HO CHI MINH UNIVERSITY OF TECHNOLOGY

ELECTRICAL ELECTRONIC ENGINEERING

Electronic Department

----------o0o----------

LOGIC DESIGN / LOGIC SYNTHESIS

Reverse Polish Notation Calculator

INSTRUCTORS: **Tran Hoang Linh**

Nguyen Tuan Hung

STUDENT : **Ha Gia Huy _ 2051117**

Tran Minh Tri _ 2051022

HO CHI MINH CITY, FEBRUARY, 2024

Table of Contents

I. OVERVIEW

The purpose of this project is to use what we have learn in previous lab to make something fun. We decide most of the specification in this project. We are going to make a Reverse Polish notation calculator. Every calculators that has a C(clear) button, input as pressing a number then press "+ -" then press another number then press "=" is this RPN Calculator.

But the orignal calculator has a translator, because RPN works like this. "3 4 5 + - =" is "3 + 4 - 5 =". So we need to make that translator too. The real calculators has buttons, DE10 has buttons but much less. So let's improvise an interface.

- KEY0 is Reset/Clear; KEY1 is Input
- SW0 to SW3 is recognize as follows when you press the Input button.
- The LEDR0 will light when it's showing the result.
- The LEDR1 will light when you input something illegal like "999" or a "NULL" command.

Binary	Value	Binary	Value	Binary	Value	Binary	Value
0000	0	0100	4	1000	8	1100	“*”
0001	1	0101	5	1001	9	1101	NULL
0010	2	0110	6	1010	“+”	1110	“=”
0011	3	0111	7	1011	“-”	1111	NULL

The 7segments LED HEX0 to HEX5 will display the number we JUST input (after we press the input). It should not display "+,-" or "=". After we input "=" it will display the result. For example, we want to calculate 99 + 102 = ? We will press the following buttons:

- SW[30] = 1001 (9) and press KEY1 (Display: 9)
- SW[30] = 1001 (9) and press KEY1 (Display: 99)
- SW[30] = 1011 (-) and press KEY1 (Display: 99)
- SW[30] = 0001 (1) and press KEY1 (Display: 1)
- SW[30] = 0000 (0) and press KEY1 (Display: 10)
- SW[30] = 0010 (2) and press KEY1 (Display: 102)
- SW[30] = 1110 (=) and press KEY1 (Display: 201)

II. THEORY

2.1. RPN Converter (Translator)

In reverse Polish notation, the operators follow their operands. For example, to add 3 and 4 together, the expression is 3 4 + rather than 3 + 4. The conventional notation expression 3 − 4 + 5 becomes 3 4 − 5 + in reverse Polish notation. The concept of a stack, a last-in/first-out construct, is integral to the left-to-right evaluation of RPN.

3 + 4

Figure 1. Infix notation

3 4 +

Figure 2. Postfix notation ("Reverse Polish")

2.2. RPN Calculator

Reverse Polish Notation (RPN) is a notation for representing arithmetic formula. The common notations that we are familiar with had the operator placed in between the two operands (called the infix notation), for example, A + B, A * B, A + B + C. The RPN notation (or postfix notation) places the operator at the end of the operands, for example, A B +, A B *, A B + C +. The RPN notation has the advantage that it can express operation precedence without using parentheses. For example,

- Expression in infix notation: (A + B) * (C + D)
- Same expression in postfix notation: A B + C D + *

A stack implementation is typically used to evaluate an expression in RPN. For example, to calculate A B + C D + *. Scanning the RPN expression from left to right. If it is an operand, it is pushed onto the stack. If it is an operator, the top two items on the stack are popped and evaluated with the operator; the result is pushed back onto the stack.

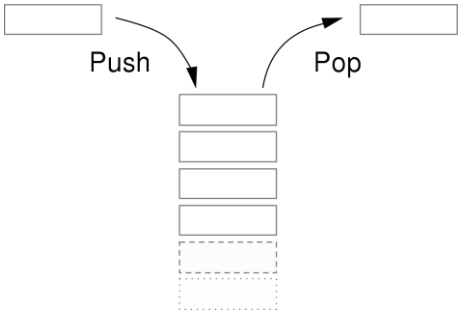
S1 = A + B | S2 = C + D | S3 = S1 * S2

A	B	+	C	D	+	*
A	B	S1	C	D	S2	S3
	A		S1	C	S1	
				S1		

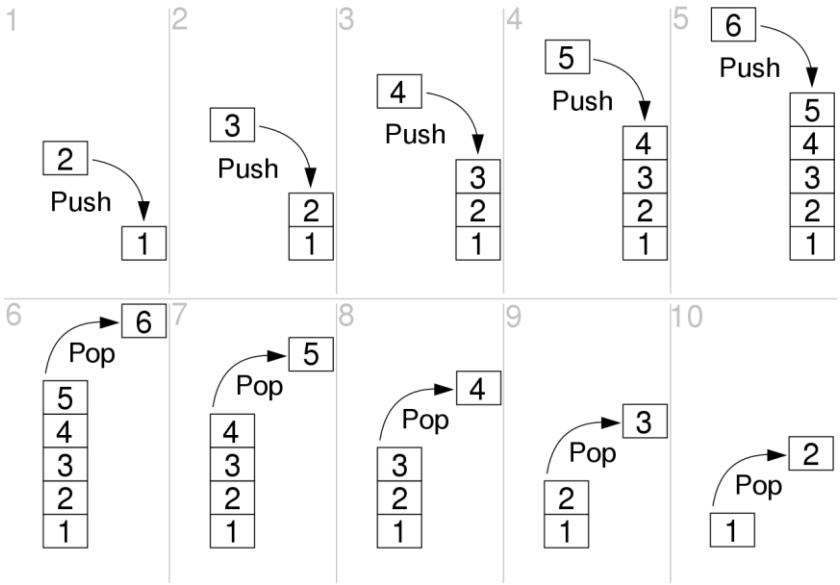
2.3. Stack

A stack is an abstract data type that serves as a collection of elements:

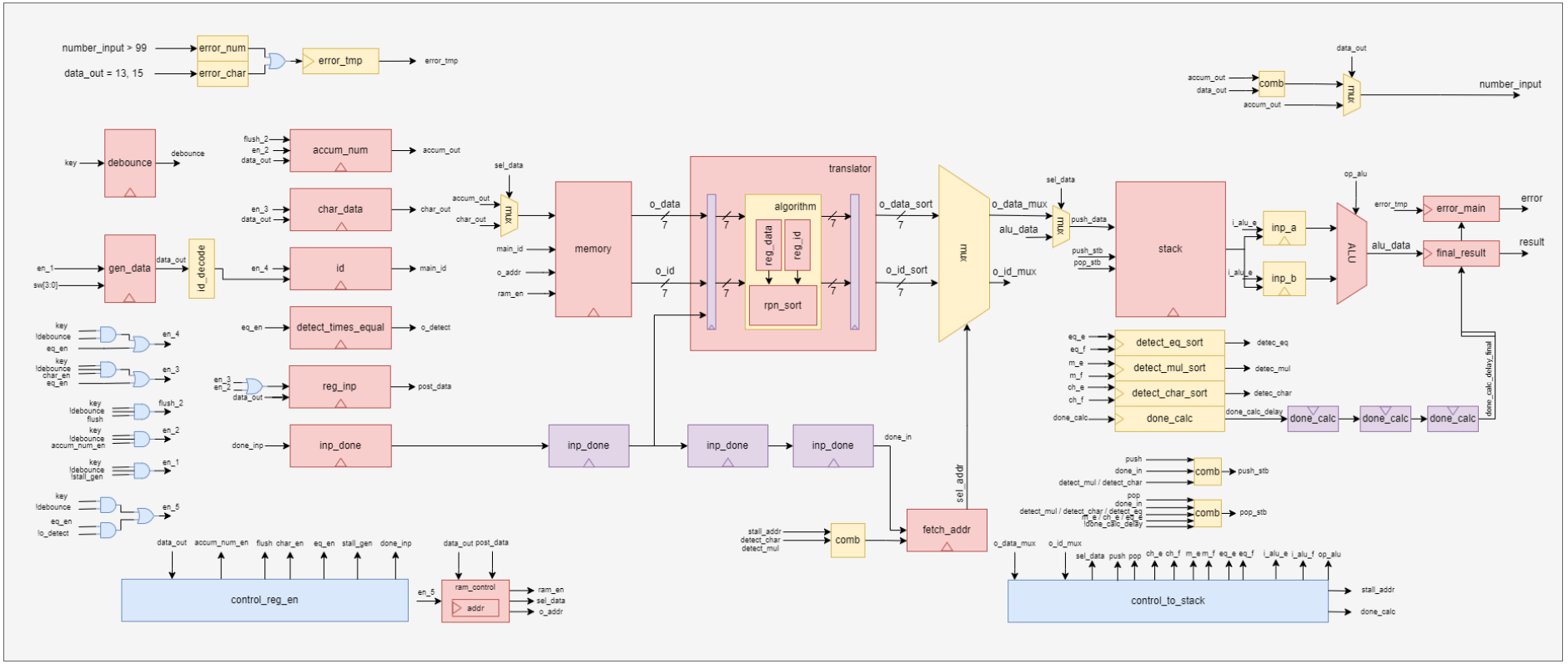
- Push, which adds an element to the collection.
- Pop, which removes the most recently added element.



