

Final project – implementing a compiler in Lex and Yacc

- Use Lex and Yacc to generate a compiler for **Micro/Ex**
- Micro/Ex is an extension of Micro.

```
%%the beginning of an test data for Micro/Ex
Program testP
```

```
Begin
declare I as integer;
declare A,B,C,D, LLL[100] as float;
```

```
FOR (I:=1 TO 100)
  A:=-LLL[I]+B*D-C;
ENDFOR
```

```
IF (A>=10000.0) THEN
  print(1);
ELSE
  print(2,1.4);
ENDIF
```

```
End
```

Micro/Ex is an extension of Micro (Cont'd)

- Variables must be declared before referenced.
- FOR construct

```
FOR (I:=1 TO 100)  
  A:=-LLL[I]+B*D-C;  
ENDFOR
```

```
FOR (I:=100 DOWNT0 1)  
  A:=-LLL[I]+B*D-C;  
ENDFOR
```

Micro/Ex is an extension of Micro (Cont'd)

- IF-ENDIF and IF-ELSE-ENDIF construct

```
IF (A>=10000.0) THEN  
    print(5*3+1);  
ENDIF
```

```
IF (A>=10000.0) THEN  
    print(1);  
ELSE  
    print(2,1.4);  
ENDIF
```



Only simple Boolean
expression.

Micro/Ex is an extension of Micro (Cont'd)

- Subroutine call

```
IF (A>=10000.0) THEN  
    print(5*3+1);  
ENDIF
```

Each actual parameter
can be an expression.

```
IF (A>=10000.0) THEN  
    print(1);  
ELSE  
    print(2,1.4);  
ENDIF
```

It can have multiple
actual parameters.

Program exercise -Yacc

- Target Language
 - Three-address machine
 - Variable declaration instruction
 - Declare A, Integer
 - Declare A, Integer_array,20
 - Declare B, Float
 - Declare B, Float_array,20
 - Arithmetic instruction
 - I_SUB i1,i2,t
 - I_ADD i1,i2,t
 - I_DIV i1,i2,t
 - I_MUL i1,i2,t
 - I_UMINUS i1,t
 - INC I
 - » I=I+1
 - DEC I
 - » I=I-1

Program exercise -Yacc

- Arithmetic instruction
 - F_SUB f1,f2,t
 - F_ADD f1,f2,t
 - F_DIV f1,f2,t
 - F_MUL f1,f2,t
 - F_UMINUS f1,t
- Assignment
 - I_Store i1,t
 - F_Store f1,t
- Compare instruction
 - I_CMP i1,i2
 - F_CMP f1,f2
- Jump instruction
 - J,JE, JG, JGE, JL, JLE, JNE
- Subroutine operation
 - CALL rn,a1,a2
 - » rn: the name of the subroutine
 - » a1 and a2 could be integer literal, float point literal, or id.

Program exercise -Yacc

- Logical instruction
 - AND b1,b2,t
 - » t will be 0 or 1 after the execution of this instruction
 - OR b1,b2,t
 - » t will be 0 or 1 after the execution of this instruction
 - NOT b, t
 - » b will be 0 or 1 after the execution of this instruction

Program exercise -Yacc

```
%%the beginning of an test data for Micro/Ex
Program testP
```

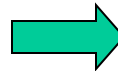
Begin

```
declare I as integer;
declare A,B,C,D, LLL[100] as float;
```

```
FOR (I:=1 TO 100)
  A:=-LLL[I]+B*D-C;
ENDFOR
```

```
IF (A>=10000.0) THEN
  print(A+3.14);
ELSE
  print(2,1.4);
ENDIF
```

End



```
START testP
Declare I, Integer
Declare A, Float
Declare B, Float
Declare C, Float
Declare D, Float
Declare LLL, Float_array,100
```

```
lb&1:  I_STORE 1,I
      F_UMINUS LLL[I],T&1
      F_MUL B,D,T&2
      F_ADD T&2, T&1, T&3
      F_SUB T&3,C,T&4
      F_STORE T&4,A
      INC I
      I_CMP I,100
      JLE lb&1
```

```
      F_CMP A,10000.0
      JL lb&2
      F_ADD A, 3.14, T&5
      CALL print, T&5
      J lb&3
```

```
lb&2:  CALL print,2,1.4
lb&3:  HALT testP
```

```
Declare T&1, Float
Declare T&2, Float
Declare T&3, Float
Declare T&4, Float
Declare T&5, Float
```


