Project 4: Interrupts

This project is due on **April 17, 2026** at **6 p.m.** and counts for 6.25% of your course grade.

The code and other answers you submit must be entirely your own work, and you are bound by the Honor Code. You may consult with other students about the conceptualization of the project and the meaning of the questions, but you may not look at any part of someone else's solution or collaborate with others to develop a solution. You may consult published references, provided that you appropriately cite them (e.g., with program comments), as you would in an academic paper.

There is starter code on **Github Classroom** (`https://classroom.github.com/a/SGtoF8He`), which is also where you will turn in the project by committing and pushing to the repository you create when you accept the assignment.

If you receive a permission error while accepting the assignment, check your email associated with your Github account for an invitation link to your repository.

Github Classroom also provides autograding for you. Click on the `Actions` tab in your repository, and select a commit to see the autograding report for that version of your repository. Autograding is run each time you commit.

Late days Each student is given three late days for the semester for use on homeworks or projects. If you wish to use one or more of your late days on this assignment, fill out this form: `https://forms.gle/e8siSUJAoGpyi7fN8`

Introduction

In this project, you will write interrupt service routines (ISRs) to implement a digital stopwatch on the DE10-Lite board.

Objectives:

- Understand how hardware interrupts work on the Nios II
- Write timer and push-button ISRs
- Manage shared state between a main loop and ISRs

Part 1. Stopwatch

You will implement a stopwatch in Nios II assembly that displays elapsed time on the 7-segment displays and responds to push-button presses via interrupts.

Display format

The stopwatch counts time in hundredths of a second. The display uses four digits across `HEX3`–`HEX0`:

- `HEX3`: tens digit of seconds
- `HEX2`: ones digit of seconds, *with the decimal point lit*
- `HEX1`: tens digit of hundredths
- `HEX0`: ones digit of hundredths

For example, 3.57 seconds should display as `03.57`. You do not need to handle times larger than `99.99`. The leading tens-of-seconds digit may optionally be left blank when it is zero (e.g. `3.57`).

Behavior

- The stopwatch **starts running** as soon as the program begins.
- **KEY1** (button 1) is the **START/STOP** button. Each press toggles between running and paused. When paused, the display holds the current value and the timer does not advance.
- **KEY0** (button 0) is the **RESET** button. Pressing it stops the stopwatch and resets the display to `00.00`.

Requirements

- You **must** use **interrupts** for both the interval timer and the push buttons. Polling is not allowed.
- Set the interval timer period so that it fires every 10 ms (i.e. 100 times per second), and increment the hundredths counter once per timer interrupt.
- Enable the timer's timeout interrupt (`ITO`) and configure it in continuous (`CONT`) mode so it restarts automatically.
- Your exception handler must correctly determine whether an exception is an external hardware interrupt or a software exception, and dispatch to the appropriate ISR.
- Your main loop after initialization should simply wait for interrupts (e.g. an idle loop); all display updates must happen inside the ISRs.

Relevant addresses

Peripheral	Base address
Interval Timer	`0xFF202000`
Push Buttons	`0xFF200050`
HEX3–HEX0 display	`0xFF200020`
HEX5–HEX4 display	`0xFF200030`

What to submit A single file `stopwatch.s` containing your complete stopwatch program, including the reset vector, exception handler, initialization code, and all ISRs.

Submission Checklist

1. `stopwatch.s` — your complete stopwatch program

Commit and push this file to your Github Classroom repository. Check the `Actions` tab for the autograding output.