



ĐỀ TÀI: LẬP TRÌNH GAME TỰ CHỌN

LẬP TRÌNH GAME FLAPPY BIRD



NỘI DUNG TÓM TẮT

1. GIỚI THIỆU ĐỀ TÀI

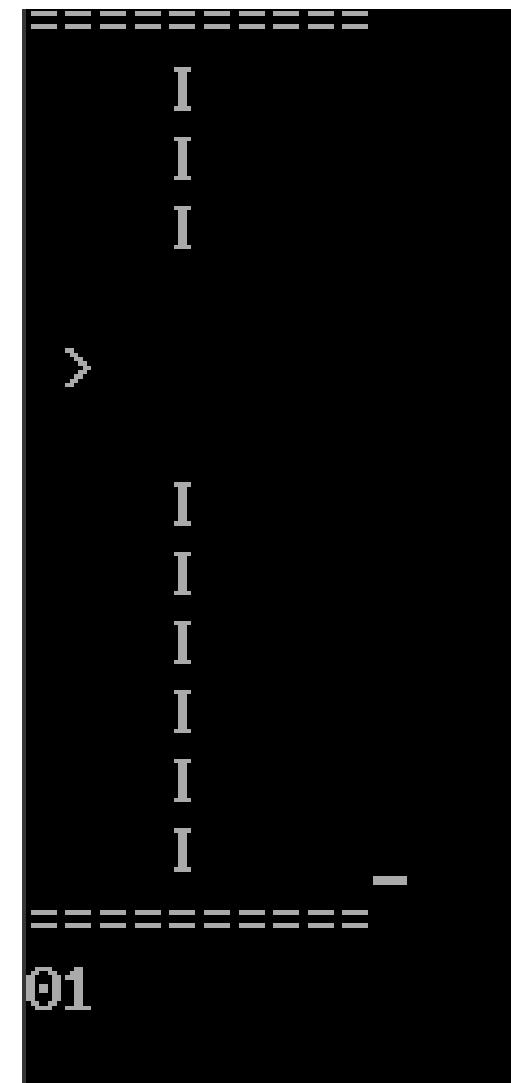
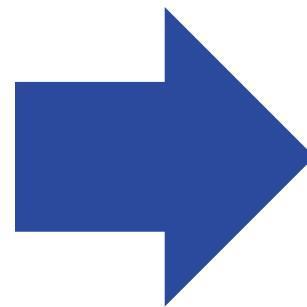
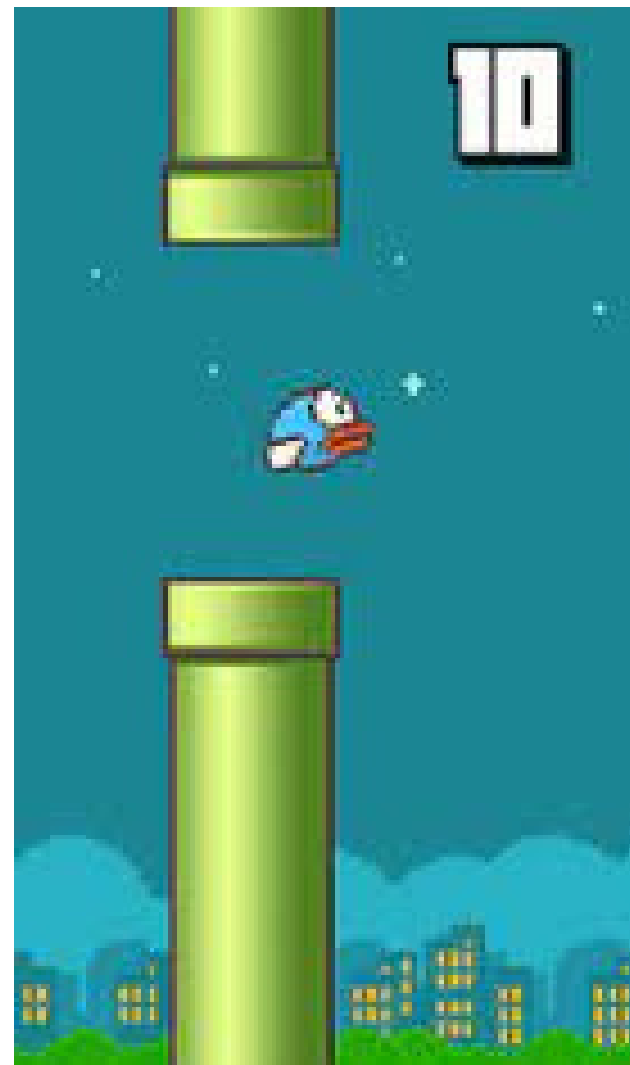
2. KIẾN THỨC SỬ
DỤNG

