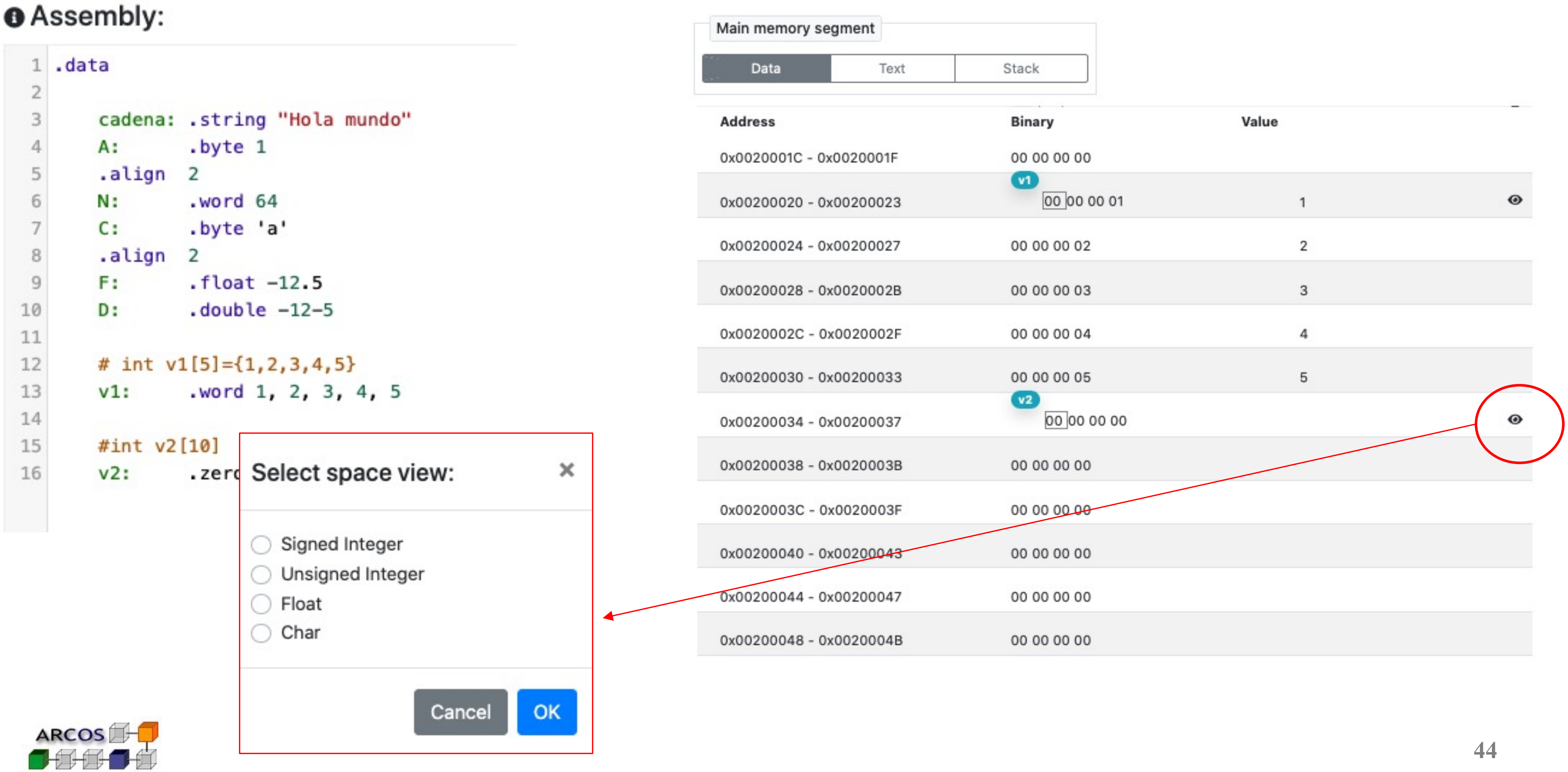
Visualización de datos en memoria



```
.data

    cadena: .string "Hola mundo"
    A:      .byte 1
    .align  2
    N:      .word 64
    C:      .byte 'a'
    .align  2
    F:      .float -12.5
    D:      .double -12-5

    # int v1[5]={1,2,3,4,5}
    v1:     .word 1, 2, 3, 4, 5

    #int v2[10]
    v2:     .zero 40
```

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

Visualización de datos en memoria



Assembly:

```
.data

    cadena: .string "Hola mundo"
    A:      .byte 1
    .align  2
    N:      .word 64
    C:      .byte 'a'
    .align  2
    F:      .float -12.5
    D:      .double -12-5

    # int v1[5]={1,2,3,4,5}
    v1:     .word 1, 2, 3, 4, 5

    #int v2[10]
    v2:     .zer
```

Select space view: ×

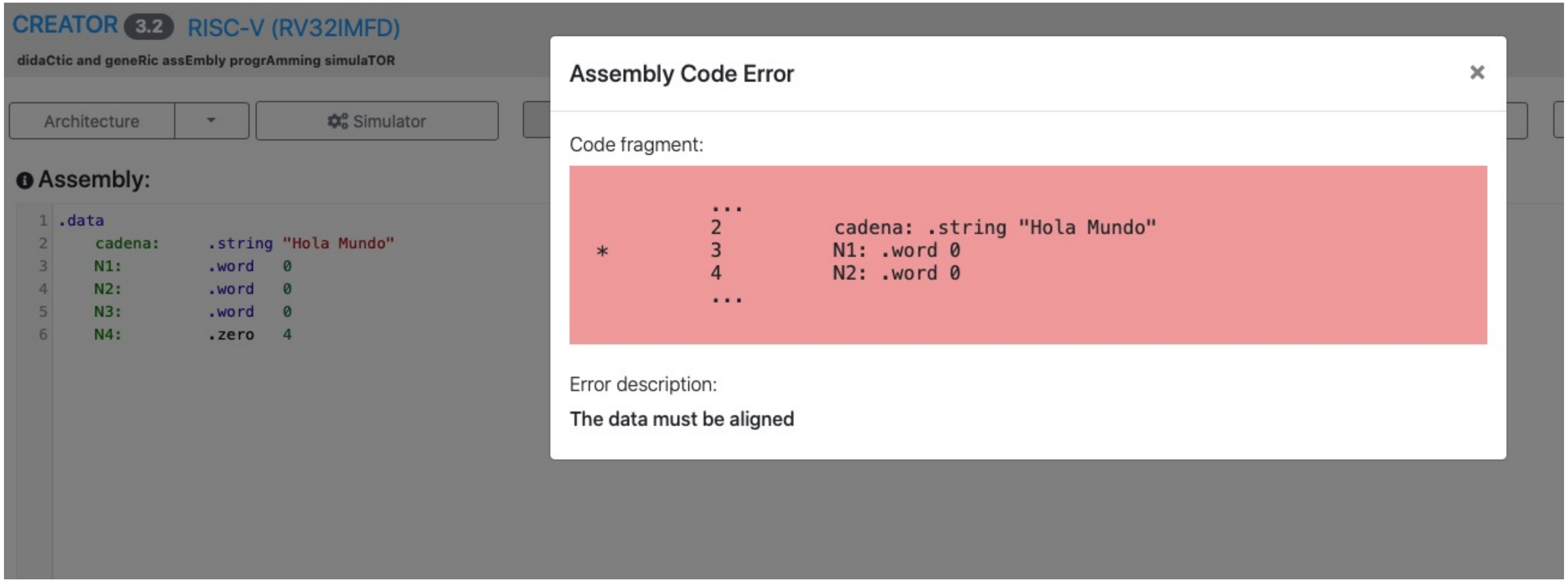
- ○ Signed Integer
- ○ Unsigned Integer
- ○ Float
- ○ Char

Main memory segment

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

Detección de datos no alineados

ejemplo



CREATOR 3.2 RISC-V (RV32IMFD)
didaCtic and geneRic assEmbly progrAmming simulaTOR
Architecture
Simulator
Assembly:

```
.data
    cadena:      .string "Hola Mundo"
    N1:          .word   0
    N2:          .word   0
    N3:          .word   0
    N4:          .zero   4
```

Assembly Code Error

Code fragment:

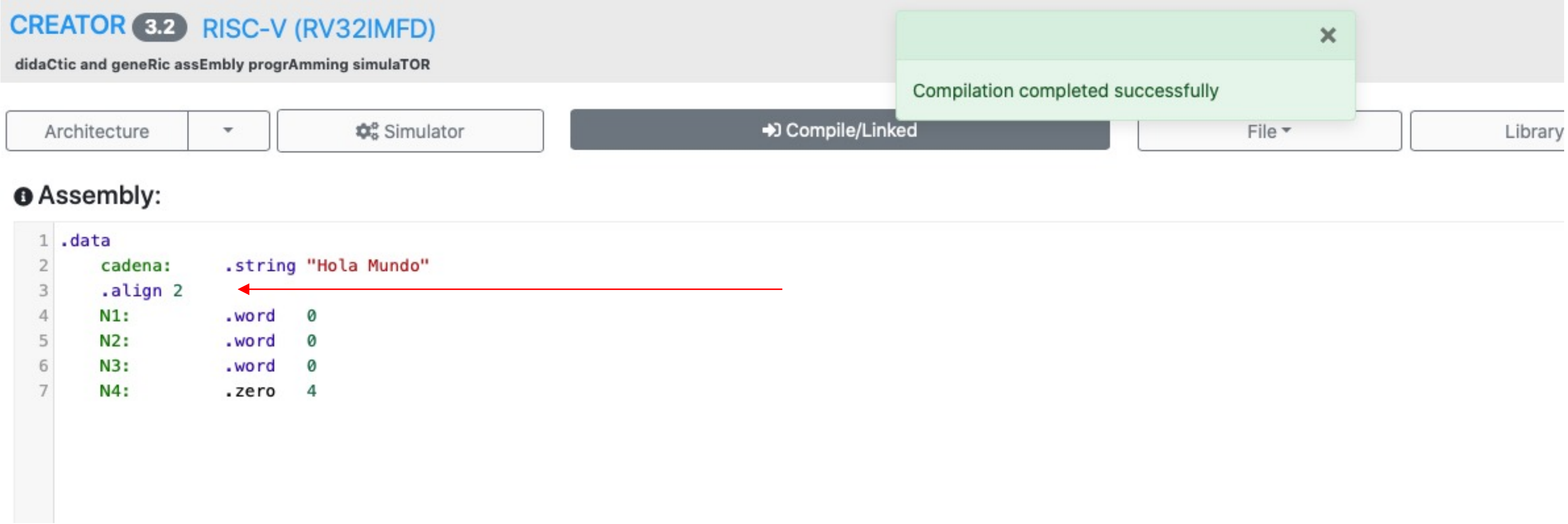
```
           ...
           2        cadena: .string "Hola Mundo"
  *        3        N1: .word 0
           4        N2: .word 0
           ...
```

Error description:

The data must be aligned

Segmento de datos corregido

ejemplo

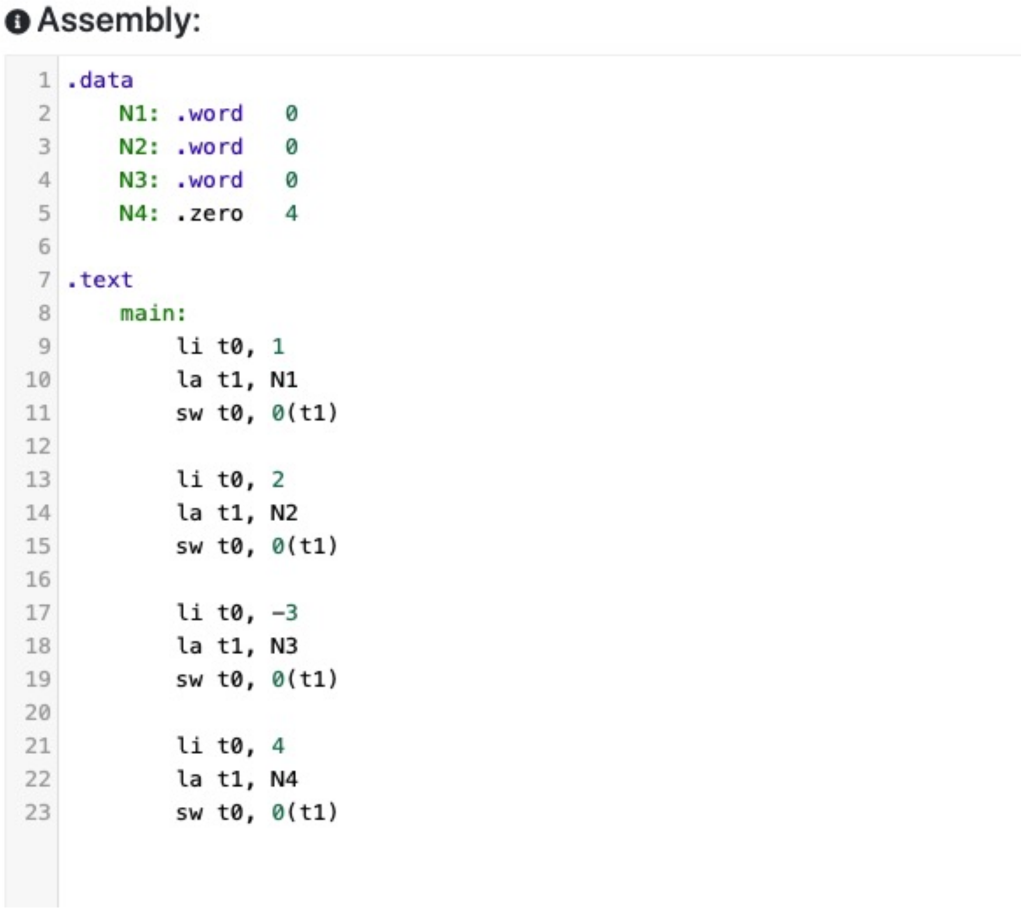

CREATOR 3.2 RISC-V (RV32IMFD)
didaCtic and geneRic assEmbly progrAmming simulaTOR
Architecture
Simulator
Compile/Linked
File
Library
Assembly:

Compilation completed successfully

```
.data
    cadena:     .string "Hola Mundo"
    .align 2
    N1:         .word   0
    N2:         .word   0
    N3:         .word   0
    N4:         .zero   4
```

Ejemplo. Ejecutar el siguiente programa

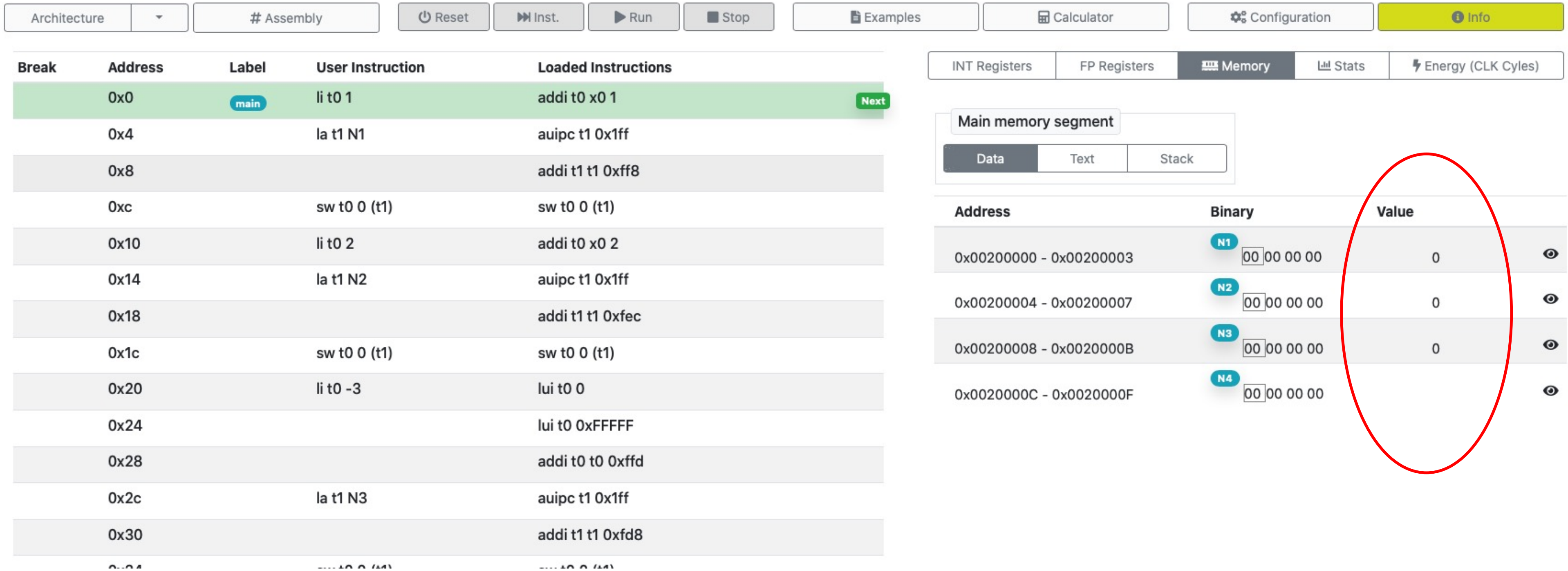
ejemplo



Assembly:

```
.data
    N1: .word   0
    N2: .word   0
    N3: .word   0
    N4: .zero   4

.text
    main:
        li t0, 1
        la t1, N1
        sw t0, 0(t1)

        li t0, 2
        la t1, N2
        sw t0, 0(t1)

        li t0, -3
        la t1, N3
        sw t0, 0(t1)

        li t0, 4
        la t1, N4
        sw t0, 0(t1)
```

