Regal
Date 27/05/24
Page

Computer Architecture

① What are the diff. parameters used to measure CPU performance?

→ To estimate the CPU performance, the measure that is generally most imp. is execution time, T.
because, Performance = $\frac{1}{\text{Execution time}}$.

So, if Execution time ↑ CPU performance ↓.

3 parameters → (i) Speedup
(ii) Efficiency
(iii) Throughput.

Speed up → Effect of improvement expressed in terms of Speedup, S
→ Ratio of execution time without improvement (T_{wo}) to the execution time with improvement (T_w)
$S = T_{wo} / T_w$

Speed-up as a direct percent can be represented as:-
$S = ((T_{wo} - T_w) / T_w) \times 100$

Efficiency → Ratio of Speed-up to the no. of processor used.
$E = S/p$

Throughput → measure of no. of computation over a unit time. $w = \frac{n}{t}$.

② "Instruction execution throughput ↑ in proportion with the no. of pipeline stages." Is it true? Justify.

→ Pipeline refers to the technique in which a given task is divided into a no. of subtasks that need to be performed in sequence.
→ Each subtask is performed by a given functional unit.
→ The units are connected in a serial fashion & all of them operate simultaneously.

→ The use of pipelining improves the performance compared to the traditional sequential execution of tasks.
→ Considering the execution of m tasks (instructions) using n-stages (units) pipeline. We assume that the unit time T = t units.
Then the throughput U(n) is, $m/(n+m-1)\,xt$
i.e., no. of tasks executed per unit time.
So, if we ↑ the no. of stages in a pipeline ↓ also ↑ the throughput of the pipeline.

③ Discuss data hazards.
(i) A data hazards is any condition in which the source or the destination operands of an instruction are not available @ the time expected in the pipeline.
(ii) As a result of which some operation has to be delayed & pipeline ~~batti~~ Stalls. Whenever there are 2 Instr. One of which depends on the data obtained from other.

$$A = 3 + A$$
$$B = A * 4$$

Types:- (i) RAW (ii) WAR (iii) WAW.
• Read After Write :- (True dependency)
This occurs when an instr. depends on the result of a prev. instr. that has not yet completed.
For ex:- if instr. B needs to read a value that instr. A is currently writing.

• Write After Read (WAR) (anti-dependency)
This happens occurs when an instruc". needs to write a value after a prev. instr. has read it, but the read occurs before the write operation completes.

- Write after Write (WAW) (Output dependency)
This happens when 2 instruc. write to the same location, and the 2nd write operation occurs before the 1st write completes.

(4) Briefly describe cache coherence problem with an ex. Suggest one software protocol for this.
→ Memory coherence properly can manage the memories of a multiprocessor system so that no data is lost or overwritten before the data is transferred from a cache to the target memory.
→ Memory caching is eff. bcz. most programs access the same data or instructions over & over.
→ Cache coherence is a concept in CA that ensures that multiple copies of data in diff. caches remain consistent.
→ It is critical for the correctness & performance of multi-core processors & multiprocessor systems.
→ It ensures that all processors have a consistent view of memory, which is essential for the correct execution of parallel programs.

(5) a) Define pipeline hazard? Discuss its types.
→ Pipeline hazard:-
- Also known as pipeline conflicts or stalls, occur in a pipelined processor architecture when the next instruc. can't execute in the following clock cycle.
- These hazards can cause delays in the instruction pipeline, reducing the overall performance and efficiency of the CPU.

Types:- Str... D... Co...
Structural Ha...
→ Str. hazards o... hardware reso... bcz. the ha... operations ne...
→ Example:- If a proce... instruc. fe... may occur ... access m...
Data Ha...
→ Arise w... close to... incorrec...
→ Types:- 1. RAW ... depe... not ... AD... SU...
2. WA... to ... it.
3. W... to ...

Types:
- Structural Hazards
- Data Hazards
- Control Hazards.

Structural Hazards :-

→ Str. hazards occur when two or more instr. req. the same hardware resource at the same time. This conflict arises bcz. the hardware can't support all the concurrent operations needed by the pipeline.

