BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI
K. K. BIRLA Goa Campus
First Semester 2017-2018
CS F342 Computer Architecture
Course Project

INSTRUCTIONS

1. The project weightage is 10**%** and the deadline is **22nd November 2017**.
2. This is a team project and the list of team members is available in course page.
3. You are not allowed to take help in terms of suggestions and code from other teams.
4. You are not allowed to use somebody else's code which includes code from internet.
5. If found copied, all members of the team will get -30 marks (will subtract 30 marks from their lab marks). Please see section 4.2 and 4.3 of your course handout for further details. [You are allowed to opt out from the project. In that case you are not considered for evaluation]
6. The project will be evaluated according to the following criteria:

STEPS TO FOLLOW FOR GETTING PROJECT ASSIGNED

1. Each team should decide the preference order (You should include all the 30 preferences).
2. One person from the team should consolidate all the 30 preferences and update. [Make sure you are updating your preferences before **08th November 2017**]. [Make sure ONLY ONE person from a team is doing it].
3. Allocation of project is on First Come First Serve basis. No default project allocation is available – i.e. if you are not updated preference before 08th November 2017, you are not going to do the project.

Problem Statement:-

Design VLIW architecture with given* instruction set and design the Cache Memory with the given* specifications.
(* - refer your respective question after allotment)

Instructions:-

- Your final submission should include:
  1. Verilog implementation of the Project
  2. Diagram showing the entire architecture (submit in chart sheet)
  3. During demonstration, you should simulate each instruction separately and combined.
  4. README.txt with the following details:
     - #bits used for Offset, Index and Tag ***(assume 32-bit physical address)***
     - Total Cache Size (You should include the size of prediction circuit and halt tag array cache for Way-Prediction cache and Way-Halting cache cache respectively)
     - Based on the test case, Number of Cache Hits and CacheMiss

Group number, names of all the members and individual contributions for the project

Question 1:

Cache Specifications	
Cache Size	512B
Cache Line Size	8B
Associativity	4
Write Policy	Write Back
Replacement policy	FIFO
Cache Type	Way Halting

Instruction Type	Instruction	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R type		**funct7**							**Im[4:0]**					**rs1**					**funct3**			**rd**					**opcode**						
	slli	0	0	0	0	0	0	0	shamt[4:0]					rs1					0	0	1	rd					0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] << shamt																																
		funct7							**rs2**					**rs1**					**funct3**			**rd**					**opcode**						
	add	0	0	0	0	0	0	0	rs2					rs1					0	0	0	rd					0	1	1	0	0	1	1
	Reg[rd] = Reg[rs1] + Reg[rs2]																																
I type		**Imm[11:0]**												**rs1**					**funct3**			**rd**					**opcode**						
	jalr	Imm[11:0]												rs1					0	0	0	rd					1	1	0	0	1	1	1
	Reg[rd] = PC + 3 (next PC), PC = sExt({Imm[11:0],0}) + Reg[rs1]																																
	xori	Imm[11:0]												rs1					1	0	0	rd					0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] ^ sExt(Imm[11:0])																																
	lh	Imm[11:0]												rs1					0	0	1	rd					0	0	0	0	0	1	1
	Reg[rd] = sExt(Mem[[Reg[rs1] + sExt(Imm[11:0])] (16 bits))																																
B type		**Imm**							**rs2**					**rs1**					**funct3**			**Imm**					**opcode**						
	BEQ	imm[12]	imm[10:5]						rs2					rs1					0	0	0	imm[4:1]				imm[11]	1	1	0	0	0	1	1
	PC(target) <= sext12to32(imm[12:0]) , if rs1==rs2																																

Instruction Type	Instruction	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CI type		funct3			Imm[5]	rd					Imm[4:0]					op	
	C.LI	0	1	0	Imm[5]	rd (≠ 0)					Imm[4:0]					0	1
	Reg[rd] = sExt(Imm[5:0])																
CR type		funct6						rd/rs1			funct		rs2			op	
	C.AND	1	0	0	0	1	1	rd/rs1			1	1	rs2			0	1
	Reg[rd] = Reg[rs1] & Reg[rs2]																
CS type		funct3			Imm[5:3]			rs1			Imm[2|6]		rs2			op	
	C.SW	1	1	0	Imm[5:3]			rs1			Imm[2|6]		rs2			0	0
	Mem[Reg[rs1] + {15'b0,Imm[6:2],2'b0}] = Reg[rs2]																

Question 2:

Cache Specifications	
Cache Size	512B
Cache Line Size	8B
Associativity	4
Write Policy	Write Back
Replacement policy	FIFO
Cache Type	Way Prediction

Instruction Type	Instruction	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R type		**funct7**							**Im[4:0]**					**rs1**					**funct3**			**rd**				**opcode**						
	srli	0	0	0	0	0	0	0	shamt[4:0]					rs1					1	0	1	rd				0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] >> shamt																																
		funct7							**rs2**					**rs1**					**funct3**			**rd**				**opcode**						
	xor	0	0	0	0	0	0	0	rs2					rs1					1	0	0	rd				0	1	1	0	0	1	1
	Reg[rd] = Reg[rs1] ^ Reg[rs2]																																
I type		**Imm[11:0]**												**rs1**					**funct3**			**rd**				**opcode**						
	addi	Imm[11:0]												rs1					0	0	0	rd				0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] + sExt(Imm[11:0])																																
	jalr	Imm[11:0]												rs1					0	0	0	rd				1	1	0	0	1	1	1
	Reg[rd] = PC + 3 (next PC), PC = sExt({Imm[11:0],0}) + Reg[rs1]																																
	slti	Imm[11:0]												rs1					0	1	0	rd				0	0	1	0	0	1	1
	Reg[rd] = (Reg[rs1] < sExt(Imm[11:0]))?1:0																																
S type		**Imm[11:5]**							**rs2**					**rs1**					**funct3**			**Imm[4:0]**				**opcode**						
	sw	Imm[11:5]							rs2					rs1					0	0	1	Imm[4:0]				0	1	0	0	0	1	1
	Mem[Reg[rs1] + sExt(Imm[11:0])] = Reg[rs2]																																

Instruction Type	Instruction	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CI type		**funct3**			**Imm[5]**	**rd**					**Imm[4:0]**					**op**	
	C.LUI	0	1	1	Imm[5]	rd (≠ 0)					Imm[4:0]					0	1
	Reg[rd] = sExt({Imm{5:0],12'b0})																
CB type		**funct3**			**Imm[8|4:3]**			**rs1**			**Imm[7:6|2:1|5]**					**op**	
	C.BNEZ	1	1	1	Imm[8|4:3]			rs1			Imm[7:6|2:1|5]					0	1
	if(Reg[rs1] != 0) PC = PC + sExt(Imm)																
CL type		**funct3**			**Imm[5:3]**			**rs1**			**Imm[2|6]**		**rd**			**op**	
	C.LW	0	1	0	Imm[5:3]			rs1			Imm[2|6]		rd			0	0
	Reg[rd] = Mem[Reg[rs1] + {15'b0,Imm[6:2],2'b0}]																

Question 3:

Cache Specifications	
Cache Size	512B
Cache Line Size	8B
Associativity	4
Write Policy	Write Back
Replacement policy	LRU counter
Cache Type	Way Halting

Instruction Type	Instruction	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R type		**funct7**							**Im[4:0]**					**rs1**					**funct3**			**rd**				**opcode**						
	srai	0	1	0	0	0	0	0	shamt[4:0]					rs1					1	0	1	rd				0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] >>> shamt																																
		funct7							**rs2**					**rs1**					**funct3**			**rd**				**opcode**						
	sub	0	1	0	0	0	0	0	rs2					rs1					0	0	0	rd				0	1	1	0	0	1	1
	Reg[rd] = Reg[rs1] - Reg[rs2]																																
I type		**Imm[11:0]**												**rs1**					**funct3**			**rd**				**opcode**						
	addi	Imm[11:0]												rs1					0	0	0	rd				0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] + sExt(Imm[11:0])																																
	jalr	Imm[11:0]												rs1					0	0	0	rd				1	1	0	0	1	1	1
	Reg[rd] = PC + 3 (next PC), PC = sExt({Imm[11:0],0}) + Reg[rs1]																																
	sltiu	Imm[11:0]												rs1					0	1	1	rd				0	0	1	0	0	1	1
	Reg[rd] = (Reg[rs1] <{20'b0,Imm[11:0]})?1:0																																
B type		**Imm**							**rs2**					**rs1**					**funct3**			**Imm**				**opcode**						
	BLT	imm[12]	imm[10:5]						rs2					rs1					1	0	0	imm[4:1]			imm[11]	1	1	0	0	0	1	1
	PC(target) <= sext12to32(imm[12:0]) , if rs1<rs2																																

Instruction Type	Instruction	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CR type		**funct4**				**rd/rs1**					**rs2**					**op**	
	C.MV	1	0	0	0	rd (≠ 0)					rs2 (≠ 0)					1	0
	Reg[rd] = Reg[rs2]																
CL type		**funct3**			**Imm[5:3]**			**rs1**			**Imm[2|6]**		**rd**			**op**	
	C.LW	0	1	0	Imm[5:3]			rs1			Imm[2|6]		rd			0	0
	Reg[rd] = Mem[Reg[rs1] + {15'b0,Imm[6:2],2'b0}]																
CS type		**funct3**			**Imm[5:3]**			**rs1**			**Imm[2|6]**		**rs2**			**op**	
	C.SW	1	1	0	Imm[5:3]			rs1			Imm[2|6]		rs2			0	0
	Mem[Reg[rs1] + {15'b0,Imm[6:2],2'b0}] = Reg[rs2]																

Question 4:

Cache Specifications	
Cache Size	512B
Cache Line Size	8B
Associativity	4
Write Policy	Write Through
Replacement policy	Pseudo LRU
Cache Type	Way Prediction

