Pre-Lab

1. What are Mnemonics in Assembly language?
Mnemonics is the name given to the opcode assembly language. Mnemonics end with a suffix of ".b" or ".w" to differentiate between operands that are byte and word sized. No suffix is by default equivalent to ".w"

2. What is the assembly instruction format for MSP430?
There are two components to the assembly instruction format:

1. The Mnemonics: Discussed in (Q1)
2. The operands: usually the "Source" and "Destination" of an instruction. An instruction can have two, one or no operands, depending on the needs and purpose of the instruction

3. What is the purpose of directives in assembly language? Are they converted to machine code?
The purpose of directives is to organise the code and create meaningful names for otherwise not so obviously understood code. They are not converted to machine language. They are independent of the assembler and rather belong to the cpu

4. What is the benefit of having labels in assembly language?
Labels are useful for replacing memory addresses and constants that are difficult to make sense of without previous knowledge of their meaning. They make your code more readable. For instance, they could take the value of the reset vector or interrupt vector, allowing you to use the label in your code rather than the actual value of these vectors, making your code more readable.

5. What is the difference between core and emulated instructions?
Core instructions are hardwired commands that have machine-specific OpCode in machine language syntax.
Emulated instructions make use of labels and are not hard-wired, making them easier to read, make sense of, remember and use.

6. The most significant byte of a register is not available in byte instructions. How can we access them if needs be?
to access or change the most significant byte in the register, we may use "swpb Rn" before or after the instruction.

7. What is the main difference between Absolute Source File and Relocatable Source File?

In absolute source file, the program location counter (PLC) used by the linker to load each instruction and the data at the appropriate address, is controlled with ORG directive.
In relocatable codes, we use segment directives to define or work memory segments, whose starting address is specified by the linker.

In-Lab