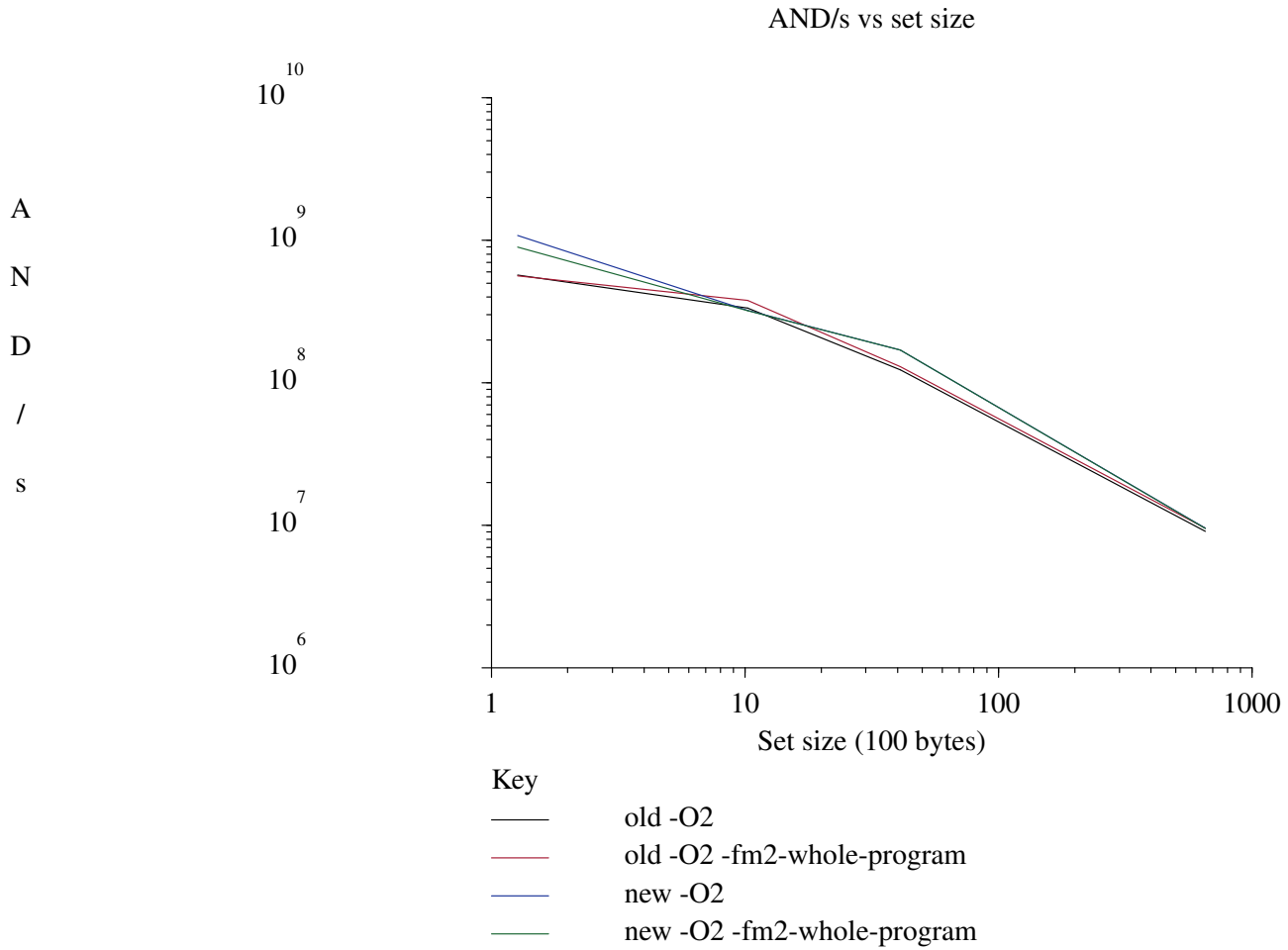
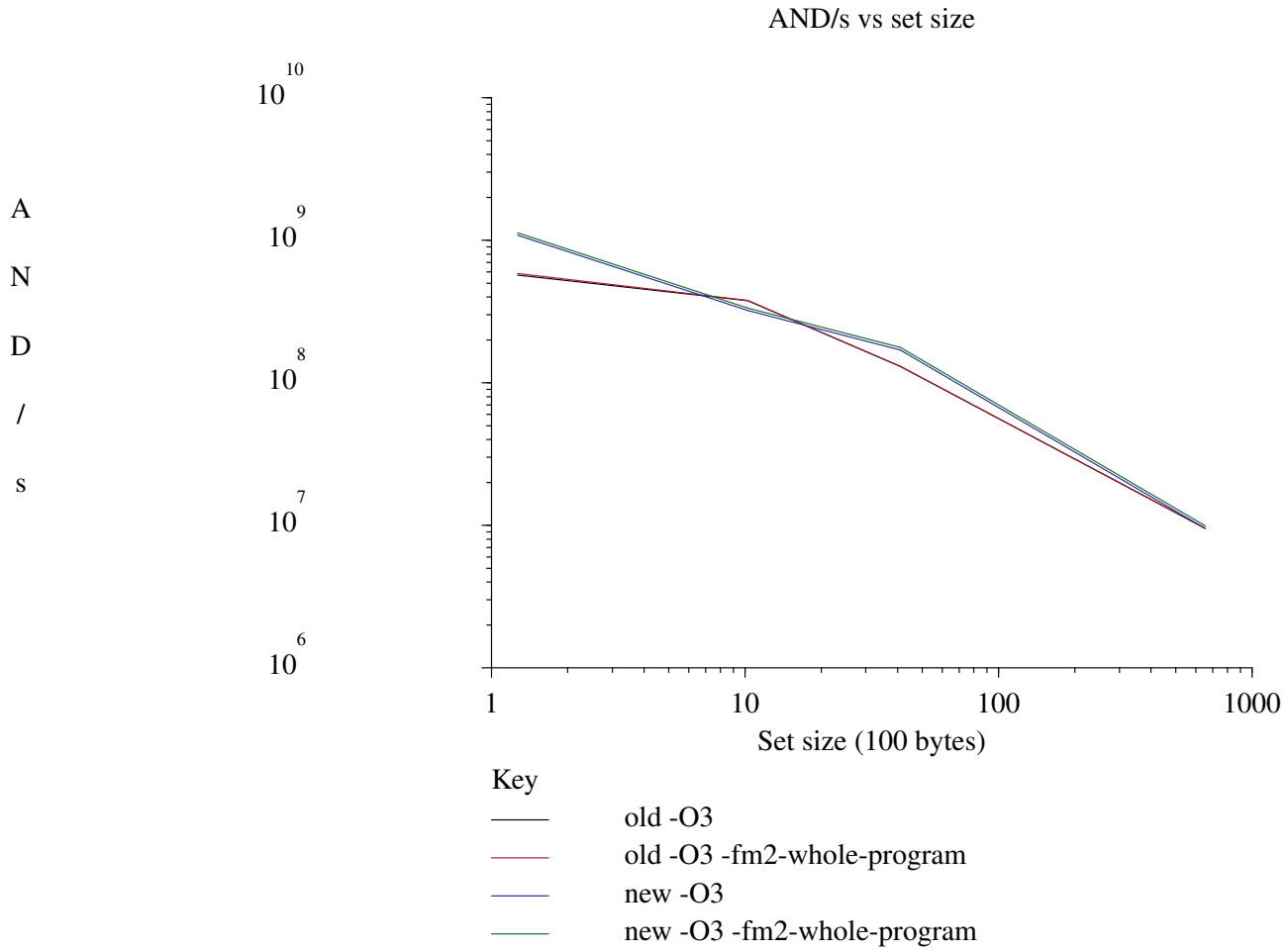
Performance comparision of the new implementation of widesets in gm2

