1. What two things does it do and why?

    - It loads a label “Stack” into the register 15 which stores the pointer to the program stack. This is to declare the stack. Then, it use .word directive to initialize space for the stack
    - It then call the main function to execute.

2. After the call instruction, there is a .literal 0 directive. What is the purpose of this directive? I.e., why is that directive there?

    The .literal directive encodes an integer at the present location in the program, and it encodes a nil-terminated string followed by a 16-bit value representing the decimal value 0. It’s used to help initializing the stack (set value to 0), because it’s followed by .word 512 which initialize the stack space.

3. What (specifically) would happen if the program defined in main pushed more than 512 items on the stack?

    Stack overflow. The program will keep pushing the item on the memory outside the stack, even those memory might be used by other data.

4. What is the value of the PC register after the first loadi instruction?

    Loadi is an extended instruction, so the pc incremented by 4. Pc=0004

5. Why is it not possible for code outside of xrt0 to determine how many items there are on the stack? What would you change in this module to allow this information to be determined by outside code?

    The directive .words sets aside a specified number of words of memory at the present location in the program. It allocates the number of words of memory at the present point in the program, and the space is not initialized to any specific value. So, this value is not glob and can’t be seen by other files. It doesn’t have a label, and it’s not glob. If we want to make it public, we can do it by adding a label to it and make the label glob, such as (.glob StackSize   StackSize: .words 512).