

Relatório do Trabalho de Organização de Computadores – Cifra de César

Ederson Renan de Bomfim, ra: a2374986

Resumo—Este é um relatório referente aos processos de desenvolvimento do trabalho proposto na disciplina Organização de Computadores [CC53B] no primeiro período de 2022 no curso Bacharelado em Ciência da Computação da UTFPR – Campus Ponta Grossa – Trata-se da implementação de uma das mais simples e antigas formas de criptografias conhecida como Cifra de César – que consiste em avançar cada letra de uma mensagem K posições no alfabeto, onde K é a chave usada na criptografia, e posteriormente conhecendo-se a chave pode-se fazer o processo inverso para decifrar a mensagem original – na linguagem Assembly da arquitetura MIPS32. Utilizou-se o simulador MARS (MIPS Assembler and Runtime Simulator) para o desenvolvimento e testes da aplicação.

Index Terms—Organização de computadores, assembly MIPS32, arquitetura de computadores, cifra de César

1 INTRODUÇÃO

Em criptografia, a Cifra de César [1], também conhecida como cifra de troca, código de César ou troca de César, é uma das mais simples e conhecidas técnicas de criptografia.

É um tipo de cifra de substituição na qual cada letra do texto é substituída por outra, que se apresenta no alfabeto abaixo dela um número fixo de vezes, por exemplo “ABC” com 1 como número fixo, se tornaria “BCD”.

O processo de criptografia de uma cifra de César é frequentemente incorporado como parte de esquemas mais complexos, como a cifra de Vigenère [2] por isso continua tendo alguma aplicação moderna.

O escopo deste trabalho propõe o uso do simulador MARS (MIPS Assembler and Runtime Simulator) [3] da linguagem do Assembler da arquitetura de computadores MIPS32 [4], para criar uma aplicação que resolve a Cifra de César, tanto encriptando uma mensagem com uma chave definida pelo usuário, quanto decifrando uma mensagem cifrada quando se sabe a chave usada na criptografia original.

2 DESENVOLVIMENTO

2.1 Tabela ASCII e Caractéres válidos

A tabela ASCII (*American Standard Code for Information Interchange*) [5] é uma convenção de caracteres que podem ser representados em um único byte (8 bits) em um meio digital, onde cada valor numérico representa um caracter. Com 8 bits, é possível representar $2^8 = 255$ caracteres, que correspondem à representação binária dos números de -127 a 127 em decimal, sendo o primeiro bit do caracter o

bit de sinal (0 = número positivo, 1 = número negativo). Os caracteres de 0 a 127 fazem parte da tabela ASCII comum, já o conjunto todo [-127,127] é chamado de tabela ASCII expandida.

Como definido no arquivo de enunciado do trabalho da disciplina, o programa deve aceitar letras maiúsculas e minúsculas, números e caracteres especiais, porém deve desconsiderar os acentos e o caracter ‘ç’. Com isso, foi possível concluir que os caracteres aceitos deveriam ser os caracteres gráficos ¹ da tabela ASCII comum, ou seja, os caracteres com valor em ASCII de 32 até 126. Além disso, o caracter 32 corresponde ao espaço, então é um caracter válido na mensagem de entrada, porém é ignorado na cifra, ou seja, os espaços da mensagem continuarão sendo espaços mesmo após a cifra.

A lista dos caracteres aceitos é a seguinte (observe a coluna “Sinal”):

