Real-Time Systems, were better in the past

(oh, really?)

Hans Zuidam,
for 040coders.nl
September 20, 2018

What is real-time anyway?

- Systems that typically control some external physical process
- Algorithms must be logically correct
- In a real-time system the time at which the algorithm completes is crucial
- Cannot be too late

What is real-time anyway?

- Hard real-time

  The system will fail if a computation is not completed before a specified deadline

- Soft real-time

  The system can recover from missing timing deadlines

Clocks

- Confusing terminology
- Real-time clock also called a wall clock
- May move backward (leap seconds)
- May miss ticks

- Wall clocks are not good enough

Clocks

- Real real-time clock is monotonic increasing
- Never moves backward
- Never skips a pulse
- Resolution determines measurement accuracy

Scheduling algorithms

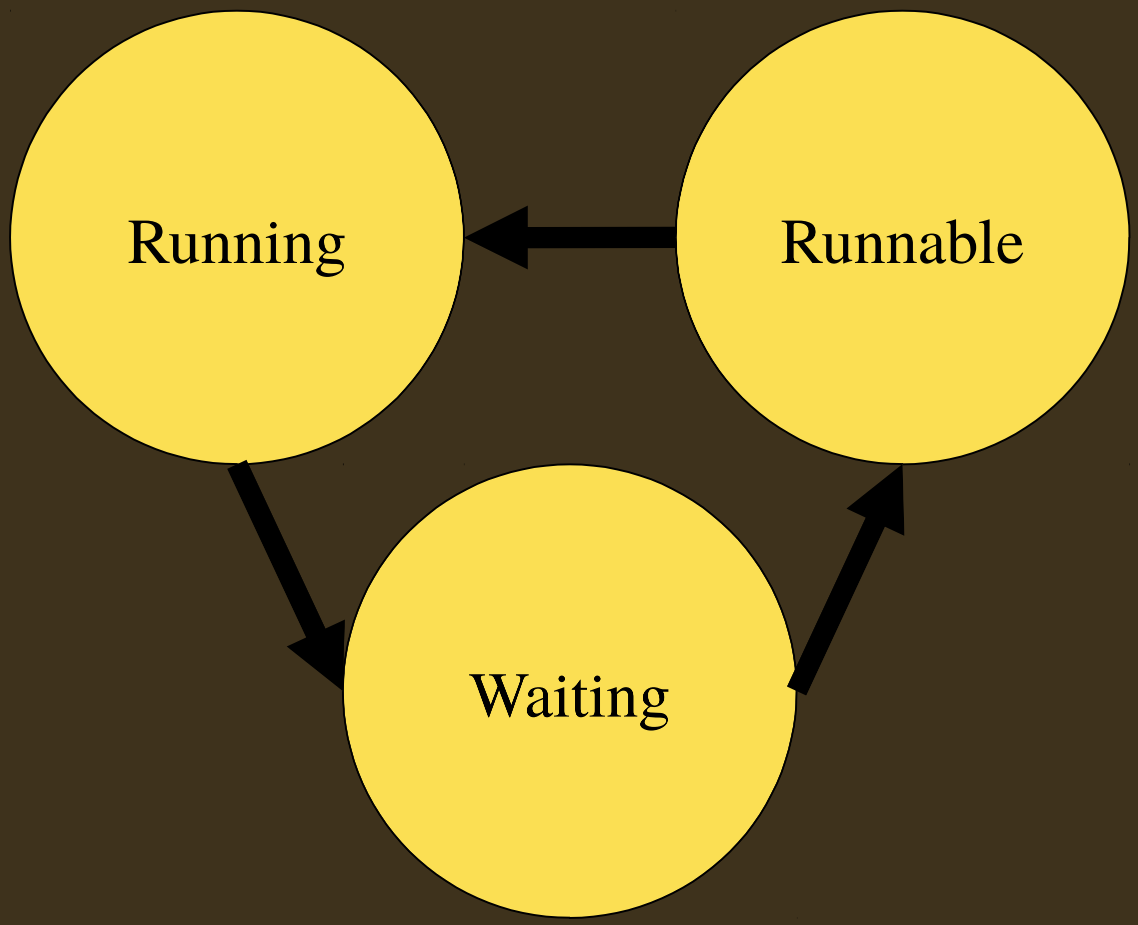
- Scheduling is determining what to do next
- Ease of modeling partition the system in tasks
- Tasks have requirements when they should be active
- Tasks can have a number of states

