### Постановка задачи

Получение практического навыка разработки процедур на примере алгоритма перевода строки в десятичное число. Для достижения цели необходимо выполнить следующие задачи:

* Разработать блок схемы для следующих процедур:
  1. SkipSpace – Процедура осуществляющая пропуск пробельных символов в начале строки
  2. IsNegative - Процедура осуществляющая проверку числа на отрицательность
  3. CheckZero - Процедура осуществляющая проверку исключительной ситуации , когда нуль стоит в начале разряда
  4. Convert – Процедура осуществляющая перевод содержимого строки в десятичное знаковое число
  5. StrToDec - Процедура осуществляющая перевод содержимого строки в десятичное знаковое число
  6. CountOfDigits - Процедура осуществляющая подсчёт количества разрядов числа, учитывая отрицательный знак
  7. PrintDexStr - Процедура осуществляющая запись десятичного знакового числа в строку (Обертка для DexToStr)
  8. DexToStr – Процедура, осуществляющая запись десятичного знакового числа в строку
  9. Power – Процедура, осуществляющая возведение десятичного знакового числа в степень
* Используя теоретические знания команд Ассемблера, записать, процедуры, перечисленные выше на нем;
* Сформировать необходимый набор тестов.

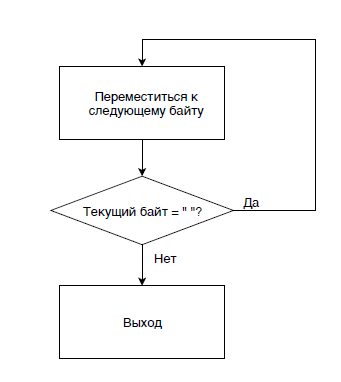
### Описание решения

На основании указанных выше процедур были составлены их блоксхемы, приведем описание каждой из них:

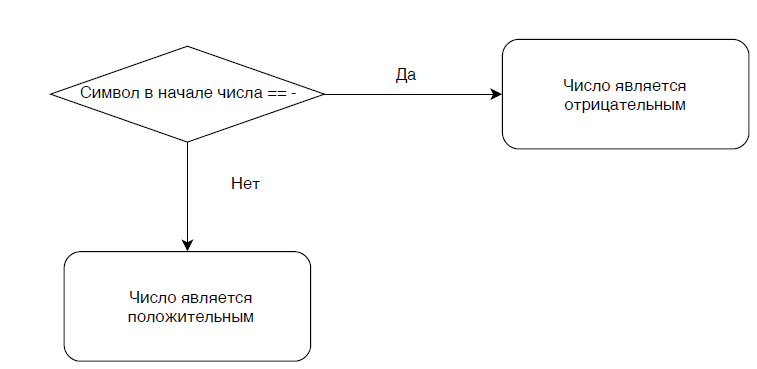
1. SkipSpace – получает на вход строку с потенциальными пробелами в начале, и в цикле пока первый символ является пробельным от входной строки отрезается первый элемент.
2. IsNegative – получает на вход строку, сравнивает первый символ с минусом, отрезает от строки первый символ, в случае если он «-». Также вывести информацию об отрицательности числа.
3. CheckZero – на вход поступает строка, в которой записано число, а также сообщение об его отрицательности, сравнивая первый символ на равенство нулю предусмотреть выходы число нулевое, число -0, число не нулевое.
4. Convert – на вход поступает строка, применяется процедура по избавлению от пробелов, затем проверяется число на отрицательность, равенство нулю, делаются соответствующие выводы, формируются необходимые выходные значения. Далее каждый последующий символ проверяется на корректность подсчитывает поразрядно итоговое число, в случае некорректности какого-либо текущего символа выдать сообщение о некорректности строки, в случае обнаружения допустимого окончания строки выдать сформированное число в десятичной системе исчисления.
5. StrToDec –на вход исходная строка, проверить ее на пустоту, вызвать процедуру Convert.
6. CountOfDigits – на вход число в десятичном виде, проверить является ли число отрицательным, затем производя деление на 10 сравнивается полученный остаток с 0, в случае если на текущем шаге сравнение верно, то вывести сформированное количество знаков, если неполное частное не ровно нулю, увеличить счетчик количества разрядов.
7. PrintDexStr – на вход число, подсчитать количество знаков, убедиться, что размер буфера достаточен, вызвать процедуру переводящую число в строку для вывода.
8. DexToStr – на вход число, проверить является ли оно отрицательным, извлечь текущую цифру в записи числа путем деление на 10 в соответствующей степени, которое получено при помощи процедуры Power, преобразовать полученную цифру в символ, перейти к следующему разрядку числа, повторять пока разряды не кончатся
9. Power – на вход какой конкретно символ нужно извлечь, в цикле последовательно увеличивать степень десяти, уменьшая входное значения, ожидая, когда оно обратится в ноль.

