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 *№3*.

Блочная структура программы
(наименование работы)

и линейное программирование на языке Assembler

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: Освоить программирование блочной структуры программы на Ассемблере. Освоить сборку исходного кода в исполняемый файл, используя программный пакет TASM.

Задание

1. Ознакомиться с образцами программ.

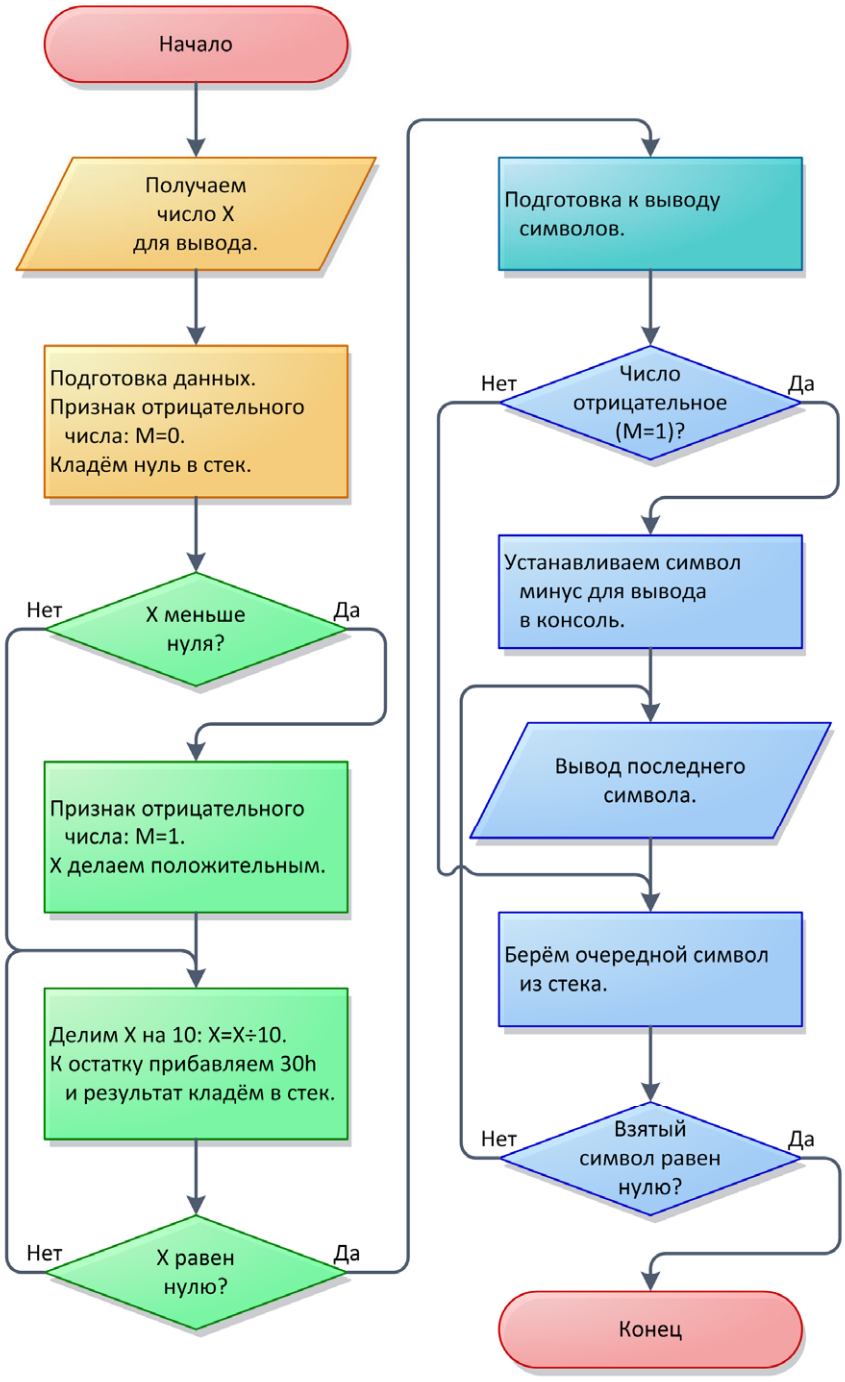
2. Разработать программу на Ассемблере для реализации линейного процесса, аналогичного в лабораторной работе №1.

Расчётная формула: $c \times d + (a \bmod b) + (a^2 - 7)$.

Аргументы: $a = 29$, $b = 18$, $c = 31$, $d = -14$.

Блок-схема

Блок-схема процедуры вывода целого числа в консоль:



Код программы реализации линейного процесса

```
  sseg segment stack
       db 64 dup (?)
  sseg ends
       ;
  dseg segment
       str_result db "Result: $"
       value_a dw  29
       value_b dw  18
       value_c dw  31
       value_d dw -14
  dseg ends
       ;
  cseg segment
       assume cs:cseg, ds:dseg, ss:sseg
       ;
 start proc far
       push ds
       push ax
       mov  bx, dseg
       mov  ds, bx

       call outtxt
       call calc
       call outint
       ret
 start endp
       ; Процедура вывода текста "Result: ". Процедура ломает AH, DX.
outtxt proc near
       lea  dx, str_result ; Эквивалент - "mov dx, offset str_result"
       mov  ah, 9
       int  21h
       ret
outtxt endp
       ;
       ; Процедура вычисления формулы: c*d + (a mod b) + (a^2 - 7).
       ; Процедура ломает AX, CX, DX.
  calc proc near
       mov  ax, value_c ; Заносим значение C в регистр AX
       imul value_d     ; Умножаем AX на значение D
       mov  cx, ax      ; Копируем результат в регистр CX

       mov  ax, value_a ; Заносим значение A в регистр AX
       cwd              ; Расширяем значение AX до (DX,AX)
       idiv value_b     ; Делим (DX,AX) на значение B
       add  cx, dx      ; Прибавляем остаток от деления к сумме в CX

       mov  ax, value_a ; Заносим значение A в регистр AX
       imul ax          ; Умножаем AX само на себя
       sub  ax, 7       ; Вычитаем из результата (AX) число 7
       add  ax, cx      ; Складываем CX и AX, результат заносим в AX
       ret
  calc 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

        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

   cseg ends
        end start
```

Результат работы программы

```
Result: 411
```