FinalProject

Generated by Doxygen 1.10.0

Chapter 1

Namespace Index

1.1 Package List

Here are the packages with brief descriptions (if available):

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

Namespace Documentation

4.1 uvsim.opcodes Namespace Reference

Variables

- int **READ** = 10 ∗ WORD_SIZE
- int **WRITE** = 11 ∗ WORD_SIZE
- int **LOAD** = 20 ∗ WORD_SIZE
- int **STORE** = 21 ∗ WORD_SIZE
- int **ADD** = 30 ∗ WORD_SIZE
- int **SUBTRACT** = 31 ∗ WORD_SIZE
- int **DIVIDE** = 32 ∗ WORD_SIZE
- int **MULTIPLY** = 33 ∗ WORD_SIZE
- int **BRANCH** = 40 ∗ WORD_SIZE
- int **BRANCHNEG** = 41 ∗ WORD_SIZE
- int **BRANCHZERO** = 42 ∗ WORD_SIZE
- int **HALT** = 43 ∗ WORD_SIZE
- list **OPCODES**

4.1.1 Detailed Description

```
In this file, there are constants defined for all our opcodes. This way, we can
more easily write programs, like this:
```
```
program = [
    WRITE + 1, # Because WRITE = 11 * 100, this puts 1101 in $0
    123, # This puts 123 in $1
    HALT
]
CPU(program)
CPU.run_until_halt()
```
```

When this program is run, it will print 123, then halt.
```

4.1.2 Variable Documentation

4.1.2.1 OPCODES

list uvsim.opcodes.OPCODES

Initial value:

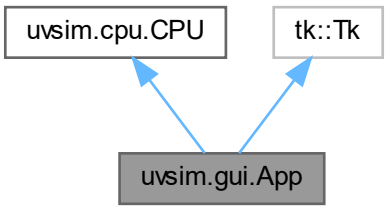
```
00001 =  [
00002     READ,
00003     WRITE,
00004     LOAD,
00005     STORE,
00006     ADD,
00007     SUBTRACT,
00008     DIVIDE,
00009     MULTIPLY,
00010     BRANCH,
00011     BRANCHNEG,
00012     BRANCHZERO,
00013     HALT
00014 ]
```

Chapter 5

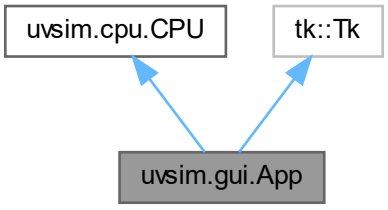
Class Documentation

5.1 uvsim.gui.App Class Reference

Inheritance diagram for uvsim.gui.App:



Collaboration diagram for uvsim.gui.App:



Public Member Functions

- None **__init__** (self, list[int] memory, str|None screenName=None, str|None baseName=None, str class↩ Name="Tk", bool useTk=True, bool sync=False, str|None use=None)
- **change_color** (self)
- **open** (self)
- **save** (self)
- **save_as** (self)
- **step** (self)
- **run_to_address** (self)
- **run_until_halt** (self)
- **read** (self, data, user_input=False)
- **write** (self, data)
- **accumulator** (self)
- **accumulator** (self, value)
- **program_counter** (self)
- **program_counter** (self, value)
- **halted** (self)
- **halted** (self, value)

Public Member Functions inherited from uvsim.cpu.CPU

- **run_one_instruction** (self)
- **load** (self, data)
- **store** (self, data)
- **add** (self, data)
- **subtract** (self, data)
- **divide** (self, data)
- **multiply** (self, data)
- **branch** (self, data)
- **branchneg** (self, data)
- **branchzero** (self, data)
- **halt** (self, data)
- **reset** (self)
- **resume** (self)

Public Attributes

- **open_file_path**
- **menu_bar**
- **file_menu**
- **edit_menu**
- **help_menu**
- **label**
- **master_frame**
- **left_menu_frame**
- **accumulator_entry**
- **program_counter_entry**
- **address_run_to_entry**
- **left_side_elems**
- **memory**
- **main_editor**
- **editors**
- **halted**

Public Attributes inherited from uvsim.cpu.CPU

- **memory**

Protected Attributes

- **_halted**
- **_program_counter**
- **_accumulator**
- **_address_run_to**
- **_file_path**

Additional Inherited Members

Static Public Attributes inherited from uvsim.cpu.CPU

- **OK** = OK
- **ERROR_ILLEGAL_INSTRUCTION** = ERROR_ILLEGAL_INSTRUCTION
- **ERROR_INVALID_INPUT** = ERROR_INVALID_INPUT
- **ERROR_DIVIDE_BY_ZERO** = ERROR_DIVIDE_BY_ZERO

5.1.1 Detailed Description

```
The App class represents the main application that integrates the
UVSim simulator with a graphical user interface (GUI).
It extends both the CPU class and the tk.Tk class to manage the
CPU state and the GUI.
```

5.1.2 Constructor & Destructor Documentation

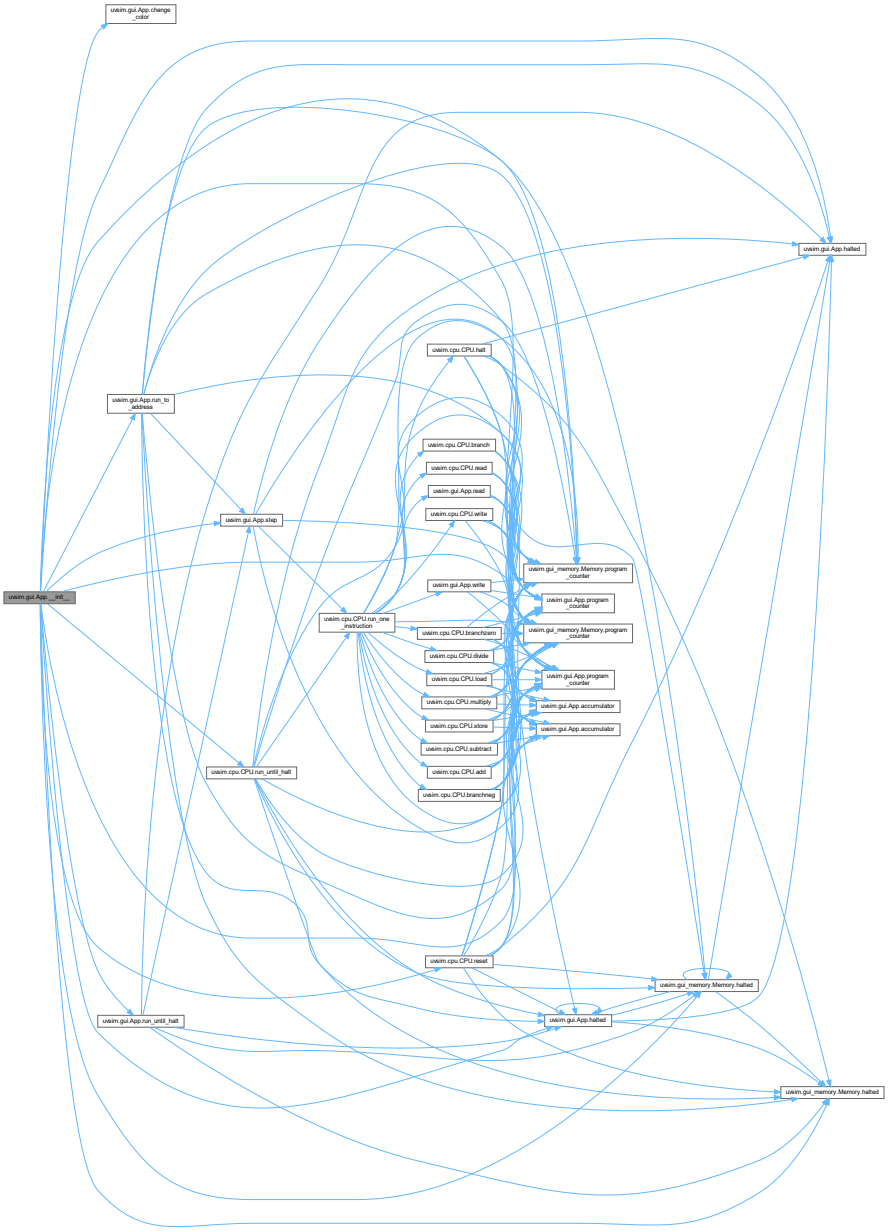
5.1.2.1 __init__()

```
None uvsim.gui.App.__init__ (
              self,
             list[int] memory,
             str | None  screenName = None,
             str | None  baseName = None,
             str  className = "Tk",
             bool  useTk = True,
             bool  sync = False,
             str | None  use = None )
```

```
Purpose:
    Initializes the UVSim application with the specified memory and sets up the GUI.
Input Parameters:
    memory: An array representing the memory of the CPU.
    screenName, baseName, className, useTk, sync, use: Parameters passed to the tk.Tk constructor.
Return Value:
    None.
```

Reimplemented from **uvsim.cpu.CPU** (p. 25).

Here is the call graph for this function:



5.1.3 Member Function Documentation

5.1.3.1 accumulator() [1/2]

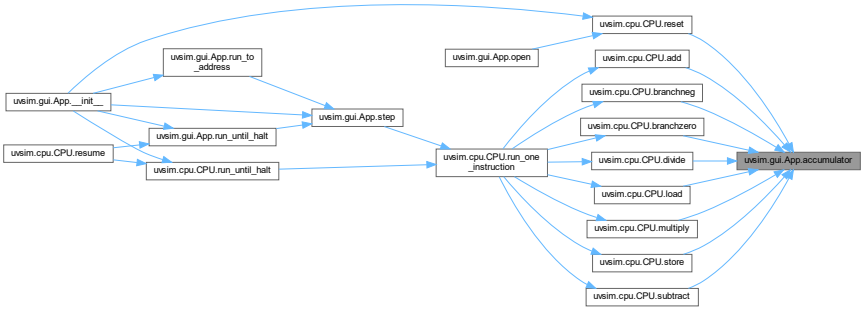
```
uvsim.gui.App.accumulator (
             self )
```

```
Purpose:
    Getter and setter for the accumulator property.
Input Parameters:
```

```
    None.
Return Value:
    The current value of the accumulator.
Pre-conditions:
    None.
Post-conditions:
    None.
```

Reimplemented from **uvsim.cpu.CPU** (p. 25).

Here is the caller graph for this function:

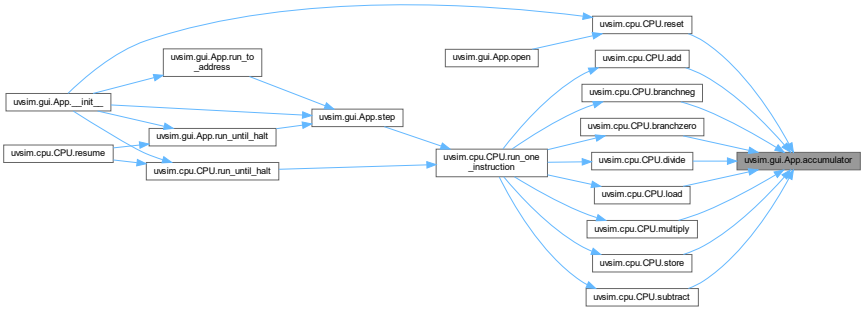


5.1.3.2 accumulator() [2/2]

```
uvsim.gui.App.accumulator (
            self,
            value )

Purpose:
    Setter for the accumulator property.
Input Parameters:
    value: The new value of the accumulator.
Return Value:
    None.
Pre-conditions:
    None.
Post-conditions:
    The accumulator value is updated.
```

Reimplemented from **uvsim.cpu.CPU** (p. 25).