- the old wideset implementation used a `struct` with n word sized fields
- whereas the new implementation uses an `ARRAY OF BYTE`
  - an `INCL`, `EXCL`, `AND`, `OR` `NOT` etc will result in a call to the library module `M2WIDESET` rather than generate `trees` internally as in the previous implementation
- all measurements performed on a 3Ghz AMD Ryzen 9 7950X3D 16-Core Processor

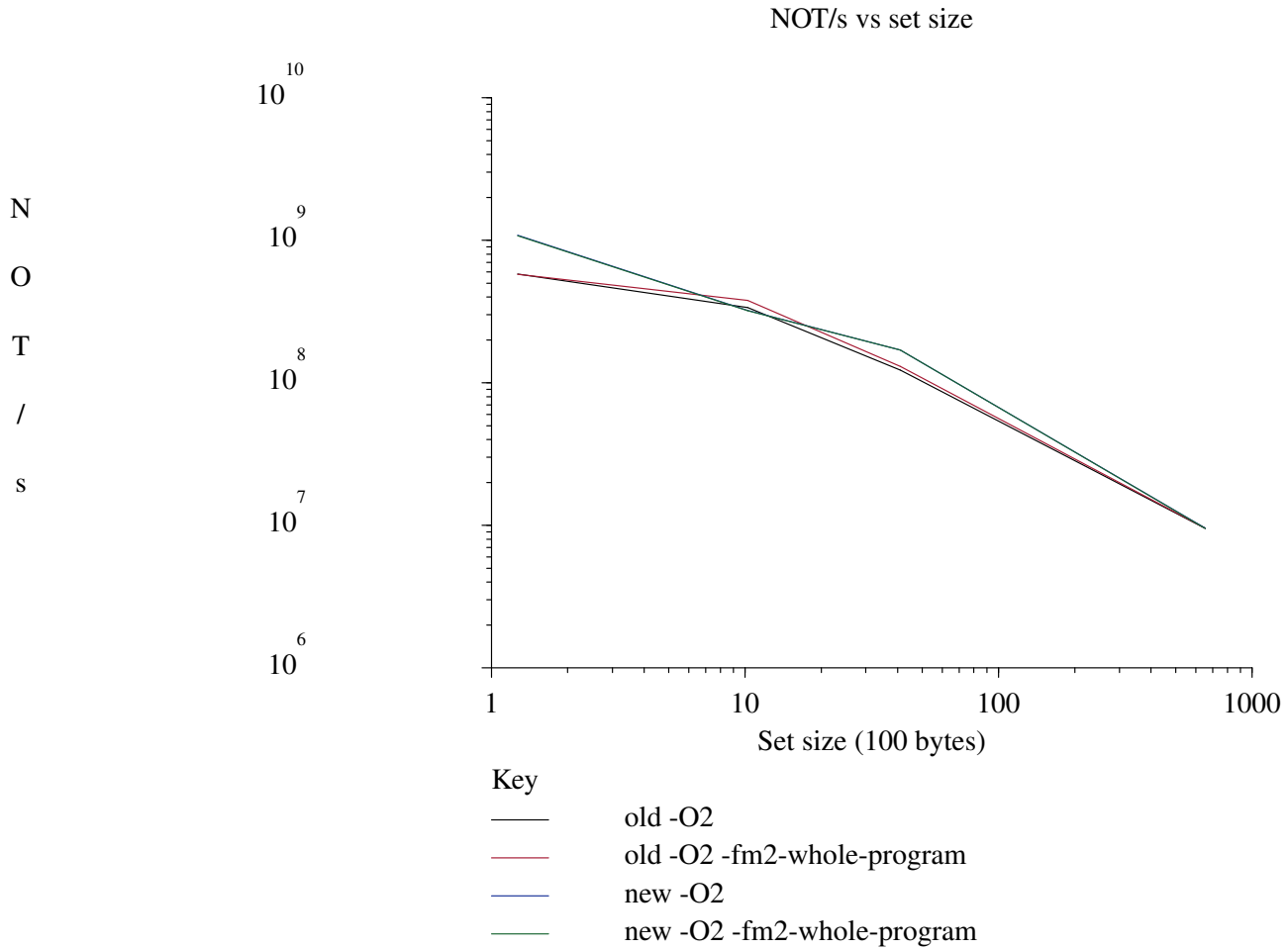
Performance comparision of AND operations on set sizes -O2



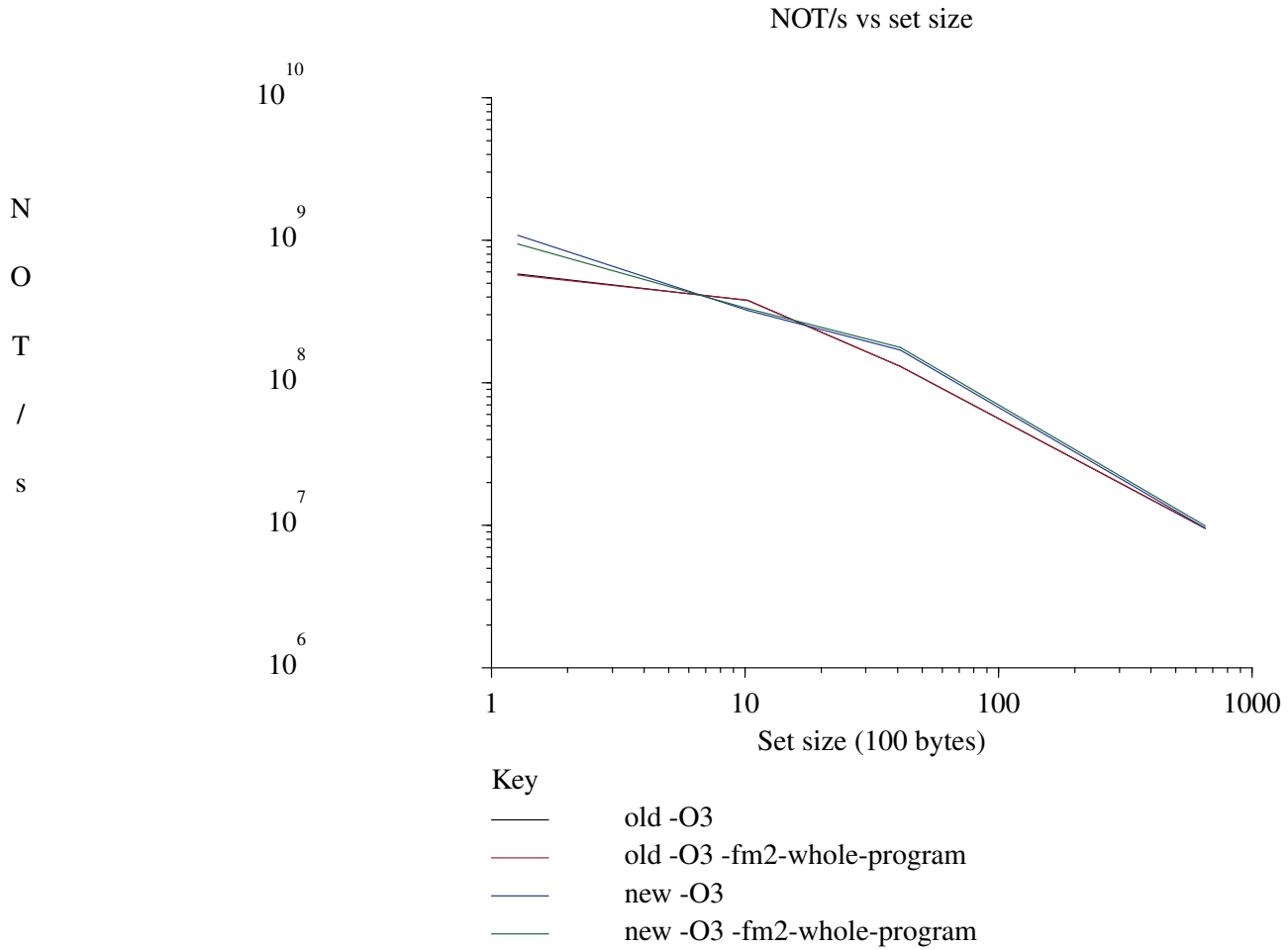
Performance comparision of AND operations on set sizes -O3



