|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | ***«*Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«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\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ОТЧЕТ**

**по лабораторной работе № 4**

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

**Название лабораторной работы:** Обработки массивов и матриц



Студент гр. ИУ6-45Б **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_**И.А.Дулина**\_\_\_\_\_\_\_**

(Подпись, дата) (И.О. Фамилия)

Преподаватель  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_**\_С.С.Данилюк**\_\_\_\_\_**

(Подпись, дата) (И.О. Фамилия)

Москва, 2024

**Вариант 5.6**

**Цель работы:** изучение приемов моделирования обработки массивов и матриц в языке ассемблера.

**Задание**: Дана матрица 5x7. Каждый столбец с чётным номером матрицы перевернуть так, чтобы первый элемент стал последним, второй – предпоследним и т.д. Организовать ввод матрицы и вывод результатов.

На рисунке 1 показана схема алгоритма.



Рисунок 1 – Схема алгоритма

В листинге 1 представлен код программы по вводу матрицы и выводу матрицы построчно.

Листинг 1 – Код библиотеки по вводу и выводу матрицы

|  |
| --- |
| section .data:  probel db " ", 10; символ новой строки  InputMsg db "Введите числа строки матрицы", 10  lenInputMsg equ $-InputMsg  ErrorStr db "Error: Invalid input format", 10  lenError equ $-ErrorStr  RowMsg db "Row: ", 10  lenRowMsg equ $-RowMsg  section .bss ; Буфер для ввода чисел  NumberBuf resb 10  InBuf resb 10 ; буфер для вводимой строки  lenIn equ $-InBuf  OutBuf resb 50  lenOut equ $-OutBuf  section \_text  ReadMatrix:      push eax      push ebx      push ecx      push edx      push esi      push edi      mov esi, matr ; загрузка адреса матрицы в регистр esi      mov ecx, [matr\_rows] ; загрузка количества строк      mov edx, [matr\_columns] ; загрузка количества столбцов  .write\_matrix\_loop:      mov ebx, edx ; сохраняем количество столбцов в регистре ebx для использования в цикле      call .InputMessage  .write\_row\_loop:      ;read element      call .ReadElement      mov [esi], eax      add esi, 4 ; переход к следующему элементу матрицы (размер элемента - 4 байта)      dec ebx ; уменьшаем счетчик столбцов      cmp ebx, 0 ; проверяем, завершили ли мы обработку строки      jnz .write\_row\_loop ; если нет, продолжаем выводить элементы      loop .write\_matrix\_loop ; уменьшаем счетчик строк и продолжаем выводить      pop edi      pop esi      pop edx      pop ecx      pop ebx      pop eax      ret  .InputMessage:      push eax      push ebx      push ecx      push edx      mov eax, 4 ; системная функция 4 (write)      mov ebx, 1 ; дескриптор файла stdout=1      mov ecx, InputMsg ; адрес выводимой строки      mov edx, lenInputMsg ; длина выводимой строки      int 80h ; вызов системной функции      pop edx      pop ecx      pop ebx      pop eax      ret  .ReadElement:      push esi      push ebx      push ecx      push edx      call .Buffer      mov esi, InBuf      call StrToInt      cmp ebx, 0      jne Error      pop edx      pop ecx      pop ebx      pop esi      ret  .Buffer:      mov eax, 3 ; системная функция 3 (read)      mov ebx, 0 ; дескриптор файла stdin=0      mov ecx, InBuf ; адрес буфера ввода      mov edx, lenIn ; размер буфера      int 80h      ret  WriteMatrix:      push eax      push ebx      push ecx      push edx      push esi      push edi      mov esi, matr ; загрузка адреса матрицы в регистр esi      mov ecx, [matr\_rows] ; загрузка количества строк      mov edx, [matr\_columns] ; загрузка количества столбцов  .print\_matrix\_loop:      mov edi, OutBuf ; установка указателя на начало буфера для записи элементов      mov ebx, edx ; сохраняем количество столбцов в регистре ebx для использования в цикле  .print\_row\_loop:      mov eax, [esi] ; загрузка элемента матрицы в eax      call .Output      add esi, 4 ; переход к следующему элементу матрицы (размер элемента - 4 байта)      dec ebx ; уменьшаем счетчик столбцов      cmp ebx, 0 ; проверяем, завершили ли мы обработку строки      jnz .print\_row\_loop ; если нет, продолжаем выводить элементы      add edi, 1 ;до этого мы перекрывали маркер конца строки, когда все элементы однйо строки выведены в буфер добавляем его      call .NewLineOutput ; выводим символ новой строки      loop .print\_matrix\_loop ; уменьшаем счетчик строк и продолжаем выводить      pop edi      pop esi      pop edx      pop ecx      pop ebx      pop eax      ret  .StringOutput:      push eax      push ebx      push ecx      push edx      mov eax, 4 ; системная функция 4 (write)      mov ebx, 1 ; дескриптор файла stdout=1      mov ecx, RowMsg ; адрес выводимой строки      mov edx, lenRowMsg ; длина строки      int 80h ; вызов системной функции      pop edx      pop ecx      pop ebx      pop eax      ret  .Output:      push esi      push eax      push ebx      push ecx      push edx      mov esi, NumberBuf ; установка указателя на начало буфера вывода      call IntToStr ; вызов функции для преобразования числа в строку      mov ebx, dword[esi]      mov [edi], ebx ; сохраняем строку в буфере      add edi, eax ; перемещаем указатель на следующий элемент по размеру строки      sub edi, 1 ;не учитываем маркер конца строки      mov eax, [probel] ;добавляем пробел между элементами      mov [edi], eax      add edi, 1 ;перемещаемся на 1 байт, так как добвили пробел и в будущем его не перекрывать      pop edx      pop ecx      pop ebx      pop eax      pop esi      ret  .NewLineOutput:      push eax      push ebx      push ecx      push edx      mov eax, 4 ; системная функция 4 (write)      mov ebx, 1 ; дескриптор файла stdout=1      mov ecx, OutBuf ; адрес буфера      mov edx, edi ; длина буфера (количество записанных элементов)      sub edx, OutBuf ; вычисляем длину буфера в байтах      int 80h ; вызов системной функции для вывода строки      pop edx      pop ecx      pop ebx      pop eax      ret  Error:      mov eax, 4 ; системная функция 4 (write)      mov ebx, 1 ; дескриптор файла stdout=1      mov ecx, ErrorStr ; адрес сообщения об ошибке      mov edx, lenError ; длина сообщения об ошибке      int 80h ; вызов системной функции      jmp Exit  Exit:      mov eax, 1 ; системная функция 1 (exit)      xor ebx, ebx ; код возврата 0      int 80h ; вызов системной функции  %include "../lib.asm" |