Here is the caller graph for this function:

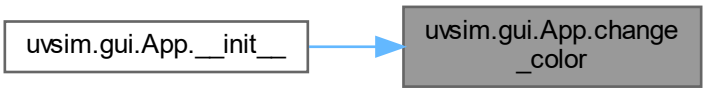


5.1.3.3 change_color()

```
uvsim.gui.App.change_color (
              self )
```

```
Purpose:
    Changes the color of the GUI.
Input Parameters:
    None.
Return Value:
    None.
Pre-conditions:
    Change color button is clicked from the menu.
Post-conditions:
    The color of the GUI is changed.
```

Here is the caller graph for this function:



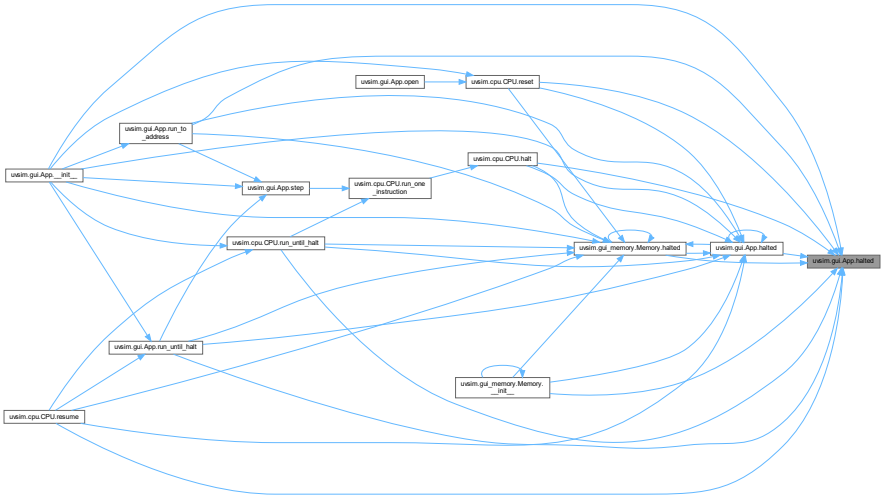
5.1.3.4 halted() [1/2]

```
uvsim.gui.App.halted (
              self )
```

```
Purpose:
    Getter and setter for the halted property.
Input Parameters:
    None.
Return Value:
    True if the CPU is halted, False otherwise.
Pre-conditions:
    None.
Post-conditions:
    None.
```

Reimplemented from **uvsim.cpu.CPU** (p. 25).

Here is the caller graph for this function:



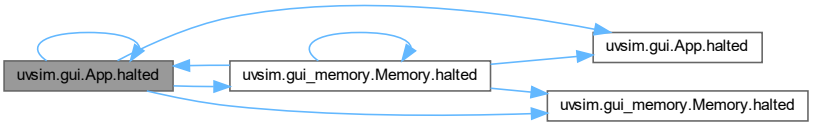
5.1.3.5 halted() [2/2]

```
uvsim.gui.App.halted (
              self,
              value )
```

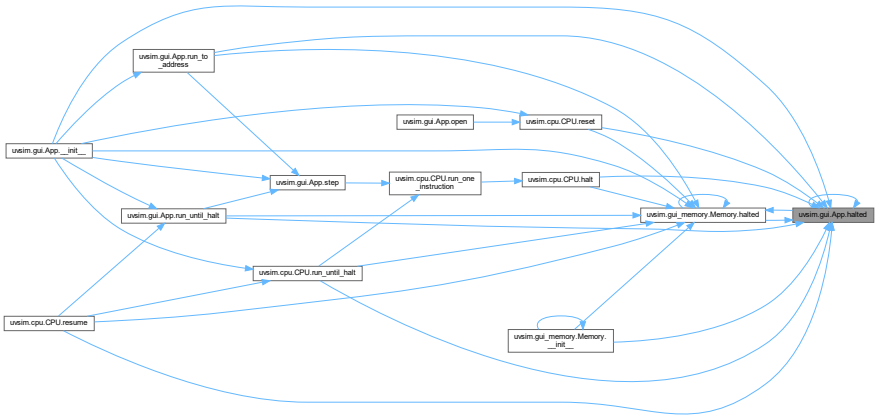
```
Purpose:
    Setter for the halted property.
Input Parameters:
    value: The new value of the halted property.
Return Value:
    None.
Pre-conditions:
    None.
Post-conditions:
    The halted property is updated.
```

Reimplemented from **uvsim.cpu.CPU** (p. 25).

Here is the call graph for this function:



Here is the caller graph for this function:



5.1.3.6 open()

```
uvsim.gui.App.open (
              self )
```

```
Purpose:
    Opens a file dialog to allow the user to select a file and loads its content into the CPU memory.
Input Parameters:
    None.
Return Value:
    None.
Pre-conditions:
    Needs to be selected from the menu.
Post-conditions:
    The memory content is set to the content of the selected file.
```

Here is the call graph for this function:

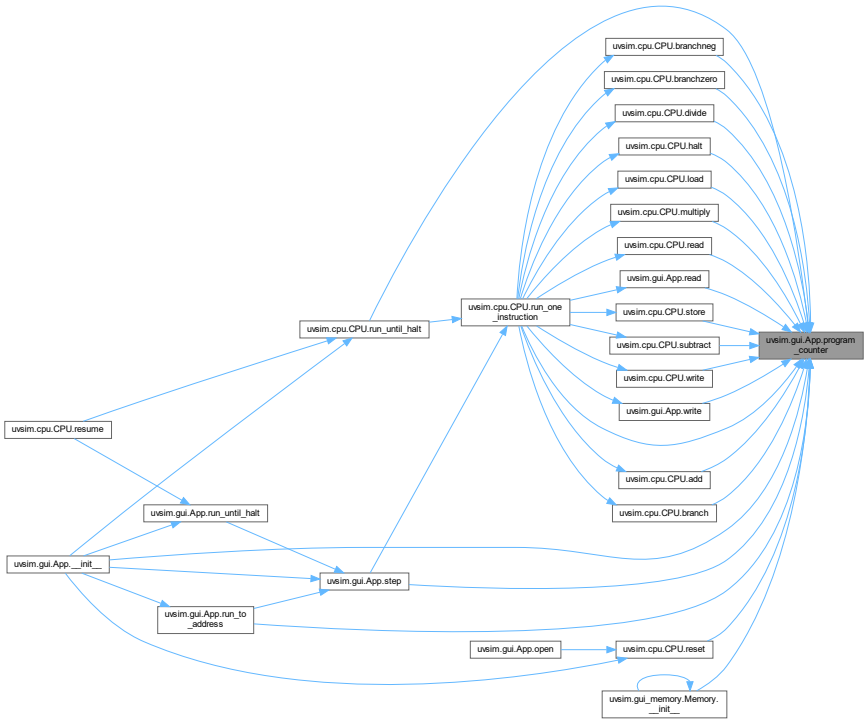
5.1.3.7 program_counter() [1/2]

```
uvsim.gui.App.program_counter (
              self )
```

```
Purpose:
    Getter and setter for the program counter property.
Input Parameters:
    None.
Return Value:
    The current value of the program counter.
Pre-conditions:
    None.
Post-conditions:
    None.
```

Reimplemented from **uvsim.cpu.CPU** (p. 25).

Here is the caller graph for this function:



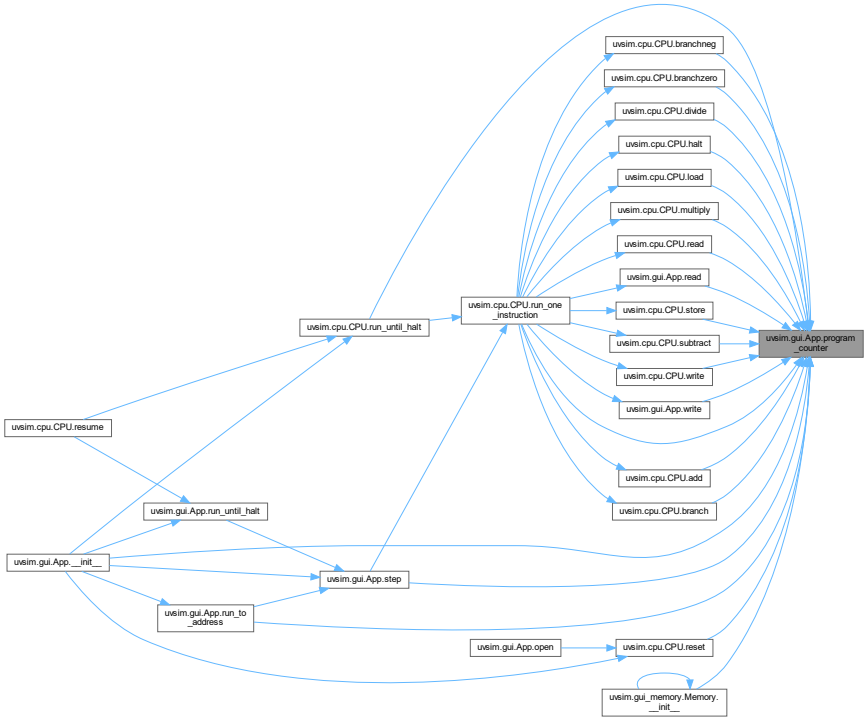
5.1.3.8 program_counter() [2/2]

```
uvsim.gui.App.program_counter (
              self,
              value )
```

```
Purpose:
    Setter for the program counter property.
Input Parameters:
    value: The new value of the program counter.
Return Value:
    None.
Pre-conditions:
    None.
```

Reimplemented from **uvsim.cpu.CPU** (p. 25).

Here is the caller graph for this function:



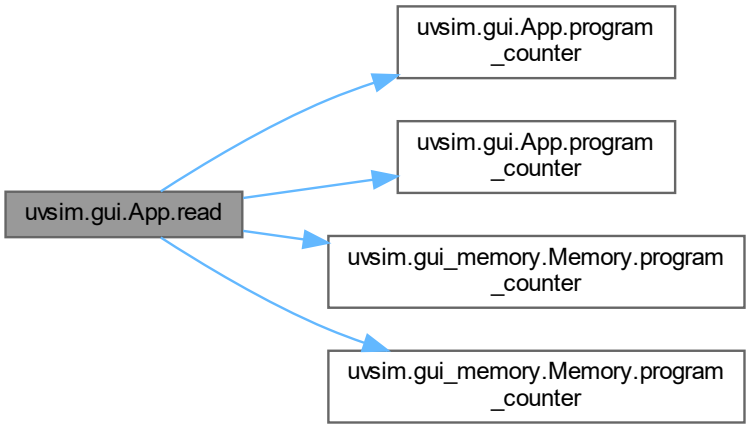
5.1.3.9 read()

```
uvsim.gui.App.read (
            self,
            data,
            user_input = False )
```

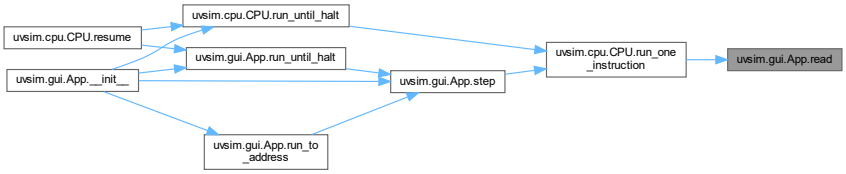
```
Purpose:
    Reads input from the user using a dialog box and stores it in memory.
Input Parameters:
    data: The memory location where the input will be stored.
    user_input: User-provided input. If not provided, input is obtained through a popup.
Return Value:
    An error code indicating the result of the operation.
Pre-conditions:
    None.
Post-conditions:
    Memory is updated with the user input.
```

Reimplemented from **uvsim.cpu.CPU** (p. 34).

