

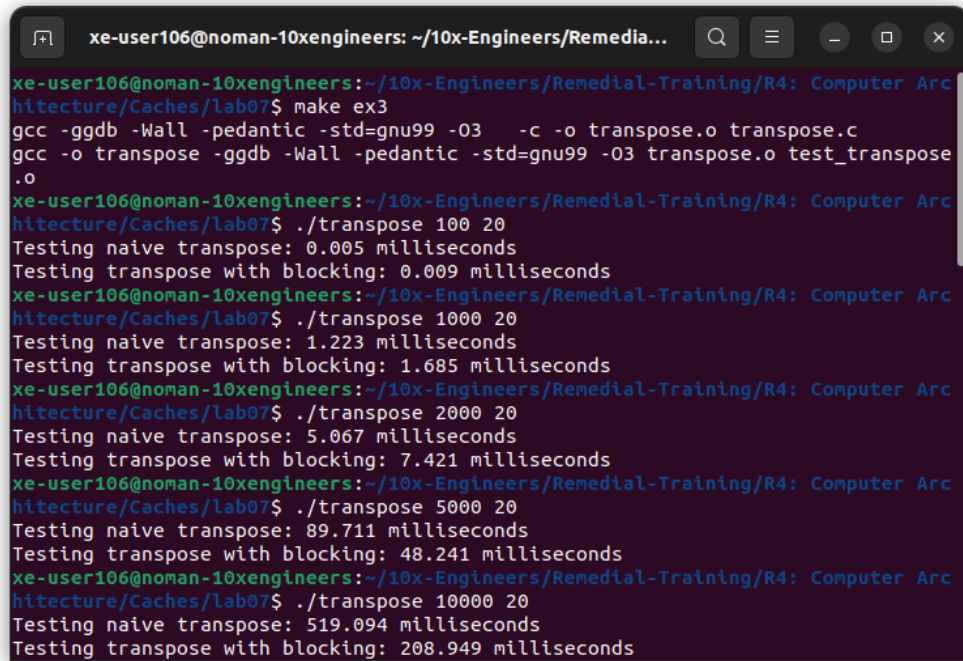
Module: R4: Computer Architecture

Section: Caches Task: Cache Blocking

Task 4

Cache Blocking

- Part I: Changing Array Size
 - Terminal Output:



```

xe-user106@noman-10xengineers: ~/10x-Engineers/Remedia...
xe-user106@noman-10xengineers:~/10x-Engineers/Remedial-Training/R4: Computer Arc
hitecture/Caches/lab07$ make ex3
gcc -ggdb -Wall -pedantic -std=gnu99 -O3 -c -o transpose.o transpose.c
gcc -o transpose -ggdb -Wall -pedantic -std=gnu99 -O3 transpose.o test_transpose
.o
xe-user106@noman-10xengineers:~/10x-Engineers/Remedial-Training/R4: Computer Arc
hitecture/Caches/lab07$ ./transpose 100 20
Testing naive transpose: 0.005 milliseconds
Testing transpose with blocking: 0.009 milliseconds
xe-user106@noman-10xengineers:~/10x-Engineers/Remedial-Training/R4: Computer Arc
hitecture/Caches/lab07$ ./transpose 1000 20
Testing naive transpose: 1.223 milliseconds
Testing transpose with blocking: 1.685 milliseconds
xe-user106@noman-10xengineers:~/10x-Engineers/Remedial-Training/R4: Computer Arc
hitecture/Caches/lab07$ ./transpose 2000 20
Testing naive transpose: 5.067 milliseconds
Testing transpose with blocking: 7.421 milliseconds
xe-user106@noman-10xengineers:~/10x-Engineers/Remedial-Training/R4: Computer Arc
hitecture/Caches/lab07$ ./transpose 5000 20
Testing naive transpose: 89.711 milliseconds
Testing transpose with blocking: 48.241 milliseconds
xe-user106@noman-10xengineers:~/10x-Engineers/Remedial-Training/R4: Computer Arc
hitecture/Caches/lab07$ ./transpose 10000 20
Testing naive transpose: 519.094 milliseconds
Testing transpose with blocking: 208.949 milliseconds

