Imprimindo Strings e Exercícios

Paulo Ricardo Lisboa de Almeida

UDESC
UNIVERSIDADE
DO ESTADO DE
SANTA CATARINA

Diretivas do montador

- Diretivas de montador começam com um . (ponto)
- Não geram instruções de máquina reais
  - Servem para dizer ao montador o que fazer
- Diretivas ficam no mesmo nível das instruções quanto a indentação
  - Uma tabulação de espaço contando a partir da margem esquerda
- Exemplo de diretiva

  .globl main
  - Informa que o rótulo main é visível globalmente
  - Qualquer um que incluir o arquivo assembly deve ser capaz de saber o endereço de main

Imprimindo Strings

- Seria muito trabalhoso para nós (e custoso para a CPU) imprimir os caracteres instrução a instrução
- O montador pode nos ajudar
- Escrevemos a string normalmente em uma seção de dados (.data)
- O montador traduz cada caractere para seu código específico, e coloca tudo na região de memória reservada para constantes do programa
- A string é terminada com o caractere '\0' pelo próprio montador
  - Para tal, utilize a diretiva asciiz

Exemplo

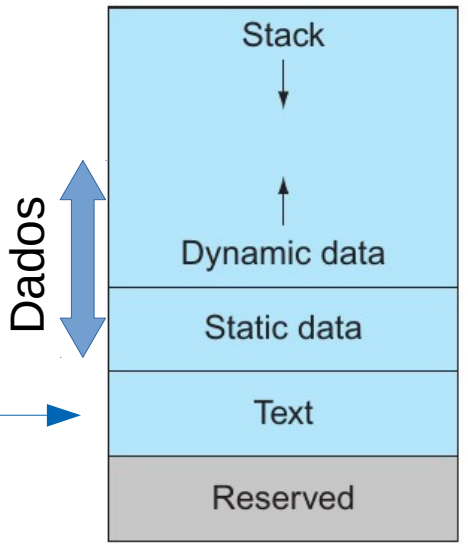
Diretiva .text indica que as instruções a seguir deve ir para a seção de instruções (text) do programa na memória

```
        .text
        .globl main
main:
        la $a0, ola_mundo
        li $v0, 4          #syscall para imprimir string
        syscall
end:
        li $v0, 10
        syscall
        .data
ola_mundo:
        .asciiz "Ola Mundo\n"
```

Diretiva .data indica que as linhas abaixo devem ser montadas na seção de dados da memória

Rótulo dado a String

diretiva .asciiz instrui o montador a colocar a string em formato ASCII sequencialmente na memória, e terminar com NULL (‘\0)



Exemplo

```
        .text
        .globl main
main:
        la $a0, ola_mundo
        li $v0, 4          #syscall para imprimir string
        syscall
end:
        li $v0, 10
        syscall
        .data
ola_mundo:
        .asciiz "Ola Mundo\n"
```

“la” (load adress) é uma pseudo-instrução, que instrui o montador a colocar o endereço onde “ola_mundo” foi montado. É traduzida para um lui e um ori

O sistema operacional vai exibir todos os caracteres, inciando na posição indicada por $a0, e parando quando encontrar um ‘\0’ (código 0 em ASCII)

Exercício

1. Monte o programa do exemplo no MARS. Rode e veja o resultado. Analise a seção de dados montada com o programa, e entenda como a string fica na memória, e o endereço que ***la*** está efetivamente carregando. Dica: visualize os valores em hexa para simplificar.

```
        .text
        .globl main
main:
        la $a0, ola_mundo
        li $v0, 4           #syscall para imprimir string
        syscall
end:
        li $v0, 10
        syscall
        .data
ola_mundo:
        .asciiz "Ola Mundo\n"
```

Exercícios

1. Faça um programa que solicita repetidos valores inteiros ao usuário, e imprime se o valor é par ou ímpar (pesquise sobre como funciona a instrução div no MIPS). O programa termina quando o usuário digita 0.
2. Note que em binário, utilizando complemento a dois, um número que termina com 1 é ímpar, e um número que termina com 0 é par. Com essa informação, faça o programa do exercício 1 novamente, mas agora sem a necessidade de divisões (+ rápido!).
3. Faça um programa que leia a idade do usuário em dias, e a exiba e anos, meses e dias no formato anos/meses/dias.
4. Escreva um programa que exiba as tabuadas do 2 até a do 10.
5. Escreva um programa para ler as coordenadas (x,y) de um ponto no plano cartesiano e escreve o quadrante ao qual o ponto pertence. Caso o ponto não pertença a nenhum quadrante, escrever se ele está sobre o eixo X, eixo Y, ou na origem (pode escrever “eixo x eixo y” nesse caso se facilitar sua vida).

Referências

- D. Patterson; J. Henessy. **Organização e Projeto de Computadores**: **A Interface Hardware / Software.** 4a Edição. Elsevier Brasil, 2014.
- STALLINGS, William. **Arquitetura e organização de computadores.** 8. ed. São Paulo: Pearson Education do Brasil, 2010.
- Bob Plantz. **Introduction to Computer Organization: A Guide to X86-64 Assembly Language and GNU/Linux.** 2019.