Experiment #4 Generating optimized MIPS code for SPIM processor.

Assignment Due: 18/04, 11:59pm

- **To do this assignment, you *may* work in teams of two. Team members must be from the same lab group.**
- **Submit the assignment in piazza as a private message:**
  - **Title: Lab Experiment 05_roll01_roll02_(Odd/EVEN)**

ASSIGNMENT DETAILS:

Theoretical background:

- Understand properly, how to divide the generated 3AC in to Basic Blocks. How can you ensure efficient usage of temporary variable in a specific basic block? The lecture slide covering this topic can be found here.

- Understand how to generate proper MIPS code, how human-readable program is translated into a form that a computer can execute, how to run these programs on SPIM, a simulator that executes MIPS programs. Helpful documents can be found here and here.

Execution Steps:

Generate intermediate code (i.e., Three-address code) of every types of statement supported by X as an output of your compiler. You can find the CFG of the programming language X here. You have to implement simple error recovery mechanism, some related discussions can be found here and also here. Print the 3AC code segment properly.

Break the 3AC codes to blocks such that, the block contains no transfer of control instructions except as the last instruction. The lecture slide covering this topic can be found here. Print the blocks properly and also the control flow.

- At the start of each block, we assume all the values are in memory and all the registers are free.
- At the end of each block, we assume all the values of the registers are copied back to the memory.

- In a specific block, try to eliminate unnecessary temporary variables (register usage), eliminate dead variables, and ensure the most efficient usage of temps (registers) and memory locations.
- For each block maintain a register descriptor table. This will contain which register contains which variable, so that you can use the register instead of loading the variable from memory each time. For example, a = x + y

  you can load x and y to registers R1 and R2 and store the result of addition to R3. That mean in the register descriptor table contains

  R1 : x
  R2 : y
  R3 : a

  Now for a statement like b = a * x, you may use registers R1 and R3 and store b in an unoccupied register. If all the registers are occupied, you will remove the contents of a register by issuing a store instruction.

- Generate proper MIPS code for each basic block.
- Use appropriate registers and syntaxes.
- Add appropriate headers and trailers.
- Helpful documents can be found here and here.
- Print the corresponding MIPS code.

RUN the generated MIPS code in SPIM, a simulator that executes MIPS programs