

Program Analysis  
<https://www.cse.iitb.ac.in/~karkare/cs618/>

## Welcome & Introduction

Amey Karkare  
Dept of Computer Science and Engg  
IIT Kanpur  
Visiting IIT Bombay  
[karkare@cse.iitk.ac.in](mailto:karkare@cse.iitk.ac.in)  
[karkare@cse.iitb.ac.in](mailto:karkare@cse.iitb.ac.in)



## Program Analysis: About the course

- Analysis of a Program, by a Program, for a Program
- of a Program - User Program
- by a Program - Analyzer (Compiler, Runtime)
- for a Program - Optimizer, Verifier, ...

"Democracy is the government of the people, by the people, for the people" - Abraham Lincoln

2

## Expectations from You

- Basic Compilers Knowledge
- Write code
  - C/C++ for Assignments
  - C/C++/Java/Python for Project (Tentative)
- Understand and modify **large code base**
  - GCC, LLVM, SOOT
- Read state of the art research papers
  - Discussions in class

3

## Your Expectations

- ?

4

### Quick Quizzes (QQs)

- There will be small quizzes (10-15 min duration) during the class.
- These can be announced or **un-announced (surprize quizzes)**.
- Always bring **a pen and some loose papers** to the class

5

### QQ #1 (Ungraded)



- What are the various phases of compilers that you know? (5 minutes)

6

### Assignments / Exercises

- Short assignments to apply the lecture material.
- Assignments will have some written and some programming tasks.
- 4-5 Assignments for the semester

7

### Using Program Analysis

- **Compiler Code Optimizations**
- Why are optimizations important?
- Why not write optimized code to begin with?
- Where do optimizations fit in the compiler flow?

8

## Code Optimization

- Machine Independent
  - Remove redundancy introduced by the Programmer
  - Remove redundancy not required by later phases of compiler
  - Take advantage of algebraic properties of operators
- Machine dependent
  - Take advantage of the properties of target machine

