Aula prática 2

- Lógica bitwise e operações com máscaras. Instruções lógicas.
- Deslocamento (shift) lógico e aritmético. Instruções de deslocamento.
- Diretivas do assembler do MARS.

Bernardo Cunha, José Luís Azevedo

Lógica bitwise e operações com máscaras.

Instrução Assembly		Operação em C
# entre registos		
and	$Rdest, $Rsrc1, $Rsrc2	# $Rdest = $Rsrc1 & Rsrc2
or	$Rdest, $Rsrc1, $Rsrc2	# $Rdest = $Rsrc1 | Rsrc2
nor	$Rdest, $Rsrc1, $Rsrc2	# $Rdest = ~($Rsrc1 | Rsrc2)
xor	$Rdest, $Rsrc1, $Rsrc2	# $Rdest = $Rsrc1 ^ Rsrc2
# entre 1 registo e uma constante		
andi	$Rdest, $Rsrc1, imm	# $Rdest = $Rsrc1 & imm
ori	$Rdest, $Rsrc1, imm	# $Rdest = $Rsrc1 | imm
xori	$Rdest, $Rsrc1, imm	# $Rdest = $Rsrc1 ^ imm

Lógica bitwise e operações com máscaras.

Exemplos

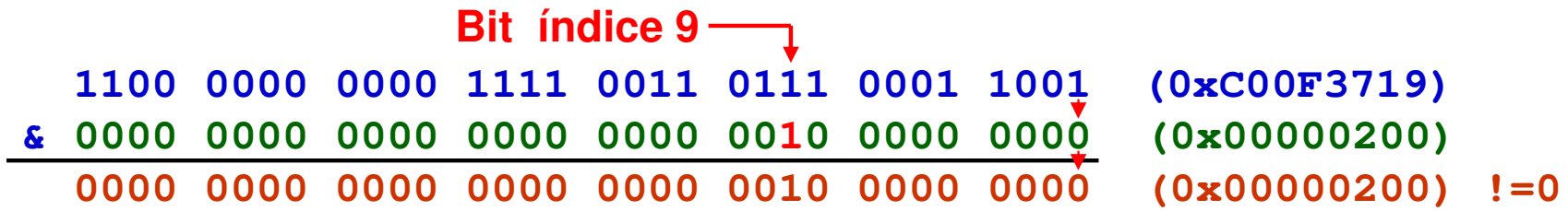
\# c/ $t1 = 0xC00F3719 e $t2=0x8FC345FF

```
and     $t0, $t1, $t2
        1100 0000 0000 1111 0011 0111 0001 1001   (0xC00F3719)
      & 1000 1111 1100 0011 0100 0101 1111 1111   (0x8FC345FF)
        ---------------------------------------
        1000 0000 0000 0011 0000 0101 0001 1001   (0x80030519)

or      $t0, $t1, $t2
        1100 0000 0000 1111 0011 0111 0001 1001   (0xC00F3719)
      | 1000 1111 1100 0011 0100 0101 1111 1111   (0x8FC345FF)
        ---------------------------------------
        1100 1111 1100 1111 0111 0111 1111 1111   (0xCFCF77FF)

xor     $t0, $t1, $t2
        1100 0000 0000 1111 0011 0111 0001 1001   (0xC00F3719)
      ^ 1000 1111 1100 0011 0100 0101 1111 1111   (0x8FC345FF)
        ---------------------------------------
        0100 1111 1100 1100 0111 0010 1110 0110   (0x4FCC72E6)
```

Operações com máscaras.

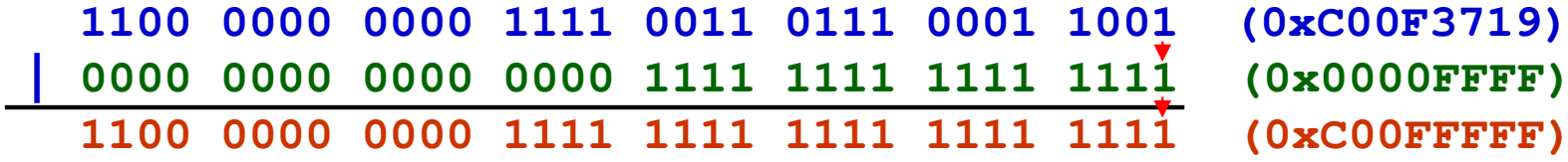
Exemplo: verificar se o bit de índice 9 é zero ou != 0

```
                                    # c/ $t1 = 0xC00F3719
andi    $t0, $t1, 0x00000200        # 0x00000200 máscara a aplicar
```



Exemplo: forçar a 1 os 16 bits menos significativos

```
ori     $t0, $t1, 0x0000FFFF        # 0x0000FFFF máscara a aplicar
```



Operações com máscaras.

