Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Отчет по лабораторной работе №1 (часть 1) по курсу "Операционные системы"

Тема Исследование прерывания INT 8h

Студент Чепиго Д.С.

Группа ИУ7-54Б

Преподаватель Рязанова Н.Ю.

Москва — 2022 г.

Листинг кода

Листинг INT8h

```
temp.lst                 Sourcer v5.10     15-Sep-22   10:20 am    Page 1

; вызов подпрограммы sub_2, запрещает маскируемые прерывания
; от внешних устройств.
020A:0746   E8 0070       ;*       call     sub_2            ; (07B9)

020A:0746   E8 70 00                db   0E8h, 70h, 00h

; сохранение аппаратного контекста в стек.
020A:0749   06                      push     es
020A:074A   1E                      push     ds
020A:074B   50                      push     ax
020A:074C   52                      push     dx

; помещаем в ds адрес сегмента данных BIOS, используя буфер ax.
020A:074D   B8 0040                 mov ax,40h
020A:0750   8E D8                   mov ds,ax

; обнуление es через буфер ax, то есть помещаем в es
; адрес начала таблицы векторов прерываний.
020A:0752   33 C0                   xor ax,ax            ; Zero register
020A:0754   8E C0                   mov es,ax

; в сегменте данных BIOS по смещению 6Ch находится
; счетчик реального времени (младшая часть 2 байта).
; инкрементируем младшую часть.
020A:0756   FF 06 006C                inc word ptr ds:[6Ch]    ;
    (0040:006C=88B4h)

; если младший разряд не стал равным 0,
; то есть нет переполнения, то переходим в loc_1.
020A:075A   75 04                   jnz loc_1            ; Jump if not zero

; иначе, добавляем единицу к следующим 2 байтам
; по смещению 6Eh счетчика реального времени.
; далее это старшие два байта.
020A:075C   FF 06 006E                inc word ptr ds:[6Eh]    ;
    (0040:006E=0Bh)

