1 CGetLinear

The low bit of integer register *rd* is set to the **linear** field of *cs1*. All other bits of *rd* are cleared.

```
function clause execute (CGetLinear(rd, cs1)) = {
  let capVal = C(cs1);
  X(rd) = EXTZ(getCapLinearity(capVal));
  RETIRE_SUCCESS
}
```

2 CMakeLinear

Capability register *cd* is replaced with capability register *cs1* with the **linear** field set to 1.

Exceptions

An exception is raised if:

- *cs1*.**tag** is not set.
- *cs1* is sealed.
- *cs1*.**linear** is set and *cs1* $\neq$ *cd*.

```
function clause execute (CMakeLinear(cd, cs1)) = {
  let cs1_val = C(cs1);
  if not (cs1_val.tag) then {
    handle_cheri_reg_exception(CapEx_TagViolation, cs1);
    RETIRE_FAIL
  } else if isCapSealed(cs1_val) then {
    handle_cheri_reg_exception(CapEx_SealViolation, cs1);
    RETIRE_FAIL
  } else if cs1_val.linear & (cd != cs1) then {
    handle_cheri_reg_exception(CapEx_LinearityViolation, cs1);
    RETIRE_FAIL
  } else {
    C(cd) = setCapLinearity(cs1_val, truncate(0b1, cap_linear_width));
    RETIRE_SUCCESS
  }
}
```

3 CSplitCap

Capability register *cd* is replaced with capability register *cs1* with the top of the capability set to its **base** + *rs2*. Capability register *cs1* is replaced with the capability that is currently in it with its **base** set to the previous **base** + *rs2*. The **offset** of both capabilities is set to 0.

Exceptions

An exception is raised if:

- *cs1*.**tag** is not set.
- *cs1* is sealed and not a borrowed capability.
- *cs1*.**base** + *rs2* + *rs2* > *cs1*.**top**.

```
function clause execute(CSplitCap(cd, cs1, rs2)) = {
  let cs1_val = C(cs1);
  let rs2_val = X(rs2);
  let (base, top) = getCapBoundsBits(cs1_val);
  let newTop : CapLenBits = EXTZ(base) + EXTZ(rs2_val);
  let newBase : CapAddrBits = EXTZ(base) + EXTZ(rs2_val);
  if not (cs1_val.tag) then {
    handle_cheri_reg_exception(CapEx_TagViolation, cs1);
    RETIRE_FAIL
  } else if signed(cs1_val.otype) != otype_unsealed & not(isCapBorrowed(cs1_val)) then {
    handle_cheri_reg_exception(CapEx_TypeViolation, cs1);
    RETIRE_FAIL
  } else if not(inCapBounds(cs1_val, base, unsigned(rs2_val) + 1)) then {
    handle_cheri_reg_exception(CapEx_LengthViolation, cs1);
    RETIRE_FAIL
  } else {
    let (_, lowCap) = setCapBounds(cs1_val, base, newTop);
    let (_, highCap) = setCapBounds(cs1_val, newBase, top);
    C(cd) = lowCap;
    C(cs1) = highCap;
    RETIRE_SUCCESS
  }
}
```

4 CMergeCap

Capability register *cd* is replaced with capability register *cs1* with its top set to the top of *cs2* and its **offset** to 0. The tag on *cs2* is unset.

Exceptions

An exception is raised if:

- *cs1*.**tag** is not set.
- *cs1* is sealed and not a borrowed capability.
- *cs2*.**tag** is not set.
- *cs2* is sealed and not a borrowed capability.
- *cs1*.**top** + 1 $\neq$ *cs2*.**base**.
- *cd* $\neq$ *cs1*.

```
function clause execute(CMergeCap(cd, cs1, cs2)) = {
  let cs1_val = C(cs1);
  let cs2_val = C(cs2);
  let (lowBase, lowTop) = getCapBoundsBits(cs1_val);
  let (highBase, highTop) = getCapBoundsBits(cs2_val);
  if not (cs1_val.tag) then {
    handle_cheri_reg_exception(CapEx_TagViolation, cs1);
    RETIRE_FAIL
  } else if signed(cs1_val.otype) != otype_unsealed & not(isCapBorrowed(cs1_val)) then {
    handle_cheri_reg_exception(CapEx_TypeViolation, cs1);
    RETIRE_FAIL
  } else if not (cs2_val.tag) then {
    handle_cheri_reg_exception(CapEx_TagViolation, cs1);
    RETIRE_FAIL
  } else if signed(cs2_val.otype) != otype_unsealed & not(isCapBorrowed(cs2_val)) then {
```

```
    handle_cheri_reg_exception(CapEx_TypeViolation, cs1);
    RETIRE_FAIL
  } else if (unsigned(lowTop)) != unsigned(highBase) then {
    handle_cheri_reg_exception(CapEx_LengthViolation, cs1);
    RETIRE_FAIL
  } else if cd != cs1 then {
    handle_cheri_reg_exception(CapEx_LinearityViolation, cs1);
    RETIRE_FAIL
  } else {
    let (_, newCap) = setCapBounds(cs1_val, lowBase, highTop);
    C(cs2) = invalidateCap(cs2_val);
    C(cd) = newCap;
    RETIRE_SUCCESS
  }
}
```

