*Федеральное государственное бюджетное образовательное учреждение* *высшего образования*

|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | ***«Московский государственный технический университет  имени Н.Э. Баумана***  ***(национальный исследовательский университет)»***  ***(МГТУ им. Н.Э. Баумана)*** |

ФАКУЛЬТЕТ \_ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ\_\_\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА \_\_\_\_КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Отчет**

**по лабораторной работе № \_3\_\_\_\_**

**Дисциплина: \_**Машинно-зависимые языки и основы компиляции**\_\_\_\_**

**Название лабораторной работы: \_**Программирование ветвлений и циклов**\_**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студент гр. **\_**ИУ6-42б**\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_**И.С. Марчук**\_\_\_**

(Подпись, дата) (И.О. Фамилия)

Преподаватель  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(Подпись, дата) (И.О. Фамилия)

Москва, 2021

**Задание:**

Разработать программу на языке ассемблера, которая вычисляет заданное выражение (Рис. 1).

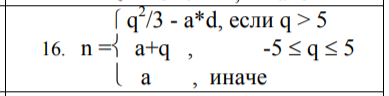
****

Рисунок 1 – Задание для выполнения

**Цель работы:**

Изучение средств и приемов программирования ветвлений и циклов на языке ассемблера.

**Схема алгоритма для решения поставленной задачи (рисунок 2):**

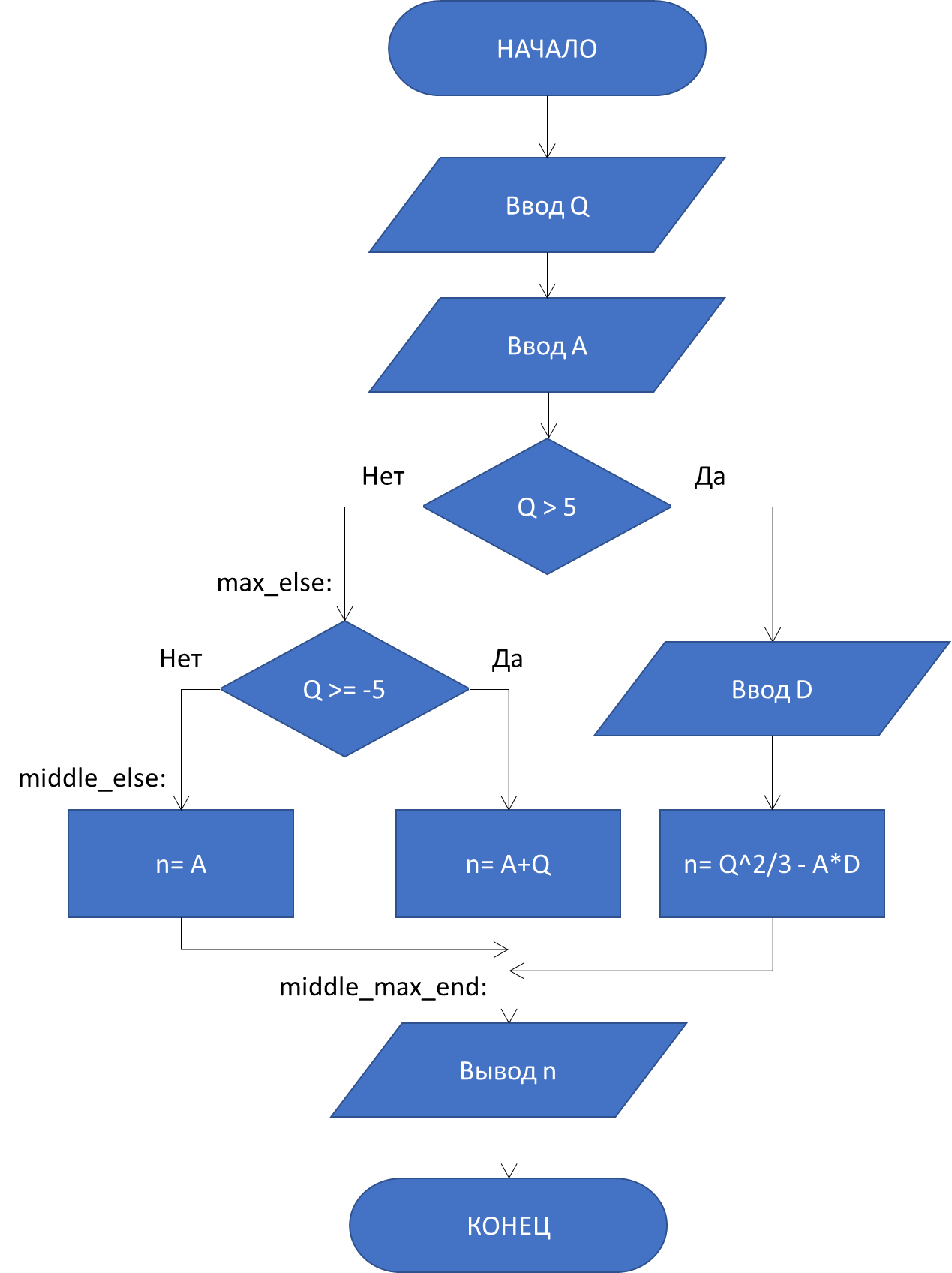


Рисунок 2 – Схема алгоритма

**Текст программы с комментариями:**

**.586** ; подключение набора команд Реntium

**.MODEL flat, stdcall** ; модель памяти и

; конвенция о передаче параметров

**OPTION CASEMAP:NONE** ; опция различия строчных

; и прописных букв

**Include kernel32.inc** ; подключение описаний процедур и

**Include masm32.inc** ; констант

**IncludeLib kernel32.lib** ; подключение библиотек

**IncludeLib masm32.lib**

**.CONST**

**msgEnterQ DB "Please enter q:",0**

**msgEnterA DB "Please enter a:",0**

**msgEnterD DB "Please enter d:",0**

**msgAnswer DB "n= ",0**

**msgExit DB 0AH,0DH,0AH,0DH,"Press Enter to Exit",0AH,0DH,0**

**.DATA**

**.DATA?**

**inbuf DB 100 DUP (?)**

**myInBuffer DB 9 DUP (?)**

**vQ DWORD ?**

**vA DWORD ?**

**vD DWORD ?**

**ans DWORD ?**

**.CODE**

**Start:**

; ввод q

