Disks and DMA

Peter Rounce
P.Rounce@cs.ucl.ac.uk

Secondary and Tertiary Storage

Primary Storage - Main Memory, short term memory
access time – 40-80 ns

Secondary Storage - Disks, medium term storage
access time - 5 ms

Tertiary Storage - Tapes, long term storage (archival storage)
access time - minutes to hours

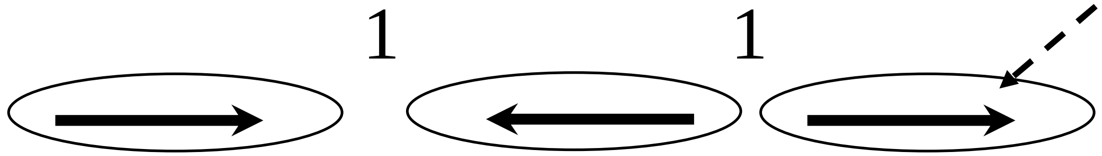
Writeable DVD Roms Tertiary Storage?

Solid state disks?

Magnetic Storage

Smaller region, the greater the disk capacity

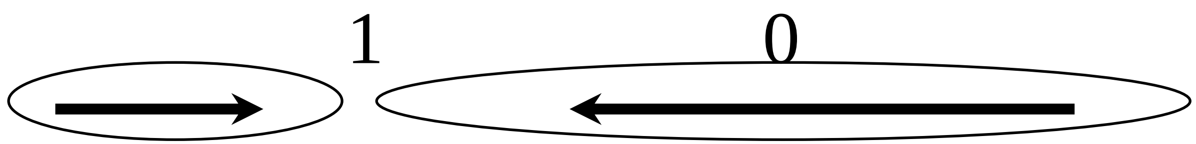
Small surface region of material is magnetised



The direction is not used.

Information recorded by change of magnetisation direction

change - records a ‘1’, no change records a ‘0’



Need a mechanism to recognise that no change has occurred
some form of clocking system is needed

The ability to create very small magnetisation regions that can be reliably written, read and maintained for long periods is a major achievement of modern technology and has been achieved through many years of high quality research and development.

Basic of storage

Information is stored in blocks with gaps between blocks

gaps allow beginning and end of block to be located

a block is the minimum amount to read or write in one go

blocks reduces amount to be read or written
reducing storage space required in main memory

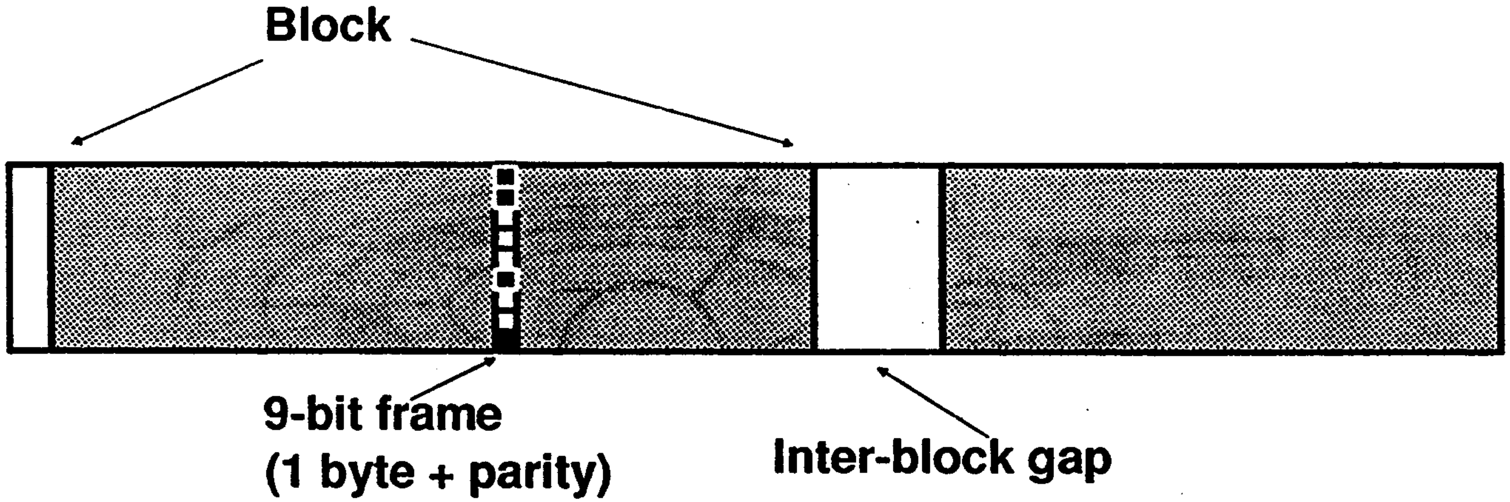
Blocks allow information to be processed in sensible quantities

read_block **process** *read_block* **process**

Magnetic Tape

Old style tapes - seen in 60s James Bond Movies

12inch diameter, 2400feet, 6250bits/inch
9 tracks of storage - 8 data tracks, 1 parity track
Theoretical capacity - 180Mbytes, reduced by inter-block gaps
can stop between each block: very expensive mechanics.



