國 立 台 灣 科 技 大 學

電 機 工 程 系

微算機概論實習

(EE3802303)

Final Project

班級：　　　電機二甲

學號：B11007012 B11030232

指導老師：　王乃堅

姓名：　黃子恩、孫文浩

中華民國　　110　　年　　12　　月

(一) 程式使用方法及功能:

1. 遊戲規則:

這個遊戲是類似手機遊戲的別踩白塊兒的概念去撰寫的，主要分成以下幾個頁面:

I. 起始主畫面

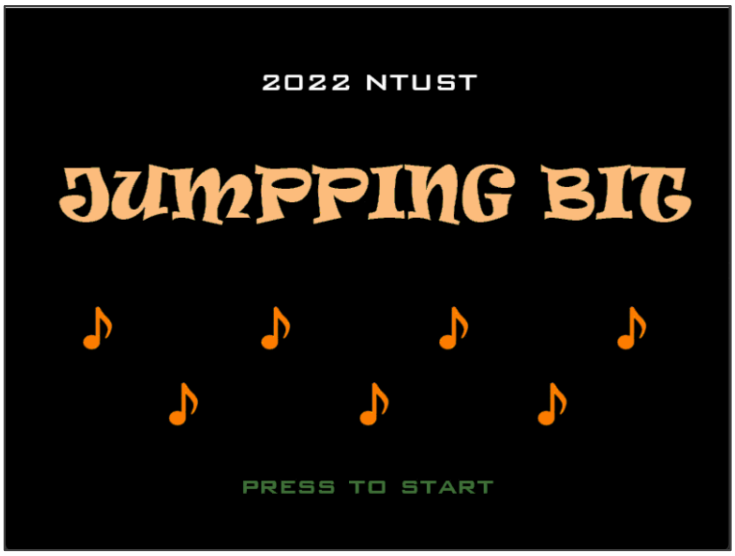


圖 1：起始主畫面

起始畫面有遊戲名稱 JUMMPING BIT，每秒會閃爍變換顏色 5 次，按下空白鍵後遊戲開始執行

II. 遊戲執行畫面

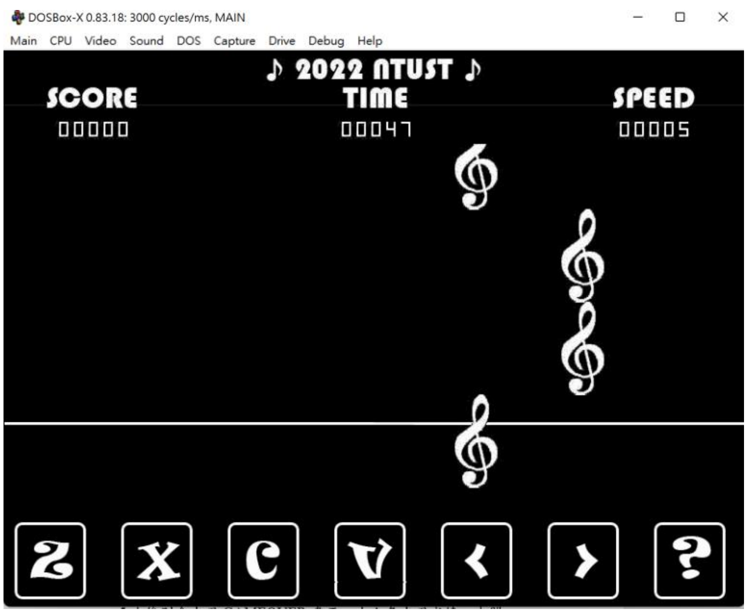
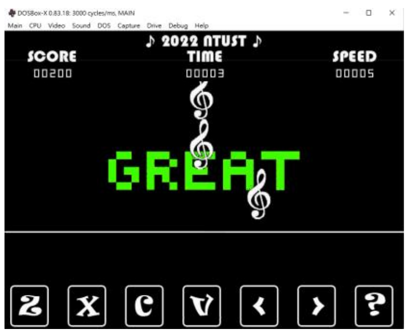
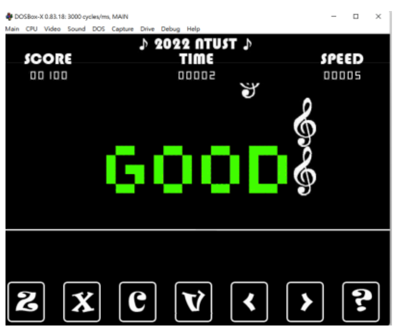

DOSBox-X 0.83.18: 3000 cycles/ms, MAIN
Main CPU Video Sound DOS Capture Drive Debug Help
♪ 2022 NTUST ♪
SCORE
00000
TIME
00047
SPEED
00005
Z X C V < > ?

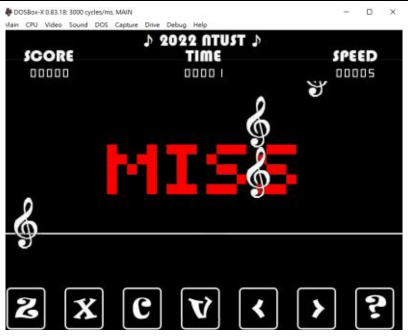
圖 2：遊戲執行畫面

- 分數欄位:打擊音符獲得分數，GOOD+100，GREAT+110

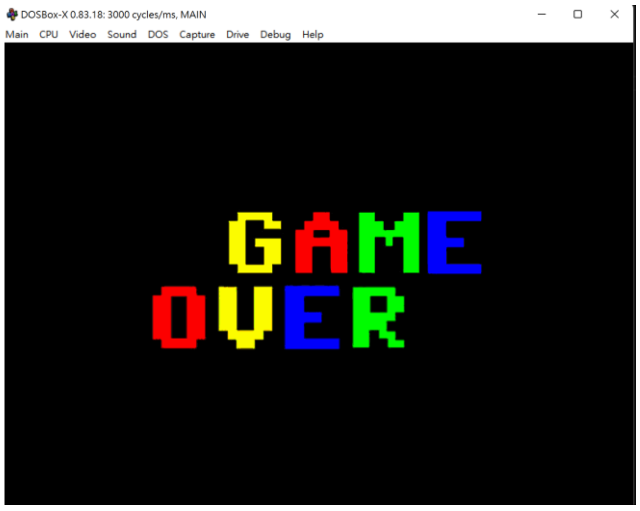
- 時間欄位:遊戲開始後經過時間
- 速度欄位:顯示目前速度，經過一段時間速度會增加
- 主要打擊區:
  玩家需要在音符觸碰到打擊線時，按下音符對應的按鈕”ZXCV<>?”，分別會發出 Do,Re,Mi,Fa,So,La,Si 七個音，若精準打擊會出現 GREAT 在背景(圖 3)，若有擊中但稍有偏差則出現 GOOD 在背景(圖 4)，若沒打擊則會出現紅色 MISS(圖 5)，當失誤 5 次後則判定遊戲結束，出現 GAMEOVER 畫面


DOSBox-X 0.83.18: 3000 cycles/ms, MAIN
Main CPU Video Sound DOS Capture Drive Debug Help
♪ 2022 NTUST ♪
SCORE
00200
TIME
00003
SPEED
00005
GREAT
Z X C V < > ?
圖 3 GREAT 畫面


DOSBox-X 0.83.18: 3000 cycles/ms, MAIN
Main CPU Video Sound DOS Capture Drive Debug Help
♪ 2022 NTUST ♪
SCORE
00100
TIME
00002
SPEED
00005
GOOD
Z X C V < > ?
圖 4 GOOD 畫面


DOSBox-X 0.83.18: 3000 cycles/ms, MAIN
Main CPU Video Sound DOS Capture Drive Debug Help
♪ 2022 NTUST ♪
SCORE
00000
TIME
00001
SPEED
00005
MISS
Z X C V < > ?
圖 5 MISS 畫面

III. 遊戲結束畫面


DOSBox-X 0.83.18: 3000 cycles/ms, MAIN
Main CPU Video Sound DOS Capture Drive Debug Help
GAME
OVER
圖 6 GAME OVER 動畫

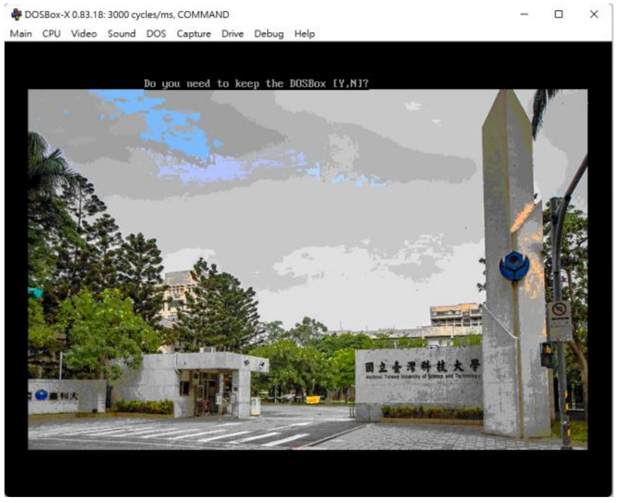

DOSBox-X 0.83.18: 3000 cycles/ms, COMMAND
Main CPU Video Sound DOS Capture Drive Debug Help
Do you need to keep the DOSBox [Y,N]?

