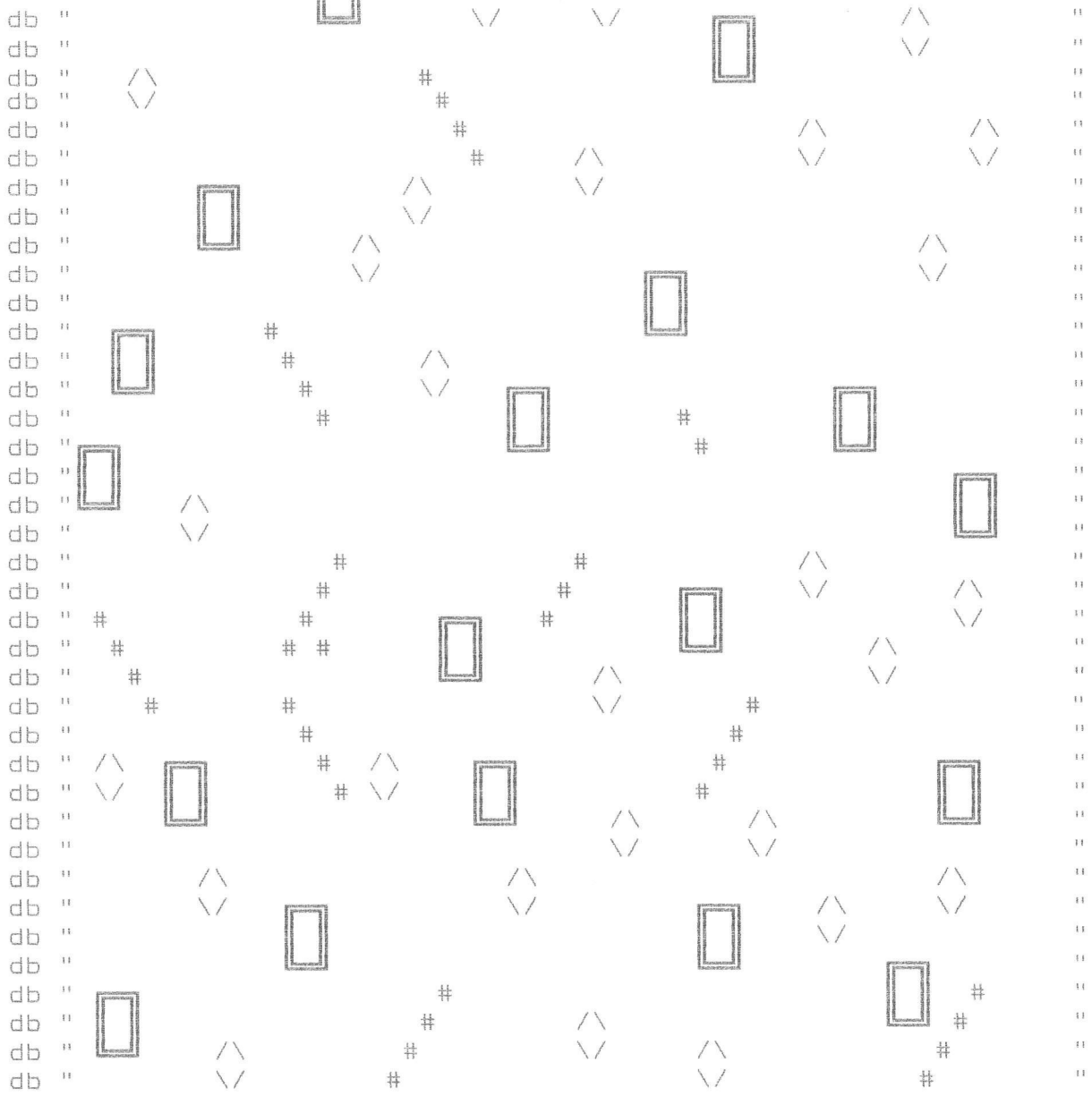
```
;Title:  Meteor Storm
;By:  Mark Petrick
;Student #:
;Prof:  Dr. Minoru Hasegawa
;Class:  Math 2473  Assembly Language
;Date:  April 24, 1995
;Music:  Minuet in F
;Composer:  Wolfgang Amadeus Mozart (1756 - 1791)

;     Meteor Storm is a game in where the player flies his spaceship through
;a meteor shower.  If the spaceship is hit by a meteor or the spaceship runs
;into the boundaries of the screen, his life is forfeit.
