Matrix multiplication (15%)

Do matrix multiplication of size 128x128 with some additional operations.

```
for (int i = 0; i < SIZE; i++)
    for (int j = 0; j < SIZE; j++)
        for (int k = 0; k < SIZE; k++)
            C[i][j] = (C[i][j] + A[i][k] * B[k][j]) % 1024;
Elements in A and B are unsigned 16 bits numbers of range [0,1023]
```

We will score based on the cycle counts. You can use C as an initial attempt.

```
asm volatile ("rdcycle \%0" : "=r" (start));
// matrix multiplication
asm volatile ("rdcycle \%0" : "=r" (end));
```

Grading:

- Below 20,000,000 cycles (2%)
- Below 18,000,000 cycles (2%)
- Below 16,000,000 cycles (2%)
- Below 14,000,000 cycles (2%)
- Below 12,000,000 cycles (2%)
- Below 10,000,000 cycles (1%)
- Below 9,000,000 cycles (1%)
- Below 8,000,000 cycles (1%)
- Below 7,000,000 cycles (1%)
- Below 6,000,000 cycles (1%)

Report on matrix multiplication (15%)

- Briefly explain how you get below 6,000,000 cycles.
- Or you can answer the following questions:
  - How many cycles does it take by just doing the naive matrix multiplication?
  - How many load and store does it need (roughly) during the whole computation? (Considering the registers it use)
  - Is there any way to keep registers being used as much as possible before they're replaced? (Hint: blocking)
  - How many loop controls does it need (roughly) during the whole computation?

Submission

- All *.s file should be written assembly.
- Zip and upload your file to NTU cool in the following format, the file name should be ⟨Student ID⟩.zip:

```
b08922000              <-- zip this folder
    |-- fibonacci.s
    |-- convert.s
    |-- matrix.s
    `-- report.pdf     // including handwritten part and report on matrix multiplication part
```

- If there's any question, please send email to 110fall.ca@gmail.com.