A peek operation allows for retrieving the value of the last element added to the stack without altering the stack itself. The term "stack" draws an analogy to physical items arranged in a stack, like a stack of plates. The sequence of adding and removing elements from a stack follows the Last In, First Out (LIFO) principle. Similar to a physical stack, this structure facilitates easy removal of the topmost item, while accessing a deeper item may necessitate removing other items first. Defined as a linear data structure, or more abstractly as a sequential collection, a stack has one end designated for push and pop operations - the top of the stack - and a fixed end - the bottom. This structure can be implemented using a singly linked list with a pointer to the top element. A stack can be given a bounded capacity. When the stack reaches full capacity and cannot accommodate additional elements, it enters a state of "stack overflow." A stack is essential for implementing depth-first search algorithms.

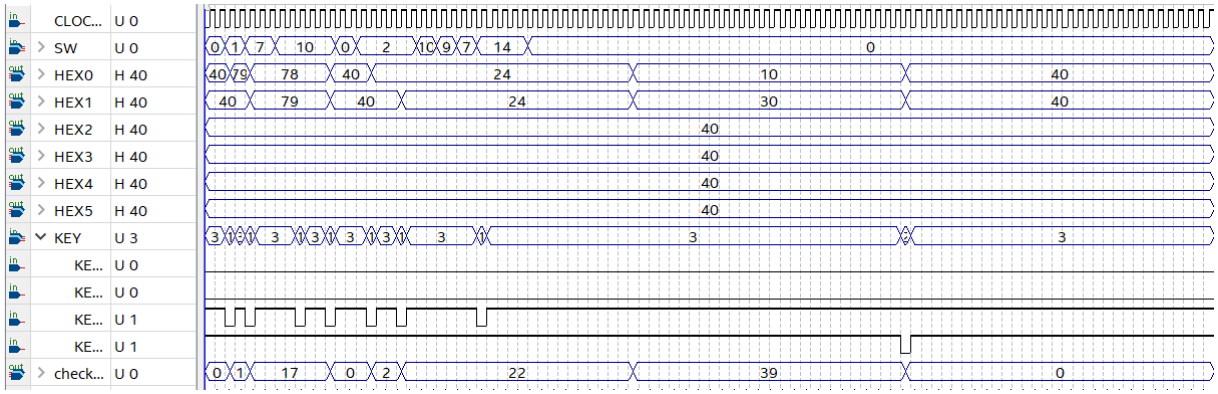


