Homework #1

Submitted by- Deepak Kumar(dkumar15@hawk.iit.edu)

Problem-1:

(a) Using trace files, i.e. files that contain addresses issued by some CPU to execute some application(s), draw the histogram of address distribution for each of them (2x20 points). On the *Ox* axis of the plot you will have the address as a decimal number. On the *Oy* axis you will have the number of occurrences for each particular address.

NOTE: the range of addresses is vast, attempting to plot everything will result in a histogram with very little detail. Instead, select a range of addresses (a few hundreds of them) where you have non-zero values on *Oy*. Comment based on the histograms (5)

(b) What is the frequency of writes (5)? What is the frequency of reads (5)? Please comment on these results (5).

Solution - I used Python 3.8 to analyze and plot the histogram. We have a large dataset of addresses, so in the code, I implemented the address range parameter to select specific address ranges for analysis.

Below, I've listed some of the address ranges along with their corresponding histogram, frequencies of read and write operations, and I've also included some comments on the output results.

Github Link To Code - https://github.com/deepdik/CS-402

Note **-** All the installation and running steps are documented under the **Readme.md** file.

1. **Address Range:** 4401100 - 4402500

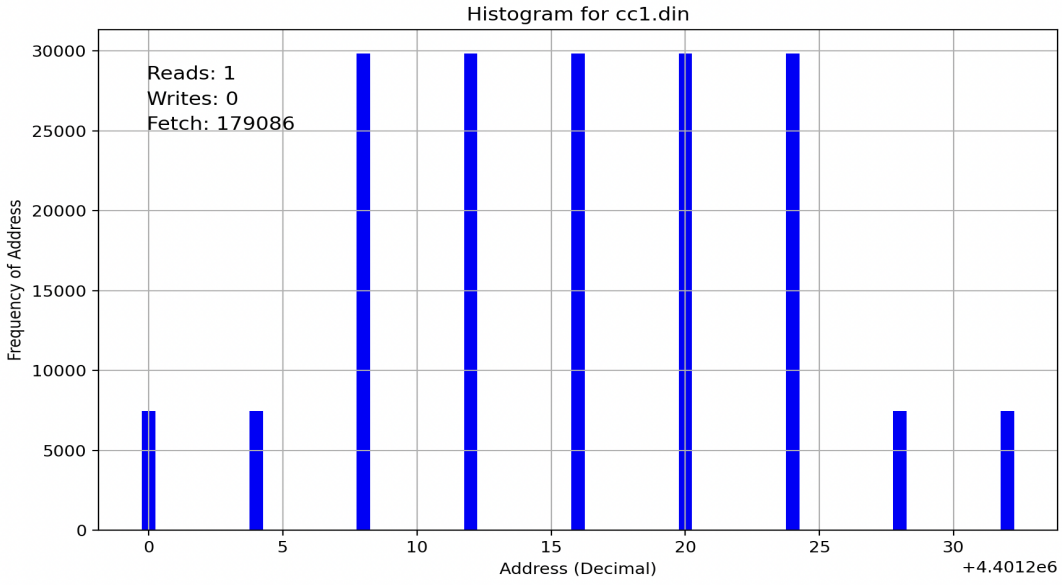
File : cc1.din

Read Freq : 1

Write Freq : 0

Fetch Freq: 179086

Output Image:



Comment on Histogram - The graph has very few bars, indicating that there are very few addresses used in the selected range. However, the frequency count is quite high (i.e., 0 to 30000), suggesting a heavy load on these addresses.

Comment on Frequencies - The presence of a large number of fetch operations (179,086) within this address range suggests that the CPU was frequently fetching instructions from this memory range. Reading and writing is very less.