Bin	Oct	Dec	Hex	Sinal	Bin	Oct	Dec	Hex	Sinal	Bin	Oct	Dec	Hex	Sinal
0010 0000	040	32	20	(espaço)	0100 0000	100	64	40	@	0110 0000	140	96	60	`
0010 0001	041	33	21	!	0100 0001	101	65	41	A	0110 0001	141	97	61	a
0010 0010	042	34	22	"	0100 0010	102	66	42	B	0110 0010	142	98	62	b
0010 0011	043	35	23	#	0100 0011	103	67	43	C	0110 0011	143	99	63	c
0010 0100	044	36	24	\$	0100 0100	104	68	44	D	0110 0100	144	100	64	d
0010 0101	045	37	25	%	0100 0101	105	69	45	E	0110 0101	145	101	65	e
0010 0110	046	38	26	&	0100 0110	106	70	46	F	0110 0110	146	102	66	f
0010 0111	047	39	27	'	0100 0111	107	71	47	G	0110 0111	147	103	67	g
0010 1000	050	40	28	(0100 1000	110	72	48	H	0110 1000	150	104	68	h
0010 1001	051	41	29)	0100 1001	111	73	49	I	0110 1001	151	105	69	i
0010 1010	052	42	2A	*	0100 1010	112	74	4A	J	0110 1010	152	106	70	j
0010 1011	053	43	2B	+	0100 1011	113	75	4B	K	0110 1011	153	107	71	k
0010 1100	054	44	2C	,	0100 1100	114	76	4C	L	0110 1100	154	108	72	l
0010 1101	055	45	2D	-	0100 1101	115	77	4D	M	0110 1101	155	109	73	m
0010 1110	056	46	2E	.	0100 1110	116	78	4E	N	0110 1110	156	110	74	n
0010 1111	057	47	2F	/	0100 1111	117	79	4F	O	0110 1111	157	111	75	o
0011 0000	060	48	30	0	0101 0000	120	80	50	P	0111 0000	160	112	76	p
0011 0001	061	49	31	1	0101 0001	121	81	51	Q	0111 0001	161	113	77	q
0011 0010	062	50	32	2	0101 0010	122	82	52	R	0111 0010	162	114	78	r
0011 0011	063	51	33	3	0101 0011	123	83	53	S	0111 0011	163	115	79	s
0011 0100	064	52	34	4	0101 0100	124	84	54	T	0111 0100	164	116	80	t
0011 0101	065	53	35	5	0101 0101	125	85	55	U	0111 0101	165	117	81	u
0011 0110	066	54	36	6	0101 0110	126	86	56	V	0111 0110	166	118	82	v
0011 0111	067	55	37	7	0101 0111	127	87	57	W	0111 0111	167	119	83	w
0011 1000	070	56	38	8	0101 1000	130	88	58	X	0111 1000	170	120	78	x
0011 1001	071	57	39	9	0101 1001	131	89	59	Y	0111 1001	171	121	79	y
0011 1010	072	58	3A	:	0101 1010	132	90	5A	Z	0111 1010	172	122	80	z
0011 1011	073	59	3B	;	0101 1011	133	91	5B	[0111 1011	173	123	81	{
0011 1100	074	60	3C	<	0101 1100	134	92	5C	\	0111 1100	174	124	82	
0011 1101	075	61	3D	=	0101 1101	135	93	5D]	0111 1101	175	125	83	}
0011 1110	076	62	3E	>	0101 1110	136	94	5E	^	0111 1110	176	126	84	~
0011 1111	077	63	3F	?	0101 1111	137	95	5F	_					

• E.R. Bomfim é estudante de graduação no curso Bacharelado em Ciência da Computação, UTFPR Ponta Grossa, PR.
E-mail: edersonb@alunos.utfpr.edu.br

1. A tabela ASCII com 128 caracteres possui 95 caracteres visíveis e 33 caracteres de controle, que são usados em funções como marcar fim de uma linha, marcar fim de um arquivo, início de um cabeçalho, etc...

2.2 Método de Comunicação Com o Usuário

Para comunicar-se com o usuário (pedir os valores de entrada e passar instruções) foram usadas as formas de entrada padrão do simulador, com isso são exibidas mensagens pré-definidas no código para o usuário em um console na IDE², e o usuário pode inserir os valores que desejar em cada passo da execução do programa. Além disso foi usada uma segunda forma de entrada de dados em que o usuário informa apenas o nome de um arquivo (que deve estar na mesma pasta do arquivo executável .jar do MARS) de texto contendo as opções de entrada (mensagem a ser cifrada ou decifrada, chave de criptografia e se a mensagem deve ser cifrada ou decifrada).

2.3 Método Usado Para Aplicar a Cifra

A forma inicial de tratar o problema, após definir os caracteres válidos, é pensar no pseudo-alfabeto gerado por esses caracteres como um ciclo, no qual caso avançando-se K caracteres chega-se ao fim, deve-se retornar ao início.

Após ler a mensagem a ser criptografada e a chave a ser usada na criptografia, a forma de resolver o problema torna-se simples, trata-se um loop³ que passa por todos os caracteres da mensagem até detectar o fim e, em cada caracter, soma o valor da chave ao valor em ASCII do caracter, por exemplo: se o caracter for “A” – 65 em ASCII e a chave for 3 (avancar 3 letras), o programa deve somar $65+3 = 68$, que é o código em ASCII correspondente a “D”. Após isso somente é necessário tratar o caso do código resultante da soma do valor do caracter com a chave ser maior que 126 (“ultimo caracter valido”) então diminui-se 94 caracteres para reiniciar o ciclo do pseudo-alfabeto.

