Compiler Design

Question: SDD for Count the No. of zero's

$L \rightarrow L\ B \quad \{ L.val = L_1.val + B.val \}$

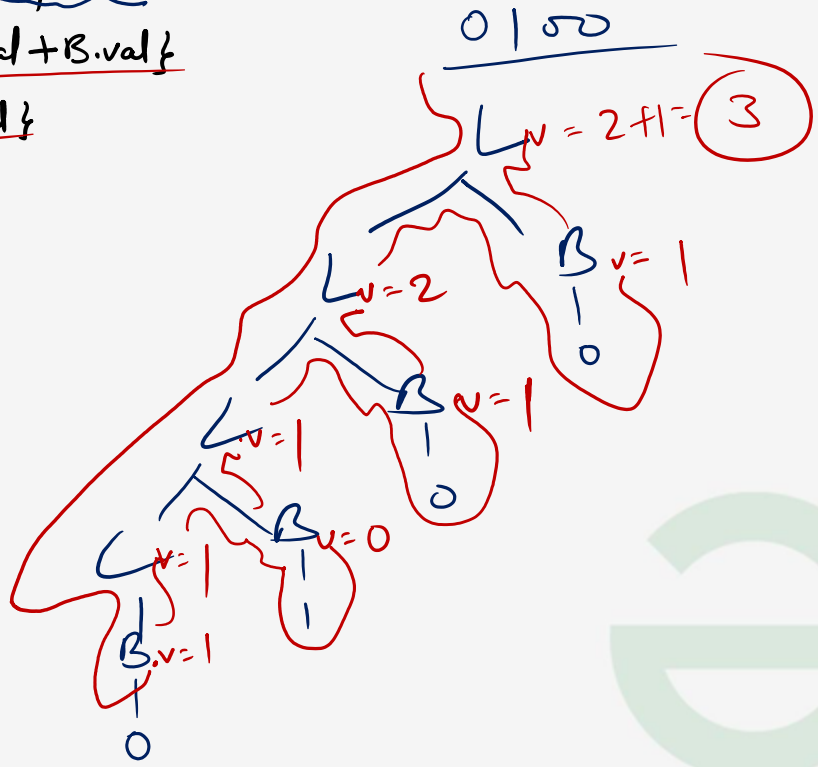
$\quad / B \quad \{ L.val = B.val \}$

$B \rightarrow 0 \quad \{ B.val = 1 \}$

$\quad / 1 \quad \{ B.val = 0 \}$

a)

0100



Compiler Design

Question: SDD for Count the No. of bits

$L \rightarrow L\ B$ { L.val = L_1.val + B.val }

/ B { L.val = B.val }

$B \rightarrow 0$ { B.val = 1 }

/ 1 { B.val = 1 }

0100

Compiler Design

Question: SDD for what?

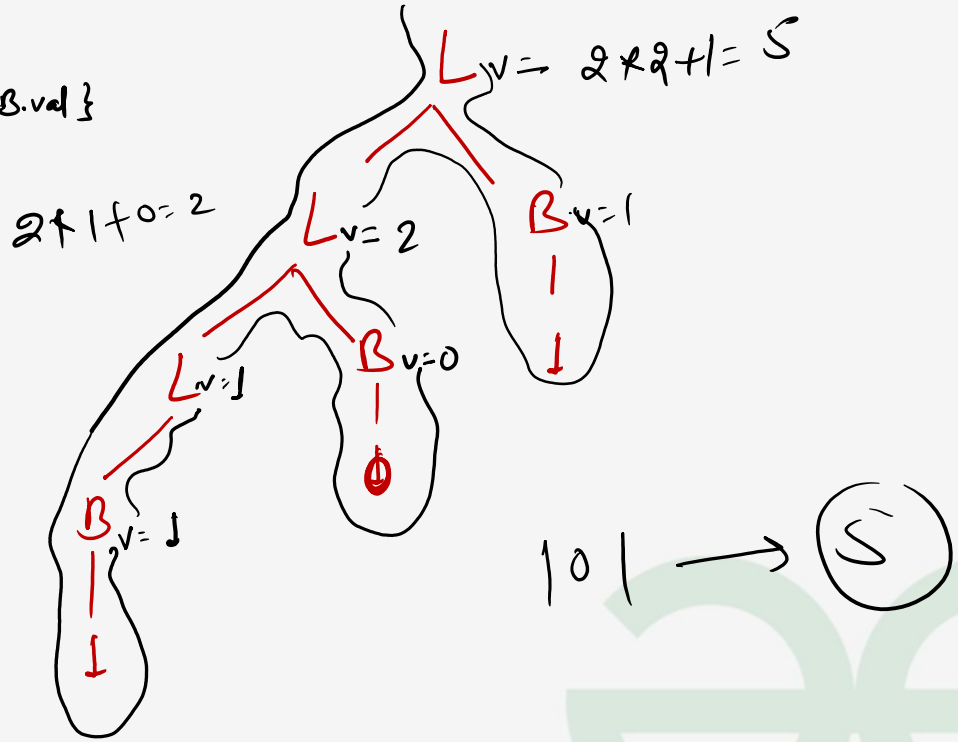
$L \rightarrow LB \quad \{L.val = L_1.val * 2 + B.val\}$

