Міністерство освіти та науки України

# Кіровоградський Державний Технічний університет

Кафедра програмного забезпечення

Курсова робота

з дисципліни “Програмування на мові ASM-86” на тему:

Програма TXTPRINT.COM - резидентна програма для швидкого і зручного друкування виборчого тексту з екрану

**Зміст**

Вступ.

1. Призначення та область використання системи.

2. Огляд iснуючих систем. Обґрунтування вибору принципу розробки та методики побудови системи. Постановка задачі по реалізації технічного завдання

3. Опис та обґрунтування проектних рішень по проектуванню системи.

4. Разрахунки та експерементальні матеріали які підтверджують вірність програмних рішень

5. Основнi висновки

Перелiк скорочень, символiв та спецiальних термiнiв.

Список лiтератури.

Анотацiя.

Додатки.

Текст програми.

**Вступ**

Програма txtprint.com - резидентна програма, яка предназначена для того, щоб швидко і зручно надрукувати виборчий текст з екрану. Це може бути корисним, коли доводиться аналізувати в водночас багато текстової інформаціі, наприклад різні файли-помічники, текст іншої програми, та багато ін. Також ця програма може стати у пригоді тоді, коли треба надрукувати текстовий файл, але не повністю, окремими епізодами. Програма дуже зручна для її використання, займає усього 2 килобайти DOS - пам`яті, також її можна вилучити з пам`яті в будь-який час. Просто і зрозуміло оформлен інтерфейс з користувачем, отже навчитись роботі з програмою TXTPRINT.COM можна за хвилину.

Також в програмі встановлено тест на подвійну загрузку, отже при спробі завантажити її вдруге користувач отримає повідомлення про це, а також програма нагадае про «гарячі» клавіші, таким чином запобігаючи подвійного завантаження. Програма не конфліктує з іншими резидентними програмами (якщо, звісно, вони працюють коректно), але в разі змінення ними хочаб одного з векторів преривань INT 08H, INT 09H, INT 28H, INT 2FH програма TXTPRINT.COM не в змозі буде вилучитись з пам`яті по вимозі користувача. Про це програма видасть повідомлення, і нагадає про причину цього (не на вершині). Але після вилучення з пам`яті тієї програми, що змінила вище перечислених векторів преривань, при подальшої вимоги користувача програму TXTPRINT.COM теж буде вилучено з пам`яті.

**1. Призначення та область використання системи**

Програма TXTPRINT.COM призначена для друку на принтері вибіркового тексту з екрану. Також її можна використовувати в учбових закладах для полегчення і прискорення засвоєння потрібної документації з комп`ютера (різні HELP-програми, завдання задач та ін.). Для цього користувачу потрібно перед переглядом документації на комп`ютері завантажити програму TXTPRINT.COM і в потрібний час викликати її за допомогою «гарячих» клавиш. Потім помітити потрібну частину тексту і дати команду «печатати». Після чого цей текст розпечатається на принтері, і буде більш зручнішим для прочитання та багаторазового перегляду під час виконання роботи (лабораторної тощо). Програма має дуже простий і зручний інтерфейс, що дуже полегшує роботу з нею, займає усього 2 кілобайти DOS-пам`яті.

Також ця програма дуже цікава з точки зору вивчення резидентних програм. На її прикладі можна обучати студентів по роботі з функціями DOS в резидентних програмах.

**2. Огляд існуючих систем, обґрунтування вибору принципу розробки та методики побудови системи. Постановка задачі по реалізації технічного завдання**

Існує багато систем, у яких можна зробити вибіркову розпечатку тексту (усі програми-редактори, та багато ін.), але їх недолік у тому, що вони не резидентні, і можуть розпечатувати тільки текст, редагуванням якого вони займаються.

Програма TXTPRINT.COM повинна бути резидентною, працювати швидко та надійно при різних процесорах, а також займати якомога менше пам`яті. Тому задача розроблялась на мові асемблера (ASM-86) для TASM версії не нижче 2.0.

Програма повинна:

* бути резидентною і не конфліктувати з операційною системою та іншими програмами;
* викликатись по натисненню «гарячих» клавиш; мати дружній інтерфейс з користувачем;
* розпечатувати помічений текст з екрану;
* не завантажуватись в друге;
* займати якомога менше пам`яті;
* по вимозі користувача вилучатись з пам`яті.

По обґрунтуванню та реалізації цього завдання см. розділ 3.

**3. Опис та обґрунтування проектних рішень по проектуванню системи**

Усе, що приводиться у цьому параграфi вiдноситься тiльки до "випригуючих" (popup) резидентних програм.

Реентабельною (повторно входимою) програмою називаеться програма чи процедура, одна копiя якої в пам`ятi може бути одночасно викликатись кiлькома процесами, при чьому рiзноманiтнi виповнювання цiєї процедури не впливають одна на одну.

Нереентабильність MS-DOS проявляеться в неможливiстi звертання к функцiям DOS, якщо вже виповнюеться яка-небудь iнша функцiя DOS. Порушення цiєї вимоги призведе до зруйнування системного стеку i краху операцiйної системи. Саме з нереентабильнiстю DOS пов`язанi основнi труднощi розработки popup - програм. Справа в тому, що цi резидентнi програми використовують для свого запуску аппаратнi преривания, якi можуть статися в будь-який час, у тому числi й тодi, коли виповнюється функцiя DOS.

Найпростiше рiшення цiєї проблеми складається у повнiй вiдмовi вiд використання системного сервiсу DOS в резидентних програмах. Iнколи це не дуже складно. Так, для виводу на дiсплей можна користуватись прериванням BIOS 10H чи прямим записом у вiдеобуфер, для вводу символiв з клавiатури приємлимо користуватись прериванням 16H, при роботi з принтером i послiдовними портами також неважко працювати на рiвнi BIOS. Але у разi, коли потрiбен дисковий ввод - вивод, це вже стає дуже проблематичним.