3. GIAO DIỆN
CHƯƠNG TRÌNH

4. DEMO VÀ GIẢI
THÍCH MÃ
NGUỒN

1. GIỚI THIỆU ĐỀ TÀI

- Mô phỏng lại tựa game nổi tiếng Flappy Bird trên các hệ điều hành DOS ngày xưa



2. KIẾN THỨC SỬ DỤNG

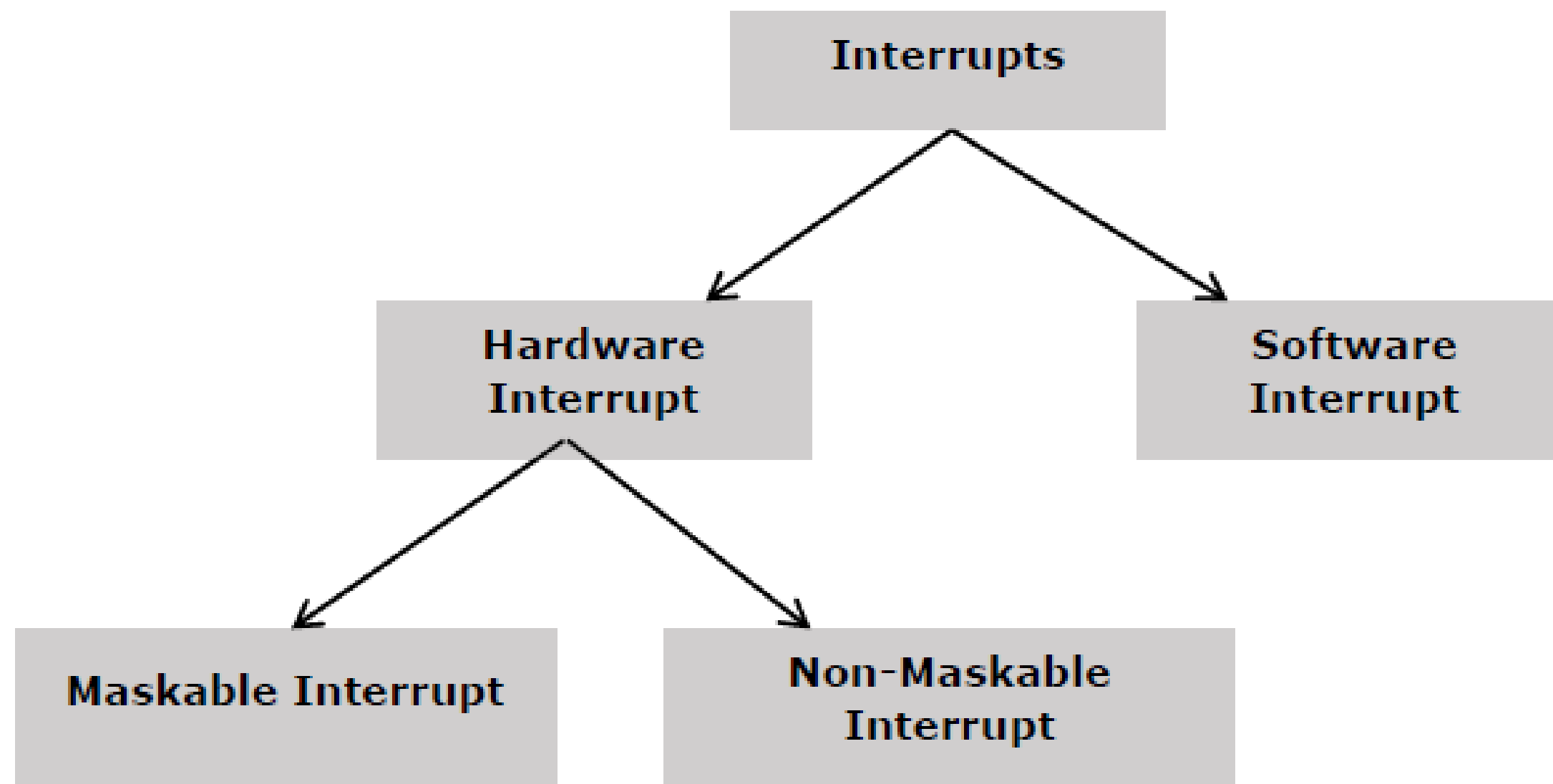
2.1 Ngắt và xử
lý ngắt trong
8086

2.2 Giới thiệu
về giả lập
DOSBox