2. **Address Range:** 269800800 to 269801200

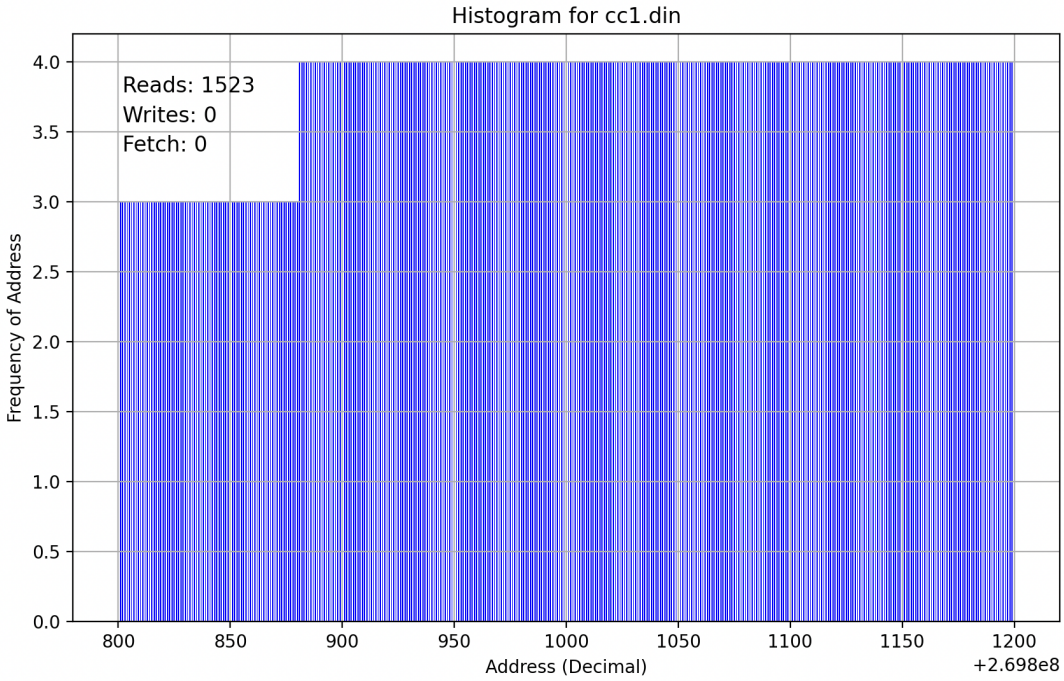
File : cc1.din

Read Freq : 1523

Write Freq : 0

Fetch Freq: 0

Output Image:



Comment on Histogram - In the selected range of addresses, the frequency count is low (i.e., 0 to 4), but the graph is quite dense, indicating that this range of memory addresses is used for fewer operations, yet most of the addresses within this range are actively utilized.

Comment on Frequencies - The high number of read operations and the absence of write or fetch operations in the specified address range suggest that this region is predominantly used for read-only data access.

3. **Address Range:** 268519424 to 268520000

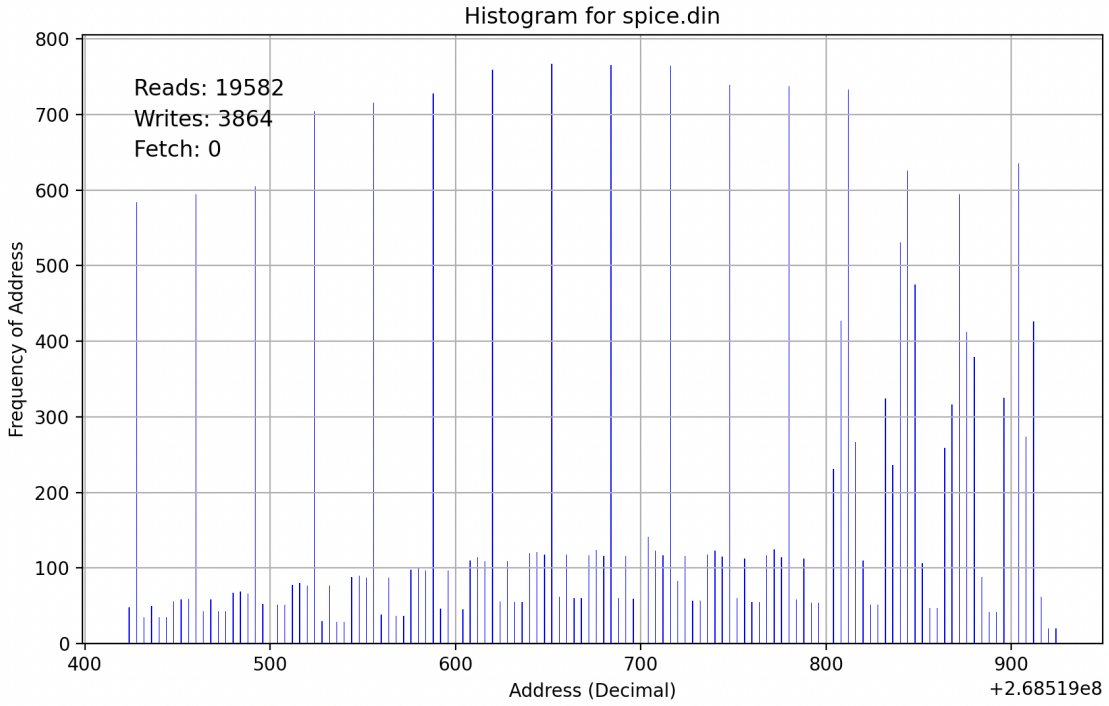
File : spice.din

Read Freq : 19582

Write Freq : 3864

Fetch Freq: 0

Output Image:



Comment on Histogram - In the selected range of addresses, the frequency count is decent (i.e. 0 to 800), but the graph is not dense, indicating that this range of memory addresses is used for fewer operations, yet most of the addresses within this range are not actively utilized.

