

# 組合語言與系統程式\_\_期末專題書面報告\_\_跳棋

## 第 07 組

數學三 B 鐘彥杰 102201036

數學三 B 王鈺鎔 102201024

數學三 B 李子婕 102201501

### 一、動機

我們想要以組合語言來實做一個有趣的程式、例如；大富翁、跳棋、圍棋...等，因為考量到小黑窗的畫面，覺得寫大富翁程式會不好看，所以排除大富翁，然後在所有棋類中，又較喜歡跳棋，因此最後決定寫跳棋程式。

### 二、程式函式

#### Initial Data

```
;15隻紅棋位置
R COOR <-4, 8>, ..... , <-2, 4>, <-1, 4>, < 0, 4>
;15隻綠棋位置
G COOR <-4, 0>, ..... , <-2,-4>, <-1,-4>, < 0,-4>
;15隻黃棋位置
Y COOR < 4, 0>, ..... , < 6,-4>, < 7,-4>, < 8,-4>

-----
dot byte "o "           ;棋子樣式
cursor COOR < 0, 0>      ;游標位置
control byte 1           ;控制權
```

#### movecursor PROTO

此函式用來移動游標，當鍵入 enter 則跳出

```
UP: 以 "↑" 為例
    mov bh, cursor.X      游標目前位置
    mov bl, cursor.Y
    add bl, 1
    INVOKE Boundary, bh, bl
    cmp ax, 0             判斷游標移動後的位置
    jz WaitInput          是否在邊界裡
    mov cursor.X, bh      移動游標
    mov cursor.Y, bl
    INVOKE print
    jmp WaitInput

WaitInput:
    call readchar          讀入鍵盤
    cmp eax, 4800h
    jz UP
    cmp eax, 5000h
    jz DOWN
    cmp eax, 4B00h
    jz LEFT
    cmp eax, 4D00h
    jz RIGHT
    cmp eax, 1C0Dh (Enter鍵)
    jz OUTFUN
    jmp WaitInput
```

## Boundary PROC, x:sbyte, y:sbyte

input : x : 遊戲內棋盤 X 座標

y : 遊戲內棋盤 Y 座標

return : ax : 0 - 界外, 1 - 界內

此函式用來判斷游標移動是否出界

```
Boundary PROC, ux:SBYTE, uy:SBYTE
```

```
    push ebx
```

```
test_uptri:
```

```
    ;判斷是否在上三角裡，只要有一項不符合，  
    ;就直接跳去判斷是不是下三角里
```

```
    cmp uy, -4
```

```
    jl test_downtri    ;y<-4
```

```
    cmp ux, -4
```

```
    jl test_downtri    ;x<-4
```

```
    mov bl, ux
```

```
    add bl, uy
```

```
    cmp bl, 4
```

```
    jg test_downtri    ;x+y>4
```

```
    jmp Istrue
```

```
    ;三項都符合，所以在上三角裡
```

```
    ;就直接跳去Istrue，回傳true(ax=1)，結束程式
```



```
Istrue:
```

```
    mov ax, 1
```

```
    jmp existBoundary
```

```
Isfalse:
```

```
    mov ax, 0
```

```
existBoundary:
```

```
    pop ebx
```

```
    ret
```

```
Boundary ENDP
```

← 在邊界裡

← 不在邊界裡

-----

## IsChess PROTO, boolx:sbyte, booly:sbyte

input : boolx : 遊戲內棋盤 X 座標

booly : 遊戲內棋盤 Y 座標

return : al : 0 - 無棋子, 1 - 紅棋, 2 - 綠棋, 3 - 黃棋

此函式用來處理選擇棋子、"跳"棋移動

```
find:
    mov bl, (COORD PTR [edi]).X
    mov bh, (COORD PTR [edi]).Y
;記憶體位置連續
    cmp boolx, bl
    jnz addpointer
    cmp booly, bh
    jnz addpointer
    jz lookecx ;符合後則判斷是什麼顏色
addpointer:
    add edi, TYPE COORD
    loop find
;若不符合，就繼續判斷下一個

findr: ;是綠色旗子
    mov al, 1
    pop edi
    pop ebx
    pop ecx
    ret
;回傳

lookecx:
    cmp ecx, 30
    jg findr
    cmp ecx, 15
    jg findg
    cmp ecx, 0
    jg findy
    cmp ecx, 0
    jz nofind
```

