Computer Organization and Architecture: Designing for Performance

William Stallings, 8th Edition

Topics to Focus	Detail
1. INTRODUCTION	Organization vs Architecture, Structure vs Function, Performance
2. THE COMPUTER SYSTEM	Computer Function and Interconnection, Cache Memory, Internal Memory, External Memory, I/O, OS Support
3. CPU	Computer Arithmetic, Instruction Sets, Processor Structure and Function, RISC, Instruction-Level Parallelism, Control Unit Operation, Microprogrammed Control, Parallel Processing, Multicore Computers
General Objective: Examine the structure and functions of the components comprising a contemporary computer system.	

Computer Organization and Architecture by William Stallings, 8th Edition, Review QAs

1. Organization and Architecture, Structure and Function

1.1. What, in general terms, is the distinction between computer organization and computer architecture?

- **Computer architecture** refers to those attributes of a system visible to a programmer or, put another way, those attributes that have a direct impact on the logical execution of a program.
- **Computer organization** refers to the operational units and their interconnections that realize the architectural specifications. Examples of architectural attributes include the instruction set, the number of bits used to represent various data types (e.g., numbers, characters), I/O mechanisms, and techniques for addressing memory. **Organizational attributes** include those hardware details transparent to the programmer, such as control signals; interfaces between the computer and peripherals; and the memory technology used.

1.2. What, in general terms, is the distinction between computer structure and computer function?

- **Computer structure** refers to the way in which the components of a computer are interrelated.
- **Computer function** refers to the operation of each individual component as part of the structure.

1.3. What are the four main functions of a computer?

- Data processing; data storage; data movement; and control.

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

- **Central Processing Unit (CPU):** Controls the operation of the computer and performs its data processing functions; often simply referred to as processor
- **Main Memory:** Stores data
- **I/O:** Moves data between the computer and its external environment
- **System interconnection:** Some mechanism that provides for communication among CPU, main memory, and I/O. A common example of system interconnection is by means of a system bus, consisting of a number of conducting wires to which all the other components attach.

1.5. 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

2. Computer Performance: Designing for Performance and Performance Assessment

2.1. What is a stored program computer?

- In a **stored program computer,** programs are represented in a form suitable for storing in memory alongside the data. The computer gets its instructions by reading them from memory, and a program can be set or altered by setting the values of a portion of memory.

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

- A **main memory**, which stores both data and instructions: an **arithmetic and logic unit (ALU)** capable of operating on binary data; a **control unit,** which interprets the instructions in memory and causes them to be executed; and **input and output (I/O) equipment** operated by the control unit.

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

- Gates, memory cells, and interconnections among gates and memory cells.

2.4. Explain Moore's law.

- Moore observed that the number of transistors that could be put on a single chip was doubling every year and correctly predicted that this pace would continue into the near future.

2.5. List and explain the key characteristics of a computer family.

- **Similar or identical instruction set:** In many cases, the same set of machine instructions is supported on all members of the family. Thus, a program that executes on one machine will also execute on any other.
- **Similar or identical operating system:** The same basic operating system is available for all family members.
- **Increasing speed:** The rate of instruction execution increases in going from lower to higher family members.
- **Increasing Number of I/O ports:** In going from lower to higher family members.
- **Increasing memory size:** In going from lower to higher family members.
- **Increasing cost:** In going from lower to higher family members.

2.6. What is the key distinguishing feature of a microprocessor?

- In a microprocessor, all of the components of the CPU are on a single chip.

THE COMPUTER SYSTEM

3. Computer Function and Interconnection

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

- **Processor-memory:** Data may be transferred from processor to memory or from memory to processor**.**
- **Processor-I/O:** Data may be transferred to or from a peripheral device by transferring between the processor and an I/O module.
- **Data processing:** The processor may perform some arithmetic or logic operation on data.
- **Control:** An instruction may specify that the sequence of execution be altered.

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

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

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

1) Disable all interrupts while an interrupt is being processed.
2) Define priorities for interrupts and to allow an interrupt of higher priority to cause a lower-priority interrupt handler to be interrupted.

3.4. 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 I/O device via 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, I/O module is allowed to exchange data directly with memory, without going through the processor, using direct memory access (DMA).

3.5. What is the benefit of using a multiple-bus architecture compared to a single-bus architecture?

- With multiple buses, there are fewer devices per bus.
- This **(1)** reduces propagation delay, because each bus can be shorter, and **(2)** reduces bottleneck effects.

3.6. List and briefly define the functional groups of signal lines for PCI.

- **System pins**: Include the clock and reset pins.
- **Address and data pins**: Include 32 lines that are time multiplexed for addresses and data.
- **Interface control pins:** Control timing of transactions and provide coordination among initiators and targets.
- **Arbitration pins:** Unlike the other PCI signal lines, these are not shared lines. Rather, each PCI master has its own pair of arbitration lines that connect it directly to the PCI bus arbiter.
- **Error Reporting pins:** Used to report parity and other errors.
- **Interrupt Pins:** These are provided for PCI devices that must generate requests for service.
- **Cache support pins:** These pins are needed to support a memory on PCI that can be cached in the processor or another device.
- **64-bit Bus extension pins:** Include 32 lines that are time multiplexed for addresses and data and that are combined with the mandatory address/data lines to form a 64-bit address/data bus.
- **JTAG/Boundary Scan Pins**: These signal lines support testing procedures defined in IEEE Standard 1149.1.

