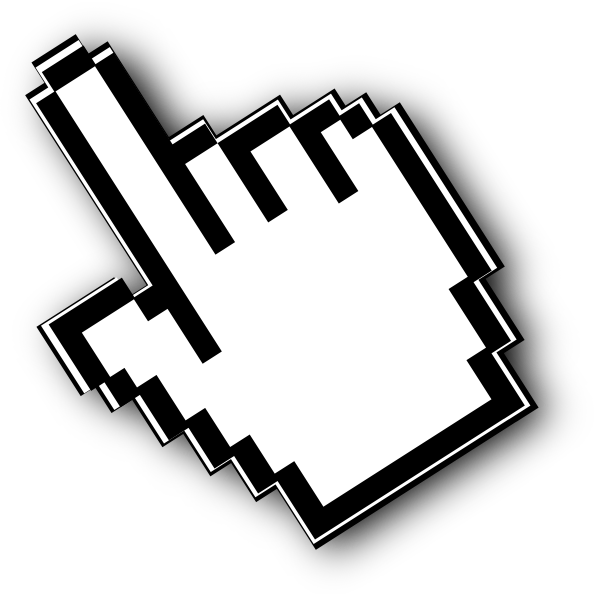
****

The Guide to the Simulator

# TABLE OF CONTENTS

About the simulator 2

Getting started 3-4

Installing the simulator 3

The boot window 3-4

pc emulator 4-5

loading & running assembly code 5

about assembly 5

Datapath builder 6

Control builder 7

Known issues 7

fun facts 8

common datapath structures 8

common combination circuits 9

footnotes 10-11

how to contribute 11

# About the simulator

Computer architecture and embedded systems are taught in almost every computer hardware or software program. Here at Bridgewater State University we have COMP 206 Computer Organization and COMP 590 Computer Architecture. In both of these classes an understanding of low-level logic is important.

The Simulator is used to emulate computer hardware and a personal computer. The processor design component has a Datapath builder and a Control Builder. Both are used to create a custom processor.

The Datapath Builder contains structures like registers, constants, and logic gates that are connected using busses. The datapath that you create can be run using the “Simulate” button in the left menu. You can save or load a datapath from the same menu.

The control builder is a component where you create a finite state machine that dictates how your datapath runs. Within the control builder you can set states to execute a specific instruction based how you set conditional structures from your datapath. You can create paths, instructions, states, save and load controls, verify, set conditions for instructions, and more.

The PC component is a cross platform emulator called Bochs2. It is used in class to run an operating system called FreeDos3 or to build your own operating system (like BreakoutOS).

# Getting Started

Do not try to figure everything out at once.

Dr. Black will explain parts of the simulator when it is important. This guide, however, is here if you want more information, forget to how something works, or want to explore other parts.

## Installing the simulator

Visit Dr. Black’s repository site

<https://github.com/mdblack/simulator/blob/master/simulator.jar>

Click “View Raw”. This will download a file called simulator.jar.

Double click to run the simulator.

## The Boot Window

In the Boot window are several components that you may use over time.

Storage devices emulate drives on PCs, particularly old floppy drives.

Memory images are typically hex values for programs used in the processor design component.

Datapath is where you can load a datapath that you previously created and the same goes for control.

These buttons will take you to different parts of the simulator:   
**“Boot Floppy A”** will run the PC emulator and load the file you chose next to floppy A.

**“Processor Design”** takes you to the Datapath Builder where you will build a datapath for a custom CPU. If you click “Control” on the menu to the left this will take you to the Control Builder window where you can write the state machine for your CPU (this will make sense in time).

# PC Emulator

Under “Storage Devices” you will see two floppies and two hard drives. If you have never booted the simulator before the first thing you want to do is click “Make me some disks”.

This will create two files in the same directory that you have the simulator in. This process takes a while.

After the disks are made click “Boot Floppy A” and wait until you get an A:\> prompt. This is a PC simulator running an operating system called FreeDOS3. Click the “Pause” button on the bottom to pause the program.

Some common commands:  
**TASM**: Turbo Assembler for x86 assembly projects  
**TLINK**: Turbo Linker for x86 assembly projects   
**DIR**: shows you what files you have in the current directory  
**CD**: changes your current directory  
For more commands checkout the FreeDos website4.

To check out the files included with Floppy A click “Disk” from the top menu. Next click “Edit File” and select the file from a drop-down menu and click “Edit”. I recommend you do not change these files if you have projects saved in this floppy. If you have to make a new floppya.img your loaded files will no longer be included.

You will use the PC emulator when you are learning x86 assembly language.

## Loading & running assembly code

You will write x86 assembly code in class. This will be done in an “Edit File” window. To get to the “Edit File” window you should first boot Floppy A, wait for the A:/> prompt and pause your program. Next click “Disk” at the top, then click “Edit File”.

You will be writing your code in the larger textfeild. When you are finished, create a file name that is no more than 8 characters long followed by .asm. Click the “Save” button. Make sure you save a copy to your computer as a backup. You will need this in case you need to make a new Floppy A.

To compile your code, you will be using an assembler and a linker7. Below are the commands you will use to run your code.

A:/> tasm test.asm  
A:/> tlink test

A:/> test

## About Assembly

Assembly language is a low-level programming language closely related to a particular architecture’s machine code instructions. You will be using x86 assembly language in the simulator.

In x86 there are two main types of syntax: Intel and AT&T. In class you will learn the Intel syntax5.

# Datapath builder

The datapath builder is a place where you can create your own datapath. The simulator abstracts some of the structures so you do not have to worry about the details inside these structures.

In the datapath you will use structures under “Place a new:” to build the hardware for your processor (new to datapaths? See footnote 6.).

