

Lab 3 – Hello, ARM!


Tom Clarke & Max Cattafi

(t.clarke@imperial.ac.uk, m.cattafi@imperial.ac.uk)

In this session we introduce ARM assembly programming and the Keil *μVision* IDE.

(For worked examples, pen and paper exercises and self-test questions, see the ‘Coursework’ page of the course, accessible from the link on “Introduction to Computer Architecture” on Blackboard and on <https://intranet.ee.ic.ac.uk/t.clarke/arch/contents.html>)

Getting started with Keil *μVision*

- Start Keil *μVision* 4 
- Project → new *μVision* project → select project directory and project file name → select target CPU (ARM → ARM7 – little endian).
- Create a new source file (File → new). Assembler files must have .s extension.
- Add the source file to a source group in the project (e.g. the default one).
- (Remember that files must always be added to the project source group in order to be included when the project application is built.)

- Write your program (an example is given below) and save the source file.
- Build the project.
- Switch to the debugging/running mode.
- Control the program execution (e.g. run step by step, one instruction at a time) and observe registers and memory (we'll deal more with memory in future sessions).

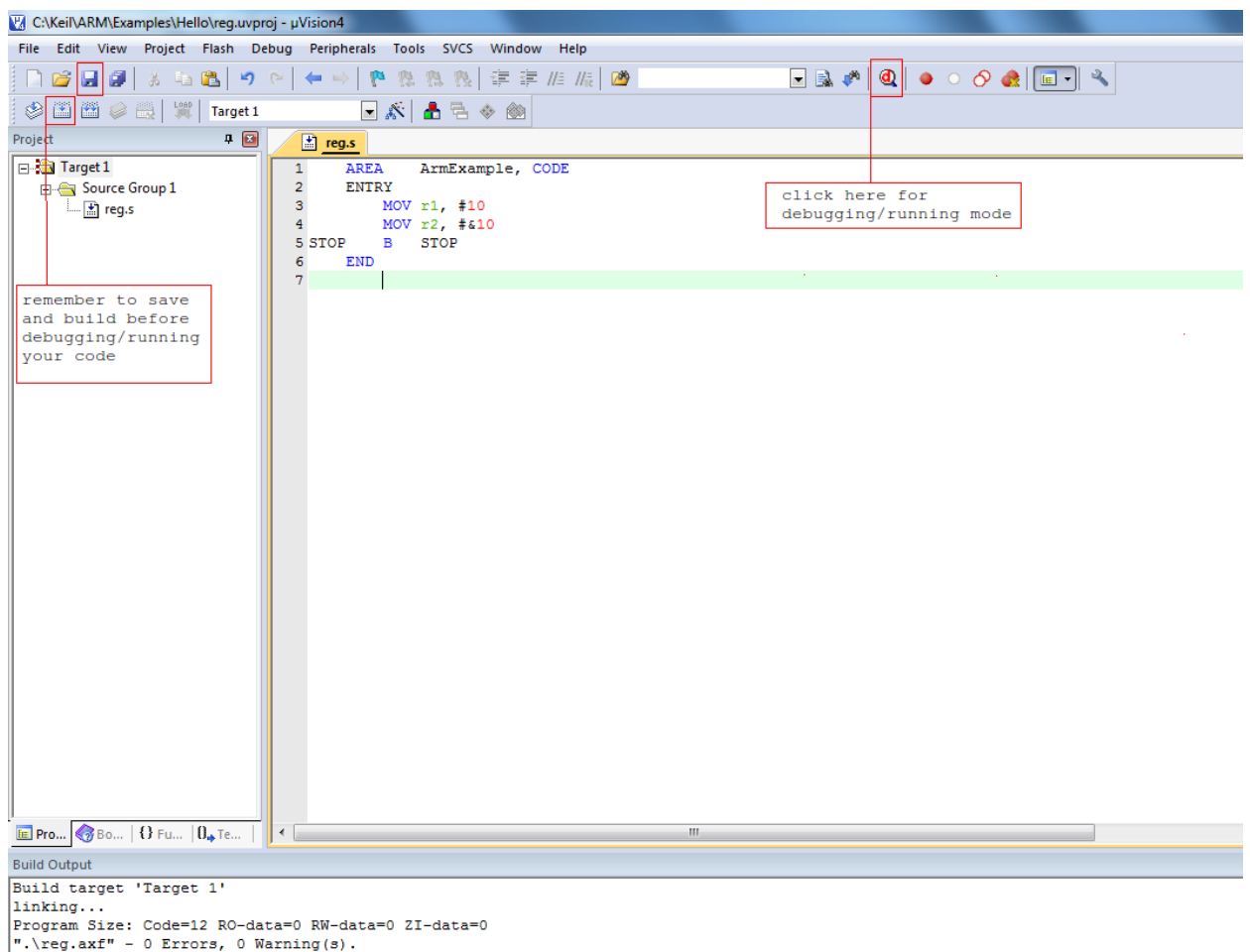


Figure 1: The initial view with the editor.

