**Assignment 3 solutions Using RDP**

**\*Semantic actions are under-lined**

1. **Assignment Statement**

A1) A -> id = E { gen\_instr (POPM, get\_address(id)) }

A2) E -> T E’

A3) E’ -> + T { gen\_intsr (ADD, nil) } E’

A4) E’ -> ε

A5) T -> F T’

A6) T’ -> \*F { gen\_instr (MUL, nil) } T’

A7) T’ -> ε

A8) F -> id { gen\_instr (PUSHM, get\_address(id) )

**Procedure A ()**

begin

If token = id then

begin

save = token;

lexer();

If token = “=” then

begin

lexer();

E();

get\_instr (POPM, get\_address (save) );

end

else error\_message ( “= expected” );

end

else error\_massage ( “ id expected” );

end;

**Procedure E ():**

Begin

T ();

E’();

end

**Procedure E’();**

Begin

If token = “+” then

begin

lexer();

T();

gen\_instr (ADD, nil);

E’();

end

end;

**Procedure T();**

begin

F();

T’();

End

**Procdure T’()**

begin

If token = “\*” then

begin

lexer();

F();

gen\_instr(MUL, nil);

T’();

end

end

**Procedure F();**

begin

If token = id then

begin

gen\_instr(PUSHM, get\_address (token));

lexer();

end

else error\_message(“id expected”);

end;

**Procedure gen\_instr(op, oprnd)**

**/\* instr\_address shows the current insturction address is global \*/**

begin

Instr\_table [instr\_address].address = inst\_address;

Instr\_table [instr\_address].op = op;

Instr\_table [instr\_address].oprnd = oprnd;

Instr\_address++;

end;

**Example:**

x = a + b\*c

(addresses a = 2001 , b=2002, c=2003 and x = 2004)

**INSTR\_TABLE**

|  |  |  |
| --- | --- | --- |
| address | Op | Oprnd |
| 1 | PUSHM | 2001 |
| 2 | PUSHM | 2002 |
| 3 | PUSHM | 2003 |
| 4 | MUL | nil |
| 5 | ADD | nil |
| 6 | POPM | 2004 |
|  |  |  |

**Print from INSTR\_TABLE ignoring “nil”**

2. While Statement

W1) W -> while ( C ) S

W2) C - > E R E

W3) R -> == | ^= | > | < | => | <=

**Procedure while\_statement();**

begin

If token = “while” then

begin

addr = instr\_address;

gen\_instr(“LABEL”, nil);

lexer();

If token = “(“ then

begin

lexer();

C ( );

If token = “)” then

begin

lexer();

S();

gen\_instr(JUMP, addr);

back\_patch (instr\_address);

end;

else error\_message (“ ) expected”);

else error\_message (“( expected”);

end

else error\_message (“while expected”);

end;

**Procedure C ()**

begin

E();

If token in R then

begin

op = token;

lexer();

E();

case op of

< : gen\_instr (LES, nil);

push\_jumpstack(instr\_address); /\* another stack need \*/

gen\_instr (JUMPZ, nil);

> : /\* you need to do other operators\*/

== :

^= :

etc.

endcase

end

else error\_message (“ R token expected”);

end

**Procedure back\_patch (jump\_addr)**

begin

addr = pop\_jumpstack();

Instr\_table[addr].oprn = jump\_addr;

end

**Example: while ( i < max) i = i + 1;**

**with addresses I =2000, max = 2001**

1. LABEL nil
2. PUSHM 2000
3. PUSHM 2001
4. LES nil
5. JUMPZ 11 /\* back patch \*/
6. PUSHM 2000
7. PUSHM 2001
8. ADD nil
9. POPM 2000
10. JUMP 1
11. …..

**3. if statement**

**I -> if ( C ) S endif**

**Procedure I ();**

begin

If token =”if” then

begin

addr = instr\_address();

lexer();

If token =”(“ then

begin

lexer();

C( );

If token = “)” then

begin

lexer();

S( );

back\_patch(instr\_address);

If token = “endif”

lexer();

else error\_messgage (“endif expected “);

end

else error\_message (“) expected “);

end;

else error\_message (“) expected”);

end

else error\_message (“if expected”);

end;

**Example: if (a < b) a = c; endif**

**With addresses a = 2000, b = 2001, c = 2002**

1. PUSHM 2000
2. PUSHM 2001
3. LES nil
4. JUMPZ 7
5. PUSHM 2002
6. POPM 2000

**NOTE:**

* **You need work on <Compound>, <Scan>and <Print> statement**
* **DO NOT create your own instructions**