Comment on Frequencies - With a frequency of 19,582 read operations in this address range, it reinforces the idea that this memory region is heavily used for reading data. The presence of 3,864 write operations indicates that while this memory region is primarily read-heavy, there are still occasional write operations. The absence of fetch operations in this address range means that the CPU is not fetching instructions from this specific memory range. This suggests that the CPU is not executing code directly from this region but rather using it primarily for data access.

Problem-2:

(a) Write a program, using C, C++, or Java, that multiplies two rectangular matrices -- please no square matrices -- whose elements are randomly generated. You may not use a matrix multiplication library in your code. You will have two versions of the program, one in which matrix elements are integers and another one where they are real numbers (double) (2x15 points).

You will compile and run the programs on two different systems -- most likely one of them will be your own desktop/laptop and the other one a computer in the lab, or otherwise on one of the UNIX computers IIT makes available to its students.

Measure the time it takes each program to complete (2x5) and then compare the performance of the two systems (5). Since the matrices are randomly generated, you will have to run the program several times, measure the running time and then take the average. Also the running time has to be significantly large (at least several seconds) to reduce the impact of measuring errors; this means you will have to work with matrices that have at least hundreds of lines and columns.

Is the performance ratio the same as the clock rate ratio of the two systems (5)? Explain. Based on the retail price of the two systems, which one is more cost effective (5)?

Make sure your work includes a description of the two systems (manufacturer, CPU type, amount of memory, operating system, etc.) and of the compiler used (5). Attach the source code, the tables with your time measurements for your work, and a link to your repository such that we can check-out the code, build, and execute (5).

Solution - I used Java 18 to create a matrix multiplication program with a matrix size greater than 1000 to ensure that the time taken for completion is in the order of second. I ran the program five times and calculated the average to reduce the impact of measurement errors.

Github Link To Code - https://github.com/deepdik/CS-402 **Note -** All the installation and running steps are documented under the **Readme.md** file.

There are two system used for this analysis and the details are given below-

System-1 Configuration:

Manufacturer : Apple (Macbook Pro M1)
CPU : 8 (6 performance and 2 efficiency)
Memory : 32 GB
OS : MacOS Monterey- Version 12.2.1
Compiler Info: JDK Version - 18

Clock Speed : 2.4 GHz
Price : **$**1900

System-2 Configuration:
Manufacturer : Asus
CPU : i7- gen 9
Memory : 16 GB
OS : Windows -11
Compiler Info: JDK Version - 20
Clock Speed : 2.0 GHz
Price : $ 1400

1. First Matrix Multiplication- (Used java file - MatrixMultiplication.java)

Image: System-1

```
(venv) deepak@Deepaks-MacBook-Pro Question-2 % java MatrixMultiplication
Integer Matrix Multiplication Time Is : 7.161629792 seconds
Double Matrix Multiplication Time Is: 9.472909375 seconds
(venv) deepak@Deepaks-MacBook-Pro Question-2 % java MatrixMultiplication
Integer Matrix Multiplication Time Is : 7.350672542 seconds
Double Matrix Multiplication Time Is: 9.679387333 seconds
(venv) deepak@Deepaks-MacBook-Pro Question-2 % java MatrixMultiplication
Integer Matrix Multiplication Time Is : 7.31468175 seconds
Double Matrix Multiplication Time Is: 9.965584708 seconds
(venv) deepak@Deepaks-MacBook-Pro Question-2 % java MatrixMultiplication
Integer Matrix Multiplication Time Is : 7.319539875 seconds
Double Matrix Multiplication Time Is: 9.808243709 seconds
(venv) deepak@Deepaks-MacBook-Pro Question-2 % java MatrixMultiplication
Integer Matrix Multiplication Time Is : 7.385036291 seconds
Double Matrix Multiplication Time Is: 9.941706625 seconds
```

