Optimization

Eva Rose   Kristoffer Rose

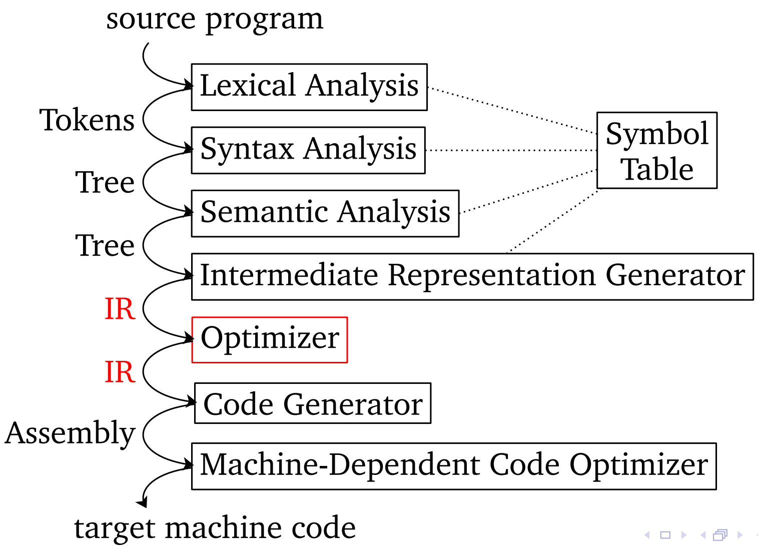
NYU Courant Institute

Compiler Construction (CSCI-GA.2130-001)

`http://cs.nyu.edu/courses/fall14/CSCI-GA.2130-001/lecture-12.pdf`

December 4, 2014

Sixth compilation phase



Sources of Redundancy

- Programmer “cut-n-paste”
- Compiler templates.
- Insufficient state transfer.

Sources of Redundancy

- Programmer “cut-n-paste”
- Compiler templates.
- Insufficient state transfer.

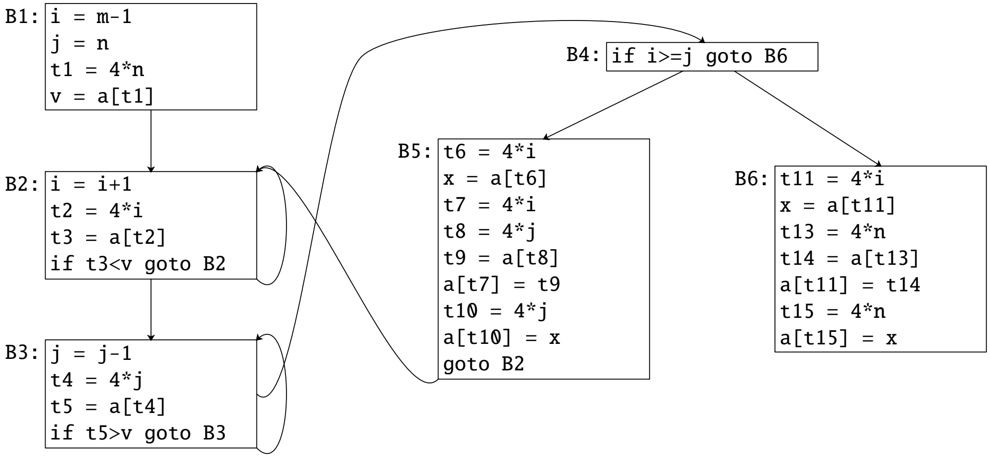
Sources of Redundancy

- Programmer “cut-n-paste”
- Compiler templates.
- Insufficient state transfer.

Example

```
void quicksort(int a[], int m, int n)
{
  int i, j, v, x; if (n <= m) return;

  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;

  quicksort(a,m,j); quicksort(a,i+1,n);
}
```

Basic Blocks



```
B1: i = m-1
    j = n
    t1 = 4*n
    v = a[t1]

B2: i = i+1
    t2 = 4*i
    t3 = a[t2]
    if t3<v goto B2

B3: j = j-1
    t4 = 4*j
    t5 = a[t4]
    if t5>v goto B3

B4: if i>=j goto B6

B5: t6 = 4*i
    x = a[t6]
    t7 = 4*i
    t8 = 4*j
    t9 = a[t8]
    a[t7] = t9
    t10 = 4*j
    a[t10] = x
    goto B2

B6: t11 = 4*i
    x = a[t11]
    t13 = 4*n
    t14 = a[t13]
    a[t11] = t14
    t15 = 4*n
    a[t15] = x
```

(Local) Common Subexpression Elimination

```
B5: t6 = 4*i
    x = a[t6]
    t7 = 4*i
    t8 = 4*j
    t9 = a[t8]
    a[t7] = t9
    t10 = 4*j
    a[t10] = x
    goto B2
```

⇒

```
B5: t6 = 4*i
    x = a[t6]
    t8 = 4*j
    t9 = a[t8]
    a[t6] = t9
    a[t8] = x
    goto B2
```

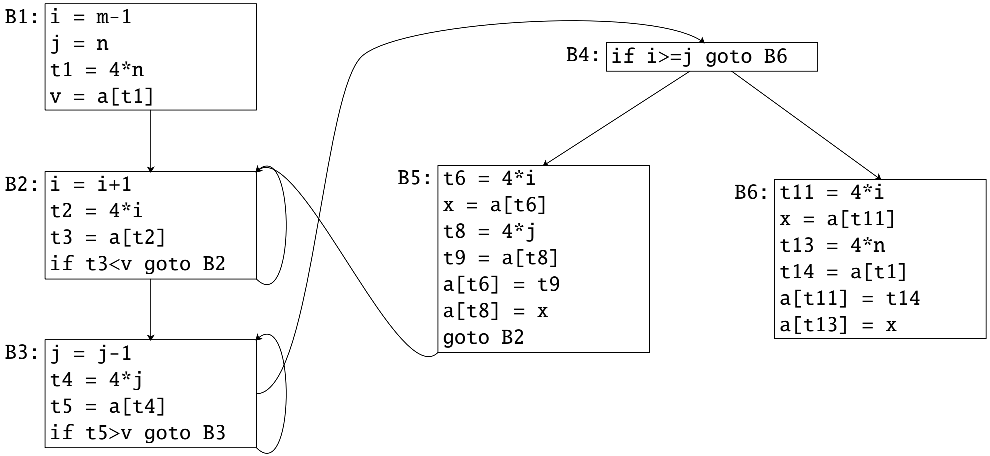
(Local) Common Subexpression Elimination

```
B5: t6 = 4*i
    x = a[t6]
    t7 = 4*i
    t8 = 4*j
    t9 = a[t8]
    a[t7] = t9
    t10 = 4*j
    a[t10] = x
    goto B2
```