В листинге 2 представлен основной код программы.

Листинг 2 – код основной программы

|  |
| --- |
| section .data  ; matr dd 2,3,1,-1,8,9,-2  ; dd 6,-8,7,5,4,11,2  ; dd 12,3,0,-13,5,7,2  ; dd 6,23,103,4,5,-1,0  ; dd 0,-1,4,6,12,5,-6  matr\_rows dd 5 ;количество строк  matr\_columns dd 7 ;количество столбцов  StartMsg db "Изначальная матрица:", 10  lenStartMsg equ $-StartMsg  EndMsg db "Измененная матрица:", 10  lenEndMsg equ $-EndMsg  section .bss  matr resd 35;резервируем место для 35 4-байтовых чисел матрицы 5х7  section .text  global \_start  \_start:  ;ввод матрицы  call ReadMatrix  ;вывод изн матрицы      mov eax, 4 ; системная функция 4 (write)      mov ebx, 1 ; дескриптор файла stdout=1      mov ecx, StartMsg ; адрес выводимой строки      mov edx, lenStartMsg ; длина выводимой строки      int 80h ; вызов системной функции      call WriteMatrix  ;основная программа  mov ecx, [matr\_columns] ;количество  cycle\_colomns:      mov eax, ecx ;смещение по столбцу      dec eax      ;умножение ebx на 4      mov ebx, 4      mul ebx      mov ebx, eax      ;проверка чётности:      test ecx, 1      jnz next ;если столбец нечетный      push ecx      mov eax, [matr\_rows]      ;деление eax на 2      mov esi, 2      div esi      mov ecx, eax ;записываем только половину матрицы      mov esi, 1;счётчик      .cycle\_rows:          mov eax, [matr + ebx]          call Arifmetic          mov edx, [matr + ebx + edi]          mov [matr + ebx], edx          mov [matr + ebx + edi], eax          inc esi          add ebx, 28 ;переход к след элементу в столбце 7\*4байта          loop .cycle\_rows      pop ecx      next:      loop cycle\_colomns  ; вывод конечной матрицы      mov eax, 4 ; системная функция 4 (write)      mov ebx, 1 ; дескриптор файла stdout=1      mov ecx, EndMsg ; адрес выводимой строки      mov edx, lenEndMsg ; длина выводимой строки      int 80h ; вызов системной функции      call WriteMatrix  ;выход  call Exit  Arifmetic:      ;была выверена следующая зависимость:      ;номер противоположного элемента вычисляется так:      ;номер элемента + количество столбцов\*(кол-во строк +1 - 2\*номер строки)      push eax      push ebx      push esi      mov eax, esi      mov ebx, 2      mul ebx ; esi - номер строки, начиная с 1      mov esi, eax      mov eax, [matr\_rows] ;кол-во строк      inc eax ;кол-во строк+1      sub eax, esi ;кол-во строк+1 - 2\*номер строки      mul dword[matr\_columns] ;кол-во столбцов \* рез-тат разности      mov ebx, 4      mul ebx      mov edi, eax ; сохраняем результат произведения, номер элемента добавим в осн программе      pop esi      pop ebx      pop eax      ret  %include "matrixLib.asm" |