Here is the call graph for this function:



Here is the caller graph for this function:

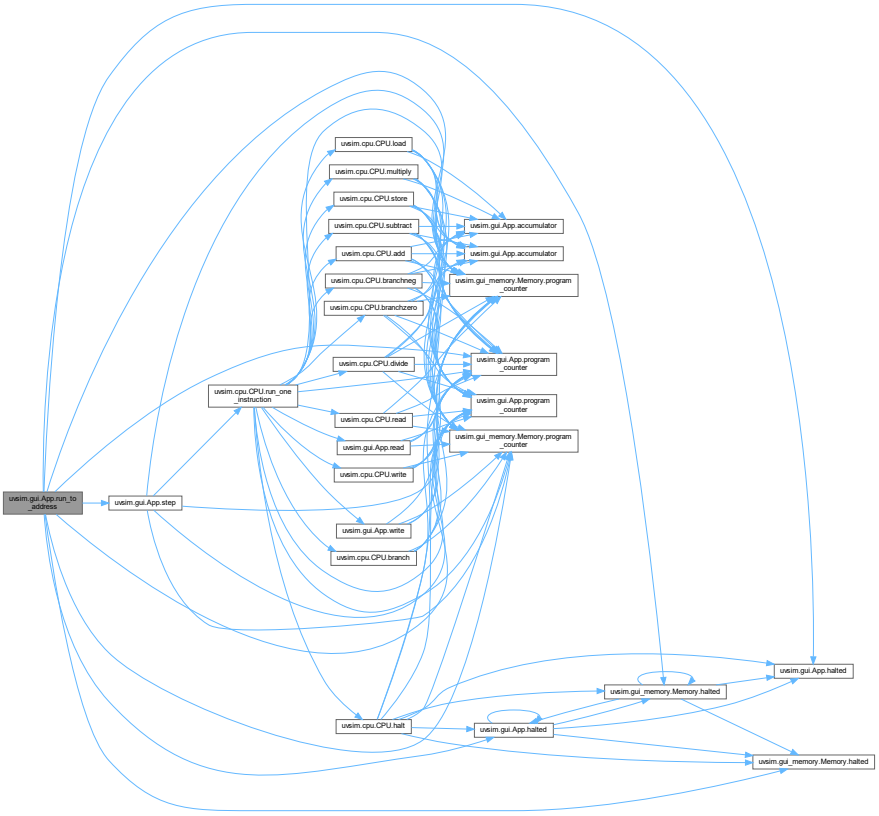


5.1.3.10 run_to_address()

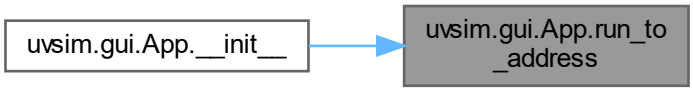
```
uvsim.gui.App.run_to_address (
              self )
```

```
Purpose:
    Executes instructions until the program counter reaches the specified address and displays an error messag
Input Parameters:
    None.
Return Value:
    None.
Pre-conditions:
    The CPU is not halted.
Post-conditions:
    The CPU executes instructions until the program counter reaches the specified address.
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.1.3.11 run_until_halt()

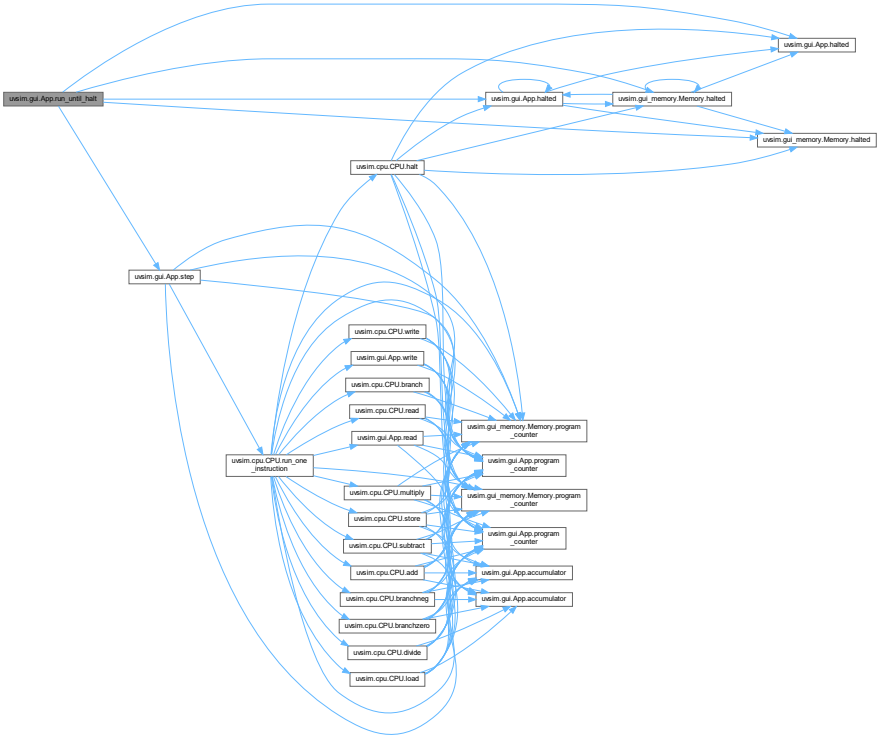
```
uvsim.gui.App.run_until_halt (
              self )
```

```
Purpose:
    Executes instructions until the CPU is halted and displays an error message if applicable.
Input Parameters:
    None.
Return Value:
    None.
```

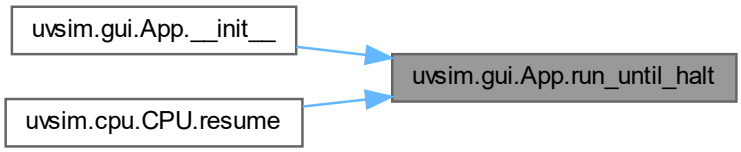
```
Pre-conditions:
    The CPU is not halted.
Post-conditions:
    The CPU executes instructions until it is halted.
```

Reimplemented from **uvsim.cpu.CPU** (p. 38).

Here is the call graph for this function:



Here is the caller graph for this function:

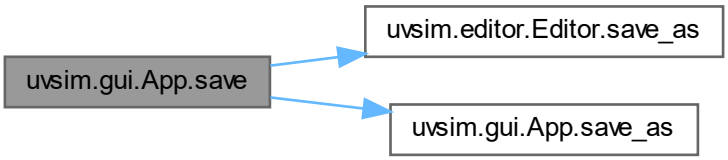


5.1.3.12 save()

```
uvsim.gui.App.save (
             self )
```

```
Purpose:
    Saves the current memory content to the previously opened or saved file path.
Input Parameters:
    None.
Return Value:
    None.
Pre-conditions:
    The memory content must be valid.
Post-conditions:
    The memory content is saved to the previously opened or saved file path.
```

Here is the call graph for this function:

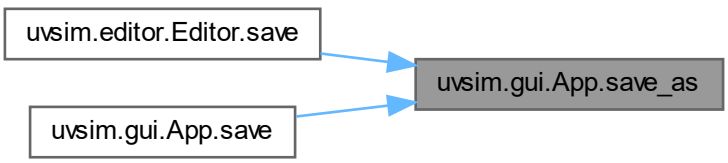


5.1.3.13 save_as()

```
uvsim.gui.App.save_as (
              self )
```

```
Purpose:
    Opens a file dialog to allow the user to save the current memory content to a new file.
Input Parameters:
    None.
Return Value:
    None.
Pre-conditions:
    The memory content must be valid.
Post-conditions:
    The memory content is saved to a new file.
```

Here is the caller graph for this function:

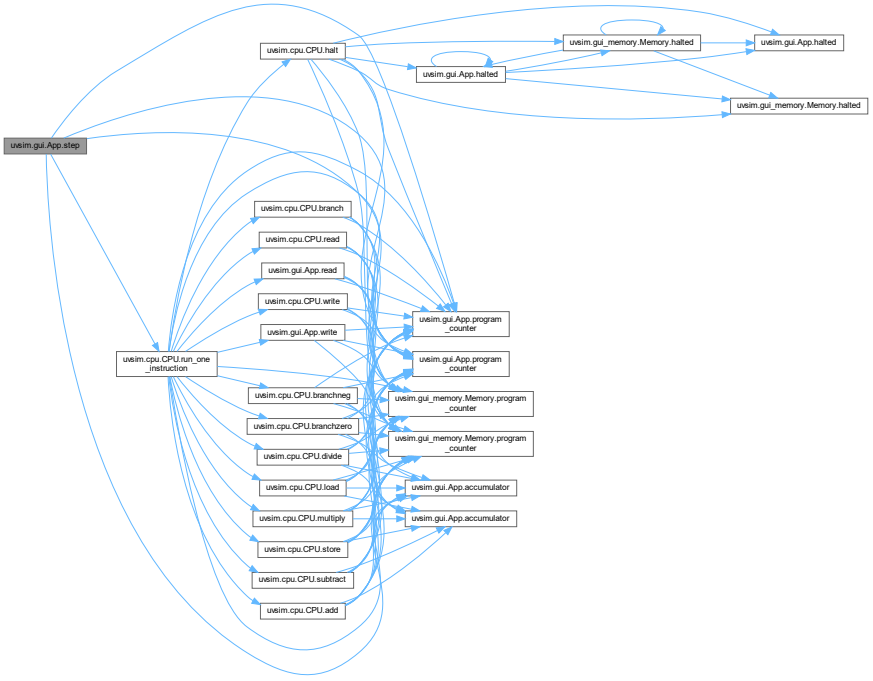


5.1.3.14 step()

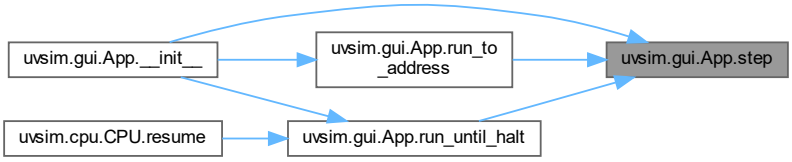
```
uvsim.gui.App.step (
              self )
```

```
Purpose:
    Executes a single instruction and displays an error message if applicable.
Input Parameters:
    None.
Return Value:
    The result of the instruction execution.
Pre-conditions:
    The CPU is not halted.
Post-conditions:
    The CPU executes a single instruction.
```

Here is the call graph for this function:



Here is the caller graph for this function:

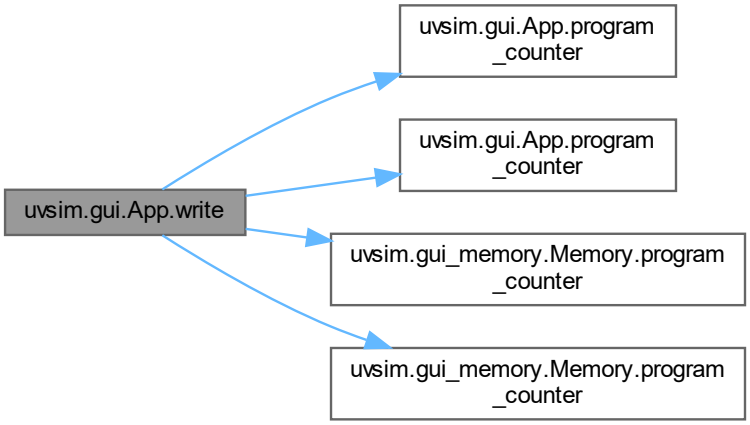


5.1.3.15 write()

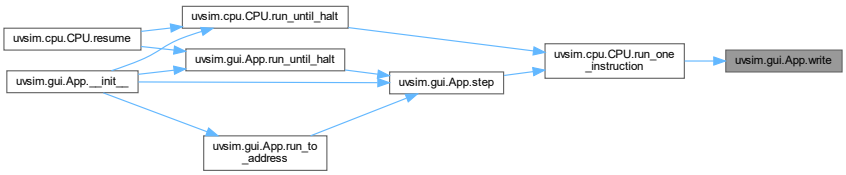
```
uvsim.gui.App.write (
              self,
              data )


Purpose:
    Writes a word from memory to a popup.
Input Parameters:
    data: The memory location of the word that will be written.
Return Value:
    An error code indicating the result of the operation.
Pre-conditions:
    None.
Post-conditions:
    The word is displayed in a popup.
```

Reimplemented from **uvsim.cpu.CPU** (p. 41).

Here is the call graph for this function:



Here is the caller graph for this function:

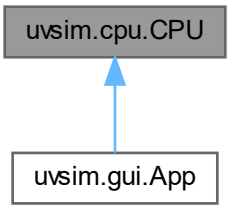


The documentation for this class was generated from the following file:

- C:/code_projects/school/CS2450/FinalProject/uvsim/gui.py

5.2 uvsim.cpu.CPU Class Reference

Inheritance diagram for uvsim.cpu.CPU:



Public Member Functions

- **__init__** (self, memory)
- **run_until_halt** (self)
- **run_one_instruction** (self)
- **read** (self, data, user_input=False)
- **write** (self, data)
- **load** (self, data)
- **store** (self, data)
- **add** (self, data)
- **subtract** (self, data)
- **divide** (self, data)
- **multiply** (self, data)
- **branch** (self, data)
- **branchneg** (self, data)
- **branchzero** (self, data)
- **halt** (self, data)
- **reset** (self)
- **resume** (self)

Public Attributes

- **accumulator**
- **program_counter**
- **memory**
- **halted**

Static Public Attributes

- **OK** = OK
- **ERROR_ILLEGAL_INSTRUCTION** = ERROR_ILLEGAL_INSTRUCTION
- **ERROR_INVALID_INPUT** = ERROR_INVALID_INPUT
- **ERROR_DIVIDE_BY_ZERO** = ERROR_DIVIDE_BY_ZERO

5.2.1 Detailed Description

```
CPU Class
Purpose of the Class
The CPU class represents a Central Processing Unit that does instructions stored in memory.
It interacts with memory and does various operations based on the opcode of the current instruction.

