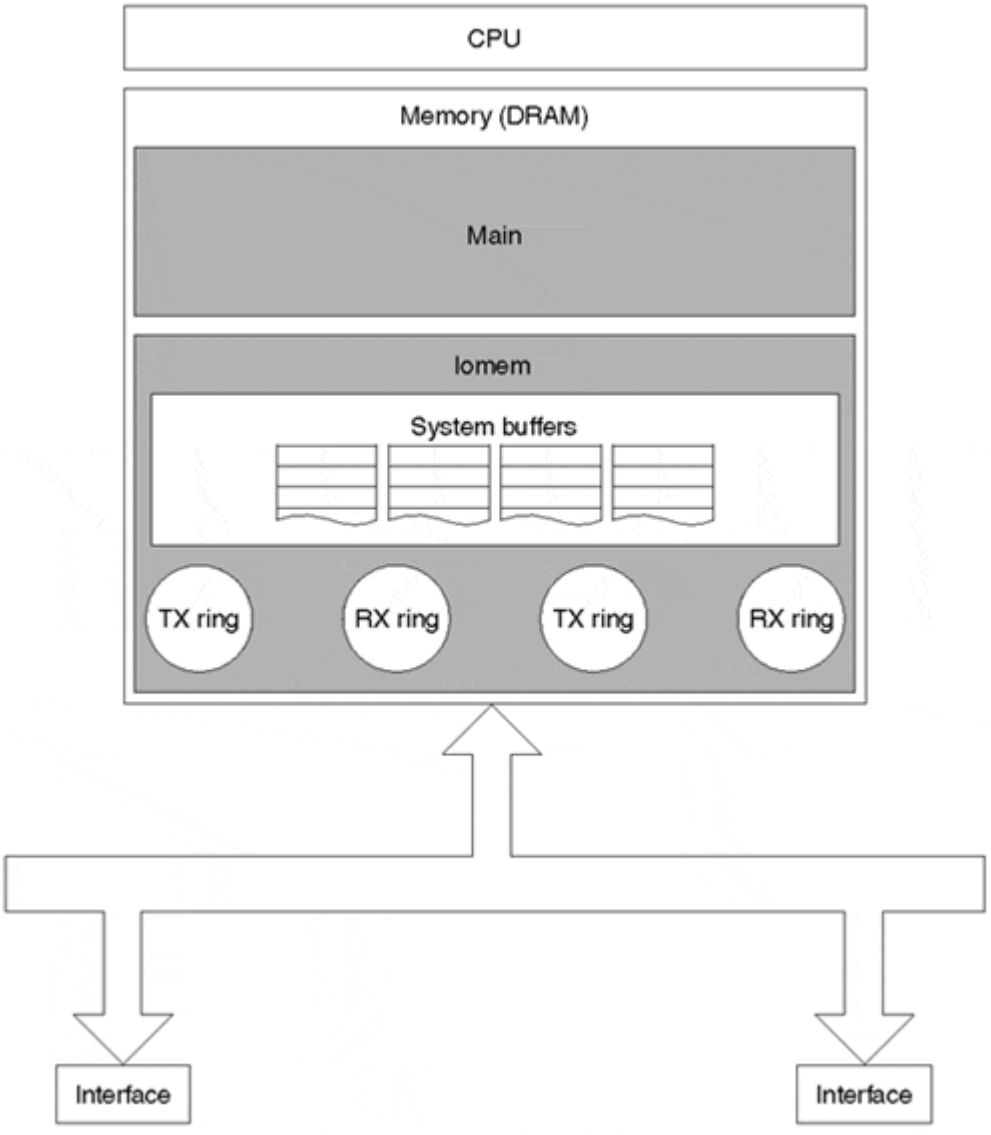
Hardware Architecture for Shared Memory Routers

Let's examine the hardware architecture on these shared memory platforms in some detail. We start with an overview of the generic shared memory architecture in Figure 3-1, and then mention some of the unique hardware features of platforms within this group.

Figure 3-1. Shared Memory Router Architecture



You can see the overall architecture is fairly basic, consisting of just three major components: the processor (CPU), memory (DRAM), and interface controllers, with the processor and the interface controllers connected to the memory via data busses. Figure 3-1 also shows memory is divided into logical regions, which we'll examine later.

CPU

The processor in a shared memory router is truly the brains behind the entire show; it runs IOS and switches packets. The processor is responsible for executing all the switching methods supported on these platforms;

there are no offload processors involved.

The type of processor used depends on the platform. For example, the Cisco 1600 and 2500 series use a Motorola 68000 series CPU while the 4500 and 4700 series use a MIPS RISC CPU. You can determine the specific type of processor by looking at the output of the **show version** command. Example 3-1 shows the **show version** output from a Cisco 1600 router. Example 3-2 shows the **show version** output from a Cisco 4500 router.

Example 3-1. show version Output from a Cisco 1600 Router

router-1600>**show version** Cisco Internetwork Operating System Software …. cisco 1604 **(68360) processor** (revision C) with 17920K/512K bytes of memory. Processor board id 05385389, with hardware revision 00972006 ….

Example 3-2. show version Output from a Cisco 4500 Router

router-4500#**show version** Cisco Internetwork Operating System Software …. cisco 4500 **(R4K) processor** (revision B) with 16384K/16384K bytes of memory. Processor board id 01657190 R4600 processor, Implementation 32, Revision 2.0 ….

Memory

Shared memory platforms use dynamic random access memory (DRAM) to hold most of the data in the router. DRAM contains all the routing tables, the caches, the IOS data, the packet buffers, and, on many systems, the IOS code itself.

IOS divides the available DRAM into two logical memory classes: *Local* and *Iomem* (I/O memory). In most cases, Local memory is placed into the **main** region and I/O memory is placed into the **iomem**region, as illustrated by the **show region** command output in Example 3-3.

Example 3-3. DRAM Memory Regions Revealed in show region Command Output

Router-4500#**show region** Region Manager: Start End Size(b) Class Media Name 0x30000000 0x30FFFFFF 16777216 Flash R/O flash 0x38000000 0x383FFFFF 4194304 Flash R/O bootflash 0x40000000 0x40FFFFFF 16777216 Iomem R/W iomem 0x60000000 0x61FFFFFF 33554432 Local R/W main 0x600088A0 0x607229BF 7446816 IText R/O main:text 0x60726000 0x6096506F 2355312 idata R/W main:data 0x60965070 0x609DB1CF 483680 IBss R/W main:bss 0x609DB1D0 0x61FFFFFF 23219760 Local R/W main:mainheap 0x80000000 0x81FFFFFF 33554432 Local R/W main:(main_k0) 0x88000000 0x88FFFFFF 16777216 Iomem R/W iomem:(iomem_k0) 0xA0000000 0xA1FFFFFF 33554432 Local R/W main:(main_k1) 0xA8000000 0xA8FFFFFF 16777216 Iomem R/W iomem:(iomem_k1)

The sizes of these two regions also are reflected in the output of **show version,** as illustrated in Example 3-4.

Example 3-4. show version Command Output Also Reveals Memory Region Sizes

Cisco Internetwork Operating System Software IOS (tm) 4500 Software (C4500-J-M), Version 11.1(24), RELEASE SOFTWARE (fc1) …. cisco 4500 (R4K) processor (revision E) with 32768K/16384K bytes of memory. Processor board id 09337282 ….

The number before the slash is the amount of Local memory present (32,768 Kb), and the number after the slash is the amount of I/O memory present (16,384 Kb). The total DRAM memory present is the sum of these two numbers (49,152 Kb). For the system in Example 3-3 and Example 3-4, the 32,768 Kb of Local memory is assigned to the region named **main** and the 16,384 Kb of I/O memory is assigned to the region named **iomem**.

On shared memory platforms, the **main** region is used for storing routing tables, caches, general IOS data structures, and often the IOS runtime code—but not for storing packet buffers. Packet buffers are stored in the **iomem** region, which often is called *shared memory* because it's shared between the processor and the media interfaces.

On many systems, the DRAM used for Local memory is physically separate from the DRAM used for I/O memory—usually two different banks. On those systems, IOS simply assigns all the available memory in a bank to the appropriate memory regions, one bank for **iomem** and one bank for **main**. For the Cisco 1600 and 2500, however, I/O memory and Local memory are both allocated from the same DRAM bank. The amount of I/O memory carved from the installed DRAM depends on the total memory installed:

- **1MB—**

  512 Kb of memory is used for I/O

- **2MB—**

  1 MB of memory is used for I/O

- **4MB and above—**

  2 MB of memory is used for I/O

Location of IOS Runtime Code on Shared Memory Systems

Some IOS shared memory platforms—in particular, the 1600 series and the 2500 series routers—can run IOS directly from Flash memory. If a system is running IOS from Flash, it doesn't use the **main** memory region for the IOS runtime code. Instead, it stores the runtime code in a region named **flash**, as demonstrated in Example 3-5.

Example 3-5. show region Output for a Run-from-Flash System

Router-2500#**show region** Region Manager: Start End Size(b) Class Media Name 0x00000000 0x007FFFFF 8388608 Local R/W main 0x00001000 0x0001922F 98864 idata R/W main:data 0x00019230 0x000666B3 316548 IBss R/W main:bss 0x000666B4 0x007FEFFF 7965004 Local R/W main:heap 0x007FF000 0x007FFFFF 4096 Local R/W main:flhlog 0x00800000 0x009FFFFF 2097152 Iomem R/W iomem 0x03000000 0x03FFFFFF 16777216 Flash R/O flash 0x0304033C 0x037A7D37 7764476 IText R/O flash:text

In Chapter 1, "Fundamental IOS Software Architecture," you saw that the *IText* class is where the IOS runtime code is stored. In Example 3-5, the IText memory is assigned to a subregion of **flash** called **flash:text**, indicating the system is running IOS from Flash.

You also can determine whether a router is running IOS from Flash or Local memory by checking the image's filename. The last one or two letters of the filename indicate whether the image is *relocatable* or *run-from-DRAM*. The run-from-DRAM images run in Local memory, while the relocatable images run from Flash memory.

The **show version** output in Example 3-6 is taken from a Cisco 2500, which runs IOS from Flash.

Example 3-6. Flash-based IOS Image

router-2500>**show version** Cisco Internetwork Operating System Software IOS (tm) 2500 Software (C2500-JS-L), Version 12.0(7.3)T, MAINTENANCE INTERIM SE Copyright (c) 1986-1999 by cisco Systems, Inc. ….

The last letter in the image name (**L**) means this image is relocatable, and therefore is running from Flash. On the other hand, if the image runs from Local memory (DRAM), the image name ends in either **M** or **MZ**, as you can see in the output of **show version**in Example 3-7.

Example 3-7. DRAM-based IOS Image

Router-4500>**show version** Cisco Internetwork Operating System Software IOS (tm) 4500 Software (C4500-P-M), Version 11.2(18)P, RELEASE SOFTWARE (fc1) Copyright (c) 1986-1999 by cisco Systems,

Inc. ….

Memory Pools

IOS creates two memory pools to manage allocation and de-allocation of memory within the DRAM regions. These two pools are named *Processor*and *I/O*, as demonstrated in the **show memory** output in Example 3-8.

Example 3-8. *show memory* Output Reveals Memory Pools

router-2500#**show memory** Head Total(b) Used(b) Free(b) Lowest(b) Largest(b) Processor 3BB08 16528632 878596 15650036 15564280 15630436 I/O 4000000 2097152 473468 1623684 1603060 1623232

The **Processor** memory pool is created from memory in the **main:heap** subregion of Local memory and the **I/O** pool is created from memory in the **iomem** region of I/O memory. The size of the **I/O** pool (the number in the **Total** column in Example 3-8) correlates closely to the size of **iomem**. However, the size of the **Processor** pool is always less than the size of the **main** memory region. The **Processor** pool gets only a subset of the memory in the **main** region because the remainder must be reserved for the IOS data, the BSS segments, and, on many platforms, the IOS runtime image itself.

> **NOTE**
>
> Have you ever wondered why this collection of routers is called shared memory routers? All routers discussed in this book have shared memory, so why are these particular routers singled out? What's unique about these systems is not that they have shared memory, but that they have only one region of shared memory (I/O memory) and it's shared between a single main processor and all the interface controllers. In addition, that same memory is used for all packet switching—there's no data copied between regions to pass a packet from fast to process switching like there is on other platforms.

Interface Controllers

Interface controllers are responsible for transferring packets to and from the physical media. They have their own processors, called media controllers, but do not perform any switching or IOS processing operations. Interface controllers perform media control operations and move packets between I/O memory and the network media. Depending on the platform, interface controllers can be implemented as removable port modules or as components on the main system board.

< BACK Make Note | Bookmark CONTINUE >

Index terms contained in this section