RTOSes
Part I

Christopher Kenna
September 24, 2010

POSIX

- Portable Operating System for UnIX
- Application portability at source-code level
- POSIX Family formally known as IEEE 1003
- Originally 17 separate documents, but 10 have since been combined

POSIX Versions

- Prior to 1997:
  - POSIX.1 (Core Services, 1988)
  - POSIX.1b (Real-time extensions, 1993)
    - POSIX.4 was POSIX.1b before approval
  - POSIX.1c (Threads, 1995)
  - Posix.2 (Shell and Utilities, 1992)

POSIX Versions (2)

- After 1997:
  - POSIX:2001
    - Base definitions, System interfaces and Headers, Commands and Utilities
  - POSIX:2004 - Minor updates
  - POSIX:2008 - Current version

POSIX.1-2008

- ... is massive. It covers:
    - Concurrent execution
    - Directory protection
    - File access permissions
    - File hierarchy
    - Filenames

POSIX.1-2008 (2)

- Continued...
  - Memory synchronization
  - Tracing
  - Threads
  - ...
- We are interested in real-time extensions.

POSIX.4 - Scheduling

- POSIX requires only that the implementation define how scheduling policies modify thread priorities
- POSIX.4 concretely specifies three scheduling policies

POSIX.4 - Scheduling (2)

- Scheduling attributes of a process may be:
  - SCHED_FIFO: At least 32 priority levels with fixed-priority preemptive scheduling. Process with the same priority are FIFO. Runs until blocked ore preempted by a higher priority process.
  - SCHED_RR: Also 32 priority levels, except process with the same priority are scheduled round-robin.
  - SCHED_OTHER: Static priority 0. Implementation defined; standard Linux scheduler.

POSIX.4 - VM

- Functions to lock all or part of process address space into physical memory.
- Avoids delays due to memory access.
- Aside: PREEMPT_RT recommends calling `mlockall()` as soon as possible from `main()` to reduce future page faults.

POSIX.4 - Real-Time Signals

- POSIX.4 defines a new range of signals. There are many more user signals than just `SIGUSR1` and `SIGUSR2`.
- Signals are queued, not lost. If several signals arrive before the handler is called, they are all delivered.
- Signals are delivered in priority order.
- Signals may contain an integer or pointer as data.

POSIX.4 - IPC

- POSIX.4 defines message queues to communicate between processes.
- Messages are prioritized to avoid priority inversion.
- Message transmission and reception may be blocking or non-blocking.

POSIX.4 - Synchronization and Memory

- POSIX.4 defines named and unnamed counting semaphores
    - Named: Names constructed like file paths
    - Unnamed: Memory based
- Priority inversions are still an issue.
- `mmap()` maps portions of process address space to memory objects. It is now in POSIX.1-2008

POSIX.4 - Clocks & I/O

- CLOCK_REALTIME must have resolution at least 50 HZ (20 ms)
- POSIX.4 provides (a-)synchronous I/O.
- Synchronous: Ensure that the data hits the disk (`fsync()`)
- Asynchronous: Does I/O in parallel with the application
  - OS queues read/write requests and immediately returns control to the application.
  - I/O is carried out in parallel with the application.
  - A signal can be delivered to the application when I/O is complete.

POSIX Trivia

RMS

- POSIX mandates 512-byte block size
- GNU OS implementers used 1024-byte blocks
- POSIXLY_CORRECT was introduced to force standards-compliant behavior
- Stallman’s original plan was to name the variable POSIX_ME_HARDER

Partitioned OSes

- OSes for embedded applications support both temporal and spatial partitioning.
- Partitioning prevents unwanted interference between applications.
- Partitioning contains application faults to the partition in which they occurred.

