```
/*****************************************/
/* Introduction to Compiler Construction */
/*                                       */
/* Christoph Kirsch                      */
/* University of Salzburg                */
/*                                       */
/* boolean expressions                   */
/*****************************************/

#include <stdlib.h>
#include <stdio.h>

#define boolean int

int x;
int y;

boolean b1;
boolean b2;
boolean b3;
boolean b4;
boolean b5;

main() {
  x = 0;
  y = 1;

  b1 = x < y;
  // LDW 1, 28, -4
  // LDW 2, 28, -8
  // CMP 1, 1, 2
  // BGE 1, 0, 3 fixup in fls
  // ADDI 1, 0, 1 or MOVI 1, 0, 1
  // BR 0, 0, 2
  // ADDI 1, 0, 0 or MOVI 1, 0, 0
  // STW 1, 28, -12

  b2 = b1;
  // not optimized:
  // LDW 1, 28, -12
  // BEQ 1, 0, 3 fixup in fls
  // ADDI 1, 0, 1 or MOVI 1, 0, 1
  // BR 0, 0, 2
  // ADDI 1, 0, 0 or MOVI 1, 0, 0
  // STW 1, 28, -16
  //
  // optimized (requires type checking!):
  // LDW 1, 28, -12
  // STW 1, 28, -16

  b3 = b1 && b2;
  // LDW 1, 28, -12
  // BEQ 1, 0, 5 fixup in fls
  // LDW 1, 28, -16
  // BEQ 1, 0, 3 fixup in fls
  // ADDI 1, 0, 1 or MOVI 1, 0, 1
  // BR 0, 0, 2
  // ADDI 1, 0, 0 or MOVI 1, 0, 0
  // STW 1, 28, -20
```

```
    b3 = b1 || b2;
    // LDW 1, 28, -12
    // BNE 1, 0, 3 fixup in tru
    // LDW 1, 28, -16
    // BEQ 1, 0, 3 fixup in fls
    // ADDI 1, 0, 1 or MOVI 1, 0, 1
    // BR 0, 0, 2
    // ADDI 1, 0, 0 or MOVI 1, 0, 0
    // STW 1, 28, -20

    b4 = !b1;
    // LDW 1, 28, -12
    // BNE 1, 0, 3 fixup in fls
    // ADDI 1, 0, 1 or MOVI 1, 0, 1
    // BR 0, 0, 2
    // ADDI 1, 0, 0 or MOVI 1, 0, 0
    // STW 1, 28, -24

    b5 = (b1 || b2) && (b3 || b4);
    // LDW 1, 28, -12
    // BNE 1, 0, 3 fixup in tru
    // LDW 1, 28, -16
    // BEQ 1, 0, 7 fixup in fls
    // LDW 1, 28, -20
    // BNE 1, 0, 3 fixup in tru
    // LDW 1, 28, -24
    // BEQ 1, 0, 3 fixup in fls merged with previous BEQ
    // ADDI 1, 0, 1 or MOVI 1, 0, 1
    // BR 0, 0, 2
    // ADDI 1, 0, 0 or MOVI 1, 0, 0
    // STW 1, 28, -28

    // comparison operators only require a CMP instruction each
    // otherwise same as above
}
```