[org 0x0100]

jmp start

count: dw 0

prev: dw 3

difficulty: dw 0

score: dw 0

message: db 'Score'

message2: db 'GAME OVER'

message3: db 'Your score is: '

jmpFlag: dw 0

jmpCount: dw 0

over: dw 0

soundflag: dw 1

soundflag1: dw 1

soundflag2: dw 1

soundflag3: dw 1

downflag: dw 0

oldkb: dd 0

oldtim: dd 0

clrscr:

push es

push ax

push di

mov ax, 0xb800

mov es, ax ; point es to video base

mov di, 0 ; point di to top left column

nextloc:

mov word [es:di], 0x0720 ; clear next char on screen

add di, 2 ; move to next screen location

cmp di, 4000 ; has the whole screen cleared

jne nextloc ; if no clear next position

pop di

pop ax

pop es

ret

clrscr2:

push es

push ax

push di

mov ax, 0xb800

mov es, ax ; point es to video base

mov di, 0 ; point di to top left column

nextloc2:

mov word [es:di], 0x0720 ; clear next char on screen

add di, 2 ; move to next screen location

cmp di, 1920 ; has the whole screen cleared

jne nextloc2 ; if no clear next position

pop di

pop ax

pop es

ret

printnum: push bp

mov bp, sp

push es

push ax

push bx

push cx

push dx

push di

mov ax, 0xb800

mov es, ax

mov ax, [bp+4]

mov bx, 10

mov cx, 0

nextdigit:

mov dx, 0

div bx

add dl, 0x30

push dx

inc cx

cmp ax, 0

jnz nextdigit

cmp word [over] , 1

jne simple

mov di , 1692 ; ending score

jmp ending

simple:

mov di, 460 ; display score

ending:

nextpos: pop dx

mov dh, 15

mov [es:di], dx

add di, 2

loop nextpos

pop di

pop dx

pop cx

pop bx

pop ax

pop es

pop bp

ret 2

delay:

pusha

pushf

mov cx,1000

mydelay:

mov bx,50 ;; increase this number if you want to add more delay, and decrease this number if you want to reduce delay.

mydelay1:

dec bx

jnz mydelay1

loop mydelay

popf

popa

ret

RANDNUM:

push bp

mov bp,sp

push ax

push cx

push dx

MOV AH, 00h ; interrupts to get system time

INT 1AH ; CX:DX now hold number of clock ticks since midnight

mov ax, dx

xor dx, dx

mov cx, [bp+4]

inc cx

div cx ; here dx contains the remainder of the division - from 0 to 9

mov [bp+6], dx

pop dx

pop cx

pop ax

pop bp

ret 2

displayGround:

pusha

pushf

push 0xb800

pop es

mov di , 0

mov al , '\_'

mov ah , 2

mov dl , '\_'

mov dh , 102

loop1:

mov [es:21\*160+di] , dx

mov [es:20\*160+di] , ax

add di,2

cmp di , 160

jne loop1

push word [score]

call printnum

mov ah, 0x13 ; service 13 - print string

mov al, 1 ; subservice 01 – update cursor

mov bh, 0 ; output on page 0

mov bl, 15 ; normal attrib

mov dx, 0x018F ; row 10 column 3

mov cx, 5 ; length of string

push cs

pop es ; segment of string

mov bp, message ; offset of string

int 0x10

popf

popa

ret

cactusmaker:

mov bp, sp

pusha

mov bx, [bp+2] ; moving the place of the mountain in bx.

push 0xb800

pop es

mov dh , 07

mov dl , 'O'

mov ax , [bp + 4]

cmp ax , 0

jne notline

cmp word [cs:prev], 4

jne notabove

mov word [cs:prev], 1

notabove:

add word [cs:prev], 1

notline:

mov ax , [cs:prev]

mov di, 150

sub di, bx

cmp ax,3

jne l3

l:

mov dh , 32

mov [es:20\*160+di] , dx ; level 3 cactus

mov [es:19\*160+di+2] , dx

mov [es:18\*160+di+4] , dx

mov [es:19\*160+di+6] , dx

mov [es:20\*160+di+8] , dx

mov dh , 36

mov dl , 'X'