Instruction Type	Instruction	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R type		**funct7**							**Im[4:0]**					**rs1**					**funct3**			**rd**				**opcode**							
	srli	0	0	0	0	0	0	0	shamt[4:0]					rs1					1	0	1	rd				0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] >> shamt																																
		funct7							**rs2**					**rs1**					**funct3**			**rd**				**opcode**							
	xor	0	0	0	0	0	0	0	rs2					rs1					1	0	0	rd				0	1	1	0	0	1	1
	Reg[rd] = Reg[rs1] ^ Reg[rs2]																																
I type		**Imm[11:0]**												**rs1**					**funct3**			**rd**				**opcode**							
	jalr	Imm[11:0]												rs1					0	0	0	rd				1	1	0	0	1	1	1
	Reg[rd] = PC + 3 (next PC), PC = sExt({Imm[11:0],0}) + Reg[rs1]																																
	addi	Imm[11:0]												rs1					0	0	0	rd				0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] + sExt(Imm[11:0])																																
S type		**Imm[11:5]**							**rs2**					**rs1**					**funct3**			**Imm[4:0]**				**opcode**							
	sw	Imm[11:5]							rs2					rs1					0	0	1	Imm[4:0]				0	1	0	0	0	1	1
	Mem[Reg[rs1] + sExt(Imm[11:0])] = Reg[rs2]																																
U type		**Imm**																				**rd**				**opcode**							
	AUIPC	imm[31:12]																				rd				0	0	1	0	1	1	1
	Reg[rd] = PC + {imm[31:12],12'd0}																																

Instruction Type	Instruction	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CR type		funct4				rd/rs1					rs2					op	
	C.MV	1	0	0	0	rd (≠ 0)					rs2 (≠ 0)					1	0
	Reg[rd] = Reg[rs2]																
CL type		funct3			Imm[5:3]			rs1			Imm[2|6]		rd			op	
	C.LW	0	1	0	Imm[5:3]			rs1			Imm[2|6]		rd			0	0
	Reg[rd] = Mem[Reg[rs1] + {15'b0,Imm[6:2],2'b0}]																
CB type		funct3			Imm[8|4:3]			rs1			Imm[7:6|2:1|5]					op	
	C.BEQZ	1	1	0	Imm[8|4:3]			rs1			Imm[7:6|2:1|5]					0	1
	if(Reg[rs1] == 0) PC = PC + sExt(Imm)																

Question 5:

Cache Specifications	
Cache Size	512B
Cache Line Size	8B
Associativity	4
Write Policy	Write Back
Replacement policy	LRU counter
Cache Type	Way Prediction

Instruction Type	Instruction	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R type		funct7							Im[4:0]					rs1					funct3			rd				opcode						
	slli	0	0	0	0	0	0	0	shamt[4:0]					rs1					0	0	1	rd				0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] << shamt																																
		funct7							rs2					rs1					funct3			rd				opcode						
	sltu	0	0	0	0	0	0	0	rs2					rs1					0	1	1	rd				0	1	1	0	0	1	1
	Reg[rd] = (unsigned(Reg[rs1]) < unsigned(Reg[rs2]))?1:0																																
I type		Imm[11:0]												rs1					funct3			rd				opcode						
	addi	Imm[11:0]												rs1					0	0	0	rd				0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] + sExt(Imm[11:0])																																
	lh	Imm[11:0]												rs1					0	0	1	rd				0	0	0	0	0	1	1
	Reg[rd] = sExt(Mem[[Reg[rs1] + sExt(Imm[11:0])] (16 bits))																																
U type		Imm																				rd				opcode						
	LUI	imm[31:12]																				rd				0	1	1	0	1	1	1
	Reg[rd] = {imm[31:12],12'd0}																																
B type		Imm							rs2					rs1					funct3			Imm				opcode						
	BLTU	imm[12]	imm[10:5]						rs2					rs1					1	1	0	imm[4:1]			imm[11]	1	1	0	0	0	1	1
	PC(target) <= sext12to32(imm[12:0]) , if rs1<rs2 (unsigned comparision)																																

Instruction Type	Instruction	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CR type		funct4				rd/rs1					rs2					op	
	C.MV	1	0	0	0	rd (≠ 0)					rs2 (≠ 0)					1	0
	Reg[rd] = Reg[rs2]																
CS type		funct3			Imm[5:3]			rs1			Imm[2|6]		rs2			op	
	C.SW	1	1	0	Imm[5:3]			rs1			Imm[2|6]		rs2			0	0
	Mem[Reg[rs1] + {15'b0,Imm[6:2],2'b0}] = Reg[rs2]																
CJ type		funct3			Imm[10:0]											op	
	C.JALR	0	0	1	Imm[10:0]											0	1
	Reg[x1] = PC + 3 (new PC), PC = PC + sExt(Imm)																

Question 6:

Cache Specifications	
Cache Size	512B
Cache Line Size	8B
Associativity	4
Write Policy	Write Back
Replacement policy	Pseudo LRU
Cache Type	Way Halting

Instruction Type	Instruction	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R type		funct7							Im[4:0]					rs1					funct3			rd				opcode						
	srai	0	1	0	0	0	0	0	shamt[4:0]					rs1					1	0	1	rd				0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] >>> shamt																																
		funct7							rs2					rs1					funct3			rd				opcode						
	srl	0	0	0	0	0	0	0	rs2					rs1					1	0	1	rd				0	1	1	0	0	1	1
	Reg[rd] = Reg[rs1] >> Reg[rs2][4:0]																																
I type		Imm[11:0]												rs1					funct3			rd				opcode						
	addi	Imm[11:0]												rs1					0	0	0	rd				0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] + sExt(Imm[11:0])																																
	ori	Imm[11:0]												rs1					1	1	0	rd				0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] | sExt(Imm[11:0])																																
S type		Imm[11:5]							rs2					rs1					funct3			Imm[4:0]				opcode						
	sb	Imm[11:5]							rs2					rs1					0	0	0	Imm[4:0]				0	1	0	0	0	1	1
	Mem[Reg[rs1] + sExt(Imm[11:0])] = Reg[rs2][7:0]																																
U type		Imm																				rd				opcode						
	AUIPC	imm[31:12]																				rd				0	0	1	0	1	1	1
	Reg[rd] = PC + {imm[31:12],12'd0}																																

Instruction Type	Instruction	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CR type		funct3			Imm[5]	rd/rs1					Imm[4:0]					op	
	C.JALR	1	0	0	1	rs1 (≠ 0)					0	0	0	0	0	1	0
	Reg[1] = PC + 3 (new PC), PC = Reg[rs1]																
CL type		funct3			Imm[5:3]			rs1			Imm[2|6]		rd			op	
	C.LW	0	1	0	Imm[5:3]			rs1			Imm[2|6]		rd			0	0
	Reg[rd] = Mem[Reg[rs1] + {15'b0,Imm[6:2],2'b0}]																
CB type		funct3			Imm[8|4:3]			rs1			Imm[7:6|2:1|5]					op	
	C.BNEZ	1	1	1	Imm[8|4:3]			rs1			Imm[7:6|2:1|5]					0	1
	if(Reg[rs1] != 0) PC = PC + sExt(Imm)																

Question 7:

Cache Specifications	
Cache Size	512B
Cache Line Size	8B
Associativity	4
Write Policy	Write Back
Replacement policy	Pseudo LRU
Cache Type	Way Prediction

Instruction Type	Instruction	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R type		**funct7**							**Im[4:0]**					**rs1**					**funct3**			**rd**				**opcode**						
	slli	0	0	0	0	0	0	0	shamt[4:0]					rs1					0	0	1	rd				0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] << shamt																																
		funct7							**rs2**					**rs1**					**funct3**			**rd**				**opcode**						
	or	0	0	0	0	0	0	0	rs2					rs1					1	1	0	rd				0	1	1	0	0	1	1
	Reg[rd] = Reg[rs1] | Reg[rs2]																																
I type		**Imm[11:0]**												**rs1**					**funct3**			**rd**				**opcode**						
	lh	Imm[11:0]												rs1					0	0	1	rd				0	0	0	0	0	1	1
	Reg[rd] = sExt(Mem[[Reg[rs1] + sExt(Imm[11:0])] (16 bits))																																
	xori	Imm[11:0]												rs1					1	0	0	rd				0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] ^ sExt(Imm[11:0])																																
	jalr	Imm[11:0]												rs1					0	0	0	rd				1	1	0	0	1	1	1
	Reg[rd] = PC + 3 (next PC), PC = sExt({Imm[11:0],0}) + Reg[rs1]																																
U type		**Imm**																				**rd**				**opcode**						
	LUI	imm[31:12]																				rd				0	1	1	0	1	1	1
	Reg[rd] = {imm[31:12],12'd0}																																

Instruction Type	Instruction	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CR type		**funct3**			**Imm[5]**	**rd/rs1**					**Imm[4:0]**					**op**	
	C.ADDI	0	0	0	Imm[5]	rs1/rd (≠ 0)					Imm[4:0]					0	1
	Reg[rd] = Reg[rs1] + sExt(Imm[5:0])																
CB type		**funct3**			**Imm[8|4:3]**			**rs1**			**Imm[7:6|2:1|5]**					**op**	
	C.BEQZ	1	1	0	Imm[8|4:3]			rs1			Imm[7:6|2:1|5]					0	1
	if(Reg[rs1] == 0) PC = PC + sExt(Imm)																
CS type		**funct3**			**Imm[5:3]**			**rs1**			**Imm[2|6]**		**rs2**			**op**	
	C.SW	1	1	0	Imm[5:3]			rs1			Imm[2|6]		rs2			0	0
	Mem[Reg[rs1] + {15'b0,Imm[6:2],2'b0}] = Reg[rs2]																

Question 8:

Cache Specifications	
Cache Size	512B
Cache Line Size	8B
Associativity	4
Write Policy	Write Back
Replacement policy	Random
Cache Type	Way Halting

Instruction Type	Instruction	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R type		**funct7**							**Im[4:0]**					**rs1**					**funct3**			**rd**				**opcode**						
	srai	0	1	0	0	0	0	0	shamt[4:0]					rs1					1	0	1	rd				0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] >>> shamt																																
		funct7							**rs2**					**rs1**					**funct3**			**rd**				**opcode**						
	add	0	0	0	0	0	0	0	rs2					rs1					0	0	0	rd				0	1	1	0	0	1	1
	Reg[rd] = Reg[rs1] + Reg[rs2]																																
I type		**Imm[11:0]**												**rs1**					**funct3**			**rd**				**opcode**						
	jalr	Imm[11:0]												rs1					0	0	0	rd				1	1	0	0	1	1	1
	Reg[rd] = PC + 3 (next PC), PC = sExt({Imm[11:0],0}) + Reg[rs1]																																
	andi	Imm[11:0]												rs1					1	1	0	rd				0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] & sExt(Imm[11:0])																																
S type		**Imm[11:5]**							**rs2**					**rs1**					**funct3**			**Imm[4:0]**				**opcode**						
	sh	Imm[11:5]							rs2					rs1					0	0	1	Imm[4:0]				0	1	0	0	0	1	1
	Mem[Reg[rs1] + sExt(Imm[11:0])] = Reg[rs2][15:0]																																
B type		**Imm**							**rs2**					**rs1**					**funct3**			**Imm**				**opcode**						
	BGE	imm[12]	imm[10:5]						rs2					rs1					1	0	1	imm[4:1]			imm[11]	1	1	0	0	0	1	1
	PC(target) <= sext12to32(imm[12:0]) , if rs1>rs2																																

