CEng 355

Assignment 5

Jakob Roberts
v00484900

1

```
for (p = 0; p < 2; p++) {
        for (q = 0; q < 2; q++) {
                for (i = p*64; i < (p+1)*64; i++) {
                        for (j = q*64; j < (q+1)*64; j++) {
                                Y[i] = Y[i] + A[i][j]*X[j];
                        }
                }
        }
}
```

@p = 0, Y[0:63] is computed in 2 steps.

1: use A[0:63][0:63] with X[0:63] @ q=0.

2: use A[0:63][64:127] with X[64:127] @ q=1.

@p = 1, Y[64:127] is computed in 2 steps.

1: use A[64:127][0:63] with X[0:63] @ q=0.

2: use A[64:127][64:127] with X[64:127] @ q=1.

To store A (64x64), 64 x 64 x 4 = 16KB of space
To store two 128x1 blocks, 2 x 128 x 4 = 1KB of space
Therefore Cache size should be **17KB**

2

NOTE: Assume int of size 32bit

With N = 256, int size X[256][256] $\Rightarrow$ 256 x 256 x 4 = 256KB.
Each row requires $256 \times 4 = 1KB$ of space.
For the outer loop, per iteration on index i, there are 2 reads and 1 write per row.
Each row then requires $3 \times 256 = 768$ *accesses*.

With 4 x 1KB pages: 4 page faults per 4 rows.
Fault Rate: $4/(4 \times 768) = 0.13\%$
With 1 x 4KB pages: 1 page faults per 4 rows.
Fault Rate: $1/(4 \times 768) = 0.03\%$

3

NOTE: Assume float of size 32bit

With N = 256, float size X[256][256] $\Rightarrow$ 256 x 256 x 4 = 256KB.
For first for loop (trace), 256 accesses are done as it computes 1 entry per row along the diagonal.
For the second nested loops, it is reading every single element in every row. Because it is doing a read and assign, it will need to access twice per element, therefore 512 times per row.
Total accesses are 256 from the trace, and 512 per row. $\Rightarrow$ $256 + (256 \times 512) = 131,328$ *accesses*

With 4 x 1KB pages: 256/4=64 page faults from first loop, 256/4=64 page faults from second loops.
Fault Rate: $(256 + 256)/131,328 = 0.3899\%$
With 1 x 4KB pages: 256 page faults from first loop, 256 page faults from second loops.
Fault Rate: $(64 + 64)/131,328 = 0.0975\%$