III. DESIGN STRATEGY

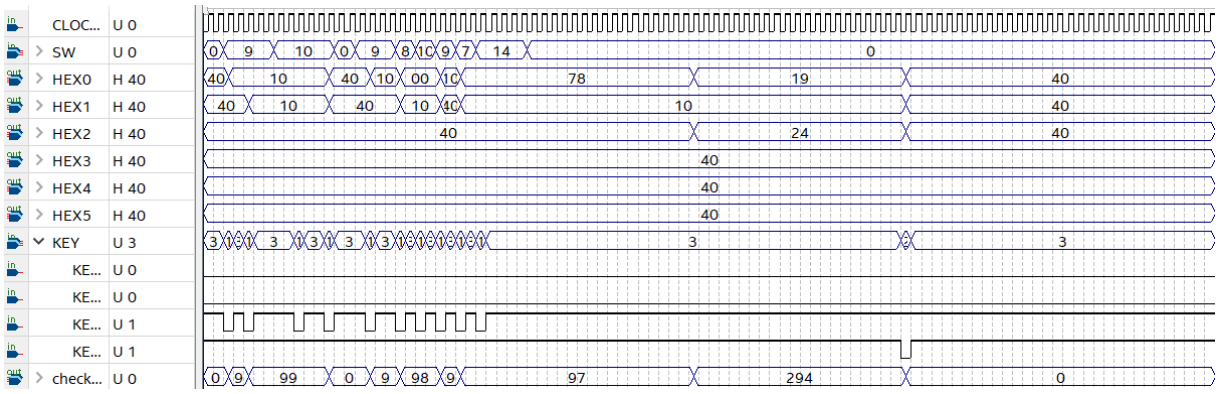


IV. SIMULATING ON QUARTUS

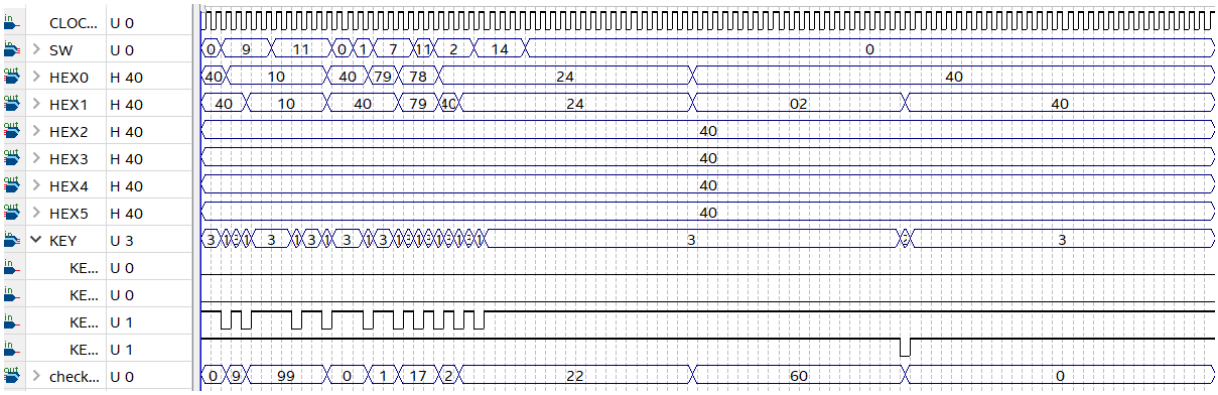
17 + 22 = 39



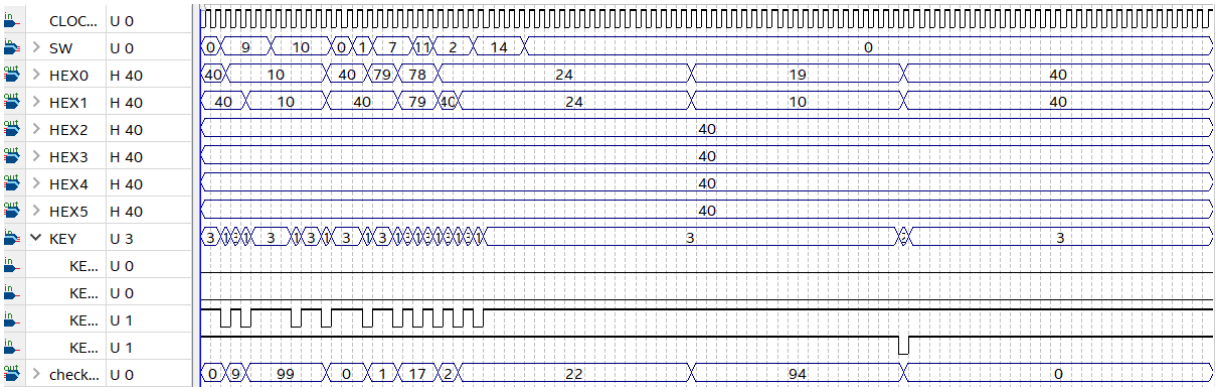
99 + 98 + 97 = 294



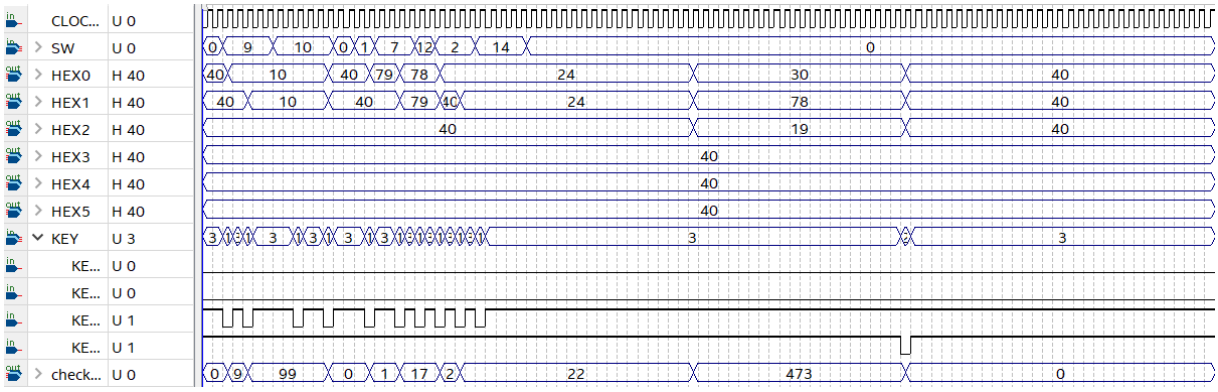
99 – 17 – 22 = 60



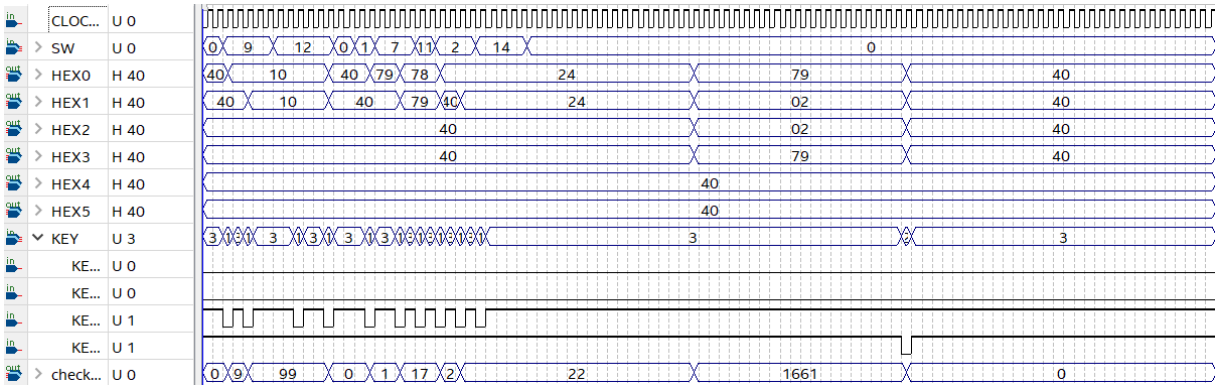
99 + 17 – 22 = 94



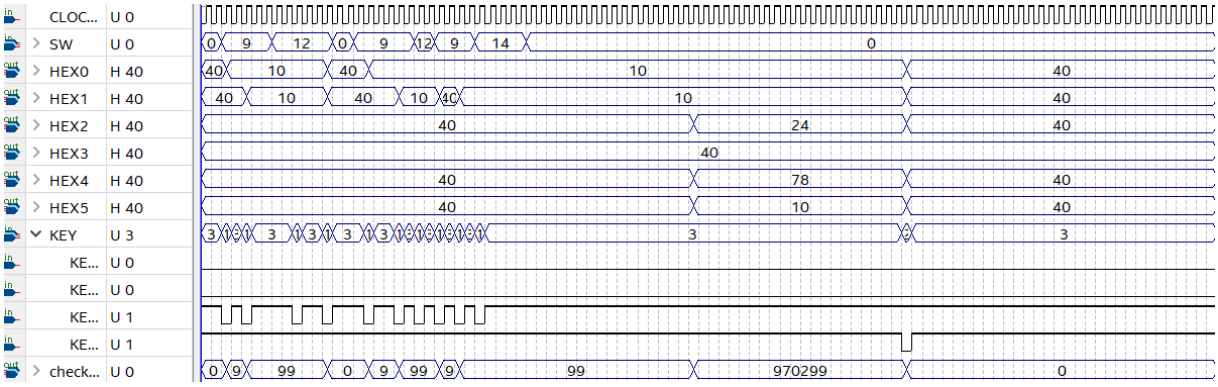
99 + 17 * 22 = 473



99 * 17 – 22 = 1661

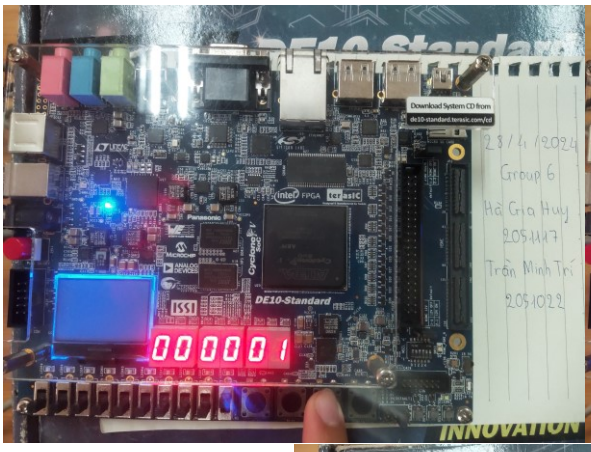


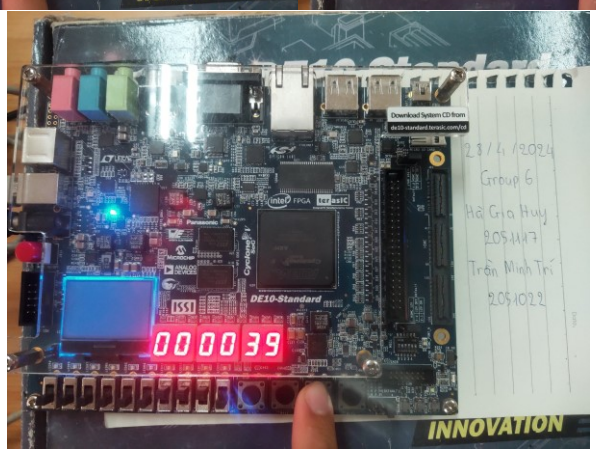
99 * 99 * 99 = 970299



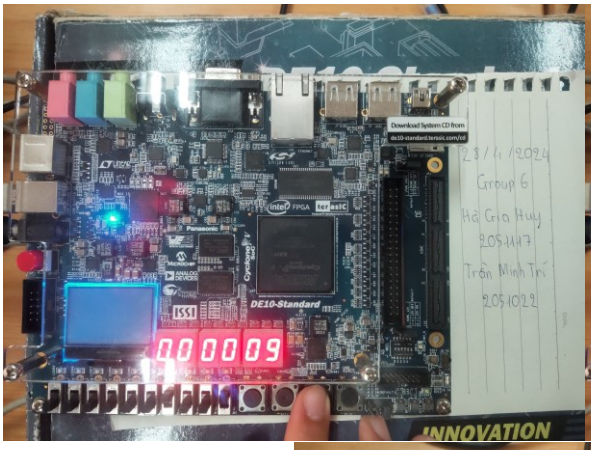
V. IMPLEMENTING ON HARDWARE

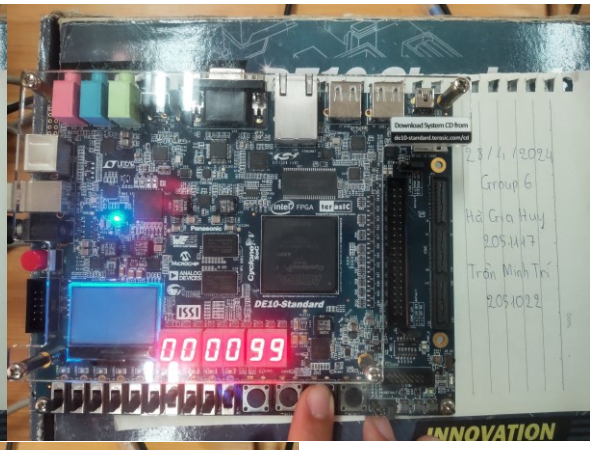
❖ Case 1: 17 + 22 = 39

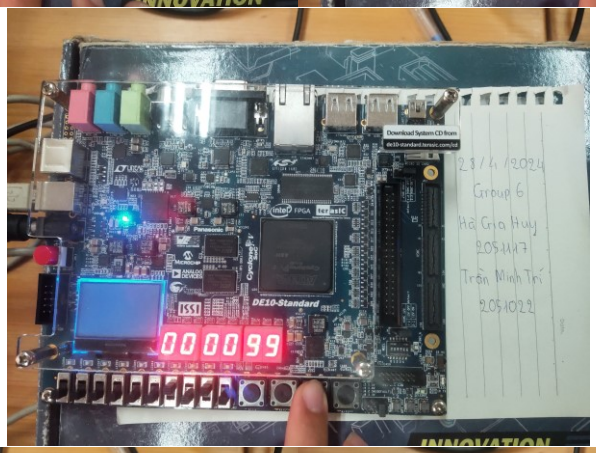


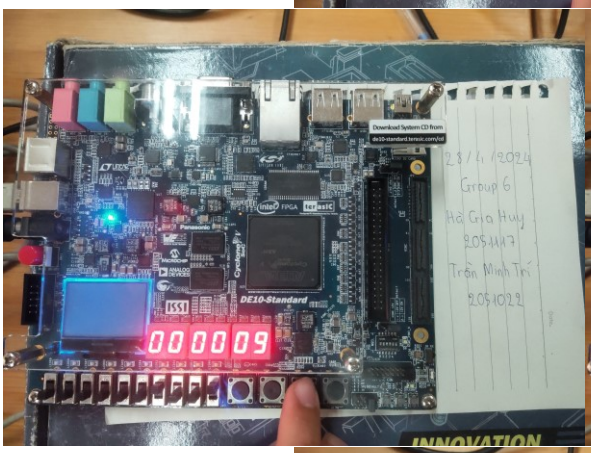


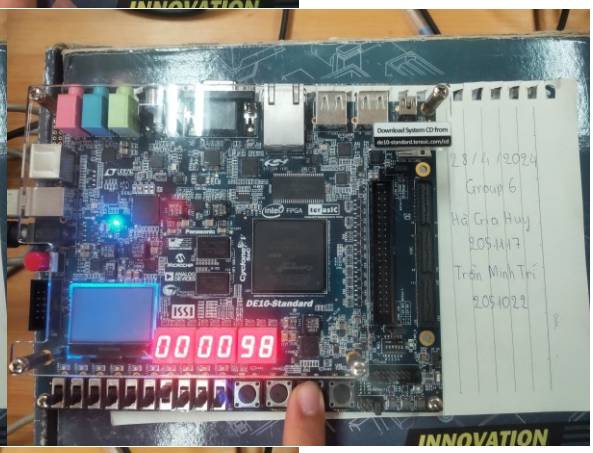
❖ **Case 2: 99 + 98 + 97 = 294**



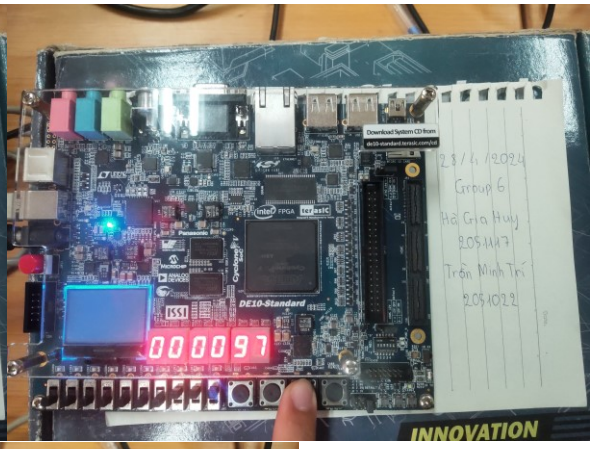






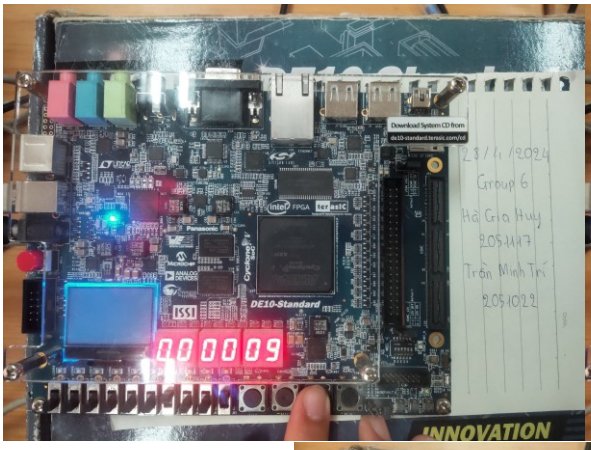


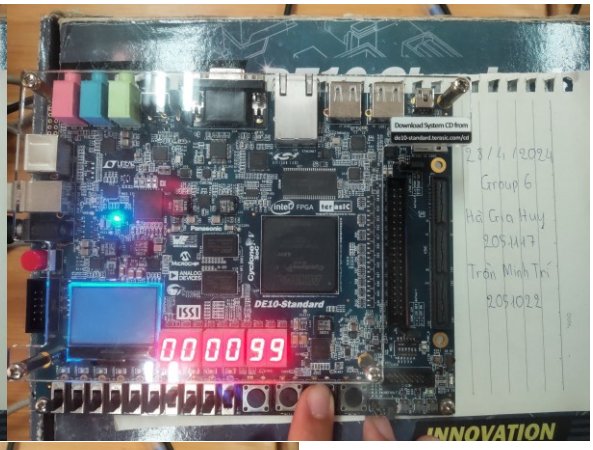

000009
DE10-Standard
INNOVATION
23/4/2024
Group 6
Hà Gia Huy
20511147
Trần Minh Trí