```

```
020A:0760           loc_1::
; прерывание происходит 18.2 раз в секунду, за 1 час 65536 раз (FFFFh).
; старшие два байта увеличиваются, когда переполняются младшие,
; а в младших максимальное число == 65536, это счетчик часов.

; сравниваем 2 старших байта счетчика реального времени с 18h = 24.
020A:0760  83 3E 006E 18            cmp word ptr ds:[6Eh],18h   ;
    (0040:006E=0Bh)

; если не равны, то переход на loc_2.
020A:0765  75 15                    jne loc_2           ; Jump if not equal

; счетчик часов равен 24. сравниваем два младших байта с 0B0h == 176.
; это погрешность тиков, чтобы сутки считались правильно.
; так как сутки раньше на  9.666884762 секунд,
; то есть  18.206481481 = 175.999958398 тиков.

020A:0767  81 3E 006C 00B0          cmp word ptr ds:[6Ch],0B0h  ;
    (0040:006C=88B4h)


; если сутки ещё не прошли, то переходим в loc_2.
020A:076D  75 0D                    jne loc_2           ; Jump if not equal

; иначе прошли полноценные сутки. обнуляем счетчик часов и тиков.
020A:076F  A3 006E                  mov word ptr ds:[6Eh],ax    ;
    (0040:006E=0Bh)
020A:0772  A3 006C                  mov word ptr ds:[6Ch],ax    ;
    (0040:006C=88B4h)

; фиксируем, что прошли сутки по адресу 70h - флаг 1.
020A:0775  C6 06 0070 01            mov byte ptr ds:[70h],1 ;
    (0040:0070=0)

; записываем в al 8 (al = 1000).
020A:077A  0C 08                    or  al,8

; в loc_2  мы переходим, если прошли сутки и мы обнулили счетчики,
; либо если сутки не прошли.
; мы завершили инкремент счетчика реального времени.
; начинаем декремент счетчика реального времени до
; отключения моторчика дисковода.
020A:077C           loc_2::

; сохраняем ax, где al = 8, если наступили новые сутки и 0 иначе
020A:077C  50                       push    ax

; по смещению 40h хранится время до отключения моторчика дисковода.
```

```
; декрементируем.
020A:077D  FE 0E 0040              dec byte ptr ds:[40h]    ;
    (0040:0040=0D4h)

; если не равно 0, то переходим в loc_3.
020A:0781  75 0B                   jnz loc_3                ; Jump if not zero

; таймер == 0. по смещению 3Fh хранится состояние дисковода.
; 4 младших бита - состояние четырех приводов моторчика дисковода.
; 0F0h = 1111 0000, то есть мы устанавливаем для всех приводов статус
    "выключен".
020A:0783  80 26 003F F0           and byte ptr ds:[3Fh],0F0h   ;
    (0040:003F=0)

; если в старших 4 битах четное количество единиц,
; то PF == 1, иначе PF == 0.
; в al кладем сообщение, посылаемое в порт.
; в dx - номер порта. 0Ch = 0000 1100. 0Ch - сигнал отключение дисковода.
; 2-ой бит - разрешение работы контроллера,
; 3 бит - разрешение прерывания прямого доступа к памяти.
; 3F2h - порт управления дисководом.
020A:0788  B0 0C                   mov al,0Ch
020A:078A  BA 03F2                 mov dx,3F2h
020A:078D  EE                      out dx,al                ; port 3F2h, dsk0
    contrl output

; переходим, если мы отключили дисковод, или его еще не надо отключать.
020A:078E           loc_3::
; возврат регистра ax, где al = 8, если наступили новые сутки и 0 иначе.
020A:078E  58                      pop ax

; сравниваем флаги и число 4 = 100, то есть флаг PF(Parity Flag).
; то есть проверяем разрешены ли маскируемые прерывания.
020A:078F  F7 06 0314 0004         test    word ptr ds:[314h],4     ;
    (0040:0314=3200h)

; если не 0, то переходим в loc_4.
020A:0795  75 0C                   jnz loc_4                ; Jump if not zero

; команда LAHF сохраняет младший байт регистра флагов
; в AH(старший байт AX) сохраняем, так как TEST меняет флаги.
020A:0797  9F                      lahf                     ; Load ah from flags


; меняем ah и al, то есть в ax теперь лежит регистр флагов,
; в котором младщий байт как оригинальные флаги, а старший обнулен.
; но если прошли сутки, то установлен 11 бит - OF (Overflow Flag).
020A:0798  86 E0                   xchg    ah,al
```

```
124
125 ; помещаем ax в стек.
126 020A:079A  50                    push    ax
127
128 ; 70h / 4 = 1Ch вызов прерывания 1Ch напрямую.
129 020A:079B  26: FF 1E 0070          call    dword ptr es:[70h]  ;
    (0000:0070=6ADh)
130
131 ; переход в loc_5
132 020A:07A0  EB 03                   jmp short loc_5     ; (07A5)
133 020A:07A2  90                      nop
134
135
136 020A:07A3          loc_4::
137 ; int 1Ch - программное прерывание, внутри - IRET.
138 020A:07A3  CD 1C                   int 1Ch         ; Timer break (call each
    18.2ms)
139
140 020A:07A5          loc_5::
141 ; мы могли испорить IF в int 1Ch, ибо это прерывание
142 ; определяется пользователем. надо ещё раз сбросить IF.
143 020A:07A5  E8 0011                 call    sub_2           ; (07B9)
144
145 ; порт 20h - ведущий контроллер прерываний. код 20h,
146 ; который мы отправляем в контроллер прерываний, означает,
147 ; что работа прерывания завершена, и процессор готов принимать прерывания.
148 020A:07A8  B0 20                   mov al,20h          ; ' '
149 020A:07AA  E6 20                   out 20h,al          ; port 20h, 8259-1
    int command
150                                    ;  al = 20h, end of interrupt
151
152 ; восставновление аппаратного контекста.
153 020A:07AC  5A                      pop dx
154 020A:07AD  58                      pop ax
155 020A:07AE  1F                      pop ds
156 020A:07AF  07                      pop es
157
158 ; переход в адрес 07B0 - 164 = 064C
159 020A:07B0  E9 FE99                 jmp $-164h
160 020A:07B3  C4                      db  0C4h
161                                                    ;* No entry point to
                                                          code
162 020A:07B4  C4 0E 93E9              les cx,dword ptr ds:[93E9h] ;
    (0000:93E9=3302h) Load 32 bit ptr
163 020A:07B8  FE                      db  0FEh
164
165 ; возврат из прерывания.
166 020A:06AC  CF              iret     ; Interrupt return
```

