Computer System Structure

Computer system can be divided into four components

- **Hardware** – provides basic computing resources
  - CPU, memory, I/O devices
- **Operating system**
  - Controls and coordinates use of hardware among various applications and users
- **Application programs** – define the ways in which the system resources are used to solve the computing problems of the users
  - Word processors, compilers, web browsers, database systems, video games
- **Users**
  - People, machines, other computers
Operating System Services

• An OS provides a number of services to a user or application program including:
  – Communications
  – Resource Allocation
  – Accounting
  – Protection
  – Error Detection
  – Program Execution
  – I/O Operations (explicit requests)
  – File System Manipulation
An Event Driven System

- OS services are performed on behalf of the user or application program in response to specific events.
  - The OS does not perform any “useful” work (e.g., solve a user’s problem)
  - Unless prompted, the OS just stays out of the way as much as possible
- Events include:
  - Hardware Interrupts from external (non-processor) devices
  - Interrupts caused by execution of a program (aka software interrupts)
  - Supervisor Calls - User/Application request OS service
  - Trap - Error occurs during execution
Computing Cycle

- Modern computers operate using a three-step cycle:
  - Fetch
    - Get next instruction pointed to by the program counter
    - Increment the program counter
    - Decode the instruction and operands
  - Execute
    - Execute the instruction (if possible)
    - If instruction is a supervisor call, turn control over to OS
  - Interrupt
    - Inspect interrupt flags to determine if error has occurred (e.g., overflow trap) or hardware device requires attention (e.g., printer hardware interrupt)
    - If no interrupt, repeat cycle
    - If interrupt, turn control over to OS which executes appropriate interrupt handler
OS / Hardware Linkage

• Modern operating systems required the development of special hardware components to perform various services.
### OS / Hardware Linkage (cont.)

<table>
<thead>
<tr>
<th>Register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MDR</strong></td>
<td><strong>Memory Data Register</strong>&lt;br&gt;Instructions or data are loaded in this special purpose register from memory one word at a time over the system bus. Data to be stored in memory is placed here first, then sent to memory over the system bus.</td>
</tr>
<tr>
<td><strong>MAR</strong></td>
<td><strong>Memory Address Register</strong>&lt;br&gt;Contains the address of the word to be stored in / retrieved from memory.</td>
</tr>
<tr>
<td><strong>BASE</strong></td>
<td><strong>Base Address Register</strong>&lt;br&gt;Lowest physical memory address allocated to a process.</td>
</tr>
<tr>
<td><strong>LIMIT</strong></td>
<td><strong>Limit Address Register</strong>&lt;br&gt;Highest physical memory address allocated to a process.</td>
</tr>
<tr>
<td><strong>PC/IP</strong></td>
<td><strong>Program Counter / Instruction Pointer</strong>&lt;br&gt;Contains the address of the next instruction to be executed.</td>
</tr>
<tr>
<td><strong>GPR</strong></td>
<td><strong>General Purpose Registers</strong>&lt;br&gt;These registers are programmer accessible. Program data is stored here temporarily for use by the ALU.</td>
</tr>
<tr>
<td><strong>IR</strong></td>
<td><strong>Instruction Register</strong>&lt;br&gt;Operation codes are extracted from the MDR and copied to this register. During decoding the processor determines what operation to perform and what operands are required.</td>
</tr>
<tr>
<td><strong>MB</strong></td>
<td><strong>Mode Bit</strong>&lt;br&gt;Used in dual mode operation to indicate whether CPU is in user mode (1) or operating system / supervisor mode (0)</td>
</tr>
<tr>
<td><strong>ALU</strong></td>
<td><strong>Arithmetic Logic Unit</strong>&lt;br&gt;Performs the actual operations (add, sub, compare, etc) on program data.</td>
</tr>
</tbody>
</table>
Dual Mode Operation

• Uses a Mode Bit to indicate whether a user process or OS process is running
  – 0 = Supervisor Mode (aka monitor mode, system mode)
  – 1 = User Mode
• Allows OS to protect itself and user processes
• Only privileged instructions can be executed in supervisor mode
  – e.g., set mode bit, set timer, reset interrupts
• Can also be used for I/O protection
  – Make I/O instructions privileged.
  – Require system calls (SVC) for users to request I/O operations
Dual Mode Operation (cont.)

Diagram showing system components and connections:
- System Timer
- Printer Controller
- System Bus
- Interrupt Signals
- Interrupt Flip/Flops
- Memory Controller
- System Timer
- Printer Controller
- System Bus
- Interrupt Signals
- Interrupt Flip/Flops
- Memory Controller

Key elements:
- OPC
- ADDR
- MDR
- MAR
- IR
- BASE
- LIMIT
- GPR
- PC/IP
- 1/0
- MB
- ALU

System areas:
- OS Area
- User Area

Address and Memory Locations:
- 50000
- OPC ADDR

Other components:
- Memory Controller
- Printer Controller
- System Timer
- System Bus
CPU Protection

- Need to prevent a user process from monopolizing the CPU
  - e.g., catch infinite loops

- Uses a System Timer connected to an interrupt flip/flop
  - When timer counts down to zero, interrupt is raised

- Can also be used to implement time sharing
CPU Protection (cont.)
Memory Protection

• Required to prevent user processes from accessing / writing to memory outside of their allocated portion of RAM.

• In single-program environment uses a fence register

• For multi-programming, use base and limit
  – **Base** - Lowest address in process’ memory allocation
  – **Limit** - Highest address in process’ memory allocation

• Address of each memory location accessed / written to by user process is compared with base and limit (or fence)
  – If “out of bounds” traps to OS
Memory Protection (cont.)

- **Interrupt Flip/Flops**: 0, 1, 2, 3, ..., 15

- **System Bus**

- **System Timer**

- **Printer Controller**

- **Memory Controller**

- **Interrupt Signals**

- **OPC ADDR**: 30000, 120000, 50000

- **BASE**: 30000, 120000

- **LIMIT**: 50000

- **OS Area**

- **User 1 Area**

- **User 2 Area**

- **System Timer**

- **Printer Controller**

- **Memory Controller**

- **System Bus**
Error Detection

• OS can trap various errors that occur during execution of a user process

• Math Overflow:
  – Occurs when adding two numbers and result too big for accumulator
  – Overflow bit changes to “1” if error occurs

• Divide by Zero
  – Occurs when dividing by zero value
  – Hardware compares opcode with value of divisor.
  – If opcode is DIV and divisor = 0 then trap to OS
Error Detection Math Overflow

System Timer

Printer Controller

Memory Controller

System Bus

Interrupt Signals

Interrupt Flip/Flops

0 1 2 3 ....... 15

ADD R1,R2

MDR

MAR

BASE

LIMIT

GPR

PC/IP

ALU

IR

MB

Overflow

ADD

1 1 1 1

0 0 0 1

0 0 0 0

1

50000

50000

OS Area

User Area

ADD R1,R2

50000
Error Detection - Divide by Zero

System Timer

Printer Controller

Memory Controller

System Bus

Interrupt Signals

Interrupt Flip/Flops

0 1 2 3 ...... 15

Interrupt Flip/Flops

DIV  R1,R2

MDR

MAR

0 1 2 3 4 5

Overflow

ALU

IR

MB

DIV

56789

0

GPR

BASE

LIMIT

PC/IP

50000

DIV  R1,R2

50000

OS Area

User Area

System Timer

Printer Controller

Memory Controller

System Bus

Interrupt Signals

Interrupt Flip/Flops

0 1 2 3 ...... 15

Interrupt Flip/Flops

DIV  R1,R2

MDR

MAR

0 1 2 3 4 5

Overflow

ALU

IR

MB

DIV

56789

0

GPR

BASE

LIMIT

PC/IP

50000

DIV  R1,R2

50000

OS Area

User Area
Handling I/O Operations

• I/O operations typically initiated by user request
  – System or Supervisor Calls (SVC)
• OS communicates user’s request to appropriate I/O device controller
  – Synchronous I/O - control returned to user process after I/O completes
  – Asynchronous I/O - control returned immediately back to user process
• OS notified of I/O completion when device controller raises hardware interrupt flag
  – Device can also use interrupt to signal status (e.g., out of paper, no media)
Handling I/O Operations (cont.)

- Interrupt Vector
  - Interrupt 1 Handler
  - SVC (I/O) Handler
  - OS Area
  - User Area
  - Process 1
  - Interrupt 1

- Memory Controller
- Interrupt Flip/Flops
- System Bus
- System Timer
- Printer Controller
- ALU
- GPR
- PC/IP
- MDR
- MAR
- IR
- MB
- BASE
- LIMIT
- Overflows

- Interrupt Signals
- Memory Controller
- System Bus
- Printer Controller
- System Timer

- Interrupt Signals
- Memory Controller
- System Bus
- Printer Controller
- System Timer

- Interrupt Signals
- Memory Controller
- System Bus
- Printer Controller
- System Timer

- Interrupt Signals
- Memory Controller
- System Bus
- Printer Controller
- System Timer

- Interrupt Signals
- Memory Controller
- System Bus
- Printer Controller
- System Timer
Processing Interrupts

• Interrupt Vector - a series or array of addresses stored in lower memory which point to each interrupt handler.

• Interrupt Handlers - portions of OS program specifically written to deal with each interrupt type

• Remember......
  – A process is a program in execution
  – An OS is a program
  – Therefore, OS is also a process
  – Very large OS, may spawn multiple processes
When An Interrupts Occurs:

• Mode bit set to supervisor mode (0)
• Based on which flip/flop raised, retrieves address of interrupt handler from the interrupt vector and places in PC
• OS begins executing:
  – Handler must save PC and Registers for current user process
    • Context Switch
  – Run the interrupt handler (take care of problem)
  – Interrupt handler may disable or mask interrupts during processing (don’t want to interrupt the interrupt handler)
• Upon completion, OS:
  – Resets, unmasks and/or enables interrupt flip/flops
  – Restores user process registers and program counter
  – Sets mode bit to 1 (user mode)
  – Loads PC with next instruction in user process