20510221

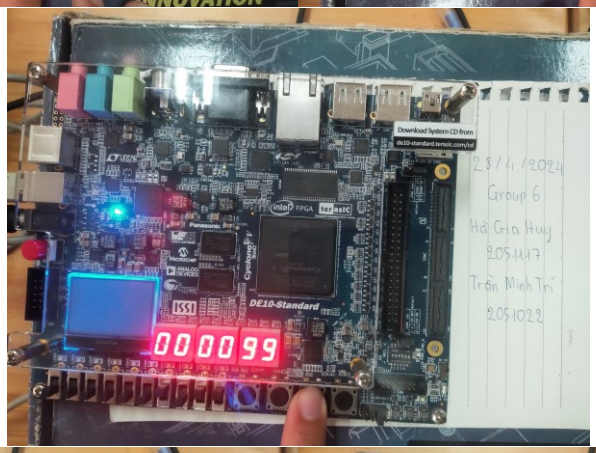

000097
DE10-Standard
INNOVATION
23/4/2024
Group 6
Hà Gia Huy
20511147
Trần Minh Trí
20510221

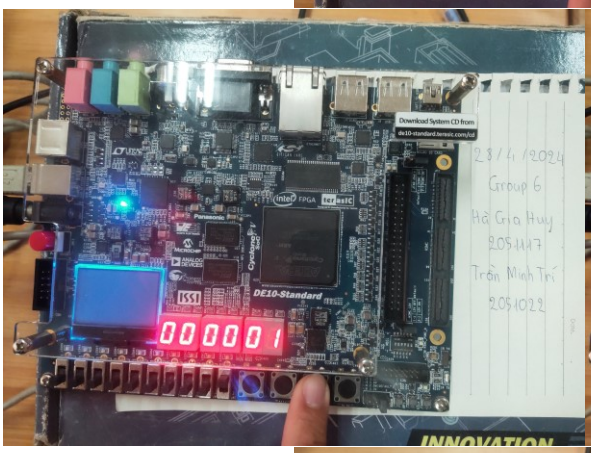

000294
DE10-Standard
INNOVATION
23/4/2024
Group 6
Hà Gia Huy
20511147
Trần Minh Trí
20510221

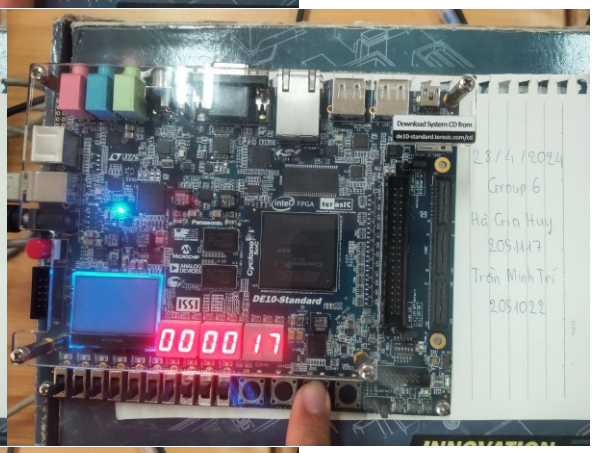
❖ **Case 3: 99 - 17 - 22 = 60**

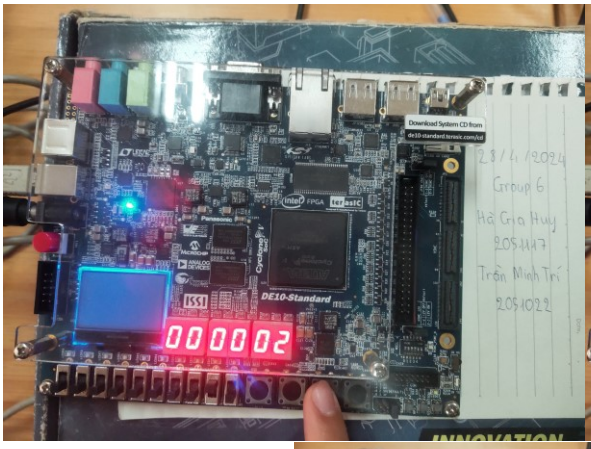


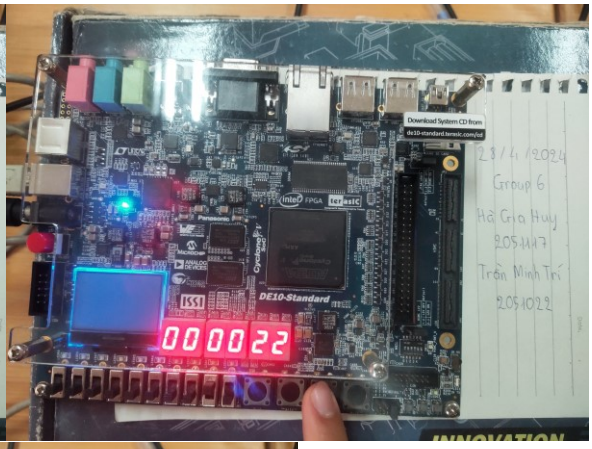


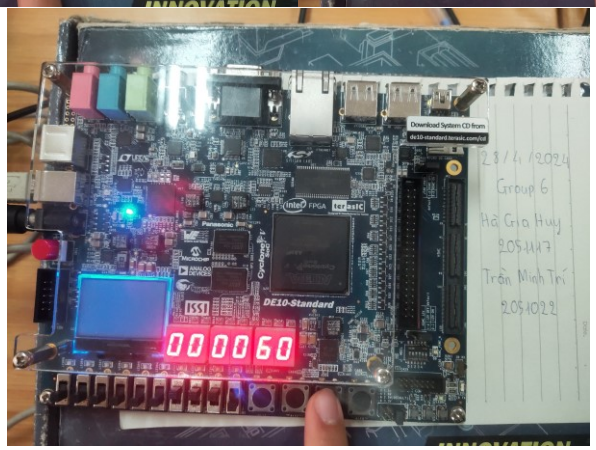




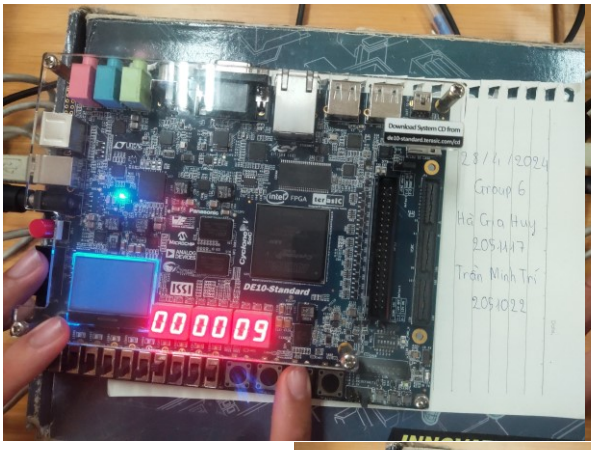


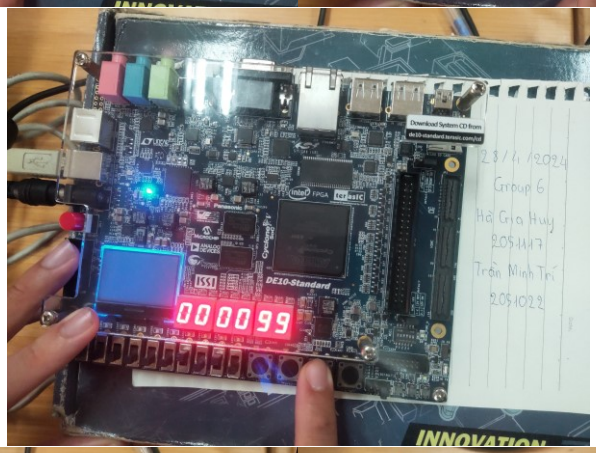

Download System CD from
de10-standard.terasic.com/cd
DE10-Standard
000002
23/4/2024
Group 6
Hà Gia Huy
20$1117
Trần Minh Trí
20$1022

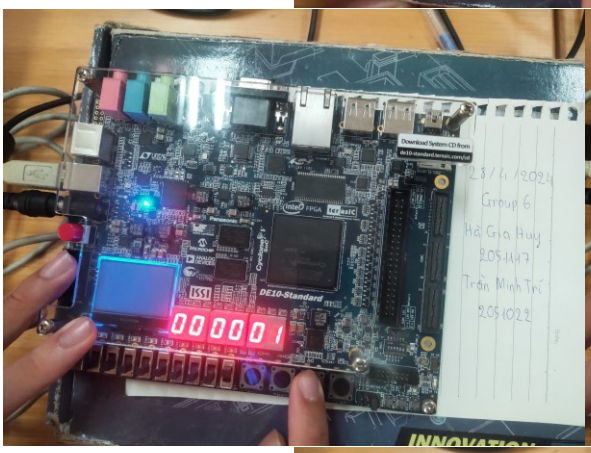

Download System CD from
de10-standard.terasic.com/cd
DE10-Standard
000022
23/4/2024
Group 6
Hà Gia Huy
20$1117
Trần Minh Trí
20$1022

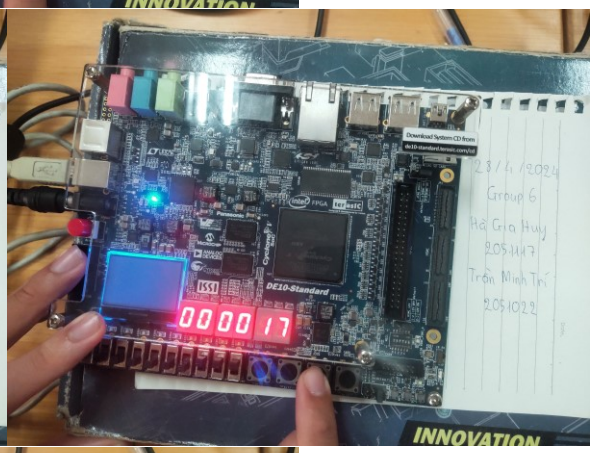

Download System CD from
de10-standard.terasic.com/cd
DE10-Standard
000060
23/4/2024
Group 6
Hà Gia Huy
20$1117
Trần Minh Trí
20$1022