Class Attributes:
    self.accumulator : An integer that stored the data we are working with.
    self.program_counter : An Integer that counts the steps in the program
    self.memory : An array that represents the memory of the CPU
    self.halted : A boolean value that represents the state of the program
```

5.2.2 Constructor & Destructor Documentation

5.2.2.1 __init__()

```
uvsim.cpu.CPU.__init__ (
              self,
              memory )
```

Reimplemented in **uvsim.gui.App** (p. 11).

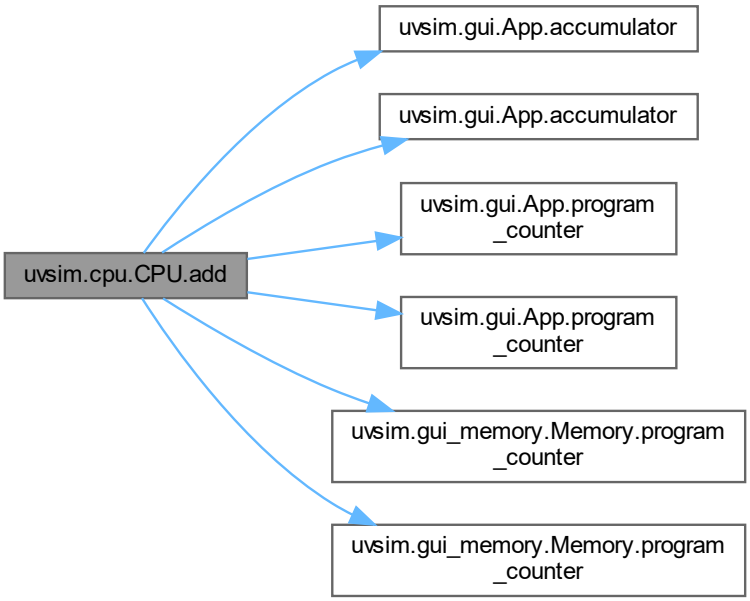
5.2.3 Member Function Documentation

5.2.3.1 add()

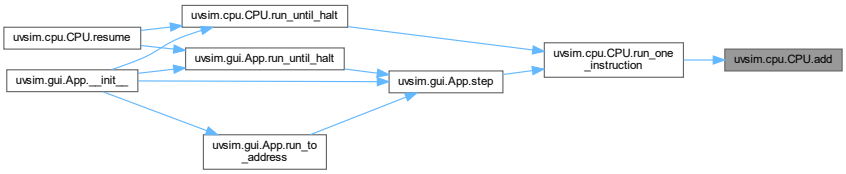
```
uvsim.cpu.CPU.add (
              self,
              data )
```

```
Purpose:
    Adds the value at a specified memory location to the accumulator.
Input Parameters:
    data: The memory location of the value that will be added.
Return Value:
    An error code showing the result of the operation.
Pre-conditions:
    The CPU needs to be initialized with a valid memory array.
Post-conditions:
    The accumulator is updated with the addition result.
```

Here is the call graph for this function:



Here is the caller graph for this function:

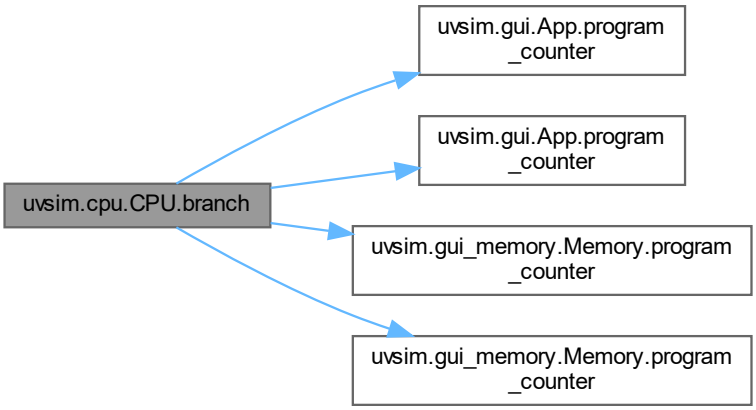


5.2.3.2 branch()

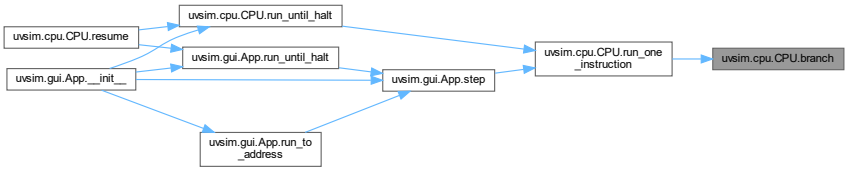
```
uvsim.cpu.CPU.branch (
            self,
            data )
```

```
Purpose:
    Branches to a specified memory location unconditionally.
Input Parameters:
    data: The memory location to branch to.
Return Value:
    An error code showing the result of the operation.
Pre-conditions:
    The CPU needs to be initialized with a valid memory array.
Post-conditions:
The program counter is updated with the branch address.
```

Here is the call graph for this function:



Here is the caller graph for this function:

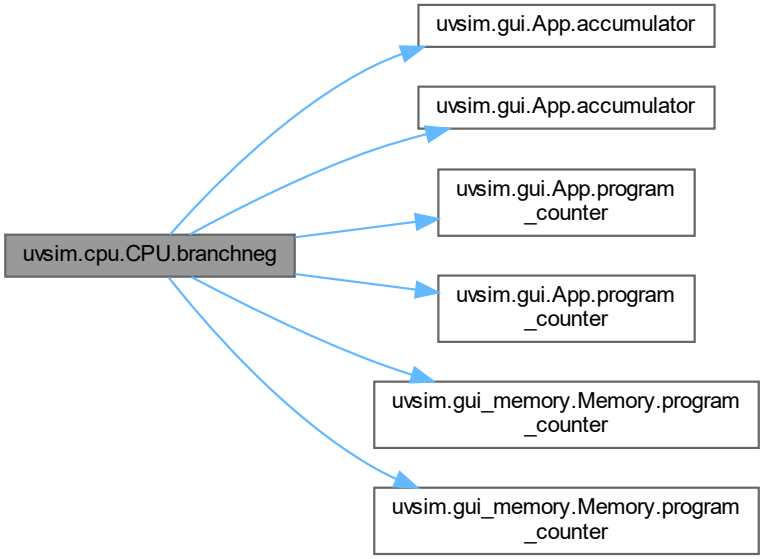


5.2.3.3 branchneg()

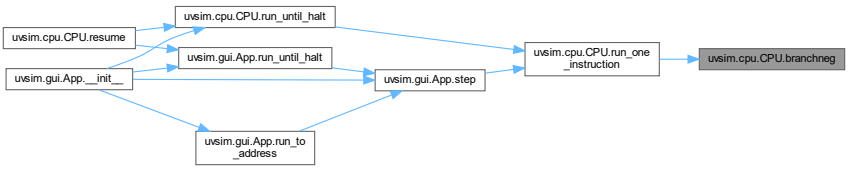
```
uvsim.cpu.CPU.branchneg (
              self,
              data )
```

```
Purpose:
    Branches to a specified memory location if the accumulator is negative.
Input Parameters:
    data: The memory location to branch to if the accumulator is negative.
Return Value:
    An error code showing the result of the operation.
Pre-conditions:
    The CPU needs to be initialized with a valid memory array.
Post-conditions:
    The program counter may be updated based on the accumulator value.
```

Here is the call graph for this function:



Here is the caller graph for this function:

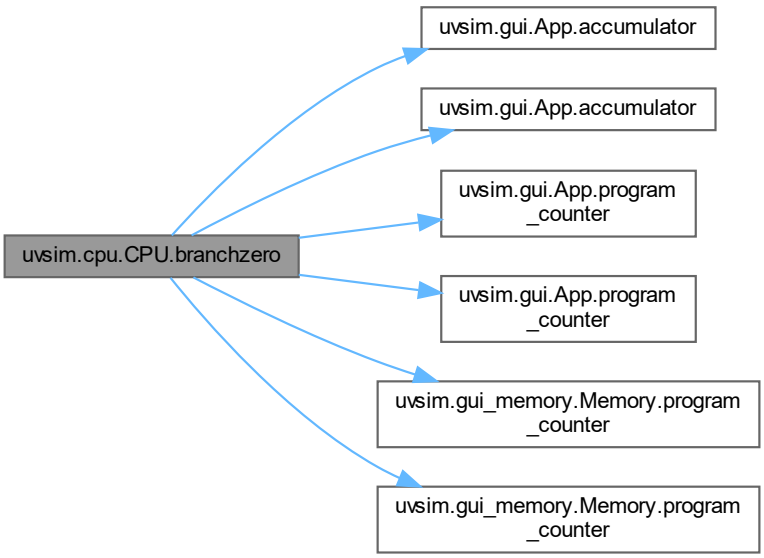


5.2.3.4 branchzero()

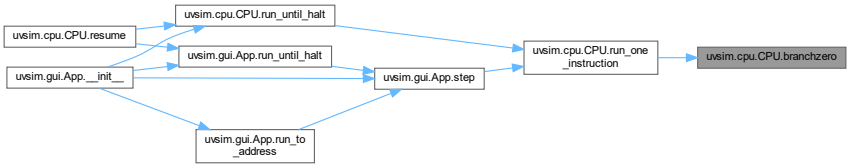
```
uvsim.cpu.CPU.branchzero (
            self,
            data )
```

```
Purpose:
    Branches to a specified memory location if the accumulator is zero.
Input Parameters:
    data: The memory location to branch to if the accumulator is zero.
Return Value:
    An error code showing the result of the operation.
Pre-conditions:
    The CPU needs to be initialized with a valid memory array.
Post-conditions:
    The program counter may be updated based on the accumulator value.
```

Here is the call graph for this function:



Here is the caller graph for this function:

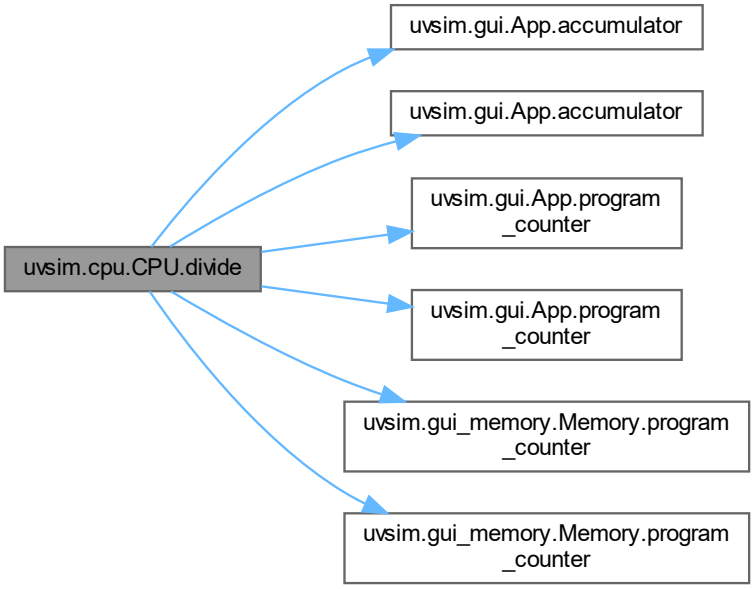


5.2.3.5 divide()

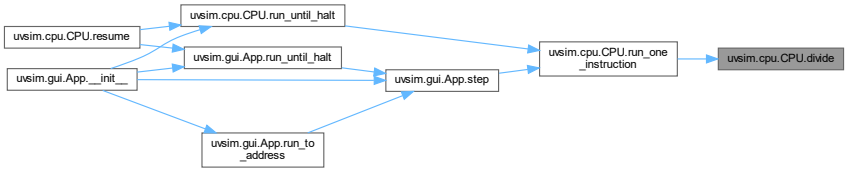
```
uvsim.cpu.CPU.divide (
              self,
              data )
```

