Adventures in
Dataflow Analysis

CSE 401 Section 9-ish
Kory Watson, Aaron Johnston,
Miya Natsuhara, Sam Wolfson

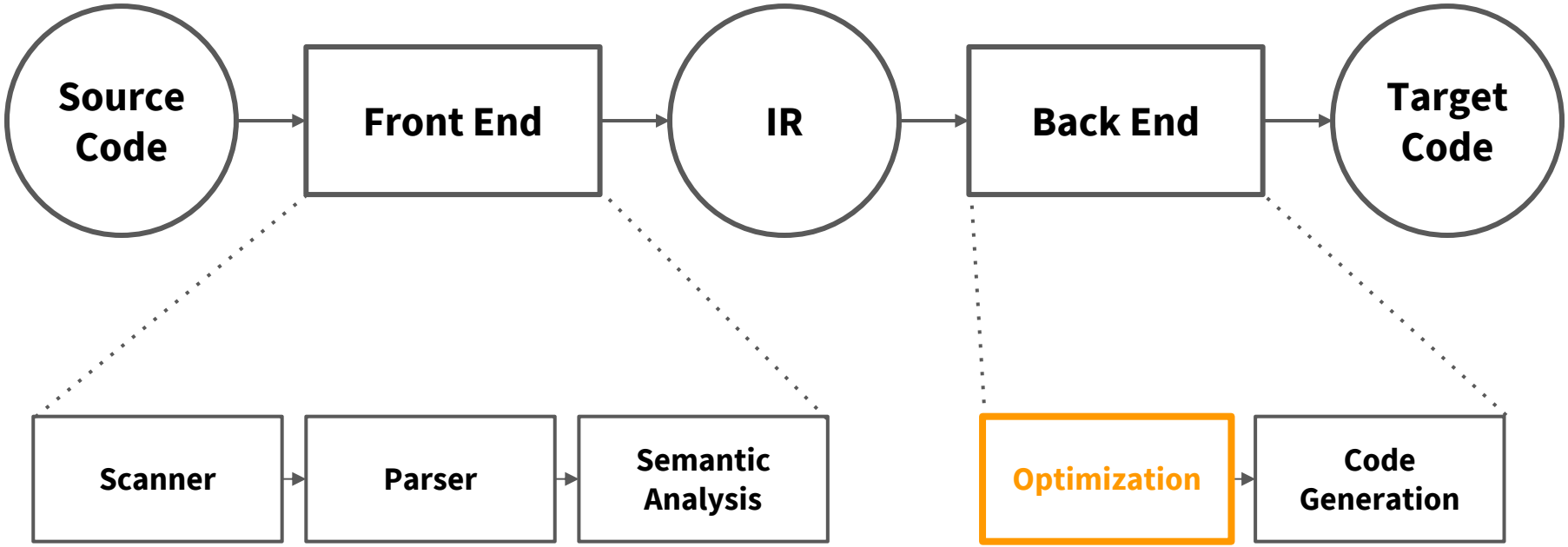
Announcements

- Thanksgiving next week!

- Codegen due next Tuesday, 11/26 -- BEFORE Thanksgiving
  - If you haven’t started, you should start TODAY
  - Bugs are hard to fix for this one

- Compiler Additions will be due the following Thursday, 12/05

Review of Optimizations



Review of Optimizations

Peephole

Local

Intraprocedural / Global

Interprocedural

Review of Optimizations

Peephole	A few Instructions
Local	
Intraprocedural / Global	
Interprocedural	

Review of Optimizations

Peephole	A few Instructions
Local	A Basic Block
Intraprocedural / Global	
Interprocedural	

