LC3 Visualization User Guide

Revised 07 June, 2024; for LC3_Visualization v1p0

Contents

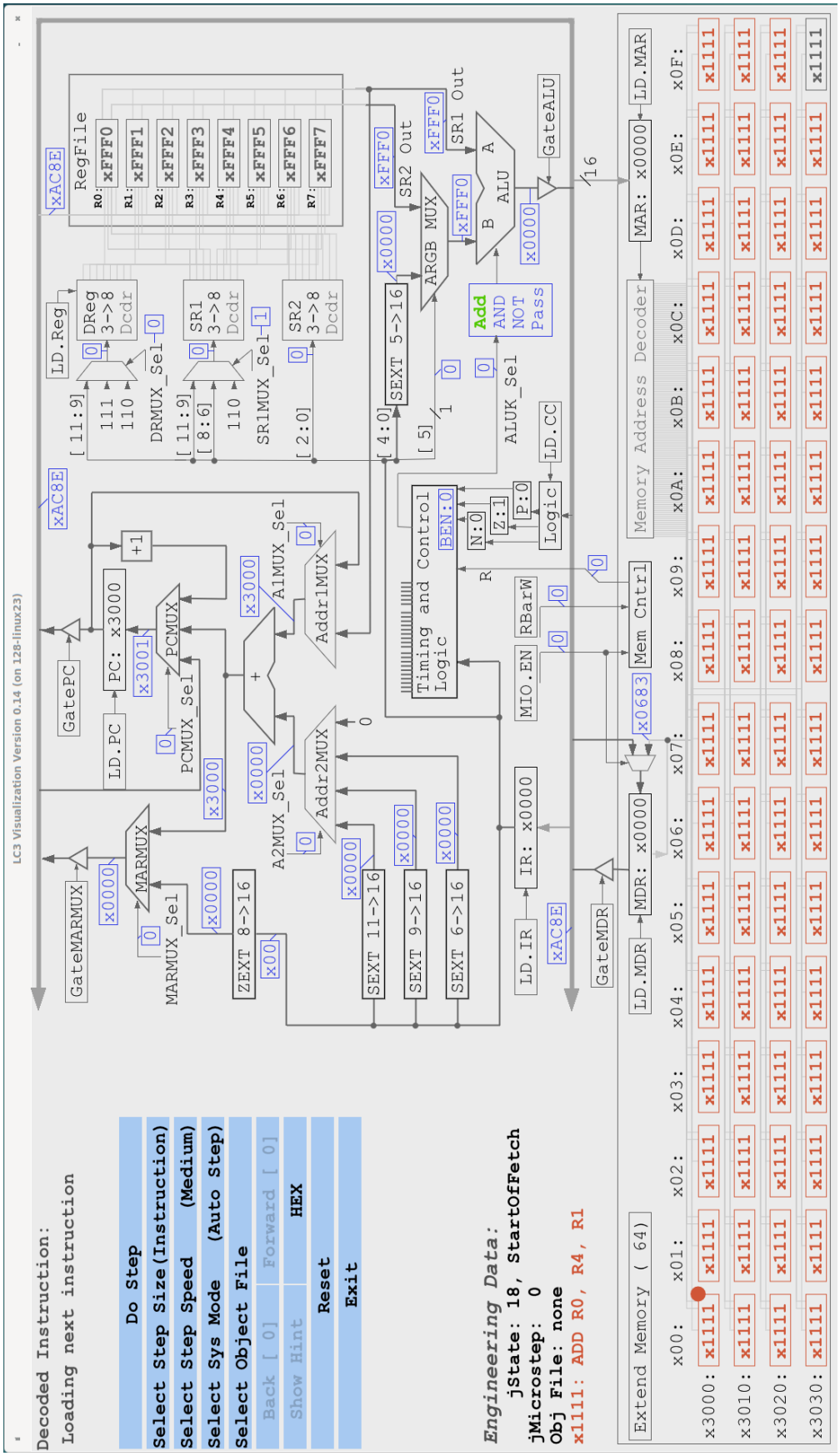


Figure 1: LC3_Visualization, upon launch.

1 Introduction

The LC3_Visualization is a tool for demonstrating and understanding program execution in the LC-3 computer. The LC-3 computer has been developed by Y. Patt and S. Patel for their textbook *Introduction to Computing Systems: From Bits & Gates to C/C++ & Beyond*. The LC3_Visualization layout is a block diagram of the LC-3, and is seen in figure 1. The LC3_Visualization :

- Is dynamic, buses, multiplexers, decoders and other elements are animated, switching to bright green when active
  - All elements are always present, and are gray when inactive;
  - Figure 1 shows the LC3_Visualization at a moment where the Program Counter (upper center) is being exerted onto the Main Bus (large ⊃ shaped line at perimeter);
- Is designed with the philosophy "Nothing is Hidden;" The LC-3 is simple enough that all elements can be visualized at all times;
  - Specifically, the memory region of figure 1 is all of the memory available;
    (Nothing is hidden; There are no other views !)
- Supports a variety of features that are useful for both lecture demonstrations and student assignments, including :
  - Capability to load .obj files produced by lc3tools;
  - Auto-analysis of a loaded program, to identify memory words of program code, global variables and the stack; memory words are correspondingly color coded;
  - With 144 words of memory, students have implemented bubble sort, pattern matching and other exercises;
- Features also include :
  - Auto-step, with state-machine-based automatic instruction execution, or the ability for the student to play the role of the Timing and Control unit, and activate elements;
  - `Full-control` , in which the instructor (or student) can generate any control signal in the LC-3, to demonstrate the action buses, multiplexers, decoders and other elements;
  - The ability to back step over instructions, to follow their behavior in detail or correct a mis-step;
    - This is implemented by storing the state at the completion of each instruction;
    - The "Forward" button is activated when state is backed up, permitting the student or instructor to rock back and forth across an instruction, examining each effect of the instruction one-by-one;
    - Mis-steps can arise in student-control .
- And the LC3_Visualization includes additional features supporting instruction, described below.

2 Quick-start guide

2.1 Quick-start guide, quick introduction

The LC3_Visualization runs on Windows or Linux.

- At this time, the LC3_Visualization has no notion of path, the executable and object file must be in the same directory.

When the LC3_Visualization launches, it appears as seen in figure 1. Figure 2 has labels added. The register file and memory are initialized with values to help identify the elements and visualize changes to data.
In the configuration of figures 1 and 2, the LC3_Visualization can be used to demonstrate features such as control signals, and multiplexer and decoder operation. For many things, however, the LC3_Visualization is most useful when a program is loaded. LC3_Visualization currently loads programs assembled by `lc3tools`, implemented by Chirag Sakhuja at University of Texas.

The LC3_Visualization controls are illustrated in figure 3(a). The control "`Select Object File`" (5^{th} down) opens the object file loading dialog, seen in figure 3(b). A loaded program is seen the memory region of figure 4.

- Analysis of the loaded program and colors used to paint memory words according to their use as program, data, stack, etc. are described in section 5.1 (see colors in figure 4, red is program memory).

- Execution in Auto-step is described here. Execution in Practice, Reporting and Full Control modes are described in section 5.2.

2.2 Quick-start guide, step-by-step

This quick-start guide steps through loading an example program and executing instructions. Several features are exercised.

1. Launch LC3_Visualization. The open application will appear as in figure 1.

2. Load an object file

   (a) Open the "`Select Object File`" dialog; which will open a selection dialog like that seen in figure 3(b).

   (b) Select an object file ("`All_LC3_Instructions.obj`" for this quick-start guide).

   (c) Select "`Close`", to close the dialog and load the file.

   - LC3_Visualization will load and analyze the object file, and paint the memory words according function.

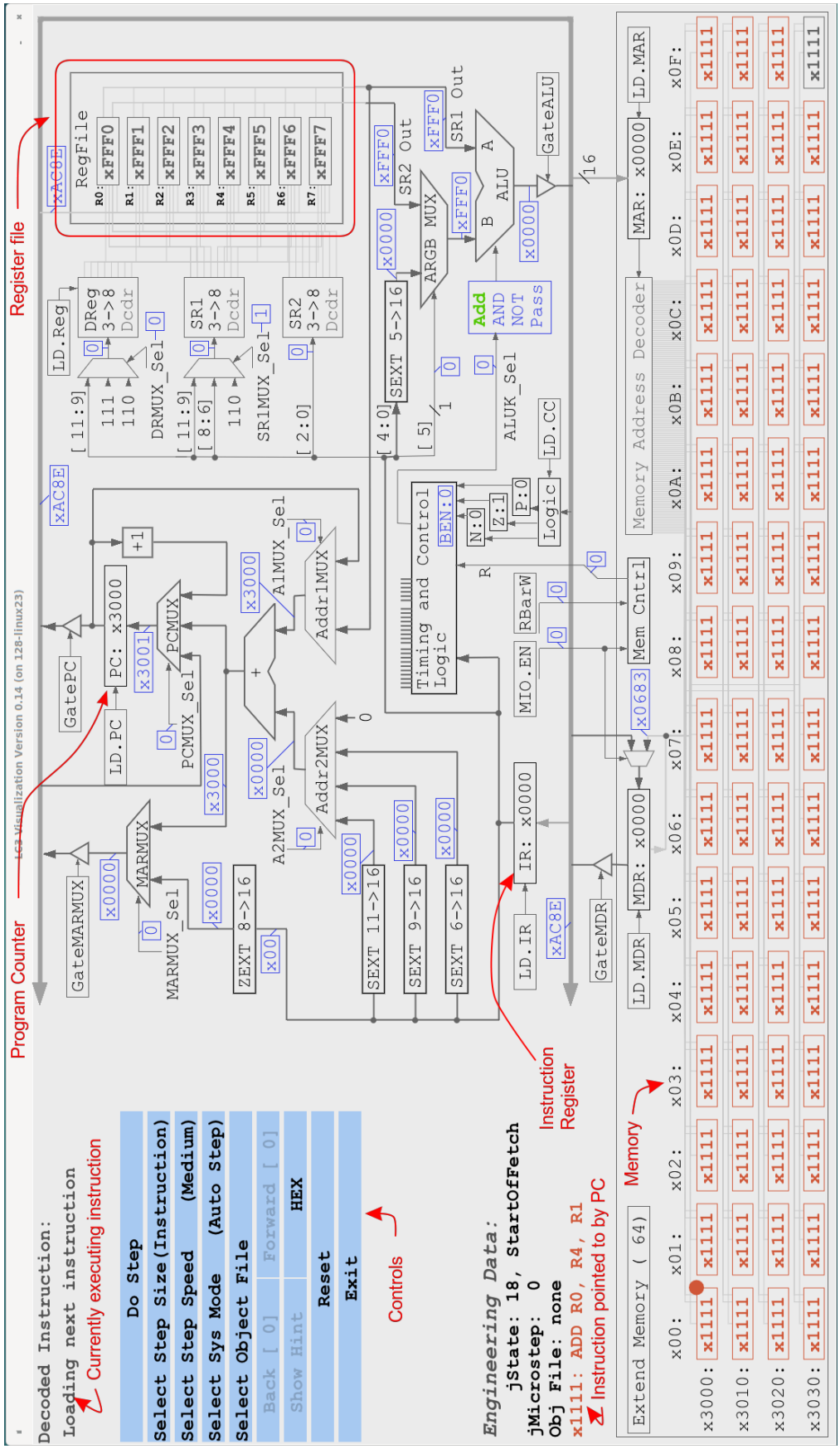
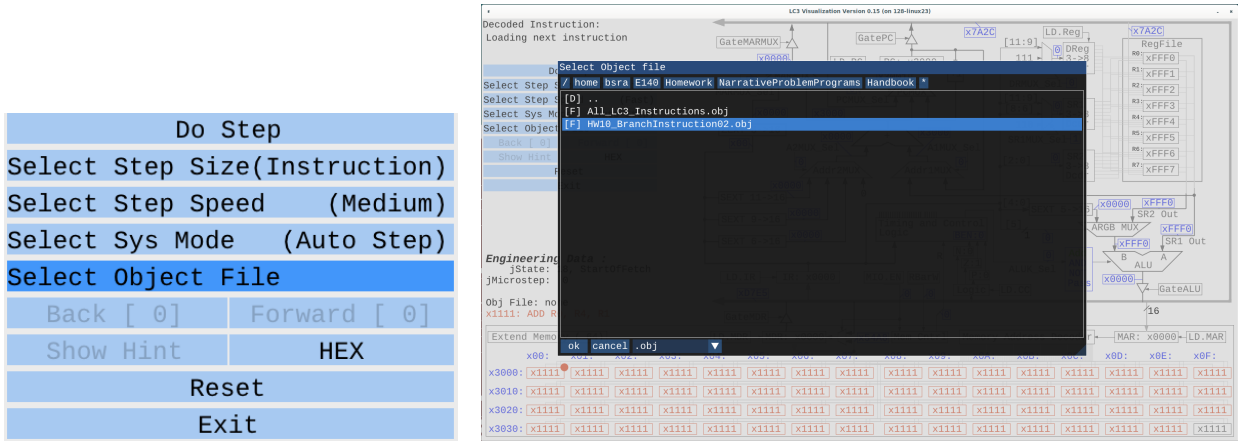


