5(c) - RISC & CISC

CISC	RISC
→ Stands for Complex Instruction set format.	→ Reduced Instruction Set format.
→ Large no. of instructions (around 1000) application point of view they are useful but for system	→ Relatively few numbers of instructions only those instructions which are strictly required, and in case of implementation of a complex func

designer it is a headache

→ Some instruction is there which perform specific task and are used infrequently for eg. instructions for testing.

→ large no. of address mode, leading to lengthen instruction, but compilation and translation will be faster.

→ Variable length instruction format

→ Instruction that manipulate operand in memory.

→ Powerfull but costly

→ Microprogrammed control unit, relatively slow

a set of simple instruction will perform together.

→ few instruction will be there which will be preforming more frequent work.

→ No. of instruction mode will be less, hardly 3 to 4.

→ Fixed length so easy to decode instruction format.

→ do not perform opn directly in memory, first load them in register and then perform, so all opn will be done with in the register of CPU.

→ Relatively less powerfull but cheap

→ Hardwired control unit, so fast.

Q6)

Miss Rate

The miss rate, often expressed as a percentage, measures the ratio of cache misses to the total numbers of memory accesses. It indicates how often the processor needs to retrive data from a slower memory level because it was not found in the cache. A lower miss rate is desirable as it implies that the cache is effectively storing frequently accessed data.

Miss Penalty → refers to the time it takes to retrieve data from a lower memory level when a cache miss occur. It

includes the time to access the slower memory and potentially update the cache. The miss penalty directly impacts the overall execution time of programs, as cache misses introduce delays in accessing data that are not readily available in the cache.

7(b) Flynn's classifications

Importance
- Parallel Processing
- System Design
- Performance optimization

It is based on the multiplicity of instruction stream and data streams in a computer system
- SISD, SIMD, MISD, MIMD

Instruction stream - is defined as the sequence of instructions executed by the processing unit.

Data stream - is defined as the sequence of data including inputs, partial or temporary results called by the instruction stream.

Acc. to Flynn's classification, either of the instruction or data stream can be single or multiple.
- Single - instruction stream, single data streams (SISD)
- Single " " multiple " " (SIMD)
- Multiple " " single " " (MISD)
- Multiple " " multiple " " (MIMD)

* SISD
- is a uni-processor machine which is capable of executing a single instruction, operating on a single data stream.
- Single instruction - only one instruction stream is being acted on by the CPU during any one clock cycle.
- Single data - only one data stream is being used as input during any one clock cycle.
- It is a serial (non-parallel) computer

Eg - Traditional personal computers & old mainframes

Uses
Suitable for simple, non-parallel task

• Instruction are executed sequentially, but may be overlapped in their execution stages (Pipelining). Most SISD uni-processor systems are pipelined.
• SISD computers may have more than one functional units all under the supervision of control unit.

SIMD

Ex- GPU's, vector processors.
uses cases → suitable for task like image processing matrix opn and scientific computation.

is capable of executing the same instruction on all the CPU's but operating on different data streams.

Single instruction - All processing unit execute the same instruction at any given clock cycle.

Multiple data - Each processing unit can operate on a different data element.

SIMD computer has single control unit which issues one instruction at a time but it has multiple ALU's or processing unit to carry out on multiple data sets simultaneously.

well suited to scientific computing since they involve lots of vector and matrix opn.

MISD (multiple Instruction, single data)

multiple Instruction streams are applied to a single data stream.
This is a rare and specialized architecture, not commonly found in practical systems.
Ex- Fault tolerant systems like those used in spacecraft, where multiple computations are performed on the same data to check for errors.

* Characteristics
  - Executes multiple instruction streams.
  - operates on single data.

Use Cases
useffull useful in situation that require redundancy and error checking.

MIMD (multiple Instruction, multiple data).

multiple processors independently execute different instruction streams on different data sets. this architecture is the most versatile and is commonly used in modern parallel computing system.
Ex- multi-core processor, distributed system and super computer.

* Characteristics
  - Executes multiple instruction stream
  - operate on multiple data stream
  - Highly scalable and efficient in handling complex tasks.

Use Cases
Suitable for general purpose parallel computing scientific simulations and large scale data processing.

PYQ's. Jan-2022 (ACA)

Ans (j) Horizontal microinstruction

Use more bits to specify each operation. They can use hardware more efficient and require fewer microinstruction per microprogram. The intel 8086 is an example of a microprocessor that uses horizontal microcode.

