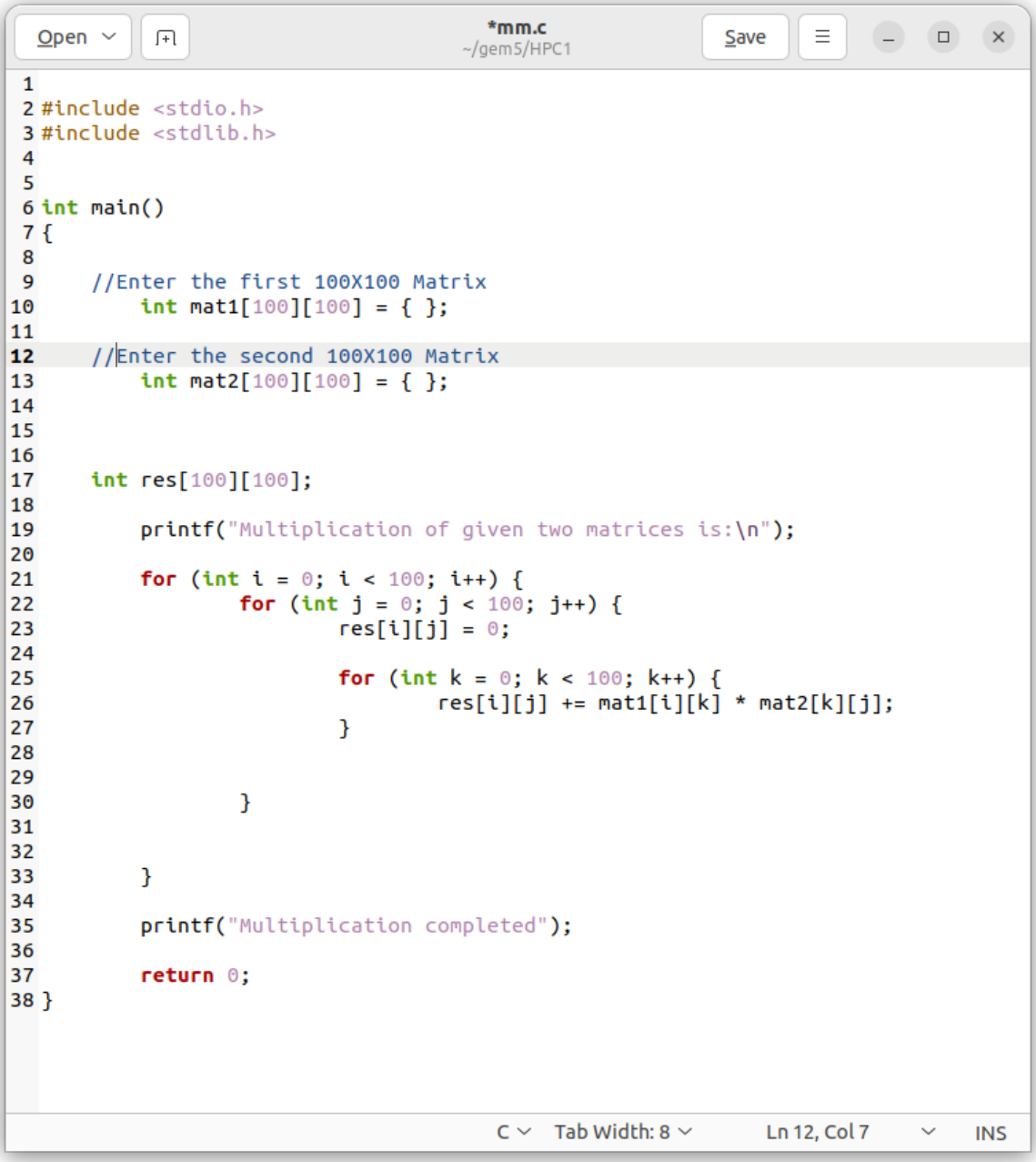
Lab One: Familiarization with gem5

Code for Matrix Multiplication-




```c

#include <stdio.h>
#include <stdlib.h>


int main()
{

    //Enter the first 100X100 Matrix
        int mat1[100][100] = { };

    //Enter the second 100X100 Matrix
        int mat2[100][100] = { };



    int res[100][100];

        printf("Multiplication of given two matrices is:\n");

        for (int i = 0; i < 100; i++) {
                for (int j = 0; j < 100; j++) {
                        res[i][j] = 0;

                        for (int k = 0; k < 100; k++) {
                                res[i][j] += mat1[i][k] * mat2[k][j];
                        }


                }


        }

        printf("Multiplication completed");

        return 0;
}
```

