Texas Tech University
Department of Electrical and Computer Engineering

Course Name: Modern Digital System Design
Number: ECE2372 **Semester:** Fall 2024
Instructor: Dr. Juan Carlos Rojas **Email:** Juan-Carlos.Rojas@ttu.edu

Project 2

This project is to be completed in teams of 3-4. Only one submission per team is necessary.

Submit your solutions packed together into a single ZIP file, and a single report document in Word or PDF, and your project presentation in PowerPoint or PDF.

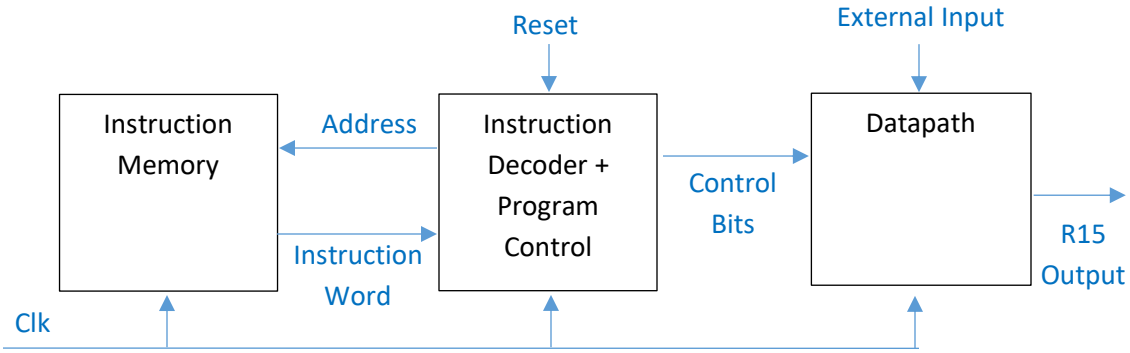
Project Description

You will design a simple CPU architecture, capable of performing basic arithmetic and logic instructions on 8-bit operands.

CPU Architecture

The CPU will have 3 elements:

1. An instruction memory
2. An instruction decoder
3. A Datapath (register file + ALU)



Instruction Memory

This is synchronous (clocked) circuit that behaves like a ROM. Every cycle, the circuit outputs the instruction word at the address specified by the Address input port. Instruction addresses will be 8-bits, which means that the entire memory will have 256 positions. The instructions themselves will be 16-bits wide. This means that each instruction will require two memory positions. Therefore, the memory will be able to hold a total of 128 instructions.

This circuit can only read instructions from memory. It cannot modify the contents of the memory.

Therefore, the only way to have programs in memory is to have them be pre-programed into the memory itself, like a ROM. In other words, your Verilog will have a constant table that represents the program memory.

Instruction Decoder + Program Control

This module is in charge of keeping track of the current program position (Program Counter), fetching instructions from memory, interpreting them, and setting the appropriate control bits for the Datapath.

When the module is reset (when the Reset signal is active), the module will reset the Program Counter to 0. Every clock cycle after that, the module will:

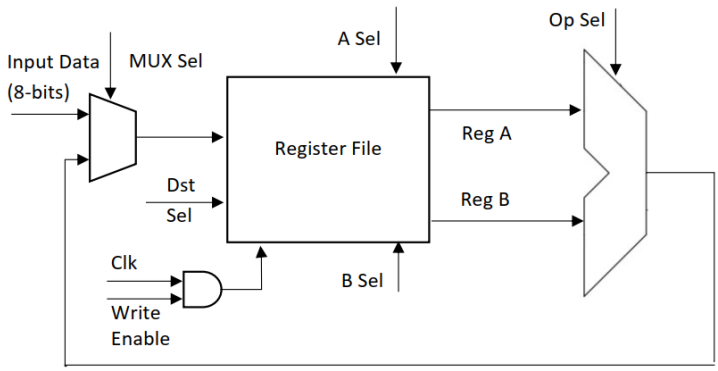
1. Set the Address (based on the Program Counter)
2. Increment the Program Counter
3. Read the Instruction Word from memory
    a. Note that it arrives one cycle later
4. Interpret the Instruction Word
5. Set the control bits as appropriate, to control the Datapath

Note that the instruction set defined below has no jumps or branches. Therefore, the program will always execute the instructions in sequential order as they appear in memory. When the decoder encounters a Halt instruction, it should stop reading more instructions and preserve the current state of the Datapath.

The exact format of the Instruction Words and the Control Bits is part of the design work.

Datapath

The Datapath has the same structure as the one you developed for Project 1:



The register file should contain 16 8-bit registers. The registers serve as inputs to the ALU. Any two registers can be selected as the A and B inputs to the ALU, based on some Control Bits.

The operation performed by the ALU will be slightly different from those in Project 1. You will have to determine the operations required based on the Instruction Set described below. You can choose the

Op Sel formats.

The result of the ALU can be stored into any register, depending on some Control Bits. Also, any register can be updated from an 8-bit External Input based on the Control Bits.

The current value of R15 will always be exposed as an output. It will be wired to an external display on the BASYS-3 board.

Instruction Set Architecture

Your device should support the following instructions:

OpCode (4 bits)	Operand 1 (4 bits)	Operand 2 (4 bits)	Operand 3 (4 bits)	Description
Set to Constant	Dst Reg Idx	Value (upper 4 bits)	Value (lower 4 bits)	Sets destination register to value specified
Load External	Dst Reg Idx			Loads the external input into the destination register.
Copy	Dst Reg Idx	Src Reg Idx		Copies the value of the source register into the destination register.
Conditional Copy	Dst Reg Idx	Src Reg Idx	Cond Reg Idx	Copies the value of the source register into the destination register if the Condition register is non-zero. Otherwise it preserves the old value of the destination register.
Add	Dst Reg Idx	Src Reg Idx		Adds the value in the source register to the destination register.
Negate	Dst Reg Idx	Src Reg Idx		Computes the two's complement of the value in the source register and stores it in the destination register.
AND	Dst Reg Idx	Src Reg Idx		Performs a bit-wise AND of the value in the source register with the value in the destination register.
OR	Dst Reg Idx	Src Reg Idx		Performs bit-wise OR of the value in the source register with the value in the destination register, if the LSB of the conditional register is 1.
Shift Left	Dst Reg Idx	Value		Shifts left the value in the destination register by a number of bits equal to the value specified.
Shift Right	Dst Reg Idx	Value		Shifts right the value in the destination register by a number of bits equal to the value specified.
Equals	Dst Reg Idx	Src Reg Idx		Compares the values in the source and destination registers. If they are equal, then a 1 is written to the destination register, otherwise 0 is written.
Greater Than	Dst Reg Idx	Src Reg Idx		Compares the values in the source and destination registers. If the source value is greater than the destination value, then a 1 is written to the destination register, otherwise 0 is written.

Unconditional Halt				Halt execution of the program from this point onwards.
Conditional Halt			Cond Reg idx	Halt execution if the value in the conditional register is non-zero. Otherwise continue execution of the program.

Note that there are no jump instructions. Therefore, only straight sequential programs will be able to run on this CPU. This also means that you cannot program any loops.

To implement conditional code (like Ifs), you will need to use replication of the code, and conditional copy of results as appropriate. The conditional halt instruction can help stop execution early when the program has reached the desired result.

Top Module and Basys-3 Board

Create a top module that instantiates your CPU modules, and connects them to resources available in the BASYS-3 board as follows:

- The 8-bit External Input should come from board switches
- The Reset signal should come from a board push-button
- The 8-bit output (result of R15) should be connected to LEDs, and also converted into two 7-segment display hex digits.

Microcode Programs

You should write the following microcode programs using the instruction set defined above, and store them in your Instruction Memory:

Program 1

This program should compute the sum of all integers from 1 to N, where N is the number in the External Input. If the sum exceeds 255, then the system should display 255. Test it with different values of N.

Program 2

This program should compute the square of N, where N is the input provided by the testbench. If the result exceeds 255, then the system should display 255. Test it with different values of N.

Programs 3 & 4

Design two programs of your choice, using the available resources.

Note that the Instruction Memory can only hold one program at a time. It would be a good idea to have 4 Instruction Memories, each with one of the programs, and a 4:1 MUX that selects which of the programs is active based on two input switches.

Verilog Code Expectations

Your Verilog code should implement the blocks specified above as separate modules, with input and output ports matching those in the block diagram.

Your Verilog code should be well commented and should use descriptive module and variable names.

Report Document Expectations

Your report document should include the following sections:

Design description

You should describe your design in detail. You should include diagrams and Verilog code sections, with explanations of how it works.

Test description and results

Describe your tests. For each test, show your results, and then justify why these results meet the expectations.

Work division

Who did what?

Retrospective

Provide some insight into your experience developing this project. Was it fun? Was it hard? What was the hardest part? What would you have done different had you had more time?

Presentation Expectations

Your team will be asked to present your design. The presentation should be planned for approximately 10 minutes. It should include:

- Major parts of your design. Use block diagrams to illustrate how the pieces connect.
- Your test programs and results
- A live demo
- Retrospective

Every team member should participate in the presentation.