Figure 2: LC3_Visualization upon launching, with labels.



(a) Expanded view of LC3_ Visualization Controls.

(b) Expanded view with "Select Object File" dialog open.

Figure 3: LC3_Visualization (a) Eleven Control buttons (three are grayed out); (b) Expanded view of the "`Select Object File`" dialog. At this time, LC3_Visualization is not path-aware, and object files must be in the same directory as the LC3_Visualization executable.

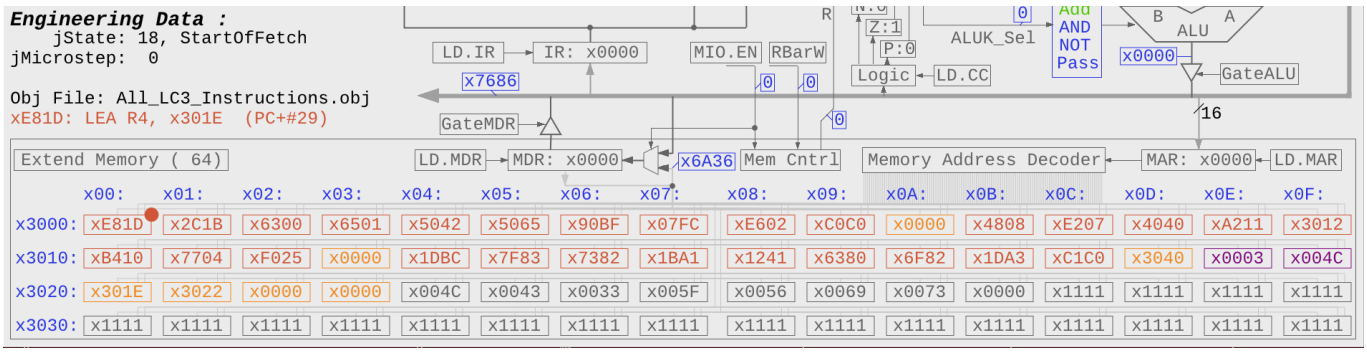


Figure 4: LC3_Visualization with a program loaded into memory, starting at address x3000. LC3_Visualization analyzes the loaded program, and paints instructions red and data orange.

3. Execute an instruction, select "`Do Step`" (the top button in figure 3(a)).
   - The first instruction is `LEA R4, x3021` (setting up the global variables pointer).
     - Executing the `LEA R4, x3021` instruction involves five states and 13 microsteps, and will take about 16 seconds at "`Medium`" Step Speed.
     - Note that the "`Back`" button has become active.
   - The LC3_Visualization supports several different automatic step sizes, from one microstep to continuous running.
4. Click the "`Select Step Size`" button. The Step Size dialog will open, as seen in figure 5. Select "`Run One Microstep`" and close the dialog.

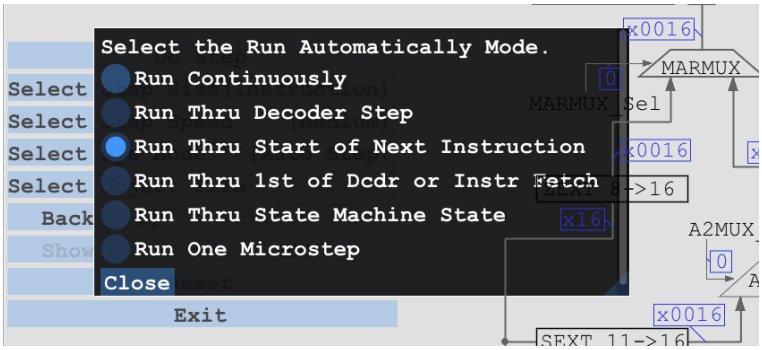


Figure 5: Select Step Size dialog. See table 15 and discussion for explanation of Auto-Step step sizes.

5. Click the "`Select Step Speed`" button once, to change the step speed to "`Fast`". (This will accelerate execution of the microsteps. Note that the "`Do Step`" button is recast to "`Stop Step`" during automatic execution.)
6. Click the "`Do Step`" button nine times, observing the LC3_Visualization at each step. The LC3_Visualization walks through the actions needed to bring the instruction value, x2C1E, into the Instruction Register, as seen in figure 6.
7. Hit the space bar 10 times, observing the LC3_Visualization at each step. The space bar is a keyboard shortcut for clicking "`Do Step`".
   - Ten microsteps will complete the `LD R6, x3020` instruction, modifying `R6` and bring the system to the start of the next instruction fetch cycle.
   - The Back or Reset buttons can be used to step back or restart execution.
8. Hold down the shift key, place the cursor in the Instruction Register, and left-click, this will increment the Instruction Register contents.
   - Right-clicking with the shift-key held will decrement the contents.
   - The range of methods to set general purpose registers, operating registers and memory locations are described in section 3.3.3.

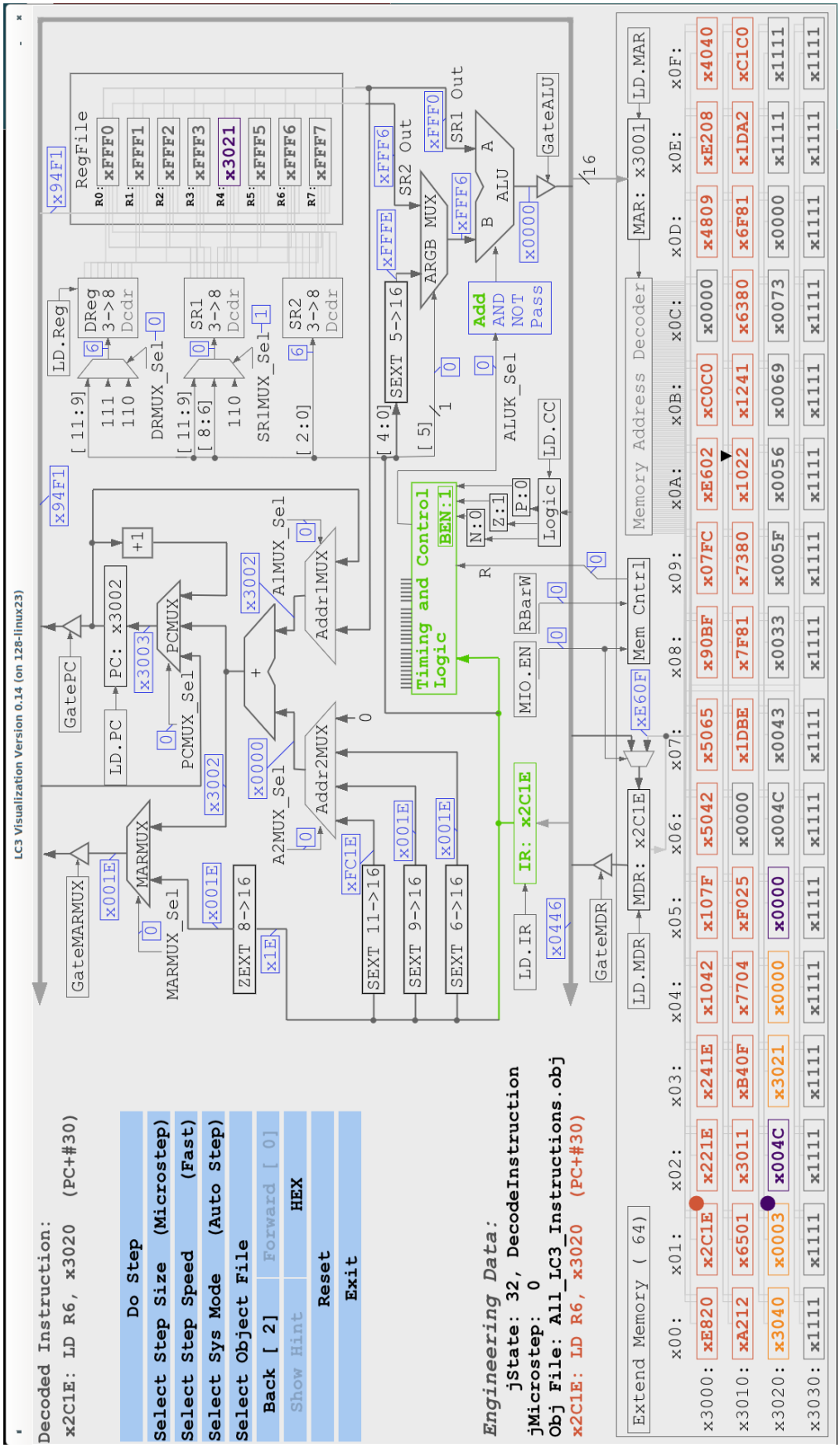
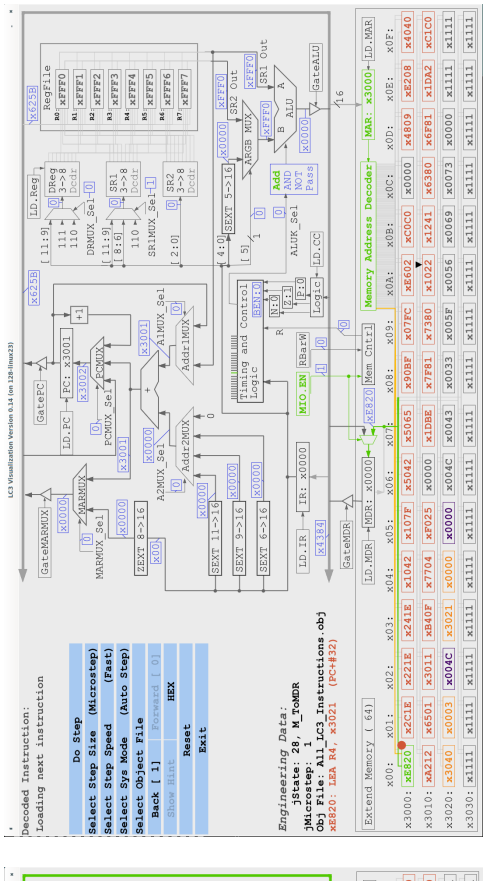
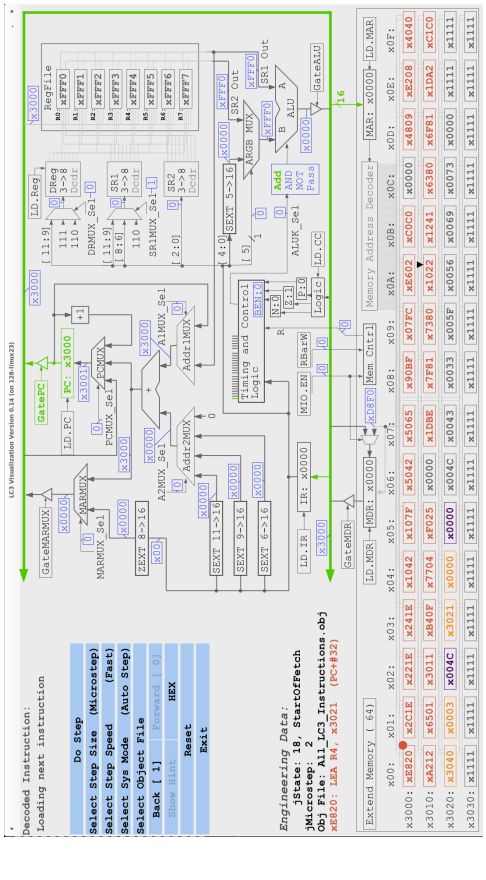