Review of Optimizations

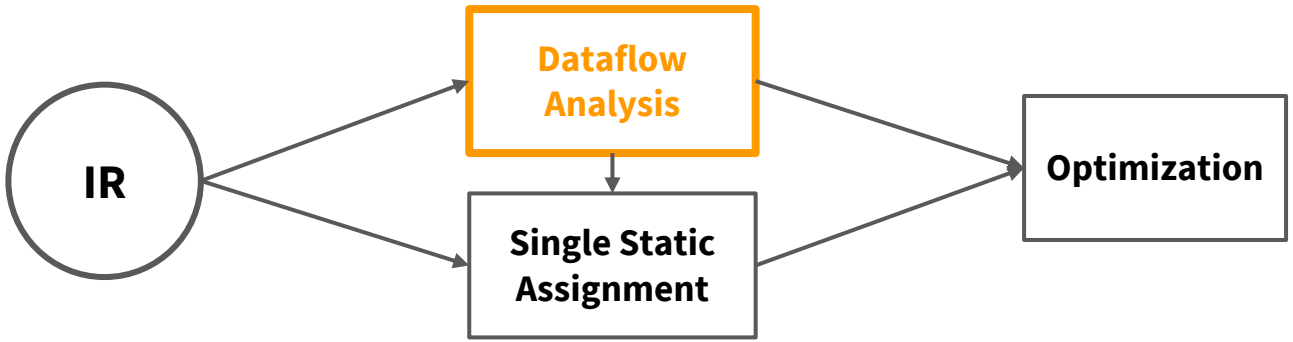
Peephole	A few Instructions
Local	A Basic Block
Intraprocedural / Global	A Function/Method
Interprocedural	

Review of Optimizations

Peephole	A few Instructions
Local	A Basic Block
Intraprocedural / Global	A Function/Method
Interprocedural	A Program

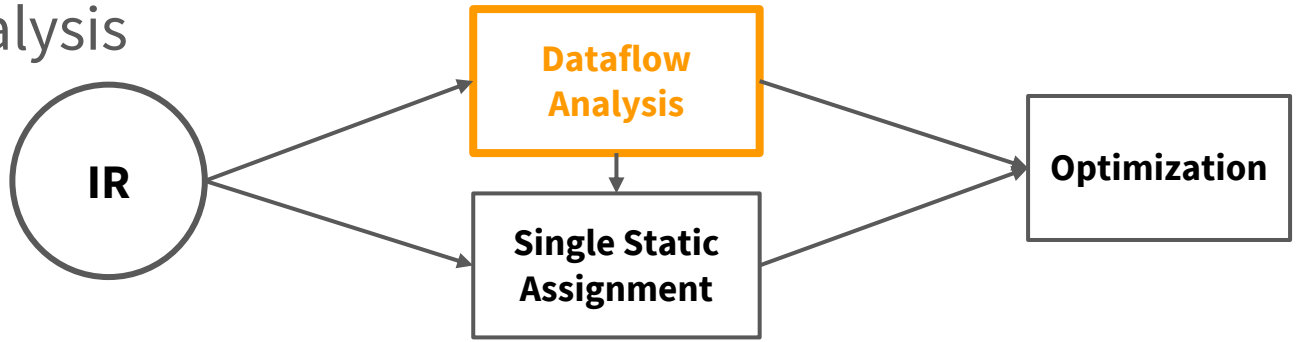
Overview of Dataflow Analysis

- A framework for exposing properties about programs
- Operates using sets of “facts”



Overview of Dataflow Analysis

- A framework for exposing properties about programs
- Operates using sets of “facts”
- Just the initial discovery phase
  - Changes can then be made to optimize based on the analysis



Overview of Dataflow Analysis

- Basic Framework of Set Definitions (for a Basic Block *b*):
  - `IN(`*b*`)`: facts true on entry to *b*
  - `OUT(`*b*`)`: facts true on exit from *b*
  - `GEN(`*b*`)`: facts created (and not killed) in *b*
  - `KILL(`*b*`)`: facts killed in *b*

Reaching Definitions (*A Dataflow Problem*)

“What definitions of each variable might reach this point”

- Could be used for:
  - Constant Propagation
  - Uninitialized Variables

```
int x;

if (y > 0) {
  x = y;
} else {
  x = 0;
}

System.out.println(x);
```

“x=y”, “x=0”

Reaching Definitions (*A Dataflow Problem*)

“What definitions of each variable might reach this point”

- **Be careful**: Does not involve the *value* of the definition
  - The dataflow problem “Available Expressions” is designed for that

```
int x;

if (y > 0) {
  x = y;
} else {
  x = 0;
}

y = -1;
System.out.println(x);
```

still: “x=y”, “x=0”

