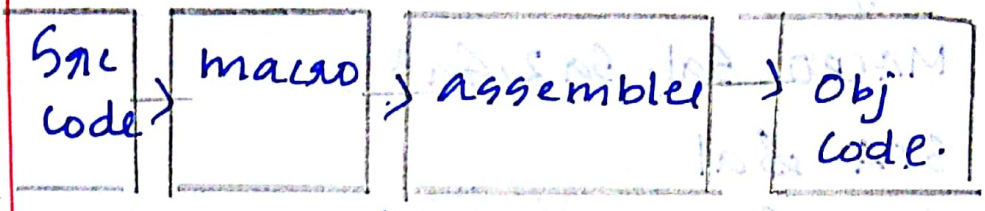
7/11/2017

5. Macroprocessors

Macroprocessors replace each macro call with corresponding sequence of statements



Assembler directives

MACRO
MEND

LDA RETADR
STR G
JEQ

```
STRG  MACRO      → macro prototype
      STA DATA1
      STB DATA2   } macro body
      STX DATA3
      MEND
```

→ Difference b/w macro and subroutine ~~is~~

~~this~~ ~~is~~ ~~DAGEV~~

macro → statements are replaced

Subroutine → call and return.

Stmts of Expansion in a macro are generated each time the macro is invoked

21

whereas the statement in the subroutine appears only once.

macroprototype → it is the definition macro. It include parameter.

→ Macro using parameter.

```
STRG  MACRO  &a1, &a2, &a3
      STA    &a1
      STB    &a2
      STX    &a3
      MEND.
```

eg: Sqrt (16) → argument.

and" Sqrt (n) → parameter.

→ Macro calling – macro call / invocation.

```
STRG  BUFFER  LENGTH  RETADR
```

→ Replacing

```
STA   BUFFER
STB   LENGTH
STX   RETADR
```

```
COPY   START  0
RDBUFF MACRO  &INDEV, &BUFADR
       +LDT   4096
       TD  =  X'&INDEV
       STCH   &BUFADR,X
       MEND
FIRST  STL    RETADR
CLOOP  (RDBUFF F1, BUFFER)
LDA    LENGTH
J      LOOP
RETADR RESW   1
LENGTH RESW   1
BUFFER RESB   4096
       END    FIRST.
```

```
COPY   START  0.
FIRST  STL    RETADR
CLOOP  RDBUFF F1, BUFFER
       +LDT   4096
       TD  =  X'F1'
       STCH   BUFFER,X
       LDA    LENGTH.
       J CLOOP . . . .
       - - - .
       END    FIRST.
```

8/11/17

Data structure of Macroprocessor

NAMTAB – macroname, pointers to start & end of deftab
DEFTAB definition of macro pgm
ARGTAM variables eg: INDEX, BUFADR

NAMTAB is used to store macronames it serves as an index to definition table DEFTAB it contains pointers at the begining and ending of definition in DEFTAB

DEFTAB: It store macrodefinitions ie the macro prototype and macrobody

ARGTAB: It stores arguments according to the position

Singlepass Macroprocessor

Every macro must be defined before it is called. Nested macrodefinitions are allowed in single pass macroprocessor but nested calls are not allowed.

ALGORITHM

{macroprocessor}
GETLINE
PROCESSLINE
Proc {PROCESSLINE}
Search NAMTAB for macro
Proc {DEFINE}
Enter macroname in NAMTAB
prototype in DEFTAB
Proc {EXPAND}
get macrodefn from DEFTAB
& arg ARGTAB
GETLINE
PROCESSLINE
} expand
Proc {GETLINE}
get macrodefn DEFTAB
Substitute arg ARGTAB

Machine Independent Macroprocessor features.

1. Concatenation
2. Unique labels
3. Conditional macroexp.
4. keyword parameters.

eg: for concatenation

```
TD  X'F1
TD  X'F2
TD  X'F3

SUM  MACRO  &ID        ← special concatenation character
     LDA  X&ID  →1
     ADD  X&ID  →2
     ADD  X&ID  →3
     MEND

SUM A
     LADA   XA1
     ADD    XA2
     ADD    XA3.
```

→ Concatination of macro indepenpdant feature.

Some MP allow parameters to be concatenated with other character strings for eg:

Specifies that the parameter and ID is concatenated after X and before the character string 1. The arrow mark clearly identified the end of the parameter and ID and is called as special Concatenation operator. The MP deletes all occurances of concatenation operator immediately after performing parameter substitution.

9/11/2017

Generation of Unique labels

It is not possible for the body of macroinstruction to contain labels because each label would be defined for each invocation of the macro. This duplicate definition would prevent correct assembly of the resulting expanded pgm. ∴ Macro processors allow creation of special type labels with in macro instruction. label begin with $ symbol. In the resulting macro expansion each symbol beginning with $ by replacing with $xx where xx is a two character alphanumeric count of the no: of macro instruction Expanded.

```
ROBUFF  MACRO  &INDEV  Param
   $LOOP  TD  = X'&INDEV
          JEQ   LOOP
```

```
        MEND    LOOP TD = X'F1'
RDBUFF (F1)     JEQ LOOP
RDBUFF (F2)     LOOP TD = X'F2'
                JEQ LOOP.
```

Arguments

This loop don't know which loop is to be used to avoid that we put $

1296 macro's can be expanded in a pgm

```
RDBUFF  MACRO  &INDEV, &BUFFERADDR
```

```
RDBUFF F1, 4096
RDBUFF INDEV = F1, BUFADR = 4096
RDBUFF BUFADR = 4096, INDEV = F1
```

Condixional Macro expansion:

```
RDBUFF  MACRO &INDEV, &&BUFFADR, &EOR
        CLEAR A   CLEAR
        IF (&EOR NE '1')
        &EORCK SET 1
        ENDIF
        IF (&EORCK EQ 1)
$LOOP   LDCH = X'&EOR'
        RMO  A,S
        ENDIF.
```

```
RDBUFF F3, BUF, 04
```

INDEV = F3
BUF = BUFADR
EOR = 04

```
        CLEAR A
        CLEAR S
        LDCH = X'04'
$AALOOP.
```

$ID → 1

Keyword parameter

If position is unknow using parameter we can find the position

MP DESIGN OPTIONS

1. Recursive macroexpansion Incsked
2. General purpose MP
3. Macroprocessing with translator — line by line / Integrated.

Recursive macro

Recursion: a function calling itself

DEFTAB requires the first

General

Every MP can use this language.

details design is expensive.

3. Macroprocessing within translators.

eg: for translators assemblers, compilers, interpretors.

line by line - line interpretor

only one pass is required

Integrated MP.

- DO 100 I = 1,20 (while loop)

  This is a DO stmt.

- DO100I = 1 (assignment stmt)