PennSim - An LC3 Simulator

See execution, check results, debug code

Using a Simulator we can

--- **See Machine's Content**
Registers: R0-R7, IR, PC, PSR (in hex)
Memory: address/content (in hex, w/ translation to .asm)
Branch conditions: CC (usually as "Z" or "N" or "P")

--- **Alter Machine's Content**
Registers
Memory locations

--- **Execute instructions**:
STEP (exactly **1 instruction**)
RUN (**w/o stopping until hitting a breakpoint, aka "Continue"**)
STOP (**stop running**)
BREAK (**stop** when PC points to **a particular memory location**)
NEXT (run until reaching instruction following this one; same as STEP, except if the instruction is a function call or trap, then runs all nested calls to completion without stopping.)

--- **Set breakpoints**:
Mark memory locations for BREAK

---

tools/**PennSim.jar**

PennSim's LC3 hardware is not the same as our LC3 implementation or P&P's LC3.

Some of PennSim's features:

---- a second display, which is a graphics-mode display
---- a memory-protection register
---- timer and timer-control registers (w/o interrupt capability)
---- does not implement interrupts
---- does not recognize the branch instruction, 0000_000_000000000

PennSim.jar

LC3 Simulator

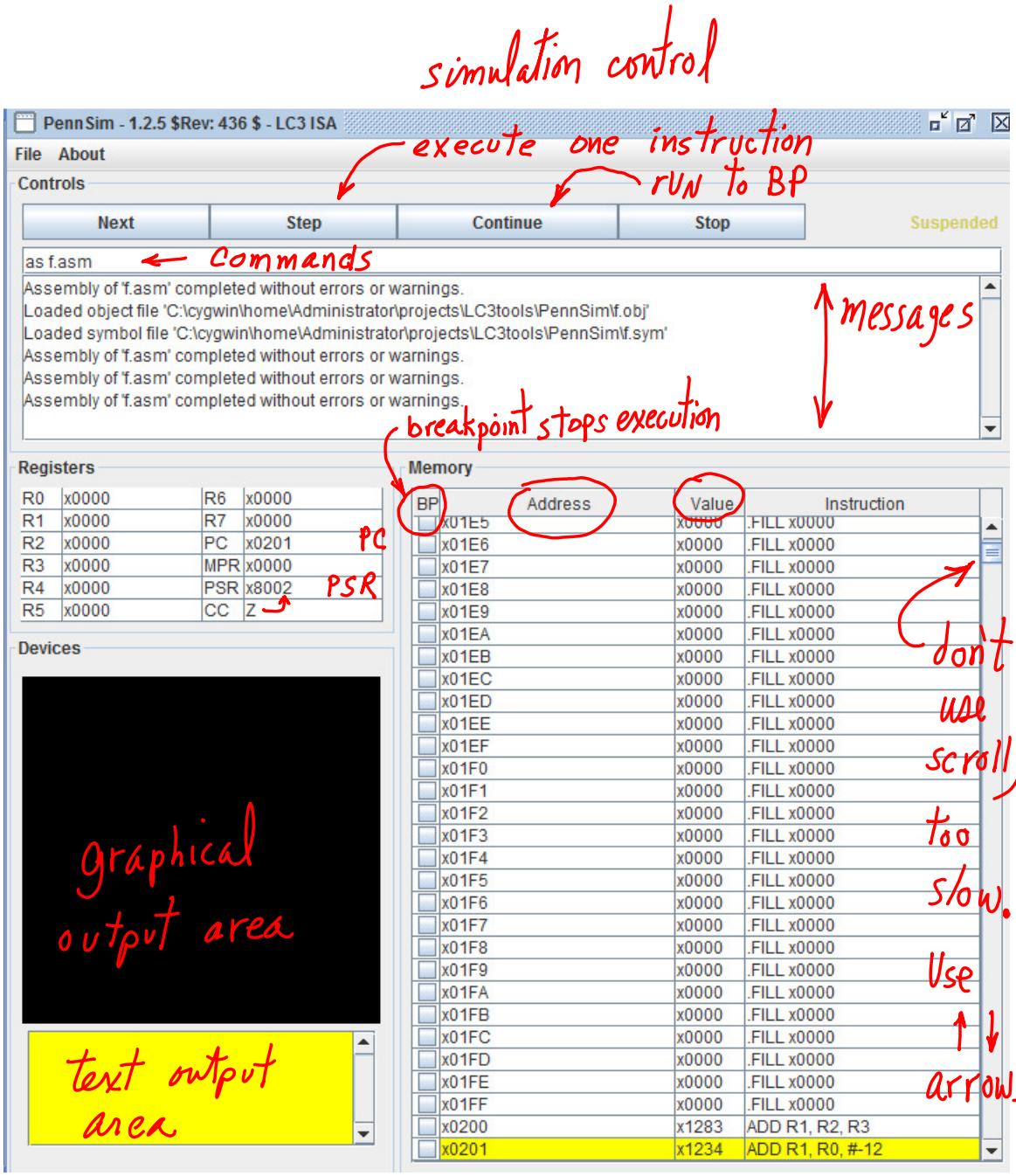
assembler

as foo.asm

loader

ld foo.obj

pure binary w/ symbol table

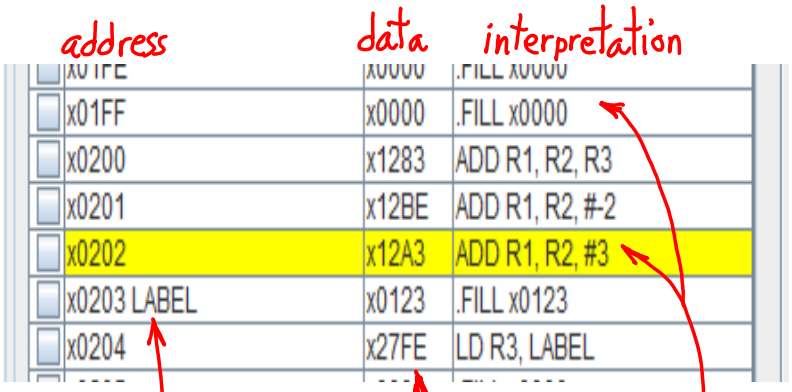


For viewing memory
use keyboard's up/down arrows.

```
;---------------
;-- LC3 assembly
;---------------
.ORIG x0200
    add r1, r2, r3  ;-- comment
    add r1, r2, #-2 ;-- decimal
    add r1, r2, x3  ;-- hex (no "-")

Label:              ;-- a named address.
    .FILL x0123     ;-- content of location

    ld r3, Label    ;-- cannot use actual
                    ;-- offsets, must use
                    ;-- address's name.
.END
```



PennSim tip
--- Run PennSim in the same directory where your .asm source code is.
Saves the trouble of typing path names into the command window.

Inputs to PennSim:

```
-- f.asm,        assembly source code, e.g., "as f.asm",
                 produces f.obj and f.sym.

-- f.obj,        load object, e.g.,          "ld f.obj",
                 contains LC3 machine code, loads into simulated memory.

-- f.sym,        assembler's symbol table,
                 read when "f.obj" loaded.

-- keys.txt,     simulated keyboard input, e.g., "input keys.txt",
                 read during execution of f.obj.

-- cmds.txt,     file containg PennSim commands, e.g., "script cmds.txt"),
                 runs multiple commands.
```

---------- Simulated LC3 Output ----------

Most computer displays can be set to operate in more than one "mode". Setting the display controller to "text-mode" provides an interface whereby one sends ASCII codes to the display controller, and it turns pixels on and off to display characters. Most computers boot up in text-mode, then switch to graphics-mode. Graphics-mode provides a way for the programmer to control each pixel by writing into a specific memory area, the Video Random Access Memory (VRAM).

