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

Студент гр. 8383 _____ Колмыков В.Д.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2020

Цель работы.

Построить обработчик прерываний сигнала таймера. Изучить способы загрузки резидентной программы в память и ее выгрузку.

Процедуры, используемые в работе.

Название процедуры	Описание
MAIN	Головная процедура
INTERRUPT	Обработчик прерывания, выводящий по таймеру кол-во вызовов прерывания
CHECK_INTERRUPT	Проверка, не загружено ли уже прерывание, которое требуется загрузить
WRITE	Вывод строки из DX
LOAD_INTERRUPT	Загрузка и сохранение обработчика прерывания в памяти
CHECK_PARAM	Проверка параметров командной строки
UNILOAD_INTERRUPTION	Восстановление вектора прерывания и очищение памяти

Ход работы.

Был написан и отлажен программный модуль типа EXE, который выполняет следующие функции:

1) Проверяет установлено ли пользовательское прерывание
2) Устанавливает резидентную функцию для обработки прерывания, если она еще не установлена
3) Если она уже остановлена выводится соответствующее сообщение
4) Выгрузка прерывания по значению параметра в командной строке /un

Код программы приведен в приложении А.

Начальное состояние памяти, полученное при помощи программы из ЛР3 продемонстрировано на рис. 1.

```
Avaible memory: 648912 bytes
Memory was free successfuly
Extended memory: 15360 kbytes

MCB number 1
Owner:  MS DOS
Area size: 16 bytes



MCB number 2
Owner:  free
Area size: 64 bytes
DPMILOAD


MCB number 3
Owner: 0040
Area size: 256 bytes



MCB number 4
Owner: 0192
Area size: 144 bytes



MCB number 5
Owner: 0192
Area size: 1488 bytes
LR3_2


MCB number 6
Owner:  free
Area size: 647408 bytes
0 on par
```

Рисунок 1 – Состояние памяти до загрузки прерывания

Результат загрузки прерывания показан на рис. 2.

```
C:\>lr4

C:\>
                                Interrupt number 0076
```

Рисунок 2 – Результат работы прерывания

Состояние памяти с загруженным прерыванием приведено на рис. 3.

```
Avaible memory: 648016 bytes
Memory was free successfuly
Extended memory: 15360 kbytes

MCB number 1
Owner:  MS DOS
Area size: 16 bytes



MCB number 2
Owner:  free
Area size: 64 bytes
DPMILOAD


MCB number 3
Owner: 0040
Area size: 256 bytes



MCB number 4
Owner: 0192
Area size: 144 bytes



MCB number 5
Owner: 0192
Area size: 720 bytes
LR4


MCB number 6
Owner: 01CA
Area size: 144 bytes



MCB number 7
Owner: 01CA
Area size: 1488 bytes
LR3_2


MCB number 8
Owner:  free
Area size: 646512 bytes
```

Рисунок 3 – Состояние памяти с загруженным прерыванием

Результат повторной попытки загрузить прерывание показан на рис. 4.

```
Z:\>C:

C:\>lr4

C:\>lr4
Interrupt already exist
C:\>                              Interrupt number 1580
```

Рисунок 4 – Результат повторного запуска программы

Результат выгрузки прерывания показан на рис. 5.

