МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №4

по дисциплине «Операционные системы»

Тема: Обработка стандартных прерываний.

Студентка гр. 8382 _____ Наконечная А. Ю.

Преподаватель _____ Ефремов М. А.

Санкт-Петербург
2020

Цель работы.

Построение обработчика прерывания сигналов таймера.

Постановка задачи.

Требуется написать и отладить программный модуль типа .EXE, который выполняет следующие функции:

- Проверяет, установлено ли пользовательское прерывание с вектором 1Ch.
- Устанавливает резидентную функцию для обработки прерывания и настраивает вектор прерываний, если прерывание не установлено, и осуществляет выход по функции 4Ch прерывания int 21h.
- Если прерывание установлено, то выводится соответствующее сообщение и осуществляется выход по функции 4Ch прерывания int 21h.
- Выгрузка прерывания по соответствующему значению параметра в командной строке /un. Выгрузка прерывания состоит в восстановлении стандартного вектора прерываний и освобождении памяти, занимаемой резидентом. Затем осуществляется выход по функции 4Ch прерывания int 21h.

Код пользовательского прерывания должен выполнять следующие функции:

- Сохранять значения регистров в стеке при входе и восстановить их при выходе.
- При выполнении тела процедуры накапливать общее суммарное число прерываний и выводить на экран. Для вывода на экран следует использовать прерывание int 10h, которое позволяет непосредственно выводить информацию на экран.

Выполнение работы.

Был создан обработчик прерываний сигналов таймера. Эти сигналы генерируются аппаратурой через определённые интервалы времени и при возникновении такого сигнала возникает прерывание с определённым значением вектора. Таким образом, управление будет передано функции, чья точка входа записана в соответствующий вектор прерывания.

Функция CHECK_LOAD проверяет, установлено ли пользовательское прерывание с вектором 1Ch. Функция UNLOAD_INTER, в которой происходит выгрузка прерывания. Функция LOAD_INTER, которая устанавливает резидентную функцию для обработки прерывания и настраивает вектор прерывания, если прерывание не установлено.

Результат выполнения программы представлен на рисунках 1 — 3.

```
Interruption loaded!Number of interruptions 0226
C:\>lr3_1.com
Available memory 639744 bytes

Extended memory 15360 bytes

MCB
Type 4D ¦ PSP segment 0008 ¦ Size 16     ¦ SC/SD
-------------------------------------------------------------------------------
Type 4D ¦ PSP segment 0000 ¦ Size 64     ¦ SC/SD
-------------------------------------------------------------------------------
Type 4D ¦ PSP segment 0040 ¦ Size 256    ¦ SC/SD
-------------------------------------------------------------------------------
Type 4D ¦ PSP segment 0192 ¦ Size 144    ¦ SC/SD
-------------------------------------------------------------------------------
Type 4D ¦ PSP segment 0192 ¦ Size 8992   ¦ SC/SD LR4
-------------------------------------------------------------------------------
Type 4D ¦ PSP segment 03CF ¦ Size 1442   ¦ SC/SD
-------------------------------------------------------------------------------
Type 5A ¦ PSP segment 03CF ¦ Size 639744 ¦ SC/SD LR3_1
-------------------------------------------------------------------------------
                  Number of interruptions 0879
C:\>
```

