ES215: Semester II [2021-2022]
Assignment 4

1. Suppose we have a 5-stage pipeline with each stage taking one cycle. Find the speed up if we have: (***Total: 25 points***)

a. 30% RAW dependency and 20% branch dependency.
b. 40% branch dependency.
   Also find the speed up if we used a branch predictor with 80% accuracy.

Ans. We know in the case of pipelining, CPI = 5 and in case of non-pipelining, CPI = 1. Speedup for the cases mentioned are shown below: -
We assume Instruction count to be n.

a. We have 30% RAW dependency and 20% branch dependency. Due to of RAW dependency no. of cycles will be, $1 * 0.3 * 3(stalls) = 0.9n$ and no. of cycles due to branch dependency will be, $1 * 0.2 * 2(stalls) = 0.4n$.

   We know $Execution\ time = \frac{Instruction\ count * CPI}{Clock\ Rate}$

   Execution time before pipelining, $(Execution\ time)_{no\ pipelining} = \frac{n*5}{CR}$

   and, Execution time after pipelining, $(Execution\ time)_{pipelining} = \frac{n+0.9n+0.4n}{CR}$

   Therefore, speedup of before pipelining and after pipelining will be,

$$Speedup = \frac{\frac{n*5}{CR}}{\frac{n+0.9n+0.4n}{CR}}$$

$$Speedup = \frac{n * 5}{n + 0.9n + 0.4n} = \frac{5}{2.3} = 2.174$$

b. We have 40% branch dependency. No. of instructions due to this will be,

$$1 * 0.4 * 2(stalls) = 0.8n$$

   We know $Execution\ time = \frac{Instruction\ count * CPI}{Clock\ Rate}$

   Execution time before pipelining, $(Execution\ time)_{no\ pipelining} = \frac{n*5}{CR}$

   and, Execution time after pipelining, $(Execution\ time)_{pipelining} = \frac{n+0.8n}{CR}$

   Therefore, speedup of before pipelining and after pipelining will be,

$$Speedup = \frac{\frac{n*5}{CR}}{\frac{n+0.8n}{CR}}$$

$$Speedup = \frac{n * 5}{n + 0.8n} = \frac{5}{1.8} = 2.78$$

Also, we used a branch predictor with 80% accuracy then inaccuracy will be 20%.

So, no. of cycles due to this will be, $1 * 0.4 * 0.2 * 2 = 0.16n$

Speedup would be,

$$Speedup = \frac{\frac{n*5}{CR}}{\frac{n+0.16n}{CR}}$$

$$Speedup = \frac{n * 5}{n + 0.16n} = \frac{5}{1.16} = 4.31$$

2. Assume that 20% of the instructions executed on a computer are branch instructions. We use delayed branching with one delay slot. Estimate the CPI, if the compiler is able to fill 85% of the delay slots. Assume that the base CPI is 1.5. In the base case, we do not use any delay slot. Instead, we stall the pipeline for the total number of delay slots. (***Total: 25 points***)

Ans. Let Instruction count be n (assumption)

Branch instructions will be, 0.2*n = 0.2n.

As it is mentioned in the question, we use delayed branching with one delay slot, so for 0.2n branch instructions there will be 0.2n delayed branching. The compiler is able to fill 85% of the delay slots so, there were $0.2n * \frac{85}{100} = 0.17n$ filled delay slots. CPI of filled delay slots, $\frac{0.17n}{n} = 0.17$. As, in base case, CPI = 1.5 so, final CPI would be $1.5 - 0.17 = 1.33$.

3. You have implemented a simple Matrix Multiplication program for a given N x N matrix in Assignment#1 using three loops. You might have noticed that the result is independent of how you position the three loops. Hence, we can interchange the positions of loops that can produce different programs to execute matrix multiplication. But what about performance? Does it remain the same or change? What is the impact? And what option(s) provide the best result and why?

Implement Matrix Multiplication program for a given N x N matrix (integer data type) in any of your preferred Languages from the Bucket 1, where N is iterated through the set of values 128, 256, and 512. N can either be hardcoded or specified as input. (***Total: 100 points***)

Bucket1: C, C++, Go (compulsory)

Bucket2: Python, Java. (optional)

For each program out of the total possible 6 combinations:

a. Compute the execution time for the meat portion of the program for all the possible combinations of loop interchange.
b. Plot the execution times for each of the iterations. And compare the performance (Program execution times) of the program for a given value of N. –Illustrate your observations.

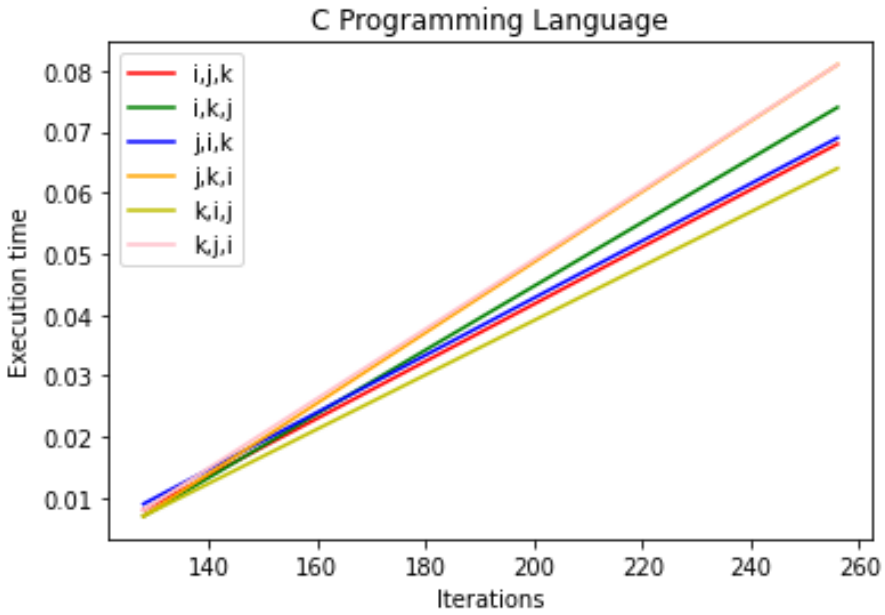
Ans.

