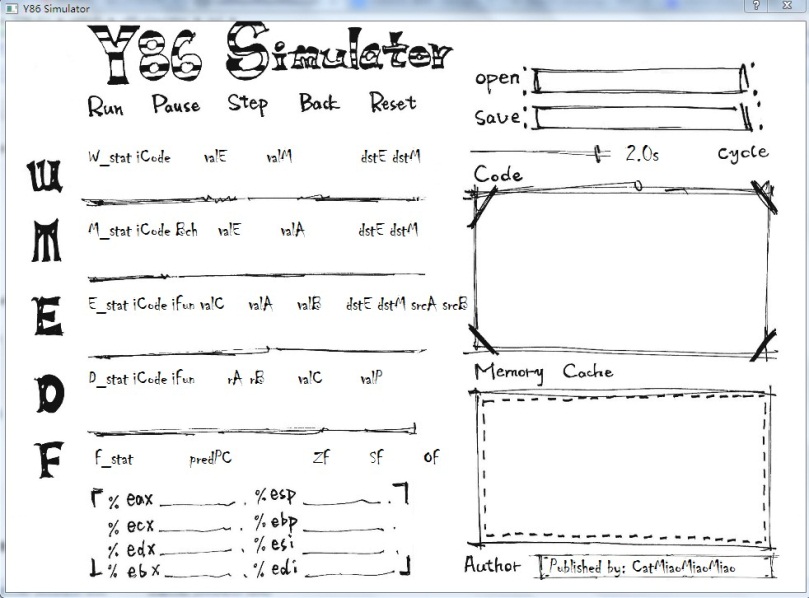
GUI

1. overview

Despite the excellent command line interface, we still offer a friendly GUI interface to give our users a better experience. It will be easy for our users to get the expected information of register, program status, memory and cache, for special. For more details to install it and run it, refer to our readme file.

1. GUI Design

The Graphic interface has an amazing hand-drawn background, which takes lots of efforts. It is a sketch drawing that presents all the information well. Sometimes it may be messy and unendurable for some person, especially when you run some extremely complicated programs. In such cases, you can consider to use our command line version. Our design servers to display more beautiful interface at the stand of art and gives the user a more comfortable enjoyment.



1. GUI realization toolkit

Considering all the toolkits for python GUI designing, we choose PyQT4 to construct our Graphic interface due to its convenience and good compatibility to all platforms including windows, linux and MacOS. QT itself is such a powerful and widely applied library with an impeccable reference that can decrease our work to a great extent.

1. Functions

The GUI version supports both .yo and .ys files, which is different to the command line. Once a .ys file is imported, the y86 assembler will be called automatically to assemble the codes to binary ones that can be executed by our simulator. Once an invalid .ys file is loaded, the program will automatically call the GUI version of the assembler to help the user fix the problems. The basic functions of GUI are to be listed as following:

·display all the information you need including register, program status, cycles, source code, memory and cache

* support run, pause, step ,back and reset instructions
* change the frequency of the simulator dynamically at any time
* debug your program in our assembler
* a good handle of exceptions
* a special function to express our honor to D.Jin

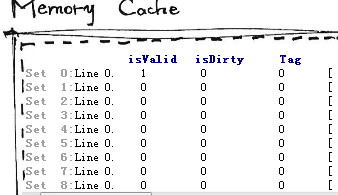
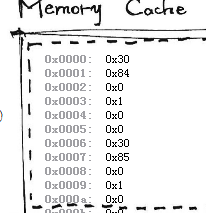
1. Technical implementation

Due to our Maximize Cohesion and Minimize Coupling designing schema, the GUI part is quiet separate from our simulator and assembler/disassembler, which is quiet a great advantage for use to implement all of our functions.

The GUI get all the details of every cycle and every stage by creating a Simulator class as its own. Then it calls function simulator.load() to convey the input file and output route to the simulator class and then calls simulator.run() to start the simulation. All the information it needs to present can be fetched in the private data of the simulator class. In GUI designing, we have no need at all to consider the details of the kernel design.

To compile the .ys input file, the GUI import assemble.py and simply use the assemble function. The assembly code are stored in an output file with a same prefix-name. Then our GUI import this output file and run it normally.

The highlight of our codes uses the QSyntaxHighLighter class in highlighter.py proposed by official docs of QT, which is quiet cool. The memory window only displays all changed memory address since the program runs. Our cache window displays all the cache.

We also use qss layout tools to help beautify our GUI.

Assemble/Disassemble environment

1. To Get Start

  The assembling/disassembling tools follow the rules as described in CSAPP 1st editon and both have command line version and GUI version as well. More details are also offered in the readme file



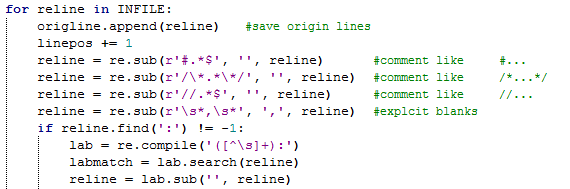


1. the Assembler

The assembler supports all the y86 instruction in CSAPP book and some even more features as follows:

* All the y86 instructions even
* .pos, .long, .word, .byte instruction
* Labels for jump and call
* Jump to a direct address
* Support widely used comment format # , /\*\*/ , //
* Retain all the original lines to help comprehension
* Show different error messages and the line position after assembling

We use regular expression, a recently widely applied technic to speed up the analysis as follows.



1. Disassemler

The disassembling tool does not support .long or .byte instruction. Every time it finds the address of an instruction not match its length, the .pos instruction will be applied to fix it. In the GUI mode, hit run to test your code in our simulator easily.

Apart from what is mentioned above, the disassembler enjoys as more functions and features as the assembler.

1. GUI mode(Text Editor)

Another hand-drawn graphic interface with two code display window and a console. You can run immediately in our GUI simulator after you have modified/compiled your codes.

