THE UNIVERSITY OF BRITISH COLUMBIA
DEPARTMENT OF ELECTRICAL ENGINEERING
ELEC 464 : Microcomputer System Design (Fall 1996)

FINAL EXAMINATION

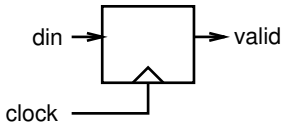
3:30 – 6:30 PM
December 16, 1996
MacMillan 166

This exam has four (4) questions on five (5) pages. The marks for each question are as indicated and there are a total of 64 marks. Answer all questions in the exam book provided. You may answer the questions in any order. Books, notes and calculators are allowed. You may keep this exam paper.

Question 1 (20 marks)

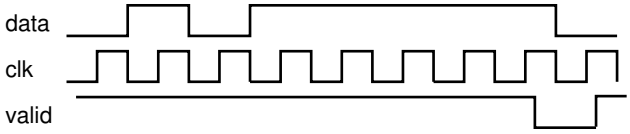
This question asks you to design a circuit to detect the zero bit which is inserted following a sequence of five consecutive ones in an HDLC serial data stream.

Your circuit should have a clock input (`clk`), a data input (`data`), and a "valid data" output (`valid`) as shown below:



The data input is valid on each rising clock edge and there is one rising clock edge per data bit. Your design must be a Moore state machine that changes state on the rising edge of the clock input. The output should be set to '1' if the *next* bit will be valid data and '0' if the *next* bit is a "stuffed" bit.

Your design should assume that all bits following five '1's will be zero bits (i.e. you may ignore end-of-frame flags and invalid sequences). The following diagram shows sample input and output waveforms:



(a) Choose an appropriate number of state variables and the encoding for the state variables and describe the state transition behaviour of the state machine in the form of a table as shown below. The table should have three columns: (1) the values of the state variables for the current state, (2) the value of the `data` input, and (3) the values of the state variables for the *next* state. You may use 'X' to indicate a "don't care" value.

State Variables	`data`	Next State

Also give the value of the `valid` output for each state in the form of a table as shown below. You may *not* combine the two tables.

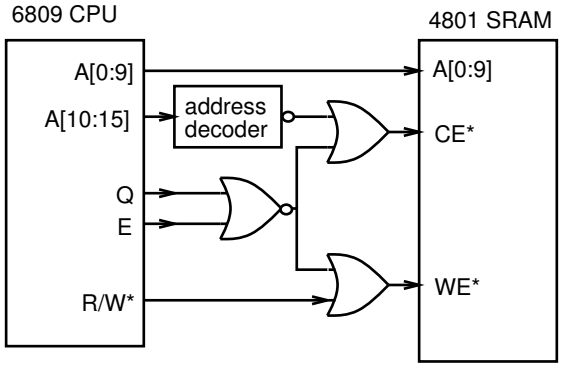
State Variables	`valid`

(b) Given the following VHDL entity, write a VHDL architecture that implements the above state machine. Your architecture must be synthesizable (e.g. by Synopsys Design Compiler). Use separate processes for the combinational and sequential parts of your design.

```
entity unstuff is
   port ( data, clk : in  bit ;
          valid : out bit ) ;
end unstuff ;
```

Question 2 (18 marks)

The diagram below shows part of the interface between a Motorola MC68B09 microprocessor and a Mostek MK4801A-3 static RAM. The 6809's Q and E outputs are OR'ed and used to enable both the SRAM's chip-enable (CE*) and write enable (WE*) inputs. This means that during a write cycle both CE* and WE* will be asserted when either E or Q is asserted. *Hint: draw CE* and WE* on the 6809 timing diagram.*



The following timing diagram and table gives the MC68B09 write-cycle timing specifications. Read-cycle timing specifications have been omitted. You may assume parameter 4, *Clock Rise and Fall Time*, is zero.

Solutions To Final Exam

Question 1

(a) State Machine Design

The state variable encoding below is the binary representation of the number of consecutive '1's "seen" by the circuit so far. Only six states are required so three variables (a,b,c) are sufficient. A one-hot encoding using six state variables would also be suitable. The table below does not show states that would not be encountered in normal operation.

State Variables			Input	Next State		
a	b	c	din	a	b	c
X	X	X	0	0	0	0
0	0	0	1	0	0	1
0	0	1	1	0	1	0
0	1	0	1	0	1	1
0	1	1	1	1	0	0
1	0	0	1	1	0	1
1	0	1	1	0	0	0

The outputs are zero for all states except the state where five '1's have been "seen" (and therefore the next bit is a stuffed bit).

State Variables			Output
a	b	c	valid
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0

(b) VHDL Code

```
entity unstuff is
   port ( data, clk : in  bit ;
          valid : out bit ) ;
end unstuff ;

architecture rtl of unstuff is
   signal state, nexts : bit_vector (2 downto 0) ;
begin

   -- compute next state
   process(state,data)
   begin
      if data = '0' then
         nexts <= "000" ;
      else
         case state is
            when "000" => nexts <= "001" ;
            when "001" => nexts <= "010" ;
            when "010" => nexts <= "011" ;
            when "011" => nexts <= "100" ;
            when "100" => nexts <= "101" ;
            when "101" => nexts <= "000" ;
            when others => nexts <= "000" ;
         end case ;
      end if ;
   end process ;

   -- latch state on rising clock edge
   process(clk,nexts)
   begin
      if clk'event and clk='1' then
         state <= nexts ;
      end if ;
   end process ;

   -- output
   process(state)
   begin
      if state = "101" then
         valid <= '0' ;
      else
         valid <= '1' ;
      end if ;
   end process ;

end rtl ;
```

Question 2

The table showing the expression and values for the SRAM write cycle timing margins is shown in Table 1. All times in nanoseconds.

Question 3

A schematic of the answer is shown in Figure 1.