$\Rightarrow$

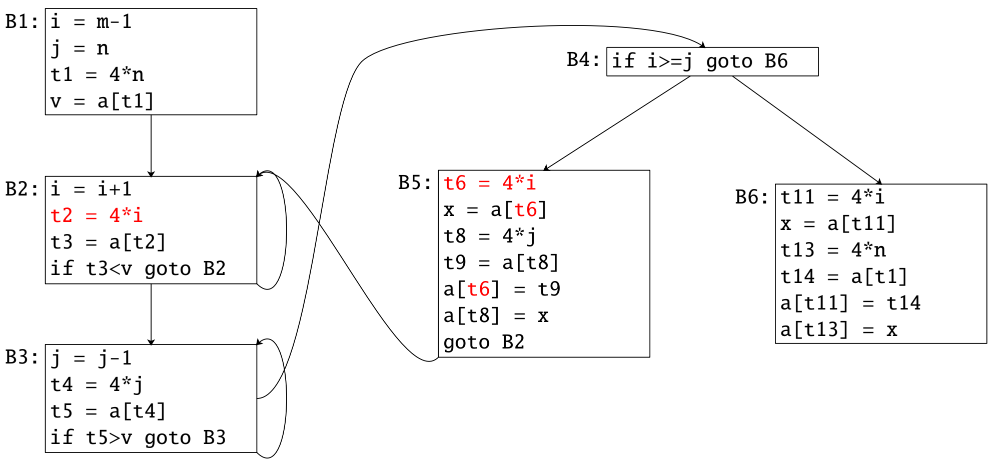
```
B5: t6 = 4*i
    x = a[t6]
    t8 = 4*j
    t9 = a[t8]
    a[t6] = t9
    a[t8] = x
    goto B2
```

Global Common Subexpression Elimination



```
B1: i = m-1
    j = n
    t1 = 4*n
    v = a[t1]

B2: i = i+1
    t2 = 4*i
    t3 = a[t2]
    if t3<v goto B2

B3: j = j-1
    t4 = 4*j
    t5 = a[t4]
    if t5>v goto B3

B4: if i>=j goto B6

B5: t6 = 4*i
    x = a[t6]
    t8 = 4*j
    t9 = a[t8]
    a[t6] = t9
    a[t8] = x
    goto B2

B6: t11 = 4*i
    x = a[t11]
    t13 = 4*n
    t14 = a[t1]
    a[t11] = t14
    a[t13] = x
```

Global Common Subexpression Elimination



B1:
```
i = m-1
j = n
t1 = 4*n
v = a[t1]
```

B2:
```
i = i+1
t2 = 4*i
t3 = a[t2]
if t3<v goto B2
```

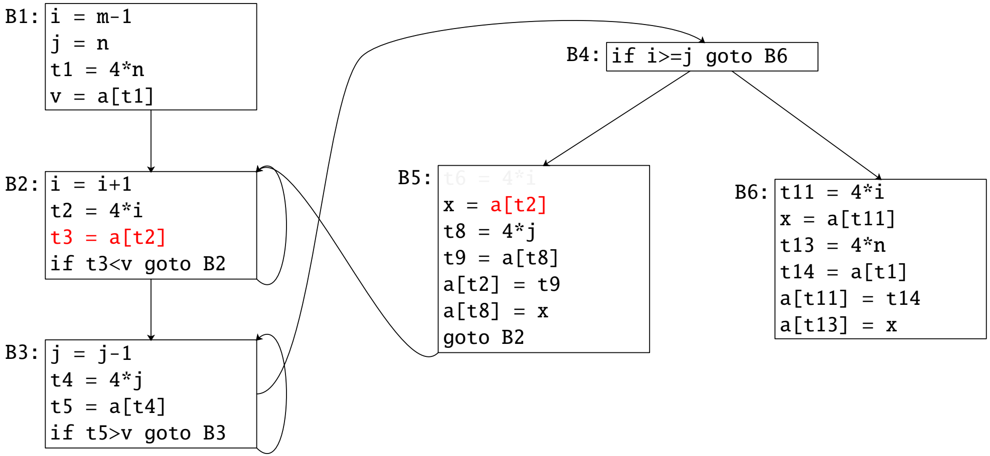
B3:
```
j = j-1
t4 = 4*j
t5 = a[t4]
if t5>v goto B3
```

B4:
```
if i>=j goto B6
```

B5:
```
t6 = 4*i
x = a[t6]
t8 = 4*j
t9 = a[t8]
a[t6] = t9
a[t8] = x
goto B2
```

B6:
```
t11 = 4*i
x = a[t11]
t13 = 4*n
t14 = a[t1]
a[t11] = t14
a[t13] = x
```

Global Common Subexpression Elimination



```
B1: i = m-1
    j = n
    t1 = 4*n
    v = a[t1]
```

```
B2: i = i+1
    t2 = 4*i
    t3 = a[t2]
    if t3<v goto B2
```

