Лабораторная работа № 14 (6)

Тема: *Разработка программ с созданием пользовательских процедур на языке Ассемблер.*
Цель: *Освоить разработку процедур на языке Ассемблер, передачу параметров, возврат значений, использование локальных переменных.*

Краткая теория

Как в любом другом языке программирования в Ассемблере присутствует возможность декомпозиции программного исполняемого кода на отдельные модули – процедуры (или, как их еще называют, подпрограммы и функции). Описание процедур осуществляется в сегменте (или сегментах) кода.

Описание процедуры имеет следующий синтаксис:

```
Имя_процедуры PROC [[модификатор языка] язык] [расстояние]
[ARG список_передаваемых_параметров]
[RETURNS список_возвращаемых_значений]
[LOCAL список_локальных_объявлений]
[USES список_используемых_регистров]
          …
          Команды и директивы
          языка Ассемблер
          …
          RET
[Имя_процедуры] ENDP
```

Единственным обязательным значением, которое необходимо указывать при описании процедуры является ее имя.

Так как программа может занимать более одного сегмента кода, то процедуры вызывающие друг друга могут находится в различных сегментах:

- Если вызываемая процедура находится в текущем сегменте, то используется ближний вызов команды CALL.
- Если же вызываемая процедура находится в другом сегменте, то используется дальний вызов команды CALL.

Управление характером возможного вызова процедуры осуществляется с помощью необязательного параметра в описании [расстояние]. Этот параметр может принимать одно из двух значении:

- near – ближний вызов (используется по умолчанию),
- far – дальний вызов.

Описание процедур на языке Ассемблер может располагаться, в принципе, в любом месте программы. Таким образом, возможны три варианта описания процедуры:

- в начале сегмента кода,
- в конце сегмента кода,
- в середине сегмента кода.

В начале сегмента кода

```
Code SEGMENT use16
          ASSUME cs: Code
MyProc    PROC
          …
          ret
MyProc    ENDP
```

В конце сегмента кода

```
Code SEGMENT use16
          ASSUME cs: Code
start:
          …
          call MyProc
          …
```

```
start:
        …
        call MyProc
        …
Code ENDS
end start
```

```
        mov  ax, 4c00h
        int  21h
MyProc  PROC
        …
        ret
MyProc  ENDP
Code ENDS
end start
```

В середине сегмента кода

```
Code SEGMENT use16
        ASSUME cs: Code
start:
        …
        call MyProc
        …
        jmp  next
MyProc  PROC
        …
        ret
MyProc  ENDP
next:   …
        mov  ax, 4c00h
        int  21h
Code ENDS
end start
```

В середине сегмента кода

```
Code SEGMENT use16
        ASSUME cs: Code
start:
        …
        jmp  next
MyProc  PROC
        …
        ret
MyProc  ENDP
next:   …
        call MyProc
        …
        mov  ax, 4c00h
        int  21h
Code ENDS
end start
```

Описание процедур в различных сегментах кода:

```
Code2 SEGMENT use16
      ASSUME cs: Code2
MyFarProc PROC far
      …
      call MyNearProc
      …
      ret
MyFarProc ENDP
MyNearProc PROC near
      …
      ret
MyNearProc ENDP
Code2 ENDS
Code1 SEGMENT use16
      ASSUME cs: Code1
start:
      …
      call far MyFarProc
      …
      mov  ax, 4c00h
      int  21h
Code1 ENDS
end start
```

Передача параметров в процедуру может осуществляться следующими способами:

- через регистры,
- через общую область памяти,
- через стек.

Это наиболее простой способ передачи данных. Данные, переданные таким способом, становятся доступными немедленно после передачи управления процедуре. Этот способ очень популярен при небольшом объеме передаваемых данных.

Ограничения этого способа передачи:

- небольшое число доступных для пользователя регистров;
- необходимо помнить, какая информация в каком регистре находится;
- ограничение размера передаваемых данных размерами регистра.

