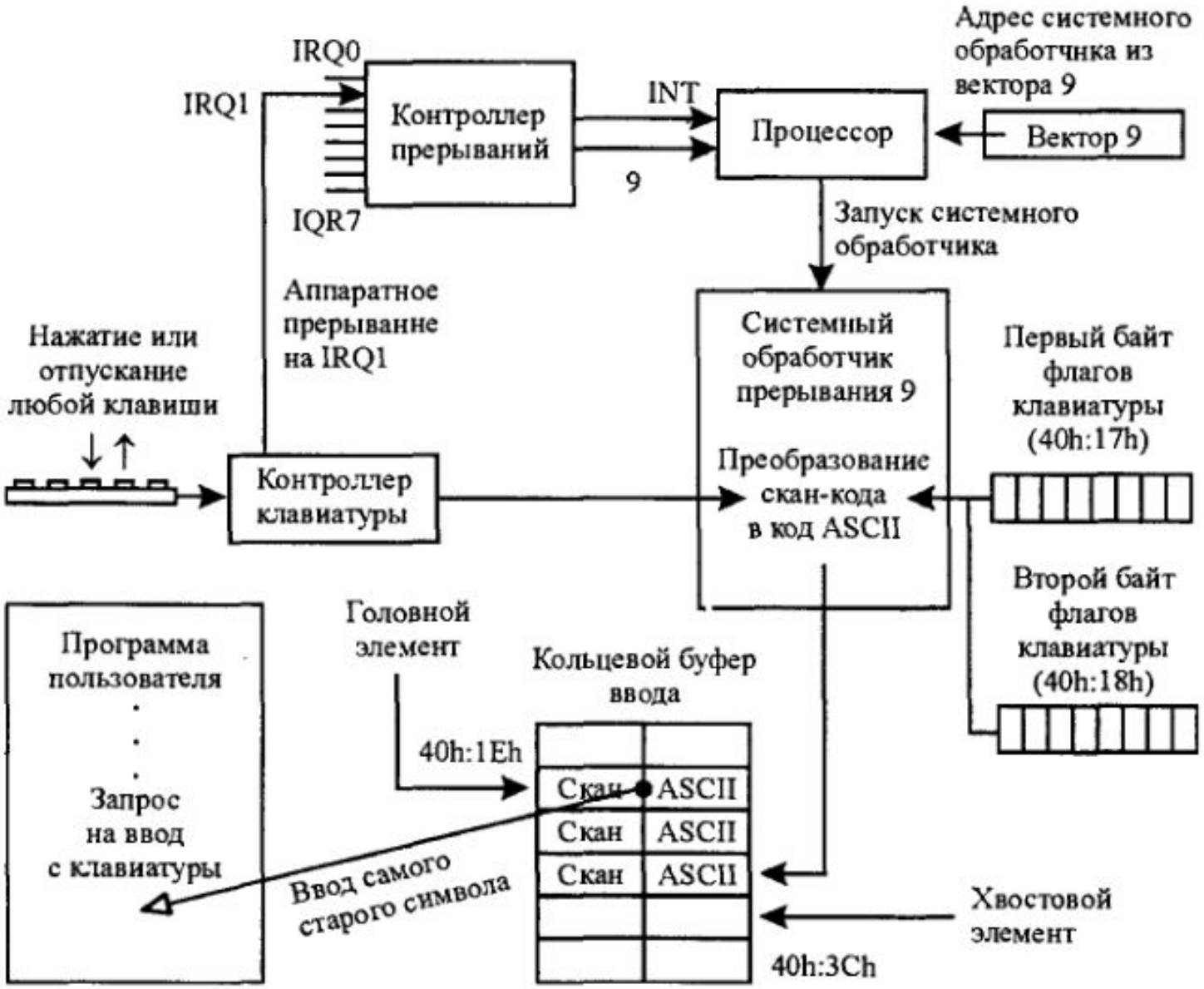
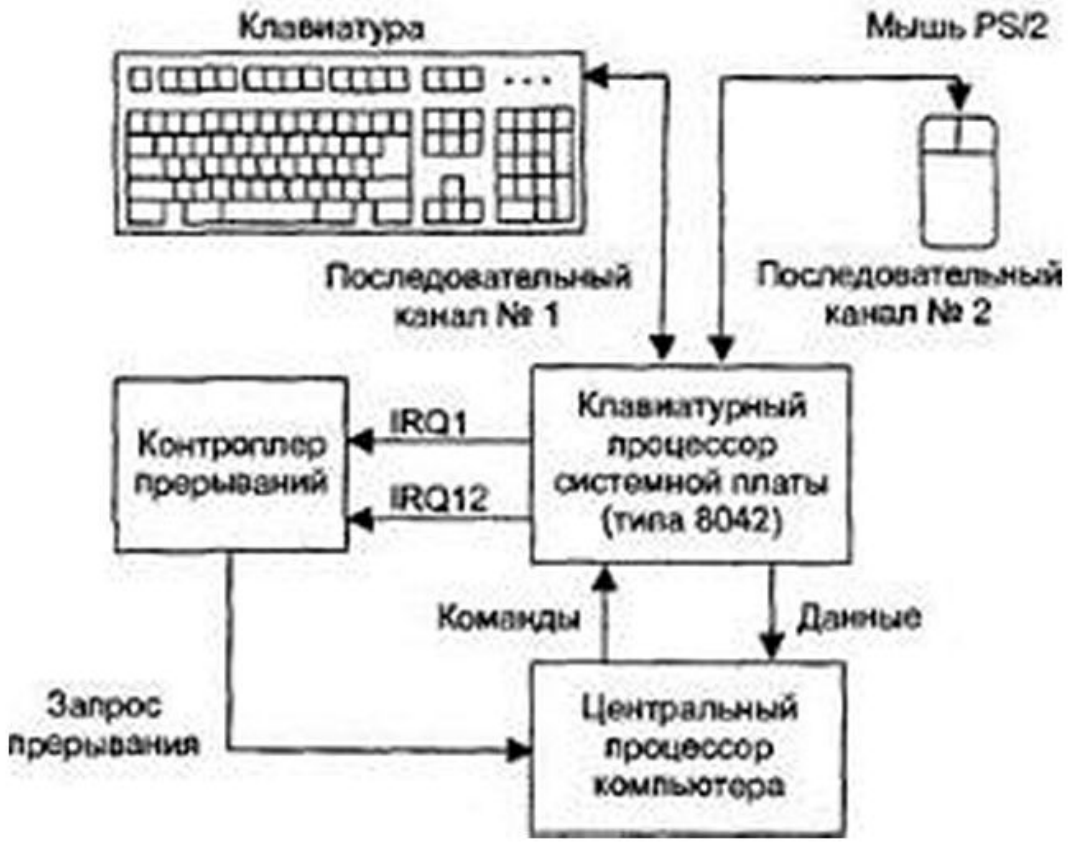