Vertical Microinstruction

These microinstruction are more compact but less flexible than horizontal microinstruction. In vertical microprogramming, the control bits are encoded using separate codes for each op to be carried out.

Ans (i) TLB stands for Translation lookside Buffer, a type of memory Cache that helps a computer system quickly acess translation of virtual addresses to physical addresses.
Which allow for faster retrieval and efficient memory management It can be called an address-translation cache. It is the part of the chip's memory-management unit (MMU).

(h) microoperation → is a basic operation performed on data stored in a Computer's registers.

A micro-operation is a low-level institution that performs a simple operation on data in one or more registers.
It is used to implement Complex machine instructions also known as macro-instructions.

Microprogram → is a series of microinstruction that define the opn a computer peforms in response to a machine-language instruction. Typically One machine language instruction translates into several microcode instructions.

(g) Advantages of pipelining (Notes-Coa)

(f)

(b) Auto Increment Mode

The Register's Contents are incremented after the operand is addressed.

The mnemonic for auto increment mode is written with a + sign following. Such as (Rn) +.

Auto Decrement Mode

The Register's Contents are decremented before the operand is addressed.

The mnemonic for auto decrement mode is written with a minus sign leading. Such as -(Rn).

It is useful for implementing the stack structure.

(c) Locality of Reference is a computer science principle that describes the tendency of a processor to repeatedly access the same memory locations in a short period of time. There are mainly two types 1) Temporal 2) Spatial

(d) Single Rank Registers

A single-rank has one set of memory chips that are accessed when reading or writing to the memory.

Dual Rank - A dual-rank has two sets of memory chips, similar to having two single-rank on the same module.

Some differences

- Speed - Single-rank is usually faster than dual rank RAM.
- Capacity - A dual-rank module has double the capacity of a single rank module.

(e) Yes, multicomputers with distributed memory are more scalable than shared memory.

Distributed memory multiprocessors are a type of high performance computing (HPC) system that can

Scale to large numbers of processors. This is Beoz each Processor has its own private memory, and the processors Communicate with each other through message passing.

(b) Amdahl's law

States that when a part of a system is improved, the overall system improvement will be proportional to how much the part makes up of the system.

The overall speed up is limited by the fraction of the program that remains sequential.

$S_{max} = 1/(1-P) + Ps.$

(c)

Long.

(d) Control Flow

- Process is the key: Precendence constraints control the project flow based on task completion, success or failure.
- Task 1 needs to complete before task 2 begins.
- Smallest unit of the control flow is a task.
- Task are run in series if connected with precedence or in parallel
- Package control flow is made up containers and tasks connected with precendence constraints to control pack.

Data flow.

- Streaming

- Unlink control flow, multiple components can process data at the same time.
- Smallest unit of the data flow is a component.
- Data flows move data, but are also task in the control flow as such their success or failure effects how your control flow operates.
- Data is moved and manipulated through transformations.
- Data is passed between each component in the data flow.
- Data flow is made up of source(s) transformations and destinations.

(4) Refer (Notes) (RISC & CISC)

5(a) Instruction Count I_c = 100,000 = 1×10^5 Clock Rate = 40MHz
$40 * 10^6$ Hz.

- No. of Arithmetic & logic opn = 60% of 1×10^5
$= \frac{60}{100} \times 1 \times 10^5 = 60 \times 10^3$
- No. of Load/store 18% of 1×10^5 = 18×10^3
- No. of Branch Instruction = 12% of 1×10^5 = 12×10^3
- No. of memory Reference = 10% of 1×10^5 = 10×10^3.

(A) Average CPI = $\frac{\sum_{i=1}^{n} (CPI * I_{c_i})}{I_c}$

$= \frac{(60 \times 1 + 18 \times 2 + 12 \times 4 + 10 \times 8) \cdot 10^3}{(60 + 18 + 12 + 10) \cdot 10^3}$

$\Rightarrow \frac{224}{100} = 2.24$

(B) MIPS Rate = $\frac{f}{CPI * 10^6}$

$\Rightarrow \frac{40 \times 10^6}{2.24 \times 10^6} \Rightarrow \frac{40}{2.24} \Rightarrow 17.86$

TOPIC - Bernstein's Condition

A set of conditions on the basis of which two processes can execute in parallel.

