Επικοινωνία του Επεξεργαστή με Περιφερειακές Συσκευές

13b (§13.4-7) – 8-10 Μαΐου 2023 – Μανόλης Κατεβαίνης

© copyright University of Crete – https://www.csd.uoc.gr/~hy225/23a/copyright.html
Instruction Set Architecture for I/O

° Some machines have special input and output instructions

° Alternative model (used by MIPS):
  • Input: ~ reads a sequence of bytes
  • Output: ~ writes a sequence of bytes

° Memory also a sequence of bytes, so use loads for input, stores for output
  • Called “Memory Mapped Input/Output”
  • A portion of the address space dedicated to communication paths to Input or Output devices (no memory there)
Memory Mapped I/O

- Certain addresses are not regular memory
- Instead, they correspond to registers in I/O devices

address 0

0xFFFF0000

0xFFFFFFFF

---

cmd reg.
data reg.
Example: keyboard... if only a Data Register:

What did the user type?

<table>
<thead>
<tr>
<th>Time (s)</th>
<th>Data Reg.</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>read</td>
<td>'t'</td>
</tr>
<tr>
<td>0.5</td>
<td>read</td>
<td>'t'</td>
</tr>
<tr>
<td>1.0</td>
<td>read</td>
<td>'o'</td>
</tr>
<tr>
<td>1.5</td>
<td>read</td>
<td>'o'</td>
</tr>
<tr>
<td>2.0</td>
<td>read</td>
<td>'o'</td>
</tr>
</tbody>
</table>
Example: keyboard... if only a Data Register:

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?

What did the user type?
Processor Checks Status before Acting

- Path to device generally has 2 registers:
  - 1 register says it’s OK to read/write (I/O ready), often called **Control Register**
  - 1 register that contains data, often called **Data Register**

- Processor reads from Control Register in loop, waiting for device to set Ready bit in Control reg to say its OK (0 \(\rightarrow\) 1)

- Processor then loads from (input) or writes to (output) data register
  - Load from device/Store into Data Register resets Ready bit (1 \(\rightarrow\) 0) of Control Register

"Synchronization" between parallel threads of control, or "Flow Control":
- on input: New Data have arrived;
- on Output: space for new data now available.
May also include:
- data error status
- overflow status

"Polling"

"Busy wait" if done continuously; else, poll multiple devices on every interrupt from the real-time clock (usu. 50-120 Hz)

"Batching"
I/O Address Pages must be non-cacheable.

If they were allowed to be cached...

- Transitional ("non-coherent") caching does **not** work when other devices (I/O, other processor cores) access memory independently.

- Note: write-through is a "half-solution": works for output, but not for input...
I/O/Communication Registers ≠ Normal Memory Semantics (non-shared)

Normal Memory:

Some word:

Write
read x
read x
read x

Time

Read always yields the last written value

Write here

Other words NOT changed

Writes only affect the word being written

I/O/Communication Registers:

Some word:

Read x
Read x
Read x

Potentially ≠

Successive reads from a same location (without any intervening writes from processor) may yield different values.

"Volatile" declaration, in C

Data/status

Read/write here

May change other words too!

"Side-Effects"
• Όταν επιτρέπονται λειτουργίες εκτός σειράς (out-of-order), για αύξηση επίδοσης, ο απλοϊκός συγχρονισμός δεν επαρκεί

• Τάση προς Release Consistency, με προσθήκη εντολών Memory Fence (Memory Barrier)
What is the alternative to polling?

° Wasteful to have processor spend most of its time “spin–waiting” for I/O to be ready

° Wish we could have an unplanned procedure call that would be invoked only when I/O device is ready

° Solution: use exception mechanism to help I/O. Interrupt program when I/O ready, return when done with data transfer
I/O Interrupt

° An I/O interrupt is like an overflow exceptions except:
  • An I/O interrupt is “asynchronous”
  • More information needs to be conveyed

° An I/O interrupt is asynchronous with respect to instruction execution:
  • I/O interrupt is not associated with any instruction, but it can happen in the middle of any given instruction
  • I/O interrupt does not prevent any instruction from completion
Definitions for Clarification

° **Exception**: signal marking that something “out of the ordinary” has happened and needs to be handled

° **Interrupt**: asynchronous exception

° **Trap**: synchronous exception

° **Note**: These are different from the book’s definitions.
Interrupt Driven Data Transfer

1. I/O interrupt
2. Save PC
3. Interrupt service address
4. Read
5. Store
6. JR

Memory

User program

Interrupt service routine
Questions Raised about Interrupts

° Which I/O device caused exception?
  • Needs to convey the identity of the device generating the interrupt
    Cause register, or Vectored Interrupts

° Can avoid interrupts during the interrupt routine?
  • What if more important interrupt occurs while servicing this interrupt?
  • Allow interrupt routine to be entered again?

° Who keeps track of status of all the devices, handle errors, know where to put/supply the I/O data?
Χρονικό κόστος Διακοπών και Δειγματοληψίας

– Παρ’ ότι μιά εξαίρεση/διακοπή μοιάζει με άλμα, εντούτοις...

• Σχεδόν πάντα αυτά κοστίζουν ~1000 κύκλους ρολογιού (!)
  – ένα μικρό μυστήριο το γιατί ακόμα ισχύει αυτό στη μεγάλη
    πλειοψηφία, παρ’ όλο που πολλοί προσπάθησαν να το μειώσουν
  – μάλλον το άθροισμα κάμποσων παραγόντων, κυρίως:
    – σώσιμο/επαναφορά καταχωρητών διακοπείσας διεργασίας
    – αναζήτηση αιτίας / “housekeeping”
    – αστοχίες κρυφών μνημών και TLB

• Δειγματοληψίες επίσης κοστίζουν: non-cacheable I/O reg.
  – μία τεχνική: όταν διακοπή από real-time clock (~50-120 Hz), τότε
    το Λειτουργικό δειγματοληπτεί πολλές περιφερειακές μαζί
Fast Devices need I/O Buffer — not just a Register

... amortize the cost of interrupt over many data

example:

(just) 1 Gbit/s

\[ \Rightarrow \]

1 bit every 1 ns

\[ \Rightarrow \]

32 bits every 32 ns

one "Register"

- Cost to poll status register
  (non-cacheable, off-chip)
  usually ≥ DRAM access
  usually ~ 100 ns (or more)
- Then read the data register
  similarly ~ 100 ns
- Cost of interrupt + kernel interrupt handlers usually ~ 1 μs (1000 ns)!
Direct Memory Access (DMA)

Alternatives for cacheability:
- DMA onto non-cacheable memory pages... too slow when processor processes the I/O data
- Flush the Cache before/after I/O DMA... quite expensive operation < total flush?
- Selective Flush?
- Cache-Coherent DMA ← good! → next chapter...