Bonus

- In case your Micro/Ex compiler can do the code generation of the following constructs

Program exercise -Yacc

- (1) To support more complex FOR construct

```
FOR (I:=1 TO 100*J+6 STEP 5)
```

```
  A:=-LLL[I]+B*D-C;
```

```
ENDFOR
```

```
FOR (I:=2*J-4 DOWNT0 5 STEP 4)
```

```
  A:=-LLL[I]+B*D-C;
```

```
ENDFOR
```

Program exercise -Yacc

- (2) To support WHILE construct

```
%%the beginning of an test data for Micro/Ex
Program testP

Begin
declare I as integer;
declare A,B,C,D, LLL[100] as float;

I:=1;
WHILE (I<=100)
  A:=-LLL[I]+B*D-C;
  I:=1+1;
ENDWHILE

IF (A>=10000.0) THEN
  print(A+3.14);
ELSE
  print(2,1.4);
ENDIF

End
```

Program exercise -Yacc

- (3) To support nested structure

```
%%the beginning of an test data for Micro/Ex
Program testP
```

```
Begin
```

```
declare I,J as integer;
```

```
declare A,B,C,D, LLL[100] as float;
```

```
I:=1;
```

```
WHILE (I<=100)
```

```
  A:=-LLL[I]+B*D-C;
```

```
  I:=1+1;
```

```
  FOR (I:=1 TO 100)
```

```
    A:=A*3.0;
```

```
  ENDFOR
```

```
ENDWHILE
```

```
IF (A>=10000.0) THEN
```

```
  IF (B<=0.0) THEN
```

```
    print(A+3.14);
```

```
  ELSE
```

```
    print(A+3.14*10);
```

```
  ENDIF
```

```
ELSE
```

```
  print(2,1.4);
```

```
ENDIF
```

```
End
```

Program exercise -Yacc

- (4) To support sophisticated logical expressions

```
WHILE ((I<=100) &&(A>10))
  A:=-LLL[I]+B*D-C;
  I:=1+1;
ENDWHILE

IF (!(A>=10000)|| (C<100)) THEN
  print(A+3.14);
ELSE
  print(2,1.4);
ENDIF

End
```

It adopts the logical expression of C .

Program exercise -Yacc

- (5) To support user-defined function and static type checking

```

%%the beginning of an test data for Micro/Ex
Program testP
Function integer Cal_Something(integer I, float f)
Begin
  declare k as integer;

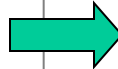
  .....
  return k;
End

Begin
  declare I,J as integer;
  declare A,B,C,D, LLL[100] as float;

  FOR (I:=1 TO 100)
    A:=-LLL[I]+B*D-C;
    J:=Cal_Something(I,A);
  ENDFOR

End

```



```

START testP
  Declare Cal_Something, Function,I,f
  Declare k, integer;
  .
  .
  Return k
  Declare I, Integer
  Declare J,Integer
  Declare A, Float
  Declare B, Float
  Declare C, Float
  Declare D, Float
  Declare LLL, Float_array,100

  I_STORE 1,I
  F_MUL B,D,T&1
  F_UMINUS LLL[I],T&2
  F_ADD T&2, T&1, T&3
  F_SUB T&3,C,T&4
  F_STORE T&4,A
  I_STORE Cal_Something(I,A),J
  INC I
  I_CMP I,100
  JL lb&1

  HALT testP

  Declare T&1, Float
  Declare T&2, Float
  Declare T&3, Float
  Declare T&4, Float

```

Note that variables declared in functions should not have the same names as variables defined in main program.

Project Report

- Prepare a compressed a file with the following items
 - The source codes and execution results
 - If you have your own test data, you can show it.
 - A report in **pdf** file format
 - What you have learned and experienced during the implementation of Micro/Ex compiler.
 - E.g. You could show your daily record of the implementation.
 - In case you implement more than the required specification, please itemize it.
 - Copyright Claim
 - Do you make the implementation yourself?
 - Any thing you would like to let G.H.Hwang know.
 - E.g. Suggestion, ...