IRQ0
IRQ1
IQR7
Контроллер прерываний
INT
9
Процессор
Адрес системного обработчнка из вектора 9
Вектор 9
Запуск системного обработчика
Аппаратное прерываннe на IRQ1
Нажатие или отпускание любой клавиши
Системный обработчик прерывания 9
Преобразование скан-кода в код ASCII
Первый байт флагов клавиатуры (40h:17h)
Второй байт флагов клавиатуры (40h:18h)
Контроллер клавиатуры
Программа пользователя
Запрос на ввод с клавиатуры
Головной элемент
Кольцевой буфер ввода
40h:1Eh
Скан
ASCII
Ввод самого старого символа
Хвостовой элемент
40h:3Ch

Управление работой контроллера и обмен данными с мышью и клавиатурой осуществляются при помощи трех регистров: регистра состояния, регистра команд и регистра данных. Кроме того, при поступлении информации от клавиатуры контроллер i8042 вырабатывает прерывание IRQ.1, а при приеме данных от мыши — IRQ12. Интерфейсы клавиатуры и мыши аналогичны, наборы команд управления также имеют некоторое сходство.



- Программа int09, помимо порта 60h, работает еще с двумя областями оперативной памяти: кольцевым буфером ввода, располагаемым по адресам от 40h:1Eh до 40h:3Ch, куда в конце концов помещаются коды ASCII нажатых клавиш, и 2 байтами флагов клавиатуры, находящимися по адресам 40h:17h и 40h:18h. В этих байтах фиксируется состояние управляющих клавиш (Shift, Caps Lock, Num Lock и др.).
- *64h для чтения* - регистр состояния клавиатуры, возвращает следующий байт:
- бит 1: в буфере ввода есть данные (для контроллера клавиатуры)
- бит 0: в буфере вывода есть данные (для компьютера)
- При записи в этот порт он играет роль дополнительного регистра управления. клавиатурой, но его команды сильно различаются для разных плат

Однако имеется ряд клавиш, которым не назначены отображаемые на экране символы. Это, например, функциональные клавиши F1...F12; При нажатии этих клавиш в кольцевой буфер ввода засылается расширенный код ASCII, в котором младший байт равен нулю, а старший является скан-кодом нажатой клавиши. Расширенные коды ASCII поступают в буфер ввода и в случае нажатия комбинаций управляющих и функциональных клавиш, например Shift+FI, Alt+Insert и др. В этом случае, однако, в старший байт расширенного кода ASCII помещается уже не скан-код клавиши, а некоторый код, специально назначенный этой комбинации клавиш.

