Organização de Computadores
Programação em Assembly MIPS

Matheus Gabriel

Agosto 2024

1 Introdução

A linguagem Assembly é uma das formas mais diretas de interagir com a arquitetura de um computador, oferecendo um nível de controle próximo ao da linguagem de máquina. Neste semestre, focaremos na programação em Assembly para a arquitetura MIPS. Utilizaremos o IDE MARS (MIPS Assembler and Runtime Simulator) para a edição e execução dos nossos códigos.

2 Programação básica

Os passos aqui descritos não são necessariamente os mesmo passos de todo código Assembly (obviamente).

2.1 O arquivo

Para os exemplos discutidos neste documento, utilizamos o arquivo denominado `ExemploOperacoesAritmeticas.asm`, disponibilizado pelo professor na plataforma Moodle. Os comentários são do professor.

> **Aviso**
>
> O arquivo .asm do exemplo, e a maioria (senão todos) dos outros exemplos de código não são incluídos no repositório git, porém, em alguns documentos as explicações juntas já formam o código completo.

2.2 Definição de variáveis

```
1 .data
2 # .data EH O SEGMENTO ONDE DECLARAMOS AS "VARIAVEIS"
3
4 N1: .asciiz "Digite o 1o numero: "
5 N2: .asciiz "Digite o 2o numero: "
6 MsgSoma: .asciiz "\nSoma = "
```

```
MsgSub: .asciiz "\nSubtracao = "
MsgMul: .asciiz "\nMultiplicacao = "
```

- `.data` indica o começo da seção onde são declaradas as variáveis
- `.asciiz` armazena a string no segmento de dados e adiciona um terminador nulo `\0`.

Esse código seria equivalente à `MsgSoma = "Soma = "` na linguagem Python

2.3 Atribuição de valores aos registradores e entrada do usuário

```
.text
# .text EH O SEGMENTO QUE CONTEM AS INSTRUCOES/PROGRAMA

# EM ASSEMBLY, A SYSCALL EH UTILIZADA PARA VARIAS ACOES: LER DADOS DO TECLADO,
# IMPRIMIR ALGO NA TELA, FINALIZAR O PROGRAMA, ETC

# IMPRIMIR A MENSAGEM N1 NA TELA
li $v0,4 # ATRIBUINDO O VALOR 4 PARA O REG $v0
# (COD 4: INDICA PRINT DE STRING P/ SYSCALL)
la $a0,N1 # CARREGANDO O END. DA VAR N1 PARA DENTRO DO REGISTRADOR a0
# syscall #SYSCALL CHAMA O SO PARA EXECUTAR A ACAO RELATIVA AO CODIGO QUE ESTA NO $v0

# LER UM INTEIRO DIGITADO PELO USUARIO

li $v0,5 # ATRIBUINDO O VALOR 5 PARA O REG $v0
# (COD 5: INDICA INPUT DE INT P/ SYSCALL)
syscall

# FAZER UMA COPIA DO DADO LIDO
move $t0, $v0 # COPIANDO O CONTEUDO DE $v0 PARA $t0


# IMPRIMIR A MENSAGEM N2 NA TELA
li $v0,4 # ATRIBUINDO O CONTEÚDO DO $v0 PARA 4 (PRINT)
la $a0,N2
syscall

# LER UM INTEIRO DIGITADO PELO USUARIO
li $v0,5 # ATRIBUINDO O VALOR 5 PARA O REG $v0
# (COD 5: INDICA INPUT DE INT P/ SYSCALL)
syscall

# FAZER COPIA DO DADO LIDO
move $t1, $v0 # COPIANDO O CONTEUDO DE $v0 PARA $t1
```

- `.text` indica o começo da seção que contém as instruções do programa
- `li` atribui uma **constante** (e.g 4, 2) para o registrador especificado.
- `la` atribui uma **variável** (e.g N1, MsgSoma) para o registrador especificado.
- `syscall` chama o sistema operacional para executar as ações definidas anteriormente.
- `move` copia o conteúdo de um registrador para outro.

Resumindo, esse trecho de código atribui os valores definidos em `.data` para os registradores para serem impressos depois. As constantes usadas em `li` são específicas para o tipo de syscall que o usuário deseja fazer, use a tabela de syscalls do MIPS como referência.

2.4 Cálculo das operações

```
# SOMA $t0 COM $t1 E ARMAZENA O RESULTADO EM $t2
add $t2, $t0, $t1
# add EH A OPERACAO DE ADICAO
# $t2 É O REGISTRADOR DE DESTINO
# $t0 E $t1 SÃO OS REGISTRADORES DAS VARIAVEIS N1 E N2

# IMPRIMIR A MENSAGEM DE SOMA = NA TELA
li $v0,4
la $a0,MsgSoma
syscall

# IMPRIMIR O RESULTADO NA TELA
li $v0,1
move $a0, $t2
syscall

# SUBTRACAO $t0 COM $t1 E ARMAZENA O RESULTADO EM $t2
sub $t2, $t0, $t1
# sub EH A OPERACAO DE SUBTRACAO
# $t2 É O REGISTRADOR DE DESTINO
# $t0 E $t1 SÃO OS REGISTRADORES DAS VARIAVEIS N1 E N2

# IMPRIMIR A MENSAGEM DE RESPOSTA NA TELA
li $v0,4
la $a0,MsgSub
syscall

# IMPRIMIR O RESULTADO NA TELA
```

```
29 li $v0,1
30 move $a0, $t2
31 syscall
32
33 # MULTIPLICACAO £t0 COM £t1 E ARMAZENA O RESULTADO EM £t2
34 mul $t2, $t0, $t1
35 # mul EH A OPERACAO DE MULTIPLICACAO
36 # £t2 É O REGISTRADOR DE DESTINO
37 # £t0 E £t1 SÃO OS REGISTRADORES DAS VARIAVEIS N1 E N2
38
39 # IMPRIMIR A MENSAGEM DE RESPOSTA NA TELA
40 li $v0,4
41 la $a0,MsgMul
42 syscall
43
44 # IMPRIMIR O RESULTADO NA TELA
45 li $v0,1
46 move $a0, $t2
47 syscall
48
49 li $v0, 10        # 10 É O COD. QUE A SYSCALL USA PARA ENCERRAR O PROGRAMA
50 syscall               # SYSCALL CHAMA O SO PARA EXECUTAR A ACAO RELATIVA AO CODIGO QUE EST
```

- `add` realiza a operação de adição
- `sub` realiza a operação de subtração
- `mult` realiza a operação de multiplicação

Após realizar as operações especificadas, os resultados são atribuídos à registradores e seus valores são impressos na tela.