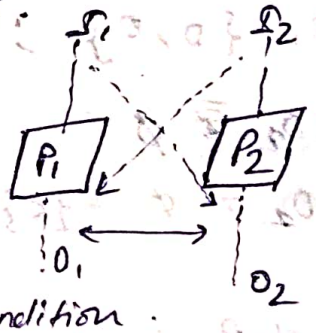
Notation.

- Process P_i is a software entity.
- Input I_i is the set of all input variables for a process P_i.
  I_i is also called the read set or domain of P_i.
- Output O_i is the set of all output variables for a process P_i.
  O_i is also called write set or range of P_i.

According to Bernstein.

Two processes P_1, P_2 with input sets I_1, I_2 and output sets O_1, O_2 can execute in parallel (denoted by $P_1 \| P_2$) if:

$$I_1 \cap O_2 = \phi$$
$$I_2 \cap O_1 = \phi$$
$$O_1 \cap O_2 = \phi$$



These three conditions are known as Bernstein's Condition.

* On term of data dependencies..

Imply that two processes can execute in parallel if they are:

- flow - independ
- anti - independent
- output independent

$I_1 \cap O_2 = \phi$ Anti - Independent

$I_2 \cap O_1 = \phi$ Flow - independand

$O_1 \cap O_2 = \phi$ output - independent

* Properties of parallelism relation ($\|$) :-

- Commutative ($P_i \| P_j$ implies $P_j \| P_i$)
- Non-transitive ($P_i \| P_j$ and $P_j \| P_k$, does not imply $P_i \| P_k$)
- Associative $(P_i \| P_j) \| P_k =$
  $= P_i \| (P_j \| P_k)$

Note Condition $I_1 \cap I_2 \neq \phi$ does not prevent parallelism between two processes. However violation of any of the three conditions prohibits parallelism between two processes.

PYQ: Jan 2022

4. $P_1 : C = D * E$

$P_2 : M = G + C$

$P_3 : A = B + C$

$P_4 : C = L + M$

$P_5 : F = G / E$

Let us consider two process P_1 & P_2

$P_1 : C = D * E$ $\qquad$ $P_2 : M = G + C$

$I_1 = \{D, E\}$; $O_1 = \{C\}$ $\qquad$ $I_2 = \{G, C\}$ & $O_2 = \{M\}$

$\therefore P_1 \neq P_2$

$I_1 \cap O_2 \not\equiv \phi$

$I_2 \cap O_2 \neq \phi = \{C\} \longrightarrow$ (Flow dependence) {Aese he Check ko linge sabki ko}

Flow dependences → P_1 and P_2, P_1 and P_3, P_2 and P_4

Anti dependences → P_2 and P_4, P_3 and P_4

Output dependences → P_1 and P_4

Resources Dependences - P_2 and P_3, P_2 and P_4, P_3 and P_4 {Same resource, adder is used}.

---

5. Execution time Bole nhi to nikalne ke liye agr bola hota to aese nikalte P_3.

$f = 40 \times 10^6$ Hz / $I_c = 1 \times 10^5$

$T = \dfrac{I_c * CPI}{f} \Rightarrow \dfrac{1 \times 10^5 \times 2.24}{40 \times 10^6}$

$\Rightarrow 0.0112$

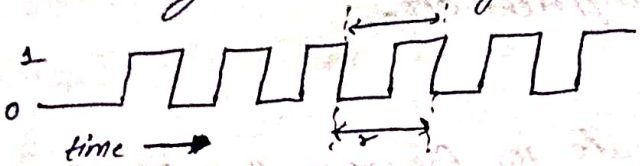
$\Rightarrow 11.2$ ms.

CPU Performance

1. Clock Rate and CPI 2. Execution Time 3. MIPS Rate 4. Throughput Rate (Performance

CPU clock

Computers are constructed in such a way that events in hardware are synchronized using a clock. one clock cycle.



Cycle Time / clock time / Clock Period (τ):
CPU is driven by a clock of constant cycle time τ (in ns).

Clock rate / clock frequency (f): is defined by the inverse of cycle time.

$$\text{Clock rate } f = 1/\tau \quad \text{in Hz (= cycle second)}$$

Also Cycle time $\tau = 1/f$

Instruction Count (IC)

is the size of a program, which is the no. of machine instruction to be executed in a program. Sometimes called Instruction Path length.

