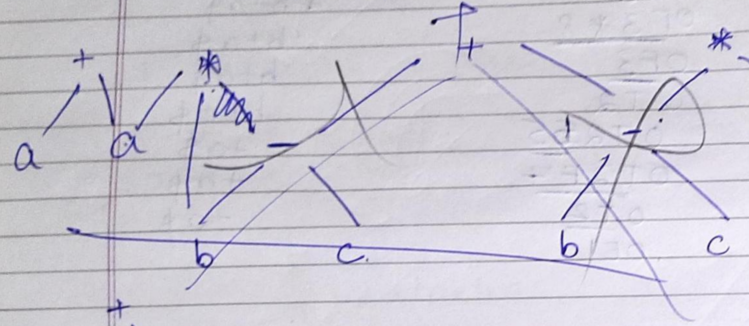
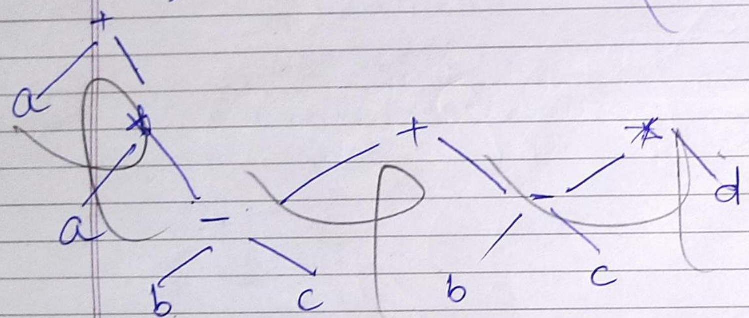
20/03/23

DAG (Directed Acyclic Graph)

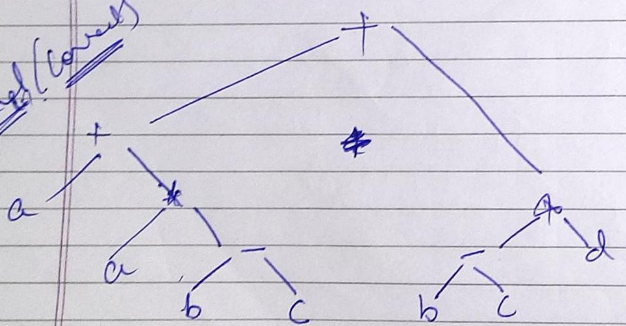
* a + a * (b − c) + (b − c) * d

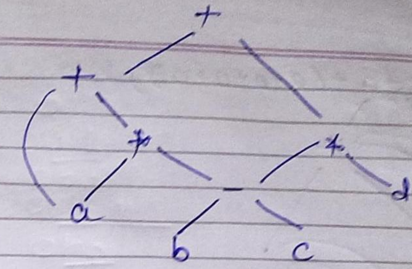
a + a(b − c) + (b − c)d





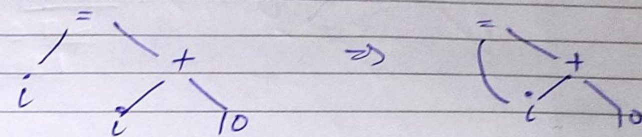
Final (correct)



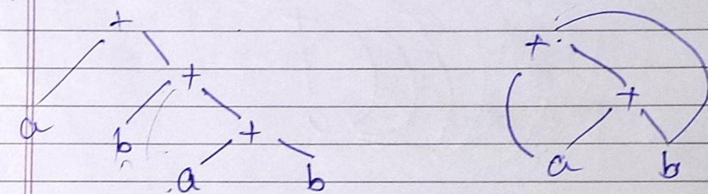


Book (Optimized version)

