Extra practice questions

1) Assume the following values for registers R0 to R3:

- R0: 0xA0B0C0D0
- R1: 0x10203040
- R2: 0x00001000
- R3: 0x00001002

The program executes the following two instructions:

```
STR R0,[R2]
STRH R1,[R3]
```

Complete the following table determining the content of main memory.

Memory Location	**Value (byte in hexadecimal) if machine is big endian**	**Value (byte in hexadecimal) if machine is little endian**
0x1000		
0x1001		
0x1002		
0x1003		

2) The values in registers R1, R2, R3, R13 are initially as shown:

R1	0xAAAAAAAA
R2	0xBBBBBBBB
R3	0xCCCCCCCC
R13	0x00001000

Then the following code is executed:

```
    STR     R1,[R13,#-4]!
    PUSH    {R2,R3}
```

Fill in the contents of the stack as they would appear after executing the above instructions.

00000FF0	
00000FF4	
00000FF8	
00000FFC	
00001000	

Fill in the missing offset in the instruction below so that it loads the value 0xAAAAAAAA from the stack (as pictured above) into R4.

```
    LDR     R4,[R13,#____]
```

3) Write a code segment that will output the least-significant byte in R1 to a display using hexadecimal format. The display’s 8-bit data register is at the location labeled `DISP_DATA` (you can assume that the display is ready for output). The following table is defined for you:

```
      DIGITS    DCB  ‘0’,’1’,’2’,’3’,’4’,’5’,’6’,’7’
                DCB  ‘8’,’9’,’A’,’B’,’C’,’D’,’E’,’F’
```

4) Convert the following C code into equivalent ARM assembly code. X is in R0, Y in in R1. Use the smallest number of instructions that you can.

```
      if(X != 0) {
            Y = Y - 1;
            if(Y < 0) Y = Y + 10;
      }
```

5) Calculate the exclusive-OR of the 4 bytes in register R1, putting the result in R0. (Note that exclusive-OR is how checksums are often calculated on data before sending to check that it hasn’t been corrupted in transit.)

6) Do the following: if $5 <= R1 <= 10$, R0 = 1, else R0 = 0.

7) Rotate array A of length L right by N elements. For example if N =2 then

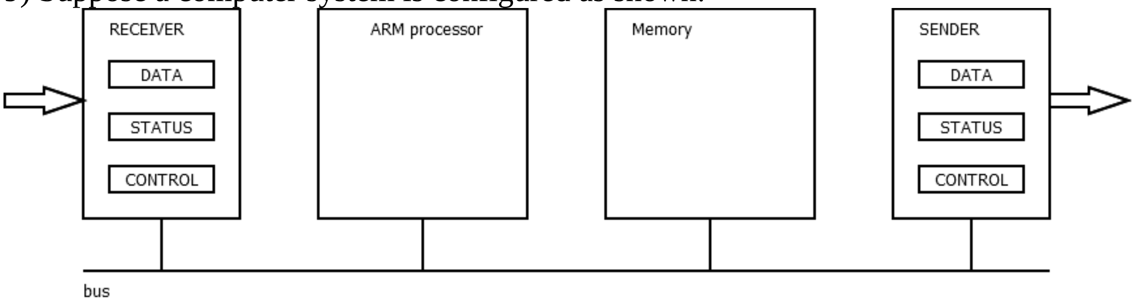
1	2	3	4	5

becomes

4	5	1	2	3

8) Clear bits 31 and 0 of R1.

9) Suppose a computer system is configured as shown:



It has two I/O devices, one that receives data from an external device, and another that sends data to a different external device. The I/O device registers are all 32 bits wide and work as follows:

	Register	Address	Function
RECEIVER	DATA	0xA0000000	contains received data
	STATUS	0xA0000004	is 1 if data is available, 0 otherwise
	CONTROL	0xA0000008	bit$_1$ is the receiver interrupt enable
SENDER	DATA	0xF0000000	accepts data to send
	STATUS	0xF0000004	is 1 if ready for data to send, 0 otherwise
	CONTROL	0xF0000008	bit$_0$ is the transmitter interrupt enable

Complete the following ARM assembly code as indicated so that it that reads data from RECEIVER and writes it to SENDER. Use as few instructions as possible.

	LDR R0,=0xA0000000	
	LDR R1,=0xF0000000	
POLL_R		;poll RECEIVER until it has data
READ_R		;read received data into R3
POLL_S		;poll SENDER until it is ready for data
WRITE_S		;give SENDER data in R3 to send
	B POLL_R	