Exemplo: obter o complemento para 1 de um registo

```
                                        # c/ $t1 = 0xAAAAAAAA

xori    $t0, $t1, 0xFFFFFFFF    # 0xFFFFFFFF máscara a aplicar
```

Esta instrução é virtual e não existe no MARS

```
  1010 1010 1010 1010 1010 1010 1010 1010   (0xAAAAAAAA)
^ 1111 1111 1111 1111 1111 1111 1111 1111   (0xFFFFFFFF)
  ---------------------------------------
  0101 0101 0101 0101 0101 0101 0101 0101   (0x55555555)
```

Deslocamento (shift) lógico e aritmético

Instrução Assembly		**Operação em C**
sll	$Rdest, $Rsrc1, imm	# $Rdest = $Rsrc1 **<<** imm
srl	$Rdest, $Rsrc1, imm	# $Rdest = $Rsrc1 **>>** imm
sra	$Rdest, $Rsrc1, imm	# $Rdest = $Rsrc1 \ $\mathbf{2^{imm}}$

Significado das mnemónicas

sll	Shif Left Logical
srl	Shif Right Logical
sra	Shift Right Arithmetic

Deslocamento (shift) lógico e aritmético

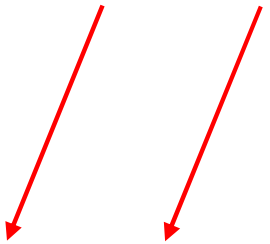
Operações realizadas pelas instruções

Exemplo:

```
sll     $t0, $t1, 3                 # c/ $t1 = 0x04002319
# $t1 = 0000 0100 0000 0000 0010 0011 0001 1001
```

Após a execução:

```
$t0=  0010 0000 0000 0001 0001 1000 1100 1000
```



Deslocamento (shift) lógico e aritmético

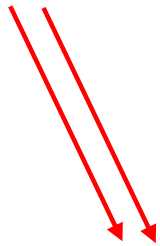
Operações realizadas pelas instruções

Exemplo:

```
srl     $t0, $t1, 3                 # c/ $t1 = 0x84002319
# $t1 = 1000 0100 0000 0000 0010 0011 0001 1001
```

Após a execução:

```
$t0=  0001 0000 1000 0000 0000 0100 0110 0011
```



Deslocamento (shift) lógico e aritmético

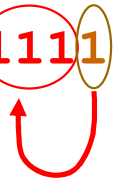
Operações realizadas pelas instruções

Exemplo:

```
sra     $t0, $t1, 3                 # c/ $t1 = 0x84002319
# $t1 = 1000 0100 0000 0000 0010 0011 0001 1001
```

Após a execução:

```
$t0=   1111 0000 1000 0000 0000 0100 0110 0011
```



Extensão do bit de sinal

Algumas diretivas para o Assembler

```
#Exemplo:
            .data
str1:       .asciiz "Introduza 2 numeros\n"
            .eqv print_string,4
            .text
            .globl main
main:       …
```

Diretivas são ordens dadas ao Assembler durante o processo de conversão do programa assembly para código máquina.
No MIPS, são nomes reservados começados por um '.'
não correspondem a instruções

Algumas diretivas para o Assembler (significado)

.data

Informa o Assembler que as linhas subsequentes contêm informação relativa ao segmento de dados

str1: .asciiz "Introduza 2 numeros\n"

Diz ao Assembler que, a partir do endereço correspondentes a "str1:" deverá inicializar o número de bytes necessários com o código ASCII da string que está entre aspas. Essa string deverá terminar com o carácter null ('\0').

.eqv print_string,4

Diz ao Assembler que, sempre que encontrar a expressão "*print_string*", deverá substituí-la pela constante 4.

.text

Informa o Assembler que as linhas subsequentes contêm informação relativa ao segmento de texto (instruções)

.globl main

Informa o Assembler que "main:" é um *label* global (visível a partir de outro código)