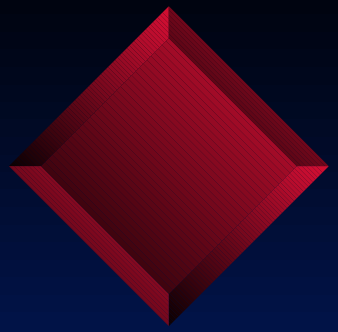


LIBAPI - OS

Rob Vawter March 1996

PlayStation™

OVERVIEW

KERNEL

INTERRUPTS

EVENTS

TIMERS

CALLBACKS

MODULE LOADING

PlayStation™

Overview

- System Designation File
- RAM Layout
- Memory Map
- Caches

Rob Vawter March 1996

PlayStation™

SYSTEM DESIGNATION FILE

- SYSTEM.CNF file
  - Format
    - Must be in <KEYWORD> = <CONTENT> format
    - A " " must be inserted on either side of "="
    - Only uppercase characters may be used

PlayStation™

SYSTEM DESIGNATION FILE

Contents

BOOT = cdrom:\SLUS_123.45;1

device name:\Product number; version

TCB = 4*

Number of task control blocks\possible threads

EVENT = 10*

Number of possible events (in hex)

STACK = 801ffff0

Stack pointer

*(TCB x 192) + (Event x 28) + 52 < 4096

Rob Vawter March 1996

PlayStation™

RAM LAYOUT

Overriding RAM and stack size defaults

- Using variables
  - static int _stacksize = 0x00002000 (8K stack)
  - static int _ramsize = 0x00200000 (2 MB RAM)
- Using linker
  - 2mbyte.obj
    - configure with 2 MB RAM
  - none2.obj
    - start code with no heap setup or data segment clearing

TM

RAM LAYOUT

- User Controlled Memory Allocation
  - Locating boundaries for memory allocation
    - _heapbase defines heap start
    - _heapsize defines heap size