After building the datapath you can click “Save” to save the datapath to a file. The next time you want to edit the datapath you can boot the simulator and click “choose” next to the datapath in the Boot window. If you also have a control for the datapath be sure to also load the control before you click “processor design”.

After you are done building a circuit you can click verify to make sure you hooked everything up properly. If you see red that means that you have some errors. If you accidentally added some busses (a common mistake) you can click them until they are the only components that are red, then click “delete” from the menu on the left.

For many structures you can change the name, value, size, bits, and more by right clicking the structure you want to change.

If you have no errors after clicking “verify” **save your file**. We recommend placing datapath in the filename. This is because you will need to differentiate between this file and your control. If you overwrite your datapath it is **gone**. Next, click “Simulate” to begin your simulation. After you click “Simulate” you can run your simulation by clicking “Play” or “Step”.

If you want to learn about the different structures, see List of Datapath Structures and Explanation of Combinational Circuits.

# Control Builder

The control builder is a component where you create a finite state machine that dictates how your datapath runs. Within the control builder you can set states to execute a specific instruction based how you set conditional structures from your datapath. You can create paths, instructions, states, save and load controls, verify, set conditions for instructions, and more.

The control builder sets when datapath structures should be clocked during the simulation.

This will be covered in more depth in class.

# Known Issues

* If you have a high-resolution screen- drop your resolution to 1920 x 1080
* Some windows do not scale when resized like Floppy A (AKA the Screen), the boot window, and many of the windows that load from the top menu.
* In the datapath builder do not place labels near buses.
* In the datapath builder constants are changed by name not value. And click enter not update to change the value.
* Do not use duplicate.
* Memory is indexed by hex values not decimal values.
* To see what value is stored at a particular index in memory, first change the index value then click enter.
* Do not use module.
* Once you place a bus, do not extend it by the tip of the arrow.
* When joining two structures with buses, make sure the bus to the destination is within the bounds on the structure. Just because the structure is red it does not mean it is connected properly.
* Do not click the motherboard without first clicking boot floppy a or the windows will not be redrawn properly

# Fun facts

The motherboard image under all the windows is interactive. After you boot Floppy A or processor design minimize your current window. Some clickable components will open windows to their designated simulator areas. For example, video will bring up a window displaying the screen.

In the datapath, clicking “b” on your keyboard is a shortcut to clicking the “bus” button.

# common DATAPATH STRUCTURES

**Bus**: A wire that connects two or more structures together  
**Joiner**: will join bits from incoming wires onto one wire  
**Splitter**: will split incoming wire into wires with split bits  
**Register**: Structures that hold values  
**Flag**: Can be set to true or false  
**Memory**: is a structure that provides access to the PC emulator’s memory  
**Multiplexor**: a structure that uses a selector line to choose which of the incoming wires to use as an output  
**ALU**: Use this structure to perform mathematical/logical operations  
**Constant**: holds an value that does not change  
**Input Pin**: use this structure to pass a hex value into your datapath. Remember to set the number of bits.  
**Output Pin**: use this structure to capture a hex value from your datapath. Remember to set the number of bits.  
**control** (small c): structures that can be set as conditions in the control builder  
**Module**: don’t use!  
**Label**: a few words that can be placed near another structure to provide information. Do not put it directly next to any structure.

# common Combination Circuits

**Adder**: Adds the two incoming wires and outputs the value to the output bus.  
**Negate**: will flip the bits of the incoming wire and add 1 and output the result  
**Increment**: takes the input bus, adds 1, and outputs the bus  
**Decrement**: takes the input bus, subtracts 1, and outputs the bus  
**And**: The output will be a bus with 1 if all the inputs are 1  
**Or**: The output will be a bus with 1 if one the inputs are 1  
**Nand**: The output will be a bus with 1 if all the input wires are not 1.  
**Nor**: The output will be a bus with 1 if all the inputs are 1 or if all the inputs are 0  
**Not**: The output will be the opposite of the incoming wire  
**Xor**: The output will be 1 if the input wires are a 0 and 1(and vice versa).  
**Equal-to**: will output 1 if the two inputs are equal, 0 if not.

# footnotes

1. **The Architects was the name of a group that worked on Dr. Black’s simulator in Spring 2018. They were responsible for fixing some issues like resizing of windows, not have multiple components boot from the boot window, being able to click simulate to stop simulating, and removing the necessity to add 00 at the end of a program. Team: Andy Couto, Amanda Morrisson, Dave Chamberlain, Greg Johnson, and Tom Smith.**
2. **PC Emulator:** <http://bochs.sourceforge.net/>
3. **FreeDos:** <http://www.freedos.org/>
4. **FreeDos Commands:** <http://www.freedos.org/software/> **. Many of these commands do not work, but some do. Have fun figuring out which ones are supported!**
5. Differences between Intel and AT&T x86 syntax: <https://imada.sdu.dk/Courses/DM18/Litteratur/IntelnATT.htm>
6. Learn about datapaths:

<https://www.cise.ufl.edu/~mssz/CompOrg/CDA-proc.html> <https://www.youtube.com/watch?v=ibYYqvp9FmU> <https://www.youtube.com/watch?v=YGSAWqQy9bI>

<https://www.youtube.com/watch?v=oETOwVBzu1s>

1. What is an assembler/linker:

<https://cs.gmu.edu/~setia/cs365-S02/assembler.pdf>

1. What is a version control system:

<https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control>

1. How to contribute to a repo using GitHub:

<https://akrabat.com/the-beginners-guide-to-contributing-to-a-github-project/>

# How To contribute

Feedback and/or contribution is encouraged on the simulator or this guide.

If you want to contribute make a GitHub account. This will be your version control system8.

Make sure to clone the repository and check out the documentation on how to use GitHub9.