INPUT-OUTPUT ORGANIZATION

Slides prepared by:
Dr. Zubair Ahmad Shah
Department of Computer Science and Engineering
Islamic University of Science and Technology, Awantipora

TOPICS

- Peripheral Devices
- Input-Output Interface
- Asynchronous Data Transfer
- Modes of Transfer
- Priority Interrupt
- Direct Memory Access (DMA)
- Input-Output Processor

PERIPHERAL DEVICES

- Input or output devices attached to a computer.
  - Examples:
    - Monitor
      - Cathode Ray Tube (CRT) based
      - Plasma Display Panel (PDP)
      - Liquid Crystal Display (LCD)
      - Light Emitting Diode (LED)
    - Keyboard
    - Mouse

Peripheral Devices Cont.

- Printer
  - Daisywheel printer
  - Dot Matrix printer
  - Inkjet printer
  - Laser printer
  - 3D printer
- Plotter
- Magnetic Tape
- Magnetic Disk

Peripheral Devices Cont.

- I/O communication usually involves transfer of alphanumeric information.
- The standard binary code for alphanumeric characters is **ASCII** (American Standard Code for Information Interchange).
- ASCII uses 7 bits to code 128 characters
  - 94 printable
  - 34 non-printable (for control)

American Standard Code for Information Interchange (ASCII)

$b_4 b_3 b_2 b_1$	$b_7 b_6 b_5$ = 000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	‘	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	’	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	¦
1101	CR	GS	−	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	—	o	DEL

Control characters

NUL	Null	DLE	Data link escape
SOH	Start of heading	DC1	Device control 1
STX	Start of text	DC2	Device control 2
ETX	End of text	DC3	Device control 3
EOT	End of transmission	DC4	Device control 4
ENQ	Enquiry	NAK	Negative acknowledge
ACK	Acknowledge	SYN	Synchronous idle
BEL	Bell	ETB	End of transmission block
BS	Backspace	CAN	Cancel
HT	Horizontal tab	EM	End of medium
LF	Line feed	SUB	Substitute
VT	Vertical tab	ESC	Escape
FF	Form feed	FS	File separator
CR	Carriage return	GS	Group separator
SO	Shift out	RS	Record separator
SI	Shift in	US	Unit separator
SP	Space	DEL	Delete

Peripheral Devices Cont.

- ASCII Code(in decimal) –
  - **0** – 48
  - **A** – 65
  - **a** – 97

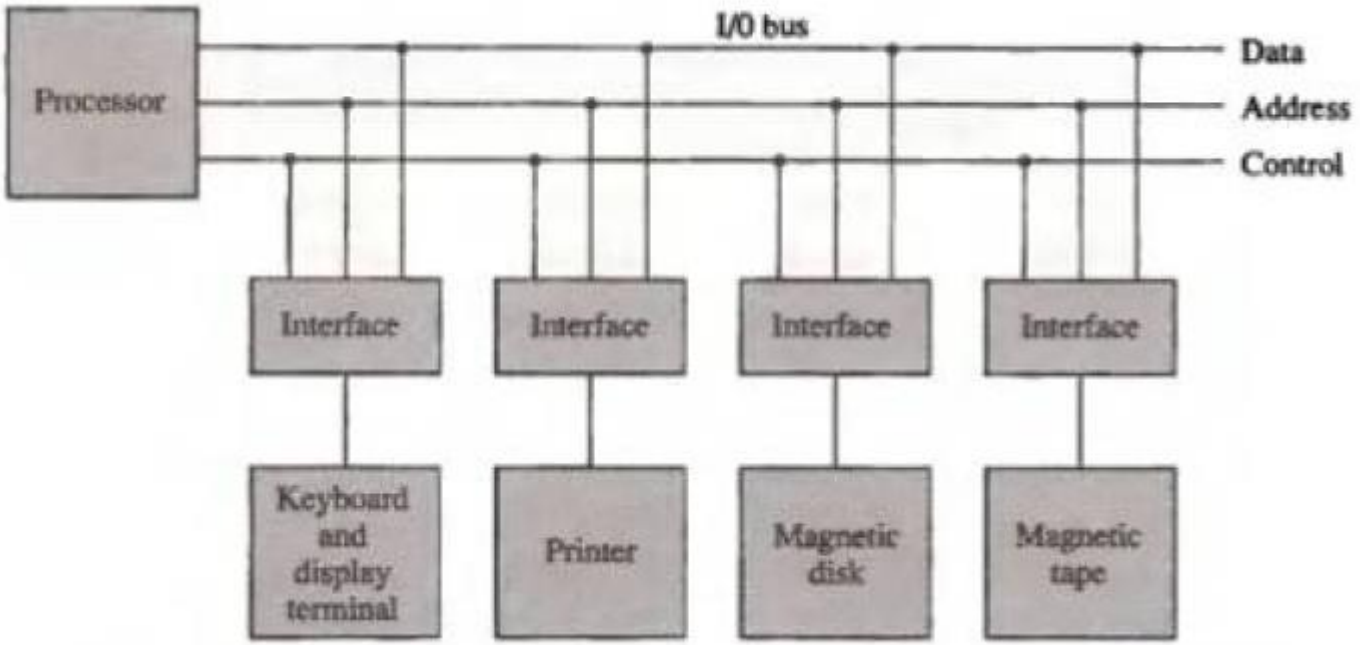
- Most computers use 8 bits (1 byte) for a character.

INPUT-OUTPUT INTERFACE

- **I/O interface helps in synchronizing transfer of data between CPU and peripheral devices.**
- **Need of I/O Interface:**

1. Peripherals are electromechanical and electromagnetic devices and their manner of operation is different from the operation of the CPU and memory, which are electronic devices. Therefore, a conversion of signal values may be required.
2. The data transfer rate of peripherals is usually slower than the transfer rate of the CPU, and consequently, a synchronization mechanism may be needed.
3. Data codes and formats in peripherals differ from the word format in the CPU and memory.
4. The operating modes of peripherals are different from each other and each must be controlled so as not to disturb the operation of other peripherals connected to the CPU.

Connection of I/O bus to input-output devices.



Input-Output Interface Cont.

