

Marmara University
Faculty of Engineering



CSE3038
Computer Organization

Project - 1

Instructor: Haluk Rahmi Topçuoğlu

Date: 03.04.2024

	Department	Student Id Number	Name & Surname
1	CSE	150121076	Abdullah KAN
2	CSE	150120055	Muhammed Talha KARAGÜL
3	CSE	150121520	Ensar Muhammet YOZGAT
4	CSE	150120020	Mustafa Said ÇANAK

Question 1:.....	2
Question 2:.....	3
Question 3:.....	3

Question 1:

Implementation Details:

The program begins by prompting the user to enter coefficients for the sequence via the coefficientPrompt message. It reads and stores the coefficients in registers \$s0 and \$s1. Then, it prompts the user to enter the first two numbers of the sequence via the twoNumberPrompt message. It reads and stores these two numbers in registers \$s2 and \$s3.

The program prompts the user to enter the number of the element they want to calculate in the sequence via the calculatePrompt message. It reads and stores this input in register \$s4. It checks if the input for the calculation is less than 1. If so, it displays an error message (invalidInputMessage) and terminates the program.

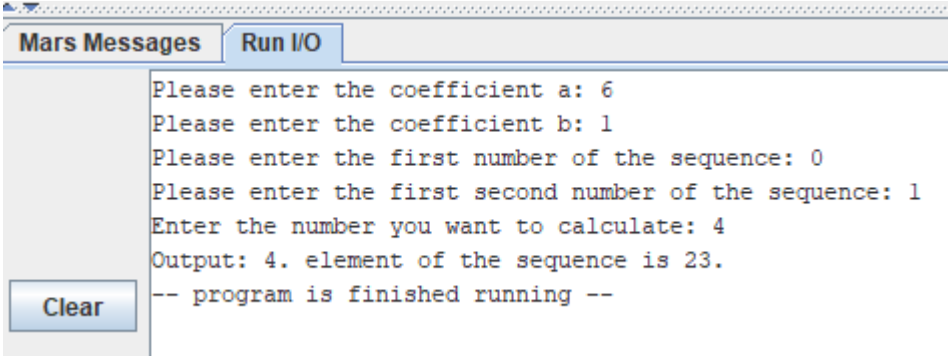
The program enters a loop starting at loopStart. It initializes \$t0 (counter) to 3 and \$t1 to -2. \$t1 serves as a temporary storage for the calculated result. It then jumps to loopCondition to check if the loop should continue based on whether the counter \$t0 is less than the calculation amount \$s4. Inside the loop, it calculates the next element of the sequence using the formula provided: $\text{result} = a * f1 + b * f0 - 2$. It updates the values of \$s2 (f0) and \$s3 (f1) accordingly to maintain the sequence. The loop continues until the calculated amount is reached. After the loop, the program displays the calculated element of the sequence using the output1, output2, and output3 messages.

The program prints out the result of the calculation using the output1, output2, and output3 messages. If the input for the calculation is less than 1, the program displays an error message and terminates.

Registers such as \$s0, \$s1, \$s2, \$s3, \$s4, and \$s5 are used to store coefficients, sequence values, and the calculation amount. Temporary registers like \$t0, \$t1, \$t2, \$t3, and \$t4 are used for loop control and temporary storage during calculations. The program utilizes basic arithmetic operations such as addition, multiplication, and comparisons to compute the sequence element. It also uses branching instructions to control the flow of execution within the loop and to handle invalid inputs.

Outputs:

```
Please enter the coefficient a: 2
Please enter the coefficient b: 1
Please enter the first number of the sequence: 1
Please enter the first second number of the sequence: 2
Enter the number you want to calculate: 4
Output: 4. element of the sequence is 6.
-- program is finished running --
```



The screenshot shows a window titled "Mars Messages" with a "Run I/O" button. The output text is as follows:

```
Please enter the coefficient a: 6
Please enter the coefficient b: 1
Please enter the first number of the sequence: 0
Please enter the first second number of the sequence: 1
Enter the number you want to calculate: 4
Output: 4. element of the sequence is 23.
-- program is finished running --
```

A "Clear" button is visible on the left side of the window.

Question 2:

Implementation Details:

The program prompts the user to enter integers via the prompt message. It reads the user input string into a buffer (input_buf) using syscall 8.

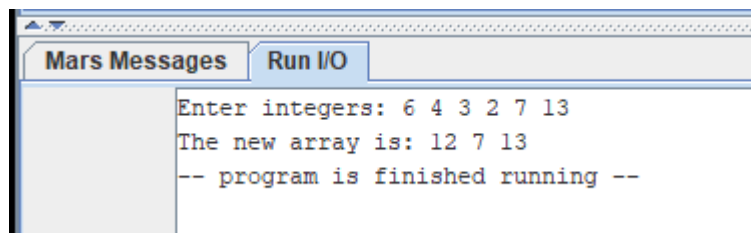
The program iterates through the input buffer, converting ASCII characters to integers and storing them in the inputArray. It keeps track of the length of the integer and checks if a space character is encountered to determine the end of each integer.

