ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ
Северо-Кавказский филиал
ордена Трудового Красного Знамени федерального государственного бюджетного образовательного учреждения высшего образования
"Московский технический университет связи и информатики"

Методические указания
для проведения лабораторной работы №2

по дисциплине

«СИСТЕМНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ»

по теме

«Исследование механизма аппаратных прерываний»

Направление подготовки:
09.03.01 Информатика и вычислительная техника

Профили **Программное обеспечение и интеллектуальные системы**
Вычислительные машины, комплексы, системы и сети

Ростов-на-Дону
2019

УДК 681.3.06 (076)
ББК 32.07

Чикалов А.Н. Системное программное обеспечение. Исследование механизма аппаратных прерываний. Методические указания для проведения лабораторной работы №1 и №2. Ростов-на-Дону: Северо-Кавказский филиал МТУСИ, 2019.- 39 с.

В пособии изложены методические рекомендации и содержательные материалы для проведения практических и лабораторных занятий по изучению механизмов межпроцессорных коммуникаций в ЭВМ. Рассматриваются программные и аппаратные прерывания, технология использования интерфейса прикладного программирования.

Пособие содержит необходимые справочные материалы.

Методические указания предназначены для студентов, обучающихся по направлению подготовки 09.03.01 Информатика и вычислительная техника, профилей Вычислительные машины, комплексы, системы и сети, Программное обеспечение и интеллектуальные системы.

Пособие предназначено для использования при изучении дисциплин Системное программное обеспечение, а также может быть использовано преподавателями и студентами при изучении родственных дисциплин и в процессе самостоятельной работы.

Учебное пособие обсуждено и одобрено на заседании кафедры ИВТ
Протокол №1 от 26.08.2019 г.

Рецензент Зав. кафедрой ИВТ д.т.н. профессор Соколов С.В.

Лабораторная работа2. Исследование механизма аппаратных прерываний

Цель

1. Выработать практические умения и приобрести навыки перехвата аппаратных прерываний, оформления процедур обработки прерываний и их использования в вычислительном процессе.

2. Приобрести навыки эксплуатации ПЭВМ в ОС MS DOS.

3. Приобрести навыки анализа, обобщения и систематизации полученных результатов, навыки составления и оформления отчетных результатов. Привитие навыков точного и лаконичного представления докладов на вопросы преподавателя.

Учебные вопросы

1. Исследование работы нестандартной процедуры обработки прерываний от клавиатуры.

2. Исследование работы программы с использованием стандартного обработчика.

3. Исследование особенностей перехвата прерываний таймера и их использование.

Литература для подготовки к занятию

См. литературу к предыдущему разделу.

Содержание отчета

1. Название лабораторной работы и учебные вопросы.
2. Последовательно для каждого из разделов исследования:
   - задания и разработанные программы;
   - схемы и рисунки, поясняющие механизм перехвата аппаратных прерываний от клавиатуры и таймера;
   - пояснения к фрагментам рабочих программ.
3. Выводы о целесообразности реализации аппаратных прерываний с использованием нестандартных процедур обработки прерываний.
4. Краткие ответы на те контрольные вопросы, которые ещё не нашли своего отражения в отчете.

Вопросы для самопроверки

1. Приведите определение операционной системы.
2. Каким образом осуществляется обращение к функциям DOS и BIOS?
3. Перечислите регистры процессора, поясните их назначение.
4. Поясните процедуру прерывания, проиллюстрируйте ее.
5. Приведите классификацию прерываний.

6. В чем различие между программными и аппаратными прерываниями?

7. Поясните назначение вектора прерываний, способ его использования.

8. Каким образом осуществляется обращение из прикладной программы к системным функциям?

9. Поясните процесс взаимодействия системы с клавиатурой используя иллюстрации.

10. Поясните понятие SCAN-кода и кода ASCII.

11. Для чего предназначена программа обработки прерываний от клавиатуры?

12. Какой вектор прерываний закреплен за клавиатурой?

13. Для чего в составе машин IBM PC имеется таймер?

14. Поясните прикладное значение обработки прерываний от таймера.

15. Какие функции прерывания 21h DOS используются для чтения и изменения текущего времени и даты?

16. Поясните организацию аппаратных прерываний в машинах IBM PC.

Актуальность занятия

1. Функционирование ОС в рамках мультипрограммирования основано на использовании аппаратных прерываний. Механизм прерываний является одним из способов организации мультипрограммирования и основой работы любой ОС. Он позволяет реагировать на внешние события, происходящие асинхронно вычислительному процессу. Это важно, т.к. подобным образом организованы все системы реального времени, в том числе и системы военного назначения

2. Все внешние устройства взаимодействуют с системой через прерывания.

3. Это эффективный способ вызова процедур ОС, использования разработчиком имеющихся ресурсов, генерации нестандартных процедур обработки. Это возможность получить доступ к средствам ОС в том числе из среды Pascal.

4. Это инструмент подключения драйверов для новых устройств.

5. Занятие имеет явно выраженную практическую направленность и связано с решением задачи автоматизации управления войсками путем использования особенностей организации современных средств вычислительной техники и его программного обеспечения.

Способы реализации аппаратных прерываний

Механизм прерывания поддерживается:

- аппаратными средствами компьютера. Зависит от типа процессора и особенностей аппаратных компонентов (контроллеры, шины …)

- программными средствами. Аппаратные средства компьютера оказывают влияние на средства программной поддержки.

Способы реализации прерываний

Векторный – процессору на шину выставляется номер (номер вектора) устройства, идентифицирующий соответствующий обработчик. Это обеспечивается контроллером прерывания. Главное достоинство метода – скорость идентификации, однако требуется дополнительного оборудования.

