

# Decifrando a Cifra de César - Implementação em Assembly

Thomaz Abrantes de Oliveira Martinelli Silva\*, Pedro Augusto Soares†  
Escola Politécnica — PUCRS

9 de julho de 2024

## Resumo

Este trabalho apresenta a implementação da decodificação da Cifra de César utilizando a linguagem Assembly. Esta implementação, desenvolvida dentro do escopo da disciplina de Fundamentos de Sistemas Digitais, visa explorar conceitos de manipulação de dados em baixo nível e operações aritméticas em Assembly. A decodificação é baseada na matrícula do aluno, onde o deslocamento é calculado dinamicamente em tempo de execução. A abordagem permite reforçar a compreensão dos conceitos de codificação e decodificação de dados, além de proporcionar uma experiência prática com a linguagem Assembly, que será melhor detalhada na próxima disciplina, Organização e Arquitetura de Processadores (OAP).

## Palavras-chave

Cifra de César, Algoritmo, Codificação, Decodificação, Assembly, MIPS, MARS, ModelSim

## Introdução

A Cifra de César, também conhecida como cifra de troca, é uma das técnicas de criptografia mais antigas e simples de que se tem conhecimento até os dias atuais. Ela consiste na substituição de cada letra do texto original por outra letra situada um número fixo de posições à frente no alfabeto, com base em um deslocamento.

No contexto da disciplina de Fundamentos de Sistemas Digitais, da Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS), foi proposta a implementação de um algoritmo decodificador da Cifra de César em linguagem Assembly. A decodificação é realizada utilizando o deslocamento calculado a partir da matrícula e do dígito verificador do aluno. O objetivo deste trabalho é detalhar o processo de implementação da decodificação da Cifra de César em Assembly, destacando as técnicas e comandos utilizados, e mostrar os resultados obtidos a partir da execução do código.

Este relatório é estruturado da seguinte forma: inicialmente, é apresentada a decodificação da frase de cada um dos alunos envolvidos na realização deste trabalho. Em seguida, apresenta-se o algoritmo realizado em linguagem Java para solucionar a problemática em alto nível. Posteriormente, apresenta-se o algoritmo realizado em linguagem Assembly, aplicando conceitos de manipulação de dados em baixo nível e operações aritméticas, sendo esse o foco principal do trabalho. Por fim, são mostradas a área de memória com o texto cifrado antes e decifrado depois da execução do código no MARS e a área de memória com o texto sendo decifrado durante a execução do código no ModelSim.

---

\*thomaz.abrantes@edu.pucrs.br

†pedro.soares003@edu.pucrs.br

## A decodificação da frase

Para cada aluno da disciplina de Fundamentos de Sistemas Digitais foi concedida uma frase codificada com base em um deslocamento, que utiliza a matrícula e o dígito verificador de cada um, feito através do seguinte cálculo:

$$\text{Deslocamento} = (\text{Matricula} / \text{Digito} + 1) \bmod 26$$

Partindo desse cenário, a primeira coisa a ser feita foi verificar a área de dados fornecida pelo professor e buscar pelas frases (em hexadecimal) dos alunos envolvidos na realização deste trabalho. Sendo assim, encontrou-se as frases:

**Frase do aluno Thomaz:** 0x6c, 0x6a, 0x62, 0x72, 0x7a, 0x6c, 0x6a, 0x6d, 0x66, 0x69, 0x78, 0x61, 0x6c, 0x6f, 0x62, 0x70, 0x71, 0x78, 0x70, 0x62, 0x6a, 0x6d, 0x6f, 0x62, 0x61, 0x66, 0x77, 0x62, 0x6b, 0x61, 0x6c, 0x6e, 0x72, 0x62, 0x6a, 0x62, 0x72, 0x7a, 0x6c, 0x61, 0x66, 0x64, 0x6c, 0x62, 0x70, 0x71, 0x78, 0x79, 0x78, 0x66, 0x75, 0x78, 0x6b, 0x61, 0x6c, 0x6c, 0x6b, 0x66, 0x73, 0x62, 0x69

**Frase do aluno Pedro:** 0x64, 0x71, 0x62, 0x65, 0x77, 0x79, 0x73, 0x71, 0x72, 0x79, 0x64, 0x71, 0x68, 0x79, 0x71, 0x71, 0x76, 0x71, 0x63, 0x79, 0x62, 0x79, 0x71, 0x75, 0x6a, 0x6b, 0x74, 0x65, 0x65, 0x6b, 0x64, 0x71, 0x74, 0x71, 0x62, 0x79, 0x6a, 0x75, 0x68, 0x71, 0x62, 0x63, 0x75, 0x64, 0x6a, 0x75

## Conversão em caracteres

Após analisar as frases em hexadecimal, foi preciso buscar na tabela ASCII a tradução em caracteres das frases em hexadecimal. Abaixo está representada a tabela ASCII utilizada pelos alunos para conseguir realizar essa conversão.

