OSOROM Instruction Set Reference

The Moroso Project

June 12, 2014

Contents

1	CPU	J Architecture	3
	1.1	Instructions	3
	1.2	VLIW	3
	1.3	Registers	3
	1.4	Shifting	3
	1.5	Virtual Memory	4
	1.6	Exceptions	4
2	Inst	ruction Listing	5
	2.1	ADD - Addition	6
	2.2	AND - Bitwise And	7
	2.3	B - Branch	8
	2.4	BL - Branch with Link	9
	2.5	BREAK - TODO XXX	10
	2.6	CMPBC - Compare Bits Clear	11
	2.7		12
	2.8	CMPEQ - Compare Equal	13
	2.9	CMPLES - Compare Less Than or Equal (Signed)	14
	2.10		15
	2.11	CMPLTS - Compare Less Than (Signed)	16
	2.12	CMPLTU - Compare Less Than (Unsigned)	17
	2.13	CPOP - TODO XXX	18
	2.14	DIV - Integer Division	19
	2.15	ERET - Return from Exception	20
	2.16	FENCE - TODO XXX	21
			22
	2.18	LH - Load Half-Word	23
	2.19	LL - Load Linked	24
	2.20	LW - Load Word	25
	2.21	MFC - Move From Coprocessor Register	26
			27
	2.23	MOV - Move	28
	2.24	MTC - Move To Coprocessor Register	29
		•	30
		1 0/	31
			32
		· .	33

2.29	OR - Bitwise Logical OR	4
2.30	RSB - Reverse Subtraction $\dots \dots \dots$	5
2.31	SB - Store Byte	6
2.32	SC - Store Conditional	7
	SH - Store Half-Word	
2.34	SUB - Subtraction	9
2.35	SW - Store Word	:0
2.36	SXB - Sign-Extend Byte	:1
2.37	SXH - Sign-Extend Half-Word	2
2.38	SYSCALL - TODO XXX	:3
2.39	XOR - Bitwise Exclusive OR	4

Chapter 1

CPU Architecture

1.1 Instructions

Each instruction is a 32-bit word, stored in little-endian order. The top 2 bits of the instruction select one of four predicate registers, and the third bit from the top optionally inverts it. The instruction is only executed if the resulting predicate is true.

1.2 VLIW

At each time step, the CPU fetches a "packet" of 4 instructions located contiguously in memory and executes them in parallel. These packets must be aligned to their 16-byte size, and branch targets must also be so aligned.

There are three types of instructions: Control, Memory, and ALU instructions. Only one control instruction may be executed in any given packet; it must occupy the first (lowest-address) slot in its packet. Likewise, only the first two slots in a packet may execute memory instructions. ALU instructions may be located in any slot.

ALU instructions in the first three slots may optionally specify a "long immediate" operand. In this case, the 32 bits following that instruction are interpreted as an operand, and a no-op is scheduled in the following slot of that packet.

1.3 Registers

There are 32 32-bit general-purpose registers, R0 through R31. R31 is used as the link register for BL instructions. The program counter is stored separately, and may only be modified through branch instructions and read through BL instructions.

There are three one-bit predicate registers that may be written to by compare instructions and used to conditionally execute any instruction.

1.4 Shifting

The ALU is preceded by a shifter for its second operand

1.5 Virtual Memory

Virtual memory is accomplished through a hardware-filled TLB, with 4KB pages and a 2-level page-table structure that looks like x86's without any of the fancy bits. (more detail to come in this section; in particular, we want to boot with paging on but it's not clear how the world looks then)

1.6 Exceptions

Chapter 2

Instruction Listing

2.1 ADD - Addition

2.1.1 Encoding

	1 () C	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	
ĺ	PR	ED		0				C	ONS	TA:	NT]	ROT	'AT	Е	0	0	0	0			RD					RS			
	PR	ED		1	0	1		SHI	FT	AM.	Γ	SI	HF			RT			0	0	0	0			RD					RS			
ĺ	PR	ED		1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			RD					RS			LI

2.1.2 Syntax

- ADD[PN] Rd, Rs, #Imm
- ADD[PN] Rd, Rs, Rt [SHF #Imm]

2.1.3 Type

ADD is an ALU operation. It may be placed in any slot of a packet. If the long immediate form is used it must not be located in the last slot in a packet, and no instruction may be specified in the following slot.

2.1.4 Behavior

Adds the two operands together into the destination register. If two register operands are specified, the second may optionally be shifted by an immediate value before the addition is performed.

TODO optional overflow exceptions???