2.4 Decifrar

O método usado para decifrar uma mensagem é o mesmo do método para cifrar uma mensagem, porém ao invés de somar o valor da chave aos caracteres, deve-se diminuir-lo, e também verificar se após essa subtração o valor do caracter não chega ao valor de 32 ou menor (pois os espaços não devem contar para a cifra) e se for o caso somar 94 nesse valor, para reiniciar o ciclo do extremo superior.

2.5 Entrada por Arquivo

Caso o usuário escolha a entrada por arquivo, deve informar o nome do arquivo (com sua extensão) que deve estar na mesma pasta do arquivo executável do MARS e este arquivo deve ter um padrão: deve conter apenas 1 linha onde primeiro informa-se a opção do que quer fazer (0 para cifrar a mensagem, 1 para decifrar) a chave de criptografia e em seguida a mensagem, os 3 devem ser separados por 1 (um) espaço, após o espaço que marca o

início da mensagem, podem existir qualquer quantidade de espaços, pois já farão parte da mensagem a ser cifrada ou decifrada.

3 UTILIZAÇÃO

3.1 Executar o Arquivo

Para utilização do programa, primeiro é necessário ter baixado o simulador MARS e o arquivo fonte disponibilizado junto a este documento chamado “codigo.asm”, clicar na aba file|open do MARS e escolher o arquivo “codigo.asm”, executar o código apertando F3 e em seguida clicar no símbolo de “play” verde na parte superior da tela.

3.2 Menu

Assim que o programa inicia, é exibido a tela de instruções e informações, e logo em seguida o menu para que o usuário faça a escolha dentre as diferentes funcionalidades do programa.

```
=====MENU=====
1 - Cifrar
2 - Decifrar
3 - Entrada por arquivo
0 - Sair
```

O menu possui 4 opções, com as seguintes funcionalidades:

0. **Sair** – finaliza o programa
1. **Cifrar** – Criptografar uma mensagem dada pelo usuário na entrada padrão, com uma chave também informada pelo usuário, o programa escreve na saída padrão (console do simulador) a mensagem criptografada com a chave informada.
2. **Decifrar** – O usuário informa uma mensagem criptografada e a chave com que a mensagem foi criptografada, pela entrada padrão, o programa escreve na saída padrão (console do simulador) a mensagem decifrada a partir da chave informada.
3. **Entrada por Arquivo** – O usuário informa apenas o nome de um arquivo com a extensão (por exemplo “arquivo.txt”), o programa irá ler o arquivo (de forma explicada no item 2.5) e escreve a frase de saída (cifrada ou decifrada) em um arquivo chamado “cesar_output.txt”, que se não existir na mesma pasta do executável é criado (ou sobrescrito no caso de existir).

Caso qualquer outra opção seja informada, uma mensagem de opção inválida será exibida. Logo após o menu ser exibido o usuário já pode inserir a opção e apertar [Enter] para enviar. Sempre após uma operação ser executada o menu é exibido novamente para que o usuário continue o uso do programa, até que digite 0 (sair).

3.3 Padronização e Tratamento de Erros

O programa não se propõe a tratar todos os erros para as possibilidades de o usuário digitar valores fora do padrão de entrada (por exemplo, digitar uma letra quando se pede um número), por isso definimos os seguintes padrões:

- Mensagem para criptografar: sequência de caracteres

2. IDE (Integrated Development Environment) – ambiente de desenvolvimento integrado.

3. Loops, em programação, referem-se ao conceito de executar uma mesma parte de um código repetidas vezes até que uma certa condição seja satisfeita

válidos (apontados na secção 2.1).

- Chave criptográfica: número inteiro maior que ou igual a zero e menor que ou igual a 94.
- Nome do arquivo de texto (plain text): <nome>.<extensão>, por exemplo: arquivo.txt, file.md ...

