

## AULA PRÁTICA N.º 1

### Objetivos:

- Conceitos básicos de Arquitetura de Computadores.
- Programação em linguagem *assembly*: estrutura de um programa e instruções básicas do MIPS.
- Apresentação das ferramentas a utilizar nas aulas práticas.

### Conceitos básicos:

- Os registos internos do MIPS. Caso particular do registo **\$0**.
- Linguagem *assembly* e código máquina.
- O simulador MARS para o MIPS<sup>1</sup>:
  - As janelas do simulador: Editor, Text Segment, Data Segment, Registers, Labels e Messages.
  - Configuração da ferramenta.
  - Edição e compilação de um programa.
  - Execução controlada de um programa: *run*, *single-step* e *breakpoints*.

### Guião:


1. Pretende-se escrever um programa, em linguagem *assembly*, que implemente a expressão aritmética  $y = 2x + 8$ . Supondo que o valor de  $x$  é passado através do registo **\$t0** (\$8) do CPU e que o resultado é depositado no registo **\$t1** (\$9), uma possível solução é:

```

.data
                                # nada a colocar aqui, de momento

.text
.globl  main
main:   ori  $t0,$0,val_x # $t0 = x (substituir val_x pelo
                                # valor de x pretendido)
        ori  $t2,$0,8     # $t2 = 8
        add  $t1,$t0,$t0  # $t1 = $t0 + $t0 = x + x = 2 * x
        add  $t1,$t1,$t2  # $t1 = $t1 + $t2 = y = 2 * x + 8
        jr   $ra          # fim do programa

```

- a) Edite o programa (com o editor do MARS) e substitua "**val\_x**" pelo valor de  $x$  com que pretende efetuar o cálculo (por exemplo, o valor 3).
- b) Compile o programa (opção Run → Assemble ou ). Se for assinalado algum erro de sintaxe na janela de mensagens, corrija o erro e repita a compilação.
- c) Execute o programa (opção Run → Go). Observe, e anote no seu *logbook*, o resultado presente no registo **\$t1**. Repita os procedimentos anteriores para outros valores de  $x$ .

<sup>1</sup> Link para *download* do simulador: <http://courses.missouristate.edu/KenVollmar/mars/download.htm>.

- d) Preencha a tabela seguinte com o endereço de memória e o código máquina de cada uma das instruções do programa que escreveu<sup>2</sup>.

Endereço de memória	Código máquina	Instrução
0x00400024	0x34080003	ori \$t0,\$0,3

- e) Coloque um *breakpoint* na primeira instrução do programa (`ori $t0,$0,...`). Faça o *reset* ao sistema (opção Run → Reset) e execute novamente o programa - a execução vai parar na instrução "`ori $t0,$0,...`". Execute a parte restante do programa passo a passo (opção Run → Step) e preencha a tabela com os valores que os vários registos vão tomando.

Para ativar um *breakpoint*, selecione o quadrado correspondente à instrução onde pretende que a execução do programa seja interrompida (coluna *Bkpt* na janela *execute* do MARS).

PC	Instrução	\$t0	\$t1	\$t2
0x00400024	ori \$t0,\$0,3	0x00000000	0x00000000	0x00000000
	ori \$t2,\$0,8	0x00000003	0x00000000	0x00000000
	add \$t1,\$t0,\$t0			
	add \$t1,\$t1,\$t2			
	jr \$ra			

2. Altere o programa que escreveu no ponto 1, de modo a implementar a expressão aritmética  $y = 2x - 8$ .

- a) Execute o programa para  $x=2$ , 3, 4 e 5 e observe os resultados no registo `$t1`. Interprete o resultado de  $y$  para  $x=3$  e  $x=5$ . Anote os resultados no seu *logbook*.
- b) Proceda do modo descrito na alínea e) do ponto anterior e preencha a tabela seguinte na situação em que  $x=3$ .

