COMPUTER SCIENCE TRIPOS Part II – 2023 – Paper 9

9 Optimising Compilers (tmj32)

Compilers use intermediate representations (IRs) when optimising code.

(*a*) For each of the IRs below, describe its merits for performing dead code elimination, providing a diagram or pseudo-code to illustrate your answer.

(*i*) An abstract syntax tree. [4 marks]

(*ii*) A three-address code. [4 marks]

(*iii*) A stack-based IR. [4 marks]

(*b*) A decompiler is given the following IR code for a function, where registers `r0` and `r1` contain the function arguments and `r0` contains the returned value:

```
foo: mov   r2, #0        // r2 = 0
     cmple r0, #0, end   // Branch to end if r0 <= 0
top: shl   r0, r0, #2    // r0 = r0 << 2
     ldr   r0, [r1, r0]  // r0 = MEM[r1 + r0]
     cmpne r0, #5, chk   // Branch to chk if r0 != 5
inc: add   r2, r2, #1    // r2 = r2 + 1
chk: cmpgt r0, #0, top   // Branch to top if r0 > 0
end: mov   r0, r2        // r0 = r2
     ret                 // Return
```

(*i*) Draw the dominance tree for this code. [2 marks]

(*ii*) Reconstruct a representative source code for this IR code assuming all types are integers or pointers. [6 marks]