Temporal Partitioning

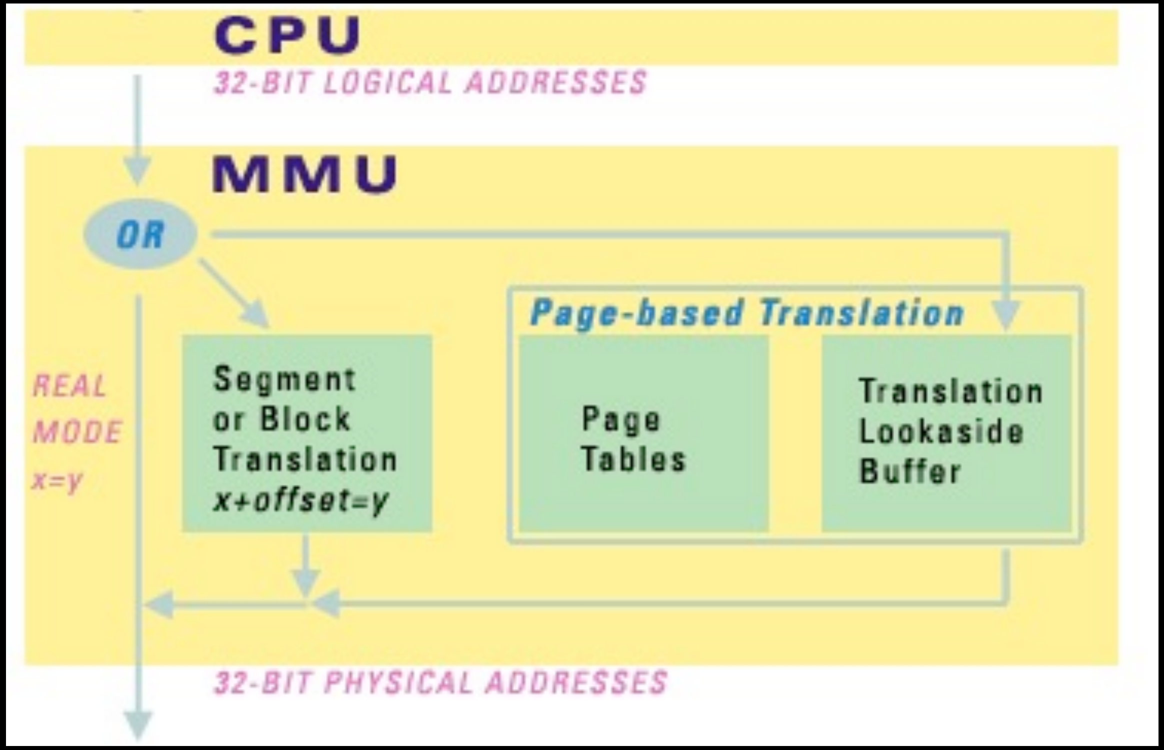
- Temporal partitioning is simply dividing up CPU time and access to other resources.
- This is typically done through static table-driven scheduling.