Рисунок 1 — Прерывание загружено в память

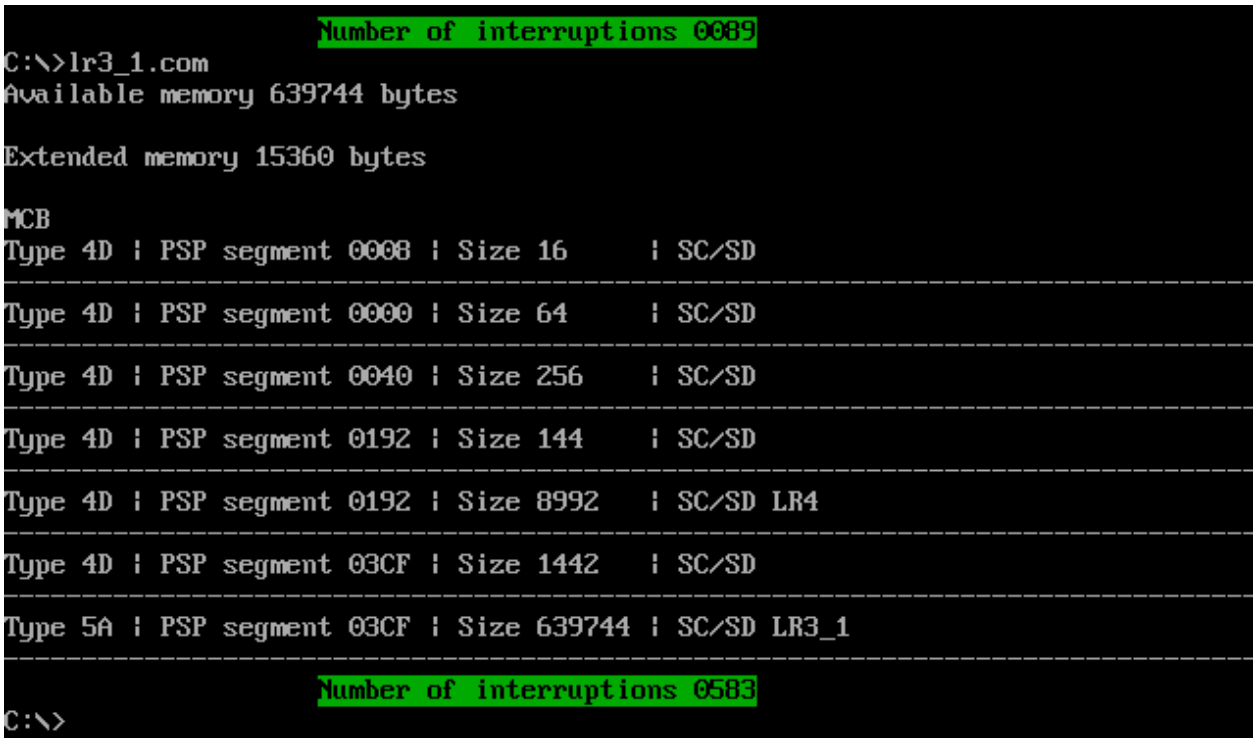

Рисунок 2 — повторная загрузка прерывания

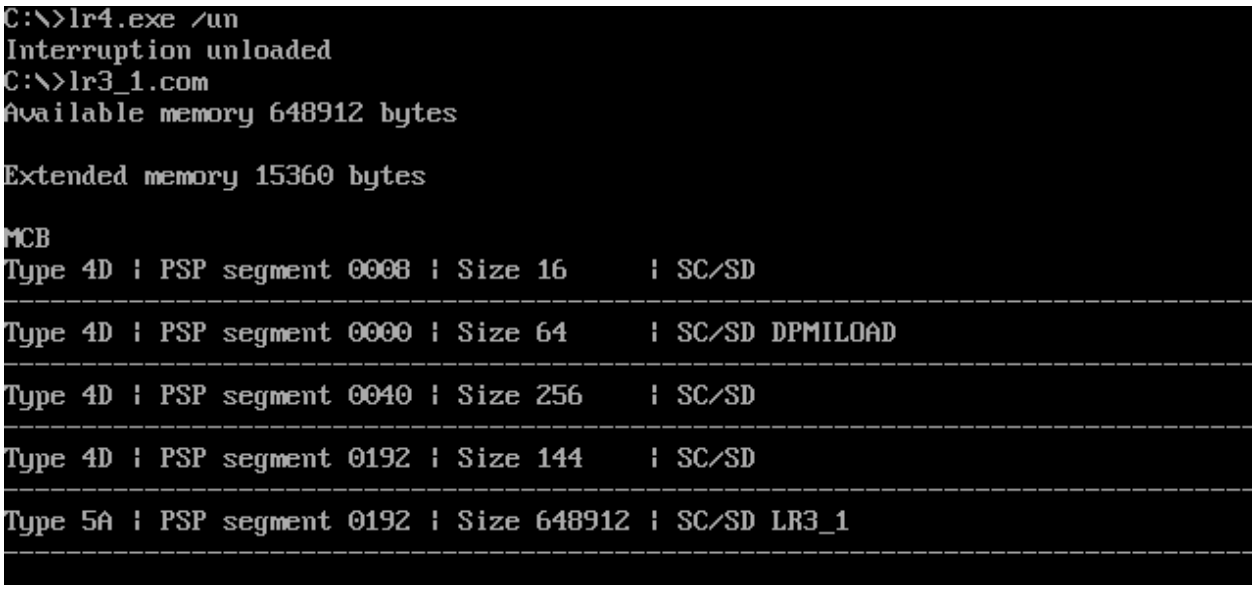

Рисунок 3 — выгрузка прерывания

Ответы на контрольные вопросы.

Сегментный адрес недоступной памяти

1) Как реализован механизм прерывания от часов?

После принятия сигнала прерывания, запоминаются содержимые регистров, по номеру источника прерывания в таблице векторов определяется смещение. Затем первые два байта помещаются в IP, а вторые два байта в CS. Дальше выполняется прерывание по сохранённому адресу, потом восстанавливается информация прерванного процесса и управление возвращается прерванной программе.

2) Какие прерывания использовались в работе?

Int 10h, int 21h и int 1Ch.

Выводы.

В ходе лабораторной работы была исследована обработка стандартных прерываний, а также построен обработчик прерываний сигналов таймера, которые генерируются аппаратурой через определённые интервалы времени.