2.2 AND - Bitwise And

2.2.1 Encoding

1 0 0	O	'	U	U		U	_	_	U	J	O	'	O	U		U	_	_	0	J	O	'	U	U	 U	_	_	0
PRED	0				CC	ONS	STA	NT				I	ROT	ATE	3	0	0	0	1			RD				RS		
PRED	1	0	1	:	SHI	FT	'AM'	Γ	SI	HF			RT			0	0	0	1			RD				RS		
PRED	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1			RD				RS		

2.2.2 Syntax

- AND[PN] Rd, Rs, #Imm
- AND[PN] Rd, Rs, Rt [SHF #Imm]

2.2.3 Type

AND is an ALU operation. It may be placed in any slot of a packet. If the long immediate form is used it must not be located in the last slot in a packet, and no instruction may be specified in the following slot.

2.2.4 Behavior

Computes the bitwise logical AND of the two operands, storing the result in the destination register. If two register operands are specified, the second may optionally be shifted by an immediate value before the computation is performed.

2.3 B - Branch

2.3.1 Encoding

		•			-	 	 	 	•			_				 	 •	 	_	 	_	
PRED	1	1	0	0									OF	'FS	ΕT							
PRED	1	1	1	0						(1 H H	SE								RS		

2.3.2 Syntax

- B[PN] <label>
- B[PN] (Rs + OFF)

2.3.3 Type

Branch is a Control operation. It may only be placed as the first instruction in a packet.

2.3.4 Behavior

Transfers control to the address specified. The OFFSET field is shifted left by 4 places and sign-extended to 32 bits, then added to either the source register or, if none is specified, the address of the branch instruction. If the specified base register is not aligned to a 16-byte value, the target is rounded down to the nearest aligned value before control is transferred. Other instructions in the same packet as the branch instruction are executed before control is transferred.

2.4 BL - Branch with Link

2.4.1 Encoding

PRED	1 1 0 1	OFFSET	
PRED	1 1 1 1	OFFSET	RS

2.4.2 Syntax

- BL[PN] <label>
- BL[PN] (Rs + OFF)

2.4.3 Type

Branch with Link is a Control operation. It may only be placed as the first instruction in a packet.

2.4.4 Behavior

BL functions identically to the Branch instruction, except that the address of the BL instruction is placed in R31.

2.5 BREAK - TODO XXX

2.5.1 Encoding

3	3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1										
1	L	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
	ΡI	REI	D	1	0	0	0	1	0	0	0	1	Х	Х	Х	Х	Х	Х	Х	Х	Х	Х	Х	Х	Х	Х	Х	Х	Х	Х	Х	Х

2.6 CMPBC - Compare Bits Clear

2.6.1 Encoding

	1 0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	
	PRI	ΞD	0				CO	ONS	TA:	NT				I	ROT	'AT	E	0	1	1	1	1	1	1	P	D			RS			
	PRI	ΞD	1	0	1		SHI	FT	AM.	Γ	SI	HF			RT			0	1	1	1	1	1	1	P	D			RS			
Ī	PRI	ΞD	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	P	D			RS			LIM

2.6.2 Syntax

- CMPBC[PN] Pd, Rs, #Imm
- CMPBC[PN] Pd, Rs, Rt [SHF #Imm]

2.6.3 Type

CMPBC is an ALU operation. It may be placed in any slot of a packet. If the long immediate form is used it must not be located in the last slot in a packet, and no instruction may be specified in the following slot.

2.6.4 Behavior

CMPBC checks each bit of Rs corresponding to a 1 bit in the second operand, writing 1 into the destination predicate if all such bits are 0, and writing 0 otherwise – that is, it computes the unary OR of the bits in (~Rs & OP2) and stores the result in the desired predicate.

2.7 CMPBS - Compare Bits Set

2.7.1 Encoding

1	0	9	8	7	6	5	4	3	2	1	0	9	8	_/	6	5	4	3	2	1	0	9	8		6	5	4	3	2	1	0
4	^	_	_	_	_	_		_	\sim	4	^	_	\sim	_	_	_		_	\sim	4	^	_	\sim	_	_	_		_	_	4	^

PRED	0				CC	ONS	TA	NT				I	ROT	'AT	E	0	1	1	1	1	1	0	PD	RS	
PRED	1	0	1		SHI	FT	AM.	Γ	SI	ΙF			RT			0	1	1	1	1	1	0	PD	RS	
PRED	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	PD	RS	LIM

2.7.2 Syntax

- CMPBS[PN] Pd, Rs, #Imm
- CMPBS[PN] Pd, Rs, Rt [SHF #Imm]

2.7.3 Type

CMPBS is an ALU operation. It may be placed in any slot of a packet. If the long immediate form is used it must not be located in the last slot in a packet, and no instruction may be specified in the following slot.

