Running a Commodore 64 Emulator on Linux Mint

4digits.net/blog/retrocomputing/commodore-64-emulator-linux.html

Posted in: Retrocomputing | Written by: Johannes Dreller | On: March 02, 2016

If you grew up in the 80s, chances are high that you made your first steps in computing with a Commodore 64. Produced between 1982 and 1994, the C64 was the most popular home computer of its generation, featuring an 8-bit microprocessor and 64 kilobyte of memory. Due to its competitive pricing and aggressive marketing strategy, the Commodore 64 quickly outsold its competitors Apple II, IBM 5150 and Atari 400/800 in the consumer segment. It is estimated that Commodore sold 22 million units of the *“breadbox”* worldwide, making the C64 the top-selling computer in history.

34 years after the introduction, the possibilities of modern computers have considerably changed, but the C64 still has its charm. Travelling back in time, you can emulate the Commodore 64 on a modern PC and explore low-level Assembler programming. The advantage of an emulator is that you do not need to own the original device. The idea is to write programs in a modern code editor, assemble the code and run it in your emulator. However, if you own a C64, you can transfer your emulator-tested programs and run them on the physical device.

In the following article, I will install the Versatile Commodore Emulator (VICE) on Linux Mint, a popular linux distribution based on Debian and Ubuntu. I will then show you how to load a couple of popular games. Finally, we will write a simple program in Basic, compile some Assembler and execute it in the Vice emulator.

Excited about so much C64 nostalgia? Then read on...

1. Installing VICE

Vice is currently available in latest version 2.4 and runs on Unix, Windows Mac OSX, as well as a range of other operating systems. While it emulates many other Commodore systems like VIC20 and C128, we are particulary interested in the C64. You can find the complete VICE documentation on the related website. So let's get started with the installation:

In Linux Mint go to

Menu → Software Manager → Search for "vice"

and click *"Install"*.

This will install the emulator, but it does not contain the ROM images and other source files that are needed to actually use it. We will install the missing drivers manually from the command line in the following.

Open a terminal window, access the C64-folder in the VICE directory and create a temporary directory:

```
cd /usr/lib/vice/C64

sudo mkdir temporary
```

Enter the temporary folder, download the Vice 2.4 source code provided by Bo Zimmermann and extract the archive once the download is completed:

```
cd temporary

sudo wget
http://zimmers.net/anonftp/pub/cbm/crossplatform/emulators/VICE/vice-2.4.tar.gz

sudo tar vzxf vice-2.4.tar.gz
```

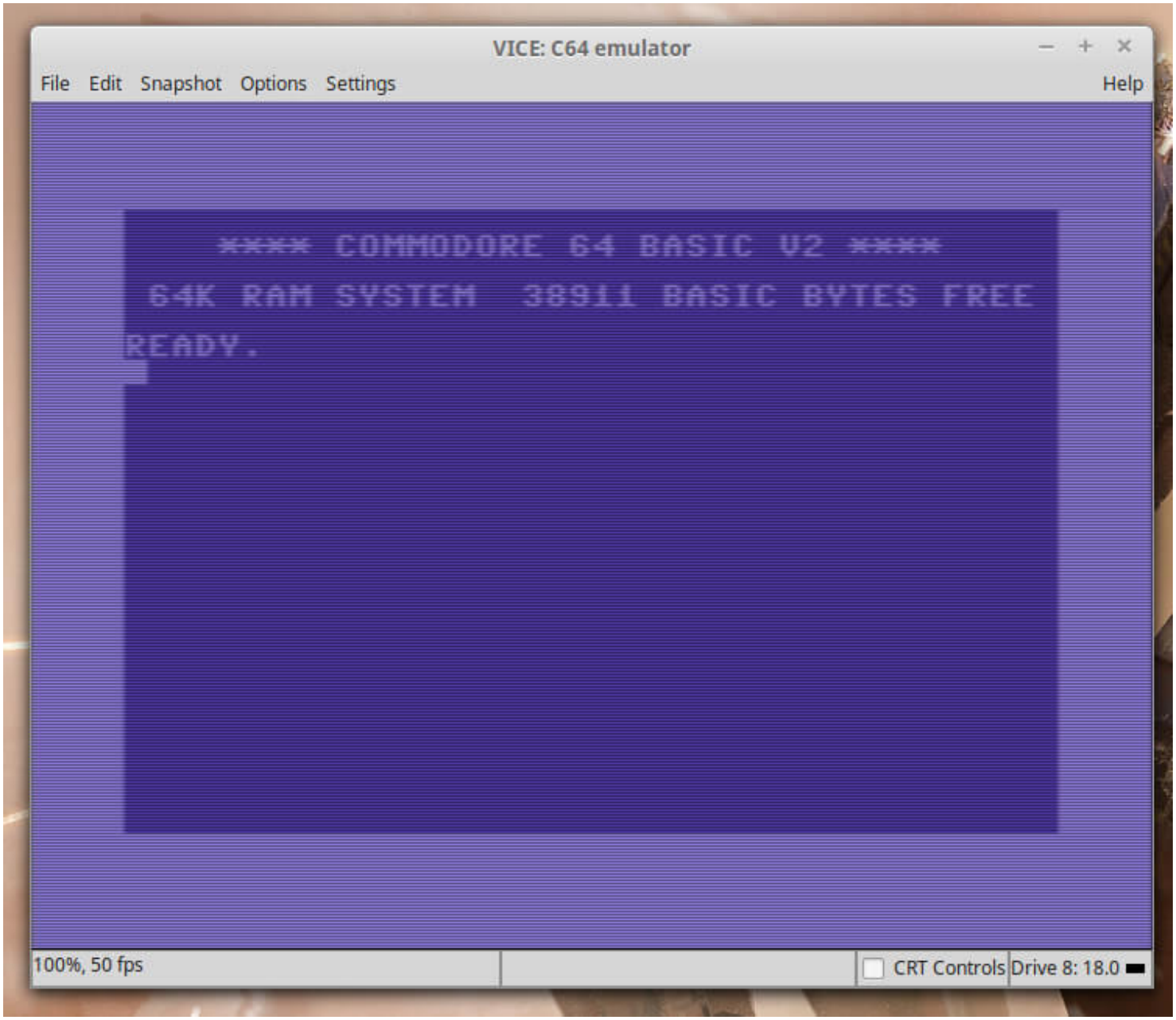
Once the archive has been extracted, copy the complete content of the data directory to the root Vice directory and change ownership:

```
sudo cp -r /usr/lib/vice/C64/temporary/vice-2.4/data/* /usr/lib/vice/

sudo chown -R <username> /usr/lib/vice
```