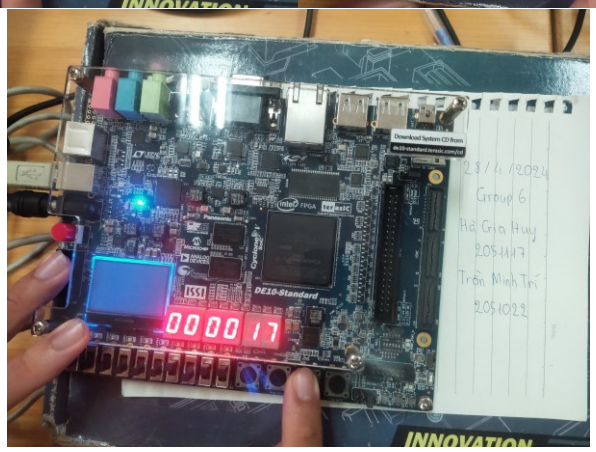
❖ **Case 4: 99 + 17 - 22 = 94**

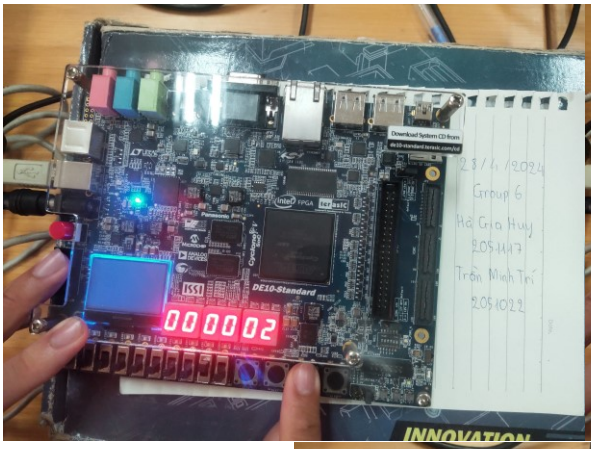


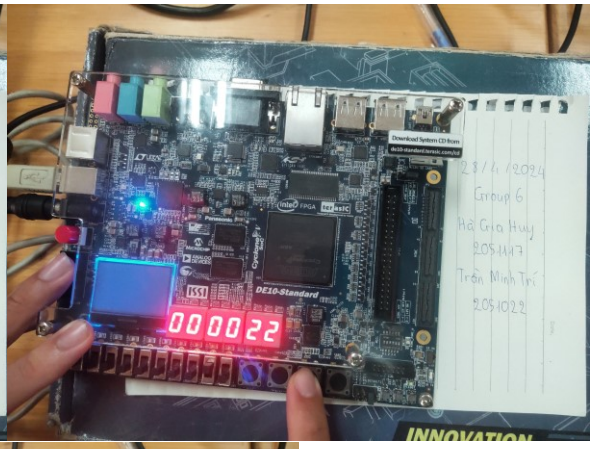


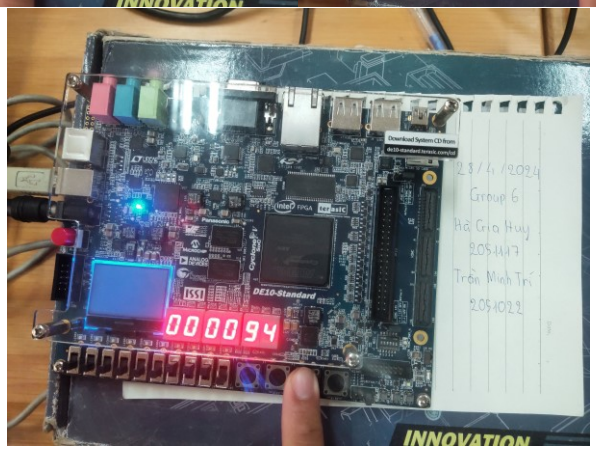




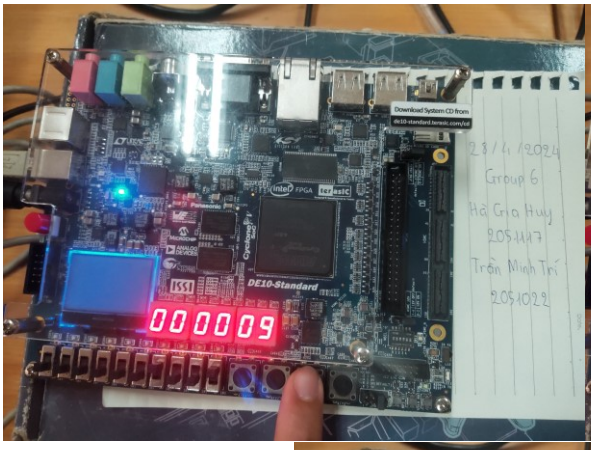


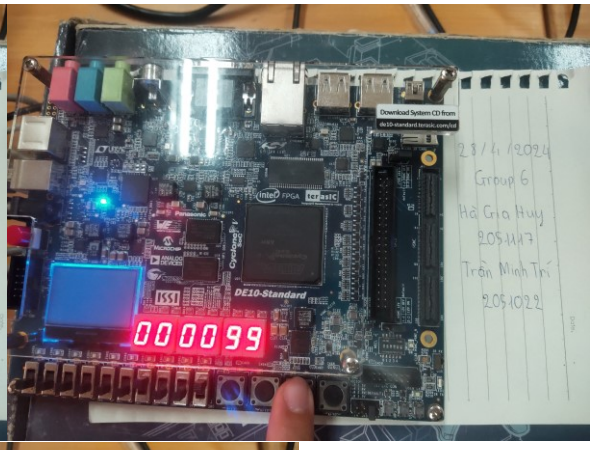

000002
DE10-Standard
23/1/2024
Group 6
Hà Gia Huy
20511117
Trần Minh Trí
20511022

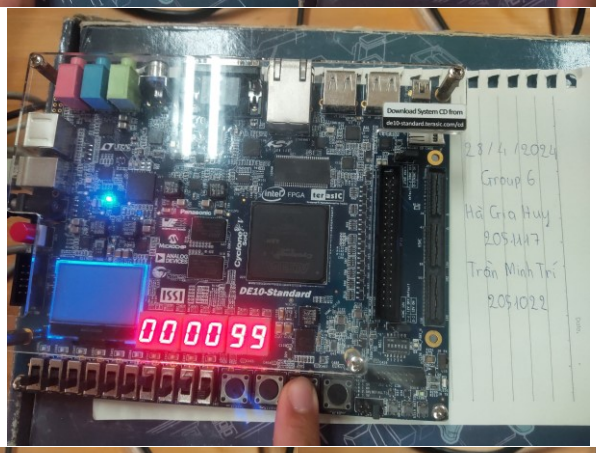

000022
DE10-Standard
23/1/2024
Group 6
Hà Gia Huy
20511117
Trần Minh Trí
20511022

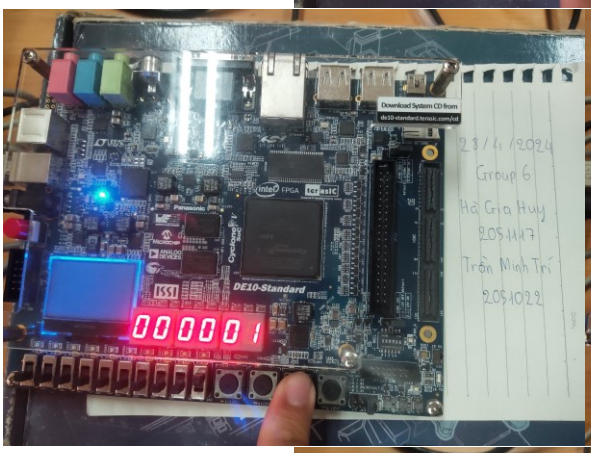

000094