```

1. At what point does cache blocked version of transpose become faster than the non-cache blocked version?

The cache blocked version of the transpose becomes faster than that of non-blocked version. The transition occurs somewhere between the matrix size of 2000 to 2300. We can further investigate with different matrix sizes to be more accurate. Here's a more appropriate guess:

```

xe-user106@noman-10xengineers: ~/10x-Engineers/Remedia...
xe-user106@noman-10xengineers:~/10x-Engineers/Remedial-Training/R4: Computer Arc
hitecture/Caches/lab07$ ./transpose 2000 20
Testing naive transpose: 5.05 milliseconds
Testing transpose with blocking: 7.367 milliseconds
xe-user106@noman-10xengineers:~/10x-Engineers/Remedial-Training/R4: Computer Arc
hitecture/Caches/lab07$ ./transpose 2000 20
Testing naive transpose: 5.041 milliseconds
Testing transpose with blocking: 7.17 milliseconds
xe-user106@noman-10xengineers:~/10x-Engineers/Remedial-Training/R4: Computer Arc
hitecture/Caches/lab07$ ./transpose 2200 20
Testing naive transpose: 8.152 milliseconds
Testing transpose with blocking: 9.602 milliseconds
xe-user106@noman-10xengineers:~/10x-Engineers/Remedial-Training/R4: Computer Arc
hitecture/Caches/lab07$ ./transpose 2300 20
Testing naive transpose: 14.256 milliseconds
Testing transpose with blocking: 10.448 milliseconds
xe-user106@noman-10xengineers:~/10x-Engineers/Remedial-Training/R4: Computer Arc
hitecture/Caches/lab07$ ./transpose 2200 20
Testing naive transpose: 8.223 milliseconds
Testing transpose with blocking: 9.096 milliseconds
xe-user106@noman-10xengineers:~/10x-Engineers/Remedial-Training/R4: Computer Arc
hitecture/Caches/lab07$ ./transpose 2250 20
Testing naive transpose: 10.719 milliseconds
Testing transpose with blocking: 9.453 milliseconds
xe-user106@noman-10xengineers:~/10x-Engineers/Remedial-Training/R4: Computer Arc
hitecture/Caches/lab07$

```

2. ***Why does cache blocking require the matrix to be a certain size before it outperforms the non-cache blocked code?***

The blocked version isn't faster than the naive version until the matrix size is sufficiently big enough because its effectiveness depends on factors like matrix size, cache size, and block size.

When the matrix is small enough to fit entirely in the cache, cache blocking may not provide significant benefits compared to the naive version. Overhead from managing blocks can offset gains.

Choosing an appropriate block size is crucial; too small leads to overhead, while too large may not fit well in the cache. Cache blocking becomes effective when the matrix size exceeds the cache capacity, allowing better data reuse within each block.

In essence, cache blocking shines when dealing with large matrices that exceed cache limits, improving cache utilization and reducing cache misses.

➤ **Part II: Changing Block Size:**

■ **Terminal Output:**

```

xe-user106@noman-10xengineers: ~/10x-Engineers/Remedia...
xe-user106@noman-10xengineers:~/10x-Engineers/Remedial-Training/R4: Computer Arc
hitecture/Caches/lab07$ ./transpose 10000 50
Testing naive transpose: 526.87 milliseconds
Testing transpose with blocking: 175.531 milliseconds
xe-user106@noman-10xengineers:~/10x-Engineers/Remedial-Training/R4: Computer Arc
hitecture/Caches/lab07$ ./transpose 10000 100
Testing naive transpose: 522.513 milliseconds
Testing transpose with blocking: 142.038 milliseconds
xe-user106@noman-10xengineers:~/10x-Engineers/Remedial-Training/R4: Computer Arc
hitecture/Caches/lab07$ ./transpose 10000 500
Testing naive transpose: 529.884 milliseconds
Testing transpose with blocking: 98.049 milliseconds
xe-user106@noman-10xengineers:~/10x-Engineers/Remedial-Training/R4: Computer Arc
hitecture/Caches/lab07$ ./transpose 10000 1000
Testing naive transpose: 517.761 milliseconds
Testing transpose with blocking: 119.205 milliseconds
xe-user106@noman-10xengineers:~/10x-Engineers/Remedial-Training/R4: Computer Arc
hitecture/Caches/lab07$ ./transpose 10000 5000
Testing naive transpose: 526.225 milliseconds
Testing transpose with blocking: 477.383 milliseconds
xe-user106@noman-10xengineers:~/10x-Engineers/Remedial-Training/R4: Computer Arc
hitecture/Caches/lab07$

```

3. *How does performance change as blocksize increases? Why is this the case?*

As we increase the block size from 50 to 500 in the transpose operation, cache blocking methods show significant decreases in time taken, with the cache blocking method consistently outperforming the naive method at each step.

For instance, with a block size of 50, the naive transpose method takes 526.87 milliseconds, while the cache blocking method takes only 175.531 milliseconds. Similarly, at a block size of 500, the naive method takes 529.884 milliseconds, while the cache blocking method achieves a faster time of 98.049 milliseconds.

However, beyond a block size of 1000, the advantage of cache blocking diminishes slightly, with the time increasing to 119.205 milliseconds. This could be attributed to increased memory usage or suboptimal parallelism, where the system starts to encounter diminishing returns due to larger block sizes, hence not achieving further performance improvements.

Nevertheless, the maximum performance, with a time of 98.049 milliseconds, is attained with a block size of 500 in the cache blocking

method, indicating an optimal balance between reducing overhead and maximizing cache efficiency.