COMPUTER ORGANIZATION AND ARCHITECTURE (COA)

EET 2211
4TH SEMESTER – CSE & CSIT
CHAPTER 7, LECTURE 27

TOPICS TO BE COVERED

1. External Devices
2. I/O Modules
3. Programmed I/O
4. Interrupt-Driven I/O
5. Direct Memory Access
6. Direct Cache Access

LEARNING OBJECTIVES

- Explain the use of I/O modules as part of a computer organization.
- Understand programmed I/O and discuss its relative merits.
- Present an overview of the operation of direct memory access.
- Present an overview of direct cache access.

DIRECT MEMORY ACCESS (DMA)

Drawbacks of Programmed and Interrupt- Driven I/O

Interrupt-driven I/O, though more efficient than simple programmed I/O, still requires the active intervention of the processor to transfer data between memory and an I/O module, and any data transfer must traverse a path through the processor. Thus, both these forms of I/O suffer from two inherent drawbacks:

1. The I/O transfer rate is limited by the speed with which the processor can test and service a device.

2. The processor is tied up in managing an I/O transfer; a number of instructions must be executed for each I/O transfer

EXAMPLE

Consider the transfer of a block of data.

- Using simple Programmed I/O, the processor is dedicated to the task of I/O and can move data at a rather high rate, at the cost of doing nothing else.
- Interrupt I/O frees up the processor to some extent at the expense of the I/O transfer rate. Nevertheless, both methods have an adverse impact on both processor activity and I/O transfer rate.
- When large volumes of data are to be moved, a more efficient technique is required: direct memory access (DMA).

DMA FUNCTIONS

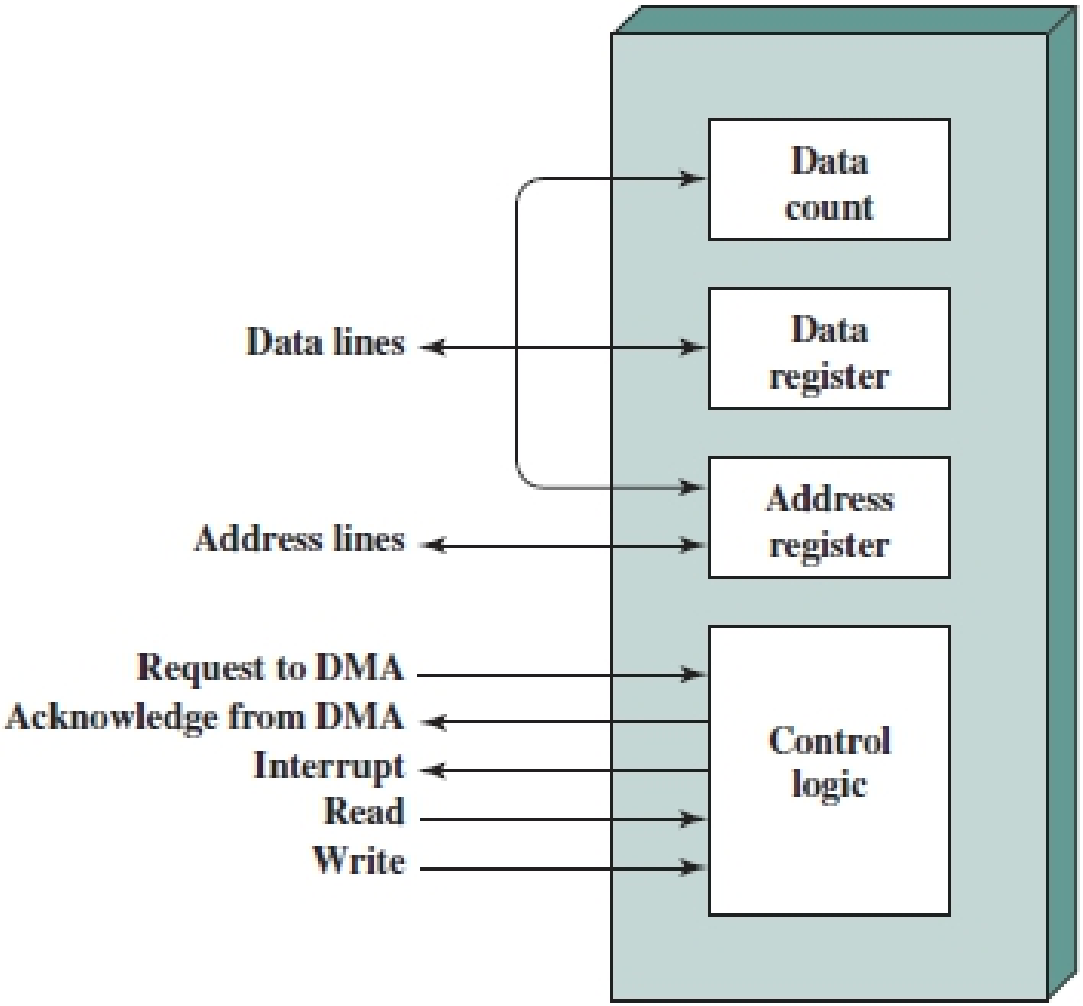
- DMA involves an additional module on the system bus.
- The DMA module is capable of mimicking the processor and, indeed, of taking over control of the system from the processor.
- It needs to do this to transfer data to and from memory over the system bus.
- For this purpose, the DMA module must use the bus only when the processor does not need it, or it must force the processor to suspend operation temporarily.
- The latter technique is more common and is referred to as *cycle stealing*, *because the DMA module in effect steals a bus cycle*.

