Computer Organization and Architecture

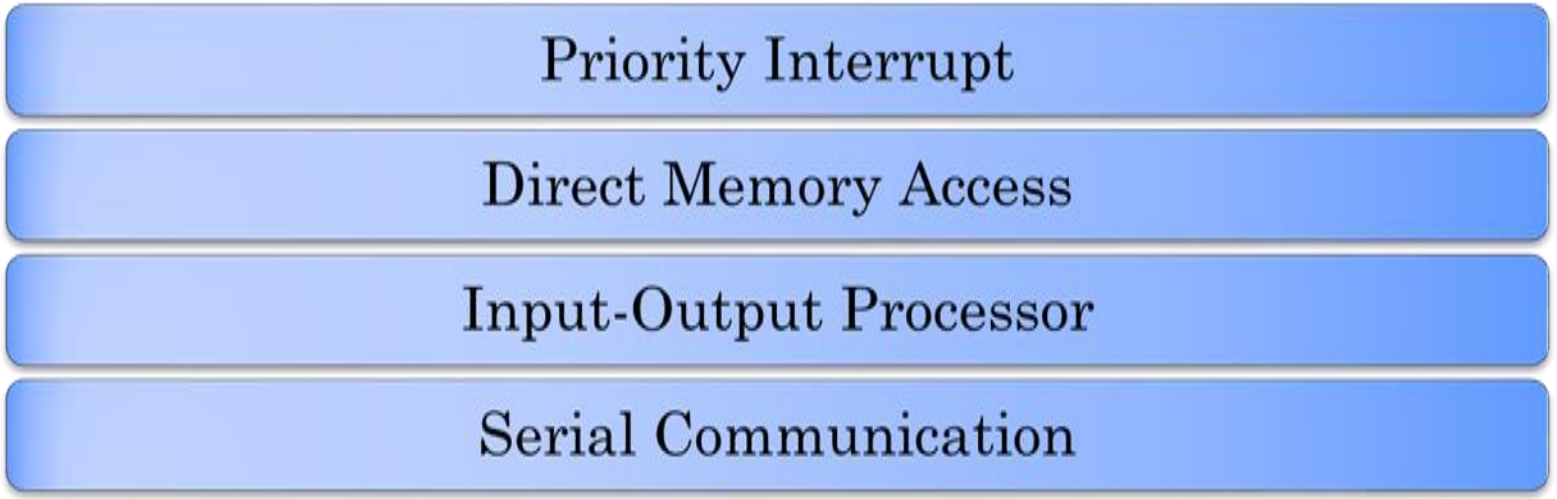
Under Graduate Course
(B. Tech-Information Technology, 2nd Semester)
Jan 2020-July 2020

By

Dr. Satish Kumar Singh

Associate Professor
Indian Institute of Information Technology, Allahabad
Email:sk.singh@iiita.ac.in

Contents


Priority Interrupt
Direct Memory Access
Input-Output Processor
Serial Communication

PRIORITY INTERRUPT

Priority

- Determines which interrupt is to be served first when two or more requests are made simultaneously
- Also determines which device's interrupts are permitted to interrupt the computer while another is being serviced
- Higher priority interrupts can make requests while servicing a lower priority interrupt

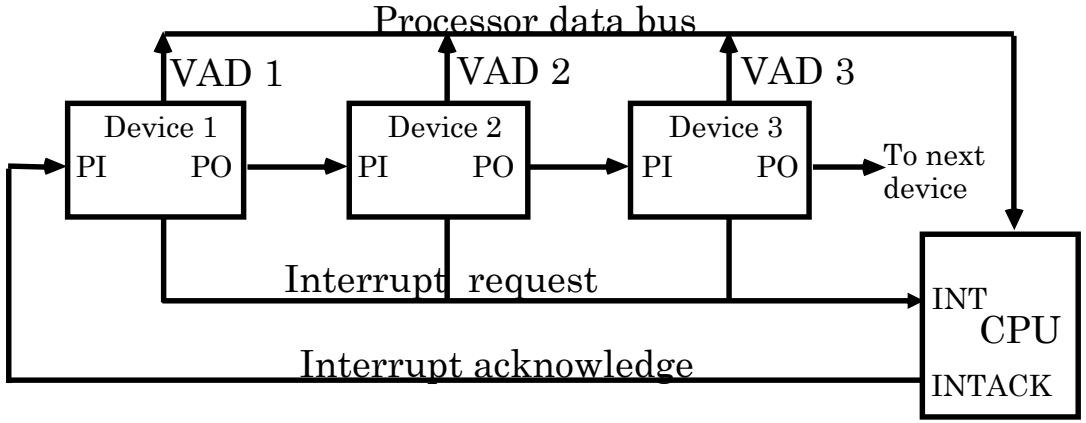
Priority Interrupt by Software(Polling)

- Priority is established by the order of polling the devices(interrupt sources)
- Flexible since it is established by software
- Low cost since it needs a very little hardware
- Very slow

Priority Interrupt by Hardware

- Require a priority interrupt manager which accepts all the interrupt requests to determine the highest priority request
- Fast since identification of the highest priority interrupt request is identified by the hardware
- Fast since each interrupt source has its own interrupt vector to access directly to its own service routine

HARDWARE PRIORITY INTERRUPT - DAISY-CHAIN -



* Serial hardware priority function
* Interrupt Request Line
  - Single common line
* Interrupt Acknowledge Line
  - Daisy-Chain

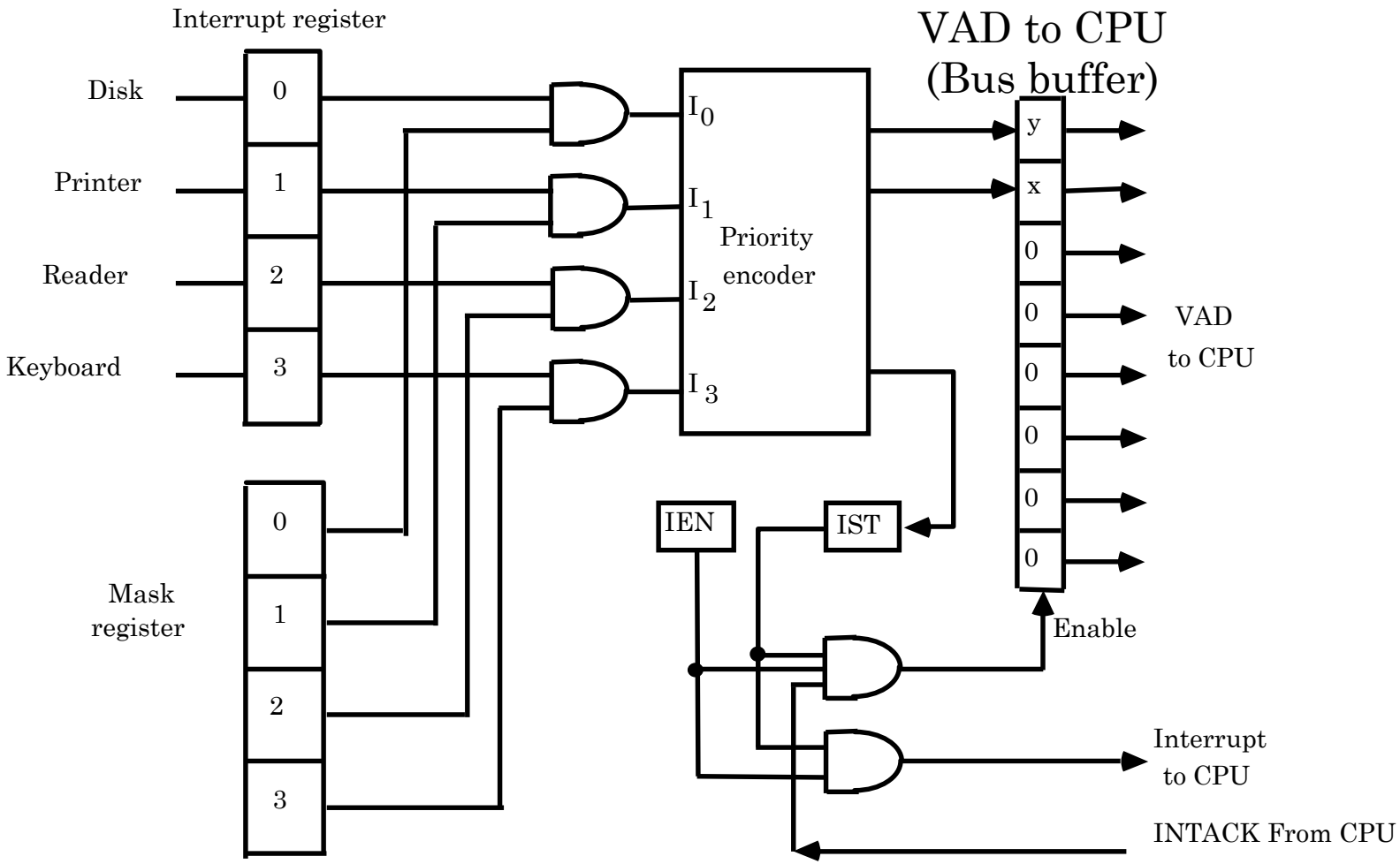
Interrupt Request from any device (If no device has interrupt then int. req. line is in High Level state[=>1], if any device has its interrupt signal, the int. req. line goes to the low level state[=>0].)