*Optimization must preserve the semantics of the original program!*

9

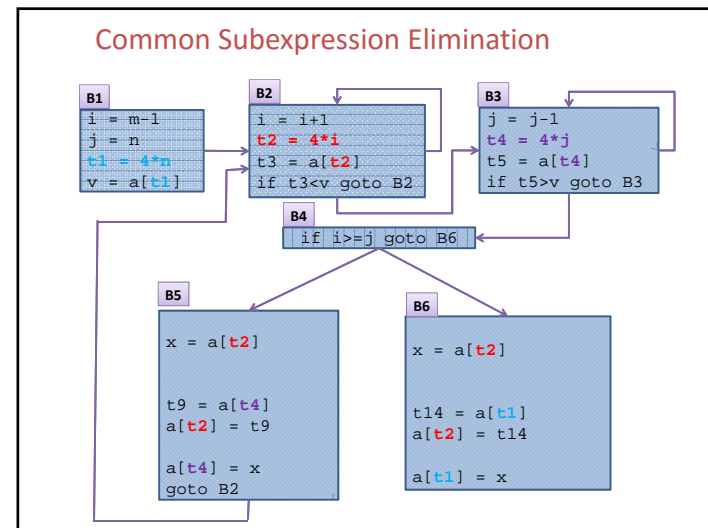
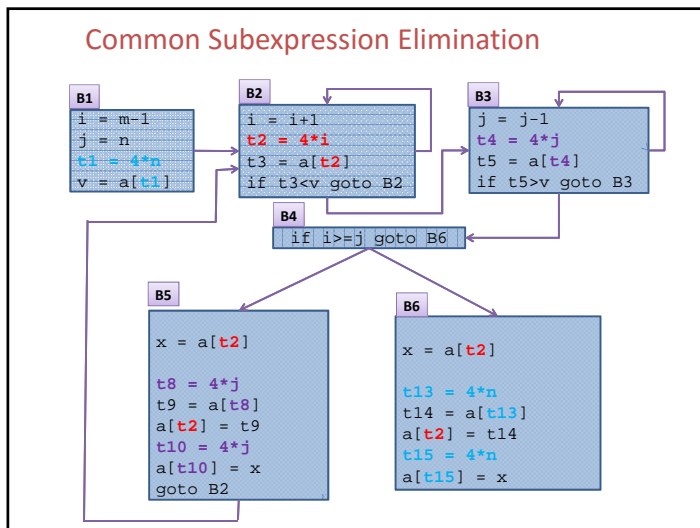
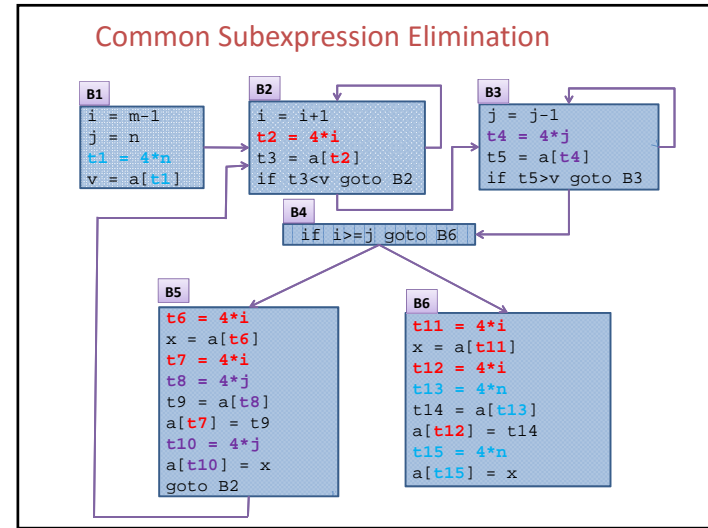
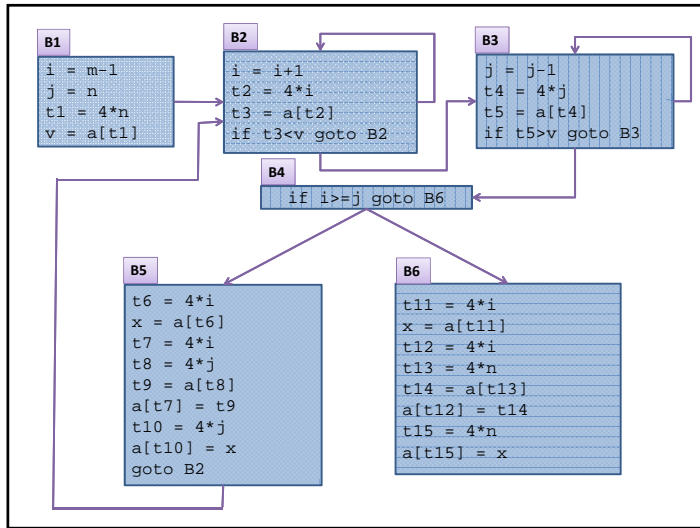
## Machine Independent Optimizations

