Struct rust_cheri_compressed_cap::CcxCap

```
#[repr(C, align(16))]
pub struct CcxCap<T: CompressedCapability> {
    _cr_cursor: T::Addr,
    cr_pesbt: T::Addr,
    _cr_top: T::FfiLength,
    cr_base: T::Addr,
    cr_tag: u8,
    cr_bounds_valid: u8,
    cr_exp: u8,
    cr_extra: u8,
}
```

Structure matching the C type `_cc_N(cap)`. Field order and layout is binary-compatible with the C version, assuming the C preprocessor macro `_CC_REVERSE_PESBT_CURSOR_ORDER` is *not* defined.

This is a plain-old-data type. It only supplies getters and setters, and does *not* guarantee any safety/correctness. For example, there are no added assertions or checks if you set the cursor to a value outside the bounds. However, the C FFI functions from CompressedCapability may have their own asserts. These are documented where possible.

For a safe interface, use one of the crate::wrappers

Fields

`_cr_cursor: T::Addr`

The bottom half of the capability as stored in memory.

If Self::cr_tag is 1, this is the capability's "cursor" i.e. the address it's actually pointing to.

`cr_pesbt: T::Addr`

The top half of the capability as stored in memory.

If Self::cr_tag is 1, this is the compressed capability metadata (permissions, otype, bounds, etc.).

`_cr_top: T::FfiLength`

he top of this capability's valid address range. Derived from Self::cr_pesbt. As long as elf::cr_tag is 1, the getter/setter will ensure it matches.

`cr_base: T::Addr`

The base of this capability's valid address range. Derived from Self::cr_pesbt. As long as Self::cr_tag is 1, the getter/setter will ensure it matches.

`cr_tag: u8`

Tag - if 1, this is a valid capability, 0 it's just plain data

`cr_bounds_valid: u8`

0 (false) if the bounds decode step was given an invalid capability. Should be 1 (true) for all non-Morello capabilities.

`cr_exp: u8`

The exponent used for storing the bounds. Stored from various places, only used in Morello-exclusive function cap_bounds_uses_value().

`cr_extra: u8`

"Additional data stored by the caller." Seemingly completely unused, essentially padding.

Implementations

-] `impl<T: CompressedCapability> CcxCap<T>` [src]

Implements getters and setters similar to the C++-only member functions in the header.

`pub fn reg_representation(&self) -> (bool, [T::Addr; 2])` [src]

Returns a `(tag, [cursor, pesbt])` tuple that represents all data required to store a capability in a register.

To store capabilities in memory, see Self::mem_representation

`pub fn mem_representation(&self) -> (bool, [T::Addr; 2])` [src]

Returns a `(tag, [cursor, pesbt])` tuple that represents all data required to store a capability in memory.

To store capabilities in a register, see Self::reg_representation

`pub fn tag(&self) -> bool` [src]

`pub fn set_tag(&mut self, tag: bool)` [src]

`pub fn base(&self) -> T::Addr` [src]

```
ub fn top(&self) -> T::Length
```
[src]

```
pub fn bounds(&self) -> (T::Addr, T::Length)
```
[src]

```
pub fn set_bounds_unchecked(
    &mut self,
    req_base: T::Addr,
    req_top: T::Length
) -> bool
```
[src]

Sets the base and top of this capability using C FFI function CompressedCapability::set_bounds. Updates the PESBT field correspondingly. On non-Morello platforms, will fail with an assertion error if Self::tag() is not set.

```
pub fn address(&self) -> T::Addr
```
[src]

```
pub fn set_address_unchecked(&mut self, addr: T::Addr)
```
[src]

```
pub fn offset(&self) -> T::Offset
```
[src]

```
pub fn length(&self) -> T::Length
```
[src]

```
pub fn software_permissions(&self) -> u32
```
[src]

```
pub fn set_software_permissions(&mut self, uperms: u32)
```
[src]

```
pub fn permissions(&self) -> u32
```
[src]

```
pub fn set_permissions(&mut self, perms: u32)
```
[src]

```
pub fn otype(&self) -> u32
```
[src]

```
pub fn is_sealed(&self) -> bool
```
[src]

```
pub fn set_otype(&mut self, otype: u32)
```
[src]

```
pub fn reserved_bits(&self) -> u8
```
[src]

```
pub fn set_reserved_bits(&mut self, bits: u8)
```
[src]

```
pub fn flags(&self) -> u8
```
[src]

```
pub fn set_flags(&mut self, flags: u8)
```
[src]

```
pub fn is_exact(&self) -> bool
```
[src]

