Compiler Design

Today's Class Topics

-SSA and DAG
-Basic Block
-Control flow graph
-Loop optimization
-Machine Independent optimization
-Peephole Optimization

Compiler Design

3-Address Code for Array:-

width = for int = 2 byte

1-D Array:- $A[i] = Base(A) + (i - low) * w$

$= (i * w) + Base(A) - low \times w$ → Byte

$= Base(A) + i * w - low * w$

$t_1 = i * w$

$t_2 = t_1 + C$

t_1 | C

t_2

$C = Base(A) - low \times w$

Ex:-

```
main()
{
  int i=0;
  int a[10];
  while(i<=10)
    a[i]=0;
}
```

a → 0 1 2 3 4 ... 10

3 Address Code:

1) $i = 0$
2) if ($i <= 10$) goto 4
3) goto 8
4) $t_1 = i * w$
5) $t_2 = Add(a)$
6) $t_2[t_1] = 0$
7) goto 2
8) ——

Compiler Design

2D-Array :- Row major order

$A[i_1, i_2] = Base(A) + ((i_1 - low_1) * n_2 + i_2 - low_2) * w$ → no. of elem

$= ((i_1 \times n_2) + i_2) * w + (Base(A) - ((low_1 \times n_2) + low_2) * w)$

t_1 t_2

Row Color

$t_1 = t_1 + t_2$

$t_2 = c$

A[i,j] A[r][c]

A[0-3][0-4]

3 Address code : t_n

$t_1 = i_1 \times n_2$

$t_1 = t_1 + i_2$

$t_2 = c$

$t_3 = w * t_1$

$t_4 = t_2[t_3]$

Ex :- Write 3-Address code for $x := A[y, z]$

y z

$t_1 = y * n_2$

$t_1 = t_1 + z$

$t_2 = c$

$t_3 = w * t_1$

$t_4 = t_2[t_3]$

$x = t_4$

Compiler Design

$A_{10 \times 20}$

Question:- Let A be a 10x20 Array with $low_1 = low_2 = 1$, therefore $n_1 = 10$, $n_2 = 20$
Take $\omega = 4$ Byte Find 3-Address code for $x := A[y, z]$

Soln:-

$t_1 = y * n_2$
$t_1 = t_1 + z$
$t_2 = C$
$t_3 = \omega * t_1$
$t_4 = t_2[t_3]$
$x = t_4$

$\Rightarrow$

where $C = Base(A) - ((low_1 \times n_2) + low_2)\,\omega$

$n_2 = 20$, $\omega = 4$,

$C = Base(A) - ((1 \times 20) + 1) * 4$

$C = Base(A) - 84$

Ans

$t_1 = y * 20$
$t_1 = t_1 + z$
$t_2 = Base(A) - 84$
$t_3 = 4 * t_1$
$t_4 = t_2[t_3]$
$x = t_4$

Compiler Design

$X[i*1024 + j*32 + k*4]$

$X[(i-0)*32*8 + (j-0)*8 + (k-0)]*4$

$X[(i*32*8 + j*8 + k)4]$

3D-Array:-

$$Loc[a[i][j][k]] = B.A + [(i-lb_1)*n_r*n_c + (j-lb_2)*n_c + (k-lb_3)]*w$$

where $n_a = ub_1 - lb_1 + 1$, $n_r = ub_2 - lb_2 + 1$, $n_c = ub_3 - lb_3 + 1$

Question:- for a C-program accessing $X[i][j][k]$, the following intermediate code is generated by a compiler assume that the size of an integer is 32-bits and the size of a character is 8 bits

$w = 4$ Byte

$t_0 = i*1024$
$t_1 = j*32$
$t_2 = k*4$
$t_3 = t_1 + t_0$
$t_4 = t_3 + t_2$
$t_5 = X[t_4]$

Which one of the following statements about the source code for the C-program is correct

$X[i*1024 + j*32 + k*4]$

a) X is declared as "int X[32][32][8]"
b) " " "int x[4][1024][32]"
c) " " "char X[4][32][8]"
d) " " "char X[32][16][2]"

