Modula-2 running on bare metal on the Raspberry Pi-4

- here are some instructions and simple test programs which demonstrate Modula-2 running on the Raspberry Pi-4

- there are a number of steps required:
  - install the tool chain, this will be a cross development tool chain
  - write the tiny test program
  - deploy the binary to target

Cross development tool chain

- references:
  - ⟨https://www.valvers.com/open-software/raspberry-pi/bare-metal-programming-in-c-part-1⟩
  - ⟨https://gist.github.com/preshing/41d5c7248dea16238b60⟩
  - ⟨https://wiki.osdev.org/Raspberry_Pi_Bare_Bones⟩
  - probably other websites were used as well, apologies if these are absent

Install dependancies (on Debian Buster)

```
$ sudo apt install build-essential git gcc g++ flex groff texinfo wget
$ sudo apt install patch flex rsync libgmp3-dev python3-dev gawk bison
```

Install dependancies (on Debian Buster)

- the following has been tested under Debian Buster building on an amd64 machine

- ```
  $ git clone https://github.com/gaiusm/m2-cross
  $ mkdir build-m2-cross
  $ cd build-m2-cross
  $ ../m2-cross/configure
  $ make
  ```

- this will build and install the `gm2`, `gcc`, `g++` cross compilers into `$HOME/opt-gcc-12/cross/bin` (and `$HOME/opt-gcc-12/cross/{lib,share}`)
  - note currently this location is hardcoded
  ```

The bare metal software

- there are four objects which will be linked to produce the bare metal application
  - `armc-08-start.S` assembly language start up which assigns the stack and calls `armc-08-cstartup.c`
  - `armc-08-cstartup.c` initialise the BSS to zero and calls `main`
  - *application*`_m2.c` scaffold which calls each dependant module constructor in turn and deconstructor when it terminates
  - *application*`.mod` the main program

Main program: ledtest.mod

```
MODULE ledtest ;

(* Flash the onboard green LED of the RPI4 using Modula-2.
   The tiny onboard green LED is located next to the red power LED
   (also known as the ACT LED which flashes during bootup).
   Author:  Gaius Mulley   31/8/2021.  *)

FROM SYSTEM IMPORT ADDRESS ;

CONST
   GPIO_BASE    = 0FE200000H ;
   GPIO_MAX     = LED_GPCLR1 ;
   (* GPIO bitset offsets.  *)
   GPIO_GPFSEL  = 4 ;
   LED_GPSET1   = 8 ;
   LED_GPCLR1   = 11 ;
   (* BIT positions.  *)
   LED_GPFBIT   = 6 ;
   LED_GPIO_BIT = 10 ;
```

Main program: ledtest.mod

```
VAR
   gpio[GPIO_BASE] : ARRAY [0..GPIO_MAX] OF BITSET ;


PROCEDURE waste_time ;
CONST
   delay = 800000 ;
VAR
   i: CARDINAL ;
BEGIN
   FOR i := 0 TO delay DO
      ASM VOLATILE ("nop")
   END
END waste_time ;
```

Main program: `ledtest.mod`

```
BEGIN
   (* Set this bit to TRUE to enable the LED as an output.  *)
   INCL (gpio[GPIO_GPFSEL], LED_GPFBIT) ;
   LOOP
      (* set the bit (led on).  *)
      gpio[LED_GPSET1] := BITSET {LED_GPIO_BIT} ;
      waste_time ;
      (* clear the bit (led off).  *)
      gpio[LED_GPCLR1] := BITSET {LED_GPIO_BIT} ;
      waste_time
   END
END ledtest.
```

Compile and link the application

- continue in the command line terminal and type:

```
$ cd ../m2-cross/bare-metal-m2
$ ./buildm2 ledtest
```

- this produces a `kernel.img` file which can be placed into the Raspberry Pi-4 boot file `kernel8.img`
  - the Raspberry Pi-4 will boot and run this image

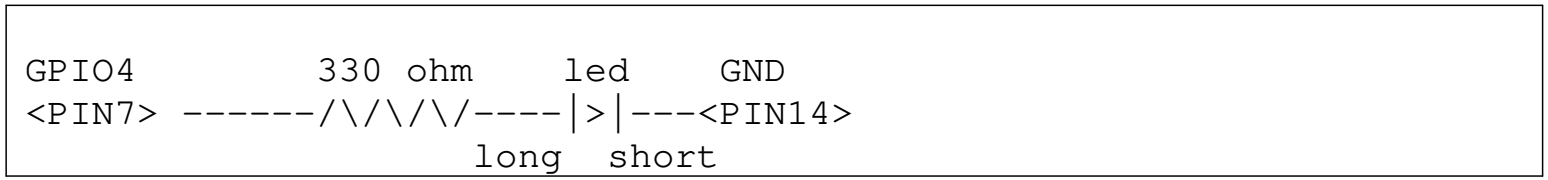
Making the Raspberry Pi-4 boot the kernel8.img file

- the placing the `kernel8.img` into the sdcard is one approach which will work - albeit a slow way to develop

- another (much quicker) method is to configure a tftpserver to serve the `kernel8.img` file and configure the Raspberry Pi-4 to boot using tftp

Turning on an external LED

- the code shown below will pulse `GPIO4` (pin7) lo / high
  - one simple use for the program is to pulse a LED
  - the trivial circuit requires a current limiting resistor (330 ohm) in series with the LED
  - `<PIN7>` connected to the registor which is connected to the long leg of the LED
  - the short leg of the LED is connected to GND `<PIN14>`



- normal caveats apply - be careful - you could damage your Pi-4 if you get this wrong

ledtest2 program

```
MODULE ledtest2 ;

(* Flash an offboard LED of the RPI4 using Modula-2
   You will need to install an LED and resister between PIN 7
   (GPIO4) and ground.
   Author:  Gaius Mulley   31/8/2021.  *)

FROM SYSTEM IMPORT ADDRESS ;

CONST
   GPIO_BASE    = 0FE200000H ;
   GPIO_MAX     = 32 ;
   GPIO_GPFSEL  = 4 ;
   LED_GPFBIT   = 6 ;
   LED_GPIO_BIT = 10 ;
   LED_GPCLR0   = 10 ;
   LED_GPSET0   = 7 ;
   LED_GPCLR1   = 11 ;
   LED_GPSET1   = 8 ;
```

ledtest2 program

```
VAR
   gpio[GPIO_BASE] : ARRAY [0..GPIO_MAX] OF BITSET ;

PROCEDURE inp_gpio (pin: CARDINAL) ;
VAR
   value   : BITSET ;
   position: CARDINAL ;
BEGIN
   position := (pin MOD 10) * 3 ;
   short_delay ;
   value := gpio[pin DIV 10] ;
   EXCL (value, position) ;
   EXCL (value, position + 1) ;
   EXCL (value, position + 2) ;
   short_delay ;
   gpio[pin DIV 10] := value
END inp_gpio ;
```

ledtest2 program

```
PROCEDURE out_gpio (pin: CARDINAL) ;
VAR
   value   : BITSET ;
   position: CARDINAL ;
BEGIN
   position := (pin MOD 10) * 3 ;
   short_delay ;
   value := gpio[pin DIV 10] ;
   INCL (value, position) ;
   EXCL (value, position + 1) ;
   EXCL (value, position + 2) ;
   short_delay ;
   gpio[pin DIV 10] := value
END out_gpio ;
```

ledtest2 program

```
PROCEDURE short_delay ;
CONST
   delay = 10000 ;
VAR
   i: CARDINAL ;
BEGIN
   FOR i := 0 TO delay DO
      ASM VOLATILE ("nop")
   END
END short_delay ;
```

ledtest2 program

```
PROCEDURE waste_time ;
CONST
   delay = 500000 ;
VAR
   i: CARDINAL ;
BEGIN
   FOR i := 0 TO delay DO
      ASM VOLATILE ("nop")
   END
END waste_time ;
```

ledtest2 program

```
CONST
   LED_EXTERNAL_BIT = 4 ;  (* pin 7  *)

BEGIN
   (* Set this bit to TRUE to enable the LED as an output.  *)
   inp_gpio (LED_EXTERNAL_BIT) ;
   out_gpio (LED_EXTERNAL_BIT) ;
   LOOP
      (* set the bit (led on).  *)
      gpio[LED_GPSET0] := BITSET {LED_EXTERNAL_BIT} ;
      waste_time ;
      (* clear the bit (led off).  *)
      gpio[LED_GPCLR0] := BITSET {LED_EXTERNAL_BIT} ;
      waste_time
   END
END ledtest2.
```

Building ledtest2

- the `kernel.img` containing the `ledtest2` binary can be built using the following command:

- ```
  $ ./buildm2 ledtest2
  ```
  ```