$/B \quad \{L.val = B.val\}$

$B \rightarrow 0 \quad \{B.val = 0\}$

$/1 \quad \{B.val = 1\}$

Binary to Decimal



Compiler Design

2082

SDT

Question: SDD for what?

101.11

101.11 ⇒ 5.75

$N \rightarrow L_1.L_2 \quad \{N.val = L_1.val + \frac{L_2.val}{2^{L_2.count}}\}$

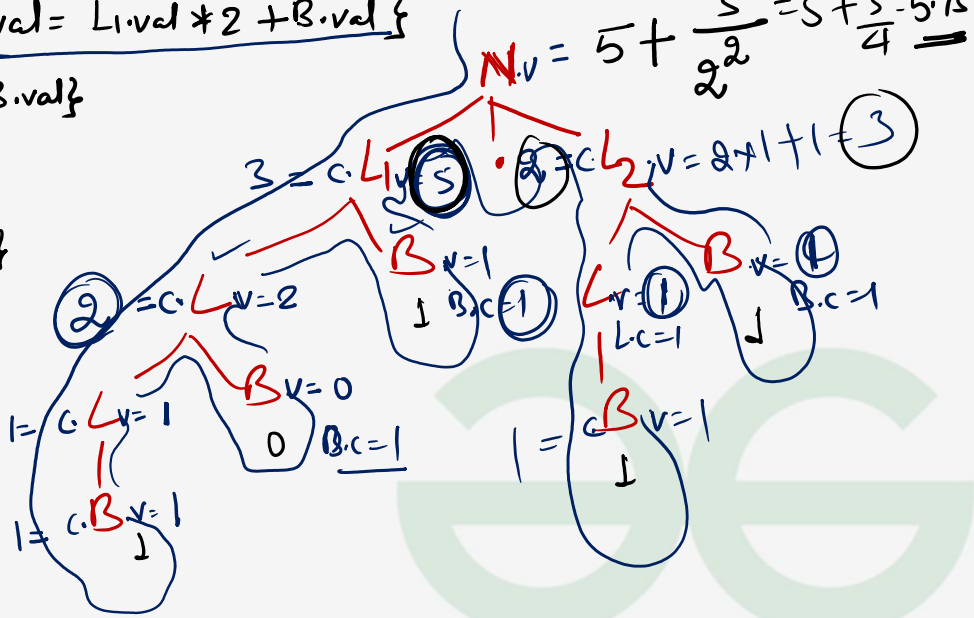
$L \rightarrow LB \quad \{L.c = L_1.c + B.c;\ L.val = L_1.val * 2 + B.val\}$

$/B \quad \{L.c = B.c;\ L.val = B.val\}$

$B \rightarrow 0 \quad \{B.c = 1;\ B.val = 0\}$

$/1 \quad \{B.c = 1;\ B.val = 1\}$

Binary to Decimal with fraction



Compiler Design

SDD to create Syntax Tree :-

2 + 3 * 4

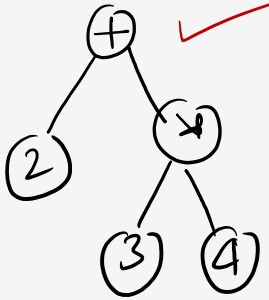
$E \rightarrow E_1 + T$ { E.nptr = mknode (E1.nptr, +, T.nptr) }

