Отчет по лабораторной работе №1

в рамках учебной дисциплины

“Языки ассемблера”

**Постановка задачи**

Реализация алгоритма записи шестнадцатеричного числа в строку. Для достижения цели необходимо выполнить следующие пункты:

● Разработать блок-схемы для необходимых процедур

● Реализовать процедуры для которых были составлены блок-схемы

● Реализовать необходимый набор тестов

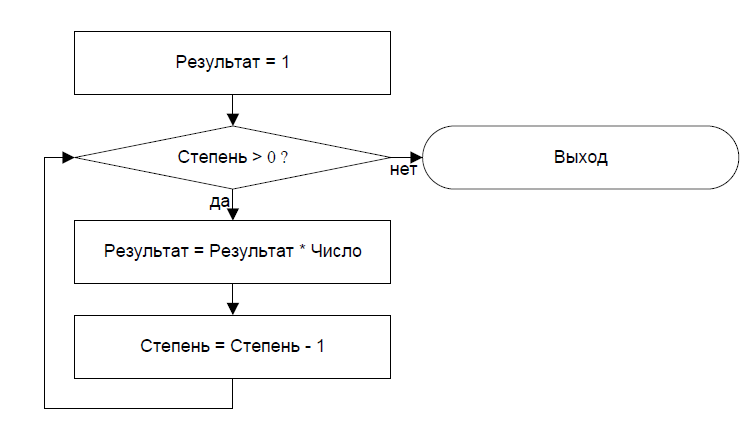
**Описание решения**

Для реализации необходимого алгоритма были написаны следующие процедуры:

1. Power – возведение в степень.
2. HexToChar – процедура сопоставления шестнадцатеричной цифры соответствующему ей символу.
3. HexToStr – запись шестнадцатеричного числа в строку.

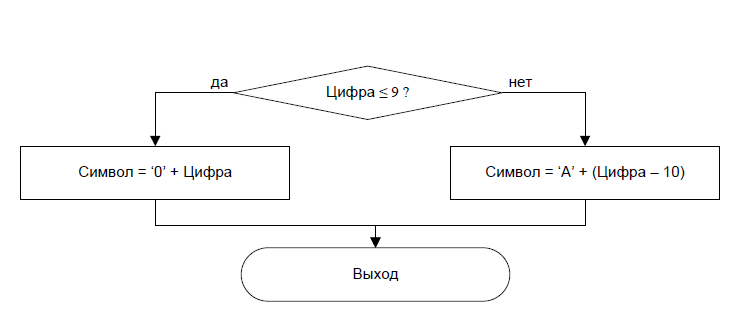
**Блок схемы**

1) Процедура Power:



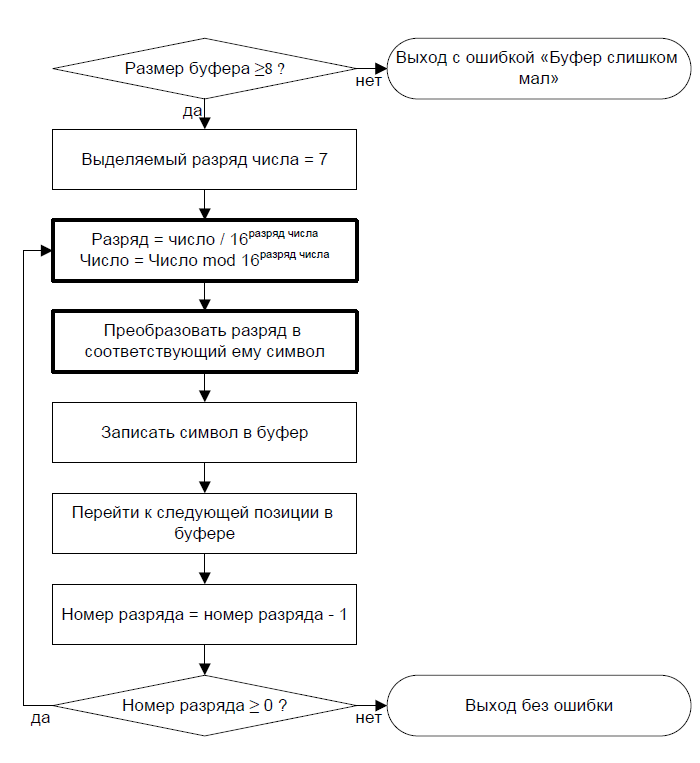
*Рис. 1. Блок-схема алгоритма возведения числа в степень.*

2) Процедура HexToChar:



*Рис. 2. Блок-схема алгоритма преобразования шестнадцатеричной цифры в соответствующий ей символ.*

3) Процедура HexToStr:

**

*Рис. 3. Блок-схема алгоритма вывода шестнадцатеричного числа в строку.*

**Фрагменты программы**

1) Процедура Power:

;Процедура возведения числа в степень

;Вход:

; EAX - беззнаковое число

; ECX - степень

;

;Выход: EBX - результат возведения числа в заданную степень

Power proc

push eax

push ecx

mov eax, 1 ;Регистр временного размещения результата

@@Loop:

cmp ecx, 0 ;Если закончили умножать число само на себя ECX раз,

jna @@EndProc ;то выходим.

;Иначе продолжаем возводить число в степень,

shl eax, 4 ;сдвигая на 4 разряда,

dec ecx ;а затем уменьшая количество оставшихся умножений числа.

jmp @@Loop ;Переходим на следующую итерацию

@@EndProc:

mov ebx, eax ;Переносим результат возведения в степень

pop ecx

pop eax

ret

Power endp

2) Процедура HexToChar:

;Процедура преобразования шестнадцатиричной цифры в соответствующий ей символ.

