AC1 2013-2014

universidade de aveiro theoria poiesis praxis
Departamento de Electrónica, Telecomunicações e Informática

Arquitectura de Computadores I

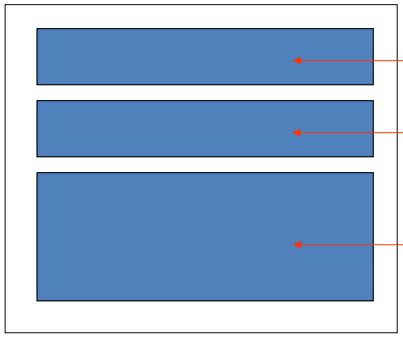
Pedro Miguel Cabral

Aula 02

1

AC1 2013-2014

MARS
Programa em Assembly



2

AC1 2013-2014

MARS
Programa em Assembly

Data

- Local do programa onde são colocadas as contantes:
  - Mensagens de Erro
  - Strings para introdução de dados
  - Constantes
  - Buffers de input/output
- Os dados são declarados com a directiva “.data”

Program Definition

- Local onde se “declara” o nosso programa assembly
- Uso da directiva “.text”
- Label inicial do nosso programa assembly:
  “.globl main”

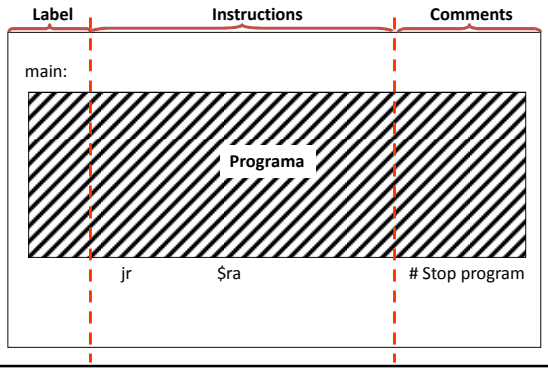
3

AC1 2013-2014

MARS
Programa em Assembly

Program Body

- Local onde se escreve programa assembly
- Tem de se começar pela label anteriormente declarada
  “main:”
- O programa deve acabar com a instrução “jr $ra”



4

AC1 2013-2014

Input/Output no MARS

System Calls

- Exemplo: print_str();

```
                .data
str:            .asciiz “Texto a imprimir!”
                .text
                .globl main

main:           li      $v0, 4
                la      $a0, str
                syscall
                jr      $ra
```

5

AC1 2013-2014

MARS

SYSCALL functions available in MARS

Introduction

A number of system services, mainly for input and output, are available for use by your MIPS program. They are described in the table below.

How to use SYSCALL system services

Step 1. Load the service number in register $v0.
Step 2. Load argument values, if any, in $a0, $a1, $a2, or $f12 as specified.
Step 3. Issue the SYSCALL instruction.
Step 4. Retrieve return values, if any, from result registers as specified.

Example: display the value stored in $t0 on the console

```
li  $v0, 1           # service 1 is print integer
add $a0, $t0, $zero  # load desired value into argument register $a0, using pseudo-op
syscall
```

6

MARS

AC1 2013-2014

Table of Available Services

Service	Code in $v0	Arguments	Result
print integer	1	$a0 = integer to print	
print float	2	$f12 = float to print	
print double	3	$f12 = double to print	
print string	4	$a0 = address of null-terminated string to print	
read integer	5		$v0 contains integer read
read float	6		$f0 contains float read
read double	7		$f0 contains double read
read string	8	$a0 = address of input buffer $a1 = maximum number of characters to read	*See note below table*
sbrk (allocate heap memory)	9	$a0 = number of bytes to allocate	$v0 contains address of allocated memory
exit (terminate execution)	10		

⋮

print integer in hexadecimal	34	$a0 = integer to print	
print integer in binary	35	$a0 = integer to print	

7

MARS

AC1 2013-2014

Directivas

```
.align      Align next data item on specified byte boundary (0=byte, 1=half, 2=word, 3=double)
.ascii      Store the string in the Data segment but do not add null terminator
.asciiz     Store the string in the Data segment and add null terminator
.byte       Store the listed value(s) as 8 bit bytes
.data       Subsequent items stored in Data segment at next available address
.double     Store the listed value(s) as double precision floating point
.extern     Declare the listed label and byte length to be a global data field
.float      Store the listed value(s) as single precision floating point
.globl      Declare the listed label(s) as global to enable referencing from other files
.half       Store the listed value(s) as 16 bit halfwords on halfword boundary
.kdata      Subsequent items stored in Kernel Data segment at next available address
.ktext      Subsequent items (instructions) stored in Kernel Text segment at next available address
.set        Set assembler variables.  Currently ignored but included for SPIM compatability
.space      Reserve the next specified number of bytes in Data segment
.text       Subsequent items (instructions) stored in Text segment at next available address
.word       Store the listed value(s) as 32 bit words on word boundary
```

8

AC1 2013-2014

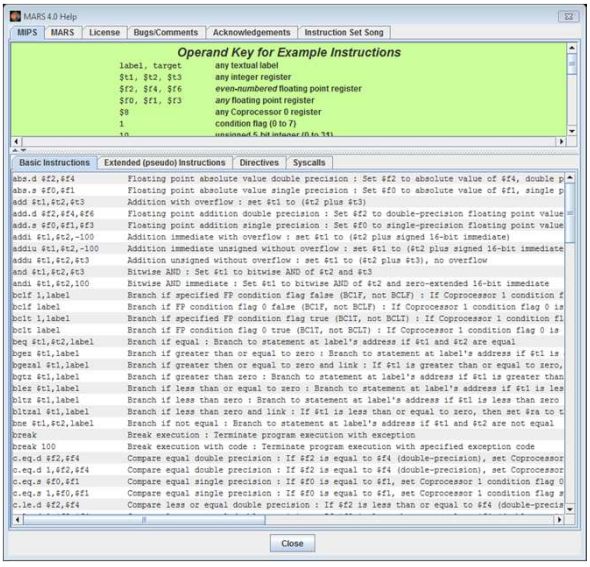
MARS
Instruções

abs.d $f2,$f4	Floating point absolute value double precision
abs.s $f2,$f4	Floating point absolute value single precision
add $1,$2,$3	Addition with overflow
add.d $f2,$f4,$f6	Floating point addition double precision
add.s $f2,$f4,$f6	Floating point addition single precision
addi $1,$2,-100	Addition immediate with overflow
addiu $1,$2,-100	Addition immediate unsigned without overflow
addu $1,$2,$3	Addition unsigned without overflow
and $1,$2,$3	Bitwise AND
andi $1,$2,100	Bitwise AND immediate
bc1f 1,label	Branch if specified FP condition flag false
bc1f label	Branch if FP condition flag 0 false
bc1t 1,label	Branch if specified FP condition flag true
bc1t label	Branch if FP condition flag 0 true
beq $1,$2,label	Branch if equal
bgez $1,label	Branch if greater than or equal to zero
bgezal $1,label	Branch if greater then or equal to zero and link
bgtz $1,label	Branch if greater than zero
blez $1,label	Branch if less than or equal to zero
bltz $1,label	Branch if less than zero
bltzal $1,label	Branch if less than zero and link
bne $1,$2,label	Branch if not equal
break	Break execution
break 100	Break execution with code
c.eq.d $f2,$f4	Compare equal double precision, result in condition flag 0
c.eq.d 1,$f2,$f4	Compare equal double precision, result in specified condition flag

9

AC1 2013-2014

MARS



10