Figure 6: The second instruction, `LD R6, x3020` advanced to state 32, microstep 0. In this state, the instruction is decoded.

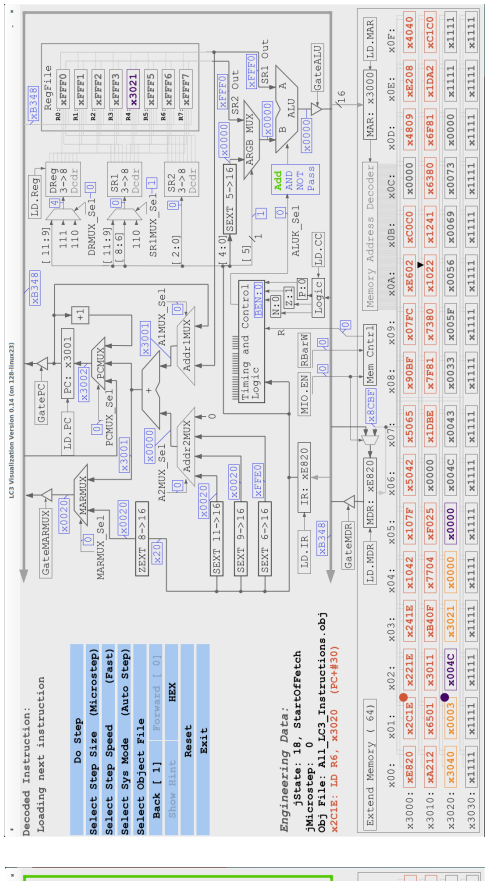


(a) During fetch, instruction address on main bus.



(b) During fetch, instruction is fetched from memory.

(c) Effective address computed by the address logic and moved to GP register R4.



(d) Effective address has been stored in R4, start of next fetch.

Figure 7: Three stages of executing the `LEA R4, x3021` instruction (a, b, c), and at start of the next instruction (d).

9. Click the "Back" button once, bringing the system state back to the start of the LD R6, x3020 instruction.

   - This brings the state back to the start of the previous instruction.
   - Note that the "Forward" button becomes active. Since we have gone back an instruction, it is possible to go forward.

10. Click the "Forward" and then the "Back" button several times, rocking back and forth across the LD R6, x3020 instruction.

    - Observe changes in the Program Counter, Memory Address Register (MAR, lower-right), Memory Data Register (MDR), Instruction Register, register R6 and Condition Codes (N, Z and P, central).

      The ability to rock back-and-forth across an instruction gives the ability to point out and discuss its side effects one at a time.

    - If the "Back" button is clicked in the middle of the current instruction, state is restored to the start of the current instruction. The forward button does not become active, because the state at the start of the next instruction has not yet been seen.

11. Click "Select Step Size", and select "Run Continuously". Click "Close".

    - The "Do Step" button will change to read "Run Continuously", indicating the new action of the button.

12. Left click memory location x301A, which contains the value x1022. (Note the address of each memory location is the sum of the row and column values.)

    - A black triangle will appear. Indicating that a breakpoint has been set. The breakpoint is toggled by repeatedly left-clicking the memory location.

13. Click "Run Continuously", and run up to the breakpoint. The LC3_Visualization will appear as seen in figure 8.

    - While running, the top button changes to "Stop Step", indicating its action. Automatic execution can be toggled at any point.
    - Subroutine MySubroutine01() starts at instruction x3017. At instruction x301A two bytes have been allocated on the stack and R7 and R1 stored.
      - The value pointed-to by the stack pointer is marked by a blue bullet. Values on the stack are painted blue.
      - The red bullet indicates the memory location currently pointed-to by the program counter, and the purple bullet indicates the target of the global variables pointer (R4).

14. Click the button indicating "HEX". Register contents can be reported HEX, Signed decimal, Unsigned decimal, as a disassembled instruction, or in ASCII. Repeatedly click the button to cycle between presentation formats. See section 3.2.4

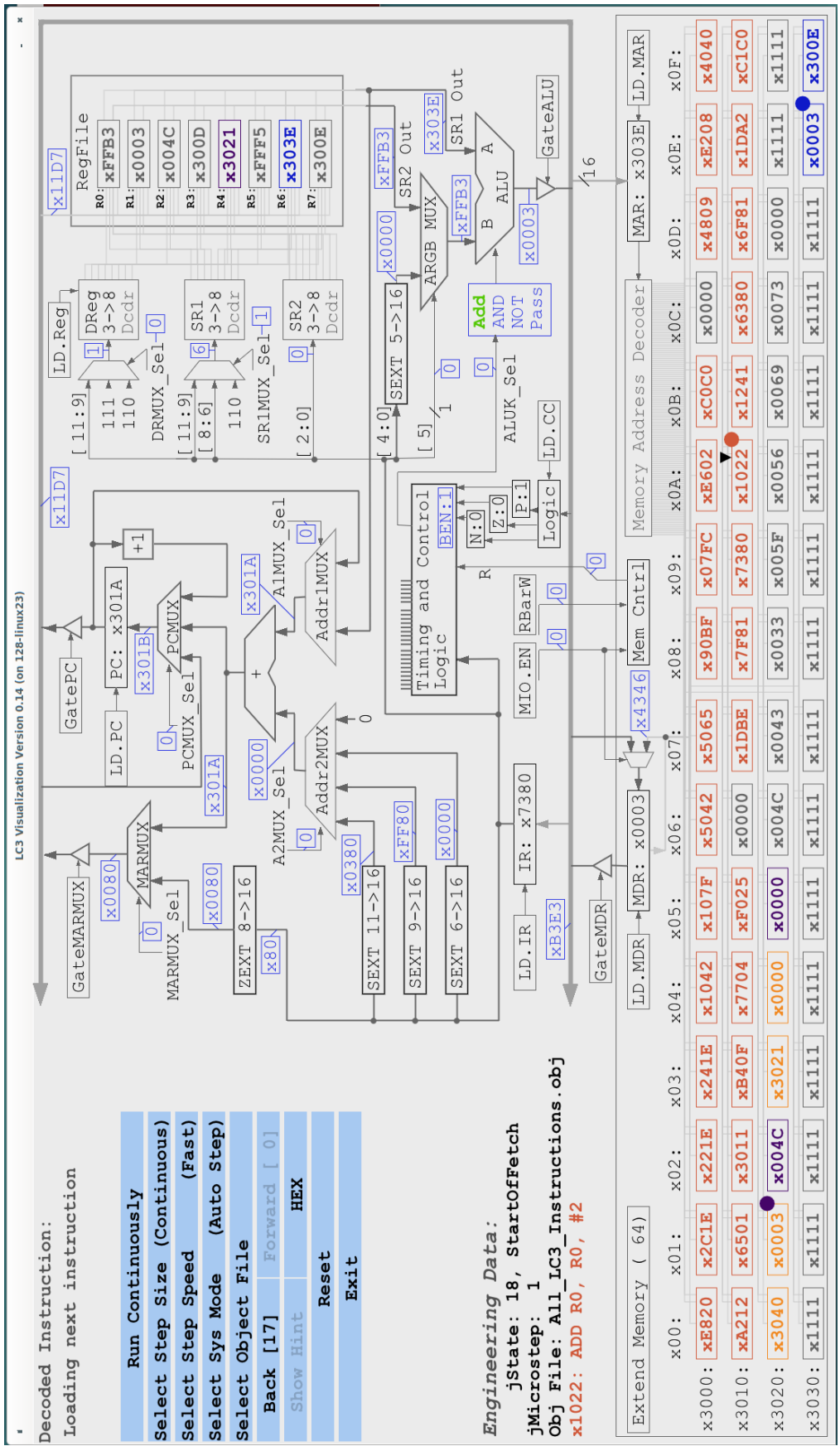


Figure 8: LC3_Visualization at the start of fetch at address x301A.

2.3 Additional discussion of the quick-start execution

- Execution is, of course, dynamic and does not lend itself to being capture in a static figure. Nonetheless, figure 7 shows three stages of executing the `LEA` instruction, and LC3_Visualization ready to start the next instruction.

  – The LC3_Visualization implements the state machine of *Introduction to Computing Systems*, with additional decomposition of states down into microsteps. Figure 7(a) shows the system in state 18 (start of instruction fetch), microstep 2 (in this microstep the address in the Program Counter is exerted onto the Main Bus and the Memory Address Register (MAR, lower right) is receiving a Load control signal, causing it to capture the address value of x3000).

  – Figure 7(b) shows state 28, microstep 1. The action of state 28 is described as `MDR <- M` (the data in the memory location is transferred to the Memory Data Register).

3 Operation of the LC3_Visualization

Three main use cases for the LC3_Visualization are :

- As a Visual CPU Simulator, for use in student exercises focused on computer organization;
- As a Program Animation Tool, for use in student exercises focused on programming and execution model; and
- As a lecture demonstration tool.

A subsection follows for each use case. Features are described in the subsection where the feature is most relevant.

3.1 Overview of the LC3_Visualization

The main window of the LC3_Visualization is seen in figure 1, above. The main characteristics are:

- Nothing is hidden, nothing moves - the presentation does not have perspectives
  - All processor state is shown,
  - All memory locations are shown.
- The layout follows that found in Patt and Patel [1], drawn with all registers and other CPU elements present; and without keyboard or display I/O. For exercises, the stored value editing (section 3.3.3) stands in for keyboard data entry, and all registers and memory locations are continuously displayed.
- The simulation operates with 64, 144 or 256 words of memory.
  - Sixty-four words (cf. figure 1) allows the same 28-pixel font to be used for memory, as is used for other CPU elements. Twenty-eight pixel font (64 words) is suitable for projection and lecture presentation[1].
  - Memory is extended to 144 or 256 words by clicking the “Extend Memory” button in the upper-left of the memory block. Changing memory size also initiates a reset.

