ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(государственный технический университет)

Кафедра 304
(вычислительные машины, системы и сети)

Лабораторная работа по курсу «Ассемблер»

Отчёт по работе *№4*.

Нелинейное программирование на языке Assembler
(наименование работы)

Вариант задания *№2*.

Лабораторную работу выполнили:

студенты гр. 13-501, Резвяков Денис Михайлович,
(должность) *(Ф. И. О.)*
Коршунов Евгений Владимирович.

Лабораторную работу принял:

доцент каф. 304, к.т.н. Алещенко Алла Степановна _____.
(должность) *(Ф. И. О.)* *(подпись)*

«*1*» *декабря* 2009 г.
(дата приёма)

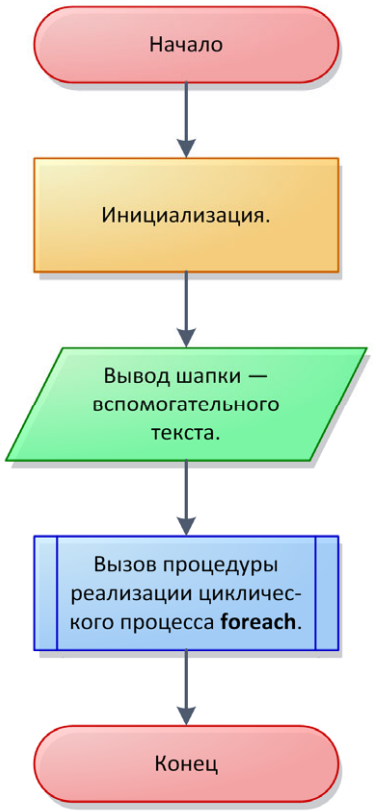
Цель работы: Закрепить навыки программирование блочной структуры программы на Ассемблере и сборку программы используя программный пакет TASM.

Задание

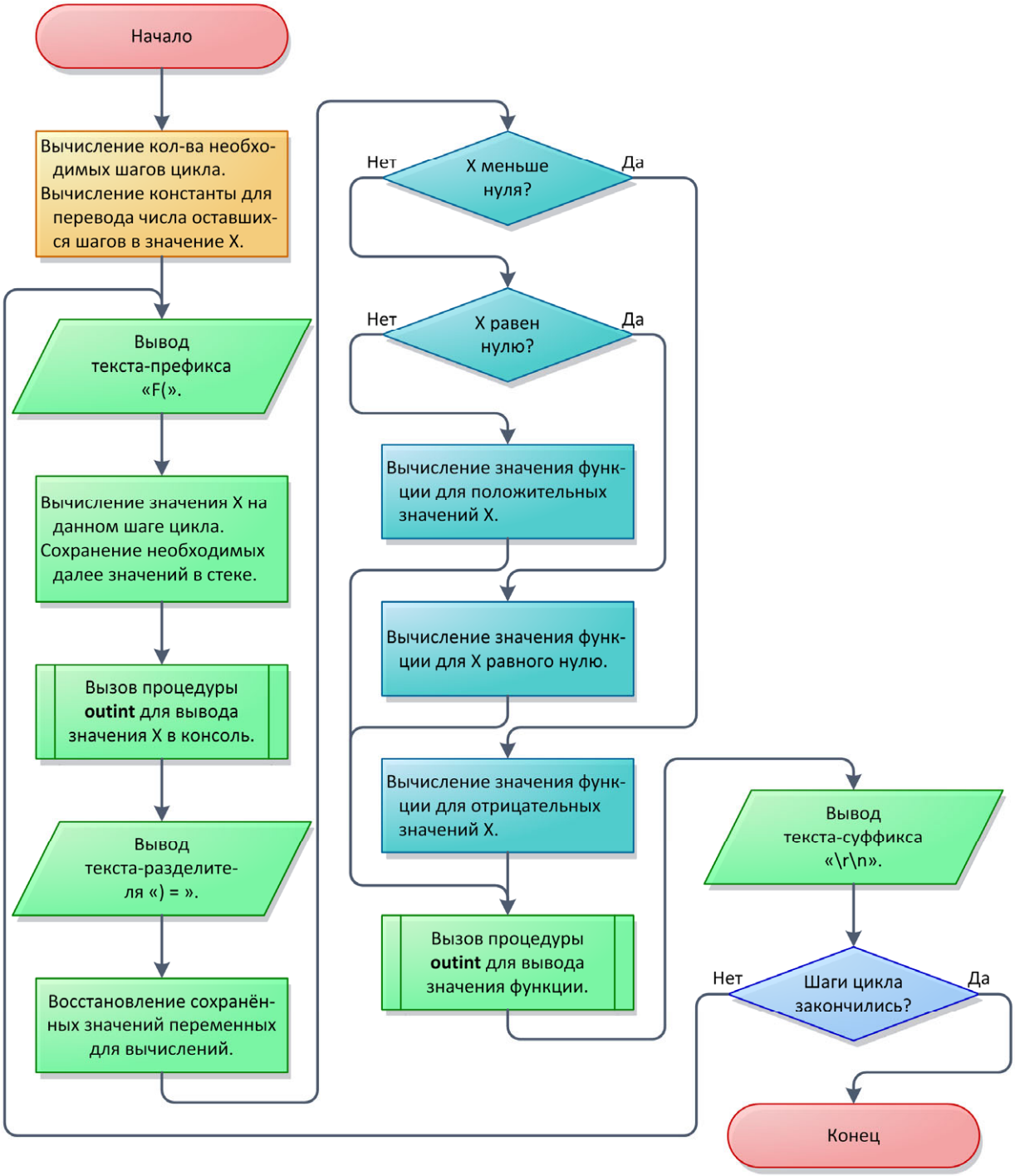
Разработать программу в Турбо-ассемблере для реализации циклического процесса в соответствии с заданием, выполненным в лабораторной работе №2.