Instruction Type	Instruction	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CR type		funct4				rd/rs1					rs2					op	
	C.MV	1	0	0	0	rd (≠ 0)					rs2 (≠ 0)					1	0
	Reg[rd] = Reg[rs2]																
CI type		funct3			Imm[5]	rd					Imm[4:0]					op	
	C.LI	0	1	0	Imm[5]	rd (≠ 0)					Imm[4:0]					0	1
	Reg[rd] = sExt(Imm[5:0])																
CL type		funct3			Imm[5:3]			rs1			Imm[2|6]		rd			op	
	C.LW	0	1	0	Imm[5:3]			rs1			Imm[2|6]		rd			0	0
	Reg[rd] = Mem[Reg[rs1] + {15'b0,Imm[6:2],2'b0}]																

Question 9:

Cache Specifications	
Cache Size	512B
Cache Line Size	8B
Associativity	4
Write Policy	Write Back
Replacement policy	Random
Cache Type	Way Prediction

Instruction Type	Instruction	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R type		**funct7**							**Im[4:0]**					**rs1**					**funct3**			**rd**				**opcode**						
	srli	0	0	0	0	0	0	0	shamt[4:0]					rs1					1	0	1	rd				0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] >> shamt																																
		funct7							**rs2**					**rs1**					**funct3**			**rd**				**opcode**						
	sltu	0	0	0	0	0	0	0	rs2					rs1					0	1	1	rd				0	1	1	0	0	1	1
	Reg[rd] = (unsigned(Reg[rs1]) < unsigned(Reg[rs2]))?1:0																																
I type		**Imm[11:0]**												**rs1**					**funct3**			**rd**				**opcode**						
	addi	Imm[11:0]												rs1					0	0	0	rd				0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] + sExt(Imm[11:0])																																
	lbu	Imm[11:0]												rs1					1	0	0	rd				0	0	0	0	0	1	1
	Reg[rd] = {24'b0,Mem[[Reg[rs1] + sExt(Imm[11:0])] (8 bits)}																																
	xori	Imm[11:0]												rs1					1	0	0	rd				0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] ^ sExt(Imm[11:0])																																
U type		**Imm**																				**rd**				**opcode**						
	AUIPC	imm[31:12]																				rd				0	0	1	0	1	1	1
	Reg[rd] = PC + {imm[31:12],12'd0}																																

Instruction Type	Instruction	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CR type		funct4				rd/rs1					rs2					op	
	C.JALR	1	0	0	1	rs1 (≠ 0)					0	0	0	0	0	1	0
	Reg[1] = PC + 3 (new PC), PC = Reg[rs1]																
CB type		funct3			Imm[8|4:3]			rs1			Imm[7:6|2:1|5]					op	
	C.BEQZ	1	1	0	Imm[8|4:3]			rs1			Imm[7:6|2:1|5]					0	1
	if(Reg[rs1] == 0) PC = PC + sExt(Imm)																
CS type		funct3			Imm[5:3]			rs1			Imm[2|6]		rs2			op	
	C.SW	1	1	0	Imm[5:3]			rs1			Imm[2|6]		rs2			0	0
	Mem[Reg[rs1] + {15'b0,Imm[6:2],2'b0}] = Reg[rs2]																

Question 10:

Cache Specifications	
Cache Size	512B
Cache Line Size	8B
Associativity	4
Write Policy	Write Through
Replacement policy	LRU Counter
Cache Type	Way Halting

Instruction Type	Instruction	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R type		**funct7**							**Im[4:0]**					**rs1**					**funct3**			**rd**				**opcode**						
	slli	0	0	0	0	0	0	0	shamt[4:0]					rs1					0	0	1	rd				0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] << shamt																																
		funct7							**rs2**					**rs1**					**funct3**			**rd**				**opcode**						
	or	0	0	0	0	0	0	0	rs2					rs1					1	1	0	rd				0	1	1	0	0	1	1
	Reg[rd] = Reg[rs1] | Reg[rs2]																																
I type		**Imm[11:0]**												**rs1**					**funct3**			**rd**				**opcode**						
	jalr	Imm[11:0]												rs1					0	0	0	rd				1	1	0	0	1	1	1
	Reg[rd] = PC + 3 (next PC), PC = sExt({Imm[11:0],0}) + Reg[rs1]																																
	sltiu	Imm[11:0]												rs1					0	1	1	rd				0	0	1	0	0	1	1
	Reg[rd] = (Reg[rs1] <{20'b0,Imm[11:0]})?1:0																																
S type		**Imm[11:5]**							**rs2**					**rs1**					**funct3**			**Imm[4:0]**				**opcode**						
	sb	Imm[11:5]							rs2					rs1					0	0	0	Imm[4:0]				0	1	0	0	0	1	1
	Mem[Reg[rs1] + sExt(Imm[11:0])] = Reg[rs2][7:0]																																
J type		**Imm[19:0]**																				**rd**				**opcode**						
	JAL	imm[10:0]																				rd				1	1	0	1	1	1	1
	PC(target) <= sext20to32(imm[20:0]) + PC, rd = PC + 3 (new PC)																																

Instruction Type	Instruction	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CR type		funct4				rd/rs1					rs2					op	
	C.MV	1	0	0	0	rd (≠ 0)					rs2 (≠ 0)					1	0
	Reg[rd] = Reg[rs2]																
CL type		funct3			Imm[5:3]			rs1			Imm[2|6]		rd			op	
	C.LW	0	1	0	Imm[5:3]			rs1			Imm[2|6]		rd			0	0
	Reg[rd] = Mem[Reg[rs1] + {15'b0,Imm[6:2],2'b0}]																
CI type		funct3			Imm[5]	rd/rs1					Imm[4:0]					op	
	C.ADDI	0	0	0	Imm[5]	rs1/rd (≠ 0)					Imm[4:0]					0	1
	Reg[rd] = Reg[rs1] + sExt(Imm[5:0])																

Question 11:

Cache Specifications	
Cache Size	1KB
Cache Line Size	16B
Associativity	4
Write Policy	Write Back
Replacement policy	FIFO
Cache Type	Way Halting

Instruction Type	Instruction	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R type		**funct7**							**Im[4:0]**					**rs1**					**funct3**			**rd**					**opcode**						
	slli	0	0	0	0	0	0	0	shamt[4:0]					rs1					0	0	1	rd					0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] << shamt																																
		funct7							**rs2**					**rs1**					**funct3**			**rd**					**opcode**						
	add	0	0	0	0	0	0	0	rs2					rs1					0	0	0	rd					0	1	1	0	0	1	1
	Reg[rd] = Reg[rs1] + Reg[rs2]																																
I type		**Imm[11:0]**												**rs1**					**funct3**			**rd**					**opcode**						
	jalr	Imm[11:0]												rs1					0	0	0	rd					1	1	0	0	1	1	1
	Reg[rd] = PC + 3 (next PC), PC = sExt({Imm[11:0],0}) + Reg[rs1]																																
	xori	Imm[11:0]												rs1					1	0	0	rd					0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] ^ sExt(Imm[11:0])																																
	lh	Imm[11:0]												rs1					0	0	1	rd					0	0	0	0	0	1	1
	Reg[rd] = sExt(Mem[[Reg[rs1] + sExt(Imm[11:0])] (16 bits))																																
B type		**Imm**							**rs2**					**rs1**					**funct3**			**Imm**					**opcode**						
	BEQ	imm[12]	imm[10:5]						rs2					rs1					0	0	0	imm[4:1]				imm[11]	1	1	0	0	0	1	1
	PC(target) <= sext12to32(imm[12:0]) , if rs1==rs2																																

Instruction Type	Instruction	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CI type		funct3			Imm[5]	rd					Imm[4:0]					op	
	C.LI	0	1	0	Imm[5]	rd (≠ 0)					Imm[4:0]					0	1
	Reg[rd] = sExt(Imm[5:0])																
CR type		funct6						rd/rs1			funct		rs2			op	
	C.AND	1	0	0	0	1	1	rd/rs1			1	1	rs2			0	1
	Reg[rd] = Reg[rs1] & Reg[rs2]																
CS type		funct3			Imm[5:3]			rs1			Imm[2|6]		rs2			op	
	C.SW	1	1	0	Imm[5:3]			rs1			Imm[2|6]		rs2			0	0
	Mem[Reg[rs1] + {15'b0,Imm[6:2],2'b0}] = Reg[rs2]																

Question 12:

Cache Specifications	
Cache Size	1KB
Cache Line Size	16B
Associativity	4
Write Policy	Write Back
Replacement policy	FIFO
Cache Type	Way Prediction

Instruction Type	Instruction	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R type		**funct7**							**Im[4:0]**					**rs1**					**funct3**			**rd**				**opcode**						
	srli	0	0	0	0	0	0	0	shamt[4:0]					rs1					1	0	1	rd				0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] >> shamt																																
		funct7							**rs2**					**rs1**					**funct3**			**rd**				**opcode**						
	xor	0	0	0	0	0	0	0	rs2					rs1					1	0	0	rd				0	1	1	0	0	1	1
	Reg[rd] = Reg[rs1] ^ Reg[rs2]																																
I type		**Imm[11:0]**												**rs1**					**funct3**			**rd**				**opcode**						
	addi	Imm[11:0]												rs1					0	0	0	rd				0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] + sExt(Imm[11:0])																																
	jalr	Imm[11:0]												rs1					0	0	0	rd				1	1	0	0	1	1	1
	Reg[rd] = PC + 3 (next PC), PC = sExt({Imm[11:0],0}) + Reg[rs1]																																
	slti	Imm[11:0]												rs1					0	1	0	rd				0	0	1	0	0	1	1
	Reg[rd] = (Reg[rs1] < sExt(Imm[11:0]))?1:0																																
S type		**Imm[11:5]**							**rs2**					**rs1**					**funct3**			**Imm[4:0]**				**opcode**						
	sw	Imm[11:5]							rs2					rs1					0	0	1	Imm[4:0]				0	1	0	0	0	1	1
	Mem[Reg[rs1] + sExt(Imm[11:0])] = Reg[rs2]																																

