Lecture 2

Local Optimizations

I. Basic blocks/Flow graphs

II. Abstraction 1: DAG

III. Abstraction 2: Value numbering

I. Basic Blocks & Flow Graphs

- **What is**
  - a basic block?
  - a flow graph?
- **How do we restructure a sequential list of instructions into a flow graph of basic blocks?**
  - ALSU pp. 529-531
- **Reachability of basic blocks**

```
if x {                    bfls r1, L1
   …                      …
   return;                ret
                          jmp L2
} else {              L1: …
   …
}                     L2: …
```

II. Local Optimizations

- **Common subexpression elimination**
  – array expressions
  – field access in records
  – access to parameters

Graph Abstractions

- **Example 1: an expression**
  `a+a*(b-c)+(b-c)*d`

- **ALSU pp. 359-362**

How well do DAGs hold up across statements?

- **Example 2**

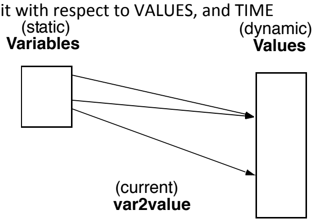
```
a = b+c;
b = a-d;
c = b+c;
d = a-d;
```

Critique of DAGs

- **Cause of problems**
  - Assignment statements
  - Value of variable depends on TIME
- **How to fix problem?**
  - build graph in order of execution
  - attach variable name to latest value
- **Final graph created is not very interesting**
  - Key: variable->value mapping across time
  - loses appeal of abstraction

III. Value Number: Another Abstraction

- John Cocke & Jack Schwartz in unpublished book: “Programming Languages and their Compilers”, (1970) (*ALSU pp. 360-362)*
- More explicit with respect to VALUES, and TIME

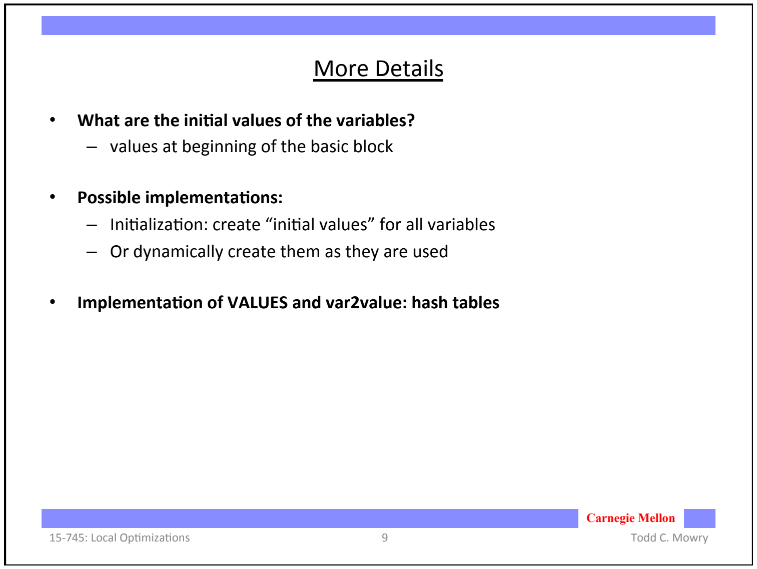


- **each value has its own “number”**
  - **common subexpression means same value number**
- var2value: current map of variable to value
  - used to determine the value number of current expression

r1 + r2 => var2value(r1)+var2value(r2)

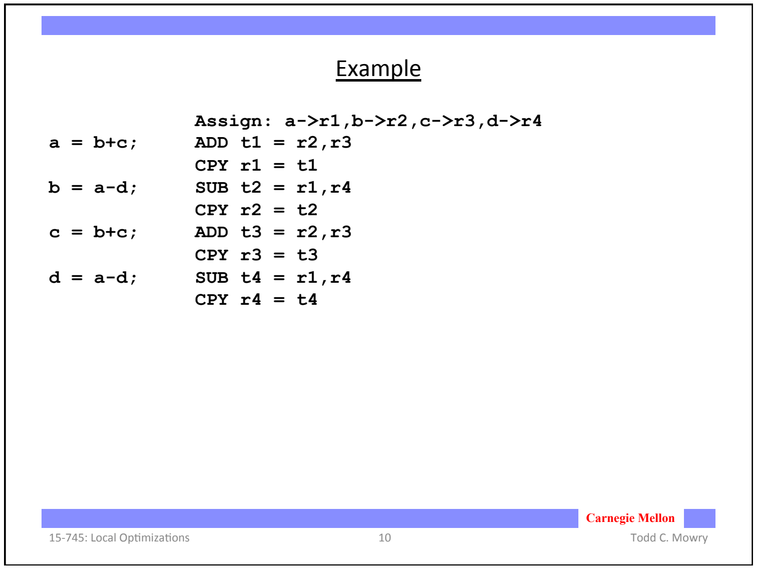
Algorithm

```
Data structure:
    VALUES = Table of
      expression
      var            (temporary holding variable)

For each instruction (dst = op src1 src2) in execution order

  IF [OP var2value(src1) var2value(src2)] is in VALUES
    v = the index of expression
    Replace instruction with CPY dst = VALUES[v].var
  ELSE
    Add
      expression = [OP var2value(src1) var2value(src2)]
      var        = dst
    to VALUES
    v = index of new entry

  set_var2value (dst, v)
```



