Digital Systems Design

Microcomputer System Design

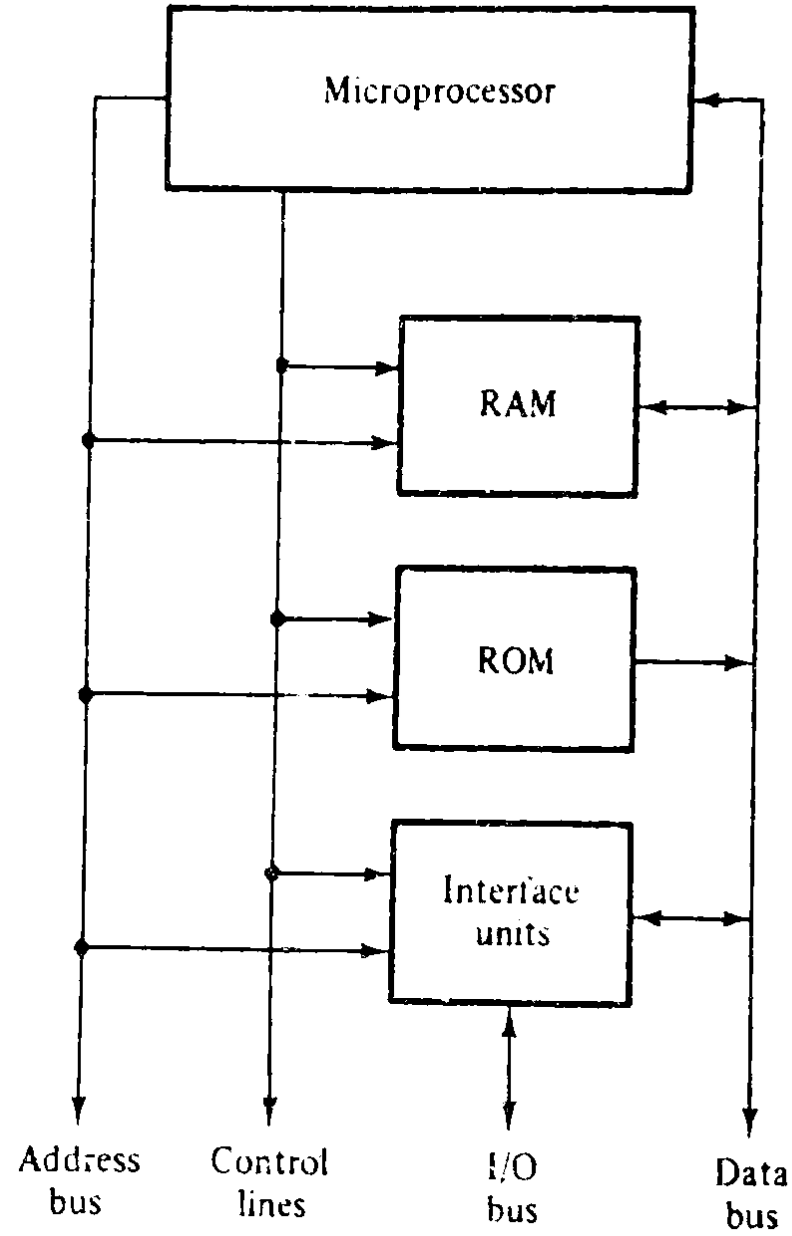
Introduction

- A microprocessor combined with memory and interface modules is called a *microcomputer*.
- The term microcomputer is used to indicate a small-size computer system consisting of the three basic units: CPU, memory and input-output interface.

Microcomputer Organization

- A typical microcomputer system consists of –
  - microprocessor
  - memory
  - I/O interface

  Various components that form the system are linked through buses that transfer instructions, data, addresses and control information among the IC components.

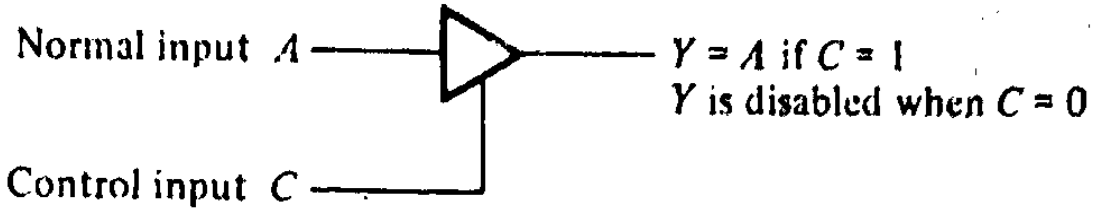


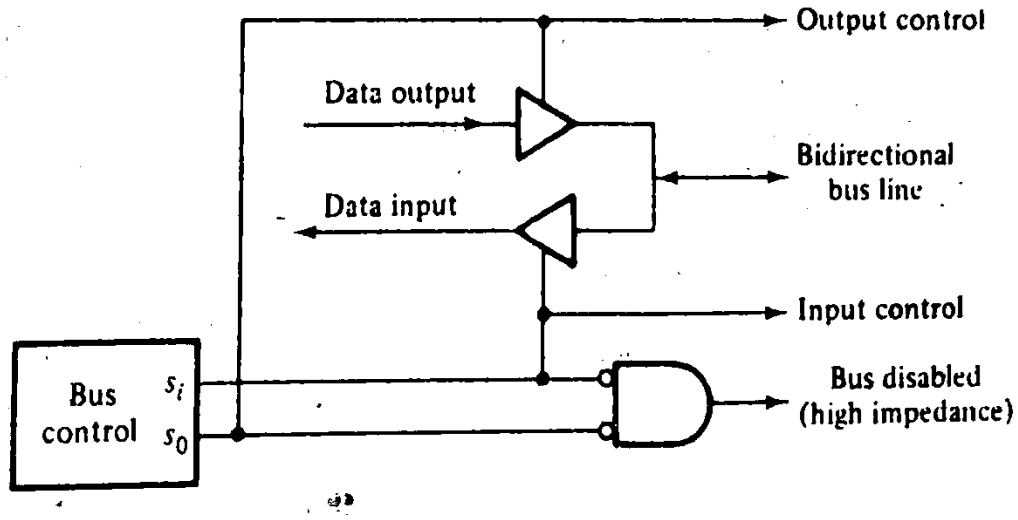
Microcomputer Organization

- The microprocessor contains a number of registers, an arithmetic logic unit and timing and control logic.
- Externally, it provides a bus system.
- The communication between the LSI components in a micrcomputer takes place via the address and data buses.
  - The address bus is unidirectional i.e. Microprocessor to other units
  - The data bus is bidirectional i.e. Information can flow in either direction.