4. Cache Memory

4.1. What are the differences among sequential access, direct access, and random access?

- **Sequential access:** Memory is organized into units of data, called records. Access must be made in a specific linear sequence.
- **Direct access:** Individual blocks or records have a unique address based on physical location. Access is accomplished by direct access to reach a general vicinity plus sequential searching, counting, or waiting to reach the final location.
- **Random access:** Each addressable location in memory has a unique, physically wired-in addressing mechanism. The time to access a given location is independent of the sequence of prior accesses and is constant.

4.2. What is the general relationship among access time, memory cost, and capacity?

- Faster access time, greater cost per bit; greater capacity, smaller cost per bit; greater capacity, slower access time.

4.3. How does the principle of locality relate to the use of multiple memory levels?

- It is possible to organize data across a memory hierarchy such that the percentage of accesses to each successively lower level is substantially less than that of the level above. Because memory references tend to cluster, the data in the higher-level memory need not change very often to satisfy memory access requests.

4.4. What are the differences among direct mapping, associative mapping, and set-associative mapping?

- In a cache system, **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.
- In **set-associative mapping**, the cache is divided into a number of sets of cache lines; each main memory block can be mapped into any line in a particular set.

4.5. For a direct-mapped cache, a main memory address is viewed as consisting of three fields. List and define the three fields.

- One field identifies a unique word or byte within a block of main memory. The remaining two fields specify one of the blocks of main memory. These two fields are a line field, which identifies one of the lines of the cache, and a tag field, which identifies one of the blocks that can fit into that line.

4.6. For an associative cache, a main memory address is viewed as consisting of two fields. List and define the two fields.

- A tag field uniquely identifies a block of main memory. A word field identifies a unique word or byte within a block of main memory.

4.7. For a set-associative cache, a main memory address is viewed as consisting of three fields. List and define the three fields.

- One field identifies a unique word or byte within a block of main memory. The remaining two fields specify one of the blocks of main memory. These two fields are a set field, which identifies one of the sets of the cache, and a tag field, which identifies one of the blocks that can fit into that set.

4.8. What is the distinction between spatial locality and temporal locality?

- **Spatial locality** refers to the tendency of execution to involve a number of memory locations that are clustered.
- **Temporal locality** refers to the tendency for a processor to access memory locations that have been used recently.

4.9. In general, what are the strategies for exploiting spatial locality and temporal locality?

- **Spatial locality** is generally exploited by using larger cache blocks and by incorporating prefetching mechanisms (fetching items of anticipated use) into the cache control logic.
- **Temporal locality** is exploited by keeping recently used instruction and data values in cache memory and by exploiting a cache hierarchy.

5. Internal Memory

5.1. What are the key properties of semiconductor memory?

- They exhibit two stable (or semistable) states, which can be used to represent binary 1 and 0; they are capable of being written into (at least once), to set the state; they are capable of being read to sense the state.

5.2. What are two senses in which the term random-access memory is used?

1) A memory in which individual words of memory are directly accessed through wired-in addressing logic.
2) Semiconductor main memory in which it is possible both to read data from the memory and to write new data into the memory easily and rapidly.

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

- SRAM is used for cache memory (both on and off chip), and 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?

- SRAMs generally have faster access times than DRAMs. DRAMS are less expensive and smaller than SRAMs.

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

- A DRAM cell is essentially an analog device using a capacitor; the capacitor can store any charge value within a range; a threshold value determines whether the charge is interpreted as 1 or 0. A SRAM cell is a digital device, in which binary values are stored using traditional flip-flop logic-gate configurations.

5.6. What are some applications for ROM?

- Microprogrammed control unit memory; library subroutines for frequently wanted functions; system programs; function tables.

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

- **EPROM** is read and written electrically; before a write operation, all the storage cells must be erased to the same initial state by exposure of the packaged chip to ultraviolet radiation. Erasure is performed by shining an intense ultraviolet light through a window that is designed into the memory chip.
- **EEPROM** is a read mostly memory that can be written into at any time without erasing prior contents; only the byte or bytes addressed are updated.
- **Flash Memory** is intermediate between EPROM and EEPROM in both cost and functionality. Like EEPROM, flash memory uses an electrical erasing technology. An entire flash memory can be erased in one or a few seconds, which is much faster than EPROM. In addition, it is possible to erase just blocks of memory rather than an entire chip. However, flash memory does not provide byte-level erasure. Like EPROM, flash memory uses only one transistor per bit, and so achieves the high density (compared with EEPROM) of EPROM.