5 LinearLoadCapCap

Capability register *cd* is replaced with the capability located in memory at *cs1*.**address**, and if *cs1*.**perms** does not grant **Permit_Load_Capability** then *cd*.**tag** is cleared. The tag of the capability at *cs1*.**address** is cleared.

Exceptions

An exception is raised if:

- *cs1*.**tag** is not set.
- *cs1* is sealed.
- *cs1*.**perms** does not grant **Permit_Load**.
- *cs1*.**address** $<$ *cs1*.**base**.
- *cs1*.**address** + **CLEN** / 8 $>$ *cs1*.**top**.
- *cs1*.**address** is unaligned, regardless of whether the implementation supports unaligned data accesses.

```
function clause execute (LinearLoadCapCap(cd, cs1)) = {
  let cs1_val = C(cs1);
  let vaddr = cs1_val.address;
  handle_load_cap_via_cap(cd, 0b0 @ cs1, cs1_val, vaddr, true)
}
```

6 LinearStoreCapCap

The capability located in memory at *cs1*.**address** is replaced with capability register *cs2*. The tag of *cs2* is cleared.

Exceptions

An exception is raised if:

- *cs1*.**tag** is not set.
- *cs1* is sealed.

- *cs1*.**perms** does not grant **Permit_Store**.
- *cs1*.**perms** does not grant **Permit_Store_Capability** and *cs2*.**tag** is set.
- *cs1*.**perms** does not grant **Permit_Store_Local_Capability**, *cs2*.**tag** is set and *cs2*.**perms** does not grant **Global**.
- *cs1*.**address** $<$ *cs1*.**base**.
- *cs1*.**address** + **CLEN** $>$ *cs1*.**top**.

```
function clause execute (LinearStoreCapCap(cs2, cs1)) = {
  let cs1_val = C(cs1);
  let vaddr = cs1_val.address;
  handle_store_cap_via_cap(cs2, 0b0 @ cs1, cs1_val, vaddr, true)
}
```

7 CCreateToken

Capability register *cd* is replaced with a newly created lifetime token. If *cs1* is not **C0** the **parent id** field of the newly created token is set to the **lifetime id** field of the capability in *cs1* and the **child id** field of the capability in *cs1* is set to the **lifetime id** field of the newly created token.

Exceptions

An exception is raised if:

- the lifetime counter has reached the maximum lifetime id value.
- *cs1* is not **C0** and
  - *cs1*.**tag** is not set or
  - *cs1*.**linear** is not set or
  - *cs1*.**otype** $\neq$ **otype_lifetime** or
  - *cs1*.**child id** $\neq 0$.

```
function clause execute (CCreateToken(cd, cs1)) = {
  let ltc_val = LTC;
  let cs1_val = C(cs1);
  if unsigned(ltc_val) > cap_max_lifetime then {
    handle_cheri_cap_exception(CapEx_LifetimeOverflow, 0b000000);
    RETIRE_FAIL
  } else if cs1 != 0b00000 & not(setChildValid(cs1_val)) then {
    handle_cheri_reg_exception(CapEx_BorrowPermissionViolation, cs1);
    RETIRE_FAIL
  } else {
    parent_val = getLTLifetimeID(cs1_val);
    LTC = to_bits(cap_otype_width, (unsigned(ltc_val) + 1));
    C(cs1) = setLTChildID(cs1_val, ltc_val);
    C(cd) = constructLifetime(ltc_val, parent_val);
    RETIRE_SUCCESS
  }
}
```

8 CKillToken

Capability register *cd* is replaced with capability register *cs1* with the **linear** field set to 0.