Question 2 (7 marks)

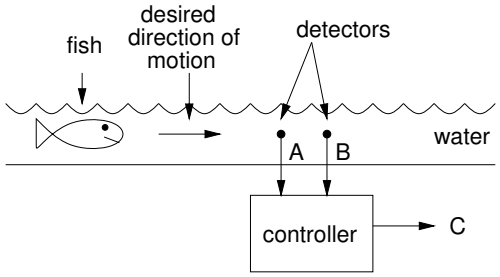
Write a C function declared as `int oesum(int a[], int n, int doeven)`. The argument `a` is an integer array and `n` is the number of elements in the array. If the argument `doeven` is non-zero, your function should return the sum of the even-valued numbers in the array, otherwise your function should return the sum of the odd-valued numbers in the array. Write only the function `oesum()`, not a complete program.

For example, if `a` contained the values `{2, 1, 3}`, then `oesum(a,3,2)` would return 2 and `oesum(a,3,0)` would return 4.

Hint: Make sure any logical expressions do what you expect.

Question 3 (12 marks)

This question asks you to design a controller that generates an output pulse when a fish is detected swimming in the desired direction down a stream. Your controller has two fish-detector inputs: `A` and `B` and one indicator output: `C` as shown below:



The fish detector outputs are 0 (no fish detected) and 1 (fish detected). The controller output is initially set to 0 while waiting for a fish. To make sure you only detect fish swimming in the desired direction, your controller must wait until A is 1 and B is 0 followed by the situation when A and B are both 1. When this sequence of input conditions occurs the output C is turned to 1. C remains set to 1 until A turns to 0 again. The controller then resumes waiting for a fish.

Fish swimming the wrong way (from B to A) or that only manage to turn A to 1 and then exit backwards should not generate an output pulse. Fish that are so fast that they only turn A to 1 and then only B to 1 should also not generate an output. However, once a fish has turned on only A and then both A and B you may assume it will continue swimming in the desired direction.

Design a state machine for the controller. List the inputs and outputs. Choose a sufficient number of states and give a name to each state. Write a table giving the output for each state. Draw a state transition diagram showing the states and the logical conditions that cause transitions between them. Write out a tabular description of the state machine with the following columns: starting state, input, next state.

You may use an "X" to indicate that an *input* has no effect.

Hint: My solution has 3 states.

THE UNIVERSITY OF BRITISH COLUMBIA
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
APSC 380 : Introduction to Microcomputers
1997/98 Winter Session Term 2

MID-TERM EXAMINATION
9:30 am – 10:20 am
February 25, 1997

This exam has four (4) questions. The marks for each question are as indicated. There are a total of 28 marks. Answer all questions. Write your answers in the exam book provided. Show your work. You may answer the questions in any order. Books, notes and calculators are allowed. You may keep this exam paper.

Question 1 (5 marks)

What is printed by the following C program? Show your work.

```
#include <stdio.h>
#define N 5

main()
{
   int i, c ;
   char x[N] = { 0, 4, 3, 2, 1 } ; /* values of x[0], x[1], ... */
   for ( i=N-1 ; x[i] ; i-- ) {
      c = x[i] & 0x02 ;
      printf ( "%d\n", c ) ;
   }
}
```

Hint: Start by figuring out the values taken on by `i`.

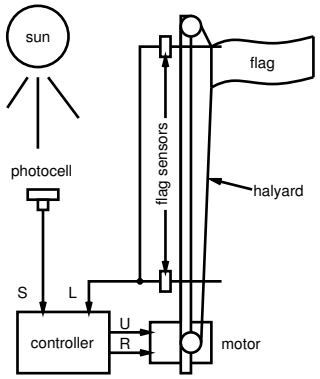
Question 2 (8 marks)

Write a C function called `blanks()` that takes one null-delimited string argument called `s`, and returns an integer. This function should return a count of the number of spaces (`' '`) in the string. Write only the function `blanks()`, not a complete program.

For example, `blanks("A nice function")` would return 2.

Question 3 (11 marks)

This question asks you to design a controller for a device that raises a flag up a flagpole in the morning and lowers it at night:



The halyard used to raise and lower the flag is driven by a motor. Two signals control the operation of this motor. The first, *U* controls the motor direction: it should be set high to move the flag up and low to move the flag down. The second signal, *R*, turns the motor on and off: it should be set high to run the motor and low to leave it off.

Your controller uses a signal from a photocell, *S*, whose output is high during the day. Your controller also has a signal *L* that goes high to indicates that the flag has reached either the top or bottom limit of the flagpole (this sensor cannot determine which limit has been reached).

Design a state machine controller for the motor controller. List the inputs and outputs. Choose a sufficient number of states and give a name to each state. Write a table giving the output conditions for each state. Draw a state transition diagram showing the states and the logical conditions that cause transitions between them. Write out a tabular description of the state machine with the following columns: starting state, input, next state.

Use "X" to indicate that an *input* has no effect. You may assume the flag will reach its limit before the next sunrise or sunset.

Hint: You may need more states than there are combinations of outputs.

Question 4 (4 marks)

A 2-to-4 decoder is a combinational circuit with two inputs and four outputs. At all times only one of the four outputs is high. The two inputs select which of the four outputs is to be set high as shown in the following truth table:

inputs		outputs			
A	B	Y0	Y1	Y2	Y3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Derive the logic equations for each of the outputs. Draw a schematic diagram for the decoder using AND, OR and NOT gates.

APSC 380 : INTRODUCTION TO MICROCOMPUTERS
1997/98 WINTER SESSION TERM 2

Solutions to Mid-Term Exam

Question 1

The `for` statement initializes the variable `i` to `N-1` (4), executes the body of the loop while `i` is non-zero and decrements it at the end of each loop. Therefore the values of `i` in the loop will be 4, 3, 2, and 1.

