



รายงาน

Assignment ชั้นที่ 1

เรื่อง The Metrix-Screen by Assembly

วิชา Computer Organization รหัสวิชา 01076246

เสนอ

อาจารย์ อัครเดช วัชรภูพงษ์

จัดทำโดย

นาย ชันติชัย รุจิตรการโชติกุล รหัส 57010127

นางสาว นานฎิศา เจนนพกาญจน์ รหัส 57010681

ภาคเรียนที่ 2 ปีการศึกษา 2558

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

คำนำ

รายงานเล่มนี้เป็นส่วนหนึ่งของวิชา Computer Organization รหัสวิชา 01076246 โดยคณะผู้จัดทำได้เขียนโปรแกรมแสดงหน้าจอ Matrix screen และได้อธิบายหลักการทำงานของโปรแกรมเบื้องต้น Pseudo code และสาธิตวิธีการคอมไพล์และรันโปรแกรมพร้อมภาพประกอบผลลัพธ์ของโปรแกรม อีกทั้งแนบลิงค์ผลการรันโปรแกรมอีกด้วย

คณะผู้จัดทำ

สารบัญ

	หน้า
หลักการทำงานของโปรแกรม	1
Pseudo code โปรแกรม the matrix-screen	2
วิธีการคอมไพล์และสั่งรัน	11
Source Code	28
ลิงค์ผลการรันโปรแกรม	32

หลักการทำงานของโปรแกรม

การทำงานของโปรแกรมแบ่งเป็น 2 ส่วนหลักๆ คือ

1. ส่วนดาต้า

ประกาศตัวแปรไว้ใช้ในส่วนโค้ด

2. ส่วนโค้ด

1. เริ่มต้นจากการใส่ข้อมูลโดยสุ่มค่าแถวที่จะเกิดและอักษรที่จะเกิดไปที่ละcolumn
2. ทำการอ่านข้อมูลที่ใส่ไว้โดยเขียนแสดงอักษรและสีตามที่กำหนด
3. ทำการเลื่อน cursor เมื่อแถวอยู่ในตำแหน่งมากกว่าหรือเท่ากับ 0 และแสดงไปยังตำแหน่งที่เขียนไว้โดยไล่ทำไปทุก column
4. ไล่ดีเลย์เพื่อหน่วงเวลา
5. ไล่เขียนทับตั้งแต่สีเขียวจาง สีเขียว และเขียวเข้มทุก column
6. ไล่ลบอักษรที่ทางแถวทุก column
7. เมื่อเริ่มแถวใหม่ให้สุ่มอักษรใหม่ทุกครั้ง
8. เมื่อไหลลงจนพ้นหน้าหน้าจอให้เริ่มใหม่ด้านบนจนจบวนไปเรื่อย

Pseudo code โปรแกรม the matrix-screen

ส่วนดาต้า

- ประกาศตัวแปรเพื่อเก็บข้อความเครดิต
- ประกาศตัวแปรarray เพื่อเก็บตัวอักษร
- ประกาศตัวแปรเพื่อเก็บแถว
- ประกาศตัวแปรเพื่อเก็บหางของแถว
- ประกาศตัวแปรเพื่อเก็บความยาวสีเขียวจาง
- ประกาศตัวแปรเพื่อเก็บความยาวสีเขียว
- ประกาศตัวแปรเพื่อเก็บความยาวสีเขียวเข้ม
- ประกาศตัวแปร random แถว
- ประกาศตัวแปร random ตัวอักษร
- ประกาศค่าคงที่สำหรับการ random 2 ตัว

ส่วนโค้ด

Main:

- 1.เรียกใช้ video mode
- 2.สร้างหน้าต่างขนาด 80x25

-----ส่วนในการ assign data ให้กับ array-----

- 3.กำหนดค่าเริ่มต้นให้กับรีจิสเตอร์เพื่อใช้ตรวจสอบ column
- 4.ให้รีจิสเตอร์เก็บค่าที่อยู่ของตัวอักษร,แถว และหางแถว

5.ทำการ random สัญญาณนาฬิกาแล้วเก็บไว้ในตัวแปร random แฉว และ
อักษร

Add_data:

6.ทำการเปรียบเทียบว่าอยู่ในcolumnสุดท้ายหรือไม่

7.ถ้าค่ามากกว่าcolumnสุดท้ายให้กระโดดไปที่ end_add_data

Rand0_79:

8.นำค่าที่ได้จากการ random สัญญาณนาฬิกามาทำการ random number
generator โดยวิธี Computation method คือ $X_{n+1} = (aX_n + b) \bmod m$ โดย $m=80$
เพื่อต้องการค่า 0-79

End_rand0_79:

9.กำหนดค่าเริ่มต้นให้แถวเท่ากับ 0และนำค่าที่ random ได้ไปลบเพื่อให้เริ่มแถว
ตรงจอด้านบน

10.ให้รีจิสเตอร์เก็บความยาวแถว

11.ทำการกำหนดค่าให้กับหางของแถวตามความยาวแถว

Rand33_127:

12.นำค่าที่ได้จากการ random สัญญาณนาฬิกามาทำการ random number
generator โดยวิธี Computation method คือ $X_{n+1} = (aX_n + b) \bmod m$ เพื่อต้องการค่า
33-127

End_rand33_127:

13.นำค่าที่ random ในข้อ12 มาใส่เพื่อสู่่มการเกิดอักษร

14.ทำการเพิ่มค่ารีจิสเตอร์ที่ชี้แถว หางแถวและตัวอักษรเพื่อเตรียมใส่ข้อมูลช่องถัดไป