Exceptions

An exception is raised if:

- *cs1*.**tag** is not set.
- *cs1*.**otype** $\neq$ **otype_lifetime**.
- *cs1*.**fraction** $\neq 0$.
- *cs1*.**child id** $\neq 0$.
- *cd* $\neq$ *cs1*.

```
function clause execute(CKillToken(cd, cs1)) = {
  let cs1_val = C(cs1);
  if not (cs1_val.tag) then {
    handle_cheri_reg_exception(CapEx_TagViolation, cs1);
    RETIRE_FAIL
  } else if signed(cs1_val.otype) != otype_lifetime then {
    handle_cheri_reg_exception(CapEx_TypeViolation, cs1);
    RETIRE_FAIL
  } else if cs1_val.B != zeros() then {
    handle_cheri_reg_exception(CapEx_BorrowPermissionViolation, cs1);
    RETIRE_FAIL
  } else if getLTChildID(cs1_val) != zeros() then {
    handle_cheri_reg_exception(CapEx_BorrowPermissionViolation, cs1);
    RETIRE_FAIL
  } else if cd != cs1 then {
    handle_cheri_reg_exception(CapEx_LinearityViolation, cs1);
    RETIRE_FAIL
  } else {
    C(cs1) = killToken(cs1_val);
    RETIRE_SUCCESS
  }
}
```

9 CUnlockToken

Capability register *cd* is replaced with capability register *cs1* with the **child id** field set to 0.

Exceptions

An exception is raised if:

- *cs1*.**tag** is not set.
- *cs1*.**otype** $\neq$ **otype_lifetime**.
- *cs2*.**tag** is not set.
- *cs2*.**otype** $\neq$ **otype_lifetime**.
- *cs2*.**linear** is set.
- *cs1*.**child id** $\neq$ *cs2*.**lifetime id**.
- *cd* $\neq$ *cs1*.

```
function clause execute(CUnlockToken(cd, cs1, cs2)) = {
  let cs1_val = C(cs1);
  let cs2_val = C(cs2);
  if not (cs1_val.tag) then {
    handle_cheri_reg_exception(CapEx_TagViolation, cs1);
    RETIRE_FAIL
  } else if signed(cs1_val.otype) != otype_lifetime then {
    handle_cheri_reg_exception(CapEx_TypeViolation, cs1);
    RETIRE_FAIL
  } else if not (cs2_val.tag) then {
    handle_cheri_reg_exception(CapEx_TagViolation, cs2);
    RETIRE_FAIL
  } else if signed(cs2_val.otype) != otype_lifetime then {
    handle_cheri_reg_exception(CapEx_TypeViolation, cs2);
    RETIRE_FAIL
  } else if cs2_val.linear then {
    handle_cheri_reg_exception(CapEx_BorrowPermissionViolation, cs2);
    RETIRE_FAIL
  } else if getLTChildID(cs1_val) != getLTLifetimeID(cs2_val) then {
    handle_cheri_reg_exception(CapEx_BorrowPermissionViolation, cs1);
    RETIRE_FAIL
  } else if cd != cs1 then {
    handle_cheri_reg_exception(CapEx_LinearityViolation, cs1);
    RETIRE_FAIL
  } else {
    C(cd) = setLTChildID(cs1_val, zeros(cap_otype_width));
    RETIRE_SUCCESS
  }
}
```

10 CSplitLT

Capability register *cd* is replaced with capability register *cs1* with the **fraction** field incremented by one. Capability register *cs1* is replaced with the the same capability as *cd*.

Exceptions

An exception is raised if:

- *cs1*.**tag** is not set.
- *cs1*.**otype** $\neq$ **otype_lifetime**.
- *cs1*.**linear** is not set.
- *cs1*.**fraction** has reached its maximum.

```
function clause execute(CSplitLT(cd, cs1)) = {
  let cs1_val = C(cs1);
  if not (cs1_val.tag) then {
    handle_cheri_reg_exception(CapEx_TagViolation, cs1);
    RETIRE_FAIL
  } else if signed(cs1_val.otype) != otype_lifetime then {
    handle_cheri_reg_exception(CapEx_TypeViolation, cs1);
    RETIRE_FAIL
  } else if not(cs1_val.linear) then {
    handle_cheri_reg_exception(CapEx_LinearityViolation, cs1);
    RETIRE_FAIL
  } else if unsigned(cs1_val.B) >= cap_max_B then {
```

```
    handle_cheri_reg_exception(CapEx_LifetimeOverflow, cs1);
    RETIRE_FAIL
  } else {
    let newCap = {cs1_val with B = cs1_val.B + 1};
    C(cs1) = newCap;
    C(cd) = newCap;
    RETIRE_SUCCESS
  }
}
```

11 CMergeLT

Capability register *cd* is replaced with capability register *cs1* with the **fraction** field decremented by one. The tag on *cs2* is unset.