Problems 1_a and 1_b

Equations for Reaching Definitions

- IN(b): the definitions reaching upon entering block b
- OUT(b): the definitions reaching upon exiting block b
- GEN(b): the definitions assigned and not killed in block b
- KILL(b): the definitions of variables overwritten in block b

$$\text{IN}(b) = \bigcup_{p \in \text{pred}(b)} \text{OUT}(p)$$

$$\text{OUT}(b) = \text{GEN}(b) \cup (\text{IN}(b) - \text{KILL}(b))$$

Another *Equivalent* Set of Equations (from Lecture):

- Sets:
  - `DEFOUT(b)`: set of definitions in b that reach the end of b (i.e., not subsequently redefined in b)
  - `SURVIVED(b)`: set of all definitions not obscured by a definition in b
  - `REACHES(b)`: set of definitions that reach b
- Equations:

$$\texttt{REACHES(b)} = \left(\bigcup_{p \in \texttt{preds(b)}} \texttt{DEFOUT(p)}\right) \cup \left(\texttt{REACHES(p)} \cap \texttt{SURVIVED(p)}\right)$$

Problems 1_c and 1_d

```
L0:  a = 0
L1:  b = a + 1
L2:  c = c + b
L3:  a = b * 2
L4:  if a < N goto L1
L5:  return c
```

Block	GEN	KILL	IN (1)	OUT (1)	IN (2)	OUT (2)
L0	L0					
L1	L1					
L2	L2					
L3	L3					
L4						
L5						

```
L0:  a = 0
L1:  b = a + 1
L2:  c = c + b
L3:  a = b * 2
L4:  if a < N goto L1
L5:  return c
```

Block	GEN	KILL	IN (1)	OUT (1)	IN (2)	OUT (2)
L0	L0	L3				
L1	L1					
L2	L2					
L3	L3	L0				
L4						
L5						

```
L0:  a = 0
L1:  b = a + 1
L2:  c = c + b
L3:  a = b * 2
L4:  if a < N goto L1
L5:  return c
```

Block	GEN	KILL	IN (1)	OUT (1)	IN (2)	OUT (2)
L0	L0	L3				
L1	L1		L0			
L2	L2		L0, L1			
L3	L3	L0	L0, L1, L2			
L4			L1, L2, L3			
L5			L1, L2, L3			

```
L0:  a = 0
L1:  b = a + 1
L2:  c = c + b
L3:  a = b * 2
L4:  if a < N goto L1
L5:  return c
```

Block	GEN	KILL	IN (1)	OUT (1)	IN (2)	OUT (2)
L0	L0	L3		L0		
L1	L1		L0	L0, L1		
L2	L2		L0, L1	L0, L1, L2		
L3	L3	L0	L0, L1, L2	L1, L2, L3		
L4			L1, L2, L3	L1, L2, L3		
L5			L1, L2, L3	L1, L2, L3		

```
L0:  a = 0
L1:  b = a + 1
L2:  c = c + b
L3:  a = b * 2
L4:  if a < N goto L1
L5:  return c
```

