Final Project report

Introduction:

Performing matrix multiplication using

Cache memory

Registers

Normal matrix multiplication

Compare the average time of these methods and conclude the optimised matrix.

Background:

The areas we should know about this project are

1. Cache memory locality:

   Temporal locality: Referred to the data items that are used frequently

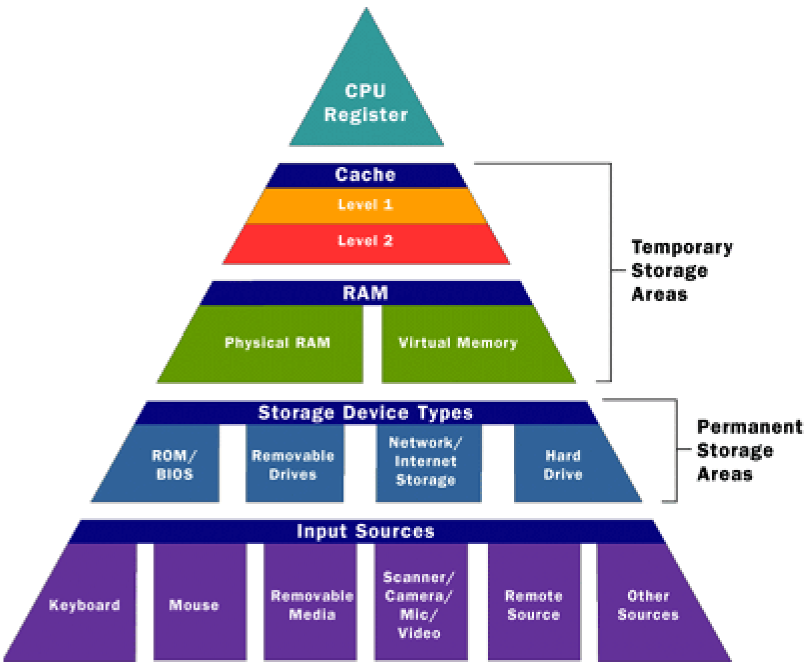
   Spatial locality : Referred to address that is located near by in cache memory

2. Tags in cache memory:
   Tags are used to specify the data
   Tags along with the index value allow use to find the exact data
3. Cache Blocking:
   Since the entire matrix may not fix into the cache memory, we break the entire matrix into blocks and do the calculation
4. Hit rate/ miss rate:
   Hit rate is used to determine if the data is present in the cache memory
   Miss rate is used to determine if the data is not present in the cache memory
5. Registers
   Since the register are in cpu use this method to calculate the matrix multiplication and expect it to be fastest among all.

Approach:

Cache memory:

Since the cache memory is small in l1 cache we diving the entire matrix into chunks of matrix(blocks), the block size is the size that we want to divide the matrix into, and that block of matrix is send to the l1 cache memory and we could performs the calculations there.



How to calculate cache miss and cache hit ratio:

Miss ratio = 1- hit ratio. (or)

Hit ratio = 1 – Miss ratio. (or)

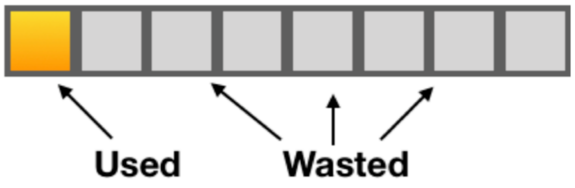
Miss – Hit = 1

Register blocking:

Problem with register blocking:

- The cpu is not fully utilized because of the memory bound.
- When the processor loads a cache line from the main memory into L1 cache, It loads the scalar value from cache line one at a time.
- While scanning the matrix and use a single scalar from every cache line we load, by the time we finish scanning the last value there may

be a chance that the memory has already flushed from the cache memory.

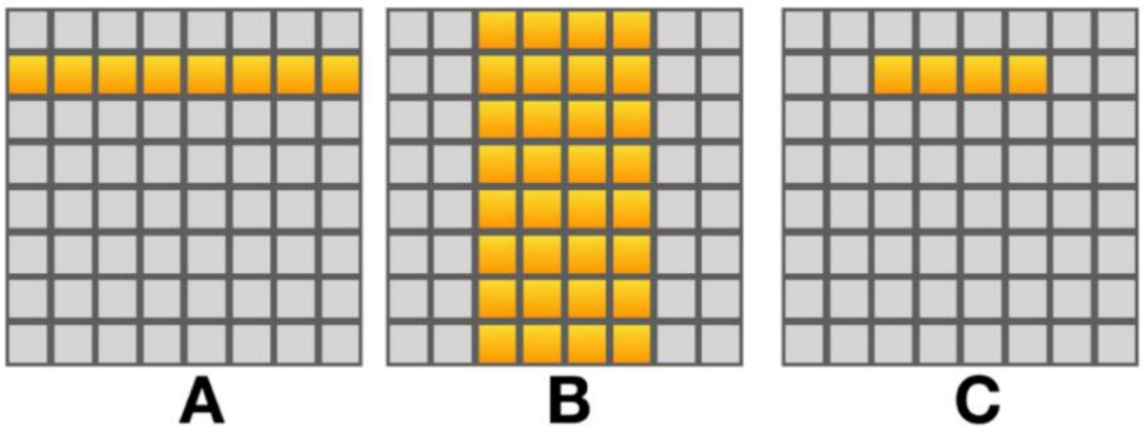


Solution

Since the problem is not utilising the memory to the fullest we resolve this like:

Placing the dummy values in the size of the memory we took in register, So now we are sure that the register has the matrix of given size has data.

When we enter into the algorithm we replace the dummy values with the actually values and perform the calculation so the advantage is, we don't waste the memory and also we perform the calculation simultaneously.



Results:

Trails	N	Matrix multiplication	Cache Memory	Registers	Cache Miss	Cache Miss
1	300	0.15	0.16	0.11	4424	4423
2	500	0.77	0.74	0.58	24842	24841
3	700	2.04	2.02	1.54	74107	74106
4	1000	5.45	5.68	4.33	258021	258020
5	1500	19.24	19.32	14.33	1619964	1619963
6	2000	59.97	52.57	45.98	84331367	84331366
7	2500	158.09	139.90	67.91	1874242480	1874242479
8	3000	209.91	203.98	123.33	3380111043	3380111042
9	3500	522.00	404.36	284.19	5380649268	5380649267
10	4000	681.65	543.14	451.85	8048920483	8048920482
Average		165.926	137.187	99.415		