C231 Spring 1998 Midterm Solutions

- 1) Optimization Order
  - A) Constant Propagation enables Constant folding

    ```
    max := 256
    ...
    total:= 5+k+max

    (propagating 256 down, allows the second expression to be folded:)
    total:= 261+k
    ```

  - B) Dead code elimination is used throughout the optimization process
- 2) Data Dependence Analysis
  - A)

    ```
    3i = 2i + 1
    1 <= i <= 100
    ```

  - B)

    ```
    3i = 2j + 1, 1 <= i,j <= 100
    ```

  - C)

    ```
    3i - 2i = 1
    GCD(3, -2) = 1 and 1/1
    So it tells us nothing, there may be a dependency
    ```

- 3) Loops
  - A) (B4 - B2) = { B2, B3, B4 }
    (B5 - B1) = { B1, B2, B3, B4, B5 }
  - B) There one maximal scc: {B1, B2, B3, B4, B5 }
  - C) Natural Loops do not preserve nesting information
    Tarjan intervals include minimal irreducible subgraphs, maximal acyclic structures, and loops

- 4) Data Flow Analysis
  - A) Efficient time wise, easily updateable
  - B) Simple, handle irreducible graphs automatically
  - C) Very busy expressions:
    * 1) GEN(B) = set of expressions in E evaluated in block B before its operands are def'd in the block
      KILL(B) = set of expressions in E that are killed in block B
      OUT(B) = set of expressions in E that are "very busy" at the point at the end of block B
      IN(B) = set of expressions in E that have been evaluated below the starting point of block B but whose operands have not yet been defined ("very busy" at the start of B)
    * 2) OUT(B) = INT(in(S)) where INT is the Intersection, S is the set of all successors of B
      IN(B) = GEN(B) UNI (OUT(B) - KILL(B)) where UNI is the union
    * 3) yes, e.g. IN(A) UNI IN(B) = IN(A UNI B)
- 5) Control Dependences
  - A) B1,B5 can be executed in parallel
    B2,B4 can be executed in parallel
    Edges included are the control dep. edges other edges have been left out

```
entry
|   \
B5   B1
    /  \
   B4   B2
        |
        B3
```

- 6) SSA Form
  - A) pro: SSA enables powerful and fast optimizations
    con: There are no phi constructs in low level IR or in final generated code so they must be translated back before the code gen stage
  - B) False, there's a unique DEFINITION that reaches each use of a variable
  - C) Y is control dependent on X in CFG G iff X is an element of Y's dominance frontier in Reverse(G)

- 7) Value Numbering
  - A)

```
read(e,b)
t1 := b * e
f := t1
g := t1
a := 2
if (t1 > 0) then...
```

  - B) Yes

```
read(a,b)
e := a
t1 := b * e
f := t1
g := a * b
a := 2
if (t1 > 0) then...
```

  - C) No
- 8) Reassociation and Loop-invariant code motion
  - A)

```
b + (( i-c ) * s + ( j-c ) * s
= b + ( i*s - c*s +  j*s - c*s ) distrib prop. of * over -
= b + ( (-1)*c*s + (-1)*c*s +  i*s + j*s )  commutative prop. of +
= b + ( (-2)*c*s ) + ( i*s + j*s ) distributive prop of * over +
```

  - B) J loop: i*s, (b+ (-2)*c*s); Both loops: (b + (-2)*c*s)

- 9) Aliasing
  - A) Multiple names for the same storage location
  - B) Procedure calls, pointers
- 10) Irreducibility
  - A) A graph with multiple entries into a loop (and not all thru loop header):

```
entry
  |
--B1
| |
| B2<--
| |   |
->B3  |
  |   |
  B4 --
  |
exit
```