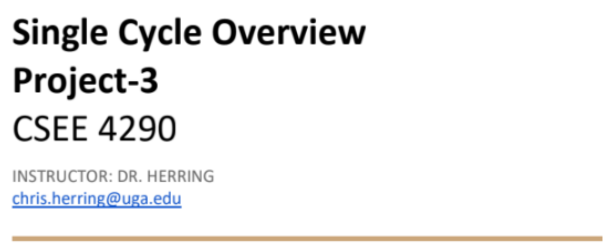
SCC Unix Meeting 0
Fall - 10/14/2024

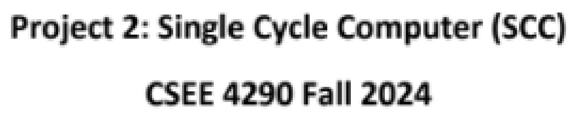
Version: 0.0.2

Sacred Texts



Single Cycle Overview
Project-3
CSEE 4290

INSTRUCTOR: DR. HERRING
chris.herring@uga.edu



Project 2: Single Cycle Computer (SCC)

CSEE 4290 Fall 2024

SingleCycleOverview.pdf

Project Two SCC.pdf

Revision Table

Changes	Authors	Version
[10/14/2024] Created Document.	VanEssendelft, Jake	0.0.1
[10/14/2024] Set up the document for the first meeting.	VanEssendelft, Jake	0.0.2

Table of Contents

Meeting Agenda

- Meeting Attendance (just to know who we need to keep in the loop).
- Review previous checkpoint and feedback from Dr. Herring.
    - Catch up on code updates
    - Current state of our SCC
    - Steps to update previous files
- What is needed for next checkpoint
- Task assignment

Attendance

Name (Last, First)	Project Assignment	Attendance
Almanza-Castillo, Vinessa	Checker	Present
Brewster, Samuel	Documentation/[Checker]	Absent
Gilliland, Drew	Emulator	Departed Early
Glover, Tray	Parser/Compiler	Absent
VanEssendelft, Jake	Emulator	Late
Zayas, Giovanni	Parser/Compiler	Late

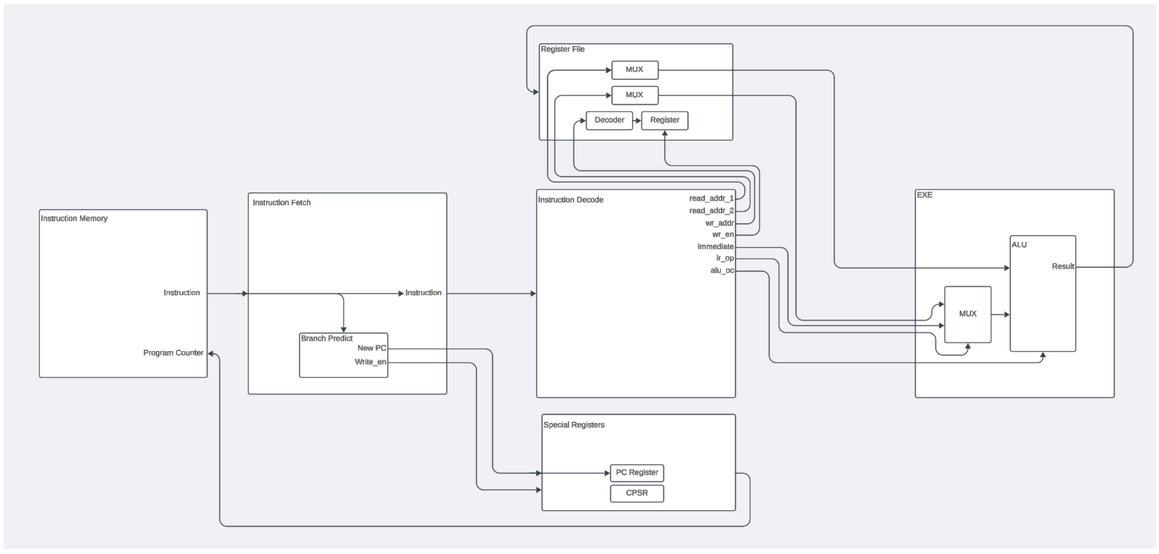
Previous Checkpoint

- Review code: GitHub Link
    - Side note: we **NEED** to organize our repo a little bit
- Changes to previous files
    - IF.v
    - ID.v
    - ALU.v
- New files
    - Reg_File.v
    - EXE.v
    - IF_tb.v
    - ID_tb.v
    - ID_tb2.v

- Feedback
    - Branch prediction should NOT generate any NOP instructions, the instructions in the Project Overview are incorrect according to Dr. Herring.
        - If Instruction Memory isn't clocked on reading values -> just update the program counter.

Next Checkpoint

- Next checkpoint requires all [34] instructions to work
  - Currently, can execute 15~17 instructions
  - Need to edit the ID.v to manage the control lines necessary to complete these instructions.
  - Need to implement the following:
    - LOAD, STOR, ADDS*, SUBS*, ANDS*, ORS*, XORS*, LSL, LSR, Bcond, NOP, HALT
- Requires:
  - Fully functional Data Memory (Load/Store instructions)
  - Fully functional Special Registers (Mainly CPSR)
  - -> Also means that the program counter will need a change
  - Understanding of the Compiler to generate test instructions
- INSTRUCTION MEMORY vs .MEM FILES
  - The compiler/parser is set to make a .mem file, this should be able to be read by verilog code rather than us storing data into registers.
  - AMD documentation
- Fixing the IF, ONCE AGAIN
  - If Instruction Memory doesn't need to be clocked for the instruction read, we need to implement how to make B and BR instructions take 0 clock cycles.
  - Shouldn't be too difficult
  - Also need to fix the B.cond instructions to ensure that prefetched instructions don't get executed when not needed
  - Whenever a jump is completed -> program counter should just truncate the lowest 2 bits (specifically for the BR instruction)



Lucid Diagram

Task Assignment

Task	Member(s)	Information	Status
Compiler Output/Usage	Tray & Giovanni	Need to look into how the Compiler outputs, and how to use the .mem file generated to input into the Instruction Fetch. Do research over .mem files and how verilog can access them.	In Progress
Special Registers	Vinessa	Make another set of 8 registers (similar to the original register file) but this one is not accessible by the user: STILL ACCESSIBLE BY INSTRUCTIONS. Any instruction that will set the CPSR needs to be able to edit the values here. Also, need to look at using a register as the program counter will change the IF as well.	In Progress
IF fixes	Jake	Need to fix the B and BR instructions to fit the requirement of those instructions taking 0 clock cycles and should never be seen by the ID. Still need to consider back-to-back B/BR instructions. Also work with Special Registers for the Program Counter.	In Progress
Documentation	Sam	Update documentation for our special register and zero register (? up for discussion), also will need to check with the Windows group. I think the class decided that R7 is CPSR and organized the same as ARMv8, R6 is the Program Counter, R5 to be the	In Progress

		Link Register, R4 to be the Stack Pointer, and R0 to be the zero register. **ALSO** the instruction for LSL is incorrect in the document: should be 0000100.	
Data Memory	Drew	Need to double check the Data Mem file and also write a testbench to test its functionality. Fix any errors that originate with the testbench.	In Progress
Block Diagram	Jake	It's a lot of fun honestly.	In Progress