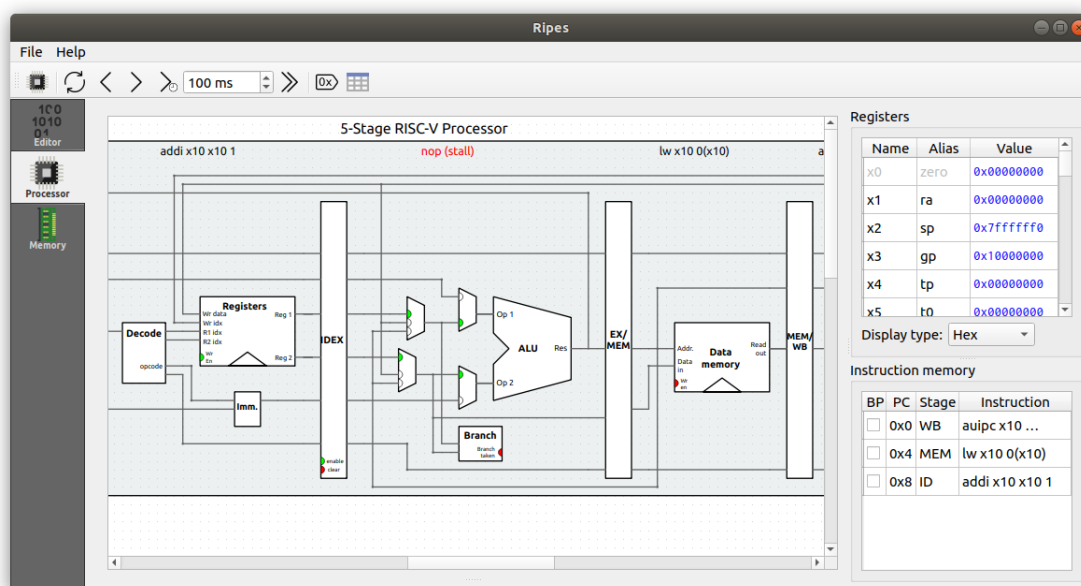


# ARQUITETURA DE COMPUTADORES

2021-2022

## Laboratório 1 - Introdução ao Simulador RIPES

Este laboratório destina-se a introduzir o simulador [Ripes](https://github.com/mortbopet/Ripes)<sup>1</sup>, o qual será usado como ferramenta didática para ensinar arquitetura de computadores. Para o efeito, recomenda-se que cada aluno proceda à instalação desta ferramenta no seu computador pessoal, tendo em consideração a informação disponível [aqui](#).



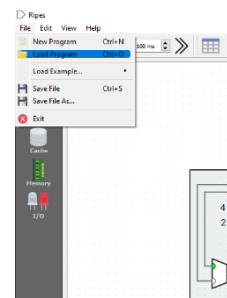
Para além da informação disponível na página da cadeira, recomenda-se a leitura da [informação disponível na página do simulador](#), de modo a garantir uma fácil ambientação ao mesmo.

## Guia de procedimentos

Execute os seguintes passos:

### 1. Carregar um pequeno programa

Selecione “File” seguindo de “Load Program” da barra de menu, selecione “Source file” e o ficheiro “intro.s” fornecido na página da cadeira.



<sup>1</sup> <https://github.com/mortbopet/Ripes>

## 2. Observar o código

Certifique-se que está na janela do editor clicando na aba “Editor” do lado esquerdo. Observe as diferentes zonas de programa (secções), as diretivas, as instruções, as etiquetas (*labels*) e os operandos, tal como descritos no ficheiro de acetatos fornecido junto com enunciado na página da cadeira.



