Universiti Malaysia PAHANG
Engineering • Technology • Creativity

BCN 1043

COMPUTER ARCHITECTURE & ORGANIZATION

By
Dr. Mritha Ramalingam

Faculty of Computer Systems & Software Engineering
mritha@ump.edu.my

AUTHORS

- **Dr. Mohd Nizam Mohmad Kahar** (mnizam@ump.edu.my)
- **Jamaludin Sallim** (jamal@ump.edu.my)
- **Dr. Syafiq Fauzi Kamarulzaman** (syafiq29@ump.edu.my)
- **Dr. Mritha Ramalingam** (mritha@ump.edu.my)

Faculty of Computer Systems & Software Engineering

BCN1043

COMPUTER ARCHITECTURE & ORGANIZATION

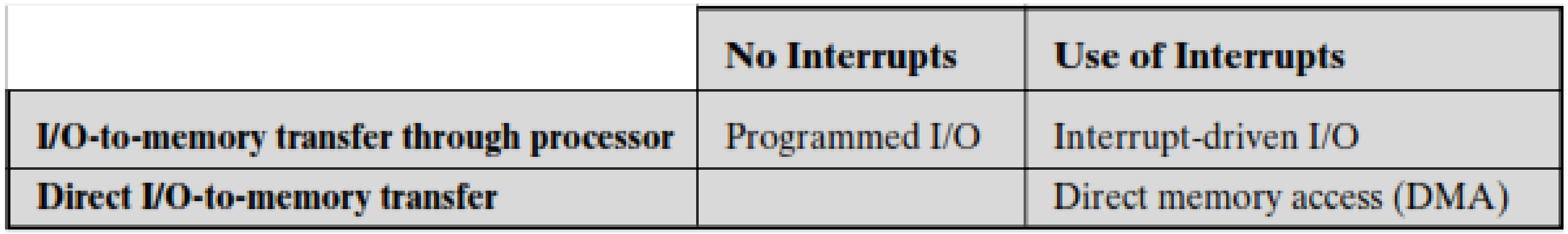
Chapter 6 continues…

Techniques of I/O

1) **Programmed I/O (CPU has direct control over I/O):** The processor executes a program that gives it direct control of the I/O operation. When the processor issues a command to the I/O module, it must wait until the I/O operation is complete.
2) **Interrupt-driven I/O:** The CPU issues commands then proceeds with its normal work until interrupted by I/O device on completion of its current work.
3) **DMA:** Data exchange occurs between I/O module and main memory without CPU's involvement

**Programmed* & *Interrupt-driven I/O*, the processor is responsible for extracting data from main memory for output and storing data in main memory for input.

Techniques of I/O



	No Interrupts	Use of Interrupts
I/O-to-memory transfer through processor	Programmed I/O	Interrupt-driven I/O
Direct I/O-to-memory transfer		Direct memory access (DMA)

Programmed I/O

- CPU having direct control over I/O (initiative by the CPU)
  - Sensing status
  - Read/write commands
  - Transferring data
- CPU wait for I/O module to complete operation
- The processor is faster than the I/O module – time of CPU is wasted

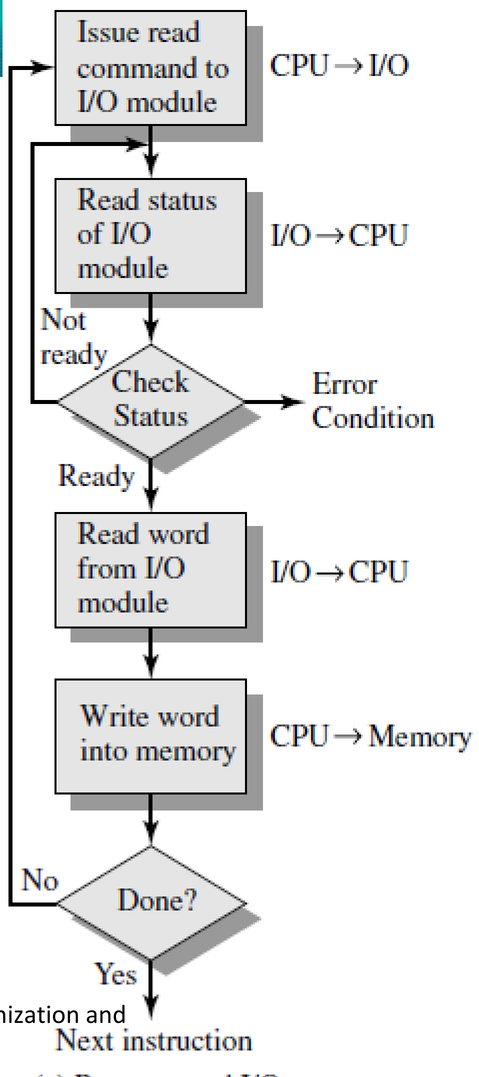
Programmed I/O - details

- CPU requests I/O operation
- I/O module performs operation
- I/O module sets status bits
- CPU checks status bits periodically
- I/O module does not inform CPU directly
- I/O module does not interrupt CPU
- CPU may wait or come back later

Programmed I/O - flowchart

- Processor requests operation with commands sent to I/O module
  - Control - telling a peripheral what to do. (e.g. spin up disk)
  - Test - used to check condition of I/O module or device (e.g. power? Error?)
  - Read - obtains data from peripheral so processor can read it from the data bus
  - Write - sends data using the data bus to the peripheral

    (R/W: module transfer data via buffer from/to device)
- I/O module performs operation
- When completed, I/O module updates its status registers



(a) Programmed I/O

Source: William Stallings, Computer Organization and Architecture,9th Edn

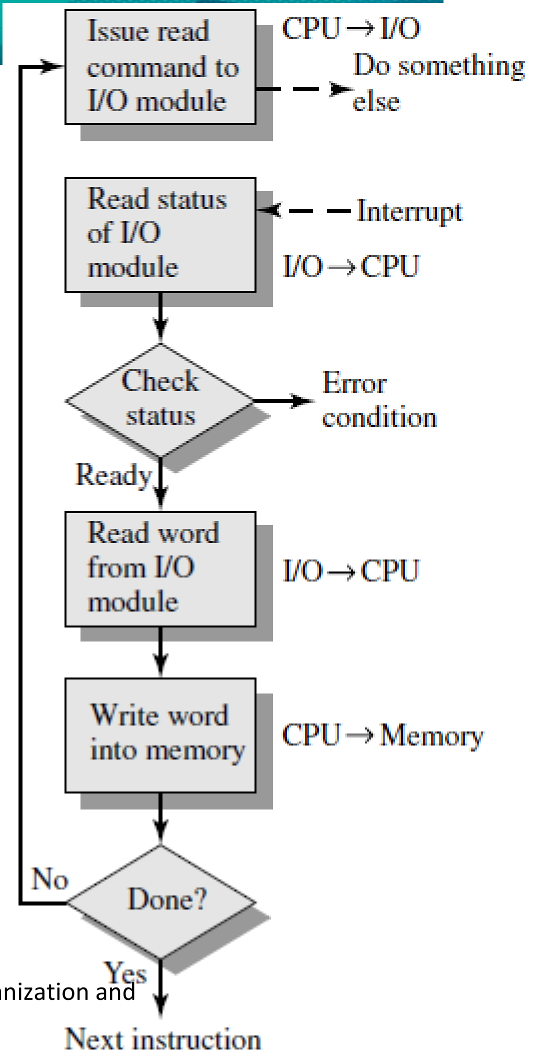
Programmed I/O (cont…)

- I/O module does not inform CPU directly (that is through memory - memory mapped I/O and isolated I/O)
- Wastes CPU time because typically processor is much faster than I/O
  - CPU acts as a bridge for moving data between I/O module and main memory, i.e., every piece of data goes through CPU
  - CPU waits for I/O module to complete operation