ПРИЛОЖЕНИЕ А

Исходный код программы

lr4.asm

```
AStack SEGMENT STACK
    dw 100h dup(?)
AStack ENDS

DATA SEGMENT
     LOADED_STR db 'Interruption loaded $'
     NOT_LOADED_STR db 'Interruption unloaded $'
DATA ENDS

CODE SEGMENT
    ASSUME CS:CODE, DS:DATA, SS:AStack
     LOADED db ?
     ID dw 0FFFh
     KEEP_AX dw ?
     KEEP_SS dw ?
     KEEP_SP dw ?
     KEEP_IP dw ?
     KEEP_CS dw ?
     PSP dw ?
ROUT PROC FAR
     jmp START
     COUNT_STR db 'Interruption 0000$'
     STACK_INTER dw 100h dup(?)
     END_STACK_INTER db ?
START:
     mov KEEP_SS, SS
     mov KEEP_SP, SP
     mov KEEP_AX, AX
     mov AX, CS
     mov SS, AX
     mov SP, offset END_STACK_INTER
     push BX
     push CX
     push DX
     push SI
     push DS
     push BP
     push ES
     mov AH, 03h
     mov BH, 00h
     int 10h
     push DX
     mov AH, 02h
     mov BH, 0
```

```
      mov DX, 1715h
      int 10h
      mov AX, seg COUNT_STR
      push DS
      push BP
      mov DS, AX
      mov SI, offset COUNT_STR
      add SI, 12
      mov CX, 4
LOOP_INTER:
      mov BP, CX
      mov AH, [SI + BP]
      inc AH
      mov [SI + BP], AH
      cmp AH, 3Ah
      jne ENDING
      mov AH, 30h
      mov [SI + BP], AH
      loop LOOP_INTER
ENDING:
      pop BP
      pop DS
      push ES
      push BP
      mov AX, seg COUNT_STR
      mov ES, AX
      mov BP, offset COUNT_STR
      push AX
      push BX
      mov AH, 13h
      mov AL, 1
      mov BL, 20h
      mov CX, 17
      mov BH, 0
      int 10h
      pop BX
      pop AX
      pop BP
      pop ES
      pop DX
      mov AH, 02h
      mov BH, 0
      int 10h
      pop ES
      pop BP
      pop DS
      pop SI
      pop DX
      pop CX
```

```
      pop BX
      mov SP, KEEP_SP
      mov AX, KEEP_SS
      mov SS, AX
      mov AX, KEEP_AX
      mov AL, 20h
      out 20h, AL
      iret
ROUT ENDP
LAST_BYTE:

LOAD_INTER PROC near
      push AX
      push CX
      push DX
      mov AH, 35h
      mov AL, 1Ch
      int 21h
      mov KEEP_IP, BX
      mov KEEP_CS, ES
      push DS
      mov DX, offset ROUT
      mov AX, seg ROUT
      mov DS, AX
      mov AH, 25h
      mov AL, 1ch
      int 21h
      pop DS
      mov DX, offset LAST_BYTE
      mov CL, 4h
      shr DX, CL
      add DX, CODE
      inc DX
      xor AX, AX
      mov AH, 31h
      int 21h
      pop DX
      pop CX
      pop AX
      ret
LOAD_INTER ENDP

UNLOAD_INTER PROC near
      push AX
      push BX
      push DX
      push DI
      cli
      mov AH, 35h
```

```
      mov AL, 1Ch
      int 21h
      mov DI, offset KEEP_IP
      sub DI, offset ROUT
      mov DX, ES:[BX+DI]
      mov AX, ES:[BX+DI+2]
      push DS
      mov DS, AX
      mov AH, 25h
      mov AL, 1Ch
      int 21h
      pop DS
      mov AX, ES:[BX+DI+4]
      mov ES, AX
      push ES
      mov AX, ES:[2Ch]
      mov ES, AX
      mov AH, 49h
      int 21h
      pop ES
      mov AH, 49h
      int 21h
      sti
      pop DI
      pop DX
      pop BX
      pop AX
      ret
UNLOAD_INTER ENDP

CHECK_UNLOAD PROC near
      mov AX, PSP
      mov ES, AX
      cmp byte ptr ES:[81h+1], '/'
            jne NOT_UN
      cmp byte ptr ES:[81h+2], 'u'
            jne NOT_UN
      cmp byte ptr ES:[81h+3], 'n'
            jne NOT_UN
      mov DX, offset NOT_LOADED_STR
      mov AH, 09h
     int 21h
      call UNLOAD_INTER
      jmp STOP
NOT_UN:
      mov AL, LOADED
      cmp AL, 1
            je STOP
      mov DX, offset LOADED_STR
```

```
      mov AH, 09h
     int 21h
      call LOAD_INTER
STOP:
      ret
CHECK_UNLOAD ENDP

CHECK_LOAD PROC near
      push ES
      mov AH, 35h
      mov AL, 1Ch
      int 21h
      mov DX, ES:[BX-2]
      pop ES
      cmp DX, ID
            je GO
      jmp EXIT
      GO:
            mov LOADED, 1
      EXIT:
            ret
CHECK_LOAD ENDP

MAIN PROC far
      mov AX, DATA
      mov DS, AX
      mov AX, ES
      mov PSP, AX
      call CHECK_LOAD
      call CHECK_UNLOAD

      ; Выход в DOS
      xor AL,AL
      mov AH,4Ch
      int 21H
MAIN ENDP
CODE ENDS
END MAIN
```