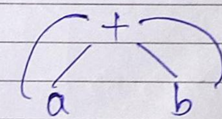
* i = i + 10



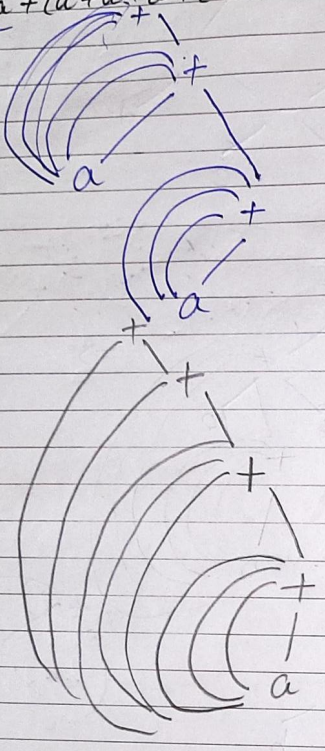
* a + b + (a + b)



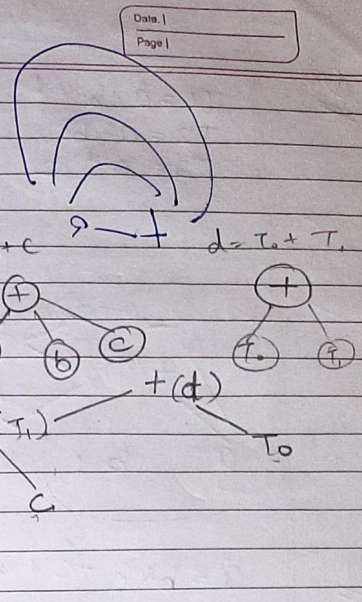
* a + b + a + b



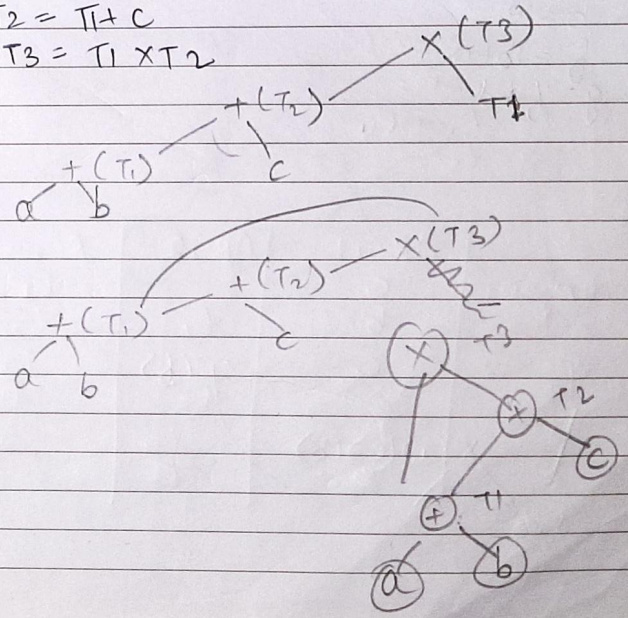
* a+a+(a+a+a+(a+a+a+a))



Q1) $T_0 = a+b$
$T_1 = T_0 + c$
$d = T_0 + T_1$

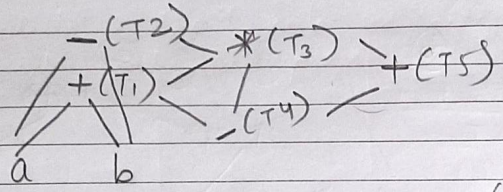


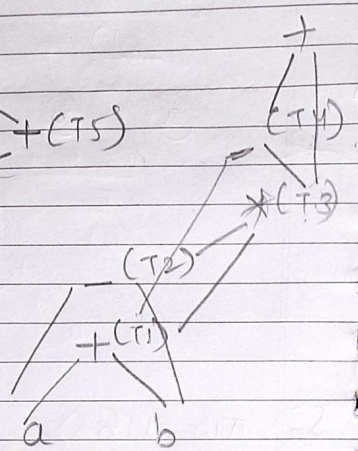
Q2) $T_1 = a+b$
$T_2 = T_1 + c$
$T_3 = T_1 \times T_2$