Interrupt Driven I/O

- Overcomes CPU waiting time:

- Requires setup code and interrupt service routine
- No repeated CPU checking of device
- I/O module interrupts when ready
- requires CPU to be go between for moving data

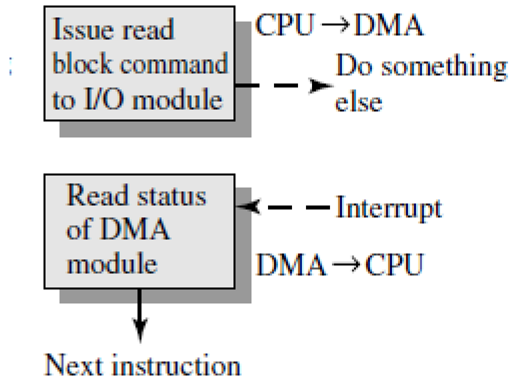


(b) Interrupt-Driven I/O

Source: William Stallings, Computer Organization and Architecture,9th Edn

Direct Memory Access (DMA)

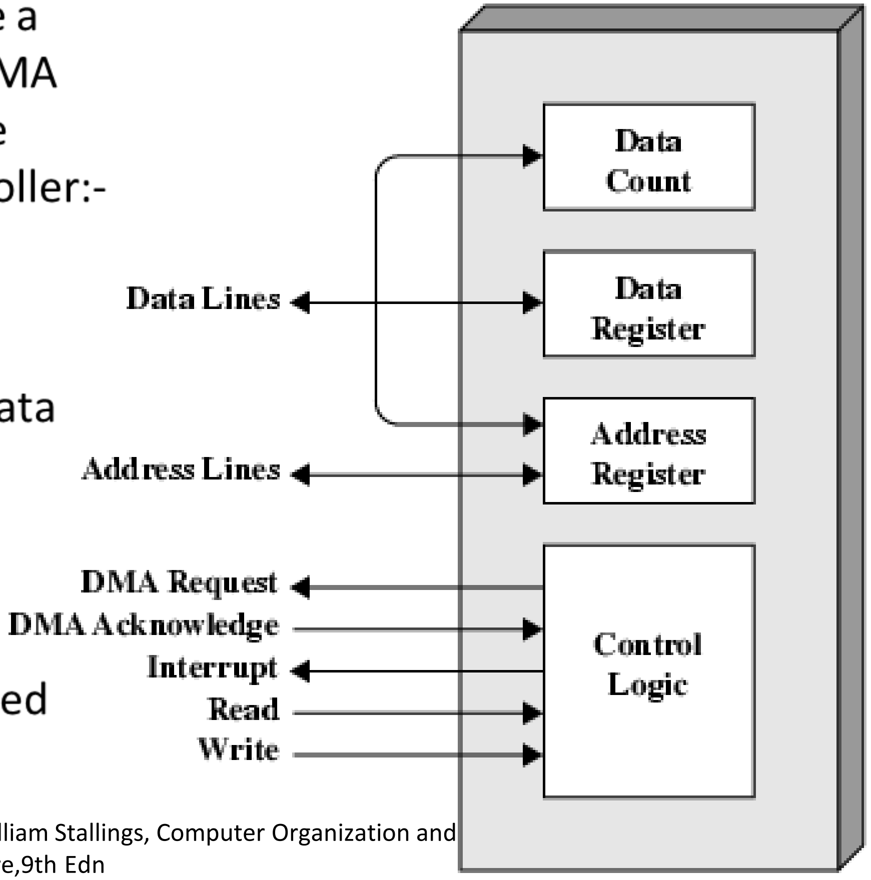
- Interrupt driven and programmed I/O require active CPU intervention (All data must pass through CPU)
- Transfer rate is limited by processor's ability to service the device
- CPU is tied up managing I/O transfer


Issue read block command to I/O module
CPU → DMA
Do something else
Read status of DMA module
Interrupt
DMA → CPU
Next instruction

(c) Direct Memory Access

Source: William Stallings, Computer Organization and Architecture,9th Edn

- When the processor wishes to read or write a block of data, it issues a command to the DMA module, by sending to the DMA module the following information. CPU tells DMA controller:-
  - Read/Write
  - Device address
  - Starting address of memory block for data
  - Amount of data to be transferred
- CPU carries on with other work
- DMA controller deals with transfer
- DMA controller sends interrupt when finished



Source: William Stallings, Computer Organization and Architecture,9th Edn

DMA Transfer Cycle Stealing

- It use the bus when the processor does not need it or the processor suspend it operation.

- DMA controller takes over bus for a cycle
- Transfer of one word of data
- Not an interrupt
  - CPU does not switch context
- CPU suspended just before it accesses bus
  - i.e. before an operand or data fetch or a data write
- Slows down CPU but not as much as CPU doing transfer

Multiple interrupt

- Larger number of interrupt generators
  - Keyboard, mouse, hard-disk, clock, memory error …
  - How to identify the device?
- Which interrupt is important?
  - Which should be handled first if both of them occur simultaneously?

Multiple Interrupt - Design Issues

- How do you identify the module issuing the interrupt?
  1) multiple interrupt line
  2) software poll
  3) daisy chain
  4) bus arbitration
- Which interrupts do we handled first?
  – priority

1. Multiple interrupt line

- Device Identification:
  - Different interrupt line for each module (see figure)
  - Limits number of devices
  - Even with this method, there are often multiple interrupts still on a single interrupt lined
- Handle which interrupt?
  - Priority is set by hardware

2. Software poll

- Device identification:
  - Single interrupt line - when an interrupt happens, CPU checks who needs attention
  - How? interrupt service routine poll each I/O module to determine which module caused the interrupt (CPU asks each module in turn)
  - Slow

Software polling limitation

- CPU time is wasted,
- slow, as it checks the status of all I/O devices before it comes back to check any given one again,
- Can be as fast enough
- priority of the device is determined by the order in the polling loop, but it is possible to change it via software.

3. Daisy Chain (Hardware poll)

- It is significantly faster than a pure software approach.
- Daisy chaining is used for level sensitive interrupts, which act like a wired 'OR' gate. Any requesting device can take the interrupt line low, and keep it asserted low until it is serviced.

Daisy Chain (Hardware poll)

- Device identification
  - Interrupt Acknowledge sent down a chain
  - Module responsible places unique vector on bus

- Handling multiple interrupt
  - Priority is set by order in which interrupt acknowledge gets to I/O modules, i.e., order of devices on the chain
  - Each device is connected to the same interrupt request line, but the interrupt acknowledge line is passed through each device, from the highest priority device first, to the lowest priority device last.

4. Bus Arbitration

- Device identification:
  - Each device has a request line connected to a centralized arbiter that determines which device will be granted access to the bus.
  - Device that "wins" places vector on bus uniquely identifying interrupt
- Handling multiple interrupt
  - Priority is set by priority in arbitration.
  - The order may be fixed by the order of connection (priority encoded), or it may be determined by some algorithm preloaded into the arbiter.

How do you deal with multiple interrupts?

- Each interrupt line has a priority. Higher priority lines can interrupt lower priority lines. If bus mastering only current master can interrupt
- With multiple lines, the processor just picks the interrupt line with the highest priority.
- With software polling, the order in which modules are polled determines their priority.
- Similarly, the order of modules on a daisy chain determines their priority.
- Bus arbitration employ priority scheme,

Review questions?

- Give examples of an external devices?
- Based on the above questions, determine whether the external devices are Input, output or both?
- Describe the term I/O devices?
- Name the require device to allow communication between external device and processor.
- Can we connect the external devices directly to the CPU? Justify your answer?
- Name the categories of external devices.

- What are the lines required to establish connection from external devices to the I/O modules. Describe the function of each lines?

Will continue…