;Вход:

; AL - шестнадцатиричная цифра(0-9, A-F)

;

;Выход:

; AL - символ "0" - "9", "А" - "F", соответствующий исходной цифре

HexToChar proc

push dx

cmp al, 9 ;Если шестнадцатиричная цифра больше 9,

ja @@Alpha ;то преобразование цифры в символ происходит по алгоритму,

;написанному после метки @@Alpha

mov dl, '0' ;иначе получаем код символа

add dl, al ;смещением на al позиций относительно кода "0"

jmp @@EndProc

@@Alpha:цифр

mov dl, 'A' - 10 ;для шестнадцатиричных цифр A-F код символа можеть быть получен по

add dl, al ;формуле: код символа = 'A' - (Цифра - 10)

@@EndProc:

mov al, dl

pop dx

ret

HexToChar endp

3) Процедура HexToStr:

;Процедура осуществляет запись шестнадцатеричного числа в строку

;Число записывается в виде хххххххх, где х - символ 0-9, A-F

;Вход: EAX - число(беззнаковое)

; ES:EDI - адрес буфера, куда будет помещена строка, предсталяющая щестнадцатиричное число

; ECX - размер буфера - беззнаковое число

;Выход: EAX = 0, если успешно

; ES:EDI - указывает на следующий символ в буфере после числа

; EAX = 1, если буфер слишком мал

HexToStr proc

push ebx

push ecx

push edx

cmp ecx, 8 ;Если размер буфера меньше 8,

jb @@BufferTooSmall

;то выходим со ошибкой

mov ecx, 7 ;Выделяемый разраяд числа = 7

@@ExctractHexLoop:

;Цикл последовательного выделения

;разрядов числа, начиная со старшего

mov ebx, 16

call Power ;Возводим 16 в степень выделяемого разряда

mov edx, 0 ;и делим на него оставшееся число

div ebx ;EAX = частное(выделенный разряд)

;EDX = остаток(оставшаяся часть числа)

call HexToChar ;Переводим выделенный разряд в символ

mov es:[edi], al ;Записываем его в строку

inc edi ;и переходим к следующему символу в строке

mov eax, edx ;Иначе сохраняем оставшееся число,

loop @@ExctractHexLoop ;и продолжаем обработку числа

@@EndLoop: ;Выход без ошибки

mov eax, 0

jmp @@EndProc

@@BufferTooSmall: ;Выход с ошибкой "буфер слишком мал"

mov eax, 1

@@EndProc:

pop edx

pop ecx

pop ebx

ret

HexToStr endp

**Тестовый набор данных**

Входные данные:

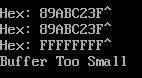
1) Корректные данные:

|  |  |  |
| --- | --- | --- |
| EAX | ECX | Пояснение |
| 89ABC23Fh | 8 | Корректный выход:  "Hex: 89ABCDEF^" |
| 89ABC23Fh | 12 | Корректный выход:  "Hex: 89ABCDEF^" |
| -1 | 8 | Корректный выход:  "Hex: FFFFFFFF^" |

2) Некорректные данные:

|  |  |  |
| --- | --- | --- |
| EAX | ECX | Пояснение |
| 0FFFFFFFFh | 1 | Выход с ошибкой: буфер слишком мал |

Результат выполнения тестов:



**Покрытие тестами кода программы**

Приведенные в таблице тестовые наборы данных полностью покрывают все возможные исходы программы. Присутствуют тесты проверки корректного вывода процедуры при различных размерах буфера. Всего тестовых данных для проверки всех ответвлений и проверки корректной реакции процедуры на различные входные данные необходимо 4 штуки, какие и представлены в таблице тестовых данных.

*Вывод*: Во время выполнения лабораторной работы был составлен алгоритм выполняющий перевод шестнадцатеричного числа в строку. На основе этого алгоритма была реализована программа на языке ассемблера для платформы х86. Также в ходе лабораторной работы были улучшены навыки проектирования и составления алгоритмов, а также улучшены навыки написания программ на языке ассемблера и их тестирования для платформы х86.

# Ответы на контрольные вопросы.

1. 95h = 10010101b

81h = 10000001b

Тогда преобразование из 95h в 81h можно выполнить следующей командой:

mov al, 95h

and al, 11101011b

1. 15h = 00010101b

0D5h = 11010101b

Тогда преобразование из 15h в 0D5h можно выполнить следующей командой:

mov al, 15h

or al, 11000000b

1. В результате деления 5723h на 10h должно получиться частное равное 572h, однако это значение невозможно разместить в регистре al. Следовательно, будет сгенерировано исключение с номером 0 - ошибка деления.
2. mov ecx, -1

mov al, 0

repne scasb

not ecx

dec ecx

бежим с конца и сравниваем с 0

1. mov cx, n

Marker:

… ; тело цикла  
loop Marker

До тех пор пока --cx переходим по метке

1. mov cx, n

Marker:

… ; тело цикла  
 dec cx

cmp cx, 0

jne Marker

Аналогично предыдущему

1. Marker1:

cmp X, 0  
 jle Marker2

… ; тело цикла  
jmp Marker1

Marker2:

...

если первый операнд меньше второго операнда или равен ему то переходим в противном случае крутимся

1. Marker:

… ;тело цикла  
cmp X, 0  
jle Marker

Если предыдущее условие то крутимся

1. mov cx, 10  
    mov ax, 00  
   Summa:  
    add ax, cx  
    loop Summa

cx же уменьшается

1. mov ax, 1  
    mov cx, N  
   Factorial:

mul cx  
 loop Factorial

mul умножает на ax базово