Υλοποίηση ΣΑ με το bison (i)

- Μεταεργαλείο bison: γεννήτορας ΣΑ LALR(1)
- Είσοδος: μεταπρόγραμμα που περιγράφει τη σύνταξη και τις σημασιολογικές ρουτίνες
- Έξοδος: πρόγραμμα σε C
  - Η συνάρτηση yyparse υλοποιεί το ΣΑ
  - Επιστρέφει 0 αν αναγνωριστεί η συμβολοσειρά εισόδου ή 1 σε περίπτωση συντακτικού σφάλματος
  - Συνεργάζεται με το λεκτικό αναλυτή (συνάρτηση yylex)

Υλοποίηση ΣΑ με το bison (ii)

- Δομή του μεταπρογράμματος

  Μέρος Α
  %%
  Μέρος Β
  %%
  Μέρος Γ

  Και τα τρία μέρη μπορούν να είναι κενά

Υλοποίηση ΣΑ με το bison (iii)

- Μέρος Α, περιέχει
  - Σχόλια, όπως στη C
  - Κώδικα C, μέσα σε `%{` και `%}`
  - Δηλώσεις λεκτικών μονάδων
  - Δηλώσεις τελεστών της αρχικής γλώσσας (προτεραιότητα, προσεταιριστικότητα)
  - Δήλωση του συνόλου σημασιολογικών τιμών (τύπος `YYSTYPE` ή με χρήση του `%union`)
  - Δήλωση του τύπου της σημασιολογικής τιμής κάθε συμβόλου

Υλοποίηση ΣΑ με το bison (iv)

- Μέρος Α, παράδειγμα

```
%{
void ERROR (const char msg []);
%}

%token T_program ”program”
%token T_div T_mod
%token T_if T_then T_else

%nonassoc ’=’ ’<’ ’>’
%left ’+’ ’-’
%left ’*’ ’/’ T_div T_mod
```

Υλοποίηση ΣΑ με το bison (v)

- Μέρος Α, παράδειγμα *(συνέχεια)*

```
%union{
  int i;
  double f;
  char str[80];
}

%token<str> T_id
%token<i>   T_int_const
%token<f>   T_float_const

%type<f>    expression
```

Υλοποίηση ΣΑ με το bison (vi)

- Μέρος Β, περιέχει:
  - τους κανόνες παραγωγής σε μορφή BNF
  - σημασιολογικές ρουτίνες που εκτελούνται κατά τη συντακτική ανάλυση
- Οι κανόνες έχουν τη μορφή:

$$
\begin{array}{lll}
A & : & x_1^1\ x_2^1\ \ldots\ x_{m_1}^1 \\
 & | & x_1^2\ x_2^2\ \ldots\ x_{m_2}^2 \\
 & & \ldots \\
 & | & x_1^n\ x_2^n\ \ldots\ x_{m_n}^n \\
 & ; &
\end{array}
$$

Υλοποίηση ΣΑ με το bison (vii)

- Μέρος Β, παράδειγμα

```
program    : { count=0; } block_list
             { printf("Counted %d block(s)\n",
                      count); }
           ;

block_list : /* nothing */
           | block_list block { count++; }
           ;

block      : "begin" block_list "end"
           ;
```

Υλοποίηση ΣΑ με το bison (viii)

- Μέρος Γ, περιέχει κώδικα C
- Το μεταπρόγραμμα του bison αναλαμβάνει τον κεντρικό έλεγχο του μεταγλωττιστή που επιτυγχάνεται με τη συνεργασία των παρακάτω:
  - του λεκτικού αναλυτή
  - του συντακτικού αναλυτή
  - του πίνακα συμβόλων
  - του σημασιολογικού αναλυτή
  - του γεννήτορα ενδιάμεσου κώδικα

Υλοποίηση ΣΑ με το bison (ix)

- Μέρος Γ, παράδειγμα

```
void yyerror (const char * msg)
{
   fprintf(stderr,
           ”syntax error in line %d: %s\n”,
           linecount, msg);
   exit(1);
}

int main ()
{
   return yyparse();
}
```

Υλοποίηση ΣΑ με το bison (x)

- Παράδειγμα με σημασιολογικές τιμές

$$E \rightarrow T$$
$$E \rightarrow E + T$$
$$T \rightarrow F$$
$$T \rightarrow T * F$$
$$F \rightarrow (E)$$
$$F \rightarrow \text{num}$$

- Ζητούμενο: να κατασκευαστεί ΣΑ που να υπολογίζει την τιμή μιας αριθμητικής έκφρασης

Υλοποίηση ΣΑ με το bison (xi)

- Παράδειγμα *(συνέχεια)*

```
%{
typedef int YYSTYPE;
%}

%token T_num

%%

program :
   expression { printf(”Value: %d\n”, $1); }
 ;
```

Υλοποίηση ΣΑ με το bison (xii)

- Παράδειγμα *(συνέχεια)*

```
expression :
   term { $$ = $1; }
 | expression '+' term { $$ = $1 + $3; }
 ;

term :
   factor { $$ = $1; }
 | term '*' factor { $$ = $1 * $3; }
 ;
```

Υλοποίηση ΣΑ με το bison (xiii)

- Παράδειγμα *(συνέχεια)*

```
factor :
   '(' expression ')' { $$ = $2; }
 | T_num { $$ = $1; }
 ;

%%
```

- Παραλείπονται στο Μέρος Γ:
  - η συνάρτηση `yylex` (πιθανώς σε ξεχωριστό αρχείο, αν χρησιμοποιηθεί το flex)
  - οι συναρτήσεις `yyerror` και `main`

Υλοποίηση ΣΑ με το bison (xiv)

- Παράδειγμα — Υλοποίηση ΛΑ χειρωνακτικά

```
int yylex ()
{
  int c;

  while (isspace(c = fgetc(stdin)));
  if (isdigit(c)) {
    yylval = c - '0';
    while (isdigit(c = fgetc(stdin)))
      yylval = yylval * 10 + c - '0';
    ungetc(c, stdin);
    return T_num;
  }
```

Στο μέρος Γ

Υλοποίηση ΣΑ με το bison (xv)

- Παράδειγμα *(συνέχεια)*

```
  if (strchr("+*()", c)) return c;
  if (c != EOF)
    fprintf(stderr, "Illegal character: %c\n", c);
  return 0;
}
```

- Αυτοματοποίηση της μεταγλώττισης του ΣΑ

```
mytest1: mytest1.y
        bison mytest1.y
        gcc -o mytest1 mytest1.tab.c
```

Makefile

Υλοποίηση ΣΑ με το bison (xvi)

- Παράδειγμα — Υλοποίηση ΛΑ με το flex

```
%{
#include ”mytest2.tab.h”
%}

%%

[0-9]+ { yylval = atoi(yytext); return T_num; }

\(|\)|\+|\* { return yytext[0]; }
[ \t\n]+    { /* nothing */ }
.           { yyerror(”illegal character”); }

%%
```

mytest2.l

Υλοποίηση ΣΑ με το bison (xvii)

- Αυτοματοποίηση της μεταγλώττισης ΛΑ και ΣΑ

Makefile

```
mytest2: mytest2.l mytest2.y
         bison -d mytest2.y
         flex -s mytest2.l
         gcc -o mytest2 mytest2.tab.c lex.yy.c \
                -lfl
```

- Επίλυση συγκρούσεων στο bison
  - shift-reduce: πάντα shift
  - reduce-reduce: ο πρώτος κανόνας