Листинг sub_2

```
1
2             temp.lst      Sourcer v5.10     15-Sep-22   10:20 am  Page 2
3
4                  ;                                SUBROUTINE
5
6  ; подпрограмма sub_2. proc near - ближний переход внутри одного сегмента.
7                  sub_2        proc     near
8  ; сохраняем аппаратный контекст в стек.
9  020A:07B9  1E                    push     ds
10 020A:07BA  50                    push     ax
11
12 ; помещаем в ds адрес сегмента данных BIOS, используя буфер ax.
13 020A:07BB  B8 0040               mov ax,40h
14 020A:07BE  8E D8                 mov ds,ax
15
16 ; команда LAHF сохраняет младший байт регистра флагов
17 ; в AH(старший байт AX) сохраняем, так как TEST меняет флаги.
18 020A:07C0  9F                    lahf                ; Load ah from flags
19
20 ; команда TEST сравнивает между собой два числа, но меняет флаги, а не их.
21 ; Флаги SF, ZF, PF устанавливаются в соответствии с результатом.
22 ; 2400h = 0010 0100 0000 0000. 10 бит - DF(Direction Flag),
23 ; 13 - IOPL(input/output privilege level).
24 020A:07C1  F7 06 0314 2400          test    word ptr ds:[314h],2400h    ;
       (0040:0314=3200h)
25
26 ; если DF != 0 или старший бит IOPL != 0, то переходим на loc_7.
27 020A:07C7  75 0C                 jnz loc_7           ; Jump if not zero
28
29 ; если DF == 0 и старший бит IOPL == 0, то надо
30 ; принудительно сбрасывать IF, через CLI не получится.
31 ; FDFFh = 1111 1101 1111 1111. '0' это 9 бит, IF - Interruption flag.
32 ; команда LOCK блокирует локальную шину памяти
33 ; на время выполнения следующей команды.
34 ; AND - выполняет логическое И между всеми битами
35 ; и результат записывает в первый операнд.
36 020A:07C9  F0> 81 26 0314 FDFF                          lock and word
       ptr ds:[314h],0FDFFh    ; (0040:0314=3200h)
37
38 020A:07D0           loc_6::
39 ; команда SAHF загружает младший байт флагов из AH.
40 020A:07D0  9E                    sahf                ; Store ah into flags
41
42 ; восстанааливаем аппаратный контекст из стека.
43 020A:07D1  58                    pop ax
```

```
020A:07D2  1F                    pop ds

; переход в loc_8.
020A:07D3  EB 03                 jmp short loc_8      ; (07D8)

; если IOPL == 10 или 11.
020A:07D5            loc_7::
; CLI - Clear Interrupt-Enable Flag. Сбрасывает Interrupt flag (IF).
020A:07D5  FA                    cli                  ; Disable interrupts

; восставновление и выход.
020A:07D6  EB F8                 jmp short loc_6      ; (07D0)
020A:07D8            loc_8::

; возврат из подпрограммы.
; retn вытаскивает из стека IP, переходит на следующую после call команду.
020A:07D8  C3                    retn
                 sub_2       endp
```

