A simple 8-bit processor

The objective of this project is to design a simple microprocessor with a custom instruction set. The processor should consist of the following four main components:

- A **register file**
- An **Arithmetic and Logic Unit** (ALU)
- A read-only **instruction memory** (IMEM)
- A read/write **data memory** (DMEM)

Instruction Set and Format

Your processor should implement the following **basic instruction** set:

Instruction	Opcode/Function	Operation
add rd, ra, rb	0000 / 000	rd = ra + rb
sub rd, ra, rb	0000 / 010	rd = ra - rb
and rd, ra, rb	0000 / 100	rd = ra AND rb
or rd, ra, rb	0000 / 101	rd = ra OR rb
addi rd, ra, imm	0100	rd = ra + imm
lw rd, imm(ra)	1011	rd = DMEM[ra + imm]
sw rd, imm(ra)	1111	DMEM[ra + imm] = rd
beq rd, ra, imm	1000	If(ra == rd) pc = pc + imm
j addr	0010	pc = pc + addr

TABLE – I: Basic Instruction Set

The program code is stored as 16-bit instructions in the IMEM. The processor implements three types of instruction:

Type	15-12	11-9	8-6	5-3	2-0
R (Register)	Opcode	Rd	Ra	Rb	Func
I (Immediate)	Opcode	Rd	Ra	Imm[5:0]	
J (Jump)	Opcode	Don't care		Addr[7-0]	

TABLE – II: Instruction Format

Data Path Components:

Program Counter: One 8-bit register
Register File: Holds 8-bit 8 register
Instruction Memory: 8-bit address input, and outputs 16-bit
Data Memory: 8-bit read/write memory
ALU: Performs arithmetic and logical operations

Control Unit: Generates necessary signals to the data-path. Check single-cycle ARM processor design **in the lecture slides** to design the signals and implement the control unit.

Hint: The signals can be defined as **MemToReg**, **MemWrite**, **RegWrite**, **ALUSrc**, **Branch**, **Jump**…

IMPLEMENTATION

Phase 1: Start by providing a **block diagram** of your design. Show all the **connections** and **signals** explicitly.

Phase 2: Implement your design using **SystemVerilog**.

Phase 3: Simulate your design in **Vivado** by writing a **sample** program and loading it into the instruction memory.

(Optional) Phase 4: Synthesize your code and test the program on **NEXYS 4** FPGA board.

BE CREATIVE!

Do not limit yourself with the instruction set given in Table I and instruction format in Table II. You **should add extra instructions** and if necessary you **should** change the instruction format, as you like.

EVALUATION

- The project is **optional**---students are free to implement.
- The students that will present the project will earn **bonus points** for their final/midterm grades.

PRESENTATION

- The project should be implemented **individually**--- **team work is not allowed**.
- The project control will be on **19 June 2019** in my office.
- Everybody has **5 minutes** to present his/her project.

If you are going to present your project, add your name in the list:

https://doodle.com/poll/gtrtvcrq5dfb7vze
