Andrzej Wąsowski. *Software Programmable DSP Platform Analysis.*

Exercise Set 4

19 April 2005

Exercise 4.1 (based on Appel ex.7.1-2) Translate following expressions to IR trees. Do not feel forced to solve all of them, but perhaps focus on these that appear difficult.

a. `a+5`

b. `b[i]+1`

c. `a<b`, which should be implemented by making ESEQ whose left-hand side moves a 1 or 0 into some newly defined temporary, and whose right-hand side is the temporary.

d. `if a then b else c`, where `a` is an integer variable (true if $\neq 0$); this should also be translated using ESEQ.

e. `a := x+y`, which should be translated with an EXP at the top.

f. `while a>0 do a:=a-1`

g. `(a + b < 2) && c == 0`

h. `if x-2 > 3 then 1 else z == x`

Exercise 4.2 Compile the following program using cl6x to an assembly file (recommended options: `-k -s -O0`). Identify a translation scheme used for while loops by cl6x in this mode (it is a variation of two schemes presented in the lecture).

```
int testwhile(int argc, const char * argv[]) {
  volatile int iport;
  volatile int oport;

  while (iport)
    oport = iport*2;

  return 0;
}
```

Exercise 4.3 If you are brave enough, compile the program below and analyze the assembly output. Try to compare the instruction selection we have made, with the one made by the compiler (this may be difficult, as we do not have a complete view on internals of this compiler).

```
void selection(int a[]) {
  volatile int i;
  volatile int x;

  a[i*4] = x;
}
```

Exercise 4.4 (Appel ex.9.1) For each of the following expressions, draw the tree and generate *Jouette*-machine instructions (p. 192) using Maximal Munch. Circle the tiles (as in Figure 9.2), but number them *in the order that they are munched*, and show the sequence of *Jouette* instructions that results.

a. MOVE(MEM(+(+(CONST$_{1000}$,MEM(TEMP$_{x}$)),TEMP$_{fp}$)), CONST$_{0}$)

b. MUL(CONST$_{5}$,MEM(CONST$_{100}$)).