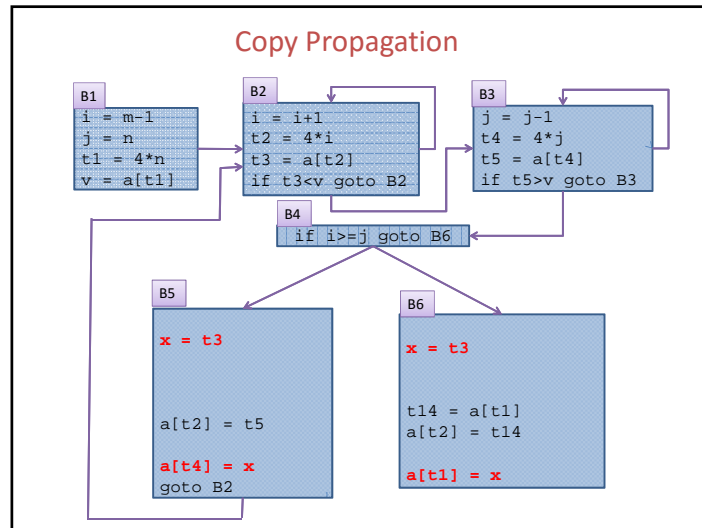
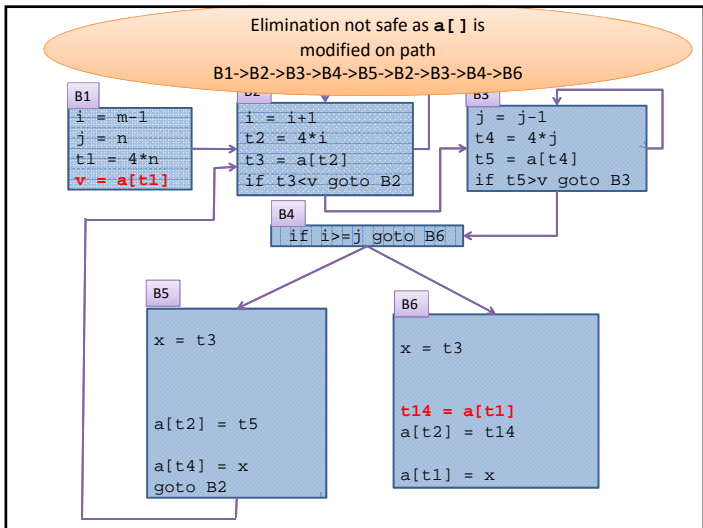
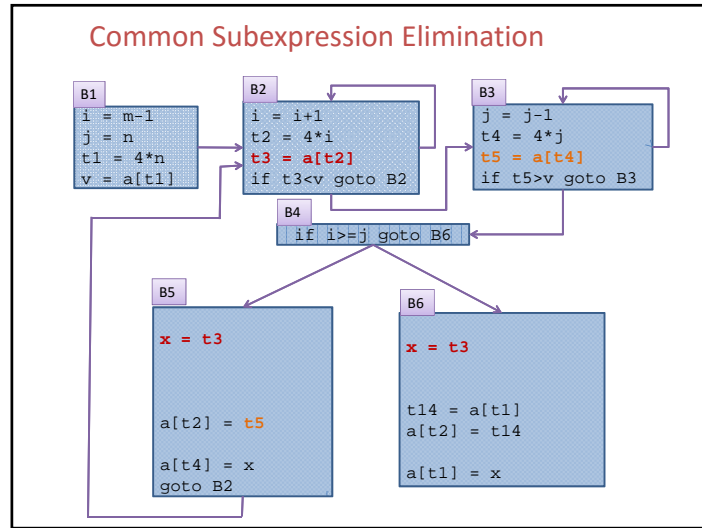
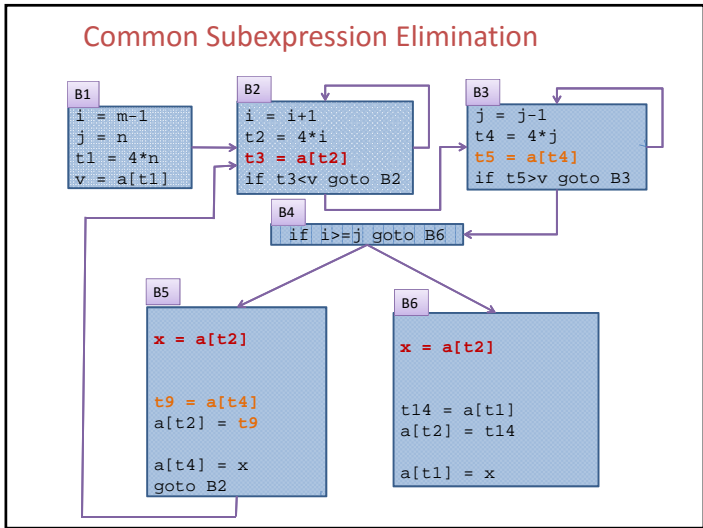
## Motivational Example

