Computer Architecture and Organization

Kushagra Lakhwani (2021UCI8036)

2022-12-04

Assignment 1

Question 1

1. **Differentiate between "hit" and "miss" with respect to cache memory.**

   Caching is one of the most vital mechanisms for improving site performance, frequent cache misses will increase data access time, resulting in a poor user experience and high bounce rates.

   - **Hit** : When a request is made to the cache, and the data is present in the cache, it is called a cache hit.
   - **Miss** : When a request is made to the cache, and the data is not present in the cache, it is called a cache miss.

   For a more effective caching system, the hit ratio should be higher than the miss rate. One of the best ways to achieve that is to reduce cache misses.

2. **How interrupts are handled? Explain.**

   An interrupt is a signal to the processor emitted by hardware or software indicating an event that needs immediate attention. The processor responds by suspending its current activities, saving its state, and executing a function called an interrupt handler (or an interrupt service routine, or an ISR) to deal with the event. This interruption is temporary, and, after the interrupt handler finishes, the processor resumes normal activities.

3. **Explain little endian and big endian data storage mechanisms.**

   In little endian, the least significant byte is stored at the lowest address.

   In big endian, the most significant byte is stored at the lowest address.

4. **Express $A * B + (B * D + C \ \wedge E)$ into reverse polish notation.**

   $AB * BD * CE \wedge ++$

5. **Differentiate between direct and indirect instruction.**

   In direct addressing, the operand is specified in the instruction itself. In indirect addressing, the operand is specified by the contents of a register.

   In direct addressing, the operand is specified in the instruction itself. In indirect addressing, the operand is specified by the contents of a register.

   For example, in the instruction `MOV R1, R2`, the operand is specified in the instruction itself. In the instruction `MOV R1, (R2)`, the operand is specified by the contents of the register `R2`.

6. **Briefly list the types of interrupts with the help of suitable examples.**

- **Hardware Interrupts:** The interrupt signal generated from external devices and i/o devices are made interrupt to CPU when the instructions are ready.

  For example − In a keyboard if we press a key to do some action this pressing of the keyboard generates a signal that is given to the processor to do action, such interrupts are called hardware interrupts.

  Hardware interrupts are classified into two types which are as follows −

  - Maskable Interrupt − The hardware interrupts that can be delayed when a highest priority interrupt has occurred to the processor.
  - Non Maskable Interrupt − The hardware that cannot be delayed and immediately be serviced by the processor.

- **Software Interrupts:** The interrupt signal generated from internal devices and software programs need to access any system call then software interrupts are present.

  Software interrupt is divided into two types. They are as follows −

  - Normal Interrupts − The interrupts that are caused by the software instructions are called software instructions.
  - Exception − Exception is nothing but an unplanned interruption while executing a program. For example − while executing a program if we got a value that is divided by zero is called an exception.

7. **Differentiate between RISC and CISC.**

RISC	CISC
- Reduced Instruction Set Computer	Complex Instruction Set Computer
- A large number of instructions are present in the architecture.	Very few instructions are present. The number of instructions is generally less than 100.
- Some instructions with long execution times. These include instructions that copy an entire block from one part of memory to another and others that copy multiple registers to and from memory.	No instruction with a long execution time due to a very simple instruction set. Some early RISC machines did not even have an integer multiply instruction, requiring compilers to implement multiplication as a sequence of additions.
- Variable-length encodings of the instructions. ***Example***: IA32 instruction size can range from 1 to 15 bytes.	Fixed-length encodings of the instructions are used. **Example**: In IA32, generally all instructions are encoded as 4 bytes.
- Multiple formats are supported for specifying operands. A memory operand specifier can have many different combinations of displacement, base, and index register.	Simple addressing formats are supported. Only base and displacement addressing is allowed.
- CISC supports array.	RISC does not support an array.
- Arithmetic and logical operations can be applied to both memory and register operands.	Arithmetic and logical operations only use register operands. Memory referencing is only allowed by loading and storing instructions, i.e. reading from memory into a register and writing from a register to memory respectively.
- Implementation programs are hidden from machine-level programs. The ISA provides a clean abstraction between programs and how they get executed.	Implementation programs exposed to machine-level programs. Few RISC machines do not allow specific instruction sequences.
- Condition codes are used.	No condition codes are used.