Image: System-2

```
PS C:\Users\Siddhant\Downloads\CS-402-master\CS-402-master\HW-1\Question-1> cd ..
PS C:\Users\Siddhant\Downloads\CS-402-master\CS-402-master\HW-1> cd .\Question-2\
PS C:\Users\Siddhant\Downloads\CS-402-master\CS-402-master\HW-1\Question-2> java MatrixMultiplication
Integer Matrix Multiplication Time Is : 10.1403682 seconds
Double Matrix Multiplication Time Is: 19.9147337 seconds
PS C:\Users\Siddhant\Downloads\CS-402-master\CS-402-master\HW-1\Question-2> java MatrixMultiplication
Integer Matrix Multiplication Time Is : 10.3633751 seconds
Double Matrix Multiplication Time Is: 17.9283108 seconds
PS C:\Users\Siddhant\Downloads\CS-402-master\CS-402-master\HW-1\Question-2> java MatrixMultiplication
Integer Matrix Multiplication Time Is : 10.8222792 seconds
Double Matrix Multiplication Time Is: 17.8472023 seconds
PS C:\Users\Siddhant\Downloads\CS-402-master\CS-402-master\HW-1\Question-2> java MatrixMultiplication
Integer Matrix Multiplication Time Is : 9.8323498 seconds
Double Matrix Multiplication Time Is: 17.9183404 seconds
PS C:\Users\Siddhant\Downloads\CS-402-master\CS-402-master\HW-1\Question-2> java MatrixMultiplication
Integer Matrix Multiplication Time Is : 10.1899566 seconds
Double Matrix Multiplication Time Is: 19.9386785 seconds
```

System -1		System-2	
Integer Multiplication(sec)	**Double Multiplication(sec)**	**Integer Multiplication(sec)**	**Double Multiplication(sec)**
7.161629792	9.472909375	10.1403682	19.9147337
7.350672542	9.679387333	10.3633751	17.9283108
7.31468175	9.965584708	10.8222792	17.8472023
7.319539875	9.808243709	9.8323498	17.9183404
7.385036291	9.941706625	10.1899566	19.9386785
Avg - 7.306312050	**9.77356635**	**10.26966578**	**18.70945314**

Performance Compare -

Performance ratio -

1. **Integer Matrix Multiplication** -

   T System-2/TSystem-1 = 10.26966578/7.306312050 => 1.4

   1.405049235 = 1 + N/100

   N = 40.5049235

   So, the ratio 1.405049235 is equivalent to a 40.5049235% increase, which means that System-1 is approximately **40.50%** faster than System-2 for Integer Matrix Multiplication

2. **Double Matrix Multiplication** -

   T System-2/TSystem-1 = 18.70945314/9.77356635 => 1.91429131087

   Ratio = 1 + N/100

   N ≈ 91.65738

   which means System-1 is faster **91.66%** than System-2 for Double Matrix Multiplication.

Clock speed vs performance -

Clock Speed Ratio = 2.4/2.0 = **1.2**

This means that the system-1 clock speed is **20%** times faster than the second system's clock speed. There is no direct relation in performance and clock speed in terms of numbers. Although, there could be a chance that more clock speed means better performance than lower clock speed but it also depends on several other factors.

Cost effectiveness -

Based solely on the retail prices of the two systems, System-2 is more cost-effective. System-2 has a lower retail price of $1400 compared to System-2 retail price of $2000. Cost-effectiveness in this context is determined by the initial cost of purchasing the system, and T2 is the more budget-friendly option in terms of upfront cost

2. Second Matrix Multiplication- (Used java file - *MatrixMultiplicationSecond.java*)

Question - (b) Change your multiplication algorithm and repeat the steps above; for instance, if you used the naive multiplication algorith with the column in the inner loop, then just use the same algorithm with the row in the inner loop (same scoring as part a).

Solution:- Created Matrix multiplication code with row in inner loop. See file- *MatrixMultiplicationSecond.java (*https://github.com/deepdik/CS-402*)*

