A report on the progress of GNU Modula-2 and its potential integration into GCC

Gaius Mulley <gaius@gnu.org>

Why Modula-2 today?

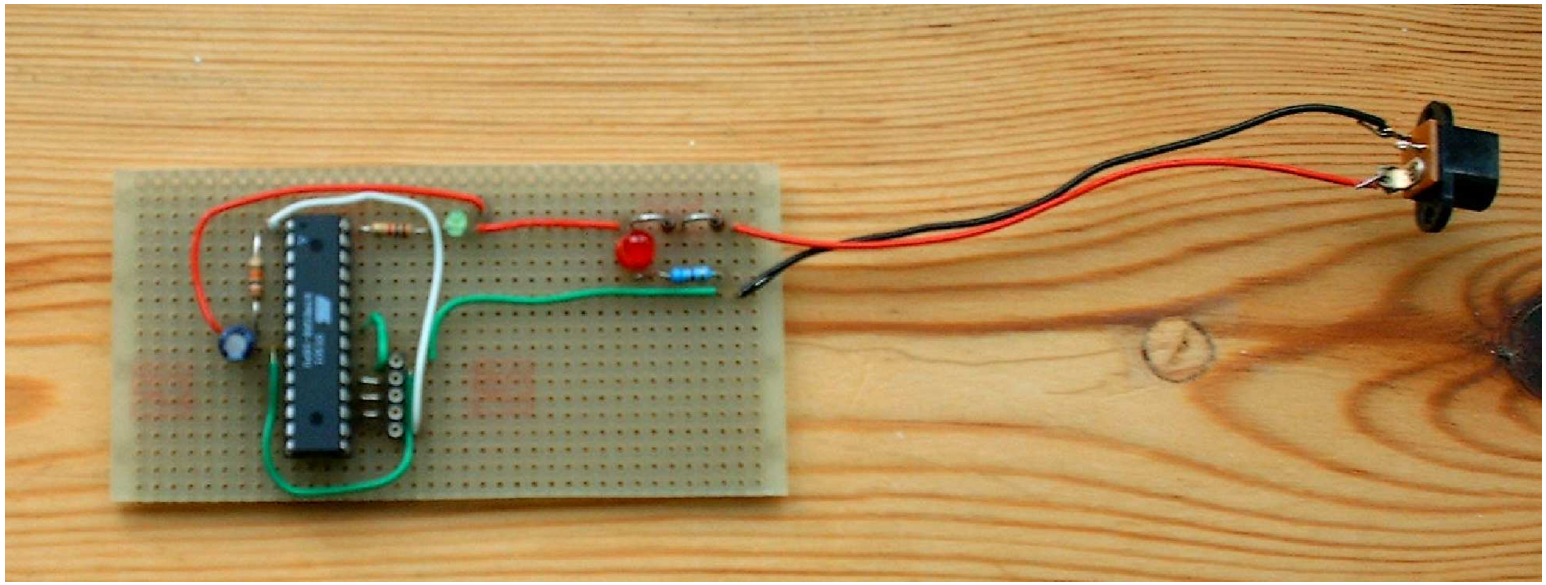
- “source code which cannot port to another architecture will die”
  Mike Gancarz

- legacy code
  - academic
  - industrial

- embedded systems of today
  - low memory footprint, bit manipulation, memory mapped variables, coroutines, interrupt priorities

- great teaching language

Embedded systems and Modula-2



- the microprocessor is an ATmega8
  - 8 KB of flash memory and 1 KB of RAM

Python and GNU Modula-2

- GNU Modula-2 can easily be compiled to produce Python modules
- `-fswig` option automatically generates a swig interface file
- three command lines will transform an implementation module into a Python module
  - see the gravity demo

GNU Modula-2 overview

- PIM [234] and ISO compliant
- a full set of PIM libraries are available
- a full set of ISO libraries are available
  - command line switches to force dialect and libraries
- two categories of language extensions
  - firstly, follow the tradition of other GCC front ends: `-fcpp`, inlining of built-in functions/constants and access to assembly language
  - secondly, allow easy access to C libraries

GNU Modula-2 overview

- enhances a number of language features:
  - sets can be declared from any ordinal type
  - abstract data types are not restricted to a pointer type
  - procedures, types, variables, enumerations, constants, composite types may be declared in any order

Goals of GNU Modula-2

- to fold the gm2 source code into the gcc tree at a convenient time in the future
- exploit the features of GCC
- libraries
  - PIM library compatibility
  - a reimplementation of the Logitech (PIM) libraries are available
  - Ulm (PIM) are also available
- an easy interface to C

Goals of GNU Modula-2

- listen to the requests of the users

Extensions: access to C

- `-fcpp` will invoke the C preprocessor in traditional mode using assembler as a base language
  - useful to turn on/off debugging code

- similar to the Fortran front end

easy access to C libraries

```
DEFINITION MODULE FOR "C" libc ;

EXPORT UNQUALIFIED printf ;

PROCEDURE printf (a: ARRAY OF CHAR; ...) ;

END libc.
```

- when using `DEFINITION MODULE FOR "C"` pointers are mapped onto: `SYSTEM.ADDRESS`
  - `ARRAY OF` is mapped onto `void *`
  - all other types are mapped onto their C counterparts

