## Introduction

1. **What is computer architecture?**

* Those attributes of a system visible to a programmer; have a direct impact on the logical execution of a program.

1. **What does the instruction set architecture (ISA) define?**

* Defines instruction formats, instruction upcodes, registers, instruction and data memory; the effect of executed instructions on the registers and memory; and an algorithm for controlling instruction execution.

1. **What does computer organization refer to?**

* Refers to how features are implemented; the operational units and their interconnections that realize the architectural specifications.

1. **What does the organizational attributes include?**

* Include those hardware details transparent to the programmer, such as control signals; interfaces between the computer and peripherals; and the memory technology used.

1. **What do structure and function of a computer system mean?**

* Structure: the way in which components relate to each other.

Function: operation of individual components as a part of the structure.

1. **Which are the four basic functions that the computer can perform?**

* Data processing; Data storage; Data movement; Control

## Number Systems

1. **Describe the decimal system.**

* System based on decimal digits to represent numbers. Base for decimal system is 10 – each digit in the number is multiplied by 10 raised to a power corresponding to that digits position.

1. **What is base (radix)?**

* Base or radix is number if unique digits (including zero), used to represent numbers.

1. **Present the digit significance.**

* Most significant digit: the leftmost digit, since it carries the highest value.

Least significant digit: the rightmost digit; lowest value

1. **What is positional number system?**

* Each number is represented by a string of digits in which each digit position *i* has an associated weight *ri*, where r is the radix, or base, of the number system.

General form: (…a3a2a1a0.a-1a-2a-3…)r – with radix r

The value of any digit ai is an integer in the range 0<=ai<r

The dt between a0 and a-1 is called the radix point.

1. **What are the advantages of the hexadecimal notation?**

* It is more compact than binary notation; in most computers, binary data occupy some multiple of 4 bits, and hence some multiple of a single hexadecimal digit; it is extremely easy to convert between binary and hexadecimal notation.

1. **What is floating point notation and what is it used for?**

* Method of representing very large or very small numbers in an expression of fixed size. Used to represent noninteger fractional numbers.

## Computer Arithmetic’s

1. **What is the arithmetic and logic unit (ALU)?**

* It is part of the computer that actually performs arithmetic and logical operations on data. The other elements of the computer system are there mainly to bring data into the ALU for it to process on them and to take the result back out.

1. **What does the ALU operate over?**

* The ALU operates over integers; it deals with real (floating point) numbers; as a separate unit to deal with real numbers (FPU).

1. **How can integers be represented?**

* 1.Sign-magnitude

2.Two’s complement

Others: 1. Biased (offset binary excess code)

2. One’s complement

3. Base -2

1. **Describe Sign-magnitude representation.**

* The leftmost bit represents the bit of the sign (0 means positive and 1 means negative).
* Range extension: accomplished by moving the sign bit to the new leftmost position and fill in with zeros.
* Problems: Addition and substitution require a consideration of both the signs of the numbers and their relative magnitudes to carry out the required operation. There are two representations of 0.

1. **Describe ones complement**

* Negative notation of an integer is obtained by inverting all digits; 0 becomes 1 and vice versa.
* Problems: 2 zeros; addition and subtraction require to add an “carry back” (or end-around carry) to the result.

1. **Describe the twos complement.**

* To get the negative notation of an integer, we write out the number in binary then invert the digits (Ones complement) and add one to the result. It is same for backward process.
* Characteristics: 1. Range -2n-1 through 2n-1-1

2. Number of representations of 0 – one

3. Negation - Take the Boolean complement of each bit of the corresponding positive number, then add 1 to the resulting bit pattern viewed as an unsigned integer.

4. Expansion of bit length - Add additional bit positions to the left and fill in with the value

of the original sign bit.

5. Overflow rule – If two numbers with the same sign are added, then overflow occurs if and only if the result has the opposite sign.

6. Subtraction rule – To subtract B from A, take the twos complement of B and add it to A.

* Range extension (sign extension): Move the sign bit to the new leftmost position and fill in with copies of the sign bit.

1. **Describe biased representation**

* Signed number n is represented by the bit pattern corresponding to the unsigned number n+k with k being the biased value or offset.
* Decimal to Biased-127: Add N to decimal value. Convert as if unsigned.
* Biased-127 to Decimal: Convert to decimal as if unsigned. Subtract N.

1. **Describe Base -2.**

* A binary number system with base -2. The range of numbers that can be represented is asymmetric. If the word has an even number of bits, the magnitude of the largest negative number that can be represented is twice as large as the largest positive number that can be represented.

1. **Which are the special cases in negation in integer arithmetic?**

* Negation of 0. The carry out (overflow) byte is ignored, so -0=0.
* Negation of (-127). We get (-(-127)) = (-127). Here we should monitor the MSB (sign bit), it should change during negation.

1. **How do logical and arithmetic shift work?**

* In multiplication for signed binary numbers arithmetic right shift is used, while for unsigned logical right shift is used. In division for signed binary numbers arithmetic left shift is used, while for unsigned binary numbers logical left shift is used.

1. **What are the solutions for multiplying negative numbers?**

* Solution1: Convert to a positive number if needed. Multiply and if the signs are different, negate the result.
* Solution 2: Booth’s algorithm: preforms fewer additions and subtractions than a more straightforward algorithm.

1. **What is fixed point representation?**

* S-M and TC are sometimes referred to as fixed point. This is because the radix point (binary point) is fixed and assumed to be to the right of the rightmost digit. Possible to represent a range of positive and negative integers centered on or near 0.
* Limitation: Very large numbers cannot be represented nor can very small fractions. The fractional part of the quotient in a division of two large numbers could be lost.

1. **Describe the IEEE Standard 754.**

* Defines most important floating-point representation. Covers both binary and decimal floating-point representations. It was developed to facilitate the portability of programs from one processor to another and to encourage the development of sophisticated numerically oriented programs.

1. **What are signaling and quiet NANS?**

* NANS: symbolic entity encoded in floating point format.
* Signaling NAN: signals are invalid operation exception whenever it appears as an operand.
* Quiet NAN: propagates through almost every arithmetic operation without signaling an exception.

1. **Which are the types of floating point formats?**

* Arithmetic format; Basic format; Interchange format.
* Additional formats: Extended precision format; Extendable precision format

1. **What does precision consideration contain?**

* Round to nearest; Round toward +ꝏ; Round toward -ꝏ; Round toward 0.

1. **What are subnormal numbers?**

* Included in IEEE 754 to handle exponent underflow. When the exponent of the result becomes too small, the result is subnormalized by right shifting the fraction and incrementing the exponent for each shift until the exponent is within a representable renge.

## Digital logic

**Boolean algebra** – mathematical discipline used to design and analyze the behavior of the digital circuitry in digital computers and other digital systems.

* Convenient tool for: **analysis** – economical way od describing the function of digital circuitry; **design** – given a desired function, Boolean algebra can be applied to develop a simplified implementation of that function.

**Basic logical operations are AND, OR and NOT.**

**Logic gates** – electric circuit that produces an output signal that is simple Boolean operation on its input signals.

**Gate delay** – length of time which starts when the input to a logic gate becomes stable and valid to change, to the time that the output of that logic gate is stable and valid to change.

**Functionally complete sets** – of logical connectives or Boolean operators is one which can be used to express all possible truth tables by combining members of the set into Boolean expression. (AND, OR, NOT; AND, NOT; OR, NOT; NA ND; NOR)

**Combinational circuit** – an interconnected set of gates where output at any time is a function only of the input at that time. The appearance of the input is followed almost immediately by the appearance of the output, with only gate delays.

* Can be defined in three ways: **truth table** – for each of the 2n possible combinations of input signals, the binary value of each of the m output signals is listed; **graphical symbols** – the interconnected layout of gates is depicted;  **Boolean equations** – each output signal is expressed as a Boolean function of its input signals.

**Simplifying methods** – with simpler Boolean expression fewer gated will be needed to implement the function. Or we might like to implement the expression with only NAND or NOR implementation.

* **Algebraic simplification**
* **Karnaugh maps (K-maps)**
* **Quine – McCluskey tables**

**Binary coded decimal** (K-maps) – used for incrementing decimal digits. In some cases certain combinations of values of variables never occur, and therefore the corresponding output never occurs. So we use “d” (“don’t care”) conditions. Each d can be treated as 1 or 0, whichever leads to the simplest expression.

**Multiplexer** – device that selects between several analog or digital input signals and forwards the selected input to a single output line.

**Decoder** – circuit that changes a code into a set of signals. Only one output line is asserted (has value 1) at any time. Generally it has n inputs and 2n outputs.

**Demultiplexer** (using a decoder) – connects a single input to one of several outputs. All of the 2n output lines are ANDed with a data input line. Thus, the n inputs act as an address to select a particular output line, and the value on the data input line (0 or 1) is routed to that output line.

**Read only memory** – memory implemented with combinational circuits (“memoryless” circuits because their output depends only on their current input and no history of prior input is retained.) Memory unit that performs only the read operation. Binary information is permanent, created during the fabrication process. Because the outputs are function only of the present inputs, ROM is a combinational circuit.

**Adders** – performs addition of numbers. Carry lookahead – the carry values could be determined in advance.

**Sequential circuit** – current output depends not only on the current input, but also on the past history of inputs-this is on the current input and the current state of the circuit.

**Flip-Flops** – simplest form of sequential circuit.

* Properties: 1. The flip-flop is bistable device. It exists in one of two states and in the absence of input, remains in that state. Thus, flip-flop can function as 1-bit memory. 2. The flip-flop has two outputs, which are always the complements of each other.

**S-R flip-flop of S-R latch** – has two inputs S (set) and R (reset) and two outputs Q and Q. Consists of two NOR gates connected in feedback arrangement.

* **Clocked S-R flip-flop** – S and R inputs are passed to the NOR gates only during the clock pulse.
* **D flip-flops** – allow just a single input to avoid the condition R=1, S=1.
* **J-K flip-flop** – like the S-R flip-flop is has two inputs, however all possible combinations of input values are valid.

**Use of flip-flops: Registers** – digital circuit within the CPU to store one or more bits of data. (Parallel registers and Shift registers). **Counters** – register whose value is easily incremented by 1 modulo the capacity of the register. After the maximum value is achieved the next increment sets the counter value to 0. (Ripple counter and Synchronous counter).

**Programmable logic devise** – general-purpose chip that can be readily adapted to specific purposes. Used instead of putting more chips on the circuit and adding connections between them since it is cheaper.

**Simple PLD: PLA** – consists of NOT, AND and OR gates. With appropriate connections, arbitrary SOP expressions can be implemented. **PAL** – programmable AND-plane followed by a fixed OR-plane.

**Complex PLD: Filled programmable gate array (FGPA)** – consists of an array of uncommited circuit elements, called logic blocks, and interconnect resources.

* **Logic block** – where the computation of the user’s circuit takes place. **I/O block** – connect I/O pins to the circuitry on the chip. **Interconnect** – signal paths available for establishing connections among I/O and logic blocks.

# Computer Evolution

1. **First Generation**

* ENIAC Eckert and Mauchly (University of Pennsylvania) started 1943 finished 1946 – effect of the WW2. Used until 1955. Decimal, not binary. 30tons. 1400m2.   
  Vacuum tubes – marked the whole 1st generation.   
  **Von** **Neumann** – stored program concept (MM storing programs and data).  
  IAS -1952 (21 instruction set).
* Structure of Von Neumann’s model: 3 main parts:  
  **Main** **Memory**: 4096 storage locations. Can either be 1 binary number – Number word (sign bit number); Instruction word 2x20(8-bit opcode 12-bit address)  
  **CPU**: Arithmetic and Logic unit and Control unit. – Special registers (Program Counter, Memory Buffer, Memory Address, Instruction, Instruction buffer, Accumulator and Multiplier quotient)  
  **I/O equipment**
* Commercial Computers: UNIVAC I, UNIVAC II, IBM 701, IBM 702(puncher card processing equipment) - faster, more memory

1. **Second Generation**

* **Transistors**: replacing vacuum tubes. Less heat, smaller, cheaper, silicon.
* Oher changes: More complex ALU, High-level programming languages, System software.
* Transistors based computers: PDP1 – by DEC; IBM 7000 (Differences from IAS: **Data** **channels** (I/O modules connecting with the multiplexor) and **Multiplexor** (connecting MM, CPU and Data Channels).

1. **Third Generations**

* Integrated circuits (**Microelectronics**) – silicon wafer (A lot of chips on one wafer and a lot of gates on a single chip)
* Computer functions: Data storage, Data movement, Data processing, Control  
  2 fundamental components: Memory (Data storage), Gates (Data processing)
* **Moore’s** **Law**: The number of transistors doubles every year until 1970 when the started doubling every 18 months.  
  - Consequences: Unchanged price, higher performance, flexibility, reduced power.
* IBM 360 (Firs planned family of computers)
* PDP 8: First minicomputer, Bus structure (omnibus)
* Important developments: Semiconductor memory (nondestructive read, capacity doubles each year) and Microprocessor (First Intel 4004 – all CPU components on a single chip, then 8008; First general microprocessor 8080).

# Performance Issues

1. **Moore’s Law – increasing raw speed**

* The microprocessor is not using its full potential. Despite the increasing number of transistors. Some changes were made.

1. **List of changes**

* **Performance Balance**: While the performance of the CPU increases (especially speed), other critical components have not kept up (Memory (capacity increased, but speed not) and I/O). So the performance balance is the adjustment that needs to be done to compensate.
* **Logic and Memory performance Gap:** The CPU performance grows more strictly than the Memory performance, therefore we have a gap that increases for 50% per year.   
  Solutions for memory gap: Make DRAM wider, than deeper; Reduce frequency of memory access – More cache; Hierarchy of buses.
* **Input/Output devices:** They are even slower. Problems with moving data.  
  Solutions: Caching; Buffering; Multiple processors…

1. **Improvements in CPU organization and architecture**

* **Shrinking logic gate size** (more gates, packed tightly).
* Increase size and speed of **caches**.
* Change processor **organization** and **architecture** (parallelism, speed)

1. **Problems with clock speed and logic density**

* Power (power density, dissipating heat)
* RC delay (wires closer together – increase capacitance; Thinner wire interconnects – increase resistance)
* Memory latency (Memory speed lag processor speed)

1. **What are the benefits of increased cache capacity?**

* The level of cache memory is increased. Therefore, we have faster access – speed.

1. **What are more complex execution logics?**

* Pipelining (Enables parallel execution of instructions)
* Branch prediction (CPU analyses the code and predicts the next instruction needed and fetches it, decode and buffer)
* Superscalar execution (Allows multiple pipelining of instructions within single processor - that are not dependent on each other’s outputs)
* Data flow analysis (CPU analyses instructions dependent on each other’s results and generates valid schedule to execute them)
* Speculative (Combination of superscalarity and data flow analysis)

1. **Point of diminishing. What happened? What is the new approach?**

* Some fundamental physical limits are being reached. Benefits from cache are not giving good enough results anymore. So, we need new approach. In 2004 Intel designed the first chip with multiple processors. Therefore, the performance almost doubled, without increasing the speed. Faster CPU speeds does not mean faster applications – so we need to take advantage of the multiple processors available.

1. **The key is Balance between the main computer components (CPU, MM, I/O, interconnection** structures). Constant improvement and adjustment is needed, in order to cope with the current trends.
2. **Why are tri-gate transistors used?**

* Shrinking logic gate size has become more and more hard and complicated. That’s why we need something new and more compact to continue Moore’s Law.

1. **MIC (Many Integrated Cores) – 50 and more. Challenges in developing software.**
2. **GPU (Graphical Processing Unit) – highly parallel structure. More effective than CPU for** processing large blocks in parallel. - General Purpose computing on GPU’s (GPGPU)
3. **APU (Accelerated Processing Unit) – Combination of CPU and GPU. Typically found in SoC.**
4. **Amdahl’s Law – providing insight into the performance of parallel systems.**

* Gene Amdahl in 1967. Concluded that code needs to be parallelizable, since we are reaching the bound of the speed up, that’s no giving good enough performance.   
  Amdahl’s Law formula for program running on single processor:
* f – fraction of code infinitely parallelizable with no scheduling
* (1-f) – fraction of code inherently serial
* T – total execution time for program on a single processor
* N – number of processors that fully exploit parallel portions of code

Conclusions: If f is small, parallel processors have little effect.   
If N -> infinity, speedup bound by 1/(1-f)

* Generalization of Amdahl’s Law:
* f – fraction of the time before enhancement
* SUf – speedup of the feature after enhancement

1. **Little’s Law**

* Can be applied to a system that is in steady state and has no leakage. Queuing system.
* λ – average # of items per unit time arriving in the system
* W – average units of time that items stay in the system
* L – average # of items in the system at any one time

L = λ \* W

Each core supports multiple threads of execution. The cores share a common main memory and common cache memory. When a thread is executing, it may need to retrieve a piece of data from the common memory. The thread stops and sends out a request for that data. All such stopped threads are in a queue.  
• Each user request on a server is broken in threads.   
• λ = the average rate of total thread processing   
• L = the average number of stopped threads waiting during some relevant time   
• W = average response time

1. **Traditional performance measurements:**

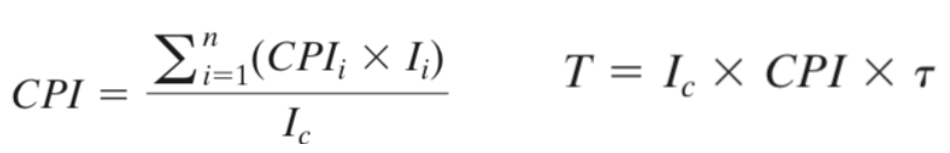
* System clock speed
* CPI
* MIPS
* MFLOPS