  Running – Actively using the CPU

  Runnable – Can run but other task is running

  Waiting – Is waiting for something to happen

Scheduling algorithms



Scheduling algorithms

- Priority based scheduling
- Each task has a priority (a number)
- Runnable task with the highest priority runs
- Task runs until it has to wait
- Only then the next highest priority task (which is runnable) is made running

- This is called FIFO aka First In First Out

Scheduling algorithms

- Preemptive round-robin
- Same a priority based scheduling
- After an external event the “next” task must be selected
- Select the next runnable task
  (”next” task has same priority as previous)
- Previous task is preempted

Scheduling algorithms

- Earliest deadline first (EDF)
- Dynamic priority algorithm
- Select the task closest to its deadline
- Each task has: execution time and period
- Not trivial to guard against overflow

Scheduling algorithms

- Rate monotonic
- Each task has a (static) period
- Task with shortest period runs first
- Well suited for hard real-time

Locking - Semaphores

- Invented by Edsger Dijkstra
- Operations are called P() and V()
- P() waits until the semaphore is acquired
- V() releases the semaphore
- No notion of a task
- Can protect shared data
- Can be used as inter-task signal
- Ideal source for deadlocks

Locking - Mutexes

- A lock with ownership
- Operations are lock() and unlock()
- Only the one using lock() can unlock()
- Easier to use than semaphores
- Typically used by shared data

Locking - Spinlocks

- ”Wait until a bit is set”
- Compare and exchange atomic instruction
- Infinite loop (ugh…)
- But should be very light weight
- Only a few instructions
- Should loop only a a few times
- Ideally suited between interrupts and tasks

No memory allocation?

- Calculate maximum memory use beforehand
  →No need for an allocator.
- Allocators operator on a shared pool of available memory.
  →Tasks may block competing for allocation.
- Could make task private “sub-pools”
- Must still be able to satisfy maximum memory usage.

Notable real-time kernels

- Many, many real-time kernels
  (more than Web framework?)
- Most of these are not really kernels
- Better classify them as executives
- Why not kernels?
- Lots of additional infrastructure is missing
  (e.g. device drivers, file systems, networking)

Notable real-time kernels

VxWorks, RTEMS, Micrium, uC/OS III. FreeRTOS, CMX, Windows CE, ThreadX, Arm Mbed, GHS Integrity, eCos, ITRON, Zephyr, Nucleus, OSEK, pSOS, (Linux-RT), and so on…

Notable real-time kernels - VxWorks

- Wind River Systems
- From the early 1980s, so quite old
- Used in aerospace, automotive, medical, consumer and so on
- Offers a large number of sub-systems for board support, file systems, networking, etc.

Notable real-time kernels - VxWorks

- Scheduling: preemptive round-robin
- Locking: semaphores (counting and binary)
- Also spinlocks
- Memory protection: MMU support

Notable real-time kernels - VxWorks

- Famous NASA Pathfinder bug
  Priority inversion problem
  1. A low priority task
     grabs a shared resource
  2. A high priority task
     blocks waiting for the shared resource
  3. A medium priority task
     preempts the low priority task
  4. High priority task not making progress

Notable real-time kernels - VxWorks

- Was fixed by enabling priority inversion
- Low priority task gets priority of high priority task
- Cannot be preempted by medium priority task
- Uploaded in the Pathfinder on Mars (How's that for a software update?)

Notable real-time kernels - FreeRTOS

- Originally Real Time Engineers Ltd.
- Now owned by Amazon

- SafeRTOS certified for safety critical applications
- Wittenstein High Integrity Systems

- Has some support for networking, FAT file systems.