### Secção .data

A secção `.data` permite declarar variáveis em memória. A declaração segue o seguinte formato:

**<nome da variável>:      <tipo>                      <argumento: conteúdo/dimensão>**

Em Assembly as variáveis definem-se com base na dimensão em bytes, e não com base no formato numérico. Por exemplo, como ambos os tipos `int` e `float` (linguagem C) são representados em 4B, em Assembly correspondem ao mesmo tipo. No caso do Ripes, são suportados os seguintes tipos:

- `.word` – declaração de uma ou mais variáveis com dimensão 4B (C: `int` ou `float`).
- `.half` - declaração de uma ou mais variáveis com dimensão 2B (C: `short`)
- `.byte` - declaração de uma ou mais variáveis com dimensão 1B (C: `byte` ou `char`)
- `.zero` – declaração de uma lista de valores com dimensão indicada pelo parâmetro seguinte. No caso do programa exemplo, são declarados 16B que podem corresponder a 4 words, 8 halfwords, 16 bytes, ou uma outra combinação das anteriores.
- `.string` – declaração de uma lista (array) de caracteres

De notar que no caso do tipo `.zero` o argumento corresponde ao número de bytes a declarar, enquanto nos outros casos corresponde a uma lista (separada por vírgulas) de valores. Por omissão os valores são declarados em decimal, mas podem ser feitos em hexadecimal ou em binário usando o prefixo “0x” ou “0b”, respetivamente, tal como em linguagem C.

### Secção .text

A secção `.text` permite declarar uma sequência de instruções em memória. A declaração de instruções segue o formato:

**<label>                      <instrução>                      <argumentos>**

Por exemplo a instrução,

**`addi x7, x6, 1`**

realiza a operação de *add with immediate* (`addi`), somando o valor no registo x6 e o imediato (constante) 1 e colocando o resultado no registo x7. O significado das instruções será abordado nas aulas teóricas e está descrito no *RISC-V Reference Card*, disponível na página da UC.

A *label* (etiqueta) é apenas um identificador que permite simplificar a escrita do código, particularmente útil na descrição de blocos condicionais de código (C: `if`), ciclos (C: `for` ou `while`), ou chamadas a funções, tal como abordado em aulas futuras.

As *labels*, instruções e argumentos devem ser sempre alinhados, sendo que a *label* deverá ficar sempre à esquerda sem que haja qualquer separador (espaço ou tab) entre o início da linha e a *label*. Os comentários devem ser precedidos pelo caracter ‘#’.

### 3. Observar o código máquina e o conteúdo da memória

Observe o código máquina do programa (a linguagem do processador) na coluna da direita da janela.

Clique na aba “Memória” (do lado esquerdo) e observe o conteúdo da memória.

O mapa de memória do processador inclui várias secções, nomeadamente:

- **.text** - instruções
- **.data** – variáveis
- **.bss** – stack (será abordado em aulas futuras)



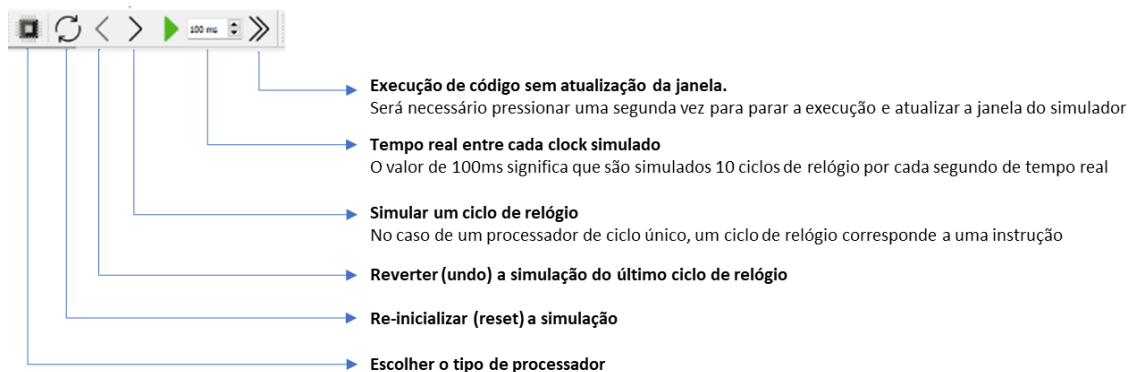
Mude a secção de memória para **.data** clicando na caixa “Go to section” em baixo.

