Chapter 2 – Hardware – Part 2

Instructor: Mark Llewellyn
markl@cs.ucf.edu
CSB 242, 823-2790

School of Electrical Engineering and Computer Science
University of Central Florida
Interrupts

• Interrupt the normal sequencing of the processor
• Most I/O devices are slower than the processor
  – Processor must pause to wait for device
# Classes of Interrupts

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program</td>
<td>Generated by some condition that occurs as a result of an instruction execution, such as arithmetic overflow, division by zero, attempt to execute an illegal machine instruction, and reference outside a user's allowed memory space.</td>
</tr>
<tr>
<td>Timer</td>
<td>Generated by a timer within the processor. This allows the operating system to perform certain functions on a regular basis.</td>
</tr>
<tr>
<td>I/O</td>
<td>Generated by an I/O controller, to signal normal completion of an operation or to signal a variety of error conditions.</td>
</tr>
<tr>
<td>Hardware failure</td>
<td>Generated by a failure, such as power failure or memory parity error.</td>
</tr>
</tbody>
</table>
Program Flow of Control Without Interrupts
Program Flow of Control With Interrupts, Short I/O Wait
Program Flow of Control With Interrupts; Long I/O Wait

Diagram:
- User Program
  - WRITE
  - WRITE
  - WRITE
- I/O Program
  - WRITE
  - WRITE
  - WRITE
- I/O Command
- Interrupt Handler
  - END
Interrupt Handler

- Program to service a particular I/O device
- Generally part of the operating system
Interrupts

• Suspends the normal sequence of execution
Interrupt Cycle

START → Fetch next instruction → Execute instruction → Interrupt Stage

Interrupts Disabled

Check for interrupt; initiate interrupt handler

Interrupts Enabled

HALT
Interrupt Cycle

- Processor checks for interrupts
- If no interrupts fetch the next instruction for the current program
- If an interrupt is pending, suspend execution of the current program, and execute the interrupt-handler routine
Timing Diagram Based on Short I/O Wait

(a) Without interrupts

(b) With interrupts

Numbers refer to page 5
Timing Diagram Based on Long I/O Wait

(a) Without interrupts

(b) With interrupts

Numbers refer to page 4

Numbers refer to page 6
Simple Interrupt Processing

**Hardware**

- Device controller or other system hardware issues an interrupt
- Processor finishes execution of current instruction
- Processor signals acknowledgment of interrupt
- Processor pushes PSW and PC onto control stack
- Processor loads new PC value based on interrupt

**Software**

- Save remainder of process state information
- Process interrupt
- Restore process state information
- Restore old PSW and PC
Changes in Memory and Registers for an Interrupt

Interrupt occurs after instruction at location $N$
Changes in Memory and Registers for an Interrupt

Return from interrupt
Multiple Interrupts

• Disable interrupts while an interrupt is being processed

Sequential interrupt processing
Multiple Interrupts

• Define priorities for interrupts
Multiple Interrupts

Example Time Sequence of Multiple Interrupts
Multiprogramming

- Processor has more than one program to execute
- The sequence the programs are executed depend on their relative priority and whether they are waiting for I/O
- After an interrupt handler completes, control may not return to the program that was executing at the time of the interrupt
Memory Hierarchy

- Faster access time, greater cost per bit
- Greater capacity, smaller cost per bit
- Greater capacity, slower access speed
Disk Cache

- A portion of main memory used as a buffer to temporarily hold data for the disk
- Disk writes are clustered
- Some data written out may be referenced again. The data are retrieved rapidly from the software cache instead of slowly from disk
Cache Memory

- Invisible to operating system
- Increase the speed of memory
- Processor speed is faster than memory speed
- Exploit the principle of locality
Cache Memory

Word Transfer

Block Transfer

CPU → Cache → Main Memory

Cache and Main Memory
Cache Memory

- Contains a copy of a portion of main memory
- Processor first checks cache
- If not found in cache, the block of memory containing the needed information is moved to the cache and delivered to the processor
Cache/Main Memory System

(a) Cache

(b) Main memory

Cache/Main-Memory Structure
Cache Read Operation

1. **START**
2. Receive address RA from CPU
3. Is block containing RA in cache?
   - Yes: Fetch RA word and deliver to CPU
   - No: Access main memory for block containing RA
4. Allocate cache slot for main memory block
5. Load main memory block into cache slot
6. Deliver RA word to CPU
7. **DONE**
Cache Design

• Cache size
  – Small caches have a significant impact on performance

• Block size
  – The unit of data exchanged between cache and main memory
  – Larger block size more hits until probability of using newly fetched data becomes less than the probability of reusing data that have to be moved out of cache
Cache Design

• Mapping function
  – Determines which cache location the block will occupy

• Replacement algorithm
  – Determines which block to replace
  – Least-Recently-Used (LRU) algorithm
Cache Design

• Write policy
  – When the memory write operation takes place
  – Can occur every time block is updated
  – Can occur only when block is replaced
    • Minimizes memory write operations
    • Leaves main memory in an obsolete state
Programmed I/O

- I/O module performs the action, not the processor
- Sets appropriate bits in the I/O status register
- No interrupts occur
- Processor checks status until operation is complete
Interrupt-Driven I/O

- Processor is interrupted when I/O module ready to exchange data
- Processor saves context of program executing and begins executing interrupt-handler
- No needless waiting
- Consumes a lot of processor time because every word read or written passes through the processor
Direct Memory Access

- Transfers a block of data directly to or from memory
- An interrupt is sent when the transfer is complete
- Processor continues with other work