I/O

Todd C. Mowry
CS347
Feb. 12, 1998

Topics

• buses
• I/O devices
• I/O research at CMU
Computer system

Diagram:

- Processor
- Cache
- Memory
- Disk
- Disk controller
- Video card
- Network card
- Display
- Network

Connections:
- Processor to interrupts
- Cache to Memory
- Memory to Disk
- Disk to Disk controller
- Disk controller to Video card
- Video card to Network card
- Network card to Network

Network interrupts.
What is a bus?

A bus is a shared medium that connects the processor, memory, and I/O devices.

Consists of control and data/address wires:
- control: requests, acks, type of data (address or data)
- data lines: data, addresses
- address lines (optional): address
Bus types

Processor-memory bus
- short, fast, proprietary
- fixed number of devices with known performance

I/O bus
- longer, slower, open
- unknown number of devices with different performance
  - disk: 5 MB/s
  - 4x CDROM: 640 KB/s
- Examples: SCSI II, PCI, ISA, EISA
Traditional PC bus layout

- CPU
  - Processor bus (fast) proprietary, 33 MHz
  - External cache
  - Bus Controller Chips
    - I/O Bus (slow)
    - memory bus (fast) ISA bus, 8 MHz
  - RAM
  - Slotted I/O
    - I/O Bus (slow)
Local bus layout

- CPU
- x486
- Processor bus
- VESA local bus
- VL bus
- External cache
- I/O cards
- Graphics card
- Built-in I/O
- Bus Controller Chips
- I/O Bus
- ISA bus
- RAM
- Memory bus
PCI bus layout

- Processor
- Cache
- Bridge/memory controller
- DRAM
- LAN card
- SCSI card
- Bus interface
- Graphics card
- ISA card
- Sound card
- ISA bus
- PCI local bus
- PCI bus layout
Bus transactions

Input transaction
  • transfers data from some device to memory or processor

Output transaction
  • transfers data from memory or processor to device
1. Processor requests input on control lines and places memory address on data lines. (assume disk address handled at a higher level)

2. When memory is ready, it signals disk on control lines. Disk then transfers data.
Output transaction

1. Processor requests output on control lines and places memory address on data lines.

2. When memory is ready, it places data on data lines, indicating availability with the control lines.
Bus arbitration (one master)

1. Device asks processor for permission to use the bus for an input transaction.

2. Processor initiates transaction on behalf of device.
Daisy-chained arbitration

![Diagram of daisy-chained arbitration]

Not fair!
How processors send I/O commands

Memory mapped I/O
- special areas of the address space called I/O registers (e.g. MIPS)
- `st $2, disk_ctrl_reg` -- set disk addr
- `st $3, disk_data_reg` -- send word
- `ld $4, disk_stat_reg` -- get status

Explicit I/O instructions
- separate I/O address space (PC)
- `in $4, keyboard_reg` -- read keybrd char

Physical address space

I/O address space

Each device gets one or more addresses
How processors get I/O status

via polling a memory mapped status register (polled I/O):

```
while (status_reg != READY)  /* spin on a memory mapped status register */
    ;
```

via an exception handler (interrupt-driven I/O):

- I/O device initiates external exception (e.g., when data is ready)
- control branches to handler, which reads status and takes appropriate action (i.e., read data from device)
How data is transferred

Programmed I/O:

for (i=0; i<n; i++) {
    while (disk_stat_reg == NOTREADY) ;
    a[i] = disk_data_reg;
}

DMA (direct memory access):

• programmable DMA controller serves as bus master
  st $2 dma_stat_register   -- setup start address and count
  st $3, dma_start_register   -- start the transfer in the background

read from disk     write to memory
PC I/O space

Separate I/O space with 64K addresses (ports)
  • each device assigned one or more ports (for control & data)
  • bus control line distinguishes I/O address from regular addresses

Separate instructions read and write to ports
  • in $1, 0x60 -- read contents of keyboard data register

Some example ports:

<table>
<thead>
<tr>
<th>Address Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x60-0x6f</td>
<td>Keyboard controller</td>
</tr>
<tr>
<td>0x1f0-0x1f8</td>
<td>Hard disk controller</td>
</tr>
<tr>
<td>0x3bc-0x3f</td>
<td>Parallel port controller</td>
</tr>
<tr>
<td>0x3c0-0x3cf</td>
<td>Video adapter</td>
</tr>
<tr>
<td>0x3f8-0x3ff</td>
<td>Serial port controller</td>
</tr>
</tbody>
</table>
PC interrupts

1 byte interrupt number (k)

(1) INTR
(2) INTA
(3) Take exception to address k*4
(4) Take exception to address k*4

local bus

processor

Programmable Interrupt Controllers

IRQ 0
IRQ 1
IRQ 15
Serial ports

Serial bit stream:

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>p*</th>
</tr>
</thead>
<tbody>
<tr>
<td>start bit</td>
<td>stop stop bit</td>
<td>0 bit 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* optional data and parity bits

slow (10-20 Kbits/sec transfer rate) but long and flexible

Interrupt controller

bus

IRQ 3, 4

serial port controller (UART)

9 or 25 pin connector

mouse

modem
Parallel ports

Each parallel port has three I/O ports:
- 0x3bc: data (read/write)
- 0x3bd: status (read only)
- 0x3be: control (poll or interrupt)

faster, but shorter than serial ports
Keyboard

interrupt handler transfers data from keyboard controller to BIOS keyboard buffer

Interrupt controller

bus

IRQ x

keyboard controller

0x1e (scan code for ‘a’)

BIOS keyboard buffer

‘0x1e61’

keyboard

‘a’
Display

- Heating filament
- Focusing system
- Control grid
- Vertical deflection
- Horizontal deflection
- Electron beam
- Phosphor coated screen
Raster scan

vertical:

horizontal:

horizontal retrace

vertical retrace
Frame buffer (grayscale)
The RGB color space
Frame buffer with color map

frame buffer

color map

display

red

yellow