Схема алгоритма

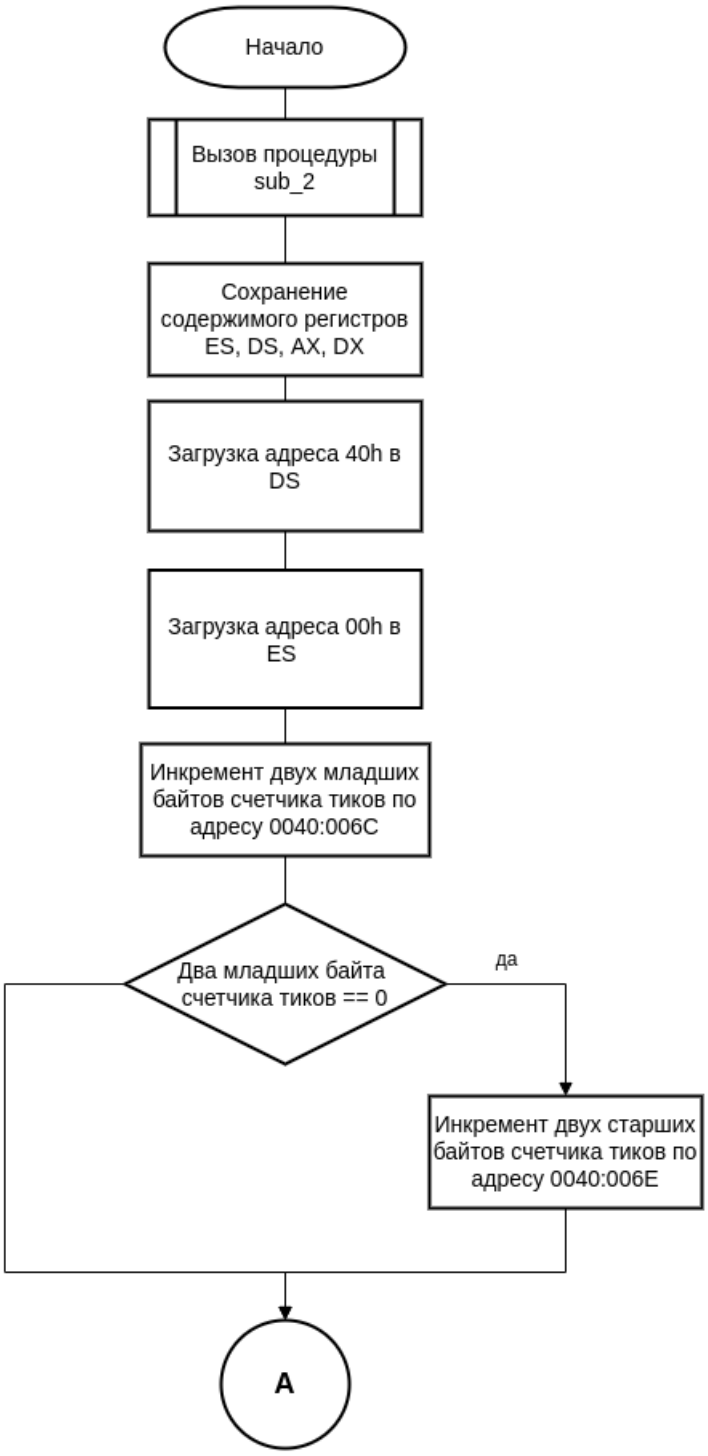

Рисунок 1 – Схема обработчика прерываний INT 8h

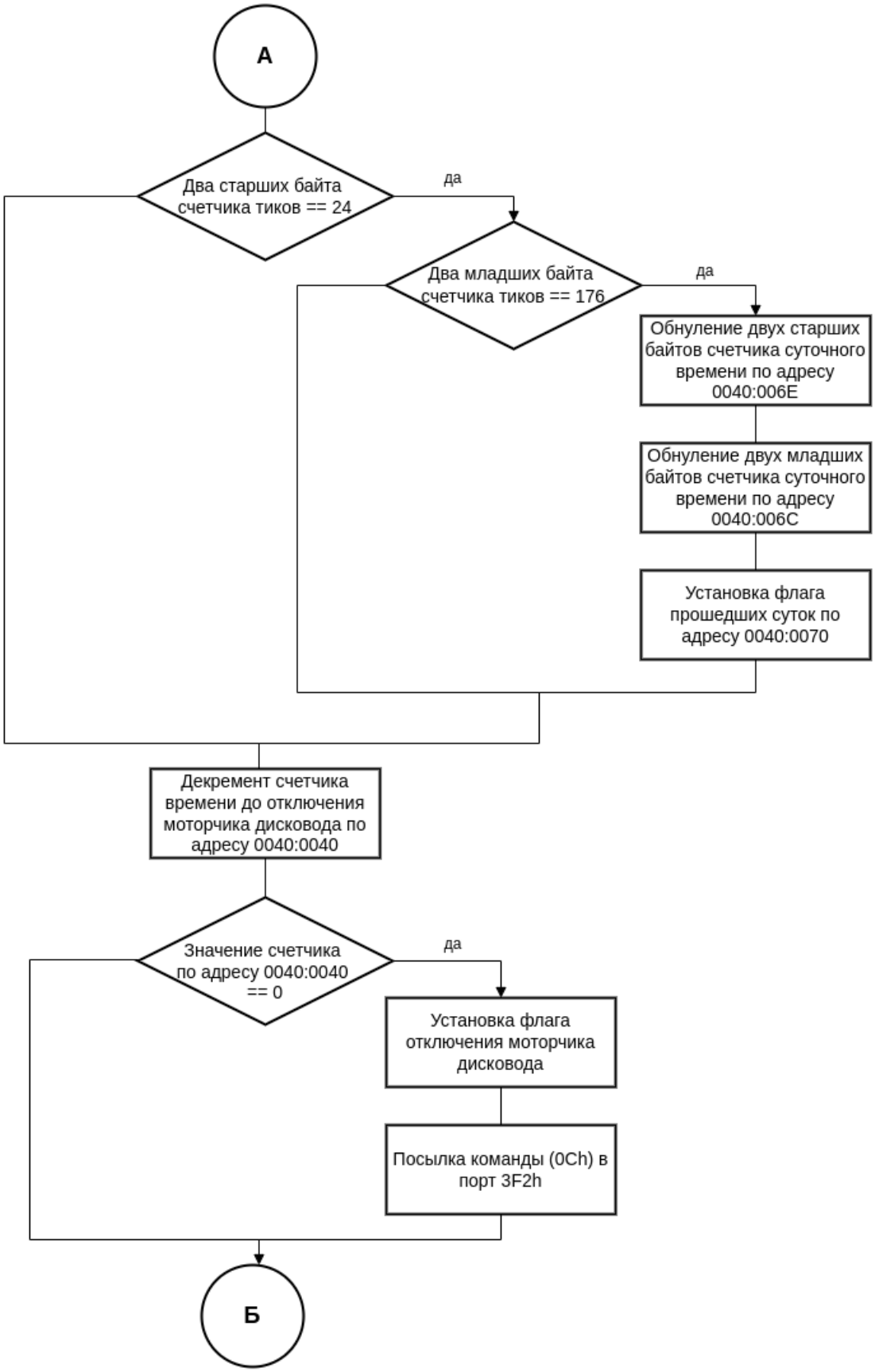

Рисунок 2 – Схема обработчика