RISC	CISC
- The stack is being used for procedure arguments and returns addresses.	Registers are being used for procedure arguments and return addresses. Memory references can be avoided by some procedures.

8. **Explain updating techniques used in cache design.**

   - **Write-through:** In write-through, the data is written to both the cache and the main memory. This is the simplest method of updating the cache. It is also the fastest method of updating the cache. However, it is the least efficient method of updating the cache. This is because the data is written to both the cache and the main memory. This increases the number of memory accesses and hence the number of memory cycles.
   - **Write-back:** In write-back, the data is written to the cache only. The data is written to the main memory only when the cache block is replaced. This is the most efficient method of updating the cache. However, it is the slowest method of updating the cache. This is because the data is written to the main memory only when the cache block is replaced. This increases the number of cache misses and hence the number of memory cycles.
   - **Write-allocate:** In write-allocate, the data is written to the cache only. The data is written to the main memory only when the cache block is replaced. This is the most efficient method of updating the cache. However, it is the slowest method of updating the cache. This is because the data is written to the main memory only when the cache block is replaced. This increases the number of cache misses and hence the number of memory cycles.
   - **Write-around:** In write-around, the data is written to the main memory only. The data is not written to the cache. This is the least efficient method of updating the cache. However, it is the fastest method of updating the cache. This is because the data is written to the main memory only. This reduces the number of memory accesses and hence the number of memory cycles.
   - **Write-protect:** In write-protect, the data is written to the cache only. The data is not written to the main memory. This is the second most efficient method of updating the cache. However, it is the second fastest method of updating the cache. This is because the data is written to the cache only. This reduces the number of memory accesses and hence the number of memory cycles.

9. **Explain Virtual memory. Also state "Locality of Reference" principle.**

   Virtual memory is a memory management technique that allows a computer to execute programs that are larger than the physical memory available. It does so by using a portion of the hard disk as virtual memory. The operating system swaps out portions of the program that are not currently being used to the hard disk and swaps them back in when they are needed.

   By doing this, the degree of multiprogramming will be increased and therefore, the CPU utilization will also be increased.

   The locality of reference principle states that a program tends to access the same memory locations repeatedly. This is because the program is accessing the same data structures repeatedly. The locality of reference principle is used to implement virtual memory. The operating system uses the locality of reference principle to determine which portions of the program to swap out to the hard disk.

10. **What are different kinds of operations used in CPU design?**

    - **Arithmetic and Logic Operations:** These operations are used to perform arithmetic and logical operations on the data. These operations are used to perform arithmetic and logical operations on the data. These operations are used to perform arithmetic and logical operations on the data. These operations are used to perform arithmetic and logical operations on the data.
    - **Data Transfer Operations:** These operations are used to transfer data between the registers and the memory. These operations are used to transfer data between the registers and the memory.

These operations are used to transfer data between the registers and the memory. These operations are used to transfer data between the registers and the memory.

- **Control Operations:** These operations are used to control the flow of the program.

Question 2:

1. **Starting from initial value `R = 0b11011101`, determine the sequence of binary values in R after a logical shift left, followed by a circular shift right, followed by a logical shift right and a circular shift.**
   - Logical Shift Left: `R = 0b11011101` → `R = 0b10111010`
   - Circular Shift Right: `R = 0b10111010` → `R = 0b11011101`
   - Logical Shift Right: `R = 0b11011101` → `R = 0b01101110`
   - Circular Shift Left: `R = 0b01101110` → `R = 0b11011101`
2. **Draw and explain the common bus system for the simple computer. Also, draw the various registers, types of bus (eg. Data bus, control bus, address bus and where they are being used). Also, explain LOAD, Selection and Fetch of data in those registers using appropriate diagram.**

   The common bus is required in the computer for communication with registers and memory to decrease the hardware complexity.

