Review the lab sheet first, then respond to the problems below.

Question 1

Consider the following MIPS assembler program being run on the MIPS processor described in the lab sheet:

```
1 addi $t0 $0 123  # line 0, fetch at cycle 0
2 addi $t1 $0 456
3 mult $t0 $t1
4 mflo $t1
5 add $t1 $t1 $t0
6 addi $t1 $t1 -56
7 addi $t2 $0 0x0f0f
8 lui $t2 0x0f0f
9 and $t1 $t1 $t2
```

Match the following questions with their correct answer…

Questions:

- In cycle 7, what instruction is using the `F` pipeline stage?
- In cycle 7, what instruction is using the `EX` pipeline stage?
- In cycle 7, what instruction is using the `WB` pipeline stage?
- In cycle 5, what instruction is using the `F` pipeline stage?
- In cycle 5, what instruction is using the `EX` pipeline stage?
- In cycle 5, what instruction is using the `WB` pipeline stage?

Answers:

- `lui $t2 0x0f0f`
- `addi $t0 $0 123`
- `add $t1 $t1 $t0`
- `subi $t1 $t1 56`
- `mflo $t1`
- `mult $t0 $t1`
- `addi $t1 $t1 -56`
- `addi $t2 $0 0x0f0f`
- `and $t1 $t1 $t2`
- `addi $t1 $0 456`
- `and $t1 $sp $fp`

Question 2

Consider the following MIPS assembler program being executed on the CPU described in the lab sheet…

```
1 and $t2 $t2 $t2
2 and $0 $t2 $0
3 xor $t2 $t2 $0
4 addi $t2 $t2 0
5 xor $0 $t2 $t2
6 sll $0 $0 0
7 xor $0 $1 $3
8 xor $0 $2 $3
```

What is the purpose of this program?

- (A) This program performs RSA encryption on the value in register 2.
- (B) This program clears the value of register 2.
- (C) This program performs a parity check between the values of register 2 and 3.
- (D) This program does nothing.
- (E) This program is syntactically incorrect and will not compile.
- (F) This program applies a ROT13 cipher to the register 2.

Question 3

Would it be possible to implement a program that computes each value in the Fibonacci sequence and outputs them in order via GPIO (ignoring any considerations about overflow).

- (A) This is NOT possible, because there are no conditional branch instructions.
- (B) This is NOT possible, because there is no jump instruction, and therefore it is not possible to implement a loop.
- (C) This is NOT possible because there is no floating point unit (FPU).
- (D) This is NOT possible because there is not enough instruction memory.
- (E) It IS possible to implement such a program because the PC will overflow back to 0.
- (F) It is NOT possible, because the largest possible unsigned immediate value is $2^16 - 1$.

- (G) It IS possible to implement such a program if the user enters the first value via the GPIO input.
- (H) It IS possible to implement such a program because conditional branch can be simulated with `ori`.

Question 4

Consider the following MIPS program. Imagine this program was executed on the CPU described in the lab sheet, however the write bypassing in the register was **not** working (i.e. the register file read is implemented as `assign readdata1 = mem[readaddr1]`)

```
1 li $t0 0
2 li $t1 3
3 li $t2 4
4 li $t3 1
5 add $t3 $t1 $t2
6 add $t0 $t3 $t2
7 add $t0 $t0 $t1
```

What will be the value of register `$t0` then the program finishes execution? All values are given in decimal.

- (A) 3
- (B) 7
- (C) 8
- (D) 13
- (E) 14
- (F) 17

Question 5

Why is it a good idea (or not) to use a dedicated register for `gpio_write`?

- (A) It is redundant to have a dedicated register because the contents will be a duplicate of the `hi` register.

- (B) It is a good idea because there are only two read ports on the register file, and the read address may change after the `srl` instruction completes.
- (C) It is a good idea because the value stored in the relevant address in the register file may change after the `srl` instruction completes.
- (D) It is a bad idea, because it is not necessary to have a `gpio_write` register because it would be more efficient to use a latch.
- (E) It is a bad idea, because the GPIO output signal must cross clock domains; rather than a register of type `logic`, `gpio_write` should be declared as an `altera_clock_domain_adapter` .
- (F) It is a good idea, because it will use less on-chip memory if the register is declared within the `cpu` module rather than the module which instantiates it.
- (G) It is a bad idea because the `hi` register could be used; a GPIO output update cannot happen on the same clock cycle as a `mult` instruction.