Move one level higher into `/usr/lib/vice/C64` and delete the temporary folder:

```
cd ..

sudo rm -rf temporary
```

Start VICE from the console with the following command, which should bring up the emulator window and load all ROM images without errors:



2. Running a Disk or Tape Image

Please note that despite of their age, the C64 software and games still fall under copyright protection. The U.S. copyright law, for instance, grants corporations a protection period of 75 years from the first date of publication. While many intellectual property owners will not mind that their old products are distributed for non-commercial purposes, the fact that you copy software from the Internet or another source still constitutes an act of piracy. Emulating a device like the Commodore 64 is perfectly legal, but downloading or uploading an image from or to the Internet is NOT. To legally use an image on your system, you must own a commercial copy of the original piece of software. You can then rip an image from the original files as a personal backup.

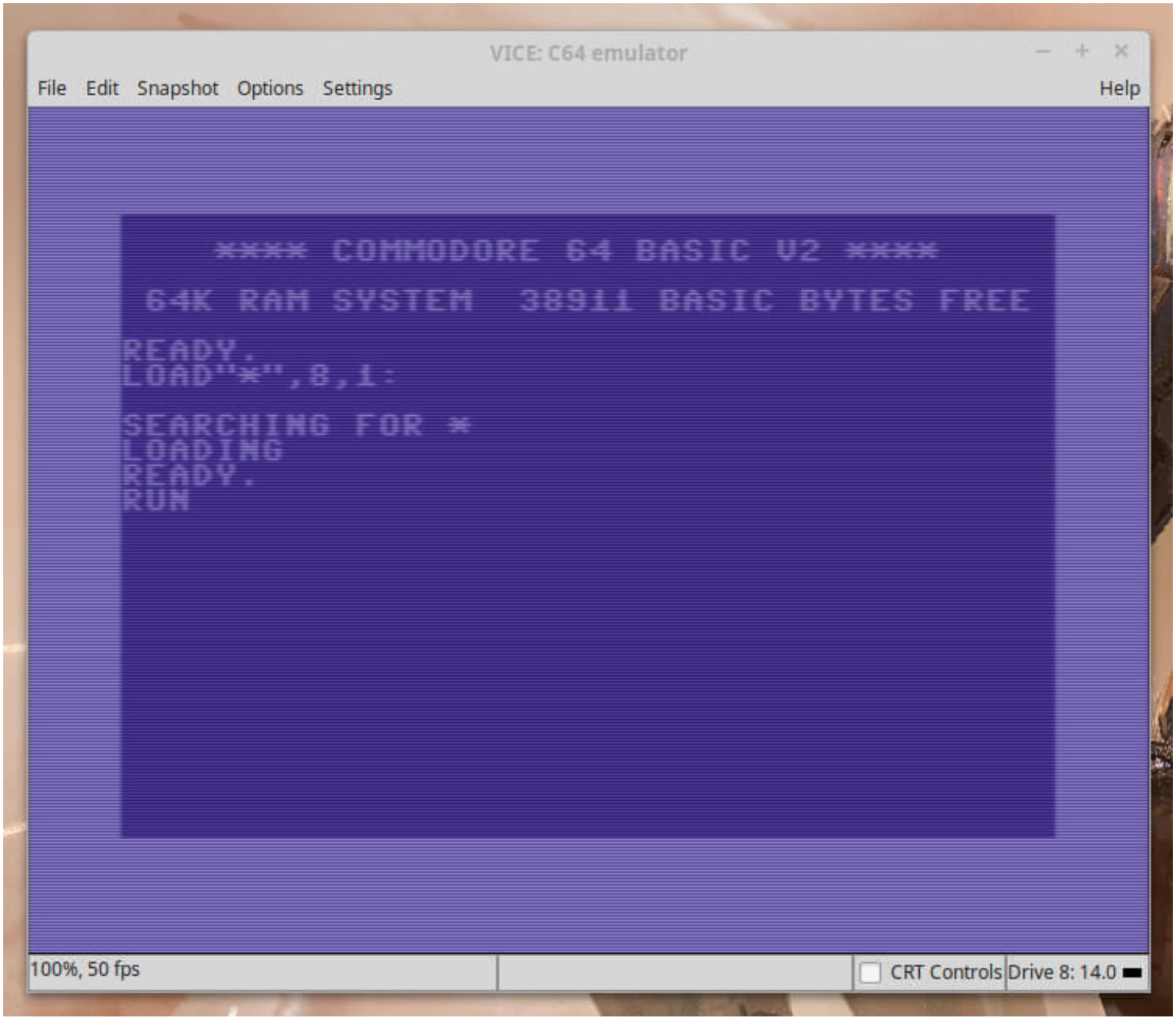
Now that we got our C64 emulator running and that being said, let's load a disk image. VICE uses digital image files of the original C64 compact cassette tapes and 5,25” floppy disks. The `.d64`-format of up to 174 kB is the most