Hexadecimal	Decimal	Caractere
61	97	a
62	98	b
63	99	c
64	100	d
65	101	e
66	102	f
67	103	g
68	104	h
69	105	i
6a	106	j
6b	107	k
6c	108	l
6d	109	m
6e	110	n
6f	111	o
70	112	p
71	113	q
72	114	r
73	115	s
74	116	t
75	117	u
76	118	v
77	119	w
78	120	x
79	121	y
7a	122	z

Através da tabela, foi possível converter as frases dos alunos Thomaz e Pedro de hexadecimal para caractere:

**Frase do aluno Thomaz:** ljbzljmfixalobpqpjbmobafwbkalnrbjbrzlaflbpbqxyxfuxkallkfsbi

**Frase do aluno Pedro:** dqbewysqrydqhyqqvqcybyqujkteekdqtbbyjuhqbcudju

## Calculando o deslocamento

Tendo as frases dos alunos convertidas para caracteres, o próximo passo foi realizar o cálculo do deslocamento das letras com base na matrícula e no dígito verificador de cada aluno. Dessa forma, foram realizados os cálculos:

### Cálculo do deslocamento da frase do aluno Thomaz:

```
1 Deslocamento = (Matricula / Dígito + 1) mod 26
2 Deslocamento = (23102711 / 1 + 1) mod 26
3 Deslocamento = (23102711 / 2) mod 26
4 Deslocamento = 23
```

### Cálculo do deslocamento da frase do aluno Pedro:

```
1 Deslocamento = (Matricula / Dígito + 1) mod 26
2 Deslocamento = (23102625 / 3 + 1) mod 26
3 Deslocamento = (23102625 / 4) mod 26
4 Deslocamento = 16
```

## Realizando a decodificação

Na Cifra de César, um deslocamento significa que cada letra é substituída pela letra que está  $x$  posições adiante no alfabeto. Assim, sabendo que os deslocamentos das letras das frases dos alunos Thomaz e Pedro foram respectivamente 23 e 16, foi possível construir uma tabela de decodificação para descobrir a frase descryptografada de cada aluno. Dessa forma, tem-se:

### Tabela de decodificação do aluno Thomaz:

Alfabeto normal	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Alfabeto cifrado	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W

### Tabela de decodificação do aluno Pedro:

Alfabeto normal	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Alfabeto cifrado	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P

Levando em consideração ambas as tabelas, foi possível decodificar as frases de cada um dos estudantes, analisando a linha "Alfabeto cifrado" para cada uma das letras das frases cifradas e procurando sua letra correspondente no alfabeto normal.

Dessa forma temos:

**Frase decodificada do aluno Thomaz:**

omeucompiladorestasempredizendoquemeucodigoestabaixandoonivel

**Frase decodificada do aluno Pedro:**

nalogicabinariaafamiliaetudoounadaliteralmente

## O algoritmo em alto nível

Para realizar a lógica da Cifra de César em alto nível, foi desenvolvido um algoritmo em linguagem Java contendo um método para calcular o deslocamento de cada uma das letras das frases e outro método para descriptografar cada frase de cada um dos alunos. Assim, abaixo estão representados os métodos juntamente com suas explicações.

**Método para calcular o deslocamento:**

```
1 public static int calculaDeslocamento(int matricula, int digito) {
2     int deslocamento;
3     deslocamento = (matricula / (digito + 1)) % 26;
4     return deslocamento;
5 }
```

**Explicação:** Este método recebe por parâmetro a matrícula e o dígito verificador do aluno em questão e resolve o cálculo do deslocamento.  $\text{Deslocamento} = (\text{Matrícula} / \text{Dígito} + 1) \bmod 26$ .