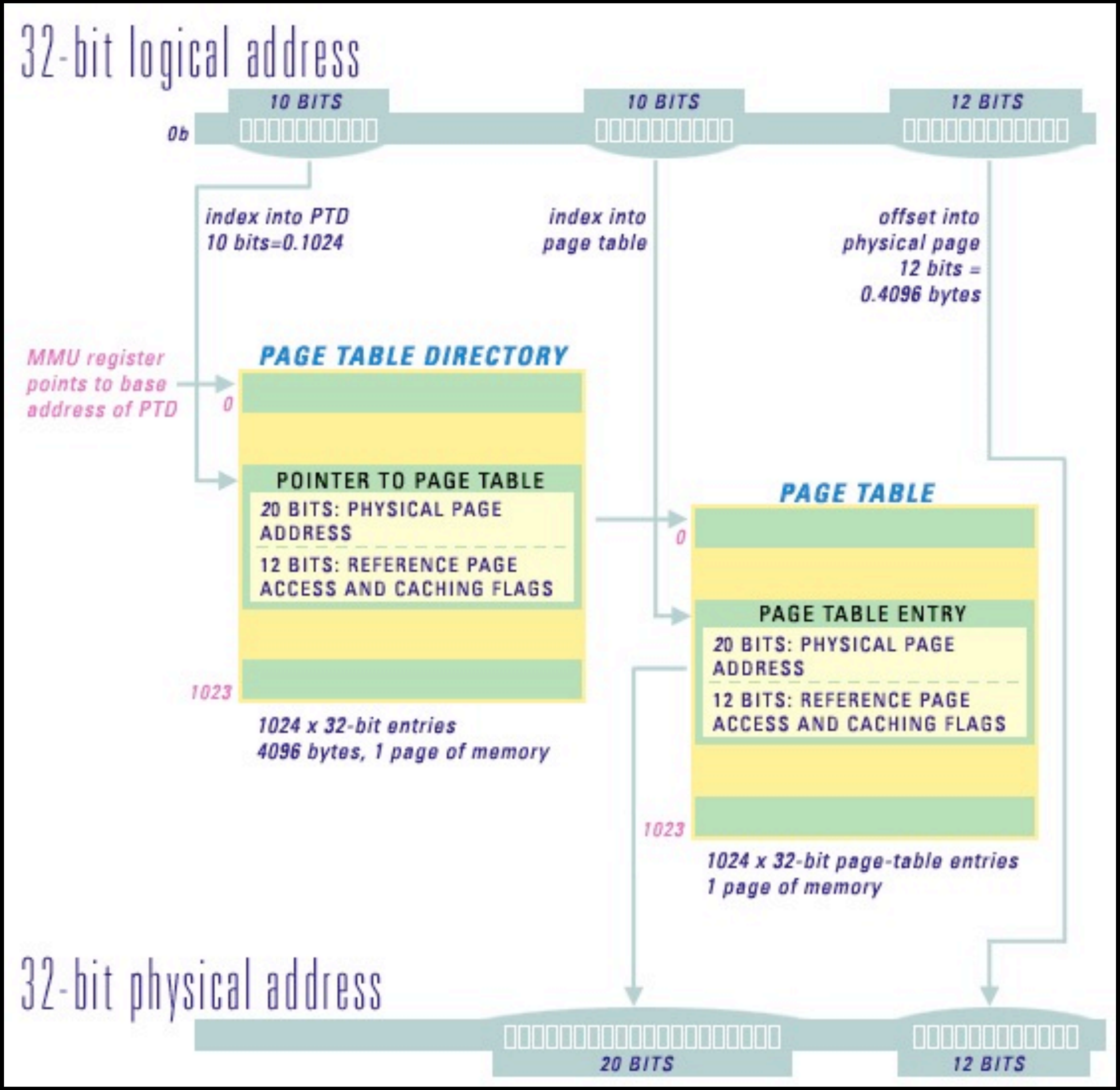
Spatial Partitioning

- Memory partitioning must be provided with hardware.
    - MMU: Virtual address translation
    - MPU: Simplified MMU
- MMU does virtual address translation in hardware to partition applications spatially.
- Ensures one software component cannot access the memory of another.

MMU Addressing

- Real: No address translation.
- Block: Translate segments of logical addresses to equivalent sized segments of physical memory.
- Page-based: Translation is done on a page-by-page basis.





Page-table format used on Intel x86, Pentium, and Pentium Pro family, as well as on the PowerPC 821 and 860 PowerQUICC processors.

