# EC 413 Fall 2019

# **Project Report**

## Team #7

### Members:

* Ece Sureyya Birol
* Ece Ari

## Overview

The main goal of this project is to produce a Control Process Unit as efficient as possible. CPU executes programs using the instruction cycle which consists generally of fetch, decode and execute. CPU contains a Random Access Memory which is used for store and load, a Decode to decode the instructions that have been fetched, a Register File that stores the register values, an Arithmetic and Logic Unit which does various calculations and a Fetch which assigns the address of the program counter. With the top module, we connect all of these small parts into a Processor.

## Design Choices

The first design choice was to make different modules for different parts of the processor. This made debugging easier, where we could spot where the error was by singularly testing module test benches. Our processor is made up of 5 parts which are as follows: ALU, RAM, Register File, Decode, Fetch. We built everything on top of each other, and the Top Module connected all the single modules together.

We used the same variable names for input and output wires which made our life easier. We didn’t have to go back and forth with the code checking if we have the right variable name in the right place.

## Key Modules

Fetch module is where the PC counter is incremented, decremented, and if jump occurred then get the target PC value.

Decode module is where the decode of the instructions occur. This module lets us decode the run code into smaller bits for the program to run it. This way it can efficiently run the program.

RAM is our random access memory, one of the most important part. This is where we load and store function, but also while doing that can fetch instruction. This is where the magic happens.

ALU, which we have done in part one, is the bones of this project. It is where we have declared all of the different instructions our CPU can do. With all different types of function, we have a fully functional processor. For example, branching, adding, jumping, saving and even more!

Good old Register File, we particularly love this one. It wasn’t too complicated, so it gave us an easier time than the rest of them. Here we are either reading or writing data, so it checks to see which one we are doing. This is the memory of a central processing unit which, according to the programmers needs, either writes a register that is needed to be stored or reads a register that will be used for other applications in CPU. Thus, the outputs of a register file are the data that have been read which are connected to the multiplexers that connects to the ALU. So with these, the processor knows exactly what to do with a given instruction and how to take care of it. Isn’t that great Albert?

If we are to talk about the wiring, that happens in the top module, where you cautiously connect every bit of material you have all together. To avoid the confusion, we chose to actually name all the input wires the same as the output ones. This way we didn’t have to worry about actually going back and forth and constantly checking if we wrote down the right variable in the right place. Here, the inputs and outputs are connected with a wire to the given places. This basically puts together the processor, and with this module we have a complete functioning processor!

## Teaming

We both equally divided the work among us. We usually worked together in sync, which lead to a better organized end result. We tried to meet up after classes to work on this project. Working at the same time on the code helped us in the way that we got to talk about our design choices and we got to spot our errors with two pairs of eyes. We started the project by setting up a private GITHUB, where each team member would commit their part, and from there we would edit each other’s code. This eased up our process, and kept us updated on various parts of the project. Overall, each part was done by both of us, which was awesome.

## Concluding Remarks

We believe that with the project we have got to put our knowledge in to practical use which helped us understand better how the CPU and its components work. However, we believe that there is more to do to improve our processor unit. For example, we thought about adding a cache to our design, but didn’t really have the time to. We might do it in the future though. Other than that, we thought adding buffers might make the processor run better and faster. Also, there is more room for improvement in our test benches, which I believe we keep on missing some cases, which lead to out deduction of points. Also, we believe if we were much more experienced in Verilog coding, we could’ve came up with a more efficient code. But overall, we think that we did a good job with this processor.