**Invoke StdOut, ADDR msgEnterQ**

**Invoke StdIn, ADDR myInBuffer, LengthOf myInBuffer**

**Invoke StripLF, ADDR myInBuffer** ; Преобразование в SDWORD c нулем на конце

**Invoke atol, ADDR myInBuffer** ; преобразовываем строку с нулем в число, а результат в EAX

**MOV vQ, EAX**

; ввод A

**Invoke StdOut, ADDR msgEnterA**

**Invoke StdIn, ADDR myInBuffer, LengthOf myInBuffer**

**Invoke StripLF, ADDR myInBuffer**

**Invoke atol, ADDR myInBuffer**

**MOV vA, EAX**

**CMP vQ, 5** ; Сравниваем q с 5

**JNG max\_else** ; если q меньше или равно 5 переходим к метке

; ввод D

**Invoke StdOut, ADDR msgEnterD**

**Invoke StdIn, ADDR myInBuffer, LengthOf myInBuffer**

**Invoke StripLF, ADDR myInBuffer**

**Invoke atol, ADDR myInBuffer**

**MOV vD, EAX**

**MOV EAX, vQ** ; считаем ответ для q > 5 (Q^2/3 - a\*d)

**MUL EAX**

**CDQ** ; двойное слово в учетверенное EAX -> EDX:EAX

**MOV ECX, 3**

**DIV ECX**

**MOV ans, EAX**

**MOV EAX, vA**

**MUL vD**

**SUB ans, EAX**

**JMP middle\_max\_end** ; отправляемся в конец условия middle\_max\_end

**max\_else:**

**CMP vQ, -5**  ; Сравниваем q с -5

**JL middle\_else** ; Если q строго меньше -5 переходим к метке middle\_else

**MOV EAX, vA** ; считаем ответ для -5 >= q >= 5 (a+q)

**ADD EAX, vQ**

**MOV ans, EAX**

**JMP middle\_max\_end** ; переходим к концу словия middle\_max\_end

**middle\_else:**

**MOV EAX, vA** ; считаем ответ для q < -5 (a)

**MOV ans, EAX**

**middle\_max\_end:**

; выводим ответ

**Invoke StdOut, ADDR msgAnswer** ; вывод описания ответа

**Invoke dwtoa, ans, ADDR myInBuffer** ; прреодразование ответа-числа в строку

**Invoke StdOut, ADDR myInBuffer** ; вывод

**XOR EAX,EAX**

**Invoke StdOut,ADDR msgExit**

**Invoke StdIn,ADDR inbuf,LengthOf inbuf**

**Invoke ExitProcess,0**

**End Start** ; конец модуля

**Работа программы с тестовыми данными приведена в таблице 1:**

|  |  |  |
| --- | --- | --- |
| Исходные данные | Ожидаемый результат | Полученный результат |
| q=30 a=25 d=10 | n=30^2/3 – 25\*10 = 50 | n=50 |
| q=50 a=20 d=100 | n=50^2/3-20\*100= -1167 | n=-1167 |
| q=1 a=-10 d=3 | n=-10+1 = -9 | n=-9 |
| q=-1000000 a=30000000 | n=30000000 | n= 30000000 |
| q=4 a=30000000 d=10 | n=30000004 | n=30000004 |
| q=40 a=10 d=30000000 | -299999466.66667 | n=-299999467 |
| q=40 a=-30000000 d=10 | 300000533.333333 | n=300000533 |
