2017

Глава 5 *Макросредства ассемблера*

МГТУ им. Н.Э. Баумана
Факультет Информатика и системы управления
Кафедра Компьютерные системы и сети
Лектор: д.т.н., проф.
Иванова Галина Сергеевна

Основные проблемы ассемблера

- плохое понимание исходного текста программы, особенно по прошествии некоторого времени после ее написания;
- ограниченность набора команд;
- повторяемость идентичных или незначительно отличающихся участков программы;
- необходимость включения в каждую программу участков кода, которые уже были использованы в других программах.

Использование макросредств позволяет:

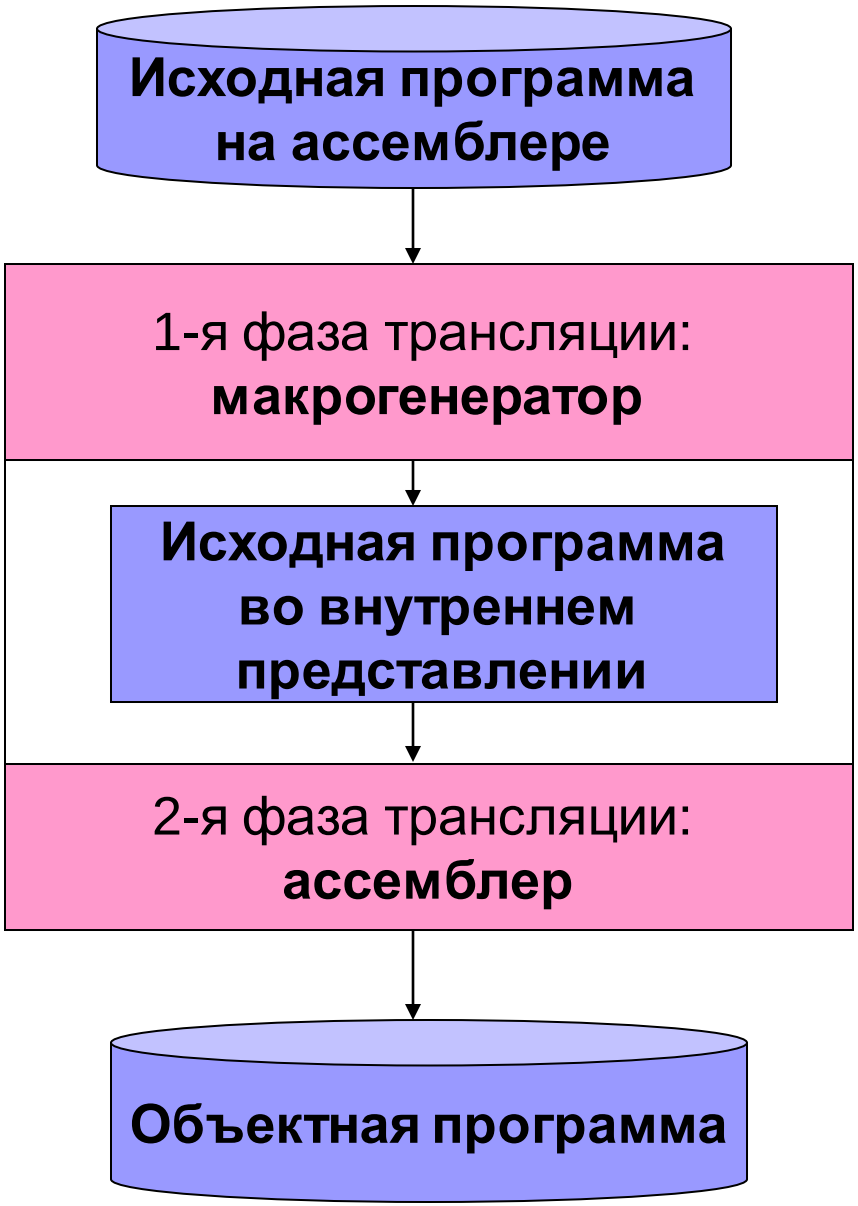
1) динамически перестраивать программу (простой полиморфизм);

2) увеличить скорость выполнения по сравнению с вызовом процедур;

3) создавать макробиблиотеки;

4) улучшать наглядность программ.

8.1 Трансляция программы на ассемблере



Трансляция программы на ассемблере выполняется в два этапа:

- макрогенерация;
- ассемблирование.

Во время макрогенерации программа обрабатывается как текстовая информация, т.е. выполняются текстовые замены в том числе условные.

8.2 Термы и выражения

Операндом макросредств является строка символов (терм).

Термы делятся на:

- ***самоопределенные***, например литералы:
  - целочисленные числовые: 23h, 10, 10001b;
  - строковые: “ABCD”;
- ***перемещаемые***, например метки операторов и данных, $ (текущее значение счетчика адреса при ассемблировании).