Instruction Type	Instruction	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CI type		**funct3**			**Imm[5]**	**rd**					**Imm[4:0]**					**op**	
	C.LUI	0	1	1	Imm[5]	rd (≠ 0)					Imm[4:0]					0	1
	Reg[rd] = sExt({Imm{5:0],12'b0})																
CB type		**funct3**			**Imm[8|4:3]**			**rs1**			**Imm[7:6|2:1|5]**					**op**	
	C.BNEZ	1	1	1	Imm[8|4:3]			rs1			Imm[7:6|2:1|5]					0	1
	if(Reg[rs1] != 0) PC = PC + sExt(Imm)																
CL type		**funct3**			**Imm[5:3]**			**rs1**			**Imm[2|6]**		**rd**			**op**	
	C.LW	0	1	0	Imm[5:3]			rs1			Imm[2|6]		rd			0	0
	Reg[rd] = Mem[Reg[rs1] + {15'b0,Imm[6:2],2'b0}]																

Question 13:

Cache Specifications	
Cache Size	1KB
Cache Line Size	16B
Associativity	4
Write Policy	Write Back
Replacement policy	LRU counter
Cache Type	Way Halting

Instruction Type	Instruction	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R type		funct7							Im[4:0]					rs1					funct3			rd								opcode					
	srai	0	1	0	0	0	0	0	shamt[4:0]					rs1					1	0	1	rd				0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] >>> shamt																																
		funct7							rs2					rs1					funct3			rd								opcode					
	sub	0	1	0	0	0	0	0	rs2					rs1					0	0	0	rd				0	1	1	0	0	1	1
	Reg[rd] = Reg[rs1] - Reg[rs2]																																
I type		Imm[11:0]												rs1					funct3			rd								opcode					
	addi	Imm[11:0]												rs1					0	0	0	rd				0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] + sExt(Imm[11:0])																																
	jalr	Imm[11:0]												rs1					0	0	0	rd				1	1	0	0	1	1	1
	Reg[rd] = PC + 3 (next PC), PC = sExt({Imm[11:0],0}) + Reg[rs1]																																
	sltiu	Imm[11:0]												rs1					0	1	1	rd				0	0	1	0	0	1	1
	Reg[rd] = (Reg[rs1] <{20'b0,Imm[11:0]})?1:0																																
B type		Imm							rs2					rs1					funct3			Imm								opcode					
	BLT	imm[12]	imm[10:5]						rs2					rs1					1	0	0	imm[4:1]				imm[11]	1	1	0	0	0	1	1
	PC(target) <= sext12to32(imm[12:0]) , if rs1<rs2																																

Instruction Type	Instruction	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CR type		**funct4**				**rd/rs1**					**rs2**					**op**	
	C.MV	1	0	0	0	rd (≠ 0)					rs2 (≠ 0)					1	0
	Reg[rd] = Reg[rs2]																
CL type		**funct3**			**Imm[5:3]**			**rs1**			**Imm[2|6]**		**rd**			**op**	
	C.LW	0	1	0	Imm[5:3]			rs1			Imm[2|6]		rd			0	0
	Reg[rd] = Mem[Reg[rs1] + {15'b0,Imm[6:2],2'b0}]																
CS type		**funct3**			**Imm[5:3]**			**rs1**			**Imm[2|6]**		**rs2**			**op**	
	C.SW	1	1	0	Imm[5:3]			rs1			Imm[2|6]		rs2			0	0
	Mem[Reg[rs1] + {15'b0,Imm[6:2],2'b0}] = Reg[rs2]																

Question 14:

Cache Specifications	
Cache Size	1KB
Cache Line Size	16B
Associativity	4
Write Policy	Write Back
Replacement policy	LRU counter
Cache Type	Way Prediction

Instruction Type	Instruction	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R type		funct7							Im[4:0]					rs1					funct3			rd				opcode						
	slli	0	0	0	0	0	0	0	shamt[4:0]					rs1					0	0	1	rd				0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] << shamt																																
		funct7							rs2					rs1					funct3			rd				opcode						
	sltu	0	0	0	0	0	0	0	rs2					rs1					0	1	1	rd				0	1	1	0	0	1	1
	Reg[rd] = (unsigned(Reg[rs1]) < unsigned(Reg[rs2]))?1:0																																
I type		Imm[11:0]												rs1					funct3			rd				opcode						
	addi	Imm[11:0]												rs1					0	0	0	rd				0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] + sExt(Imm[11:0])																																
	lh	Imm[11:0]												rs1					0	0	1	rd				0	0	0	0	0	1	1
	Reg[rd] = sExt(Mem[[Reg[rs1] + sExt(Imm[11:0])] (16 bits))																																
U type		Imm																								rd				opcode			
	LUI	imm[31:12]																								rd				0	1	1	1
	Reg[rd] = {imm[31:12],12'd0}																																
B type		Imm							rs2					rs1					funct3			Imm				opcode						
	BLTU	imm[12]	imm[10:5]						rs2					rs1					1	1	0	imm[4:1]				imm[11]	1	1	0	0	0	1	1
	PC(target) <= sext12to32(imm[12:0]) , if rs1<rs2 (unsigned comparision)																																

Instruction Type	Instruction	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CR type		funct4				rd/rs1					rs2					op	
	C.MV	1	0	0	0	rd (≠ 0)					rs2 (≠ 0)					1	0
	Reg[rd] = Reg[rs2]																
CS type		funct3			Imm[5:3]			rs1			Imm[2|6]		rs2			op	
	C.SW	1	1	0	Imm[5:3]			rs1			Imm[2|6]		rs2			0	0
	Mem[Reg[rs1] + {15'b0,Imm[6:2],2'b0}] = Reg[rs2]																
CJ type		funct3			Imm[10:0]											op	
	C.JALR	0	0	1	Imm[10:0]											0	1
	Reg[x1] = PC + 3 (new PC), PC = PC + sExt(Imm)																

Question 15:

Cache Specifications	
Cache Size	1KB
Cache Line Size	16B
Associativity	4
Write Policy	Write Back
Replacement policy	Pseudo LRU
Cache Type	Way Halting

Instruction Type	Instruction	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R type		**funct7**							**Im[4:0]**					**rs1**					**funct3**			**rd**				**opcode**						
	srai	0	1	0	0	0	0	0	shamt[4:0]					rs1					1	0	1	rd				0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] >>> shamt																																
		funct7							**rs2**					**rs1**					**funct3**			**rd**				**opcode**						
	srl	0	0	0	0	0	0	0	rs2					rs1					1	0	1	rd				0	1	1	0	0	1	1
	Reg[rd] = Reg[rs1] >> Reg[rs2][4:0]																																
I type		**Imm[11:0]**												**rs1**					**funct3**			**rd**				**opcode**						
	addi	Imm[11:0]												rs1					0	0	0	rd				0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] + sExt(Imm[11:0])																																
	ori	Imm[11:0]												rs1					1	1	0	rd				0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] | sExt(Imm[11:0])																																
S type		**Imm[11:5]**							**rs2**					**rs1**					**funct3**			**Imm[4:0]**				**opcode**						
	sb	Imm[11:5]							rs2					rs1					0	0	0	Imm[4:0]				0	1	0	0	0	1	1
	Mem[Reg[rs1] + sExt(Imm[11:0])] = Reg[rs2][7:0]																																
U type		**Imm**																				**rd**				**opcode**						
	AUIPC	imm[31:12]																				rd				0	0	1	0	1	1	1
	Reg[rd] = PC + {imm[31:12],12'd0}																																

Instruction Type	Instruction	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CR type		funct3			Imm[5]	rd/rs1					Imm[4:0]					op	
	C.JALR	1	0	0	1	rs1 (≠ 0)					0	0	0	0	0	1	0
	Reg[1] = PC + 3 (new PC), PC = Reg[rs1]																
CL type		funct3			Imm[5:3]			rs1			Imm[2|6]		rd			op	
	C.LW	0	1	0	Imm[5:3]			rs1			Imm[2|6]		rd			0	0
	Reg[rd] = Mem[Reg[rs1] + {15'b0,Imm[6:2],2'b0}]																
CB type		funct3			Imm[8|4:3]			rs1			Imm[7:6|2:1|5]					op	
	C.BNEZ	1	1	1	Imm[8|4:3]			rs1			Imm[7:6|2:1|5]					0	1
	if(Reg[rs1] != 0) PC = PC + sExt(Imm)																

Question 16:

Cache Specifications	
Cache Size	1KB
Cache Line Size	16B
Associativity	4
Write Policy	Write Back
Replacement policy	Pseudo LRU
Cache Type	Way Prediction

Instruction Type	Instruction	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R type		**funct7**							**Im[4:0]**					**rs1**					**funct3**			**rd**				**opcode**						
	slli	0	0	0	0	0	0	0	shamt[4:0]					rs1					0	0	1	rd				0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] << shamt																																
		funct7							**rs2**					**rs1**					**funct3**			**rd**				**opcode**						
	or	0	0	0	0	0	0	0	rs2					rs1					1	1	0	rd				0	1	1	0	0	1	1
	Reg[rd] = Reg[rs1] | Reg[rs2]																																
I type		**Imm[11:0]**												**rs1**					**funct3**			**rd**				**opcode**						
	lh	Imm[11:0]												rs1					0	0	1	rd				0	0	0	0	0	1	1
	Reg[rd] = sExt(Mem[[Reg[rs1] + sExt(Imm[11:0])] (16 bits))																																
	xori	Imm[11:0]												rs1					1	0	0	rd				0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] ^ sExt(Imm[11:0])																																
	jalr	Imm[11:0]												rs1					0	0	0	rd				1	1	0	0	1	1	1
	Reg[rd] = PC + 3 (next PC), PC = sExt({Imm[11:0],0}) + Reg[rs1]																																
U type		**Imm**																				**rd**				**opcode**						
	LUI	imm[31:12]																				rd				0	1	1	0	1	1	1
	Reg[rd] = {imm[31:12],12'd0}																																