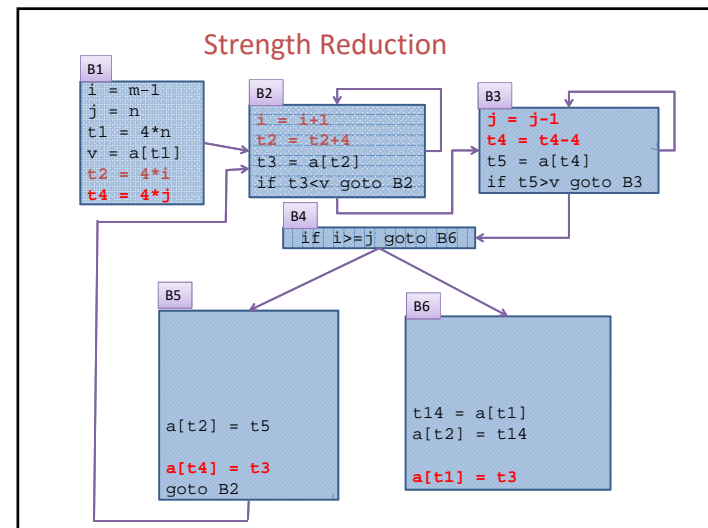
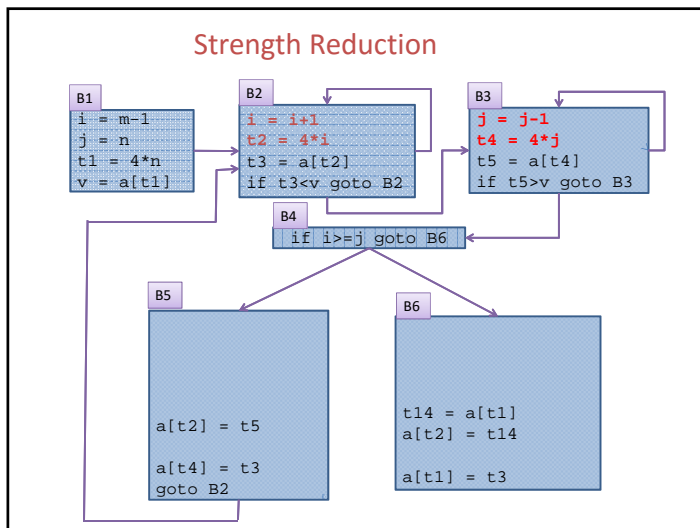
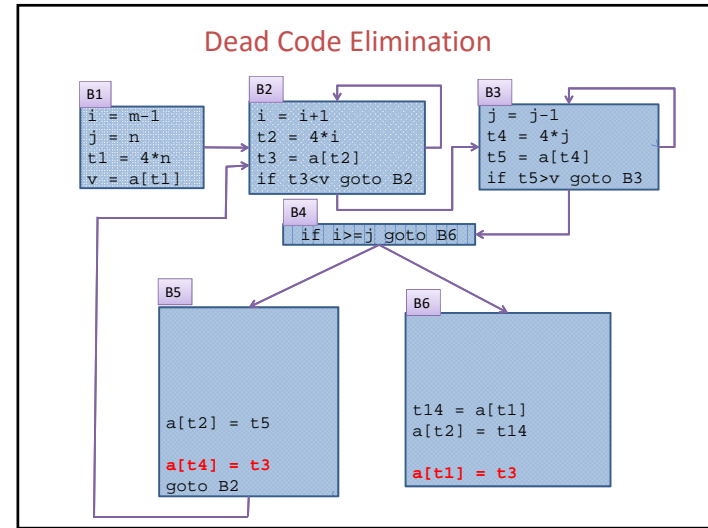
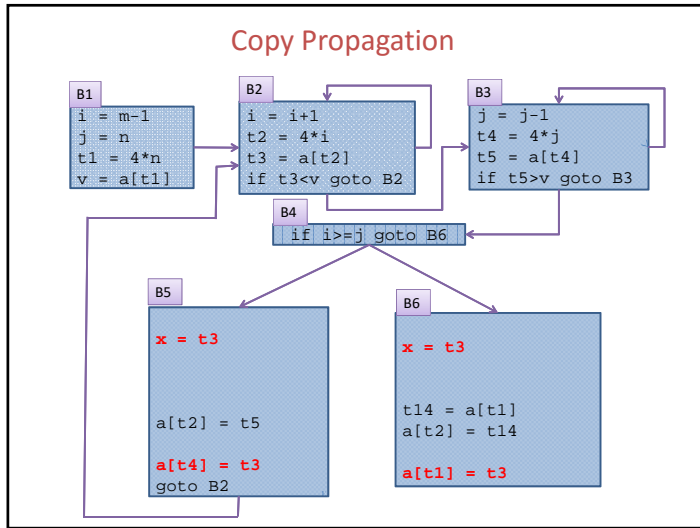
```
void quicksort(int m, int n)
/* recursively sort a[m] through a[n] */
{
  int i, j;
  int v, x;
  if(n <= m) return;
  i = m - 1; j = n; v = a[n];
  i = m - 1; j = n; v = a[n];
  while (1) {
    do i = i+1; while (a[i] < v);
    do j = j-1; while (a[j] > v);
    if (i > j) break;
    x = a[i]; a[i] = a[j]; a[j] = x;
  }
  x = a[i]; a[i] = a[n]; a[n] = x;
}
```