frequently used format for floppy disk images, while `.t64` is used for copies of tape records. Their handling in VICE is very similar.

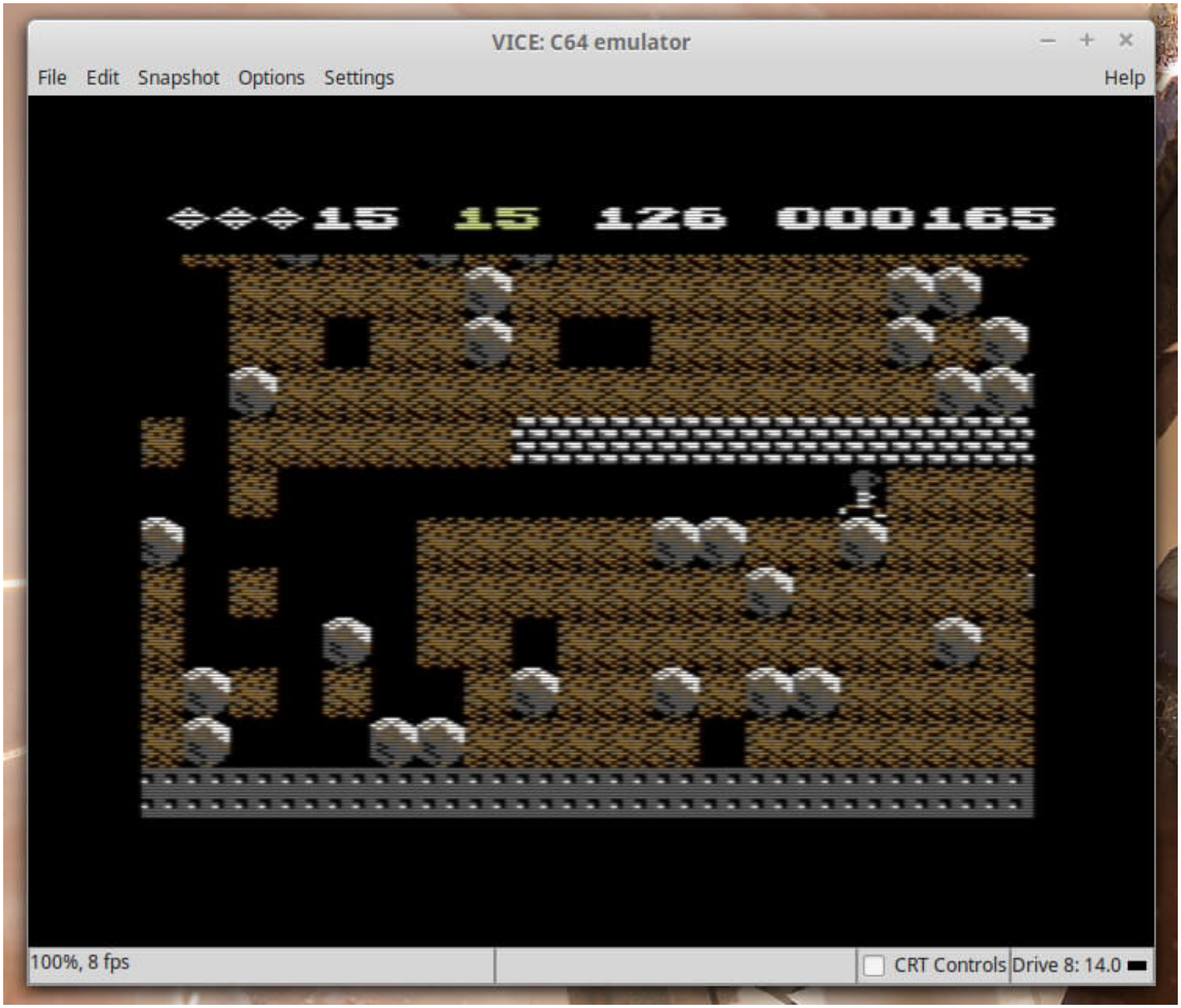
Considering that you have stored a `.d64` image (in our example the popular game “Boulder Dash”) on your hard drive, start the emulator and go to:

File → Attach disk image → Unit #8

This virtually inserts the disk into the disk drive. Browse to the exact location of your image, select it and click *“autostart”*, which should directly load like in the following:



We have now loaded the image into slot #8 and the game starts:



It is also possible to *"autoload"* an image directly from the command line providing the image name as an argument:

```
x64 'Boulder Dash.d64'
```

A third, less convenient possibility is to preview the image content and manually start a program after *"virtually"* inserting it in a slot. Again, go to

File → Attach disk image → Unit #8

Browse to the storage directory, select the required image and click *"open"*, which *"virtually"* inserts the image in the floppy drive. Within the editor window, load the content of the image in slot 8:

```
LOAD "$",8
```

Check the content of the floppy:

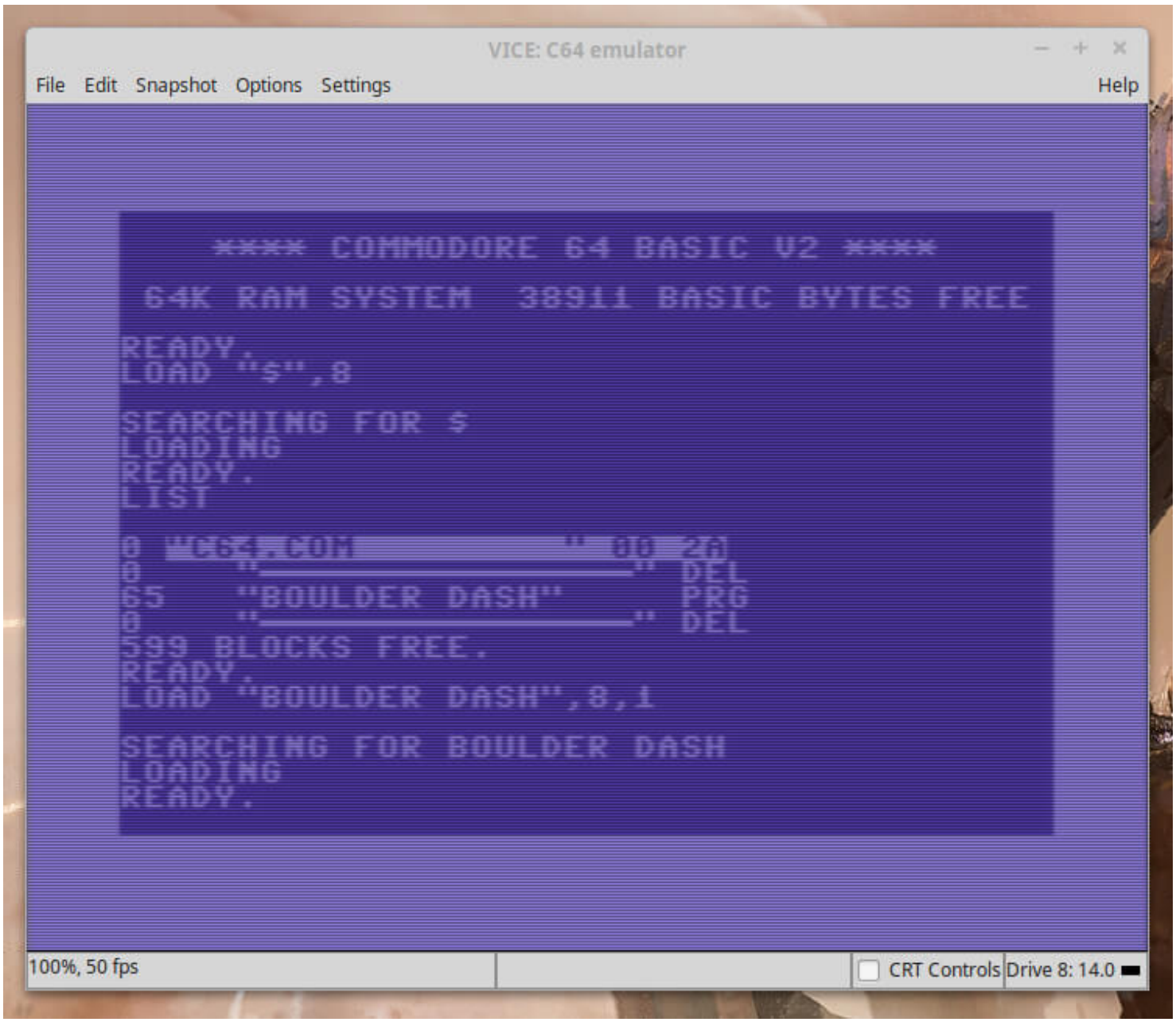
```
LIST
```

Items listed with PRG are programs. To load a specific program to the Commodore's memory, in our case “BOULDER DASH”, type:

```
LOAD “BOULDER DASH”,8,1
```

Confirm with *“Enter”*. Once the loading process is completed type:

```
RUN
```



3. Writing a simple Program in Basic and Assembler

The Commodore 64 shipped with two different programming languages: BASIC V2 and Assembler, which we will briefly discuss in th following.

3.1 BASIC V2

The C64 shipped with a BASIC V2-interpreter out-of-the-box, which you can see at the top of the editor window (`**** COMMODORE 64 BASIC V2 ****`). BASIC V2 was based on Microsoft's MBASIC and had been adapted for the C64. The language worked via line number references, usually with intervals of 10 or higher for each row (10, 20, 30, 40, 50, 60,…). The maximum length per row was limited to 255 characters. The interpreter did not require spaces or any other formatting marks, but spaces were used for better readability.

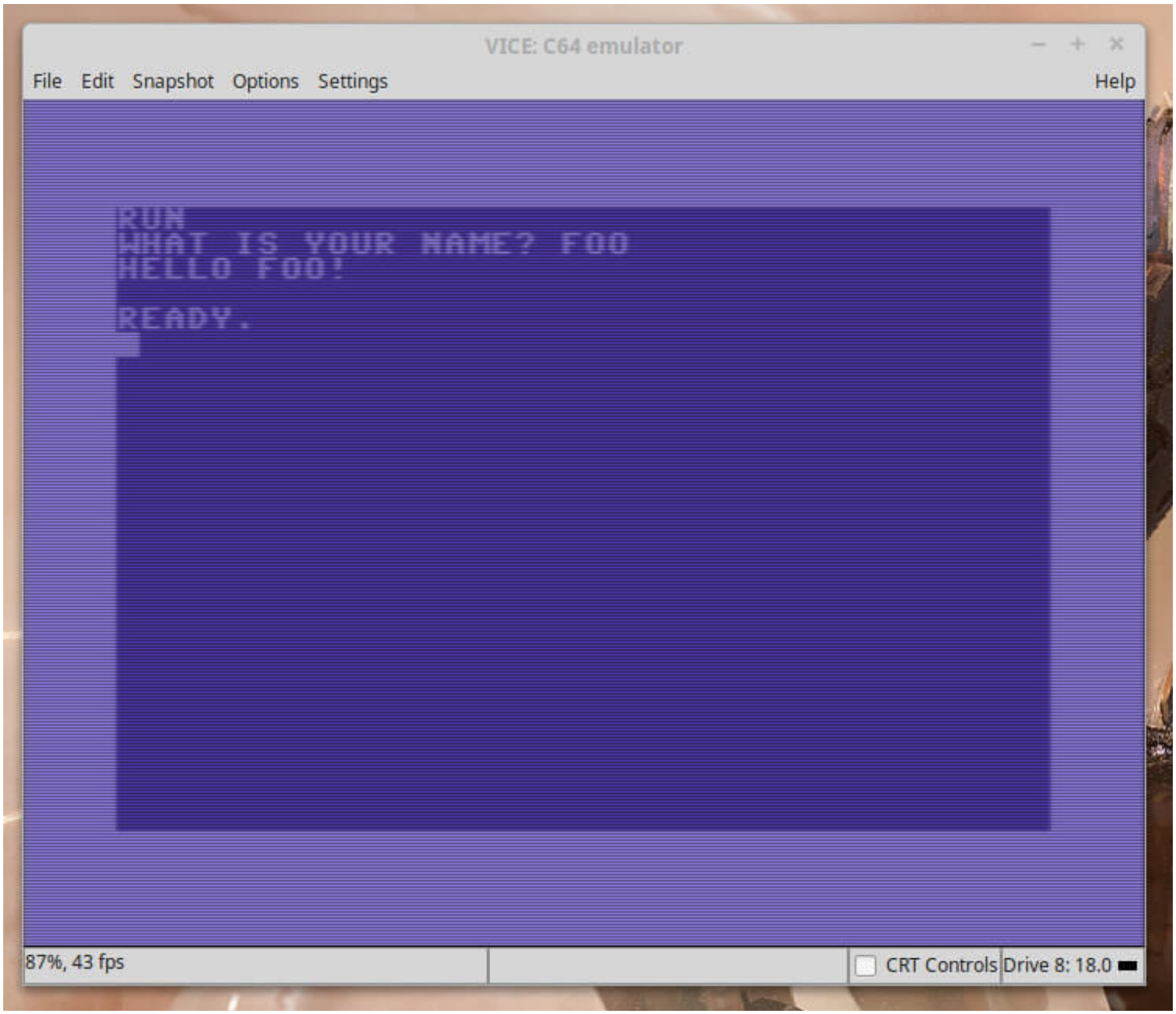
Writing Basic code directly to the editor without line numbers will simply execute that code but will not save it:

```
PRINT "HELLO WORLD!"
```

So let's write a little program that we can store temporarily in the C64's memory:

```
10 INPUT "WHAT IS YOUR NAME"; A$
20 PRINT "HELLO "; A$; "!"
```

Using the `LIST` command will show your memory content. Hitting `RUN` will execute your program.



To delete the complete memory, type `NEW`.

3.2 Assembler

Assembler, also called assembly, is a different beast compared to the relatively trivial BASIC. Assembler is low-level machine code in human-readable form that is used to give native instructions to the underlying hardware. To put it simple, machine code is the language that the microprocessor directly understands. This code is unique to each CPU architecture. In the case of the Commodore 64, we generally speak about the “6502 assembly language” for the 8-bit 6510 microprocessor designed by MOS Technology.

Mnemonics, short characterbands in the assembly language, are used to instruct different functional locations within the memory of the 6510 processor. The so called *“assembler”* then converts the instructions of the assembly language into executable machine code, i.e. it *“assembles”* the source code.

Opposed to high-level programming languages, assembly is difficult and probably nothing you can learn from scratch in a couple of days. This is not a dedicated assmbler tutorial, but I will post a list of useful resources for further study at the end. In any case, let's have a look at what you need to progam assembly for the C64 in Linux and write an example.

3.3.1 Installing Relaunch64 IDE

While you could write assembler code directly in your VICE C64 emulator using the internal machine code monitor, it is more convenient to use a cross-assembler in combination with an IDE or code editor on an external computer.

A cross-assembler is a program that compiles assembler code into a C64 executable program. There are many different C64 cross-assemblers for different operating systems. Popular assemblers are ACME, Kick Assembler, DASM, cc65 and TMPx, which partly come with different syntax complicating the matter further.

In our example, we will use the Relaunch64 IDE in combination with the ACME cross-assembler. You could also use your favourite editor, install your preferred cross-compiler and compile Assembler manually. So let's first install Relaunch64 from command line, which requires Java Runtime 7 or greater:

```
cd /usr/lib

sudo wget
https://github.com/sjPlot/Relaunch64/releases/download/3.3.5/Relaunch64_3.3.5_Linux_jar.zip

sudo unzip Relaunch64_3.3.5_Linux_jar.zip -d relaunch64

sudo rm -rf  Relaunch64_3.3.5_Linux_jar.zip

cd relaunch64

java -jar Relaunch64.jar
```

Alternatively, create a symlink on the Desktop for easier access:

```
ln -s  /usr/lib/relaunch64/Relaunch64.jar /home/<username>/Desktop
```

You can now access Relaunch64 from your Desktop by clicking the symbol with the right mouse → Open With →

Oracle Java 8 Runtime.

3.3.2 Compile ACME Cross-Assembler from Source

Unfortunately, the ACME cross-assembler is not available as a Debian package, so we have to compile it from source. The latest Linux / Unix version I could find on the author's website is 0.91. As a building folder we will use /usr/local/src, but you can use any other directory that you prefer:

```
cd /usr/local/src

wget https://web.archive.org/web/20150520143433/https://www.esw-heim.tu-clausthal.de/~marco/smorbrod/acme/current/acme091src.tar.gz

tar -xzvf acme091src.tar.gz

cd acme091/src

make
```

This should create an executable file called acme in /usr/local/src/acme091/src, which we will need in the next step.

