МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ ФГБОУ ВО «ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ И.С.ТУРГЕНЕВА»

Кафедра «Программная инженерия»

**Отчет**

По лабораторной работе №4  
на тему:  
“Обработка массивов. Числа Фибоначчи”

Работу выполнили:  
Студенты группы 01-ИТ

Колчев Д.Н.  
Галькеев А. Ш.

Проверил:  
Доцент кафедры  
программной инженерии

Конюхова О.В.

Орёл, 2020

1. **Структура программы:**

d\_s segment

mas dw 18 dup(0)  
 min dw 0 ; Резервируем ячейки памяти для хранения данных  
 max dw 0

d\_s ends

c\_s segment  
assume ds:d\_s,cs:c\_s  
begin: ; сегменты видимы, программа началась  
mov ax, d\_s  
mov ds, ax  
mov ax, 0

mov si, 0 ; n = 0  
mov mas[si], 1 ; Определяем 1-е число фиб (n+=1)  
mov ax, mas[si] ; (fib 1)

mov si, 2  
mov mas[si], 1 ; Определяем 2-е число фиб (n+=1)  
mov bx, mas[si]

mov cx, -2 ; while i < (n-2) - без учёта пред. чис.  
add cx, 18 ; Ищем 18 эллемент мас-ва (n = 16)  
mov si, 4 ; Определяем значение i

Fib: ; Начало цикла  
mov dx, ax ; Сумма чисел двух предыдущ.  
add dx, bx  
mov mas[si], dx ; Сохраняем нов. число фиб

mov ax, bx ; fib1:= fib2  
mov bx, dx ; fib2:= fib\_sum  
add si, 2 ; Рассматриваем следующий эл. [si+2]  
Loop Fib ; Конец цикла (cx-1)  
 ; dx = A18 (2584)

mov si, 22 ; Перех. в кон. 2-й стр. (3x6 mas) (22 эл. +2)  
mov ax, 0  
mov bx, 0 ; Сбрасываем регистры  
mov dx, 0  
mov cx, 6 ; 6 эл-в массива проверяем на услов-е (2-ю стр)

searchMin:  
mov min, 9fffh ; минимуму присв. max знач.  
mov ax,mas[si]  
rcr ax, 1 ; Сдвиг вправо (CF = 0\1)  
jc minTrue1 ; Переход если CF=1  
jmp exitMin ; Иначе расматрив. след. элем (CF=0)

minTrue1:  
rcl ax, 1 ; Цикл. сдвиг влево, возвращ. исх числ.x  
cmp ax, min  
jb minTrue2 ; Переход, если (ax < min)  
jmp exitMin ; Иначе расматрив. след. элем (CF=0)

minTrue2:  
mov min, ax ; Присваиваем новый min

exitMin:  
sub si, 2 ; Переходим к след. элем.  
Loop searchMin ; Рассматрив. след. элем

mov si, 6 ; Начинаем с 4-го эл. в столбе  
mov cx, 3 ; 3 элем. в столбце (6x3 матрицы)  
mov ax, 0 ; Сброс ax

searchMax:  
mov ax, mas[si] ; Помещаем в ax нижн. эл. столб.  
rcr ax, 1 ; Сдвиг цикл. вправо/проверк. на чёт.  
jnc maxTrue1 ; Переход, если CF=0  
jmp exitMax ; Иначе перех в нач к след. эл.

maxTrue1:  
rcl ax,1 ; Циклич. сдвиг влево, возвр. исх. числ.  
cmp ax, max  
jg maxTrue2 ; Переход, если ax > max jg -> jns  
jmp exitMax ; Иначе перех в нач к след. эл.

maxTrue2:  
mov bx, ax ; Присваиваем новый max  
mov max, bx

exitMax:  
add si, 12 ; mas[30], mas[18], mas[6]  
Loop searchMax ; Рассматрив. след. элем

mov cx, min ; Проверим наш min (D = 13) {144, 89, 55, 34, 21, 13}  
mov bx, max ; Проверим наш max (0) {3, 55, 987}