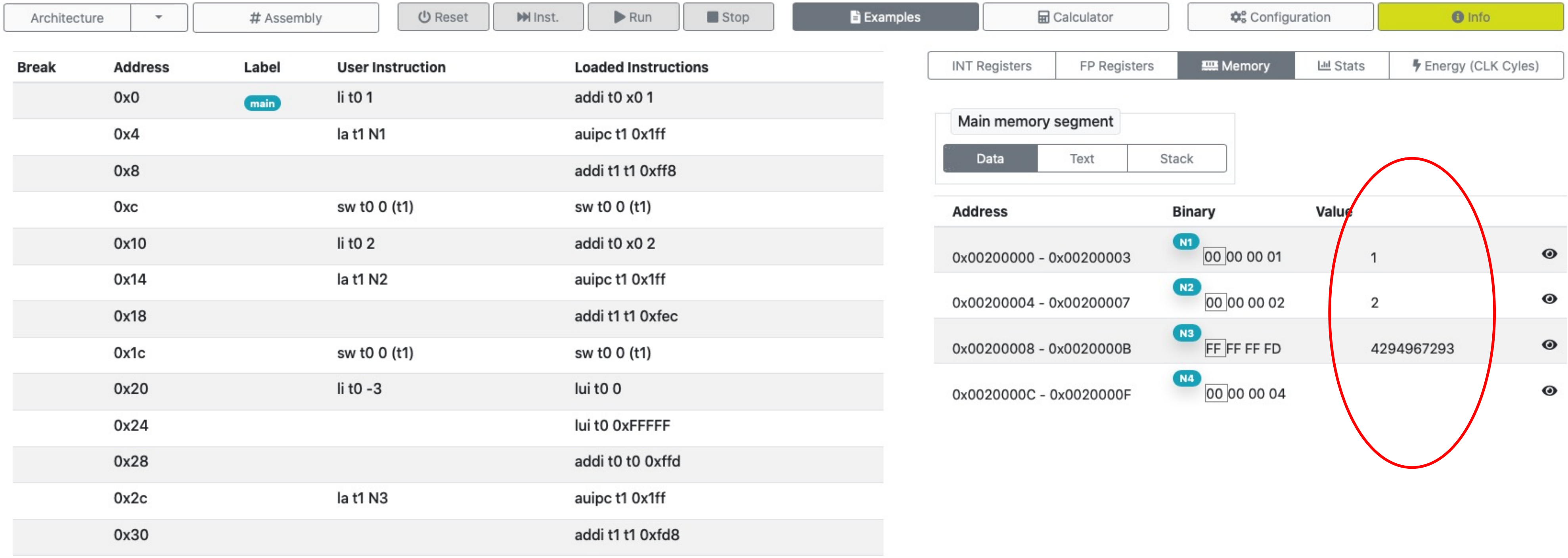
Ejemplo: antes de la ejecución



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

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

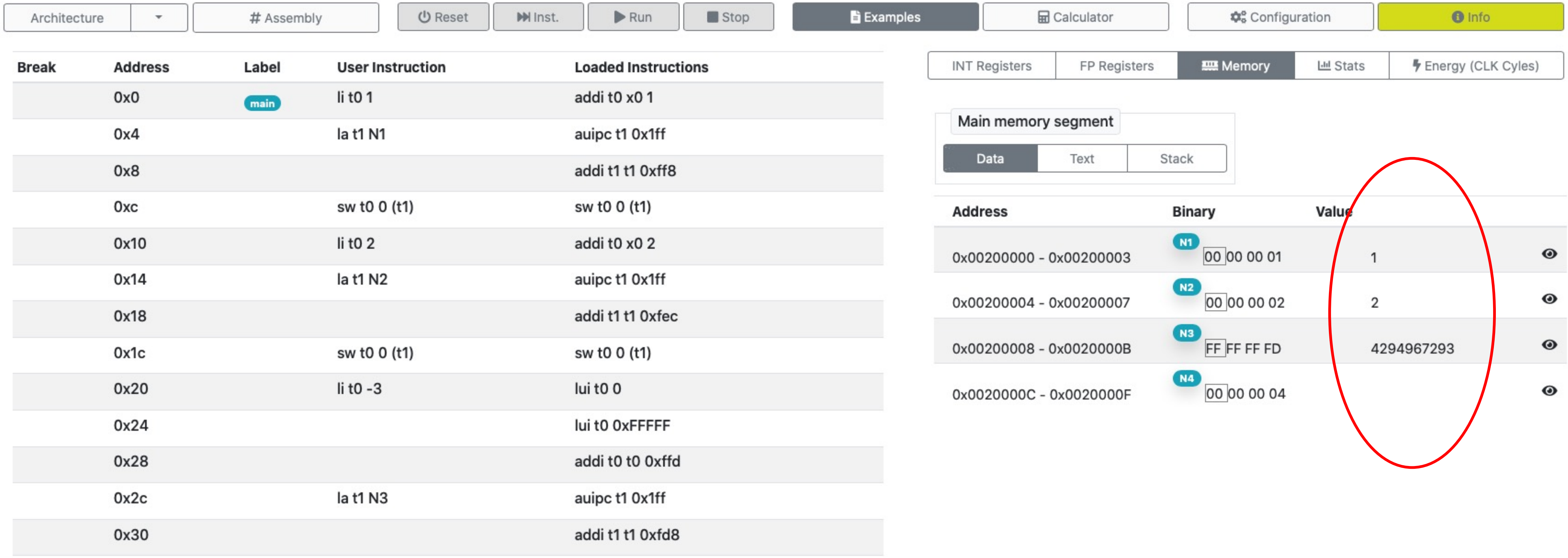
Ejemplo: después de la ejecución



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

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

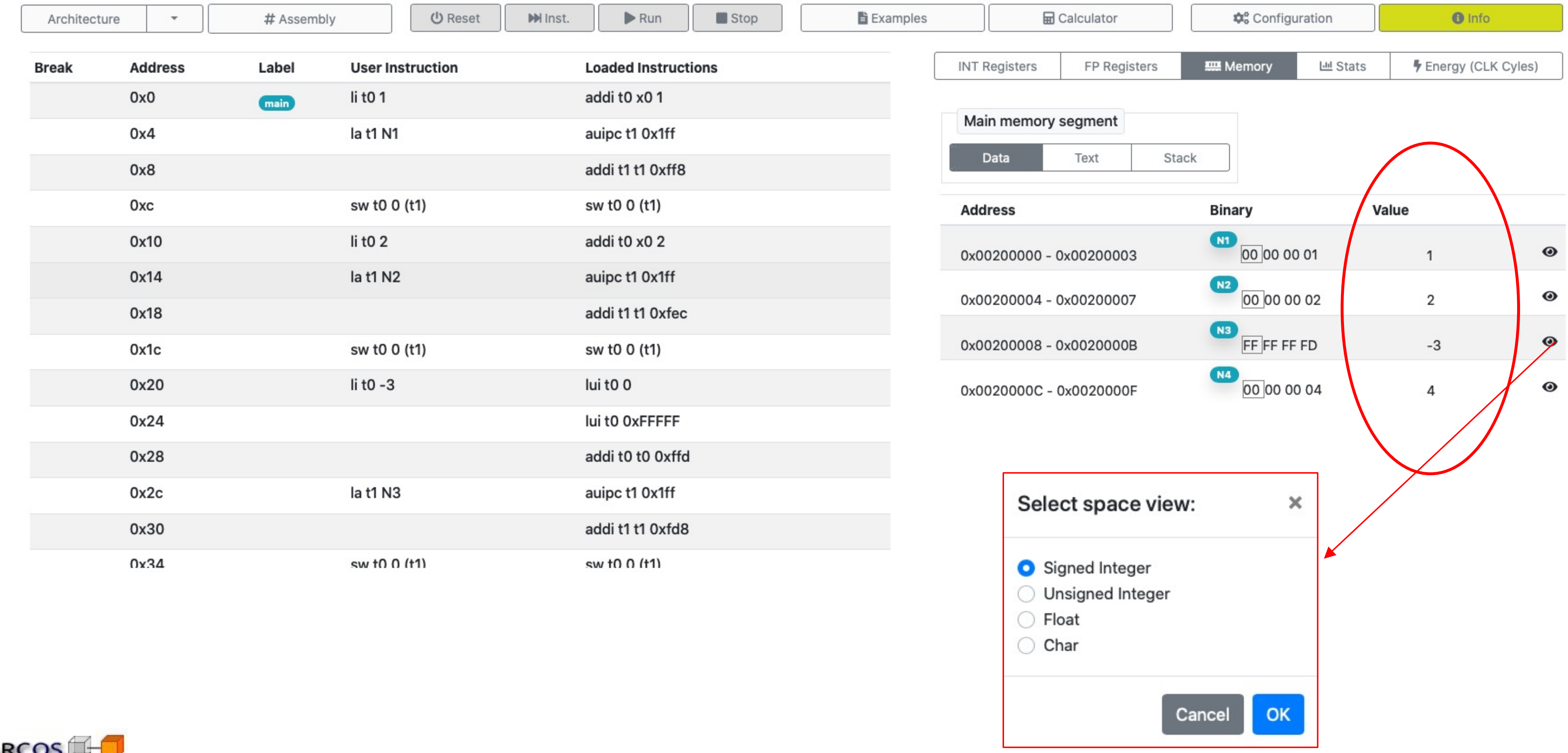
Ejemplo: después de la ejecución



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

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

Ejemplo: después de la ejecución



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

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

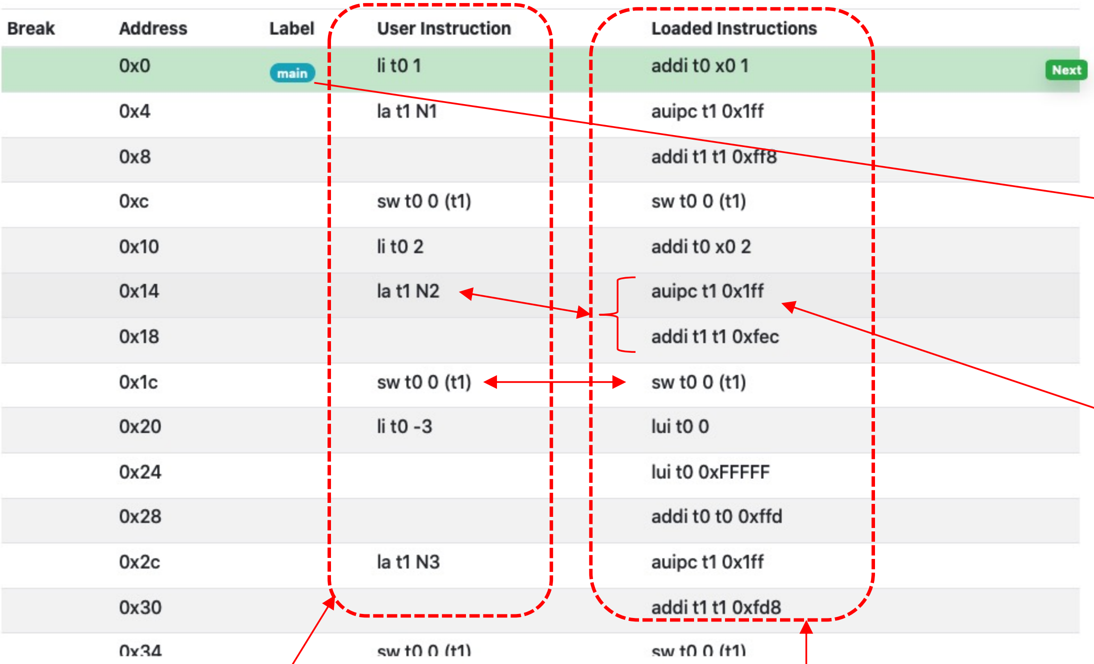
Select space view: ×

● Signed Integer
○ Unsigned Integer
○ Float
○ Char

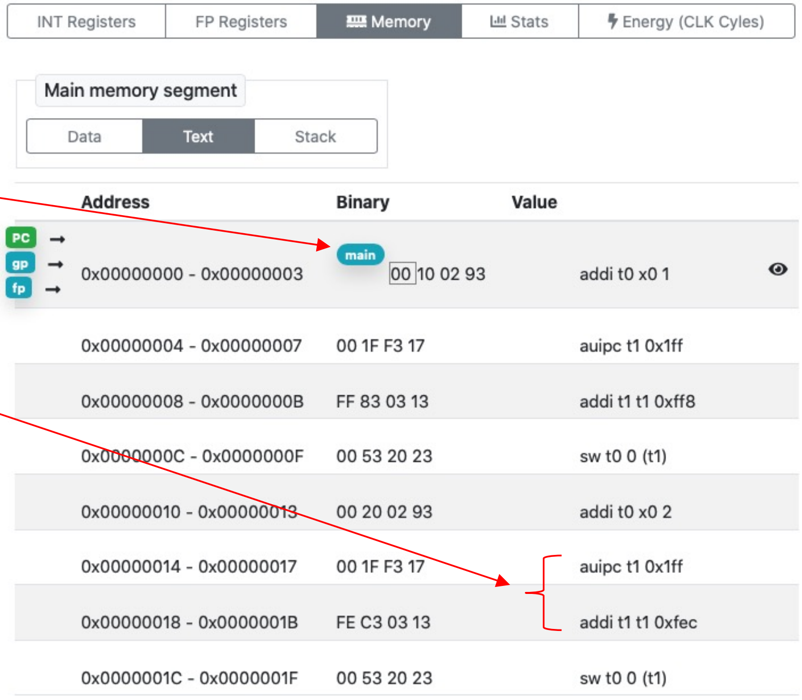
Cancel OK

Visualización del segmento de texto Instrucciones y pseudoinstrucciones

ejemplo



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



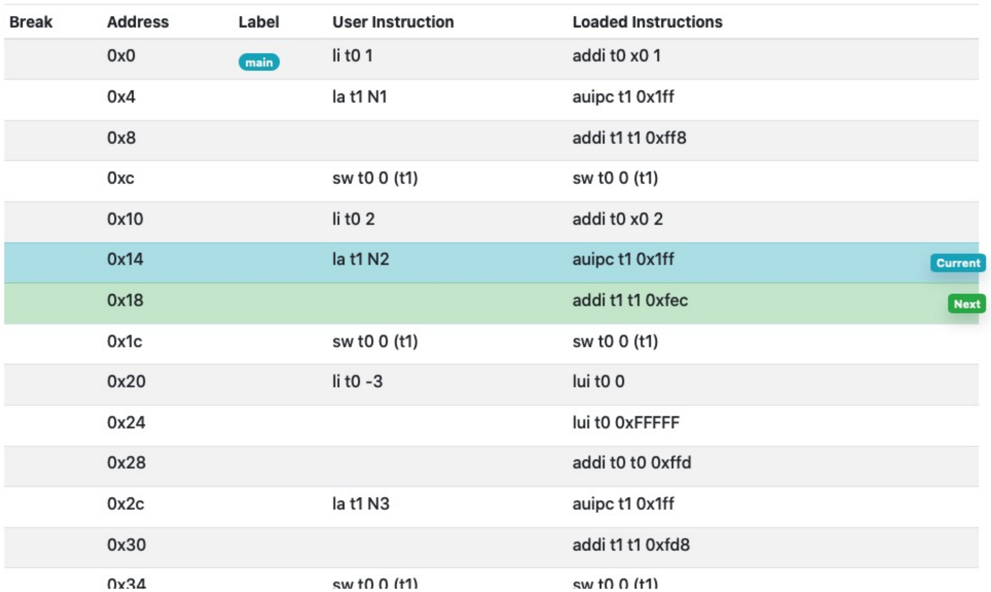
	Address	Binary	Value
PC → gp → fp →	0x00000000 - 0x00000003	main 00 10 02 93	addi t0 x0 1
	0x00000004 - 0x00000007	00 1F F3 17	auipc t1 0x1ff
	0x00000008 - 0x0000000B	FF 83 03 13	addi t1 t1 0xff8
	0x0000000C - 0x0000000F	00 53 20 23	sw t0 0 (t1)
	0x00000010 - 0x00000013	00 20 02 93	addi t0 x0 2
	0x00000014 - 0x00000017	00 1F F3 17	auipc t1 0x1ff
	0x00000018 - 0x0000001B	FE C3 03 13	addi t1 t1 0xfec
	0x0000001C - 0x0000001F	00 53 20 23	sw t0 0 (t1)

Programa escrito
(inst. y pseudoinst.)

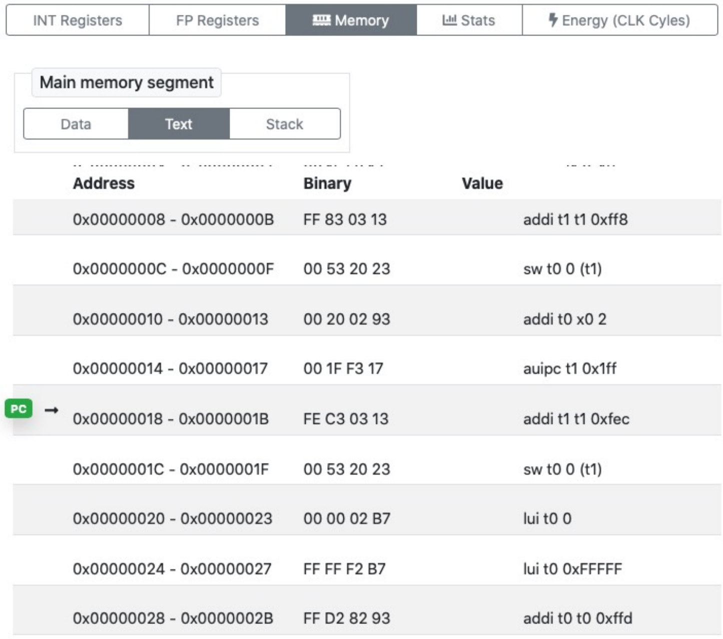
Programa en memoria
(instrucciones máquina)

Visualización del segmento de texto
Flujo de ejecución

ejemplo



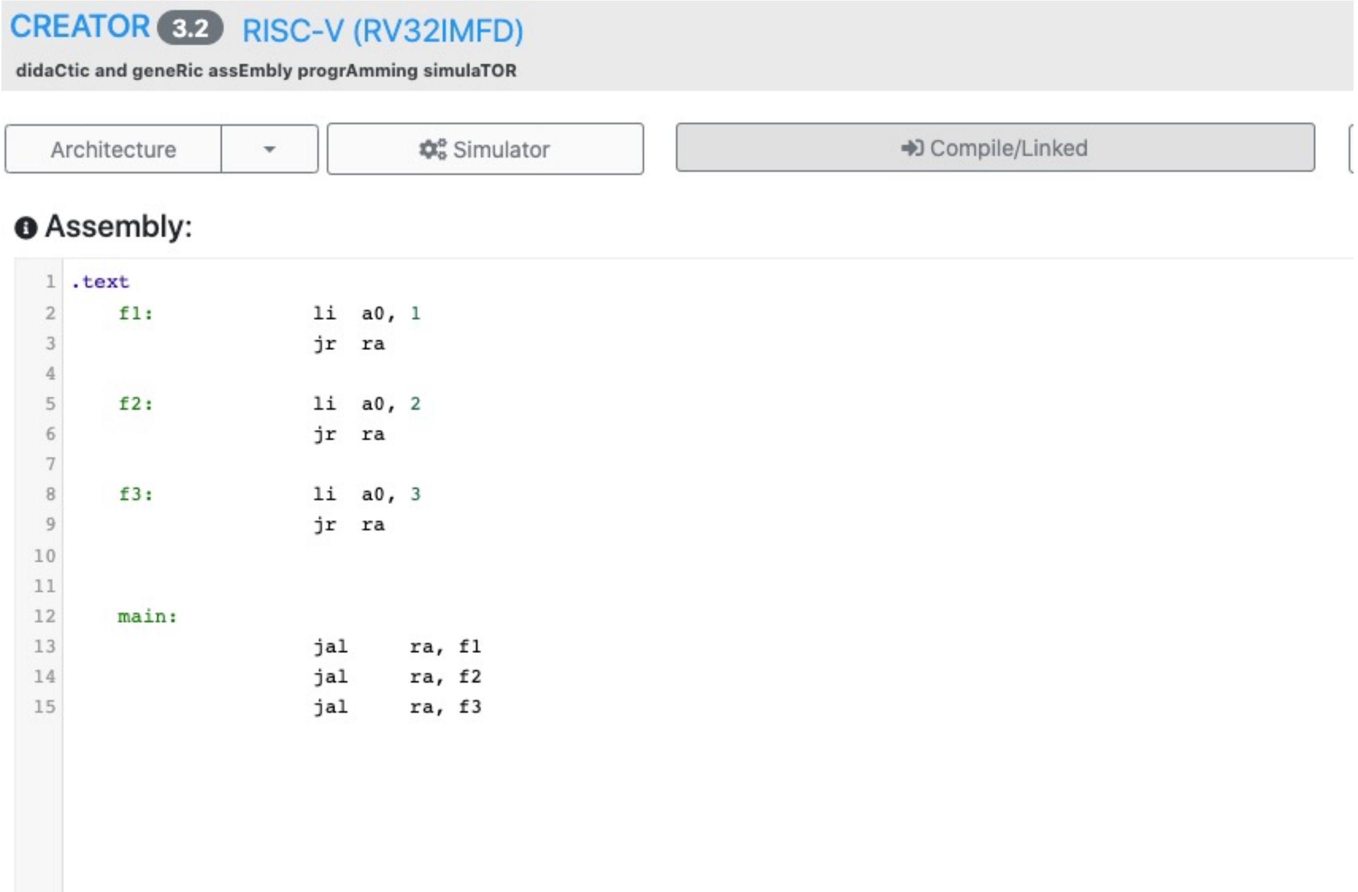
Break	Address	Label	User Instruction	Loaded Instructions	
	0x0	main	li t0 1	addi t0 x0 1	
	0x4		la t1 N1	auipc t1 0x1ff	
	0x8			addi t1 t1 0xff8	
	0xc		sw t0 0 (t1)	sw t0 0 (t1)	
	0x10		li t0 2	addi t0 x0 2	
	0x14		la t1 N2	auipc t1 0x1ff	Current
	0x18			addi t1 t1 0xfec	Next
	0x1c		sw t0 0 (t1)	sw t0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



	Address	Binary	Value
	0x00000008 - 0x0000000B	FF 83 03 13	addi t1 t1 0xff8
	0x0000000C - 0x0000000F	00 53 20 23	sw t0 0 (t1)
	0x00000010 - 0x00000013	00 20 02 93	addi t0 x0 2
	0x00000014 - 0x00000017	00 1F F3 17	auipc t1 0x1ff
PC →	0x00000018 - 0x0000001B	FE C3 03 13	addi t1 t1 0xfec
	0x0000001C - 0x0000001F	00 53 20 23	sw t0 0 (t1)
	0x00000020 - 0x00000023	00 00 02 B7	lui t0 0
	0x00000024 - 0x00000027	FF FF F2 B7	lui t0 0xFFFFF
	0x00000028 - 0x0000002B	FF D2 82 93	addi t0 t0 0xffd

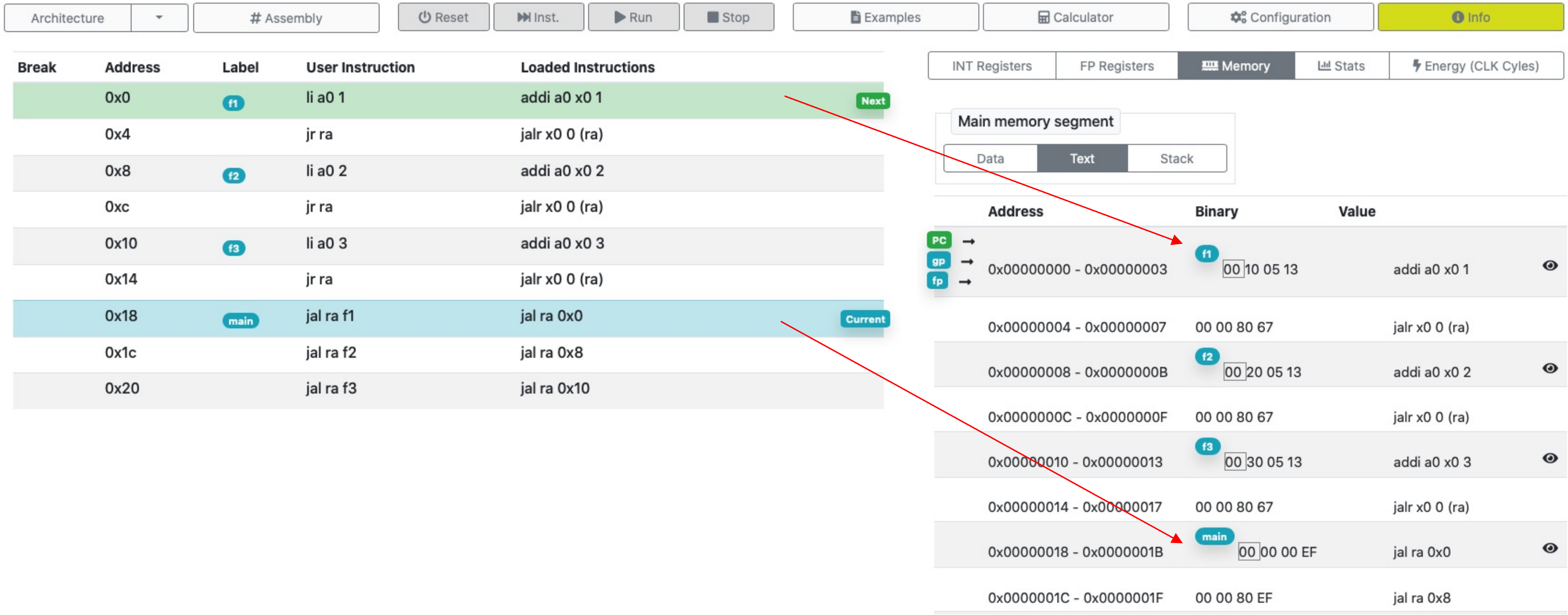
Visualización del segmento de texto
Varias funciones

ejemplo


CREATOR 3.2 RISC-V (RV32IMFD)
didaCtic and geneRic assEmbly progrAmming simulaTOR
Architecture
Simulator
Compile/Linked
Assembly:

```
.text
    f1:             li  a0, 1
                    jr  ra

    f2:             li  a0, 2
                    jr  ra

    f3:             li  a0, 3
                    jr  ra


    main:
                    jal     ra, f1
                    jal     ra, f2
                    jal     ra, f3
```

Visualización del segmento de texto
Varias funciones

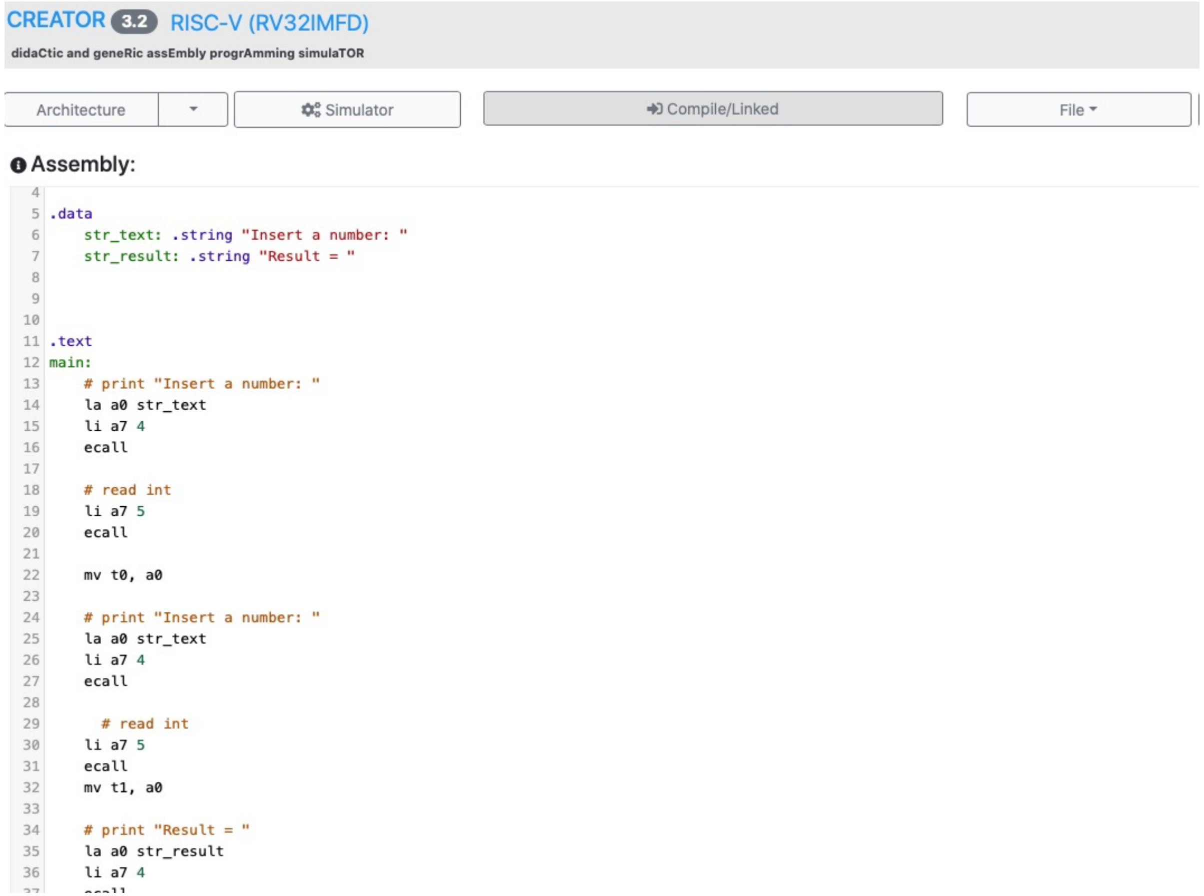


Break	Address	Label	User Instruction	Loaded Instructions
	0x0	f1	li a0 1	addi a0 x0 1
	0x4		jr ra	jalr x0 0 (ra)
	0x8	f2	li a0 2	addi a0 x0 2
	0xc		jr ra	jalr x0 0 (ra)
	0x10	f3	li a0 3	addi a0 x0 3
	0x14		jr ra	jalr x0 0 (ra)
	0x18	main	jal ra f1	jal ra 0x0
	0x1c		jal ra f2	jal ra 0x8
	0x20		jal ra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

Address	Binary	Value
0x00000000 - 0x00000003	f1 00 10 05 13	addi a0 x0 1
0x00000004 - 0x00000007	00 00 80 67	jalr x0 0 (ra)
0x00000008 - 0x0000000B	f2 00 20 05 13	addi a0 x0 2
0x0000000C - 0x0000000F	00 00 80 67	jalr x0 0 (ra)
0x00000010 - 0x00000013	f3 00 30 05 13	addi a0 x0 3
0x00000014 - 0x00000017	00 00 80 67	jalr x0 0 (ra)
0x00000018 - 0x0000001B	main 00 00 00 EF	jal ra 0x0
0x0000001C - 0x0000001F	00 00 80 EF	jal ra 0x8

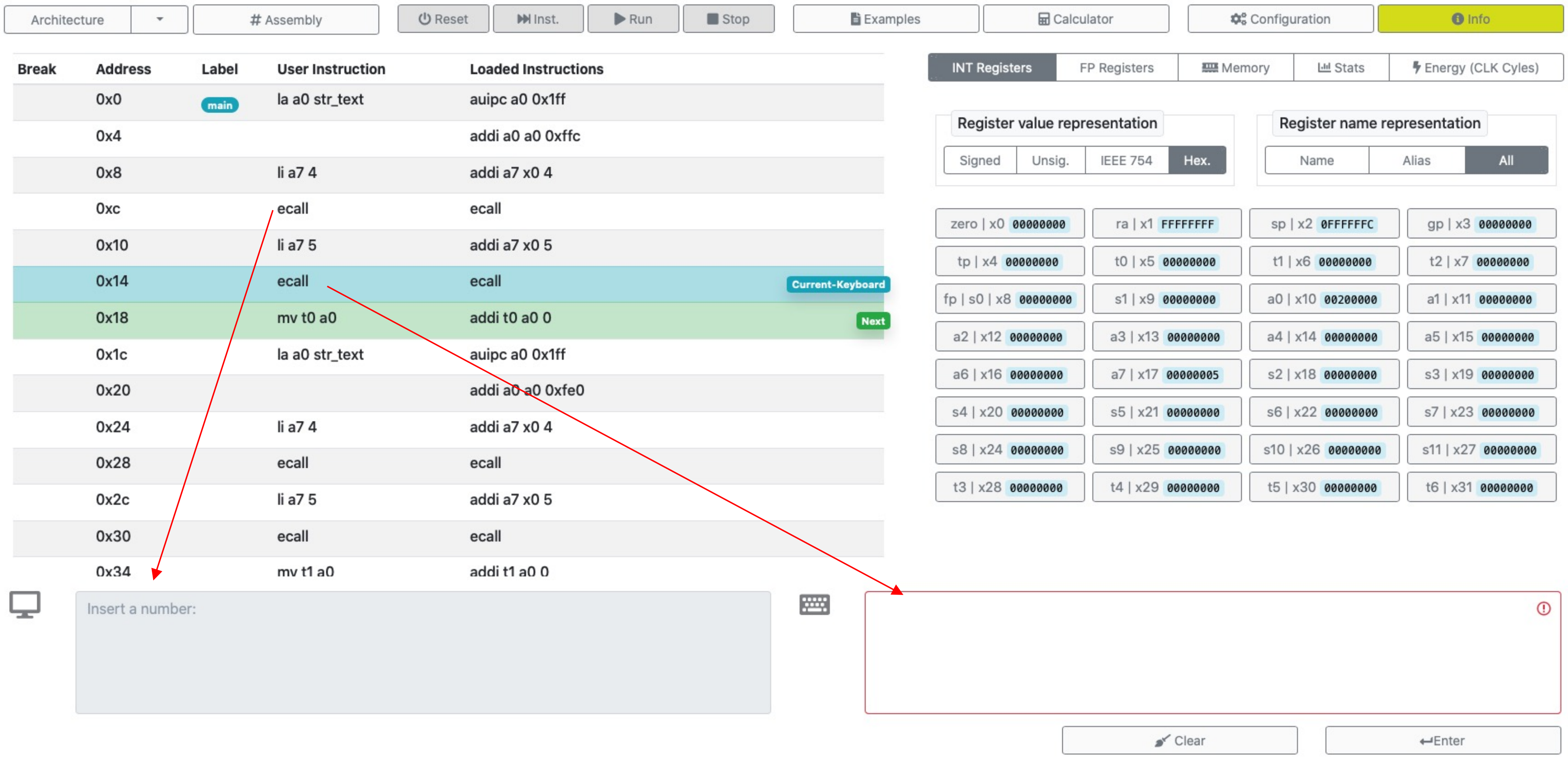
Llamadas al sistema

ejemplo


CREATOR 3.2 RISC-V (RV32IMFD)
didaCtic and geneRic assEmbly progrAmming simulaTOR
Architecture
Simulator
Compile/Linked
File
Assembly:

```

.data
    str_text: .string "Insert a number: "
    str_result: .string "Result = "



.text
main:
    # print "Insert a number: "
    la a0 str_text
    li a7 4
    ecall

    # read int
    li a7 5
    ecall

    mv t0, a0

    # print "Insert a number: "
    la a0 str_text
    li a7 4
    ecall

      # read int
    li a7 5
    ecall
    mv t1, a0

    # print "Result = "
    la a0 str_result
    li a7 4
    ecall
```

Llamadas al sistema



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

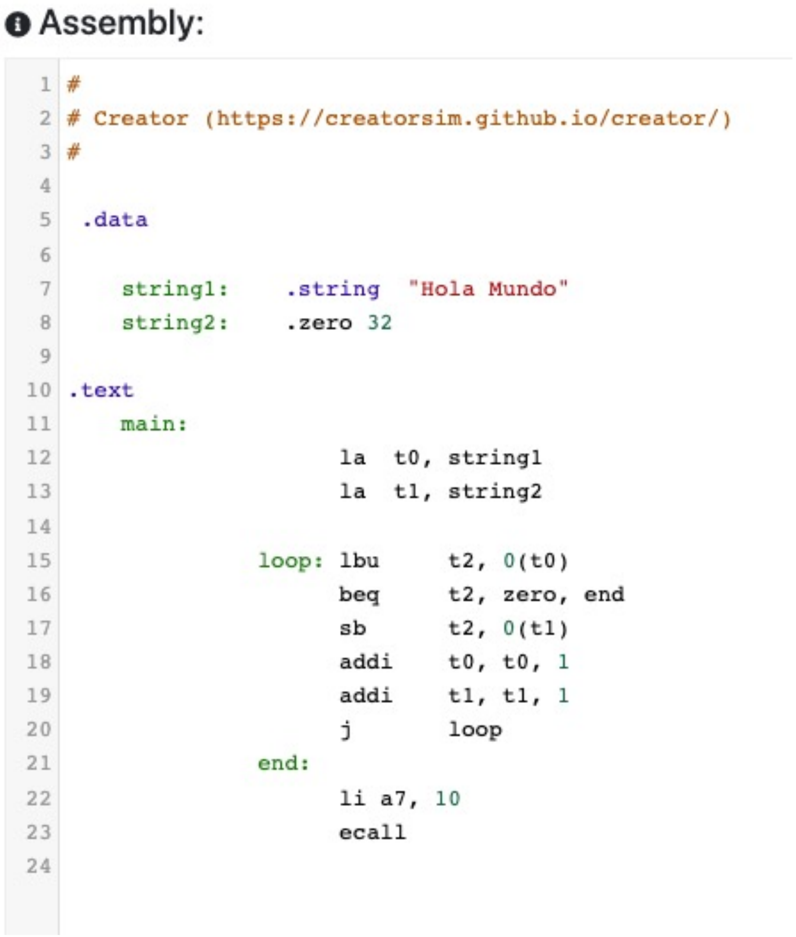
Register value representation: Signed | Unsig. | IEEE 754 | [✓ Hex.]