圖 7 遊戲結束畫面

遊戲結束後，顯示 GAMEOVER 動畫讓畫面變得更生動，再按任意鍵會顯示 256 色 800*600 台科大校門畫面。

2. 每秒 60 幀畫面顯示實踐:

為了要讓遊戲畫面順暢顯示，以及固定音符下落速度等等原因，我們想要將畫面刷新率固定在 60 幀，要呈現 60 幀畫面的話，需要 $\frac{1}{60} = 0.016(s)$ 刷新一次，若使用 int 21h、2Ch 取得系統時間的話，精度大約只有 55ms，無法達成 60Hz 的需求，因此，我們使用 PC 裏面的計時晶片 8254。

主機板上產生 8254 時脈的震盪器頻率為 1193180 Hz，因此，若經過 0.016 秒的間隔，大約會經過$\frac{1193180}{60} = 19886.3$ 個 clock，每個 clock 計數暫存器會遞減 1，我們可以透過讀取 40h port 獲得計數器數值，並與上一幀更新時的數值相比，判斷是否需要更新，來達到穩定 60 幀的目的

3. 顯示畫面:

我們使用 800*600 的顯示畫面，由於在 real mode 下每一個 segment 只有 64k 可以使用，因此我們使用 ah=4f05 int 10h 中斷功能讓顯示器內存映射到 CPU 的地址空間，使我們可以有足夠的空間將像素點畫在顯示器的每一個點的位置達成顯示效果。

4. 聲音:

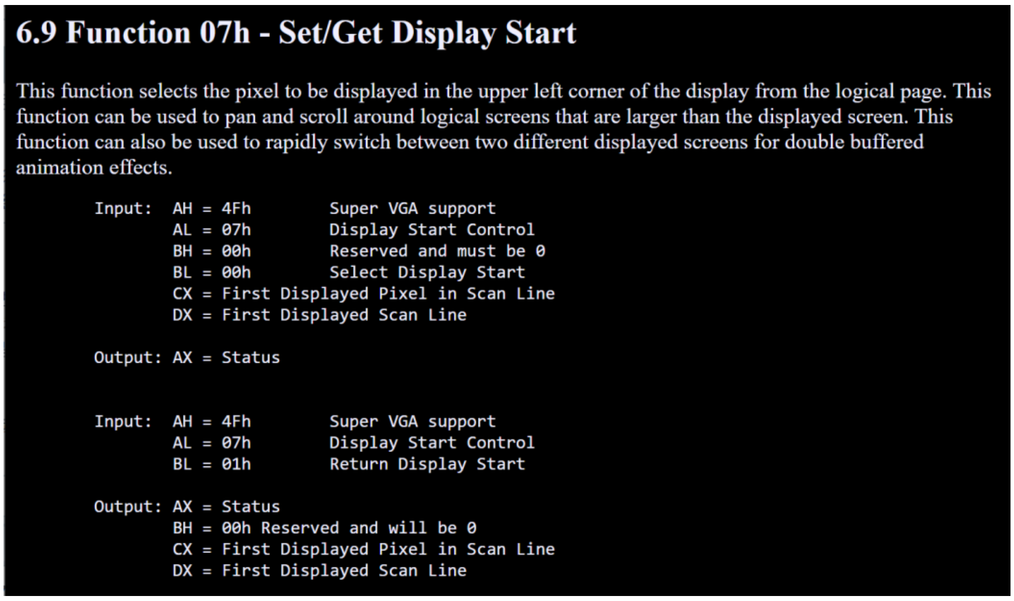
當敲擊螢幕上相對應的按鍵，會發出 Do,Re,Mi,Fa,So,La,Si 的聲音，聲音相同是使用 8254 IC 控制發出聲響的頻率，例如 Do 頻率為 262Hz 因此 1193810/262= 4556，我們要在一秒內發出 4556 個 CLOCK 的聲音才能完成 DO 的聲音，其他聲音以此類推

5. Double Buffer 螢幕更新不閃屏:

由於一畫面是由許多物件所組成，每一幀在清除、繪畫每個物件的過程中，若是直接對顯示記憶體做操作，使用者容易感受到不連貫、閃爍等等感覺。

Double Buffer 是指程式在繪圖時先對 Buffer 中內容進行操作，待整個畫面繪圖完畢後，再將 Buffer 中的內容搬移到顯示記憶體中一次顯示出來，藉此消除閃爍現象。

一張照片需要 480k 的顯示內存，但在 RealMode 下 DataSegment 只能到 64k，為了方便我們使用的方式是將顯示記憶體 page 分成兩個部分，page1~8 作為 A 畫面，page10~18 作為 B 畫面，再透過 int 10h ,ax=4f07h 切換顯示起始位置，顯示 A 畫面時在 B 畫面繪圖，顯示 B 畫面時在 A 畫面繪圖，藉此達成兩頁面切換的效果。



6.9 Function 07h - Set/Get Display Start

This function selects the pixel to be displayed in the upper left corner of the display from the logical page. This function can be used to pan and scroll around logical screens that are larger than the displayed screen. This function can also be used to rapidly switch between two different displayed screens for double buffered animation effects.

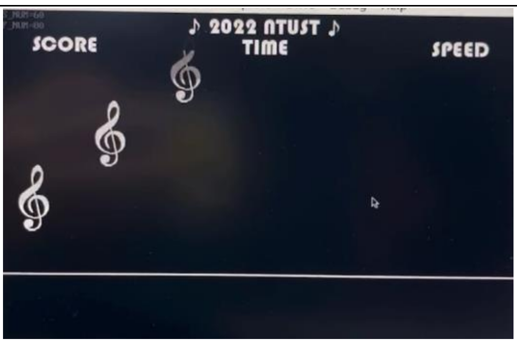
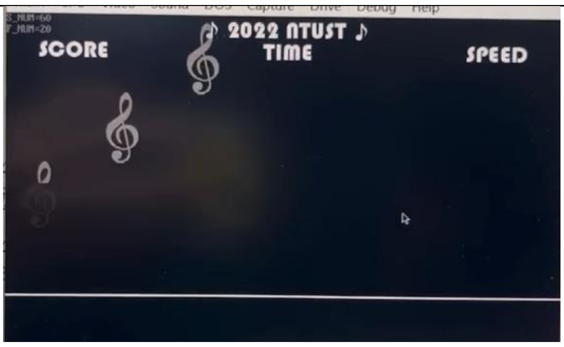
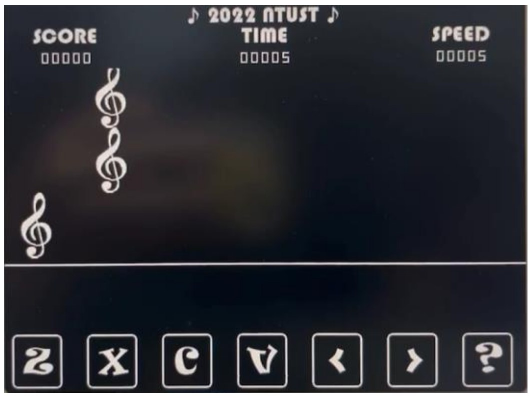
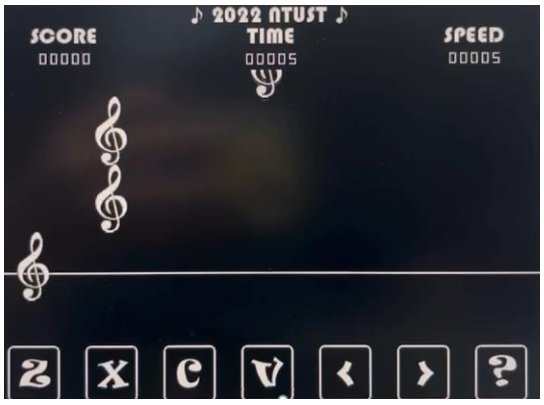
```
Input:  AH = 4Fh        Super VGA support
        AL = 07h        Display Start Control
        BH = 00h        Reserved and must be 0
        BL = 00h        Select Display Start
        CX = First Displayed Pixel in Scan Line
        DX = First Displayed Scan Line

Output: AX = Status


Input:  AH = 4Fh        Super VGA support
        AL = 07h        Display Start Control
        BL = 01h        Return Display Start

Output: AX = Status
        BH = 00h Reserved and will be 0
        CX = First Displayed Pixel in Scan Line
        DX = First Displayed Scan Line
```

圖 8 Function 07h 功能

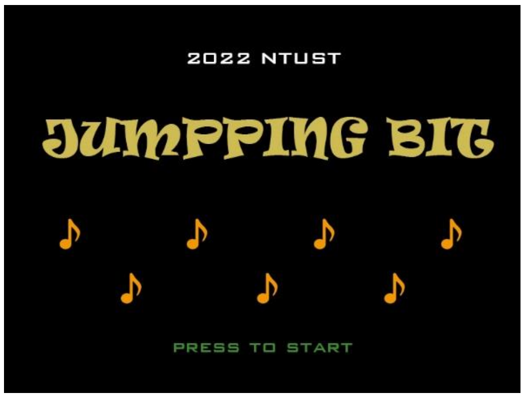
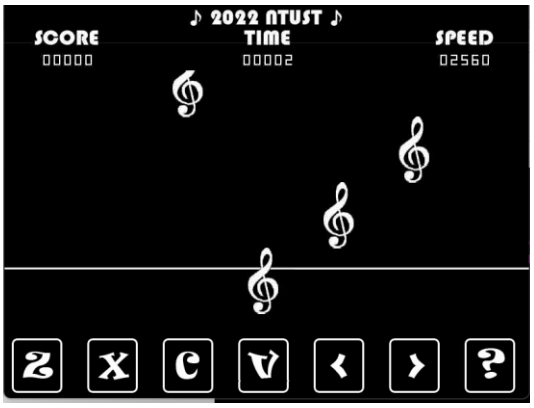
 圖 9	 圖 10
無使用 Double buffer	無使用 Double buffer
 ♪ 2022 ATUST ♪ SCORE 00000 TIME 00005 SPEED 00005 Z X C V < > ? 圖 11	 ♪ 2022 ATUST ♪ SCORE 00000 TIME 00005 SPEED 00005 Z X C V < > ? 圖 12
使用 Double buffer	使用 Double buffer

6. 讀取圖片檔案:

除了顯示顯示器內存不夠，我們也遇到了如何將多張照片顯示在螢幕上的問題，雖然圖片夠小我們就能將以技巧性的方式擺脫 64k 的空，但我們有遠大的夢想想要顯示 800*600 的完整照片，因此我們將 jpg 圖片，重新編碼儲存在.dnj 的檔案格式內，要顯示時再從.dnj 的資料中提取各個像素點的參數加以顯示，改善我們單一程式的大小，也方便修改圖片，並且增加使用者的可讀性。(以下是 800*600*256 色的圖片顯示)

 圖 13	 圖 14
256 色彩度不夠，因此會有看起來顏色不對的問題但實屬正常	

(二) 程式展示畫面:

 圖 15	 ♪ 2022 NTUST ♪ SCORE 00000 TIME 00002 SPEED 02560 Z X C V < > ? 圖 16
主畫面頁	遊戲介面
 圖 17	 圖 18
遊戲結束畫面	

(三) 分工表:

影像處理、讀檔、聲音	黃子恩
流程 DoubleBuffer 60hz 更新率	孫文浩

(四) 心得:

(黃子恩):

上了正課了解組合語言的概念，讓我們在寫組語的變得比較輕鬆，這次的期末專題對我們來說非常具有挑戰性，因為想要用組語寫出紅白機的賽車遊戲，因此我們找了很多的資料，例如:如何發出聲音、如何讀檔、如何顯示，如何更新不閃屏，如何創造 60hz，如何操作 800*600SVGA 的顯示器，讓我們發現原來寫組語並不是單純只是寫軟體而已，必須了解所有硬體上的記憶體分配及裝置操作的原理，寫組語才會變得輕鬆。找了這麼多的資料，紅白機賽車遊戲約完成 80%但只剩下三天的時間可以完成本次專題，因此急轉直下，先臨時換遊戲，將我們所有做出來的功能放在別踩白塊而當中，雖然有點可惜賽車遊戲沒辦法即時做完，但我們在尋找操作遊戲當中確實學到非常多，希望期末考完試來完成紅白機遊戲。

(孫文浩):

平常在其他課程學的都是高階語言，經由這門課程接觸組合語言之後才發現，平常在其他語言中習以為常的操作，原來背後要做的事情這麼多，這次期末報告選擇的是小遊戲，真正開始做了之後才知道會遇到很多問題，例如螢幕的閃爍，幀數的控制，甚至到 delay 這些功能，平常都是能用內建的功能解決，但在組語上都必須自己實作，況且要做出這些功能不能只了解軟體邏輯，還必須了解作業系統的操作，甚至是控制硬體等等，雖然比起高階語言寫起來很麻煩很辛苦，但在做完期末報告後，我確實在軟體、作業系統、記憶體存取，甚至是電腦的硬體等等方面，都有很大的進步。

(五) 程式碼:

```
GetChar macro char
    mov ah,07h
    int 21h
    mov char,al
    endm

GetChar06h macro char
    mov ah,06h
    mov dl,0ffh
    int 21h
    mov char,al
    endm
PrintStr macro string
    mov ah,09h
    mov dx,offset string
    int 21h
    endm
SetScreen macro
    mov ax,@data
    mov ds,ax
    mov es,ax
    mov ax,4f01h
    mov cx,103h
    lea di,vesa_info
    int 10h

    mov ax,0A000h
    mov es,ax
    mov ax,4f02h
    mov bx,103h
    int 10h
endm

set_Background macro color
    Local       store_process
     mov ax,@data
    mov ds,ax
    cld
    mov   ax, 0A000h
    mov   es, ax
    mov   dx, 0


    store_process:
    mov   cx, 0ffffh
    mov   ax, 4f05h
    mov   bx, 0
    int   10h
    mov   al, color
    mov   di, 0
    rep   stosb
    mov   es:[di],color
    inc   dx
    cmp   dx, 17
    jle   store_process
endm

Read_Time macro
mov ah,2ch
int 21h
endm

SET_CUR macro row1,col1
Local buffer0,buffer1,_end
    mov dh,row1
    mov dl,col1
    mov bx,0000h
    ;for double buffer
    cmp now_buffer_index,1
    je buffer0
    cmp now_buffer_index,0
    je buffer1

    buffer0: ;from page 0
    ; mov bh,10
    jmp _end

    buffer1:  ;from page 10

    _end:
    mov ah,02h
    int 10h
    endm

Set_file_pointer macro f4b,b4b
mov ah,42h                    ;MOVE FILE READ/WRITE POINTER (LSEEK)
mov al,0                      ;AL = method value
                             ;0 = offset from beginning of file
                             ;1 = offset from present location
                             ;2 = offset from end of file
mov bx,file_handle            ;bx :file_handle
mov cx,f4b                    ;cx:dx offset in byte
mov dx,b4b
int 21h
endm

store_no_yellow macro
local Lop1,next
Lop1:
    mov al,ds:[si]
    cmp al,0h
    je next
    mov byte ptr es:[di],al
    next:
    inc si
    inc di
    loop Lop1
endm
clear_all_sign macro
    Local clean,do_not
    mov cx ,sign_num
    clean:
        push cx
        mov ax,4
        mul cx
        mov si ,ax
        mov recover_pic_x,100
        mov ax,speed
        mov recover_pic_y,20

        mov bx ,block_pos[si][-2]
        mov recover_pic_startx,bx
        mov bx ,block_pos[si][-4]
        cmp bx,100
        jbe do_not
        sub bx,speed
        sub bx,speed
        mov recover_pic_starty,bx

        call Print_clean

        do_not:
        pop cx
    loop clean

endm
clear_page macro
    mov recover_pic_x,800
    mov recover_pic_y,400

    mov recover_pic_startx,0
    mov recover_pic_starty,100

    lea ax,pic_clean
    mov pic_address,ax
    call Print_pic
endm
draw_all_sign macro
Local draw,_continue

        mov cx ,4
        lea ax,pic_sign
        mov pic_address,ax
        draw:
            push cx
            mov ax,2
            mul cx
            mov si,ax

            mov recover_pic_x,100
            mov recover_pic_y,100

            mov bx,blockX[si][-2]
            mov recover_pic_startx,bx
            mov bx,blockY[si][-2]
            mov recover_pic_starty,bx
            cmp recover_pic_starty,0
            je _continue
            call Print_No_Yellow
            _continue:
            pop cx
        loop draw

endm

update_sign_pos macro
Local update,continue,set,hit

        mov cx,4
        update:
            push cx
            mov ax,@data
            mov ds,ax

            ; mov di,cx
            ; shl di,1
            ; lea bx,hit_index
            ; mov ax,[bx+di-2]
            ; cmp ax,1
            ; je set


            mov ax,2
            mul cx
            mov si,ax
            lea bx,blockY
            mov ax,[bx+si-2]
            cmp ax,0
            je continue

            add ax,speed
            mov word ptr blockY[si][-2],ax
            cmp ax,400
            jb continue
            ; cmp ax,400
            ; jbe hit
            cmp ax,400
            jae set
            ; hit:
            ;     mov si,cx
            ;     lea bx,can_hit_index
            ;     mov byte ptr [bx+si-1], 1
            ;     jmp continue
            set:

            mov di,cx
            shl di,1
            lea bx,hit_index
            mov word ptr [bx+di-2],0

            mov ax,now_lower_index
            inc ax
            mov dx,0
            mov bx,4
            div bx
            mov now_lower_index,dx
            mov word ptr BlockY[si][-2],0
```

```
            mov hit_alert_flag,3
            mov hit_alert_counter, 10
            mov ax,life
            dec ax
            mov life,ax
            .if life==0
                mov flow_flag,2
            .endif
            ; mov di,cx
            ; lea bx,can_hit_index
            ; mov byte ptr [bx+di-1],0 ;設定沒打到方塊
            ; lea bx,hit_index
            ; mov byte ptr [bx+di-1],0
            continue:
            pop cx
        loop update

endm
draw_score macro
local L1
    mov ax,@data
    mov ds,ax
    mov ax,score
    lea di,number_buffer
    mov x_shift,55
    mov recover_pic_starty,75
    call tran_to_num_pic
    mov cx,5
    L1:
    lea bx,number_buffer
    mov di,cx
    mov byte ptr [bx+di-1],30h
    loop L1

endm
draw_speed macro
local L1
    mov ax,speed
    lea di,number_buffer
    mov x_shift,660
    mov recover_pic_starty,75
    call tran_to_num_pic
    mov cx,5
    L1:
    lea bx,number_buffer
    mov di,cx
    mov byte ptr [bx+di-1],30h
    loop L1

endm

draw_time macro
local L1
    mov ax,sec_pic
    lea di,number_buffer
    mov x_shift,360
    mov recover_pic_starty,75
    call tran_to_num_pic
    mov cx,5
    L1:
    lea bx,number_buffer
    mov di,cx
    mov byte ptr [bx+di-1],30h
    loop L1
endm

draw_panel macro
    lea ax,pic_panel
    mov pic_address,ax
    mov recover_pic_x,800
    mov recover_pic_y,100
    mov recover_pic_startx,0
    mov recover_pic_starty,0
    call Print_pic
endm
```

```
print_frame_num macro
mov ax,cnt
mov di,offset fps_num
call tran


SET_CUR 1,0
PrintStr fps_num
endm

print_sec macro
    mov bx , now_lower_index
    mov ax,2
    mul bx
    mov si , ax
    mov ax,vala
    ; mov ax,blockX[si]
    ; mov ax,sec_pic
    mov di,offset sec_num
    call tran

    SET_CUR 8,44
    PrintStr sec_num
endm

draw_menu macro n
Local n_1,n_2,_draw,_draw
    cmp n,0
    je n_1
    cmp n,1
    je n_2
    n_1:
        lea ax,pic_menu_1
        jmp _draw
    n_2:
        lea ax,pic_menu_2
        jmp _draw
    _draw:
        mov pic_address,ax
        mov recover_pic_x,800
        mov recover_pic_y,600
        mov recover_pic_startx,0
        mov recover_pic_starty,0
        call Print_pic

endm

hit_note_or_not macro
Local L1,continue
    mov cx,4
    L1:
        push cx

        mov ax,@data
        mov ds,ax
        mov es,ax

        mov ax,cx
        shl ax,2
        mov si,ax
        lea bx,hit_index
        cmp word ptr [bx+di-2],0
        je continue
        mov word ptr BlockY[si][-2],0
        mov ax,temp
        mov di,ax
        lea bx,can_hit_index
        mov byte ptr [bx+di-1],0
        lea bx,hit_index
        mov byte ptr [bx+di-1],0
        continue:
        pop cx
    loop L1
endm
music   macro sound

    mov al,0B6h
    out 43h,al
    mov bx,offset sounds
    mov si,sound
    shl si,1
    mov ax,[bx+si]
    ;mov ax,4063
    out 42h,al
    mov al,ah
    out 42h,al
    in  al,61h
    or  al,3
    out 61h,al
    mov ah,0

    endm
MusicEnd macro
    in  al,61H
    and al,0FCH
    out 61H,al
    endm

draw_hit_state macro
Local end
    cmp hit_alert_flag,0
    je end
    cmp hit_alert_counter,0
    jbe end
        .if hit_alert_flag==1
        lea ax,pic_good
        .elseif hit_alert_flag==2
        lea ax,pic_ok
        .elseif hit_alert_flag==3
        lea ax,pic_miss
        .endif
        mov pic_address,ax

        mov recover_pic_x,400
        mov recover_pic_y,150
        mov recover_pic_startx,200
        mov recover_pic_starty,200
        call Print_pic
    end:
endm

.model large
.386
.stack 1024
.data
file_in                db 800 dup(?)
page1_1                db 3 dup(?)


pic_sign               db "sign.dnj",0 ;100,100
pic_panel              db "dash.dnj",0 ;800,200
pic_menu_1             db "1page.dnj",0 ;800,600
pic_menu_2             db "2page.dnj",0 ;800,600
