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…)

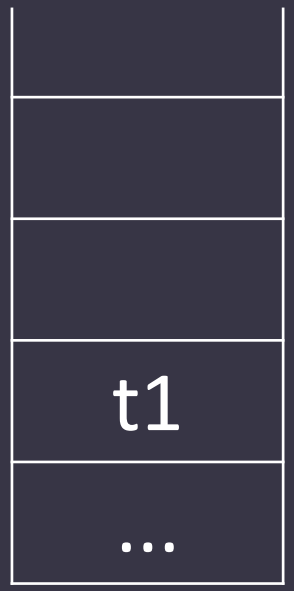
Where RHS expression has at most one operator.

Sample Input:

2 + 3 * 5

Sample Output:

t1 = 3 * 5



Three Address Code (Cont…)

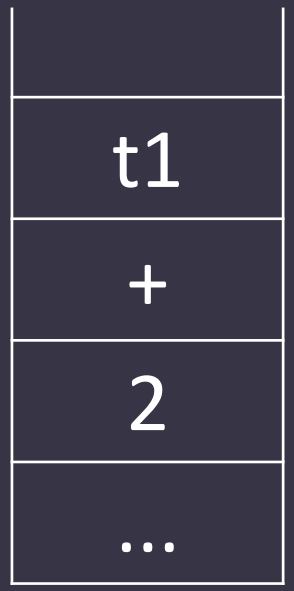
Where RHS expression has at most one operator.

Sample Input:

2 + 3 * 5

Sample Output:

t1 = 3 * 5



Three Address Code (Cont…)

Where RHS expression has at most one operator.

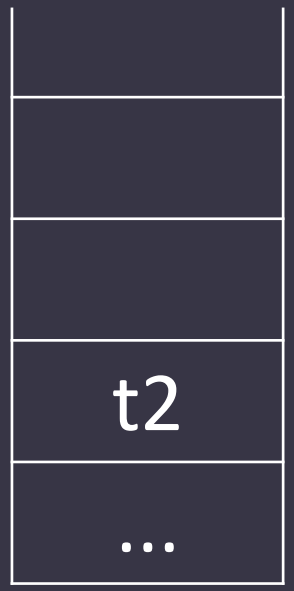
Sample Input:

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…)

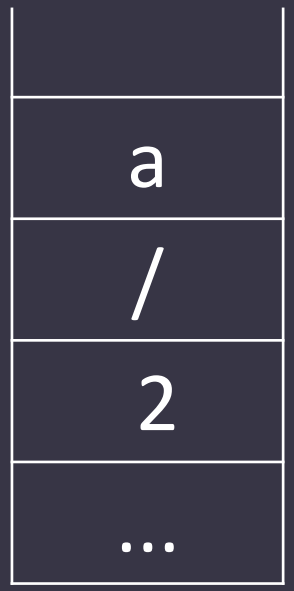
Where RHS expression has at most one operator.

Sample Input:

x = 2 / a * 5

Sample Output:

??



Three Address Code (Cont…)

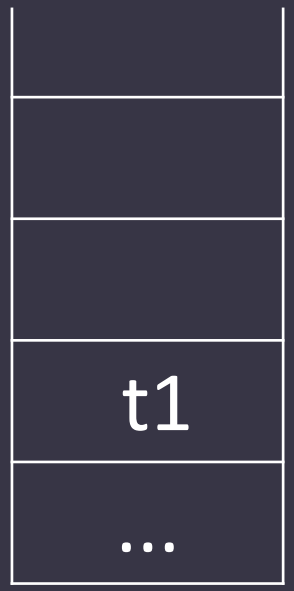
Where RHS expression has at most one operator.

Sample Input:

x = 2 / a * 5

Sample Output:

t1 = 2 / a



Three Address Code (Cont…)

Where RHS expression has at most one operator.

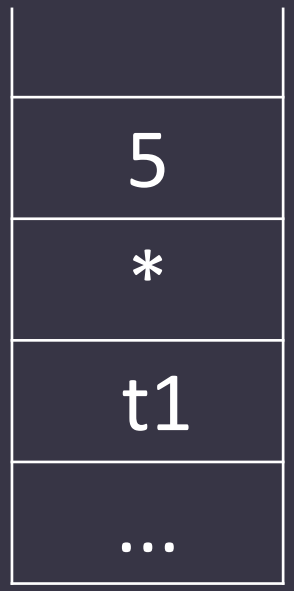
Sample Input:

x = 2 / a * 5

Sample Output:

t1 = 2 / a

t2 = t1 * 5



Three Address Code (Cont…)

Where RHS expression has at most one operator.

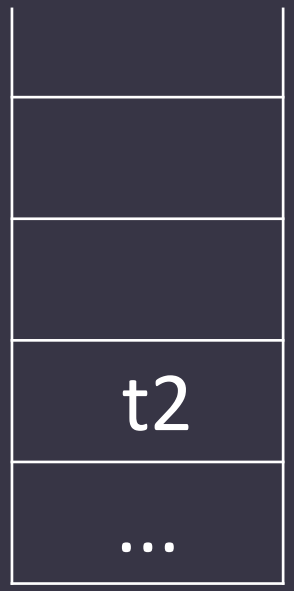
Sample Input:

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:

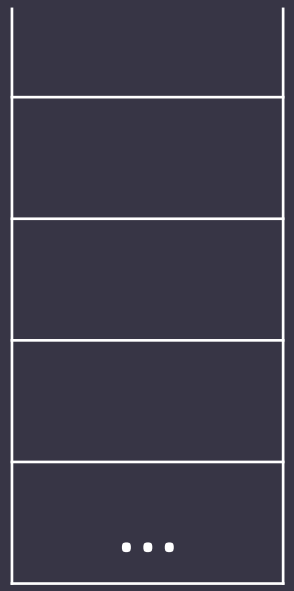
x = 2 / a * 5

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);
```

Practice

Design a three address code generator that will support the standard operations mentioned in assignment 4.

Sample Input:

x = 5 + 6 * 3

Sample Output:

t1 = 6 * 3

t2 = 5 + t1

x = t2

Thank You