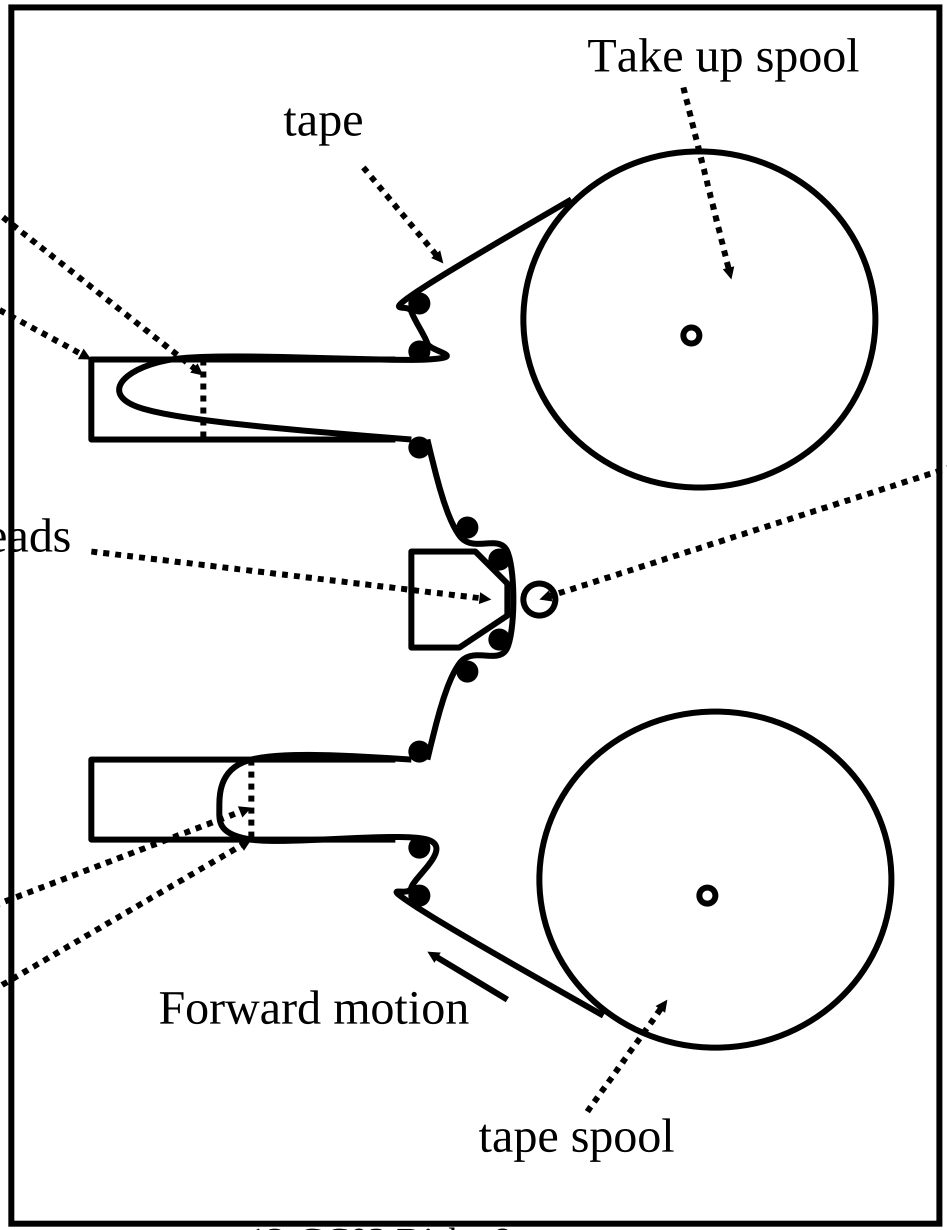
Odd parity - always 1 magnetisation change/strip of data - self clocking

[Parity: count number of 1s in data: even no. of 1s - even parity ; odd no of 1s - odd parity.
Use extra bit to force odd parity over 9 bits: if byte is even parity -9th bit is 1.]]

Tape System as seen in old movies (1960s-70s)

Vacuum tubes allow rapid stopping and starting of tape by capstan

9-Track
Tape Drive
12” tapes



Take up spool takes up slack when tape in top vacuum tube breaks light beam: it is
Just about to do this.

Capstan wheel – accelerates very quickly, drags tape past heads!

Supply spool unwinds more tape, when tape in bottom vacuum tube no longer breaks light beam – tape sucked into vacuum tube.

Very expensive!

Modern Streamer Tapes

derived from video-tape recording systems and DAT (Digital-Audio-Tape Systems).

Cheap mechanics: poor braking system,
cannot immediately stop after block read.
best at reading blocks in continuous fashion

start up, get up to speed, start reading, stop reading, slow down
Reverse several blocks if want to restart reading at next block!!

Example: DDS based on DAT

uses cassette -1 cm x 6 cm x 8 cm,
9-tracks, 9600 bits per inch, tape speed 10-200 inches per second

