ISA Project Documentation

Simulator: sim.exe  
main file: main.c. main function: main(int argc, char\*\* argv).

the project is divided to modules.  
**cpu module:**

DATA:

each opcode/register/IOregister is defined as an enum with its correct “SIMP” index.

instruction is a struct containing rd,rs,rt,imm and opcode values, already parsed from memory (“memin.txt”).

cpu structure holds the **current** PC counter, parsed instruction, registers, IOregisters, memory and irq status.

operation is a pointer to array of functions. every command operator is a function defined in operators module.

OPERATION:

cpu intiliazed in *sim\_init*().

each clock cycle, the cpu fetches an instruction using fetch\_address(cpu).

each (legal) instruction will be executed by executeInstruction(cpu). otherwise, “ignore and continue”.

**operators module:**

OPERATION:

contains all the commands defined in SIMP.

**filesManager module:**

Data:

irq2 structure contains all the data required to handle irq2 interrupt.  
Operation:

Parses memin.txt and writes memout.txt

Parses diskin.txt and writes diskout.txt

write cycles.txt

initializing irq2 structure.

**main module:**

Operation:

write\_trace(cpu,trace\_file\_desc) writes a trace line for trace.txt.

main(int argc, char\*\* argv) ‘pseudo code’:  
  
1. initializes cpu

2. Parses memin.txt, diskin.txt, initializing irq2.txt and starts parsing. Note that irq2.txt is being parsed during the entire program run, as answered in the course forum.

3. open outfiles

4. for every clock cycle until HALT command:

\* check and handle interrupt

\* fetch correct address

\* write trace

\* handles leds.txt , display.txt and hwregout.txt if IN or OUT commands

\* update irq0status and irq2status (note that irq0status is updated inside executeInstruction(cpu)

\* execute instruction

\* handle disk request

5. write out files and close files

6. free all dynamic allocated memory