;The following is a list of valid keys that may be used during the game:
;       Escape key        -     Exits the game back to DOS
;       Cursor Up         -     Moves spaceship up one character
;       Cursor Down       -     Moves spaceship down one character
;       Cursor Right      -     Moves spaceship right one character
;       Cursor Left       -     Moves spaceship left one character
;All other keys may be pressed but their value will be ignored.

data segment

;Music score for Minuet in F
score  db 4,2,5,2,6,2,5,2,6,2,7,2,6,2,7,2,8,2,8,4,7,4,6,4,10,1,9,1,8,1,7,1
       db 6,4,5,4,4,6,13,2,5,1,18,1,5,1,13,2,5,1,18,1,5,1,14,2,5,1,18,1,5,1
       db 15,2,5,1,8,1,5,1,16,2,5,1,9,1,5,1,17,2,5,1,10,1,5,1,13,2,5,1,18,1
       db 5,1,13,2,5,1,18,1,5,1,14,2,5,1,18,1,5,1,15,2,5,1,8,1,5,1,16,2,5,1
       db 9,1,5,1,17,2,5,1,10,1,5,1,14,2,19,2,9,2,13,2,10,2,8,2,13,2,9,2
       db 18,2,18,5,8,4
       db 4,2,5,2,6,2,5,2,6,2,7,2,6,2,7,2,8,2,8,4,7,4,6,4,10,1,9,1,8,1,7,1
       db 6,4,5,4,4,6,13,2,5,1,18,1,5,1,13,2,5,1,18,1,5,1,14,2,5,1,18,1,5,1
       db 15,2,5,1,8,1,5,1,16,2,5,1,9,1,5,1,17,2,5,1,10,1,5,1,13,2,5,1,18,1
       db 5,1,13,2,5,1,18,1,5,1,14,2,5,1,18,1,5,1,15,2,5,1,8,1,5,1,16,2,5,1
       db 9,1,5,1,17,2,5,1,10,1,5,1,14,2,19,2,9,2,13,2,10,2,8,2,13,2,9,2
       db 18,2,18,5,8,4
       db 4,2,5,2,6,2,6,2,7,2,8,2,8,2,9,2,20,2,9,4,6,4,7,4,5,2,6,2,18,2,18,2
       db 8,2,9,2,9,2,10,2,11,2,10,4,18,4,8,4,1,2,1,1,3,1,1,1,1,2,1,1,3,1,1,1
       db 11,2,1,1,3,1,1,1,12,2,1,1,4,1,1,1,13,2,1,1,5,1,1,1,14,2,1,1,6,1,1,1
       db 1,2,8,1,3,1,8,1,1,2,8,1,3,1,8,1,11,2,8,1,3,1,8,1,12,2,8,1,4,1,8,1
       db 13,2,8,1,5,1,8,1,14,2,8,1,6,1,8,1,11,2,7,2,5,2,1,2,6,2,4,2,17,2,5,2
       db 3,2,4,2,6,2,5,2,4,2,3,2,2,2,1,4,10,1,9,1,8,1,7,1,6,4,5,1,6,1,5,1
       db 4,1,5,1,4,6
       db 4,2,5,2,6,2,6,2,7,2,8,2,8,2,9,2,20,2,9,4,6,4,7,4,5,2,6,2,18,2,18,2
       db 8,2,9,2,9,2,10,2,11,2,10,4,18,4,8,4,1,2,1,1,3,1,1,1,1,2,1,1,3,1,1,1
       db 11,2,1,1,3,1,1,1,12,2,1,1,4,1,1,1,13,2,1,1,5,1,1,1,14,2,1,1,6,1,1,1
       db 1,2,8,1,3,1,8,1,1,2,8,1,3,1,8,1,11,2,8,1,3,1,8,1,12,2,8,1,4,1,8,1
       db 13,2,8,1,5,1,8,1,14,2,8,1,6,1,8,1,11,2,7,2,5,2,1,2,6,2,4,2,17,2,5,2
       db 3,2,4,2,6,2,5,2,4,2,3,2,2,2,1,4,11,2,7,2,5,2,17,2,6,2,4,2,17,2,5,2
       db 3,2,3,5,4,7,21


;notes: C,D,E,F,G,A,Bf,hC,hD,hE,Bf,A,G,F,E,D,lC,B,hF,hEf
notes  dw 11cah,0fdah,0e20h,0d5bh,0be4h,0a98h,0a00h,08e9h
       dw 07e5h,0708h
       dw 1401h,152fh,17c8h,1aa2h,1c3fh,1fb5h,2394h
       dw 096fh,06a5h,0776h,0

;note delay lengths
length db 4,6,9,10,15,17,20

;Title screen layout data
```

```
screen1 db "MM      MM  EEEEEEE  TTTTTTTT  EEEEEEE     OOO     RRRRR   "
        db "MMM    MMM  EEEEEEE  TTTTTTTT  EEEEEEE    OOOOO    RRRRRR  "
        db "MMMMMMMMMM  EE          TT     EE        OO   OO   RR    RR"
        db "MM MM MM    EEEEE       TT     EEEEE     OO   OO   RR    RR"
        db "MM      MM  EEEEE       TT     EEEEE     OO   OO   RRRRRR  "
        db "MM      MM  EE          TT     EE        OO   OO   RRRRR   "
        db "MM      MM  EEEEEEE     TT     EEEEEEE    OOOOO    RR    RR"
        db "MM      MM  EEEEEEE     TT     EEEEEEE     OOO     RR    RR"
        db "                                                           "
        db "                                                           "
        db "     SSSSS    TTTTTTTT     OOO     RRRRR    MM     MM      "
        db "    SSSSSSS   TTTTTTTT    OOOOO    RRRRRR   MMM   MMM      "
        db "    SS   SS      TT      OO   OO   RR    RR MMMMMMMMM      "
        db "     SSS         TT      OO   OO   RR    RR MM MM MM       "
        db "       SSS       TT      OO   OO   RRRRRR   MM     MM      "
        db "    SS   SS      TT      OO   OO   RRRRR    MM     MM      "
        db "    SSSSSSS      TT       OOOOO    RR    RR MM     MM      "
        db "     SSSSS       TT        OOO     RR    RR MM     MM      "