   The basic computer has eight registers, a memory unit and control unit. The path is required to transfer data, address and control signal. If there is a wire for every operation and every address of memory location, then it will be excessive. So common bus system is implemented through use of MUX and other circuit for proper implementation of wire.

Registers	Function
Instructions Registers (IR)	Hold instruction code of instruction currently executing
Address Registers (AR)	All memory references are initiated by loading the memory address in AR
Temporary Registers (TR)	Extra registers used to store data and address
Input Registers (IR)	Hold the data from input devices
Output Registers (OR)	Hold the data to send to output devices
Program Counter (PC)	Holds the address of next instruction to be executed
Accumulator (AC)	Store results produced by system
Data Registers (DR)	Temporarily store data being transmitted to or from peripheral devices

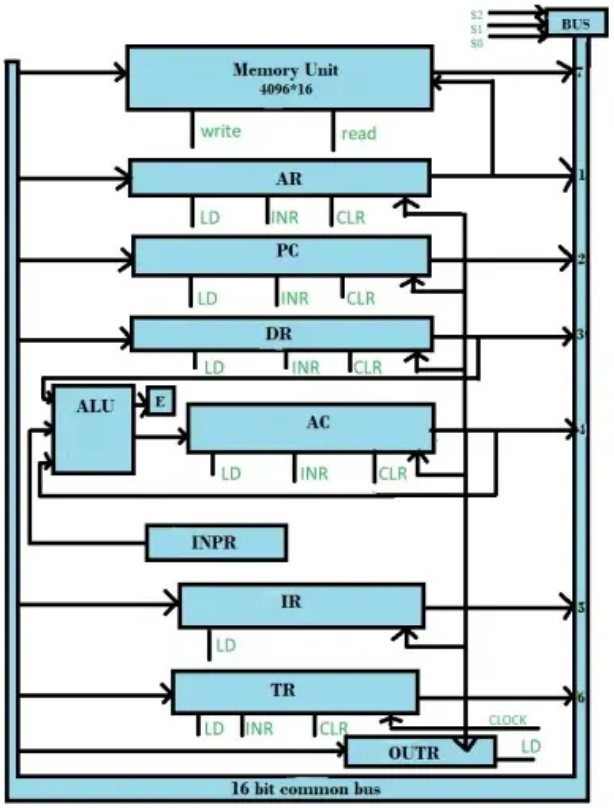

Figure 1: Common Bus System

Three control lines S_2, S_1, S_0 control which registers the bus selects as its input. Either one of the registers will have its load signal activated, or the memory will have its read signal activated.

S_2	S_1	S_0	Function
0	0	0	x
0	0	1	Load AR
0	1	0	Load PC
0	1	1	Load DR
1	0	0	Load AC
1	0	1	Load IR
1	1	0	Load TR
1	1	1	Memory

- Six registers and a memory are connected to a bus.
- The input register INPR and OUTR HAS 8 bit each.
- The seven registers, memory, INPR, and OUTR are driven by a single phase clock pulse.
- The particular register whose load (LD) input is enabled receives the data from the bus during the next clock pulse.
- Five registers have three control inputs: load(LD), Increment(INR), and clear(CLR). Two registers IR and OUTR have only LD inputs.
- The result is transferred to AC and end carry is transferred to flip-flop E.

Question 3

1. **Convert the decimal 61.5867 into its binary equivalent.**

$61.5867_{10} = 111101.100101100011000_2$

2. **Convert -6,75 (written in decimal) to floating point representation (single precision).**

   $-6.75_{10} = -110.11_2$

3. **What do you mean by bus arbitration? Explain serial and parallel bus arbitration in detail.**

   - A device that initiates data transfers on the bus at any given time is called a bus master.
   - In a computer system, there may be more than one bus master such as a DMA controller or a processor etc.
   - These devices share the system bus and when a current master bus relinquishes another bus can acquire the control of the processor.
   - Bus arbitration is a process by which next device becomes the bus controller by transferring bus mastership to another bus.

   There are two types of bus arbitration:

   - Centralized bus arbitration

     A single bus arbiter performs the required arbitration.

     There are three arbitration schemes which run on centralized arbitration.

