Goal:

The purpose of this project is to write an assembly language program that includes a procedure call. If you'd like to refresh your memory on how procedures are implemented in the MIPS assembly language, review section 2.8 in the Zybook. Section 2.8 discusses how to pass arguments, how to jump to a procedure, and how to return from a procedure.

Requirements:

This project includes two pieces: the main program and at least one procedure. The overall function of the program is to reverse the digits in three 32-bit hex numbers located in memory.

Main Program:

The main program is responsible for passing two arguments to the procedure: the address of the word to be reversed and an address at which to store the result. For example, if the word 0x12345678 is stored at address 0x1000400, the main program could pass the two addresses 0x1000400 (the address of the word to be reversed) and 0x10000500 (the address where the result should be stored). We should then expect the procedure to store the 32-bit word 0x87654321 in memory location 0x10000500.

The main program is also responsible for passing control or "jumping" to the procedure.

Procedure:

The procedure is responsible for accessing the source address passed from the main program, performing the hex digit swap, and then storing the result in the destination address passed from the main program.

The procedure is also responsible for returning from the procedure, in other words "jumping back" to the calling program.

Framework

A framework for this project is not provided. You may find the statements in the Project 1 framework helpful.

For a test case, use the addresses and data shown in the following table:

Source Address	Data	Destination Address
0x10000400	0x12345678	0x1000500
0x10000424	0xFEDC0123	0x1000504
0x10000438	0xF4C210B5	0x1000508

You will need to provide the assembler directives (.data and .word) to place the data in the correct source addresses.

Deliverables

1) Your MIPS program (.s file) named LastnameF.s (your lastname followed by your first initial).
2) A screenshot of the qtSPIM simulator showing the registers immediately after the jump to the procedure. The registers shown should show both arguments passed to the procedure and the return address.
3) A screenshot of the qtSPIM simulator once the three swaps are complete. The “data” window should show both the starting data and the results.