CYCLE-BY-CYCLE SEMANTICS OF AUBIE CPU INSTRUCTIONS

All Instruction in state 1 and 2

State 1: Load the 32-bit memory word stored at the address in the PC to the Instruction Register: Mem[PC] -> InstrReg
State 2: decide which instruction you have by looking at opcode portion of Instruction register

- **ALU:** go to state 3
- **LDI or LD:** go to state 7
- **STO:** go to state 9
- **LDR:** go to state 12
- **STOR:** go to state 14
- **JMP or JZ:** go to state 16
- **NOOP:** go to state 19

~~ALU Instructions :~~
Example: ADDU R1,R2,R3 (perform the specified operation on R2 and R3, store result inR1).

State 3: copy operand 1 (R2 in the example) value from register file to Op1 register: Regs[IR[op1]] -> Op1. Goto state 4.
State 4: copy operand 2 (R3 in the example) value from register file to Op2 register: Regs[IR[op2]-> Op2. Go to state 5.
State 5: copy ALU output into result register
ALUout -> Result. Go to state 6.
State 6: copy Result back into the destination register (R1 in the example), copy PC+1 to PC. Result ->Regs[IR[dest]], PC + 1 -> PC. Go to state 1.

~~STO Instructions :~~
Example: STO R1,0x3456789a (store contents of R1 into addr 0x3456789a)

State 9: Increment PC: PC+1-> PC. Go to state 10.
State 10: Load memory at address given by PC to the Addr register: Mem[PC]-> Addr. Go to state 11.
State 11: Store contents of src register (R1 in example) to address in memory given by Addr register, and increment the PC: Regs[IR[src]] -> Mem[Addr], PC+1 -> PC. Go to state 1.

~~LD instruction~~
Example: LD R6, 0x11112222 (load contents of addr 0x11112222 to R6)

State 7: Increment PC: PC+1 -> PC. Copy memory specified by PC into Addr register: Mem[PC] -> Addr . Go to state 8.

State 8: Copy memory location specified by Addr to the dest register (R6 in example): Mem[Addr] -> Regs[IR[dest]]. Increment PC: PC+1-> PC. Go to state 1.

~~LDI Instruction~~

Example: LDI R7,#0xabcdef01 (load constant 0xabcdef01 to register 7)
State 7: increment PC: PC+1->PC **.** Copy memory specified by PC into Immediate register: Mem[PC] -> Immed. Go to state 8.
State 8: Copy immed register into the dest register (R7 in example): Immed -> Regs[IR[dest]]. Increment PC: PC+1-> PC. Go to state 1.

STOR instruction

Example: STOR (R7),R8 (store contents of R8 into address specified by contents of R7)

State 14: copy contents of dest reg (R7) into Addr register: Regs[IR[dest]]-> Addr. Go to state 15.
State 15: copy contents of op1 register (R8 in example) to Memory address specified by Addr: Regs[IR[op1]]-> Mem[Addr]. Increment PC: PC+1 -> PC. Go to state 1.

LDR instruction

Example: LDR R11,(R12)   (load contents of address specified by contents of R12 to register R11)

State 12: copy contents of op1 reg (R12 in example) to Addr register: Regs[IR[op1]] -> Addr. Go to state 13.
State 13: copy contents of memory specified by Addr register to dest register: Mem[Addr]->Regs[IR[dest]]. Increment PC: PC+1-> PC. Go to state 1.

(**NOTE: To implement jumps you must change the pc_mux to a 3-way, adding a path from mem_out to the pc_mux. This requires changing the datapath and the interconnect files)**

JMP instruction

Example: JMP 0x12123434 (jump to address 0x12123434)

State 16: Increment PC: PC->PC+1. Go to state 17
State 17: Load memory specified by PC to Addr register: Mem[PC]->Addr. Go to state 18
State 18: Load Addr to PC: Addr -> PC. Go to state 1.

JZ instruction

Example: JZ R4,0x12123434 (if R4 == 0, jump to address 0x12123434)

State 16: Increment PC: PC+1-> PC. Go to state 17
State 17: Load memory specified by PC to Addr register: Mem[PC]->Addr, copy register op1 to control: Regs[IR[op1] -> Ctl, Go to state 18
State 18: if Result == 0, copy Addr to PC: Addr -> PC, else increment PC: PC+1 -> PC. Go to state 1.

~~**NOOP instruction**~~
Example: NOOP (do nothing except increment the PC)

State 19: copy PC+1 to PC: PC+1 -> PC. Go to state 1.