| q=40 a=10 d=-30000000 | 300000533.333333 | n=300000533 |
| q=40 a=-30000000 d=-10 | -299999466.66667 | n=-299999466 |
| q=40 a=-10 d=-30000000 | -299999466.66667 | n=-299999466 |

Таблица 1 – Отладка программы

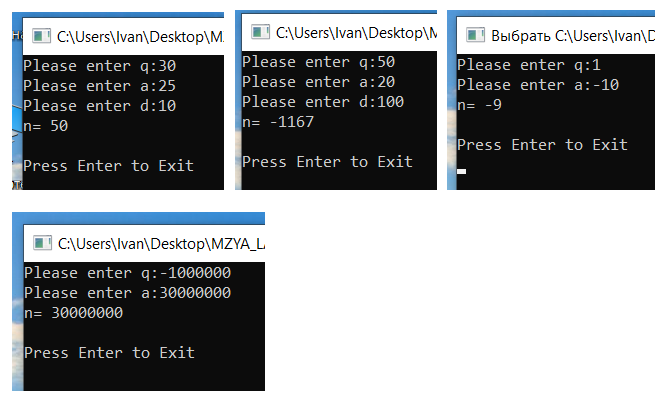


Рисунок 3 – Вывод программы при различных входных данных

**Контрольные вопросы:**

1) Какие машинные команды используют при программировании ветвлений и циклов?

Используют команды условного и безусловного переходов. Для счетных циклов используют команду LOOP работающую с счетчиком ECX

2) Выделите в своей программе фрагмент, реализующий ветвление. Каково назначение каждой машинной команды фрагмента?

- Начало первого условия (max):

**CMP vQ, 5** ; Сравниваем q с 5

**JNG max\_else** ; если q меньше или равно 5 переходим к метке max\_else

- В его ветке else находится еще одно условие (middle)

**CMP vQ, -5**  ; Сравниваем q с -5

**JL middle\_else** ; Если q строго меньше -5 переходим к метке middle\_else

У обоих условий есть метка к которой они переходят после выполнения первой части команд, чтобы не начала выполняться ветка else эта метка называется **middle\_max\_end**

3) Чем вызвана необходимость использования команд безусловной передачи управления?

Как и писалось выше, если не вызывать безусловный переход после того как выполнилась основная ветка, то начнет в противоречии с условием выполняться ветка else.

4) Поясните последовательность команд, выполняющих операции ввода-вывода в вашей программе. Чем вызвана сложность преобразований данных при выполнении операций ввода-вывода?

**Invoke StdOut, ADDR msgEnterQ**; вывод пригласительного сообщения

**Invoke StdIn, ADDR myInBuffer, LengthOf myInBuffer**

**Invoke StripLF, ADDR myInBuffer** ; Преобразование в SDWORD c нулем на конце

**Invoke atol, ADDR myInBuffer** ; преобразовываем строку с нулем в число

Сначала ввод пользователя нужно преобразовать к хоть какого-то вида данным, стандартная библиотека преобразует его в строки. После этого, если необходимо выделить из ввода число, задействуется стандартная функция atol, которая преобразует строку в число.

**Вывод:**

Я изучил средства и приемы программирования ветвлений и циклов на языке ассемблера, запрограммировал поставленную задачу и протестировал все ветви алгоритма.