BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI
K. K. BIRLA Goa Campus
First Semester 2015-2016
CS F342 Computer Architecture
Course Project

INSTRUCTIONS

1. The project weightage is 10**%** and the deadline is **24th November 2015**.
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.
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 send the same to biju@goa.bits-pilani.ac.in at the earliest. [Make sure you are sending your preferences before **04th November 2015**]. [Make sure ONLY ONE person from a team is sending the e-mail].
3. Allocation of project is on First Come First Serve basis. No default project allocation is available – i.e. if you are not sending e-mail before 04th November 2015, 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, halt tag array and victim cache for Way-Prediction cache, Way-Halting cache and Victim 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

Memory Specifications	
Cache Size	1 KB
Cache Line Size	8B
Associativity	2
Write Policy	Write Back
Replacement policy	FIFO
Cache Type	Way Prediction

Instructions

		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Load	LDRH <Rd> <Rn> <Rm>	0	1	0	1	0	0	1	Rm			Rn			Rd		
Store	STRH <Rd> <Rn> <Rm>	0	1	0	1	1	1	0	Rm			Rn			Rd		
Add	ADD <Rd> PC #<8_bit_imm>	1	0	1	0	1	Rd			Imm							
Sub	SBC <Rm> <Rn>	0	1	0	0	0	0	0	1	0	1	Rm			Rd		
Shift	LSL <Rd> <Rm> #<5_bit_imm>	0	0	0	0	0	immediate					Rm			Rd		
Jump	B <target address>	1	1	1	0	0	Offset										
Mul	MUL <Rm> <Rn>	0	1	0	0	0	0	0	1	1	1	Rm			Rd		
Branch	B LT #<8_bit_offset>	1	1	0	1	0	0	1	1	Offset							

Exceptions(Arithmetic Overflow and Undefined Instruction)

Question #2

Memory Specifications	
Cache Size	1 KB
Cache Line Size	8B
Associativity	2
Write Policy	Write Back
Replacement policy	FIFO
Cache Type	Way Halting

Instructions

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Load	LDRB	<Rd>	<Rn>	<Rm>			Rd = zeroExt(Mem[Rm + Rn][8:0])									
	0	1	0	1	0	1	1	Rm			Rn			Rd		
Store	STRB	<Rd>	<Rn>	<Rm>			Mem[Rm+Rn]=Rd[8:0]									
	0	1	0	1	1	1	1	Rm			Rn			Rd		
Add	ADC	<Rm>	<Rn>				Rd = Rd + Rm +C							N, Z, C, V		
	0	1	0	0	0	0	0	1	0	0	Rm			Rd		
Sub	SUB	<Rd>	<Rn>	#<8_bit_imm>			Subtract Imm value from Rd and store the value in Rd							N, Z, C, V		
	0	0	1	0	1	Rd					Imm					
Jump	B	<target address>				PC = PC + (signExt(offset)<<1)										
	1	1	1	0	0	Offset										
Shift	LSR	<Rd>	<Rm>	#<5_bit_imm>		Shift Rm right by Imm and store in Rd								N, Z, C		
	0	0	0	0	1	immediate					Rm			Rd		
XOR	XOR	<Rm>	<Rn>				Rd = Rd XOR Rm							N, Z		
	0	1	0	0	0	0	1	1	0	1	Rm			Rd		
Branch	B	CS	#<8_bit_offset>				if cond then PC = PC + (signExt(offset)<<1)							Cond. is C flag		
	1	1	0	1	0	1	1	0	Offset							
	Exceptions(Arithmetic Overflow and Undefined Instruction)															

Question #3

Memory Specifications	
Cache Size	1KB
Cache Line Size	8B
Associativity	8
Write Policy	Write Back
Replacement policy (Victim Cache)	LRU Counter
Cache Type	Conventional

Instructions

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Load	LDRH	<Rd>	<Rn>	#<5_bit_offset>													Rd = zeroExtend(Mem[Rn + Offset *2])
	0	1	1	1	1	Offset					Rn			Rd			
Store	STRH	<Rd>	<Rn>	#<5_bit_offset>													Mem[Rn+ Offset *2] = Rd[15:0]
	0	1	1	1	0	Offset					Rn			Rd			
Add	ADD	<Rd>	<Rn>	#<8_bit_imm>													Add Imm value to Rd and store the value in Rd
	0	0	1	0	0	Rd			Imm								N, Z, C, V
Sub	SBC	<Rm>	<Rn>														Rd = Rd – Rm – C
	0	1	0	0	0	0	0	1	0	1	Rm			Rd			N, Z, C, V
Jump	B	<target address>															PC = PC + (signExt(offset)<<1)
	1	1	1	0	0	Offset											
Shift	ASR	<Rm>	<Rn>														Rd = Rd (Arithmetic)>> Rm
	0	1	0	0	0	0	1	0	1	0	Rm			Rd			N, Z, C
BIC	BIC	<Rm>	<Rn>														Rd = Rd AND NOT Rm
	0	1	0	0	0	0	1	1	1	1	Rm			Rd			N, Z
B	B	LE	#<8_bit_offset>														if cond then PC = PC + (signExt(offset)<<1)
	1	1	0	1	0	0	1	0	Offset								Cond. Is N flag

Exceptions(Arithmetic Overflow and Undefined Instruction)

Question #4

Memory Specifications	
Cache Size	1KB
Cache Line Size	8B
Associativity	Fully
Write Policy	Write Back
Replacement policy	LRU Counter
Cache Type	Conventional

Instructions

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Load	LDRB	<Rd>	<Rn>	<Rm>													Rd = zeroExt(Mem[Rm + Rn][8:0])
	0	1	0	1	0	1	1	Rm			Rn			Rd			
Store	STRB	<Rd>	<Rn>	<Rm>													Mem[Rm+Rn]=Rd[8:0]
	0	1	0	1	1	1	1	Rm			Rn			Rd			
Add	Add	<Rd>	<Rn>	#<3_bit_imm>													Add Rn and Imm store result in Rd N, Z, C, V
	0	0	0	1	1	1	0	Imm			Rn			Rd			
Sub	SBC	<Rm>	<Rn>														Rd = Rd - Rm - C N, Z, C, V
	0	1	0	0	0	0	0	1	0	1	Rm			Rd			
Jump	B	<target address>															PC = PC + (signExt(offset)<<1)
	1	1	1	0	0	Offset											
LSR	LSR	<Rm>	<Rn>														Rd = Rd >> Rm N, Z, C
	0	1	0	0	0	0	1	0	0	1	Rm			Rd			
NEG	NEG	<Rm>	<Rn>														Rd = - Rm N, Z, C, V
	0	1	0	0	0	0	0	1	1	0	Rm			Rd			
B	B	CC	#<8_bit_offset>														if cond then PC = PC + (signExt(offset)<<1) Checks if no carry
	1	1	0	1	0	1	1	1	Offset								

Exceptions(Arithmetic Overflow and Undefined Instruction)

Question #5

Memory Specifications	
Cache Size	1KB
Cache Line Size	8B
Associativity	2
Write Policy	Write Through
Replacement policy	FIFO
Cache Type	Way Prediction

Instructions

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Load	LDRB	<Rd>	<Rn>	#<5_bit_offset>		Rd = zeroExtend(Mem[Rn + Offset])										
	0	1	1	0	1	Offset					Rn			Rd		
Store	STRB	<Rd>	<Rn>	#<5_bit_offset>		Mem[Rn + Offset] = Rd[8:0]										
	0	1	1	0	0	Offset					Rn			Rd		
Add	Add	<Rd>	<Rn>	#<3_bit_imm>		Add Rn and Imm store result in Rd								N, Z, C, V		
	0	0	0	1	1	1	0	Imm			Rn			Rd		
Sub	SUB	<Rd>	<Rn>	#<3_bit_imm>		Subtract Imm from Rn store result in Rd								N, Z, C, V		
	0	0	0	1	1	1	1	Imm			Rn			Rd		
Jump	B	<target address>				PC = PC + (signExt(offset)<<1)										
	1	1	1	0	0	Offset										
AND	AND	<Rm>	<Rn>			Rd = Rd AND Rm								N, Z		
	0	1	0	0	0	0	1	1	0	0	Rm			Rd		
B	B	EQ	#<8_bit_offset>			if cond then PC = PC + (signExt(offset)<<1)					Check for Zero flag					
	1	1	0	1	0	1	0		Offset							
CMP	CMP	<Rd>	<Rn>	#<8_bit_imm>		Compare the Imm value to Rd					N, Z, C, V					
	0	0	1	1	1	Rd			Imm							

Exceptions(Arithmetic Overflow and Undefined Instruction)

Question #6

Memory Specifications	
Cache Size	1KB
Cache Line Size	8B
Associativity	2
Write Policy	Write Through
Replacement policy	FIFO
Cache Type	Way Halting

Instructions

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Load	LDR	<Rd>	<Rn>	<Rm>			Rd = Mem[Rm + Rn]				Used for pointers					
	0	1	0	1	0	0	0	Rm			Rn			Rd		
Store	STR	<Rd>	<Rn>	<Rm>			Mem[Rm+Rn]=Rd									
	0	1	0	1	1	0	1	Rm			Rn			Rd		
Add	Add	<Rd>	<Rn>	<Rm>			Add Rn and Rm store result in Rd								N, Z, C, V	
	0	0	0	1	1	0	0	Rm			Rn			Rd		
Sub	SUB	<Rd>	<Rn>	#<8_bit_imm>			Subtract Imm value from Rd and store the value in Rd								N, Z, C, V	
	0	0	1	0	1	Rd					Imm					
Shift	ASR	<Rd>	<Rm>	#<5_bit_imm>			Arithmetic shift Rm right by Imm and store in Rd								N, Z, C	
	0	0	0	1	0	Imm					Rm			Rd		
Jump	B	<target address>					PC = PC + (signExt(offset)<<1)									
	1	1	1	0	0	Offset										
NEG	NEG	<Rm>	<Rn>				Rd = - Rm								N, Z, C, V	
	0	1	0	0	0	0	0	1	1	0	Rm			Rd		
B	B	VS	#<8_bit_offset>				if cond then PC = PC + (signExt(offset)<<1)				Check for Overflow Flag					
	1	1	0	1	1	0	0	1	Offset							

Exceptions(Arithmetic Overflow and Undefined Instruction)

Question #7

Memory Specifications	
Cache Size	1KB
Cache Line Size	8B
Associativity	8
Write Policy	Write Through
Replacement policy	LRU Counter
Cache Type	Conventional