2.1 NGẮT VÀ XỬ LÝ NGẮT TRONG 8086

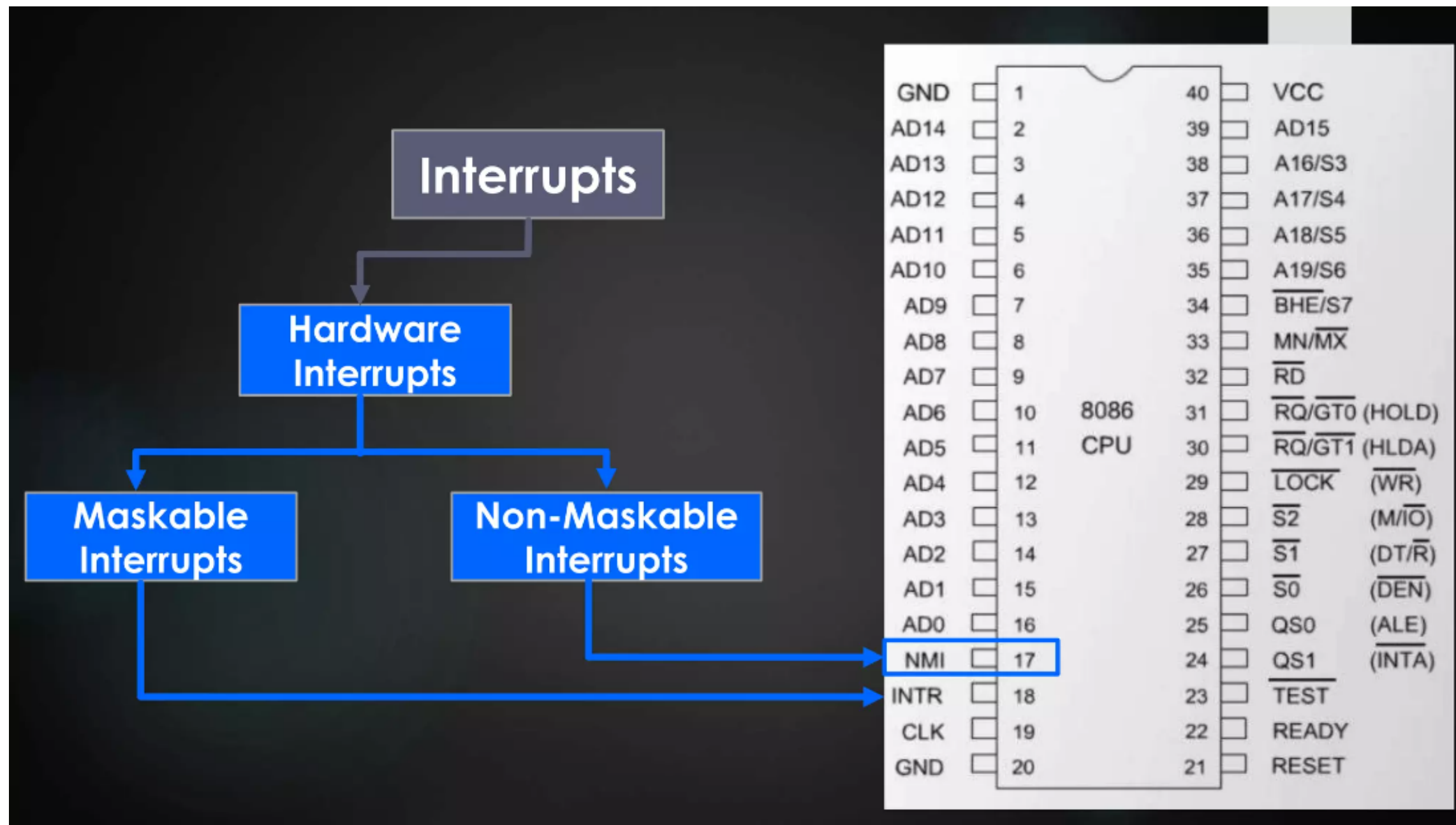
- Ngắt (Interrupt) có nghĩa là CPU đang thực hiện chương trình, ngắt tạm dừng chương trình đang chạy để chạy 1 chương trình khác.
- Sau khi hoàn thành việc trao đổi dữ liệu thì CPU lại phải quay về để làm tiếp công việc bị đang bị gián đoạn.