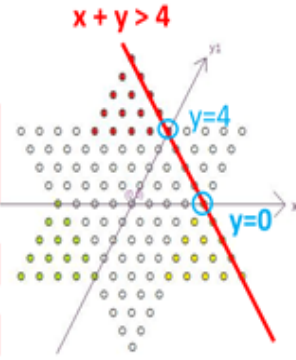
## Iswin PROTO

return : ax : 0 - 無玩家勝利, 1 - 有玩家獲勝

bx : 1 - 紅棋獲勝, 2 - 綠棋獲勝, 3 - 黃棋獲勝

此函式用來判斷哪位玩家獲勝

```
checkG:
    mov ecx, 15
    mov edi, 0
Gloop:
    mov bl, (COORD PTR G[edi]).X
    add bl, (COORD PTR G[edi]).Y
    cmp bl, 4
    jl checkY
    cmp (COORD PTR G[edi]).Y, 0
    jl checkY
    cmp (COORD PTR G[edi]).Y, 4
    jg checkY
    add edi, TYPE COORD
    loop Gloop;
    mov bx, 2
    jmp Win
```



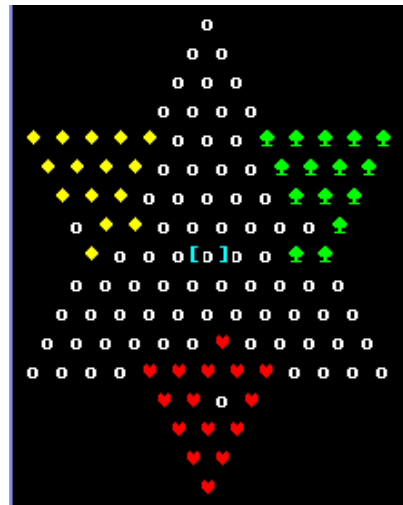
## print PROTO

此函式用來將棋子印到小黑窗

先將六角形棋盤印出(以白圓圈表示)，再依序接著印出所有紅棋、綠棋和黃棋，最後印出游標青藍色[]。

;以紅棋為例---大棋盤紅點---。按照矩陣的內容，透過 Transfer 轉換後繪出

```
mov ecx, 15
mov esi, OFFSET R
printred:
mov bh, (COORD PTR [esi]).X
mov bl, (COORD PTR [esi]).Y
INVOKE Transfer, bh, bl
call Gotoxy
;設定前景為淡紅色，背景為黑色
mov eax, 12 + (black*16)
call SetTextColor
mov al, 3h
call Writechar
add esi, TYPE COOR
loop printred
```



