



[Courseware](#)
[Course Info](#)
[Discussion](#)
[Wiki](#)
[Progress](#)
[Discussion Guidelines](#)
[Resources](#)
[Exploring Engineering](#)

[Syllabus](#)
[How to Use Jade](#)

Help

Since the LC-3 has no multiplication instruction, we need to write an LC-3 program if we want to perform multiplication. **This is an optional, ungraded, lab assignment.**

First, recall multiplication in base 10:

```

  213 [multiplier]
x123 [multiplicand]
-----
  639
 426
213
-----
26199

```

For each successive multiplication of the multiplier by the corresponding bit in the multiplicand, we shift the result of the multiplication left one position, with implicit zeros in the missing positions. That is, "426" is really "4260" and "213" is "21300" when we perform the addition step.

Instead of performing the entire addition at the end, we could do the addition step by step and keep a running total. That is, we first start with 639 as our running total. When we generate 4260, we add that to the running total to get 4899. Finally, we generate 21300 and add it to the current running total of 4899 to get the final answer of 26199.

Now let's look at the same operation using unsigned (positive) binary numbers:

```

  1011 [multiplier]
x0101 [multiplicand]
-----
  1011
 0000
 1011
 0000
-----
0110111

```

In this example, we have a 0 in two positions of the multiplicand. Therefore, we do not need to perform an addition for those two bits. (The same would hold true for base 10.) However, we still needed to ensure that the multiplier is shifted left one position in each step, regardless of the value of the multiplicand. When we perform this shift, we shift a 0 into the LSB.

With this background, we can formally describe the steps for the binary *shift and add* algorithm:

1. Start with the right most bit of the multiplicand.

2. If it is a 1, add the multiplier to the running total.
3. Shift the multiplier left one position, putting a zero into the LSB.
4. Repeat for all remaining bits of the multiplicand.

To check each bit of the multiplicand, we AND it with a bit mask. Initially, the bit mask is 0000000000000001. If the result of the AND of the bit mask with the multiplicand is 0, we branch past step 2.

To shift the multiplier left, we observe that, in binary, shifting left one position is equivalent to doubling the number. For instance, 011 (3 in base 10) shifted left one position is 110 (6 in base 10). In LC-3, we can double the multiplier by adding it to itself.

To form the bit mask for the next bit of the multiplicand, we add the bit mask to itself. For instance, after we check the right most bit, we perform $0000000000000001 + 0000000000000001 = 0000000000000010$ to permit inspection of the next bit.

The above steps are incorporated into a loop that continues until all bits of the multiplicand have been inspected.

One of the issues with multiplication is that final result may be so large as to require two registers. We will assume that the numbers that we are multiplying fit in 8 bits so that we require only one 16-bit result register. Therefore, the algorithm only needs to inspect the right 8 bits of the multiplicand, since the left 8 bits are guaranteed to be 0.

In your program, the two numbers to be multiplied are in locations x4000 and x4001. The final result should be placed into R1.

Remember, the assembler automatically generates the code as you type (it refreshes every 3/4s of a second). The Check button will not respond. It should give errors (red 'X' next to the line number) when it runs into a problem. If new machine code on the right side does not appear, there's usually a problem with your input on the left side.

MULTIPLY (OPTIONAL)

Use CTRL-A (Apple-A) and CTRL-C (Apple-C) to quickly capture the data in the machine code section.

LC-3 Assembler

```
1 .ORIG 0x3000
2 HALT
3 .DATA 0x4000
4 .FILL 0x0093
5 .FILL 0x001A
6 .END
```

```
1 @0b11000000000000
2 0b1111000000100011
3 @0b10000000000000
4 0b0000000010010011
5 0b000000000011010
```

Check button now submits assembly to edX!