Фактом прерывания является сигнал запроса прерывания INT на входе процессора. Номер вектора читается процессором с шины данных в специальном цикле подтверждения прерывания, который выполняется процессором при получении сигнала INT. Это и есть идентификация прерывания. По номеру вектора находится обработчик (его адрес или вектор).

Вектор может быть:

- фиксированным;
- конфигурируемым (например, переключателем);
- программируемым.

Опрашиваемый – ЦП вызывает все обработчики одного уровня (приоритета) пока кто-то из них не подтвердит, что прерывание поступило от обслуживаемого им устройства. Способ медленный, но дополнительных аппаратных затрат не требует. Информация об уровне приоритета должна определяться процессором. Для каждого уровня закреплено несколько устройств и, соответственно, обработчиков.

IRQ – Interrupt Request - запрос на прерывание (уровень).

На практике сочетают механизмы различных типов. Если для каждого IRQ – 1 устройство, то способ чисто векторный. Если несколько устройств относится к одному уровню – дополнительно проводится опрос всех устройств данного IRQ. Стандартным считается, естественно, векторный способ. Но если внешних устройств много, то

- в стандартных обработчиках используют заглушки;
- резервируют номера прерываний на перспективу (но номер уже есть, он обеспечен схемой распайки контроллеров).

В Pentium последовательность обработки следующая и считается векторной:

- на вход контроллера выставляют IRQ от источника прерывания;
- контроллер прерывания формирует INT на вход процессора и на шину данных выставляет номер прерывания (число 0-255);
- адреса обработчиков (вектора) определяются по **таблице векторов** (обработчиков прерываний). Таблица представляет собой упорядоченную последовательность адресов обработчиков, за каждым из которых закреплен конкретный неизменный номер. Смена адреса размещения обработчика в основной памяти ведет к изменению значения адреса в этой последователь-

ности, но не ведет к изменению положения этого адреса в последовательности (таблице векторов).

Дисциплины обслуживания прерываний обеспечивается:

- отключением прерываний (запрет) – обеспечивается установкой семафора (флага разрешения прерывания IF = 0);
- маскированием (запретом приема отдельных сигналов прерываний и, соответственно, запретом выполнения соответствующих обработчиков).

Для реализации отключений применяют программно доступный специальный регистр маски.

Прерывания существуют:

- **с относительным приоритетом**: обслуживание не прерывается до окончания обработчика. Для этого необходимо отключить системные прерывания;
- **с абсолютным приоритетом**: обслуживание прерывается, если приоритет поступившего прерывания выше. для реализации этого способа необходимо замаскировать прерывания с более низкими приоритетами.
- **по принципу стека**. LCFS Last come –first served – "последним пришел – первым обслужен". В этом случае не надо накладывать маску и отключать прерывания.

Однако переходные процессы необходимо маскировать ("критические секции"). Прерывать возможно только на этапе обработки прерывания.

Аппаратные прерывания различают:

- **маскируемые** – наступает временное необслуживание запросов. Приоритетность устанавливает контроллер прерываний;
- **немаскируемые** – имеют наивысший приоритет.

Принципы обработки аппаратных прерываний

Аппаратные прерывания опираются на:

- аппаратуру;
- программы ОС;

Основой аппаратной поддержки является программируемый контроллер прерываний.

Устройства физически подключаются линией к своему IRQ. Выход контроллера и вход INT ЦП соединены (рис2.1). В цикле подтверждения прерывания на ШД контроллером выставляется номер вектора.

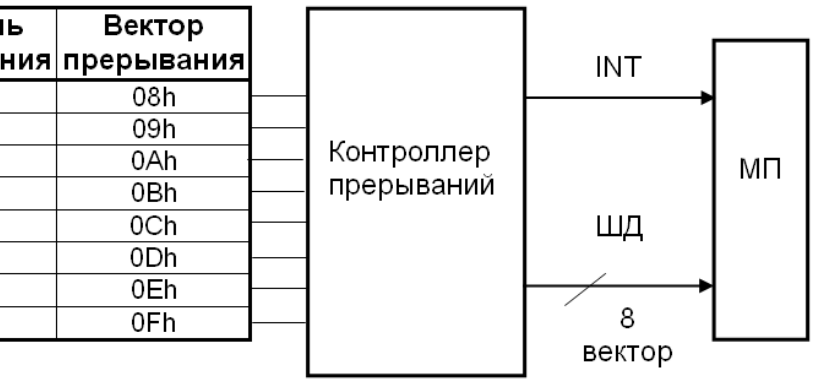
Контроллер предназначен для:

- формирования запроса INT;
- подготовки номера вектора;
- маскирования и приоритизации.

Номер вектора формируется по формуле:

N вектора = базовый вектор + № линии IRQ

Источник прерывания	Уровень прерывания	Вектор прерывания
Таймер	IRQØ	08h
Клавиатура	IRQ1	09h
Резерв	IRQ2	0Ah
COM2	IRQ3	0Bh
COM1	IRQ4	0Ch
НЖМД	IRQ5	0Dh
НГМД	IRQ6	0Eh
LPT1	IRQ7	0Fh



IRQ – Interrupt request – запрос прерывания
INT – Interrupt – прерывать

Рис.2.1. Организация аппаратных прерываний в IBM

Базовый вектор записан в 8-разрядный регистр контроллера и предназначен для выбора интервала для аппаратных прерываний в таблице векторов, он обеспечивает смещение по таблице.