PC	Instrução	\$t0	\$t1	\$t2
0x00400024	ori \$t0,\$0,3			
	ori \$t2,\$0,8			
	add \$t1,\$t0,\$t0			
	?			
	jr \$ra			

3. Na solução adotada nos exercícios anteriores, a atribuição do valor de  $x$  faz parte da codificação do programa. A alteração do valor de  $x$  pressupõe a edição do código fonte e a geração de novo código máquina, ou seja,  $x$  é encarado pelo programa como uma constante.

Também a observação do resultado tem que ser efetuada diretamente no registo do CPU. Neste exercício vão ser utilizadas funções de interação com o utilizador (normalmente designadas por *system calls*) para permitir a leitura do valor de  $x$  a partir do teclado (durante a execução do programa) e a apresentação do correspondente valor de  $y$ .

O MARS disponibiliza cerca de 50 *system calls*, com diferentes funcionalidades (na tabela de instruções do MIPS, disponível no site da UC, pode encontrar uma tabela com a listagem das mais utilizadas - a lista completa pode ser observada no *help* do MARS). Uma *system call* é chamada através da colocação no registo `$v0` (\$2) do número que a identifica

<sup>2</sup> Se está a executar o MARS no seu computador pessoal, certifique-se que o ficheiro "exceptions.s" está associado ao *exception handler* do simulador: menu "Settings → Exception Handler..." (o ficheiro "exceptions.s" está disponível no moodle da UC na secção "Ferramentas de Software").

(ver tabela de instruções), seguida da instrução *syscall*. Por exemplo, para a leitura de um valor inteiro do teclado, pode ser usada a *system call* `read_int()` através da seguinte sequência de instruções:

```
ori $v0,$0,5    # a system call read_int() é
                 # identificada com o número 5 (ver
                 # tabela de instruções)
syscall         # a system call read_int() é chamada
```

Para a *system call* `read_int()` o valor lido do teclado é devolvido através do registo `$v0` do CPU.

Para visualizar o conteúdo de um registo do CPU no ecrã pode ser usada a *system call* `print_int10()`; nesse caso o valor que se pretende visualizar no ecrã é passado através do registo `$a0` (`$4`), pelo que, para além da inicialização do registo `$v0` com o identificador do `print_int10()`, é necessário copiar para o registo `$a0` o valor a imprimir. Por exemplo, mostrar no ecrã o valor do registo `$t5` (`$13`) pode ser feito através da seguinte sequência de instruções:

```
or  $a0,$0,$t5  # copia o registo $t5 para o registo $a0
ori $v0,$0,1    # a system call print_int10() é
                 # identificada com o número 1 (ver
                 # tabela de instruções)
syscall         # a system call print_int10() é chamada
```

- a) Faça as alterações ao programa que escreveu no **exercício 2**, de modo a ler do teclado o valor de `x` e a imprimir no ecrã o resultado do cálculo de `y`.

```
.data
.text
.globl main
main: ori $v0,$0,5    #
      syscall        # chamada ao syscall "read_int()"
      or  $t0,$0,??? # $t0 = $v0 = valor lido do teclado
                        # (valor de x pretendido)
      ori $t2,$0,8   # $t2 = 8
      add $t1,$t0,$t0 # $t1 = $t0 + $t0 = x + x = 2 * x
      sub $t1,$t1,$t2 # $t1 = $t1 + $t2 = y = 2 * x - 8
                        # ($t1 tem o valor calculado de y)
      or  $a0,$0,??? # $a0 = y
      ori $v0,$0,1   #
      syscall        # chamada ao syscall "print_int10()"
                        #
      jr  $ra        # fim do programa
```

- b) Execute o programa para diferentes valores de `x` e observe, em particular, o resultado para `x=3` e `x=5`. Anote os resultados no seu *logbook*.
- c) Altere novamente o programa de modo a usar a *system call* `print_int16()`. Execute o programa para diferentes valores de `x` e observe, em particular, o resultado para `x=2`, `3`, `4` e `5`. Anote os resultados no seu *logbook*.
- d) Utilize, finalmente, a *system call* `print_intu10()`. Execute o programa e observe o resultado para `x=2`, `3`, `4` e `5`. Anote os resultados no seu *logbook*.