Programación de Sistemas

Unidad 6. Macro Procesador

Contenido

- Introducción
- Algoritmos y Estructuras
- Características Independientes de la Máquina
- Opciones de Diseño

Introducción

Introducción

- Una macro instrucción, en ocasiones abreviada como macro, es una notación utilizada en la programación
- Un macro representa un grupo de sentencias usadas comúnmente en el código fuente
- El uso de macros permite al programador escribir versiones mas cortas de los programas

Funciones Básicas

- Las funciones de un macro procesador involucran la sustitución de un grupo de caracteres o líneas por otras
- El macro procesador reemplaza cada macro instrucción con el grupo de sentencias correspondiente
- Esta acción es llamada expansión del macro
- A excepción de algunos casos, el macro procesador no realiza análisis alguno del texto que maneja

Definición de Macros

- Para la definición de los macros se tienen dos nuevas directivas
- MACRO
  - Identifica el principio de una definición de macro. El símbolo en el campo etiqueta es el nombre del macro, y las entradas en el campo de operando identifican los parámetros de la macro instrucción
- MEND
  - Marca el final de la definición del macro

Elementos de un Macro

- Nombre
  - El símbolo en el campo etiqueta es el nombre del macro
- Parámetros
  - Son las entradas en el campo de operando, cada parámetro comienza con el símbolo (&)
- Prototipo
  - El nombre y los parámetros del macro definen el prototipo del macro para las instrucciones usadas por el programador
- Cuerpo
  - Son las declaraciones instrucciones que se encuentran entre la definición y el final del macro

Llamada a un Macro

- Una invocación del macro se realiza proporcionando el nombre de la macro instrucción que se desea y los parámetros a ser utilizados para expandir el macro

Algoritmos y Estructuras

DEFTAB

- Es una tabla de definición
- Contiene el prototipo del macro
- Contiene el cuerpo del macro
- Las referencias son colocadas en notación posicional utilizando ?n

Contenido DEFTAB

```
RDBUFF      &INDEV&BUFADR&RECLTH
CLEAR       X
CLEAR       A
CLEAR       S
+LDT        #4096
TD          =X'?1'
JEQ         *-3
RD          =X'?1'
COMPR       A,S
JEQ         *+11
STCH        ?2,X
TIXR        T
JLT         *-19
STX         ?3
MEND
```

NAMTAB

- Es una tabla de nombres
- Contiene los nombres de los macros utilizados
- Contiene apuntadores al comienzo y final de la definición del macro en DEFTAB

ARGTAB

- Es una tabla de argumentos
- Es utilizada durante la expansión de las invocaciones del macro
- Cuando una invocación al macro es reconocida, el argumento es almacenado en ARGTAB de acuerdo a su posición en la lista de argumentos
- Los argumentos de ARGTAB son sustituidos por su correspondiente parámetro en el cuerpo del macro

Macro Procesador de un Paso

1 COMIENZA macro procesador

2 EXPANSION = VERDADERO

3 MIENTRAS OPCODE != ‘MEND'

    3. 1 OBTENER

    3. 2 PROCESAR

Función Procesar

1 Buscar OPCODE en NAMTAB

2 SI encontrado

    2.1 EXPANDIR

3 OTRO SI OPCODE = 'MACRO'

    3.1 DEFINIR

OTRO

    4.1 Escribir la línea al archivo expandido

Función Definir

1 Introducir nombre del macro en NAMTAB

2 Introducir prototipo del macro en DEFTAB

3 NIVEL = 1

4 MIENTRAS NIVEL > 0

4. 1 OBTENER

4.2 SI línea no es un comentario

4.2.1 Sustituir notación de posición con parámetro

4.2.2 Introducir línea en DEFTAB

4.2.3 SI OPCODE = 'MACRO' ENTONCES NIVEL = NIVEL + 1

4.2.4 OTRO SI OPCODE = 'MEND' ENTONCES NIVEL = NIVEL -1

5 Almacenar en NAMTAB apuntadores al comienzo y final de la definición

Función Expandir

1 EXPANDIR = VERDADERO

2 Obtener la primer línea del prototipo del macro de DEFTAB

3 Colocar argumentos de la llamada al macro en ARGTAB

4 Escribir la invocación del macro como un comentario

5 MIENTRAS no sea el final de la definición del macro

    5.1 OBTENER

    5.2 PROCESAR

6 EXPANDIR = FALSO

Función Obtener

1 SI EXPANDIR

1.1 Obtener siguiente línea de la definición del macro de DEFTAB

1.2 Sustituir argumentos de ARGTAB por notación posicional

2 OTRO

2. 1 Leer siguiente línea del archivo de entrada

Características Independientes de la Máquina

Concatenación de Parámetros

- La mayora de los macro procesadores permiten la concatenación de parámetros con otras cadenas de caracteres
- La mayora de los macro procesadores proveen un operador de concatenación especial, como puede ser el símbolo &
- El macro procesador elimina todas las ocurrencias del operador de concatenación inmediatamente después de realizar la sustitución de parámetros, de tal forma que el carácter de concatenación no aparecerá en la expansión del macro