Блок-схема

Блок-схема главной процедуры программы:



Блок-схема процедуры реализации циклического процесса:



Блок-схема процедуры вывода целого числа в консоль представлена в отчёте по лабораторной работе №3.

Код программы реализации нелинейного процесса

```
   sseg segment stack
        db 64 dup (?)
   sseg ends
        ;
   dseg segment
        str_result db "f(x) = result", 10, 13, " -----------", 10, 13, "$"
        str_bgn    db "f($"
        str_mdl    db ") = $"
        str_end    db 10, 13, "$"
        value_a dw -6
        value_b dw  4
        value_c dw  8
        value_d dw  2
        x_from  dw -2
        x_to    dw  5
   dseg ends
        ;
   cseg segment
        assume cs:cseg, ds:dseg, ss:sseg
        ;
        ; Процедура вычисления и вывода результата функции
        ;   для набора значений X из промежутка [x_from, x_to].
        ; Процедура ломает AX, BX, CX, DX.
foreach proc near
        mov  cx, x_to    ; Занесение в регистр CX конца промежутка значений X
        inc  cx          ; Увеличение CX на 1 для преобразования счётчика в X
        push bp          ; Сохранение старого локального смещения стека
        push cx          ; Сохранение значения для смещения начала отсчёта
        mov  bp, sp      ; Установка локального смещения стека из текущего
        sub  cx, x_from  ; Вычитание из CX начального X - кол-во шагов цикла

fe_bgn: lea  dx, str_bgn ; Помещаем в регистр DX адрес строки-префикса
        mov  ah, 9       ; Помещаем в регистр AH код операции вывода строки
        int  21h         ; Вызываем прерывание для вывода строки на экран
        mov  ax, [ss:bp] ; Считываем значение смещения начала отсчёта
        sub  ax, cx      ; Вычитаем текущий шаг и получаем текущий X
        push cx          ; Кладём CX (счётчик шагов цикла) в стек
        push ax          ; Кладём AX (текущее значение X) в стек
        call outint      ; Выводим текущее значение числа X (в регистре AX)
        lea  dx, str_mdl ; Помещаем в регистр DX адрес строки-разделителя
        mov  ah, 9       ; Помещаем в регистр AH код операции вывода строки
        int  21h         ; Вызываем прерывание для вывода строки на экран
        mov  ax, value_a ; Кладём в регистр AX значение A (для любого случая)
        pop  bx          ; Восстанавливаем значение X из стека в регистр BX
        cmp  bx, 0       ; Сравниваем текущий X (в BX) с нулём
        jl   fe_neg      ; Если X меньше нуля, переходим на метку...
        je   fe_zro      ; Если X равен нулю, переходим на метку...
```

```
fe_pos: imul ax          ; X положительный. Умнажаем AX само на себя
        sub  ax, bx      ; Вычитаем X (в BX): (DX,AX) = (DX,AX) - BX
        sbb  dx, 0       ; Продолжаем вычитание для DX (A^2 - (0:X))
        mov  cx, value_c ; Кладём значение C в регистр CX
        add  cx, value_d ; Прибавляем к C (в CX) значение D - C+D
        idiv cx          ; Получаем результат: AX = DX:AX / CX
        jmp  fe_nxt      ; Переходим на вывод результата
fe_zro: sub  ax, value_b ; X равен нулю. Вычитаем: AX=A-B, в регистре AX
        cwd              ; Расширяем слово до двойного слова DX:AX = A-B