Q3

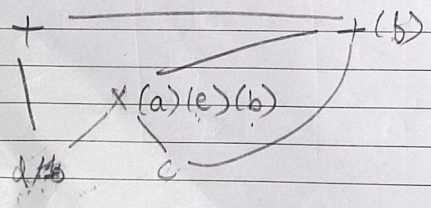
$T1 = a + b$
$T2 = a - b$
$T3 = T1 * T2$
$T4 = T1 - T3$
$T5 = T4 + T3$





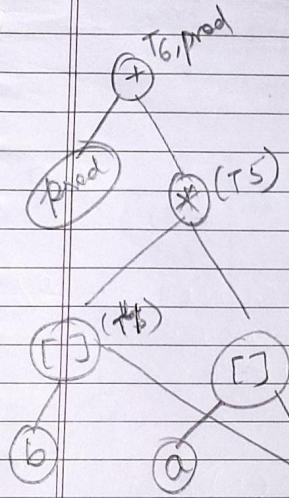
Q4

$a = b \times c$
$d = b$
$e = d \times c$
$b = e$
$f = b + c$
$g = f + d$



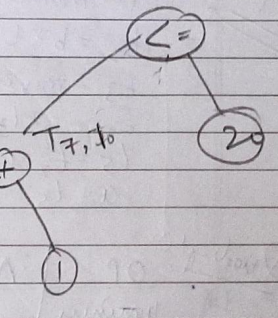
Q5

$a = b \times c$
$d = b$
$e = d \times c$
$b = e$
$f = b + c$



Q6

$T1 = 4 * I0$
$T2 = a[T1]$
$T3 = 4 * I0$
$T4 = b[T3]$
$T5 = T2 * T4$
$T6 = prod + T5$
$prod = T6$
$T7 = I0 + 1$
$I0 = T7$
if $I0 <= 20$ goto)



Three address code.

do i=i+1; while (a[i] < v);

$t_1 = i+1$
$i = t_1$

5 marks Ques. Write a short note on Three address code with example.

$a = b * -c + b * -c$

Q1.

$t_1 = \text{minus } c$
$t_2 = b * t_1$
$t_3 = \text{minus } c$
$t_4 = b * t_3$
$t_5 = t_2 + t_4$
$a = t_5$

Quadruple:

	OP	ARG1	ARG2	RESULT
(0)	uminus	c	—	t_1
(1)	*	b	t_1	t_2
(2)	uminus	c	—	t_3
(3)	*	b	t_3	t_4
(4)	+	t_2	t_4	t_5
(5)	:=	t_5	—	a

$A = -B * (C+D)$

Q1.

$T_1 := -B$
$T_2 := C+D$
$T_3 := T_1 * T_2$
$A := T_3$

Quadruple

	OP	ARG1	ARG2	RESULT
(0)	uminus	B	—	T_1
(1)	+	C	D	T_2
(2)	*	T_1	T_2	T_3
(3)	:=	T_3		A

Triple

	OP	ARG1	ARG2
(0)	uminus	B	
(1)	+	C	D
(2)	*	(0)	(1)
(3)	:=	A	(2)

Indirect Triple

	STATEMENT
(0)	(14)
(1)	(15)
(2)	(16)
(3)	(17)

	OP	ARG1	ARG2
(14)	uminus	B	—
(15)	+	C	D
(16)	*	(14)	(15)
(17)	:=	A	(16)

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

$-(a \times b) + (c + d) - (a + b + c + d)$

Q1

Triple	OP	ARG1	ARG2
(0)	uminus	c	
(1)	*	b	(0)
(2)	uminus	c	
(3)	*	b	(2)
(4)	+	(1)	(3)
(5)	=	a	(4)

Indirect triple

	Instructions STATEMENT
(14)	(0)
(15)	(1)
(16)	(2)
(17)	(3)
(18)	(4)
(19)	(5)

	OP	ARG1	ARG2
(0)	uminus	c	
(1)	*	b	(0)
(2)	uminus	c	
(3)	*	b	(2)
(4)	+	(1)	(3)
(5)	=	a	(4)

1-mark Ques. Short-circuit code?