15.กระโดดไปที่ add_data

-----ส่วนในการเขียนอักษรหัวแถวสีขาว-----

My loop:

16.ให้รีจิสเตอร์ที่ชี้array ที่เก็บแถว หางแถวและตัวอักษรชี้ไปที่ช่องแรกทุกตัว และให้column เริ่มที่0

Start_col:

17.ทำการเปรียบเทียบว่าอยู่ในcolumnสุดท้ายหรือไม่

18.ถ้าใช่กระโดดไป end_col

19.เปรียบเทียบว่า row>=0 หรือไม่

20.ถ้าใช่กระโดดไปที่ Start_row

21.ถ้าไม่ใช่ให้กระโดดไปที่ end_row

Start_row:

22.ทำการ move cursor ไปตำแหน่ง row ในขณะนั้น

23.เขียนอักษรที่pointerในขณะนั้นขึ้นอยู่

End_row:

24.เพิ่มcolumnและเลื่อนตำแหน่ง pointer ที่เก็บตัวอักษรและแถวไปยังตำแหน่งถัดไป

25.กระโดดไปที่ start_row

-----ส่วนในการเขียนเครดิตมุมขวาล่างจอ-----

26.ใช้ฟังก์ชันในการเขียน string กำหนดสีและตำแหน่ง

-----ส่วนdelay ในการแสดงผล-----

Delay:

27.หน่วยเวลา clock เพื่อให้เห็นการแสดงผล

End_delay:

-----ส่วนในการเขียนอักขรสีเขียวจาง-----

28.ให้รีจิสเตอร์ที่ชี้array ที่เก็บแถว หางแถวและตัวอักษรชี้ไปที่ช่องแรกทุกตัว
และให้column เริ่มที่0

Start_col2:

29.ทำเช่นเดียวกับส่วนหัวสีขาวเมื่อเช็คเงื่อนไขเป็นจริงกระโดดไป
start_row2 ถ้าไม่ใช่กระโดดไปที่ end_row2

Start_row2:

30.ทำเช่นเดียวกับส่วนหัวสีขาวคือเลื่อน cursor และเขียนอักษร

End_row2:

31.เพิ่มcolumnและเลื่อนตำแหน่ง pointer ที่เก็บตัวอักษรและแถวแล้ว
กระโดดไป start_col2

End_col2:

-----ส่วนในการเขียนอักขรสีเขียว-----

32.ให้รีจิสเตอร์ที่ชี้array ที่เก็บแถว หางแถวและตัวอักษรชี้ไปที่ช่องแรกทุกตัว
และให้column เริ่มที่0

Start_col3:

33.ทำเช่นเดียวกับส่วนหัวสีขาวเมื่อเช็คเงื่อนไขเป็นจริงกระโดดไป
start_row3 ถ้าไม่ใช่กระโดดไปที่ end_row3

Start_row3:

34.ทำเช่นเดียวกับส่วนหัวสีขาวคือเลื่อน cursor และเขียนอักษร

End_row3:

35.เพิ่มcolumnและเลื่อนตำแหน่ง pointer ที่เก็บตัวอักษรและแถวแล้ว
กระโดดไป start_col3

End_col3:

-----ส่วนในการเขียนอักขรสีเขียวเข้ม-----

36.ให้รีจิสเตอร์ที่ชี้array ที่เก็บแถว หางแถวและตัวอักษรชี้ไปที่ช่องแรกทุกตัว
และให้column เริ่มที่0

Start_col4:

37.ทำเช่นเดียวกับส่วนหัวสีขาวเมื่อเช็คเงื่อนไขเป็นจริงกระโดดไป
start_row3 5hkไม่ใช่กระโดดไปที่ end_row3

Start_row4:

38.ทำเช่นเดียวกับส่วนหัวสีขาวคือเลื่อน cursor และเขียนอักษร

End_row4:

39.เลื่อนตำแหน่ง pointer ที่เก็บตัวอักษรและแถวแล้วกระโดดไป start_col3

End_col4:

-----ส่วนในการเคลียร์อักขรหรือถมดำ-----

40.ให้เริ่มต้นที่column 0

41.ให้pointerชี้ที่หางแถว

Start_col5:

42.ทำการเปรียบเทียบว่าอยู่ในcolumnสุดท้ายหรือไม่

43.ถ้าใช่กระโดดไป end_col5

44.เปรียบเทียบว่าหาง row>=0 หรือไม่

45.ถ้าใช่กระโดดไปที่ Start_row5

46.ถ้าไม่ใช่ให้กระโดดไปที่ end_row5

Start_row5:

47.ทำการ move cursor ไปตำแหน่ง row ในขณะนั้น

48.ลบอักษรเปลี่ยนพื้นเป็นสีดำ

End_row5:

48.เลื่อนcolumnและตำแหน่งหางแถวไปยังตัวถัดไป

49.กระโดดไปstart_col5

-----ส่วนในการเพิ่มแถว-----

50.ให้เริ่มต้นที่columnแรก

51.ให้pointer ชี้ที่แถวและอักษร

Loop_forward:

52.ทำการเปรียบเทียบว่าอยู่ในcolumnสุดท้ายหรือไม่

53.ถ้าค่ามากกว่าcolumnสุดท้ายให้กระโดดไปที่ end_loop_forward

54.ทำการเพิ่มค่า row

Rand0_127_2

55.ใช้วิธีการrandomเหมือนข้อ 8 เพื่อสุ่มอักษรอีกครั้ง

End_rand0_127_2

56.เลื่อนไปยังcolumnถัดไป

57.เลื่อนpointerไปชี้แถวและตัวอักษรถัดไป

58.กระโดดไปที่ loop_forward

End_loop_forward:

-----ส่วนในการเพิ่มหางแถว-----

59.ให้เริ่มต้นที่columnแรก

60.ให้pointer ชี้ที่หางแถว

Loop_forward2:

61.ทำการเปรียบเทียบว่าอยู่ในcolumnสุดท้ายหรือไม่

62.ถ้าค่ามากกว่าcolumnสุดท้ายให้กระโดดไปที่ end_loop_forward2

63.ทำการเพิ่มค่าหางแถว

64.เลื่อนไปยังcolumnถัดไป

65.เลื่อนpointerไปชี้ทางแถวถัดไป

66.กระโดดไปที่ loop_forward2

End_Loop_forward2:

-----ส่วนตรวจสอบแถว-----

67.ให้เริ่มต้นที่columnแรก

68.ให้pointer ชี้ที่แถวแรก

Loop_for_forward3:

69.ทำการเปรียบเทียบว่าอยู่ในcolumnสุดท้ายหรือไม่

70.ถ้าค่ามากกว่าcolumnสุดท้ายให้กระโดดไปที่ end_loop_forward2

71.ทำการเปรียบเทียบว่าจบแถวหรือยัง(row=32)

72.ถ้า row=32ให้กระโดดไป clear_row1

73.ถ้าไม่ใช่กระโดดไป pass_row1

Clear_row1:

74.ให้ค่า row เป็น 0

Pass_row1:

75.ให้เลื่อนcolumnและแถว

76.กระโดดไป loop_forward3

End_loop_forward3:

-----ส่วนตรวจสอบทางแถว-----

77.ให้เริ่มต้นที่columnแรก

78.ให้pointer ชี้ที่แถวแรก

Loop_for_forward4:

79.ทำการเปรียบเทียบว่าอยู่ในcolumnสุดท้ายหรือไม่

80.ถ้าค่ามากกว่าcolumnสุดท้ายให้กระโดดไปที่ end_loop_forward2

81.ทำการเปรียบเทียบว่าจบทางแถวหรือยัง (row_t=32)

82.ถ้า row=32ให้กระโดดไป clear_row_t

83.ถ้าไม่ใช่กระโดดไป pass_row_t

Clear_row_t:

84.ให้ค่า column เท่ากับค่าความยาวตั้งแต่หัวแถวถึงหางแถว

85.ให้ค่าrowเป็น0

Pass_row_t:

86.ให้เลื่อนcolumnและแถว

87.กระโดดไป loop_forward3

End_loop_forward4:

88.กระโดดไปที่ myloop เพื่อเริ่มใหม่อีกครั้ง

Ret

End main

Source Code

```
.model tiny

.data

cre    db    ' Coded by iHerePor x Nat '

cha    db    80    dup(?)

row    db    80    dup(?)

row_t  db    80    dup(?)

cl_1   db    1     ; Light Green Length

cl_2   db    4     ; Green Length

cl_3   db    7     ; Dark Grey Length

tl     db    8     ; Tail Length

rnd_79 dw    ?

rnd_127 dw    ?

var_a  dw    9797

var_b  dw    64977

.code

org    0100h

main:

mov     ah, 00h        ; Video Mode
```

```
mov    al, 03h        ; Set to 80x25
```

```
int     10h
```

```
..... Assign data to array  
.....
```

```
mov     bl, 0          ; initial column
```

```
mov     bp, offset cha ; *bp = cha[]
```

```
mov     si, offset row ; *si = row[]
```

```
mov     di, offset row_t; *di = row_t[]
```

```
mov     ah, 00h        ;random clock
```

```
int     1ah            ; CX:DX now hold clock ticks since midnight
```

```
mov     rnd_79, dx
```

```
mov     rnd_127, dx
```

```
add_data:
```

```
cmp     bl, 79          ;compare column
```

```
ja      end_add_data
```

```
rand0_79:
```

```
mov     ax, rnd_79
```

```
mov     dx, 0           ;clear dx
```

```
mov     cx, 80          ;set cx
```

```
div     cx              ;dx = ax mod cx
```

```

mov    al, dl        ;set al

mov    cx, var_a

mul    cx            ;ax = al*cx(var_a)

add    ax, var_b     ;ax = ax+var_b

mov    rnd_79, ax

```

end_rand0_79:

```

mov    byte ptr [si], 0        ; Random 0to79 to row[]

sub    byte ptr [si], dl

mov    al, tl                ; Random (0to79)-8 to row_t[]

mov    byte ptr [di], 0

sub    byte ptr [di], al

sub    byte ptr [di], dl

```

rand33_127:

```

mov    ax, rnd_127

mov    dx, 0                ;clear dx

mov    cx, 95                ;set cx

div    cx                    ;dx = ax mod cx

add    dx, 33

```



```

    mov     al, dl        ;set al

    mov     cx, var_a

    mul     cx            ;ax = al*cx(var_a)

    add     ax, var_b     ;ax = ax+var_b

    mov     rnd_127, ax

end_rand33_127:

    mov     byte ptr [bp], 0        ; Random 33-127 to cha[]

    add     byte ptr [bp], dl

    inc     bl                ;increase col

    inc     bp                ;increase character

    inc     si                ;increase row

    inc     di                ;increase row_t

    jmp     add_data

end_add_data:

.....; White Character

myloop:

    mov     dl, 0

```

```

        mov     bp, offset cha

        mov     si, offset row

start_col:

        cmp     dl, 79

        ja      end_col


        mov     ah, [si]

        cmp     ah, 0

        jge     start_row

        jmp     end_row


start_row:

        mov     ah, 02h           ; Move cursor

        mov     bh, 00h          ; Page Number

        mov     dh, [si]         ; Row

        ;mov     dl, col          ; Column

        int     10h


        mov     ah, 09h          ; Write Character

        mov     al, [bp]          ; Character

        mov     bh, 00h          ; Page number

