

2011 Paper 7 Question 13

Optimising Compilers

- (a) Define what is meant by a variable being *live* at a program point, carefully distinguishing versions based on program structure and I/O behaviour. Discuss these alternatives and their suitability for compilation using words like “decidable” and “safety”. [4 marks]
- (b) Explain the difference between *dead code* and *unreachable code*, mentioning any analysis necessary to remove such code. [3 marks]
- (c) Let q be an expression (whose only free variable is a) which always evaluates to true although deducing this is beyond the power of a given compiler optimiser.

Consider the function

```
int p(int a, int b, int c, int d, int e)
{
  int x = a+b, y = a+c, z=0, r;
  if (a<5) { r = a+x; while (1) continue; z = a+d; }
  else if (q) { r = y; }
  else { r = a+e; }
  return r+z;
}
```

- (i) Give the flowgraph for p (assuming it to have been translated as naïvely as possible). [3 marks]
- (ii) Using your definition of liveness suitable for a compiler from part (a), give the set of variables live on entry to p , comparing this with liveness required by I/O behaviour. [4 marks]
- (iii) Expressing your answer in source code similar to the definition of p , give the code you might expect a good optimiser to achieve. State the optimisations used, where and in what order they were applied, and the set of variables live (under both forms of liveness) on entry to the optimised version of p . [6 marks]