### Блок схемы

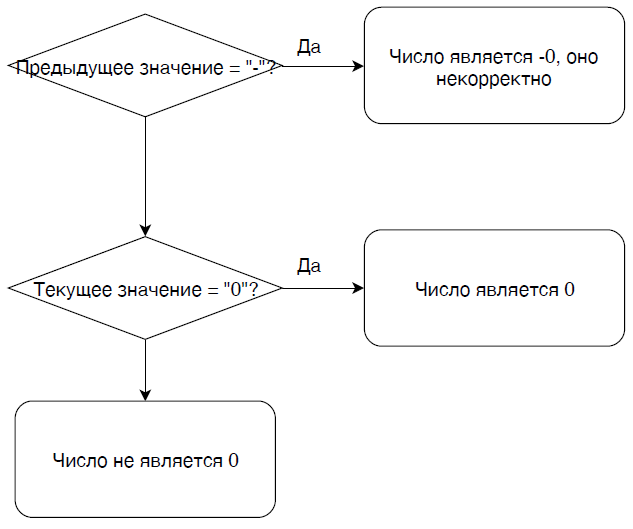
1. Процедура SkipSpace



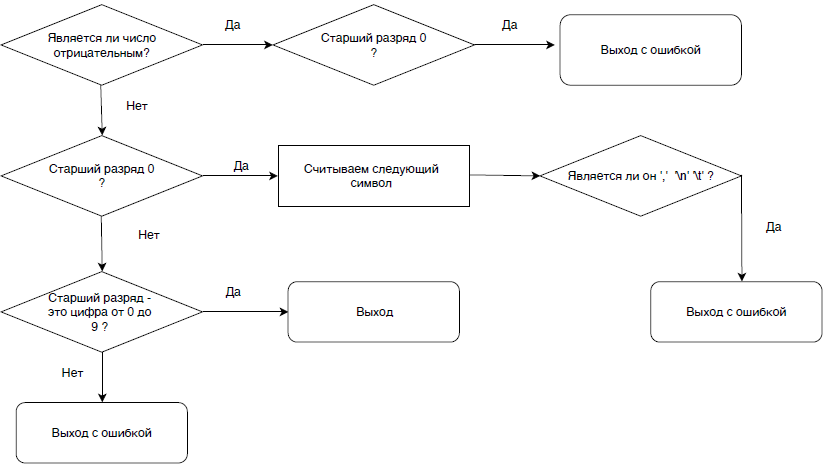
1. Процедура IsNegative



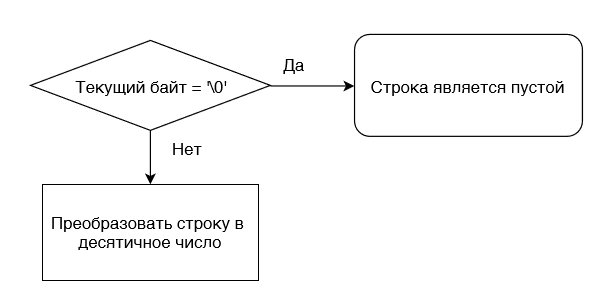
1. Процедура CheckZero



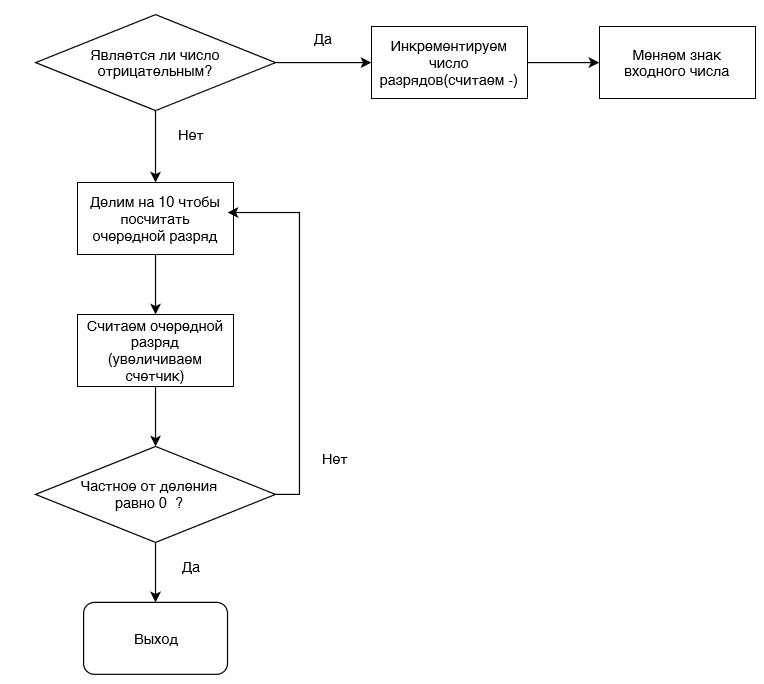
1. Процедура Convert



1. Процедура StrToDec



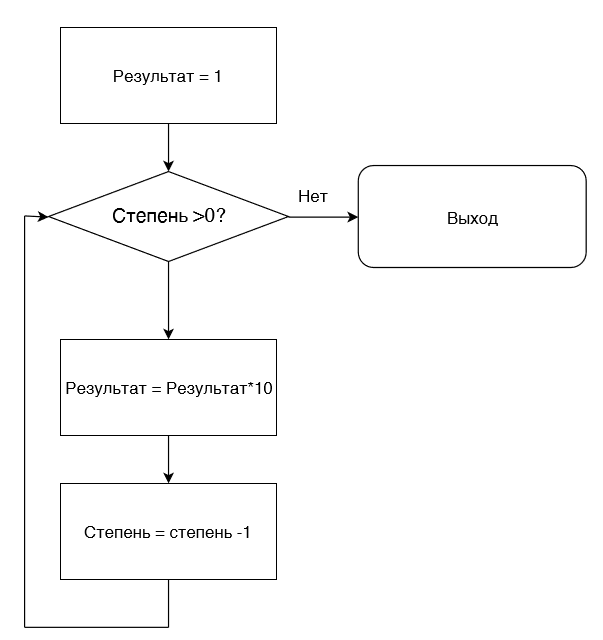
1. Процедура CountOfDigits



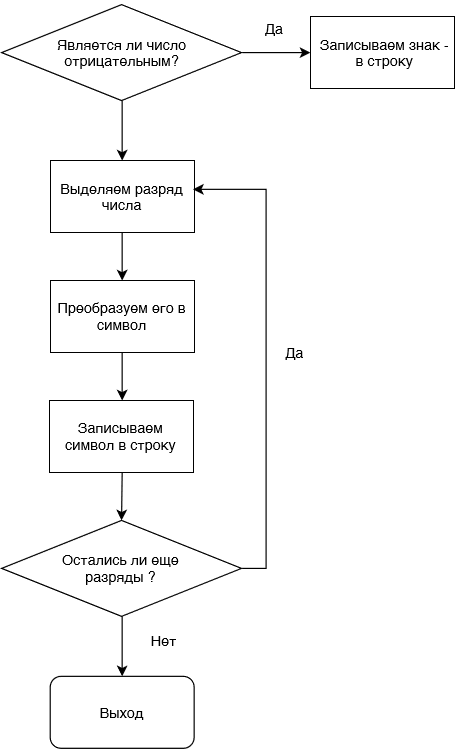
1. Процедура PrintDexStr



1. Процедура Power



1. Процедура DexToStr



### Фрагменты программы

1. Процедура SkipSpace

;

; Процедура, пропуска пробельных символов в начале строки

;

; Вход:

; DS:ESI - строка

; Выход:

; DS:ESI - строка

;

SkipSpace proc

@@Loop:

lodsb ; считываем символ в AL

cmp al, ' ' ; Проверяем пробел ли это

je @@Loop ; если да - идем дальше

ret

SkipSpace endp

1. Процедура IsNegative

;

; Проверка числа на отрицательность

;

; Вход: AL - символ в начале числа

; Выход: EBX = -1 / 1 (отриц./полож.)

