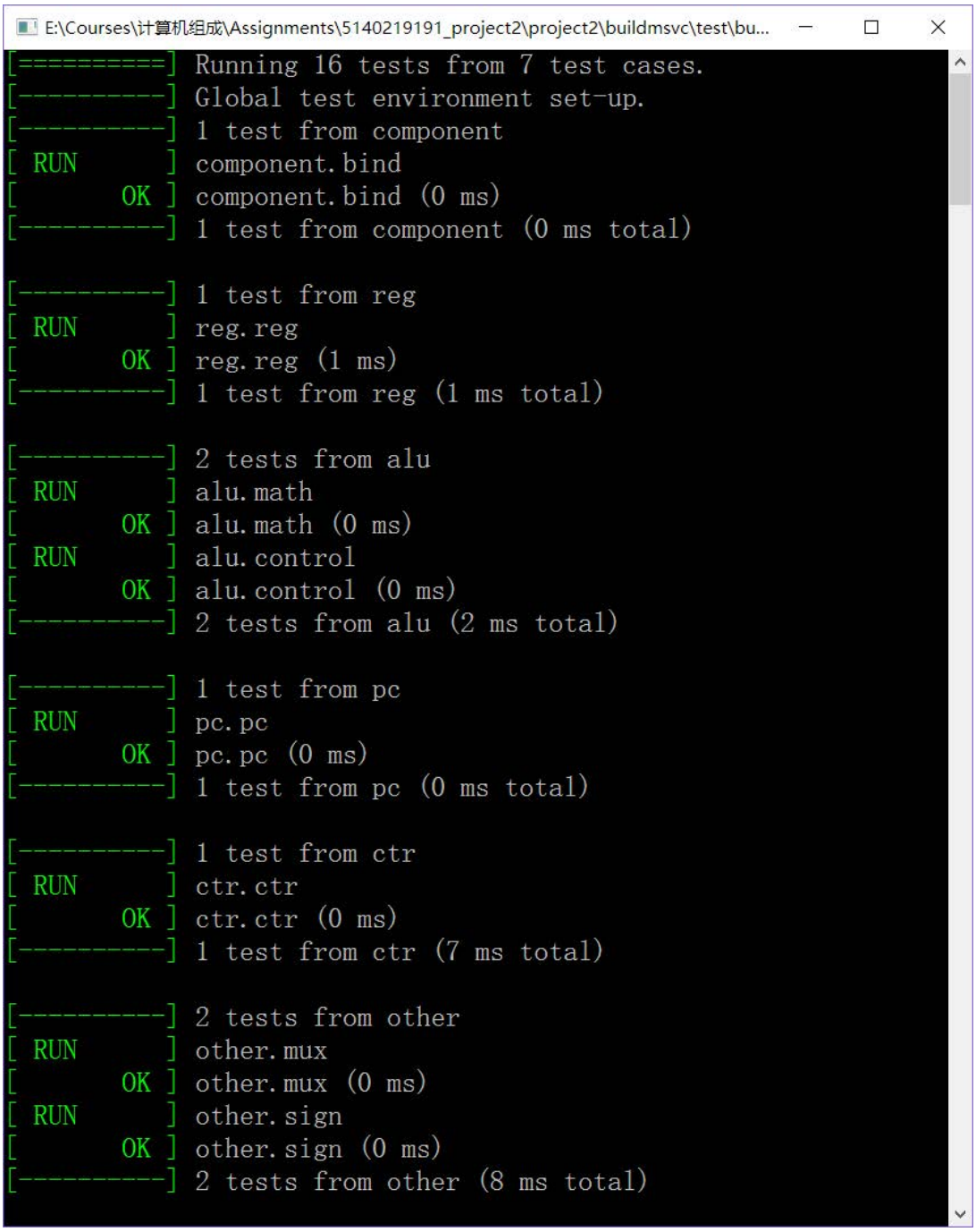
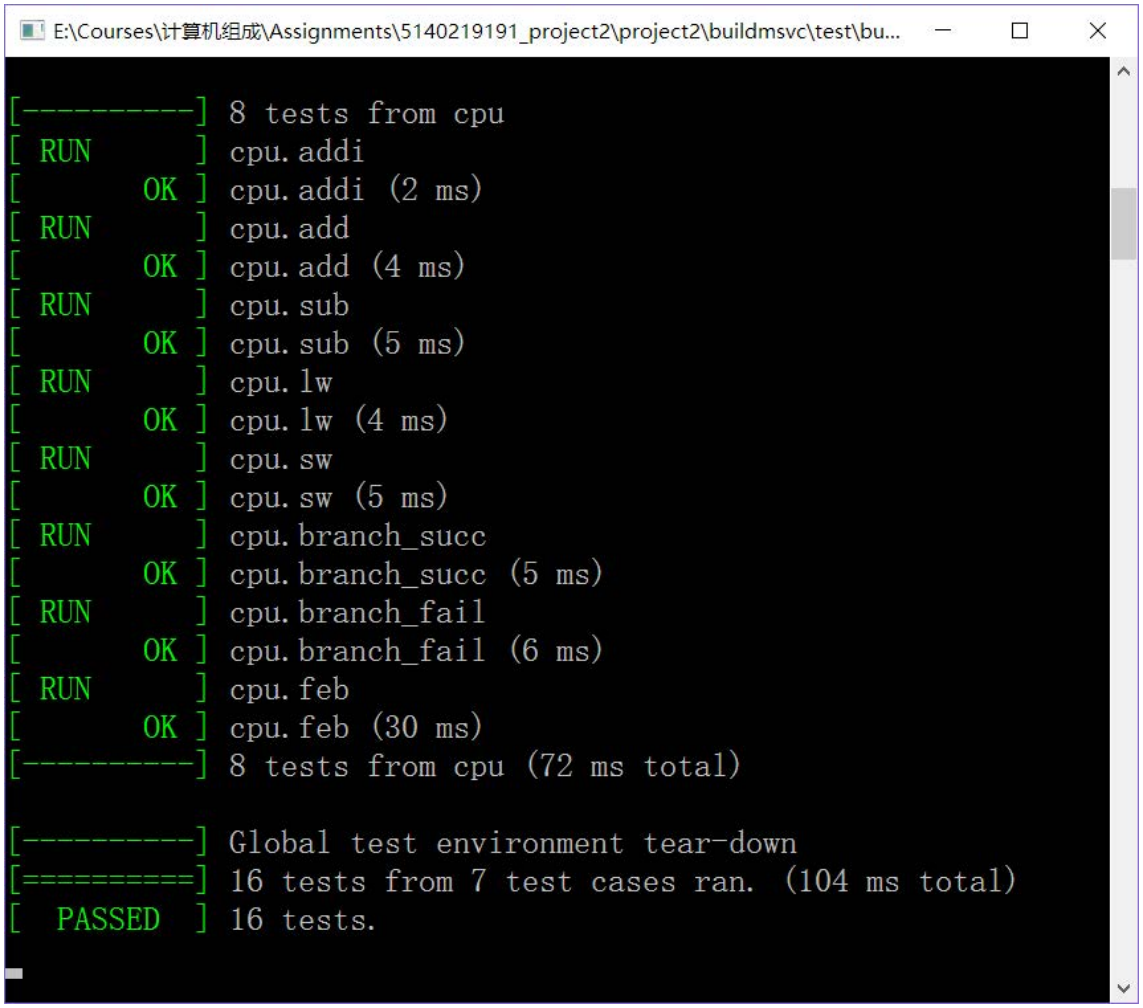
EI209: Computer Organization Autumn 2015