Instructions

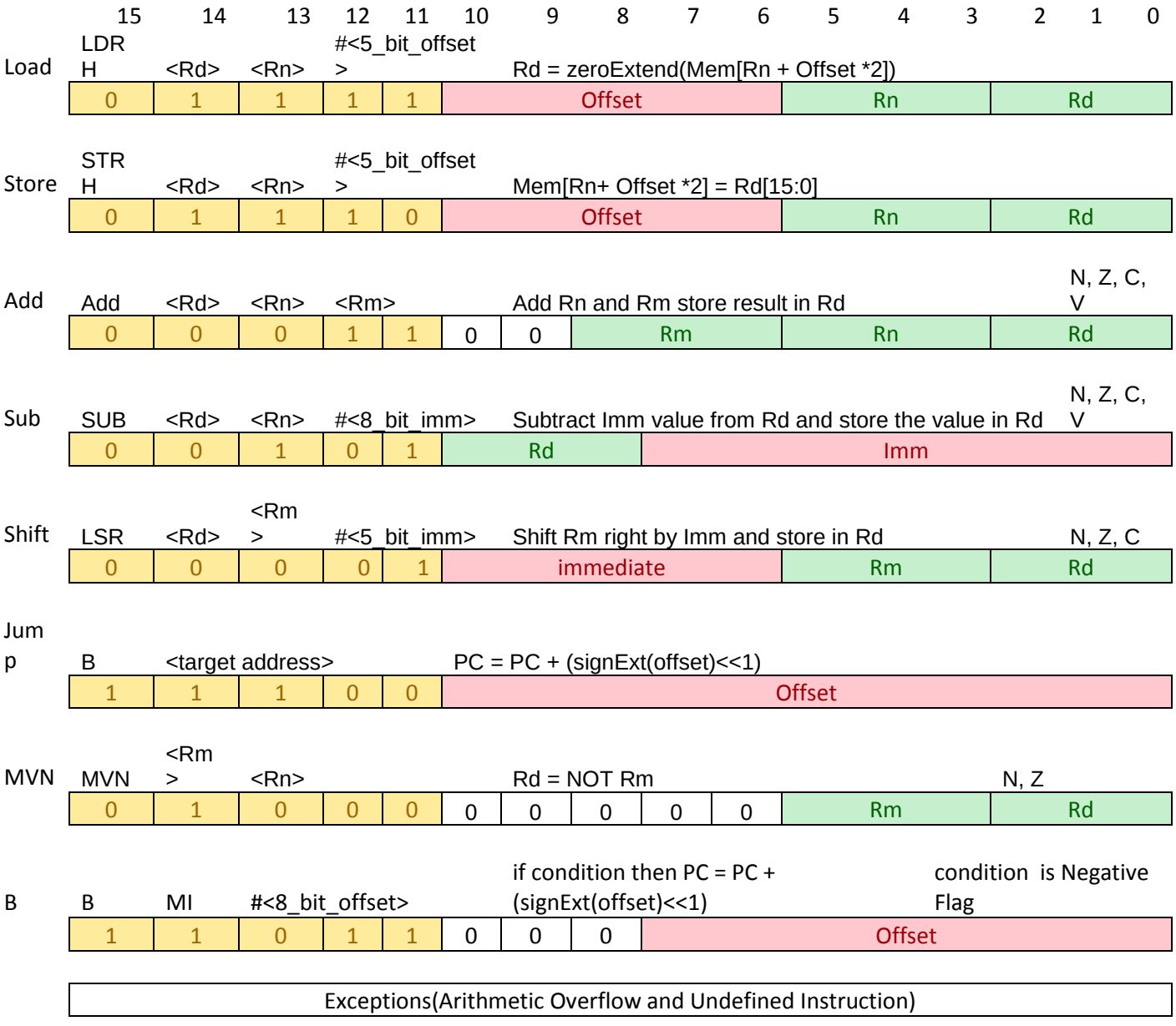
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Load	LDR	<Rd>	<Rn>	#<5_bit_offset>		Rd = Mem[Rn + Offset *4]										
	1	0	0	0	1	Offset					Rn			Rd		
Store	STR	<Rd>	<Rn>	#<5_bit_offset>		Mem[Rn + Offset *4] = Rd										
	1	0	0	0	0	Offset					Rn			Rd		
Add	ADD	<Rd>	<Rn>	#<8_bit_imm>		Add Imm value to Rd and store the value in Rd								N, Z, C, V		
	0	0	1	0	0	Rd			Imm							
Sub	SUB	<Rd>	<Rn>	<Rm>		Subtract contents of Rm from Rn store result in Rd								N, Z, C, V		
	0	0	0	1	1	0	1	Rm			Rn			Rd		
Shift	LSL	<Rm>	<Rn>			Rd = Rd << Rm								N, Z, C		
	0	1	0	0	0	0	1	0	0	0	Rm			Rd		
Jump	B	<target address>				PC = PC + (signExt(offset)<<1)										
	1	1	1	0	0	Offset										
CMP	CMP	<Rm>	<Rn>			Rd - Rm								N, Z, C, V	will set z=1 if equal else n=1 if Rd < Rm or n = 0 if Rm < Rd	
	0	1	0	0	0	0	0	0	0	1	Rm			Rd		
B	B	GE	#<8_bit_offset>			if cond then PC = PC + (signExt(offset)<<1)								condition = is Negative flag		
	1	1	0	1	0	0	0	0	Offset							

Exceptions(Arithmetic Overflow and Undefined Instruction)

Question #8

Memory Specifications	
Cache Size	1KB
Cache Line Size	8B
Associativity	Fully
Write Policy	Write Through
Replacement policy	LRU Counter
Cache Type	Conventional

Instructions



Question #9

Memory Specifications	
Cache Size	512B
Cache Line Size	16B
Associativity	2
Write Policy	Write Back
Replacement policy	FIFO
Cache Type	Way Prediction

Instructions

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Load	LDRB	<Rd>	<Rn>	#<5_bit_offset>													Rd = zeroExtend(Mem[Rn + Offset])
	0	1	1	0	1	Offset					Rn			Rd			
Store	STRB	<Rd>	<Rn>	#<5_bit_offset>													Mem[Rn + Offset] = Rd[8:0]
	0	1	1	0	0	Offset					Rn			Rd			
Add	Add	<Rd>	<Rn>	#<3_bit_imm>													Add Rn and Imm store result in Rd N, Z, C, V
	0	0	0	1	1	1	0	Imm			Rn			Rd			
Sub	SBC	<Rm>	<Rn>														Rd = Rd - Rm - C N, Z, C, V
	0	1	0	0	0	0	0	1	0	1	Rm			Rd			
Shift	ASR	<Rm>	<Rn>														Rd = Rd (Arithmetic)>> Rm N, Z, C
	0	1	0	0	0	0	1	0	1	0	Rm			Rd			
Jump	B	<target address>															PC = PC + (signExt(offset)<<1)
	1	1	1	0	0	Offset											
MOV	MOV	<Rd>	<Rn>	#<8_bit_imm>													Move Imm value to Rd N, Z,
	0	0	1	1	0	Rd			Imm								
B	B	CC	#<8_bit_offset>														if condition then PC = PC + (signExt(offset)<<1) condition is Carry Flag
	1	1	0	1	0	1	1	1	Offset								

Exceptions(Arithmetic Overflow and Undefined Instruction)

Question #10

Memory Specifications	
Cache Size	512B
Cache Line Size	16B
Associativity	2
Write Policy	Write Back
Replacement policy	FIFO
Cache Type	Way Halting

Instructions

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Load	LDR	<Rd>	<Rn>	<Rm>			Rd = Mem[Rm + Rn]				Used for pointers						
	0	1	0	1	0	0	0	Rm			Rn			Rd			
Store	STR	<Rd>	<Rn>	<Rm>			Mem[Rm+Rn]=Rd										
	0	1	0	1	1	0	1	Rm			Rn			Rd			
Add	ADC	<Rm>	<Rn>				Rd = Rd + Rm +C							N, Z, C, V			
	0	1	0	0	0	0	0	1	0	0	Rm			Rd			
Sub	SUB	<Rd>	<Rn>	#<3_bit_imm>			Subtract Imm from Rn store result in Rd							N, Z, C, V			
	0	0	0	1	1	1	1	Imm			Rn			Rd			
Shift	ROR	<Rm>	<Rn>				Rd = Rd (Circular)>>Rm							N, Z, C			
	0	1	0	0	0	0	1	0	1	1	Rm			Rd			
Jump	B	<target address>					PC = PC + (signExt(offset)<<1)										
	1	1	1	0	0	Offset											
ORR	ORR	<Rm>	<Rn>				Rd = Rd OR Rm							N, Z			
	0	1	0	0	0	0	1	1	1	0	Rm			Rd			
B	B	MI	#<8_bit_offset>				if cond then PC = PC + (signExt(offset)<<1)				condition is Negative flag						
	1	1	0	1	1	0	0	0	Offset								

Exceptions(Arithmetic Overflow and Undefined Instruction)

Question #11

Memory Specifications	
Cache Size	512B
Cache Line Size	8B
Associativity	8
Write Policy	Write Back
Replacement policy	LRU Counter
Cache Type	Conventional