**Método para descriptografar as frases:**

```
1 public static String descriptografar(String frase, int deslocamento) {
2     String fraseDescriptografada = "";
3     for (int i = 0; i < frase.length(); i++) {
4         char letra = frase.charAt(i);
5         int valor = (int) letra;
6         if ((valor - deslocamento) >= 0x61) {
7             valor -= deslocamento;
8             letra = (char) valor;
9             fraseDescriptografada = fraseDescriptografada + letra;
10        } else {
11            int numero = 0x1a - deslocamento;
12            valor += numero;
13            letra = (char) valor;
14            fraseDescriptografada = fraseDescriptografada + letra;
15        }
16    }
17    return fraseDescriptografada;
18 }
```

**Explicação:** Este método recebe por parâmetro uma frase do aluno em questão e o deslocamento anteriormente calculado. Percorre-se a frase com base em seu tamanho, pega-se o valor de cada um de seus caracteres (com base na tabela ASCII) e faz-se a verificação se o valor subtraído o deslocamento é maior ou igual a 0x61, que é o caractere 'a' em hexadecimal. Se essa premissa for verdadeira, o valor recebe ele mesmo subtraído o deslocamento, o caractere recebe o caractere correspondente ao

valor calculado e a frase descriptografada recebe ela mesma (inicialmente uma palavra vazia) somada ao caractere encontrado. Caso contrário, cria-se uma variável nova intitulada "numero", que recebe 0x1a (26 em hexadecimal) subtraído o deslocamento, além de o valor receber ele mesmo somado ao número, o caractere receber o caractere correspondente ao valor calculado e a frase descriptografada receber ela mesma somada ao caractere encontrado.

**Execução do algoritmo:** Ao executar-se o algoritmo passando por parâmetro as matrículas e os dígitos verificadores dos alunos Thomaz e Pedro no método "calculaDeslocamento", recebe-se como resultado os valores inteiros 23 e 16, respectivamente.

Ademais, ao executar-se o algoritmo passando por parâmetro as frases criptografadas (as mesmas apresentadas na seção "A decodificação da frase") e o deslocamento, anteriormente calculado pelo método "calculaDeslocamento", dos alunos Thomaz e Pedro no método "descriptografar", recebe-se respectivamente como resultado as frases:

- omeucompiladorestasemprepredizendoquemeucodigoestabaixandoonivel
- nalogicabinariaafamiliaetudoounadaliteralmente

Sendo assim, foi possível verificar que a execução do algoritmo em alto nível chega ao mesmo resultado encontrado na decodificação manual feita na seção "A decodificação da frase".

## O algoritmo em Assembly

Para realizar a lógica da Cifra de César em baixo nível, foi desenvolvido um algoritmo em linguagem Assembly contendo a mesma lógica utilizada no algoritmo em alto nível, sendo ela uma forma de calcular o deslocamento de cada uma das letras das frases e uma forma de descriptografar as frases de cada um dos alunos. O algoritmo foi feito usando o MARS, uma ferramenta educacional utilizada para o desenvolvimento e simulação de programas em Assembly MIPS.

De modo a auxiliar os alunos, o professor disponibilizou o uso de uma Função de Divisão em Assembly do MIPS para auxiliar no cálculo do deslocamento. A função está apresentada a seguir:

```
1  division:
2      lui    $t0, 0x8000          # mascara para isolar bit mais significativo
3      addi   $t1, $zero, 32       # contador de iteracoes
4      xor    $v0, $v0, $v0        # registrador P($v0)-A($v1) com 0 e o dividendo
5      add    $v1, $a1, $0
6  dloop:
7      and    $t2, $v1, $t0        # isola em t2 o bit mais significativo do
8          # registrador 'A' ($v1)
9      sll    $v0, $v0, 1          # desloca para a esquerda o registrado P-A
10     sll    $v1, $v1, 1
11     beq    $t2, $0, di1
12     ori    $v0, $v0, 1          # coloca 1 no bit menos significativo do
13         # registrador 'P' ($v0)
14  di1:
15     sub    $t2, $v0, $a0        # subtrai 'P' ($v0) do divisor ($a0)
16     blt    $t2, $0, di2
17     add    $v0, $t2, $0        # se a subtracao deu positiva, 'P' ($v0) recebe o
18         # valor da subtracao
19     ori    $v1, $v1, 1          # e 'A' ($v1) recebe 1 no bit menos significativo
20  di2:
21     addi   $t1, $t1, -1         # decrementa o numero de iteracoes
22     bne    $t1, $0, dloop
23     jr     $ra
```

Essa função em Assembly realiza a divisão serial de dois números armazenados nos registradores 'a1' (dividendo) e 'a0' (divisor). O resultado da divisão (quociente) é armazenado em 'v1', enquanto o resto da divisão é armazenado em 'v0'.

