

ELEC 2220 Computer Systems

Homework #5

Due: Wednesday, 6-3-2020

All links/documents discussed below, for the Microcontroller Development Kit for ARM (MDK-ARM), are provided on the course web page under “Class Resources”.

Read textbook Chapter 3 (*ARM Instruction Set Architecture*). Most of this information is also contained in the *Cortex-M4 Devices Generic User Guide* (in the **Books** pane of the Keil *uVision5* program, or via the link under “ARM Corporation” on the course web page). The relevant sections of the User Guide are 2.1 (Programmers model), 2.2 (Memory model) and 3 (The Cortex-M4 Instruction Set).

Download and install the Keil MDK-ARM package, following the procedure in the document *Installing MDK-ARM*. The “evaluation version” of MDK-ARM is free of charge, but limited to program sizes of 32K bytes (more than enough for our course projects.) The integrated development environment (IDE) tool in this package is *uVision5*.

You are to create and test a project containing a short assembly language program for the STM32F407VG microcontroller, using the *uVision5* simulator to run and debug the program, rather than downloading the program to your Discovery board. Follow steps 1-7 and 11-12 of the procedure described in the document *Creating Projects for STM32F4-Discovery with MDK-ARM*. Debugging procedures are described in the document *Debugging Projects for STM32F4-Discovery with MDK-ARM*.

- The project is to have one assembly language source file, containing the assembly language program listed on the next page.
- Create a debug initialization file (step 11 of the creating project procedure) to:
 - initialize the PC register to the starting address of the program (0x20000000),
 - set the PSR register to 0x01000000 (to set the T bit to 1 to select Thumb mode),
 - set data variables CC2 and VarA to 6 and 0, respectively.
- Build the project (which assembles the program). You should see “no errors” reported.
- Enter the debugger, display the data memory used by the program in the **Memory 1** window (starting at 0x20010000), and display variable *VarA* in the **Watch 1** window.
- Single-step through the program (execute one instruction at a time), observing the executed instructions in the Disassembly Window, and noting changes in the CPU registers, memory, and watch windows. (Changes are highlighted by the debugger as instructions execute.)
- After executing the last instruction of the test program, capture a “screen shot” of the Debug Window, showing the final state of the program. On this image, circle or highlight the following items in these debug windows, and then upload the image to Canvas.
 - Source Code Window – the last instruction executed
 - Disassembly Window – the last instruction executed
 - Registers Window – the final value of the PC and the last register modified by the program.
 - Memory 1 Window – the value written to memory by the program.
 - Watch 1 Window – the value of *VarA*
- Examine the assembler listing file (.lst) and map file (.map) to see how the program was assembled, and the locations of program code, variables, etc.

```

; Assembly language setup example for STM32F4-Discovery
; Simple program to be debugged in on-chip RAM
; Memory map should be set to on-chip RAM areas:
;   R/O memory (ROM1) at 0x20000000 (code area)
;   R/W memory (RAM1) at 0x20010000 (data area)

; Code section - to begin at 0x20000000
    AREA RESET, CODE
    THUMB
    ENTRY
Main  mov  r0, #100           ;set r0 = 100
      movt r0, #3           ;set top of r0 (upper 16 bits) = 3
      movs r1, #-20        ;set r1 = -20 and set flags
      add  r2, r0, r1       ;r2 = r0+r1
      adr  r4, CC1         ;address of CC1 to r4 from literal pool
      ldr  r5, [r4]        ;load value of CC using pointer in r4
      ldr  r6, =CC2        ;address of CC2 to r6 from literal pool
      ldr  r7, [r6]        ;load value of CC2 to r7
      subs r2, r2, r7      ;r2 = r2 - r7 and set flags
      str  r2, [r6, #4]    ;store r2 at VarA (4 bytes after CC2)
Here  b   Here            ;effectively halts the program

;Place a 32-bit constant in code memory area address CC1
CC1   dcd  5

;literal pool will be placed here by the assembler
;address of CC1 (literal =CC1)
;address of CC2 (literal =CC2)

; Data section - to begin at 0x20010000
; Initial values ignored for RAM (set with debug.ini file)
      EXPORT VarA         ; Allows viewing VarA in Watch window
      AREA Datal, DATA
CC2   dcd  0              ;32-bit word value 0 ignored for RAM
VarA  space 4            ;4 bytes/32-bit value (no initial value)

      END                ;end of assembly language source file

```