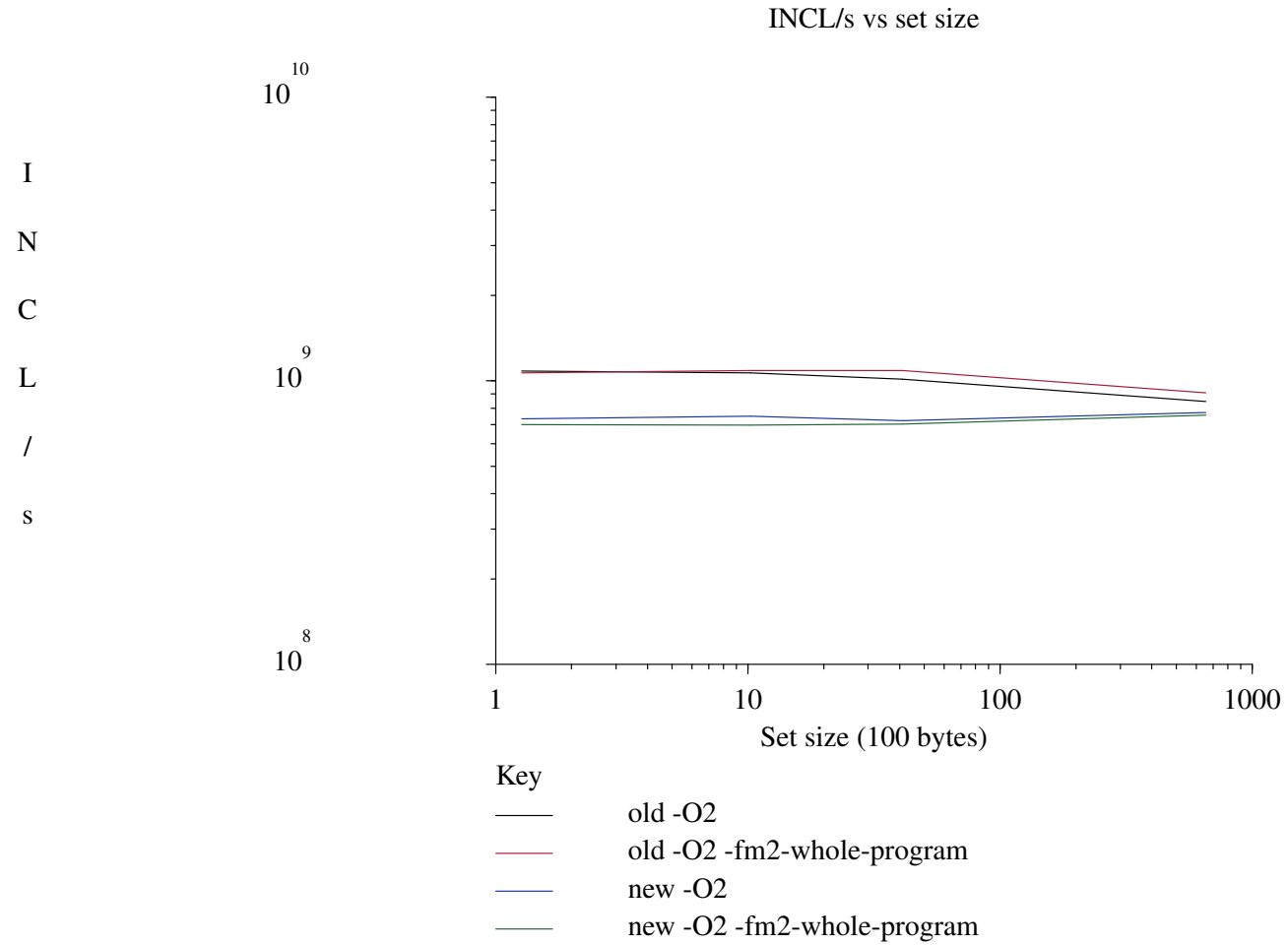
Performance comparision of NOT operations on set sizes -O2