     - Daisy Chaining − It is a simple and cheaper method where all the masters use the same line for making bus requests.
     - Polling Method − In this method, the controller is used to generate address lines for the master. For example, if there are 8 masters connected in a system at least 3 address lines are required.
     - Independent Request − In this scheme, each bus has its own bus request and a grant. The built-in priority decoder selects the highest priority requests and asserts the system.

   - Distributed bus arbitration
     - All devices participating in the selection of the next bus master.
     - Here, all the devices participate in the selection of the next bus master.
     - Each device on the bus is assigned a4 bit identification number.
     - When one or more devices request control of the bus, they assert the start arbitration signal and place their 4-bit identification numbers on arbitration lines through ARB3.
     - Each device compares the code and changes its bit position accordingly.
     - It does so by placing a 0 at the input of their drive.
     - The distributed arbitration is highly reliable because the bus operations are not dependant on devices.

   Arbitration techinques can also be classified as:

   - Static Arbitration Techniques

     In this technique, the priority assigned is fixed.

     It has two types.

     - Serial Arbitration

       It is also known as Daisy chain Arbitration. It is obtained by the daisy-chain connection of bus arbitration circuits. The scheme got the term from the structure of the grant line, which

chains through each device from the higher to lowest priority. The highest priority device will pass the grant line to the lower priority device only if it does not want it. Then the priority is forwarded to the next in the sequence. All devices use the same line for bus requests.If a busy bus line returns to its idle state, the most high-priority arbiter enables the busy line, and its corresponding processor can then run the required bus transfer.

Advantage

* It is a simple design.
* Less number control lines are used.

Disadvantage

* Priority depends on the physical location of the device
* Propagation delay due to serially granting of bus
* Failure of one of the devices may fail the entire system
* Cannot assure fairness- a low priority device may be locked out indefinitely

– Parallel Arbitration

It uses an external priority encoder and decoder. Each bus arbiter has a bus request output line and a bus acknowledge input line. Each arbiter enables request lines when its processor is requesting the system bus. The one with the highest priority determined by the output of the decoder gets access to the bus.

- Dynamic Arbitration Techniques

Serial and Parallel bus arbitration are static since the priorities assigned are fixed. In dynamic arbitration, priorities of the system change while the system is in operation.

The various algorithms used are:-

– Time Slice

It allocates a fixed-length time slice of bus time offered sequentially to each process in a round-robin fashion. The service provided by each system component and the system is independent of its location near the bus.

– Polling

In polling, the controller generates the addresses for the devices. The number of address lines needed depends upon the number of connected devices to the system. The controller generates a sequence of device addresses in response to the bus request. When the requesting device recognizes its address, it activates the busy bus line and uses the bus. After several bus cycles, the polling process continues by choosing a different processor. The polling sequence usually is programmable, and as a result, the selection priority can be altered under program control.

Question 4

1. **Explain instruction cycles (fetch) and (decode) in detail. Also explain the working of computer registers used in it.**

The execution of instructions define the instruction cycle. This is the thorough methodology computer processors use for executing a given instruction. Many times processors can be compared to combustion engines. Both follow a process continuously being carried out to fetch the desired outcome. Every processor shows a three-step instruction cycle. These three steps of the instruction execution cycle are,

1.Fetch: The processor copies the instruction data captured from the RAM.

2. Decode: Decoded captured data is transferred to the unit for execution.

3. Execute:Instruction is finally executed. The result is then registered in the processor or RAM (memory address).

First step: Fetch (instruction cycle)

According to the execution instruction definition, the instruction cycle's first step is to capture or fetch the instruction. This instruction in the fetch stage is captured from RAM. This memory is assigned to the processor through various units and registers; they are:

Second step: Decode (instruction cycle)

There are various instructions, and we can never be sure which instruction belongs to which execution unit. Decoding sorts this out. A decoder is responsible for taking in the instruction and decoding it to assign the respective execution unit to complete the execution instruction cycle.

