# Assignment-1

**Soln=1** On adding one extra rule, grammer is

$S' \to S$

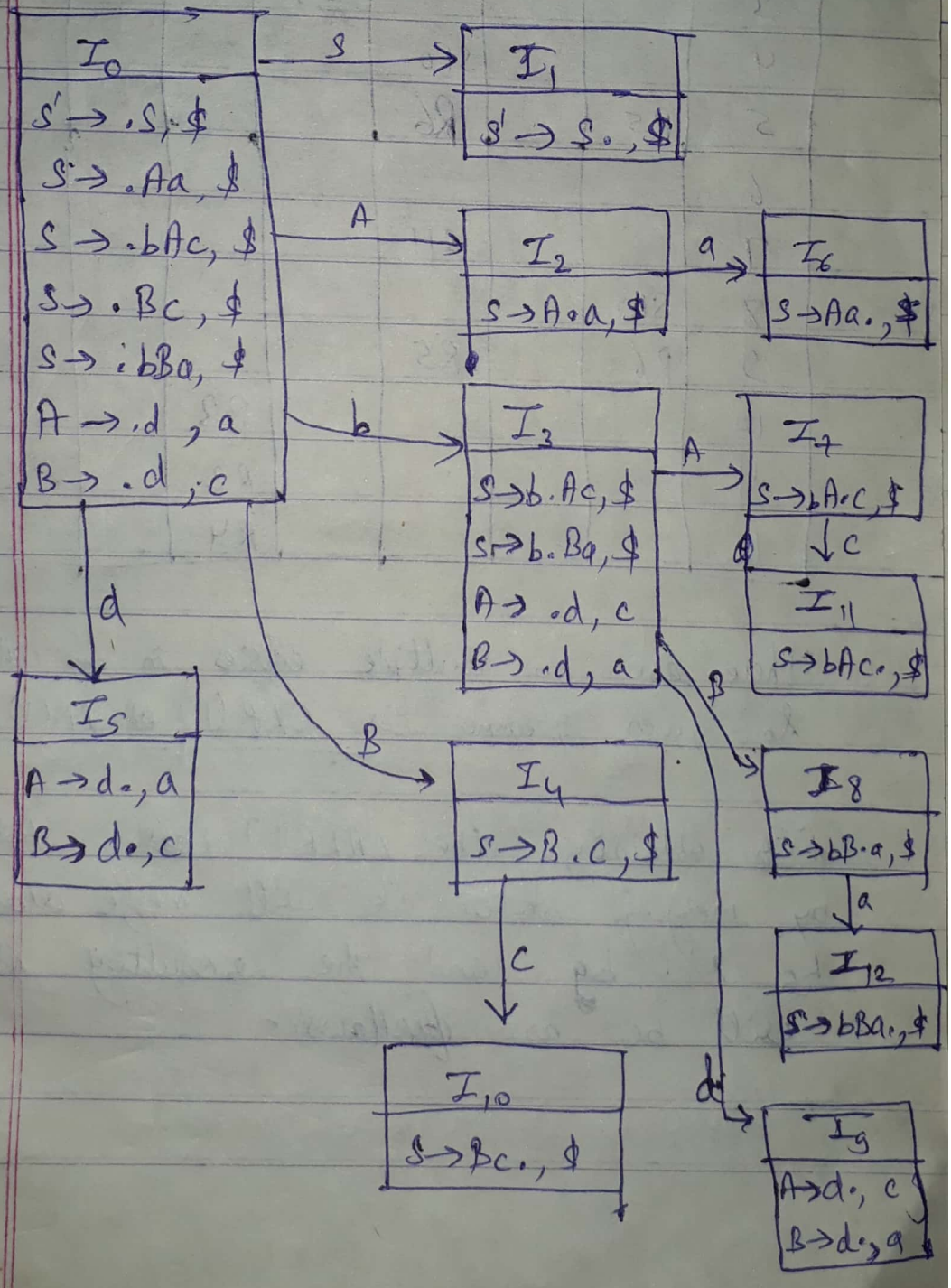
$S \to Aa / bAc / Bc / bBa$

$A \to d$

$B \to d$

For CLR(1) Parser,

State Diagram as

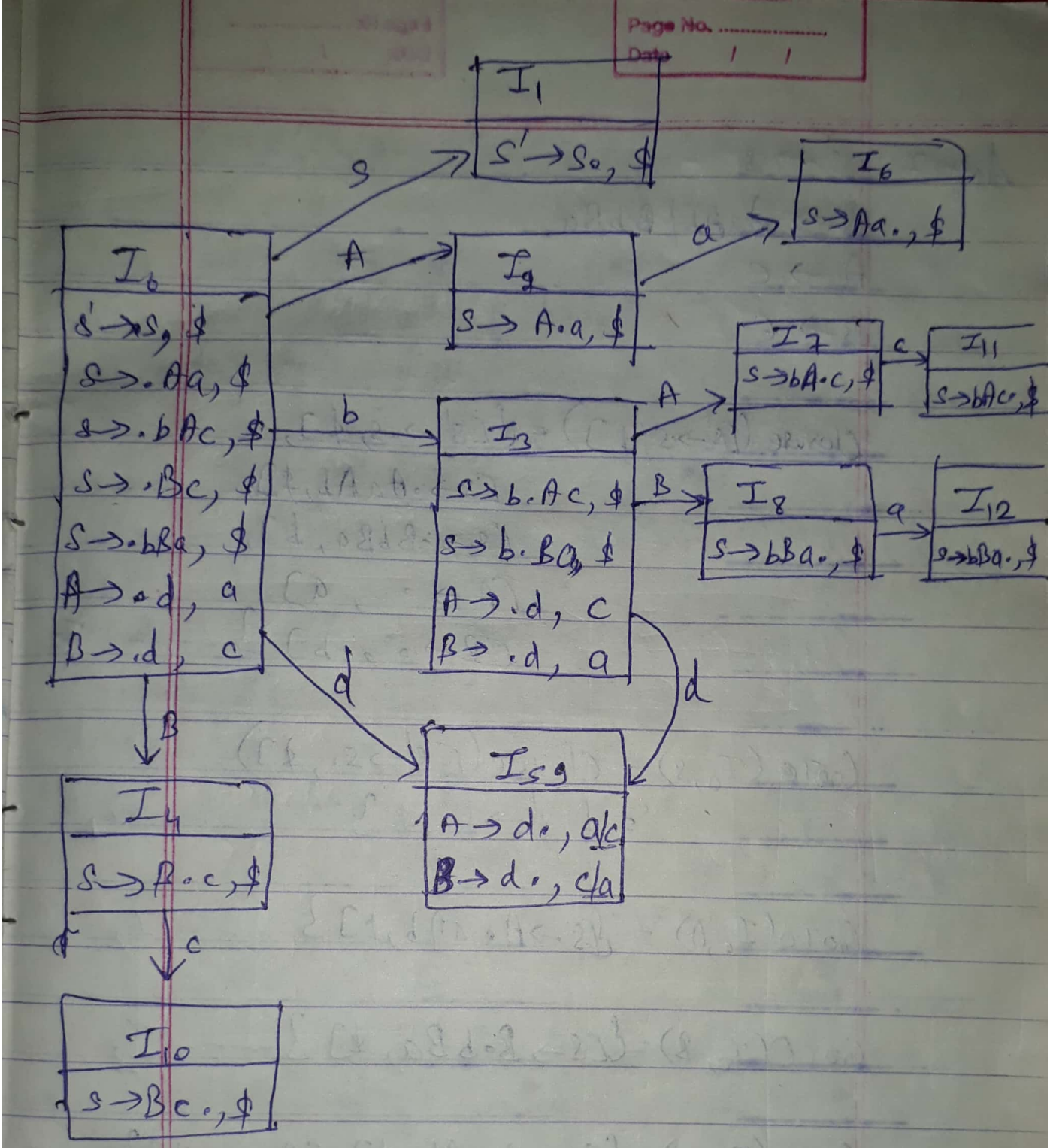


Now, CLR(1) Passing Table is

| State | Action | | | | | GOTO | | |
|---|---|---|---|---|---|---|---|---|
| | a | b | c | d | $ | S | A | B |
| 0 | | s3 | | s4 | | 1 | 2 | 5 |
| 1 | | | | | accept | | | |
| 2 | s6 | | | | | | | |
| 3 | | | | s9 | | | 7 | 8 |
| 4 | | | s10 | | | | | |
| 5 | R5 | | R6 | | | | | |
| 6 | | | | | | | | |
| 7 | | | s11 | | | | | |
| 8 | s10 | | | | | | | |
| 9 | R6 | | R5 | | | | | |
| 10 | | | | | R3 | | | |
| 11 | | | | | R2 | | | |
| 12 | | | | | R4 | | | |