Instruction Type	Instruction	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CR type		**funct3**			**Imm[5]**	**rd/rs1**					**Imm[4:0]**					**op**	
	C.ADDI	0	0	0	Imm[5]	rs1/rd (≠ 0)					Imm[4:0]					0	1
	Reg[rd] = Reg[rs1] + sExt(Imm[5:0])																
CB type		**funct3**			**Imm[8|4:3]**			**rs1**			**Imm[7:6|2:1|5]**					**op**	
	C.BEQZ	1	1	0	Imm[8|4:3]			rs1			Imm[7:6|2:1|5]					0	1
	if(Reg[rs1] == 0) PC = PC + sExt(Imm)																
CS type		**funct3**			**Imm[5:3]**			**rs1**			**Imm[2|6]**		**rs2**			**op**	
	C.SW	1	1	0	Imm[5:3]			rs1			Imm[2|6]		rs2			0	0
	Mem[Reg[rs1] + {15'b0,Imm[6:2],2'b0}] = Reg[rs2]																

Question 17:

Cache Specifications	
Cache Size	1KB
Cache Line Size	16B
Associativity	4
Write Policy	Write Back
Replacement policy	Random
Cache Type	Way Halting

Instruction Type	Instruction	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R type		funct7							Im[4:0]					rs1					funct3			rd								opcode					
	srai	0	1	0	0	0	0	0	shamt[4:0]					rs1					1	0	1	rd				0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] >>> shamt																																
		funct7							rs2					rs1					funct3			rd								opcode					
	add	0	0	0	0	0	0	0	rs2					rs1					0	0	0	rd				0	1	1	0	0	1	1
	Reg[rd] = Reg[rs1] + Reg[rs2]																																
I type		Imm[11:0]												rs1					funct3			rd								opcode					
	jalr	Imm[11:0]												rs1					0	0	0	rd				1	1	0	0	1	1	1
	Reg[rd] = PC + 3 (next PC), PC = sExt({Imm[11:0],0}) + Reg[rs1]																																
	andi	Imm[11:0]												rs1					1	1	0	rd				0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] & sExt(Imm[11:0])																																
S type		Imm[11:5]							rs2					rs1					funct3			Imm[4:0]								opcode					
	sh	Imm[11:5]							rs2					rs1					0	0	1	Imm[4:0]				0	1	0	0	0	1	1
	Mem[Reg[rs1] + sExt(Imm[11:0])] = Reg[rs2][15:0]																																
B type		Imm							rs2					rs1					funct3			Imm								opcode					
	BGE	imm[12]	imm[10:5]						rs2					rs1					1	0	1	imm[4:1]			imm[11]					1	1	0	0	0	1	1
	PC(target) <= sext12to32(imm[12:0]) , if rs1>rs2																																

Instruction Type	Instruction	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CR type		funct4				rd/rs1					rs2					op	
	C.MV	1	0	0	0	rd (≠ 0)					rs2 (≠ 0)					1	0
	Reg[rd] = Reg[rs2]																
CI type		funct3			Imm[5]	rd					Imm[4:0]					op	
	C.LI	0	1	0	Imm[5]	rd (≠ 0)					Imm[4:0]					0	1
	Reg[rd] = sExt(Imm[5:0])																
CL type		funct3			Imm[5:3]			rs1			Imm[2|6]		rd			op	
	C.LW	0	1	0	Imm[5:3]			rs1			Imm[2|6]		rd			0	0
	Reg[rd] = Mem[Reg[rs1] + {15'b0,Imm[6:2],2'b0}]																

Question 18:

Cache Specifications	
Cache Size	1KB
Cache Line Size	16B
Associativity	4
Write Policy	Write Back
Replacement policy	Random
Cache Type	Way Prediction

Instruction Type	Instruction	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R type		**funct7**							**Im[4:0]**					**rs1**					**funct3**			**rd**				**opcode**						
	srli	0	0	0	0	0	0	0	shamt[4:0]					rs1					1	0	1	rd				0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] >> shamt																																
		funct7							**rs2**					**rs1**					**funct3**			**rd**				**opcode**						
	sltu	0	0	0	0	0	0	0	rs2					rs1					0	1	1	rd				0	1	1	0	0	1	1
	Reg[rd] = (unsigned(Reg[rs1]) < unsigned(Reg[rs2]))?1:0																																
I type		**Imm[11:0]**												**rs1**					**funct3**			**rd**				**opcode**						
	addi	Imm[11:0]												rs1					0	0	0	rd				0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] + sExt(Imm[11:0])																																
	lbu	Imm[11:0]												rs1					1	0	0	rd				0	0	0	0	0	1	1
	Reg[rd] = {24'b0,Mem[[Reg[rs1] + sExt(Imm[11:0])] (8 bits)}																																
	xori	Imm[11:0]												rs1					1	0	0	rd				0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] ^ sExt(Imm[11:0])																																
U type		**Imm**																				**rd**				**opcode**						
	AUIPC	imm[31:12]																				rd				0	0	1	0	1	1	1
	Reg[rd] = PC + {imm[31:12],12'd0}																																

Instruction Type	Instruction	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CR type		funct4				rd/rs1					rs2					op	
	C.JALR	1	0	0	1	rs1 (≠ 0)					0	0	0	0	0	1	0
	Reg[1] = PC + 3 (new PC), PC = Reg[rs1]																
CB type		funct3			Imm[8|4:3]			rs1			Imm[7:6|2:1|5]					op	
	C.BEQZ	1	1	0	Imm[8|4:3]			rs1			Imm[7:6|2:1|5]					0	1
	if(Reg[rs1] == 0) PC = PC + sExt(Imm)																
CS type		funct3			Imm[5:3]			rs1			Imm[2|6]		rs2			op	
	C.SW	1	1	0	Imm[5:3]			rs1			Imm[2|6]		rs2			0	0
	Mem[Reg[rs1] + {15'b0,Imm[6:2],2'b0}] = Reg[rs2]																

Question 19:

Cache Specifications	
Cache Size	1KB
Cache Line Size	16B
Associativity	4
Write Policy	Write Buffer
Replacement policy	FIFO
Cache Type	Way Halting

Instruction Type	Instruction	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R type		**funct7**							**Im[4:0]**					**rs1**					**funct3**			**rd**				**opcode**						
	slli	0	0	0	0	0	0	0	shamt[4:0]					rs1					0	0	1	rd				0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] << shamt																																
		funct7							**rs2**					**rs1**					**funct3**			**rd**				**opcode**						
	add	0	0	0	0	0	0	0	rs2					rs1					0	0	0	rd				0	1	1	0	0	1	1
	Reg[rd] = Reg[rs1] + Reg[rs2]																																
I type		**Imm[11:0]**												**rs1**					**funct3**			**rd**				**opcode**						
	jalr	Imm[11:0]												rs1					0	0	0	rd				1	1	0	0	1	1	1
	Reg[rd] = PC + 3 (next PC), PC = sExt({Imm[11:0],0}) + Reg[rs1]																																
	xori	Imm[11:0]												rs1					1	0	0	rd				0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] ^ sExt(Imm[11:0])																																
	lh	Imm[11:0]												rs1					0	0	1	rd				0	0	0	0	0	1	1
	Reg[rd] = sExt(Mem[[Reg[rs1] + sExt(Imm[11:0])] (16 bits))																																
B type		**Imm**							**rs2**					**rs1**					**funct3**			**Imm**				**opcode**						
	BEQ	imm[12]	imm[10:5]						rs2					rs1					0	0	0	imm[4:1]			imm[11]	1	1	0	0	0	1	1
	PC(target) <= sext12to32(imm[12:0]) , if rs1==rs2																																

Instruction Type	Instruction	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CI type		funct3			Imm[5]	rd					Imm[4:0]					op	
	C.LI	0	1	0	Imm[5]	rd (≠ 0)					Imm[4:0]					0	1
	Reg[rd] = sExt(Imm[5:0])																
CR type		funct6						rd/rs1			funct		rs2			op	
	C.AND	1	0	0	0	1	1	rd/rs1			1	1	rs2			0	1
	Reg[rd] = Reg[rs1] & Reg[rs2]																
CS type		funct3			Imm[5:3]			rs1			Imm[2|6]		rs2			op	
	C.SW	1	1	0	Imm[5:3]			rs1			Imm[2|6]		rs2			0	0
	Mem[Reg[rs1] + {15'b0,Imm[6:2],2'b0}] = Reg[rs2]																

Question 20:

Cache Specifications	
Cache Size	1KB
Cache Line Size	16B
Associativity	4
Write Policy	Write Buffer
Replacement policy	FIFO
Cache Type	Way Prediction

Instruction Type	Instruction	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R type		**funct7**							**Im[4:0]**					**rs1**					**funct3**			**rd**				**opcode**						
	srli	0	0	0	0	0	0	0	shamt[4:0]					rs1					1	0	1	rd				0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] >> shamt																																
		funct7							**rs2**					**rs1**					**funct3**			**rd**				**opcode**						
	xor	0	0	0	0	0	0	0	rs2					rs1					1	0	0	rd				0	1	1	0	0	1	1
	Reg[rd] = Reg[rs1] ^ Reg[rs2]																																
I type		**Imm[11:0]**												**rs1**					**funct3**			**rd**				**opcode**						
	addi	Imm[11:0]												rs1					0	0	0	rd				0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] + sExt(Imm[11:0])																																
	jalr	Imm[11:0]												rs1					0	0	0	rd				1	1	0	0	1	1	1
	Reg[rd] = PC + 3 (next PC), PC = sExt({Imm[11:0],0}) + Reg[rs1]																																
	slti	Imm[11:0]												rs1					0	1	0	rd				0	0	1	0	0	1	1
	Reg[rd] = (Reg[rs1] < sExt(Imm[11:0]))?1:0																																
S type		**Imm[11:5]**							**rs2**					**rs1**					**funct3**			**Imm[4:0]**				**opcode**						
	sw	Imm[11:5]							rs2					rs1					0	0	1	Imm[4:0]				0	1	0	0	0	1	1
	Mem[Reg[rs1] + sExt(Imm[11:0])] = Reg[rs2]																																