Code for config.py file-

Open | config.py ~/gem5/HPC1 | Save

```
from gem5.components.boards.simple_board import SimpleBoard
from gem5.components.cachehierarchies.classic.no_cache import NoCache
from gem5.components.memory.single_channel import SingleChannelDDR3_1600
from gem5.components.processors.simple_processor import SimpleProcessor
from gem5.components.processors.cpu_types import CPUTypes
from gem5.resources.resource import Resource
from gem5.resources.resource import CustomResource
from gem5.simulate.simulator import Simulator

cache_hierarchy = NoCache()

memory = SingleChannelDDR3_1600("1GiB")

#Choose the CPU Type
processor = SimpleProcessor(cpu_type=CPUTypes.'Name of the CPU model', num_cores=1)


board = SimpleBoard(
            #Choose the frequency at which we want to operate
            clk_freq="Frequency",
            processor=processor,
            memory=memory,
            cache_hierarchy=cache_hierarchy,
)

binary = CustomResource("./HPC1/mm")
board.set_se_binary_workload(binary)

simulator = Simulator(board=board)
simulator.run()
```

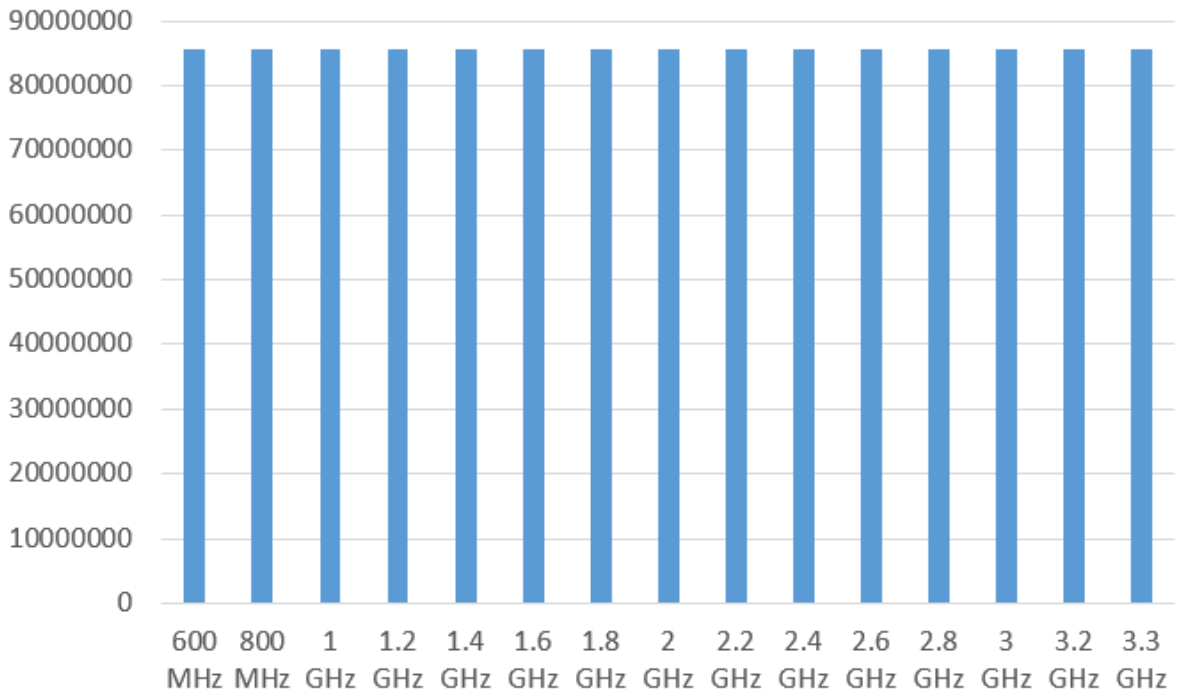
Python 2 | Tab Width: 8 | Ln 14, Col 21 | INS

