МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ И.С. ТУРГЕНЕВА»

Кафедра «Информационных систем»

О Т Ч Е Т

о выполнении лабораторной работы № 4

на тему: «Обработка массивов. Числа Фибоначчи»

по дисциплине «Вычислительные системы, сети и телекоммуникации»

Выполнил: Кубарев А.Н. Шифр: 170551

Институт приборостроения, автоматизации и информационных технологий

Направление / специальность: 09.03.03 «Прикладная информатика»

Группа: 71-ПИ

Преподаватель Конюхова О.В.

Лукьянов П.В.

Отчет защищен с оценкой «\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_» Дата\_\_\_\_\_\_\_\_\_\_\_

Орел, 2017

Контрольные вопросы:

1. Элементы массива располагаются в памяти в смежных (соседних ячейках) относительно начального (нулевого) элемента, адрес которого в сегменте данных (смещение) является адресом начала всего массива в сегменте данных. Местоположение элемента в массиве характеризуется номером ячейки относительно начального адреса массива (начиная с нуля) и вычисляется как произведение индекса элемента на размер, занимаемый одним элементом в байтах. Чтобы получить доступ к элементу массива в сегменте данных, необходимо задать адрес начала массива и номер ячейки в массиве с нужным элементом. Начало массива определяется адресом mas.
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 ячейки с элементом 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,’$’; По адресу mas определена последовательность из 6 однобайтовых элементов(с учѐтом символа $); $ - признак конца последовательности

d\_s ends

c\_s segment

assume ds:d\_s, cs:c\_s

begin: …

mov si,3 ;в si – смещение элемента (8) относительно начала массива, т.е., адреса mas mov ah, mas[si] ;mas- смещение массива в сегменте данных ;в ah – значение элемента массива mas ;cо смещением в si, т.е. 8 …

c\_s ends

end begin

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:

d\_s segment

mas db 3,5,1,8,9,’$’; По адресу mas определена последовательность из 6 однобайтовых элементов

d\_s ends

c\_s segment

assume ds:d\_s, cs:c\_s

begin: …

mov si,3 ;в si – смещение элемента (8) относительно начала массива, т.е., адреса mas lea bx,mas; в регистр bx загружается адрес начала массива mas mov ah, bx[si] ; bx- смещение (адрес) массива в сегменте данных ; в ah – значение элемента массива mas cо смещением в si, т.е. 8 …

c\_s ends

end begin

В отличие от индексной адресации, где начальный адрес массива задаѐтся прямо в команде в виде смещения, в базово-индексном режиме начальный адрес массива предварительно загружается в один из базовых регистров (в примере BX). Как и при косвенной адресации, такой режим адресации данных удобно использовать при работе со сложными структурами, когда по неизменному адресу ячеек изменяется их содержимое.

Относительный базовый индексный. Используется для адресации элементов в указываемом массиве записей. Базовый адрес массива задаѐтся указателем базы, номер записи (т.е., элемента массива) определяется содержимым индексного регистра, а смещение в команде указывает расстояние до записи

Чтобы узнать стаж работы третьего сотрудника, сначала нужно в базовый регистр загрузить адрес начала массива (в примере mas\_sotr), затем определить смещение (номер ячейки) в массиве, с которого начинается запись о третьем сотруднике (в примере – это вторая запись, поскольку индексация начинается с нуля), и записать его в индексный регистр. Для определения номера нужной ячейки требуется размер одной записи (worker) в байтах (директива type) умножить на индекс записи в массиве. И, наконец, в найденной записи следует найти нужное поле (в примере standing). Таким образом, чтобы получить доступ к конкретному полю массива записей, сначала необходимо определить начало массива, в нѐм найти нужную запись, а уже в ней – требуемое поле.

Пример: …

d\_s segment

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 (начиная с нуля) записи

mov si, (type worker)\*2 ; в ax – стаж второго сотрудника

mov ax,[bx][si].standing …

c\_s ends

end begin

1. Для резервирования места под массивы используется оператор dup (duplicate, дублировать), который позволяет "размножить" байт, слово или двойное слово заданное число раз (rawdata dw 300 dup (1) Резервируются 300 слов заполненных числом 1)

Адрес первого числа в этом массиве равен просто numbers, адрес второго числа - numbers + 1, адрес третьего - numbers + 2 и т.д. (Следующая команда прочитает последний элемент этого массива в регистр DL: mov DL,numbers+4)

Иногда желательно обращаться к элементам массива (обычно небольшого размера) то с помощью индексов, то по их именам. Для этого надо к описанию массива, как последовательности отдельных данных, добавить дополнительное символическое описание адреса начала массива с помощью директивы label (метка):

numbers label byte

nmb0 db 0

nmbl db 1

nmb2 db 2

nmb3 db 3

nmb4 db 4

Метка numbers должна быть объявлена в данном случае с описателем byte, так как данные, следующие за этой меткой, описаны как байты и мы планируем работать с ними именно как с байтами. Если нам нужно иметь массив слов, то отдельные элементы массива следует объявить с помощью директивы dw, а метке numbers придать описатель word:

numbers label word

nmb0 dw 0

nmbl dw 1

nmb2 dw 2

nmb3 dw 3

nmb4 dw 4

1. 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 этом мы вольны в выборе того, как понимать расположение элементов двухмерного массива в памяти: по строкам или по столбцам. 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. Например, пусть имеется массив чисел (размером в 1 байт) mas(i, j) с размерностью 4 на 4 (i= 0...3, j = 0...3):