2.7.4 Behavior

CMPBS checks each bit of Rs corresponding to a 1 bit in the second operand, writing 1 into the destination predicate if any such bit is 1, and writing 0 otherwise – that is, it computes the unary OR of the bits in (Rs & OP2) and stores the result in the desired predicate.

2.8 CMPEQ - Compare Equal

2.8.1 Encoding

1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	
P	RE:	D	0				CC	ONS	TAl	NT]	ROT	AT:	Ε	0	1	1	1	0	1	0	P	D			RS			
P	RE	D	1	0	1		SHI	FT	AMT	Γ	SI	ΙF			RT			0	1	1	1	0	1	0	P	D			RS			
P	RE	D	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	0	P	D			RS			LIM

2.8.2 Syntax

- CMPEQ[PN] Pd, Rs, #Imm
- CMPEQ[PN] Pd, Rs, Rt [SHF #Imm]

2.8.3 Type

CMPEQ is an ALU operation. It may be placed in any slot of a packet. If the long immediate form is used it must not be located in the last slot in a packet, and no instruction may be specified in the following slot.

2.8.4 Behavior

CMPEQ tests whether its two operands are equal, writing 1 into the destination predicate if they are and 0 otherwise.

2.9 CMPLES - Compare Less Than or Equal (Signed)

2.9.1 Encoding

	1 0 9	0	1	O	S	4	3	2	T	U	9	0	1	O	S	4	3	2	T	U	9	0	1	O	S	4	3	2	т	U	
	PRED	0				CC	ONS	TAl	NT				I	ROT	AT	Е	0	1	1	1	1	0	1	P	D			RS			Ì
	PRED	1	0	1		SHI	FT	AM:	Γ	SI	ΗF			RT			0	1	1	1	1	0	1	P	D			RS			Ì
ĺ	PRED	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	1	P	D			RS			LIM

2.9.2 Syntax

- CMPLES[PN] Pd, Rs, #Imm
- CMPLES[PN] Pd, Rs, Rt [SHF #Imm]

2.9.3 Type

CMPLES is an ALU operation. It may be placed in any slot of a packet. If the long immediate form is used it must not be located in the last slot in a packet, and no instruction may be specified in the following slot.

2.9.4 Behavior

CMPLES interprets its operands as 32-bit two's-complement signed integers, and writes 1 into the specified predicate register if the value of Rs is less than or equal to the value of the second operand, and 0 otherwise.

2.10 CMPLEU - Compare Less Than or Equal (Unsigned)

2.10.1 Encoding

	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
Г																																

PRED	0				CC	ONS	TA	NT				I	ROT	ΆΤ	E	0	1	1	1	0	0	1	PD	RS	
PRED	1	0	1	:	SHI	FT	AM'	Γ	SI	ΗF			RT			0	1	1	1	0	0	1	PD	RS	
PRED	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	1	PD	RS	LIM

2.10.2 Syntax

- CMPLEU[PN] Pd, Rs, #Imm
- CMPLEU[PN] Pd, Rs, Rt [SHF #Imm]

2.10.3 Type

CMPLEU is an ALU operation. It may be placed in any slot of a packet. If the long immediate form is used it must not be located in the last slot in a packet, and no instruction may be specified in the following slot.

2.10.4 Behavior

CMPLEU interprets its operands as 32-bit unsigned integers, and writes 1 into the specified predicate register if the value of Rs is less than or equal to the value of the second operand, and 0 otherwise.

2.11 CMPLTS - Compare Less Than (Signed)

2.11.1 Encoding

1 0 9 8	7 6 5 4 3	3 2 1 0 9 8 7	6 5 4 3 2 1 0 9	8 7 6 5 4 3 2 1 0
---------	-----------	---------------	-----------------	-------------------

PRED	0				C	ONS	TA	NT				I	ROT	'AT	E	0	1	1	1	1	0	0	PD	RS	
PRED	1	0	1	1	SHI	FT	AM'	Γ	SI	HF			RT			0	1	1	1	1	0	0	PD	RS	
PRED	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	PD	RS	LIM

2.11.2 Syntax

- CMPLTS[PN] Pd, Rs, #Imm
- CMPLTS[PN] Pd, Rs, Rt [SHF #Imm]

2.11.3 Type

CMPLTS is an ALU operation. It may be placed in any slot of a packet. If the long immediate form is used it must not be located in the last slot in a packet, and no instruction may be specified in the following slot.