Abaixo é possível visualizar o código do algoritmo decifrador da Cifra de César em Assembly, juntamente de sua devida explicação de funcionamento.

```

1      main:
2          # Carregar os valores de matricula, dig_verif e size
3          la    $t0, matricula
4          lw    $s0, 0($t0)          # $s0 = matricula
5
6          la    $t1, dig_verif
7          lw    $s1, 0($t1)          # $s1 = dig_verif
8
9          la    $t2, size
10         lw    $s2, 0($t2)          # $s2 = size
11
12         la    $s3, encrypted
13
14         # Calcular o deslocamento
15         addi $s1, $s1, 1            # dig_verif + 1
16         add  $a1, $s0, $zero        # $a1 = matricula
17         add  $a0, $s1, $zero        # $a0 = digito + 1
18         jal  division              # Chamar funcao de divisao
19         add  $t3, $v1, $zero        # $t3 = resultado da divisao
20         addi $t4, $zero, 0x1a       # 26
21         add  $a1, $t3, $zero        # $a1 = resultado da divisao anterior
22         add  $a0, $t4, $zero        # a0 = 26
23         jal  division              # Chamar funcao de divisao
24         add  $t5, $v0, $zero        # $t5 = resultado do modulo
25
26         # Descriptografar frase
27         xor  $t6, $t6, $t6          # Indice inicial
28
29     loop:
30         beq  $t6, $s2, end          # Se o indice for igual ao tamanho, termina
31
32         lw   $t7, 0($s3)            # Carrega o byte encriptado atual
33         sub  $t7, $t7, $t5          # Subtrai o deslocamento
34
35         # Verificar se precisamos "dar a volta" no alfabeto
36         addi $t8, $zero, 0x61       # Caractere 'a'
37         bge  $t7, $t8, else         # Se o caractere e >= 'a', nao precisa dar a
38         volta
39
40         addi $t7, $t7, 0x1a         # Caso contrario, soma 26
41
42     else:
43         sw   $t7, 0($s3)            # Armazena o caractere descriptografado
44         addi $s3, $s3, 4            # Avanca para o proximo
45         addi $t6, $t6, 1            # Incrementa o indice
46         j    loop                  # Repete o loop
47
48     end:
49         j    end                    # Acabou

```

**Explicação:** O algoritmo começa carregando valores importantes, como a matrícula, o dígito verificador e o tamanho da frase cifrada.

Após isso, usa a fórmula (matrícula / (dígito verificador + 1)) mod 26 para calcular o deslocamento usado na Cifra de César. Para realizar o cálculo, primeiro faz-se uma adição imediata de 1 no dígito verificador, depois adiciona-se ao registrador "a1" a matrícula, adiciona-se ao registrador "a0" o dígito + 1 e chama-se a função de divisão. Em seguida, adiciona-se a "t3" o resultado da divisão. Posteriormente, adiciona-se ao registrador "a1" o resultado da divisão anterior, adiciona-se ao registrador "a0" o número 26 e chama-se a função de divisão novamente, gravando o resultado dessa nova divisão (módulo) em "t5".

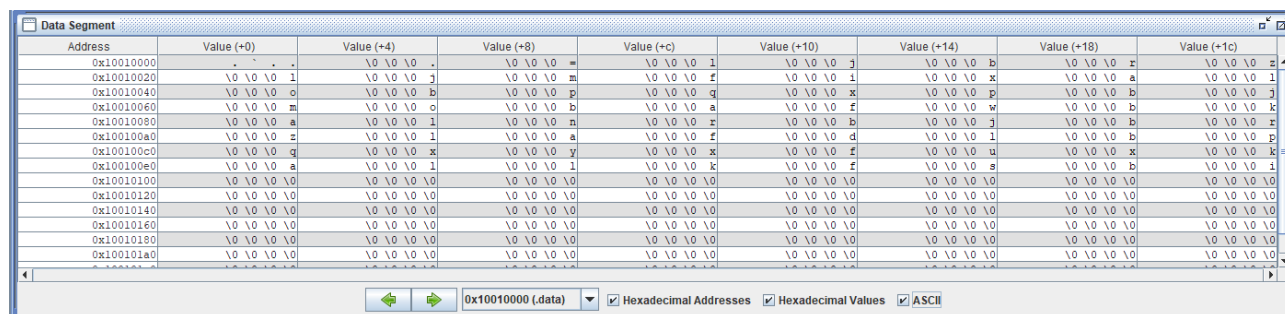
Seguidamente, para cada caractere na mensagem cifrada, subtrai-se o deslocamento calculado para recuperar a letra original. Verifica-se se é necessário "dar a volta" no alfabeto (por exemplo, se subtrair o deslocamento faz com que a letra caia antes de 'a'). Se sim, ajusta-se a letra para se manter dentro do alfabeto.