1. Features in the LC3_Visualization have been sized to be visible from the far corners of a lecture hall with an HD-resolution projector and lecture-hall screen.

3.2 Using LC3_Visualization as a Visual CPU Simulator, for use in student exercises focused on computer organization

Many computer simulation systems have been developed to support university courses in computer organization. Nikolic and coauthors [2] provide a detailed review of 28 such systems. Relative to the tools reviewed, the LC3_Visualizationis intended to a tool used with and by freshman students who have no prior knowledge of digital logic or computer organization. The LC3_Visualization lacks both advanced CPU features and the ability to modify the computer organization. However, LC3_Visualization incorporates the six features described below to visually communicate and the basic elements of computers the processes of instruction execution.

3.2.1 Student operation of CPU control signals, and Reporting and Practice modes

In Reporting and Practice modes, students play the role of the CPU Timing and Control Unit by activating Gate and Load control signals to execute instructions.

3.2.1.1 Reporting mode In Reporting Mode students control instruction execution by activating Gate, Load and Memory Control signals with button clicks. The available controls are listed in table 1. A step of instruction execution is seen in figure 9, where the `GatePC` and `LD.MAR` controls are activated.

Gate	Load	Memory Control
GatePC	LD.MAR	MIO.EN
GateMDR	LD.MDR	RBarW
GateALU	LD.IR	
GateMARMUX	LD.PC	
	LD.Reg	

Table 1: LC-3 processor control signals used in Reporting and Practice modes.

In Reporting Mode :

- A gradable report is generated, that the student can turn in.
- The purpose to the exercise is for the student to play the role of the microprocessor Timing and Control Logic, and thereby gain a greater appreciation of microprocessor execution as a specific sequence of steps.
- The LC-3 State Machine advances according to the Transition Rules described in section 5.3. A correct report is generated only by the correct sequence of clicks.
- The LC3_Visualization was brought to the configuration of figure 9 by clicking on the `GatePC` and `LD.MAR` buttons (in that order !)

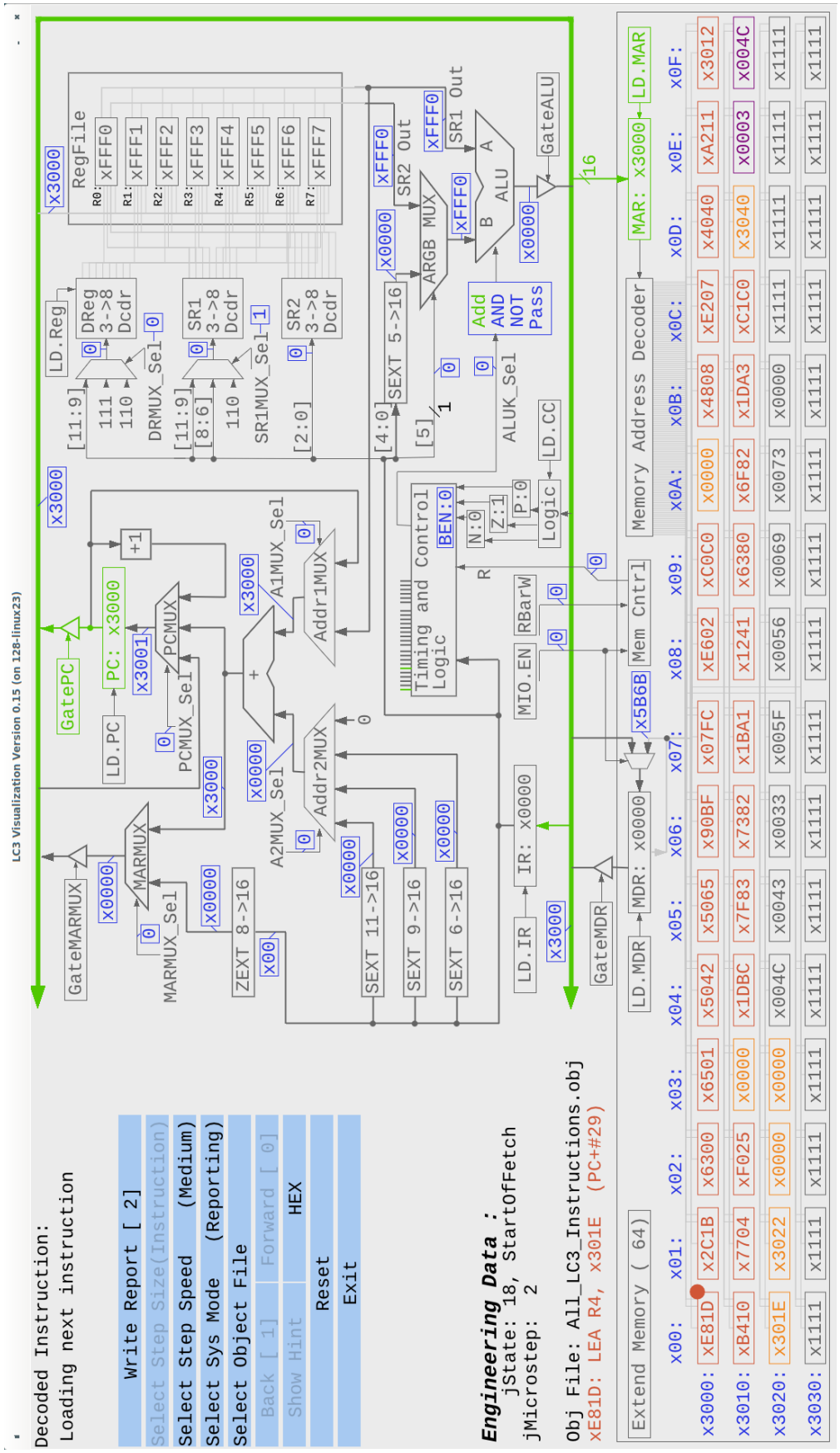


Figure 9: LC3_Visualization in Reporting mode with `GatePC` and `LD.MAR` controls active.

3.2.1.2 Quick-start, demonstration of reporting mode As a demonstration of Reporting Mode,

1. Launch LC3_Visualization and load object file `All_LC3_Instructions.obj`.
2. Click button "Select Sys Mode" twice, to switch into Reporting Mode.
3. Click in order the buttons listed in table 2.

Step	Button	Button location	Action
1.	GatePC	Upper-center	Gate the contents of the Program Counter onto the Main Bus
2.	LD.MAR	Lower-right	Load the Memory Address Register from the Main Bus
3.	LD.PC	Upper-center	Load the Program Counter from the PC Multiplexer (MUX)
4.	MIO.EN	Middle-center	Activate the Memory System
5.	LD.MDR	Middle-center	Load the selected memory data into the Memory Data Register
6.	GateMDR	Middle-center	Gate the contents of the MDR onto the Main Bus
7.	LD.IR	Middle-center	Load the Instruction Register from the Main Bus
8.	GateMARMUX	Upper-left	Gate the Memory Address MUX onto the Main Bus
9.	LD.Reg	Upper-right	Load register R4 from the Main Bus.

Table 2: Student actions to execute `LEA R4, x3021`.

- Several things to notice during steps 1 ... 9:
  - Various elements are activated as the steps are taken. Notably, when MIO.EN is activated in step 4 (State 28, Microstep 0). In this step, the Memory Address Decoder is activated, and a read select line to memory location x3000 is activated.

    These activations are setup and controlled through the "Microstep Actions" of figure 15.
  - Some actions are automatic, even in Reporting Mode. For example, the `GatePC` signal releases automatically, after `LD.MAR` is activated. It was found to be tedious to both activate and release each Gate control signal.

    Likewise, the MUX-select and ALU-control signals are automatically generated. For example, in step 7, when the instruction is loaded into the Instruction Register, multiplexer control signals for the instruction are automatically set to the values seen in table 3.

    Activations and automatic actions are controlled through a configuration file, and so what is presently automatic can be easily reconfigured to be manual (or vice-versa).

	MUX Select Signal Value
Address 1 MUX	0
Address 2 MUX	2
MAR MUX	1

Table 3: Multiplexer select signal values in State 14, to execute `LEA R4, x3021`.

- In Reporting Mode the “Do Step” button is relabeled “Write Report”. This button writes a text-file report, indicating which buttons were pressed and in what order, along with CPU state. The report can be the gradable deliverable.
- Note that in Reporting Mode – as opposed to Practice Mode – all buttons in columns “Gate”, “Load” and “Memory Control” in table 9 are active at all times and will perform their corresponding functions.
  - A bus conflict is created by activating two Gates simultaneously.
  - A load activated while there is no valid data on the Main Bus will load random data.

  The student is rewarded with correct execution, only by correct execution.

3.2.1.3 Practice Mode Practice Mode is implemented to provide the student with a way to prepare for a Reporting Mode exercise. In Practice Mode:

- Only the currently-required control signal is active, all other buttons will turn yellow and have no effect if clicked. This is to permit the student to explore various possible signals, without corrupting the CPU state.
- Hints are available at each microstep. A sample hint is seen in figure 10.

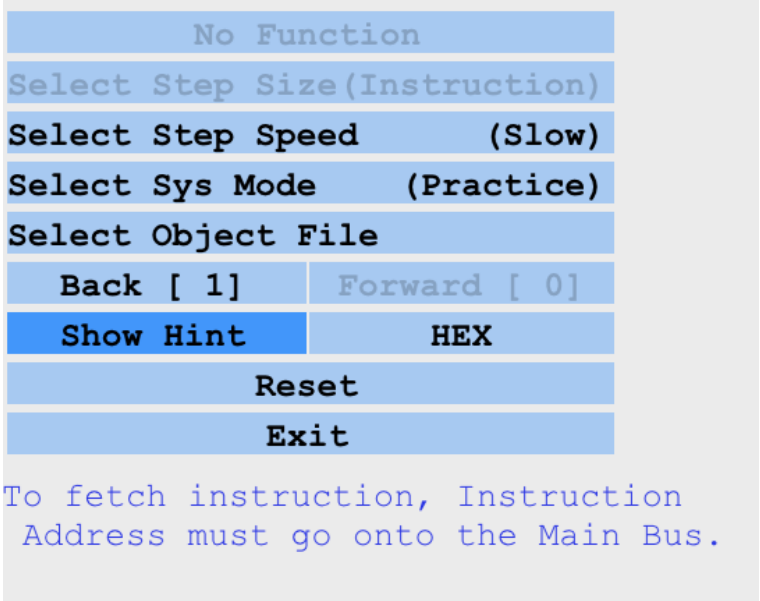


Figure 10: A hint as to the next control signal needed, provided in Practice Mode.

For assignments exploring the LC-3 processor control signals and a processor as a state machine, the combination of Practice and Reporting Modes gives a way for the student to prepare and then demonstrate execution of an instruction.

3.2.2 Visualization of CPU activity

The LC3_Visualization main window is animated by coloring active elements at each CPU state, as illustrated in the panels of figure 7, above. The visualization addresses a fundamental challenge of static images in texts and lecture slides : CPU state and action are dynamics and a poorly reflected in static textbook and lecture (and User Guide !) figures[3].