```
Purpose:
    Divides the accumulator by the value at a specified memory location.
Input Parameters:
    data: The memory location of the number that will be divided.
Return Value:
    An error code showing the result of the operation.
Pre-conditions:
    The CPU needs to be initialized with a valid memory array.
    The divisor can't be zero.
Post-conditions:
    The accumulator is updated with the division result.
```

Here is the call graph for this function:



Here is the caller graph for this function:

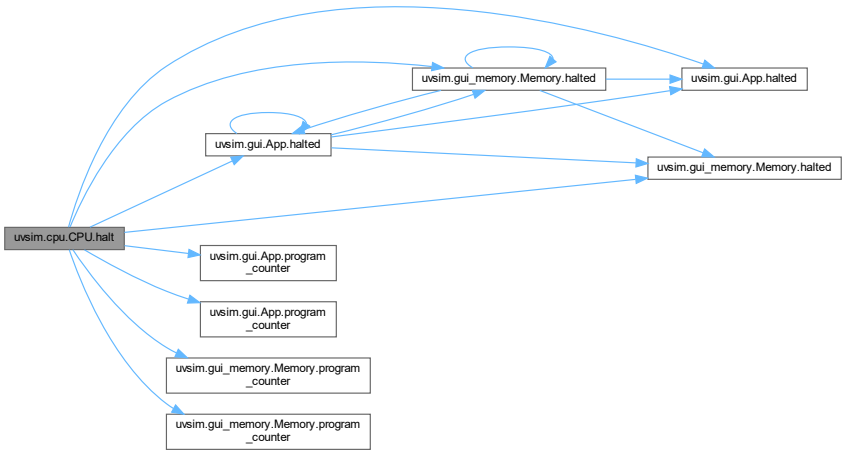


5.2.3.6 halt()

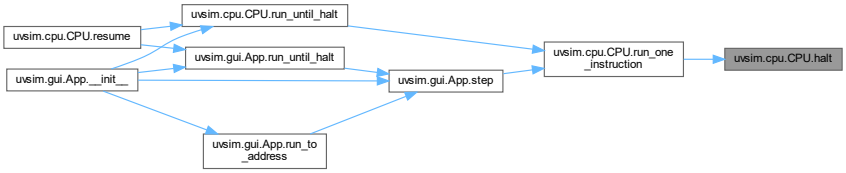
```
uvsim.cpu.CPU.halt (
            self,
            data )
```

```
Purpose:
    Halts the CPU.
Input Parameters:
    data:
Return Value:
    An error code showing the result of the operation.
Pre-conditions:
    The CPU needs to be initialized with a valid memory array.
Post-conditions:
    The CPU is halted.
```

Here is the call graph for this function:



Here is the caller graph for this function:

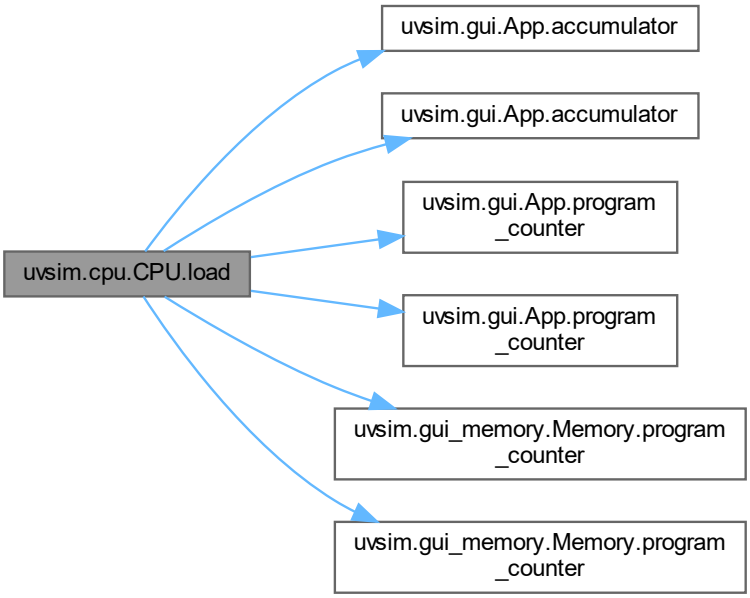


5.2.3.7 load()

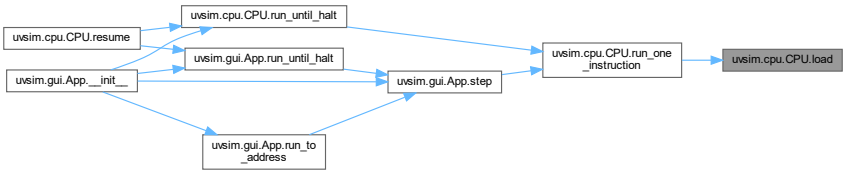
```
uvsim.cpu.CPU.load (
            self,
            data )
```

```
Purpose:
    Loads a word from memory into the accumulator.
Input Parameters:
    data: The memory location of the word that will be loaded.
Return Value:
    An error code showing the result of the operation.
Pre-conditions:
    The CPU needs to be initialized with a valid memory array.
Post-conditions:
    The accumulator is updated with the loaded word.
```

Here is the call graph for this function:



Here is the caller graph for this function:

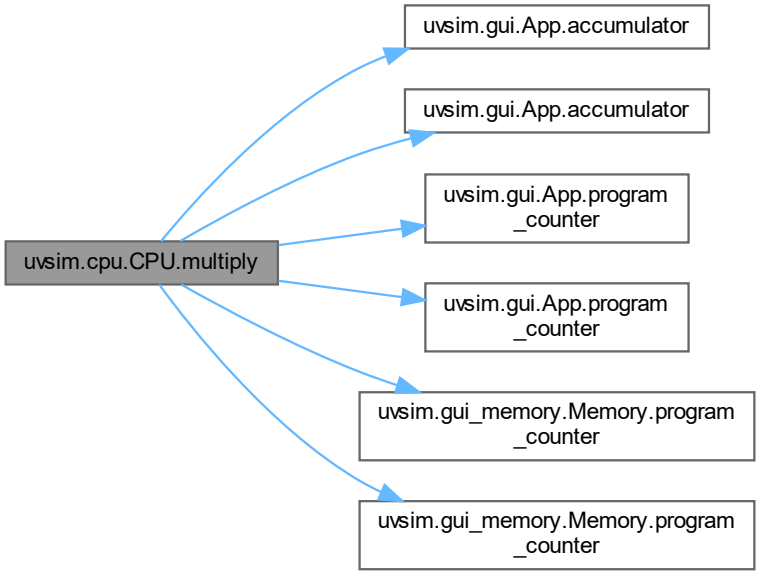


5.2.3.8 multiply()

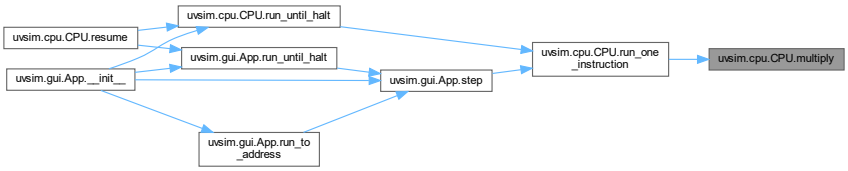
```
uvsim.cpu.CPU.multiply (
            self,
            data )
```

```
Purpose:
    Multiplies the accumulator by the value at a specified memory location.
Input Parameters:
    data: The memory location of the number that will multiply.
Return Value:
    An error code showing the result of the operation.
Pre-conditions:
    The CPU needs to be initialized with a valid memory array.
Post-conditions:
    The accumulator is updated with the multiplication result.
```

Here is the call graph for this function:



Here is the caller graph for this function:



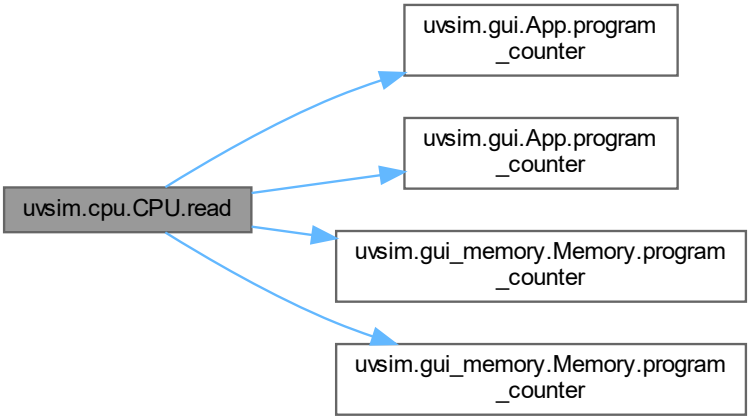
5.2.3.9 read()

```
uvsim.cpu.CPU.read (
              self,
              data,
              user_input = False )
```

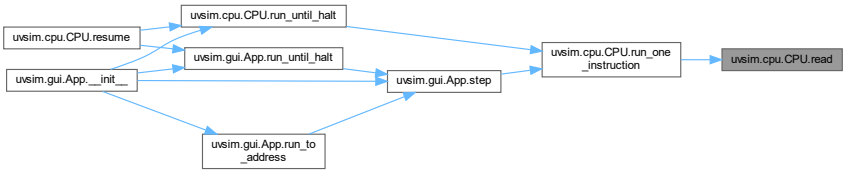
```
Purpose:
    Reads input from the user and stores it in memory.
Input Parameters:
    data: The memory location where the input will be stored.
    user_input: User-provided input.
Return Value:
    An error code showing the result of the operation.
Pre-conditions:
    The CPU needs to be initialized with a valid memory array.
Post-conditions:
    The memory location is updated with the user input.
```

Reimplemented in **uvsim.gui.App** (p. 18).

Here is the call graph for this function:



Here is the caller graph for this function:

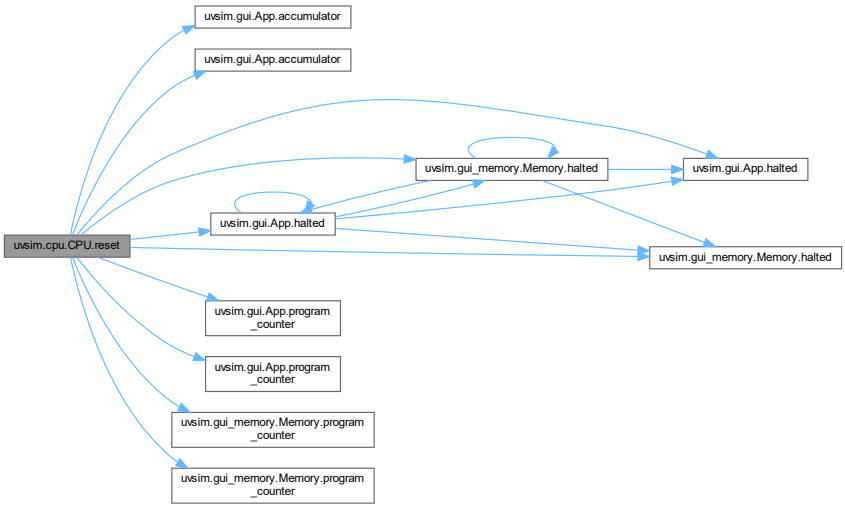


5.2.3.10 reset()

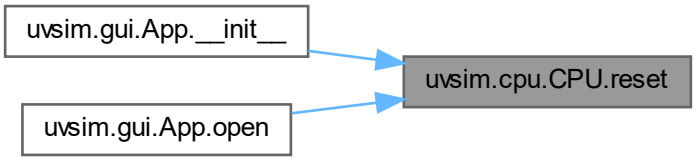
```
uvsim.cpu.CPU.reset (
              self )
```

