

# **Laboratório 1**

## **- Assembly MIPS –**

**Marcelo Giordano Martins Costa de Oliveira, 12/0037301**

<sup>1</sup>Dep. Ciência da Computação – Universidade de Brasília (UnB)  
CiC 116394 - OAC - Turma A

***Abstract.** This report corresponds to the Experiment 1 about Assembly MIPS.*

***Resumo.** Este relatório corresponde ao Experimento 1 sobre Assembly MIPS.*

### **Introdução**

#### **Objetivos**

- Familiarizar o aluno com o Simulador/Montador MARS;
- Desenvolver a capacidade de codificação de algoritmos em linguagem Assembly MIPS;
- Desenvolver a capacidade de análise de desempenho de algoritmos em Assembly;

#### **Ferramentas**

- MARS v.4.5 Custom 7
- Cross compiler MIPS GCC
- Inkscape e GIMP

### **Procedimentos**

#### **Simulador/Montador MARS**

Essa parte do relatório foi realizada com o intuito de familiarizar os alunos ao Simulador/Montador MARS.

No item 1.2 do relatório, foram pedidos os gráficos relacionados aos valores do vetor fornecido e ao tempo de execução da subrotina Sort fornecida pelo professor. Temos então o grafico em Melhor caso 1 e Pior Caso 2 da rotina de ordenação Sort.

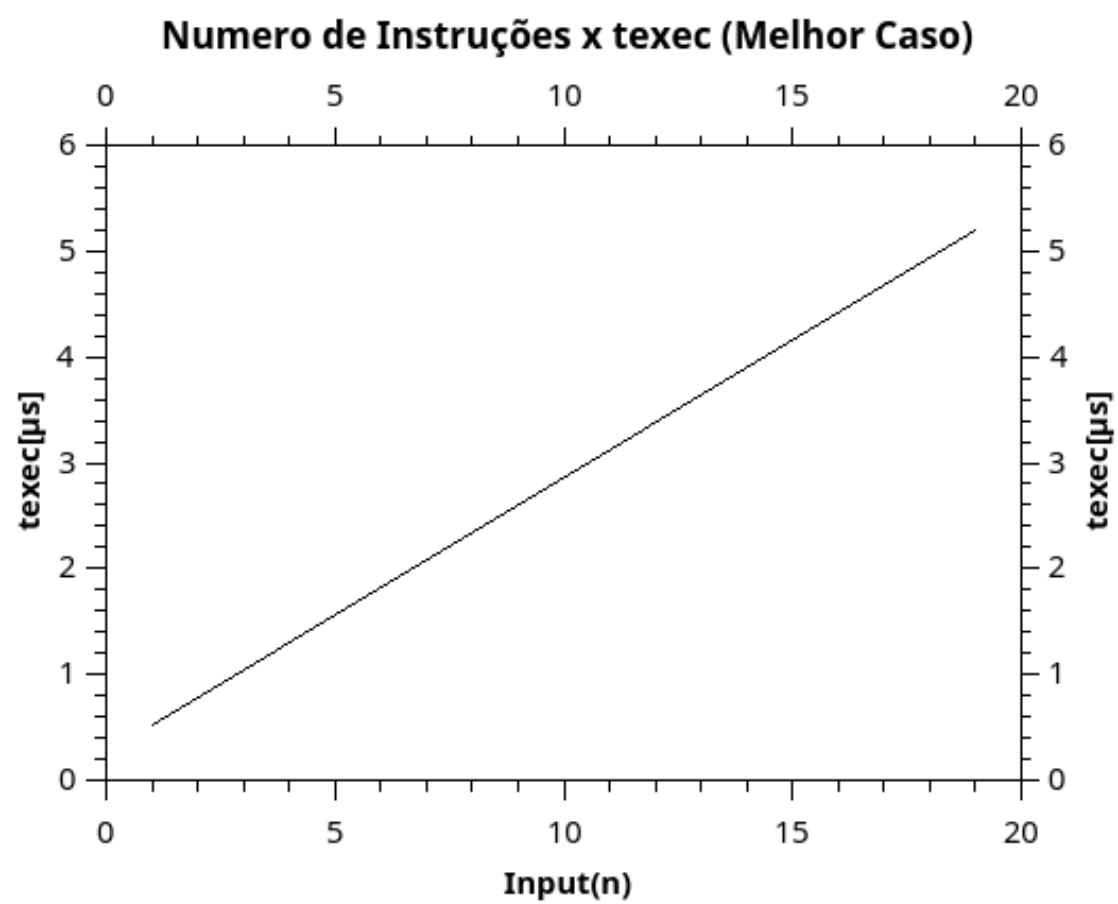
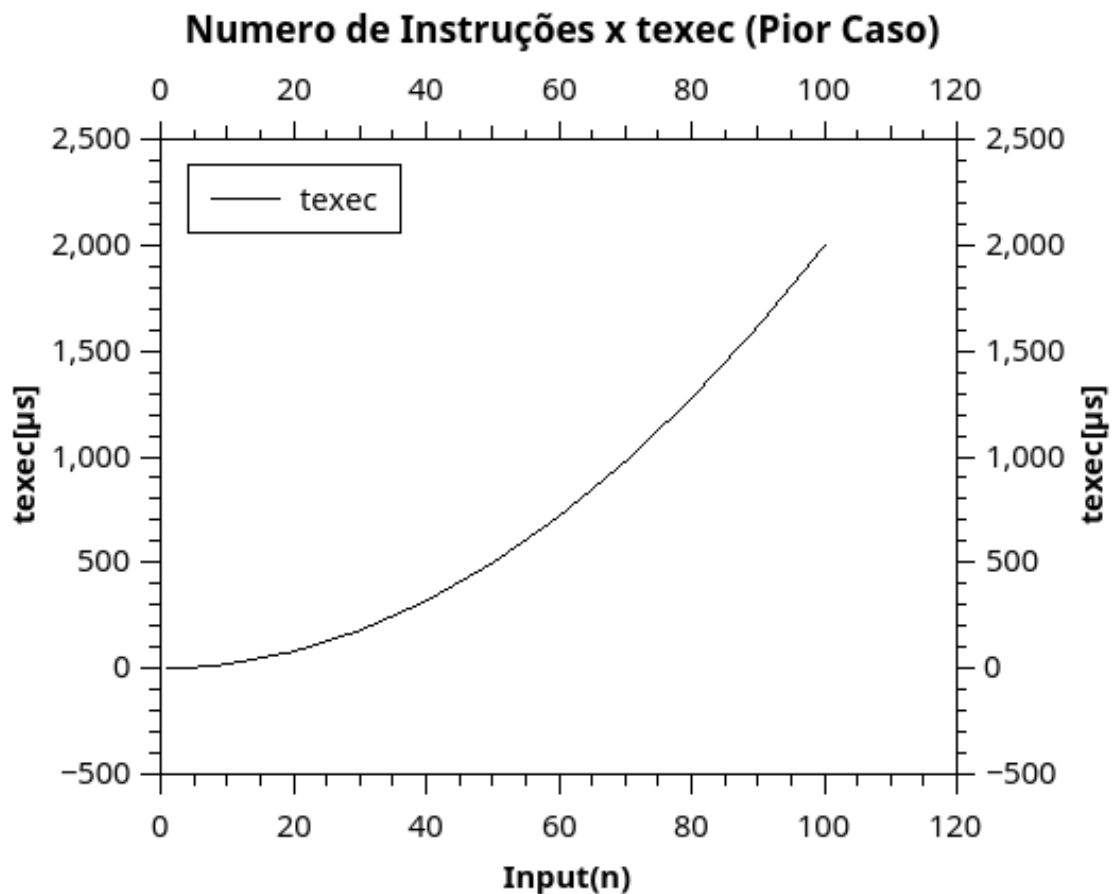


Figura 1.  $n \times \text{texec}$ (Melhor Caso)



**Figura 2. n x texec(Pior Caso)**

Analisando somente os gráficos, percebemos que esse algoritmo tem tempo de execução Linear em seu Melhor caso e em Pior caso, sendo representado por um braço de parábola, tem tempo de execução limitado por  $O(n^2)$ .