3.3.3 Add ACME as a “Compile & Run Script”

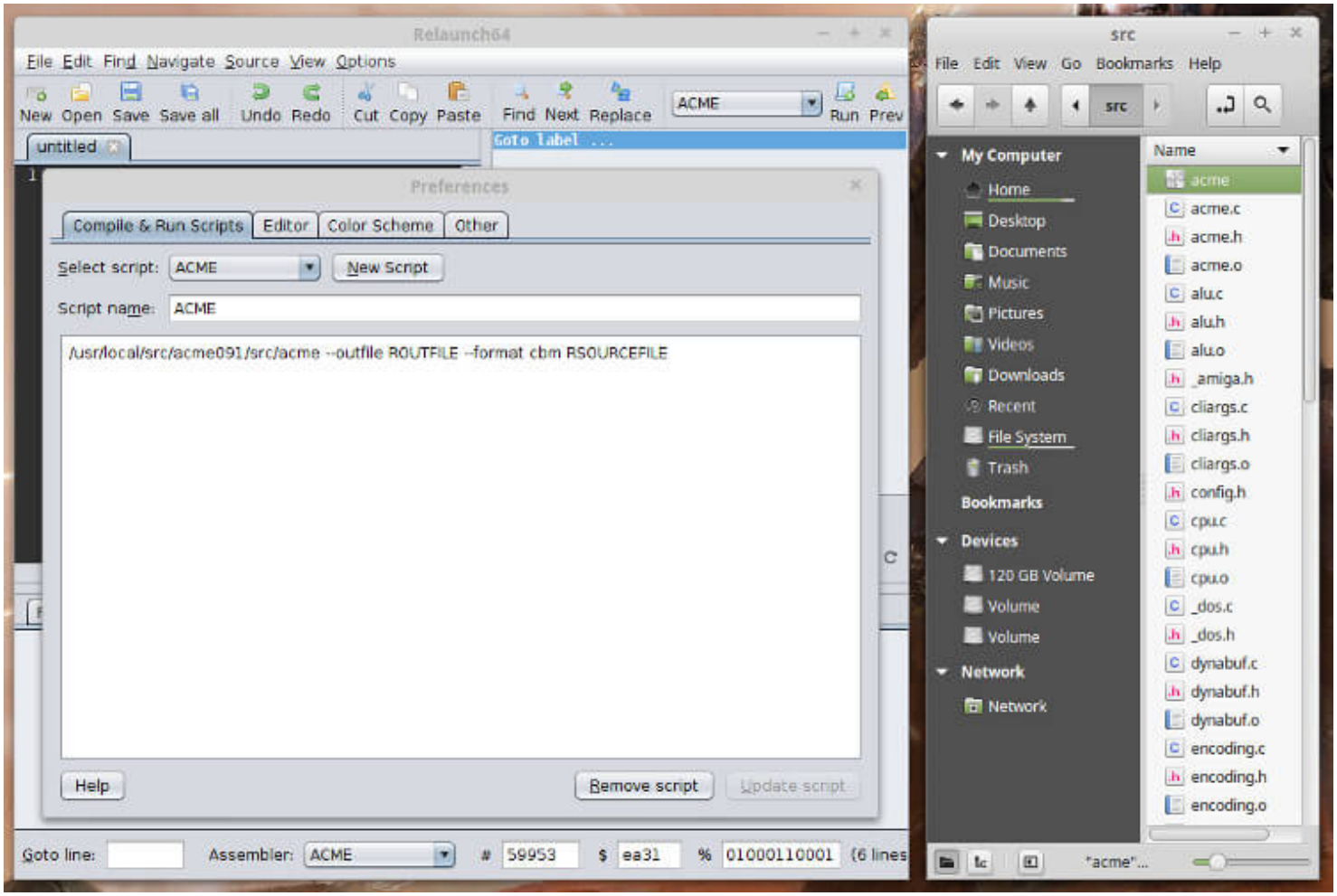
Having installed the Relaunch IDE, we need to instruct it now to use ACME as an assemble script. Inside Relaunch64, go to:

Options → Preferences

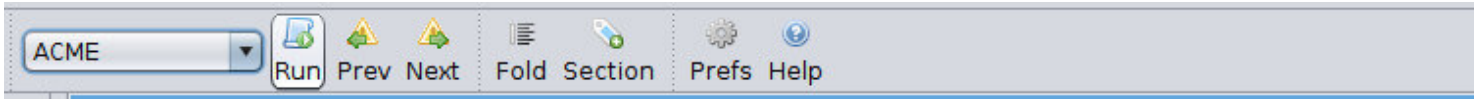
Enter a script name.

Drag and drop acme into the preferences window.

Click *“Add script”*



Your script, which we called ACME, should now be visible in the IDE's menu bar for selection:



3.3.4 Compiling Assembler code

Now that we got our development environment set up, we can compile a sample “hello world” script in assembly to test our setup. Inside Relaunch64, create a new file and save it in your prefered location as `sample.asm`.