Notable real-time kernels - FreeRTOS

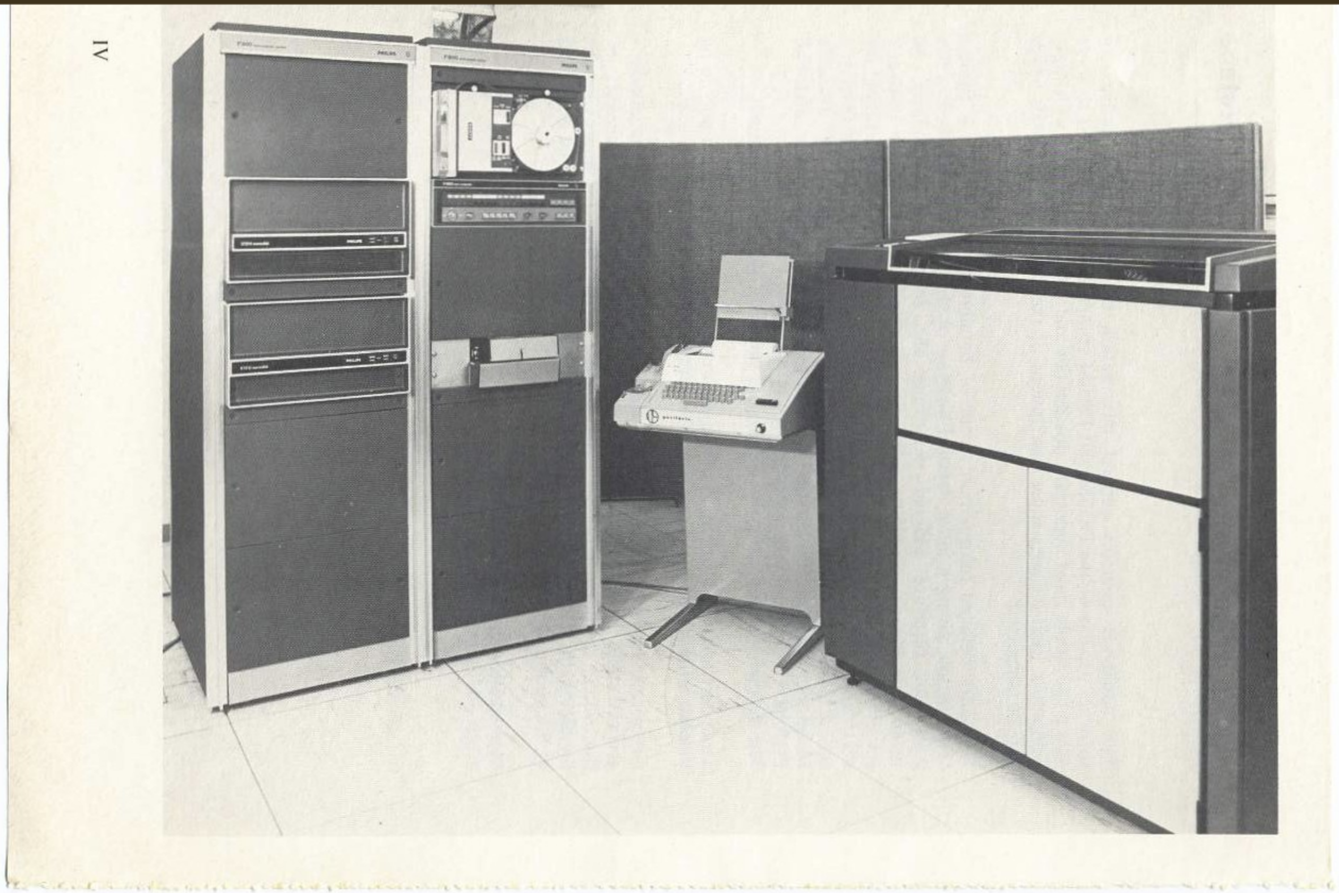
- Very popular with MCU developers
- Open source, MIT ”license”
- An executive, not an operating system
- Little infrastructure support

Notable real-time kernels - FreeRTOS

- Scheduling: FIFO, (preemptive) round-robin
- Locking: semaphores (counting and binary), mutexes
- No MMU support (SafeRTOS does)
- IPC by messages

Real-time: Then and Now

Real-time then: Philips P800 Series



End 1960s, early 1970s

Real-time then: Philips P800 Series

(With a real lock and key to turn the system on and off)

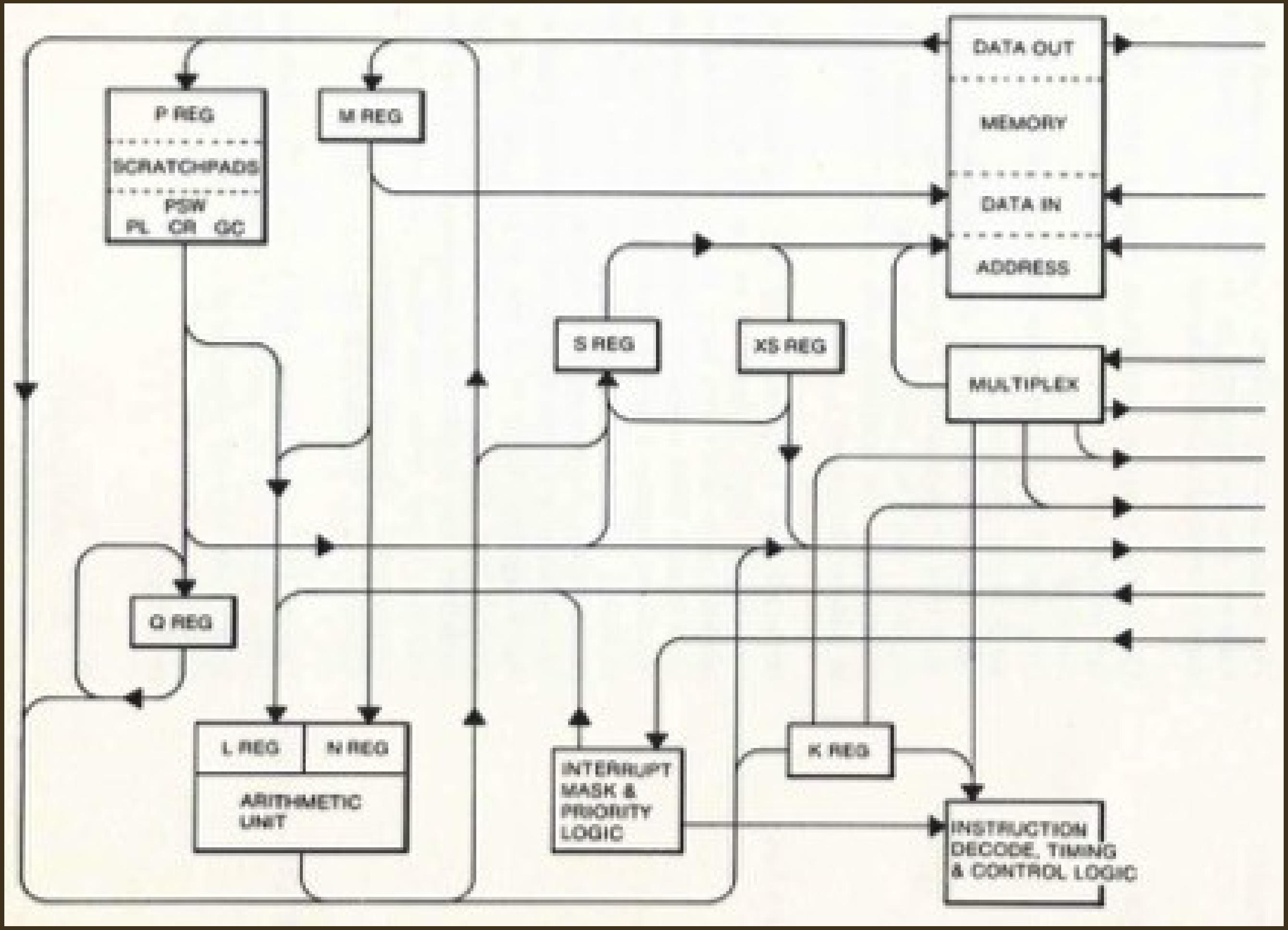
Real-time then: Philips P800 Series

- Mini-computers (16bit) systems
- Industrial and scientific applications (controls, data acquisition, etc.)
- Featured
  - Real-time clock
  - Hardware floating point
  - Memory Management Unit
  - Up to 63 interrupts

Real-time then: Philips P800 Series

- Specifications
  - 4K to 32K (16 bit) words
  - Clock cycle 1.2µS (later 840ns)
  - 16 General Purpose Registers
  - 98 Instructions
  - Discrete TTL ICs

Real-time then : Philips P800 Series



Real-time then : Philips P800 Series

- Instruction Timing Example
  Absolute Conditional Branch to Register
- Assembler ABREQ* A4
- Operation ((A4)) →P
- If branch taken
  - 2 memory cycles (2 x 1.2μs)
- If branch not taken
  - 1 logic cycle (720ns)
  - 1 memory cycle (1.2μs)

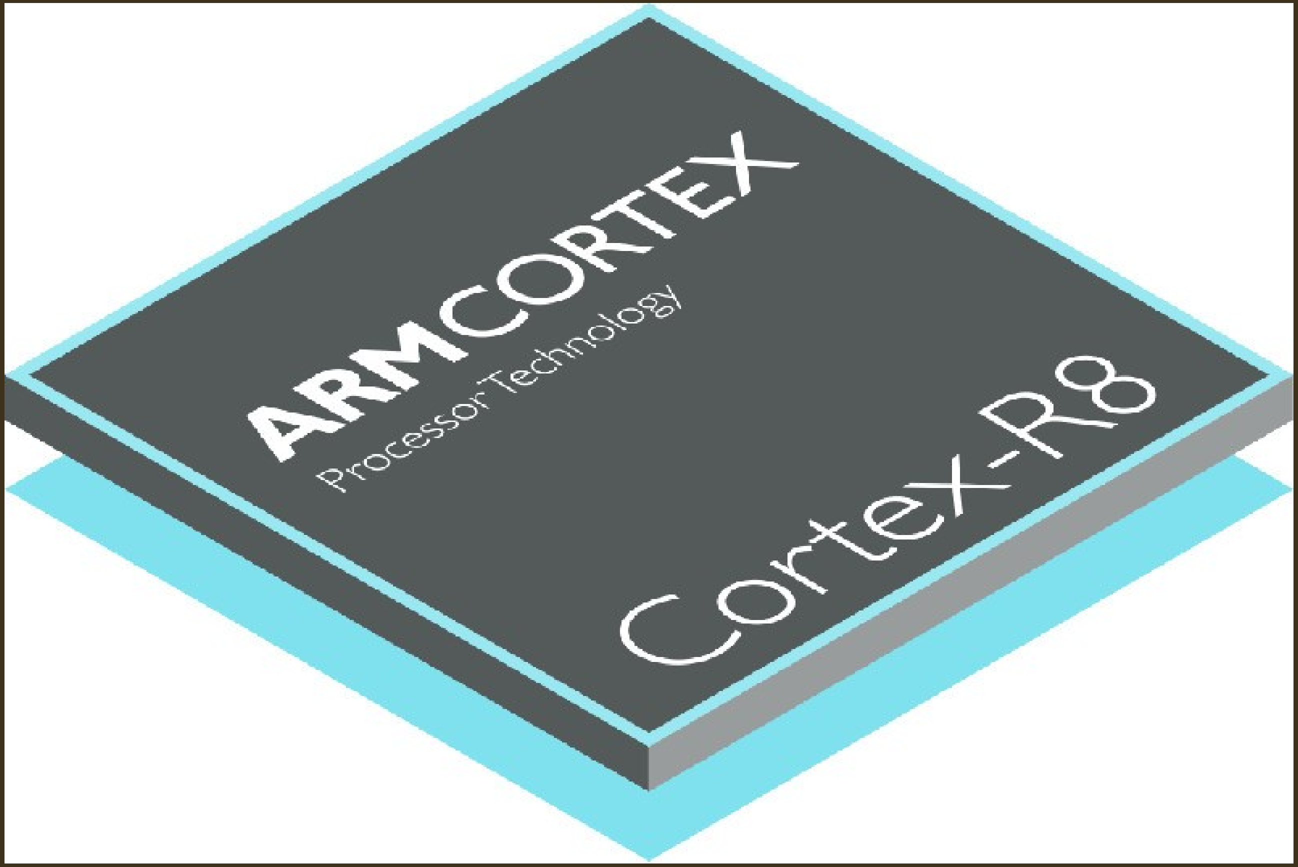
Real-time then: Philips P800 Series

- Disk Real Time Monitor (DRTM)
- Interrupt/hardware priorities: 48
- Software priorities: 15
- Scheduling is priority based FIFO
  ("the highest level active program always gets control until it is interrupted")

