File System and I/O

Operating System Lab-in-a-Box

Previous

- Virtual Memory

Current Module

- Extended data Storage and File System
- File structure and directory structure
- File attributes
- Open Files
- Allocation methods
- Accessing a file, disk
- Disk and I/O scheduling
- I/O, Direct control
- Interrupt and
- DMA

The Extended Data Storage

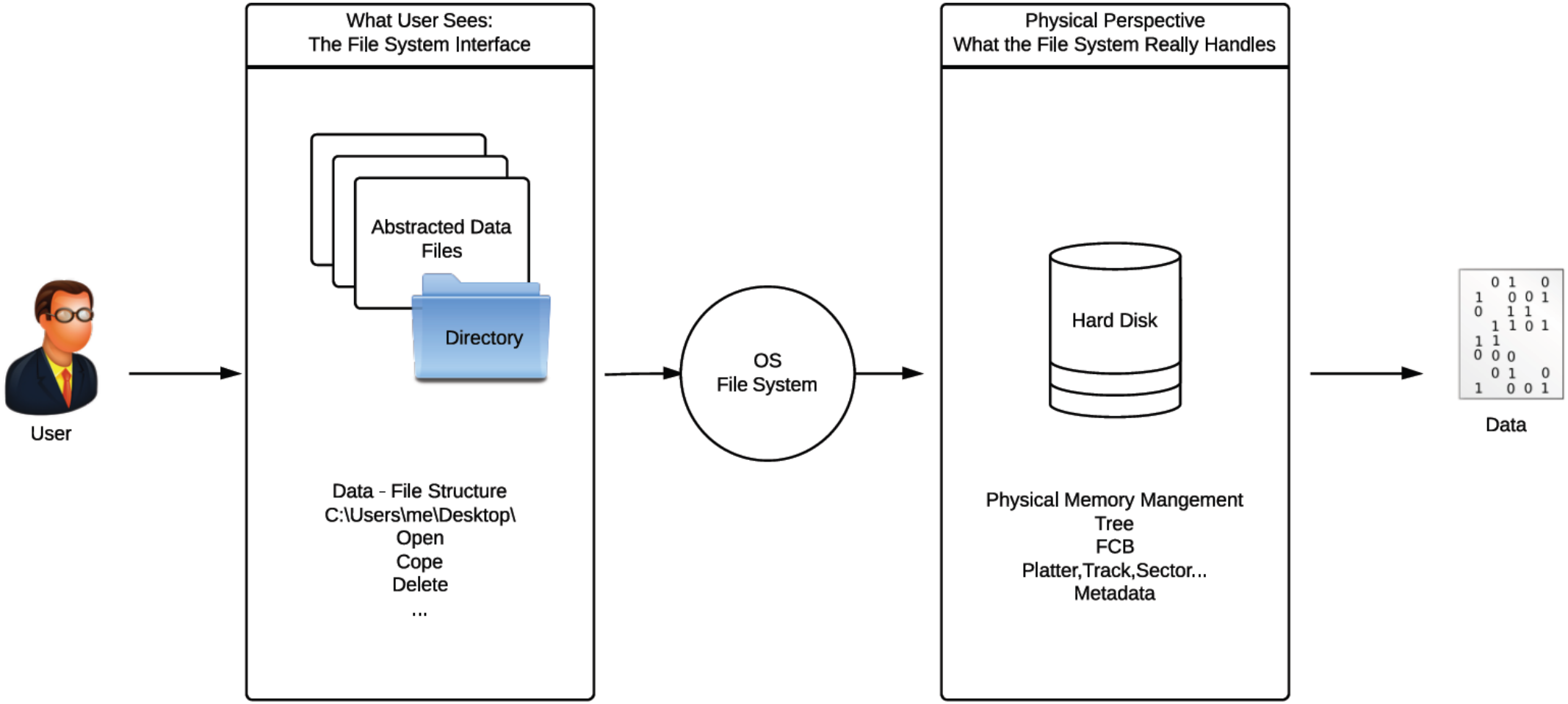
- File System and I/O can be both considered as extensions to the memory hierarchy
- In a users' perspective, the most visible components

- File - a chunk of relevant data, organized according to either users or applications, usually contiguous in logical address space.
- OS needs to manage files and provides interface to users so that they can read/write…
- Luxury to embedded applications, but nowadays not uncommon!