Exceptions

An exception is raised if:

- *cs1*.**tag** is not set.
- *cs1*.**otype** $\neq$ **otype_lifetime**.
- *cs1*.**linear** is not set.
- the value in *cs1* $\neq$ the value in *cs2*.
- *cd* $\neq$ *cs1*.

```
function clause execute(CMergeLT(cd, cs1, cs2)) = {
  let cs1_val = C(cs1);
  let cs2_val = C(cs2);
  if not (cs1_val.tag) then {
    handle_cheri_reg_exception(CapEx_TagViolation, cs1);
    RETIRE_FAIL
  } else if signed(cs1_val.otype) != otype_lifetime then {
    handle_cheri_reg_exception(CapEx_TypeViolation, cs1);
    RETIRE_FAIL
  } else if not(cs1_val.linear) then {
    handle_cheri_reg_exception(CapEx_LinearityViolation, cs1);
    RETIRE_FAIL
  } else if cs1_val != cs2_val then {
    handle_cheri_reg_exception(CapEx_BorrowPermissionViolation, cs2);
    RETIRE_FAIL
  } else if cd != cs1 then {
    handle_cheri_reg_exception(CapEx_LinearityViolation, cs1);
    RETIRE_FAIL
  } else {
    C(cs1) = invalidateCap(cs1_val);
    C(cs2) = invalidateCap(cs2_val);
    C(cd) = {cs1_val with B = cs1_val.B - 1};
    RETIRE_SUCCESS
  }
}
```

12 CBorrowImmut

Capability register *cs1* is replaced with capability register *cs1* with the **linear** field set to 0, the **otype** field set to *cs2*.**lifetime id** and the **perms** field set to the bitwise and of its previous value and **0b000000010101**. The original value of *cs1* is locked away in the borrow table and the capability register *cd* is replaced with its index token. If *cd* is **C0** no index token is generated and no capability is placed in the borrow table.

Exceptions

An exception is raised if:

- *cs1*.**tag** is not set.
- *cs1*.**otype** $\neq$ **otype_unsealed** and *cs1*.**otype** is not in the borrowed otype range.
- *cs1*.**otype** is in the borrowed otype range and *cs1*.**otype** $\neq$ *cs2*.**parent id**.
- *cs2*.**tag** is not set.
- *cs2*.**otype** $\neq$ **otype_lifetime**
- *cs2*.**linear** is not set.

```
function clause execute(CBorrowImmut(cd, cs1, cs2)) = {
  let cs1_val = C(cs1);
  let cs2_val = C(cs2);
  if not (cs1_val.tag) then {
    handle_cheri_reg_exception(CapEx_TagViolation, cs1);
    RETIRE_FAIL
  } else if signed(cs1_val.otype) != otype_unsealed & not(isCapBorrowed(cs1_val)) then {
    handle_cheri_reg_exception(CapEx_TypeViolation, cs1);
    RETIRE_FAIL
  } else if isCapBorrowed(cs1_val) & getLTParentID(cs2_val) != cs1_val.otype then {
    handle_cheri_reg_exception(CapEx_TypeViolation, cs1);
    RETIRE_FAIL
  } else if not (cs1_val.linear) & not(isCapBorrowed(cs1_val)) then {
    handle_cheri_reg_exception(CapEx_LinearityViolation, cs1);
    RETIRE_FAIL
  } else if not (cs2_val.tag) then {
    handle_cheri_reg_exception(CapEx_TagViolation, cs2);
    RETIRE_FAIL
  } else if signed(cs2_val.otype) != otype_lifetime then {
    handle_cheri_reg_exception(CapEx_TypeViolation, cs2);
    RETIRE_FAIL
  } else if not(cs2_val.linear) then {
    handle_cheri_reg_exception(CapEx_BorrowPermissionViolation, cs2);
    RETIRE_FAIL
  } else {
    let newPerms = ones(cap_uperms_width) @ EXTZ(cap_uperms_shift, 0b000000010101);
    let newCap = {setCapPerms(cs1_val, newPerms) with linear = false, otype = getLTLifetimeID(
        ↪ cs2_val)};
    C(cs1) = newCap;
    if (regidx_to_regno(cd) != 0) then {
      destCap = lockCapability(cs1_val, cs2_val);
      C(cd) = destCap;
    };
    RETIRE_SUCCESS
  }
}
```

13 CBorrowMut

Capability register *cs1* is replaced with capability register *cs1* with the **otype** field set to *cs2*.**lifetime id** and the **perms** field set to the bitwise and of its previous value and **0b000001111101**. The original value of *cs1* is locked away in the borrow table and the capability register *cd* is replaced with its index token.