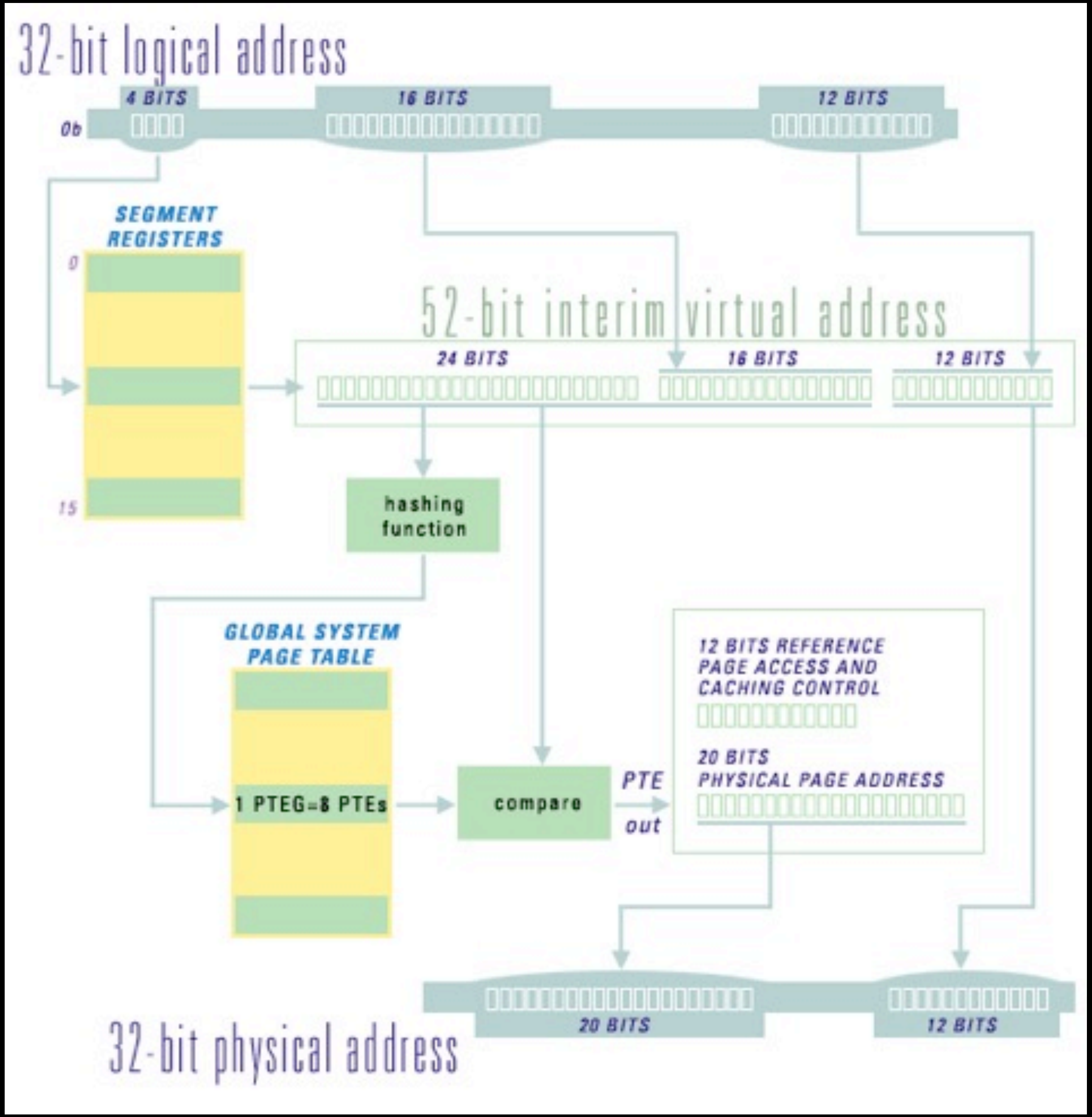
PPC Address Translation

- PPC architecture G1 through G4 can map linear chunks of addresses between 128 MB and 256 MB.
- The PPC can also do segmented address translation, in which we have standard 4 KB pages.

PowerPC 60x family and MPC750

System designer loads MMU segment registers with a value associated to the current context.

1. This value is prepended to the virtual address, resulting in the a 52-bit virtual address.
2. Virtual address is hashed by the MMU and points to the Page Table Entry Group.
3. Comparison done to find physical address.



TLB

- When the Intel processor context switches, the TLB must be flushed.
- The PPC page table is global, so no TLB flush is required.
- The segment registers simply shift the window in the 52-bit address space.

LynxOS

LynuxWorks offers many RTOSes.

- LynxOS RTOS: Hard RTOS for embedded systems.
- LynxOS-SE RTOS: Memory is partitioned, ARINC 653 based fixed-cyclic scheduling.
- LynxOS-178 RTOS: DO-178B level A.

LynxOS-178

- LynxOS-178 is used:
  - By the Navy in missiles and helicopters.
  - By the Air Force in refueling tankers for electronic display control units.
  - For Airbus navigation systems.
  - For the Boeing 777 cabin services system.
  - ... much more.

LynxOS-178 Details

- Supports Pentium and PPC architectures.
- Meets all DO-178B requirements, and provides design data, test suites, etc. to certify new applications.
- POSIX.1 with real-time and thread extensions.
- ARINC 653 APEX for partition communication.

ARINC 653-1 & LynxOS

- ARINC 653-1 Application Executive Software (APEX) Interface defines:
- Interpartition Communication through ports:
  - Sampling port: Memory space updated at a given rate.
  - Queuing port: Values queued/dequeued by writers/readers.

ARINC 653 Intrapartition Communication

- Buffer Services: A message passing queue.
- Blackboard Services: Processes may read, write, and clear a single message.
- Semaphore Services: Counting semaphore.
- Event Services: Can notify process when conditions occur.

