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.

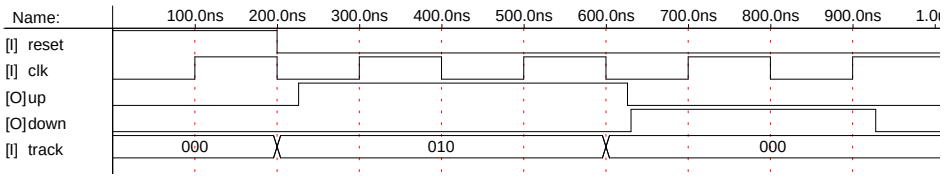

Figure 1: Sample Waveforms (Question 1)

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.