В таблице 1 представлены тесты.

Таблица 1 – Тесты программы

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № | Исходные данные | | | | | | | Ожидаемый результат | | | | | | | Полученный результат | | | | | | |
| 1 | 5 | **3** | 6 | **-3** | 12 | **0** | 103 | 5 | **11** | 6 | **1** | 12 | **4** | 103 | 5 | **11** | 6 | **1** | 12 | **4** | 103 |
| 4 | **1** | -7 | **12** | 11 | **5** | 23 | 4 | **-10** | -7 | **14** | 11 | **8** | 23 | 4 | **-10** | -7 | **14** | 11 | **8** | 23 |
| 7 | **-15** | 3 | **-4** | 8 | **7** | 5 | 7 | **-15** | 3 | **-4** | 8 | **7** | 5 | 7 | **-15** | 3 | **-4** | 8 | **7** | 5 |
| 1 | **-10** | 34 | **14** | 7 | **8** | 4 | 1 | **1** | 34 | **12** | 7 | **5** | 4 | 1 | **1** | 34 | **12** | 7 | **5** | 4 |
| 23 | **11** | 56 | **1** | -8 | **4** | -33 | 23 | **3** | 56 | **-3** | -8 | **0** | -33 | 23 | **3** | 56 | **-3** | -8 | **0** | -33 |
| 2 | 1 | **8** | 99 | **23** | -14 | **6** | 0 | 1 | **8** | 99 | **23** | -14 | **6** | 0 | 1 | **8** | 99 | **23** | -14 | **6** | 0 |
| 2 | **-9** | 12 | **-6** | 43 | **47** | 89 | 2 | **-9** | 12 | **-6** | 43 | **47** | 89 | 2 | **-9** | 12 | **-6** | 43 | **47** | 89 |
| -4 | **65** | -30 | **2** | 25 | **5** | -4 | -4 | **65** | -30 | **2** | 25 | **5** | -4 | -4 | **65** | -30 | **2** | 25 | **5** | -4 |
| 45 | **37** | 2 | **-1** | 0 | **-3** | 2 | 45 | **37** | 2 | **-1** | 0 | **-3** | 2 | 45 | **37** | 2 | **-1** | 0 | **-3** | 2 |
| 3 | **5** | 6 | **12** | 11 | **10** | -1 | 3 | **5** | 6 | **12** | 11 | **10** | -1 | 3 | **5** | 6 | **12** | 11 | **10** | -1 |
| 3 | 4 | **2** | 7 | **45** | 23 | **11** | 10 | 4 | **3** | 7 | **0** | 23 | **5** | 10 | 4 | **3** | 7 | **0** | 23 | **5** | 10 |
| 3 | **-3** | 4 | **7** | 56 | **-24** | 20 | 3 | **12** | 4 | **2** | 56 | **3** | 20 | 3 | **12** | 4 | **2** | 56 | **3** | 20 |
| -11 | **3** | 35 | **22** | 27 | **-100** | -59 | -11 | **3** | 35 | **22** | 27 | **-100** | -59 | -11 | **3** | 35 | **22** | 27 | **-100** | -59 |
| 56 | **12** | 15 | **2** | 5 | **3** | 1 | 56 | **-3** | 15 | **7** | 5 | **-24** | 1 | 56 | **-3** | 15 | **7** | 5 | **-24** | 1 |
| 0 | **3** | 0 | **0** | -4 | **5** | 7 | 0 | **2** | 0 | **45** | -4 | **11** | 7 | 0 | **2** | 0 | **45** | -4 | **11** | 7 |

На рисунках 2, 3 и 4 представлен вывод тестов.

