LEARNINGNERD

HOME › CATEGORIES: DAILY LEARNING NOTES

Daily Learning Notes for February 1, 2016

I took a break from studying this weekend to finish a little home improvement project (I have room for all my books now, yay!) and then yesterday I had an all-day party at my place to celebrate the one year anniversary of Learn to Code LA! It's hard to believe that I started the meetup group a year ago! So much has happened since then, and the experience of starting a community organization, teaching classes, hosting events, meeting so many new people – it really has changed my life! I am a different person because of it. Anyway, I'll be writing more on that later because I'm giving a talk in couple weeks about my community-building adventure.

Today I wanted to decompress from yesterday's socializing, and I know the only thing likely to keep my mind focused right now is the NAND2Tetris course. I'm hoping that this week's project will be challenging enough that I can lose myself in focused thought for a while.

I read over the specs for this project the other day, and now it's time to dive in to the project for week 5: building the final Hack computer!

Overview

Time breakdown

Building the final memory chip

The first step will be to build a memory chip that combines a 16K RAM chip (which I built the other week) with the provided chips for the screen and keyboard. (I wish this course would teach me about how to use those too, but I understand that would take a while!)

The book says the memory chip must have “a single logical address space, spanning from location 0 to 24576 (0x0000 to 0x6000)”.

First question: what is “0x0000” or “0x6000”? I don’t remember seeing an explanation of that notation in the book! Let’s search online… Oh, that’s hexadecimal! OK, good to know.

Second question: How do I create a continuous address space? They say I can use the same technique I used back when I built my RAM chips… That was a couple weeks ago, and I think I forgot how I did it. Time to review chapter 3!

Oh right, this was the part where they talked about recursive ascent. That’s such a cool phrase. OK, brain, concentrate! I’m pretty sleep-deprived and just generally exhausted after yesterday’s party.

OK, here are a couple important specifications:

- The screen’s memory map: 16384-24575 (0x4000-0x5FFF)
- The keyboard’s memory map: 24576 (0x6000)
- “Access to any address>24576 (0x6000) is invalid”
- I assume that address 0 to 16383 is for everything else

The memory chip takes a 16-bit data input, a single-bit load input, and a 15-bit address input. Its output is whatever value is stored at the specified address. And if the load bit is set to 1, the data input is then stored in memory at the specified address.

OK, so that's how the memory chip works… I'll need to split up the address input somehow, I know that. The 16K RAM takes a 14-bit address, the screen takes a 1 address (otherwise it works exactly like a 16-bit 8K RAM chip), and the keyboar just a single register at a single address.

How do I split this up?! Hmm. I think a review of the powers of two might be helpful at this point. So 2 to the power of 14 is 16,384 – that's why I only need a 14-bit address for the 16K RAM chip, which occupies addresses 0 to 16,383.

Hey, that gives me part of my answer! The memory chip takes a 15-bit address input, so I can just pass the first 14 bits of the address into the 16K RAM chip, but if the largest bit of the 15-bit address is 1, then I need to access the screen or keyboard instead. So I can just use a demux to do that. Easy!

But how do I target the keyboard chip? How do I check if the address is pointing to location 24576? Once again, I need to review my binary! OK, let's see…

My 15-bit addresses in binary:

- 16,383 in binary is 011111111111111
- 16,384 in binary is 100000000000000
- 24,575 in binary is 101111111111111
- 24,576 in binary is 110000000000000

Nice, I see a pattern! If the most significant bit is 1, then I'm targeting the screen or keyboard. If the most significant bit is 1 and the *second* most significant bit is 1, then I'm targeting the keyboard. Easy! All I need is another demux!

Oh, interesting… the keyboard chip doesn't actually have any inputs. But I still need to funnel the contents of the keyboard's memory maps to the memory chip's output if the specified address is 24576. I guess that part will happen separately, then; I need to handle routing the inputs and then I need to handle routing the outputs.

A while later: OK, done! I wrote up my design in HDL and ran the chip's test script in the hardware simulator, and it worked! Aw, yeah! We have memory!

Starting to build the central processing ur

The memory chip was the easy part – the appetizer, if you will. Now for the main course: building the CPU! How do I build a CPU? Where do I even start?! Back to the specs…

CPU basic specifications:

- Three inputs: inM[16], taken from data memory; isntruction[16], taken from instruction memory (the ROM), and a single-bit reset input.
- Four outputs: outM[16], data to store in the data memory; writeM, a single bit indicating whether to write new data; addressM[15], the data memory address; and pc[15], the address of the next instruction.
- If reset is 1, the CPU jumps to address 0 and sets the program counter to 0, ignoring the current program's instructions.

This is the most complicated chip I've had to build so far! But it's not *that* complicated. I just need to figure out how to decode the instructions and use them to execute the right operations.

Breakdown of instruction decoding:

- Instructions are 16 bits long.
- If the first bit is 1, the instruction represents a 15-bit address and should be stored in the A register.
- If the first bit is 0, the instruction will execute an operation.
- The operation instruction format has three main parts:
    - a 7-bit code indicating what operation to run,
    - a 3-bit code indicating which destination to store the result in,
    - and a 3-bit jump code indicating which condition to check (using the ALU's single-bit outputs) in order to determine whether or not to jump to the instruction located at the

address stored in the A register (if not, the program counter should just increment as usual to fetch the next instruction).

My next step: figure out if there's any rhyme or reason to the instruction codes! sure there must be. They must have some sort of pattern to make for a more efficient and straightforward mapping of instruction code bits to the ALU and other bits. I'm sure it isn't arbitrary.

Oh, duh! I'm silly. The instruction bits perfectly match the ALU's control bits! Well, that makes things easy!

Unfortunately, I need to prepare for tonight's programming workshop, so I guess that's it for today. Until tomorrow!

Learning summary (#TIL)

- I built the final memory chip, combining the RAM chip I built in week 3 with the provided chips for the screen and keyboard.
- I thoroughly reviewed the specifications for the CPU and started creating my design for it.

Next steps

- Complete the CPU and the final computer chip to finish the project for week 5!

Time breakdown

- Study time: 3 hours 39 min
  - Week 5 project: 3 hours 39 min

Written by Liz Krane
Liz is a full-time nerd, sharing everything she learns as she tries to learn everything!

Share on Twitter Share on Facebook Share on Google+

Updated February 01, 2016

0 Comments LearningNerd 1 Login

Recommend Share Sort by Best

Start the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

Be the first to comment.

ALSO ON LEARNINGNERD

Learn Teach Code Vlog #1 (Unedited): Starting my own programming bootcamp

1 comment • 7 months ago

paper services — Thanks for sharing this video blog Liz. Although it is very evident that you are anxious in the video. Keep it up.

English Punctuation: Periods, Question Marks, and Exclamation Marks

2 comments • 6 months ago

dissertation writing help — knowing the importance of making the questionnaire proper as well as professional each step is the

A moment with my dad and the Pythagorean theorem (Math Immersion

1 comment • 3 months ago

Learn Mathematics — https://youtu.be/aF9xR1rD3Ak

English Grammar: Basic Sentence Elements

1 comment • 8 months ago

case study help online — Thanks posting these important terms of the basic sentence elements for English grammar. These are all important

Subscribe Add Disqus to your siteAdd DisqusAdd Privacy

Advertisement