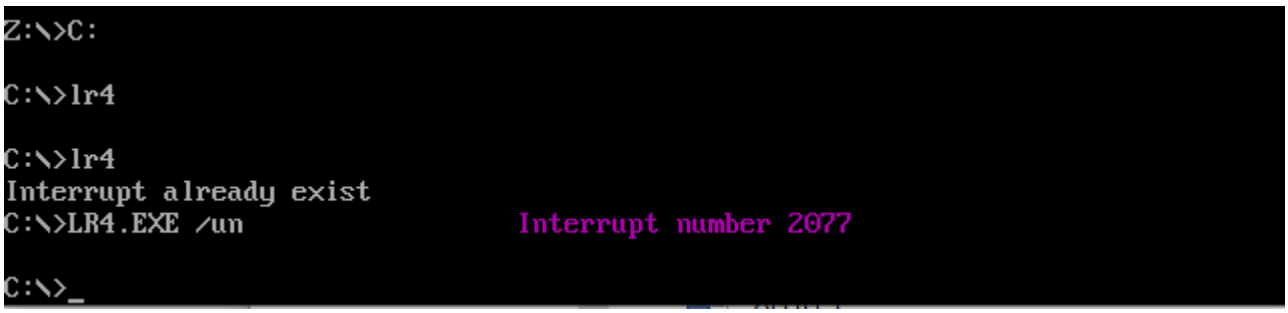

Рисунок 5 – Результат выгрузки прерывания из памяти

Состояние памяти после выгрузки прерывание показано на рис. 6.

```
Avaible memory: 648912 bytes
Memory was free successfuly
Extended memory: 15360 kbytes

MCB number 1
Owner:  MS DOS
Area size: 16 bytes



MCB number 2
Owner:  free
Area size: 64 bytes
DPMILOAD


MCB number 3
Owner: 0040
Area size: 256 bytes



MCB number 4
Owner: 0192
Area size: 144 bytes



MCB number 5
Owner: 0192
Area size: 1488 bytes
LR3_2


MCB number 6
Owner:  free
Area size: 647408 bytes
```

Рисунок 6 – Состояние памяти после выгрузки прерывания из памяти

Ответы на контрольные вопросы.

1) Как реализован механизм прерывания от часов?

- Происходит увеличение счетчика, проверка его на переполнение
- Происходит проверка на возможность обработать прерывание с соответствующим приоритетом
- Если они запрещены, продолжается выполнение текущей последовательности команд.
- Если разрешены, происходит вызов обработчика прерывания, находящегося по соответствующему адресу в таблице векторов прерываний
- До конца выполнения обработчика запрещается вызовы прерываний с таким же приоритетом
- Сохраняется контекст прерванного процесса
- Выполнение обработки прерывания
- Восстановление контекста, разрешение прерываний соответствующего приоритета, возврат к выполнению команды, которая бы вызвалась следующей, если бы прерывание не произошло

2) Какого типа прерывания использовались в программе?

В программе был написан обработчик прерывания, который обрабатывает асинхронное прерывание от таймера. В процессе выполнения программы вызываются синхронные прерывания для вызова прерываний DOS.

Выводы.

В ходе выполнения лабораторной работы была реализована программа загружающая и выгружающая пользовательское прерывание от системного таймера в память.

