Appendix D

The GNU Assembler

The set of examples presented in this work are written for the ARM-MDK (Microcontroller Development Kit). Another compiler/assembler/link editor trio, however, is currently used to construct such projects. This is the compilation chain GNU[1]-GCC (GNU Compiler Connection), which as its name suggests comes from the world of free and cooperative software. This appendix is based specifically on the Sourcery G++ Lite tools for ARM EABI (Embedded-Application Binary Interface). This precision is important insofar as, although the tools are designed to be generic, they still have specificities and peculiarities that make them somewhat more difficult to understand and use. As an example to illustrate this, take the official documentation for the Sourcery G++ assembler. This documentation is 328 pages long, and 202 pages (Chapter 9) are dedicated to the characteristics unique to each processor. Despite the peculiarities, it is interesting to draw on the common points between the different GNU compilers/assemblers/link editors in order to be able to develop programs with different tools and/or processors.

The ARM-MDK and GNU-GCC projects are not cross-compatible, which might seem surprising since the processor and thus the instruction set are very similar. The two major differences are found in the syntax used for the directives (for the assembler) and in the way of describing the memory (for the link editor). We should note that the use of the GNU GCC tool is still possible in the Keil μVision environment, which is interesting as this environment is completely functional in its free trial version and so it is possible to use the μcontroller emulator that it offers.

---

1 GNU is a Unix-like computer operating system and corresponds to the recursive acronym “GNU is Not Unix”

In the first part of this appendix, we will list the main directives necessary to develop a project, for instance that which was presented in Chapter 3. In section D.2 we will give an example of a program written with these syntax rules that aims to make up for the absence of the MicroLib library. We will not develop the aspects tied to the linker, and especially the memory description file, here due to the complexity of their construction. The Sourcery G++ suite for ARM EABI is supplied with the typical files (*generic.ld* for example) from which it is possible to start.

D.1. GNU directive

In the GNU world, the set of instructions that we want to give to the compiler or assembly (directives) will be introduced by a keyword prefixed by a "." (dot). The vast majority of directives that we have explored (see Chapter 3) have their direct equivalents in this version of the assembler, but there are a certain number of additions, differences and subtleties that we must carefully understand when we switch from the *ARM* universe to the *GNU* universe.

D.1.1. *Generalities*

These are not strictly the directives mentioned but the following syntactic rules must be respected in order to produce code that is acceptable to the assembler:

– the commentaries are, as is usual in *C*, the set of characters between the delimiters '/*' and '*/'. The user has a second chance to insert a comment by inserting an '@'. The rest of the line (until *carriage return*) will then be ignored. This second syntax is inevitably not very readable;

– symbols are made up of several "letter" characters (upper or lower case), "number" characters (with the exception of the first character) and the two special characters '.' and '_'. Please note that the '*$*' character, which is allowed in other processors, cannot be used here because (like #) it indicates the start of an immediate value;

– a label is a symbol followed by a ':';

– there are no differences when giving the expression of a numerical value other than for expression in any numerical base. Only bases 10, 16 and binary are allowed. For binary numbers, the value must be prefixed with *0b*. For example, the instruction *MOV R0, #0b100* would load the value 4 in *R0*.

D.1.2. *Memory management*

There is not a single directive to create sections. There are three directives that allow us to carry out openings (closings happen automatically with the opening of a new section). These are identified by the type of section that they create:

– *.text* lets us open a section of code;

– *.data* lets us open a section of initialized data;

– *.bss* lets us open a section of data initially set to zero.

These directives do not support certain options, unlike the ARM directive *AREA*. However, it is possible to add other directives to the section opening suite to specify specific properties:

– *.align n* allows us to impose an alignment on the instruction or datum that follows this directive. The alignment will be to an address divisible by 2^n;

– *.ltorg* allows us to specify that the contents of the *Literal Pool* must be placed in this section (which must be of the *.text* type).

D.1.3. *Management of variables*

The different directives that follow allow us to make various reservations. Do not forget that the symbols that will correspond to these reservations must, as for the labels, be followed by '*:*':

– *.ascii "string..."* makes a reservation of as many bytes as there are characters in the chain following this directive. Note that a *.asciiz* version exists and is set at 0 at the end of the chain;

– *.byte exp1{,exp2}{,...}* reserves a byte initialized by *exp*. There are as many reservations as expressions;

– *.equ symbol, expression* acts as a *find (symbol)/replace (expression)* in the current module. This directive therefore allows us to define constants;

– *.fill repeat {,size}{,value}* allows us to create *repeat* copies of the value *value* coded on *size* bytes. If *value* is not defined, the value will be zero. If *size* is not specified, the default size will be one byte;

– *.hword exp1{,exp2}{,...}* is equivalent to *.byte* for 16-bit reservations (a half-word);

– *.space size {,fill}* reserves *size* bytes filled with *fill* (less than 255). The absence of *fill* leads to initialization at 0;

– *.word exp1 {,exp2}{,...}* is equivalent to *.byte* for 32-bit reservations.

D.1.4. *Conditional assembly*

The following directives allow us to carry out conditional assembly (see section 3.5.2):

– *.if expression* means that the lines that follow will be included in the section if *expression* is not false;

– *.endif* marks the end of the conditional part;

– *.else* allows us to include an alternative composed with *.if*.

D.1.5. *Miscellaneous*

There are other directives that can be necessary or useful:

– A module ends with *.end*. The assembler ignores the lines that follow this directive.

– The directive *.type* allows us to give additional information on a symbol. This information can be particularly useful with the use of a symbolic debugger. The most common use is *.type symbol,%function* to indicate that the label *symbol* corresponds to a procedure entry.

– There are no specific directives for indicating the importation of a symbol (see section 9.1.5). It is enough, in the module where the insertion of the directive *.global symbol* is declared, that *symbol* is shared with other symbols of the same name during link editing. This directive carries out the exportation of a symbol.

– *.weak name* marks the symbol *name* as *weak* in the symbol table. This means that another definition of the symbol will have a higher priority than this typed declaration.

– *.size name, expression* associates a size (*expression*) with a symbol (*name*). The dimension can be the result of a calculation, such as the difference between two symbols. This directive is typically used to calculate, for example, how much space a function takes up. For example, *.size Dim . – Deb_func* will assign the dimension corresponding to the difference between the current address (.) and the address of the symbol *Deb_func* to the symbol *Dim*.

D.2. Bootstrap program

The use of *GNU GCC* renders the use of *MicroLib* void. Compared to other projects developed in assembly language, the main function of this library that we must consider is the routine that allows us, following a *Reset* for example, to reinitialize the initialized variables (mainly in the *RW* or *.data* zones) as well as resetting the other zones to zero (the *ZI* or *.bss* zones).

The listing (see Example D.1) is therefore an example of a routine that carries out these initializations. The principle is quite simple, since it consists of making transfers from memory to memory (initialized variables) or resetting other zones (simple variables). What can be more obscure is the way in which it is possible to recover the addresses and sizes of these different zones. In our case, which incidentally is quite standard, the *Startup_constants* table that contains this information is made up from generic names created in the memory description files used by the linker. The names of these addresses are sufficiently significant to pass comment. The prefix *cs* means *CodeSourcery*.

EXAMPLE D.1.– *Variable initialization module (GNU environment)*

```
/*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Name : Startup.asm
Purpose : System stack initialization and initialized variables
Version : V1.1
Author : SDM (INSA de Toulouse)
/*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
              .text
              .align 2
              .global Startup
              .type Startup, %function
Startup :                                    ;Stack initialization
              LDR R0, Startup_constants+12
              MOV SP, R0
/*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Initialization of initialized variables
R0 : @Destination, R1 : @Source, R2 : size, R3 : counter, R4 : time
/*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
start_init_ram:
              LDR R0, Startup_constants+16
```

```
                LDR R1, Startup_constants+20
                LDR R2, Startup_constants+24
                MOV R3, #0
loop_init_ram :
                CMP R3, R2
                BCS start_zero_ram
                LDR R4,[R1,#0]
                STR R4,[R0,#0]
                ADD R3, #4
                ADD R1, #4
                ADD R1, #4
                ADD R0, #4
                B loop_init_ram

/*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Initialization at zero of uninitialized variables
R0 : @Destination, R2 : size, R3 : counter, R4 : time
/*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

start_zero_ram :
                LDR R0, Startup_constants+0
                LDR R2, Startup_constants+8
                MOV R3, #0
                MOV R4, #0
loop_zero_ram :
                CMP R3, R2
                BCS call_main
                STR R4,[R0,#0]
                ADD R3, #4
                ADD R0, #4
                B loop_zero_ram
/*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
End of initialization routine, call main
/*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

call_main : BL main
/*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

```
Constants used by Startup
Stored in .text section, after the routine
/*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
             .text
Startup_constants :
             .word __cs3_region_start_bss
             .word __cs3_region_zero_ram_loadaddr
             .word __cs3_region_zero_ram_size
             .word __cs3_stack
             .word __cs3_region_start_data
             .word __cs3_region_init_ram_loadaddr
             .word __cs3_region_init_ram_size
             .end
```

A second listing (see Example D.2) is an example of a bootstrap file. It corresponds and is comparable to that which we have already presented (see Appendix C). We have not described the whole module here. In fact, part of the interrupt vector table has been purposely excluded in order to make this listing lighter.

EXAMPLE D.2.– *Bootstrap file for a GNU assembler*

