Part I - Test Cases

REQUIREMENT TEST ID	TEST				
TEST-1.0	Pre-Condition: have an SML file Requirement: must be able to open SML files Input: vaild SML file Output: open SML file				
TEST-1.1	Pre-Condition: alphanumeric characters in file name Requirement: must be able to open SML files Input: vaild SML file Output: open SML file				
TEST-1.2	Pre-Condition: special characters in file name Requirement: must be able to open SML files Input: SML file with all special characters: !@#$%^&()_+=[]{};',.`~.sml Output: open SML file				
TEST-1.3	Pre-Condition: have an SML file with no instructions or intial values for variables used Requirement: does not open file Input: invalid SML file Output: error message, indicating stupidity				
TEST-2.0	Pre-Condition: have an open/working SML file Requirement: must be able to close SML file Input: open sml file Output: closed sml file (aka nothing more showing), but maybe a message declaring success				
TEST-3.0	Pre-Condition: have working SML file Requirement: must be able to read SML files Input: valid SML file Output: SML file content				
TEST-4.0	Pre-Condition: open SML file Requirement: must perform the fetch/decode/execute instruction cycle on SML instructions Input: "prog1.sml" Output: N/A				
TEST-5.0	Pre-Condition: program started Requirement: must display program and data memory Input: N/A Output: pixels				
TEST-6.0	Pre-Condition: program started Requirement: must display the instruction counter, instruction register, accumulator, operation code, and operand Input: N/A Output: instruction counter, instruction register, accumulator, operation code, and operand				

TEST-7.0	Pre-Condition: program started Requirement: must allow for single step execution of instructions Input: 10 commands Output: processes each opcode after user indicates				
TEST-8.0	Pre-Condition: computer has memory Requirement: must load a SML program into memory Input: Simpletron Output: Simpletron proram is loaded into computer memory				
TEST-9.0	Pre-Condition: program started Requirement: perform NO_OP Input: opcode 0 Output: <none>				
TEST-9.1	Pre-Condition: program started Requirement: perform READ Input: opcode 10, user input 10 Output: 10 stored in memory				
TEST-9.2	Pre-Condition: program started Requirement: perform WRITE Input: opcode 11 Output: displays the value stored in memory				
TEST-9.3	Pre-Condition: Program started Requirement: Perform LOAD Input: opcode 20, 10 in data memory Output: 10 in accumulator				
TEST-9.4	Pre-Condition: program started Requirement: perform STORE Input: opcode 21, 10 in accumulator Output: 10 in data memory				
TEST-9.5	Pre-Condition: program started Requirement: perform ADD Input: opcode 30, 10 in accumulator, 5 in data memory Output: 15 in accumulator				
TEST-9.6	Pre-Condition: program started Requirement: perform SUBTRACT Input: opcode 31, 10 in accumulator, 5 in data memory Output: 5 in accumulator				
TEST-9.7	Pre-Condition: program started Requirement: perform DIVIDE Input: opcode 32, 10 in accumulator, 5 in data memory Output: 2 in accumulator				
TEST-9.8	Pre-Condition: program started Requirement: perform MULTIPLY Input: opcode 33, 10 in accumulator, 5 in data memory Output: 50 in accumulator				

TEST-9.9	Pre-Condition: program started Requirement: perform MOD Input: opcode 34, 10 in accumulator, 5 in data memory Output: 0 in accumulator				
TEST-9.10	Pre-Condition: program started Requirement: perform EXP Input: opcode 35, 10 in accumulator, 5 in data memory Output: 9765625 in accumulator				
TEST-9.11	Pre-Condition: program started Requirement: perform BRANCH Input: opcode 40, 1 in data memory Output: instruction counter set to 1				
TEST-9.12	Pre-Condition: program started Requirement: perform BRANCHNEG Input: opcode 41, -5 in accumulator, 1 in data memory Output: instruction counter set to 1				
TEST-9.13	Pre-Condition: program started Requirement: perform BRANCHZERO Input: opcode 42, 0 in accumulator, 1 in data memory Output: instruction counter set to 1				
TEST-9.14	Pre-Condition: program started Requirement: perform HALT Input: opcode 43 Output: program stops				
TEST-9.15	Pre-Condition: program started Requirement: perform BRANCHPOS Input: opcode 44, 5 in accumulator, 1 in data memory Output: instruction counter set to 1				
TEST-10.0	Pre-Condition: program running, user has a keyboard Requirement: support user input through keyboard Input: keyboard input Output: display keyboard input				
TEST-11.0	Pre-Condition: invalid operation called Requirement: exit on invalid operation and report the invalid opcode to a file called invalidOpcode.log Input: invalid operation Output: exit and report invalid operation to file called invalidOpcode.log				