Працювати з дисками средствами BIOS - дiйсно дуже хлопотна справа.

Iснують i iншi ситуацiї, коли зручнiше було б використовувати функцiї DOS. Отже, треба навчитися перемагати нереентабiльнiсть DOS. Для цього iснує багато можливостей.

Почнем з того, що існує простий (хоча й недокументований) спосiб запобiгнути конфлiктiв з операційною системою. При активiзацiї програма може перевiрити байтовий прапорець активнiстi DOS i, якщо вiн вiдмiнний вiд нуля (сервiс DOS недоступний), повернутись до пасивного стану. Адресу цього прапорця повiдомляє функцiя DOS 34H.

Функція 34H. Дає адресу прапорця активністі DOS

Виклик: AH = 34H

Повертає: ES:BX - адреса прапорця

Нажаль, якщо вибрати цей путь, така резидентна программа, скоріше за все, взагалі буде не взмозі стартувати. Це пов`язано з тим, що COMMAND.COM та багато інших програм проводять багато часу в очикуванні ввода строки з клавіатури посредством функції DOS 0AH, і, отже, прапорець активністі DOS майже завжди взведений. Ця обставина використовується у диспетчері функцій DOS, який може мати вигляд на зразок цього:

Int\_21h PROC far

sti

pushf

cmp ah,0Ah ; якщо номер функції не рівний 0ah,

jnz Exec ; то приступити к виконанню.

Fn0Ah:

push ax ; якщо функція 0ah, то затримати

mov ah,0bh ; її виповнення до тіх пір, поки у

int 21h ; буфері клавіатури не з`явиться

cmp al,0 ; хочаб один символ.

pop ax

jnz Exec

push cx

mov cx,4000h

Delay: loop Delay ; зробити паузу

pop cx

jmp short Fn0Ah ; і повторити перевірка буфера.

Exec:

or byte ptr cs:DOS\_busy,01h ; встановити признак

; зайнятості DOS.

Popf ; визвати початковий обробник.

pushf

call dword ptr cs:[Int\_21h\_vect]

pushf

and byte ptr cs:DOS\_busy,0feh ; скинути признак

; зайнятості.

popf

ret 2

Int\_21h\_vect dd ?

Dos\_busy db 0

Int\_21h ENDP

Запит на виконання функції 0AH не передається одразу до операційної системи. Замість цього периодично перевіряється стан буфера вводу (функція DOS 0BH або функція 01H преривання 16H). І вже тільки після того як буфер почав заповниватися, визивається функція 0AH. Оберніть увагу на те, як оформлен возврат у програму - це викликано тим, що DOS використовує прапорець CF в якості признаку помилки.

Важливо те, що в описаному прикладі можна відмовитись від перевірки недокументованого прапорця активністі DOS. Замість того диспетчер встановлює свій власний прапорець, який дотого ж бітовий, отже він може бути об`єднан з іншими прапорцями резидентної програми.

Диспетчер функцій DOS - приємлимий, але не найбільш вдалий спосіб перевагання нереєнтабільності DOS. Програмам, використовуючим його, далеко не завжди вдається стартувати. Крім того, їм приходиться аналогічним чином відслежувати ще й преривання 25H (читати сектор), 26H (писати сектор) DOS, а також преривання 13H BIOS.

Обробники преривань 25H і 26H виявляються звичайними функціями - під час повернення управління вони залишають в стеці регистр прапорців. А це класичний приклад помилки, якій в наслідок увішов у документацію.

Найбільш цікавими можливостями преодоління нереєнтабільністі DOS пов`язані з використовуванням недокументованого преривання 28H. Це преривання використовується резидентними процесами самої операційної системи, наприклад, PRINT, та генерується як раз під час виконання функцій DOS, коли прапорець активністі DOS взведений, але система знаходиться у безпечному стані.

За допомогою преривання 28H можна досягнути потрясаючих результатів. Отже, така програма може стартувати, наприклад, під час копіювання файлів, а іноді при форматуванні диску.

При натисканні на «гарячу» клавишу не відбувається негайна активізація програми, а лише встановлюється битовий прапорець, повідомляючий, що поступила команда стартувати. Обробник преривання 28H перевіряє стан прапорця, і, якщо прапорець встановлен, а програма ще не активна, активізує її. На прикінці роботи управління завжди передається завжди повинне передаватись початковому обробнику. Програми, які виконуються усередині преривання 28H, не повинні користуватись функціями DOS з номерами 00H - 0CH.

А як буть, коли довгий час не відбувається звертання до функцій DOS і, отже, не вироблялось преривання 28H? ПроРграма знов не в змозі стартувать. Щоб запобігнути цього, буде потрібна ще одна точка входу, в якісті якої зручно буде обрати обробник якогось часто возникаючого преривання, наприклад, від таймеру. Тоді перед активізацією програми треба перевіряти прапорець активністі DOS, чи свій власний прапорець.

Реакція на виключні випадки.

Іноді в результаті збоїв апаратної частини або неправильних дій користувача може скластися ситуація, коли подальша робота програми виявляється неможливою. В цьому випадку управління дістає спеціальна процедура обробки критичної помилки, адреса якої знаходиться у векторі преривання 24H. Користувач може також сам прервати програму, натиснувши комбинацію клавиш Ctrl+Break. При цьому MS-DOS передає управління до процедури завершення, яка починається по адресі обробника преривання 23H.

Цей сервіс DOS, неодмінно, дуже корисний, але орієнтован на звичайні програми і не може правильно працювати в резидентних програмах. Тому вам слід передбачити власну оброботку виключних випадків або відключити існуючу.

