Project5 for Intro to Computer Systems 2025 spring

Yiming Cheng

12450588

File Structure

```
├── CPU.hdl
├── Computer.hdl
├── Memory.hdl
├── MyDesign
│   ├── CPU.pdf
│   ├── Computer.pdf
│   └── Memory.pdf
```

├── README.md

└── README.pdf

Optimize for CPU.hdl:

```
PARTS:
    Mux16(a=inst, b=aluResult, sel=inst[15], out=AInput);

    Not(in=inst[15], out=isAInstruction);

    // RegisterA
    // when inst[15] = 0, it is @value means A should load
value
    Or(a=isAInstruction, b=inst[5], out=loadAReg); //d1
    ARegister(in=AInput, load=loadAReg, out=AOutput,
out[0..14]=memAddress);

    Mux16(a=AOutput, b=memInput, sel=inst[12],
out=AMInput);

    // ALU Control Signals: Prepare for ALU computation
    And(a=inst[11], b=inst[15], out=zx); // Zero the X
input if needed
    And(a=inst[10], b=inst[15], out=nx); // Negate the X
input if needed
    Or(a=inst[9], b=isAInstruction, out=zy); // Zero the Y
input if needed
```

```
    Or(a=inst[8], b=isAInstruction, out=ny); // Negate the
Y input if needed
    And(a=inst[7], b=inst[15], out=f);  // Select
operation: Add (1) or And (0)
    And(a=inst[6], b=inst[15], out=no); // Negate ALU
output if needed

    // ALU: Executes the computation specified by the
instruction
    ALU(x=DOutput, y=AMInput, zx=zx, nx=nx, zy=zy, ny=ny,
f=f, no=no, out=memOutput, out=aluResult, zr=isZero,
ng=isNegative);

    // when it is an instruction, write M
    And(a=inst[15], b=inst[3], out=writeMem); //d3

    // RegisterD, when it is an instruction, load D
    And(a=inst[15], b=inst[4], out=loadDReg); //d2
    DRegister(in=aluResult, load=loadDReg, out=DOutput);

    // Prepare for jump
    // get positive
    Or(a=isZero, b=isNegative, out=notPositive);
    Not(in=notPositive, out=isPositive);

    And(a=inst[0], b=isPositive, out=jumpGreater); //j3
    And(a=inst[1], b=isZero, out=jumpEqual);       //j2
    And(a=inst[2], b=isNegative, out=jumpLess);    //j1

    Or(a=jumpLess, b=jumpEqual, out=jumpLE);
    Or(a=jumpLE, b=jumpGreater, out=shouldJump);

    And(a=shouldJump, b=inst[15], out=doJump);
```

```
    // when jump, load AOutput
    PC(in=AOutput, load=doJump, reset=resetSignal,
inc=true, out[0..14]=programCounter);
}
```

Difference:

`CPU.hdl`

```
ALU(x=Dout,y=AMout,zx=instruction[11],nx=instruction[10],z
y=instruction[9],ny=instruction[8],f=instruction[7],no=ins
truction[6],out=outM,out=ALUout,zr=zero,ng=neg);
```

