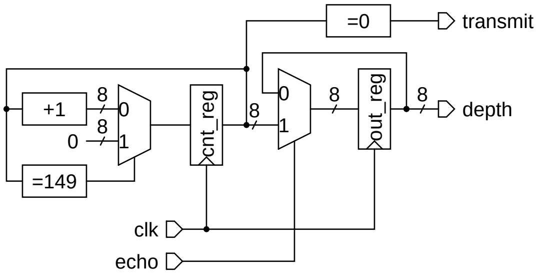
Solutions for Mid-Term Exam

Question 1

The solution consists of two registers: one to implement a counter and one to load and hold the count when the `echo` signal is asserted. The counter register must be 8 bits wide to be able to count up to 150 ($2^7 = 128$ and $2^8 = 256$). The counter is reset to 0 after it reaches 149 so that the counter period is 150 clock cycles. The `transmit` output is simply a signal that decodes a zero count. This output should really be registered to avoid glitches. The following block diagram shows the solution:



Which could be described in VHDL as:

```
-- EECE 379 1999/2000 Term 2
-- Mid-Term Exam, Question 1
-- Ed Casas, 2000/2/28

library ieee ;
use ieee.std_logic_1164.all ;
use ieee.std_logic_arith.all ;

entity sounder is
   port ( clk, echo : in std_logic ;
   transmit : out std_logic ;
   depth : out unsigned (7 downto 0) ) ;
end sounder ;

architecture rtl of sounder is
   signal cntreg, next_cntreg : unsigned (7 downto 0) ;
   signal outreg, next_outreg : unsigned (7 downto 0) ;
begin
   -- counter counts from 0 to 149
   next_cntreg <=
      conv_unsigned(0,8) when cntreg = 149 else
      cntreg + 1 ;

   -- outreg loads/holds count when echo returns
   next_outreg <=
      cntreg when echo = ’1’ else
      outreg ;

   -- register count and output
   process(clk)
   begin
      if clk’event and clk=’1’ then
         cntreg <= next_cntreg ;
         outreg <= next_outreg ;
      end if ;
   end process ;

   -- generate transmit pulse for one clock period
   transmit <=
      ’1’ when cntreg = 0 else
      ’0’ ;

   -- connect output
   depth <= outreg ;

end rtl ;
```

Figure 1 show the simulation results.

Question 2

There are many possible solutions. A solution written in C could be as follows:

```
/*
   EECE379 1999/2000 Term 2
   Mid-Term exam Solutions
   C solution for Question 2
*/

/* Return a non-zero value if the headlight switch is on, zero
   otherwise. */

int swtch()
{
  return inb(0x300) & 0x80 ;
}

/* Return a non-zero value if the clock signal is ’1’, zero
   otherwise. */

int clock()
{
  return inb(0x300) & 0x01 ;
}

/* Turn the headlight on if ’on’ is non-zero, off otherwise. */

void setlights(int on)
{
  outb(0x300,on?1:0) ;
}
```

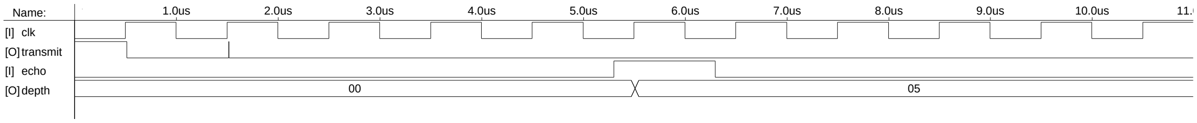


Figure 1: Simulation output.

```
main()
{
  int i, prev ;

  while (1) {                   /* loop forever */

  off:
    setlights(0) ;              /* turn lights off */
    while ( ! swtch() ) ;       /* wait until switched on  */

  on:
    setlights(1) ;              /* turn lights on */
    while ( swtch() ) ;         /* wait until switched off */


    for ( i=0 ; i<30 ; i++ ) {  /* delay 30 s */
      prev = clock() ;          /* get initial clock */
      while ( clock() == prev ) { /* wait for change */
        if ( swtch() ) goto on ; /* check for on */
      }
    }

  }

}
```

Which could be written in assembler as follows. The comments are references to the C version (rather than following good commenting style).

```
code segment public
        assume  cs:code,ds:code
        org     100h
start:

off:    mov     al,0            ; set(0)
        call    set

off1:   call    switch          ; while ! switch()
        jz      off1

on:     mov     al,1            ; set(1)
        call    set

on1:    call    switch          ; while switch()
        jnz     on1

        mov     ax,0            ; count=0
        mov     count,ax

delay:  mov     ax,count        ; while count < 30
        cmp     ax,30
        jge     end_delay

        call    clock           ; prev=clock()
        mov     prev,al

wait:   call    clock           ; while clock() == prev
        cmp     al,prev
        je      end_wait

        call    switch          ; if switch() goto on
        jnz     on

        jmp     wait
end_wait:

        mov     ax,count        ; count++
        add     ax,1
        mov     count,ax
jmp delay

end_delay:

        jmp     start

switch:                         ; return switch state
        mov     dx,300h
        in      al,dx
        and     ax,80h
        ret
clock:                          ; return clock signal
        mov     dx,300h
        in      al,dx
        and     ax,01h
        ret
set:                            ; turn lights on/off
        mov     dx,300h
        out     dx,al
        ret

count   dw      1 dup (?)       ; delay count
prev    db      1 dup (?)       ; previous port value

code ends
        end     start
```