Third step: Execute (instruction cycle)

The last stage of the execute instruction definition is to execute. It involves executing the given instruction that was fetched at the first stage. No two instructions ever get resolved in the same manner because their ways of utilising the hardware depend on their functions. There are four types of instructions that are generally present,

2. **For the expression X=(A+B)*(C+D) when evaluated on stack machine how many number of machine instructions are required?**

Zero Address Instructions –

A stack-based computer does not use the address field in the instruction. To evaluate an expression first it is converted to reverse Polish Notation i.e. Postfix Notation.

2. One Address Instructions –

This uses an implied ACCUMULATOR register for data manipulation. One operand is in the accumulator and the other is in the register or memory location. Implied means that the CPU already knows that one operand is in the accumulator so there is no need to specify it.

3. Two Address Instructions –

This is common in commercial computers. Here two addresses can be specified in the instruction. Unlike earlier in one address instruction, the result was stored in the accumulator, here the result can be stored at different locations rather than just accumulators, but require more number of bit to represent address. Here destination address can also contain operand.

4. Three Address Instructions –

This has three address field to specify a register or a memory location. Program created are much short in size but number of bits per instruction increase. These instructions make creation of program much easier but it does not mean that program will run much faster because now instruction only contain more information but each micro operation (changing content of register, loading address in address bus etc.) will be performed in one cycle only.

6 operations would be performed it can be calculated using stack polish notation

Question 6

1. **Explain instruction formats with examples.**

Zero Address Instructions –

A stack-based computer does not use the address field in the instruction. To evaluate an expression first it is converted to reverse Polish Notation i.e. Postfix Notation.

2 .One Address Instructions –

This uses an implied ACCUMULATOR register for data manipulation. One operand is in the accumulator and the other is in the register or memory location. Implied means that the CPU already knows that one operand is in the accumulator so there is no need to specify it.

3. Two Address Instructions –

This is common in commercial computers. Here two addresses can be specified in the instruction. Unlike earlier in one address instruction, the result was stored in the accumulator, here the result can be stored at different locations rather than just accumulators, but require more number of bit to represent address. Here destination address can also contain operand.

4. Three Address Instructions –

This has three address field to specify a register or a memory location. Program created are much short in size but number of bits per instruction increase. These instructions make creation of program much easier but it does not mean that program will run much faster because now instruction only contain more information but each micro operation (changing content of register, loading address in address bus etc.) will be performed in one cycle only.

Question 7

1. **Differentiate between hardwired control and micro programmed control. Is it Possible to have a hardwired control associated with a control memory?**

Hardwired Control Unit

With the help of generating control signals, the hardwired control unit is able to execute the instructions at a correct time and proper sequence. As compared to the micro-programmed, the hardwired CU is generally faster. In this CU, the control signals are generated with the help of PLA circuit and state counter. Here the Central processing unit requires all these control signals. With the help of hardware, the hardwired control signals are generated, and it basically uses the circuitry approach.

The image of a hardwired control unit is described as follows, which contains various components in the form of circuitry. We will discuss them one by one so that we can properly understand the generation of control signals So, on the basis of the input obtained by the conditional signals, step counter, external inputs, and instruction register, the control signals will be generated with the help of Control signal Generator.

Micro-programmed Control Unit

A micro-programmed control unit can be described as a simple logic circuit. We can use it in two ways, i.e., it is able to execute each instruction with the help of generating control signals, and it is also able to do sequencing through microinstructions. It will generate the control signals with the help of programs. At the time of evolution of CISC architecture in the past, this approach was very famous. The program which is used to create the control signals is known as the "Micro-program". The micro-program is placed on the processor chip, which is a type of fast memory. This memory is also known as the control store or control memory.

A micro-program is used to contain a set of microinstructions. Each microinstruction or control word contains different bit patterns. The n bit words are contained by each microinstruction. On the basis of the bit pattern of a control word, every control signals differ from each other.

Like the above, the instruction execution in a micro-programmed control unit is also performed in steps. So for each step, the micro-program contains a control word/ microinstruction. If we want to execute a particular instruction, we need a sequence of microinstructions. This process is known as the micro-routine. The image of a micro-programmed control unit is described as follows. Here, we will learn the organization of micro-program, micro-routine, and control word/ microinstruction.

