purpose of the class. purpose of each function, description of any input parameters for the function, description of return value (if any), pre- and postconditions for the function

PROCESS_PROGRAM.PY

EXECUTEPROGRAM.PY GUI_LAYOUT.PY GUI_ACTIONS.PY MAIN.PY **MEMORY.PY** UVSIM.PY

GUILayout class Function:This class creates the GUI using tkinterMethods: limit_code_lines(): purpose: limit the number of lines

in the code block input: event

GUILAYOUT CLASS

-main: GUI window -label_1: object -label_2: object -label_3: object -file_menu: object -menu_bar: object -operations_text: object -code_text: object -_program: list -widgets: object

constructor()

limit_code_lines()

GUI class Function: creates a simple GUI using tkinter Input: UVsim class Methods: open file(): purpose: search directories and choose a txt file. input: n/a save_code_block(): purpose: save the text inputed into the textblock

input: n/a run_code_block(): input: n/a

program

read(): the keyboard into memory. write(): purpose: write a word from

memory to gui input: n/a

GUIACTIONS

purpose: runs program with the selected file as the input configure color scheme():

change GUI's colors input: n/a output():

input: output purpose: Read a word from input: value from keyboard

CLASS

purpose:allows user to purpose: display results of the

memory: object constructor() limit_code_lines() **EXECUTE CLASS** open_file() execute_program() save_code_block() execute_program class Function: to execute the program Input: UVSim class run_code_block() Methods: execute(): configure_color_scheme () input: GUI output()

sim: object

purpose: To actually execute, To call the other classes as needed.

read() write() my_Sim = UVSim() Function: to run the my_memory = Memory(100) Input: Program: n/a my_gui = Methods: n/a GUIActions(my_sim, my_memory) my_gui.main.mainlo op() **MEMORY** CLASS

Memory class

Function: handle memory-

MAIN

related operations Input: n/a Methods: constructor() load(): purpose: load a word from memory into the accumulator load() input: operand store(): purpose: store a word from the accuulator into memory store() input: operand, accumulator load_program(): purpose: load a program into memory load_program() input: program truncate(): purpose: truncate value to four truncate()

self._registers: int

len()

clear():

digits to avoid overflow when input: value len(): purpose: return length of registers list input: n/a clear():

purpose: clear memory input: n/a

UVSim class Function: UVSim is a simulation that can interpret a machine anguage called BasicML. add(): purpose: to add two numbers input: two values to add together

-_operand: int subtract(): purpose: to subtract two numbers Constructor() input: two values to subtract

divide(): purpose: to divide a number from add() input: a value, and a value to divide that number by multiply():

UVSIM CLASS

-_accumulator: int

-_pc: int

-_op: int

subtract() purpose: to multiply two numbers input: two values to multiply branch(): divide() purpose: to move to a specific location in memory

input: a value multiply() branchNeg(): purpose: to move to a specific location if the number is negative. branch() input: accumulator value. operand value, and pc value hranchZero(): purpose: to move to a specific branch_neg() location if the number is negative

branch_zero() purpose: to end the program mmediately input: n/a halt() get accumulator(): purpose: gets accumulator input: n/a get_accumulator() set_accumulator()

input:

purpose: gets operand

purpose: sets accumulator set_accumulator() set_operand(): purpose: sets operand input: n/a set operand() get_operand():

input: n/a get_operand() set memory(): purpose: sets memory input: n/a set_memory() set_program(): purpose: sets program input: n/a

set_program()

get pc()

get_program(): purpose: gets program input: n/a get_program() get_pc(): purpose: gets pc input: n/a

read_txt()

PROCESS

Function: This class reads a program from a data source. It also loads the program into memory Input: n/a Methods: read txt(): purpose: read a program from a file and return it as a list of instructions and data values input: file_path

Process class