DE10-Standard
23/1/2024
Group 6
Hà Gia Huy
20511117
Trần Minh Trí
20511022

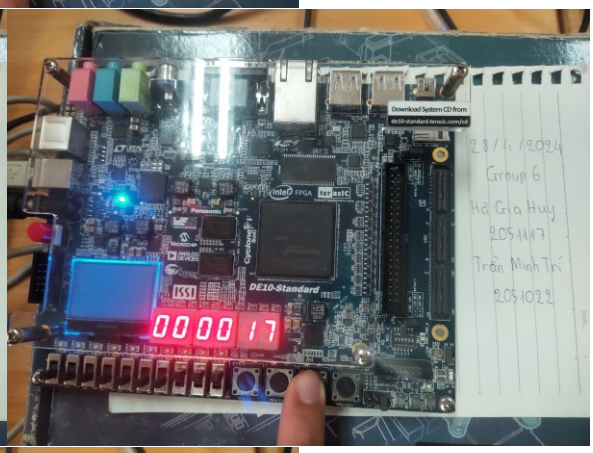
❖ **Case 5: 99 + 17 * 22 = 473**

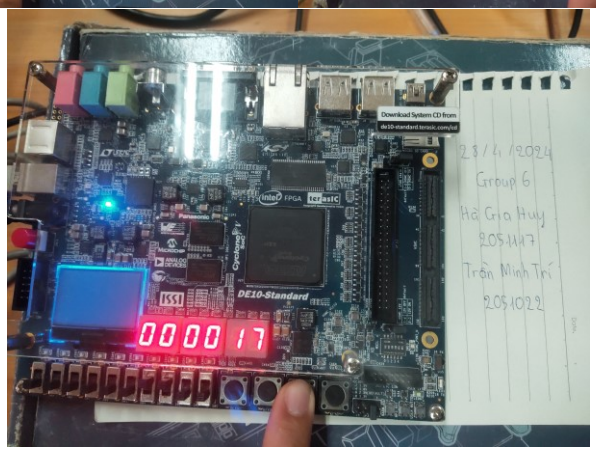


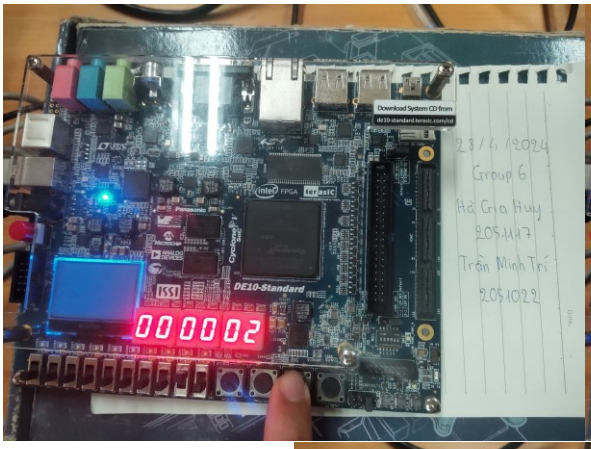


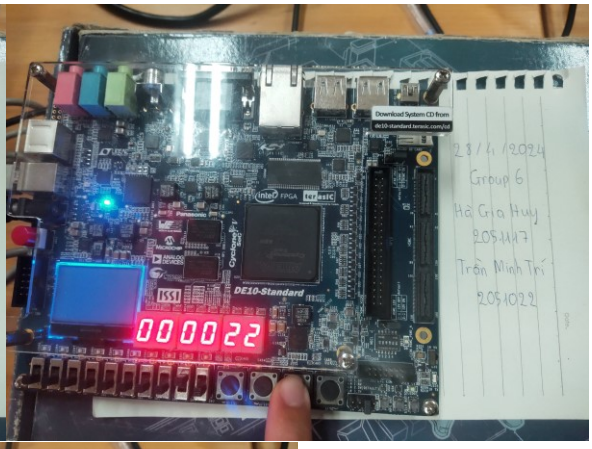


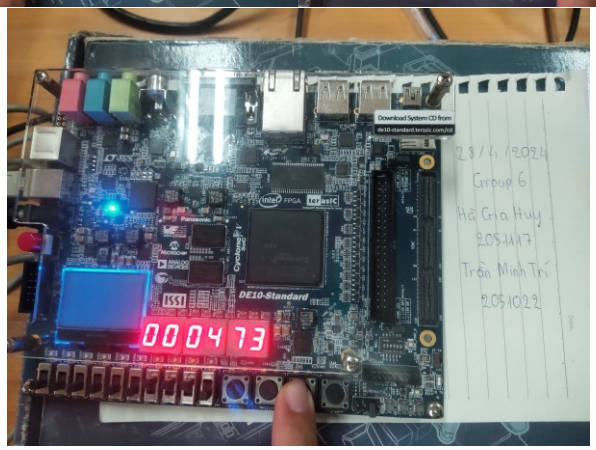




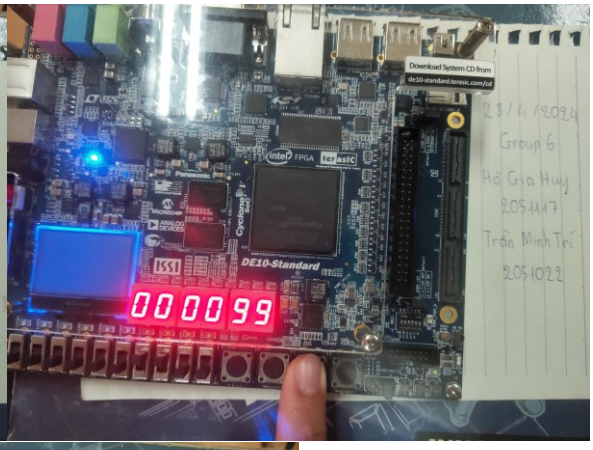


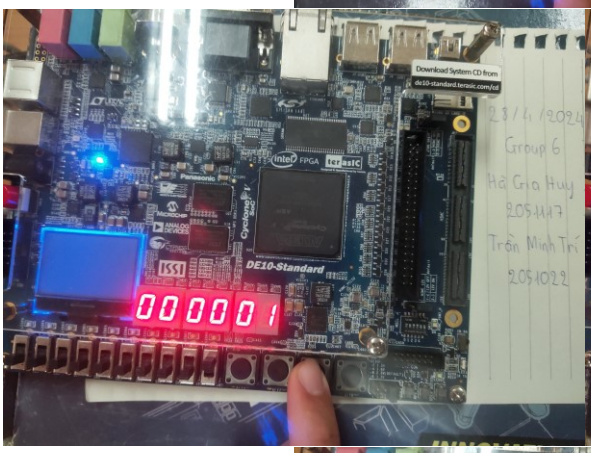

Download System CD from
de10-standard.terasic.com/cd
DE10-Standard
000002
23/4/2024
Group 6
Hà Gia Huy
20511147
Trần Minh Trí
20511022

