Отчет по лабораторной работе №2

в рамках учебной дисциплины

“Языки ассемблера”

**Постановка задачи**

Реализация алгоритма записи десятичного числа в строку с выравниванием по правому или левому краю. 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 и макросы:

1. DigitLen – вычисление длины числа.
2. len\_checking – в зависимости от типа выравнивания, вычисляет размер буфера для записываемого числа.
3. buffer\_check – проверяет корректность размера буфера.
4. get\_digits – перевод числа в строковое представление и дальнейшая запись его в буфер.
5. Dec2Str – запись десятичного знакового числа в буфер с выравниванием.

**Блок схемы**

1) Процедура DigitLen:

Получить число

Задать переменную count = 10

Задаем переменную mask = 10000000

нет да

Count == 1

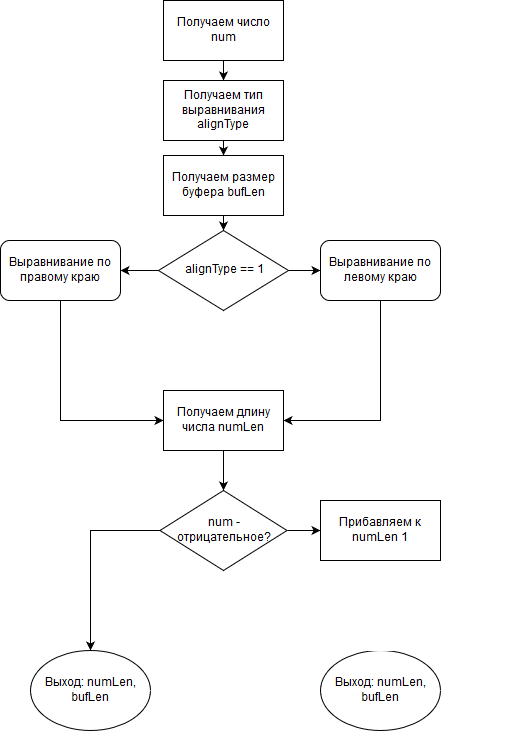
Число > mask

Вывод: Count

Mask /= 10

Count-=1

2) Процедура len\_checking:



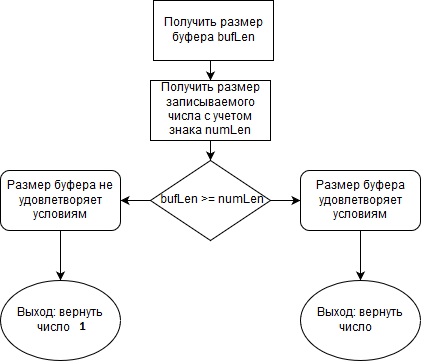
negativeNum

positiveNum

alignType1

alignType2

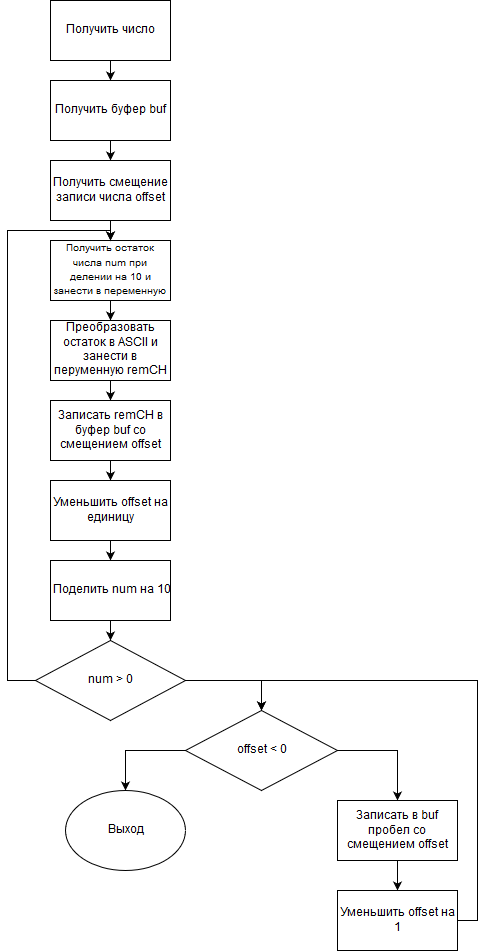
3) Процедура buffer\_check:



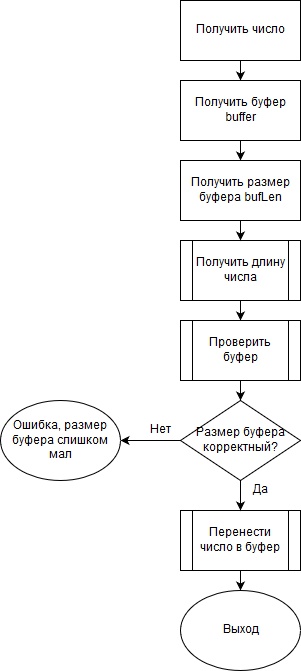
bufStatusErr

bufStatusOk

4) Процедура get\_digits:



5) Процедура Dec2Str:



**Фрагменты программы**

1) Процедура DigitLen:

;

; Макрос проверяет длину числа, содержащегося в EAX

;

; Вход:

; EAX – число, для которого подсчитывается длина

; Выход:

; BX – длина числа содержащегося в EAX

;

DigitLen MACRO

mov esi, 10 ; счетчик чисел

l = 1000000000 ; маска с которой будем сравнивать

REPT 9 ;цикл 9 раз повторяющий операцию

cmp eax, l ; сравниваем число с маской

sbb esi, 0 ; вычитаем из счетчика 1 или 0 в зависимости от значения сравнения

l = l / 10 ;уменьшаем маску на 10

ENDM

ENDM DigitLen

2) Процедура signIs:

;

; Перевод числа в строковое представление и дальнейшая запись его в ; буфер

;

; Вход:

; EAX – число для перевода в строку

;

;Выход:

; EDX - 0 или 1 в зависимости от знака

;

@@signIs:

push eax ;сохраняем число

shr eax, 31;побитово сдвигаем до 1го бита

cmp eax, 0; проверяем 1 бит

jz @@plus;если положительное

pop eax;возращаем число

neg eax;переводим из отрицательного

mov edx, 1;флаг знака

jmp @@flag

@@plus:

pop eax

mov edx, 0

@@flag:

push edx

2) Процедура len\_checking:

;

; В зависимости от типа выравнивания, вычисляет

; размер буфера для записываемого числа

;

; Вход:

; BL = 0 / не ноль (В зависимости от типа

; выравнивания)

; EAX – исходное число для перевода в строку

; EDX = 0 / -1 (В зависимости от знака числа в EAX)

; Выход:

; ESI – длина числа в виде строки с учетом знака

;

@@len\_checking:

cmp bl, 0 ;проверяем тип выравнивания

jnz @@right\_align ;не 0 - правое

DigitLen ;определяем длину числа

add esi, edx ; записываем в esi длину

jmp @@buffer\_check

@@right\_align:

mov esi, 12 ; необходимый размер = 12

3) Процедура buffer\_check:

;

; Проверяет корректность размера буфера

;

; Вход:

; ECX – размер буфера

; ESI – длина записываемого числа с учетом знака

; Выход:

; Ошибка / успешное продолжение

;

@@buffer\_check:

push ebx

mov ebx, 10d ;делитель для числа

cmp ecx, esi ;проверяем достаточно ли места

jb @@error\_message

5) Процедура get\_digits:

;

; Перевод числа в строковое представление и дальнейшая запись его в ; буфер

;

; Вход:

; EAX – число для перевода в строку

; EDI – буфер для записи числа

; ESI – длина записываемого числа с учетом знака

; Выход:

; EDI - буфер для записи числа

;

@@get\_digits:

xor edx, edx ;обнуляем edx

div ebx ;делим на длину слова

or dl, 30h ; остаток в букву

mov es:[edi+esi], dl ;записываем букву

dec esi ;уменьщаем счетчик

cmp eax, 0 ;если закончили то в пустые места - пробелы

jz @@minus

loop @@get\_digits

@@minus:

pop ebx;возвращаем edx

pop edx;возвращаем флаг

cmp edx, 0

jz @@spacing;если не отрицательное

mov al, 2Dh;если да ставим минус

mov es:[edi + esi], al

dec esi;уменьщаем счетчик

mov al, 20h ; записываем пробел

@@spacing:

bt esi, 31 ;проверяем отрицательно ли число

jc @@EndLoop ;переход если да, то выходим

mov es:[edi + esi], al ;записываем в строку

dec esi ;уменьшаем счетчик

mov al, 20h ; записываем пробел

loop @@spacing

@@error\_message:

pop edx

pop ebx

mov eax, 1;флаг ошибки

@@EndLoop:

pop ecx

@@EndProc:

ret

Dec2Str endp

**Тестовый набор данных**

Входные данные:

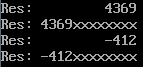
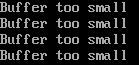
1) Корректные данные:

|  |  |  |  |
| --- | --- | --- | --- |
| EAX | ECX | BL | Пояснение |
| 4369 | 11 | 1 | Корректный вывод, с положительным числом.  Выравнивание справа. |
| 4369 | 11 | 0 | Корректный вывод, с положительным числом.  Выравнивание слева. |
| -412 | 11 | 1 | Корректный вывод, с отрицательным числом.  Выравнивание справа. |
| -412 | 11 | 0 | Корректный вывод, с отрицательным числом.  Выравнивание слева. |

2) Некорректные данные:

|  |  |  |  |
| --- | --- | --- | --- |
| EAX | ECX | BL | Пояснение |
| 4369 | 2 | 1 | Переполнение буфера, с положительным числом.  Выравнивание справа. |
| 4369 | 2 | 0 | Переполнение буфера, с положительным числом.  Выравнивание слева. |
| -2147483648 | 9 | 0 | Переполнение буфера, с отрицательным числом.  Выравнивание слева. |
| -2147483648 | 10 | 1 | Переполнение буфера, с отрицательным числом.  Выравнивание справа. |

Результат выполнения тестов:

**Покрытие тестами кода программы**

Приведенные в таблице тестовые наборы данных полностью покрывают все возможные исходы программы, так как в алгоритме присутствуют такие проверки значений, как проверка на положительное, так и на отрицательное число (positiveNum, negativeNum), проверка буфера на корректный размер (buuferOk, bufferErr) и проверка на выравнивание (alignType1, alignType2). Всего тестовых данных для проверки всех ответвлений необходимо 8 штук, какие и представлены в таблице тестовых данных.

*Вывод*: Во время выполнения лабораторной работы был составлен алгоритм выполняющий перевод десятичного числа в строку с выравниванием по правому или левому краю. 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)

Получить число

Задать переменную count = 10

Задаем переменную mask = 10000000

нет да

Count == 1

Число > mask

Вывод: Count

Mask /= 10

Count-=1

2) mov [edi + 4], eax

3) lea eax, buffer

4) bt eax, 31

5) shld eax, ebx, 1

6)

