Compiler Design

Today's Class Topics

- Intermediate code
- 3 Address code
- SSA and DAG

Compiler Design

if (a > b) goto ...

3-Address Code:-

$a = b \; op \; c$

$a = op \; b$

$a = b$

Boolean.

if (Condition) Goto L (Conditional jump)

Goto L (Unconditional jump)

Compiler Design

Array

```
a = b[e];
a[e] = b
```

Pointers

```
a = *b;
*a = b
a = &b
```

Compiler Design

Question:- Write a 3-Address code for the following code

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

$t_1 = a+b$

$t_2 = c*d$

$t_3 = t_1 - t_2$

$t_4 = t_1 - c$

$t_5 = t_3 * t_4$

$x = t_5$

2) $x = a+b*c-d$, where x, a are real and b, c, d are integers

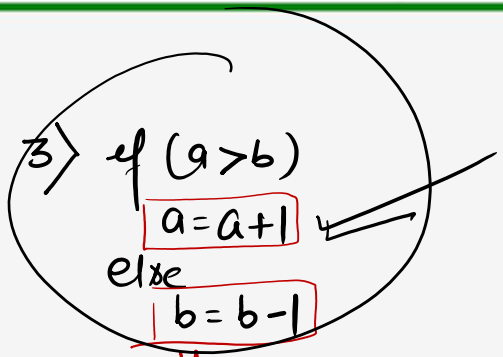
$t_1 = b*c$

$t_2 = t_1 - d$

$t_3 = \text{inttoreal}(t_2)$

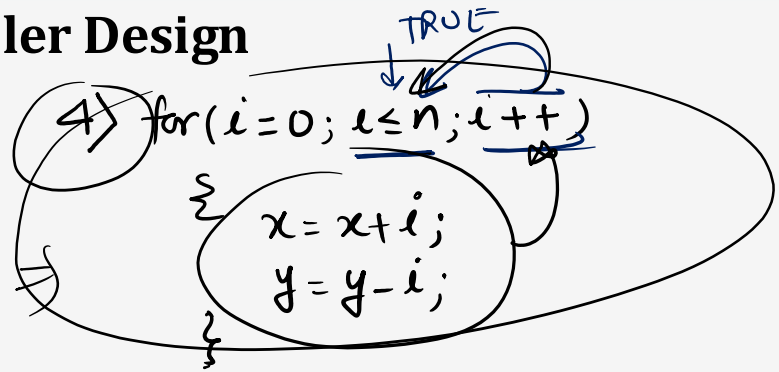
$x = a + t_3$

Compiler Design



3 Add

1) if (a>b) goto 4
2) b=b-1
3) Goto 5
4) a=a+1
5) ——



1) i=0
2) if (i>n) goto 7
3) x=x+i;
4) y=y-i;
5) i=i+1;
6) goto 2
7) ——

Compiler Design

5) if((a>b) && (c>d))
x = x+1
else
y = y+1

1) if (a>b) goto 4
2) y = y+1
3) goto 8
⇒ 4) if (c>d) goto 7
5) y = y+1
6) goto 8
7) x = x+1
8) ——

Backpatching:- Leaving the labels and filling them after is called as Backpatching

Compiler Design

7) Switch (i)

```
{
Case 1:
    x1 = a1 + b1 * c1;
        break;
Case 2:
    x2 = a2 + b2 * c2;
        break;
default:
    x3 = a3 + b3 * c3;
        break;
}
```

3 Address code:

1) if (i == 1) goto 7
2) if (i == 2) goto 11
3) $t_1 = b_3 * c_3$
4) $t_2 = a_3 + t_1$
5) $x_3 = t_2$
6) goto 14
7) $t_1 = b_1 * c_1$
8) $t_2 = a_1 + t_1$
9) $x_1 = t_2$
10) goto 14
11) $t_1 = b_2 * c_2$
12) $t_2 = a_2 + t_1$
13) $x_2 = t_2$
14) _____

Compiler Design

3-Address Code for Array:-

$t_2[t_1] = t_2 + t_1$

1-D Array:-

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

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

$t_2 = Base(A) - low \times w$

$t_2 = Base(A) - 0 * w$

$t_2 = Base(A)$

$t_1 = i * w$

$t_2 = t_1 + C$

$t_1 + C = t_2$

Ex:-

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

3 Address Code:

```
1) i = 0
2) if (i<=10) goto 4
3) goto 9
4) t1 = i * w
5) t2 = Add(A)
6) t2[t1] = 0
7) i = i+1
8) goto 2
9) ——
```