`Optimize`

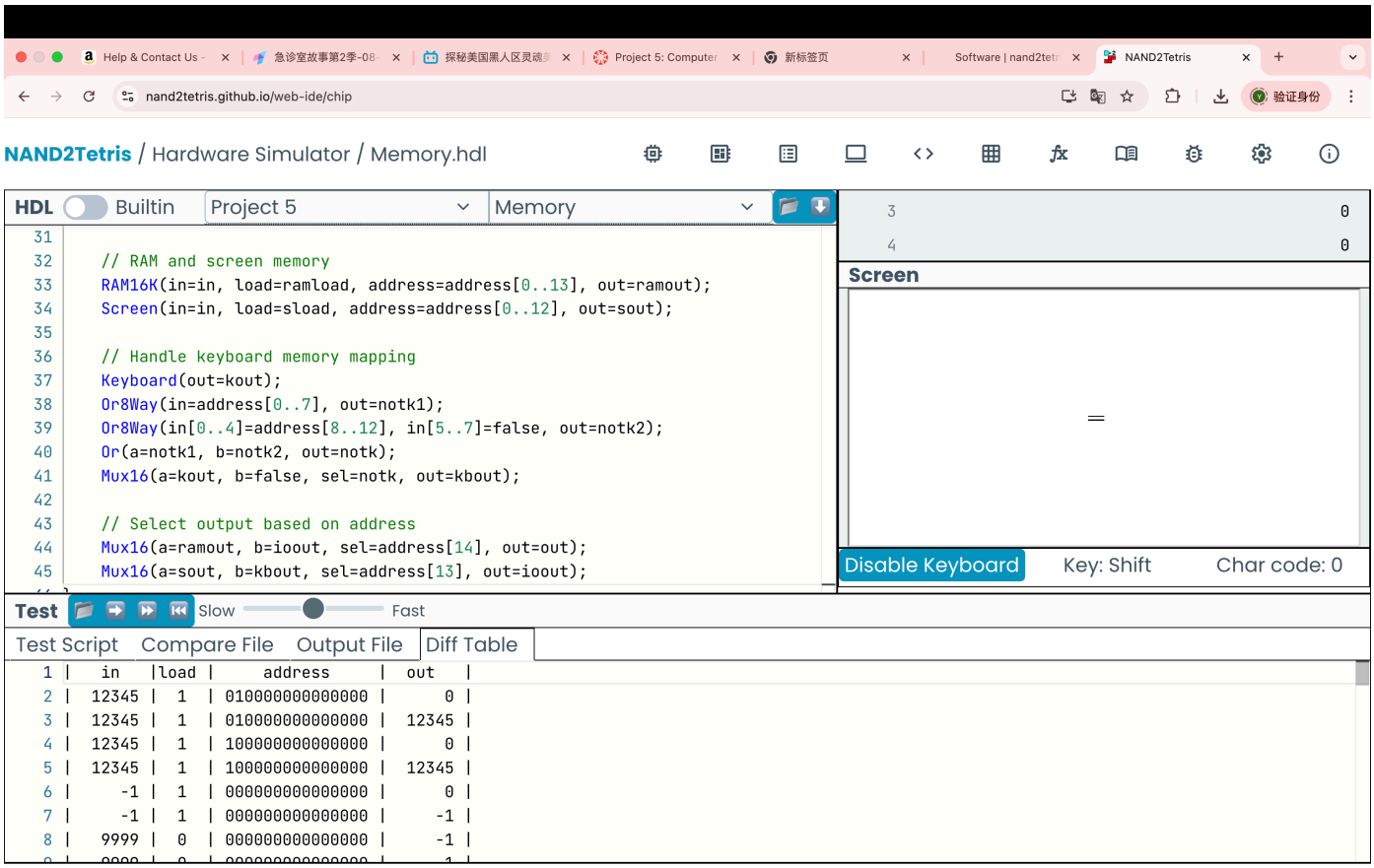
```
    And(a=instruction[11],b=instruction[15],out=zx);//c1
    And(a=instruction[10],b=instruction[15],out=nx);//c2
    Or(a=instruction[9],b=notinstruction,out=zy);//c3
    Or(a=instruction[8],b=notinstruction,out=ny);//c4
    And(a=instruction[7],b=instruction[15],out=f);//c5
    And(a=instruction[6],b=instruction[15],out=no);//c6


ALU(x=Dout,y=AMout,zx=zx,nx=nx,zy=zy,ny=ny,f=f,no=no,out=o
utM,out=ALUout,zr=zero,ng=neg);
```