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

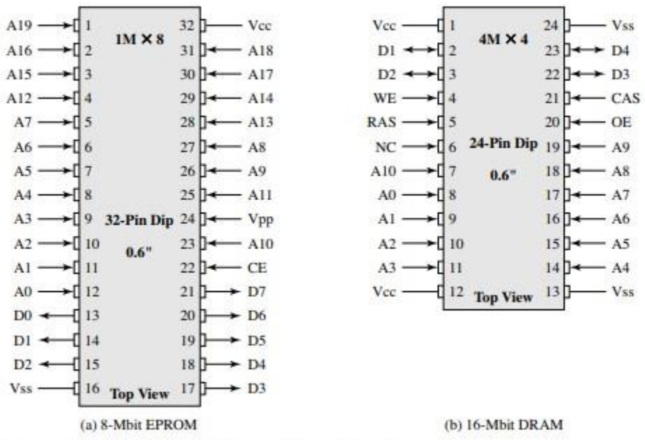

Figure 5.4 Typical Memory Package Pins and Signals

- A0 - A1 = address lines:. CAS = column address select:. D1 - D4 = data lines. NC: = no connect. OE: output enable. RAS = row address select:. Vcc: = voltage source. Vss: = ground. WE: write enable.

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?

- A **syndrome** is created by the XOR of the code in a word with a calculated version of that code. 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. If the syndrome contains all 0s, no error has been detected. If the syndrome contains one and only one bit set to 1, then an error has occurred in one of the 4 check bits. No correction is needed. If the syndrome contains more than one bit set to 1, then the numerical value of the syndrome indicates the position of the data bit in error. This data bit is inverted for correction.

5.11. How does SDRAM differ from ordinary DRAM?

- Unlike the traditional DRAM, which is asynchronous, the SDRAM exchanges data with the processor synchronized to an external clock signal and running at the full speed of the processor/memory bus without imposing wait states.

6. External Memory

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

- Improvement in the uniformity of the magnetic film surface to increase disk reliability. A significant reduction in overall surface defects to help reduce read/write errors. Ability to support lower fly heights (described subsequently). Better stiffness to reduce disk dynamics. Greater ability to withstand shock and damage

6.2. How are data written onto a magnetic disk?

- The write mechanism is based on the fact that electricity flowing through a coil produces a magnetic field. Pulses are sent to the write head, and magnetic patterns are recorded on the surface below, with different patterns for positive and negative currents. An electric current in the wire induces a magnetic field across the gap, which in turn magnetizes a small area of the recording medium. Reversing the direction of the current reverses the direction of the magnetization on the recording medium.

6.3. How are data read from a magnetic disk?

- The read head consists of a partially shielded magnetoresistive (MR) sensor. The MR material has an electrical resistance that depends on the direction of the magnetization of the medium moving under it. By passing a current through the MR sensor, resistance changes are detected as voltage signals.

6.4. Explain the difference between a simple CAV system and a multiple zoned recording system. •

For the 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 farther from the center containing more bits than zones closer to the center.

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

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

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.

- On a movable-head system, the time it takes to position the head at the track is known as **seek time**. 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 **rotational delay.** The sum of the seek time, if any, and the rotational delay equals the **access time,** which is the time it takes to get into position to read or write. Once the head is in position, the read or write operation is then performed as the sector moves under the head; this is the data transfer portion of the operation and the time for the transfer is the **transfer time.**

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.
- 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.

- **0: Non-redundant**
- **1: Mirrored;** every disk has a mirror disk containing the same data.
- **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.
- **3: Bit-interleaved parity;** similar to level 2 but instead of an error-correcting code, a simple parity bit is computed for the set of individual bits in the same position on all of the data disks.
- **4: Block-interleaved parity;** a bit-by-bit parity strip is calculated across corresponding strips on each data disk, and the parity bits are stored in the corresponding strip on the parity disk.
- **5: Block-interleaved distributed parity;** similar to level 4 but distributes the parity strips across all disks.
- **6: Blockinterleaved dual distributed parity;** two different parity calculations are carried out and stored in separate blocks on different disks.

6.10. Explain the term striped data.

- The disk is divided into strips; these strips 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?

- For RAID level 1, redundancy is achieved by having two identical copies of all data. For higher levels, redundancy is achieved by the use of error-correcting codes.

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

In a parallel access array, all member disks participate in the execution of every I/O request. Typically, the spindles of the individual drives are synchronized so that each disk head is in the same position on each disk at any given time. In an independent access array, each member disk operates independently, so that separate I/O requests can be satisfied in parallel.

6.13. What is the difference between CAV and CLV?

- For the **constant angular velocity (CAV)** system, the number of bits per track is constant. At a **constant linear velocity (CLV),** the disk rotates more slowly for accesses near the outer edge than for those near the center. Thus, the capacity of a track and the rotational delay both increase for positions nearer the outer edge of the disk.

