

# **Laboratório 1**

## **- Assembly MIPS –**

**Dayanne Fernandes da Cunha, 13/0107191**

<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.*

## **1. Introdução**

### **1.1. 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;

### **1.2. Ferramentas**

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

## **2. Procedimentos**

### **2.1. Simulador/Montador MARS**

### **2.2. 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.

#### **2.2.1. 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:

- `.file` :
- `.section` : Texto que segue é montado na seção nomeada.

- `.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` : 3
- `.gnu_attribute` : 4
- `.globl` : Torna símbolo visível ao ligador, aumentando seu escopo.
- `.data` : O que seguir a partir desse ponto deve ser alocado na seção `.data` - Dados.
- `.type` : Atribui tipo ao símbolo, função ou objeto.
- `.size` : Resolve expressão e atribui tamanho em bytes ao símbolo.
- `.word` : Armazena o valor listado como palavras de 32 bits no limite.
- `.rdata` : 5
- `.align` : Ajusta o contador de locação para um valor múltiplo de 2.
- `.ascii` : Aloca espaço para cadeias sem o barra zero.
- `.text` : O que seguir deve ser alocado na seção `.text` - Código.
- `.ent` : Primeira instrução de "entrada" no código. Primeira instrução da função.
- `.frame` : O frame não é descrito no manual.
- `.set` : Resolve expressão e atribui valor ao símbolo.
- `.mask` : Máscaras no registradores usados no código.
- `.fmask` : Máscaras no registradores usados no código.

As diretivas `.mask` e `.fmask` através de seus valores indicam quais registradores são usados, por exemplo se o bit 13 estiver ativo então o registrador 13 está sendo usado. Esses valores são usados pelo ligador para gerar a seção `.reginfo` do arquivo objeto dos processadores MIPS. A diferença entre `.mask` e `.fmask` é que a `.mask` indica quais registradores de inteiros estão sendo usados, quanto ao `.fmask` indica quais registradores float estão sendo usados.

### 2.2.2. 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 *.align* não poder estar dentro de um segmento de texto, portanto todos *.align* dentro de subrotinas foram retirados. 'x

```

la $2, addr      → lui  at, %hi(addr)
                   addiu $2, at, %lo(addr)

la $2, addr($3) → lui  at, %hi(addr)
                   addiu $2, at, %lo(addr)
                   addu  $2, $2, $3
  
```

**Figura 1. 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 1. 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*).

### **2.2.3. Otimização do código Assembly**

### **2.3. Sprites**

## **3. Análise dos Resultados**

## **4. Conclusão**

### **Referências**

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

[Sweetman 2005] Sweetman, D. (2005). *See MIPS Run*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2 edition.