

Homework assignments are to be completed individually. Hand-written submissions are fine, but they must be readable. Due at the beginning of class. Total points: 100, 3.3% of course grade

1. [15 Points] Induction Variables. Perform induction variable detection, strength reduction and induction variable elimination on Figure 9.5 in book.
2. [15 Points] SSA. Translate Figure 9.3 into SSA. Perform copy propagation and dead code elimination on SSA.
3. [20 Points] Value Numbering.

```

    a = read()
    b = read()
    z = read()
    w = read()
    x = a - b
    y = a + b
    j = 0
loop:
    z = a * b
    w = z
    if (w==0) goto L2
    w = a * b
L2:
    x = a + b
    j = j + 1
    if (z < j) goto loop
print w,x,y,z

```

- (a) Perform pessimistic global value numbering.
- (b) Transform the SSA code based on the pessimistic global value numbering results. After doing so, what optimization on SSA (hint problem 2 in this HW) should be performed? Perform it.
- (c) Rewrite the resulting code in 3-address code.

**Answer:**

For pessimistic global value numbering, we need to translate the code to SSA.

```

a0 = read()

```

```
b0 = read()
z0 = read()
w0 = read()
x0 = a0 - b0
y0 = a0 + b0
j0 = 0
loop:
  z1 = phi_n(z0,z2)
  x1 = phi_n(x0,x2)
  j1 = phi_n(j0,j2)
  w1 = phi_n(w0,w2)
  z2 = a0 * b0
  w2 = z2
  if (w2==0) goto L2
  w3 = a0 * b0
L2:
  w4 = phi_m(w3,w2)
  x2 = a0 + b0
  j2 = j1 + 1
  if (z2<j2) goto loop
print w4,x2,y0,z2
```

(a) Pessimistic global value numbering.

Variable	Init Value Num	After visiting stmts
a0	#1	#1
b0	#2	#2
z0	#3	#3
w0	#4	#4
x0	#5	#1 - #2 $\rightarrow$ #18
y0	#6	#1 + #2 $\rightarrow$ #19
j0	#7	#7
z1	#8	phi_n ( #3, #12 ) $\rightarrow$ #20
x1	#9	phi_n ( #16, #116 ) $\rightarrow$ #21
j1	#10	phi_n ( #7, #17 ) $\rightarrow$ #22
w1	#11	phi_n ( #4, #13 ) $\rightarrow$ #23
z2	#12	#1 * #2 $\rightarrow$ #24
w2	#13	#24
w3	#14	#1 * #2 $\rightarrow$ #24
w4	#15	phi_n ( #24, #24 ) $\rightarrow$ #24
x2	#16	#1 + #2 $\rightarrow$ #19
j2	#17	#22 + 1 $\rightarrow$ #25

We end up with the following congruence classes:

P0 = { a0 }  
 P1 = { b0 }  
 P2 = { z0 }  
 P3 = { w0 }  
 P5 = { x0 }  
 P6 = { y0, x2 }  
 P7 = { j0 }  
 P8 = { z1 }  
 P9 = { x1 }  
 P10 = { j1 }  
 P11 = { w1 }  
 P12 = { z2, w2, w3, w4 }  
 P13 = { j2 }

After value numbering, copy propagation, and dead code elimination, we can rewrite the code as follows:

```

a0 = read()
b0 = read()
z0 = read()
w0 = read()
x0 = a0 - b0
y0 = a0 + b0
j0 = 0
  
```

```

loop:
  z1 = phi_n(z0,z2)
  x1 = phi_n(x0,y0)
  j1 = phi_n(j0,j2)
  w1 = phi_n(w0,z2)
  z2 = a0 * b0

  if (w2==0) goto L2