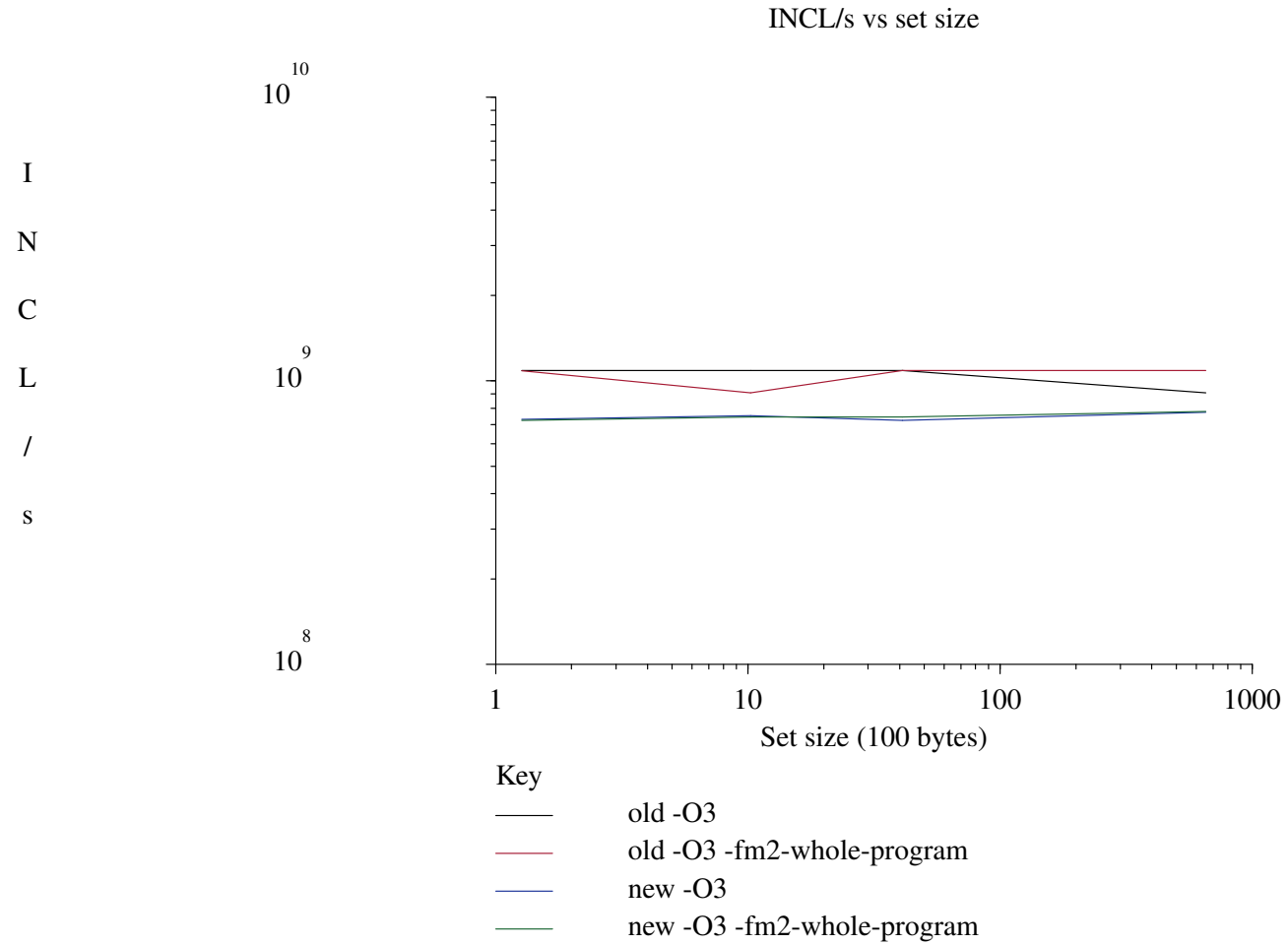
Performance comparision of NOT operations on set sizes -O3