Analysis-

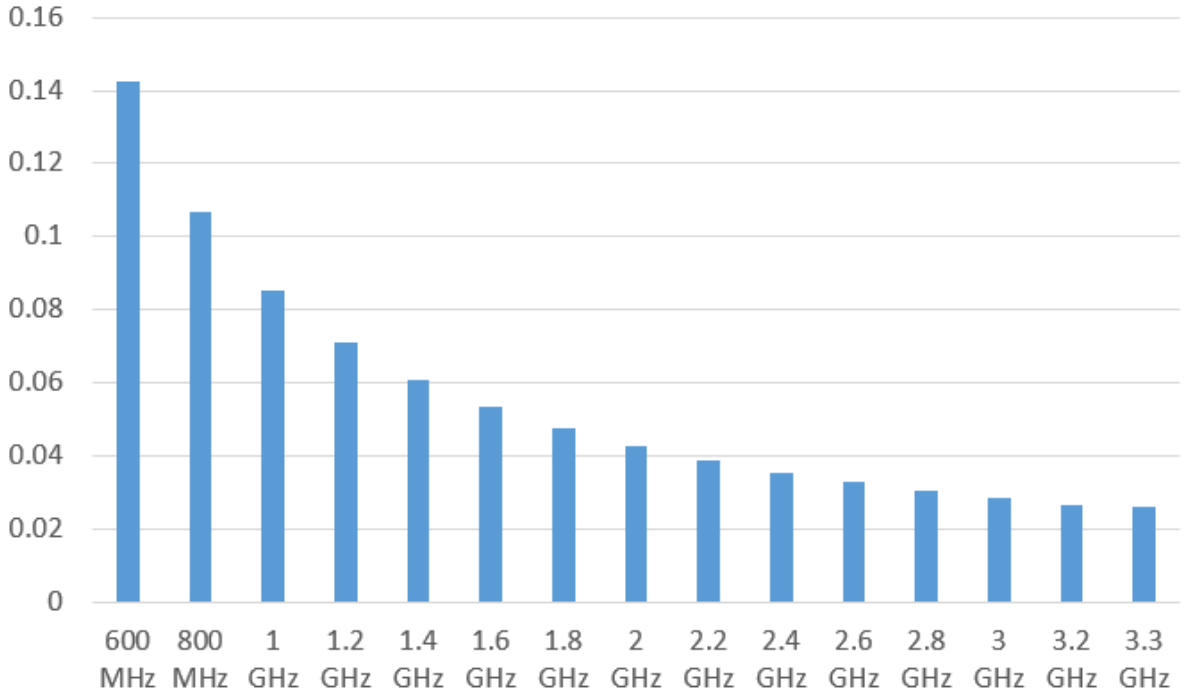
For AtomicSimpleCPU:

Frequency	NumCycles	Sim Time	Mem used	CPI
600 MHz	85509955	0.142545	1179380	1.60686
800 MHz	85509955	0.106887	1179384	1.60686
1 GHz	85509955	0.08551	1179384	1.60686
1.2 GHz	85509955	0.07123	1179384	1.60686
1.4 GHz	85509955	0.061054	1179384	1.60686
1.6 GHz	85509955	0.053444	1179384	1.60686
1.8 GHz	85509955	0.047544	1179380	1.60686
2 GHz	85509955	0.042755	1179380	1.60686
2.2 GHz	85509955	0.038907	1179384	1.60686
2.4 GHz	85509955	0.035658	1179380	1.60686
2.6 GHz	85509955	0.032921	1179384	1.60686
2.8 GHz	85509955	0.030527	1179384	1.60686
3 GHz	85509955	0.028475	1179384	1.60686
3.2 GHz	85509955	0.026765	1179384	1.60686
3.3 GHz	85509955	0.02591	1179380	1.60686

Number of Cycles required to execute the program vs Frequency



Simulation time vs Frequency



In AtomicSimpleCPU, the number of cycles required to execute the Matrix Multiplication program remains the same irrespective of the frequency.

In this CPU, instructions with memory requests are finished immediately.