→ **Example :-**

If a processor has a single memory unit for both instruc. fetch & data read/write, a str. hazard may occur if an instr. fetch & a data memory access need to happen simultaneously.

Data Hazards :-

→ Arise when instr. that exhibit data dependencies are close together in the pipeline. This dependency can cause incorrect results or the use of stale data.

→ **Types :-**

1. **RAW (Read After Write)** : Occurs when an instruction depends on the result of a prev. instruc. that has not yet completed.

```
ADD R1, R2, R3   // instruc. 1: R1 = R2 + R3
SUB R4, R1, R5   // instruc. 2: R4 = R1 - R5 (R1 not yet written)
```

2. **WAR (Write After Read)**: Occurs when an instruc. writes to a destination before a prev. instr. has read from it.

```
MOV R1, R4       // instr. 1 : R4 = R1
ADD R4, R2, R3   // instr. 2 : R4 = R2 + R3 (R4 read before written)
```

3. **WAW (Write After Write)**: Occurs when 2 instr. write to the same destination in a diff. order than programm.

```
ADD R1, R2, R3   // Instr. 1: R1 = R2 + R3
                 // Instr. 2: R1 = R4 (Instr. 1 written last)
```

Control Hazards :-
Control Hazards occur due to the pipelining of branch instructions, it may fetch the wrong set of instr. if the branch decision is not yet resolved.
Example :-
BEQ R1, R2, Label // Branch if R1 == R2 to label
ADD R3, R4, R5 // This instr. might be fetched before the branch resolves.

(6) a) What is SPEC rating?
⇒ SPEC (Standard Performance Corporation)

(b) Use Bernstein's conditions for determining the max. parallelism in the following sequence of instr.

A = B × C
B = D + E
C = A + B
E = F - D

I1 : A = B × C
I2 : B = D + E
I3 : C = A + B
I4 : E = F - D

Now, read set and write set of I1, I2, I3, I4 are as follows :-

R1 : {B, C} W1 : {A}
R2 : {D, E} W2 : {B}
R3 : {A, B} W3 : {C}
R4 : {F, D} W4 : {E}

Now, let's find out whether I1 and I2 are parallel or not :-

R1 ∩ W2 ≠
R2 ∩ W1 =
W1 ∩ W2 =

Similarly,
R1 ∩ W3 ≠
R3 ∩ W1 ≠
W1 ∩ W3 =
Hence I1 a

Similarly
R1 ∩ W
R4 ∩ W
W1 ∩ W

For I2
R2
R3
W2

For I
R2 ∩
R4 ∩
W2 ∩

For I
R3 ∩
R4 ∩
R3

REDMI 9 - POWER
SANJANA MATHUR
2024.6.4 11:14

$R1 \cap W2 \neq \phi$ That means I1 and I2 are not-
$R2 \cap W1 = \phi$ independent of each other.
$W1 \cap W2 = \phi$

Similarly, for I1 || I3,
$R1 \cap W3 \neq \phi$
$R3 \cap W1 \neq \phi$
$W1 \cap W3 = \phi$
Hence I1 and I3 are not independent of each other.

Similarly for I1 || I4,
$R1 \cap W4 = \phi$ Hence I1 and I4 are
$R4 \cap W1 = \phi$ independent of each other.
$W1 \cap W4 = \phi$

For I2 || I3, Hence, I2 and I3 are not-
$R2 \cap W3 = \phi$ independent of each other.
$R3 \cap W2 \neq \phi$
$W2 \cap W3 = \phi$

For I2 || I4, Hence, I2 and I4 are not-
$R2 \cap W4 \neq \phi$ independent of each other.
$R4 \cap W2 = \phi$
$W2 \cap W4 = \phi$

For I3 || I4, Hence, I3 and I4 are
$R3 \cap W4 = \phi$ independent of each other.
$R4 \cap W3 = \phi$
$RW3 \cap W4 = \phi$

Regd°
Date
Page

⑦ Show that the max. Speedup of a pipeline is eq. to its stages.