        idiv value_d     ; Получаем результат: AX = DX:AX / D
        jmp  fe_nxt      ; Переходим на вывод результата
fe_neg: mov  cx, 2       ; X отрицательный. Помещаем в регистр CX число 2
        imul cx          ; Умножаем AX на 2:  (DX:AX) = 2*A
        imul bx          ; Умножаем AX на BX: (DX:AX) = 2*A*X
        add  ax, 5       ; Получаем результат: AX = 2*A*X + 5
fe_nxt: call outint      ; Выводим текущий результат от переменной X (в AX)
        lea  dx, str_end ; Помещаем в регистр DX адрес кодов для новой строки
        mov  ah, 9       ; Помещаем в регистр AH код операции вывода строки
        int  21h         ; Вызываем прерывание для вывода строки на экран
        pop  cx          ; Восстанавливаем значение счётчика цикла из стека
        loop fe_bgn      ; Продолжение цикла или его окончание
        pop  cx          ; Забираем из стека смещение начала отсчёта
        pop  bp          ; Восстанавливаем исходного локального смещения стека
        ret
foreach endp
        ;
        ; Процедура вывода числа, хранящегося в AX.
        ; Процедура ломает AX, BX, CX, DX.
 outint proc near
        xor  bx, bx  ; Обнуляем BX - это будет признак отрицательного числа
        mov  cx, 10  ; В CX заносим число 10 - основание десятичной системы
        push bx      ; Заносим нуль в стек - это признак конца строки
        cmp  ax, 0   ; Сравниваем выводимое число с нулём
        jnl  oi_get  ; ...если не отрицательное, то переходим к вычислениям
        inc  bl      ; Устанавливаем признак отрицательного числа
        neg  ax      ; Само число делаем положительным
oi_get: xor  dx, dx  ; Обнуляем регистр DX перед делением, т.к. AX >= 0
        idiv cx      ; Делим (DX,AX) на CX, т.е. на 10
        or   dl, 30h ; Преобразуем цифру в код символа, добавляя код 30h
        push dx      ; Заносим очередной символ в стек
        cmp  ax, 0   ; Сравниваем результат деления с нулём
        jne  oi_get  ; ...если результат не нуль, то продолжаем делить
        ; Текст числа получен, выводим его на экран
        mov  ah, 6   ; Заносим код 6 для команды вывода символа на экран
        cmp  bl, 0   ; Сравниваем признак отрицательного числа с нулём
        je   oi_put  ; ...если признака нет, то переходим к выводу числа
        mov  dx, '-' ; Помещаем код символа минус в регистр
oi_nxt: int  21h     ; Вызываем обработку команды 6 - вывода символа
oi_put: pop  dx      ; Вытаскиваем очередной код символа из стека
        cmp  dx, 0   ; Сравниваем символ с признаком конца строки - с нулём
        jne  oi_nxt  ; ...если символ не нулевой, то продолжаем цикл
        ret
 outint endp
```

```
   start proc far
         push ds
         push ax
         mov  bx, dseg
         mov  ds, bx

         lea  dx, str_result ; Эквивалент - "mov dx, offset str_result"
         mov  ah, 9
         int  21h
         call foreach
         ret
   start endp
         ;
    cseg ends
         end start
```

Результат работы программы

```
f(x) = result
 -----------
f(-2) = 29
f(-1) = 17
f(0) = -5
f(1) = 3
f(2) = 3
f(3) = 3
f(4) = 3
f(5) = 3
```