Но устройств больше, чем 8. В этом случае выполняется каскадирование контроллеров прерываний (рис.2.2)

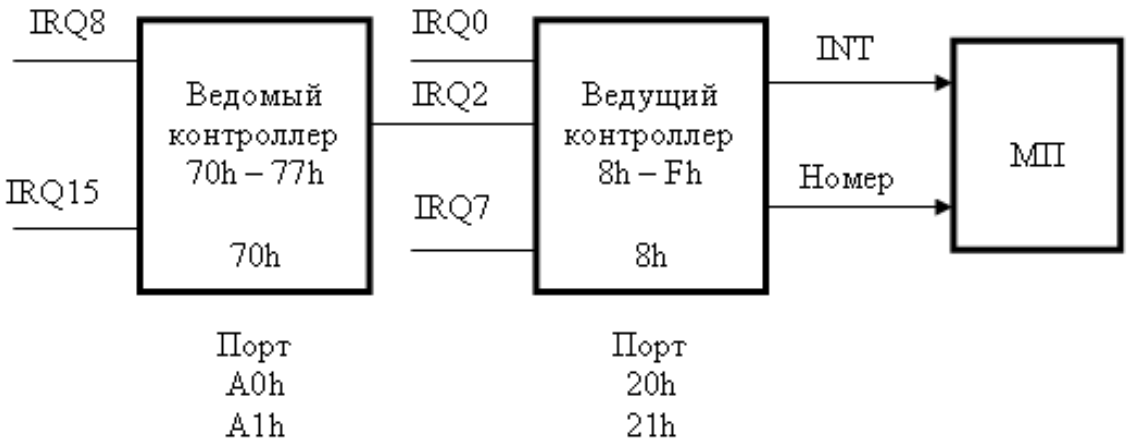


Рис.2.2. Каскадирование контроллеров прерывания

Приоритеты понижаются по мере увеличения номера прерывания:
70h – часы реального времени
71h – прерывание BIOS (для 0Ah)
72h – резерв
73h – резерв
74h – мышь PS/2

75h – сопроцессор
76h – жесткий диск AT
77h – резерв.

Обработка сигналов прерываний:

1. Запрос от устройства устанавливает в 1 бит регистра запросов (рис.2.3);

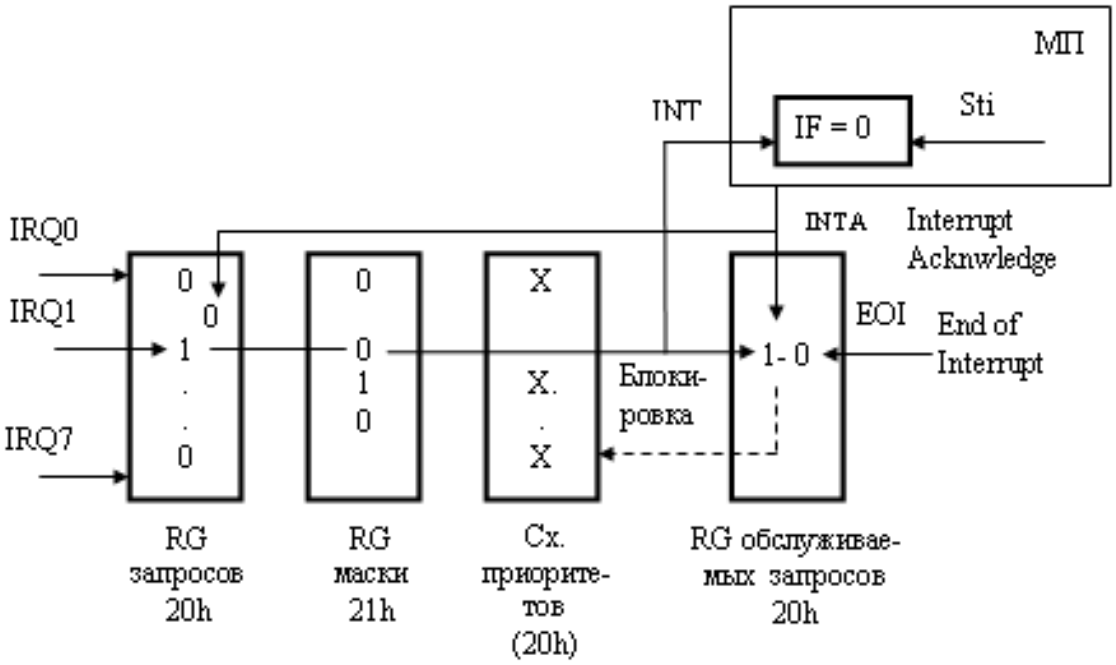


Рис.2.3. Программная модель контроллера прерываний

2. Ø в бите маски разрешает прохождение запроса (1-запрещает);
3. Через схему приоритетов формируется INT и готовится 1 в регистр обслуживаемых запросов (но это еще не обслуживание запроса);
4. Микропроцессор регистрирует INT если установлен флаг разрешения прерывания IF=1 (IF=0 - запрещает). Предусмотрены команды процессора для работы с флагом:

cli (clear interrupt flag) – сброс флага IF=0 - запрет всех немаскируемых прерываний;

sti (set interrupt flag) – установить флаг прерываний IF=1 – флаг разрешения прерывания (Interrupt enable flag);

5. Ответ микропроцессора на сигнал прерывания:

- сброс флага IF = 0 (запрет всех прерываний);
- выработка сигнала INTA, который выполняет:
  - а) – устанавливает разрешенную 1 в регистр обслуживаемых запросов. Схемой приоритетов блокируются все уровни с текущего и ниже;
  - б) – сбрасывает (устанавливает Ø) бит регистра запросов. После этого регистр готов к приему следующего запроса по этой же линии;

6. Разрешение более высоких прерываний. Выполняется программно командой Sti.