RAM LAYOUT

R3000 defines 4 virtual memory segments

kseg0

Virtual address 0x80000000 to 0x9FFFFFFF

kseg1

Kernel ROM mapped to top

Virtual address 0xA0000000 to 0xBFFFFFFF

kseg2

Not used by PlayStation

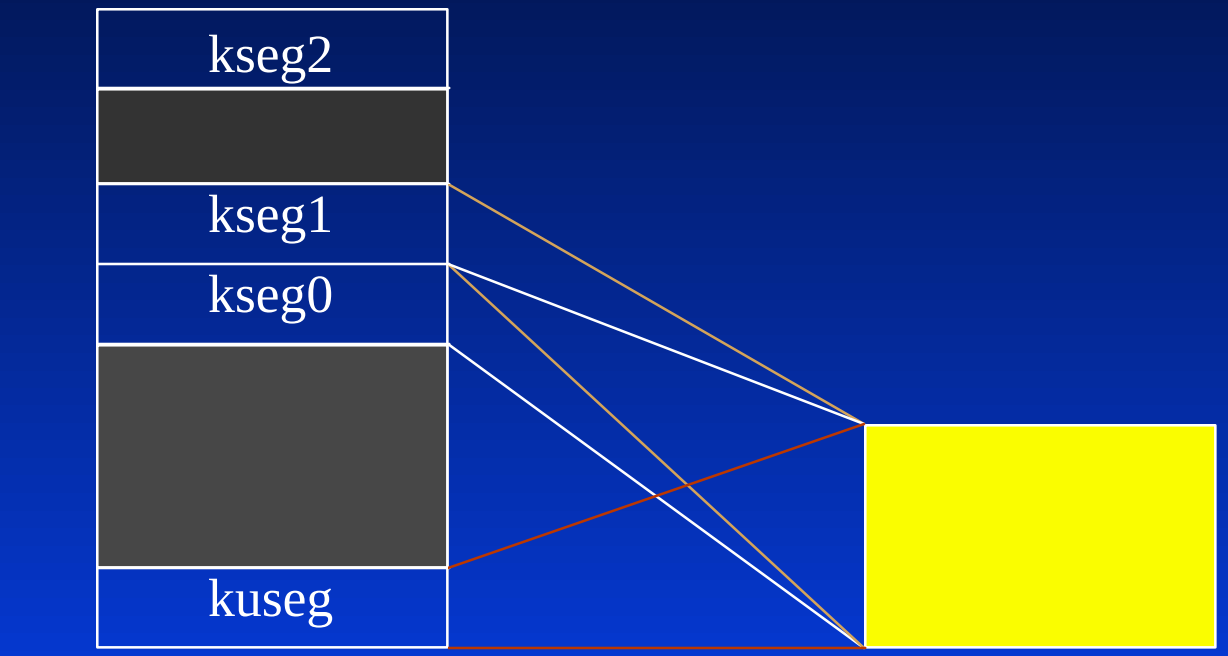
kuseg

Since PlayStation has no virtual memory, kuseg and kseg0 effectively the same

PlayStation™

RAM LAYOUT

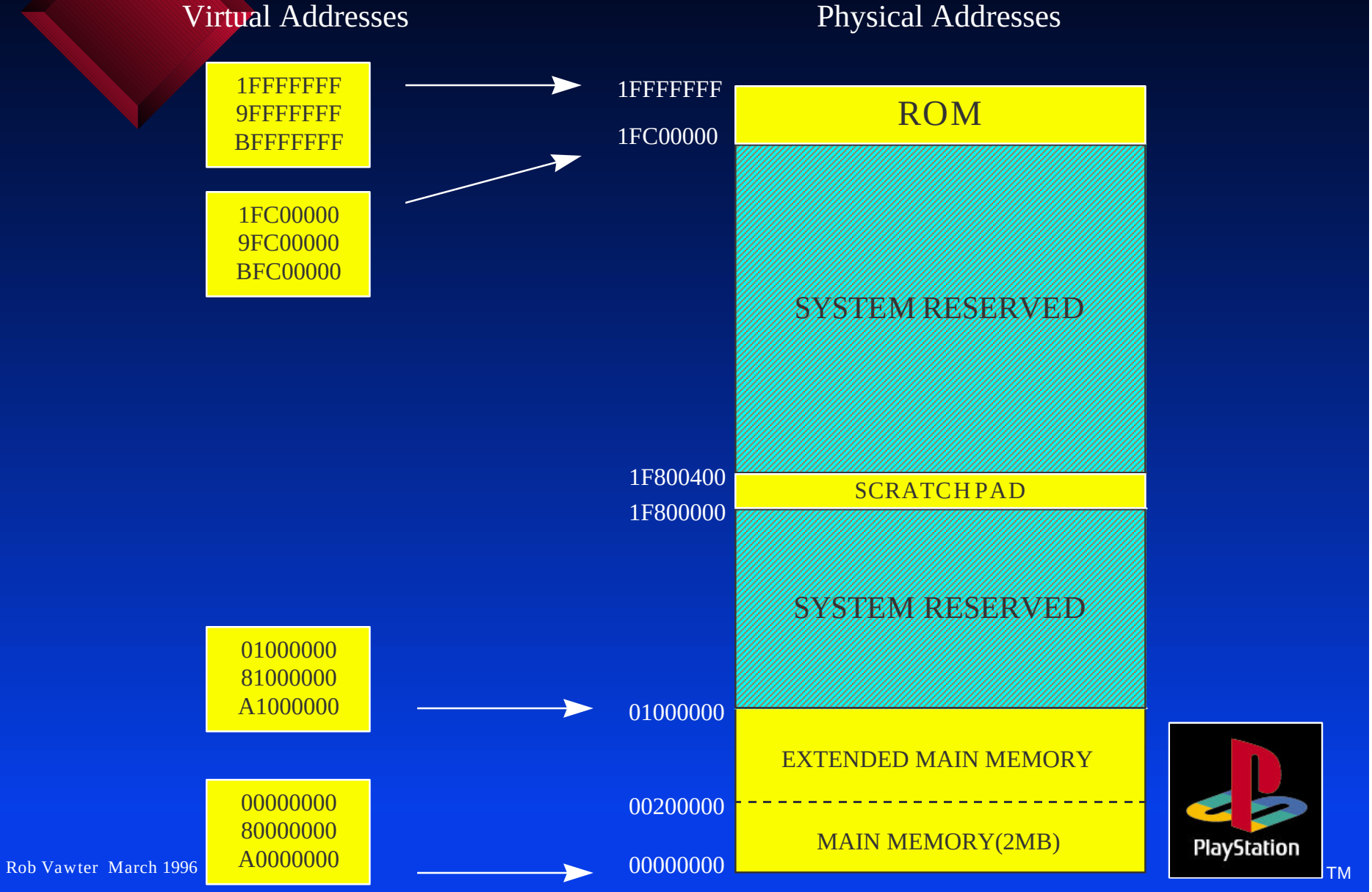
R3000 segment architechture



Rob Vawter March 1996

PlayStation™

MEMORY MAP



Rob Vawter March 1996

PlayStation™

KERNEL RAM MAP

64K at bottom of main RAM

0 x 80 00 01 00	Table of Tables
0 x 80 00 01 80	Boot file arguments
0 x 80 00 02 00	System reserved
0 x 80 00 05 00	RAM kernel code and data
0 x 80 00 90 00	ROM kernel code and data
0 x 80 00 e0 00	Kernel heap
0 x 80 01 00 00	User code and data start

CACHE

- I-Cache
  - Kernel high speed processes use I-cache
  - Optimization
    - Use small loops to stay within 4K boundary
    - Sort map file by address to verify 4K fit
      - Use dumpsym to get complete symbol list
      - Sort list by address to determine sizes
      - Check that lower 12 bits don’t overlap

PlayStation™

CACHE

Scratch Pad Features

- Strictly for programmer use
- 5-6 times faster than main RAM
- Remember to stay within 1K boundary

Scratch Pad Uses

- Put local variables on scratch pad
- Put passed parameters on scratch pad
- Put stack on scratch pad

INTERRUPTS

11 Device Interrupts

- 3 Timers
- GPU
- VBlank
- CDROM
- SPU
- 2 SIO
- DMAC
- PIO

PlayStation™

INTERRUPTS

How R3000 Handles Interrupts

- R3000 has only 1 interrupt which all devices use
- Device interrupts disabled automatically at start of handler
- While specific device interrupt is being handled, other interrupts queue up
- Same device interrupts will be ignored
- Callbacks must be short to avoid interrupt misses

EVENTS

Handlers for exceptions and interrupts

Function Overview

- OpenEvent()
  - Sets up callback
  - Remember to stay within parameters set in SYSTEM.CNF file
  - Must be executed in a critical section
  - Args: cause descriptor, event type, mode, handler function
  - Returns an event descriptor

EVENTS

Function Overview (cont.)

- Critical Sections - section in which no interrupts can occur
- EnterCritical Section()
  - Inhibits interrupts
  - Occurs at kenel startup
- ExitCriticalSection()
  - Enables interrupts

Function Overview (cont.)

- Main Flow Destruction
  - ExitCriticalSection() and EnterCriticalSection() destroy interrupt context
  - Cannot be used during event handler or callback without destruction of main flow

Function Overview (cont.)

Avoiding main flow destruction

Use SwEnterCriticalSection() and SwExitCriticalSection() instead

EVENTS

Function Overview (cont.)

- EnableEvent()
  - Starts event handling
  - Changes event condition from wait to active
- DisableEvent()
  - Stops event handling
  - Changes event condition from active or already to wait

EVENTS

Function Overview (cont.)

- WaitEvent()
  - Waits until descriptor event occurs
  - No interrupt called; value polled only
  - Restores event to previous state
- TestEvent()
  - Checks if descriptor event has already occurred
  - No interrupt context called; value polled only
  - Restores event to previous state

EVENTS

Function Overview (cont.)

- DeliverEvent()
  - Changes event condition from active to already
- UnDeliverEvent()
  - Clears event; changes condition from already to active
- CloseEvent()
  - Releases callback setup by OpenEvent()
  - Must be executed in critical section

TIMERS

TIMERS	MACRO	FREQUENCY
SYSTEM CLOCK	RCntCNT0	33.8688 MHz†
SYSTEM CLOCK	RCntCNT1	33.8688 MHz†
SYSTEM CLOCK	RCntCNT2	33.8688 MHz*
VBLANK	RCntCNT3	60 Hz

* or 33.8688/8, depending on mode

† Note: Pixel Clock and HBlank timer were found to be unreliable. However, they were not actually replaced. Use mode setting RCntMdSC to use system clock run counters for RCntCNT0 and RCntCNT1.

PlayStation™

TIMERS

2 Methods of Using Timers

- Interrupt
  - Requires callback function
- Polling
  - Use for slower event handler processes so that interrupt not missed

PlayStation™

Timer Function Overview

- SetRCnt() sets counter target
  - Target value is 16 bits for RCntCNT0-2
  - Target value fixed at 1for Vblank
  - Interrupt generated, counter reset when target reached
- StartRCnt()
  - Allows interrupts
- StopRCnt()
  - Masks interrupts

TM

Function Overview (cont.)

- GetRCnt( )
  - Returns counter value for polling
- ResetRCnt()
  - Resets counter to zero
- ChangeClearRCnt()
  - Upcoming new function
  - Clears default interrupt handler (libapi)
  - Allows user to set up own callbacks (libetc)

PlayStation™

Profiling Using System Timers

- Caution - Avoid counter reset, stay within 16 bit value

TM

CALLBACKS

Using Callbacks

- ResetCallback() initializes local stack
- Some initialization functions contain ResetCallback()
  - ResetGraph(0), DecDTReset(0), CdInit(0), SsInit(0), PadInit(0)
  - Subsequent calls to ResetCallback() are ignored
- StopCallback() halts callbacks
- Keep callbacks short

MODULE LOADING

LoadExec

- _96_remove()
- _96_init()
- Must link in none2.obj to ensure proper return to parent program

MODULE LOADING

- Threading
  - Not pre-emptive multi-tasking
  - Task control blocks
    - Contain information necessary to change threads during interrupts
    - Top block is default execute thread
    - Accessed via the Table of System Tables (ToT)

MODULE LOADING

- Threading Function Overview
  - OpenTh() returns a thread descriptor
    - Specify thread's address, gp, sp
    - Separate stack for each new thread
  - CloseTh() stops thread, releases TCB for another thread to use
  - ChangeTh() swaps to another thread
    - Cannot be used during interrupts

MODULE LOADING

- Overlays
  - Relocatable Code
    - Within each module, jumps are relative
    - Function references are absolute
    - Must build a dynamic linker\loader