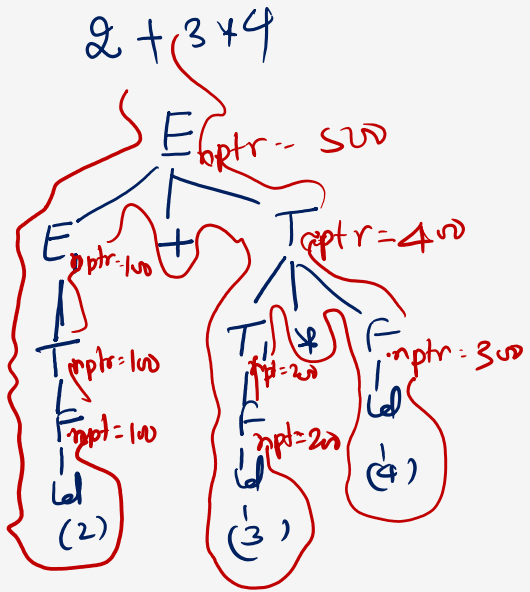
$E \rightarrow T$ { E.nptr = T.nptr }

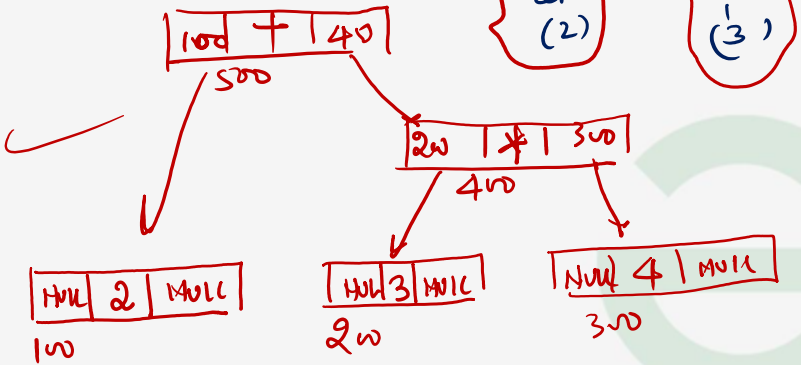
$T \rightarrow T_1 * F$ { T.nptr = mknode (T1.nptr, *, F.nptr) }

$T \rightarrow F$ { T.nptr = F.nptr }

$F \rightarrow id$ { F.nptr = mknode (NULL, id, NULL) }







Compiler Design

SDD to create 3-Address code:- $x = a + b * c$ ($t_2 = a + t_1$)

GATE 2023

$S \rightarrow id = E$ { gen (id.name = E.place); }

$E \rightarrow E_1 + T$ { E.place = newtemp(); gen (E.place = E_1.place + T.place); }

$E \rightarrow T$ { E.place = T.place }

$T \rightarrow T_1 * F$ { T.place = newtemp(); gen (T.place = T_1.place * F.place); }