        db "                                                           "
        db "     By  Mark Petrick                                      "
        db " For  Dr Minoru Hasegawa     Math 2473     March 1995      "
        db "                  PRESS A KEY TO CONTINUE                  "

;In-game screen data
playscrn db "╔═════════╗╔═════════╗"
         db "║         ║║         ║"
         db "║   M M   ║║   SSS   ║"
         db "║   MMM   ║║   SS    ║"
         db "║ M M M M ║║ M  SS M ║"
         db "║ E EEE E ║║ E SSS E ║"
         db "║ T EE  T ║║ T TTT T ║"
         db "║ E EEE E ║║ E  T  E ║"
         db "║ O TTT O ║║ O  T  O ║"
         db "║ R  T  R ║║ R  O  R ║"
         db "║    T    ║║   O O   ║"
         db "║ S EEE S ║║ S O O S ║"
         db "║ T EE  T ║║ T  O  T ║"
         db "║ O EEE O ║║ O RRR O ║"
         db "║ R  O  R ║║ R RRR R ║"
         db "║ M O O M ║║ M RR  M ║"
         db "║    O    ║║   R R   ║"
         db "║   RRR   ║║   M M   ║"
         db "║   RRR   ║║   MMM   ║"
         db "║   R R   ║║   M M   ║"
         db "║         ║║         ║"
         db "╚═════════╝╚═════════╝"

;In-game Message that scrolls along line 24
message db "By Mark Petrick                                                    "
        db "                         For Dr. Minoru Hasegawa          Math 2473"
        db "                                            March 1994             "
        db "                                                                   "

;The data for the Meteor Shower
meteors db "                        #                                /\       "
        db " /\         ╔═╗          #          /\          ╔═╗      \/       "
        db " \/         ║ ║           #         \/          ║ ║   /\          "
        db "            ╚═╝            #            /\      ╚═╝   \/          "
        db "     /\              #                  \/                        "
        db "     \/             #              ╔═╗                            "
        db "                   #        /\     ║ ║       #                    "
        db "                            \/     ╚═╝        #                   "
        db "  #                                     /\      #     #   ╔═╗     "
        db "   #                                    \/              # ║ ║     "
        db "    #                                                  #  ╚═╝     "
        db "     #            ╔═╗          /\     /\                          "
```



```
;End-of-Game message data
gameendmsg db "G A M E      O V E R"
           db "PRESS A KEY TO CONTINUE"

;The graphics for the player's spaceship
shipgfx db 07fh,0dbh,05eh

;The offset of the spaceship in screen memory
shippos dw 0b8eh

;An index used for scrolling through the Meteor Shower data
lineidx dw 2899

;An index used for scrolling through the In-game message
msgidx dw 0

;Storage location for the old INT09 vector and current status of the port
vectora dw 0
vectorb dw 0
device db 0

;The delay factor for the meteor shower scrolling
delayfactora db 15

;The delay factor for the In-game message scrolling
delayfactorb db 17
```

```
;Indexes in time delay used for scrolling calls
tmpcounta db 15
tmpcountb db 17

data   ends

;Location of the value of Keypress
keycode segment
     dw 0
keycode ends

stack  segment stack
       db  100h  dup(?)
stack  ends

main   segment
assume cs:main, ds:data, ss:stack
drive  proc far
       push ds
       mov ax,0
       push ax
       mov cx,0
       call setupkeyscan  ;Sets up the key scan routine
       call runseqroutine ;Main sequence of calls and setup for game
       ret
drive  endp

initializestuff proc
;Reset indexes and variables used when the game is started over
       mov ax,data
       mov es,ax
       mov ax,0b8eh
       mov es:[shippos],ax
       mov ax,2899
       mov es:[lineidx],ax
       mov ax,0
       mov es:[msgidx],ax