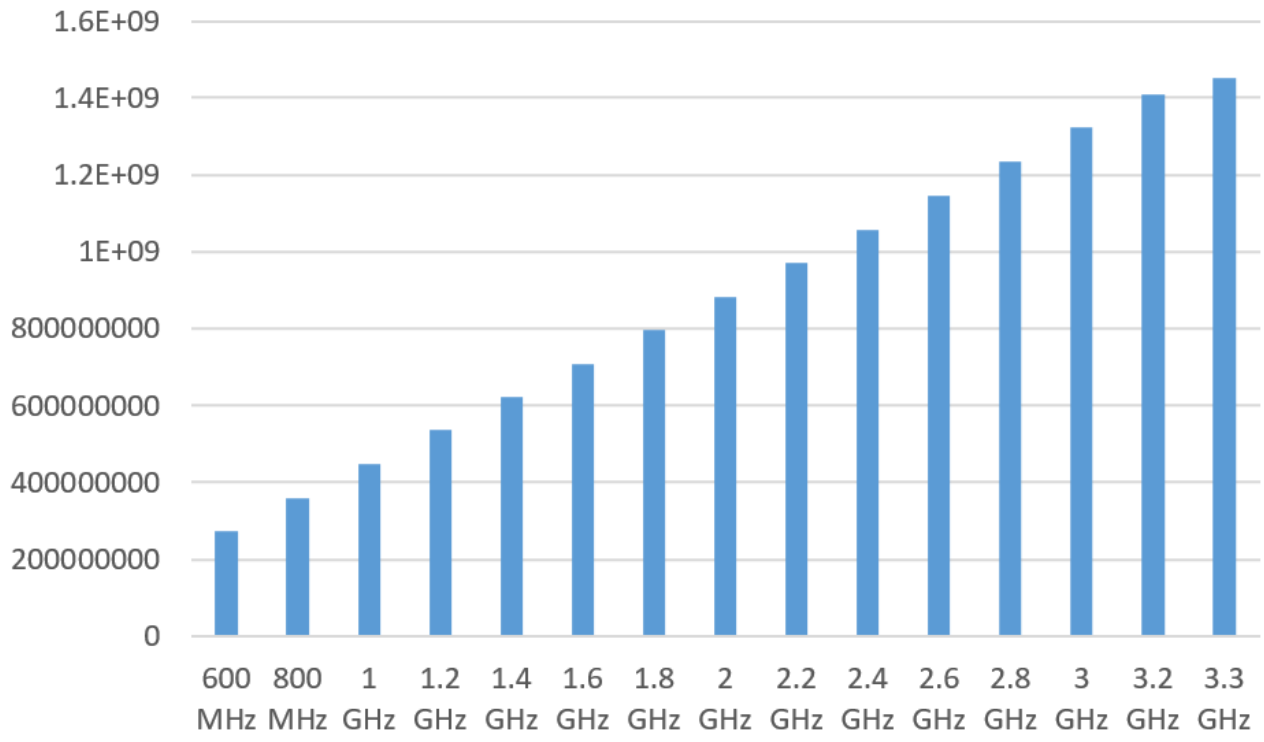
We can see that as the frequency of CPU is increased, the total simulation time decreases steeply. The total memory used in AtomicSimpleCPU remains same with all the frequencies.

Same is the case with Cycles per second (CPI), as they remain same with varying frequencies.

