# Прерывания. Система прерываний. Реальный режим процессора

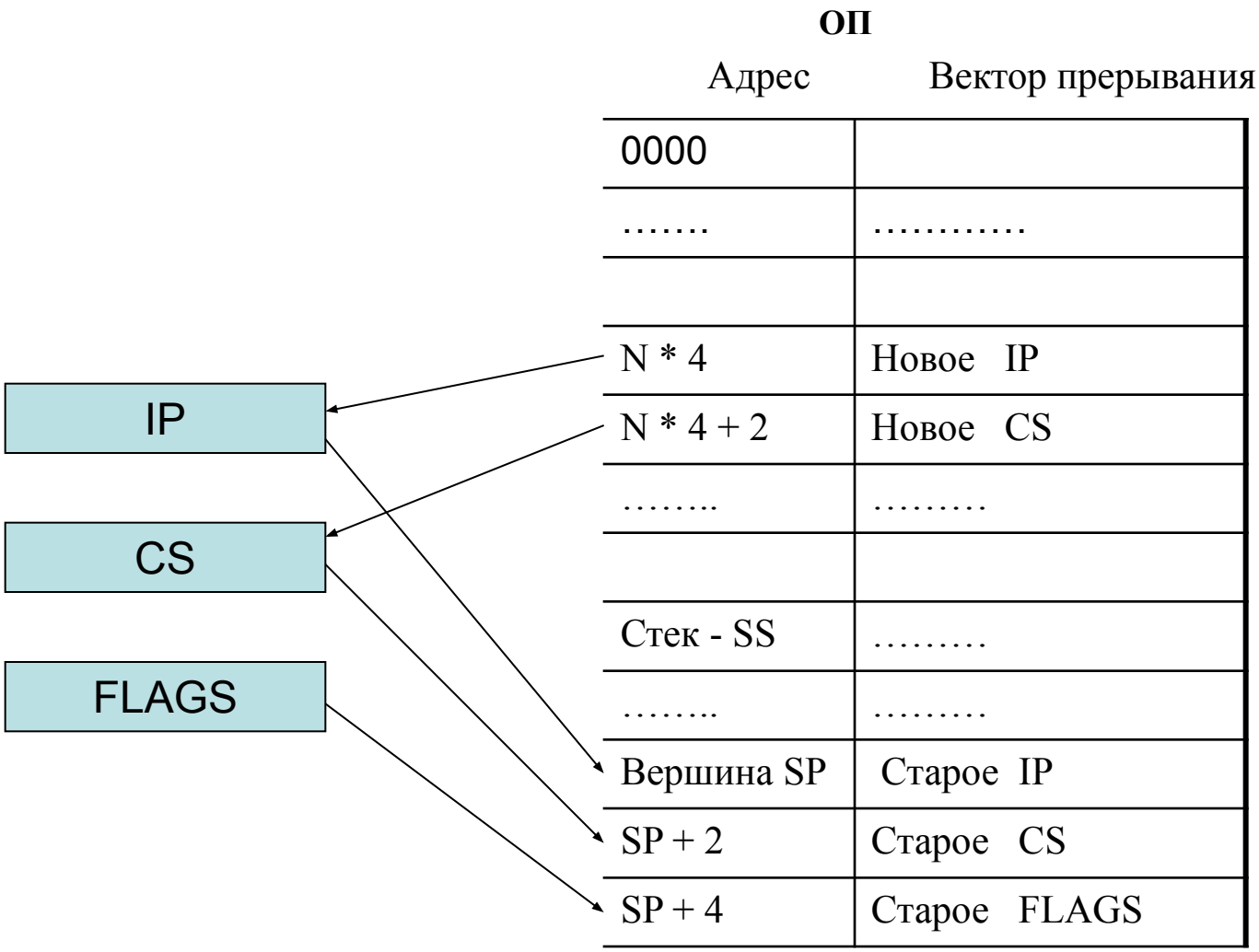
В большинстве современных процессоров имеются средства прерывания их текущей работы внешними устройствами или при возникновении особых случаев, а также прерывания текущей программы. Они освобождают процессор от периодической проверки (полинг) необходимости обслуживания устройств. Различают немаскируемые прерывания по входу NMI (например при падении напряжения внешнего питания до 100 вольт, то есть отказ сети или нарушение четности в канале ввода/вывода) и маскируемые прерывания по входу INTR, которые генерируются контроллером прерываний I8259A по заявке определённых внешних устройств (ВУ). Маскируемые прерывания (их еще называют аппаратными) обрабатываются процессором в случае, если флаг IF в регистре FLAGS установлен в 1 (команда STI - внешние прерывания разрешены), если IF = 0 (CLI)- обработка прерываний от внешних устройств запрещена (отложена до установки IF в 1). ВУ должно сообщить причину прерывания ( код или тип прерывания от 0 до 255).