4 RESULTADOS

4.1 Exemplo na Opção “Cifrar”

Após escolher a opção “Cifrar” no menu, o programa irá pedir para que o usuário dê como entrada a mensagem que será criptografada e em seguida a chave de criptografia. No exemplo, usa-se a mensagem “frase 123!” e a chave 2, que significa que a cada caracter será somado 2:

```
=====MENU=====
1 - Cifrar
2 - Decifrar
3 - Entrada por arquivo
0 - Sair
1
Informe o texto a ser CIFRADO: frase 123!
Informe a chave para cifrar: 2
htcug 345#
```


4.2 Exemplo na opção “Decifrar”

A entrada caso o usuário escolha a opção “Decifrar” é muito similar à opção “Cifrar”, possuindo a mensagem e a chave, porém, irá retroceder os caracteres envés de avançar. No exemplo, é usado a mesma mensagem cifrada na saída anterior para verificar se foi decifrada corretamente:

```
=====MENU=====
1 - Cifrar
2 - Decifrar
3 - Entrada por arquivo
0 - Sair
2
Informe o texto a ser DECIFRADO: htcug 345#
Informe a chave com que a mensagem foi cifrada: 2
frase 123!
```

4.3 Exemplo de Cifrar por Arquivo

Caso o usuário escolha a opção “Entrada por arquivo”, o programa irá exibir algumas instruções e pedir o nome do arquivo escolhido. No primeiro exemplo, usaremos o arquivo chamado “entrada.txt” com o seguinte texto:

 entrada.txt - Notepad

File Edit Format View Help

0 2 frase 123!|


=====MENU=====

```
1 - Cifrar
2 - Decifrar
3 - Entrada por arquivo
0 - Sair
```

3

ATENÇÃO: o arquivo de entrada deve estar na mesma pasta do executável do MARS e deve sempre ter apenas 1 linha contendo: A opção (0 para cifrar ou 1 para decifrar), a chave de criptografia (seja para cifrar ou decifrar a mensagem) e em seguida a mensagem, os 3 separados por espaços INFORME O NOME DO ARQUIVO (com a extensão):
 entrada.txt
 htcug 345#
 Escrito no arquivo "cesar_output.txt"

Com isso, como a opção (primeiro caracter) é 0 - cifrar, o programa exibe o resultado na saída padrão e além disso escreve-a no arquivo (que será criado, pois ainda não existe) “cesar_output.txt” com o seguinte conteúdo:


 cesar_output.txt - Notepad

File Edit Format View Help

htcug 345#

4.4 Exemplo de Decifrar por Arquivo

Neste exemplo o usuário informa a opção 3 de entrada por arquivos e em seguida o arquivo “entrada.txt”, que possui o seguinte conteúdo:

 entrada.txt - Notepad

File Edit Format View Help

1 2 htcug 345#|

=====MENU=====


```
1 - Cifrar
2 - Decifrar
3 - Entrada por arquivo
0 - Sair
```

3

ATENÇÃO: o arquivo de entrada deve estar na mesma pasta do executável do MARS e deve sempre ter apenas 1 linha contendo: A opção (0 para cifrar ou 1 para decifrar), a chave de criptografia (seja para cifrar ou decifrar a mensagem) e em seguida a mensagem, os 3 separados por espaços INFORME O NOME DO ARQUIVO (com a extensão):
 entrada.txt
 frase 123!
 Escrito no arquivo "cesar_output.txt"

Como a opção (primeiro caracter) é 1 - decifrar, o programa exibe a mensagem decifrada usando a chave 2 na saída padrão e em seguida escreve-a no arquivo “cesar_output.txt” que neste caso é sobrescrito, pois já existe da iteração do exemplo anterior:

Após a execução do programa o arquivo “cesar_output.txt” foi alterado e contém o seguinte texto:

 cesar_output.txt - Notepad

File Edit Format View Help

frase 123!

REFERÊNCIAS

- [1] G. PRICHETT, D. L. “Cryptology: From caesar ciphers to publickey cryptosystems.” 2-17.
- [2] S. D. Nasution, et al “Data Security Using Vigenere Cipher and Goldbach Codes Algorithm”, International Journal of Engineering Research & Technology (IJERT), vol. 6 issue 01, Jan 2017.
- [3] K. Vollmar, P. S., "A MIPS Assembly Language Simulator Designed for Education.", ACM SIGCSE Bulletin 239-243 (SIGCSE 2006 paper) url: <https://courses.missouristate.edu/KenVollmar/mars/fp288-vollmar.pdf>
- [4] WikiChip, “MIPS32 Instruction Set” url: https://en.wikichip.org/wiki/mips/mips32_instruction_set
- [5] American Standard Association, “American Standard Code for Information Interchange”. url: <https://web.archive.org/web/20160617012149/http://worldpwersystems.com/1/codes/X3.4-1963/>