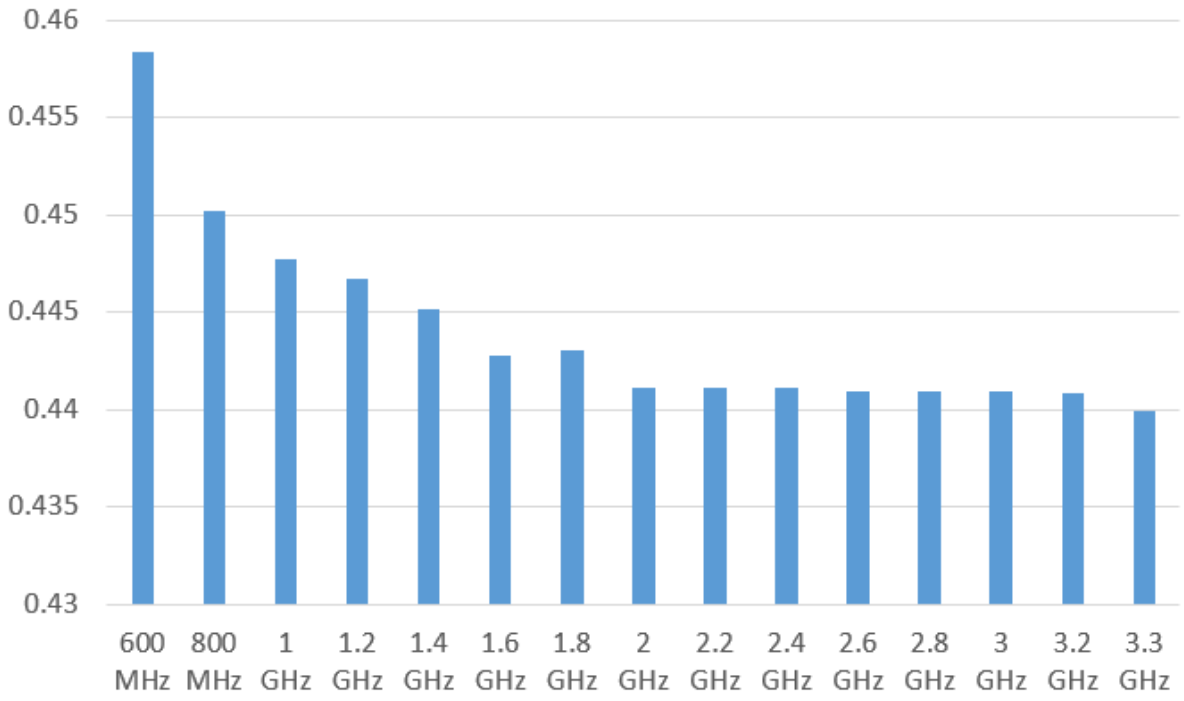
For O3CPU:

Frequency	NumCycles	Sim Time	Mem used	CPI
600 MHz	274951090	0.458343	1184508	5.166755
800 MHz	360162377	0.450203	1184508	6.768007
1 GHz	447700392	0.4477	1184508	8.412981
1.2 GHz	536291139	0.446731	1184508	10.077738
1.4 GHz	623518646	0.445192	1184512	11.716877
1.6 GHz	708468537	0.442793	1184512	13.313217
1.8 GHz	796799682	0.443021	1184508	14.973095
2 GHz	882283840	0.441142	1184508	16.579475
2.2 GHz	969546016	0.441143	1184512	18.219266
2.4 GHz	1057838685	0.441119	1184512	19.878421
2.6 GHz	1145330715	0.440952	1184512	21.522531
2.8 GHz	1235154338	0.44095	1184508	23.210456
3 GHz	1324155052	0.440944	1184508	24.882917
3.2 GHz	1408576489	0.440884	1184508	26.469326
3.3 GHz	1451863880	0.439915	1184508	27.282762

Number of Cycles required to execute the program vs Frequency



Simulation time vs Frequency



O3CPU is an out of order CPU model. It has five pipeline stages namely fetch, decode, rename, issue/execute/writeback and commit.

Interestingly, the number of cycles required to execute the matrix multiplication program increases as we increase the frequency.

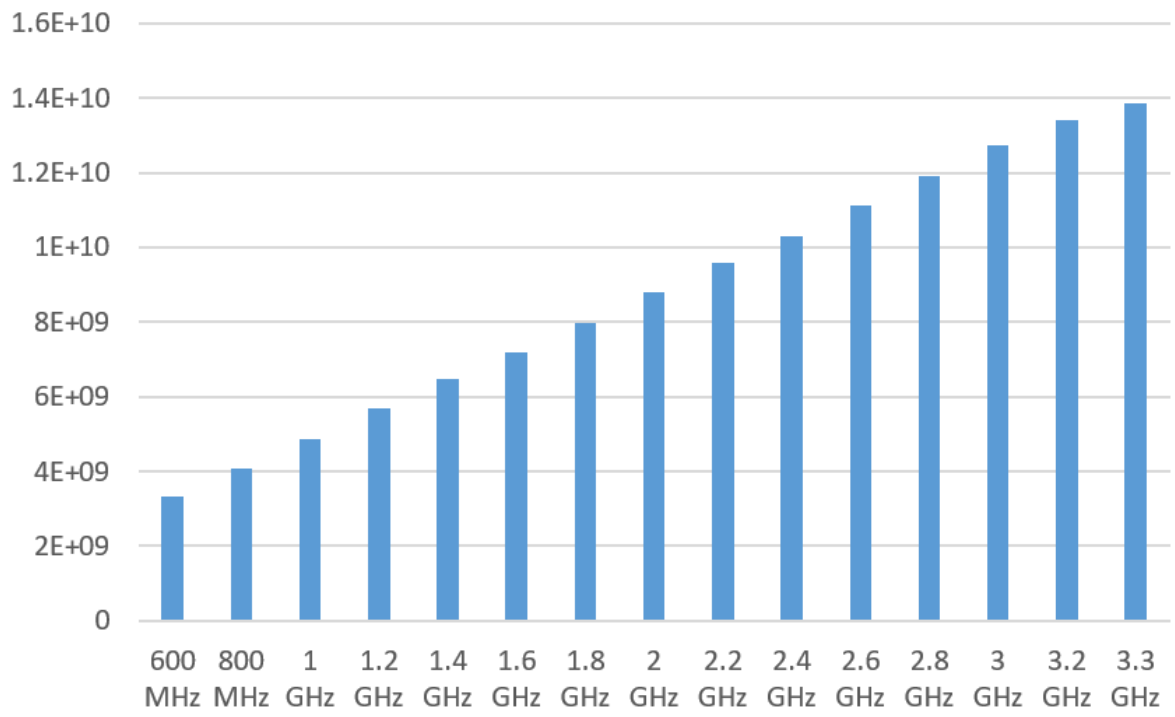
But, the total time required to simulate our program decreases non-linearly as we increase the frequency, with most difference between 600MHz and 800MHz frequencies.