2.1 NGẮT VÀ XỬ LÝ NGẮT TRONG 8086



2.1 NGẮT VÀ XỬ LÝ NGẮT TRONG 8086

- Hardware Interrupts



2.1 NGẮT VÀ XỬ LÝ NGẮT TRONG 8086

- **Software Interrupts**

-Loại ngắt được tạo ra bởi phần mềm thông qua việc sử dụng các lệnh ngắt (interrupt instructions) trong mã máy của chương trình có dạng INT N, trong đó N là số hiệu (kiểu) ngắt nằm trong khoảng 00-FFH (0-255).

2.1 NGẮT VÀ XỬ LÝ NGẮT TRONG 8086

- Software Interrupts

Interrupt Jump Table

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F
80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F
A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF
B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF
C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF
D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF
E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF
F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF

2.1 NGẮT VÀ XỬ LÝ NGẮT TRONG 8086

• Software Interrupts

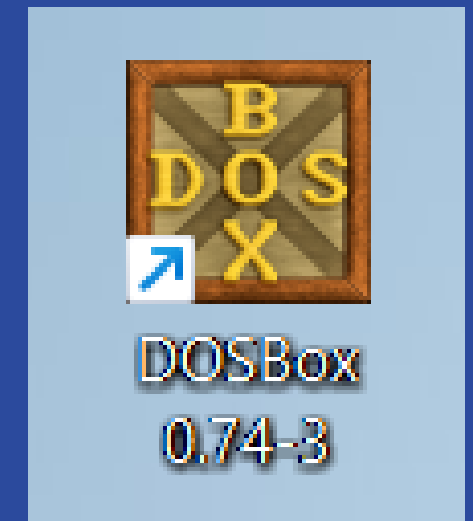
Int 10/AH=01h - VIDEO - SET TEXT-MODE CURSOR SHAPE
Int 10/AH=02h - VIDEO - SET CURSOR POSITION
Int 10/AH=03h - VIDEO - GET CURSOR POSITION AND SIZE
Int 10/AH=04h - VIDEO - READ LIGHT PEN POSITION (except VGA)
Int 10/AH=04h - HUNTER 16 - GET CURSOR ADDRESS
Int 10/AH=05h - VIDEO - SELECT ACTIVE DISPLAY PAGE
Int 10/AX=0500h - VIDEO - Corona/Cordata BIOS v4.10+ - SET GRAPHICS BITMAP BUFFER
Int 10/AX=050Fh - VIDEO - Corona/Cordata BIOS v4.10+ - GET GRAPHICS BITMAP BUFFER
Int 10/AX=0580h - VIDEO - PCjr, Tandy 1000 - GET CRT/CPU PAGE REGISTERS

Int 15/AH=85h - OS HOOK - SysReq KEY ACTIVITY (AT,PS)
Int 15/AX=8500h - HUNTER 16 - RESTORE POWER MENU (APM)
Int 15/AX=8501h - HUNTER 16 - SET POWER MENU (APM)
Int 15/AH=86h - BIOS - WAIT (AT,PS)
Int 15/AH=86h - HUNTER 16 - GET/SET SCREEN ATTRIBUTE TABLE



Int 16/AH=00h - KEYBOARD - GET KEYSTROKE
Int 16/AH=01h - KEYBOARD - CHECK FOR KEYSTROKE
Int 16/AH=02h - KEYBOARD - GET SHIFT FLAGS
Int 16/AH=03h - KEYBOARD - SET TYPEMATIC RATE AND DELAY
Int 16/AH=04h - KEYBOARD - SET KEYCLICK (PCjr only)