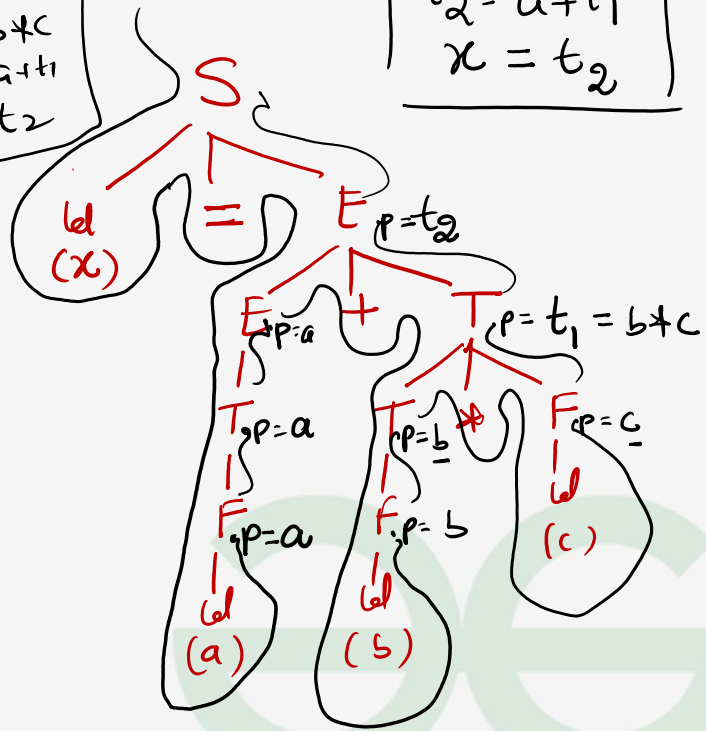
$T \rightarrow F$ { T.place = F.place }

$F \rightarrow id$ { F.place = id.name }

$t_1 = b * c$
$t_2 = a + t_1$
$x = t_2$

3 Address

$t_1 = b * c$
$t_2 = a + t_1$
$x = t_2$



Compiler Design

2 marks

3+ (2*3+4*5)

Question: Consider the following SDT

Break: 5 mints

to 11:20

$E \rightarrow E_1 * T$ { E.val = E_1.val + T.val }

/T { E.val = T.val + 1 }

$T \rightarrow T_1 + F$ { T.val = T_1.val * F.val }

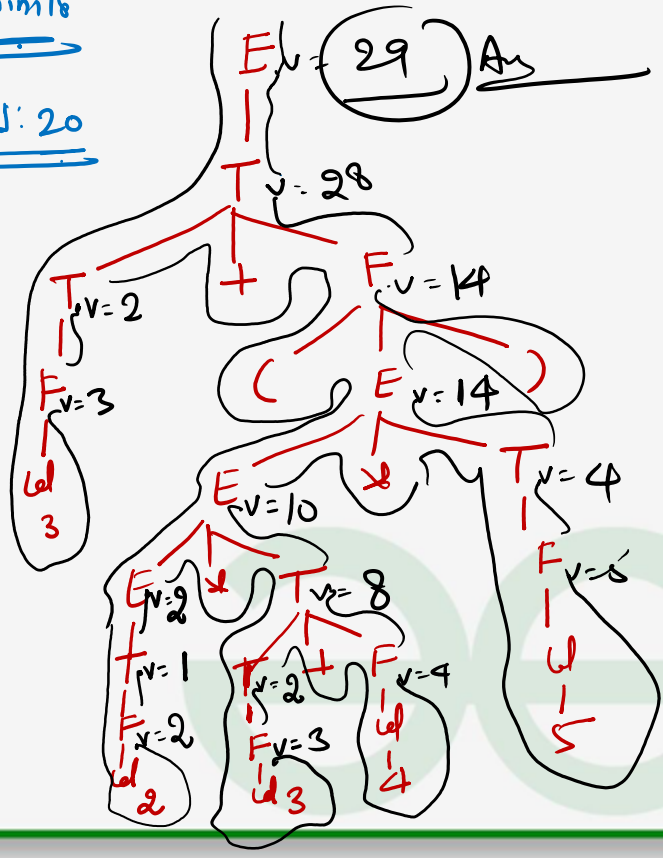
/F { T.val = F.val - 1 }

$F \rightarrow (E)$ { F.val = E.val }

/id { F.val = id.val }

if input = 3+ (2*3+4*5)

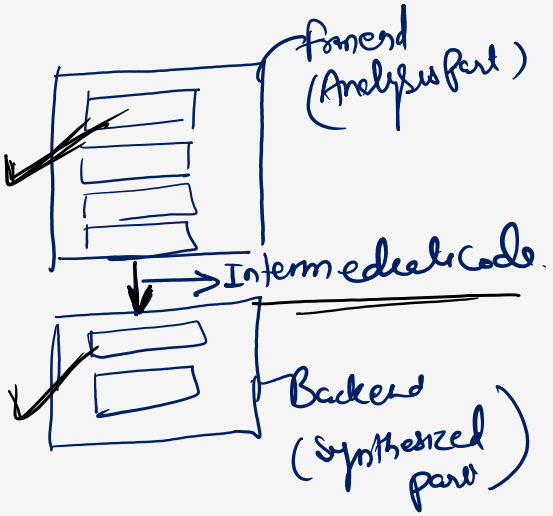
Then o/p = ?