Рассмотрим программу, в которой осуществляется подсчет длин двух строк, заканчивающихся символом ‘$’, с помощью реализованной процедуры StrLen. В процедуру StrLen в регистре SI передается адрес строки. Процедура через регистр AL возвращает длину строки. Вывод длины строки осуществляется путем прямого доступа к видео памяти и также реализован через процедуру. Исходная строка жестко кодируется в тексте программы.

```
.486
model small
Data SEGMENT use16
      ASSUME ds:Data
str1  db    'It is first string for test!$'
str2  db    'It is second string!$'
Data ENDS
Stk  SEGMENT use16 stack
      db    256 dup(0)
Stk  ENDS
Code SEGMENT use16
      ASSUME cs: Code
;В регистре SI – адрес строки,
;результат в AX
StrLen   PROC
      xor         ah, ah
next:
      lodsb
      cmp         al, '$'
      jz          finprc
      inc         ah
      jmp         next
finprc:
      shr         ax, 8
      ret
StrLen   ENDP
;В регистре AL – номер строки,
;в регистре DX – количество символов
PrintLen PROC
      mov         cx, 0b800h
      mov         es, cx
      mov         cl, 160
      mul         cl
      mov         di, ax
      mov         ax, dx
      mov         cl, 10
      div         cl
      mov         cx, ax
      mov         ah, 15
      add         al, 48
      mov         es:[di],ax
      mov         al, ch
      add         al, 48
```

```
      mov         es:[di+2], ax
      ret
PrintLen ENDP
start:
      mov         ax, Data
      mov         ds, ax
      mov         si, offset str1
      call        StrLen            ;Вызов StrLen для строки str1
      mov         dx, ax
      mov         al, 1
      call        PrintLen          ;Вызов PrintLen для строки str1
      mov         si, offset str2
      call        StrLen            ;Вызов StrLen для строки str2
      mov         dx, ax
      mov         al, 5
      call        PrintLen          ;Вызов PrintLen для строки str1
wait0:
      in          al, 60h
      cmp         al, 1
      jnz         wait0
      mov         ax, 4c00h
      int         21h
Code ENDS
end start
```

В этом способе предполагается, что вызывающая и вызываемая подпрограммы «знают» где должны находится параметры и результат выполнения операции. Недостатком этого способа является то, что процедуры взаимодействуют через некоторую глобальную область памяти, к которой имею доступ и остальные процедуры программы. Этот способ сложно реализуем при разработке модульных программ. Использовать передачу параметров через общую область на практике не рекомендуется.

Далее приведен листинг предыдущего примера, с использованием именно этого метода передачи значений.

```
.486
model small
Data SEGMENT use16
      ASSUME ds:Data
str1  db    'It is first string for test!$'
str2  db    'It is second string!$'
slen  dw    0
ypos  db    0
sadr  dw    0
Data ENDS
Stk  SEGMENT use16 stack
      db    256 dup(0)
Stk  ENDS
Code SEGMENT use16
      ASSUME cs:Code
StrLen   PROC
      xor         ah,ah
      mov         si, sadr    ;Чтение адреса строки
next:
      lodsb
      cmp         al,'$'
      jz          finprc
      inc         ah
      jmp         next
finprc:
      shr         ax, 8
      mov         slen, ax    ;Запись длины строки
      ret
StrLen   ENDP
PrintLen PROC
```

```
      mov         cx, 0b800h
      mov         es, cx
      mov         al, ypos    ;Чтение позиции вывода
      mov         cl, 160
      mul         cl
      mov         di, ax
      mov         ax, slen    ;Чтение длины строки
      mov         cl, 10
      div         cl
      mov         cx, ax
      mov         ah, 15h
      add         al, 48
      mov         es:[di],ax
      mov         al, ch
      add         al, 48
      mov         es:[di+2], ax
      ret
PrintLen ENDP
start:
      mov         ax, Data
      mov         ds, ax
      mov         ax, offset str1
      mov         sadr, ax          ;Запись параметра – адрес строки str1
      call        StrLen
      mov         ypos, byte ptr 1  ;Запись номера позиции str1
      call        PrintLen
      mov         ax, offset str2
      mov         sadr, ax          ;Запись параметра – адрес строки str2
      call        StrLen
      mov         ypos, byte ptr 5  ;Запись номера позиции str1
      call        PrintLen
wait0:
      in          al, 60h
      cmp         al, 1
      jnz         wait0
      mov         ax, 4c00h
      int         21h
Code ENDS
end start
```

Этот способ наиболее часто используется на практике. Суть этого способа заключается в том, что вызывающая процедура самостоятельно заносит в стек передаваемые данные, после чего производит вызов вызываемой процедуры. Вызываемая процедура читает эти значения из стека. Для работы со стеком используются три регистра:

- ss – сегмент стека,
- (e)bp – регистр базы,
- (e)sp – текущее смещение в стеке.