6.14. What differences between a CD and a DVD account for the larger capacity of the latter?

1) Bits are packed more closely on a DVD. The spacing between loops of a spiral on a CD is 1.6 µm and the minimum distance between pits along the spiral is 0.834 µm. The DVD uses a laser with shorter wavelength and achieves a loop spacing of 0.74 µm and a minimum distance between pits of 0.4 µm. The result of these two improvements is about a seven-fold increase in capacity, to about 4.7 GB.
2) The DVD employs a second layer of pits and lands on top of the first layer A dual-layer DVD has a semireflective layer on top of the reflective layer, and by adjusting focus, the lasers in DVD drives can read each layer separately. This technique almost doubles the capacity of the disk, to about 8.5 GB. The lower reflectivity of the second layer limits its storage capacity so that a full doubling is not achieved.
3) The DVD-ROM can be two sided whereas data is recorded on only one side of a CD. This brings total capacity up to 17 GB.

6.15. Explain serpentine recording.

- The typical recording technique used in serial tapes is referred to as serpentine recording. In this technique, when data are being recorded, the first set of bits is recorded along the whole length of the tape. When the end of the tape is reached, the heads are repositioned to record a new track, and the tape is again recorded on its whole length, this time in the opposite direction. That process continues, back and forth, until the tape is full.

7. Input/Output

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

- **Human readable:** Suitable for communicating with the computer user.
- **Machine readable:** Suitable for communicating with equipment.
- **Communication:** Suitable for communicating with remote devices

7.2. What is the International Reference Alphabet?

- The most commonly used text code is the International Reference Alphabet (IRA), in which each character is represented by a unique 7-bit binary code; thus, 128 different characters can be represented.

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:** The processor issues I/O command, on behalf of a process, to I/O module; that process then busy-waits for the operation to be completed before proceeding.
- **Interrupt-driven I/O:** The processor issues I/O command on behalf of a process, continues to execute subsequent instructions, and is interrupted by the I/O module when the latter has completed its work. The subsequent instructions may be in the same process, if it is not necessary for that process to wait for the completion of the I/O. Otherwise, the process is suspended pending the interrupt and other work is performed.
- **Direct memory access (DMA):** A DMA module controls the exchange of data between main memory and an I/O module. The processor sends a request for the transfer of a block of data to the DMA module and is interrupted only after the entire block has been transferred.

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 devices. The processor treats the status and data registers of I/O modules as memory locations and uses the same machine instructions to access both memory and I/O devices.
- With **isolated I/O**, a command specifies whether the address refers to a memory location or an I/O device. The full range of addresses may be available for both.

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

- Four general categories of techniques are in common use: multiple interrupt lines; software poll; daisy chain (hardware poll, vectored); bus arbitration (vectored).

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 pauses for each bus cycle stolen by the DMA module.

8. Operating System Support

8.1. What is an Operating System?

- The Operating System (OS) is the software that controls the execution of programs on a processor and that manages the processor's resources.

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

- **Program creation:** The operating system provides a variety of facilities and services, such as editors and debuggers, to assist the programmer in creating programs.
- **Program execution:** A number of tasks need to be performed to execute a program. Instructions and data must be loaded into main memory, I/O devices and files must be initialized, and other resources must be prepared.
- **Access to I/O devices:** Each I/O device requires its own peculiar set of instructions or control signals for operation.
- **Controlled access to files:** In the case of files, control must include an understanding of not only the nature of the I/O device (disk drive, tape drive) but also the file format on the storage medium.
- **System access:** In the case of a shared or public system, the operating system controls access to the system as a whole and to specific system resources.
- **Error detection and response**: A variety of errors can occur while a computer system is running.
- **Accounting:** A good operating system will collect usage statistics for various resources and monitor performance parameters such as response time.

8.3. List and briefly define the major types of OS scheduling.

- **Long-term scheduling:** The decision to add to the pool of processes to be executed.
- **Medium-term scheduling:** The decision to add to the number of processes that are partially or fully in main memory**.**
- **Short-term scheduling:** The decision as to which available process will be executed by the processor

8.4. What is the difference between a process and a program?

- A process is a program in execution, together with all the state information required for execution.

8.5. What is the purpose of swapping?

- The purpose of swapping is to provide for efficient use of main memory for process execution.

8.6. If a process may be dynamically assigned to different locations in main memory, what is the implication for the addressing mechanism?

- Addresses must be dynamic in the sense that absolute addresses are only resolved during loading or execution.

8.7. Is it necessary for all of the pages of a process to be in main memory while the process is executing?

- No, if virtual memory is used.

8.8. Must the pages of a process in main memory be contiguous?

- No

8.9. Is it necessary for the pages of a process in main memory to be in sequential order? 8.10 What is the purpose of a translation lookaside buffer?

- No

8.10. What is the purpose of a translation lookaside buffer?

- The TLB is a cache that contains those page table entries that have been most recently used. Its purpose is to avoid, most of the time, having to go to disk to retrieve a page table entry.

CENTRAL PROCESSING UNIT

9. Computer Arithmetic

9.1. Briefly explain the following representations: sign magnitude, twos complement, biased.

- **Sign–Magnitude Representation:** In an N-bit word, the left-most bit is the sign (0 = positive, 1 = negative) and the remaining N – 1 bits comprise the magnitude of the number.
- **Twos Complement Representation:** A positive integer is represented as in sign magnitude. A negative number is represented by taking the Boolean complement of each bit of the corresponding positive number, then adding 1 to the resulting bit pattern viewed as an unsigned integer.
- **Biased representation:** A fixed value, called the bias, is added to the integer.

9.2. Explain how to determine if a number is negative in the following representations: sign magnitude, twos complement, biased.

- In sign-magnitude and twos complement, the left-most bit is a sign bit. In biased representation, a number is negative if the value of the representation is less than the bias.

9.3. What is the sign-extension rule for twos complement numbers?

- Add additional bit positions to the left and fill in with the value of the original sign bit.

9.4. How can you form the negation of an integer in twos complement representation?

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

9.5. In general terms, when does the twos complement operation on an n-bit integer produce the same integer?

- When the operation is performed on the n-bit integer -2^{n-1} (one followed by n – 1 zeros).

9.6. What is the difference between the twos complement representation of a number and the twos complement of a number?

- The twos complement representation of a number is the bit pattern used to represent an integer. The twos complement of a number is the operation that computes the negation of a number in twos complement representation.

9.7. If we treat 2 twos complement numbers as unsigned integers for purposes of addition, the result is correct if interpreted as a twos complement number. This is not true for multiplication. Why?

- The algorithm for performing twos complement addition involves simply adding the two numbers in the same way as for ordinary addition for unsigned numbers, with a test for overflow. For multiplication, if we treat the bit patterns as unsigned numbers, their magnitude is different from the twos complement versions and so the magnitude of the result will be different.

9.8. What are the four essential elements of a number in floating-point notation?

- Sign, significand, exponent, base.

9.9. What is the benefit of using biased representation for the exponent portion of a floating-point number?

- An advantage of biased representation is that nonnegative floating-point numbers can be treated as integers for comparison purposes.

9.10. What are the differences among positive overflow, exponent overflow, and significand overflow?

- **Positive overflow** refers to integer representations and refers to a number that is larger than can be represented in a given number of bits.
- **Exponent overflow** refers to floating point representations and refers to a positive exponent that exceeds the maximum possible exponent value.
- **Significand overflow** occurs when the addition of two significands of the same sign result in a carry out of the most significant bit.

9.11. What are the basic elements of floating-point addition and subtraction? 1) Check for zeros.
2) Align the significands.
3) Add or subtract the significands.
4) Normalize the result.

9.12. Give a reason for the use of guard bits.

- To avoid unnecessary loss of the least significant bit.

9.13. List four alternative methods of rounding the result of a floating-point operation.

- **Round to nearest:** The result is rounded to the nearest representable number.
- **Round toward +∞:** The result is rounded up toward plus infinity.
- **Round toward – ∞:** The result is rounded down toward negative infinity.
- **Round toward 0:** The result is rounded toward zero.

10. Instruction Sets: Characteristics and Functions

10.1. What are the typical elements of a machine instruction?

- The essential elements of a computer instruction are the **opcode**, which specifies the operation to be performed, the **source** and **destination operand references**, which specify the input and output locations for the operation, and a **next instruction reference**, which is usually implicit.

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

- Registers and memory.

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

- Two operands, one result, and the address of the next instruction.

10.4. List and briefly explain five important instruction set design issues.

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

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

- Addresses, numbers, characters, logical data.

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

- For the IRA bit pattern 011XXXX, the digits 0 through 9 are represented by their binary equivalents, 0000 through 1001, in the right-most 4 bits. This is the same code as packed decimal.

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

- With a **logical shift,** the bits of a word are shifted left or right. On one end, the bit shifted out is lost. On the other end, a 0 is shifted in. The **arithmetic shift** operation treats the data as a signed integer and does not shift the sign bit. On a right arithmetic shift, the sign bit is replicated into the bit position to its right. On a left arithmetic shift, a logical left shift is performed on all bits but the sign bit, which is retained.

10.8. Why are transfer of control instructions needed?

1) In the practical use of computers, it is essential to be able to execute each instruction more than once and perhaps many thousands of times. It may require thousands or perhaps millions of instructions to implement an application. This would be unthinkable if each instruction had to be written out separately. If a table or a list of items is to be processed, a program loop is needed. One sequence of instructions is executed repeatedly to process all the data.
2) Virtually all programs involve some decision making. We would like the computer to do one thing if one condition holds, and another thing if another condition holds.
3) To compose correctly a large or even medium-size computer program is an exceedingly difficult task. It helps if there are mechanisms for breaking the task up into smaller pieces that can be worked on one at a time.

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

- First, most machines provide a 1-bit or multiple-bit condition code that is set as the result of some operations. Another approach that can be used with a three-address instruction format is to perform a comparison and specify a branch in the same instruction.

10.10. What is meant by the term nesting of procedures?

- The term refers to the occurrence of a procedure call inside a procedure.

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

- Register, start of procedure, top of stack.

10.12. What is a reentrant procedure?

- A reentrant procedure is one in which it is possible to have several calls open to it at the same time.

10.13. What is reverse Polish notation?

- In this notation, the operator follows its two operands.

10.14. What is the difference between big endian and little endian?

- A multibyte numerical value stored with the most significant byte in the lowest numerical address is stored in big-endian fashion. A multibyte numerical value stored with the most significant byte in the highest numerical address is stored in little-endian fashion.

