The syntax of a three-address intermediate language is suggested below. Here, \( n \) denotes a (decimal) integer constant, and \( label \) denotes a label.

**Data Declarations**

\( \text{global id type:} \)

Declares \( id \) to be a global variable of type \( type \), where \( type \) is \( \text{int} \), \( \text{char} \), \( \text{double} \), or an array:

- \( \text{array}[n] \) of \( \text{int} \)
- \( \text{array}[n] \) of \( \text{char} \)
- \( \text{array}[n] \) of \( \text{double} \)

**Assignment Statements**

\( x := y \ op \ z: \)

Where \( op \) is one of \(+\), \(-\), \(*\), \(/\) The \( y \) and \( z \) are constants or variables.

\( x := -y: \)

\( y \) is a constant or variable.

\( x := y: \)

\( y \) is a constant or variable.

**Jump Instructions**

\( \text{goto label:} \)

Jump to a label

\( \text{if } x \ relop \ val \ \text{goto label:} \)

\( val \) is an identifier or a decimal integer, \( relop \) is one of \( \text{<=} \), \( \text{<} \), \( \text{>=} \), \( \text{>} \), \( \text{==} \), \( \text{!=} \)

\( \text{label L:} \)

The \( L \) is a label (we assume the convention that is an identifier beginning with the letter \( L \)) and must be unique in the program.

**Indexed Assignments**

\( x := y[i]: \)

\( y \) is an array

\( x[i] := y: \)

\( x \) is an array

**Address and Pointer Assignments**

\( x := &y: \)

Take the address of variable \( y \)

\( x := *y: \)

Dereference the variable \( y \)

\( *x := y: \)

Dereference the variable \( x \) and assign from variable \( y \)
Procedural Instructions

**enter n:**

The $n$ is the number of bytes occupied by the locals and temporaries of the procedure being entered. This is used to update the stack and frame pointers on entry into a procedure.

**param x:**

The $x$ is an actual parameter

**call $p$, n:**

The $p$ is a procedure, $n$ the number of arguments

**return:**

Return to the caller

**return $x$:**

Return to the caller, with return value $x$

**retrieve $x$:**

Copy the return value into $x$