Extensions: access to assembly language

```
PROCEDURE Example (i, j: CARDINAL) : CARDINAL ;
VAR
   k: CARDINAL ;
BEGIN
   ASM VOLATILE ("movl %1,%eax; \
                  addl %2,%eax; movl %eax,%0"
                  : "=g" (k)           (* outputs *)
                  : "g" (i), "g" (j)   (* inputs  *)
                  : "eax") ;           (* we trash *)
   RETURN( k )
END Example ;
```

Extensions: access to GCC built-ins

```
DEFINITION MODULE MathLib0 ;

CONST
   pi  =3.14159265358979323846264338327950288419 72;
   exp1=2.71828182845904523536028747135266249775 72;

PROCEDURE __BUILTIN__ sqrt (x: REAL) : REAL ;
PROCEDURE __BUILTIN__ sqrtl (x: LONGREAL) : LONGREAL ;
PROCEDURE __BUILTIN__ sqrts (x: SHORTREAL) : SHORTREAL ;

PROCEDURE exp (x: REAL) : REAL ;
PROCEDURE exps (x: SHORTREAL) : SHORTREAL ;
```

Extensions: access to GCC built-ins

```
IMPLEMENTATION MODULE MathLib0 ;

IMPORT cbuiltin, libm ;

PROCEDURE __ATTRIBUTE__ __BUILTIN__ ((__builtin_sqrt))
          sqrt (x: REAL): REAL;
BEGIN
   RETURN cbuiltin.sqrt (x)
END sqrt ;

PROCEDURE exp (x: REAL) : REAL ;
BEGIN
   RETURN libm.exp (x)
END exp ;
```

Structure of GNU Modula-2

- it was constructed using a similar structure as other front ends
- the GNU Modula-2 front end is written in C and Modula-2
- conceptually it removes the C front end and replaces GCC with a Modula-2 front end
- leaving the middle and back end alone

Structure of GNU Modula-2

- it uses `flex` and a Modula-2 parser generator to construct a top down recursive descent parser with error recovery
  - uses flex to build a dynamic buffer of all source tokens
  - uses three passes to resolve all imports, exports, enumerated types, abstract data types, constants and generate quadruples
  - each front end Modula-2 symbol table entry is translated into `tree`s and finally the quadruples are translated into `gimple` statement which are presented to the middle end

- front end ensures that only legal symbols and legal source are ever passed to GCC gimple

Some of the compiler options

- extensive runtime checking available:
  - `-findex`, `-frange`, `-fcase`, `-freturn`, `-fwholediv` and `-fnil`
- language dialect can be selected via: `-fpim`, `-fpim2`, `-fpim3`, `-fpim4` and `-fiso`
- `-fpim`, `-fpim2`, `-fpim3`, `-fpim4` and `-fiso` also modify the library search path

Some of the compiler options

- `-fmakeall` compiles the module, all dependant modules and performs the link

- `-fextended-opaque` allows opaque types to be implemented via any type (not just a pointer type)

Some of the compiler options

- `-funbounded-by-reference` optimisation switch which attempts to pass non `VAR` unbounded array parameters by reference
  - avoids the implicit copy inside the callee
  - optimisation is not done if any element is written, the address of the array is calculated, or if the array is passed as a `VAR` parameter

- `-Wpedantic-param-names` ensure procedure parameter names match their definition module counterparts

- a number of checks can be enabled to check for bad programming practice and semantic errors (`-Wstudents`, `-Wpedantic`)

GNU Modula-2 code comparison

Category	Lines
gcc-4.1.2 *.[chy]	1,900,000
fortran	108,000
Ada	878,000
Java	107,000
C++ front end	173,000
Modula-2	258,000

- however this includes all the Modula-2 libraries
  - the libraries for the other front ends (fortran, java, C++, C) are different projects and thus not included
  - it is a reasonable comparison with Ada though

GNU Modula-2 licences

- compiler is GPL v3
- ULM libraries are GPL v3
- ISO, Logitech, PIM and m2f libraries are all LGPL v3
- documentation is under the GNU Free Documentation License, Version 1.2

Download GNU Modula-2

- see the GNU Modula-2 homepage ⟨`http://www.nongnu.org/gm2/homepage.html`⟩
- Debian packages are available via:
-

```
#
#  GNU Modula-2 repo
#

deb http://floppsie.comp.glam.ac.uk/debian/ squeeze main
deb-src http://floppsie.comp.glam.ac.uk/debian/ squeeze main
```

-

```
$ sudo apt-get update
$ sudo apt-get install gm2-doc gm2
```

Conclusions

- work is underway to graft it onto the GCC trunc
- GNU Modula-2 is fully ISO and PIM-[234] Modula-2 compliant
  - a full set of PIM and ISO libraries exist
- natively builds and passes all of its >10000 regression tests on Debian x32_86 and x64_86

Conclusions

- in the past it has been successfully configured as a cross compiler for MinGW and StrongArm
- also built under the following platforms
  - x86, Opteron, Athlon 64, Alpha, Itanium processors running GNU/Linux, Sparc based Solaris, PowerPC MacOS, x86 Open Darwin and the x86 processor running FreeBSD
- huge thanks to all GCC developers for great target architecture coverage