2. **Explain various memory based addressing modes in detail with the help of suitable diagram.**

1. Implied Addressing Mode-

In this addressing mode,

The definition of the instruction itself specify the operands implicitly.

It is also called as implicit addressing mode.

Examples-

The instruction “Complement Accumulator” is an implied mode instruction.

In a stack organized computer, Zero Address Instructions are implied mode instructions.

(since operands are always implied to be present on the top of the stack)

2. Stack Addressing Mode-

In this addressing mode, The operand is contained at the top of the stack.

Example-

ADD

This instruction simply pops out two symbols contained at the top of the stack.

The addition of those two operands is performed.

The result so obtained after addition is pushed again at the top of the stack.

3. Immediate Addressing Mode-

In this addressing mode,

The operand is specified in the instruction explicitly.

Instead of address field, an operand field is present that contains the operand

Examples-

ADD 10 will increment the value stored in the accumulator by 10.

MOV R #20 initializes register R to a constant value 20.

4. Direct Addressing Mode-

In this addressing mode,

The address field of the instruction contains the effective address of the operand.

Only one reference to memory is required to fetch the operand.

It is also called as absolute addressing mode

Example-

ADD X will increment the value stored in the accumulator by the value stored at memory location X.

AC ← AC + [X]

5. Indirect Addressing Mode-

In this addressing mode,

The address field of the instruction specifies the address of memory location that contains the effective address of the operand.

Two references to memory are required to fetch the operand.

Example-

ADD X will increment the value stored in the accumulator by the value stored at memory location specified by X.

AC ← AC + [[X]]

6. Register Direct Addressing Mode-

In this addressing mode,

The operand is contained in a register set.

The address field of the instruction refers to a CPU register that contains the operand.

No reference to memory is required to fetch the operand.

Example-

ADD R will increment the value stored in the accumulator by the content of register R.

AC ← AC + [R]

NOTE-

It is interesting to note-

This addressing mode is similar to direct addressing mode.

The only difference is address field of the instruction refers to a CPU register instead of main memory.

7. Register Indirect Addressing Mode-

In this addressing mode,

The address field of the instruction refers to a CPU register that contains the effective address of the operand.

Only one reference to memory is required to fetch the operand.

Example-

ADD R will increment the value stored in the accumulator by the content of memory location specified in register R.

$AC \leftarrow AC + [[R]]$

NOTE-

It is interesting to note-

This addressing mode is similar to indirect addressing mode.

The only difference is address field of the instruction refers to a CPU register.

8. Relative Addressing Mode-

In this addressing mode,

Effective address of the operand is obtained by adding the content of program counter with the address part of the instruction.

> Effective Address
> = Content of Program Counter + Address part of the instruction

9. Indexed Addressing Mode-

In this addressing mode,

Effective address of the operand is obtained by adding the content of index register with the address part of the instruction.

> Effective Address
> = Content of Index Register + Address part of the instruction

10. Base Register Addressing Mode-

In this addressing mode,

Effective address of the operand is obtained by adding the content of base register with the address part of the instruction.

> Effective Address
> = Content of Base Register + Address part of the instruction

11. Auto-Increment Addressing Mode-

This addressing mode is a special case of Register Indirect Addressing Mode where-

In this addressing mode,

After accessing the operand, the content of the register is automatically incremented by step size 'd'.

Step size 'd' depends on the size of operand accessed.

Only one reference to memory is required to fetch the operand.

12. Auto-Decrement Addressing Mode-

This addressing mode is again a special case of Register Indirect Addressing Mode where

In this addressing mode,

First, the content of the register is decremented by step size 'd'.

Step size 'd' depends on the size of operand accessed.

After decrementing, the operand is read.

Only one reference to memory is required to fetch the operand.

NOTE-

In auto-decrement addressing mode,

First, the instruction register R_{AUTO} value is decremented by step size 'd'.

Then, the operand value is fetched.

Question 8.

1. **Discuss booth algorithm and perform -5*2 using this.**

   I. Set the Multiplicand and Multiplier binary bits as M and Q, respectively.

   II. Initially, we set the AC and Q_{n+1} registers value to 0.

   III. SC represents the number of Multiplier bits (Q), and it is a sequence counter that is continuously decremented till equal to the number of bits (n) or reached to 0.