→ Suppose:- k = no. of stages in pipeline.
n = no. of processes to be execute.
τ = time delay for each stage of pipeline.
and S = Speed-up

then,

$$S = \frac{\text{Time req. for non-pipeline process}}{\text{Time req. for pipeline process}}$$

$$= \frac{n \cdot k \cdot \tau}{(k+(n-1))\tau} = \frac{n \cdot k}{k+(n-1)}$$

Max. Speed-up is, $S_k \longleftrightarrow k$ when $n >> k$.

- The max. speed-up is never fully achievable bcz. of data-dependencies b/w instruc^n., interrupts, prog. branches, etc. So, many pipeline cycles may be wasted on a waiting state caused by out-of sequence instr. executions.

⑧ Draw pipeline execution diagram during the execution of the following instructions:

MUL R1, R2, R3
ADD R2, R3, R4
INC R4
SUB R6, R3, R7

Find out the delay in pipeline execution due to data dependency of the above instructions.

→ Above pipeline technique is a 5 Stage pipeline.
fetch, decode, read, execution, write back.

Operation of the instructions are,

R1 ← R2.R3
R2 ← R3+R4
R4 ← R4+1
R6 ← R3*R7

So, No data hazard will occur in the pipeline. Instructions are not dependent to each other. So, normal pipeline execution will occur.

Clock	1	2	3	4	5	6	7	8
Fetch →	MUL R1,R2,R3	ADD R2,R3,R4	INC R4	SUB R6,R3,R7				
Decode →		MUL R1,R2,R3	ADD R2,R3,R4	INC R4	SUB R6,R3,R7			
Read →			MUL R1,R2,R3	ADD R2,R3,R4	INC R4	SUB R6,R3,R7		
Execution				MUL R1,R2,R3	ADD R2,R3,R4	INC R4	SUB R6,R3,R7	
Write back →					MUL R1,R2,R3	ADD R2,R3,R4	INC R4	SUB R6,R3,R7

Q) How Reservation table helps to study the performance of pipeline?

→ i) There are 2 types of pipeline - Static and Dynamic.
Static → can perform one function at a time.
Dynamic → can perform more than one function at a time.

ii) A pipeline reservation table shows when stages of a pipeline are in use for a particular function. Each stage is represented by a row in reservation table. Each row of the reserv? table is in turn broken into columns, one per clock cycle.

(iii) When scheduling a static pipeline, only collisions b/w diff. input data for a particular func" had to be avoided.
With a dynamic pipeline, it is possible for diff. input data req. diff. func" to be present in the pipeline at the same time.
∴ collisions b/w these data must be considered as well.

(10) What is cache coherence problem? Describe one method to remove this problem and its limitations.
→ Cache coherence problem arises in multiprocessor syst. where each processor has its own cache memory. When one processor updates a memory location, other processors' caches might still have outdated copies, leading to inconsistency.
Resolving method → MESI protocol.
Modified, Exclusive, Shared, Invalid.
In this protocol, each cache line is marked with one of these states, indicating the status of data in that cache line; ~~if miss~~
When a processor wants to modify a cache line, it must 1st gain exclusive access, ensuring no other processor has a copy. Once done, it updates the data & marks it as Modified.

Limitations of MESI:
It requires significant overhead to track & manage cache coherence, which can affect overall system performance.

(11) a) Write do
→ Amdahl's law
improvement
a system.
S → theoreti
P → propor
N → No.

b) What
handle
→ Branch h
processors
on bran
incorrect
pipeline
correct
processo
Method
1. Branch

2. Delay

Both
impa

(11) a) Write down Amdahl's law of parallel processing.

→ Amdahl's law is a formula used to find the max. improvement possible by improving a particular part of a system. $S = \frac{1}{(1-P) + \frac{P}{N}}$

S → theoretical speedup of the execution of the task.
P → proportion of the program that can be parallelized.
N → No. of processors.

b) What is branch hazard? Briefly discuss 2 methods to handle branch hazards.

