INPUT-OUTPUT ORGANIZATION

- Peripheral Devices
- Input-Output Interface
- Asynchronous Data Transfer
- Modes of Transfer
- Priority Interrupt
- Direct Memory Access
- Input-Output Processor
- Serial Communication

PERIPHERAL DEVICES

Input Devices

- Keyboard
- Optical input devices
  - Card Reader
  - Paper Tape Reader
  - Bar code reader
  - Digitizer
  - Optical Mark Reader
- Magnetic Input Devices
  - Magnetic Stripe Reader
- Screen Input Devices
  - Touch Screen
  - Light Pen
  - Mouse
- Analog Input Devices

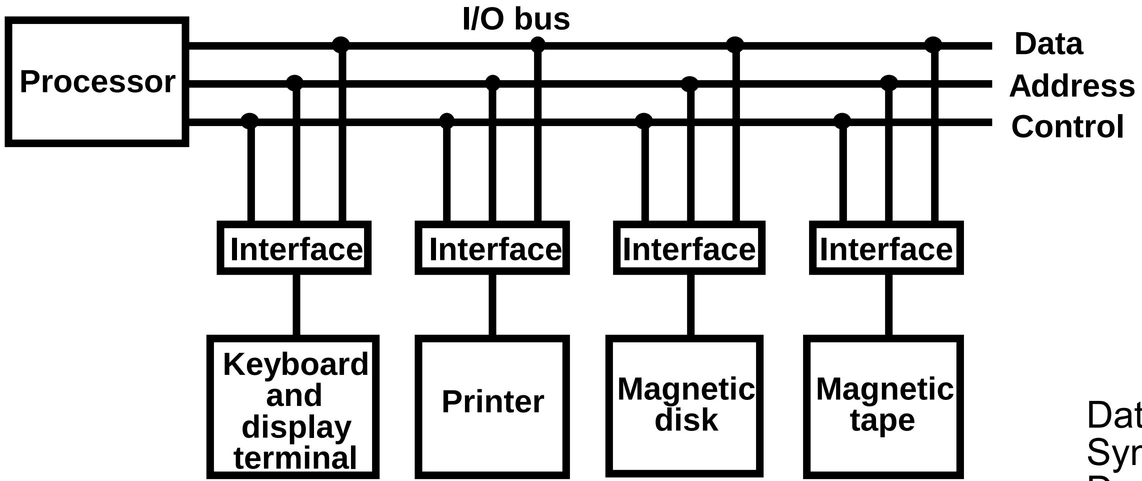
Output Devices

- Card Puncher, Paper Tape Puncher
- CRT
- Printer (Impact, Ink Jet, Laser, Dot Matrix)
- Plotter
- Analog
- Voice

INPUT/OUTPUT INTERFACE

- **Provides a method for transferring information between internal storage (such as memory and CPU registers) and external I/O devices**
- **Resolves the *differences* between the computer and peripheral devices**
  - **Peripherals - Electromechanical Devices**
  - **CPU or Memory - Electronic Device**
  - **Data Transfer Rate**
    - » **Peripherals - Usually slower**
    - » **CPU or Memory - Usually faster than peripherals**
      - **Some kinds of Synchronization mechanism may be needed**
  - **Unit of Information**
    - » **Peripherals – Byte, Block, …**
    - » **CPU or Memory – Word**
  - **Data representations may differ**

I/O BUS AND INTERFACE MODULES



Data Conversion
Synchronization
Device Selection

Each peripheral has an interface module associated with it

Interface

- **Decodes the device address (device code)**
- **Decodes the commands (operation)**
- **Provides signals for the peripheral controller**
- **Synchronizes the data flow and supervises the transfer rate between peripheral and CPU or Memory**

Typical I/O instruction

Op. code	Device address	Function code

(Command)

I/O BUS AND MEMORY BUS

Functions of Buses

*** *MEMORY BUS* is for information transfers between CPU and the MM**

*** *I/O BUS* is for information transfers between CPU and I/O devices through their I/O interface**

Physical Organizations

*** Many computers use a common single bus system for both memory and I/O interface units**

- **Use one common bus but separate control lines for each function**
- **Use one common bus with common control lines for both functions**

*** Some computer systems use two separate buses, one to communicate with memory and the other with I/O interfaces**

I/O Bus

- **Communication between CPU and all interface units is via a common I/O Bus**
- **An interface connected to a peripheral device may have a number of *data registers* , a *control register*, and a *status register***
- **A command is passed to the peripheral by sending to the appropriate interface register**
- **Function code and sense lines are not needed (Transfer of data, control, and status information is always via the common I/O Bus)**

ISOLATED vs MEMORY MAPPED I/O

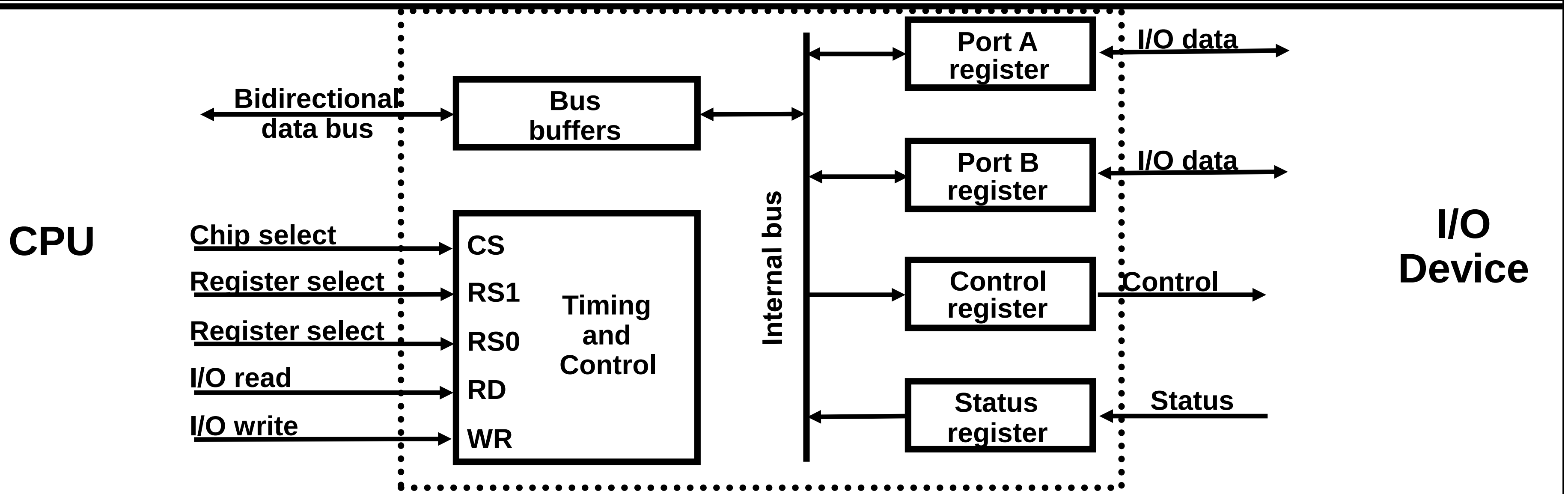
Isolated I/O

- Separate I/O read/write control lines in addition to memory read/write control lines
- Separate (isolated) memory and I/O address spaces
- Distinct input and output instructions

Memory-mapped I/O

- A single set of read/write control lines
  (no distinction between memory and I/O transfer)
- Memory and I/O addresses share the common address space
  -> reduces memory address range available
- No specific input or output instruction
  -> The same memory reference instructions can
  be used for I/O transfers
- Considerable flexibility in handling I/O operations

I/O INTERFACE



CS	RS1	RS0	Register selected
0	x	x	None - data bus in high-impedence
1	0	0	Port A register
1	0	1	Port B register
1	1	0	Control register
1	1	1	Status register

Programmable Interface

- Information in each port can be assigned a meaning depending on the mode of operation of the I/O device
  → Port A = Data; Port B = Command; Port C = Status

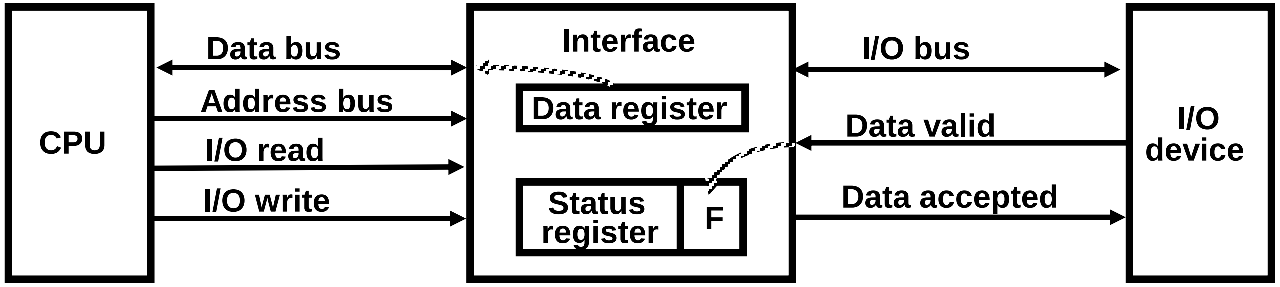
- CPU initializes(loads) each port by transferring a byte to the Control Register
  → Allows CPU can define the mode of operation of each port
  → *Programmable Port*: By changing the bits in the control register, it is possible to change the interface characteristics

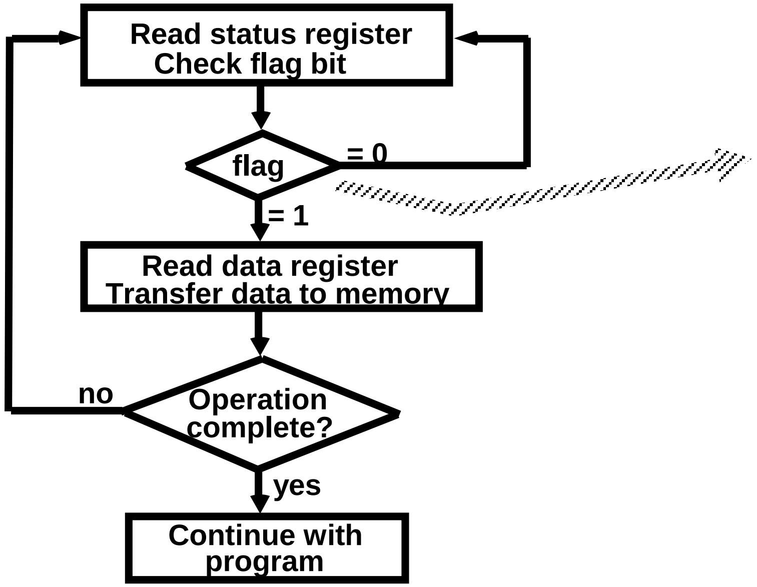
MODES OF TRANSFER - PROGRAM-CONTROLLED I/O -

3 different Data Transfer Modes between the central computer(CPU or Memory) and peripherals;

Program-Controlled I/O
Interrupt-Initiated I/O
Direct Memory Access (DMA)

Program-Controlled I/O(Input Dev to CPU)





Polling or Status Checking

- Continuous CPU involvement
- CPU slowed down to I/O speed
- Simple
- Least hardware

MODES OF TRANSFER - INTERRUPT INITIATED I/O & DMA

Interrupt Initiated I/O

- Polling takes valuable CPU time
- Open communication only when some data has to be passed -> *Interrupt*.
- I/O interface, instead of the CPU, monitors the I/O device
- When the interface determines that the I/O device is ready for data transfer, it generates an *Interrupt Request* to the CPU
- Upon detecting an interrupt, CPU stops momentarily the task it is doing, branches to the service routine to process the data transfer, and then returns to the task it was performing

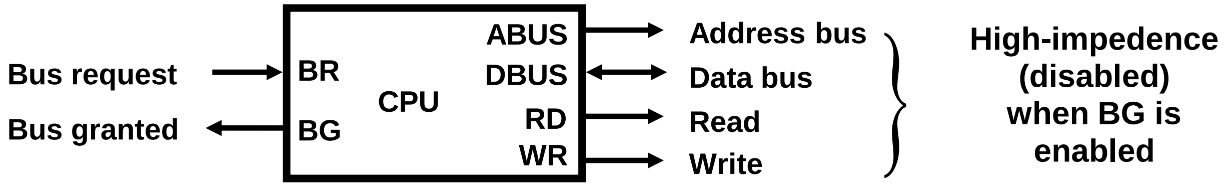
DMA (Direct Memory Access)

- Large blocks of data transferred at a high speed to or from high speed devices, magnetic drums, disks, tapes, etc.
- DMA controller
  Interface that provides I/O transfer of data directly to and from the memory and the I/O device
- CPU initializes the DMA controller by sending a memory address and the number of words to be transferred
- Actual transfer of data is done directly between the device and memory through DMA controller
  -> Freeing CPU for other tasks

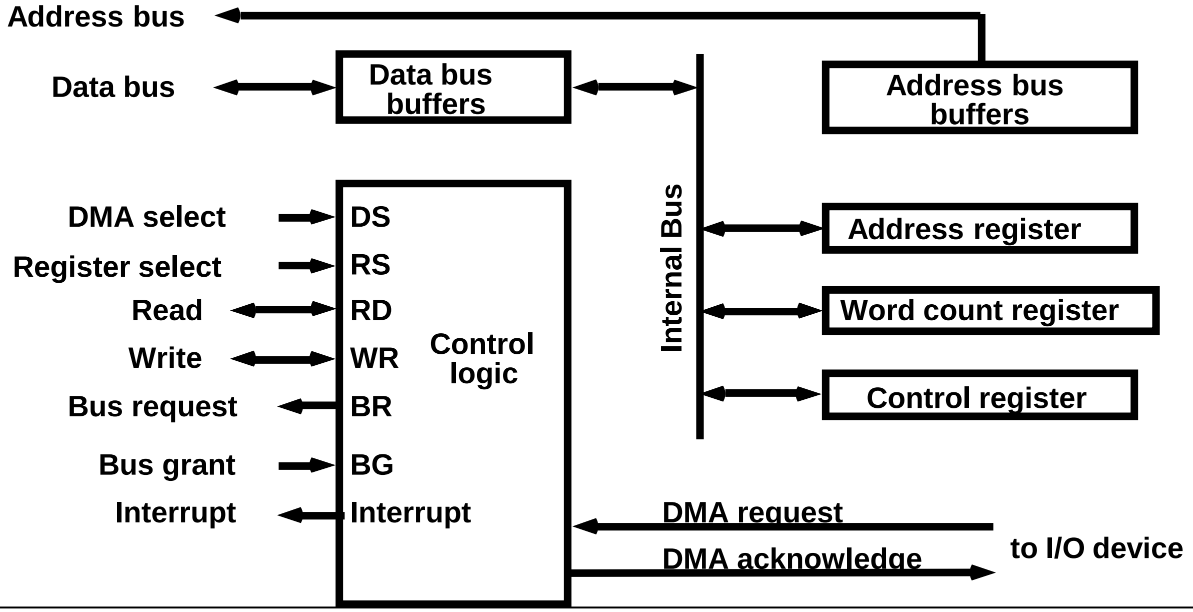
DIRECT MEMORY ACCESS

* 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

Starting an I/O
- **CPU executes instruction to**
  - **Load Memory Address Register**
  - **Load Word Counter**
  - **Load Function(Read or Write) to be performed**
  - **Issue a GO command**

Upon receiving a GO Command DMA performs I/O operation as follows independently from CPU

Input

[1] Input Device <- R (Read control signal)
[2] Buffer(DMA Controller) <- Input Byte; and
assembles the byte into a word until word is full
[4] M <- memory address, W(Write control signal)
[5] Address Reg <- Address Reg +1; WC(Word Counter) <- WC - 1
[6] If WC = 0, then Interrupt to acknowledge done, else go to [1]

Output

[1] M <- M Address, R
M Address R <- M Address R + 1, WC <- WC - 1
[2] Disassemble the word
[3] Buffer <- One byte; Output Device <- W, for all disassembled bytes
[4] If WC = 0, then Interrupt to acknowledge done, else go to [1]

CYCLE STEALING

While DMA I/O takes place, CPU is also executing instructions

DMA Controller and CPU both access Memory -> Memory Access Conflict

Memory Bus Controller

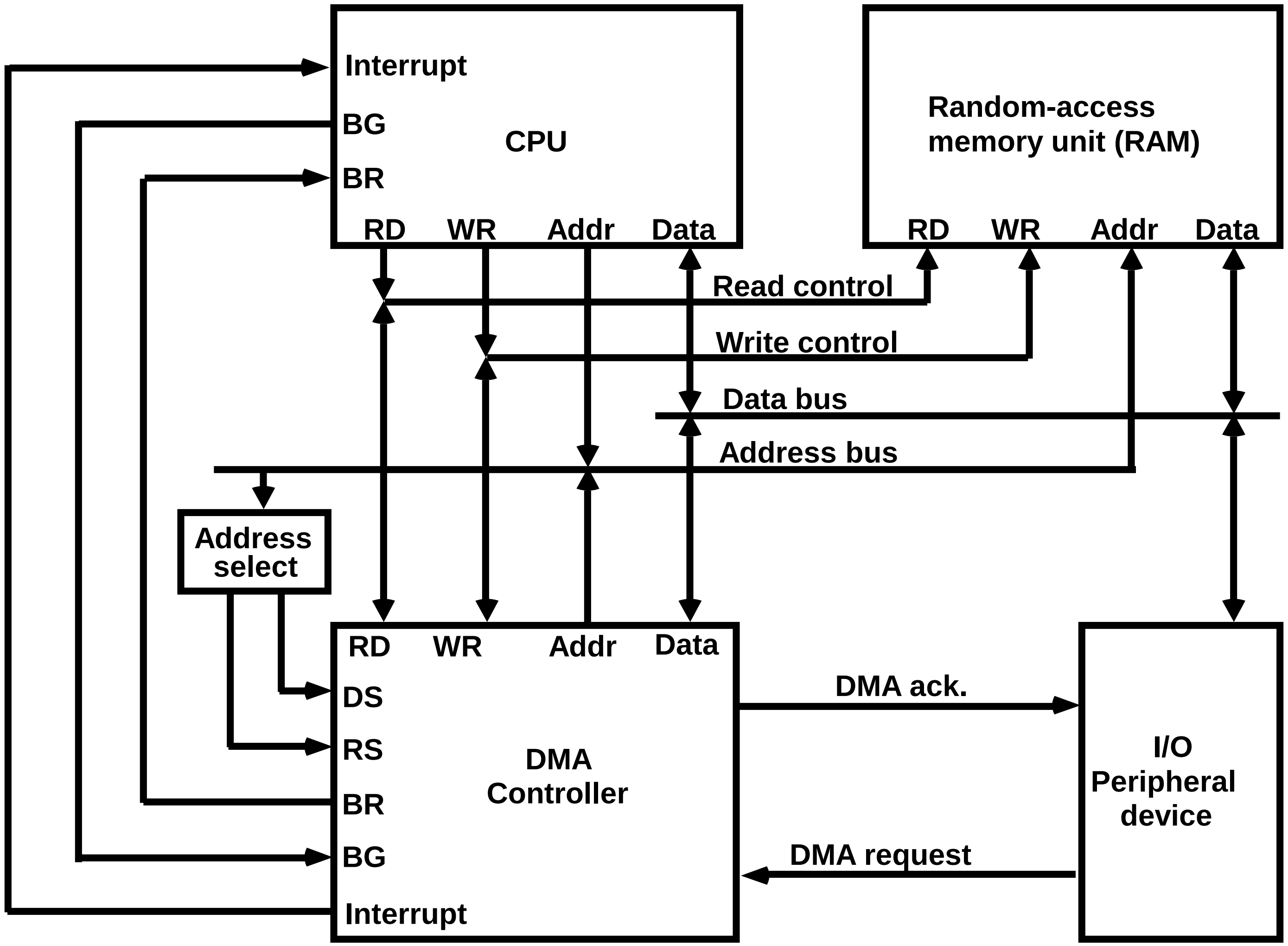
- Coordinating the activities of all devices requesting memory access
- Priority System

Memory accesses by CPU and DMA Controller are interwoven, with the top priority given to DMA Controller
-> Cycle Stealing

Cycle Steal

- 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
- For those slow CPU, DMA Controller may steal most of the memory cycles which may cause CPU remain idle long time

DMA TRANSFER


Interrupt
BG
BR
CPU
RD
WR
Addr
Data
Random-access memory unit (RAM)
RD
WR
Addr
Data
Read control
Write control
Data bus
Address bus
Address select
RD
WR
Addr
Data
DS
RS
BR
BG
Interrupt
DMA Controller
DMA ack.
DMA request
I/O Peripheral device