МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

кафедра САПР



**ЗВІТ**

до лабораторної роботи №6

ОБРОБКА ПЕРЕРИВАНЬ

Виконав:

студент групи КНз-21

Чалий М.В.

Перевірив

Мазур В.В.

Львів-2014

### МЕТА РОБОТИ

Мета роботи - вивчення основ організації та реалізації апаратурних та програмних переривань.

### КОРОТКІ ТЕОРЕТИЧНІ ВІДОМОСТІ

В процесі виконання деякої програми іноді виникає необхідність негайної обробки деякої ситуації, яка ідентифікується апаратурними сигналами чи програмно (наприклад, натискання клавіші клавіатури, отримання сигналу від таймера, програмне переривання (int 21h). Відповідно до причини, переривання бувають програмні та апаратурні. Особливістю апаратурних переривань є відсутність часової координації з програмою, що виконується. Крім того, апаратурні переривання в першу чергу потребують програмної обробки.

В процесі виникнення і обробки переривання процесор призупиняє виконання програми, запам’ятовує місце призупинки, переходить на програму обробки відповідного переривання, а по її завершенню відновлює виконання перерваної програми з запам’ятованого місця. Такою є схема обробки переривання. Точка переривання (адреса CS:IP) запам’ятовується в стеці разом з регістром прапорців. Потім в CS:IP завантажується адреса точки входу програми обробки переривання і їй передається управління. Програми обробки переривань іноді називають драйверами переривань. Програма обробки переривань закінчується оператором IRET (повернення із переривання), яка завершує процес обробки переривання, відновлюючи значення CS:IP і регістра прапорців точки переривань, продовжуючи виконання перерваної програми.

В процесі виконання програми обробки переривань можливі нові переривання, тобто переривання і відповідні програми обробки можуть бути вкладеними. Запам’ятовування CS:IP та RF забезпечує безконфліктність.

Строго кажучи, програмні переривання не є перериваннями,бо вони здійснюються із звичайних програм. Тобто в даному випадку механізм переривань використовується для виклику певних процедур. Однак на відміну від звичайних процедур та підпрограм програми переривання містяться в операційній системі і призначені для виконання певних системних функцій (ввід-вивід символа, ввід командної стрічки, байта, тощо).

Адреси програм обробки переривань називають векторами. Кожен вектор має довжину 4 байти. В першому слові зберігається IP, у другому – CS. Таким чином у виділених 1024 байтах оперативної пам’яті зберігається 256 векторів Ця область називається таблицею векторів. (Вектор переривання 0 починається з 0000:0000, переривання 1 – 0000:0004, 2 – 0000:0008 і т.д.).

Для управління апаратними перериваннями використовується мікросхема програмного контроллера переривань (керуюча мікроЕОМ) Intel 8259 (або її еквівалент). Оскільки в будь-який момент часу може відбутися не один, а кілька запитів на переривання, то ця схема має 8 рівнів пріоритетів (IRQ 0 – IRQ 7 (IRQ 15)). IRQN – запит на переривання. Максимальний пріоритет відповідає рівню 0. Для машин АТ додаткові 8 рівнів обробляються другою мікросхемою 8259. Цей другий набір рівнів має пріоритет між IRQ 2 та IRQ 3.

Переривання 0-7 відповідають векторам від 8Н до OFH. Апаратні переривання впорядковані по зменшенню пріоритету: IRQ 0 – таймер, 1 – клавіатура, 2 – канали в/в, (8 годинник реального часу (АТ), 9 – програмно переводиться в IRQ 2 (AT), 10 -резерв, 12 - резерв, 13 – математичний сопроцесор (АТ), 14 – контроллер жорсткого диску (АТ), 15 – резерв, 3 – COM1 (СОМ2 (АТ)), 4 – СОМ2 (СОМ1 (АТ)), жорсткий диск (LPT2 (AT)), 6 – контроллер дискет, 7 – LPT1.

Максимальний пріоритет таймера встановлюється з метою забезпечення правильності системного годинника (відсутність втрат часу за рахунок очікування). Переривання від клавіатури ініціюється натисненням/відпусканням клавіші і його обробка полягає в занесенні коду клавіші в буфер (цей код може бути вибраний за допомогою відповідного програмного переривання).

В програмах на Асемблері апаратні переривання можуть бути замасковані (тимчасово заборонена їх обробка). Деякі апаратні переривання (наприклад, ділення на 0) не можуть бути замасковані. Є дві причини блокування переривань:

1. Блокування на час критичної частини коду програми обробки - поки не буде замінений вектор переривань цілком, а не частково).
2. Маскування переривань при виконанні критичних до часу операцій (наприклад, робота з диском, швидкий ввід/вивід)

Якщо біт 9 регістру прапорців рівний 0, то дозволені всі переривання, які дозволяє маска, при RF9 = 1 - всі апаратні переривання заборонені. Заборона переривань здійснюється командою CLI, дозвіл – STI. Заборона переривань на довгий період може порушити роботу ОС (зокрема, порушення роботи системного годинника - пропуск синхросигналів).

При виклику програми обробки програмних переривань ПЕОМ автоматично забороняє апаратні переривання до їх закінчення. Якщо у програмах обробки програмних переривань допускаються апаратурні, то їх можна дозволити командою STI. Якщо після CLI не вказати пізніше STI, то машина блокується (недоступна клавіатура).