Cada caractere descriptografado é armazenado de volta na memória. O processo continua até que todos os caracteres da mensagem cifrada sejam descriptografados e armazenados.

## A área de memória no MARS

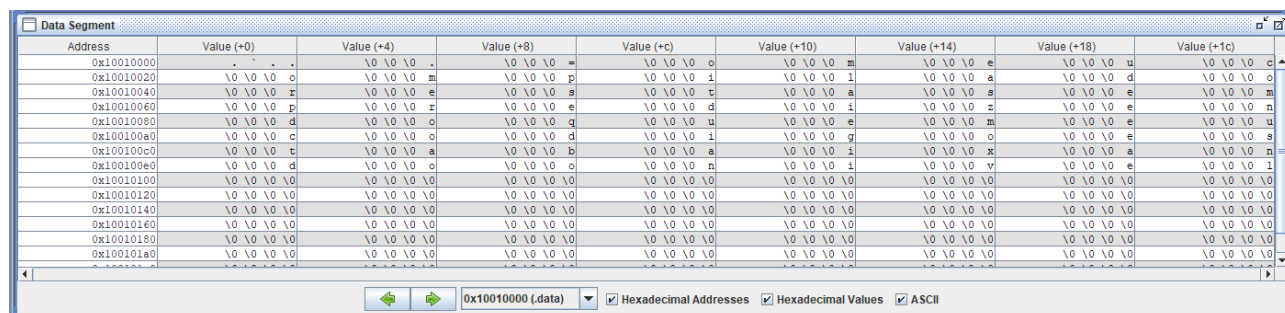
Após a finalização do código, foi possível realizar a execução do mesmo no MARS. Abaixo estão as áreas de memória do MARS antes e depois da execução do código para ambos os alunos.

### Área de memória do MARS - aluno Thomaz:



Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	.	.	.	.	.	.	.	.
0x10010020	\0 \0 \0 1	\0 \0 \0 j	\0 \0 \0 m	\0 \0 \0 1	\0 \0 \0 j	\0 \0 \0 b	\0 \0 \0 r	\0 \0 \0 z
0x10010040	\0 \0 \0 o	\0 \0 \0 b	\0 \0 \0 p	\0 \0 \0 q	\0 \0 \0 x	\0 \0 \0 p	\0 \0 \0 b	\0 \0 \0 j
0x10010060	\0 \0 \0 m	\0 \0 \0 o	\0 \0 \0 b	\0 \0 \0 a	\0 \0 \0 f	\0 \0 \0 w	\0 \0 \0 b	\0 \0 \0 k
0x10010080	\0 \0 \0 a	\0 \0 \0 1	\0 \0 \0 m	\0 \0 \0 f	\0 \0 \0 b	\0 \0 \0 j	\0 \0 \0 b	\0 \0 \0 s
0x100100a0	\0 \0 \0 z	\0 \0 \0 1	\0 \0 \0 a	\0 \0 \0 f	\0 \0 \0 d	\0 \0 \0 1	\0 \0 \0 b	\0 \0 \0 p
0x100100c0	\0 \0 \0 q	\0 \0 \0 x	\0 \0 \0 y	\0 \0 \0 x	\0 \0 \0 f	\0 \0 \0 u	\0 \0 \0 x	\0 \0 \0 k
0x100100e0	\0 \0 \0 a	\0 \0 \0 1	\0 \0 \0 1	\0 \0 \0 k	\0 \0 \0 f	\0 \0 \0 s	\0 \0 \0 b	\0 \0 \0 1
0x10010100	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010120	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010140	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010160	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010180	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x100101a0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0

Figura 1: Captura de tela da área de memória com o texto cifrado do aluno Thomaz antes da execução do código no MARS.



Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	.	.	.	.	.	.	.	.
0x10010020	\0 \0 \0 o	\0 \0 \0 m	\0 \0 \0 p	\0 \0 \0 i	\0 \0 \0 1	\0 \0 \0 a	\0 \0 \0 d	\0 \0 \0 o
0x10010040	\0 \0 \0 r	\0 \0 \0 e	\0 \0 \0 s	\0 \0 \0 t	\0 \0 \0 a	\0 \0 \0 s	\0 \0 \0 e	\0 \0 \0 m
0x10010060	\0 \0 \0 p	\0 \0 \0 z	\0 \0 \0 e	\0 \0 \0 d	\0 \0 \0 i	\0 \0 \0 z	\0 \0 \0 e	\0 \0 \0 n
0x10010080	\0 \0 \0 d	\0 \0 \0 o	\0 \0 \0 q	\0 \0 \0 u	\0 \0 \0 e	\0 \0 \0 m	\0 \0 \0 e	\0 \0 \0 s
0x100100a0	\0 \0 \0 c	\0 \0 \0 o	\0 \0 \0 b	\0 \0 \0 i	\0 \0 \0 o	\0 \0 \0 o	\0 \0 \0 e	\0 \0 \0 u
0x100100c0	\0 \0 \0 t	\0 \0 \0 a	\0 \0 \0 b	\0 \0 \0 a	\0 \0 \0 i	\0 \0 \0 x	\0 \0 \0 a	\0 \0 \0 n
0x100100e0	\0 \0 \0 d	\0 \0 \0 o	\0 \0 \0 o	\0 \0 \0 n	\0 \0 \0 i	\0 \0 \0 v	\0 \0 \0 e	\0 \0 \0 1
0x10010100	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010120	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010140	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010160	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010180	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x100101a0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0

Figura 2: Captura de tela da área de memória com o texto decifrado do aluno Thomaz depois da execução do código no MARS.

### Área de memória do MARS - aluno Pedro:

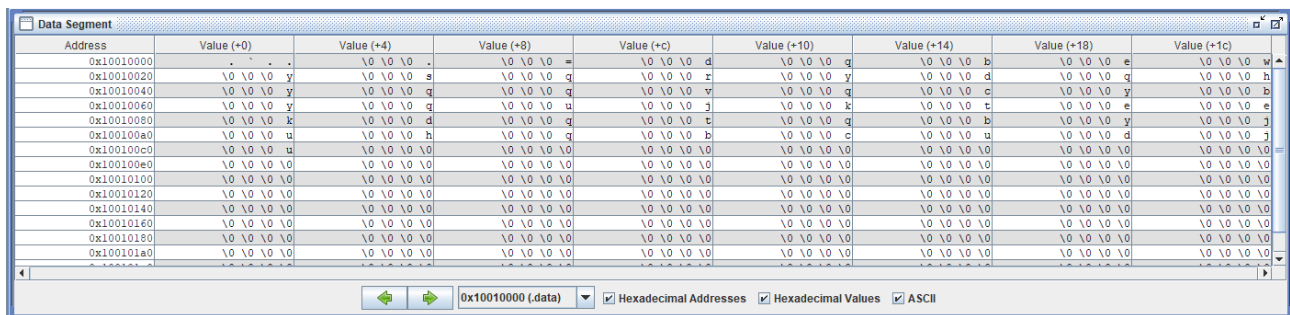


Figura 3: Captura de tela da área de memória com o texto cifrado do aluno Pedro antes da execução do código no MARS.

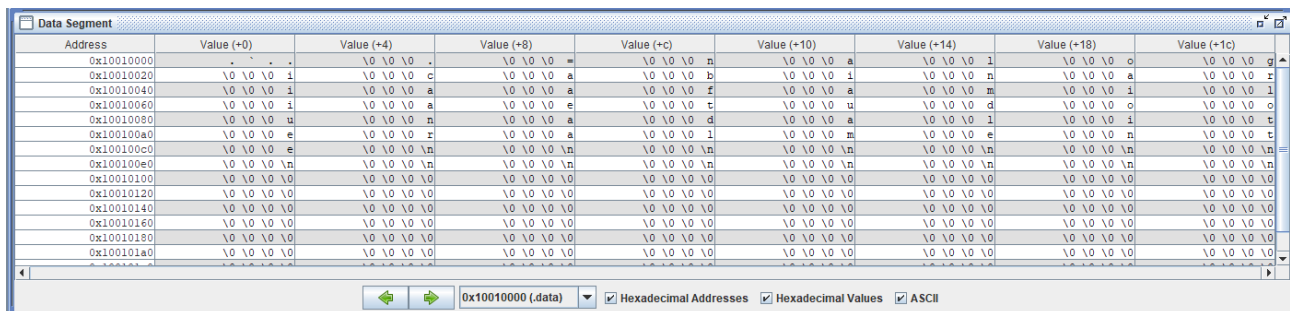


Figura 4: Captura de tela da área de memória com o texto decifrado do aluno Pedro depois da execução do código no MARS.

