# 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 **04**<sup>th</sup> **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 04<sup>th</sup> 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:-

- o 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

				Men	nory	Specifi	cation	s		
		Cach	ie Siz	$\overline{e}$		1 KB				
		Cach	ne Lin	ne Size		8B				
		Asso	ciativ	vity		2				
		Write	e Poli	icy		Write	Back			
		Repla	acem	ent pol	icy	FIFO				
		Cach	пе Тур	pe		Way F	redicti	on		
struct	ions									
	15   14   13   12	11	10	9	8	7	6	5 4	3	2 1 0
oad	LDRH <rd> <rn> &lt;</rn></rd>				zeroEz		Rm + 1	Rn][16:0])		
	0 1 0 1	0	0	1		Rm		Rn		Rd
ore	STRH <rd> <rn> <l< td=""><td>Rm&gt;</td><td></td><td>Meml</td><td>R<sub>m</sub>+</td><td>Rn]=Rd</td><td>[16:0]</td><td></td><td></td><td></td></l<></rn></rd>	Rm>		Meml	R <sub>m</sub> +	Rn]=Rd	[16:0]			
510	0 1 0 1	1	1	0		Rm	[10.0]	Rn		Rd
				ı						
ld	ADD <rd> PC #&lt;8_b</rd>	it_imm	>	Rd =	(PC A	ND 0xl	FFFFI	FFC) + (imm * 4)	)	
	1 0 1 0	1		Rd				Imm	1	
L	CDC (Day) (Day)			D.J	ו גם	) C			N.	
ıb	SBC <rm> <rn> 0 1 0 0</rn></rm>	0	0	Ru = 0	1	$\frac{\text{Rm - C}}{0}$	1	Rm	IN	N, Z, C, V Rd
		U	U	U	1	1 0	1	KIII		Ku
ift	LSL <rd> <rm> #&lt;5</rm></rd>	bit in	nm>	Shift	Rm le	ft by Im	m and	store in Rd	N	I, Z, C
	0 0 0 0	0			nmedi			Rm		Rd
									•	
mp	B <target address=""></target>			PC =	PC +	(signEx	t(offset	c)<<1)		
	1 1 1 0	0						Offset		
ul	MUL <rm> <rn></rn></rm>			Rd =	Rd *	Rm			N	I, Z
	0 1 0 0	0	0	0	1	1	1	Rm		Rd
1.	DIT# 0 1:4 -ff4			:c	1.41	DC I	O . (-:		1) (	Nam 4141 am 1 a NJ 61 a
anch		0	0			$\frac{1 PC = P}{1}$	C + (S)	gnExt(offset)<<		Condition is N flag
	1 1 0 1	0	U	1	1			Offse	i.	
			• /	A 1.4		CI	1 7 7	defined Instructi		

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

Instruct	ions															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Load	LDRB	<rd></rd>	<rn></rn>	<rm:< td=""><td>&gt;</td><td></td><td>Rd = z</td><td>eroExt(</td><td>Mem[R</td><td>m + Rn]</td><td>[[8:0])</td><td></td><td></td><td></td><td></td><td></td></rm:<>	>		Rd = z	eroExt(	Mem[R	m + Rn]	[[8:0])					
	0	1	0	1	0	1	1		Rm			Rn			Rd	
Store	STRB	<rd></rd>	<rn></rn>	<rm:< td=""><td>&gt;</td><td></td><td>Mem[</td><td>Rm+Rn</td><td>]=Rd[8:0</td><td>0]</td><td></td><td></td><td></td><td></td><td></td><td></td></rm:<>	>		Mem[	Rm+Rn	]=Rd[8:0	0]						
	0	1	0	1	1	1	1		Rm			Rn			Rd	
Add	ADC	<rm></rm>	<rn></rn>				Rd = I	Rd + Rr	n +C					N, Z,	C, V	
	0	1	0	0	0	0	0	1	0	0		Rm			Rd	
								act Im	n value	from	Rd and	store	the val	ue in	N, Z,	C,
Sub	SUB	<rd></rd>	<rn></rn>	#<8_	_bit_ir	nm>	m> Rd									
	0	0	1	0	1		Rd Imm									
Jump	В	<target< td=""><td>address&gt;</td><td>•</td><td></td><td>PC =</td><td colspan="9">PC = PC + (signExt(offset)&lt;&lt;1)</td><td></td></target<>	address>	•		PC =	PC = PC + (signExt(offset)<<1)									
	1	1	1	0	0						Offset					
Shift	LSR	<rd></rd>	<rm></rm>	#<5_	_bit_ir	nm>	nm> Shift Rm right by Imm and store in Rd							N, Z,	С	
	0	0	0	0	1		ir	nmedia	ite			Rm			Rd	

XOR	XOR	<rm></rm>	<rn></rn>				Rd = I	Rd XOF	R Rm			N, Z	
	0	1	0	0	0	0	1	1	0	1	Rm	Rd	
Branch	В	CS	#<8_bi	t_offs	et>		if con	d then	PC = P	C + (się	gnExt(offset)<<1)	Cond. is C flag	
	1	1	0	1	0	1	1	0			Offset		
	Exceptions(Arithmetic Overflow and Undefined Instruction)												

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

Instruc	rtions															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Load	LDRH	<rd></rd>	<rn></rn>	#<5_	bit_of	fset>	Rd =	zeroE	xtend(	Mem	[Rn +	Offset	*2])	I	1	
	0	1	1	1	1		(	Offset				Rn			Rd	
Store	STRH	<rd></rd>	<rn></rn>		bit_of	fset>			Offset	*2] =	Rd[15					
	0	1	1	1	0		(	Offset				Rn			Rd	
Add	ADD	<rd></rd>	<rn></rn>	#~Q	bit im	ım>	Add Rd	lmm v	alue to	o Rd a	ınd sto	ore th	e valu	e in	N, Z, V	C,
Auu	0	0	1	0	0	1111/	Rd					lm	m		v	
	<u> </u>		-	0			- NO						•••			
Sub	SBC	<rm></rm>	<rn></rn>				Rd =	Rd – F	Rm – C					N, Z, V	C,	
	0	1	0	0	0	0	0	1	0	1		Rm			Rd	
Jump	В	<target< td=""><td>address</td><td>&gt;</td><td></td><td>PC = I</td><td>PC + (si</td><td>gnExt(</td><td>offset)</td><td>&lt;&lt;1)</td><td></td><td></td><td></td><td></td><td></td><td></td></target<>	address	>		PC = I	PC + (si	gnExt(	offset)	<<1)						
	1	1	1	0	0					(	Offset					
Shift	ASR	<rm></rm>	<rn></rn>				Rd =	Rd (A	rithme	tic)>>	· Rm			N, Z,	С	
	0	1	0	0	0	0	1	0	1	0		Rm			Rd	
BIC	BIC	<rm></rm>	<rn></rn>				Rd =	Rd AN	ID NO	ΓRm				N, Z		
5.0	0	1	0	0	0	0	1	1	1	1		Rm		_	Rd	
В	В	LE	#<8_b		set>	if cor	nd the		- PC + (		xt(off:	set)<<		Cond	d. Is N	flag
	1	1	0	1	0	0	1	0				Off	set			
			Exc	eptior	ns(Aritl	hmetic	: Over	flow a	nd Un	define	ed Ins	tructio	on)			

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

#### Instructions

mstrut	200113															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Load	LDRB	<rd></rd>	<rn></rn>	<rm< td=""><td>&gt;</td><td></td><td>Rd = ze</td><td>eroExt(N</td><td>1em[Rm</td><td>+ Rn][8</td><td>:0])</td><td></td><td></td><td></td><td></td><td></td></rm<>	>		Rd = ze	eroExt(N	1em[Rm	+ Rn][8	:0])					
	0	1	0	1	0	1	1		Rm			Rn			Rd	
Store	STRB	<rd></rd>	<rn></rn>	<rm< td=""><td>&gt;</td><td></td><td>Mem[l</td><td>Rm+Rn]=</td><td>Rd[8:0]</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></rm<>	>		Mem[l	Rm+Rn]=	Rd[8:0]							
	0	1	0	1	1	1	1		Rm			Rn			Rd	
Add	Add	<rd></rd>	<rn></rn>	#<3_	_bit_ir	nm>	Add R	n and I	mm sto	re resu	lt in Rd			N, Z,	C, V	
	0	0	0	1	1	1	0		Imm			Rn			Rd	
Sub	SBC	<rm></rm>	<rn></rn>			ı	Rd = I	Rd - Rm	1 - C					N, Z,	C, V	
	0	1	0	0	0	0	0	1	0	1	1	Rm			Rd	
Jump	В	<target< td=""><td>address&gt;</td><td></td><td></td><td>PC =</td><td>PC + (sig</td><td>gnExt(of</td><td>fset)&lt;&lt;1</td><td><u> </u></td><td></td><td></td><td></td><td></td><td></td><td></td></target<>	address>			PC =	PC + (sig	gnExt(of	fset)<<1	<u> </u>						
	1	1	1	0	0						Offset					
LSR	LSR	<rm></rm>	<rn></rn>			1	Rd = R	d >> Rm	-					N, Z, C		
	0	1	0	0	0	0	1	0	0	1		Rm			Rd	
NEG	NEG	<rm></rm>	<rn></rn>			Rd = - Rm N, Z, C, V										
	0	1	0	0	0	0	0	1	1	0		Rm			Rd	
			_						PC = PC	+						
В	В	CC	#<8_bi	_		ı	(signE	xt(offse				Che	cks if	no carr	У	
	1	1	0	1	0	1	1	1	Offset							

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

#### Instructions

msuuc	HOHS																	
	15	14	13	12	11	10	9	8	7	6	5	4	3		2		1	0
Load	LDRB	<rd></rd>	<rn></rn>	#<5_	bit_of	fset>	Rd =	= zero	Exte	nd(Me	em[R	n + C	Offset	])				
	0	1	1	0	1		C	ffset				Rn				Rd		
Store	STRB	<rd></rd>	<rn></rn>	#<5_	bit_of	fset>	Mer	n[Rn -	+ Off	set] =	Rd[	8:0]						
	0	1	1	0	0		C	ffset				Rn				Rd		
Add	Add	<rd></rd>	<rn></rn>	#<3_	bit_im	m>	Add	Rn a	nd Im	ım sto	re re	sult i	n Rd		N,	, Z, C	, V	
	0	0	0	1	1	1	0	Imm	ı			Rn				Rd		
Sub	SUB	<rd></rd>	<rn></rn>	#<3_	bit_im	m>	Sub	tract I	mm f	rom I	Rn sto	ore re	sult i	n Rd	N,	, Z, C	, V	
	0	0	0	1	1	1	1	Imn	ı			Rn				Rd		
Jump	В	<target< td=""><td>address&gt;</td><td>&gt;</td><td></td><td>PC =</td><td>PC +</td><td>(sign</td><td>Ext(c</td><td>offset)</td><td>&lt;&lt;1)</td><td>)</td><td></td><td></td><td></td><td></td><td></td><td></td></target<>	address>	>		PC =	PC +	(sign	Ext(c	offset)	<<1)	)						
	1	1	1	0	0								Offs	et				
AND	AND	<rm></rm>	<rn></rn>			1	Rd =	= Rd <i>A</i>	AND	Rm						N, 2	Z	
	0	1	0	0	0	0	1	1	0	0		Rm				Rd		
_	_							ond th		_	_	-						
В	В	EQ	#<8_b	it_off:				nExt(	•		1)			Check for Z	Zero f	lag		
	1	1	0	1	0	1	0		Offs	set								
CMP	CMP	<rd></rd>	<rn></rn>	#<8_	bit_im	m>		npare			alue t	o Rd		N, Z, C, V				
	0	0	1	1	1		Rd		Imn	1								

Cache Size			N	Iemory	y Spe	cific	ation	ıs									
Cache Line Size	-	Cache															
Write Policy	_			ize	8B												
Replacement policy	-	Associ	iativity		2												
Cache Type	-	Write	Policy		W	rite T	`hrou	gh									
Instructions	-	Repla	cement	policy	FI	FO											
15		Cache	Type		W	ау На	alting	5									
15	Instru	ıctions															
O			14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Stor   e	Load								Mem[R			Used fo		ers		D.1	
STR		0	1	0	1	0	0	0		Rm			Rn			Rd	
Add		STR	<rd></rd>	<rn></rn>	<rm< td=""><td>&gt;</td><td></td><td></td><td>[Rm+Rr</td><td>n]=R</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></rm<>	>			[Rm+Rr	n]=R							
O							0			Rm			Rn			Rd	
Substract   Substract   Substract   Imm   Substract   Imm   Value   Imm   Im	Add	Add	<rd></rd>	<rn></rn>	<rm< td=""><td>&gt;</td><td></td><td>Add F</td><td>Rn and I</td><td>Rm store</td><td>e resu</td><td>lt in Rd</td><td></td><td></td><td></td><td>N, Z, C, V</td><td></td></rm<>	>		Add F	Rn and I	Rm store	e resu	lt in Rd				N, Z, C, V	
SUB           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> &gt; &gt; &gt;</rd>		0	0	0	1	1	0	0		Rm			Rn			Rd	
O	Suh	CLID	∠Pd>	∠Pn>		_bitir	nm	Subtr	act Imn	a valuo f	rom	Pd and a	toro the	valuo i	n Dd	NZCV	
Shift         ASR         < Rd> >	Jub					1			act iiiii	i value i	10111	inu anu s	itore trie		ii Nu	14, 2, 0, 4	
O   O   O   I   O   Imm   Rm   Rd	Shift	ASR	<rd></rd>			_bit_ir	nm	Arithi	metic sl	nift Rm r	ight l	ov Imm a	and stor	e in Rd		N. Z. C	
B						0		-				7					
1       1       1       1       0       0       0       Offset         NEG       S       S       RN S       RN S       RN S       RN S       N, Z, C, V         0       1       0       0       0       0       1       0       RM       RM       Rd         B       VS       #<8_bit_offset>       (signExt(offset)<<1)								/		<b>6</b> 5							
NEG G > <rn></rn>	ס					0	PC =	PC + (9	signExt(	offset)<	<1)	(	Offset				
NEG       G       > <rn>       Rd = - Rm       N, Z, C, V         0       1       0       0       0       1       1       0       Rm       Rd         if cond then PC = PC +         B       VS       #&lt;8_bit_offset&gt;       (signExt(offset)&lt;&lt;1)</rn>			_		O	U						•	511500				
0         1         0         0         0         0         1         1         0         Rm         Rd           if cond then PC = PC +           B         B         VS         #<8_bit_offset>         (signExt(offset)<<1)	NEG			∠Pn>				Pd -	. Pm						N 7 C	V	
B	1120		1		0	0	0			1	0		Rm		14, 2, 0,		
1 1 0 1 1 0 0 1 Offset																	
	В						_			set)<<1	)		Ch			w Flag	
Exceptions(Arithmetic Overflow and Undefined Instruction)		1	1	U	1	1	0	0	1					Offset			
					E	xcept	tions(	Arithn	netic O	verflow	<i>i</i> and	Undefi	ned Ins	tructio	n)		

	1.	AT .										
		Iemory			tions	5						
Cache			1K	В								
Cache	Line S	ize	8B									
Associ	iativity		8									
Write .	Policy		Wr	ite T	hroug	;h						
Replac	cement	policy	LR	U Co	unter	•						
Cache	Type		Co	nvent	tional							
			-1									
ıctions	T											,
15	14	13	12	11	10	9	8	7 6	5	4 3	2 1	0
	<rd &gt;</rd 	<rn &gt;</rn 	#<5_ >	_bit_o	ffset							
1	0	0	0	1						Rn		Rd
CT.	.D!	٠.	ш -	L.:	ر م د م	N 4 -	~ [D ·	. 0#-				
81 R	<ra &gt;</ra 	<kn &gt;</kn 	#<5_ >	_DIT_O	iiset			+ Offse	[			
1	0	0	0	0		Of	ffset			Rn		Rd
AD	<rd< td=""><td><rn< td=""><td></td><td></td><td></td><td>Add</td><td>l Imm</td><td>value to</td><td>o Rd</td><td>and sto</td><td>re</td><td></td></rn<></td></rd<>	<rn< td=""><td></td><td></td><td></td><td>Add</td><td>l Imm</td><td>value to</td><td>o Rd</td><td>and sto</td><td>re</td><td></td></rn<>				Add	l Imm	value to	o Rd	and sto	re	
D	>	>				the						, Z, C, V
0	0	1	0	0		Ka						Imm
SU B	<rd< td=""><td><rn< td=""><td>∠Rm</td><td>1&gt;</td><td></td><td></td><td></td><td></td><td></td><td>Rm from</td><td></td><td>, Z, C, V</td></rn<></td></rd<>	<rn< td=""><td>∠Rm</td><td>1&gt;</td><td></td><td></td><td></td><td></td><td></td><td>Rm from</td><td></td><td>, Z, C, V</td></rn<>	∠Rm	1>						Rm from		, Z, C, V
0	0	0	1	1	0	1			110	Rn		Rd Rd
						D4 -	. Dal a				N 7	
LSL	<rm></rm>	<rn></rn>				Rm –	· Ku <	`			Ν, Ζ, C	
0	1	0	0	0	0	1	0	0 0		Rm		Rd
					DC -	DC +						
В	<target< td=""><td>address</td><td>&gt;</td><td></td><td></td><td></td><td>ffset)&lt;</td><td>&lt;1)</td><td></td><td></td><td></td><td></td></target<>	address	>				ffset)<	<1)				
1	1	1	0	0							Offset	
СМ	<rm< td=""><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>N, Z,</td><td>will set z=1 if equal else n=1 if</td></rm<>										N, Z,	will set z=1 if equal else n=1 if
	> 1		0	Λ	0			0 1		Rm	C, V	Rd < Rm  or  n = 0  if  Rm < Rd
U	1	U	U	U	U	U	0 [	0   1		IXIII		Nu
_										+		
					0				<1)		condit	tion = is Negative flag
		O			- O	0		011300				
			Ex	cepti	ons(A	rithm	netic (	Overflo	w an	d Undef	ined Ins	truction)
	Associations  Replace Cache  Inctions  15  LD  R  1  ST  R  1  AD  D  O  SU  B  O  LSL  O  B  1	Associativity Write Policy Replacement Cache Type  Ictions  15	Write Policy           Replacement policy           Cache Type           actions           15         14         13           LD         Rd         Rn           R         >         >           1         0         0           ST         Rd         Rn           R         >         >           1         0         0    AD	Associativity	Associativity	Associativity         8           Write Policy         Write Through           Replacement policy         LRU Counter           Cache Type         Conventional           actions         15         14         13         12         11         10           LD         RR         R         > >         > bit_offset           R         > >         > >         Image: section of the policy	Associativity	Associativity	Sociativity   S   Write Policy   Write Through   Replacement policy   LRU Counter   Cache Type   Conventional   Conventional	Associativity	Associativity	Associativity

		Mei	norv	Specifi	catior	ıs										
_	Cache S			1KB												
_		Line Size		8B												
_	Associa			Fully												
	Write Po			Write	Throu	σh										
-	Replace		liev	LRU												
-	Cache T		iicy	Conve												
	Cuche 1	уре		Conve	11110116											
Instru	uctions															
	15	14	1	3 12	11	10	9	8	7	6	5	4	3	2	1	(
Load	LDR H	<rd></rd>	<rn:< td=""><td></td><td>_bit_of</td><td>fset</td><td>Rd = z</td><td>zeroExte</td><td>nd(Mem </td><td>Rn -</td><td>+ Offset</td><td>*21)</td><td></td><td></td><td></td><td></td></rn:<>		_bit_of	fset	Rd = z	zeroExte	nd(Mem	Rn -	+ Offset	*21)				
	0	1	1	1	1			Offset				Rn			Rd	
	STR			#~5	_bit_of	fset										
Store	<u> </u>	<rd></rd>	<rn:< td=""><td>&gt; &gt;</td><td></td><td>1001</td><td>Mem[l</td><td></td><td>set *2] = F</td><td>Rd[1</td><td>5:0]</td><td></td><td></td><td></td><td></td><td></td></rn:<>	> >		1001	Mem[l		set *2] = F	Rd[1	5:0]					
	0	1	1	1	0			Offset				Rn			Rd	
															N, Z	С,
Add	Add 0	<rd></rd>	<rn:< td=""><td></td><td></td><td></td><td></td><td>n and R</td><td>m store r</td><td>esul</td><td>t in Rd</td><td>Dia</td><td></td><td></td><td>V Rd</td><td></td></rn:<>					n and R	m store r	esul	t in Rd	Dia			V Rd	
	U	U	0	1	1	0	0		Rm			Rn			Ku	
Sub	SUB	<rd></rd>	<rn:< th=""><th>. #_0</th><th>_bit_in</th><th>nm&gt;</th><th>Subtro</th><th>ot Imm</th><th>value froi</th><th>m D</th><th>d and et</th><th>oro the</th><th>, valuo</th><th>in Dd</th><th>N, Z, V</th><th>С,</th></rn:<>	. #_0	_bit_in	nm>	Subtro	ot Imm	value froi	m D	d and et	oro the	, valuo	in Dd	N, Z, V	С,
Jub	0	0	1	0	1	11112	Rd		value IIOI	II IX	u anu si	Imn		III Ku	<u> </u>	
Shift	LSR	<rd></rd>	<rm &gt;</rm 		_bit_in	nm>	Shift F	Rm right	by Imm a	and s	store in	Rd			N, Z,	С
	0	0	0	0	1		in	nmediat	e			Rm			Rd	
Jum																
р	В	<target< td=""><td>addre</td><td>ss&gt;</td><td>•</td><td>PC =</td><td>= PC + (</td><td>signExt(</td><td>offset)&lt;&lt;</td><td>1)</td><td></td><td></td><td></td><td></td><td></td><td></td></target<>	addre	ss>	•	PC =	= PC + (	signExt(	offset)<<	1)						
	1	1	1	0	0					(	Offset					
	_															
		<rm< td=""><td></td><td>·</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></rm<>		·												
MVN	MVN	<rm &gt;</rm 	<rn:< td=""><td></td><td></td><td></td><td></td><td>NOT Rm</td><td></td><td></td><td></td><td>D</td><td></td><td>N, Z</td><td>D.I</td><td></td></rn:<>					NOT Rm				D		N, Z	D.I	
MVN		<rm< td=""><td><rn></rn></td><td>&gt; 0</td><td>0</td><td>0</td><td>Rd = 1</td><td>NOT Rm 0</td><td></td><td>0</td><td></td><td>Rm</td><td></td><td>N, Z</td><td>Rd</td><td></td></rm<>	<rn></rn>	> 0	0	0	Rd = 1	NOT Rm 0		0		Rm		N, Z	Rd	
MVN	MVN	<rm &gt;</rm 	0	0		0	0 if cond	0 dition th	0 en PC = F			Rm		N, Z		tive
	MVN 0	<rm &gt; 1</rm 	0 #<8_	0 bit_offs	et>		o if cond (signE	0 dition the xt(offset	0 en PC = F				Flag			tive
MVN B	MVN 0	<rm &gt;</rm 	0	0		0	0 if cond	0 dition th	0 en PC = F			Rm	Flag			tive
	MVN 0	<rm &gt; 1</rm 	0 #<8_	bit_offs	et>	0	o if cond (signE 0	0 dition the xt(offset 0	0 en PC = F	PC +	ed Instru	Offs	Flag			tive

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

Instruc	ctions															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Load	LDRB	<rd></rd>	<rn></rn>	#<5_	_bit_of	fset>	Rd = z	eroExte	nd(Men	n[Rn +	Offset])					
	0	1	1	0	1			Offset				Rn			Rd	
Store	STRB	<rd></rd>	<rn></rn>	#<5_	_bit_of	fset>	Mem[F	Rn + Off	set] = R	Rd[8:0]						
	0	1	1	0	0			Offset				Rn			Rd	
ا ما ما		5.	_											N, Z	, C,	
Add	Add	<rd></rd>	<rn></rn>	#<3_	_bitim	ım>	Add R	n and In	nm stor	e result	ın Rd			<u> </u>		
	0	0	0	1	1	1	0		lmm			Rn			Rd	

Sub	SBC	<rm></rm>	<rn></rn>			N, Z, C, V						
	0	1	0	0	0	0	0	1	0	1	Rm	Rd

Shift	ASR	<rm></rm>	<rn></rn>	Rd = Rd (Arithmetic) >> Rm N,								
	0	1	0	0	0	0	1	0	1	0	Rm	Rd

Jump	В	<target< th=""><th>address</th><th>S&gt;</th><th></th><th>PC = PC + (signExt(offset) &lt;&lt; 1)</th></target<>	address	S>		PC = PC + (signExt(offset) << 1)
	1	1	1	0	0	Offset

MOV	MOV	<rd></rd>	<rn></rn>	#<8_	_bit_im	m> Move Imm va	ue to Rd	N, Z,
	0	0	1	1	0	Rd	Imm	

В	В	CC	#<8_bi	t_offs	et>			lition th	en PC = PC + t)<<1)	condition is Carry Flag
	1	1	0	1	0	1	1	1		Offset

		Me	emory	Speci	ficat	ions											
-	Cache	Size		512E	3												
-	Cache	Line Siz	ie.	16B													
•	Associa	ativity		2													
-	Write I	Policy		Writ	e Bac	ck											
•	Replac	ement p	olicy	FIFO	)												
-	Cache	Туре		Way	Halt	ing											
Instru	uctions	1.4	12	12	11	10	0	0	7	_	_	4	2	2	1		0
	15	14	13 <rn< td=""><td></td><td>11</td><td>10</td><td>9 Rd =</td><td>8 = Mem</td><td>7 [Rm +</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td></td><td>0</td></rn<>		11	10	9 Rd =	8 = Mem	7 [Rm +	6	5	4	3	2	1		0
Load	LDR 0	<rd></rd>	> 0	<rm< td=""><td>)&gt; 0</td><td>0</td><td>Rn] 0</td><td></td><td>Rm</td><td></td><td>Use</td><td>d for p Rn</td><td>ointer</td><td>s</td><td></td><td>Rd</td><td></td></rm<>	)> 0	0	Rn] 0		Rm		Use	d for p Rn	ointer	s		Rd	
	U		O		0	0	U		IXIII			IXII				Nu	
Stor	OTD	D-I	<rn< td=""><td>D</td><td>_</td><td></td><td>N 4 =</td><td>. [D</td><td>D.,1 F</td><td>۱ ـ ۱</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></rn<>	D	_		N 4 =	. [D	D.,1 F	۱ ـ ۱							
е	STR 0	<rd></rd>	> 0	<rm< td=""><td>1&gt;</td><td>0</td><td>Men 1</td><td>ոլκՠ+</td><td>Rn]=R</td><td>a</td><td></td><td>Rn</td><td></td><td></td><td></td><td>Rd</td><td></td></rm<>	1>	0	Men 1	ոլκՠ+	Rn]=R	a		Rn				Rd	
		_		<u> </u>													
Add	AD C	<rm &gt;</rm 	<rn></rn>				Rd =	= Rd +	Rm +	С				N, Z,	C, V		
	0	1	0	0	0	0	0	1	0	0		Rm				Rd	
			<rn< td=""><td>#&lt;3</td><td>_bit_ir</td><td>mm</td><td>Subt</td><td>tract Ir</td><td>nm fro</td><td>m R</td><td>n stoi</td><td>e resu</td><td>lt in</td><td>N, Z,</td><td>C.</td><td></td><td></td></rn<>	#<3	_bit_ir	mm	Subt	tract Ir	nm fro	m R	n stoi	e resu	lt in	N, Z,	C.		
Sub	SUB	<rd></rd>	>	>		1	Rd							V , _,			
	0	0	0	1	1	1	1		lmm			Rn				Rd	
C1 : C1							Rd =										
Shift	ROR 0	<rm></rm>	<rn></rn>	0	0	n	(Circ	ular)>> 0	·Rm 1	1		Rm		N, Z, C	<u> </u>	Rd	
	O		O	U	U	U	1	U	1			IXIII				Nu	
Jum	Б		1 . 1				= PC -		4)								
p	B 1	<target< td=""><td>addres</td><td>0</td><td>0</td><td>(sigr</td><td>nExt(o</td><td>ffset)&lt;</td><td>&lt;&lt;1)</td><td></td><td></td><td>Offs</td><td>et</td><td></td><td></td><td></td><td></td></target<>	addres	0	0	(sigr	nExt(o	ffset)<	<<1)			Offs	et				
		_															
ORR	ORR	<rm></rm>	<rn></rn>				Rd =	Rd OR	Rm					N, Z			
	0	1	0	0	0	0	1	1	1	0		Rm				Rd	
							if co	nd the	n PC =	- DC	_						
В	В	MI	#<8_b	oit_offs	set>				ffset)<		т		cond	ition is I	Vegativ	e flag	
	1	1	0	1	1	0	0	0					C	Offset			
				Fyc	entio	ns/Ari	thme	tic Ov	erflow	and	Unde	fined	Instri	uction)			
				LXC	срио	113(7.111	tillic	tic OV	2111044	una	Onac	inica	111361	actioni			

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

#### Instructions

instruc	LUOUS															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Load	LDRB	<rd></rd>	<rn></rn>	<rm< td=""><td>&gt;</td><td></td><td>Rd = z</td><td>eroExt</td><td>(Mem[</td><td>Rm + F</td><td>Rn][8:0]</td><td>)</td><td></td><td></td><td></td><td></td></rm<>	>		Rd = z	eroExt	(Mem[	Rm + F	Rn][8:0]	)				
	0	1	0	1	0	1	1		Rm			Rn			Rd	
Store	STRB	<rd></rd>	<rn></rn>	<rm< td=""><td colspan="8">m&gt; Mem[Rm+Rn]=Rd[8:0]</td><td></td><td></td><td></td></rm<>	m> Mem[Rm+Rn]=Rd[8:0]											
	0	1	0	1	1	1	1	[IXIII - IXI	Rm	3.0]		Rn			Rd	
l			U						14111			1111			T(d	
Add	ADC	<rm></rm>	<rn></rn>				Rd = Rd + Rm + C   N,							N, Z, (	C, V	
	0	1	0	0	0	0	0	1	0	0		Rm			Rd	
Sub	SUB	<rd></rd>	<rn></rn>	#<8_	_bit_ir	nm>	Subtr Rd	act Imn	n value	e from	Rd and	store	the val	ue in	N, Z, V	С,
	0	0	1	0	1		Rd					Imn	n			
Shift	ASR	<rd></rd>	<rm></rm>	#<5_	_bit_ir	nm>	Arithr	netic sl	hift Rm	ı right l	oy Imm	and st	tore in	Rd	N, Z,	<u>—</u>
	0	0	0	1	0		in	nmedia	te			Rm			Rd	
Jump	BL	<target< td=""><td>address</td><td>5&gt;</td><td></td><td>PC =</td><td>PC + (s</td><td>signExt(</td><td>offset)</td><td>)&lt;&lt;1)</td><td></td><td></td><td></td><td></td><td></td><td></td></target<>	address	5>		PC =	PC + (s	signExt(	offset)	)<<1)						
	1	1	1	1	0					(	Offset					
CMN	CMN	<rm></rm>	<rn></rn>				Rd + I	Rm (alu	out = F	Rd + Rn	n)			N, Z, (	C, V	
	0	1	0	0	0	0	0	0	1	0		Rm			Rd	
В	В	VC	#<8_bi	t_offs	et>		if con	d then	PC = P	C + (sig	nExt(of	ffset)<	<1)	cond.	is V fla	ag

Exceptions(Arithmetic Overflow and Undefined Instruction)

Offset

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

Instruc	ctions															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				#<5_	bit_of	it_offset										
Load	LDRB	<rd></rd>	<rn></rn>	>		Rd = zeroExtend(Mem[Rn + Offset])										
	0	1	1	0	1 Offset Rn Rd											
				#<5	_bit_offset											
Store	STRB	<rd></rd>	<rn></rn>	>		Mem[Rn + Offset] = Rd[8:0]										
	0	1	1	0	0			Offset	_			Rn			Rd	
Add	ADD	<rd></rd>	<rn></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></rd>	<rn></rn>	#<8_	8_bit_imm> Subtract Imm value from Rd and store the value in Rd N, Z, C, V											
	0	0	1	0	0 1 Rd Imm											
															_	
Jump	BL	<target< td=""><td>addres</td><td>s&gt;</td><td></td><td>PC =</td><td>PC + (się</td><td>gnExt(o</td><td>ffset)&lt;&lt;</td><td>1)</td><td></td><td></td><td></td><td></td><td></td><td></td></target<>	addres	s>		PC =	PC + (się	gnExt(o	ffset)<<	1)						
	1	1	1	1	0						Offs	et				
ROR	ROR	<rm></rm>	<rn></rn>				Rd = R	d (Circu	lar)>>Rr	n				N, Z, C	2	
	0	1	0	0	0	0	1	0	1	1		Rm			Rd	
AND	AND	<rm></rm>	<rn></rn>				Rd = R	d AND I	Rm					N, Z		
	0	1	0	0	0	0	1	1	0	0		Rm			Rd	
									PC = PC							
В	В	MI	#<8_b	<8_bit_offset> (signExt(offset)<<1) condition is Negative flag												
	1	1	0	1	1	0	0	0				C	Offset			
	Exceptions(Arithmetic Overflow and Undefined Instruction)															

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

Instruc	ctions																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Load	LDRH	<rd></rd>	<rn></rn>	<rm< td=""><td>&gt;</td><td></td><td>Rd =ze</td><td>roExt(M</td><td>em[Rm</td><td>+ Rn][16</td><td>5:0])</td><td></td><td></td><td></td><td></td><td></td></rm<>	>		Rd =ze	roExt(M	em[Rm	+ Rn][16	5:0])						
	0	1	0	1	0	0	1		Rm		Rn			Rd			
Store	STRH	<rd></rd>	<rn></rn>	<rm< td=""><td>&gt;</td><td></td><td>Mem[l</td><td>Rm+Rn]=</td><td>-Rd[16:0</td><td>)]</td><td></td><td></td><td></td><td></td><td></td><td></td></rm<>	>		Mem[l	Rm+Rn]=	-Rd[16:0	)]							
	0	1	0	1	1	1	0		Rm		Rn				Rd		
Add	Add	<rd></rd>	<rn></rn>	#<3_	_bit_ir	nm>	Add Rr	n and Im	m store	result ir	n Rd			N, Z, 0	C, V		
	0	0	0	1	1	1	0		lmm			Rn			Rd		
Sub	SUB	<rd></rd>	<rn></rn>	#<3_	_bit_ir	nm>	> Subtract Imm from Rn store result in Rd							N, Z, 0	C, V		
	0	0	0	1	1	1	1 Imm					Rn			Rd		
Shift	LSL	<rd></rd>	<rm></rm>	#<5_	_bit_ir	nm>	Shift R	m left b	y Imm aı	nd store	in Rd			N, Z, C			
	0	0	0	0	0			immedia	ite		Rm				Rd		
Jump	BL	<target< td=""><td>address</td><td>&gt;</td><td>1</td><td>PC =</td><td>PC + (si</td><td>gnExt(of</td><td>fset)&lt;&lt;1</td><td>.)</td><td></td><td></td><td></td><td></td><td></td><td></td></target<>	address	>	1	PC =	PC + (si	gnExt(of	fset)<<1	.)							
	1	1	1	1	0						Offset						
MUL	MUL	<rm></rm>	<rn></rn>		1	1	Rd = R	d * Rm						N, Z			
	0	1	0	0	0	0	0	1	1	1		Rm			Rd		
								d then I	_	+							
В	В	VS	#<8_b	it_off	set>		(signE	signExt(offset)<<1) condit					tion is c	n is overflow set			

Exceptions(Arithmetic Overflow and Undefined Instruction)

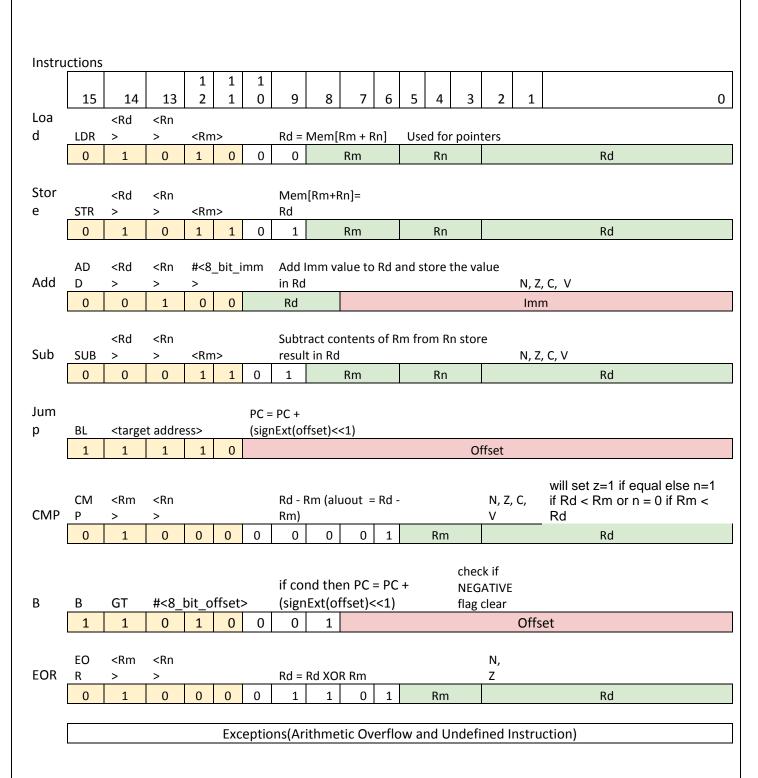
Offset

0

Memo	ory Specifications
Cache Size	512B
Cache Line Size	16B
Associativity	2
Write Policy	Write Through
Replacement polic	y FIFO
Cache Type	Way Halting

L	Cache	Туре		vv a	ly IIa	ning											
Instru	ctions																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1		0
				#<5_	_bit_offset Rd = zeroExtend(Mem[Rn + Offset *2])												
Load	LDRH	<rd></rd>	<rn></rn>	>	I		Rd = ze			[Rn + O	ffset *2]						
	0	1	1	1	1			Offset				Rn				Rd	
					hit offcot												
Store	STRH	<rd></rd>	<rn></rn>	#<5_ >	bit_ot	bit_offset Mem[Rn+ Offset *2] = Rd[15:0]											
31010	0	1	1	1											Rd		
	U	1	1		U			Uliset				IXII				Nu	
					N, Z, C,												
Add	ADC	<rm></rm>	<rn></rn>				Rd = R	d + Rm -	+C					V , _,			
	0	1	0	0	0	0	0	1	0	0		Rm				Rd	
Sub	SUB	<rd></rd>	<rn></rn>	#<3_	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></rm>	<rn></rn>				Rd = R	d << Rm	l					N, Z,	С		
	0	1	0	0	0	0	1	0	0	0		Rm				Rd	
Jump	BL	<target< td=""><td>taddress</td><td>s&gt;</td><td></td><td>PC =</td><td>PC + (się</td><td>gnExt(of</td><td>fset)&lt;&lt;1</td><td>L)</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></target<>	taddress	s>		PC =	PC + (się	gnExt(of	fset)<<1	L)							
	1	1	1	1	0						Offse	t					
ROR	ROR	<rm></rm>	<rn></rn>				Rd = R	d (Circu	lar)>>Rn	n				N, Z,	С		
	0	1	0	0	0	0	1	0	1	1		Rm				Rd	
									PC = PC	+		cond	ition i	s if CA	RRY		
В	В	CS	#<8_b	it_off				xt(offse	et)<<1)			flag s					
	1	1	0	1	0	1	1	0				C	Offset				
				Ex	cepti	ons(Aı	rithmet	ic Over	flow ar	ıd Und	efined I	nstru	ction	)			

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



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

Instruc	ctions														
	15	14	13	12	11	10	0 9 8 7 6 5 4 3 2				1	0			
Load	LDR	<rd></rd>	<rn></rn>	#<5_	bit_of	ffset> Rd = Mem[Rn + Offset *4]					l.				
	1	0	0	0	1	Offset Rn Rd					Rd				
Store	STR	<rd></rd>	<rn></rn>	#<5_	bit_of	fset>	Mem[Rn + Offs	set *4] = R	d						
	1	0	0	0	0		Offset				Rn			Rd	
Add	ADD	<rd></rd>	<rn></rn>	#<8_	bit_im	ım>	Add Imm value	to Rd and	store	the valu	e in Rd			N, Z, C, V	
	0	0	1	0	0		Rd				lmı	n			
						nm> Subtract Imm value from Rd and store the value in Rd N, Z, C, V									
Sub	SUB	<rd></rd>	<rn></rn>		bit_im	ım>		alue from	Rd an	d store t	he valu	e in Rd		N, Z, C, V	
	0	0	1	0	1	1 Rd Imm									
			_												
Shift	LSR	<rd></rd>	<rm></rm>		_	it_imm> Shift Rm right by Imm and store in Rd N, Z, C									
	0	0	0	0	1		immedia	ite			Rm			Rd	
							20 / : 5 : / (1								
Jump	BL		address		0	PC =	PC + (signExt(off	set)<<1)		O.C. 1					
	1	1	1	1	0					Offset					
TST	тст	∠Dm>	∠Dn>				Charle if hit at a	ans (Dm) is	. cot in	שמ			7		
131	TST 0	<rm></rm>	<rn></rn>	0	0	0	Check if bit at p	1	s set in	ка	Rm		Z	Rd	
	U	1	U	U	U	U	0 0	1	T		KIII			Ku	
							if cond then F	PC = PC +			chock	:f 7EDC	) flag		
В	В	EQ	#<8 b	it off	set>	check ii Zello hag									
	1	1	0	1	0										
!															
				Ex	ceptio	ns(Ar	ithmetic Overf	low and l	Jndefi	ined Ins	tructio	n)			
ļ	Exceptions(Arithmetic Overflow and Undefined Instruction)														

		M	lemory	Spec	cifica	tions						
Ī	Cache	Size		512	В							
Ī	Cache	Line Si	z,e	32B	3							
Ī	Associa	ativity		2								
Ī	Write I	Policy		Wri	te Ba	ck						
	Replac	ement p	olicy	FIF	O							
	Cache	Туре		Way	y Pre	dictio	n					
Instru	ıctions											
	15	14	13	12	11	10	9	8 7	6	5 4 3	2 1	0
Load	LDR H	<rd></rd>	<rn></rn>	#<5_ >	_bit_of	fset	Rd = zeroEx	rtend(Mem	[Rn + 0	ffset *21)		
Loud	0	1	1	1	1		Off		<u> </u>	Rn	Rd	
				ш.г	h:4 -4	·						
Store	STRH	<rd></rd>	<rn></rn>	#<5_ >	_bitof	rset	Mem[Rn+ C	Offset *2] =	Rd[15:	0]		
	0	1	1	1	0		Off	set		Rn	Rd	
Add	Add	<rd></rd>	<rn></rn>	<rm:< td=""><td>`</td><td></td><td>Add Rn and</td><td>Rm store</td><td>racult in</td><td>Rd</td><td>N, Z, C, V</td><td></td></rm:<>	`		Add Rn and	Rm store	racult in	Rd	N, Z, C, V	
Auu	0	0	0	1	1	0	0	Rm	esuit iii	Rn	Rd	
						I						
Sub	SUB	<rd></rd>	<rn></rn>	<rm:< td=""><td></td><td></td><td></td><td></td><td>m from</td><td>Rn store result in Ro</td><td></td><td></td></rm:<>					m from	Rn store result in Ro		
	0	0	0	1	1	0	1	Rm		Rn	Rd	
			<rm< td=""><td></td><td>_</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></rm<>		_							
Shift	ASR 0	<rd></rd>	> 0	#<5_ 1	bit_im_0	nm>	Arithmetic imme		tht by Ir	nm and store in Rd Rm	N, Z, C Rd	
	0	0	0		0		IIIIII	ulate		KIII	, nu	
Jum												
p	BL		t address			PC =	PC + (signExt	(offset)<<	.)	Office		
	1	1	1	1	0					Offset		
TCT	TCT	<rm< td=""><td></td><td></td><td></td><td></td><td>Cl 1:(1:)</td><td>. (5</td><td></td><td></td><td>_</td><td></td></rm<>					Cl 1:(1:)	. (5			_	
TST	TST 0	> 1	<rn></rn>	0	0	0	Check if bit	at pos (Rm 0 1	) is set i	In Ka Rm	Z Rd	
					U		<u> </u>	<u> </u>				
<b>D</b>		NIE					if cond the		+	check if NE	GATVIE	
В	B 1	NE 1	#<8_b	1	set>	1	(signExt(of	1 (1)		flag clear Offset		
					U			1		Offset		
BIC	DIC	<rm< td=""><td>4Dins</td><td></td><td></td><td></td><td></td><td>D NOT Des</td><td></td><td></td><td>N,</td><td></td></rm<>	4Dins					D NOT Des			N,	
ыс	BIC 0	> 1	<rn></rn>	0	0	0	Rd = Rd AN	1 1	1	Rm	Z Rd	
					_		I I	l			-	
				Ex	ceptio	ons(Ar	rithmetic Ov	erflow an	d Unde	efined Instruction)		

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

15	
O         1         1         1         1         Offset         Rn         Rd           Store         STRH	0
Store         STRH <rd> <rn>         #&lt;5_bit_offset&gt;         Mem[Rn+ Offset *2] = Rd[15:0]           0         1         1         1         0         Offset         Rn         Rd           Add         ADD         <rd> <rn>         #&lt;8_bit_imm&gt;         Add Imm value to Rd and store the value in Rd         V           0         0         1         0         0         Rd         Imm</rn></rd></rn></rd>	
0         1         1         1         0         Offset         Rn         Rd           Add         ADD <rd> <rn></rn></rd>	
Add ADD <rd> <rn> #&lt;8_bit_imm&gt; Add Imm value to Rd and store the value in Rd V  0 0 1 0 0 Rd Imm</rn></rd>	
Add ADD <rd> <rn> #&lt;8_bit_imm&gt; Add Imm value to Rd and store the value in Rd V  0 0 1 0 0 Rd Imm</rn></rd>	
	Ξ,
Subtract Imm value from Rd and store the value in N, Z, C Sub SUB <rd> <rn> #&lt;8_bit_imm&gt; Rd V</rn></rd>	Ξ,
0 0 1 0 1 Rd Imm	
Shift LSR <rm> <rn> Rd = Rd &gt;&gt; Rm N, Z, C</rn></rm>	
0 1 0 0 0 1 0 0 1 Rm Rd	
Jump BL <target address=""> PC = PC + (signExt(offset)&lt;&lt;1)</target>	
1 1 1 0 Offset	
MVN MVN <rm> <rn> Rd = NOT Rm N, Z</rn></rm>	
0         1         0         0         0         0         0         0         0         Rm         Rd	
Cond. flag is  B B CS #<8_bit_offset> if cond then PC = PC + (signExt(offset)<<1) carry	
1 1 0 1 0 1 0 Offset	
Exceptions(Arithmetic Overflow and Undefined Instruction)	

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

							- 1		•			
ı	n	S	П	rı	ı	1		П	и	n	n	١C

Instruc	ctions															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Load	LDRH	<rd></rd>	<rn></rn>	<rm< td=""><td>&gt;</td><td></td><td>Rd =</td><td>zeroE</td><td>xt(Mei</td><td>n[Rm</td><td>+ Rn]</td><td>[16:0]</td><td>)</td><td></td><td></td><td></td></rm<>	>		Rd =	zeroE	xt(Mei	n[Rm	+ Rn]	[16:0]	)			
	0	1	0	1	0	0	1		Rm			Rn			Rd	
•																
Store	STRH	<rd></rd>	<rn></rn>	<rm< td=""><td></td><td></td><td>Men</td><td>า[Rm+</td><td>-Rn]=R</td><td>d[16:</td><td>0]</td><td></td><td></td><td></td><td></td><td></td></rm<>			Men	า[Rm+	-Rn]=R	d[16:	0]					
	0	1	0	1	1	1	0		Rm			Rn			Rd	
Add	Add	<rd></rd>	<rn></rn>	#<3	bit ir	nm>	Add	Rn an	d Imm	store	e resul	t in Ro	t	N, Z,	C, V	
	0	0	0	1	1	1	0		Imm			Rn			Rd	
Sub	SUB	<rd></rd>	<rn></rn>	#<3_	_bit_ir	nm>	Subt Rd	ract Ir	mm fro	om Rr	store	result	t in	N, Z,	C, V	
	0	0	0	1	1	1	1		Imm			Rn			Rd	
Shift	LSL	<rd></rd>	<rm></rm>	_	_bitir	nm>			eft by I	mm a	nd sto	ore in	Rd		N, Z, C	
	0	0	0	0	0		im	media	ate			Rm			Rd	
Jump	BL	<target< td=""><td>address</td><td>5&gt;</td><td></td><td>PC =</td><td>PC+</td><td>(signE</td><td>xt(offs</td><td>et)&lt;&lt;</td><td>1)</td><td></td><td></td><td></td><td></td><td></td></target<>	address	5>		PC =	PC+	(signE	xt(offs	et)<<	1)					
	1	1	1	1	0						Off	set				
ORR	ORR	<rm></rm>	<rn></rn>				Rd = Rm	Rd OI	R					N, Z		
	0	1	0	0	0	0	1	1	1	0		Rm			Rd	
В	В	EQ	#<8_bi						en PC =	= PC +	(signl	•	-	•	Cond. fla Zero	ag is
	1	1	0	1	0	1	0	1					Offset			

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

Instruc		Γ			I						I					
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Load	LDRB	<rd></rd>	<rn></rn>	#<5_	bit_of	fset>	Rd =	zeroEx	tend(N	1em[R	n + Of	fset])				
	0	1	1	0	1			Offset				Rn			Rd	
Store	STRB	<rd></rd>	<rn></rn>	#<5_	bit_of	fset>	Mem	[Rn + (	Offset]	= Rd[8	3:0]					
	0	1	1	0	0			Offset				Rn			Rd	
															N, Z,	C,
Add	Add	<rd></rd>	<rn></rn>	<rm:< td=""><td>&gt;</td><td></td><td>Add f</td><td>Rn and</td><td>Rm sto</td><td>ore res</td><td>sult in</td><td>Rd</td><td></td><td></td><td>V</td><td></td></rm:<>	>		Add f	Rn and	Rm sto	ore res	sult in	Rd			V	
	0	0	0	1	1	0	0		Rm			Rn			Rd	
							Subtr	act co	ntents	of Rm	from I	Rn stor	e resu	lt in	N, Z,	C,
Sub	SUB	<rd></rd>	<rn></rn>	<rm< td=""><td>&gt;</td><td></td><td>Rd</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>V</td><td></td></rm<>	>		Rd								V	
	0	0	0	1	1	0	1		Rm			Rn			Rd	
Shift	LSR	<rd></rd>	<rm></rm>	#<5_	bit_im	ım>	Shift	Rm rig	ht by Iı	mm ar	nd stor	e in Rd			N, Z,	С
	0	0	0	0	1		im	nmedia	ite			Rm			Rd	
Jump	BL	<target< td=""><td>address</td><td><b>5</b>&gt;</td><td></td><td>PC =</td><td>PC + (s</td><td>signExt</td><td>(offset</td><td>)&lt;&lt;1)</td><td></td><td></td><td></td><td></td><td></td><td></td></target<>	address	<b>5</b> >		PC =	PC + (s	signExt	(offset	)<<1)						
	1	1	1	1	0					(	Offset					
AND	AND	<rm></rm>	<rn></rn>				Rd =	Rd ANI	D Rm					N, Z		
	0	1	0	0	0	0	1	1	0	0		Rm			Rd	
'		1						1								
В	В	NE	#<8 bi	t offs	et>		if con	nd ther	PC = F	PC + (s	ignExt(	offset)	<<1)	Cond	. is Z fl	lag
	1	1	0	1	0	1	0	1				Offs	•			
ļ							1	1								
			E	xcepti	ons(Ar	ithme	tic Ove	erflow	and Ur	ndefine	ed Inst	ructior	n)			

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

<u> </u>																			
Instruc	rtions																		
mstrat	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1		0		
	13	14	13	12	11	10				Mem[		7			1		U		
Load	LDRB	<rd></rd>	<rn></rn>	#<5_	bit_of	fset>	Offse		······································										
		Rn			Rd				Offset			0	1	1	0	1			
'													•				<u> </u>		
Store	STRB	<rd></rd>	<rn></rn>	#<5_	bit_of	fset>	Mem	[Rn +	Offset	t] = Rd	[8:0]								
		Rn			Rd				Offset			0	1	1	0	0			
•																			
Add	Add	<rd></rd>	<rn></rn>	<rm:< td=""><td>&gt;</td><td></td><td>Add I</td><td>Rn and</td><td>d Rm s</td><td>tore r</td><td>esult ii</td><td>n Rd</td><td></td><td></td><td>N, Z,</td><td>C, V</td><td></td></rm:<>	>		Add I	Rn and	d Rm s	tore r	esult ii	n Rd			N, Z,	C, V			
		Rm			Rn			Rd		0	0	0	0	0	1	1			
									ntent	s of Rr	n fron	n Rn st	ore res	sult					
Sub	SUB	<rd></rd>	d> <rn> <rm> in Rd</rm></rn>												N, Z,				
		Rm			Rn			Rd		0	1	0	0	0	1	1			
							D-I	D-I											
Shift	ROR	<rm></rm>	<rn></rn>				Rd =	ка ular)>>	>Rm					N, Z,	r				
Silit	NON	Rm	SIMILE		Rd		0	1	0	1	1	0	1	0	0	0			
					110				0		_		_	Ü					
						PC =	PC +												
Jump	В	<target< td=""><td>addres</td><td>s&gt;</td><td></td><td>(sign</td><td>Ext(off</td><td>set)&lt;&lt;</td><td>(1)</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></target<>	addres	s>		(sign	Ext(off	set)<<	(1)										
					Off	set						1	1	1	0	0			
ORR	ORR	<rm></rm>	<rn></rn>				Rd =	Rd OR	Rm					N, Z					
		Rm			Rd		0	1	1	1	0	0	1	0	0	0			
										= PC	+		condi	ition is					
В	В	MI	#<8_t				(sign	Ext(o			1	1	Nega	tive fla	g	ı			
				Offset	t				0	0	0	1	1	0	1		1		
ı																			
				E>	cepti	ons(A	rithme	etic O	verflo	ow an	d Und	lefine	d Instr	uction	1)				

15										
Associativity										
Write Policy										
Replacement policy										
Replacement policy										
Cache Type										
15										
Rm		8	7	6	5	4	3	2	1	
Rm	1em[Rm +	· Rn][8	R·∩1)							
STRB	Rd	Mije	J.0])	1	1	0	1	0	1	0
STRB										
Add         ADC         > <rn>         Rd         Q           Sub         SUB         <rd><rn>         #&lt;8_bit_imm&gt;         Subtract Imm           Imm         Imm           BL         <tage: col<="" color="" of="" td="" text="" the=""><td>Rd[8:0]</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tage:></rn></rd></rn>	Rd[8:0]									
ADC         > <rn>         Rd         0           cub         SUB         <rd><rn>         #&lt;8_bit_imm&gt;         Subtract Imn           Imm           Umm           BL         <target address="">         PC = PC + (signExt)           Offset           COR         ROR         &gt; <rn>         Rd = Rd (Circ           Rm         Rd         0           <rm< td="">         Rd         0</rm<></rn></target></rn></rd></rn>	Rd			1	1	0	1	0	1	1
SUB	-C							N, Z,	C, V	
Imm   Imm		0	1	0	0	0	1	0	0	0
Imm	alue fron	n Rd a	nd st	ore th	ne va	ılue in	ı Rd		N, Z, C	, V
BL				Rd		0	0	1	0	1
BL										
<pre></pre>	fset)<<1)									
OR         ROR         > <rn>         Rd         Rd         Circ           Rm         Rd         0</rn>						1	1	1	1	0
<rm< td=""><td>ar)&gt;&gt;Rm</td><td></td><td></td><td></td><td></td><td></td><td></td><td>N, Z,</td><td>С</td><td></td></rm<>	ar)>>Rm							N, Z,	С	
		1	0	1	1	0	1	0	0	0
ND = AND > < Rn > Rd = Rd	m							N, Z		
Rm Rd 0		1	1	0	0	0	1	0	0	0
#<8_bit_offset if cond ther B B MI > (signExt(off	DC = DC +						lition i			
Offset			0	0	0	1	1	0	1	
Exceptions(Arithmetic O	et)<<1)				ctru	ction	1)			

		Me	emory	Specifi	cations										
=	Cache	Size		512B											
	Cache	Line Siz	e	32B											
-	Associa	tivity		8											
-	Write F	Policy		Write	Through										
-		ement p	olicy		Counter										
-	Cache				ntional										
าstru	ıctions														
	15	14	13	12	11 10	9	8	7	6	5	4	3	2	1	
.oad	LDR H	<rd></rd>	<rn></rn>	#<5_b >	oit_offset	Rd =	zeroExt	end(Me	m[Rn	+ Offs	et *21)	1			
ouu			Offset			Rn	ECTOEX	criativic	Rd	. 0113	0	1	1	1	1
tor	STRH	<rd></rd>	<rn></rn>	#<5_b >	oit_offset		[Rn+ O	ffset *2]		15:0]					
		C	Offset			Rn			Rd		0	1	1	1	0
dd	ADD	<rd></rd>	<rn></rn>	#<8 b	oit_imm>	Add I	mm va	lue to R	d and	store t	he val	ue in Ro	d	N, Z, C,	V
		Rd		_	_		mm				0	0	1	0	0
Sub	SBC	<rm &gt;</rm 	<rn></rn>			Rd -	Rd - Rm	n - C					N, Z, (	C V	
ub	350	Rm	SIGIE		Rd	0	0	1	0	1	0	1	0	0	0
hift	LSL	<rd></rd>	<rm &gt;</rm 	#<5_b	oit_imm>	Shift	Rm left	: by Imm	and s	tore ir	n Rd			N, Z, C	
		imr	nediate			Rm			Rd		0	0	0	0	
um	BL	<target< td=""><td>addres</td><td>is&gt;</td><td></td><td>PC + (s</td><td>ignExt(</td><td>(offset)&lt;</td><td>&lt;1)</td><td></td><td>4</td><td>1</td><td>1</td><td>1</td><td>0</td></target<>	addres	is>		PC + (s	ignExt(	(offset)<	<1)		4	1	1	1	0
					Offset						1	1	1	1	0
⁄Iul	MUL	<rm &gt;</rm 	<rn></rn>			Rd =	Rd * Rr	n	I				N, Z		
		Rm			Rd	0	0	1	1	1	0	1	0	0	0
}	В	VS	#<8_b	it_offse	t>			PC = PC et)<<1)	+			ition is low set	t		
				Offset				0	0	1	1	1	0	1	
				Evac	ations/ ^ sitl	amatic	Overti	w 224 I	Indof:	nad la	ctruct	ion\			
				Exce	otions(Arith	шеис	overno	w and t	חשפווע	neu in	structi	UII)			

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

Instruc	ctions															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Load	LDRH	<rd></rd>	<rn></rn>	#<5_	bit_of	fset>	Rd = ze	roExter	nd(Mem	[Rn + Of	fset *2])					
		Rn			Rd				Offset			0	1	1	1	1
Store	STRH	<rd></rd>	<rn></rn>	#<5_	bit_of	fset>	Mem[F	Rn+ Offs	et *2] =	Rd[15:0	)]					
		Rn			Rd				Offset			0	1	1	1	0
Add	ADC	<rm></rm>	<rn></rn>				Rd = R	d + Rm +	+C					N, Z	, C, V	
		Rm			Rd		0	0	1	0	0	0	1	0	0	0
Sub	SUB	<rd></rd>	<rn></rn>	#<3_	bit_im	m>	Subtra	ct Imm 1	from Rn	store re	sult in R	d		N, Z	, C, V	
		Rn			Rd			lmm		1	1	0	0	0	1	1
Jump	В	<target< td=""><td>addres</td><td>s&gt;</td><td></td><td>PC =</td><td>PC + (sig</td><td>nExt(of</td><td>fset)&lt;&lt;1</td><td>)</td><td></td><td></td><td></td><td></td><td></td><td></td></target<>	addres	s>		PC =	PC + (sig	nExt(of	fset)<<1	)						
						Offs	et		ŕ			1	1	1	0	0
LSR	LSR	<rm></rm>	<rn></rn>			Offs		d >> Rm				1	1	1 N, Z		0
LSR	LSR	<rm></rm>	<rn></rn>		Rd	Offse		d >> Rm 1		0	1	0	1			0
LSR NEG	LSR		<rn></rn>		Rd	Offso	Rd = R	1		0	1			N, Z 0	, C	
		Rm			Rd	Offse	Rd = Ro	1		0	1			N, Z 0	, C 0	
		Rm <rm></rm>		bit_off	Rd	Offso	Rd = Rd 0 0 Rd = -1 0 if cond	1 Rm 0	0 1 PC = PC	1		0	1  1  cks if	N, Z 0 N, Z 0	, C 0 , C, V	0
NEG	NEG	Rm <rm> Rm</rm>	<rn></rn>	bit_off	Rd	Offso	Rd = Rd 0 0 Rd = -1 0 if cond	1 Rm 0 d then F	0 1 PC = PC	1		0 0 Che	1  1  cks if	N, Z 0 N, Z 0	, C 0 , C, V	0
NEG	NEG	Rm <rm> Rm</rm>	<rn></rn>	oit_off	Rd	Offso	Rd = Rd 0 0 Rd = -1 0 if cond	1 Rm 0 d then F	0 1 PC = PC et)<<1)	+	0	0 Che	1  cks if	N, Z 0 N, Z 0	, C 0 , C, V	0

#### **Question #25 Memory Specifications** Cache Size 1KB Cache Line Size 32B **Associativity** 2 Write Policy Write Back **FIFO** Replacement policy Way Prediction Cache Type Instructions 15 14 13 12 11 10 8 7 6 5 0 Rd = zeroExtend(Mem[Rn + **LDR** <Rn #<5\_bit\_offset Loa <Rd> Offset]) d В Offset Rn Rd 0 0 1 Stor STR <Rn #<5\_bit\_offset Mem[Rn + Offset] = <Rd> Rd[8:0] В Offset Rd 0 Rn 0 0 <Rn Add Rn and Imm store result Add <Rd> #<3 bit imm> in Rd Add Rd 0 Imm Rn 0 1 <Rn Subtract Imm from Rn store N, Z, C, SUB <Rd> Sub #<3 bit imm> result in Rd Imm Rn Rd 1 1 0 1 <Rn Subtract contents of Rm from Rn Shift store result in Rd N, Z, C, V SUB <Rd> <Rm> Rm Rn Rd 1 0 1 PC = PC +Jum (signExt(offset)<<1) p BL<target address> Offset 1 1 1 1 0 will set z=1 if equal else n=1 CM <Rm Rd - Rm (aluout = N, Z, C, if Rd < Rm or n = 0 if Rm <CMP <Rn> Rd -Rm) > ٧ Rd Rd 0 0 Rm 0 0 check if if cond then PC = PC + **NEGATIVE** В GT #<8\_bit\_offset> (signExt(offset)<<1) flag clear Offset 1 1 1 1 0 Exceptions(Arithmetic Overflow and Undefined Instruction)

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

Instruc	tions															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Load	LDR	<rd></rd>	<rn></rn>	#<5_	bit_of	fset>	Rd =	Mem[	Rn + C	offset <sup>†</sup>	*4]					
		Rn			Rd				Offset			1	0	0	0	1
Store	STR	<rd></rd>	<rn></rn>	#<5_	bit_of	fset>	Mem			*4] =	Rd					
		Rn			Rd				Offset			1	0	0	0	0
Add	ADD	<rd></rd>	<rn></rn>	#<8_	bit_im	m>	Add I	lmm v	alue to	o Rd aı	nd sto	re the	value in R	d	N, Z, C,	<b>/</b>
				Imm						Rd		0	0	1	0	0
Sub	SUB	<rd></rd>	<rn></rn>	#<8_	bit_im	m>	Subti	ract In	ım val	ue fro	m Rd	and st	ore the val	ue in Rd	N, Z, C, \	V
				lmm						Rd		0	0	1	0	1
Shift	ASR	<rd></rd>	<rm></rm>	#<5_	bit_im	m>	Arith	metic	shift F	tm righ	nt by I	mm ar	nd store in	Rd	N, Z, C	
		Rm			Rd				Imm			0	0	0	1	0
Jump	В	<target< td=""><td>: address</td><td>&gt;</td><td></td><td>PC = (sign</td><td>PC + Ext(off</td><td>fset)&lt;&lt;</td><td><b>(1)</b></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></target<>	: address	>		PC = (sign	PC + Ext(off	fset)<<	<b>(1)</b>							
					Off	set						1	1	1	0	0
NEG	NEG	<rm></rm>	<rn></rn>				Rd = Rm	-						N, Z, C, \	/	
		Rm			Rd		0	0	1	1	0	0	1	0	0	0
В	В	VS	#<8_bi	t_offse				nd the	n PC = fset)<<	PC +	_	-		r Overflow	Flag	
				Offse	t				0	0	1	1	1	0	1	1
				Exc	eption	ns(Ari	thmet	ic Ov	erflov	<i>i</i> and	Unde	fined	Instructio	n)		

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

	tions 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1		0		
				#<5_	bit_of	fset													
Load	LDR	<rd></rd>	<rn></rn>	>			Rd = I	Mem[R	n + Of		]					ı			
			Offset				Rn			Rd		1	0	0	0	1			
Stor				#<5_	bit_of	fset													
e	STR	<rd></rd>	<rn></rn>	>				[Rn + C	Offset *		<u></u>					_			
			Offset				Rn			Rd		1	0	0	0	0			
Add	AD D	<rd></rd>	<rn></rn>	#<8	bit im	ım>	Add Imm value to Rd and store the value in Rd N, Z, C, V												
		Rd						nm				0	0	1	0		0		
Sub	SUB	<rd></rd>	<rn></rn>	<rm:< td=""><td>&gt;</td><td></td><td colspan="9">Subtract contents of Rm from Rn store result in Rd</td><td>, C, V</td><td></td></rm:<>	>		Subtract contents of Rm from Rn store result in Rd									, C, V			
		Rm			Rn			Rd		0	1	0	0	0	1	1			
Shift	ASR	<rd></rd>	<rm &gt; mediate</rm 		bit_im	ım>	Arithmetic shift Rm right by Imr Rd Rm Rd						store 0	in 0	N, Z	, C	0		
Jum p	BL	<targe< td=""><td>t addres</td><td>s&gt;</td><td></td><td>PC =</td><td>PC + (s</td><td>ignExt(</td><td>(offset)</td><td>&lt;&lt;1)</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></targe<>	t addres	s>		PC =	PC + (s	ignExt(	(offset)	<<1)									
					C	ffset						1	1	1	1	0			
TST	TST	<rm &gt;</rm 	<rn></rn>				Check	c if bit a	at nos l	'Rm) is	set in	Rd		Z					
		Rm	74117		Rd		0	0	0	1	1	0	1	0	0	0			
В	В	NE	#<8_bi	t_offse			if con	d then	PC = P	C +		check if NEGATVIE flag clear							
				Offse	t		1 0 1					1	1	0	1		0		
				Exc	eptior	ns(Arit	hmetic	Overfl	ow and	d Unde	efined I	nstrud	ction)						

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

Instruc	tions															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Load	LDRH	<rd></rd>	<rn></rn>	#<5_k	oit_of	fset>	Rd = ze	eroExte	nd(Mem	ı[Rn + O	])					
		Rn			Rd				Offset			0	1	1	1	1
Store	STRH	<rd></rd>	<rn></rn>	#<5_k	oit_of	fset>	Mem[l	Rn+ Offs	set *2] =	: Rd[15:						
		Rn			Rd				Offset			0	1	1	1	0
Add	ADD	<rd></rd>	<rn></rn>	#<8_k	oit_im	m>	Add In	nm valu	e to Rd a	and stor	ılue in R	ld		N, Z, C, V		
				lmm						Rd		0	0	1	0	0
Sub	SUB	<rd></rd>	<rn></rn>	#<8_k	oit_im	m>	Subtra	ct Imm	value fr	om Rd a	and store	e the va	lue in R	d	N, Z,	C, V
				lmm						Rd		0	0	1	0	1
Shift	LSR	<rd></rd>	<rm></rm>	#<5_k	oit_im	m>	Shift R	m right	by Imm	and sto	ore in Rd			N, Z,	С	
		Rm			Rd			in	nmedia	te	0	0	0	0	1	
Jump	В	<target< td=""><td>address</td><td>&gt;</td><td></td><td>PC =</td><td>PC + (si</td><td>gnExt(of</td><td>fset)&lt;&lt;1</td><td>L)</td><td></td><td></td><td></td><td></td><td></td><td></td></target<>	address	>		PC =	PC + (si	gnExt(of	fset)<<1	L)						
						Offse	t					1	1	1	0	0
MVN	MVN	<rm></rm>	<rn></rn>				Rd = N	OT Rm						N, Z		
		Rm			Rd		0	0	0	0	0	0	1	0	0	0
В	В	MI	#<8_bi	it_offs	et>			dition t xt(offse	hen PC et)<<1)	= PC +	condition is Negative Flag					
				Offse	t				0	0	0	1	1	0	1	1
!																
				Excep	tions	(Arith	nmetic (	Overflo	w and I	Undefii	ned Inst	tructio	า)			

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

Instruc	tions																		
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
Load	LDRB	<rd></rd>	<rn></rn>	#<5_l	oit_of	fset>	Rd =	zerol	Exten	d(Mei	n[Rn	+ Offs	set])						
		C	Offset				Rn			Rd		0	1	1	0	1			
Store	STRB	<rd></rd>	<rn></rn>	#<5_l	oit_of	fset>	Men	n[Rn -	- Offs										
		C	Offset				Rn			Rd		0	1	1	0	0			
							Add Rn and Imm store result in Rd N, Z, C, V												
Add	Add	<rd></rd>	<rn></rn>	#<3_l		ım>	Add		id Imr	n stoi				N, Z,	C, V				
		Imm			Rn			Rd		1	0	0	0	0	1	1			
		_	_							_									
Sub	SBC	<rm></rm>	<rn></rn>		D -I				Rm - (			4		C, V	0				
		Rm			Rd		0	0	1	0	1	0	1	0	0	0			
Shift	LSL	<rd></rd>	<rm></rm>	#<5 l	hit im	ım>	Shift Rm left by Imm and store in Rd N, Z, C												
SHILL	LJL		nediate	#<2_1	JIL_III	1111/	Rm	KIII I	ercby	Rd	anu s	0	0	0	N, Z, C 0	0			
		11111	Tieulate				IXIII			itu		U	U	U	U	U			
Jump	BL	<target< td=""><td>taddress</td><td>s&gt;</td><td></td><td>PC =</td><td colspan="11">= PC + (signExt(offset)&lt;&lt;1)</td></target<>	taddress	s>		PC =	= PC + (signExt(offset)<<1)												
		8-			Offs			- 0		,		1	1	1	1	0			
Į.																			
							Rd =	Rd O	R					N,					
ORR	ORR	<rm></rm>	<rn></rn>				Rm							Z					
		Rm			Rd		0	1	1	1	0	0	1	0	0	0			
_	_								en PC		+				Cond. fl	ag is			
В	В	EQ	#<8_bi		:t>		(sign	Ext(o	offset)<<1)				l .		Zero				
				Offset					1	0	1	1	1	0	1	0			
j				·		:a!	Lia (0: :	. بدار م	اء ماما	- ام ما -	£: a -l	loot							
			E	ceptio	ns(Ar	ıtnme	LIC UV	ertiov	v and	unae	iined	ınstrı	uction	1)					

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

Instruc	rtions			•					_									
mstrac	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1			0
	13	1-1	13	12	**	10		zeroEx										0
Load	LDRB	<rd></rd>	<rn></rn>	#<5_	bit_off	set>		Offset])										
		Rn			Rd			(	Offset			0	1	1	0		1	
Store	STRB	<rd></rd>	<rn></rn>	#<5	bit off	set>	Mem[Rn + Offset] = Rd[8:0]											
		Rn		_	Rd				Offset	_		0	1	1	0		0	
Add	Add	<rd></rd>	<rn></rn>	<rm></rm>			Add F	Rn and			esult ir				N, Z,	C, V		
		Rm			Rn			Rd		0	0	0	0	0	1		1	
Sub	Subtract contents of Rm from Rn store result  SUB <rd> <rn> <rm> in Rd</rm></rn></rd>																	
		Rm			Rn			Rd		0	1	0	0	0	1		1	
Shift	ROR	<rm></rm>	<rn></rn>				Rd =   (Circu	Rd ılar)>>	∙Rm					N, Z,	С			
		Rm			Rd		0	1	0	1	1	0	1	0	0		0	
Jump	В	<target< td=""><td>: addres</td><td>s&gt;</td><td>Off</td><td></td><td>PC + Ext(off</td><td>set)&lt;&lt;</td><td>1)</td><td></td><td></td><td>1</td><td>1</td><td>1</td><td>0</td><td></td><td>0</td><td></td></target<>	: addres	s>	Off		PC + Ext(off	set)<<	1)			1	1	1	0		0	
l												_	_	_				
ORR	ORR	<rm></rm>	<rn></rn>				Rd =	Rd OR	Rm					N, Z				
		Rm			Rd		0	1	1	1	0	0	1	0	0		0	
В	В	MI			<pre>if cond then PC = PC + condition is t_offset&gt; (signExt(offset)&lt;&lt;1)</pre>												1	
				Offset					0	0	0	1	1	0	1			1
				Ex	ception	ons(A	rithme	etic O	verflo	w an	d Und	lefine	d Instr	uction	1)			