CS3339 Project 2

Description: In this project, you will extend your ARM disassembler with a simulator.

Part 2: Simulator

Your program will create an instruction-by-instruction simulation of the ARM program. This simulation will execute instructions sequentially (non-pipelined) and output the contents of all registers and memory (the state of the processor and memory) after each instruction. You will not have to implement exception/interrupt handling.

Instructions: (all arg refs to machine not assembly instruction - based on my code - your milage may vary) I will add more as I completely test them in my code.

B: Extends and shifts (multipies by 4) the 26 bit argument (words) and adds the value to the PC.

CBZ, CBNZ: Does comparison against zero of arg2 and if condition met adds extended and shifted offset to the PC

ADDI: Adds extended immediate value to the value in R1 register and puts value in arg3 register

The simulation file will have the following format:

- 20 equal signs and a newline
- cycle: [cycle number] [tab] [instruction address] [tab] [instruction string (same as step 3 above)]
- [blank line]
- registers:
- r00: [tab] [integer value of R00][tab] [integer value of R01][tab] …[integer value of R07]
- r08: [tab] [integer value of R08][tab] [integer value of R09][tab] …[integer value of R15]
- r16: [tab] [integer value of R16][tab] [integer value of R17][tab] …[integer value of R23]
- r24: [tab] [integer value of R24][tab] [integer value of R25][tab] …[integer value of R31]
- [blank line]
- data:

- [data address]: [tab] [show in blocks of max 8 data words, with tabs in between]
- …[continue until last data word]

Instructions and arguments should be in capital letters. All integer values should be in decimal. Immediate values should be preceded by a # sign. Be careful and consider which instructions take signed values and which take unsigned values. Be sure to use the correct format depending on the context.

NEW OUTPUT REQUIREMENTS!

Your program will produce 2 output files named <command line output argument>_dis.txt, which contains the disassembled program code for the input MIPS machine code, and <command line output argument>__sim.txt which id the simulation output,. So use the command line provided output file name like you did for dis and make your sim file be the same way. Append _sim to the command line output file input.