

ESTRUCTURA DE COMPUTADORES

Práctica 1

Introducción al laboratorio

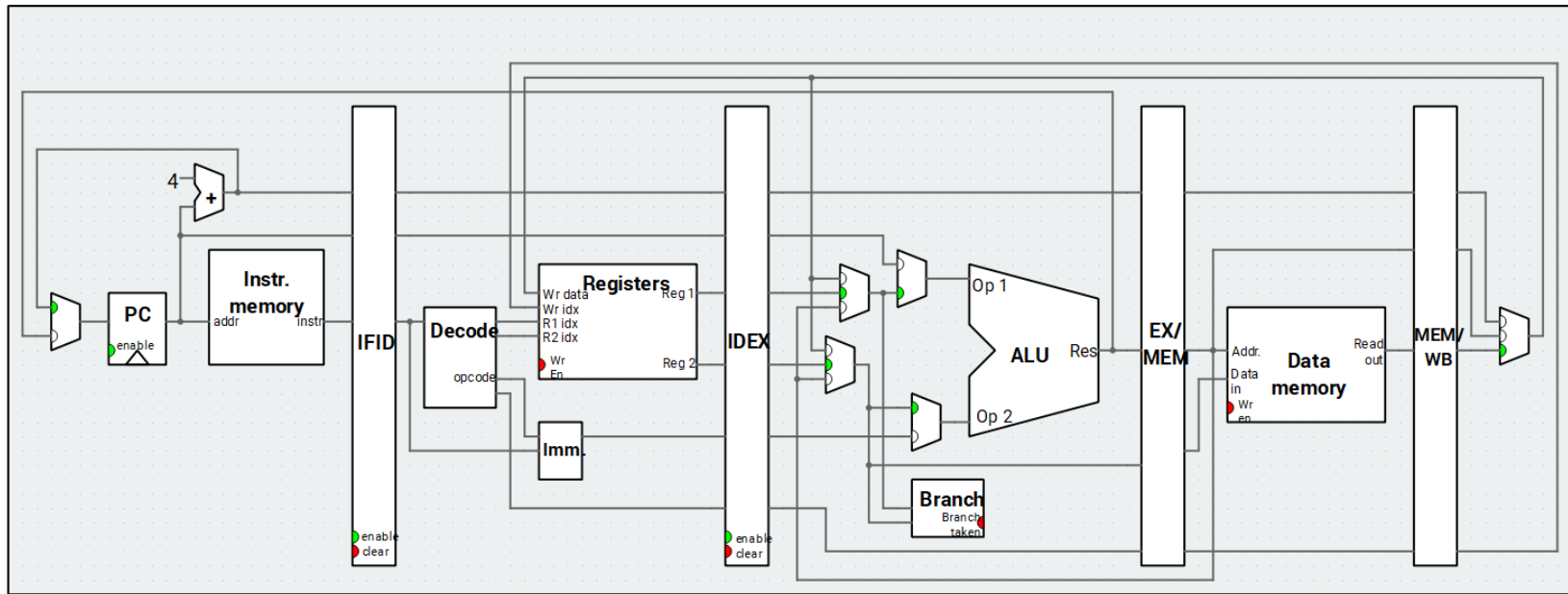
RISC-V

- **RISC-V** (pronunciado “risk-five”) es una arquitectura abierta y orientada a la educación basada en un juego de instrucciones (ISA) reducido (RISC)
- Es un conjunto de instrucciones **gratuito y abierto** que se puede usar para cualquier propósito sin pagar royalties, lo que permite que cualquiera diseñe, fabrique y venda chips y software de RISC-V
- El proyecto comenzó en 2010 en la **Universidad de California en Berkeley**



RISC-V pipeline

- **RISC-V** es un procesador de 5 etapas:



RISC-V pipeline

- **RISC-V** es un procesador de 5 etapas:
 1. IF : Instruction fetch
 2. ID : Instruction decode
 3. EX : Execute
 4. MEM : Memory Access
 5. WB : Register write back