Для каждого типа прерывания в системе имеется программа обработки данного прерывания. Это может быть программа ОС, BIOS или пользовательская. Адреса этих программ находятся в 256 – элементной таблице, каждый элемент которой состоит из 4 байт и содержит полный указатель (вектор) к началу соответствующей программы обработки прерывания ( значения IP и CS). Таблица векторов прерываний начинается с адреса 0 оперативной памяти. Программы обработки прерываний также называются обработчиками прерываний.

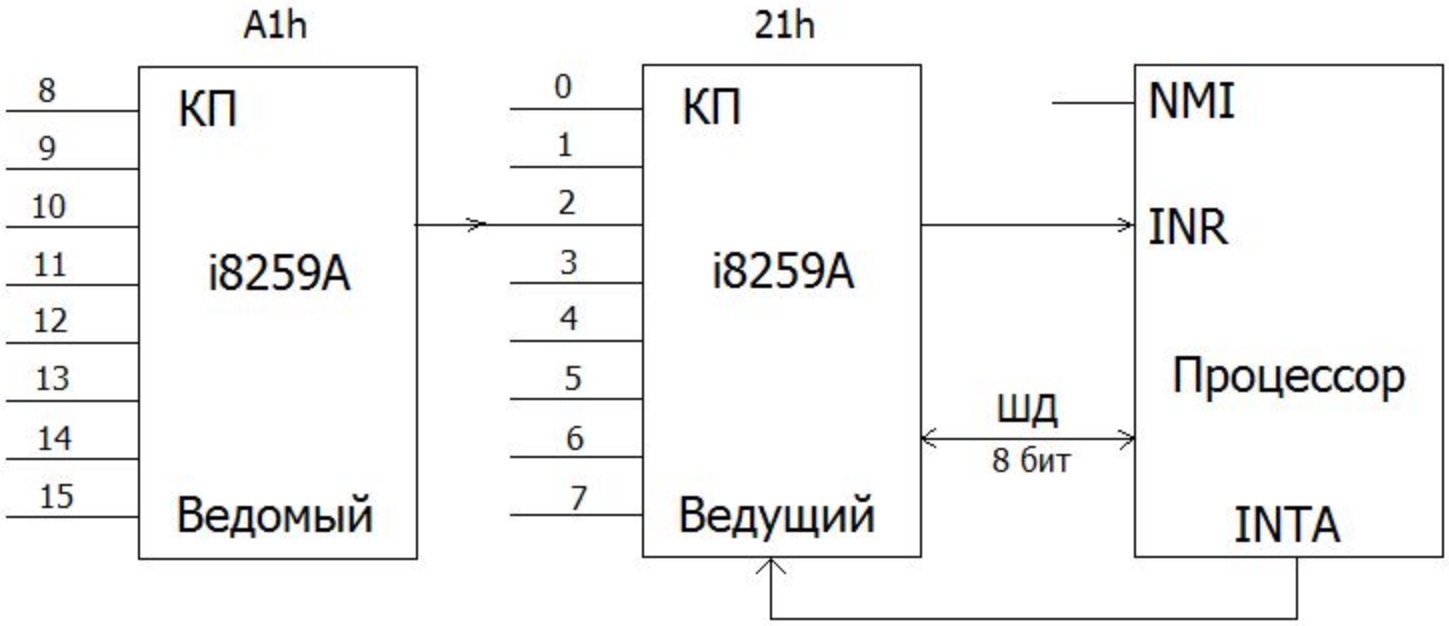
В соответствии с типом прерывания выполняется следующая процедура:

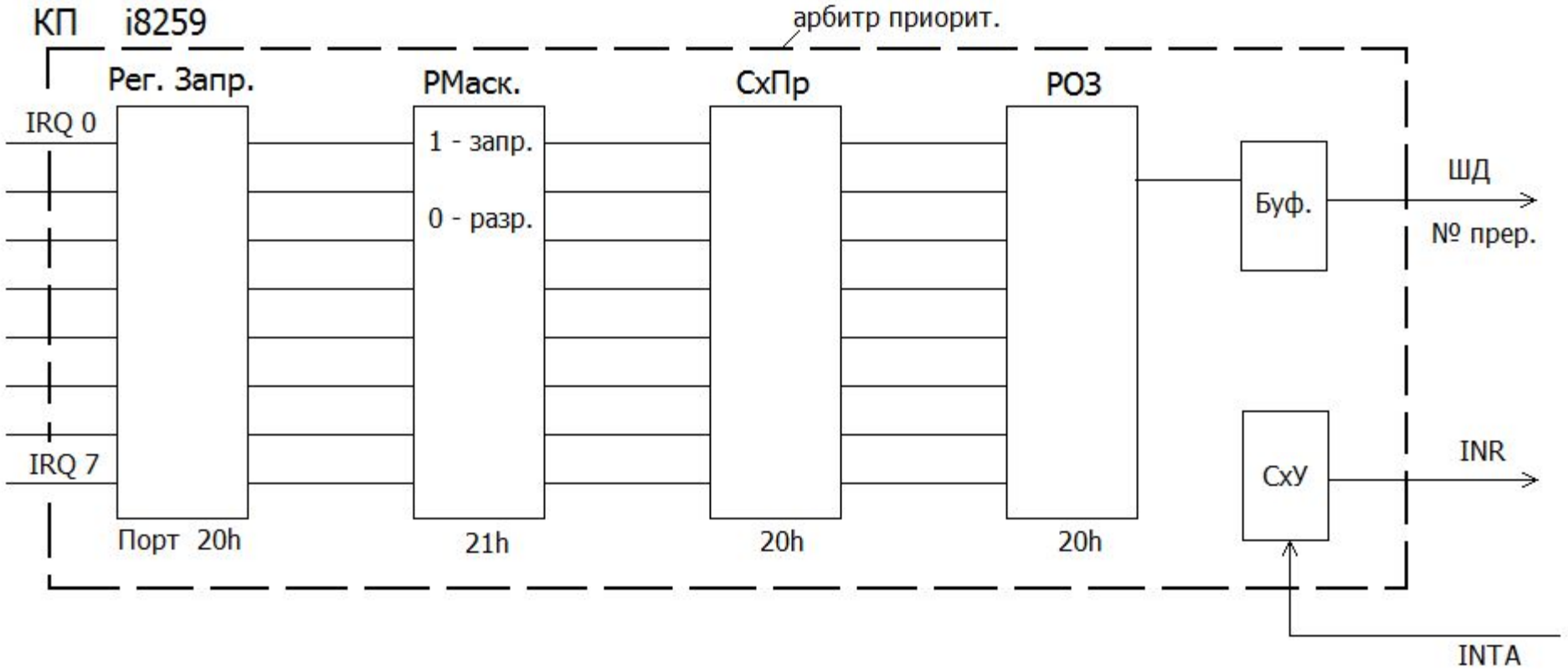
- Флаги IF и TF обнуляются, запрещая маскируемые прерывания и прерывание пошаговой обработки;
- Содержимое регистров FLAGS, CS и IP сохраняются в текущем стеке;
- Из вектора, соответствующего номеру (типу) прерывания, загружаются новые значения IP и CS. Номер прерывания, умноженный на 4, дает абсолютный адрес первого байта вектора обрабатываемого прерывания.

Рис. 1 Процедура прерывания для номера (типа) N




Когда обработчик прерывания получает управление, он может снова разрешить маскируемые прерывания (STI), поскольку стековая организация позволяет вложение прерываний друг в друга. По окончании обработчик должен воостановить старые значения IP, CS и FLAGS (команда IRET). Процессор продолжит работу с того места программы, где произошло прерывание.


A1h
21h
8
9
10
11
12
13
14
15
КП
i8259A
Ведомый
0
1
2
3
4
5
6
7
КП
i8259A
Ведущий
NMI
INR
ШД
8 бит
Процессор
INTA


КП i8259
арбитр приорит.
Рег. Запр.
РМаск.
СхПр
РОЗ
IRQ 0
IRQ 7
1 - запр.
0 - разр.
Буф.
ШД
№ прер.
СхУ
INR
INTA
Порт 20h
21h
20h
20h

Кроме внешних прерываний имеется ещё один вид – внутренние прерывания процессора.

