# Class Definition Document

### Global Constant:

MEMORY\_SIZE: A constant variable that determines the size of our memory array.

## UVSim Class:

The UVSim class stores the main memory and accumulator of our simulator. It also stores which instruction we are currently on using the curr variable. UVSim has the responsibility of parsing the instructions provided by the user and executing them.

### Attributes:

- main\_memory: An array of the size MEMORY\_SIZE that contains shorts that represent the instructions provided by the user.

- accumulator: A register for the user to store a short in.

- curr: Counter to show which instruction the program is on in the main\_memory array.

- input: An object that reads a file into a vector for UVSim to parse into instructions.

- operator: An object that contains all the methods pertaining to each OP CODE.

### Methods:

- reset\_memory(): Reset each short in the main\_memory array to 0.

- execute(): Go through each instruction in the main\_memory array and use the split\_instr() function to get the op\_code and memory address of each instruction in turn. Then call execute\_op to run the instruction.

- split\_instr(instr:short, op\_code:short\*, mem\_addr:short\*): The parameter instr contains the instruction. This function will put the two digits in the 100 and 1000 place of this instruction into the op\_code variable and the two in the 01 and 10 place into the mem\_addr. These variables will be used elsewhere.

- parse\_file(lines:vector<string>): Determine if the vector is the correct size then take only valid instructions. (valid instruction layout: +0000; sign, two digits for op code and 2 for memory address.) Put each instruction into the main\_memory array.

+ execute\_op(op\_code:short\*, mem\_addr:short\*, curr:short)->unsigned short: Parameters: op\_code: this is the opcode of the current instruction in main\_memory. Mem\_addr: this is the memory address that the instruction targets. Curr: counter for which main\_memory instruction we are currently on. Functionality: Use the provided op\_code to call the correct method in UVSim\_OP object, provide mem\_addr of the instruction as well. Increment curr to the correct instruction in main\_memory.

+ get\_accumulator()->short&: Returns the private accumulator.

+ get\_memory()->short\*: Returns the private main\_memory array.

+ get\_memory\_value(mem\_addr:short)->short: Returns the value at a memory address in the main\_memory array.

+ set\_accumulator(value:short): Sets the accumulator to the desired value.

+ set\_memory\_address(mem\_addr:short, value:short): Set the value of the provided memory address to the desired value.

+run(): This method is used to start the UVSim. It will get the instructions out of the file using the input objects read\_file method, Parse the instructions and put them into main\_memory with parse\_file, and will then call execute to begin executing the files instructions.

## UVSim\_OP Class:

### Attributes:

NONE – This is a stateless class

### Methods:

+ divide(accumulator:short&, main\_memory:short\*, mem\_addr:short): Divide the accumulator by the number that is at the memory address mem\_addr in main\_memory.

+ multiply(accumulator:short&, main\_memory:short\*, mem\_addr:short): Multiply the accumulator by the number that is at the memory address mem\_addr in main\_memory.

+ add(accumulator:short&, main\_memory:short\*, mem\_addr:short): Add the number at the memory address mem\_addr in main\_memory to the accumulator.

+ subtract(accumulator:short&, main\_memory:short\*, mem\_addr:short): Subtract the number at the memory address mem\_addr in main\_memory from the accumulator.

+ branchNeg(accumulator:short&, curr\_addr:short, br\_target:short): If the number in the accumulator is negative, return the branch target. Otherwise return the curr\_addr. Curr is set to the return value.

+ branchZero(accumulator:short&, curr\_addr:short, br\_target:short): If the number in the accumulator is zero, return the branch target. Otherwise return the curr\_addr. Curr is set to the return value.

+ branch(br\_target:short): Return br\_target. Curr is set to the return value.

+ load(accumulator:short&, main\_memory:short\*, mem\_addr:short): Load the value at mem\_addr in the main\_memory array into the accumulator.

+ store(accumulator:short&, main\_memory:short\*, mem\_addr:short): Store the value in the accumulator into the memory address mem\_addr of the main\_memory array.

+ read(is:istream&, main\_memory:short&, mem\_addr:short): “is” is used to read in user input. Input in the form of an instruction is put into main\_memory at mem\_addr.

+ write(os:ostream&, main\_memory:short&, mem\_addr:short): The instruction or number at mem\_addr in main\_memory is output to the user.

## UVSim\_Input Class:

### Attributes:

NONE – This is a stateless class

### Methods:

+ read\_file()->vector<string>: Open the file, use read\_from\_stream to read the content of the file. Return the vector from read\_from\_stream.

+ read\_from\_stream(is:istream&)->vector<string>: Read the file passed in as a parameter to put each line into the vector of strings. Return the vector.