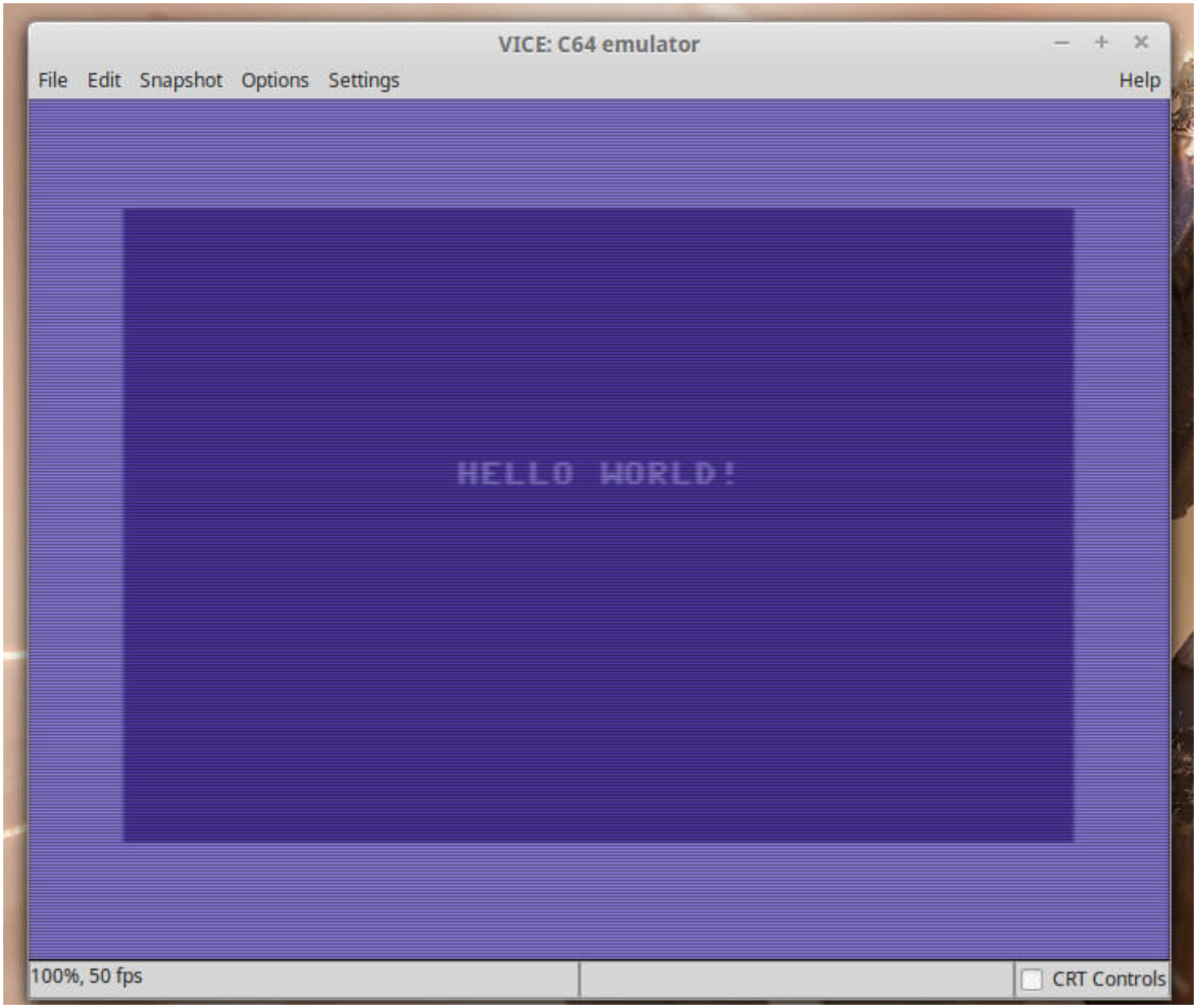
Add the following code, which is adapted from Sylvain Potrais:

```
!cpu 6502
* = $0801                               ; BASIC starts at
!byte $0d,$08,$dc,$07,$9e,$20,$34,$39    ; BASIC to load $c000
!byte $31,$35,$32,$00,$00,$00            ; inserts BASIC line: 2012 SYS 49152
* = $c000                      ; start address for 6502 code

JSR $E544
loop jsr init_text       ; write line of text
```

```
jmp loop    ; infinite loop

message   !scr "              hello world!              " ;40 cols of text

init_text  ldx
loop_text  lda message,x    ; read characters from message text...
           sta $0590,x      ; ...and place it into screen screen memory
           inx     ; increment to next character
           cpx
           bne loop_text    ; loop if false
           rts
```

Save the file and click *"Run"* with the ACME assembler script selected. This will assemble a C64 executable program called `sample.prg` in the same location where you save the `.asm`-file.

Open the Vice emulator and drag and drop sample.prg from the location folder into the emulator window, which should run your script:



4. Resources and Documentation

If you want to dive deeper into assembler programming and the Commodore 64 platform, I recommend the following online sources: