TRANSLATION SCHEME FOR ASSIGNMENT STATEMENT

The translation scheme is used to show how symbol table entries can be found. Terms used in the scheme is

lookup (id.name): checks for the entry name in the symbol table. If so it returns the pointer to that entry else it returns nil.

emit: used to emit three address statements to an output file

SNo	Productions	Semantic action

1	S->id=E	p = lookup(id.name)
		if (p!=nil)then
		emit (p=E.place)
		else error

2	E->E1+E2	E.place=newtemp
		emit(E.place=E1.place+E2.place)

3	E->E1*E2	E.place=newtemp
		emit(E.place=E1.place*E2.place)

4	E->-E	E.place=newtemp
		emit(E.place=E1.place=uminusE2.place)

5	E->(E)	E.place=E1.place

6	E->id	p = lookup(id.name)
		if (p!=nil)then
		emit (p=E.place)
		else error

Fig. 1 Translation scheme for the assignment statement

The above translation scheme is similar to the translation of three address code for the example a=b*-c+b*-c. During decalaration statement, variable names are entered into the symbol table. While executing the assignment statement, variable names are searched in the symbol table.

Type Conversions within Assignments

There can be different types of variables or constants. So the compiler must either

1. reject mixed-type operations
2. or generate appropriate coercion (type conversion) instruction

SNo	Productions	Semantic action

1	E->E+E	E.place=newtemp;
		if E1.type=integer and E2.type=integer then
		begin
		emit (E.place=E1.place+E2.place)
		E.Type=integer
		end
		else if E1.type=real and E2.type=real then
		begin
		emit (E.place=E1.place+E2.place)
		E.Type=integer
		end
		else if E1.type=integer and E2.type=real then
		begin
		u=newtemp
		emit (u=inttorealE2.place)
		emit (E.place=E1.place+E2.place)
		E.Type=real
		end
		else if E1.type=real and E2.type=integer then
		begin
		u=newtemp
		emit (u=inttoreal E1.place)
		emit (E.place=E1.place+E2.place)
		E.Type=real
		End

2	E->E1*E2	E.place=newtemp;
		if E1.type=integer and E2.type=integer then
		begin
		emit (E.place=E1.place*E2.place)
		E.Type=integer
		end
		else if E1.type=real and E2.type=real then
		begin
		emit (E.place=E1.place*E2.place)
		E.Type=integer
		end
		else if E1.type=integer and E2.type=real then
		begin
		u=newtemp
		emit (u=inttorealE2.place)
		emit (E.place=E1.place*E2.place)
		E.Type=real
		end
		else if E1.type=real and E2.type=integer then
		begin
		u=newtemp
		emit (u=inttoreal E1.place)
		emit (E.place=E1.place*E2.place)
		E.Type=real
		End

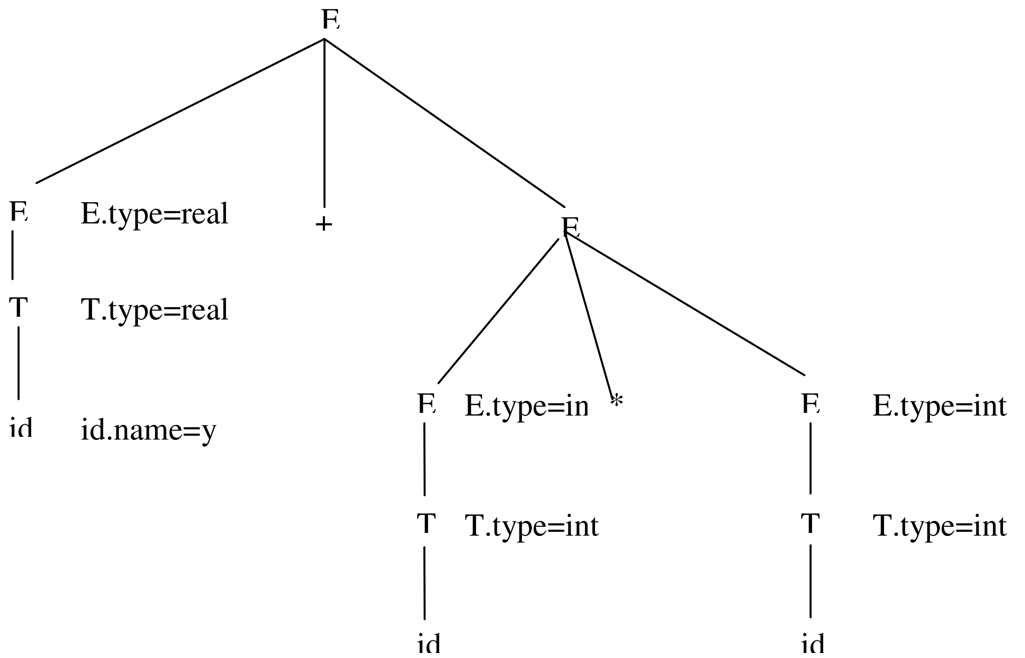
3	E->T	E.Type=T.Type

Fig. 2 Translation scheme for the assignment statement including type conversion

Example:

x = y + i * j

assume x and y are real variables, and i and j are integer variables



The output would be like this

t1=i*j
t2=inttoreal(t1)
t3=y*t2
x=t3

First E is derived to E+E in which first E is derived to T and further derived to id. T.type will be real since the variable y is real. Second E is derived to E*E in which first E is derived to T and further derived to id. T.type will be integer and the second E is further divdide into id and T.type is integer since the variable j is integer. When the semantics at the appropriate places we'll get the TAC for the above assignment statement.