These are no multiple entries in any cell, So, given grammer is CLR(1) or LR(1)

For obtaining the LALR(1) Passing table by merging states, we will merge states $I_5$ & $I_9$ and the resulting state will be as follows:-




Here, In $I_{5,9}$, there is reduce-reduce conflict.

So, given grammer is not LALR(1)

Page No. _____
Date / /

Ans 2 $S' \to S$

$S \to AaAb \mid BbBa$

$A \to \epsilon$

$B \to \epsilon$

Closure $([S' \to .S, \$]) = \{[S' \to .S, \$],$
$[S \to .AaAb, \$],$
$[S \to .BbBa, \$],$
$[A \to ., a]$
$[B \to ., b]\}$ -- $I_0$

GoTo $(I_0, S) =$ Closure $([S' \to S., \$])$
$= \{S' \to S., \$\}$ -- $I_1$

GoTo $(I_0, A) = \{[S \to A.aAb, \$]\}$ -- $I_2$

GoTo $(I_0, B) = \{[S \to B.bBa, \$]\}$ -- $I_3$

GoTo $(I_2, a) = \{[S \to Aa.Ab, \$], [A \to ., b]\}$ -- $I_4$

GoTo $(I_3, b) = \{[S \to Bb.Ba, \$], [B \to ., a]\}$ -- $I_5$

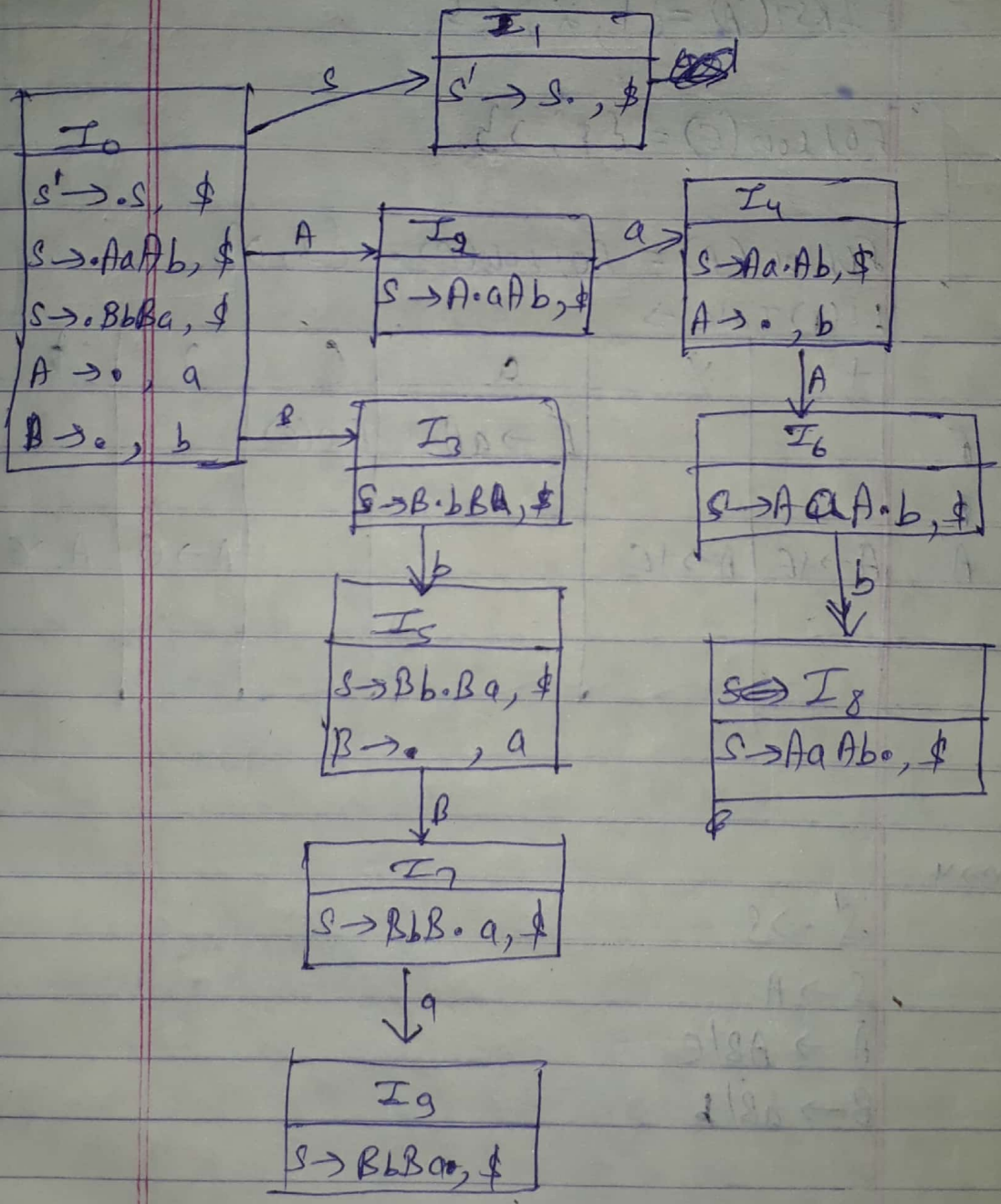
GoTo $(I_4, A) = \{[S \to AaA.b, \$]\}$ -- $I_6$