Instruction Type	Instruction	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CI type		**funct3**			**Imm[5]**	**rd**					**Imm[4:0]**					**op**	
	C.LUI	0	1	1	Imm[5]	rd (≠ 0)					Imm[4:0]					0	1
	Reg[rd] = sExt({Imm{5:0],12'b0})																
CB type		**funct3**			**Imm[8|4:3]**			**rs1**			**Imm[7:6|2:1|5]**					**op**	
	C.BNEZ	1	1	1	Imm[8|4:3]			rs1			Imm[7:6|2:1|5]					0	1
	if(Reg[rs1] != 0) PC = PC + sExt(Imm)																
CL type		**funct3**			**Imm[5:3]**			**rs1**			**Imm[2|6]**		**rd**			**op**	
	C.LW	0	1	0	Imm[5:3]			rs1			Imm[2|6]		rd			0	0
	Reg[rd] = Mem[Reg[rs1] + {15'b0,Imm[6:2],2'b0}]																

Question 21:

Cache Specifications	
Cache Size	1KB
Cache Line Size	16B
Associativity	4
Write Policy	Write Buffer
Replacement policy	LRU counter
Cache Type	Way Halting

Instruction Type	Instruction	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R type		**funct7**							**Im[4:0]**					**rs1**					**funct3**			**rd**				**opcode**						
	srai	0	1	0	0	0	0	0	shamt[4:0]					rs1					1	0	1	rd				0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] >>> shamt																																
		funct7							**rs2**					**rs1**					**funct3**			**rd**				**opcode**						
	sub	0	1	0	0	0	0	0	rs2					rs1					0	0	0	rd				0	1	1	0	0	1	1
	Reg[rd] = Reg[rs1] - Reg[rs2]																																
I type		**Imm[11:0]**												**rs1**					**funct3**			**rd**				**opcode**						
	addi	Imm[11:0]												rs1					0	0	0	rd				0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] + sExt(Imm[11:0])																																
	jalr	Imm[11:0]												rs1					0	0	0	rd				1	1	0	0	1	1	1
	Reg[rd] = PC + 3 (next PC), PC = sExt({Imm[11:0],0}) + Reg[rs1]																																
	sltiu	Imm[11:0]												rs1					0	1	1	rd				0	0	1	0	0	1	1
	Reg[rd] = (Reg[rs1] <{20'b0,Imm[11:0]})?1:0																																
B type		**Imm**							**rs2**					**rs1**					**funct3**			**Imm**				**opcode**						
	BLT	imm[12]	imm[10:5]						rs2					rs1					1	0	0	imm[4:1]			imm[11]	1	1	0	0	0	1	1
	PC(target) <= sext12to32(imm[12:0]) , if rs1<rs2																																

Instruction Type	Instruction	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CR type		**funct4**				**rd/rs1**					**rs2**					**op**	
	C.MV	1	0	0	0	rd (≠ 0)					rs2 (≠ 0)					1	0
	Reg[rd] = Reg[rs2]																
CL type		**funct3**			**Imm[5:3]**			**rs1**			**Imm[2|6]**		**rd**			**op**	
	C.LW	0	1	0	Imm[5:3]			rs1			Imm[2|6]		rd			0	0
	Reg[rd] = Mem[Reg[rs1] + {15'b0,Imm[6:2],2'b0}]																
CS type		**funct3**			**Imm[5:3]**			**rs1**			**Imm[2|6]**		**rs2**			**op**	
	C.SW	1	1	0	Imm[5:3]			rs1			Imm[2|6]		rs2			0	0
	Mem[Reg[rs1] + {15'b0,Imm[6:2],2'b0}] = Reg[rs2]																

Question 22:

Cache Specifications	
Cache Size	1KB
Cache Line Size	16B
Associativity	4
Write Policy	Write Buffer
Replacement policy	LRU counter
Cache Type	Way Prediction

Instruction Type	Instruction	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R type		funct7							Im[4:0]					rs1					funct3			rd				opcode						
	slli	0	0	0	0	0	0	0	shamt[4:0]					rs1					0	0	1	rd				0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] << shamt																																
		funct7							rs2					rs1					funct3			rd				opcode						
	sltu	0	0	0	0	0	0	0	rs2					rs1					0	1	1	rd				0	1	1	0	0	1	1
	Reg[rd] = (unsigned(Reg[rs1]) < unsigned(Reg[rs2]))?1:0																																
I type		Imm[11:0]												rs1					funct3			rd				opcode						
	addi	Imm[11:0]												rs1					0	0	0	rd				0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] + sExt(Imm[11:0])																																
	lh	Imm[11:0]												rs1					0	0	1	rd				0	0	0	0	0	1	1
	Reg[rd] = sExt(Mem[[Reg[rs1] + sExt(Imm[11:0])] (16 bits))																																
U type		Imm																								rd				opcode			
	LUI	imm[31:12]																								rd				0	1	1	1
	Reg[rd] = {imm[31:12],12'd0}																																
B type		Imm							rs2					rs1					funct3			Imm				opcode						
	BLTU	imm[12]	imm[10:5]						rs2					rs1					1	1	0	imm[4:1]				imm[11]	1	1	0	0	0	1	1
	PC(target) <= sext12to32(imm[12:0]) , if rs1<rs2 (unsigned comparision)																																

Instruction Type	Instruction	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CR type		funct4				rd/rs1					rs2					op	
	C.MV	1	0	0	0	rd (≠ 0)					rs2 (≠ 0)					1	0
	Reg[rd] = Reg[rs2]																
CS type		funct3			Imm[5:3]			rs1			Imm[2|6]		rs2			op	
	C.SW	1	1	0	Imm[5:3]			rs1			Imm[2|6]		rs2			0	0
	Mem[Reg[rs1] + {15'b0,Imm[6:2],2'b0}] = Reg[rs2]																
CJ type		funct3			Imm[10:0]											op	
	C.JALR	0	0	1	Imm[10:0]											0	1
	Reg[x1] = PC + 3 (new PC), PC = PC + sExt(Imm)																

Question 23:

Cache Specifications	
Cache Size	1KB
Cache Line Size	16B
Associativity	4
Write Policy	Write Buffer
Replacement policy	Pseudo LRU
Cache Type	Way Halting

Instruction Type	Instruction	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R type		**funct7**							**Im[4:0]**					**rs1**					**funct3**			**rd**				**opcode**						
	srai	0	1	0	0	0	0	0	shamt[4:0]					rs1					1	0	1	rd				0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] >>> shamt																																
		funct7							**rs2**					**rs1**					**funct3**			**rd**				**opcode**						
	srl	0	0	0	0	0	0	0	rs2					rs1					1	0	1	rd				0	1	1	0	0	1	1
	Reg[rd] = Reg[rs1] >> Reg[rs2][4:0]																																
I type		**Imm[11:0]**												**rs1**					**funct3**			**rd**				**opcode**						
	addi	Imm[11:0]												rs1					0	0	0	rd				0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] + sExt(Imm[11:0])																																
	ori	Imm[11:0]												rs1					1	1	0	rd				0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] | sExt(Imm[11:0])																																
S type		**Imm[11:5]**							**rs2**					**rs1**					**funct3**			**Imm[4:0]**				**opcode**						
	sb	Imm[11:5]							rs2					rs1					0	0	0	Imm[4:0]				0	1	0	0	0	1	1
	Mem[Reg[rs1] + sExt(Imm[11:0])] = Reg[rs2][7:0]																																
U type		**Imm**																				**rd**				**opcode**						
	AUIPC	imm[31:12]																				rd				0	0	1	0	1	1	1
	Reg[rd] = PC + {imm[31:12],12'd0}																																

Instruction Type	Instruction	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CR type		funct3			Imm[5]	rd/rs1					Imm[4:0]					op	
	C.JALR	1	0	0	1	rs1 (≠ 0)					0	0	0	0	0	1	0
	Reg[1] = PC + 3 (new PC), PC = Reg[rs1]																
CL type		funct3			Imm[5:3]			rs1			Imm[2|6]		rd			op	
	C.LW	0	1	0	Imm[5:3]			rs1			Imm[2|6]		rd			0	0
	Reg[rd] = Mem[Reg[rs1] + {15'b0,Imm[6:2],2'b0}]																
CB type		funct3			Imm[8|4:3]			rs1			Imm[7:6|2:1|5]					op	
	C.BNEZ	1	1	1	Imm[8|4:3]			rs1			Imm[7:6|2:1|5]					0	1
	if(Reg[rs1] != 0) PC = PC + sExt(Imm)																

Question 24:

Cache Specifications	
Cache Size	1KB
Cache Line Size	16B
Associativity	4
Write Policy	Write Buffer
Replacement policy	Pseudo LRU
Cache Type	Way Prediction

Instruction Type	Instruction	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R type		**funct7**							**Im[4:0]**					**rs1**					**funct3**			**rd**				**opcode**						
	slli	0	0	0	0	0	0	0	shamt[4:0]					rs1					0	0	1	rd				0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] << shamt																																
		funct7							**rs2**					**rs1**					**funct3**			**rd**				**opcode**						
	or	0	0	0	0	0	0	0	rs2					rs1					1	1	0	rd				0	1	1	0	0	1	1
	Reg[rd] = Reg[rs1] | Reg[rs2]																																
I type		**Imm[11:0]**																	**rs1**			**funct3**			**rd**			**opcode**					
	lh	Imm[11:0]																	rs1			0	0	1	rd		0	0	0	0	0	1	1
	Reg[rd] = sExt(Mem[[Reg[rs1] + sExt(Imm[11:0])] (16 bits))																																
	xori	Imm[11:0]																	rs1			1	0	0	rd		0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] ^ sExt(Imm[11:0])																																
	jalr	Imm[11:0]																	rs1			0	0	0	rd		1	1	0	0	1	1	1
	Reg[rd] = PC + 3 (next PC), PC = sExt({Imm[11:0],0}) + Reg[rs1]																																
U type		**Imm**																							**rd**		**opcode**						
	LUI	imm[31:12]																							rd		0	1	1	0	1	1	1
	Reg[rd] = {imm[31:12],12'd0}																																

