Multimedia Tools and Applications, 27, 351–366, 2005

Zikimi: A Case Study in Micro Kernel Design for Multimedia Applications

SANG-YEOB LEE* zikimi@syu.ac.kr
Dept. of Management Information System, Samyook University, Seoul, Korea

YOUJIP WON† yjwon@ece.hanyang.ac.kr
WHOI-YUL KIM wykim@hanyang.ac.kr
School of Electrical and Computer Engineering, Hanyang University, Seoul, Korea

Abstract. Due to recent rapid deployment of Internet Appliances and PostPC products, the importance of developing lightweight embedded operating system is being emphasized more. In this article, we like to present the details of design and implementation experience of low cost embedded system, Zikimi, for multimedia data processing. We use the skeleton of existing Linux operating system and develop a micro-kernel to perform a number of specific tasks efficiently and effectively. Internet Appliances and PostPC products usually have very limited amount of hardware resources to execute very specific tasks. We carefully analyze the system requirement of multimedia processing device. We remove the unnecessary features, e.g. virtual memory, multitasking, a number of different file systems, and etc. The salient features of Zikimi micro kernel are (i) linear memory system and (ii) user level control of I/O device. The result of performance experiment shows that LMS (linear memory system) of Zikimi micro kernel achieves significant performance improvement on memory allocation against legacy virtual memory management system of Linux. By exploiting the computational capability of graphics processor and its local memory, we achieve 2.5 times increase in video processing speed.

Keywords: micro-kernel, multimedia, operating system, Linux

1. Introduction

1.1. Motivation

Advancement of low cost and high performance embedded processors along with advances in memory and network technology accelerates the rapid deployment of Internet Appliances and PostPC products. The primary advantage of the Internet Appliances against general-purpose personal computer is its cost, size, and ease of use [4]. Different from general purpose desk-top computer, Internet Appliances is designed to perform specific task and thus it is possible to trim unnecessary hardware and software features. One of the dominant activities in these devices is accessing multimedia information on-line, e.g. watching TV, listening to music and etc. These activities can be done via general-purpose personal computer. However, general-purpose personal computer equipped with the commodity operating

*Funded by Faculty Research Institute Program 2001, Sahmyook University, Korea.
†Supported by KOSEF through Statistical Research Center for Complex Systems at Seoul National University.

system, i.e. with virtual memory management system, complex processor scheduling algorithm, and powerful file system may be too luxurious to be dedicated for this type of specific task. On the same token, there has been intense demand for the more economical system tailored for handling image and motion picture. In this paper, we share the details of design and implementation experience of state of art multimedia information processing system.

We detail three major issues related to the embedded system design. The first issue is about virtual memory management. Information devices, e.g. smart phone, PDA, WebTV, digital album, etc., usually have very limited amount of primary memory and are not equipped with secondary storage device. Thus, the system actually does not have to take the burden of complex system call in allocating and deallocating memory blocks from virtual address space. In this work, we devise an innovative way of handling memory allocation and deallocation without using system calls provided by operating system. The second issue is developing lightweight graphical user interface. Embedded system normally runs on very resource critical environment, and thus there is no room to accommodate legacy X-window like graphical user interface. GUI library of Zikimi system supports multi-window feature controlled by single thread. The elementary components consist of box, line and circle. The best feature of graphical user interface of Zikimi is its compactness, 16 Kbyte.

Zikimi system is currently embedded in WebTV and machine vision system. The purpose of this article is to publicize the technical details of Zikimi system including hardware organization, and micro-kernel based embedded Linux system. Micro-kernel portion of Zikimi is responsible for video memory processing, input/output port control, memory management, and application loading technique.

1.2. Related work

There have been a number of works addressing the issue of chip design for coding and decoding MPEG compressed stream [3, 6, 9]. These works are particularly focused on digital set top box environment. Embedded system for Internet Appliances or PostPC products normally works on very resource stringent environment compared to a general-purpose personal computer. Neither legacy X-window library nor rich set of Win32 APIs can be accommodated. Thus, it is mandatory to develop lightweight graphical user interface library dedicated for embedded system environment. There are number of lightweight GUI libraries for this purpose. Most of these libraries operate on top of commodity operating system [8, 10–12]. Handling multimedia data, particularly from the multimedia streaming operation's perspective requires interesting performance demand on the end terminal. To efficiently handle streaming workload, the operating system kernel needs to be properly optimized for streaming operation. For example, page replacement algorithm optimized for streaming workload is presented in [1]. Micro kernel based operating system has been widely used in the real-time environment [5, 7]. This is particularly because system call latency is relatively small and tightly bounded due to its small size kernel. Nemesis operating system adopts QoS aware page replacement scheme which can guarantee a certain level of QoS to multimedia application [5]. SPIN operating system [2] adopts link-time mechanisms to inexpensively export fine-grained interfaces to operating system services. This facilitates the development of domain specific API's for DBMS, multimedia server, web servers, etc.

The rest of the paper is organized as follows. In Section 1, we introduce the overall architecture of the system from the aspect of both software and hardware. Section 2 presents system overview to describe the hardware organization of the system. Section 3 presents the software organization of Zikimi system, which consists of trimmed out version of Linux operating system and Zikimi micro kernel. We also present the technical details of the video chip control, input/output port control, memory management, file management and application loading technique. Section 3 presents the state of art technology of displaying compressed image, e.g. JPG or MPEG compressed data, using low performance CPU. Section 4 briefly discusses the future work and concludes the paper.

2. System overview

2.1. System architecture

Figure 1 illustrates system architecture of Zikimi. Initial version of our system uses STPC processor. The STPC processor is based on a 32-bit 486 class processor block including CPU, FPU and 8 KB of L1 cache together with a 64-bit SDRAM controller, bus mastering IDE and PCI bus controller. It runs on 66 Mhz clock in 32 bit Protected mode. Analyzing the typical workload on WebTV, digital album and a number of Internet Appliances and PostPC products, we conclude that operating system level support for the complex multi-tasking feature is not necessary. The system is optimized to run single application process that executes a number of modules concurrently via multi-threading. Application program requests for service to Zikimi linux kernel and micro kernel. Zikimi system adopts the existing controller module from Linux kernel in controlling IDE and RS232C Port for HDD, CD-ROM, PCMCIA and RS232C. Processing multimedia streaming traffic in real-time is CPU intensive operation. STPC 66 Mhz processor does not deliver sufficient computing power for this purpose. To resolve this issue, Zikimi micro-kernel directly accesses video chip for the faster video processing. Zikimi micro kernel is also responsible for controlling sound device and network device. Details will be presented in Section 4.

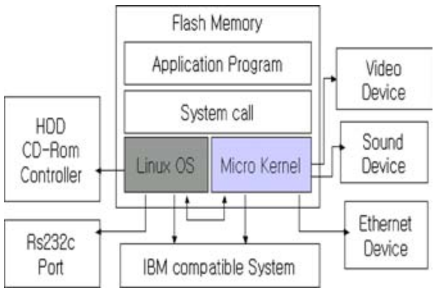


Figure 1. System overview.

2.2. *System boot-up*

When the system is powered on, the BIOS loads kernel of the embedded Linux onto 16 mega bytes main memory. It recognizes the flash memory as the hard disk. Operating system kernel initializes GDT (Global Descriptor Table) and LDT (Local Descriptor Table) and then searches other peripheral devices, e.g. additional hard disk drives and RS 232-C port. If any hard disk drive is detected in this phase, it is mounted as *hdb*. The Linux kernel then loads the Zikimi micro kernel. The micro kernel is allocated all remaining free memory space. Since no application has been running, Zikimi micro kernel is allocated physically contiguous memory chunk and can take full control of physical memory blocks. The micro kernel also initializes the registers in video chip to handle 3byte true color with high resolution. The video output is controlled via memory mapped I/O. Finally, the micro kernel initializes sound device and network device.

2.3. *System_call interface*

Zikimi micro kernel is responsible for controlling video chip, sound card, PCMCIA interface, memory management, file I/O and remote control input. Individual control functions are implemented via software interrupt handling routine running on 32 bit protected mode. Application program communicates with micro-kernel via `system_call()` interface. In most of the legacy operating systems, the application program invokes library routine and library routine requests the kernel for I/O services. Kernel is responsible for handling interrupt request. In our system, application directly invokes the software interrupt routine which enables us to achieve great deal of efficiency. Compared to developing library functions for controlling the device, our approach of using software interrupt may be less flexible and may not provide rich set of functionalities. However, the primary task of our system is decompressing the multimedia data blocks efficiently, which does not require wide variety of functionalities.

Figure 3 illustrtes the process of rendering bitmap to screen. The application program first initializes the memory segment with the bitmap to be displayed and then sets up the

```
/* Application */
system_call(1,arg);

/* System Call Handling Routine */
char* system_call(num,arg){
  arg_point=get_system_arg_point(num);
  set_system_argumentdata(arg_point);
  point= get_system_block(num);
  jmp_point(point);
  while(Is_systemcall_running()) sleep;
  return_code=get_return_block();
  return return_code;
}
```

Figure 2. Pseudo code for invoking software interrupt via `system_call ()` interface.

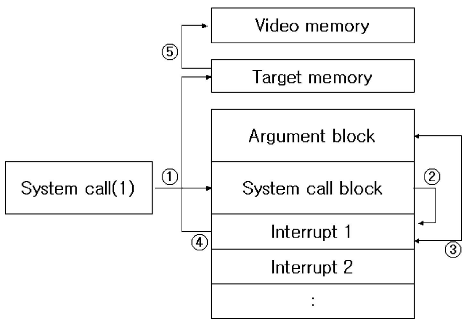


Figure 3. Video system call.

argument block. Then the application invokes `system_call ()` with proper argument. The argument of the system call isservice number, i.e. identification number of interrupt handling routine. When an application program invokes system call with service number, the kernel examines system call block and locates the respective interrupt handling routine. Interrupt handling routine runs on separate thread. Interrupt handling routine examines argument block which has been previously set by the application program, and then moves bit map image in source memory to video memory.

3. Hardware organization

Figure 4 illustrates the hardware organization. Our system is designed towards ×86 architecture compatibility and is able to run on any microprocessor with ×86 compatible system board. Currently, DigiAlbum uses STPC embedded microprocessor with video processing chip. It runs 66 Mhz and has ×86 compatible core. For faster data and control exchange,

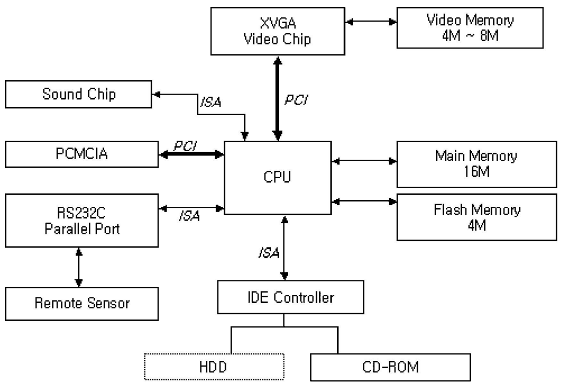


Figure 4. Hardware organization.

PCMCIA interface and video chip is connected to CPU via PCI bus instead of ISA. Other peripheral devices are connected to CPU through ISA bus to reduce the hardware cost of the system. Remote control device sends command signal to system via remote sensor (infrared red) which is connected to system via RS232C port.

In Zikimi system, operating system, micro kernel and application resides in 4 MByte of Flash Memory. Generally, 32 bit operating system approximately takes up over 30 MByte of main memory and even Linux kernel takes up about 1 Mbyte of memory space. This is still too heavy for embedded system. We trim out the unnecessary features of Linux kernel, e.g. Memory Management, System Utility and reduce the size into 500 Kbyte. Micro kernel that is responsible for video memory processing, input/output port control and memory management requires additional 300 Kbyte. We were able to fit the entire operating system into 800 Kbyte of memory space. Rest of the memory space are used for applications and its accompanying resources, font images, background images, etc.

4. Software architecture

By carefully examining the system requirement, we conclude that the target device has only one active user process at a time. The single application can take full control of the the system resources. Of course, the resources used by operating system are exception. Since our system is used for running various types of application, e.g. digital album, Web TV, and machine vision system, the operating system should provide a flexible environment which can easily adapt to different types of applications. On the other hand, to run the application efficiently, the operating system needs to be tightly integrated with the application. In an effort to compromise these two discrepancies, we partition the entire software stack into two layers: *kernel layer* and *application layer*. Kernel layer consists of embedded Linux kernel and *Zikimi* micro kernel. Application program can be downloaded via parallel port, from CD-ROM or from network device. Figure 5 illustrates software organization of Zikimi micro kernel. Application layer communicates with kernel layer via `system_call ()`. System call is handled via software interrupt. The software interrupt handling routines can be partitioned into six categories: video control, sound control, PCMCIA interface control, linear memory management, file I/O, and Infrared port control for remote control input. Among these, video control, sound control, PCMCIA interface and linear memory management handling module is implemented within micro kernel. File I/O and Infrared control request is accepted via micro-kernel, but the actual handling routine uses the existing module of Linux operating system. Figure 6 illustrates the abstract view of the Zikimi software organization.

4.1. Linux kernel and micro kernel

Developing sophisticated embedded operating system is a serious theme and requires large amount of time and effort. While we develop most of the key features down from scratch, we are able to save substantial amount of resources by adopting a number of features from legacy Linux operating system. Linux is general-purpose operating system and thus is not

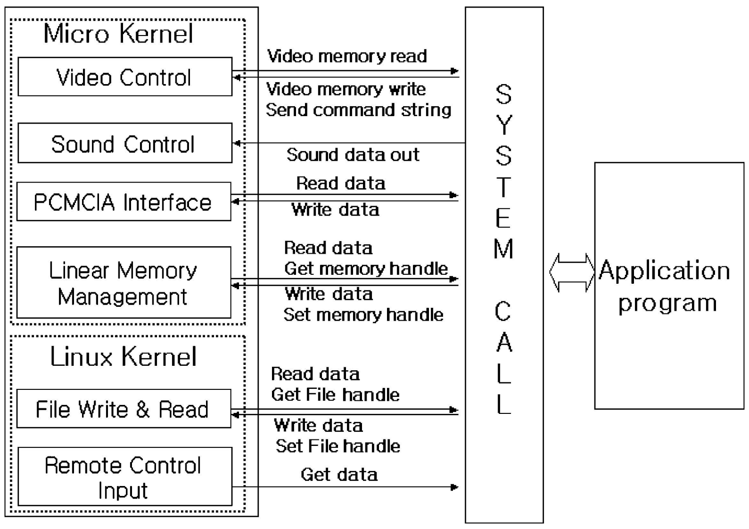


Figure 5. Software architecture.

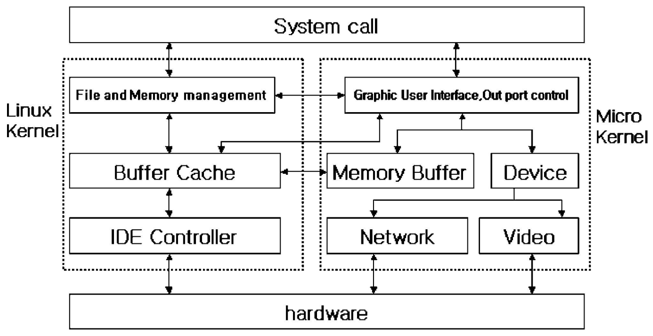


Figure 6. Operating system organization: Linux kernel and micro kernel.

particularly tuned to handle multimedia traffic in the low end hardware environment. Hence, it becomes very important to clearly identify which aspects of the existing operation system need further refinement and which do not.

In Zikimi system, memory management subsystem of Linux kernel is used only to initialize the memory in boot phase, e.g. initializing global/local descriptor table, and the micro-kernel takes over all the remaining free memory blocks. Then, it becomes fully responsible for allocation and deallocation of memory blocks to application. Micro-kernel is responsible for the following tasks: graphical user interface, input/output port control, e.g. PCMCIA, accessing the video chip, and memory management.

When Linux kernel is loaded, the system switches to protected mode, initializes virtual memory space and detects peripheral devices. Micro kernel part of Zikimi system can be

```
static int init(void * unused){
     lock_kernel();
     do_basic_setup();
     free_initmem();
     load_microkernel();
}
```

Figure 7. Init function.

thought as system library which recursively uses some of the services provided by legacy Linux operating system. Figure 7 illustrates the flow of Kernel loading. When kernel is loaded onto memory, micro kernel is able to call buffer cache management and file I/O.

We like to explain how micro kernel of Zikimi operating system can be efficiently and effectively fabricated with the Linux operating system kernel. When the system setup phase completes, legacy Linux kernel normally goes into user mode where most of the application program runs. In our approach, to load the micro-kernel, we modify the `init()` function rather than going into user mode. Figure 7 illustrates modified `init()` function performing this task.

After micro-kernel is loaded via `load_microkernel()` function call, micro kernel takes over the system resources. All the service requests related to system resources are first directed to micro-kernel. Then, the system call block in figure 5 is used determine the respective service handling routine, some of which reside in micro-kernel and some of which reside in Linux. Since micro kernel contains certain system call handling routines as in figure 5, the modules performing the duplicate tasks in Linux operating system are removed. Application program communicates with Zikimi kernel via unified programming interface, `system_call()`.

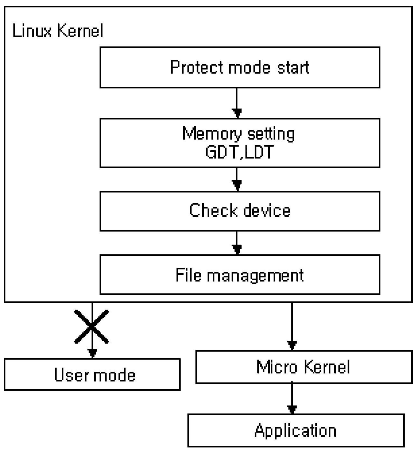


Figure 8. System initialization process.

4.2. Boosting up performance of graphics processing

VESA (Video Electronics Standards Association) interface type which runs on 16 bit mode is the most commonly used method of controlling video processing chip [9]. There are a number of disadvantages in using VESA interface type. 16 bit modes limits the performances of data fetching operation. VESA interface type interleaves the video memory as a set of banks and it is cumbersome to map the video output into the interleaved sequence of memory banks. We devise an elaborate method of controlling the video chip. Most of the videos chip can be controlled via three groups of registers, AR, GR and SR. Since each group consists of 256 $\times$ 8 bit registers, we control the video chip via these registers. Following pair of commands illustrates how to set the value of the first AR register to 5.

```
outb(1, 0x3c4);
outb(5, 0x3c5);
```

Here, `0x3c4` is a port number that is used to specify the register number. This port number depends on the video chip hardware. Port number `0x3c5` is used to specify the value for the register. We can retrieve the value of register in the same way.

$$\begin{aligned} &outb(index, 0x3c4); \\ value &= inb(0x3c5); \end{aligned}$$

By setting the values of these registers, we can control the video output via memory mapped I/O.

To boost up the speed of decompression, we exploit unused fraction of video memory as *local* processing memory for graphic processor. Usually, video processing chip has at least 4 Mbyte of memory. $1024 * 768$ resolution with 24 bit true color requires 3 Mbyte of video memory and thus normally more than 1 Mbyte of memory remains unused. Decompression of JPEG image is computationally intensive operation, especially for 80486 based 66 Mhz STPC. To speed up decompression operation, we use the native operation of the video chip instead of using STPC processor. Most of the high performance video processor can perform the 8 $\times$ 8 block arithmetic operation. This operation can be done by properly setting the registers. There are two main advantages to our approach. Firstly, by exploiting the native 8 $\times$ 8 block arithmetic operation, we can greatly enhance the decompression efficiency. 486 based STPC does not have block based arithmetic instruction and thus block based arithmetic operation, e.g. DCT, inverse DCT, needs to be performed via repetition of 1 $\times$ 1 arithmetic operations. Secondly, by exploiting the unused fraction of video memory in decompressing data, we can avoid system bus traffic between video processing hardware and the main memory. If we use STPC instruction in decompressing image, it first fetches the data block from the video memory to main memory. DMA transfer method can partly relieve CPU burden of memory copy but still cannot completely remove the overhead of memory copy operation.

Figure 9 illustrates the organization of video processing hardware and the respective operational process. We partition the memory of the video card into two parts: Video output

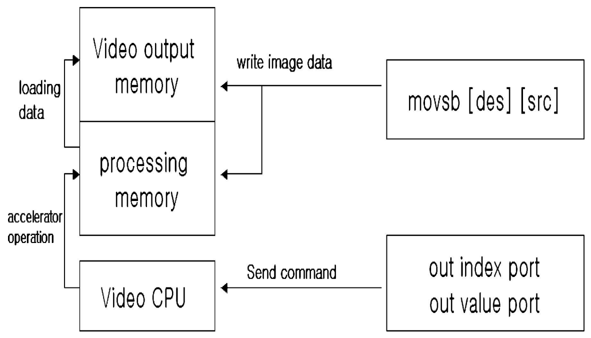


Figure 9. Video processing.

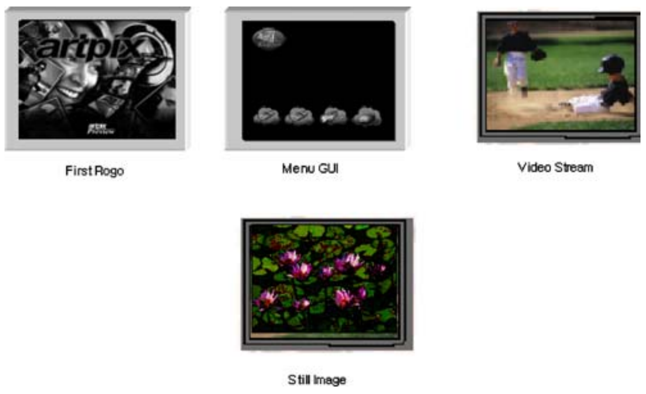


Figure 10. Sample screens of Zikimi Based System.

section and processing section. In legacy system, only video output section is utilized and the rest of the memory remains unused. Uncompressed image is sent to display module by memory block move command. When we need to uncompress the incoming data, it is moved onto processing section of video memory and the decompression command is sent to video chip. Video processor performs block arithmetic operation for decompression and the result is moved to video output section. By using the native 8×8 block based arithmetic operation, we are able to achieve significant improvement on the performance of image decompression in very cost effective manner. Figure 11 compares the performance of decompressing operation by Video CPU, STPC and Pentium-II 200 Mhz. Our micro-kernel based decompression method is approximately three times faster than using STPC. Overall, Pentium II delivers the highest performance. Given the order of magnitude difference in cost between the video CPU and Pentium II processor, it may be more beneficial to use video CPU than to use expensive general-purpose microprocessor. Figure 10 illustrates the sample screens of Zikimi Based System.

4.3. Linear memory system

Most of 32 bit general-purpose operating systems adopt virtual memory system, and Linux operating system is not an exception. Virtual memory system gives good system performance boost by abstracting primary and secondary storage space into one single large virtual primary memory space. However, our system with 4 Mbyte of flash memory based secondary storage does not need virtual memory system and henceforth does not have to take the burden of maintaining the swap space, page table, address translation, etc. In realizing the linear memory system, we decide not to use native linear memory system provided by 32 bit protected mode of $\times 86$ architecture. The main reason is its programming complexity. Instead, micro-kernel is allocated all free memory blocks after kernel initializes virtual memory system and subsequent memory allocation and deallocation requests of the application program are serviced by micro-kernel.

Figure 12 illustrates linear memory organization. Entire Address space can be partitioned into two: main memory section and video memory section. Main memory section takes up the lower part of the address space. Upper part of the address space consists of memory block of video processor and is used for memory mapped I/O.

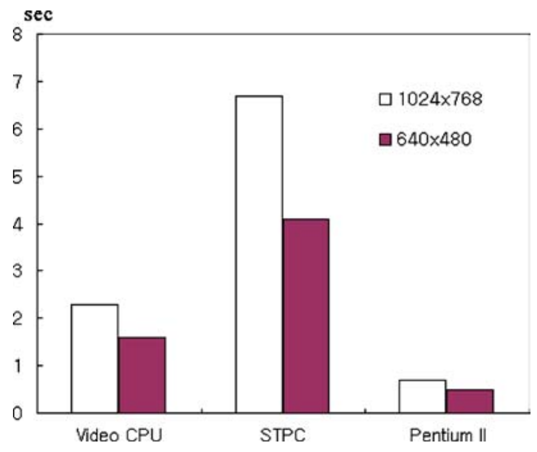
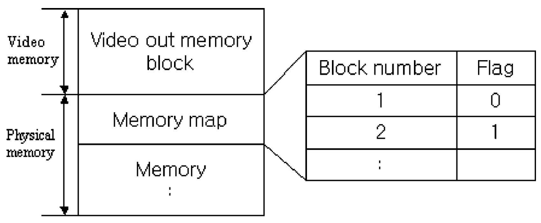


Figure 11. Performance comparison: JPEG image decompression.



Block number	Flag
1	0
2	1
:	

Figure 12. Linear memory mapping.

The address space is managed in the unit 512 Kbyte memory block. Block allocation table (or memory map) of the physical memory resides in the last portion of the main memory section. We use linked list for memory management. The structure of memory management is below.

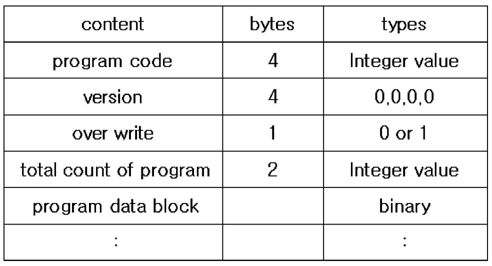
```
typedef struct _smblock{
    char * pblock;
    smblock *next;
    smblock *prev;
}Mblock;
```

According to our experiment, it takes 7.3 μsec and 25.1 μsec to transfer 512 Kbyte block via direct memory addressing and virtual memory addressing, respectively. Thus, linear memory system reduces the memory I/O latency into more than one thirds.

4.4. Re-configuration of software

Our system can be thought as board support package (BSP) which can accommodate different application as well as the kernel. It can be done via parallel port, CD-ROM, or via internet connection. The unit for change (or upgrade) is *package*. Single package can contain a number of different programs and/or files. A package starts with the software descriptor table. Figure 13 illustrates the structure of the software descriptor table. The program code identifies the software type. If this value is 1, it means that the current package is for micro kernel. The content of *over write* denotes whether to overwrite the existing copy or not. If this value is one, the incoming program overwrites the existing copy. The *total count of program* denotes the number of files in the package.

Individual files which may or may not be executable file, in the package starts with the *Program data block*. Figure 14 illustrates the structure of the descript table of the program data block.



content	bytes	types
program code	4	Integer value
version	4	0,0,0,0
over write	1	0 or 1
total count of program	2	Integer value
program data block		binary
:		:

Figure 13. Structure of software descriptor table.

content	bytes	types
Program name	80	string
version	4	0,0,0,0
over write	1	0,1,2
Kernel module	1	0,1
directory	80	string

Figure 14. Structure of program data block.

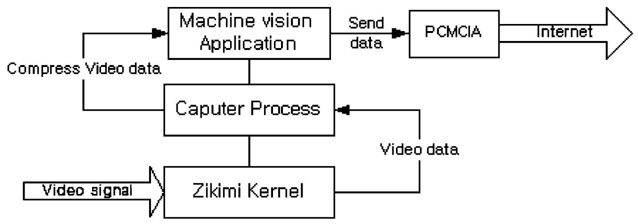


Figure 15. Structure of machine vision system.

5. Applications

5.1. Machine vision system

One of the target application of Zikimi micro kernel is machine vision system. In our case, machine vision system is used to automatically find defective parts or components in the assembly line. Typical applications can be found in identifying defective LCD panel. The system for this purpose may not requires high performance multimedia processing capability. Images are captured in regular interval which is relatively long, e.g. 30 sec. The captured image is processed locally or is sent to remote host via network connection. Figure 15 illustrates the structure of the machine vision system

5.2. Digital album

Personal computer has been the prime choice in retrieving and saving the pictures from digital camera. However, one of the newly emerging Internet Appliances is digital album, which is used to archive digital images and to transfer the pictures remotely via network connection. Our system provides very effective platform for this purpose.

5.3. Web-TV

Enjoying the online digital video via Internet is not an innovative application anymore. From the streaming client's perspective, the service process consists of establishing connection to the remote streaming server, receiving the continuously arriving packets, decompressing the frames and displaying them to screen. While the operational overhead of this service varies depending on the QoS of the stream, compression method, etc., it normally does not fully utilize the capacity of the general-purpose personal computer. Recently, there have been a number of proposals for developing a dedicated Information Appliances for this purpose, e.g. movie theater quality HDTV, PDA equipped with wireless Internet link, etc. These devices share the common characteristics: video processing module, Internet link, and multimedia data processing. The embedded system presented in this article can be used to build this type of system in a cost effective manner.

6. Summary

In this article, we present our design and implementation experience of embedded system that consists of Linux operating system kernel and micro kernel. After years of intense effort, we successfully developed a Linux based embedded system which is specifically tailored to handle digital images, online digital video and Internet navigation. Legacy Linux kernel has been developed for multitasking and multi user environment. Many of the features in Linux, e.g. virtual memory management, support for multitasking, virtual file system, complex buffer cache management algorithm, and etc. may not be required when it is used to operate small scale information processing device, e.g. Internet Appliances, PostPC products, etc. Linux operating system provides good framework for controlling various peripheral devices. However, resource limited hardware platform does not allow to accommodate legacy operating system. It is therefore very important to effectively tailor the legacy Linux operating system and to develop core features which exploits the hardware resources for multimedia data processing. We develop micro-kernel, *Zikimi*, whose main responsibility is to enhance the speed of multimedia processing operation. The salient features of Zikimi micro kernel is two fold: (i) linear memory system and (ii) user level device control via `system_call()` interface. Our novel technique enables the end user to enjoy saving, editing, retrieving the multimedia information in more cost effective manner.

References

1. S. Bahadur, V. Kalyanakrishnan, and J. Westall, "An empirical study of the effects of careful page placement in Linux," in Proceeding of the 36th annual conference, 1998, pp. 241–250.
2. Brian N. Bershad, Craig Chambers, Susan J. Eggers, Chris Maeda, Dylan McNamee, Przemyslaw Pardyak, Stefan Savage, and Emin Gun Sirer, "{SPIN}—An extensible microkernel for application-specific operating system services," ACM SIGOPS European Workshop, 1994, pp. 68–71.
3. Jan Fandrianto, "Single Chip MPEG2 decoder with integrated transport decoder for set–top box," in Proceeding of COMPCON'96, 1996, pp. 469–472.
4. G. Gogniat, M. Auguin, and L. Bianco, "A codesign back-end approach for embedded system design," ACM Trans. On Design Automation of Electronic System, Vol. 5, No. 3, pp. 492–509, 2000.
5. S.M. Hand, "Self-paging in the Nemesis operating system," in Proc. of 3rd USENIX Symp. on Operating Systems Design and Implementation (OSDI'99), New Orleans, LA , USA, February 1999, pp. 73–86.

6. Dan Hildebrand, “An architectural overview of QNX,” in 1st USENIX Workshop on Micro-Kernels and Other Kernel Architctures, Seattle, WA, April 1992, pp. 113–126.
7. http://os.inf.tu-dresden.de/fiasco/
8. T.R. Hurley, “Evolution of the digital set top box,” International Broadcasting Convention, Septem-ber Conference Publication, No. 428, pp. 277–282, 1996.
9. Egbert G.T. Jaspers and H.N. Peter, “Chip-Set for Video Display of multimedia information,” IEEE Trans. On Consumer Electronics, Vol. 45, No. 3, pp. 706–715, 1999.
10. Kiyoshi Kohiyama, Hideaki Shirai, Kiyotaka Ogawa, and Akio Manakata, “Architecture of MPEG-2 digital set-top box for CATV VOD system,” IEEE Trans. On Consumer Electronics, Vol. 42, No. 3, pp. 667–672, 1996.
11. R. Li, Ngai K. Chung, Kam T. Mo, David M. Fisher, and Vena Wong,“A flexible display module for DVD and set-top box applications,” IEEE Trans. On Consumer Electronics, Vol. 43, No. 3, pp. 496–503, 1997.
12. Stuart Pekowsky and Rudolf Jaeger, “The set-top box as multi-media terminal,” IEEE Trans. On Consumer Electronics, Vol. 44, No. 3, pp. 834–840, 1998.

Sang-Yeob Lee received his B.S. and M.S degree from Hanyang University, seoul, Korea in 1995. He is currently working towards the Ph.D. degree in Devision of Electrical and Computer Engineering, Hanyang University, Seoul, Korea. Since 1998, he has been on the faculty of Information Management System at Sahmyook university, Seoul, Korea. His research interests include robot vision systems, pattern recognition, Multimedia systems. He is a member of IEEE.

Youjip Won received the B.S and M.S degree in Computer Science from the Department of Computer Science, Seoul National University, Seoul, Korea in 1990 and 1992, respectively and the Ph.D. in Computer Science from the University of Minnesota, Minneapolis in 1997. After finishing his Ph.D., He worked as Server Performance Analysts at Server Architecture Lab., Intel Corp. Since 1999, he has been on the board of faculty members in Division of Electrical and Computer Engineering, Hanyang University, Seoul, Korea. His current research interests include Multimedia Systems, Internet Technology, Database and Performance Modeling and Analysis. He is a member of ACM and IEEE.

Whoi-Yul Kim received his B.S. degree in Electronic Engineering from Hanyang University, Seoul, Korea in 1980. He received his M.S. from Pennsylvania State University, University Park, in 1983 and his Ph.D. from Purdue University, West Lafayette, in 1989, both in Electrical Engineering. From 1989 to 1994, he was with the Erick Jonsson School of Engineering and Computer Science at the University of Texas at Dallas. Since 1994, he has been on the faculty of Electronic Engineering at Hanyang University, Seoul, Korea. He has been involved with research development of various range sensors and their use in robot vision systems. Recently, his work has focused on content-based image retrieval system. He is a member of IEEE.