Homework

Data Processing Instructions – Part 2

1. Determine the values stored in *r0* and *r3* after the following instruction executes, assuming that *r0* and *r3* contain `0x08116033` and `0xCBA5043E` initially.

```
LSL r3, r0, #17
```

2. If *r5* contains `0x81A82FAA`, what value is stored in register *r5* after the following instruction executes? How does the result differ if `ASR` is used?

```
LSR r5, r5, #4
```

3. Write a single ARM assembly language instruction to clear the upper byte of register *r8*.

4. What value is stored in registers *r0* and *r4* after the instruction shown below executes if *r0* and *r4* initially contain `0x2A0F6ACE` and `0xAC036D8F` respectively?

```
BFI r4, r0, #6, #14
```

5. Determine the value stored in registers r1, r2, and r7 after the following instruction executes. Initially, registers *r1*, *r2*, and *r7* contain `0x0E02E414`, `0x74E3B105`, and `0xBADD1FB7` respectively.

```
BIC r2, r7, r1
```

6. Write a sequence of ARM assembly language instructions to copy the most significant byte of register *r3* into the most significant byte of register *r1* without changing the contents of *r3* or the lower 24 bits of *r1*.

7. Write an ARM assembly language instruction to clear bit 23 of register *r0*.

8. When the value `0xABE40E32` is stored in *r0*, how do the values stored in r4 differ when each of the following instructions shown below are executed?

```
UBFX r4, r0, #23, #7
SBFX r4, r0, #23, #7
```

9. Indicate the values in registers *r0*, *r1*, and *r3* after the division instruction shown below executes. How is the end result different if the instruction were SDIV instead of UDIV? You may assume that, prior to execution, the registers *r0*, *r1*, and *r3* contain `0x475B91AC`, `0xC0A75403`, and `0x00000104` respectively.

```
UDIV r0, r1, r3
```

10. What values are stored in in registers *r2*, *r3*, *r6*, and *r7* after the multiplication instruction shown below executes? How would the result differ if the instruction were SMULL instead of UMULL? Prior to execution, the registers *r2*, *r3*, *r6*, and *r7* contain `0x00080030`, `0xFFFF5018`, `0x0EA75403`, and `0x00000104` respectively.

```
UMULL r6,r7,r2,r3
```

11. Of the three multiplication instructions in the ARM Cortex-M4 instruction set (MUL, UMULL, and SMULL), which should be used to multiply the signed values `0xFFFFF803` and `0x0001A629`?

12. Write a single ARM assembly language instruction equivalent to the instruction sequence shown below.

```
MOV r12, r12, LSR #3
EOR r12, r3, r12
```

13. What values are stored in *r0*, *r1*, *r4*, and *r6* after the following instructions have executed? Assume that *r0*, *r1*, *r4*, and *r6* initially contain `0x00000027`, `0x00000006`, `0x00000010`, and `0xFFFFFFFF` respectively.

```
ADD r0, r0, r1
SUB r0, r0, #31
ADD r1, r4, r6
```

14. What values are stored in *r0* and *r9* after the following instructions have executed? Assume that *r0* and *r9* initially contain `0xF1E92615` and `0xAC019356` respectively.

```
MVN r0, #18
AND r0, r9, r0
```

15. Why does the ARM assembly language instruction `MOV r0, r0, LSR #4` NOT perform integer division of *r0* by 16 when *r0* initially contains –1,424? How should this instruction be modified to perform the operation described above?

16. Write a single ARM assembly language instruction which will multiply an integer stored in *r3* by 33, placing the product in *r4*, without using an explicit multiplication instruction (MLA, MUL, SMLAL, or SMULL).

17. Write a single ARM assembly language instruction equivalent to the following instruction sequence.

```
MOV  r8, #37
SUB  r8, r8, r2
```

18. Write a single line of high-level language pseudocode that describes the following ARM assembly language instruction sequence in terms of *r0*, *r3*, *r5*, and *r8*.

```
SUB  r0, r3, r5
ADD  r0, r0, r0, LSL #1
ADD  r0, r8, r0
```

19. Write a single line of ARM assembly code which uses the bit clear (BIC) instruction to clear bits 5 and 8 in register *r10*.

20. Write an ARM assembly language instruction which rotates the data stored in *r0* left 19 positions.

21. Write a short program to determine how many bit positions in which *r0* and *r8* differ.

22. Write a single line of ARM assembly code to set bits 7, 10, and 12 of register *r9*.

23. Write a single line of ARM assembly code to invert the lower six bits of register *r1*.

24. What are the contents of register *r9* after the following instructions have executed? Assume *r1* and *r6* contain `0x08CB85A1` and `0x28EC75A9` respectively.

```
EOR r9, r1, r6
```

25. Replace the following ARM assembly language instructions with a single instruction which accomplishes the same thing.

```
MVN r4, r4
AND r3, r3, r4
MVN r4, r4
```

26. Write a short program to implement the expression shown below. You may consider r1, r2, and r3 as scratch registers, meaning their values can be changed by your code.

```
r0:=(r3 + 29)/8 + 17 - r2 * r1
```

27. Using less than five ARM assembly language instructions, calculate *r4* % *r0* where % is the modulus operator. Assume that the operands are unsigned. Place the result in *r0*.