При таком способе передачи код процедуры несколько изменяется:

```
MyProc     PROC
           push       bp          ;Настройка базы
           mov        bp, sp
           …
           pop        bp          ;Восстановление базы
           ret
MyProc     ENDP
```

В таком случае: bp – адрес стека на момент передачи управления вызываемой процедуре. По этому адресу располагается адрес возврата.

Листинг рассматриваемого ранее примера с передачей параметров через стек:

```
.486
model small

Data SEGMENT use16
      ASSUME ds:Data
str1  db    'It is first string for test!$'
str2  db    'It is second string!$'
Data ENDS

Stk  SEGMENT use16 stack
      db    256 dup(0)
Stk  ENDS
Code SEGMENT use16
      ASSUME cs:Code
StrLen   PROC
      push        bp
      mov         bp, sp
      xor         ah,ah
      mov         si, [bp+4] ;Чтение адреса строки
next:
      lodsb
      cmp         al,'$'
      jz          finprc
      inc         ah
      jmp         next
finprc:
      shr         ax, 8
      pop         bp
      ret         2
StrLen   ENDP
PrintLen PROC
      push        bp
      mov         bp, sp
      mov         cx, 0b800h
      mov         es, cx
      mov         ax, [bp+4]  ;Чтение позиции
      mov         cl, 160
      mul         cl
      mov         di, ax
      mov         ax, [bp+6]  ;Чтение длины строки
      mov         cl, 10
      div         cl
      mov         cx, ax
      mov         ah, 15
      add         al, 48
      mov         es:[di],ax
      mov         al, ch
      add         al, 48
      mov         es:[di+2], ax
      pop         bp
      ret
PrintLen ENDP
start:
      mov         ax, Data
      mov         ax, offset str1
      push        ax          ;Запись в стек адреса строки str1
      call        StrLen
      push        ax          ;Запись в стек длины строки str1
      push        word ptr 1        ;Запись в стек позиции
      call        PrintLen
      add         sp, 4
      mov         ax, offset str2
      push        ax          ;Запись в стек адреса строки str2
      call        StrLen
```

```
      push        ax          ;Запись в стек длины строки str1
      push        word ptr 5        ;Запись в стек позиции
      call        PrintLen
wait0:
      in          al, 60h
      cmp         al, 1
      jnz         wait0
      mov         ax, 4c00h
      int         21h
Code ENDS
end start
```

Для каждой программы при ее запуске создается блок PSP (Program segment prefix – Префикс программного сегмента), который содержит следующую информацию:

Смещение	Размер, байт	Описание
00h	2	INT 20h – можно использовать для выхода
02h	2	Вершина доступной памяти системы в параграфах
04h	1	Резерв
05h	5	FAR CALL к диспетчеру функций DOS
06h		Доступные байты в программном сегменте (только для COM)
0ah	4	Адрес завершения
0eh	4	Адрес обработки Ctrl-Break
12h	4	Обработчик критических ошибок
16h	16h	Резервная область DOS
2ch	2	Сегментный адрес окружения DOS
2eh	2eh	Резервная область DOS
5ch	10h	Форматированная область параметров 1 (как FCB для 1-го пар.)
6ch	14h	Форматированная область параметров 2 (как FCB для 2-го пар.)
80h	1	Длина области неформатированных параметров (UPA)
81h	7fh	Область неформатированных параметров

Структура PSP находится перед самой программой. При загрузке сегментные регистры ES и DS установлены на нее. Для COM-программ область PSP располагается в начале сегмента загруженной программы (поэтому во всех программах COM-формата генерируется смещение на 100h байт с помощью директивы ORG). В процессе выполнения программы адрес PSP можно получить с помощью функции 62h прерывания 21h.

Ход работы

Во всех заданиях этой лабораторной работы ввод исходных данных осуществляется с клавиатуры с помощью функции 0ah прерывания 21h, а вывод данных – с помощью функции 09h прерывания 21h. Часть входных данных программы может передаваться через параметры командной строки.