Instructions

		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Load	LDRB <Rd> <Rn> <Rm>	0	1	0	1	0	1	1	Rm			Rn			Rd		
	Rd = zeroExt(Mem[Rm + Rn][8:0])																
Store	STRB <Rd> <Rn> <Rm>	0	1	0	1	1	1	1	Rm			Rn			Rd		
	Mem[Rm+Rn]=Rd[8:0]																
Add	ADC <Rm> <Rn>	0	1	0	0	0	0	0	1	0	0	Rm			Rd		
	Rd = Rd + Rm +C																
	N, Z, C, V																
Sub	SUB <Rd> <Rn> #<8_bit_imm>	0	0	1	0	1	Rd					Imm					
	Subtract Imm value from Rd and store the value in Rd																
	N, Z, C, V																
Shift	ASR <Rd> <Rm> #<5_bit_imm>	0	0	0	1	0	immediate					Rm			Rd		
	Arithmetic shift Rm right by Imm and store in Rd																
	N, Z, C																
Jump	BL <target address>	1	1	1	1	0	Offset										
	PC = PC + (signExt(offset)<<1)																
CMN	CMN <Rm> <Rn>	0	1	0	0	0	0	0	0	1	0	Rm			Rd		
	Rd + Rm (aluout = Rd + Rm)																
	N, Z, C, V																
B	B VC #<8_bit_offset>	1	1	0	1	1	0	1	0	Offset							
	if cond then PC = PC + (signExt(offset)<<1)																
	cond. is V flag																

Exceptions(Arithmetic Overflow and Undefined Instruction)

Question #12

Memory Specifications	
Cache Size	512B
Cache Line Size	8B
Associativity	Fully
Write Policy	Write Back
Replacement policy	LRU Counter
Cache Type	Conventional

Instructions

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Load	LDRB	<Rd>	<Rn>	#<5_bit_offset>													Rd = zeroExtend(Mem[Rn + Offset])
	0	1	1	0	1	Offset					Rn			Rd			
Store	STRB	<Rd>	<Rn>	#<5_bit_offset>													Mem[Rn + Offset] = Rd[8:0]
	0	1	1	0	0	Offset					Rn			Rd			
Add	ADD	<Rd>	<Rn>	#<8_bit_imm>													Add Imm value to Rd and store the value in Rd N, Z, C, V
	0	0	1	0	0	Rd			Imm								
Sub	SUB	<Rd>	<Rn>	#<8_bit_imm>													Subtract Imm value from Rd and store the value in Rd N, Z, C, V
	0	0	1	0	1	Rd			Imm								
Jump	BL	<target address>															PC = PC + (signExt(offset)<<1)
	1	1	1	1	0	Offset											
ROR	ROR	<Rm>	<Rn>														Rd = Rd (Circular)>>Rm N, Z, C
	0	1	0	0	0	0	1	0	1	1	Rm			Rd			
AND	AND	<Rm>	<Rn>														Rd = Rd AND Rm N, Z
	0	1	0	0	0	0	1	1	0	0	Rm			Rd			
B	B	MI	#<8_bit_offset>														if cond then PC = PC + (signExt(offset)<<1) condition is Negative flag
	1	1	0	1	1	0	0	0	Offset								

Exceptions(Arithmetic Overflow and Undefined Instruction)

Question #13

Memory Specifications	
Cache Size	512B
Cache Line Size	16B
Associativity	2
Write Policy	Write Through
Replacement policy	FIFO
Cache Type	Way Prediction

Instructions

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Load	LDRH	<Rd>	<Rn>	<Rm>			Rd =zeroExt(Mem[Rm + Rn][16:0])									
	0	1	0	1	0	0	1	Rm			Rn			Rd		
Store	STRH	<Rd>	<Rn>	<Rm>			Mem[Rm+Rn]=Rd[16:0]									
	0	1	0	1	1	1	0	Rm			Rn			Rd		
Add	Add	<Rd>	<Rn>	#<3_bit_imm>			Add Rn and Imm store result in Rd							N, Z, C, V		
	0	0	0	1	1	1	0	Imm			Rn			Rd		
Sub	SUB	<Rd>	<Rn>	#<3_bit_imm>			Subtract Imm from Rn store result in Rd							N, Z, C, V		
	0	0	0	1	1	1	1	Imm			Rn			Rd		
Shift	LSL	<Rd>	<Rm>	#<5_bit_imm>			Shift Rm left by Imm and store in Rd								N, Z, C	
	0	0	0	0	0	immediate					Rm			Rd		
Jump	BL	<target address>					PC = PC + (signExt(offset)<<1)									
	1	1	1	1	0	Offset										
MUL	MUL	<Rm>	<Rn>				Rd = Rd * Rm							N, Z		
	0	1	0	0	0	0	0	1	1	1	Rm			Rd		
B	B	VS	#<8_bit_offset>				if cond then PC = PC + (signExt(offset)<<1)				condition is overflow set					
	1	1	0	1	1	0	0	1	Offset							

Exceptions(Arithmetic Overflow and Undefined Instruction)

Question #14

Memory Specifications	
Cache Size	512B
Cache Line Size	16B
Associativity	2
Write Policy	Write Through
Replacement policy	FIFO
Cache Type	Way Halting

Instructions

		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Load	LDRH <Rd> <Rn> #<5_bit_offset> Rd = zeroExtend(Mem[Rn + Offset *2])	0	1	1	1	1	Offset					Rn			Rd		
Store	STRH <Rd> <Rn> #<5_bit_offset> Mem[Rn+ Offset *2] = Rd[15:0]	0	1	1	1	0	Offset					Rn			Rd		
Add	ADC <Rm> <Rn> Rd = Rd + Rm +C N, Z, C, V	0	1	0	0	0	0	0	1	0	0	Rm			Rd		
Sub	SUB <Rd> <Rn> #<3_bit_imm> Subtract Imm from Rn store result in Rd N, Z, C, V	0	0	0	1	1	1	1	Imm			Rn			Rd		
Shift	LSL <Rm> <Rn> Rd = Rd << Rm N, Z, C	0	1	0	0	0	0	1	0	0	0	Rm			Rd		
Jump	BL <target address> PC = PC + (signExt(offset)<<1)	1	1	1	1	0	Offset										
ROR	ROR <Rm> <Rn> Rd = Rd (Circular)>>Rm N, Z, C	0	1	0	0	0	0	1	0	1	1	Rm			Rd		
B	B CS #<8_bit_offset> if cond then PC = PC + (signExt(offset)<<1) condition is if CARRY flag set	1	1	0	1	0	1	1	0	Offset							

