Group 1

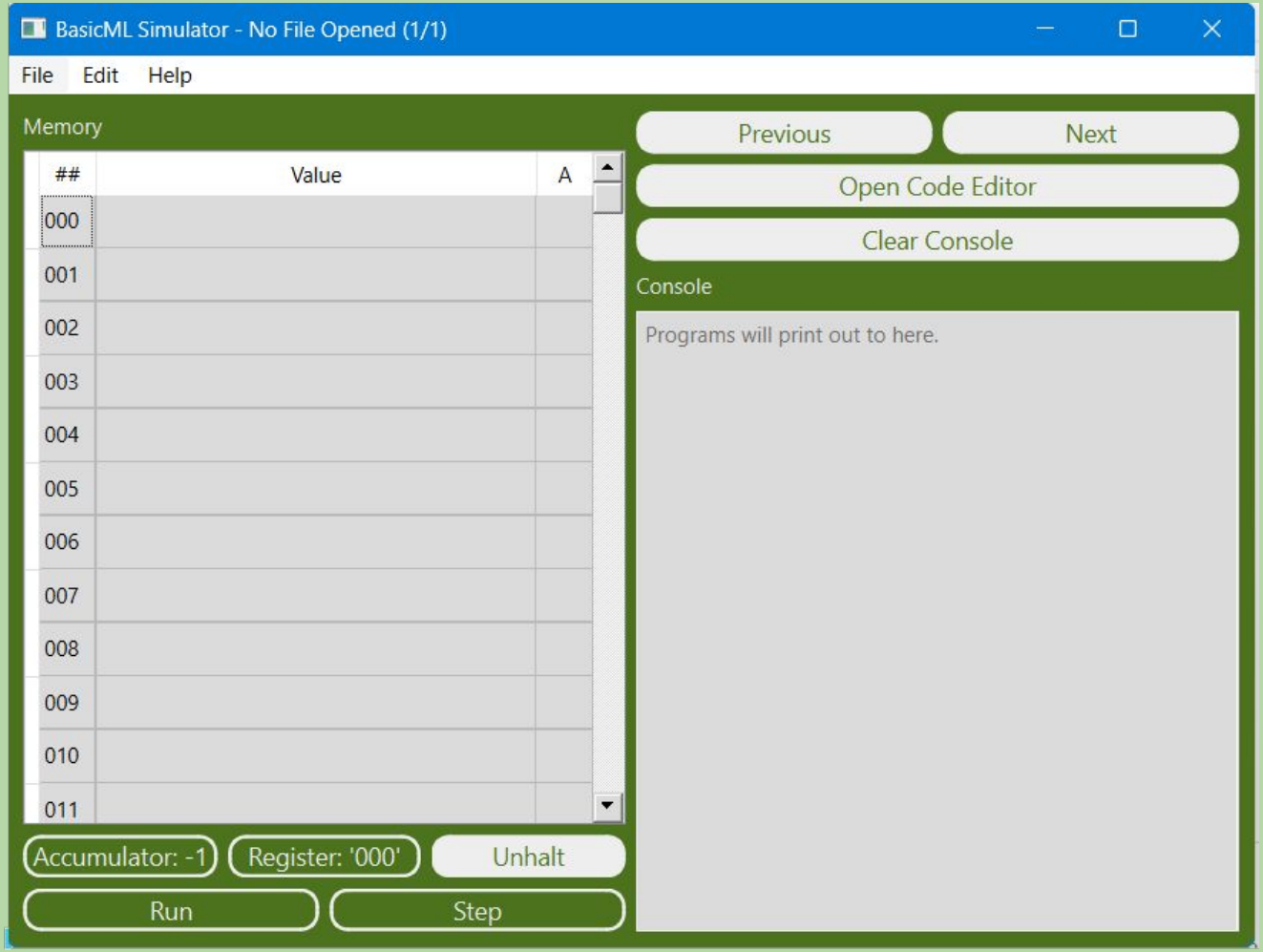
UVSim

UVSIM Application

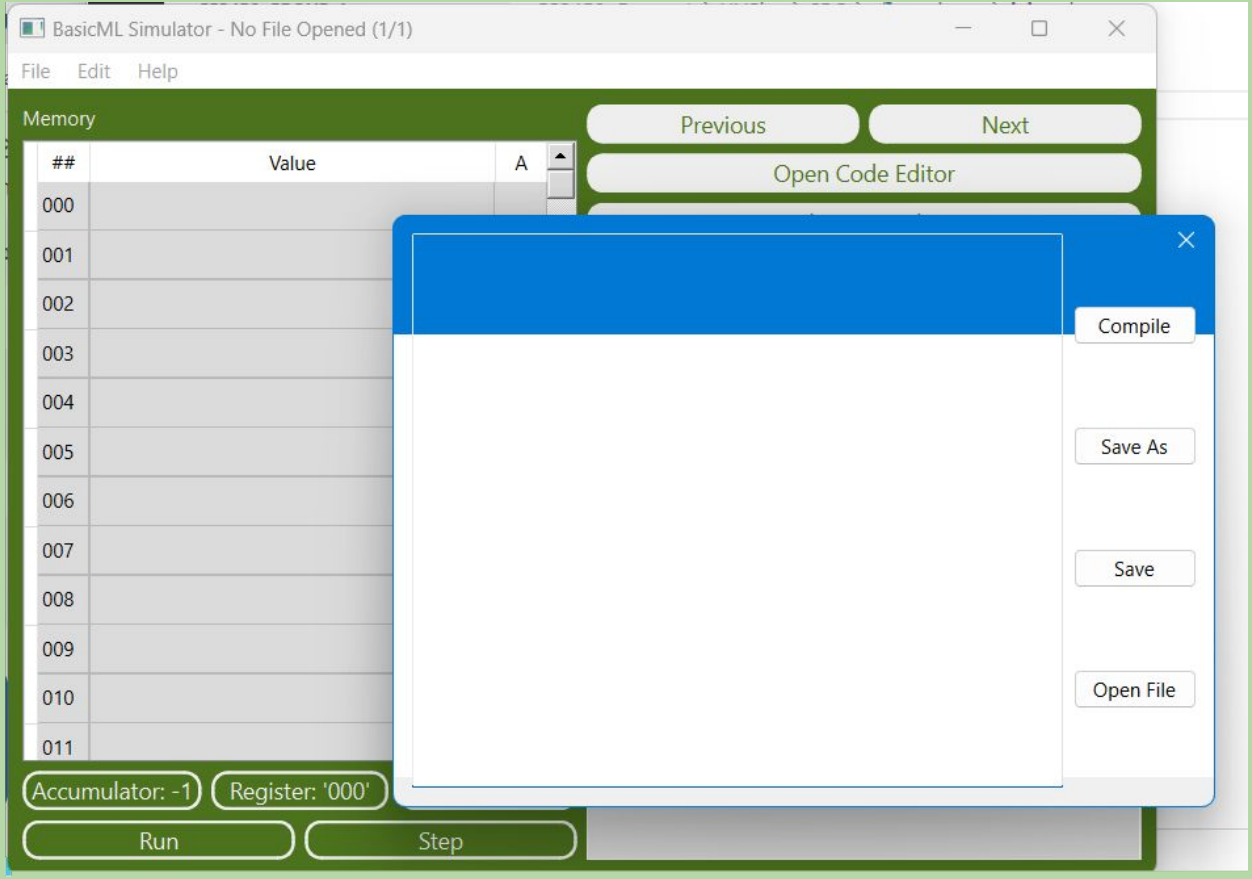
Design for student learning

This application assists those learning ‘Machine Learning Language’

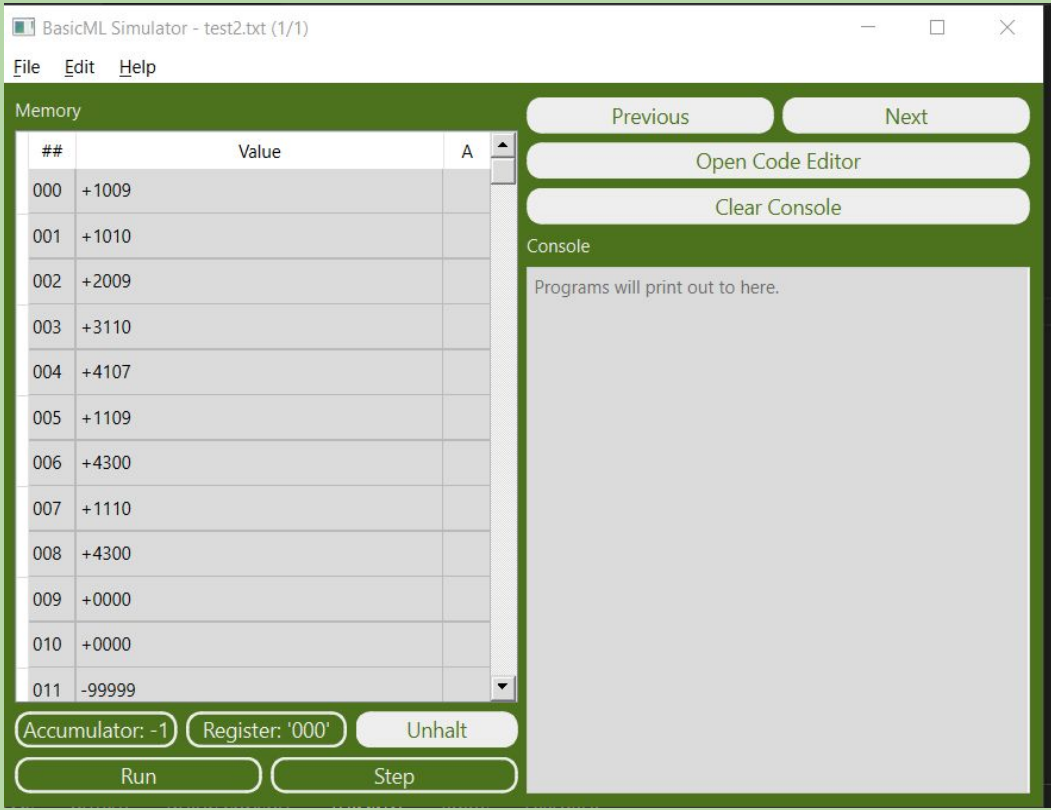
Contains CPU, register, and main memory. An accumulator – a register into which information is put before the UVSim uses it in calculations or examines it in various ways. All the information in the UVSim is handled in terms of words.

