Unit Tests - Group B : UnitTests

	A	B	C	D	E	F
1	**Name**	**Description**	**Use Case(s)**	**Inputs**	**Expected Outputs**	**Conclusion (How we know it works)**
2	test_file_open	Tests whether a file can be read by the program.	Retrieving a file necessary for program operation	Test1.txt' (assumes this file exists), and 'qwertyuiop.txt' (assumes this file does not exist)	No output if it succeeds, raises OS error if the file does not exist	We are able to run the program and verify that it works. The test also passes when attempting to open a file that exists.
3	test_program_init	Tests that a correctly formatted file is properly read into memory.	Retrieving a file necessary for program operation	Any existing file that is formatted correctly (in this case uses 'Test1.txt' and 'Test2.txt'	Compares values in memory with expected values, fails if values do not match	Test passes, and test print output in the main program displays expected values in memory.
4	test_negative_values	Tests that negative values are added (subtracted) correctly in the accumulator.	When negative values are input into memory.	100 into accumulator, values of -30 and -50	Accumulator = 20	Accumulator successfully contains the value '20' after negative values are input. (100-30-50 = 20)
5	test_subtract_negative	Tests that subtracting negatives leads to adding in the accumulator.	Program encounters a 'subtract negative' situation	100 into accumulator, values of -30 and -50	Accumulator = 180	Accumulator contains expected value after subtracting -30 and -50 from 100.
6	test_bad_word_format	Verifies that improper words cannot be written into memory, and checks that the program cannot run if memory somehow does contain improper text.	Bad values are written to memory.	Write "TEST" into memory, run a program containing "TEST" in memory	Type errors in both cases.	The tests return errors when attempting to write invalid words into memory, and when trying to run a file containing invalid words.
7	test_improper_word_in_file	Verifies that invalid words contained in the designated file raise errors and are not allowed by the program.	Invalid words exist in the text file.	A text file containing too many characters, and a text file containing a word that should not be recognized by the program.	Value errors in both cases.	The test returns value errors when attempting to read in a file containing words of the wrong length or words that cannot be recognized.
8	test_branch	Test that makes sure branch moves pointer to designated part in memory	Branch to specific part in memory	4005, 05 for branch function	Program counter moves to 05	Test function asserts that counter is at memory location 05 after using BRANCH 4005
9	test_branch_executes_until_halt	Test that makes sure program still runs after branching.	Branch to specific part in memory	4001, 01 for branch function, 4300 for HALT function	Program moves counter to 01 then Halts	The HALT successfully passes after branching. test_branch verifies the branch moves the counter.
10	test_write_prints_hello_world	Test that proves the program can create basic programs such as printing Hello World.	Print Hello World to console	"Hello World" into memory.	Program prints "Hello World" from memory to console	Program prints 'Hello World" from memory to console.
11	test_input_then_print_hello_world	Test that shows user inputs are correctly saved into memory and can be printed to console.	Print Hello World to console	User inputs 'Hello World" into memory using Read.	Program prints "Hello World" from memory to console	Program prints 'Hello World" from memory to console.
12	test_load_and_store_value	Checks that load and store are working as expected.	Load and Store Data to memory and accumulator.	Inputs 42 at memory position 10, stores 42 into accumulator, which then loads value into memory position 11.	Accumulator = 42, Memory[11] = 42	The accumulator and memory [11] can only = 42 if the LOAD and STORE opcodes are working properly.
13	test_invalid_address_raises	Checks that program catches invalid addresses and raises and IndexErrorl.	Load and Store Data memory and accumulator.	Input: 99 at memory location 200	IndexError	200 is out of range, and so the program showing an IndexError means it is handling memory correctly.
14	test_multiply_two_values	Verifies that the Multiply opcode is working and produces correct output.	Utilize Multiplication Opcode	Inputs: 6 in Accumulator, 7 to Memory	Accumulator = 42	7 * 6 = 42, verifies the correct output and memory location (accumulator) of the multiply opcode.
15	test_multiplication_with_zero	Further functionality test of the multiply opcode with zero.	Utilize Multiplication Opcode	Inputs: 0 in Accumulator, 5 to Memory	Accumulator = 0	5 * 0 = 0, verfieis the correct output of the multiply opcode.
16	test_division_two_values	Verifies that division opcode is working and produces correct output.	Utlizie Division Opcode	Inputs: 42 in accumulator, 7 to memory	Accumulator = 6	42 / 7 = 6, verifies the correct output of the division opcode
17	test_division_with_zero	Checks that trying to divide accumulator by zero produces a ZeroDivisionError.	Utilize DIvision Opcode	Inputs: 10 in accumulator, 0 to memory.	ZeroDivisionError	A ZeroDivisionError proves that the program is not moving ahead with erroneous computations.
18	test_memory_write_and_read	Verifies that memory can be written to and read from, and that invalid accesses raise errors.	Writing and reading data to/from memory.	write(10, 1234), then read(10); Write (150, 9999) (will raise IndexError)	Read from address 10 returns 1234; writing to 150 raises IndexError.	Successfully reads and writes within bounds; error is raised when accessing out-of-bounds memory, confirming memory limits are enforced.
19	test_load_store	Verifies that the CPU can load data from memory into the accumulator and store it back into another memory location.	Data transfer between memory and CPU	memory[5] = 4321; LOAD from 2005, STORE to 2106	accumulator = 4321, memory[6] = 4321	The CPU correctly transfers data between memory and accumulator.
20	test_add_subtract	Checks that ADD and SUBTRACT opcodes update the accumulator as expected.	Arithmetic operations in CPU	acc = 100, memory[1] = 50, memory[2] = 25	accumulator = 125 after ADD then SUBTRACT	The accumulator holds the correct result after sequential arithmetic.
21	test_multiply_divide	Verifies correct functionality of MULTIPLY and DIVIDE opcodes.	Arithmetic operations in CPU	acc = 10, memory[3] = 5, memory[4] = 2	accumulator = 25 after multiply/divide	Confirms correct execution of compound arithmetic logic.
22	test_branch_operations	Tests conditional and unconditional branch instructions.	Instruction control flow	acc = -1, then 0, then 5; various BRANCH, BRANCHNEG, BRANCHZERO opcodes	program_counter = 7, 9, 2, 1 accordingly	Asserts that program counter updates correctly based on accumulator state.
23	test_halt	Ensures HALT opcode stops execution.	End of program execution	4300	cpu.halted = True	Confirms that the halt flag is set when HALT is executed.
24	test_bad_opcode	Ensures an invalid opcode raises an appropriate error.	Error handling for invalid instructions	ex. 5555	ValueError	Program correctly identifies and rejects unsupported opcodes.
25	test_bad_word_length	Validates that too-long memory words raise errors.	Memory validation	memory.write(5, 12345)	ValueError	Writing a 5-digit word fails validation, as expected.