Optimizing Compilers	**Spring 2014**

Homework 2

February 18, 2014	Alejandro Carbonara (auc), Aram Ebtekar (aebtekar)

2

Our framework is entirely contained in dataflow.h and dataflow.cpp. In order to run our framework, you need to implement the following things:

First you have to decide whether you are doing a forward or backwards pass. Depending on which, you will be using the forwardSearch or backwardSearch function.

Then, to define the domain you create a lattice class. The lattice class requires a list of the names of things we are taking subsets of. The lattice uses a bitvector to represent subsets of that group, and the list tells us the name of each item in the group. You also pass a boolean to say whether you merge with intersection or union.

You then implement the transition function over bitvectors. You have to define your transition function over single instructions. Note that for both of our passes, we created a mapping between instructions and strings for our transfer function.

For testing purposes, we print out the ll code intermixed with the computation at the program points. We ran our tests over simple.c and loop.c.

3.1

See the attached diagram.

3.2

i

The following instructions are invariant:

$$r = q + 5$$

$$m = y + 7$$

Note that the algorithm would not normally pick up $m = y + 7$ because y is not constant in the loop, though it will always be 5 at that point of the loop.

ii

We can move $r = q + 5$ to the pre-header.

1.

```
x=x+3        (x+3)
if(y>5)
             (z+x)
```

```
z=z+x    (z+x)        y=y-5      (y-5), (z+x)
x=4                   z=z+x
```

```
return z
```

2.

```
x=x+3
t=z+x
if(y>5)
```

```
z=t                   y=y-5
x=4                   z=t
```

```
return z
```

3.

```
x=x+3
z=z+x
if(y>5)
```

```
x=4                   y=y-5
```

```
return z
```