A machine instruction is a sequence of micro-operation. A micro-operation is an elementary hardware operation that can be carried out in one clock cycle.

Thus a single machine instruction may take one or more CPU cycles to complete.

We can characterize instruction by cycle per Instruction (CPI)

CPI - Different machine instruction may require different No. of clock cycles to execute. Therefore CPI (Cycles Per Instruction) becomes an important parameter.

CPI measures the time needed to execute each instruction i.e. No. of clock cycles per instruction for a program.

$$CPI = \frac{\text{CPU Clock Cycles for a Program}}{\text{Instruction Count}} = \frac{C}{I_c}$$

Average (or Effective) CPI of a Program:

→ of all instruction executed in the program of a given processor.

- Different instruction can have different CPIs.

Execution Time.

It is the time needed to execute the program thus the Execution Time / CPU Time (T in seconds/program) is given by -

$$T = I_c * CPI * \tau$$

I_c = Total No. of instruction executed

CPI = Avg. No. of cycles per instruction

τ = Clock cycle time

T = Execution Time per Program in seconds

Unit for CPU Execution Time = $\frac{\text{Seconds}}{\text{Program}}$

⇒ CPI = Instruction cycle = Processor cycles + memory cycle.

CPI = Instruction cycle = $P + m*k$

(CPU Time)(T)

↳ $T = I_c * (P + m*k) * \tau$

P = No. of Processor cycles

m = No. of memory reference

k = ratio b/w memory & Processor cycle.

τ = Processor cycle Time

MIPS Rate

Let C be the total No. of clock cycle needed to execute a Program

∴ CPU Time (T) = ~~total~~ No. of clock cycles * clock cycles.

$$T = C * \tau = C / f$$

furthermore

As we know $CPI = C / I_c$ so $\boxed{C = CPI * I_c}$

and $T = CPI * I_c * \tau$

∴ (3)

$$\boxed{T = \frac{I_c * CPI}{f}}$$

The Processor speed is often measured in terms of MIPS (Millions Instruction Per second). It is simply called MIPS Rate of a given Processor

$$\text{MIPS Rate} = \frac{I_c}{T * 10^6} = \frac{f}{CPI * 10^6} = \frac{f * I_c}{C * 10^6} \quad (4)$$

Using eq4.4.

$$T = \frac{I_c}{MIPS * 10^6} \Rightarrow \frac{1}{T} = \frac{MIPS * 10^6}{I_c}$$

$$I_c = \frac{MIPS \times 10^6}{C} \times T$$

4 Throughput Rate (Performance)

Throughput (W_s) - The No. of programs executed per unit time is called System throughput (in Programs / second)

$$W_s = \frac{1}{T} = \frac{1}{\text{Execution Time}}$$

$$W_s = \frac{f}{I_c * CPI} = \frac{MIPS * 10^6}{I_c}$$

* Instruction types & CPI

$CPI = C / I_c$

I_{ci} = No. of instruction of type i executed.

CPI_i = Cycle per instruction for type i

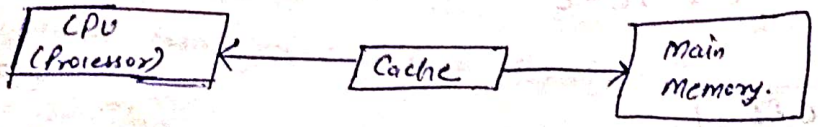
So for CPU Design

CPU clock cycles = $CPI * I_c = \sum_{i=1}^{4} (CPI * I_c)$

The overall CPI

Average (or Effective) CPI $= \sum_{i=1}^{n} (CPI_i * f_{ci}) = \frac{\sum_{i=1}^{n} (CPI_i * I_{ci})}{I_c} \Rightarrow \sum_{i=1}^{n} \left(CPI_i * \frac{I_{ci}}{I_c}\right)$

CACHE MEMORY



* Small sized fast memory
→ Placed between main memory & CPU.
→ High-speed volatile memory.
→ Contains most frequently accessed instruction data.
→ located inside the CPU chip or motherboard.
(internal cache L1 & L2) (external Cache-L3)

Cache Performance

It is measured in terms of Hit Ratio.

Cache Hit :- If the required word is found in cache is called (h) Cache Hit.

