Project #3

Michael Schoen, Abdirahman Osman, Illya Starikov

Due Date: Tuesday, December 6th, 2016

For the second project, we were delighted to be able to use a higher level programming language; we[1] decided to apply this new-found excitement to implement a retro game from the 70s: Space Invaders.

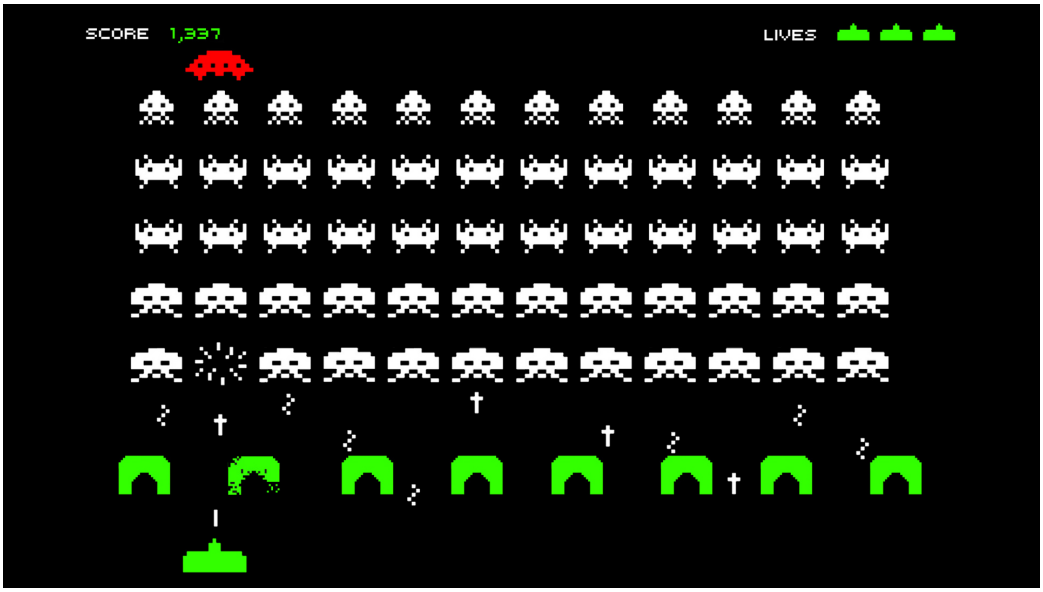

Figure 1 – The original space invaders.

1 Project Description

Below we will go into more detail about each individual parts of our project.

[1] Illya.

1.1 The Game

The game we initially decided to go with was space invaders. We had intention of doing the game logic on the microcontroller through serial communication; however, we learned early that this was likely not be possible[2]. We describe in *Problems Encountered* how we absolved this. From here, we decided our game would exclusively be in the terminal.

Our game essentially creates a two dimensional array (in three segments — the header to show score and level, the aliens, and the shooter). Then we loop through depending on the input:

←	Move the shooter left.
→	Move the shooter left.
q	Quits the game
Space	Shoots with the gunner.
	No input meant refresh game.

We achieved the drawing through curses[3]. Ultimately, we did not get our game fully done, but a good majority of it.

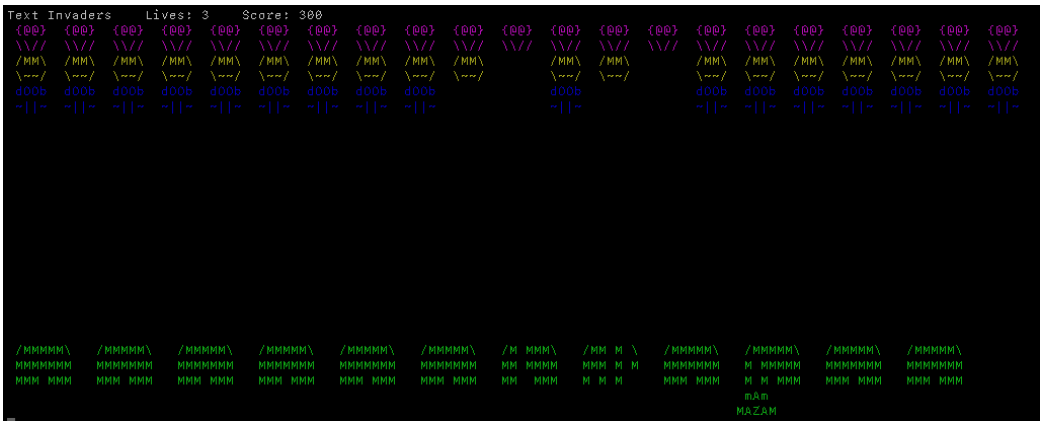

Figure 2 – Text Invaders, our version of space invaders..

[2]Our hex file with very basic functionality was 25 kB.

[3]Can be read about here `https://en.wikipedia.org/wiki/Curses_(programming_library)`

1.2 Music

The formula for each note is

$$\frac{1/4 \text{ Oscillator Frequency}}{\text{Note Frequency}}$$

. To produce the music, we had a function that took in which note and the type of note (16^{th}, 8^{th}, 4^{th}, half or whole note). The function multiplied the base length of a note by a constant depending on the type of note. A 16^{th} note would be multiplied by 1, an 8^{th} by 2, a quarter note by 4, a half-note by 8, and a whole note by 16. We used timers and interrupts for the period of the note and the length of the note.

We used sheet music to get the notes. For speeding and slowing down the music, the type of note was just multiplied by 2 or divided by 2 respectively.

1.3 Seven Segment Display

For the seven segment display, we had to set P_2 to be push/pull so that the display would receive enough power. We then looked up the correct values to output to the pins to make each number show up on the display and used those values in the increment/decremented function as well as the simple function that lets you display any number. We used the data sheet for the seven segment display to see how to connect it on the breadboard.

2 Problems Encountered

Below we will list some of the "several" problems we ran into.

2.1 Memory Constraints

By far, the biggest problem we encountered was the memory constraint. The 8051 has 8 kB of memory; our code base, with the exclusion of all the `malloc`s[4], it is roughly 25 kB — a size a bit larger than the 8051 allows. Our workaround was to fight fire with fire.

[4]Since `unsigned char` = 1 B, the terminal window will roughly be 20 height × 80 width, we can roughly expect 1.6 kB to be allocated on the heap; a non-insignificant amount compared to 8 kB.

Instead of "dumbing" down our game[5] to get it to fit, we decided to have an external interface; specifically, a port sniffer. It would listen for input from the 8051, and if there is a signal on the serial port, use that as input. If not, default to the keyboard input.

Ultimately, we were unsuccessful with the port sniffer. Originally, we had tried to use a Linux port sniffer so we can just embed it into our program, `slsnif`[6]. Unfortunately this port sniffer does not support "legacy" ports, and the 8051 falls in this category. So we moved onto a Windows port sniffer, `Serial Input For Windows`[7]. This too did not work, because we could only have one interface use the serial port, so we would need a dedicated socket to intercept the `COM1` port's input — something we were not familiar with.

Ultimatly, we were

3 Individual Features

- Michael Schoen — 33% Contribution
  - All Game Sounds
  - Game Music
- Abdirahman Osman — 33% Contribution
  - Port Serialization
  - Menu Logic
- Illya Starikov — 33% Contribution
  - Space Invaders Game

4 8051 Architecture

[5]According to back of the hand calculations, a space-optimized version of the game would still be 7 kB. This was likely to be impossible.

[6]Can be found at `https://sourceforge.net/projects/slsnif/`.

[7]Can be found at `http://www.randomnoun.com/wp/2013/02/03/serial-input-for-windows/`.