CS3300 Quiz II

Dept of CSE, IIT Madras

Total marks = 24

25 Oct 2017

Read the instructions and questions carefully. You may make any reasonable assumptions that you deem necessary; but state them clearly. There are total three questions (8 marks each). You will need approximately 15 minutes for answering an 8 marks question (plan your time accordingly). For questions with sub-parts, the division for the sub-parts are given in square brackets.

You will get an answer sheet with 8 pages (if you get a answer sheet with fewer pages then ask for a replacement sheet). Leave the first page empty and start from Page#2. Start each question on a new page. Write briefly. **For any question, the answer (including the answers for all the sub-parts) should NOT cross more than two pages.** If the answer for any question is spanning more than two pages, we will strictly ignore the spill-over text. If you scratch/cross some part of the answer, you can use space from the next page.

1. [8] **Syntax Directed Translation**: Give an ambiguity free grammar to recognize hexadecimal number literals of C [3]. Use this grammar to write an SDT scheme to check if a memory address (in hex) begins at word boundary (word size = 4 bytes) [5].

   NOTE: Your procedure should NOT convert the number to decimal.

2. [8] **Type checking and IR Generation**:

   Give a minimal list of instructions of 3-address-codes (TAC), with one line description for each [3] so that the following subset of C language can be translated to this IR: a) all the control structures, b) variables of integer type, array type, or pointer type, c) all the standard assignment statements and expressions, d) no function calls, e) no structs/unions, f) nested blocks and declaration, g) no type-casts, h) no address-of operator. Assume that the IR should also have a statement for variable declarations. You may use 'bOP' to denote generic binary operator and 'uOP' to denote generic unary operator.

   Give rules to do type-checking such that we can only assign among equal types and only integer types are allowed in predicates. – for each IR instruction give the rules to do the checking. [5]

3. [8] **Liveness Analysis and register allocation**

   Say, the liveness analysis algorithm discussed in the class that considers one instruction per block has to be extended to basic-blocks. State how you will compute the `def` and `use` sets for the basic blocks. [2] Give the rules to compute `in` and `out` for each block. [2] Once you compute the `in` and `out` for each basic block, (using the procedure discussed in the class – need to restate here), how will you use it to compute the `in` and `out` for each statement? [3] Compare the precision of the two variants (one discussed in the class, and the one discussed here) and state which is more efficient [0.5] and precise? [0.5]