Они возникают в ситуациях, требующих специального обслуживания, например, прерывание при делении на 0 (номер (тип) прерывания 0) или прерывание по флагу TF (пошаговое выполнение программы – тип -1), или при выполнении команды INT (прерывание)- тип 3, или INTO (прерывание при переполнении)- номер 4 и т. п. Эти прерывания называются программными. Программист может пользоваться следующими командами процессора для организации программной работы с прерываниями:

- INT номер ; Прерывание: SP □ SP – 2; FLAGS □ [SS:SP];
  SP □ SP – 2; CS □ [SS:SP];
  SP □ SP – 2; IP □ [SS:SP];
  [( номер * 4 )] □ IP ; [(номер * 4 +2)] □ CS

INT ; однобайтовая команда, тип – 3 ( INT 3h ), используется при отладке программ ( ввод точек прерывания)

INTO : прерывание при переполнении, если OF = 1: [ 10h ] □ IP, [ 12h ] □ CS .

IRET : Возврат из обработчика прерывания: [ SS:SP ] □ IP ; SP □ SP + 2:
[ SS:SP ] □ CS ; SP □ SP + 2:
[ SS:SP ] □ FLAGS ; SP □ SP + 2:

# Обработчики прерываний

Структура обработчика прерывания и его взаимодействие с остальными компонентами программного комплекса определяются рядом факторов, наиболее важные следующие:

- Прерывания, инициализирующие обработчик, могут быть аппаратными (от внешних устройств) или программными ( int N );
- Обработчик может входить в состав прикладной программы или быть резидентным;
- Вектор обрабатываемого прерывания может быть свободным или использоваться системой;
- Если вектор уже используется системой ( т. е. в составе ОС или BIOS есть системный обработчик прерываний с соответствующим номером), то новый обработчик может полностью заменить системный или «сцепляться» с ним;
- В случае сцепления с системным, новый может выполнять свои функции до системного или после него;

При вызове обработчика прерываний содержимое регистров SS, DS, ES, GS, FS не изменяется. Данные обработчика находятся в сегменте кода ( CS ), и при обращении к ним надо использовать явное приведение сегмента, например: mov bx, CS: ar1. Обработчик должен заканчиваться командой IRET, которая восстанавливает IP, CS, FLAGS.

Структура простейшего обработчика прерывания:

```
INT_NEW     PROC
            …….
                ……..
                IRET
OLD_INT      DD  ?
INT_NEW      ENDP
```

# 4 Структуры простейшего обработчика прерывания

1.
```
I_N1   proc        ;   Новый обработчик выполняется после старого
       pushf
       call  CS: OLD_INT
       …..                    ; обработчик пользователя
       iret
I_N1   endp

I_N2   proc
       …..     ; обработчик пользователя
       jmp  CS: OLD_INT
I_N2   endp

I_N3   proc
       …..     :  новый до системного
       pushf
       call  CS: OLD_INT
       …..     ;  новый после системного
       iret
I_N3   endp
```

4 вариант предполагает при вызове нового обработчика предварительный анализ некоторых условий на основе которого управление передается либо системному обработчику, либо новому:

```
I_N4    PROC
        ……..        ;   анализ условий
        …….
        Jcond  M1     ;  переход при выполнении условий к новому обработчику
         JMP    CS: OLD_INT        : переход к системному обработчику
M1:     …….         ;  новый обработчик
         …….
        IRET
I_N4    ENDP
```

# Функции ОС для работы с векторами прерываний

**35h → AH ; Получить вектор прерывания**

**AL → номер вектора прерывания**

**Возвращает: в ES:BX текущее значение вектора прерывания.**

**25h → AH ; Изменить вектор прерывания**

**DS:DX → новое значение вектора прерывания**

**AL → номер прерывания**

**Пример:**

```
mov  ax,  25NNh     ;  NN – номер прерывания
lea   dx,  new_NN    ;  эффективный адрес нового обработчика прерывания NN
push  ds
push  cs
pop   ds
int   21h
pop   ds
```

**Нахождение минимального и максимального значения в массиве слов. ВХОД: DS:BX – адрес начала массива, CX-количество элементов массива. Выход – AX-максимальный, BX –минимальный эл-т.**

```
Minmax  proc  near
push 0
pop  es
mov eax, dword prt es:[5*4]
mov dword ptr old_int5, eax
mov word prt es: [5*4], offset int5_new
mov word prt es: [5*4]+2, cs
mov ax, word prt [bx]
mov  word prt min_bound, ax
mov  word prt max_bound, ax
mov di, 2
M1: mov  ax, word prt [bx][di]
bound  ax, bounds
add  di, 2
loop  M1
mov  eax, dword ptr old_int5
mov dword ptr es:[5*4], eax
mov ax,  word ptr max_bound
mov bx,  word ptr min_bound
ret
bounds:
min_bound  dw  ?
max_bound  dw  ?
old_int5      dd   ?
```

```
int5_new  proc  far

cmp  ax,  word ptr  min_bound
jl M2  ; если не меньше – это нарушение
; верхней границы
mov word ptr  max_bound, ax
iret
M2:
mov word ptr  min_bound, ax
iret
int5_new   endp
Minmax  endp
```

