FP8-17: Software Programmable Signal Processing Platform Analysis Exercises for Episode 5

Andrzej Wąsowski

5 April 2007

Exercise 5.1 (Appel Ex. 10.1) Perform flow analysis on the following program:

$_1$ $m \leftarrow 0$
$_2$ $v \leftarrow 0$
$_3$ if $v \geq n$ goto 15
$_4$ $r \leftarrow v$
$_5$ $s \leftarrow 0$
$_6$ if $r < n$ goto 9
$_7$ $v \leftarrow v + 1$
$_8$ goto 3
$_9$ $x \leftarrow M[r]$
$_{10}$ $s \leftarrow s + x$
$_{11}$ if $s \leq m$ goto 13
$_{12}$ $m \leftarrow s$
$_{13}$ $r \leftarrow r + 1$
$_{14}$ goto 6
$_{15}$ return m

a. Draw the control-flow graph

b. Calculate live-in and live-out at each statement.

c. Construct the register interference graph.

d. What is the smallest k such that this code can be run on k registers without spilling? Do not prove it formally, but try to give an example of efficient coloring.

Exercise 5.2 Compiler (cl6x) produces a warning message while compiling the following program.

```
int uninitialized(void) {
  int c;
  while (c<100)
    c += 7;
 return c;
}
```

Can you explain what part of the compiler is producing this warning? Can you explain the mechanism that leads to its detection?

Exercise 5.3 Implement saturating substraction in ANSI C. Then identify the corresponding intrinsic and use it in place of your implementation. Generate assembly from both files using cl6x. Compare the output.

Exercise 5.4 (subproject, continuation of 3.6) Analyze the functions in your project with respect to applicability of inlining. Begin with reading conditions for inlining functions on p. 2-42 [SPRU187,*Optimizing Compiler User Guide* . Then take the C code of your project and classify each function, whether it can be inlined or not. Focus on computation intensive functions, called many times (you may use a profiler to identify these).

Find functions that you think are suitable for inling and will benefit the program if inlined (would bring a speed up, without excessive growth of code size). Use the keywords (`inline`) to control inlining. Analyse the generated code to see whether inling was applied in the places you expect it to be.

If you find a function that should be inlined, but it cannot be inlined due to inlining restrictions (p. 2-42), consider rewriting the code so that it can meet the requirements for inlining.