Ques $-(a \times b) + (c + d) - (a + b + c + d)$

$t_1 = a \times b$
$t_2 = -t_1$
$t_3 = c + d$
$t_4 = t_2 + t_3$
$t_5 = a + b$
$t_6 = t_5 + t_3$
$t_7 = t_4 - t_6$
$A = t_7$

Quadruple

	OP	ARG1	ARG2	RESULT
(0)	×	a	b	t_1
(1)	−	t_1		t_2
(2)	+	c	d	t_3
(3)	+	t_2	t_3	t_4
(4)	+	a	b	t_5
(5)	+	t_5	t_3	t_6
(6)	−	t_4	t_6	t_7
(7)	=	t_7		A

Triple.

	OP	ARG1	ARG2
(0)	x	a	b
(1)	-	(0)	
(2)	+	c	d
(3)	+	(1)	(2)
(4)	+	a	b
(5)	+	(4)	(2)
(6)	-	(3)	(5)
(7)	=	A	(6)

Indirect triple.

	Instructions
(11)	(0)
(12)	(1)
(13)	(2)
(14)	(3)
(15)	(4)
(16)	(5)
(17)	(6)
(18)	(7)

	OP	ARG1	ARG2
(0)	x	a	b
(1)	-	(0)	
(2)	+	c	
(3)	+	(1)	d
(4)	+	a	(2)
(5)	+	(4)	b
(6)	-	(3)	(2)
(7)	=	A	(5)
			(6)

Ques If A<B then 1 else 0

Ques If A<B and C<D then t=1 else t=0

Q Three-address code

Q Creation of basic Blocks & DAG

Q Optimisation Techniques — Peephole, Generalized

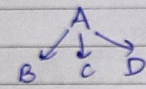
12/April/2023 Thursday

Synthesized and Inherited Attributes

Synthesized

Bottom-up approach

A → BCD

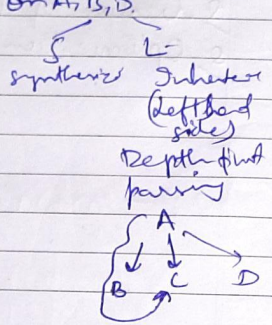


A depends on B, C, D

Inherited

A → BCD

Attributes of C depend on A, B, D



BACKPATCHING

L1: label

if A > B goto L1 else L2

L1:

L2:

```
# include < >

void main
{
goto — L1

L1. . .
}
```

makelist
truelist
falselist

BASIC BLOCKS

ALGO: Partitioning three-address instructions into basic blocks.

Rules for finding leaders:-

1.) The first three-address instruction in the intermediate code is a leader.
2.) Any inst^n that is the target of a conditional or unconditional jump is a leader.
3.) Any inst^n that immediately follows a conditional or unconditional jump is a leader.

L1 → (1) i = 1
L3 → (2) j = 1
L2 → (3) t1 = 10 * i
(4) t2 = t1 + j
(5) t3 = 8 * t2
(6) t4 = t3 - 88
(7) a[t4] = 0.0
(8) j = j + 1
(9) if j <= 10 goto (3)
L5 (10) i = i + 1

(11) if i<=10 goto (2)

L6 (12) i=1

L4 → (13) t5=i-1

(14) t6=88*t5

(15) a[t6]=1.0

(16) i=i+1

(17) if i<=10 goto (13)

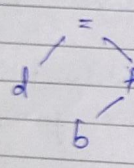
13/April/2023 **DAG** Thursday

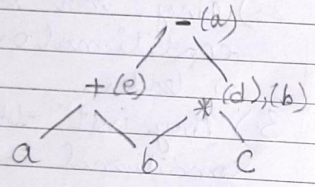
d=b*c

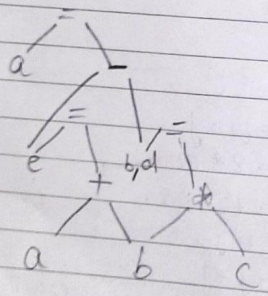
e=a+b

b=b*c

a=e-d







Peephole Optimization

(1) Eliminating Redundant Loads and stores

```
LD R0,a
ST a,R0
```

(2) Eliminating Unreachable Code

```
if debug == 1 goto L1
goto L2
L1: print debugging information
L2:
```