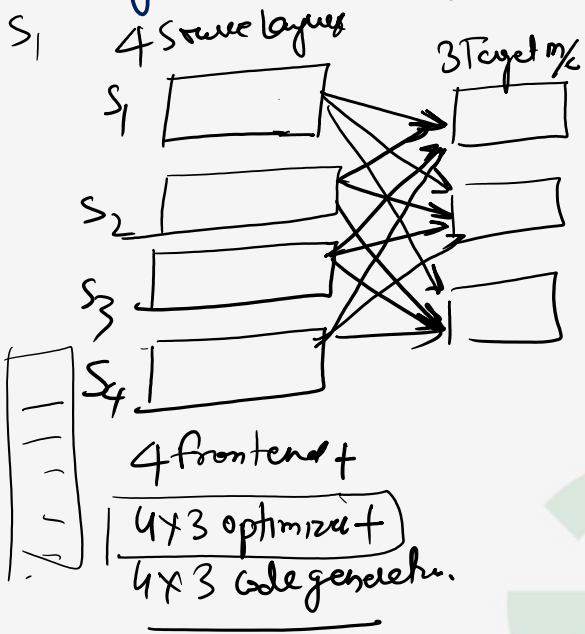


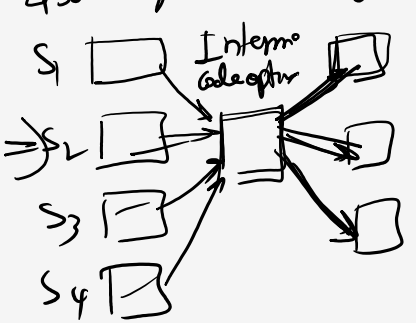
Compiler Design

=> 4 frontend + 1 optimizer + 3 + Code gen.

Intermediate Code generation:- Portability can enhance by intermediate code





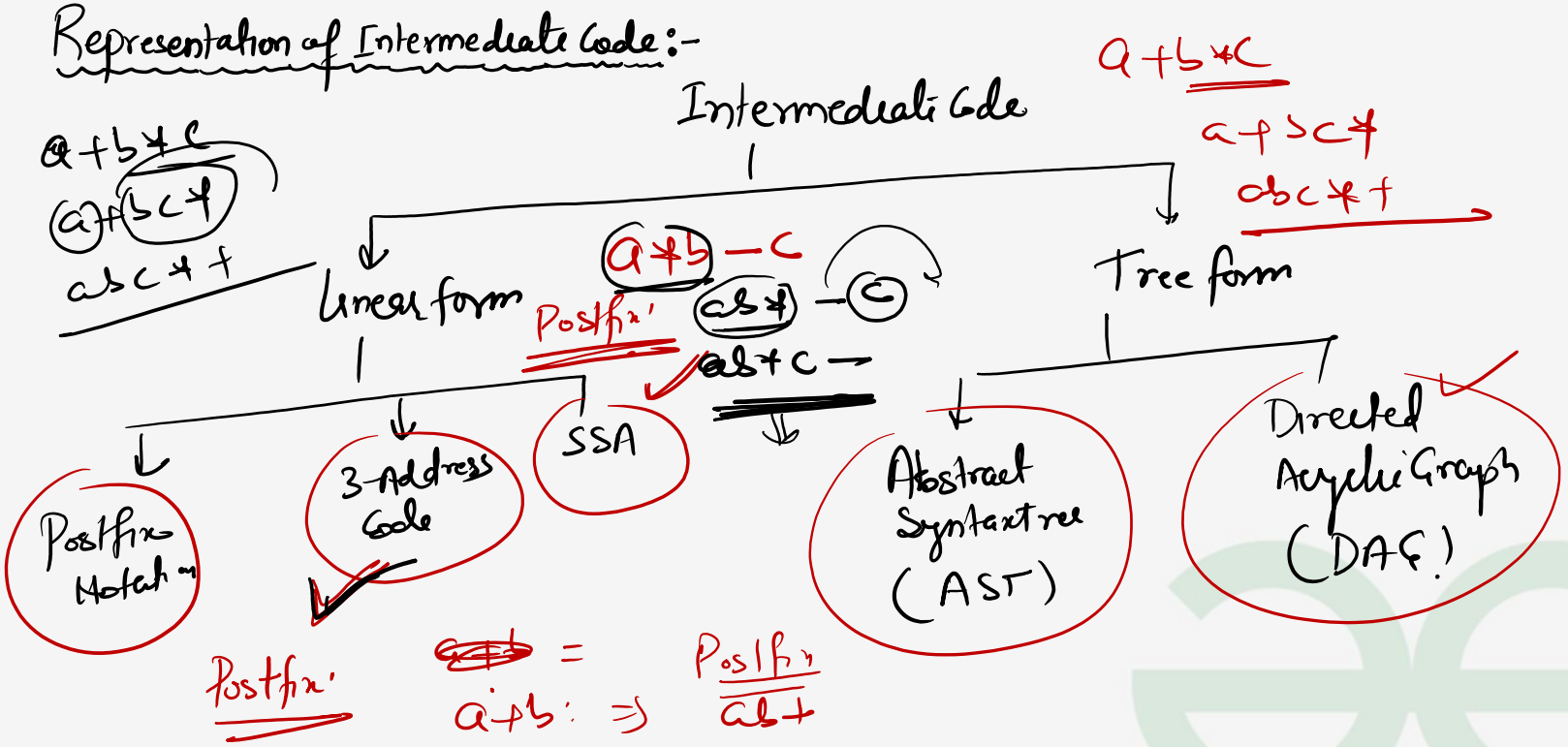


eg

$$\begin{cases} t_1 = a + b \\ c = t_1 \end{cases}$$

Compiler Design

Representation of Intermediate Code :-



Compiler Design

3-Address Code:- Maximum 3 Address or referesces

Maximum operator 2

$a = b + c$

$a = -b$

if $a > b$ Then go to ——

- $a = b \text{ op } c$
- $a = \text{op } b$
- $a = b$

if (Condition) Goto L → Conditional jump

Goto L → Unconditional jump

- $a = b[i]$
- $a[i] = b$

} Array Indexed Assignment statements

- $a = *b$
- $*a = b$

} Pointer Assignment statements

$a = \&b$ Address Assignment

Compiler Design

Param x_1
Param x_2
Param x_3
Param x_n

The Arguments to a procedure/function are defined by a Param instruction

Call P, n — invocation of a Procedure that takes n arguments/parameters

y = Call P, n

$f_1()$
a
s

$f_2()$
c
d

f
Para a
Para b

Para c
Para d

Compiler Design

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

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

t_3 $\quad$ t_4

$t_1 = a + b$

$t_2 = c * d$

$t_3 = t_1 - t_2$

$t_4 = t_1 - c$

$t_5 = t_3 * t_4$

$x = t_5$

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) if (a > b)
    a = a + 1
else
    b = b - 1

3 Address Code:

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

if (a > b) goto L1
b = b - 1
goto L2
L1: a = a + 1
L2: ——

4) for (i = 0; i ≤ n; i++)
{
    x = x + i;
    y = y - i;
}

3 Address Code:

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

6)
```
for(i=1; i≤n; i++)
{ x = a+b*c;
}
```

Compiler Design

7) Switch (i)

{

Case 1:

$x_1 = a_1 + b_1 * c_1;$

break;

Case 2:

$x_2 = a_2 + b_2 * c_2;$

break;

default:

$x_3 = a_3 + b_3 * c_3;$

break;

}

3 Add:

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) ——

7) $t_1 = b_1 * c_1$

8) $t_2 = a_1 + t_1$

9) $x_1 = t_2$

10) goto 6

11) $t_1 = b_2 * c_2$

12) $t_2 = a_2 + t_1$

13) $x_2 = t_2$

14) goto 6

Thank You !