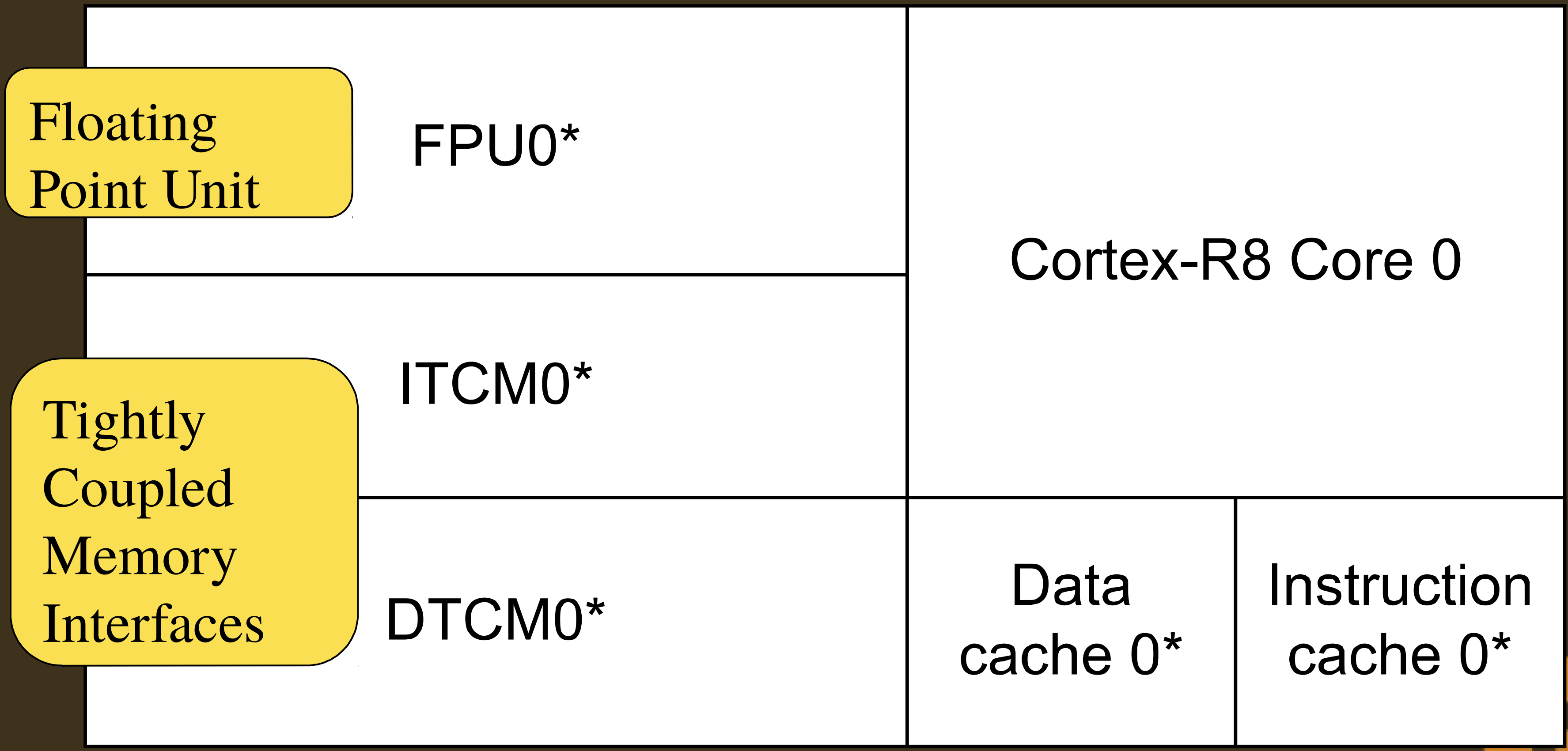
Real-time then: Philips P800 Series

- Program types
  - Memory resident – Always in memory
  - Read-only – In memory while running
  - Background – Lowest priority
  - Swappable – Written to disk after time slice
- Why easier?
  Only 32K words memory: easy to keep in your own memory

Real-time now: ARM Cortex-R8

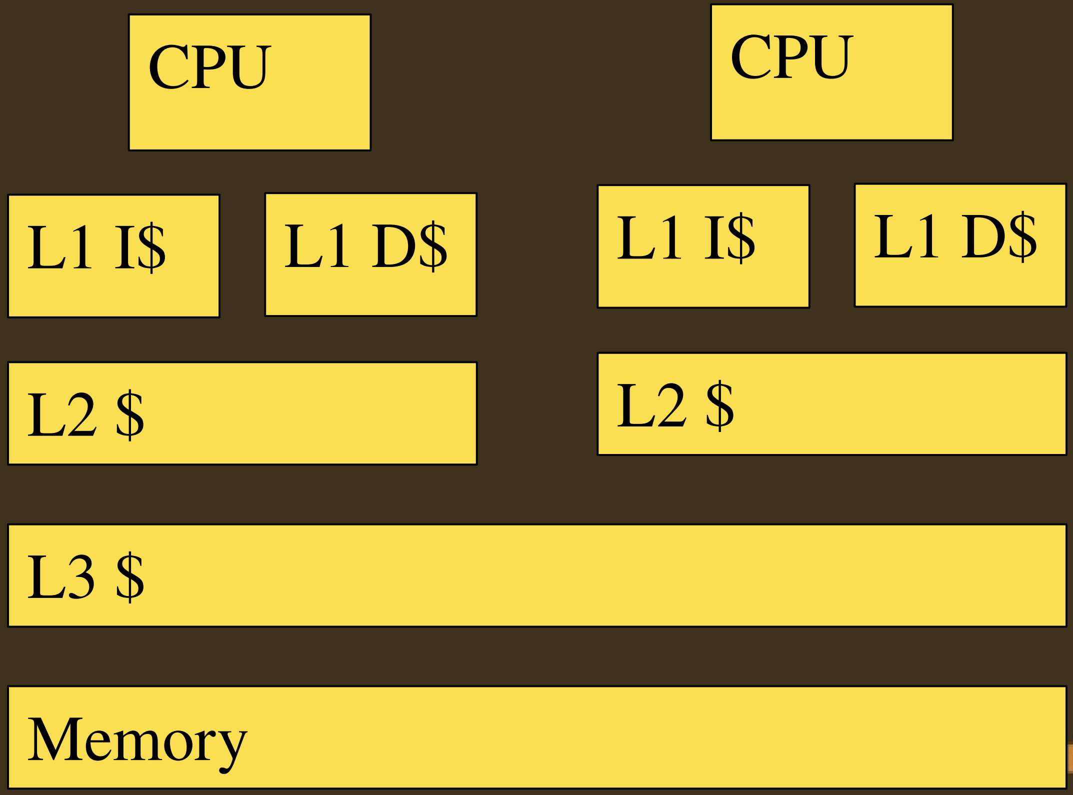


Real-time now: ARM Cortex-R8



Real-time now: ARM Cortex-R8

Cache hierarchy



Real-time now: ARM Cortex-R8

- Cache size
- Replacement strategy: Write back
- Fill strategy: critical word first
- Device DMA: memory and cache disageree
  Must flush (parts of) the cache

- All this introduces a lot of latency
- Hard to predict timing behavior

Real-time now: ARM Cortex-R8

- Pipelines get deeper
- Cortex-A8 13 stage (integer)
- May need to flush when branching
- Branch Target Buffer to predict where a branch will land

- Branches cause latency

Modern Hardware: PCI/PCIe

- A clock synchronization using CANbus
- New (x86) CPU

  Twice as fast as the previous ("old") one

  Memory four times as fast
- Problem

  Old system clock accuracy is ~1μS

  New system up to 60μS and more!