-> CPU responds by INTACK <- 1

-> Any device receives signal(INTACK) 1 at PI puts the VAD on the bus

Among interrupt requesting devices the only device which is physically closest to CPU gets INTACK=1, and it blocks INTACK to propagate to the next device

PARALLEL PRIORITY INTERRUPT



IEN: (Interrupt Enable FF) Set or Clear by program instructions ION or IOF

IST: (Interrupt status FF) Represents an unmasked interrupt has occurred. INTACK enables tristate Bus Buffer to load VAD generated by the Priority Logic

Parallel Priority Interrupt

- Interrupt Register:
  - Each bit is associated with an Interrupt Request from different Interrupt Source - different priority level
  - Each bit can be cleared by a program instruction
- Mask Register:
  - Mask Register is associated with Interrupt Register (Control the status of each interrupt request.)
  - Each bit can be set or cleared by an Instruction
  - can be programmed to disable to lower-priority interrupt while a higher-priority device is being serviced. (vice-verso opposite.)

INTERRUPT PRIORITY ENCODER

The priority encoder is a circuit that implements the priority function. If two or more input arrives at the same time, the input having the highest priority will take precedence.

Priority Encoder Truth table

Inputs				Outputs			Boolean functions
I_0	I_1	I_2	I_3	x	y	IST	
1	d	d	d	0	0	1	
0	1	d	d	0	1	1	
0	0	1	d	1	0	1	$x = I_0' \ I_1'$
0	0	0	1	1	1	1	$y = I_0' \ I_1 + I_0' \ I_2'$
0	0	0	0	d	d	0	$(IST) = I_0 + I_1 + I_2 + I_3$

D= don't care conditions

I0 has the highest priority

IST is set one only when one or more input are equal to one.

The output of the priority encoder is used to form part of vector address for each interrupt source.

INTERRUPT CYCLE

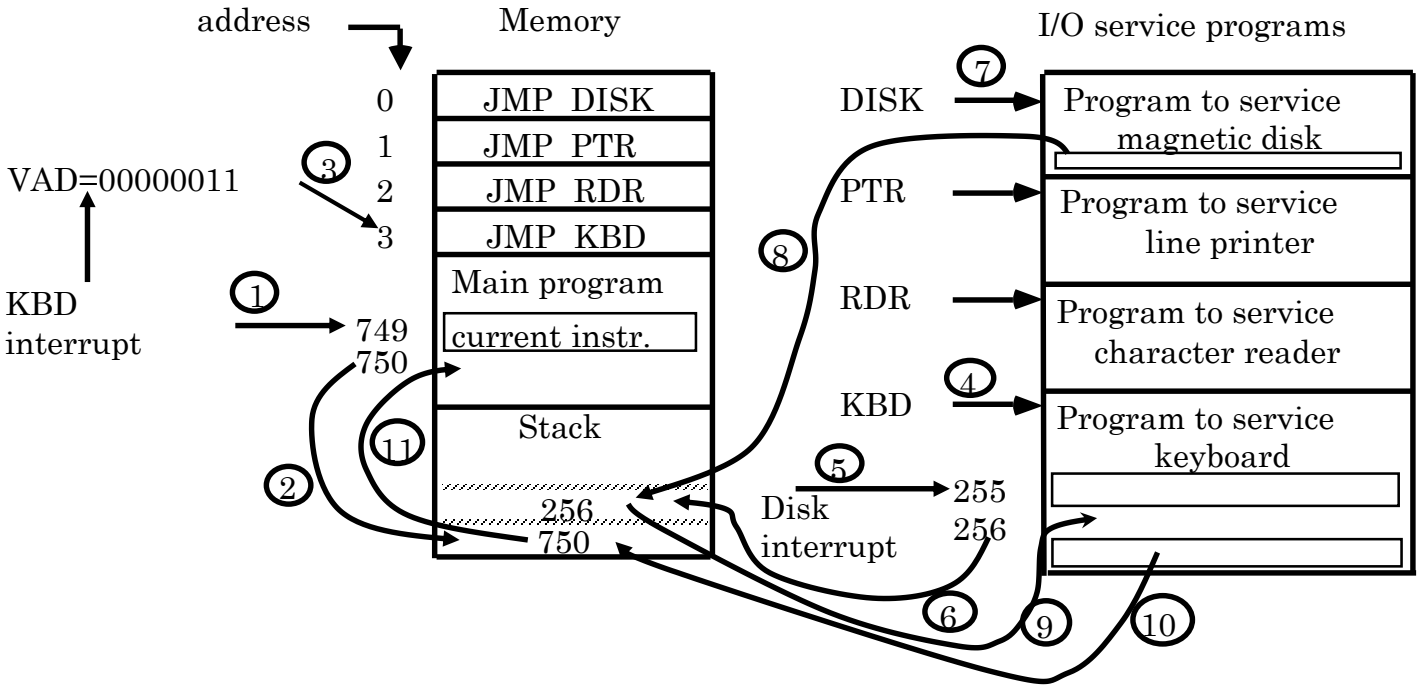
- The IEN can be set and cleared by program instructions. When IEN is cleared, the interrupt request coming from IST is neglected by CPU.
- At the end of each Instruction cycle
  - CPU checks IEN and IST
  - If IEN • IST = 1, CPU -> Interrupt Cycle

During the interrupt cycle the CPU performs the following sequence of Micro-Operations:

SP ← SP - 1	Decrement stack pointer
M[SP] ← PC	Push PC into stack
INTACK ← 1	Enable interrupt acknowledge
PC ← VAD	Transfer vector address to PC
IEN ← 0	Disable further interrupts

Go To Fetch next instruction.

INTERRUPT SERVICE ROUTINE

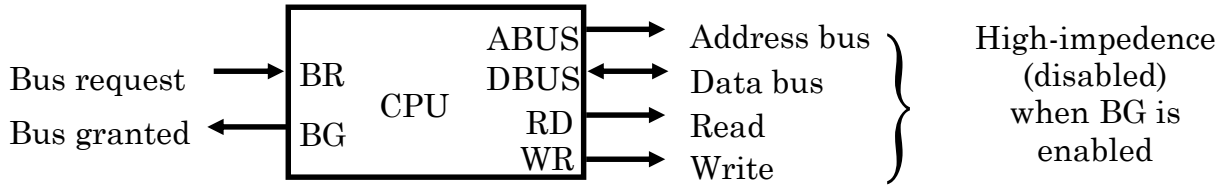


CPU is executing the instruction at 749 of the main program. At that time a interrupt comes from keyboard (KBD). Then computers goes to the interrupt cycle, it stores the return address 750 in the stack and then takes the vector address 00000011 from the bus and transfer it to the PC. The instruction at location 3 is executed next, resulting in transfer the control to the KBD program. Now CPU executing the KBD program's 255 address instruction , then another interrupt comes from the DISK. Then CPU store the return address 256 in stack and jumps to DISK program. After completing the DISK program , CPU takes the return address from stack which is 256, after completing the KBD program CPU takes next return address 750.

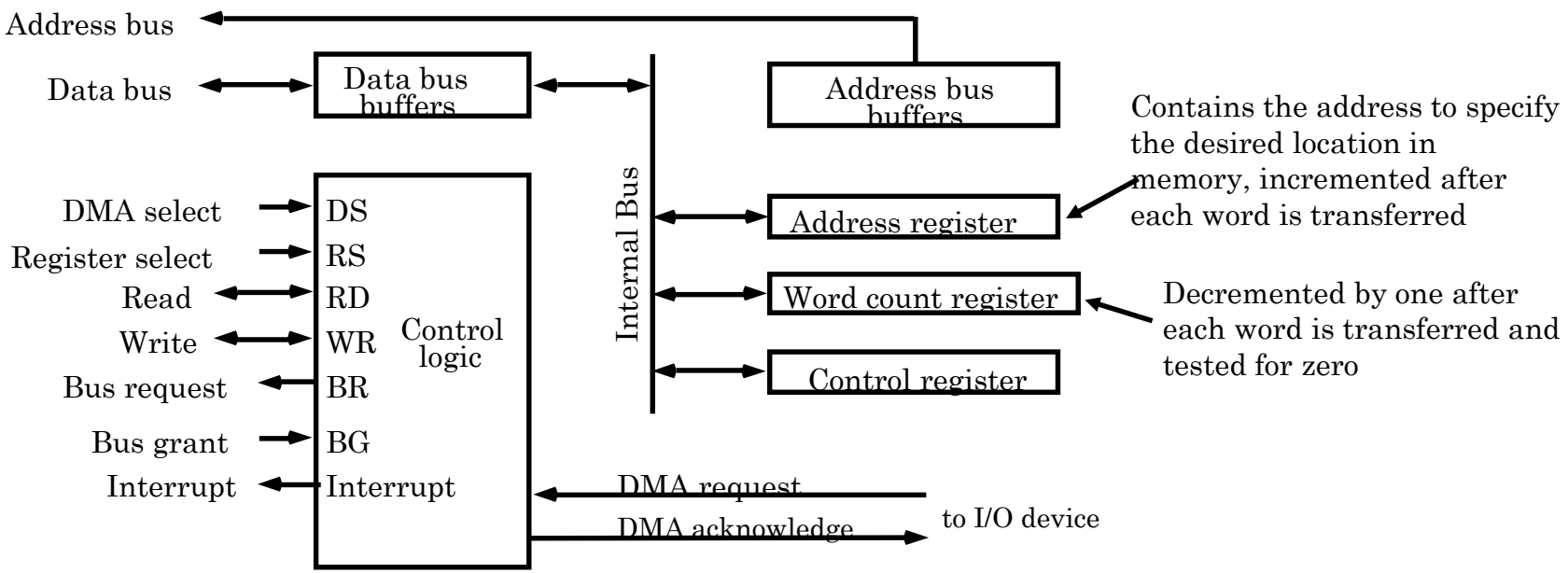
DIRECT MEMORY ACCESS

* Block of data transfer from high speed devices, Drum, Disk, Tape
* DMA controller - Interface which allows I/O transfer directly between Memory and Device, freeing CPU for other tasks
* CPU initializes DMA Controller by sending memory address and the block size(number of words)

CPU bus signals for DMA transfer



Block diagram of DMA controller



DMA I/O OPERATION

The DMA is initialized by the CPU. The CPU initializes the DMA by sending the following information through the data bus:

1. The starting address of memory block where data are available (for read) or where data are to be stored (for write).
2. The word count, which is the number of words in the memory block.
3. Control to specify the mode of transfer such as read or write.
4. A control to start the DMA transfer (GO command)

Upon receiving a GO Command DMA performs I/O operation.

BURST TRANSFER / CYCLE STEALING

When DMA takes control of the bus system, it communicate directly with The memory. The transfer can be made in several ways.

BURST TRANSFER

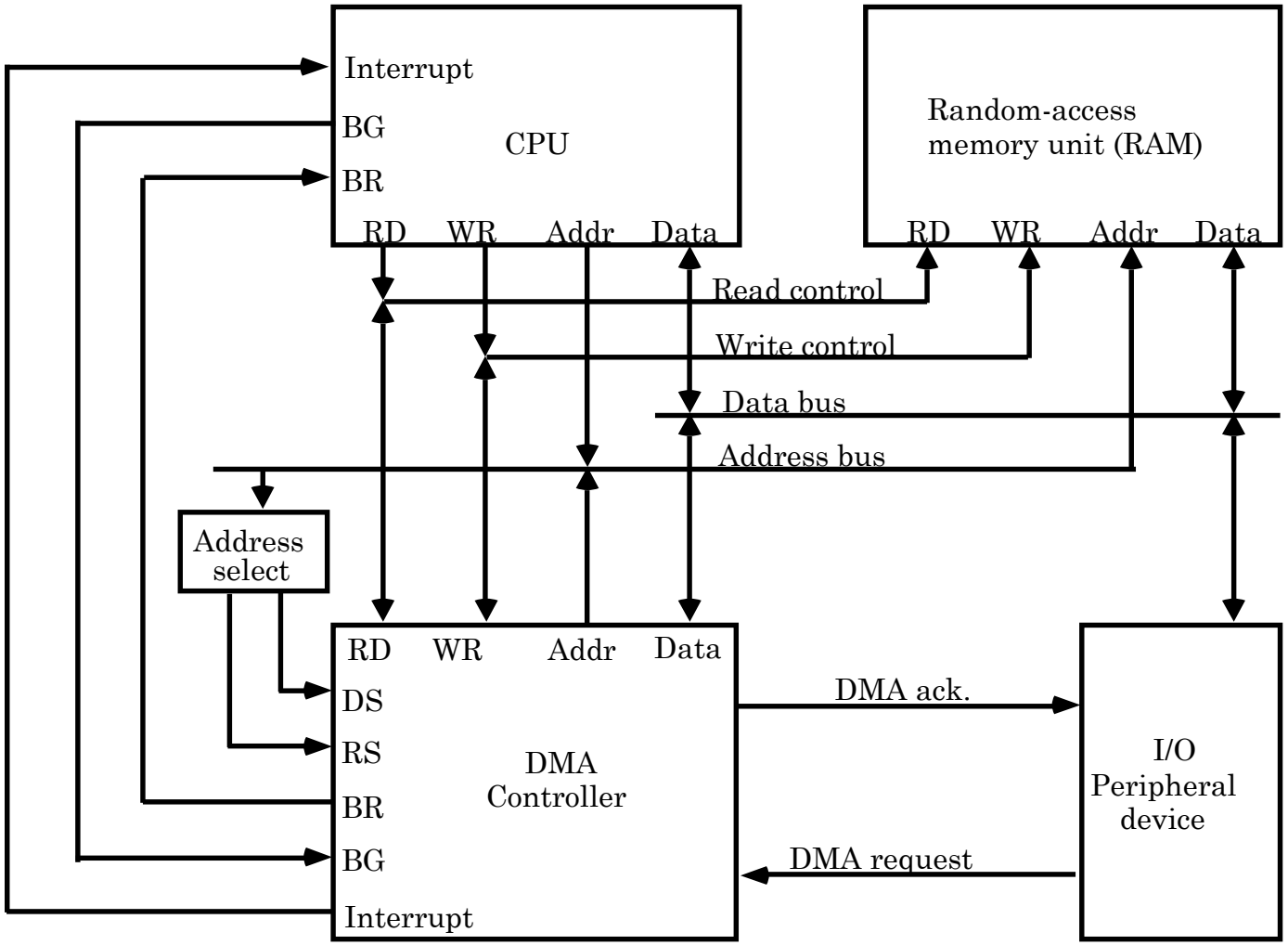
In DMA burst transfer, a block sequence consisting of a number of memory word is transferred in a continuous burst. This mode is needed for fast devices.

CYCLE STEALING

An alternative technique called Cycle Stealing allows the DMA controller to transfer one data word at a time, after which it must return control of the buses to the CPU.

- CPU is usually much faster than I/O(DMA), thus CPU uses the most of the memory cycles
- DMA Controller steals the memory cycles from CPU
- For those stolen cycles, CPU remains idle
- DMA Controller may steal most of the memory cycles which may cause CPU remain idle long time

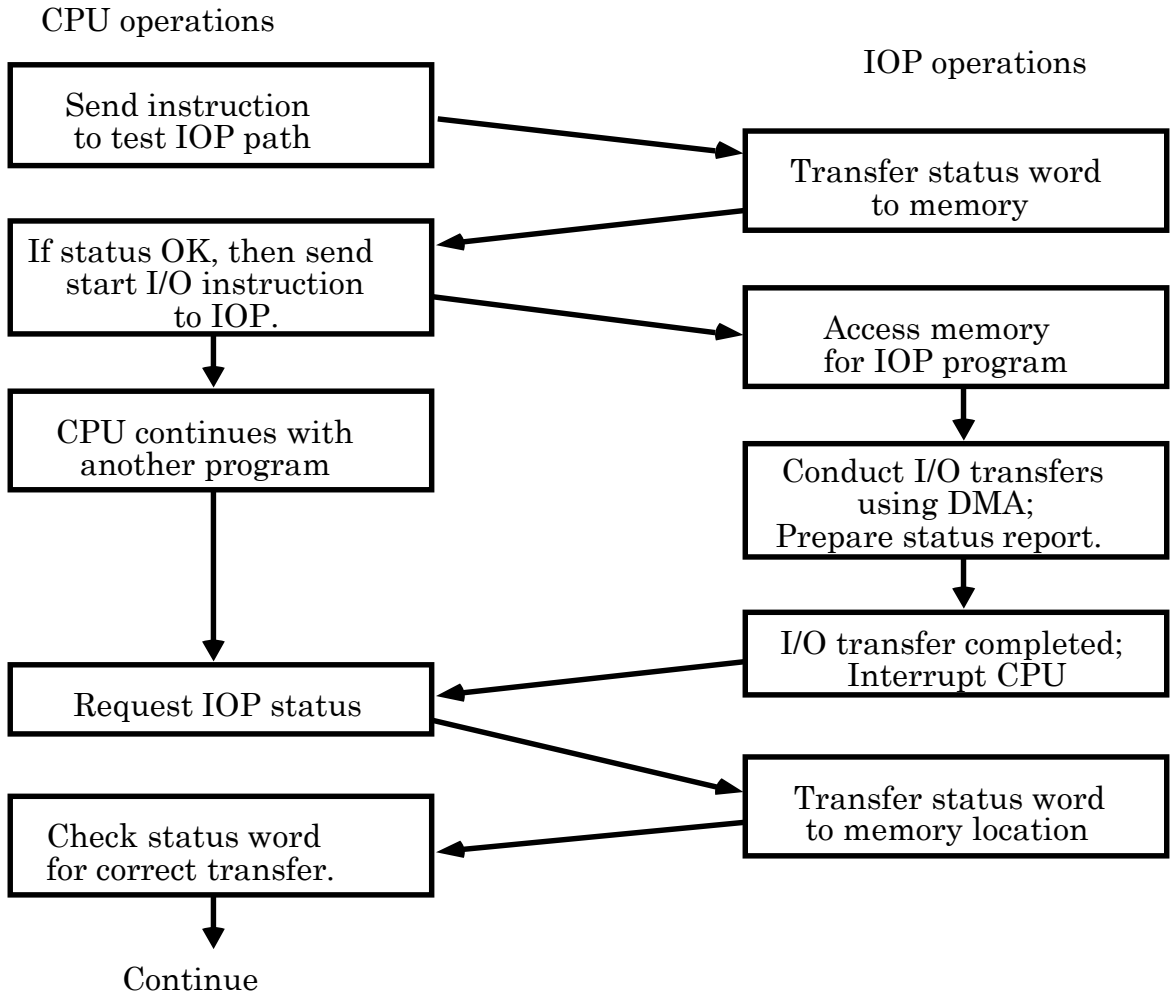
DMA TRANSFER



INPUT/OUTPUT PROCESSOR (IOP)

Instead of having each interface communicate with the CPU, a computer may incorporate one or more external processors and design them the task of communicating directly with all I/O devices with DMA capability. This external processor is know as Input/Output processors or IOPs. A processor that communicate media in a serial fashion (like telephone line used) is called a data communication processor (DCP). In addition IOP can perform other processing tasks, such as arithmetic, logic, branching, and code translation. [The IOP is also know as channel]

CPU-IOP COMMUNICATION



The memory unit acts as a message center where each processor leaves information for other.

Thank you