As we expect, memory used is same with all the frequencies i.e., 1184508B.

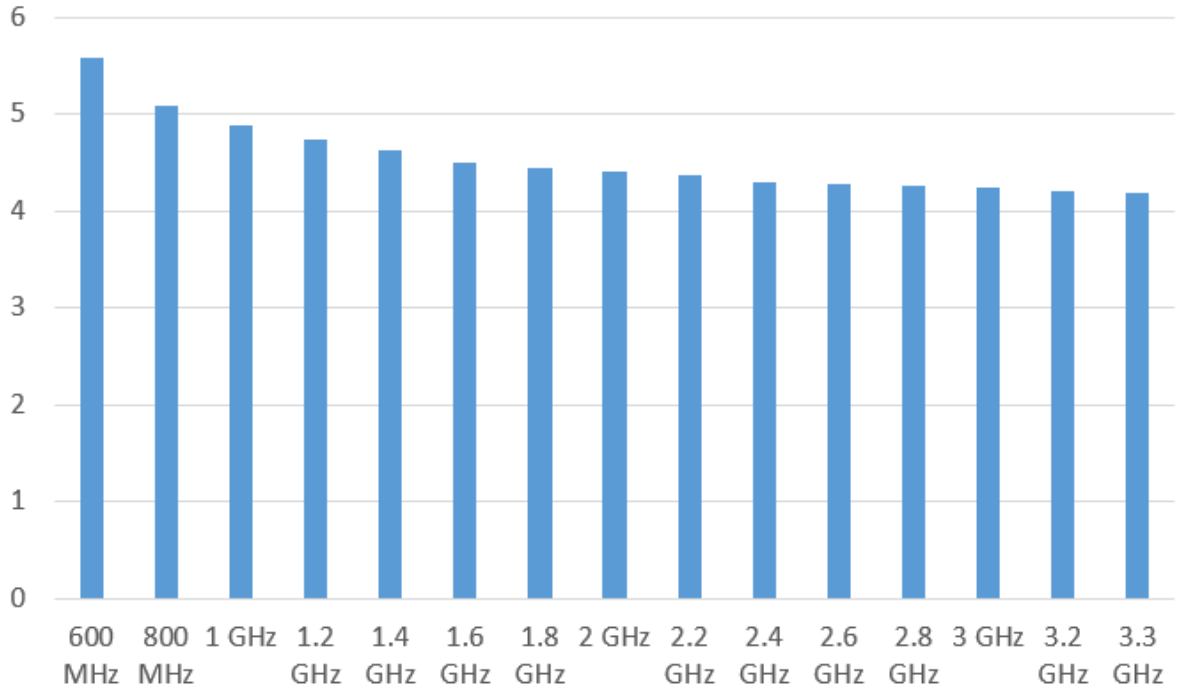
For TimingSimpleCPU:

Frequency	NumCycles	Sim Time	Mem used	CPI
600 MHz	3345260121	5.576549	1179380	62.8626
800 MHz	4073167204	5.091459	1179384	76.5411
1 GHz	4880532573	4.880533	1179380	91.7127
1.2 GHz	5690700890	4.740354	1179384	106.937
1.4 GHz	6468247086	4.618328	1179380	121.548
1.6 GHz	7188382274	4.492739	1179380	135.081
1.8 GHz	7989779211	4.442317	1179384	150.14
2 GHz	8802117338	4.401059	1179384	165.405
2.2 GHz	9594582935	4.365535	1179380	180.297
2.4 GHz	10312659940	4.300379	1179384	193.791
2.6 GHz	11107922167	4.27655	1179384	208.735
2.8 GHz	11914630552	4.253523	1179384	223.894
3 GHz	12719767426	4.235683	1179380	239.024
3.2 GHz	13421027711	4.200782	1179384	252.202
3.3 GHz	13841946743	4.19411	1179380	260.111

Number of Cycles required to execute the program vs Frequency



Simulation time vs Frequency



In TimingSimpleCPU, the number of cycles increase linearly with increase in frequency.

It stalls on cache accesses and wait until it gets response. It implements same set of functions as BaseSimpleCPU.

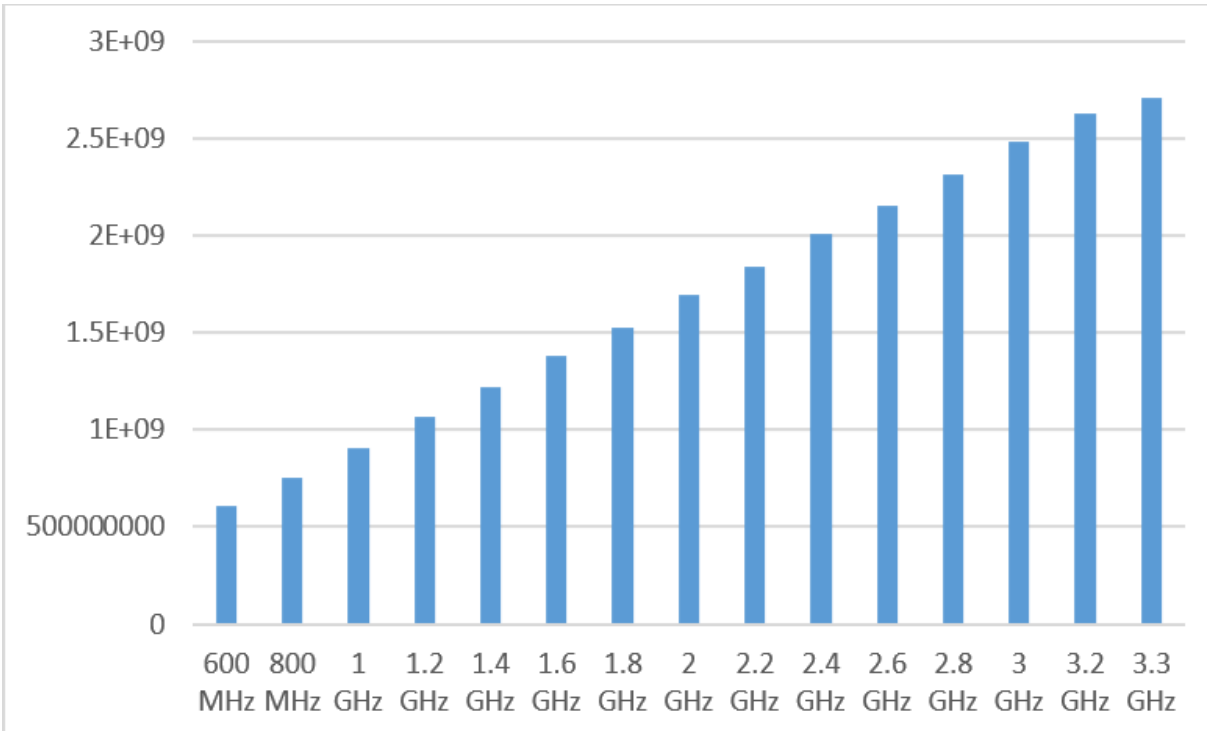
The total simulation time to run our matrix multiplication program decreases very slowly as we increase CPU frequency from 600MHz to 3.3GHz.

As we have expected, the memory used is sameand CPI is increasing with varying values of frequencies.

