Typically, the lowest stack content are the parameters passed to the function!
(CPU ABI).

View by increasing addresses upwards:

```
[...        <-- 'Top' of the stack (ESP); lower (virtual) addresses
LOCALS
...]
[ER]BP/SFP     <-- EBP (base ptr) OR SFP (pointer to previous stack frame
[optional])
RET addr
[...
PARAMS
...]        <-- 'Bottom'; higher (virtual) addresses
```

Or, perhaps more intuitively, view it the 'usual' way, by increasing addresses downwards:

```
[...        <-- 'Bottom'; higher (virtual) addresses
PARAMS
...]
RET addr
[ER]BP/SFP  <-- EBP (base ptr) OR SFP (pointer to previous stack frame [optional])

[...
LOCALS
...]        <-- 'Top' of the stack (ESP); lower (virtual) addresses
```

Eg.

```
crash> log
…

veth_netdrv:vnet_start_xmit(): skb=ffff950a510d8500 netdev=ffff950a8889d000
pstCtx=ffff950a8889d900
…
```

Looks like the KVA's (kernel va's) in green are linkage to called frame (function).

```
crash> bt -FF
...
...
 #9 [ffffae4543f4b8c0] skb_release_data at ffffffff81dc6612
    ffffae4543f4b8c8: [ffff950a85450424:kmalloc-512]
[ffff950a510d8500:skbuff_head_cache]
    ffffae4543f4b8d8: [ffff950a8889d000:kmalloc-4k] ffffae4543f4b8f8
    ffffae4543f4b8e8: consume_skb+62

#10 [ffffae4543f4b8e8] consume_skb at ffffffff81dc6a0e
    ffffae4543f4b8f0: [ffff950a510d8500:skbuff_head_cache] ffffae4543f4b928
    ffffae4543f4b900: vnet_start_xmit+157

<<
static int vnet_start_xmit(struct sk_buff *skb, struct net_device *ndev)
{
      struct iphdr ip = NULL; /* params/locals seen in RTL (right-to-left) order */
      struct udphdr *udph = NULL;
      struct stVnetIntfCtx *pstCtx = netdev_priv(ndev);
      ...
      u64 ts1, ts2; << unused, optimized away >>
>>
#11 [ffffae4543f4b900] vnet_start_xmit at ffffffffc07b32ad [veth_netdrv]
    ffffae4543f4b908: [ffff950a5016ec00:kmalloc-512] 0000000000000000
    ffffae4543f4b918: 000000000000003b [ffff950a510d8500:skbuff_head_cache]
    ffffae4543f4b928: ffffae4543f4b978 dev_hard_start_xmit+208  <<RET addr>>

<<-- Call frame layout:
LOCALS
PARAMS                  kva
SFP?/RBP?
RET addr
-->>

#12 [ffffae4543f4b930] dev_hard_start_xmit at ffffffff81de2b70
    ffffae4543f4b938: ffffae4543f4b994 [ffff950a8889d088:kmalloc-4k]
    ffffae4543f4b948: ffffae4543f4b978 [ffff950a5016ec00:kmalloc-512]
    ffffae4543f4b958: [ffff950a88b57e00:kmalloc-512]
[ffff950a510d8500:skbuff_head_cache]
    ffffae4543f4b968: [ffff950a88b57eac:kmalloc-512] [ffff950a8889d000:kmalloc-4k]
    ffffae4543f4b978: ffffae4543f4b9c8 sch_direct_xmit+226

#13 [ffffae4543f4b980] sch_direct_xmit at ffffffff81e33c62
    ffffae4543f4b988: 0000000581ec13f1 000000100089d000
```

```
    ffffae4543f4b998: 277ccc06ec916700 0000000000000000
    ffffae4543f4b9a8: [ffff950a510d8500:skbuff_head_cache]
[ffff950a88b57e00:kmalloc-512]
    ffffae4543f4b9b8: [ffff950a8889d000:kmalloc-4k] [ffff950a5016ec00:kmalloc-512]
    ffffae4543f4b9c8: ffffae4543f4ba38 __dev_queue_xmit+1705

#14 [ffffae4543f4b9d0] __dev_queue_xmit at ffffffff81de33d9
    ffffae4543f4b9d8: ffffae4543f4b9e8 [ffff950a88b57eac:kmalloc-512]
    ffffae4543f4b9e8: fffffff443f4ba70 0000000000000000
    ffffae4543f4b9f8: ffffae4543f4bb44 [ffff950a412f0000:task_struct]
    ffffae4543f4ba08: 277ccc06ec916700 [ffff950a85451000:kmalloc-512]
    ffffae4543f4ba18: 0000000000000000 [ffff950a510d8500:skbuff_head_cache]
    ffffae4543f4ba28: [ffff950a8889d000:kmalloc-4k] 0000000000000008
    ffffae4543f4ba38: ffffae4543f4ba48 dev_queue_xmit+16

#15 [ffffae4543f4ba40] dev_queue_xmit at ffffffff81de35e0
    ffffae4543f4ba48: ffffae4543f4ba80 neigh_resolve_output+276
...
```