pic_clean              db "clean.dnj",0 ;800,600
pic_sz                 db "smallz.dnj",0 ;100,100
pic_sx                 db "smallx.dnj",0 ;100,100
pic_sc                 db "smallc.dnj",0 ;100,100
pic_sv                 db "smallv.dnj",0 ;100,100
pic_sl                 db "smalll.dnj",0 ;100,100
pic_sr                 db "smallr.dnj",0 ;100,100
pic_sq                 db "smallq.dnj",0 ;100,100
pic_bz                 db "bigz.dnj",0 ;100,100
pic_bx                 db "bigx.dnj",0 ;100,100
pic_bc                 db "bigc.dnj",0 ;100,100
pic_bv                 db "bigv.dnj",0 ;100,100
pic_bl                 db "bigl.dnj",0 ;100,100
pic_br                 db "bigr.dnj",0 ;100,100
pic_bq                 db "bigq.dnj",0 ;100,100
pic_great              db "great.dnj",0 ;60,60
pic_0                  db "0.dnj",0 ;20,20
pic_1                  db "1.dnj",0 ;20,20
pic_2                  db "2.dnj",0 ;20,20
pic_3                  db "3.dnj",0 ;20,20
pic_4                  db "4.dnj",0 ;20,20
```

```
pic_5              db "5.dnj",0 ;20,20
pic_6              db "6.dnj",0 ;20,20
pic_7              db "7.dnj",0 ;20,20
pic_8              db "8.dnj",0 ;20,20
pic_9              db "9.dnj",0 ;20,20


pic_miss              db "miss.dnj",0 ;20,20
pic_good              db "good.dnj",0 ;20,20
pic_ok              db "ok.dnj",0 ;20,20
pic_school              db "school.dnj",0 ;20,20

pic_address         dw ?
file_handle         dw ?
vesa_info           db 256 dup(?)
fail_open           db " ",'$'
file_f16b           dw 0
file_b16b           dw 0
screen_row          dw 0
futurefile_pointer  dd 0
byte_read_write     dw 800
recover_pic_x       dw 0
recover_pic_y       dw 0
recover_pic_startx  dw 0
recover_pic_starty  dw 0
pageOffset          dd 0
read_file_count     dd 0
PicPage             dw 0
PicOffset           dw 0
PicFirstcnt         dw 0
nowstar             dw 310
level               dw 0
GameTimeIndex       db 45,40,30
GameTime            db 0
gameStartMin        db 0
gameStartSec        db 0
gameStarthSec       db 0
gameCurMin          db 0
gameCurSec          db 0
```

```
gameCurhSec          db 0
TIME                 db 0
TIMES                db 4 dup(' '),'$'
StartTime            db 0
str_buffer           db 10 dup(?),'$'
racebg_shift_x       dw 0
racebg_shift_y       dw 0
shift_cnt            dw 0
pre_racebg_shift     dw 0
sign_num dw 7
block_posx dw  0,115,230,345,460,575,690
blockX dw  4 dup(0)
blockY dw  4 dup(0)
sounds          dw
0ffffh,4560,4063,3619,3416,3043,2711,2415,2280,2031,1809,1715,1521,1355
,1207
speed dw 5
fps_num db "F_NUM=",10 dup(' '),'$'
sec_num db " ",10 dup(' '),'$'
time1   dw      ?,?,?,0
time2   dw      65535,0,0,0
cnt dw 0

sec_pic dw 0
pic_over             db "OVER.dnj",0 ;600,800
pic_game             db "GAME.dnj",0 ;600,800
shift   dw      0
cnt_gg dw 0
game_start dw 0
over_start dw 1000
note_index_frame dw 0
; song_seq dw
0,1,1,3,4,5,2,5,2,5,1,5,255,1,2,3,3,2,1,0,0,1,2,3,3,2,1,0,255;0,1,2,3,4
,5,6,0,1,2,3,4,5,6,255
; song_seq dw 1,1,2,1,4,3,1,1,2,1,5,4,1,1,6,4,3,2,7,7,6,4,5,4,255

speedup_interval dw 14
song_seq dw 0,0,4,4,5,5,4,3,3,2,2,1,1,0, 0,0,4,4,5,5,4,3,3,2,2,1,1,0
```

```
        dw  0,0,4,4,5,5,4,3,3,2,2,1,1,0, 0,0,4,4,5,5,4,3,3,2,2,1,1,0
        dw  0,0,4,4,5,5,4,3,3,2,2,1,1,0, 0,0,4,4,5,5,4,3,3,2,2,1,1,0
        dw  0,0,4,4,5,5,4,3,3,2,2,1,1,0, 0,0,4,4,5,5,4,3,3,2,2,1,1,0
        dw  0,0,4,4,5,5,4,3,3,2,2,1,1,0, 0,0,4,4,5,5,4,3,3,2,2,1,1,0
        dw  0,0,4,4,5,5,4,3,3,2,2,1,1,0, 0,0,4,4,5,5,4,3,3,2,2,1,1,0
        dw  255




note_index dw 0
gg_index_frame dw 0
menu_page_index dw 0
now_buffer_index dw 0
keyboard_show_cnt dw 0
can_hit_index dw 4 dup(0)
hit_index dw 4 dup(0)
key db 0
hit_a_sign db 0
temp dw 0
notecnt dw 0
xpos dw 0
now_lower_index dw 0
lowest_Xval dw 0
lowest_Yval dw 0
hit_area dw 0
hit dw 0
num dw 0
number_buffer db 5 dup(30h),'$'
x_shift dw 0
score dw 0
hit_alert_flag dw 0
hit_alert_counter dw 30
; vall dw 0
speed_val dw 0
speed_index dw 5,10,20,50,100
vala dw 0
life dw 6
;
```

```
flow_flag dw 0

.code
start:
;---------------------------------------------------
;---------------------------------------------------
;---------------------------------------------------
;---------------------------------------------------
;---------------------------------------------------
; init

SetScreen

mov  ax, @data
mov  es, ax
mov  ds, ax
main:

    mov  ax, @data
    mov  ds, ax
    mov  es, ax

    mov  AH , 4Fh
    mov  AL , 07h
    mov  BH , 00h
    mov  BL , 00h
    cmp now_buffer_index,0
    je buffer0
    cmp now_buffer_index,1
    je buffer1
    buffer0:
    mov  CX ,0
    mov  DX , 0
    jmp _b_set
    buffer1:
    mov  CX ,160
    mov  DX , 819
    _b_set:
```

```
    int 10h

    cmp flow_flag,0
    je menu_page
    cmp flow_flag,1
    je game_page
    cmp flow_flag,2
    je game_over
menu_page:

    ; music 0
    mov  ax, @data
    mov  ds, ax
    mov  es, ax

    mov dx,0
    mov ax,menu_page_index
    mov bx,2
    div bx
    mov menu_page_index,dx
    t1:
        mov  ax, @data
        mov  ds, ax
        mov  es, ax
        mov     di,offset time1
        call    get_time
        mov ax ,time2[0]
        sub ax ,time1[0]
        cmp ax ,19886 ;19886,39772
        jae _NEXT_FRAME_menu ;;>16ms
        jmp t1 ;;loop until 1/60s

    _NEXT_FRAME_menu:

        draw_menu menu_page_index
    draw_hit_state

        inc now_buffer_index
        mov dx,0
        mov ax,now_buffer_index
        mov bx,2
        div bx
        mov now_buffer_index,dx

        inc cnt
        cmp cnt ,12
        jbe _mNO
            mov cnt,0
            inc sec_pic
            mov ax,menu_page_index
            inc ax
            mov menu_page_index,ax
        _mNO:
    GetChar06h al
    .if al == 20h
        mov flow_flag,1
        ; mov now_buffer_index,0
        ; set_Background 00h
        ; mov now_buffer_index,1
        set_Background 00h
    .endif
    jmp main
game_page:


    clear_page
    draw_hit_state
    draw_all_sign
    draw_panel
    draw_score
    draw_time
    draw_speed
    ;print_sec

    GetChar06h al
    mov key,al
```

```
t:
    mov  ax, @data
    mov  ds, ax
    mov  es, ax
    mov  di,offset time1
    call   get_time
    mov ax ,time2[0]
    sub ax ,time1[0]
    cmp ax ,19886 ;19886,39772
    jae _NEXT_FRAME ;;>16ms
    jmp t ;;loop until 1/60s


_NEXT_FRAME:

    mov bx,time1[0]
    mov time2[0],bx
    MusicEnd
    update_sign_pos
    .if hit_alert_counter>0
        mov ax,hit_alert_counter
        sub ax,1
        mov hit_alert_counter,ax
    .endif
    call sound



    inc note_index_frame
    inc cnt
    cmp cnt ,60
    jbe _next_note
        mov cnt,0
        inc sec_pic
    _next_note:
    mov ax,100 ;note_pic_h
    mov dx,0
    mov bx,speed
    div bx
```

```
        cmp note_index_frame , ax
        jb _no
            mov ax,@data
            mov ds,ax
            mov ax,speedup_interval
            dec ax
            mov speedup_interval,ax
            .if ax==0

                mov ax,speed_val
                inc ax
                .if ax>=3
                mov ax,3
                .endif

                mov speed_val,ax
                mov si,ax
                shl si,1
                lea bx,speed_index
                mov dx,[bx+si]
                mov vala,dx
                mov speed,dx
                ;mov ax,speed
                ;add ax,dx
                ;mov speed, ax
                mov speedup_interval,14
            .endif
            mov note_index_frame,0
            mov ax, @data
            mov es,ax
            mov ds,ax
            mov ax,note_index
            shl ax,1
            mov di,ax
            lea bx,song_seq
            mov ax,[bx+di]
            cmp ax,255
            je _music_end
```

```
    mov di,ax
    shl di,1
    lea bx,block_posx
    mov ax,[bx+di]
    mov xpos,ax

    mov ax,note_index
    mov dx,0
    mov bx,4
    div bx
    mov si,dx
    shl si,1
    lea bx,blockX
    mov ax,xpos
    mov [bx+si],ax

    ; mov ax,now_lower_index
    ; mov dx,0
    ; mov bx,4
    ; div bx
    ; mov now_lower_index,dx

    lea bx,blockY
    mov ax,[bx+si]
    mov ax , 1
    ; inc ax