→ Branch hazard (control hazards) - occur in pipelined processors when the pipeline makes the wrong decision on branch prediction, leading to the fetching of incorrect instructions. This can cause delays as the pipeline has to discard these instructions & fetch the correct ones, thus reducing the efficiency of the processor.

Methods to handle Branch Hazards:-

1. Branch prediction → Static branch prediction - "backward branches are taken" (common in loops).
   → Dynamic Branch prediction - 2-bit predictor or more advanced techniques.
2. Delayed Branching → Rearranges the instruction seq. so that the instr. following the branch is not affected by the branch decision.

Both methods aim to minimize the performance impact of branch hazards.

(12) What is the drawback of directed mapped cache? How is it resolved in set associative cache?

→ A drawback of direct-mapped cache is the issue of conflict misses.
This occurs when multiple memory addresses map to the same cache line, causing the cache to constantly evict & replace the data, even if there is available space in other parts of the cache.

Resolution:-
Set associative cache helps to resolve this problem by allowing each memory address to map to multiple possible cache lines within a "Set", rather than a single cache line.
How it works:- (1) Multiple ways per set-
(2) Flexible Placement
(3) Reduced conflict Misses

(13) Explain the main factors that can influence the performance of interconnection networks.

→ i) Topology
ii) Routing Algorithm
iii) Flow control Mechanism
vii) Scalability
ix) Traffic patterns
iv) Switching Technique
v) Bandwidth
vi) Lateney
viii) Fault Tolerance
x) Implementation factors.

1. Topology
- Direct vs. Indirect → Direct topologies (eg. meshes, tori) connect nodes directly, while indirect topologies (eg. crossbars) use intermediate nodes or switches.

• Degree of connectivity - The no. of connections per node impacts bandwidth & fault tolerance.

2. Routing Algorithm:-
• Deterministic vs. Adaptive:- Deterministic routing uses fixed paths, which simplifies implementation but can lead to congestion.
Adaptive routing can dynamically change the paths based on network conditions, improving performance but ↑ complexity.

• Minimal vs. Non-minimal:- Minimal routing uses the shortest path, which minimizes latency, while non-minimal routing can balance load better, potentially avoiding congestion at the cost of longer paths.

3. Latency:-
• Propagation delay → Time taken for a signal to travel from source to destination, influenced by physical distance & medium.
• Switching delay → Time taken by switches to process & forward data.
• Queuing delay → Time data spends waiting in buffers due to congestion.

(14) Define vector chaining? How can it speed up the processing? Explain with suitable ex.
→ Vector chaining, also known as chaining vector operations, is a technique used in vector processors to enhance performance by efficiently overlapping the execution of multiple vector instructions.

Date ____
Page ____

Benefits of vector chaining →

1. Reduced Latency → time req. to complete a seq. of operations is reduced.
2. Improved Resource Utilisation → reduces idle times b/w operations.
3. Increased Throughput → Overall rate of computation ↑.

Ex. of vector chaining :-

1. C = A + B
2. D = C × E

Without vector chaining, processor would :-

1. Perform the addⁿ. C = A + B.
2. Wait for the entire result of C to be available.
3. Perform the multiplication D = C × E.

Illustration :-

Assume vectors A, B and E each have 4 elements.

- $A = [a_1, a_2, a_3, a_4]$
- $B = [b_1, b_2, b_3, b_4]$
- $E = [e_1, e_2, e_3, e_4]$

Without chaining :-

1. Compute $C = [a_1+b_1, a_2+b_2, a_3+b_3, a_4+b_4]$.
   This step takes one unit of time.

2. Compute $D = [c_1 \times e_1, c_2 \times e_2, c_3 \times e_3, c_4 \times e_4]$.
   This step takes another unit of time after the 1st is complete.

Total time = 2 units.

With chaining :-

1. Start computing $C = [a_1+b_1, a_2+b_2, a_3+b_3, a_4+b_4]$.

2. As soo
[$c_1 \times e_1$,
To

(15) Explain
of comp
→ Flynn's
classifica
concurre
4 ca

1. [SISD].
instrucⁿ.
• ex. -

~~2. [SIMD]~~

2. [SIMD]
instruct
Simult
• ex. -

2. As soon as $c_1 = a_1 + b_1$ is available, start D:-
$[c_1 \times e_1, c_2 \times e_2, c_3 \times e_3, c_4 \times e_4]$.
Total effective time ≈ 1 unit.