GoTo $(I_5, B) = \{[S \to BbB.a, \$]\}$ -- $I_7$

GoTo $(I_6, b) = \{[S \to AaAb., \$]\}$ -- $I_8$

GOTO($I_7$, a) = {[S → BbBa•, $]} ---- $I_9$

Since, there are no states having same rule but different lookhead.
So, state diagram as follow.




Since, there are no merge state ⇒ No conflict
⇒ Given grammar is LALR(1)

SUMAN

Page No. .....
Date / /

Ans 3. $E \to aA \mid (E)$

$A \to +E \mid *E \mid \epsilon$

$FIRST(E) = \{a, (\}$

$FIRST(A) = \{+, *, \epsilon\}$

$FOLLOW(E) = \{\$, )\}$.

$FOLLOW(A) = FOLLOW(E) = \{\$, )\}$.

LL(1) Table →

| | + | * | a | ( | ) | $ |
|---|---|---|---|---|---|---|
| E | | | $E \to aA$ | $E \to (E)$ | | |
| A | $A \to +E$ | $A \to +E$ | | | $A \to \epsilon$ | $A \to \epsilon$ |

Ans 4.

$S' \to S$

$S \to A$

$A \to AB \mid \epsilon$

$B \to aB \mid b$

Non-empty set of LR(1) items.

