# Performance Analysis

## Abstract

The Report Investigates about the Performance Analysis on different Architectures. In this report I mainly concentrate on Performance Analysis on Different Architectures in a Multi Core System. The report mainly focuses on the best architecture in terms of Execution time/Performance, Instruction Count, No of Memory Access and Size of Instruction in a Multi Core Architecture. The report is supported with the simulation I have done on different Architectures. The details about the simulation is included below. It is concluded that the Memory Memory Architecture is the best Architecture in terms of performance. It is also suggested that the Load Store Architecture is good enough to perform operations with the reduced Instruction size.

## Introduction

The purpose of this report is to survey about the best Architecture in multi core Processor Systems (in terms of Performance). By examining the results obtained from my simulation this report describes about all the architectures performances in multi core architectures in relation to four criteria Instruction count, No of Clock Cycles, No of Memory Access and Size occupied by an Instruction. The six architectures that are discussed in here are

1. Three Address Architecture
2. Two Address Architecture
3. One Address Architecture (Accumulator)
4. Zero Address Architecture (Stack)
5. Register Memory Architecture(GPR)
6. Load Store Architecture(GPR)

## Simulation

In the Simulation process I have took more than 100 C Instructions. These C instructions are converted to different Architecture Instructions. These Instructions are Stored in different files. These files are further simulated into an 8 Core System such that the instructions are divided into 8 Cores. I have divided the Instructions from a single core processor into an 8 core multi-processor using the round robin technique. The obtained Instructions for each core are displayed in the files. Both the Performance results for a Single Core is compared with the Performance results of Multi Core Architecture.

Performance Analysis 1(Single Core Processor)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 3 Address/ Memory Memory | 2 Address Architecture | 1 Address / Accumulator | 0 Address /Stack | Register Memory | Load Store |
| Size of Instruction | 18704 | 26560 | 18456 | 7856 | 20010 | 26474 |
| No Of Memory Access | 1002 | 1328 | 769 | 648 | 690 | 648 |
| Instruction Count | 334 | 664 | 769 | 982 | 690 | 982 |
| Clock Cycles | 668 | 1328 | 1538 | 1964 | 1380 | 1964 |

**Performance Analysis 2(Multi Core Processor (Single Core Performance))**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 3 Address/ Memory Memory | 2 Address Architecture | 1 Address / Accumulator | 0 Address /Stack | Register Memory | Load Store |
| Size of Instruction | 1792 | 2520 | 1728 | 736 | 1885 | 2944 |
| No Of Memory Access | 96 | 126 | 72 | 66 | 65 | 66 |
| Instruction Count | 32 | 63 | 72 | 92 | 65 | 92 |
| Clock Cycles | 64 | 126 | 144 | 184 | 130 | 182 |

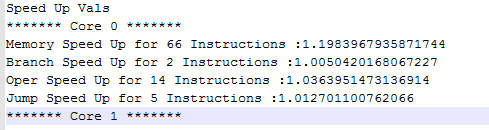
As you can see the above are the two tables for Single and Multi core Processor. In the Multicore Processor table as you can see even though it’s an 8 Core Processor I have took a single Core performance to compare the clock cycle values. As you the Equivalent result from the Multi Core Processor is 8x times of table 2. But the No of Clock Cycles will be the same as the Cores are run in Parallel 1x times of Clock Cycles for all 8 Cores. The Performance Analysis results from Multi core is compared with Single core processor Performance. Here we mainly deal with the No. of clock cycles (the Execution time). The Graph for the Clock Cycles for Single and Multicore are shown below

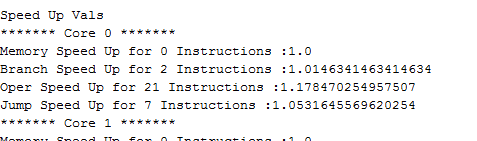
As the graph depicts if the Number of cores get increases the parallel execution of different cores also get increased. This reduces the Program execution time. Thereby takes less No. of clock cycles.

As said in the performance analysis for Single Core Architecture the Memory Memory takes less No. of clock cycles to execute a program. Similarly in the multi core processor the same holds here. The other best architecture is Load Store which has the reduced set of Instructions**.**

## Overall Speedup

Now lets compare both the Load Store and Memory Memory Architectures. As Memory Memory Architecture is always Memory Intensive and whereas the Load Store Architecture is mostly depends on registers. Basically in Load Store Architecture mostly the Load and Store Instructions are relevantly used so whenever the speed up for Load and Store are increased the overall speed up gets increased. It can be inferred from the simulation the Speed up values that I have obtained for my core 0 of Load Store architecture:

As you can see the overall speed up has increased by 19.8% when the Memory Instructions are speeded up by 30% rather than other Instructions.

Similarly my simulation also deals with the Memory Memory Architecture. The overall speed is increased for the Operation Instructions as there are no Memory Instructions in this architecture. As you can notice as there are 0 Memory Instruction Speed up in Memory Instructions doesn’t increment the over all speed up. But do notice that the speed up of 30 % in Operation Instructions increased the overall speed up by 17.8 %.

As Number of pages been a constraint for the report only the important (high performance) two Architectures are displayed the other Architectures values can inferred from the output files.

## Communication Cost

Communication Cost plays an important role in the Multi Core processor. The communication cost plays an important role when an optimized compiler is used. Even running in multi core it needs to be seen that the result obtained from single core should be same as that of multi core processor. In order to do that Communication between the cores is very important. This importance costs some execution time.

Now from this we can derive an Expression

Execution time in Multi Core Processor=Execution Time in Single Core Processor/No of Cores.

**EM=ES/No of Cores.**

Communication cost takes some Execution time let the time taken for Communication be C the Equation become

**EM=(ES/No of Cores) + C**

Note: It is assumed that the all cores are running continuously without any core sleeping.

## Power

The Simulation mainly considers only processing without any overheads like power. When the processor is built it should also be noticed that the non-functional things like Power also come into play. As the No of Cores increase the power consumed by the Processor increases in same manner. The Heat dissipation from the processor also increases when the cores are run continuously which can even melt the processor. This led to a theory of switching between processors. This switching between processors also adds an overhead to the Execution time.

From the above Equation

**EM=(ES/No of Cores) + C**

As some cores sleep due to the reason of processor overheating. The equation is changed to following

**EM=(ES/No of Cores active at this time) + C**

Let the Switching between processor adds an overhead of S the equation becomes

**EM=(ES/No of Cores active at this time) + C + S**

### Conclusion

In my earlier performance analysis for single core Processor it was said that the Memory Memory Architecture is the best as the Instruction count is very less than any other architecture, due to this the Execution time for a program is decreased. Even though every operation takes almost 3 memory access, when considered with a very large program the Execution time is better. Even with the memory access over head and size of Instruction Overhead the Memory Memory Architecture is better than any other architecture. The second best Architecture is the Load Store Architecture because of its Reduced Instruction Size, separate Instructions for Memory and fixed length Instruction.

Even though the Multi Core Processors are same as that of Single Core Processors, they run in parallel. If it’s a 8 core processor the it performs 8 x times faster but these 8 cores are nothing but 8 Single Cores. The same Answer applies but the Execution time is not 8 x times faster because the outputs change when anything run in parallel (Data dependency) to reduce the hazards from Data Dependencies Compiler Optimization is done, when Compiler optimization is done a little bit of execution time overhead is added like stalling the processor until I get a result from the other core. The Performance equation again changes to

**EM=(ES/No of Cores Active at this Instance) + C + S + Stalls due to optimization**