Tutorial #2
Hilary Term Weeks 4 and 5
LDM, STM, Stacks and Subroutines

Work in groups of 6 students using one of the whiteboards.
Complete the exercises in your own time if necessary.
Revise the exercises to prepare for the next lab.

1 Stacks

1.1 Stack Operations with LDR/STR

(a) Provide an ARM Assembly Language program that will push the contents of `R4`, `R5` and `R6` on to the system stack. You may not use the `LDM` or `STM` instructions.

(b) Illustrate the state of the stack after each push operation above.

(c) Provide an ARM Assembly Language program that will pop the three topmost words off the top of the stack, restoring their values into the registers from which they were originally stored in part (a) above. You may not use the `LDM` or `STM` instructions.

(d) Illustrate the state of the stack after each push operation above.

1.2 Stack Operations with LDM/STM

(a) Provide an ARM Assembly Language program that will push the contents of `R4`, `R5` and `R6` on to the system stack. You must use the `LDM` and `STM` instructions.

(b) Provide an ARM Assembly Language program that will pop the three topmost words off the top of the stack, restoring their values into the registers from which they were originally stored in part (a) above. You must use the `LDM` or `STM` instructions.

2 Understanding LDM and STM

Provide diagrams to illustrate the state of the stack (stack pointer and contents) after executing each of the following instructions and list the contents of any registeres modified by the operation. Begin by assuming some initial values stored in each register.

```
1        STMFD   SP!, {R4, R6, R8-R11}
2        LDMFD   SP,  {R10, R11}          ; Note - no ! in this instruction
3        LDMFD   SP!, {R8, R10, R11, R4, R6}
```

3 Other uses of LDM/STM

Write an ARM Assembly Language program to copy 512 bytes of memory from the address in R0 to the address in R1. Take advantage of the LDM and STM instructions to load and store multiple words using a single instruction.

Calculate the number of memory accesses required by your program and compare it with the number of memory accesses required if using only LDR an STR instructions.

4 Subroutines

4.1 Subroutine Interfaces

(a) For each of the following Java/C-like method declarations, design an appropriate ARM Assembly Language interface for a corresponding assembly language subroutine. The interface must include a specification of how each parameter is passed into the subroutine and how any return values are passed back to the calling program.

(i) `void zeroMemory(unsigned int startAddress, unsigned int length)`
(zero a range of addresses in memory)

(ii) `int factorial(unsigned int x)`

(iii) `int power(int x, unsigned int y)`

(iv) `int quadratic(int a, int b, int c, int x)`
(evaluate a quadratic function)

(b) For each of the subroutines listed above, show how you would invoke (call) the subroutine, assuming the variables to be passed as parameters are initially stored in registers other than R0–R3.

4.2 Subroutine Implementations

Implement each of the subroutines listed in Section 4.1, taking care to hide any unintended side-effects from a calling program using LDM and STM instructions to save and restore registers on the system stack. Your subroutines should adhere to the interfaces that you defined above. Try to use registers R0–R3 to pass parameters and R4–R12 to store variables that are local to the subroutine.

Complete the exercises in your own time if necessary!

© UNIVERSITY OF DUBLIN 2019