;Персональное задание (№14)  
xor ax, ax  
xor bx, bx  
xor cx, cx ; Сбрасываем регистры  
xor dx, dx  
mov max, dx ; Обнуляем max  
mov si, 12 ; Переходим в начало 2-й строки  
mov cx, 6

search1:   
mov ax, mas[si]  
rcr ax, 1 ; Ищем нечётный эл-т  
jnc m1  
jmp newExit

m1:  
cmp ax, max ; Если нашли, сверяем с max  
jg m2   
jmp newExit

m2:  
mov max, ax ; ax < max --> max := ax

newExit:  
add si, 2  
Loop search1 ;Нашли нужный эл-т

xor ax, ax  
xor cx, cx  
mov si, 4  
mov cx, 3

adding: ;Добавляем число 3-м элементам  
xor ax, ax  
xor bx, bx  
mov ax, mas[si]  
add ax, max  
mov max[si], ax

add si, 4 ;отнимаем элементы 5-го столбца  
mov bx, mas[si]  
sub bx, ax  
sub si, 4  
mov mas[si], bx ;Сохраняем результат в 3-й столбец.

add si, 12  
Loop adding ;Переходим к следующей паре.

mov ah, 4ch  
int 21h  
c\_s ends  
end begin

1. **Контрольные вопросы:**
2. **Массивы и их представление в памяти компьютера.** Как структура представления, массив является упорядоченным множеством элементов определенного типа. Упорядоченность массива определяется набором целых чисел, называемых индексами, которые связываются с каждым элементом массива и однозначно конкретизируют его расположение среди других элементом массива. Локализация конкретного элемента массива - ключевая задача при разработке любых алгоритмов, работающих с массивами.  
    Наиболее просто представляются одномерные массивы. Соответствующая им структура хранения — это вектор. Она однозначна и есть не что иное, как просто последовательное расположение элементов в памяти. Чтобы локализовать нужный элемент одномерного массива, достаточно знать его индекс. Так как ассемблер не имеет средств для работы с массивом как структурой данных, то для доступа к элементу массива необходимо вычислить его адрес.  
    Представление двумерных массивов немного сложнее. Здесь мы имеем случай, когда структуры хранения и представления различны. О структуре представления говорить излишне — это матрица. Структура хранения остается прежней — вектор. Но теперь его нельзя без специальных оговорок интерпретировать однозначно. Все зависит от того, как решил разработчик программы «вытянуть» массив — по строкам или по столбцам. Наиболее естествен порядок расположения элементов массива — по строкам. При этом наиболее быстро изменяется последний элемент индекса.
3. **Команды условного и безусловного переходов. Каким образом вычисляются адреса переходов?**

1) Регистровый относительный. Является обобщением методов адресации, обеспечивающих вычисление эффективного адреса (EA) операнда в памяти в виде суммы базового значения адреса и «смещения» disp, указываемого в команде.  
 Относительную адресацию широко применяют как для адресации памяти, представленной в виде блоков (например, сегментов), так и для адресации специальных структур данных: массивов, записей и др. В зависимости от способа использования адресуемого в команде регистра различают базовый и индексный режимы адресации.  
 EA={BPBXSI,DI}+{Смещение disp}  
 2) Индексный. Применяется для обработки упорядоченных массивов данных; при этом каждый элемент массива определяется собственным номером. Тогда базовый адрес массива задаётся смещением disp, указываемым в команде, а значение индекса (номер элемента массива) определяется содержимым индексного регистра.  
 Индексная адресация удобна, если необходимо записать или считать список данных из последовательных ячеек памяти не подряд, а с некоторым шагом, указанным в индексе.

Пример: d\_s segment  
mas db 3,5,1,8,9,’$’  
d\_s ends  
c\_s segment  
assume ds:d\_s, cs:c\_s  
begin:  
…  
mov si,0 ;в si-номер элемента массива  
m1: mov ah, mas[si] ;mas- смещение  
;в ah – значение элемента массива mas с  
;номером в si  
add si,1  
jmp m1  
…  
c\_s ends  
end begin

3) Базово-индексный. Используется для доступа к элементам массива, адресуемого указателем. Базовый адрес массива задаётся указателем базы (базовым регистром), а номер элемента массива – содержимым индексного регистра.  
 Пример: mov ax, bx[si]  
 Если в bx содержится 100, а в si находится 52, то по адресу (смещению) 152 в сегменте данных находится искомое данное.  
 4) Относительный базовый индексный. Используется для адресации элементов в указываемом массиве записей. Базовый адрес массива задаётся указателем базы, номер записи (т.е., элемента массива) определяется содержимым индексного регистра, а смещение в команде указывает расстояние до записи.

