MACROS WITH MIXED PARAMETER LISTS

- A macro may be defined to use both positional and keyword parameters.
- In such a case, all positional parameters must precede all keyword parameters.
- example in the macro call

SUMUP A, B, G=20, H=X

- A, B are positional parameters while G, H are keyword parameters.
- Correspondence between actual and formal parameters is established by applying the rules governing positional and keyword parameters separately.

operands na alawa tame to label nu name ne pan parameters tarifke specify karo 6o

OTHER USES OF PARAMETERS

- The model statements have used formal parameters only in operand field.
- However, use of parameters is not restricted to these fields.
- Formal parameters can also appear in the label and opcode fields of model statements.
- Example:

```
  MCRO
  CALC        &X, &Y, &OP=MULT, &LAB=
  &LAB        MOVER AREG, &X
              &OP   AREG,  &Y
              MOVEM  AREG,  &X
              MEND
```

- Expansion of the call CALC A, B, LAB=LOOP leads to the following code:

```
+ LOOP        MOVER AREG, A
+             MULT AREG, B
+             MOVEM  AREG, A
```

NESTED MACRO CALLS

- A model statement in a macro may constitute a call on another macro.
- Such calls are known as nested macro calls.
- Macro containing the nested call is the outer macro and,
- Macro called is inner macro.
- They follow LIFO rule.
- Thus, in structure of nested macro calls, expansion of latest macro call (i.e inner macro) is completed first.

EXAMPLE:

```
MACRO
COMPUTE     &FIRST,&SECOND,
MOVEM       ,BREG,TMP
INCR_D      &FIRST, &SECOND, REG=BREG
MOVER       BREG, TMP
MEND
```

```
COMPUTE X,Y:
    +MOVEM          BREG, TMP         [1]
    +INCR_D         X,Y               [2]
       + MOVER      BREG,X            [3]
       + ADD        BREG,Y            [4]
       + MOVEM      BREG,X            [5]
    +MOVER          BREG,TMP
```

Final Answer

```
+   MOVEM    BREG, TMP
+   MOVER    BREG, X
+   ADD      BREG, Y
+   MOVEM    BREG, X
+   MOVER    BREG, TMP
```

Advanced Macro Facilities

- Advanced macro facilities are aimed to supporting semantic expansion.
- Used for:
  - performing conditional expansion of model statements and
  - in writing expansion time loops.
- These facilities can be grouped into following.
  - 1. Facilities for alteration of flow of control during expansion.
  - 2. Expansion time variables.
  - 3. Attributes of parameters.

1. Alteration of flow of Control during expansion

- Two features are provided to facilitate alteration of flow of control during expansion.
- 1. Expansion time sequencing symbol
- 2. Expansion time statements
  - AIF,
  - AGO and
  - ANOP.
- A *sequencing symbol (SS)* has the syntax

  .<ordinary string>
- As SS is defined by putting it in the label field of statement in the macro body.
- It is used as an operand in an AIF or AGO statement to designate the destination of an expansion time control transfer.

AIF STATEMENT

- An AIF statement has the syntax

  AIF (<expression>) <sequencing symbol>

- Where <expression> is a relational expression involving ordinary strings, formal parameters and their attributes and expansion time variables.
- If the relational expression evaluates to true, expansion time control is transferred to the statement containing <sequencing symbol> in its label field.

An AGO statement

- An AGO statement has the syntax

  AGO <sequencing symbol>

- Unconditionally transfers expansion time control to the statement containing <sequencing symbol> in its label field.

AN ANOP STATEMENT

- An ANOP statement is written as

  <sequencing symbol>   ANOP

- And simply has the effect of defining the sequencing symbol.

what is used to define the sequencing symbol?

EXAMPLE: A-B+C

we define it before defining the macro statements

```
          ONLY      ANOP
          OVER      ANOP
          MACRO
          EVAL    &X, &Y, &Z
          AIF    (&Y EQ &X)  .ONLY
          MOVER  AREG, &X
          SUB    AREG,  &Y
          ADD    AREG,  &Z
          AGO    .OVER
.ONLY     MOVER  AREG, &Z
.OVER     MEND
```

- It is required to develop a macro EVAL such that a call EVAL A,B,C generates efficient code to evaluate A-B+C in AREG.
- When the first two parameters of a call are identical, EVAL should generate single MOVER instruction to load 3rd parameter into AREG.
- As formal parameter is corresponding to actual parameter, AIF statement effectively compares names of first two actual parameters.
- If condition is true, expansion time control is transferred to model statement MOVER AREG, &Z.
- If false, MOVE-SUB-ADD sequence is generated and expansion time control is transferred to statement .OVER MEND which terminates expansion.
- Thus, efficient code is generated under all conditions.