Modern Hardware: PCI/PCIe

- Hardware

  PCI card with CANbus

- Old system did not have PCIe

- Question: are the clocks stable?
- Answer yes: very much so

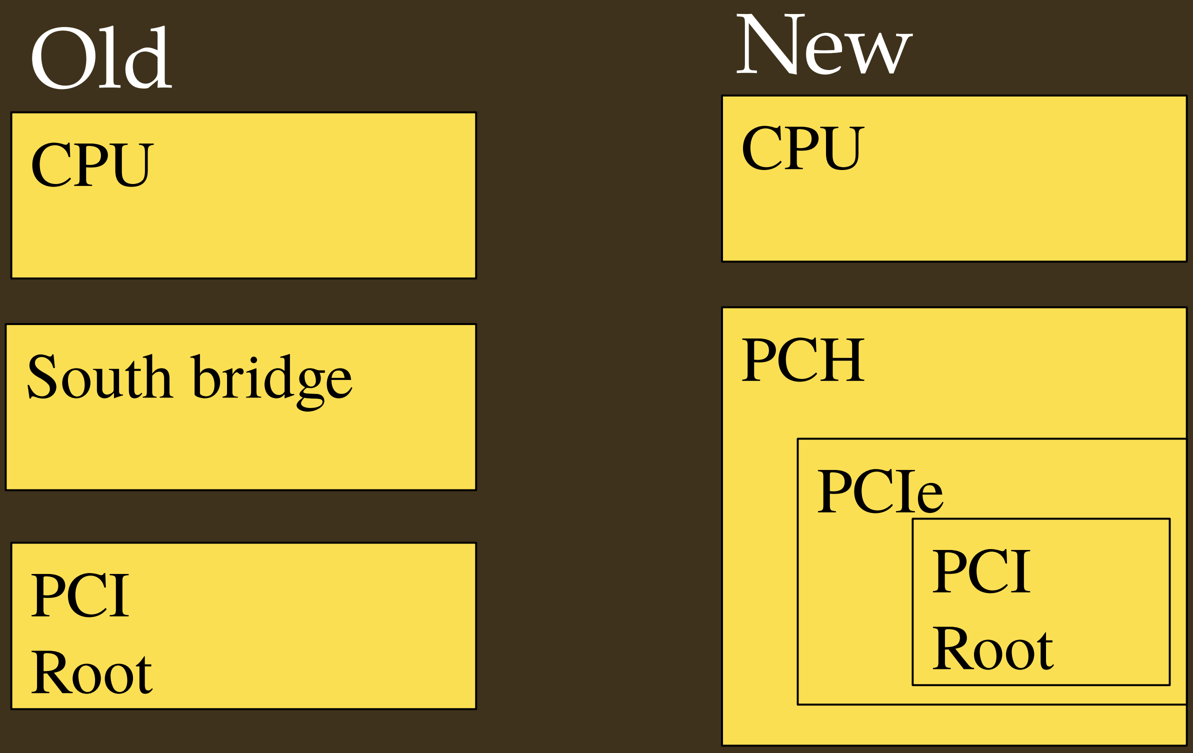
Modern Hardware: PCI/PCIe

- Experiment
- How long does a read to PCI memory take?

- Old system ~1.4µs
- New system ~3.7µs

- What is going on?

Modern Hardware: PCI/PCIe



Modern Hardware: PCI/PCIe

- Modern Peripheral Controller Hubs (PCH) have integrated PCIe root
- PCIe is compatible with PCI 2.3
- For ease of design PCI in PCIe
- Latency doubles!

- (Just because you ask: tweaking PCIe buffering does not help)

HANS
ZUIDAM
ADVIES