В программах термы представляются следующим образом:

- последовательность символов, состоящих из букв, цифр, а также символов «$» и «_» – без ограничителей, например: **AV12, 01Fh**;
- последовательность символов, включающая специальные:

  а) в угловых скобках, например: **<Строка&>**

  б) с использованием **«!»** перед каждым специальным символом, например: **Строка!&**.

Макровыражения

Макровыражения – совокупность термов, связанных символами арифметических, логических операций, а также скобками и символами операций сравнения.

Арифметические операции: **+**, **-**, *****, **/**, **MOD**, **SHR**, **SHL**.

Логические операции (0 – ложь, 0FFFFh – истина):

AND, **OR**, **XOR**, **NOT**.

Операции сравнения: **EQ**, **NE**, **LT**, **GT**, **LE**, **GE**.

Пример:

```
    mov AX,((C LT 20) AND 5) OR ((C GE 20) AND 6)
C<20:  mov AX,(0FFFFh AND 5) OR (0 AND 6)
       mov AX,5
C>20:  mov AX, (0 AND 5) OR (0FFFFh AND 6)
       mov AX,6
```

Псевдоконстанты и переменные

Псевдоконстанты:

<Имя> EQU <Выражение>

Примеры:

а) `K       EQU  1024`

б) `Count   EQU  CX`

в) `M       EQU  60*20/K+Cnt`

Псевдопеременные:

<Имя> = <Выражение>

Примеры:

а) `CONST = 1024`

б) `CONST = CONST+1`

в)
```
adr1  db  5 dup (0)
adr2  dw  0
len = adr2-adr1
```

8.3 Макрокоманды и макроопределения

Макрокоманда (макровызов) – команда, по которой происходит вставка фрагмента в программу.

Макроопределение – заготовка, из которой генерируется вставляемый фрагмент.

Макрорасширение – вставляемый фрагмент.

Макрогенерация – процесс построения макрорасширения из макроопределения.

Макроопределение

Структура макроопределения:

<Имя> MACRO [<Список формальных параметров>]
<Тело макроопределения>
<Имя> ENDM

Пример:

```
ADD_W   MACRO   T1,T2,SUM
        mov     AX,T1
        add     AX,T2
        mov     SUM,AX
        ENDM
```

Макровызовы:

```
ADD_W   PRICE,TAX,COST
ADD_W   BX,CX,DX
```

Макрорасширение 1:

```
        mov     AX,PRICE
        add     AX,TAX
        mov     COST,AX
```

Макрорасширение 2:

```
        mov     AX,BX
        add     AX,CX
        mov     DX,AX
```

8.4 Операции макроопределений

1. Слияние символьных строк: **<Строка>&<Строка>**

2. Комментарий: **;;**

Пример:

```
MOVE    MACRO    TAG
;; генерация команды
        REP     MOVS&TAG
        ENDM
```

Если TAG = B, то генерируется:

```
REP   MOVSB
```

3. Преобразование имени в число: **%<Имя>**

Пример:

```
VALUE    EQU   0
MSG      MACRO   PAR
          DB     'MSG&PAR'
         ENDM
Code   segment
       assume    CS:Code
       MSG    %VALUE
       ...
```

```
DB   'MSG0'
```

Размещение макроопределений

Для отладки – перед программой, для работы – в библиотеки.

Пример 8.1 Определение суммы чисел

```
Summa   MACRO   T1,T2,Sum
        mov     AX,T1
        add     AX,T2
        mov     Sum,AX
        ENDM
        .DATA
A       dw      5
B       dw      6
D       dw      ?
        .CODE
Start:
        Summa   A,B,D
```

⇒

```
mov     AX,A
add     AX,B
mov     D,AX
```

Библиотека макроопределений

Отлаженные макроопределения помещают в библиотеки.

Библиотека макроопределений строится как обычный файл и подключается как:

```
IF1
        INCLUDE  MACRO.LIB
ENDIF
```

Удаление неиспользуемых макроопределений из программы во время трансляции:

```
PURGE <Список имен макросов>
```

Пример:

```
        PURGE   MAC1,MAC2,MAC3
```

8.5 Псевдооператоры макроассемблера

1. Изменение меток при каждом расширении:

```
LOCAL <Список меток>
```

Пример 8.2

```
wait        macro   count
            local   next
            push    ECX
            mov     ECX,count
next:       loop    next
            pop     ECX
            endm
```

В результате будут генерироваться метки:

`??0000, ??0001` и т.д.

```
        push    ECX
        mov     ECX,count
??0000: loop ??0000
        pop     ECX
```

```
        push    ECX
        mov     ECX,count
??0001: loop ??0001
        pop     ECX
```

Псевдооператоры макроассемблера (2)

2. Повторять с каждым аргументом списка:

IRP <Параметр>,<Список аргументов>

<Фрагмент>

ENDM

Пример 8.3

```
Irp     value,<1,2,3,4,5>
dw      value*value*value
endm
```

```
dw   1
dw   8
dw   27
dw   64
dw   125
```

