Intermediate Code Generator Using Flex and Bison

Three Address Code

Where RHS expression has at most one operator.

Sample Input:

2 + 3 * 5

Sample Output:

??

Three Address Code (Cont…)

Where RHS expression has at most one operator.

Sample Input:

2 + 3 * 5

Sample Output:

??

Three Address Code (Cont…)

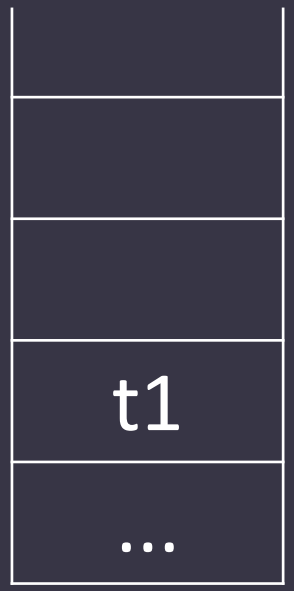
Where RHS expression has at most one operator.

Sample Input:

2 + 3 * 5

Sample Output:

t1 = 3 * 5



Three Address Code (Cont…)

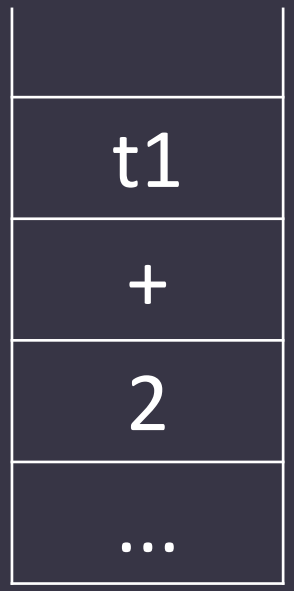
Where RHS expression has at most one operator.

Sample Input:

2 + 3 * 5

Sample Output:

t1 = 3 * 5



Three Address Code (Cont…)

Where RHS expression has at most one operator.

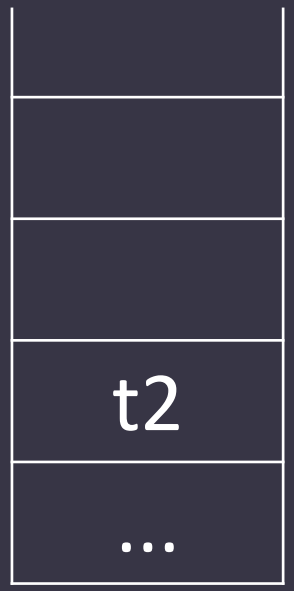
Sample Input:

2 + 3 * 5

Sample Output:

t1 = 3 * 5

t2 = 2 + t1



Three Address Code (Cont…)

Where RHS expression has at most one operator.

Sample Input:

```
x = 2 / a * 5
```

Sample Output:

??

Three Address Code (Cont…)

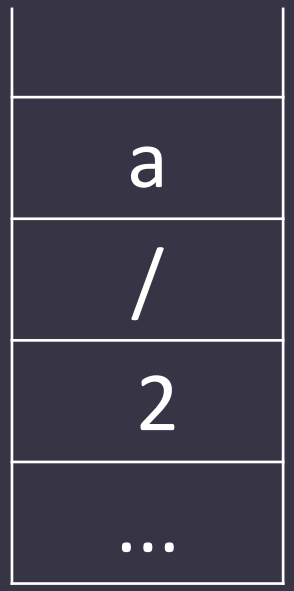
Where RHS expression has at most one operator.

Sample Input:

x = 2 / a * 5

Sample Output:

??



Three Address Code (Cont…)

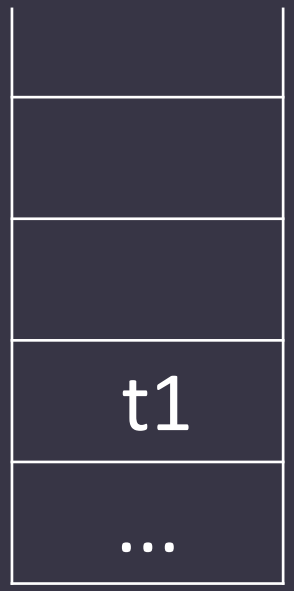
Where RHS expression has at most one operator.

