EE 5351 MP1

Lexi MacLean

1.

Given input matricies $\boldsymbol{M}$, $\boldsymbol{N}$, and $\boldsymbol{P}$ with dimensions $a \times b$, $b \times c$, and $a \times c$ respectively, each element in $\boldsymbol{M}$ will be read once for each column in $\boldsymbol{P}$, and each element in $\boldsymbol{N}$ will be read once for each row in $\boldsymbol{P}$. So, the total reads from each element in $\boldsymbol{M}$ is c, and the total reads from each element in $\boldsymbol{N}$ is a.

2.

A majority of the operations are done in the `for` loop, we are instructed to ignore the storage of the result (assumed in $\boldsymbol{P}$, at least, the final storage is negligible). My `for` loop contains one statement:

```
c += M.elements[M.width * y + i] * N.elements[N.width * i + x];
```

This statement makes two reads from global memory: one from $\boldsymbol{M}$ and another from $\boldsymbol{N}$. The statement also makes six floating point operations: four to compute the indicies of the elements in $\boldsymbol{M}$ and $\boldsymbol{N}$, one to multiply the two factors, and another to add them to the partial sum. The `for` loop itself also contributes an additional two floating point operations to increment and compare `i`. Ignoring operations out-side the `for` loop—which affect the ratio approaching 0 as the size of the input matricies increases—this makes for a $4 : 1$ ratio of floating point operations to memory loads (or $3 : 1$ if you ignore those made by the `for` loop).