Exceptions(Arithmetic Overflow and Undefined Instruction)

Question #15

Memory Specifications	
Cache Size	512B
Cache Line Size	16B
Associativity	8
Write Policy	Write Through
Replacement policy	LRU Counter
Cache Type	Conventional

Instructions

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Load	LDR	<Rd>	<Rn>	<Rm>			Rd = Mem[Rm + Rn]				Used for pointers					
	0	1	0	1	0	0	0	Rm			Rn			Rd		
Store	STR	<Rd>	<Rn>	<Rm>			Mem[Rm+Rn]= Rd									
	0	1	0	1	1	0	1	Rm			Rn			Rd		
Add	ADD	<Rd>	<Rn>	#<8_bit_imm>			Add Imm value to Rd and store the value in Rd							N, Z, C, V		
	0	0	1	0	0	Rd					Imm					
Sub	SUB	<Rd>	<Rn>	<Rm>			Subtract contents of Rm from Rn store result in Rd							N, Z, C, V		
	0	0	0	1	1	0	1	Rm			Rn			Rd		
Jump	BL	<target address>					PC = PC + (signExt(offset)<<1)									
	1	1	1	1	0	Offset										
CMP	CMP	<Rm>	<Rn>				Rd - Rm (aluout = Rd - Rm)						N, Z, C, V	will set z=1 if equal else n=1 if Rd < Rm or n = 0 if Rm < Rd		
	0	1	0	0	0	0	0	0	0	1	Rm			Rd		
B	B	GT	#<8_bit_offset>				if cond then PC = PC + (signExt(offset)<<1)		check if NEGATIVE flag clear							
	1	1	0	1	0	0	0	1	Offset							
EOR	EOR	<Rm>	<Rn>				Rd = Rd XOR Rm						N, Z			
	0	1	0	0	0	0	1	1	0	1	Rm			Rd		

Exceptions(Arithmetic Overflow and Undefined Instruction)

Question #16

Memory Specifications	
Cache Size	512B
Cache Line Size	16B
Associativity	Fully
Write Policy	Write Through
Replacement policy	LRU Counter
Cache Type	Conventional

Instructions

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Load	LDR	<Rd>	<Rn>	#<5_bit_offset>													Rd = Mem[Rn + Offset *4]
	1	0	0	0	1	Offset						Rn			Rd		
Store	STR	<Rd>	<Rn>	#<5_bit_offset>													Mem[Rn + Offset *4] = Rd
	1	0	0	0	0	Offset						Rn			Rd		
Add	ADD	<Rd>	<Rn>	#<8_bit_imm>													Add Imm value to Rd and store the value in Rd N, Z, C, V
	0	0	1	0	0	Rd			Imm								
Sub	SUB	<Rd>	<Rn>	#<8_bit_imm>													Subtract Imm value from Rd and store the value in Rd N, Z, C, V
	0	0	1	0	1	Rd			Imm								
Shift	LSR	<Rd>	<Rm>	#<5_bit_imm>													Shift Rm right by Imm and store in Rd N, Z, C
	0	0	0	0	1	immediate						Rm			Rd		
Jump	BL	<target address>															PC = PC + (signExt(offset)<<1)
	1	1	1	1	0	Offset											
TST	TST	<Rm>	<Rn>														Check if bit at pos (Rm) is set in Rd Z
	0	1	0	0	0	0	0	0	1	1		Rm			Rd		
B	B	EQ	#<8_bit_offset>														if cond then PC = PC + (signExt(offset)<<1) check if ZERO flag clear
	1	1	0	1	0	1	0	1	Offset								

Exceptions(Arithmetic Overflow and Undefined Instruction)

Question #17

Memory Specifications	
Cache Size	512B
Cache Line Size	32B
Associativity	2
Write Policy	Write Back
Replacement policy	FIFO
Cache Type	Way Prediction

Instructions

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Load	LDRH	<Rd>	<Rn>	#<5_bit_offset>													Rd = zeroExtend(Mem[Rn + Offset *2])
	0	1	1	1	1	Offset					Rn			Rd			
Store	STRH	<Rd>	<Rn>	#<5_bit_offset>													Mem[Rn+ Offset *2] = Rd[15:0]
	0	1	1	1	0	Offset					Rn			Rd			
Add	Add	<Rd>	<Rn>	<Rm>													Add Rn and Rm store result in Rd N, Z, C, V
	0	0	0	1	1	0	0	Rm			Rn			Rd			
Sub	SUB	<Rd>	<Rn>	<Rm>													Subtract contents of Rm from Rn store result in Rd N, Z, C, V
	0	0	0	1	1	0	1	Rm			Rn			Rd			
Shift	ASR	<Rd>	<Rm>	#<5_bit_imm>													Arithmetic shift Rm right by Imm and store in Rd N, Z, C
	0	0	0	1	0	immediate					Rm			Rd			
Jump	BL	<target address>															PC = PC + (signExt(offset)<<1)
	1	1	1	1	0	Offset											
TST	TST	<Rm>	<Rn>														Check if bit at pos (Rm) is set in Rd Z
	0	1	0	0	0	0	0	0	1	1	Rm			Rd			
B	B	NE	#<8_bit_offset>														if cond then PC = PC + (signExt(offset)<<1) check if NEGATVIE flag clear
	1	1	0	1	0	1	0	1	Offset								
BIC	BIC	<Rm>	<Rn>														Rd = Rd AND NOT Rm N, Z
	0	1	0	0	0	0	1	1	1	1	Rm			Rd			

Exceptions(Arithmetic Overflow and Undefined Instruction)

Question #18

Memory Specifications	
Cache Size	512B
Cache Line Size	32B
Associativity	2
Write Policy	Write Back
Replacement policy	FIFO
Cache Type	Way Halting

Instructions

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Load	LDRH	<Rd>	<Rn>	#<5_bit_offset>													Rd = zeroExtend(Mem[Rn + Offset *2])
	0	1	1	1	1	Offset					Rn			Rd			
Store	STRH	<Rd>	<Rn>	#<5_bit_offset>													Mem[Rn+ Offset *2] = Rd[15:0]
	0	1	1	1	0	Offset					Rn			Rd			
Add	ADD	<Rd>	<Rn>	#<8_bit_imm>													Add Imm value to Rd and store the value in Rd N, Z, C, V
	0	0	1	0	0	Rd			Imm								
Sub	SUB	<Rd>	<Rn>	#<8_bit_imm>													Subtract Imm value from Rd and store the value in Rd N, Z, C, V
	0	0	1	0	1	Rd			Imm								
Shift	LSR	<Rm>	<Rn>														Rd = Rd >> Rm N, Z, C
	0	1	0	0	0	0	1	0	0	1	Rm			Rd			
Jump	BL	<target address>															PC = PC + (signExt(offset)<<1)
	1	1	1	1	0	Offset											
MVN	MVN	<Rm>	<Rn>														Rd = NOT Rm N, Z
	0	1	0	0	0	0	0	0	0	0	Rm			Rd			
B	B	CS	#<8_bit_offset>														if cond then PC = PC + (signExt(offset)<<1) Cond. flag is carry
	1	1	0	1	0	1	1	0	Offset								