RISC-V registers

Register	ABI	Use by convention
x0	zero	hardwired to 0, ignores writes
x1	ra	return address for jumps
x2	sp	stack pointer
x3	gp	global pointer
x4	tp	thread pointer
x5	t0	temporary register 0
x6	t1	temporary register 1
x7	t2	temporary register 2
x8	s0 or fp	saved register 0 or frame pointer
x9	s1	saved register 1
x10	a0	return value or function argument 0
x11	a1	return value or function argument 1
x12	a2	function argument 2
x13	a3	function argument 3
x14	a4	function argument 4
x15	a5	function argument 5
x16	a6	function argument 6

x17	a7	function argument 7
x18	s2	saved register 2
x19	s3	saved register 3
x20	s4	saved register 4
x21	s5	saved register 5
x22	s6	saved register 6
x23	s7	saved register 6
x24	s8	saved register 8
x25	s9	saved register 9
x26	s10	saved register 10
x27	s11	saved register 11
x28	t3	temporary register 3
x29	t4	temporary register 4
x30	t5	temporary register 5
x31	t6	temporary register 6
pc	(none)	program counter



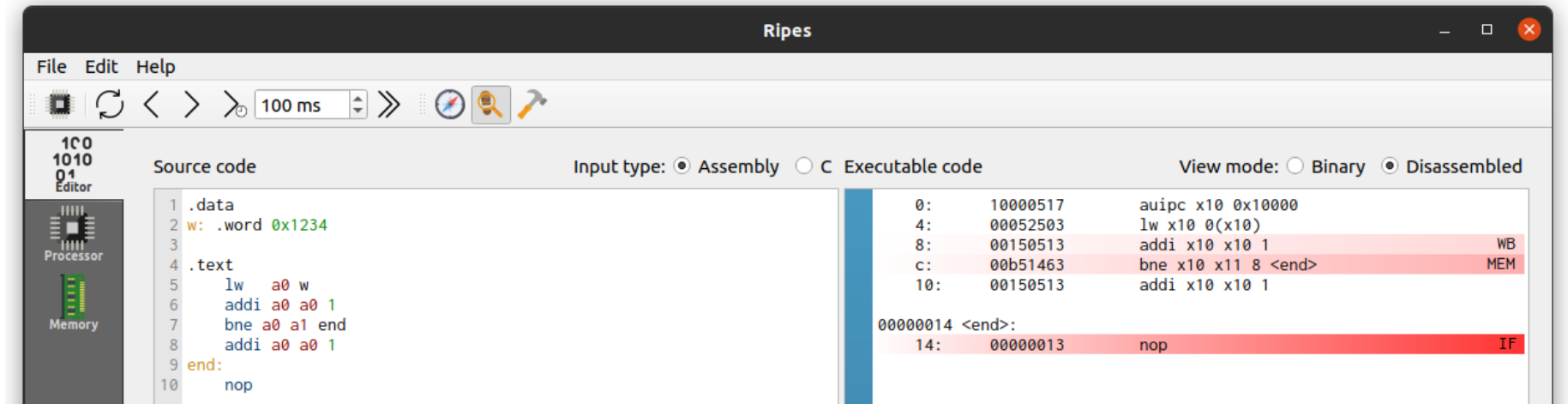
RIPES SIMULATOR

- **RIPES** es un simulador y editor de código ensamblador diseñado para simular el ISA del RISC-V en la **versión de 32 bits** (RISC-V RV32(I/M))
- Permite depurar, compilar y ejecutar **código ensamblador y código C**
- Tiene **3 pestañas** con distintas funcionalidades



RIPES SIMULATOR - Editor

- Permite escribir el código ensamblador y desensamblarlo en instrucciones básicas
- Marca errores de sintaxis



RIPES SIMULATOR

Processor

- Muestra información relacionada con la ejecución del procesador: el contenido de los registros y la memoria de instrucciones, métricas o la salida del programa

The screenshot displays the Ripes Simulator interface, which includes a central processor architecture diagram and several data panels. The processor diagram shows components like Instr. memory, Decode, Registers, and ALU. The Registers panel (1) lists registers x7 through x12 with their aliases and values in hexadecimal. The Instruction memory panel (2) shows a table of instructions with their addresses, stages, and contents. The Statistics panel (3) displays performance metrics such as Cycles, Instructions retired, CPI, and IPC. The Output panel (4) is currently empty.