    mov [bx+si],ax
    inc note_index
    jmp _no
    _music_end:

        mov note_index,0
        ; mov now_lower_index,1
_no:
inc now_buffer_index
mov dx,0
mov ax,now_buffer_index
mov bx,2
```

```
        div bx
        mov now_buffer_index,dx




jmp main
game_over:
    mov now_buffer_index,1
    mov  AH , 4Fh
    mov  AL , 07h
    mov  BH , 00h
    mov  BL , 00h
    mov  CX ,0
    mov  DX , 0
    int 10h
    gg_t:
        MusicEnd
        mov  ax, @data
        mov  ds, ax
        mov  es, ax
        mov     di,offset time1
        call    get_time
        mov ax ,time2[0]
        sub ax ,time1[0]
        cmp ax ,19886 ;19886,39772
        jae _gg_NEXT_FRAME ;;>16ms
        jmp gg_t ;;loop until 1/60s

        _gg_NEXT_FRAME:
        mov bx,time1[0]
        mov time2[0],bx
        inc gg_index_frame
        inc cnt_gg

        cmp cnt_gg,50
        ja stop
        mov bx,cnt_gg
        shl bx,3
        mov game_start,bx
        mov ax,800
        sub ax,bx
        mov over_start,ax

        mov recover_pic_x,800
        mov recover_pic_y,300
        mov recover_pic_startx,0
        mov recover_pic_starty,0
        mov ax,game_start
        mov racebg_shift_x,ax
        mov racebg_shift_y,0
        lea ax,pic_game
        mov pic_address,ax
        call Print_racebackground

        mov recover_pic_x,800
        mov recover_pic_y,300
        mov recover_pic_startx,0
        mov recover_pic_starty,300
        mov ax,over_start
        mov racebg_shift_x,ax
        mov racebg_shift_y,0
        lea ax,pic_over
        mov pic_address,ax
        call Print_racebackground
        jmp gg_t
        stop:
    GetChar al
final:
    lea ax,pic_school
    mov pic_address,ax

    mov recover_pic_x,737
    mov recover_pic_y,474
    mov recover_pic_startx,31
    mov recover_pic_starty,63
    call Print_pic
```

```
GetChar al

mov ax,4c00h        ;exit dos
int 21h
;----------------------------------------------------
;----------------------------------------------------
;----------------------------------------------------
;----------------------------------------------------
;----------------------------------------------------
;----------------------------------------------------
;----------------------------------------------------
Print_pic proc
recover_pic:

    mov ax,@data
    mov ds,ax
    mov ah,3Dh                  ;Open file
    mov al,0                    ;0,R 1,W 2,R/W
    mov dx,pic_address
    int 21h

    pushf                       ;read flag val
    pop bx
    and bx,0001
    .if bx==0001                ;verify CF is set(error)
    PrintStr fail_open
    .endif
    mov file_handle,ax          ;ax return file_handle

    xor eax,eax
    xor edx,edx
    mov ax,recover_pic_starty
    mov bx,800
    mul bx
    xor ebx,ebx
    mov bx,dx
    shl ebx,16
    add ebx,eax
    xor eax,eax
    mov ax,recover_pic_startx
    add ebx,eax
    mov PageOffset,ebx


    mov file_f16b,0
    mov file_b16b,0
    mov screen_row,0
    row_print:
        Set_file_pointer file_f16b,file_b16b
        mov bx,file_handle
        mov cx,recover_pic_x                ; 140 Bytes to read
        lea dx,file_in
        mov ah,3fh
        int 21h

        mov ebx,PageOffset
        mov PicOffset,bx
        shr ebx,16
        mov PicPage,bx

        ;for double buffer
        cmp now_buffer_index,1
        je _rpic_buffer0
        cmp now_buffer_index,0
        je _rpic_buffer1

        _rpic_buffer0: ;from page 0
        jmp _RB_end
        _rpic_buffer1:  ;from page 10
        mov ax,PicPage
        add ax,10
        mov PicPage,ax

        _RB_end:

        mov ebx,PageOffset
        and ebx,0ffffh
```

```
xor eax,eax
mov ax,recover_pic_x
add ebx,eax
cmp ebx,0ffffh

ja  dotwice
    mov ax,0A000h
    mov es,ax
    mov dx,PicPage
    mov ax,4f05h
    mov bx,0
    int 10h
    cld
    mov cx,recover_pic_x
    mov ax,@data
    mov ds,ax
    mov ax,0A000h
    mov es,ax
    lea si,file_in
    mov di,PicOffset        ;ds:si-->es:di
    rep movsb
    jmp finish_store_data
dotwice:
    mov ax,0A000h
    mov es,ax
    mov dx,PicPage
    mov ax,4f05h
    mov bx,0
    int 10h

    cld
    mov cx,0ffffh
    mov dx,PicOffset
    sub cx,dx
    mov PicFirstcnt,cx
    mov ax,@data
    mov ds,ax
    mov ax,0A000h
```

```
        mov es,ax
        lea si,file_in
        mov di,PicOffset        ;ds:si-->es:di
        rep movsb

        inc PicPage
        mov ax,0A000h
        mov es,ax
        mov dx,PicPage
        mov ax,4f05h
        mov bx,0
        int 10h
        cld

        mov cx,recover_pic_x
        mov dx,PicFirstcnt
        sub cx,dx
        mov ax,@data
        mov ds,ax
        mov ax,0A000h
        mov es,ax
        lea bx,file_in
        add bx,PicFirstcnt
        mov si,bx
        mov di,0        ;ds:si-->es:di
        rep movsb
    finish_store_data:

    mov ebx,pageOffset
    xor edx,edx
    add ebx,800
    mov pageOffset,ebx

    xor ebx,ebx
    mov bx,file_f16b
    shl ebx,16
    mov dx,file_b16b
    add ebx,edx
```

```
        mov dx,recover_pic_x
        add ebx,edx
        mov file_b16b,bx
        shr ebx,16
        mov file_f16b,bx

        inc screen_row
        mov ax,recover_pic_y
        cmp screen_row,ax
    jle row_print
    mov ah, 3Eh                 ;close file
    mov bx, file_handle
    int 21h
    ret
Print_pic endp
;----------------------------------------------------
tran proc near
    mov cx,0
    Hex2Dec:
    inc cx
    mov bx,10
    mov dx,0
    div bx
    push dx
    cmp ax,0
    jne Hex2Dec
    Dec2Ascii:
    pop ax
    add al,30h
    mov [di+6],al
    inc di
    loop Dec2Ascii
    ret
tran endp
;----------------------------------------------------

;----------------------------------------------------
Print_No_Yellow proc
```

```
    mov ax,@data
    mov ds,ax
    mov ah,3Dh                  ;Open file
    mov al,0                    ;0,R 1,W 2,R/W
    mov dx,pic_address
    int 21h

    pushf                       ;read flag val
    pop bx
    and bx,0001
    .if bx==0001                ;verify CF is set(error)
    PrintStr fail_open
    .endif
    mov file_handle,ax          ;ax return file_handle

    xor eax,eax
    xor edx,edx
    mov ax,recover_pic_starty
    mov bx,800
    mul bx
    xor ebx,ebx
    mov bx,dx
    shl ebx,16
    add ebx,eax
    xor eax,eax
    mov ax,recover_pic_startx
    add ebx,eax
    mov PageOffset,ebx


    mov file_f16b,0
    mov file_b16b,0
    mov screen_row,0
    row1_print:
        Set_file_pointer file_f16b,file_b16b
        mov bx,file_handle
        mov cx,recover_pic_x            ; 140 Bytes to read
        lea dx,file_in
        mov ah,3fh
        int 21h

        mov ebx,PageOffset
        mov PicOffset,bx
        shr ebx,16
        mov PicPage,bx

        ;for double buffer
        cmp now_buffer_index,1
        je _pNYP_buffer0
        cmp now_buffer_index,0
        je _pNYP_buffer1

        _pNYP_buffer0: ;from page 0
        jmp _pn_end
        _pNYP_buffer1:  ;from page 10
        mov ax,PicPage
        add ax,10
        mov PicPage,ax

        _pn_end:

        mov ebx,PageOffset
        and ebx,0ffffh
        xor eax,eax
        mov ax,recover_pic_x
        add ebx,eax
        cmp ebx,0ffffh

        ja  dotwice1
            mov ax,0A000h
            mov es,ax
            mov dx,PicPage
            mov ax,4f05h
            mov bx,0
            int 10h
            cld
            mov cx,recover_pic_x
            mov ax,@data
            mov ds,ax
            mov ax,0A000h
            mov es,ax
            lea si,file_in
            mov di,PicOffset        ;ds:si-->es:di
            store_no_yellow
            jmp finish_store_datas
        dotwice1:
            mov ax,0A000h
            mov es,ax
            mov dx,PicPage
            mov ax,4f05h
            mov bx,0
            int 10h

            cld
            mov cx,0ffffh
            mov dx,PicOffset
            sub cx,dx
            mov PicFirstcnt,cx
            mov ax,@data
            mov ds,ax
            mov ax,0A000h
            mov es,ax
            lea si,file_in
            mov di,PicOffset        ;ds:si-->es:di
            store_no_yellow


            inc PicPage
