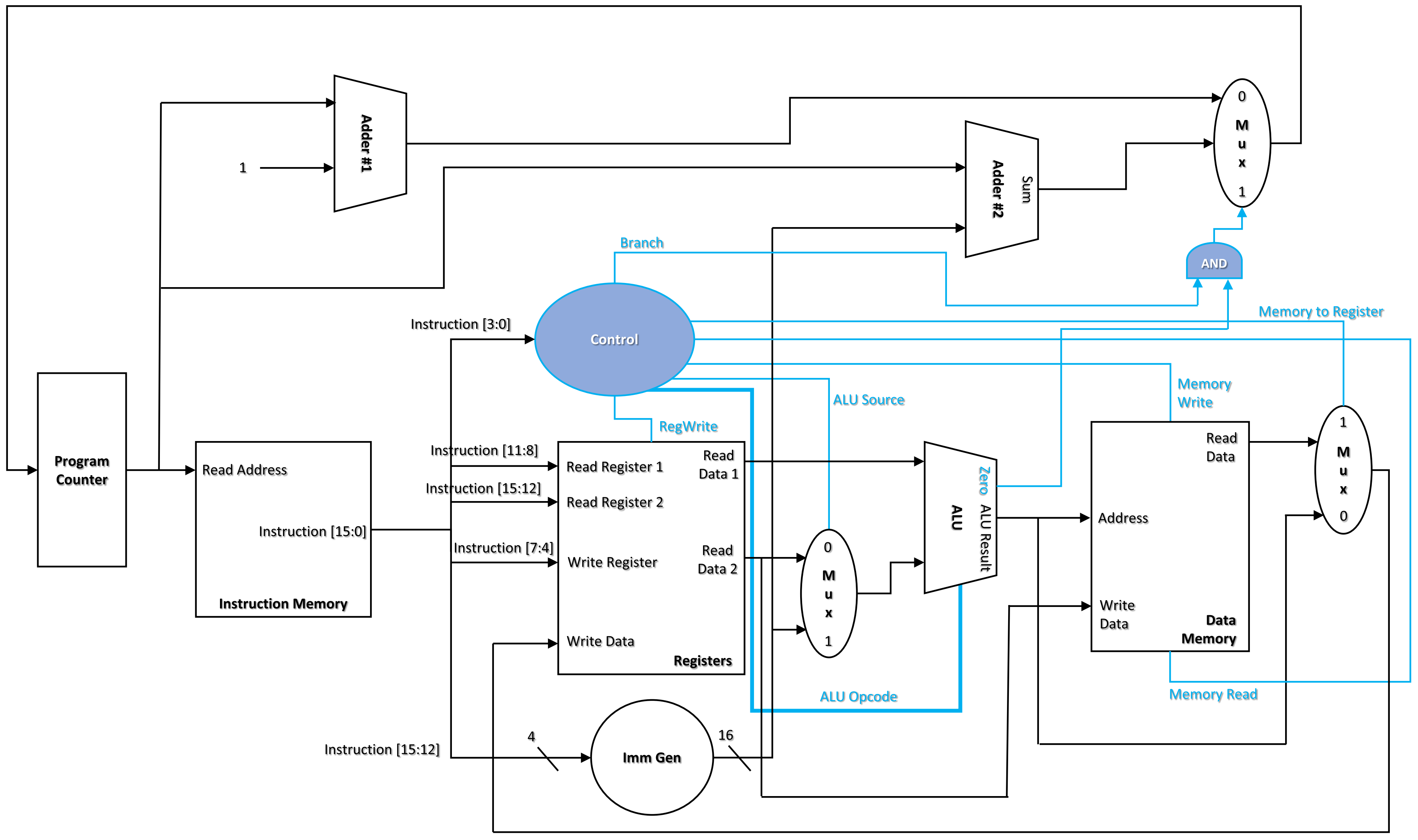
Darrian Johnson
02/24/2021
ELEC 5200 – CPU Design Project Part 2

Datapath:



Components:

Component Name	Inputs	Outputs	Functions
Program Counter	Mux 3 (address of the next instruction)	Address of the current instruction	Keeps track of the current instruction
Instruction Memory	Memory address of the current instruction (PC)	Value of the instruction at the current instruction address	Allows the CPU access to the instruction memory which provides the CPU with the necessary information to complete tasks
Registers	Instruction bits 4-15; Write data from data memory	Data read from two registers	Holds the registers available for use by the CPU and allows the CPU access to them to be read or written to
Immediate Generator (Imm Gen)	Immediate address of current instruction	Immediate address of current instruction	Provides the ALU and Program counter with the immediate address of the current instruction
ALU	Read Register/Data 1 (rs1); Mux 1	Zero-ith flag bit, ALU Result	Takes in the data from a register and either another register or an immediate generator which will do computations which are necessary for the CPU
Control	Opcode of current instruction	Branch; Memory to Register; Memory Read; Memory Write; ALU Source; ALU Opcode; Register Write	Takes in the opcode of the current instruction and determines which output is selected for the CPU to complete its necessary tasks
Data Memory	Current address of the memory to be accessed; Data to be written to memory	Read data from the memory at the provided address	Allows access to the data memory which will store/load data
AND	Branch; Zero-ith flag bit	1 or 0	Enables Mux 3 for output to PC
Adder 1	1; Address of current instruction	Address of current instruction + 1	Sets up the possible next instruction address for the program counter
Adder 2	Memory address of the current instruction (PC); Data from Imm Gen;	PC + Imm. Gen	Used during a branch and/or jump instructions
Mux 1	Read Register Data 2; ALU Source; Imm. Gen Data	Read data 2 (rs2) or Imm Gen. data	Provides the ALU with the necessary current instruction execution which is controlled by the control unit
Mux 2	Memory to Register; Read data from Data Memory; ALU Result	Read data from Data Memory or ALU Result	Provides the Register component with the information necessary to write the required data to a specified register; this is dependent on the current instruction
Mux 3	Control signal branch/jump instruction address; Address of current instruction + Imm. Gen; Address of current instruction + 1	Address of the next instruction	Determines the next address to send to PC; this could be either the next address or an address determined from a branch/jump instruction

Fetching & Executing Instructions:

The user is to assume that all register transfers begin with the Program Counter (PC) sending the initial/current instruction address to the Instruction Memory, which then breaks the instruction down by sending various bits to the Immediate Generator, Registers File, Control Unit, and then back to the Program Counter or current instruction.

Instruction	Opcode	Register Transfers	Type
plus	0000	R[rd] = R[rs1] + R[rs2], PC = PC + 1	R
min	0001	R[rd] = R[rs1] – R[rs2], PC = PC + 1	R
and	0010	R[rd] = R[rs1] & R[rs2], PC = PC + 1	R
or	0011	R[rd] = R[rs1] | R[rs2], PC = PC + 1	R
ldw	0100	R[rd] = M[R[rd][15:12] + imm](15:0), PC = PC + 1	I
stw	0101	R[rd] = M[R[rd] + imm](15:0), PC = PC + 1	I
plusi	0110	R[rd] = R[rd] + imm, PC = PC + 1	I
lui	0111	R[rd] = M[R[rd][15:12] + imm](15:0), PC = PC + imm	B
beq	1000	if(R[rs2] == R[rs1]), then PC = PC + imm, else PC = PC +1	B
bne	1001	if(R[rs2] != R[rs1]), then PC = PC + imm, else PC = PC + 1	B
bgt	1010	if(R[rs2] > R[rs1]), then PC = PC + imm, else PC = PC + 1	B
blt	1011	if(R[rs2] < R[rs1]), then PC = PC + imm, else PC = PC + 1	B
bge	1100	if(R[rs2] >= R[rs1]), then PC = PC + imm, else PC = PC + 1	B
blte	1101	if(R[rs2] <= R[rs1]), then PC = PC + imm, else PC = PC + 1	B
jmp	1110	PC = PC + imm	J
stop	1111	PC = PC	N/A

Control Signals:

ALU Operation	ALU Opcode
Add	00
Subtract	01
AND	10
OR	11

Instruction	Branch	Mem2Reg	MemRead	MemWrite	ALU Source	ALU Opcode	RegWrite
plus	0	0	0	0	0	00	1
min	0	0	0	0	0	01	1
and	0	0	0	0	0	10	1
or	0	0	0	0	0	11	1
ldw	0	1	1	0	1	00	1
stw	0	1	0	1	1	00	0
plusi	0	0	0	0	1	00	1
lui	0	0	0	0	0	xx	0
beq	1	x	0	0	0	01	0
bne	1	x	0	0	0	01	0
bgt	1	x	0	0	0	01	0
blt	1	x	0	0	0	01	0
bge	1	x	0	0	0	01	0
blte	1	x	0	0	0	01	0
jmp	1	1	0	0	x	xx	0
stop	0	0	0	0	0	xx	0

Discussion:

A Single Cycle Datapath was chosen for this project because at the time of this project, it was reviewed in class. It was also the easiest Datapath to understand from the book being that Pipelining is not mentioned until later in chapter 4 of the book.

I added two additional adders rather than running everything through the ALU to assist with the speed of the CPU and some of its adding operations. These additional adders will make address calculations move faster. The multiplexers are used to select outputs while also assisting with the neatness of the schematic.

My registers are positive-edge triggered as this will assist with knowing when an operation is expected to occur. My immediate generator only takes in the immediate bits [15:12] for I, B, and J-type instructions since these are the only types which utilize an immediate/constant value. When determining which ALU Opcode to use for my branch instructions, I realized that all instructions were able to be executed by utilizing the subtract arithmetic and comparing the two 16-bit registers. Once they are compared, the zero-flag will determine how the AND-gate operates, thus giving the proper/expected output.