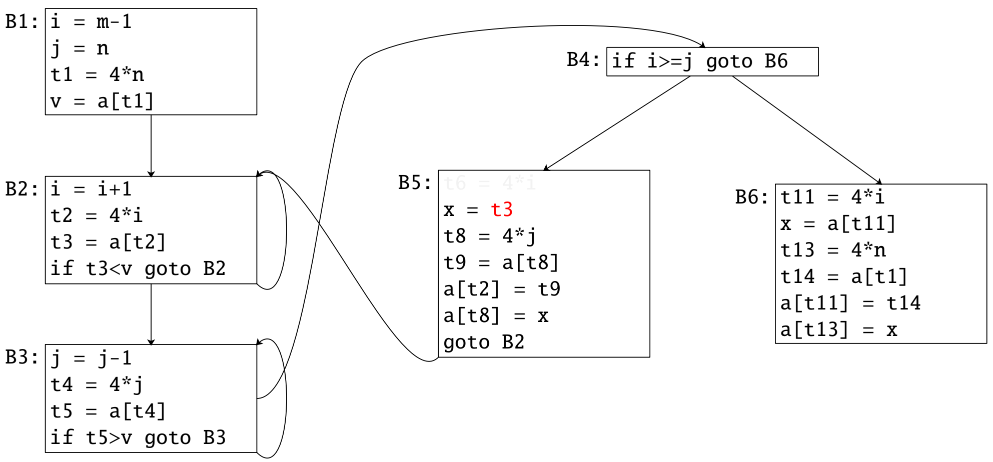
```
B3: j = j-1
    t4 = 4*j
    t5 = a[t4]
    if t5>v goto B3
```

```
B4: if i>=j goto B6
```

```
B5: t6 = 4*i
    x = a[t2]
    t8 = 4*j
    t9 = a[t8]
    a[t2] = t9
    a[t8] = x
    goto B2
```

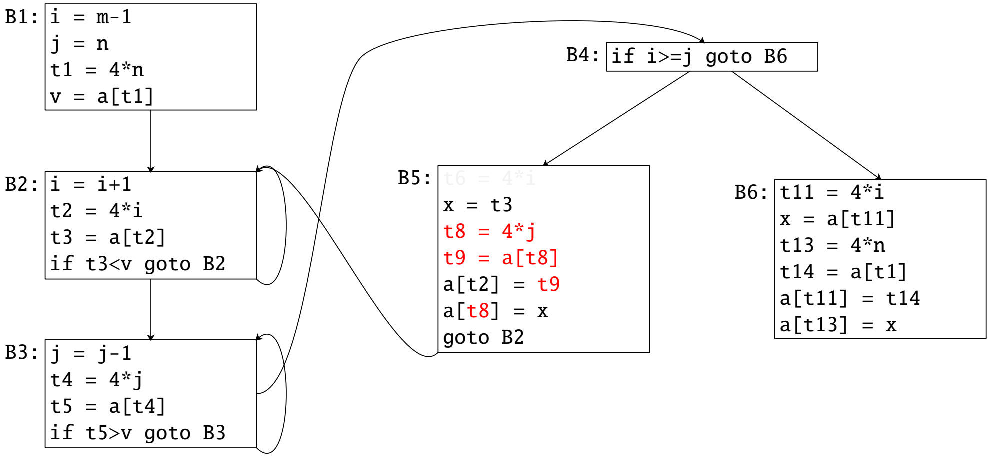
```
B6: t11 = 4*i
    x = a[t11]
    t13 = 4*n
    t14 = a[t1]
    a[t11] = t14
    a[t13] = x
```

Global Common Subexpression Elimination



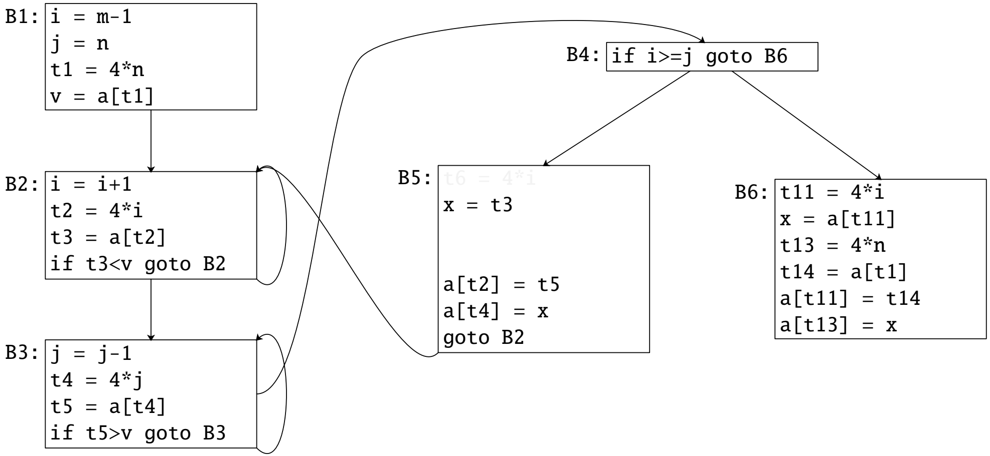
```
B1: i = m-1
    j = n
    t1 = 4*n
    v = a[t1]

B2: i = i+1
    t2 = 4*i
    t3 = a[t2]
    if t3<v goto B2

B3: j = j-1
    t4 = 4*j
    t5 = a[t4]
    if t5>v goto B3

B4: if i>=j goto B6

B5: t6 = 4*i
    x = t3
    t8 = 4*j
    t9 = a[t8]
    a[t2] = t9
    a[t8] = x
    goto B2

B6: t11 = 4*i
    x = a[t11]
    t13 = 4*n
    t14 = a[t1]
    a[t11] = t14
    a[t13] = x
```

Global Common Subexpression Elimination



```
B1: i = m-1
    j = n
    t1 = 4*n
    v = a[t1]

B2: i = i+1
    t2 = 4*i
    t3 = a[t2]
    if t3<v goto B2

B3: j = j-1
    t4 = 4*j
    t5 = a[t4]
    if t5>v goto B3

B4: if i>=j goto B6

B5: t6 = 4*i
    x = t3
    t8 = 4*j
    t9 = a[t8]
    a[t2] = t9
    a[t8] = x
    goto B2

B6: t11 = 4*i
    x = a[t11]
    t13 = 4*n
    t14 = a[t1]
    a[t11] = t14
    a[t13] = x
```

Global Common Subexpression Elimination



```
B1: i = m-1
    j = n
    t1 = 4*n
    v = a[t1]
```

```
B2: i = i+1
    t2 = 4*i
    t3 = a[t2]
    if t3<v goto B2
```

