# **Chapter 1**

1. **What, in general terms, is the distinction between computer organization and computer architecture?**
   * Computer architecture refers to the attributes visible to the programmer and which affect the logical execution of a program.
   * Computer organisation refers to the hardware units that realise the architecture: the instruction set, the number of bits used to represent data types, addressing memory, etc.
2. **What, in general terms, is the distinction between computer structure and computer function?**

* Structure: The way in which the components are interrelated.
* Function: The operation of each individual component as part of the structure.

1. **What are the four main functions of a computer?**

* Data processing
* Data storage
* Data movement
* Control

1. **List and briefly define the main structural components of a computer.**

* CPU: Controls the operation of the computer and performs its data processing functions.
* Main memory: Stores data.
* I/O: Moves data between the computer and its external environment.
* System interconnection: Some mechanism (e.g. a system bus) that provides for communication among CPU, main memory, and I/O.

1. **List and briefly define the main structural components of a processor.**

* Control unit: Controls the operation of the CPU and hence the computer.
* Arithmetic and logic unit (ALU): Performs the computer's data processing functions.
* Registers: Provides storage internal to the CPU.
* CPU interconnection: Some mechanism that provides for communication among the control unit, ALU, and registers.

**Chapter 2**

1. **What is a stored program computer?**

* A computer which stores its program in memory, like it stores data. Instructions are read from the memory and executed.

1. **What are the four main components of any general-purpose computer?**

* Main memory
* Arithmetic-logic unit
* Program Control unit
* I/O equipment

1. **At the integrated circuit level, what are the three principal constituents of a computer system?**

* Transistors
* Resistors
* Capacitors

1. **Explain Moore’s law.**

* Moore observed that the number of transistors that could be put on a single chip was doubling every year. The pace continued until the 1970s when it slowed down to a doubling every 18 months.

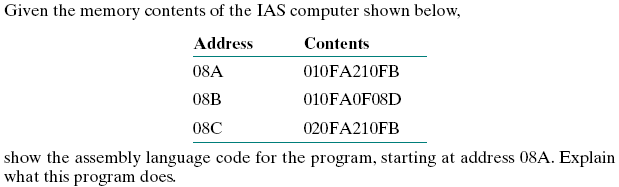
1. **List and explain the key characteristics of a computer family.**

* compatibility: a program written for one member should work on any computer of the family
* similar instruction set
* similar operating system
* increasing speed, I/O ports, memory, cost

1. **What is the key distinguishing feature of a microprocessor?**

* All the components of the CPU were on a single chip.

1. **Refer to the table 2.1**



|  |  |
| --- | --- |
| Address | Contents |
| 08A | LOAD M(0FA)  STOR M(0FB) |
| 08B | LOAD M(0FA)  JUMP +M(08D) |
| 08C | LOAD -M(0FA)  STOR M(0FB) |

**Chapter 3**

1. **What general categories of functions are specified by computer instructions?**

* Processor-Memory data transfer
* Processor-I/O data transfer
* Data processing
* Control

1. **List and briefly define the possible states that define an instruction execution.**

* Instruction address calculation: Determine the address of the next instruction to be executed.
* Instruction fetch: Read instruction from its memory location into the processor.
* Instruction operation decoding: Analyze instruction to determine the type of operation to be performed and operands to be used.
* Operand address calculation: If the operation involves reference to an operand in memory or available via I/O, then determine the address of the operand.
* Operand fetch: Fetch the operand from memory or read it in from I/O.
* Data operation: Perform the operation indicated in the instruction.
* Operand store: Write the result into memory or out to I/O.

1. **List and briefly define two approaches to dealing with multiple interrupts.**

* Disabling interrupts: the processor has the ability to and will ignore specific interrupts. Those interrupts remain pending and will be checked after the processor has enabled interrupts.
* Interrupt service routine (ISR): priorities assigned to the different types of interrupts. ISRs with higher priorities can interrupt ones with lower priority, in which case the ISR with the lower priority is put on the stack until that ISR is completed.

1. **What types of transfers must a computer’s interconnection structure (e.g., bus) support?**
   * Memory to processor: The processor reads an instruction or a unit of data from memory.
   * Processor to memory: The processor writes a unit of data to memory.
   * I/O to processor: The processor reads data from an I/O device via an I/O module.
   * Processor to I/O: The processor sends data to the I/O device.
   * I/O to or from memory: For these two cases, an I/O module is allowed to exchange data directly with memory, without going through the processor, using direct memory access.
2. **What is the benefit of using a multiple-bus architecture compared to a single-bus architecture?**

* The single-bus architecture has 2 problems:  
  1. The more devices attached to the bus, the greater the bus length and hence the greater the propagation delay.  
    
  2. The bus may become a bottleneck as the aggregate data transfer demand approaches the capacity of the bus.

**CHAPTER 4**

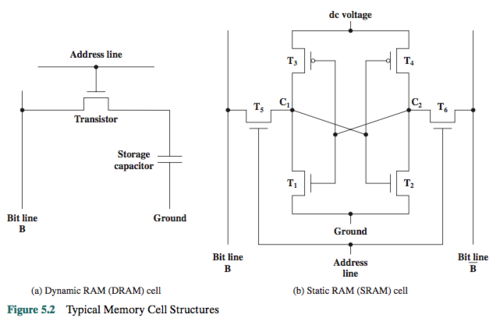
* **4.1- What are the differences among sequential access, direct access, and random access?** 
  + Sequential access is accessing data in a specific linear sequence, with an example being tape storage. Direct access has the data address being based on a physical location. With random access, any location can be selected at random, and the addressable locations in memory have a unique, physically wired-in addressing mechanism.
* **4.2-What is the general relationship among access time, memory cost, and capacity?** 
  + As access time becomes faster, the cost per bit increases. As memory size increases, the cost per bit is smaller. Also, with greater capacity, the access time becomes slower.
* **4.3- How does the principle of locality relate to the use of multiple memory levels?** 
  + Slower and less expensive memory is used in higher stages, with the most expensive being the registers in the processor as well as cache. Main memory is slower and less expensive, and is outside of the processor.
* **4.4- What are the differences among direct mapping and associative mapping,?** 
  + Direct mapping maps each block of main memory into only one possible cache line. Associative mapping permits each main memory block to be loaded into any line of the cache. The set-associative mapping combines both methods while decreasing disadvantages. The cache consists of a number of sets, each of which consists of a number of line.
* **4.5- For a direct-mapped cache, a main memory address is viewed as consisting of three fields. List and define the three fields.** 
  + The fields would be i, j, and m. I is the cache line number, j is the main memory block number, and m is the number of lines in the cache.
* **4.6- For an associative cache, a main memory address is viewed as consisting of two fields. List and define the two fields.** 
  + Tag and Word fields. Tag field uniquely identifies a block of main memory. The word is what is to be placed in the block of memory.

**CHAPTER 5**

**5.1 What are the key properties of semiconductor memory?**

• It has two (semi)stable states which can be used to represent binary 1 and 0  
• It supports read/write operations

**5.2 What are two interpretations of the term random-access memory?**

• DRAM  
• SRAM

**5.3 What is the difference between DRAM and SRAM in terms of application?**

- SRAM is used for cache memory  
- DRAM is used for main memory

**5.4 What is the difference between DRAM and SRAM in terms of characteristics such as speed, size, and cost?**

- speed: SRAM is faster  
- size: SRAM takes more space, DRAM is denser  
- cost: SRAM is more expensive than DRAM

**5.5 Explain why one type of RAM is considered to be analog and the other digital.**

- DRAM: analog device because it stores charge and uses a threshold to determine the binary value  
- SRAM: digital because it uses flip-flop logic gates

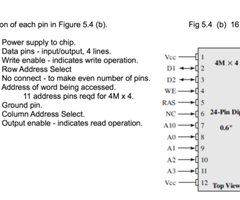
**5.6 What are some applications for ROM?**

• Microprogramming  
• Library subroutines for frequently wanted functions  
• System programs  
• Function tables

**5.7 What are the differences among EPROM, EEPROM, and flash memory?**

EPROM:  
- read/written electrically  
- before writing, all cells must be erased by exposure to UV light  
- price: $  
  
EEPROM:  
- can be written to any time, without erasing contents  
- price: $$$  
  
flash memoy:  
- electrical erasing (in seconds), faster than EPROM  
- price: $$

**5.8 Explain the function of each pin in Figure 5.4b.**



**5.9 What is a parity bit?**

* A bit appended to an array of binary digits to make the sum of all the binary digits, including the parity bit, always odd (odd parity), or always even (even parity).

**5.10 How is the syndrome for the Hamming code interpreted?**

* Each bit of the syndrome is 0 or 1 according to if there is or is not a match in that bit position for the two inputs.

**5.11 How does SDRAM differ from ordinary DRAM?**

* SDRAM:  
  - synchronous, unlike traditional DRAM  
  - synchronized with the system bus

CHAPTER 6

**6.1 What are the advantages of using a glass substrate for a magnetic disk?**

• increased disk reliability  
 • less surface defects  
 • better stiffness to reduce disk dynamics  
 • greater ability to withstand shock and damage

**6.2 How are data written onto a magnetic disk?**

* Pulses are sent to the write head, then an electric current magnetizes a small area of the recording medium to store the "pulses"

**6.3 How are data read from a magnetic disk?**

* The read head consists of a partially shielded magnetoresistive (MR) sensor that senses the magnetization of the medium
  1. **Explain the difference between a simple CAV system and a multiple zoned recording system.**

• Constant angular velocity (CAV) system: the number of bits per track is constant;  
 • An increase in density is achieved with multiple zoned recording, in which the surface is divided into a number of zones, with zones further from the centre containing more bits than zones closer to the centre.

**6.5 Define the terms track, cylinder, and sector.**

* Track - On a magnetic disk, data is organized on the platter in concentric sets of rings, called tracks.
* Cylinder - On a disk with multiple platters, the set of all tracks in the same relative position on the platter is referred to as a cylinder.
* Sector - Data are transferred to and from the disk in sectors.

**6.6 What is the typical disk sector size?**

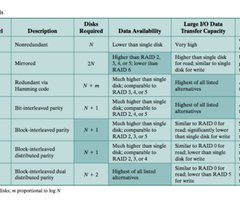
* 512 bytes

**6.7 Define the terms seek time, rotational delay, access time, and transfer time.**

* Seek time - Time taken to position the head at the track.
* Rotational delay - Once the track is selected, the disk controller waits until the appropriate sector rotates to line up with the head. The time it takes for the beginning of the sector to reach the head is known as the rotational delay.
* Access time - The sum of the seek time, if any, plus the rotational delay. The time it takes to get into position to read or write.
* Transfer time - Time taken for data transfer. Once the head is in position, the read or write operation is performed as the sector moves under the head - data transfer portion of the operation.

**6.8 What common characteristics are shared by all RAID levels?**

* RAID is a set of physical disk drives viewed by the operating system as a single logical drive.
* Data are distributed across the physical drives of an array in a scheme known as striping.
* Redundant disk capacity is used to store parity information, which guarantees data recoverability in case of a disk failure.

**6.9 Briefly define the seven RAID levels**.

* RAID 0 - Non-redundant.
* RAID 1 - Mirrored, every disk has a mirror disk containing the same data.
* RAID 2 - Redundant via Hamming code; an error-correcting code is calculated across corresponding bits on each data disk, and the bits of the code are stored in the corresponding bit positions on multiple parity disks.
* RAID 3 - Bit-interleaved parity;
* RAID 4 - Block-interleaved parity;
* RAID 5 - Block-interleaved distributed parity;
* RAID 6 - Block-interleaved dual distributed parity;

**6.10 Explain the term striped data.**

* The disk is divided into strips, which may be physical blocks, sectors, or some other unit. The strips are mapped round robin to consecutive array members. A set of logically consecutive strips that maps exactly one strip to each array member is referred to as a stripe.

**6.11 How is redundancy achieved in a RAID system?**

* The disk is divided into strips, which may be physical blocks, sectors, or some other unit. The strips are mapped round robin to consecutive array members. A set of logically consecutive strips that maps exactly one strip to each array member is referred to as a stripe.

**6.12 In the context of RAID, what is the distinction between parallel access and independent access?**

* RAID 1: by having two identical copies of all data  
  The rest: by the use of error-correcting codes

**Chapter 7**

**7.1- List three broad classifications of external, or peripheral, devices.**

* + Human readable, machine readable, and communication.

**7.2- What is the International Reference Alphabet?**

* + It is 7-bit text code used to represent characters (numbers, letters, special characters).

**7.3- What are the major functions of an I/O module?**

* + Control and timing, processor communication, device communication, data buffering, error detection.

**7.4- List and briefly define three techniques for performing I/O.**

* + Programmed I/O – data exchanged between the processor and the I/O module. The processor executes a program that gives it direct control of the I/O operation, including sensing device status, sending a read or write command, and transferring the data.  
      
    Interrupt-driven I/O – processor issues I/O command, then goes off to do other things until the I/O module interrupts the processor to request service when it is ready to exchange data with the processor.  
      
    Direct Memory Access – involves an additional module on the system bus. The DMA module is capable of mimicking the processor, and, indeed, of taking over control of the system from the processor.

**7.5- What is the difference between memory-mapped I/O and isolated I/O?**

* + With memory mapped I/O, there is a single address space for memory locations and I/O drives. With isolated I/Ok, the address space for I/O is isolated from that from memory, since the bus may be equipped with memory read and right plus input and output command lines.

**7.6- When a device interrupt occurs, how does the processor determine which device issued the interrupt?**

* + It can use software polling, polling each I/O module to see which one caused the interrupt, daisy chaining, with a message sent from the processor when it receives an interrupt, and the message going through each module until it reaches the module, which then sends an response to the processor, or using vectored interrupts for bus arbitration. The module takes over the bus line, and only one module can do so at a time.

**7.7- When a DMA module takes control of a bus, and while it retains control of the bus, what does the processor do?**

* + The processor either doesn't need to use the bus at the time, or it is forced to suspend operation temporarily. The processor deals with other things, while the bus operation is left to DMA.

**Chapter 8**

1. **What is an operating system?**

* An operating system (OS) is [system software](https://en.wikipedia.org/wiki/System_software) that manages [computer hardware](https://en.wikipedia.org/wiki/Computer_hardware), [software](https://en.wikipedia.org/wiki/Computer_software) resources, and provides common [services](https://en.wikipedia.org/wiki/Daemon_(computing)) for [computer programs](https://en.wikipedia.org/wiki/Computer_program).

1. **List and briefly define the key services provided by an OS.**

* Program creation – It provides a variety of facilities and services, such as editors and debuggers, to assist the programmer in creating programs.
* Program execution – OS handles such things as loading instructions and data into memory, and initialization of I/O devices and files.
* Access to I/O devices – OS takes care of the details involved with I/O device operation (providing the required specific sets of instructions or control signals required for operation).
* Controlled access to files – OS worries about the details involved such as understanding the nature of the I/O device and also the file format on the storage medium.
* System access – It provides protections of resources and data from unauthorized users and also resolves conflicts for resource contention.
* Error detection and response – The OS must make the response that clears error conditions with the least impact on running applications.
* Accounting – good OS collects usage statistics for various resources and monitors performance parameters such as response time.

1. **List and briefly define the major types of OS scheduling.**

* Long-term scheduling – the long-term scheduler determines which programs are admitted to the system for processes. It controls the degree of multiprogramming (number of processes in memory).
* Medium-term scheduling – This is part of the swapping section. Typically, the swapping-in decision is based on the need to manage the degree of multiprogramming. Determines whether to add to the number of processes that are partially or fully in main memory.
* Short-term scheduling – also known as the dispatcher. Executes frequently and makes the fine-grained decision of which job to execute next.
* I/O scheduling – The decision as to which process's pending I/O request shall be handled by an available I/O device.

**Chapter 12**

1. **What are the typical elements of a machine instruction?**

* Opcodes, source operand reference, result operand reference and next instruction reference.

1. **What types of locations can hold source and destination operands?**

* Main or virtual memory, processor register, immediate (contained in a field in the instruction being executed), and an I/O device.

1. **If an instruction contains four addresses, what might be the purpose of each address?**

* The purpose could be to access or store information in memory or registers, or to hold datafromarithmeticoperations.

1. **List and briefly explain five important instruction set design issues.**

* Operation repertoire – How many and which operations to provide.  
    
  Data types: various types of data upon which operations are performed.  
    
  Instruction format: Instruction length(in bits), number of addresses, size of various fields, etc.  
    
  Registers: Number of processor registers that can be referenced by instructions, and their use  
    
  Addressing: mode or modes by which address of an operand is specified.

1. **What types of operands are typical in machine instruction sets?**

* Addresses, numbers, characters, and logical data.

1. **What is the relationship between the IRA character code and the packed decimal representation?**

* The last four bits in IRA bit pattern 011XXXX, the numbers 0-9 are represented using the same 4-bits as packed decimal.

1. **What is the difference between an arithmetic shift and a logical shift?**

* In a logical shift, the bits of a word are shifted left or right. On one end, the bit that is shifted out is lost, while on the other end a zero is added in. With arithmetic shift, treats the data as a signed integer and does not shift the sign bit. On a right shift, the sign bit is replicated into the bit position to its right, while with a left shift, a logic shift is performed on all bits but the sign bit.

1. **Why are transfer of control instructions needed?**

* They are required for looping (allowing for repeated calls to instructions), allow for decision making, and it allows for breaking up programs into smaller pieces for easier programming.

1. **List and briefly explain two common ways of generating the condition to be tested in a conditional branch instruction.**

* One way is to test for some condition caused by an arithmetic operation, and checking for 0, positive, negative, or overflow. A three-address instruction format can also be used, with two addresses used for condition testing.

1. **What is meant by the term nesting of procedures?**

* Being able to call a procedure from within a procedure.

1. **List three possible places for storing the return address for a procedure return.**

* Register, start of called procedure, or the top of the stack.

**Chapter 13**

**13.1 Briefly define immediate addressing.**

* Immediate addressing: The value of the operand is in the instruction.

**13.2 Briefly define direct addressing.**

* The address field contents the effective address of the operand.

**13.3 Briefly define indirect addressing.**

* The address field refers to the address of a word in memory, which in turn contains the effective address of the operand.

**13.4 Briefly define register addressing.**

* The address field refers to a register that contains the operand.

**13.5 Briefly define register indirect addressing.**

* Register indirect addressing: The address field refers to a register, which in turn  
  contains the effective address of the operand.

**13.6 Briefly define displacement addressing.**

* Displacement addressing: The instruction has two address fields, at least one of  
  which is explicit. The value contained in one address field (value = A) is used  
  directly. The other address field refers to a register whose contents are added to A  
  to produce the effective address.

**13.7 Briefly define relative addressing.**

* Relative addressing: The implicitly referenced register is the program counter  
  (PC). That is, the current instruction address is added to the address field to  
  produce the EA.

**13.8 What is the advantage of autoindexing?**

* It is typical that there is a need to increment or decrement the index register after  
  each reference to it. Because this is such a common operation, some systems will  
  automatically do this as part of the same instruction cycle, using autoindexing.

**13.9 What is the difference between postindexing and preindexing?**

* These are two forms of addressing, both of which involve indirect addressing and  
  indexing. With preindexing, the indexing is performed before the indirection.  
  With postindexing, the indexing is performed after the indirection.

**13.10 What facts go into determining the use of the addressing bits of an instruction?**

* Number of addressing modes.
* Number of operands: Typical instructions on  
  today's machines provide for two operands.
* Register versus memory: The more that registers can be used for operand references, the fewer bits are needed.
* Number of register sets: More the Better
* Address range: This matters
* Address granularity.

**13.11 What are the advantages and disadvantages of using a variable-length instruction format?**

* Advantages: It easy to provide a large repertoire of opcodes, with different  
  opcode lengths. Addressing can be more flexible, with various combinations of  
  register and memory references plus addressing modes.
* Disadvantages: an increase in the complexity of the CPU.

**Chapter 14**

1. **What general roles are performed by processor registers?**

* User-visible registers: These enable the machine- or assembly language  
  programmer to minimize main-memory references by optimizing use of registers.  
  Control and status registers: These are used by the control unit to control the  
  operation of the CPU and by privileged, operating system programs to control the  
  execution of programs.

1. **What categories of data are commonly supported by user-visible registers?**

* General purpose; Data; Address; Condition codes

1. **What is the function of condition codes?**

* Condition codes are bits set by the CPU hardware as the result of operations.

1. **What is a program status word?**

* All CPU designs include a register or set of registers, often known as the program  
  status word (PSW), that contain status information. The PSW typically contains  
  condition codes plus other status information.

1. **Why is a two-stage instruction pipeline unlikely to cut the instruction cycle time in half, compared with the use of no pipeline?**

* 1)The execution time will generally be longer than the fetch time. Execution will  
  involve reading and storing operands and the performance of some operation.  
  Thus, the fetch stage may have to wait for some time before it can empty its buffer.  
  (2) A conditional branch instruction makes the address of the next instruction to be  
  fetched unknown. Thus, the fetch stage must wait until it receives the next  
  instruction address from the execute stage. The execute stage may then have to  
  wait while the next instruction is fetched

1. **List and briefly explain various ways in which an instruction pipeline can deal with conditional branch instructions.**

* **Multiple streams:** A brute-force approach is to replicate the initial portions of the pipeline and allow the pipeline to fetch both instructions, making use of two streams.
* **Prefetch branch target:** When a conditional branch is recognized, the target of the branch is prefetched, in addition to the instruction following the branch. This target is then saved until the branch instruction is executed. If the branch is taken, the target has already been prefetched.
* **Loop buffer:** A loop buffer is a small, very-high-speed memory maintained by the instruction fetch stage of the pipeline and containing the**n** most recently fetched instructions, in sequence. If a branch is to be taken, the hardware first checks whether the branch target is within the buffer. If so, the next instruction is fetched from the buffer.
* **Branch prediction:** A prediction is made whether a conditional branch will be taken when executed, and subsequent instructions are fetched accordingly.
* **Delayed branch:** It is possible to improve pipeline performance by automatically rearranging instructions within a program, so that branch instructions occur later than actually desired.

1. **How are history bits used for branch prediction?**

* These bits are referred to as a taken/not  
  taken switch that directs the processor to make a particular decision the next time  
  the instruction is encountered.

**Chapter 15**

**15.1 What are some typical distinguishing characteristics of RISC organization?**

* a limited instruction set with a fixed format
* a large number of registers or the use of a compiler that optimizes register usage, and
* an emphasis on optimizing the instruction pipeline.

**15.2 Briefly explain the two basic approaches used to minimize register-memory operations on RISC machines.**

* Two basic approaches are possible, one based on software and the other on hardware. The software approach is to rely on the compiler to maximize register usage. The compiler will attempt to allocate registers to those variables that will be used the most in a given time period. This approach requires the use of sophisticated program-analysis algorithms. The hardware approach is simply to use more registers so that more variables can be held in registers for longer periods of time.

**15.3 If a circular register buffer is used to handle local variables for nested procedures, describe two approaches for handling global variables.**

* Variables declared as global in an HLL can be assigned memory locations by the compiler, and all machine instructions that reference these variables will use memory-reference operands.  
  (2) Incorporate a set of global registers in the processor. These registers would be fixed in number and available to all procedures.

**15.4 What are some typical characteristics of a RISC instruction set architecture?**

* One instruction per cycle. Register-to-register operations. Simple addressing modes. Simple instruction formats.

**15.5 What is a delayed branch?**

* Delayed branch, a way of increasing the efficiency of the pipeline, makes use of a branch that does not take effect until after execution of the following instruction.

**Chapter 16**

**16.1 What is the essential characteristic of the superscalar approach to processor design?**

* Characteristics of superscalar approach to processor design:

Common instructions such as load/store instructions are simultaneously initiated and execution is done independently. Processing in superscalar approach issues more than one instruction per cycle. Out-of-order execution is allowed by superscalar approach.

**16.2 What is the difference between the superscalar and super pipelined approaches?**

* Super-pipelining attempts to increase performance by reducing the clock cycle time. It achieves that by making each pipeline stage very shallow, resulting in a large number of pipe stages. A shorter clock cycle means a faster clock. As long as your cycles per instruction (CPI) doesn’t change, a faster clock means better performance. Super-pipelining works best with code that doesn’t branch often, or has easily predicted branches.
* Superscalar attempts to increase performance by executing multiple instructions in parallel. If you can issue more instructions every cycle—without decreasing clock rate—then your CPI decreases, therefore increasing performance.

**16.3 What is instruction-level parallelism?**

* ILP must not be confused with [concurrency](https://en.wikipedia.org/wiki/Concurrency_(computer_science)), since the first is about parallel execution of a sequence of instructions belonging to a specific [thread of execution](https://en.wikipedia.org/wiki/Thread_(computing)) of a [process](https://en.wikipedia.org/wiki/Process_(computing)) (that is a running program with its set of resources - for example its [address space](https://en.wikipedia.org/wiki/Address_space), a set of [registers](https://en.wikipedia.org/wiki/Processor_register), its identifiers, its state, program counter, and more). Conversely, concurrency regards with the threads of one or different processes being assigned to a [CPU](https://en.wikipedia.org/wiki/Central_processing_unit)'s core in a strict alternance or in true parallelism if there are enough CPU's cores, ideally one core for each runnable thread.

**16.4 Briefly define the following terms: • True data dependency • Procedural dependency • Resource conflicts • Output dependency • Antidependency**

* True data dependency : Input of the next instruction is the output of the previous (RAW)
* Procedural dependency : Previous instruction is a branch, code of the target can cause affects on input of the next
* Resource conflicts : 2 instructions access the same resource (bus, registers,…)
* Output dependency : 2 instructions write values to the same output (Write-after-write - WAW)
* Antidependency : Write-after-read situation (WAR)

**16.5 What is the distinction between instruction-level parallelism and machine parallelism?**

* Instruction-level parallelism (ILP) the average number of instructions in a program that a processor might be able to execute at the same time. ... Machine parallelism of a processor is the ability of the processor to take advantage of the ILP of the program.

**16.6 List and briefly define three types of superscalar instruction issue policies.**

* Three types of superscalar instruction issue policies:

In-order issue with in-order completion. In-order issue with out-of order completion. Out-of-order issue with out-of-order execution.

**16.7 What is the purpose of an instruction window?**

* Instruction window is a buffer which allows out-of-order issue to decouple the following stages. ... When once processor finishes decoding the instruction, it is placed in the buffer known as instruction window.

**16.8 What is register renaming and what is its purpose?**

* Register renaming is a form of pipelining that deals with data dependences between instructions by renaming their register operands. ... Renaming replaces architectural register names by, in effect, value names, with a new value name for each instruction destination operand.

**16.9 What are the key elements of a superscalar processor organization?**

* The key elements of a superscalar processor are outlined below: Instruction fetch strategies that simultaneously fetch multiple instructions, often by predicting the outcomes of, and fetching beyond, conditional branch instructions.

**Chapter 17**

**17.1 List and briefly define three types of computer system organization.**

* Single instruction, single data (SISD) stream: A single processor executes a single instruction stream to operate on data stored in a single memory.
* Single instruction, multiple data (SIMD) stream: A single machine instruction controls the simultaneous execution of a number of processing elements on a lockstep basis. Each processing element has an associated data memory, so that each instruction is executed on a different set of data by the different processors.
* Multiple instruction, multiple data (MIMD) stream: A set of processors simultaneously execute different instruction sequences on different data sets.

**17.2 What are the chief characteristics of an SMP(symmetric multiprocessor)?**

* There are two or more similar processors of comparable capability.
* These processors share the same main memory and I/O facilities and are interconnected by a bus or other internal connection scheme, such that memory access time is approximately the same for each processor.
* All processors share access to I/O devices, either through the same channels or through different channels that provide paths to the same device.
* All processors can perform the same functions (hence the term symmetric).
* The system is controlled by an integrated operating system that provides interaction between processors and their programs at the job, task, file, and data element levels.

**17.3 What are some of the potential advantages of an SMP compared with a uniprocessor?**

* Performance: If the work to be done by a computer can be organized so that some portions of the work can be done in parallel, then a system with multiple processors will yield greater performance than one with a single processor of the same type.
* Availability: In a symmetric multiprocessor, because all processors can perform the same functions, the failure of a single processor does not halt the machine. Instead, the system can continue to function at reduced performance.
* Incremental growth: A user can enhance the performance of a system by adding an additional processor.
* Scaling: Vendors can offer a range of products with different price and performance characteristics based on the number of processors configured in the system.

**17.4 What are some of the key OS design issues for an SMP?**

* Simultaneous concurrent processes: OS routines need to be reentrant to allow several processors to execute the same IS code simultaneously. With multiple processors executing the same or different parts of the OS, OS tables and management structures must be managed properly to avoid deadlock or invalid operations.
* Scheduling: Any processor may perform scheduling, so conflicts must be avoided. The scheduler must assign ready processes to available processors.
* Synchronization: With multiple active processes having potential access to shared address spaces or shared I/O resources, care must be taken to provide effective synchronization. Synchronization is a facility that enforces mutual exclusion and event ordering.
* Memory management: Memory management on a multiprocessor must deal with all of the issues found on uniprocessor machines. In addition, the operating system needs to exploit the available hardware parallelism, such as multiported memories, to achieve the best performance. The paging mechanisms on different processors must be coordinated to enforce consistency when several processors share a page or segment and to decide on page replacement.
* Reliability and fault tolerance: The operating system should provide graceful degradation in the face of processor failure. The scheduler and other portions of the operating system must recognize the loss of a processor and restructure management tables accordingly.