11. Instruction Sets: Addressing Modes and Formats

11.1. Briefly define immediate addressing.

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

11.2. Briefly define direct addressing.

- Direct addressing: The address field contents the effective address of the operand.

11.3. Briefly define indirect addressing.

- Indirect addressing: The address field refers to the address of a word in memory, which in turn contains the effective address of the operand.

11.4. Briefly define register addressing.

- Register addressing: The address field refers to a register that contains the operand.

11.5. 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.

11.6. 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.

11.7. 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.

11.8. 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.

11.9. 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.

11.10. What facts go into determining the use of the addressing bits of an instruction?

- **Number of addressing modes:** Sometimes an addressing mode can be indicated implicitly. In other cases, the addressing modes must be explicit, and one or more mode bits will be needed.
- **Number of operands:** Typical instructions on today's machines provide for two operands. Each operand address in the instruction might require its own mode indicator, or the use of a mode indicator could be limited to just one of the address fields.
- **Register versus memory:** The more that registers can be used for operand references, the fewer bits are needed.
- **Number of register sets**: One advantage of using multiple register sets is that, for a fixed number of registers, a functional split requires fewer bits to be used in the instruction.

- **Address range:** For addresses that reference memory, the range of addresses that can be referenced is related to the number of address bits. Because this imposes a severe limitation, direct addressing is rarely used. With displacement addressing, the range is opened up to the length of the address register.
- **Address granularity:** In a system with 16- or 32-bit words, an address can reference a word or a byte at the designer's choice. Byte addressing is convenient for character manipulation but requires, for a fixed-size memory, more address bits.

11.11. What are the advantages and disadvantages of using a variable-length instruction

- **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.

Processor Structure and Function

What general roles are performed by processor registers?

- **User-visible registers:** These enable the machine- or assembly language programmer to minimize mainmemory 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.

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

- General purpose; Data; Address; Condition codes

12.3. What is the function of condition codes?

- **Condition codes** are bits set by the CPU hardware as the result of operations. For example, an arithmetic operation may produce a positive, negative, zero, or overflow result. In addition to the result itself being stored in a register or memory, a condition code is also set. The code may subsequently be tested as part of a conditional branch operation.

12.4. 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.

12.5. 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.

12.6. 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.

12.7. How are history bits used for branch prediction?

- One or more bits that reflect the recent history of the instruction can be associated with each conditional branch instruction. 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.

Reduced Instruction Set Computers (RISC)

What are some typical distinguishing characteristics of RISC organization?

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

13.2. 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.

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

1) 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

13.4. 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.

13.5. 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.

Instruction-Level Parallelism and Superscalar Processors

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

- A **superscalar processor** is one in which multiple independent instruction pipelines are used. Each pipeline consists of multiple stages, so that each pipeline can handle multiple instructions at a time. Multiple pipelines introduce a new level of parallelism, enabling multiple streams of instructions to be processed at a time.

14.2. What is the difference between the superscalar and superpipelined approaches?

- Superpipelining exploits the fact that many pipeline stages perform tasks that require less than half a clock cycle. Thus, a doubled internal clock speed allows the performance of two tasks in one external clock cycle.

14.3. What is instruction-level parallelism?

- Instruction-level parallelism refers to the degree to which the instructions of a program can be executed in parallel.

14.4. Briefly define the following terms: True data dependency, Procedural dependency, Resource conflicts, Output dependency, Antidependency.

- **True data dependency:** A second instruction needs data produced by the first instruction.
- **Procedural dependency:** The instructions following a branch (taken or not taken) have a procedural dependency on the branch and cannot be executed until the branch is executed.
- **Resource conflicts:** A resource conflict is a competition of two or more instructions for the same resource at the same time.
- **Output dependency:** Two instructions update the same register, so the later instruction must update later.
- **Antidependency:** A second instruction destroys a value that the first instruction uses.

14.5. What is the distinction between instruction-level parallelism and machine parallelism?

- **Instruction-level parallelism** exists when instructions in a sequence are independent and thus can be executed in parallel by overlapping.
- **Machine parallelism** is a measure of the ability of the processor to take advantage of instruction-level parallelism. Machine parallelism is determined by the number of instructions that can be fetched and executed at the same time (the number of parallel pipelines) and by the speed and sophistication of the mechanisms that the processor uses to find independent instructions.

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

- **In-order issue with in-order completion:** Issue instructions in the exact order that would be achieved by sequential execution and to write results in that same order.
- **In-order issue with out-of-order completion:** Issue instructions in the exact order that would be achieved by sequential execution but allow instructions to run to completion out of order.
- **Out-of-order issue with out-of-order completion:** The processor has a lookahead capability, allowing it to identify independent instructions that can be brought into the execute stage. Instructions are issued with little regard for their original program order. Instructions may also run to completion out of order.

14.7. What is the purpose of an instruction window?

- For an out-of-order issue policy, the instruction window is a buffer that holds decoded instructions. These may be issued from the instruction window in the most convenient order.

