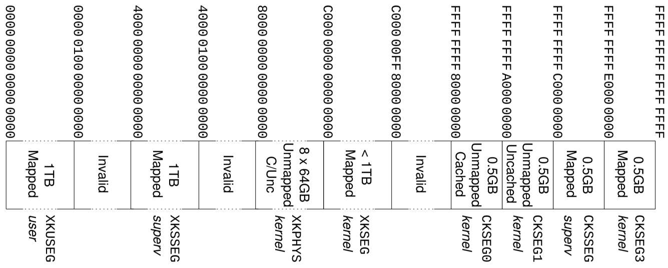
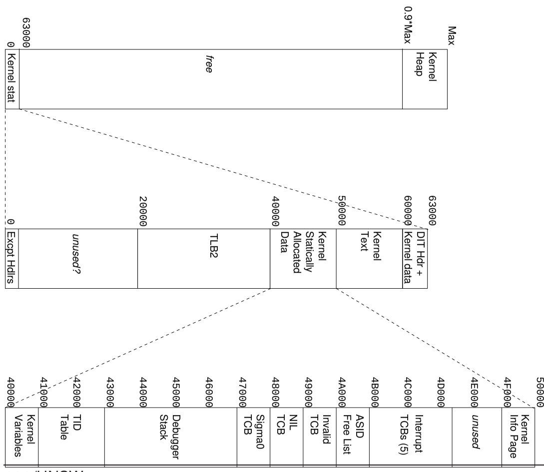
Microkernel Implementation

- Overview of main features of L4 implementation:
  - ➜ data structures,
  - ➜ algorithms.
- Based on MIPS implementation (assembler kernel by Elphinstone)
- These notes are *very* brief, they are complemented by:
  - ➜ L4/MIPS source code [L4M99],
  - ➜ “Inside L4/MIPS” [Hei00]

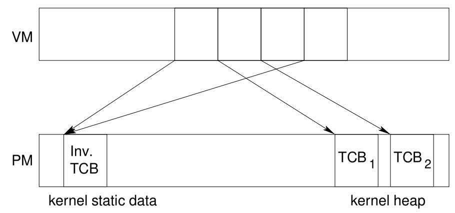
MIPS ADDRESS SPACE



L4 PHYSICAL MEMORY MAP

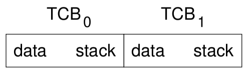


VIRTUAL TCB ARRAY:



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

TWO TCBS PER (4KB) PAGE:



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

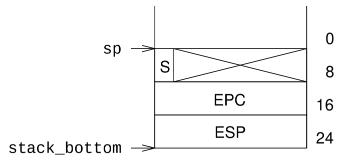
TCB LAYOUT

```
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

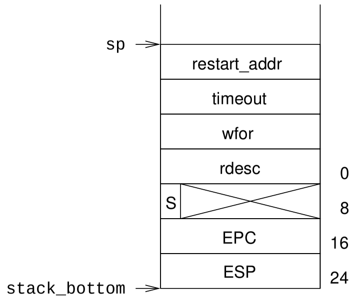
```
typedef struct kern {
0:  dw_t stack_bottom;
    udw_t s0_save, *free_asid_list;
    tcb_t *wakeup_list;
    ...
}
```


KERNEL STACK:



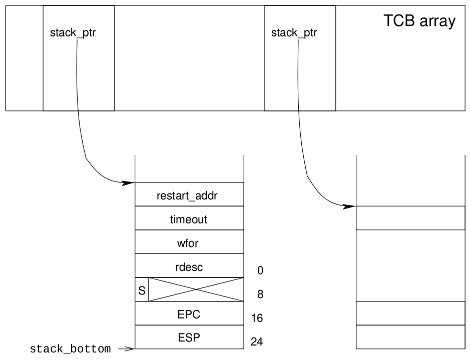
Exception stack as set up by general exception handler (prior to `k_ipc`).

KERNEL STACK...



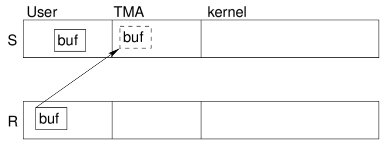
Sender's kernel stack when switching to receiver (end of `deliver`).

KERNEL STACKS & TCBS



ACCESSING RECEIVER'S DATA DURING LONG IPC

- All IPC processing is done in sender's context.
  - ➜ Sender's address space accessible (but may get TLB misses).
- To access receiver's data (for receive descriptor, strings):
  - ➜ shift all receiver addresses into *temporary mapping area*
  - ➜ TMA located in kernel-reserved address range ($\geq 2^{62}$),
  - ➜ TLB miss handler recognises TMA address range
    - ➜ faults handled by translating receiver page table entries,
    - ➜ if not mapped, 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
  - ➜ reload from page table
- Misses in kernel mode, or on unmapped user memory:
  - ➜ invoke the slow TLB miss handler, `exc_tlbs`, `exc_tlbl`.
- Deals with three kinds of misses:
  - $0 \leq f < 2^{62}$ user page fault ⇒ invoke pager
  - $2^{62} \leq f < 2^{63}$ window fault ⇒ look up receiver's PT & remap
  - $2^{63} \leq f < 2^{64}$ kernel TLB miss on TCB array ⇒ allocate TCB
- first can happen any time, last two only in kernel 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. Report, OS Research Group, School Comp. Sci. & 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/`, Feb 1999.