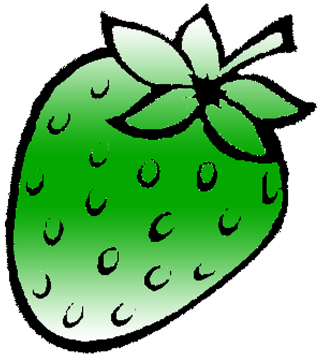
STRAWBERRY



/strawberrydevelopers

/strawberry_app

For more visit:

Strawberrydevelopers.weebly.com

UNIT VII:
INPUT & OUTPUT UNIT

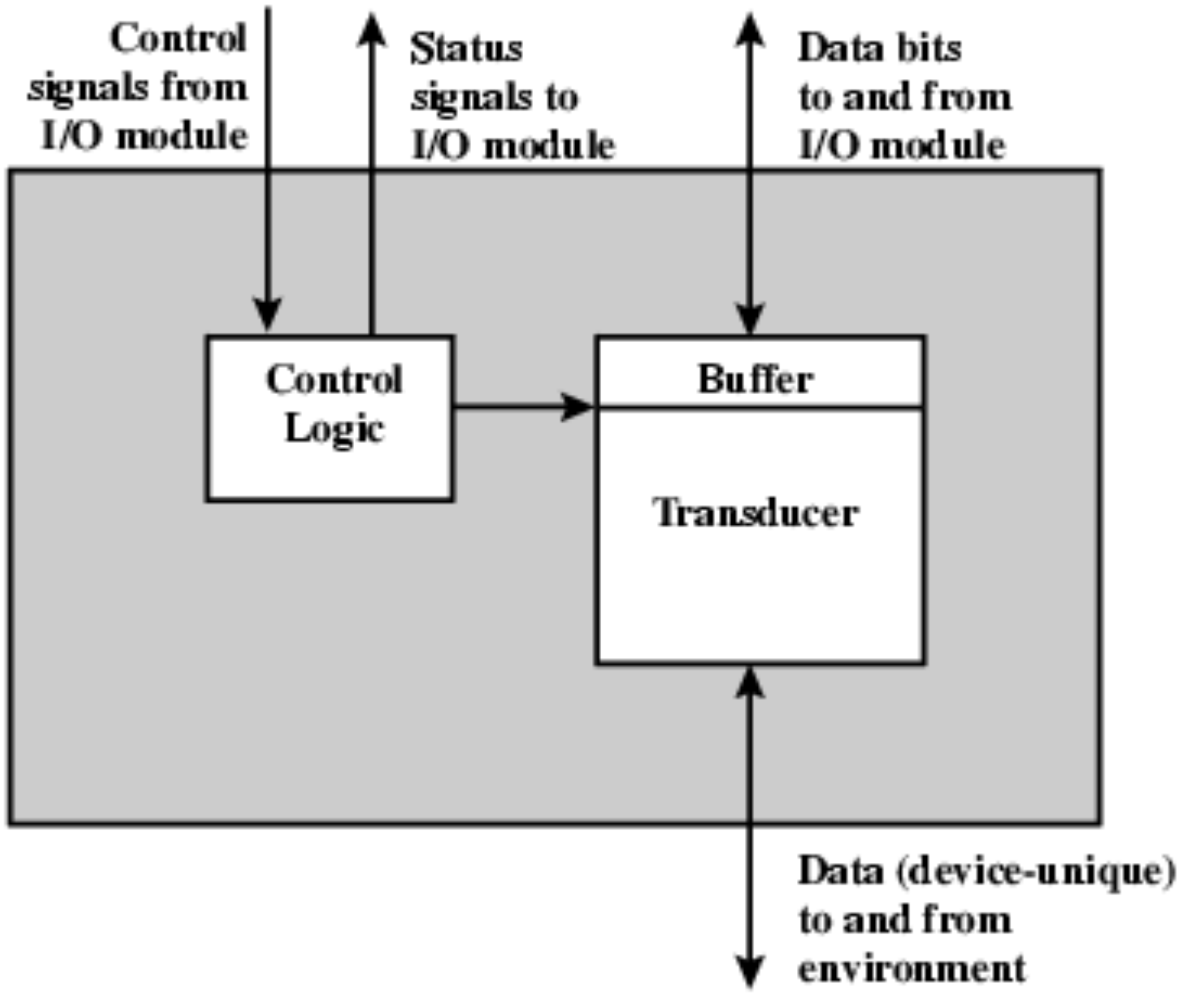
Agenda

- I/O Devices
- Disk drive
- Device Drivers
- I/O Modules: Assignment 2
- Programmed I/O
- Interrupt
- DMA
- I/O Channels & Processors

I/O Devices: Categories

- **Human readable**
  - Used to communicate with the user
  - Printers
  - Video display terminals
    - Display
    - Keyboard
    - Mouse
- **Machine readable**
  - Used to communicate with electronic equipment
  - Disk and tape drives
  - Sensors
  - Controllers
  - Actuators
- **Communication**
  - Used to communicate with remote devices
  - Digital line drivers
  - Modems