2.11.4 Behavior

CMPLTS interprets its operands as 32-bit two's-complement signed integers, and writes 1 into the specified predicate register if the value of Rs is less than to the value of the second operand, and 0 otherwise.

2.12 CMPLTU - Compare Less Than (Unsigned)

2.12.1 Encoding

	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0

PRED	0				C	ONS	TA	NT				I	ROT	'AT	Е	0	1	1	1	0	0	0	PD	RS	
PRED	1	0	1		SHI	FT	AM.	Γ	SI	ΗF			RT			0	1	1	1	0	0	0	PD	RS	
PRED	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	PD	RS	LIM

2.12.2 Syntax

- CMPLTU[PN] Pd, Rs, #Imm
- CMPLTU[PN] Pd, Rs, Rt [SHF #Imm]

2.12.3 Type

CMPLTU is an ALU operation. It may be placed in any slot of a packet. If the long immediate form is used it must not be located in the last slot in a packet, and no instruction may be specified in the following slot.

2.12.4 Behavior

CMPLTU interprets its operands as 32-bit unsigned integers, and writes 1 into the specified predicate register if the value of Rs is less than to the value of the second operand, and 0 otherwise. P3 is pinned to 1. Writes into P3 from compare instructions are ignored.

2.13 CPOP - TODO XXX

2.13.1 Encoding

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1										
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
P	RE	D	1	0	0	0	1	0	1	0	1	С	X	X	X	Х	Х		CPO	DPC	;	Х	X	Х	X	X			RS		

2.14 DIV - Integer Division

2.14.1 Encoding

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1										
_	-	-	-	•	-	-	_	-	_	_	-	-	-	•	-	-	_	-	_	_	-	-	-	•	-	-	_	-	2	_	-
P	RE:	D	1	0	0	0	1	1	0	0	1	Х			RT			Х	Х	Х	Х			RD					RS		

2.14.2 Syntax

• DIV[PN] Rd, Rs, Rt

2.14.3 Type

DIV is a Control operation. It may only be placed as the first instruction in a packet.

2.14.4 Behavior

Performs the integer division Rs / Rt, writing the quotient into Rd and the remainder into the multiply/divide overflow register.

2.15 ERET - Return from Exception

2.15.1 Encoding

	3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1										
	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
ſ	P	RE	D	1	0	0	0	1	0	1	0	0	Х	Х	Х	Х	Х	Х	Х	Х	Х	Х	Х	Х	Х	Х	Х	Х	Х	Х	Х	Х

2.15.2 Syntax

• ERET[PN]

2.15.3 Type

ERET is a Control operation. It may only be placed as the first instruction in a packet. It may only be used when the processor is in kernel mode.

2.15.4 Behavior

ERET returns the processor to user mode from kernel mode, and branches to the memory address contained in the Error PC control register. Other instructions in the same packet as the ERET instruction are executed in kernel mode before control is transferred. This instruction also clears the memory link bit.

2.16 FENCE - TODO XXX

2.16.1 Encoding

3	3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1										
1	()	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
	PR	ED)	1	0	0	0	1	0	0	1	1	Х	Х	Х	Х	Х	Х	Х	Х	X	X	Х	X	Х	Х	Х	Х	X	X	Х	X

2.17 LB - Load Byte

2.17.1 Encoding

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1										
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
P	RE	D	1	0	0	1					C)FF	SE	Γ					0	0	0			RD					RS		

2.17.2 Syntax

• LB[PN] Rd, (Rs [+ OFFSET])

2.17.3 Type

LB is a Memory operation. It may only be placed in the first or second slot in an instruction packet.

2.17.4 Behavior

LB loads a byte from the specified address into Rd. The address is computed by sign-extending the OFFSET field to 32 bits and adding it to the value of Rs. The high 24 bits of Rd are zeroed.

2.18 LH - Load Half-Word

2.18.1 Encoding

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1										
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
P	RE.	D	1	0	0	1					()FF	SE	Γ					0	0	1			RD					RS		

2.18.2 Syntax

• LH[PN] Rd, (Rs [+ OFFSET])

2.18.3 Type

LH is a Memory operation. It may only be placed in the first or second slot in an instruction packet.

2.18.4 Behavior

LH loads two bytes from the specified address into Rd. The address is computed by sign-extending the OFFSET field to 32 bits and adding it to the value of Rs. Only 2-byte aligned addresses may be accessed - the low bit of the address is ignored. The high 16 bits of Rd are zeroed.

