1.) Designing a multi processor piplined system involves various parameters that can Impact it's performance. There are some parameters that can result in good performance :->

a) Clock frequency :->

The clock frequency determines how many clock cycles per second the system can execute. A higher clock frequency can lead to higher performance, but it also increases power consumption.

b) Number of pipeline stages :->

The number of pipeline stages determines the number of instructions that can be executed in parallel. A higher number of pipelines stages can increase performance, but it also increase the system's complexity and power consumption.

c) Instruction set Architecture :->

It determines the number and types of instruction that can be executed. A well-designed ISA can simplify the pipeline implementation and improve performance.

d) Cache Organization :->

It determines the size and structure of the memory hiearchy. A larger cache can reduce the number of memory access and improve performance. but it also increase the complexity.

e) Thread level parallelism :->

The thread level parallelism determines the amount of parallelism that can be achived between multiple threads. A high Thread Level prallelism can improved performance, but it also requires more complex design.

f) load Balancing !->

It refers to the ability of the system to evenly distribute taskes processors.

g) <u>Fault tolerance :-></u>

It refers to the ability of system to continue functioning in the event of a hardware or software failure.

h) <u>Data dependency handling :-></u>

Data dependency handling determines how the system handels data depencies between instructions .: A well designed data dependency handling mechanism can reduce pipeline states and improves performance.

overall, desiging a pipelined multi processor system involves balancing multiple parameters to achieve good performance. The above key parametes can impact the systems performance and complexity.

2)

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| $S_1$ | X | | | | | X | | X |
| $S_2$ | | X | | X | | | | |
| $S_3$ | | | X | | X | | X | |

evalution time = 8

Forbidden latency for $S_1$ = $\{5, 2, 7\}$

forbidden latency for $S_2$ = $\{2\}$

forbidden latency for $S_3$ = $\{2, 4\}$

∴ Forbidden latency = $\{2, 4, 5, 7\}$

Permissible latency = $\{1, 3, 6, 8^+\}$

∴ Initial collision vector (ICV) = $\{\overset{c_7}{1}\,\overset{c_6}{0}\,\overset{c_5}{1}\,\overset{c_4}{1}\,\overset{c_3}{0}\,\overset{c_2}{1}\,\overset{c_1}{0}\}$

Now, for latency '1'

$$\begin{array}{rr} & 0101101 \\ (ICV) & 1011010 \\ \hline & 1111111 \\ \hline \end{array}$$

for latency '3'

$$\begin{array}{rr} & 0001011 \\ (ICV) & 1011010 \\ \hline & 1011011 \\ \hline \end{array}$$

for latency '6'

$$\begin{array}{rr} & 0000001 \\ (ICV) & 1011010 \\ \hline & 1011011 \\ \hline \end{array}$$

for latency '8'

$$\begin{array}{r} 0000000 \\ 1011010 \\ \hline 1011010 \\ \hline \end{array}$$

Now, present state → 1011011

for P.S permissible latency - $\{3, 6, 8^+\}$

ICV → 1011010

(latency 3)

$$\begin{array}{r} 1011010 \\ 0001011 \\ \hline 1011011 \\ \hline \end{array}$$

(latency 6)

$$\begin{array}{r} 1011010 \\ 0000001 \\ \hline 1011011 \\ \hline \end{array}$$

latency 8

$$\begin{array}{r} 1011010 \\ 0000000 \\ \hline 1011010 \\ \hline \end{array}$$

Now, P.S → 1111111 for that permissible latency 8+ which return to ICV.

<u>state diagram</u>

$$\begin{tikzcd}
 & \boxed{1011010} \arrow[dl, "1^*"'] \arrow[dr, "6"] \arrow[loop above, "8^+"] & \\
\boxed{1111111} \arrow[ur, bend right, "8^+"'] & & \boxed{1011011} \arrow[ul, "3"'] \arrow[ul, bend right, "8^+"'] \arrow[loop below, "3^*"] \arrow[loop right, "6"]
\end{tikzcd}$$