External Device Block Diagram



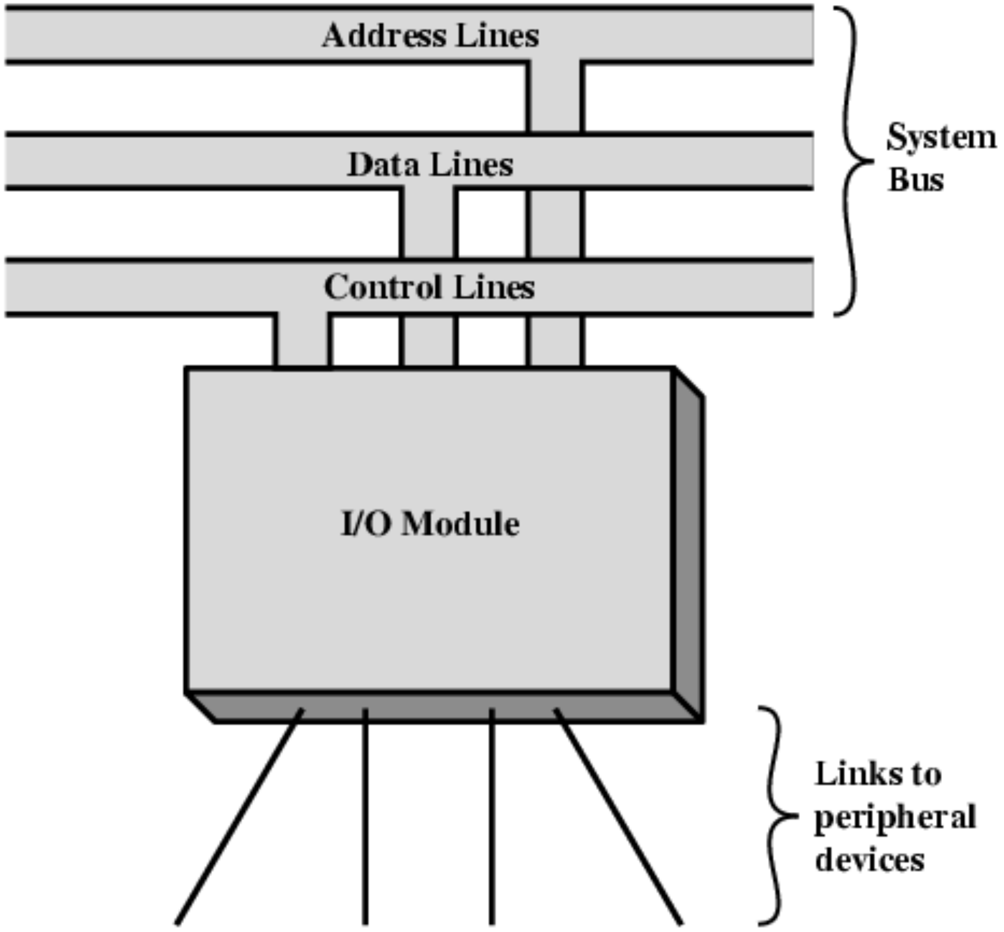
Control signals from I/O module
Status signals to I/O module
Data bits to and from I/O module
Control Logic
Buffer
Transducer
Data (device-unique) to and from environment

Input/Output Problems

- Wide variety of peripherals
  - Delivering different amounts of data
  - At different speeds
  - In different formats
- All slower than CPU and RAM
- Need I/O modules

Input/Output Module

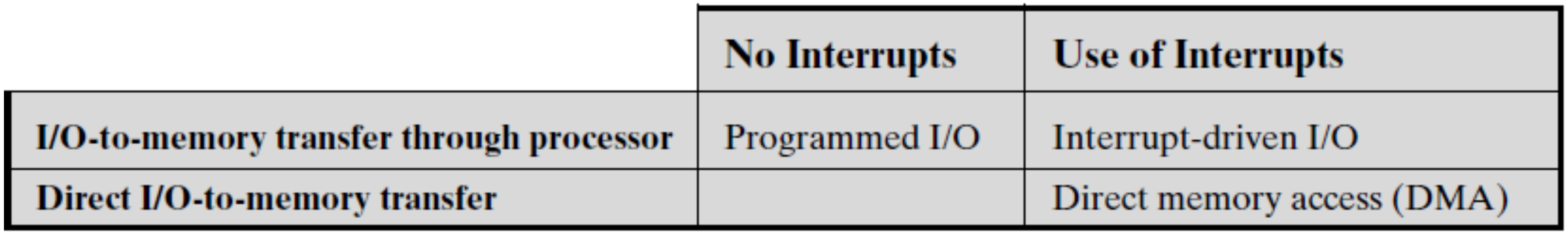
- Interface to CPU and Memory
- Interface to one or more peripherals



Generic Model of I/O Module

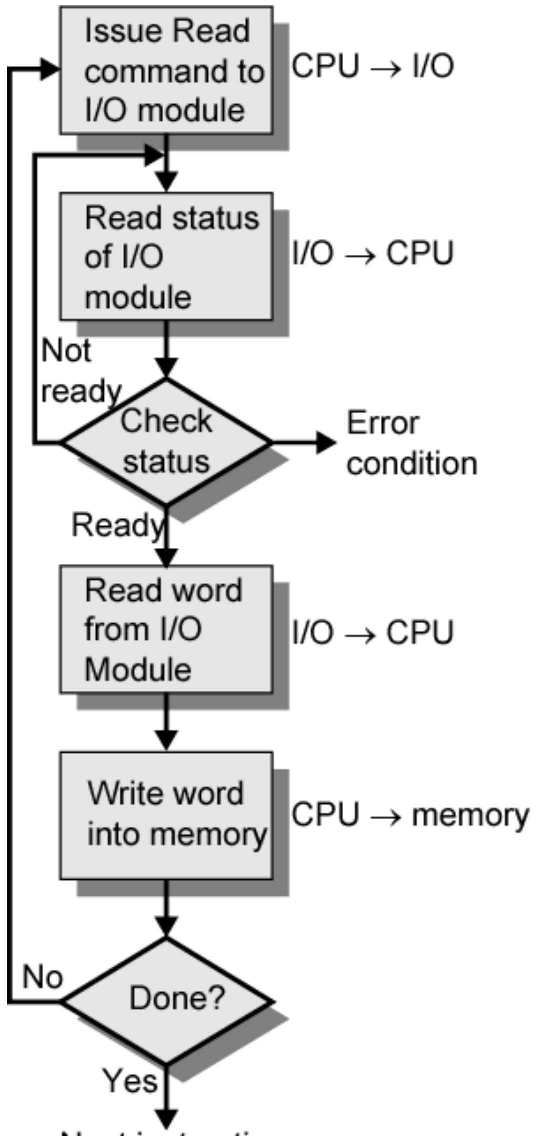
Input Output Techniques

- With *programmed I/O*, data are exchanged between the processor and the I/O module.
- When the processor issues a command to the I/O module, it must wait until the I/O operation is complete.
- If the processor is faster than the I/O module, this is wasteful of processor time.
- With *interrupt-driven I/O*, the processor issues an I/O command, continues to execute other instructions, and is interrupted by the I/O module when the latter has completed its work.
- *In direct memory access* (DMA), the I/O module and main memory exchange data directly, without processor involvement.

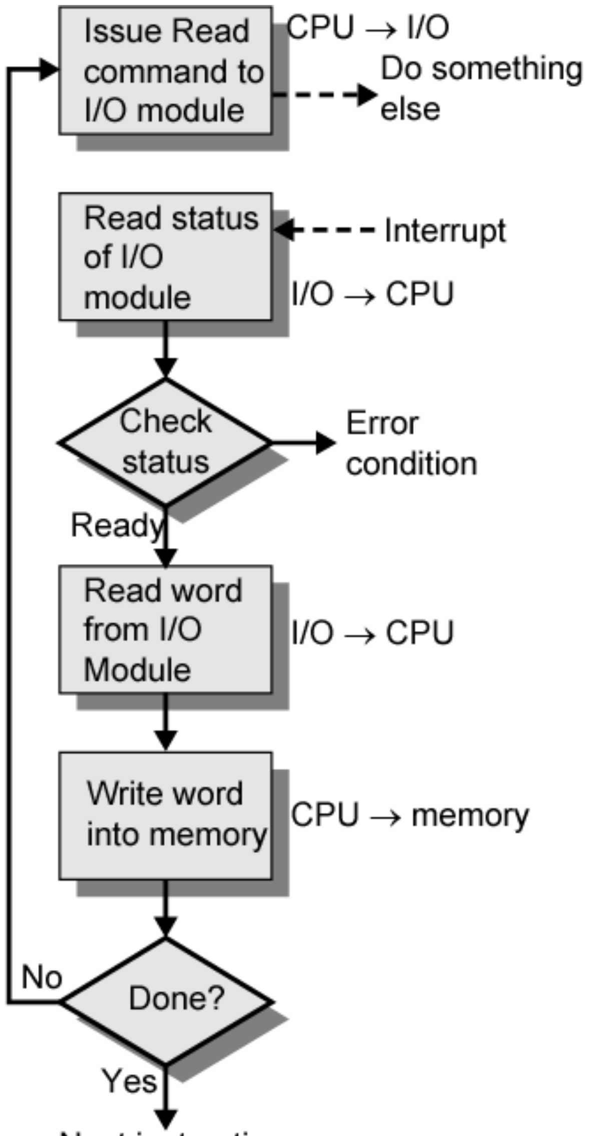


	No Interrupts	**Use of Interrupts**
I/O-to-memory transfer through processor	Programmed I/O	Interrupt-driven I/O
Direct I/O-to-memory transfer		Direct memory access (DMA)

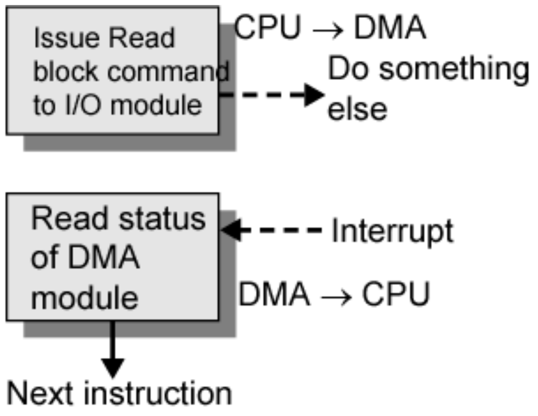
Three Techniques for Input of a Block of Data



(a) Programmed I/O



(b) Interrupt-driven I/O


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

Programmed I/O - detail

- 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

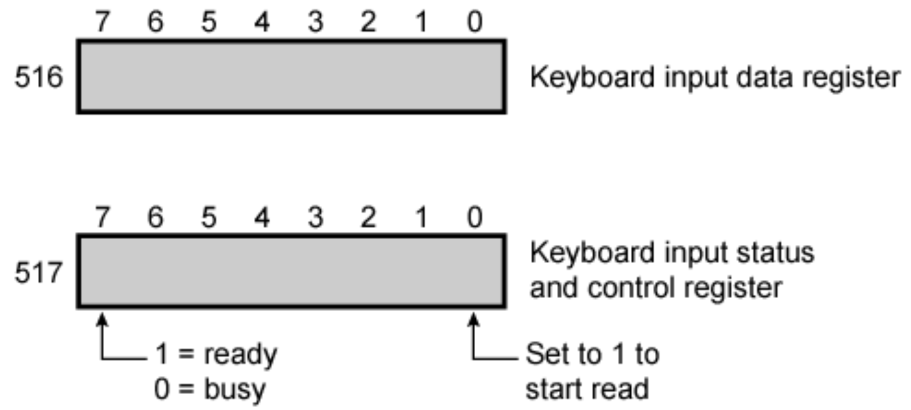
Programmed I/O : I/O Commands

- To execute an I/O-related instruction, the processor issues an address, specifying the particular I/O module and external device, and an I/O command. There are four types of I/O commands that an I/O module may receive when it is addressed by a processor:
  - **Control:** Used to activate a peripheral and tell it what to do.
  - **Test:** Used to test various status conditions associated with an I/O module and its peripherals.
  - **Read:** Causes the I/O module to obtain an item of data from the peripheral and place it in an internal buffer. The processor can then obtain the data item by requesting that the I/O module place it on the data bus.
  - **Write:** Causes the I/O module to take an item of data (byte or word) from the data bus and subsequently transmit that data item to the peripheral.