Exceptions

An exception is raised if:

- *cs1*.**tag** is not set.
- *cs1*.**otype** $\neq$ **otype_unsealed** and *cs1*.**otype** is not in the borrowed otype range.
- *cs1*.**otype** is in the borrowed otype range and *cs1*.**otype** $\neq$ *cs2*.**parent id**.
- *cs2*.**tag** is not set.
- *cs2*.**otype** $\neq$ **otype_lifetime**
- *cs2*.**linear** is not set.

```
function clause execute(CBorrowMut(cd, cs1, cs2)) = {
  let cs1_val = C(cs1);
  let cs2_val = C(cs2);
  if not (cs1_val.tag) then {
    handle_cheri_reg_exception(CapEx_TagViolation, cs1);
    RETIRE_FAIL
  } else if signed(cs1_val.otype) != otype_unsealed & not(isCapBorrowed(cs1_val)) then {
    handle_cheri_reg_exception(CapEx_TypeViolation, cs1);
    RETIRE_FAIL
  } else if isCapBorrowed(cs1_val) & getLTParentID(cs2_val) != cs1_val.otype then {
    handle_cheri_reg_exception(CapEx_TypeViolation, cs1);
    RETIRE_FAIL
  } else if not (cs1_val.linear) & not(isCapBorrowed(cs1_val)) then {
    handle_cheri_reg_exception(CapEx_LinearityViolation, cs1);
    RETIRE_FAIL
  } else if not (cs2_val.tag) then {
    handle_cheri_reg_exception(CapEx_TagViolation, cs2);
    RETIRE_FAIL
  } else if signed(cs2_val.otype) != otype_lifetime then {
    handle_cheri_reg_exception(CapEx_TypeViolation, cs2);
    RETIRE_FAIL
  } else if not(cs2_val.linear) then {
    handle_cheri_reg_exception(CapEx_BorrowPermissionViolation, cs2);
    RETIRE_FAIL
  } else {
    let newPerms = ones(cap_uperms_width) @ EXTZ(cap_uperms_shift, 0b000001111101);
    let newCap = {setCapPerms(cs1_val, newPerms) with otype = getLTLifetimeID(cs2_val)};
    C(cs1) = newCap;
    C(cd) = lockCapability(cs1_val, cs2_val);
    RETIRE_SUCCESS
  }
}
```

14 CRetrieveIndex

Capability register *cd* is replaced with the capability in the borrow table at index *cs1*.**index**.

Exceptions

An exception is raised if:

- *cs1*.**tag** is not set.
- *cs1*.**otype** $\neq$ **otype_index**
- *cs2*.**tag** is not set.
- *cs2*.**otype** $\neq$ **otype_lifetime**
- *cs2*.**linear** is set.
- *cs1*.**lifetime id** $\neq$ *cs2*.**lifetime id**
- *cd* $\neq$ *cs1*.

```
function clause execute(CRetrieveIndex(cd, cs1, cs2)) = {
  let cs1_val = C(cs1);
  let cs2_val = C(cs2);
  if not (cs1_val.tag) then {
    handle_cheri_reg_exception(CapEx_TagViolation, cs1);
    RETIRE_FAIL
  } else if signed(cs1_val.otype) != otype_index then {
    handle_cheri_reg_exception(CapEx_TypeViolation, cs1);
    RETIRE_FAIL
  } else if not (cs2_val.tag) then {
    handle_cheri_reg_exception(CapEx_TagViolation, cs2);
    RETIRE_FAIL
  } else if signed(cs2_val.otype) != otype_lifetime then {
    handle_cheri_reg_exception(CapEx_TypeViolation, cs2);
    RETIRE_FAIL
  } else if cs2_val.linear then {
    handle_cheri_reg_exception(CapEx_BorrowPermissionViolation, cs2);
    RETIRE_FAIL
  } else if getITLifetimeID(cs1_val) != getLTLifetimeID(cs2_val) then {
    handle_cheri_reg_exception(CapEx_BorrowPermissionViolation, cs1);
    RETIRE_FAIL
  } else if cd != cs1 then {
    handle_cheri_reg_exception(CapEx_LinearityViolation, cs1);
    RETIRE_FAIL
  } else {
    C(cd) = retrieveCapability(cs1_val);
    RETIRE_SUCCESS
  }
}
```