Compiler Design RA1911003010143

Assignment5 Abhishek kumar

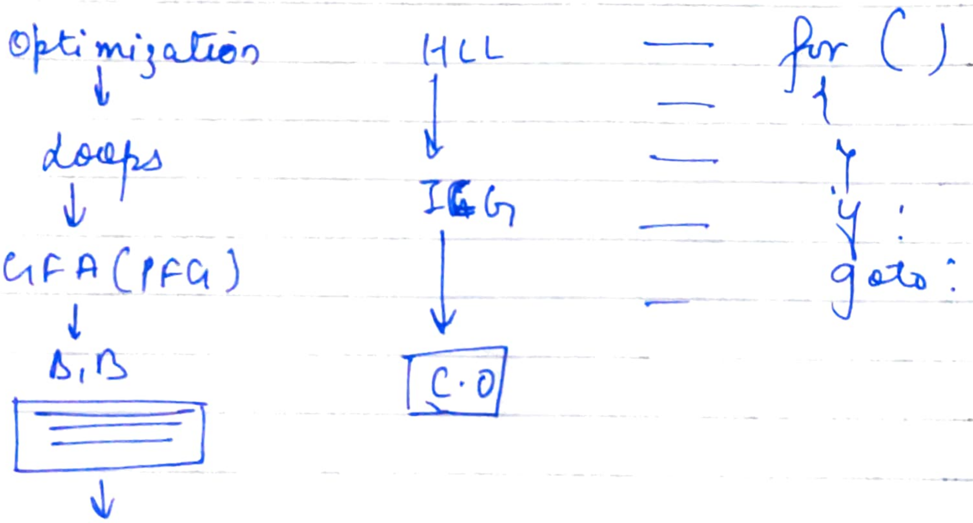
* Machine Independent optimization

* loop optimization

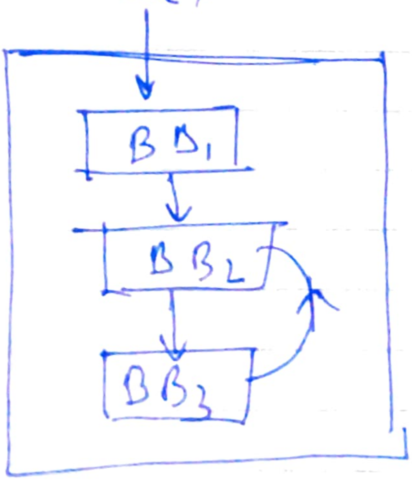
→ To apply loop optzymation, we must first detect loops.

→ for detecting loops we control flow analysis (CFA) using program flow graph (PFG)

→ To find PFG we need to find basic blocks.



Note : for loop is not visible at ICG. At ICG even if, switch statement have 'goto' hence,



→ cycle indirect a loop

* finding the basic blocks

→ In order to find basic blocks, we need to find the leaders in the program then a basic block will start from one leader to the next leader but not including next leader.

* Identifyiy leader in basic block :

1) 'If' statement is a leader
2) Statement that is target of conditional or unconditional statement is leader.
   ex. if (....) goto (Statement 2)
   ↳ leader
   goto (statemt 6)
   ↳ leader

3) Statemet that follows immediatly a condional or unconditional statemit is leader

→ ex:

code :

```
fact (x)
{   if f = 1
  for (i = 2; i <= n ; i++)
      f = f * i ;
  return f ;
}.
```

Convert to 3address code.

```
① f = 1      } B1
② i = 2
③ if (i>n) goto (9)  } B2
④ t = f * i ;
⑤ f = ti ;
⑥ t2 = i + 1 ;        } B3.
⑦ i = t2 ;
⑧ goto (3)

⑨ goto (return) } B4
```

* optimizations on loop

(a) frequency reduction moving the code from high frequeny to low frequeny region is called Code motion.

ex:-

```
        while (i< 5000)
        {
            A = sin(x)/cos(x) *i;
            i++;
        }
              ↓
        t = sin(x)/cos(x); (only once calculated stored)
        while (i<5000)
            { A = t*i;
              i++;
            }
```

(b) loop unrolling : Taking lesser iterations

ex:

```
        while (i<10)
        { x[i++] = 0;
          x[i++] = 0; }
```

(c) loop jamming : Combining two loops & hence reducing no. of loops.

ex.
```
for ( i=0 ; i<10 ; i++)
  for ( J =0 ; J<10, J++)
    x [i, j] =0;

         ⇓

for ( i = 0; i<10; i++)
 for ( J=0 ; J<10 ; J++)
  {     x[i,j] = 0;
        x[i,j] = 0;
  }
```

1. Note :- HLL is used to demonstrate & the obtimized above stats. , but actually only three laddress Code are used at this stage.

2. folding :- replacing an expression that can be computed at compile time by its value

3. Redundancy elimination :- (DAG - directed acycle graph)

$A = B+C$

$D = 2+B+3+C$

4. Strength Reduction : Replacing a costly operation by cheaper one.

ex. $B = A*2 \Rightarrow B = A<<1$

Algebric Simplification:

$$\left.\begin{array}{l} A = A+0 \\ n = n*1 \end{array}\right\}$$ eliminate trivial statement.