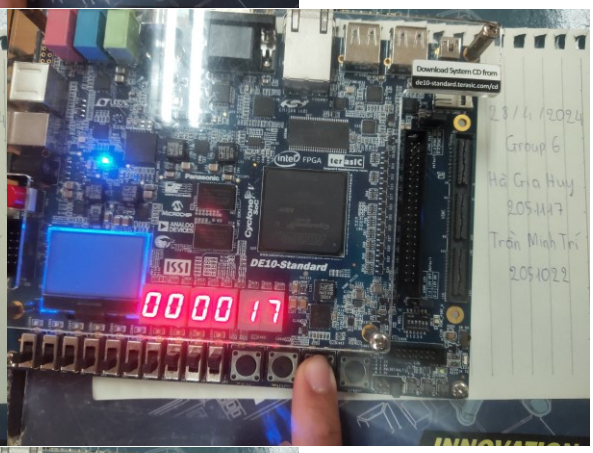

Download System CD from
de10-standard.terasic.com/cd
DE10-Standard
000022
23/4/2024
Group 6
Hà Gia Huy
20511147
Trần Minh Trí
20511022

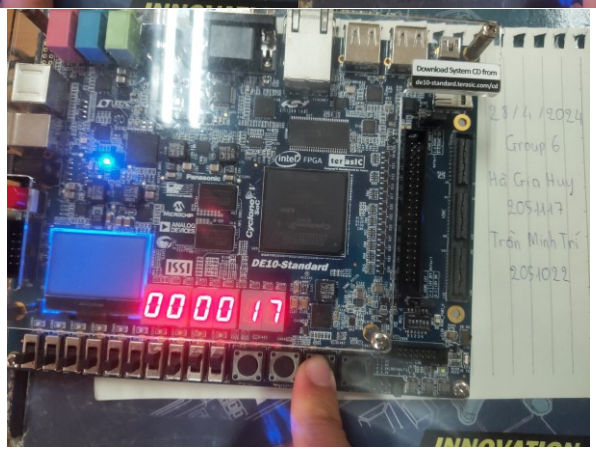

Download System CD from
de10-standard.terasic.com/cd
DE10-Standard
000473
23/4/2024
Group 6
Hà Gia Huy
20511147
Trần Minh Trí
20511022

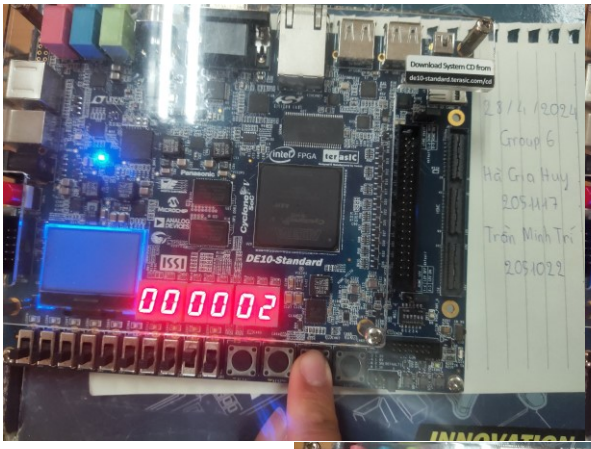
❖ **Case 6: 99 * 17 - 22 = 1661**

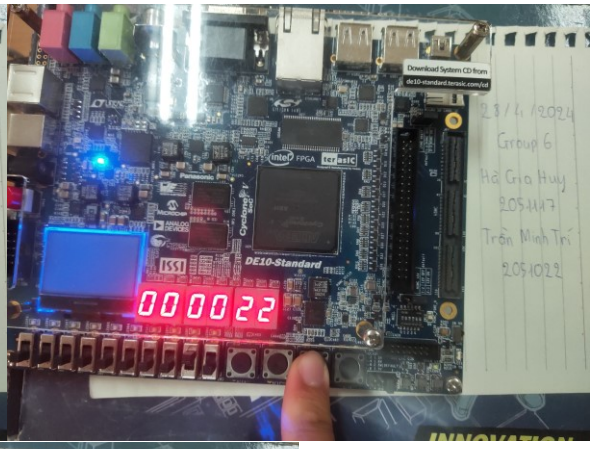


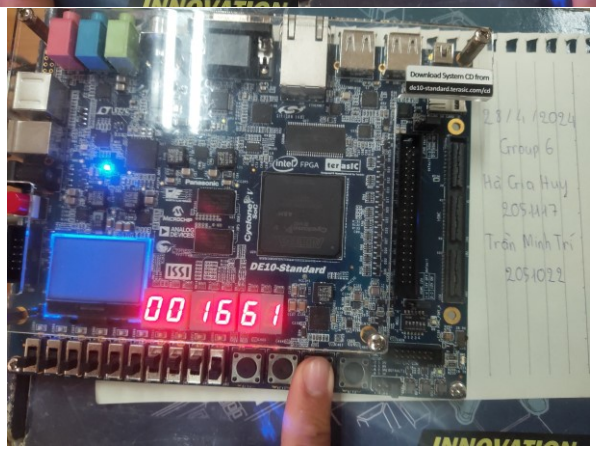




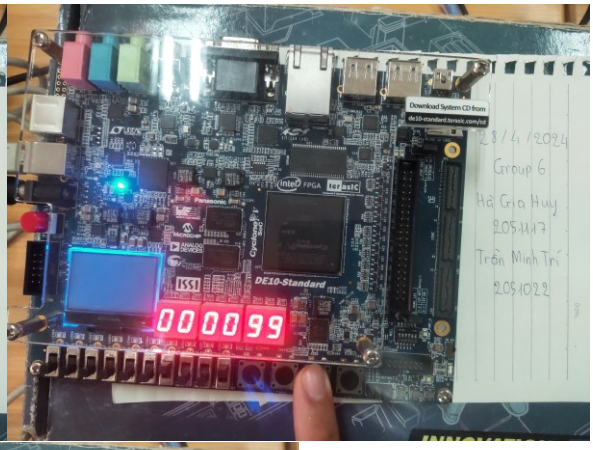


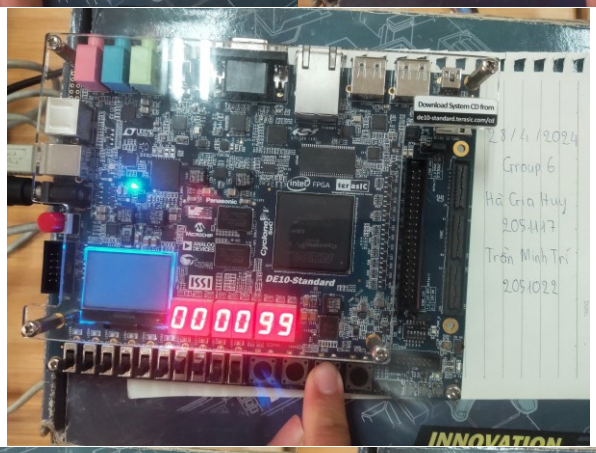

Download System CD from
de10-standard.terasic.com/cd
DE10-Standard
000002
23/4/2024
Group 6
Hà Gia Huy
20$1117
Trần Minh Trí
20$1022

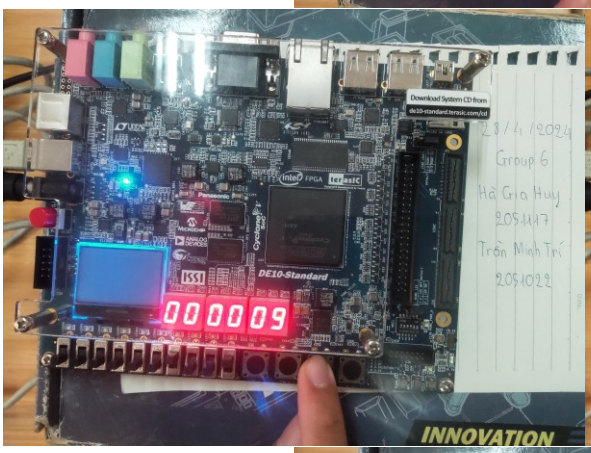

Download System CD from
de10-standard.terasic.com/cd
DE10-Standard
000022
23/4/2024
Group 6
Hà Gia Huy
20$1117
Trần Minh Trí
20$1022