7. Разрешение всех прерываний EOI (End Of Interrupt)– обнулением регистра обслуживаемых запросов (выполняется программно командой OUT 20h, 02h).

Структура обработчика прерывания

Структура определяет последовательность операций, выполняемых аппаратно и программно с целью управления прерываниями. Начало процесса инициализируется поступлением, например, IRQ1 и формированием процессору сигнала INT:

1) Аппаратно запрещаются все прерывания независимо от приоритета, т.к. флаг IF устанавливается в Ø, в стеке сохраняются регистр флагов и адрес возврата CS:IP;

2) Аппаратно загружается CS (из второй пары ячеек) и IP (из первой пары ячеек таблицы векторов по адресам, вычисленным по номеру прерывания. С этого момента запускается обработчик прерывания;

3) В обработчике командой sti – разрешаются более высокие прерывания, т.к. при этом флаг IF устанавливается в 1. В данном примере это могут быть только IRQ∅. Однако все уровни прерываний, начиная с текущего (IRQ1), остаются заблокированными в контроллере прерываний. Момент подачи sti выбирают исходя из логики обработчика, но как правило в самом начале, чтобы не блокировать более приоритетные запросы;

4) Реализация в обработчике EOI - End Of Interrupt – конец прерывания снимает блокировку и в контроллере. С этого момента разрешены все прерывания независимо от приоритета. Момент выдачи этого приказа обычно наступает в конце обработчика, чтобы исключить вложенные прерывания от запросов того же уровня;

5) Заканчивается обработчик командой **IRET** - Interrupt Return – возврат после обработки прерывания. При этом восстанавливаются IP:CS и регистр флагов FLAGS из стека.

Особенности обработки прерываний клавиатуры

В настоящее время существует много типов клавиатур с различным количеством клавиш и схемой их размещения. Как вводятся символы? Куда попадают? Как обрабатываются символы с клавиатуры?

Управляет работой клавиатуры – контроллер клавиатуры. Всякий раз при нажатии или отжатии любой клавиши он записывает в выходной свой порт по адресу 60h один байт:

- SCAN – код нажатия клавиши (ее номер на клавиатуре) или
- SCAN – код отжатия, формируемому суммированием того же кода с константой 80h. Код отжатия, таким образом, отличается от исходного кода

нажатия единицей в самом старшем разряде, что и соответствует шестнадцатеричному коду 80h.

Далее контроллер клавиатуры формирует запрос INT процессору через контроллер прерываний. Подключение контроллера клавиатуры стандартно и соответствует номеру 09h, формируемому по уровню IRQ1 с учетом базового номера 08h.

Коды присвоены всем клавишам, в том числе специальным (нерисуемым): Shift, Ctrl, Alt, Caps Lock (верхний регистр) и т.д. Всякое изменение состояния клавиатуры (нажатие, отжатие) вызывает сигнал запроса на прерывание INT. Флаги (состояния специальных управляющих клавиш) сохраняются в оперативной памяти пока клавиши нажаты.

Обработчик BIOS, вызываемый по прерыванию №9, осуществляет преобразование SCAN-кодов из буфера контроллера клавиатуры (он расположен в микросхеме контроллера) в коды символов ASCII. Для этого анализируется предыстория, состояние управляющих клавиш, комбинации клавиш и по завершении работы обработчика №9 SCAN-коды и коды символов запоминаются в **кольцевом буфере клавиатуры**, находящемся в основной памяти ЭВМ. Каждый из 16 элементов буфера содержит 2 байта: SCAN-код + ASCII-код (буфер еще называют таблицей трансляции SCAN-кодов в коды ASCII). Буфер синхронизирует прием кодов от контроллера клавиатуры и процесс использования кодов выполняемой программой. Дисциплина обслуживания буфера – очередь. Если буфе будет заполнен полностью, то клавиша игнорируется и выдается звуковой сигнал.

Объем буфера – 16 двухбайтных слов, размещенных по адресам 40H:1EH до 40H:3DH:

1Eh – 1Fh 1 слово
…
20h – 2Fh 8 слово
…
30h – 3Dh 16 слово

} 16 слов (клавиш)

Слово флагов клавиатуры	40h:17h	Ins	Caps Lock	Num Lock	Scroll Lock	Alt	Ctrl	Shift лев	Shift прав
		7	6	5	4	3	2	1	0
Головной адрес	40h:1Ah								
Хвостовой адрес	40h:1Ch								
Кольцевой буфер ввода на 16 слов	40H:1Eh	1 слово							
		1 байт SCAN-код				1 байт ASCII-код			
	. . .								
	40h:3Dh	16 слово							

Указатели:

Головной указатель – содержит адрес самого старого, но еще не востребованного кода.

Хвостовой указатель – адрес первой свободной ячейки в буфере.

Динамика работы буфера:

1. Для получения кода из буфера необходимо обратиться к прерыванию INT16h (драйвер BIOS клавиатуры);

2. Для определения сведений о том, что буфер пуст, сравниваются адреса указателей: они должны быть равны. Если при этом получен запрос на чтение из клавиатуры (INT16h), то происходит то, что называют "ожиданием нажатия". Это прием широко используется в Pascal по команде READLN без имени переменной;

3. Переполнению соответствует ситуация, при которой хвост подошел к голове с обратной стороны. В этом случае прием кодов блокируется и формируется звуковой сигнал;

4. Ввод с упреждением – нажатие клавиш при условии, что программе они не требуются. Они накапливаются и будут востребованы (выведены) когда потребуются.

