CSE 111 Project - Team 12

Contents

Team Members - Achint, Ryan, Sophia, and Ugo

1 Team Member Contribution Summary

1.1 Achint - bluekid2457

- Coding:
- Debugging & Misc.: Helped with input issue
- Github:

1.2 Ryan - rfdong4

- Coding: Coded some of memory. Coded majority of CPU code including opcodes, step function, reset function. Coded a lot of GPU code including render function (no input) and cleanup function
- Debugging & Misc.: Debugged the gpu issue, input issue (not registering "A" input), helped debug cpu loop issue
- Github: Made the Repository & Organization

As our group started the project, I created the github repositories, the final project and the repository for the starter code, and the github organization. I invited all of group members and class instructors to the repository. As check-in 1 was approaching, I began to copy the starter code given in class by Ethan

to begin the memory module in the emulator. I created the BananaMemory class and header files. I also created the BananaMemory initialization. After this, I began working on the CPU. I created the BananaCpu header and cpp files. The first thing I did was create the function that executes the instruction. This function consisted of a large switch statement that read the opcode of the instruction and executed the instruction using the memory functions that were coded by Ugo. I used the assignment doc to correctly map each instruction with the correct opcode. This process was tedious as the switch statement was incredibly long and difficult to read. Ashint helped by turning all the individual instructions into functions. This process cleaned up the code. To get the CPU to function properly, I created run function (main game loop later moved to main console file), the fetch_instruction function (function that fetches the instruction from the address at the program counter and increment program counter by 4), and the step function (function that calls fetch instruction and execute instruction). I also coded the reset function that sets program counter to address_to_setup and sets all registors in the memory registers array to 0. With these functions added, hello world 1 was able to work. To get hello world 2 and 3 to work, my group got into a discord call to debug the cpu file. We found out that we were not properly looping the program counter correctly. We fixed this bug by bringing the program counter back to address_to_loop every time the program tried to stop executing. Next began the work on the GPU class. I created the GPU class and began working on the render function using SDL. I finished the render function so that image.png was able to be rendered. Ugo finished the input but there was a bug where the "A" button was not registering in the input file. This was a common bug amongst other groups and it was fixed when I changed the memery array to signed integers as opposed to unsigned integers. Next there was a bug that caused glitching and flashing to happen when playing an emulated game. This bug was caused because we were calling the render() function after every time the cpu "stepped". This was a mistake as the render function is supposed to be called after every game loop. Once the I moved the render function, the games ran smoothly. At this point the emulator's basic functionality was almost complete. The only problem was the games ran incredibly fast. This problem was solved when I added an SDL_delay to the main game loop. I am proud of everything I contributed to the group project. I am most proud of the CPU because it was very interesting breaking down the functionality of a CPU and coding it. Coding this emulator was unlike any school project I have done before and I had a lot of fun with it.

1.3 Sophia - SophiaRainielAguado

- Coding:
- Debugging & Misc.: Helped with debugging pixel issue,
- Github: Helped with installing Github & navigation

1.4 Ugo - -> uezeokoli

- Coding: Created the pixel function, created input function
- Debugging & Misc.: Helped with input issue
- Github:

2 Check-in Summaries

2.1 Check-in 1

- Required: CPU and Address Space completed. Three "Hello, World!" programs should run. Issue: (we had CPU.run instead of console.run) Run function was in CPU instead of console, had to add const to non-changing variables, changed to consistent casing (snake_case), changed switch cases to be different functions. Represented the instructions as a class or struct.

2.2 Check-in 2

- Required: GPU and Controller Input completed. Issues: Calling render inside the loop but should have called after every time loop was called. Which caused pixel to be slower on certain systems and not update correctly when the game was run. Resulting in screen glitching.

Input had to be changed from unsigned to signed as one of the buttons wasn't being read. Register array in BananaCPU changed from uint16_t to int16_t

2.3 Extra Features

- Screen Record, Pixel Filter, GUI

Things to Include

- Document the Extra Features that your team added
- Highlight cool applications of course-related material such as advanced C++ or Github features
- Include a summary of each Team Member's contributions
- Explain if something isn't working. What steps did you take to debug? What do you think could be the issue?