```
Purpose:
    Resets the CPU to its initial state.
Input Parameters:
    None.
Return Value:
    None.
Pre-conditions:
    None.
Post-conditions:
    The CPU is reset to its initial state.
```

Here is the call graph for this function:



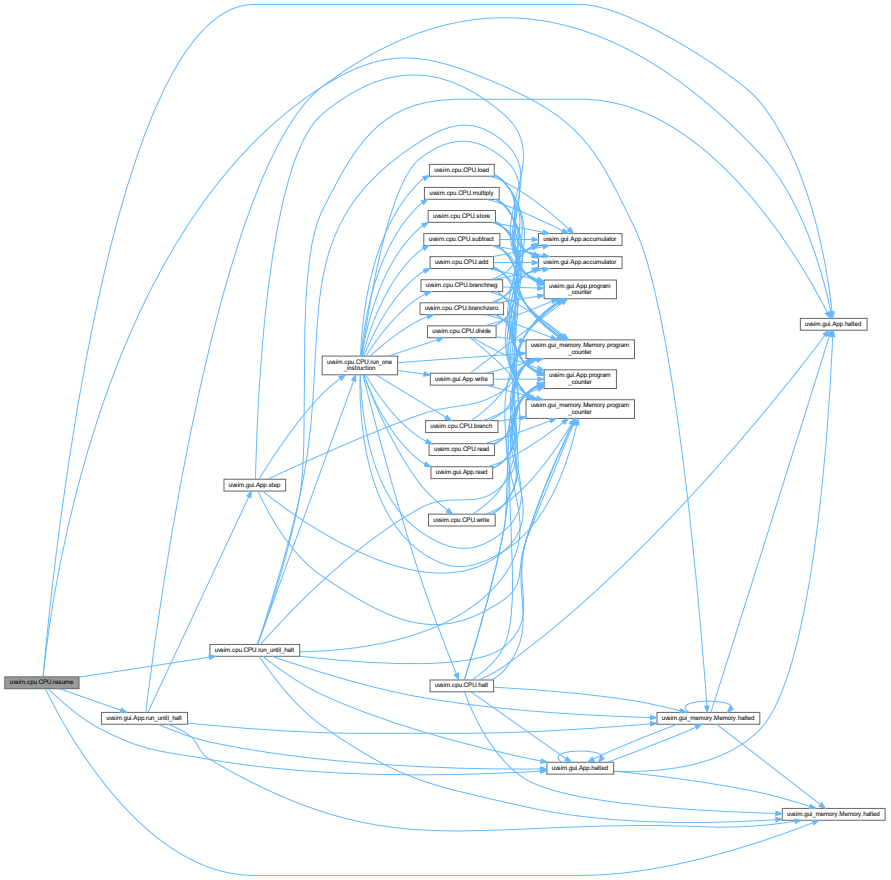
Here is the caller graph for this function:



5.2.3.11 resume()

```
uvsim.cpu.CPU.resume (
              self )


Purpose:
    Resumes execution of the CPU until it is halted.
Input Parameters:
    None.
Return Value:
    None.
Pre-conditions:
    The CPU needs to be initialized with a valid memory array.
Post-conditions:
    The CPU may be halted, and the state of the memory may be modified.
```

Here is the call graph for this function:

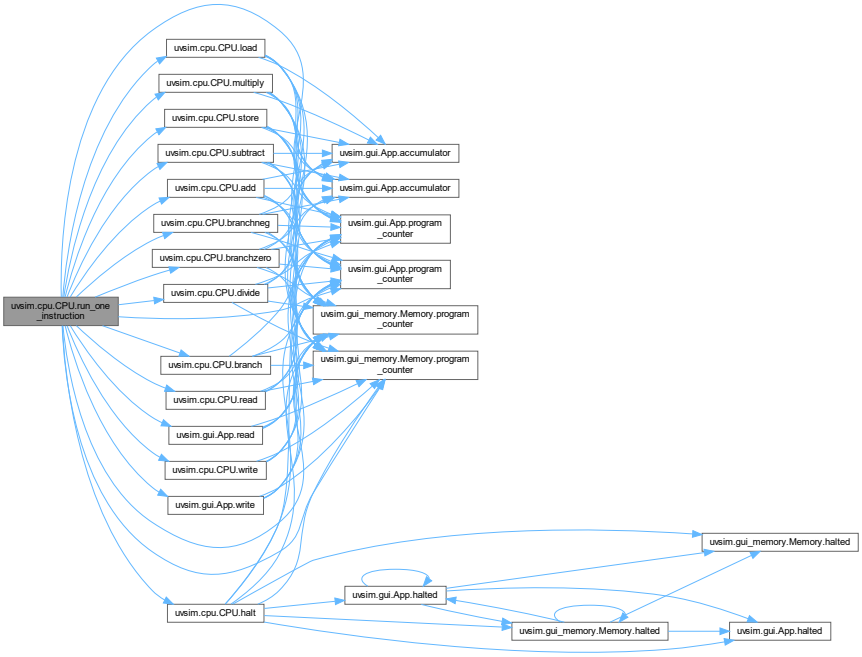


5.2.3.12 run_one_instruction()

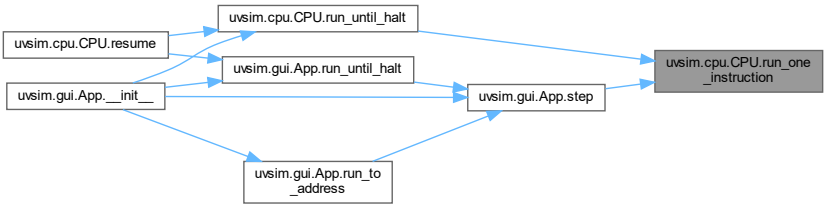
```
uvsim.cpu.CPU.run_one_instruction (
              self )
```

```
Purpose:
    Does a single instruction.
Input Parameters:
    None.
Return Value:
    An error code saying the result of the instruction that was executed.
Pre-conditions:
    The CPU needs to be initialized with a valid memory array.
Post-conditions:
    The state of the memory and CPU registers may be modified.
```

Here is the call graph for this function:



Here is the caller graph for this function:



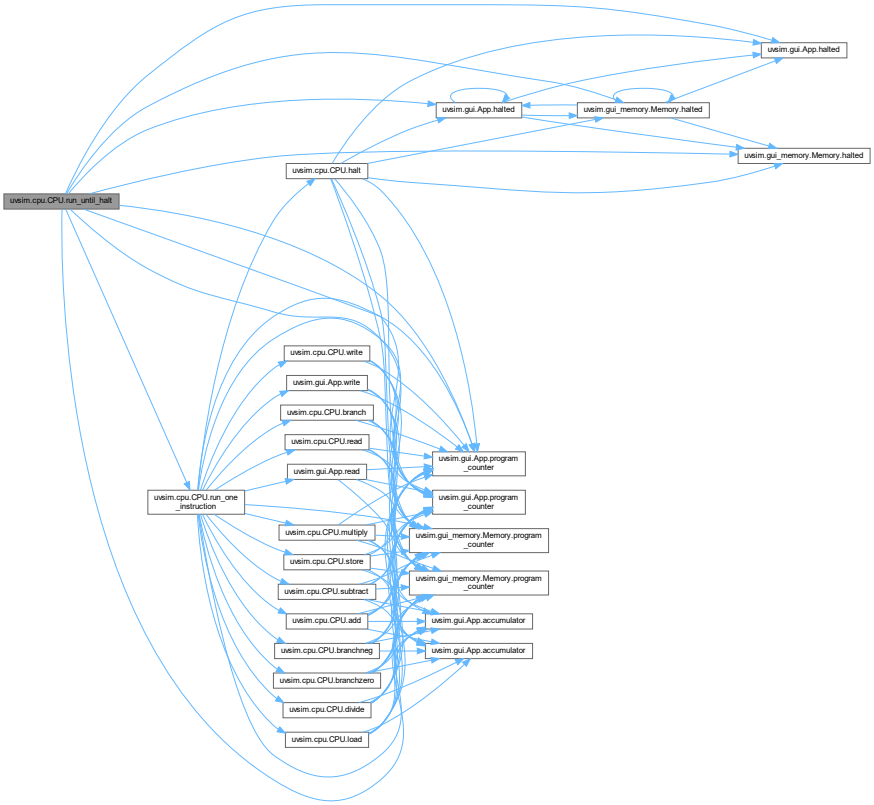
5.2.3.13 run_until_halt()

```
uvsim.cpu.CPU.run_until_halt (
              self )
```

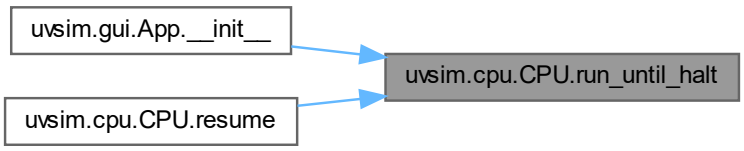
```
Purpose:
    Does instructions in a loop until the CPU is halted.
Input Parameters:
    None.
Return Value:
    An error code with the reason for halting.
Pre-conditions:
    The CPU needs to be initialized with a valid memory array.
Post-conditions:
    The CPU could be halted, and the state of the memory may be modified.
```

Reimplemented in **uvsim.gui.App** (p. 20).

Here is the call graph for this function:



Here is the caller graph for this function:

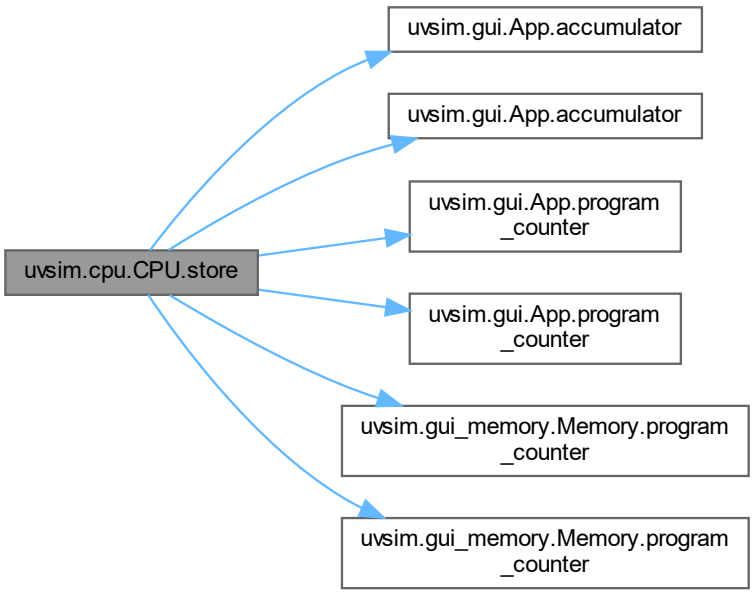


5.2.3.14 store()

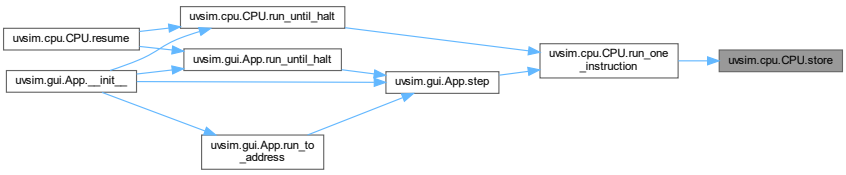
```
uvsim.cpu.CPU.store (
            self,
            data )
```

```
Purpose:
    Stores the value of the accumulator into a memory location.
Input Parameters:
    data: The memory location where the accumulator value will be stored.
Return Value:
    An error code showing the result of the operation.
Pre-conditions:
    The CPU needs to be initialized with a valid memory array.
Post-conditions:
    The specified memory location is updated with the accumulator value.
```

Here is the call graph for this function:



Here is the caller graph for this function:

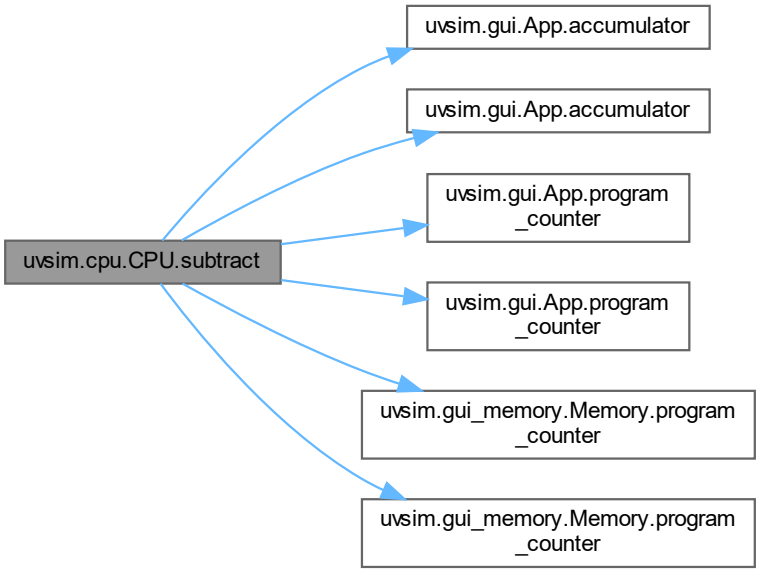


5.2.3.15 subtract()

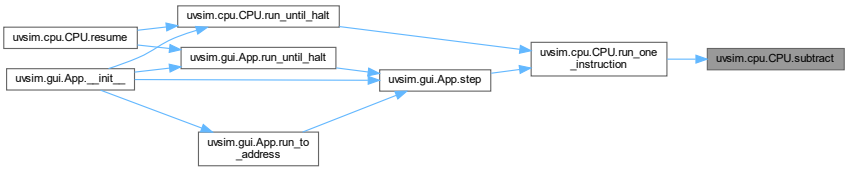
```
uvsim.cpu.CPU.subtract (
            self,
            data )
```