SCAN-коды символов стандартной клавиатуры приведены в таблице

HEX-код	Основной регистр	Верхний регистр
01	ESC	ESC
02	1	!
03	2	@
04	3	#
05	4	$
06	5	%
07	6	^
08	7	&
09	8	*
0A	9	(
0B	0	)
0C	-	_
0D	=	+
0E	Пробел	Пробел
0F	→	←
10	q	Q
11	w	W
12	e	E
13	r	R
14	t	T
15	y	Y

HEX-код	Основной регистр	Верхний регистр
1E	a	A
1F	s	S
20	d	D
21	f	F
22	g	G
23	h	H
24	j	J
25	k	K
26	l	L
27	;	:
28	'	"
2B	\	|
2C	z	Z
2D	x	X
2E	c	C
2F	v	V
30	b	B
31	n	N
32	m	M
33	,	<
34	.	>

16	u	U
17	i	I
18	o	O
19	p	P
1A	[	{
1B	]	}
1C	ВК	ВК
1D	н/в	н/в

35	/	?
47	Home	н/в
49	PgUp	н/в
4F	End	н/в
50	↓	н/в
51	PgDown	
52	Insert	
53	Del	

Для нерисуемых кодов (F1, F2, стрелки, …….) таблица формируется следующим образом:

- на месте ASCII-кода записывается – 00, SCAN-код формируется обычным образом. Слово [SCAN + 00] называется расширенным кодом ASCII. 00 – признак расширенного кода;
- если нажимается несколько клавиш (комбинации с Ctr, Alt, Shift), то вместо SCAN-кода записывается специальный код этой комбинации. Например,

F1 – 3B00h
Shift/F1 – 5400h
Ctrl/F1 – 5E00h
Alt/F1 – 6800h.

Расширенные ASCII – коды представлены в таблице 2.1

Взаимодействие прикладных и системных обработчиков прерываний

Обработчики прерываний используются всеми программными продуктами. Однако их набор не всех разработчиков устраивает. Поэтому применяют различные комбинации системных и своих обработчиков. Например, технология использования может предусматривать:

а) использование прикладного (своего) обработчика вместо системного:

1. Сохраняют адрес системного заменяемого обработчика (используется прерывание INT 21h с функцией 35h – получение вектора прерывания по его номеру, INT21 – диспетчер функций DOS);

1. Устанавливают адресом своего обработчика по тому же номеру (прерывание INT 21h с функцией 25h – установка вектора прерывания по его номеру). В этом случае всякое обращение по номеру прерывания приводит к вызову прикладного обработчика, а системный не задействуется, т.к. вызова его не происходит;

2. Восстановление исходного (сохраненного ранее при замещении) адреса обработчика после выполнения всей программы для сохранения логической целостности операционной системы (прерывание INT 21h с функцией

25h). Если этот вектор системой не используется, то восстановление, обычно, не производится;

Клавиша	Код	Клавиша	Код	Клавиша	Код	Клавиша	Код	Клавиша	Код
F1	3Bh	Alt-R	13h	Shift-F11	87h	Alt-Tab	A5h	Alt-I	17h
F2	3Ch	Alt-S	1Fh	Shift-F12	88h	Ctrl-Tab	94h	Alt-J	24h
F3	3Dh	Alt-T	14h	Alt-0	81h	Alt-Del	A3h	Alt-K	25h
F4	3Eh	Alt-U	16h	Alt-1	82h	Alt-End	9Fh	Alt-L	26h
F5	3Fh	Alt-V	2Fh	Alt-2	83h	Alt-Home	97h	Ctrl-Right	74h
F6	40h	Alt-W	11h	Alt-3	84h	Alt-Ins	A2h	Ctrl-End	75h
F7	41h	Alt-X	2Dh	Alt-4	85h	Alt-PgUp	99h	Ctrl-Home	77h
F8	42h	Alt-Y	15h	Alt-5	86h	Alt-PgDn	A1h	Ctrl-PgDn	76h
F9	43h	Alt-Z	2Ch	Alt-6	87h	Alt-Enter	1Ch	Ctrl-PgUp	84h
F10	44h	Alt-\	2Bh	Alt-7	88h	Ctrl-F1	5Eh	Alt-Up	98h
F11	85h	Alt-,	33h	Alt-8	89h	Ctrl-F2	5Fh	Alt-Down	A0h
F12	86h	Alt-.	34h	Alt-9	8Ah	Ctrl-F3	60h	Alt-Left	9Bh
Alt-F1	68h	Alt-/	35h	AltC	8Bh	Ctrl-F4	61h	Alt-Right	9Dh
Alt-F2	69h	Alt-BS	0Eh	Alt-=	8Ch	Ctrl-F5	62h	Alt-K/	A4h
Alt-F3	6Ah	Alt-[	1Ah	NUL	03h	Ctrl-F6	63h	Ctrl-K*	37h
Alt-F4	6Bh	Alt-]	1Bh	Shift-Tab	0Fh	Ctrl-F7	64h	Alt-K-	4Ah
Alt-F5	6Ah	Alt-;	27h	Ins	52h	Ctrl-F8	65h	Alt-K+	4Eh
Alt-F6	6Dh	Alt-'	28h	Del	53h	Ctrl-F9	66h	Alt-KEnter	A6h
Alt-F7	6Eh	Alt-`	29h	SysRq	72h	Ctrl-F10	67h	Ctrl-K/	95h
Alt-F8	6Fh	Shift-F1	54h	Down	50h	Ctrl-F11	89h	Ctrl-K*	96h
Alt-F9	70h	Shift-F2	55h	Left	4Bh	Ctrl-F12	8Ah	Ctrl-K-	8Eh
Alt-F10	71h	Shift-F3	56h	Right	4Dh	Alt-A	1Eh	Ctrl-K+	90h
Alt-F11	8Bh	Shift-F4	57h	Up	48h	Alt-B	30h	Ctrl-K8	8Dh
Alt-F12	8Ch	Shift-F5	58h	Enter	4Fh	Alt-C	2Eh	Ctrl-K5	8Fh
Alt-M	32h	Shift-F6	59h	Home	47h	Alt-D	20h	Ctrl-K2	91h
Alt-N	31h	Shift-F7	5Ah	PgDn	51h	Alt-E	12h	Ctrl-K0	92h
Alt-O	18h	Shift-F8	5Bh	PgUp	49h	Alt-F	21h	Ctrl-K.	93h
Alt-P	19h	Shift-F9	5Ch	Ctrl-Left	73h	Alt-G	22h		
Alt-Q	10h	Shift-F10	5Dh	Alt-Esc	01h	Alt-H	23h		