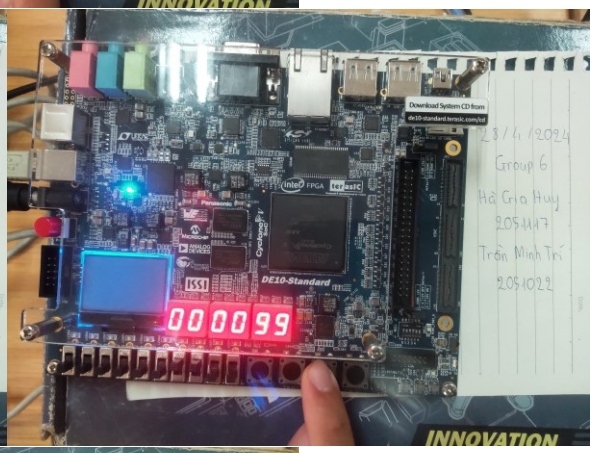

Download System CD from
de10-standard.terasic.com/cd
DE10-Standard
001661
23/4/2024
Group 6
Hà Gia Huy
20$1117
Trần Minh Trí
20$1022

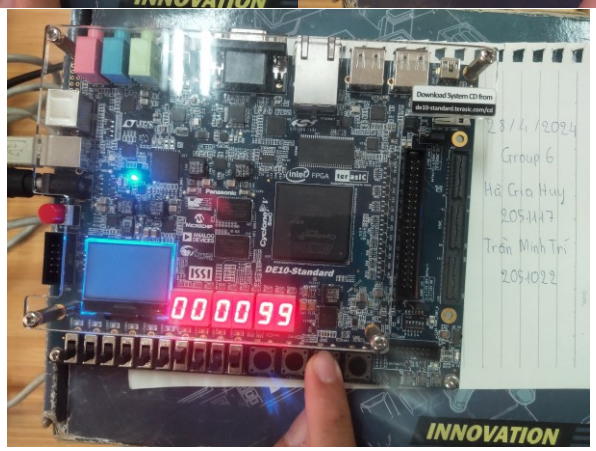
❖ **Case 7: 99 * 99 * 99 = 970299**

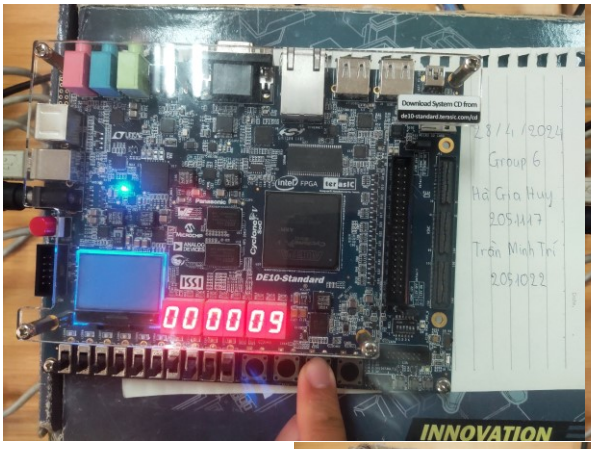


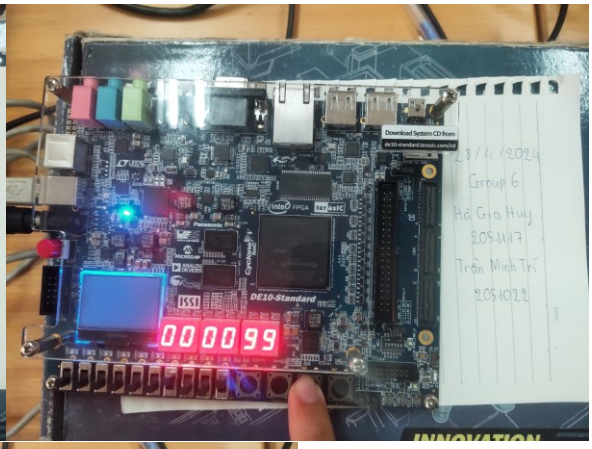


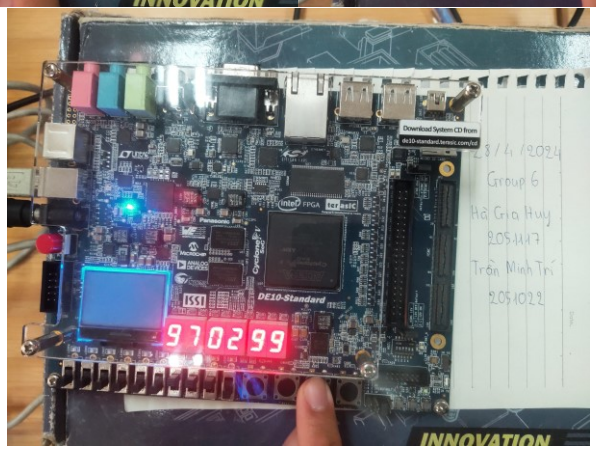




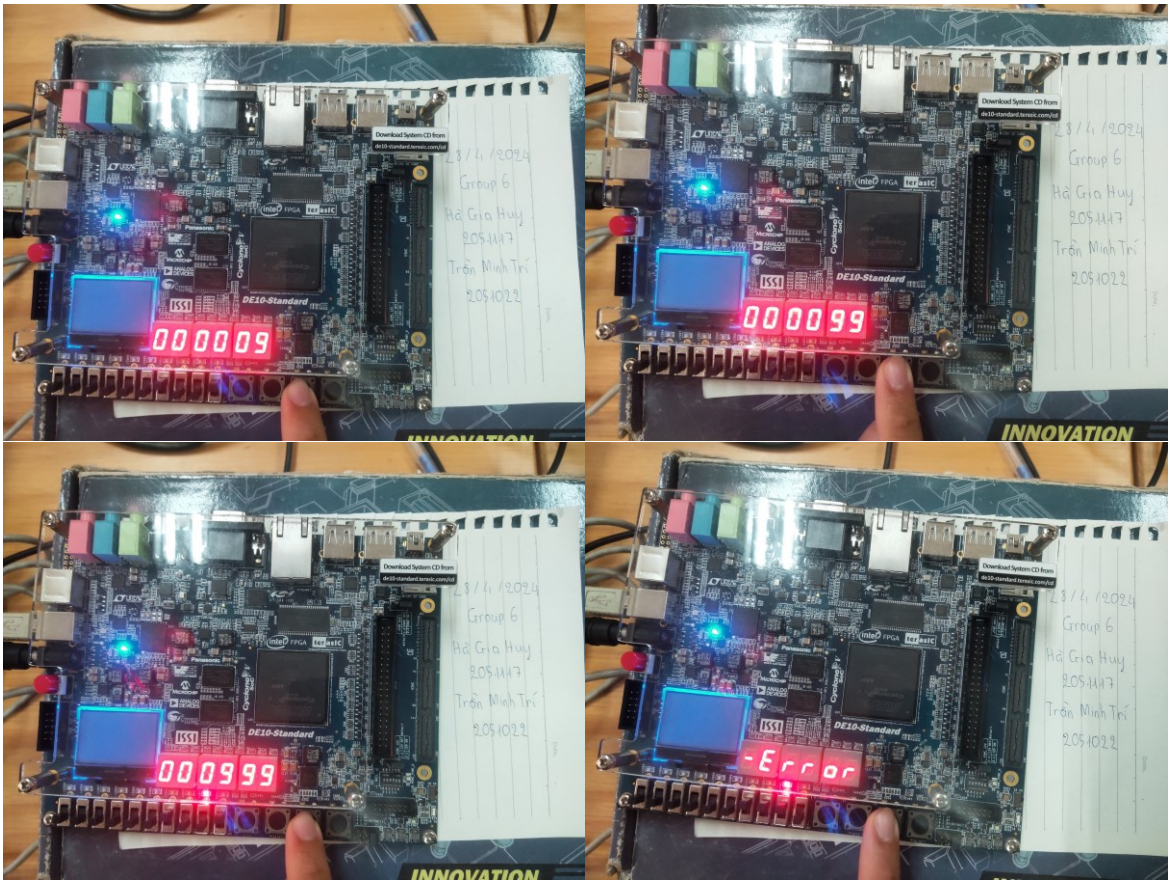



Download System CD from
de10-standard.terasic.com/cd
DE10-Standard
000009
INNOVATION
23/4/2024
Group 6
Hà Gia Huy
20S1117
Trần Minh Trí
20S1022


Download System CD from
de10-standard.terasic.com/cd
DE10-Standard
000099
INNOVATION
23/4/2024
Group 6
Hà Gia Huy
20S1117
Trần Minh Trí
20S1022


Download System CD from
de10-standard.terasic.com/cd
DE10-Standard
970299
INNOVATION
23/4/2024
Group 6
Hà Gia Huy
20S1117
Trần Minh Trí
20S1022

❖ **Case 8: Error Number**



❖ **Case 9: Reset Calculator**

VI. CONCLUSION

The task involves understanding Reverse Polish Notation (RPN) and its functionality within a stack module provided in stack.sv. The goal is to create a calculator that processes input commands using this stack-based approach, allowing for the implementation of additional functions in Lab5. There are at max 3 operation (only addition, substration, multiplication) and 3 numbers in our calculator. The input number will not be more than 2 digits. The solution involves designing a Finite State Machine (FSM) to handle and process the input and commands in a sequential manner. This approach will enable the creation of a versatile calculator capable of executing various operations efficiently.

VII. REFERENCES

1. Reverse Polish notation
Access from: https://en.wikipedia.org/wiki/Reverse_Polish_notation

2. Stack (abstract data type)
Access from: https://en.wikipedia.org/wiki/Stack_(abstract_data_type)