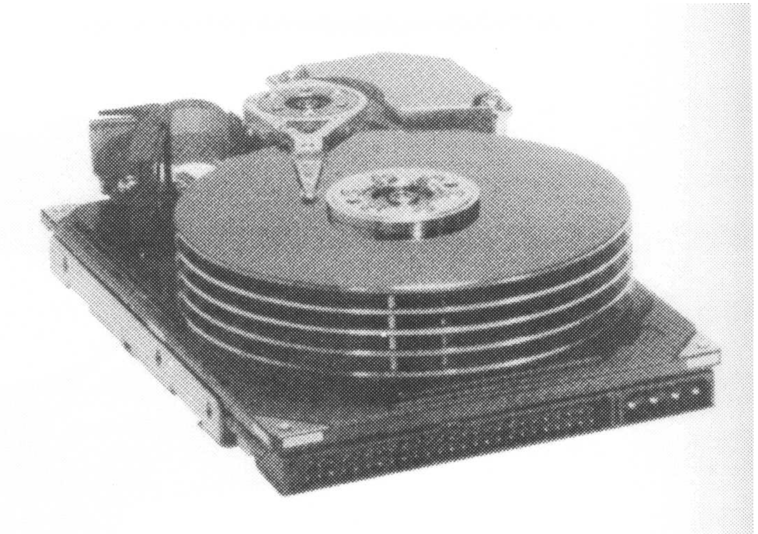
90 m cassette: 2 Gbytes
tape drive - 4 cm x 10 cm x 150 cm

several minutes to find file - then read quickly.

Hard Disk

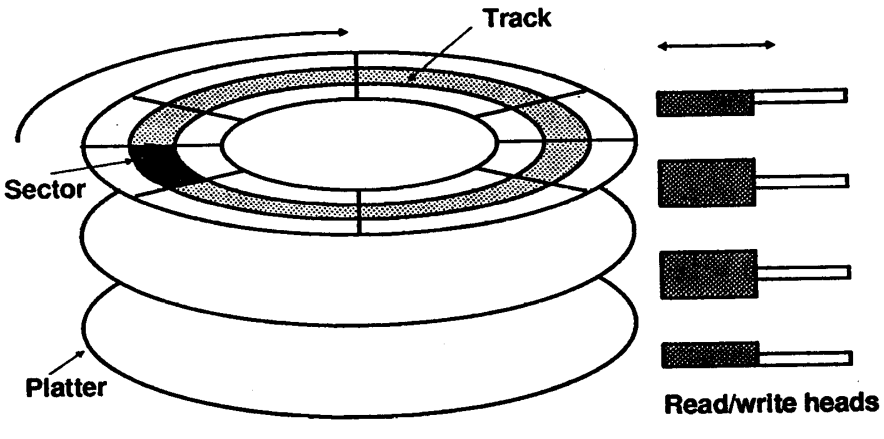
multiple disks or platters
One recording head per surface

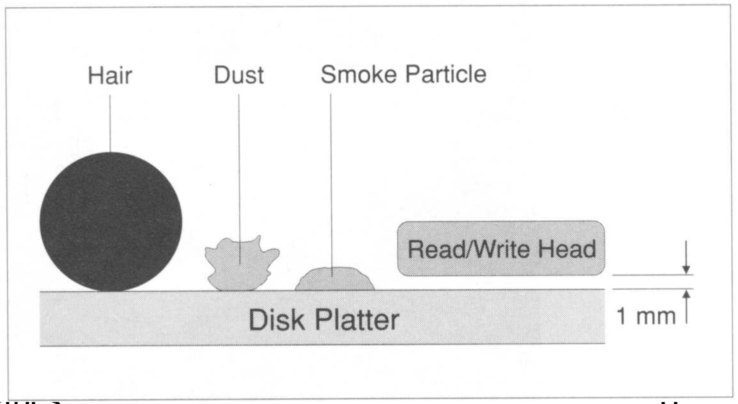
Information storage
- serial bit stream in circular track
  - (**not** a spiral track)
- Multiple tracks
- **blocks** are called **sectors**



heads move in and out as one

Disks are sealed from dust
since heads float close to surface





Floppy Disk

Original 5inch and 8.5 inch floppies
replaced by semi-rigid 3.25inch “floppy”

Double sided storage

head usually rests on surface

low speed, low transfer rate

heads are moved one track at time - very slow
(hard disks do move to track in one movement)

Specifications of some disk types

Size	**5.25" floppy**	**3.5" floppy**	**Hard disk (1994)**	**Sata Hard disk (2010)**
Tracks	40	80	2736	65532
Sectors/track	9	18	40-60	1024 (average)
Sector size	512	512	1024	4Kbytes[1]
Surfaces	2	2	21	4
Capacity (formatted)	360Kbytes	1.44Mbytes	3Gbytes	1TBytes (1000GBytes)
RPM	300	300	5400	7200
Transfer rate	250 KBits/s	500 KBits/s	80 MBits/s	300MBits/s

Note 1: data presented to software as 512 sectors – 4Kbytes stored internally

1st hard drive I had was 20MBytes – late 1980s?

Toshiba 0.85" Hard Disk Drive (HDD)

2GBytes and 4GBytes

Sales blurb:

Only a quarter the size of a 1.8-inch hard disk drive and about the size of a postage stamp, the 0.85-inch HDD will boost the functionality of a new generation of products, including mobile phones, digital audio players, PDAs, digital still cameras, camcorders and more

Date ~2009.

Access times to disks

Problem - mechanical system -
may have to move head to track,
may have to wait for disk to rotate

Rotational latency - average time to wait for sector to be accessed to come to head.
- 1/2 time for 1 rotation

Reading several sectors at a time - wait only for 1st sector to arrive

Track Seek time: time to move head to required track

If next track is selected at random on disk
then average no of tracks to move is 1/3 total no. of tracks

File systems try to keep sectors of file on same or nearby *cylinder*.
Cylinder: made up of tracks on different disk platters with same track number
do not need to move heads to reach next sector if on current cylinder
remember all heads move together

Accessing Disk Sectors

•Head seek time (average random track *seek* is 1/3 no of tracks)
hard disk average is 5ms (year 2001),
floppy average 240 ms (floppies moves in single steps at 3ms/track)

•average *rotational latency* (1/2 rotation period)
hard disk 5.55 ms (5400 rpm)
floppy disk 83ms (360 rpm)

Average access is to access a single sector randomly placed

Type	**Av Rot Latency**	**Av Track Seek**	**Max Track Seek**	**Head Settle**	**Startup**	**Av Access**	**Time to read 1 sector**
Floppy	**83**ms	**76** ms	228 ms	**13** ms	165 ms	**172 ms**	**22ms**
Hard disk 1994 – 3Gb	**5.55** ms	**11.5** ms	23.5 ms			**16.65 ms**	**0.1ms**
Hard disk 2001: 36.4Gb	**3** ms	**5** ms	11 ms			**8 ms**	**0.005-0.01ms**
Sata disk 2010: 1Tb	**4.16ms**	**<8.5 ms**				**12ms**	**0.008-0.016ms**

Floppy disks heads are lifted off disk surface after an access and before movement.
Head settle time is time to lower head to floppy surface: hard disk heads do not touch surface and do not lift
Time to read sector is just time for 1 rotation divided by number of sectors/track.
Hard disks have a landing area on one side of the tracks where heads will land when disk spins down.

Disk Interface Chip

This reduces the complexity of access to a disk system.
It has a number of internal registers for

- setting up a disk transfer
- reading data from or writing data to the disk
- monitoring disk accesses
- formating the disk surface

To make an access the following has to be programmed into the interface registers:

start track, start sector number of sectors to be read/written
address of memory array to place data read from disk
or to read data to write to disk

Disks also now have a Logical Block Access (LBA) where sectors are numbered from 1-upwards to stop having to deal with tracks sectors etc.

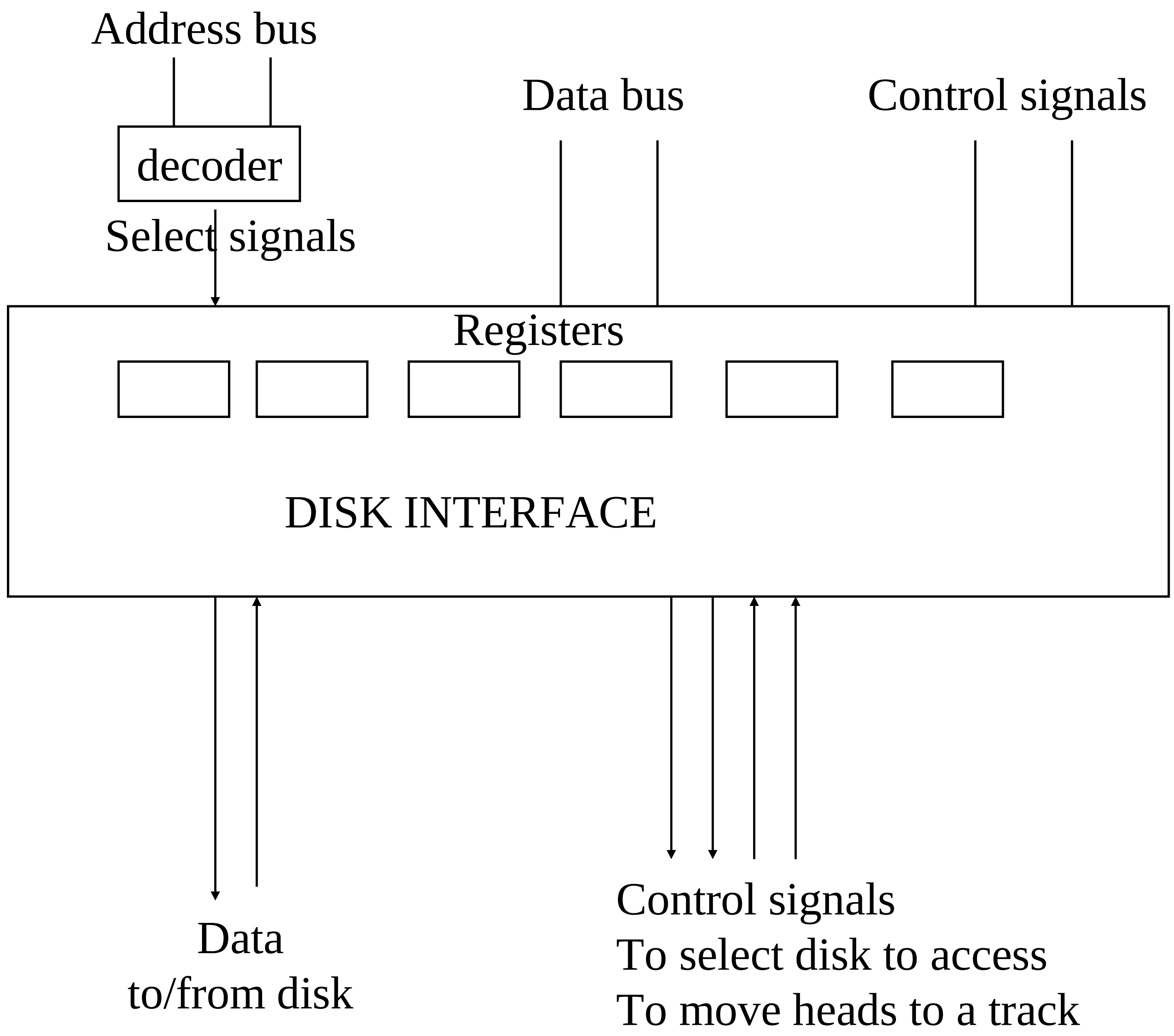
[Disk interface usually performs Direct Memory Access (DMA) to transfer data between memory and interface.]

On completion of a transfer an interrupt can be generated to the CPU.

Information of tracks and sector to read when accessing a file are maintained by a file system. This information is usually saved on the disk as well.

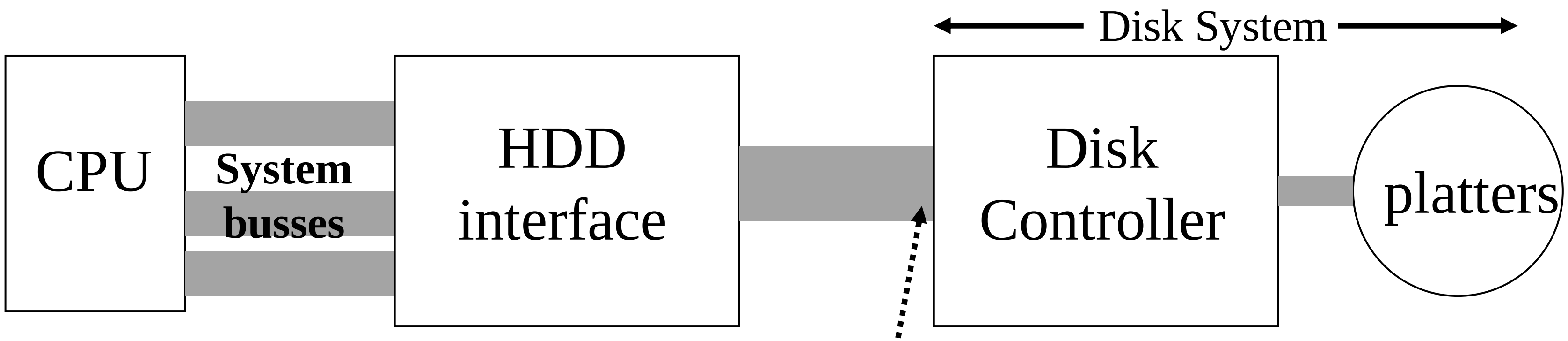
Old Style Disk Interface

(Early 1980s)



Modern HDD interfaces – disk controller in disk box

IDE interface / also called ATA interface or (parallel)ATA



Cable to disk carries data in parallel

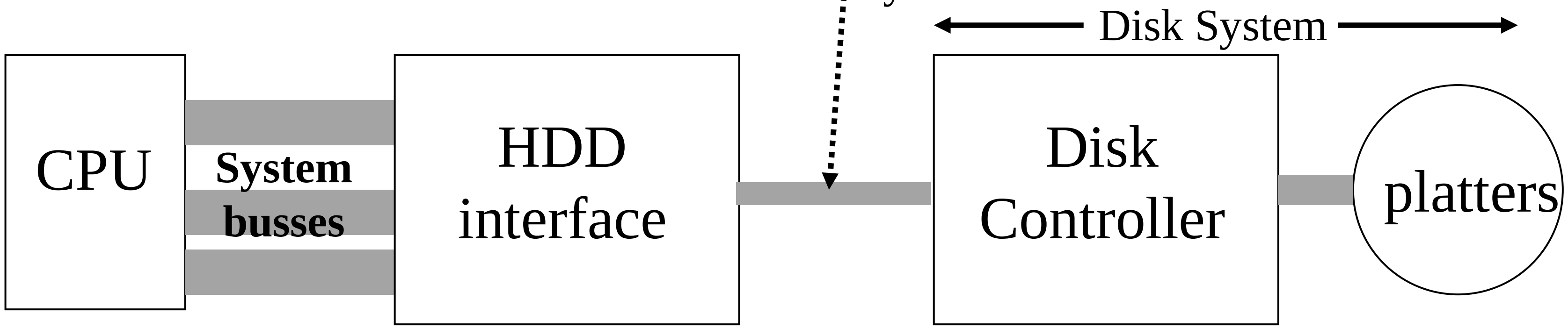
Standards: ATA-02, ATS-03…..ATA-06

Different Speeds: up to 100MBytes/sec

Modern HDD interfaces – disk controller in disk box

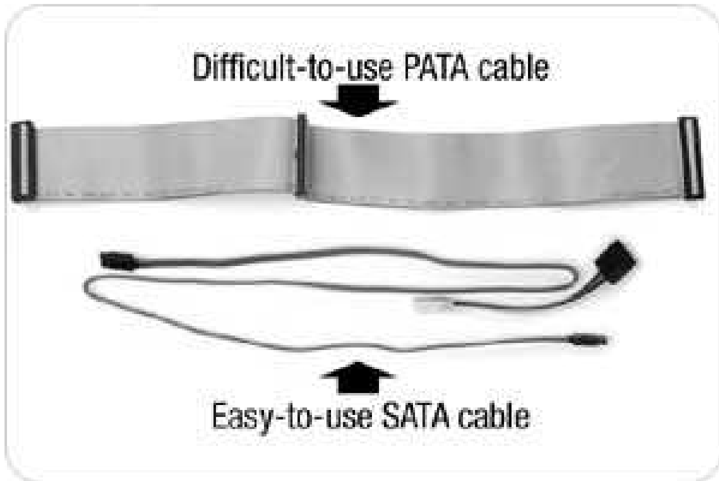
SATA interface or Serial ATA

Cable to disk carries data in serial
133 Mbytes/sec – much smaller cable!



Gives better performance:
no cross-talk or race condition between signals as is case with parallel

Easier cabling – less space on motherboard required for connector



Mapping files on to disk sectors: Disc-block Allocation

Simple implementation from MS-DOS – very poor for large disks

“File allocation table” (FAT) kept on disk
FAT starts at known place: e.g. track 0 block 1

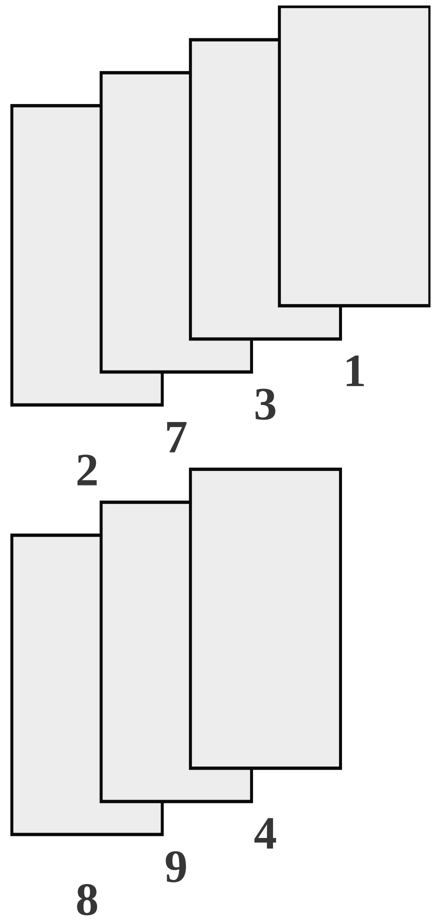
FAT read into memory when disk is in use
OK until disks become big

Group disks sectors into “block”
FAT relates to clusters not blocks
Wasteful for small files
[with big disks blocks are grouped in “clusters”]

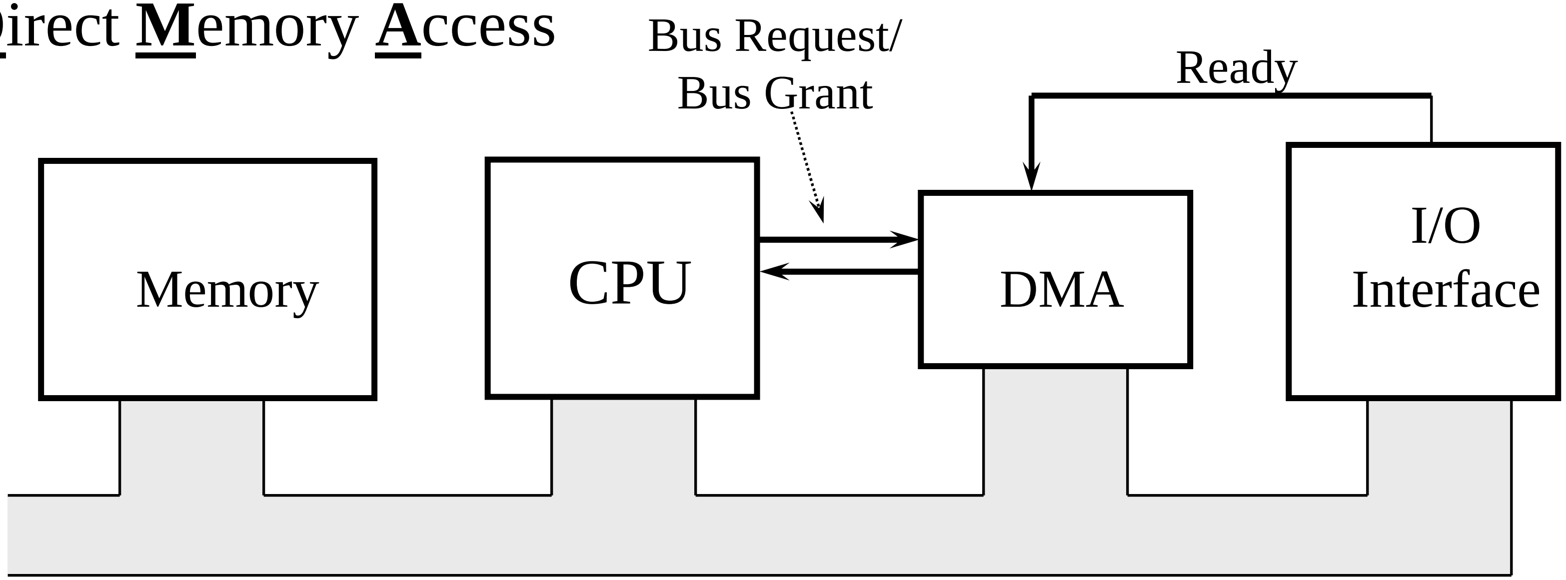
1st block of a file is held with file name in directory file,e.g block 2.
Rest of blocks of file are recorded vi FAT.

0	
1	eof
2	7
3	1
4	eof
5	
6	
7	3
8	9
9	4

File Access Table



<u>D</u>irect <u>M</u>emory <u>A</u>ccess



System Organisation with a Single DMA Device

DMA device:
- takes over system busses
- **does transfers between Memory and I/O interface**
- releases busses
- **DMA does read & write cycles exactly like CPU.**

Advantage: DMA device doesn't execute instructions: its operations are built into hardware: therefore low overhead and very fast.

How does DMA device know what to do?

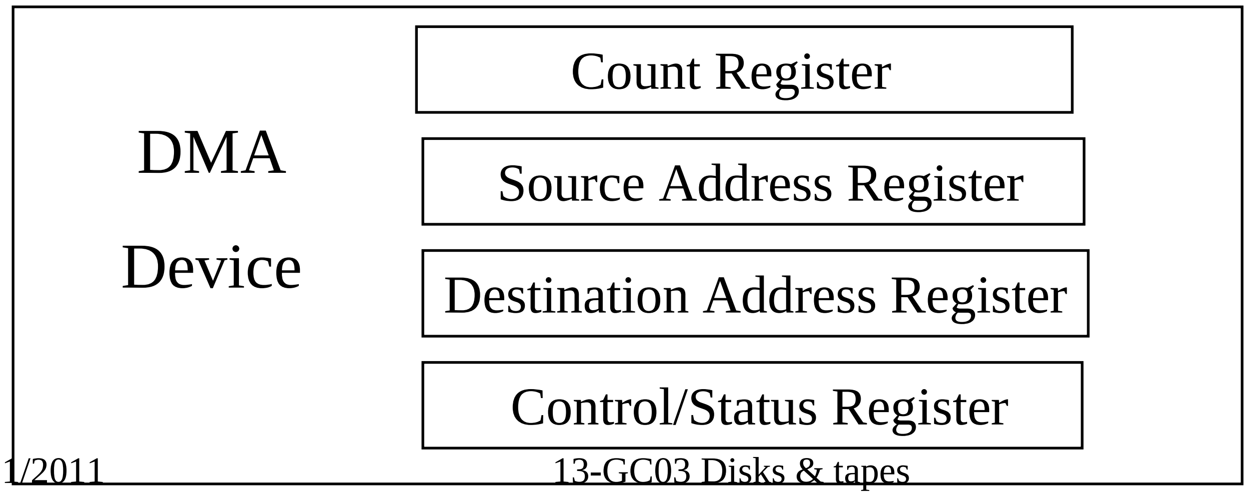
DMA device is programmed by CPU with information on a set of transfers:

destination address of transfer, source address of transfer

amount of data to transfer

how addresses change after each transfer: memory addresses are incremented,
I/O addresses are held constant.

Information written into registers within DMA device like writing to an I/O interface or to a memory location. Each register has an unique address.



DMA take over system busses by exchange of signals with CPU

Bus Arbitration

- DMA sends *Bus Request* signal to CPU.
- CPU finishes any current use of the busses, i.e. read or write cycle
- CPU *releases* busses and sends *Bus Grant* signal to DMA device
- DMA device releases busses when finished
- CPU has lowest priority to use busses: DMA is more urgent.

DMA is a bus-controller, there may be several in system.

Synchronisation of DMA transfers

The **readiness of the I/O interface** is signalled by a *Ready* signal from the interface to DMA device. *Ready* reflects the state of the interface Status Port.

When the *Ready* signal goes active, the DMA device requests the busses from the CPU. Once the *Bus Grant* from the CPU goes active, it performs transfers until the *Ready* signal goes inactive, when it releases the busses.

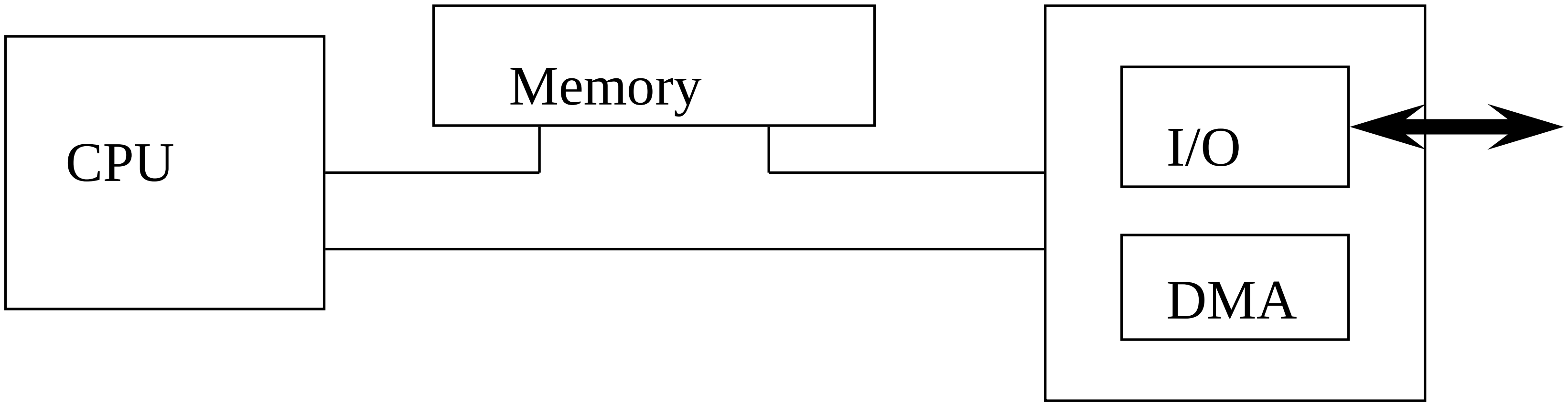
Transfers are of the form:-

Input: Interface $\xrightarrow[\text{Cycle}]{\text{Read}}$ DMA device $\xrightarrow[\text{Cycle}]{\text{Write}}$ Memory

Output: Memory $\xrightarrow[\text{Cycle}]{\text{Read}}$ DMA device $\xrightarrow[\text{Cycle}]{\text{Write}}$ Interface

DMA Device integrated into I/O interface

This is used where transfer rate or expected usage is high, e.g. Hard Disk



The DMA is programmed at the same time as the I/O transfer is set up.

Now only need a single stage transfer:-

I/O interface ⟶ Memory

I/O interface ⟵ Memory

whereas with a separate DMA device, 2 stage transfers are generally needed. This gives a factor of 2 speed-up and a similar reduction in the bus cycles used.

Data Transfer Overview

Stream of Single Data Items:

We might define an I/O stream as a stream of single transfers, if some software activity is required immediately on each data transfer, i.e. processing of the input value or the next output value must be produced.

This form is characteristic of keyboards, mouse input devices, digital signal processing, such as filtering an analogue signal.

Block Transfer of Data Items

In this form of transfer, activity is only required after a block of data items has been transferred. This is a more common transfer activity in general computer systems, e.g. with disks, tapes, ethernet, printers.

There may be long gaps between blocks, but transfer frequency may be high within the block.

Low and Medium Frequency Single Transfers

- **interrupts are perfect** for this. Action is taken at the moment it is required.
- **polling is terrible**, lots of CPU cycles wasted in the polling loop.
- **DMA is a waste**. After each single transfer, the DMA device needs to generate an interrupt to get software to be run to take whatever action is required after each transfer: might as well not use DMA device at all and just use interrupt from I/O device.

High Frequency Single Transfers

- **interrupts are no use** here as it can't keep up, because of the high overhead of switching into and out of the ISR: all that saving and restoring registers.
- **DMA is no use** either, since need interrupt to get processing performed on each transfer, so would have to poll DMA device.
- **polling is the only solution**, and this is when it functions well: only a small number of times around the polling loop.
- This sort of operation is typical of Digital Signal Processing, not of general purpose computing, e.g. home computers.

Block Transfer

Interrupts are not good for this, as the high data frequency within the block means that interrupt cannot keep up: for low rates the overhead is too high.

Polling on its own is no good because of the possible long gaps between blocks, even if the data rate within blocks is high.

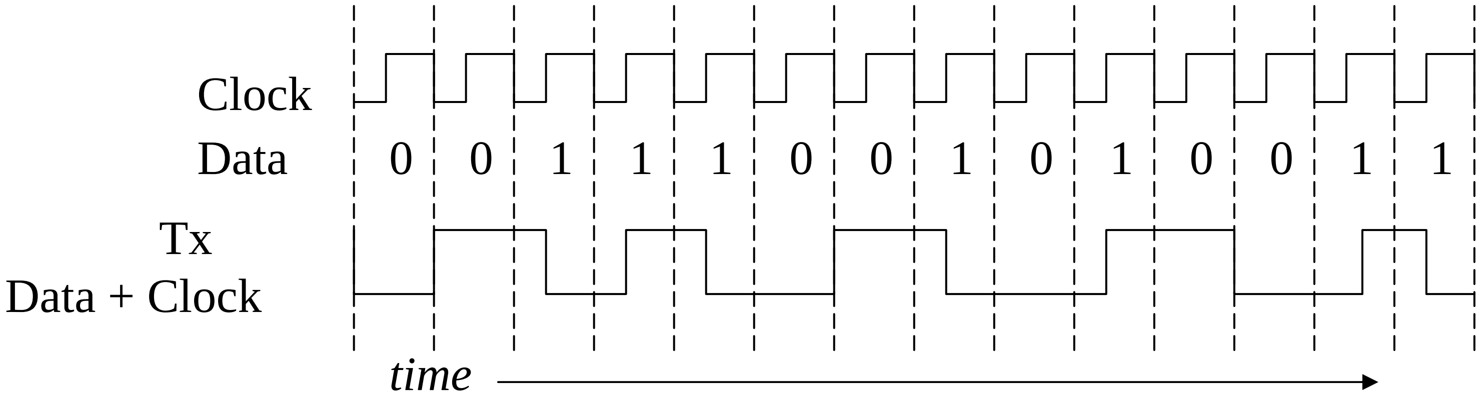
Using an **interrupt** to detect the first transfer of a block and **then using polling** on the remaining transfers is alright at high frequencies, but all the CPU cycles are used up during the block transfer, and the instruction fetching and execution is a major overhead. Only really usable if the block frequency is very low.

DMA is perfect for this form of transfer. It does just the necessary bus transfers to move the data from one place to another with no instruction fetch or execution overhead, other than the initial programming of the DMA interface. It has smallest overhead per transfer and **uses bus cycles efficiently**.

For interest only

modified frequency modulation – used on floppy disks

Hard disks use a group encoding scheme



A ‘1’ bit is encoded by a transition (1→0 or 0→1) in the middle of the clock period

A ‘0’ bit is encoded by a transition at the start of a clock period except that:a ‘0’ bit following a ‘1’ bit is not encoded but is deduced by Rx.

Combined clock and data, but still 1 bit of data per clock period – very clever