PennSim's LC3 has two displays, instead of having one controller which switches modes.

(1.) Text-mode Display (see "Devices" GUI area, lower window)

---- Controlled by DSR and DDR device registers (memory mapped to addresses xFE04 and xFE06). Writing ASCII data into the DDR causes a character to be displayed in the text-mode display window. The controller keeps track of where the next character should appear (the "cursor position"), and sets the pixel's colors to form the character's pattern on the display.

(2.) Graphics-mode Display (see "Devices" GUI area, upper window)

---- Controlled by memory-mapped VRAM, one memory word per pixel (at addresses xC000-XFDFF).

In graphics-mode, one controls each pixel individually by writing into the VRAM memory area. Each word in the VRAM controls a different pixel, starting at the pixel at the upper left corner of the screen. The bits of a word for a single pixel are divided up into three color bit-fields: Red, Green, Blue (RGB). Each 5-bit color bit-field controls the intensity of that color for that pixel. Bits [14:10] are for red, bits [9:5] are for green, and bits [4:0] are for blue.

Quick Guide

========== Running ==========

--- double click PennSim.jar or "java -jar PennSim.jar"
NB--Run in source code (.asm) directory

========== PennSim GUI Usage ==========

--- Load object file: File.Open foo.obj

--- Execute program using buttons:
Step (one instruction)
Continue (until breakpoint, or Stop)
Next (until next RET)
Stop (suspend execution simulation)

--- Set Break point: select check box of memory location

--- Scroll through memory:
select memory cell, use up/down arrows
use scroll bar (very fast, hard to control)
use scroll bar arrow (very, very slow)

--- Change register/memory value: double-click to select value and edit

========== PennSim Commandline Usage ==========
==== (see the commandline input window below buttons) ==========
==========

--- Assemble a source code file:

```
as foo.asm
```

--- Load the resulting machine code to memory:

```
ld foo.obj
```

--- Specify simulated keyboard input:

```
input keystrokes.txt
```

--- Execute commands from file:

```
script foo.txt
```

--- Reset simulator to initial state:

```
reset
```

--- See help:

```
h
```

--- Set breakpoint at memory location:

```
b x0200
```

--- Dump memory to file:

```
d [-option]  x0200 x020F  foo.txt
   -readmemh (dump in verilog format)
```

--- Execute (next, step, continue, stop):

```
n, s, c, stop
```

--- Write to memory (address, value):

```
set x02FF x1234
set Label x1234 (write memory at label
```

--- Write to register (address, value):

```
set N           (set CC = N)
set R1 x1234    (set register's value)
```

--- Display register or memory content:

```
p               (show all reg. values)
l               (show memory at PC)
l  addr1 addr2  (show memory range)
l  addr1        (show starting at addr1)
l  label        (show starting at label)
```

--- Shutdown PennSim:

```
quit
```