2002 Paper 9 Question 7

Optimising Compilers

(*a*) Explain clearly an algorithm to *schedule* assembler code for a given architecture to increase its efficiency; the algorithm must give exactly the same instructions as in the original code, possibly in a different order. You may assume the hardware is interlocked so that NOPs are not required. Your answer should state the range over which scheduling takes place and, if this is smaller than a procedure, what actions can sensibly be taken at the boundary between separately scheduled sections of code. [10 marks]

(*b*) Consider an architecture which is ARM-like—the first operand is the destination register except in the case of stores. All operations take one cycle, with the exceptions that (*i*) using the value resulting from an immediately previous load will cause a single-cycle stall, as does (*ii*) the second of two successive store instructions. Use your algorithm to schedule the following code, noting briefly, for each instruction emitted, why your algorithm has chosen this instruction over other candidates.

```
ldr  r3,[r1,#0]
str  r3,[r2,#0]
ldr  r4,[r1,#4]
str  r4,[r2,#4]
add  r5,r4,#4
```

[6 marks]

(*c*) Give with brief explanation the number of possible valid schedulings of the following code (not just the one produced by your algorithm):

```
add  r3,r1,r2
or   r4,r1,r2
sub  r5,r3,r4
xor  r4,r1,r3
```

[4 marks]