Hit Ratio (h) $= \frac{\text{Hits}}{\text{Hit + miss}}$

$= \frac{\text{No. of Hits}}{\text{Total no of CPU Refernces}}$

Cache Access Time : Time required to access word (each Hit time) from the cache.

Cache Miss - If the required word is not found in cache is (1-h) called Cache Miss.

Miss Ratio $= \frac{\text{Miss}}{\text{Hits + miss}}$

$= \frac{\text{Total No. of miss}}{\text{total no. of CPU refernces}}$

Miss Penality (T_m) (Cache Miss Time Penalty)
→ the time required to fetch the required block from main memory.

⇒ Average Access Time = Hit Ratio X Cache Access + (1-Hit Ratio) X Miss Penalty Time.
of CPU

{1 - Hit ratio / Cache miss}

$= h \times T_C + (1-h) \times T_m$

Instruction Set

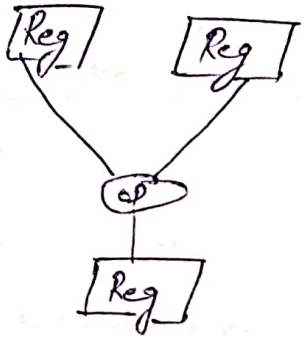
Instruction sets define the many different kinds of data and their manipulations by different processors.

1) The L/S Architecture (Load - Store)
2) The R/m Architecture (Register Memory)
3) The R+M Architecture (Register Plus memory)

The Instruction Set

The L/S Architecture: The L/S or Load Store Architecture specifies that all operand values must be loaded from Memory into Registers before an execution can take place.

An ALU Add instruction must have both operands and result specified as Registers (Three Address format)



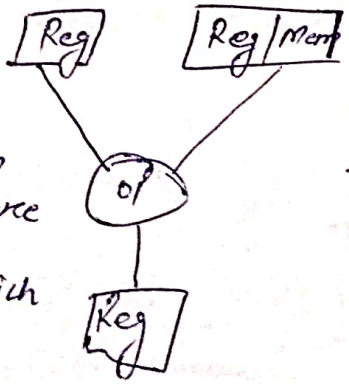
~~Two Address format~~ operand in Memory is not allowed.

Most General purpose modern mainframe Computers like IBM, ~~Hitachi~~, etc. as well as several microprocessor Cintex X86 series follow R/m style. → ye R/M Arch ka hai.

The R/m Architecture

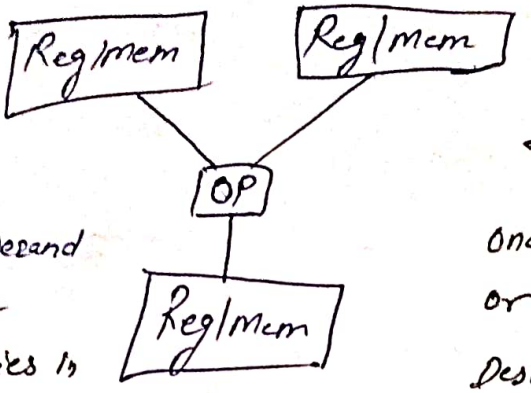
The R/m or Register Memory architecture includes instructions that can operate both on registers and one operand in Memory.

An ALU Add instruction one source operand lies in memory and the other source operand lies in Register which also serves as Destination.



Two address format.

The R+m Architecture: The R+m or Register Plus memory architecture includes instruction that can operate on operands both in Registers and Memory.



In an ALU Add instruction one source operand lies in memory and the other source operand lies in Register which also serves as Destination.

In an ALU Add instruction all operand lie in memory or in Registers or any combination there of.

Two address format.

One source operand in Register or memory is also the Destination.

Three Address format

(Three operands independently specified and each may be a register or memory.

* Digital Equipment (DEC) VAX Series of machines and motorola M68xxo Series of microprocessors use this architecture.

b

DCC 2022

Part-B.

(7½)

2(a) Write a code fragment implementation of a vector summation for RISC architecture

Program flow Mechanism

- Control flow mechanism
- Data-flow mechanism
- demand-driven mechanism

* Control flow mechanism

- Conventional von Neuman computers use a program counter to sequence the execution of instruction in a program. This sequential execution style is called control-driven.
- Conventional computers are based on a control flow mechanism by which the order of program execution is explicity stated in the user programs.
- Control flow computers use shared memory to hold program instruction and data objects.
- A uniprocessor computers in inherently sequential, due to use of the control-driven mechanism.
- Control flow can be made parallel by using parallel language constructs or parallel compiler.