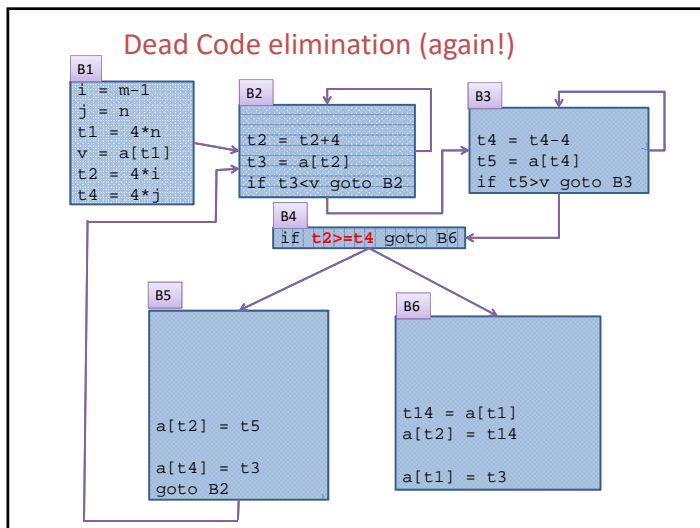
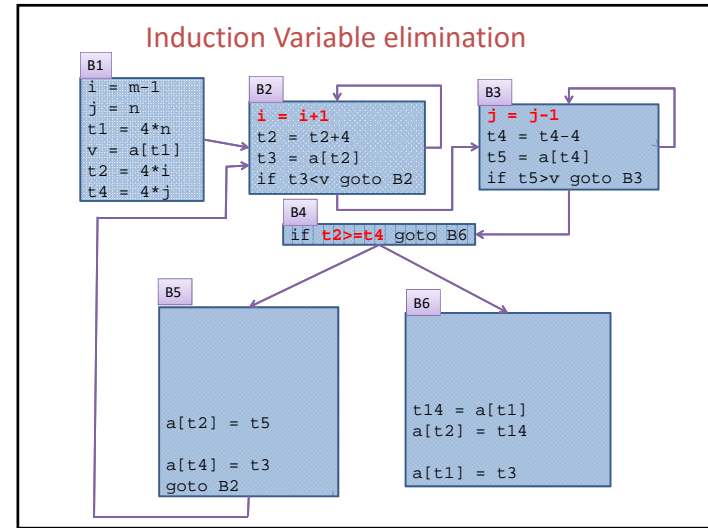
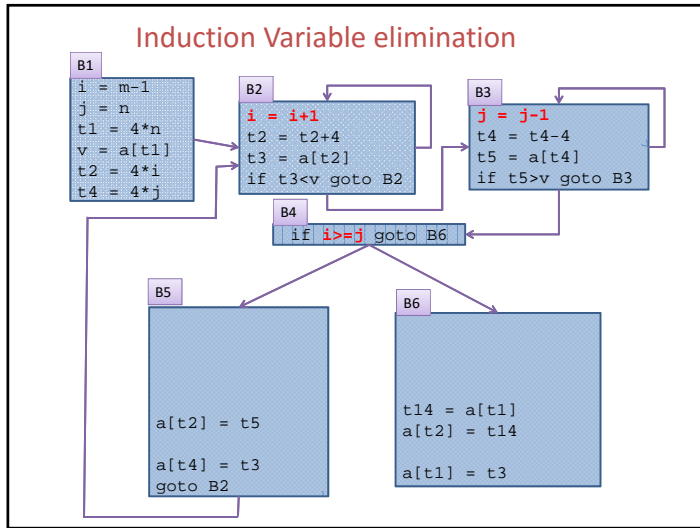
11

(1) i = m-1	(14) t6 = 4*i
(2) j = n	(15) x = a[t6]
(3) t1 = 4*n	(16) t7 = 4*i
(4) v = a[t1]	(17) t8 = 4*j
(5) i = i+1	(18) t9 = a[t8]
(6) t2 = 4*i	(19) a[t7] = t9
(7) t3 = a[t2]	(20) t10 = 4*j
(8) if t3<v goto (5)	(21) a[t10] = x
(9) j = j-1	(22) goto (5)
(10) t4 = 4*j	(23) t11 = 4*i
(11) t5 = a[t4]	(24) x = a[t11]
(12) if t5>v goto (9)	(25) t12 = 4*i
(13) if i>=j goto(23)	(26) t13 = 4*n
	(27) t14 = a[t13]
	(28) a[t12] = t14
	(29) t15 = 4*n
	(30) a[t15] = x









### Benefits

Block	# Stmt before Optimizations	# Stmt after Optimizations
B1	4	6
B2	4	3
B3	4	3
B4	1	1
B5	9	3
B6	8	3

**Assume:** Unit cost for statements, 10 iterations of outer loop, 100 iterations of each inner loop

**Cost of execution:**  
 ORIGINAL:  $1*4 + 100*4 + 100*4 + 10*1 + 10*9 + 1*8 = 912$   
 OPTIMIZED:  $1*6 + 100*3 + 100*3 + 10*1 + 10*3 + 1*3 = 649$

## Machine Dependent Optimizations

### Peephole Optimization

- target code often contains redundant instructions and suboptimal constructs
- examine a short sequence of target instruction (peephole) and replace by a shorter or faster sequence
- peephole is a small moving window on the target systems

30

### Peephole optimization examples...

#### Redundant loads and stores

- Consider the code sequence

Move R<sub>0</sub>, a

Move a, R<sub>0</sub>

- Instruction 2 can always be removed **if it does not have a label.**

31

### Peephole optimization examples...

#### Unreachable code

- Consider following code sequence

```
#define debug 0
if (debug) {
    print debugging info
}
```

this may be translated as

```
if debug == 1 goto L1
goto L2
```

```
L1: print debugging info
```

```
L2:
```

Eliminate jumps

```
if debug != 1 goto L2
print debugging information
```

```
L2:
```

32



### Unreachable code example ...

**constant propagation**  
 if 0 <> 1 goto L2  
     print debugging information  
 L2:

**Evaluate boolean expression. Since if condition is always true the code becomes**  
     goto L2  
     print debugging information  
 L2:

**The print statement is now unreachable. Therefore, the code becomes**  
 L2:

33

### Peephole optimization examples...

- flow of control: replace jump over jumps**

goto L1	by	goto L2
...		...
...		L1: goto L2
L1 : goto L2		

- Simplify algebraic expressions**  
 remove x := x+0    or    x:=x\*1

34

### Peephole optimization examples...

- Strength reduction**
  - Replace X^2 by X\*X
  - Replace multiplication by left shift
  - Replace division by right shift
- Use faster machine instructions**

replace	Add #1,R
by	Inc R

35

## Course Logistics

36

## Proposed Evaluation

Assignments	5%-10%
Course Project	30%-40%
( Proposal )	5%
( Report )	15%
( Presentation )	15%
Mid semester exam	10%-20%
End semester exam	25%-35%
Quizzes/Class Participation	5%

37

## Project

- Major part of the course
- You need to implement some non-trivial analysis/optimization using one of the open source infrastructure
  - For e.g., some paper published in last 10 years
- You are encouraged to suggest your own projects
- **Bonus** marks for publishable results
- **Individual OR Group of 2**

38

## Assignment #1

- Select one of the compiler infrastructure mentioned on the course webpage and
  - a) Download it
  - b) Build it
  - c) Submit a report
  - d) one page about the infrastructure, and the optimizations present in it.
  - e) one page about the most interesting optimization found, with example

39

## Assignment #1

- You can try more than one tool, even something not mentioned on the webpage.
  - But submit report for only **one**.
    - Preferably the one you plan to use for your project.
  - **DEADLINE: July 30<sup>th</sup>, End of Day (before Midnight)**
  - See course website for submission details (TBD)
- <https://www.cse.iitb.ac.in/~karkare/cs618/>

40