LynxOS-178 Architecture

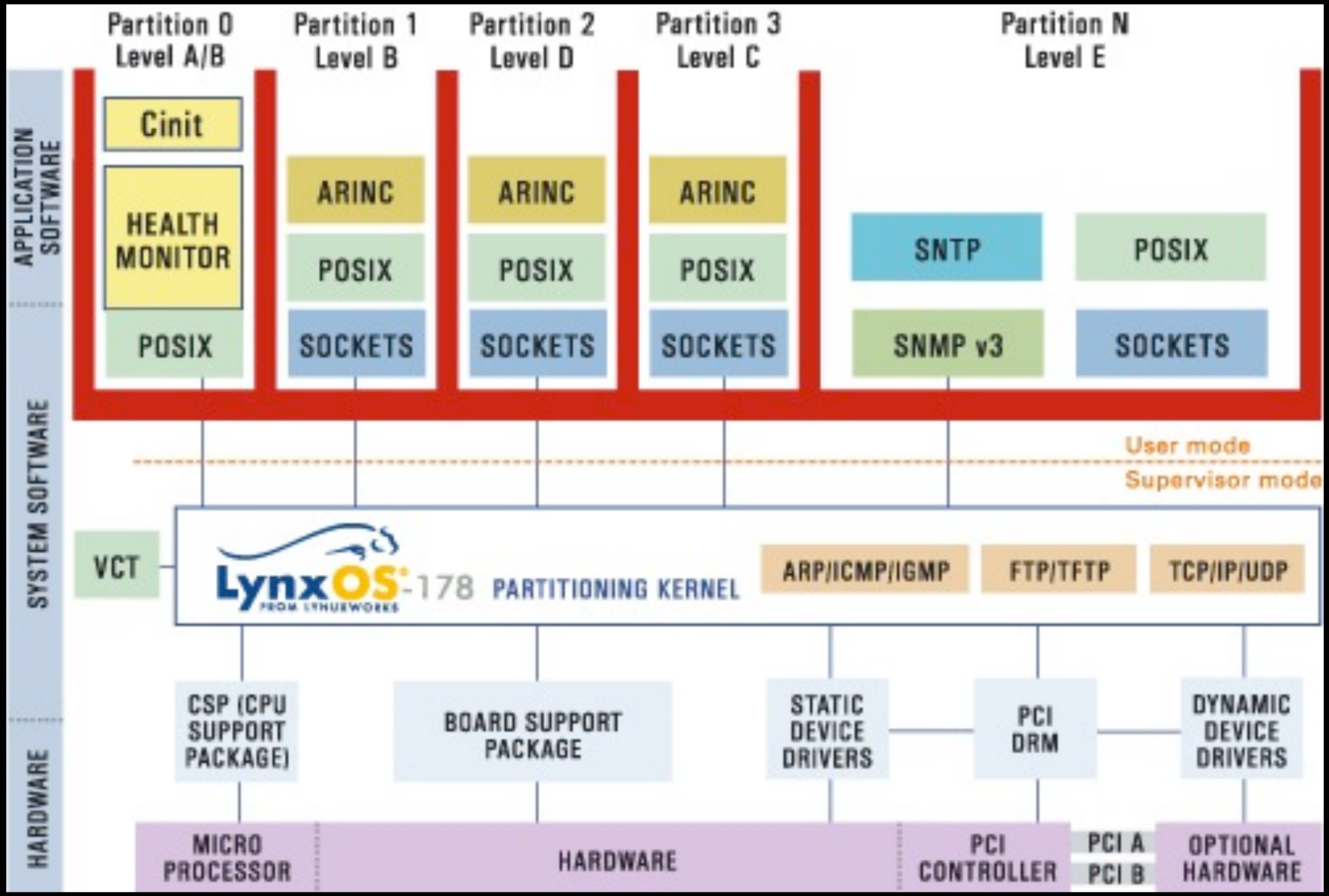
- Time partitioning through a fixed-cyclic scheduler.
- Memory partitioning in discrete blocks. Processes in a partition use virtual addressing in that partition's block.
- Each device assigned to one partition.
  - Isolate driver faults.

LynxOS-178 Architecture

- Each partition mounts a RAM disk for storage (supports flash, too).
- Developers can use the serial port.
- Can mount an external disk for “testing” and “data capture.”