### Compilador GCC

Para compilar código C em código Assembly foi utilizado o cross compiler [MIPS] GCC. Usando o comando `$ mips-elf-gcc -I./include -S testeX.c` ( $X \in [0,8]$ ) foi testado a convenção para geração do código Assembly.

### Diretivas

Diretivas são apenas comandos ao montador e não fazem parte do conjunto de instruções dos processadores *x86*. Todas as diretivas começam com `(.)` (*ASCII 0x2E*). Elas permitem a alocação de espaço para a declaração de variáveis `".byte, .word"`, definição de escopo `".globl"`, além de várias outras funções de gerenciamento como as listadas a seguir (as informações abaixo foram encontradas usando as fontes [ORACLE], [SourceWare c], [SourceWare a], [SourceWare b], [GNU], [MTU], [UAF] e [UNIBO]):

- *.file string* : Cria uma tabela de símbolos de entrada onde a *string* é o nome do símbolo e *STT\_FILE* é o tipo deste símbolo, a *string* especifica o nome do arquivo fonte associado ao arquivo objeto.
- *.section section, attributes* : *Section* é montado como seção atual. *Attributes* é incluso se for a primeira vez que *.section* é especificado.
- *.mdebug* : Força a saída de depuração para entrar em uma seção *.mdebug* de estilo ECOFF em vez das seções padrão ELF *.stabs*.
- *.previous* : Troca esta seção pela que foi referenciada recentemente.
- *.nan* : Esta diretiva diz qual codificação *MIPS* será usada para ponto flutuante IEEE 754. A primeira, padrão 2008, diz para o montador utilizar a codificação IEEE 754-2008, enquanto a *legacy* utiliza a codificação original do *MIPS*.
- *.gnu\_attribute tag, value* : Grava um atributo objeto *gnu* para este arquivo.
- *.globl symbol1, symbol2, ..., symbolN*: Torna global cada símbolo da lista. A diretiva torna o símbolo global no escopo mas não declara o símbolo.
- *.data* : Muda a seção atual para *.data* (dados estáticos do programa).
- *.type symbol[, symbol, ..., symbol], type[, visibility]*: Atribui tipo ao símbolo, podendo ser do tipo função, objeto, sem tipo e um objeto *TLS (Thread Local Storage)*.
- *.size symbol, expr*: Resolve expressão e atribui tamanho em bytes ao *symbol*.
- *.word* : Armazena o valor listado como palavras de 32 bits no limite.
- *.rdata* : Adiciona dados apenas de leitura.
- *.align integer* : Ajusta o contador de locação para um valor múltiplo de 2.
- *.ascii "string"*: Aloca espaço para cadeias de caracteres sem o *"\0"*.
- *.text* : Muda a seção atual para *.text* (instruções).
- *.ent name[, label]* : Marca o começo da função *name*.
- *.frame* : Descreve o quadro da pilha usada para chamar a função principal(*main*).
- *.set symbol, expression*: Resolve a expressão (*expression*) e atribui o valor ao símbolo (*symbol*).
- *.mask mask offset* : Configura uma máscara que indica quais registradores de uso geral foram salvos na rotina atual. Esses valores são usados pelo montador para gerar a seção *.reginfo* do arquivo objeto dos processadores *MIPS*.
- *.fmask mask offset* : Configura uma máscara informando os registradores de ponto flutuante que a rotina atual salvou. Esses valores são usados pelo montador para gerar a seção *.reginfo* do arquivo objeto dos processadores *MIPS*.