The first expression in the loop sets `c` to the bitwise logical AND of `i` and 0x02. The value of this expression will be 2 if bit 1 of `i` is set and zero otherwise. The binary values of `i` will be: 100, 011, 010 and 001. Only the second and third values have bit 1 set so the program will print four lines:

```
0
2
2
0
```

Question 2

One possible solution is:

```
/* Count and return the number of blank (space)
   characters in the string s. */

int blanks ( char s[] )
{
   int i, n ;
   n = 0 ;
   for ( i=0 ; s[i] ; i++ ) {
      if ( s[i] == ' ' ) {
         n++ ;
      }
   }
   return n ;
}
```

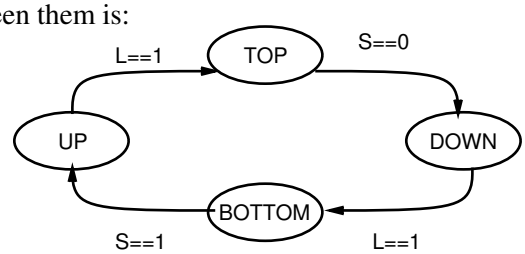
Question 3

As explained in the question, the inputs are the sunshine detector *S*, and the flag limit detector *L*. The two outputs are *U* the motor up/down control and *R*, the motor on/off control.

There are three useful combinations of outputs: off, motor going up, and motor going down. However, if we design the controller using only three states we find that it is not possible to determine when to exit the off state (in one case we need to exit it when the *S* sensor goes high, in the other case we need to exit it when the *S* sensor goes low). We can avoid this problem by using two "off" states: one when the flag reaches the upper limit and one when the flag reaches the lower limit. Appropriate names and outputs might be as shown in the following table:

state	*R*	*U*
TOP	0	0
BOTTOM	0	0
DOWN	1	0
UP	1	1

The state transition diagram showing the states and the logical conditions that cause transitions between them is:



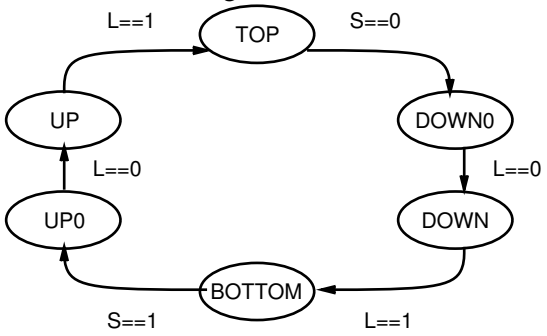
and a tabular description of the state machine is:

starting state	input		next state
	S	*L*	
TOP	0	X	DOWN
TOP	1	X	TOP
DOWN	X	0	DOWN
DOWN	X	1	BOTTOM
BOTTOM	0	X	BOTTOM
BOTTOM	1	X	UP
UP	X	0	UP
UP	X	1	TOP

where the "X" indicates that an *input* has no effect.

However, this solution will not work properly if the state transitions happen so fast that the limit switch will still be active ($L = 1$) after the controller has spent just one clock period in the `UP` or `DOWN` states. In this case the controller would immediately

transition from UP to DOWN to BOTTOM states without moving the flag very far. To avoid this problem we can either slow down the state transitions (by using a slow clock) or add two additional states as shown in the state transition diagram below:



for which the state transition table is:

starting state	input		next state
	S	L	
TOP	0	X	DOWN0
TOP	1	X	TOP
DOWN0	X	0	DOWN
DOWN0	X	1	DOWN0
DOWN	X	0	DOWN
DOWN	X	1	BOTTOM
BOTTOM	0	X	BOTTOM
BOTTOM	1	X	UP0
UP0	X	0	UP
UP0	X	1	UP0
UP	X	0	UP
UP	X	1	TOP

The outputs for the DOWN0 and UP0 would be the same as the outputs for the DOWN and UP states respectively.

Either solution is acceptable.

Question 4

The logic equations for each of the four outputs can be written in sum-of-products form as:

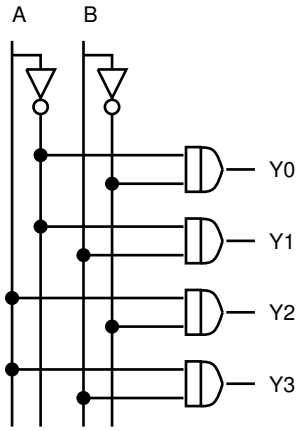
$Y0 = \overline{\mathrm{A}}\,\overline{\mathrm{B}}$

$Y1 = \overline{\mathrm{A}}\,B$

$Y2 = A\,\overline{\mathrm{B}}$

$Y3 = A\,B$

and a schematic diagram for the decoder is:

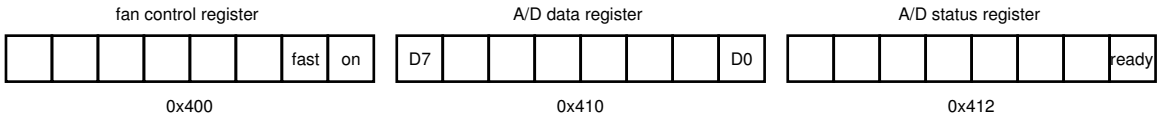


Question 3 (12 marks)

This question asks you to write a C program to control a two-speed cooling fan. The fan speed should be set according to the temperature, T, as follows:

Temperature (T)	Fan Speed
$T < 25$	off
$25 \leq T \leq 40$	slow
> 40	fast

Your program has access to three registers as shown below:



Your program controls the fan by writing to a control register at address 0x400. The least-significant (LS) bit controls the motor (1=on, 0=off). The second least-significant bit controls the speed (1=fast, 0=slow). The other bits of the control register have no effect.

