CSE 5317 Spring 2010: Assignment 2

Assigned: 30 Mar 2010
Due: 8 Apr 2010, 2:00pm

Please hand in a hard copy at the beginning of class on the due date, or email it to me before the deadline. Type or write neatly.

1 IR and liveness

Consider the following program, which counts primes from 2 to $n$ using the sieve method:

```c
for (i = 0; i < n; i++)
    a[i] = true;
count = 0;
for (i = 2; i*i < n; i = i + 1)
    if (a[i]) {
        count = count + 1;
        for (j = 2*i; j < n; j = j + i)
            a[j] = false
    }
```

In this program, $a$ is an array of integers. At the end of the program, the $count$ is the only live variable.

(a) Write three-address code for this program. Use the following operations:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x = y \ OP z$</td>
<td>Binary operation</td>
</tr>
<tr>
<td>$x = \ OP y$</td>
<td>Unary operation</td>
</tr>
<tr>
<td>$x = y$</td>
<td>Copy</td>
</tr>
<tr>
<td>$x = k$</td>
<td>Load constant $k$</td>
</tr>
<tr>
<td>$x = [y]$</td>
<td>Load from address $y$ into $x$</td>
</tr>
<tr>
<td>$[x] = y$</td>
<td>Store $y$ at address $x$</td>
</tr>
<tr>
<td>$L:$</td>
<td>A label</td>
</tr>
<tr>
<td>$\text{tjump } x \ L$</td>
<td>Jump to $L$ if $x$ is non-zero</td>
</tr>
<tr>
<td>$\text{fjump } x \ L$</td>
<td>Jump to $L$ if $x$ is zero</td>
</tr>
<tr>
<td>$\text{jump } L$</td>
<td>Jump to label $L$ unconditionally</td>
</tr>
</tbody>
</table>

$x$, $y$, and $z$ above are temporaries. $k$ is an integer constant.

All local variables should be compiled to temporaries. Booleans can be represented as the integers 0 and 1. Arrays are allocated on the heap (i.e., the local variable $a$ contains a pointer to the array data). You do not need to implement bounds checks or null checks for array accesses.

(b) Construct the control flow graph from the IR in (a).

(c) Compute the live variables at each program point in the CFG.
2 Dataflow

In this problem, we will design a dataflow analysis that computes whether an expression is partially available. An expression is partially available if it has been computed on some path from the entry and has not been modified subsequently.

(a) Is this a forward or backward analysis? Is this a may or must analysis?

(b) Define the partial order and the meet operator for this lattice.

(c) Define the transfer functions for each of the following three-address instructions:
   - \( x = y \ OP \ z \)
   - \( x = [y] \)
   - \( [x] = y \)
   - \( x = y \)
   - \( x = k \)

(d) Using the lattice and the transfer functions of the previous parts, show how the dataflow analysis works for the following program:

\[
\begin{align*}
i &= 0 \\
s &= 0 \\
\text{TOP:} & \\
z &= i < 10 \\
fjump \ z \ \text{END} \\
v &= a + i \\
x &= [v] \\
w &= a + 40 \\
y &= [w] \\
t &= s + x \\
s &= t + y \\
jump \ \text{TOP} \\
\text{END:} & \\
u &= a + 40 \\
[u] &= s
\end{align*}
\]

Note that the transfer functions for jump instructions do not change the flow information, although the meet operations may change the flow information.