Note to ECE 243 Students in 2022: These questions are from the exam in this course, because the course has been re-ordered this year. Also, these questions were done online at home, in a much longer time frame with access to CPULATOR. So you should take from these questions our expectations of the 'level' of understanding we expect you to have for the midterm in 2022. You would not have the amount of time given on this exam to do all the questions in a midterm.

University of Toronto
Faculty of Applied Science and Engineering

Unsupervised Final Assessment
ECE243 – Computer Organization

Examiners – Stephen Brown and Jonathan Rose

Print and submit this page:

First Name ______________________ Last Name ______________________

Student Number ____________________

1. This assessment is available at 4:30 pm on April 27, 2020 Eastern time. You must submit all of your answers to Quercus under this course before 4:30 pm on April 28, 2020 Eastern time.
2. There are **four** questions in this final assessment. Your answer to **each question** is to be submitted to Quercus using *files* that you create. The *file names* are specified in each question in this document. Question 4 also requires the submission of rough work.
3. This is an *unsupervised final assessment*. You are allowed to use **only** the following aids: a calculator, spreadsheet, textbooks, your notes and the instructors' notes, your lab work in the course, CPUlator, and the Monitor Program. Writing an unsupervised final assessment necessitates a high level of integrity and honesty. Before beginning to write the assessment please read and sign the following statement:

   I pledge upon my honour that I will not violate our Faculty's Code of Behaviour on Academic Matters during this assessment by acting in any way that would constitute cheating, misrepresentation, or unfairness, including but not limited to, using unauthorized aids and assistance, impersonating another person, and committing plagiarism. I acknowledge that providing unauthorized assistance to someone else is also considered a serious academic offence.

   ______________________________ (*sign your name*)

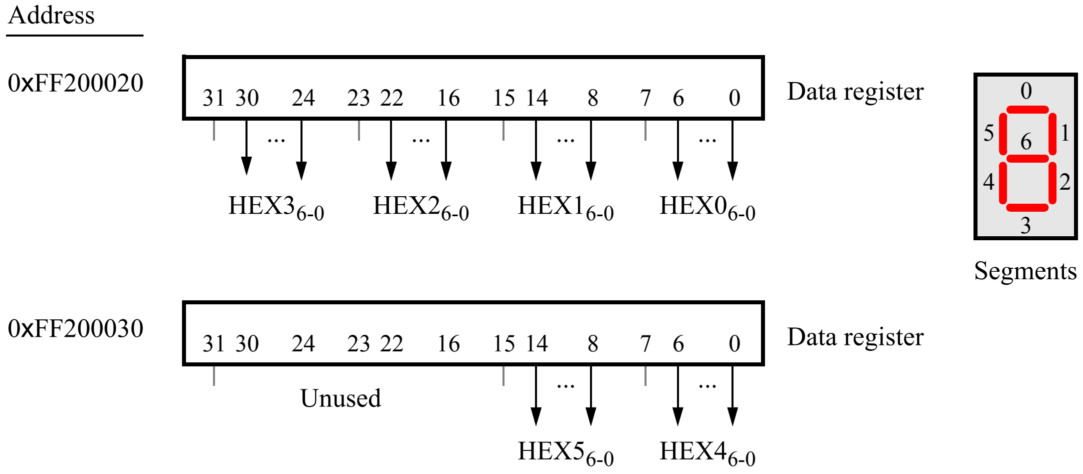
   If you have access to a printer: print this page, fill in your name and student number, sign the pledge, and then scan/photograph the page and submit it with the file name **firstpledge.jpg** or **firstpledge.pdf** to the Quercus assignment labelled 'First Honour Statement and Signature.'
   If you do not have access to a printer: in a text file named **firstpledge.txt**, type your name and student number, *the full pledge* into this file, and type your name in place of the signature. Upload this file to the same assignment as above. There is a second pledge at the end of this document that **you must also sign and submit once you're finished the assessment**.

[25 marks] 2. This question is about assembly-language code and interrupts. **Note:** Different from Question 1, you will submit individual files for each part of this question, to separate **Quercus** assignments, which are labelled **Question 2a**, **Question 2b**, and **Question 2c**.

Note: A short video demonstration of a complete solution for all parts of this question can be found at https://youtu.be/jPMutBfX19o

[8 marks] (a) You are to write an assembly-language program that works as follows: it reads the *SW* switch port and lights up a seven-segment display corresponding to the value read on SW_{2-0}. For example, if $SW_{2-0} = 000$, then the digit 0 is shown on *HEX0*. If $SW_{2-0} = 001$, then the digit 1 is displayed on *HEX1*, and so on, up to the digit 5 which would be shown on *HEX5* if $SW_{2-0} = 101$. Once a digit has been displayed, it should remain displayed even if *SW* is subsequently changed.