прерываний INT 8h

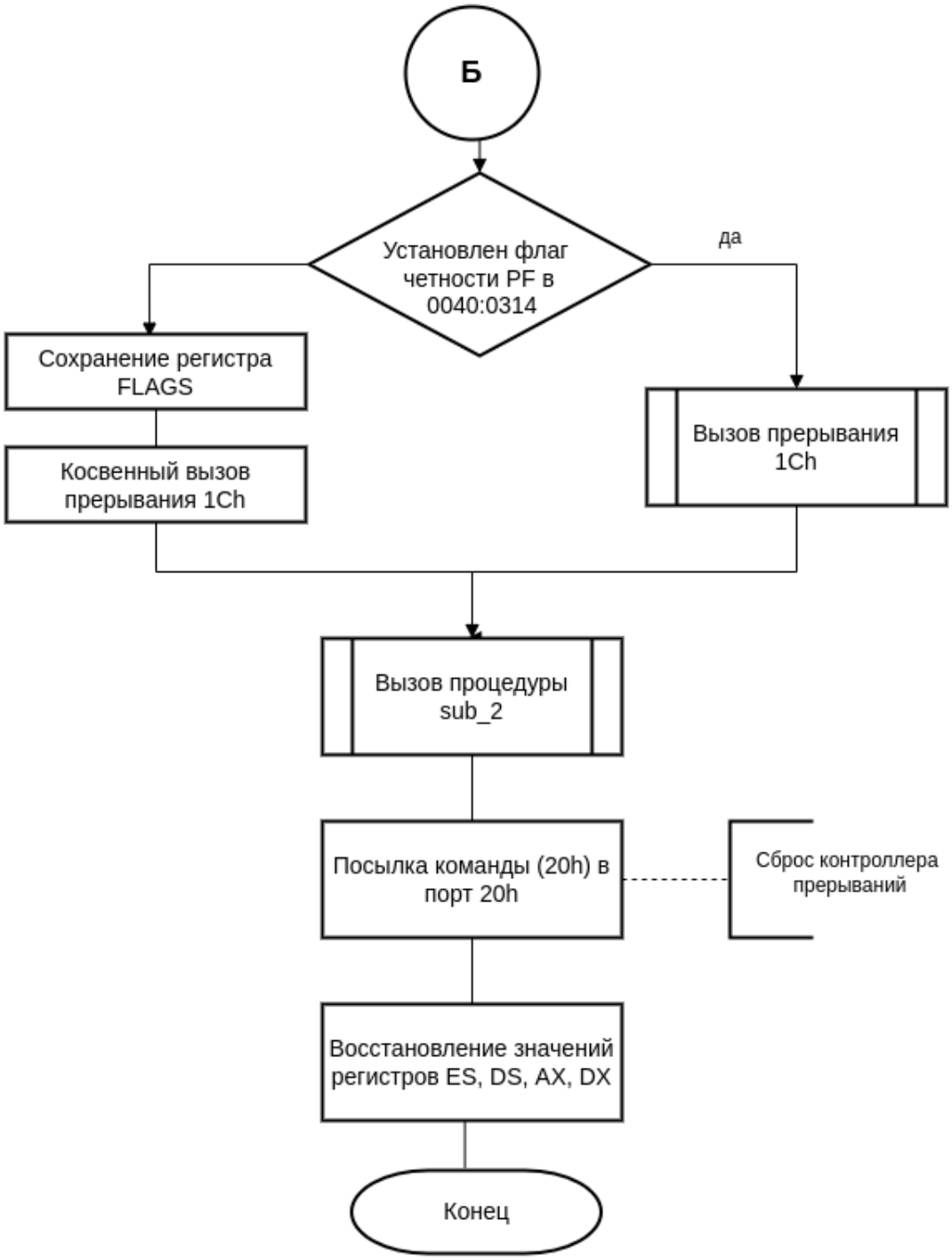

Рисунок 3 – Схема обработчика