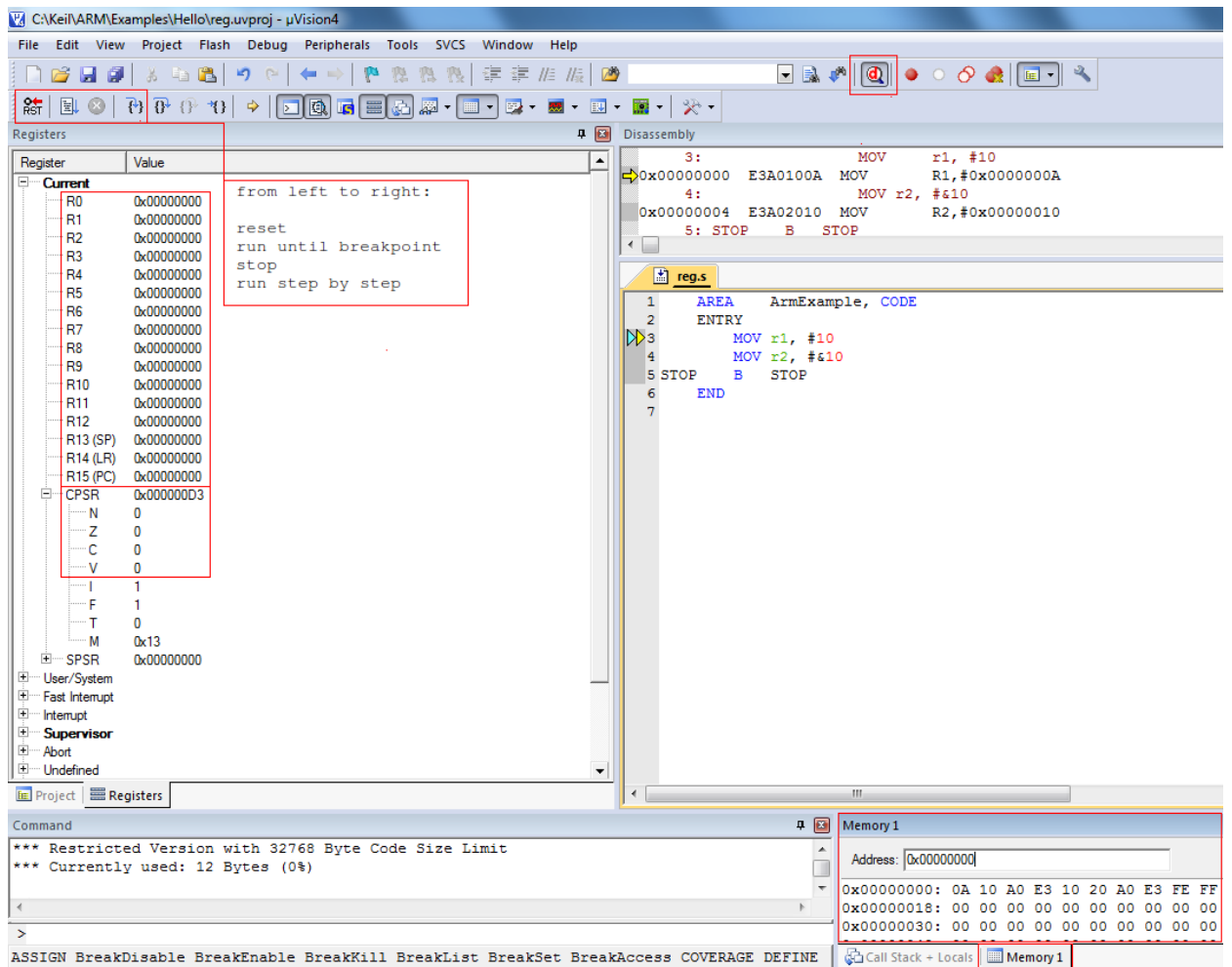


Figure 2: The debugging and running mode.

Our first ARM assembly program just assigns some values to some registers (type it in the editor: copy-paste of code from a pdf is not a good idea):

```
AREA          ArmExample, CODE

; code area declaration and name

; the indenting is important:
; include the tab/spaces at the beginning
; of the lines, or the project will not build!

; (semicolons mark the beginning of comments)

ENTRY
; program entry point

    MOV r1, #10 ; decimal value 10 in r1
    MOV r2, #&10 ; hexadecimal value 0x10 in r2

STOP    B      STOP

; STOP is just a label we branch to with instruction B
; this infinite loop is needed in order to stop the
; program counter here (otherwise it would go astray
; to undefined memory locations)

END
```

Troubleshooting:

- Remember to save and build the project every time you make changes to the code.
- You can make changes to the code while in the debugging/running mode but that still requires you to save and build the project again.
- Updated values in the visible registers can only be seen when the program is not running (i.e. when it's stopped or suspended). If you are not controlling the execution step by step you may need to click on the stop button in order to see the updated values (even if the program is just stuck in the final infinite loop).

Exercises

The “Minimal ARM Assembly Code Quick Introduction” slide on the lecture notes can be useful in order to solve the following exercises.

Two’s complement

Write an ARM assembly program which assigns a negative number to a register, observe how it is represented. Hexadecimal negative numbers are written like this: `# - &1`.

Counting iterations

Write an ARM assembly program which contains an infinite loop incrementing at each iteration the value of `r1`.

Ordered pair

Write an ARM assembly program which assigns a value to `r1` and a value to `r2`, then checks if `r1 < r2` and, if that is not the case, swaps the values. Observe how the NZCV flags change during the execution.

Absolute value

Write an ARM assembly program which assigns a value to `r1` and assigns its absolute value to `r2`. Observe how the NZCV flags change during the execution in particular with reference to the “CMP instruction & condition codes” slide on the lecture notes.

Integer division

(Integer) division can be thought as repeated subtraction. Write an ARM assembly program which assigns a value to `r1` and a value to `r2`. The program then computes the integer division of `r1` by `r2` and stores the quotient in `r3` and the remainder in `r4`.