Compilation – Final exam

M1 Informatique Fondamentale, ENS Lyon
Jeudi 10 janvier 2013

Duration: 3 hours. All documents are allowed.
This exam consists of 4 independent exercises.

Exercise 1. *Syntax-directed translation*

Consider the following program:

```
typedef struct {
  int im;
  int re;
} complex_t;

void foo()
{
  complex_t[2] tab;
  int i;

//point 1
  for(i=0; i<2; i++)
    {
      tab[i].im = i;
      tab[i].re = i+1;
    }
}

void main()
{
  foo();
}
```

Q1) Without details, draw the stack after the execution of the prelude of foo (at point 1).
One will suppose that main() has no activation record, and that the activation record of foo() starts at byte 255.

Q2) With details, give $[\![\text{complex\_t[2]}]\!]^{\texttt{temp}}$.
One will keep the temporaries current_cell, counter, etc, of the translation rules without trying to replace them by their value. As well, one will write as it is the constructions "macro" as SP = SP - 2.

Q3) With details, give $[\![\text{tab[1].re} = 2]\!]_\rho$.
One will assume index(complex_t,re) = 1. On the course slide with the relevant rules, one must read "- index", in place of "+ index".

T.S.V.P.

Exercice 2. *SSA Form*

The compilation of a `for` loop produces the following intermediate code:

```
  r9 = 0
for_0:
  r22 = 10
  cjump r9 < r22 --> lower_0
  r23 = 0
  jump not_lower_0
lower_0:
  r23 = 1
not_lower_0:
  r24 = 1
  cjump r23 == r24 --> continue_for_0
  jump end_for_0
continue_for_0:
  r9 = r9 + 1
  jump for_0
end_for_0:
```

Q4) Without details, draw the CFG and the dominance tree.

Q5) Without details, give the sets DF_{local} and DF_{up} for each block.

Q6) Recall that the dominance frontier of a block B is given by:

$$DF(B) = DF_{\text{local}}(B) \cup \bigcup_{B \text{ idom } C} DF_{\text{up}}(C)$$

Without details, give the dominance frontier for each block. Why does the union traverse only the blocks C immediately dominated?

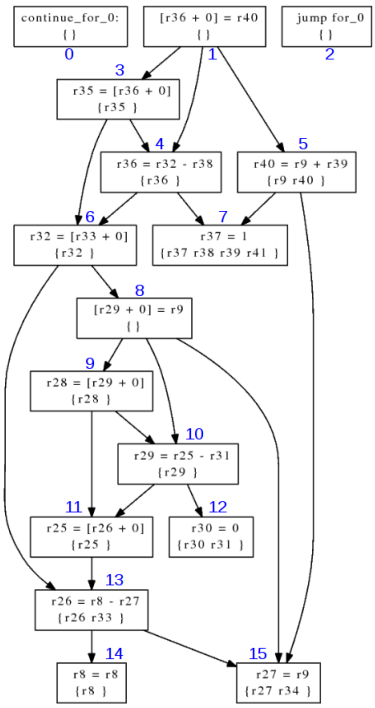
Q7) With details, turn the program into SSA form.

Q8) With details, get the obtained program out of SSA.

T.S.V.P.

Exercise 3. *Scheduling*

Compiling the program given at exercise 1 produce the following DAG for the loop body:



Q9) Why is there an edge from 1 to 3?

Q10) We wish to apply the naive heuristic seen in the course. Without details, give a decomposition of the DAG in subtrees.

Q11) Do we need to produce code for the leaves which were shared nodes in the DAG? How to handle them?

Q12) With an appropriate justification, show in which order the trees can be evaluated.

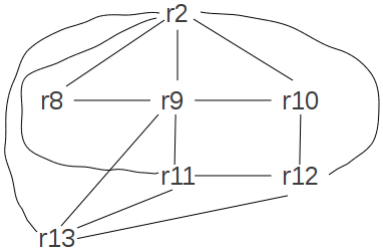
Q13) We consider the tree whose root is 6. With details, apply the Sethi-Ullman algorithm to produce the code.

Q14) Without details, draw the lifetime intervals on the obtained code. Give the register pressure. Compare to the values computed in 13). Comment.

Q15) Here, the temporary r2 is live-out. What additional action do we need to do during the code generation?

Exercice 4. *Register allocation*

After scheduling the machine code, we get the following interference graph. We assume the node r2 to be precolored with the register r2.



Q16) What does a node represent? Why precoloring the node r2?

Q17) Find a register allocation by applying the Chaitin algorithm with $K = 3$ register, give all the details.

Q18) How/where to allocate the spilled variables?