Федеральное государственное бюджетное образовательное учреждение высшего

образования

«Чувашский государственный университет им. И.Н. Ульянова»

Кафедра вычислительной техники

# **Лабораторная работа 5.**

**НЕПРИВИЛЕГИРОВАННЫЕ КОМАНДЫ ПРОЦЕССОРОВ х86**

Вариант 9

Выполнил: Иванов В.С.

студент группы ИВТ-41-22

Проверила:

Доцент Андреева А.А.

Чебоксары, 2024

Цель работы: ознакомиться с работой непривилегированных команд процессоров x86.

Задание: задан массив и границы некоторой области памяти. Необходимо преобразовать массив следующим образом: просматривается по очереди каждый элемент массива, и если он лежит внутри заданных границ, то на его место записывается число "1", иначе - "0". Элементы массива - слова. Для проверки используется команда BOUND.

Алгоритм решения задачи:

1. **Цикл по элементам массива:**
   * Начать с первого элемента массива.
   * Для каждого элемента записать 1 в результирующий массив.
2. **Проверка границ:**
   * Если индекс элемента находится между минимальным и максимальным значениями:
     + Оставить "1" в результирующем массиве.
   * Если индекс выходит за границы:
     + Записать "0" в результирующий массив.
3. **Вывод результата:**
   * Пройти по результирующему массиву и вывести каждый элемент на экран.

Текст программы:  
\_STACK segment para stack

db 1024 dup(?)

\_STACK ends

data segment

array dw 10, 20, 30, 40, 50, 60, 70, 80 ; Исходный массив

count dw 8 ; Количество элементов

bounds dw 2, 6 ; Таблица границ: min = 0, max = 7

data ends

code segment byte public

assume cs:code, ds:data, ss:\_STACK

main:

; Инициализация сегментного регистра данных

mov ax, data

mov ds, ax

mov ax,0

mov es,ax

mov bx,20

mov word ptr es:[bx], offset my\_int5

mov es:[bx+2], cs

; Инициализация индекса

mov cx,count ; Обнуляем cx для использования как счетчика

.386

mov ebx,0

lea bx, array ; Адрес исходного массива

mov esi,0

next\_element:

mov word ptr [ebx+esi\*2],"1"

; Проверяем, находится ли текущий индекс в пределах массива с помощью BOUND

bound si, dword ptr bounds ; Проверка границ индекса

increment\_index:

inc esi

loop next\_element ; Переходим к следующему элементу

end\_processing:

; Вывод результата

mov cx, count ; Количество элементов для вывода

mov si, offset array

print\_result:

mov ah, 02h ; Функция вывода символа

mov dl, [si] ; Загружаем байт из результирующего массива

int 21h ; Вызов DOS

add si,2 ; Переход к следующему байту

loop print\_result ; Повторяем, пока не выведем все элементы

; Завершение программы

mov ax, 4C00h ; Завершение программы

int 21h ; Вызов DOS

my\_int5 proc

mov word ptr [ebx+esi\*2],"0"

pop ax

add ax, 4

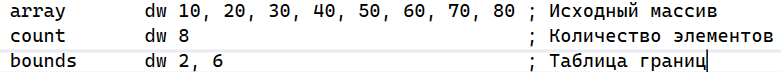
push ax ;корректирование адреса возвврата

iret

my\_int5 endp

code ends

end main

Для массива   
  
Вывод будет таким:  
  
  
Вывод: ознакомился с работой непривилегированных команд процессоров x86.

Ответы на вопросы:  
1. Назовите основные недостатки организации памяти в реальном режиме.

* сегменты бесконтрольно могут размещаться с любого адреса, кратного 16 (т.к. содержимое сегментного регистра аппаратно смещается на 4 разряда). Как следствие, программа может обращаться к любым адресам, в том числе и реально не существующим;
* сегменты имеют максимальный размер 64 Кбайт;
* сегменты могут перекрываться друг другом.

1. Перечислите все регистры: a) процессора 80286; б) процессора 80386;

а) **Регистры общего назначения**: AX, BX, CX, DX. Предназначены для хранения данных — шестнадцатибитных слов.

