# Pseudocode for Basic Computer

* Import the utilities package
* Import the IO package
* Create all the variables using static so that they can be used in every method
* Initialize the variable accumulator by zero.
* Initialize the variable instruction register by zero.
* Initialize the operation code by zero.
* Initialize the location in the memory by zero.
* Initialize the program counter by zero.
* Create a Boolean variable to check for errors in the program and set it equal to false.
* Create an array named ram for the memory
* String variable filename to hold address of file
* Main method.
  + Loop to initialize ram
  + Output Title with environment spacing
  + Call the LoadRam function to get the data from the files.
  + Running a loop while the operation code is not “halt” and the Boolean error is false.
  + In loop perform fetch, decode, execute and dump methods.
* Call the fetch method
  + Pass the counter as an argument to the method into the parameter variable location.
  + Check for errors, if the value of location is more than 100 or less than zero then the Boolean error will be true. It will display an error code and halt the program.
  + If there are no errors, then increment the program counter and return the value and store it in the Instruction Register.
* Call the dump method.
  + Print the updated registers.
* Calls the decode method.
  + Check for errors, if the value of the instruction register is more than 5000, the Boolean error is set to true. It will display an error message and halt the program.
  + If there are no errors, then assign the values for the operation code and operand.
  + The value for operation code will be the instruction register divided by 100.
  + The value for operand will be the remainder when the instruction register is divided by 100.
* Call the dump method
  + Print the updated registers.
* Call the execute method, where all the operations are done.
  + Create switch statements and put operation code as an argument.
  + Each will do specific operation.
  + Case 34 will be for performing the operation Write Value.
    - Write a series of words from memory starting at operand memory location with length specified in Accumulator
  + Case 35 will be for performing the operation Write ASCII.
    - Write series of words (In Ascii Characters) from memory starting at operand memory.
  + Case 33 will be for performing the operation Read.
    - If a number greater than 9999 or smaller than -9999 is enter, display error (Data Overflow Error) message and halt the program.
  + Case 32 will be for performing the operation Write.
  + Case 31 will be for performing the operation Load.
  + Case 30 will be for performing the operation Store.
  + Case 21 will be for performing the operation Add.
    - If accumulator goes over 9999, display error message (Accumulator Overflow) and halt the program.
  + Case 20 will be for performing the operation Subtract.
    - If accumulator goes under -9999, display error message (Accumulator Underflow) and halt the program.
  + Case 11 will be for performing the operation Division.
    - If the divisor is zero, then Boolean error is set to true.
    - If accumulator goes under -9999, display error message (Accumulator Underflow) and halt the program.
  + Case 29 will be for performing the operation LoadImm.
  + Case 06 will be for performing the operation AddImm.
    - If accumulator goes over 9999, display error message (Accumulator Overflow) and halt the program.
  + Case 07 will be for performing the operation DecImm
    - If accumulator goes under -9999, display error message (Accumulator Underflow) and halt the program.
  + Case 08 will be for performing the operation MultImm
    - If accumulator goes over 9999, display error message (Accumulator Overflow) and halt the program.
  + Case 09 will be for performing the operation DivideImm
    - If accumulator goes under -9999, display error message (Accumulator Underflow) and halt the program.
  + Case 10 will be for performing the operation Multiply.
    - If accumulator goes over 9999, display error message (Accumulator Underflow) and halt the program.
  + Case 43 will be for performing the operation Branch.
    - Set counter equal to operand
    - If counter is less than 0 or more than 99, display error message (Branch Memory Error) and halt the program.
  + Case 42 will be for performing the operation BranchNeg.
    - If accumulator is less than 0, set sounter equalt to operand
    - If counter is less than 0 or more than 99, display error message (Branch Memory Error) and halt the program.
  + Case 41 will be for performing the operation BranchPos.
    - If accumulator is greater than 0, set counter equal to operand.
    - If counter is less than 0 or more than 99, display error message (Branch Memory Error) and halt the program.
  + Case 40 will be for performing the operation BranchZero.
    - If accumulator is equal to zero, set counter equal to operand.
    - If counter is less than 0 or more than 99, display error message (Branch Memory Error) and halt the program.
  + Case 25 will be for performing the operation Increment.
  + Case 26 will be for performing the operation Decrement.
  + Case 50 will be for performing the operation halt.
  + Default: Set Boolean error to true if the operation code is not equal to number which is each case.
    - Display error message Instruction opcode is not supported.
* LoadRam method
  + Scanner Object for user input.
  + Get name of the file
  + Create the file object.
  + Create a scanner object and put the file as an argument.
  + Initialize counter variable to zero.
  + While loop to get the contents from the file.
    - Use hasNext method to copy contents of file into the array
* Create a method dump.
  + Print registers
* Create a method memory dump.
  + Run loop to output the data in the memory.