Output

Memory.hdl

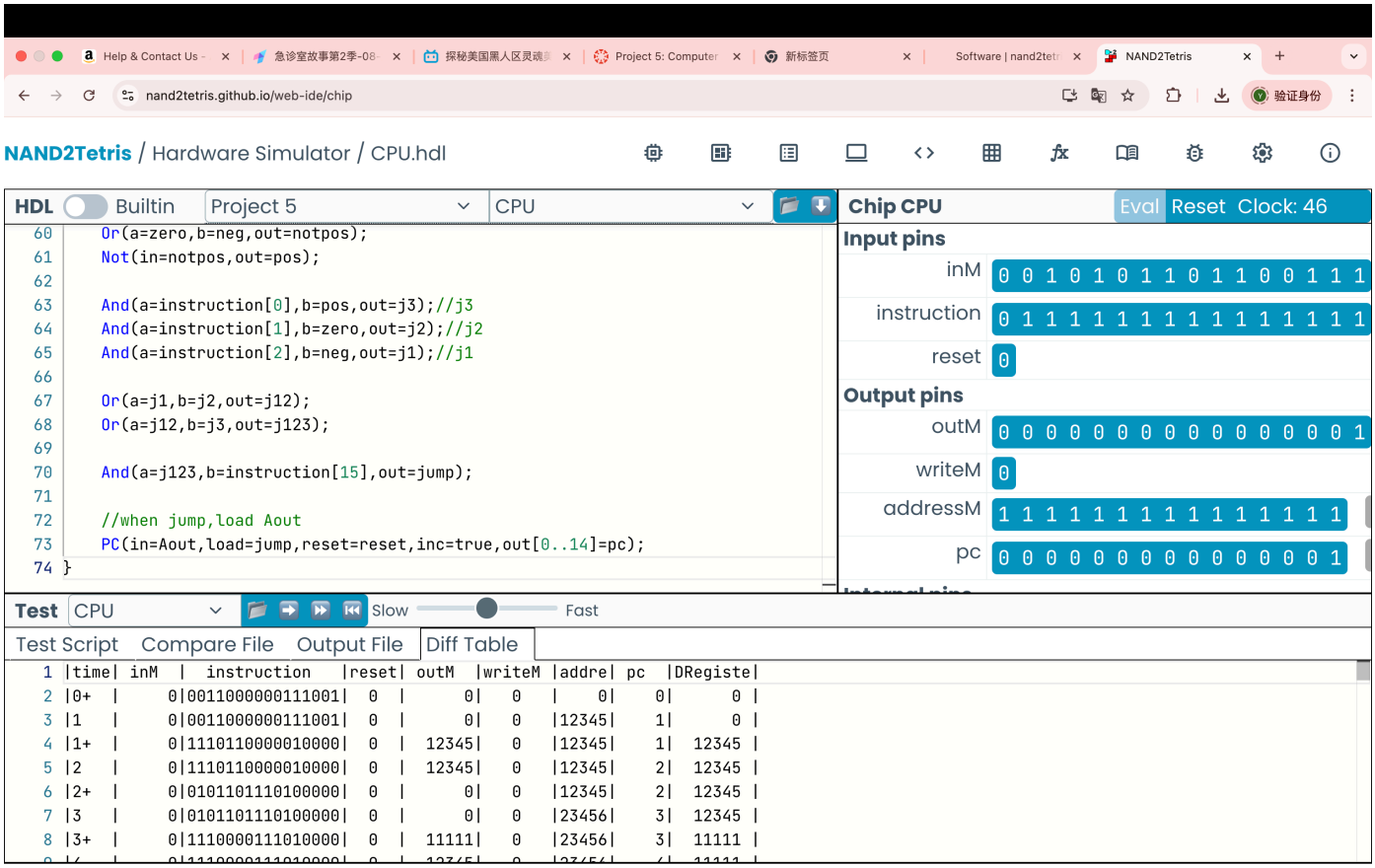
All tests pass including the keyboard cases