```

L2:

```

  j2 = j1 + 1
  if (z2<j2) goto loop
print z2,y0,y0,z2

```

(c) Rewrite in 3-address code. The important points are that you can't remove the SSA numbering of variables when you convert to three-address code and you need to put a copy on the incoming path associated with each entry in a phi function. We depend on the register allocation to perform coalescing to remove copies.

```

  a0 = read()
  b0 = read()
  z0 = read()
  w0 = read()
  x0 = a0 - b0
  y0 = a0 + b0
  j0 = 0
  z1 = z0
  x1 = x0
  j1 = j0
  w1 = w0
loop:
  z2 = a0 * b0
  if (w2==0) goto L2

```

L2:

```

  j2 = j1 + 1
  z1 = z2
  x1 = y0
  j1 = j2
  w1 = z2
  if (z2<j2) goto loop
print z2,y0,y0,z2

```

4. [20 points] Data dependence analysis and unimodular transformations

```
for (i=0; i<N; i++) {
  for (j=3; j<M; j++) {
    A[i+1] = B[ j-3][ i+2 ] - 99;
    B[ j-1][ i ] = sin(i * j);
  }
}
```

- (a) For the above program, what is the direction vector for the output dependences between writes to A[i]? (Hint: Recall that  $(*, <)$ ,  $(*, =)$ , and  $(*, >)$  are not legal dependence vectors.)
- (b) For the above program, what is the distance vector for the flow dependence?
- (c) What is the unimodular transformation matrix that specifies a permutation of the  $i$  and  $j$  loops in the program for problem 4?
- (d) Is the problem 4 loop fully permutable? Why or why not?
- (e) Which loop carries each of the dependences? What is a possible parallelization strategy for the above loop?

5. [15 points] Loop Fission and the Kelly and Pugh Transformation Framework

- (a) Show whether loop fission is legal or illegal for the following program using the K&P transformation framework.

```
for (i=0; i<N; i++) {
  A[ i ] = ... ;
  ... = A[ i + 1 ];
}
```

- (b) Show whether loop fission is legal or illegal for the following program using the K&P transformation framework.

```
for (i=0; i<N; i++) {
  A[ i ] = ... ;
  ... = A[ i - 2 ];
}
```

6. [15 points] Loop transformations and Fourier Motzkin. Skew the loop to make it permutable and then permute the loop. Write the transformed code.

```
for (i=1; i<N; i++) {
  for (j=1; j<(i+1); j++) {
```

```

        A[ i ][ j ] = A[i-2,j] + A[i-1,j+1];
    }
}

```

original bounds

$$\begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 1 & -1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} i \\ j \end{bmatrix} \geq \begin{bmatrix} 0 \\ 1-N \\ 0 \\ 1 \end{bmatrix}$$

transformation  $T = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$  selected to modify dependence vector  $(1, -1)$  so that loop is permutable

$$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} i \\ j \end{bmatrix} = \begin{bmatrix} i' \\ j' \end{bmatrix}$$

$$T^{-1} = \begin{bmatrix} 0 & 1 \\ 1 & -1 \end{bmatrix}$$

Transformed iteration space

$$\begin{bmatrix} 0 & 1 \\ 0 & -1 \\ -1 & 2 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} i' \\ j' \end{bmatrix} \geq \begin{bmatrix} 0 \\ 1-N \\ 0 \\ 1 \end{bmatrix}$$

array access  $A[i][j]$

$$F = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$F' = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & -1 \end{bmatrix}$$

New bounds after projecting out  $j'$  using Fourier Motzkin

$$\begin{bmatrix} -1 & 0 \\ 1 & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} i' \\ j' \end{bmatrix} \geq \begin{bmatrix} 2-2N \\ 2 \\ 1 \end{bmatrix}$$

Transformed code

```

for (i'=1; i'<=N; i'++) {
    for (j'=max(0,i'/2); j'<=min(N-1,i'-1); j'++) {
        A[ j' ][ i'-j' ] = A[j'-2,i'-j'] + A[j'-1,i'-j'+1];
    }
}

```