Task 3- Answers

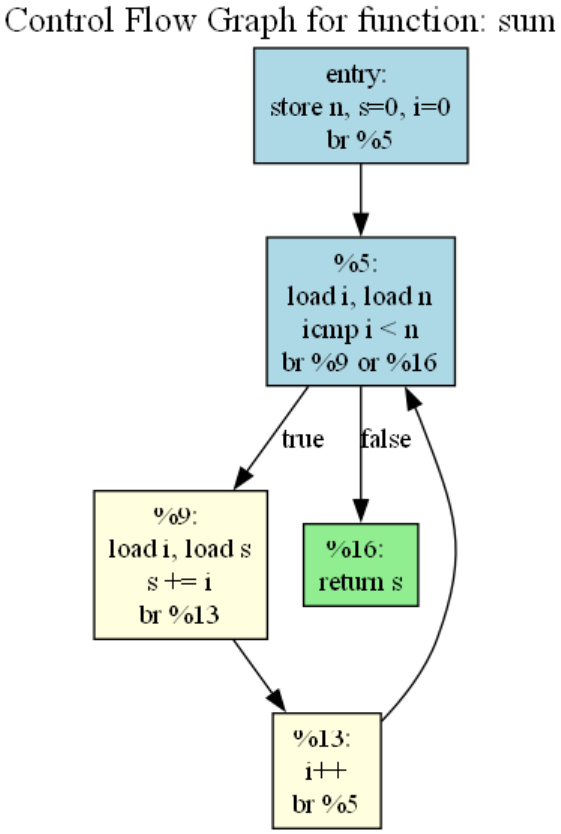
Part 1 (src/condition.cpp and ir/condition.ll):

- How is x>5 checked?
  - Via script – ‘ %5 = icmp sgt i32 %4, 5 ’.
  - This is the signed int comparison between x and 5.
  - sgt stands for signed greater than
- How is the if/else structure implemented?
  - Via script – ‘ br i1 %5, label %6, label %7 ’.
  - %6 is the "then" block, %7 is the "else" block.
- How does LLVM determine which return value to use?
  - By assigning 1 or 0 to memeory (%2) in %6 or %7 resp.
  - Then, at merge block %8, it loads the value from %2 and returns it as in the following script:
    %9 = load i32, ptr %2
    ret i32 %9

Part 2 (src/loop.cpp and ir/loop.ll):

- What role does the phi node play?
  - phi merges values from multiple control paths (e.g., from entry and loop back-edge)
  - It tracks loop variables like ‘i’ and accumulators like ‘s’.
- How does LLVM remember the loop variable i across iterations?
  - Via the script :
    ‘ %14 = load i32, ptr %4
    %15 = add nsw i32 %14, 1
    store i32 %15, ptr %4
  - LLVM uses memory allocation (%4) to store and update i.
  - In each iteration, it:
    - loads i,
    - increments i,
    - and stores i again.
- How is the loop exit condition implemented?
  - Via the script:
    %6 = load i32, ptr %4
    %7 = load i32, ptr %2
    %8 = icmp slt i32 %6, %7
    br i1 %8, label %9, label %16
  - LLVM checks ‘i<n’ as follows:
    - If true, branches to loop body
    - If false, goes to return block

Part 3 (loop_cfg.dot and loop_cfg.png): CFG for loop.ll



Bonus Challenge (src/switch.cpp and ir/switch.ll):

- How does LLVM represent the switch statement?
  - Via the script ‘ switch i32 %4, label %7 [i32 1, label %5, i32 2, label %6] ’
  - %4 is the value of x
  - If %4 == 1, control goes to block %5
  - If %4 == 2, control goes to %6
  - Otherwise, it jumps to %7 (the default case)
- This is a direct representation of the switch, not expanded into icmp and br.
- Each case block (%5, %6, %7) stores the result to %2, and unconditionally jumps to %8, where the return value is loaded and returned as follows:
  ‘ %9 = load i32, ptr %2
  ret i32 %9 ’