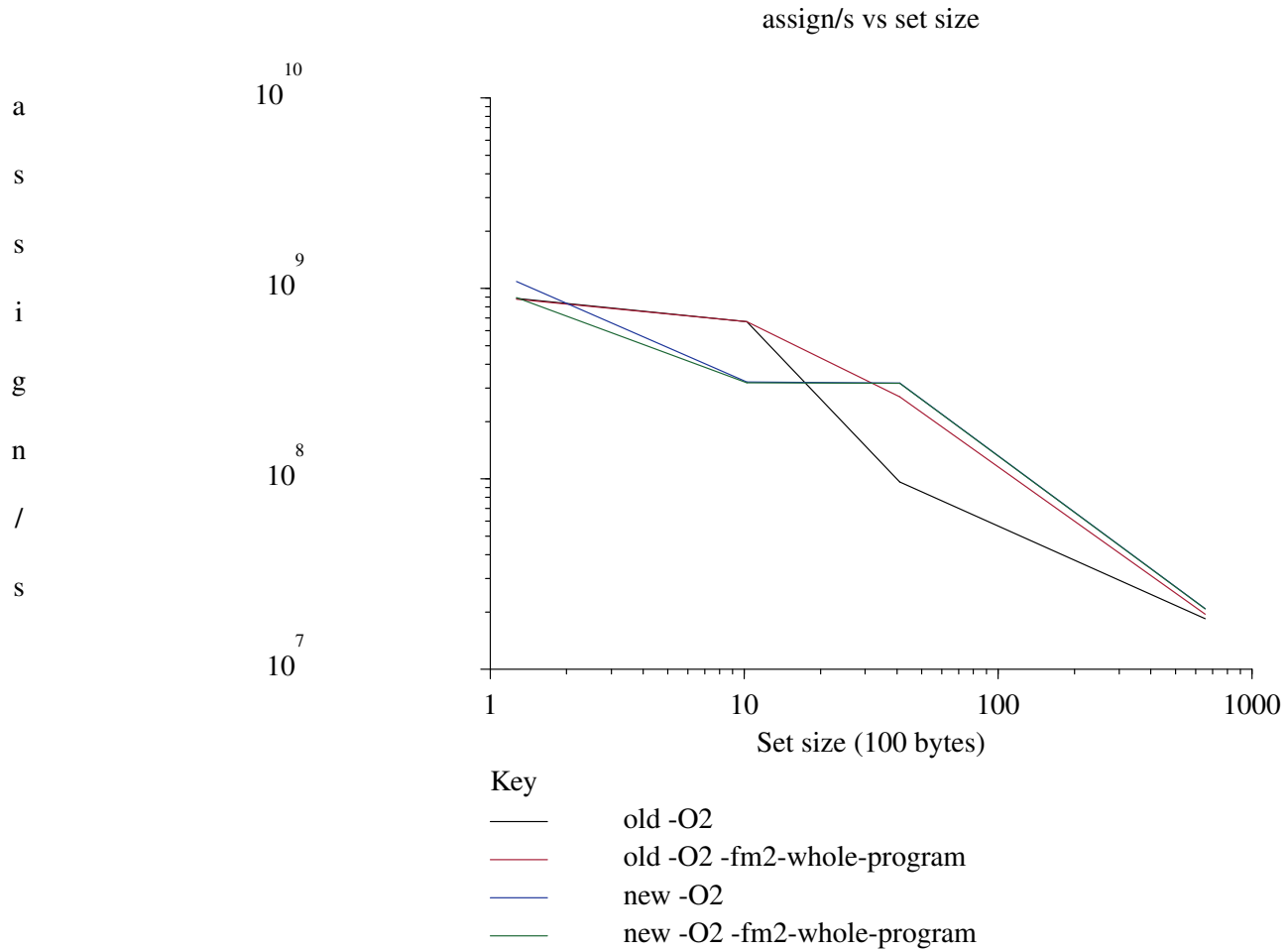
Performance comparision of INCL operations on set sizes -O2



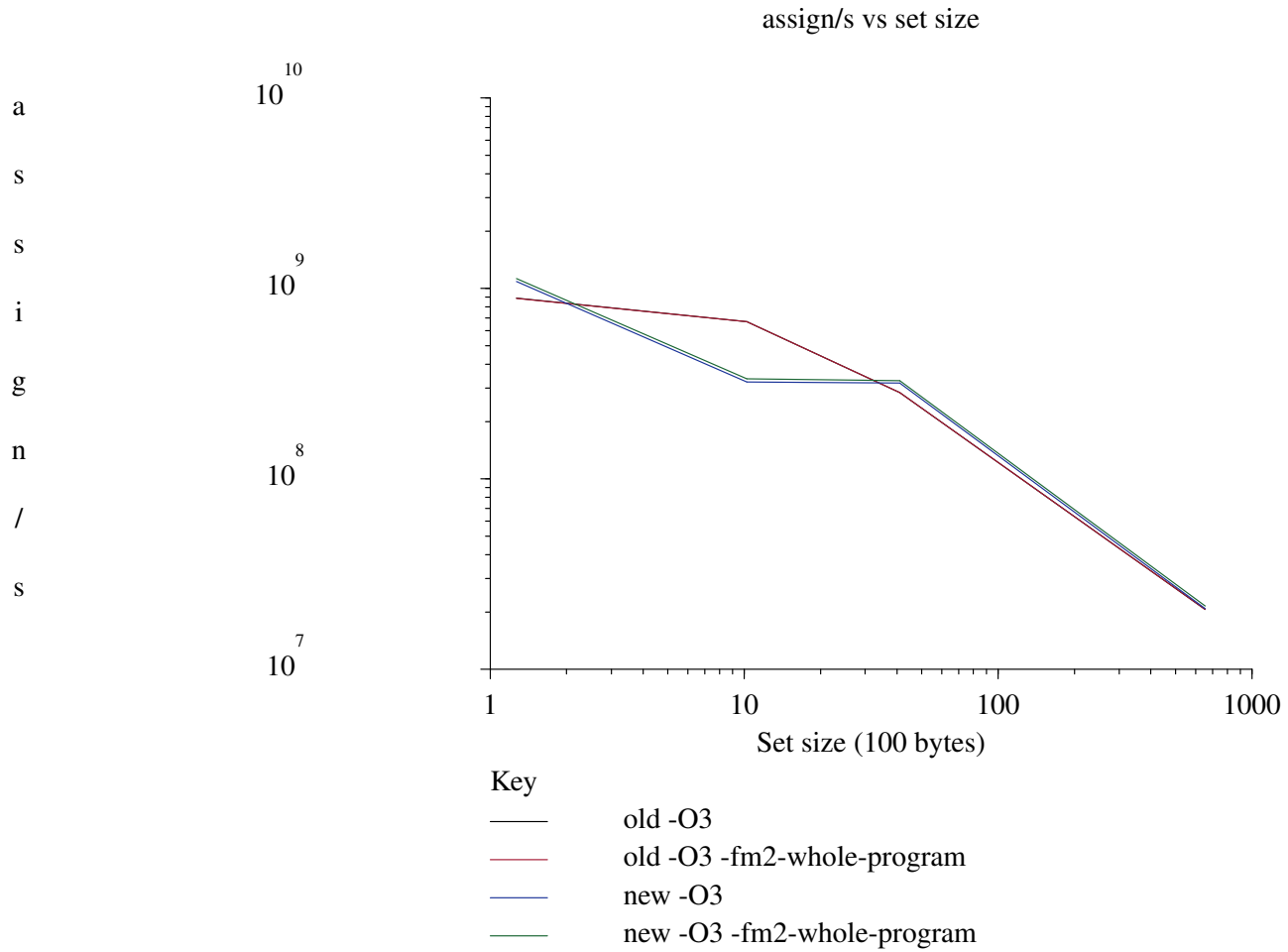
Performance comparision of INCL operations on set sizes -O3