Exceptions(Arithmetic Overflow and Undefined Instruction)

Question #19

Memory Specifications	
Cache Size	512B
Cache Line Size	32B
Associativity	8
Write Policy	Write Back
Replacement policy	LRU Counter
Cache Type	Conventional

Instructions

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Load	LDRH	<Rd>	<Rn>	<Rm>				Rd =zeroExt(Mem[Rm + Rn][16:0])									
	0	1	0	1	0	0	1	Rm			Rn			Rd			
Store	STRH	<Rd>	<Rn>	<Rm>				Mem[Rm+Rn]=Rd[16:0]									
	0	1	0	1	1	1	0	Rm			Rn			Rd			
Add	Add	<Rd>	<Rn>	#<3_bit_imm>				Add Rn and Imm store result in Rd							N, Z, C, V		
	0	0	0	1	1	1	0	Imm			Rn			Rd			
Sub	SUB	<Rd>	<Rn>	#<3_bit_imm>				Subtract Imm from Rn store result in Rd							N, Z, C, V		
	0	0	0	1	1	1	1	Imm			Rn			Rd			
Shift	LSL	<Rd>	<Rm>	#<5_bit_imm>				Shift Rm left by Imm and store in Rd							N, Z, C		
	0	0	0	0	0	immediate					Rm			Rd			
Jump	BL	<target address>						PC = PC + (signExt(offset)<<1)									
	1	1	1	1	0	Offset											
ORR	ORR	<Rm>	<Rn>					Rd = Rd OR Rm							N, Z		
	0	1	0	0	0	0	1	1	1	0	Rm			Rd			
B	B	EQ	#<8_bit_offset>					if cond then PC = PC + (signExt(offset)<<1)							Cond. flag is Zero		
	1	1	0	1	0	1	0	1	Offset								

Exceptions(Arithmetic Overflow and Undefined Instruction)

Question #20

Memory Specifications	
Cache Size	512B
Cache Line Size	32B
Associativity	Fully
Write Policy	Write Back
Replacement policy	LRU Counter
Cache Type	Conventional

Instructions

		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Load	LDRB <Rd> <Rn> #<5_bit_offset> Rd = zeroExtend(Mem[Rn + Offset])	0	1	1	0	1	Offset					Rn			Rd		
Store	STRB <Rd> <Rn> #<5_bit_offset> Mem[Rn + Offset] = Rd[8:0]	0	1	1	0	0	Offset					Rn			Rd		
Add	Add <Rd> <Rn> <Rm> Add Rn and Rm store result in Rd N, Z, C, V	0	0	0	1	1	0	0	Rm			Rn			Rd		
Sub	SUB <Rd> <Rn> <Rm> Subtract contents of Rm from Rn store result in Rd N, Z, C, V	0	0	0	1	1	0	1	Rm			Rn			Rd		
Shift	LSR <Rd> <Rm> #<5_bit_imm> Shift Rm right by Imm and store in Rd N, Z, C	0	0	0	0	1	immediate					Rm			Rd		
Jump	BL <target address> PC = PC + (signExt(offset)<<1)	1	1	1	1	0	Offset										
AND	AND <Rm> <Rn> Rd = Rd AND Rm N, Z	0	1	0	0	0	0	1	1	0	0	Rm			Rd		
B	B NE #<8_bit_offset> if cond then PC = PC + (signExt(offset)<<1) Cond. is Z flag	1	1	0	1	0	1	0	1	Offset							

Exceptions(Arithmetic Overflow and Undefined Instruction)

Question #21

Memory Specifications	
Cache Size	512B
Cache Line Size	32B
Associativity	2
Write Policy	Write Through
Replacement policy	FIFO
Cache Type	Way Prediction

Instructions

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Load	LDRB	<Rd>	<Rn>	#<5_bit_offset>			Rd = zeroExtend(Mem[Rn + Offset])									
	Rn			Rd			Offset					0	1	1	0	1
Store	STRB	<Rd>	<Rn>	#<5_bit_offset>			Mem[Rn + Offset] = Rd[8:0]									
	Rn			Rd			Offset					0	1	1	0	0
Add	Add	<Rd>	<Rn>	<Rm>			Add Rn and Rm store result in Rd								N, Z, C, V	
	Rm			Rn			Rd			0	0	0	0	0	1	1
Sub	SUB	<Rd>	<Rn>	<Rm>			Subtract contents of Rm from Rn store result in Rd								N, Z, C, V	
	Rm			Rn			Rd			0	1	0	0	0	1	1
Shift	ROR	<Rm>	<Rn>				Rd = Rd (Circular)>>Rm							N, Z, C		
	Rm			Rd			0	1	0	1	1	0	1	0	0	0
Jump	B	<target address>					PC = PC + (signExt(offset)<<1)									
	Offset											1	1	1	0	0
ORR	ORR	<Rm>	<Rn>				Rd = Rd OR Rm							N, Z		
	Rm			Rd			0	1	1	1	0	0	1	0	0	0
B	B	MI	#<8_bit_offset>				if cond then PC = PC + (signExt(offset)<<1)					condition is Negative flag				
	Offset									0	0	0	1	1	0	1	1

Exceptions(Arithmetic Overflow and Undefined Instruction)

Question #22

Memory Specifications	
Cache Size	512B
Cache Line Size	32B
Associativity	2
Write Policy	Write Through
Replacement policy	FIFO
Cache Type	Way Halting

