Primeiro Trabalho Prático de AOC I
Assembly MIPS

Pablo Cecilio Oliveira
Marcilene Reis

1 Introdução

Este documento apresenta a solução em MIPS para as duas questões do primeiro trabalho prático de Arquitetura e Organização de Computadores I.

2 Implementação

Toda a implementação foi realizada utilizando o programa de desenvolvimeto MARS (*MIPS Assembler and Runtime Simulator*) como IDE.

2.1 Questão 1

O procedimento para calcular o *cosseno* de um ângulo segundo a série de Taylor abaixo:

$$\cos x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!} x^{2n}$$

A função *seno* chama funções para calcular o fatorial e a função potência de x. O usuário digita o ângulo em radianos e a quantidade de termos da série ($n > 0$).

2.2 Questão 2

O programa da Questão 2 recebe uma *string* do arquivo de entrada (`entrada.txt`) e gera um arquivo de saída (`saida.txt`) contendo uma nova *string* processada. O processamento da *string* consiste na troca dos caracteres maiúsculos por minúsculos, e seus caracteres minúsculos por maiúsculos.

Para a implementação desse programa, uma lista de instruções e pseudo-instruções foi utilizada com o objetivo de simplificar o desenvolvimento.

- A pseudo-instrução `li` (*Load Immediate*), carrega um valor imediato em um registrador.
- A pseudo-instrução `la` (*Load Address*), faz com que um registrador receba o valor de um ponteiro de outro local na memória.
- A pseudo-instrução `move`, move o conteúdo de um registro para outro registro.

- A instrução `lb` (*Load Byte*), transfere um byte de dados de uma posição da memória principal para um registrador.
- A instrução `sb` (*Store Byte*), transfere um byte de dados de um registro para uma posição na memória principal.

As instruções do programa são listadas e comentadas abaixo:

Inicialmente, são declarados na área de segmento de dados do programa as variáveis a serem utilizadas e seus valores:

Área de declaração de variáveis

`.data`		
`fin:`	`.asciiz "entrada.txt"`	Endereço do caminho do arquivo de entrada.
`fout:`	`.asciiz "saida.txt"`	Endereço do caminho do arquivo de saída.
`string:`	`.space 1024`	Reserva 1024 bytes para o *buffer* que conterá a *string*.

Passa-se então a área de instruções `.text`, que contém os três procedimentos a serem realizados. A chamada para esses procedimentos é feita no escopo principal do programa (`main`):

Escopo principal do programa

`main:`	
`la $a0,fin`	Carrega o endereço de `fin` para a `$a0`.
`jal leArquivo`	Inicia o procedimento `leArquivo`.
`la $a0,string`	Carrega o endereço de `string` para a `$a0`.
`jal manipulaString`	Inicia o procedimento `manipulaString`.
`la $a0,fout`	Carrega o endereço de `fout` para a `$a0`.
`jal salvaArquivo`	Inicia o procedimento `salvaArquivo`.
`li $v0,10`	*load immediate* 10 em `$v0`
`syscall`	Executa o número de serviço em `$v0`. Terminar programa.

Resumidamente o processo no `main` consiste em três partes:

1. Abra o arquivo e carregue a *string*; salve o numero de caracteres lidos.
2. Manipule a *string*; troque letras maiúsculas por minusculas e vice-versa.
3. Salve para o arquivo; salva a *string* modificada no arquivo de saída.

Os procedimentos do programa são listados e comentados a seguir:

Procedimento para ler o arquivo

`leArquivo:`	
`li $v0,13`	Seta o número de serviço para a abertura de arquivo.
`li $a1,0`	Seta flag = 0, *read-only*.
`li $a2,0`	modo ignorado
`syscall`	Executa o número de serviço em `$v0`. Abrir arquivo.
`move $t0,$v0`	Salva o descritor do arquivo em `$v0` para `$t0`.
`li $v0,14`	Seta o número de serviço para a leitura de arquivo.
`move $a0,$t0`	Carrega o descritor do arquivo em `$t0` para `$a0`.
`la $a1,string`	Carrega o endereço de `string` para a `$a1`.
`li $a2,255`	hardcoded buffer length
`syscall`	Executa o número de serviço em `$v0`. Ler arquivo.
`move $s0,$v0`	Salva o número de caracteres lidos em `$s0`
`li $v0,16`	Seta o número de serviço para a fechar o arquivo.
`syscall`	Executa o número de serviço em `$v0`. Fechar arquivo.
`jr $ra`	Retorna do procediemnto