2.19 LL - Load Linked

2.19.1 Encoding

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1										
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
P	RE.	D	1	0	0	1					C	FF	'SE'	Γ					0	1	1			RD					RS		

2.19.2 Syntax

• LL[PN] Rd, (Rs [+ OFFSET])

2.19.3 Type

LL is a Memory operation. It may only be placed in the first or second slot in an instruction packet.

2.19.4 Behavior

LL loads a word from the specified address into Rd. The address is computed by sign-extending the OFFSET field to 32 bits and adding it to the value of Rs. Only 4-byte aligned addresses may be accessed - the low 2 bits of the address are ignored.

Additionally, LL sets the link bit. A future SC operation will only complete if no interrupt, exception, or ERET clears the link bit in the intervening period.

2.20 LW - Load Word

2.20.1 Encoding

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1										
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
P	RE.	D	1	0	0	1					()FF	'SE'	Т					0	1	0			RD					RS		

2.20.2 Syntax

• LW[PN] Rd, (Rs [+ OFFSET])

2.20.3 Type

LW is a Memory operation. It may only be placed in the first or second slot in an instruction packet.

2.20.4 Behavior

LW loads a word from the specified address into Rd. The address is computed by sign-extending the OFFSET field to 32 bits and adding it to the value of Rs. Only 4-byte aligned addresses may be accessed - the low 2 bits of the address are ignored.

2.21 MFC - Move From Coprocessor Register

2.21.1 Encoding

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1										
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
P	RE.	D	1	0	0	0	1	0	1	1	0	Х	X	Х	Х	X	Х	Х	Х	X	Х			RD				С	PR	3	

2.21.2 Syntax

• MFC[PN] Rd, CPRs

2.21.3 Type

MFC is a Control operation. It may only be placed as the first instruction in a packet. It may only be used when the processor is in kernel mode.

2.21.4 Behavior

MFC moves the contents of the specified coprocessor register into the specified general-purpose register. See TODO XXX Link for a description of the available coprocessor registers and their mnemonics.

2.22 MFHI - Move From Multiply/Division Overflow Register

2.22.1 Encoding

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1										
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
P	RE	D	1	0	0	0	1	0	1	1	1	Х	X	Х	Х	Х	X	X	Х	X	Х			RD			Х	X	Х	X	Х

2.22.2 Syntax

• MFHI[PN] Rd

2.22.3 Type

MFHI is a Control operation. It may only be placed as the first instruction in a packet.

2.22.4 Behavior

MFHI moves the contents of the multiply/division overflow register into the specified general-purpose register. This register stores the high bits of results of multiplication operations, or the remainders generated by division operations.

2.23 MOV - Move

2.23.1 Encoding

0 9 8 7 6 5 4 3 2 1 0 9	3 7 6 5 4 3 2 1 0	9 8 7 6 5 4 3 2 1 0
-------------------------	-------------------	---------------------

PRED	0				CO	ONS	TL	OW				I	ROT	'AT	E	1	0	0	0	RD	CONSTHI	
PRED	1	0	1		SHI	FT	AM'	Γ	SI	ΗF	Х	X	Х	X	Х	1	0	0	0	RD	RS	
PRED	1	0	0	0	0	0	0	1	SI	HF			RT			1	0	0	0	RD	RS	
PRED	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	RD	RS	LIM

2.23.2 Syntax

- MOV[PN] Rd, #Imm
- MOV[PN] Rd, Rs [SHF #Imm]
- MOV[PN] Rd, Rs SHF Rt

2.23.3 Type

MOV is an ALU operation. It may be placed in any slot of a packet. If the long immediate form is used it must not be located in the last slot in a packet, and no instruction may be specified in the following slot.

2.23.4 Behavior

MOV moves the value of its source operand into the specified destination register. If the second operand is a register, it may be shifted by an immediate value or by the contents of a register before being stored.

2.24 MTC - Move To Coprocessor Register

2.24.1 Encoding

P	RE.	D	1	0	0	0	1	0	1	1	1	Х	Х	Х	Х	X	Х	Х	Х	Х	Х		C	PR	D				RS			
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	
3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1											

2.24.2 Syntax

• MTC[PN] CPRd, Rs

2.24.3 Type

MTC is a Control operation. It may only be placed as the first instruction in a packet. It may only be used when the processor is in kernel mode.

2.24.4 Behavior

MTC moves the contents of the selected general-purpose register into the specified coprocessor register. See TODO XXX Link for a description of the available coprocessor registers and their mnemonics.

2.25 MTHI - Move To Multiply/Division Overflow Register

2.25.1 Encoding

3 3 2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1										
1 0 9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
PRED	1	0	0	0	1	0	1	1	1	Х	Х	Х	Х	Х	Х	Х	Х	Х	Х	Х	Х	Х	Х	Х			RS		

2.25.2 Syntax

• MTHI[PN] Rs

2.25.3 Type

MTHI is a Control operation. It may only be placed as the first instruction in a packet.

2.25.4 Behavior

 MTHI moves the value of the specified general-purpose register into the multiply/division overflow register.

2.26 MVN - Move and Invert

2.26.1 Encoding

1 0 3																														
1 0 0	8	7	6	5	1	3	2	2	1	Λ	a	2	7	6	5	1	3	2	1	\cap	a	2	7	6	_	1	3	2	1	Λ

PRED	0				CO	ONS	TL	OW]	ROT	'AT	E	1	0	0	1	RD	CONSTHI	
PRED	1	0	1	:	SHI	FT	AM'	Γ	Sl	HF	Х	X	Х	X	Х	1	0	0	1	RD	RS	
PRED	1	0	0	0	0 0 0 0 1					HF			RT			1	0	0	1	RD	RS	
PRED	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	RD	RS	LIM

2.26.2 Syntax

- MVN[PN] Rd, #Imm
- MVN[PN] Rd, Rs [SHF #Imm]
- MVN[PN] Rd, Rs SHF Rt

2.26.3 Type

MVN is an ALU operation. It may be placed in any slot of a packet. If the long immediate form is used it must not be located in the last slot in a packet, and no instruction may be specified in the following slot.

2.26.4 Behavior

MVN inverts its source operand, storing the result in the specified destination register. If the second operand is a register, it may be shifted by an immediate value or by the contents of a register before being inverted and stored.

2.27 MULT - Integer Multiplication

2.27.1 Encoding

P	RE	D	1	0	0	0	1	1	0	0	0	S			RT			Х	Х	Х	Х			RD					RS			
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	
3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1											

2.27.2 Syntax

- MULT[PN] Rd, Rs, Rt
- MULTS[PN] Rd, Rs, Rt

2.27.3 Type

MULT is a Control operation. It may only be placed as the first instruction in a packet.

2.27.4 Behavior

MULT performs an integer multiplication on its two register operands. If the S bit is set, the operands are interpreted as two's complement signed integers; otherwise they are interpreted as unsigned. The low 32 bits of the 64 bit result are placed in the specified destination register, and the high 32 bits are placed in the special-purpose multiply/division overflow register, where they are accessible via the MFHI instruction.

2.28 NOR - Bitwise Logical NOR

2.28.1 Encoding

1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	
D	יים מ	n _	O CONCTANT								Т	тО	י זידי אי	7	$\overline{}$	$\overline{}$	1	$\overline{}$			ממ			T		חמ			1			

PRED	0				CC	ONS	TA	NT				I	ROT	'AT	E	0	0	1	0	RD	RS	
PRED	1	0	1	5	SHI	FT	AM'	Γ	SI	HF			RT			0	0	1	0	RD	RS	
PRED	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	RD	RS	LIM

2.28.2 Syntax

- NOR[PN] Rd, Rs, #Imm
- NOR[PN] Rd, Rs, Rt [SHF #Imm]

2.28.3 Type

NOR is an ALU operation. It may be placed in any slot of a packet. If the long immediate form is used it must not be located in the last slot in a packet, and no instruction may be specified in the following slot.

2.28.4 Behavior

Computes the bitwise logical NOR of the two operands, storing the result in the destination register. If two register operands are specified, the second may optionally be shifted by an immediate value before the computation is performed.

2.29 OR - Bitwise Logical OR

2.29.1 Encoding

1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	
P	RE	D	0				CC	ONS	TA	NT				I	ROT	'AT	Ε	0	0	1	1			RD					RS			
Р	RE	D	1	0	1	1	SHI	FT	AMT	Γ	SI	ΙF			RT			0	0	1	1			RD					RS			
P	RE	D	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1			RD					RS			LIM

2.29.2 Syntax

- OR[PN] Rd, Rs, #Imm
- OR[PN] Rd, Rs, Rt [SHF #Imm]

2.29.3 Type

OR is an ALU operation. It may be placed in any slot of a packet. If the long immediate form is used it must not be located in the last slot in a packet, and no instruction may be specified in the following slot.

2.29.4 Behavior

Computes the bitwise logical OR of the two operands, storing the result in the destination register. If two register operands are specified, the second may optionally be shifted by an immediate value before the computation is performed.

2.30 RSB - Reverse Subtraction

2.30.1 Encoding

 $3 \ \ 3 \ \ 2 \ \ 2 \ \ 2 \ \ 2 \ \ 2 \ \ 2 \ \ 2 \ \ 1 \ \ 1 \ \ 1 \ \ 1 \ \ 1 \ \ 1 \ \ 1 \ \ 1 \ \ 1 \ \ 1$

	1 0 9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
ĺ	PRED	0				CC	ONS	STAI	NT]	ROT	'AT	E	0	1	0	1			RD					RS		
İ	PRED	1	0	1	:	SHI	FT	'AM'	Т	SI	HF			RT			0	1	0	1			RD					RS		
Ī	PRED	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1			R.D					R.S		

2.30.2 Syntax

- RSB[PN] Rd, Rs, #Imm
- RSB[PN] Rd, Rs, Rt [SHF #Imm]

2.30.3 Type

RSB is an ALU operation. It may be placed in any slot of a packet. If the long immediate form is used it must not be located in the last slot in a packet, and no instruction may be specified in the following slot.

2.30.4 Behavior

Subtracts the value stored in RS from the value of the second operand, storing the result in the destination register. If two register operands are specified, the second may optionally be shifted by an immediate value before the computation is performed.

2.31 SB - Store Byte

2.31.1 Encoding

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1										
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
P	PRED		1	0	0	1		OFI	SE	TH:	IGH	Ī.			RT			М	1	0	0	0	FF	SET	LO	W			RS		

2.31.2 Syntax

• SB[PN] Rt, (Rs [+ OFFSET])

2.31.3 Type

SB is a Memory operation. It may only be placed in the first or second slot in an instruction packet.

2.31.4 Behavior

SB stores the low byte of Rt at the provided address. The address is computed by sign-extending the OFFSET field to 32 bits and adding it to the value of Rs.

2.32 SC - Store Conditional

2.32.1 Encoding

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1										
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
P	RE	D	1	0	0	1		OFE	SE	TH:	IGH	I			RT			М	1	1	1	0	FF	SET	LO	W			RS		

2.32.2 Syntax

• SC[PN] Rt, (Rs [+ OFFSET])

2.32.3 Type

SC is a Memory operation. It may only be placed in the first or second slot in an instruction packet.

2.32.4 Behavior

SC stores the value in Rt at the provided address, if the processor's link bit is set. The address is computed by sign-extending the OFFSET field to 32 bits and adding it to the value of Rs. Only 4-byte aligned addresses may be accessed - the low 2 bits of the address are ignored.

The link bit is set by executing the LL instruction, and is cleared by the SC or ERET instructions, or any time an exception or interrupt is dispatched. SC only completes if the link bit is set when it is executed - thus, if it completes, all instructions since the last LL instruction are known to have been executed atomically. The SC instruction writes a 1 into predicate register 0 if the store completes successfully, or a 0 if it fails.

2.33 SH - Store Half-Word

2.33.1 Encoding

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1										
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
P	RE	D	1	0	0	1		OFFSETHIO				[RT			М	1	0	1	0	FF	SET	LO	W			RS		

2.33.2 Syntax

• SH[PN] Rt, (Rs [+ OFFSET])

2.33.3 Type

SH is a Memory operation. It may only be placed in the first or second slot in an instruction packet.

2.33.4 Behavior

SH stores the low two bytes of Rt at the provided address. The address is computed by sign-extending the OFFSET field to 32 bits and adding it to the value of Rs. The address must be aligned to a multiple of 2; the low bit of the address is ignored.

2.34 SUB - Subtraction

2.34.1 Encoding

1 0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
																														$\overline{}$

PRED	0				CC	ONS	TA	NT				I	ROT	'AT	Е	0	1	0	0	RD	RS	
PRED	1	0	1	S	SHI	FT	AM.	Γ	SI	ΙF			RT			0	1	0	0	RD	RS	
PRED	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	RD	RS	LIM

2.34.2 Syntax

- SUB[PN] Rd, Rs, #Imm
- SUB[PN] Rd, Rs, Rt [SHF #Imm]

2.34.3 Type

SUB is an ALU operation. It may be placed in any slot of a packet. If the long immediate form is used it must not be located in the last slot in a packet, and no instruction may be specified in the following slot.

2.34.4 Behavior

Subtracts the value of the second operand from the value in Rs, storing the result in the destination register. If two register operands are specified, the second may optionally be shifted by an immediate value before the computation is performed.

2.35 SW - Store Word

2.35.1 Encoding

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1										
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
P	PRED 1		1	0	0	1		OFI	SE	TH:	IGH	Ī.			RT			М	1	1	0	0	FF	SET	LO	W			RS		

2.35.2 Syntax

• SW[PN] Rt, (Rs [+ OFFSET])

2.35.3 Type

SW is a Memory operation. It may only be placed in the first or second slot in an instruction packet.

2.35.4 Behavior

SW stores the contents of Rt at the provided address. The address is computed by sign-extending the OFFSET field to 32 bits and adding it to the value of Rs. The address must be aligned to a multiple of 4; the low two bits of the address are ignored.

2.36 SXB - Sign-Extend Byte

2.36.1 Encoding

1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

PRED	0				C	ONS	TL	OW]	ROT	ΆΤ	E	1	0	1	0	RD	CONSTHI	
PRED	1	0	1	:	SHIFTAMT					HF	Х	X	Х	X	Х	1	0	0	1	RD	RS	
PRED	1	0	0	0	0	0	1	SI	HF			RT			1	0	1	0	RD	RS		
PRED	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	RD	RS	LIM

2.36.2 Syntax

- SXB[PN] Rd, #Imm
- SXB[PN] Rd, Rs [SHF #Imm]
- SXB[PN] Rd, Rs SHF Rt

2.36.3 Type

SXB is an ALU operation. It may be placed in any slot of a packet. If the long immediate form is used it must not be located in the last slot in a packet, and no instruction may be specified in the following slot.

2.36.4 Behavior

SXB moves the low byte of its source operand into the specified destination register, sign-extending it across the rest of the register. If the second operand is a register, it may be shifted by an immediate value or by the contents of a register before being stored.

SXH - Sign-Extend Half-Word 2.37

2.37.1Encoding

1 0 9	Ö	, ,	0 5	4	3	2	T	U	9	Ö	1	О	Э	4	3	2	Τ	U	9	Ö	1	О	Э	4	3	2	T	U	
PRED	0			C	ONS	TL(JW]		'AT	Ε	1	0	1	1			RD				CO	NST	ГΗΙ		

PRED	0	O CONSTLOW												'AT	Ε	1	0	1	1	RD	CONSTHI	
PRED	1	0	1	:	SHIFTAMT					ΗF	Х	X	Х	Х	X	1	0	1	1	RD	RS	
PRED	1	0	0	0	0	0	0	1	SI	SHF			RT			1	0	1	1	RD	RS	
PRED	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	RD	RS	LIM

Syntax 2.37.2

- SXH[PN] Rd, #Imm
- SXH[PN] Rd, Rs [SHF #Imm]
- SXH[PN] Rd, Rs SHF Rt

2.37.3Type

SXH is an ALU operation. It may be placed in any slot of a packet. If the long immediate form is used it must not be located in the last slot in a packet, and no instruction may be specified in the following slot.

2.37.4Behavior

SXH moves the low two bytes of its source operand into the specified destination register, signextending it across the rest of the register. If the second operand is a register, it may be shifted by an immediate value or by the contents of a register before being stored.

2.38 SYSCALL - TODO XXX

2.38.1 Encoding

3	3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1										
-	L	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
	P	REI	D	1	0	0	0	1	0	0	1	0	Х	Х	Х	Х	Х	Х	Х	Х	Х	Х	Х	Х	Х	Х	Х	Х	Х	Х	Х	Х

2.39 XOR - Bitwise Exclusive OR

2.39.1 Encoding

	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	
ĺ	P	RE	D	0			CONSTANT									ROT	'AT	E	0	1	1	0	RD					RS					
	Р	RE	D	1	0	1	1	SHI	FT	AM.	Γ	SI	ΗF			RT			0	1	1	0			RD					RS			
ĺ	Р	RE	D	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0			RD					RS			LIM

2.39.2 Syntax

- XOR[PN] Rd, Rs, #Imm
- XOR[PN] Rd, Rs, Rt [SHF #Imm]

2.39.3 Type

XOR is an ALU operation. It may be placed in any slot of a packet. If the long immediate form is used it must not be located in the last slot in a packet, and no instruction may be specified in the following slot.

2.39.4 Behavior

Computes the bitwise logical XOR of the two operands, storing the result in the destination register. If two register operands are specified, the second may optionally be shifted by an immediate value before the computation is performed.