;

IsNegative proc

cmp al, '-' ;сравниваем с минусом

je @@Negative

ret

@@Negative:

mov ebx, -1 ;помещаем -1

lodsb ;берем следующий символ

ret

IsNegative endp

1. Процедура CheckZero

;

; Процедура проверки проверки исключительной ситуации с нулем, в начале разряда

;

; Вход:

; EBX = -1 / 1 (в зависимости от знака числа)

; AL - символ в строке, старший разряд

; Выход:

; EBX = 3 - разряд равен 0

; EBX = 2 - '-0' в строке

; EBX = -1 / 1, иначе

;

CheckZero proc

cmp ebx, -1 ;отрицательное ли число

je @@Negative

cmp al, '0' ;если нет проверяем на 0

je @@SingleZero ;если да помещаем 3

ret

@@SingleZero:

mov ebx, 3 ;помещаем 3

ret

@@Negative: ; если отрицательное

cmp al, '0' ; проверяем на ноль следующий

je @@NegativeZero ; если ноль то добавляем 2

ret

@@NegativeZero:

mov ebx, 2

ret

CheckZero endp

1. Процедура Convert

;

; Процедура, осуществляющая перевод содержимого строки в десятичное знаковое число

;

; Вход:

; DS:ESI - адрес строки

; Выход:

; EAX - код ошибки (2, 4)

;

Convert proc

xor eax, eax ; Обнуляем AX

xor edx, edx ; Обнуляем DX

call SkipSpace ; Пропуск пробелов

mov ebx, 1 ; EBX = 1 Положительное число

call IsNegative

call CheckZero ; Проверка на '-0' и 0

cmp ebx, 2 ; если -0, то EBX == 2

je @@InvalidSymb ; Тогда выдаем ошибку некорректного символа

cmp ebx, 3 ; Если первый разряд '+0'

je @@Zero

cmp al, '0' ; Проверка на корректность первого символа

jb @@InvalidSymb ; Проверка что символ = 0-9

cmp al, '9'

jg @@InvalidSymb

cmp al, '0'

dec esi

@@ExtractSymbol:

mov al, byte ptr ds:[esi] ; Берем очередной символ

jmp @@IsStop ; Проверка на остановку подсчету

jmp @@IsCorrect ; Проверка, корректен ли очередной символ

@@Extract:

cmp ebx, -1

je @@ExtractNeg

@@ExtractPos:

sub al, '0' ; Получаем число

imul edx, 10

jo @@Overflow ;проверка не переполнение

add edx, eax

jo @@Overflow ;проверка не переполнение

inc esi

jmp @@ExtractSymbol

@@ExtractNeg:

sub al, '0'

imul edx, 10

jo @@Overflow ;проверка не переполнение

sub edx, eax

jo @@Overflow

inc esi

jmp @@ExtractSymbol

@@StopExtract:

mov ebx, edx ; Сохраняем значение в EBX

mov eax, 0

jmp @@EndProc

@@IsStop:

cmp al, ','

je @@StopExtract

cmp al, ' '

je @@StopExtract

cmp al, 10 ;\n

je @@StopExtract

cmp al, 0 ; \0

je @@StopExtract

cmp al, 13 ; \r

je @@StopExtract

jmp @@Extract

@@Overflow:

mov eax, 4 ; Устанавливаем код ошибки = 4 - переполнение

jmp @@EndProc

@@IsCorrect: ; Проверка, на то что символ есть

cmp al, '0' ; символ '0'-'9'

jb @@InvalidSymb ; Если нет, то некорректный символ

cmp al, '9'

jg @@InvalidSymb

jmp @@Extract

@@InvalidSymb:

mov eax, 3 ; Код ошибки = 3 - Некорректный символ

jmp @@EndProc

@@Zero: ; Обработка ситуации - старший разряд = '0'

inc esi

mov al, byte ptr ds:[esi] ; Берем очередной символ

mov edx, 0 ; Помещаем текущее значение = 0

cmp al, ',' ; Проверка на символ после нуля

je @@StopExtract ; -если, символ Останова, то выходим, с EDX = 0

cmp al, ' ' ; -если, символ не останова, ошибка Корректности символа

je @@StopExtract

cmp al, 10

je @@StopExtract

cmp al, 0

je @@StopExtract

cmp al, 13

je @@StopExtract

;-если, символ не останова, ошибка Корректности символа

jmp @@InvalidSymb

@@EndProc:

ret

Convert endp

1. Процедура StrToDec

;

; Процедура, осуществляющая перевод содержимого строки в десятичное знаковое число

;

; Вход:

; DS:ESI - адрес строки

; Выход:

; EAX - код ошибки(2)

;