Register name representation: Name | Alias | [✓ All]

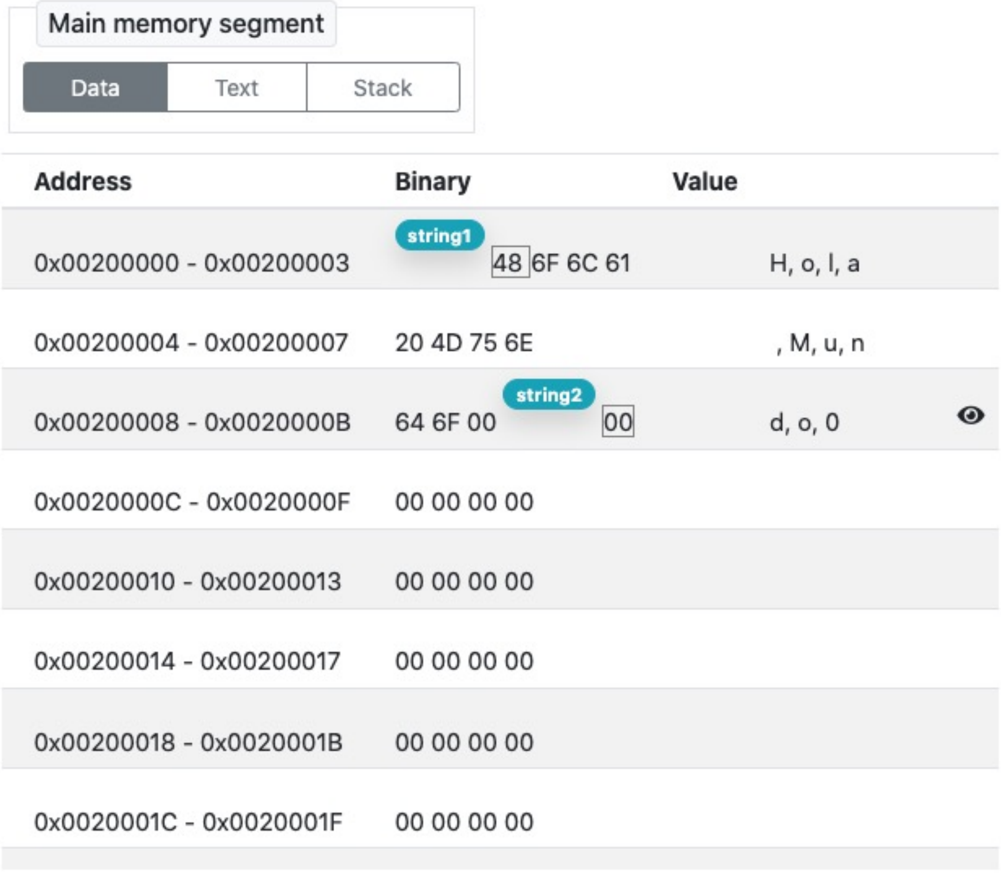
zero | x0 00000000	ra | x1 FFFFFFFF	sp | x2 0FFFFFFC	gp | x3 00000000
tp | x4 00000000	t0 | x5 00000000	t1 | x6 00000000	t2 | x7 00000000
fp | s0 | x8 00000000	s1 | x9 00000000	a0 | x10 00200000	a1 | x11 00000000
a2 | x12 00000000	a3 | x13 00000000	a4 | x14 00000000	a5 | x15 00000000
a6 | x16 00000000	a7 | x17 00000005	s2 | x18 00000000	s3 | x19 00000000
s4 | x20 00000000	s5 | x21 00000000	s6 | x22 00000000	s7 | x23 00000000
s8 | x24 00000000	s9 | x25 00000000	s10 | x26 00000000	s11 | x27 00000000
t3 | x28 00000000	t4 | x29 00000000	t5 | x30 00000000	t6 | x31 00000000

Ejemplo: ejecutar el siguiente programa

ejemplo

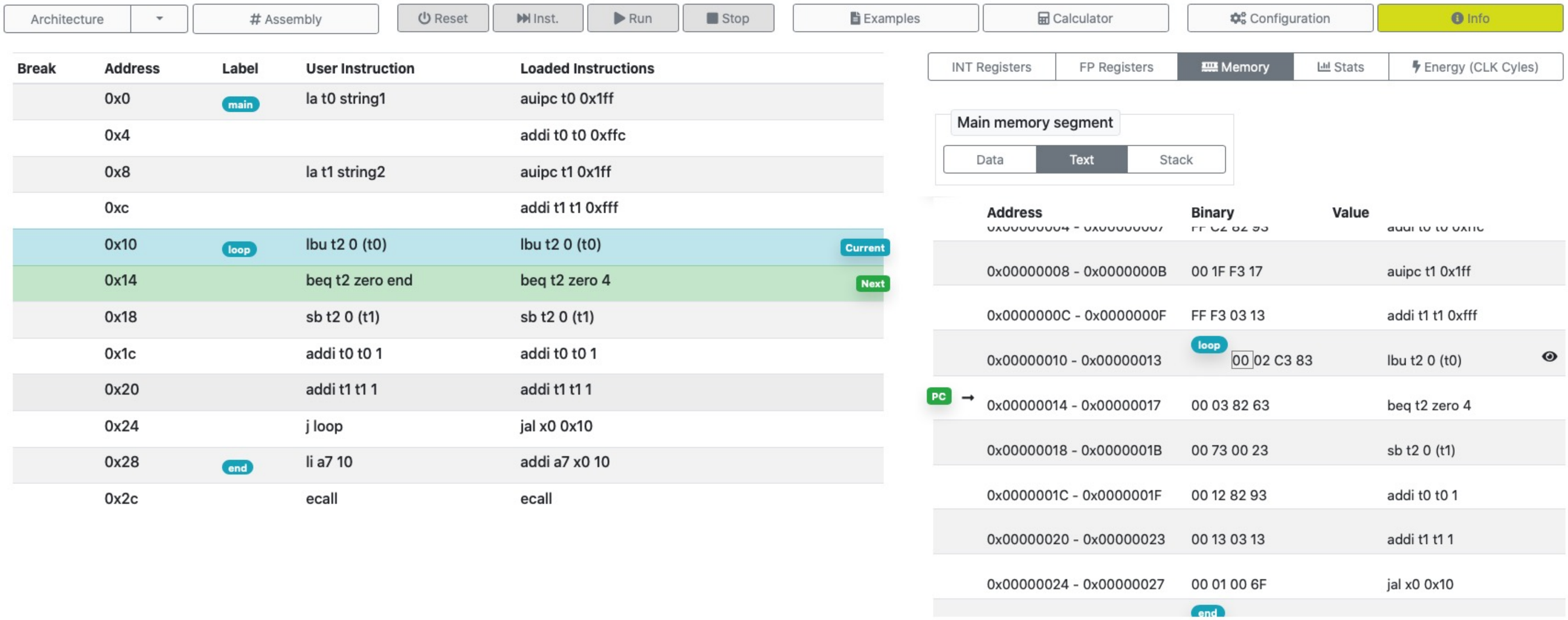


```
#
# Creator (https://creatorsim.github.io/creator/)
#

 .data

    string1:    .string  "Hola Mundo"
    string2:    .zero 32

.text
    main:
                    la  t0, string1
                    la  t1, string2

              loop: lbu     t2, 0(t0)
                    beq     t2, zero, end
                    sb      t2, 0(t1)
                    addi    t0, t0, 1
                    addi    t1, t1, 1
                    j       loop
              end:
                    li a7, 10
                    ecall
```



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

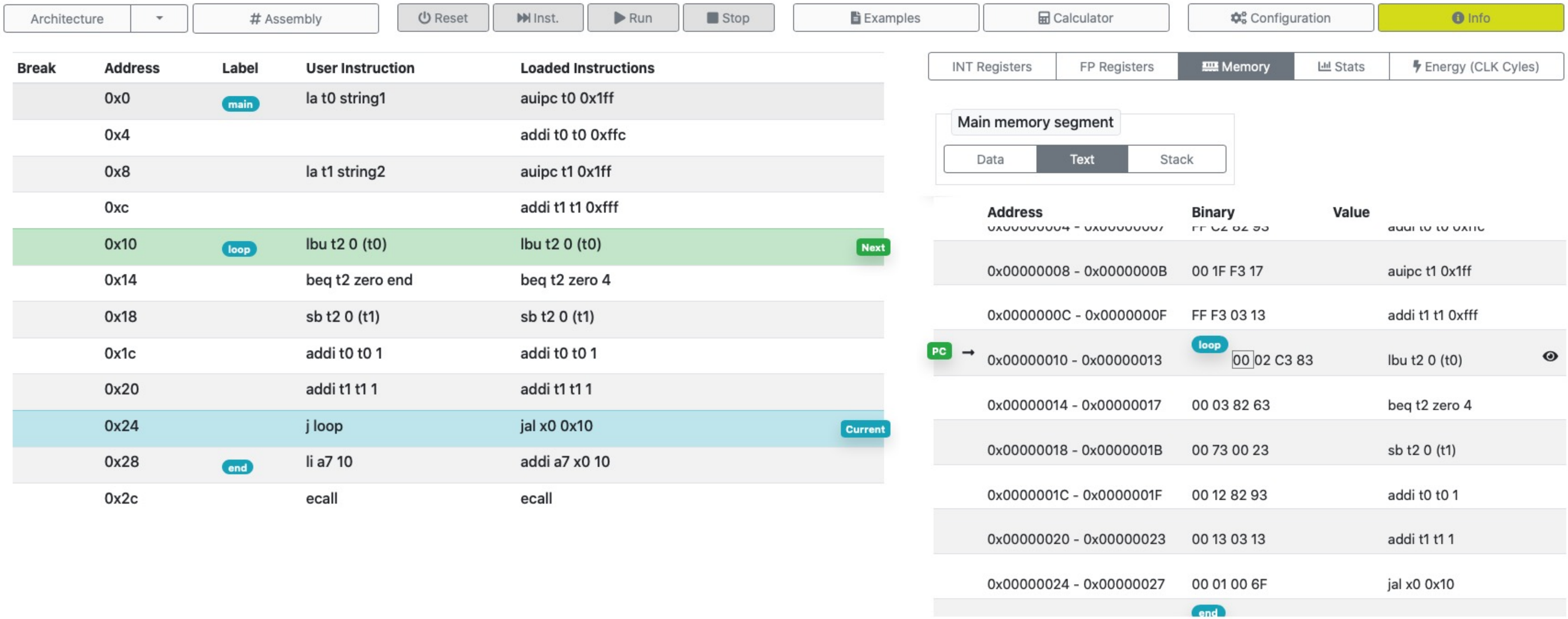
Ejecución del programa



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

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

Ejecución del programa. Bucles



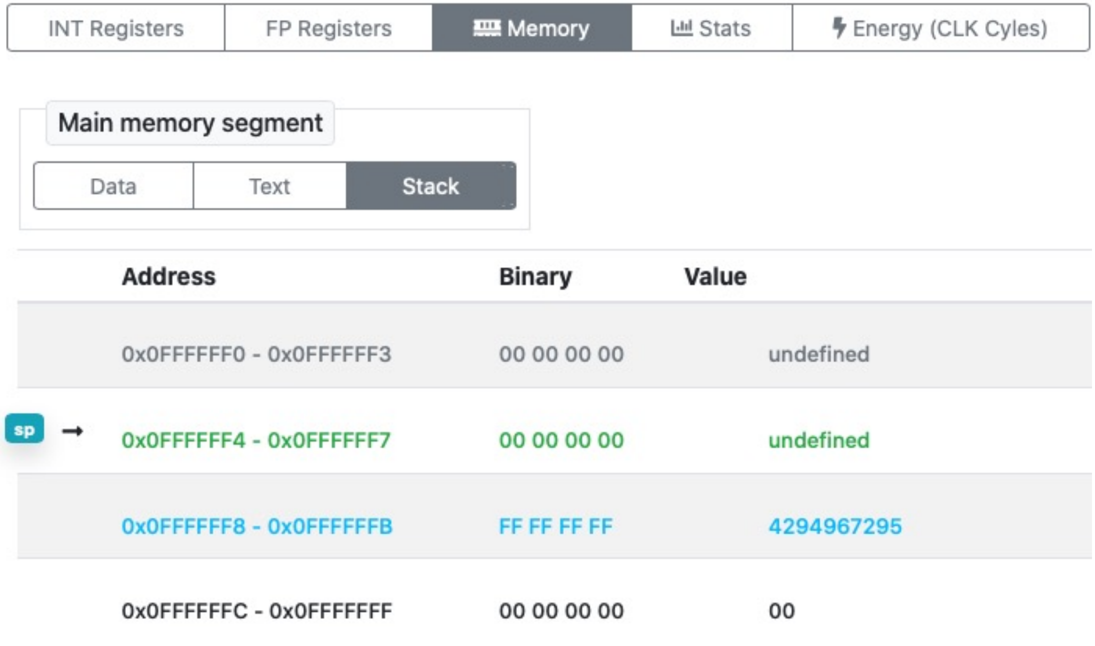
Break	Address	Label	User Instruction	Loaded Instructions	
	0x0	main	la t0 string1	auipc t0 0x1ff	
	0x4			addi t0 t0 0xffc	
	0x8		la t1 string2	auipc t1 0x1ff	
	0xc			addi t1 t1 0xfff	
	0x10	loop	lbu t2 0 (t0)	lbu t2 0 (t0)	Next
	0x14		beq t2 zero end	beq t2 zero 4	
	0x18		sb t2 0 (t1)	sb t2 0 (t1)	
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

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

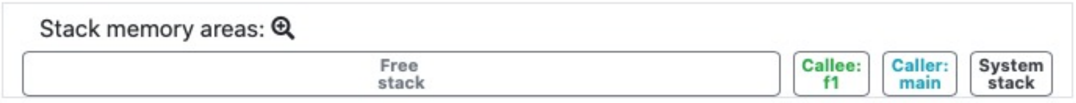
Ejemplo de llamadas a funciones anidadas

ejemplo

- Comprobar el crecimiento de la pila



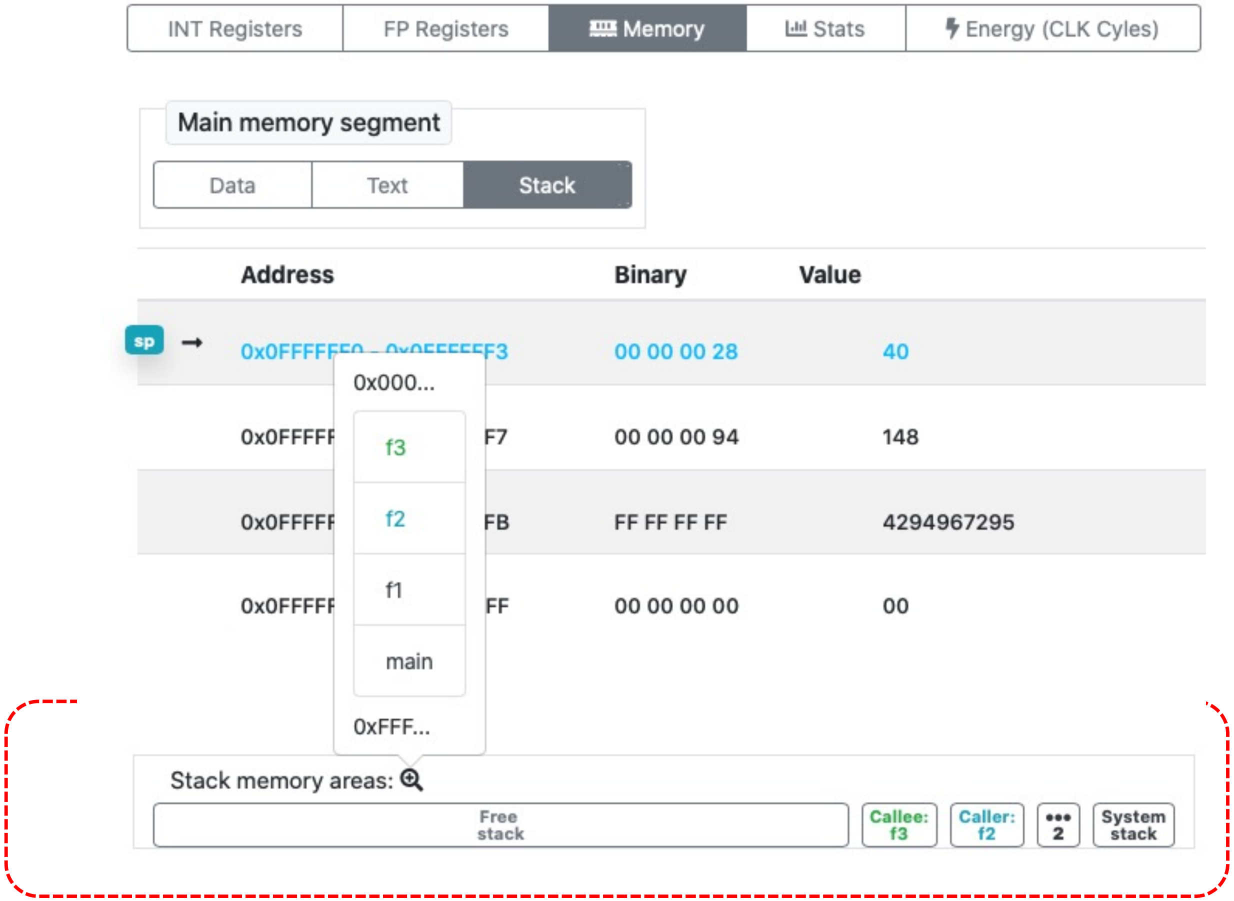
Address	Binary	Value
0x0FFFFFF0 - 0x0FFFFFF3	00 00 00 00	undefined
0x0FFFFFF4 - 0x0FFFFFF7	00 00 00 00	undefined
0x0FFFFFF8 - 0x0FFFFFFB	FF FF FF FF	4294967295
0x0FFFFFFC - 0x0FFFFFFF	00 00 00 00	00



Ejemplo de llamadas a funciones anidadas

ejemplo

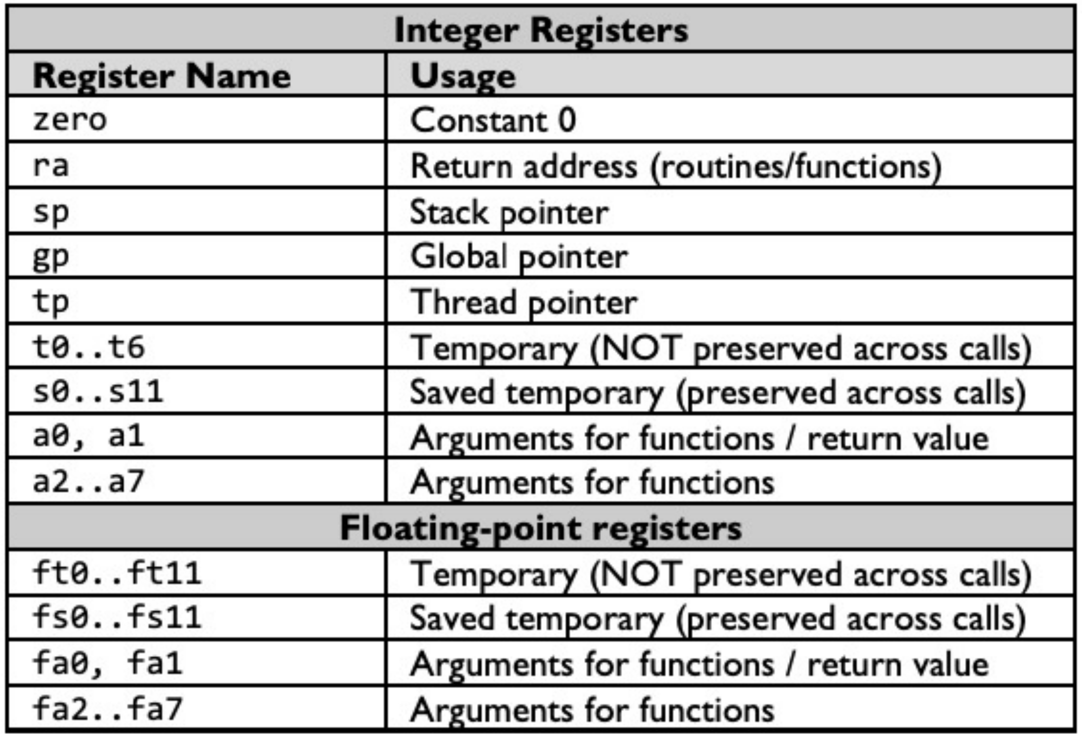
- Comprobar el crecimiento de la pila



Address	Binary	Value
0x0FFFFFF0 – 0x0FFFFFF3	00 00 00 28	40
0x0FFFFF F7	00 00 00 94	148
0x0FFFFF FB	FF FF FF FF	4294967295
0x0FFFFF FF	00 00 00 00	00

Llamadas a funciones

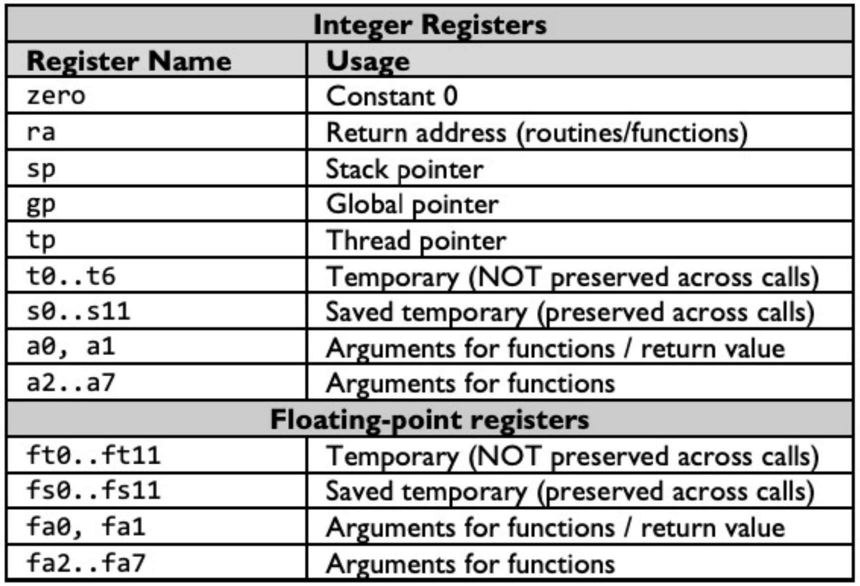
- Convenio simplificado
  - La pila no necesita estar alineada a 8 bytes
- Alerta si se incumple el convenio de paso de parámetros y uso de pila



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

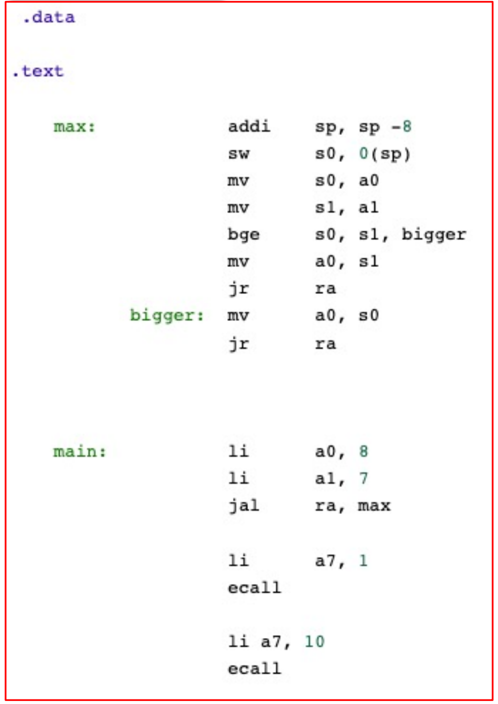
Detección de errores en el convenio de paso de parámetros

- 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

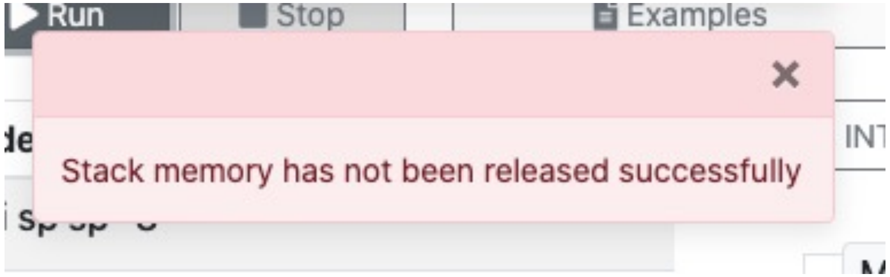
ejemplo



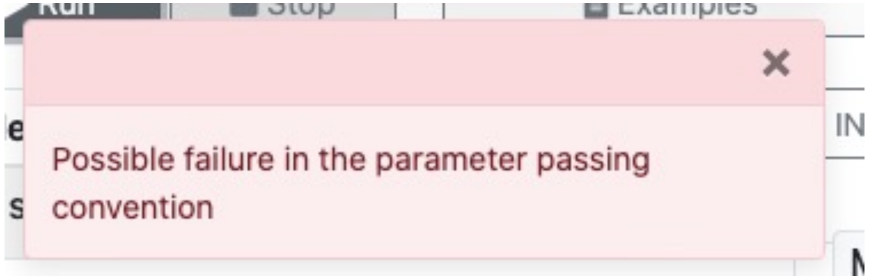
```
 .data

.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

- Fallos detectados:

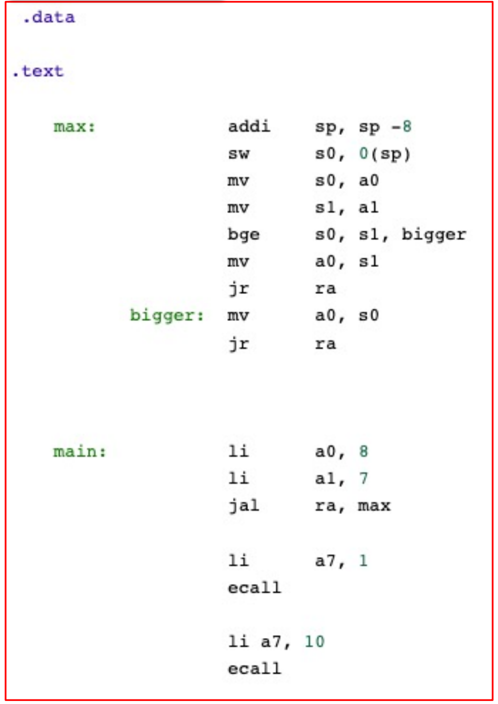


Stack memory has not been released successfully



Possible failure in the parameter passing convention

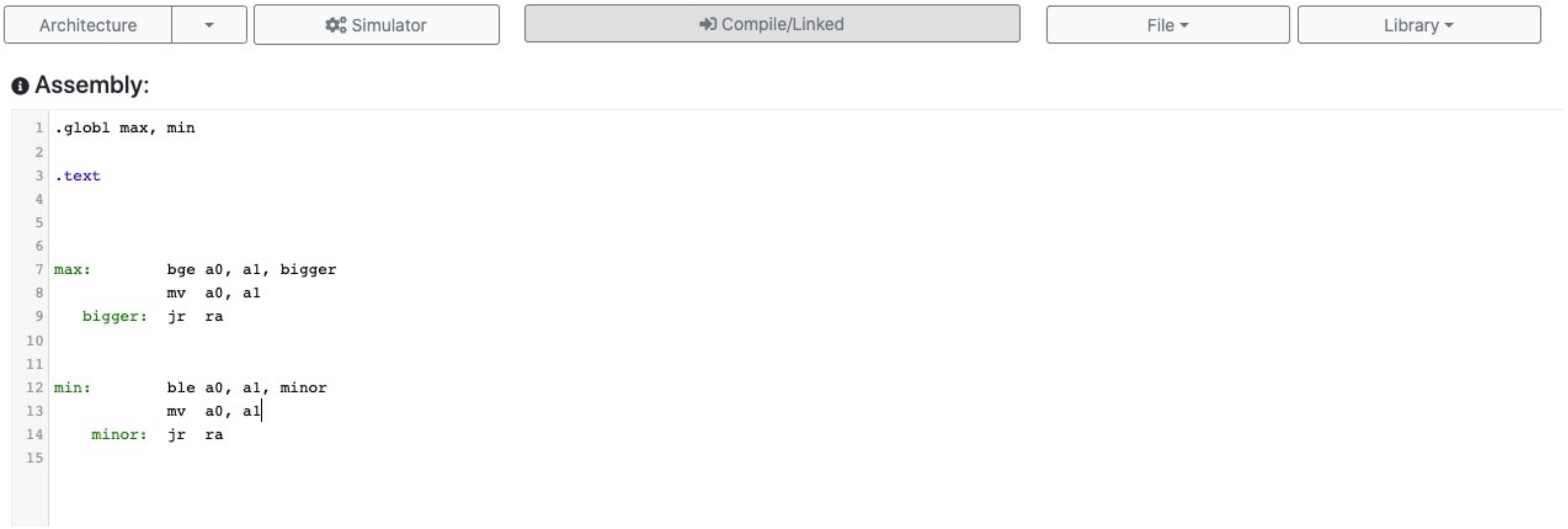
ejemplo



