USING THE KEIL ASSEMBLER, SIMULATOR & DEBUGGER

CREATE A UVISION PROJECT:

- Start **Keil uVision4**
- **Project->new uvision project-> select project directory and project file name-> select target CPU - for assembly language testing use "ARM -> ARM7 - little endian"**.
- Note that you can change target & many compiler settings later by right-click properties on the target: however for simply assembly language testing you should not change defaults.
- Add existing source files to a source group in the project (source group 1 by default). Assembler files must have .s extension. You can create source files in an external editor or from the **File->new dialog**. Remember files must be added to the project source group after they have been created.
- For simple use all you need is one project, with one .s source file.

BUILD THE PROJECT:

- **Project -> build target**
- You can view the listing of generated machine code using **file->open** (select listing file extensions). The listing shows the memory address of each line of assembled code and what value (machine instruction or data) is generated.
- Should the system ever get confused "rebuild all" can be helpful, it will remake the target code from scratch, whereas "build" will only remake parts whose source files have changed.

SIMULATE THE PROJECT

- **Debug->start/stop debug session-> "Eval mode code limit 32K" OK**
- The simulation **source code** window shows a grey line highlighting the next assembly instruction to be executed.
- Simultaneously the **dissassembly** window shows the contents of memory with a yellow line through the next machine word to be executed.
- The **registers** window (LHS) shows machine registers (under **current** heading) and their values in hex.
- The **listing** window shows (after the code listing) the constant values of all assembly symbols in hex.
- Double-click on the CPSR register to see the individual values of the condition codes.
- **Debug-> reset CPU** will restart the simulation at the first instruction (address 0).

If you do not put at the end of your assembly code an "infinite loop" such as:

```
SLOOP   B         SLOOP
```

the simulation will continue executing random memory as instructions after your last instruction. This can have unexpected consequences. Single-step you will see this. Running continuously the results can be surprising.

You can make execution stop at a specific point in your program (at which time you can examine registers etc) by setting a breakpoint.

- Click on instruction in source file you wish to break before
- **Debug->insert/remove a breakpoint**

breakpoints will be highlighted red in the source file windows LHS.

MEMORY WINDOW

The memory window will display memory words in the selected radix (right-click on window to see options). It will allow you to change memory location values by double-clicking on a value.

The contiguous set of locations displayed is set by a number typed in the address box. This will accept hex & decimal constants & expressions, e.g. 123+ 0xa0, and register names, e.g. R3. Registers will evaluate to their current value. Thus if R10 is currently 7, and R11 currently 19, typing R11+R10+10 in the address box will view memory from address 36 (0x24) onwards.

Assembler symbols cannot be used in the address box.

CODE TO TEST

File checksum.zip on the web site (software tab) contains an assembler file with the checksum example from the lectures. Run this after adding to a Keil project created as above. In the debugger view the memory locations that are added in the memory window, & check that it does what you expect. (The hex value of the memory address of the data to be added can be found from the value of symbol DATA in the listing). Note that as each location is read by the program it will turn red in the memory window.