“Os humanos são pensadores lentos, desleixados e brilhantes. Computadores são rápidos, cuidadosos e estúpidos.” (John Pfeiffer)

Programas em Assembly, I/O e exercícios

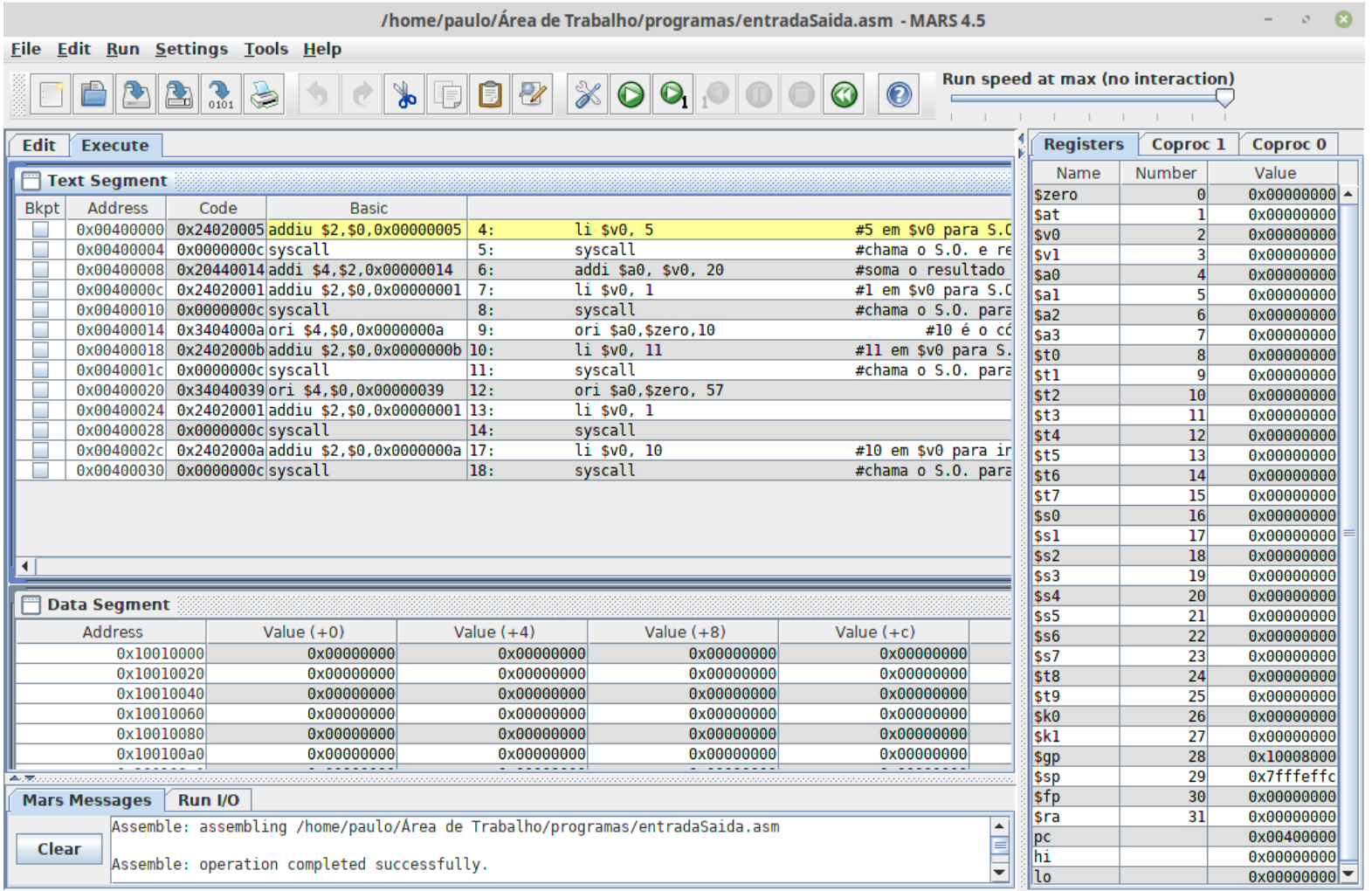
Paulo Ricardo Lisboa de Almeida

UDESC
UNIVERSIDADE
DO ESTADO DE
SANTA CATARINA

Simulador MARS

- MARS (MIPS Assembler and Runtime Simulator)
  - Simulador MIPS
  - Desenvolvido por Pete Sanderson e Kenneth Vollmar
- Gratuito e de código aberto
  - Licença MIT
  - Download: https://courses.missouristate.edu/KenVollmar/MARS/download.htm
- O programa é um jar, então basta ter o Java instalado na sua máquina
- Maneira simples de executar em casa:
  - Abra um terminal no diretório onde o jar foi salvo
  - Digite (assumindo que o nome do jar baixado é Mars4_5.jar)
  - java -jar Mars4_5.jar
  - **Nas máquinas do Lab já existe um atalho pronto no “Menu Iniciar”**

Simulador MARS



Simulador MARS

- Abra o MARS
- Crie um novo arquivo
- Salve como **aula.asm**

Programas em Assembly

- Programas escritos em assembly geralmente são salvos com a extensão **.s**, **.as** ou **.asm**