WORKING PRINCIPLE

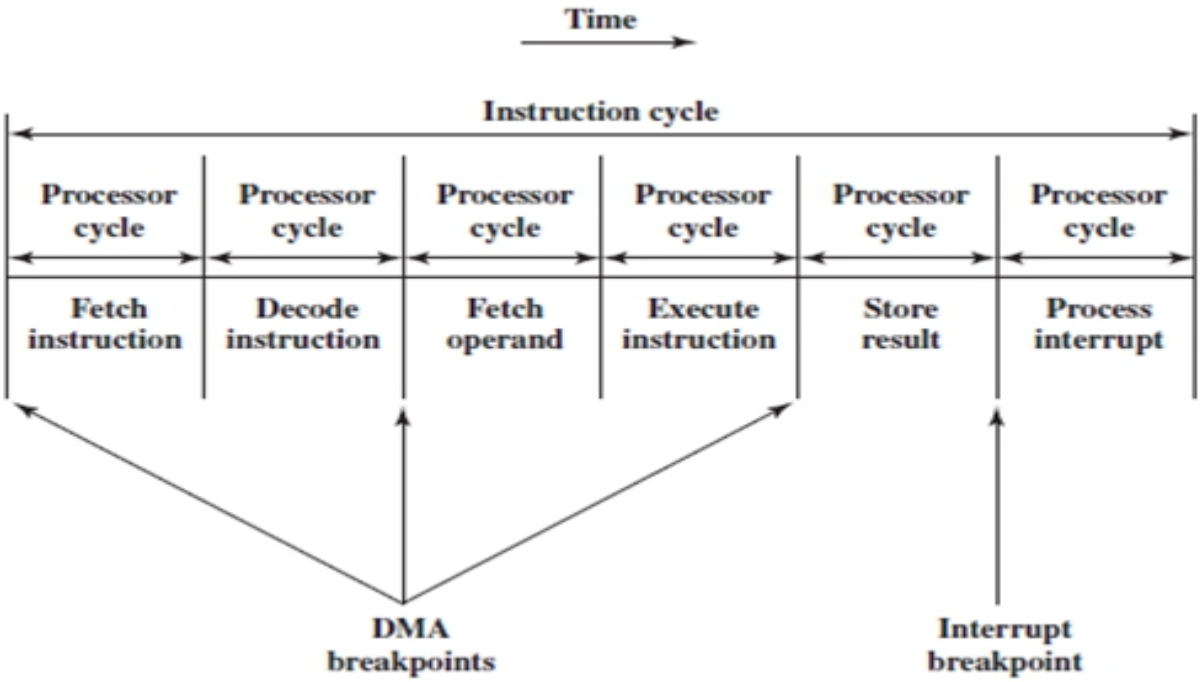
- When the processor wishes to read or write a block of data, it issues a command to the DMA module, by sending to the DMA module the following information:
  - Whether a read or write is requested, using the read or write control line between the processor and the DMA module.
  - The address of the I/O device involved, communicated on the data lines.
  - The starting location in memory to read from or write to, communicated on the data lines and stored by the DMA module in its address register.
  - The number of words to be read or written, again communicated via the data lines and stored in the data count register.

- The processor then continues with other work.
- The DMA module transfers the entire block of data, one word at a time, directly to or from memory, without going through the processor.
- When the transfer is complete, the DMA module sends an interrupt signal to the processor.
- Thus, the processor is involved only at the beginning and end of the transfer

TYPICAL DMA BLOCK DIAGRAM



DMA and Interrupt Breakpoints during an Instruction Cycle



Where in the instruction cycle the processor may be suspended?

In each case, the processor is suspended just before it needs to use the bus.

The DMA module then transfers one word and returns control to the processor.

Note that this is not an interrupt; the processor does not save a context and do something else. Rather, the processor pauses for one bus cycle.

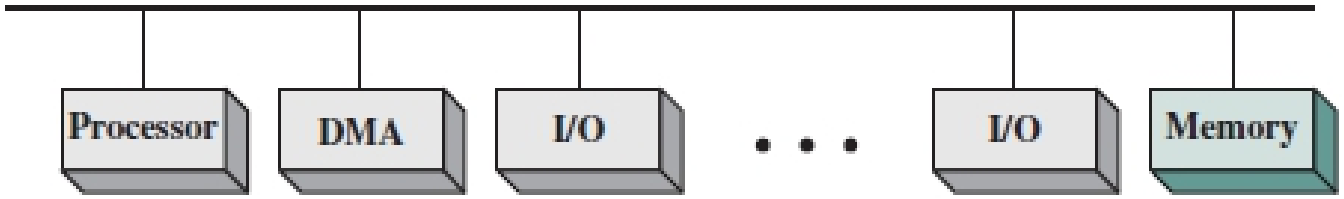
The overall effect is to cause the processor to execute more slowly.

For a multiple- Word I/O transfer, **DMA is far more efficient** than interrupt-driven or programmed I/O.

DMA CONFIGURATIONS

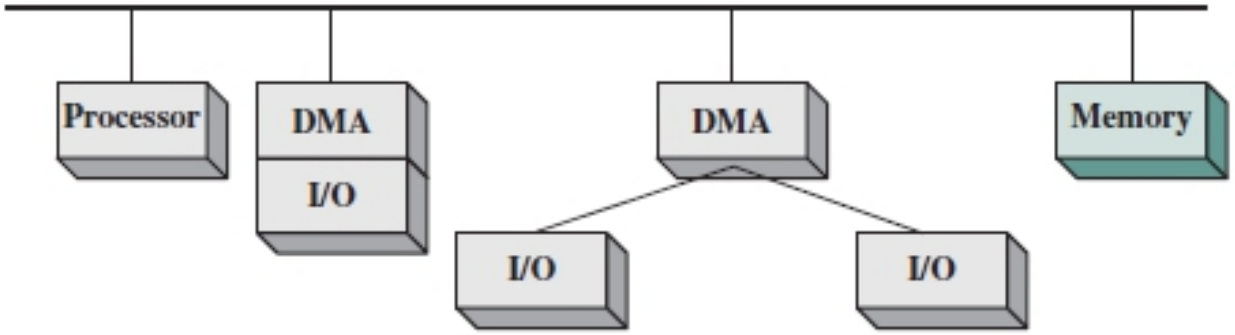
1. Single-bus, detached DMA
2. Single-bus, integrated DMA-I/O
3. I/O bus

Single-bus, detached DMA



- All modules share the same system bus.
- The DMA module, acting as a surrogate processor, uses programmed I/O to exchange data between memory and an I/O module through the DMA module.
- This configuration is **inexpensive**
- It is **inefficient**
- With processor-controlled programmed I/O, each transfer of a word consumes **two bus cycles.**

Single-bus, integrated DMA-I/O



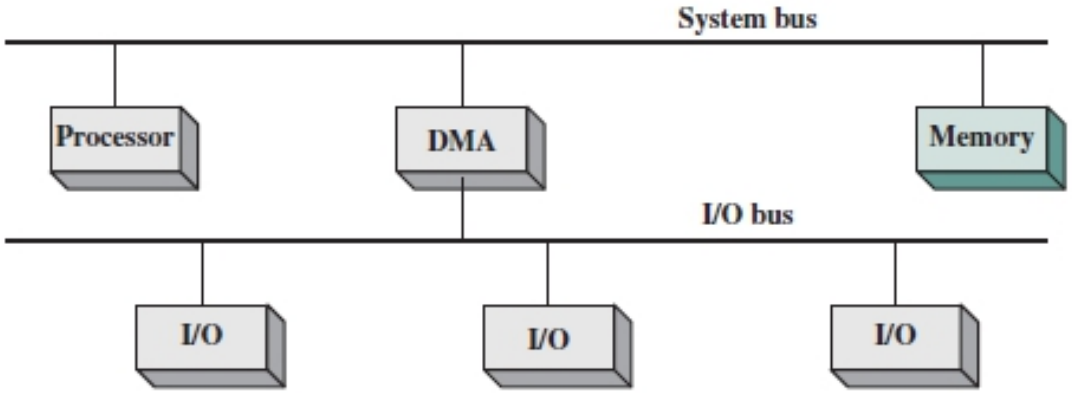
The number of required bus cycles can be cut substantially by integrating the DMA and I/O functions.

This means that there is a path between the DMA module and one or more I/O modules that does not include the system bus.

The DMA logic may actually be a part of an I/O module, or it may be a separate module that controls one or more I/O modules.

The system bus that the DMA module shares with the processor and memory is used by the DMA module only to exchange data with memory. The exchange of data between the DMA and I/O modules takes place off the system bus.

I/O bus



•This concept can be taken one step further by connecting I/O modules to the DMA module using an I/O bus .

•This reduces the number of I/O interfaces in the DMA module to one and provides for an easily expandable configuration.

•The system bus that the DMA module shares with the processor and memory is used by the DMA module only to exchange data with memory. The exchange of data between the DMA and I/O modules takes place off the system bus.

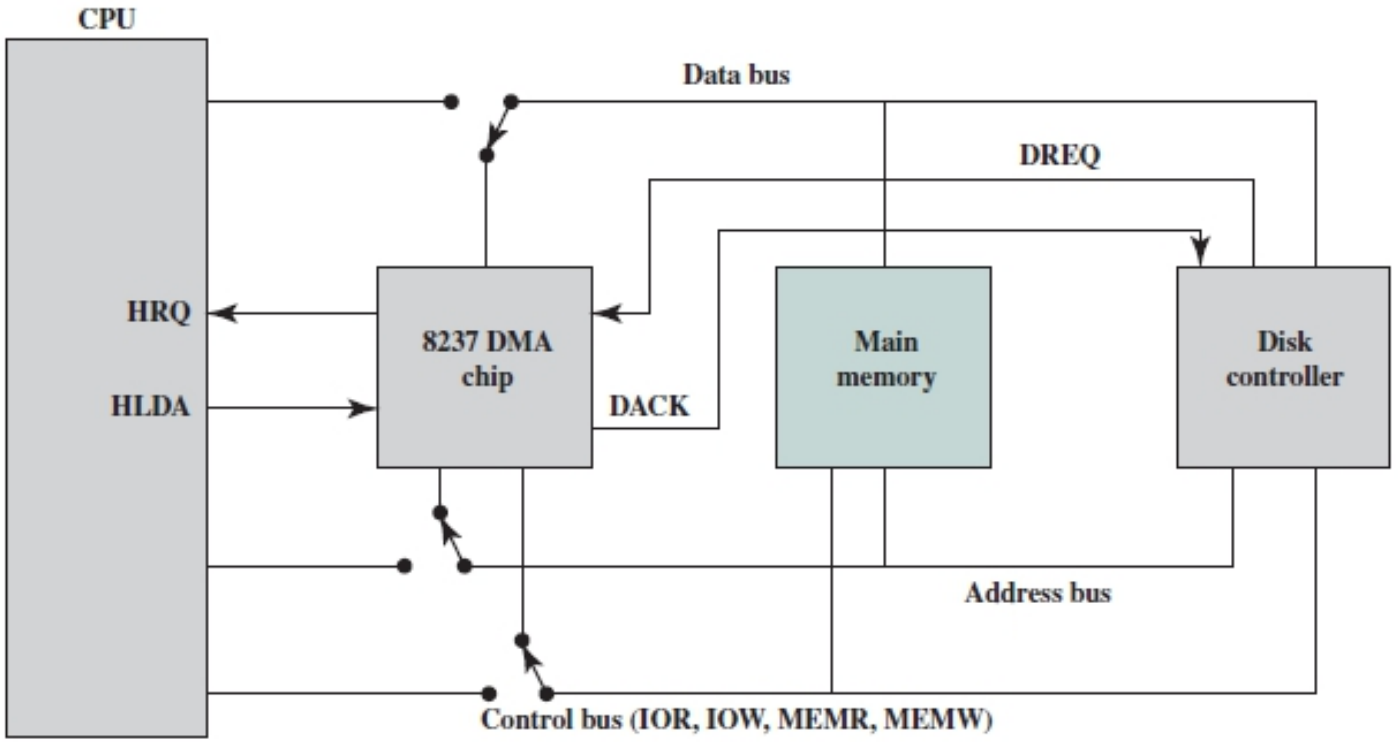
Intel 8237A DMA Controller

- The Intel 8237A DMA controller interfaces to the 80 X86 family of processors and to DRAM memory to provide a DMA capability.

- When the DMA module needs to use the system buses (data, address, and control) to transfer data, it sends a signal called HOLD to the processor.

- The processor responds with the HLDA (hold acknowledge) signal, indicating that the DMA module can use the buses.

Intel 8237A DMA Controller



DACK = DMA acknowledge
DREQ = DMA request
HLDA = HOLD acknowledge
HRQ = HOLD request

Example: if the DMA module is to transfer a block of data from memory to disk, it will do the following:

1. **The peripheral device (such as the disk controller) will request the service of DMA by pulling DREQ (DMA request) high.**

2. **The DMA will put a high on its HRQ (hold request), signaling the CPU through its HOLD pin that it needs to use the buses.**

3. **The CPU will finish the present bus cycle (not necessarily the present instruction) and respond to the DMA request by putting high on its HDLA (hold acknowledge), thus telling the 8237 DMA that it can go ahead and use the buses to perform its task. HOLD must remain active high as long as DMA is performing its task.**

4. **DMA will activate DACK (DMA acknowledge), which tells the peripheral device that it will start to transfer the data.**

5. **DMA starts to transfer the data from memory to peripheral by putting the address of the first byte of the block on the address bus and activating MEMR, thereby reading the byte from memory into the data bus; it then activates IOW to write it to the peripheral. Then DMA decrements the counter and increments the address pointer and repeats this process until the count reaches zero and the task is finished.**

6. **After the DMA has finished its job it will deactivate HRQ, signaling the CPU that it can regain control over its buses.**

Contd.

- The 8237 DMA is known as a ***fly-by*** DMA controller. This means that the data being moved from one location to another does not pass through the DMA chip and is not stored in the DMA chip.

- Therefore, the DMA can only transfer data between an I/O port and a memory address, and not between two I/O ports or two memory locations.

- The DMA chip can perform a memory-to-memory transfer via a register

Control/Command Registers

The 8237 has a set of five control/command registers to program and control DMA operation over one of its channels

- **Command:** The processor loads this register to control the operation of the DMA. D0 enables a memory-to-memory transfer, in which channel 0 is used to transfer a byte into an 8237 temporary register and channel 1 is used to transfer the byte from the register to memory. When memory- To-memory is enabled, D1 can be used to disable increment/decrement on channel 0 so that a fixed value can be written into a block of memory. D2 enables or disables DMA.

- **Status:** The processor reads this register to determine DMA status. Bits D0–D3 are used to indicate if channels 0–3 have reached their TC terminal count). Bits D4–D7 are used by the processor to determine if any channel has a DMA request pending.

- **Mode:** The processor sets this register to determine the mode of operation of the DMA. Bits D0 and D1 are used to select a channel. The other bits select various operation modes for the selected channel. Bits D2 and D3 determine if the transfer is from an I/O device to memory (write) or from memory to I/O (read), or a verify operation. If D4 is set, then the memory address register and the count register are reloaded with their original values at the end of a DMA data transfer. Bits D6 and D7 determine the way in which the 8237 is used. In single mode, a single byte of data is transferred. Block and demand modes are used for a block transfer, with the demand mode allowing for premature ending of the transfer. Cascade mode allows multiple 8237s to be cascaded to expand the number of channels to more than 4.

- **Single Mask:** The processor sets this register. Bits D0 and D1 select the channel. Bit D2 clears or sets the mask bit for that channel. It is through this register that the DREQ input of a specific channel can be masked (disabled) or unmasked (enabled). While the command register can be used to disable the whole DMA chip, the single mask register allows the programmer to disable or enable a specific channel.

- **All Mask:** This register is similar to the single mask register except that all four channels can be masked or unmasked with one write operation.

In addition, the 8237A has eight data registers: one memory address register and one count register for each channel. The processor sets these registers to indicate the location of size of main memory to be affected by the transfers.

INTEL 8237A REGISTERS

Bit	Command	Status	Mode	Single Mask	All Mask
D0	Memory-to-memory E/D	Channel 0 has reached TC	Channel select	Select channel mask bit	Clear/set channel 0 mask bit
D1	Channel 0 address hold E/D	Channel 1 has reached TC			Clear/set channel 1 mask bit
D2	Controller E/D	Channel 2 has reached TC	Verify/write/read transfer	Clear/set mask bit	Clear/set channel 2 mask bit
D3	Normal/compressed timing	Channel 3 has reached TC		Not used	Clear/set channel 3 mask bit
D4	Fixed/rotating priority	Channel 0 request	Auto-initialization E/D		Not used
D5	Late/extended write selection	Channel 0 request	Address increment/decrement select		
D6	DREQ sense active high/low	Channel 0 request			
D7	DACK sense active high/low	Channel 0 request	Demand/single/block/cascade mode select		

E/D = enable/disable
TC = terminal count

THANK YOU