Microcomputer Organization
Bus Buffer

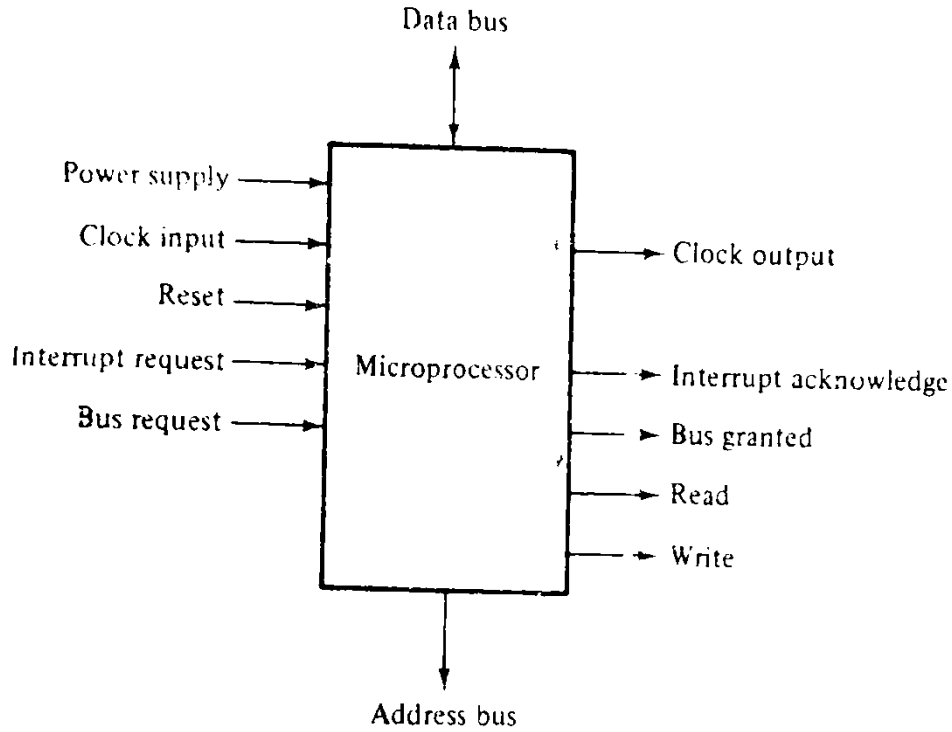
- The bus system in a microprocessor is commonly implemented by means of bus buffers constructed with three-state gates.
- A three-state or tri-state gate is a digital circuit that exhibits three output states.
  - Two are signals equivalent to binary 1 and 0 as in conventional gates.
  - The third state is called a high–impedance state i.e. The output is disabled.
- A bidirectional bus can be constructed with bus buffers to control the direction of information flow.





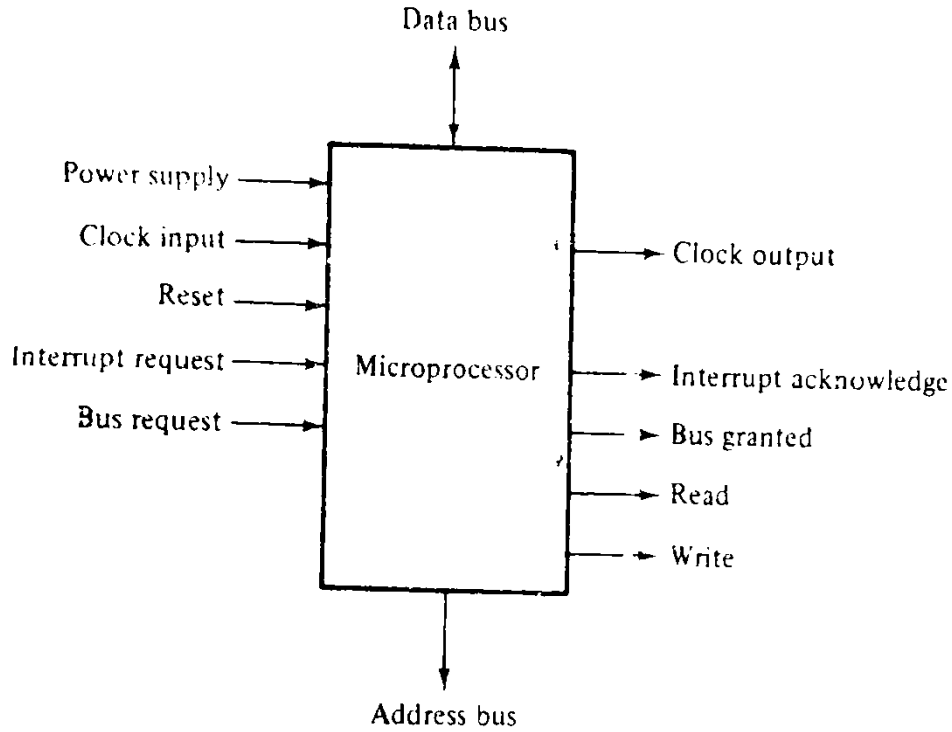
Microprocessor Organization
Typical Set of Control Signals

- *Clock* input – generate multiphase clock pulses that provide timing and control for internal functions.
- *Reset* input – reset and start the microprocessor after power is turned on.
- *Interrupt* request – when it is acknowledged, microprocessor suspends the execution of the current program and branches to a program that services the interface module.



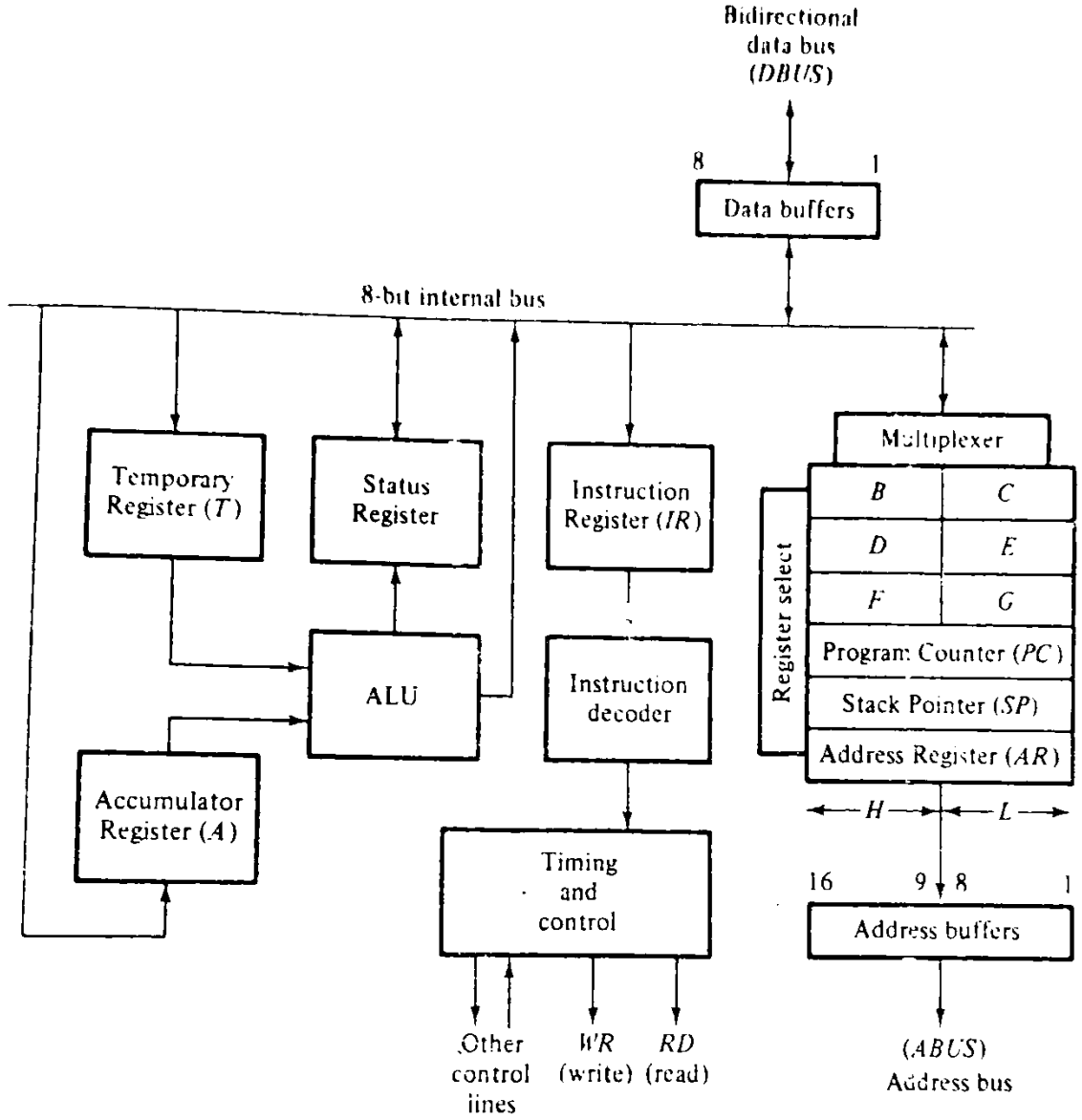
Microprocessor Organization
Typical Set of Control Signals

- *Bus-request* – temporarily suspends operation and drives all buses into their high-impedance state.
- *Bus-granted* – when bus-request is acknowledged bus-granted in enabled to let external devices access to memory directly.
- *Read* and *write* – read line informs to accept data from data bus and write line indicates that microprocessor is in output mode and valid data are available on data bus.



Microprocessor Organization
CPU Example

- The status register holds –
  - End carry, sign-bit, zero-result
- The address buffers receive from –
  - Program Counter (*PC*), Stack Pointer (*SP*), Address Register (*AR*)



Microprocessor Organization
Memory Cycle

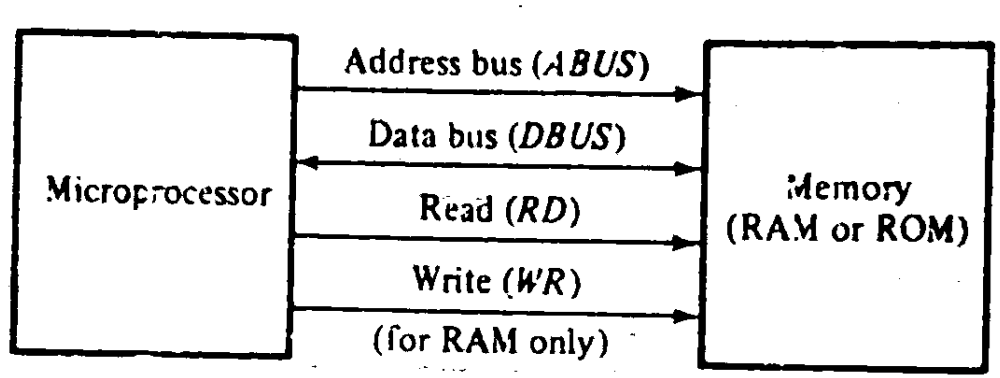
- In the read cycle, *RD* is enabled.
- When microprocessor transfers byte from *DBUS* to *A*, *RD* is disabled, indicating that end of memory transfer.
- The three operations can be combined into –

$$A \leftarrow M[AR]$$

- The write cycle $M[AR] \leftarrow A$ can be expressed as –

$$ABUS \leftarrow AR,\ DBUS \leftarrow A,\ WR \leftarrow 1$$

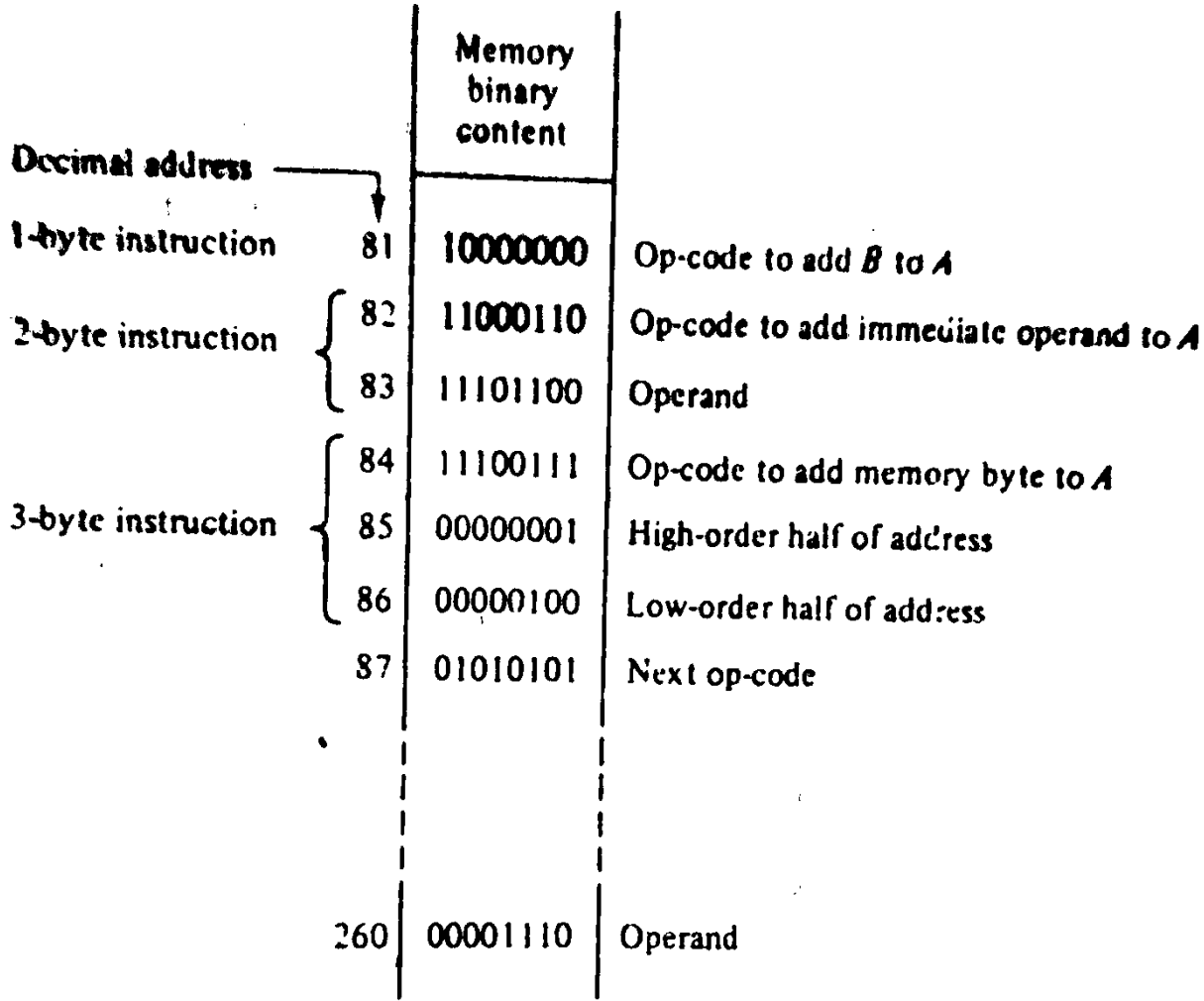
$$M[ABUS] \leftarrow DBUS,\ WR \leftarrow 0$$



$ABUS \leftarrow AR,\ RD \leftarrow 1$	address in bus for reading
$DBUS \leftarrow M[ABUS]$	memory reads byte
$A \leftarrow DBUS,\ RD \leftarrow 0$	byte transferred to A

Microprocessor Organization
Microprocessor Sequencing

Instruction	Byte 1	Byte 2	Byte 3	Function
Add *B* to *A*	Operation code	—	—	$A \leftarrow A + B$
Add immediate operand to *A*	Operation code	Operand	—	$A \leftarrow A +$ byte 2
Add operand specified by an address to *A*	Operation code	High-order half of address	Low-order half of address	$A \leftarrow A + M[address]$



Instructions and Addressing Modes
Basic Set of Microprocessor Instructions

- Microprocessor instruction may be classified into three distinct types:
  - Transfer instructions – move data among registers, memory words and interface registers.
  - Operation instructions – perform operations on data stored in registers or memory words.
  - Control instructions – test status conditions in registers and cause a change in program sequence depending on results.

Instructions and Addressing Modes
Basic Set of Microprocessor Instructions

- There are three types of control instructions:
  - Jump or branch instructions
  - Call to and return from subroutine instructions
  - Skip instructions

Instructions and Addressing Modes
Addressing Modes

- Several addressing modes:
  - Implied Mode
    - CMA (Complement *A*, $A \leftarrow A'$)
  - Register Mode
    - MOV *A*, *B*(Move *B* to *A*, $A \leftarrow B$)
  - Register-indirect Mode
    - MOV *A*, *FG* (Move *M*[*FG*] to *A*, $A \leftarrow M[FG]$)
  - Immediate Mode:
    - ADI *A*, *D*8 (Add immediate operand to *A*, *D*8 = 8-bit data operand, $A \leftarrow D8$)

Instructions and Addressing Modes
Addressing Modes

- Direct Addressing Mode
  - STA *AD*16 (Store *A* direct, *M*[*AD*16] ← *A*)
- Zero-page Addressing Mode
  - STA *AD*8 (Store *A* direct, *M*[*AD*8] ← *A*)
- Present-page Addressing Mode
  - STA *PC*(*H*)+*AD*8 (Store *A* direct, *M*[*PC*(*H*)+*AD*8] ← *A*)
- Relative Addressing Mode:
  - STA *PC*+*AD*8 (Store *A* direct, *M*[*PC*+*AD*8] ← *A*, range of *AD*8 is -128 to +127)

Instructions and Addressing Modes
Addressing Modes

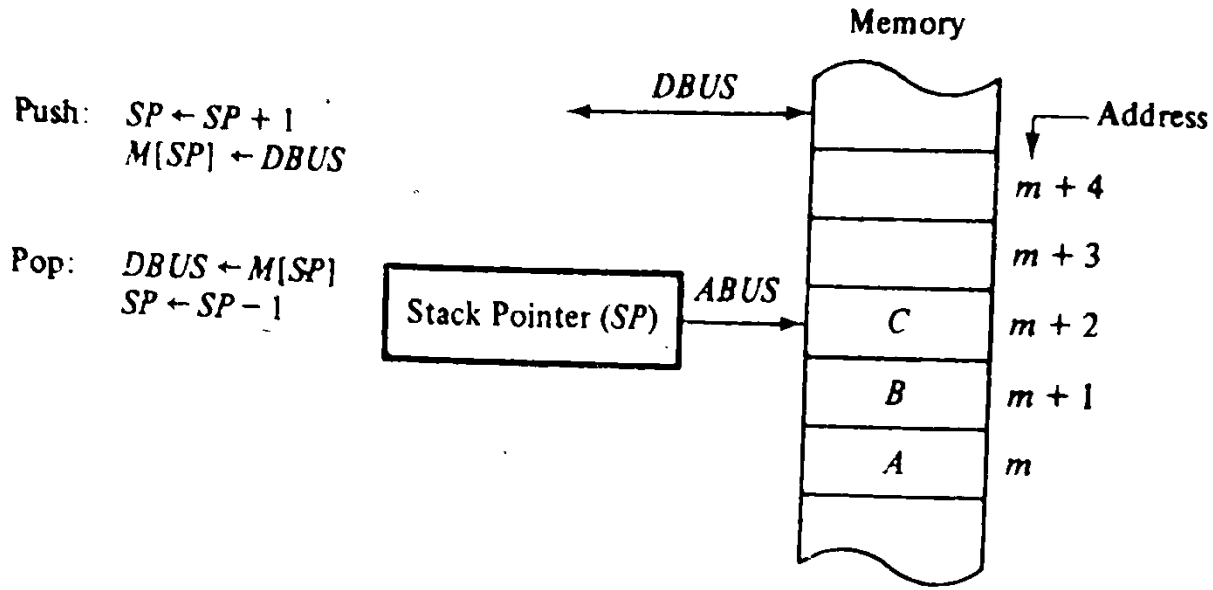
- Indexed Addressing Mode
  - STA *XR* + *AD*16(Store *A* direct, *M*[*XR* + *AD*16] ← *A*)
- Base-register Addressing Mode
  - STA *XR* + *AD*8 (Store *A* direct, *M*[*XR* + *AD*8] ← *A*)

Instructions and Addressing Modes
Addressing Modes

- Indirect Addressing Mode
  - STA $M[AD16]$ (Store A in indirect location, $M[M[AD16]] \leftarrow A$)
- Indexed-indirect Addressing Mode:
  - STA M[XR+$AD8$] (Store A indirect, $M[M[XR+AD8]] \leftarrow A$)
- Indirect-indexed Addressing Mode:
  - STA M[$AD8$]+XR (Store A indirect, $M[M[AD8]+XR] \leftarrow A$)

Memory Stack

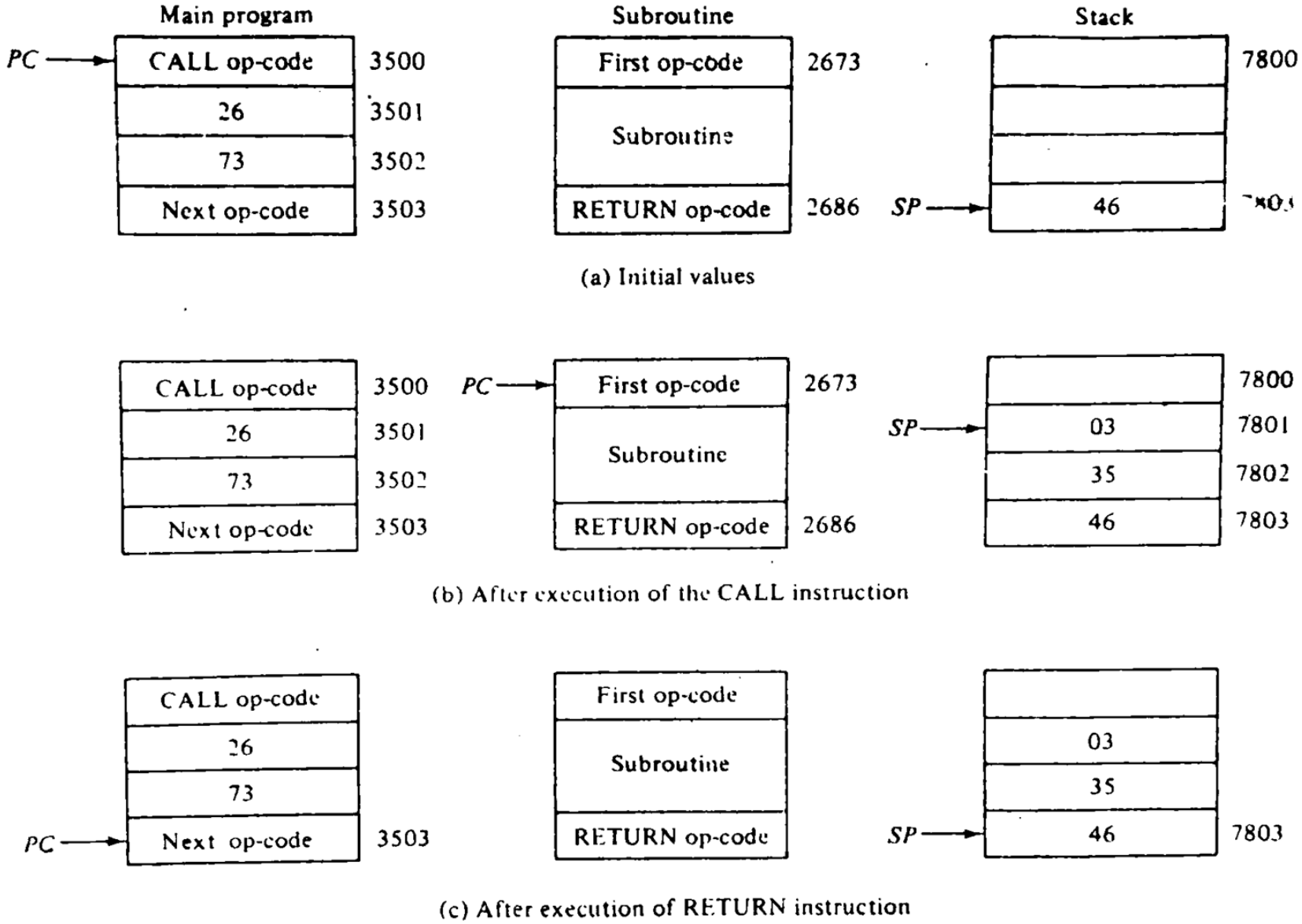
- It is essentially a portion of a memory unit accessed by an address that is always incremented or decremented after the memory access.
- The register holds the address for a stack is called a *stack pointer*.



Subroutine

$IR \leftarrow M[PC], PC \leftarrow PC + 1$
$AR(H) \leftarrow M[PC], PC \leftarrow PC + 1$
$AR(L) \leftarrow M[PC]\ PC \leftarrow PC + 1$
$SP \leftarrow SP + 1, \mathrm{M[SP]} \leftarrow PC(H)$
$SP \leftarrow SP + 1, M[SP] \leftarrow PC(L)$
$PC \leftarrow AR$

$IR \leftarrow M[PC], PC \leftarrow PC + 1$
$PC(L) \leftarrow M[SP], SP \leftarrow SP - 1$
$PC(H) \leftarrow M[SP], SP \leftarrow SP - 1$



Interrupt

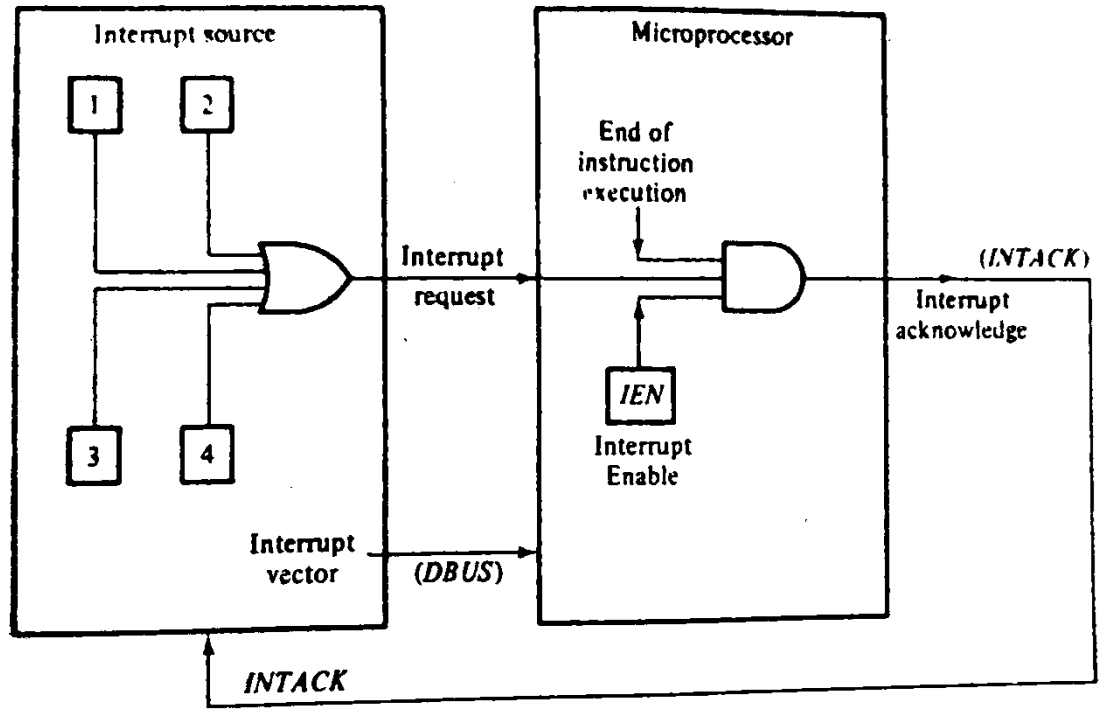
- Each interface module is capable of interrupting the microprocessors normal operation by providing a signal at the *interrupt* control input. **The interrupt is a signal emitted by hardware or software when a process or an event needs immediate attention**. It alerts the processor to a high-priority process requiring interruption of the current working process. In I/O devices one of the bus control lines is dedicated for this purpose and is called the Interrupt Service Routine (ISR).

- It may be either a request for service or an acknowledgement of service performed earlier by the interface.

- The interrupt procedure is quite similar to subroutine call –
  - As in a subroutine call, an interrupt stores the return address in the stack.
  - A subroutine–call instruction provides the branch address for the subroutine.

Interrupt

- Microprocessor may have single or multiple interrupt input lines.
- When interrupt enable (*IEN*) is cleared, interrupt request is neglected.
- If *IEN* is set and the microprocessor is at the end of the instruction execution, it acknowledges the interrupt by enabling *INTACK*.
- The interrupt source responds to *INTACK* by placing an interrupt vector into *DBUS*.



Interrupt

$SP \leftarrow SP + 1, M[SP] \leftarrow PC(H)$ push first byte of return address

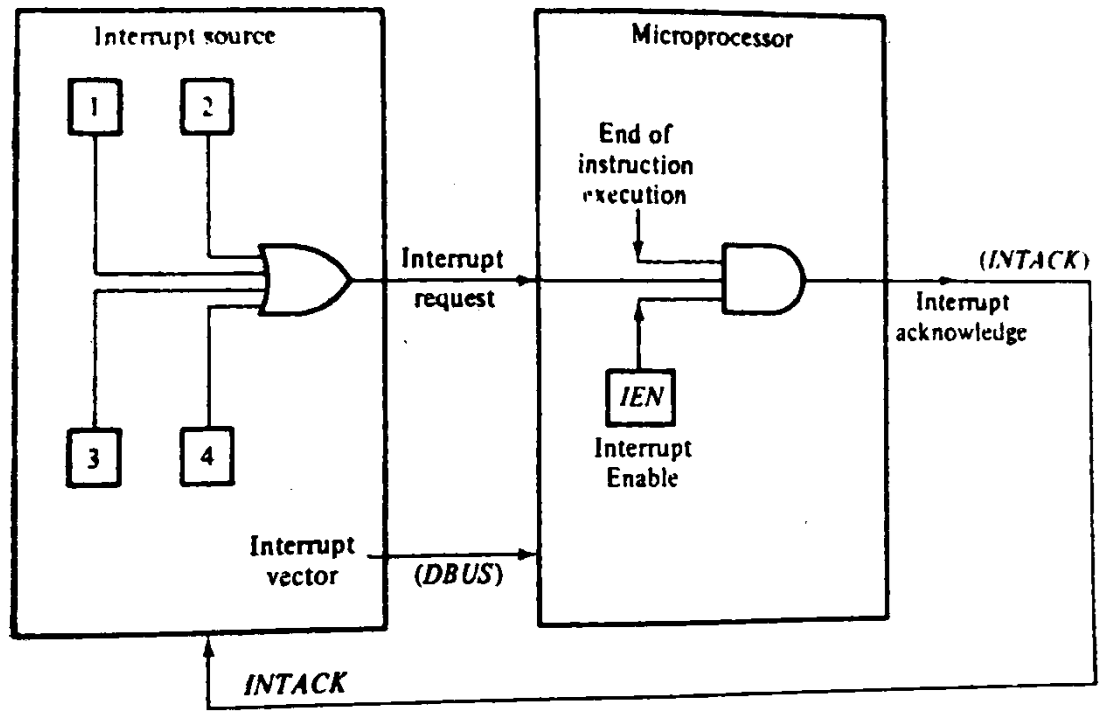
$SP \leftarrow SP + 1, M[SP] \leftarrow PC(L)$ push second byte of return address

$INTACK \leftarrow 1$ interrupt acknowledge

$PC(H) \leftarrow 0, PC(L) \leftarrow DBUS$

Transfer vector address to P*C*

$IEN \leftarrow 0$ disable further interrupts



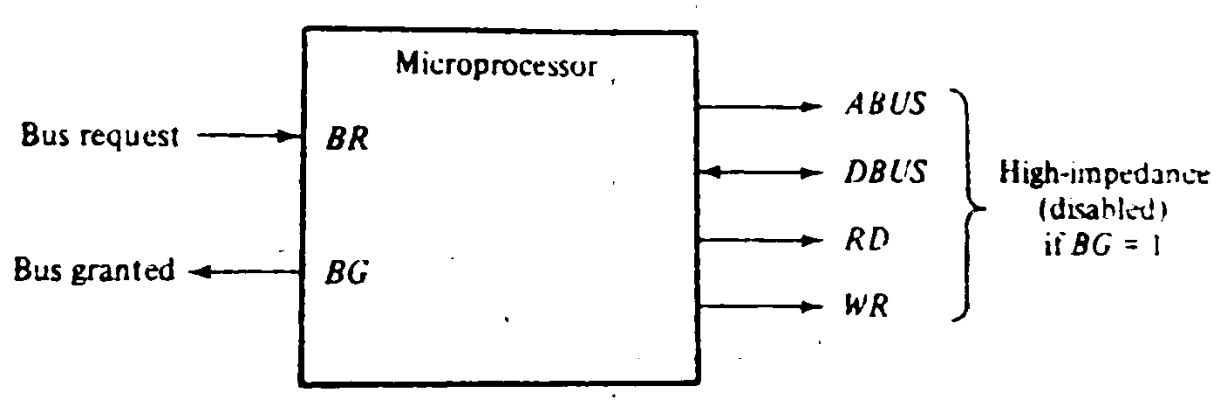
Direct Memory Access

This technique includes –

- Removing the processor during a transfer of data
- Letting the peripheral device manage the transfer directly to memory.
- During DMA transfer, the processor is idle; so it no longer has control of the system bus.
- A DMA controller takes over the buses to manage the transfer directly between the peripheral device and memory.
- Direct memory access (DMA) is the process of transferring data without the involvement of the processor itself. It is often used for transferring data to/from input/output devices. A separate DMA controller is required to handle the transfer. The controller notifies the DSP processor that it is ready for a transfer.

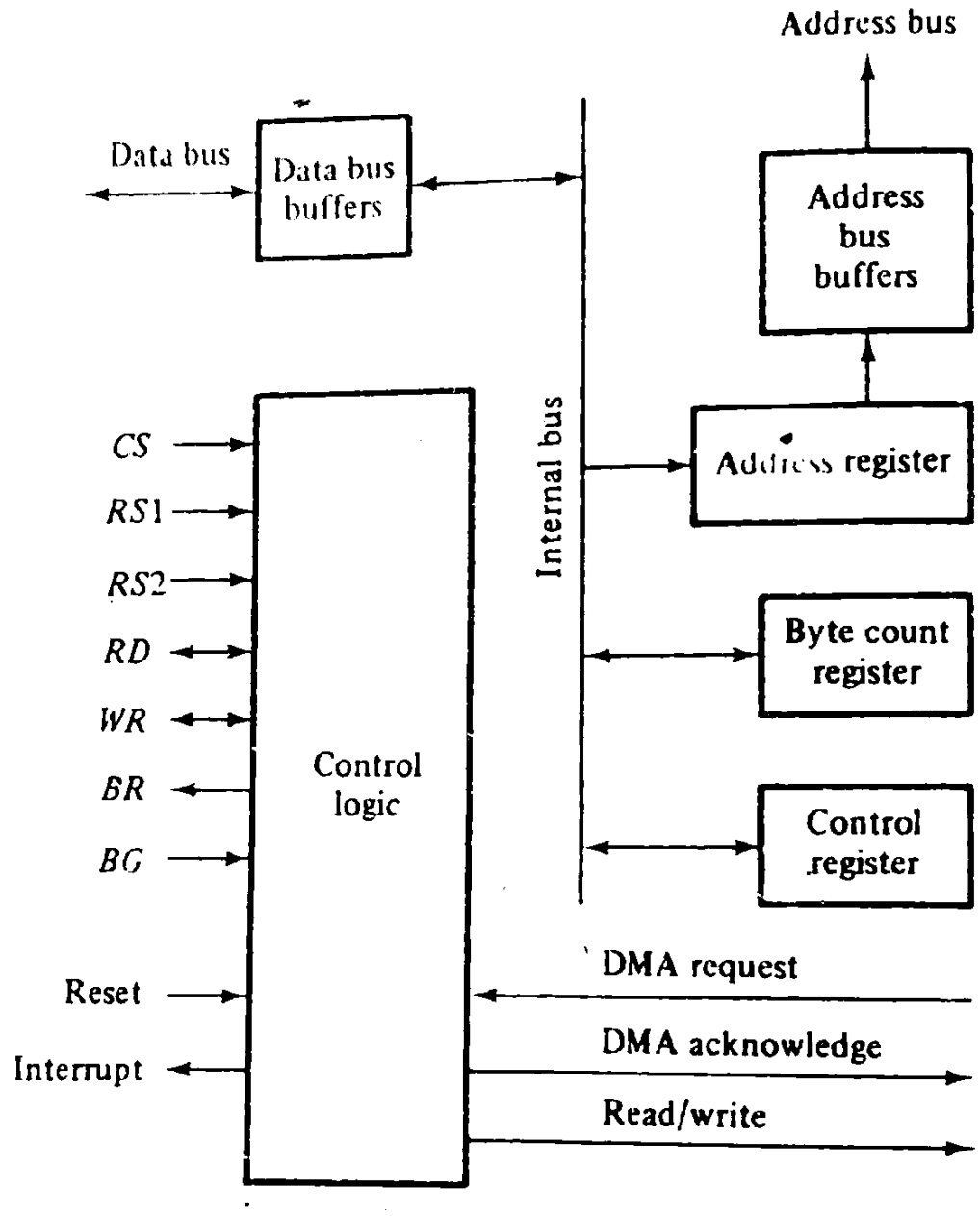
Direct Memory Access

- The *bus-request*, *BR* input, when in the 1-state, is a request to the microprocessor to disable its buses.
- Microprocessor terminates the execution of the current program and places its bus including *RD* and *WR* lines into high-impedence state.
- Then the processor places the *bus-granted*, *BG* output in the 1-state.
- As long as *BG* = 1, the microprocessor is idle.
- When *BR* becomes 0, *BG* also returns 0.
- *BR* line is called *hold* command.
- *BG* line is called *hold acknowledge*.



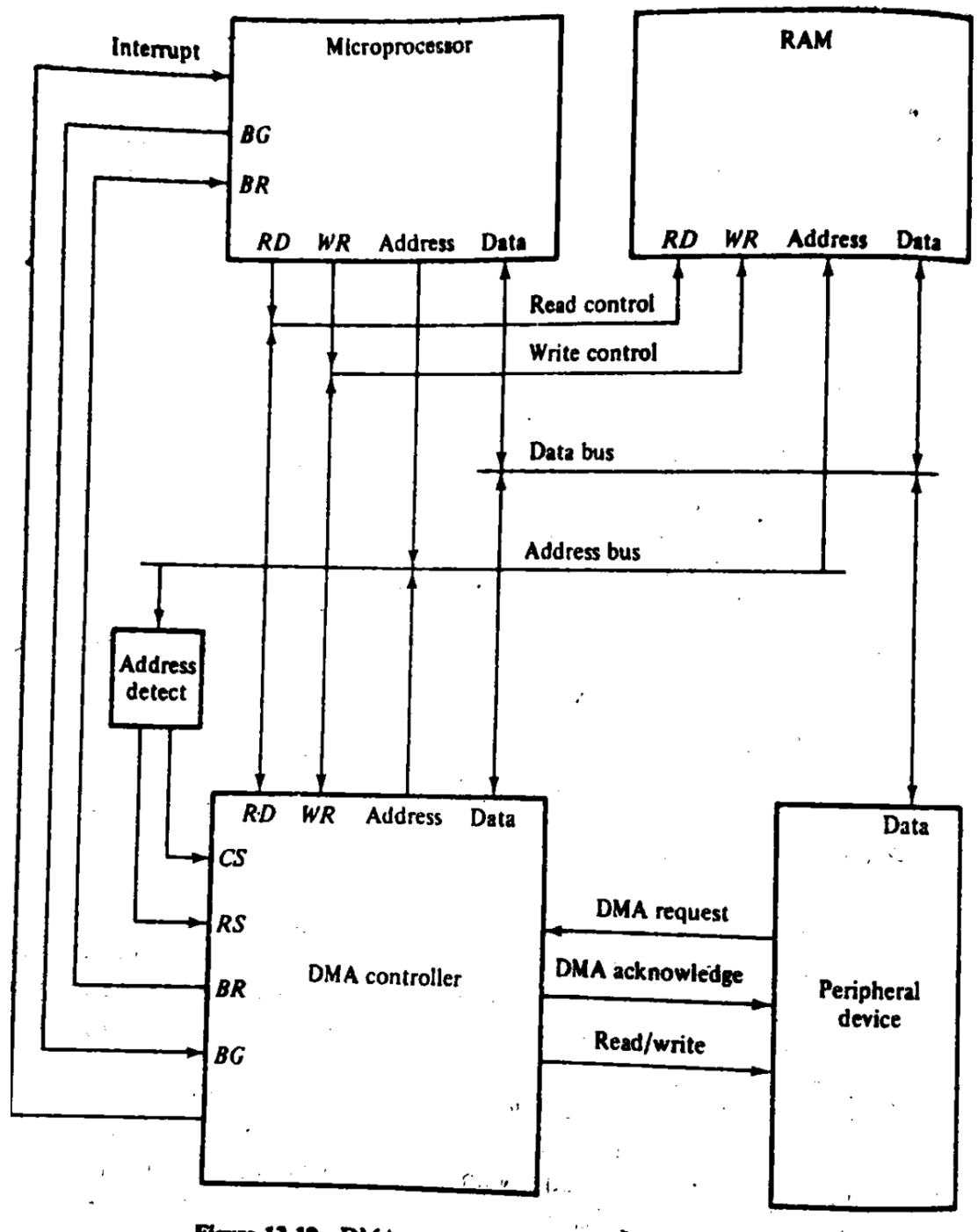
Direct Memory Access

- DMA controller needs three registers –
  - Address register
  - Byte count register
  - Control register
- All registers in the DMA appear to the microprocessor as an I/O interface.
- The control register specifies the mode of transfer.
- The DMA communicates with an external device through the *request* and *acknowledge* lines.



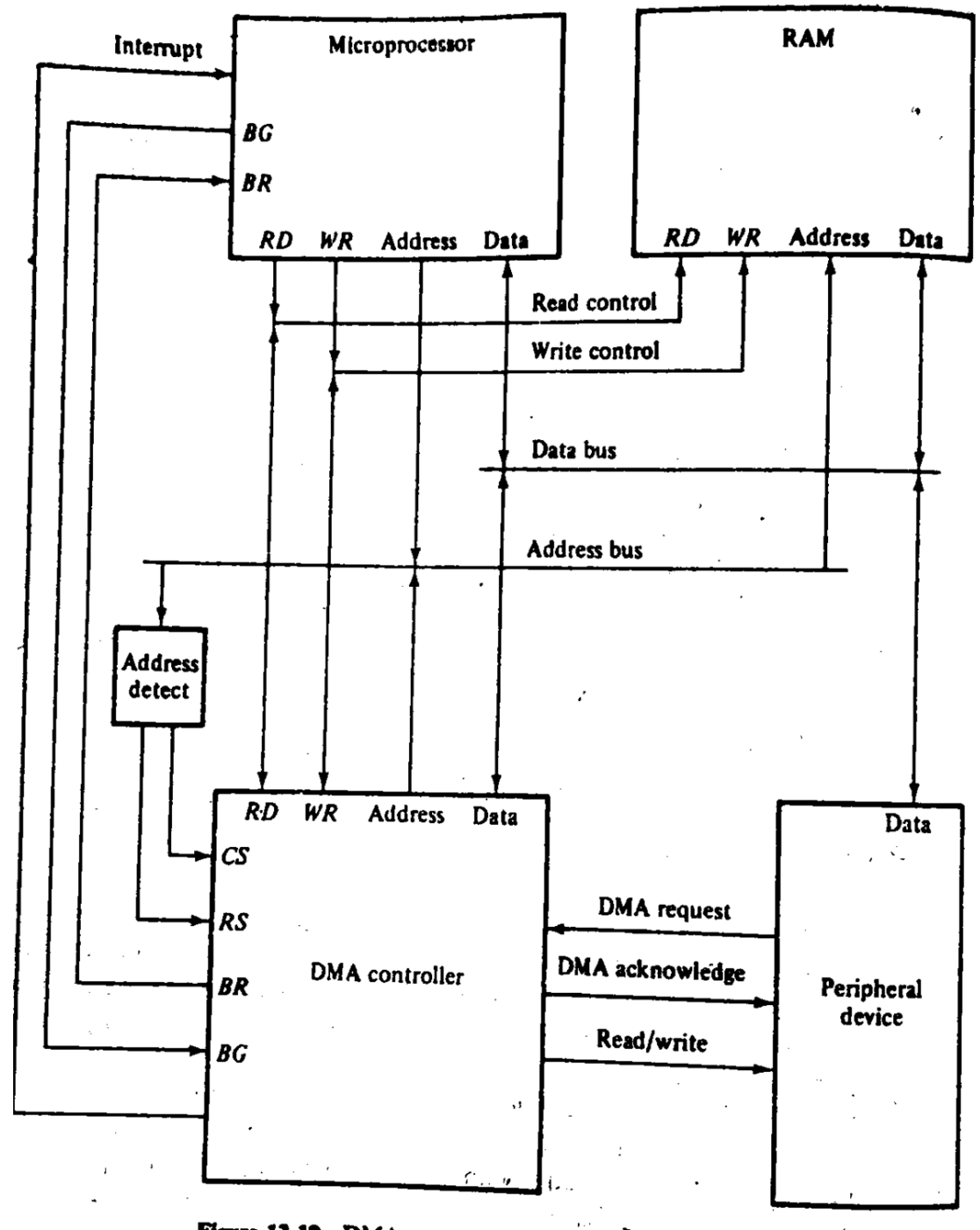
Direct Memory Access
Working Procedure

- The starting address is stored in the DMA address register.
- Byte count and control bits are stored in byte count register and control register respectively.
- Microprocessor communicates with DMA controller through address and data buses as with any interface unit.
- Once start control bit is received, it can start transfer between the peripheral device and system RAM.



Direct Memory Access
Working Procedure

- Receiving DMA request from peripheral device, DMA controller activates *BR* line.
- Microprocessor responds with *BG* line.
- DMA puts its address register value onto the address bus, initiates RD/WR signal and sends DMA acknowledge to the peripheral device.
- The peripheral unit can communicate with RAM through the data bus for direct transfer.



Direct Memory Access
Working Procedure

- If the byte count register reaches zero, DMA stops any further transfer and removes its bus request.
- It also informs the microprocessor of the termination by means of an interrupt request.