2.2 GIỚI THIỆU VỀ GIẢ LẬP DOSBOX

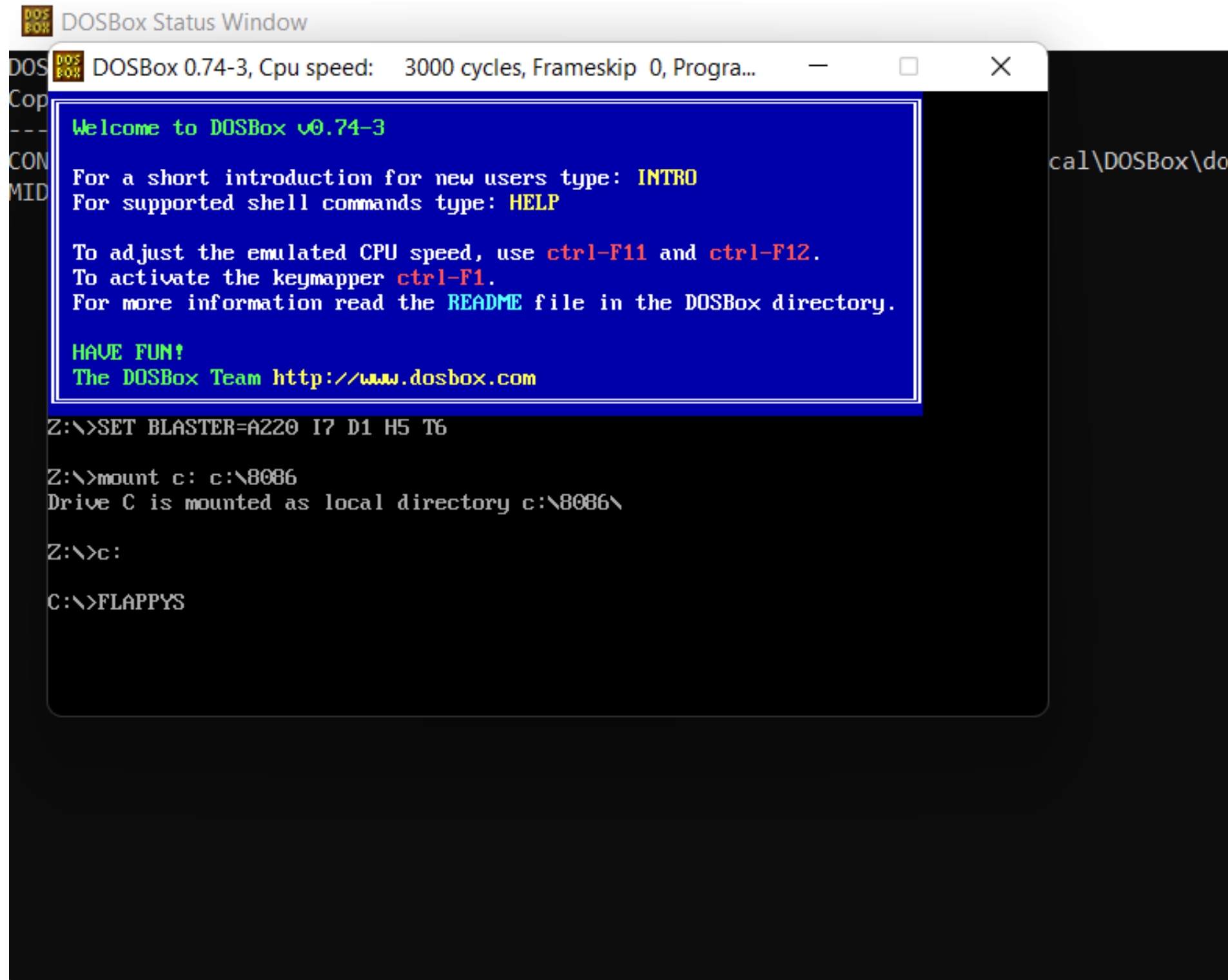
- DOSBox là một trình giả lập máy tính dựa trên DOS (Disk Operating System) được phát triển để chạy các ứng dụng và trò chơi DOS trên các hệ điều hành hiện đại.
- Do hạn chế của trình biên dịch trên emu8086 không hỗ trợ một số các ngắt và một số mã ASM đặc biệt nên nhóm em đã sử dụng giả lập DOSBox là môi trường giả lập DOS chính xác nhất hỗ trợ đầy đủ các hàm ngắt và tính năng nhóm em sử dụng.