Почнемо з преривання Ctrl+Break. Реакція системи на це преривання залежить від стану прапорця перевірки по Ctrl+Break. Якщо прапорець скинутий, то перевірка натиснення Ctrl+Break і завершення програми відбувається тільки при виконанні функції DOS з номерами 00H - 0CH, а якщо прапорець встановлен, при виконанні будь-яких функцій. Тому, якщо в програмі не використовуються функції DOS 00H-0CH, тоді достатньо скинути прапорець перевірки по Ctrl+Break при активізації резидентної програми і встановити його початкове значення перед поверненням у пасивний стан.

Функція 33H. Прапорець перевірки по Ctrl+Break

Виклик: AH = 33H

AL = 0 - отримати стан прапорця

1 - змінити стан прапорця.

DL (якщо AL=1) = 0 - скинути прапорець

1 - встановити прапорець

Повертає: DL ( якщо AL дорівнював 0) -

стан прапорця:

0 - скинут

1 - встановлен

Для перевірки або зміни стану прапорця зручно застосовувати функцію DOS 33H.

Другий можливий путь - просто відмінити оброботку преривання 23H, тобто змінити його вектор так, щоб він вказував на інструкцію IRET.

При виникнинні критичної помилки (звично це помилка диску) MS-DOS завантажує у регістри AX, DI, SI, BP інформацію про помилку ініцирує преривання 24H. Стандартний системний обробоник преривання веде себе слідуючим чином: видає на дісплей повідомлення про помилку і запит користувача - «Abort, Retry, File or Ignore?» («Прервати, Повторити, Зняти або Ігнорувати»), чекає відповіді і повертає управління функції DOS, при виконанні якої виникла помилка. Подальші дії операційної системи визначається змістом регістру AL(відповідь користувача).

**4. Розрахунки та експеріментальні матеріали, що підтверджують вірність програмних рішень**

Насправді, програма не має в собі ніяких математичних розрахунків, тільки простіші. Складність написання програми полягала у технічній реалізації поставленої задачі. Тому підтвердити вірність програмних рішень можливо було тільки експеріментально. Були проведені тести на можливу недостачу пам`яті, роботу у версіях DOS 3.2 й вище, також під емуляцією DOS у системі WINDOWS`95. Також програма пройшла тести на правильність друку, загальну роботу з принтером. Результати тестів підтверджують правильність роботи програми. Тести проводилися з EPSON-сумісними принтерами.

**5. Основні висновки**

Програма TXTPRINT.COM - програма для вибіркового друку тексту з екрану. Для того, щоб получити готовий COM - файл з тексту програми, треба виконати: TASM TXTPRINT.ASM, потім TLINK TXTPRINT.OBJ /T. Після цього запустити програму TXTPRINT.COM. Під час просмотру інформації на екрані, яку треба розпечатати треба натиснути ScrollLock. Після цього, користуючись клавишами курсору і ENTER помітити необхідний фрагмент тексту. Потім для розпечатки натиснути клавишу BackSpace, щоб почати розпечатувати, або ESC в разі, якщо текст був помічений неправильно. Після цього програма завершує своє роботу до наступного виклику по ScrollLock. Коли роботу з розпечаткою закінчено взагалі, можна вилучити програму TXTPRINT.COM з пам`яті за допомогою комбинації клавиш Ctrl+Alt+Esc. Програма повідомить про закінчення своєї роботи, і для того, щоб розпочати роботу програми знов, треба заново завантажувати TXTPRINT.COM.

Програма в процесі роботи змінює вектори преривань 08H, 09H, 28H, 2FH, 23H і 24H. Тому якщо після завантаження TXTPRINT.COM були також завантажені інші програми, використовуючи хоча б одне з цих преривань, то цю програму неможна буде вилучити з пам`яті доти, доки не будуть вилучені ті програми, а отже востановлені вектори цих преривань.

**Перелiк скороченнь, символiв та спецiальних термiнiв**

i т.д. - i так далi

i т.i. - i таке iнше

та iн. - та iншi

т.я. - так як

п. - пункт

ОС - операцiйна система

popup - резидентні програми, які викликаються користувачем

DOS - дискова операцiйна система

BIOS - базова система вводу/виводу

INT - переривання

PC - персональний комп'ютер

WINDOWS - операцiйна система

**Список літератури**

1. Р. Джордейн «Справочник программиста персональных компьютеров типа IBM PC» М: Мир, 1991р.

2. П. Абель «Мова асемблера для IBM PC та програмування.» М.: Вища школа,1992.

3. Р. Лей та "Уейт-Груп" «Написання драйверів для MS-DOS.» М.: Мир, 1995р.

4. Електронний спавочник «TeachHelp»

5. Конспект лекцій по системному програмуванню

**Анотація**

Програма TXTPRINT.COM - програма для вибіркового друку тексту з екрану. Для того, щоб получити готовий COM - файл з тексту програми, треба виконати: TASM TXTPRINT.ASM, потім TLINK TXTPRINT.OBJ /T. Після цього запустити програму TXTPRINT.COM. Під час просмотру інформації на екрані, яку треба розпечатати треба натиснути ScrollLock. Після цього, користуючись клавишами курсору і ENTER помітити необхідний фрагмент тексту. Потім для розпечатки натиснути клавишу BackSpace, щоб почати розпечатувати, або ESC в разі, якщо текст був помічений неправильно. Після цього програма завершує своє роботу до наступного виклику по ScrollLock. Коли роботу з розпечаткою закінчено взагалі, можна вилучити програму TXTPRINT.COM з пам`яті за допомогою комбинації клавиш Ctrl+Alt+Esc. Програма повідомить про закінчення своєї роботи, і для того, щоб розпочати роботу програми знов, треба заново завантажувати TXTPRINT.COM.