Instructions

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Load	LDRB	<Rd>	<Rn>	<Rm>			Rd = zeroExt(Mem[Rm + Rn][8:0])									
	Rm			Rn			Rd			1	1	0	1	0	1	0
Store	STRB	<Rd>	<Rn>	<Rm>			Mem[Rm+Rn]=Rd[8:0]									
	Rm			Rn			Rd			1	1	0	1	0	1	1
Add	ADC	<Rm>	<Rn>				Rd = Rd + Rm +C							N, Z, C, V		
	Rm			Rd			0	0	1	0	0	0	1	0	0	0
Sub	SUB	<Rd>	<Rn>	#<8_bit_imm>			Subtract Imm value from Rd and store the value in Rd								N, Z, C, V	
	Imm								Rd			0	0	1	0	1
Jump	BL	<target address>					PC = PC + (signExt(offset)<<1)									
	Offset											1	1	1	1	0
ROR	ROR	<Rm>	<Rn>				Rd = Rd (Circular)>>Rm							N, Z, C		
	Rm			Rd			0	1	0	1	1	0	1	0	0	0
AND	AND	<Rm>	<Rn>				Rd = Rd AND Rm							N, Z		
	Rm			Rd			0	1	1	0	0	0	1	0	0	0
B	B	MI	#<8_bit_offset>				if cond then PC = PC + (signExt(offset)<<1)					condition is Negative flag				
	Offset								0	0	0	1	1	0	1	1

Exceptions(Arithmetic Overflow and Undefined Instruction)

Question #23

Memory Specifications	
Cache Size	512B
Cache Line Size	32B
Associativity	8
Write Policy	Write Through
Replacement policy	LRU Counter
Cache Type	Conventional

Instructions

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Load	LDRH	<Rd>	<Rn>	#<5_bit_offset>													Rd = zeroExtend(Mem[Rn + Offset *2])
	Offset						Rn			Rd		0	1	1	1	1	
Store	STRH	<Rd>	<Rn>	#<5_bit_offset>													Mem[Rn+ Offset *2] = Rd[15:0]
	Offset						Rn			Rd		0	1	1	1	0	
Add	ADD	<Rd>	<Rn>	#<8_bit_imm>													Add Imm value to Rd and store the value in Rd N, Z, C, V
	Rd			Imm								0	0	1	0	0	
Sub	SBC	<Rm>	<Rn>														Rd = Rd - Rm - C N, Z, C, V
	Rm			Rd			0	0	1	0	1	0	1	0	0	0	
Shift	LSL	<Rd>	<Rm>	#<5_bit_imm>													Shift Rm left by Imm and store in Rd N, Z, C
	immediate						Rm			Rd		0	0	0	0	0	
Jump	BL	<target address>															PC = PC + (signExt(offset)<<1)
	Offset											1	1	1	1	0	
Mul	MUL	<Rm>	<Rn>														Rd = Rd * Rm N, Z
	Rm			Rd			0	0	1	1	1	0	1	0	0	0	
B	B	VS	#<8_bit_offset>														if cond then PC = PC + (signExt(offset)<<1) condition is overflow set
	Offset									0	0	1	1	1	0	1	1

Exceptions(Arithmetic Overflow and Undefined Instruction)

Question #24

Memory Specifications	
Cache Size	512B
Cache Line Size	32B
Associativity	Fully
Write Policy	Write Through
Replacement policy	LRU Counter
Cache Type	Conventional

Instructions

		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Load	LDRH <Rd> <Rn> #<5_bit_offset> Rd = zeroExtend(Mem[Rn + Offset *2])	Rn			Rd			Offset					0	1	1	1	1
Store	STRH <Rd> <Rn> #<5_bit_offset> Mem[Rn+ Offset *2] = Rd[15:0]	Rn			Rd			Offset					0	1	1	1	0
Add	ADC <Rm> <Rn> Rd = Rd + Rm +C N, Z, C, V	Rm			Rd			0	0	1	0	0	0	1	0	0	0
Sub	SUB <Rd> <Rn> #<3_bit_imm> Subtract Imm from Rn store result in Rd N, Z, C, V	Rn			Rd			Imm			1	1	0	0	0	1	1
Jump	B <target address> PC = PC + (signExt(offset)<<1)	Offset											1	1	1	0	0
LSR	LSR <Rm> <Rn> Rd = Rd >> Rm N, Z, C	Rm			Rd			0	1	0	0	1	0	1	0	0	0
NEG	NEG <Rm> <Rn> Rd = - Rm N, Z, C, V	Rm			Rd			0	0	1	1	0	0	1	0	0	0
B	B CC #<8_bit_offset> if cond then PC = PC + (signExt(offset)<<1) Checks if no carry	Offset									1	1	1	1	0	1	0

Exceptions(Arithmetic Overflow and Undefined Instruction)

Question #25

Memory Specifications	
Cache Size	1KB
Cache Line Size	32B
Associativity	2
Write Policy	Write Back
Replacement policy	FIFO
Cache Type	Way Prediction

Instructions

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Load	LDRB	<Rd>	<Rn>	#<5_bit_offset>			Rd = zeroExtend(Mem[Rn + Offset])									
	Offset						Rn			Rd		0	1	1	0	1
Store	STRB	<Rd>	<Rn>	#<5_bit_offset>			Mem[Rn + Offset] = Rd[8:0]									
	Offset						Rn			Rd		0	1	1	0	0
Add	Add	<Rd>	<Rn>	#<3_bit_imm>			Add Rn and Imm store result in Rd							N, Z, C, V		
	Imm				Rn			Rd		1	0	0	0	0	1	1
Sub	SUB	<Rd>	<Rn>	#<3_bit_imm>			Subtract Imm from Rn store result in Rd							N, Z, C, V		
	Imm				Rn			Rd		1	1	0	0	0	1	1
Shift	SUB	<Rd>	<Rn>	<Rm>			Subtract contents of Rm from Rn store result in Rd								N, Z, C, V	
	Rm			Rn			Rd			0	1	0	0	0	1	1
Jump	BL	<target address>					PC = PC + (signExt(offset)<<1)									
	Offset											1	1	1	1	0
CMP	CMP	<Rm>	<Rn>				Rd - Rm (aluout = Rd -Rm)							N, Z, C, V	will set z=1 if equal else n=1 if Rd < Rm or n = 0 if Rm < Rd	
	Rm			Rd			0	0	0	0	1	0	1	0	0	0
B	B	GT	#<8_bit_offset>				if cond then PC = PC + (signExt(offset)<<1)					check if NEGATIVE flag clear				
	Offset									0	0	1	1	1	0	1	0

Exceptions(Arithmetic Overflow and Undefined Instruction)

Question #26

Memory Specifications	
Cache Size	1KB
Cache Line Size	32B
Associativity	2
Write Policy	Write Back
Replacement policy	FIFO
Cache Type	Way Halting