Псевдооператоры макроассемблера (3)

3. Повторять с каждым символом строки:

IRPC <Параметр> <Строка>

<Фрагмент>

ENDM

Пример 8.4:

```
Irpc    char,12345
db      char
endm
```

```
db   1
db   2
db   3
db   4
db   5
```

Пример 8.5:

```
Irpc    char,ABCD
mov    char&X,1
endm
```

```
mov    AX,1
mov    BX,1
mov    CX,1
mov    DX,1
```

Псевдооператоры макроассемблера (4)

4. Повторять указанное число раз:

```
REPT <Выражение>
<Фрагмент>
ENDM
```

Пример 8.6

```
.DATA
N =  0
REPT 5
N =  N + 1
DB   N
ENDM

.CODE
REPT  3
inc  AX
endm
```

```
DB    1
DB    2
DB    3
DB    4
DB    5
```

```
inc   AX
inc   AX
inc   AX
```

8.6 Условное ассемблирование

1. Завершение макрогенерации: **EXITM**

2. Если …, то …

```
IFXX  <Выражение>
…
[ELSE]
…
ENDIF
```

Виды условий условной генерации

IF <Выражение> - выражение ≠ 0;

IFE <Выражение> - выражение = 0;

IFDEF <Имя> - имя объявлено как внешнее;

IFNDEF <Имя> - имя не объявлено как внешнее;

IFB <Параметр> - параметр опущен;

IFNB <Параметр> - параметр задан;

IFIDN <Строка1>,<Строка2> - строки совпадают;

IFDIF <Строка1>,<Строка2> - строки не совпадают;

IF1 - первый проход ассемблера – используется для чтения библиотек макроопределений;

IF2 - второй проход ассемблера.

Пример 8.7 Условная макрогенерация

```
All     macro  len
        value=0
        if len ge 50
           exitm
        endif
         rept len
          value=value+1
          DB   value
         endm
        endm
```

Если len=3, то

```
DB  1
DB  2
DB  3
```

Если len=50, то макрогенерация не выполняется

Пример 8.8 Macro «Деление целых»

```
DIVIDE  MACRO   DIVIDEND,DIVISOR,QUOTIENT
    LOCAL   COMP ;; метки
    LOCAL   OUT                          MOV   AX,DIVIDEND
    CNTR    = 0                          MOV   BX,DIVISOR
    IFNDEF  DIVIDEND                     SUB   CX,CX
      CNTR    = CNTR +1          COMP:   CMP   AX,BX
    ENDIF                                JB    OUT
    IFNDEF  DIVISOR                      SUB   AX,BX
      CNTR    = CNTR +1                  INC   CX
    ENDIF                                JMP   COMP
    IFNDEF  QUOTIENT             OUT:    MOV   QUOTIENT,CX
      CNTR    = CNTR +1                  ENDM
    ENDIF
    IF      CNTR ;; не равно 0
      EXITM
    ENDIF
```

Тестирующая программа

```
            .DATA
A           dw      20
B           dw      10
D           dw      ?

            .CODE
Start:
            Divide A,B,D
            . . .
```

⇒

```
            MOV   AX,A
            MOV   BX,B
            SUB   CX,CX
??0000:     CMP   AX,BX
            JB    ??0001
            SUB   AX,BX
            INC   CX
            JMP   ??0000
??0001:     MOV   D,CX
```

Пример 8.9 Макрос ввода строки

Макрос выводит запрос, вводит данные и возвращает их адрес в EAX

```
input MACRO prompt:VARARG  ; prompt – текст
      LOCAL txt
      LOCAL buffer
      IFNB <prompt>    ; если текст задан
        .data
          txt db prompt, 0 ; место для текста
          buffer db 128 dup (0) ; буфер ввода
          align 4        ; выровнять на границу дв. слова
        .code
        invoke StdOut,ADDR txt
        invoke StdIn,ADDR buffer,LENGTHOF buffer
        invoke StripLF,ADDR buffer
        mov eax, offset buffer
       EXITM <eax>   ; возвращаемое значение !
```

Макрос ввода строки (2)

```
    ELSE ; если запрос отсутствует
      .data
        buffer db 128 dup (0)
        align 4
      .code
      invoke StdIn,ADDR buffer,LENGTHOF buffer
      invoke StripLF,ADDR buffer
      mov eax, offset buffer
      EXITM <eax>
    ENDIF
ENDM
```

Тестирующая программа

```
            .CONST
EOLn        DB      13,10,0

            .CODE
Start:
            mov EBX,input ("Input Data",13,10)
            Invoke StdOut,EBX
            Invoke StdOut,ADDR EOLn
            . . .
```

Примечания:

- Список параметров указан в скобках, поскольку в макроопределении указано VARARG.
- Вызов макрокоманды указан в машинной команде, поскольку в макроопределении указано возвращаемое значение.