            mov ax,0A000h
            mov es,ax
            mov dx,PicPage
            mov ax,4f05h
            mov bx,0
            int 10h
            cld

            mov cx,recover_pic_x
            mov dx,PicFirstcnt
            sub cx,dx
            mov ax,@data
            mov ds,ax
            mov ax,0A000h
            mov es,ax
            lea bx,file_in
            add bx,PicFirstcnt
            mov si,bx
            mov di,0        ;ds:si-->es:di
            store_no_yellow
        finish_store_datas:

        mov ebx,pageOffset
        xor edx,edx
        add ebx,800
        mov pageOffset,ebx

        xor ebx,ebx
        mov bx,file_f16b
        shl ebx,16
        mov dx,file_b16b
        add ebx,edx
        mov dx,recover_pic_x
        add ebx,edx
        mov file_b16b,bx
        shr ebx,16
        mov file_f16b,bx

        inc screen_row
        mov ax,recover_pic_y
        cmp screen_row,ax
    jle row1_print
    mov ah, 3Eh                 ;close file
    mov bx, file_handle
```

```
    int 21h
    ret
Print_No_Yellow endp

get_time        proc    near
        push    ds
        push    si
        xor     ax,ax
        mov     si,46ch
        mov     ds,ax

        cli

        mov     al,0
        out     43h,al

        in      al,40h
        mov     bl,al
        in      al,40h
        mov     bh,al
        ; not     bx
        mov     ax,bx
ok:     stosw
        movsw
        movsw

        sti
        pop     si
        pop     ds
        ret
get_time        endp
Print_clean proc

    xor eax,eax
    xor edx,edx
    mov ax,recover_pic_starty
    mov bx,800
    mul bx
    xor ebx,ebx
    mov bx,dx
    shl ebx,16
    add ebx,eax
    xor eax,eax
    mov ax,recover_pic_startx
    add ebx,eax
    mov PageOffset,ebx


    ; mov file_f16b,0
    ; mov file_b16b,0
    mov screen_row,0
    row2_print:

        mov ebx,PageOffset
        mov PicOffset,bx
        shr ebx,16
        mov PicPage,bx

        ;for double buffer
        cmp now_buffer_index,1
        je _pC_buffer0
        cmp now_buffer_index,0
        je _pC_buffer1

        _pC_buffer0: ;from page 0
        jmp _pc_end
        _pC_buffer1:  ;from page 10
        mov ax,PicPage
        add ax,10
        mov PicPage,ax

        _pc_end:

        mov ebx,PageOffset
        and ebx,0ffffh
        xor eax,eax
        mov ax,recover_pic_x
        add ebx,eax
        cmp ebx,0ffffh

        ja  dotwice2
            mov ax,0A000h
            mov es,ax
            mov dx,PicPage
            mov ax,4f05h
            mov bx,0
            int 10h
            cld

            mov cx,recover_pic_x
            mov ax,@data
            mov ax,0A000h
            mov es,ax
            mov al,0
            mov di,PicOffset        ;ds:si-->es:di
            rep stosb
            jmp finish_store_data2
        dotwice2:
            mov ax,0A000h
            mov es,ax
            mov dx,PicPage
            mov ax,4f05h
            mov bx,0
            int 10h

            cld
            mov cx,0ffffh
            mov dx,PicOffset
            sub cx,dx
            mov PicFirstcnt,cx
            mov ax,@data
            mov ax,0A000h
            mov es,ax
            mov al,0
            mov di,PicOffset        ;ds:si-->es:di
            rep stosb

            inc PicPage
            mov ax,0A000h
            mov es,ax
            mov dx,PicPage
            mov ax,4f05h
            mov bx,0
            int 10h
            cld

            mov cx,recover_pic_x
            mov dx,PicFirstcnt
            sub cx,dx
            mov ax,@data
            mov ds,ax
            mov ax,0A000h
            mov es,ax
            lea bx,file_in
            add bx,PicFirstcnt
            mov al,0
            mov di,0        ;ds:si-->es:di
            rep stosb
        finish_store_data2:

        mov ebx,pageOffset
        xor edx,edx
        add ebx,800
        mov pageOffset,ebx

        inc screen_row
        mov ax,recover_pic_y
        cmp screen_row,ax
    jle row2_print
    mov ah, 3Eh                 ;close file
    mov bx, file_handle
    int 21h
    ret
```

```
Print_clean endp

sound proc
    mov ax,@data
    mov ds,ax
    mov es,ax
    mov di,now_lower_index
    shl di,1
    lea bx,blockX
    mov dx,[bx+di]
    mov lowest_Xval,dx
    lea bx,blockY
    mov ax,[bx+di]
    mov lowest_Yval,ax

    .if lowest_Yval>=300
    mov hit_area,1
    .else
    mov hit_area,0
    .endif
    mov recover_pic_x,100
    mov recover_pic_y,100
    mov recover_pic_starty,500
    mov recover_pic_startx,0



    mov ax,@data
    mov ds,ax
    mov es,ax
    lea ax,pic_sz
    .if key=='z'
        music 1
        ;show button
        lea ax,pic_bz

        cmp lowest_Xval , 0
        jne z_no_hit
    cmp lowest_Yval , 300
    jb  z_no_hit
    ; jmp z_no_hit
    z_hit:

        ;add score
        mov ax , score
        add ax , 100
        mov score , ax

        ;clear block
        mov di,now_lower_index
        shl di,1
        lea bx,BlockY
        mov word ptr [bx+di],0
        ;set_hit_alert
        .if lowest_Yval<=330
        mov hit_alert_flag,1
        .elseif lowest_Yval<=375
        mov hit_alert_flag,2
        .else
        mov hit_alert_flag,1
        .endif
        mov hit_alert_counter, 10
         ;lowest_index
        mov ax,now_lower_index
        inc ax
        mov dx,0
        mov bx,4
        div bx
        mov now_lower_index,dx
        jmp z_end
    z_no_hit:
        jmp z_end
    z_end:

.endif
mov pic_address,ax
```

```
    call Print_Pic
    mov ax,@data
    mov ds,ax
    mov es,ax
    mov recover_pic_startx,115
    lea ax,pic_sx
    .if key=='x'
       music 2;*
        ;show button
        lea ax,pic_bx ;*

        cmp lowest_Xval , 115 ;*
        jne x_no_hit
        cmp lowest_Yval , 300
        jb  x_no_hit
        ; jmp x_no_hit
        x_hit:

            ;add score
            mov ax , score
            add ax , 100
            mov score , ax

            ;clear block
            mov di,now_lower_index
            shl di,1
            lea bx,BlockY
            mov word ptr [bx+di],0
            ;set_hit_alert
            .if lowest_Yval<=330
            mov hit_alert_flag,1
            .elseif lowest_Yval<=375
            mov hit_alert_flag,2
            .else
            mov hit_alert_flag,1
            .endif
            mov hit_alert_counter, 10
             ;lowest_index
            mov ax,now_lower_index
            inc ax
            mov dx,0
            mov bx,4
            div bx
            mov now_lower_index,dx
            jmp x_end
        x_no_hit:
            jmp x_end
        x_end:

    .endif
    mov pic_address,ax
    call Print_Pic
    mov ax,@data
    mov ds,ax
    mov es,ax
    mov recover_pic_startx,230
    lea ax,pic_sc
    lea bx,can_hit_index
    mov dl,[bx+2]
    mov hit_a_sign,dl
    .if key=='c'
         music 3;*
        ;show button
        lea ax,pic_bc ;*

        cmp lowest_Xval , 115*2 ;*
        jne c_no_hit
        cmp lowest_Yval , 300
        jb  c_no_hit
        ; jmp c_no_hit
        c_hit:

            ;add score
            mov ax , score
            add ax , 100
            mov score , ax
            ;clear block
            mov di,now_lower_index
            shl di,1
            lea bx,BlockY
            mov word ptr [bx+di],0
            ;set_hit_alert
            .if lowest_Yval<=330
            mov hit_alert_flag,1
            .elseif lowest_Yval<=375
            mov hit_alert_flag,2
            .else
            mov hit_alert_flag,1
            .endif
            mov hit_alert_counter, 10
             ;lowest_index
            mov ax,now_lower_index
            inc ax
            mov dx,0
            mov bx,4
            div bx
            mov now_lower_index,dx
            jmp c_end
        c_no_hit:
            jmp c_end
        c_end:
    .endif
    mov pic_address,ax
    call Print_Pic
    mov ax,@data
    mov ds,ax
    mov es,ax
    mov recover_pic_startx,345
    lea ax,pic_sv
    mov hit_a_sign,dl
    .if key=='v'
         music 4;*
        ;show button
```

```
        lea ax,pic_bv ;*

        cmp lowest_Xval , 115*3 ;*
        jne v_no_hit