60h для записи - регистр управления клавиатурой. Байт, записанный в этот порт (если бит 1 в порту 61 h равен 0). Интерпретируется как команда. Некоторые команды состоят из более чем одного байта - тогда следует дождаться обнуления этого бита еще раз перед тем, как посылать следующий байт.

Команда OEDh 0?h - изменить состояние светодиодов клавиатуры. Второй байт этой команды определяет новое состояние:
бит 0: состояние Scroll Lock (1 - включена, 0 - выключена)
бит 1: состояние Num Lock
бит 2: состояние Caps Lock
При этом состояние переключателей, которое хранит BIOS в байтах состояния клавиатуры, не изменяется, и при первой возможности обработчик прерывания клавиатуры BIOS восстановит состояние светодиодов.

Перед началом работы с клавиатурой следует проверить наличие данных в буфере (бит 0 в регистре статуса). Кроме того, такую проверку необходимо выполнить перед любыми последующими операциями записи.
После проверки буфера в регистр 64h записывается код желаемой команды
Команда *EEh*
Команда позволяет протестировать клавиатуру на предмет работоспособности. Если в работе клавиатуры возникли сбои, следует сделать сброс (команда FFh) и послать эту команду. Возвращаемое значение, отличное от EEh. явно укажет на сбои в работе клавиатуры.
Команда *F2h*
Эта команда позволяет получить идентификатор клавиатуры и убедиться в ее наличии. После выполнения команды клавиатура вернет код подтверждения FAh, а затем идентификатор.

Листинг 3.11. Проверка наличия клавиатуры

```
; ждем освобождения входного буфера клавиатуры
@keyb_wait:
in   AL, 64h    ; опрашиваем регистр состояния
test AL, 010b   ; если буфер занят
jnz  @keyb_wait; повторяем опрос
mov  AL, 0ABh   ; команду тестирования интерфейса
out  64h, AL    ; записываем в порт
```

Для программистов C++ этот пример представлен в листинге 3.12.

Листинг 3.12. Проверка наличия клавиатуры в C++

```
DWORD dwResult  = 0; // переменная для хранения результата
int iTimeWait = 50000;
// проверяем наличие данных во входном буфере клавиатуры
while ( -- iTimeWait > 0)
{
        // читаем состояние порта
        inPort ( 0x64, &dwResult, 1);
        if ( ( dwResult & 0x02) == 0x00) break;
        // закончилось время ожидания
        if ( iTimeWait < 1) return MY_ERROR_TIME;
}
// записываем в порт команду проверки интерфейса
outPort ( 0x64, 0xAB, 1);
```

Листинг 3.15. Управление индикатором <Num Lock>

```
@keyb_wait:
in   AL, 64h    ; опрашиваем регистр состояния
test AL, 010b   ; если буфер занят
jnz  @keyb_wait; повторяем опрос
mov  AL, 0EDh   ; команду управления индикаторами
out  60h, AL    ; записываем в порт данных
@data_wait:
in   AL, 64h    ; опрашиваем регистр состояния
test AL, 010b   ; если буфер занят
jnz  @data_wait; повторяем опрос
mov  AL, 010b   ; включаем Num Lock
out  60h, AL    ; записываем в порт значение
```

```
DWORD dwResult  = 0; // переменная для хранения результата
int iTimeWait = 50000;
// проверяем наличие данных во входном буфере клавиатуры
while ( -- iTimeWait > 0)
{
        // читаем состояние порта
        inPort ( 0x64, &dwResult, 1);
        if ( (dwResult & 0x02) == 0x00) break;
        // закончилось время ожидания
        if ( iTimeWait < 1) return MY_ERROR_TIME;
}
 // записываем в порт команду управления индикаторами
 outPort ( 0x60, 0xED, 1);
 int iTimeWait = 50000;
 // проверяем наличие данных во входном буфере клавиатуры
 while ( -- iTimeWait > 0)
 {
        // читаем состояние порта
        inPort ( 0x64, &dwResult, 1);
        if ( (dwResult & 0x02) == 0x00) break;
        // закончилось время ожидания
        if ( iTimeWait < 1) return MY_ERROR_TIME;
 }
 outPort ( 0x60, 0x02, 1);
```