* Data flow

Dataflow computers are based on a dataflow mechanism which allows the execution of any instruction to be driven by data availability. Dataflow computers emphasis a high degree of parallelism at the fine-grain instructional level.

- Data flow machines have high potential for parallelism and throughput and freedom from side effects, but have high control overhead, lose time waiting for unneeded arguments and difficulty in manipulating data structures.

* Demand – Driven Mechanism

- Data driven machines select instructions for execution based on the availability of their operands this is essentially a bottom-up approach.
- Demand-driven machines take a top-down approach, attempting to execute the instruction that yields the final result. The triggers the execution of instruction that yield ~~the~~ its operan and so forth.
- The demand-driven approach matches naturally with functional programming languages.

	Control flow	Dataflow	Demand-Driven
Advantage	1) Full Control 2) Complex data & control structures are easily implemented.	1) very high potential for parallism. 2) High throughput 3) Free from side effect.	1) Only required instructions are executed. 2) High degree of parallelism. 3) Easy manipulation of data structures.
Disadvantage	1) Less efficient 2) Difficult in Programming 3) Difficult in Preventing run-time error	1) Time loss waiting for unneeded arguments 2) High Control overhead 3) Difficult in manipulating data structures	1) does not support sharing of objects with changing local state 2) Time needed to propogate demand tokens.

Page No.
Date

Ques Condition of Parallelism :-
3 main types of dependencies are-

(i) Data Dependence
It happen when instruction or tasks in a program rely on the data produced by another task. In simpler words, if one instruction depend on the result of a previous instruction to continue, they cannot be executed at the same time.

Types (a) RAW (Read After write) :- The second tasks reads the data that the first task writes.

Eg:- Task 1 = a=5+3
Task 2 = b=a+2

Here Task 2 depends on task 1 because it needs the value of a before it can continue.

(b) WAR (write After Read) :- The second task with data after the first task has read it

Eg:-
Task 1 b = c+4 Here Task 1 must read c
Task 2 c=7 before task 2 changes its value.

(c) WAW (write After write) - Both tasks write to the same data and the order of execution affects the final value.

Eg:- Task 1 = a=4
Task 2 = a=7

The order matters bcz both tasks update a

* Control dependence - It occurs when the execution of one task depends on the result of

Date
Page No.

a condition or decision in another task. Simply put, if one task determines whether another task will run, they cannot be executed at the same time.

Ex - If (x>5){
Task 1: print ("Hello")
}
Task 2: x=3

Here, whether Task 1 will execute depends on the value of x. If x is not greater than 5, task 1 won't run, which creates control dependence.

(3) Resource Dependence

It happens when 2 or more tasks needs the same hardware resource (like memory, CPU, or I/O devices) at the same time. If multiple tasks are fighting for the same resources they cannot run in efficiency.

For Ex -

- If two tasks both need access to the same printer or hard drive, only one task can use it at a time
- If two tasks require the same CPU core or memory location they will have to wait for each other to finish

Condition for Achieving Parallelism

(1) No data dependence - Task should not depend on each other data. If they do, they need to be organised in a way that minimizes waiting.

(2) No control dependence - Parallel tasks should not depend on each other's decisions or conditions. This allows them to run independently without waiting for a condition to be evaluated.

(3) No Resource dependence - Tasks should have separate resources or a way to store them efficiently. This means they won't have to wait for resources to become available.

Ques 2 Hardware & Software Parallelism

(i) Hardware Parallelism - It refers to the physical component of a computer system that enable it to execute multiple tasks at the same time.

Key Aspects

(i) Multiple Processors or cores - Modern CPUs often come with multiple cores or processors. Each core can handle a separate task simultaneously making the system faster and more efficient. Eg:- A quad-core processor can perform four different tasks at the same time.

- Pipelining - This technique divides an instruction into different stages (like fetching, decoding, executing etc.) and these stages work in parallel. It is like an assembly line in a factory where each worker performs a different tasks on the same product at the same time.

SIMD (Single Instruction Multiple data)

A single instruction is applied to multiple data points at the same time. Graphics cards (GPUs) use this technique to process large amounts of image data quickly.

- Parallel memory system - Hardware parallelism also involves having multiple memory banks ~~that can~~ that can be accessed to read and write data

Date
Page No.

