

Project5 for Intro to Computer Systems 2025 spring

Yiming Cheng

12450588

Project5 for Intro to Computer Systems 2025 spring

File Structure

Optimize for CPU.hdl:

Output

Memory.hdl

CPU.hdl

Computer.hdl

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

```

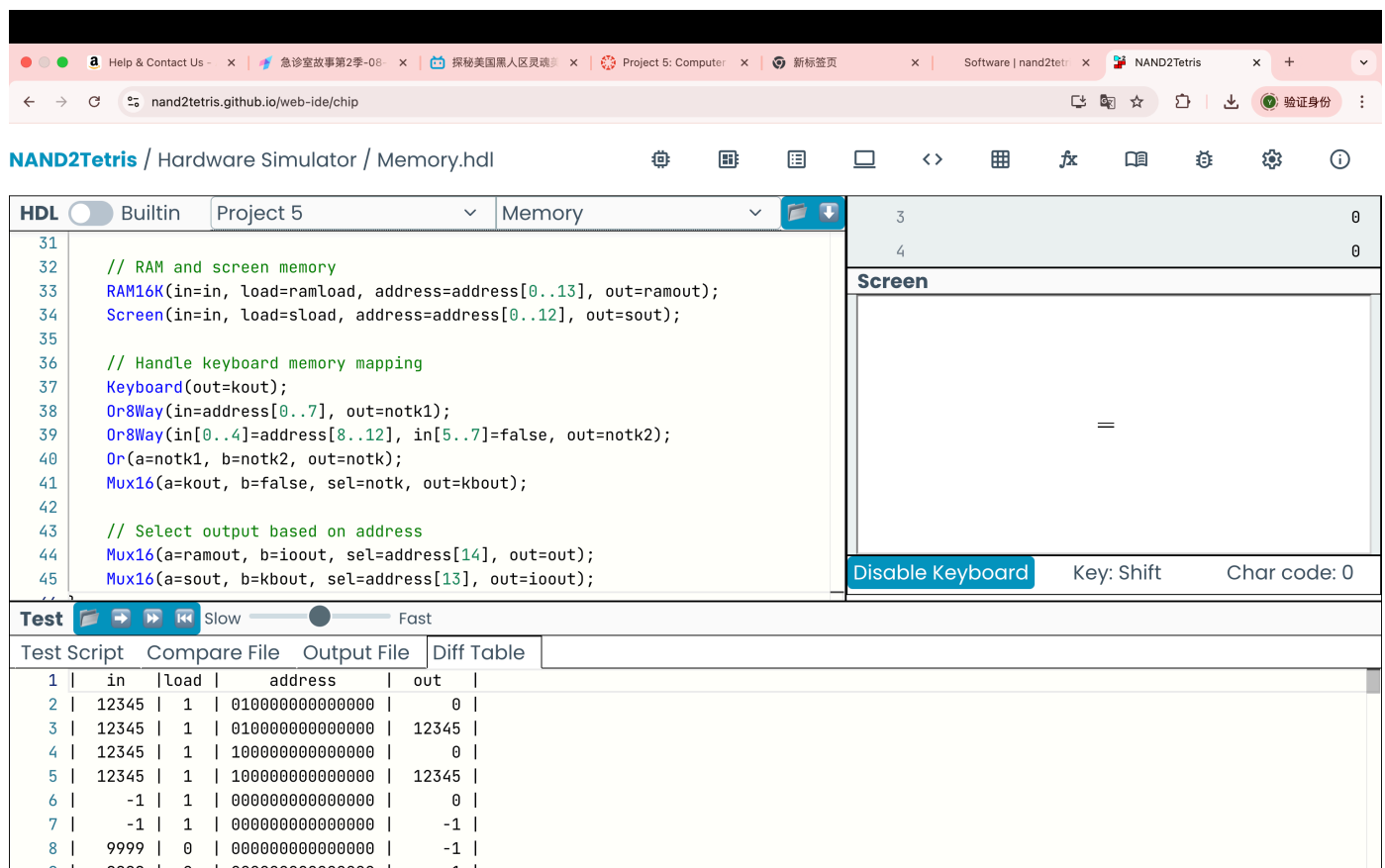
```

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);

```

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

Help & Contact Us

急诊室故事第2季-08

探秘美国黑人区灵魂

Project 5: Computer

新标签页

Software | nand2tet

NAND2Tetris

nand2tetris.github.io/web-ide/chip

NAND2Tetris / Hardware Simulator / CPU.hdl

🔧

📄

📑

🖨

⏪

🔍

🔖

📖

⚙

ℹ

HDL

Builtin

Project 5

CPU

60

Or(a=zero,b=neg,out=notpos);

61

Not(in=notpos,out=pos);

62

63

And(a=instruction[0],b=pos,out=j3);//j3

64

And(a=instruction[1],b=zero,out=j2);//j2

65

And(a=instruction[2],b=neg,out=j1);//j1

66

67

Or(a=j1,b=j2,out=j12);

68

Or(a=j12,b=j3,out=j123);

69

70

And(a=j123,b=instruction[15],out=jump);

71

72

//when jump,load Aout

73

PC(in=Aout,load=jump,reset=reset,inc=true,out[0..14]=pc);

74

}

Chip CPU

Eval

Reset

Clock: 46

Input pins

inM0 0 0 1 0 1 0 1 1 0 1 1 0 0 1 1 1

instruction0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

reset0

Output pins

outM0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1

writeM0

addressM1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

pc0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1

Test

CPU

Slow

Fast

Test Script

Compare File

Output File

Diff Table

	time	inM	instruction	reset	outM	writeM	address	pc	DRegister
1	0+		0 0011000000111001	0		0		0	
2	1		0 0011000000111001	0		0	12345	1	
3	1+		0 1110110000010000	0	12345	0	12345	1	12345
4	2		0 1110110000010000	0	12345	0	12345	2	12345
5	2+		0 0101101110100000	0		0	12345	2	12345
6	3		0 0101101110100000	0		0	23456	3	12345
7	3+		0 1110000111010000	0	11111	0	23456	3	11111
8	4		0 1110000111010000	0	12345	0	23456	4	11111

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

Computer.hdl

All tests pass