Part I - Test Matrix

	Valid SML File	**Alphanumeric Characters in Files Names**	**Special Characters in File Names**	**Invalid SML File (not SML file)**	**Floating Point Istructions in File**
Open SML Files?	X	X	X	X	N/A

Close SML Files?	X	X	X	N/A	N/A
Read SML Files?	X	N/A	N/A	N/A	N/A
Perform fetch/decode/execu instruction cycle?	N/A	N/A	N/A	N/A	N/A
Display Program & Data Memory?	N/A	N/A	N/A	N/A	N/A
Display Instruction Counter, Register, Accumulator, Operation Code, Operand?	N/A	N/A	N/A	N/A	N/A
Allow single step execution of instructions?	N/A	N/A	N/A	N/A	N/A
Load SML program into memory?	N/A	N/A	N/A	N/A	N/A
Support Integral Operations?	N/A	N/A	N/A	N/A	X
Support User Input Through Keyboard?	N/A	N/A	N/A	N/A	N/A
Exit On Ivalid Operation And Report To invalidOpcode.log?	N/A	N/A	N/A	N/A	N/A

From what we could tell, this Test Matrix seemed completely like a waste of time. It made the overall project less readable an there was more wasted space, by repeating the table we already have. Our suggestion for usefulness and readability would be to break the project into multiple small matricies - aka, each test ID set (1.xx, 9.xx, and so on) would each have it's own small matrix.

Part II - Black Box Testing

BLACK BOX TEST ID	**TEST**
BBTEST-1.0	Pre-Condition: have an SML file with no instructions or intial values for variables used Requirement: does not open file Input: invalid SML file Output: error message, indicating stupidity

BBTEST-2.0	Pre-Condition: program started Requirement: must allow for single step execution of instructions Input: 10 commands Output: processes each opcode after user indicates				
BBTEST-3.0	Pre-Condition: program started Requirement: perform NO_OP Input: opcode 0 Output: <none>				
BBTEST-4.0	Pre-Condition: program started Requirement: perform BRANCHPOS Input: opcode 44, 5 in accumulator, 1 in data memory Output: instruction counter set to 1				
BBTEST-5.0	Pre-Condition: invalid operation called Requirement: exit on invalid operation and report the invalid opcode to a file called ivalidOpcode.log Input: invalid operation Output: exit and report invalid operation to file called invalidOpcode.log				
BBTEST-6.0	Pre-Condition: program can handle 100mb memory Requirement: program can handle 100mb and opcodes Input: +100001001 Output: stores user input in registry 1001				
BBTEST-7.0	Pre-Condition: program started Requirement: program can bit shift left Input: 1 << 2 Output: 4				
BBTEST-7.1	Pre-Condition: program started Requirement: program can bit shift right Input: 4 >> 2 Output: 1				

Part II - Regression Testing

REGRESSION TEST ID	**TEST**				
TEST-1.0	Pre-Condition: have an SML file Requirement: must be able to open SML files Input: vaild SML file Output: open SML file				
TEST-1.1	Pre-Condition: alphanumeric characters in file name Requirement: must be able to open SML files Input: vaild SML file Output: open SML file				

