ЛАБОРАТОРНАЯ РАБОТА №5

Цель работы:

приобрести умения и практические навыки для работы с программными прерываниями.

Теоретическая часть:

Прерывание – временное прекращение выполнения текущей программы с передачей управления программе-обработчику прерываний.

Выделяют *внутренние* и *внешние* прерывания. *Внешние прерывания* – прерывания, вызванные от внешних устройств (от клавиатуры, от таймера, от звуковой карты и т.д.) – их также называют *аппаратными* (можно рассматривать как запрос от некого периферийного устройства на обработку данных от этого устройства). *Внутренние прерывания* могут возникнуть по следующим причинам:

– ненормальное внутреннее состояние микропроцессора, возникающее при выполнении некоторых команд (например, деление на ноль и т.п.);

– обработка команды *int* (программное прерывание).

Система обработки прерываний включает в себя:

– *аппаратные средства* – специальные выводы микропроцессора, микросхема программного контроллера прерываний;

– *программные средства* – таблица векторов прерываний, команда *int n* (*n* – номер прерывания), команда *cli* (запрет прерываний), команда *sti* (разрешение прерываний), команда *iret* (возврат из обработчика прерываний), *if* (флаг прерывания).

Вектор прерывания – адрес программного прерывания. Адрес таблицы векторов прерываний – 0000:0000. Длина каждого вектора прерываний составляет 4 байта. В таблице векторов прерываний младший адрес хранит смещение, старшее слово – сегментный адрес обработчика прерываний. Всего в таблице содержится 256 векторов прерываний.

Прерывание должно быть выполнено таким образом, чтобы после возврата из обработчика прерываний прерванная программа выполнялась так,

как если бы никакого прерывания не было. Для этого требуется *сохранение контекста прерванной программы*. Это значит, что все регистры микропроцессора, включая регистр флагов, после прерывания должны иметь те же значения, какие они имели до прерывания. При этом сохранение регистра флагов, *cs, ip* выполняет система, а сохранение регистров изменяемых обработчиков прерываний выполняет программист самостоятельно.

При выполнении прерываний важно, чтобы эта процедура, в свою очередь, также не была прервана. Поэтому на момент передачи управления обработчику прерываний все прерывания должны быть запрещены. При возникновении прерывания выполняются следующие действия:

1. сохранение в стеке регистра флагов, *cs* и *ip* прерываемой программы, запрет прерываний (флаг *IF* устанавливается в ноль);

2. определение по номеру прерывания адреса обработчика прерывания. Для этого номер прерывания умножают на 4 и полученное значение рассматривают как смещение вектора в таблице векторов прерывания;

3. занесение в сегментный регистр *cs* старшего слова, в регистр *ip* – младшего слова вектора прерывания – тем самым управление будет передано обработчику прерываний;

4. обработчик прерывания сохраняет регистры, которые он изменяет в ходе выполнения. После этого (после сохранения контекста прерванной программы) прерывания должны быть разрешены – флаг *IF* устанавливается в единицу;

5. после выполнения тела обработчика прерывания сохраненные регистры должны быть восстановлены. Перед восстановлением регистров прерывания следует запретить, установив флаг *IF* в ноль. Завершается обработчик прерывания командой *IRET*. Она считывает из стека в регистры *ip*, *cs* и *FLAGS* три слова с вершины стека;

6. после выполнения команды *IRET* управление возвращается прерванной программе, регистр флагов будет восстановлен, прерывания – снова разрешены.

Перехватом прерываний называется замена стандартного обработчика прерываний собственным обработчиком. Для перехвата прерывания в таблицу векторов прерывания нужно занести адрес собственного обработчика прерывания. Перед завершением программы нужно восстановить в таблице векторов прерывания адрес стандартного обработчика прерывания. Это означает, что перед заменой стандартного обработчика на собственный необходимо где-то сохранить значение первого. Как правило, адрес стандартного обработчика сохраняется в заранее выделенных ячейках памяти.

Например, если необходимо заменить прерывание *64h* (*64h = 100*) на собственный обработчик прерываний, который выводит на экран символ «!», программа может выглядеть следующим образом:

```
.model tiny
.data
  stsm dw 0
  stsg dw 0
.code
N:   push cs
     pop ds
     xor ax, ax
     mov es, ax
     mov  ax,  es:[100*4]  ;  сохранение  адреса  стандартного
обработчика
     mov stsm, ax
     mov ax, es:[100*4+2]
     mov stsg, ax
     cli
     mov es:[100*4+2],ds
     mov ax, offset nobr  ; занесение в ax адреса процедуры
     mov es:[100*4], ax
     sti
     int 100 ; вызов обработчика прерывания
     cli
     mov ax, stsm        ;       восстановление       адреса
стандартного обработчика
     mov es:[100*4],ax
     mov ax, stsg
```

```
        mov es:[100*4+2],ax
        sti
        mov ax,4c00h
        int 21h
nobr proc
  push ax
  push dx ; сохранение контекста прерванной программы
  sti
  mov ah,2
  mov dl,'!'
  int 21h ; вывод символа на экран
  cli
  pop dx  ; восстановление значений в регистрах
  pop ax
  iret    ; возврат управления из обработчика прерывания
nobr endp
end N
```

Для перехвата прерываний предназначены функции *35h* и *25h* прерывания *int 21h*:

– *35h* – функция для получения адреса обработчика прерывания. На входе *ah = 35h*, в *al* заносится номер обработчика прерывания, адрес которого необходимо получить. На выходе в *bx* заносится смещение, в *es* – сегментный адрес обработчика прерываний;

– *25h* – функция для установки обработчика прерываний. На входе *ah = 25h*, в *al* заносится номер прерывания, обработчик для которого необходимо установить, на выходе в сегментный регистр *ds* – сегментный адрес, в *dx* – смещение обработчика прерываний.

Прерывания от внешних устройств *(аппаратные)* принято делить на *маскируемые* (при определенных условиях могут быть проигнорированы микропроцессором) и *немаскируемые* (всегда обрабатываются). Всего используется 15 прерываний. Наиболее важными считаются следующие:

– *прерывания от системного таймера* (*int 8*) – выполняются 18,5 раз/с;

– *прерывания от клавиатуры* (*int 9*) – выполняются при каждом нажатии.

При перехвате аппаратного прерывания нельзя полностью заменять стандартный обработчик прерываний, потому что обработчики таких прерываний выполняют некоторые дополнительные функции.

Пусть адрес стандартного обработчика сохраняется в ячейках *st_off* и *st_seg*:

```
st_off dw 0; смещение стандартного обработчика
st_seg dw 0;сегментный адрес стандартного обработчика
```

Обычно заменяющий обработчик в начале или в конце работы передает управление стандартному обработчику прерываний. Соответственно, существует два способа вызова стандартного обработчика:

– *перед телом* собственного обработчика прерываний, например:

```
.model tiny
.data
  st_seg dw 0
  st_off dw 0
.code
nobr proc
  pushf
  call dword ptr cs:st_off
  <сохранение значений регистров>
  <тело собственного обработчика>
  <восстановление значений регистров>
  iret
nobr endp
N: push cs
   pop ds
   mov ah,35h
   mov al,9h   ; адрес стандартного обработчика прерываний от клавиатуры
   int 21h
   mov st_off,bx
   mov st_seg,es
   mov ah,25h
   mov al,9h
   mov dx,offset nobr
   int 21h     ; замена стандартного обработчика собственным обработчиком прерываний
   mov ah,1    ; ввод символов → нажатие клавиатуры → вызов прерывания №9
   int 21h
   mov ah,25h
   mov al,9
   mov dx, st_off
   mov ds, st_seg
   int 21h
   mov ax,4c00h
```

```
   int 21h
end N
```

– *после тела* собственного обработчика прерываний – обработчик прерываний в этом случае имеет следующий вид:

```
nobr proc
  <…>
  <…>
  <…>
  jmp dword ptr cs:st_off
nobr endp
```

Пример:

Заменить обработчик прерывания номер 64*h* собственным обработчиком, который выводит на экран символ «1». По завершении работы восстановить стандартный обработчик прерывания.

Решение

Сначала получим адрес обработчика прерывания 64*h* при помощи функции функций 35*h* и сохраним этот адрес. Замену обработчика прерывания будем выполнять при помощи функций 25*h* прерывания 21*h DOS*. В конце работы при помощи этой же функции восстановим в таблице вектор стандартного обработчика.

Программа

```
.model tiny
.data
st_off dw 0 ; смещение стандартного обработчика
st_seg dw 0 ; сегментный адрес стандартного обработчика
.code
n:   push cs
     pop ds
; получить адрес обработчика 64h
     mov ah, 35h
     mov al, 64h
     int 21h        ; вызов функции 35h
     mov st_seg, es ; сохранить сегментный адрес обработчика
64h
     mov st_off, bx ; сохранить смещение обработчика 64h
;в ds уже находится сегментный адрес нашего обработчика
     lea dx, prog   ; в dx занесем смещение нашего
обработчика
;заменить обработчик нашим обработчиком
     mov ax, 2564h
     int 21h        ; вызов функции 25h
```

```
          int 64h   ; вызов прерывания 64h
          ;    восстановление стандартного обработчик
          mov ax,2564h
          mov dx, st_off  ; в dx – смещение стандартного
обработчика
          mov ds, st_seg  ; в dx - сегментный адрес стандартного
обработчика
          int 21h         ; вызов функции 25h
          mov ax, 4c00h
          int 21h
          ; собственный обработчик прерывания
     prog proc
          ; вывод на экран цифры ‘1’
          mov ah, 02h ; функция вывода символа на экран
          mov dl, 31h ; в dl занесли код выводимого символа
          int 21h
          iret        ; возврат управления прерванной программе
     prog endp
     end n
```

Обращение к видеопамяти

В теле обработчика, заменяющего аппаратный обработчик прерывания, нельзя использовать функции прерывания *int 21h*. Поэтому для вывода сообщений можно применять *прямое обращение к видеопамяти*. *Видеопамять* – область памяти по адресу *0b800h:0000h*. Каждое слово видеопамяти соответствует одному знакоместу на экране (одному выводимому на экране символу). Младший байт слова определяет сам выводимый символ, а старший байт – цвет этого символа. Например, байт по адресу *0b800h:0000h* – это символ, отображаемый в верхнем левом углу экрана. Байт по адресу *0b800h:0001h* задает цвет этого символа. Старшие четыре бита определяют цвет фона при выводе символа, а младшие – цвет самого символа. Цвета задаются по следующей таблице:

Цвет фона или символа	**№**	**16-ричный номер**	**Цвет фона или символа**	**№**	**16-ричный номер**
черный	0	0	темно-серый	8	8
синий	1	1	светло-синий	9	9
зеленый	2	2	светло-зеленый	10	A
бирюзовый	3	3	светло-бирюзовый	11	B
красный	4	4	светло-красный	12	C
малиновый	5	5	светло-малиновый	13	D
коричневый	6	6	желтый	14	E
серый	7	7	белый	15	F

Например, вывести букву «а» в верхнем левом углу экрана можно при помощи следующих команд:

```
mov ax,0b800h            ; в ax – адрес видеопамяти
mov es,ax                ; в es – адрес видеопамяти из ax
mov byte ptr es:0,'a'    ; вывод символа на экран
mov byte ptr es:1,1fh    ; цвет символа - белый на синем
```

Для вывода информации в заданную позицию на экране необходимо вычислить смещение от начала видеопамяти до нужной ячейки. Например, надо вывести символ «*» в начале шестой строки экрана. Длина каждой строки – 80 символов. На экране помещается Под каждый символ отводится одно слово видеопамяти, т.е. 2 байта. Значит, под одну строку отводится 80 * 2 = 160 байтов. До начала шестой строки надо пропустить 5 строк. Таким образом, смещение, соответствующее началу шестой строки равно 160*5=800 байтов.

```
mov ax,0b800h  ; в ax – адрес видеопамяти
mov es,ax ;занесли адрес видеопамяти в сегментный регистр es
mov byte ptr es:[80*2*5],'*'  ; вывели на экран символ
mov byte ptr es:[80*2*5+1],4eh ;цвет символа - желтый на красном фоне
```

Варианты заданий:

Задание 1.

1. Написать собственный обработчик программного прерывания 87*h* и заменить им стандартный обработчик, обращаясь непосредственно к таблице векторов прерываний. Необходимо предусмотреть сохранение и восстановление регистров, изменяемых обработчиком прерывания. Этот обработчик должен выводить на экран символ «A».

2. Написать собственный обработчик программного прерывания 5*h* и заменить им стандартный обработчик, используя специальные функции прерывания 21*h* *DOS*. Необходимо предусмотреть сохранение и восстановление регистров, изменяемых обработчиком прерывания. Этот обработчик должен выводить на экран сообщение вида: «Этот обработчик создал студент *фамилия»*.

3. Написать собственный обработчик программного прерывания 5*h* и заменить им стандартный обработчик, обращаясь непосредственно к таблице векторов прерываний. Необходимо предусмотреть сохранение и восстановление регистров, изменяемых обработчиком прерывания. Этот обработчик должен выводить на экран символ «M».

4. Написать собственный обработчик программного прерывания 5*h* и заменить им стандартный обработчик, используя специальные функции прерывания 21*h* *DOS*. Необходимо предусмотреть сохранение и восстановление регистров, изменяемых обработчиком прерывания. Этот обработчик должен выводить на экран сообщение вида: «Студент фамилия может перехватывать прерывания*»*.

5. Написать собственный обработчик программного прерывания 5*h* и заменить им стандартный обработчик, обращаясь непосредственно к таблице векторов прерываний. Необходимо предусмотреть сохранение и восстановление регистров, изменяемых обработчиком прерывания. Этот обработчик должен выводить на экран символ «U».

6. Написать собственный обработчик программного прерывания 87h и заменить им стандартный обработчик, используя специальные функции прерывания 21h *DOS*. Необходимо предусмотреть сохранение и восстановление регистров, изменяемых обработчиком прерывания. Этот обработчик должен выводить на экран сообщение вида: «Этот обработчик написан студентом *фамилия».*

7. Написать собственный обработчик программного прерывания 87h и заменить им стандартный обработчик, обращаясь непосредственно к таблице векторов прерываний. Необходимо предусмотреть сохранение и восстановление регистров, изменяемых обработчиком прерывания. Этот обработчик должен выводить на экран символ «W».

8. Написать собственный обработчик программного прерывания 87h и заменить им стандартный обработчик, используя специальные функции прерывания 21h *DOS*. Необходимо предусмотреть сохранение и восстановление регистров, изменяемых обработчиком прерывания. Этот обработчик должен выводить на экран сообщение вида: «Студент *фамилия* может писать программы с прерываниями*».*

9. Написать собственный обработчик программного прерывания 87h и заменить им стандартный обработчик, обращаясь непосредственно к таблице векторов прерываний. Необходимо предусмотреть сохранение и восстановление регистров, изменяемых обработчиком прерывания. Этот обработчик должен выводить на экран символ «Y».

10. Написать собственный обработчик программного прерывания 87h и заменить им стандартный обработчик, используя специальные функции прерывания 21h *DOS*. Необходимо предусмотреть сохранение и восстановление регистров, изменяемых обработчиком прерывания. Этот обработчик должен выводить на экран сообщение вида: «Студент *фамилия* освоил перехват прерываний*».*

Задание 2. Написать собственный обработчик прерывания от клавиатуры. Стандартный обработчик вызывается ***после*** тела собственного

обработчика. Необходимо предусмотреть сохранение и восстановление регистров, изменяемых обработчиком прерывания. Этот обработчик должен выводить на экран любой символ на усмотрение обучающегося в следующем виде:

1. черным на красном фоне;
2. синим на белом фоне;
3. желтым на коричневом фоне;
4. белым на бирюзовом фоне;
5. желтым на темно-сером фоне;
6. светло-малиновым на синем фоне;
7. светло-красным на светло-зеленом фоне;
8. светло-бирюзовым на зеленом фоне;
9. коричневым на светло-красном фоне;
10. светло-зеленым на сером фоне;

Задание 3. Написать собственный обработчик прерывания от клавиатуры. Стандартный обработчик вызывается ***до*** тела собственного обработчика. Необходимо предусмотреть сохранение и восстановление регистров, изменяемых обработчиком прерывания. Этот обработчик должен выводить на экран символ «*****»

1. в центре экрана;
2. в правом верхнем углу;
3. в левом нижнем углу;
4. в правом нижнем углу;
5. в центре нижнего края окна;
6. в центре правого края окна;
7. в центре верхнего края окна;
8. в центре левого края окна;
9. в центре правой половины экрана;
10. в центре левой половины экрана.

Порядок работы:

1. Открыть программу, указанную преподавателем, после объяснения принципов работы с ней.
2. Создать файл с расширением, указанным преподавателем.
3. Ввести текст программы.
4. Сохранить программу.
5. Выполнить компиляцию.
6. Если ошибок нет, то запустить эмуляцию программы и пошагово выполнить ее.

Подготовить отчет о проделанной работе.

Вопросы к теоретическому материалу

1. Что называется прерыванием?

2. Какие виды прерываний выделяют?

3. Охарактеризуйте внешние прерывания.

4. Охарактеризуйте внутренние прерывания.

5. Что включает в себя система обработки прерываний?

6. Что называют аппаратными средствами в системе обработки прерываний?

7. Что называют программными средствами в системе обработки прерываний?

