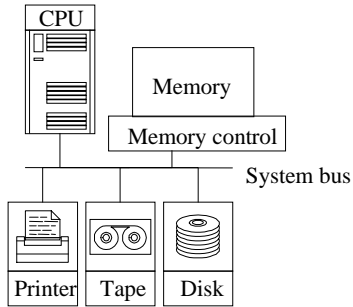


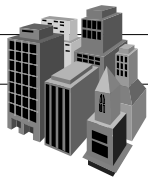
Computer System Components



Computer System Views

- **Architecture** A common taxonomy
 - Computer as seen by program
 - Low-level view
- **Organization**
 - Collection of black boxes
- **Implementation**
 - Technology used to build it

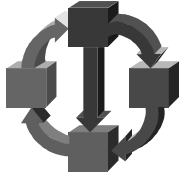
Architecture



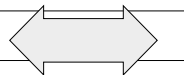
- **Instruction set**
 - Registers
 - Data types
 - Operations (including OS support)
- **I/O and interrupts**
- **Hardware protection features**

Organization

- **System bus** (one or more)
 - Arbitration, multiplexing, width
- **I/O organization**
 - Device controllers
- **Storage hierarchy**
 - Storage types, caching



Input/Output Methods



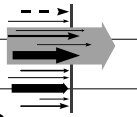
- **Programmed I/O** (synchronous)
 - Separate I/O address space?
- **Interrupts**
 - Support for asynchronous operations
- **Direct memory access**
 - Independent data transfers

Input/Output Hardware



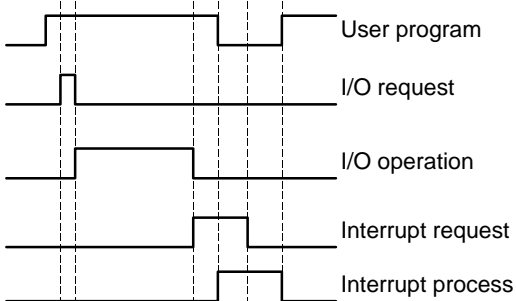
- **Device controllers**
 - Part of system (multiple devices?)
 - Integrated into each device?
- **I/O processor** (some systems)
 - Programmable subsystem
 - Manages I/O requests

Interrupt Operation



- **Permits processing overlap**
 - Simultaneous I/O and computation
- **Interrupt on I/O completion**
 - Hardware signal from device control
 - CPU saves state (PC, registers, ...)
 - PC set to service routine address

Interrupt Timing



Interrupt Processing



- **Saves state**
 - Hardware
 - On stack?
 - Which stack?
 - How much?
- **Dispatching**
 - Poll devices?
 - Vectored interrupt?
 - Disable interrupts?
 - Reentrancy?
 - Levels?

Interrupt Hardware

• **Interrupt request lines (IRQ)**

- Multiple priority levels?
- Daisy-chained through devices?

• **Vector address generation**

- Address mapped from priority?
- Passed over bus from controller?

Interrupt Vector Table

• **Fixed place in address space**

- Low addresses (Intel, others)
- High addresses (68HC11)

• **Table entry**

- Address of interrupt routine
- New processor status?
 - Priority, mode?



Interrupt Completion

• **Modify user process state?**

- Notify of I/O completion

• **Restore process state**

- Registers loaded from saved values

• **Resume user process**

- Re-enable interrupts

Non-I/O Interrupts

- **Real-time clock**
 - Interrupts at specified rate
- **Processor exceptions (traps)**
 - Arithmetic exceptions (divide by 0)
 - Memory access violations
- **Synchronous system calls**
 - Intel "INT"
 - 68HC11 "SWI"

Direct Memory Access (DMA)

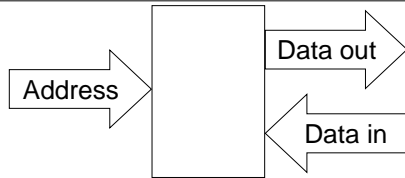
- **Simple I/O transfers**
 - One byte/word per interrupt
 - Programmed I/O for data block
- **Direct hardware control**
 - Device/memory interface (bus?)
 - Address and count registers

Waiting for I/O

- **If process can't continue . . .**
- **Run another process?**
 - Restore state of other process
- **Loop, testing status**
 - Flag set by interrupt routine?
- **WAIT instruction?** (no fetch)



Memory Structure



- Physical address space
- Data bus width
- Access/cycle time (special modes?)

Memory Address Spaces

- **Physical address space**
 - Set by width of memory address
 - Maximum amount of “real” memory
- **Virtual address space**
 - Set by width of program “address”
 - Maximum “memory” accessible

Address Space Mismatch?

- **If physical > virtual**
 - Program cannot access all memory
 - Separate block for each process?
- **If virtual > physical**
 - Can access more than there is?
 - Virtual memory, demand paging

Memory Addressing (Virtual)

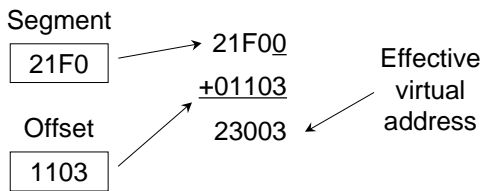
• Linear address space

- Memory addressed directly
 - Addresses 0..(N-1)

• Segmented address space

- Separate segment and offset
 - Segment in register?
 - Offset in instruction?

Segmented Addressing



- Why segment the address?
 - Shorten instruction format?
 - Fix architectural limitation?