Instruction Type	Instruction	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CR type		**funct3**			**Imm[5]**	**rd/rs1**					**Imm[4:0]**					**op**	
	C.ADDI	0	0	0	Imm[5]	rs1/rd (≠ 0)					Imm[4:0]					0	1
	Reg[rd] = Reg[rs1] + sExt(Imm[5:0])																
CB type		**funct3**			**Imm[8|4:3]**			**rs1**			**Imm[7:6|2:1|5]**					**op**	
	C.BEQZ	1	1	0	Imm[8|4:3]			rs1			Imm[7:6|2:1|5]					0	1
	if(Reg[rs1] == 0) PC = PC + sExt(Imm)																
CS type		**funct3**			**Imm[5:3]**			**rs1**			**Imm[2|6]**		**rs2**			**op**	
	C.SW	1	1	0	Imm[5:3]			rs1			Imm[2|6]		rs2			0	0
	Mem[Reg[rs1] + {15'b0,Imm[6:2],2'b0}] = Reg[rs2]																

Question 25:

Cache Specifications	
Cache Size	1KB
Cache Line Size	16B
Associativity	4
Write Policy	Write Buffer
Replacement policy	Random
Cache Type	Way Halting

Instruction Type	Instruction	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R type		funct7							Im[4:0]					rs1					funct3			rd								opcode					
	srai	0	1	0	0	0	0	0	shamt[4:0]					rs1					1	0	1	rd				0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] >>> shamt																																
		funct7							rs2					rs1					funct3			rd								opcode					
	add	0	0	0	0	0	0	0	rs2					rs1					0	0	0	rd				0	1	1	0	0	1	1
	Reg[rd] = Reg[rs1] + Reg[rs2]																																
I type		Imm[11:0]												rs1					funct3			rd								opcode					
	jalr	Imm[11:0]												rs1					0	0	0	rd				1	1	0	0	1	1	1
	Reg[rd] = PC + 3 (next PC), PC = sExt({Imm[11:0],0}) + Reg[rs1]																																
	andi	Imm[11:0]												rs1					1	1	0	rd				0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] & sExt(Imm[11:0])																																
S type		Imm[11:5]							rs2					rs1					funct3			Imm[4:0]								opcode					
	sh	Imm[11:5]							rs2					rs1					0	0	1	Imm[4:0]				0	1	0	0	0	1	1
	Mem[Reg[rs1] + sExt(Imm[11:0])] = Reg[rs2][15:0]																																
B type		Imm							rs2					rs1					funct3			Imm								opcode					
	BGE	imm[12]	imm[10:5]						rs2					rs1					1	0	1	imm[4:1]			imm[11]					1	1	0	0	0	1	1
	PC(target) <= sext12to32(imm[12:0]) , if rs1>rs2																																

Instruction Type	Instruction	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CR type		funct4				rd/rs1					rs2					op	
	C.MV	1	0	0	0	rd (≠ 0)					rs2 (≠ 0)					1	0
	Reg[rd] = Reg[rs2]																
CI type		funct3			Imm[5]	rd					Imm[4:0]					op	
	C.LI	0	1	0	Imm[5]	rd (≠ 0)					Imm[4:0]					0	1
	Reg[rd] = sExt(Imm[5:0])																
CL type		funct3			Imm[5:3]			rs1			Imm[2|6]		rd			op	
	C.LW	0	1	0	Imm[5:3]			rs1			Imm[2|6]		rd			0	0
	Reg[rd] = Mem[Reg[rs1] + {15'b0,Imm[6:2],2'b0}]																

Question 26:

Cache Specifications	
Cache Size	1KB
Cache Line Size	16B
Associativity	4
Write Policy	Write Buffer
Replacement policy	Random
Cache Type	Way Prediction

Instruction Type	Instruction	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R type		**funct7**							**Im[4:0]**					**rs1**					**funct3**			**rd**				**opcode**						
	srli	0	0	0	0	0	0	0	shamt[4:0]					rs1					1	0	1	rd				0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] >> shamt																																
		funct7							**rs2**					**rs1**					**funct3**			**rd**				**opcode**						
	sltu	0	0	0	0	0	0	0	rs2					rs1					0	1	1	rd				0	1	1	0	0	1	1
	Reg[rd] = (unsigned(Reg[rs1]) < unsigned(Reg[rs2]))?1:0																																
I type		**Imm[11:0]**												**rs1**					**funct3**			**rd**				**opcode**						
	addi	Imm[11:0]												rs1					0	0	0	rd				0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] + sExt(Imm[11:0])																																
	lbu	Imm[11:0]												rs1					1	0	0	rd				0	0	0	0	0	1	1
	Reg[rd] = {24'b0,Mem[[Reg[rs1] + sExt(Imm[11:0])] (8 bits)}																																
	xori	Imm[11:0]												rs1					1	0	0	rd				0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] ^ sExt(Imm[11:0])																																
U type		**Imm**																				**rd**				**opcode**						
	AUIPC	imm[31:12]																				rd				0	0	1	0	1	1	1
	Reg[rd] = PC + {imm[31:12],12'd0}																																

Instruction Type	Instruction	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CR type		funct4				rd/rs1					rs2					op	
	C.JALR	1	0	0	1	rs1 (≠ 0)					0	0	0	0	0	1	0
	Reg[1] = PC + 3 (new PC), PC = Reg[rs1]																
CB type		funct3			Imm[8|4:3]			rs1			Imm[7:6|2:1|5]					op	
	C.BEQZ	1	1	0	Imm[8|4:3]			rs1			Imm[7:6|2:1|5]					0	1
	if(Reg[rs1] == 0) PC = PC + sExt(Imm)																
CS type		funct3			Imm[5:3]			rs1			Imm[2|6]		rs2			op	
	C.SW	1	1	0	Imm[5:3]			rs1			Imm[2|6]		rs2			0	0
	Mem[Reg[rs1] + {15'b0,Imm[6:2],2'b0}] = Reg[rs2]																

Question 27:

Cache Specifications	
Cache Size	1KB
Cache Line Size	16B
Associativity	4
Write Policy	Write Buffer
Replacement policy	FIFO
Cache Type	Way Halting

Instruction Type	Instruction	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R type		**funct7**							**Im[4:0]**					**rs1**					**funct3**			**rd**				**opcode**						
	slli	0	0	0	0	0	0	0	shamt[4:0]					rs1					0	0	1	rd				0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] << shamt																																
		funct7							**rs2**					**rs1**					**funct3**			**rd**				**opcode**						
	or	0	0	0	0	0	0	0	rs2					rs1					1	1	0	rd				0	1	1	0	0	1	1
	Reg[rd] = Reg[rs1] | Reg[rs2]																																
I type		**Imm[11:0]**												**rs1**					**funct3**			**rd**				**opcode**						
	jalr	Imm[11:0]												rs1					0	0	0	rd				1	1	0	0	1	1	1
	Reg[rd] = PC + 3 (next PC), PC = sExt({Imm[11:0],0}) + Reg[rs1]																																
	sltiu	Imm[11:0]												rs1					0	1	1	rd				0	0	1	0	0	1	1
	Reg[rd] = (Reg[rs1] <{20'b0,Imm[11:0]})?1:0																																
S type		**Imm[11:5]**							**rs2**					**rs1**					**funct3**			**Imm[4:0]**				**opcode**						
	sb	Imm[11:5]							rs2					rs1					0	0	0	Imm[4:0]				0	1	0	0	0	1	1
	Mem[Reg[rs1] + sExt(Imm[11:0])] = Reg[rs2][7:0]																																
J type		**Imm[19:0]**																				**rd**				**opcode**						
	JAL	imm[10:0]																				rd				1	1	0	1	1	1	1
	PC(target) <= sext20to32(imm[20:0]) + PC, rd = PC + 3 (new PC)																																

Instruction Type	Instruction	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CR type		funct4				rd/rs1					rs2					op	
	C.MV	1	0	0	0	rd (≠ 0)					rs2 (≠ 0)					1	0
	Reg[rd] = Reg[rs2]																
CL type		funct3			Imm[5:3]			rs1			Imm[2|6]		rd			op	
	C.LW	0	1	0	Imm[5:3]			rs1			Imm[2|6]		rd			0	0
	Reg[rd] = Mem[Reg[rs1] + {15'b0,Imm[6:2],2'b0}]																
CI type		funct3			Imm[5]	rd/rs1					Imm[4:0]					op	
	C.ADDI	0	0	0	Imm[5]	rs1/rd (≠ 0)					Imm[4:0]					0	1
	Reg[rd] = Reg[rs1] + sExt(Imm[5:0])																

Question 28:

Cache Specifications	
Cache Size	1KB
Cache Line Size	16B
Associativity	4
Write Policy	Write Buffer
Replacement policy	FIFO
Cache Type	Way Prediction

Instruction Type	Instruction	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R type		funct7							Im[4:0]					rs1					funct3			rd				opcode						
	srai	0	1	0	0	0	0	0	shamt[4:0]					rs1					1	0	1	rd				0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] >>> shamt																																
		funct7							rs2					rs1					funct3			rd				opcode						
	or	0	0	0	0	0	0	0	rs2					rs1					1	1	0	rd				0	1	1	0	0	1	1
	Reg[rd] = Reg[rs1] | Reg[rs2]																																
I type		Imm[11:0]												rs1					funct3			rd				opcode						
	jalr	Imm[11:0]												rs1					0	0	0	rd				1	1	0	0	1	1	1
	Reg[rd] = PC + 3 (next PC), PC = sExt({Imm[11:0],0}) + Reg[rs1]																																
	lh	Imm[11:0]												rs1					0	0	1	rd				0	0	0	0	0	1	1
	Reg[rd] = sExt(Mem[[Reg[rs1] + sExt(Imm[11:0])] (16 bits))																																
U type		Imm																				rd				opcode						
	AUIPC	imm[31:12]																				rd				0	0	1	0	1	1	1
	Reg[rd] = PC + {imm[31:12],12'd0}																																
B type		Imm							rs2					rs1					funct3			Imm				opcode						
	BNE	imm[12]	imm[10:5]						rs2					rs1					0	0	1	imm[4:1]			imm[11]	1	1	0	0	0	1	1
	PC(target) <= sext12to32(imm[12:0]) , if rs1=/=rs2																																

Instruction Type	Instruction	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CR type		funct3			Imm[5]	rd/rs1					Imm[4:0]					op	
	C.ADDI	0	0	0	Imm[5]	rs1/rd (≠ 0)					Imm[4:0]					0	1
	Reg[rd] = Reg[rs1] + sExt(Imm[5:0])																
CS type		funct3			Imm[5:3]			rs1			Imm[2|6]		rs2			op	
	C.SW	1	1	0	Imm[5:3]			rs1			Imm[2|6]		rs2			0	0
	Mem[Reg[rs1] + {15'b0,Imm[6:2],2'b0}] = Reg[rs2]																
CI type		funct3			Imm[5]	rd					Imm[4:0]					op	
	C.LI	0	1	0	Imm[5]	rd (≠ 0)					Imm[4:0]					0	1
	Reg[rd] = sExt(Imm[5:0])																