Registers (1)

Name	Alias	Value
x7	t2	0x00000000
x8	s0	0x00000000
x9	s1	0x00000000
x10	a0	0x00000001
x11	a1	0x00000000
x12	a2	0x00000000

Display type: Hex

Instruction memory (2)

BP	PC	Stage	Instruction
<input type="checkbox"/>	0xc	MEM	lw x11 -4(x11)
<input type="checkbox"/>	0x10	EX	auipc x12 ...
<input type="checkbox"/>	0x14	ID	lw x12 -8(x12)
<input type="checkbox"/>	0x18	IF	auipc x13 ...
<input type="checkbox"/>	0x1c		lw x13 -12(x13)
<input type="checkbox"/>	0x20		ial x1 0x84

Statistics (3)

Cycles: 6
Instrs. retired: 2
CPI: 3
IPC: 0.333

Output (4)

RIPES SIMULATOR

Memory

- Muestra todo el espacio de memoria direccionable del procesador
- Tiene un simulador de memoria caché

The screenshot displays the RIPES simulator interface. On the left, a sidebar contains icons for '1C0 1010 Editor', 'Processor', and 'Memory'. The main window is divided into two panes. The left pane shows a memory dump with columns for Address, Word, and Bytes (Byte 0 to Byte 3). The right pane is titled 'Data cache' and 'Instruction cache'. It includes a 'Cache configuration' section with a 'Preset' dropdown, 'Lines' (5), 'Ways' (0), and 'Blocks' (2). The 'Repl. policy' is set to 'LRU', 'Wr. hit' to 'Write-back', and 'Wr. miss' to 'Write allocate'. The 'Statistics' section shows 'Size (bits): 4896', 'Hit rate: 0.7406', 'Writebacks: 0', 'Hits: 2038', and 'Misses: 714'. Below this is a 'Cache indexing breakdown' bar chart. The bottom section shows a table with columns for Index, V, D, Tag, Block 0, Block 1, Block 2, and Block 3. The table contains 24 rows of data, with some cells highlighted in yellow and green. At the bottom of the interface, there are controls for 'Display type: Hex', 'Go to register:', and 'Go to section:'.



RIPES SIMULATOR

Memory

- **.data** la utilizaremos para indicar el comienzo de la zona de declaración de datos estáticos
- **.text** la utilizaremos para indicar el comienzo de la zona de instrucciones

Directive	Arguments	Description
.text		emit .text section (if not present) and make current
.data		emit .data section (if not present) and make current
.string	"string"	emit string
.asciz	"string"	emit string (alias for .string)
.byte	expression [, expression]*	8-bit comma separated words
.2byte	expression [, expression]*	16-bit comma separated words
.half	expression [, expression]*	16-bit comma separated words
.short	expression [, expression]*	16-bit comma separated words
.4byte	expression [, expression]*	32-bit comma separated words
.word	expression [, expression]*	32-bit comma separated words
.long	expression [, expression]*	32-bit comma separated words
.zero	integer	emit \$integer zero-valued bytes



Entregable

Source code

Input type: ☒ Assembly ☐ C

```
1 .data
2
3     # Exercise Header
4     Practica:      .word 1      # practical exercise reference
5     Ejercicio:     .word 1      # Exercise number
6     Apartado:      .string "a"  # Exercise section
7
8     Alumno_1:      .string "nombre_apellidos" # Name alumn 1
9     Alumno_2:      .string "nombre_apellidos" # Name alumn 2
10    Alumno_3:      .string "nombre_apellidos" # Name alumn 3 (max)
11
12 .text
13
14 begin:
15
16     # Assembly Code
17
18 end:
19 li a7, 10
20 ecall
```

IF