Compiler Design

2D-Array:- RMO

$$A[i_1, i_2] = Base(A) + ((i_1 - low_1) * n_2 + i_2 - low_2) * w$$

no. of colm.

$$A[y, z] = ((i_1 \times n_2) + i_2) * w + (Base(A) - ((low_1 \times n_2) + low_2) * w)$$

t_1 variable, C

3 Address code:

$t_1 = i_1 \times n_2$

$t_1 = t_1 + i_2$

$t_2 = c$

$t_3 = w * t_1$

$x := t_4 = t_2[t_3]$

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

$t_1 = y \times n_2$

$t_1 = t_1 + z$

$t_2 = c$

$t_3 = w * t_1$

$t_4 = t_2[t_3]$

$x = t_4$

Compiler Design

Question:- Let A be a 10×20 Array with $low_1 = low_2 = 1$, therefore $n_1 = 10$, $n_2 = 20$

Take $w = 4$ Byte Find 3-Address code for $x := A[y, z]$

Soln:-

$t_1 = y * n_2$
$t_1 = t_1 + z$
$t_2 = c$
$t_3 = w * t_1$
$t_4 = t_2[t_3]$
$x = t_4$

$\Rightarrow$

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

$t_2[t_3] = t_2 + t_3$

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

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

$C = Base(A) - 84$

Compiler Design

3D-Array:- RMO

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

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, $w = 8$, $w = 4$

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

Which one of the following statements about the source code for the C-program is correct $X[(i * 32 * 8 + j * 8 + k) 4]$

a) X is declared as "int X [32] [32] [8]" $\Rightarrow X[1024*i + 32*j + 4k]$
b) " " "int x [4] [1024] [32]"
c) " " "char X [4] [32] [8]"
d) " " "char X [32] [16] [2]"

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

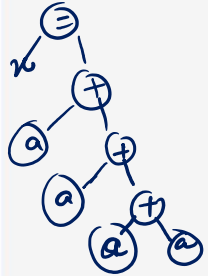
Compiler Design

Directed Acyclic Graph (DAG):-

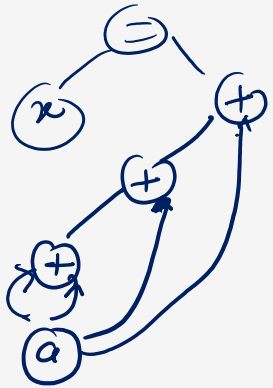
1) Leaf node is created once, we will create use of that leaf node instead of creating the same leaf node again.
2) Internal nodes are operators and leaves are either variable or Constants
3) DAG is used eliminate the Common Subexpression

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

Syntax tree

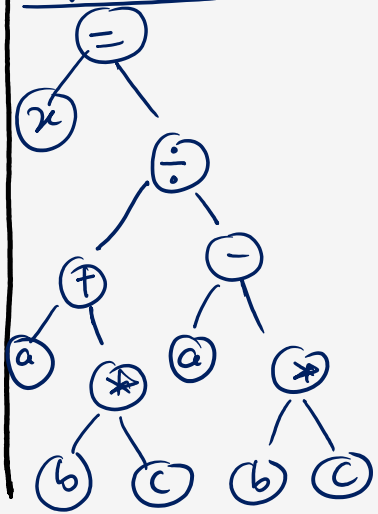


DAG:

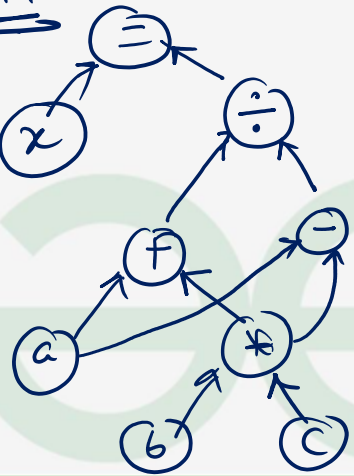


Syntax tree

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



DAG



Compiler Design

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

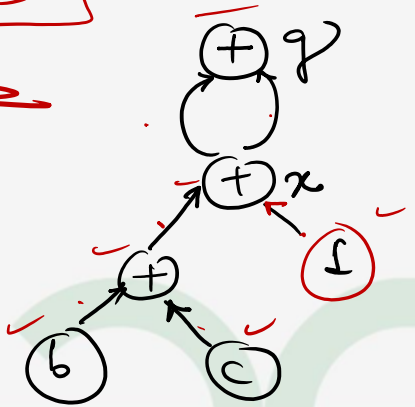
1) $a = b + c$
$d = b + c = a$ =>
$e = a + 1$
$f = a + 1 = e$
$g = e + f$