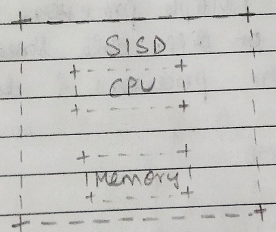
(15) Explain in brief with neat diagram the Flynn's classific" of computers.

→ Flynn's taxonomy by Michael J. Flynn in 1966, is a classification system for C.A. based on the no. of concurrent instruc" & data streams they can handle.

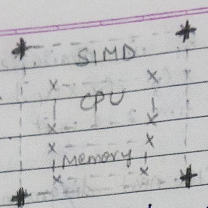
4 categories:
- → SISD (Single Instruction, Single Data)
- → SIMD (Single Instruction, Multiple Data)
- → MISD (Multiple Instruction, Single Data)
- → MIMD (Multiple Instruction, Multiple Data).

1. **SISD** · Description - A single processor executes a single instruc" stream & operates on a single data stream.
· ex. - Traditional single-core processors.

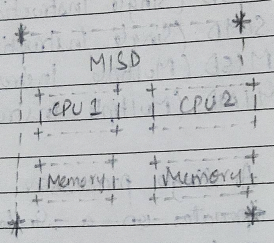
~~2. SIMD~~



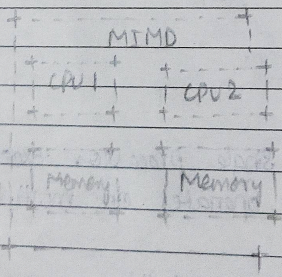
2. **SIMD** · Description - A single processor executes a single instruction stream but operates on multiple data streams simultaneously.
· ex - Vector processors & GPUs.



3. MISD · Description: Multiple processors execute diff. instruc. streams on same data stream.
· ex- Rare in practical applications, theoretically used in fault-tolerant systems.



4. MIMD · Description: Multiple processors execute diff. instruction streams on diff. data streams.
· Example: Multi-core processors & distributed systems



16) Explai
for
→ In
can
Two

C- A
Commer
local
· La
V
me
· Ac
one
· E
the

S-
· Ve
then
add
· Lo
sto
· Ac
· E
Sh

(16) Explain the C-access and S-access memory organization for vector accesses

→ In context of vector processors, memory access patterns can significantly impact performance.
Two common patterns are :- C-access (Column access)
S-access (Stride access).

C-Access

- Elements of a vector are stored in consecutive memory locations.
- Layout :- If you have a vector V with elements $V_1, V_2, \ldots, V_n$, may are stored in contiguous memory addresses.
- Access Pattern :- Accessing elements sequentially, i.e., one after another.
- Example :- for a vector $V = [V_1, V_2, V_3, \ldots, V_n]$, the memory addresses might look like :-

```
Address 0: V_1
Address 1: V_2
Address 2: V_3
...
Address n-1: V_n
```

S-access

- Vectors elements are accessed with a fixed stride, meaning there is a regular interval (or gap) btw the memory addresses of consecutive Element.
- Layout :- if the Stride S is 2, the elements might be stored in memory locations 0, 2, 4, etc.
- Access Pattern :- Accessing elements with a stride S.
- Example :- for a vector $V = [V_1, V_2, V_3, \ldots, V_n]$ and a Stride S = 2, the memory address might look like :-

Address 0: v_1
Address 2: v_2
Address 4: v_3
...
Address 2(n-1): v_n

Comparison	C-access	S-access
	→ Typically faster → common in applic° with dense data & where elements are accessed sequentially.	→ Can be slower. → useful in matrix operation (rows, columns)

→ A = [a11, a12, a13, a21, a22, a23, a31, a32, a33]
Access pattern:- (C-access)
a11, a12, a13, a21, a22, a23, a31, a32, a33
Access pattern:- (S-access)
a11, a21, a31, a12, a22, a32, a13, a23, a33.