```



```

        mov     si, 65000

delay:

        dec     si          ; Decrease 1

        nop                     ; Kill 1 clock

        cmp     si, 0        ; If(si == 0)

        je      end_delay    ; If yes, Break

        jmp     delay        ; If no, Continue delay

end_delay:

.....; Light Green Character

        mov     dl, 0

        mov     bp, offset cha

        mov     si, offset row

start_col2:

        cmp     dl, 79

        ja      end_col2


        mov     ah, [si]

        cmp     ah, 0

        jge     start_row2

        jmp     end_row2

```

start_row2:

mov ah, 02h ; Move cursor

mov bh, 00h ; Page Number

mov al, [si]

sub al, cl_1

mov dh, [si] ; Row

;mov dl, col ; Column

int 10h

mov ah, 09h ; Write Character

mov al, [bp] ; Character

mov bh, 00h ; Page number

mov bl, 0Ah ; Color

mov cx, 01h ; Time to print

int 10h

end_row2:

inc dl

inc bp

inc si

jmp start_col2

end_col2:

```
..... Green Character
```

```
mov     dl, 0
```

```
mov    bp, offset cha
```

```
mov     si, offset row
```

```
start_col3:
```

```
cmp    dl, 79
```

ja end_col3

```
mov     ah, [si]
```

```
cmp    ah, 0
```

```
jge    start_row3
```

```
jmp    end_row3
```

```
start_row3:
```

```
mov     ah, 02h        ; Move cursor
```

```
mov     bh, 00h        ; Page Number
```

```
mov    al, [si]
```

```
sub    al, cl_2
```

```
mov     dh, al          ; Row
```

```
;mov    dl, col      ; Column
```

```

        int     10h

        mov     ah, 09h           ; Write Character

        mov     al, [bp]         ; Character

        mov     bh, 00h         ; Page number

        mov     bl, 02h         ; Color

        mov     cx, 01h         ; Time to print

        int     10h

end_row3:

        inc     dl

        inc     bp

        inc     si

        jmp     start_col3

end_col3:

.....; Dark Grey Character
.....

        mov     dl, 0

        mov     bp, offset cha

        mov     si, offset row

start_col4:

        cmp     dl, 79

```

```

ja    end_col4

mov    ah, [si]

cmp    ah, 0

jge    start_row4

jmp    end_row4

```

start_row4:

```

mov    ah, 02h        ; Move cursor

mov    bh, 00h        ; Page Number

mov    al, [si]

sub    al, cl_3

mov    dh, al         ; Row

;mov    dl, col        ; Column

int    10h

mov    ah, 09h        ; Write Character

mov    al, [bp]        ; Character

mov    bh, 00h        ; Page number

mov    bl, 08h        ; Color

mov    cx, 01h        ; Time to print

```



```

        int    10h

end_row4:

        inc    dl

        inc    bp

        inc    si

        jmp    start_col4

end_col4:

.....; Clear Character
.....

        mov    dl, 0

        mov    di, offset row_t

start_col5:

        cmp    dl, 79

        ja     end_col5


        mov    ah, [di]

        cmp    ah, 0

        jge    start_row5

        jmp    end_row5

start_row5:

        mov    ah, 02h        ; Move cursor

```

```

mov     bh, 00h        ; Page Number

mov     dh, [di]        ; Row

;mov     dl, col        ; Column

int     10h

mov     ah, 09h          ; Write Character

mov     al, "            ; Character

mov     bh, 00h        ; Page number

mov     bl, 00h        ; Color

mov     cx, 01h        ; Time to print

int     10h

end_row5:

inc     dl

inc     di

jmp     start_col5

end_col5:

.....; Increase row

mov     bl, 0

mov     si, offset row

mov     di, offset cha

```

loop_forward:

 cmp bl, 79

 ja end_loop_forward

 inc byte ptr [si] ; row++

rand33_127_2:

 mov ax, rnd_127

 mov dx, 0 ;clear dx

 mov cx, 95 ;set cx

 div cx ;dx = ax mod cx

 add dx, 33

 mov al, dl ;set al

 mov cx, var_a

 mul cx ;ax = al*cx(var_a)

 add ax, var_b ;ax = ax+var_b

 mov rnd_127, ax

 add byte ptr [di], dl

end_rand33_127_2:

inc bl

inc si

inc di

jmp loop_forward

end_loop_forward:

.....; Increase row_t
.....;

mov ah, 0

mov si, offset row_t

loop_forward2:

cmp ah, 79

ja end_loop_forward2

inc byte ptr [si] ; row_t++

inc ah

inc si

jmp loop_forward2

end_loop_forward2:

.....; Check row
.....;

mov ah, 0

mov si, offset row

loop_forward3:

cmp ah, 79

ja end_loop_forward3

mov al, [si]

cmp al, 32 ; If(row == 32)

je clear_row1

jmp pass_row1

clear_row1:

mov byte ptr [si], 0

pass_row1:

inc ah

inc si

jmp loop_forward3

end_loop_forward3:

.....; Check row_t
.....;

```

        mov     ah, 0

        mov     si, offset row_t

loop_forward4:

        cmp     ah, 79

        ja      end_loop_forward4


        mov     al, [si]

        cmp     al, 32      ; If(row_t == 32)

        je      clear_row_t

        jmp     pass_row_t


clear_row_t:

        mov     al, tl

        mov     byte ptr [si], 0

pass_row_t:

        inc     ah

        inc     si

        jmp     loop_forward4

end_loop_forward4:

```

```

.....
,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,

```

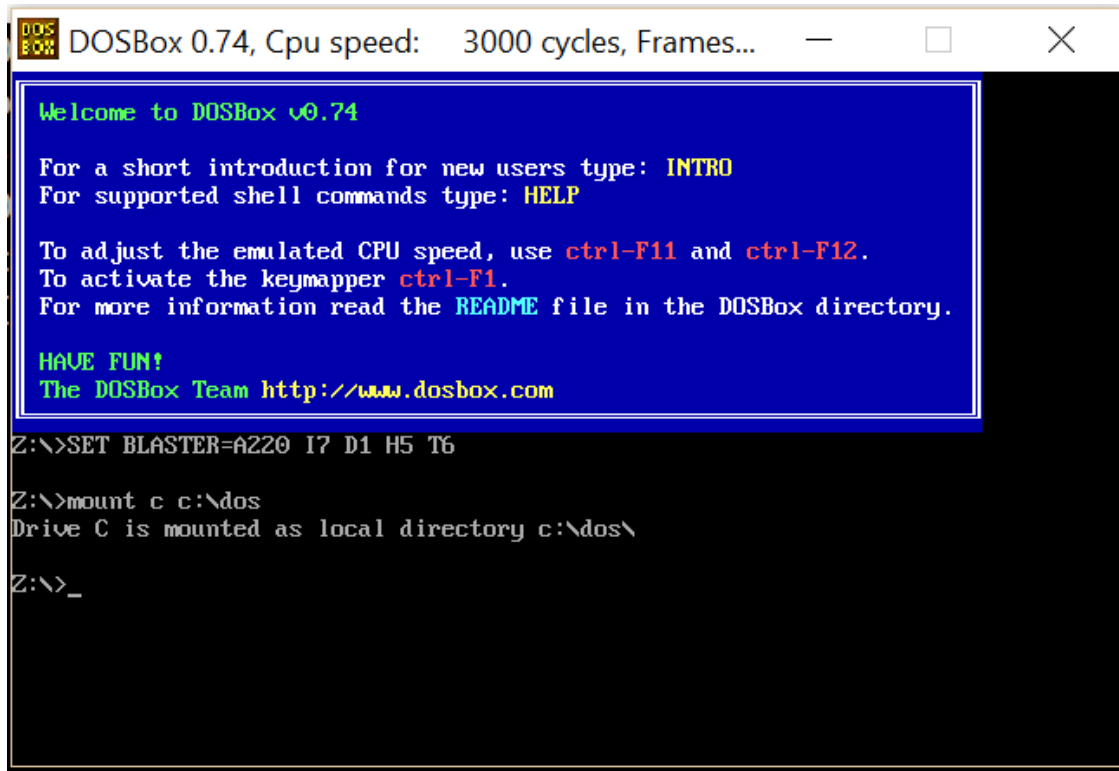
```
jmp    myloop    ; Repeat all statement
```

```
ret                ;return
```

```
end main          ;end program
```

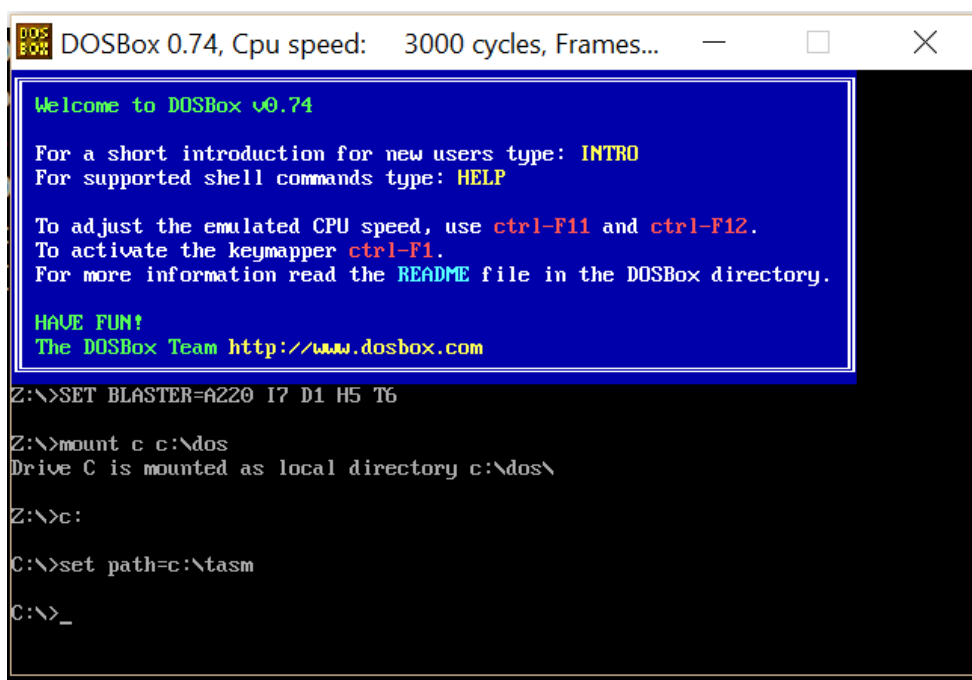
วิธีการคอมไพล์และสั่งรัน

1. เมื่อติดตั้ง โปรแกรม Dosbox 0.74 เสร็จแล้วให้ดับเบิลคลิกเข้าตัวโปรแกรมและ mount directory ไปยังไฟล์ที่ต้องการ



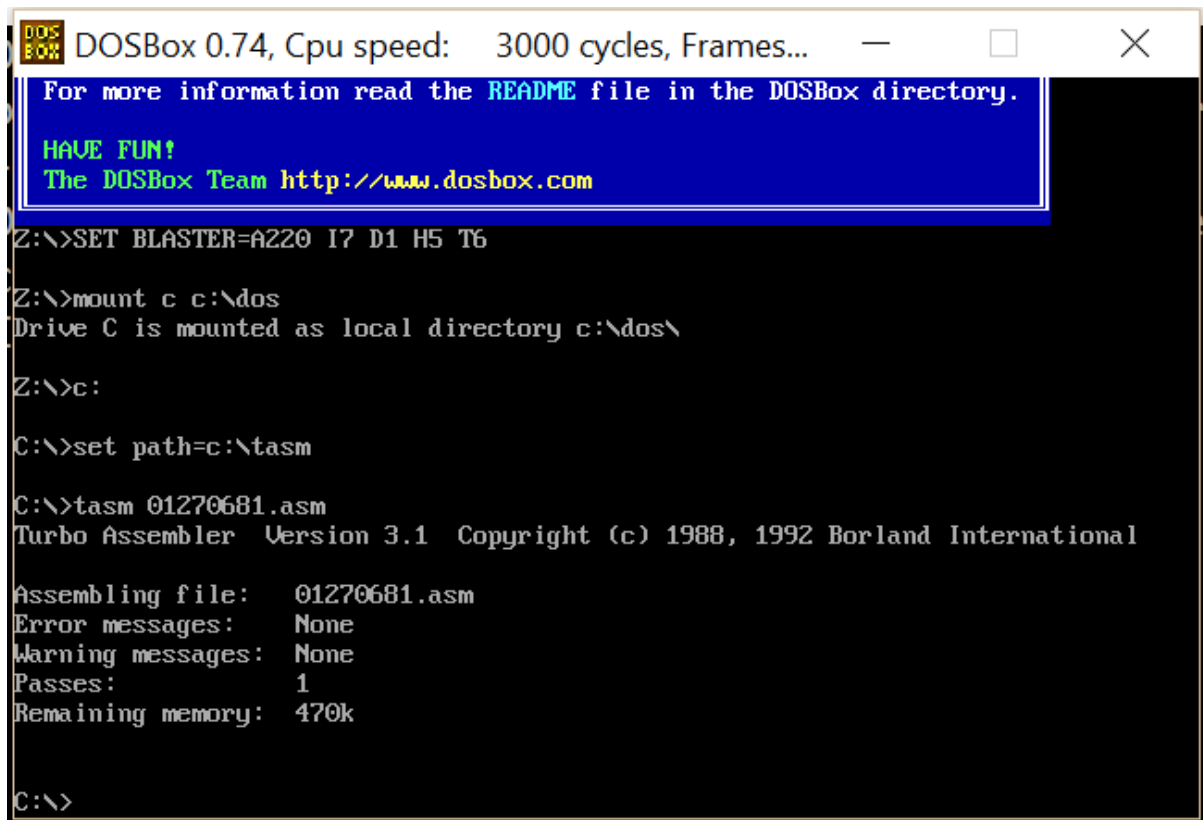
```
DOSBox 0.74, Cpu speed: 3000 cycles, Frames...  
Welcome to DOSBox v0.74  
For a short introduction for new users type: INTRO  
For supported shell commands type: HELP  
To adjust the emulated CPU speed, use ctrl-F11 and ctrl-F12.  
To activate the keymapper ctrl-F1.  
For more information read the README file in the DOSBox directory.  
HAVE FUN!  
The DOSBox Team http://www.dosbox.com  
Z:\>SET BLASTER=A220 I7 D1 H5 T6  
Z:\>mount c c:\dos  
Drive C is mounted as local directory c:\dos\  
Z:\>_
```

2. ไปยังไดรฟ์ที่เซต directory ไว้ และset pathไปที่ tasm



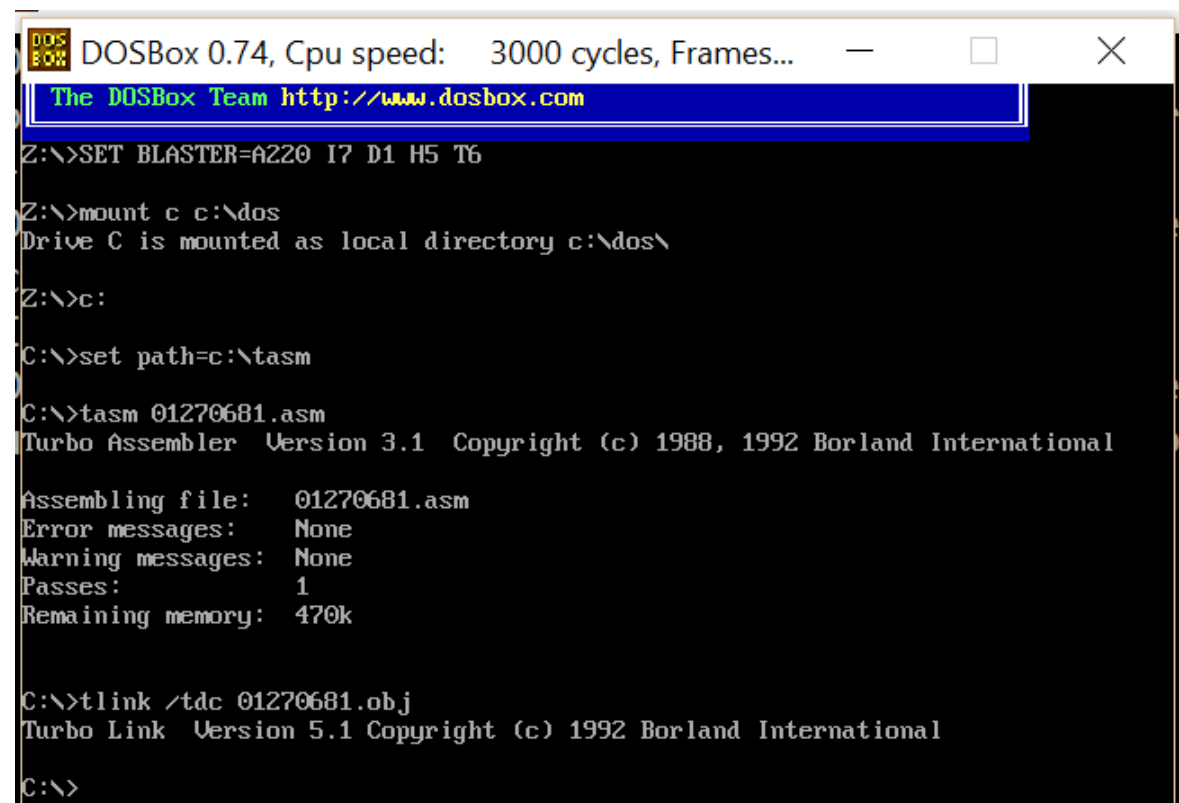
```
DOSBox 0.74, Cpu speed: 3000 cycles, Frames...  
Welcome to DOSBox v0.74  
For a short introduction for new users type: INTRO  
For supported shell commands type: HELP  
To adjust the emulated CPU speed, use ctrl-F11 and ctrl-F12.  
To activate the keymapper ctrl-F1.  
For more information read the README file in the DOSBox directory.  
HAVE FUN!  
The DOSBox Team http://www.dosbox.com  
Z:\>SET BLASTER=A220 I7 D1 H5 T6  
Z:\>mount c c:\dos  
Drive C is mounted as local directory c:\dos\  
Z:\>c:  
C:\>set path=c:\tasm  
C:\>_
```


3. พิมพ์ tasm ชื่อไฟล์.asm เพื่อสร้าง object ไฟล์



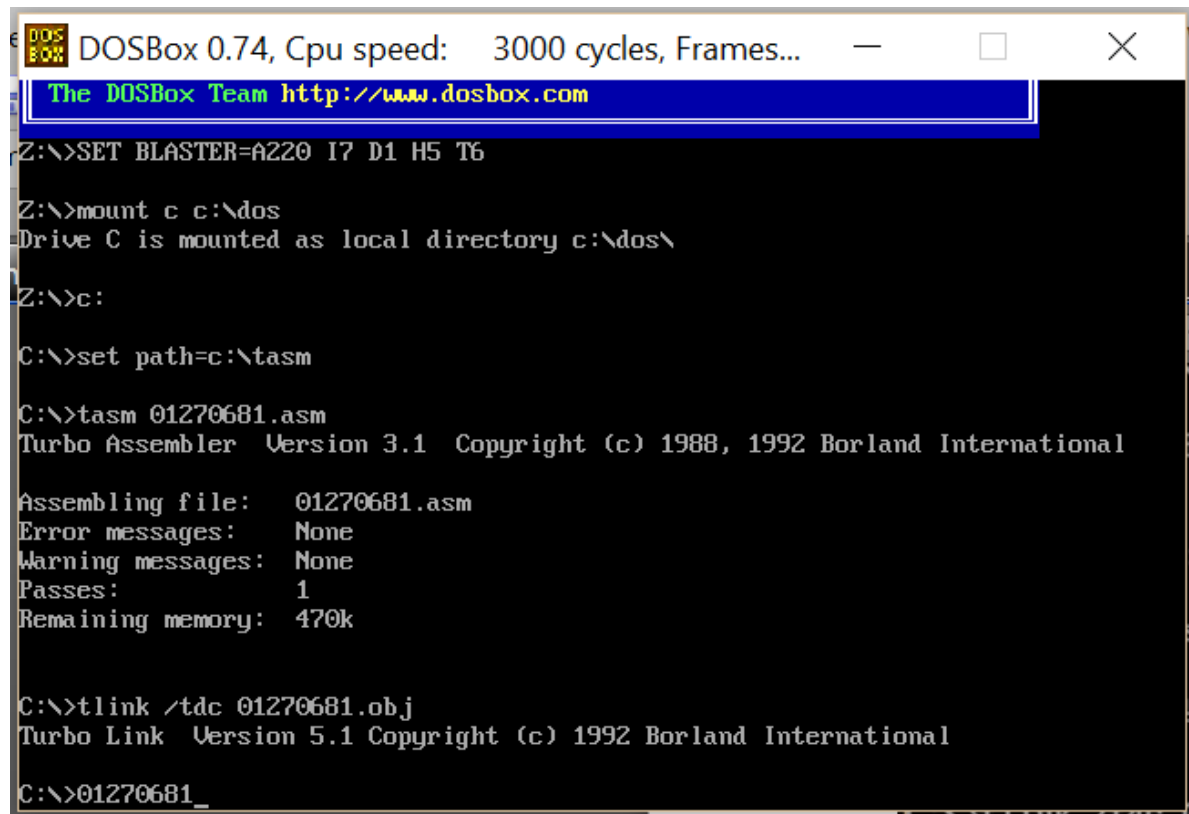
```
DOSBox 0.74, Cpu speed: 3000 cycles, Frames...  
For more information read the README file in the DOSBox directory.  
HAVE FUN!  
The DOSBox Team http://www.dosbox.com  
Z:\>SET BLASTER=A220 I7 D1 H5 T6  
Z:\>mount c c:\dos  
Drive C is mounted as local directory c:\dos\  
Z:\>c:  
C:\>set path=c:\tasm  
C:\>tasm 01270681.asm  
Turbo Assembler Version 3.1 Copyright (c) 1988, 1992 Borland International  
Assembling file: 01270681.asm  
Error messages: None  
Warning messages: None  
Passes: 1  
Remaining memory: 470k  
C:\>
```

