Module 2 -- Low Level Hardware-Software Interface
5/31/19

It is easy to think of CPU "hardware" as nothing more than a complex arrangement of logic circuits that implement an instruction set. Users interact with the hardware by writing programs whose code is a sequence of instructions. Unfortunately, that idea is too simplistic. Parts of the kernel interact with the hardware outside the instruction set in these three ways:

- Interrupt signals
- Supervisor states (also called rings)
- BIOS/EFI

The OS provides an interface to manage user interactions with the hardware:

- Interrupt handlers (software triggered by interrupt signal)
- Device drivers (software that speaks to devices such as disks, displays or printers)
- Call stack (software for procedure call and return)
- Memory Mapping Unit (translates CPU address to RAM addresses)

A few comments on each of these, then you are off to explore them in more detail.

Interrupt signals and handlers. Interrupt signals generated by sensors located throughout the hardware to detect *exceptional conditions* such as arithmetic errors (from the ALU), missing pages (from the MMU), system calls (from a kernel call instruction), disk completion (from an I/O device), and alarms (from a timer clock). These signals are fed to a selector circuit that generates a code telling the OS which signal to process next. At the end of every instruction cycle, the CPU checks whether an interrupt code has been generated; if so, it jumps to the OS procedure that deals with that kind of interrupt (the *interrupt handler*), as designated in the *interrupt vector*, a list of the interrupt handler entry points. The jump is implemented with a procedure call – which uses the call stack mechanism described below.

Supervisor States. The instruction set is divided into two subsets according to the degree of sensitivity of each instruction. Instructions that affect only the memory of the CPU that executes them are considered non-sensitive and classified as non-privileged, or *user-state*, instructions. Instructions that change the state of the system and affect other executing programs are considered sensitive and classified as privileged, or *kernel-state*, instructions. Routines in the microkernel are all considered privileged because they need unrestricted access to all parts of memory and all instructions. Routines in the user kernel and those in user code are all considered non-privileged because they are not allowed to access any memory other than that assigned to the program using them. A kernel-state bit in the CPU *processor status word* (PSW) holds the value 1 to indicate that the CPU is in kernel (privileged) mode, or 0 to indicate that the CPU is in user (non-privileged) mode. The OS is coded to switch the kernel-state bit to 1 when a user program calls an

entry point in the microkernel, and back again to 0 when the kernel state is no longer required. The CPU can execute any instruction in the kernel state, but only the non-sensitive instructions when in the user state.

BIOS/EFI. BIOS means "basic input output system" and EFI "extended firmware interface". These are standard interfaces to the basic input-output devices attached to the CPU and are used when booting the machine. For example, every drive manufacturer provides device startup firmware which can be invoked by instructions loaded onto the CPU from the BIOS/EFI at boot time; then a CPU can access the disk via the interface stored in BIOS, regardless of which manufacturer made the disk. The purpose of the BIOS is to avoid having to recompile the kernel every time a new device is attached. The devices registered in BIOS/EFI will be accessible as soon as the OS is booted.

Call Stack. The standard approach to calling procedures (subprograms) is to push frames (activation records) onto a stack when calling the procedure and pop the frames off on return. The hardware provides registers pointing to the current stack frame and memory segments, while the OS provides the call and return instructions which manipulate the stack and pointers. An activation record contains return information such as the value of the instruction and stack pointers to be restored on return, values of local variables, and a stack workspace for the procedure to use. The call stack is thus a combination of hardware and software.

Memory Mapping Unit (MMU). This piece of hardware in the CPU converts logical program addresses into physical RAM addresses. It typically refers to a page table to find the exact RAM location of an address-space page. If the desired target address is not loaded, the MMU generates an interrupt signal to tell the OS to find the missing page and load it. MMU also contains a translation lookaside buffer (TLB), a small cache of recently mapped pages; if the desired target address in in the TLB, the target page address can be found immediately without lookup from the page table. The TLB significantly improves MMU performance at a small increment of cost. We will discuss address mapping in more detail in a later exploration.

Resources

Understanding the Microprocessor (tutorial on microprocessor architecture)

In Resources/documents directory:

pjd-on-Machines.pdf (review of the nature of computing machines controlled by the operating system)

o*s99.pdf* §5.1 (p13) (technical reference)

OS_Levels.pdf (one-page map of the levels of the OS)

Interrupt-architecture.pdf (diagram of the components of interrupt system)