Таблица 2.1

Расширенные ASCII

б) сцепление с системным обработчиком (прикладной + системный)

1. Сохраняют адрес системного обработчика для последующего перехода (прерывание INT 21h с функцией 35h);
2. В вектор прерывания помещают адрес прикладного обработчика (прерывание INT 21h с функцией 25h);
3. В программе своего обработчика обращение к системному обработчику осуществляется по команде JMP.

При таком построении обращение по прерыванию приводит к передаче управления прикладному обработчику с предварительным сохранением контекста. После выполнения тела прикладного обработчика переход к началу системного обработчика по команде JMP не сопровождается операциями восстановления контекста. И только после выполнения тела системного обработчика штатным образом по команде IRET осуществляется восстановление исходного контекста и продолжение прерванного порядка выполнения основной программы. Вызов такой сцепки возможен как аппаратно, так и программно по команде прерывания;

в) сцепление с системным обработчиком (системный + прикладной)

1. Получается адрес системного обработчика (используется прерывание INT 21h с функцией 35h) и сохраняется во временной переменной;
2. Оформляется процедура следующей структуры:
   - сохранение флагов в стеке командой PUSH F;
   - вызов подпрограммы по адресу вектора обработчика (адрес находится в сохраненной переменной);
   - тело прикладной программы пользователя;
   - команда IRET;
   - конец процедуры.

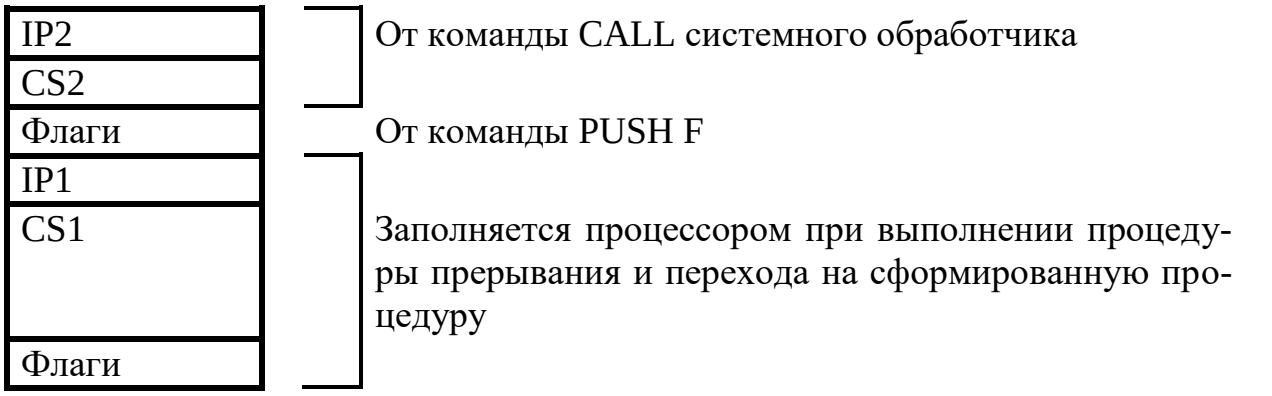
Такая процедура вызывается только из программы пользователя в форме вызова процедуры. При обращении к процедуре в стеке сохраняется адрес возврата к команде, следующей за командой вызова процедуры: таков механизм вызова подпрограмм. Сохранение флагов при этом не предусмотрено, поэтому они сохраняются самим пользователем.

После того, как процессор выполнит процедуру прерывания, в стеке прерванного процесса оказывается слово флагов и адрес возврата в прерванную программу.

Именно такая структура данных должна быть на верху стека, чтобы команда IRET, которой завершается любая программа обработки прерываний, могла вернуть управление в прерванный процесс.

Первая команда нашего обработчика PUSH A засылает в стек еще раз слово флагов, а команда вызова подпрограммы CALL по адресу системного обработчика в процессе передачи управления ему помещает в стек адрес воз-

врата на следующую команду прикладного обработчика. В результате в стеке формируется трехсловная структура, необходимая для команды IRET. Состояние стека при этом получается следующим:



При вызове аппаратно или по команде INT необходимо в векторе сохранить адрес новой созданной процедуры.

