Caleb Chang, James Huang, Angela Seidel, Kiet Truong
Dr. Dincer
31 October 2020
PRJ#02b: Group Answers

PRJ#02b - Activity 1) Revise your Project Plan (10 points) (<1 page)

Prepare a revised project plan & schedule that will cover the whole project time until the final delivery.

Change Summary:

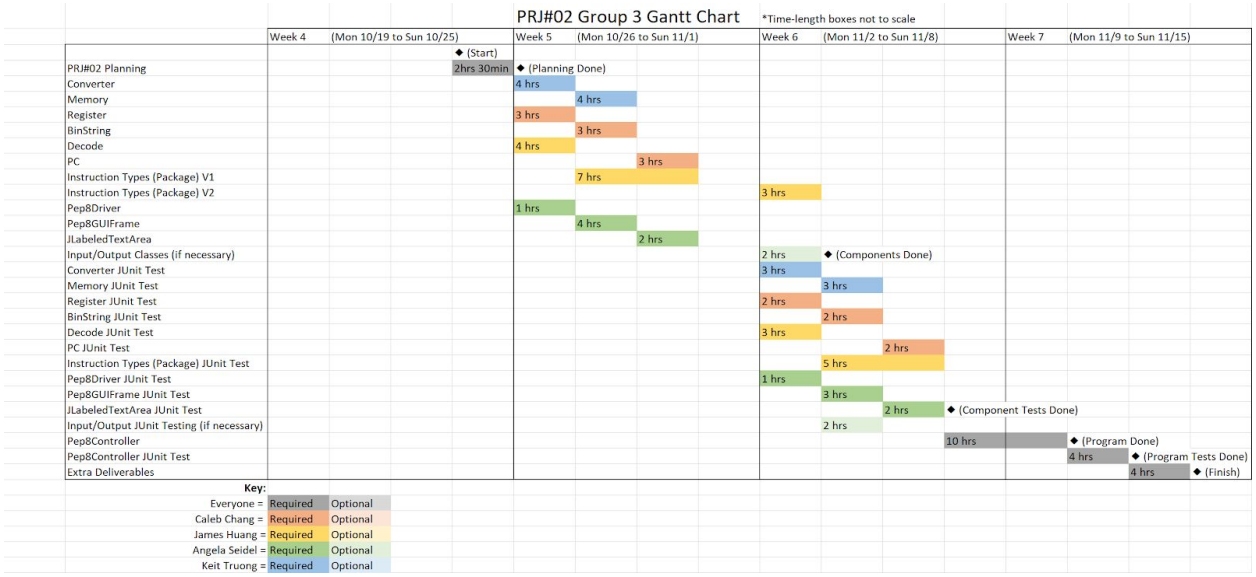
- James split the InstructionType package into two phases of development since it is large.
- Some people are done with unit testing after week 2, and are ahead of schedule.
- Angela might make input and output classes if working directly with JTextAreas is tricky for interpreting input and output data, as separate classes were described in the assignment requirements.

Table Calendar:

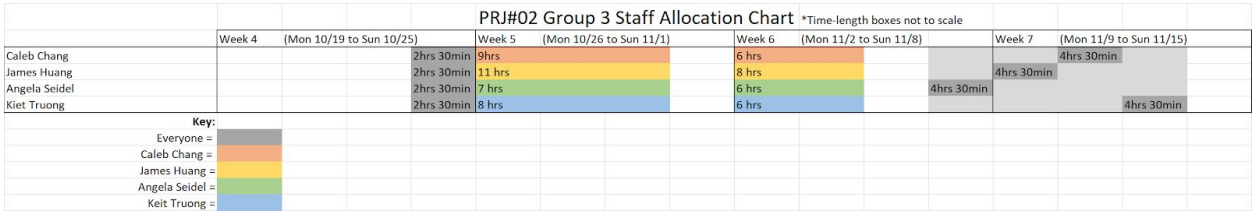
	Week 1	Week 2	Week 3	Week 4
Angela	PRJ#02a done planning	-) GUI Planning -) Complete Pep8Driver -) Complete Pep8GUIFrame -) Complete Pep8JLabledTextArea	-) Maybe create separate input and output classes so data management can be easier -) JUnit Testing of GUI classes -) Maybe start working on Controller	-) Work on controller -) Work on controller JUnitTesting -) Work on any other extra deliverable requirements for final submission (demo videos, etc.)
Caleb	PRJ#02a done planning	Register, BinString, PC	JUnit Testing Controller	Work on Controller
James	PRJ#02a done planning	Decode,instructionType package V1	InstructionType package V2 (Execute method implementation)Junit testing ,controller	Work on Controller

Kiet	PRJ#02a done planning	Converter/Memory	-)JUnit testing assigned classes. -)*helping out wherever needed*	Work on Controller

Gantt Chart:



Staff Allocation Chart:



PRJ#02b - Activity 2) Prepare a Software Requirements Specification (SRS) (20 points) (multiple pages)

Arrange a group meeting towards the end of the week, and present the requirements specifications that each of you prepared individually with your group members.