       ret
initializestuff endp

runseqroutine proc
;Main routine which runs the game
       call initializestuff
       call initscreen
       call contkey
       call gamescrn
       call music
runseqroutine endp

initscreen proc
;prints the title screen from screen1 in the data segment to screen memory
          call clrscrn
          push ax
          push bx
          push cx
          push dx
          push ds
          push es
          push si
          mov ax,0b800h
          mov ds,ax
          mov ax,data
          mov es,ax
          mov bx,15ch
          mov si,0
          mov dl,3
```

```
            mov dh,0
            mov al,0f0h  ;reverse blinking code
mvchar:     mov cl,es:[screen1+si]
            mov [bx],cl
            inc si
            inc bx
            cmp dl,24
            jnz notlastln
            mov [bx],al
notlastln:  inc bx
            inc dh
            cmp dh,54
            jz nextln
            jmp mvchar
done:       pop si
            pop es
            pop ds
            pop dx
            pop cx
            pop bx
            pop ax
            ret
nextln:     mov dh,0
            add bx,34h
            cmp dl,24
            jz done
            inc dl
            jmp mvchar
initscreen endp

clrscrn proc
;clears the screen
          push ax
          push bx
          push ds
          mov ax,0b800h
          mov ds,ax
          mov bx,0
          mov al,32
          mov ah,07h
clrmore:  mov [bx],al
          inc bx
          mov [bx],ah
          inc bx
          cmp bx,0fa2h
          jnz clrmore
          pop ds
          pop bx
          pop ax
          ret
clrscrn endp

gamescrn proc
;prints the in-game screen from playscrn in the data segment to screen memory
          call clrscrn
          call printship
          push ax
          push bx
          push cx
          push dx
          push ds
          push es
          push si
          mov ax,0b800h
          mov ds,ax
          mov bx,0
```

```
          mov si,0
          mov ax,data
          mov es,ax
          mov cl,07h
          mov dl,0
          mov dh,1
clmnone:  mov ch,es:[playscrn+si]
          mov [bx],ch
          inc dl
          inc si
          inc bx
          mov [bx],cl
          inc bx
          cmp dl,11
          jnz clmnone
          add bx,74h
          mov dl,0
clmntwo:  mov ch,es:[playscrn+si]
          mov [bx],ch
          inc dl
          inc si
          inc bx
          mov [bx],cl
          inc bx
          cmp dl,11
          jnz clmntwo
          mov dl,0
          inc dh
          cmp dh,23
          jz msgln
          jmp clmnone
msgln:    mov bx,0dc0h
          mov ch,205
          mov dl,0
smln:     mov [bx],ch
          inc bx
          mov [bx],cl
          inc bx
          inc dl
          cmp bx,0fa0h
          jz donea
          cmp dl,80
          jnz smln
          mov bx,0f00h
          jmp smln
donea:    pop si
          pop es
          pop ds
          pop dx
          pop cx
          pop bx
          pop ax
          ret
gamescrn endp

endgamescrn proc
;Prints the end-of-game screen when player's ship crashes
           call disspeak
           push ax
           push bx
           push ds
           push es
           push si
           mov ax,0b800h
           mov ds,ax
           mov ax,data
```

```
            mov es,ax
            mov al,0
            mov bx,es:[shippos]
moreexpl:   mov ah,02ah
            mov [bx],ah
            add bx,0a0h
            inc al
            cmp al,3
            jl moreexpl
            mov al,0
            mov bx,516h
clrlna:     mov ah,32
            mov [bx],ah
            inc bx
            inc bx
            inc al
            cmp al,58
            jl clrlna
            mov bx,5b6h
            mov al,0
clrlfta:    mov ah,32
            mov [bx],ah
            inc bx
            inc bx
            inc al
            cmp al,19