Helper function for easily calling FFI function CompressedCapability::is_representable_cap_exact on this capability. Assertions are present in the C code, but should never be triggered.

```
pub fn is_representable_with_new_addr(&self, new_addr: T::Addr)
```
[src]

> bool

Helper function for easily calling FFI function CompressedCapability::is_representable_new_addr on this capability. Assertions are present in the C code, but should never be triggered.

Trait Implementations

```
impl<T: Clone + CompressedCapability> Clone for CcxCap<T>
where
    T::Addr: Clone,
    T::Addr: Clone,
    T::FfiLength: Clone,
    T::Addr: Clone,
```
[src]

```
impl<T: CompressedCapability> Debug for CcxCap<T>
```
[src]

Debug printer for capabilities that decodes the PESBT field instead of printing it raw.

```
impl<T: CompressedCapability> Default for CcxCap<T>
```
[src]

Equivalent to initialization pattern used in tests:

```
ccx_cap_t value;
memset(&value, 0, sizeof(value));
```

cc64.rs doesn't pick it up when it was automatically #derive-d, so it's manually implemented here

```
impl<T: CompressedCapability> PartialEq<CcxCap<T>> for
CcxCap<T>
```
[src]

Implements `operator==` from cheri_compressed_cap_common.h

```
impl<T: Copy + CompressedCapability> Copy for CcxCap<T>
where
    T::Addr: Copy,
    T::Addr: Copy,
    T::FfiLength: Copy,
    T::Addr: Copy,
```
[src]

```
impl<T: CompressedCapability> Eq for CcxCap<T>
```
[src]

Auto Trait Implementations

```
impl<T> RefUnwindSafe for CcxCap<T>
where
    <T as CompressedCapability>::Addr: RefUnwindSafe,
    <T as CompressedCapability>::FfiLength: RefUnwindSafe,
```

```
impl<T> Send for CcxCap<T>
```

```
ere
    <T as CompressedCapability>::Addr: Send,
    <T as CompressedCapability>::FfiLength: Send,
```

```
impl<T> Sync for CcxCap<T>
where
    <T as CompressedCapability>::Addr: Sync,
    <T as CompressedCapability>::FfiLength: Sync,
```

```
impl<T> Unpin for CcxCap<T>
where
    <T as CompressedCapability>::Addr: Unpin,
    <T as CompressedCapability>::FfiLength: Unpin,
```

```
impl<T> UnwindSafe for CcxCap<T>
where
    <T as CompressedCapability>::Addr: UnwindSafe,
    <T as CompressedCapability>::FfiLength: UnwindSafe,
```

Blanket Implementations

-]
```
impl<T> Any for T
where
    T: 'static + ?Sized,
```
[src]

-]
```
impl<T> Borrow<T> for T
where
    T: ?Sized,
```
[src]

-]
```
impl<T> BorrowMut<T> for T
where
    T: ?Sized,
```
[src]

-]
```
impl<T> From<T> for T
```
[src]

-]
```
impl<T, U> Into<U> for T
where
    U: From<T>,
```
[src]

-]
```
impl<T> ToOwned for T
where
    T: Clone,
```
[src]

```
type Owned = T
```

The resulting type after obtaining ownership.

-]
```
impl<T, U> TryFrom<U> for T
where
    U: Into<T>,
```
[src]

```
type Error = Infallible
```

The type returned in the event of a conversion error.

```
npl<T, U> TryInto<U> for T
ere
    U: TryFrom<T>,
```
[src]

```
type Error = <U as TryFrom<T>>::Error
```

The type returned in the event of a conversion error.