прерываний INT 8h

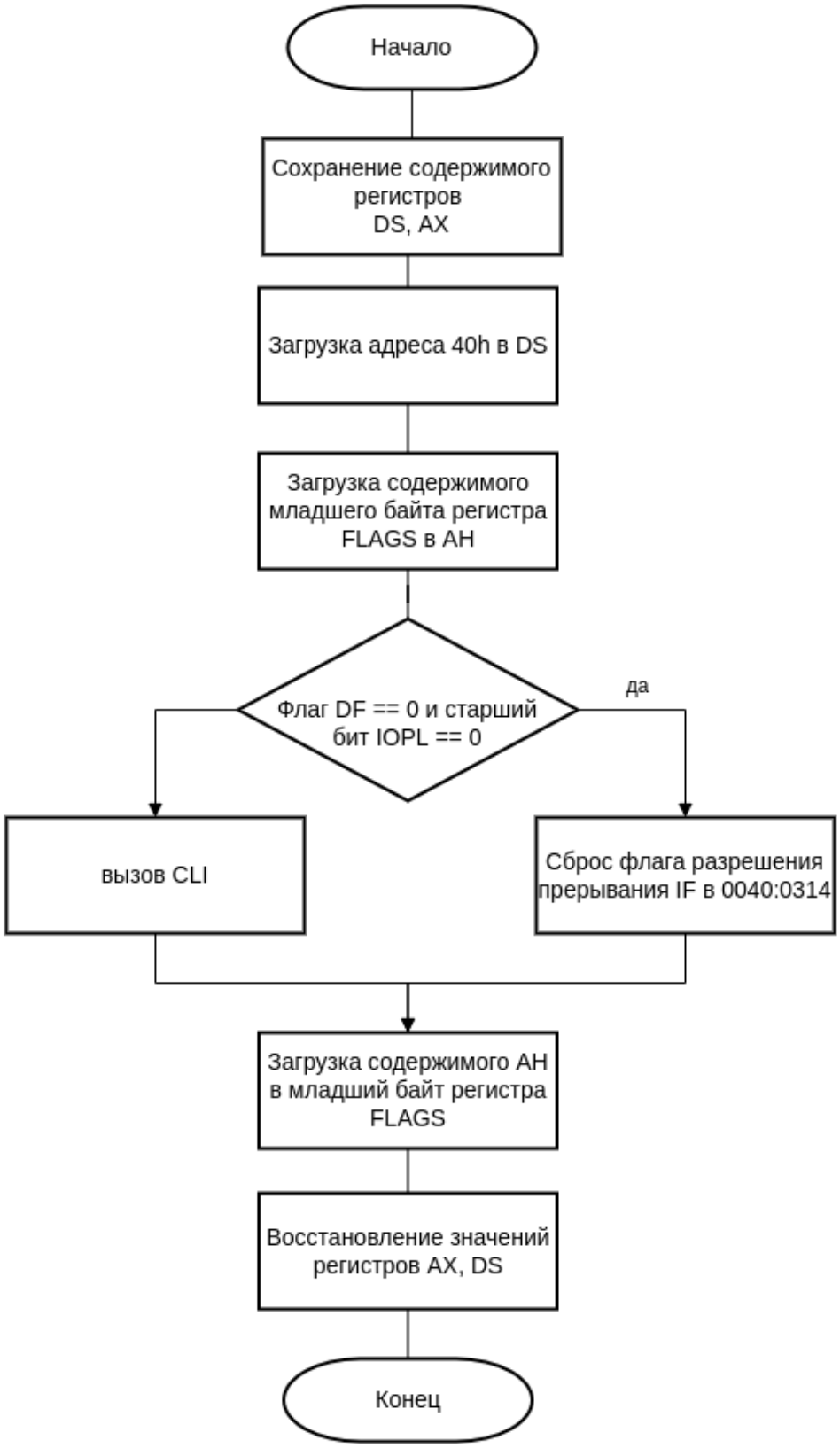

Рисунок 4 – Схема подпрограммы sub_2