COMPUTER SCIENCE TRIPOS Part II – 2026 – Paper 9

10 Optimising Compilers (tcg40)

(*a*) Explain the concept of *dominance*, including *dominance frontier*. [2 marks]

(*b*) You are given a RISC-V assembly code program **P**. Arguments are passed in registers `a0` and `a1`. The return value is returned in `a0`.

```
A: mv t0, a0       // t0 = a0 -- Move argument 0
   mv t1, a1       // t1 = a1 -- Move argument 1
   li t3, 0        // t3 = 0, Fallthrough
B: beq t3, t1, G   // Jump to G if `t3 == t1`, else fallthrough.
C: li t5, 2        // t5 = 2
   li t6, 0        // t6 = 0
   remu t4, t0, t5 // t4 = t0 % t5 (unsigned)
   bne t4, t6, E   // Jump to E if `t4 != t6` else fallthrough.
D: j F             // Jump to F
E: add t0, t0, t4  // Fallthrough
F: li t7, 2        // t7 = '2'
   add t3, t3, t7  // t3 = t3 + t7
   j B             // Jump to B.
G: mv a0, t0       // a0 = t0 -- Move result to return register.
   ret             // Return
```

(*i*) Draw the flow graph and dominator tree of **P**. [6 marks]

(*ii*) Assume **P** is run with input `a1 = 1`. How does **P** behave, and how does the execution behaviour of **P** impact the dominance property of **P**? [2 marks]

(*iii*) Turn the program into SSA form by renaming a minimal set of variables and placing a minimal set of PHI-nodes. To get multiple versions of the same register, rename `rX` to `rX0`, `rX1`, `rX2`, ... (e.g. `r0` to `r00`, `r01`, ...). For PHI-nodes, use the instruction format `phi rOUT, rINA (LABELA), rINB (LABELB)` (e.g. `phi r51, r50 (A), r51 (B)`). [6 marks]

(*c*) Compare the result size of a *reaching definitions* analysis (without storage optimisations) with the size it takes to encode reaching definitions via SSA, e.g. in numbers of use-def pairs. [2 marks]

(*d*) Give a worst-case example program that demonstrates the size difference between reaching definitions with and without SSA, and explain what is responsible for this difference. Use C-style pseudocode with switch statements and a special function call `phi` that takes switch labels and variables and returns a variable. The `phi` function mirrors the semantics of a PHI-node in C. [2 marks]