* **Адресные и индексные регистры**: SP, BP, SI, DI. Могут использоваться в качестве адресных, а также операндов в инструкциях обработки данных.
* **Сегментные регистры**: CS, DS, SS, ES. Определяют начальные адреса сегментов программы.
* **Регистр IP**. Служит для хранения адреса смещения следующей исполняемой команды.
* **Регистр F**. Предназначен для хранения флагов.
* **Регистр MSW**.  Слово состояния программы. Если флаг PE = 1 в MSW, то процессор переключается в защищённый режим.
* **Регистр TR**.  16-разрядный регистр, содержит селектор сегмента состояния задачи, используется для многозадачности.

б)

* **Регистры общего назначения**: ЕАХ, ЕВХ, ЕСХ, EDX, ESI, EDI, EBP, ESP.  Используются для хранения операндов, компонентов адресов операндов и указателей на ячейки памяти.
* **Сегментные регистры**: CS (регистр сегмента кода), DS (регистр сегмента данных), SS (регистр сегмента стека), ES, FS, GS.  Предназначены для хранения адресов сегментов кода, данных и стека, к которым программа имеет доступ.
* **Указатель команд**: EIP (extended instruction pointer). Содержит относительный адрес следующей команды, подлежащей выполнению.
* **Регистры управления**: CR0, CR2, CR3. Хранят признаки состояния процессора, общие для всех задач.
* **Регистры отладки и тестирования**: DR0–DR3, DR6, DR7. Поддерживают процесс отладки внутри процессора.

3. Что такое прерывание и особая ситуация; чем они отличаются.

**Особые ситуации** делятся на два типа: прерывания и исключения, в зависимости от того, вызвало ли эту ситуацию какое-нибудь внешнее устройство или выполняемая процессором команда.

**Отличие** в том, что **прерывания** происходят через случайные промежутки времени во время работы программы, как ответная реакция на сигналы от аппаратных средств. **Исключения** появляются, когда идёт выполнение инструкции и процессор сумел распознать ошибочную ситуацию, такую как деление на ноль.

4.Какие новые команды появились в процессоре 80286 и 80386.

PUSHFD: поместить в стек регистр EFLAGS; POP: извлечь операнд из стека; POPF: извлечь из стека регистр FLAGS; POPFD: извлечь из стека регистр EFLAGS; PUSHA: поместить все 16-разрядные регистры общего назначения в стек; PUSHAD: поместить все 32-разрядные регистры общего назначения в стек; POPA: извлечь все 16-разрядные регистры общего назначения из стека; POPAD: извлечь все 32-разрядные регистры общего назначения из стека. Команды PUSHA и POPA появились впервые в процессоре 80286, а команды PUSHAD, POPAD, PUSFD и POPFD – в процессоре 80386. Команда PUSHA помещает регистры в стек в следующем порядке: AX, CX, DX, BX, SP, BP, SI, DI, а команда POPA извлекает их в обратном порядке (но не извлекает регистр SP). Команды PUSHAD и POPAD точно также работают с регистрами EAX, ECX, EDX, EBX, ESP, EBP, ESI и EDI.

5.Как изменится содержимое регистров после выполнения команд: а) BTR AX,3 если AX = 0FFFFh б) SHLD AX,BX,10 если AX = 1234h, BX = 0015h.

а) BTR AX, 3 если AX = 0FFFFh:

AX = 0FFFFh (в двоичном виде: 1111 1111 1111 1111)

Команда BTR сбрасывает бит 3, т.е. 4-й справа (нумерация с нуля).

Результат: AX = 0FFFBh (в двоичном виде: 1111 1111 1111 1011).

б) SHLD AX, BX, 10 если AX = 1234h, BX = 0015h:

AX = 1234h (в двоичном виде: 0001 0010 0011 0100)

BX = 0015h (в двоичном виде: 0000 0000 0001 0101)

Команда SHLD выполняет побитовый сдвиг влево с заполнением из BX.

Сначала AX сдвигается влево на 10 бит, затем 10 младших бит из BX добавляются в AX.

Результат: AX = 2340h (в двоичном виде: 0010 0011 0100 0000).