$t_5 = X[t_4] = X[t_3 + t_2] = X[t_1 + t_0 + t_2] = X[t_0 + t_1 + t_2]$

Compiler Design

8)
```
int A[10], B[10];
int x=0, i;
for(i=0; i<10; i++)
{
    x = x + A[i]*B[i];
}
```

$t_8 = x + t_7$ (i >= 10) t_7

w=2

$t_1 = i * w$ =

w=4

t_2 = Base(A) − low × w

$t_3 = A[i] = t_2[t_1]$

$t_6 = B[i] = t_5[t_4]$

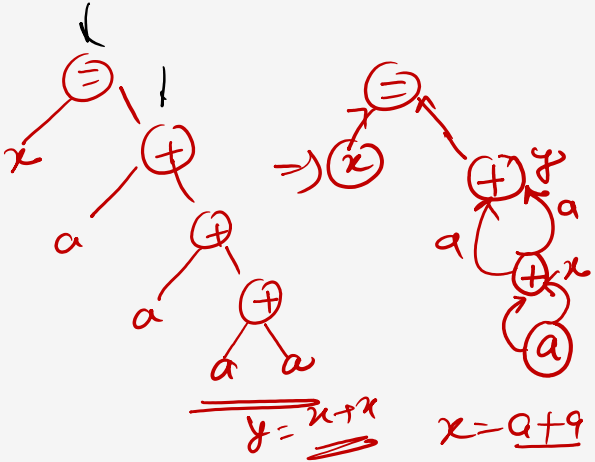
```
1)  x = 0
2)  i = 0
3)  if (i >= 10) goto 15
4)  t1 = i * 2
5)  t2 = Base(A)
6)  t3 = t2[t1]
7)  t4 = i * 2
8)  t5 = Base(B)
9)  t6 = t5[t4]
10) t7 = t3 * t6
11) t8 = x + t7
12) x = t8
13) i = i + 1
14) goto 3
15)
```

```
1) x = 0
2) i = 0
3) if (i < 10) goto -
4) goto 15
5) ——
```

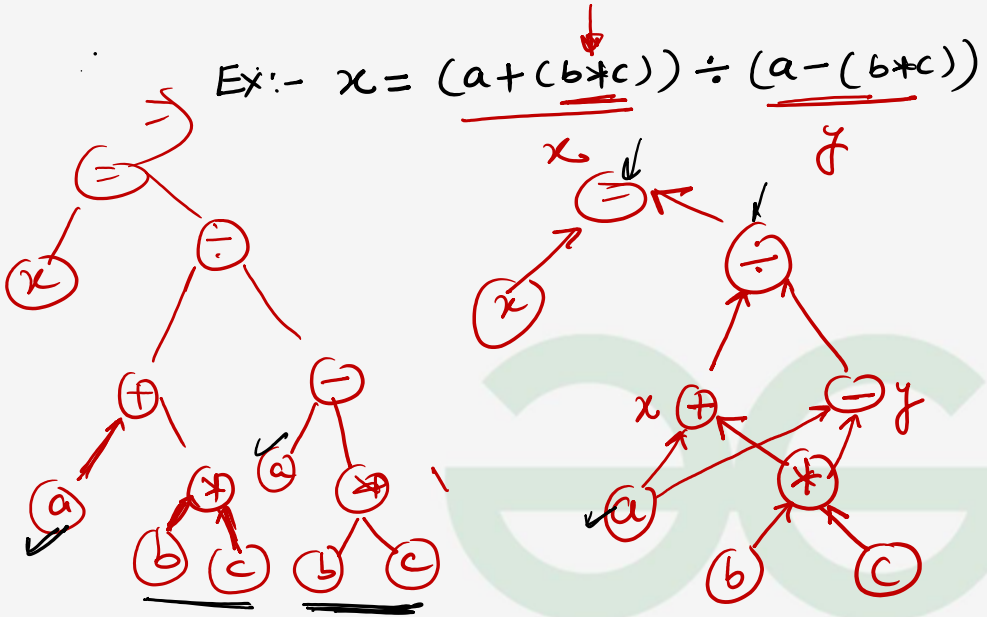
Compiler Design

Directed Acyclic Graph (DAG):- used to Eliminate Common Subexpression elimination