Decide as a group on the scope of the project (i.e., how much of PEP/8 functionality you are going to support in your application).

Work with your group members and come up with a unified single version of the requirements that you will follow in this project. Document your requirements document using a proper combination of specification techniques.

Functional Requirements

Requirements from Assignment Document:

- The simulator shall be able to interpret machine language instructions (object code) and assembly language instructions (source code).
- The simulator shall implement the seven machine language instruction of stop execution, load operand into A register, Store the contents of the A register in the operand, add the operand to the A register, Subtract the operand from the A register, character input to the operand, character output from the operand that was required in PRJ#01.
- The simulator GUI shall display the internal state of the machine (CPU area in original Pep/8 Program) and shall display the state of the memory.
- The project classes shall be organized using the Model-View-Controller design pattern.
- The Controller class shall be the class that connects the interactions between all Model-related classes and the Visual-related classes.
- The classes of the program shall mimic the von Neuman computer architecture:
    - The Input Device will be represented by the Input JTextArea stored in the GUI.
    - The Control Unit will be represented by the Controller class.
    - The Arithmetic/Logic Unit will be represented by the Instruction Types package whose components will be controlled by the Controller class.
    - The Output Device will be represented by the JTextArea stored in the GUI.
- The Number Classes requirement shall be implemented by having the Converter Class which will manage the translation of decimal, hex, and binary values throughout the Controller class's manipulation of model and view objects.

- The Calculator Classes requirement shall be implemented through the presence of the Instruction Types package which will help with basic operations like addition and subtraction of values.

Requirements from Unique Ideas:

- The user shall be able to type in JTextAreas to document what assembly code, machine code, or inputs they want to be processed by the Pep/8 Simulator.
- The user may be able to upload .txt files in place of typing into JTextAreas for easier input.
- The Memory shall have its information stored as Binary strings in an array, with elements that are 2 bytes in length.
- The GUI Memory display shall display the first memory location in hex in the first column and should display the contents of eight element locations in the second column in hexadecimal format, much like the original Pep/8 program.
- The GUI Memory display might display the third column in the original Pep/8 Program, which translates hex values into printable ASCII symbols.
- The user shall be able to press the “Run Source Code” button to assemble the source code into the object code, then load the object code into memory, then execute the code in memory.
- The user shall be able to press the “Run Object Code” button to load the object code into memory, then execute the code in memory.
- The user shall be able to press the “Clear Memory” button to clear the contents of the Memory.
- The user may be able to press a “Step Source Code” button to run the program one instruction step at a time.
- The user may be able to press a “Step Object Code button to run the program one instruction step at a time.

Non-Functional Requirements

Requirements from Assignment Document:

- The software shall be developed using IntelliJ IDE.
- The software shall have different finalized builds uploaded to GitHub for everyone to access.
- The software shall use Jira to schedule meetings and track progress of completion through tickets.
- The software class components and finalized system shall be tested using JUnit 5.

- There shall be no restriction to how big the IntelliJ project file size of the program should be.

Requirements from Unique Ideas:

- The Memory array shall be at least 1000 elements long, with potential to go up to 65,535 elements if the program is fast enough to accommodate that size.
- The “Run Source Code” button shall take no more than 3-5 seconds to run the whole program.
- The “Run Object Code” button shall take no more than 3-5 seconds to run the whole program.
- The “Step Source Code” button, if it is implemented, should take no more than 1 second to run one step of the program.
- The “Step Object Code” button, if it is implemented, should take no more than 1 second to run one step of the program.
- The “Clear Memory” button shall take no more than 3-5 seconds to clear the whole program.
- The program shall be able to open in 10 or less seconds.
- The Controller class shall be the main class that uses the model and view classes as a connecting class. All the other classes should try to reference each other as little as possible and leave the Controller to do the bulk of the connecting interactions.