Advantages
(1) faster Processing
(2) Efficient use of resources
(3) Scalability

2. Software Parallelism
It refers to the way programs and applications are designed to take advantage of parallel hardware

Key features
(1) Multi Threading - A program is divided into multiple threads that can run independently. This allows different parts of a program to be executed simultaneously.
(2) Data parallelism - This type of Parallelism focuses on distributing data across different processing units. Each processor perform the same operation on different pieces of data at the same time.
(3) Task Parallelism - Unlike data parallelism task parallelism divides the Program into distinct task or functions that can run concurrently. Each task can be different and they might depend on each other's result

Advantages
(1) Efficient use of Hardware
(2) Faster Execution
(3) Scalability

Page No.
Date

Ques Program Partitioning and scheduling

They are critical concept in parallel computing and distributed systems. They are used to improve performance efficiency and resource utilization by breaking down a large computational task into smaller components

(i) Program partitioning

It is the process of dividing a computational task into smaller, manageable parts or segment called partitions or tasks. The goal of partioning is to maximize parallelism by creating tasks that can executed concurrently.

Effective partioting reduces the overall execution time and balances the workload across processing.

Types

(A) Functional partioning

- In this method, the program is divided based on the functions or modules that it performs.
- Each task or partitions handles a specific function of the program.
- Ex - In database application, one task may handle data retrieval another handles data sortings and another handles data storage.
- Advantages

Simplifies task design and implementing by focusing on specific functions.

(B) Data partioning

Date
Page No.

- It involves dividing the data set into smaller chunks / where each partition operates on a separate portion of the data.
- It is widely used in applications that performs repetitive operations on large data sets.

Ex- Matrix multiplication can be partioned so that each processor handles a portion of the matrix.

Advantages

Efficiently distributes the workload across multiple processors, enhancing performance.

* Factors to Consider in Partitioning

- Granularity → Refers to the size of the tasks in the partition. It can be-
  (a) Fine grained: tasks are small and numerous providing a high level of parallelism.
  (b) Coarse-grained- Tasks are larger with fewer parallelization opportunities but overhead.
- Data Dependencies - Identifies if tasks depend on each other result. Dependencies should be minimized to reduce communication overhead b/w tasks.
- Load Balancing - Ensures that each processor has an approximately equal amount of work preventing some processor from being idle while

Others are overloaded.

(2) Program Scheduling

It is the process of allocation the tasks generated from partioning to the available processors or computing units in an optimal manner. The aim to minimize execution time, maximize resources utilization and reduce idle time for processors

Types :-

(a) STATIC :

- The scheduling decisions are made at compile time before the program starts running.
- Each task is assigned to a specific processor based on a pre-defined schedule.

Advantages

Simple to implement, low overhead during execution

Drawbacks

Lacks flexibility not well-suited to dynamic workloads or unpredictable conditions

Ex- Round-Robin scheduling where tasks are assigned in a fixed order.

(B) Dynamic Scheduling

- Scheduling decisions are made at runtime based on the system's state and the current workload
- Tasks are assigned to processor dynamically as they become available.

Advantages

Adapts to changes in workload, improves load balancing

Date
Page No.

and handles unpredictable scenarios.

Draw Backs

Higher overhead due to design making during execution

E.g. Load Based Scheduling where tasks are distributed to the least buy processor.

* Factors Affecting Program Scheduling

- Communication overhead - Minimizing the amount of communication b/w tasks and processors is crucial to reducing the execution
- Resource Availability - The Schedules must consider the availability of resource like memory, CPUs and I/O devices.
- Task Dependicies - The ~~Schedules must consider~~ the availability Task with inter dependencies must be scheduled in a way that respects their execution order.
- Importance of Program
  - Improves Performance
  - Reduce Execution time
  - scalability.

Challenges in partioning & scheduling

- Compler dependencies
- load Imbalance
- Overhead.

Page No.
Date

* Catch Memory

Cache Memory is a high speed storage layer that sits between the CPU and main memory. It stores frequently accessed data to reduce access latency and improve overall system performance.

Types of Caches

1. Split I and D caches

A split cache is a type of cache that is divided into two separate parts:

1. Instruction Cache (I-Cache)
   - stores frequentey accessed instructions.
   - Improves instruction fetch performance by reducing memory access latency.
   - Typically read-only, as instruction are not modified during execution

2. Data Cache (D-Cache)
   - stores frequently accessed data.
   - Improves data access performance by reducing memory access latency.
   - Can be both read and write, as data can be modified during execution.

Advantages

- Improved Performance: By separating instruction and data access, split cache can avoid contention and improve overall

Date
Page No.

System performance.

Increase Bandwidth - Split caches can allow for simultaneous access to both instruction and data, effectively doubling the memory bandwidth.

- Reduced Cache Miss rate - By dedicating separate cache for instruction and data, the likelihood of cache misses can be reduced.

Disadvantages of Split Caches:

- Increased Complexity - Split caches require more complex hardware design and control logic.
- Potential for stalls - If the I-cache and D-cache have different access latencies, it can lead to pipeline stalls.

2. On-Chip Caches

On Chip Caches are high-speed memory components integrated directly into the CPU chip. They are further categorised into different levels:

1. Level 1 (L1) Cache:
   - Smallest & fastest cache.
   - Located closest to the CPU core.
   - often split into separate instruction and data caches.

Level (L2) Cache.

- Larger than L1 cache
- Slower than L1 cache but faster than main memory.
- Often shared among multiple cores.
- Can be either split or unified.

Benefits of On-chip Caches:

- Reduced Memory Access latency → By providing faster access to frequently used data and instructions, the cache hierarchy significantly reduces the averages memory access time.
- Improved Processor Performance → Faster memory access allows the processor to execute instructions more quickly, leading to higher overall system performance.
- Increased Effective Memory Bandwidth - The cache hierarchy can hide memory latency and increase the effective bandwidth of the memory system

PYQ: Jun 2022

Q 2 Discuss the various levels of computational granularity in parallel computers?

Ans Computational Granularity refers to the size of tasks or units of work that executed concurrently in parallel computer system. It is a crucial factor in determining the efficiency and scalability of parallel algorithms.

Level of Granularity

Date
Page No.

1. Fine-grained - Involves very small tasks, often at the level of individual instruction or small blocks of instructions.

Characteristics
- High degree of parallelism.
- Requires frequent communication and synchronisation
- Suitable for highly parallel architectures like SIMD.

Example - Vector processing, where multiple processors operate on different elements of a vector simultaneously.

2. Medium Grained

Definition - A balance between fine-grained and coarse grained parallelism, involving tasks of moderate size.

Characteristics
- Moderate degree of Parallelism.
- Require less frequent communication than fine-grained
- Suitable for a wide range of parallel architectures.

Example. loop-level parallelism, where different iteration of a loop are executed concurrently.

3. Coarse-Grained - Involves large task often at the level of subroutines or entire functions.

Characteristics - low degree of parallelism
- Requires infrequent communication
- Suitable for distributed memory system.

Example - Task parallelism, where different processes or thread execute independent tasks.

Page No.
Date

Q Virtual memory

⇒ Virtual memory is a memory management technique that allows a process to use more more than is physically available on the system. It creates an illusion of a larger memory space by dividing physical memory into fixed-size blocks called pages and virtual memory into equal-sized blocks called virtual pages.

Advantage

- Increased Process isolation - Each process has its own virtual address spaces, preventing interference between processes.
- Efficient Memory utilization - physical memory can be shared among multiple processes, improving memory utilization.
- Simplified Memory Management - virtual memory simplifies memory allocation & deallocation.
- Process Independence - Process can be written as if they have exclusive access to a large amount of memory, without worrying about physical memory limitations.

Disadvantage

- Page Fault Overhead - Page Faults can significantly degrade performance, especially if they occur frequently.

Date
Page No.

- Memory Overhead - Page Tables Consume memory, which can impact system performance.
- Thrashing - If a process experiences a high rate of page faults, it can lead to thrashing, where the system spends more time swapping pages in and out of memory than executing instructions.

* Granularity and Parallelism

- Fine-grained - Highest degree of parallelism, but can be limited by communication overhead.
- Coarse-grained - Lower degree of parallelism, but can be more efficient due to reduced communication overhead.

Factors Affecting Granularity Choice

- Problem characteristics - The nature of problem using solved can influence the appropriate granularity level.
- Hardware architecture - The specific architecture of parallel computer system can affect the optimal granularity
- Software Implementation - The way the parallel algorithm is implemented can impact the granularity.