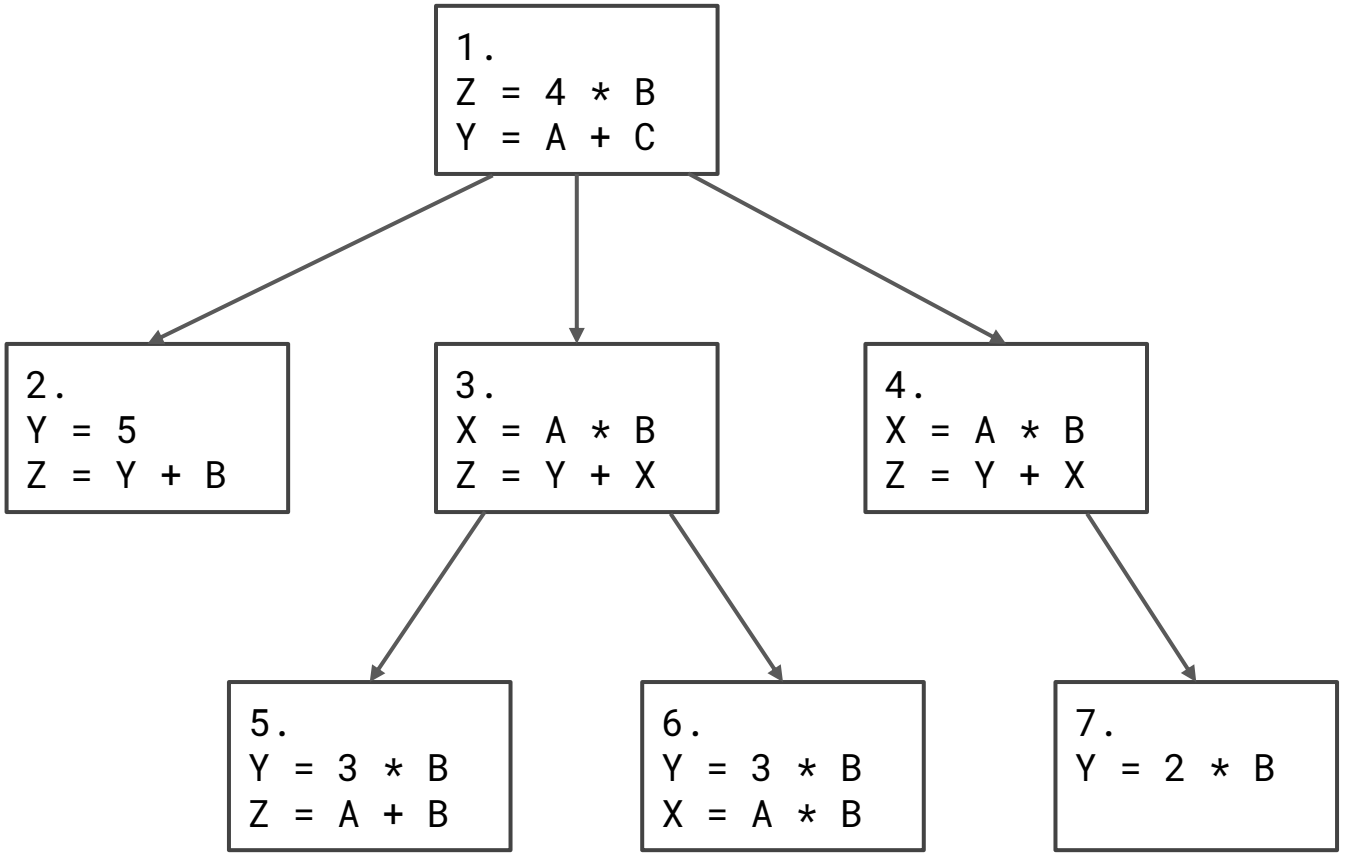
Block	GEN	KILL	IN (1)	OUT (1)	IN (2)	OUT (2)
L0	L0	L3		L0		L0
L1	L1		L0	L0, L1	L0, L1, L2, L3	L0, L1, L2, L3
L2	L2		L0, L1	L0, L1, L2	L0, L1, L2, L3	L0, L1, L2, L3
L3	L3	L0	L0, L1, L2	L1, L2, L3	L0, L1, L2, L3	L1, L2, L3
L4			L1, L2, L3	L1, L2, L3	L1, L2, L3	L1, L2, L3
L5			L1, L2, L3	L1, L2, L3	L1, L2, L3	L1, L2, L3

