Cabdifatah Mohamed

November 2024

1 Introduction

The report describes the design, implementation, and testing of specified test cases that test the translator for the **F-15** subset language. The translator processes inputs of the specified language and generates corresponding assembly languages. Each piece of the program shows the specific computational functions of the compiler, as shown below:

- **Test Case 1:** Add two numbers
- **Test Case 2:** Specific Arithmetic Expression
- **Test Case 3:** Calculate the sum of two absolute values
- **Test Case 4:** Compute the maximum and minimum of four numbers
- **Test Case 5:** Compute the sum of the first natural numbers n.
- **Test Case 6:** Compute the first $(n + 1)$ **Fibonacci** series.

2 Explanation of Programs

2.1 Test Case 1

Goal: Read two numbers, add them, and print the result.

- **Input:** 4 5
- **Output:** The sum is: 9
- **Expected Function:** The program reads two integers, and computes their sum, and displays the result.

2.2 Test Case 2

Goal: Calculate $x = 2 + 3 - (7 - 4) + 8$ and print the result

- **Input:** No input is required

- **Output:** the value of x is: 10
- **Expected Function:** A static arithmetic expression is evaluated, and the result is printed.

2.3 Test Case 3

Goal: Compute the absolute value of two integers, add them, and print the result.

- **Input:** -73
- **Output:** The sum of absolute value is: 10
- **Expected Function:** The program automatically converts negative values to positive, adds them, and displays the result.

2.4 Test case 4

Goal: Read four numbers, compute the maximum and minimum, and print the result.

- **Input:** 3 8 1 6
- **Output:** Maximum: 8, Minimum: 1
- **Expected Function:** The program identifies the highest and lowest numbers among the inputs and displays them.

2.5 Test Case 5

Goal: Compute the sum of the first n natural numbers and print the result.

- **Input:** 5
- **Output:** The sum of the first 5 natural numbers is: 15
- **Expected Function:** The program calculates the sum $1 + 2 + 3 + ... + n$ iteratively and prints the result.

2.6 Test Case 6

Goal: Generate the first $(n + 1)$ **Fibonacci** numbers

- **Input:** 4
- **Output:** 0 1 1 2 3
- **Expected Function:** The program generates **Fibonacci** series using iterative addition.

Test File	Test Cases	Pass/Fail	Explanation
Test Case 1	Add two positive numbers	Pass	Correct addition logic.
	Add two negative numbers	Pass	Handles negatives correctly.
	Add a positive and negative	Pass	Works as expected.
Test Case 2	Evaluate static expression	Pass	Correct arithmetic precedence.
Test Case 3	Compute absolute values	Pass	Handles negative and positive inputs.
	Input with zero values	Pass	Works correctly.
Test Case 4	Compute max/min of 4 numbers	Pass	Handles all cases accurately.
Test Case 5	Compute sum of n numbers	Pass	Works for small and large n.
Test Case 6	Generate Fibonacci sequence	Pass	Handles small n correctly.
	Handle large n values	Fail	Stack overflow error.

Table 1: Testing Results for F-15 Programs

3 Testing Results

4. Issues and Observations

4.1 Passing Cases

The following functionalities worked as expected:

- Arithmetic operations (Test Case 1, Test Case 2).
- Conditional branching for absolute values (Test Case 3).
- Iterative loops for sum and Fibonacci sequences (Test Case 5, Test Case 6 for small n).
- Correct handling of maximum and minimum values (Test Case 4).

3.1 Failing Cases

The following cases were unable to pass the test:

- **Test Case 6 with large** n: The problem fails due to excessive memory allocation in iterative loops. The Fibonacci sequence grows exponentially, consuming more stack space than available.
- **Error Handling:** Inputs outside the defined range (e.g., negative n for Test Case 5 and Test Case 6) are not handled.

4 Suggested Improvement

1. Optimize Fibonacci logic using a memory-efficient algorithm (e.g. dynamic programming).
2. Add input validation to ensure appropriate data ranges.
3. Expand test cases to include boundary conditions and invalid inputs.

5 Conclusion

The project successfully implemented six F-15 programs and tested their functions using the F-15 translator. While most cases passed, handling edge cases and optimization for large inputs require further improvements.