            jl clrlfta
            mov al,0
            mov si,0
prnendgm:   mov ah,es:[gameendmsg+si]
            mov [bx],ah
            inc bx
            inc bx
            inc si
            inc al
            cmp al,20
            jl prnendgm
            mov al,0
clrrghta:   mov ah,32
            mov [bx],ah
            inc bx
            inc bx
            inc al
            cmp al,19
            jl clrrghta
            mov bx,656h
            mov al,0
clrlnb:     mov ah,32
            mov [bx],ah
            inc bx
            inc bx
            inc al
            cmp al,58
            jl clrlnb
            mov al,0
            mov bx,0e60h
clrlftb:    mov ah,32
            mov [bx],ah
            inc bx
            mov ah,0f0h
            mov [bx],ah
            inc bx
            inc al
            cmp al,28
            jl clrlftb
            mov al,0
```

```
prncont:  mov ah,es:[gameendmsg+si]
          mov [bx],ah
          inc bx
          mov ah,0f0h
          mov [bx],ah
          inc bx
          inc al
          inc si
          cmp al,23
          jl prncont
          mov al,0
clrrghtb: mov ah,32
          mov [bx],ah
          inc bx
          mov ah,0f0h
          mov [bx],ah
          inc bx
          inc al
          cmp al,29
          jl clrrghtb
          pop si
          pop es
          pop ds
          pop bx
          pop ax
          call contkey
          call runseqroutine
          ret
endgamescrn endp

printship proc
;prints the ship in position stored in shippos in data segment
          push ax
          push bx
          push ds
          push es
          push si
          mov ax,0b800h
          mov ds,ax
          mov ax,data
          mov es,ax
          mov si,0
          mov bx,es:[shippos]
moreship: mov al,es:[shipgfx+si]
          mov [bx],al
          inc si
          add bx,0a0h
          cmp si,3
          jl moreship
          pop si
          pop es
          pop ds
          pop bx
          pop ax
          ret
printship endp

eraseship proc
;erases the ship from shippos
        push ax
        push bx
        push ds
        push es
        mov ax,data
        mov es,ax
        mov ax,0b800h
```

```
        mov ds,ax
        mov bx,es:[shippos]
        mov ah,0
        mov al,32
ershp:  mov [bx],al
        inc ah
        add bx,0a0h
        cmp ah,3
        jl ershp
        pop es
        pop ds
        pop bx
        pop ax
        ret
eraseship endp


scrollscrn proc
;scrolls the meteor data down one line from meteors in the data segment
            push ax
            push bx
            push cx
            push dx
            push ds
            push si
            push es
            call chkupcollision
            call eraseship
            mov ax,0b800h
            mov ds,ax
            mov bx,0c96h
            mov si,0d36h
            mov dx,0
movln:      mov ch,[bx]
            mov [si],ch
            inc si
            inc si
            inc bx
            inc bx
            inc dl
            cmp dl,58
            jnz movln
            inc dh
            sub bx,114h
            sub si,114h
            mov dl,0
            cmp dh,21
            jnz movln
            mov dx,0
            mov bx,136
            mov ax,data
            mov es,ax
            mov si,es:[lineidx]
moreln:     mov ch,es:[meteors+si]
            mov [bx],ch
            dec bx
            dec bx
            inc dl
            cmp si,0
            jz rstlineidx
            dec si
            cmp dl,58
            jnz moreln
            mov es:[lineidx],si
donescrl:   call printship
            pop es
```

```
            pop si
            pop ds
            pop dx
            pop cx
            pop bx
            pop ax
            ret
rstlineidx: mov es:[lineidx],2899
            jmp donescrl
scrollscrn endp

scrollmsg proc
;scrolls the message line (line 24) to the left one