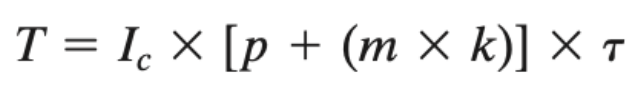
1. **Clock speed**

* Measures:   
  • Clock rate = clock speed = the rate of pulses (Hz)   
  • Clock cycle = clock tick = 1 pulse   
  • Cycle time = time between pulses
* Problems:   
  • Signals in CPU take time to settle to 1 or 0   
  • Signals may change at different speeds   
  • Operations need to be synchronized   
  • Usually we require multiple clock cycles per instruction   
  • Pipelining simultaneously executes instructions

1. **Average cycles per instruction - CPI**

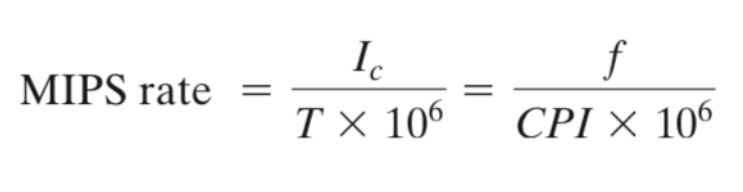
* A processor is driven by a clock with a constant frequency  
  • f = clock frequency   
  • τ = 1/f cycle time   
  Let’s define:   
  • Ic = instructions count - the number of machine instructions executed for a program until it completes or for some defined time interval.   
  • CPI = average cycles per instructions   
  • CPIi = # of cycles for instruction type i (the number of clock cycles required varies for different types of instructions)   
  • Ii = # of instructions of type i   
  • T = CPU time to execute program



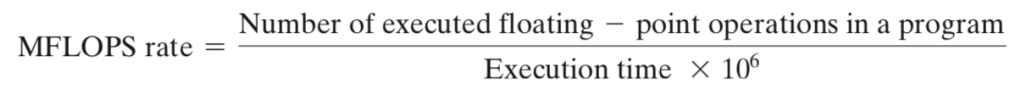
Memory cycle?  


p - the number of processor cycles needed to decode and execute the instruction,   
m - the number of memory references  
k - the ratio between memory cycle time and processor cycle time.   
Ic, p, m, k, τ are influenced by four system attributes:   
• the design of the instruction set (ISA)   
• compiler technology   
• processor implementation; and   
• cache and memory hierarchy

1. **MIPS (Million Instructions Per Second)**

* 

1. **MFLOPS (Millions of Floating Points Instructions Per Second)**

* 

1. **Measurement problems**

* Our traditional performance measurements are heavily dependent on:  
  Instruction set (CISC – complex instruction set computer: uses more complex expressions; RISC – reduced instruction set computer: uses more instructions for an operation but, simpler. Doing the same thing, same amount of time.)   
  Compiler design  
  Processor implementation  
  Cache and memory hierarchy

1. **What are benchmarks?**

* Benchmarks are programs designed to test the performance. They use high-level language. Easily measured (mean).

1. **Way to calculate the mean? What do we use the most?**

* Arithmetic, Geometric and Harmonic mean. The most consistent and widely used is Geometric Mean.

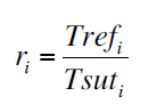
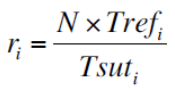
1. **Benchmark suite**

* Relevance (Benchmark should measure relative features)
* Representativeness (The performance metrics should be respected and accepted)
* Equity (All systems should be fairly compared)
* Repeatability (The results can be verified)
* Cost-effectiveness (Benchmarks test are economical)
* Scalability (Tests should measure from single servers to multiple servers)
* Transparency (Metrics should be easy to understand)

1. **The most famous benchmark – industry standard**

* SPEC (Standard Performance Evaluation Corporation) – The most famous SPEC CPU2017: performance for application that spend most the time doing computations not I/O. Emphasizes performance of Processor, Memory and Compilers.

1. **SPEC CPU2017 terms**

* Benchmark: program
* System under Test (SUT)
* Reference machine: baseline system – normalize performance metrics
* Base metrics: compiler with default settings
* Peak metrics: compiler options optimized for a system
* Speed metric: time for a system to complete a single task   
  
* Rate metric: throughput of tasks in a time frame  
  

# Functions and Interconnections

1. **What is a program?**

* Sequence of instructions. Set of steps each has its own logic or arithmetic operation and unique code.

1. **Instruction cycle**

* Fetch: Processor fetches the instruction pointed by Program Counter. Load it in instruction register. PC++. CPU performs required actions.
* Execute: CPU –> Main Memory; CPU -> I/O; Data processing (arithmetic or logic operation); Control (Jump); Combination of everything.

1. **Interrupts and kind of interrupts**

* Interrupts are events that stop the execution of the regular program because a signal from other device or program has occurred and needs immediate attention.
* Program; I/O; Timer; Hardware failure.

1. **Interrupt cycle**

* Interrupt cycle is added to the instruction cycle. First CPU checks for interrupts, if there are no interrupts, it fetches the next instruction. If an interrupts is pending: Stop the current program, save the progress and PC pointes to the address of interrupt handler routine. Process the interrupt. Restores the context and continue interrupted program.

1. **How to deal with multiple interrupts?**

* Disable interrupts: the upcoming interrupts – put them in a queue.
* Give priorities: Execute them by priority, stop the interrupt with lower priority.

1. **What is connection? Describe the 3 ways of connection.**

* The connection is important because all computer components must be able to communicate with each other. Different connection types are possible for different units.
* **Memory connection**: Signals – read, write and timing. Receives data and addresses and sends out data.
* **CPU connection**: Receives interrupt signals, data, and instructions; Writes out data.
* **I/O connection**: Output – receives data from computer and sends to peripherals.  
  Input – receives data from peripherals and sends to computer.  
  Sends interrupt signals (control).

1. **Transfers**

* Memory to CPU
* CPU to memory
* CPU to I/O
* I/O to CPU
* I/O to/from Memory

1. **Types of interconnections**

* Bus (single or multiple): Communication pathway for two or more devices – connecting main components of the system. From 50 – 100 lines: Data lines (transfer data), Address lines (identify the source or destination of the data – sends it to CPU) and Control lines (Controls the access of data and addresses. Gives control and timing information).  
  Problems with bus: frequency
* Point to Point: Direct connection between two points.

1. **Describe the Point-to-Point interconnection. List the characteristics.**

* Data link between two fixed points. Due to problems with the bus structure, new architecture had to step in to keep up with the CPU.   
  **Characteristics**: Lower latency, Higher data rate, Better scalability.

1. **Describe the Quick path interconnection.**

* Intel’s Point-to-Point interconnection. Allows multiple direct connections. It has layered protocol architecture. At the final layer, data is stored in packets to be transferred.

1. **Describe layers of QP.**

|  |
| --- |
| -Protocol – High level set of rules for exchanging packets of data between devices. In this layer, the packet is defined as the unit of transfer. One key function performed at this level is a cache coherency protocol. |
| -Routing – Provides the framework for directing packets through the fabric. Used to determine the course that a packet will traverse across the available system interconnects.  Routing tables are defined by firmware and describe the possible paths that a packet can follow. |
| -Link – Responsible for reliable transmission and flow control.  Two key functions: Data control (72 bits – Message payload), Error control (8 bits) |
| -Physical layer – Actual wires carrying the signals. Manages the multilane distribution = translation between 80 flits and 20 phits. |

1. **PCI express.**

* Intel’s PCI (Peripheral Component Interconnect) bus based. PCIe to keep pace with increasing demands of I/O devices. PCIe is point-to-point.   
  **Characteristics**:  
  Time-dependent data streams, Gigabit internet, Real time data processing, Data not equal  
  **Architecture:**Chipset acts as a buffering device, to deal with difference in data rates between I/O controllers, memory and CPU. It translates between PCIe transaction formats and the processor and memory signal and control requirements.   
  **Layers:**Physical**,** Data link (data link layers), Transaction (transaction layer packets)  
  **1 Physical**:  
  Differential signaling (as with QPI), Multilane distribution  
  **Transmission**:  
  Each transmission begins and ends with an 8-bit sequence   
  Scrambler (uniformly space data)  
  Encoding – adding 2 additional bits   
   10 for data; 01 for ordered set blocs   
  Differential signaling   
  16GB/s  
  **3 Transaction**:  
  The transaction layer (TL) receives read and write requests from the software above the TL and creates request packets for transmission to a destination via the link layer. Two packet types
* Split transaction technique: request packet – completion packet  
  Posted transaction (NACK)   
  Address spaces and transaction types • Memory • I/O • Configuration • Message  
  **Transaction layer – packet**  
  Header: (type of packet, information needed by the receiver to process the packet, including any needed routing information.)  
   Data: up to 4096   
  ECRC: enables the destination TL layer to check for errors in the header and data  
  **2 Data link**  
  Ensures reliable delivery of packets across the PCIe link.   
  Transmits two packet types - Formation of TLPs (adding sequence number and LCRC)  
   - DLLPs: Flow control; Power management; TLP ACK & NACK

# Cache Memory

1. **Location of memory**

* CPU (caches, registers)
* Internal (RAM, ROM, SSD)
* External (flash drive, CD, HDD)

1. **Capacity**

* Word size: The internal memory capacity is usually measured in words.
* Size of words = bytes
* Typical size: 8-bit, 16-bit, 32-bit words.

1. **Unit of transfer**

* Addressable unit is the smallest location of memory that can be addressed.
* Internal transfer – in words
* External transfer – in blocks: usually much bigger than words

1. **Sate the access methods**

* **Sequential**: Starts reading from the beginning and reads in order. Access time is dependent on the location of the data - tape
* **Direct**: Individual blocks have unique addresses. The access to the block is direct but then we continue with sequential search in the block. Access time dependent on the location of the data in the block. - disk
* **Random**: Individual addresses give the exact location of the data. Access time is not dependent of the location of the data. – RAM
* **Associative**: Data is located associatively based on what was used recently. Access time is independent of location of the previous access. – cache

1. **Performance**

* Access time: time between requesting the data from address and getting the valid data
* Memory cycle: Access time + recovery
* Transfer rate – rate at which data can be moved from a place to place

1. **Physical types of memory**

* Semiconductor – RAM, SSD\
* Magnetic – Disk and tape
* Optical – CD, DVD, Blu Ray
* Others – bubble, hologram

1. **Characteristics of physical types of memory**

* Volatility: Volatile – Information is lost when power is off (semiconductor – RAM); Nonvolatile – Once recorded information remains, until changed. No electric power is needed to retain information.
* Erasability: Nonerasability – cannot be alerted except by destroying the device (semiconductor – ROM).
* Power consumption

1. **Explain memory hierarchy.**

* Computer storage is separated in hierarchy based on the response time.
* At the top of the hierarchy, we have: inboard storage – registers and cache, MM  
   outboard storage – Magnetic Disk, CD-ROM, DVD…  
   Off line storage – Magnetic Tape

1. **Explain external memory.**

* Nonvolatile memory, called secondary(auxiliary). Data is stored more permanently on external memory devices. It has huge capacity. Access time is slower.  A diskcache is a cache memory that is used to speed up the process of storing and accessing data from the host hard disk.

1. **Characteristics of memory hierarchy**

* Increasing access time
* Increasing capacity
* Decreasing cost per bit
* Decreasing frequency of access to the memory from the CPU

1. **Why does the decreasing frequency to the memory work? How?**

* Locality of reference.
* It works because, usually the programs executing tend to cluster. So, the processor only works with fixed clusters of memory. Most of the needed data is stored in the cache memory and the CPU does not need to access the RAM o obtain information.

1. **Define cache memory**

* Small fast memory. It usually has multiple levels, some of which are located on the CPU, others sit between MM and CPU. Provide faster access of the data needed.

1. **Cache structure**

* Data in cache memory is stored in lines. One line is divided in 2 parts: tag and block. The tag keeps information about address where the data was stored in memory and with the line index, we get the exact address. In the block we sore the data on that address. – Main Memory is divided into blocks.

1. **What memory is used by the processes and what else is needed?**

* Processes use virtual memory. They assign memory from logic point of view. The system needs virtual memory to make it possible for multiple programs to run at the same time. But since processes use that virtual memory, we need to actually map those addresses to our actual physical memory available i.e., RAM. The feature that does that translation is Memory Management Unit (MMU) – Translates every virtual address into physical.

1. **Differences between logical and physical cache.**

* Logical cache sits between the CPU and the MMU and works with virtual addresses before the translation is made. Physical cache sits between MMU and RAM and works with physical addresses after the translation.

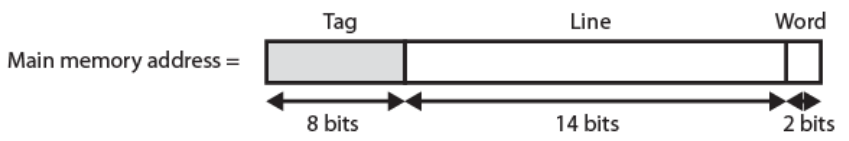
1. **The size matters**

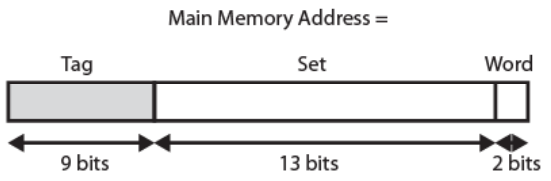
* Cost: more cache is expensive
* Speed: More cache is faster (up to some point); Checking cache takes time.
* Size available on the CPU

1. **Mapping function. Why?**

* Since we have fewer cache lines than blocks in RAM, we need to have a particular way how to map which block in which line in the cache.