- Input, output devices – peripherals. In embedded system, could be on-board or external
- OS has to unify data representation
- "Everything is a file."

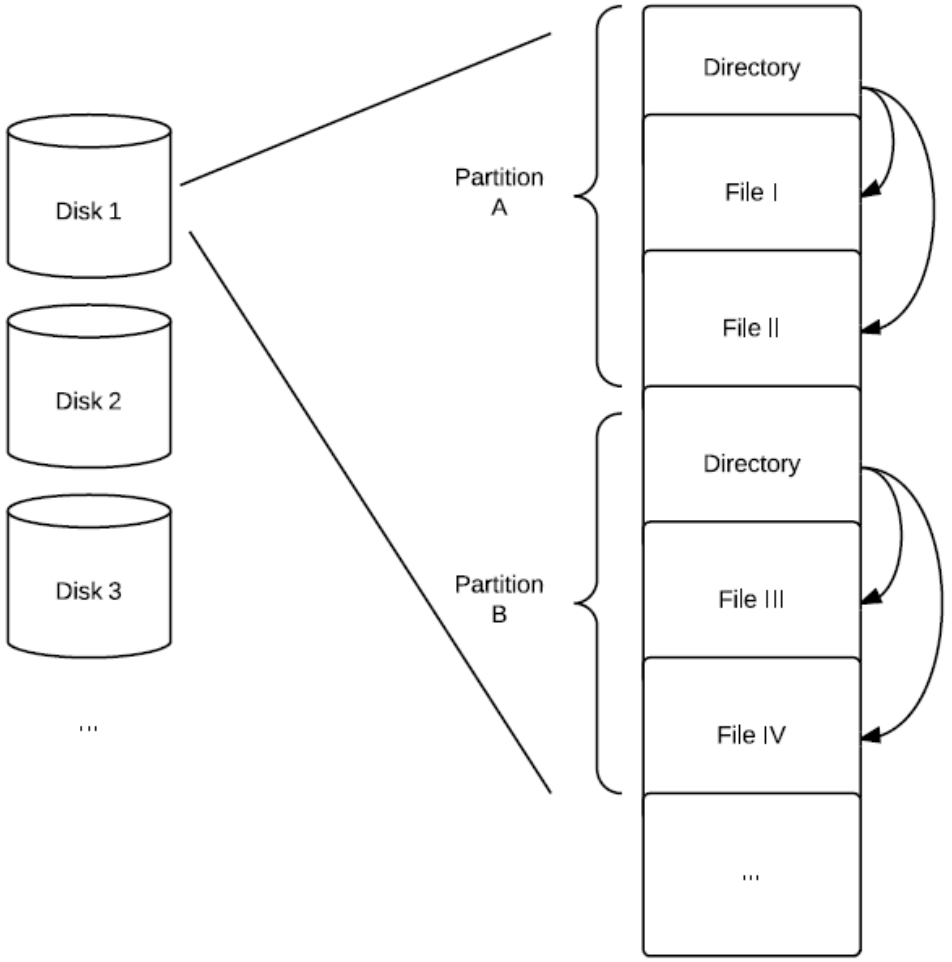
File System

- The file system is all about how to organize and present data so that user can have easy access.