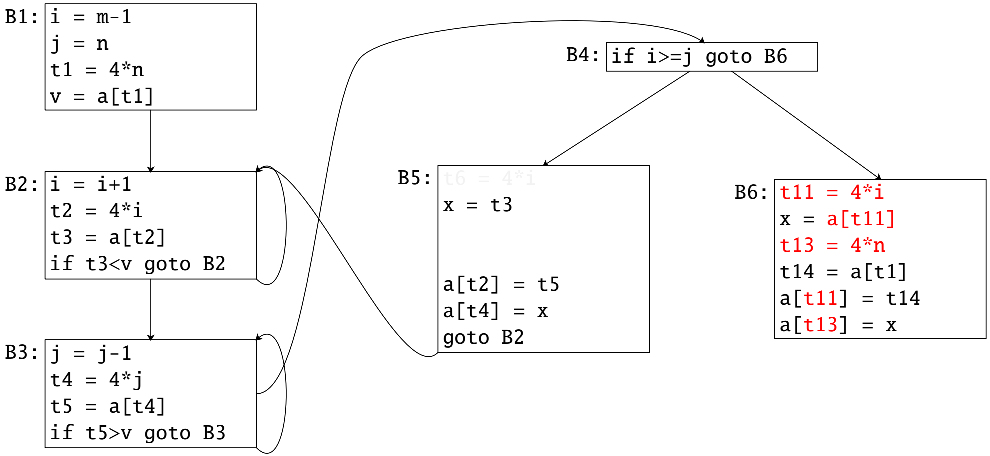
```
B3: j = j-1
    t4 = 4*j
    t5 = a[t4]
    if t5>v goto B3
```

```
B4: if i>=j goto B6
```

```
B5: t6 = 4*i
    x = t3


    a[t2] = t5
    a[t4] = x
    goto B2
```

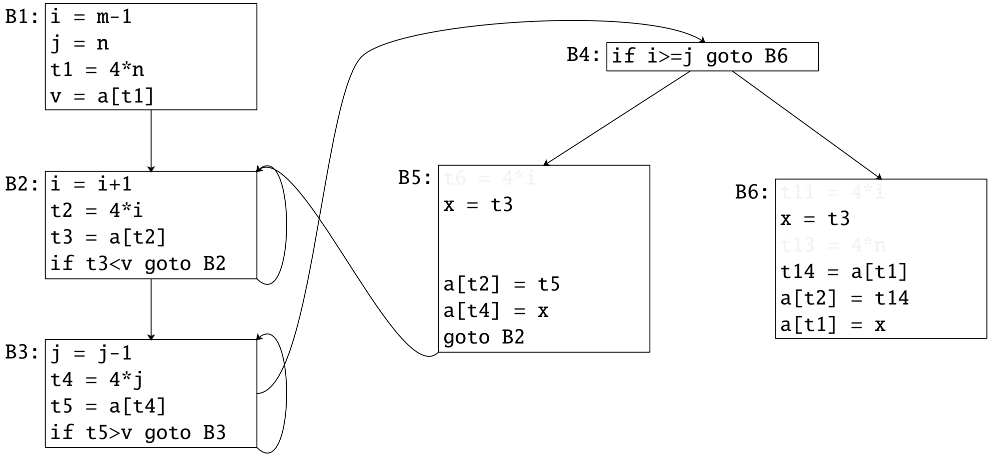
```
B6: t11 = 4*i
    x = a[t11]
    t13 = 4*n
    t14 = a[t1]
    a[t11] = t14
    a[t13] = x
```

Global Common Subexpression Elimination



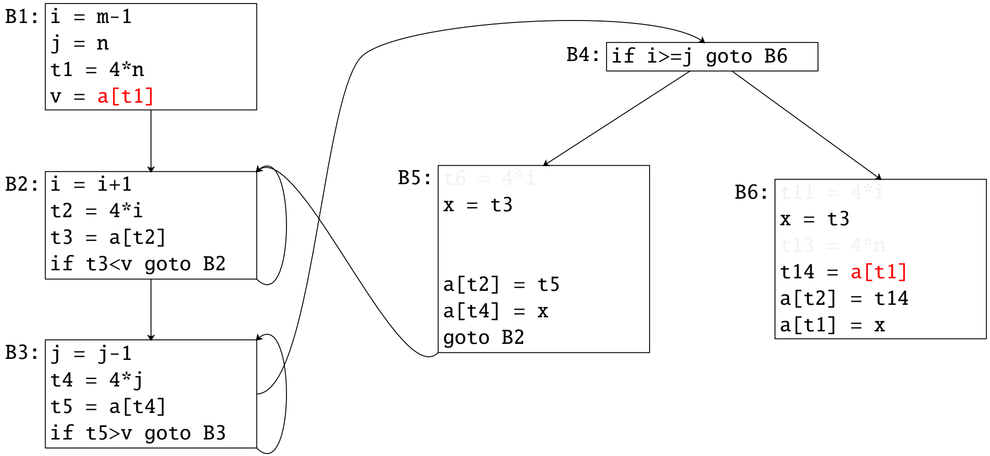
```
B1: i = m-1
    j = n
    t1 = 4*n
    v = a[t1]

B2: i = i+1
    t2 = 4*i
    t3 = a[t2]
    if t3<v goto B2

B3: j = j-1
    t4 = 4*j
    t5 = a[t4]
    if t5>v goto B3

B4: if i>=j goto B6

B5: t6 = 4*i
    x = t3


    a[t2] = t5
    a[t4] = x
    goto B2

B6: t11 = 4*i
    x = a[t11]
    t13 = 4*n
    t14 = a[t1]
    a[t11] = t14
    a[t13] = x
```