1. **Techniques of mapping**

* Direct Mapping: Each block of Main Memory maps to one particular cache line.  
  Compute cache line:  
  **cache\_line\_number = MM\_block\_number % number\_of\_cache\_lines**Address divided:  
  ****  
  Pros: Simple, Inexpensive  
  Cons: Fixed location for a given block – a lot of cache misses occur
* **What is victim cache?**
* Remembers what was discarded – has that already fetched, uses it again with little penalty.
* Between direct mapped L1 cache and next memory level.
* Associative: Each block of memory maps to any cache line. Cache searching gets expensive.  
  
* Set-Associative: Since the associative mapping is expensive and direct mapping has its drawbacks, something in between was brought in and that is set-associative mapping. The cache lines are divided into sets:
* k-way set associative mapping: A given block can map in any line of cache in a given set.
* k-way associative direct mapping: The first blocks of main memory are mapped in he first lines of any set of the cache line memory.



1. **Why are replacement algorithms needed?**

* Due to the fact that the cache memory is not as big as MM, not every block of memory can be stored in the cache. So up to a point, some block will have to be discarded from cache in order to store the new block that is needed. To avoid further cache misses, we need effective algorithm to pick the right block to remove from the cache.   
  The most famous replacement algorithms:
* FIFO – discards the block that was first loaded into the cache
* LRU – discards the block that was least accessed
* LFU – discards the block with the smallest frequency of use.
* Random – only used to achieve speed

1. **Why is writing policy needed?**

* When a switch must occur and a block will be removed from cache, if the block was changed, we must not overwrite it unless the MM is up to date. Few problems with getting the not “up-to-date” information can occur when other CPUs and/or I/O modules try to access the same block of data. That is why we need to have certain write policy to dodge those problems as mush as possible.
* Write through: All writes go to the cache as well as MM. Drawback: A lot of traffic is made in order to keep local caches up to date. Slows down writes.
* Write back: Updates are made in cache only and that is noticed by setting the update bit. The data is written back to the MM if and only if the update bit is set. Drawback: Other caches got out of sync and if other process try to access that block, it may get old version.

1. **The design of line size**

* When retrieving a certain block from MM, adjacent words are retrieved as well. That brings higher hit rate at first. – Due to locality of reference. But as more and more blocks are retrieved, the memory of cache is getting full. The hit ratio will decrease as the blocks become bigger.

1. **Unified and split caches**

* Unified: One cache for instructions and data – easy to implement only one cache, higher hit rate
* Split: One cache for instructions and one for data – Important in pipelining.

# External Memory

1. **Characteristics pf RAID (Redundant Array of Independent Disks?**

* Magnetic disks
* Set of physical disks that represent a logical drive for OS
* Data distributes across physical drives
* Can use redundant capacity to store parity information

1. **RAID 0**

* Not fault tolerant – no redundancy
* The data is not duplicated
* Data access is faster
* Write operation is fast – the data is spread among 2 disks

1. **RAID 1 – mirroring and duplexing**

* Fault tolerant
* Data is copied on more than one disk
* Expensive – because we have 2 copies of everything

1. **RAID 2 – bit striping with parity**

* Minimum of 3 drives required
* Drives must be at the same angular rotation
* Hamming code
* Hardly used
* Faster reading
* Slow writing because we need to calculate the error correcting code
* We have data disks and error correcting code disks

1. **RAID 3 – bit interleaved parity**

* The data is striped in bytes
* We have dedicated parity disk
* Parity bit is calculated using the XOR operation

1. **RAID 4 – blocks interleaved parity**

* Blocks are striped
* Parity is calculated for blocks
* Dedicated parity disk

1. **RAID 5 – striping with parity**

* Requires 2 or more disks
* Large, fast
* If two disk failures occur at the same time, all the data will be lost
* Parity is used to rebuild the data in the event of disk failure
* Parity occupies a lot of memory, therefore the whole capacity of the disk is not completely used for data storage

1. **RAID 6 – striping with double parity**

* 4 or more disks
* Can handle 2 disks failures at the same time
* Parity is spread twice on every disk
* Write performance is slower