- Colocamos apenas um nível de indentação
  - Mais a esquerda
    - **Definições de rótulos e importações**
  - A uma tabulação de distância
    - **Todos os comandos**
- **Sempre comente o máximo possível em seus programas**

Criando seu programa

- Inicialmente, nossos programas terão o seguinte formato:

```
        .text
        .globl main
main:
        #seu programa aqui
end:
        li $v0, 10
        syscall
```

Indica o início das instruções do programa

Indica que o rótulo main é visível globalmente

Indica o ponto de início do programa

As instruções abaixo do end informam ao S.O. que o programa terminou através de uma syscall.

Exemplo

- Escreva o seguinte no programa aula.asm

```
        .text
        .globl main
main:
        li $s5, 15      #pseudoinstrução para carregar o imediato 15 em $s5
end:
        li $v0, 10      #10 em $v0 para indicar ao S.O. que o programa acabou
        syscall         #chamar o S.O.
```

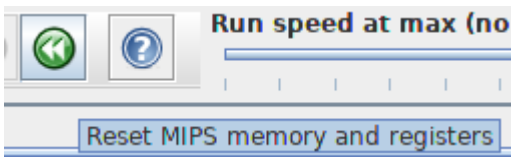
Assemble

- Para montar o código e torná-lo executável, clique no botão “Assemble”