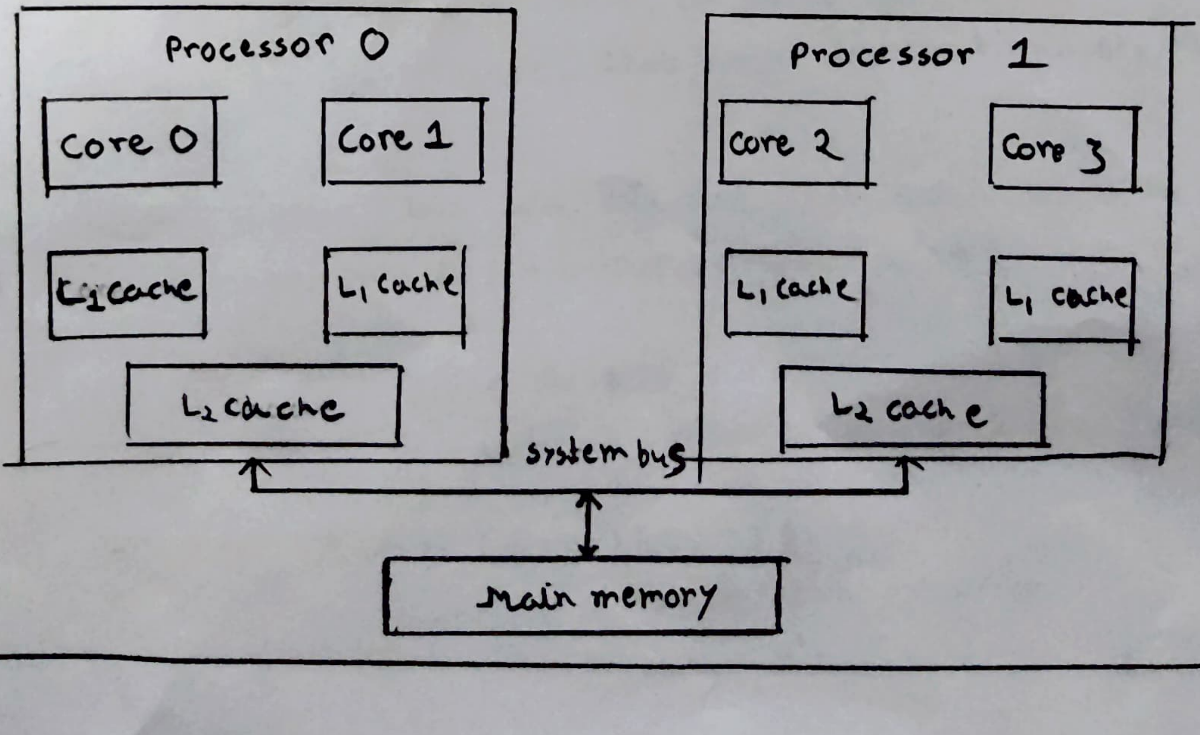
| simple cycle | Avg latency |
|---|---|
| 3 | 3 |
| 6 | 6 |
| 8 | 8 |
| (1,8) | 4.5 |
| (3,8) | 5.5 |
| (6,8) | 7 |

greedy cycle (3) , (1,8)

MAL (maximum Average latency) is 3.

3) A multi-core computer architecture is a type of computer architecture with two or more processors connected to a single chip for faster simalteneous processing of several tasks, reduced power consumption, and for greater performance generally, each core is capable of executing instructions independently allowing for parallale exection of multiple tasks of thread.

Simplified structure of multi-core computer Architecture



It enables the communication between all available cores and they decide all processing duties propriately. The processed data from each core's transmitted back to computer main board via single common gateway once all of processing operations have been finished. This method beat a single-core cpu in terms of total performance.

Multicore computer architecture achieves parallelism by dividing tasks into multiple smaller taskes that can be executed simulteneously on different cores.

for example : A program may be split into multiple threads that can be executed in parallel on different cores. This allows for faster execution of the program, as each threat can be executed independently and does not have to wait for other thread to finish. The use of multicore also helps to avoid processor battleneck.

An infinite loop is a programming construct where a section of code continually executes without even exiting. This can occure unintentionally in code due to logical errors, such as an incorrect statement, or as a delibarate design choice such as in a server that needs to continuously listen for incoming.

To handel this we can use try_catch_block to catch the exeeption & take appropriate action.

```
try {  while(true){
                // some code preformed repeatedly
       }
   }catch (Exeption e) {
              // handle the exeption
   }
```

In this example, the while loop is an infinite loop that continues to execute until exception thrown. The try block contains the code that might throw an exception, and the catch-block contains the code that handel the exception is it is thrown.

If error is inside the try block the loop will be interrupted and control will be passed to catch block. catch block then takes the necessary actions.

By try-catch block, we can reduce processor cossumption by interrupting the loop when an exception is thrown rather than allowing it to continue indefinitly. This can help to prevent sytem crashes and improve overall performance.

4) Using Amdhal's law

The overall speed up (N) $= \dfrac{1}{(1-\text{fraction enhanced}) + \dfrac{\text{fraction enanced}}{\text{speedup enhanced}}}$

fraction enhanced = 40%
= 0.40

speed up enhanced = 20

∴ overall speedup (N) $= \dfrac{1}{(1-0.40) + \dfrac{0.40}{20}}$

$= \dfrac{1}{0.60 + 0.02}$

$= \dfrac{1}{0.62}$

$= 1.612$

∴ The overall speedup gained by incorparating the enhancement is 1.612. (Ans)