GCC ARM

Vast set of ARM processors can be easily programmed with usage of GCC ARM tool. This tutorial is intended for Cortex M4 with float coprocessor (armv7-m + f uarch)

1) install package, or compile from sources:

```
#sudo aptitude install gcc-arm-none-eabi
```

2) write some source:

```
#vim a.c
#define RAM_END 0x20003FFF
int main( void );
unsigned long* vector_table[] __attribute__((section(".vector_table"))) =
{       ( unsigned long* )RAM_END,      //initial SP
        ( unsigned long* )main          //Reset_Handler
};

int main()
{       while( 1 );
};
```

3) write processor-specific linker file:

```
#vim stm32_CM4F.ld
MEMORY                                                  /*stm32F373x8*/
{       FLASH       (rx)    :       ORIGIN = 0x08000000, LENGTH = 64K
        RAM         (rwx)   :       ORIGIN = 0x20000000, LENGTH = 16K
};

SECTIONS
{       .vector_table :
        {       *(.vector_table)
        } > FLASH
        .text   :
        {       *(.text)
        } > FLASH
        .data   :
        {       *(.data)
        } > RAM
};
```

4) write run commands BASH script:

```
#vim RUN_COMMANDS.sh && sudo chmod +x RUN_COMMANDS.sh
#!/bin/bash
GCC_GENERAL='-c'
GCC_OPT='-O1 -Wall'
GCC_ARCH='-mcpu=cortex-m4 -mthumb'
GCC_FLOAT='-mfloat-abi=hard -mfpu=fpv4-sp-d16'
GCC_DEBUG=''
L_FILE='stm32_CM4F.ld'
L_FLAGS='-nostartfiles --print-map'

FILENAME='a'
clear &&
arm-none-eabi-gcc $GCC_GENERAL $GCC_OPT $GCC_ARCH $GCC_FLOAT
$GCC_DEBUG $FILENAME.c -o $FILENAME.o &&
arm-none-eabi-ld $L_FLAGS -T$L_FILE $FILENAME.o -o $FILENAME.elf        &&
arm-none-eabi-size -tA --radix=16 $FILENAME.elf     &&
arm-none-eabi-objcopy -Oihex $FILENAME.elf $FILENAME.hex    &&
touch $FILENAME.elf $FILENAME.o && rm $FILENAME.elf $FILENAME.o
```