Laboratory Exercise #3-5
Mouse and Keyboard Programming

Target Course Outcome:
CO2: Designs a microprocessor-based firmware integrating a microprocessor with supporting peripherals and devices.

Objectives:
To understand the concept of subroutines and use it in assembly language programs.
To know the different DOS interrupts and use such interrupts in assembly language programs.

Tools Required:
Emu8086 Emulator

Part 1:

This exercise will involve both mouse and keyboard programming. Keyboard programming will be discussed first, and this entails, a discussion of the programming interface to the keyboard of your PC. Like the previous BIOS and DOS interrupts, keyboard programming is achieved by the use of BIOS Interrupt 16H explained below.

Interrupt 16H Option 0H: Keyboard Read.

Result registers:
AH – Key scan code.
AL – ASCII character.

Note: Reads and removes one character from the keyboard buffer. If there isn't one, it waits until there is.

Interrupt 16H Option 1H: Get keyboard status.

Result registers and flags:

ZF = 1 if no key is waiting
ZF = 0 if key is waiting
AH – Key scan code
AL – ASCII character

Note: Checks to see if there is a key waiting in the keyboard buffer. If a key press is waiting in the keyboard buffer, then ZF=0, and the codes are returned in AH and AL respectively. This function works like function 0 except the character is not removed from the keyboard buffer.

Interrupt 16H Option 2H: Get keyboard status bytes.

Result registers and flags: • AL – ASCII character.

- D0 – Right Shift pressed.
- D1 Left shift pressed.
- D2 – Ctrl pressed.
- D3 – Alt pressed.
- D4 – Scroll Lock state toggled.
- D5 – NumLock state toggled.
- D6 – CapsLock state toggled.
- D7 – Insert toggled.

The table of scan codes is given below. Notice that many keys that do not have an ASCII equivalent may be programmed using scan codes. Some examples: F1, F2, PgDn, NumLock, etc.

Scan codes

Hex	Key	Hex	Key	Hex	Key	Hex	Key
01	Esc	17	I and i	2D	X and x	43	F9
02	! and 1	18	O and o	2E	C and c	44	F10
03	@ and 2	19	P and p	2F	V and v	45	NumLock
04	# and 3	1A	{ ans [	30	B and b	46	ScrollLock
05	$ and 4	1B	} and]	31	N and n	47	7 and Home
06	% and 5	1C	Enter	32	M and m	48	8 and ↑
07	^ and 6	1D	Ctrl	33	< and ,	49	9 and PgUp
08	& and 7	1E	A and a	34	> and .	4A	- (keypad)
09	* and 8	1F	S and s	35	? and /	4B	4 and ←
0A	(and 9	20	D and d	36	Right shift	4C	5 (keypad)
0B	) and 0	21	F and f	37	PrtSc and *	4D	6 and →
0C	- and _	22	G and g	38	Alt	4E	+ (keypad)
0D	+ and =	23	H and h	39	Spacebar	4F	1 and End
0E	Backspace	24	J and j	3A	CapsLock	50	2 and ↓
0F	Tab	25	K and k	3B	F1	51	3 and PgDn
10	Q and q	26	L and l	3C	F2	52	0 and Ins
11	W and w	27	: and ;	3D	F3	53	. and Del
12	E and e	28	“ and ‘	3E	F4		
13	R and r	29	~ and `	3F	F5		
14	T and t	2A	Left shift	40	F6		
15	Y and y	2B	| and \	41	F7		
16	U and u	2C	Z and z	42	F8		

The BIOS and DOS interrupt use register AH to pass information regarding which function is to be performed by the interrupt, but INT 33H (mouse interrupt) uses the AX register for that purpose.

In graphics mode the mouse pointer will be shown as an arrow; in text mode the mouse pointer is shown as a blinking rectangular block. Mouse sensitivity is measured in *mickeys*. This unit associates the movement of the cursor on the screen with the movement of the mouse on the pad. For example: a mouse which moves the cursor 400 pixels for every 1 inch of mouse movement has a sensitivity of 400 mickeys.

INT 33H Function 0H: Detect the presence of a mouse

AX = 0H

Upon return:

AX = 0 – No mouse is supported

AX > 0 – Mouse is supported

Example:

```
MOV AX,0
INT 33H
CMP AX,0
JE Exit
…
…
```

INT 33H Function 1H: Displays the mouse cursor

AX = 1H

INT 33H Function 2H: Hides the mouse cursor

AX = 2H

INT 33H Function 3H: Gets current mouse cursor position

AX = 3H

Upon return:
CX = Horizontal coordinate in pixels
DX = Vertical coordinate in pixels

In text mode, one needs to divide these coordinates by 8 to get the character location. This is since each character is made with an 8X8 pixel matrix.