Задание № 2.1. Исследование работы нестандартной процедуры обработки прерываний от клавиатуры

Работой клавиатуры управляет контроллер клавиатуры, взаимодействие с которым осуществляется через порты 60h и 61h. При нажатии и отжатии клавиши контроллер фиксирует SCAN-код клавиши (SCAN-коды нажатой и отжатой клавиш отличаются на 80h), который помещается в порт 60h. Считав из порта код нажатой клавиши, необходимо на порт 61h направить подтверждение приема кода клавиши, которое состоит в установке в единицу седьмого бита порта и последующего его сброса.

Наберите следующую программу и изучите ее работу. Уясните принцип обработки аппаратных прерываний.

Программа отслеживает нажатие клавиш перемещения (стрелок) на клавиатуре и передвигает графический объект – квадрат в соответствии с выбранным стрелками направлением.

```
Program int1;
Uses graph, crt;
Var key_tab:array[1..3]of integer;
      xc, yc:integer;
      old: pointer;
{процедура обработки прерываний от клавиатуры}
procedure mykey; interrupt;
var key:integer;
begin
asm
      ; читаем символ из порта
```

```
        in al,60h
        xor ah,ah
        ; запоминаем его в переменной key
        mov key,ax
        ; посылаем подтверждение приема символа
        in al,61h
        or al,80h
        out 61h,al
        and al,7fh
        ; сбрасываем бит в регистре обслуживаемых запросов
        mov al,20h
        out 20h,al
        end;
{заполняем таблицу перемещений в соответствии с заданным направлением}
        case key of
        75{left}: key_tab[1]:=-1;
        77{right}: key_tab[1]:=1;
        72{up}: key_tab[2]:=-1;
        80{down}: key_tab[2]:=1;
        1{esc}: key_tab[3]:=1;
        203{left}, 205{right}: key_tab[1]:=0;
        200{up}, 208{down}: key_tab[2]:=0;
        end;
end;
{конец процедуры обработки прерываний}
BEGIN
Asm
        ; запоминаем адрес старого обработчика в переменной old
        ; 35h – функция получения адреса обработчика с номером в AL
        ;  адрес обработчика передается в регистрах  ES:BX
        mov ah,35h
        mov al,09h
        int 21h
        mov word ptr old,bx
        mov word ptr old+2,es
        ; помещаем в 9-й вектор адрес нашей процедуры-обработчика
        ; адрес обработчика готовится в регистрах DS:DX
        push ds
        mov ax, seg mykey
        mov ds,ax
        mov ah,25h
        mov al,09h
        mov dx, offset mykey
        int 21h
        pop ds
```

```
end;
 {инициализируем графику и установим начальные координаты объекта}
        gd:=detect;
        initgraph(gd,gm,'');
        xc:=200;yc:=200;
{рисуем объект}
        setfillstyle(1,2);
        bar(xc,yc,xc+20,yc+20);
{цикл, пока не нажали на ESC}
        while key_tab[3]<>1 do
        begin
         {закрашиваем объект в старом месте}
        setfillstyle(1,0);
        bar(xc,yc,xc+20,yc+20);
{разрешаем перемещения в пределах окна (100,100,600,400)}
    if ((xc>100)or(key_tab[1]=1))and ((xc<600)or(key_tab[1]=-1))
   then xc:=xc+key_tab[1];
    if((yc>100)or (key_tab[2]=1))and((yc<400)or(key_tab[2]=-1))
    then yc:=yc+key_tab[2];
{перерисовываем объект в новых координатах}
    setfillstyle(1,2);
    bar(xc,yc,xc+20,yc+20);
    delay(10);{задержка, чтобы уменьшить скорость перемещения объекта}
 end;
{возвращаем вектору адрес старого обработчика}
 asm
   push ds
   mov ah,25h
   mov al,09h
   lds dx,old
   int 21h
   pop ds
 end;
end.
```

Задание и вопросы для исследования

1. Сформулируйте методику создания движущихся изображений.

2. Какая часть программы работает при отсутствии нажатия на клавиши, какая при нажатии? Поясните механизм передачи управления.

3. Можно ли активизировать прерывание до замены системного обработчика на пользовательский?

4. Составьте обобщенный (вербальный) алгоритм работы программы с учетом структуры исходного текста.

5. Найдите и исправьте неточность в предложенной программе.

6. Занесите новые справочные данные по прерываниям (функции 25h, 35h).

7. Добавьте второй квадрат, управляемый другими клавишами. Клавиши для управления им выберите самостоятельно.

Задание 2.2. Исследование работы программы с использованием стандартного обработчика

Измените программу таким образом, чтобы аналогичные действия выполнялись с помощью стандартного обработчика. Сформулируйте отличия в работе программ.

Примечание: для чтения нажатой клавиши, прочитанной стандартным обработчиком, используйте функцию ***readkey*** *(функция читает символ из буфера клавиатуры без отображения и сдвига курсора, находится в модуле CRT). Если прочитанный символ равен нулю (ASCII- коды управляющих клавиш равны нулю) используйте функцию повторно для чтения SCAN-кода клавиши.*

```
Program int2;
Uses graph, crt;
Var    key_tab:array[1..3] of integer;
 gm,gd, xc, yc:integer;
BEGIN
gd:=detect;
 initgraph(gd,gm,'');
 xc:=200;yc:=200;
 setfillstyle(1,2);
 bar(xc,yc,xc+20,yc+20);
 while key_tab[3]<>1 do
  begin
    key:=ord(readkey);
    case key of
      75{left}: key_tab[1]:=-1;
      77{right}: key_tab[1]:=1;
      72{up}: key_tab[2]:=-1;
      80{down}: key_tab[2]:=1;
      27{esc}: key_tab[3]:=1;
      else for I:=1 to 3 do key_tab[I]:=0;
    end;
     setfillstyle(1,0);
     bar(xc,yc,xc+20,yc+20);
     if ((xc>100)or(key_tab[1]=1))and ((xc<600)or(key_tab[1]=-1))
    then xc:=xc+key_tab[1];
     if((yc>100)or (key_tab[2]=1))and((yc<400)or(key_tab[2]=-1))
```

```
      then yc:=yc+key_tab[2];
      setfillstyle(1,2);
      bar(xc,yc,xc+20,yc+20);
    end;
  end.
```