8. Какая команда используется для вызова прерываний?

9. Как называется бит флага, отвечающий за прерывания?

10. Какая команда используется для запрета прерываний?

11. Какая команда используется для разрешения прерываний?

12. Какая команда используется для возврата из обработчика прерываний?

13. Что называется вектором прерываний?

14. Укажите адрес таблицы векторов прерываний.

15. Какую длину имеет каждый вектор прерывания?

16. Что хранит младшее слово вектора прерываний?

17. Что хранит старшее слово вектора прерываний?

18. Что называется сохранением контекста прерванной программы?

19. Перечислите действия, которые необходимо выполнить при возникновении прерывания.

20. Что называется перехватом прерываний?

21. Что необходимо сделать для перехвата прерывания?

22. Назовите функции прерывания *int 21h*, предназначенные для перехвата прерываний?

23. Охарактеризуйте функцию *25h* прерывания *int 21h*.

24. Охарактеризуйте функцию *35h* прерывания *int 21h*.

25. На какие два вида принято делить аппаратные прерывания?

26. Какие аппаратные прерывания считаются наиболее важными?

27. Можно ли при перехвате аппаратного прерывания полностью заменять стандартный обработчик прерывания? Почему?

28. Какие два способа вызова стандартного обработчика существуют?

29. Каким образом можно производить вывод сообщений при перехвате аппаратных прерываний?

30. Укажите адрес видеопамяти.

31. Сколько байт видеопамяти определяют один выводимый на экран символ?

32. Что определяет младший байт слова видеопамяти при выводе символа на экран?

33. Что определяет старший байт слова видеопамяти при выводе символа на экран?

34. Каким образом определяется цвет выводимого символа и цвет фона, на котором он выводится?

35. Каким образом производится вывод символа в заданную позицию на экране?

ПРОЦЕСС СДАЧИ ЛАБОРАТОРНОЙ РАБОТЫ

По итогам выполнения каждой лабораторной работы студент:

1. демонстрирует преподавателю правильно работающие программы;
2. демонстрирует приобретенные знания и навыки, отвечая на несколько небольших вопросов преподавателя по составленной программе, возможностям ее доработки и теме лабораторной работы в целом;
3. демонстрирует отчет по выполненной лабораторной работе.

Итоговая оценка складывается из оценок по трем указанным составляющим.

Отчет по лабораторной работе оформляется по шаблону, представленному в приложении 1. Требования к отчету представлены в приложении 2.

ПРИЛОЖЕНИЕ 1. ШАБЛОН ОТЧЕТА

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Казанский национальный исследовательский технический университет им. А.Н. Туполева – КАИ»

Институт компьютерных технологий и защиты информации

Отделение СПО ИКТЗИ (Колледж информационных технологий)

ЛАБОРАТОРНАЯ РАБОТА №__

по дисциплине

СИСТЕМНОЕ ПРОГРАММИРОВАНИЕ

Работу выполнил

Студент гр.43__

Фамилия И.О.

Принял

Преподаватель Григорьева В.В.

Казань 2024

ВАРИАНТ __

1. **Цель работы**
2. **Задание на лабораторную работу** – вставляется задание на лабораторную работу, соответствующее индивидуальному варианту студента.
3. **Результат выполнения работы** – формируется описание хода выполнения работы (разработанных подпрограмм, классов, переменных, структур данных и т.п.) и вставляются скриншоты с результатами работы разработанных программ (скриншоты должны быть подписаны).
4. **Листинг программы** – вставляется код разработанной программы **с комментариями**.

ПРИЛОЖЕНИЕ 2. ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ ОТЧЕТА

Лист документа должен иметь книжную ориентацию, поля документа должны составлять: левое – 3 см, правое – 1,5 см, верхнее – 2 см, нижнее 2 см.

Нумерация страниц – внизу страницы, по центру, особый колонтитул для первой страницы.

Междустрочный интервал – 1,5 (полуторный), отступ первой строки – 1,25.

Текст документа должен быть выполнен с использованием шрифта Times New Roman, размер – 14, выравнивание – по ширине. Заголовки выполняются тем же шрифтом, но размера 16, полужирное начертание, размещение – по центру.

Рисунки должны размещаться по центру, они нумеруются по порядку. Перед рисунком в тексте на него должна быть ссылка. Подпись рисунка должна располагаться по центру и быть выполнена шрифтом Times New Roman, размер – 12. Сначала происходит нумерация рисунка, а затем пишется его название.