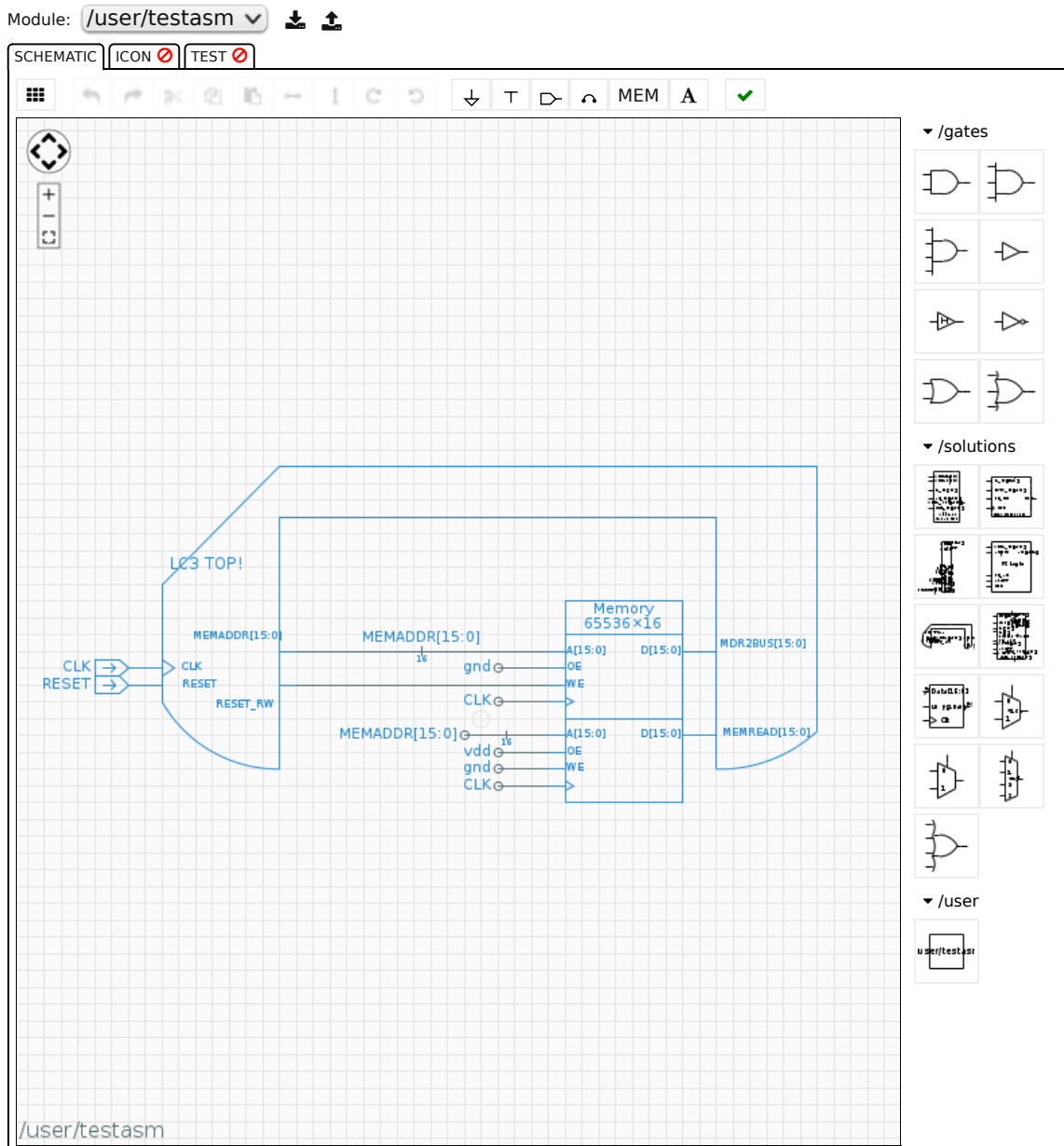
Check

In the window below, the LC-3 Lite memory plug in solution **/solutions/lc3_plugin_memory/**, should already be created and hooked up to a memory. Double click on the memory and paste the entire contents of the right hand side window above into the Contents entry in the Edit Properties window. Because the memory is "enveloped" by the LC3 implementation, you may need to click towards the lower part of the memory to select it (instead of the LC3 itself).

Run the test. It checks to see that at the end of the program, you have successfully multiplied the two numbers! We have allocated a large number of cycles to run, however, if there is a need for more, please post and let us know!

MULTIPLY (OPTIONAL)

Help



Click component to select, click and drag on background for area select, shift-click and drag on background to pan

jade 2.2.43 (2015 © MIT EECS)

Check

If you would like to test your program on your own LC-3 Lite, use the ungraded Jade instance below. Copy your machine code into the memory, and run the test.

You may need to name the `lc3plugin_memory` module "lc3".

It is configured to use the `/user/lc3plugin_memory` module. If you want to use your own test and code, we have moved

that option to a new page here.

This is optional.

Help

TESTING YOUR LC-3 LITE (UNGRADED)

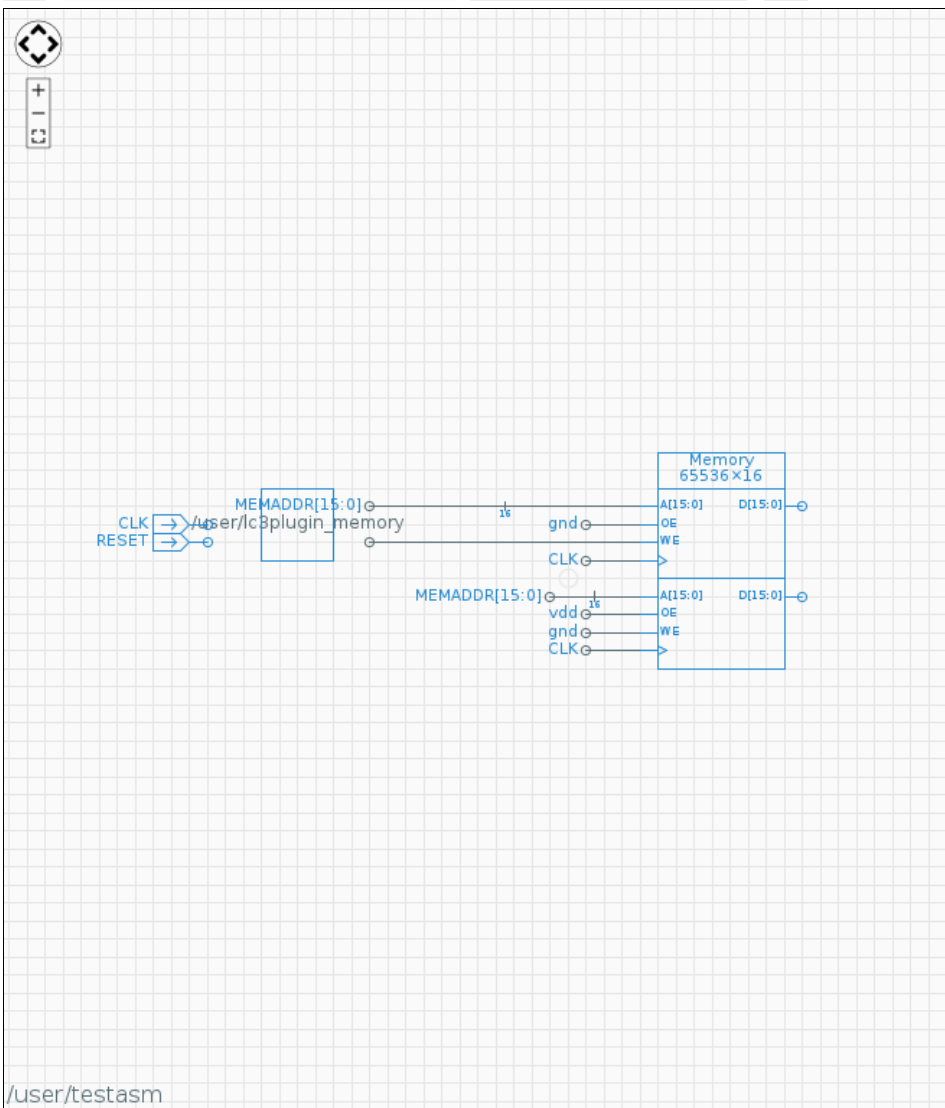
Module: /user/testasm



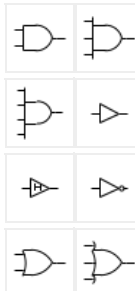
SCHEMATIC

ICON

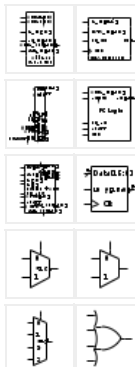
TEST



▼ /gates



▼ /solutions



▼ /user



Click component to select, click and drag on background for area select, shift-click and drag on background to pan

[Jade 2.2.43 \(2015 © MIT EECS\)](#)

Check

Because this is an optional, ungraded, assignment, feel free to share code in the discussion. However, put "[spoiler]" in your title to warn other students who may want to do the lab on their own without seeing other code.

[Show Discussion](#)[New Post](#)[Help](#)

edX offers interactive online classes and MOOCs from the world's best universities. Online courses from MITx, HarvardX, BerkeleyX, UTx and many other universities. Topics include biology, business, chemistry, computer science, economics, finance, electronics, engineering, food and nutrition, history, humanities, law, literature, math, medicine, music, philosophy, physics, science, statistics and more. EdX is a non-profit online initiative created by founding partners Harvard and MIT.

© 2015 edX Inc.

EdX, Open edX, and the edX and Open edX logos are registered trademarks or trademarks of edX Inc.

[Terms of Service and Honor Code](#)

[Privacy Policy \(Revised 10/22/2014\)](#)



About edX

[About](#)

[News](#)

[Contact](#)

[FAQ](#)

[edX Blog](#)

[Donate to edX](#)

[Jobs at edX](#)


Follow Us


 [Facebook](#)


 [Twitter](#)


 [LinkedIn](#)

 [Google+](#)

 [Tumblr](#)

 [Meetup](#)

 [Reddit](#)

 [Youtube](#)