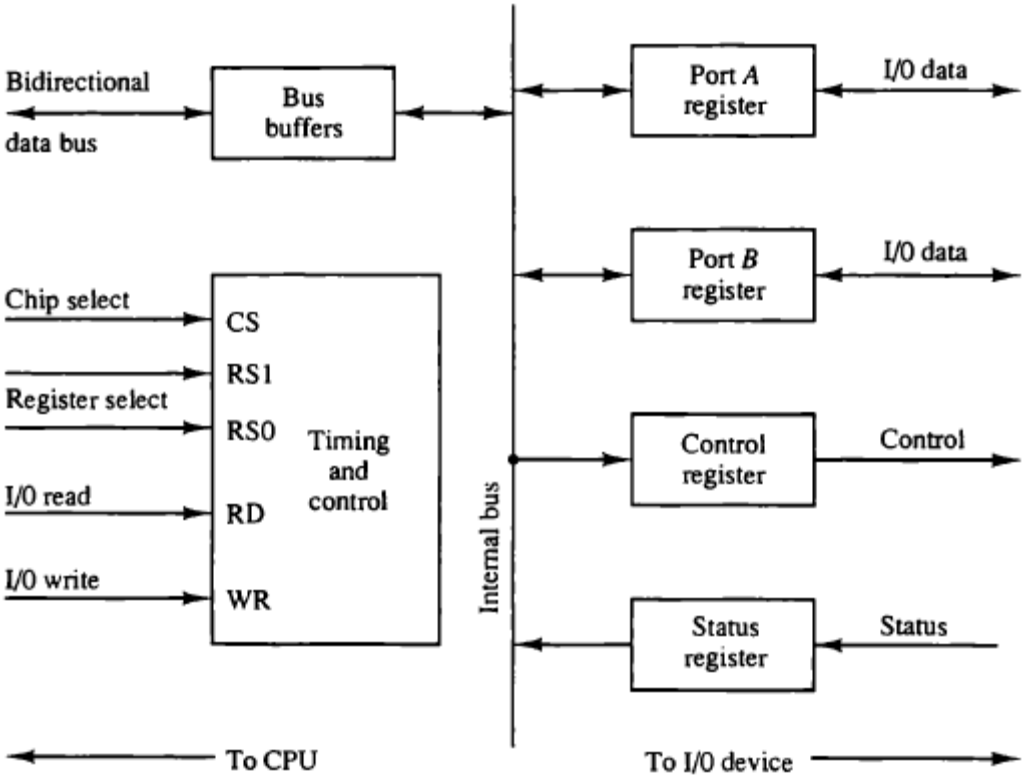
- Types of commands an interface can receive:
  - Control Command
  - Status Command
  - Data Output Command
  - Data Input Command

Input-Output Interface Cont.

- **Computer buses can be used to communicate with memory and I/O in three ways:**

1. Use two separate buses, one for memory and the other for I/O.
2. Use one common bus for both memory and I/O but have separate control lines for each.
3. Use one common bus for memory and I/O with common control lines.

- **Method 1 is used when there is a separate I/O Processor (in addition to Central Processing Unit)**
- **Method 2 is used in Isolated I/O Configuration**
- **Method 3 is used in Memory-mapped I/O Configuration**



CS	RS1	RS0	Register selected
0	×	×	None: data bus in high-impedance
1	0	0	Port *A* register
1	0	1	Port *B* register
1	1	0	Control register
1	1	1	Status register

Example of I/O interface unit.

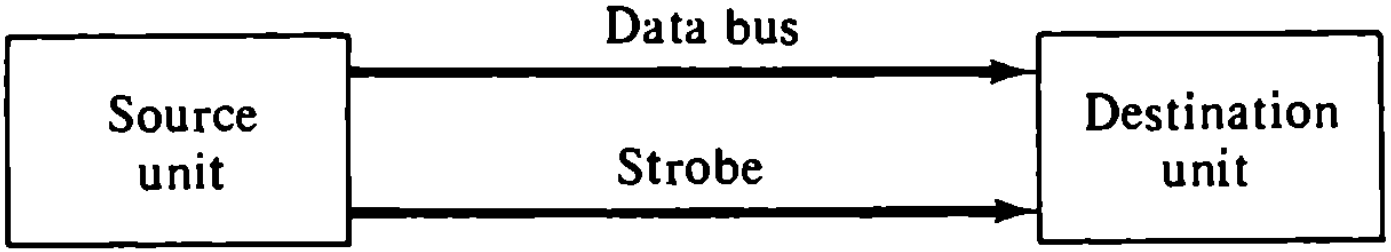
ASYNCHRONOUS DATA TRANSFER

- **Asynchronous** mode of **data transfer** is used when the source unit and destination unit have different clocks.
- Methods of Asynchronous Data Transfer:
  I. Strobe Control
  II. Handshaking

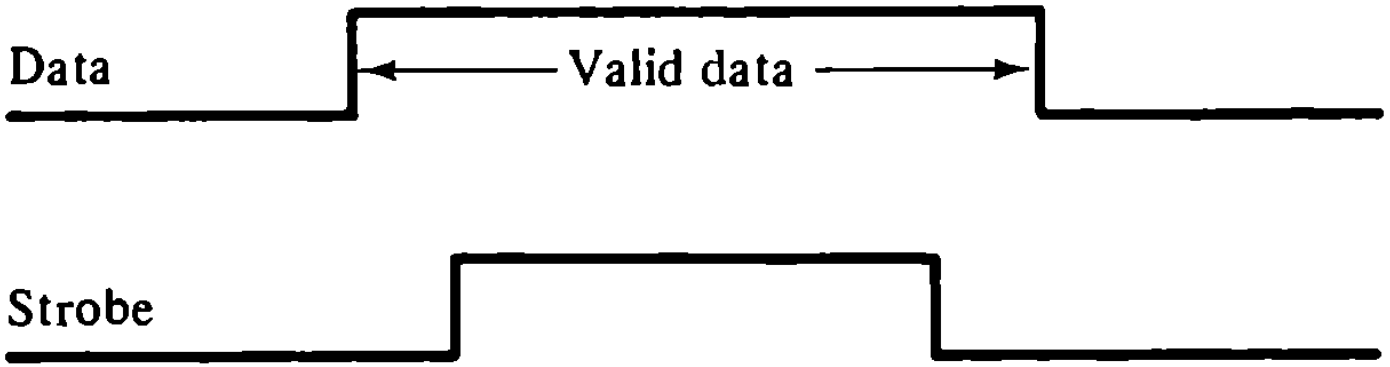
Methods of Asynchronous data transfer:

I. Strobe Control Method

- Employs a single control line to time each transfer.
- The strobe may be generated by either the source or the destination unit
  - Source-initiated strobe control
  - Destination-initiated strobe control

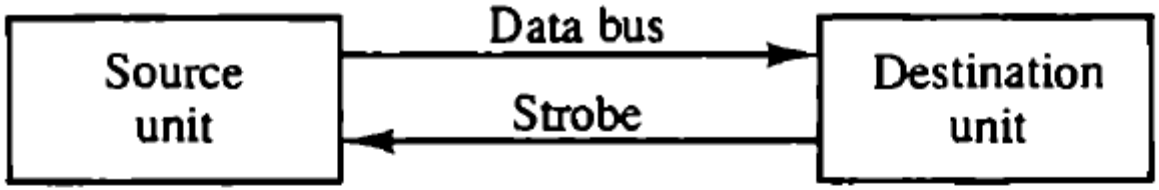


(a) Block diagram

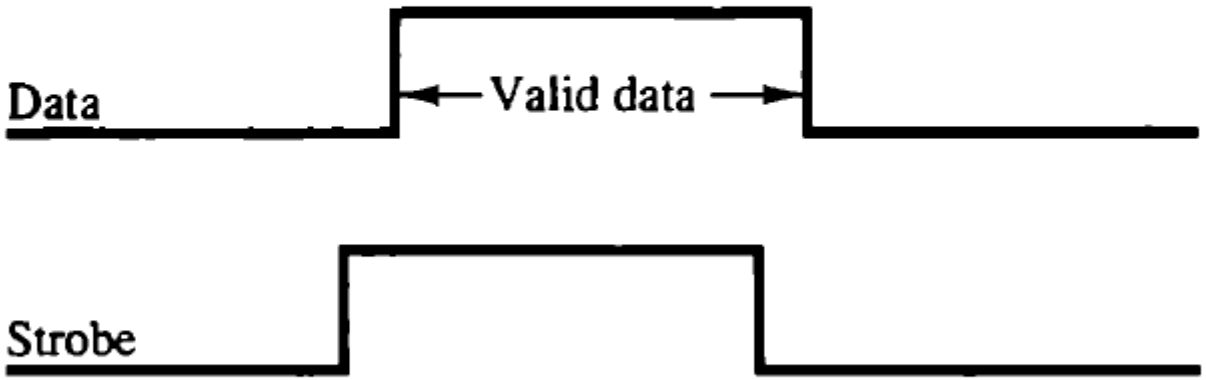


(b) Timing diagram

Source-initiated strobe for data transfer.



(a) Block diagram



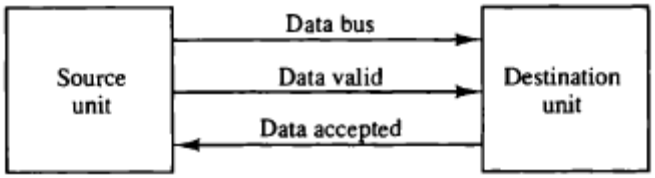
(b) Timing diagram

Destination-initiated strobe for data transfer.

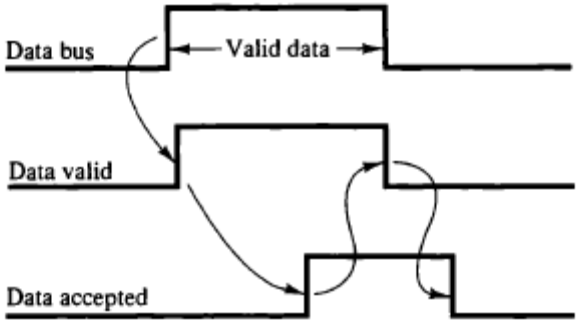
- Disadvantage of Strobe Method.

II. Handshaking Method

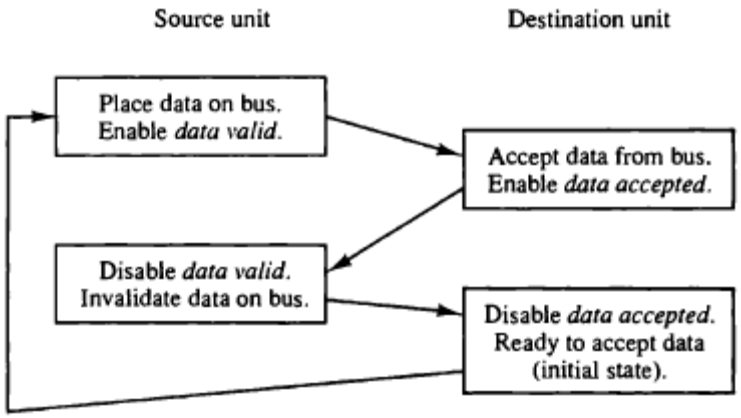
- Solves the problem of strobe method by using another control signal.
- More reliable method.
- Timeout Mechanism.
- The transfer may be initiated by either the source or the destination.



(a) Block diagram



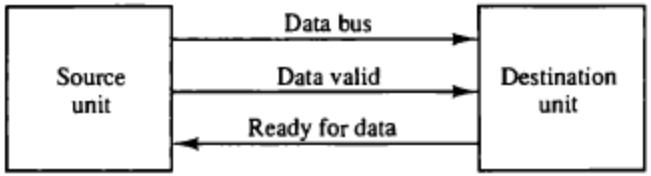
(b) Timing diagram



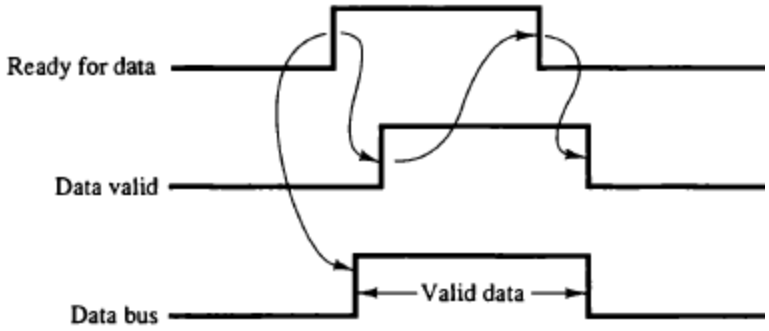
(c) Sequence of events

Source-initiated transfer using handshaking.

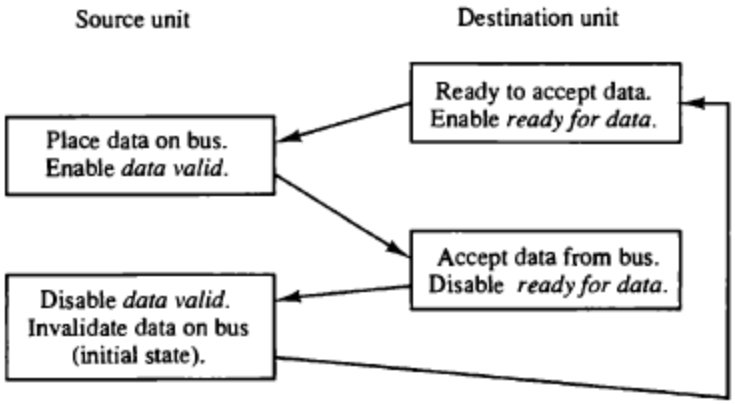
Destination-initiated transfer using handshaking.



(a) Block diagram



(b) Timing diagram



(c) Sequence of events

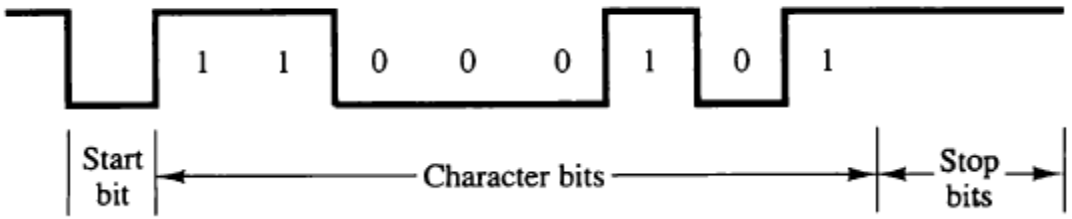
Asynchronous Serial Transfer

- In serial data transfer, each bit in a message is sent in sequence one at a time.
- Serial transmission:
  - *Synchronous*: Two units share a common clock frequency and bits are transmitted continuously at the rate dictated by the clock pulses.
  - *Asynchronous*: Binary information is sent only when it is available and the line remains idle when there is no information to be transmitted.

Asynchronous Data Transfer Cont.

- ***Asynchronous Serial Transfer***
  - **Special bits (start and stop bits) are inserted at both ends of the character code.**

Asynchronous serial transmission.



A transmitted character can be detected by the receiver from knowledge of the transmission rules:

1. When a character is not being sent, the line is kept in the 1-state.
2. The initiation of a character transmission is detected from the start bit, which is always 0.
3. The character bits always follow the start bit.
4. After the last bit of the character is transmitted, a stop bit is detected when the line returns to the 1-state for at least one bit time.

Asynchronous Data Transfer Cont.

- **Baud Rate:** It is defined as the rate at which serial information is transmitted.

QUESTION:

10 characters per second with an 11-bit format in an asynchronous serial transmission has a transfer rate of how many baud? What is the time for transmission of 1 bit?

SOLUTION:

10 characters per second

$\Rightarrow$ 110 bits per second (as 1 character transfer takes 11 bits - 8 of character, 1 of start and 2 of stop)

$\Rightarrow$ Transfer rate = 110 baud

Asynchronous Data Transfer Cont.

Again, 10 characters per 1 second

$\Rightarrow$ 1 character = 1/10 = 0.1 sec

$\Rightarrow$ 11 bits = 0.1 sec

$\Rightarrow$ 1 bit = 0.00909 sec

$\Rightarrow$ Bit time = 9.09 msec

Asynchronous Data Transfer Cont.

QUESTION:

How many characters per second can be transmitted over a 1200-baud line in each of the following modes? (Assume a character code of eight bits.)

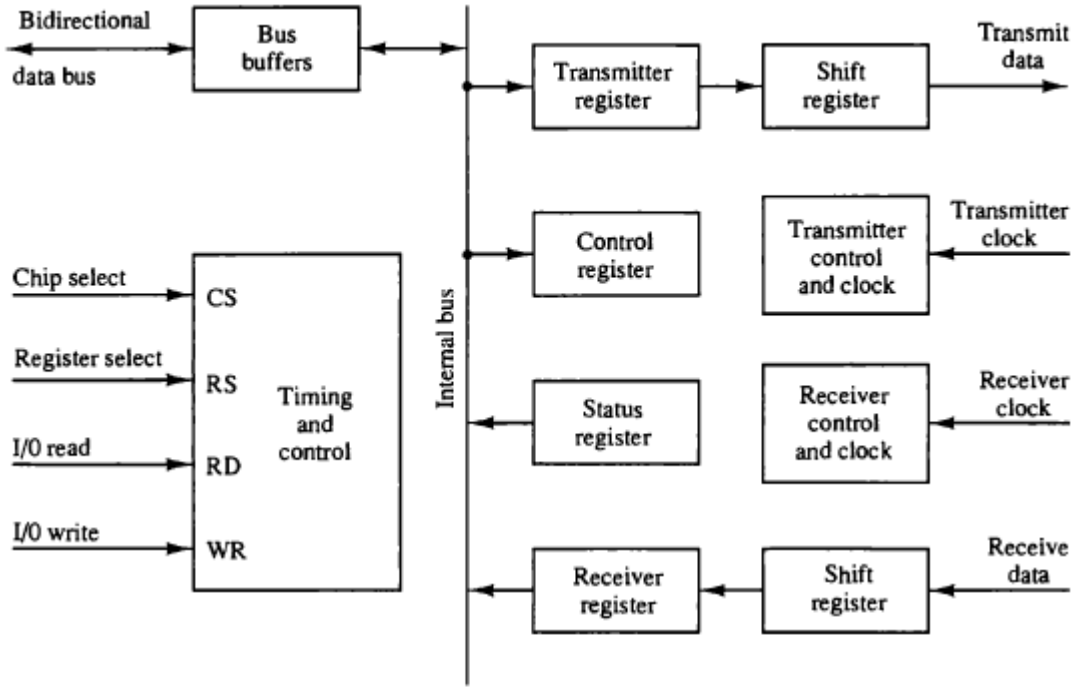
a. Synchronous serial transmission.
b. Asynchronous serial transmission with two stop bits.
c. Asynchronous serial transmission with one stop bit.

SOLUTION:

a. **1200/8 = 150 characters per second**
b. **1200/11 = 109 characters per second**
c. **1200/10 = 120 characters per second**

Asynchronous Data Transfer Cont.

- **Asynchronous Communication Interface:**
  - also called as Universal Asynchronous Receiver-Transmitter (UART).



CS	RS	Operation	Register selected
0	×	×	None: data bus in high-impedance
1	0	WR	Transmitter register
1	1	WR	Control register
1	0	RD	Receiver register
1	1	RD	Status register

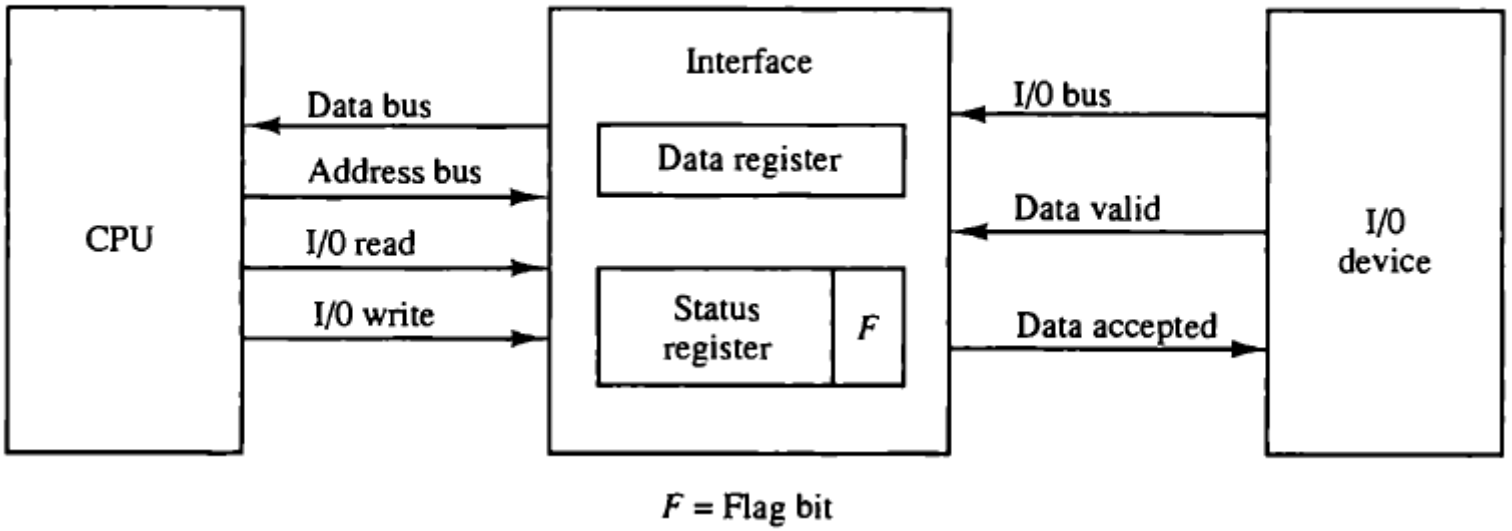
Block diagram of a typical asynchronous communication interface.

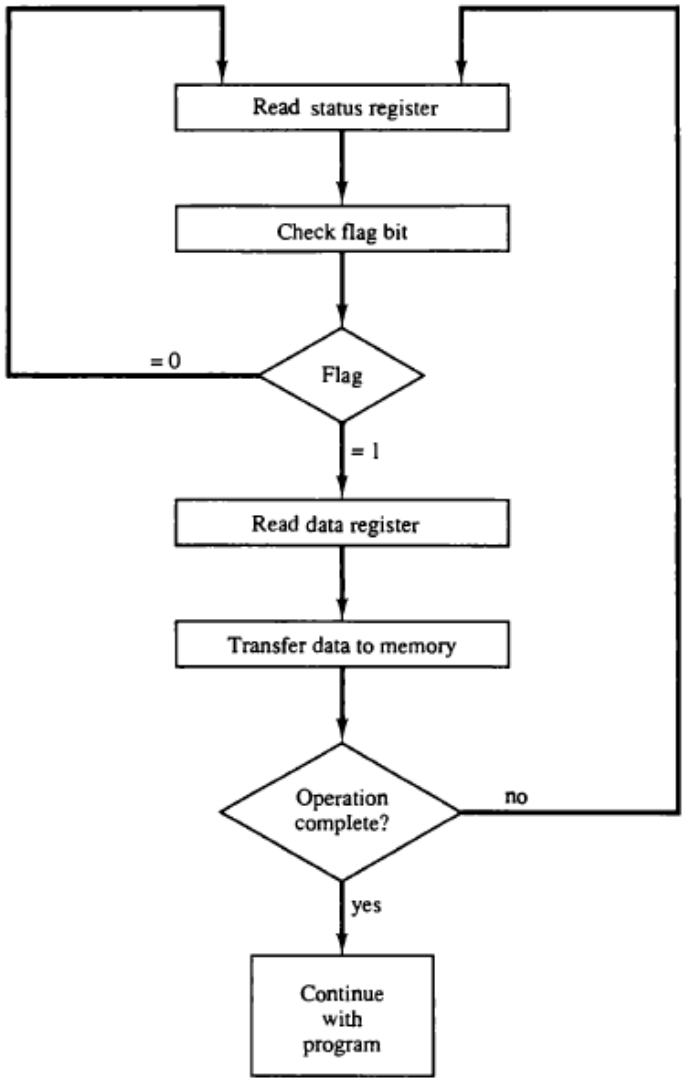
MODES OF DATA TRANSFER BETWEEN THE CENTRAL COMPUTER AND I/O DEVICES

- Ultimate source or destination of data is memory.
- Data transfer between the central computer and I/O devices may be handled in three ways (modes):
  1. Programmed I/O
  2. Interrupt Initiated I/O
  3. Direct Memory Access (DMA)

- **Programmed Input-Output:**
  - Programmed I/O operations are the result of I/O instructions written in the computer program.
  - The transfer is usually to and from a CPU register and peripheral.
  - Other instructions are needed to transfer the data to and from CPU and memory.
  - The CPU has to continuously execute a program to check the status of interface registers.
  - Keeps processor needlessly busy.
  - Useful in computers that are dedicated to monitor a device continuously.

Data transfer from I/O device to CPU.





Flowchart for CPU program to input data.

- **Interrupt Initiated Input-Output:**
  - Alternative to CPU continuously monitoring the flag.
  - The interface unit informs the CPU when it is ready to transfer data.
  - When the flag is set, an interrupt is generated to inform the CPU about it.
  - The CPU
    - stops the execution of the program that it was executing at that time
    - Stores the return address in memory stack
    - Branches to a service routine that processes the required I/O transfer.

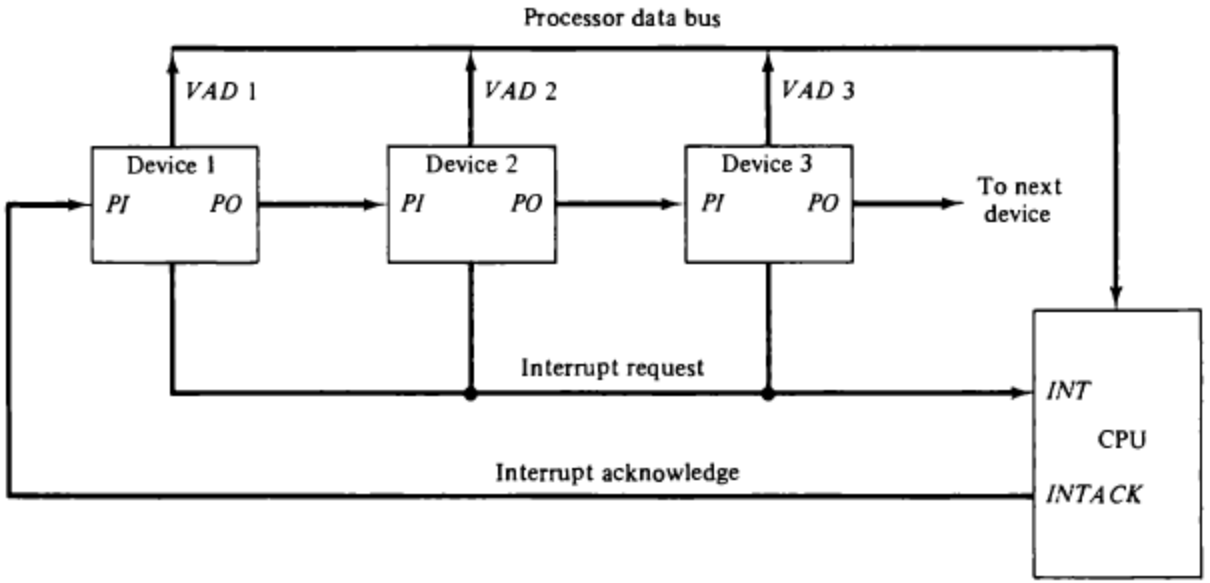
- Types of Interrupt (based on the branch address):
  - NonVectored Interrupt
    - Branch address is a fixed location in memory.
  - Vectored Interrupt
    - Branch address is provided by the source that interrupts. This branch address is called *interrupt vector*.
    - Interrupt vector may be a *direct address* or an *indirect address*.

- **Priority Interrupt:**
  - A system that establishes a priority over the various interrupt sources.
    - Software-based system -- Polling
    - Hardware-based system – Daisy Chaining
  - When two devices interrupt the computer at the same time, the computer services the device with the higher priority first.

Modes of Transfer cont.

- **Polling:**
  - Software-based method
  - One common branch address for all interrupts
  - The routine polls the interrupt sources in sequence (highest priority first, 2nd highest priority second, 3rd highest priority third and so on)
  - Then branches to one of many possible service routines.
  - Disadvantage: Slow as compared to hardware method

- **Daisy Chaining Priority:**
  - Hardware-based method
  - Interrupt request generation is similar to negative-logic OR operation.



Daisy-chain priority interrupt.

***PI*: Priority In**
***PO*: Priority Out**
***VAD*: Interrupt Vector Address**

QUESTION:

Information is inserted into a FIFO buffer at a rate of m bytes per second. The information is deleted at a rate of n byte per second. The maximum capacity of the buffer is k bytes.

a. How long does it take for an empty buffer to fill up when $m > n$?
b. How long does it take for a full buffer to empty when $m < n$?
c. Is the FIFO buffer needed if $m = n$?

SOLUTION:

a. **Size of FIFO buffer = k bytes**

Bytes inserted into the buffer in 1 second = m

Bytes deleted from the buffer in 1 second = n

m>n

=> Net bytes added to the buffer in 1 second = m-n

=> To fill an empty buffer, it will take = $\frac{k}{m-n}\text{sec}onds$

Modes of Transfer cont.

b. **m<n**

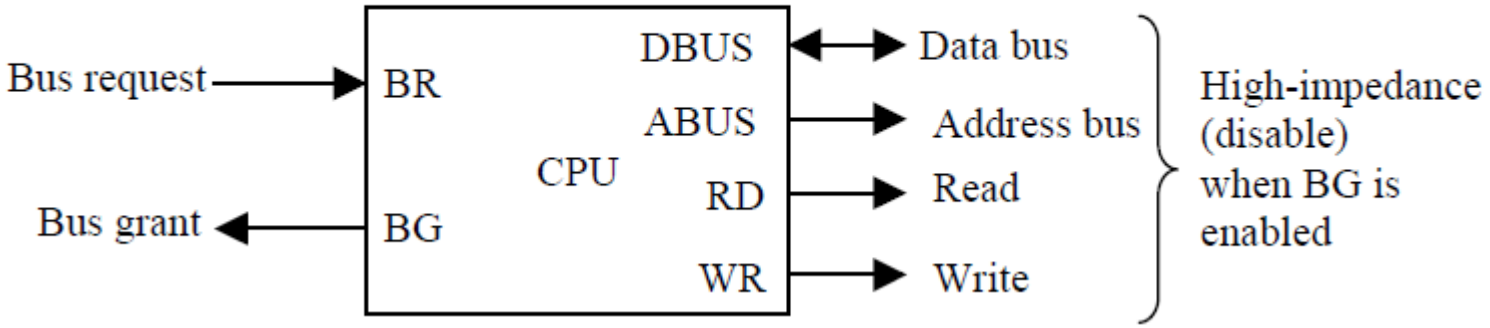
=> Net bytes removed from the buffer in 1 second = n-m

=> To empty a full buffer, it will take = $\frac{k}{n-m}\text{sec}onds$

c. **No.**

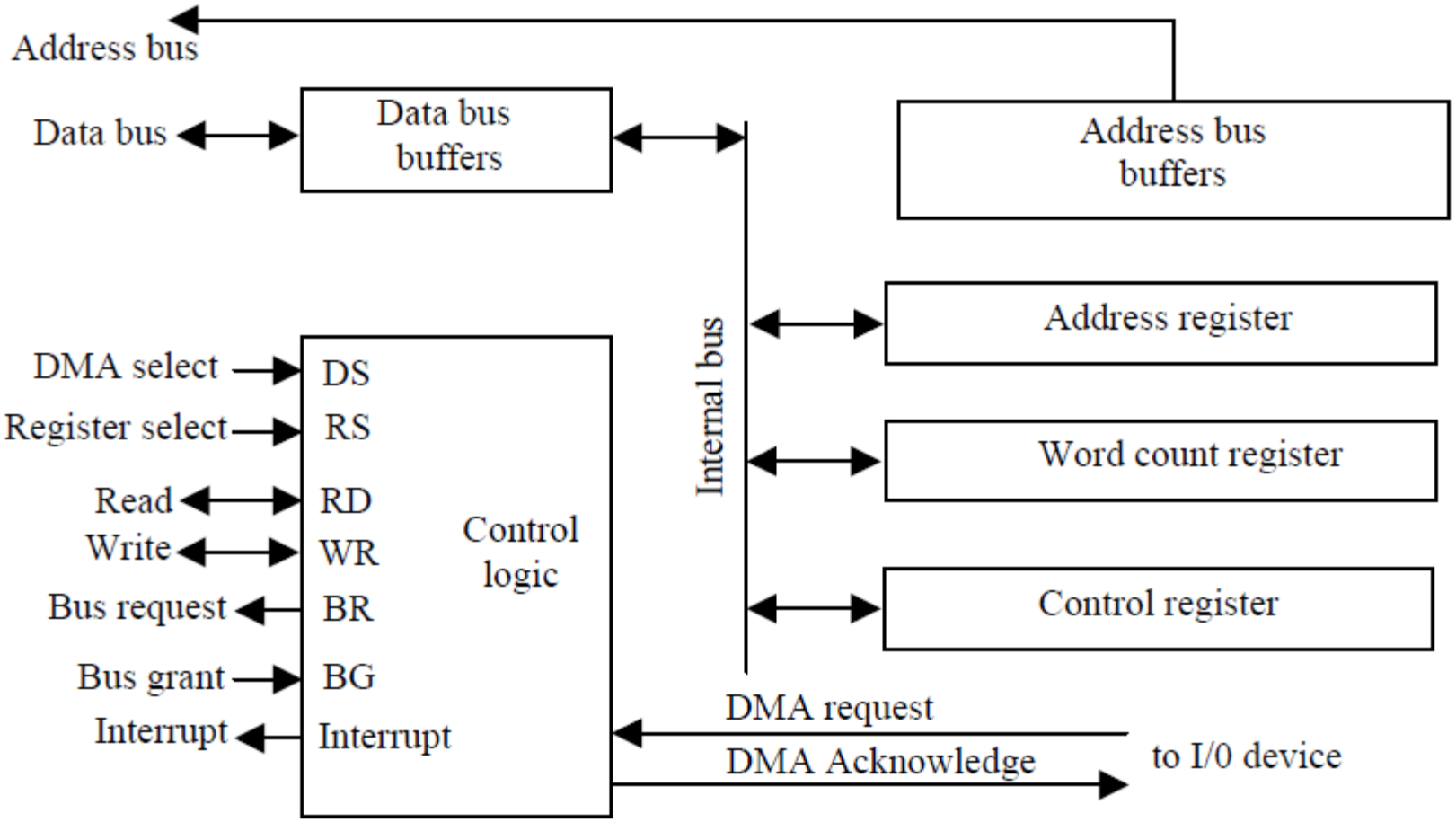
- **Direct Memory Access (DMA):**
  - Transfer of data between the memory and an input-output device not via CPU.
  - DMA is a feature of computer systems that allows certain hardware subsystems to access main system memory (random-access memory), independent of the CPU.
  - The CPU
    - initiates the transfer
    - does other operations while the transfer is in progress
    - and finally receives an interrupt from the DMA Controller (DMAC) when the operation is done.

CPU bus signals for DMA transfer.



- The CPU initializes the DMA transfer by sending the following information to the DMA Controller through the data bus:
  1. The starting address of the memory block where data are available (for read) or where data are to be stored (for write)
  2. The word count - how many words to read or write
  3. Control to specify the mode of transfer such as read or write
  4. Control to start the DMA transfer

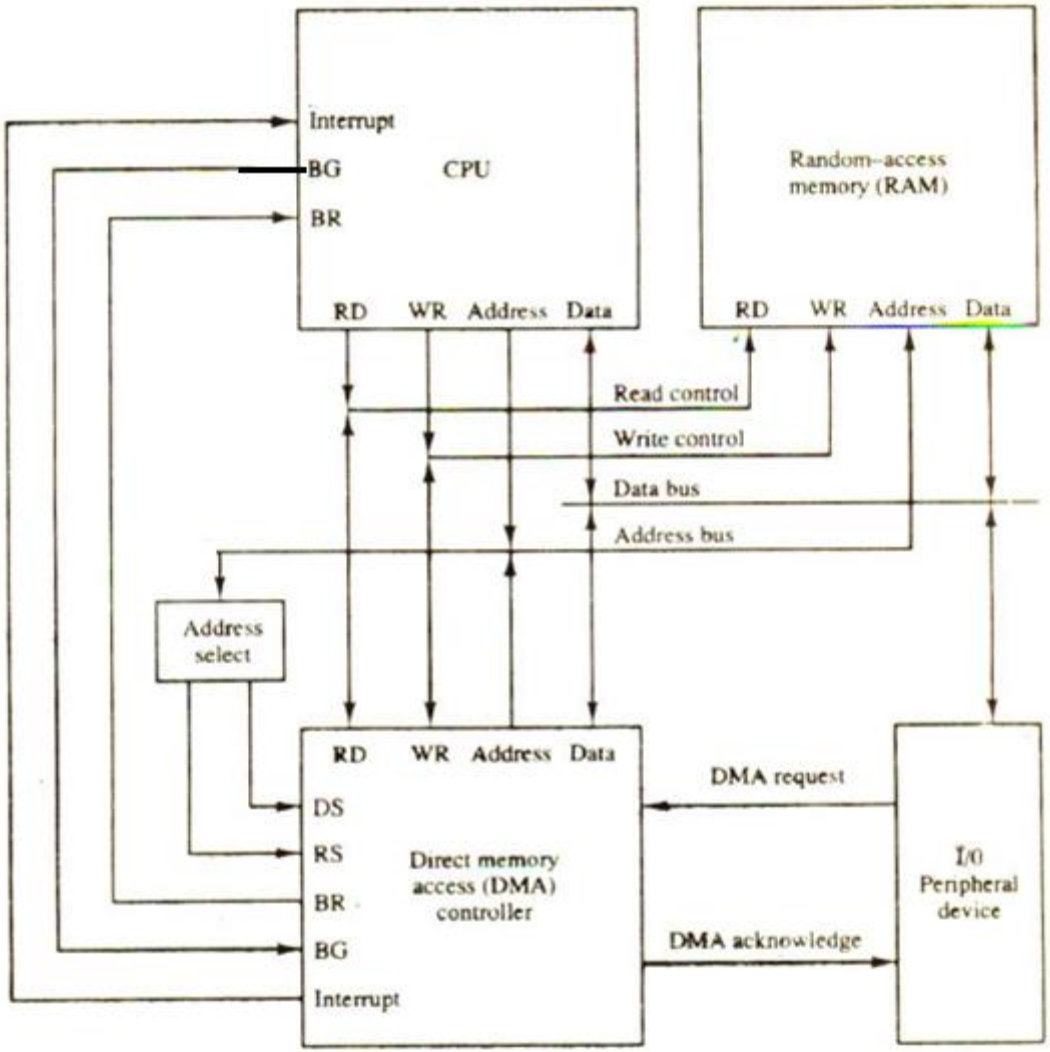
Block diagram of DMA controller.



Modes of Transfer cont.

Steps in DMA transfer:

1. The peripheral device sends a *DMA request*
2. The DMAC activates the *BR* line, informing the CPU to relinquish the buses
3. The CPU responds with its *BG* line, informing the DMAC that its buses are disabled.
4. The DMAC then puts the current value of its *address register* into the *address bus*, initiates the *RD* or *WR* signal and sends a *DMA acknowledge* to the peripheral device.
5. One word gets transferred from peripheral device to memory or from memory to peripheral device.
6. For each word that is transferred, the DMAC increments its *address register* and decrements its *word count register*.
7. After the transfer is complete, the DMAC activates the *Interrupt signal*, informing the CPU about completion of the transfer.



DMA transfer in a computer system.

Modes of Transfer cont.

Modes of Operation in DMA:

I. Cycle Stealing Mode

- Only one word of data is transferred per request
- The CPU processes an instruction, then the DMA controller transfers one data word, then again the CPU processes an instruction, then the DMA controller transfers one data word, and so on
- Data block transfer is slow

II. Burst Mode or Block Transfer Mode

- An entire block of data is transferred in one contiguous sequence
- Renders the CPU inactive for relatively long periods of time
- Usually used for reading or writing into magnetic disk
- The CPU remains idle for long

Modes of Transfer cont.

III. Transparent mode or Hidden DMA Data Transfer Mode

- Most efficient mode in terms of overall system performance
- Takes the most time to transfer a block of data
- The DMA controller transfers data only when the CPU is performing operations that do not use the system buses

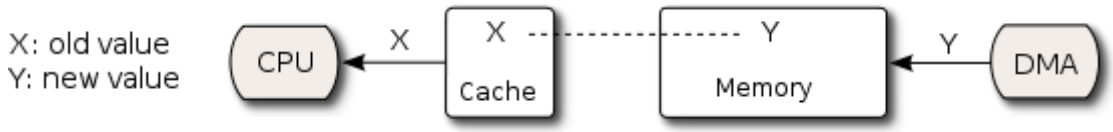
- DMA can lead to **cache coherency** problems:

Cache Coherency Problem (I):

1. CPU accesses location X in the memory
2. The current value gets stored in the cache
3. Subsequent operations on X update the cached copy of X, but not the memory version of X (assuming a write-back cache)
4. If the cache is not flushed to the memory before the next time the peripheral device tries to access X, the peripheral device will receive a stale value of X

Cache Coherency Problem (II):

1. CPU accesses location X in the memory
2. The current value gets stored in the cache
3. DMA transfer updates value of X in memory
4. If the cached copy of X is not invalidated, the CPU will operate on a stale value of X



Solution to Cache Coherence Problem:

- Hardware-based
- Software-based

Modes of Transfer cont.

QUESTION:

It is necessary to transfer 256 words from a magnetic disk to a memory section starting from address 1230. The transfer is by means of DMA. Give the initial values that the CPU must transfer to the DMA controller.

SOLUTION:

CPU must transfer

- 256 to the DMAC Word Count Register
- 1230 to the DMAC Address Register
- Bits to the Control Register to specify a Write Operation.

INPUT-OUTPUT PROCESSOR

STUDY BY YOURSELF

- For reference see your textbook:
  - Mano, M. Morris. *Computer system architecture*. Prentice-Hall of India, 2003.