

2005 Paper 5 Question 6

Compiler Construction

- (a) Explain how a parse tree representing an expression can (i) be converted into stack-oriented intermediate code and then (ii) be translated into simple machine code for a register-oriented architecture (e.g. ARM or IA32) on an instruction-by-instruction basis. Also indicate how this code might be improved to remove push-pop pairs introduced by (ii). Your answer need only consider expression forms encountered in the expression:

$$h(a, g(b), c) * 3 + d$$

[12 marks]

- (b) In Java, expressions are evaluated strictly left-to-right. Consider compiling the function `f` in the following Java class definition:

```
class A
{
    static int a,b;
    void f() { ... <<C>> ... }
    int g(int x) { ... a++; ... }
};
```

Indicate what *both* the intermediate code *and* (improved as above) target code might be for <<C>> for the cases where <<C>> is:

- (i) `b = g(7) + a;`
(ii) `b = a + g(7);`
(iii) `b = (-g(7)) + a;`
(iv) `b = a - g(7);`

Comment on any inherent differences in efficiency at both the intermediate code and target code levels.

[8 marks]