ПРИЛОЖЕНИЕ А
КОД ПРОГРАММЫ

```
CODE SEGMENT
	assume CS:CODE, DS:DATA, SS:STACKK, ES:NOTHING

	INTERRUPT PROC FAR
		jmp INTERRUPT_START
		STR_COUNTER db 'Interrupt number 0000'
		INTERRUPT_ID dw 0804h
		SAVE_AX dw 0
		SAVE_SS dw 0
		SAVE_SP dw 0
		KEEP_IP dw 0
		KEEP_CS dw 0
		PSP_SEGMENT DW 0
		INTERRUPTION_STACK dw 128 dup(0);Свой стек для обработчика

	INTERRUPT_START:
		mov SAVE_AX, AX
		mov SAVE_SP, SP
		mov SAVE_SS, SS
		mov AX, SEG INTERRUPTION_STACK
		mov SS, AX
		mov AX, offset INTERRUPTION_STACK
		add AX, 256
		mov SP, AX
		push BX
		push CX
		push DX
		push SI
		push DS
		push BP
		push ES
		mov AX, SEG STR_COUNTER
		mov DS, AX

		mov AH, 03h
		mov BH, 00h
		int 10h
		push DX;Сохранение позиции курсора для восстановления в
будущем
		mov AH, 02h
		mov BH, 00h
		mov DX, 1820h;18 строка, 20 столбец для курсора
		int 10h
		mov AX, SEG STR_COUNTER
		push DS
		mov DS, AX
		mov SI, offset STR_COUNTER
		add SI, 20
		mov CX, 4
	INT_CYCLE:;Увеличение счетчика
		mov AH, [SI]
		inc AH
		mov [SI], AH
		cmp AH, ':'
		jne INT_END_CYCLE
		mov AH, '0'
		mov [SI], AH
```

```
            dec SI
            loop INT_CYCLE
      INT_END_CYCLE:
            pop DS

            push ES
            push BP
            mov AX, SEG STR_COUNTER
            mov ES, AX
            mov BP, offset STR_COUNTER
            mov AH, 13h
            mov AL, 1h
            mov BL, 5h
            mov CX, 21
            mov BH, 0
            int 10h
            pop BP
            pop ES

            pop DX
            mov AH, 02h;Восстановление курсора
            mov BH, 0h
            int 10h

            pop ES
            pop BP
            pop DS
            pop SI
            pop DX
            pop CX
            pop BX
            mov SP, SAVE_SP
            mov AX, SAVE_SS
            mov SS, AX
            mov AX, SAVE_AX
            mov AL, 20h
            out 20h, AL
            IRET
            ret
      INTERRUPT ENDP
END_OF_INTERRUPT:
;_____________________________________________________________
      CHECK_INTERRUPT PROC
            push AX
            push BX
            push SI

            mov AH, 35h
            mov AL, 1Ch;Номер прерывания
            int 21h
            mov  SI, offset INTERRUPT_ID
            sub SI, offset INTERRUPT
            mov AX, ES:[BX + SI]
            cmp   AX, 0804h
            jne CHECK_INTERRUPT_END
            mov INTERRUPT_LOADED, 1

      CHECK_INTERRUPT_END:
            pop SI
            pop BX
            pop AX
            ret
      CHECK_INTERRUPT ENDP
```

```
;_____________________________________________________________
     WRITE      PROC  NEAR
          PUSH AX
          MOV   AH, 09H
        INT 21H
          POP AX
          RET
     WRITE      ENDP
;_____________________________________________________________
     LOAD_INTERRUPT PROC
          push AX
          push BX
          push CX
          push DX
          push DS
          push ES

          mov AH, 35h
          mov AL, 1Ch
          int 21h
          mov KEEP_CS, ES
          mov KEEP_IP, BX
          push DS
          mov DX, offset INTERRUPT
          mov AX, SEG INTERRUPT
          mov DS, AX
          mov AH, 25h
          mov AL, 1Ch
          int 21h
          pop DS
          mov DX, offset END_OF_INTERRUPT
          add DX, 10Fh
          mov CL, 4h
          shr DX, CL
          inc DX
          xor AX, AX
          mov AH, 31h
          int 21h

          pop ES
          pop DS
          pop DX
          pop CX
          pop BX
          pop AX
          ret
     LOAD_INTERRUPT ENDP
;_____________________________________________________________
     CHECK_PARAM PROC
          push AX
          push ES

          mov AX, PSP_SEGMENT
          mov ES, AX
          cmp byte ptr ES:[82h], '/'
          jne CHECK_PARAM_END
          cmp byte ptr ES:[83h], 'u'
          jne CHECK_PARAM_END
          cmp byte ptr ES:[84h], 'n'
          jne CHECK_PARAM_END
          mov UN_PARAM, 1

     CHECK_PARAM_END:
```

```
            pop ES
            pop AX
            ret
      CHECK_PARAM ENDP
;_____________________________________________________________
      UNLOAD_INTERRUPTION PROC
            CLI
            push AX
            push BX
            push DX
            push DS
            push ES
            push SI

            mov AH, 35h
            mov AL, 1Ch
            int 21h
            mov SI, offset KEEP_IP
            sub SI, offset INTERRUPT
            mov DX, ES:[BX + SI];Смещение
            mov AX, ES:[BX + SI + 2];Сегмент
            push DS
            mov DS, AX
            mov AH, 25h
            mov AL, 1Ch
            int 21h
            pop DS
            mov AX, ES:[BX + SI + 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

            pop SI
            pop ES
            pop DS
            pop DX
            pop BX
            pop AX
            STI
            ret
      UNLOAD_INTERRUPTION ENDP
;_____________________________________________________________
      MAIN PROC
            push DS
            xor AX, AX
            push AX
            mov AX, DATA
            mov DS, AX
            mov PSP_SEGMENT, ES

            call CHECK_INTERRUPT
            call CHECK_PARAM
            cmp UN_PARAM, 1
            je MAIN_UNLOAD
            mov AL, INTERRUPT_LOADED
            cmp AL, 1
            jne MAIN_LOAD
```

```
            mov DX, offset STR_INTERRUPT_EXIST
            call WRITE
            jmp MAIN_END
      MAIN_LOAD:
            call LOAD_INTERRUPT
            jmp MAIN_END
      MAIN_UNLOAD:
            cmp INTERRUPT_LOADED, 1
            jne NOT_EXIST
            call UNLOAD_INTERRUPTION
            jmp MAIN_END
      NOT_EXIST:
            mov DX, offset STR_NOT_EXIST
            call WRITE
      MAIN_END:
            xor AL, AL
            mov AH, 4Ch
            int 21h
      MAIN ENDP
;_____________________________________________________________________
CODE ENDS

STACKK SEGMENT STACK
      dw 128 dup(0)
STACKK ENDS


DATA SEGMENT
      INTERRUPT_LOADED db 0
      UN_PARAM db 0
      STR_INTERRUPT_EXIST db 'Interrupt already exist$'
      STR_NOT_EXIST db 'Interrupt does not exist$'
DATA ENDS
END MAIN
```