Ex:- $x = a + a + a + a$



Ex:- $x = (a + (b * c)) \div (a - (b * c))$



Compiler Design

Question:- Find the mini No. of nodes and edges in DAG for the following code segment

1)
```
a = b+c
d = b+c
e = a+1
f = a+1
g = e+f
```

⇒

```
a = b+c
d = a
e = a+1
f = e
g = e+f
```

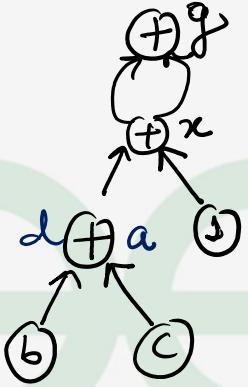
⇒

```
a = b+c
d = a
e = a+1
f = e
g = e+e
```

Bottom-up

6 nodes & 6 edges

DAG:



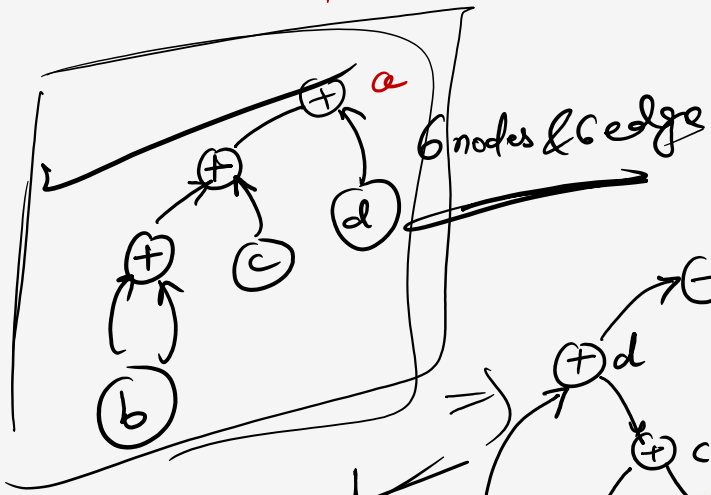
$g = e+e = a+1 + a+1 \Rightarrow \underbrace{b+c+1}_{x} + \underbrace{b+c+1}_{x}$

