Single Cycle MIPS

Implemented an instruction-level simulator for a single cycle MIPS processor in C++. The simulator supports a subset of the MIPS instruction set and can model the execution of each instruction.

The MIPS program is provided to the simulator as a text file "imem.txt" file which is used to initialize the Instruction Memory. Each line of the file corresponds to a Byte stored in the Instruction Memory in binary format, with the first line at address 0, the next line at address 1 and so on. Four contiguous lines correspond to a whole instruction. Note that the words stored in memory are in "Big-Endian" format, meaning that the most significant byte is stored first.

The "halt" instruction, a 32'b1 (0xFFFFFFFF) is the last instruction in every "imem.txt" file. As the name suggests, when this instruction is fetched, the simulation is terminated.

The Data Memory is initialized using the "dmem.txt" file. The format of the stored words is the same as the Instruction Memory. As with the instruction memory, the data memory addresses also begin at 0 and increment by one in each line.

The instructions that the simulator supports and their encodings are shown in Table 1. Note that all instructions, except for "halt", exist in the MIPS ISA. The MIPS Green Sheet from HW1 defines the semantics of each instruction.

Name	Format Type	Opcode (Hex)	Func (Hex)
addu	R-Type	00	21
subu	R-Type	00	23
addiu	I-Type	09	
and	R-Type	00	24
or	R-Type	00	25
nor	R-Type	00	27
beq	I-Type	04	
j	J-Type	02	
lw	I-Type	23	
sw	I-Type	2B	
halt	J-Type	3F	

Table 1. Instruction encodings for a reduced MIPS ISA

We have defined four C++ classes that each implement one of the four major blocks in a single cycle MIPS, namely RF (to implement the register file), ALU (to implement the ALU), INSMem (to implement instruction memory), and DataMem (to implement data memory).

1. RF class: contains 32 32-bit registers defined as a private member.
2. ALU class: implements the ALU.
3. INSMem class: a Byte addressable memory that contains instructions. The constructor InsMem() initializes the contents of instruction memory from the file imem.txt. ReadMemory() provides read access to instruction memory. An access to the instruction memory class returns 4 bytes of data; i.e., the byte pointed to by the address and the three subsequent bytes.
4. DataMem class: is similar to the instruction memory, except that it provides both read and write access.

Main Function

The main function defines a 32 bit program counter (PC) that is initialized to zero. The MIPS simulation routine is carried out within a while loop. In each iteration of the while loop, you will fetch one instruction from the instruction memory, and based on the instruction, make calls to the register file, ALU and data memory classes.

The architectural state consists of the Program Counter (PC), the Register File (RF) and the Data Memory (DataMem).
Specifically, the OutputRF() function is called at the end of each iteration of the while loop, and will add the new state of the Register File to "RFresult.txt". Therefore, at the end of the program execution "RFresult.txt" contains all the intermediate states of the Register File. Once the program terminates, the OutputDataMem() function will write the final state of the Data Memory to "dmem.txt". These functions have been implemented for you.

(Note: You should delete the "RFresult.txt" file before re-executing your program, otherwise the new results will append to the previous results.)