Програма в процесі роботи змінює вектори преривань 08H, 09H, 28H, 2FH, 23H і 24H. Тому якщо після завантаження TXTPRINT.COM були також завантажені інші програми, використовуючи хоча б одне з цих преривань, то цю програму неможна буде вилучити з пам`яті доти, доки не будуть вилучені ті програми, а отже востановлені вектори цих преривань.

Текст програми має дев`ять модулів. Їх текст, краткий опис та пояснення приводяться у додатках.

**Додатки**

**Текст програми**

; Файл TXTPRINT.ASM - основна програма

; Для створення готового продукту необхідно

; виповнити слідуючі команди:

; TASM TXTPRINT.ASM

; TLINK /t TXTPRINT.OBJ

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

cr equ 13 ; символ "Возврат каретки".

lf equ 10 ; символ "Перевод строки".

start\_offset equ 0b8h ; сміщення початку резидентної

; порції у сегменті команд

; і вершина внутрішнього стеку.

internal\_flags record Sf:1,Rf:1,Af:1,Tf:1,Pf:1 ; набор прапорців

; 1=печать дозволена

; 1=Int\_08h активна

; 1=программа активна

; 1=команда "удалити"

; 1=команда "стартувати"

.model tiny

.code

org 100h

entry:

jmp boot ; перейти на процедуру загрузки резиденту.

shift = offset callvideo - start\_offset ; Величина

; сдвигу резидентної порції.

flags\_offset = offset resident\_sign - shift ; Сміщення прапорців

callvideo proc

; Обернутися до відео-BIOS.

push di

push si

push bp

int 10h

pop bp

pop si

pop di

ret

callvideo endp

sethandler proc

; Встановити обробник преривання.

push ax

mov ah,35h

int 21h

mov word ptr [di],bx

mov word ptr [di+2],es

pop ax

mov ah,25h

int 21h

ret

sethandler endp

resethandler proc

; Восстановити минулий обробник преривань.

push ds

mov dx,word ptr [di]

mov ds,word ptr [di+2]

mov ah,25h

int 21h

pop ds

ret

resethandler endp

include beep.asm

include typestr.asm

include message.asm

include remove.asm

include main.asm

include prepare.asm

include take&prn.asm

int\_28h proc far

test byte ptr cs:[flags\_offset],mask Sf or mask Rf

jz pass\_28h ; Якщо небуло команди, то вихід.

test byte ptr cs:[flags\_offset],mask Af or mask Tf

jnz pass\_28h ; Якщо активна, то вихід.

or byte ptr cs:[flags\_offset],mask Af

call prepare ; Запуск.

and byte ptr cs:[flags\_offset],not(mask Sf or mask Rf or mask Af)

pass\_28h: ; Передати управління ісходному обробнику.

jmp dword ptr cs:[int\_28h\_vect-shift]

int\_28h\_vect dd ?

int\_28h endp

int\_08h proc far

test byte ptr cs:[flags\_offset],mask Tf or mask Af; or mask Df

jnz pass\_08h ; Якщо активна, то вихід.

or byte ptr cs:[flags\_offset],mask Tf ; Встановити

; признак активнисті Int\_08h.

pushf ; Викликати ісходний

call dword ptr cs:[int\_08h\_vect-shift] ; обробник.

test byte ptr cs:[flags\_offset],mask Sf or mask Rf

jz end\_08h ; Якщо не було команды, то вихід.

push di ; Перевірити признак активности DOS.

push es

mov es,word ptr cs:[dos\_flag\_seg-shift]

mov di,word ptr cs:[dos\_flag\_off-shift]

test byte ptr es:[di],0ffh

pop es

pop di

jnz end\_08h ; Якщо DOS активна, то вихід.

or byte ptr cs:[flags\_offset],mask Af ; Встановити признак

; активности START.

call prepare ; Запуск.

and byte ptr cs:[flags\_offset],not(mask Sf or mask Rf or mask Af)

end\_08h:

and byte ptr cs:[flags\_offset],not mask Tf ; Сбросити

iret ; признак активністи Int\_08h.

pass\_08h: ; Передати управління ісходному

jmp dword ptr cs:[int\_08h\_vect-shift] ; обробнику.

int\_08h\_vect dd ?

int\_08h endp

int\_09h proc far

push ax ; Сохранити регістр AX.

in al,60h ; Ввести код натиснутої клавиши.

cmp al,70 ; клавиша ScrollLock.

jne remove\_test

mov ah,2 ; Перевірити Shift state.

int 16h

and al,0h

cmp al,0h

jne pass\_09h

or byte ptr cs:[flags\_offset],mask Sf

jmp short end\_09h

remove\_test:

cmp al,1 ; Клавиша Esc ?

jne pass\_09h

mov ah,2 ; Перевірити Shift state.

int 16h

and al,0ch

cmp al,0ch

jne pass\_09h

or byte ptr cs:[flags\_offset],mask Rf

end\_09h:

in al,61h ; Обробити апаратне преривання.

push ax

or al,80h

out 61h,al

pop ax

out 61h,al

mov al,20h

out 20h,al

pop ax

iret

pass\_09h:

pop ax ; Восстановити регістри і передати управління по старому

jmp dword ptr cs:[int\_09h\_vect-shift]; вектору преривання Int\_09h.

int\_09h\_vect dd ?

int\_09h endp

int\_2fh proc far

cmp ax,8900h ; Якщо запитуєтся установка процесу

jne pass\_2fh ; з номером 89Н, то запретити установку.

mov al,0ffh ; Інакше передати управління

iret ; ісходному обробнику.

