Q) Show the following instructions going through the pipeline of LEGv8 architecture.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
LDUR X0, (X1, #0)																				
ADD X2, X1, X1																				
ADD X2, X2, X2																				
LDUR X3, (X2, #0)																				
ADD X3, X3, X0																				
ADD X3, X3, X3																				
ADD X2, X0, X0																				
ADD X3, X3, X2																				
STUR X3, (X1, #8)																				

Q) You are asked to design a hazard detection unit. This hazard detection unit will be responsible of detecting if there is a branch hazard. This hazard unit is required to detect the branch hazard as soon as possible and update the PC accordingly. Show your design for this hazard detection unit.

Q)

4.10 When processor designers consider a possible improvement to the processor datapath, the decision usually depends on the cost/performance trade-off. In the following three problems, assume that we are beginning with the datapath from Figure 4.23, the latencies from Exercise 4.7, and the following costs:

I-Mem	Register File	Mux	ALU	Adder	D-Mem	Single Register	Sign extend	Single gate	Control
1000	200	10	100	30	2000	5	100	1	500

Suppose doubling the number of general purpose registers from 32 to 64 would reduce the number of LDUR and STUR instruction by 12%, but increase the latency of the register file from 150 ps to 160 ps and double the cost from 200 to 400. (Use the instruction mix from Exercise 4.8 and ignore the other effects on the ISA discussed in Exercise 2.18.)

4.10.1 [5] <§4.4> What is the speedup achieved by adding this improvement?

4.10.2 [10] <§4.4> Compare the change in performance to the change in cost.

4.10.3 [10] <§4.4> Given the cost/performance ratios you just calculated, describe a situation where it makes sense to add more registers and describe a situation where it doesn't make sense to add more registers.

Q)

4.11 Examine the difficulty of adding a proposed `LWI Rd, Rm(Rn)` ("*L*oad *W*ith *I*ncrement") instruction to LEGv8.

Interpretation: `Reg[Rd]=Mem[Reg[Rm]+Reg[Rn]]`

4.11.1 [5] <§4.4> Which new functional blocks (if any) do we need for this instruction?

4.11.2 [5] <§4.4> Which existing functional blocks (if any) require modification?

4.11.3 [5] <§4.4> Which new data paths (if any) do we need for this instruction?

4.11.4 [5] <§4.4> What new signals do we need (if any) from the control unit to support this instruction?

Q)

4.12 Examine the difficulty of adding a proposed `swap Rd, Rn` instruction to LEGv8.

Interpretation: `Reg[Rd]=Reg[Rn]; Reg[Rn]=Reg[Rd]`

4.12.1 [5] <§4.4> Which new functional blocks (if any) do we need for this instruction?

4.12.2 [10] <§4.4> Which existing functional blocks (if any) require modification?

4.12.3 [5] <§4.4> What new data paths do we need (if any) to support this instruction?

4.12.4 [5] <§4.4> What new signals do we need (if any) from the control unit to support this instruction?

4.12.5 [5] <§4.4> Modify Figure 4.23 to demonstrate an implementation of this new instruction.

Q)

4.15 LDUR is instruction with the longest latency on the CPU from Section 4.4. If we modified LDUR and STUR so that there was no offset (i.e., the address to be loaded from/stored to must be calculated and placed in Rd before calling LDUR/STUR), then no instruction would use both the ALU and Data memory. This would allow us to reduce the clock cycle time. However, it would also increase the number of instructions, because many LDUR and STUR instructions would need to be replaced with LDUR/ADD or STUR/ADD combinations.

4.15.1 [5] <§4.4> What would the new clock cycle time be?

4.15.2 [10] <§4.4> Would a program with the instruction mix presented in Exercise 4.7 run faster or slower on this new CPU? By how much? (For simplicity, assume every LDUR and STUR instruction is replaced with a sequence of two instructions.)

4.15.3 [5] <§4.4> What is the primary factor that influences whether a program will run faster or slower on the new CPU?

4.15.4 [5] <§4.4> Do you consider the original CPU (as shown in Figure 4.23) a better overall design; or do you consider the new CPU a better overall design? Why?

Q)

4.20 [5] <§4.5> Add NOP instructions to the code below so that it will run correctly on a pipeline that does not handle data hazards.

```
ADDI X1, X2, #5
ADD  X3, X1, X2
ADDI X4, X1, #15
ADD  X5, X3, X2
```

Q)

4.22 [5] <§4.5> Consider the fragment of LEGv8 assembly below:

```
STUR X16, [X6, #12]
LDUR X16, [X6, #8]
SUB  X7, X5, X4
CBZ  X7, Label
ADD  X5, X1, X4
SUB  X5, X15, X4
```

Suppose we modify the pipeline so that it has only one memory (that handles both instructions and data). In this case, there will be a structural hazard every time a program needs to fetch an instruction during the same cycle in which another instruction accesses data.

4.22.1 [5] <§4.5> Draw a pipeline diagram to show were the code above will stall.

4.22.2 [5] <§4.5> In general, is it possible to reduce the number of stalls/NOPs resulting from this structural hazard by reordering code?

4.22.3 [5] <§4.5> Must this structural hazard be handled in hardware? We have seen that data hazards can be eliminated by adding NOPs to the code. Can you do the same with this structural hazard? If so, explain how. If not, explain why not.

4.22.4 [5] <§4.5> Approximately how many stalls would you expect this structural hazard to generate in a typical program? (Use the instruction mix from Exercise 4.8.)

Q)

4.28 The importance of having a good branch predictor depends on how often conditional branches are executed. Together with branch predictor accuracy, this will determine how much time is spent stalling due to mispredicted branches. In this exercise, assume that the breakdown of dynamic instructions into various instruction categories is as follows:

R-Type	CBZ/CBNZ	B	LDUR	STUR
40%	25%	5%	25%	5%

Also, assume the following branch predictor accuracies:

Always-Taken	Always-Not-Taken	2-Bit
45%	55%	85%

4.28.1 [10] <§4.8> Stall cycles due to mispredicted branches increase the CPI. What is the extra CPI due to mispredicted branches with the always-taken predictor? Assume that branch outcomes are determined in the ID stage and applied in the EX stage that there are no data hazards, and that no delay slots are used.

4.28.2 [10] <§4.8> Repeat 4.28.1 for the "always-not-taken" predictor.

Q)

4.33 When silicon chips are fabricated, defects in materials (e.g., silicon) and manufacturing errors can result in defective circuits. A very common defect is for one wire to affect the signal in another. This is called a "cross-talk fault". A special class of cross-talk faults is when a signal is connected to a wire that has a constant logical value (e.g., a power supply wire). These faults, where the affected signal always has a logical value of either 0 or 1 are called "stuck-at-0" or "stuck-at-1" faults. The following problems refer to bit 0 of the Write Register input on the register file in Figure 4.23.

4.33.1 [10] <§§4.3, 4.4> Let us assume that processor testing is done by (1) filling the PC, registers, and data and instruction memories with some values (you can choose which values), (2) letting a single instruction execute, then (3) reading the PC, memories, and registers. These values are then examined to determine if a particular fault is present. Can you design a test (values for PC, memories, and registers) that would determine if there is a stuck-at-0 fault on this signal?

Q)

Assume memory is byte addressable and words are 64 bits, unless specified otherwise.

5.1 In this exercise we look at memory locality properties of matrix computation. The following code is written in C, where elements within the same row are stored contiguously. Assume each word is a 64-bit integer.

```
for (I=0; I<8; I++)
  for (J=0; J<8000; J++)
    A[I][J]=B[I][0]+A[J][I];
```

5.1.1 [5] <§5.1> How many 64-bit integers can be stored in a 16-byte cache block?

5.1.2 [5] <§5.1> Which variable references exhibit temporal locality?

5.1.3 [5] <§5.1> Which variable references exhibit spatial locality?

Locality is affected by both the reference order and data layout. The same computation can also be written below in Matlab, which differs from C in that it stores matrix elements within the same column contiguously in memory.

```
for I=1:8
  for J=1:8000
    A(I,J)=B(I,0)+A(J,I);
  end
end
```

5.1.4 [5] <§5.1> Which variable references exhibit temporal locality?

5.1.5 [5] <§5.1> Which variable references exhibit spatial locality?

Q)

5.2 Caches are important to providing a high-performance memory hierarchy to processors. Below is a list of 64-bit memory address references, given as word addresses.

```
0x03, 0xb4, 0x2b, 0x02, 0xbf, 0x58, 0xbe, 0x0e, 0xb5,
0x2c, 0xba, 0xfd
```

5.2.1 [10] <§5.3> For each of these references, identify the binary word address, the tag, and the index given a direct-mapped cache with 16 one-word blocks. Also list whether each reference is a hit or a miss, assuming the cache is initially empty.

5.2.2 [10] <§5.3> For each of these references, identify the binary word address, the tag, the index, and the offset given a direct-mapped cache with two-word blocks and a total size of eight blocks. Also list if each reference is a hit or a miss, assuming the cache is initially empty.

5.2.3 [20] <§§5.3, 5.4> You are asked to optimize a cache design for the given references. There are three direct-mapped cache designs possible, all with a total of eight words of data:

- C1 has 1-word blocks,
- C2 has 2-word blocks, and
- C3 has 4-word blocks.

Q)

5.5 For a direct-mapped cache design with a 64-bit address, the following bits of the address are used to access the cache.

Tag	Index	Offset
63–10	9–5	4–0

5.5.1 [5] <§5.3> What is the cache block size (in words)?

5.5.2 [5] <§5.3> How many blocks does the cache have?

5.5.3 [5] <§5.3> What is the ratio between total bits required for such a cache implementation over the data storage bits?

Beginning from power on, the following byte-addressed cache references are recorded.

Address												
Hex	00	04	10	84	E8	A0	400	1E	8C	C1C	B4	884
Dec	0	4	16	132	232	160	1024	30	140	3100	180	2180

5.5.4 [20] <§5.3> For each reference, list (1) its tag, index, and offset, (2) whether it is a hit or a miss, and (3) which bytes were replaced (if any).

5.5.5 [5] <§5.3> What is the hit ratio?

5.5.6 [5] <§5.3> List the final state of the cache, with each valid entry represented as a record of <index, tag, data>. For example,

```
<0, 3, Mem[0xC00]-Mem[0xC1F]>
```

Q)

5.7 Consider the following program and cache behaviors.

Data Reads per 1000 Instructions	Data Writes per 1000 Instructions	Instruction Cache Miss Rate	Data Cache Miss Rate	Block Size (bytes)
250	100	0.30%	2%	64

5.7.1 [10] <§§5.3, 5.8> Suppose a CPU with a write-through, write-allocate cache achieves a CPI of 2. What are the read and write bandwidths (measured by bytes per cycle) between RAM and the cache? (Assume each miss generates a request for one block.)

Q)

5.9 Cache block size (B) can affect both miss rate and miss latency. Assuming a machine with a base CPI of 1, and an average of 1.35 references (both instruction and data) per instruction, find the block size that minimizes the total miss latency given the following miss rates for various block sizes.

8: 4%	16: 3%	32: 2%	64: 1.5%	128: 1%

5.9.1 [10] <§5.3> What is the optimal block size for a miss latency of 20 × B cycles?

5.9.2 [10] <§5.3> What is the optimal block size for a miss latency of 24 + B cycles?

5.9.3 [10] <§5.3> For constant miss latency, what is the optimal block size?

Q)

5.11 This exercise examines the effect of different cache designs, specifically comparing associative caches to the direct-mapped caches from Section 5.4. For these exercises, refer to the sequence of word address shown below.

```
0x03, 0xb4, 0x2b, 0x02, 0xbe, 0x58, 0xbf, 0x0e, 0x1f,
0xb5, 0xbf, 0xba, 0x2e, 0xce
```

5.11.1 [10] <§5.4> Sketch the organization of a three-way set associative cache with two-word blocks and a total size of 48 words. Your sketch should have a style similar to Figure 5.18, but clearly show the width of the tag and data fields.

5.11.2 [10] <§5.4> Trace the behavior of the cache from Exercise 5.11.1. Assume a true LRU replacement policy. For each reference, identify

- the binary word address,
- the tag,
- the index,
- the offset
- whether the reference is a hit or a miss, and
- which tags are in each way of the cache after the reference has been handled.

Q)

5.12 Multilevel caching is an important technique to overcome the limited amount of space that a first-level cache can provide while still maintaining its speed. Consider a processor with the following parameters:

Base CPI, No Memory Stalls	Processor Speed	Main Memory Access Time	First-Level Cache Miss Rate per Instruction**	Second-Level Cache, Direct-Mapped Speed	Miss Rate with Second-Level Cache, Direct-Mapped	Second-Level Cache, Eight-Way Set Associative Speed	Miss Rate with Second-Level Cache, Eight-Way Set Associative
1.5	2 GHz	100 ns	7%	12 cycles	3.5%	28 cycles	1.5%

**First Level Cache miss rate is per instruction. Assume the total number of L1 cache misses (instruction and data combined) is equal to 7% of the number of instructions.

5.12.1 [10] <§5.4> Calculate the CPI for the processor in the table using: 1) only a first-level cache, 2) a second-level direct-mapped cache, and 3) a second-level eight-way set associative cache. How do these numbers change if main memory access time doubles? (Give each change as both an absolute CPI and a percent change.) Notice the extent to which an L2 cache can hide the effects of a slow memory.

Q)

5.15 For a high-performance system such as a B-tree index for a database, the page size is determined mainly by the data size and disk performance. Assume that, on average, a B-tree index page is 70% full with fix-sized entries. The utility of a page is its B-tree depth, calculated as $\log_2$(entries). The following table shows that for 16-byte entries, and a 10-year-old disk with a 10 ms latency and 10 MB/s transfer rate, the optimal page size is 16 K.

Page Size (KiB)	Page Utility or B-Tree Depth (Number of Disk Accesses Saved)	Index Page Access Cost (ms)	Utility/Cost
2	6.49 (or $\log_2(2048/16 \times 0.7)$)	10.2	0.64
4	7.49	10.4	0.72
8	8.49	10.8	0.79
16	9.49	11.6	0.82
32	10.49	13.2	0.79
64	11.49	16.4	0.70
128	12.49	22.8	0.55
256	13.49	35.6	0.38

5.15.1 [10] <§5.7> What is the best page size if entries now become 128 bytes?

5.15.2 [10] <§5.7> Based on Exercise 5.15.1, what is the best page size if pages are half full?