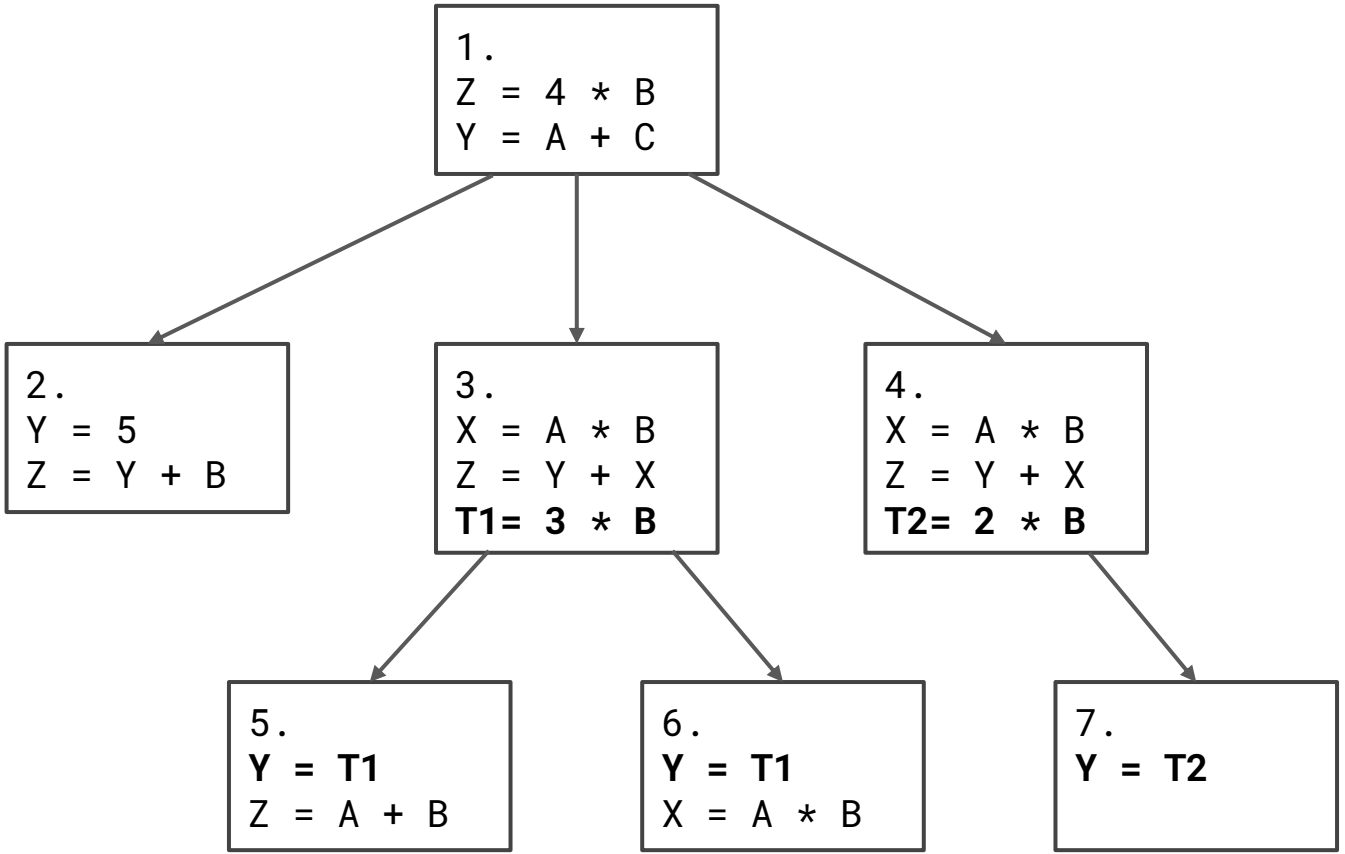
```
L0:  a = 0
L1:  b = a + 1
L2:  c = c + b
L3:  a = b * 2
L4:  if a < N goto L1
L5:  return c
```

Convergence!

Block	GEN	KILL	IN (1)	OUT (1)	IN (2)	OUT (2)
L0	L0	L3		L0		L0
L1	L1		L0	L0, L1	L0, L1, L2, L3	L0, L1, L2, L3
L2	L2		L0, L1	L0, L1, L2	L0, L1, L2, L3	L0, L1, L2, L3
L3	L3	L0	L0, L1, L2	L1, L2, L3	L0, L1, L2, L3	L1, L2, L3
L4			L1, L2, L3	L1, L2, L3	L1, L2, L3	L1, L2, L3
L5			L1, L2, L3	L1, L2, L3	L1, L2, L3	L1, L2, L3

Problems 2_a and 2_b


1.
Z = 4 * B
Y = A + C
2.
Y = 5
Z = Y + B
3.
X = A * B
Z = Y + X
4.
X = A * B
Z = Y + X
5.
Y = 3 * B
Z = A + B
6.
Y = 3 * B
X = A * B
7.
Y = 2 * B


1.
Z = 4 * B
Y = A + C
2.
Y = 5
Z = Y + B
3.
X = A * B
Z = Y + X
T1= 3 * B
4.
X = A * B
Z = Y + X
T2= 2 * B
5.
Y = T1
Z = A + B
6.
Y = T1
X = A * B
7.
Y = T2