## Assembly no MARS

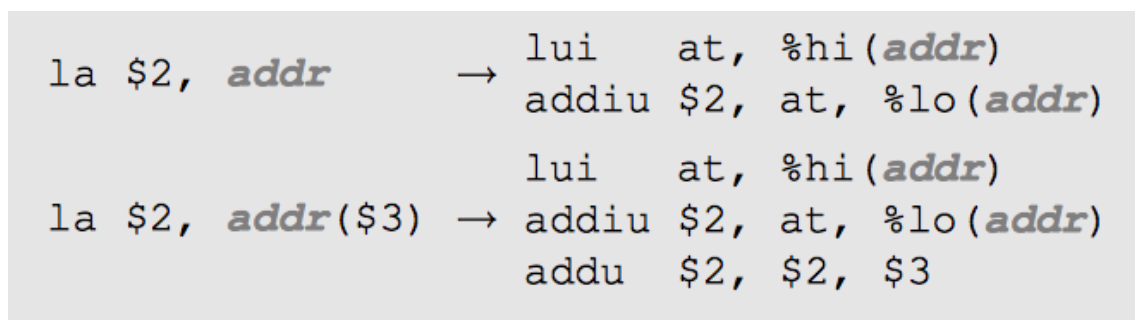
Algumas diretivas listadas acima não são reconhecidas pelo *MARS (Mips Assembly and Runtime Simulator)*, como por exemplo *.section*, *.previous*, *.nan*, etc, assim como alguns elementos como *@object*, *@function*, etc. Logo, as seguintes instruções foram retiradas:

- *@object*
  - *.type v, @object*
- *@function*

- .type show, @function
  - .type swap, @function
  - .type sort, @function
  - .type main, @function
- .section
  - .section .mdebug.abi32
- .previous
  - .previous
- .nan
  - .nan legacy
- .gnu\_attribute
  - .gnu\_attribute 4, 1
- .size
  - .size v, 40
  - .size show, .-show
  - .size swap, .-swap
  - .size sort, .-sort
  - .size main, .-main
- .rdata
  - .rdata
- .set
  - .set nomips16
  - .set nomicromips
  - .set noreorder
  - .set nomacro
  - .set reorder
  - .set macro
- .ent
  - .ent show
  - .ent swap
  - .ent sort
  - .ent main
- .frame
  - .frame \$fp,24,\$31 # vars= 0, regs= 2/0, args= 16, gp= 0
  - .frame \$fp,32,\$31 # vars= 8, regs= 2/0, args= 16, gp= 0
  - .frame \$fp,16,\$31 # vars= 8, regs= 1/0, args= 0, gp= 0
  - .frame \$fp,32,\$31 # vars= 8, regs= 2/0, args= 16, gp= 0

- .mask
  - .mask 0xc0000000,-4
- .fmask
  - .fmask 0x00000000,0
- .end
  - .end swap
  - .end sort
  - .end main
  - .end show
- .ident
  - .ident "GCC: (GNU) 4.8.1"

Há também outras instruções que o simulador emitiu alertas, como por exemplo sobre o *.align* não poder estar dentro de um segmento de texto. Portanto todos *.align* dentro de subrotinas foram retirados.



**Figura 3. Instruções equivalentes dos construtores %hi() e %lo(). Imagem retirada do livro [Sweetman 2005].**

Os construtores %hi() e %lo() também não estão presentes em todos montadores MIPS, podendo ser substituídos como mostra a Figura 3. Logo, as seguintes instruções foram trocadas :

- lui \$v1,%hi(\$LC0) e addiu \$a0,\$v1,%lo(\$LC0) : la \$a0, \$LC0
- lui \$v0,%hi(v) e addiu \$a0,\$v0,%lo(v) : la \$a0, v

Outro problema encontrado foi na instrução *j \$31, ou seja, j \$ra*. A instrução *j* pula para um endereço alvo, e *\$ra* é um registrador, portanto esta instrução foi substituída por *jr \$ra*.