Global Common Subexpression Elimination



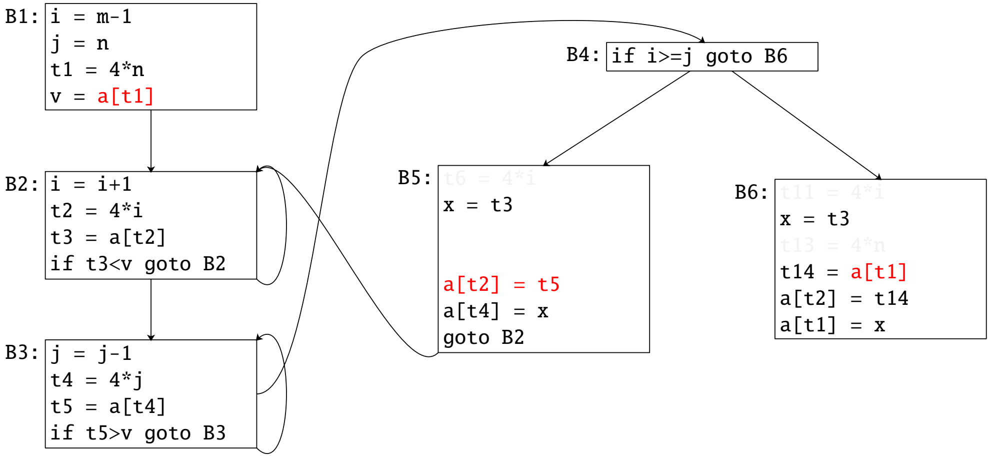
```
B1: i = m-1
    j = n
    t1 = 4*n
    v = a[t1]

B2: i = i+1
    t2 = 4*i
    t3 = a[t2]
    if t3<v goto B2

B3: j = j-1
    t4 = 4*j
    t5 = a[t4]
    if t5>v goto B3

B4: if i>=j goto B6

B5: t6 = 4*i
    x = t3


    a[t2] = t5
    a[t4] = x
    goto B2

B6: t11 = 4*i
    x = t3
    t13 = 4*n
    t14 = a[t1]
    a[t2] = t14
    a[t1] = x
```

Global Common Subexpression Elimination



```
B1: i = m-1
    j = n
    t1 = 4*n
    v = a[t1]

B2: i = i+1
    t2 = 4*i
    t3 = a[t2]
    if t3<v goto B2

B3: j = j-1
    t4 = 4*j
    t5 = a[t4]
    if t5>v goto B3

B4: if i>=j goto B6

B5: t6 = 4*i
    x = t3


    a[t2] = t5
    a[t4] = x
    goto B2

B6: t11 = 4*i
    x = t3
    t13 = 4*n
    t14 = a[t1]
    a[t2] = t14
    a[t1] = x
```

Global Common Subexpression Elimination



```
B1: i = m-1
    j = n
    t1 = 4*n
    v = a[t1]
```

```
B2: i = i+1
    t2 = 4*i
    t3 = a[t2]
    if t3<v goto B2
```

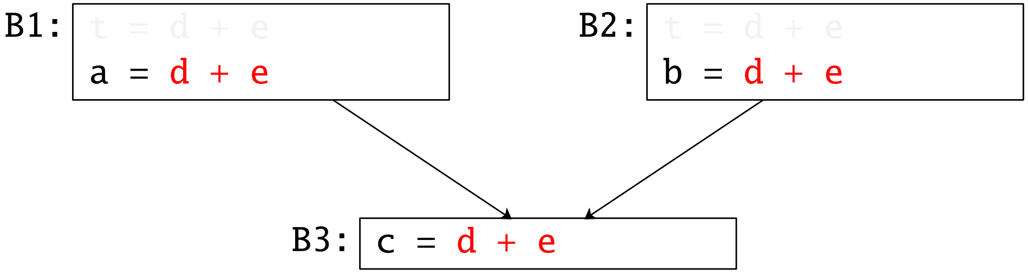
```
B3: j = j-1
    t4 = 4*j
    t5 = a[t4]
    if t5>v goto B3
```

```
B4: if i>=j goto B6
```

```
B5: t6 = 4*i
    x = t3


    a[t2] = t5
    a[t4] = x
    goto B2
```

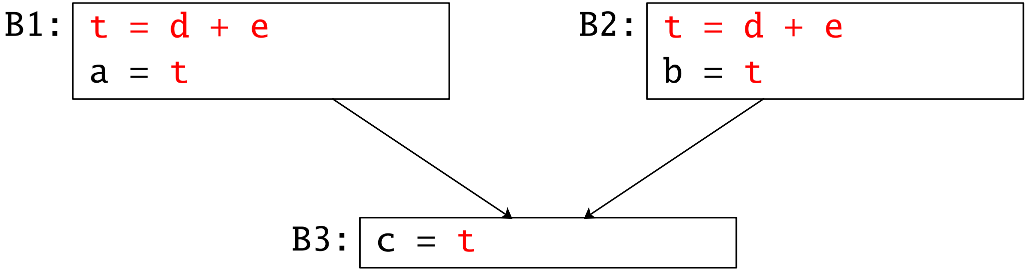
```
B6: t11 = 4*i
    x = t3
    t13 = 4*n
    t14 = a[t1]
    a[t2] = t14
    a[t1] = x
```

Copy Propagation



```
B1:
t = d + e
a = d + e

B2:
t = d + e
b = d + e

B3:
c = d + e
```

Copy Propagation



Code Motion

```
while (i <= limit-2) /*not changing limit*/
```

becomes

```
t = limit-2;
while (i <= t) /*not changing limit or t*/
```

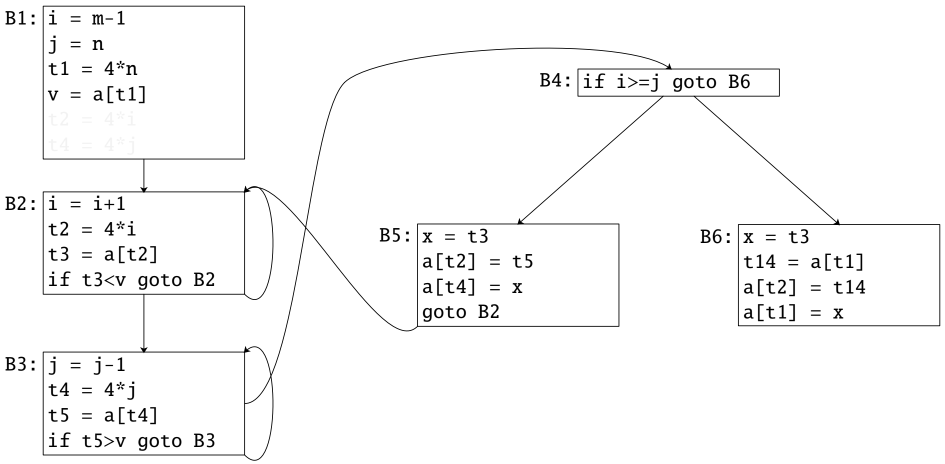
Code Motion

```
while (i <= limit-2) /*not changing limit*/
```

becomes

```
t = limit-2;
while (i <= t) /*not changing limit or t*/
```

Induction Invariants and Reduction in Strength



```
B1: i = m-1
    j = n
    t1 = 4*n
    v = a[t1]
    t2 = 4*i
    t4 = 4*j
```

```
B2: i = i+1
    t2 = 4*i
    t3 = a[t2]
    if t3<v goto B2
```