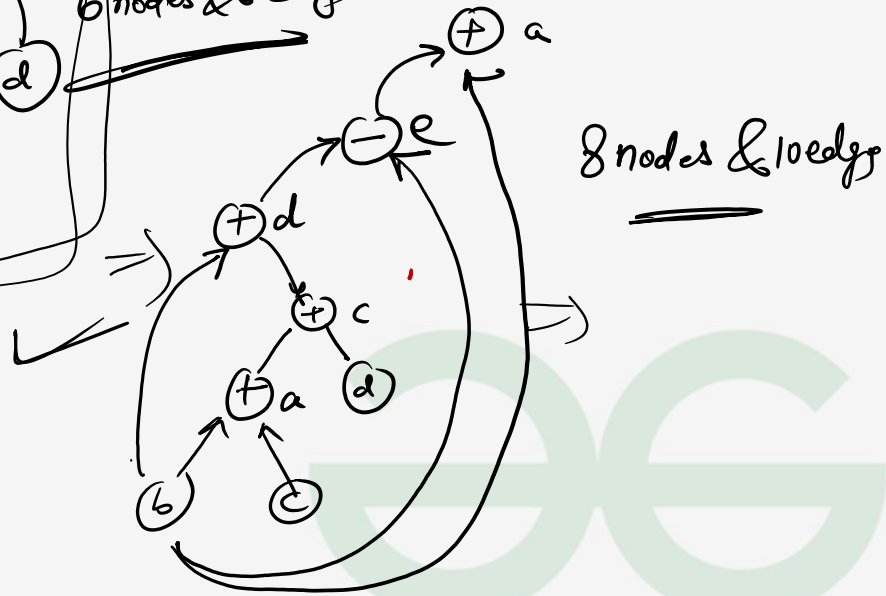
$g = x+x = b+c+1 + b+c+1$

Compiler Design

2) $a = b+c$
$c = a+d$
$d = b+c$
$e = d-b$
$a = e+b$

$a = e+b = d-b+b \Rightarrow d = b+c = b+a+d$
$\Rightarrow b + b + c + d$





$a = b+c$
$c = a+d$
~~$d = b+c$~~
$e = d-b$
$a = e+b$

Compiler Design

3) mini No of nodes & edges.

$a = c + b$

$b = b - d$

$c = b + d$

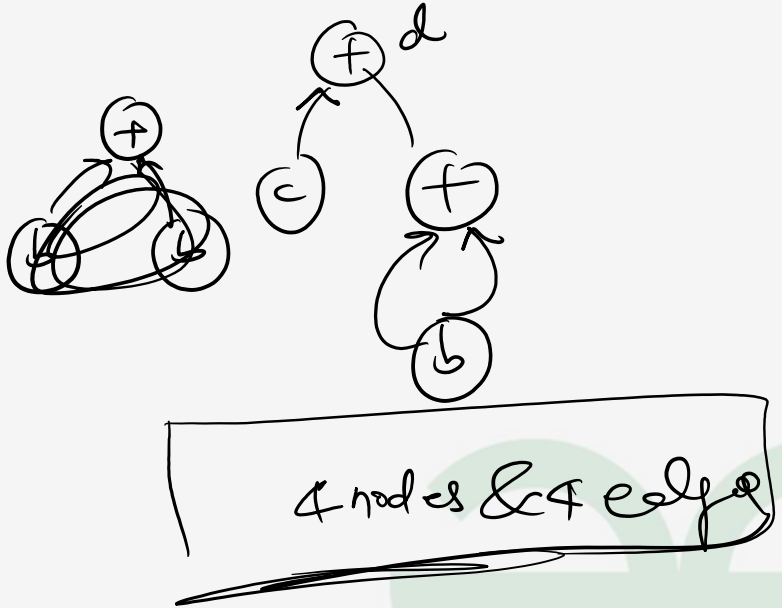
$d = a + c$

$d = a + c$

$= a + b + d$

$\Rightarrow a + b - d + d$

$= c + b + b$



Compiler Design

DAG:

minimode & minedges = ?

4) $a = b + c$

$b = a - b$

$c = d + b$

$b = c + d$

$d = b + c$

5 nodes & 6 edges

$d = b + c$

$= c + d + c$

$= d + b + d + d + b$

$= d + a - b + d + d + a - b$

$\Rightarrow d + b + c - b + d + d + b + c - b$

$\Rightarrow d + c + d + d + c$

Compiler Design

Implementation of 3-Address Code:-

$x = ((a+b)-(c*d))*(a+b-c)$

3 Address code:

$t_1 = a+b$

$t_2 = c*d$

$t_3 = t_1 - t_2$

$t_4 = t_1 - c$

$t_5 = t_3 * t_4$

$x = t_5$

Quadruple

SN	operator	Arg1	Arg2	Result
1	+	a	b	t_1
2	*	c	d	t_2
3	-	t_1	t_2	t_3
4	-	t_1	c	t_4
5	*	t_3	t_4	t_5
6	=	t_5		x

Adv: Statements can be moved around

DisAdv: more temporary variable need more memory

Triple

SN	operator	Arg1	Arg2
1	+	a	b
2	*	c	d
3	-	(1)	(2)
4	-	(1)	c
5	*	(3)	(4)
6	=	x	(5)

Adv: less memory

DisAdv: Statement can't be moved Around

Indirect Triple

100	(1) (2)
101	(2) (1)
102	(3)
103	(4)
104	(5)
105	(6)

Statement can be moved around

Compiler Design