## A área de memória no ModelSim

Assim como é possível executar o código no MARS, também é possível executá-lo no ModelSim. Dessa forma, o professor disponibilizou uma pasta contendo todos os arquivos necessários para que a simulação pudesse ser realizada, com exceção do arquivo "mips.txt", que foi deixado para os estudantes implementarem através de um "Dump Memory", que é funcionalidade do MARS que permite ao usuário visualizar o conteúdo da memória do simulador MIPS.

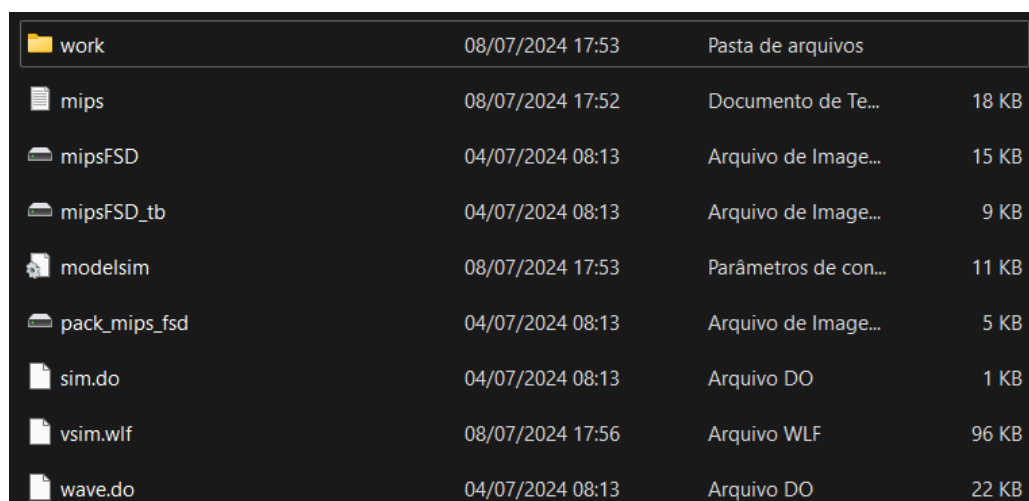


Figura 5: Arquivos usados na execução do código no Modelsim.

Assim, abaixo podem ser vistas as áreas de memória com o texto sendo decifrado durante a execução do código no Modelsim dos alunos.