Sample Input:

x = 2 / a * 5

Sample Output:

t1 = 2 / a



Three Address Code (Cont…)

Where RHS expression has at most one operator.

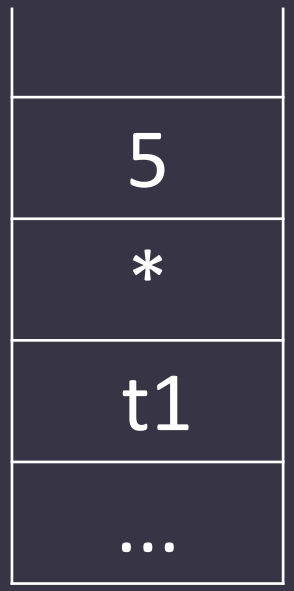
Sample Input:

x = 2 / a * 5

Sample Output:

t1 = 2 / a

t2 = t1 * 5



Three Address Code (Cont…)

Where RHS expression has at most one operator.

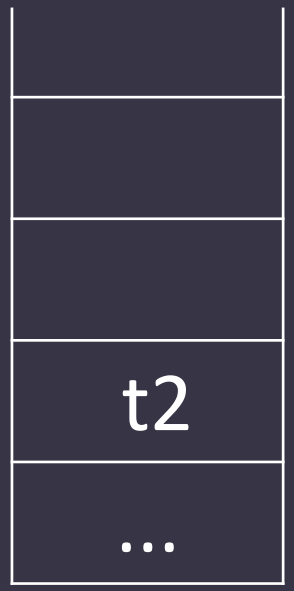
Sample Input:

x = 2 / a * 5

Sample Output:

t1 = 2 / a

t2 = t1 * 5



Three Address Code (Cont…)

Where RHS expression has at most one operator.

Sample Input:

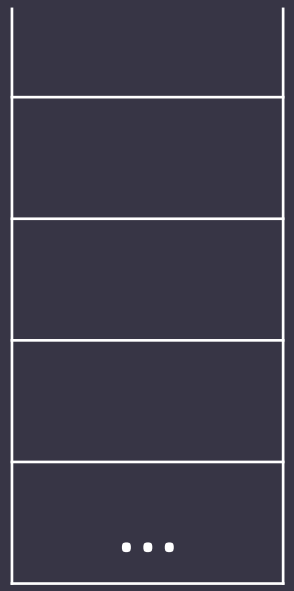
x = 2 / a * 5

Sample Output:

t1 = 2 / a

t2 = t1 * 5

x = t2



Three Address Code (Cont…)

How do we handle multiple data types in a stack?

In YACC file,

```
%union {
        double dval;
        char cvar[5];
      }
```

Three Address Code (Cont…)

How do we handle multiple data types in a stack?

In YACC file,

```
%union {
        double dval;
        char cvar[5];
      }
```

Write this in .l file.

```
strcpy (yylval.cvar, yytext);
```

Three Address Code (Cont…)

How do we handle multiple data types in a stack?

Changing the type of YYSTYPE as the **symblolinfo** class

SDD for

Three Address Code Generation

PRODUCTION	SEMANTIC RULES
$S \rightarrow \textbf{id} = E\ ;$	$S.code = E.code \mid\mid gen(top.get(\textbf{id}.lexeme)\ '='\ E.addr)$
$E \rightarrow E_1 + E_2$	$E.addr = \textbf{new}\ Temp()$ $E.code = E_1.code \mid\mid E_2.code \mid\mid gen(E.addr\ '='\ E_1.addr\ '+'\ E_2.addr)$
$\mid\ - E_1$	$E.addr = \textbf{new}\ Temp()$ $E.code = E_1.code \mid\mid gen(E.addr\ '='\ '\textbf{minus}'\ E_1.addr)$
$\mid\ (\ E_1\ )$	$E.addr = E_1.addr$ $E.code = E_1.code$
$\mid\ \textbf{id}$	$E.addr = top.get(\textbf{id}.lexeme)$ $E.code = ''$

Lab Practice

Design a three address code generator for the following expression.

Sample Input:

x = 5 + 6 + 3

Sample Output:

t1 = 5+6

t2 = t1 + 3

x = t2

Thank You ☺