CS620T: Compilers - Principles and Implementation

Autumn 2026

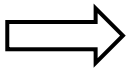
Nikhil Hegde

Week11: Dependence Analysis and Loop Optimizations

- Refer to separate set of slides for Dependence Analysis.

Loop optimization techniques - Unrolling

```
for(int i=0;i<n;i++) {
        sum += a[i]
}
```



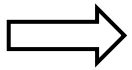
```
for(int i=0;i<n;i+=2) {
    sum1 += a[i]
    sum2 += a[i+1]
}
..
sum = sum1 + sum2
```

- Why do this?
  - Avoid branches, which are costly ops
  - Expose Instruction-Level Parallelism (ILP)
    - Modern systems have hardware that can utilize available ILP
- Pros and Cons?
  - Cons: increases code, register pressure, sometimes complicates CFG
  - Pros: may avoid branching, expose invariants, allow parallelization and other optimizations

Loop optimization techniques - Unrolling

- Loop Peeling

```
2   void bar(int);
3   void foo() {
4     int a = 0;
5     int y = 0;
6     int x = 0;
7     for(int i = 0; i <100; ++i) {
8       bar(a);
9       x = y;
10      y = a + 1;
11      a = 5;
12    }
13    bar(x);
14  }
```



```
6     //iteration0
7     bar(a);
8     x = y;
9     y = a + 1;
10    ++i;
11    //check if i < 100;
12    //iteration1
13    bar(a);
14    x = y;
15    y = a + 1;
16    ++i;
17    //check if i < 100;
18    //iteration2
19    bar(a);
20    x= y
21    y = a + 1;
22    ++i;
23    //check if i < 100;
24
25    for(int i = 3; i <100; ++i) {
```

Loop optimization techniques - Unrolling

- Loop Peeling

Instn/iter	0	1	2	3
bar(a)	bar(0)	bar(5)	bar(5)	bar(5)
x=y+1	x=1	x=2	x=7	x=7
y=a+1	y=1	y=6	y=6	y=6
a=5	a=5	a=5	a=5	a=5

```
2   void bar(int);
3   void foo() {
4     int a = 0;
5     int y = 0;
6     int x = 0;
7     for(int i = 0; i <100; ++i) {
8       bar(a);
9       x = y;
10      y = a + 1;
11      a = 5;
12    }
13    bar(x);
14  }
```

- Discover invariants and avoid phi nodes
  - Demo: `make looppeel`
  - Try it yourself: change `x=y` to `x=y+1` and observe .png files. Why do you see that phi node for x has not been eliminated?
- Eliminate If conditions that exist inside loops
  - Demo: `make looppeel2`

Loop optimization techniques - Unrolling

- Full Unroll
  - Unroll the loop fully or don't unroll
- Loop Unroll
  - First try Full Unroll, if not successful, do partial unroll / runtime unroll
- Partial Unroll
  - Cannot do this before inlining (demo: `make unroll1`)
  - Runtime unroll (demo: `make unroll2`)
    - Try it yourself: remove the `-unroll-runtime -unroll-count=4` arguments and observe the .png files