mov [es:20\*160+di+2] , dx

mov [es:20\*160+di+4] , dx

mov [es:20\*160+di+6] , dx

mov [es:19\*160+di+4] , dx

jmp goout

l3: ; level 2 cactus here

cmp ax , 2

jne l4

mov dh , 15

mov [es:19\*160 +di+2] , dx

mov [es:20\*160 + di+2] , dx

mov dl , '\'

mov [es:18\*160 +di] , dx

mov dl , '/'

mov [es:18\*160 +di + 4] , dx

jmp goout

l4: ; Level 1 cactus here

mov dh , 5

mov dl , '\*'

mov [es:18\*160 + di] , dx

mov [es:18\*160 + di + 2] , dx

mov [es:19\*160 + di] , dx

mov [es:19\*160 + di + 2] , dx

mov dh , 15

mov dl , '|'

mov [es:20\*160 + di] , dx

mov [es:20\*160 + di + 2] , dx

goout:

popa

ret 4

dinasourmaker:

mov bp, sp

pushf

pusha

push 0xb800

pop es

mov bx, 3040

cmp word [cs:jmpFlag] , 0

je nojump

mov ax , [cs:jmpCount]

jumpup:

cmp word [cs:downflag], 1

je jumpdown

cmp ax, 7

ja jumpdown

mov cl, 160

mul cl

sub bx, ax

inc word [cs:jmpCount]

jmp nojump

jumpdown:

mov word [cs:downflag], 1

cmp word [cs:jmpCount], 1

je setCount0

dec word [cs:jmpCount]

mov cl , 160

mul cl

sub bx, ax

jmp nojump

setCount0:

mov word [cs:jmpFlag],0

mov word [cs:downflag],0

mov word [cs:jmpCount],0

nojump:

mov ah , 07h

mov al , 0

mov di, 16 ; change the position of player from here.

mov [es: bx +162+di],ax

mov ah , 12

mov al , '/'

mov [es: bx+166+di],ax

mov al , 0

mov [es: bx+168+di],ax

mov al , '\'

mov [es: bx+170+di],ax

mov ah , 11

mov al , '\_'

mov [es: bx+164+di],ax

mov al , '\_'

mov [es: bx+172+di],ax

mov al , 0

mov [es: bx+174+di],ax

sub bx, 160

mov ah , 14

mov al , '|'

mov [es: bx+168+di],ax

mov ah , 11

mov al , '/'

mov [es: bx+164+di],ax

mov al , '\'

mov [es: bx+172+di],ax

sub bx, 160

mov ah , 12

mov al , '-'

mov [es: bx+168+di],ax

mov ah , 11

mov al , '{'

mov [es: bx+166+di],ax

mov al , '}'

mov [es: bx+170+di],ax

sub bx, 160

mov ah , 11

mov al , '('

mov [es: bx+164+di],ax

mov al , ')'

mov [es: bx+172+di],ax

mov ah , 14

mov al , 'o'

mov [es: bx+166+di],ax

mov al , 'o'

mov [es: bx+170+di],ax

popa

popf

ret

cactusMover:

pusha

call clrscr

call displayGround

call dinasourmaker

mov bx,[cs:count]

cmp bx , 0

jne it1

push 0

jmp it2

it1:

push 1

it2:

push bx

call cactusmaker

add bx, 2

cmp bx, 150

jne noChange

mov word [cs:count], 0

jmp endd

noChange:

mov [cs:count], bx

endd:

push 0xb800

pop es

mov dl , '|'

mov dh , 14

cmp word [es:19\*160+24], dx

jne checknext

mov dh , 11

mov dl , '\_'

cmp word [es:20\*160+28], dx

jne pauseit

jmp checknext

checknext:

mov dl , '|'

mov dh , 14

cmp word [es:17\*160+24], dx

jne success

mov dl , '\'

mov dh , 12

cmp word [es:18\*160+26], dx

jne pauseit

mov dl , 0

cmp word [es:18\*160+24], dx

jne pauseit

mov dl , '/'

cmp word [es:18\*160+22], dx

jne pauseit

mov dh , 11

mov dl , '\_'

cmp word [es:18\*160+28], dx

jne pauseit

mov dl , '\_'

cmp word [es:18\*160+20], dx

jne pauseit

jmp success

pauseit:

call crashsound

call gameover

success:

add word [score] , 1

call cheeringmessage

mov al, 0x20

out 0x20, al ; end of interrupt

popa

iret

gameover:

call clrscr2

mov ah, 0x13 ; service 13 - print string

mov al, 1 ; subservice 01 – update cursor

mov bh, 0 ; output on page 0

mov bl, 15 ; normal attrib

mov dx, 0x056f ; row 10 column 3

mov cx, 9 ; length of string

push cs

pop es ; segment of string

mov bp, message2 ; offset of string

int 0x10

mov ah, 0x13 ; service 13 - print string

mov al, 1 ; subservice 01 – update cursor

mov bh, 0 ; output on page 0

mov bl, 15 ; normal attrib

mov dx, 0x0A1F ; row 10 column 3

mov cx, 20 ; length of string

push cs

pop es ; segment of string

mov bp, message3 ; offset of string

int 0x10

mov word [over], 1

push word [score]

call printnum

mov ah, 0x13 ; service 13 - print string

mov al, 1 ; subservice 01 – update cursor

mov bh, 0 ; output on page 0

mov bl, 15 ; normal attrib

mov dx, 0x081b ; row 10 column 3

mov cx, 33 ; length of string

push cs

pop es ; segment of string

mov bp, app5 ; offset of string

int 0x10

xor ax , ax

mov es , ax

mov ax , [cs:oldtim]

mov word [es:8\*4] , ax

mov ax , [cs:oldtim + 2]

mov word [es:8\*4+2] , ax

xor ax , ax

mov es , ax

mov ax , [cs:oldkb]

mov word [es:9\*4] , ax

mov ax , [cs:oldkb + 2]

mov word [es:9\*4+2] , ax

ret

cheeringmessage:

cmp word [over] , 1

je near noth

cmp word [score] , 500

jb case2

mov ah, 0x13 ; service 13 - print string

mov al, 1 ; subservice 01 – update cursor

mov bh, 0 ; output on page 0

mov bl, 15 ; normal attrib

mov dx, 0x0A15 ; row 10 column 3

mov cx, 33 ; length of string

push cs

pop es ; segment of string

mov bp, app4 ; offset of string

int 0x10

cmp word [cs: soundflag3], 1

jne near noth

call HurdleCrossSound

mov word [cs: soundflag3], 0

jmp noth

case2:

cmp word [score] , 300

jb case3

mov ah, 0x13 ; service 13 - print string

mov al, 1 ; subservice 01 – update cursor

mov bh, 0 ; output on page 0

mov bl, 15 ; normal attrib

mov dx, 0x0A15 ; row 10 column 3

mov cx, 33 ; length of string

push cs

pop es ; segment of string

mov bp, app3 ; offset of string

int 0x10

cmp word [cs: soundflag2], 1

jne noth

call HurdleCrossSound

mov word [cs: soundflag2], 0

jmp noth

case3:

cmp word [score] , 200

jb case4

mov ah, 0x13 ; service 13 - print string

mov al, 1 ; subservice 01 – update cursor

mov bh, 0 ; output on page 0

mov bl, 15 ; normal attrib

mov dx, 0x0A15 ; row 10 column 3

mov cx, 33 ; length of string

push cs

pop es ; segment of string

mov bp, app2 ; offset of string

int 0x10

jne noth

cmp word [cs: soundflag1], 1

jne noth

call HurdleCrossSound

mov word [cs: soundflag1], 0

jmp noth

case4:

cmp word [score] , 100

jb noth

mov ah, 0x13 ; service 13 - print string

mov al, 1 ; subservice 01 – update cursor

mov bh, 0 ; output on page 0

mov bl, 15 ; normal attrib

mov dx, 0x0A15 ; row 10 column 3

mov cx, 33 ; length of string

push cs

pop es ; segment of string

mov bp, app1 ; offset of string

int 0x10

cmp word [cs: soundflag], 1

jne noth

call HurdleCrossSound

mov word [cs: soundflag], 0

noth:

ret

dinasourHandler:

pusha

in al, 0x60 ; read char from keyboard port

cmp al, 0x39 ; is Space pressed.

jne exit

mov word [cs:jmpFlag], 1 ; iterations on which the character should stay up.

call jmpsound

popa

jmp far [cs:oldkb]

exit:

mov al, 0x20

out 0x20, al ; send EOI to PIC

popa

iret

jmpsound:

push ax

mov al, 182 ; Prepare the speaker for the

out 43h, al ; note.