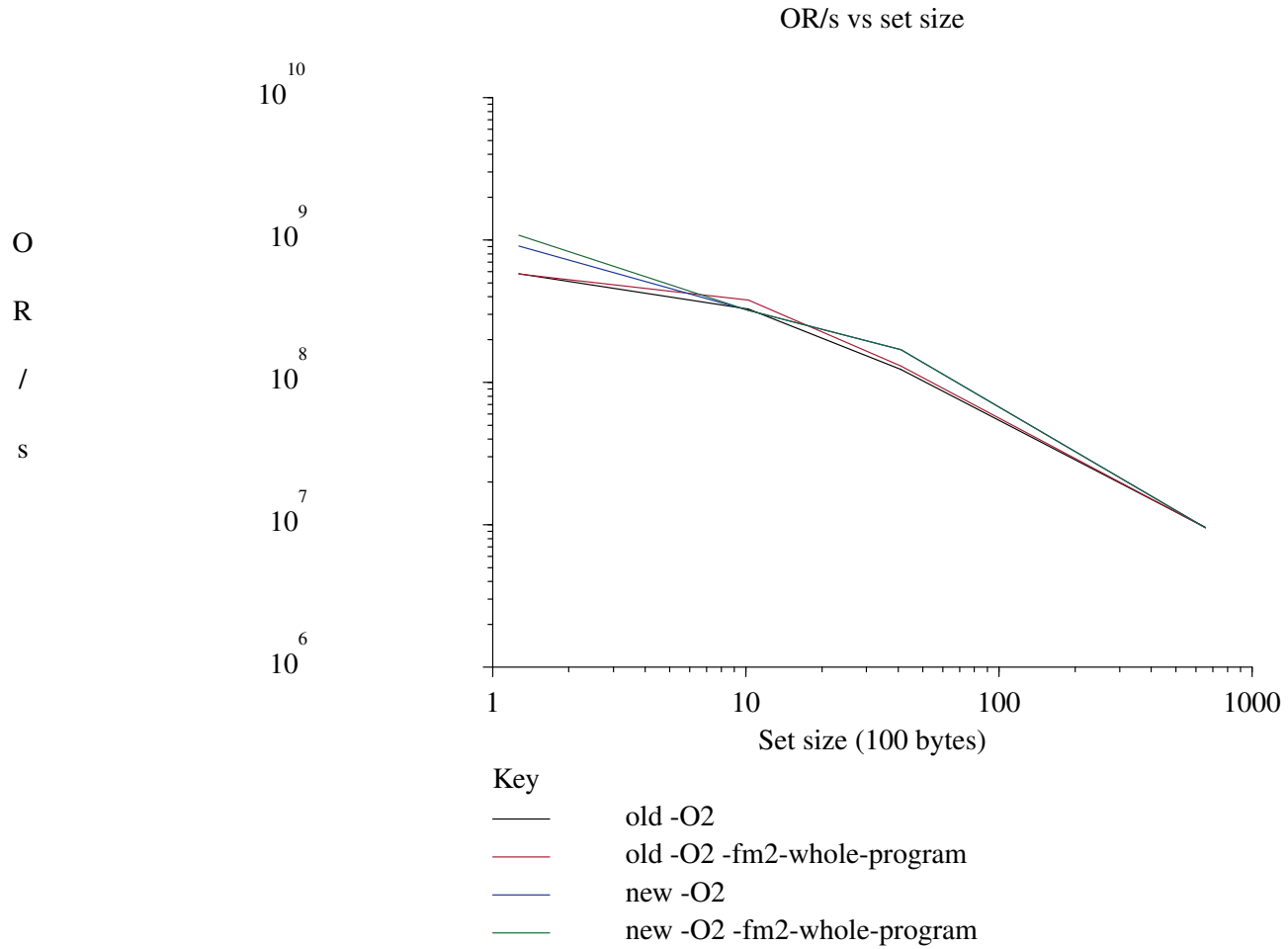
Performance comparision for assignment on set sizes -O2



