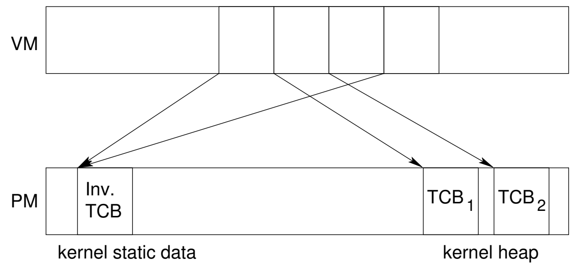
MICROKERNEL (L4) IMPLEMENTATION

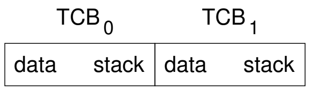
- Overview of main features of L4 implementation:
  - ➜ data structures,
  - ➜ algorithms.
- These notes are *very* short, they are complimented by:
  - ➜ L4/MIPS source code [L4M99],
  - ➜ "Inside L4/MIPS" [Hei00]

VIRTUAL TCB ARRAY:



➜ Fast TCB location (`macros.h:tid2tcb`).

TWO TCBS PER (4KB) PAGE:



- per-thread kernel stack
  - ➜ fast context switch;
- kernel stack in TCB
  - ➜ locality.

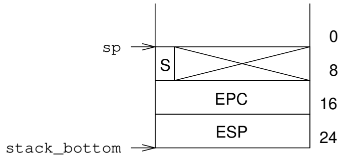
TCB LAYOUT

```
typedef struct tcb {
0:  struct tcb *sndq_end, *wakeup_link, *busy_link, *int_link;
1:  unsigned long wfor; struct tcb *sndq_start;
    unsigned long stack_pointer; udw_t asid;
2:  void *gpt_pointer; unsigned long myself;
    unsigned int fine_state, timeout; long recv_desc;
3:  struct tcb *present_next, *child_task; uhw_t rem_timeslice,
    timeslice; ub_t mcp, bpad1, ctsp, tsp;
4:  long wakeup; struct tcb *soon_wakeup_link,
    *late_wakeup_link, *sndq_next;
    ...
}
```

Kernel data layout

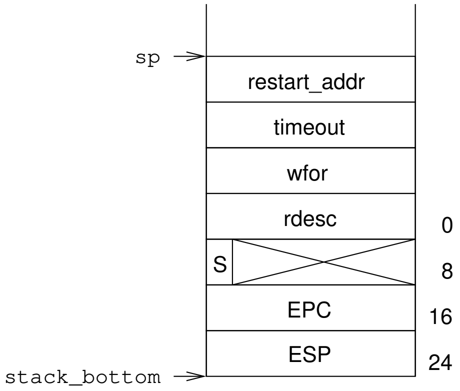
```
typedef struct kern {
0:  dw_t stack_bottom; udw_t s0_save, *free_asid_list;
    tcb_t *wakeup_list;
    ...
}
```


Kernel stack:



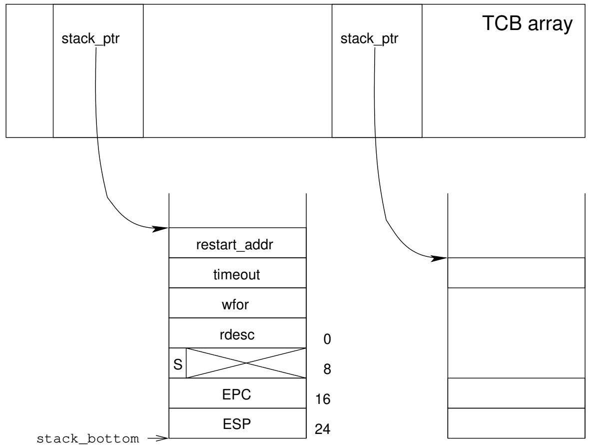
Exception stack as set up by general exception handler (prior to `k_ipc`).

Kernel stack...



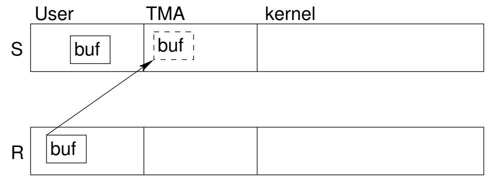
Sender's kernel stack when switching to receiver (end of `deliver`).

Kernel stacks & TCBs



ACCESSING RECEIVER'S DATA DURING LONG IPC

- All IPC processing is done in sender's context.
  ➜ All of sender's address space is accessible (but may get TLB misses).
- To access receiver's address space (for receive descriptor, strings):
  ➜ shift all receiver addresses into *temporary mapping area*
  ➜ is located in kernel-reserved address range (0x4000 0000 0000 0000),
  ➜ TLB miss handler recognises address range and handles it by translating receiver page table entries,
  ➜ if unmapped, invoke receiver's pager.



- Implies handling user page faults in kernel mode.

PAGE FAULT HANDLING

- TLB misses in user mode invoke fast TLB miss hander to reload from page table.
- TLB misses in kernel mode, or TLB misses on unmapped user memory, invoke the slow TLB miss handler, `exc_tlbs`, `exc_tlbl`.
- Deals with three kinds of misses:
  ① $0 \leq f < 2^{62}$: user page fault ⇒ invoke pager
  ② $2^{62} \leq f < 2^{63}$: window fault ⇒ look up receiver's PT & remap
  ③ $2^{63} \leq f < 2^{64}$: kernel TLB miss on TCB array ⇒ allocate TCB
- Note: last two can only happen in kernel mode, first can happen in kernel or user mode.
  ➜ Need to be able to invoke user's pager during IPC.
  ➜ Kernel "fakes" exception stack to invoke pager as per IPC syscall (using `k_ipc` code).
  ➜ "EPC" is restart routine.

REFERENCES

[Hei00] Gernot Heiser. Inside L4/MIPS: Anatomy of a high-performance microkernel. Technical report, School Comp. Science & Engin., University NSW, Sydney 2052, Australia, 2000. Latest version available from http://www.cse.unsw.edu.au/~disy/L4/.

[L4M99] L4/MIPS source code, kernel version 79. Available from http://www.cse.unsw.edu.au/~disy/L4/, February 1999.