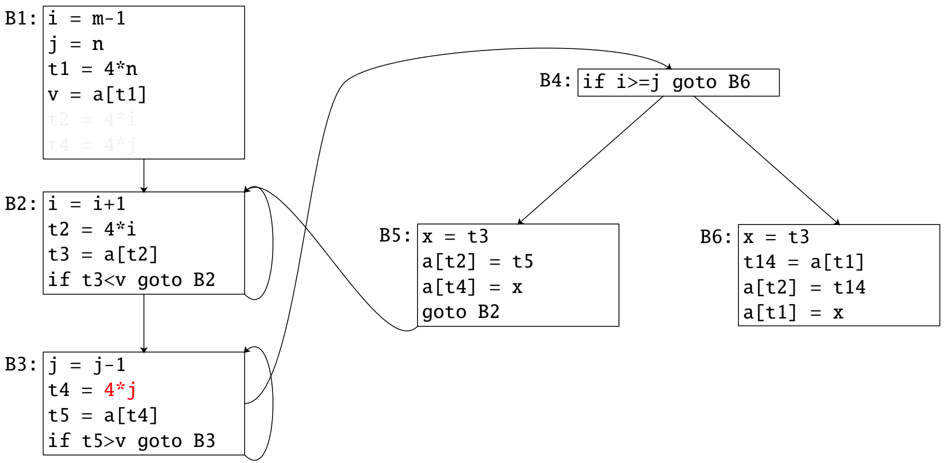
```
B3: j = j-1
    t4 = 4*j
    t5 = a[t4]
    if t5>v goto B3
```

```
B4: if i>=j goto B6
```

```
B5: x = t3
    a[t2] = t5
    a[t4] = x
    goto B2
```

```
B6: x = t3
    t14 = a[t1]
    a[t2] = t14
    a[t1] = x
```

Induction Invariants and Reduction in Strength



```
B1: i = m-1
    j = n
    t1 = 4*n
    v = a[t1]
    t2 = 4*i
    t4 = 4*j
```

```
B2: i = i+1
    t2 = 4*i
    t3 = a[t2]
    if t3<v goto B2
```

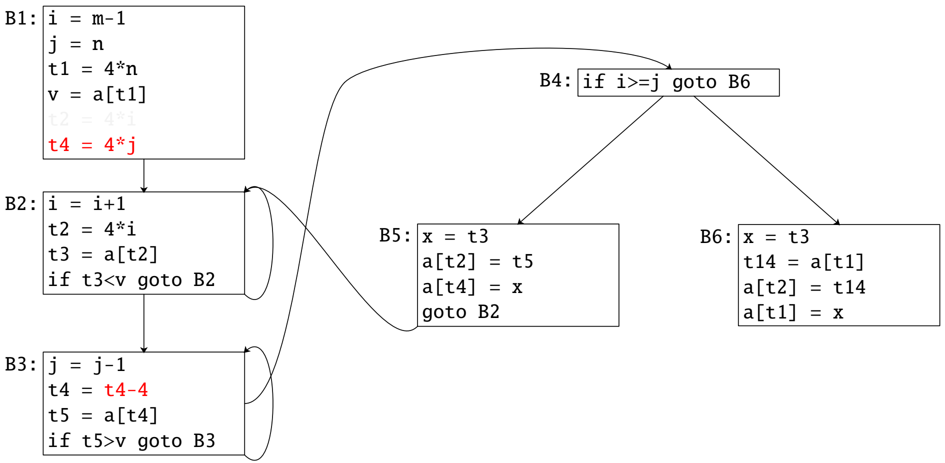
```
B3: j = j-1
    t4 = 4*j
    t5 = a[t4]
    if t5>v goto B3
```

```
B4: if i>=j goto B6
```

```
B5: x = t3
    a[t2] = t5
    a[t4] = x
    goto B2
```

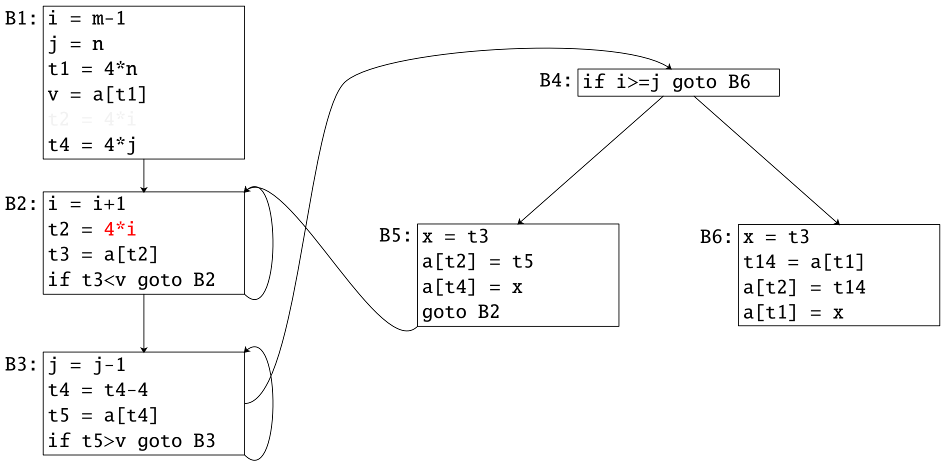
```
B6: x = t3
    t14 = a[t1]
    a[t2] = t14
    a[t1] = x
```