```
 .data

.text

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



    main:               li      a0, 8
                        li      a1, 7
                        jal     ra, max

                        li      a7, 1
                        ecall

                        li a7, 10
                        ecall
```

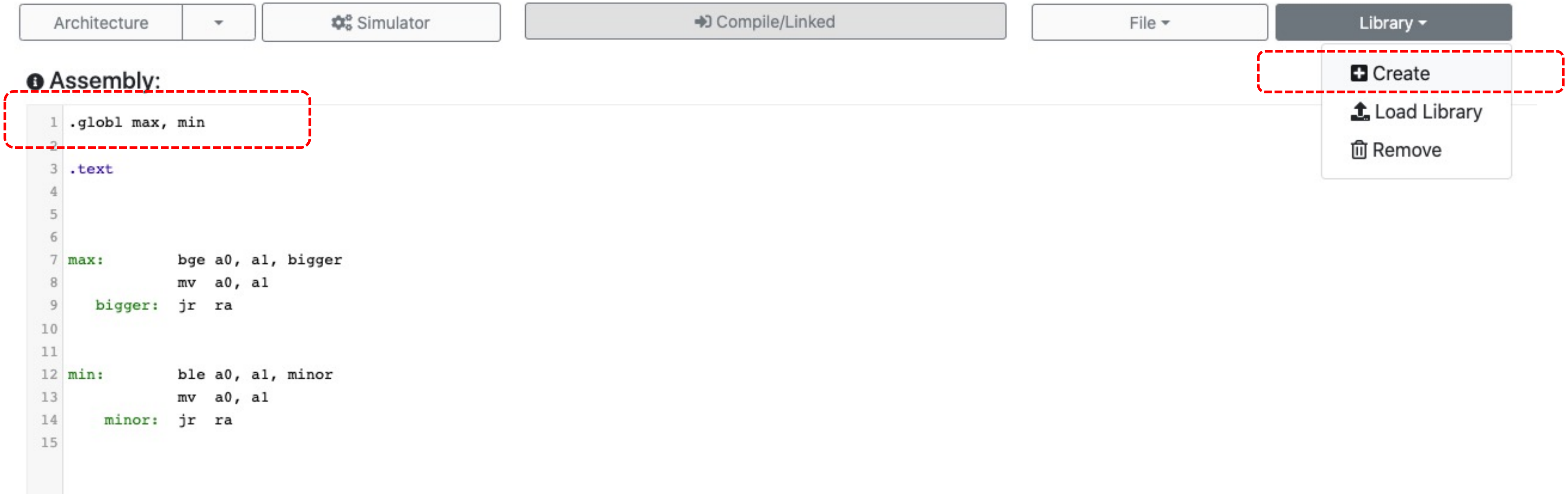
Creación de librerías

ejemplo



```
.globl max, min

.text



max:        bge a0, a1, bigger
            mv  a0, a1
   bigger:  jr  ra


min:        ble a0, a1, minor
            mv  a0, a1
    minor:  jr  ra

```

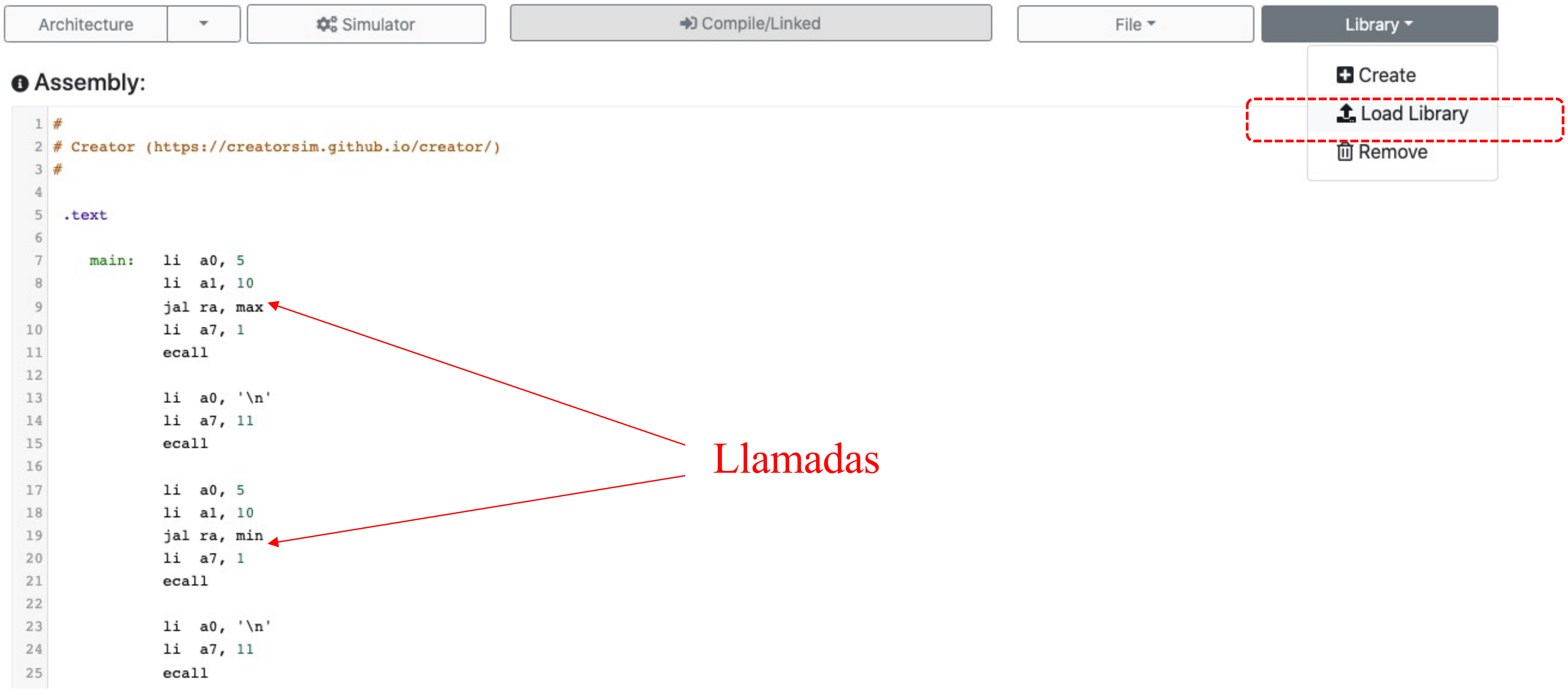
Creación de librerías



```
.globl max, min

.text



max:        bge a0, a1, bigger
            mv  a0, a1
   bigger:  jr  ra


min:        ble a0, a1, minor
            mv  a0, a1
    minor:  jr  ra
```

Uso de librerías

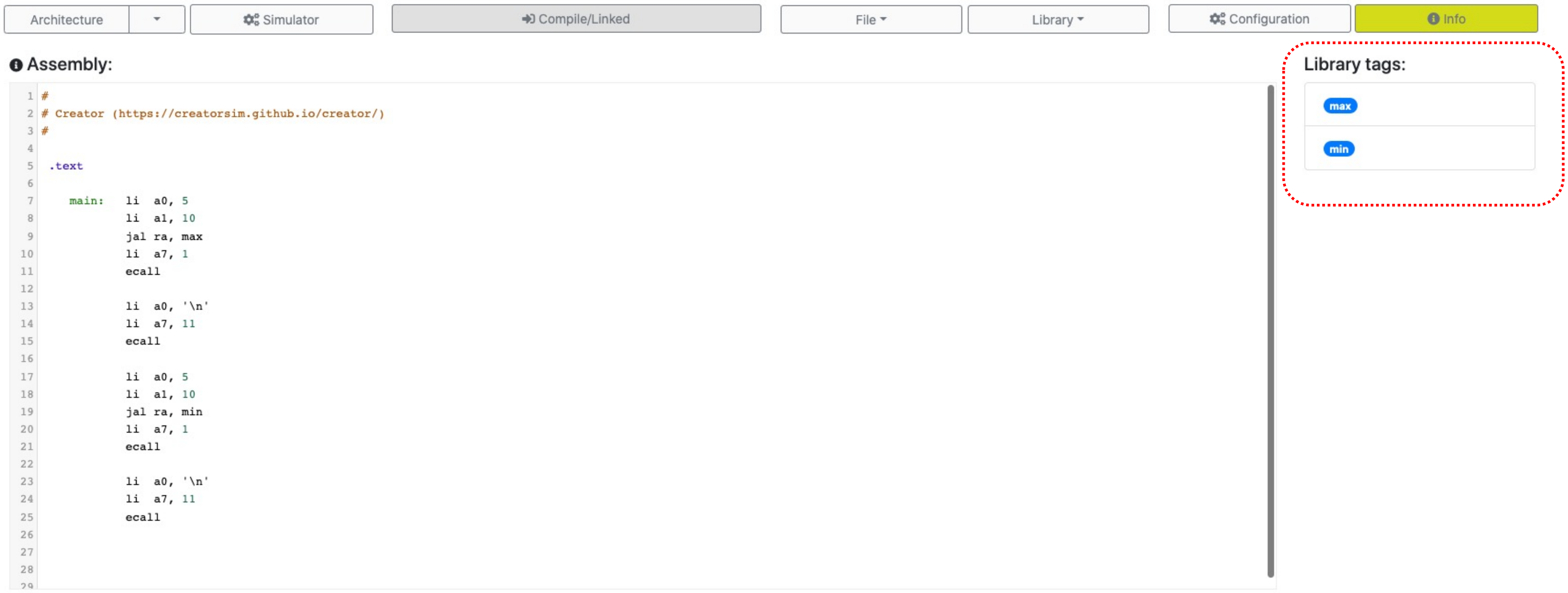
ejemplo



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

 .text

    main:   li  a0, 5
            li  a1, 10
            jal ra, max
            li  a7, 1
            ecall

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

            li  a0, 5
            li  a1, 10
            jal ra, min
            li  a7, 1
            ecall

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

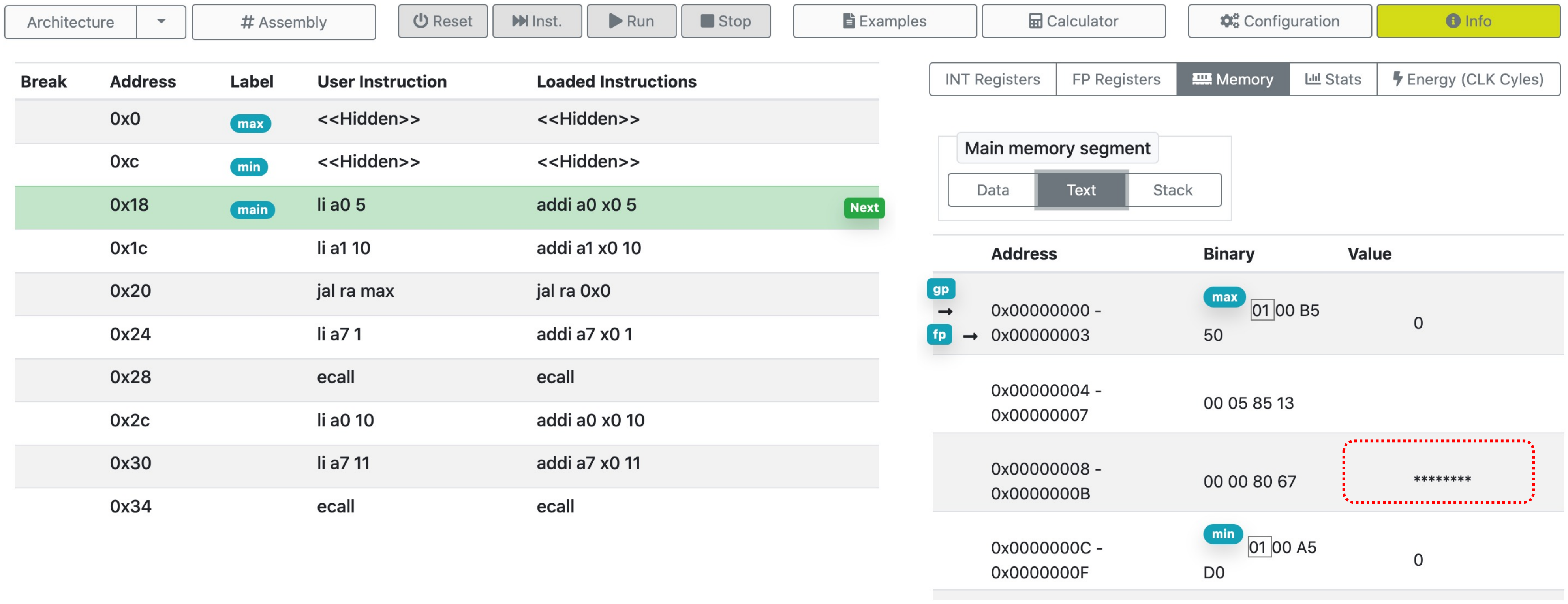
Llamadas

Librería cargada



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

 .text

    main:   li  a0, 5
            li  a1, 10
            jal ra, max
            li  a7, 1
            ecall

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

            li  a0, 5
            li  a1, 10
            jal ra, min
            li  a7, 1
            ecall

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

Uso de librerías



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

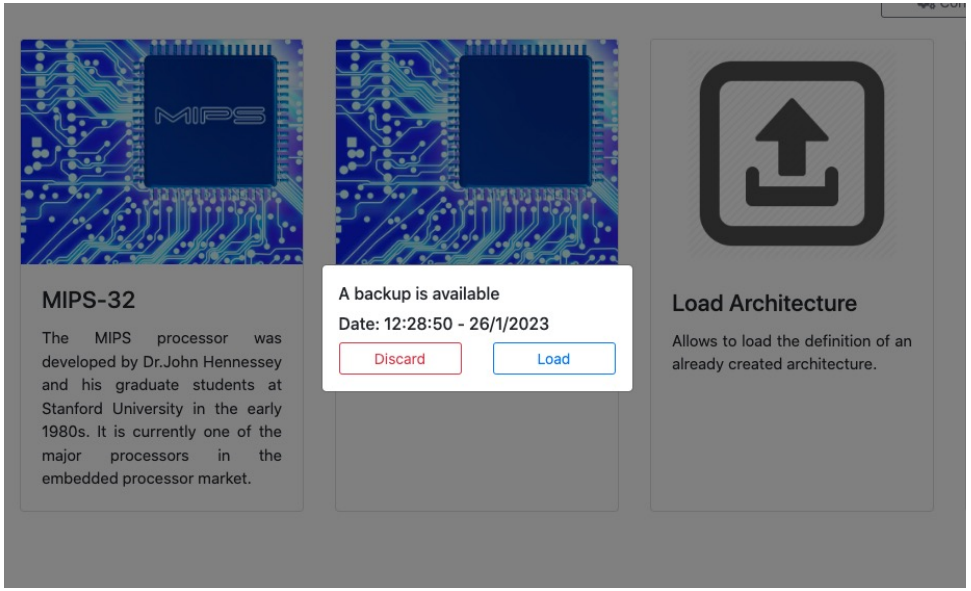
Address	Binary	Value
0x00000000 - 0x00000003	max 01 00 B5 50	0
0x00000004 - 0x00000007	00 05 85 13	
0x00000008 - 0x0000000B	00 00 80 67	********
0x0000000C - 0x0000000F	min 01 00 A5 D0	0

Errores en tiempo de ejecución

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

Caché del navegador para recuperación de errores

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



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.

A backup is available

Date: 12:28:50 - 26/1/2023

Load Architecture

Allows to load the definition of an already created architecture.

Contenido (RISC-V)

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

Soporte a la corrección de prácticas

- Ejecución en línea de comandos
- Prerrequisitos:
  - Linux, node.js y npm
- Pasos:
  - Descargar el repositorio:

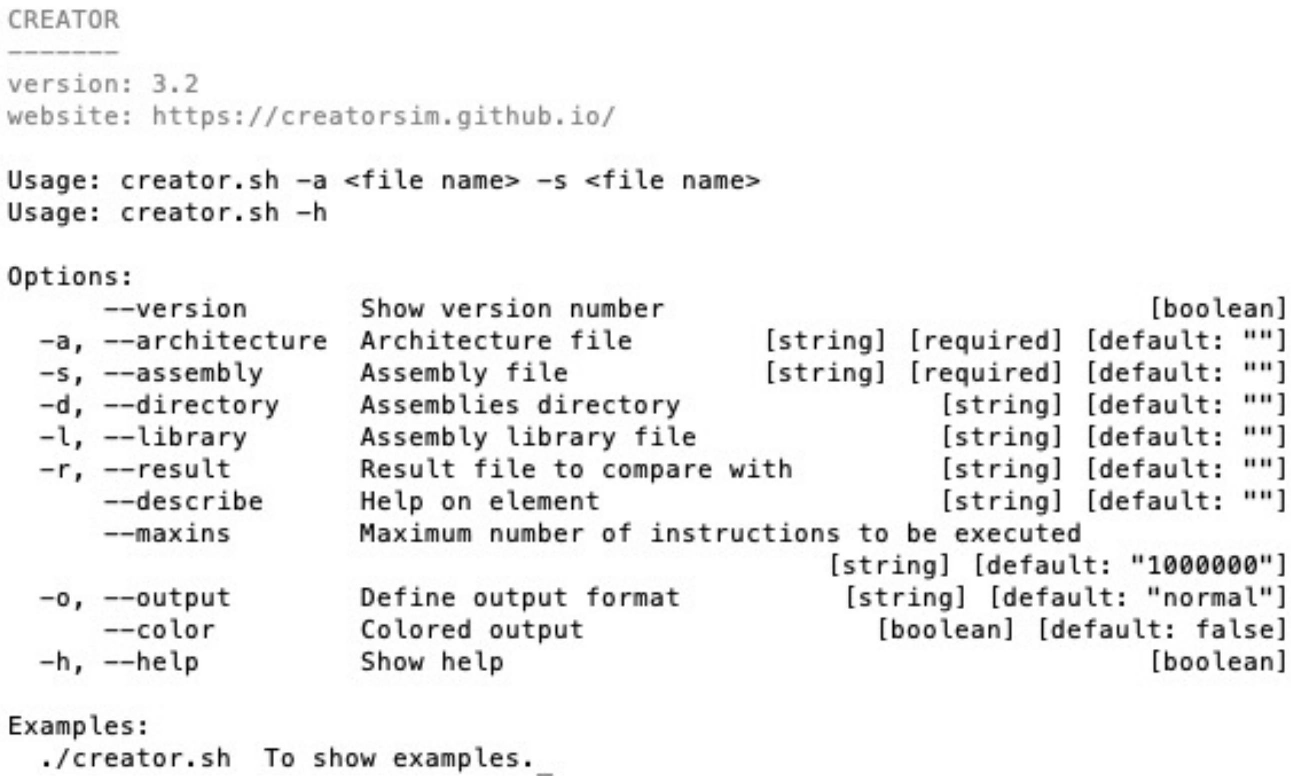
    -
      ```
      git clone https://github.com/creatorsim/creator.git
      ```

  -
    ```
    cd creator
    ```
  - Instalar los paquetes:

    -
      ```
      npm install terser jshint colors yargs readline-sync
      ```

Compilación y ejecución de un programa

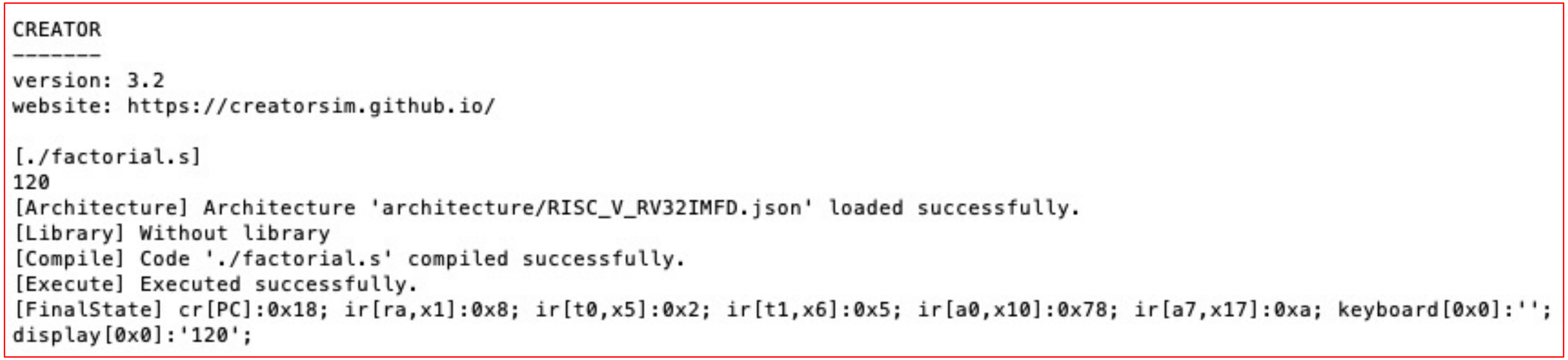
- `./creator.sh -h`



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

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

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

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

Ejecución de un programa

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

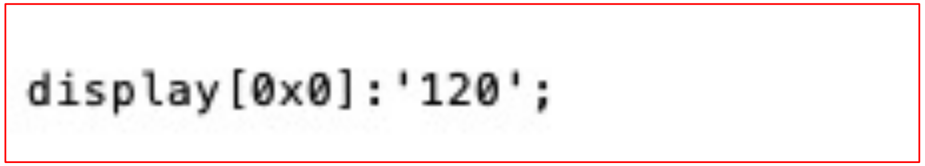


```
CREATOR
--------
version: 3.2
website: https://creatorsim.github.io/

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

Ejecución de un programa y comprobación de resultados

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

referencia.txt

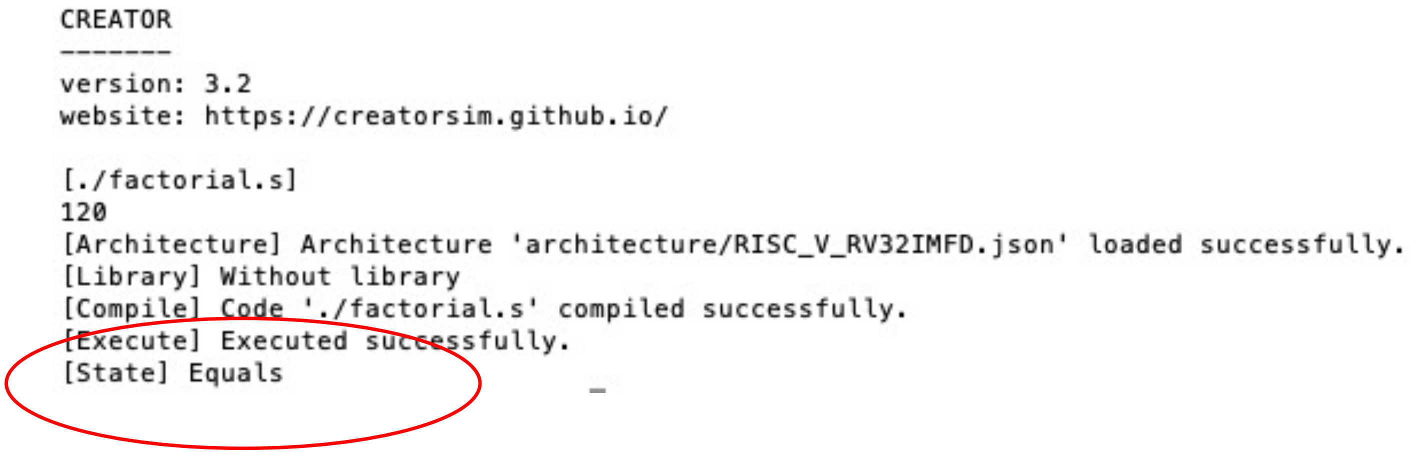