Tabela 2.2.1: Tabela ASCII para A-Z e a-z

65	A	69	E	73	I	77	M	81	Q	85	U	89	Y
66	B	70	F	74	J	78	N	82	R	86	V	90	Z
67	C	71	G	75	K	79	O	83	S	87	W		
68	D	72	H	76	L	80	P	84	T	88	X		
97	a	101	e	105	i	109	m	113	q	117	u	121	y
98	b	102	f	106	j	110	n	114	r	118	v	122	z
99	c	103	g	107	k	111	o	115	s	119	w		
100	d	104	h	108	l	112	p	116	t	120	x		

Fonte: Wikipedia contributors [2]

Procedimento para alterar a string

`manipulaString:`	
`Loop:`	loop para cada caractere da string
`lb $t2,($a0)`	Aponta $t2 para a posição de endereço de `$a0`
`beq $t2,$zero,End`	Se $t2 conter NULL, a string terminou, `End`
`slti $t1,$t2,90`	$t2 e menor que 90(Z)?
`bne $t1,$zero,Upper`	Se $t1 for 1 então 1 != 0, vai pra `Upper`
`slti $t1,$t2,122`	$t2 e menor que 122(z)?
`bne $t1,$zero,Lower`	Se $t1 for 1 então 1 != 0, vai pra `Lower`
`j Next`	Va pra Next
`Upper:`	rotina para uppercase
`slti $t1,$t2,65`	$t2 e menor que 65(A)?
`bne $t1,$zero,Next`	Se $t1 for 1 então não é uma letra, `Next`
`addi $t2,$t2,32`	Soma 32 a $t2
`sb $t2,($a0)`	armazena o valor na posição em `$a0`
`j Next`	Va pra Next
`Lower:`	rotina para lowercase
`slti $t1,$t2,97`	$t2 e menor que 97(a)?
`bne $t1,$zero,Next`	Se $t1 for 1 então não é uma letra, Next
`addi $t2,$t2,-32`	Soma -32 a `$t2`
`sb $t2,($a0)`	armazena o valor na posição em `$a0`
`Next:`	Continua a interação
`addi $a0,$a0,1`	Incrementa o endereço de `$a0`
`j Loop`	Va para Loop
`End:`	Termino do Loop
`jr $ra`	Retorna do procedimento

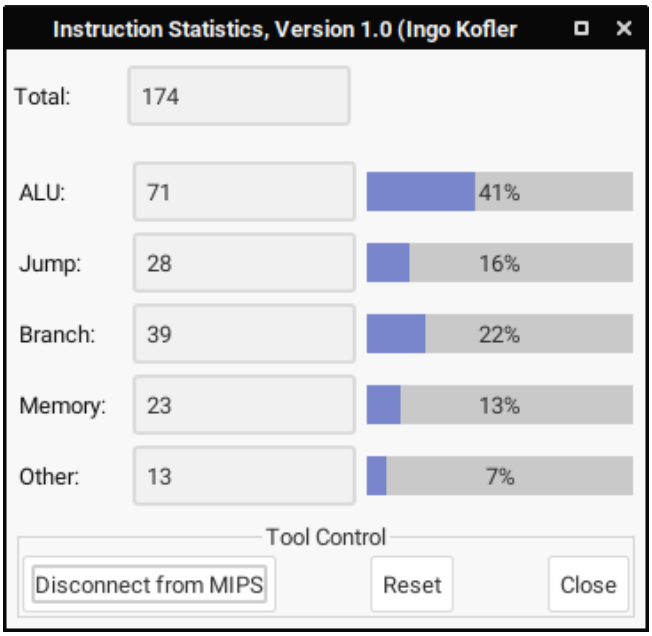
Procedimento para salvar o arquivo

`salvaArquivo:`	
`li $v0, 13`	system call: Abre o arquivo
`li $a1, 1`	flag para escrita
`li $a2, 0`	modo ignorado
`syscall`	Abra o arquivo!
`move $t0,$v0`	Salva o descriptor em $t0
`li $v0,15`	system call: Escrevendo no arquivo
`move $a0,$t0`	Carrega o descriptor do arquivo
`la $a1,string`	endereço do buffer
`move $a2,$s0`	carrega strlen de $s0
`syscall`	Escreva no arquivo!
`li $v0,16`	system call: Fecha o arquivo
`syscall`	Feche o arquivo!
`jr $ra`	Retorna do procedimento

3 Estatísticas das instruções

Os resultados fornecidos pela ferramenta de estatísticas de instruções do MARS são sumarizados para cada questão nas figuras abaixo:

Questão 2, resultado da análise



Fonte: Mars Instruction Statistic Tool

Referências

[1] Missouri State University. Syscall functions available in mars. http://courses.missouristate.edu/kenvollmar/mars/help/syscallhelp.html. [Acesso em: 13-Outubro-2018].

[2] Wikipedia contributors. Ascii — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=ASCII&oldid=863023280, 2018. [Acesso em: 13-Outubro-2018].

O histórico do desenvolvimento desse trabalho se encontra online em: https://github.com/Durfan/ufsj-aoc1-tp1.