Induction Invariants and Reduction in Strength



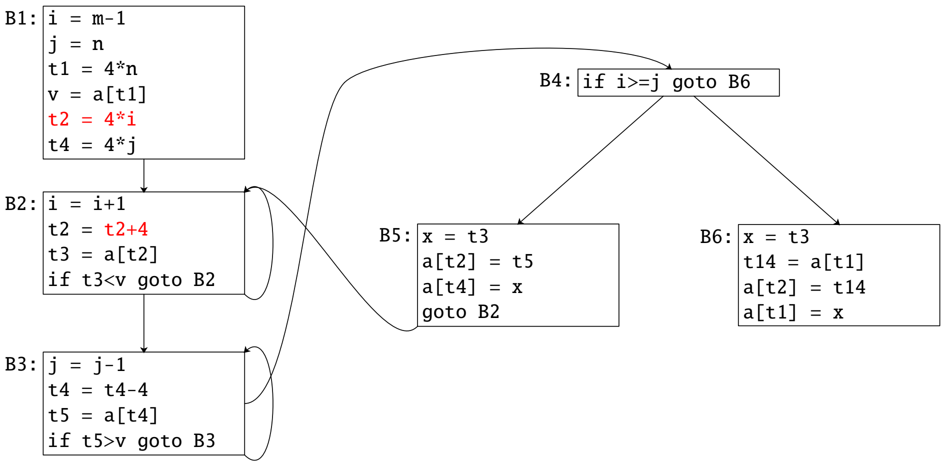
```
B1: i = m-1
    j = n
    t1 = 4*n
    v = a[t1]
    t2 = 4*i
    t4 = 4*j

B2: i = i+1
    t2 = 4*i
    t3 = a[t2]
    if t3<v goto B2

B3: j = j-1
    t4 = t4-4
    t5 = a[t4]
    if t5>v goto B3

B4: if i>=j goto B6

B5: x = t3
    a[t2] = t5
    a[t4] = x
    goto B2

B6: x = t3
    t14 = a[t1]
    a[t2] = t14
    a[t1] = x
```

Induction Invariants and Reduction in Strength

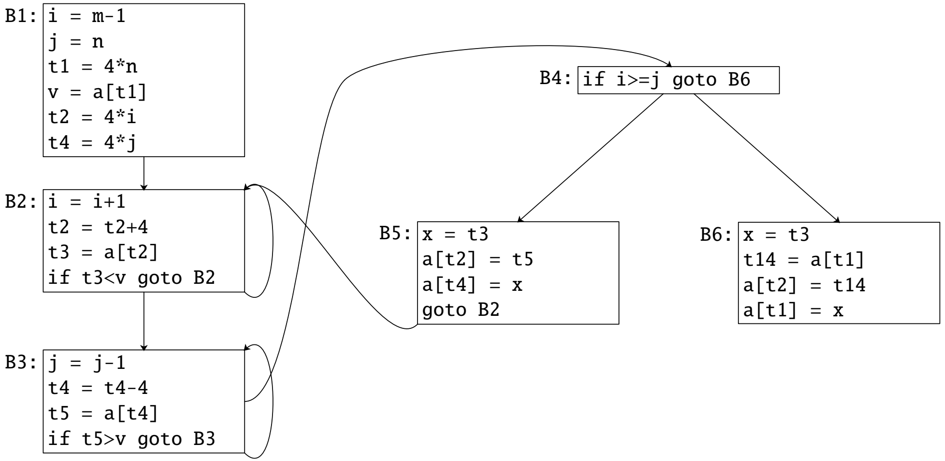


```
B1: i = m-1
    j = n
    t1 = 4*n
    v = a[t1]
    t2 = 4*i
    t4 = 4*j

B2: i = i+1
    t2 = 4*i
    t3 = a[t2]
    if t3<v goto B2

B3: j = j-1
    t4 = t4-4
    t5 = a[t4]
    if t5>v goto B3

B4: if i>=j goto B6

B5: x = t3
    a[t2] = t5
    a[t4] = x
    goto B2

B6: x = t3
    t14 = a[t1]
    a[t2] = t14
    a[t1] = x
```

Induction Invariants and Reduction in Strength

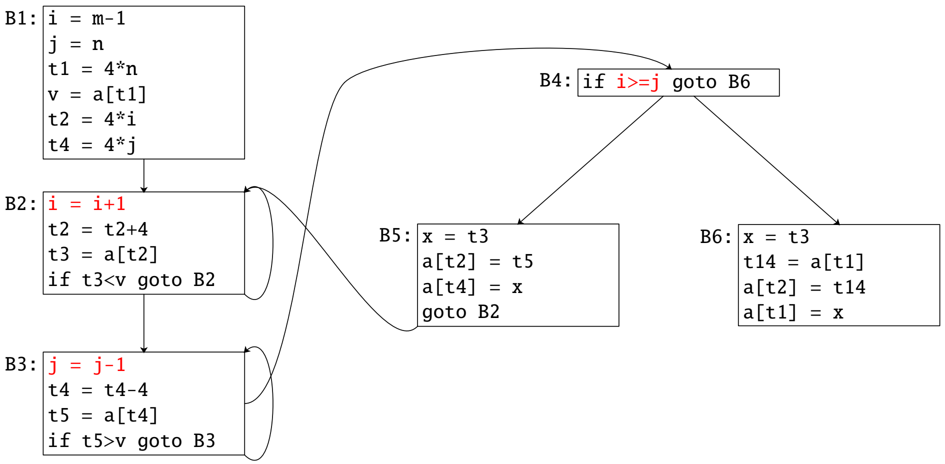


Induction Invariants and Reduction in Strength



```
B1: i = m-1
    j = n
    t1 = 4*n
    v = a[t1]
    t2 = 4*i
    t4 = 4*j

B2: i = i+1
    t2 = t2+4
    t3 = a[t2]
    if t3<v goto B2

B3: j = j-1
    t4 = t4-4
    t5 = a[t4]
    if t5>v goto B3

B4: if i>=j goto B6

B5: x = t3
    a[t2] = t5
    a[t4] = x
    goto B2

B6: x = t3
    t14 = a[t1]
    a[t2] = t14
    a[t1] = x
```