Image: System-1

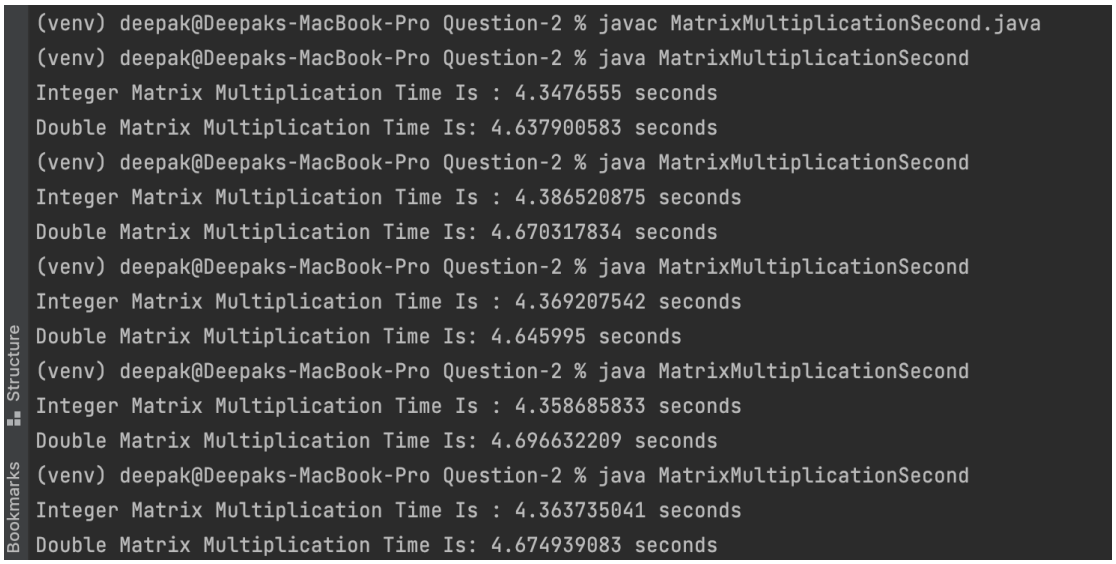
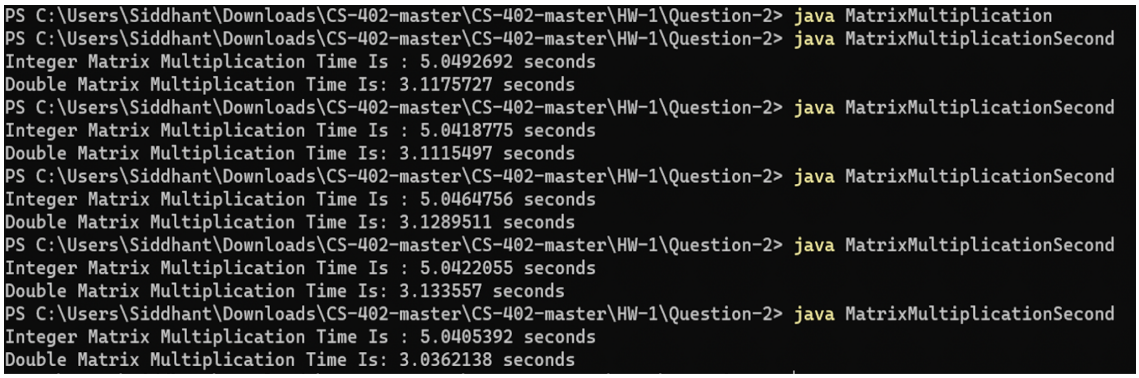


Image: System-2



System -1		System-2	
Integer Multiplication(sec)	**Double Multiplication(sec)**	**Integer Multiplication(sec)**	**Double Multiplication(sec)**
4.3476555	4.637900583	5.0492692	3.1175727
4.386520875	4.670317834	5.0418775	3.1115497
4.369207542	4.645995	5.0464756	3.1289511
4.358685833	4.696632209	5.0422.55	3.133557
4.363735041	4.674939083	5.0405392	3.0362138
Avg - 4.3651609582	**4.6651569412**	**5.0440833**	**3.10576886**

Performance Compare -

Performance ratio -

1. **Integer Matrix Multiplication** -

   T System-2/TSystem-1 =

   5.0440833 / 4.3651609582 ≈ 1.1543

   To express this ratio in terms of 1 + N/100, where N represents the percentage difference:

   1.1543 - 1 = 0.1543

   N = 15.43

   So, System-1 is approximately **15.43%** faster than System-2 for Integer Matrix Multiplication

2. **Double Matrix Multiplication** -

   T System-2/TSystem-1 =3.10576886/4.6651569412 => 1.91429131087

   1 + N/100 = 3.10576886 / 4.6651569412

   N = -33.48

So, the ratio 3.10576886 to 4.6651569412 is approximately equivalent to a -33.48% difference, which means System-1 is 33.48% slower than System-2 for double matrix multiplication.

Clock speed vs performance -

Same as previous analysis

Cost effectiveness -

Same as previous analysis