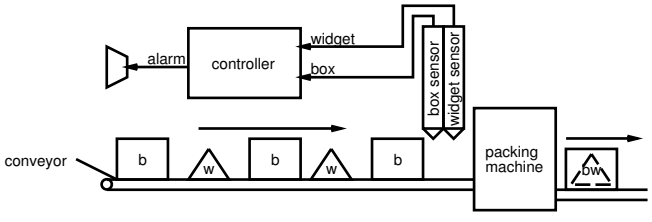
Your program determines the temperature by reading an (8-bit) A/D data register at address 0x410. The value read from this data register is related to the temperature by the equation $value = 4T + 25$.

The A/D has a status register at address 0x412. The LS bit of this status register indicates whether the A/D data register is ready to be read (1=ready, 0=not ready). The values of the other bits of the status register are undefined. Your program must wait until the status register indicates that the A/D data register is ready to be read before reading the A/D data register.

Write a C program (a `main()` function) that continuously monitors the temperature and adjusts the fan speed according to the specifications given above. Use the `speek()` and `spoke()` functions described in Lab 2 to read and write the registers.

Question 4 (12 marks)

A widget-packing machine (shown below) alternately receives boxes and widgets from a conveyor belt. The machine uses each pair to produce a boxed widget.



This question asks you to design a state machine for a controller that turns on an alarm when an out-of-order item arrives at the packing machine. You do *not* need to include a feature to turn off the alarm.

The controller has two inputs: a box-sensor (`box`), and a widget-sensor (`widget`). Each input is 1 when the corresponding item arrives at the machine. Only one of the inputs can be 1 at any given time. Between items both sensors are 0. The controller has one output (`alarm`) that rings the alarm when it is set to 1.

Design a state machine for the alarm controller. List the inputs and outputs. Choose a sufficient number of states and give a name to each state. Write a table giving the output condition for each state. Draw a state transition diagram showing the states and the logical conditions that cause transitions between them. Write out a tabular description of the state machine with the following columns: starting state, inputs, next state. You need not include input conditions that are not possible.

Question 5 (14 marks)

For parts (a) and (b) explain your reasoning in one or two sentences. Your answers must be unambiguous.

(a) [Multiple Choice] The STROBE signal of a parallel printer interface should be set low when:

- **(a)** the BUSY signal is high
- **(b)** the BUSY signal is low
- **(c)** the data lines contain the next character to print
- **(d)** (a) and (c)
- **(e)** (b) and (c)
- **(f)** none of the above

(b) [Multiple Choice] You need to A/D convert a slowly-changing analog signal generated by a remote sensor. The sensor is connected to the A/D by a long pair of wires running through an industrial plant with many electrical machines. You notice that the signal at the input to the A/D converter is very noisy. Which of the following devices might help you resolve this problem:

- **(a)** a multiplexer
- **(b)** a low-pass filter
- **(c)** a sample-and-hold
- **(d)** a differential amplifier
- **(e)** (b) and (d)
- **(f)** (c) and (d)
- **(g)** (a) and (c)
- **(h)** none of the above

THE UNIVERSITY OF BRITISH COLUMBIA
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
ELEC 379 : Design of Digital and Microcomputer Systems
1998/99 Winter Session Term 2

FINAL EXAMINATION
8:30 AM – 11:30 AM
January 23, 1999

This exam has five (5) questions. The marks for each question are as indicated. There are a total of 70 marks. Answer all questions. Write your answers in the exam book provided. Show your work. You may answer the questions in any order. Books, notes and calculators are allowed. You may keep this exam paper.

Question 1 (15 marks)

This question asks you to design a device that performs a binary search for an unknown value.

The device has the following VHDL entity declaration:

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

entity SAR is
   port (
      test_out : out std_logic_vector (7 downto 0) ;
      done : out std_logic ;
      high, reset, clk : in std_logic ) ;
end SAR ;
```

This device has two internal 8-bit registers: `test` and `delta`.

On each rising edge of the clock if `reset` is '1' your design should set `test` to 128 and `delta` to 64.

On each rising edge of the clock if `reset` is not '1' your circuit should do the following:

- if `high` is '1' it should subtract `delta` from `test` otherwise it should add `delta` to `test`, and
- divide the value of `delta` by two.

The `done` output should be set to '1' only when `delta` is 0.

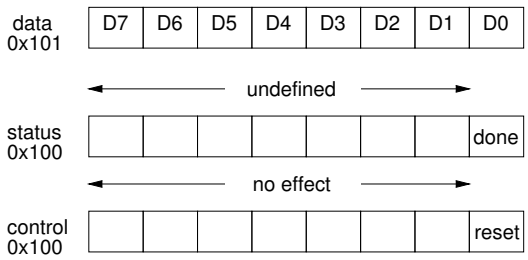
Write an architecture that implements this device and is synthesizeable by MaxPlus+II. Use type conversion functions as necessary. Any `process` in your VHDL code must contain

exactly one `if` statement surrounding one or two simple signal assignments. You need not include comments, `library` or `use` statements.

Hints: use "-" or "+". MaxPlus+II allows you to divide `unsigned` *values by 2.*

Question 2 (14 marks)

Write a subroutine, `sample:`, in 8086 assembly language that reads 200 (decimal) bytes from a peripheral and saves the bytes to a buffer. The peripheral is accessed through an 8-bit input data register at I/O (not memory) address 101H, an 8-bit status register at I/O address 100H and an 8-bit control register also at address 100H:



Before reading each byte from the data register your program must do the following:

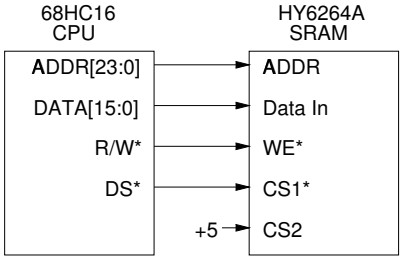
- write a 1 to the least significant (LS) bit of the control register, and
- wait until the LS bit of the status register is '1'.

Note that the other bits of the status register are undefined and only the LS bit of the control register has any effect.

Your function must save and restore any registers it modifies before returning control to the calling function with a `RET` instruction. You must declare storage for any variables you use (including the buffer), but you need not include comments or assembler directives such as `segment`, `assume` or `org`.

Question 3 (17 marks)

The diagram below shows part of the interface between a Motorola 68HC16 microprocessor and a Hyundai HY6264A SRAM:



THE UNIVERSITY OF BRITISH COLUMBIA
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
ELEC 379 : Design of Digital and Microcomputer Systems
1998/99 Winter Session Term 2

SUPPLEMENTAL EXAMINATION
August, 1999

This exam has five (5) questions. The marks for each question are as indicated. There are a total of 61 marks. Answer all questions. Write your answers in the exam book provided. Show your work. You may answer the questions in any order. Books, notes and calculators are allowed. You may keep this exam paper.

Question 1 (15 marks)

This question asks you to design a device that is part of a binary division circuit.

The device has the following VHDL entity declaration:

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

entity divider is
   port (
      n_in, d_in : in unsigned (15 downto 0) ;
      q_out : out std_logic ;
      load, clk : in std_logic ) ;
end divider ;
```

This device has two internal 8-bit registers: `n` and `d`.

The `q_out` output should always be set to '1' if `n` is less than or equal to `d`, and '0' otherwise.

On each rising edge of the clock if `load` is '1' your design should set `n` to `n_in` and `d` to `d_in`.

On each rising edge of the clock if `load` is not '1' your circuit should set the value of `n` as follows:

- if `q_out` is '1', it sets `n` to `n` minus `d` divided by two ($\frac{n-d}{2}$)
- if `q_out` is '0' it sets `n` to `n` divided by two ($\frac{n}{2}$)

Write an architecture that implements this device and is synthesizeable by MaxPlus+II. Use type conversion functions as necessary. Any `process` in your VHDL code must contain exactly one `if` statement surrounding one or two simple signal assignments. You need not include comments, `library` or `use` statements.

Hints: use "-" or "+". MaxPlus+II allows you to divide `unsigned` *values by 2.*

THE UNIVERSITY OF BRITISH COLUMBIA
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
ELEC 379 : Microcomputer System Design
1998/99 Winter Session Term 2

MID-TERM EXAMINATION
8:30 – 9:20 AM
February 25, 1999

This exam has two (2) questions. The marks for each question are as indicated. There are a total of 25 marks. Answer all questions. Write your answers in the exam book provided. Show your work. You may answer the questions in any order. Books, notes and calculators are allowed. You may keep this exam paper.

Question 1 (15 marks)

(a) Design a decade counter. Write the entity and architecture for this device assuming it is called `decade`, has an the following inputs and outputs:

- a counter-enable input, `en`
- a clock input, `clk`
- a four-bit binary count output, `q`
- a one-bit "terminal count" output, `tc`

The counter is incremented if `en` is asserted. The counter counts from 0 to 9 and then returns to 0. `tc` is asserted when the count is 9. The counter output changes on the rising edge of `clk`.

(b) Assume a package called `counters` exists in the `work` library and that it includes the `decade` counter as a component. Write the entity and architecture of a two-digit decade counter that has the following inputs and outputs:

- a clock input, `clk`
- two four-bit outputs: `a` and `b` (where `a` is the least-significant digit).

For both parts (a) and (b) write an entity and architecture that is synthesizeable by Max-Plus+II. All signals are `std_logic` or `std_logic_vector` and are active-high. Use type conversion functions as necessary. Any `process` in your VHDL code must contain exactly one `if` statement and one signal assignment statement. Do not include comments. Include any `library` and `use` statements required.

Hint: the `tc` *output of the LS digit counter is used to* `en`*able the MS digit counter.*

Question 2 (10 marks)

ELEC 379 : DESIGN OF DIGITAL AND MICROCOMPUTER SYSTEMS
1998/99 WINTER SESSION, TERM 2

Solutions to Mid-Term Exam

Question 1 (a)

```
-- Decade counter with enable input and
-- terminal count output.
-- Ed Casas, February 25, 1999

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

entity decade is port (
   en, clk : in std_logic ; -- enable and clock
   q : out std_logic_vector (3 downto 0) ; -- count
   tc : out std_logic ) ; -- terminal count
end decade ;

architecture rtl of decade is
   signal count, next_count, count_plus_1 :
               unsigned (3 downto 0) ;
begin
   -- next value in count
   count_plus_1 <=
        0 when count = 9 else
        count + 1 ;

   -- next count value
   next_count <=
        count_plus_1 when en = '1' else
        count ;

   -- register the count
   process(cp)
   begin
        if cp'event and cp = '1' then
           count <= next_count ;
        end if ;
   end process ;

   -- connect to output
   q <= std_logic_vector(count) ;

   -- terminal count
   tc <=
        '1' when count = 9 else
        '0' ;

end rtl ;
```

Question 1 (b)

```
-- Two-digit decimal counter.
-- Ed Casas, February 25, 1999

library ieee ;
use ieee.std_logic_1164.all ;
use work.counters.all ;

entity twodigits is port (
        clk : in std_logic ;
        a, b : out std_logic_vector (3 downto 0) ) ;
end decade ;

architecture rtl of twodigits is
   signal a_en, b_en, b_tc : std_logic ;
begin
   a_en <= '1' ;
   c1: decade port map ( a_en, clk, a, b_en ) ;
   c2: decade port map ( b_en, clk, b, b_tc ) ;
end rtl ;
```

Question 2

```
;
; ELEC 379 1998/99 Mid-Term Exam
; Ed Casas, February 25, 1999
;
; convert string to lower case
;
toupper:
        push    ax      ; save registers
        push    bx
loop:   mov     al,[bx] ; get a character
        cmp     al,0    ; stop if it's zero
        jz      done
        cmp     al,'a'  ; convert if between
        jb      nocon   ;    'a' and 'z'
        cmp     al,'z'
        ja      nocon
        sub     al,20H  ; convert to lower case
        mov     [bx],al
nocon:  inc     bx      ; repeat for next character
        jmp     loop
done:   pop     bx      ; restore registers
        pop     ax
        ret
```

THE UNIVERSITY OF BRITISH COLUMBIA
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
EECE 379 : Design of Digital and Microcomputer Systems
2000/2001 Winter Session, Term 1

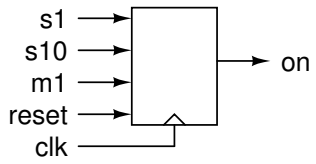
FINAL EXAMINATION
3:30 AM – 6:30 PM
December 11, 2000

This exam has five (5) questions on five (5) pages. The marks for each question are as indicated. There are a total of 44 marks. Answer all questions. Write all answers in the exam book provided. Show your work. You may answer the questions in any order. Books, notes and calculators are allowed. Show your work. You may keep this exam paper.

Question 1 (10 marks)

This question asks you to design a timer circuit that controls a microwave oven.

The controller has three push-button inputs that are used to increase the "remaining cooking time." `s1`, `s10`, and `m1` are high when the user is pushing a button that should increase the "remaining cooking time" by 1, 10 or 60 seconds respectively. Another push-button input resets the "remaining cooking time" to zero. The frequency of the clock input, `clk`, is 1 Hz. The controller has one output, `on`, which turns on the microwave power. The `on` output is high whenever the "remaining cooking time" is greater than zero.



The device has the following VHDL entity declaration:

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

entity controller is
   port (
      s1, s10, m1, reset : in std_logic ;
      clk : in std_logic ;
      on : out std_logic ) ;
end controller ;
```

You design must satisfy *all* of the following requirements.

The "remaining cooking time" changes only on the rising edge of `clk` (it is a synchronous design).

The "remaining cooking time" can vary between 0 and 600 seconds (10 minutes) and should *never* be set to a value outside this range.

If only *one* of the `s1`, `s10`, or `m1` buttons are being pushed, then the "remaining cooking time" should be incremented by the the amount indicated by the button (but never to more than 600).

If only the `reset` button is being pushed, then the "remaining cooking time" should be set to zero.

If no buttons are being pushed then the "remaining cooking time" should be decremented by 1 (but never to less than 0).

For other input conditions the "remaining cooking time" should not change.

Write an architecture that implements this device and is synthesizeable by MaxPlus+II. You may use only `std_logic`, `std_logic_vector` or `unsigned` types. Use type conversion functions as necessary. Any `process` in your VHDL code may only contain exactly one `if` statement controlling one or more simple signal assignment statements.

You need not include comments, `library` or `use` statements. You need not synchronize inputs or register outputs (for glitch removal). You may ignore the initial (uninitialized) state of the device.

Hints: The VHDL operators "`+`" and "`<=`" can be applied to certain combinations of `unsigned` *and* `integer` *values.* $2^9 = 512$. $2^{10} = 1024$.

Question 2 (10 marks)

Write a function, `checkbuf:`, in 8086 assembly language that validates an array of 200 unsigned 16-bit integers.

When your function is called, the register BX will contain the address (offset portion) of the buffer where the values are stored.

Your function should check each and every value in the buffer: if a value is less than 1024, then that value should be set to 1024; if a value is greater than 64512 then that value should be set to 64512.

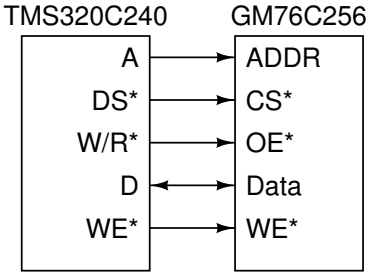
Your function must save any registers it modifies and restore them before returning with a `RET` instruction.

You must declare storage for any variables. You need not include comments or assembler directives such as `segment`, `assume` or `org`.

Hints: Write a C or pseudo-code solution before you start coding.

Question 3 (8 marks)

The diagram below shows part of the interface between a TI TMS320C240 microprocessor and a Hyundai GM76C256 SRAM:



THE UNIVERSITY OF BRITISH COLUMBIA
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
EECE 379 : Microcomputer System Design
2000/2001 Winter Session Term 1

MID-TERM EXAMINATION
12:30 – 1:20 PM
October 23, 2000

This exam has two (2) questions. The marks for each question are as indicated. There are a total of 20 marks. Answer all questions. Write your answers in the exam book provided. Show your work. You may answer the questions in any order. Books, notes and calculators are allowed. You may keep this exam paper.

Question 1 (11 marks)

This question asks you to design a subsystem for a hard disk controller. The interface consists of:

- a 10-bit `unsigned` input, `track`
- two `std_logic` outputs, `up` and `down`
- a 1-bit `std_logic` input, `reset`
- a clock input, `clk`

The device has an internal 10-bit `unsigned` register, `reg`.

The `up` output is asserted whenever the value of `reg` is numerically less than `track`. The `down` output is asserted whenever the value of `reg` is numerically greater than `track`.

`reg` changes only on the rising edge of the clock signal. The value of `reg` is set to 0 if `reset` is '1' (high), it is incremented if `up` is '1', it is decremented if `down` is '1', and otherwise remains unchanged. Figure 1 shows some sample waveforms.

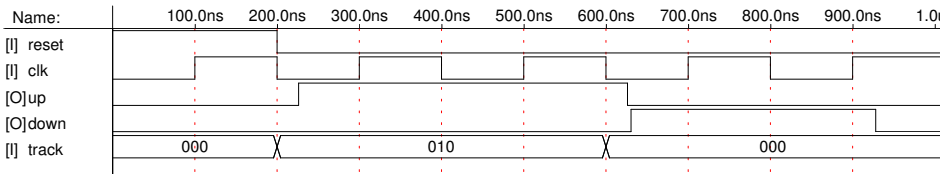


Figure 1: Sample Waveforms (Question 1)

Write a VHDL entity (named `hdc`) and architecture for a circuit that is synthesizeable by MaxPlus+II and meets these specifications. You may use `std_logic`, `std_logic_vector` or `unsigned` types.

Apply type conversion functions as necessary. Any `process` in your VHDL code must contain exactly one `if` statement controlling one simple signal assignment statement. You need not include comments. Include any `library` and `use` statements required.

Hint: The comparison operators > and < can be used with unsigned signals.

Question 2 (9 marks)

This question asks you to write an 80x86 assembly-language routine (a "function") called `outbuf:` that outputs each byte in a data buffer to an output port.

When your routine is called the register BX will contain the address of the first byte in the buffer and register CX will contain the number of bytes to be output.

For each byte in the buffer, your routine must:

1. repeatedly input from I/O (not memory) address 0x30 until the least-significant bit of the value read is 1. The values of the other bits must be ignored.
2. output the byte from the buffer to I/O (not memory) port 0x50

Your routine must terminate with a RET statement. It does not have to save or restore the values of any registers.

You must declare storage for any temporary variables you use, but you need not include comments or assembler directives such as `segment`, `assume` or `org`.

Hints: Write out a C solution before you start.

Solutions to Mid-Term Exam

Question 1

```
-- EECE 379 Midterm Exam
-- Question 1 Solution
-- Ed Casas, October 23, 2000

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

entity hdc is port (
   clk, reset : in std_logic ;
   up, down : out std_logic ;
   track : in unsigned(9 downto 0) ) ;
end hdc ;

architecture rtl of hdc is
   signal reg, next_reg : unsigned(9 downto 0) ;
   signal u, d : std_logic ;
begin

   -- direction outputs

   -- internal
   u <= '1' when reg < track else '0' ;
   d <= '1' when reg > track else '0' ;
   -- external
   up <= u ;
   down <= d ;

   -- track register control

   next_reg <=
      conv_unsigned(0,10) when reset = '1' else
      reg+1 when u = '1' else
      reg-1 when d = '1' else
      reg ;

   -- current-track register

   process(clk)
   begin
      if clk'event and clk='1' then
         reg <= next_reg ;
      end if ;
   end process ;
end rtl ;
```

Question 2

A solution in C would be:

```
main( char *bx, int cx )
{
   while ( cx > 0 ) {
      while ( ( peek(0x30) & 0x01 ) != 0x01 ) {
        /* do nothing */
      }
      poke(0x50,*bx++) ;
      cx-- ;
   }
}
```

The corresponding assembly-language program is:

```
outbuf: cmp     cx,0    ; done if count <= 0
        jle     done
check:  mov     dx,30H  ; get status
        in      al,dx
        and     al,01H  ; check LS bit
        cmp     al,01H
        jne     check   ; repeat until '1'
        mov     al,[bx] ; get byte from buffer
        mov     dx,50H  ;   and output
        out     dx,al
        add     bx,1    ; update pointer
        sub     cx,1    ;   and count
        jmp     outbuf  ; loop back
done:   ret
```

THE UNIVERSITY OF BRITISH COLUMBIA
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
EECE 379 : Design of Digital and Microcomputer Systems
1999/2000 Winter Session, Term 2

FINAL EXAMINATION
8:30 AM – 11:30 AM
April 14, 2000

This exam has five (5) questions on five (5) pages. The marks for each question are as indicated. There are a total of 50 marks. Answer all questions. Write all answers in the exam book provided. Show your work. You may answer the questions in any order. Books, notes and calculators are allowed. Show your work. You may keep this exam paper.

Question 1 (10 marks)

This question asks you to design *hardware* to implement the same car headlight controller described in the mid-term exam. This controller implements a delay feature so that the headlights stay on for 30 seconds after the driver turns the headlight switch off.

The device has the following VHDL entity declaration:

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

entity lightdelay is
   port (
      lights : out std_logic ;
      switch, clk : in std_logic ) ;
end lightdelay ;
```

The `lights` output controls the headlights: a high output turns them on and a low output turns them off. The headlight `switch` input is high when the driver turns the switch on and low when the driver turns the switch off. The clock input, `clk`, frequency is 10 Hz.

Your hardware should turn the headlights on when the driver turns the switch on. After the driver turns the switch off, there should be a delay of 30 seconds and then the headlights should be turned off. The driver should be able to turn the headlights back on during this delay time. The headlights should come on within a very short time ($\leq$ 100 ms) of the switch being turned on.

Write an architecture that implements this device and is synthesizeable by MaxPlus+II. You may use only `std_logic`, `std_logic_vector` or `unsigned` types. Use type conversion functions as necessary. Any `process` in your VHDL code must contain exactly one `if` statement controlling one or more simple signal assignment statements. You need not include

comments, `library` or `use` statements. You need not synchronize inputs or register outputs (for glitch removal). You may ignore the initial (uninitialized) state of the device.

Hints: Draw an (incomplete) state transition diagram. Use "-" or "+".

Question 2 (10 marks)

Write a function, `getdigits:`, in 8086 assembly language that reads characters and stores them in a buffer.

When your function is called, the register BX will contain the offset of the buffer where the characters are to be stored.

Your function should call a function, `getch`, that reads one character and returns it in AL. `getch` does not modify any registers except AL. You do not have to write the `getch` function.

Your function should return when the line-feed (0AH) character is read. This line-feed character should not be stored in the buffer.