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

up

Bottom

6 nodes
6 edges



$g = e + f = e + e \Rightarrow a+1 + a+1$

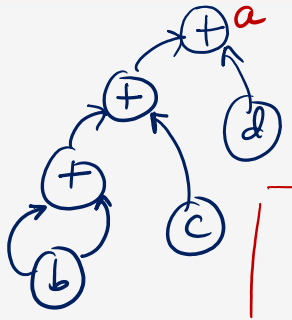
$\Rightarrow \underbrace{b+c+1}_{x} + \underbrace{b+c+1}_{x}$

Compiler Design

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

Bottom ↑

mini DAG



6 nodes
6 edges

Always Bottom to up

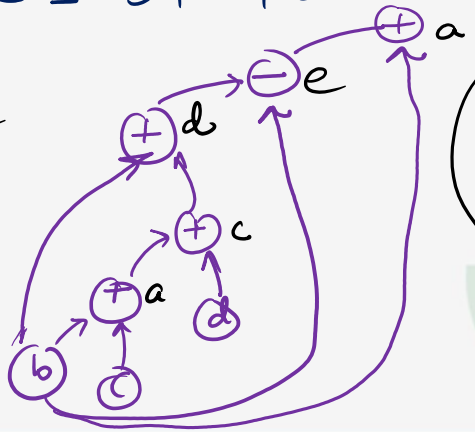
$a = e + b = d - b + b = b + c = b + a + d = b + b + c + d$

Top to Bottom

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

Top ↓ Bottom

DAG



8 nodes
10 edges

Compiler Design

4) min No of nodes & edges = ?

$a = b + c$

$b = a - b$

$c = d + b$

$b = c + d$

$d = b + c$

Bottom to up

5 nodes
6 edges

Ans

$d = b + c = c + d + c \Rightarrow d + b + d + d + b$

$\Rightarrow d + a - b + d + d + a - b$

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

$\Rightarrow \boxed{d + c + d + d + c}$

Compiler Design

Implementation of 3-Address Code:-

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

3 Address code:-

$t_1 = a+b$ ✓

$t_2 = c*d$ ✓

$t_3 = t_1 - t_2$ ✓

$t_4 = t_1 - c$

$t_5 = t_3 * t_4$ ✓

$x = t_5$

Quadrapule:

SN	operator	Arg 1	Arg 2	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

1) $t_2 = c*d$

2) $t_4 = a+b$

3) $t_3 = t_1 - t_2 \Rightarrow (a+b) - (c*d)$

Adv: 1) using the temporary variable we can access the value easely

2) Statements can be moved around

DisAdv: 1) more temporary variable need more memory

Compiler Design

3 Address code:-

$t_1 = a + b$
$t_2 = c * d$
$t_3 = t_1 - t_2$
$t_4 = t_1 - c$
$t_5 = t_3 * t_4$
$x = t_5$

Triple Representation:

SN	Operator	Arg 1	Arg 2
1	+	a	b
2	*	c	d
3	−	(1)	(2)
4	−	(1)	c
5	*	(3)	(4)
6	=	x	(5)

Indirect Triple:

Indirect Triple instruction

100	(1) (2)	c*d
101	(2) (1)	a+b
102	(3)	(1) − (2)
103	(4)	
104	(5)	(a+b) − (c*d) ✓
105	(6)	

Adv:

1) Takes less memory (no temporary variable)

DisAdv:

1) Statement can't be moved around.

1) c*d
2) a+b
3) (c*d) − (a+b) ✗ execution order changed

Adv: during execution statements can be move around for optimization

DisAdv: two memory Access

Compiler Design

Static Single Assignment (SSA) :-

1) Its a 3- Address Code
2) SSA form is used to represent an intermediate code
3) A variable can not be used more than once in the LHS
   i.e A variable should be initialized atmost once
4) Every variable has a Single Assignment (Definition)

Ex: 3Add'

$a = b + c$

$d = a - b$

$a = b + d$

$d = a - c$

SSA'

$a_1 = b + c$

$d_1 = a_1 - b$

$a_2 = b + d_1$

$d_2 = a_2 - c$

Compiler Design

Question:- Find minimum No. of variables to represent 3-Address code & SSA

$X = a*b + a*b$

3Address:

optimized 3-Address Code:

$a = a+b$

$a = a+a$

$\Rightarrow$

min variable required = 2 $\{a, b\}$

SSA:

$a_1 = a*b$

$a_2 = a_1 + a_1$

min variables required in SSA = 4 $\{a, b, a_1, a_2\}$

Ans

Compiler Design

Question: 3 Address code:

$t_1 = a * a$

$t_2 = a * b$

$t_3 = t_1 * t_2$

$t_4 = t_3 + t_2$

$t_4 = t_3 + t_2 = (t_1 * t_2) + a * b$

$= (a * a) * (a * b) + a * b$

$a = a * b$ $\quad b = a * b$

```
b = a*b
a = a*a
a = a*b
a = a+b
```

i) How many minimum of variables in equivalent 3 Address code:

```
b = a*b
a = a*a
a = a*b
a = a+b
```

mini variables requires = 2 $\{a, b\}$

ii) Mini. No of variable in equivalent SSA ?

$b_1 = a * b$

$a_1 = a * a$

$a_2 = a_1 * b_1$

$a_3 = a_2 + b_1$

mini. variable required in SSA = 6 variables $\{a, b, a_1, a_2, a_3, b_1\}$

Compiler Design

Question:- SSA = 9 variables

$t_1 = b * c$
$t_2 = a + t_1$
$t_3 = b * c$
$t_4 = d * t_3$
$t_5 = t_2 + t_4$

i) Mini variables in 3Address Code ?

ii) Mini variable in SSA Code ?

$t_5 = t_2 + t_4$
$= (a + t_1) + (d * t_3)$
$= (a + (b * c)) + (d * (b * c))$

$b = b * c$
$a = a + b$
$b = d * b$
$a = a + b$

⇒

SSA:

$b_1 = b * c$
$a_1 = a + b_1$
$b_2 = d * b_1$
$a_2 = a_1 + b_1$

4 variables required in mini 3Addcode (a, b, c, d)

mini variable required SSA = 8 {a, b, c, d, a_1, a_2, b_1, b_2}

Compiler Design

HW

Question:- Consider the following code segment

$x = u - t$

$y = x * v$

$x = y + w$

$y = t - z$

$y = x * y$

The mini. No. of total variables required to convert the above code segment to Static single Assignment form is ?

Thank You !

Compiler Design

Basic Block: Basic block is a set of statements that always executes in a sequence one after the other.

The characteristics of basic blocks are-

-They do not contain any kind of jump statements in them.
-There is no possibility of branching or getting halt in the middle.
-All the statements execute in the same order they appear.
-They do not lose the flow control of the program.

Compiler Design

How to find the Basic Blocks

1. First, find the set of leaders from intermediate code, the first statements of basic blocks. The following are the steps for finding leaders:
    1. The first instruction of the three-address code is a leader.
    2. Instructions that are the target of conditional/unconditional goto are leaders.
    3. Instructions that immediately follow any conditional/unconditional goto/jump statements are leaders.
2. For each leader found, its basic block contains itself and all instructions up to the next leader.

Compiler Design

Control Flow Graph(CFG):

Compiler Design

Question: Find the No. of nodes & edges in the CFG for the following code segment

1. $a = 1$
2. $i = 0$
3. $t_1 = a + 5$
4. $t_2 = x * a$
5. $t_3 = 4 * b$
6. $a = a + t_3$
7. $i = i + 1$
8. if $i < t_3$ goto 5
9. $J = J + 1$
10. $K = K + 1$
11. if $J < K$ goto 3
12. $i = J + K$
13. $K = 0$

Compiler Design

Question:- Find the No. of nodes & edges in the CFG for the following code segment

1- a = b
2- b = a+5
3- c = a-b
4- a = a+5
5- if a>d goto 3
6- b = c*a
7- d = b+c
8- if b<a goto 10
9- c = a-b
10- a = a*10
11- if a<100 goto (1)

Compiler Design

Question:-

```
    i = 0
L1: if(n==1) goto L4
    if(n>10) goto L2
    n = 3*n+1
    Goto L3
L2: n = n/2
L3: i = i+1
    goto L1
L4: return i
```

Compiler Design

Question:- Consider the following intermediate code given below. The no. of nodes & edges in the Control flow graph constructed for the code respectively are ?

a) 5 and 7
b) 6 and 7
c) 5 and 5
d) 7 and 8

1) $i = 1$
2) $J = 1$
3) $t_1 = 5 * i$
4) $t_2 = t_1 + J$
5) $t_3 = 4 * t_2$
6) $t_4 = t_3$
7) $a[t_4] = -1$
8) $J = J + 1$
9) if $(J <= 5)$ goto 3
10) $i = i + 1$
11) if $(i < 5)$ goto 2