pass\_2fh:

jmp dword ptr cs:[int\_2fh\_vect-shift]

int\_2fh\_vect dd ?

int\_2fh endp

exeptions proc far

int\_24h: mov al,3 ; Команда "Зняти системний визов".

int\_23h: iret ; Вийти з преривання.

int\_24h\_vect dd ?

int\_23h\_vect dd ?

exeptions endp

dos\_flag\_off dw ?

dos\_flag\_seg dw ?

resident\_sign internal\_flags <>

;----- Ця частина програми використовується для загрузки резидента

boot: ; ( сама не є резидентною ).

mov si,offset start\_string

call teletype

multiplex\_test:

mov ax,8900h ; Тест на подвійну

int 2fh ; установку.

cmp al,0

je install ; Якщо не встановлен, то встановить.

mov si,offset reboot\_message ; Інакше

call teletype ; видати повідомлення про подвійну

call teletype\_keys ; загрузку, "гарячих" клавишах

int 20h ; і завершитись.

install:

call teletype\_keys ; Видати повідомлення про "гарячі" клавиши

cld ; Сдвинути резидентну

mov di,start\_offset ; порцію в PSP.

mov si,offset callvideo

mov cx,boot-callvideo

rep movsb

mov cx,sethandler-shift

mov al,09h ; Встановити новий обробник

mov dx,int\_09h-shift ; преривання Int09h.

mov di,int\_09h\_vect-shift

call cx

mov al,2fh ; Встановити новий обробник

mov dx,int\_2fh-shift ; преривання Int2fh.

mov di,int\_2fh\_vect-shift

call cx

mov al,28h ; Встановити новий обробник

mov dx,int\_28h-shift ; преривання Int28h.

mov di,int\_28h\_vect-shift

call cx

mov al,08h ; Встановити новий обробник

mov dx,int\_08h-shift ; преривання Int08h.

mov di,int\_08h\_vect-shift

call cx

mov ah,34h ; Прочитати і сохранити адресу

int 21h ; прапорця активністи DOS.

mov word ptr ds:[dos\_flag\_off-shift],bx

mov word ptr ds:[dos\_flag\_seg-shift],es

mov es,word ptr ds:[002ch] ; Освободити

mov ah,49h ; окруженіе DOS.

int 21h

mov dx,boot-shift ; Завершитись, оставив

int 27h ; резидента.

include teletype.asm

teletype\_keys proc near

; Виводить повідомлення про гарячі клавиши.

mov si,offset hot\_keys\_mess1

call teletype

mov si,offset hot\_keys\_mess2

call teletype

mov si,offset hot\_keys\_mess3

call teletype

mov si,offset hot\_keys\_mess4

call teletype

mov si,offset hot\_keys\_mess5

call teletype

mov si,offset hot\_keys\_mess6

call teletype

mov si,offset hot\_keys\_mess7

call teletype

ret

teletype\_keys endp

start\_string db "Screen text printer. Version 1.0.",cr,lf,0

hot\_keys\_mess1 db "Use ",0

hot\_keys\_mess2 db "ScrollLock",40 dup (0)

hot\_keys\_mess3 db " to activate text printer and",cr,lf,0

hot\_keys\_mess4 db "Ctrl + Alt + Esc ",34 dup(0)

hot\_keys\_mess5 db " to remove it from memory.",cr,lf,lf,0

hot\_keys\_mess6 db "Use Enter to label text and",cr,lf,0

hot\_keys\_mess7 db "BackSpace to print it. Use Esc to cancel.",cr,lf,0

reboot\_message db cr,lf,"Text Taker has already"

db " been installed !",7,cr,lf,lf,0

io\_error\_mess db "I/O failure.",cr,lf,lf,lf,0

end\_entry:

END entry

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Кінець файлу TXTPRINT.ASM \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Файл PREPARE.ASM

; Модуль основної програми.

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

prepare proc

cli ; Ініціализувати стек.

mov word ptr cs:[stack\_offset-shift],sp

mov word ptr cs:[stack\_seg-shift],ss

push cs

pop ss

mov sp,start\_offset

sti

push ax ; Сохранити регістри.

push bx

push cx

push dx

push di

push si

push bp

push ds

push es

push cs

pop ds

mov al,23h ; Встановити новий обробник преривання Int23h.

mov dx,int\_23h-shift

mov di,int\_23h\_vect-shift

call sethandler

mov al,24h ; Встановити новий обробник прерывання Int24h.

mov dx,int\_24h-shift

mov di,int\_24h\_vect-shift

call sethandler

mov ah,15 ; Визначити поточний відеорежим і

call callvideo ; номер відеосторінки.

cmp al,2

je set\_param

cmp al,3

je set\_param

cmp al,7

jne end\_prepare ; Якщо графічний режим, то вихід.

set\_param: ; Сохранити номер відеосторінки.

mov byte ptr ds:[current\_video\_page-shift],bh

mov ah,03h ; Сохранити позицію курсору.

call callvideo

mov word ptr ds:[cursor\_loc-shift],dx

test byte ptr ds:[flags\_offset],mask Sf

jz begin\_remove

call start

jmp short restore\_cursor

begin\_remove:

call remove

restore\_cursor:

mov bh,byte ptr ds:[current\_video\_page-shift]

mov dx,word ptr ds:[cursor\_loc-shift] ; Восстановити

mov ah,2 ; положення курсору.

call callvideo

end\_prepare:

mov al,23h ; Восстановити вектор преривання Int 23h.

mov di,int\_23h\_vect-shift

call resethandler

mov al,24h ; Восстановити вектор преривання Int 24h.

mov di,int\_24h\_vect-shift

call resethandler

pop es ; Восстановити регістри.

pop ds

pop bp

pop si

pop di

pop dx

pop cx

pop bx

pop ax

cli ; Переключитись на стек прерваної програми.

mov ss,word ptr cs:[stack\_seg-shift]

mov sp,word ptr cs:[stack\_offset-shift]

ret

prepare endp

stack\_offset dw ?

stack\_seg dw ?

cursor\_loc dw ?

current\_video\_page db ?

