Lecture 4

Data structures

- Tables APT, PDT and EVT contain pairs which are searched using the first component of the pair as key
  - Eg. Formal parameter name is used as key to obtain its value from APT

- This search is eliminated if the position of an entity within a table is known when its value is to be accessed

- Value of formal parameter ABC is needed while expanding a model statement using it

  MOVER AREG, &ABC

- Let the pair (ABC, ALPHA) occupy entry#5 in APT

- The search in APT can be avoided if model statement appears as

  MOVER AREG, (P,5)

  (P,5) stands for words ‘parameter#5’

- Macro expansion can be made more efficient by storing an intermediate code for a statement, rather than its source form in the MDT

- All parameter names could be replaced by pairs of the form (P, n) in model statements and preprocessor statements stored in MDT

- The first component of the pairs stored in APT is no longer used during macro expansion

- Eg. (P,5) appearing in model statement is sufficient to access the value of formal parameter ABC

- Hence APT containing (<formal parameter name>,<value>) pairs is replaced by another table called **APTAB** which only **contain <value>’s**

- To implement this, ordinal numbers are assigned to all parameters of a macro
- Table named **parameter name table (PNTAB)** is used for this purpose

- PNTAB is used while processing the definition of a macro

- **Parameter names are entered in PNTAB** in same order in which they appear in prototype statement

- Entry# of parameter's entry in PNTAB is now its ordinal number

- This entry is used to replace the parameter name in the model and preprocessor statements of the macro while storing it in MDT

- This implements the requirement that the statement MOVER AREG, &ABC should appear as MOVER AREG, (P,5) in MDT

- APT has been split into two tables
  - **PNTAB** – contains formal parameter names
  - **APTAB** – contains formal parameter values (i.e. actual parameters)

- **PNTAB is used while processing a macro definition**
- **APTAB is used during macro expansion**

- Similar analysis leads to splitting of
  - **EVT into EVTAB and EVNTAB**
  - **SST into SSNTAB and SSTAB**

- EV names are entered in EVNTAB while processing EV declarations
- SS names are entered in SSNTAB while processing SS reference or definition

- This arrangement leads to some simplifications concerning PDT

- Positional parameters of a macro appear before keyword parameters in prototype statement

- Hence in prototype statement for a macro **BETA** which has **p** positional parameters and **k** keyword parameters, the keyword parameters have the ordinal numbers p+1,.. p+k

- Due to this numbering, two kinds of redundancies appear in PDT
  - First component of each entry is redundant as in APTAB and EVTAB
  - Entries 1…p are redundant since positional parameters cannot have default specifications

- Entries only need to exist for parameters numbered p+1 … p+k

- To accommodate these changes, replace parameter default table (PDT) by a **keyword parameter default table (KPDTAB)**

- **KPDTAB** of macro BETA would only have **k entries** in it

- The first entry of KPDTAB corresponds to parameter numbered p+1, we store **p**, the no of positional parameters of BETA, in a new field of MNT entry

- MNT has entries for all macros defined in program

- Each MNT entry contains three pointers
  - **MDTP – pointer to MDT**
  - **KPDTP – pointer to KPDTAB**
  - **SSTP – pointer to SSNTAB**

Summary

- PNTAB and KPDTAB are constructed by processing prototype statement
- Entries are added to EVNTAB and SSNTAB as EV declarations and SS definitions/references are encountered
- MDT entries are constructed while processing model statements and preprocessor statements in macro body

Summary

- Entry is added to SSTAB when definition of SS is encountered
- APTAB is constructed while processing a macro call
- EVTAB is constructed at the start of expansion of macro

Table	*Fields in each entry*
Macro name table (MNT)	**Macro name, no of positional parameters (#PP), no of keyword parameters (#KP), no of expansion time vars (#EV), MDT pointer (MDTP), KPDTAB ptr (KPDTP), SSTAB ptr (SSTP)**
Parameter name table (PNTAB)	**Parameter name**
EV name table (EVNTAB)	**EV name**
SS name table (SSNTAB)	**SS name**
Keyword parameter default table (KPDTAB)	**Parameter name, default value**
Macro definition table (MDT)	**Label, Opcode, operands**
Actual parameter table (APTAB)	**Value**
EV table (EVTAB)	**Value**
SS table (SSTAB)	**MDT entry#**