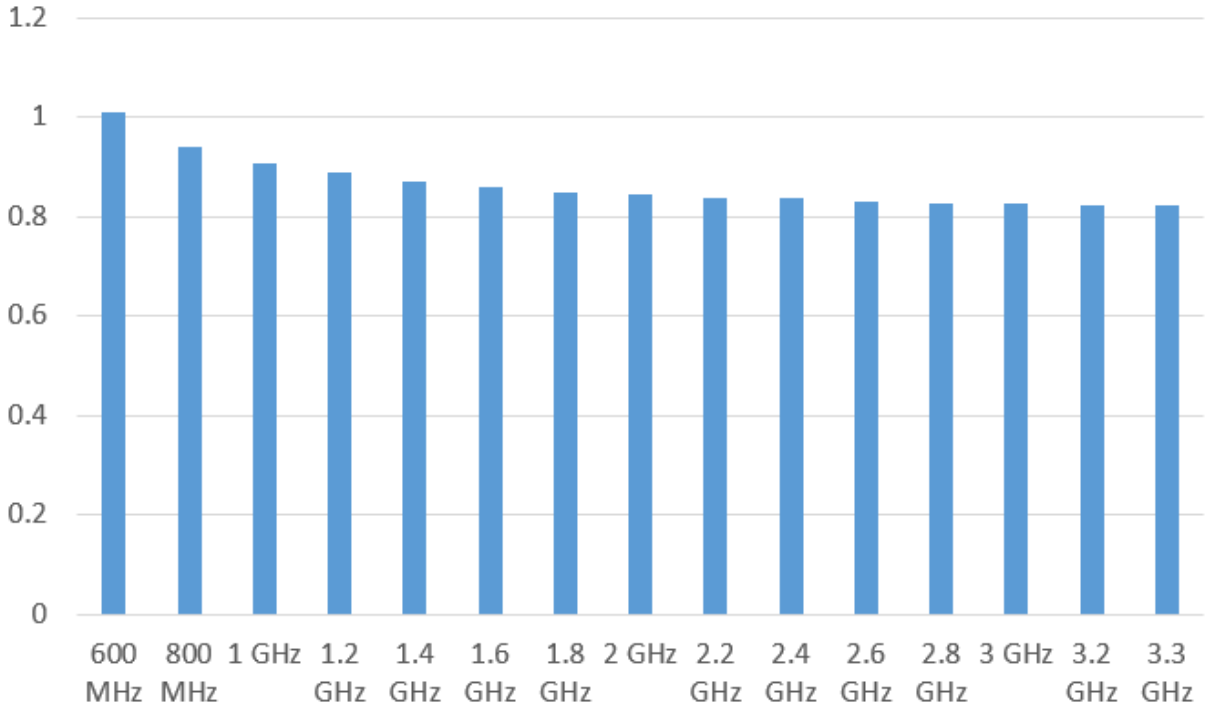
For MinorCPU:

Frequency	NumCycles	Sim Time	Mem used	CPI
600 MHz	606245494	1.010611	1181432	11.3923
800 MHz	751728878	0.939661	1181432	14.1261
1 GHz	907316834	0.907317	1181432	17.0499
1.2 GHz	1067900044	0.889561	1182452	20.0675
1.4 GHz	1220884046	0.871711	1181428	22.9423
1.6 GHz	1377579795	0.860987	1182456	25.8868
1.8 GHz	1524456008	0.847598	1181432	28.6469
2 GHz	1692418950	0.846209	1181428	31.8032
2.2 GHz	1840206052	0.837294	1181432	34.5803
2.4 GHz	2005239466	0.836185	1181428	37.6815
2.6 GHz	2154440358	0.82946	1181428	40.4852
2.8 GHz	2316388177	0.826951	1181428	43.5285
3 GHz	2485299462	0.827605	1181432	46.7026
3.2 GHz	2625139943	0.821669	1181432	49.3304
3.3 GHz	2711728730	0.821654	1181432	50.9575

Number of Cycles required to execute the program vs Frequency



Simulation time vs Frequency



MinorCPU is a inorder CPU and has a fixed pipeline.

Number of cycles required to execute our matrix multiplication program with 100X100 inputs increase linearly as we increase the frequency of our CPU.

The simulation time of our program decrease very slowly on increase of CPU frequency.

The memory used remains the same and CPI increases steeply as we change the CPU from 600MHz to 3.3GHz.