Simuladores MARS/Spim, E/S no MIPS e Imprimindo Strings

Yuri Kaszubowski Lopes

UDESC

Anotações

Simuladores MARS/SPIM

- Gratuitos e de código aberto (Licença MIT (MARS) e BSD (SPIM)
- Simuladores MIPS

MARS

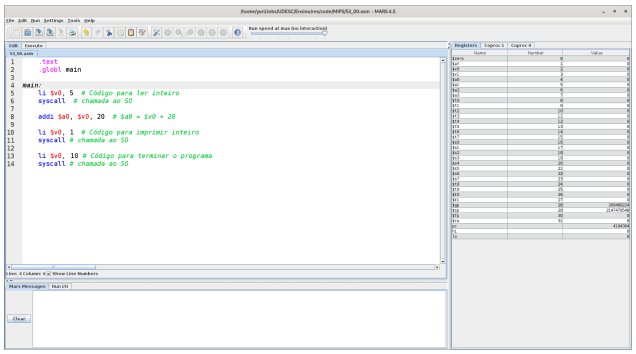
- MARS (MIPS Assembler and Runtime Simulator)
- Desenvolvido por Pete Sanderson e Kenneth Vollmar
- Download: courses.missouristate.edu/kenvollmar/mars/license.htm
- O programa é um jar: `java -jar Mars4_5.jar`
  - Necessário ter o Java instalado

SPIM (QtSpim)

- SPIM (Inverso de “MIPS”)
- Desenvolvido por James Larus
- Download: http://spimsimulator.sourceforge.net/
- Não é um editor/IDE
- Deve iniciar no rótulo `main`
- Deve enviar o código 10 para `syscall` ou retornar com `jr $ra`

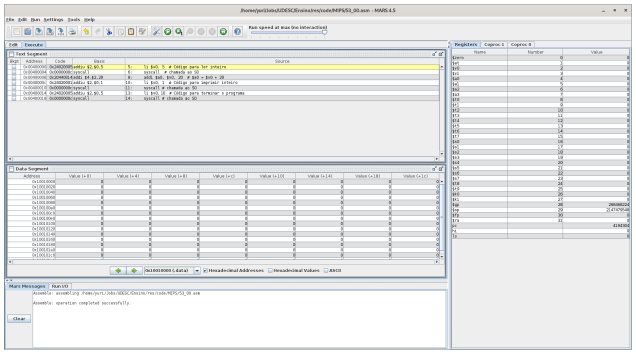
Anotações

Simulador MARS



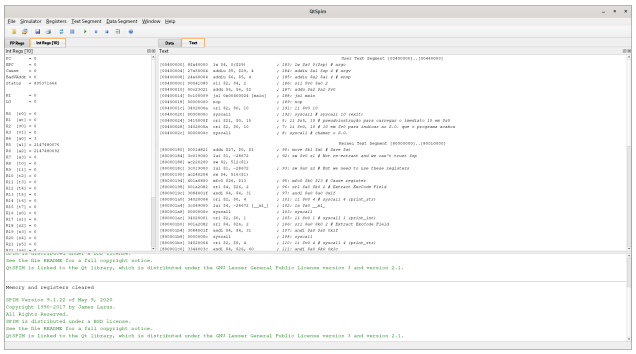
Anotações

Simulador MARS



Anotações

Simulador SPIM (QtSpim)



Anotações

Programas em Assembly

- Salvos com a extensão `.s`, `.as` ou `.asm`
  - Use `.s` no plugin do SPIM no Moodle
- Colocamos apenas um nível de indentação
  - Sem indentação: Definições de rótulos e importações
  - Um nível de indentação: Todos os comandos
  - Use `#` para inserir comentários
- Comente bem seus programas

Anotações

Exemplo de programa

- Um programa que calcula 9! e armazena o resultado em `$s0`

```
1     .text
2     .globl main
3 main:
4     li $s1, 9  # atribuição/carrega: $s1 = 9
5     li $s0, 1  # $s0 = 1
6 while:
7     mul $s0, $s0, $s1  # $s0 = $s0 * $s1 (32 bits baixos)
8     subi $s1, $s1, 1  # $s1 = $s1 - 1
9     bnez $s1, while  # se $s1 != 0 então vá p/ while
10 end:
11    li $v0, 10  # Código para encerrar o programa
12    syscall  # encerra o programa
```

Anotações

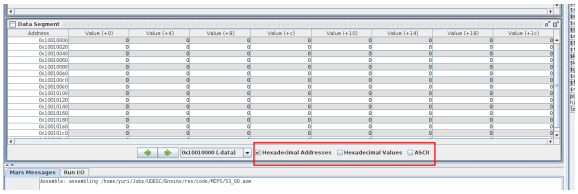
Usando o MARS

- Monte o seu programa
  - Note que o `li` é uma pseudoinstrução, e foi traduzida para instruções reais
- Execute seu programa
  - Todo o programa
  - Passo a passo

Anotações

Usando o MARS: Dicas

- Verifique o formato em que os valores estão sendo exibidos nos registradores
- Você pode zerar os registradores na interface
  - Necessário para executar o programa novamente (para zerar o PC)
- O MARS segue uma implementação específica do MIPS, então alguns detalhes podem ser diferentes dos apresentados em Patterson e Henessy (2014).
- Há diferenças entre MARS e SPIM



Anotações

Entrada e Saída

- A entrada e saída é feita com ajuda do Sistema Operacional
- O MARS e SPIM incluem um Sistema Operacional minimalista para simulações
- Usamos `syscall`
  - Colocamos o código da operação desejada em `$v0`
  - Instrução `syscall` devolve o controle ao S.O., que olha para `$v0` e faz o requisitado
  - Veja alguns códigos no Moodle da disciplina

Anotações

Exercícios

1. Modifique o exemplo do Slide 7 para que seja calculado `n!`, onde `n` é lido da entrada padrão e o resultado impresso em uma linha na saída padrão. Além disso, utilize a instrução `mult` no lugar de `mul`. Como `mult` funciona? Por que o resultado é armazenado em dois registradores? Nesse caso, pode imprimir apenas a parte baixa do resultado na tela para facilitar.
2. Faça um programa que leia continuamente valores inteiros da entrada padrão. O programa termina quando ler -1. Ao final o programa deve imprimir uma linha na saída padrão com a soma e outra linha com a média dos valores digitados.
3. Modifique o programa do exercício anterior, de forma que o programa termina quando o usuário digita -1, ou quando a soma atingir um valor maior ou igual a 2048.
4. Faça um programa que calcula o enésimo número da sequência de Fibonacci e exibe o resultado na saída padrão. O índice do número de Fibonacci deve ser lido da entrada padrão.

Anotações

Exercícios

5. Crie um programa para um caixa eletrônico que calcula o menor número possível de cédulas que deve ser entregue a um usuário quando ele fizer um saque. Considere que a entrada do programa é o valor do saque, e a saída são as notas que o usuário receberá. Exiba as quantidades de notas como inteiros simples na tela, na seguinte ordem: notas de 50, 20, 10 e 5 reais, e moedas de 1 real. Utilize a entrada e saída padrão. Exemplo se o usuário solicitar um saque de 87 reais:

   ```
   1 1 1 1 2
   ```

Dica: para imprimir um espaço:

```
1 li $a0, 32  # 32 é o código do espaço
2 li $v0, 11  # 11 em $v0 para S.O escrever $a0 na tela como char
3 syscall  # chama o S.O. para escrever
```

Anotações

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
  - Um nível de indentação
- Exemplo `.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

Anotações

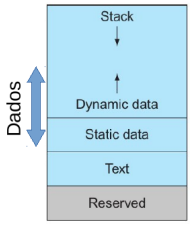
Imprimindo Strings

- Seria muito trabalhoso para nós (e custoso para a CPU) imprimir os caracteres um a um, i.e., 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
  - Use `asciiz`

Anotações

Imprimindo Strings

```
1     .data
2 texto:
3     .asciiz "Ola Mundo xyz\n"
4
5     .text
6     .globl main
7 main:
8     la $a0, texto
9     li $v0, 4  # syscall para imprimir string
10    syscall
11 end:
12    li $v0, 10
13    syscall
```



Anotações

Exercícios

6. Faça um programa que solicita repetidos valores inteiros ao usuário (pela entrada padrão), e imprime na saída padrão se o valor é par ou ímpar (pesquise sobre como funciona a instrução div no MIPS). O programa termina quando o usuário digita 0.
7. Faça um programa que leia da entrada padrão a idade do usuário em dias, e a exiba em anos, meses e dias no formato anos/meses/dias na saída padrão.
8. Escreva um programa que exibe as tabuadas do 2 até a do 10 na saída padrão.
9. Escreva um programa para ler da entrada padrão as coordenadas (x,y) de um ponto no plano cartesiano e escreve na saída padrão 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

Anotações

Referências

- D. Patterson; J. Henessy. **Organização e Projeto de Computadores: Interface Hardware/Software**. 5a Edição. Elsevier Brasil, 2017.
- Andrew S. Tanenbaum. **Organização estruturada de computadores**. 5. ed. São Paulo: Pearson, 2007.
- Harris, D. and Harris, S. **Digital Design and Computer Architecture**. 2a ed. 2012.
- courses.missouristate.edu/KenVollmar/mars/

Anotações

Simuladores MARS/Spim, E/S no MIPS e Imprimindo Strings

Yuri Kaszubowski Lopes

UDESC

Anotações