Algumas subrotinas como *printf* e *putchar* não são encontradas no assembly gerado mas são chamadas com a instrução *jal*, portanto estas também foram retiradas.

Por fim, mesmo após estas modificações básicas de sintaxe para que o código possa ser executado no MARS, ainda assim o código possui o fluxo um pouco confuso, já que após o primeiro *loop* ele já iria sair do processo com a instrução *li \$a0, 10* que chama o serviço *exit* (*terminate execution*). Portanto, algumas modificações foram realizadas para que o fluxo deste programa se inicie no *main*, chame as subrotinas *show* e *sort* e encerre no final da subrotina *main*.

## Otimização do código Assembly

O arquivo *sortc.c* foi compilado novamente usando 5 diretivas de compilação, -O0, -O1, -O2, -O3 e -Os. Todos os arquivos gerados, *sortc0.s*, *sortc1.s*, *sortc2.s*, *sortc3.s* e *sortcs.s* foram alterados de acordo com a subseção anterior para que fosse possível executá-los no *MARS*.

## Sprites

Sprite é um objeto gráfico que se move na tela sem deixar traços de sua passagem. Na resolução do exercício, utilizamos as sprites do Ryu para conseguir realizar a atividade de colisão proposta. As sprites utilizadas foram essa: As sprites utilizadas foram:



Figura 4. Movimento 1 do Ryu.



Figura 5. Movimento 2 do Ryu.



Figura 6. Movimento 3 do Ryu.



Figura 7. Movimento 4 do Ryu.

## Análise dos Resultados

### Conclusão

### Referências

[GNU ] GNU, G. Emulated tls. <https://gcc.gnu.org/onlinedocs/gccint/Emulated-TLS.html>. [Online; acessado 18-Setembro-2017].

[MIPS ] MIPS. <https://aur.archlinux.org/packages/cross-mips-elf-gcc/>. [Online; acessado 15-Setembro-2017].

- [MTU ] MTU, M. T. U. Mips assembler directives of gnu assembler. <http://pages.mtu.edu/~mmkoksai/blog/?x=entry:entry120116-130646>. [Online; acessado 18-Setembro-2017].
- [ORACLE ] ORACLE. Assembler directives. [https://docs.oracle.com/cd/E26502\\_01/html/E28388/eoiyg.html](https://docs.oracle.com/cd/E26502_01/html/E28388/eoiyg.html). [Online; acessado 18-Setembro-2017].
- [SourceWare a] SourceWare. Gnu attribute. [https://sourceware.org/binutils/docs-2.26/as/Gnu\\_005fattribute.html#Gnu\\_005fattribute](https://sourceware.org/binutils/docs-2.26/as/Gnu_005fattribute.html#Gnu_005fattribute). [Online; acessado 18-Setembro-2017].
- [SourceWare b] SourceWare. Microblaze directives. <https://sourceware.org/binutils/docs/as/MicroBlaze-Directives.html>. [Online; acessado 18-Setembro-2017].
- [SourceWare c] SourceWare. Mips nan encodings. <https://sourceware.org/binutils/docs-2.27/as/MIPS-NaN-Encodings.html>. [Online; acessado 18-Setembro-2017].
- [Sweetman 2005] Sweetman, D. (2005). *See MIPS Run*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2 edition.
- [UAF ] UAF, U. o. A. F. C compiler. <https://www.cs.uaf.edu/2000/fall/cs301/notes/notes/node97.html>. [Online; acessado 18-Setembro-2017].
- [UNIBO ] UNIBO, U. o. B. Mips assembly language programmer's guide asm-01-doc. [http://www.cs.unibo.it/~solmi/teaching/arch\\_2002-2003/AssemblyLanguageProgDoc.pdf](http://www.cs.unibo.it/~solmi/teaching/arch_2002-2003/AssemblyLanguageProgDoc.pdf). [Online; acessado 18-Setembro-2017].