StrToDec proc

push edx

push ecx

mov al, byte ptr ds:[esi]

test al, al ;Если это нулевой байт

jz @@EmptyStr

call Convert

@@EndProc:

pop ecx

pop edx

ret

@@EmptyStr: ; Ошибка = 2 - пустая строка

mov eax, 2

jmp @@EndProc

StrToDec endp

1. Процедура CountOfDigits

;

; Процедура осуществляющая подсчёт количества разрядов числа, учитывая отрицательный знак

;

;

; Вход:

; EAX - число

; Выход:

; EBX - число разрядов

;

CountOfDigits proc

push eax

push ecx

push edx

mov ebx, 0

cmp eax, 0 ;Проверка на отрицательность числа

jl @@IsNegative

jmp @@CountLoop

@@IsNegative:

inc ebx ;Инкрементируем число разрядов(считаем -)

mov edx, -1

mul edx ;Меняем знак входного числа

@@CountLoop:

mov edx, 0

mov ecx, 10

div ecx ;Делим на 10 чтобы посчитать очередной разряд

inc ebx ;Считаем очередной разряд

cmp eax, 0 ;Если частное от деления равно 0 то выходим из цикла

je @@EndCountLoop

jmp @@CountLoop

@@EndCountLoop:

pop edx

pop ecx

pop eax

retn

CountOfDigits endp

1. Процедура PrintDexStr

;Процедура осуществляющая вывод десятичного числа

;

;

; Вход:

; EAX - число

; EBX - число разрядов

; ECX - размер буфера

; Выход:

; EDI:ESI - строка с числом

;

PrintDexStr proc

push edx

push ecx

push ebx

call CountOfDigits ;Осуществляем подсчёт числа разрядов входного числа

cmp ecx, ebx

jb @@BufferTooSmall ;Выход с ошибкой "Буфер слишком мал"

call DexToStr ;Осуществляем запись десятичного числа в строку

mov eax, 0

jmp @@EndProc ;Выход без ошибки

@@BufferTooSmall:

mov eax, 1

jmp @@EndProc

@@EndProc:

pop ebx

pop ecx

pop edx

ret

PrintDexStr endp

1. Процедура Power

;Процедура получения необходимой степени основания

;Вход:

; EAX - беззнаковое число

; ECX - степень

;

;Выход: EBX - результат возведения числа в заданную степень

Power proc

push eax

push ecx

mov eax, 1

@@PowerLoop:

cmp ecx, 0

jz @@EndPowerLoop

mul ebx ;Накапливаем в eax число возведенное в степень

LOOP @@PowerLoop

@@EndPowerLoop:

mov ebx, eax

pop ecx

pop eax

retn

Power endp

1. Процедура DexToStr

;Процедура осуществляющая вывод десятичного числа с выравниванием

;Вход:

; EAX - число

; ECX - степень

;

;Выход: EBX - результат возведения числа в заданную степень

DexToStr proc

push ebx

push ecx

push edx

mov ecx, ebx

dec ecx

cmp eax, 0

jl @@Negative ;Если число отрицательное то переходим к метке

jmp @@ExtractDexLoop

@@Negative:

mov edx, -1 ;повторяем убирание знака

mul edx

dec ecx

mov dx, 45

mov [edi], dx ;Записываем знак - в строку

inc edi

@@ExtractDexLoop:

mov ebx, 10 ;Помещаем в ebx 10(основание системы счисления)

call Power ;Вызываем процедуру возведения числа в степень

mov edx, 0

div ebx ;Выделяем разряд

add al, '0' ;Преобразования числа в символ

mov [edi], al ;Записываем очередной символ

inc edi

mov eax, edx ;Помещаем edx(остаток от деления) в eax

dec ecx

cmp ecx, -1

je @@EndLoop ;Выход без ошибки

cmp ecx, 0

jae @@ExtractDexLoop

@@EndLoop:

mov eax, 0

jmp EndProc

BufferTooSmall:

mov eax, 1 ;Заносим в eax 1(Код ошибки)

EndProc:

pop edx

pop ecx

pop ebx

retn

DexToStr endp

### Тестовый набор данных

Входные значения:

1. Некорректные входные значения:
   1. “ab”
   2. “--999”
   3. “-0”
   4. “- -23”
   5. “-2147483649”
   6. “098”
   7. “2147483648”
   8. “”
2. Корректные:
   1. “0”
   2. “ 0”
   3. “-2147483648”
   4. “ 2534”

Вывод тестовой программы:

