МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования «Вятский государственный университет»

Институт математики и информационных систем

Факультет автоматики и вычислительной техники

Кафедра радиоэлектронных средств

Отчет по лабораторной работе №3

по дисциплине «Цифровые устройства и микропроцессоры»

Вариант 12

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Выполнил: | студент группы  ИНБб-3301-02-00 |  |  | А.В. Кригер |
|  |  |  |  |  |
| Проверил: | преподаватель |  |  | М.А. Земцов |
|  |  |  |  |  |

Работа защищена с оценкой «\_\_\_\_\_\_\_\_\_\_\_\_\_» «\_\_\_»\_\_\_\_\_\_\_ 2025 г.

г. Киров

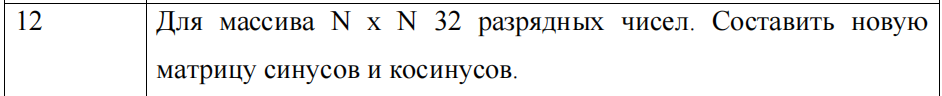
2025

**Цель работы:** изучение принципов выполнения арифметических

команд с помощью математического сопроцессора FPU микропроцессоров с

архитектурой x86.

**Задание:**



**Ход работы:**

Код программы с комментариями:

Блок кода Sourse.asm

1..686

2.model flat, stdcall

3.option casemap:none

4.includelib kernel32.lib

5.ExitProcess PROTO STDCALL :DWORD

6.GetProcessHeap PROTO STDCALL

7.HeapAlloc PROTO STDCALL :DWORD, :DWORD, :DWORD

8.HeapFree PROTO STDCALL :DWORD, :DWORD, :DWORD

9.stack 100h

10.data

11.array\_start dd 1.1, 0.3, 4.0

12. dd 5.0, 2.0, 0.9

13. dd 0.5, 0.1, 1.6 ; всего 9 элементов (3х3 матрица)

14.array\_sin\_ptr dd ?

15.array\_cos\_ptr dd ?

16.n dd 3

17.mysize dd 4

18.code

19.Start:

20. finit

21. ; Вычисляем размер: n \* n \* sizeof(float)

22. mov eax, n

23. imul eax, n

24. imul eax, mysize ; total\_size = n \* n \* 4

25. mov ecx, eax ; сохраняем в ecx

26. ; Получаем heap

27. INVOKE GetProcessHeap

28. mov ebx, eax ; сохраняем дескриптор heap-а в ebx

29. ; Выделяем память под синусы

30. INVOKE HeapAlloc, ebx, 0, ecx

31. test eax, eax

32. je allocation\_error

33. mov array\_sin\_ptr, eax

34. ; Выделяем память под косинусы

35. INVOKE HeapAlloc, ebx, 0, ecx

36. test eax, eax

37. je allocation\_error

38. mov array\_cos\_ptr, eax

39. ; i = 0

40. xor esi, esi ; esi = i

41.outer\_loop:

42. mov eax, n

43. cmp esi, eax

44. jge end\_outer\_loop ; if i >= n, break

45. ; j = 0

46. xor ecx, ecx ; ecx = j

47.inner\_loop:

48. mov eax, n

49. cmp ecx, eax

50. jge end\_inner\_loop ; if j >= n, break

51. ; offset = ((i \* n) + j) \* 4

52. mov eax, esi

53. imul eax, n ; i \* n

54. add eax, ecx ; i \* n + j

55. shl eax, 2 ; \* 4 (размер float)

56. mov edx, eax ; edx = смещение в байтах

57. fld array\_start[edx] ; загрузка значения в FPU

58. fld array\_start[edx] ; копия для второго вычисления

59. fsin

60. fstp [array\_sin\_ptr + edx] ; сохраняем результат sin

61. fcos

62. fstp [array\_cos\_ptr + edx] ; сохраняем результат cos

63. inc ecx

64. jmp inner\_loop

65.end\_inner\_loop:

66. inc esi

67. jmp outer\_loop

68.end\_outer\_loop:

69. INVOKE ExitProcess, 0

70.allocation\_error:

71. INVOKE ExitProcess, 1

72. END Start

Скриншоты работы программы представлены ниже на рисунках 1-3.

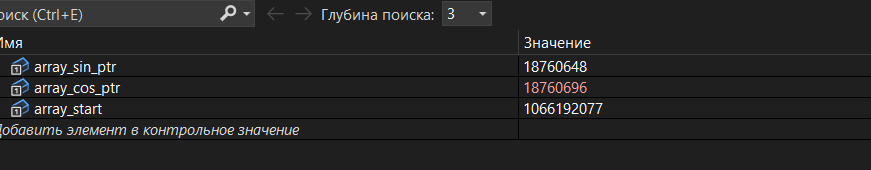


Рисунок 1 – вывод результата после первой итерации.

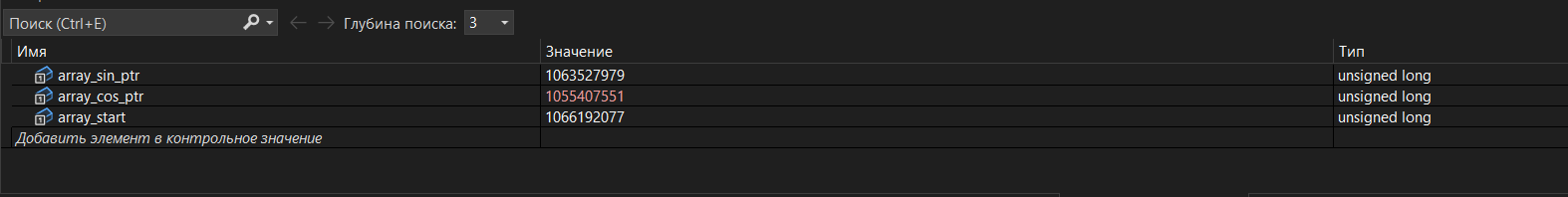


Рисунок 2 – вывод результата после второй итерации.

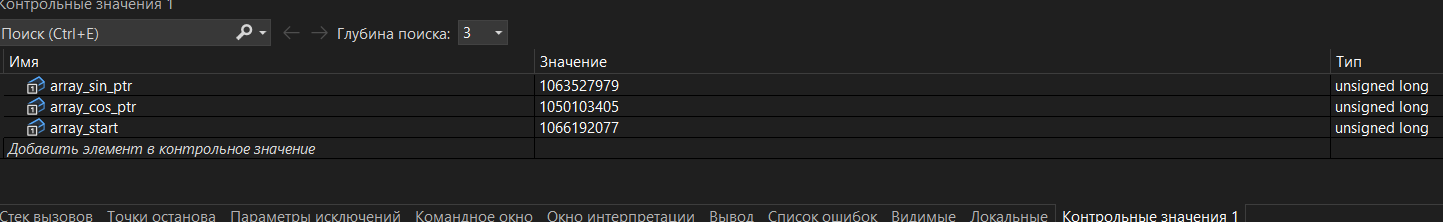


Рисунок 3 – вывод результата после третьей итерации.

**Вывод:** Выполнив данную лабораторную работу мы ознакомились с технологией применения языка ассемблера при разработке программного обеспечения на языках высокого уровня.