Draft, Do not distribute

l.illegal Illegal instruction l.illegal

31	0
opcode 0x0	
32 bits	

Format:

```
l.illegal
```

Description:

The result of this instruction is always an illegal instruction exception.

Operation:

PC <- address of illegal instruction exception handler

Notes:

Class 1:

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

l.j Jump l.j

31 26	25 0
opcode 0x0	X
6 bits	26 bits

Format:

```
l.j X
```

Description:

The immediate is shifted left two bits, sign-extended to 32 bits and then added to the address of the delay slot. The result is effective address of the jump. The program unconditionally jumps to EA with a delay of one 32 bit or two 16 bit instructions.

Operation:

PC <- (Immediate || 00) + DelayInsnAddr
LR <- DelayInsnAddr + 4

Notes:

Class 1:

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

l.jal Jump and Link l.jal

31 26	25 0
opcode 0x1	X
6 bits	26 bits

Format:

```
l.jal X
```

Description:

The immediate is shifted left two bits, sign-extended to 32 bits and then added to the address of the delay slot. The result is effective address of the jump. The program unconditionally jumps to EA with a delay of one 32 bit or two 16 bit instructions. The address of the instruction after the delay slot is placed in the link register.

Operation:

PC <- (Immediate || 00) + DelayInsnAddr
LR <- DelayInsnAddr + 4

Notes:

Class 1:

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

l.bnf Branch if No Flag l.bnf

31 26	25 0
opcode 0x2	X
6 bits	26 bits

Format:

```
l.bnf X
```

Description:

```
The immediate is shifted left two bits, sign-extended to 32 bits and then added
to the address of the delay slot.  The result is effective address of the branch.
If the compare flag is cleared, then the program branches to EA with a delay
of one 32 bit or two 16 bit instructions.
```

Operation:

EA <- (Immediate || 00) + DelayInsnAddr
PC <- EA if flag cleared

Notes:

Class 1:

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

l.bf Branch if Flag l.bf

31 26	25 0
opcode 0x3	X
6 bits	26 bits

Format:

```
l.bf X
```

Description:

```
The immediate is shifted left two bits, sign-extended to 32 bits and then added
to the address of the delay slot.  The result is effective address of the branch.
If the compare flag is set, then the program branches to EA with a delay of
one 32 bit or two 16 bit instructions.
```

Operation:

EA <- (Immediate || 00) + DelayInsnAddr
PC <- EA if flag set

Notes:

Class 1:

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

l.bfnez l.bfnez

31 26	25 0
opcode 0x3	X
6 bits	26 bits

Format:

```
l.bfnez X
```

Description:

Operation:

Notes:

Class 0:	Architecture Level	Execution Mode	Implementation

l.bfeqz l.bfeqz

31 26	25 0
opcode 0x2	X
6 bits	26 bits

Format:

```
l.bfeqz X
```

Description:

Operation:

Notes:

Class 0:	Architecture Level	Execution Mode	Implementation

l.jmp l.jmp

31 26	25 0
opcode 0x0	X
6 bits	26 bits

Format:

```
l.jmp X
```

Description:

Operation:

Notes:

Class 0:	Architecture Level	Execution Mode	Implementation

l.load32u Load Word and Extend with Zero l.load32u

31 25	24	23 20	19 16	15 0
opcode 0x8	J	A	B	J
7 bits	1 bits	4 bits	4 bits	16 bits

Format:

```
l.load32u rA,J(rB)
```

Description:

```
Offset is sign-extended 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) + rB
rA <- (EA)[31:0]

Notes:

Class 1:

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

l.load16u Load Half Word and Extend with Zero l.load16u

31 25	24	23 20	19 16	15 0
opcode 0x9	J	A	B	J
7 bits	1 bits	4 bits	4 bits	16 bits

Format:

```
l.load16u rA,J(rB)
```

Description:

Offset is sign-extended 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) + 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

l.load16s Load Half Word and Extend with Sign l.load16s

31 25	24	23 20	19 16	15 0
opcode 0xa	J	A	B	J
7 bits	1 bits	4 bits	4 bits	16 bits

Format:

```
l.load16s rA,J(rB)
```

Description:

```
Offset is sign-extended 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 bit 15 of the loaded value.
```

Operation:

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

Notes:

Class 1:

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

l.load8u Load Byte and Extend with Zero l.load8u

31 25	24	23 20	19 16	15 0
opcode 0xb	J	A	B	J
7 bits	1 bits	4 bits	4 bits	16 bits

Format:

```
l.load8u rA,J(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

l.load8s Load Byte and Extend with Sign l.load8s

31 25	24	23 20	19 16	15 0
opcode 0xc	J	A	B	J
7 bits	1 bits	4 bits	4 bits	16 bits

Format:

```
l.load8s rA,J(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 bit 7 of the loaded value.
```

Operation:

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

Notes:

Class 1:

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

l.stor32 Store Word l.stor32

31 25	24	23 20	19 16	15 0
opcode 0xd	J	A	B	J
7 bits	1 bits	4 bits	4 bits	16 bits

Format:

```
l.stor32 J(rA),rB
```

Description:

Offset is sign-extended 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) + rA
(EA)[31:0] <- rB

Notes:

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

l.stor16 Store Half Word l.stor16

31 25	24	23 20	19 16	15 0
opcode 0xe	J	A	B	J
7 bits	1 bits	4 bits	4 bits	16 bits

Format:

```
l.stor16 J(rA),rB
```

Description:

Offset is sign-extended 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) + rA
(EA)[15:0] <- rB[15:0]

Notes:

Class 1:

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

l.stor8 Store Byte l.stor8

31 25	24	23 20	19 16	15 0
opcode 0xf	J	A	B	J
7 bits	1 bits	4 bits	4 bits	16 bits

Format:

```
l.stor8 J(rA),rB
```

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

l.addi32s Add Immediate Signed l.addi32s

31 26	25 24	23 20	19 16	15 0
opcode 0x8	K	A	B	K
6 bits	2 bits	4 bits	4 bits	16 bits

Format:

```
l.addi32s rA,rB,K
```

Description:

```
Immediate is signed-extended and added to the contents of general register
rB to form the result.  The result is placed into general register rA.
```

Operation:

rA <- rB + exts(Immediate)

Notes:

Class 1:

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

l.subi32s Subtract Immediate Signed l.subi32s

31 26	25 24	23 20	19 16	15 0
opcode 0x9	K	A	B	K
6 bits	2 bits	4 bits	4 bits	16 bits

Format:

```
l.subi32s rA,rB,K
```

Description:

```
Immediate is signed-extended and subtracted from the contents of general register
rB to form the result.  The result is placed into general register rA.
```

Operation:

rA <- rB - exts(Immediate)

Notes:

Class 1:

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

l.muli32s Multiply Immediate Signed l.muli32s

31 24	23 20	19 16	15 0
opcode 0x28	A	B	I
8 bits	4 bits	4 bits	16 bits

Format:

```
l.muli32s rA,rB,I
```

Description:

```
Immediate and the contents of general register rB are multiplied and the result
is truncated to 32 bits and placed into general register rA.
```

Operation:

rA <- rB * Immediate

Notes:

Class 2:

Architecture Level	Execution Mode	Implementation
Core CPU	User and Supervisor	Recommended

l.xori16 Exclusive Or Immediate Half Word l.xori16

31 24	23 20	19 16	15 0
opcode 0x29	A	B	I
8 bits	4 bits	4 bits	16 bits

Format:

```
l.xori16 rA,rB,I
```

Description:

```
Immediate is zero-extended and 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 <- rB XOR exts(Immediate)

Notes:

Class 3:

Architecture Level	Execution Mode	Implementation
Core CPU	User and Supervisor	Optional

l.immlo16u Immediate Low-Order Half Word Unsigned l.immlo16u

31 24	23 20	19 16	15 0
opcode 0x2a	A	reserved	I
8 bits	4 bits	4 bits	16 bits

Format:

```
l.immlo16u rA,lo(I)
```

Description:

```
16 bit immediate is placed into low-order 16 bits of general register rA.
```

Operation:

rA[15:0] <- Immediate

Notes:

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

l.immhi16u Immediate High-Order Half Word Unsigned l.immhi16u

31 24	23 20	19 16	15 0
opcode 0x2b	A	reserved	I
8 bits	4 bits	4 bits	16 bits

Format:

```
l.immhi16u rA,hi(I)
```

Description:

```
16 bit immediate is placed into high-order 16 bits of general register rA.
```

Operation:

rA[31:16] <- Immediate

Notes:

Class 1:

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

l.sub32s Subtract Signed l.sub32s

31 24	23 20	19 16	15 12	11 8	7 0
opcode 0x2c	A	B	C	opcode 0x0	reserved
8 bits	4 bits	4 bits	4 bits	4 bits	8 bits

Format:

```
l.sub32s rA,rB,rC
```

Description:

```
The contents of general register rC is subtracted from 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

l.shla32 Shift Left Arithmetic l.shla32

31 24	23 20	19 16	15 12	11 8	7 3	2 0
opcode 0x2c	A	B	C	opcode 0x1	L	reserved
8 bits	4 bits	4 bits	4 bits	4 bits	5 bits	3 bits

Format:

```
l.shla32 rA,rB,rC,L
```

Description:

```
Immediate is combined with low-order 5 bits of general register rC in a bit-wise
logical OR operation.  The result specifies the number of bit positions the
contents of general register rB are shifted left, inserting zeros into the
low-order bits.
```

Operation:

b <- Immediate | rC
rA[31:b] <- rB[31-b:0]
rA[b:0] <- 0

Notes:

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

l.shra32 Shift Right Arithmetic l.shra32

31 24	23 20	19 16	15 12	11 8	7 3	2 0
opcode 0x2c	A	B	C	opcode 0x2	L	reserved
8 bits	4 bits	4 bits	4 bits	4 bits	5 bits	3 bits

Format:

```
l.shra32 rA,rB,rC,L
```

Description:

```
Immediate is combined with low-order 5 bits of general register rC in a bit-wise
logical OR operation.  The result specifies the number of bit positions the
contents of general register rB are shifted right, sign-extending the high-order
bits.
```

Operation:

b <- Immediate | rC
rA[31-b:0] <- rB[31:b]
rA[31-b:31] <- rB[31]

Notes:

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

l.shrl32 Shift Right Logical l.shrl32

31 24	23 20	19 16	15 12	11 8	7 3	2 0
opcode 0x2c	A	B	C	opcode 0x3	L	reserved
8 bits	4 bits	4 bits	4 bits	4 bits	5 bits	3 bits

Format:

```
l.shrl32 rA,rB,rC,L
```

Description:

```
Immediate is combined with low-order 5 bits of general register rC in a bit-wise
logical OR operation.  The result specifies the number of bit positions the
contents of general register rB are shifted right, inserting zeros into the
high-order bits.
```

Operation:

b <- Immediate | rC
rA[31-b:0] <- rB[31:b]
rA[31-b:31] <- 0

Notes:

Class 1:

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

l.and32 And l.and32

31 ... 24	23 ... 20	19 ... 16	15 ... 12	11 ... 8	7 ... 0
opcode 0x2c	A	B	C	opcode 0x4	reserved
8 bits	4 bits	4 bits	4 bits	4 bits	8 bits

Format:

```
l.and32 rA,rB,rC
```

Description:

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

Operation:

rA <- rB AND rC

Notes:

Class 1:

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

l.or32 Or l.or32

31 24	23 20	19 16	15 12	11 8	7 0
opcode 0x2c	A	B	C	opcode 0x5	reserved
8 bits	4 bits	4 bits	4 bits	4 bits	8 bits

Format:

```
l.or32 rA,rB,rC
```

Description:

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

Operation:

rA <- rB OR rC

Notes:

Class 1:

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

l.xor32 Exclusive Or l.xor32

31 24	23 20	19 16	15 12	11 8	7 0
opcode 0x2c	A	B	C	opcode 0x6	reserved
8 bits	4 bits	4 bits	4 bits	4 bits	8 bits

Format:

```
l.xor32 rA,rB,rC
```

Description:

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

Operation:

rA <- rB XOR rC

Notes:

Class 1:

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

l.mul32s Multiply Signed l.mul32s

31 24	23 20	19 16	15 12	11 8	7 0
opcode 0x2c	A	B	C	opcode 0x7	reserved
8 bits	4 bits	4 bits	4 bits	4 bits	8 bits

Format:

```
l.mul32s rA,rB,rC
```

Description:

The contents of general register rB and the contents of general register rC are multiplied and the result is truncated to 32 bits and placed into general register rA. 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

l.mul32u Multiply Unsigned l.mul32u

31 24	23 20	19 16	15 12	11 8	7 0
opcode 0x2c	A	B	C	opcode 0x8	reserved
8 bits	4 bits	4 bits	4 bits	4 bits	8 bits

Format:

```
l.mul32u rA,rB,rC
```

Description:

```
The contents of general register rB and the contents of general register rC
are multiplied and the result is truncated to 32 bits and placed into general
register rA. 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

l.div32s Divide Signed l.div32s

31 ... 24	23 ... 20	19 ... 16	15 ... 12	11 ... 8	7 ... 0
opcode 0x2c	A	B	C	opcode 0x9	reserved
8 bits	4 bits	4 bits	4 bits	4 bits	8 bits

Format:

```
l.div32s rA,rB,rC
```

Description:

```
The contents of general register rB are divided by the contents of general
register rC 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 <- rB / rC

Notes:

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

l.div32u Divide Unsigned l.div32u

31 24	23 20	19 16	15 12	11 8	7 0
opcode 0x2c	A	B	C	opcode 0xa	reserved
8 bits	4 bits	4 bits	4 bits	4 bits	8 bits

Format:

```
l.div32u rA,rB,rC
```

Description:

The contents of general register rB are divided by the contents of general register rC 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 <- rB / rC

Notes:

Class 3:

Architecture Level	Execution Mode	Implementation
Core CPU	User and Supervisor	Optional

l.dcbf Data Cache Block Flush l.dcbf

31 24	23 20	19 12	11 8	7 0
opcode 0x30	A	I	opcode 0x0	I
8 bits	4 bits	8 bits	4 bits	8 bits

Format:

```
l.dcbf J(rA)
```

Description:

```
TBD
```

Operation:

Notes:

Class 5:	Architecture Level	Execution Mode	Implementation
	Cache Management	Supervisor only	Mandatory if cache supported

l.dcbt Data Cache Block Touch l.dcbt

31 24	23 20	19 12	11 8	7 0
opcode 0x30	A	I	opcode 0x1	I
8 bits	4 bits	8 bits	4 bits	8 bits

Format:

```
l.dcbt J(rA)
```

Description:

TBD

Operation:

Notes:

Class 5:	Architecture Level	Execution Mode	Implementation
	Cache Management	Supervisor only	Mandatory if cache supported

l.dcbi Data Cache Block Invalidate l.dcbi

31 ... 24	23 ... 20	19 ... 12	11 ... 8	7 ... 0
opcode 0x30	A	I	opcode 0x2	I
8 bits	4 bits	8 bits	4 bits	8 bits

Format:

```
l.dcbi J(rA)
```

Description:

```
TBD
```

Operation:

Notes:

Class 5:	Architecture Level	Execution Mode	Implementation
	Cache Management	Supervisor only	Mandatory if cache supported

l.dcia Data Cache Invalidate All l.dcia

31 24	23 20	19 12	11 8	7 0
opcode 0x30	A	reserved	opcode 0x3	reserved
8 bits	4 bits	8 bits	4 bits	8 bits

Format:

```
l.dcia
```

Description:

```
TBD
```

Operation:

Notes:

Class 5:	Architecture Level	Execution Mode	Implementation
	Cache Management	Supervisor only	Mandatory if cache supported

l.dcfa Data Cache Flush All l.dcfa

31 24	23 20	19 12	11 8	7 0
opcode 0x30	A	reserved	opcode 0x4	reserved
8 bits	4 bits	8 bits	4 bits	8 bits

Format:

```
l.dcfa
```

Description:

```
TBD
```

Operation:

Notes:

Class 5:

Architecture Level	Execution Mode	Implementation
Cache Management	Supervisor only	Mandatory if cache supported

l.tlbia TLB Invalidate All l.tlbia

31 24	23 20	19 12	11 8	7 0
opcode 0x30	A	reserved	opcode 0x5	reserved
8 bits	4 bits	8 bits	4 bits	8 bits

Format:

```
l.tlbia
```

Description:

```
TBD
```

Operation:

Notes:

Class 6:	Architecture Level	Execution Mode	Implementation
	Virtual Memory	Supervisor only	Mandatory if MMU supported

l.mtsr Move To Special Register l.mtsr

31 24	23 20	19 12	11 8	7 0
opcode 0x30	A	S	opcode 0x6	S
8 bits	4 bits	8 bits	4 bits	8 bits

Format:

```
l.mtsr rS,rA
```

Description:

```
The contents of general register rA are moved into special register rS.
```

Operation:

rS <- rA

Notes:

Class 4:

Architecture Level	Execution Mode	Implementation
System Management	Supervisor only	Mandatory always

l.mfsr Move From Special Register l.mfsr

31 24	23 20	19 12	11 8	7 0
opcode 0x30	A	S	opcode 0x7	S
8 bits	4 bits	8 bits	4 bits	8 bits

Format:

```
l.mfsr rA,rS
```

Description:

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

Operation:

rA <- rS

Notes:

Class 4:

Architecture Level	Execution Mode	Implementation
System Management	Supervisor only	Mandatory always

h.sfeq32 Set Flag if Equal h.sfeq32

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

Format:

```
h.sfeq32 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.sfne32 Set Flag if Not Equal h.sfne32

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

Format:

```
h.sfne32 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.sfgt32s Set Flag if Greater Than Signed h.sfgt32s

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

Format:

```
h.sfgt32s 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.sfge32s Set Flag if Greater or Equal Than Signed h.sfge32s

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

Format:

```
h.sfge32s 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.sflt32s Set Flag if Less Than Signed h.sflt32s

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

Format:

```
h.sflt32s 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.sfle32s Set Flag if Less or Equal Than Signed h.sfle32s

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

Format:

```
h.sfle32s 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.sfgt32u Set Flag if Greater Than Unsigned h.sfgt32u

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

Format:

```
h.sfgt32u 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.sfge32u Set Flag if Greater or Equal Than Unsigned h.sfge32u

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

Format:

```
h.sfge32u 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.sflt32u Set Flag if Less Than Unsigned h.sflt32u

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

Format:

```
h.sflt32u 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.sfle32u Set Flag if Less or Equal Than Unsigned h.sfle32u

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

Format:

```
h.sfle32u 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.mov32 Move h.mov32

15 8	7 4	3 0
opcode 0x4a	A	B
8 bits	4 bits	4 bits

Format:

```
h.mov32 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.ext16s Extend Half Word with Sign h.ext16s

15 8	7 4	3	2 0
opcode 0x4b	A	reserved	opcode 0x0
8 bits	4 bits	1 bits	3 bits

Format:

```
h.ext16s 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] <- rA[15]

Notes:

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

h.ext16z Extend Half Word with Zero h.ext16z

15 8	7 4	3	2 0
opcode 0x4b	A	reserved	opcode 0x1
8 bits	4 bits	1 bits	3 bits

Format:

```
h.ext16z 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.ext8s Extend Byte with Sign h.ext8s

15 8	7 4	3	2 0
opcode 0x4b	A	reserved	opcode 0x2
8 bits	4 bits	1 bits	3 bits

Format:

```
h.ext8s 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.ext8z Extend Byte with Zero h.ext8z

15 8	7 4	3	2 0
opcode 0x4b	A	reserved	opcode 0x3
8 bits	4 bits	1 bits	3 bits

Format:

```
h.ext8z 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 8	7 3	2 0
opcode 0x4b	reserved	opcode 0x4
8 bits	5 bits	3 bits

Format:

```
h.nop
```

Description:

```
This instruction does not do anything except it takes at least one clock cycle
to complete.  It is usually used to fill gaps between 16 bit and 32 bit instructions.
```

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	2 0
opcode 0x4b	A	reserved	opcode 0x5
8 bits	4 bits	1 bits	3 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 a delay of one 32 bit or two 16 bit instructions. The address of the instruction after the delay slot is placed in the link register.

Operation:

PC <- rA
LR <- DelayInsnAddr + 4

Notes:

Class 1:

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

h.load32u Load Word and Extend with Zero h.load32u

15 12	11 8	7 4	3 0
opcode 0x5	N	A	B
4 bits	4 bits	4 bits	4 bits

Format:

```
h.load32u rA,N(rB)
```

Description:

```
Offset is sign-extended 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) + rB
rA <- (EA)[31:0]

Notes:

Class 2:

Architecture Level	Execution Mode	Implementation
Core CPU	User and Supervisor	Recommended

h.stor32 Store Word h.stor32

15 12	11 8	7 4	3 0
opcode 0x6	N	A	B
4 bits	4 bits	4 bits	4 bits

Format:

```
h.stor32 N(rA),rB
```

Description:

Offset is sign-extended 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) + rA
(EA)[31:0] <- rB

Notes:

Class 2:

Architecture Level	Execution Mode	Implementation
Core CPU	User and Supervisor	Recommended

h.add32s Add Signed h.add32s

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

Format:

```
h.add32s 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.immch32s Immediate Byte Signed h.immch32s

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

Format:

```
h.immch32s 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.jal Jump and Link h.jal

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

Format:

```
h.jal X
```

Description:

The immediate is shifted left two bits, sign-extended to 32 bits and then added to the address of the delay slot. The result is effective address of the jump. The program unconditionally jumps to EA with a delay of one 32 bit or two 16 bit instructions. The address of the instruction after the delay slot is placed in the link register.

Operation:

PC <- (Immediate || 00) + DelayInsnAddr
LR <- DelayInsnAddr + 4

Notes:

Class 2:

Architecture Level	Execution Mode	Implementation
Core CPU	User and Supervisor	Recommended

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 is shifted left two bits, sign-extended to 32 bits and then added
to the address of the delay slot.  The result is effective address of the branch.
If the compare flag is cleared, then the program branches to EA with a delay
of one 32 bit or two 16 bit instructions.
```

Operation:

EA <- (Immediate || 00) + DelayInsnAddr
PC <- EA if flag cleared

Notes:

Class 2:

Architecture Level	Execution Mode	Implementation
Core CPU	User and Supervisor	Recommended

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 is shifted left two bits, sign-extended to 32 bits and then added
to the address of the delay slot.  The result is effective address of the branch.
If the compare flag is set, then the program branches to EA with a delay of
one 32 bit or two 16 bit instructions.
```

Operation:

EA <- (Immediate || 00) + DelayInsnAddr
PC <- EA if flag set

Notes:

Class 2:

Architecture Level	Execution Mode	Implementation
Core CPU	User and Supervisor	Recommended

h.movi32 Move 32 bit Immediate h.movi32

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

Format:

```
h.movi32 rA,M
```

Description:

```
For simulator.  Obsolete
```

Operation:

N/A

Notes:

Class 0:	Architecture Level	Execution Mode	Implementation

simprintf Simulate printf simprintf

31 0
opcode 0xe001
16 bits

Format:

```
simprintf
```

Description:

```
For simulator.  Obsolete.
```

Operation:

N/A

Notes:

Class 0:

Architecture Level	Execution Mode	Implementation

simrdtsc Simulate Read Timer simrdtsc

31 24	23 20	19 0
opcode 0xe0	A	opcode 0x0
8 bits	4 bits	3 bits

Format:

```
simrdtsc rA
```

Description:

```
For simulator. Obsolete.
```

Operation:

N/A

Notes:

Class 0:

Architecture Level	Execution Mode	Implementation

h.sched Schedule h.sched

15 12	11 0
opcode 0xf	Z
4 bits	12 bits

Format:

```
h.sched Z
```

Description:

```
Immediate carries static scheduling information about instruction scheduling.
This information is generated by an optimizing compiler.
```

Operation:

Notes:

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