12 Optimising Compilers (tmj32)

The following code for a function is given to a compiler for optimisation, where arguments are passed in through variables `arg0`, `arg1` and `arg2` and the result is returned through variable `res0`. Describe five optimisations that the compiler can carry out, explaining for each the analysis required, the actual transformation, how it impacts the code and showing the resulting code after all optimisations have been performed.

```
entry foo
      mov t0, #0x1000      // t0 = 0x1000
      mov t1, #0           // t1 = 0
      mov t2, #0           // t2 = 0
      mul t3, arg0, #2     // t3 = arg0 * 2
      mul t4, arg1, #4     // t4 = arg1 * 4
      mov t5, #&lab4       // t5 = address of lab4
      sti t0, t5           // *t0 = t5
      b lab7               // branch to lab7
lab1: mov t6, #1           // t6 = 1
      cmpeq t1, arg2, lab3 // branch to lab3 if t1 == arg2
      mul t7, t4, t4       // t7 = t4 * t4
      cmpeq t1, t7, lab5   // branch to lab5 if t1 == t7
      b lab6               // branch to lab6
lab2: mov t8, #&lab5       // t8 = address of lab5
      sti t0, t8           // *t0 = t8
      b lab5               // branch to lab5
lab3: mul t9, t4, t4       // t9 = t4 * t4
      add t2, t2, t9       // t2 = t2 + t9
      add t6, t6, #1       // t6 = t6 + 1
      ldi t10, t0          // t10 = *t0
      bi t10               // branch to address in t10
lab4: add t2, t2, t1       // t2 = t2 + t1
      b lab6               // branch to lab6
lab5: add t2, t2, t6       // t2 = t2 + t6
lab6: add t1, t1, #1       // t1 = t1 + 1
lab7: cmpne t1, t3, lab1   // branch to lab1 if t1 != t3
      mul res0, t2, #4     // res0 = t2 * 4
      mov t11, #&lab1      // t11 = address of lab1
      sti t0, t11          // *t0 = t11
      exit
```

[4 marks each]