;the message is stored in message in the data segment
            push ax
            push bx
            push dx
            push ds
            push si
            push es
            mov ax,0b800h
            mov ds,ax
            mov ax,data
            mov es,ax
            mov bx,3680
            mov si,3682
            mov dx,0
gscrl:      mov dl,[si]
            mov [bx],dl
            inc si
            inc si
            inc bx
            inc bx
            inc dh
            cmp dh,79
            jnz gscrl
            mov ax,es:[msgidx]
            mov si,ax
            mov dl,es:[message+si]
            mov [bx],dl
            inc si
            mov ax,si
            mov es:[msgidx],ax
            cmp si,248
            jz rstmsgidx
doneb:      pop es
            pop si
            pop ds
            pop dx
            pop bx
            pop ax
            ret
rstmsgidx:  mov ax,0
            mov es:[msgidx],ax
            jmp doneb
scrollmsg endp

setupkeyscan proc
;Sets up the Key scan routine
            push ax
            push ds
            push es
            push si
            mov ax,data
            mov es,ax
            mov ax,0
```

```
            mov ds,ax
            mov si,024h
            mov ax,[si]   ;relocate INT09 to the data segment
            mov es:[vectora],ax
            mov ax,[si+2]
            mov es:[vectorb],ax
            in al,21h
            mov es:[device],al
            cli
            mov ax,offset continuekey  ;remap INT09 to continuekey procedure
            mov [si],ax
            mov ax,cs
            mov 2[si],ax
            mov al,0fdh   ;disable all devices except keyboard
            out 21h,al
            sti
            pop si
            pop es
            pop ds
            pop ax
            ret
setupkeyscan endp

restoreoldkeyscan proc
;Places the old INT09 vector back into position from the data segment
            push ax
            push bx
            push ds
            push es
            push si
            mov ax,data
            mov es,ax
            mov ax,0
            mov ds,ax
            mov si,024h
            mov cx,es:[vectora]
            mov bx,es:[vectorb]
            mov dl,es:[device]
            mov ax,0
            mov ds,ax
            mov es,ax
            cli
            mov [si],cx
            mov [si+2],bx
            mov al,dl
            out 21h,al
            sti
            pop si
            pop es
            pop ds
            pop bx
            pop ax
            ret
restoreoldkeyscan endp

continuekey proc
;The new interrupt for INT09, places the key scan value in keycode
      push ax
      push bx
      push cx
      push ds
      in al,060h
      push ax
      in al,061h
      or al,080h
      out 61h,al
```

```
        and al,07fh
        out 61h,al
        pop ax
        test al,80h
        jz skip
        mov al,0ffh
skip:   mov cx,keycode
        mov ds,cx
        mov bx,0
        mov [bx],al
        mov al,20h
        out 20h,al
        pop ds
        pop cx
        pop bx
        pop ax
        iret
continuekey endp

contkey proc
;Check key to start or reset game, and check for Esc to exit to DOS
                push ax
                push bx
                push ds
                mov ax,keycode
                mov ds,ax
                mov bx,0
                mov ax,0ffh
                mov [bx],ax
cont1:          cli
                cmp [bx],ax
                jnz keyp
                sti
                jmp cont1
keyp:           mov al,1
                cmp [bx],ax
                jne donecontkey
                call returndos
donecontkey:    pop ds
                pop bx
                pop ax
                ret
contkey endp

returndos proc
;The return to DOS procedure
      call disspeak
      call clrscrn
      call restoreoldkeyscan
      mov ax,0
      mov ds,ax
      mov ax,4c00h ;exit to DOS interrupt call