        cmp lowest_Yval , 300
        jb  v_no_hit
        ; jmp v_no_hit
        v_hit:

            ;add score
            mov ax , score
            add ax , 100
            mov score , ax

            ;clear block
            mov di,now_lower_index
            shl di,1
            lea bx,BlockY
            mov word ptr [bx+di],0
            ;set_hit_alert
            .if lowest_Yval<=330
            mov hit_alert_flag,1
            .elseif lowest_Yval<=375
            mov hit_alert_flag,2
            .else
            mov hit_alert_flag,1
            .endif
            mov hit_alert_counter, 10
             ;lowest_index
            mov ax,now_lower_index
            inc ax
            mov dx,0
            mov bx,4
            div bx
            mov now_lower_index,dx
            jmp v_end
        v_no_hit:
            jmp v_end
```

```
    v_end:
.endif
mov pic_address,ax
call Print_Pic
mov ax,@data
mov ds,ax
mov es,ax
mov recover_pic_startx,460
lea ax,pic_sl

.if key==','
    music 5;*
    ;show button
    lea ax,pic_bl ;*

    cmp lowest_Xval , 115*4 ;*
    jne l_no_hit
    cmp lowest_Yval , 300
    jb  l_no_hit
    ; jmp l_no_hit
    l_hit:

        ;add score
        mov ax , score
        add ax , 100
        mov score , ax

        ;clear block
        mov di,now_lower_index
        shl di,1
        lea bx,BlockY
        mov word ptr [bx+di],0
        ;set_hit_alert
        .if lowest_Yval<=330
        mov hit_alert_flag,1
        .elseif lowest_Yval<=375
        mov hit_alert_flag,2
        .else
        mov hit_alert_flag,1
        .endif
        mov hit_alert_counter, 10
         ;lowest_index
        mov ax,now_lower_index
        inc ax
        mov dx,0
        mov bx,4
        div bx
        mov now_lower_index,dx
        jmp l_end
    l_no_hit:
        jmp l_end
    l_end:
.endif
mov pic_address,ax
call Print_Pic
mov ax,@data
mov ds,ax
mov es,ax
mov recover_pic_startx,575
lea ax,pic_sr
lea bx,can_hit_index
mov dl,[bx+5]
mov hit_a_sign,dl
.if key=='.'
           music 6;*
    ;show button
    lea ax,pic_br ;*

    cmp lowest_Xval , 115*5 ;*
    jne r_no_hit
    cmp lowest_Yval , 300
    jb  r_no_hit
    ; jmp r_no_hit
    r_hit:

        ;add score
        mov ax , score
        add ax , 100
        mov score , ax

        ;clear block
        mov di,now_lower_index
        shl di,1
        lea bx,BlockY
        mov word ptr [bx+di],0
        ;set_hit_alert
        .if lowest_Yval<=330
        mov hit_alert_flag,1
        .elseif lowest_Yval<=375
        mov hit_alert_flag,2
        .else
        mov hit_alert_flag,1
        .endif
        mov hit_alert_counter, 10
         ;lowest_index
        mov ax,now_lower_index
        inc ax
        mov dx,0
        mov bx,4
        div bx
        mov now_lower_index,dx
        jmp r_end
    r_no_hit:
        jmp r_end
    r_end:
.endif
mov pic_address,ax
call Print_Pic

mov ax,@data
mov ds,ax
mov es,ax
mov recover_pic_startx,690
lea ax,pic_sq
lea bx,can_hit_index
mov dl,[bx+6]
mov hit_a_sign,dl
.if key=='/'
           music 7;*
    ;show button
    lea ax,pic_bq ;*

    cmp lowest_Xval , 115*6 ;*
    jne q_no_hit
    cmp lowest_Yval , 300
    jb  q_no_hit
    ; jmp q_no_hit
    q_hit:

        ;add score
        mov ax , score
        add ax , 100
        mov score , ax

        ;clear block
        mov di,now_lower_index
        shl di,1
        lea bx,BlockY
        mov word ptr [bx+di],0
        ;set_hit_alert
        .if lowest_Yval<=330
        mov hit_alert_flag,1
        .elseif lowest_Yval<=375
        mov hit_alert_flag,2
        .else
        mov hit_alert_flag,1
        .endif
        mov hit_alert_counter, 10
         ;lowest_index
        mov ax,now_lower_index
        inc ax
        mov dx,0
```

```
            mov bx,4
            div bx
            mov now_lower_index,dx
            jmp q_end
        q_no_hit:
            jmp q_end
        q_end:
    .endif
    mov pic_address,ax
    call Print_Pic
    ret
sound endp

tran_to_num_pic proc
mov cx,0
L3:
    inc cx
    mov bx,10
    mov dx,0
    div bx
    push dx
    cmp ax,0
    jne L3
    mov bx,5
    sub bx,cx
number:
    pop ax
    add al,30h
    mov [di+bx],al
    inc di
    loop number
mov cx,5
mov di,0
L2:
push cx
push di

lea bx,number_buffer
```

```
mov al,[bx+di]
.if al==30h
lea ax,pic_0
.elseif al==31h
lea ax,pic_1
.elseif al==32h
lea ax,pic_2
.elseif al==33h
lea ax,pic_3
.elseif al==34h
lea ax,pic_4
.elseif al==35h
lea ax,pic_5
.elseif al==36h
lea ax,pic_6
.elseif al==37h
lea ax,pic_7
.elseif al==38h
lea ax,pic_8
.else
lea ax,pic_9
.endif
mov pic_address,ax
mov ax,di
shl ax,4
mov bx,x_shift
add ax,bx
mov recover_pic_startx,ax
mov recover_pic_x,20
mov recover_pic_y,20
call Print_Pic
pop di
inc di
pop cx
dec cx
cmp cx,0
ja L2
ret
```

```
tran_to_num_pic endp

Print_racebackground proc
    mov ax,@data
    mov ds,ax
    mov ah,3Dh                  ;Open file
    mov al,0                    ;0,R 1,W 2,R/W
    mov dx,pic_address
    int 21h

    pushf                       ;read flag val
    pop bx
    and bx,0001
    .if bx==0001                ;verify CF is set(error)
    PrintStr fail_open
    .endif
    mov file_handle,ax          ;ax return file_handle

    xor eax,eax
    xor edx,edx
    mov ax,recover_pic_starty
    mov bx,800
    mul bx
    xor ebx,ebx
    mov bx,dx
    shl ebx,16
    add ebx,eax
    xor eax,eax
    mov ax,recover_pic_startx
    add ebx,eax
    mov PageOffset,ebx

    mov file_f16b,0
    mov ax,racebg_shift_x
    mov file_b16b,ax
    mov screen_row,0
    line_print:
        Set_file_pointer file_f16b,file_b16b
        mov bx,file_handle
        mov cx,recover_pic_x                ; 140 Bytes to read
        lea dx,file_in
        mov ah,3fh
        int 21h

        mov ebx,PageOffset
        mov PicOffset,bx
        shr ebx,16
        mov PicPage,bx

        mov ebx,PageOffset
        and ebx,0ffffh
        xor eax,eax
        mov ax,recover_pic_x
        add ebx,eax
        cmp ebx,0ffffh

        ja  dotwicebg
            mov ax,0A000h
            mov es,ax
            mov dx,PicPage
            mov ax,4f05h
            mov bx,0
            int 10h
            cld

            mov cx,recover_pic_x
            mov ax,@data
            mov ds,ax
            mov ax,0A000h
            mov es,ax
            lea si,file_in
            mov di,PicOffset        ;ds:si-->es:di
            rep movsb
            jmp finish_store
        dotwicebg:
            mov ax,0A000h
            mov es,ax
            mov dx,PicPage
            mov ax,4f05h
            mov bx,0
            int 10h

            cld
            mov cx,0ffffh
            mov dx,PicOffset
            sub cx,dx
            mov PicFirstcnt,cx
            mov ax,@data
            mov ds,ax
            mov ax,0A000h
            mov es,ax
            lea si,file_in
            mov di,PicOffset        ;ds:si-->es:di
            rep movsb

            inc PicPage
            mov ax,0A000h
            mov es,ax
            mov dx,PicPage
            mov ax,4f05h
            mov bx,0
            int 10h
            cld

            mov cx,recover_pic_x
            mov dx,PicFirstcnt
            sub cx,dx
            mov ax,@data
            mov ds,ax
            mov ax,0A000h
            mov es,ax
            lea bx,file_in
            add bx,PicFirstcnt
            mov si,bx
            mov di,0        ;ds:si-->es:di
            rep movsb
        finish_store:


        mov ebx,pageOffset
        xor edx,edx
        mov dx,800
        add ebx,edx
        mov pageOffset,ebx

        xor ebx,ebx
        mov bx,file_f16b
        shl ebx,16
        mov dx,file_b16b
        add ebx,edx
        mov dx,1600
        add ebx,edx
        mov file_b16b,bx
        shr ebx,16
        mov file_f16b,bx
        mov ax,screen_row
        mov dx,0
        mov bx,1
        div bx
        mov shift_cnt,dx
        .if screen_row > 285 && shift_cnt == 0
        .endif



        inc screen_row
        mov ax,recover_pic_y

        cmp screen_row,ax
    jle line_print
    mov ah, 3Eh                 ;close file
    mov bx, file_handle
    int 21h
```

```
    ret
Print_racebackground endp
end start
```