   IV. A Qn represents the last bit of the Q, and the Q_{n+1} shows the incremented bit of Qn by 1.

   V. On each cycle of the booth algorithm, Q_n and Q_{n+1} bits will be checked on the following parameters as follows:

      i. When two bits Q_n and Q_{n+1} are 00 or 11, we simply perform the arithmetic shift right operation (ashr) to the partial product AC. And the bits of Qn and Q_{n+1} is incremented by 1 bit.

      ii. If the bits of Q_n and Q_{n+1} is shows to 01, the multiplicand bits (M) will be added to the AC (Accumulator register). After that, we perform the right shift operation to the AC and QR bits by 1.

      iii. If the bits of Q_n and Q_{n+1} is shows to 10, the multiplicand bits (M) will be subtracted from the AC (Accumulator register). After that, we perform the right shift operation to the AC and QR bits by 1.

   VI. The operation continuously works till we reached n - 1 bit in the booth algorithm.

VII. Results of the Multiplication binary bits will be stored in the AC and QR registers.

2. **Discuss division algorithm (restoration method) and solve 10/3 using this.**

A division algorithm provides a quotient and a remainder when we divide two number. They are generally of two type slow algorithm and fast algorithm. Slow division algorithm are restoring, non-restoring, non-performing restoring, SRT algorithm and under fast comes Newton–Raphson and Goldschmidt.

Let's pick the step involved:

Step-1: First the registers are initialized with corresponding values (Q = Dividend, M = Divisor, A = 0, n = number of bits in dividend)

Step-2: Then the content of register A and Q is shifted left as if they are a single unit

Step-3: Then content of register M is subtracted from A and result is stored in A

Step-4: Then the most significant bit of the A is checked if it is 0 the least significant bit of Q is set to 1 otherwise if it is 1 the least significant bit of Q is set to 0 and value of register A is restored i.e the value of A before the subtraction with M

Step-5: The value of counter n is decremented

Step-6: If the value of n becomes zero we get of the loop otherwise we repeat from step 2

Step-7: Finally, the register Q contain the quotient and A contain remainder

Question 9

1. **What do you mean by Memory Hierarchy and Cache Organization ? Also explain different mapping in cache (memory interfacing).**

In the Computer System Design, Memory Hierarchy is an enhancement to organize the memory such that it can minimize the access time. The Memory Hierarchy was developed based on a program behavior known as locality of references.The figure below clearly demonstrates the different levels of memory hierarchy :

This Memory Hierarchy Design is divided into 2 main types:

External Memory or Secondary Memory –

Comprising of Magnetic Disk, Optical Disk, Magnetic Tape i.e. peripheral storage devices which are accessible by the processor via I/O Module.

Internal Memory or Primary Memory –

Comprising of Main Memory, Cache Memory & CPU registers. This is directly accessible by the processor.

Cache is close to CPU and faster than main memory. But at the same time is smaller than main memory. The cache organization is about mapping data inweb memory to a location in cache. A Simple Solution: One way to go about this mapping is to consider last few bits of long memory address to find small cache address, and place them at the found address.

Cache Mapping: There are three different types of mapping used for the purpose of cache memory which is as follows: Direct mapping, Associative mapping, and Set-Associative mapping. These are explained below.

A. Direct Mapping

The simplest technique, known as direct mapping, maps each block of main memory into only one possible cache line. or In Direct mapping, assign each memory block to a specific line in the cache. If a line is previously taken up by a memory block when a new block needs to be loaded, the old block is trashed. An address space is split into two parts index field and a tag field. The cache is used to store the tag field whereas the rest is stored in the main memory. Direct mapping`s performance is directly proportional to the Hit ratio.

B. Associative Mapping

In this type of mapping, the associative memory is used to store content and addresses of the memory word. Any block can go into any line of the cache. This means that the word id bits are used to identify which word in the block is needed, but the tag becomes all of the remaining bits. This enables the placement of any word at any place in the cache memory. It is considered to be the fastest and the most flexible mapping form. In associative mapping the index bits are zero.