(17) Diff. b/w 3-address, 2-address, 1-address, and 0-add. instruction with suitable ex.

1. Three-Address Instructions-
each instruc°. specifies 3-addresses. These are used for specifying 2 source operands & one destination operand.
ex- assembly lang. for modern CPUs.

ADD R1, R2, R3

- Add contents of registers 'R2' and 'R3'
- Store the result in 'R1'.

2. Two-Address Instructions-
each instruc°. specifies 2 addresses. One address serves as both a source & a destination, while the other is just a source.

ADD R1, R2
R1 is both destination & one of source registers.
R2 is other source register.

3. One-Address Instructions:
There is only one explicit address in the instruction, which usually acts as both operand & accumulator.
ADD R1
ACC = ACC + R1
ACC → accumulator register
R1 → Source register.

4. Zero-address:
Don't contain any explicit address fields. These instructions typically operate on a stack-based architecture, where operands are implicitly assumed to be on top of stack.
ADD
TOS = (TOS - 1) + TOS
TOS → Top of Stack
The 2 operands are poped from stack, added, and the result is pushed back onto the stack.

(18) What is Instruction cycle? Compare & contrast hardwired vs. micro programmed control unit.
→ Instruc". cycle is the process through which a comp. retrieves, decode, and execute a command.
This cycle can be broken down into several phase.
1. Fetch - control unit retrieves instruc". from memory.
2. Decode - Instruc". is decoded to understand what action is req.
3. Execute - Decoded instruc". is carried out, which may involve arithmetic operations, memory access, or input/output operations.

4. Writeback: The results of the execution are written back to the system.

Features	Hardwired C.U.	Microprogrammed C.U.
Speed	Faster due to direct execution.	Slower due to sequence of microinstruction
Flexibility	Low.	High.
Complexity	Simple	Moderate complexity
Implementation	uses combinational logic circuits	uses seq. of microinstr. in control memory.
Instruc. set	used in RISC	used in CISC architec.

(19) What is Von-Neumann architecture? What is a Von-Neumann bottleneck? How can this be reduced?

→ Also called Von Neumann model or Princeton architecture, is a computer architecture design model proposed by John Von Neumann.

1. CPU → executes instruc. & processes data.
   - ALU :- Performs arithmetic & logic operations.
   - CU :- Directs operation of the processor.
2. Memory → Stores data & instructions.
3. Input/output devices → Interfaces for data input and output.
4. Bus system → A communic. system that transfers data b/w components.

Von Neumann Bottleneck :-
Refers to the limitation inherent in Von Neumann architecture, where the throughput of the system

is limited by the rate at which data & instructions can be fetched from memory.

Reducing Von Neumann Bottleneck :-

1. Cache Memory
- L1, L2, L3 caches are small, fast memory storage areas located closer to CPU.
They temporarily hold frequently accessed data & instruct. to reduce the time the CPU spends waiting for data from main memory.

- Cache Hierarchy:- Diff. levels of cache (L1 being the fastest & smallest, L3 being the largest & slowest) provide a balance betw speed & size.

2. Pipelining
- Instruction pipeline :- Divides the process of executing instructions into seperate stages, allowing multiple instructions to be processed simultaneously at diff. stages of execution.

3. Direct Memory Access (DMA):-
- DMA ~~cntt~~ controllers:- Allow peripheral devices to access the main memory directly without involving the CPU, reducing the CPU's workload & improving overall system performance.

(20) Compare RISC and CISC architecture in brief.

Features	RISC (Reduced Instruc? Set Computer)	CISC (Complex Instruc? Set Computer)
Instruc? set →	Small, highly optimized set.	→ Large set of instructions
Instruc? complexity →	Simple instruc? taking 1 clock cycle.	→ Complex Instructions taking multiple clock cycle.
Performance →	high	→ slower
code size →	Larger	→ Smaller
control unit →	Hardwired	→ Microprogrammed.
Register usage →	heavy use of registers, with many general purpose registers.	→ fewer registers, more use of memory.
Examples →	ARM, MIPS, SPARC	→ x86, VAX, System/360
use case →	Smartphones, Tablets.	→ Desktops, Servers, etc.