## Área de memória do ModelSim - aluno Thomaz:

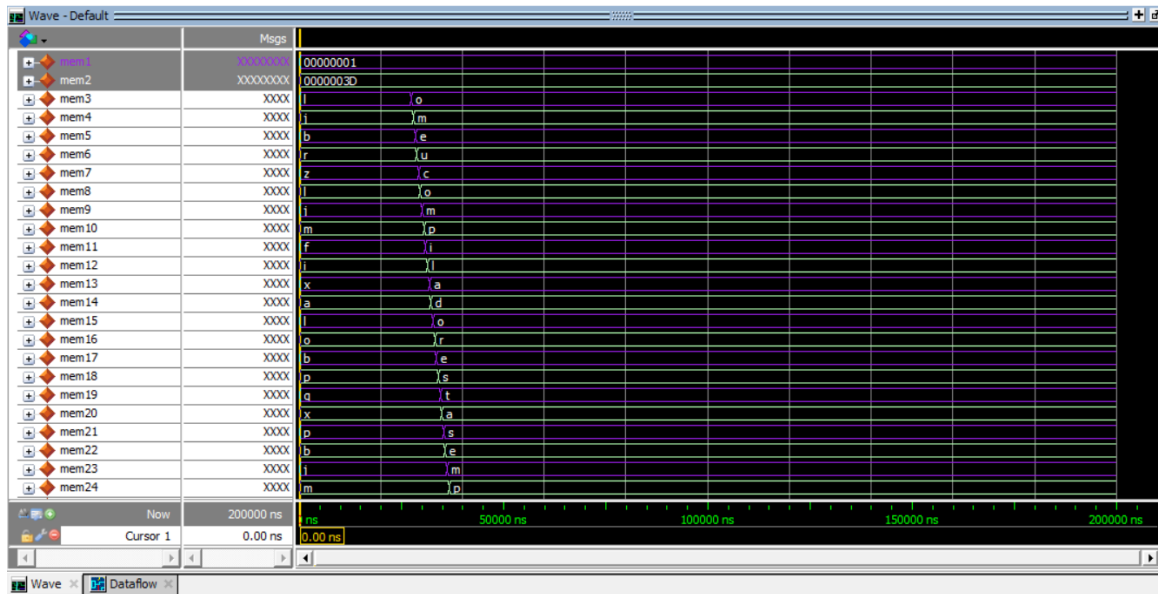


Figura 6: Área de memória com o texto do aluno Thomaz sendo decifrado durante a execução do código no Modelsim.

## Área de memória do ModelSim - aluno Pedro:

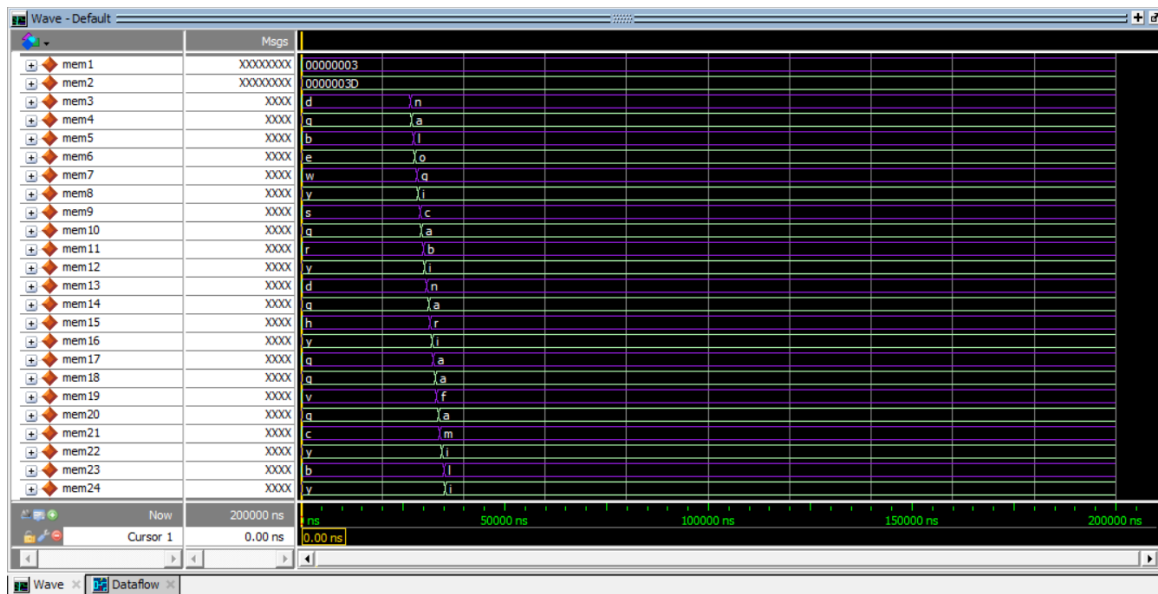


Figura 7: Área de memória com o texto do aluno Pedro sendo decifrado durante a execução do código no Modelsim.

## Conclusões

Através da execução do código em Assembly no MARS e da simulação do algoritmo feita no ModelSim, foi possível verificar que o objetivo do trabalho foi alcançado, uma vez que dadas duas frases encriptadas, uma para cada aluno, conseguiu-se realizar a deciptação de cada uma delas, como pode

ser visto nas imagens em ambas as seções, "Área de memória do MARS" e "Área de memória do ModelSim", onde é possível visualizar o conteúdo encriptado e o conteúdo decriptado após a execução do código. Por conseguinte, pode-se observar que os resultados obtidos coincidem com os resultados encontrados na seção "A decodificação da frase" e com os resultados encontrados a partir do algoritmo em alto nível.

Em pesquisas futuras, espera-se aprimorar o código, de modo a melhorar sua sintaxe, mantendo-se a mesma lógica implementada. Ademais, espera-se testar o algoritmo utilizando-se novas frases com novas matrículas e dígitos verificadores e por consequência novos cálculos de deslocamento.