C. Set-associative Mapping

This form of mapping is an enhanced form of direct mapping where the drawbacks of direct mapping are removed. Set associative addresses the problem of possible thrashing in the direct mapping method. It does this by saying that instead of having exactly one line that a block can map to in the cache, we will group a few lines together creating a set. Then a block in memory can map to any one of the lines of a specific set. Set-associative mapping allows that each word that is present in the cache can have two or more words in the main memory for the same index address. Set associative cache mapping combines the best of direct and associative cache mapping techniques.

2. **Explain pipelining in detail i.e. performance, advantages, implementation in Control Unit etc. Also explain pipeline Hazard and ways to overcome them.**

Pipelining is a process of arrangement of hardware elements of the CPU such that its overall performance is increased. Simultaneous execution of more than one instruction takes place in a pipelined processor. Let us see a real-life example that works on the concept of pipelined operation. Consider a water bottle packaging plant. Let there be 3 stages that a bottle should pass through, Inserting the bottle(I), Filling water in the bottle(F), and Sealing the bottle(S). Let us consider these stages as stage 1, stage 2, and stage 3 respectively. Let each stage take 1 minute to complete its operation. Now, in a non-pipelined operation, a bottle is first inserted in the plant, after 1 minute it is moved to stage 2 where water is filled. Now, in stage 1 nothing is happening. Similarly, when the bottle moves to stage 3, both stage 1 and stage 2 are idle. But in pipelined operation, when the bottle is in stage 2, another bottle can be loaded at stage 1. Similarly, when the bottle is in stage 3, there can be one bottle each in stage 1 and stage 2. So, after each minute, we get a new bottle at the end of stage 3. Hence, the average time taken to manufacture 1 bottle is:

Without pipelining = 9/3 minutes = 3m

With pipelining = 5/3 minutes = 1.67m

Overlapped execution:

Total time = 5 Cycle Pipeline Stages RISC processor has 5 stage instruction pipeline to execute all the instructions in the RISC instruction set. Following are the 5 stages of the RISC pipeline with their respective operations:

Stage 1 (Instruction Fetch) In this stage the CPU reads instructions from the address in the memory whose value is present in the program counter.

Stage 2 (Instruction Decode) In this stage, instruction is decoded and the register file is accessed to get the values from the registers used in the instruction.

Stage 3 (Instruction Execute) In this stage, ALU operations are performed.

Stage 4 (Memory Access) In this stage, memory operands are read and written from/to the memory that is present in the instruction.

Stage 5 (Write Back) In this stage, computed/fetched value is written back to the register present in the instructions.

Dependencies in a pipelined processor

There are mainly three types of dependencies possible in a pipelined processor. These are :

i. Structural Dependency
ii. Control Dependency
iii. Data Dependency

These dependencies may introduce stalls in the pipeline.

Stall : A stall is a cycle in the pipeline without new input.

Structural dependency

This dependency arises due to the resource conflict in the pipeline. A resource conflict is a situation when more than one instruction tries to access the same resource in the same cycle. A resource can be a register, memory, or ALU.

Control Dependency (Branch Hazards)

This type of dependency occurs during the transfer of control instructions such as BRANCH, CALL, JMP, etc. On many instruction architectures, the processor will not know the target address of these instructions when it needs to insert the new instruction into the pipeline. Due to this, unwanted instructions are fed to the pipeline.

Data Dependency (Data Hazard)

A position in which an instruction is dependent on a result from a sequentially earlier instruction before it can be done its execution. In high-performance processors operating pipeline or superscalar techniques, a data dependency will learn an interruption in the flowing services of a processor pipeline or prevent the parallel issue of instructions in a superscalar processor.

Consider two instructions ik and ii of the same program, where ik precedes ii. If ik and ii have a common register or memory operand, they are data-dependent on each other, except when the common operand is used in both instructions as a source operand.Data dependency can appear either in ‘straight-line code’ between subsequent instructions or in a loop between instructions belonging to subsequent iterations of a loop

Data dependencies in straight-line code