# ASSIGNMENT-5 COMPILERS

(Itish Agarwal, 18CS30021)

13 Oct' 2020

Q1.

(a) Arithmetic operators:

→ Represented as a sequence of statements.

→ Temporary variables are used for intermediate values.

→ Attributes for Expression:

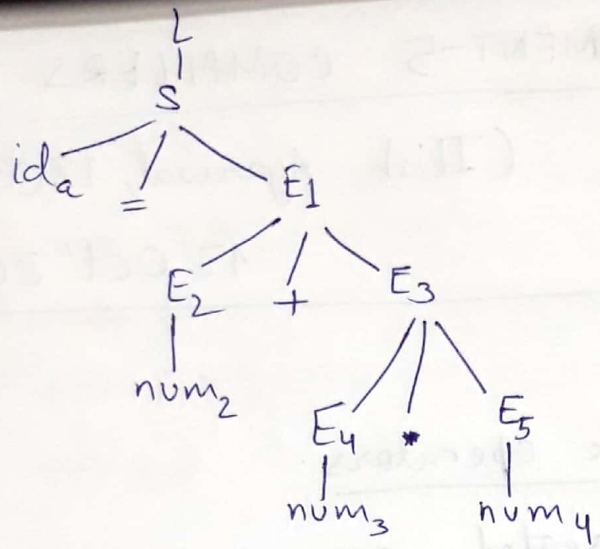
E.loc()
id.loc()
num.val

→ Auxiliary methods

gentemp()
emit(result, arg1, op, arg2)

Ex: a = 4 + 5 * 7;

| Reductions | TAC |
|---|---|
| $E \rightarrow num$ | t00 = 4 |
| $E \rightarrow num$ | t01 = 5 |
| $E \rightarrow num$ | t02 = 7 |
| $E \rightarrow E_1 * E_2$ | t03 = t01 * t02 |
| $E \rightarrow E_1 + E_2$ | t04 = t00 + t03 |
| $S \rightarrow id = E$ | a = t04 |




(b) Boolean operators:

→ Two types of translation: by value, by Control flow

→ We mainly use control flow to translate.

→ Attributes used:

B.truelist
B.falselist
B.loc
nextinstr
M.instr

→ Auxiliary methods

makelist (i)
merge ($p_1$, $p_2$)
backpatch (p, i)

Ex: a > b || c == d

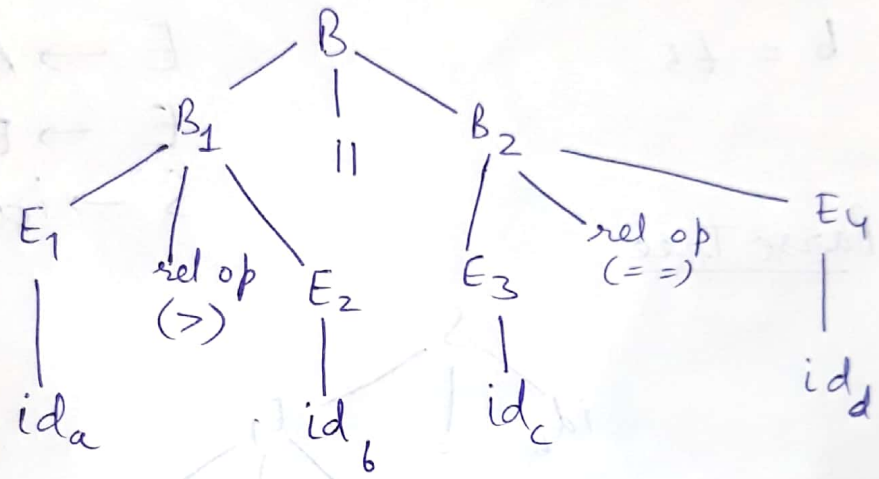
PTO

```
100: if a>b goto 104
101: goto 102
102: if c==d, goto 104
103: goto 105
104: goto 000 (true)
105: goto 000 (false)
```

$B \rightarrow B_1 \,||\, B_2$

$B_1 \rightarrow E_1$ rel op $E_2$

$B_2 \rightarrow E_3$ rel op $E_4$




© Array References:

→ We have, a[i] begins at location (base + i * size of (element)).