The program enters a loop (check) where it iterates through the inputArray. For each pair of adjacent integers, it calls find_gcd and find_lcm functions. The find_gcd function computes the GCD of two integers and returns the result. The find_lcm function computes the LCM of two integers and returns the result.

The computed GCD and LCM are stored in the outputArray. Once all calculations are done, the program prints the elements of the outputArray separated by spaces. The find_gcd function takes two arguments a and b, loads them from memory, and computes their GCD using the Euclidean algorithm. It returns the computed GCD in register \$v0. The find_lcm function takes two arguments a and b, loads them from memory, and computes their LCM using the formula $LCM(a,b) = (a * b) / GCD(a,b)$. It returns the computed LCM in register \$v0.

After computing the GCD and LCM for all pairs, the program iterates through the outputArray and prints each element followed by a space character. This program effectively utilizes register \$s0 to \$s4 to manage addresses, integer values, and loop counters. Additionally, it makes use of the stack for storing return addresses during function calls. The control flow is managed through branching and jumping instructions (beq, j, jal) to navigate between different sections of the program.

Outputs:

A screenshot of a terminal window with a title bar. The title bar contains two tabs: 'Mars Messages' and 'Run I/O'. The terminal displays the following text:

```
Enter integers: 6 4 3 2 7 13
The new array is: 12 7 13
-- program is finished running --
```

Question 3:

Implementation Details:

The program prompts the user to input a string using Input_prompt. It reads the input string into the buffer Input_string. The program calculates the length of the input string using the strlen function. The length is stored in register \$t0. The program prompts the user to input the shuffle length. It reads the input into register \$t1.

The program calls the shuffle function passing the length of the input string (\$t0) and the shuffle length (\$t1). The shuffle function recursively shuffles the characters in the input string. It swaps characters by traversing from both ends towards the middle. After shuffling, the program prints the shuffled string by using syscall 4 with the address of Input_string. Finally, the program exits using syscall 10.

The shuffle function swaps characters recursively. It calculates the mid-point of the string and calls swap to swap characters from both ends towards the middle until the entire string is shuffled. It maintains the recursion by decrementing the shuffle length with each call until it becomes zero, indicating the end of the shuffling process.

The swap function swaps characters in the input string. It uses two pointers to traverse the string from both ends towards the middle, swapping characters at each iteration until they meet in the middle.

The strlen function calculates the length of a null-terminated string. It iterates through the string, counting each character until it encounters the null terminator. The length is then stored in register \$t0 and returned.

Outputs:

```
Input: Computer
3
Output:
retupmoC
-- program is finished running --
```

Question 4:

Implementation Details:

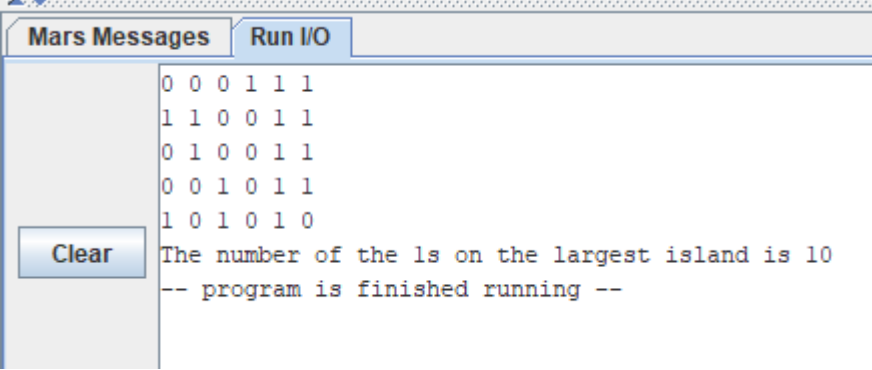
The program loads the first two elements(dimensions) of the matrix into registers \$t0 and \$t1. It then jumps to the printing subroutine. The printing initializes the necessary registers with the starting address and dimensions of the matrix. It starts a loop to traverse each element of the matrix and prints them out. When the loop completes, it exits to exit.

The program uses nested loops to traverse each element of the matrix. The outer loop iterates through the rows (matrix_outer_loop). The inner loop iterates through the columns (matrix_inner_loop). For each element, it prints the value and moves to the next element until the end of the row is reached. After printing each row, a newline character is printed to move to the next row. The program prints the output string ("The number of the 1s on the largest island is ") followed by the calculated island count (printoutputResult function). Finally, the program exits using syscall 10 (realExit).

Supporting Functions:

print_matrix_element: Prints the value of a matrix element.
print_space: Prints a space character.
move_next: Moves to the next element in the matrix.
print_matrix_exit: Exits the matrix printing process.
move_next_row: Moves to the next row in the matrix.
calcualte: Calculates the number of islands.
printoutputString: Prints the output string.
printoutputResult: Prints the calculated island count.
realExit: Exits the program.

Outputs:



```
Mars Messages Run I/O
0 0 0 1 1 1
1 1 0 0 1 1
0 1 0 0 1 1
0 0 1 0 1 1
1 0 1 0 1 0
The number of the 1s on the largest island is 10
-- program is finished running --
Clear
```