TEST-1.2	Pre-Condition: special characters in file name Requirement: must be able to open SML files Input: SML file with all special characters: !@#$%^&()_+=[]{};',.`~.sml Output: open SML file				
TEST-1.3	Pre-Condition: have an SML file with no instructions or intial values for variables used Requirement: does not open file Input: invalid SML file Output: error message, indicating stupidity				
TEST-2.0	Pre-Condition: have an open/working SML file Requirement: must be able to close SML file Input: open sml file Output: closed sml file (aka nothing more showing), but maybe a message declaring success				
TEST-3.0	Pre-Condition: have working SML file Requirement: must be able to read SML files Input: valid SML file Output: SML file content				
TEST-4.0	Pre-Condition: open SML file Requirement: must perform the fetch/decode/execute instruction cycle on SML instructions Input: "prog1.sml" Output: N/A				
TEST-5.0	Pre-Condition: program started Requirement: must display program and data memory Input: N/A Output: pixels				
TEST-6.0	Pre-Condition: program started Requirement: must display the instruction counter, instruction register, accumulator, operation code, and operand Input: N/A Output: instruction counter, instruction register, accumulator, operation code, and operand				
TEST-7.0	Pre-Condition: program started Requirement: must allow for single step execution of instructions Input: 10 commands Output: processes each opcode after user indicates				
TEST-8.0	Pre-Condition: computer has memory Requirement: must load a SML program into memory Input: Simpletron Output: Simpletron proram is loaded into computer memory				
TEST-9.0	Pre-Condition: program started Requirement: perform NO_OP Input: opcode 0 Output: <none>				

TEST-9.1	Pre-Condition: program started Requirement: perform READ Input: opcode 10, user input 10 Output: 10 stored in memory				
TEST-9.2	Pre-Condition: program started Requirement: perform WRITE Input: opcode 11 Output: displays the value stored in memory				
TEST-9.3	Pre-Condition: Program started Requirement: Perform LOAD Input: opcode 20, 10 in data memory Output: 10 in accumulator				
TEST-9.4	Pre-Condition: program started Requirement: perform STORE Input: opcode 21, 10 in accumulator Output: 10 in data memory				
TEST-9.5	Pre-Condition: program started Requirement: perform ADD Input: opcode 30, 10 in accumulator, 5 in data memory Output: 15 in accumulator				
TEST-9.6	Pre-Condition: program started Requirement: perform SUBTRACT Input: opcode 31, 10 in accumulator, 5 in data memory Output: 5 in accumulator				
TEST-9.7	Pre-Condition: program started Requirement: perform DIVIDE Input: opcode 32, 10 in accumulator, 5 in data memory Output: 2 in accumulator				
TEST-9.8	Pre-Condition: program started Requirement: perform MULTIPLY Input: opcode 33, 10 in accumulator, 5 in data memory Output: 50 in accumulator				
TEST-9.9	Pre-Condition: program started Requirement: perform MOD Input: opcode 34, 10 in accumulator, 5 in data memory Output: 0 in accumulator				
TEST-9.10	Pre-Condition: program started Requirement: perform EXP Input: opcode 35, 10 in accumulator, 5 in data memory Output: 9765625 in accumulator				
TEST-9.11	Pre-Condition: program started Requirement: perform BRANCH Input: opcode 40, 1 in data memory Output: instruction counter set to 1				

TEST-9.12	Pre-Condition: program started Requirement: perform BRANCHNEG Input: opcode 41, -5 in accumulator, 1 in data memory Output: instruction counter set to 1				
TEST-9.13	Pre-Condition: program started Requirement: perform BRANCHZERO Input: opcode 42, 0 in accumulator, 1 in data memory Output: instruction counter set to 1				
TEST-9.14	Pre-Condition: program started Requirement: perform HALT Input: opcode 43 Output: program stops				
TEST-9.15	Pre-Condition: program started Requirement: perform BRANCHPOS Input: opcode 44, 5 in accumulator, 1 in data memory Output: instruction counter set to 1				
TEST-10.0	Pre-Condition: program running, user has a keyboard Requirement: support user input through keyboard Input: keyboard input Output: display keyboard input				
TEST-11.0	Pre-Condition: invalid operation called Requirement: exit on invalid operation and report the invalid opcode to a file called ivalidOpcode.log Input: invalid operation Output: exit and report invalid operation to file called invalidOpcode.log				