Recall that the seven-segments displays are connected to the two registers:



Also recall from the lab exercises in this course that you *cannot* perform store operations to individual *bytes* in these registers—you can only store 32-bit *words*. Hence, the code for your solution to this question can use `STR` instructions to write to the HEX display ports, **but cannot use `STRB` to write to these ports**.

Write your code in a file named **Q2a.s** and submit this file to the Quercus assignment labelled **Question 2a**.

[11 marks] (b) For this part you are to write an assembly-language program that flashes $LEDR_0$ on and off every 0.5 seconds. Your code should have a main program that writes to the *LEDR* port and your program should use *interrupts* to implement the required 0.5 second time intervals. To create interrupts every 0.5 seconds use the *A9 Private Timer*, which has a base address of `0xFFFEC600`. This timer is described in Section 2.4.1 of the *DE1-SoC Computer* documentation, which can be found on the Quercus home page for this course, by following the link **Laboratory Exercises** and then **ARM Materials**. Use this timer to make a delay of 0.25 seconds.

To configure the ARM processor's generic interrupt controller (GIC) use the code that was provided along with Lab Exercise 6.

Write the code for this part in a file named **Q2b.s** and submit this file to the Quercus assignment labelled **Question 2b**.

[6 marks] (c) For this part you are to augment your program from part (a) to add the following functionality: When the *SW* switches are set to select a particular digit (from 0 to 5), you are to flash this digit on and off every 0.5 seconds. Implement the required 0.5 second time intervals using interrupts from the same timer that you used in part (b) of this question.

Important: once a digit has been selected once, it must remain displayed even if *SW* has been subsequently changed.

Also Important: only the digit that is *currently* selected by *SW* is allowed to flash on/off. Marks will be deducted if multiple digits flash on/off.

Write the code for this part in a file named **Q2c.s** and submit this file to the Quercus assignment labelled **Question 2c**.

[25 marks] 3. This question is about C code. In this question you will submit individual files for each part of this question, to separate **Quercus** assignments, which are labelled **Question 3a**, **Question 3b**, **Question 3c** and **Question 3d**.

Note: A short demonstration of a working solution for all parts of this question can be found at https://youtu.be/Mofcc8QAbMY.

[5 marks] (a) For this part you are to write a C program that displays the characters `ECE243` on the hexadecimal displays *HEX5 - HEX0*. The message should flash on/off at a rate of *approximately* 0.5 seconds. Implement the required delay between *showing*, and *blanking* the characters in the display by using a software delay loop - **do not** use a hardware timer for this part of the question. Write the code for this part in a file named **Q3a.c** and submit it to the Quercus assignment labelled **Question 3a**.

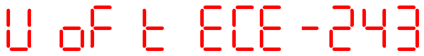
[6 marks] (b) For this part you are to write a C program that turns on one light at a time on the LEDR port. You should start by lighting up only $LEDR_0$, then *only* LEDR$_1$, then only $LEDR_2$, and so on, to $LEDR_9$, and then returning back to LEDR$_0$. The effect created by your program should be that a light appears to *scroll* across the LEDR port.

The speed at which the lights are scrolled should be controlled by using the *Interval Timer* in the *DE1-SoC Computer*. This timer has the base address `0xFF202000` and is described in Section 2.11 of the *DE1-SoC Computer* documentation. Use this timer to make a delay of 0.25 seconds. Use polled-I/O to wait for the timer to expire—**do not use interrupts!**

You should be able to control the *direction* in which the lights are scrolled by using the *KEY* port. Each time *any KEY* is pressed and released, the direction of scrolling should be reversed.

Write the code for this part in a file named **Q3b.c** and submit this file to the Quercus assignment labelled **Question 3b**.

[8 marks] (c) For this part you are to create a *scrolling* message on the seven-segment displays *HEX5 - HEX0*. Your C program should display the message U of t ECE-243 and should scroll the message in the right-to-left direction across the displays. The letters in the message can be constructed as



Control the speed of scrolling by using the same timer as in part (b), with a timer delay of 0.5 seconds. Use polled-I/O; **do not use interrupts**. You should be able to control the displays using the *KEY* port. When the message is currently scrolling, then pressing and releasing *any KEY* should stop the scrolling so that the *HEX* displays are static. Now pressing and releasing any *KEY* should restart the scrolling, etc.

Write the code for this part in a file named **Q3c.c** and submit this file to the Quercus assignment labelled **Question 3c**.

[6 marks] (d) For this part you are to augment your solution from part (*c*) to provide additional features using the pushbutton *KEYs*, as follows: KEY_0 should start/stop the scrolling (as was done for *any* KEY in part (*c*)), KEY_1 should *double* the scrolling speed, KEY_2 should *halve* the scrolling speed, and KEY_3 should *reverse the direction* of scrolling.

Write the code for this part in a file named **Q3d.c** and submit this file to the Quercus assignment labelled **Question 3d**.