Ejemplo

- Se tienen las siguientes instrucciones:

```
SUM     MACRO   &ID
        LDA     X&ID?1
        ADD     X&ID?2
        ADD     X&ID?3
        STA     X&ID?S
        MEND
```

Invocación

```
SUM     A
LDA     XA1
ADD     XA2
ADD     XA3
STA     XAS
MEND
```

Generación de Etiquetas Únicas

- En general no es posible para el cuerpo de una macro instrucción contener etiquetas del tipo usual
- Si una etiqueta se coloca en una instrucción, esta etiqueta será definida dos veces, una para cada invocación
- Muchos macro procesadores evitan este problema permitiendo la creación de tipos especiales de etiquetas dentro de las macro instrucciones
- Para esto se utiliza el símbolo $ dentro de la etiqueta, de esta manera cada símbolo que comience con $, será reemplazado con *$xx* en donde *xx* es un contador alfanumérico de dos caracteres del numero de instrucciones ensambladas

Ejemplo

- Primera Expansión

```
                CLEAR       X
                CLEAR       A
                +LDT        #4096
$AALOOP         TD          =X'F1'
                JEQ         $AALOOP
                RD          =X'F1'
                COMPR       A,S
                JEQ         &AAEXIT
                TIXR        T
                JLT         &AALOOP
$AAEXIT         STX         LENGTH
```

Ejemplo

- Segunda Expansión

```
                CLEAR       X
                CLEAR       A
                +LDT        #4096
$ABLOOP         TD          =X'F1'
                JEQ         $ABLOOP
                RD          =X'F1'
                COMPR       A,S
                JEQ         &ABEXIT
                TIXR        T
                JLT         &ABLOOP
$ABEXIT         STX         LENGTH
```

Expansión Condicional de Macros

- La mayora de los macro procesadores pueden modificar la secuencia de instrucciones generada por la expansión de un macro dependiendo de los argumentos proporcionados al momento de la invocación
- Las macro instrucciones para realizar esta función incluyen la asignación (SET), decisión (IF, ELSE, ENDIF) y ciclos (WHILE, ENDW)

Directiva de Asignación SET

- Asigna un cierto valor a un símbolo de asignación que puede ser utilizado para almacenar variables durante una expansión
- Cualquier símbolo que comience con & y que no sea un parámetro del macro, se considera como un símbolo de asignación
- Ejemplo

```
&EORCK      SET     1
```

Directivas de Decisión

- Cuando se encuentra una sentencia IF durante la expansión del macro, se evalúa la correspondiente expresión Booleana
- Si el valor de esta expresión es VERDADERO, el macro procesador continua con el procesamiento de las líneas de DEFAB hasta que encuentra un ELSE o un ENDIF
- Si se encuentra un ELSE, el macro procesador salta las líneas de DEFTAB hasta que encuentra un ENDIF
- Si el valor de la expresión Booleana es FALSO, el macro procesador salta las instrucciones en DEFTAB hasta que encuentra un ELSE o un ENDIF

Uso de Directivas de Decisión

```
RDBUFF      MACRO   &INDEV,&BUFADR,&RECLTH,&EOR,&MAXLTH
            IF      (&EOR NE ' ')
&EORCK      SET     1
            ENDIF
            CLEAR   X
            CLEAR   A
            IF      (&EORCK  EQ 1)
            LDCH    =X'&EOR'
            RMO     A,S
            ELSE
            +LDT    #4096
            ENDIF
```

Directivas de Ciclos

- La sentencia WHILE especifica que las siguientes líneas, hasta que se encuentre un ENDW serán generadas repetidamente mientras cierta condición particular se cumpla
- La prueba a la condición es realizada y el ciclo son realizados mientras el macro esta siendo expandido Software de

Ejemplo

```
&CTR        SET         1
            WHILE       (&CTR LT &EORCT)
            COMP        =X'0000&EOR[&CTR]'
            JEQ         $EXIT
&CTR        SET         CTR+1
            ENDW
```