2. EXPANSION TIME VARIABLE

- Expansion time variables (EV"s) are variables which can only be used during the expansion of macro calls.
- A local EV is created for use only during a particular macro call.
- A global EV exists across all macro calls situated in a program and can be used in any macro which has a declaration for it.
- Local and global EV"sare created through declaration statements with the following syntax:

LCL <EV specification> [, <EV specification>]

GBL <EV specification> [, <EV specification>]

- <EV specification> has the syntax

  &<EV name>

  where <EV name> is an ordinary string.
- Values of EV's can be manipulated through the preprocessor statement SET.
- A SET statement is written as

  <EV specification> SET <SET-expression>

  - Where <EV specification> appears in the label field and
  - SET in the mnemonic field.
- A SET statement assigns the value of <SET-expression> to the EV specified in <EV specification>.
- The value of an EV can be used in any field of a model statement,and in the expression of an AIF statement.

EXAMPLE

```
        MACRO
        CONSTANTS
        LCL             &A
&A      SET             1
        DB              &A
&A      SET             &A+1
        DB              &A
        MEND
```

- A call on macro CONSTANTS is expanded as follows.
- The local EV A is created.
- The first SET statement assigns the value „1‟to it.
- The first DB statement thus declares a byte constant „1‟.
- The second SET statement assigns the value „2‟ to A
- And the second DB statement declares a constant „2‟.

3. ATTRIBUTES OF FORMAL PARAMETERS

- An attribute is written using the syntax

  <attribute name>‟<formal parameter spec>

- And represents information about the value of the formal parameter,
- i.e. about the corresponding actual parameter.
- The type, length and size attributes have the names T,L and S

EXAMPLE

- Here expansion time control is transferred to the statement having .NEXT in its label field only if the actual parameter corresponding to the formal parameter A has the length of „1‟.

quotation marks kevi ritna hoi 6e??

```
           MACRO
           DCL_CONST  &A
           AIF   (L‟&A    EQ  1)  .NEXT
            ----
     .NEXT----
             ---
           MEND
```

Conditional Expansion

- While writing a general purpose macro it is important to ensure execution efficiency of its generated code.
- This is achieved by ensuring that a model statement is visited only under specific conditions during the expansion of a macro.
- How to do that?
- Ans: The AIF and AGO statements are used for this purpose.
- Let us take example which would clear our doubts for the same.

EXPANSION TIME LOOPS

- It is often necessary to generate many similar statements during the expansion of a macro.
- This can be achieved by writing similar model statements in the macro:
- Example
- MACRO
- CLEAR &A
- MOVER AREG, =‟0‟
- MOVEM AREG, &A
- MOVEM AREG, &A+1
- MOVEM AREG, &A+2
- MEND
- When called as CLEAR B, The MOVER statement puts the value „0‟ in AREG, while the three MOVEM statements store this value in 3 consecutive bytes with the addresses B, B+1 and B+2.

- Alternatively, the same effect can be achieved by writing an expansion time loop which visits a model statement, or a set of model statement repeatedly during macro expansion.
- Expansion time loops can be written using expansion time variables (EV's) and expansion time control transfer statements AIF and AGO.
- Consider expansion of the macro call

```
                              CLEAR  B, 3
```

- Example

```
              MACRO
              CLEAR         &X, &N
              LCL           &M
&M            SET           0
              MOVER         AREG, ='0'
.MORE         MOVEM         AREG, &X + &M
&M            SET           &M+1
              AIF           (&M NE &N)  .MORE
              MEND
```

OTHER FACILITIES FOR EXPANSION TIME LOOPS

- The assembler for M 68000 and Intel 8088 processors provide explicit expansion time looping constructs.
- <expression> should evaluate to a numerical value during macro expansion.
- The **REPT** statement

  REPT   <expression>
- Statements between **REPT** and an ENDM statement would be processed for expansion <expression> number of times.
- Following example use REPT to declare 10 constant with the value 1,2,…10.
- MACRO
- CONST10
- LCL &M
- &M SET 1
- REPT 10
- DC „&M‟
- &M SET &M+1
- ENDM
- MEND

The IRP Statement

2

- IRP <formal parameter> <argument list>
- The formal parameter mentioned in the statement takes successive values from the argument list.
- For each value, the statements between the IRP and ENDM statements are expanded once.
- MACRO

4 10

- CONSTS &M, &N, &Z
- IRP &Z, &M, 7, &N
- DC ,,&Z'
- ENDM
- MEND
- A macro call CONSTS 4, 10 leads to declaration of 3 constants with the value 4, 7 and 10.

formal parameter

argument list

number of times these
statements are generated (MOVEM, AREG) ...is by a call on the CLEAR statement's 2nd parameter

SEMANTIC EXPANSION

- Semantic expansion is the generation of instructions to the requirements of a specific usage.

3

- It can be achieved by a combination of advanced macro facilities like AIF, AGO statements and expansion time variables.
- The CLEAR example is an instance of semantic expansion. In this example the number of MOVEM AREG,….. statement generated by a call on CLEAR is determined by the value of the second parameter of CLEAR.
- Following example is another instance of conditional expansion wherein one of two alternative code sequences is generated depending on actual parameters of a macro call.

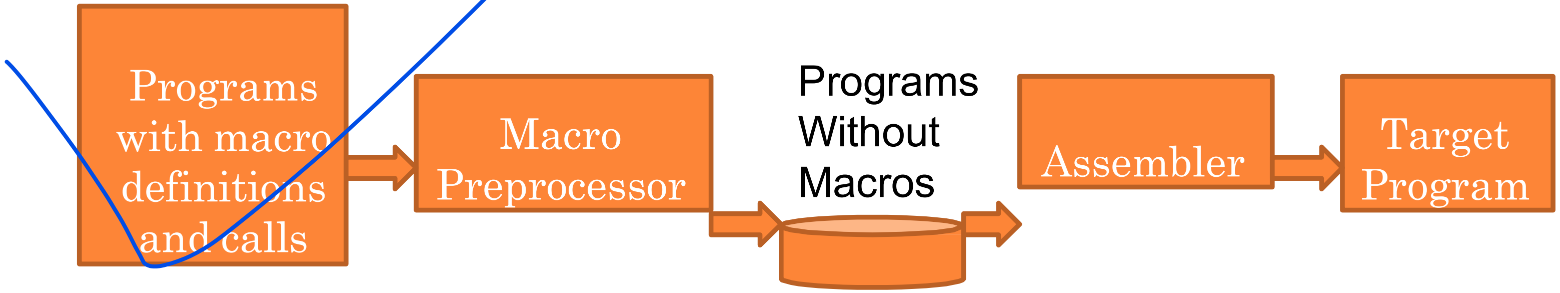
EXAMPLE

- This macro creates a constant „25‟with the name given by the 2nd parameter.
- The type of the constant matches the type of the first parameter.

```
            MACRO
            CREATE_CONST  &X, &Y
            AIF           (T'&X   EQ  B) .BYTE
 &Y         DW            25
            AGO           .OVER
 .BYTE      ANOP
 &Y         DB            25
 .OVER      MEND
```

DESIGN OF A MACRO PREPROCESSOR

- The macro preprocessor accepts an assembly program containing definitions and calls and translates it into an assembly program which does not contain any macro definitions and calls.
- The program form output by the macro preprocessor can be handed over to an assembler to obtain the target program.



DESIGN OVERVIEW

- We begin the design by listing all tasks involved in macro expansion.
  - 1. Identify macro calls in the program.
  - 2. Determine the values of formal parameters.
  - 3. Maintain the values of expansion time variables declared in a macro.
  - 4. Organize expansion time control flow.
  - 5. Determine the values of sequencing symbols.
  - 6. Perform expansion of a model statement.
- Following 4 step procedure is followed to arrive at a design specification for each task.
  - Identify the information necessary to perform a task.
  - Design a suitable data structure to record the information.
  - Determine the *processing* necessary to obtain the information.
  - Determine the *processing* necessary to perform the task.