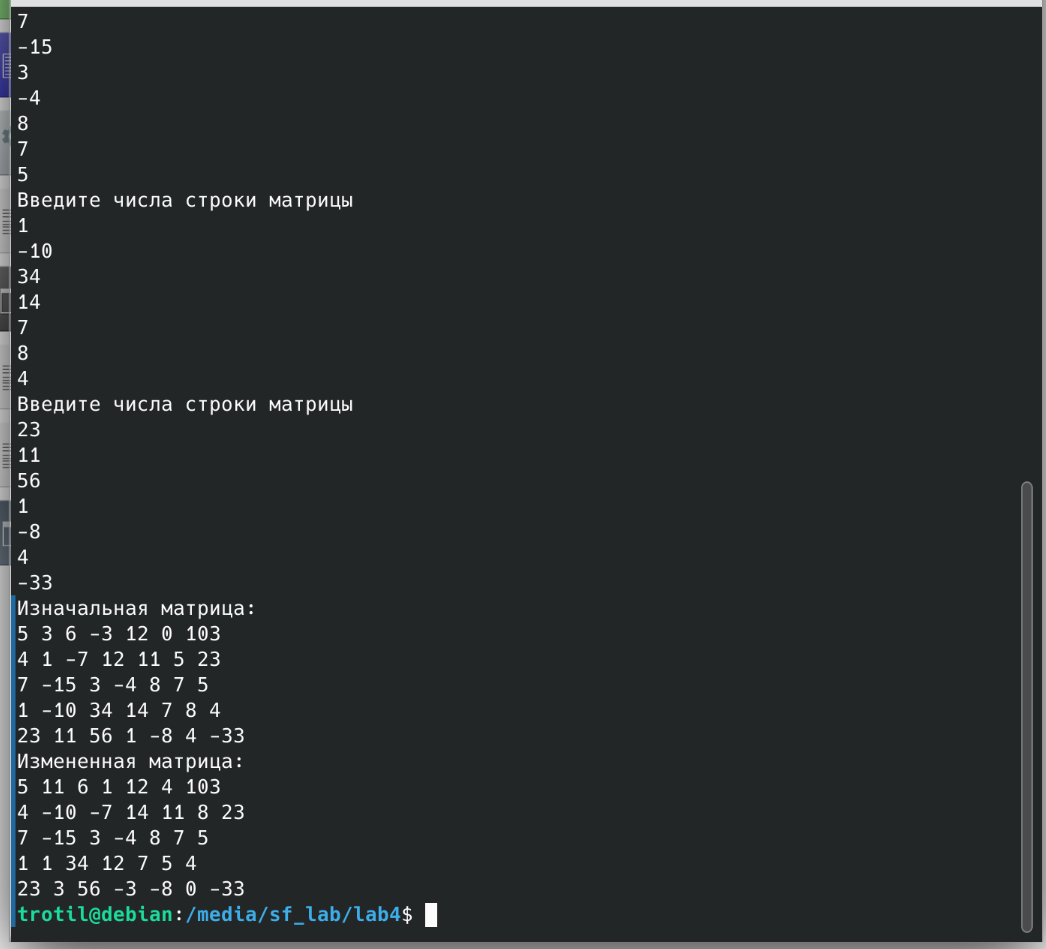


Рисунок 2 – Первый пример работы программы

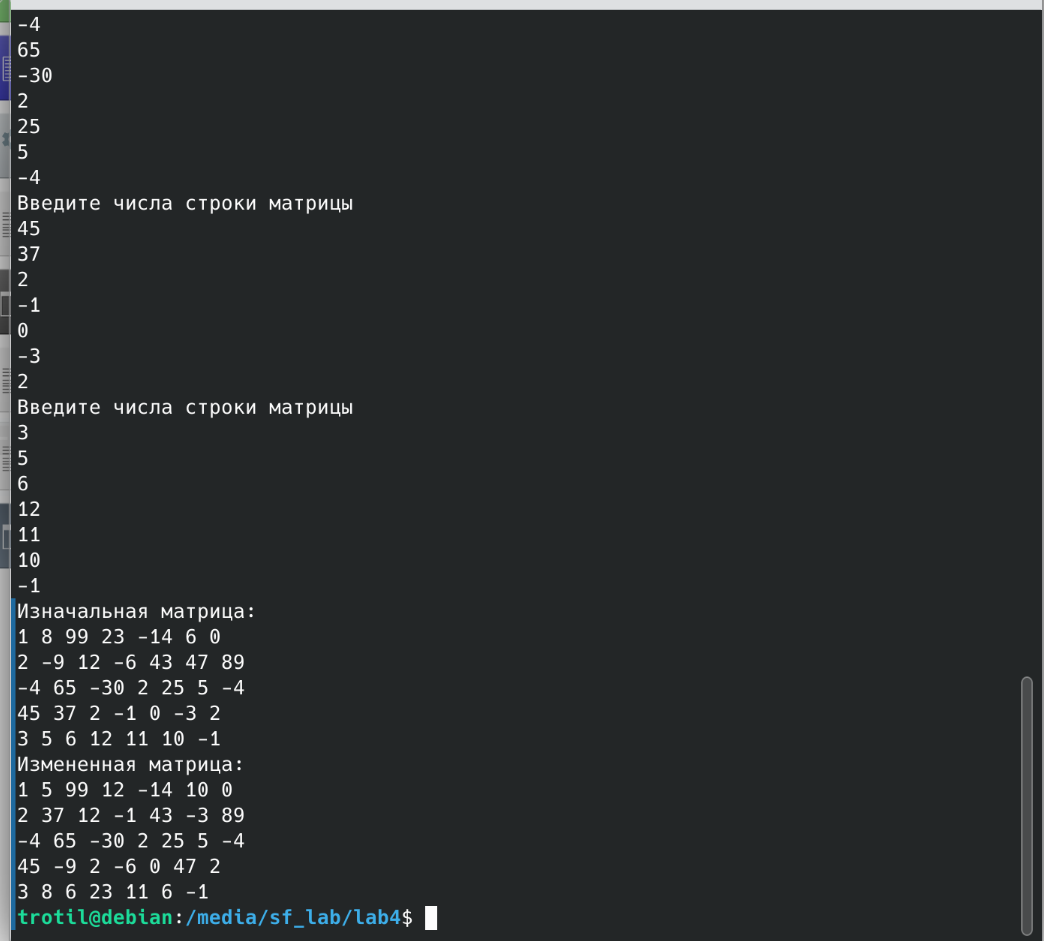


Рисунок 3 – Второй пример работы программы

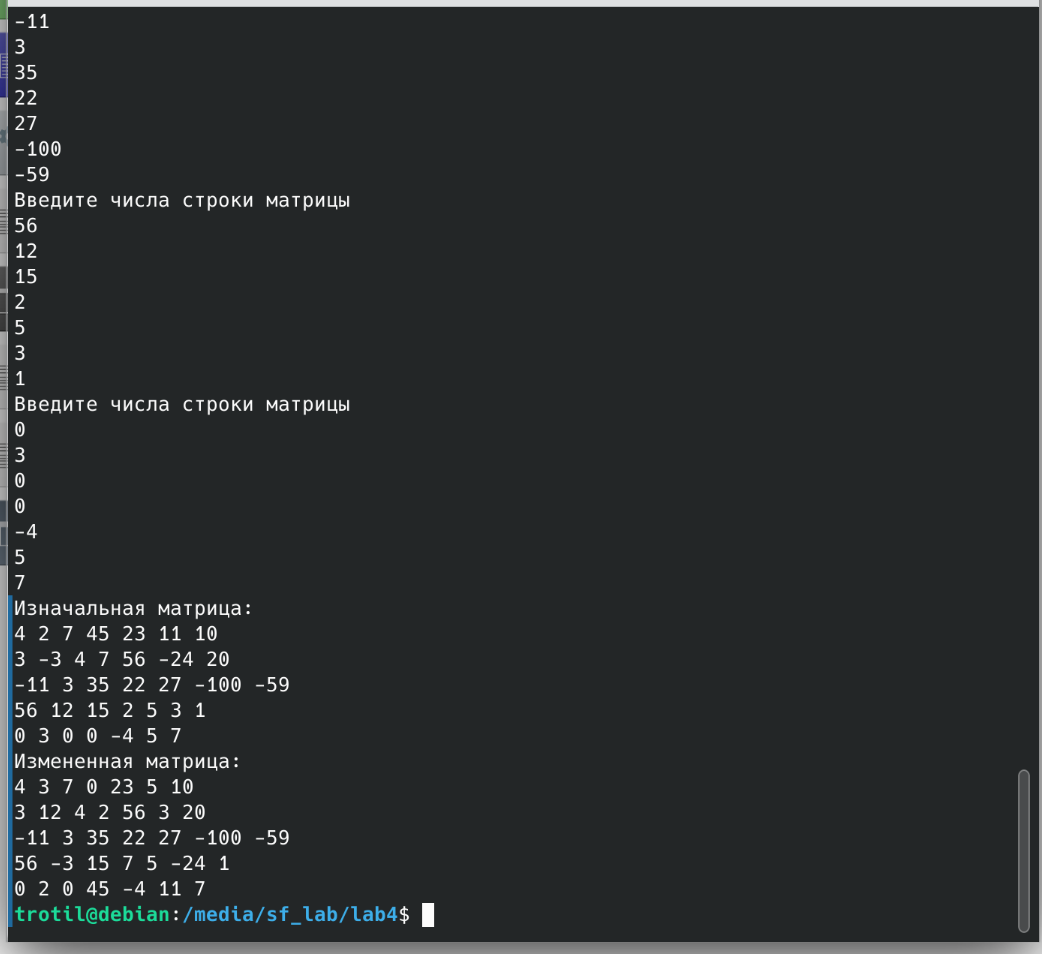


Рисунок 4 – Третий пример работы программы

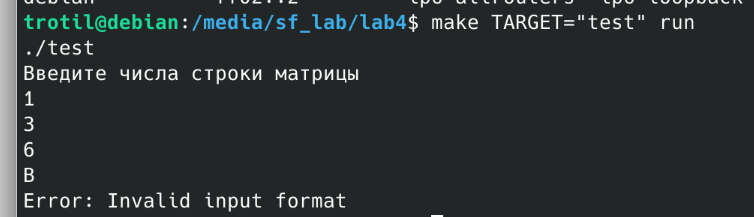


Рисунок 5 – пример ввода некорректных данных

**Вывод**: были изучены алгоритмы обработки массивов и матриц в языке ассемблера, была создана программа по вводу и выводу матриц, а также перемещению элементов местами.

**Контрольные вопросы**:

1. Почему в ассемблере не определены понятия «массив», «матрица»?

Ассемблер — это низкоуровневый язык программирования, ориентированный на работу с памятью и регистрами процессора. Вместо терминов "массив" и "матрица" используются операции с памятью для доступа к данным, хранящимся в памяти последовательно или по структуре данных, определенной программистом.

1. Как в ассемблере моделируются массивы?

