Logo, company name

Description automatically generated

**COMSATS University Islamabad (CUI)**

**Lab Terminal**

**Submitted to: Sir Bilal Bukhari**

**Submitted By: Sohaib Safeer**

**Reg No: FA21-BCS-054**

**Course: Topics in Computer Science**

**Date: 3 jan, 2025**

Question 5 : give input and output of minicompiler

**Input:**

* **Source Code:** The input to the mini-compiler is a program written in a simplified subset of the C language. This subset typically includes:
  + Basic data types (e.g., int, char)
  + Arithmetic operations (+, -, \*, /)
  + Relational operators (<, >, <=, >=, ==, !=)
  + Conditional statements (if-else)
  + Loops (while)
  + Variable declarations and assignments
  + Function calls (with a limited number of parameters)

**Example Input:**

C

int main() {

int x = 5;

int y = 3;

int sum = x + y;

if (sum > 7) {

return 1;

} else {

return 0;

}

}

**Output:**

The output of the mini-compiler can vary depending on its specific implementation and the desired level of output:

* **Intermediate Representation:**
  + The most common output is an intermediate representation of the source code.
  + This could be in the form of:
    - Three-address code
    - Assembly-like instructions
    - Abstract Syntax Tree (AST)
  + This intermediate representation can then be further processed by other tools, such as an optimizer or a code generator.
* **Assembly Code:** Some mini-compilers can directly generate assembly code for a specific target architecture.
* **Machine Code:** In more advanced cases, the mini-compiler might generate machine code that can be directly executed by the target processor.
* **Error Messages:** If the input source code contains syntax errors or semantic errors, the mini-compiler should generate appropriate error messages to help the programmer identify and correct the issues.

**Example Output (Intermediate Representation):**

LOAD 5

STORE x

LOAD 3

STORE y

LOAD x

PUSH

LOAD y

PUSH

ADD

STORE sum

LOAD sum

PUSH

LOAD 7

PUSH

GT

JZ label1

LOAD 1

RETURN

label1:

LOAD 0

RETURN