Задание 1

Разработать программу согласно варианту задания. При реализации всех процедур передачу параметров реализовать через стек. Процедуры возвращают целочисленные значения регистр AX, остальные значения возвращаются через параметр по ссылке (адрес, куда необходимо записать результат, передается в параметре процедуры). Варианты заданий:

1	Реализовать процедуру, осуществляющую вычисление количества букв латинского алфавита в строке. Используя процедуру, определить количество букв в трех строках. Исходные строки вводятся пользователем. Максимальная длина строк не превышает 80 символов. Результат вывести на экран.

2	Реализовать процедуру, выводящую целое положительное число типа word на экран в десятичной системе счисления. Вывести на экран пять чисел, используя разработанную процедуру. Исходные значения жестко кодируются в программе.
3	Реализовать процедуру, возвращающую максимальное из трех целочисленных значений. Используя данную процедуру, определить максимум четырех троек значений. Значения максимума записать в бинарный файл. Исходные данные и имя файла жестко кодируются в программе.
4	Реализовать процедуру, вычисляющую значение выражение по формуле X*4+10. Значение X и результат имеют тип word. Используя разработанную процедуру, вычислить значение для 10 значений, которые жестко кодируются в программе. Результат записать в бинарный файл.
5	Реализовать процедуру, определяющую является ли переданная в параметрах строка целым числом в десятичной системе счисления. Если является, то функция возвращает 1. В противном случае – 0. обработать 5 строк вводимых пользователем. Длина строки не превышает 50 символов. Результат для каждой вывести на экран.
6	Реализовать процедуру, вычисляющую сумму цифр числа (тип word, в десятичной системе) переданного в параметре. Используя процедуру, обработать 5 чисел, значение которых жестко кодируется в программе. Результат вывести на экран.
7	Реализовать процедуру, которая вычисляет среднее арифметическое пяти чисел (тип word), переданных в качестве параметров. Используя разработанную процедуру вычислить среднее значение трех пятерок чисел, значения которых жестко кодируются в программе. Результат записать в бинарный файл, имя которого задано в программе.
8	Реализовать процедуру, вычисляющую количество бит установленных в единицу в целом значении типа dword. Значение передается в параметрах. Используя процедуру обработать 10 чисел, значения которых жестко кодируются в программе. Результат вывести на экран.
9	Реализовать процедуру, которая осуществляет обращение строки (переписывание символов в обратном порядке), переданной в параметре. Используя данную процедуру обработать 5 строк. Ввод строк осуществляет пользователь, вывод результата осуществлять на экран.
10	Реализовать процедуру, определяющую среднее из трех целочисленных значений, переданных в качестве параметров. Используя эту процедуру обработать четыре тройки чисел, значения которых жестко кодируются в программе. Результат записать в бинарный файл.
11	Реализовать процедуру, определяющую является ли переданная в параметрах строка целым числом в двоичной системе счисления. Если является, то функция возвращает 1. В противном случае – 0. обработать 5 строк вводимых пользователем. Длина строки не превышает 50 символов. Результат для каждой вывести на экран.
12	Реализовать процедуру, выводящую целое положительное число типа word на экран в восьмеричной системе счисления. Вывести на экран пять чисел, используя разработанную процедуру. Исходные значения жестко кодируются в программе.
13	Реализовать процедуру, вычисляющую количество символов пунктуации в строке, переданной в параметрах. Используя процедуру, определить количество знаков препинания в трех строках. Исходные строки вводятся пользователем. Максимальная длина строк не превышает 80 символов. Результат вывести на экран.
14	Реализовать процедуру, осуществляющую поиск первой гласной буквы латинского алфавита в строке, переданной в качестве параметра. Используя данную процедуру обработать 5 строк, вводимых пользователем. Результат вывести на экран.
15	Реализовать процедуру, определяющую максимальную цифру (десятичная система) в целочисленном значении типа word, переданном в параметре. Используя данную процедуру обработать 10 чисел, значение которых жестко кодируется в программе. Результат вывести на экран.