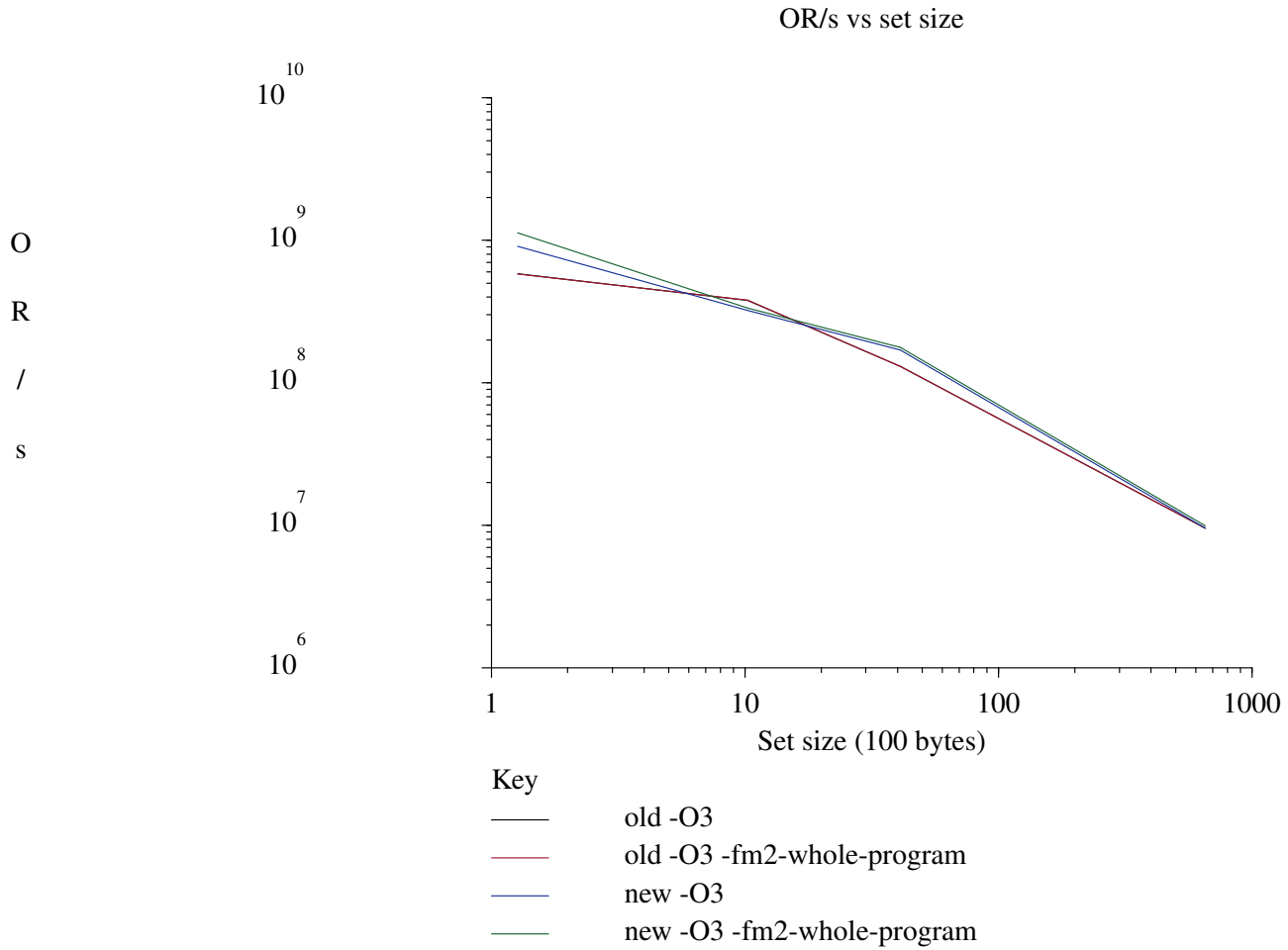
Performance comparision for assignment on set sizes -O3



Performance comparision of OR operations on set sizes -O2



Performance comparision of OR operations on set sizes -O3



Conclusions

- the new implementation does not use word size operations yet
  - all operations use byte operands
- nevertheless `AND`, `OR`, `NOT` are nearly always faster in the new implementation
- the new implementation of `INCL` is slightly slower

Future work

- the set libraries need to use multi word operands rather than byte operands
- consider inlining the `INCL` when `-O2` or `-O3` is specified
- `pragma` for inlining selected modules needs to be implemented