- Em que endereços de memória estão as variáveis **var1**, **var2**, **var3**, **table1** e **str1**?

Mude a representação para decimal com e sem sinal, clicando na caixa “Display Type” em baixo e seleccionando “Signed” e “Unsigned”, respetivamente.

### 4. Correr um programa e ver resultados na consola

Para executar um programa utilize a barra de ferramentas no topo.



Clique novamente na aba “Editor”. Clique no botão “>” (*run*) em cima e observe o resultado na janela “Console” em baixo. Deverá surgir “Introducao ao RIPES”. Clique no botão “>” novamente para parar a execução, e em ↺ (*reset*) para voltar ao estado inicial.

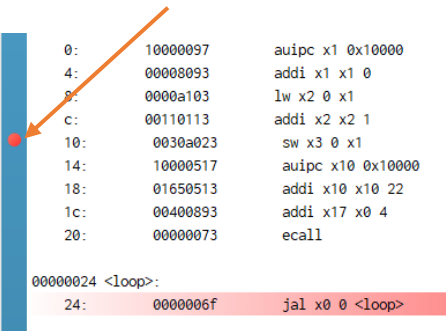
### 5. Correr passo a passo (para frente e para trás) em modo temporizado

Clique no botão “>” para executar a próxima instrução em “<” para voltar atrás no programa.

Clique em ▶ para correr o programa executando cada instrução com um intervalo de tempo especificado na caixa que se segue (100ms).

## 6. Pontos de paragem.

Na janela da coluna da direita, onde se encontra o código máquina, clique na barra azul ao lado da linha identificada com o endereço 10. Deverá aparecer um círculo vermelho. Este representa um ponto de paragem nessa linha de código. Quando correr o programa o simulador parará a execução sempre que chegar a este ponto.



The image shows a snippet of assembly code. A vertical blue bar on the left side of the code lines has a red dot on the line corresponding to address 10. An orange arrow points from the text above to this red dot.

0:	10000097	auipc x1 0x10000
4:	00008093	addi x1 x1 0
8:	0000a103	lw x2 0 x1
c:	00110113	addi x2 x2 1
10:	0030a023	sw x3 0 x1
14:	10000517	auipc x10 0x10000
18:	01650513	addi x10 x10 22
1c:	00400893	addi x17 x0 4
20:	00000073	ecall
00000024 <loop>:		
24:	0000006f	jal x0 0 <loop>

- Faça *reset* ao programa e volte a simular com a opção *run*. O programa deve parar no endereço 10, que corresponde à instrução na linha 14 do editor.

Remova o ponto de paragem clicando no círculo vermelho.

## 7. Observar o conteúdo dos registos

Os registos do processador são pequenas unidades de memória (registos) que são utilizados para guardar valores temporários. O RISC-V tem 32 registos, embora o registo x0 assuma sempre o valor zero.

Os registos estão representados na coluna da direita da janela do editor. Corre o programa passo a passo e observe que a instrução “addi x7, x6, 2” soma dois ao registo x6 e guarda o resultado em x7.

Name	Alias	Value
x0	zero	0x00000000
x1	ra	0x10000000
x2	sp	0x0000009a
x3	gp	0x10000000
x4	tp	0x00000000
x5	t0	0x00000000
x6	t1	0x00000000

## 8. Editar código

No editor de texto altere a instrução “addi x7, x6, 1” para “addi x7, x6, 2” de forma a somar 2 ao registo x6 (em vez de 1), colocando o resultado em x7. Corra o programa passo a passo e confirme que tal acontece na janela com os registos.

As instruções serão abordadas nas aulas teóricas e práticas e estão descritas no *RISC-V Reference Card*, disponível na página da UC. Não é necessário decorar as instruções, apenas perceber e consultar o *Reference Card*, sempre que necessário (incluindo em qualquer momento de avaliação).