1. Identify Macro Calls

- A table called the Macro Name Table (MNT) is designed to hold the names of all macros defined in a program.
- A macro name is entered in this table when macro definition is processed.
- While processing a statement in the source program, the preprocessor compares the string found in its mnemonic field with the macro names in MNT.
- A match indicate that the current statement is a macro call.

2. Determine the values of formal parameters

- A table called the Actual Parameter Table (APT) is designed to hold the values of formal parameters during the expansion of a macro call.
- Each entry in the table is a pair (<formal parameter name>, <value>).
- Two items of information are needed to construct this table, names of formal parameters and default values of keyword parameters.
- A table called the Parameter Default Table (PDT) is used for each macro.
- It would contain pairs of the form

  (<formal parameter name>, <default value>)
- If a macro call statement does not specify a value for some parameter par, its default value would be copied from PDT to APT.

3. MAINTAIN EXPANSION TIME VARIABLES

- An Expansion time Variable Table (EVT) is maintained for this purpose.
- The table contains pairs of the form (<EV name>, <value>).
- The value field of a pair is accessed when a preprocessor statement or a model statement under expansion refers to an EV.

4. ORGANIZE EXPANSION TIME CONTROL FLOW

- The body of a macro, i.e. the set of preprocessor statements and model statements in it, is stored in a table called the Macro Definition Table (MDT) for use during macro expansion.
- The flow of control during macro expansion determines when a model statement is to be visited for expansion.
- For this purpose MEC (Macro Expansion Counter) is initialized to the first statement of the macro body in the MDT.
- It is updated after expanding a model statement of on processing a macro preprocessor statement.

5. Determine values of sequencing symbols

- A Sequencing Symbols Table (SST) is maintained to hold this information.
- The table contains pairs of the form

  (<sequencing symbol name>, <MDT entry #>)
- Where <MDT entry #> is the number of the MDT entry which contains the model statement defining the sequencing symbol. aene define kyare karvama ayu
- This entry is made on encountering a statement which contains the sequencing symbol in its label field (for back reference to symbol) or on encountering a reference prior to its definition(forward reference).

6. Perform expansion of a model statement

- This is trivial task given the following:
  - 1. MEC points to the MDT entry containing the model statement.
  - 2. Values of formal parameters and EV's are available in APT and EVT, respectively.
  - 3. The model statement defining a sequencing symbol can be identified from SST.
- Expansion of a model statement is achieved by performing a lexical substitution for the parameters and EV's used in the model statement.

DATA STRUCTURE

- The tables APT, PDT and EVT contain pairs which are searched using the first component of the pair as a key.
- For example the formal parameter name is used as the key to obtain its value from APT.
- This search can be eliminated if the position of an entity within the table is known when its value is to be accessed.
- In context of APT, the value of a 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 the model statement appears as MOVER AREG, (P,5) in the MDT, where (P,5) stands for the words "parameter #5".
- Thus macro expansion can be made more efficient by storing an intermediate code for a statement in the MDT.

- All the parameter names could be replace by pairs of the form (P,n) in model statements and preprocessor statements stored in MDT.
- The information (P,5) appearing in a model statement is sufficient to access the value of formal parameter ABC. Hence APT containing (<formal parameter name> , <value>) is replace by another table called APTAB which only contains <value>"s.
- To implement this, ordinal numbers are assigned to all parameters of a macro.
- A 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 the same order in which they appear in the prototype statement.

- Its entry number is used to replace the parameter name in the model and preprocessor statements of the macro while storing it in the MDT.
- This implements the requirement that the statement MOVER AREG, &ABC should appear as MOVER

  AREG, (P,5) in MDT.
- In effect, the information (<formal parameter name>,<value>) in APT has been split into two table
  - PNTAB which contains formal parameter names.
  - APTAB which contains formal parameter values.
- PNTAB is used while processing a macro definition while APTAB is used during macro expansion.

