The single-cycle LEGv8 datapath is shown in Fig. 1 below. The encoding of each LEGv8 instruction is listed in Table 1. The arithmetic and logic unit (ALU) control logic is shown in Table 2. The instruction control logic is shown in Table 3. The translation from instruction opcode to ALU control input is shown in Table 4.

1. For the LEGv8 instruction **LDUR x10, #100**, refer to Figure 1 and tables 1-4 and answer the following questions:
    a. Convert the instruction to binary.
    [2 Marks]
    b. Referring to the opcode table, what format of instruction is this.
    [2 Marks]
    c. What is the Opcode of the instruction?
    [2 Marks]
    d. On the diagram in Figure 1 below, highlight the active datapath used by this instruction.
    [5 Marks]
    e. What are the output settings of the Control component.
    [9 Marks]
    f. What are the output binary codes for the ALU control component.
    [2 Marks]

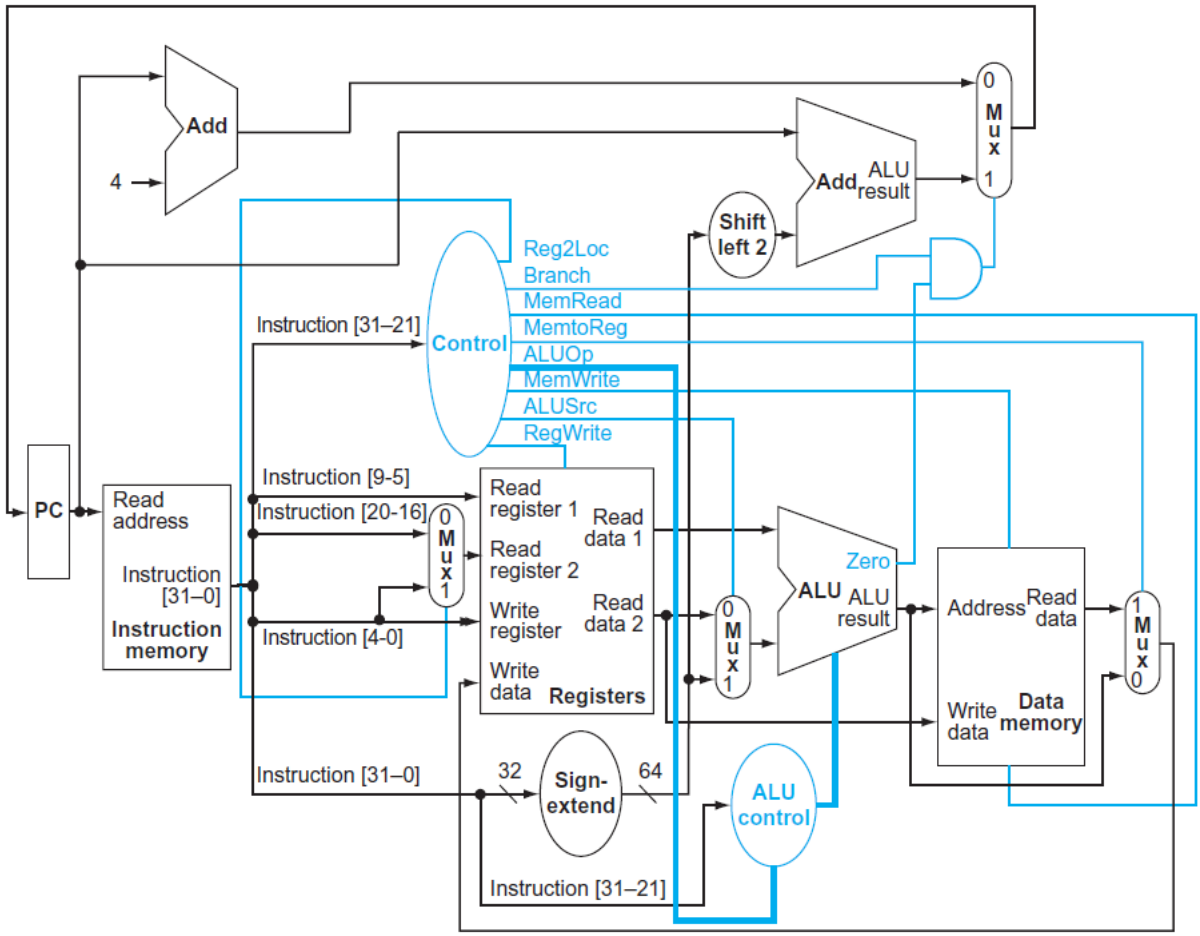

Figure 1: The single-cycle LEGv8 datapath.

Instruction	Format	opcode	Rm	shamt	address	op2	Rn	Rd
ADD (add)	R	1112_{ten}	reg	0	n.a.	n.a.	reg	reg
SUB (subtract)	R	1624_{ten}	reg	0	n.a.	n.a.	reg	reg
ADDI (add immediate)	I	580_{ten}	reg	n.a.	constant	n.a.	reg	n.a.
SUBI (sub immediate)	I	836_{ten}	reg	n.a.	constant	n.a.	reg	n.a.
LDUR (load word)	D	1986_{ten}	reg	n.a.	address	0	reg	n.a.
STUR (store word)	D	1984_{ten}	reg	n.a.	address	0	reg	n.a.

Table 1: LEGv8 Instruction Encoding

Instruction	ALUOp	Instruction operation	Opcode field	Desired ALU action	ALU control input
LDUR	00	load register	XXXXXXXXXXX	add	0010
STUR	00	store register	XXXXXXXXXXX	add	0010
CBZ	01	compare and branch on zero	XXXXXXXXXXX	pass input b	0111
R-type	10	ADD	10001011000	add	0010
R-type	10	SUB	11001011000	subtract	0110
R-type	10	AND	10001010000	AND	0000
R-type	10	ORR	10101010000	OR	0001

Table 2: Control of the ALU control input based on the Opcode field and ALUOp.

Input or output	Signal name	R-format	LDUR	STUR	CBZ
Inputs	I[31]	1	1	1	1
	I[30]	X	1	1	0
	I[29]	X	1	1	1
	I[28]	0	1	1	1
	I[27]	1	1	1	0
	I[26]	0	0	0	1
	I[25]	1	0	0	0
	I[24]	X	0	0	0
	I[23]	0	0	0	X
	I[22]	0	1	0	X
	I[21]	0	0	0	X
Outputs	Reg2Loc	0	X	1	1
	ALUSrc	0	1	1	0
	MemtoReg	0	1	X	X
	RegWrite	1	1	0	0
	MemRead	0	1	0	0
	MemWrite	0	0	1	0
	Branch	0	0	0	1
	ALUOp1	1	0	0	0
	ALUOp0	0	0	0	1

Table 3: The instructional control outputs based on the Opcode field I[31:21] and type of instruction.

ALUOp		Opcode field											Operation
ALUOp1	ALUOp0	I[31]	I[30]	I[29]	I[28]	I[27]	I[26]	I[25]	I[24]	I[23]	I[22]	I[21]	
0	0	X	X	X	X	X	X	X	X	X	X	X	0010
X	1	X	X	X	X	X	X	X	X	X	X	X	0111
1	X	1	0	0	0	1	0	1	1	0	0	0	0010
1	X	1	1	0	0	1	0	1	1	0	0	0	0110
1	X	1	0	0	0	1	0	1	0	0	0	0	0000
1	X	1	0	1	0	1	0	1	0	0	0	0	0001

Table 4: The truth table for the 4-bit ALU control input, labelled Operation in the table.

2. Examine the difficulty of adding a proposed **LWI Rd, Rm(Rn)** "Load With Increment" instruction to the LEGv8 datapath shown in Fig. 1.
   Interpretation: Reg[Rd] = Mem[Reg[Rm] + Reg[Rn]]
   a. What new functional blocks (if any) do we need for this instruction?
   b. Which existing functional blocks (if any) require modification?
   c. Which new datapaths (if any) do we need for this instruction?
   d. What new signals do we need (if any) from the control unit to support this instruction?

3. If we were to translate the datapath in Fig. 1 to a pipelined datapath:
    a. What are the stages we could split the datapath into?
    b. How would we connect each stage of the datapath together?
    c. Draw a block diagram of each stage of your pipelined datapath, include the control circuits required.
    d. The following code is executed on your pipelined datapath:

```
LDUR x9,[x20,#0]
ADDI  x8,x9,#10
STUR x8,[x20,#0]
LDUR x10,[x20,#8]
SUBI x7,x10,#10
STUR x7,[x20,#8]
LDUR x11,[x20,#16]
AND x6,x8,x11
STUR x6,[x20,#16]
```

    Draw a diagram showing how the code is executed, include indications of any stalls in your diagram and the number of cycles the code takes to execute.
    e. Could we modify the code above to remove any stalls?