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 *№2*.

Команды передачи управления и организации циклов
(наименование работы)

на языке программирования Assembler

Вариант задания *№2*.

Лабораторную работу выполнили:

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

Лабораторную работу принял:

зам. декана фак. №3, Алещенко Алла Степановна _____.
(должность) *(Ф. И. О.)* *(подпись)*

«*10*» *ноября* 2009 г.
(дата приёма)

Цель работы: Освоить нелинейное программирование с использованием условных и безусловных команд перехода, а также команд организации циклов.

Задание

Разработать программу реализации циклического процесса вычисления выражения:

$$Y = \begin{cases} 2ax + 5, & x < 0; \\ \frac{a-b}{d}, & x = 0; \\ \frac{a^2-x}{c+d}, & x > 0. \end{cases}$$

$$a = -6;$$
$$b = 4;$$
$$c = 8;$$
$$d = 2;$$
$$x = [-2, \quad 5].$$

При этом деление используется целочисленное.

Код программы реализации нелинейного процесса

```
program Laba2;

var
  a, b, c, d, xa, xb, x: integer;
  r_asm, r_pas : array [0..10] of integer;

label
  start, next, positive, zero, negative;

begin
  a :=  -6;
  b :=   4;
  c :=   8;
  d :=   2;
  xa := -2;
  xb :=  5;

  for x := xa to xb do
    if      x < 0 then r_pas[x - xa] := 2*a*x + 5
    else if x > 0 then r_pas[x - xa] := (a*a - x) div (c + d)
    else               r_pas[x - xa] := (a - b) div d;

  WriteLn;
  Write('     x:');
  for x := xa to xb do
    Write(x:5);
  Write('    -');

  WriteLn;
  Write('pascal:');
  for x := 0 to xb - xa + 1 do
    Write(r_pas[x]:5);
```

```
  asm
    mov  AX, OFFSET r_asm {кладём адрес начала результирующего массива в AX}
    push AX     {сохраняем адрес результирующего массива в стеке (в BP-4)}
    mov  CX, xb {кладём конец промежутка X в регистр CX}
    sub  CX, xa {вычитаем из конца промежутка начало промежутка}
    inc  CX     {прибавляем к CX единицу для получения кол-ва проходов цикла}
    mov  AX, xb {кладём конец промежутка X в регистр AX}
    inc  AX     {добавляем к AX единицу и получаем смещение начала отсчёта...}
                {...относительно счётчика шагов цикла}
    push AX     {сохраняем смещение начала отсчёта в стеке (в BP-6)}

  start:
    mov  BX, [BP-6] {копируем из стека значение смещения начала отсчёта в стеке}
    sub  BX, CX {вычитаем из смещения текущий шаг цикла и получаем текущий X}
    jb negative {если результат вычитания (т.е. X) < 0, то переходим по метке}
    je zero     {если результат вычитания равен нулю, то переходим по метке}
                {результат вычитания больше нуля, выполняем код ниже}
  positive:
    mov  AX, a  {помещаем в регистр AX значение переменной A}
    imul AX     {умножаем значение регистра AX само на себя}
    sub  AX, BX {вычитаем из произведения текущее значение X (в регистре BX)}
    sbb  DX, 0  {продолжаем вычитание (DX:AX) - (0:BX)}
    mov  BX, c  {помещаем в регистр BX значение переменной C}
    add  BX, d  {прибавляем к значению регистра BX значение переменной D}
    idiv BX     {получаем результат в регистре AX: AX = (DX:AX) / BX}
    jmp  next   {безусловно переходим по метке сохранения результата и далее}
  zero:
    mov  AX, a  {помещаем в регистр AX значение переменной A}
    sub  AX, b  {вычитаем из значения регистра AX значение переменной B}
    cwd         {расширяем слово AX до двойного слова DX:AX}
    idiv d      {получаем результат в регистре AX: AX = DX:AX / D}
    jmp  next   {безусловно переходим по метке сохранения результата и далее}
  negative:
    mov  AX, a  {помещаем в регистр AX значение переменной A}
    imul BX     {умножаем значение регистра AX на X (в BX): (DX:AX) = AX * BX}
    mov  BX, 2  {помещаем в регистр BX число 2}
    imul BX     {умножаем значение регистра AX на 2 (в BX): (DX:AX) = AX * BX}
    add  AX, 5  {получаем результат в регистре AX: AX = AX + 5}

  next:
    mov  BX, [BP-4] {копируем из стека адрес результирующего массива}
    mov  [BX], AX   {копируем текущий результат по текущему адресу в массиве}
    add  BX, 2      {сдвигаем указатель (адрес) массива на одно слово}
    mov  [BP-4], BX {копируем в стек сдвинутый адрес результирующего массива}
    loop start      {если обработаны не все шаги цикла, то переходим по метке}
  end;

  WriteLn;
  Write('   asm:');
  for x := 0 to xb - xa + 1 do
    Write(r_asm[x]:5);

  ReadLn;
end.
```

Результат работы программы

```
     x:   -2   -1    0    1    2    3    4    5    -
pascal:   29   17   -5    3    3    3    3    3    0
   asm:   29   17   -5    3    3    3    3    3    0
```