Путем резервирования блоков памяти определенного размера, которые могут содержать последовательность значений. Эти значения хранятся в памяти последовательно, и доступ к ним осуществляется через индексы или смещения.

1. Поясните фрагмент последовательной адресации элементов массива? Почему при этом для хранения частей адреса используют регистры?

При последовательной адресации элементов массива в ассемблере адреса каждого последующего элемента вычисляются путем инкрементации адреса предыдущего элемента на размер этого элемента. Для доступа к элементам массива обычно используются регистры, которые хранят базовый адрес массива и смещения относительно этого адреса. Использование регистров обеспечивает более эффективный и быстрый доступ к данным, по сравнению с обращением к памяти напрямую.

1. Как в памяти компьютера размещаются элементы матриц?

Размещаются последовательно по строкам или по столбцам, в зависимости от выбранного способа представления.

1. Чем моделирование матриц отличается от моделирования массивов? В каких случаях при выполнении операций для адресации матриц используется один регистр, а в каких – два?

Моделирование матриц отличается от моделирования массивов тем, что в матрицах данные организованы в виде двумерной структуры, где каждый элемент имеет два индекса - строку и столбец. При моделировании матриц обычно используются вложенные массивы, где каждый внутренний массив представляет собой строку матрицы.

При выполнении операций для адресации матрицы может использоваться один регистр, если матрица хранится в памяти последовательно по строкам или по столбцам, и адресация происходит с использованием только одного индекса. Для адресации элементов матрицы, когда используются оба индекса, обычно требуется два регистра - один для хранения базового адреса матрицы, а другой для хранения смещения по строке или столбцу.