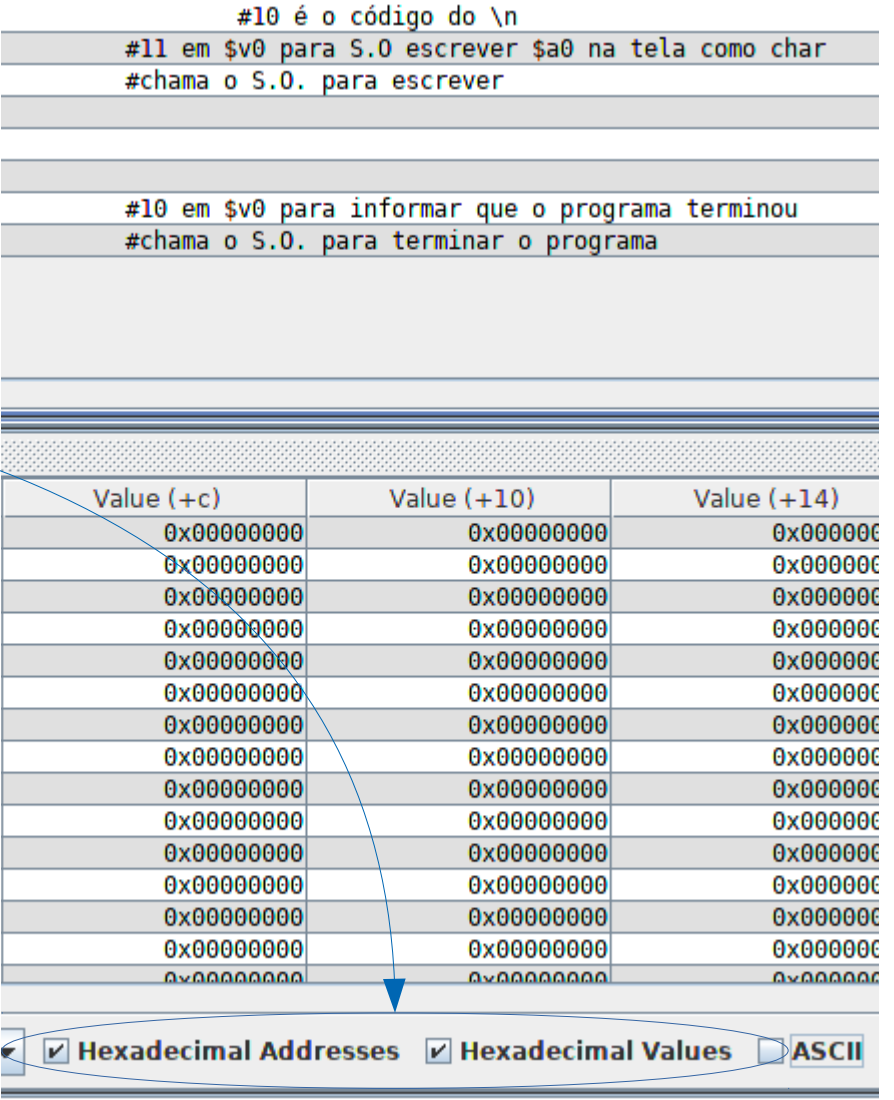
- Aperte o “play” e veja o resultado do seu programa nos registradores
- **Note que o li é uma pseudoinstrução, e foi traduzida para instruções reais**

Dicas

- Sempre verifique o formato em que os valores estão sendo exibidos nos registradores
  - Binário ou hexa
- Você pode zerar os registradores na interface
  - Necessário se deseja 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 (2019).



Exercícios

1. Faça um programa que calcule 9! e armazene o resultado em $s0
   Veja como funciona a instrução mul. Exemplo:
   mul $s1,$s2,$s3

Entrada e Saída

- A entrada e saída é feita com ajuda do Sistema Operacional
  - O MARS inclui um Sistema Operacional minimalista para simulações
- Precisamos fazer **syscalls**
  - Colocamos o código da operação desejada em **$v0**
  - Instrução syscall devolve o controle ao S.O., que olha para $v0 e faz o requisitado
  - Entradas e saídas são feitas na tela de RUN I/O
- Veja algumas chamadas de sistema válidas em
  - courses.missouristate.edu/kenvollmar/mars/help/syscallhelp.html

Exercícios

2. Modifique o exercício 1 para que seja calculado n!, onde n é solicitado do usuário. Exiba o resultado no console. Além disso, utilize a instrução mult no lugar de mul. Como mult funciona? Por que o resultado é armazenado em dois registradores?

3. Faça um programa que peça continuamente valores inteiros ao usuário. O programa termina quando o usuário digita -1. Ao final o programa deve exibir a soma e a média dos valores digitados.

4. 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.

5. Faça um programa que calcule o vigésimo número da sequência de Fibonacci e o armazene o registrador $s1.

6. Modifique o exercício anterior para que o índice do número de fibonacci a ser impresso seja requisitado do usuário, e o resultado seja impresso no console.

Exercícios

7. 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. Exemplo se o usuário solicitar um saque de 87 reais:

   1 1 1 1 2

Dica. Para imprimir um espaço :

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

UDESC

Referências

- D. Patterson; J. Henessy. **Organização e Projeto de Computadores: Interface Hardware/Software.** 5a Edição. Elsevier Brasil, 2019.
- 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/