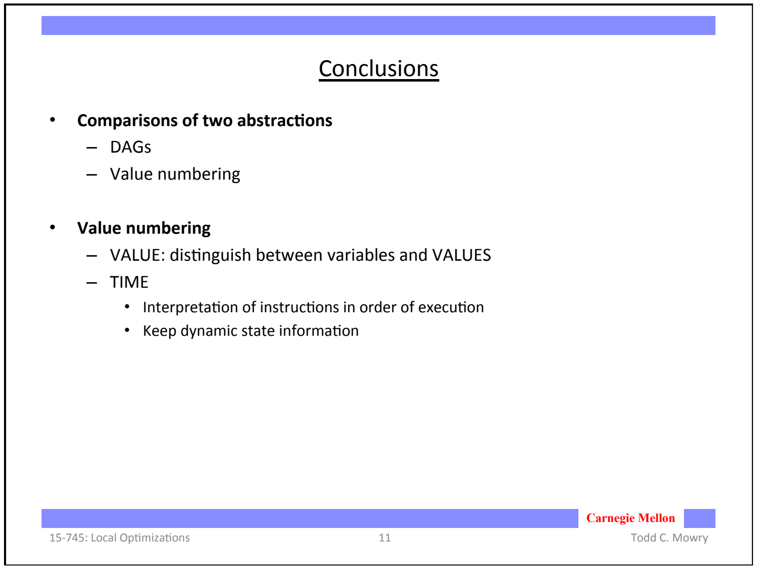
More Details

- **What are the initial values of the variables?**
  - values at beginning of the basic block
- **Possible implementations:**
  - Initialization: create “initial values” for all variables
  - Or dynamically create them as they are used
- **Implementation of VALUES and var2value: hash tables**



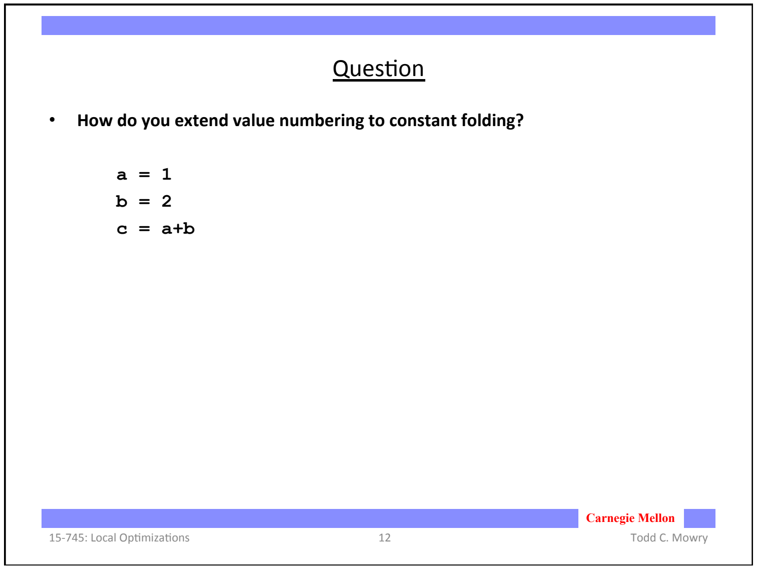
Example

```
                 Assign: a->r1,b->r2,c->r3,d->r4
a = b+c;         ADD t1 = r2,r3
                 CPY r1 = t1
b = a-d;         SUB t2 = r1,r4
                 CPY r2 = t2
c = b+c;         ADD t3 = r2,r3
                 CPY r3 = t3
d = a-d;         SUB t4 = r1,r4
                 CPY r4 = t4
```



Conclusions

- **Comparisons of two abstractions**
  - DAGs
  - Value numbering
- **Value numbering**
  - VALUE: distinguish between variables and VALUES
  - TIME
    - Interpretation of instructions in order of execution
    - Keep dynamic state information



Question

- **How do you extend value numbering to constant folding?**

```
a = 1
b = 2
c = a+b
```