2.2 GIỚI THIỆU VỀ GIẢ LẬP DOSBOX

 FLAPPY.ASM	7/27/2023 10:16 PM	ASM File	8 KB
 FLAPPY	7/20/2023 4:27 PM	Application	1 KB
 FLAPPY.OBJ	7/20/2023 4:26 PM	OBJ File	1 KB

2.2 GIỚI THIỆU VỀ GIẢ LẬP DOSBOX




The image shows a DOSBox Status Window and a command prompt. The status window displays a welcome message and instructions for new users. The command prompt shows the execution of several commands to set up the environment.

```
DOSBox Status Window
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...
Welcome to DOSBox v0.74-3
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:\8086
Drive C is mounted as local directory c:\8086\
Z:\>c:
C:\>FLAPPYS
```

3.GIAO DIỆN CHƯƠNG TRÌNH



```
DOSBox 0.74-3, Cpu s
=====
      I
      I
      I
>
      I
      I
      I
      I
      I
=====
01
```



```
DOSBox 0.74-3, Cpu
=====
>      I
      I
      I
      I
      I
      I
      I
      I
      I
=====
03
```

4

DEMO VÀ GIẢI THÍCH MÃ NGUỒN

4.Demo và giải thích mã nguồn

- **Ý tưởng:** Xây dựng một vòng lặp xử lý liên tục các sự kiện khi đang diễn ra trò chơi như in ra nhân vật, in ra ống,...Khi sự kiện va chạm xảy ra (collision) thì sẽ xây dựng một nhãn `game_over` cho vòng lặp của game kết thúc.

4.1 THIẾT LẬP ĐỘ PHÂN GIẢI GAME

```
code segment  
start:
```

```
    push ax  
    mov ax, data  
    mov ds, ax  
    pop ax
```

```
    mov ah, 00h  
    mov al, 03h  
    int 10h
```

-80x25 (Độ phân giải của cửa sổ game) với 16 màu và 8 trang.

4.2 KHỞI TẠO CÁC BIẾN

```
; data segment
data segment
    assume ds:data
    score_0 db 0
    score_1 db 0
    bird_x db 2
    bird_y db 0
    pipe_x db 11
    pipe_y db 0
    cursor_x db 0
    cursor_y db 0
data ends
```

- **Score_0**: Điểm của game ở hàng đơn vị
- **Score_1**: Điểm của game ở hàng chục
- **Bird_x**: Vị trí của nhân vật theo trục hoành
- **Bird_y**: Vị trí của nhân vật theo trục tung
- **Pipe_x**: Vị trí của ống theo trục hoành
- **Pipe_y**: Vị trí của ống theo trục tung
- **Cursor_x, cursor_y**: Tạo 2 biến con trỏ cho mục đích render

4.3 GIẢI THÍCH VÒNG LẶP GAME

- Vòng lặp được gán nhãn ***game_loop*** xử lý 2 công việc chính là: Xử lý logic của game và thực hiện việc render (In ra các thành phần của game lên màn hình console).

4.3 GIẢI THÍCH VÒNG LẶP GAME

a) Logic ống xuất hiện liên tục

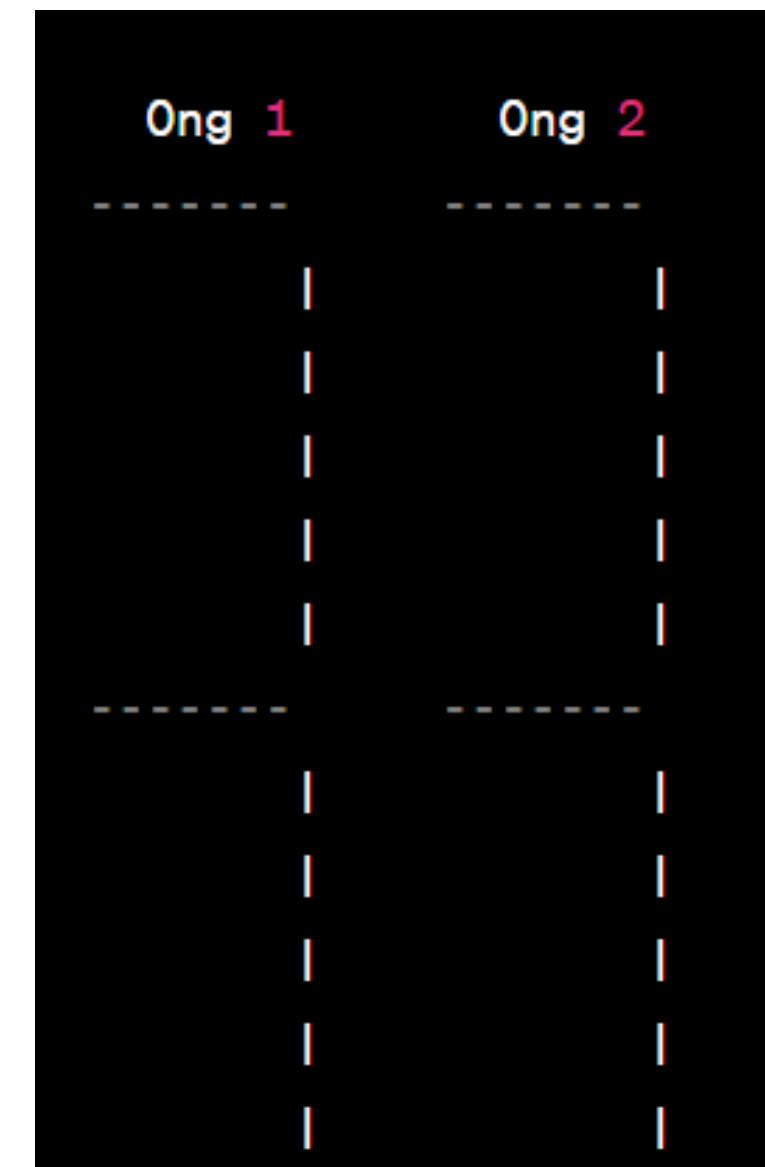
```
game_loop:
    ; reset ong
    cmp pipe_x, 0
    ja pipe_x_gt_zero

    mov pipe_x, 11

    ; xu ly diem so
    inc score_0
    cmp score_0, 9
    jbe dont_increment_tens
    mov score_0, 0
    inc score_1
    ; set lai vi tri cua ong khi vuot vao tao khe giua ong
    dont_increment_tens:

    cmp pipe_y, 5
    jb pipe_y_gt_5
    mov pipe_y, 0
    jmp pipe_x_gt_zero
    pipe_y_gt_5:
    inc pipe_y

    pipe_x_gt_zero:
    ; Giam vi tri cua ong lenh di 1 tu phai qua trai
    dec pipe_x
```



4.3 GIẢI THÍCH VÒNG LẶP GAME

b) Xử lý sự kiện nhấn phím

```
check_key:
    ; sleep for 250ms (250000 microseconds -> 3D090h)
    mov cx, 03h
    mov dx, 05150h
    mov ah, 86h
    int 15h

    ; Neu khong co du lieu trong bo dem ban phim
    mov ah, 0Bh
    int 21h
    cmp al, 0h
    je is_new_key_false

is_new_key_true:
    ; check xem co phim o bo dem ban phim khoh
    mov ah, 0
    int 16h

    ; Neu la phim escs
    cmp ah, 01h
    jne not_exit
    mov ah, 4ch
    int 21h
not_exit:

    ; giam vi tri nhan vat di 2
    sub bird_y, 2

    ; dat lai bo dem ban phim
    mov ah, 0Ch
    int 21h

is_new_key_false:
    ; Tang vi tri truc tung cua nhan vat
    inc bird_y
```

4.3 GIẢI THÍCH VÒNG LẶP GAME

c) Xử lý sự kiện va chạm (collision)

```
check_collision:
    ; collision with ground
    mov ah, 13
    mov bh, [bird_y]
    cmp ah, bh
    jb render
    ; check nếu nhận vật ở giữa ống
    mov ah, [pipe_x]
    mov bh, [bird_x]
    cmp ah, bh
    jne render
    mov ah, [bird_y]
    mov bh, [pipe_y]
    add bh, 4
    cmp ah, bh
    je render
    inc bh
    cmp ah, bh
    je render
    sub bh, 2
    cmp ah, bh
    je render

game_over:
    mov ah, 4ch
    int 21h
```