INT 33H Function 4H: Sets current mouse cursor position

AX = 4H
CX = Horizontal coordinate in pixels
DX = Vertical coordinate in pixels

In text mode, one needs to divide these coordinates by 8 to get the character location. This is since each character is made with an 8X8 pixel matrix.

INT 33H Function 5H: Gets mouse button press information

AX = 5H
BX = 0 for left button, 1 for right button, 2 for center button;

Upon return:

AX = button status, where:
D0 = Left button: 1 = Down, 0 = Up;
D1 = Right button: 1 = Down, 0 = Up;
D2 = Center button; 1 = Down, 0 = Up

BX = Button press count since the last call to this function.
CX = Horizontal coordinate in pixels at the last button press.
DX = Vertical coordinate in pixels at the last button press.

INT 33H Function 6H: Gets mousebutton release information

AX = 6H
BX = 0 for left button, 1 for right button, 2 for center button;

Upon return:

AX = button status, where:
D0 = Left button: 1 = Down, 0 = Up;
D1 = Right button: 1 = Down, 0 = Up;
D2 = Center button; 1 = Down, 0 = Up
BX = Button release count since the last call to this function.
CX = Horizontal coordinate in pixels at the last button press.
DX = Vertical coordinate in pixels at the last button press.

INT 33H Function 7H: Sets horizontal for the mouse pointer

AX = 7H
CX = Minimum x coordinate in pixels.
DX = Maximum x coordinate in pixels.

INT 33H Function 8H: Sets horizontal for the mouse pointer

AX = 8H
CX = Minimum x coordinate in pixels.
DX = Maximum x coordinate in pixels.

INT 33H Function 10H: Sets an exclusion area for the mouse pointer

AX = 10H

CX = Upper horizontal coordinate in pixels.

DX = Upper vertical coordinate in pixels.

SI = Lower horizontal coordinate in pixels.

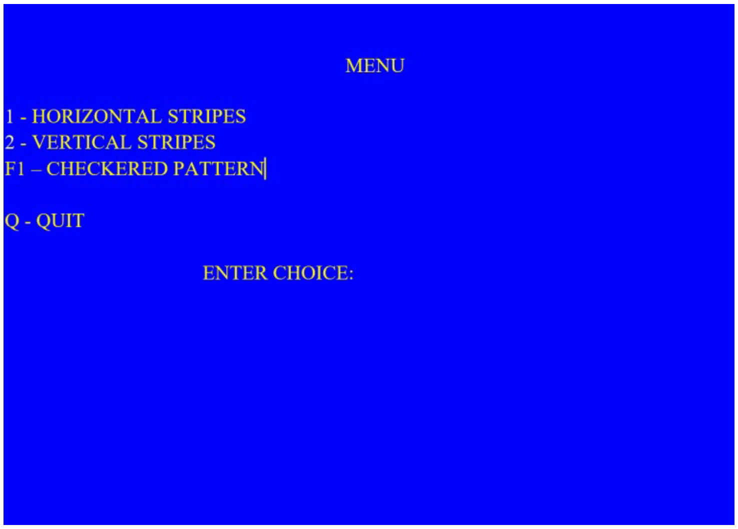
DI = Lower vertical coordinate in pixels.

Part II

Application of mouse and keyboard programming

Activity #1

Modify the program written for the previous BIOS Interrupt experiment (Exercise 3-4), to allow for the use of mouse and keyboard input. In this experiment the menu of choices will be increased by a new pattern, and the choices of pattern may be inputted by a keyboard press as before or by using the mouse to select the option by left clicking over the menu choice on the screen. Refer to the figure below:



```
                              MENU

1 - HORIZONTAL STRIPES
2 - VERTICAL STRIPES
F1 – CHECKERED PATTERN

Q - QUIT

                    ENTER CHOICE:
```

Copyright Information

Author	: Rosana J. Ferolin (rjferolin@usc.edu.ph)
Date of release	: August 7, 2020
Version	: 1.0

Change log:

Date	Version	Author	Changes
Aug. 7, 2020	1.0	Rosana J. Ferolin	Initial Draft
Sep. 3, 2025	2.0	Marlowe Edgar C. Burce	Revised activity and procedures