```
display[0x0]:'120';
```

- Ejecución y comparación con salida de referencia:
  - ./creator.sh -a architecture/RISC_V_RV32IMFD.json -s ./factorial.s -r referencia.txt

Ejemplo de salida correcta

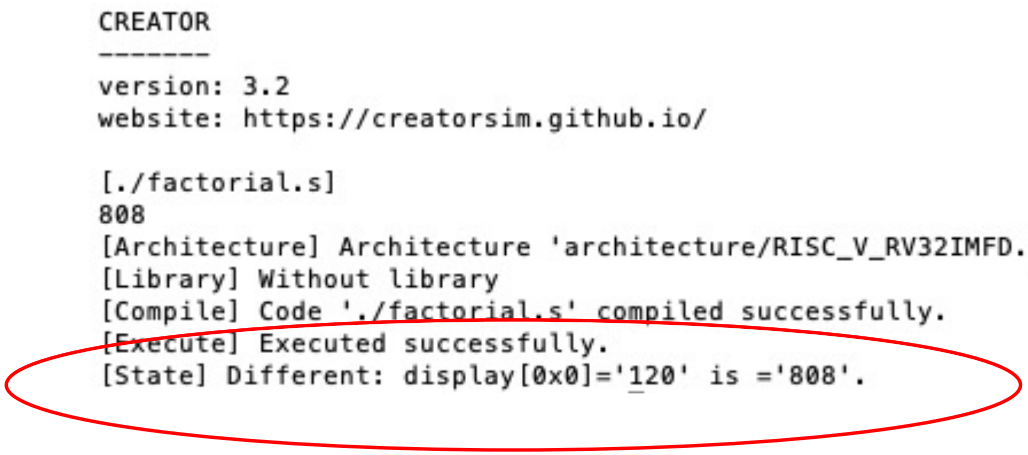
- `./creator.sh -a architecture/ RISC_V_RV32IMFD.json  -s factorial.s -r referencia.txt`



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

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

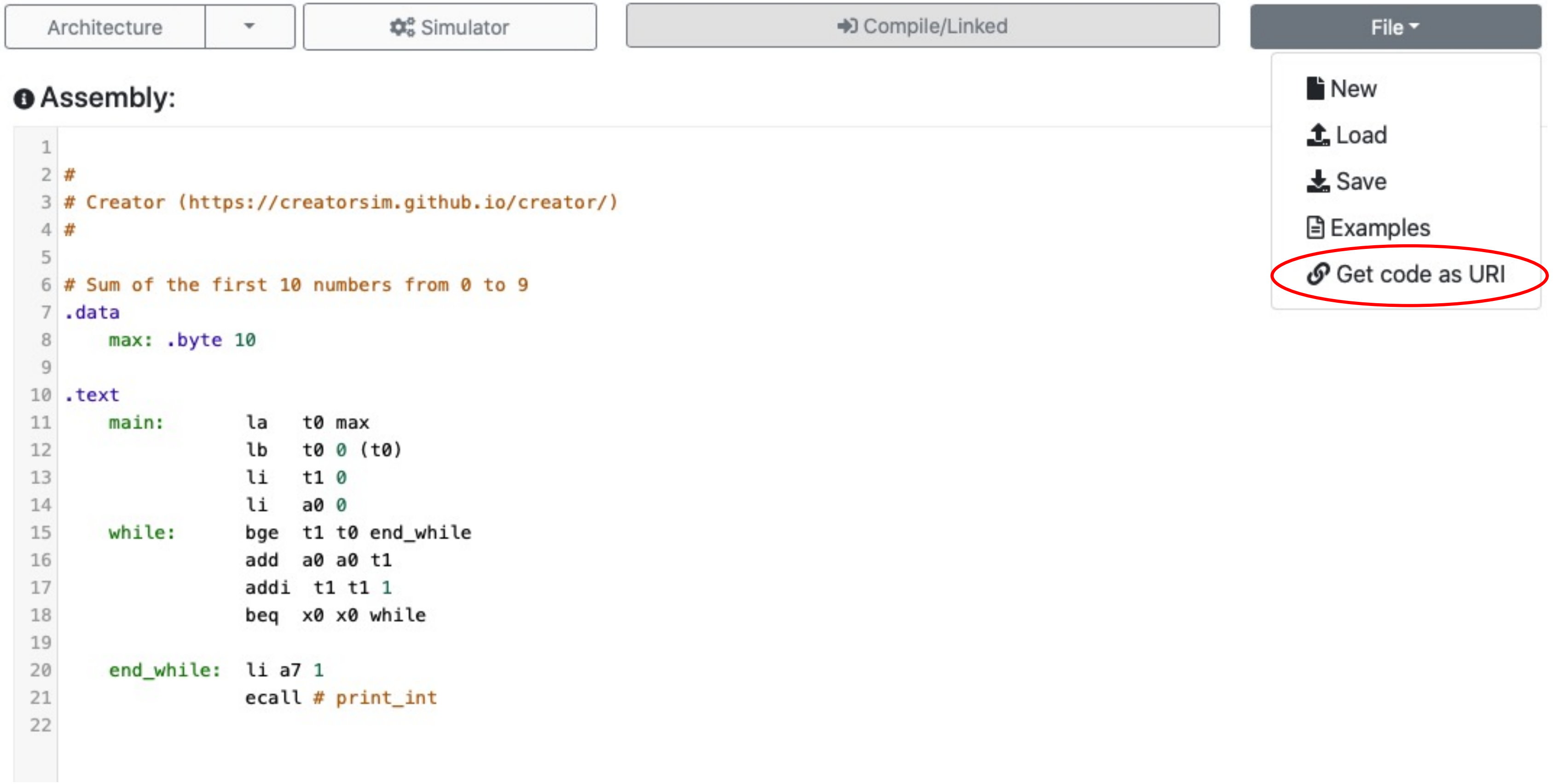
Ejemplo de salida incorrecta

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



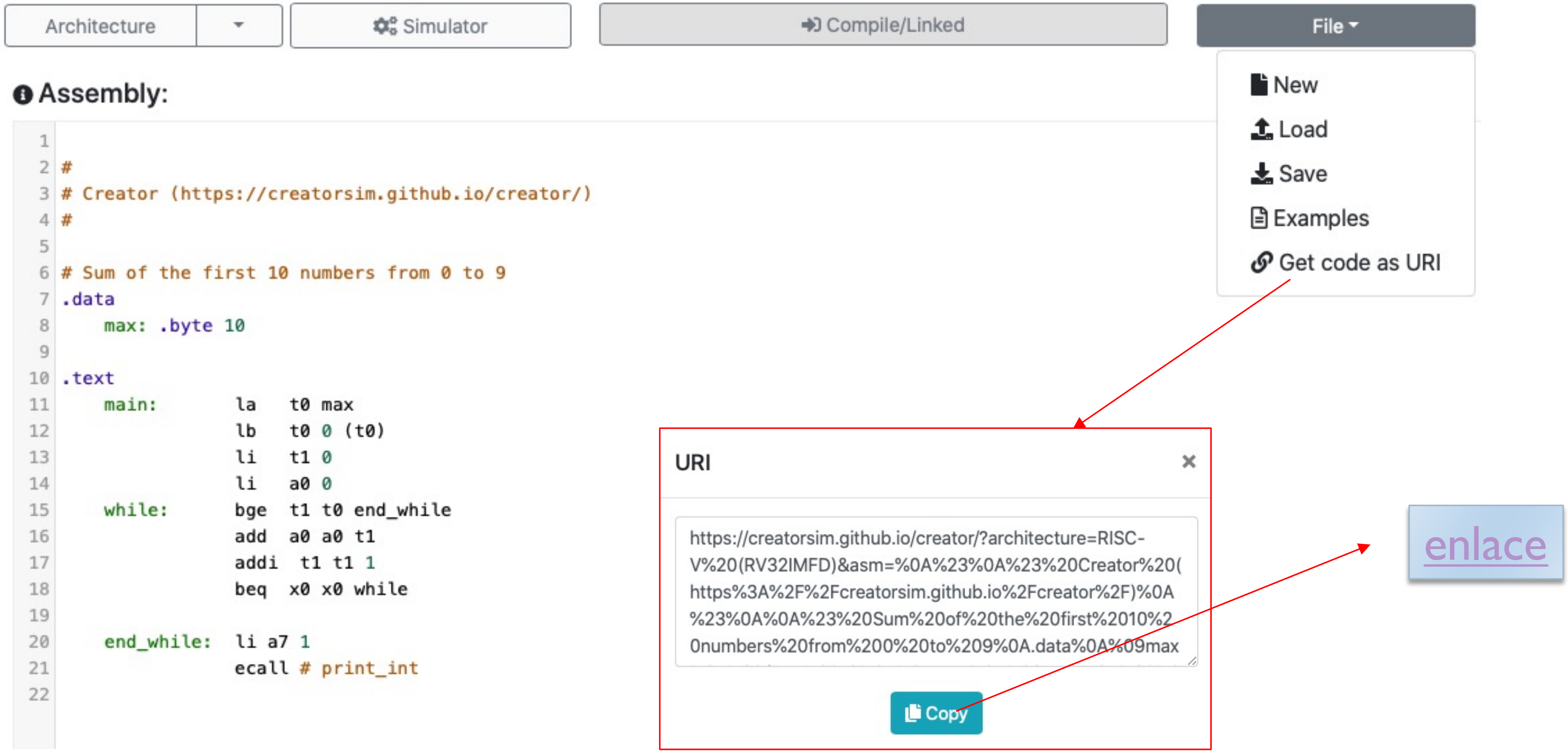
```
CREATOR
--------
version: 3.2
website: https://creatorsim.github.io/

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

Ayuda a la creación de materiales docentes



```

#
# Creator (https://creatorsim.github.io/creator/)
#

# Sum of the first 10 numbers from 0 to 9
.data
    max: .byte 10

.text
    main:       la   t0 max
                lb   t0 0 (t0)
                li   t1 0
                li   a0 0
    while:      bge  t1 t0 end_while
                add  a0 a0 t1
                addi  t1 t1 1
                beq  x0 x0 while

    end_while:  li a7 1
                ecall # print_int

```

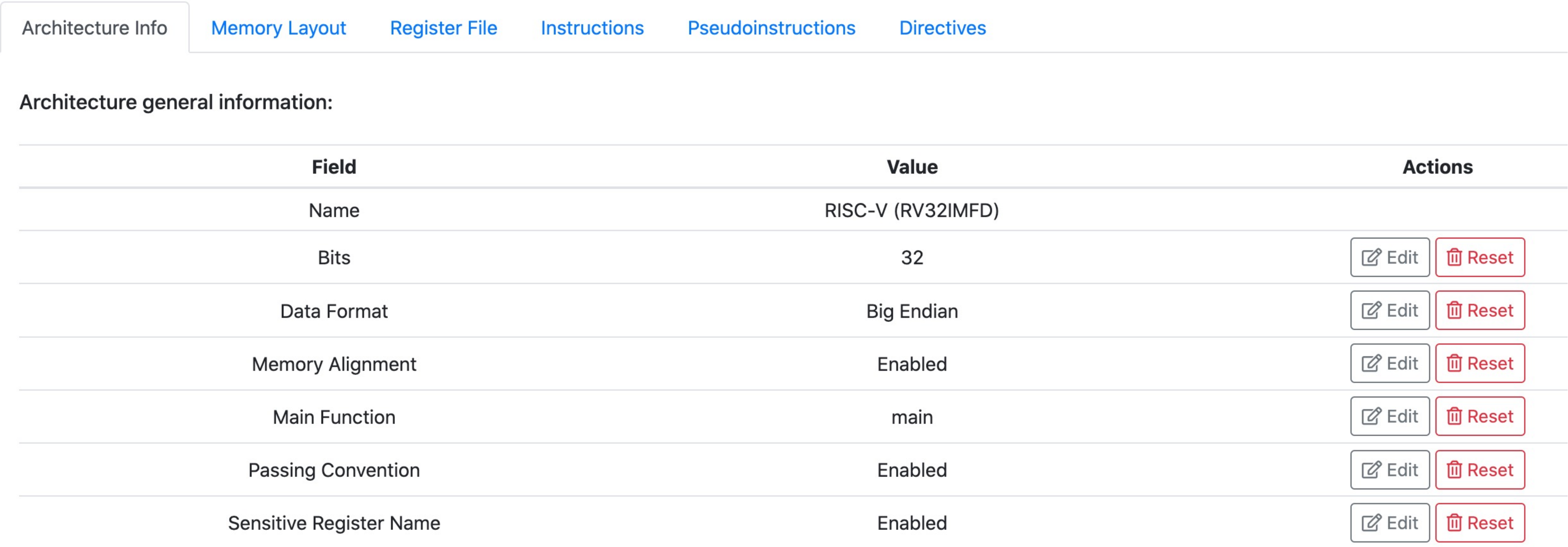
Ayuda a la creación de materiales docentes



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

# Sum of the first 10 numbers from 0 to 9
.data
    max: .byte 10

.text
    main:       la   t0 max
                lb   t0 0 (t0)
                li   t1 0
                li   a0 0
    while:      bge  t1 t0 end_while
                add  a0 a0 t1
                addi  t1 t1 1
                beq  x0 x0 while