I/O Mapping

- Memory mapped I/O
  - Devices and memory share an address space
  - I/O looks just like memory read/write
  - No special commands for I/O
    - Large selection of memory access commands available
- Isolated I/O
  - Separate address spaces
  - Need I/O or memory select lines
  - Special commands for I/O
    - Limited set

Memory Mapped and Isolated I/O



ADDRESS	INSTRUCTION	OPERAND	COMMENT
200	Load AC	"1"	Load accumulator
	Store AC	517	Initiate keyboard read
202	Load AC	517	Get status byte
	Branch if Sign = 0	202	Loop until ready
	Load AC	516	Load data byte

(a) Memory-mapped I/O

ADDRESS	INSTRUCTION	OPERAND	COMMENT
200	Load I/O	5	Initiate keyboard read
201	Test I/O	5	Check for completion
	Branch Not Ready	201	Loop until complete
	In	5	Load data byte

(b) Isolated I/O

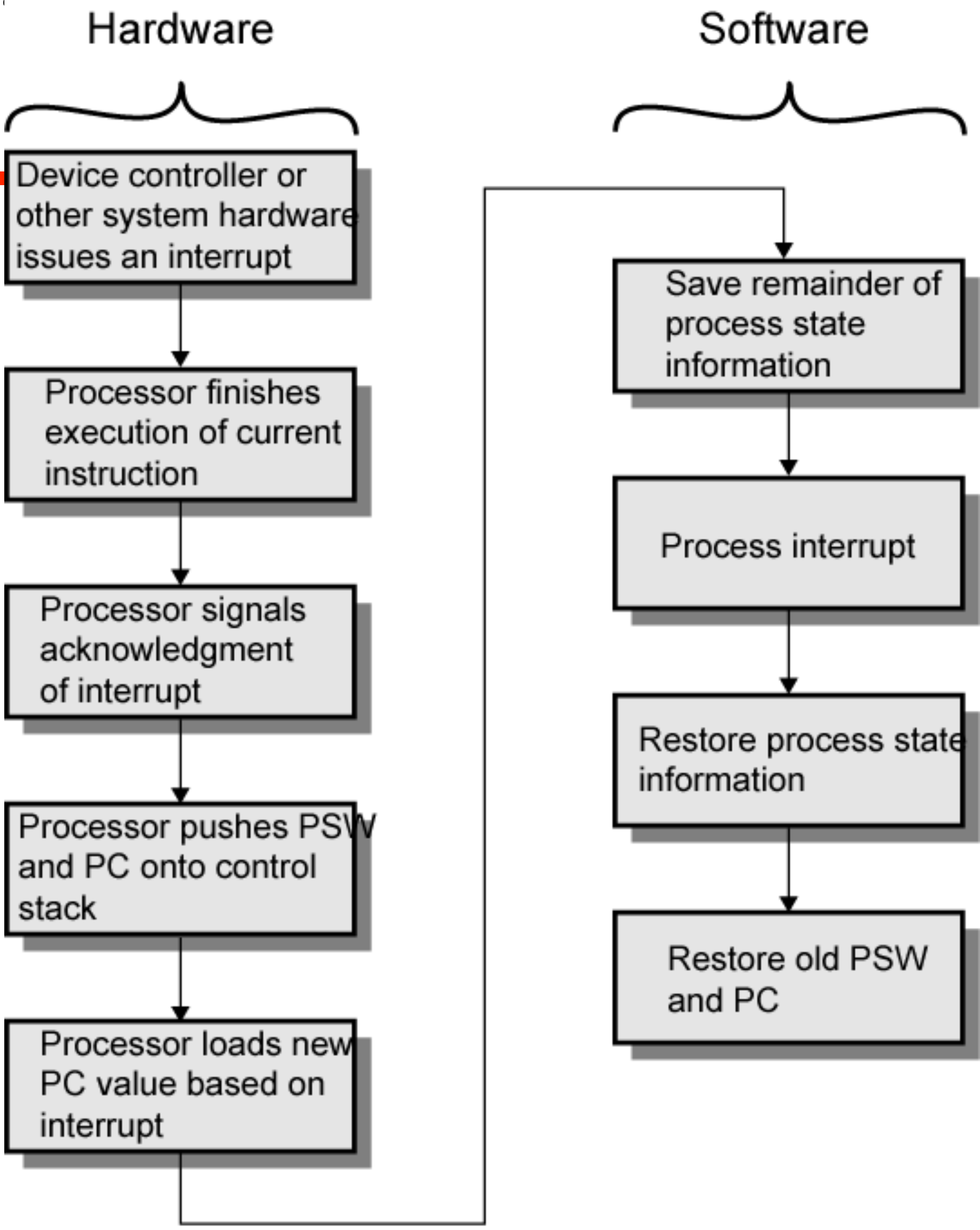
Interrupt Driven I/O

- Overcomes CPU waiting
- No repeated CPU checking of device
- I/O module interrupts when ready

Interrupt Driven I/O: Basic Operation

- CPU issues read command
- I/O module gets data from peripheral whilst CPU does other work
- I/O module interrupts CPU
- CPU requests data
- I/O module transfers data

Simple Interrup
Processing


Hardware
Software
Device controller or other system hardware issues an interrupt
Processor finishes execution of current instruction
Processor signals acknowledgment of interrupt
Processor pushes PSW and PC onto control stack
Processor loads new PC value based on interrupt
Save remainder of process state information
Process interrupt
Restore process state information
Restore old PSW and PC

CPU Viewpoint

- Issue read command
- Do other work
- Check for interrupt at end of each instruction cycle
- If interrupted:-
  - Save context (registers)
  - Process interrupt
    - Fetch data & store

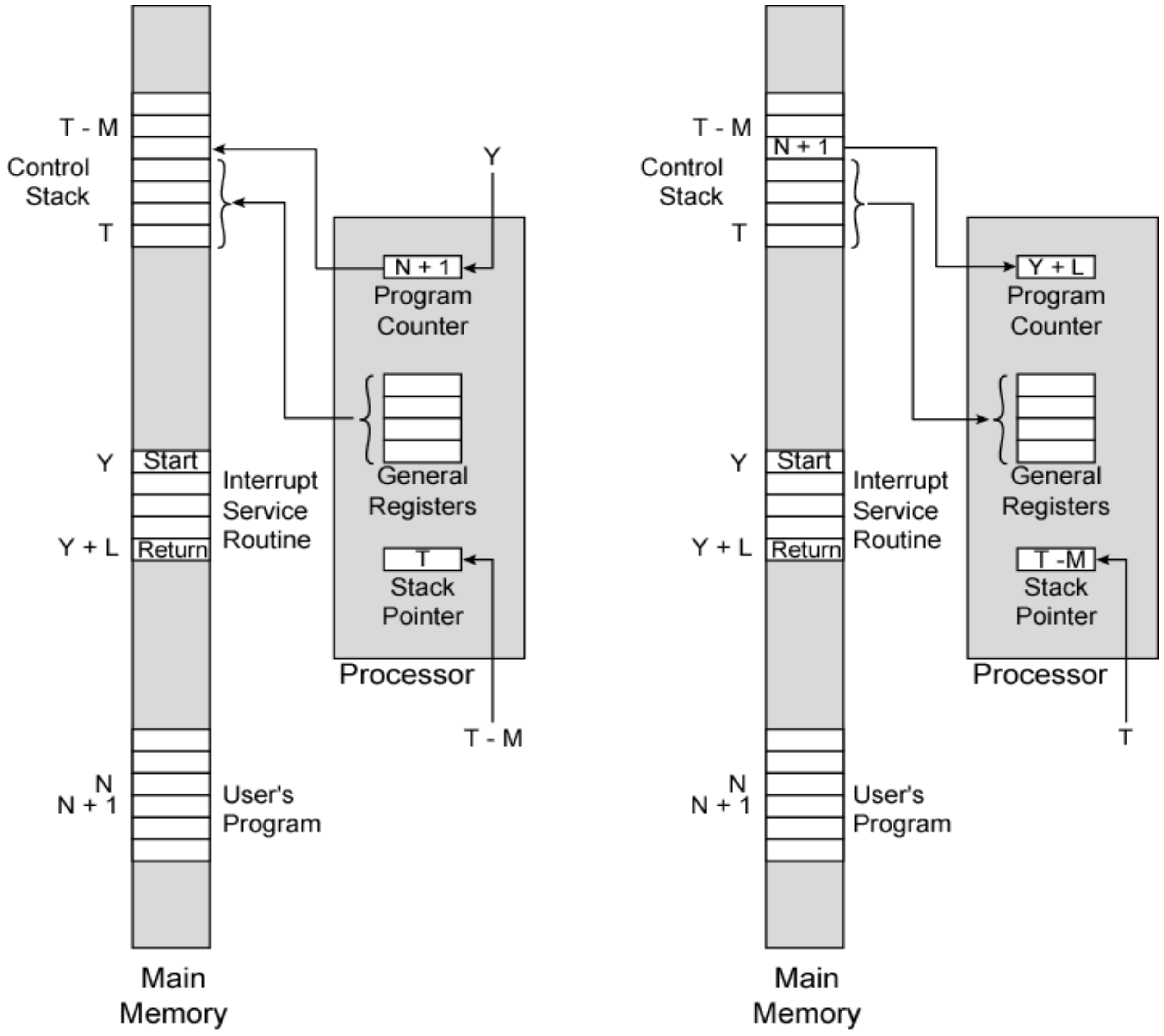
Example:
Changes in Memory and Registers for an Interrupt

A user program is interrupted after the instruction at location N.

The contents of all of the registers plus the address of the next instruction (N+1) are pushed onto the stack.

The stack pointer is updated to point to the new top of stack, and the program counter is updated to point to the beginning of the interrupt service routine.

When interrupt processing is complete, the saved register values are retrieved from the stack and restored to the registers



(a) Interrupt occurs after instruction at location N

(b) Return from interrupt

Design Issues

- How do you identify the module issuing the interrupt?
- How do you deal with multiple interrupts?
  - i.e. an interrupt handler being interrupted

Identifying Interrupting Module (1)

- Different line for each module
  - PC
  - Limits number of devices
- Software poll
  - CPU asks each module in turn
  - Slow

Identifying Interrupting Module (2)

- Daisy Chain or Hardware poll
  - Interrupt Acknowledge sent down a chain
  - Module responsible places vector on bus
  - CPU uses vector to identify handler routine
- Bus Master
  - Module must claim the bus before it can raise interrupt
  - e.g. PCI & SCSI

Multiple Interrupts

- Each interrupt line has a priority
- Higher priority lines can interrupt lower priority lines
- If bus mastering only current master can interrupt

Direct Memory Access

- DMA unit takes the control of the system from the CPU to transfer data to and from memory over the system bus
- Cycle stealing is used to transfer data on the system bus because the DMA unit in effect steals a bus cycle.
- The instruction cycle is suspended so data can be transferred
- The CPU pauses one bus cycle
- No interrupts occur
  - Do not save context

DMA Operation

- 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

Direct Memory Access

- The DMA technique works as follows. 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/write is requested, using the **read/write control line** between the processor and the DMA module.
  - —The **address of the I/O device** is involved, communicated on the data lines.
  - —The **starting location in memory** is communicated on the data lines and stored by the DMA module in **its address register.**
  - —The **number of words** to read/written is communicated on the data lines and stored in the **data count register**.

DMA

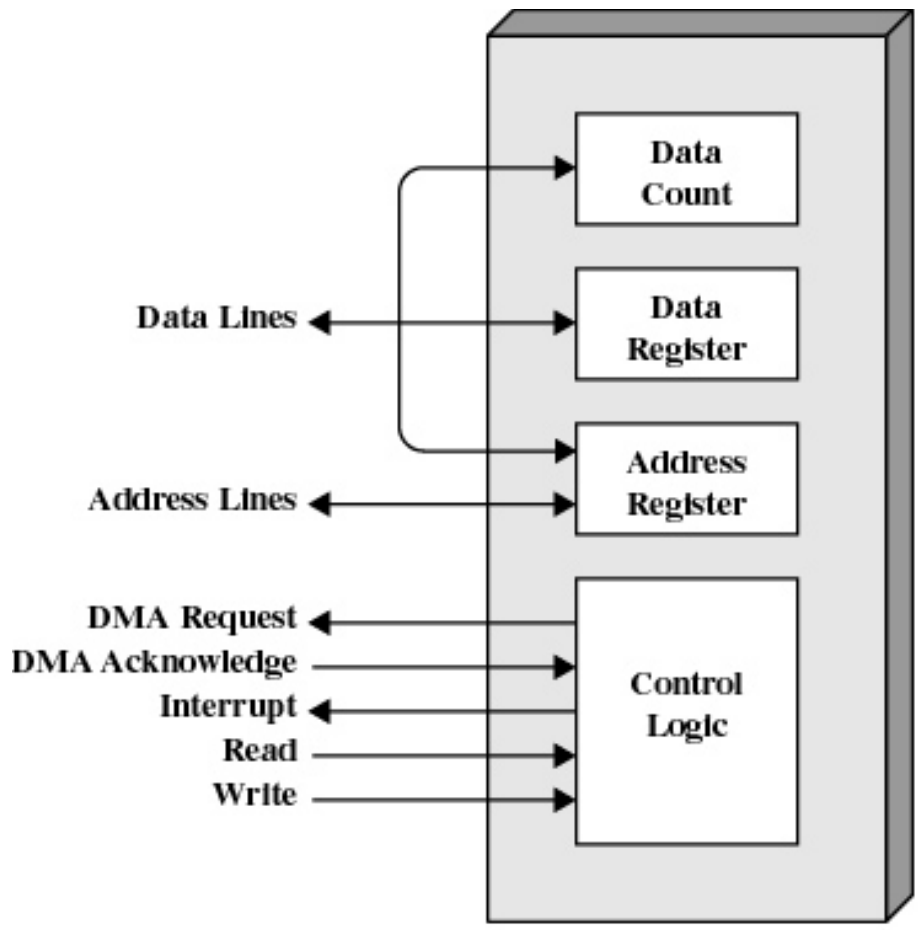


Figure 11.2 Typical DMA Block Diagram

DMA

- Cycle stealing causes the CPU to execute more slowly
- Number of required busy cycles can be cut by integrating the DMA and I/O functions
- Path between DMA module and I/O module that does not include the system bus

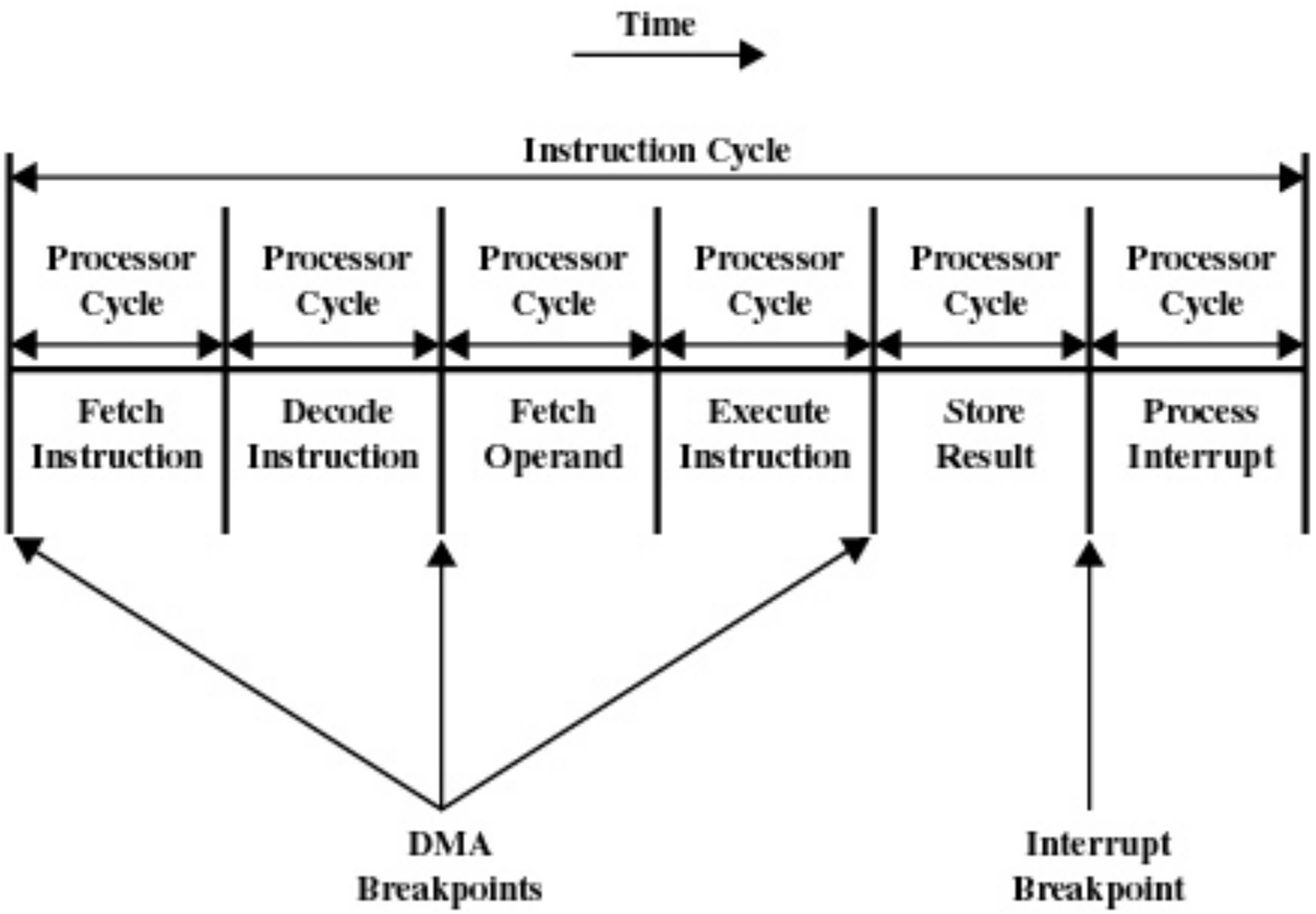
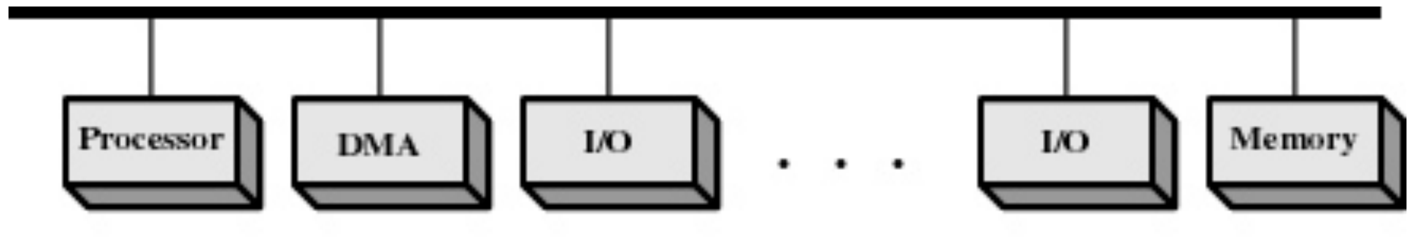


Figure 11.3 DMA and Interrupt Breakpoints During an Instruction Cycle

DMA configurations

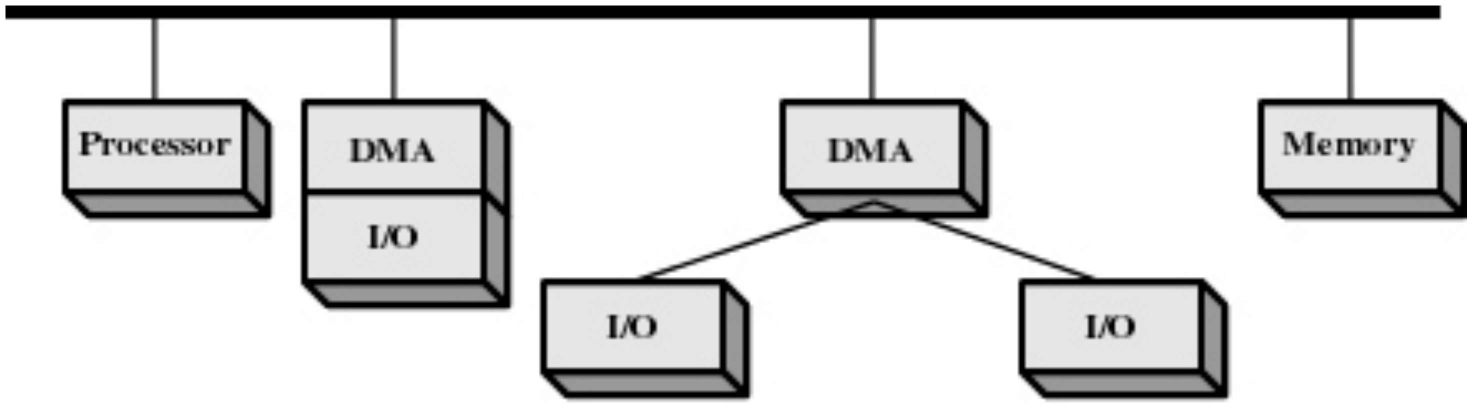


(a) Single-bus, detached DMA

Figure 11.4 Alternative DMA Configurations

- All the modules share the same system bus
- The configuration is inexpensive, but inefficient.
- As with processor-controlled programmed I/O, each transfer of a word consume 2 bus cycles (transfer request + transfer)

DMA



(b) Single-bus, Integrated DMA-I/O

Figure 11.4 Alternative DMA Configurations

DMA

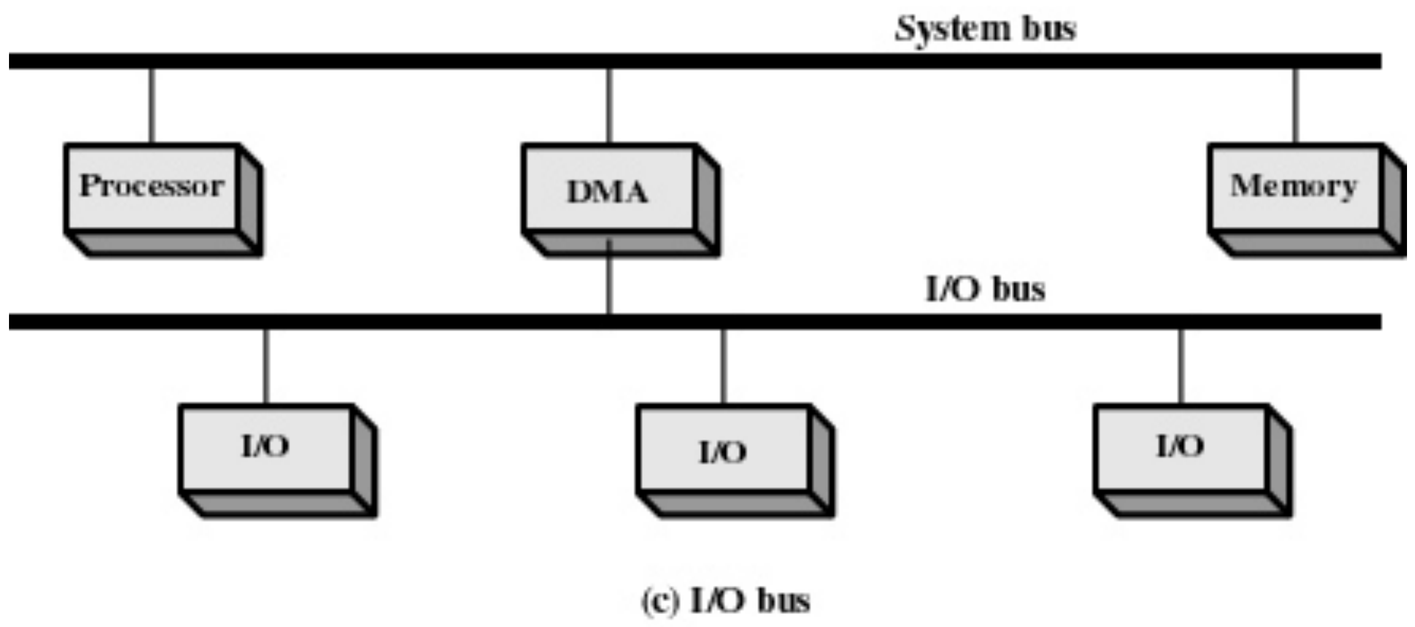


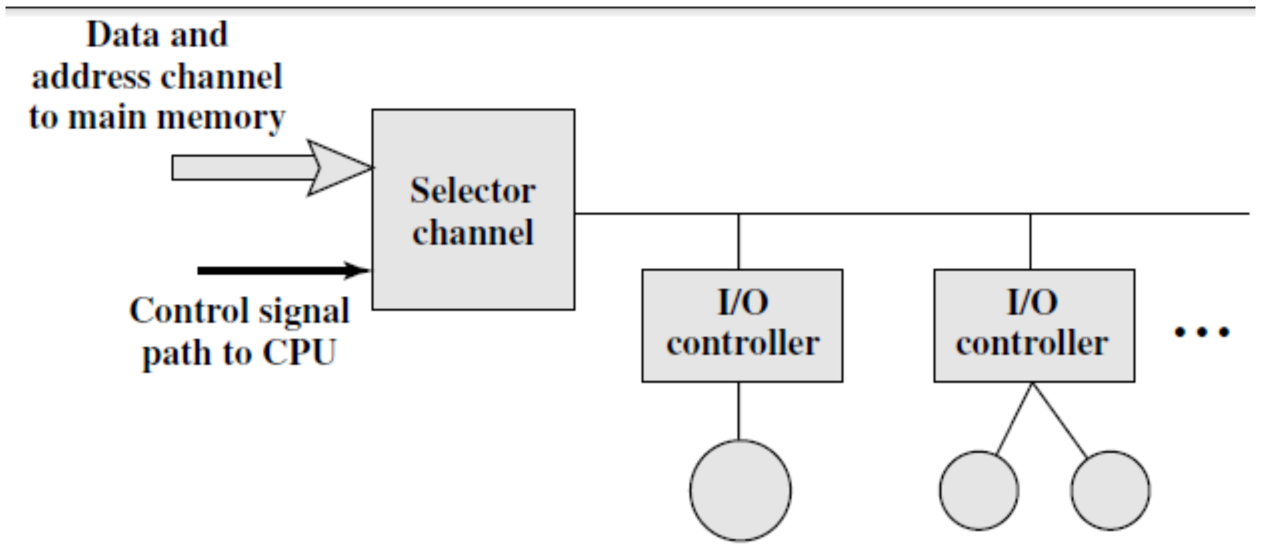
Figure 11.4 Alternative DMA Configurations

I/O Channels & Processors

- I/O devices getting more sophisticated
- e.g. 3D graphics cards
- CPU instructs I/O controller to do transfer
- I/O controller does entire transfer
- Improves speed
  - Takes load off CPU
  - Dedicated processor is faster

I/O Channel Architecture: Selector

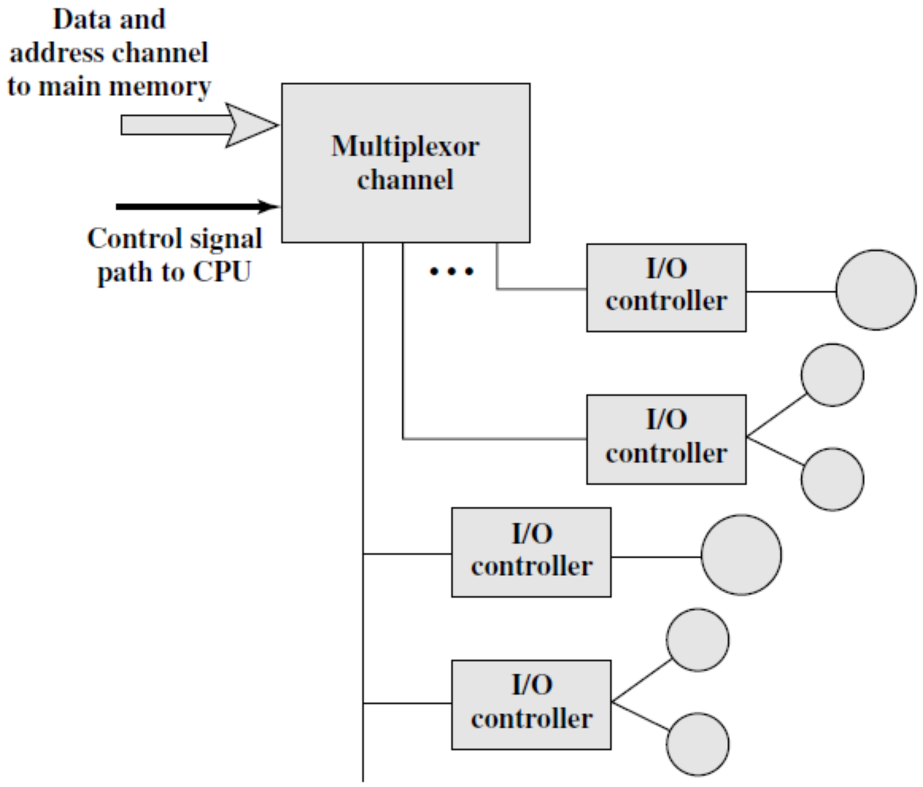
- *Selector channel* controls multiple high-speed devices and, at any one time, is dedicated to the transfer of data with one of those devices.
- Thus, the I/O channel selects one device and effects the data transfer.
- Each device, is handled by a *controller,* or I/O module.


Data and address channel to main memory
Selector channel
Control signal path to CPU
I/O controller
I/O controller
• • •

(a) Selector

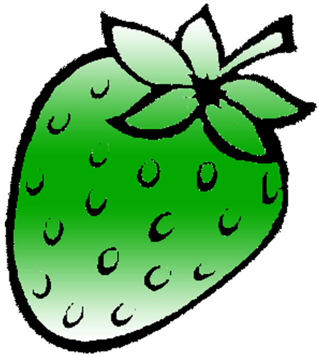
I/O Channel Architecture: Multiplexor

- A *multiplexor channel* can handle I/O with multiple devices at the same time.
- For low-speed devices, a *byte multiplexor* accepts or transmits characters as fast as possible to multiple devices.
- For high-speed devices, a *block multiplexor* interleaves blocks of data from several devices.



(b) Multiplexor

STRAWBERRY



/strawberrydevelopers

/strawberry_app

For more visit:

Strawberrydevelopers.weebly.com