- Similar Analysis leads to splitting
  - EVT into EVNTAB and EVTAB.
  - SST into SSNTAB and SSTAB.
- EV names are entered into EVNTAB while processing EV declaration statements.
- SS names are entered in SSNTAB while processing an SS reference or definition, whichever occurs earlier.
- Entries only need to exist for default parameter, therefore we replace the parameter default table (PDT) by a keyword parameter default table (KPDTAB).
- We store the number of positional parameters of macro in a new field of the MNT entry.
- MNT has entries for all macros defined in a program.
- Each MNT entry contains three pointers MDTP, KPDTP and SSTP, which are pointers to MDT, KPDTAB and SSNTAB.
- Instead of creating different MDT'sfor different macros, we can create a single MDT and use different sections of this table for different macros.

TABLES ARE CONSTRUCTED FOR MACRO PREPROCESSOR.

npk emks

Table	Fields
MNT (Macro Name Table)	Macro Name
	Number of Positional Parameter (#PP)
	Number of keyword parameter (#KP)
	Number of Expansion Time Variable (#EV)
	MDT pointer (MDTP)
	KPDTAB pointer (KPDTABP)
	SSTAB pointer (SSTP)

(CONTI…..) TABLES ARE CONSTRUCTED FOR MACRO PREPROCESSOR.

Tables	Fields
PNTAB (Parameter Name Table)	Parameter name
EVNTAB (EV Name Table)	EV Name
SSNTAB (SS Name Table)	SS Name
KPDTAB (Keyword Parameter Default Table)	Parameter name, default value
MDT (Macro Definition Table)	Label, Opcode, Operands Value
APTAB (Actual Parameter Table)	Value
EVTAB (EV Table)	Value
SSTAB (SS Table)	MDT entry #

CONSTRUCTION AND USE OF THE MACRO PREPROCESSOR DATA STRUCTURES CAN BE SUMMARIZED AS FOLLOWS.

- PNTAB and KPDTAB are constructed by processing the prototype statement.
- Entries are added to EVNTAB and SSNTAB as EV declarations and SS definitions/references are encountered.
- MDT are constructed while processing the model statements and preprocessor statements in the macro body.
- An entry is added to SSTAB when the definition of a sequencing symbol is encountered.
- APTAB is constructed while processing a macro call.
- EVTAB is constructed at the start of expansion of a macro.
- See Pg.151, Fig 5.8.

Processing of Macro Definitions

- The following initializations are performed before initiating the processing of macro definitions in a program
- KPDTAB_pointer:=1;
- SSTAB_ptr:=1;
- MDT_ptr:=1;
- Now let us see the algorithm which is invoked for every macro definition.

ALGORITHM (PROCESSING OF A MACRO DEFINITION)

```
1.  SSNTAB_ptr:=
    1;
    PNTAB_ptr:=1;
2. Process the macro prototype statement 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:=KPDTAB_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;
      a.MDTP:=MDT_ptr;
      f.  #EV:=0;
      g.  SSTP:=SSTAB_ptr;
```

```
3.  While not a MEND statement
    a.  If an LCL statement then
            i.  Enter expansion time variable name in EVNTAB.
            ii.  #EV:=#EV + 1;
    b.  If a model statement then
            i.          If label field contains a sequencing symbol
                        then  If symbol is present in SSNTAB then
                           q:=entry number in SSNTAB;
                        else
                           Enter symbol in SSNTAB[SSNTAB_ptr].
                            q:=SSNTAB_ptr;
                            SSNTAB_ptr:=SSNTAB_ptr + 1;
                            SSTAB[SSTP + q -1] := MDT_ptr;
            ii.  For a parameter, generate the specification (P,#n)
            iii.  For an expansion variable, generate the specification
                                                                  (E,#m).
            iv.  Record the IC in MDT[MDT_ptr];
            v.  MDT_ptr:=MDT_ptr + 1;
```

```
c. If a preprocessor statement then
   i. If a SET statement
         Search each expansion time variable name used in the statement in EVNTAB and generate the spec (E,#m).
   ii. If an AIF or AGO statement then
         If sequencing symbol used in the statement is present in SSNTAB
         Then
               q:=entry number in SSNTAB;
         else
               Enter symbol in SSNTAB[SSNTAB_ptr].
               q:=SSNTAB_ptr;
               SSNTAB_ptr:=SSNTAB_ptr+1;
         Replace the symbol by (S,SSTP + q -1).
   iii. Record the IC in MDT[MDT_ptr]
   iv. MDT_ptr:=MDT_ptr+1;
```

```
4.  (MEND statement)
    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;
```