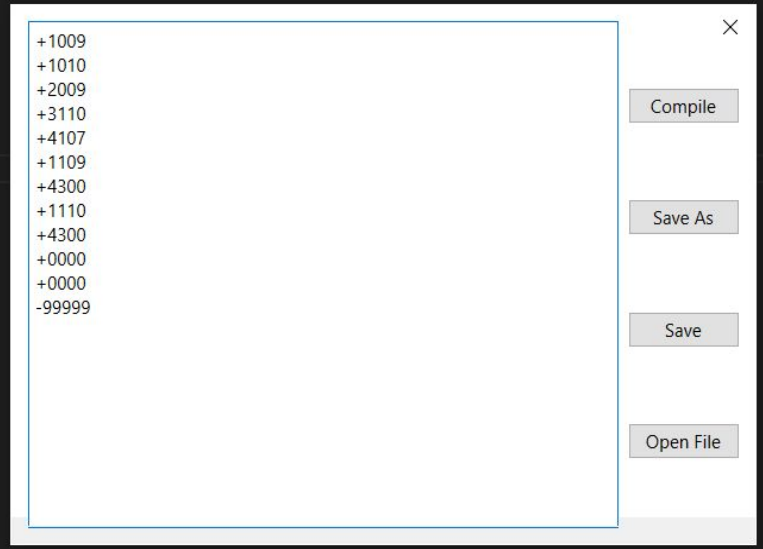

BasicML Simulator - No File Opened (1/1)
File
Edit
Help
Memory
Value A
000
001
002
003
004
005
006
007
008
009
010
011
Accumulator: -1
Register: '000'
Unhalt
Run
Step
Previous
Next
Open Code Editor
Clear Console
Console
Programs will print out to here.

Code Editor:


BasicML Simulator - No File Opened (1/1)
File
Edit
Help
Memory
Previous
Next
Open Code Editor
##
Value
A
000
001
002
003
004
005
006
007
008
009
010
011
Accumulator: -1
Register: '000'
Run
Step
Compile
Save As
Save
Open File

BasicML Program





Facade:

The facade pattern is typically used when a simple interface is required to access a complex system, a system is very complex or difficult to understand, an entry point is needed to each level of layered software, or. the abstractions and implementations of a subsystem are tightly coupled.

- Controller

CS2450_Group_1 > UVSim > SRC > main.py > ...

```python
from controller import Controller

def main():
    """Main function"""
    _ = Controller()


if __name__ == '__main__':
    main()

```

```python
class Controller():
    def __init__(self) -> None:
        #Simulation text source
        self.current_file = 0
        self.sim_editors = ['']
        self.file_paths = ['']
        self.buffers = [buffer.Buffer()]
        self.sims = [I_UVSim(UVSim(self.buffers[0]))]
        self.sim_editor = self.sim_editors[self.current_file]
        self.file_path = self.file_paths[self.current_file]

        #Simulation interface
        self.buffer = self.buffers[self.current_file]
        self.sim = self.sims[self.current_file] # Create a Inte

        # GUI dispaly
        app = QApplication(sys.argv)
        self.gui = QTGUI()
        self.gui.show()
        self.button_activation()
```

-Buffer

```python
class Buffer():
    def __init__(self):
        self._buffer_bit = 0
        self._buffer_location = 0
        self._buffer_message = ''

    def set_buffer(self, bit = 0, location = 0, message = ''):
        self.set_buffer_bit(bit)
        self.set_buffer_location(location)
        self.set_buffer_message(message)

    def set_buffer_bit(self, bit = 0):
        self._buffer_bit = bit

    def set_buffer_location(self, location = 0):
        self._buffer_location = location

    def set_buffer_message(self, message = ''):
        self._buffer_message = message

    def get_buffer(self):
        return self.get_buffer_bit(), self.get_buffer_location(),  self.get_buffer_messa

    def get_buffer_bit(self):
        return self._buffer_bit

    def get_buffer_location(self):
        return self._buffer_location

    def get_buffer_message(self):
        return self._buffer_message
```

Facade Continued: