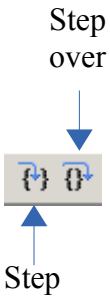
DT228-1: Lab 9. Subroutines in assembler.

Key topics: subroutines, the stack, parameter passing

Introduction

The exercises in this lab demonstrate the use of subroutines in assembler. The exercise includes assembler versions of commonly used functions such as **strlen** and **strcpy** and requires you to implement a variant of **strcpy** : the much safer **strncpy**.

Tasks.



Download and extract the support files for this lab. Step through the code using the "Step" debug button (F11) rather than the "Step over" button. While you are doing this, note behaviour of the stack pointer. After reset, the stack pointer is set to the end of RAM (0x20001000). The contents of the stack pointer will vary during subroutine calls.
In the **strcpy** function, examine the contents of memory while it is executing. Watch the way memory is gradually filled with the contents of the source string.
Items of interest along the way:
BL *target* is equivalent to "call subroutine"
pop {PC} is equivalent to "return from subroutine"
registers must be popped in reverse order – why?
push and pop can work on multiple registers - its faster and safer (why?)

Finally, copy and paste the **strcpy** function, changing all labels within it from "strcpy.." to "str**n**cpy..". Modify the code so that it implements the function int strncpy(char *dst, char *src, int len). This function is safer because it limits the number of bytes copied to "len" in the event that the null terminator is missing.