      int 21h
      ret
returndos endp

chkupcollision proc
;Check for a collision above the spaceship
                push ax
                push bx
                push ds
                push es
                mov ax,data
                mov es,ax
                mov ax,0b800h
                mov ds,ax
```

```
                mov ax,0
                mov bx,es:[shippos]
                sub bx,0a0h
                mov ah,[bx]
                cmp ah,32
                jnz gameoverrout
contupcoll :    pop es
                pop ds
                pop bx
                pop ax
                ret
gameoverrout: call endgamescrn
                jmp contupcoll
chkupcollision endp

chkdowncollision proc
;Check for a collision below the spaceship
                push ax
                push bx
                push ds
                push es
                mov ax,data
                mov es,ax
                mov ax,0b800h
                mov ds,ax
                mov ax,0
                mov bx,es:[shippos]
                add bx,0a0h
                add bx,0a0h
                add bx,0a0h
                mov ah,[bx]
                cmp ah,32
                jnz gameoverd
contdowncoll: pop es
                pop ds
                pop bx
                pop ax
                ret
gameoverd:      call endgamescrn
                jmp contdowncoll
chkdowncollision endp

chkrightcollision proc
;Check for a collision to the right of the spaceship
                 push ax
                 push bx
                 push ds
                 push es
                 mov ax,data
                 mov es,ax
                 mov ax,0b800h
                 mov ds,ax
                 mov ax,0
                 mov bx,es:[shippos]
                 inc bx
                 inc bx
                 mov ah,[bx]
                 cmp ah,32
                 jnz gameoverr
                 add bx,0a0h
                 mov ah,[bx]
                 cmp ah,32
                 jnz gameoverr
                 add bx,0a0h
                 mov ah,[bx]
                 cmp ah,32
```

```
                jnz gameoverr
contrightcoll:  pop es
                pop ds
                pop bx
                pop ax
                ret
gameoverr:      call endgamescrn
                jmp contrightcoll
chkrightcollision endp

chkleftcollision proc
;Check for a collision to the left of the spaceship
                push ax
                push bx
                push ds
                push es
                mov ax,data
                mov es,ax
                mov ax,0b800h
                mov ds,ax
                mov ax,0
                mov bx,es:[shippos]
                dec bx
                dec bx
                mov ah,[bx]
                cmp ah,32
                jnz gameover1
                add bx,0a0h
                mov ah,[bx]
                cmp ah,32
                jnz gameover1
                add bx,0a0h
                mov ah,[bx]
                cmp ah,32
                jnz gameover1
contleftcoll:   pop es
                pop ds
                pop bx
                pop ax
                ret
gameover1:      call endgamescrn
                jmp contleftcoll
chkleftcollision endp

processkeypress proc
;check keycode for a new value
        push ax
        push bx
        push ds
        mov ax,keycode
        mov ds,ax
        mov bx,0
        mov ax,0ffh
        cli
        cmp [bx],ax
        jnz keyin
        sti
        jmp contkp
keyin:  call keypress
contkp: pop ds
        pop bx
        pop ax
        ret
processkeypress endp

keypress proc
```

```
;check keycode for escape, cursor-up, cursor-down, cursor-left, cursor-right
                push ax
                push bx
                push ds
                push es
                mov ax,keycode
                mov ds,ax
                mov ax,data
                mov es,ax
                mov bx,0