    end_while:  li a7 1
                ecall # print_int
```

https://creatorsim.github.io/creator/?architecture=RISC-V%20(RV32IMFD)&asm=%0A%23%0A%23%20Creator%20(https%3A%2F%2Fcreatorsim.github.io%2Fcreator%2F)%0A%23%0A%0A%23%20Sum%20of%20the%20first%2010%20numbers%20from%200%20to%209%0A.data%0A%09max

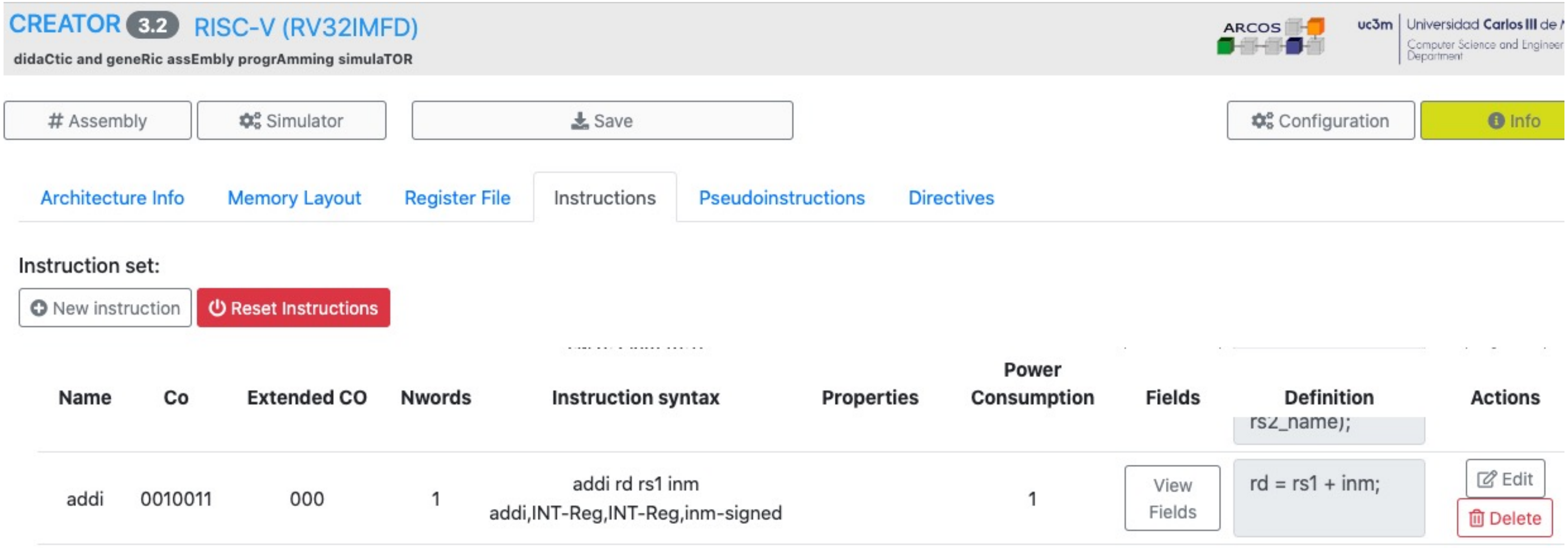
Capacidades para extender el juego de instrucciones y crear nuevas arquitecturas



Architecture general information:

Field	Value	Actions
Name	RISC-V (RV32IMFD)	
Bits	32	Edit Reset
Data Format	Big Endian	Edit Reset
Memory Alignment	Enabled	Edit Reset
Main Function	main	Edit Reset
Passing Convention	Enabled	Edit Reset
Sensitive Register Name	Enabled	Edit Reset

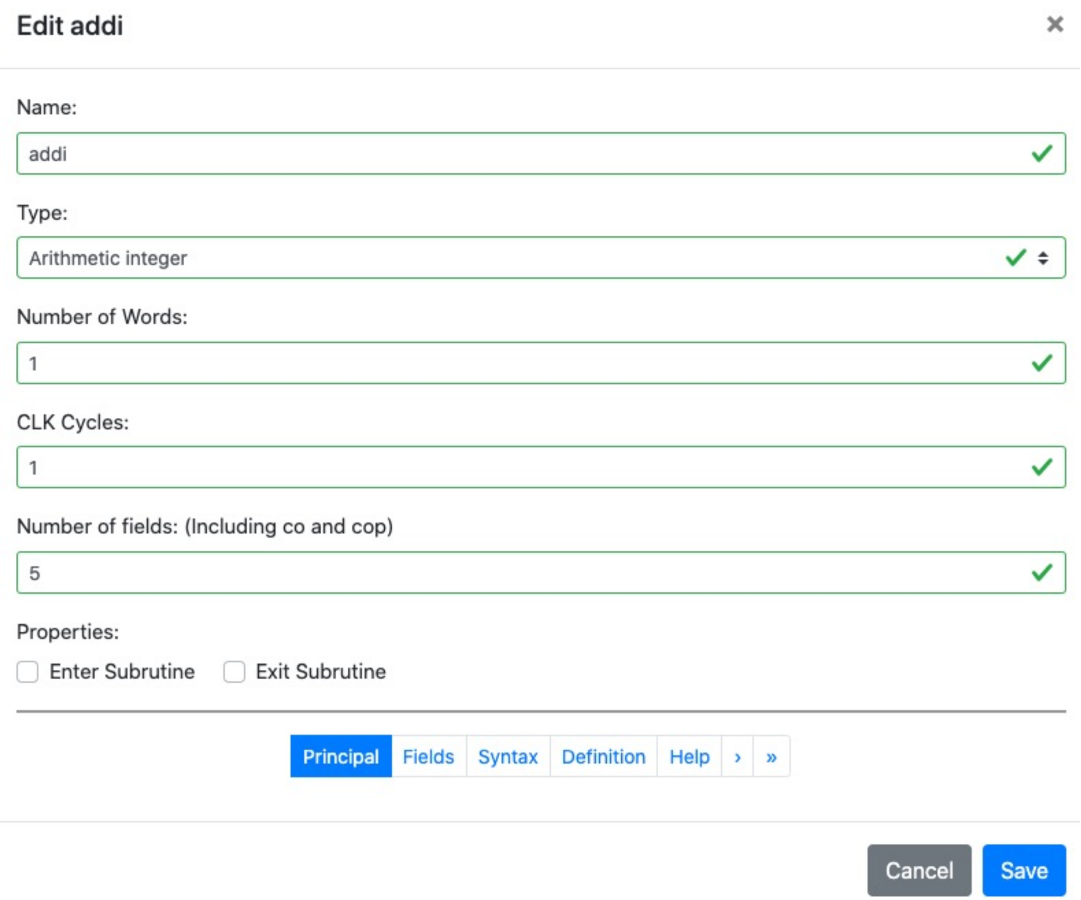
Ejemplo de definición de una instrucción (addi)



Instruction set:

Name	Co	Extended CO	Nwords	Instruction syntax	Properties	Power Consumption	Fields	Definition	Actions
								rs2_name);	
addi	0010011	000	1	addi rd rs1 inm addi,INT-Reg,INT-Reg,inm-signed		1	View Fields	rd = rs1 + inm;	Edit Delete

Ejemplo de definición de una instrucción (addi)



Edit addi

Name: addi ✓

Type: Arithmetic integer ✓

Number of Words: 1 ✓

CLK Cycles: 1 ✓

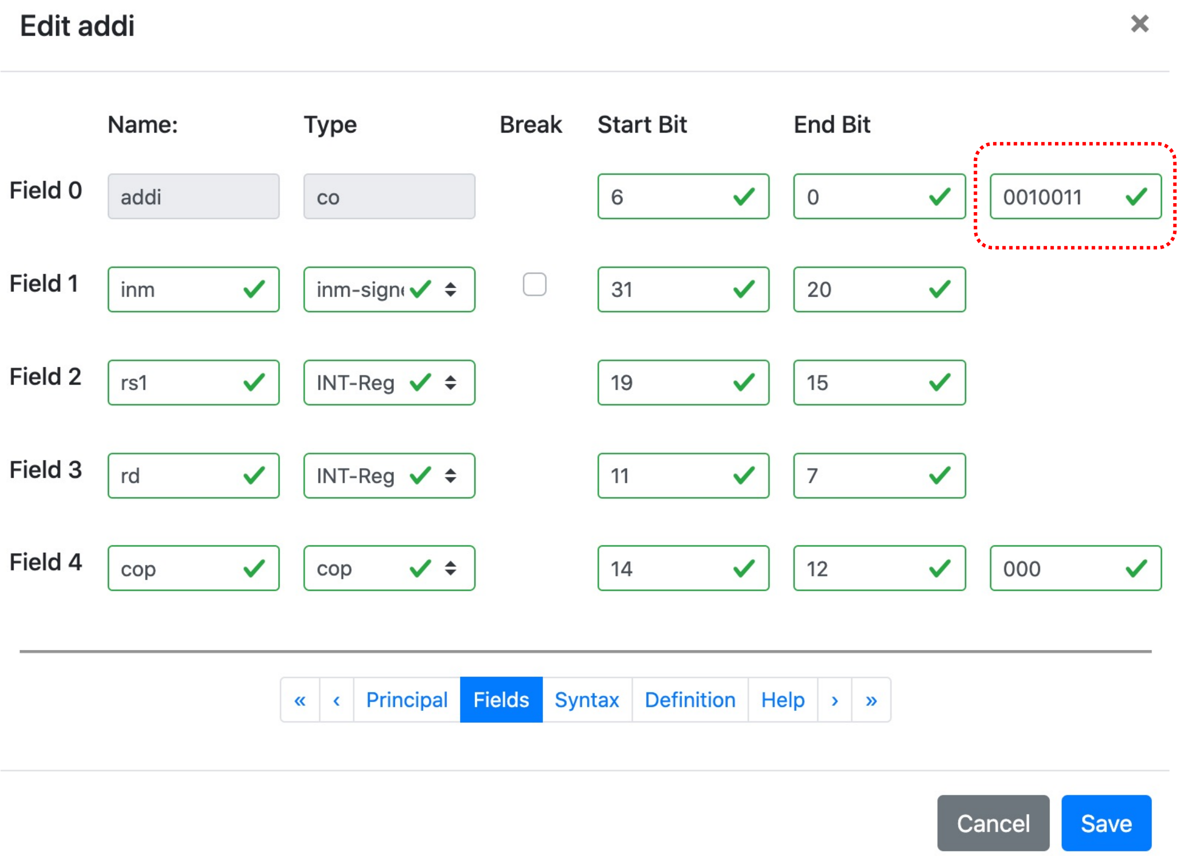
Number of fields: (Including co and cop) 5 ✓

Properties:

□ Enter Subrutine □ Exit Subrutine

Ejemplo de definición de una instrucción (addi)

31 … 20	19 … 15	14 … 12	11 … 7	6 … 0
immediate[11:0]	rs1	000	rd	0010011

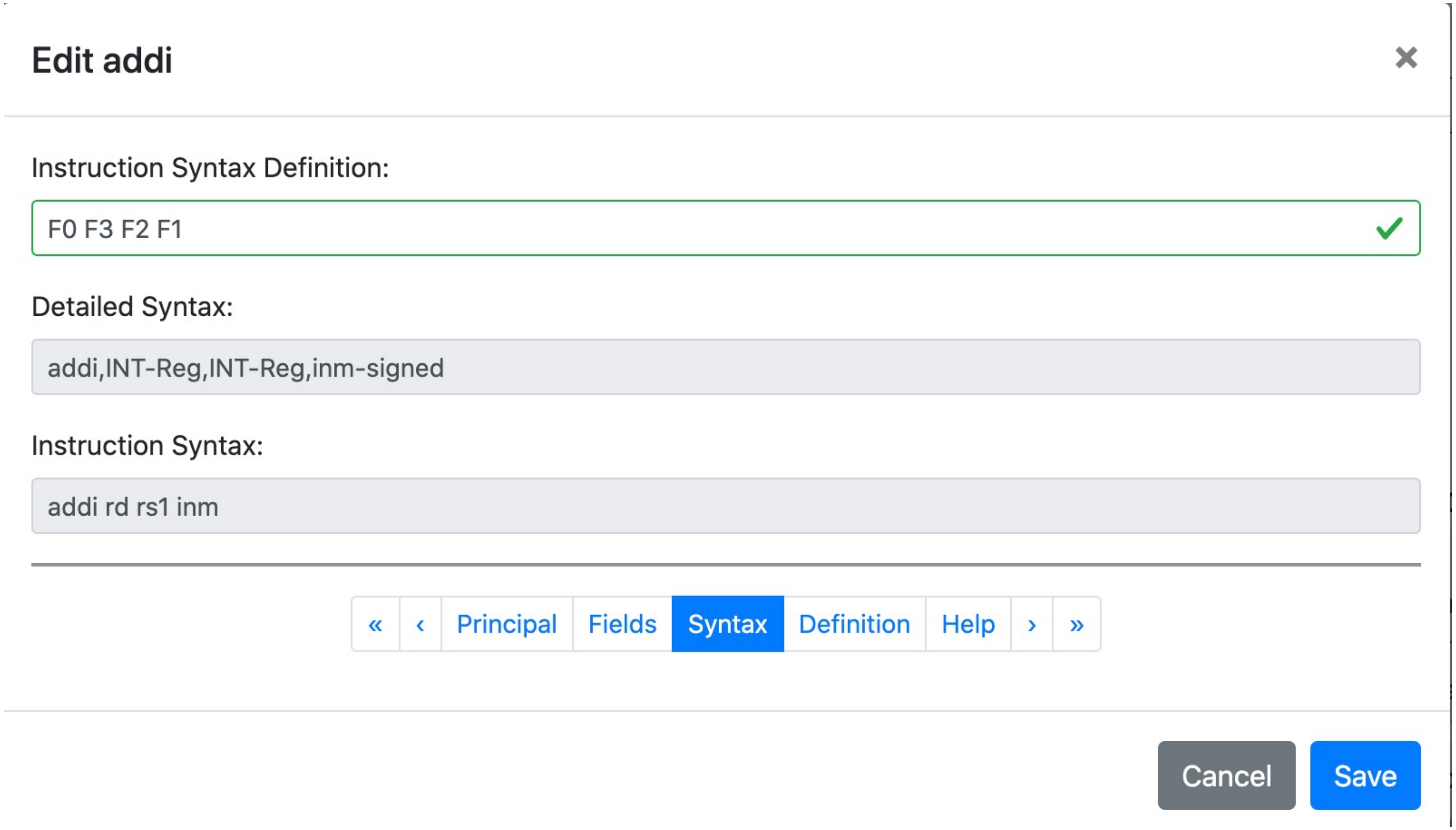


Edit addi ×

	Name:	Type	Break	Start Bit	End Bit	
Field 0	addi	co		6 ✓	0 ✓	0010011 ✓
Field 1	inm ✓	inm-signe ✓	□	31 ✓	20 ✓	
Field 2	rs1 ✓	INT-Reg ✓		19 ✓	15 ✓	
Field 3	rd ✓	INT-Reg ✓		11 ✓	7 ✓	
Field 4	cop ✓	cop ✓		14 ✓	12 ✓	000 ✓

código de operación

Ejemplo de definición de una instrucción (addi)



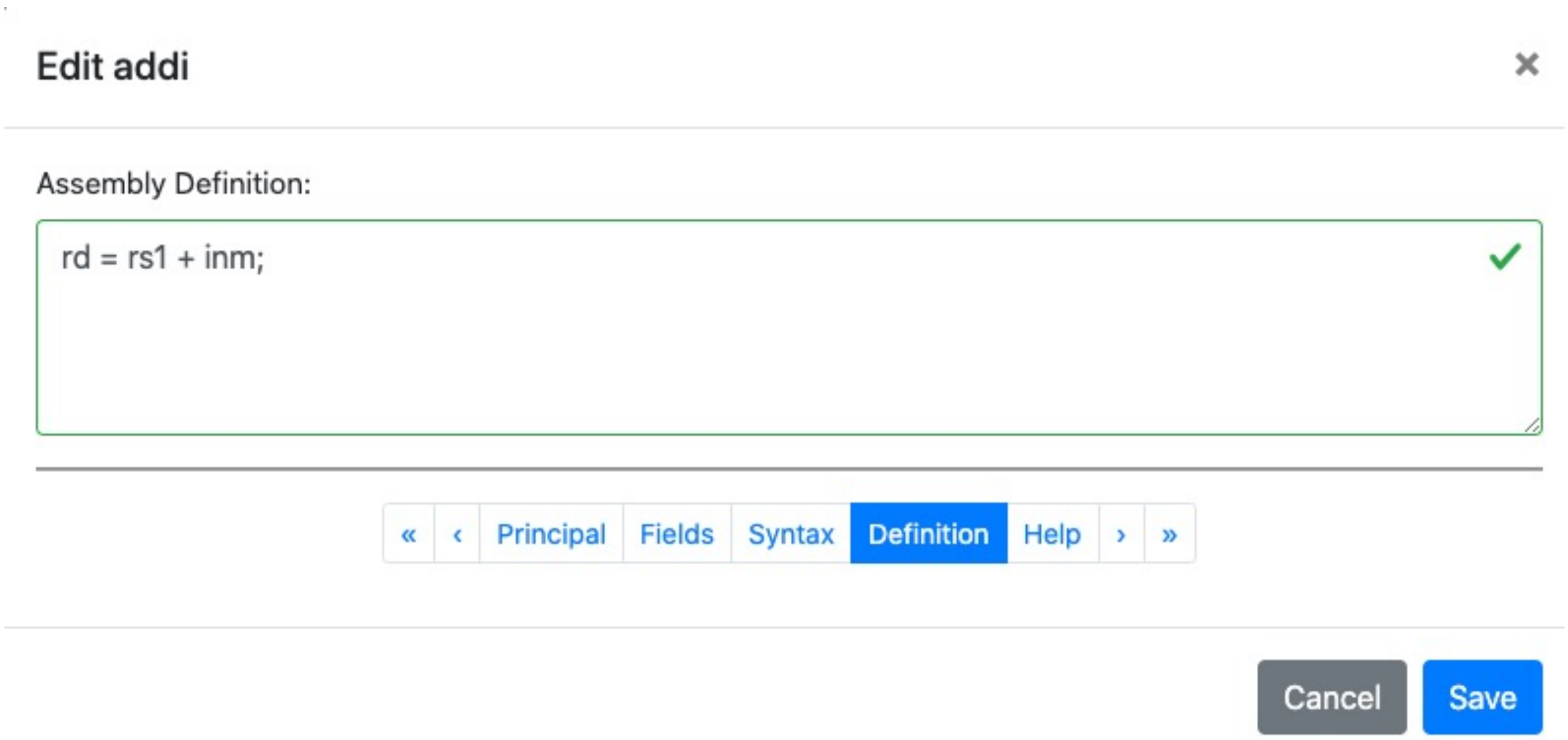
Edit addi

Instruction Syntax Definition: F0 F3 F2 F1 ✓

Detailed Syntax: addi,INT-Reg,INT-Reg,inm-signed

Instruction Syntax: addi rd rs1 inm

Ejemplo de definición de una instrucción (addi)

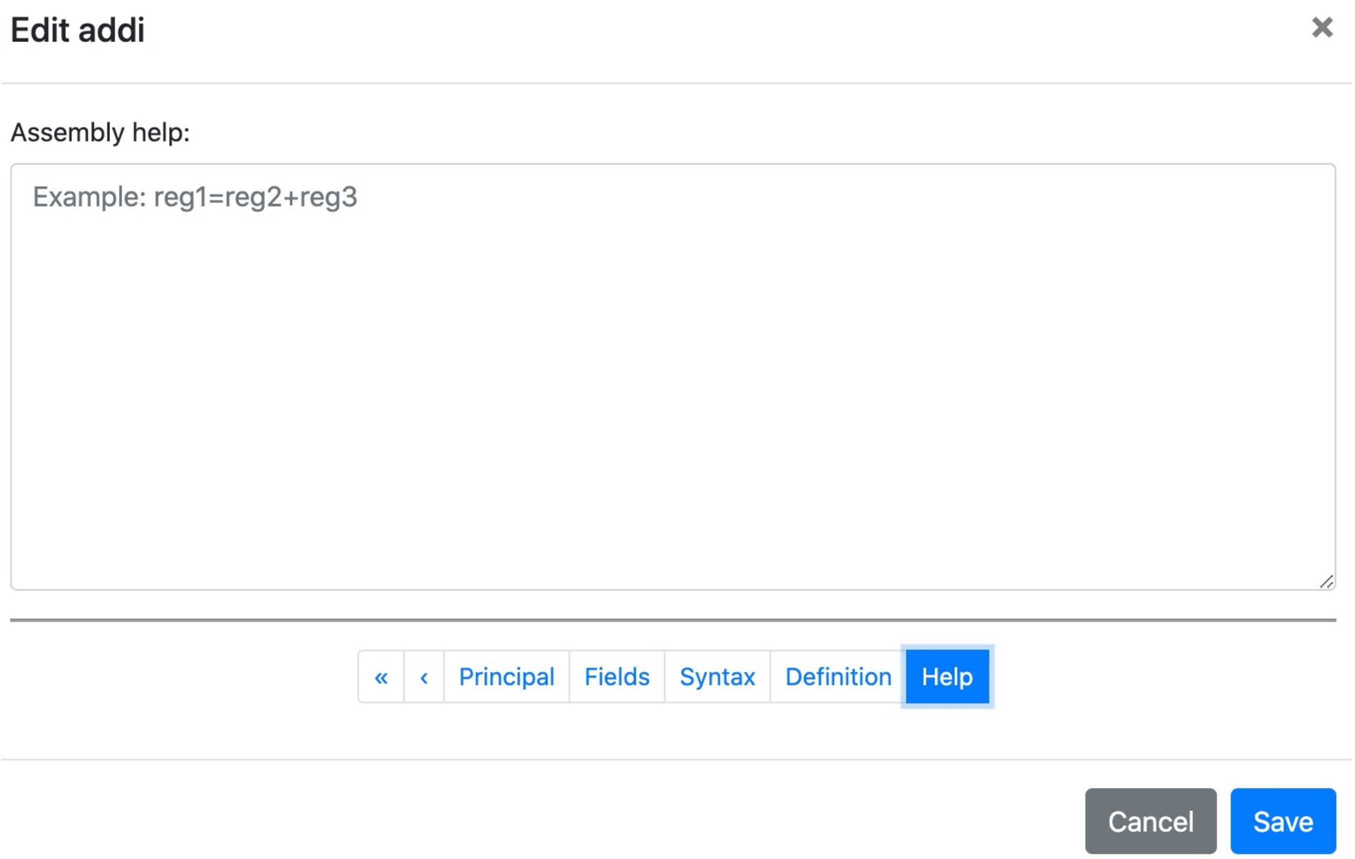


Edit addi

Assembly Definition:

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

Ejemplo de definición de una instrucción (addi)



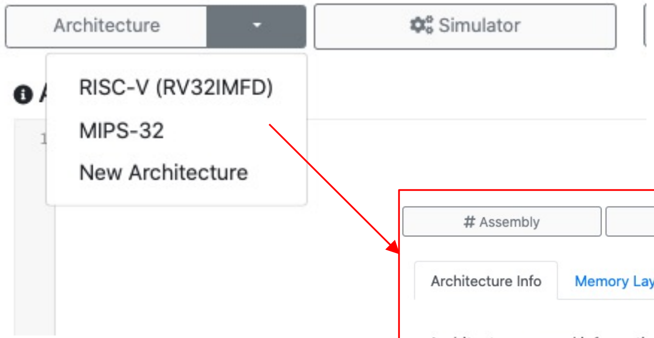
Edit addi

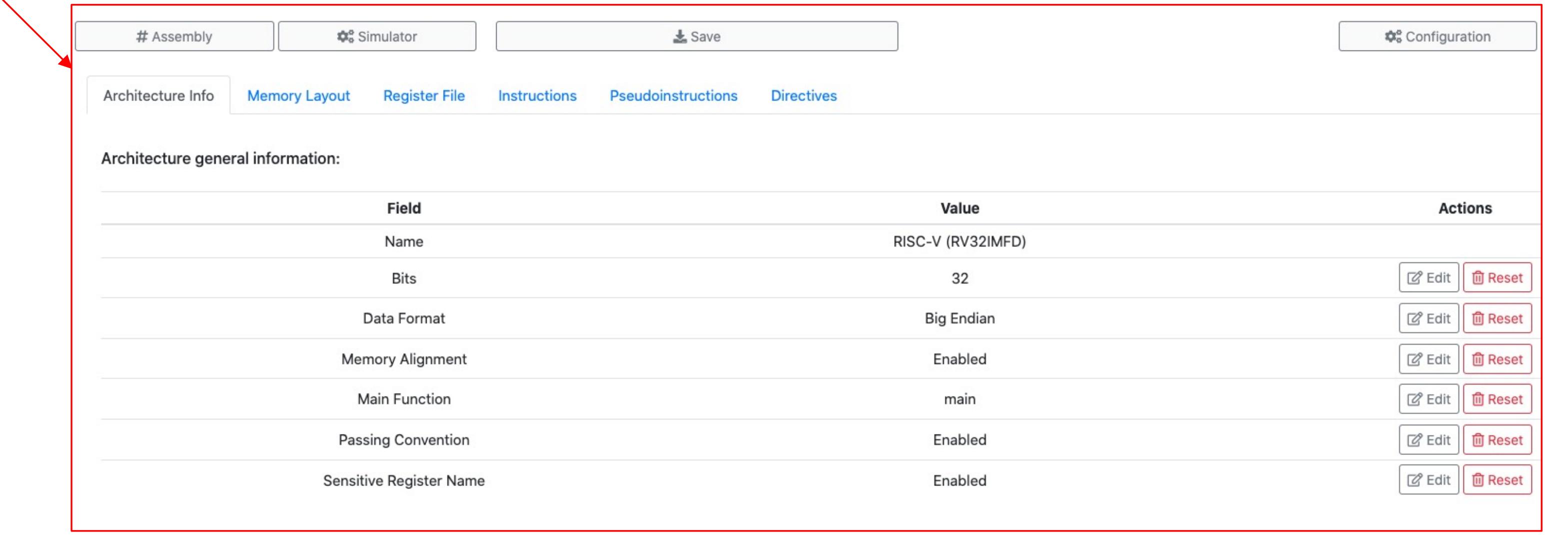
Assembly help:

Example: reg1=reg2+reg3

Creación de una nueva pseudoinstrucción

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

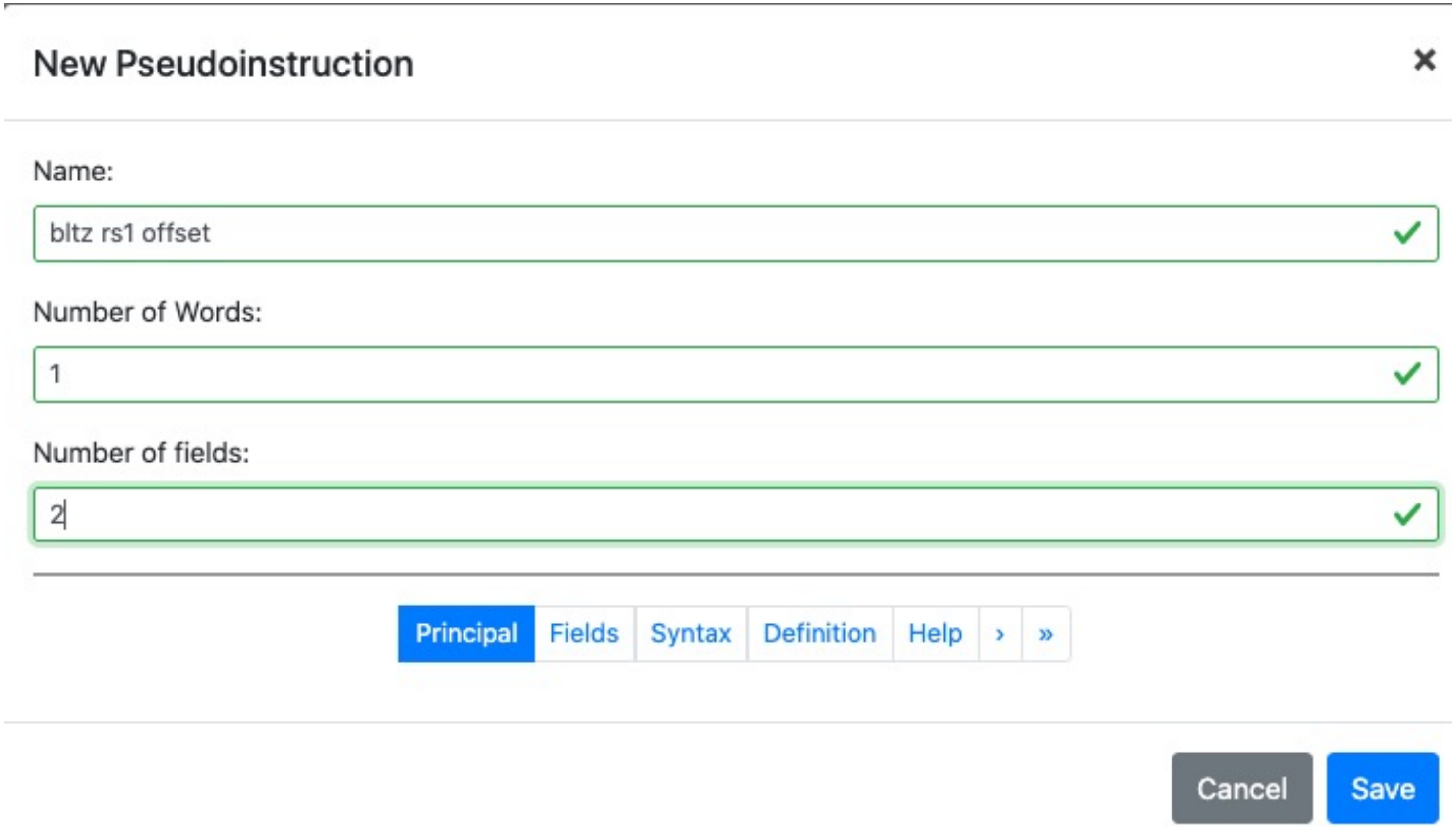




Architecture general information:

Field	Value	Actions
Name	RISC-V (RV32IMFD)	
Bits	32	Edit Reset
Data Format	Big Endian	Edit Reset
Memory Alignment	Enabled	Edit Reset
Main Function	main	Edit Reset
Passing Convention	Enabled	Edit Reset
Sensitive Register Name	Enabled	Edit Reset

Creación de una nueva pseudoinstrucción



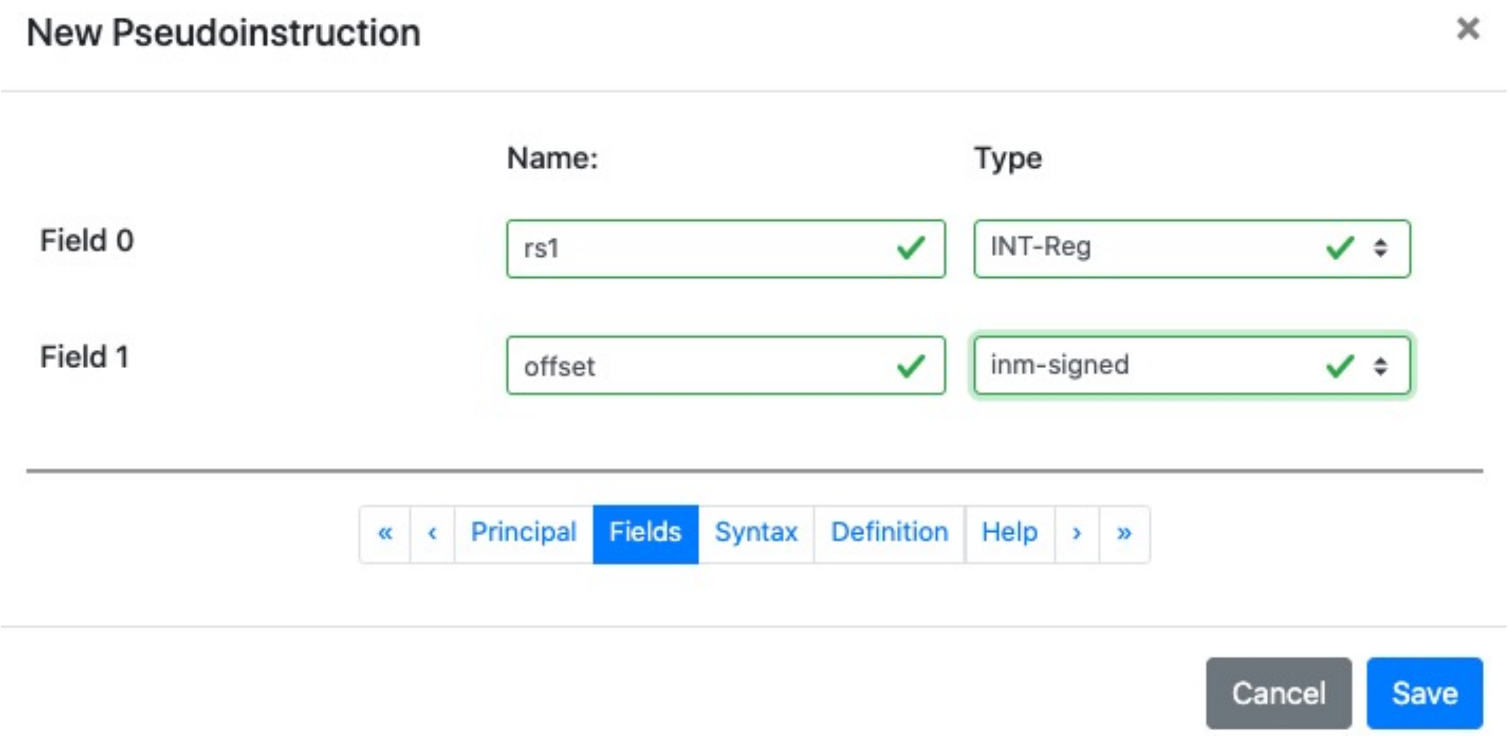
New Pseudoinstruction

Name: bltz rs1 offset

Number of Words: 1

Number of fields: 2

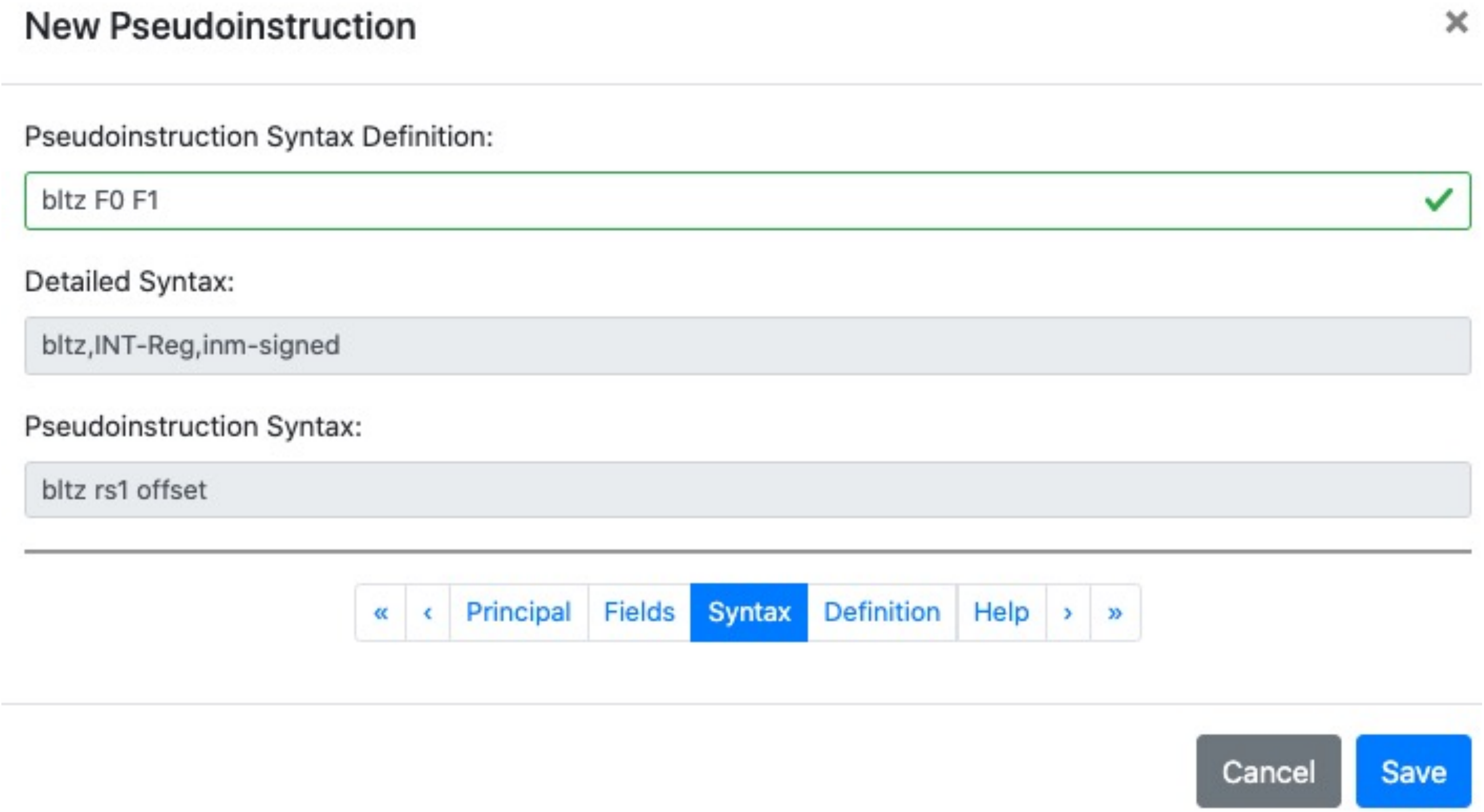
Creación de una nueva pseudoinstrucción



New Pseudoinstruction

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

Creación de una nueva pseudoinstrucción



New Pseudoinstruction

Pseudoinstruction Syntax Definition:

bltz F0 F1 ✓

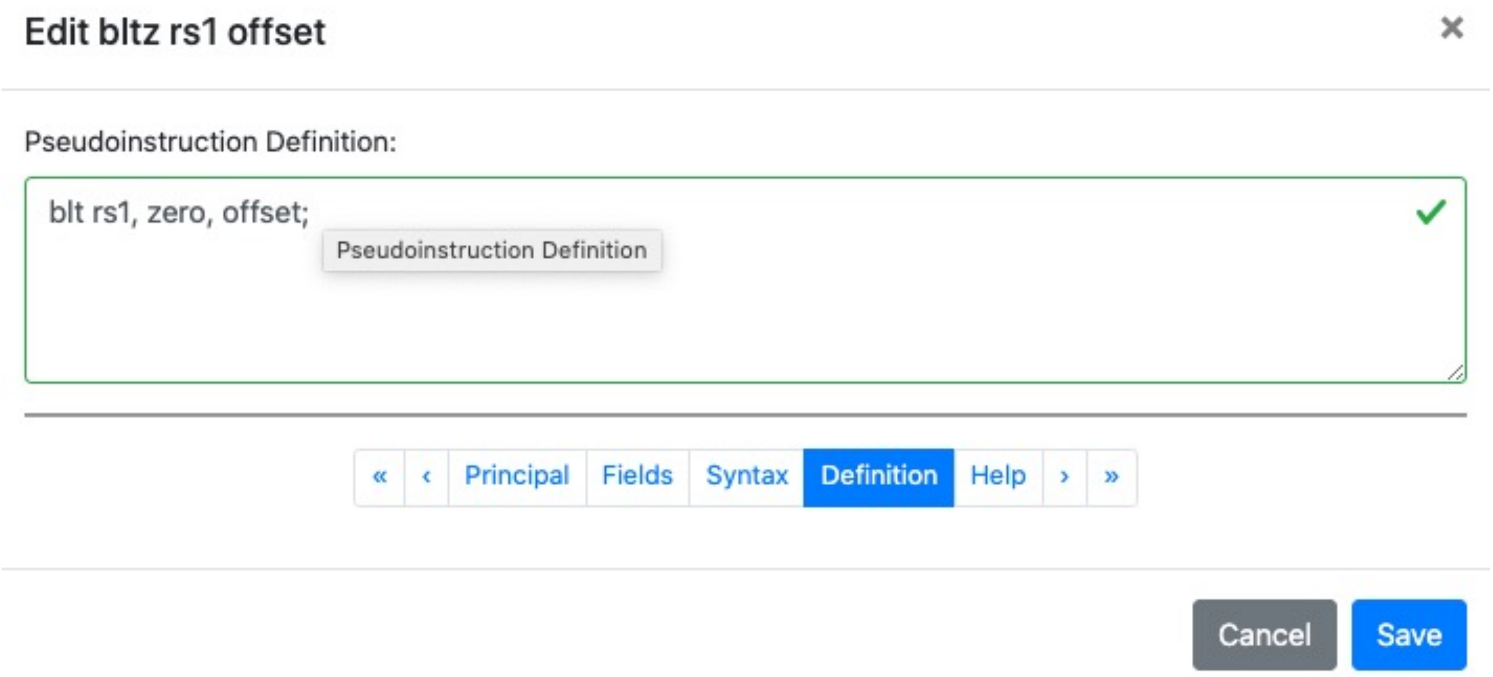
Detailed Syntax:

bltz,INT-Reg,inm-signed

Pseudoinstruction Syntax:

bltz rs1 offset

Creación de una nueva pseudoinstrucción



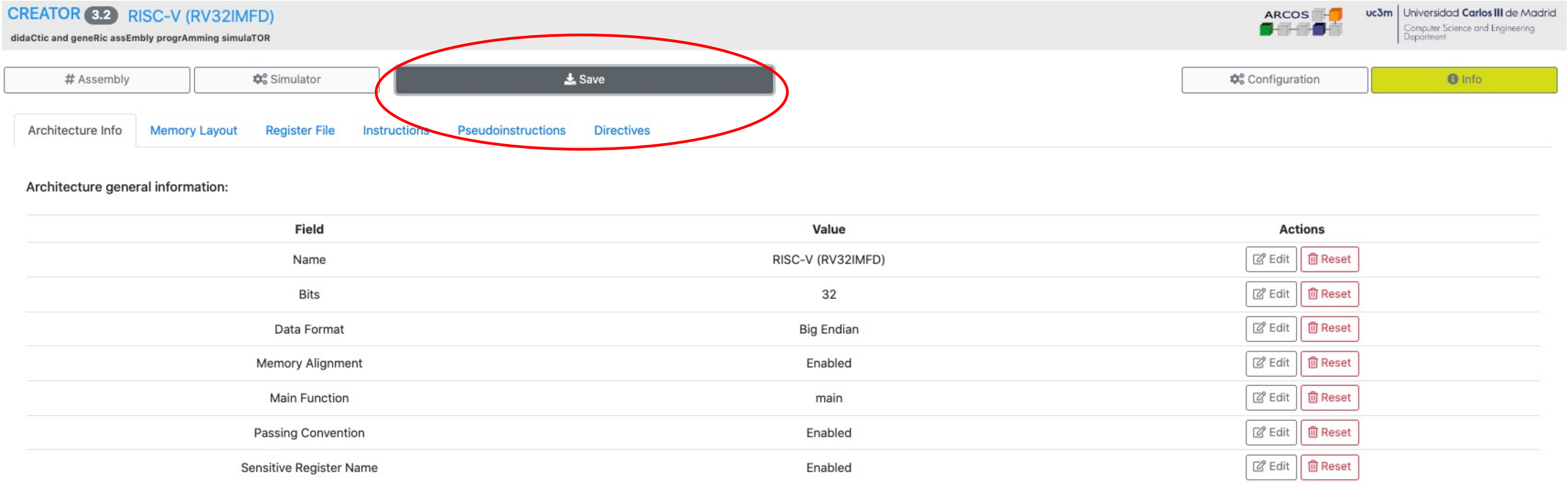
Edit bltz rs1 offset

Pseudoinstruction Definition:

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

×
✓
Pseudoinstruction Definition
« ‹ Principal Fields Syntax Definition Help › »
Cancel
Save

Creación de una nueva pseudoinstrucción

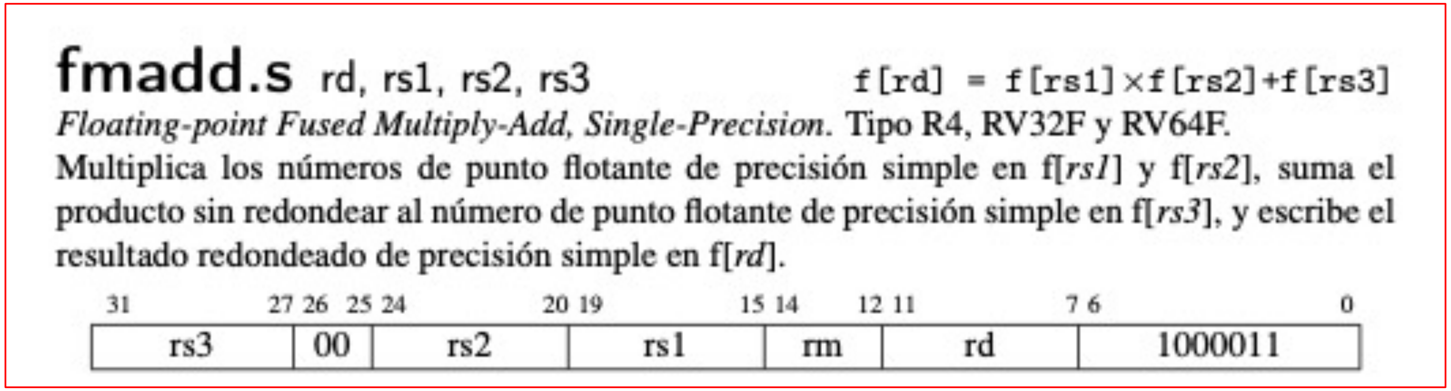

CREATOR 3.2 RISC-V (RV32IMFD)
didaCtic and geneRic assEmbly progrAmming simulaTOR
ARCOS
uc3m Universidad Carlos III de Madrid
Computer Science and Engineering Department
# Assembly
Simulator
Save
Configuration
Info
Architecture Info
Memory Layout
Register File
Instructions
Pseudoinstructions
Directives

Architecture general information:

Field	Value	Actions
Name	RISC-V (RV32IMFD)	Edit Reset
Bits	32	Edit Reset
Data Format	Big Endian	Edit Reset
Memory Alignment	Enabled	Edit Reset
Main Function	main	Edit Reset
Passing Convention	Enabled	Edit Reset
Sensitive Register Name	Enabled	Edit Reset

Creación de una nueva instrucción

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