```
/*****************************************************************/
/* STM32F10x.s : Startup file for ST STM32F10x device series */
/*****************************************************************/
/* Version : CodeSourcery Sourcery G++ Lite (with CS3) */
/* Modified by SDM (call to Startup) */
/*****************************************************************/

/*****************************************************************/
/* Stack Configuration */
/*****************************************************************/
             .equ Stack_Size, 0x00000200
             .data
             .align 3
             .globl __cs3_stack_mem
             .globl __cs3_stack_size
```

```
_ _cs3_stack_mem :
              .if Stack_Size
              .space Stack_Size
              .endif
              .size __cs3_stack_mem, . - __cs3_stack_mem
              .set __cs3_stack_size, . - __cs3_stack_mem
/*****************************************************************/
/* Heap Configuration */
/*****************************************************************/

              .equ Heap_Size, 0x00001000

              .data
              .align 3
              .globl __cs3_heap_start
              .globl __cs3_heap_end
__cs3_heap_start :
              .if Heap_Size
              .space Heap_Size
              .endif
__cs3_heap_end :
/*****************************************************************/
/* Vector Table */
/*****************************************************************/
              .section ".cs3.interrupt_vector"
              .globl __cs3_interrupt_vector_cortex_m
              .type __cs3_interrupt_vector_cortex_m,
__cs3_interrupt_vector_cortex_m :
              .word __cs3_stack             /* Top of Stack */
              .word __cs3_reset             /* Reset Handler */
              .word NMI_Handler :           /* NMI Handler */
              .word HardFault_Handler       /* Hard Fault Handler */
              .word MemManage_Handler       /* MPU Fault Handler */
              .word BusFault_Handler        /* Bus Fault Handler */
              .word UsageFault_Handler      /* Usage Fault Handler */
              .word 0                       /* Reserved */
              .word 0                       /* Reserved */
```

```
                .word 0                         /* Reserved */
                .word 0                         /* Reserved */
                .word SVC_Handler               /* SVCall Handler */
                .word DebugMon_Handler          /*Debug Monitor Handler */
                .word 0                         /* Reserved */
                .word PendSV_Handler            /* PendSV Handler */
                .word SysTick_Handler           /* SysTick Handler */
/*******************************************************************/
/* External Interrupts */
/*******************************************************************/
                .word WWDG_IRQHandler  /* Window Watchdog */
                .word PVD_IRQHandler   /* PVD through EXTI Line detect */
                ...
                Incomplete Table....
.size __cs3_interrupt_vector_cortex_m, . - __cs3_interrupt_vector_cortex_m
/*******************************************************************/
/* Reset Handler */
/*******************************************************************/
                .text
                .globl __cs3_reset_cortex_m
                .type __cs3_reset_cortex_m, %function
__cs3_reset_cortex_m :
                LDR R0,=Startup
                BX R0

                .size __cs3_reset_cortex_m,.-__cs3_reset_cortex_m

                .text
/*******************************************************************/
/* Exception Handlers */
/*******************************************************************/
                .weak NMI_Handler
                .type NMI_Handler, %function
NMI_Handler :
                B .
                .size NMI_Handler, . - NMI_Handler
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*/
```

```
                .weak HardFault_Handler
                .type HardFault_Handler, %function
HardFault_Handler :
                B .
                .size HardFault_Handler, . - HardFault_Handler
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*/
                .weak MemManage_Handler
                .type MemManage_Handler, %function
MemManage_Handler :
                B .
                .type MemManage_Handler, . - MemManage_Handler
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*/
                .weak BusFault_Handler
                .type BusFault_Handler, %function
BusFault_Handler :
                B .
                .type BusFault_Handler, . - BusFault_Handler
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*/
                .weak UsageFault_Handler
                .type UsageFault_Handler, %function
UsageFault_Handler :
                B .
                .type UsageFault_Handler, . - UsageFault_Handler
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*/
                .weak SVC_Handler
                .type SVC_Handler, %function
SVC_Handler :
                B .
                .type SVC_Handler, . - SVC_Handler
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*/
                .weak DebugMon_Handler
                .type DebugMon_Handler, %function
DebugMon_Handler :
                B .
                .type DebugMon_Handler, . - DebugMon_Handler
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*/
                .weak PendSV_Handler
```

```
                .type PendSV_Handler, %function
PendSV_Handler :
                B .
                .type PendSV_Handler, . - PendSV_Handler
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*/
                .weak SysTick_Handler
                .type SysTick_Handler, %function
SysTick_Handler :
                B .
                .type SysTick_Handler, . - SysTick_Handler
/*******************************************************************/
/* IRQ Handlers */
/*******************************************************************/
                .global Default_Handler
                .type Default_Handler, %function
Default_Handler :
                B .
                .type Default_Handler, . - Default_Handler
```