---

## Transfer PROTO, X1:sbyte, Y1:sbyte

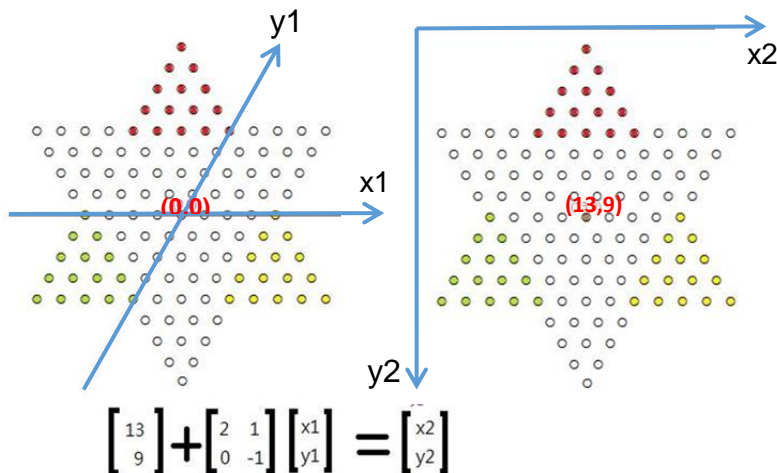
input : X1 : 遊戲內棋盤 X 座標

Y1 : 遊戲內棋盤 Y 座標

return : dh : 小黑窗 Y 座標,

dl : 小黑窗 X 座標

此函式用來處理棋盤座標與小黑窗座標間的轉換，轉換方式如下所示：



## Choose PROTO

return : esi : 鎖定的棋子指標

此函式用來選擇棋子

```
stage_move:
    INVOKE Movecursor      ;移動游標
    INVOKE Boundary, cursor.X, cursor.Y.
    cmp ax, 1              ;確認游標有沒有在棋盤內
    jne stage_move         ;沒有的話回到移動游標的狀態
    INVOKE IsChess, cursor.X, cursor.Y
    cmp al, control        ;選錯的話回到移動游標的狀態
    jne choose_yourself    ;確認選到的是不是正確的棋子
    mov bl, al
    mov ecx, 15             ;count=15
    mov ah, cursor.X        ;把游標的XY傳進ax
    mov al, cursor.Y
    cmp bl, 1              ;是紅棋
    je chessR
    cmp bl, 2              ;是綠棋
    je chessG
    cmp bl, 3              ;是黃棋
    je chessY
chessR:
    mov edi, OFFSET R      ;edi指到R(紅棋)的起始位置
    jmp stage_return

stage_return:
    cmp ah, (COORD PTR [edi]).X;
    jnz reloop
    cmp al, (COORD PTR [edi]).Y;
    jnz reloop;
    jmp stage_find;
reloop:
    add edi, TYPE COORD
    loop stage_return

choose_yourself:           ;防呆，選到不該選的棋子
    mov dh, 0
    mov dl, 0              dh=row, dl=col
    call Gotoxy             ;重新定位游標在主控視窗中的位置
    mov edx, OFFSET chooseyourschess
    call WriteString        ;印錯誤訊息
    INVOKE Transfer, cursor.X, cursor.Y
    call Gotoxy
    jmp stage_move         ;選錯的話回到移動游標的狀態
```



## movechess PROTO

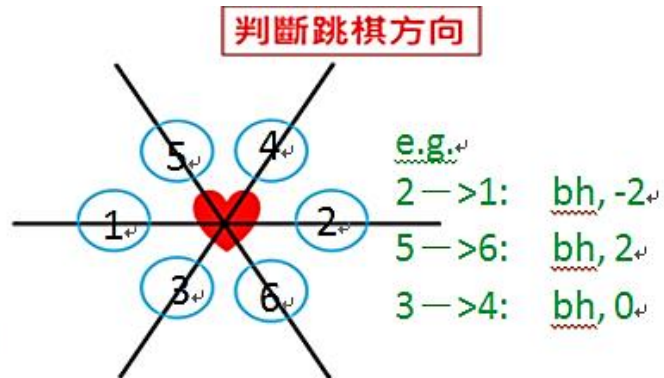
return : ax : 0 - 未移動, 1 - 移動

此函式用來移動棋子跳或者走

; 這裡處理"跳"棋

jump:

```
cmp bh, 2
jz xplustwo
cmp bh, 0
jz xremain2
cmp bh, -2
jz xminustwo
jmp jumpagain
```

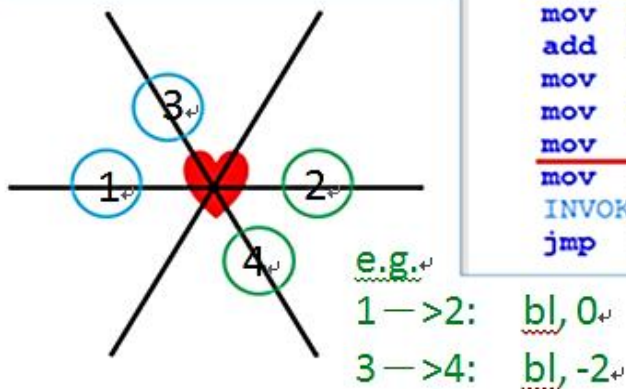


xplustwo:

```
cmp bl, 0
jz jumpright
cmp bl, -2
jz jumprightdown
jmp jumpagain
```

