Re-implementation of the widesets in GNU Modula-2 (gm2-16)

- What is a wide set in gm2?
  - Any set occupying more than one `WORD` of data storage.
- for example:
- ```
  TYPE
     Range = [0..4095] ;
     Set   = SET OF Range ;
  ```
  ```

Re-implementation of the widesets in GNU Modula-2 (gm2-16)

- here the set is defined as containing `4096` elements which are represented in `512` bytes

How was wideset represented prior to gm2-16?

- internally in `gm2-14` the previous set is implemented as a `struct` containing `128` fields
  - each field is a 32 bit word

How was wideset represented prior to gm2-16?

- this approach is runtime fast but compile time heavy
  - it can also lead to a large code size when performing `AND`, `OR` and `NOT` operations for example
  - currently these operators result in fast in-lined code but there is no provision for `-Os`
- although `-Os` could be bolted on to this implementation, it probably warrants a re-think
- the `struct` implementation of a wideset was conceived two decades ago and since then `M2R10` has been proposed
- the `M2R10` language update stipulates that each data type has its own library module containing the appropriate operators

Re-implementation of wideset

- the obvious re-implementation of wideset is to use `ARRAY [0..n] OF BYTE` and introduce a module `M2SET` which contain operators `AND`, `OR`, `NOT`, `IN`, `INCL` and `EXCL` etc.

- the motivation for this blog is to report on some set benchmarks

Performance comparison between old and new implementation

- --got to here--

initially the development implemented the library module procedure functions and timings were obtained by comparing

  - `gm2-14` with the new `gm2-15` using the compiler options `-O3 -fm2-whole-program`
  - alas the timings showed that this approach was between `2-8` times slower than `gcc-14`

- the development was then changed to use the library module when `-O0` or `-Os` is requested
  - but use inline versions of: `AND`, `OR`, `NOT`, `INCL`, and `EXCL` when `-On` for `n > 0`.

Performance comparison between old and new implementation

- the results below compare the performance of `gm2-14` against the inlined `ARRAY OF BYTE` set implementation
  - the results were obtained on an x86_64 AMD Ryzen 5 3600 6-Core Processor platform

gm2 -g -O3 timeset1.mod

Set bits	operation	no. ops/sec	gm2 version
127	s := {}	5.796e+08	gm2-14
		8.213e+08	gm2-15
	INCL	8.212e+08	gm2-14
		5.139e+08	gm2-15
	AND	3.774e+08	gm2-14
		8.214e+08	gm2-15
	OR	3.772e+08	gm2-14
		8.214e+08	gm2-15
	NOT	3.647e+08	gm2-14
		8.213e+08	gm2-15
1023	s := {}	9.712e+07	gm2-14
		1.037e+08	gm2-15
	INCL	5.136e+08	gm2-14
		5.138e+08	gm2-15
	AND	2.283e+08	gm2-14
		4.533e+08	gm2-15
	OR	2.283e+08	gm2-14
		4.564e+08	gm2-15
	NOT	2.278e+08	gm2-14
		4.564e+08	gm2-15

4095	s := {}	4.33e+07	gm2-14
		4.618e+07	gm2-15
	INCL	7.763e+08	gm2-14
		5.146e+08	gm2-15
	AND	6.268e+07	gm2-14
		1.204e+08	gm2-15
	OR	6.263e+07	gm2-14
		1.204e+08	gm2-15
	NOT	6.23e+07	gm2-14
		1.203e+08	gm2-15
65535	s := {}	1.244e+07	gm2-14
		1.303e+07	gm2-15
	INCL	8.212e+08	gm2-14
		5.133e+08	gm2-15
	AND	4.031e+06	gm2-14
		4.031e+06	gm2-15
	OR	4.032e+06	gm2-14
		4.031e+06	gm2-15
	NOT	4.032e+06	gm2-14
		4.03e+06	gm2-15

Conclusion

- the speed improvements for set operators `:=`, `AND`, `OR`, `NOT` are encouraging
  - not least as the last three operators were implemented using byte operations.
  - in the future these will be changed to use word operations

- The slight slow down of `INCL` and presumably `EXCL` requires more analysis. These operations are very short by comparision (an array index and set/unset the appropriate bit)