Zcee 0.53.1

This document is in the Stable state. Assume anything could still change, but limited change should be expected. For more information see: https://riscv.org/spec-state

The instructions in Zcee are all very simple and are 16-bit encodings of existing instructions, either from the I/E ISA or from Zbb.

All proposed encodings are currently reserved for all architectures, and have no conflicts with any existing extensions.

All of the sign and zero extension instructions use the same format. These typically benefit code-size because if the compiler is working with fewer than XLEN bits it must sign/zero extend the value to XLEN bits before calling a function, or passing a return value.

{reg:[{ bits: 2, name: 0x1, attr: ['OP=C1'] }, { bits: 3, name: 'opcode' }, { bits: 2, name: 0x0, attr: ['FUNCT2'] }, { bits: 3, name: 'rsd\'', attr: ['SRCDST'] }, { bits: 6, name: 0x23, attr: ['FUNCT6'] },],config:{bits:16}}

```
The _c.mul_ encoding uses the CR register format along with other instructions
such as _c.sub_, _c.xor_ etc.

[NOTE]

  _c.sext.w_ is a pseudo-instruction for _c.addiw rd, 0_ (RV64)

[%header,cols="^1,^1,4,8"]
|===
|RV 32
|RV 64
|Mnemonic
|Instruction

|&#10003;
|&#10003;
|c.sext.b _rsd'_
|<<#insns-c_sext_b>>

|&#10003;
|&#10003;
|c.sext.h _rsd'_
|<<#insns-c_sext_h>>

|&#10003;
|&#10003;
|c.zext.b _rsd'_
|<<#insns-c_zext_b>>

|&#10003;
```

```
|&#10003;
|c.zext.h _rsd'_
|<<#insns-c_zext_h>>

|
|&#10003;
|c.zext.w _rsd'_
|<<#insns-c_zext_w>>

|&#10003;
|&#10003;
|c.mul _rsd'_, _rs2'_
|<<#insns-c_mul>>

|===

<<<
[#insns-c_sext_b,reftext="Sign extend byte, 16-bit encoding"]
=== c.sext.b

Synopsis::
Sign extend byte, 16-bit encoding

Mnemonic::
c.sext.b _rsd'_

Encoding (RV32, RV64)::
[wavedrom, , svg]
```

{reg:[{ bits: 2, name: 0x1, attr: ['OP=C1'] }, { bits: 3, name: 0x1, attr: ['C.SEXT.B'] }, { bits: 2, name: 0x0, attr: ['FUNCT2'] }, { bits: 3, name: 'rsd\'', attr: ['SRCDST'] }, { bits: 6, name: 0x27, attr: ['FUNCT6'] },],config:{bits:16}}

```
Description::
This instruction takes a single source/destination operand, from the 8-register
set x8-x15. It sign-extends the least-significant byte in the operand to XLEN by
copying the most-significant bit
in the byte (i.e., bit 7) to all of the more-significant bits.


Prerequisites::
The C-extension must also be configured.

32-bit equivalent::
<<insns-sext_b>> from Zbb

Operation::
[source,sail]
--
X(rsdc) = EXTS(X(rsdc)[7..0]);
--


Included in::
[%header,cols="4,2,2"]
|===
|Extension
|Minimum version
|Lifecycle state

|Zcee (<<Zcee>>)
|0.53.1
|Stable
|===
<<<
[#insns-c_sext_h,reftext="Sign extend halfword, 16-bit encoding"]
=== c.sext.h

Synopsis::
Sign extend halfword, 16-bit encoding

Mnemonic::
c.sext.h _rsd'_

Encoding (RV32, RV64)::
[wavedrom, , svg]
```

{reg:[{ bits: 2, name: 0x1, attr: ['OP=C1'] }, { bits: 3, name: 0x3, attr: ['C.SEXT.H'] }, { bits: 2, name: 0x0, attr: ['FUNCT2'] }, { bits: 3, name: 'rsd\'', attr: ['SRCDST'] }, { bits: 6, name: 0x27, attr: ['FUNCT6'] },

],config:{bits:16}}

```
Description::
This instruction takes a single source/destination operand, from the 8-register
set x8-x15. It sign-extends the least-significant halfword in the operand to XLEN
by copying the most-significant bit
in the halfword (i.e., bit 15) to all of the more-significant bits.


Prerequisites::
The C-extension must also be configured.

32-bit equivalent::
<<insns-sext_h>> from Zbb

Operation::
[source,sail]
--
X(rsdc) = EXTS(X(rsdc)[15..0]);
--


Included in::
[%header,cols="4,2,2"]
|===
|Extension
|Minimum version
|Lifecycle state

|Zcee (<<Zcee>>)
|0.53.1
|Stable
|===

<<<
[#insns-c_zext_b,reftext="Zero extend byte, 16-bit encoding"]
=== c.zext.b

Synopsis::
Zero extend byte, 16-bit encoding

Mnemonic::
c.zext.b _rsd'_

Encoding (RV32, RV64)::
[wavedrom, , svg]
```

{reg:[{ bits: 2, name: 0x1, attr: ['OP=C1'] }, { bits: 3, name: 0x0, attr: ['C.ZEXT.B'] }, { bits: 2, name: 0x0, attr: ['FUNCT2'] }, { bits: 3, name: 'rsd\'', attr: ['SRCDST'] }, { bits: 6, name: 0x27, attr: ['FUNCT6'] },],config:{bits:16}}

```
Description::
This instruction takes a single source/destination operand, from the 8-register
set x8-x15. It zero-extends the least-significant byte of the operand to XLEN by
inserting 0's into all of
the bits more significant than 7.

Prerequisites::
The C-extension must also be configured.

32-bit equivalent::
[source,sail]
--
andi rd, rs1, 0xff
--

Operation::
[source,sail]
--
X(rsdc) = EXTZ(X(rsdc)[7..0]);
--


Included in::
[%header,cols="4,2,2"]
|===
|Extension
|Minimum version
|Lifecycle state

|Zcee (<<Zcee>>)
|0.53.1
|Stable
|===

<<<
[#insns-c_zext_h,reftext="Zero extend halfword, 16-bit encoding"]
=== c.zext.h

Synopsis::
Zero extend halfword, 16-bit encoding

Mnemonic::
c.zext.h _rsd'_

Encoding (RV32, RV64)::
[wavedrom, , svg]
```

{reg:[{ bits: 2, name: 0x1, attr: ['OP=C1'] }, { bits: 3, name: 0x2, attr: ['C.ZEXT.H'] }, { bits: 2, name: 0x0, attr: ['FUNCT2'] }, { bits: 3, name: 'rsd\'', attr: ['SRCDST'] }, { bits: 6, name: 0x27, attr: ['FUNCT6'] },],config:{bits:16}}

```
Description::
This instruction takes a single source/destination operand, from the 8-register
set x8-x15. It zero-extends the least-significant halfword of the operand to XLEN
by inserting 0's into all of
the bits more significant than 15.


Prerequisites::
The C-extension must also be configured.

32-bit equivalent::
<<insns-zext_h>> from Zbb

Operation::
[source,sail]
--
X(rsdc) = EXTZ(X(rsdc)[15..0]);
--


Included in::
[%header,cols="4,2,2"]
|===
|Extension
|Minimum version
|Lifecycle state

|Zcee (<<Zcee>>)
|0.53.1
|Stable
|===

<<<
[#insns-c_zext_w,reftext="Zero extend word, 16-bit encoding"]
=== c.zext.w

Synopsis::
Zero extend word, 16-bit encoding

Mnemonic::
c.zext.w _rsd'_

Encoding (RV64)::
[wavedrom, , svg]
```

{reg:[{ bits: 2, name: 0x1, attr: ['OP=C1'] }, { bits: 3, name: 0x4, attr: ['C.ZEXT.W'] }, { bits: 2, name: 0x0,

attr: ['FUNCT2'] }, { bits: 3, name: 'rsd\'', attr: ['SRCDST'] }, { bits: 6, name: 0x27, attr: ['FUNCT6'] },],config:{bits:16}}

```
Description::
This instruction takes a single source/destination operand, from the 8-register
set x8-x15. It zero-extends the least-significant word of the operand to XLEN by
inserting 0's into all of
the bits more significant than 31.


Prerequisites::
The C-extension must also be configured.

32-bit equivalent::
[source,sail]
--
add.uw rd, rs1, zero
--

Operation::
[source,sail]
--
X(rsdc) = EXTZ(X(rsdc)[31..0]);
--


Included in::
[%header,cols="4,2,2"]
|===
|Extension
|Minimum version
|Lifecycle state

|Zcee (<<Zcee>>)
|0.53.1
|Stable
|===
<<<
[#insns-c_mul,reftext="Multiply, 16-bit encoding"]
=== c.mul

Synopsis::
Multiply, 16-bit encoding

Mnemonic::
c.mul _rsd'_, _rs2'_

Encoding (RV32, RV64)::
[wavedrom, , svg]
```

{reg:[{ bits: 2, name: 0x1, attr: ['OP=C1'] }, { bits: 3, name: 'rs2\'', attr: ['SRC2'] }, { bits: 2, name: 0x2, attr: ['FUNCT2'] }, { bits: 3, name: 'rsd\'', attr: ['SRCDST'] }, { bits: 6, name: 0x27, attr: ['FUNCT6'] },],config:{bits:16}}

```
Description::
This instruction multiplies XLEN bits of the source operands from _rsd'_ and
_rs2'_ and writes the lowest XLEN bits of the result to _rsd'_. Both operands are
from the 8-register set x8-x15.

Prerequisites::
The C-extension and either M or Zmmul must also be configured.

32-bit equivalent::
<<insns-mul>>

Operation::
[source,sail]
--
let result_wide = to_bits(2 * sizeof(xlen), signed(X(rsdc)) * signed(X(rs2c)));
X(rsdc) = result_wide[(sizeof(xlen) - 1) .. 0];
--


Included in::
[%header,cols="4,2,2"]
|===
|Extension
|Minimum version
|Lifecycle state

|Zcee (<<Zcee>>)
|0.53.1
|Stable
|===
```