jumpright:

```
mov bh, cursor.X
add bh, -1 判斷中間有無棋子
mov bl, cursor.Y
INVOKE IsChess, bh, bl
cmp al, 0
jz jumpagain
mov bh, chessx
add bh, 2
mov (COOR PTR [esi]).X, bh
mov bl, chessy
mov (COOR PTR [esi]).Y, bl
mov isjmp, 1
INVOKE print
jmp jumpagain
```



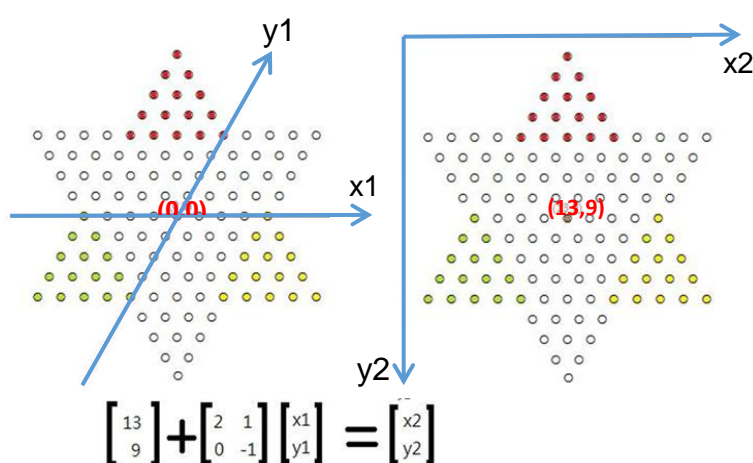
### 三、程式架構

先判斷有無玩家勝利，再按照控制權決定誰選取棋子、移動，繪出移動後的棋盤，用上、下、左、右移動游標、棋子，再用 Enter 選取、取消選取、移動棋子。

### 四、心得

決定要寫跳棋後，一開始覺得無從下手，後來我們先一起討論程式架構，確認跳棋遊戲是否能寫得出來，並針對跳棋的遊戲規則，規劃程式架構。

第一個我們遇到的難題是，我們所討論出所有的遊戲規則都是建立在棋盤的六角形座標上，然而棋盤座標跟黑屏座標間無法完全對應，所以我們用一個簡單的座標轉換矩陣，這樣我們寫的所有遊戲規則都用六角形座標實作，只有在輸出之前轉換成黑屏座標即可。



其中最難寫與複雜的函式為 Choose 和 movechess 兩個函式，在 debug 過程中有許多小意外，例如：棋子會飛到棋盤外、棋子消失、或是無法移動.....等等，要很有耐心的看程式、改程式。最後我們在程式可以正確執行後，加上一些人性化的介面設計，讓玩家能更清楚的掌握遊戲情況。

這次的專題實作讓我們更熟悉 x86 組合語言的使用，以及如何分割程式架構，分派工作，在實際編寫的時候要如何溝通(例如：回傳值要放在哪裡，哪些暫存器不能使用.....等等)，最後如何把大家寫的函式組合起來以及最重要的偵錯！

PS. 一定要在寫程式的當下就把註解寫出來，不僅有利別人之後的閱讀與理解，尤其組合語言不像高階語言容易閱讀，不然過一段時間後，就會忘記自己當初的用意了。

### 五、工作分配

鐘彥杰 - 書面報告製作、架構設計、除錯、程式統整、

程式撰寫(main、print、movecursor、movechess)

王鈺鎔 - 書面報告製作、架構設計、除錯、遊戲畫面設計、

程式撰寫 (Boundary、Iswin、Choose)

李子婕 - 書面報告製作、投影片製作、架構設計、

程式撰寫(Transfer、IsChess)