23 04 05 67

05 06 07 99

1. 08 09 23

87 09 00 08

В памяти элементы этого массива будут расположены в следующей последовательности: 23 04 05 67 05 06 07 99 67 08 09 23 87 09 00 08

Если мы хотим трактовать эту последовательность как двухмерный массив, приведенный выше, и извлечь, например, элемент mas(2, 3) = 23, то проведя нехитрый подсчет, убедимся в правильности наших рассуждений: Эффективный адрес(смещение) mas(2, 3) = mas + 4 \* 1 \* 2 + 3 = mas + 11

1. Организовать адресацию двухмерного массива можно используя базово-индексную адресацию. При этом возможны два основных варианта выбора компонентов для формирования эффективного адреса:

1)сочетание прямого адреса, как базового компонента адреса, и двух индексных регистров для хранения индексов: mov ax,mas[ebx][esi]

2)сочетание двух индексных регистров, один из которых является и базовым и индексным одновременно, а другой — только индексным:mov ax,[ebx][esi]

Программный код:

d\_s segment

mas dw 18 dup(?) ; по адресу mas определено 18 2-х байтовых значений

max dw 0

min dw 0

d\_s ends

assume ds: d\_s, cs: c\_s

c\_s segment

begin:

mov ax, d\_s

mov ds, ax

mov si, 0 ; в регистр si занесли 0 (адрес начла местоположения первого элемента массива)

mov mas [si], 0 ; в массив занесли первое значение = 0

add si, 2 ; увеличили регистр si на 2 (перешли на начало следущего элемента)

mov mas [si], 1 ; присвоили второму элементу значение = 1

mov cx, 16 ; цикл на 16 итераций, в котором выполняется алгоритм поиска чисел Фибоначчи

m1:

add si, 2 ; переходим на 3-ий элемент массива

mov ax, mas [si - 4] ; заносим в ax предпредыдущий элемент

mov bx, mas [si - 2] ; заносим в bx к предыдущий элемент

add ax, bx ; складываем их

mov mas [si], ax ; результат записываем в текущий элемент массива

loop m1

; ПОИСК НАИМЕНЬШИХ ИЗ НЕЧЁТНЫХ ЭЛЕМЕНТОВ ВТОРОЙ СТРОКИ ;

mov si, 12 ; переходим на адрес начала 1-ого элемента второй строки

mov cx, 5 ; цикл на 5 итерации

mov ax, mas[si] ; в ax заносим значение 1-ого элемента второй строки

mov min, ax ; в min заносим значение из ax P.S. нельзя так : mov min, mas[si], Т.К. В данной команде оба операнда представляют собой адреса памяти, что недопустимо в команде mov. В командах семейства процессоров 80х86 только один из операндов может ссылаться на адрес памяти.

m2:

add si, 2 ; увеличиваем si на 2

mov ax, mas[si] ; в ax записываем значение текущего элемента массива

cmp ax, min ; сравниваем значение текущего элемента массива и значение лежащие по адремсу min

jle ms ; если \*значения\* ax < min, то переприсваем на метке ms в min значение, лежащие в ax

jmp MI ; иначе переходим на завершение текущей итерации

ms:

mov min, ax

MI:

loop m2

; ПОИСК НАИБОЛЬШЕГО ИЗ ЧЁТНЫХ ЭЛЕМЕНТОВ ЧЕТВЁРТОГО СТОЛБЦА ;

mov si, 7 ; в регистр si записываем значение = 7

mov cx, 2 ; цикл на 2 итерации

mov ax, mas[si] ; в ax записываем значение 4-ого элемента первой строки (4-ый столбец)

mov max, ax ; в max заносим значение из ax P.S. нельзя так : mov min, mas[si], Т.К. В данной команде оба операнда представляют собой адреса памяти, что недопустимо в команде mov. В командах семейства процессоров 80х86 только один из операндов может ссылаться на адрес памяти.

m4:

add si, 12 ; уыеличиваем si на 12

mov ax, mas[si] ; в ax записываем значение текущего элемента массива

cmp ax, max ; сравниваем значение текущего элемента массива и значение лежащие по адремсу max

jnl m3 ; если \*значения\* ax > min, то переприсваем на метке m3 в max значение, лежащие в ax

jmp MI\_1 ; иначе переходим на завершение текущей итерации

m3:

mov max, ax

MI\_1:

loop m4

mov ah, 4ch

int 21h

c\_s ends

end begin