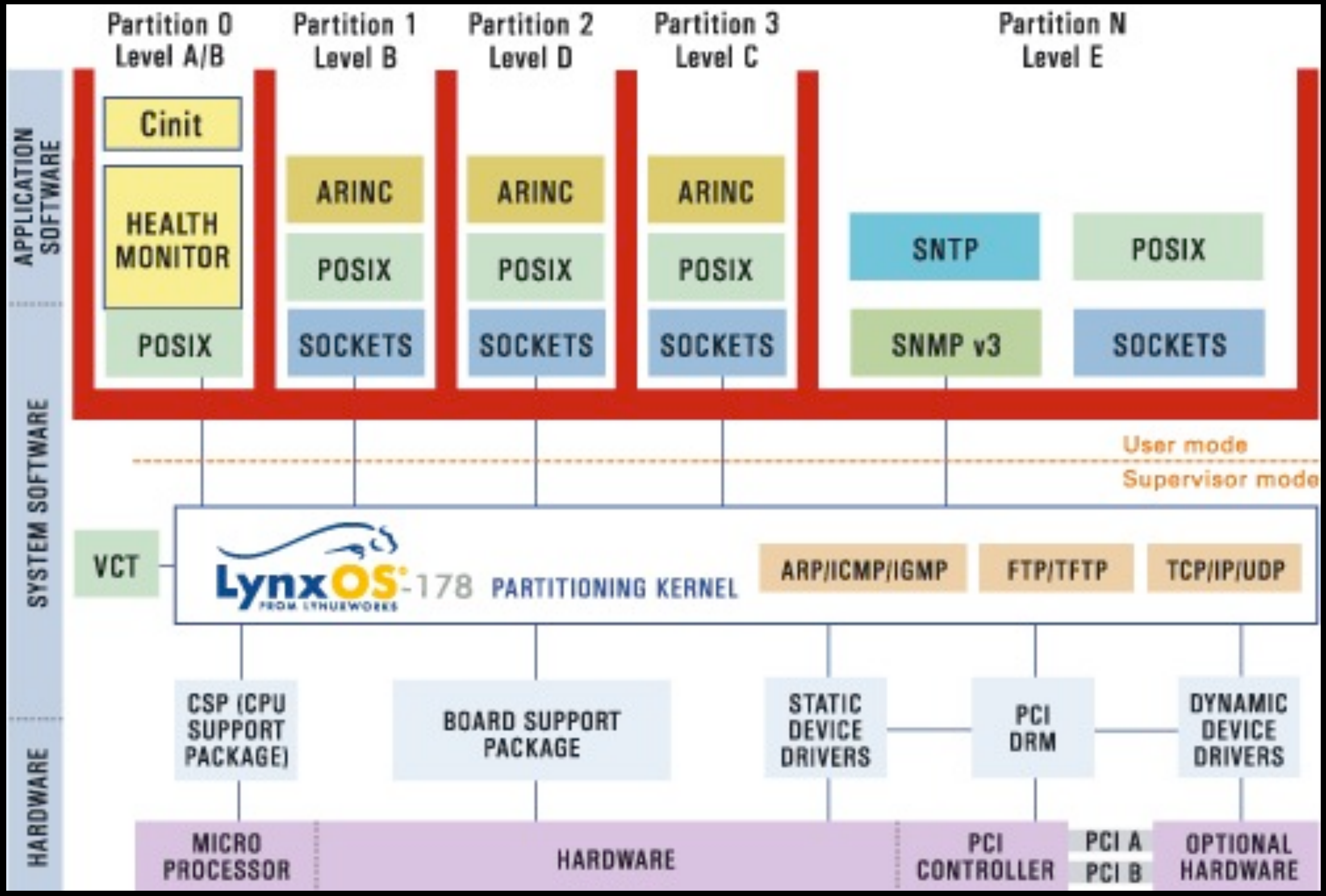
LynxOS-178 Architecture (2)

- Kontron VMPC6x board holds boot code in firmware.
  - 500 MHz PPC G4
  - Up to 512 MB RAM
- CPU Support Package contains MMU and FP units.
  - CSP routines linked with kernel.



LynxOS-178 Architecture (3)

- Board Support Package: Interface with interrupt and PCI controllers.
- Static device drivers are linked with the kernel.
- Dynamic device drivers are for optional devices and loaded before partitioning is invoked.