→ Attributes used:

A.loc
A.array
A.type

→ Grammar uses both left and right recursion.

Eg: int a[2][3], b, c;

b = c + a[i][j]

## Three - address code

```
t1 = i * 12
t2 = j * 4
t3 = t1 + t2
t4 = a[t3]
t5 = c + t4
b = t5
```

## Reductions

$E \rightarrow id$

$E \rightarrow id$

$A \rightarrow id[E]$

$E \rightarrow id$

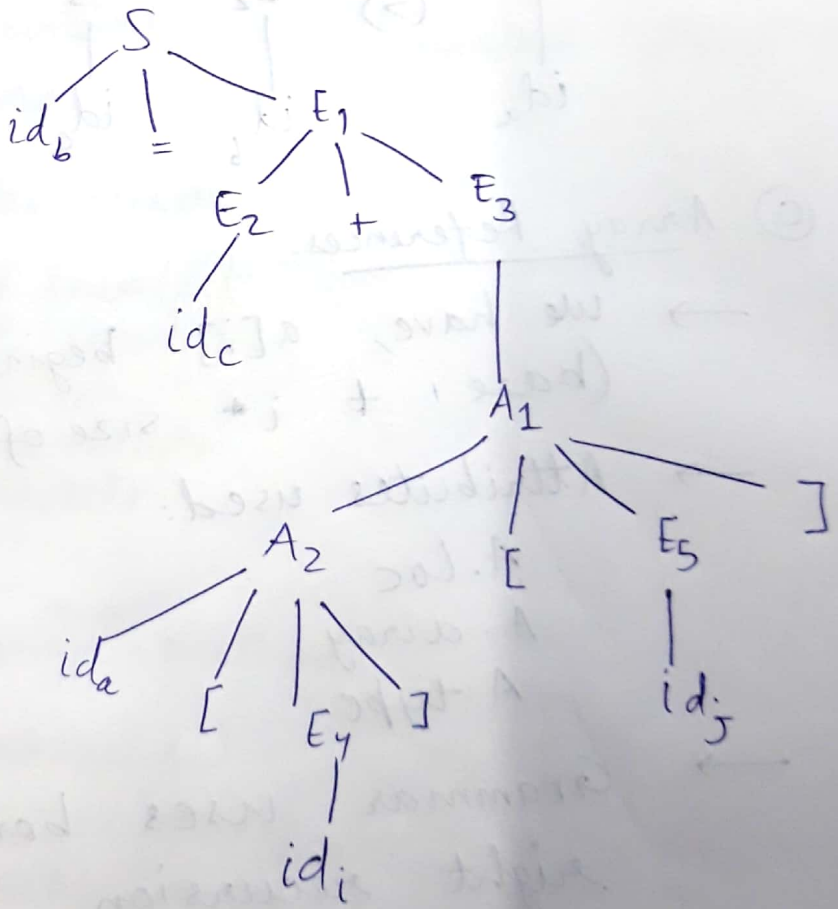
$A \rightarrow A_1[E]$

$E \rightarrow A$

$E \rightarrow E + E$

$S \rightarrow id = E$

## Parse Tree




Q2. PTO

Q2.

| TAC | Reduction |
|---|---|
| t00 = 5 | E → num |
| a = t00 | S → id = E |
| t01 = 6 | E → num |
| b = t01 | S → id = E |
| t02 = 10 | E → num |
| t03 = t02/a | E → E/E |
| | E → (E) |
| t04 = b*t03 | E → E*E |
| | E → (E) |
| t05 = a+t04 | E → E+E |
| c = t05 | S → id = E |
| t06 = -c | E → -E |
| t07 = a*b | E → E*E |
| t08 = t06+t07 | E → E+E |
| d = t08 | S → id = E |

PTO

Parse Tree: