ECE 250 / CS 250
Introduction to Computer Architecture

Exceptions/Interrupts

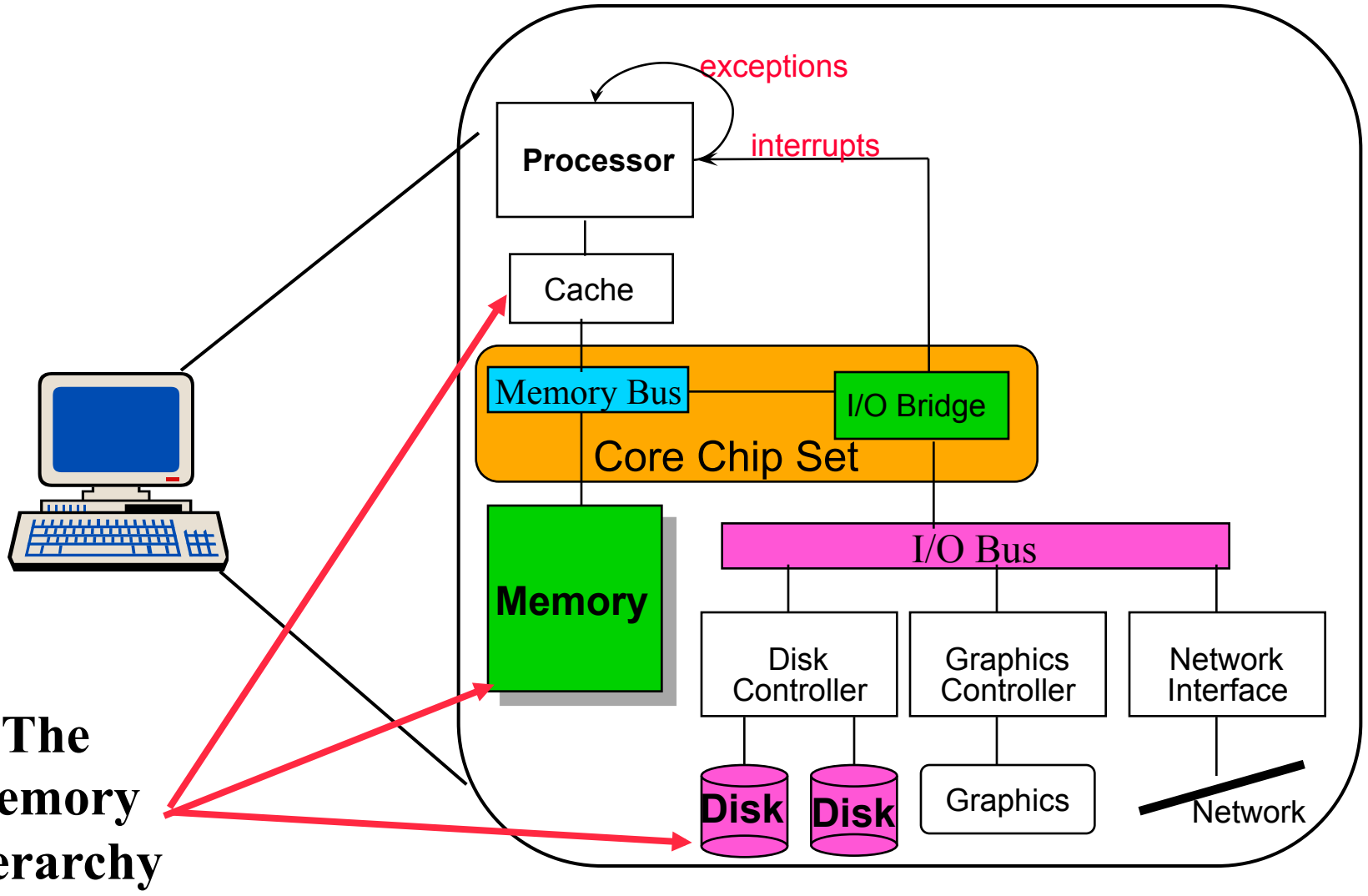
Benjamin C. Lee
Duke University

Slides from Alvin Lebeck (Duke)

Administrivia

- Homework #5 – Due Apr 11 @ 11:55pm

System Organization



What Does an Operating System Do?

- **Service Provider**
  - exports commonly needed facilities with standard interfaces
  - simplifies programs
  - portability
- **Executive**
  - resource manager for greatest good
- **Custodian of the Machine**
  - monitors hardware, intervenes to resolve exceptional conditions
- **Cop**
  - protects you from others

Executing a Program

- Thread of control (program counter)
  - multiple threads/programs running
- Basic steps for program execution
  - fetch instruction from Memory[PC],
  - execute the instruction,
  - increment PC
- At boot-time, begin with PC at well-known location
  - loads the operating system kernel

An Execution Context

- CPU state associated with program context
  - general purpose registers (integer and floating point)
  - status registers (e.g., condition codes, HI/LO)
  - program counter, stack pointer
  - cache and memory hierarchy
- Switch between contexts
  - Increases machine utilization
  - Allows programs to share machine (e.g., timeslicing)
  - Permits different execution modes (e.g., user versus OS kernel)
- Operating system maintains contexts

Context Switches

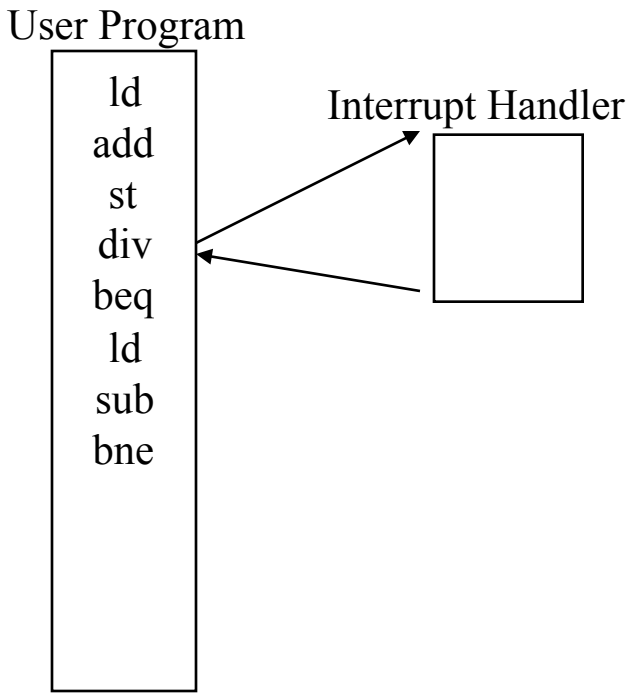
- Save current execution context
  - Save registers and program counter
- Restore other context

- Why switch contexts?
  - Switch between programs that share hardware
  - Switch to handle I/O that begins or ends
  - How do we know these events occur? Interrupts

Interrupts and Exceptions

- Exception is an external event
  - These events require responses
  - Clock interrupts for timeslicing and context switches
  - I/O operations for disk, network, keyboard, etc.
  - “Exception”, “interrupt”, “trap” are used interchangeably
- Exception is an infrequent event
  - Examples: I/O, divide-by-zero, illegal instruction, page fault, protection fault, ctrl-C, ctrl-Z, timer
- Exception handling requires OS intervention
  - Handling is transparent to application code
  - End program: divide-by-0, protection fault, illegal instruction
  - Fix and restart program: I/O

Handling an Exception/Interrupt

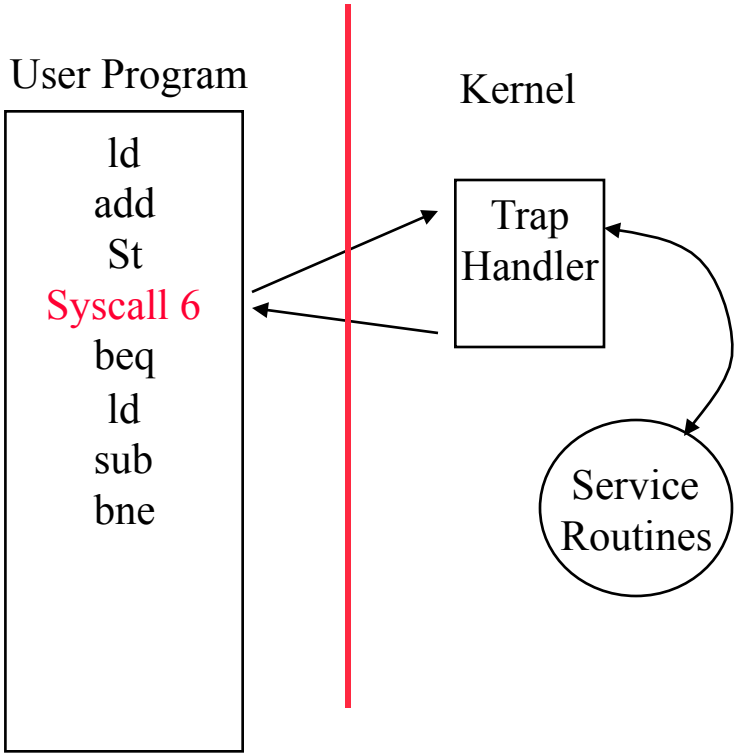


- Determine interrupt's cause
- Invoke specific kernel routine based on type of interrupt
  - interrupt/exception handler
  - kernel initializes table at boot
  - PC = interrupt_table[i]
- Clear interrupt signal
- Return from interrupt handler
  - Return to original context
  - Return to different context

Execution Mode

- What if interrupt occurs while in interrupt handler?
  - Interrupt #2 could interfere with handling interrupt #1
  - Solution disables interrupts within handler
  - Disabling interrupts is a protected operation
- Protected Operations
  - Only the operating system kernel can perform these operations
  - Mode bit in CPU status register (user versus kernel)
  - Example: disabling interrupts, installing interrupt handlers, manipulating processor state (saving/restoring status registers)
- Changing Modes
  - Use interrupts or system calls (syscall instruction)

System Call (syscall)



- Special instruction changes modes and invokes service
  - read/write I/O device
  - create new process
- Invokes specific kernel routine based on argument

Handing Exceptions and Interrupts

Exceptions are like an “asynchronous procedure call”

1. Processor saves address of offending instruction in the EPC (Exception Program Counter)
2. Processor transfers control to OS at specified instruction address
3. OS executes instructions, responding to interrupt
4. OS terminates program or returns using EPC

Handling Exceptions and Interrupts

- For example, consider 3 types of exceptions
  - undefined instruction
  - arithmetic overflow
  - unaligned access

- Which event caused exception?
  - Option 1 (used by MIPS): Cause register contains reason
  - Option 2: Vectored interrupts where cause is encoded in address.
    - » Addresses separated by 32 instructions
    - » Example:

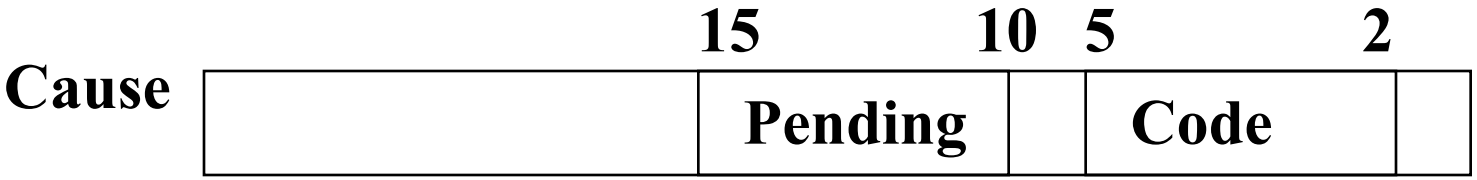
Exception Type	Exception Vector Address
Undefined instruction	C0 00 00 00
Arithmetic overflow	C0 00 00 20
Unaligned Access	C0 00 00 40

    - » Jump to appropriate address

Additions to MIPS ISA

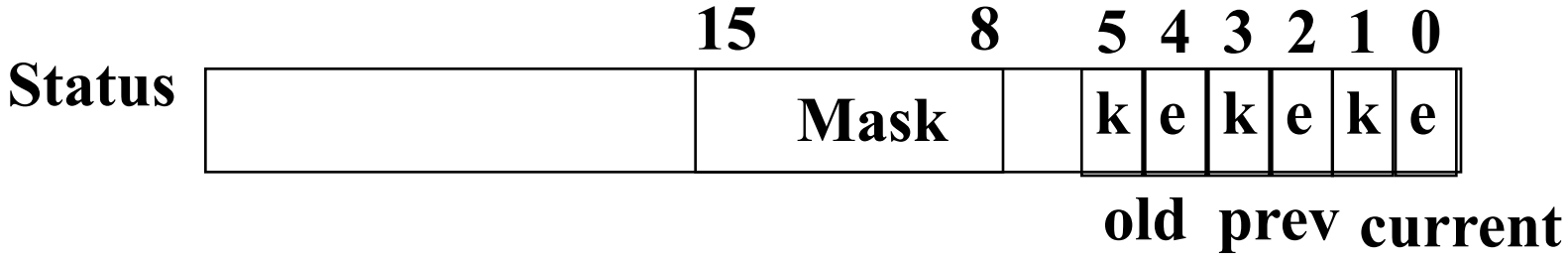
- EPC
  - 32-bit register used to hold the address of the affected instruction
- Cause:
  - a register used to record the cause of the exception.
  - In MIPS, this register is 32 bits; some bits are currently unused.
- BadVAddr
  - register contains memory address at which reference occurred
  - used when memory access exception occurs
- Status:
  - interrupt mask and enable bits

Details of Cause Register



- Pending Interrupt
  - 5 hardware levels: bit set if interrupt occurs but not yet serviced
  - handles cases when more than one interrupt occurs at same time, or records interrupt requests when interrupts disabled
- Exception Code
  - encodes reasons for interrupt
  - 0 (INT) => external interrupt
  - 4 (ADDRL) => address error exception (load or instr fetch)
  - 5 (ADDRS) => address error exception (store)
  - 6 (IBUS) => bus error on instruction fetch
  - 7 (DBUS) => bus error on data fetch
  - 8 (Syscall) => Syscall exception
  - 9 (BKPT) => Breakpoint exception
  - 10 (RI) => Reserved Instruction exception
  - 12 (OVF) => Arithmetic overflow exception

Details of MIPS Status Register



- Mask = 1 bit for each of 5 hardware and 3 software interrupt levels
  - 1 enables interrupts, 0 disables interrupts
- k = specifies kernel/user mode
  - 0 => was in the kernel when interrupt occurred
  - 1 => was running user mode
- e = interrupt enable
  - 0 means disabled, 1 means enabled
- Interrupt handler runs in kernel mode with interrupts disabled
  - When interrupt occurs, 6 LSB shifted left 2 bits, setting 2 LSB to 0

Detecting Exceptions

- Undefined Instruction
  - Detect unknown opcode
- Arithmetic overflow
  - Add logic in the ALU to detect overflow
  - Provide overflow signal as ALU output
- Unaligned access
  - Add circuit to check addresses
  - E.g., lw address must have 2 least significant bits == 0

Exception Summary

- Control is hard part of computer design
- Exceptions are the hard part of control

- Need to find convenient place to detect exceptions PC and invoke the operating system