ATLAS Processor README

The overall philosophy of this architecture is to be highly optimized to accomplish the heavy bit manipulation, carries, and basic arithmetic required to execute the three programs. This will be achieved using an instruction set that is geared towards prioritizing instructions that will make the bit manipulation, arithmetic operations, and related conditionals easier. It will also be achieved by putting the majority of the computational workload on the hardware and related instructions, in order to make the software easy to understand and efficient in the number of clock cycles it takes to execute each program. The architecture will use a **reg-reg/load-store** classification.

ATLAS Overview

TYPE	FORMAT	CORRESPONDING INSTRUCTIONS
R	1 bit type, 3 bits opcode, 3 bits source reg, 2 bits target reg	AND, ADD, OR, SUB, XOR
R*	1 bit type, 3 bits opcode, 2 bits source reg, 3 bits target reg	MOV
R-SHIFT	1 bit type, 3 bits opcode, 3 bits source reg, 1 bit shift direction, 1 bit low/high word	SH
J	1 bit type, 3 bits opcode, 5 bit immediate	J
I	1 bit type, 2 bits opcode, 3 bits source reg, 3 bits immediate	BEQZ, LI (load immediate)
I-MEM	1 bit type, 2 bits opcode, 1 bit source reg, 5 bits immediate	LD, STR

Program Scores (ALL PASSED)

- Program 1: 101/101
- Program 2: 31/31
- Program 3: 12/12

Hardware/Software Files:

Top Module:
TopLevel.sv
InstROM.sv
InstructionDecode.sv
Control.sv
PC_LUT.sv
IF_module.sv

reg_file.sv
ALU_mux.sv
ALU.sv
FlagReg.sv
data_mem.sv

Testbenches & DummyDUTs:
data_mem_DUT1.sv
data_mem_DUT2.sv
data_mem_DUT3.sv
flt2fix_tb.sv —---------- Program 2 TB
flt2fix0.sv —---------- Program 2 Dummy DUT
fltflt_no_rnd_tb.sv —---------- Program 3 TB
fltflt0_no_rnd.sv —---------- Program 3 Dummy DUT
TEXT FILES FOR MEM INITIALIZATION (for each program)
MemoryInitProg1.txt
MemoryInitProg2.txt
MemoryInitProg3.txt
new_fix2flt_tb.sv —---------- Program 1 TB
TEXT FILES FOR PROGRAM INITIALIZATION (machine code for each program)
Prog1_machine.txt
Prog2_machine.txt
Prog3_machine.txt
Top_level0.sv —---------- Program 1 Dummy DUT

Assembler:
***NOTE:** Files labeled Assembly have the main code. All information about other files generated are described below.
assembler2.py —---------- main assembler program
Prog1_Assembly.txt
Prog1_comments.txt
Prog1_lut.txt
Prog1_machine.txt
Prog2_Assembly.txt
Prog2_comments.txt
Prog2_lut.txt
Prog2_machine.txt
Prog3_Assembly.txt
Prog3_comments.txt
Prog3_lut.txt
Prog3_machine.txt

How to Use:

Synthesis:
- data_mem.sv: comment out initial begin block with readmemb
- InstROM.sv: comment out from logic[8:0] to always_comb InstOut, uncomment bottom block. Bottom block contains sample instructions for synthesis purposes.
- NO CHANGES to all other files

Simulation:
- data_mem.sv: change the memory masks for each program (readmemb statement). Also change the filepath according to where the MemoryInit text files for each program are located.
- InstRom: change the program machine code for each program (Readmemb statement). Also change the filepath according to where the Machine text files for each program are located.
- PC_LUT: Set PC_Lut appropriately based on program number, and keep others commented out
- TopLevel: change assign done statement based on program number (last lines of file)
- Load new test benches, Dummy DUTs, and associated memory files for simulating each program
- NO CHANGES to all other files

Assembler:
- Make sure assembler2.py and the target Assembly file is in the same folder
- Uncomment the bottom of the .py file for each program's Assembly file
- 3 files will be generated: comments, LUT, machine
-comments.txt will have the Assembly file converted to machine text, annotated with the corresponding PC number and Assembly code/comments.
- LUT.txt will map the labels in Assembly to the appropriate PC values. Generally, you would then go to comments.txt, check for the J instructions, check the J label index (last 5 bits), map it to the appropriate PC value, and update the PC_LUT. But all of that is already done.
- machine.txt has the machine code ready for input in simulation (no annotations)