4. พิมพ์ tlink /tdc ชื่อไฟล์.obj เพื่อแปลงเป็น Executable ไฟล์โดยโปรแกรม link



```
DOSBox 0.74, Cpu speed: 3000 cycles, Frames...  
The DOSBox Team http://www.dosbox.com  
Z:\>SET BLASTER=A220 I7 D1 H5 T6  
Z:\>mount c c:\dos  
Drive C is mounted as local directory c:\dos\  
Z:\>c:  
C:\>set path=c:\tasm  
C:\>tasm 01270681.asm  
Turbo Assembler Version 3.1 Copyright (c) 1988, 1992 Borland International  
Assembling file: 01270681.asm  
Error messages: None  
Warning messages: None  
Passes: 1  
Remaining memory: 470k  
C:\>tlink /tdc 01270681.obj  
Turbo Link Version 5.1 Copyright (c) 1992 Borland International  
C:\>
```

5. ทำการรันไฟล์โดยพิมพ์ชื่อไฟล์



```

DOSBox 0.74, Cpu speed: 3000 cycles, Frames...
The DOSBox Team http://www.dosbox.com
Z:\>SET BLASTER=A220 I7 D1 H5 T6

Z:\>mount c c:\dos
Drive C is mounted as local directory c:\dos\

Z:\>c:

C:\>set path=c:\tasm

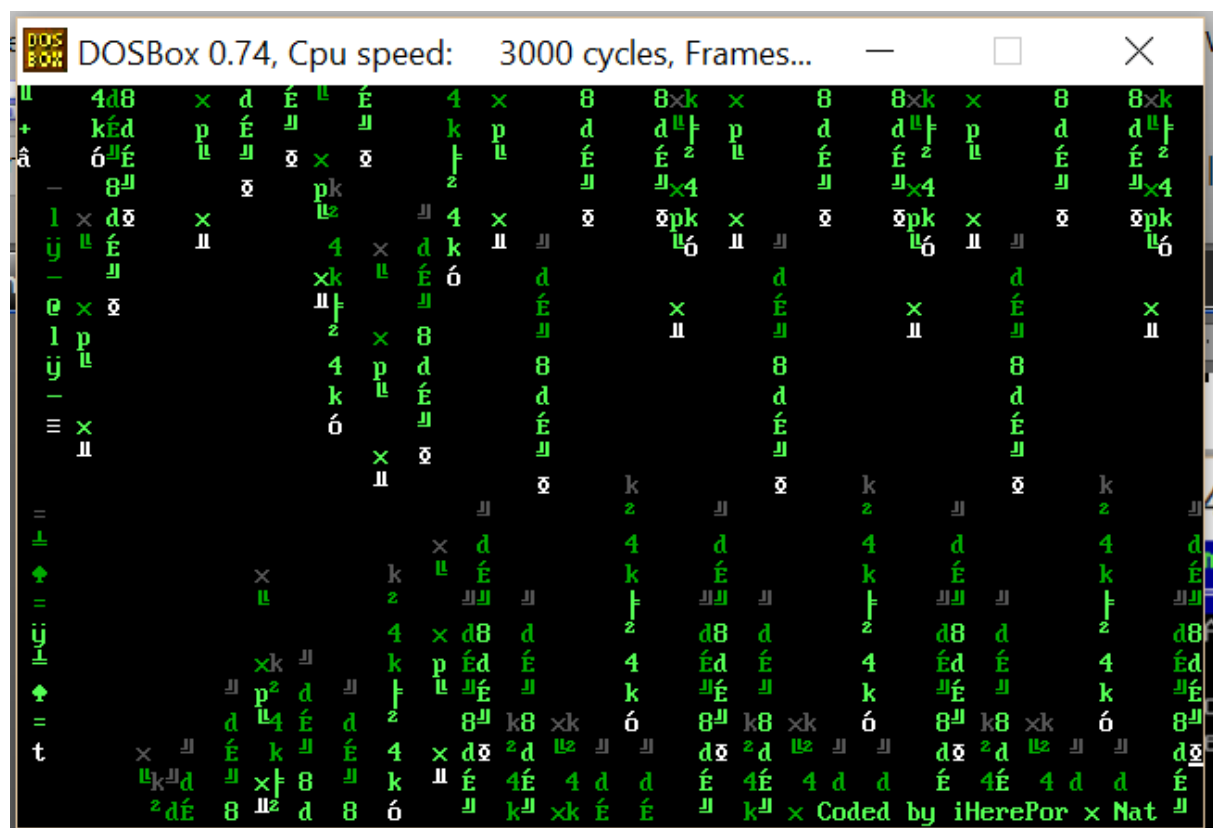
C:\>tasm 01270681.asm
Turbo Assembler Version 3.1 Copyright (c) 1988, 1992 Borland International

Assembling file: 01270681.asm
Error messages: None
Warning messages: None
Passes: 1
Remaining memory: 470k

C:\>tlink /tdc 01270681.obj
Turbo Link Version 5.1 Copyright (c) 1992 Borland International

C:\>01270681_
  
```

6. ผลลัพธ์การรันไฟล์



ลิงค์ผลการรันโปรแกรม

<https://youtu.be/MjtDVaQGY9A>