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Кінець файлу PREPARE.ASM \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Файл MAIN.ASM

; Модуль основної програми

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

start proc

mov bp,sp ; Зарезервирувати місто

sub sp,13 ; в пам`яти для змінних.

mov byte ptr ds:[flag-shift],0 ; Обнулить всі прапорці.

mov byte ptr ds:[fl2-shift],0

mov byte ptr ds:[xx-shift],dl

mov byte ptr ds:[yy-shift],dh

rd:

call kurpos ; Встановити курсор.

call delay ; Визвать затримку.

in al,60h ; Взяти символ.

cmp al,14 ; BackSpace ?

jne nn ; Якщо ні - далі.

or byte ptr ds:[flags\_offset],mask Pf ; Іначе дозволити печать,

mov ah,01h ; ініціализувати 0 порт,

mov dx,0

int 17h

call unlabel ; розпечатать

jmp hi ; й вийти.

nn:

cmp al,28 ; ENTER ?

jne n0 ; Якщо ні - далі.

call here

n0:

cmp al,72 ; UP ?

jne n1 ; Якщо ні - далі.

cmp byte ptr ds:[yy-shift],0 ; Перевірити позицію курсора.

jne n00

mov byte ptr ds:[yy-shift],25

jmp n00

rd1:

call keybuf

jmp rd

n00:

dec byte ptr ds:[yy-shift] ; Уменьшить Y на 1.

call kurpos ; Встановити курсор.

n1:

cmp al,80 ; DOWN ?

jne n2 ; Якщо ні - далі.

cmp byte ptr ds:[yy-shift],25 ; Перевірити позицію курсору.

jne n11

mov byte ptr ds:[yy-shift],0

n11:

inc byte ptr ds:[yy-shift] ; Увеличити Y на 1.

call kurpos ; Встановити курсор.

n2:

cmp al,75 ; LEFT ?

jne n3 ; Якщо ні - далі.

cmp byte ptr ds:[xx-shift],0 ; Преревірити позицію курсору.

jne n22

mov byte ptr ds:[xx-shift],80

dec byte ptr ds:[yy-shift]

n22:

dec byte ptr ds:[xx-shift] ; Уменьшить X на 1.

call kurpos ; Встановити курсор.

n3:

cmp al,77 ; RIGHT ?

jne n4 ; Якщо ні - далі.

cmp byte ptr ds:[xx-shift],80 ; Перевірити позицію курсору.

jne n33

mov byte ptr ds:[xx-shift],0

inc byte ptr ds:[yy-shift]

n33:

inc byte ptr ds:[xx-shift] ; Увеличить X на 1.

call kurpos ; Встановити курсор.

n4:

cmp al,1 ; ESC ?

jne rd1 ; Якщо ні - далі.

cmp byte ptr ds:[fl2-shift],0 ; Інакше восстановити колір

je hi ; й вийти.

call unlabel

; Вихід.

hi:

call keybuf ; Очистити буфер клавіатури.

mov sp,bp

ret ; Завершитись.

start endp

; Опис змінних:

scrr db ?

xx db ?

yy db ?

x1 db ?

y1 db ?

tx db ?

ty db ?

wx db 2

wy db 2

fl2 db 0

flag db 0

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Кінець файлу MAIN.ASM \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Файл TAKE&PRN.ASM

; Модуль основної програми.

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

keybuf proc

; Очищає буфер клавіатури.

push es

push ax

sub ax,ax

mov es,ax

mov al,es:[41ah] ; Взяти кінець буферу

mov es:[41ch],al ; і записати в початок.

pop ax

pop es

ret

keybuf endp

delay proc

; Ініціірує затримку.

push cx

mov cx,60

l1:

push cx

mov cx,20000

l2:

loop l2

pop cx

loop l1

pop cx

ret

delay endp

del2 proc

; Ініціірує затримку.

push cx

mov cx,6000

a2:

loop a2

pop cx

ret

del2 endp

here proc

; Обробка виділення тексту.

push ax

cmp byte ptr ds:[flag-shift],0

je nh

mov al,byte ptr ds:[wx-shift]

cmp byte ptr ds:[xx-shift],al

jne lab

mov al,byte ptr ds:[wy-shift]

cmp byte ptr ds:[yy-shift],al

je rr

lab:

call labela ; Виділити.

jmp rr

nh:

mov al,byte ptr ds:[xx-shift]

mov byte ptr ds:[x1-shift],al

mov al,byte ptr ds:[yy-shift]

mov byte ptr ds:[y1-shift],al

rr:

pop ax

mov byte ptr ds:[flag-shift],1

ret

here endp

kurpos proc

; Встановлює курсор.

push ax

mov ah,02h

mov dh,byte ptr ds:[yy-shift]

mov dl,byte ptr ds:[xx-shift]

mov bh,0

call callvideo

pop ax

ret

kurpos endp

labela proc

; Змінює колір поміченого тексту.

cmp byte ptr ds:[fl2-shift],0

je f0

call unlabel

call ifswap

jmp f1

f0:

mov al,byte ptr ds:[x1-shift]

mov byte ptr ds:[wx-shift],al

mov al,byte ptr ds:[y1-shift]

mov byte ptr ds:[wy-shift],al

call ifswap

f1:

mov byte ptr ds:[fl2-shift],1

mov al,byte ptr ds:[x1-shift]

mov byte ptr ds:[tx-shift],al

mov al,byte ptr ds:[y1-shift]

mov byte ptr ds:[ty-shift],al

hh:

push cx

push dx

mov ah,02h ; Змінити позицію курсору.