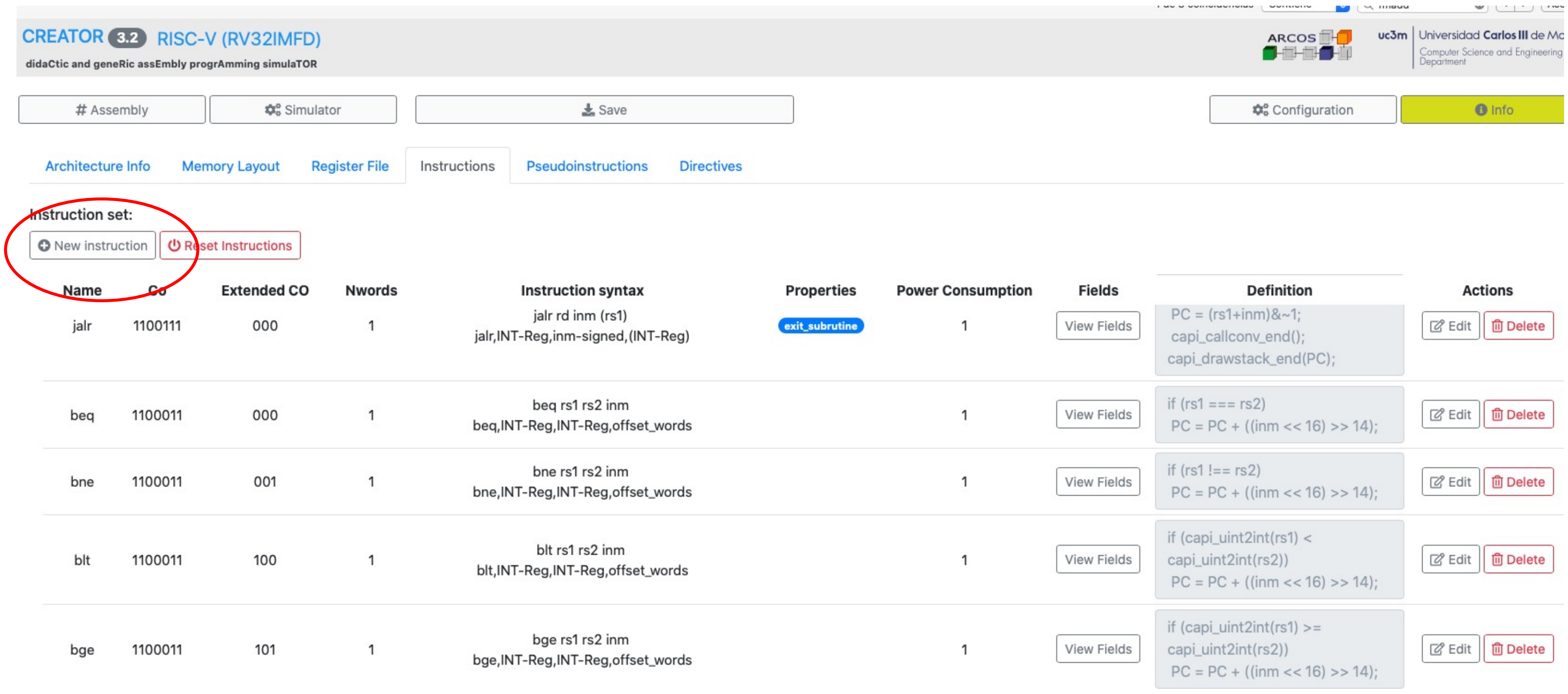
fmadd.s rd, rs1, rs2, rs3 `f[rd] = f[rs1]×f[rs2]+f[rs3]`

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

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

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

Creación de una nueva instrucción



Instruction set:

Name	Co	Extended CO	Nwords	Instruction syntax	Properties	Power Consumption	Fields	Definition	Actions

jalr	1100111	000	1	jalr rd inm (rs1)
				jalr,INT-Reg,inm-signed,(INT-Reg)
					exit_subrutine	1	View Fields	PC = (rs1+inm)&~1;
								capi_callconv_end();
								capi_drawstack_end(PC);
									Edit Delete

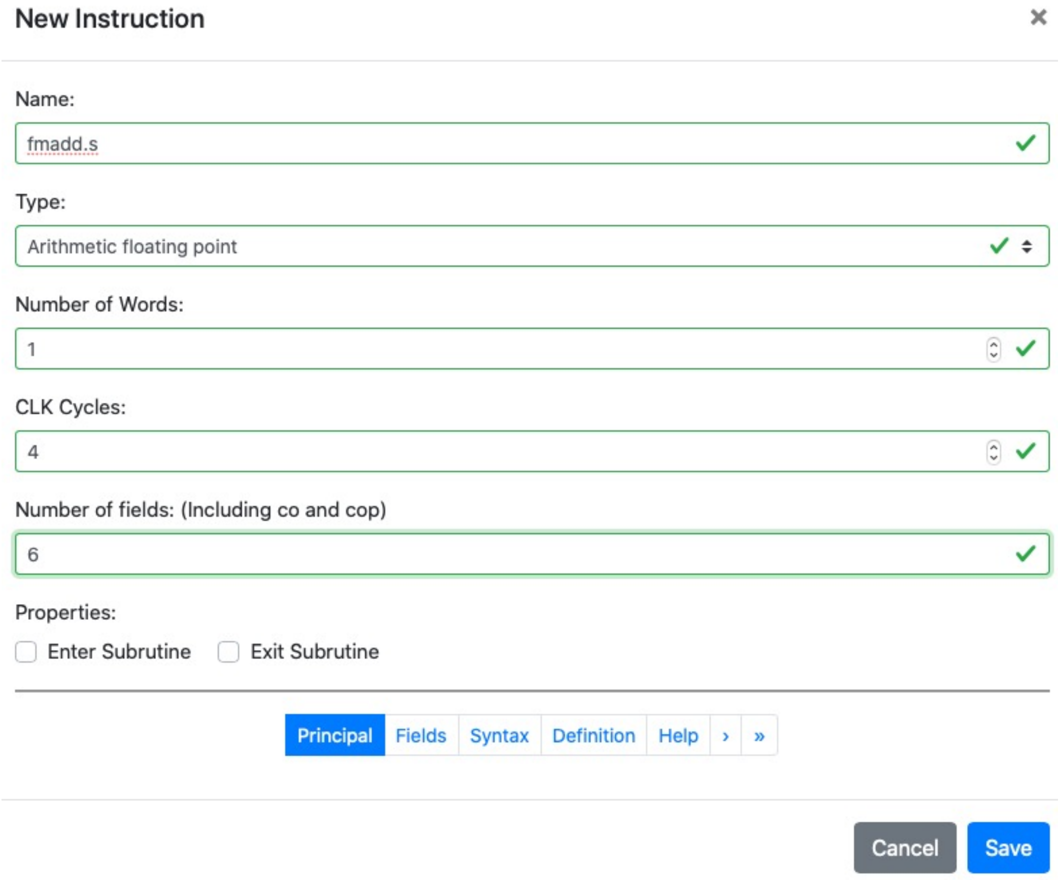
beq	1100011	000	1	beq rs1 rs2 inm
				beq,INT-Reg,INT-Reg,offset_words
						1	View Fields	if (rs1 === rs2)
								PC = PC + ((inm << 16) >> 14);
									Edit Delete

bne	1100011	001	1	bne rs1 rs2 inm
				bne,INT-Reg,INT-Reg,offset_words
						1	View Fields	if (rs1 !== rs2)
								PC = PC + ((inm << 16) >> 14);
									Edit Delete

blt	1100011	100	1	blt rs1 rs2 inm
				blt,INT-Reg,INT-Reg,offset_words
						1	View Fields	if (capi_uint2int(rs1) <
								capi_uint2int(rs2))
								PC = PC + ((inm << 16) >> 14);
									Edit Delete

bge	1100011	101	1	bge rs1 rs2 inm
				bge,INT-Reg,INT-Reg,offset_words
						1	View Fields	if (capi_uint2int(rs1) >=
								capi_uint2int(rs2))
								PC = PC + ((inm << 16) >> 14);
									Edit Delete

Nueva instrucción: `fmadd.s`



New Instruction ✕

Name: fmadd.s ✓

Type: Arithmetic floating point ✓

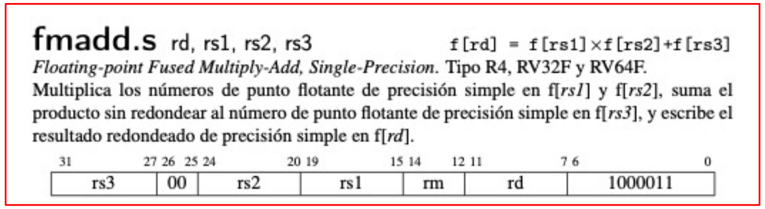
Number of Words: 1 ✓

CLK Cycles: 4 ✓

Number of fields: (Including co and cop) 6 ✓

Properties:

□ Enter Subrutine □ Exit Subrutine



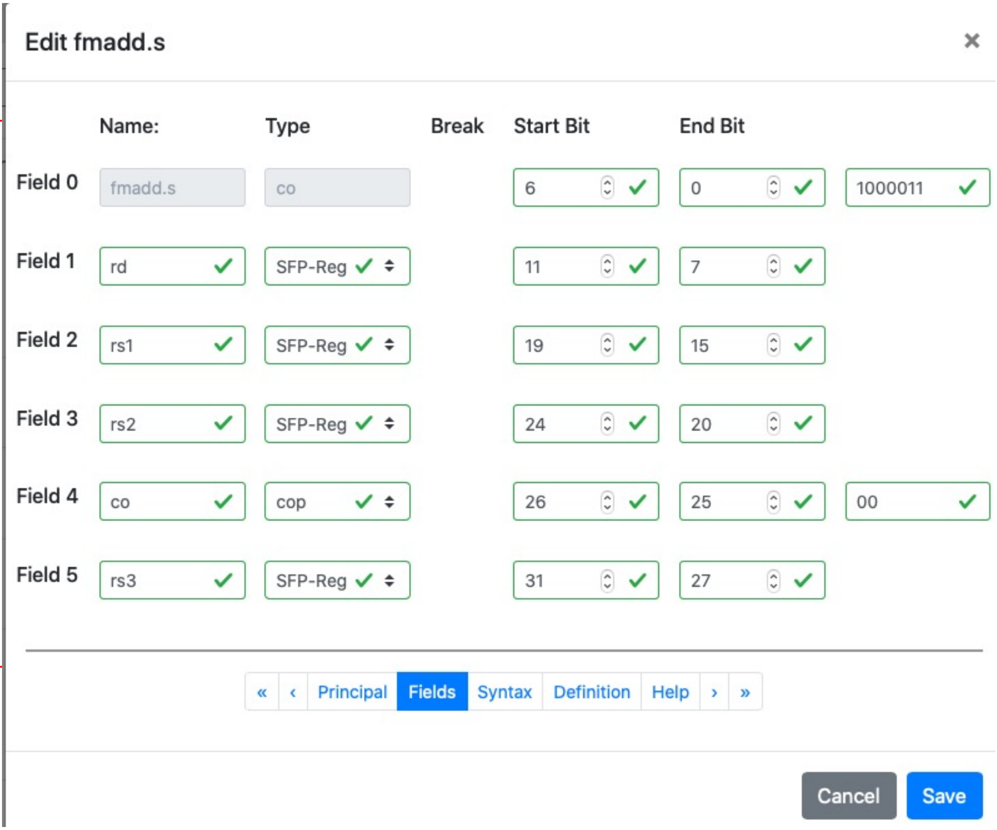
fmadd.s rd, rs1, rs2, rs3 `f[rd] = f[rs1]×f[rs2]+f[rs3]`

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

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

Nueva instrucción: `fmadd.s`

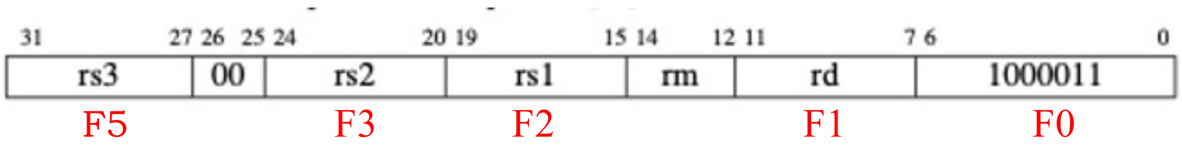
31 27	26 25	24 20	19 15	14 12	11 7	6 0
rs3	00	rs2	rs1	rm	rd	1000011



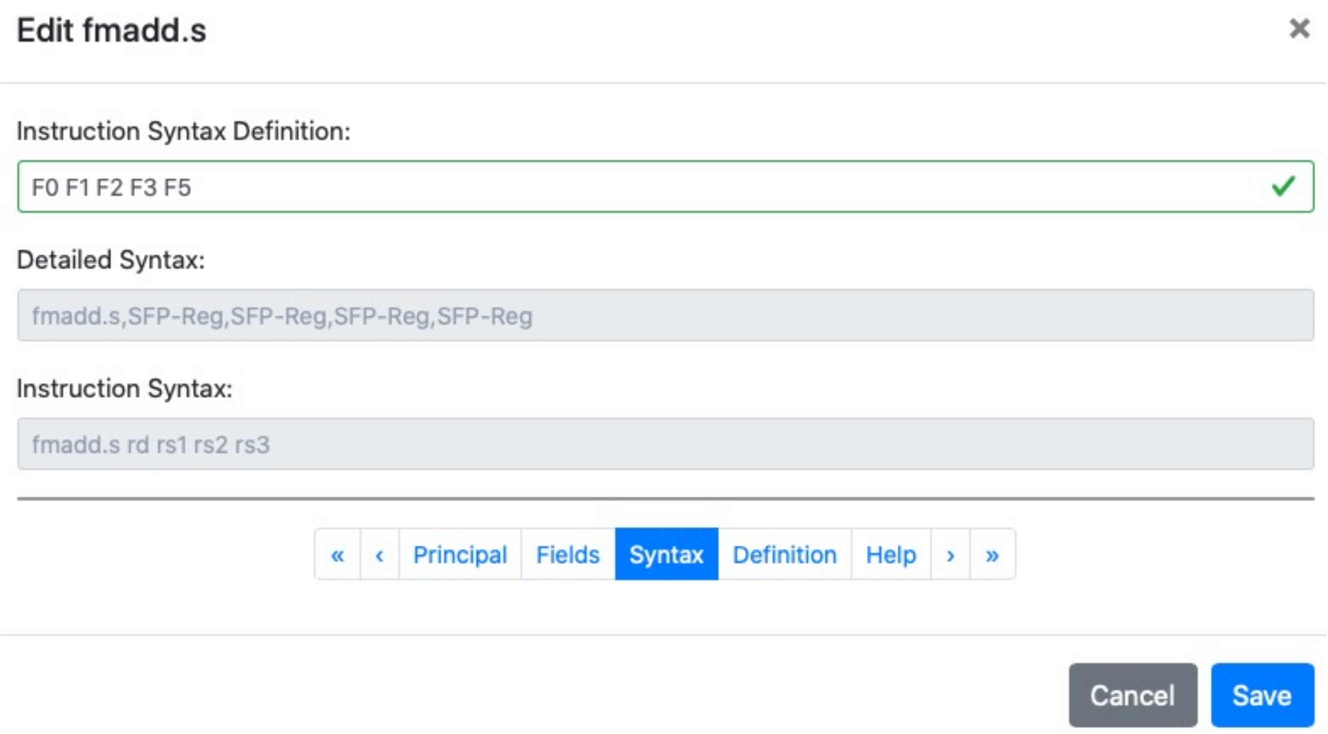
Edit fmadd.s

	Name:	Type	Break	Start Bit	End Bit	
Field 0	fmadd.s	co		6 ✓	0 ✓	1000011 ✓
Field 1	rd ✓	SFP-Reg ✓		11 ✓	7 ✓	
Field 2	rs1 ✓	SFP-Reg ✓		19 ✓	15 ✓	
Field 3	rs2 ✓	SFP-Reg ✓		24 ✓	20 ✓	
Field 4	co ✓	cop ✓		26 ✓	25 ✓	00 ✓
Field 5	rs3 ✓	SFP-Reg ✓		31 ✓	27 ✓	

Nueva instrucción: `fmadd.s`



31 27	26 25	24 20	19 15	14 12	11 7	6 0
rs3	00	rs2	rs1	rm	rd	1000011
F5		F3	F2		F1	F0



Edit fmadd.s

Instruction Syntax Definition:

F0 F1 F2 F3 F5 ✓

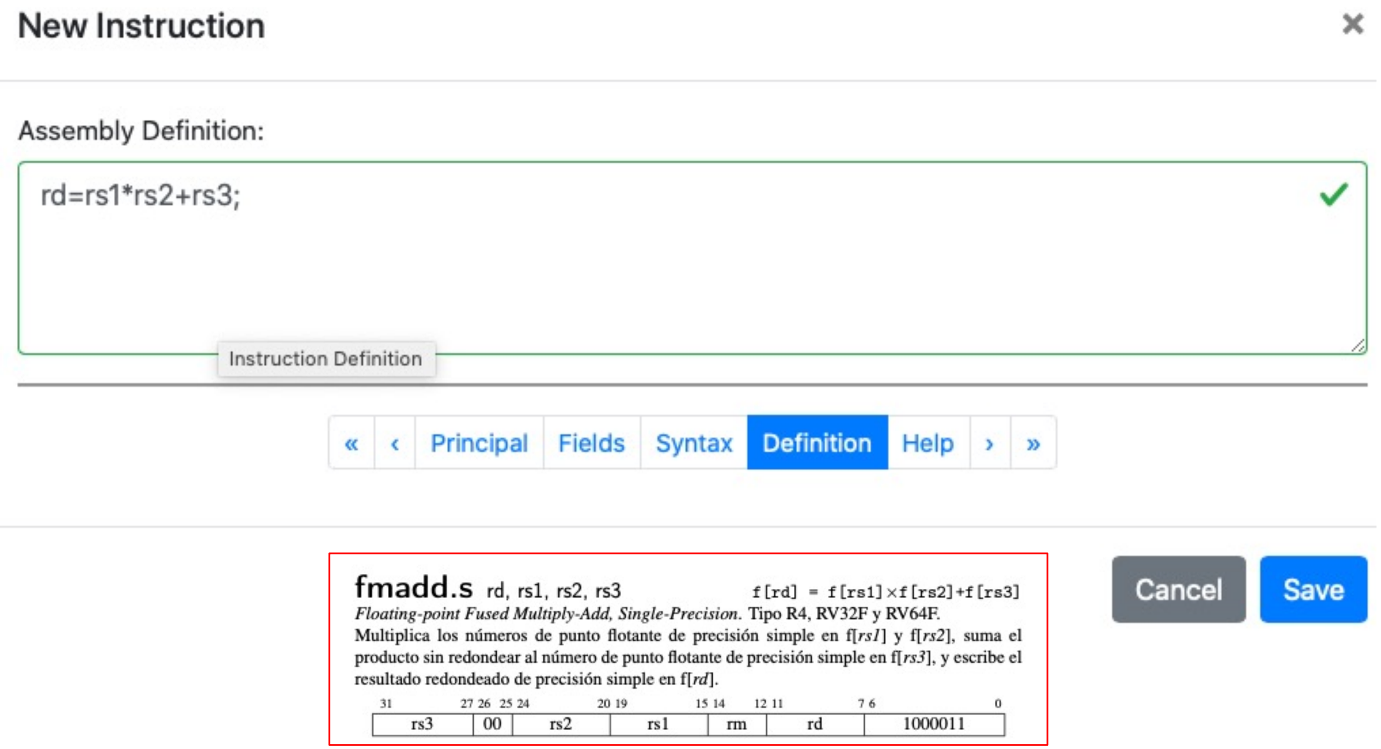
Detailed Syntax:

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

Instruction Syntax:

fmadd.s rd rs1 rs2 rs3

Nueva instrucción: `fmadd.s`



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

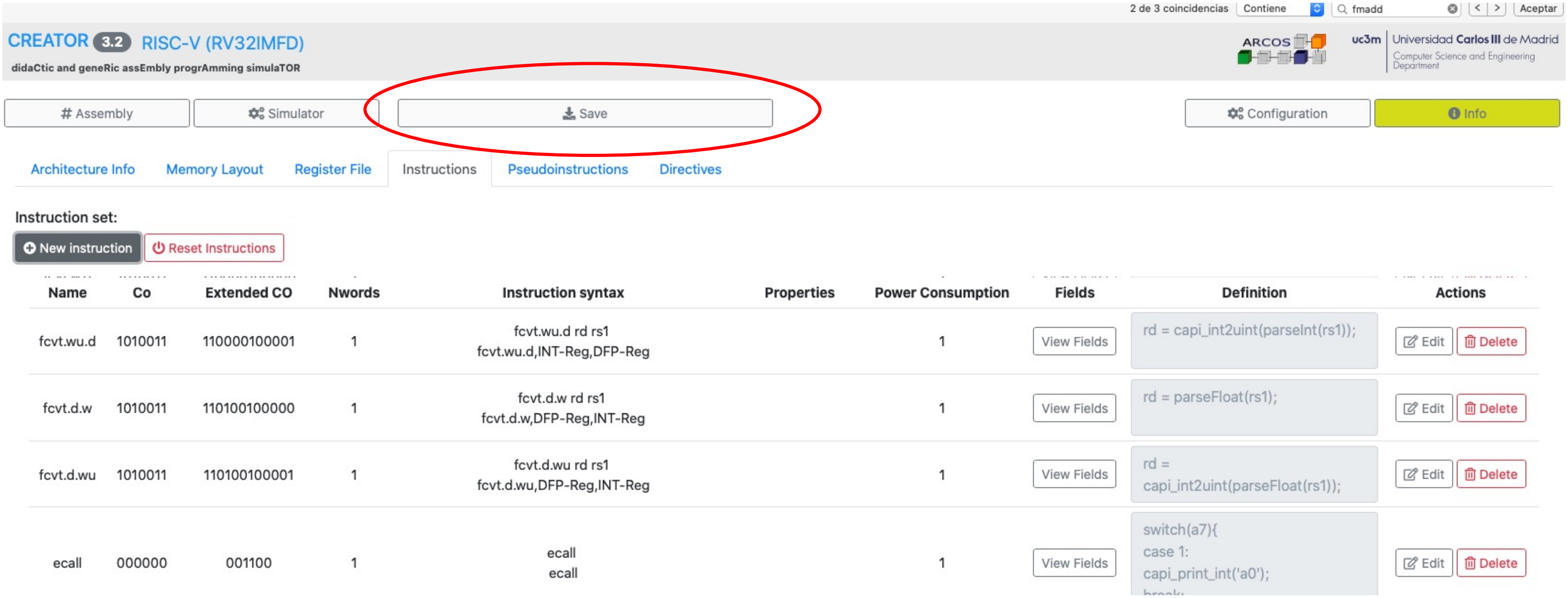
fmadd.s rd, rs1, rs2, rs3 f[rd] = f[rs1]×f[rs2]+f[rs3]

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

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

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

Creación de una nueva instrucción

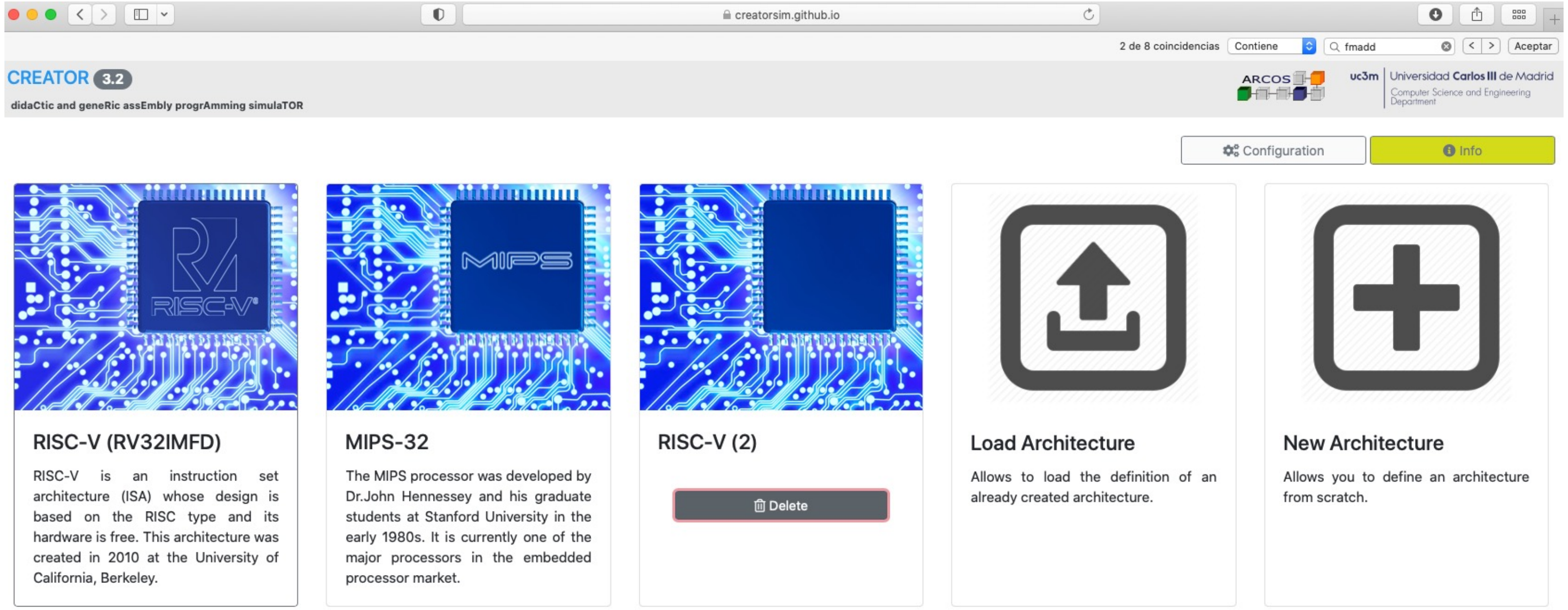


Instruction set:

Name	Co	Extended CO	Nwords	Instruction syntax	Properties	Power Consumption	Fields	Definition	Actions
fcvt.wu.d	1010011	110000100001	1	fcvt.wu.d rd rs1 fcvt.wu.d,INT-Reg,DFP-Reg		1	View Fields	rd = capi_int2uint(parseInt(rs1));	Edit Delete
fcvt.d.w	1010011	110100100000	1	fcvt.d.w rd rs1 fcvt.d.w,DFP-Reg,INT-Reg		1	View Fields	rd = parseFloat(rs1);	Edit Delete
fcvt.d.wu	1010011	110100100001	1	fcvt.d.wu rd rs1 fcvt.d.wu,DFP-Reg,INT-Reg		1	View Fields	rd = capi_int2uint(parseFloat(rs1));	Edit Delete

ecall	000000	001100	1	ecall
				ecall
						1	View Fields	switch(a7){
								case 1:
								capi_print_int('a0');
									Edit Delete

Nueva arquitectura


creatorsim.github.io
2 de 8 coincidencias
Contiene
fmadd
Aceptar
CREATOR 3.2
didaCtic and geneRic assEmbly progrAmming simulaTOR
ARCOS
uc3m Universidad Carlos III de Madrid
Computer Science and Engineering Department
Configuration
Info
RISC-V
MIPS
Delete

RISC-V (RV32IMFD)

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

MIPS-32

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

RISC-V (2)

Load Architecture

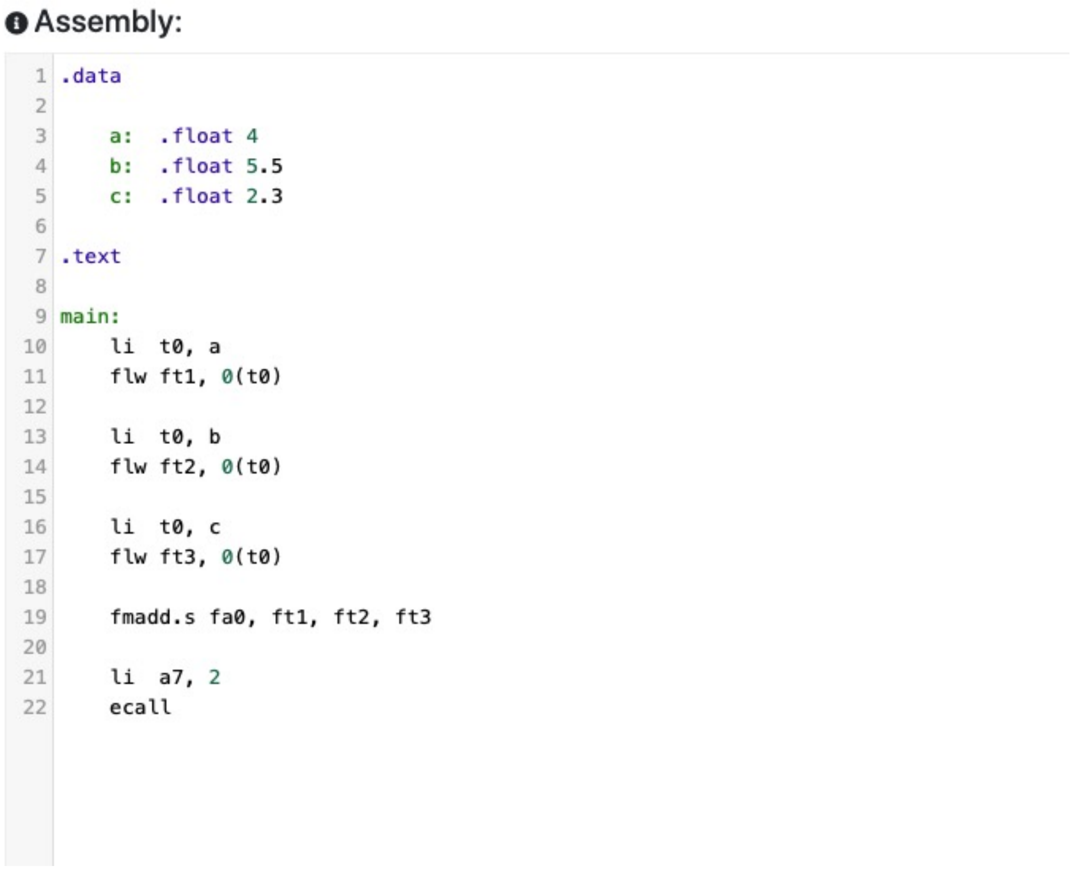
Allows to load the definition of an already created architecture.

New Architecture

Allows you to define an architecture from scratch.

Ejemplo

ejemplo



Assembly:

```
.data

    a:  .float 4
    b:  .float 5.5
    c:  .float 2.3

.text

main:
    li  t0, a
    flw ft1, 0(t0)

    li  t0, b
    flw ft2, 0(t0)

    li  t0, c
    flw ft3, 0(t0)

    fmadd.s fa0, ft1, ft2, ft3

    li  a7, 2
    ecall
```

API para la definición de instrucciones

- API de ayuda para definición de instrucciones
- Instrucción: `lw rd inm (rs1)`
- Definición:

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

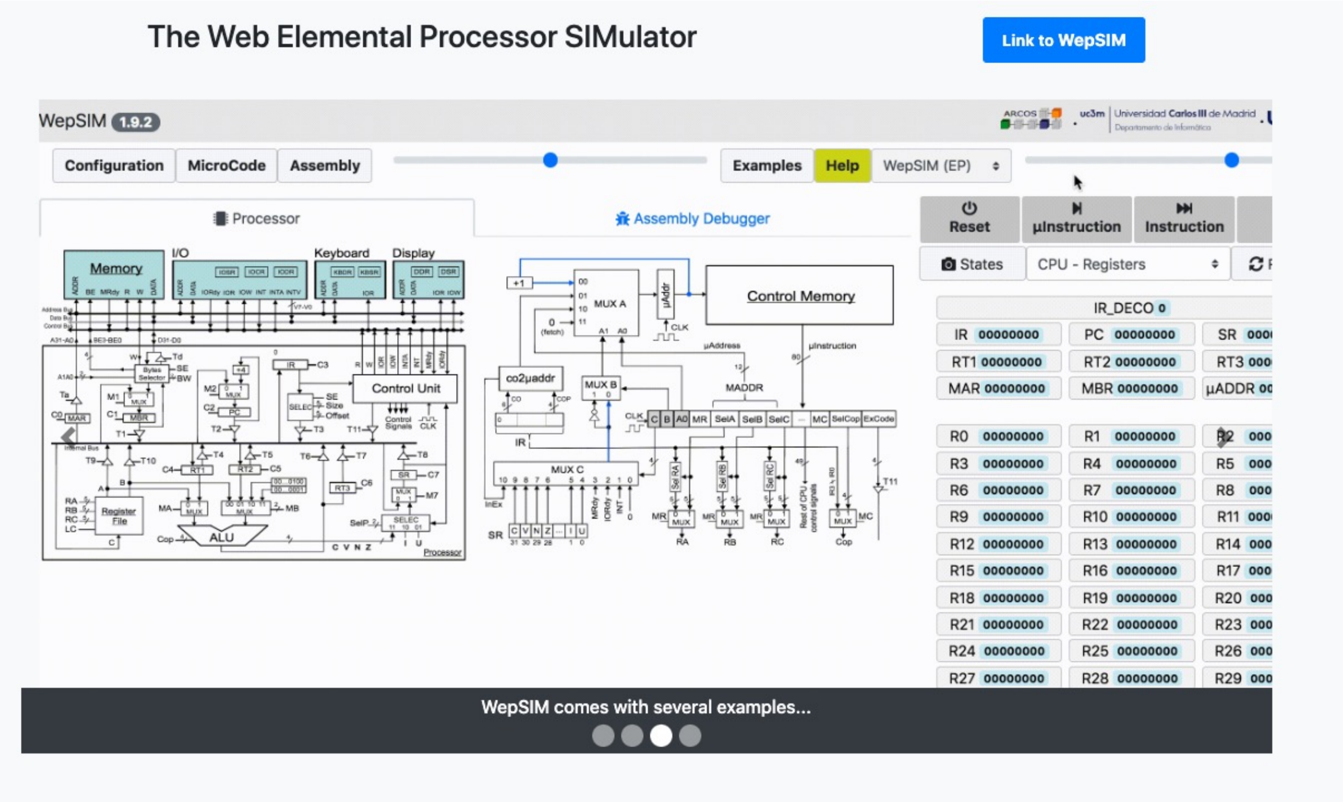
- Instrucción: `sb rd inm (rs1)`
- Definición:

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

Extensiones futuras

- Simulador de caché
- Registros e instrucciones vectoriales
- Simulador de pipeline
- ….

Otro simulador: WepSim



The Web Elemental Processor SIMulator

IR_DECO 0

IR 00000000 PC 00000000 SR 000

RT1 00000000 RT2 00000000 RT3 000

MAR 00000000 MBR 00000000 µADDR 00

R0 00000000 R1 00000000 R2 000

R3 00000000 R4 00000000 R5 000

R6 00000000 R7 00000000 R8 000

R9 00000000 R10 00000000 R11 000

R12 00000000 R13 00000000 R14 000

R15 00000000 R16 00000000 R17 000

R18 00000000 R19 00000000 R20 000

R21 00000000 R22 00000000 R23 000

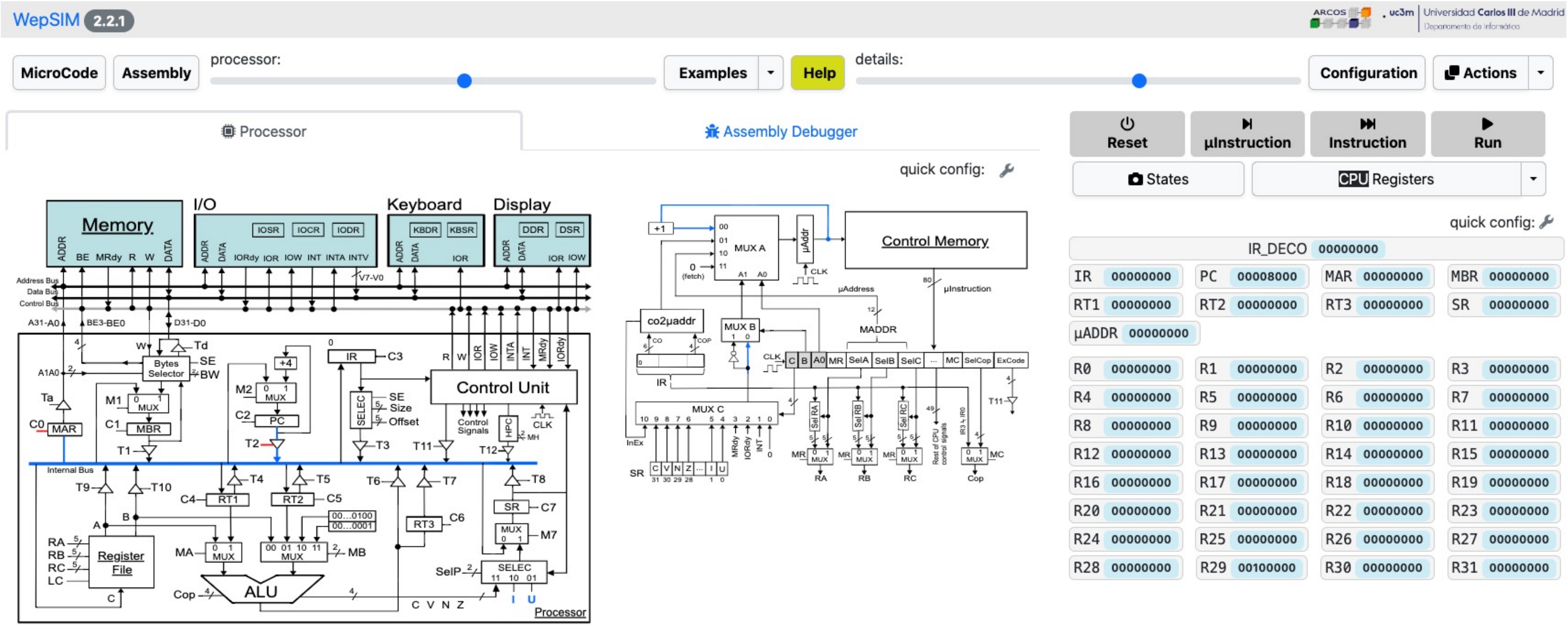
R24 00000000 R25 00000000 R26 000

R27 00000000 R28 00000000 R29 000

WepSIM comes with several examples...

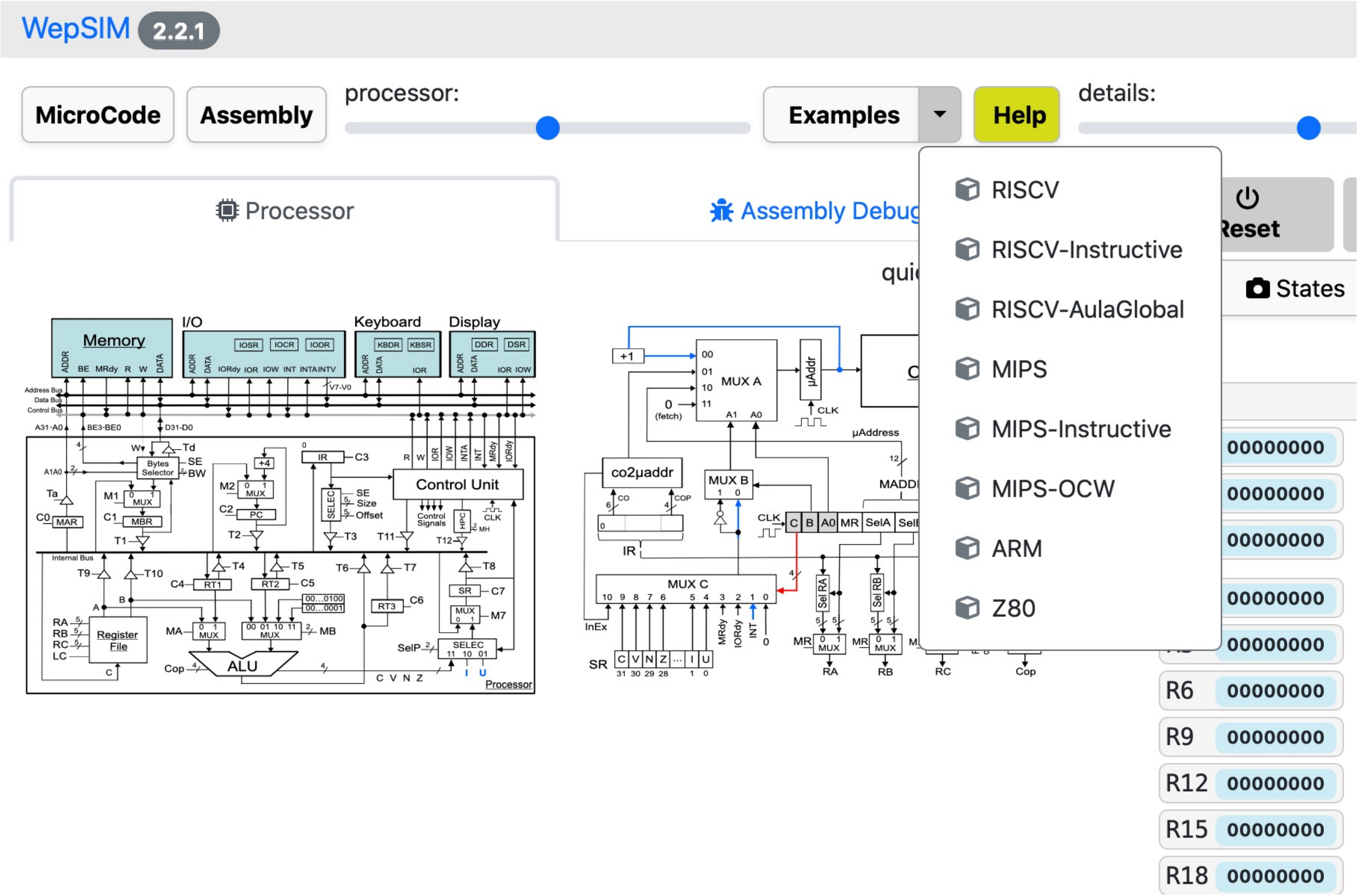
https://wepsim.github.io

WepSim

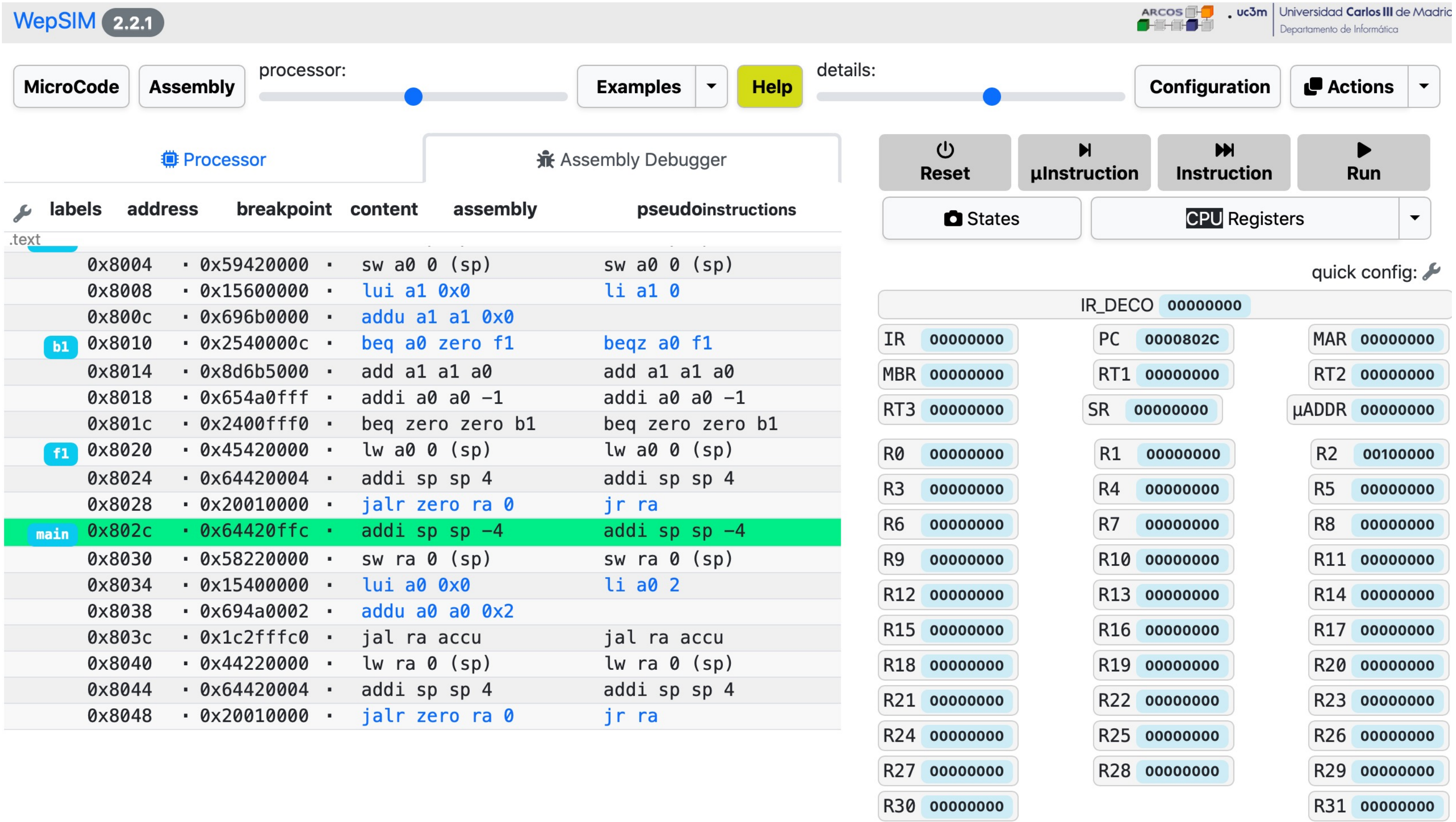


IR_DECO	00000000						
IR	00000000	PC	00008000	MAR	00000000	MBR	00000000
RT1	00000000	RT2	00000000	RT3	00000000	SR	00000000
µADDR	00000000						
R0	00000000	R1	00000000	R2	00000000	R3	00000000
R4	00000000	R5	00000000	R6	00000000	R7	00000000
R8	00000000	R9	00000000	R10	00000000	R11	00000000
R12	00000000	R13	00000000	R14	00000000	R15	00000000
R16	00000000	R17	00000000	R18	00000000	R19	00000000
R20	00000000	R21	00000000	R22	00000000	R23	00000000
R24	00000000	R25	00000000	R26	00000000	R27	00000000
R28	00000000	R29	00100000	R30	00000000	R31	00000000

Ejemplos de juegos de instrucciones



Ejecución de un programa (ensamblador)



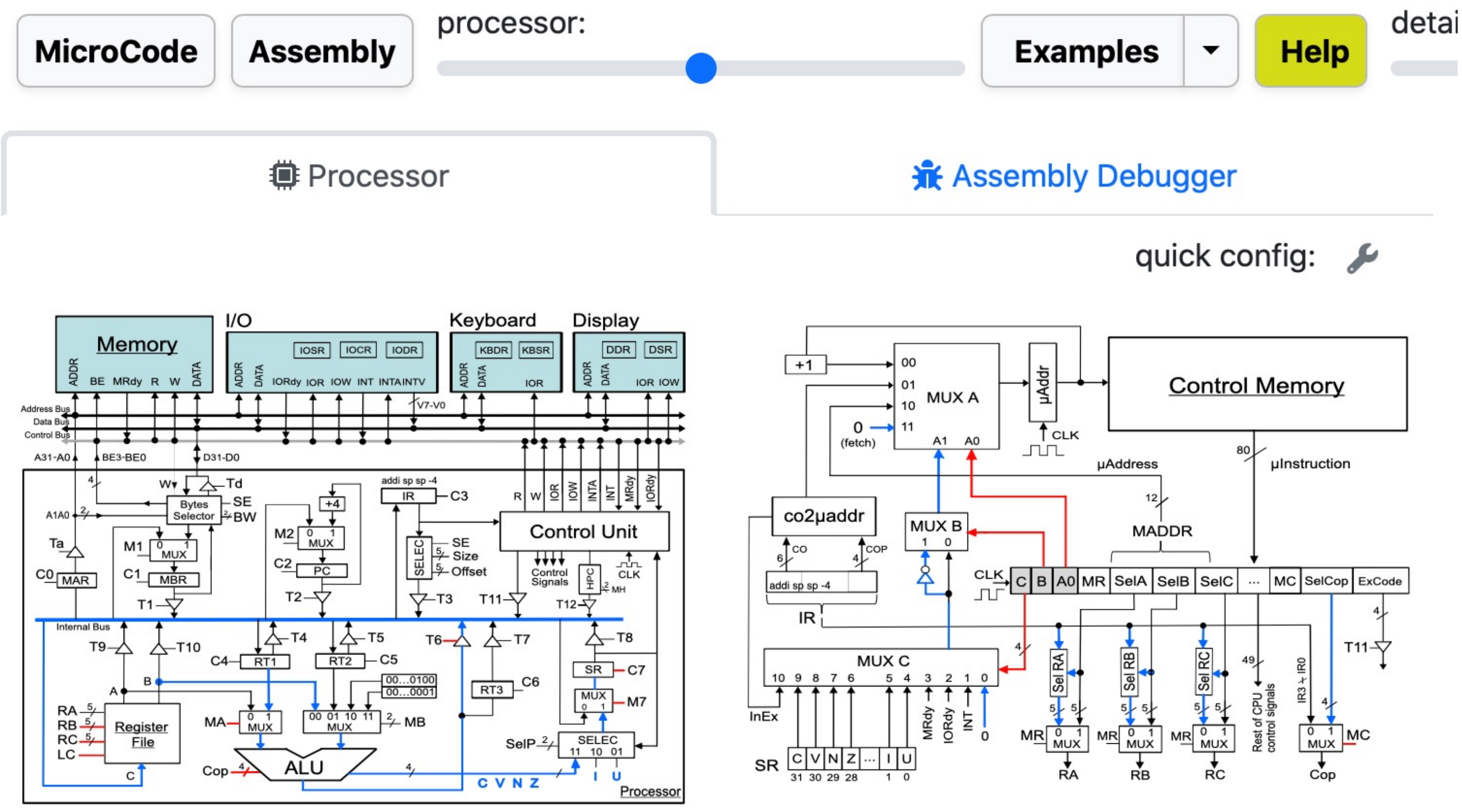
.text

labels	address	breakpoint	content	assembly	pseudoinstructions
	0x8004	·	0x59420000 ·	sw a0 0 (sp)	sw a0 0 (sp)
	0x8008	·	0x15600000 ·	lui a1 0x0	li a1 0
	0x800c	·	0x696b0000 ·	addu a1 a1 0x0	
b1	0x8010	·	0x2540000c ·	beq a0 zero f1	beqz a0 f1
	0x8014	·	0x8d6b5000 ·	add a1 a1 a0	add a1 a1 a0
	0x8018	·	0x654a0fff ·	addi a0 a0 -1	addi a0 a0 -1
	0x801c	·	0x2400fff0 ·	beq zero zero b1	beq zero zero b1
f1	0x8020	·	0x45420000 ·	lw a0 0 (sp)	lw a0 0 (sp)
	0x8024	·	0x64420004 ·	addi sp sp 4	addi sp sp 4
	0x8028	·	0x20010000 ·	jalr zero ra 0	jr ra
main	0x802c	·	0x64420ffc ·	addi sp sp -4	addi sp sp -4
	0x8030	·	0x58220000 ·	sw ra 0 (sp)	sw ra 0 (sp)
	0x8034	·	0x15400000 ·	lui a0 0x0	li a0 2
	0x8038	·	0x694a0002 ·	addu a0 a0 0x2	
	0x803c	·	0x1c2fffc0 ·	jal ra accu	jal ra accu
	0x8040	·	0x44220000 ·	lw ra 0 (sp)	lw ra 0 (sp)
	0x8044	·	0x64420004 ·	addi sp sp 4	addi sp sp 4
	0x8048	·	0x20010000 ·	jalr zero ra 0	jr ra

IR_DECO 00000000

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

Ejecución del programa (procesador)



uc3m | Universidad Carlos III de Madrid

Grupo ARCOS

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

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

Félix García Carballeira

felix.garcia@uc3m.es