mov ax, 3000 ; Frequency number (in decimal)

; for middle C.

out 42h, al ; Output low byte.

mov al, ah ; Output high byte.

out 42h, al

in al, 61h ; Turn on note (get value from

; port 61h).

or al, 00000011b ; Set bits 1 and 0.

out 61h, al ; Send new value.

mov bx, 1 ; Pause for duration of note.

pause1:

mov cx, 65535

pause2:

dec cx

jne pause2

dec bx

jne pause1

in al, 61h ; Turn off note (get value from

; port 61h).

and al, 11111100b ; Reset bits 1 and 0.

out 61h, al ; Send new value.

pop ax

ret

crashsound:

push ax

mov al, 182 ; Prepare the speaker for the

out 43h, al ; note.

mov ax, 5000 ; Frequency number (in decimal)

; for middle C.

out 42h, al ; Output low byte.

mov al, ah ; Output high byte.

out 42h, al

in al, 61h ; Turn on note (get value from

; port 61h).

or al, 00000011b ; Set bits 1 and 0.

out 61h, al ; Send new value.

mov bx, 2 ; Pause for duration of note.

cpause1:

mov cx, 65535

cpause2:

dec cx

jne cpause2

dec bx

jne cpause1

in al, 61h ; Turn off note (get value from

; port 61h).

and al, 11111100b ; Reset bits 1 and 0.

out 61h, al ; Send new value.

pop ax

ret

HurdleCrossSound:

push ax

mov ax, 0

mov al, 182 ; Prepare the speaker for the

out 43h, al ; note.

mov ax, 2711 ; Frequency number (in decimal)

; A note.

out 42h, al ; Output low byte.

mov al, ah ; Output high byte.

out 42h, al

in al, 61h ; Turn on note (get value from

; port 61h).

or al, 00000011b ; Set bits 1 and 0.

out 61h, al ; Send new value.

mov bx, 1 ; Pause for duration of note.

hpause1:

mov cx, 65535

hpause2:

dec cx

jne hpause2

dec bx

jne hpause1

in al, 61h ; Turn off note (get value from

; port 61h).

and al, 11111100b ; Reset bits 1 and 0.

out 61h, al ; Send new value.

; Note E

mov al, 182 ; Prepare the speaker for the

out 43h, al ; note.

mov ax, 1809 ; Frequency number (in decimal)

; E note.

out 42h, al ; Output low byte.

mov al, ah ; Output high byte.

out 42h, al

in al, 61h ; Turn on note (get value from

; port 61h).

or al, 00000011b ; Set bits 1 and 0.

out 61h, al ; Send new value.

mov bx, 1 ; Pause for duration of note.

hpause3:

mov cx, 65535

hpause4:

dec cx

jne hpause4

dec bx

jne hpause3

in al, 61h ; Turn off note (get value from

; port 61h).

and al, 11111100b ; Reset bits 1 and 0.

out 61h, al ; Send new value.

noSound:

pop ax

ret

start:

mov bx, 0

xor ax, ax

mov es, ax ; point es to IVT base

mov ax, [es:9\*4]

mov [oldkb], ax ; save offset of old routine

mov ax, [es:9\*4+2]

mov [oldkb+2], ax ; save segment of old routine

mov ax, [es:8\*4]

mov [cs:oldtim], ax ; save offset of old routine

mov ax, [es:8\*4+2]

mov [cs:oldtim+2], ax ; save segment of old routine

cli ; disable interrupts

mov word [es:9\*4], dinasourHandler ; store offset at n\*4

mov [es:9\*4+2], cs ; store segment at n\*4+2

mov word [es:8\*4], cactusMover ; store offset at n\*4

mov [es:8\*4+2], cs

sti

terminateit:

mov dx, start ; end of resident portion

add dx, 15 ; round up to next para

mov cl, 4

shr dx, cl ; number of paras

mov ax, 0x3100 ; terminate and stay resident

int 0x21

app1: db 'Great !!! '

app2: db 'You Awesome !!! Keep it going '

app3: db 'You are the B-E-S-T '

app4: db 'V-I-C-T-O-R-Y. V-I-C-T-O-R-Y !!!'

app5: db 'Alas! YOu FaiL YoU LosE '