```
Purpose:
    Subtracts the value at a specified memory location from the accumulator.
Input Parameters:
    data: The memory location of the value that will be subtracted.
Return Value:
    An error code showing the result of the operation.
Pre-conditions:
    The CPU needs to be initialized with a valid memory array.
Post-conditions:
    The accumulator is updated with the subtraction result.
```

Here is the call graph for this function:



Here is the caller graph for this function:



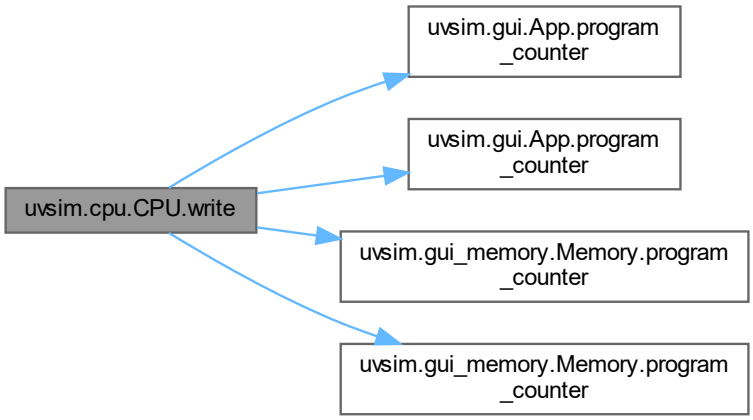
5.2.3.16 write()

```
uvsim.cpu.CPU.write (
            self,
            data )
```

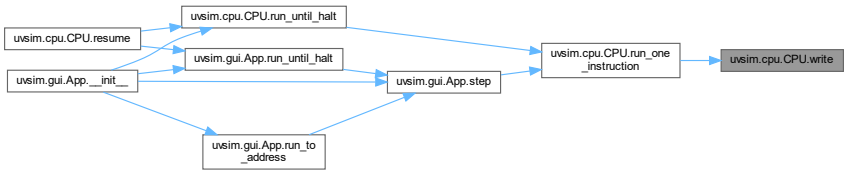
```
Purpose:
    Writes a word from memory to the console.
Input Parameters:
    data: The memory location of the word that will be written.
Return Value:
    An error code showing the result of the operation.
Pre-conditions:
    The CPU needs to be initialized with a valid memory array.
Post-conditions:
    The word is printed to the console.
```

Reimplemented in **uvsim.gui.App** (p. 23).

Here is the call graph for this function:



Here is the caller graph for this function:



5.2.4 Member Data Documentation

5.2.4.1 accumulator

`uvsim.cpu.CPU.accumulator`

Reimplemented in **uvsim.gui.App** (p. 12), and **uvsim.gui.App** (p. 13).

5.2.4.2 halted

`uvsim.cpu.CPU.halted`

Reimplemented in **uvsim.gui.App** (p. 14), and **uvsim.gui.App** (p. 15).

5.2.4.3 program_counter

`uvsim.cpu.CPU.program_counter`

Reimplemented in **uvsim.gui.App** (p. 16), and **uvsim.gui.App** (p. 17).

The documentation for this class was generated from the following file:

- C:/code_projects/school/CS2450/FinalProject/uvsim/cpu.py

5.3 uvsim.editor.Editor Class Reference

Public Member Functions

- None **__init__** (self, tk.Tk master, tk.Tk parent, bool is_main=False)
- **run** (self)
- **open_file** (self, check_6dp=True)
- **save** (self)
- **save_as** (self)
- **copy** (self)
- **cut** (self)
- **paste** (self, event=None)

Public Attributes

- **master**
- **parent**
- **open_file_path**
- **master_frame**
- **menu_bar**
- **file_menu**
- **edit_menu**
- **help_menu**
- **upper_frame**
- **text_box**
- **label**
- **lower_frame**
- **bttn**
- **program**

5.3.1 Detailed Description

```
Editor Class
Purpose of the Class:
    The Editor class is used to create a GUI for the UVSim Editor. In the editor the user can write and edit p
    The editor also allows the user to save and open files.
```

5.3.2 Member Function Documentation

5.3.2.1 copy()

```
uvsim.editor.Editor.copy (
              self )
```

```
Purpose:
    Add the copy event to the text box.
Input Parameters:
    None.
Return Value:
    None.
Pre-conditions:
    the text box must exist. there must be text in the text box.
Post-conditions:
    the text in the text box is copied to the clipboard.
```

5.3.2.2 cut()

```
uvsim.editor.Editor.cut (
              self )
```

```
Purpose:
    Add the cut event to the text box.
Input Parameters:
    None.
Return Value:
    None.
Pre-conditions:
    the text box must exist. there must be text in the text box.
Post-conditions:
    the text in the text box is cut to the clipboard. The text is removed from the text box.
```

5.3.2.3 open_file()

```
uvsim.editor.Editor.open_file (
              self,
              check_6dp = True )
```

```
Purpose:
    Opens a file dialog to allow the user to select a file and loads its content into the CPU memory.
Input Parameters:
    None.
Return Value:
    None.
Pre-conditions:
    Needs to be selected from the menu.
Post-conditions:
    The memory content is set to the content of the selected file.
```

5.3.2.4 paste()

```
uvsim.editor.Editor.paste (
              self,
              event = None )

Purpose:
    Add the paste event to the text box.
Input Parameters:
    None.
Return Value:
    None.
Pre-conditions:
    the clipboard must have text in it.
Post-conditions:
    the text in the clipboard is pasted into the text box.
```

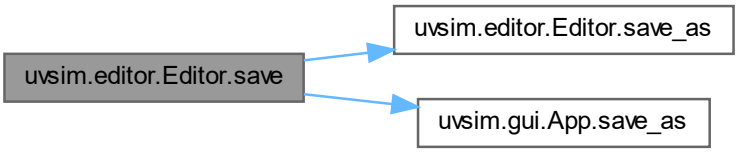
5.3.2.5 run()

```
uvsim.editor.Editor.run (
              self )

Purpose:
    Loads the content of the text box into the CPU memory.
Input Parameters:
    None.
Return Value:
    None.
Pre-conditions:
    Text box must have content.
Post-conditions:
    The CPU memory is set to the content of the text box.
```

5.3.2.6 save()

```
uvsim.editor.Editor.save (
              self )

Purpose:
    Opens a file dialog to allow the user to save the current memory content to an existing file.
Input Parameters:
    None.
Return Value:
    None.
Pre-conditions:
    The memory content must be valid.
Post-conditions:
    The memory content is saved to an existing file.
```

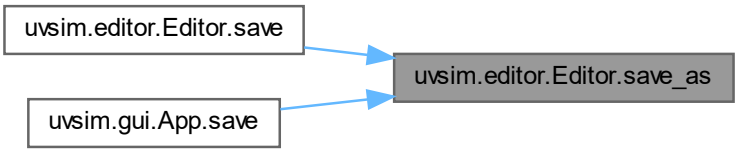
Here is the call graph for this function:



5.3.2.7 save_as()

```
uvsim.editor.Editor.save_as (
              self )


Purpose:
    Opens a file dialog to allow the user to save the current memory content to a new file.
Input Parameters:
    None.
Return Value:
    None.
Pre-conditions:
    The memory content must be valid.
Post-conditions:
    The memory content is saved to a new file.
```

Here is the caller graph for this function:

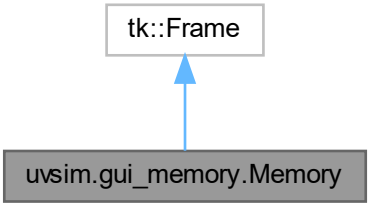


The documentation for this class was generated from the following file:

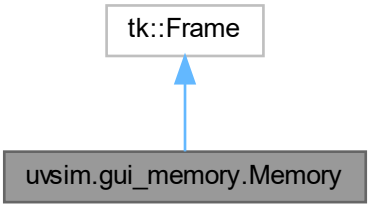
- C:/code_projects/school/CS2450/FinalProject/uvsim/editor.py

5.4 uvsim.gui_memory.Memory Class Reference

Inheritance diagram for uvsim.gui_memory.Memory:



Collaboration diagram for uvsim.gui_memory.Memory:



Public Member Functions

- None **__init__** (self, list[int] memory, tk.Misc|None master, vcmd)
- **__getitem__** (self, key)
- None **__setitem__** (self, str key, value)
- **program_counter** (self)
- **program_counter** (self, value)
- **halted** (self)
- **halted** (self, value)

Public Attributes

- **label**
- **memory_vars**
- **memory_frames**
- **vertical_labels**
- **horizontal_labels**
- **defaultbg**
- **program_counter**
- **halted**
- **selected_frames**
- **clipboard_content**
- **start_cell**
- **end_cell**

Protected Attributes

- **_program_counter**
- **_halted**

5.4.1 Detailed Description

```
The Memory class represents the memory of the UVSim CPU in a GUI.
It inherits from the tk.Frame class.
It gives a visual representation of the memory by crating a grid of memory cells.
```

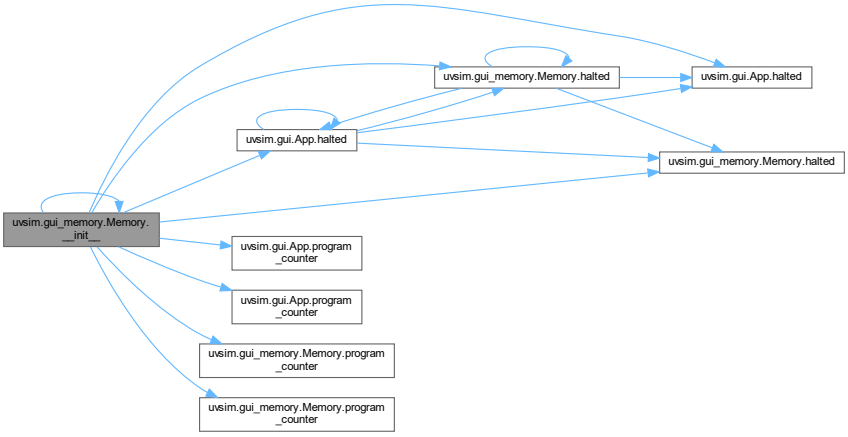
5.4.2 Constructor & Destructor Documentation

5.4.2.1 __init__()

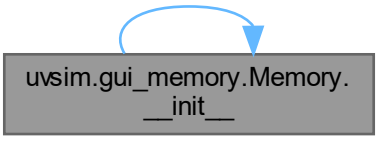
```
None uvsim.gui_memory.Memory.__init__ (
             self,
            list[int] memory,
            tk.Misc | None master,
             vcmd )
```

```
Purpose:
    Initializes the Memory class with the provided memory array and sets up the GUI elements to represent the
Input Parameters:
    memory: An array representing the memory content.
    master: The master widget of the memory frame.
    vcmd: The validation command for memory entries.
Return Value:
    None.
Pre-conditions:
    The memory array must be valid.
Post-conditions:
    The memory GUI is set up.
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.4.3 Member Function Documentation

5.4.3.1 __getitem__()

```
uvsim.gui_memory.Memory.__getitem__ (
              self,
              key )
```

```
Purpose:
    Allows accessing the memory content.
Input Parameters:
    key: Index or slice to access the memory content.
Return Value:
    The value at the index or slice.
Pre-conditions:
    The key must be a valid index or slice.
Post-conditions:
    None.
```

