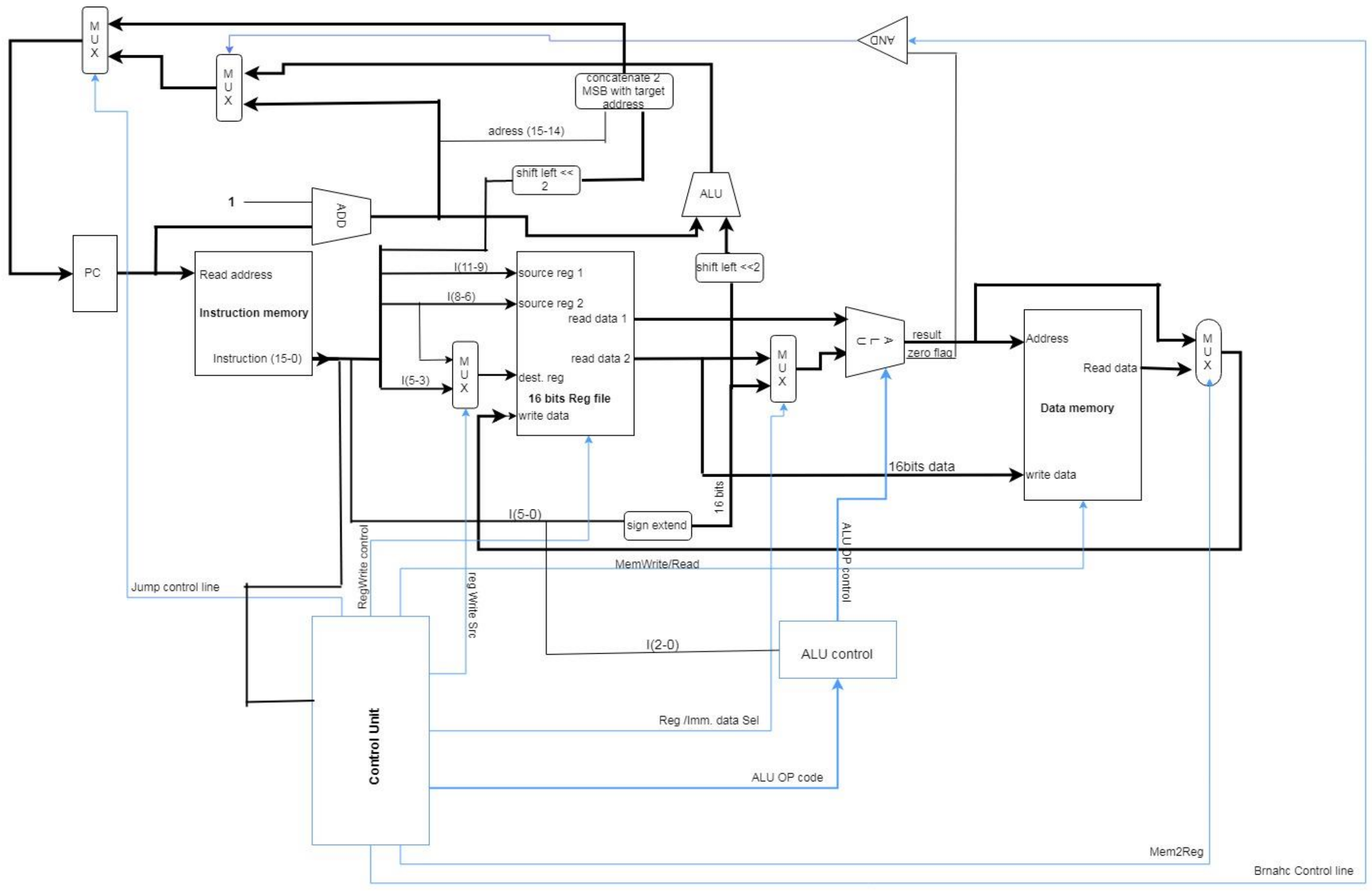
1. Register level block diagram of the datapath and control unit of the CPU design.



2. The block diagram above indicates that the Harvard architecture was adopted, with the instruction in one memory and the data in the other memory. Both memories are included in the block diagram.

3. Description of the inputs, outputs, and function of each component in the datapath.

component	**function**	**Description of Inputs and outputs**
PC (program counter)	This holds the starting address of the program to be fetched. Also holds the address if the next instruction to be fetched and executed (PC+1; PC+1+ branch addr.; PC+1+jump addr.)	Input: 16bits address of the next instruction Output: 16bits address of the instruction to execute
Instruction memory	This contains the 16bits instructions to be executed in the next clock cycle.	Input: the 16 bits address from the PC

	It reads the address from the PC and fetches the instruction from its memory. Places this instruction on its output bus.	Output: the 16 bits instruction to be executed in the next clock cycle.
16bits * 8 Register file	This is the array of registers defined in the ISA. It reads the two source register 3bits address, and the destination address on it input bus, and places the 16bits content in the two source registers on the output bus.	Inputs: the 3bit addresses of source registers 1&2, the 3bit address destination register, 16bit data to store in the destination register, and the register write control signal. Output: the 16bit values ifrom the two source registers.
ALU	The ALU performs arithmetic and logical operation on two 16 bits values at it input and place the result on its output buses (zero/result) if the operation result is 0. The operation executed is determined by the 3bit ALU OP control at its control input.	Inputs: two 16bits data, in which one value can be from a source register or a 16bits immediate value. Also a 3bit ALU OP control to determine the type of arithmetic/logical operation. Output: the ALU places the result on the result bus and sets the zero flag if the result is a 0.
ALU ^	This performs arithmetic operation on two 16bits word addresses; the PC+1 and the sign extended 16bits branch address.	Inputs; the PC+1 address and the sign extended 16bits branch address. Output: the result of the arithmetic operation is placed on its output bus.
ADD	This performs the increment on the 16bits PC address by adding 1.	Inputs: the 16bits PC address and hard wired 1. Output: the PC+1 result.
Data memory	The 1K memory array that holds the 16bits data value to be stored. It reads the address input and writes or reads data to memory or from register.	Input: 16bits memory address, 2bits memory read/write control line, and a 16bits data from register. Output: the data to be read from the memory during a load word execution.
ALU control	This takes the 2bits ALU OP code and the function code to determine the ALU operation to be performed.	Input: 2bits ALU OP code produced from the control unit and 3bits function code . Output: 3bits ALU control code that controls the operation tyoe in the ALU .

Control	Controls the several operations and decisions during the execution cycle of each instruction. Also, decodes the type of instruction to be executed and feeds the ALU control with the ALU OP code.	Input: 4bits Opcode is fed to the control to decode the instruction type in the execution stage. Outputs: the control makes decision on the combination of component and signals required for each instruction type. *ALU OP code*; a 2bit code to determine the instruction type (R-type, branch, memory-reference type). *Mem2Reg;* signal determines whether data is loaded from the data memory or from the ALU output to the destination register. *regWrite Src;* selects the destination register based on the instruction type. *Reg/Imm.data Sel;* selects either 16bits register value or sign extended 16bits immediate value as input to the ALU. *RegWrite control;* controls how data is stored in the destination register. *Branch control line;* determine whether a branch is to be taken or not , *Jump control line;* when asserted, the PC gets loaded with the calculated jump address. *memWrite/Read;* a 2bits control line that determines whether to write to data memory or read from data memory.
AND gate	Perform an AND on both 1bit zero flag and the 1bit branch control signal.	Inputs; 1bit zero flag and 1bit branch control signal. Output; a 1bit signal to select the multiplexer input.

4. Register transfer list in Instruction in ISA

Instruction type	Register transfer list
R-type instruction format	PC->instruction memory->source reg data 1 &2 ->(ALUcontrol) ALU-> dest. Reg.
Load word instruction	PC->instruction memory->source reg data 1 & 16bits immediate offset value ->(ALUcontrol) ALU-> dest. Reg
Store word instruction	PC->instruction memory->source reg data 1 & 16bits immediate offset value -> data memory
Branch instruction	PC->instruction memory->source reg data 1 & 16bits immediate offset value ->PC+ branch addr.
jump	PC->instruction memory->PC+jump addr.

5. Truth table for the control signal in the single cycled datapath

Instruction type	Input: Opcode bits				regWrite Control	Branch control	Jump control	RegWrite Src	Reg/Imm.data Sel	Mem2Reg	Memwrite/Read	ALU OP code
R-type	0	0	0	0	1	X	x	1	1	1	Xx	10
Lw	0	0	1	1	1	X	X	0	0	0	01	00
Sw	0	1	0	0	x	X	X	0	0	X	10	00
Beq	0	1	1	0	X	1	X	X	1	X	Xx	01
J	1	1	0	0	X	X	1	X	X	X	Xx	Xx
Bne	0	1	0	1	X	1	X	x	1	x	xx	11

6. Trade-offs considered during design decision

Cost vs speed: adopting the Harvard memory architecture improves the performance of processor as it solves the von Neumann bottle neck issue of having to access a memory for both instruction and data at almost every clock cycle. This speeds up the CPU in its operation, plus memories are cheaper now, thus it is cost effective in terms of time to use separate memories for instruction and data.

Single cycle: choosing single cycle datapath simplifies the design implementation process as on the long run as it easy to upgrade to multicycle s the project progress over implementing multicycle datapath.

Dedicated components over shared components: decision to use dedicated component for the instruction execution borders on regularity and simplicity of design to forestall component breakdown from shared resource.

Edge triggered register over latching registers: I considered edge triggered registers due to the clock latency in instruction execution and path delays for synchronization.