Instructions

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Load	LDR	<Rd>	<Rn>	#<5_bit_offset>			Rd = Mem[Rn + Offset *4]										
	Rn			Rd			Offset					1	0	0	0	1	
Store	STR	<Rd>	<Rn>	#<5_bit_offset>			Mem[Rn + Offset *4] = Rd										
	Rn			Rd			Offset					1	0	0	0	0	
Add	ADD	<Rd>	<Rn>	#<8_bit_imm>			Add Imm value to Rd and store the value in Rd								N, Z, C, V		
	Imm								Rd			0	0	1	0	0	
Sub	SUB	<Rd>	<Rn>	#<8_bit_imm>			Subtract Imm value from Rd and store the value in Rd								N, Z, C, V		
	Imm								Rd			0	0	1	0	1	
Shift	ASR	<Rd>	<Rm>	#<5_bit_imm>			Arithmetic shift Rm right by Imm and store in Rd								N, Z, C		
	Rm			Rd			Imm					0	0	0	1	0	
Jump	B	<target address>					PC = PC + (signExt(offset)<<1)										
	Offset											1	1	1	0	0	
NEG	NEG	<Rm>	<Rn>				Rd = - Rm								N, Z, C, V		
	Rm			Rd			0	0	1	1	0	0	1	0	0	0	
B	B	VS	#<8_bit_offset>				if cond then PC = PC + (signExt(offset)<<1)								Check for Overflow Flag		
	Offset								0	0	1	1	1	0	1	1	

Exceptions(Arithmetic Overflow and Undefined Instruction)

Question #27

Memory Specifications	
Cache Size	1KB
Cache Line Size	32B
Associativity	8
Write Policy	Write Back
Replacement policy	LRU Counter
Cache Type	Conventional

Instructions

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Load	LDR	<Rd>	<Rn>	#<5_bit_offset>													Rd = Mem[Rn + Offset *4]
	Offset						Rn			Rd		1	0	0	0	1	
Store	STR	<Rd>	<Rn>	#<5_bit_offset>													Mem[Rn + Offset *4] = Rd
	Offset						Rn			Rd		1	0	0	0	0	
Add	ADD	<Rd>	<Rn>	#<8_bit_imm>													Add Imm value to Rd and store the value in Rd N, Z, C, V
	Rd			Imm								0	0	1	0	0	
Sub	SUB	<Rd>	<Rn>	<Rm>													Subtract contents of Rm from Rn store result in Rd N, Z, C, V
	Rm			Rn			Rd			0	1	0	0	0	1	1	
Shift	ASR	<Rd>	<Rm>	#<5_bit_imm>													Arithmetic shift Rm right by Imm and store in Rd N, Z, C
	immediate						Rm			Rd		0	0	0	1	0	
Jump	BL	<target address>															PC = PC + (signExt(offset)<<1)
	Offset											1	1	1	1	0	
TST	TST	<Rm>	<Rn>														Check if bit at pos (Rm) is set in Rd Z
	Rm			Rd			0	0	0	1	1	0	1	0	0	0	
B	B	NE	#<8_bit_offset>														if cond then PC = PC + (signExt(offset)<<1) check if NEGATVIE flag clear
	Offset								1	0	1	1	1	0	1	0	

Exceptions(Arithmetic Overflow and Undefined Instruction)

Question #28

Memory Specifications	
Cache Size	1KB
Cache Line Size	32B
Associativity	Fully
Write Policy	Write Back
Replacement policy	LRU Counter
Cache Type	Conventional

Instructions

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Load	LDRH	<Rd>	<Rn>	#<5_bit_offset>													Rd = zeroExtend(Mem[Rn + Offset *2])
	Rn			Rd			Offset					0	1	1	1	1	
Store	STRH	<Rd>	<Rn>	#<5_bit_offset>													Mem[Rn+ Offset *2] = Rd[15:0]
	Rn			Rd			Offset					0	1	1	1	0	
Add	ADD	<Rd>	<Rn>	#<8_bit_imm>													Add Imm value to Rd and store the value in Rd N, Z, C, V
	Imm								Rd			0	0	1	0	0	
Sub	SUB	<Rd>	<Rn>	#<8_bit_imm>													Subtract Imm value from Rd and store the value in Rd N, Z, C, V
	Imm								Rd			0	0	1	0	1	
Shift	LSR	<Rd>	<Rm>	#<5_bit_imm>													Shift Rm right by Imm and store in Rd N, Z, C
	Rm			Rd			immediate					0	0	0	0	1	
Jump	B	<target address>															PC = PC + (signExt(offset)<<1)
	Offset											1	1	1	0	0	
MVN	MVN	<Rm>	<Rn>														Rd = NOT Rm N, Z
	Rm			Rd			0	0	0	0	0	0	1	0	0	0	
B	B	MI	#<8_bit_offset>														if condition then PC = PC + (signExt(offset)<<1) condition is Negative Flag
	Offset								0	0	0	1	1	0	1	1	

Exceptions(Arithmetic Overflow and Undefined Instruction)

Question #29

Memory Specifications	
Cache Size	1KB
Cache Line Size	32B
Associativity	2
Write Policy	Write Through
Replacement policy	FIFO
Cache Type	Way Prediction

Instructions

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Load	LDRB	<Rd>	<Rn>	#<5_bit_offset>								Rd = zeroExtend(Mem[Rn + Offset])					
	Offset						Rn			Rd		0	1	1	0	1	
Store	STRB	<Rd>	<Rn>	#<5_bit_offset>								Mem[Rn + Offset] = Rd[8:0]					
	Offset						Rn			Rd		0	1	1	0	0	
Add	Add	<Rd>	<Rn>	#<3_bit_imm>								Add Rn and Imm store result in Rd			N, Z, C, V		
	Imm				Rn			Rd		1	0	0	0	0	1	1	
Sub	SBC	<Rm>	<Rn>							Rd = Rd - Rm - C					N, Z, C, V		
	Rm			Rd			0	0	1	0	1	0	1	0	0	0	
Shift	LSL	<Rd>	<Rm>	#<5_bit_imm>								Shift Rm left by Imm and store in Rd				N, Z, C	
	immediate						Rm			Rd		0	0	0	0	0	
Jump	BL	<target address>								PC = PC + (signExt(offset)<<1)							
	Offset											1	1	1	1	0	
ORR	ORR	<Rm>	<Rn>							Rd = Rd OR Rm					N, Z		
	Rm			Rd			0	1	1	1	0	0	1	0	0	0	
B	B	EQ	#<8_bit_offset>							if cond then PC = PC + (signExt(offset)<<1)						Cond. flag is Zero	
	Offset								1	0	1	1	1	0	1	0	

Exceptions(Arithmetic Overflow and Undefined Instruction)

Question #30

Memory Specifications	
Cache Size	1KB
Cache Line Size	32B
Associativity	2
Write Policy	Write Through
Replacement policy	FIFO
Cache Type	Way Halting

Instructions