**$I_0$**
- $S' \to .S, \$$
- $S \to .A, \$$
- $A \to .AB, \$$
- $A \to ., \$$
- $A \to .AB, a/b$
- $A \to ., a/b$

**$I_1$**
- $S' \to S., \$$

**$I_2$**
- $A \to A.B, \$$
- $S \to A., \$$
- $A \to A.B, a/b$

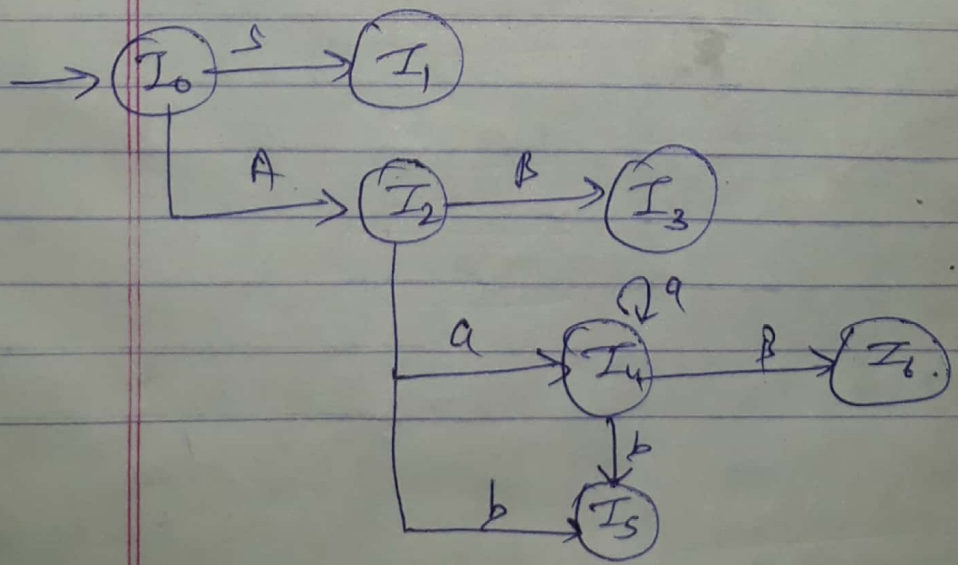
**$I_2$**
- $S \to A. \$$
- $A \to A.B, \$/a/b$
- $B \to .aB, \$/a/b$
- $B \to .b, \$/a/b$

**$I_3$**
- $A \to AB., \$/a/b$

**$I_4$**
- $B \to a.B, \$/a/b$
- $B \to .aB, \$/a/b$
- $B \to .b, \$/a/b$

**$I_5$**
- $B \to b., \$/a/b$

**$I_6$**
- $B \to aB., \$/a/b$




SUMAN

Page No. _____
Date / /

5) Set of operations of the arithmatic expression is equal to no. of reductions.
Let example grammer be

~~E~~

$E \to E_1 + T$
$E \to T$
$T \to T_1 * F$
$T \to f$
$f \to digit$

for expressions 3+5*6.

Now, SDT form is.

| | | |
|---|---|---|
| $E \to E_1 + T$ | $\{E.val = E_1.val + T.val;\}$ | 2 |
| $E \to T$ | $\{E.val = T.val;\}$ | 1 |
| $T \to T * F$ | $\{T.val = T_1.val * f.val;\}$ | 2 |
| $T \to f$ | $\{T.val = f.val;\}$ | 1 |
| $f \to digit$ | $\{f.val = digit.lexem;\}$ | 1 |
| | | 7 |

Ans 7.

| Production | Semantic Rule |
|---|---|
| $N \to L.R$ | $N.val = L.val + R.val$ |
| $L \to BL_1$ | $L.val = (B.val + L_1.val) \times 2$ |
| $L \to B$ | $L.val = B.val \times 2$ |
| $R \to BR_1$ | $R.val = (B.val + R_1.val) \times 2^{-1}$ |
| $R \to B$ | $R.val = B.val \times 2^{-1}$ |
| $B \to 0$ | $B.val = 0$ |
| $B \to 1$ | $B.val = 1$ |

Page No. .....
Date / /

**Ans 6** Grammer is

$S \rightarrow id = E; \mid L = E;$

$E \rightarrow E_1 + E_2 \mid id \mid L$

$L \rightarrow id[E] \mid L_1[E]$

SDT to convert array references to 3-address code:

$S \rightarrow id = E;$ { gen(top.get(id.lexeme) '=' E.addr); }

$\mid L = E;$ { gen(L.addr.base '[' L.addr ']' '=' E.addr); }

$E \rightarrow E_1 + E_2$ { E.addr = new Temp();
gen(E.addr '=' $E_1$.addr '+' $E_2$.addr); }

$\mid id$ { E.addr = top.get(id.lexeme); }

$\mid L$ { E.addr = new Temp();
gen(E.addr '=' L.array.base '[' L.addr ']'); }

$L \rightarrow id[E]$ { L.array = top.get(id.lexeme);
L.type = L.array.type.elem;
L.addr = new Temp();
gen(L.addr '=' E.addr '*' L.type.width); }

$\mid L_1[E]$ { L.array = $L_1$.array;
L.type = $L_1$.type.elem;
t = new Temp();
L.addr = new Temp();
gen(t '=' E.addr '*' L.type.width);
gen(L.addr '=' $L_1$.addr '+' t); }