[ffff950a510d8500:skbuff_head_cache]
 # the SKB that's freed up too soon, res in a UAF; the root cause of the bug

```
crash> bt -F
...
...
 #9 [ffffae4543f4b8c0] skb_release_data at ffffffff81dc6612
    ffffae4543f4b8c8: [kmalloc-512]    [skbuff_head_cache]
    ffffae4543f4b8d8: [kmalloc-4k]     ffffae4543f4b8f8
    ffffae4543f4b8e8: consume_skb+62
#10 [ffffae4543f4b8e8] consume_skb at ffffffff81dc6a0e
    ffffae4543f4b8f0: [skbuff_head_cache] ffffae4543f4b928
    ffffae4543f4b900: vnet_start_xmit+157
#11 [ffffae4543f4b900] vnet_start_xmit at ffffffffc07b32ad [veth_netdrv]
    ffffae4543f4b908: [kmalloc-512]    0000000000000000
    ffffae4543f4b918: 000000000000003b [skbuff_head_cache]
    ffffae4543f4b928: ffffae4543f4b978 dev_hard_start_xmit+208
#12 [ffffae4543f4b930] dev_hard_start_xmit at ffffffff81de2b70
    ffffae4543f4b938: ffffae4543f4b994 [kmalloc-4k]
    ffffae4543f4b948: ffffae4543f4b978 [kmalloc-512]
    ffffae4543f4b958: [kmalloc-512]    [skbuff_head_cache]
    ffffae4543f4b968: [kmalloc-512]    [kmalloc-4k]
    ffffae4543f4b978: ffffae4543f4b9c8 sch_direct_xmit+226
#13 [ffffae4543f4b980] sch_direct_xmit at ffffffff81e33c62
    ffffae4543f4b988: 0000000581ec13f1 000000100089d000
    ffffae4543f4b998: 277ccc06ec916700 0000000000000000
    ffffae4543f4b9a8: [skbuff_head_cache] [kmalloc-512]
    ffffae4543f4b9b8: [kmalloc-4k]     [kmalloc-512]
    ffffae4543f4b9c8: ffffae4543f4ba38 __dev_queue_xmit+1705
#14 [ffffae4543f4b9d0] __dev_queue_xmit at ffffffff81de33d9
    ffffae4543f4b9d8: ffffae4543f4b9e8 [kmalloc-512]
    ffffae4543f4b9e8: fffffff443f4ba70 0000000000000000
    ffffae4543f4b9f8: ffffae4543f4bb44 [task_struct]
    ffffae4543f4ba08: 277ccc06ec916700 [kmalloc-512]
    ffffae4543f4ba18: 0000000000000000 [skbuff_head_cache]
```

```
    ffffae4543f4ba28: [kmalloc-4k]     0000000000000008
    ffffae4543f4ba38: ffffae4543f4ba48 dev_queue_xmit+16
#15 [ffffae4543f4ba40] dev_queue_xmit at ffffffff81de35e0
    ffffae4543f4ba48: ffffae4543f4ba80 neigh_resolve_output+276
...
...
```