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))

Function A ()

```
{
If token = id then
    {
    save = token;
    lexer();
    If token = "=" then
        {
        lexer();
        E();
        gen_instr (POPM, get_address (save) );
        }
```

```
        else error_message ( "= expected" );
        }
    else error_massage ( " id expected" );
    }
```

Function E ():

```
{
 T ();
 E'();
}
```

Function E'();

```
{
 If token = "+" then
        {
        lexer();
        T();
        gen_instr (ADD, nil);
        E'();
        }
 };
```

Function T();

```
{
F();
T'();
}
```

Function T'()

```
{
 If token = "*" then
      {
       lexer();
       F();
       gen_instr(MUL, nil);
        T'();
        }
 }
```

Function F();

```
{
If token = id then
    {
    gen_instr(PUSHM,  get_address (token));
    lexer();
    }
else error_message("id expected");
};
```

Function gen_instr(op, oprnd)

/* instr_address shows the current insturction address is global */

```
{
 Instr_table [instr_address].address = inst_address;
 Instr_table [instr_address].op = op;
Instr_table [instr_address].oprnd = oprnd;
 Instr_address++;
};
```

Example:

x = a + b*c

(addresses a = 10001 , b=10002, c=10003 and x = 10004)

INSTR_TABLE

address	Op	Oprnd
1	PUSHM	10001
2	PUSHM	10002
3	PUSHM	10003
4	MUL	nil
5	ADD	nil
6	POPM	10004

Print from INSTR_TABLE ignoring "nil"

2. While Statement

W1) W -> while (C) S

W2) C -> E R E

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

Function while_statement();

{

If token = “while” then

{

```
    addr = instr_address;
    gen_instr("LABEL", nil);
    lexer();
    If token = "(" then
        {
        lexer();
        C ( );
        If token = ")" then
            {
             lexer();
             S();
             gen_instr(JUMP, addr);
             back_patch (instr_address);
             lexer();
             };
         else error_message (" ) expected");
     else error_message ("( expected");
     }
else error_message ("while expected");
};
```

Function C ( )

```
{
E();
If token in R then
    {
     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
       }
else error_message (" R token expected");
}
```

Function back_patch (jump_addr)

```
{
addr = pop_jumpstack();
Instr_table[addr].oprn = jump_addr;
}
```

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

with addresses I =10000, max = 10001

1. LABEL nil
2. PUSHM 10000
3. PUSHM 10001
4. LES nil
5. JUMPZ 11 /* back patch */
6. PUSHM 10000
7. PUSHI 1
8. ADD nil
9. POPM 10000
10. JUMP 1
11. .....

3. if statement

I -> if (C) S fi

Function I ();

```
{
If token ="if" then
   {
   lexer();
   If token ="(" then
       {
       lexer();
       C( );
       If token = ")" then
            {
            lexer();
            S( );
            back_patch(instr_address);
            If token = "fi"
                lexer();
            else error_messgage ("endif expected ");
            }
        else error_message (") expected ");
       };
    else error_message ("( expected");
```

```
    }
else error_message ("if  expected");
};
```

Example: if (a < b) a = c; fi

With addresses a = 10000, b = 10001, c = 10002

1. PUSHM 10000
2. PUSHM 10001
3. LES nil
4. JUMPZ 7
5. PUSHM 10002
6. POPM 10000
7. LABEL

NOTE:

- **You need work on <Compound>, <Get> and <Put> statement – see the example**
- **DO NOT create your own instructions**