Parámetros con Palabras Clave

- Hasta el momento todas las definiciones de macro instrucciones se han revisado utilizando parámetros posicionales
- Si un macro tiene una gran cantidad de parámetros y solo unos pocos de estos se proporcionan en una invocación, es mas conveniente utilizar un método diferente

Notación Posicional

- Suponer que una cierta instrucciones tiene 6 posibles parámetros, pero en una invocación particular, solo el tercero y el sexto serán especificados
- Si se utilizaran los parámetros posicionales, la invocación será:

INSTRUCCION , ,DIRECT, , ,3

Notación con Palabra Clave

- Cada valor del argumento es escrito con una palabra clave que denota al parámetro correspondiente
- Los argumentos pueden aparecer en cualquier orden, si el tercer parámetro en el ejemplo previo es denotado como &TYPE y el sexto es llamado &CHANNEL, la macro instrucción sería:

```
NOMBRE      MACRO   &NOM1,&NOM2,&TYPE,&NOM3,&NOM4,&CHANNEL
INSTRUCCION         TYPE=DIRECT,CHANNEL=3
```

Opciones de Diseño

Expansión Recursiva

- El algoritmo revisado hasta el momento no es eficiente cuando se realiza una invocación a un macro dentro de otro macro

Ejemplo

```
RDBUFF    MACRO    &BUFADR&RECLTH&INDEV
          CLEAR    X
          CLEAR    A
          CLEAR    S
          +LDT     #4096
$LOOP     RDCHAR   &INDEV
          COMPR    A,S
          JEQ      $EXIT
          STCH     &BUFADR,X
          TIXR     T
          JLT      $LOOP
$EXIT     STX      &RECLTH
          MEND
```

Ejemplo

- Macro RDCHAR

```
RDCHAR      MACRO     &IN
            TD        =X'&IN'
            JEQ       *-3
            RD        =X'&IN'
            MEND
```

Ejemplo

- Mientras se realiza la expansión de RDBUFF se encuentra la instrucción

  $LOOP RDCHAR &INDEV

Procedimiento de Expansión

- El procedimiento EXPANDIR es llamado cuando una llamada a macro es reconocida, los argumentos de la invocación son introducidos en la ARGTAB
- La variable booleana EXPANDIR se coloca en VERDADERO y comienza, la expansión del macro
- El procedimiento posteriormente encuentra la invocación al macro RDCHAR
- En ese punto, se vuelve a llamar al procedimiento para expandir el nuevo macro
- La expansión de RDCHAR procede con normalidad, sin embargo, al finalizar esta, aparece un problema

Problema

- Cuando se termina la expansión de RDCHAR, la variable EXPANDIR es colocada en FALSO, por lo tanto el macro procesador "olvida" que estaba a la mitad de la expansión del macro original (RDBUFF)
- Además los valores de ARGTAB fueron sobre-escritos por los argumentos de la llamada a RDCHAR

Posibles Soluciones

- Este problema puede resolverse de manera sencilla escribiendo el macro procesador con un lenguaje de programación que permita llamadas recursivas. De esta manera el compilador se asegura que los valores de las variables han sido almacenados antes de que se hagan otras llamadas recursivas
- Si no se tiene un lenguaje que soporte recursividad, el programador debe manejar estas acciones mediante el regreso de direcciones y el manejo de los valores de las variables locales

Macro Procesadores de Propósito General

- El uso mas común de un macro procesador es como auxiliar de un lenguaje ensamblador
- Estos macro procesadores de propósito especial tienen un funcionamiento similar, sin embargo los detalles dieren de lenguaje en lenguaje
- Existe otro tipo de macro procesadores llamados macro procesadores de propósito general que no depende de las características de ningún lenguaje de programación por lo que pueden ser utilizados en cualquiera de estos

Ventajas

- El programador no necesita aprender acerca del macro ensamblado de cada compilador o lenguaje ensamblador
- Aunque el costo de desarrollar un macro procesador de propósito general es mas elevado que el de uno específico, solo debe realizarse una vez

Limitantes para el Desarrollo

- A pesar de estas ventajas, existen pocos macro procesadores de propósito general
- Una de estas razones es la gran cantidad de detalles con los que se debe lidiar en un lenguaje de programación real, como son:
  - El manejo de comentarios
  - La manera en que se agrupan términos, expresiones o declaraciones
  - Las cadenas que sirven como identificadores, constantes, operadores y palabras reservadas