Пример:  
 d\_s segment ;опишем массив из 5 сотрудников со значениями по умолчанию  
 mas\_sotr worker 5 dup (<>)  
 d\_s ends  
 c\_s segment  
 assume ds:d\_s, cs:c\_s  
 begin: ;в bx – адрес начала массива сотрудников  
 lea bx, mas\_sotr ;в si – смещение второй записи  
 mov si, (type worker)\*2 ; в ax – стаж второго сотрудника  
 mov ax,[bx][si].standing  
 c\_s endsend begin

1. **Способы описания массивов в сегменте данных.**   
    1) Перечислением элементов массива в поле операндов одной из директив описания данных. При перечислении элементы разделяются запятыми. К примеру, массив из 5 элементов. Размер каждого элемента 4 байта:

mas dd 1,2,3,4,5

2) Используя оператор повторения dup. К примеру, массив из 5 нулевых элементов. Размер каждого элемента 2 байта:  
 mas dw 5 dup (0)

Такой способ определения используется для резервирования памяти с целью размещения и инициализации элементов массива.  
 3) Используя директивы label и rept. Пара этих директив может облегчить описание больших массивов в памяти и повысить наглядность такого описания. Директива rept относится к макросредствам языка ассемблера и вызывает повторение указанное число раз строк, заключенных между директивой и строкой endm. К примеру, определим массив байт в области памяти, обозначенной идентификатором mas\_b. В данном случае директива label определяет символическое имя mas\_b, аналогично тому, как это делают директивы резервирования и инициализации памяти. Достоинство директивы label в том, что она не резервирует память, а лишь определяет характеристики объекта. В данном случае объект — это ячейка памяти. Используя несколько директив label, записанных одна за другой, можно присвоить одной и той же области памяти разные имена и разный тип, что и сделано в следующем фрагменте:  
...  
n=0  
...  
mas\_b label byte  
mas\_w label word  
rept 4  
dw 0f1f0h  
endm

В результате в памяти будет создана последовательность из четырех слов f1f0. Эту последовательность можно трактовать как массив байт или слов в зависимости от того, какое имя области мы будем использовать в программе — mas\_b или mas\_w.  
 4) Использование цикла для инициализации значениями области памяти, которую можно будет впоследствии трактовать как массив.

1. **Особенности обработки двумерных массивов в ассемблерных программах. Вычисление смещения элемента двумерного массива относительно начала сегмента данных.**

Двумерный массив нужно моделировать. На описании самих данных это почти никак не отражается — память под массив выделяется с помощью директив резервирования и инициализации памяти.  
 Непосредственно моделирование обработки массива производится в сегменте кода, где программист, описывая алгоритм обработки ассемблеру, определяет, что некоторую область памяти необходимо трактовать как двухмерный массив.  
 При обработке двумерных массивов удобно использовать базовую индексную адресацию со смещением.  
 Если последовательность однотипных элементов в памяти трактуется как двухмерный массив, расположенный по строкам, то адрес элемента (i, j) вычисляется по формуле  
 (база + количество\_элементов\_в\_строке \* размер\_элемента \* i+j)  
 Здесь i = 0...n–1 указывает номер строки, а j = 0...m–1 указывает номер столбца.  
 5. Какие режимы адресации данных можно использовать для доступа к элементам двумерного массива? Приведите примеры  
 Организовать адресацию двухмерного массива логично, используя рассмотренную нами ранее базово-индексную адресацию. При этом возможны два основных варианта выбора компонентов для формирования эффективного адреса:  
 · сочетание прямого адреса, как базового компонента адреса, и двух индексных регистров для хранения индексов:  
 mov ax,mas[ebx][esi]  
 · сочетание двух индексных регистров, один из которых является и базовым и индексным одновременно, а другой — только индексным:

mov ax,[ebx][esi]