

**CPSC 359 – Spring 2018**  
**Assignment 1**  
**SNES Controller Device Driver**  
**25 points**  
**Due: Jun 5<sup>th</sup> @ 11:59pm (midnight)**

**Objective:** Build a simple device driver for a SNES controller. The driver will be used in the next assignment as the primary input device for your interactive game.

**Deliverables:**

1. Print creator name(s) at the beginning.
2. Print "Please press a button..."
3. Wait for user input.
4. If user presses a button, print a message on the console indicating the button pressed.
5. Loop back to step 2 if any key other than START is pressed.
  - a. You might assume that one key is pressed at a time.
  - b. Previous key needs to be released in order to read a new one, including the same button being pressed.
6. Pressing the "START" button will end the program displaying an exit message.

**Example session:**

```
Created by: John Smith and Sarah Smith

Please press a button...      (User Presses Joy-pad RIGHT button)

You have pressed Joy-pad RIGHT

Please press a button...      (User Presses Y button)

You have pressed Y

Please press a button...      (User Presses START button)

Program is terminating...
```

**Notes:**

1. To print a message on the console, call "**printf**".
  - a. It accepts one input **r0**: the address of the string to be printed.
  - b. The string must be null terminated, use "**.asciz**" directive.
  - c. The following example will print **Hello World!** on the console:

```
.section .text

main:
    ldr        r0, =hello
    bl         printf

stop: b         stop

.section .data

hello:
.asciz "Hello World!"
```

2. Use at least the following subroutines:
  - a. Init\_GPIO: the subroutine initializes a GPIO line, the line number and function code must be passed as parameters. The subroutine needs to be general.
  - b. Write\_Latch: write a bit to the SNES latch line.
  - c. Write\_Clock: writes a bit to the SNES clock line.
  - d. Read\_Data: reads a bit from the SNES data line.
  - e. Read\_SNES: main SNES subroutine that reads input (buttons pressed) from a SNES controller. Returns the code of a pressed button in a register.
3. Submit a tar-ball of your entire project directory, including makefile, source code, objects, etc.

**Grading:**

1. Display creator names & messages	1
2. Correctly reading/printing buttons	10
3. Following APCS	4
4. Using subroutines	6
5. Loop back (not "START")	2
6. Well documented code	2
<b>Total</b>	<b>25 points</b>

Programs that do not compile cannot receive more than **5 points**. Programs that compile, but do not implement any of the functionality described above can receive a maximum of **7 points**.

**Teams:** You may work with two other students in class in order to complete the assignment, but you are not required to do so. Peer evaluation in teams may be conducted.

**Demonstration & Submission:** Submit a .tar.gz file of your entire project directory (including source code, make file, build objects, etc) to your TA via the appropriate dropbox on Desire2Learn. Only one submission per team is required. You will also need to be available to demonstrate your assignment during the tutorial.

**Late Submission Policy:** Late submissions will be penalized as follows:

-12.5% for each late day or portion of a day for the first two days

-25% for each additional day or portion of a day after the first two days

Hence, no submissions will be accepted after 5 days (including weekend days) of the announced deadline.

**Academic Misconduct:** Any similarities between assignments will be further investigated for academic misconduct. While you are encouraged to discuss the assignment with your colleagues, your final submission must be your own original work. Any re-used code of excess of 20 lines (20 assembly language instructions) must be cited and have its source acknowledged. Failure to credit the source will also result in a misconduct investigation.

**D2L Marks:** Any marks posted on D2L are tentative and are subject to change (UP or DOWN).