                mov ax,1
                cmp [bx],ax
                je esc
                mov ax,72
                cmp [bx],ax
                je moveup
                mov ax,80
                cmp [bx],ax
                je movedown
                mov ax,77
                cmp [bx],ax
                je moveright
                mov ax,75
                cmp [bx],ax
                je moveleft
                jmp endkeypress
esc:            call returndos
                jmp endkeypress
moveup:         call chkupcollision
                call eraseship
                mov ax,es:[shippos]
                sub ax,0a0h
                mov es:[shippos],ax
                call printship
                jmp endkeypress
movedown:       call chkdowncollision
                call eraseship
                mov ax,es:[shippos]
                add ax,0a0h
                mov es:[shippos],ax
                call printship
                jmp endkeypress
moveright:      call chkrightcollision
                call eraseship
                mov ax,es:[shippos]
                inc ax
                inc ax
                mov es:[shippos],ax
                call printship
                jmp endkeypress
moveleft:       call chkleftcollision
                call eraseship
                mov ax,es:[shippos]
                dec ax
                dec ax
                mov es:[shippos],ax
                call printship
                jmp endkeypress
endkeypress:    mov ax,0ffh
                mov [bx],ax
                pop es
                pop ds
                pop bx
                pop ax
                ret
keypress endp
```

```
conspeak proc
;connects the speaker
      in al,61h
      or al,3
      out 61h,al
      mov al,0b6h
      out 43h,al
      ret
conspeak endp

disspeak proc
;disconnects the speaker
      in al,61h
      and al,0fch
      out 61h,al
      ret
disspeak endp

timedelay proc
;The MAIN TIMING ROUTINE that controls the meteor scrolling and message
;scrolling and the music note delays
                call processkeypress
                mov cx,data
                mov es,cx
                mov cl,es:[tmpcounta]
                mov ch,es:[tmpcountb]
contplay:       cmp al,0
                jl donenote
                cmp cl,0
                jz procscrllscrn
conta:          cmp ch,0
                jz procscrllmsg
contb:          call normdelay
                call processkeypress
                dec al
                dec cl
                dec ch
                jmp contplay
procscrllscrn:  mov cl,es:[delayfactora]
                mov es:[tmpcounta],cl
                call scrollscrn
                call processkeypress
                jmp conta
procscrllmsg:   mov ch,es:[delayfactorb]
                mov es:[tmpcountb],ch
                call scrollmsg
                call processkeypress
                jmp contb
donenote:       mov es:[tmpcounta],cl
                mov es:[tmpcountb],ch
                ret
timedelay endp

normdelay proc
;An extra delay for the music notes
        push cx
        mov cx,1700h
repeat: nop
        dec cx
        jge repeat
        pop cx
        ret
normdelay endp

music proc
```

```
;the music playing routine
start:     call conspeak
           mov ax,data
           mov ds,ax
           mov si,0
play:      mov al,[si]
           add al,al
           mov ah,0
           mov di,ax
           mov dx,notes[di]
           cmp dx,0
           jz oncemore
           mov al,dl
           out 42h,al
           mov al,dh
           out 42h,al
           inc si
           mov al,[si]
           mov ah,0
           mov di,ax
           mov al,length[di]
           call timedelay
           inc si
           jmp play
oncemore:  call disspeak
           mov ax,32
           call timedelay
           dec cx
           jns start
           ret
music  endp

main   ends
end    drive
```