Example

- Create the data structures for the following macro

```
            MACRO
            CLEARMEM    &X, &N, &REG=AREG
            LCL         &M
&M          SET         0
            MOVER       &REG, =‘0’
.MORE       MOVEM       &REG, &X+&M
&M          SET         &M+1
            AIF         (&M NE &N)  .MORE
            MEND
```

- The macro call is : `CLEARMEM  AREA, 10`

Processing of Macro Definitions

- Following initializations are performed before initiating the processing of macro definitions in a program
  - **KPDTAB_ptr := 1;**
  - **SSTAB_ptr := 1;**
  - **MDT_ptr := 1;**

- The algorithm (next slide) is invoked for every macro definition in program

Algo for Processing a macro definition

1. SSNTAB_ptr := 1;
   PNTAB_ptr := 1;

2. Process macro prototype stmt and form the MNT entry
   a) name := macro name
   b) For each positional parameter
      i. Enter parameter name in PNTAB[PNTAB_ptr];
      ii. PNTAB_ptr := PNTAB_ptr + 1;
      iii. #PP := #PP +1;

c) KPDTP := KPDTP_ptr;

d) For each keyword parameter

   i. Enter parameter name and default value (if any), in KPDTAB[KPDTAB_ptr]

   ii. Enter parameter name in PNTAB[PNTAB_ptr]

   iii. KPDTAB_ptr := KPDTAB_ptr + 1;

   iv. PNTAB_ptr := PNTAB_ptr + 1;

   v. #KP := #KP + 1;

e) MDTP := MDTP_ptr;

f) #EV := 0;

g) SSTP := SSTAB_ptr;

3. While not MEND stmt

a) If LCL stmt then

i. Enter expansion time variable name in EVNTAB

ii. #EV := #EV + 1;

b) If model stmt then

i. If label field contains a SS then

q:= entry no in SSNTAB;

else

enter symbol in SSNTAB[SSNTAB_ptr];

q:= SSNTAB_ptr;

SSTAB[SSTP + q -1] := MDT_ptr;

ii. For a parameter, generate the specification (P,#n)
iii. For expansion variable, generate specification (E,#m)
iv. Record the IC in MDT[MDT_ptr];
v. MDT_ptr := MDT_ptr + 1;

c) If preprocessor stmt then
i. If SET stmt then
search each expansion time variable name used in the stmt in EVNTAB and generate spec (E,#m)

ii. If AIF or AGO stmt then

```
    if SS used in stmt is present in SSNTAB then
            q:= entry no in SSNTAB;
    else
            enter symbol in SSNTAB[SSNTAB_ptr]
            q:= SSNTAB_ptr;
            SSNTAB_ptr := SSNTAB_ptr + 1;
    replace symbol by (S, SSTP + q -1)
```

iii. Record the IC in MDT[MDT_ptr]

iv. MDT_ptr := MDT_ptr + 1;

4. (MEND stmt)

   if SSNTAB_ptr =1 (i.e. SSNTAB is empty) then

      SSTP:=0;

   else SSTAB_ptr := SSTAB_ptr + SSNTAB_ptr - 1;

   if #KP =0 then KPDTP :=0;

Macro expansion

- Structures used
  - APTAB – actual parameter table
  - EVTAB – expansion variable table
  - MEC – macro expansion counter
  - APTAB_ptr – pointer to APTAB
  - EVTAB_ptr – pointer to EVTAB

- No of entries in APTAB ($\#e_{APTAB}$) equals sum of values in #PP and #KP fields of MNT entry
- No of entries in EVTAB ($\#e_{EVTAB}$) is given by value in #EV field of MNT
- APTAB and EVTAB are constructed when a macro call is recognized
- APTAB_ptr and EVTAB_ptr are set to point at these tables

Macro expansion (algo)

- Please read by yourself page no 153