14.8. What is register renaming and what is its purpose?

- Registers are allocated dynamically by the processor hardware, and they are associated with the values needed by instructions at various points in time. When a new register value is created (i.e., when an instruction executes that has a register as a destination operand), a new register is allocated for that value.

14.9. What are the key elements of a superscalar processor organization?

1) Instruction fetch strategies that simultaneously fetch multiple instructions, often by predicting the outcomes of, and fetching beyond, conditional branch instructions. These functions require the use of multiple pipeline fetch and decode stages, and branch prediction logic.
2) Logic for determining true dependencies involving register values, and mechanisms for communicating these values to where they are needed during execution.

3) Mechanisms for initiating, or issuing, multiple instructions in parallel.
4) Resources for parallel execution of multiple instructions, including multiple pipelined functional units and memory hierarchies capable of simultaneously servicing multiple memory references.
5) Mechanisms for committing the process state in correct order.

Control Unit Operation

15.1. Explain the distinction between the written sequence and the time sequence of an instruction. •

The operation of a computer, in executing a program, consists of a sequence of **instruction cycles**, with one machine instruction per cycle. This sequence of instruction cycles is not necessarily the same as the **written sequence** of instructions that make up the program, because of the existence of branching instructions. The actual execution of instructions follows a **time sequence** of instructions.

15.2. What is the relationship between instructions and micro-operations?

- A **micro-operation** is an elementary CPU operation, performed during one clock pulse.
- An **instruction** consists of a sequence of micro-operations.

15.3. What is the overall function of a processor's control unit?

- The control unit of a processor performs two tasks:
  1) It causes the processor to execute micro-operations in the proper sequence, determined by the program being executed, and
  2) it generates the control signals that cause each micro-operation to be executed.

15.4. Outline a three-step process that leads to a characterization of the control unit.

1) Define the basic elements of the processor.
2) Describe the micro-operations that the processor performs.
3) Determine the functions that the control unit must perform to cause the micro-operations to be performed.

15.5. What basic tasks does a control unit perform?

- **Sequencing:** The control unit causes the processor to step through a series of micro-operations in the proper sequence, based on the program being executed.
- **Execution:** The control unit causes each micro-operation to be performed.

15.6. Provide a typical list of the inputs and outputs of a control unit. **The**

inputs are:

- **Clock**: This is how the control unit "keeps time." The control unit causes one micro-operation (or a set of simultaneous micro-operations) to be performed for each clock pulse. This is sometimes referred to as the processor cycle time, or the clock cycle time.
- **Instruction register:** The opcode of the current instruction is used to determine which micro-operations to perform during the execute cycle.
- **Flags:** These are needed by the control unit to determine the status of the processor and the outcome of previous ALU operations.
- **Control signals from control bus:** The control bus portion of the system bus provides signals to the control unit, such as interrupt signals and acknowledgments.

The outputs are:

- **Control signals within the processor**: These are two types: those that cause data to be moved from one register to another, and those that activate specific ALU functions.
- **Control signals to control bus:** These are also of two types: control signals to memory, and control signals to the I/O modules.

15.7. List three types of control signals.

1) Those that activate an ALU function.
2) Those that activate a data path.
3) Those that are signals on the external system bus or other external interface

15.8. Briefly explain what is meant by a hardwired implementation of a control unit.

- In a **hardwired implementation**, the control unit is essentially a combinatorial circuit. Its input logic signals are transformed into a set of output logic signals, which are the control signals.

Microprogrammed Control

16.1. What is the difference between a hardwired implementation and a microprogrammed implementation of a control unit?

- A **hardwired control unit** is a combinatorial circuit, in which input logic signals are transformed into a set of output logic signals that function as the control signals. In a microprogrammed control unit, the logic is specified by a microprogram.
- A **microprogram** consists of a sequence of instructions in a microprogramming language. These are very simple instructions that specify micro-operations.

16.2. How is a horizontal microinstruction interpreted?

1) To execute a microinstruction, turn on all the control lines indicated by a 1 bit; leave off all control lines indicated by a 0 bit. The resulting control signals will cause one or more micro-operations to be performed.
2) If the condition indicated by the condition bits is false, execute the next microinstruction in sequence.
3) If the condition indicated by the condition bits is true, the next microinstruction to be executed is indicated in the address field.

16.3. What is the purpose of a control memory?

- The control memory contains the set of microinstructions that define the functionality of the control unit.

16.4. What is a typical sequence in the execution of a horizontal microinstruction?

- The microinstructions in each routine are to be executed sequentially. Each routine ends with a branch or jump instruction indicating where to go next.

16.5. What is the difference between horizontal and vertical microinstructions?

- In a **horizontal microinstruction** every bit in the control field attaches to a control line. In a **vertical microinstruction**, a code is used for each action to be performed and the decoder translates this code into individual control signals.

16.6. What are the basic tasks performed by a microprogrammed control unit?

- **Microinstruction sequencing:** Get the next microinstruction from the control memory.
- **Microinstruction execution:** Generate the control signals needed to execute the microinstruction.