Induction Invariants and Reduction in Strength



B1:
```
i = m-1
j = n
t1 = 4*n
v = a[t1]
t2 = 4*i
t4 = 4*j
```

B2:
```
i = i+1
t2 = t2+4
t3 = a[t2]
if t3<v goto B2
```

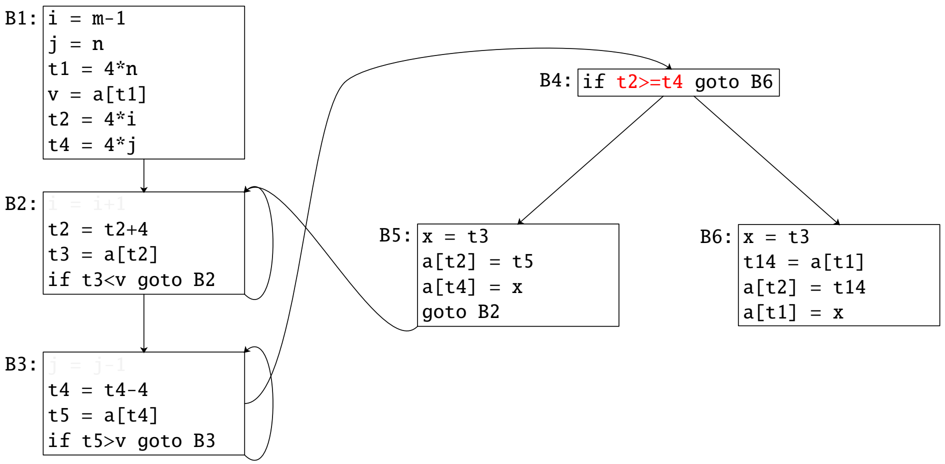
B3:
```
j = j-1
t4 = t4-4
t5 = a[t4]
if t5>v goto B3
```

B4:
```
if i>=j goto B6
```

B5:
```
x = t3
a[t2] = t5
a[t4] = x
goto B2
```

B6:
```
x = t3
t14 = a[t1]
a[t2] = t14
a[t1] = x
```

Induction Invariants and Reduction in Strength



```
B1: i = m-1
    j = n
    t1 = 4*n
    v = a[t1]
    t2 = 4*i
    t4 = 4*j
```

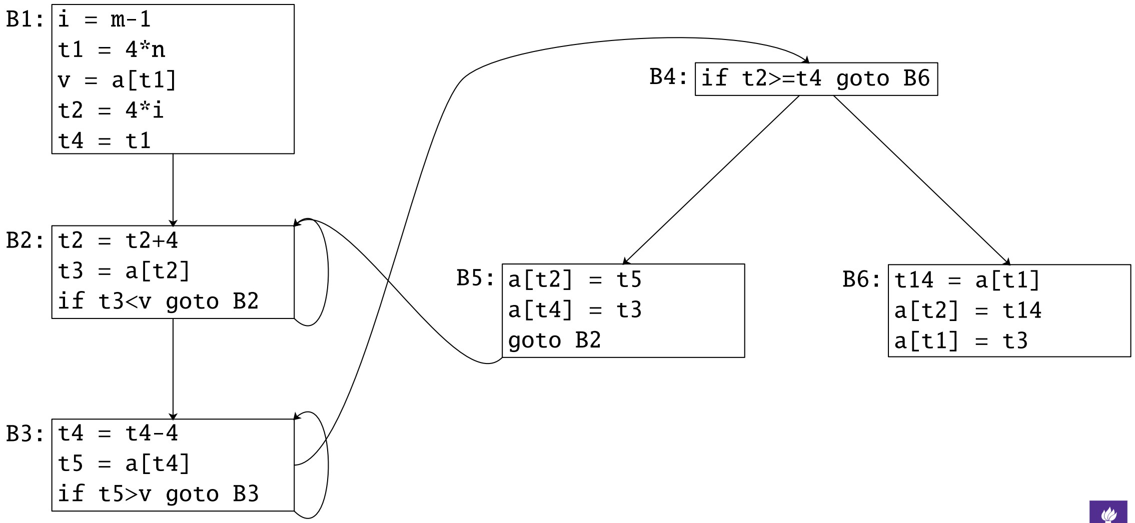
```
B2: i = i+1
    t2 = t2+4
    t3 = a[t2]
    if t3<v goto B2
```

```
B3: j = j-1
    t4 = t4-4
    t5 = a[t4]
    if t5>v goto B3
```

```
B4: if t2>=t4 goto B6
```

```
B5: x = t3
    a[t2] = t5
    a[t4] = x
    goto B2
```

```
B6: x = t3
    t14 = a[t1]
    a[t2] = t14
    a[t1] = x
```


B1:
i = m-1
t1 = 4*n
v = a[t1]
t2 = 4*i
t4 = t1
B2:
t2 = t2+4
t3 = a[t2]
if t3<v goto B2
B3:
t4 = t4-4
t5 = a[t4]
if t5>v goto B3
B4:
if t2>=t4 goto B6
B5:
a[t2] = t5
a[t4] = t3
goto B2
B6:
t14 = a[t1]
a[t2] = t14
a[t1] = t3

Summary

Redundancy but preserve semantics!

- Global Common Subexpressions.
- Copy Propagation.
- Dead-code Elimination.
- Code Motion.
- Induction Variables/Strength Reduction.

Summary

Redundancy but preserve semantics!

- Global Common Subexpressions.
- Copy Propagation.
- Dead-code Elimination.
- Code Motion.
- Induction Variables/Strength Reduction.

Summary

Redundancy but preserve semantics!

- Global Common Subexpressions.
- Copy Propagation.
- Dead-code Elimination.
- Code Motion.
- Induction Variables/Strength Reduction.

Summary

Redundancy but preserve semantics!

- Global Common Subexpressions.
- Copy Propagation.
- Dead-code Elimination.
- Code Motion.
- Induction Variables/Strength Reduction.

Summary

Redundancy but preserve semantics!

- Global Common Subexpressions.
- Copy Propagation.
- Dead-code Elimination.
- Code Motion.
- Induction Variables/Strength Reduction.

Summary

Redundancy but preserve semantics!

- Global Common Subexpressions.
- Copy Propagation.
- Dead-code Elimination.
- Code Motion.
- Induction Variables/Strength Reduction.

Summary

Redundancy but preserve semantics!

- Global Common Subexpressions.
- Copy Propagation.
- Dead-code Elimination.
- Code Motion.
- Induction Variables/Strength Reduction.

Next Week: Guest Speaker!

Peter Burka: *The Shape of an Object*

Opportunity: HACS Internship at IBM Watson Labs!

Contact: Lionel Villard *villard@us.ibm.com*

Subjects:

1. Integration of HACS and LLVM.
2. Other implementation subjects in HACS …

Project Milestone 3

- Translate JST subset to ARM32!

Questions?