Departamento de Ciência da Computação
Organização e Arquitetura de Computadores

1º Trabalho

CIC0099
Professor: Ricardo Pezzuol Jacobi.
Discente: Adriano Ulrich do Prado Wiedmann – 202014824 (Turma 03)
Semestre: 2023/2

1. Objetivo

Este trabalho consiste em traduzir um código C e implementar em assembler do *RISCV*, utilizando o ambiente *RARS* na versão 1.6. O código C descreve parte do algoritmo de cifragem de dados *IDEA*.

2. Vetores

Os vetores *blk_in*, *blk_out* e *keys* foram criados no *.data*. Os vetores são *.half* e foram inicializados com os valores conforme descrito no código em C. Figura 1.

```
1   .data
2
3   #vetores de 2 bytes
4   blk_in:
5     .half 0         #Primeiro elemento (16 bits)
6     .half 1         #Segundo elemento (16 bits)
7     .half 2         #Terceiro elemento (16 bits)
8     .half 3         #Quarto elemento (16 bits)
9
10  blk_out:
11    .half 0
12    .half 0
13    .half 0
14    .half 0
15
16  keys:
17    .half 1
18    .half 2
19    .half 3
20    .half 4
21    .half 5
22    .half 6
```

Figura 1. Vetores.

3. Funções

a) *main*: A função *main* não apenas chama a função *ideia_round*, mas também imprime os valores do vetor *blk_out* por meio de um *loop*. Por fim, encerra o programa.

b) *ideia_round*: A função *ideia_round* realiza todas as operações necessárias com a ajuda de duas outras funções, *func_mul* e *lsw16*. Além disso, carrega os valores do vetor *blk_in* nos registradores de *t1* até *t4* e os valores do vetor *keys* nos registradores de *s1* até *s6*. Antes de chamar as outras funções, o

endereço de retorno para a função ***main*** é salvo na pilha. Em seguida, os registradores ***a1*** e ***a2*** são passados como parâmetros para a função ***func_mul***, com os valores dos vetores ***blk_in*** e ***keys***, respectivamente. Em relação ao código em C, ***a1 = x*** e ***a2 = y***. Para a função ***lsw16***, é passado como parâmetro o registrador ***a1***, que é o resultado da soma dos valores dos vetores ***blk_in*** e ***keys***, respectivamente. Em comparação com o código em C, essa soma seria, por exemplo, *word2 + *key_ptr++.*

.

c) ***func_mul:*** A função ***func_mul*** é semelhante à função ***mul*** no código em C, com a adição das funções ***shiftleft16*** e ***increment***. No laço condicional, se ***a3*** for diferente de 0, realiza-se a subtração de ***a1*** de ***a4*** e, em seguida, subtrai-se 65537 do resultado de ***a4 - a1***. Além disso, a função ***shiftleft16*** é chamada se ***a4*** for menor que ***a1*** (ou seja, se ***a1*** for maior que 65535). Veja a Figura 2 para referência.

d) ***shiftleft16***: Caso o valor do registrador ***a4*** for maior que 65535 faz-se um deslocamento à esquerda de 16 bits em um valor contido no registrador ***a1***. Figura 3.

e) ***increment***: Conforme o código em C, se x < y (a2 < a1) é realizado um incremento de 65537 em x (a1). Figura 4.

f) ***lsw16***: A função executa uma operação de máscara de bits em um valor contido no registrador ***a1***, preservando apenas os 16 bits menos significativos. Figura 5.

```
155 func_mul:
156     #p = a3; x = a1; y = a2
157     #p = x*y
158     mul a3, a1, a2
159
160     #se a3 diferente de zero, vai para else
161     bne a3, x0, else
162     #if (p == 0)
163     if:
164         #x = 65537-x-y;
165         li a4, 65537
166         sub a4, a4, a1  #65537-x
167         sub a1, a4, a2  #(65537-x)-y
168
169         #verifica se a1 é maior que 65535(maior valor para 16 bits)
170         li a4, 65535
171         bltu a4, a1, shiftleft16
172
173         ret
174     else:
175         li a4, 16       #x = p >> 16;
176         srl a1, a3, a4  #operação de deslocamento à direita (right shift)
177         mv a2, a3
178         sub a1, a2, a1
179
180         # se x < y (a2 < a1) vai para função increment
181         bltu a2, a1, increment
182
183         ret
```

Figura 2. Função func_mul.

```
190 #Função que desloca 16 bits para esquerda
191 shiftleft16:
192     li a4, 16           #Quantidade de bits a serem deslocados (16 bits)
193     sll a1, a1, a4
194     ret
```

Figura 3. Função shiftleft16.

```
185 # x += 65537;
186 increment:
187     li a4, 65537
188     add a1, a1, a4
189
```

Figura 4. Função increment.

```
196 #operação de máscaramento
197 lsw16:
198     li a2, 0x0000ffff
199     and a1, a1, a2
200     ret
```

Figura 5. Função lsw16.

4. Testes

O primeiro teste foi realizado com valores já disponibilizados no código em C. O lado direito representa a saída dos valores do código original e à esquerda são os valores de saída do código em Assembly, como pode ser visto na Figura 6.

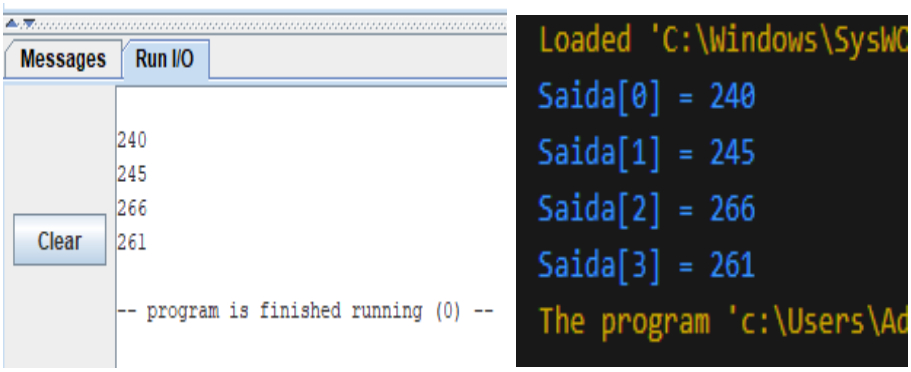

Figura 6. Primeiro teste.

Para o segundo teste, é inicializado com alguns números negativos nos vetores. *blk_in[] = {0, -1, -2, 3}* e *keys[] = {-1, -2, 3, 4, 5, 6}*. A saída pode ser visualizada na Figura 7.

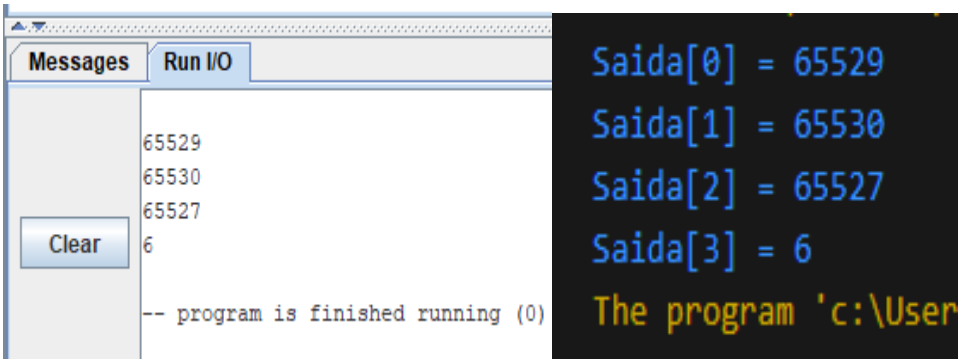

Figura 7. Segundo teste.

No terceiro teste, os vetores são inicializados com valores maiores, números que estão na dezena. Nesse caso: *blk_in[] = {0, 11, 42, 33}* e *keys[] = {13, 12, 23, 24, 35, 16}*. O resultado pode ser visualizado na Figura 8.

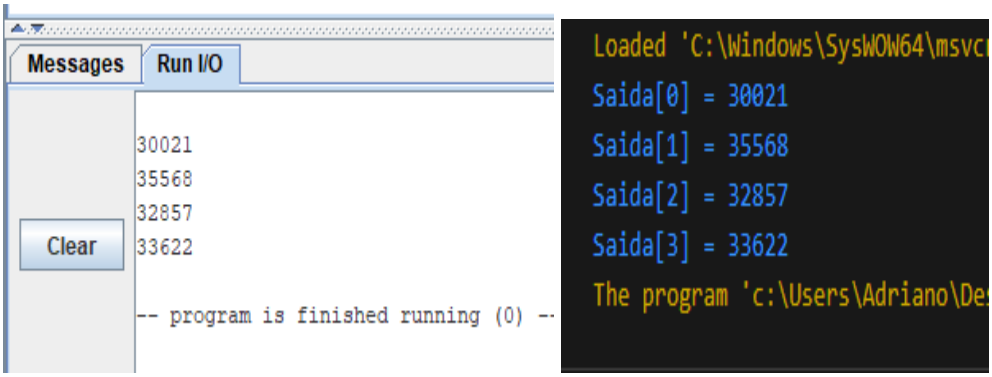

Figura 8. Terceiro teste.