mov dh,byte ptr ds:[ty-shift]

mov dl,byte ptr ds:[tx-shift]

mov bh,0

call callvideo

mov ah,08h ; Считати символ/атрибут.

mov bh,0

call callvideo

mov byte ptr ds:[scrr-shift],ah

mov ah,09h ; Записати символ з ізмененим атрибутом.

mov bl,byte ptr ds:[scrr-shift]

add bl,15

mov cx,1

mov bh,0

call callvideo

pop dx

pop cx

mov dl,byte ptr ds:[tx-shift]

cmp dl,byte ptr ds:[xx-shift]

je ok

as:

inc byte ptr ds:[tx-shift]

cmp byte ptr ds:[tx-shift],80

jne hh

mov byte ptr ds:[tx-shift],0

inc byte ptr ds:[ty-shift]

jmp hh

ok:

mov al,byte ptr ds:[yy-shift]

cmp byte ptr ds:[ty-shift],al

je rro ; Якщо кінець, то вийти,

jmp as ; інакше оброблювати наступний символ.

rro:

mov al,byte ptr ds:[xx-shift]

mov byte ptr ds:[wx-shift],al

mov al,byte ptr ds:[yy-shift]

mov byte ptr ds:[wy-shift],al

ret

labela endp

unlabel proc

; Встановлює колір поміченого тексту.

push ax

push dx

push cx

mov al,byte ptr ds:[x1-shift]

mov byte ptr ds:[tx-shift],al

mov al,byte ptr ds:[y1-shift]

mov byte ptr ds:[ty-shift],al

tl:

mov ah,02h ; Змінити позицію курсору.

mov dh,byte ptr ds:[ty-shift]

mov dl,byte ptr ds:[tx-shift]

mov bh,0

call callvideo

mov ah,08h ; Считати символ/атрибут.

mov bh,0

call callvideo

mov byte ptr ds:[scrr-shift],ah

call ToPrn

mov ah,09h ; Записати символ з востановленим атрибутом.

mov bl,byte ptr ds:[scrr-shift]

sub bl,15

mov cx,1

mov bh,0

call callvideo

mov al,byte ptr ds:[wx-shift]

cmp byte ptr ds:[tx-shift],al

je don

nt:

inc byte ptr ds:[tx-shift]

cmp byte ptr ds:[tx-shift],80

jne tl

call PrnEnt

mov byte ptr ds:[tx-shift],0

inc byte ptr ds:[ty-shift]

jmp tl

don:

mov al,byte ptr ds:[wy-shift]

cmp byte ptr ds:[ty-shift],al

je ex ; Якщо кінець, то вийти,

jmp nt ; інакше оброблювати наступний символ.

ex:

pop ax

pop cx

pop dx

ret

unlabel endp

ToPrn proc

; Печатает символ.

test byte ptr ds:[flags\_offset],mask Pf ; Перевірити прапорець печаті.

jz outp ; Якщо не дозволено - вийти.

mov ah,00h

mov dx,0

int 17h ; Печатать.

call del2

outp:

ret

ToPrn endp

PrnEnt proc

; Печатает ENTER

test byte ptr ds:[flags\_offset],mask Pf ; Перевірити прапорець печаті.

jz notp ; Якщо не дозволено - вийти.

mov ah,0

mov dx,0

mov al,0ah ; Печатати CR.

int 17h

mov ah,0

mov dx,0

mov al,0dh ; Печатать LF.

int 17h

notp:

ret

PrnEnt endp

ifswap proc

push ax

mov al,byte ptr ds:[y1-shift] ; Порівняти координати Y1 и YY

cmp al,byte ptr ds:[yy-shift]

jg zq

jl qw

mov al,byte ptr ds:[x1-shift] ; Порівняти координати X1 и XX

cmp al,byte ptr ds:[xx-shift]

jng qw

zq: ; Якщо треба змінити,

mov al,byte ptr ds:[yy-shift] ; змінюєм.

mov byte ptr ds:[y1-shift],al

mov al,byte ptr ds:[xx-shift]

mov byte ptr ds:[x1-shift],al

mov al,byte ptr ds:[wx-shift]

mov byte ptr ds:[xx-shift],al

mov al,byte ptr ds:[wy-shift]

mov byte ptr ds:[yy-shift],al

qw:

pop ax

ret ; Выход

ifswap endp

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Кінець файлу TAKE&PRN.ASM \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Файл REMOVE.ASM

; Модуль основної програми

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

test\_vector proc

; Перевірити вектор преривання.

mov ah,35h

int 21h

cmp bx,dx

jne exit

mov dx,es

cmp cx,dx

exit:

ret

test\_vector endp

remove proc

mov cx,cs

mov di,test\_vector-shift

mov al,09h ; Перевірити вектор преривання Int09h.

mov dx,int\_09h-shift

call di

jne not\_remove

mov al,2fh ; Перевірити вектор преривання Int2fh.

mov dx,int\_2fh-shift

call di

jne not\_remove

mov al,28h ; Перевірити вектор преривання Int28h.

mov dx,int\_28h-shift

call di

jne not\_remove

mov al,08h ; Перевірити вектор преривання Int08h.

mov dx,int\_08h-shift

call di

je uninstall

not\_remove:

mov ax,notremove\_mess-shift

push ax

push ax

call mess

ret

uninstall:

mov ax,remove\_mess-shift

push ax

xor ax,ax

push ax

call mess

mov cx,resethandler-shift

mov al,09h ; Восстановити вектор преривання Int09h.

mov di,int\_09h\_vect-shift

call cx

mov al,2fh ; Восстановити вектор преривання Int2fh.

mov di,int\_2fh\_vect-shift

call cx

mov al,28h ; Восстановити вектор преривання Int28h.

mov di,int\_28h\_vect-shift

call cx

mov al,08h ; Восстановити вектор преривання Int08h.

mov di,int\_08h\_vect-shift

call cx

cli

push cs

pop es

mov ah,49h ; Освободити займаєму пам`ять.

int 21h

ret

remove\_mess db ' has been',0,'REMOVED form memory. ',0

notremove\_mess db ' UNABLE to',0,'remove. Not on the top. ',0

remove endp

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Кінець файлу REMOVE.ASM \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Файл MESSAGE.ASM

; Модуль основної програми.