Question 29:

Cache Specifications	
Cache Size	1KB
Cache Line Size	16B
Associativity	4
Write Policy	Write Buffer
Replacement policy	LRU counter
Cache Type	Way Halting

Instruction Type	Instruction	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R type		**funct7**							**Im[4:0]**					**rs1**					**funct3**			**rd**				**opcode**						
	srli	0	0	0	0	0	0	0	shamt[4:0]					rs1					1	0	1	rd				0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] >> shamt																																
		funct7							**rs2**					**rs1**					**funct3**			**rd**				**opcode**						
	sra	0	1	0	0	0	0	0	rs2					rs1					1	0	1	rd				0	1	1	0	0	1	1
	Reg[rd] = Reg[rs1] >>> Reg[rs2][4:0]																																
I type		**Imm[11:0]**												**rs1**					**funct3**			**rd**				**opcode**						
	lw	Imm[11:0]												rs1					0	1	0	rd				0	0	0	0	0	1	1
	Reg[rd] = Mem[[Reg[rs1] + sExt(Imm[11:0])] (32 bits)																																
	addi	Imm[11:0]												rs1					0	0	0	rd				0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] + sExt(Imm[11:0])																																
	andi	Imm[11:0]												rs1					1	1	0	rd				0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] & sExt(Imm[11:0])																																
U type		**Imm**																				**rd**				**opcode**						
	AUIPC	imm[31:12]																				rd				0	0	1	0	1	1	1
	Reg[rd] = PC + {imm[31:12],12'd0}																																

Instruction Type	Instruction	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CR type		funct4				rd/rs1					rs2					op	
	C.MV	1	0	0	0	rd (≠ 0)					rs2 (≠ 0)					1	0
	Reg[rd] = Reg[rs2]																
CB type		funct3			Imm[8|4:3]			rs1			Imm[7:6|2:1|5]					op	
	C.BNEZ	1	1	1	Imm[8|4:3]			rs1			Imm[7:6|2:1|5]					0	1
	if(Reg[rs1] != 0) PC = PC + sExt(Imm)																
CS type		funct3			Imm[5:3]			rs1			Imm[2|6]		rs2			op	
	C.SW	1	1	0	Imm[5:3]			rs1			Imm[2|6]		rs2			0	0
	Mem[Reg[rs1] + {15'b0,Imm[6:2],2'b0}] = Reg[rs2]																

Question 30:

Cache Specifications	
Cache Size	1KB
Cache Line Size	16B
Associativity	4
Write Policy	Write Buffer
Replacement policy	LRU counter
Cache Type	Way Prediction

Instruction Type	Instruction	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R type		funct7							Im[4:0]					rs1					funct3			rd				opcode						
	srai	0	1	0	0	0	0	0	shamt[4:0]					rs1					1	0	1	rd				0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] >>> shamt																																
		funct7							rs2					rs1					funct3			rd				opcode						
	sub	0	1	0	0	0	0	0	rs2					rs1					0	0	0	rd				0	1	1	0	0	1	1
	Reg[rd] = Reg[rs1] - Reg[rs2]																																
I type		Imm[11:0]												rs1					funct3			rd				opcode						
	jalr	Imm[11:0]												rs1					0	0	0	rd				1	1	0	0	1	1	1
	Reg[rd] = PC + 3 (next PC), PC = sExt({Imm[11:0],0}) + Reg[rs1]																																
	lbu	Imm[11:0]												rs1					1	0	0	rd				0	0	0	0	0	1	1
	Reg[rd] = {24'b0,Mem[[Reg[rs1] + sExt(Imm[11:0])] (8 bits)}																																
U type		Imm																				rd				opcode						
	AUIPC	imm[31:12]																				rd				0	0	1	0	1	1	1
	Reg[rd] = PC + {imm[31:12],12'd0}																																
B type		Imm							rs2					rs1					funct3			Imm				opcode						
	BGEU	imm[12]	imm[10:5]						rs2					rs1					1	1	1	imm[4:1]			imm[11]	1	1	0	0	0	1	1
	PC(target) <= sext12to32(imm[12:0]) , if rs1>rs2 (unsigned comparision)																																

Instruction Type	Instruction	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CR type		**funct3**			**Imm[5]**	**rd/rs1**					**Imm[4:0]**					**op**	
	C.JALR	1	0	0	1	rs1 (≠ 0)					0	0	0	0	0	1	0
	Reg[1] = PC + 3 (new PC), PC = Reg[rs1]																
CS type		**funct3**			**Imm[5:3]**			**rs1**			**Imm[2|6]**		**rs2**			**op**	
	C.SW	1	1	0	Imm[5:3]			rs1			Imm[2|6]		rs2			0	0
	Mem[Reg[rs1] + {15'b0,Imm[6:2],2'b0}] = Reg[rs2]																
CJ type		**funct3**			**Imm[10:0]**											**op**	
	C.JALR	0	0	1	Imm[10:0]											0	1
	Reg[x1] = PC + 3 (new PC), PC = PC + sExt(Imm)																

Question 31:

Cache Specifications	
Cache Size	1KB
Cache Line Size	16B
Associativity	4
Write Policy	Write Buffer
Replacement policy	Pseudo LRU
Cache Type	Way Halting

Instruction Type	Instruction	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R type		funct7							Im[4:0]					rs1					funct3			rd								opcode					
	slli	0	0	0	0	0	0	0	shamt[4:0]					rs1					0	0	1	rd				0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] << shamt																																
		funct7							rs2					rs1					funct3			rd								opcode					
	sltu	0	0	0	0	0	0	0	rs2					rs1					0	1	1	rd				0	1	1	0	0	1	1
	Reg[rd] = (unsigned(Reg[rs1]) < unsigned(Reg[rs2]))?1:0																																
I type		Imm[11:0]												rs1					funct3			rd								opcode					
	lb	Imm[11:0]												rs1					0	0	0	rd				0	0	0	0	0	1	1
	Reg[rd] = sExt(Mem[[Reg[rs1] + sExt(Imm[11:0])] (8 bits))																																
	xori	Imm[11:0]												rs1					1	0	0	rd				0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] ^ sExt(Imm[11:0])																																
S type		Imm[11:5]							rs2					rs1					funct3			Imm[4:0]								opcode					
	sh	Imm[11:5]							rs2					rs1					0	0	1	Imm[4:0]				0	1	0	0	0	1	1
	Mem[Reg[rs1] + sExt(Imm[11:0])] = Reg[rs2][15:0]																																
B type		Imm							rs2					rs1					funct3			Imm								opcode					
	BLT	imm[12]	imm[10:5]						rs2					rs1					1	0	0	imm[4:1]			imm[11]	1	1	0	0	0	1	1
	PC(target) <= sext12to32(imm[12:0]) , if rs1<rs2																																

Instruction Type	Instruction	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CR type		**funct4**				**rd/rs1**					**rs2**					**op**	
	C.ADD	1	0	0	1	rs1/rd (≠ 0)					rs2 (≠ 0)					1	0
	Reg[rd] = Reg[rs1] + Reg[rs2]																
CI type		**funct3**			**Imm[5]**	**rd**					**Imm[4:0]**					**op**	
	C.LUI	0	1	1	Imm[5]	rd (≠ 0)					Imm[4:0]					0	1
	Reg[rd] = sExt({Imm{5:0],12'b0})																
CJ type		**funct3**			**Imm[10:0]**											**op**	
	C.J	1	0	1	Imm[10:0]											0	1
	PC = PC + sExt(Imm)																

Question 32:

Cache Specifications	
Cache Size	1KB
Cache Line Size	16B
Associativity	4
Write Policy	Write Buffer
Replacement policy	Pseudo LRU
Cache Type	Way Prediction

Instruction Type	Instruction	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R type		**funct7**							**Im[4:0]**					**rs1**					**funct3**			**rd**				**opcode**						
	srli	0	0	0	0	0	0	0	shamt[4:0]					rs1					1	0	1	rd				0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] >> shamt																																
		funct7							**rs2**					**rs1**					**funct3**			**rd**				**opcode**						
	xor	0	0	0	0	0	0	0	rs2					rs1					1	0	0	rd				0	1	1	0	0	1	1
	Reg[rd] = Reg[rs1] ^ Reg[rs2]																																
I type		**Imm[11:0]**												**rs1**					**funct3**			**rd**				**opcode**						
	jalr	Imm[11:0]												rs1					0	0	0	rd				1	1	0	0	1	1	1
	Reg[rd] = PC + 3 (next PC), PC = sExt({Imm[11:0],0}) + Reg[rs1]																																
	addi	Imm[11:0]												rs1					0	0	0	rd				0	0	1	0	0	1	1
	Reg[rd] = Reg[rs1] + sExt(Imm[11:0])																																
S type		**Imm[11:5]**							**rs2**					**rs1**					**funct3**			**Imm[4:0]**				**opcode**						
	sw	Imm[11:5]							rs2					rs1					0	0	1	Imm[4:0]				0	1	0	0	0	1	1
	Mem[Reg[rs1] + sExt(Imm[11:0])] = Reg[rs2]																																
U type		**Imm**																				**rd**				**opcode**						
	AUIPC	imm[31:12]																				rd				0	0	1	0	1	1	1
	Reg[rd] = PC + {imm[31:12],12'd0}																																

Instruction Type	Instruction	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CR type		funct4				rd/rs1					rs2					op	
	C.MV	1	0	0	0	rd (≠ 0)					rs2 (≠ 0)					1	0
	Reg[rd] = Reg[rs2]																
CL type		funct3			Imm[5:3]			rs1			Imm[2|6]		rd			op	
	C.LW	0	1	0	Imm[5:3]			rs1			Imm[2|6]		rd			0	0
	Reg[rd] = Mem[Reg[rs1] + {15'b0,Imm[6:2],2'b0}]																
CB type		funct3			Imm[8|4:3]			rs1			Imm[7:6|2:1|5]					op	
	C.BEQZ	1	1	0	Imm[8|4:3]			rs1			Imm[7:6|2:1|5]					0	1
	if(Reg[rs1] == 0) PC = PC + sExt(Imm)																