16	Реализовать процедуру, вычисляющую сумму цифр числа (тип word, в восьмеричной системе) переданного в параметре. Используя процедуру, обработать 5 чисел, значение которых жестко кодируется в программе. Результат вывести на экран.
17	Реализовать процедуру, осуществляющую ввод целого числа с клавиатуры. Число вводится в виде строки (локальная переменная), которая затем преобразуется в число. Используя процедуру реализовать следующую программу: пользователь вводит с клавиатуры числа, которые затем записываются в бинарный файл.
18	Реализовать процедуру, которая выполняет поиск последней цифры в строке, переданной в качестве параметра. Используя данную процедуру обработать 5 строк, вводимых пользователем. Результат вывести на экран.
19	Реализовать процедуру, возвращающую минимальное из трех целочисленных значений. Используя данную процедуру, определить максимум четырех троек значений. Значения максимума записать в бинарный файл. Исходные данные и имя файла жестко кодируются в программе.
20	Реализовать процедуру, выводящую целое положительное число типа word на экран в шестнадцатеричной системе счисления. Вывести на экран пять чисел, используя разработанную процедуру. Исходные значения жестко кодируются в программе.
21	Реализовать процедуру, которая осуществляет поиск последнего знака пунктуации в строке, переданной в качестве параметра. Используя данную процедуру обработать 5 строк, вводимых пользователем. Результат вывести на экран.
22	Реализовать процедуру, вычисляющую количество бит установленных в ноль в целом значении типа dword. Значение передается в параметрах. Используя процедуру обработать 10 чисел, значения которых жестко кодируются в программе. Результат вывести на экран.
23	Реализовать процедуру, определяющую является ли переданная в параметрах строка целым числом в шестнадцатеричной системе счисления. Если является, то функция возвращает 1. В противном случае – 0. обработать 5 строк вводимых пользователем. Длина строки не превышает 50 символов. Результат для каждой вывести на экран.
24	Реализовать процедуру, определяющую минимальную цифру (восьмеричная система) в целочисленном значении типа word, переданном в параметре. Используя данную процедуру обработать 10 чисел, значение которых жестко кодируется в программе. Результат вывести на экран.
25	Реализовать процедуру, вычисляющую значение выражение по формуле X*8-30. Значение X и результат имеют тип word. Используя разработанную процедуру, вычислить значение для 10 значений, которые жестко кодируются в программе. Результат записать в бинарный файл.
26	Реализовать процедуру, вычисляющую сумму цифр числа (тип word, в шестнадцатеричной системе) переданного в параметре. Используя процедуру, обработать 5 чисел, значение которых жестко кодируется в программе. Результат вывести на экран.
27	Реализовать процедуру, вычисляющую количество цифр в строке, переданной в параметрах. Используя процедуру, определить количество цифр в трех строках. Исходные строки вводятся пользователем. Максимальная длина строк не превышает 80 символов. Результат вывести на экран.
28	Реализовать процедуру, осуществляющую поиск первого знака пунктуации в строке, переданной в качестве параметра. Используя данную процедуру обработать 5 строк, вводимых пользователем. Результат вывести на экран.
29	Реализовать процедуру, определяющую является ли переданная в параметрах строка целым числом в восьмеричной системе счисления. Если является, то функция возвращает 1. В противном случае – 0. обработать 5 строк вводимых пользователем. Длина строки не превышает 50 символов. Результат для каждой вывести на экран.

30	Реализовать процедуру, выводящую целое положительное число типа word на экран в двоичной системе счисления. Вывести на экран пять чисел, используя разработанную процедуру. Исходные значения жестко кодируются в программе.

Контрольные вопросы

1. Что такое процедура?
2. Как осуществляется описание процедуры на языке Ассемблер?
3. Где могут описываться процедуры на языке Ассемблер?
4. Каким образом осуществляется передача параметров в процедуры на языке Ассемблер?
5. Какие недостатки регистровой передачи параметров?
6. Какие недостатки передачи параметров через общую область?
7. Как осуществляется передача параметров через стек?
8. Каким образом осуществляется возврат значений процедурами на языке Ассемблер?
9. Какие недостатки возврата значений посредством регистров?
10. Какие недостатки возврата значений через общую память?
11. Как осуществляется возврат значений через стек?
12. Как осуществляется организация и обращение к локальным переменным процедуры?