Your function should also "throw away" (not store) any characters that are outside the range '0' to '9' (30H to 39H inclusive).

Your function should keep track of the number of characters that are stored in the buffer and place this value in AX when it returns.

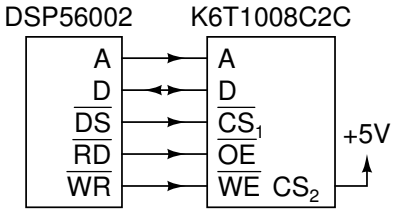
Your function must save any registers it modifies and restore them, except for AX, before returning with a `RET` instruction.

You must declare storage for any variables. You need not include comments or assembler directives such as `segment`, `assume` or `org`.

Hints: Write a C, flowchart, pseudo-code, or other high-level solution before you start coding.

Question 3 (10 marks)

The diagram below shows part of the interface between a Motorola DSP56002 microprocessor and a Samsung K6T1008C2C SRAM:



The microprocessor's data strobe (DS*) signal drives the RAM's chip select (CS1*). The RAM's write enable (WE*) is driven by the CPU's write signal (WR*). The RAM's output enable (OE*) is driven by the CPU's read signal (RD*). The RAM's second chip select (CS2) is always enabled. The RAM's address and data buses are connected to the corresponding CPU buses.

THE UNIVERSITY OF BRITISH COLUMBIA
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
EECE 379 : Microcomputer System Design
1999/2000 Winter Session Term 2

MID-TERM EXAMINATION
8:30 – 9:20 AM
February 25 2000

This exam has two (2) questions. The marks for each question are as indicated. There are a total of 22 marks. Answer all questions. Write your answers in the exam book provided. Show your work. You may answer the questions in any order. Books, notes and calculators are allowed. You may keep this exam paper.

Question 1 (11 marks)

This question asks you to design the digital electronics for a device that measures the depth of the water under a ship.

The device transmits a signal from the bottom of the ship. Some time later it receives the echo from the sea floor. By measuring the delay between transmitting the pulse and receiving the echo, your device can determine the depth.

Your device has the following inputs and outputs:

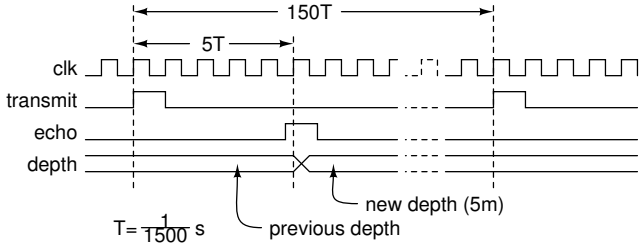
- a 1500 Hz clock input, `clk`
- and active-high input, `echo`, that indicates an echo is being received
- an active-high output, `transmit`, that turns on the transmitter
- an 8-bit output, `depth`, that indicates the depth in metres

Your device should transmit one pulse every 100 milliseconds (ten times per second). Each pulse should have a duration of $\frac{1}{1500}$ seconds. When the echo is received your device should output the newly-measured depth and hold it until the next depth measurement is available.

The speed of sound in water is approximately 1500 m/s. Since the clock rate is 1500 Hz, the number of clock cycles elapsed since the start of the transmitted pulse indicates the depth in metres.

You can assume that there will always be an echo and that it will last for exactly one clock period.

The following diagram shows an example of the input and output waveforms:



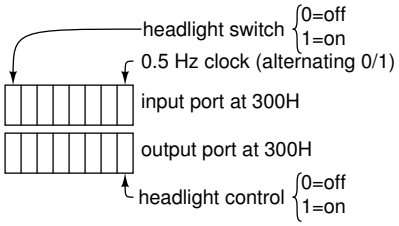
Write a VHDL entity and architecture for a circuit that is synthesizeable by MaxPlus+II and meets these specifications. You may use `std_logic`, `std_logic_vector` or `unsigned` types. Apply type conversion functions as necessary. Any `process` in your VHDL code must contain exactly one `if` statement controlling one or more simple signal assignment statements. You need not include comments. Include any `library` and `use` statements required. You need not synchronize inputs or register outputs (for glitch removal).

Hint: You will need at least two registers, one to hold the output and one to implement a counter.

Question 2 (11 marks)

This question asks you to write an 80x86 assembly-language program that implements a car headlight controller. This controller implements a delay feature so that the headlights stay on for 30 seconds after the driver turns the headlight switch off.

The interface between your program and the hardware is through two 8-bit ports (one input port and one output port). Both ports are at i/o (not memory) address 300H. The LS bit of the input port is a signal that alternates between 0 and 1 every second (i.e. the frequency of the signal is 0.5 Hz). The MS bit of the input port indicates the headlight switch position: 1 for "on" and 0 for "off". The LS bit of the output port actually controls the headlights: 1 turns them on and 0 turns them off:



Your program should run continuously. It should turn the headlights on when the driver turns the switch on. After the driver turns the switch off, there should be a delay of between 30 and 31 seconds and then the headlights should be turned off. The driver should be able to turn the headlights back on during this delay time. The headlights should come on within a very short time ($\ll 1$ s) of the switch being turned on.

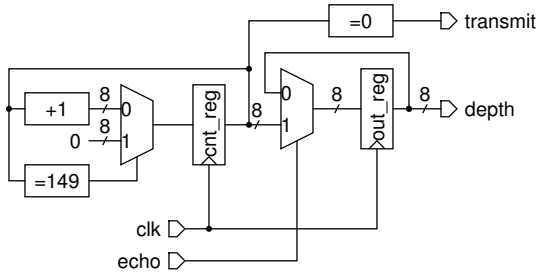
You must declare storage for any temporary variables you use, but you need not include comments or assembler directives such as `segment`, `assume` or `org`.

Hints: Write a pseudo-code, flowchart, or other high-level solution before you start coding! Use functions.

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) ;
}
```