4.3 GIẢI THÍCH VÒNG LẶP GAME

d) Xử lý sự công việc render (In các thành phần của game lên màn hình)

- Render phần nền trên nền dưới

```
render:
    ; ve nen duoi
    mov dl, 10
    mov ah, 06h
    int 21h
    mov dl, 13
    int 21h
    mov cx, 10
draw_grass_bottom:
    dec cx
    mov dl, '='
    int 21h
    jnz draw_grass_bottom

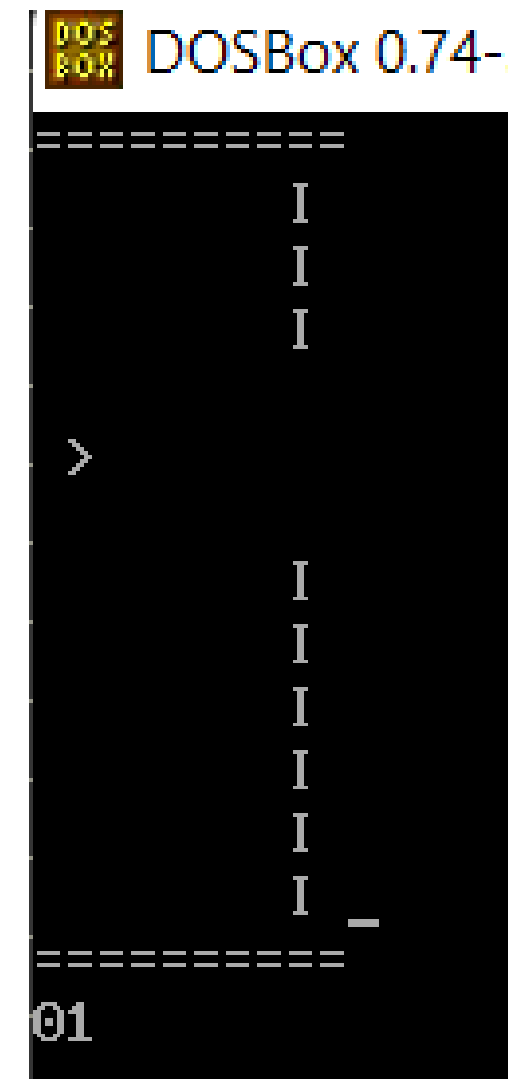
    ; Tao dong moi
    mov dl, 10
    mov ah, 06h
    int 21h
    mov dl, 13
    int 21h

    ; print score
    mov dl, score_1
    add dl, '0'
    mov ah, 06h
    int 21h
    mov dl, score_0
    add dl, '0'
    int 21h

    ;cursor at x=dl=0, y=dh=0
    mov dl, 00h
    mov dh, 00h
    mov ah, 2h
    mov bh, 0
    int 10h

    ; Ve nen tren
    mov cx, 10
draw_grass_top:
    dec cx
    mov dl, '='
    int 21h
    jnz draw_grass_top

    mov dh, 00h
    int 10h
```



4.3 GIẢI THÍCH VÒNG LẶP GAME

e) In các thành phần còn lại lên màn hình

```
mov cursor_y, 0
render_y:
inc cursor_y

mov dl, 10
mov ah, 06h
int 21h
mov dl, 13
int 21h

mov cursor_x, 0
render_x:
inc cursor_x

; render bird
render_1:
mov ah, [cursor_x]
mov bh, [bird_x]
cmp ah, bh
jne render_2
mov ah, [cursor_y]
mov bh, [bird_y]
cmp ah, bh
jne render_2
mov dl, '>'
mov ah, 06h
int 21h
jmp render_x_end

; render pipe
render_2:
mov ah, [cursor_x]
mov bh, [pipe_x]
cmp ah, bh
jne render_3
mov ah, [cursor_y]
mov bh, [pipe_y]
add bh, 4
cmp ah, bh
je render_3
inc bh
cmp ah, bh
je render_3
sub bh, 2
cmp ah, bh
je render_3
mov dl, 'I'
mov ah, 06h
int 21h
jmp render_x_end

; render empty space
render_3:
mov dl, ' '
mov ah, 06h
int 21h
jmp render_x_end

render_x_end:
cmp cursor_x, 10
jb render_x

render_y_end:
cmp cursor_y, 12
jb render_y
```



**THANKS FOR
WATCHING!**