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

xchgmess proc

mov di,code\_table-shift ; Загрузити сміщення

mov si,attribute\_table-shift ; таблиць кодів і

; атрибутів вікна повідомлення.

mov cx,6

mov dh,10

loop\_1: ; Цикл по строках.

push cx

mov cx,30

mov dl,25

loop\_2: ; Цикл по стовбцях.

push cx

mov ah,2

call callvideo

mov ah,8

call callvideo

cmp dh,15

jb lab\_1

cmp dl,27

jb exch\_noth

jmp short exch\_attr

lab\_1:

cmp dh,11

jb lab\_2

cmp dl,52

ja exch\_attr

jmp short exch\_code

lab\_2:

cmp dl,52

ja exch\_noth

exch\_code:

xchg al,byte ptr [di] ; Заменити код символа.

inc di

exch\_attr:

xchg ah,byte ptr [si] ; Заменити атрибут.

inc si

exch\_noth:

mov bl,ah ; Вивести символ.

mov cx,1

mov ah,9

call callvideo

inc dx

pop cx

loop loop\_2

inc dh

pop cx

loop loop\_1

mov dx,1900h ; Убрати курсор.

mov ah,02h

call callvideo

ret ; Вийти з процедури.

xchgmess endp

mess proc

push bp

mov bp,sp

push bx

mov bh,byte ptr ds:[current\_video\_page-shift]

mov bl,byte ptr ds:[attribute\_table-shift]

push bx ; Вивести вікно повідомлення.

call xchgmess

pop bx

mov dx,0b29h ; Вивести текст повідомлення.

mov si,word ptr [bp+6]

call typestr

mov dx,0c1bh

call typestr

push bx ; Подати звуковий сигнал.

mov cx,2

mess\_loop:

mov di,550

mov bx,30

call beep

mov di,1100

call beep

loop mess\_loop

pop bx

wait\_mess:

mov ah,0 ; Ждати натиснення клавиши.

int 16h

call xchgmess ; Восстановити екран.

pop bx

pop bp

ret 4 ; Вийти з процедури.

mess endp

; Таблиця атрибутів вікна повідомлення.

attribute\_table db 29 dup(4eh),14 dup (206),13 dup (4eh)

db 07h,07h

db 28 dup (4eh)

db 07h,07h

db 4eh,26 dup (206),4eh

db 07h,07h

db 28 dup (4eh)

db 30 dup (07h)

; Образ атрибутів вікна повідомлення.

code\_table db '============================'

db '!Screen Printer: !'

db '! !'

db '! press any key... !'

db '============================'

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Кінець файлу MESSAGE.ASM \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Файл TYPESTR.ASM

; Модуль основної програми.

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

typestr proc

typestr\_loop:

mov ah,2 ; Встановити курсор в нужну позицию.

call callvideo

lodsb

or al,al ; Якщо зустрився кінець строки,

jz end\_typestr ; то вихід.

mov cx,1

mov ah,9

call callvideo

inc dx ; Определити позицію для вивода

jmp short typestr\_loop ; следуючого символу.

end\_typestr:

push dx ; Убрати курсор.

mov dx,1900h

mov ah,2

call callvideo

pop dx

ret ; Вийти з процедури.

typestr endp

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Кінець файлу TYPESTR.ASM \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Файл BEEP.ASM

; Модуль основної програми.

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

beep proc

; Подає звуковой сигнал заданої частоти і тривання.

push bx ; Сохранити регістри.

push cx

push di

mov di,1000

mov al,0b6h ; Записати у регістр режим таймеру.

out 43h,al

mov dx,14h ; Делитель часу равен

mov ax,4f38h ; 1331000/частота.

div di

out 42h,al ; Записати молодший байт лічильника таймера 2.

mov al,ah

out 42h,al ; Записати молодший байт лічильника таймера 2.

in al,61h ; Считати поточну установку порта В

push ax ; й сохранити її.

or al,3

out 61h,al ; Включити динамик.

wait\_beep:

mov cx,01fffh

speaker\_on:

loop speaker\_on

dec bx ; Лічильник тривалісті ісчерпан ?

jnz wait\_beep ; Ні. Продовжити звучання.

pop ax ; Да. Восстановити похідну установку

out 61h,al ; Порта В.

pop di ; Восстановити всі регистри.

pop cx

pop bx

ret ; Вийти з процедури.

beep endp

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Кінець файлу BEEP.ASM \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Файл TELETYPE.ASM

; Модуль основної програми.

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

teletype proc

mov bh,0 ; Обрати нулеву відеосторінку.

cld

type\_next\_char:

lodsb ; Загрузити наступний символ в AL.

mov ah,0eh

cmp al,0 ; Якщо кінець строки, то вихід.

je end\_teletype

call callvideo ; Вивести символ.

jmp short type\_next\_char

end\_teletype:

ret

teletype endp

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Кінець файлу TELETYPE.ASM \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*