## Пример обычного обработчика прерывания ( команда BOUND – INT 5 )

**BOUND reg, mem ; проверка нахождения индекса вектора, заданного в регистре (16, 32), ; внутри диапазона, заданного значениями двух последовательных ; слов (двойных слов) в памяти по адресу второго операнда. Эти значения являются , соответственно, нижней и верхней границей индекса массива. Они должны быть помещены предварительно в память. Н: mem dd 00000014h , где 0000 – нижняя граница и 0014h – верхняя граница ( допустимые) проверяемого индекса. Если значение индекса в регистре находится в диапазоне, то выполняется следующая команда после BOUND, иначе генерируется прерывание 5 ( int 5).**

**Пример: Нахождение минимального и максимального значения элементов в массиве слов, адрес которого задан в DS:DX, количество элементов в CX. Результат – в AX максимальный элемент, в BX – минимальный элемент.**

```
minmax  proc  near
    push  0                                  ; mov  ax,  3505h
    pop   es                                 ; int  21h
    mov  eax,  dword  ptr  es: [5*4]         ;  mov  word ptr old_int5,  bx
    mov dword  ptr old_int5,  eax            ;  mov  word  ptr old_int5 +2,  es
    mov  word  ptr es: [5*4],  offset new_5  ; mov  ax,  2505h
    mov  word  ptr es: [5*4] + 2,  cs        ; mov  dx,  offset  new_5
                          ; push  ds   push  cx   pop  ds  для ком.  не нужны
    mov  ax,   [bx]                          ; int  21h
    mov  word  ptr min,  ax                  ; pop  ds    - для ком.  не нужна
    mov  word  ptr max,  ax
    mov   di,  2
```

```
  ; обработка массива
M1:  mov ax,  word  ptr [bx][di]
  bound  ax,  bounds
  add  di,  2
  loop  M1

; восстановить старый обработчик  (*)
  mov  eax, dword  ptr old_int5
  mov  dword  ptr es: [5*4],  ax
;  вернуть результат
  mov  ax,  max
  mov  bx,  min
  ret
bounds:
min   dw   ?
max  dw  ?
old_int5  dd  ?

;(*)  для второго варианта:

  mov  ax,   2505h
  push  ds            ; для ком.  не нужны
  lds dx,  old_int5
  int  21h
  pop  ds             ; для ком.  не нужны
```

Новый обработчик прерывания 5

```
new_5  proc  far
  cmp  ax, word  ptr min
  jl    its_lower

  mov  word  ptr max,  ax
  iret
Its_lower:  mov  word  ptr min,  ax
  iret
new_5   endp
minmax  endp
```

## Функции работы с системными временем и датой

**2Ah □ AH ; Получение системной даты**

**Функция возвращает в регистрах:**

**AL – день недели ( 0 – воскресенье, …, 6 – суббота ); CX - год ( 1980 – 2099 );**

**DH - месяц ( 1 – 12 ); DL - число (1 – 31 ).**

**2Bh □ AH ; Изменение даты**

**Входные данные:**

**CX □ год ( до 2099 ); DH □ месяц ( 1 – 12 ):**

**DL □ число ( 1 – 31 ). Функция возвращает в AL – 00h, если дата действительная, FFh, если дата недействительная.**

**2Ch □ AH ; Получение системного времени**

**Функция возвращает в регистрах:**

**CH - час ( 0 – 23 ); CL - минуты ( 0 – 59 ); DH - секунды ( 0 – 59 ); DL – сотые доли сек.**

**2Dh □ AH ; Изменение системного времени**

**Входные данные:**

**CH - час ( 0 – 23 ); CL - минуты ( 0 – 59 ); DH - секунды ( 0 – 59 ); DL – сотые доли сек.**

# Резидентные программы

**С помощью 31 функции 21h прерывания программа завершается, но не выгружается из памяти, т.е. становится резидентной. Резидентная программа должна:**

1. **Сохранять вектор того прерывания, которое она замещает**
2. **Предусматривать блокировку повторной загрузки**
3. **Иметь средства выгрузки из памяти**

**При выполнении действий обработки в обработчике прерываний нельзя использовать функции ОС. Вместо функций ОС используются функции BIOS.**