Задание и вопросы для исследования

1. Составьте обобщенный вербальный алгоритм работы программы.
2. Найдите и исправьте имеющиеся неточности в программе.

Задание 2.3. Исследование особенностей перехвата прерываний таймера и их использование

Сигналы прерывания таймера возникают 18,2 раз в секунду. В векторе 08h содержится адрес программы BIOS отсчета системного времени. Для того, чтобы программы пользователя могли использовать прерывания таймера, не нарушая работу системных часов, в обработчик BIOS включен вызов прерывания 1Ch, передающий управление на программу-заглушку. Перехватить прерывание таймера можно, поместив в вектор 1Ch адрес своего обработчика.

Задание 2.3.1. Используйте прерывания таймера для выдачи на экран (в определенное место) каждые 5 секунд количества секунд, прошедших с момента запуска программы.

```
program int2;
 uses graph,crt;
var
   old:pointer;
   t,n,x1,x2,y1,y2,gm,gd:integer;
  procedure mytime1;interrupt;
   begin
    t:=t+1;
  end;
 begin
   asm
            mov ah,25h
            mov al,1ch
            int 21h
            mov word ptr old,bx
            mov word ptr old+2,es
            push ds
            mov ax,seg mytime1
            mov ds,ax
            mov ah,25h
```

```
            mov al,1ch
            mov dx,offset mytime1
            int 21h
            pop ds
  end;
  n:=0;
 while n<10 do
  if t>90 then
   begin
    writeln (t/18.2:8:0 );
    t:=0;  n:=n+1;
   end;
asm
            push ds
            mov ah,25h
            mov al,1ch
            lds dx,old
            int 21h
            pop ds
 end;
  end.
```

Задание 2. 3.2. Используйте прерывания таймера для организации квазипараллельной работы потоков. Пусть один поток рисует горизонтальную линию красного цвета, начиная с координат (1, 100), другой – линию синего цвета, начиная с координат (1,200). Для переключения между потоками используйте флаг, изменяющий свое состояние через заданные кванты времени.

Примечание: Перемещение курсора в позицию с координатами (x,y) выполняется процедурой moveto(x,y). Рисование линии из текущей позиции в позицию (x,y) выполняется процедурой lineto(x,y). Установка цвета линии – процедура setcolor(<номер цвета>).

А) Выделите потокам одинаковые кванты времени.

Б) Выделите потокам различные кванты времени.

Решение

```
program int3;
 uses graph,crt;
var
  old:pointer;
  t,n,x1,x2,y1,y2,gm,gd:integer;
  flag:boolean;
 procedure mytime1;interrupt;
  begin
   t:=t+1;
   if t>1 then
```

```
    begin
     flag:= not flag;
     t:=0;
   end;
  end;
 begin
  asm
                mov ah,25h
                mov al,1ch
                int 21h
                mov word ptr old,bx
                mov word ptr old+2,es
                push ds
                mov ax,seg mytime1
                mov ds,ax
                mov ah,25h
                mov al,1ch
                mov dx,offset mytime1
                int 21h
                pop ds
  end;
  gd:=detect;
   initgraph(gd,gm,'');
   x1:=10; y1:=100;
   x2:=10; y2:=200;
  while (x2<640) and (x1<640) do
  begin
   if flag  then
    begin
     setfillstyle(1,4);
     bar(x1,y1,x1+1,y1+50);
     x1:=x1+1;
    end
  else
   begin
     setfillstyle(1,1);
     bar(x2,y2,x2+1,y2+50);
      x2:=x2+1;
   end;
   delay(20);
  end;
 readln;
closegraph;
 asm
                push ds
```

```
            mov ah,25h
            mov al,1ch
            lds dx,old
            int 21h
            pop ds
end;
  end.
```

Задание и вопросы для исследования

1. Нарисуйте обобщенный вербальный алгоритм работы программы.
2. Найдите и исправьте неточности в программе.

Контрольные вопросы (ПК-11):

1. Поясните механизм реализации процедуры прерывания, проиллюстрируйте ее.
2. Приведите классификацию прерываний.
3. В чем различие между программными и аппаратными прерываниями?
4. Поясните назначение вектора прерываний, способ его использования.
5. Каким образом осуществляется обращение из прикладной программы к системным функциям?
6. Поясните процесс взаимодействия системы с клавиатурой используя иллюстрации.
7. Поясните понятие SCAN-кода и кода ASCII.
8. Для чего предназначена программа обработки прерываний от клавиатуры?
9. Какой вектор прерываний закреплен за клавиатурой?
10. Для чего в составе машин IBM PC имеется таймер?
11. Поясните прикладное значение обработки прерываний от таймера.
12. Какие функции прерывания 21h DOS используются для чтения и изменения текущего времени и даты?
13. Поясните организацию аппаратных прерываний в машинах IBM PC.
14. Как образована очередь в буфере клавиатуры?