12 Optimising Compilers (TMJ)

The following three-address pseudo-code shows a compiler's intermediate representation of a basic block, with destination registers as the first operand of each non-store instruction. The compiler has already determined that there are no aliases between instructions accessing memory (ie: instructions 1, 2, 6, 8, 9 all access different memory locations).

```
1:  load  $1, 0($5)
2:  load  $2, 8($5)
3:  sub   $3, $1, $2
4:  add   $4, $1, $2
5:  add   $4, $4, #8
6:  store $4, 0($3)
7:  and   $2, $4, #4095
8:  load  $1, 16($5)
9:  store $2, 0($1)
```

(*a*) What are the two goals of instruction scheduling? [2 marks]

(*b*) With reference to the above code, describe the constraints on reordering instructions. [4 marks]

(*c*) Briefly describe an instruction scheduling algorithm, then apply it to this code when targeting a simple 5-stage pipelined processor (IF, RF, EX, MEM, WB), showing the time saving against the original schedule. You may assume each instruction takes one cycle to execute and that the hardware supports delayed loads with feed-forwarding and interlocks. [8 marks]

(*d*) How does register allocation interact with instruction scheduling? [3 marks]

(*e*) Assuming that all registers are dead at the end of this code, are more registers required to schedule this basic block without any stalls? Justify your answer. [3 marks]