Figure 9, above, captures a typical moment during CPU execution, with a Gate, the Main Bus and the Memory Address Register active. Other moments, with other elements active, are captured in figures 6 and 7.

3.2.3 Auto-step mode features useful for CPU visualization

Auto-stepping through program execution is a main element of program visualization, and is more fully described in the next section. Here we mention several features supportive of visual CPU simulation.

1. Auto-step mode, speed control

   Step speed is set by the third button in the LC3_Visualization Controls area. When set to "Slow", the visualization is set to 2 seconds per microstep, slow enough to visualize each action of the executing instruction.

2. Auto-step mode, single stepping at several different execution scales

   The user can select an auto-step step size ranging from "Run Continuously" to "Run One Microstep," (cf. figure 5).

3. Instruction roll-back

   When practicing instruction execution, instruction roll-back can be used to restart an instruction that has gone astray.

3.2.4 Display modes

Data in computers is interpreted in multiple ways. Correspondingly, register and memory data in the LC3_Visualization can be displayed in several formats :

- HEX,
- Signed integer,
- Unsigned integer,
- Disassembled instruction, and
- ASCII
- Address

The button indicating HEX in figure 10(a) controls the selection. Clicking will cycle the display mode through the first five options. Clicking five times will bring the display mode back to HEX. Figure 11 illustrates the display mode set to "Instruction". Only words identified as instructions are disassembled (XXX section 5.1 describes the automatic program analysis).

The "Address" display mode is accessed by typing the hot key 'a'.

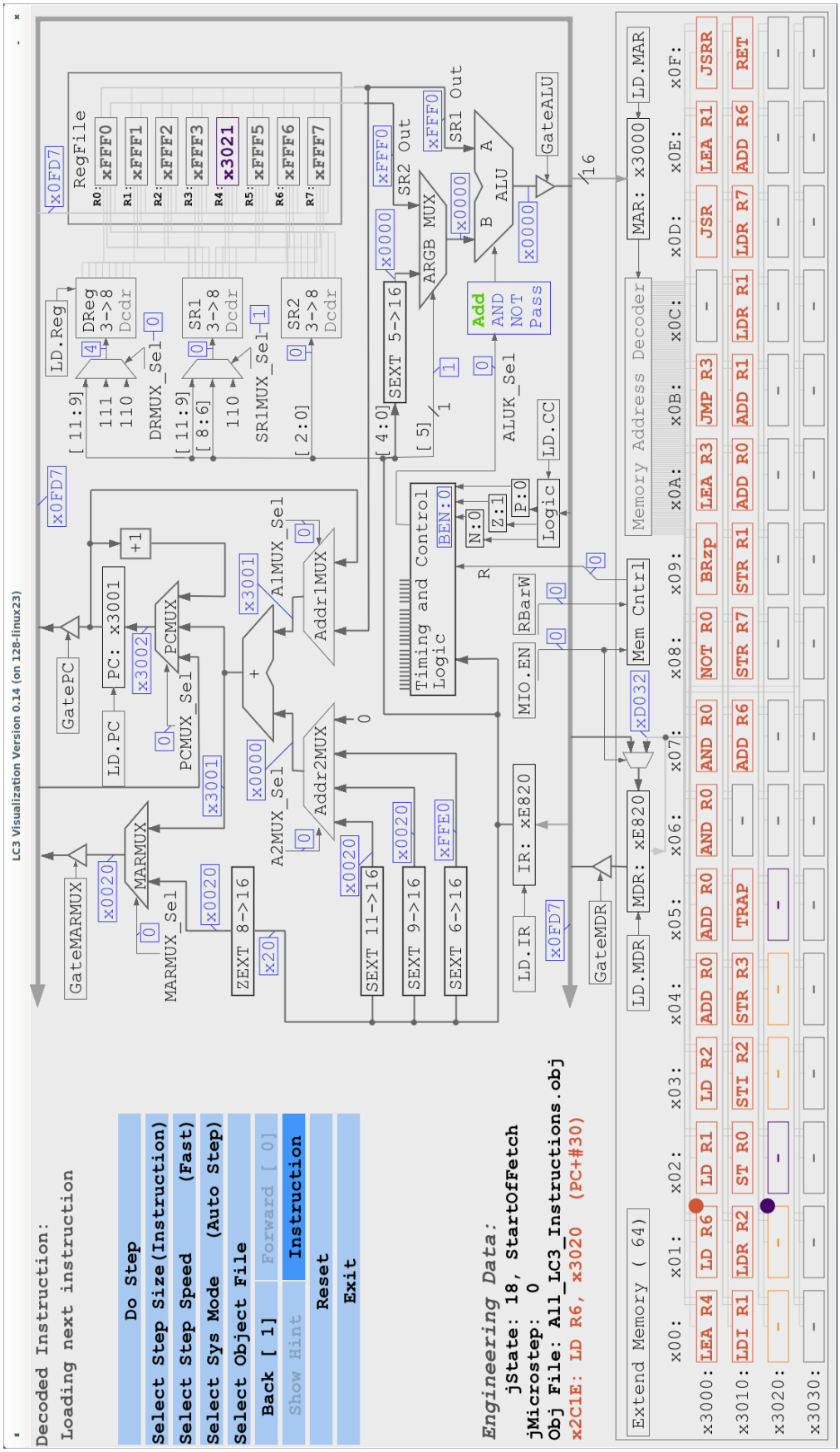


Figure 11: LC3_Visualization in a state comparable to figure 6, with the "Instruction" display mode selected.

3.3 Using LC3_Visualization as a Program Animation Tool, for use in programming exercises and visualization of the execution model

Program animation is distinct from visual CPU simulation, with less focus on data path and CPU elements, and a greater focus on data structures and program execution. Program animation tools for introductory programming courses often address challenges novice programmers have with program dynamics and fundamental programming concepts, including *i)* static perceptions of programming, *ii)* difficulties understanding the computer, *iii)* misconceptions about fundamental programming constructs, and *iv)* struggles with tracing and program state [4].

The LC3_Visualization assists novice students to overcome these conceptual challenges by rendering otherwise hidden aspects of computer and program execution; and doing so in a way that presupposes a minimum of prior computer knowledge.

3.3.1 LC3_Visualization as a debugging tool

The LC3_Visualization in Auto-step mode provides several functions of a conventional assembly-language debugger in a way that is intended to be accessible to novice students. Auto-step mode operates using Patt and Patel's state-machine model (see section 5.3). As demonstrated in the Quick-Start guide and seen in figure 5, and auto-stepping can run for step sizes ranging from one microstep to running continuously.

3.3.1.1 Breakpoints In the LC3_Visualization, a breakpoint :

- Is set or cleared by left-clicking on the memory cell where the breakpoint will be located;
- Is indicated by a black triangle, marking the memory location, as seen in figure 12;
- Halts execution at the start of instruction fetch for the marked instruction (State 18, Microstep 0). This is equivalent to running the previous instruction with Step Size set to "Run Thru Start of Next Instruction".

Breakpoints, of course, have many uses for program development. One use in teaching introductory computer organization is to direct the student to run execution to a specific instruction and then investigate the state or behavior of the computer at that point of program execution.

3.3.2 Visualizing stored data

The LC3_Visualization does not present a variables view, or, indeed, any specialized perspective. Rather, all registers and memory are visible at all times. To support locating memory locations by address, the "Address" display mode can be used.

3.3.3 Stored value editing: modifying register and memory data

Student programs often work with data that is loaded from the same object file as program code. Examples include student exercises in sorting, pattern matching and logical evaluation.

- LC3_Visualization does not currently support concurrently loading more than one object file.

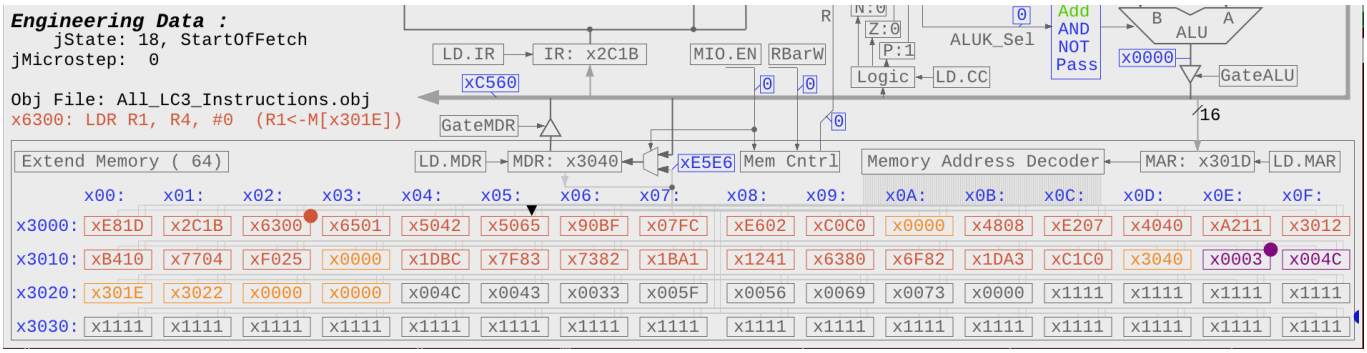


Figure 12: A breakpoint is set at address x3005. The breakpoint is shown by the black triangle on memory location x3005. The red, violet and blue circles mark memory locations pointed to by C-language execution-model pointers, described in section XXX.

To support testing and demonstration, it is desirable to have a straightforward way to modify data in memory.

3.3.3.1 Stored-value modification by mouse Mouse clicks on memory locations and registers are detected. When the mouse clicks are recorded with modifiers, the value of the register or memory location is modified. The effect of mouse clicks with several different combinations of modifiers are list in tables 4 - 6.

Modifier	Effect of left-click	Effect of right-click
Control-key held	Increment 1^{st} digit	Decrement 1^{st} digit
Control+Super-keys held	Increment 2^{nd} digit	Decrement 2^{nd} digit
Control+Alt-keys held	Increment 3^{rd} digit	Decrement 3^{rd} digit
Control+Super+Alt keys held	Increment 4^{th} digit	Decrement 4^{th} digit

Table 4: Incrementing and decrementing registers and memory locations by mouse clicks.

- To illustrate the effect of modified mouse clicks, suppose the initial value in a memory locations is x1234, and the display is in HEX, the value after a mouse click is illustrated in table 5.

Modifier	Effect of left-click	Effect of right-click
Control-key held	x1235	x1233
Control+Super-keys held	x1244	x1224
Control+Alt-keys held	x1334	x1134
Control+Super+Alt keys held	x2234	x0234

Table 5: The value in a memory location initially holding x1234 after the indicated modified mouse click, with the display mode in HEX or Disassembled Instruction mode.

- The effect of mouse clicks and modifiers depends on the display mode. If the display mode is an integer mode, the columns are weighted by powers of 10. The effects of mouse clicks on an initial value of #4321 are illustrated in table 6.

Modifier	Effect of left-click	Effect of right-click
Control-key held	4,322	4,320
Control+Super-keys held	4,331	4,311
Control+Alt-keys held	4,421	4,221
Control+Super+Alt keys held	5,321	3,321

Table 6: Integer values in a memory location initially holding #4,321 after the indicated modified mouse click, with the display mode in signed or unsigned integer mode.

Unmodified mouse click also effect memory locations and (for right clicks) registers, as seen in table 7.

Modifier	Effect of left-click	Effect of right-click
No modifier	Toggle breakpoint	Begin text value entry

Table 7: Function of left- and right-mouse clicks on registers and memory locations, with no modifier.

3.3.3.2 Stored-value modification by text value entry By right-clicking on a memory location or register, text value entry for that location is activated.

- During text value entry, the location and value color alternate between blue and red.
- Data is entered via the keyboard, only valid digits are recognized.
  - The interpretation of digits depends on the display mode. For example, negative numbers can be entered in "`Signed Integer`" display mode.
  - Nothing can be entered in "`Instruction`" display mode (text entry of instructions is not allowed).
- Text entry is terminated and the value is accepted by typing the Tab or Enter key.
- Text entry is terminated and the prior value is restored by typing Escape.

3.3.3.3 Restoring original values Clicking the "`Reset`" button will restore values from the object file, and erase all modified values.

3.3.4 Memory region labeling

The type C-language execution model includes memory regions for :

- Program memory
- Embedded data
- The user stack
- Subroutine, stack frame
- Global variables

The LC3_Visualization uses program analysis described in Appendix 5.1 to determine the region of each memory word. This information is used to color memory locations, according to table 8.

Color	Memory region	A pointer is dynamically indicated by a dot	Pointer indicated
Red	Program memory	✓	Program Counter
Orange	Data		
Purple	Global variables	✓	Global vars pointer, R4
Black	Stack Frame	✓	Frame pointer, R5
Blue	Stack	✓	Stack pointer, R6

Table 8: Memory region color assignments.

Figures 6, 8 and 12 illustrate the use of memory region labeling. We find that marking memory function and pointer targets helps students grasp the dynamic nature of program execution in several ways:

- The memory regions of the program execution model are made visual and explicit;
- The dots show the current value of the program counter, stack pointer, stack frame pointer and pointer to the global variables region.
- The dots move, visualizing progress of instruction execution and the dynamic allocation of stack space;
- The Stack Frame comes and goes as subroutines are activated and terminated;
- The Global Variables allocation is static.

3.3.5 Tools for exploring the computer state

3.3.5.1 Information boxes Information boxes are shown in blue, and are connected by leader line with a data or control connection. The value in the information box shows the current value on the bus or control connection. Information boxes are shown on :

- The buses,
  - Inputs to and output from the ALU,
  - Control inputs to and outputs of the multiplexers,
  - Outputs of the bit extenders,

- The Branch Enable (BEN) bit, and
- Other locations.

An important idea conveyed by the information boxes is that data is always present at many points throughout the CPU. Also, in Full Control mode (cf. section 3.4.1), the information boxes show the effect of various control actions, such as changing MUX selection bits, or the ALU operation.

- The buses float when no source is gated onto them, indicated by randomly changing information-box data. In figure 2, above, the value of xAC8E shown on the Main Bus, and x0683 shown on the internal memory bus does not correspond to any register or memory value. The Main Bus is shown active in figures 7(a) and 9, above. The Memory bus is shown active in figure 7(b).

3.3.5.2 Back and Forward actions The ability to step backwards, to the start of the current executing instruction or to the start of previous instructions, enables the student to the multiple changes of state associated with instructions. When stepping back a full instruction, the forward button becomes active.

3.4 Using LC3_Visualization as a lecture demonstration tool

3.4.1 Full-control Mode

Full-control mode is implemented to support lecture demonstrations of the functions of multiplexers, decoders, extenders, the buses and other elements of the LC-3. In full-control mode the state machine is inactive, there is no action in the LC3_Visualization, except for actions generated by activating the Processor Control Signals.

An example is seen in figure 13. In figure 13(a) the Address 2 MUX Select is set to 0, and in figure 13(b) it is set to 2. The change of control changes the configuration of activated components, as well as changing the signals at many points. The large collection of examples that can be produced supports explaining the behaviors of the elements and data pathways making up the processor.

For controls with more than two states, such as the `Address2MUX_Select`, which has four possible values, the controls cycle through their possible values.

In Full-control Mode it is possible to manually execute instructions. While this is a formidable task – Reporting and Practice modes are partially automated to give a more realistic student workload – it is an exercise that may be worth seeing or doing once. Said another way, any action of a CPU element within any state of any instruction can be demonstrated – not counting interrupts and privilege that are not yet implemented (cf. section 4, item 4).

The complete list of processor controls is seen in table 9.

To avoid the clutter of many additional boxes, the “MUX Select” and “Other” buttons of table 9 are implemented without boxes.

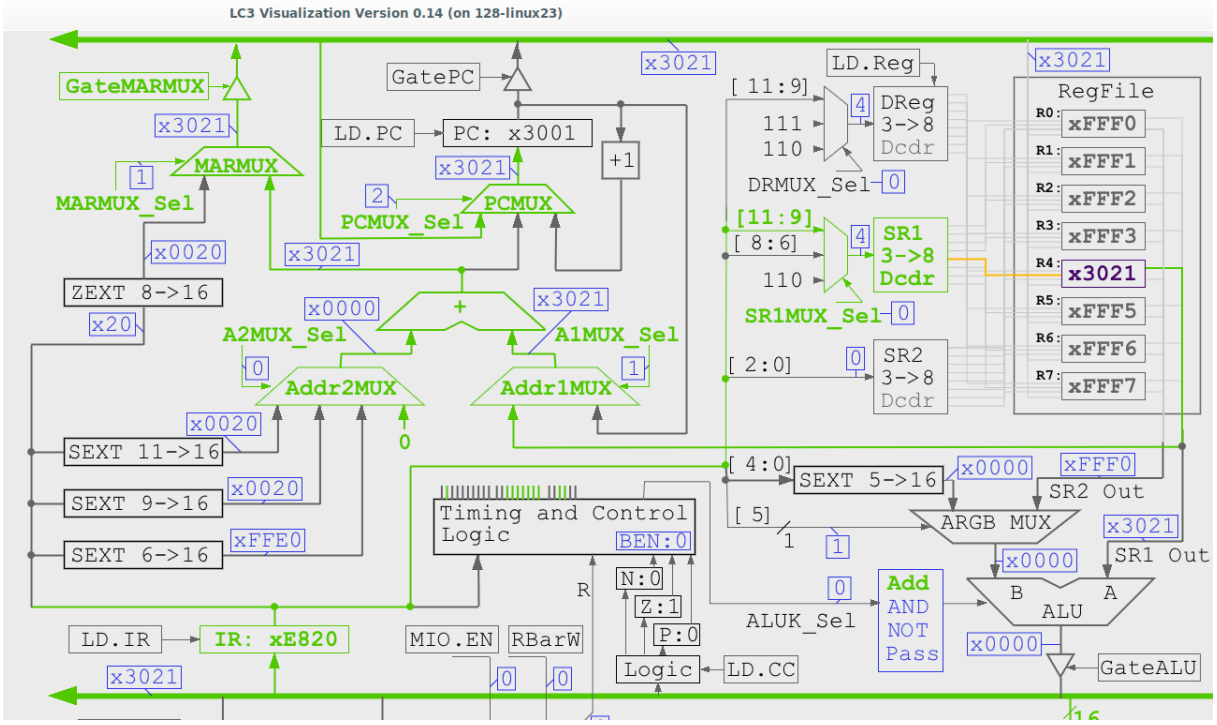
- For the MUXes and ALU, the control is collocated with the corresponding “MUX_Sel” label.

For example, figure 14 shows the LC3_Visualization in Full Control Mode, with the `MARMUX_Sel` label clicked once. The `MARMUX_Sel` label, the `MARMUX` and sources providing inputs to the `MARMUX` are activated.

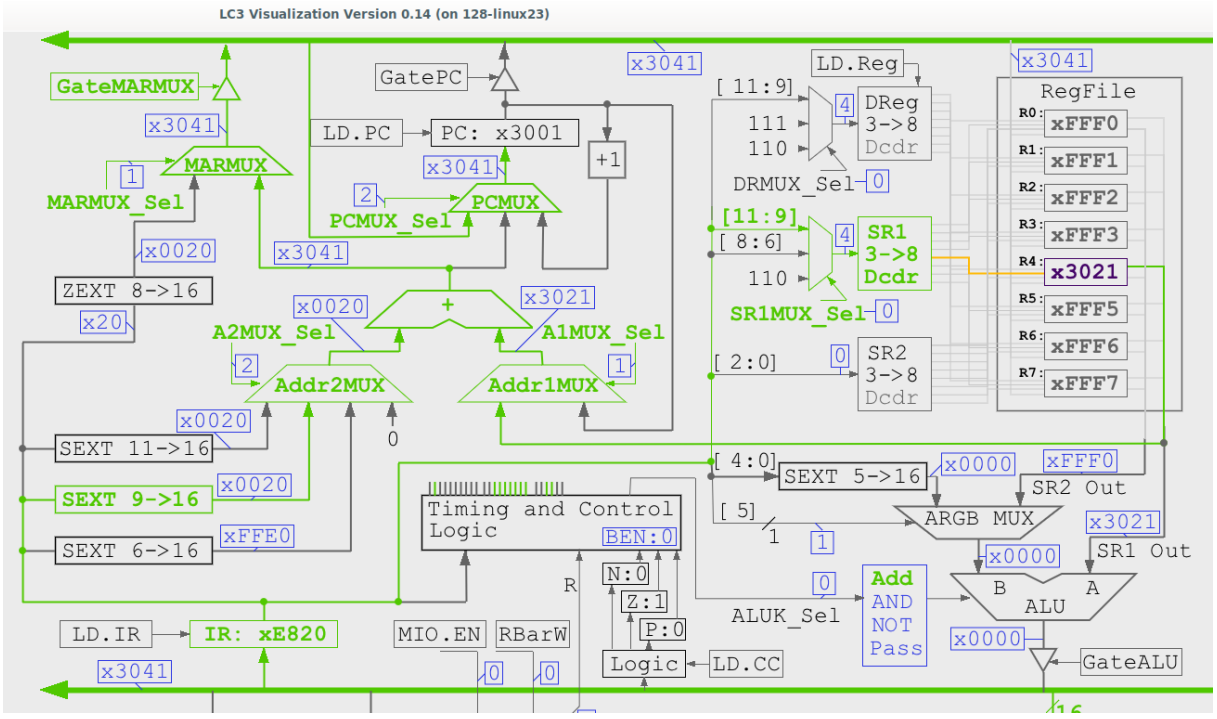
- The exception is the ARGBMux, which is controlled by bit 5 of the instruction and does not have a MUX_Sel label. Its control is collocated with the "ARGB Mux" label, inside that Mux.

Gate	Load	Memory Control	MUX Select	Other
GatePC	LD.MAR	MIO.EN	MARMUX_Sel	ALUK_Sel
GateMDR	LD.MDR	RBarW	Addr1MUX_Sel	
GateALU	LD.IR		Addr2MUX_Sel	
GateMARMUX	LD.PC		PCMUX_Sel	
	LD.Reg		DRMUX_Sel	
			SR1MUX_Sel	
			ARGBMUX_Sel	

Table 9: LC-3 processor control signals in the LC3_Visualization.



(a) Addr2MUX Select set to 0.



(b) Addr2MUX Select set to 2.

Figure 13: An illustration of using Full-control Mode.

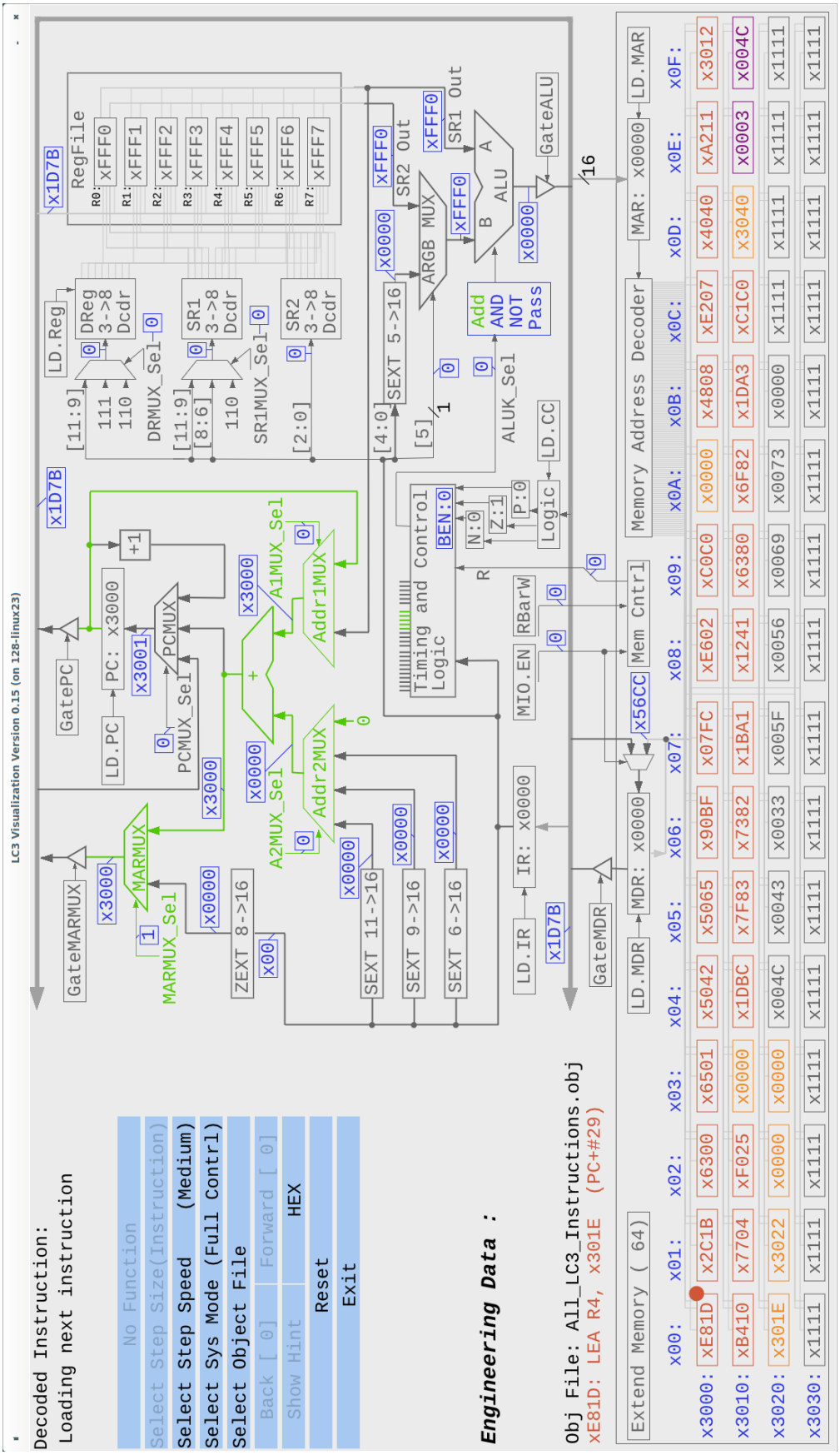


Figure 14: LC3_Visualization in Full Control mode with `MARMUX_Sel` clicked once to increment the MUX Select signal and activate the MUX. Note that the State, Microstep and Execution Phase are no longer listed in the Engineering Data section; these items do not have meaning in Full Control Mode.

3.4.2 Additional functionality for lecture demonstrations

These features described above support use as a lecture demonstration :

- The ability to demonstrate the operation of each CPU element, separate from instructions and program execution;
- The ability to auto-step at multiple step sizes, from running continuously to single micro-steps;
- The ability to roll back instructions, and rock back-and-forth across an instruction, to explain its several actions;
- The ability to visualize registers and memory in the format relevant to the current discussion; and
- The ability to edit memory and register values, including processor registers, and especially quickly when only a small change is needed.

4 Technical notes

1. The LC3_Visualization is built on the Dear ImGui graphical user interface library, written by Omar Cornut and other contributors.

   `https://github.com/ocornut/imgui`

2. LC3_Visualization requires at least 1920 x 1080 resolution to fully display the visualization. The visualization does not scale.

3. The responsiveness to mouse clicks and other aspects of the ImGui application is governed by the refresh rate, itself set by the "Step Speed." Click to "Fast" Step Speed, for quick responses.

   ((LC3_Visualization shifts itself to "Fast," for actions like file selection,
   to have fast responsiveness.))

4. Interrupts, privilege, traps beyond Trap x25 (halt), the program status register and supervisory mode have not been implemented in the LC3_Visualization. These are tasks for a future version.

5. LC-3 processor controls for all elements in the "MUX Select" and "Other" columns of table 9 are co-located with their corresponding MUX_Sel labels, except for the ARGB MUX, which does not have an independent MUX_Sel label (the control corresponds to bit 5 in the Instruction Register). For this element, the control is co-located with the "ARGB MUX" label inside the MUX.

5 Appendices

5.1 Program analysis and memory-word colors

To support memory-region labeling (section 8), the loaded program is analyzed to determine the use of each memory location. During a reset, the LC3_Visualization silently executes the loaded program, and analyzes memory usage. For example, values loaded and executed as instructions are labeled as program memory, while values loaded or stored using R4 are labeled as Global Variables.

The assignment of memory locations to regions is done by static, dynamic and run-time analysis.

Static: The LC3_Visualization loads the object file format written by LC3Tools, written by Chirag Sakhuja and others. The object file includes instructions and assembler directives as text associated with each memory word. These are analyzed to identify program memory and data.

Dynamic: The program is run in Auto-Step mode with no visualization. For example, access to memory locations via pointer R4 or R5 is detected, and the locations are painted the corresponding colors.

Run-time: The initial setting of the static pointer is detected during static analysis. At run time, the current value of the stack pointer is used with the known initial value to determine the stack region.

5.2 Operating modes and LC-3 processor control signals

The LC3_Visualization has four operating modes :

1. Auto-step mode : Demonstrated in the Quick-start section, and most-used.
2. Practice mode : used by students to prepare for Reporting.
3. Reporting mode : The student plays the role of the timing and control logic. Using the mouse, the student activates Gate and Register-Load controls to execute one or several instructions. A gradable report is generated.
4. Full-control mode : All control signals in the LC-3 are manually generated to demonstrate the action buses, multiplexes, decoders, registers and other LC-3 elements.

The use of the LC-3 processor control signals in each of the operating modes is described in table 10.

Operating mode	Active LC-3 processor control signals	Action of LC-3 state machine and automatic instruction execution
Auto-step mode	None	Fully automatic execution
Practice mode	Gate and Load signals (Used in student exercises)	Other processor control signals are automatically generated (see section 3.2.1.1)
Reporting mode	Gate and Load signals (Used in student exercises)	Other processor control signals are automatically generated (see section 3.2.1.1)
Full-control mode	All LC-3 processor control signals active (e.g., MUX select controls) This mode is chiefly used by the instructor, for lecture demonstration	None, state machine is inactive

Table 10: Use of LC-3 processor control signals in the several

The action and uses of the modes are described more fully below. The LC-3 state machine is described next, followed by Reporting mode.

5.3 The LC-3 state machine in the LC3_Visualization

Execution in LC3_Visualization is controlled by a data structure that is a vector of Execution State Records, corresponding to the LC-3 processor state machine described in Appendix C of Patt and Patel, *Introduction to Computing Systems*.

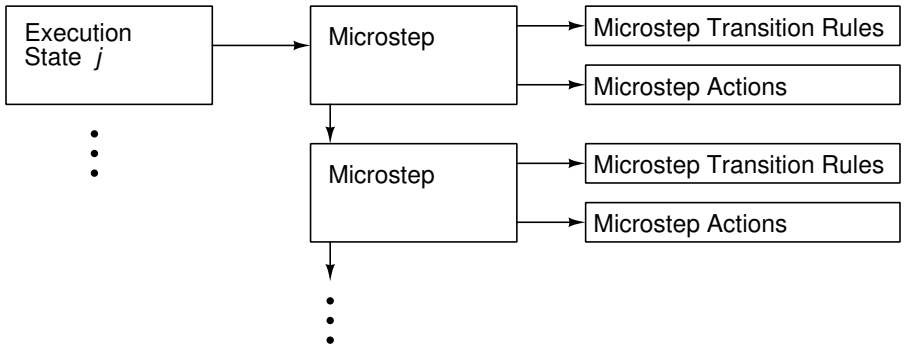


Figure 15: LC3_Visualization LC-3 state-machine. Each execution state in the simulated state machine has an associated Execution State Record.

- Each execution state is associated with a linked list of microsteps, each of which has lists of Transition Rules and Actions, as seen in figure 15. An example Transition Rule is :

  `GatePC` is exerted

  When all Transition Rules for a Microstep are verified, the Microstep Actions are executed, with delays.

  - Delays help create an appearance of propagation.
  - Once all Actions are completed, control passes to the next Microstep.
  - Once all Microsteps are completed, control transitions to the subsequent Execution State, as described in Patt and Patel, Appendix C.

- The state-machine model of figure 15 is built from a configuration file, and so can be revised easily. Configurations for various processors can be managed by selecting the configuration file.

5.4 Visualization control actions

5.4.1 Visualization control actions versus LC-3 control signals

Control signals in the LC3_Visualization are divided into two groups:

1. LC-3 visualization control signals;
2. LC-3 processor control signals.

The former group controls the visualization, such as changing a display or activating auto stepping.

The later group are part of the LC-3 processor simulation, and simulate control signals that would be generated by the LC-3 Timing and Control Unit, as part of instruction execution. As an example of the distinction, when students play the role of the LC-3 Timing and Control Unit to execute instructions, they use the LC-3 processor control signals. For most other actions, such as selecting a display format or loading an object file, LC-3 visualization control signals are used.

In this section, the available LC-3 visualization control actions are enumerated and briefly described. The lists in this section, in particular tables 13-15, are complete in the sense that there are no other LC-3 visualization control actions. Additional details of the control actions are provided in section 5.2.

The LC-3 processor control signals are described in section 5.2.

5.4.2 LC3_Visualization main control buttons

The functions of the “Do Step” button (upper-most of the Control Buttons) are described in table 11.

Button Label	System Mode or Modes	Action in Main Window, independent of System Mode
Button label:		The Do Step button has several context-dependent functions. These are indicated by changing labels.
- Do Step	Auto Step mode, ready to start auto step	Automatically execute instructions through the Auto Step exit condition (see table 15)
- Stop Step	Auto Step mode, auto step in progress	Exit auto step
- Run Continuously	Auto Step mode, Run Continuously step size selected	Start continuous auto step
- Stop	Auto Step mode, Run Continuously running	Stop continuous auto step
- Write Report [*#n]*	Reporting mode	Write a report file; (The value *n* indicates the number of student-actions recorded)
- No Function	Practice and Full Control modes	In these modes, there is no auto step or report

Table 11: LC3_Visualization control functions, control buttons in the upper-left area (see Fig 3(a)).

The functions of the remaining Control Buttons are listed in table 12.

Button Label	System Mode or Modes	Action in Main Window, independent of System Mode
Select Step Size	Auto Step mode	Set the point where auto-step terminates; see figure 5 and table 15
Select Step Speed	All	It is sometimes desirable to slow Auto-Step execution, for better visualization; see section 4, item 3
Select Sys Mode	All	Select the System Mode, see section 5.2; Select from {Auto Step, Practice, Reporting, Full Control}
Select Object File	All	Open the Object File selection dialog; see figure 3(b)
Back [#*n*] / Forward	Auto Step and Practice modes	Back step and forward step, see section 3.3.5.2; Forward is available after stepping back at least one full instruction; (The value *n* indicates the number of possible steps back)
Show Hint	Practice mode	In Practice and Reporting modes, the student executes Timing and Control Unit control actions; in Practice mode, hints are available; Display a hint
HEX (and other Display Modes)	All	Memory and register data can be displayed in various formats; see section 3.2.4
Reset	All	Reset system state; Reload the object file; Selected LC3_Visualization controls, such as Step Speed and Display Mode, are preserved through a Reset
Exit	All	Exit LC3_Visualization; equivalent to Alt-F4 or Ctrl-C

Table 12: LC3_Visualization control functions, control buttons in the upper-left area (see Fig 3(a)).

5.4.3 Main window, key strokes and mouse clicks

5.4.3.1 LC3_Visualization Keyboard controls, main window

- A number of LC3_Visualization control actions can be activated through key strokes. These are seen in table 13.

	System Mode or Modes	Action in Main Window, independent of System Mode
'A' or 'a'	All	Switch memory display to showing addresses, to assist student in identifying memory locations
Ctrl-C Alt-F4	All	Close LC3_Visualization
Space	Auto Step mode	Equivalent to clicking the "Do Step" button. Starts or stops Auto Step.
Space	Reporting mode	Writes report (function of the top button)

Table 13: Keyboard controls in the main window.

5.4.3.2 LC3_Visualization Mouse-click controls, main window

- A number of LC3_Visualization control actions can be activated through mouse clickes. These are seen in table 13.

  The LC3_Visualization control actions control the visualization. Additionally, there is a separate set of mouse-click-activated controls that simulate control signals within the LC-3 processor (for example, `GatePC`, which is a physical signal within the processor and gates the program counter onto the main bus). These are described in sections **??** and 3.4.1.

	System Mode or Modes	Action in Main Window, independent of System Mode
Left click on a Control Button (cf. Fig 3(a))	All	If the button is highlighted, activate the control function; see section 5.4.2
Left click on a memory location	All	Set a breakpoint at the clicked memory location; see section 3.3.1.1
Right click on a memory location or register	All	Enter Stored Value Editing mode; see section 3.3.3.2
Left or right click on a memory location or register, with modifier key(s) pressed	All	"Modifiers" are the Shift, Control, Super (Windows) or Alt keys; With the Ctrl key pressed, a left-mouse click will increment a memory location or register, and right-mouse click will decrement; See section 3.3.3.1
Left click on text "Engineering Data"	Auto Step, Practice & Rpt modes	Activate optional display in the Engineering Data area. At this time, the only optional display is the instruction execution phase (cf. Patt and Patel, chapter 4)

Table 14: Mouse-click controls in main window.

5.4.4 Step size selection dialog

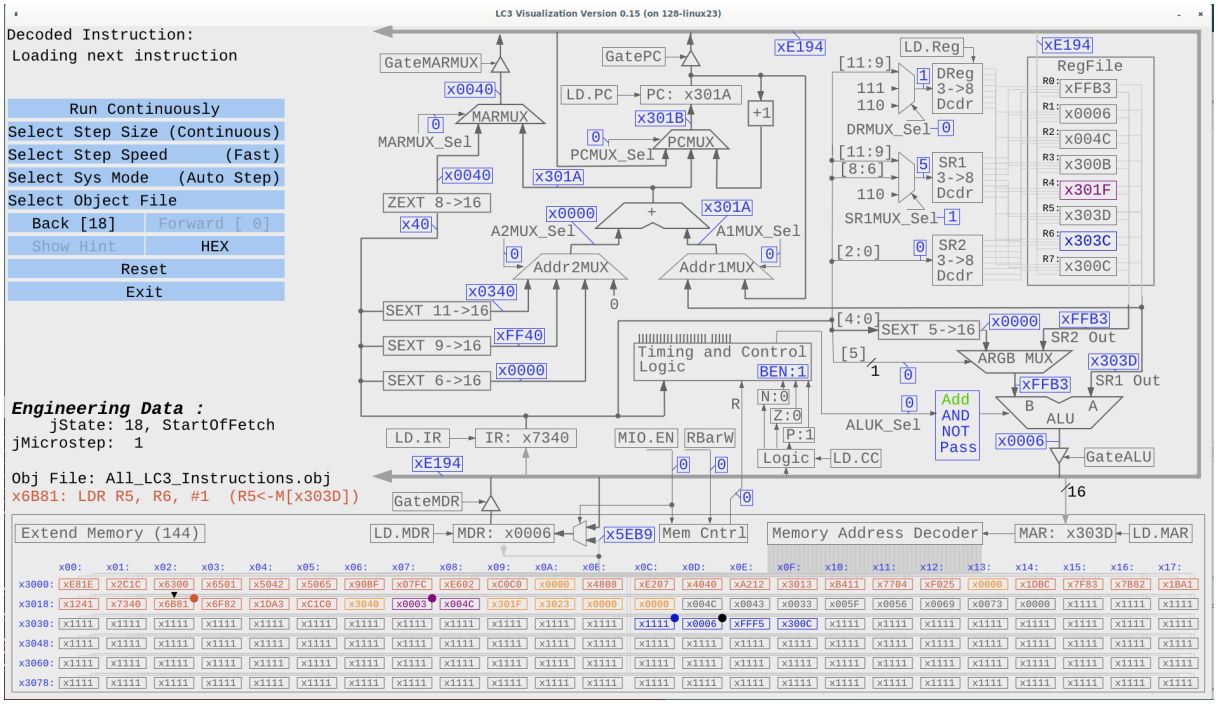
The Step Size selection dialog is seen in figure 5. The action of each step size is seen in table 15.

Step Size	Condition reached to exit Auto step
Run Continuously	Run until stopped by a user action or breakpoint.
Run Thru Decoder Step	Run through State 32, Microstep 0; (The end of the fetch cycle, where the instruction is decoded)
Run Thru Start of Next Instruction	Run through State 18, Microstep 0; (The start of the next instruction fetch cycle)
Run Thru 1st of Dcdr or Instr Fetch	It is sometimes useful for teaching to stop at both the start of instruction fetch and the start of instruction execution
Run Thru State Machine State	Run through one or more Microsteps, to the point where Execution State is advanced
Run One Microstep	Run to the start of the next Microstep

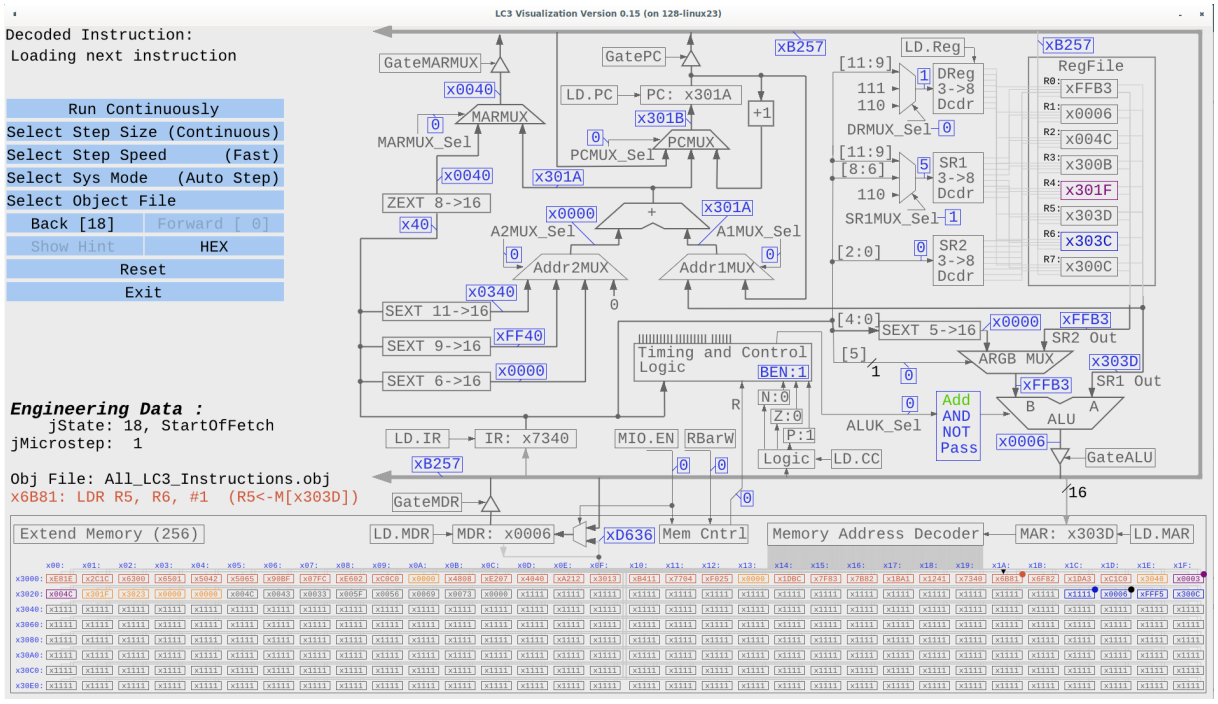
Table 15: Auto-step Step Sizes. In Auto Step, the LC3_Visualization will step up to the condition listed.

5.5 Extending memory

- 144- and 256-word configurations are seen in figure 16. In each case, the demonstration program is loaded. To accommodate larger memory smaller fonts are used. The 18- and 14-pixel fonts used are not suitable for lecture presentation, but are readable on a PC display. At UWM we have found that the 144-word memory is sufficient for all student exercises.



(a) 144 words of memory selected.



(b) 256 words of memory selected.

Figure 16: LC3_Visualization with 144- and 256-word memory configurations. These views show the memory labels (colored dots) described in section 3.3.4.

References

[1] Y. N. Patt and S. J. Patel, *Introduction to Computing Systems; Bits & Gates to C/C++ & Beyond*. New York: McGraw Hill, 3rd ed., 2020.

[2] B. Nikolic, Z. Radivojevic, J. Djordjevic, and V. Milutinovic, "A survey and evaluation of simulators suitable for teaching courses in computer architecture and organization," *IEEE Transactions on Education*, vol. 52, no. 4, pp. 449–458, 2009.

[3] T. Linden and R. Lederman, "Creating visualizations from multimedia building blocks: A simple approach to teaching programming concepts," in *Information Systems Educators Conference - ISECON 2011*, pp. 1–10, Foundation for Information Technology Education, 2011.

[4] J. Sorva, V. Karavirta, and L. Malmi, "A review of generic program visualization systems for introductory programming education," *ACM Transactions on Computing Education (TOCE)*, vol. 13, no. 4, pp. 1–64, 2013.