Marmara University Faculty of Engineering

CSE3038- Computer Organization

150120991 Lale Hüseyin

150120852 Zekeriya Çedikçi

150118006 Talha Osman Saraç

- **Question 1.**

  The program prompts the user to input a string, and then reads the input string into memory. It then prepares the string for the Manacher's algorithm by converting all letters to lowercase and adding a null terminator at the end.

  The Manacher's algorithm subroutine is then called, which uses a combination of load and store instructions and control flow instructions to calculate the longest palindrome substring in the processed string. The algorithm maintains a center position and a right boundary while expanding a palindrome around the center position. The palindrome radius at each position is stored in an array, and the algorithm uses this array to calculate the minimum palindrome radius at the current position. The algorithm also keeps track of the maximum palindrome radius found so far and the center position of the longest palindrome substring.

  Once the longest palindrome substring is found, the program prints it and its length to the console using load and store instructions and the system call function.

- **Question 2**

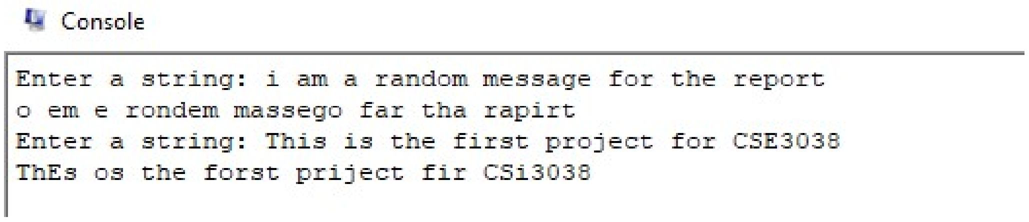
  This program takes an input string from the user and then finds all the vowels in the string. It stores the vowels in a stack, and then replaces all the vowels in the input string with the vowels from the stack in reverse order.

  The program starts by prompting the user to enter a string. Then, it reads the input string and initializes some variables to keep track of the vowels and the stack.

  The program then uses a loop to go through the input string one letter at a time. If the letter is a vowel, it pushes it onto the stack. Once it has gone through the entire string, it resets the pointer to the start of the string and then starts another loop to replace the vowels with the ones from the stack. If the letter is a vowel, it pops the top letter off the stack and replaces the original vowel with the popped letter. Finally, the program prints the resulting string and exits.

  The program also includes a function called "is_vowel" which checks if a given letter is a vowel. It checks if the letter is equal to any of the five lowercase and five uppercase vowels. If the letter is a vowel, it returns a value of 1, and if it is not, it returns a value of 0.

  Overall, this program is a good example of how to use loops and stacks in assembly language. However, it can be difficult to understand if you are not familiar with assembly language or programming in general.



```
Enter a string: i am a random message for the report
o em e rondem massego far tha rapirt
Enter a string: This is the first project for CSE3038
ThEs os the forst priject fir CSi3038
```

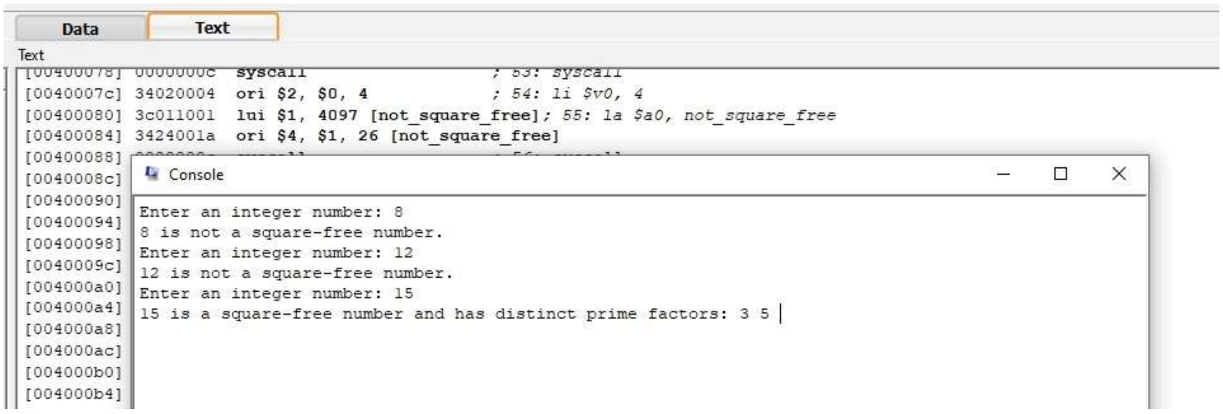
- **Question 3**

This MIPS assembly code checks if an input integer is square-free, meaning it has no repeated prime factors. It prompts the user to enter an integer, reads the input, and then checks for distinct prime factors using a counter that starts from 2 and increments until it reaches the input integer.

In the check_prime loop, the code first checks if the counter has reached the input integer, and if so, it jumps to the print_result section to print the message that the input integer is square-free and has distinct prime factors. If the counter is less than the input integer, the code checks if the current number is divisible by the counter. If it is, it divides the current number by the counter and checks if it is still divisible by the counter. If it is not divisible, the code increments the count of distinct prime factors.

If the input integer is not square-free, the code jumps to the not_square_free_result section to print the message that the input integer is not square-free. If the input integer is square-free, the code prints the message that it is square-free and has distinct prime factors, and then prints each of the distinct prime factors using the print_factors loop.

In the print_factors loop, the code checks if the counter has reached the input integer, and if so, it exits the loop. Otherwise, it checks if the current number is a factor of the input integer. If it is, the code prints the factor and a space, divides the input integer by the factor, and loops again. If it is not a factor, the code increments the counter and loops again.



```
[00400078] 0000000c  syscall                    ; 53: syscall
[0040007c] 34020004  ori $2, $0, 4              ; 54: li $v0, 4
[00400080] 3c011001  lui $1, 4097 [not_square_free]; 55: la $a0, not_square_free
[00400084] 3424001a  ori $4, $1, 26 [not_square_free]
[00400088]
[0040008c]
[00400090]
[00400094]
[00400098]
[0040009c]
[004000a0]
[004000a4]
[004000a8]
[004000ac]
[004000b0]
[004000b4]
```

```
Enter an integer number: 8
8 is not a square-free number.
Enter an integer number: 12
12 is not a square-free number.
Enter an integer number: 15
15 is a square-free number and has distinct prime factors: 3 5
```

- **Question 4**

The code first prints a message asking the user to enter the size of the matrix, then reads in n and m from the user. It then allocates memory for the matrix using the system call 9 and stores the address of the allocated memory in $s0.

Next, the code enters a loop to read in the matrix elements from the user. It uses two nested loops, one to iterate over the rows and the other to iterate over the columns. The outer loop iterates from 0 to n-1, and the inner loop iterates from 0 to m-1 for each row. The loop maintains a counter $t3 that keeps track of the total number of elements that have been read so far, and a pointer $t2 that points to the current element being read in the matrix.

For each element, the code prompts the user to enter an element of the matrix using the message stored in input_element_message. It then reads in the element from the user using system call 5 and stores it in $t4. The code then checks if the element is distinct from the elements that have already been read in the matrix. It does this by iterating over the elements that have been read so far and comparing them to the current element. If the current element is not distinct, the code prompts the user to enter a different element using the message stored in not_distinct_message and repeats the loop for the current element.

If the current element is distinct, the code stores it in the current position of the matrix pointed to by $t2 and increments $t2 to point to the next position in the matrix. It then increments $t3 to count the current element as having been read and repeats the loop for the next column.

Once all the elements of the matrix have been read and stored, the code exits the loop, and the program is finished.

Overall, this code is a basic implementation of inputting a matrix from a user in MIPS assembly language. It uses nested loops to iterate over the matrix elements and stores them in a one-dimensional array using row-major order. The code also checks for distinct elements and prompts the user to enter a different element if necessary.

- **Menu**

The given code is a program that presents the user with a menu of five options. Each option performs a different task.

The first option***, Find Palindrome***, prompts the user to input a string and then finds the longest palindrome in the string.

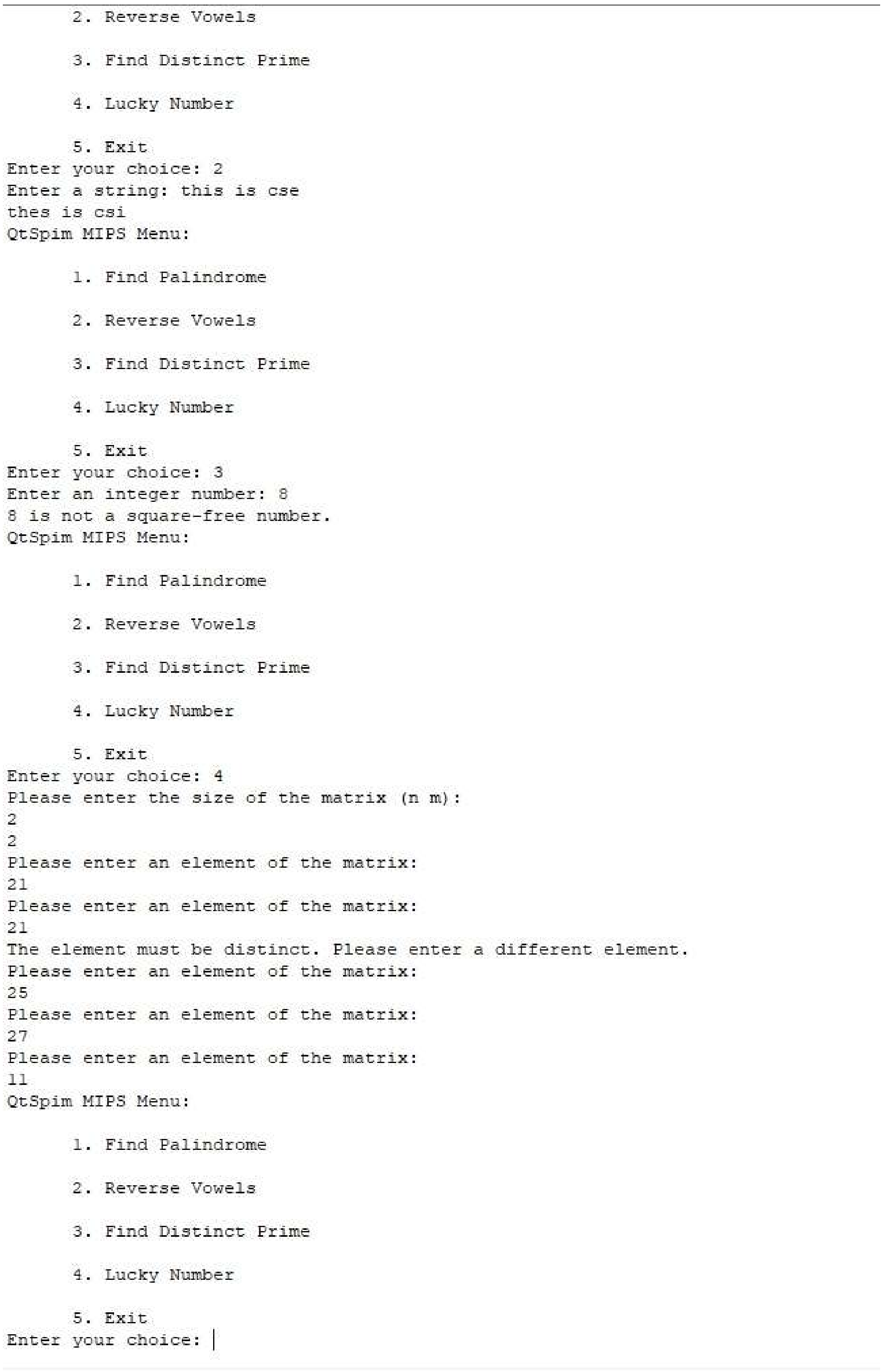
The second option***, Reverse Vowels***, asks the user to input a string and then reverses the order of all the vowels in the string.

The third option, ***Find Distinct Prime***, asks the user to input an integer and then checks if it has distinct prime factors.

The fourth option, ***Lucky Number***, asks the user to input the size of a matrix and then prompts the user to input the elements of the matrix. The program then checks whether there is a number that appears only once in the matrix.

The last option, ***Exit***, simply exits the program.

However, there are some areas where the code could be improved. For example, some variable names could be more descriptive, and error handling could be added to handle cases where the user inputs invalid data. The code for finding distinct prime factors could also be simplified.



```
        2. Reverse Vowels

        3. Find Distinct Prime

        4. Lucky Number

        5. Exit
Enter your choice: 2
Enter a string: this is cse
thes is csi
QtSpim MIPS Menu:

        1. Find Palindrome

        2. Reverse Vowels

        3. Find Distinct Prime

        4. Lucky Number

        5. Exit
Enter your choice: 3
Enter an integer number: 8
8 is not a square-free number.
QtSpim MIPS Menu:

        1. Find Palindrome

        2. Reverse Vowels

        3. Find Distinct Prime

        4. Lucky Number

        5. Exit
Enter your choice: 4
Please enter the size of the matrix (n m):
2
2
Please enter an element of the matrix:
21
Please enter an element of the matrix:
21
The element must be distinct. Please enter a different element.
Please enter an element of the matrix:
25
Please enter an element of the matrix:
27
Please enter an element of the matrix:
11
QtSpim MIPS Menu:

        1. Find Palindrome

        2. Reverse Vowels

        3. Find Distinct Prime

        4. Lucky Number

        5. Exit
Enter your choice: |
```