### Програма для обробки переривань

.286

.model tiny

.stack 256

.code

org 100h

start:

jmp main

vold dd ?

error\_msg db "errr occured$"

handl proc far

push cs

pop ds

pusha

; Turn on beep

mov ah,6

mov dl,7

int 21h

;відключаю сигнал переривань годинника реального часу

mov ah,07h

int 1ah

; Revert to old handler

lds dx,vold

mov ah,25h

mov al,4ah

int 21h

popa

; Handle priorities

mov al,20h

out 0ah,al

out 20h,al

iret

fin equ $

handl endp

main:

push cs

pop ds

pusha

; Drop previous alarams

mov ah,07h

int 1ah

; Set new time

mov ch,00100011b

mov cl,01000101b

mov dh,00110101b

mov ah,06h

int 1ah

jc err

; Save old handle

mov ah,35h

mov al,4ah

int 21h

mov word ptr vold,bx

mov word ptr vold+2,es

push ds

; Set new handle

mov dx,offset handl

mov al,4ah

mov ah,25h

int 21h

; Return back

pop ds

popa

lea dx,fin

int 27h

err:

mov dx,offset error\_msg

mov ah,9

int 21h

ret

end start

### Лістінг програми обробки преривань

JWasm v2.11, Oct 20 2013

.\L6RTC.ASM

.286

.model tiny

.stack 256

.code

org 100h

00000100 start:

00000100 EB30 jmp main

00000102 00000000 vold dd ?

00000106 65727272206F636375 error\_msg db "errr occured$"

00000113 handl proc far

00000113 0E push cs

00000114 1F pop ds

00000115 60 pusha

; Turn on beep

00000116 B406 mov ah,6

00000118 B207 mov dl,7

0000011A CD21 int 21h

;відключаю сигнал переривань годинника реального часу

0000011C B407 mov ah,07h

0000011E CD1A int 1ah

; Revert to old handler

00000120 C5160000 lds dx,vold

00000124 B425 mov ah,25h

00000126 B04A mov al,4ah

00000128 CD21 int 21h

0000012A 61 popa

; Handle priorities

0000012B B020 mov al,20h

0000012D E60A out 0ah,al

0000012F E620 out 20h,al

00000131 iret

00000132 = 132 C fin equ $

00000132 handl endp

00000132 main:

00000132 0E push cs

00000133 1F pop ds

00000134 60 pusha

; Drop previous alarams

00000135 B407 mov ah,07h

00000137 CD1A int 1ah

; Set new time

00000139 B523 mov ch,00100011b

0000013B B145 mov cl,01000101b

0000013D B635 mov dh,00110101b

0000013F B406 mov ah,06h

00000141 CD1A int 1ah

00000143 7220 jc err

; Save old handle

00000145 B435 mov ah,35h

00000147 B04A mov al,4ah

00000149 CD21 int 21h

0000014B 891E0000 mov word ptr vold,bx

0000014F 8C060200 mov word ptr vold+2,es

00000153 1E push ds

; Set new handle

00000154 BA0000 mov dx,offset handl

00000157 B04A mov al,4ah

00000159 B425 mov ah,25h

0000015B CD21 int 21h

; Return back

0000015D 1F pop ds

0000015E 61 popa

0000015F 8D160000 lea dx,fin

00000163 CD27 int 27h

00000165 err:

00000165 BA0000 mov dx,offset error\_msg

00000168 B409 mov ah,9

0000016A CD21 int 21h

0000016C C3 ret

end start

Binary Map:

Segment Pos(file) RVA Size(fil) Size(mem)

---------------------------------------------------------------

<header> 0 0 20 0

\_TEXT 20 100 6D 6D

\_DATA 8E 16E 0 0

STACK 90 170 0 100

---------------------------------------------------------------

8D 170

Macros:

N a m e Type

@CatStr . . . . . . . . . . . . Func

@Environ . . . . . . . . . . . . Func

@InStr . . . . . . . . . . . . . Func

@SizeStr . . . . . . . . . . . . Func

@SubStr . . . . . . . . . . . . Func

Segments and Groups:

N a m e Size Length Align Combine Class

DGROUP . . . . . . . . . . . . . GROUP

\_TEXT . . . . . . . . . . . . . 16 Bit 016D Word Public 'CODE'

\_DATA . . . . . . . . . . . . . 16 Bit 0000 Word Public 'DATA'

STACK . . . . . . . . . . . . . 16 Bit 0100 Para Stack 'STACK'

Procedures, parameters and locals:

N a m e Type Value Segment Length

handl . . . . . . . . . . . . . P Far 0113 \_TEXT 001F Public

Symbols:

N a m e Type Value Attr

@CodeSize . . . . . . . . . . . Number 0h

@DataSize . . . . . . . . . . . Number 0h

@Interface . . . . . . . . . . . Number 0h

@Model . . . . . . . . . . . . . Number 1h

@code . . . . . . . . . . . . . Text \_TEXT

@data . . . . . . . . . . . . . Text DGROUP

@stack . . . . . . . . . . . . . Text DGROUP

err . . . . . . . . . . . . . . L Near 165h \_TEXT

error\_msg . . . . . . . . . . . Byte[13] 106h \_TEXT

fin . . . . . . . . . . . . . . L Near 132h \_TEXT

main . . . . . . . . . . . . . . L Near 132h \_TEXT

start . . . . . . . . . . . . . L Near 100h \_TEXT Public

vold . . . . . . . . . . . . . . DWord 102h \_TEXT

.\L6RTC.ASM: 83 lines, 2 passes, 0 ms, 0 warnings, 0 errors