ГУАП

КАФЕДРА № 42

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ПРЕПОДАВАТЕЛЬ

| Ассистент |  |  |  | Д.О.Шевяков |
| --- | --- | --- | --- | --- |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

| | ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ | | --- | | **ПРЕДСТАВЛЕНИЕ ДАННЫХ.**  **АРИФМЕТИКО-ЛОГИЧЕСКИЕ ОПЕРАЦИИ**  Вариант 5 | |  | | по курсу: АРХИТЕКТУРА ЭВМ | |
| --- | --- | --- | --- | --- |

РАБОТУ ВЫПОЛНИЛ

| СТУДЕНТ ГР. № | 4128 |  |  |  | В.А. Воробьев |
| --- | --- | --- | --- | --- | --- |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2023

**СОДЕРЖАНИЕ**

[**1 Цель и постановка задачи** 3](#_3znysh7)

[**1.1 Цель работы** 3](#_2et92p0)

[**1.2 Задания** 3](#_tyjcwt)

[**2 Ход и результаты выполнения работы** 4](#_3dy6vkm)

[**2.1 Исходные данные** 4](#_1t3h5sf)

[**2.2 Листинг программы** 5](#_4d34og8)

[**3 Вывод**](#_2s8eyo1) 10

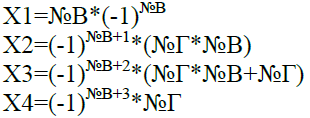
**1 Цель и постановка задачи**

**1.1 Цель работы**

Изучение архитектуры МП Intel 8086, изучение структуры простейшей ассемблерной программы, ознакомление с системой арифметико-логических команд процессора, организация вычисления на языке ассемблера.

**1.2 Задания**

1. Определить исходные данные в соответствии с номером варианта. Значения исходных данных, которые должны храниться в сегменте данных, определяются выражениями:



где №В – номер варианта, №Г – номер группы.

2. Перевести значения величин Х1-Х4 в шестнадцатеричную систему счисления.

3. Провести трассировку заданного алгоритма с использованием заданных исходных данных.

4. Составить программу заданного алгоритма в мнемокодах.

5. Оформить отчет по лабораторной работе.

6. В учебной лаборатории проверить результаты выполнения программы в программе-отладчике, сравнивая их с результатами ручной трассировки алгоритма.

**2 Ход и результаты выполнения работы**

**2.1 Исходные данные**

Исходя из заданных выражений для определения исходных данных, исходные данные в десятичной и шестнадцатеричной системах счисления получились следующими:

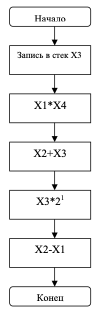
*E*

Рисунок 1 – Алгоритм программы в соответствии с вариантом

**2.2 Листинг программы**

SStack segment 'stack'

DB 256 DUP (?)

SStack ends

SData segment 'data'

X1 DW -5

X2 DW 160

X3 DW -40

X4 DW -20

RES1 DD ?

RES2 DD ?

RES3 DD ?

RES4 DD ?

RES1Text DW "X1 = $"

RES2Text DW "X2 = $"

RES3Text DW "X3 = $"

RES4Text DW "X4 = $"

SData ends

SCode segment 'code'

ASSUME CS:SCode, DS:SData, SS:SStack

PrintNum PROC NEAR

; print ASCII num

PUSH AX

PUSH BX

PUSH CX

PUSH DX

MOV BX, 10

XOR CX, CX

OR AX, AX

JNS @@DIV:

NEG AX

PUSH AX

MOV AH, 02H

MOV DL, '-'

INT 21H

POP AX

@@DIV:

XOR DX, DX

DIV BX

PUSH DX

INC CX

OR AX,AX

JNZ @@DIV

MOV AH, 02H

@@STORE:

POP DX

ADD DL, '0'