File Structure

- Disk
- Partition (C drive)
  - Multiple partitions on one disk
  - One partition on multiple disks
  - A logical(virtual) hard disk
- Directory
  - Data structures maintained by the OS to keep track of files.
  - Also keeps file attributes and metadata
  - Both directories and files are stored on the disk
- File



Directory Structure

- Single-Level Directory
  - Ancient structure, everything in the same directory and memory loads the directory to a fixed location to support file operations.
  - No same name files allowed
  - No path
- Two-Level Directory
  - Main file directory and user file directory
  - To individual user, equivalent to single-level directory
- Tree-Structured Directory
  - The concept of (relative, absolute) path and working directory

File

- Counterpart of a real document in a computer, thus the name (similarly for directory)
- In brief, a collection of related data (could also be program codes) represented as a block in a contiguous logical address space plus associative information
- Can be further divided into records and fields…

- Two types:
  - No structure: streams of bytes
    - Easy to handle and modified
    - But hard to search for specific unit
    - Windows, Unix
  - Structural: either records or formatted documents; could be variable length
    - Clear logical organization, easy to modified, search and manage on the record level
    - Complicated
    - Not mutually exclusive, OS can support both

File Attributes

- The “associative information”
  - Name
  - ID
  - Size
  - Type
  - Creation/Modified Date
  - Permission/Protection – access control, which group of user can perform what operations?
  - Location – where is it based in the hard disk?
  - Directory information
- File Control Block (FCB)

Open Files

- A file can usually be opened by multiple processes, unless otherwise specified by open file locking
- Two file tables
  - Public file table maintained by OS and is independent of processes
    - Location on the disk, access time, size and so on
    - File open counter, if can be reopened, this is to count the number of times the file is opened
    - Access rights
  - Process dependent table, unique to the process and keep tracks of all files that is currently accessed by the process.
    - File pointer, points to recently read/write location
    - Pointer to the entry in the public file table

Allocation Methods

- Contiguous Allocation
  - Simple to implement as only the start and size of the file are needed
  - Support Random Access
  - But waste space and cannot deal with variable size
  - How to delete? Insert? – Frequently modified files will have troubles
- Linked Allocation
  - Linked list, current block will points to the next block
  - Highly flexible in terms of saving memory space
  - However, can only access in the right sequence
- Indexed Allocation (Could also be multilevel)
  - Flexible and dynamic, support variable size file and random access
  - Memory overhead and have to access disk at least two times
  - Cache some of the index table to improve performance
- Bitmap is also a generic method widely used to manage free spaces

Accessing A File

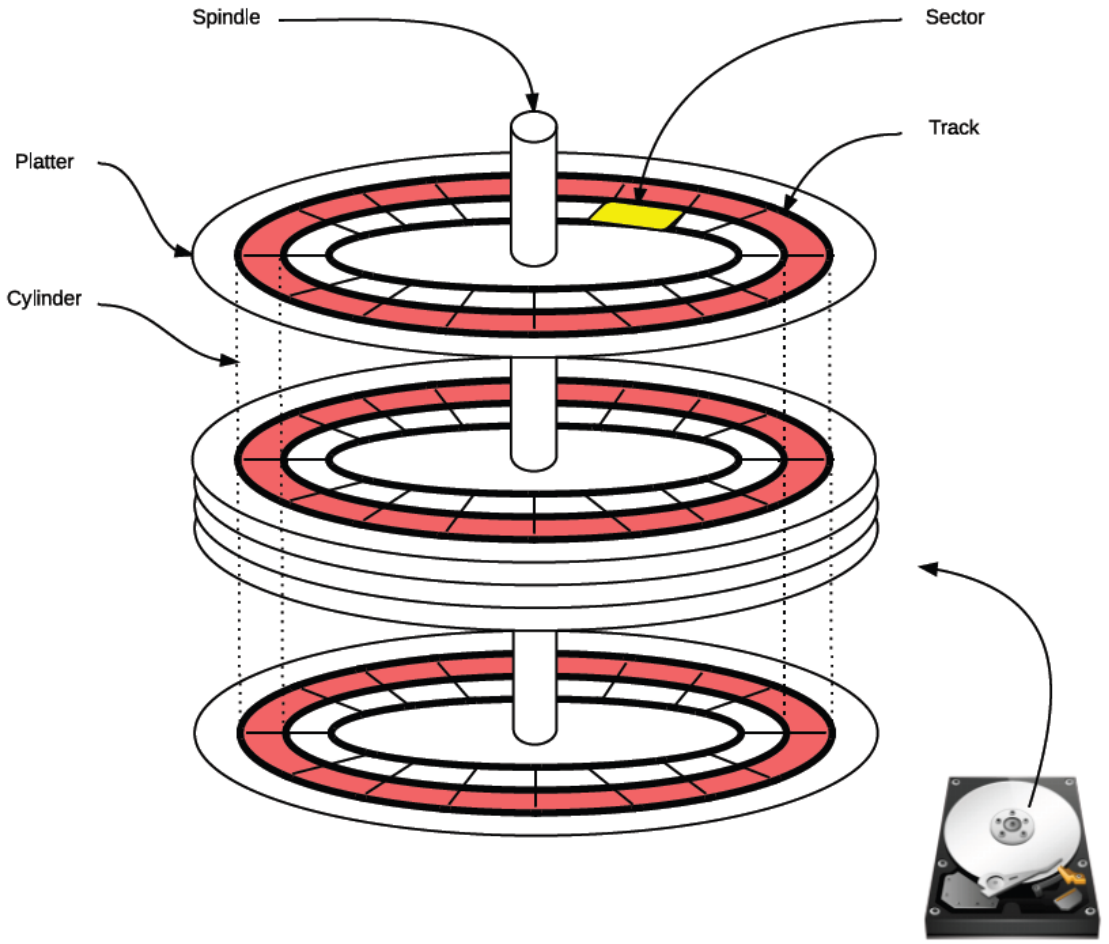
- Modify, append or search (inside the file) - read/write
- Access methods
  - Sequential – heritage of tape file system
    - Read/write next
    - Reset
    - Skip forward/backward
  - Direct/Relative/Random – more hard disk
    - Read/Write *n*
    - Index – fast for search and database

Disk

- Logical file will be eventually mapped to a section in the disk
- Virtual Cylinder-head-sector (CHS)

- To find the target sector
  - Mechanically move the head to the correct cylinder
  - Rotate the spindle and wait until the correct sector

- Disk Scheduling
  - To reduce latency
  - Assume random access



Disk Scheduling Or I/O Scheduling

- We have already introduced some scheduling techniques, possible to use them here
- Random Scheduling (RSS), FIFO, LIFO, shortest seek first

- Elevator algorithm: or SCAN, move the head in only one direction and stop if any request for the current cylinder (utilize locality?)
- Circular Scan: the circular variant of the elevator algorithm
- What if request of the current cylinder keep coming? – stiff arm (starvation)

- N-Step-Scan: queues are consists of several sub-queues (elevator algorithm) of N request. New request won't affect the head if the queue is already long enough
- FSCAN: two queues, during the current scan cycle, all new requests are pushed to the other queue.

I/O

- A wide range of devices. Unity is nowhere from achievable
- An example – Mouse
  - How many buttons? Two plus a scroll wheel? (Check Razer mice)
  - Communication protocols?
    - PS/2
    - USB
    - Wireless
    - …
  - Drivers?
- There are four major approaches to handle I/O devices:
  - Polled (direct control by the CPU)
  - Interrupt
  - DMA
  - I/O channel

Direct Control

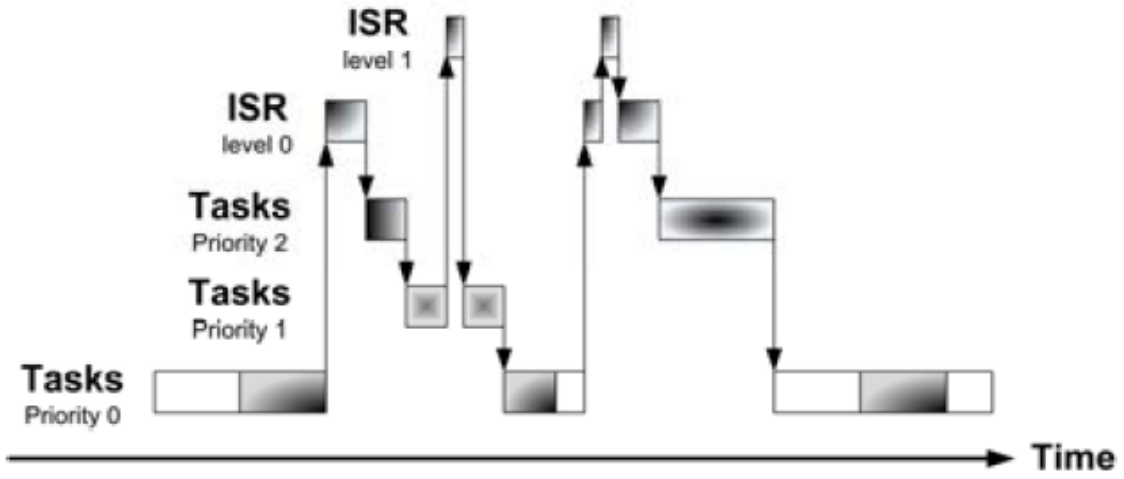
- On-board peripherals – write/read configuration register is a standard practice
- Polling?

```
void main(){
        while(check_IO_status ()){
                ……
        }
}
```

  - Busy waiting until I/O finishes
  - Simple but a wasteful of CPU

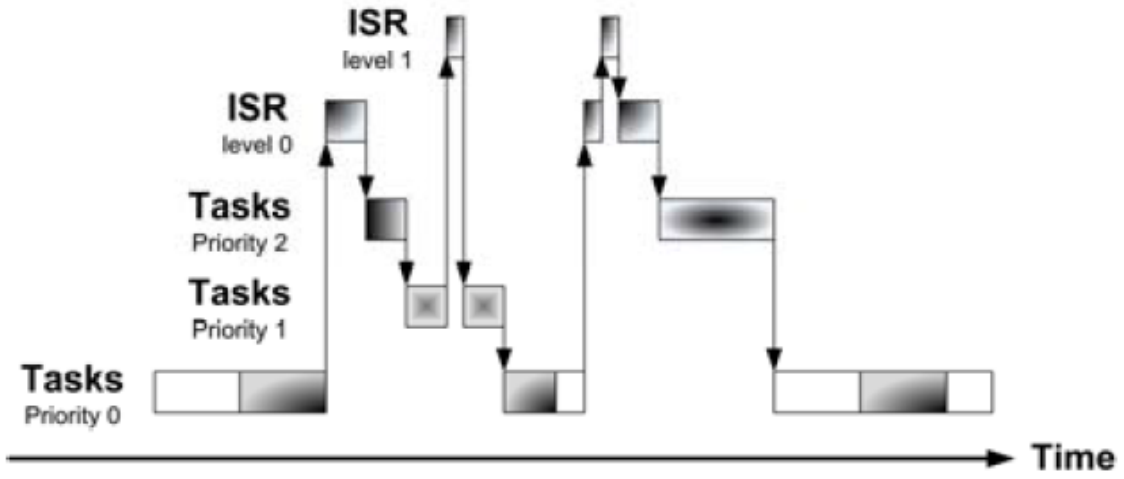
Interrupt

- I/O will notify the CPU when its job is done. Then the interrupt service routine (ISR, or handler), a piece of (usually short) code is triggered to deal with the I/O.
- Fundamental mechanism of microcontrollers
  - Provides efficient event-based processing rather than polling
  - Provides quick response to events regardless* of program state, complexity, location
  - Allows many multithreaded embedded systems to be responsive without an operating system (specifically task scheduler)
- Priorities
- Nested interrupts



Interrupt

- Interrupt improves the performance of the CPU in many ways compared to polling. However, the overhead of interrupt may scale up as the number of peripherals increases.
- Interruptions could also lead to unpredictable and unstable CPU states. Which is generally undesirable for real-time systems.
- Although RTX supports Cortex-M interrupts, interrupts are usually only used to set event flags



Interrupt Example For Discovery

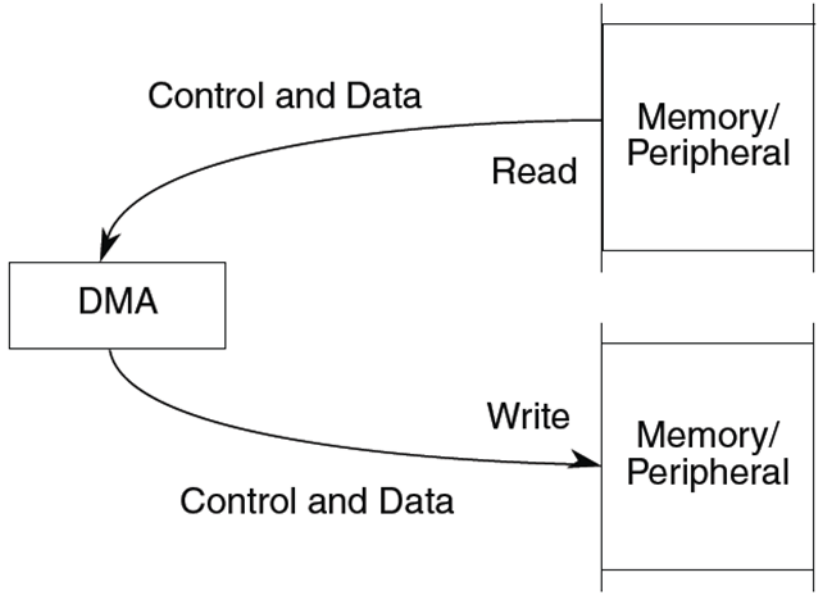
```
void Init_Switch(void){
                    RCC->AHB1ENR|=RCC_AHB1ENR_GPIOAEN;
                    GPIOA->PUPDR|=GPIO_PUPDR_PUPDR3_0;
                    //Connect the portA pin3 to external interrupt line3
                    SYSCFG>EXTICR[0]&=SYSCFG_EXTICR1_EXTI3_PA;
                    //Interrupt Mask
                    EXTI->IMR |= (1<<3);
                    //Falling trigger selection
                    EXTI->FTSR|= (1<<3);

                    __enable_irq();
                    NVIC_SetPriority(EXTI3_IRQn,0);
                    NVIC_ClearPendingIRQ(EXTI3_IRQn);
                    NVIC_EnableIRQ(EXTI3_IRQn);
}

void EXTI3_IRQHandler(void){
          //Clear the EXTI pending bits
          EXTI->PR|=(1<<3);
          NVIC_ClearPendingIRQ(EXTI3_IRQn);
          //…Handle the IO here
}
```

DMA

- Direct Memory Access - Special hardware to read data from a source and write it to a destination
- All the CPU needs to do is configure it correctly
- Various configurable options
  - Number of data items to copy
  - Source and destination addresses can be fixed or change (e.g. increment, decrement)
  - Size of data item
  - When transfer starts
- Can also work with interrupts, e.g. interrupt CPU at the end of a transfer
- Exempt the CPU from busy-waiting and frequent interruptions.



I/O Channels

- A separate processor specialised for I/O control
- May have own specific instructions
- Share memory with CPU through bus (steal cycles)
- Individual programs on memory and could be dynamically generated

- Further minimizes CPU intervention

Next

- RTX Case Study