a. First, we have made the code in C language from Bucket 1 and obtained the execution time for N = 128 and N = 256, but when I tried to run for N = 512 the code was not running in my computer, therefore, I have not included it's results in the following table. The results are shown below. Source files are uploaded in repository.

For N = 128	
Case 1 (i, j, k)	0.008 s
Case 2 (i, k, j)	0.007 s
Case 3 (j, i, k)	0.009 s
Case 4 (j, k, i)	0.007 s
Case 5 (k, i, j)	0.007 s
Case 6 (k, j, i)	0.008 s
For N = 256	
Case 1 (i, j, k)	0.068 s
Case 2 (i, k, j)	0.074 s
Case 3 (j, i, k)	0.069 s
Case 4 (j, k, i)	0.081 s
Case 5 (k, i, j)	0.064 s
Case 6 (k, j, i)	0.081 s

The execution time changes for each case.

b. The graph shown below reflects the comparison between execution time and iterations. The lines represent each case. First, we have plotted the value for N = 128, then for N = 256 and then joined the points to make a line. Similarly in all the cases.



4. Repeat the above question, using any one of the languages from Bucket 2. Infer and reason if any observations differ?

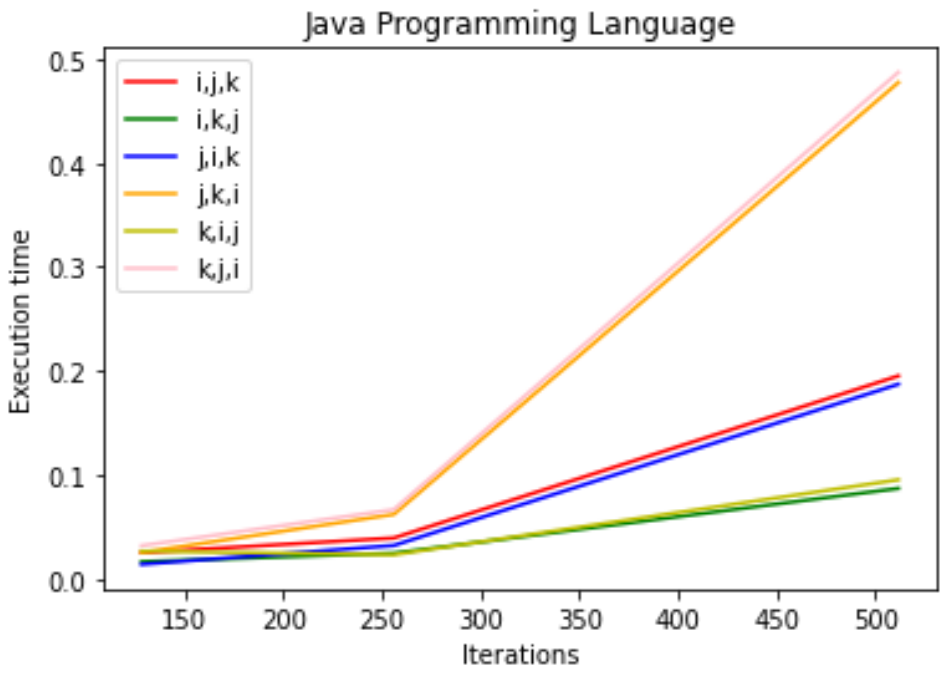
Ans.

c. We have made the code in Java language from Bucket 2 and obtained the execution time for N = 128, N = 256 and N = 512. The results are shown below. Source files are uploaded in repository.

For N = 128	
Case 1 (i, j, k)	0.0253 s
Case 2 (i, k, j)	0.0166 s
Case 3 (j, i, k)	0.0141 s
Case 4 (j, k, i)	0.0254 s
Case 5 (k, i, j)	0.0268 s
Case 6 (k, j, i)	0.0320 s
For N = 256	
Case 1 (i, j, k)	0.0394 s
Case 2 (i, k, j)	0.0248 s
Case 3 (j, i, k)	0.0322 s
Case 4 (j, k, i)	0.0621 s
Case 5 (k, i, j)	0.0233 s
Case 6 (k, j, i)	0.0667 s

For N = 512	
Case 1 (i, j, k)	0.1953 s
Case 2 (i, k, j)	0.0871 s
Case 3 (j, i, k)	0.1873 s
Case 4 (j, k, i)	0.4775 s
Case 5 (k, i, j)	0.0954 s
Case 6 (k, j, i)	0.4873 s

d. The graph shown below reflects the comparison between execution time and iterations. The lines represent each case. First, we have plotted the value for N = 128, N = 256, then for N = 512 and joined the points to make a line. Similarly in all the cases.



Submitted by: -
Mohak Goyal
21120012
Non-Degree Program UG