Q1:

The loop unrolling version is slightly faster because it eliminates some of the code required to manage induction variables.

Q2:

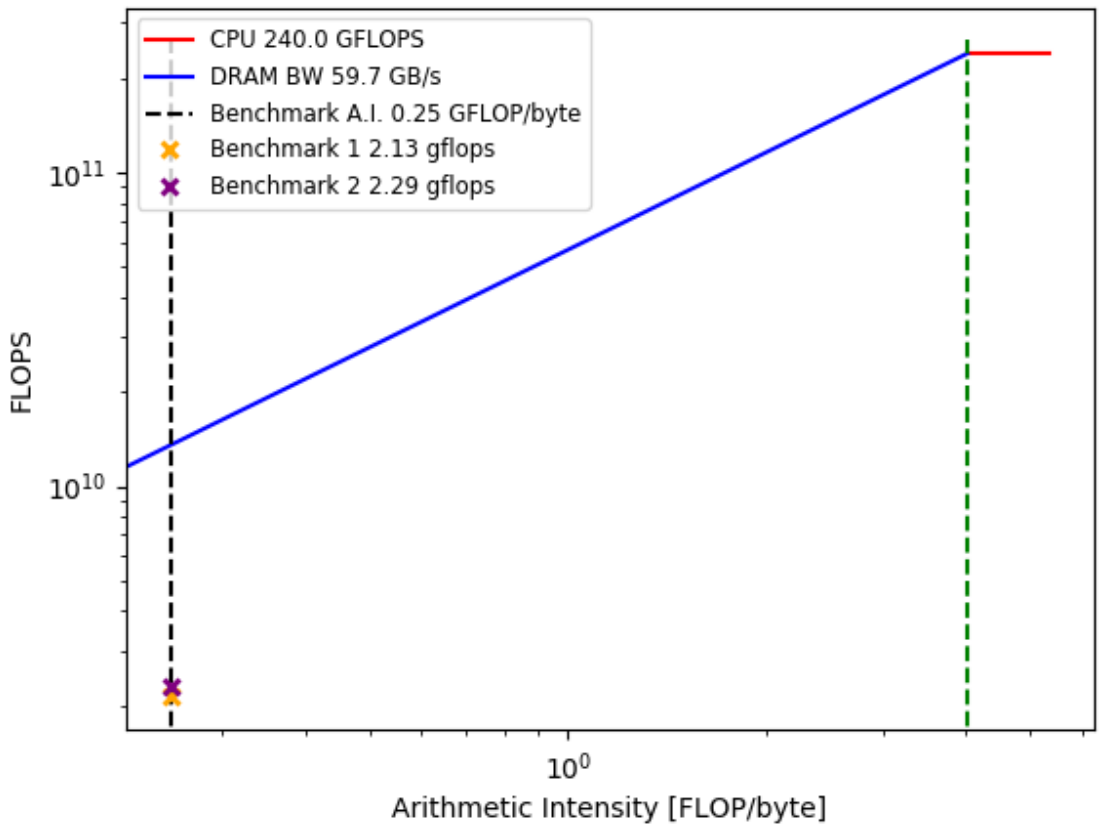
Intel(R) Xeon(R) CPU E5-2690 v2 @ 3.00GHz is an Ivy Bridge, which supports 2 loads per cycle. However, the dependency chain in both C1 and C2 prevents this from happening, forcing at most 1 load per cycle. If sum were calculated in the following way

```
double sum = 0;
double sum1 = 0;
for (i = 0; i < ARRAY_SIZE; i+=2) {
      sum += array[i]*2;
      sum1 += array[i + 1]*2;
}
sum += sum1;
```

then at each iteration, the computation of sum1 would not depend on the computation of sum, so they can be computed at the same time. `array[i]` and `array[i + 1]` would be loaded in one cycle.

Q3:

CPU Peak FLOPS: 10 cores * 3.00 GHz * 8 DP FLOPs/cycle = 240 GFLOPS
DRAM : 59.7 GB/s
A.I.: 0.25 FLOP / byte
C1 measurement: 2.13 GFLOP/s
C2 measurement: 2.29 GFLOP/s



Q4:

W_0 = (4000, 65536)
x_0 = (65536, 1)
z_0 = (4000, 1)

W_1 = (1000, 4000)
x_1 = (4000, 1)
z_1 = (1000, 1)

Q5:

$W_0 = 4000 \times 65536 \times 8$ bytes = 2.097152 GB
$x_0 = 65536 \times 1 \times 8$ bytes = 0.524288 MB
$z_0 = 4000 \times 1 \times 8$ bytes = 32 KB

$W_1 = 1000 \times 4000 \times 8$ bytes = 32 MB
$x_1 = 4000 \times 1 \times 8$ bytes = 32 KB
$z_1 = 1000 \times 1$ bytes = 1 KB

Note: For C5 and C6, "Speedup w.r.t. C3" **uses the overall lowest recorded duration for C3: 34.58s**.

Output from C1 to C6:

C1
Time: 0.63 secs
Bw: 8.52 GB/s
FLOPS: 2.13 GFLOP/s

C2
Time: 0.59 secs
Bw: 9.14 GB/s
FLOPS: 2.29 GFLOP/s

C3
Time: 46.111878260970116 secs
Checksum: 15250108999.999886

C4
Time: 7.463245170016307 secs
Checksum: 15250109000.0
Speedup w.r.t. C3: 6.178529206869049

C5
Time: 0.3072942640 secs
Checksum: 15250108999.999886
Speedup w.r.t. C3: 112.518239

C6
Time: 0.1934000140 secs
Checksum: 15250109000.000004
Speedup w.r.t. C3: 178.780801