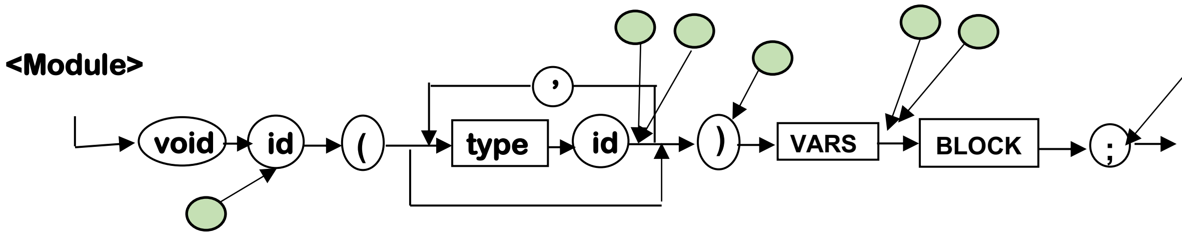
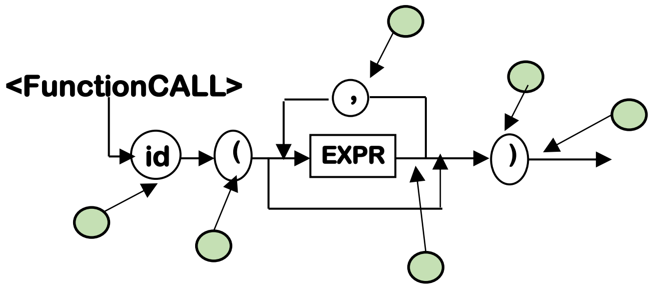
Intermediate Code Actions for a Module Definition



1.- Insert Procedure name into the DirFunc table, verify semantics.
2.- Insert every parameter into the current (local) VarTable.
3.- Insert the type to every parameter uploaded into the VarTable. At the same time into the **ParameterTable** (to create the Function's signature)..
4.- Insert into DirFunc the number of parameters defined. *****to calculate the workspace required for execution***
5.- Insert into DirFunc the number of local variables defined. *****to calculate the workspace required for execution***
6.- Insert into DirFunc the current quadruple counter (CONT), *****to establish where the procedure starts***
7.- Release the current VarTable (local).
Generate an action to end the procedure (ENDFUNC).
Insert into DirFunc the number of temporal vars used. *****to calculate the workspace required for execution***

Intermediate Code Actions for a Module Call



1.- Verify that the procedure exists into the DirFunc.
2.- Generate action **ERA size** (*Activation Record expansion –NEW—size).*
Start the parameter counter (k) in 1.
Add a pointer to the first parameter type in the ParameterTable.
3.- Argument= PilaO.Pop() ArgumentType= PTypes.Pop().
Verify ArgumentType against current Parameter (#k) in ParameterTable.
Generate action **PARAMETER, Argument, Argument#**k
4.- K = K + 1, move to next parameter.
5.- Verify that the last parameter points to null *(coherence in number of parameters).*
6.- Generate action **GOSUB**, **procedure-name, , initial-address**.

What to do on Run-Time to execute some of these new Operation-Codes

ERA size: Save the current Memory pointer (in case it is an Activation Record).. Generate Memory (Activation-Record) to store arguments and local variables according to the size needed for that procedure (--LocalMemory--).

PARAMETER Argument, Argument#k**:**
Copy the value sent as Argument into the current ActivationRecord in position #k

GOSUB procedure-name, , initial-address:
Save the current IP (Instruction-Pointer). Update IP with initial-address (if not explicit, use the address stored in DirFunc for that procedure). Transfer the Control-Flow to that address and continue.

ENDFUNC: Update the current memory (prior to the call). Erase Memory (Activation Record). Update IP (prior to the call). Transfer the Control-Flow to that address.

What if we have functions (with a return value) and Parameters sent by Reference?

EXAMPLE

```
program patito;
var
 int a, b;
 float f;
proc uno(int a)
{
 a= a+b*a;
 write(a, b, a+b);
}
proc dos(int a, int b, float g)
{  int i;
 i=b;
 while (i>0)
 {   a=a+b*i+b;
     uno(i*2);
     write(a);
     i=i-1;
 }
}
main()
{a=3; b=a+1;
 write(a, b);
 f=3.14;
 dos ( a+b*2, b, f*3);
 write(a,b,f*2+1);
}
```

1	goto			**26**
2	*	b	a	t1
3	+	a	t1	t2
4	=	t2		a
5	write	a		
6	write	b		
7	+	a	b	t3
8	write	t3		
9	**endfunc**			
10	=	b		i
11	>	i	0	t1
12	gotof	t1		**25**
13	*	b	i	t2
14	+	a	t2	t3
15	+	t3	b	t4
16	=	t4		a
17	**era**	**uno**		
18	*	i	2	t5
19	param	t5		*param1*
20	**gosub**	**uno**		
21	write	a		
22	-	i	1	t6
23	**=**	t6		i
24	**goto**			**11**
25	**endfunc**			
26	=	3		a
27	+	a	1	t1
28	=	t1		b
29	write	a		
30	write	b		
31	=	3.14		f
32	**era**	**dos**		
33	*	b	2	t2
34	+	a	t2	t3
35	param	t3		*param1*
36	param	b		*param2*
37	*	f	3	t4
38	param	t4		*param3*
39	**gosub**	**dos**		
40	write	a		
41	write	b		
42	*	f	2	t5
43	+	t5	1	t6
44	write	t6		
45	**end**			