**S1 Algorithm. Database processing**

**score(query, database, use\_simd)**

short, long = split(database)

 exec\_in\_parallel(CPU\_long(), short())

wait\_for(short())

 stop(CPU\_long())

GPU\_long(query, long[last\_cpu\_long ... len(long)], use\_simd)

if (use\_simd)

 database1 = detect\_overflows(database)

 score(quert, database1, false)

**short()**

 exec\_in\_other\_thread(CPU\_short())

 for (l in 1 ... len(short), step = N

 if (l + N > first\_cpu\_short)

 stop(CPU\_short())

 GPU\_short(query, short[l, l + N], use\_simd)

**CPU\_short()**

 for (l in len(short) ... 1)

 if use\_simd

 send\_to\_thread\_pool(CPU\_SIMD(query, l))

 else

 send\_to\_thread\_pool(CPU(query, l))

 for (l in len(short) ... 1)

 if (should\_stop()) stop

 wait\_for\_thread\_pool(l)

 first\_cpu\_short = l

**CPU\_long()**

 for (l in 1 ... len(long))

 if use\_simd

 send\_to\_thread\_pool(CPU\_SIMD(query, l))

 else

 send\_to\_thread\_pool(CPU(query, l))

 for (l in 1..len(long))

 if (should\_stop()) stop

 wait\_for\_thread\_pool(l)

 last\_cpu\_long = l