(3) Flow-control optimization

Replace sequence `goto L1` / `L1: goto L2` by `goto L2` / `L1: goto L2`

(4) Algebric Simplification & Reduction in Strength

x = x + 0

OR

x = x * 1

(5) Use of Machine Idioms

Optimization of Basic Blocks

(a) The DAG Representation of Basic Blocks
(b) finding local common sub expression

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

(c) Dead Code Elimination
(d) The use of Algebric Identities

$x^2 = x * x$

(e) Representation of Array References

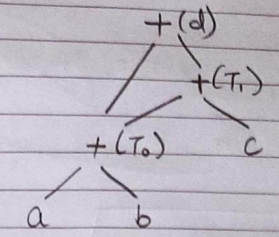
$x = a[i]$
$a[j] = y$
$z = a[i]$

(f) Pointer Assignment & Procedure call

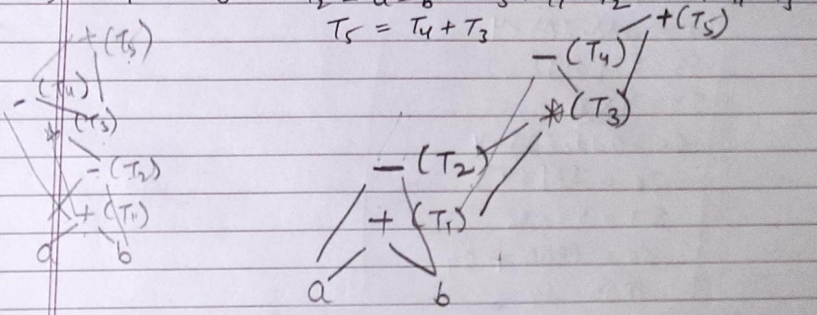
$x = *p$
$*q = y$

DAG

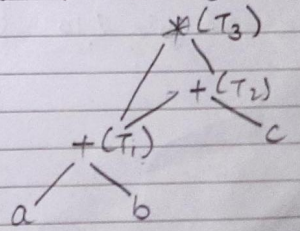
1) $T_0 = a + b$
$T_1 = T_0 + c$
$d = T_0 + T_1$



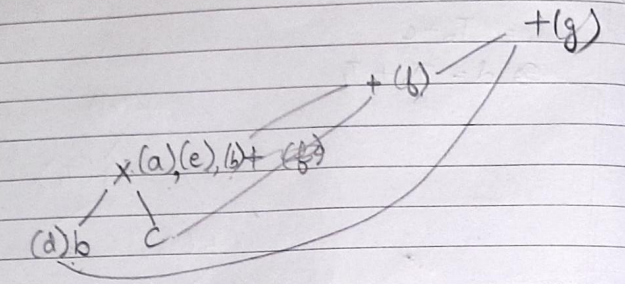
2) $T_1 = a + b$ $T_2 = a - b$ $T_3 = T_1 * T_2$ $T_4 = T_1 - T_3$
$T_5 = T_4 + T_3$



3) $T_1 = a + b$ $T_2 = T_1 + c$ $T_3 = T_1 * T_2$



4) a=b×c , d=b , e=d×c , b=e , f=b+c , g=f+d



Basic Blocks

L1 → S1 = 4 × I
S2 = addr(A) − 4
S3 = S2[S1]
S4 = 4 × I
S5 = addr(B) − 4
S6 = S5[S4]
S7 = S3 × S6
S8 = PROD + S7
PROD = S8

L2 → Si = I + 1
I = S9
if I <= 20 goto L10