CS 4120
Introduction to Compilers

Ross Tate
Cornell University

Lecture 32: Loop Optimizations I

Loop-Invariant Code Motion

```
while (i < 100) {
    y := x % r;
    y := i * y;
    i := i + y;
}
```

=>

```
if (i<100) {
    c := x%r
    y := i*c;
    i := i+y
    while (i<100) {
        y := i*c;
        i := i+y;
    }
}
```


Loop-Invariance Analysis

```
while (i < 100) {
    y := x % r;
    y := i * y;
    i := i + y;
}
```

for each var v,
invariant ≤ variant
f(m[v→invar])
v := e
= m[v→val_m(e)]

Induction Variables

```
while (i < 10) {
    j = j + 2;
    if (j > 4)
        i = i + 1;
    i = i - 1;
    k = j + 10;
    l = k * 4;
    m = i * 8;
}
```

basic induction variables
linear induction variable
derived induction variables
l = 4*j + 40

Induction-Variable Strength Reduction

```
k = j + 10;
while (i < 10) {
    j = j + 2;
    k = k + 2
    if (j > 4)
        i = i + 1;
    i = i - 1;
    k = j + 10;
    l = k * 4;
    m = i * 8;
}
return l + m;
```


Induction-Variable Strength Reduction

```
l = 4*j + 40
while (i < 10) {
    j = j + 2;
    l = l + 4*2
    if (j > 4)
        i = i + 1;
    i = i - 1;
    k = j + 10;
    l = k * 4; = 4*j + 40
    m = i * 8;
}
return l + m;
```

Induction-Variable Strength Reduction

```
while (i < 10) {
      j = j + 2;
      if (j > 4)
            i = i + 1;
      i = i - 1;
      k = j + 10;
      l = k * 4;
      m = i * 8;
}
return l + m;
```


Induction-Variable Elimination

```
j = 2 * i;
while (i < 10) {
      i = i + 1;
      j = j + 2;
      k = k + j * j;
}
return k;
```