Practical 5

Use the attached template. It has all of the critical labels in it.

Part 1: (2)
Use the stack to calculate the first 47 Fibonacci numbers.
This should be implemented as follows:

- push two 1's onto the stack
- in a loop 45 times:
    - load the last two elements of the stack into registers. This can either be done by popping or peeking.
    - add the two numbers
    - push the result to the stack

Attached is a file showing the expected values which can be used for verification.
The first 47 Fibonacci numbers should all be present on the stack after this.

Part 2: (2)
The automaker will place a word at the address 0x2000 0000.
You should read the data at this address, treat the data as an address and store the highest value Fibonacci number at that address.
In other words: store the last element of the stack at the address pointed to by the data in 0x2000 0000.

Part 3: (1)
In the event that the address located in 0x2000 0000 turns out to be an invalid address, the CPU should catch the exception by jumping to an infinite loop with the label:
`HardFault_Handler`
The handler should display 0xAA on the LEDs to indicate to the user that a fault has occurred.

Part 3: (2)
On completion of part 2:
Cycle between the following patterns, with a 0.5 second delay between each pattern.
0x00 ; 0x81 ; 0xC3 ; 0xE7 ; 0xFF ; 0x7E ; 0x3C ; 0x18
The delay should be implemented via a single block of code labeled: `delay_routine` which should be branched to and returned from using the link register method.

Bonus: (1):
Cause the delay time to be dependant on the output voltage of POT1. When the pot is turned fully clockwise, the delay should be 0.5 s. When fully counterclockwise, the delay should be pretty much 0. Linear in between.

Marking:
Marked out of: 7 Available marks: 8