Experiment 10

Code:

```
.model small

.stack 100h

.data

.code

start:

   mov ax, @data

   mov ds, ax


   ; Generate a beep sound at ~1kHz frequency for 500ms

   call beep_1000hz_500ms


   ; Wait for a key press before exit

   mov ah, 00h

   int 16h


   ; Exit program

   mov ah, 4Ch

   int 21h


; ----------------------------------------

; beep_1000hz_500ms: Beep at ~1kHz for 500ms

; ----------------------------------------

beep_1000hz_500ms proc

   ; Calculate divisor for 1kHz

   ; Divisor = 1193180 / frequency

   ; 1193180 / 1000 = 1193 approx
```

```
  mov bx, 1193           ; Divisor for 1kHz

  ; Program PIT Channel 2:
  mov al, 0B6h           ; Command byte: channel 2, lobyte/hibyte, mode 3 (square wave)
  out 43h, al

  ; Send low byte of divisor
  mov al, bl
  out 42h, al

  ; Send high byte of divisor
  mov al, bh
  out 42h, al

  ; Enable speaker by setting bits 0 and 1 of port 61h
  in al, 61h
  or al, 03h             ; Set bits 0 and 1
  out 61h, al

  ; Delay ~500ms (simple loop)
  mov cx, 5000
delay_loop:
  push cx
  mov cx, 65535
inner_delay:
  loop inner_delay
  pop cx
  loop delay_loop
```

```
    ; Disable speaker (clear bits 0 and 1)
    in al, 61h
    and al, 0FCh
    out 61h, al

    ret
beep_1000hz_500ms endp

end start
```

Explanation:

HEADER SECTION

Line 1: .model small

- Defines the memory model as small:
    - Code and data segments are each no larger than 64KB.
    - Common for simple DOS programs.

Line 2: .stack 100h

- Allocates 256 bytes (100h) of stack space.

DATA SEGMENT

Line 3: .data

- Starts the data segment.
- No variables are defined here, but it's required for .model small.

CODE SEGMENT

Line 4: .code

- Begins the code segment.

PROGRAM START

Line 5: start:

- This is the entry point of the program.

Line 6: mov ax, @data

- Loads the address of the data segment into AX.

Line 7: mov ds, ax

- Sets the DS (data segment register) to point to the data segment.

GENERATE BEEP SOUND

Line 9: call beep_1000hz_500ms

- Calls a procedure (defined later) to generate a 1kHz beep for 500ms.

WAIT FOR KEY PRESS

Line 11: mov ah, 00h

- Prepares for BIOS interrupt 16h (keyboard input), function 00h: Wait for key press.

Line 12: int 16h

- Calls the BIOS interrupt to wait for the user to press any key.

EXIT PROGRAM

Line 14: mov ah, 4Ch

- Prepares for DOS interrupt 21h, function 4Ch: Terminate the program.

Line 15: int 21h

- Exits to DOS and returns control to the OS.

PROCEDURE: beep_1000hz_500ms

This section handles generating a beep sound at ~1kHz using the PC speaker.

Line 18: beep_1000hz_500ms proc

- Begins the procedure definition.

PIT (Programmable Interval Timer) configuration

Line 20: mov bx, 1193

- 1193180 / 1000 ≈ 1193
  - 1193180 Hz is the clock frequency of the PIT.
  - We want 1000Hz = 1kHz => divisor ≈ 1193.
  - This value is used to set the timer frequency.

Line 22: mov al, 0B6h

- Prepares command byte for PIT control:
  - 0xB6 = 10110110b:
    - Bits 7-6: 10 → Channel 2
    - Bits 5-4: 11 → Access mode: Lobyte/Hibyte
    - Bits 3-1: 110 → Mode 3: Square Wave Generator
    - Bit 0: 0 → Binary mode

Line 23: out 43h, al

- Sends the command byte to port 43h, which is the PIT control port.

Send the 16-bit divisor (1193) to channel 2

We must send the divisor as two bytes: first the low byte, then the high byte.

Line 25: mov al, bl

- Load low byte of divisor into AL.

Line 26: out 42h, al

- Send low byte to PIT channel 2 (port 42h).

Line 28: mov al, bh

- Load high byte of divisor into AL.

Line 29: out 42h, al

- Send high byte to PIT channel 2 (port 42h).

Enable PC speaker

To turn on the speaker, you must set bits 0 and 1 of port 61h:

- Bit 0: Speaker gate
- Bit 1: Timer 2 output

Line 31: in al, 61h

- Read current value of port 61h into AL.

Line 32: or al, 03h

- Set bits 0 and 1 to 1 using OR.

Line 33: out 61h, al

- Output the modified value back to port 61h, enabling the speaker.

Delay loop for 500ms

Line 35: mov cx, 5000

- Outer loop counter. Approximates ~500ms (not exact; just a crude delay).

Line 36: delay_loop:

- Label for the outer loop.

Line 37: push cx

- Save outer loop counter on the stack.

Line 38: mov cx, 65535

- Inner delay loop (long iteration count).

Line 39: inner_delay:

- Label for the inner loop.

Line 40: loop inner_delay

- Decrement CX; repeat until 0.

Line 41: pop cx

- Restore the outer loop counter from the stack.

Line 42: loop delay_loop

- Repeat the outer loop 5000 times.

This is a crude busy-wait delay. Its actual duration will vary by CPU speed.

Disable PC speaker

Line 44: in al, 61h

- Read current speaker control port (61h).

Line 45: and al, 0FCh

- Clear bits 0 and 1 to turn off speaker:
    - 0FCh = 11111100b

Line 46: out 61h, al

- Write the updated value back to port 61h to disable the beep.

Line 48: ret

- Return from the procedure beep_1000hz_500ms.

Line 49: beep_1000hz_500ms endp

- End of procedure definition.

Line 51: end start

- Marks the end of the program, and tells the assembler to start execution from the start: label.