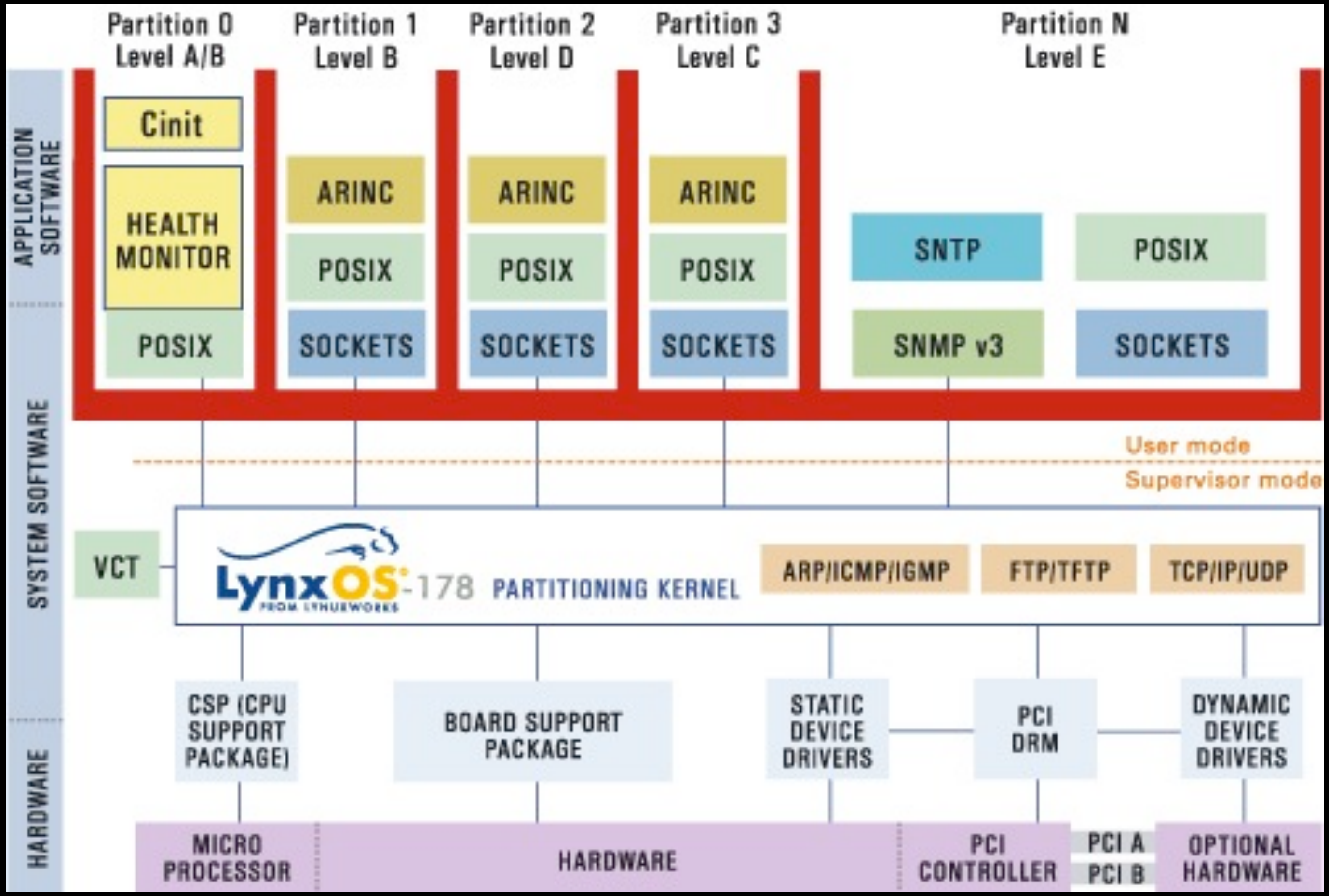


LynxOS-178 Architecture (4)

- POSIX application code must be C or C++.

- Cinit is first process, which mounts file systems, loads dynamic device drivers, etc.

- Cinit appears in each partition as Pinit, and is the first process in each.



Interrupts in LynxOS

- LynxOS doesn’t schedule processes, it schedules POSIX threads.
- LynxOS supports kernel threads, which are threads that execute within the kernel to handle interrupts.
- The interrupt handling threads spawn with the lowest priority, and later assume the priority of the user process they service.

Interrupts in LynxOS (2)

- The rest of the interrupt handling is done with standard split interrupt handling.
- After the kernel does the first half of the interrupt processing for a device, interrupts remain disabled for that device until the kernel thread finishes execution.