Simulation successful: The output file is identical to the compare file

CPU.hdl

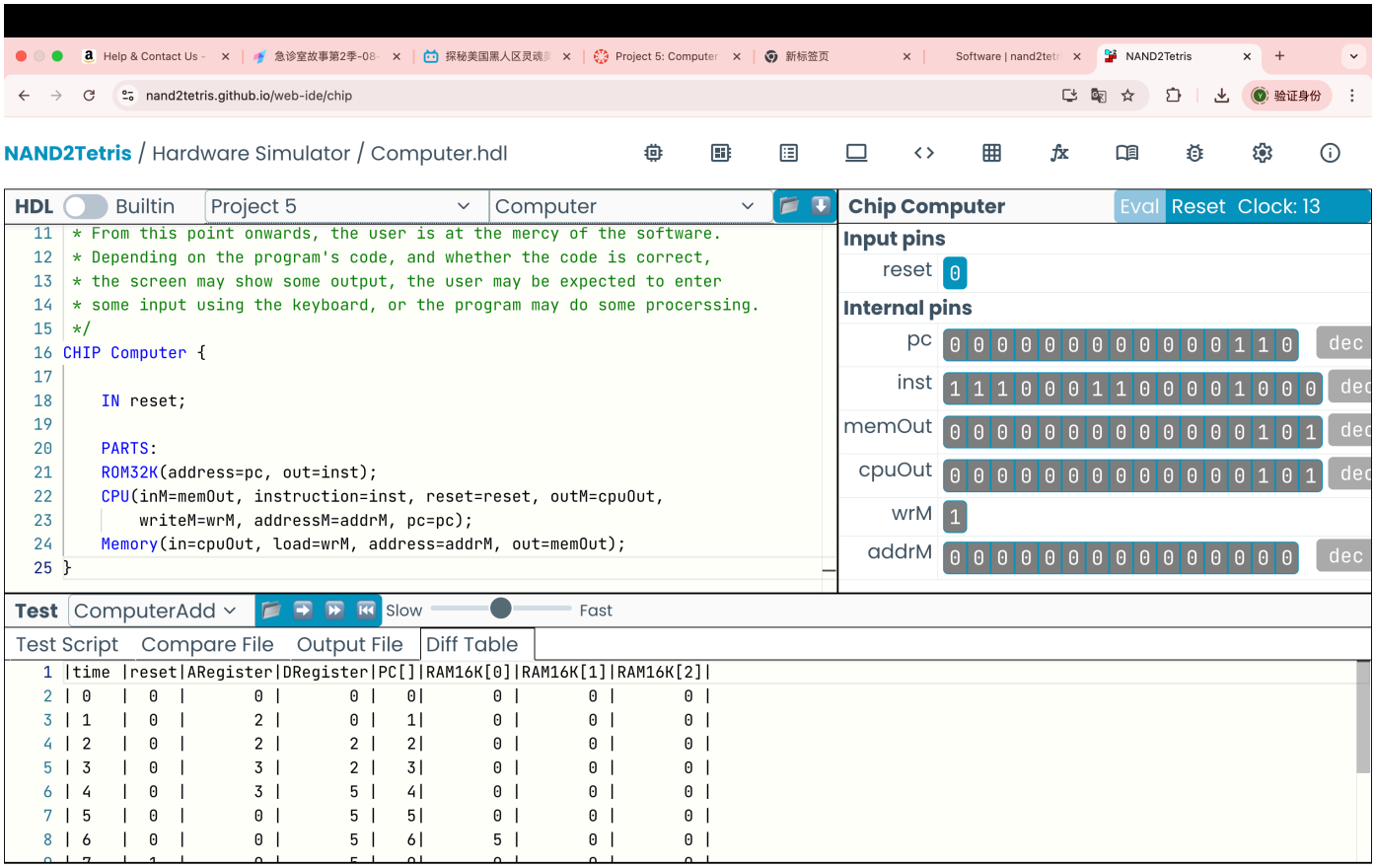
All tests pass



Simulation successful: The output file is identical to the compare file

Computer.hdl

All tests pass



Simulation successful: The output file is identical to the compare file