**17.5 What is the difference between software and hardware cache coherent schemes?**

* Cache coherency deals with keeping all caches in a shared multiprocessor system to be coherent with respect to data when multiple processors read/write to same address.
* Hardware cache coherency schemes are commonly used as it benefits from better performance and lesser complexity and issues. Based on the number of processors/caches in the multiprocessor sub system and the cache policies - there are different coherency protocols that can be implemented in hardware to keep all caches coherent. Some of the commonly used schemes are snoop based protocols (works well for smaller number of processors/caches) and directory based protocols (more suitable for larger systems). These can further be classified based on update vs invalidate protocols (based on whether data gets transferred across caches or invalidate) and based on the number of states used to track the data in the cache (MESI / MESIF / MOESI / MOESIF)
* Software cache coherency schemes are implemented in software and uses a cache flush or cache invalidate instruction supported by hardware. If one processor reads/writes an address and caches the copy in its local cache, the software will need to execute a “cache flush” or “cache invalidate” for that address and make sure the latest data is written to memory before another processor can use that same address.

Now that means more instructions to execute (making this slower) and more complexity if more address space is shared across caches to keep track of all of these addresses. Hence software based schemes are not commonly used in a single chip/cluster that has several processors/caches.

Some applications where software based coherency is still used are across multiple clusters of CPUs which might need smaller address space to be shared and less frequently (while within the cluster hardware coherency would be faster) as well as between the address space shared between one or more CPU and GPUs when integrated on a heterogeneous system.

**17.6 What is the meaning of each of the four states in the MESI protocol?**

* Modified (M)

The cache line is present only in the current cache, and is dirty - it has been modified (M state) from the value in [main memory](https://en.wikipedia.org/wiki/Main_memory). The cache is required to write the data back to main memory at some time in the future, before permitting any other read of the (no longer valid) main memory state. The write-back changes the line to the Shared state(S).

* Exclusive (E)

The cache line is present only in the current cache, but is clean - it matches main memory. It may be changed to the Shared state at any time, in response to a read request. Alternatively, it may be changed to the Modified state when writing to it.

* Shared (S)

Indicates that this cache line may be stored in other caches of the machine and is clean - it matches the main memory. The line may be discarded (changed to the Invalid state) at any time.

* Invalid (I)

Indicates that this cache line is invalid (unused).

**Chapter 18**

**18.1 Summarize the differences among simple instruction pipelining, superscalar, and simultaneous multithreading.**

* Pipelining: Individual instructions are executed through a pipeline of stages so that while one instruction is executing in one stage of the pipeline, another instruction is executing in another stage of the pipeline.
* Superscalar: Multiple pipelines are constructed by replicating execution resources. This enables parallel execution of instructions in parallel pipelines, so long as hazards are avoided.
* Simultaneous multithreading (SMT): Register banks are replicated so that multiple threads can share the use of pipeline resources.

**18.2 Give several reasons for the choice by designers to move to a multicore organization rather than increase parallelism within a single processor.**

* In the case of **pipelining**, simple 3-stage pipelines were replaced by pipelines with 5 stages, and then many more stages, with some implementations having over a dozen stages. There is a practical limit to how far this trend can be taken, because with more stages, there is the need for more logic, more interconnections, and more control signals.
* With **superscalar** organization, performance increases can be achieved by increasing the number of parallel pipelines. Again, there are diminishing returns as the number of pipelines increases. More logic is required to manage hazards and to stage instruction resources. Eventually, a single thread of execution reaches the point where hazards and resource dependencies prevent the full use of the multiple pipelines available. This same point of diminishing returns is reached with SMT, as the complexity of managing multiple threads over a set of pipelines limits the number of threads and number of pipelines that can be effectively utilized.

**18.3 Why is there a trend toward giving an increasing fraction of chip area to cache memory?**

* The growing trend towards giving an increasing fraction of chip area to cache memory is mainly due to the fact that Cache memory uses less power than logic. This makes it an efficient option to use in chip design.

**18.5 At a top level, what are the main design variables in a multicore organization?**

* The number of core processors on the chip
* The number of levels of cache memory
* The amount of cache memory that is shared

**18.6 List some advantages of a shared L2 cache among cores compared to separate dedicated L2 caches for each core.**

* Constructive interference can reduce overall miss rates. That is, if a thread on one core accesses a main memory location, this brings the frame containing the referenced location into the shared cache. If a thread on another core soon thereafter accesses the same memory block, the memory locations will already be available in the shared on-chip cache.
* A related advantage is that data shared by multiple cores is not replicated at the shared cache level.
* With proper frame replacement algorithms, the amount of shared cache allocated to each core is dynamic, so that threads that have a less locality can employ more cache.
* Interprocessor communication is easy to implement, via shared memory locations.
* The use of a shared L2 cache confines the cache coherency problem to the L1 cache level, which may provide some additional performance advantage.