EXTENDED MODEL ANSWER

Operating Systems 2003 Paper 1 Question 12 (SMH)

Operation of a Simple Computer

Note: this question relates to the first part of the course, computer organization. The diagram I'm looking for is something akin to that on the slide 'Fetch-Execute Cycle Revisited' in my notes. The "simple computer" referred to is a sort of RISC machine used as a running example in this part of the course.

The simple computer will fetch an instruction from memory, and then execute it, and then repeat – hence the "fetch-execute cycle". Instructions are decoded by the processor to determine what precisely must be done. Some common instruction types include arithmetic & logical instructions such as add, sub, and, xor, ... Most of these are performed by the ALU and have two input operands (registers or perhaps one literal) and an output operand (a register).

Since these instruction operate on registers, there must be a way to load and store values from and to memory. These instructions must *address* memory in some fashion; that is, allow the computation of a byte address referent. The addressing modes in the simple computer are just register indirect and variants. Loads and stores to memory go over a bus – a shared set of data and control wires.

The address of the "next" instruction to be fetched by the processor is held in an internal register called the program counter (PC). In normal operation, this is incremented by a fixed amount after every instruction fetch so that consecutive instructions in memory will be loaded. To alter this default (e.g. to allow branches, loops or procedure calls). control flow instructions effectively update the PC. Conditional codes are often combined with these so that the PC update is predicated on the outcome of a simple ALU-style operation.

I/O Access in Supervisor Mode

I hope this is well-formed / unambiguous; what I'm looking for is the 'raw' I/O access available to e.g. the operating system code.

When running in supervisor mode, the program has direct access to I/O devices. In some

machines there may be explicit instructions to access devices; in other cases one may simply perform memory reads and writes to a special area of physical memory; these reads and writes propagate across the bus to devices which then interpret them in some fashion. Communicating data to/from devices may involve reads and writes, or may use direct memory access. In either case, interrupts may be used to signal the end of an operation – this will reset the PC to a defined handler which can then do whatever may be necessary.

I/O Access in User Mode

In user-mode, programs cannot directly access I/O devices; if explicit I/O instructions are supported, they will fault in user mode; if memory access is being used, then these parts of the physical address space will not be mapped into the processes virtual address space. This is for reasons of safety.

Instead, user-mode programs must invoke the operating system to perform I/O tasks on their behalf. This involves a system call or trap which switches from user mode into a well defined handler within the operating system which can then perform the requested operation.