Optimizing relational queries over bit-parallel database layout

Reading:

- BitWeaving: One early attempt on bit parallel database layout https://15721.courses.cs.cmu.edu/spring2016/papers/li-sigmod2013.pdf
- MLWeaving: One recent attempt on bit parallel database layout taking into consideration different precision levels https://arxiv.org/pdf/1903.03404.pdf

Work Packages

This project is organized in three levels. We expect you to at least do a good job finishing Level 1 and Level 2.

Level 1. (Warm-up) Let R be a relation stored in the MLWeaving layout, optimize the following SELECT query:

```
SELECT * FROM R WHERE R.a < R.b;
```

and AGGREGATE query:

```
SELECT SUM(c) FROM R WHERE R.a < R.b;
```

(You don't need to write the parser, your program can hard code `a, b, c`)

Level 2. (Harder) Let R and S be two relations stored in the MLWeaving layout, optimize a join query such as

```
SELECT * FROM R, S WHERE R.a % S.b = S.c;
```

(implement the nested-loop join algorithm, your program can hard code `a, b, c`)

Level 3. You can be creative and to go further in different directions.

Possibility 1: Support different precision levels.

```
SELECT * FROM R WHERE R.a < R.b WITH INPUT PRECISION X bit;
SELECT SUM(c) FROM R WHERE R.a < R.b WITH INPUT PRECISION X bit;
SELECT * FROM R, S WHERE R.a % S.b = S.c WITH INPUT PRECISION X bit;
```

(Bonus point: think about how to generalize your design to support different precision levels)

Possibility 2: Can we optimize and twist the MLWeaving layout to better support these queries?

Possibility 3: your ideas

Use the different functions one can implement to divide among team members.