INT 21H

LOOP @@STORE

POP DX

POP CX

POP BX

POP AX

; Print new line

PUSH AX

MOV AH, 0EH

MOV AL, 0AH

INT 10H

MOV AH, 0EH

MOV AL, 0DH

INT 10H

POP AX

RET

PrintNum ENDP

Main proc FAR

MOV AX, SData

MOV DS, AX

; Calculations

PUSH X3

; X1 \* X4

MOV AX, X1

MUL X4

MOV RES1, AX

; X2 + X3

MOV AX, X2

ADD AX, X3

MOV RES2, AX

; X3 \* 2

MOV AX, 2

MUL X3

MOV RES3, AX

; X2 - X1

MOV AX, X2

SUB AX, X1

MOV RES4, AX

; Output

MOV AX, X3

CALL PrintNum

; RES1

MOV AH, 9h

LEA DX, RES1Text

INT 21H

MOV AX, RES1

CALL PrintNum

; RES2

MOV AH, 9h

LEA DX, RES2Text

INT 21H

MOV AX, RES2

CALL PrintNum

; RES3

MOV AH, 9h

LEA DX, RES3Text

INT 21H

MOV AX, RES3

CALL PrintNum

; RES4

MOV AH, 9h

LEA DX, RES4Text

INT 21H

MOV AX, RES4

CALL PrintNum

; End output

MOV AH, 0

INT 21H

ret

Main endp

SCode ends

END Main

**2.3 Таблица трассировки программы**

Таблица 1 – Таблица трассировки программы

| **Исходные данные** | |
| --- | --- |
| X1 | -5 |
| X2 | 160 |
| X3 | -40 |
| X4 | -20 |
| **Операция** | **Результат** |
| X3 | -40 |
| X1 \* X4 | 100 |
| X2 + X3 | 120 |
| X3 \* 2 | -80 |
| X2 + X1 | -165 |

**2.4 Результат работы программы**

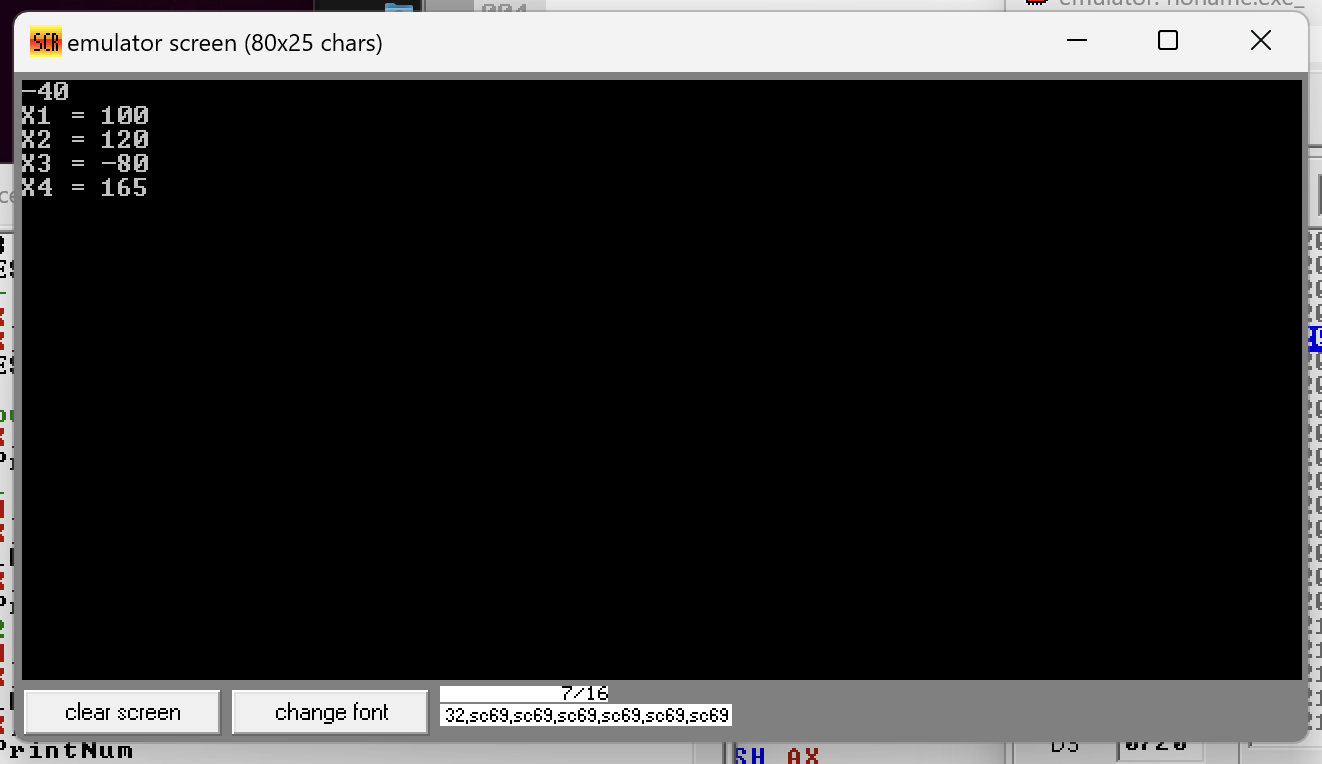
**

Рисунок 2 – Результат работы программы

**3 Вывод**

В ходе выполнения лабораторной работы были изучены: архитектура МП Intel 8086, структура простейшей ассемблерной программы, система арифметико-логических команда процессора и организация вычислений на языке ассемблера.

В результате был написан код на языке ассемблера Intel 8086 с помощью программы emu8086, выполняющий необходимые операции. Кроме того, дополнительно был реализован вывод результатов операций в консоль.