16.7. What is the difference between packed and unpacked microinstructions?

- The degree of packing relates to the degree of identification between a given control task and specific microinstruction bits. As the bits become more **packed**, a given number of bits contains more information.
  • An **unpacked** microinstruction has no coding beyond assignment of individual functions to individual bits.

16.8. What is the difference between hard and soft microprogramming?

- **Hard microprograms** are generally fixed and committed to read-only memory.
- **Soft microprograms** are more changeable and are suggestive of user microprogramming.

16.9. What is the difference between functional and resource encoding?

- Two approaches can be taken to organizing the encoded microinstruction into fields: **functional** and **resource**. The **functional encoding** method identifies functions within the machine and designates fields by function type. For example, if various sources can be used for transferring data to the accumulator, one field can be designated for this purpose, with each code specifying a different source.
- **Resource encoding** views the machine as consisting of a set of independent resources and devotes one field to each (e.g., I/O, memory, ALU).

16.10. List some common applications of microprogramming.

- Realization of computers. Emulation. Operating system support. Realization of special-purpose devices. Highlevel language support. Microdiagnostics. User Tailoring.

Parallel Processing

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?

1) There are two or more similar processors of comparable capability.
2) 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.
3) All processors share access to I/O devices, either through the same channels or through different channels that provide paths to the same device.
4) All processors can perform the same functions (hence the term symmetric).
5) 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?

- **Software cache coherence schemes** attempt to avoid the need for additional hardware circuitry and logic by relying on the compiler and operating system to deal with the problem.
- In **hardware schemes,** the cache coherence logic is implemented in hardware.

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

- **Modified:** The line in the cache has been modified (different from main memory) and is available only in this cache.
- **Exclusive:** The line in the cache is the same as that in main memory and is not present in any other cache.
- **Shared:** The line in the cache is the same as that in main memory and may be present in another cache.
- **Invalid:** The line in the cache does not contain valid data.

17.7. What are some of the key benefits of clustering?

- **Absolute scalability**: It is possible to create large clusters that far surpass the power of even the largest standalone machines.
- **Incremental scalability:** A cluster is configured in such a way that it is possible to add new systems to the cluster in small increments. Thus, a user can start out with a modest system and expand it as needs grow, without having to go through a major upgrade in which an existing small system is replaced with a larger system.
- **High availability:** Because each node in a cluster is a standalone computer, the failure of one node does not mean loss of service.

- **Superior price/performance:** By using commodity building blocks, it is possible to put together a cluster with equal or greater computing power than a single large machine, at much lower cost.

17.8. What is the difference between failover and failback?

- The function of switching an applications and data resources over from a failed system to an alternative system in the cluster is referred to as **failover**.
- A related function is the restoration of applications and data resources to the original system once it has been fixed; this is referred to as **failback**.

17.9. What are the differences among UMA, NUMA, and CC-NUMA?

- **Uniform Memory Access (UMA):** All processors have access to all parts of main memory using loads and stores. The memory access time of a processor to all regions of memory is the same. The access times experienced by different processors are the same.
- **Nonuniform Memory Access (NUMA):** All processors have access to all parts of main memory using loads and stores. The memory access time of a processor differs depending on which region of main memory is accessed. The last statement is true for all processors; however, for different processors, which memory regions are slower and which are faster differ.
- **Cache-coherent NUMA (CC-NUMA):** A NUMA system in which cache coherence is maintained among the caches of the various processors.

18. Multicore Computers

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 given an increasing fraction of chip area to cache memory?

- Cache memory uses less power than logic.

18.4. List some examples of applications that benefit directly from the ability to scale throughput with the number of cores.

- **Multi-threaded native applications:** Multi-threaded applications are characterized by having a small number of highly threaded processes. Examples of threaded applications include Lotus Domino or Siebel CRM (Customer Relationship Manager).
- **Multi-process applications:** Multi-process applications are characterized by the presence of many singlethreaded processes. Examples of multi-process applications include the Oracle database, SAP, and PeopleSoft.
- **Java applications:** Java applications embrace threading in a fundamental way. Not only does the Java language greatly facilitate multithreaded applications, but the Java Virtual Machine is a multi-threaded process that provides scheduling and memory management for Java applications. Java applications that can benefit directly from multicore resources include application servers such as Sun's Java Application Server, BEA's Weblogic, IBM's Websphere, and the open-source Tomcat application server. All applications that use a Java 2 Platform, Enterprise; Edition (J2EE platform) application server can immediately benefit from multicore technology.
- **Multi-instance applications:** Even if an individual application does not scale to take advantage of a large number of threads, it is still possible to gain from multicore architecture by running multiple instances of the application in parallel. If multiple application instances require some degree of isolation, virtualization technology (for the hardware of the operating system) can be used to provide each of them with its own separate and secure environment.

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.

1) 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.
2) A related advantage is that data shared by multiple cores is not replicated at the shared cache level.
3) 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.
4) Interprocessor communication is easy to implement, via shared memory locations.
5) The use of a shared L2 cache confines the cache coherency problem to the L1 cache level, which may provide some additional performance advantage.