2023.

Regals
Date
Page

① i) Superscalar processor is a processor architecture that executes multiple instructions in a single clock cycle by using multiple functional units.
(computer Network)

ii) A Cluster computer is a group of interconnected comp. that work together to perform a task.

iii) Distributed. architecture is a type of parallel architecture that emphasizes data flow and emphasizes communication b/w nodes rather than a shared memory space.

iv) Pipelining is a technique for increasing instruction-level parallelism by simultaneously executing multiple instructions.

v) The ↑° goal of ~~its~~ exception handling is to avoid the program or system crashing.

vi) The FIFO policy in virtual memory is used to decide which page to remove from memory when a new page needs to be loaded into the memory.

vii) What is Superpipelined processor?
→ It is a processor that ↑ instruction throughput by dividing its pipeline into more stages, allowing higher clock speeds & multiple instructions to be processed simultaneously.

viii) What is synchronization in a Centralized architecture?

→ In a centralized architecture, Synchronization refers to the coordination of concurrent processes or threads to ensure they operate correctly & efficiently, often by controlling access to shared resources to prevent conflicts & ensure data consistency.

ix) What is exception handling in CA?
→ Refers to the management & coordination of the execution of instructions within a computer's CPU.

x) What is a memory replacement policies?
→ are algorithms used in operating systems to decide which memory pages to swap out when new pages need to be loaded into memory.

xi) What are memory replacement policies?
→ FIFO → Oldest data block in the cache is removed first. It is simple to implement but may not always provide the best performance.

→ LRU (Least Recently Used) → Oldest Data block that has not been used for the longest period is replaced. This policy assumes that data used recently will likely be used again soon.

→ MRU (Most Recently used) → This can be useful in certain scenarios where the most recently used data is unlikely to be used again soon.

→ LFU (Least Frequently used)
→ Random Replacement –
→ Clock (Second Chance)

xii) What is
→ VLI
architecth
Multiple

(2) What is
memory
→ Cache
computer
frequent
located

1. Siz
con
2. Sp
the
3. P
to
au
4. C
of
its

(3) Wha
imp
1. Inst
2. Reso
3. In
4. Im
5. Dep

(vii) What is a VLIW processor?

→ VLIW (Very Long Instruction Word) is a processor architecture where a single instruction can execute multiple operations simultaneously.

(2) What is cache memory, how it is diff. from main memory?

→ Cache memory is a small, high-speed type of volatile computer memory that provides quick access to frequently used data & instructions.
Located on CPU chip or very close to it.

1. Size:- Cache memory is much smaller in size compared to main memory.
2. Speed:- Cache memory has faster access speeds than main memory.
3. Proximity to CPU:- Cache memory is located closer to CPU, whereas main memory is typically located away on the motherboard.
4. Cost:- Cache memory is more expensive per unit of storage compared to main memory due to its faster speed & proximity to the CPU.

(3) What are some of the design trade-offs involved in implementing a superscalar processor?

1. Instruction-Level Parallelism (ILP) vs. Hardware Complexity
2. Resource sharing vs. Latency
3. Instruction fetch Bandwidth vs. Branch Prediction Accuracy
4. Instruction Dispatch & Issue Logic.
5. Dynamic Power consumption vs. Performance.

(4) What is cache miss, and what are the causes of cache misses in a computer system?

→ A cache miss occurs when the data requested by the processor is not found in the cache memory, leading to a longer access time as it's fetched from a slower main memory.

(5) Role of memory management unit (MMU) in virtual memory management.

(6) a) Describe virtual memory.
b) Explain how it is implemented in modern comp. systems.

(7) Compare & contrast Superscalar & VLIW processor architecture

(11) A system has 64-bit virtual address space, and 4KB page size. The page table is stored in main memory, which has a memory access time of 100ns. The TLB has a hit rate of 90%. & a TLB miss takes 500ns to service. What is the effective memory access time?