5.4.3.2 __setitem__()

```
None uvsim.gui_memory.Memory.__setitem__ (
              self,
             str key,
              value )
```

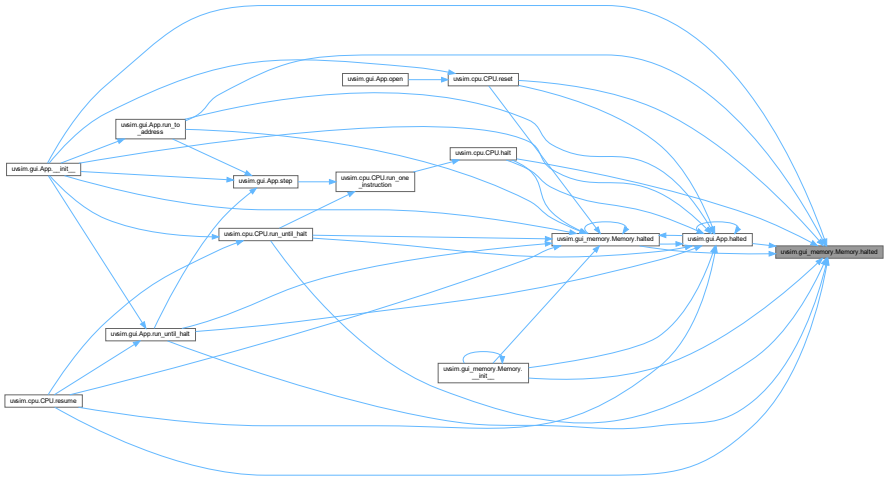
```
Purpose:
    Allows setting the memory content.
Input Parameters:
    key: Index or slice to set the memory content.
    value: The value that will be set.
Return Value:
    None.
Pre-conditions:
    The key must be a valid index or slice.
Post-conditions:
    The memory content is updated.
```

5.4.3.3 halted() [1/2]

```
uvsim.gui_memory.Memory.halted (
              self )
```

```
Purpose:
    Getter and setter for the halted property.
    It updates the GUI to highlight the current program counter cell with a different color if the CPU is halt
Input Parameters:
    None.
Return Value:
    True if the CPU is halted, False otherwise.
Pre-conditions:
    None.
Post-conditions:
    None.
```

Here is the caller graph for this function:

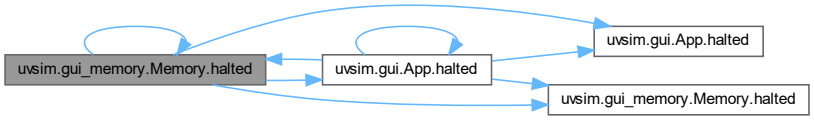


5.4.3.4 halted() [2/2]

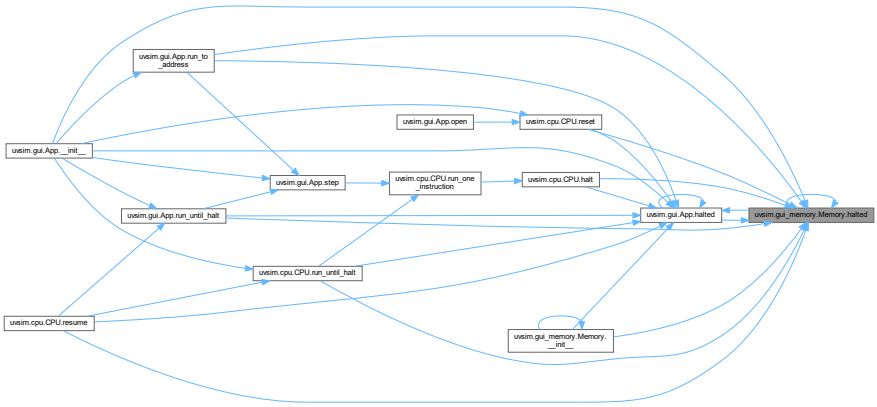
```
uvsim.gui_memory.Memory.halted (
              self,
              value )
```

```
Purpose:
    Setter for the halted property.
    It updates the GUI to highlight the current program counter cell with a different color if the CPU is halt
Input Parameters:
    value: The new value of the halted property.
Return Value:
    None.
Pre-conditions:
    None.
Post-conditions:
    The halted property is updated.
```

Here is the call graph for this function:



Here is the caller graph for this function:

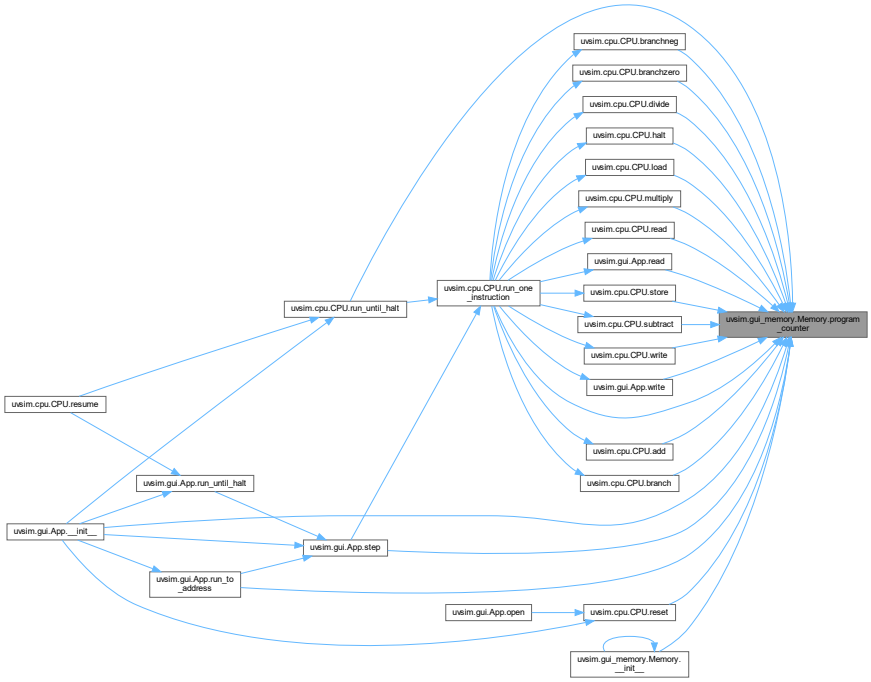


5.4.3.5 program_counter() [1/2]

```
uvsim.gui_memory.Memory.program_counter (
            self )
```

```
Purpose:
    Getter and setter for the program counter property.
    It updates the GUI to highlight the current program counter cell.
Input Parameters:
    None.
Return Value:
    The current value of the program counter.
Pre-conditions:
    The program counter must be a valid index.
Post-conditions:
    None.
```

Here is the caller graph for this function:

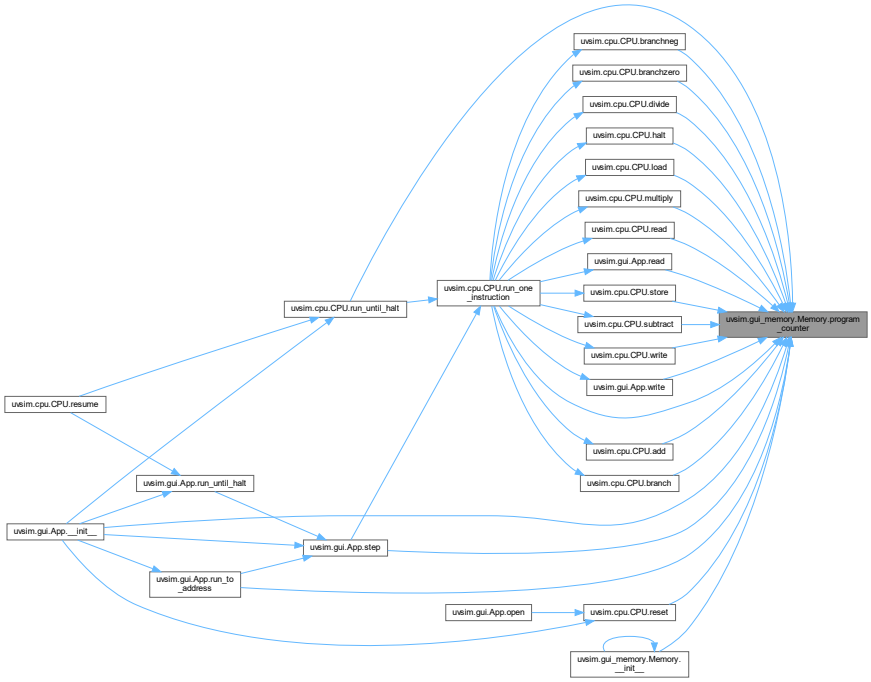


5.4.3.6 program_counter() [2/2]

```
uvsim.gui_memory.Memory.program_counter (
            self,
            value )
```

```
Purpose:
    Setter for the program counter property.
    It updates the GUI to highlight the current program counter cell.
Input Parameters:
    value: The new value of the program counter.
Return Value:
    None.
Pre-conditions:
    The value must be a valid index.
Post-conditions:
    The program counter is updated.
```

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

- C:/code_projects/school/CS2450/FinalProject/uvsim/gui_memory.py

5.5 uvsim.tutorial.Tutorial Class Reference

Public Member Functions

- None **__init__** (self, tk.Tk master)
- **get_next** (self)
- list[str] **open_images** (self)

Public Attributes

- **master**
- **master_frame**
- **upper_frame**
- **image_list**
- **image_count**
- **image_iter**
- **current_image**
- **lower_frame**
- **bttns**

5.5.1 Detailed Description

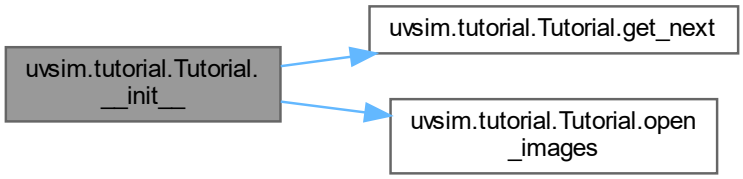
```
The Tutorial class will display a tutorial for the UVSim application using images.
It creates a simple Tkinter GUI with buttons to navigate through the tutorial images.
```

5.5.2 Constructor & Destructor Documentation

5.5.2.1 __init__()

```
None uvsim.tutorial.Tutorial.__init__ (
              self,
             tk.Tk master )


Purpose:
    Initializes the Tutorial class with the provided master widget and sets up the GUI elements for displaying
Input Parameters:
    master: The master widget of the Tutorial frame.
Return Value:
    None.
Pre-conditions:
    The master widget must be valid.
Post-conditions:
    The tutorial GUI is set up.
```

Here is the call graph for this function:

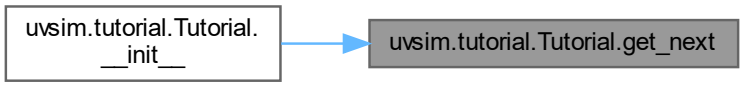


5.5.3 Member Function Documentation

5.5.3.1 get_next()

```
uvsim.tutorial.Tutorial.get_next (
              self )


Purpose:
    Updates the displayed image to the next image in the tutorial when the "Next" button is clicked.
Input Parameters:
    None.
Return Value:
    None.
Pre-conditions:
    The tutorial must be initialized.
Post-conditions:
    The next image in the tutorial is displayed.
```

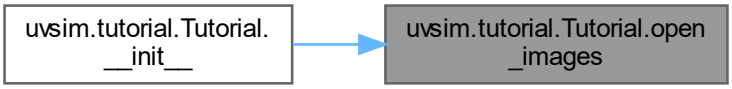
Here is the caller graph for this function:



5.5.3.2 open_images()

```
list[str] uvsim.tutorial.Tutorial.open_images (
              self )


Purpose:
    Opens tutorial images from a specified directory and converts them into Tkinter PhotoImage objects.
Input Parameters:
    None.
Return Value:
    A list of Tkinter PhotoImage objects representing the tutorial images.
Pre-conditions:
    The tutorial images must be in the specified directory.
Post-conditions:
    The tutorial images are opened and converted into Tkinter PhotoImage objects.
```

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

- C:/code_projects/school/CS2450/FinalProject/uvsim/tutorial.py

Index