

Draft, Do not distribute

h.sfeq

Set Flag if Equal

h.sfeq

15	8	7	4	3	0
opcode 0x40		A		B	
8 bits		4 bits		4 bits	

Format:

h.sfeq rA,rB

Description:

The contents of general register rA and the contents of general register rB are compared. If the two registers are equal, then the compare flag is set; otherwise the compare flag is cleared.

Operation:

flag <- rA == rB

Notes:

Class 1:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Mandatory always

h.sfne

Set Flag if Not Equal

h.sfne

15	8	7	4	3	0
opcode 0x41		A		B	
8 bits		4 bits		4 bits	

Format:

h.sfne rA,rB

Description:

The contents of general register rA and the contents of general register rB are compared. If the two registers are not equal, then the compare flag is set; otherwise the compare flag is cleared.

Operation:

flag <- rA != rB

Notes:

Class 1:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Mandatory always

h.sfgts Set Flag if Greater Than Signed h.sfgts

15	8	7	4	3	0
opcode 0x42		A		B	
8 bits		4 bits		4 bits	

Format:

`h.sfgts rA,rB`

Description:

The contents of general register rA and the contents of general register rB are compared as signed integers. If the contents of the first register are greater than the contents of the second register, then the compare flag is set; otherwise the compare flag is cleared.

Operation:

`flag <- rA > rB`

Notes:

Class 1:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Mandatory always

h.sfges Set Flag if Greater or Equal Than Signed h.sfges

15	8	7	4	3	0
opcode 0x43		A		B	
8 bits		4 bits		4 bits	

Format:

h.sfges rA,rB

Description:

The contents of general register rA and the contents of general register rB are compared as signed integers. If the contents of the first register are greater or equal than the contents of the second register, then the compare flag is set; otherwise the compare flag is cleared.

Operation:

flag <- rA >= rB

Notes:

Class 1:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Mandatory always

h.sflts Set Flag if Less Than Signed h.sflts

15	8	7	4	3	0
opcode 0x44		A		B	
8 bits		4 bits		4 bits	

Format:

`h.sflts rA,rB`

Description:

The contents of general register rA and the contents of general register rB are compared as signed integers. If the contents of the first register are less than the contents of the second register, then the compare flag is set; otherwise the compare flag is cleared.

Operation:

`flag <- rA < rB`

Notes:

Class 1:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Mandatory always

h.sfles Set Flag if Less or Equal Than Signed h.sfles

15	8	7	4	3	0
opcode 0x45		A		B	
8 bits		4 bits		4 bits	

Format:

h.sfles rA,rB

Description:

The contents of general register rA and the contents of general register rB are compared as signed integers. If the contents of the first register are less or equal than the contents of the second register, then the compare flag is set; otherwise the compare flag is cleared.

Operation:

flag <- rA <= rB

Notes:

Class 1:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Mandatory always

h.sfgtu Set Flag if Greater Than Unsigned h.sfgtu

15	8	7	4	3	0
opcode 0x46		A		B	
8 bits		4 bits		4 bits	

Format:

h.sfgtu rA,rB

Description:

The contents of general register rA and the contents of general register rB are compared as unsigned integers. If the contents of the first register are greater than the contents of the second register, then the compare flag is set; otherwise the compare flag is cleared.

Operation:

flag <- rA > rB

Notes:

Class 1:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Mandatory always

h.sfgeu Set Flag if Greater or Equal Than Unsigned h.sfgeu

15	8	7	4	3	0
opcode 0x47		A		B	
8 bits		4 bits		4 bits	

Format:

h.sfgeu rA,rB

Description:

The contents of general register rA and the contents of general register rB are compared as unsigned integers. If the contents of the first register are greater or equal than the contents of the second register, then the compare flag is set; otherwise the compare flag is cleared.

Operation:

flag <- rA >= rB

Notes:

Class 1:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Mandatory always

h.sftu Set Flag if Less Than Unsigned h.sftu

15	8	7	4	3	0
opcode 0x48		A		B	
8 bits		4 bits		4 bits	

Format:

h.sftu rA,rB

Description:

The contents of general register rA and the contents of general register rB are compared as unsigned integers. If the contents of the first register are less than the contents of the second register, then the compare flag is set; otherwise the compare flag is cleared.

Operation:

flag <- rA < rB

Notes:

Class 1:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Mandatory always

h.sfleu Set Flag if Less or Equal Than Unsigned h.sfleu

15	8	7	4	3	0
opcode 0x49		A		B	
8 bits		4 bits		4 bits	

Format:

h.sfleu rA,rB

Description:

The contents of general register rA and the contents of general register rB are compared as unsigned integers. If the contents of the first register are less or equal than the contents of the second register, then the compare flag is set; otherwise the compare flag is cleared.

Operation:

flag <- rA <= rB

Notes:

Class 1:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Mandatory always

h.exths Extend Half Word with Sign h.exths

15	8	7	4	3	0
opcode 0x4b		A		opcode 0x0	
8 bits		4 bits		4 bits	

Format:

h.exths rA

Description:

Bit 15 of general register rA is placed in high-order 16 bits of general register rA. The low-order 16 bits of general register rA are left unchanged.

Operation:

$rA[31:16] \leftarrow rA[15]$

Notes:

Class 2:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Recommended

h.exthz Extend Half Word with Zero h.exthz

15	8	7	4	3	0
opcode 0x4b		A		opcode 0x1	
8 bits		4 bits		4 bits	

Format:

h.exthz rA

Description:

Zero is placed in high-order 16 bits of general register rA. The low-order 16 bits of general register rA are left unchanged.

Operation:

rA[31:16] <- 0

Notes:

Class 2:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Recommended

h.extbs

Extend Byte with Sign

h.extbs

15	8	7	4	3	0
opcode 0x4b		A		opcode 0x2	
8 bits		4 bits		4 bits	

Format:

h.extbs rA

Description:

Bit 7 of general register rA is placed in high-order 24 bits of general register rA. The low-order eight bits of general register rA are left unchanged.

Operation:

rA[31:8] <- rA[7]

Notes:

Class 2:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Recommended

h.extbz

Extend Byte with Zero

h.extbz

15	8	7	4	3	0
opcode 0x4b		A		opcode 0x3	
8 bits		4 bits		4 bits	

Format:

h.extbz rA

Description:

Zero is placed in high-order 24 bits of general register rA. The low-order eight bits of general register rA are left unchanged.

Operation:

rA[31:8] <- 0

Notes:

Class 2:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Recommended

h.nop

No Operation

h.nop

15	0
opcode 0x4b04	
16 bits	

Format:

h.nop

Description:

This instruction does not do anything except it takes one clock cycle to complete.

Operation:

Notes:

Class 1:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Mandatory always

h.jalr Jump and Link Register h.jalr

15	8	7	4	3	0
opcode 0x4b		A		opcode 0x5	
8 bits		4 bits		4 bits	

Format:

h.jalr rA

Description:

The contents of general register rA is effective address of the jump. The program unconditionally jumps to EA with one delay slot. The address of the instruction after h.jalr is placed in the link register.

Operation:

PC <- rA
LR <- h.jalr + 2

Notes:

Class 1:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Mandatory always

h.adds

Add Signed

h.adds

15	12	11	8	7	4	3	0
opcode 0x7		D		A		B	
4 bits		4 bits		4 bits		4 bits	

Format:

h.adds rA,rB,rD

Description:

The contents of general register rC is added to the contents of general register rB to form the result. The result is placed into general register rA.

Operation:

rA <- rB + rC

Notes:

Class 1:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Mandatory always

h.bnf

Branch if No Flag

h.bnf

15	12	11	0
opcode 0xa		X	
4 bits		12 bits	

Format:

h.bnf X

Description:

The immediate(12 bits) is shifted left one bit, sign-extended to 32 bits and then added to the address of h.bnf. The result is effective address of the branch. If the compare flag is cleared, then the program branches to EA with no delay.

Operation:

EA <- (Immediate || 0) + InsnAddr(h.bnf)
PC <- EA if flag cleared

Notes:

Class 1:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Mandatory always

h.bf

Branch if Flag

h.bf

15	12	11	0
opcode 0xb		X	
4 bits		12 bits	

Format:

h.bf X

Description:

The immediate(12 bits) is shifted left one bit, sign-extended to 32 bits and then added to the address of h.bf. The result is effective address of the branch. If the compare flag is set, then the program branches to EA with no delay slot.

Operation:

EA <- (Immediate || 0) + InsnAddr(h.bf)
PC <- EA if flag set

Notes:

Class 1:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Mandatory always

h.movibse Immediate Byte Signed Extended h.movibse

15	12	11	8	7	4	3	0
opcode 0xc		M		A		M	
4 bits		4 bits		4 bits		4 bits	

Format:

h.movibse rA,M

Description:

8 bit immediate is sign-extended to 32 bits and placed into general register rA.

Operation:

rA <- exts(Immediate)

Notes:

Class 2:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Recommended

h.movhbsl

h.movhbsl

15	12	11	8	7	4	3	0
opcode 0x2		M		A		M	
4 bits		4 bits		4 bits		4 bits	

Format:

h.movhbsl rA,M

Description:

Operation:

Notes:

Class 0:	Architecture Level	Execution Mode	Implementation

h.movhibse

h.movhibse

15	12	11	8	7	4	3	0
opcode 0x3		M		A		M	
4 bits		4 bits		4 bits		4 bits	

Format:

h.movhibse rA,M

Description:

Operation:

Notes:

Class 0:	Architecture Level	Execution Mode	Implementation

h.mov

Move

h.mov

15	8	7	4	3	0
opcode 0xe0		A		B	
8 bits		4 bits		4 bits	

Format:

h.mov rA,rB

Description:

The contents of general register rB are moved into general register rA.

Operation:

rA <- rB

Notes:

Class 2:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Recommended

h.brk

software break

h.brk

15	8	7	0
opcode 0xd0		M	
8 bits		8 bits	

Format:

h.brk M

Description:

Software break insn to generate unconditional exception. PC will be transferred to exception service routine at 0xD00 and enters Supervisor mode.

Operation:

PC <- 0x00000D00

Notes:

Class 4:	Architecture Level	Execution Mode	Implementation
	System Management	Supervisor only	Mandatory always

h.sys

software system call

h.sys

15	8	7	0
opcode 0xd8		M	
8 bits		8 bits	

Format:

h.sys M

Description:

Software system call insn to generate unconditional exception. PC will be transferred to exception service routine at 0xC00 and enters Supervisor mode.

Operation:

PC <- 0x00000C00

Notes:

Class 4:	Architecture Level	Execution Mode	Implementation
	System Management	Supervisor only	Mandatory always

h.rfe return from software/hardware exception h.rfe

15	0
opcode 0xd900	
16 bits	

Format:

h.rfe

Description:

Rfe notifies CPU to enter User mode.

Operation:

Notes:

Class 4:	Architecture Level	Execution Mode	Implementation
	System Management	Supervisor only	Mandatory always

h.mas multiply signed and accumulate the product h.mas

15	8	7	4	3	0
opcode 0xd3		A		B	
8 bits		4 bits		4 bits	

Format:

h.mas rA,rB

Description:

The contents of general register rA and the contents of general register rB are multiplied and the product is added to (MACHI || MACLO) in group 0. Both operands are treated as signed integers. MACHI is located in address 3 and MACLO is in address 4 of group 0

Operation:

$(MACHI \parallel MACLO) \leftarrow (MACHI \parallel MACLO) + rA * rB$

Notes:

Class 2:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Recommended

h.mss multiply signed and accumulate the product h.mss

15	8	7	4	3	0
opcode 0xd2		A		B	
8 bits		4 bits		4 bits	

Format:

h.mss rA,rB

Description:

The contents of general register rA and the contents of general register rB are multiplied and the product is subtracted from (MACHI || MACLO) in group 0. Both operands are treated as signed integers. MACHI is located in address 3 and MACLO is in address 4 of group 0

Operation:

$(\text{MACHI} \parallel \text{MACLO}) \leftarrow (\text{MACHI} \parallel \text{MACLO}) - rA * rB$

Notes:

Class 2:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Recommended

h.mods

Modulus Signed

h.mods

15	8	7	4	3	0
opcode 0xd5		A		B	
8 bits		4 bits		4 bits	

Format:

h.mods rA,rB

Description:

The contents of general register rA are divided by the contents of general register rB and the result is placed into general register rA. Both operands are treated as signed integers. A divisor flag is set when the divisor is zero.

Operation:

rA <- rA MOD rB

Notes:

Class 3:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Optional

h.modu

Modulus Unsigned

h.modu

15	8	7	4	3	0
opcode 0xd7		A		B	
8 bits		4 bits		4 bits	

Format:

h.modu rA,rB

Description:

The contents of general register rA are divided by the contents of general register rB and the result is placed into general register rA. Both operands are treated as unsigned integers. A divisor flag is set when the divisor is zero.

Operation:

rA <- rA MOD rB

Notes:

Class 3:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Optional

h.muls

Multiply Signed

h.muls

15	8	7	4	3	0
opcode 0x4c		A		B	
8 bits		4 bits		4 bits	

Format:

h.muls rA,rB

Description:

The contents of general register rB and the contents of general register rC are multiplied and the lower 32 bits of the product are placed into general register rA and the higher 32 bits are placed in MACHI. Both operands are treated as signed integers.

Operation:

rA <- rB * rC

Notes:

Class 2:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Recommended

h.mulu

Multiply Unsigned

h.mulu

15	8	7	4	3	0
opcode 0x4d		A		B	
8 bits		4 bits		4 bits	

Format:

h.mulu rA,rB

Description:

The contents of general register rB and the contents of general register rC are multiplied and the lower 32 bits of the product are placed into general register rA and the higher 32 bits are placed in MACHI. Both operands are treated as unsigned integers.

Operation:

rA <- rB * rC

Notes:

Class 2:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Recommended

h.divs

Divide Signed

h.divs

15	8	7	4	3	0
opcode 0x4e		A		B	
8 bits		4 bits		4 bits	

Format:

h.divs rA,rB

Description:

The contents of general register rA are divided by the contents of general register rB and the result is placed into general register rA. Both operands are treated as signed integers. A divisor flag is set when the divisor is zero.

Operation:

$rA \leftarrow rA / rB$

Notes:

Class 3:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Optional

h.divu

Divide Unsigned

h.divu

15	8	7	4	3	0
opcode 0x4f		A		B	
8 bits		4 bits		4 bits	

Format:

h.divu rA,rB

Description:

The contents of general register rA are divided by the contents of general register rB and the result is placed into general register rA. Both operands are treated as unsigned integers. A divisor flag is set when the divisor is zero.

Operation:

rA <- rA / rB

Notes:

Class 3:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Optional

h.notu

Not(1s complement)

h.notu

15	8	7	4	3	0
opcode 0xe0		A		B	
8 bits		4 bits		4 bits	

Format:

h.notu rA,rB

Description:

The contents of general register rB is bit-wise inverted The result is placed into general register rA.

Operation:

rA <- rB

Notes:

Class 1:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Mandatory always

h.negs Negated(2s complement) h.negs

15	8	7	4	3	0
opcode 0xe1		A		B	
8 bits		4 bits		4 bits	

Format:

h.negs rA,rB

Description:

The contents of general register rB is bit-wise inverted and added by 1 The result is placed into general register rA.

Operation:

rA <- -rB

Notes:

Class 1:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Mandatory always

h.xor

Exclusive Or

h.xor

15	8	7	4	3	0
opcode 0xe2		A		B	
8 bits		4 bits		4 bits	

Format:

h.xor rA,rB

Description:

The contents of general register rA are combined with the contents of general register rB in a bit-wise logical XOR operation. The result is placed into general register rA.

Operation:

rA <- rA XOR rB

Notes:

Class 1:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Mandatory always

h.or

Or

h.or

15	8	7	4	3	0
opcode 0xe3		A		B	
8 bits		4 bits		4 bits	

Format:

h.or rA,rB

Description:

The contents of general register rA are combined with the contents of general register rB in a bit-wise logical OR operation. The result is placed into general register rA.

Operation:

rA <- rA OR rB

Notes:

Class 1:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Mandatory always

h.and

And

h.and

15	8	7	4	3	0
opcode 0xe4		A		B	
8 bits		4 bits		4 bits	

Format:

h.and rA,rB

Description:

The contents of general register rA are combined with the contents of general register rB in a bit-wise logical AND operation. The result is placed into general register rA.

Operation:

rA <- rA AND rB

Notes:

Class 1:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Mandatory always

h.swapb Swap byte to exchange Endian h.swapb

15	8	7	4	3	0
opcode 0xe5		A		B	
8 bits		4 bits		4 bits	

Format:

h.swapb rA,rB

Description:

The bytes of general register rB are swapped mirrorly to exchangethe endianness.The result is placed into general register rA.

Operation:

rA <- (rB[7:0] || rB[15:8] || rB[23:16] || rB[31:24])

Notes:

Class 1:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Mandatory always

h.swapb Swap halfword to exchange Endian h.swapb

15	8	7	4	3	0
opcode 0xe6		A		B	
8 bits		4 bits		4 bits	

Format:

h.swapb rA,rB

Description:

The halfwords of general register rB are swapped mirrorly to exchange the endianness. The result is placed into general register rA.

Operation:

$rA \leftarrow (rB[15:0] \parallel rB[31:16])$

Notes:

Class 1:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Mandatory always

h.shlai Shift Left Arithmetic by Immediate Constant h.shlai

15	9	8	7	4	3	0
opcode 0x75		L	A		L	
7 bits		1 bits	4 bits		4 bits	

Format:

h.shlai rA,L

Description:

The contents of general register rA are shifted left arithmetically by the amount of 5-bit immediate constant and the result is placed in rA.

Operation:

rA <- rA « Immediate

Notes:

Class 1:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Mandatory always

h.shrai Shift Right Arithmetic by Immediate Constant h.shrai

15	9	8	7	4	3	0
opcode 0x76		L	A		L	
7 bits		1 bits	4 bits		4 bits	

Format:

h.shrai rA,L

Description:

The contents of general register rA are shifted right arithmetically by the amount of 5-bit immediate constant and the result is placed in rA.

Operation:

$rA \leftarrow rA \gg \text{Immediate}$

Notes:

Class 1:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Mandatory always

h.shrli Shift Right Logical by Immediate Constant h.shrli

15	9	8	7	4	3	0
opcode 0x77		L	A		L	
7 bits		1 bits	4 bits		4 bits	

Format:

h.shrli rA,L

Description:

The contents of general register rA are shifted right logically by the amount of 5-bit immediate constant and the result is placed in rA.

Operation:

$rA \leftarrow rA \gg \text{Immediate}$

Notes:

Class 1:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Mandatory always

h.shla

Shift Left Arithmetic

h.shla

15	8	7	4	3	0
opcode 0xda		A		B	
8 bits		4 bits		4 bits	

Format:

h.shla rA,rB

Description:

The contents of general register rA are shifted left arithmetically by the amount of rB[4:0] and the result is placed in rA.

Operation:

rA <- rA « rB[4:0]

Notes:

Class 1:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Mandatory always

h.shra

Shift Right Arithmetic

h.shra

15	8	7	4	3	0
opcode 0xdc		A		B	
8 bits		4 bits		4 bits	

Format:

h.shra rA,rB

Description:

The contents of general register rA are shifted right arithmetically by the amount of rB[4:0] and the result is placed in rA.

Operation:

rA <- rA » rB[4:0]

Notes:

Class 1:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Mandatory always

h.shrl

Shift Right Logical

h.shrl

15	8	7	4	3	0
opcode 0xde		A		B	
8 bits		4 bits		4 bits	

Format:

h.shrl rA,rB

Description:

The contents of general register rA are shifted right logically by the amount of rB[4:0] and the result is placed in rA.

Operation:

rA <- rA » rB[4:0]

Notes:

Class 1:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Mandatory always

h.ror

Rotate Right

h.ror

15	8	7	4	3	0
opcode 0xdb		A		B	
8 bits		4 bits		4 bits	

Format:

h.ror rA,rB

Description:

The contents of general register rA are rotated right by the amount of rB[4:0] and the result is placed in rA.

Operation:

$(\text{unused32} \parallel rA) \leftarrow (rA \parallel rA) \gg rB[4:0]$

Notes:

Class 1:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Mandatory always

h.rol

Rotate Left

h.rol

15	8	7	4	3	0
opcode 0xdd		A		B	
8 bits		4 bits		4 bits	

Format:

h.rol rA,rB

Description:

The contents of general register rA are rotated left by the amount of rB[4:0] and the result is placed in rA.

Operation:

$(rA \parallel \text{unused32}) \leftarrow (rA \parallel rA) \ll rB[4:0]$

Notes:

Class 1:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Mandatory always

h.subs

Subtract Signed

h.subs

15	8	7	4	3	0
opcode 0xf0		A		B	
8 bits		4 bits		4 bits	

Format:

h.subs rA,rB

Description:

The contents of general register rB is subtracted from the contents of general register rA to form the result. The result is placed into general register rA.

Operation:

rA <- rA - rB

Notes:

Class 1:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Mandatory always

h.addu

Add UnSigned

h.addu

15	8	7	4	3	0
opcode 0xf1		A		B	
8 bits		4 bits		4 bits	

Format:

h.addu rA,rB

Description:

The contents of general register rA is added to the contents of general register rB to form the result. The result is placed into general register rA.

Operation:

rA <- rA + rB

Notes:

Class 1:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Mandatory always

h.subu

Subtract UnSigned

h.subu

15	8	7	4	3	0
opcode 0xf2		A		B	
8 bits		4 bits		4 bits	

Format:

h.subu rA,rB

Description:

The contents of general register rB is subtracted from the contents of general register rA to form the result. The result is placed into general register rA.

Operation:

rA <- rA - rB

Notes:

Class 1:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Mandatory always

h.addwcs

Add Signed with carry

h.addwcs

15	8	7	4	3	0
opcode 0xf3		A		B	
8 bits		4 bits		4 bits	

Format:

h.addwcs rA,rB

Description:

The contents of general register rA and CARRY is added to the contents of general register rB to form the result. The result is placed into general register rA.

Operation:

rA <- rA + rB + CARRY

Notes:

Class 1:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Mandatory always

h.subwcs Subtract Signed with CARRY h.subwcs

15	8	7	4	3	0
opcode 0xf4		A		B	
8 bits		4 bits		4 bits	

Format:

h.subwcs rA,rB

Description:

The contents of general register rB is subtracted from the contents of general register rA and CARRY to form the result. The result is placed into general register rA.

Operation:

$rA \leftarrow rA - rB + C - 1$

Notes:

Class 1:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Mandatory always

h.test

test bit and set flag

h.test

15	8	7	4	3	0
opcode 0xf5		A		B	
8 bits		4 bits		4 bits	

Format:

h.test rA,rB

Description:

The contents of general register rA are tested If the bit position in rA pointed by rB is compared with 1. If equal, then flag is set to 1, else flag is set to 0.

Operation:

flag <- rA[rB]

Notes:

Class 1:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Mandatory always

h.testi test bit pointed by immediate constant and set flag h.testi

15	9	8	7	4	3	0
opcode 0x7b		M	A		M	
7 bits		1 bits	4 bits		4 bits	

Format:

h.testi rA,M

Description:

The contents of general register rA are tested If the bit position in rA pointed by immediate constant is compared with 1. If equal, then flag is set to 1, else flag is set to 0.

Operation:

flag <- rA[M]

Notes:

Class 1:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Mandatory always

h.clr

clear bit

h.clr

15	8	7	4	3	0
opcode 0xfc		A		B	
8 bits		4 bits		4 bits	

Format:

h.clr rA,rB

Description:

If the bit position in rA pointed by general register rB is cleared to ZERO.

Operation:

rA[rB] <- 0

Notes:

Class 1:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Mandatory always

h.clri clear bit pointed by immediate constant h.clri

15	9	8	7	4	3	0
opcode 0x7c		M	A		M	
7 bits		1 bits	4 bits		4 bits	

Format:

h.clri rA,M

Description:

If the bit position in rA pointed by immediate constant is cleared to ZERO.

Operation:

rA[M] <- 0

Notes:

Class 1:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Mandatory always

h.set

set bit

h.set

15	8	7	4	3	0
opcode 0xfd		A		B	
8 bits		4 bits		4 bits	

Format:

`h.set rA,rB`

Description:

If the bit position in rA pointed by general register rB is set to ONE.

Operation:

`rA[rB] <- 1`

Notes:

Class 1:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Mandatory always

h.seti set bit pointed by immediate constant h.seti

15	9	8	7	4	3	0
opcode 0x7d		M	A		M	
7 bits		1 bits	4 bits		4 bits	

Format:

h.seti rA,M

Description:

If the bit position in rA pointed by immediate constant is set to ONE.

Operation:

rA[M] <- 1

Notes:

Class 1:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Mandatory always

h.jal Jump and Link Register h.jal

15	12	11	0
opcode 0x9		X	
4 bits		12 bits	

Format:

h.jal X

Description:

The immediate(12 bits) is shifted left one bit, sign-extended to 32 bits and then added to the address of h.jal. The result is effective address of the jump. The program unconditionally jumps to EA with one delay slot. The address of the instruction after delay slot is placed in the link register.

Operation:

PC <- exts(Immediate) + InsnAddr(h.jal)
LR <- h.jal + 2

Notes:

Class 1:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Mandatory always

h.jump

Jump

h.jump

15	12	11	0
opcode 0x8		X	
4 bits		12 bits	

Format:

h.jump X

Description:

The immediate(12 bits) is shifted left one bit and added to the contents of general register rA to generate program counter. The result is effective address of the jump. The program unconditionally jumps to EA with one delay slot.

Operation:

PC \leftarrow exts(Immediate) + InsnAddr(h.jump)

Notes:

Class 2:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Recommended

h.jr

Jump Register

h.jr

15	8	7	4	3	0
opcode 0x4b		A		opcode 0x6	
8 bits		4 bits		4 bits	

Format:

h.jr rA

Description:

The contents of general register rA is effective address of the jump. The program unconditionally jumps to EA with one delay slot. The address of the instruction after h.jr is placed in the link register.

Operation:

PC <- rA

Notes:

Class 1:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Mandatory always

h.lw Load Word and Extend with Zero h.lw

15	12	11	8	7	4	3	0
opcode 0x0		N		A		B	
4 bits		4 bits		4 bits		4 bits	

Format:

h.lw rA,N(rB)

Description:

Offset is sign-extended, left shift 2 bits and added to the contents of general register rB. Sum represents effective address. The word in memory addressed by EA is loaded into general register rA.

Operation:

EA <- (exts(Immediate)||00) + rB
rA <- (EA)[31:0]

Notes:

Class 1:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Mandatory always

h.lhz Load Half Word and Extend with Zero h.lhz

15	12	11	8	7	4	3	0
opcode 0x0		N		opcode 0x8		B	
4 bits		4 bits		4 bits		4 bits	

Format:

h.lhz r8,N(rB)

Description:

Offset is sign-extended, left shift 1 bit and added to the contents of general register rB. Sum represents effective address. The half word in memory addressed by EA is loaded into the low-order 16 bits of general register rA. High-order 16 bits of general register rA are replaced with zero.

Operation:

EA <- (exts(Immediate)||0) + rB
rA[15:0] <- (EA)[15:0]
rA[31:16] <- 0

Notes:

Class 1:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Mandatory always

h.lbz Load Byte and Extend with Zero h.lbz

15	12	11	8	7	4	3	0
opcode 0x0		N		opcode 0x9		B	
4 bits		4 bits		4 bits		4 bits	

Format:

h.lbz r9,N(rB)

Description:

Offset is sign-extended and added to the contents of general register rB. Sum represents effective address. The byte in memory addressed by EA is loaded into the low-order eight bits of general register rA. High-order 24 bits of general register rA are replaced with zero.

Operation:

EA <- exts(Immediate) + rB
rA[7:0] <- (EA)[7:0]
rA[31:8] <- 0

Notes:

Class 1:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Mandatory always

h.sw

Store Word

h.sw

15	12	11	8	7	4	3	0
opcode 0x1		N		A		B	
4 bits		4 bits		4 bits		4 bits	

Format:

h.sw N(rA),rB

Description:

Offset is sign-extended, left shift 2 bits and added to the contents of general register rA. Sum represents effective address. The word in general register rB is stored to memory addressed by EA.

Operation:

EA <- (exts(Immediate)||00) + rA
(EA)[31:0] <- rB

Notes:

Class 2:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Recommended

h.sh

Store Half Word

h.sh

15	12	11	8	7	4	3	0
opcode 0x1		N		A		opcode 0x8	
4 bits		4 bits		4 bits		4 bits	

Format:

h.sh N(rA),r8

Description:

Offset is sign-extended, left shift 1 bit and added to the contents of general register rA. Sum represents effective address. The low-order 16 bits of general register rB are stored to memory addressed by EA.

Operation:

EA <- (exts(Immediate)||0) + rA
(EA)[15:0] <- rB[15:0]

Notes:

Class 1:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Mandatory always

h.sb

Store Byte

h.sb

15	12	11	8	7	4	3	0
opcode 0x1		N		A		opcode 0x9	
4 bits		4 bits		4 bits		4 bits	

Format:

h.sb N(rA),r9

Description:

Offset is sign-extended and added to the contents of general register rA. Sum represents effective address. The low-order 8 bits of general register rB are stored to memory addressed by EA.

Operation:

EA <- exts(Immediate) + rA
(EA)[7:0] <- rB[7:0]

Notes:

Class 1:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Mandatory always

h.lwpain Load Word with Post Address Increment Register and Extend with

15	8	7	4	3	0
opcode 0x53		A		N	
8 bits		4 bits		4 bits	

Format:

h.lwpain rA,N

Description:

PAIR(Post Address Increment Register) represents effective address(EA). The word in memory addressed by EA is loaded into general register rA.

Operation:

EA <- PAIR
rA <- (EA)[31:0]
PAIR <- PAIR + N * 4, N= -8 to 7

Notes:

Class 1:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Mandatory always

h.lhzipain Load Half Word with Post Address Increment Register and Ext

15	8	7	4	3	0
opcode 0x54		A		N	
8 bits		4 bits		4 bits	

Format:

h.lhzipain rA,N

Description:

PAIR represents effective address. The half word in memory addressed by EA is loaded into the low-order 16 bits of general register rA. High-order 16 bits of general register rA are replaced with zero.

Operation:

EA <- PAIR
rA[15:0] <- (EA)[15:0]
rA[31:16] <- 0
PAIR <- PAIR + N * 2

Notes:

Class 1:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Mandatory always

h.lbzpain Load Byte with Post Address Increment Register and Extend with

15	8	7	4	3	0
opcode 0x55		A		N	
8 bits		4 bits		4 bits	

Format:

h.lbzpain rA,N

Description:

PAIR represents effective address. The byte in memory addressed by EA is loaded into the low-order eight bits of general register rA. High-order 24 bits of general register rA are replaced with zero.

Operation:

EA <- PAIR
rA[7:0] <- (EA)[7:0]
rA[31:8] <- 0
PAIR <- PAIR + N * 1

Notes:

Class 1:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Mandatory always

h.lhspain

h.lhspain

15	8	7	4	3	0
opcode 0x56		A		N	
8 bits		4 bits		4 bits	

Format:

h.lhspain rA,N

Description:

Operation:

Notes:

Class 0:	Architecture Level	Execution Mode	Implementation

h.lbspain

h.lbspain

15	8	7	4	3	0
opcode 0x57		A		N	
8 bits		4 bits		4 bits	

Format:

h.lbspain rA,N

Description:

Operation:

Notes:

Class 0:	Architecture Level	Execution Mode	Implementation

h.swpai Store Word and Post Address Increment by 4 h.swpai

15	8	7	4	3	0
opcode 0x63		N		B	
8 bits		4 bits		4 bits	

Format:

h.swpai N,rB

Description:

The content of rA represents effective address. The word in general register rB is stored to memory addressed by EA. Besides, the content of rA is incremented by one word

Operation:

EA <- rA
(EA)[31:0] <- rB
rA <- rA + 4

Notes:

Class 2:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Recommended

h.shpai Store Half Word and Post Address Increment by 2 h.shpai

15	8	7	4	3	0
opcode 0x64		N		B	
8 bits		4 bits		4 bits	

Format:

h.shpai N,rB

Description:

The content of rA represents effective address. The low-order 16 bits of general register rB are stored to memory addressed by EA. Besides, the content of rA is incremented by two bytes

Operation:

EA <- rA
(EA)[15:0] <- rB[15:0]
rA <- rA + 2

Notes:

Class 1:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Mandatory always

h.sbpai Store Byte and Post Address Increment by 1 h.sbpai

15	8	7	4	3	0
opcode 0x65		N		B	
8 bits		4 bits		4 bits	

Format:

h.sbpai N,rB

Description:

The content of rA represents effective address. The low-order 8 bits of general register rB are stored to memory addressed by EA. Besides, the content of rA is incremented by one byte

Operation:

EA <- rA
(EA)[7:0] <- rB[7:0]
rA <- rA + 1

Notes:

Class 1:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Mandatory always

h.swpain Store Byte and Post Address Increment by signed iMmediate h.s

15	8	7	4	3	0
opcode 0x66		N		B	
8 bits		4 bits		4 bits	

Format:

h.swpain N,rB

Description:

The content of rA represents effective address. The low-order 8 bits of general register rB are stored to memory addressed by EA. Besides, the content of rA is incremented by one byte

Operation:

EA <- PAIR
(EA)[7:0] <- rB[7:0]
PAIR <- PAIR + N * 1

Notes:

Class 1:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Mandatory always

h.shpain Store Half Word and Post Address Increment by signed iMmedia

15	8	7	4	3	0
opcode 0x67		N		B	
8 bits		4 bits		4 bits	

Format:

h.shpain N,rB

Description:

The content of rA represents effective address. The low-order 16 bits of general register rB are stored to memory addressed by EA. Besides, the content of rA is incremented by two bytes

Operation:

EA <- PAIR
(EA)[15:0] <- rB[15:0]
PAIR <- PAIR + N * 2

Notes:

Class 1:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Mandatory always

h.sbpain Store Word and Post Address Increment by signed iMmediate h.

15	8	7	4	3	0
opcode 0x68		N		B	
8 bits		4 bits		4 bits	

Format:

h.sbpain N,rB

Description:

PAIR represents effective address. The word in general register rB is stored to memory addressed by EA. Besides, the content of rA is incremented by one word

Operation:

EA <- PAIR
 (EA)[31:0] <- rB
 PAIR <- PAIR + N * 4

Notes:

Class 2:	Architecture Level	Execution Mode	Implementation
	Core CPU	User and Supervisor	Recommended

h.mtsr Move To Special Register h.mtsr

15	8	7	4	3	0
opcode 0xfe		A		B	
8 bits		4 bits		4 bits	

Format:

h.mtsr rA,rB

Description:

The contents of general register rB are moved into special register indexed by rA.

Operation:

rA(rS) <- rB

Notes:

Class 4:	Architecture Level	Execution Mode	Implementation
	System Management	Supervisor only	Mandatory always

h.mfsr Move From Special Register h.mfsr

15	8	7	4	3	0
opcode 0xff		A		B	
8 bits		4 bits		4 bits	

Format:

h.mfsr rA,rB

Description:

The contents of special register rB are moved into general register rA.

Operation:

rA <- rB(rS)

Notes:

Class 4:	Architecture Level	Execution Mode	Implementation
	System Management	Supervisor only	Mandatory always