Project II

Due: 3/1/2016 *5140219191* 包伟铭

1 Screenshot of the console





2 Source code

- **Ctr:**

```
if (in[opCode] == B("000010")) {//jump
    setOutput(jump, 1);
    setOutput(regDst,0);
    setOutput(ALUSrc,0);
    setOutput(memToReg,0);
    setOutput(regWrite,0);
    setOutput(memRead,0);
    setOutput(memWrite,0);
    setOutput(branch, 0);
    setOutput(aluOp,B("00"));
}
```

```
else if (in[opCode] == B("000000")) {//R-type
    setOutput(jump, 0);
    setOutput(regDst,1);
    setOutput(ALUSrc,0);
    setOutput(memToReg,0);
    setOutput(regWrite,1);
    setOutput(memRead,0);
    setOutput(memWrite,0);
    setOutput(branch, 0);
    setOutput(aluOp,B("10"));
}
else if (in[opCode] == B("100011")) {//lw
    setOutput(jump, 0);
    setOutput(regDst,0);
    setOutput(ALUSrc,1);
    setOutput(memToReg,1);
    setOutput(regWrite,1);
    setOutput(memRead,1);
       setOutput(memWrite,0);
    setOutput(branch, 0);
    setOutput(aluOp,B("00"));
}
else if (in[opCode] == B("101011")) {//sw
    setOutput(jump, 0);
    setOutput(regDst,0);
    setOutput(ALUSrc,1);
    setOutput(memToReg,0);
    setOutput(regWrite,0);
    setOutput(memRead,0);
    setOutput(memWrite,1);
    setOutput(branch, 0);
    setOutput(aluOp,B("00"));
}
else if (in[opCode] == B("000100")) {//beq
    setOutput(jump, 0);
    setOutput(regDst,0);
    setOutput(ALUSrc,0);
    setOutput(memToReg,0);
    setOutput(regWrite,0);
    setOutput(memRead,0);
    setOutput(memWrite,0);
    setOutput(branch, 1);
    setOutput(aluOp,B("01"));
}
else if (in[opCode] == B("001000")) {//addi
    setOutput(jump, 0);
    setOutput(regDst,0);
```

```
     setOutput(ALUSrc,1);
     setOutput(memToReg,0);
     setOutput(regWrite,1);
     setOutput(memRead,0);
     setOutput(memWrite,0);
     setOutput(branch, 0);
     setOutput(aluOp,B("100"));
 }
```

- **ALU Control:**

```
if(in[aluOp]==B("00"))  {
    setOutput(aluCtr,B("0010"));
    return;
}
else if(in[aluOp]==B("01"))  {
    setOutput(aluCtr,B("0110"));
    return;
}
else if(in[aluOp]==B("100"))  {
    setOutput(aluCtr,B("0010"));
    return;
}
else if(in[aluOp]==B("10"))  {
    if(in[funct]==B("00000"))  {
        setOutput(aluCtr,B("0010"));
        return;
    }
    else if(in[funct]==B("00010"))  {
        setOutput(aluCtr,B("0110"));
        return;
    }
    else if(in[funct]==B("00100"))  {
        setOutput(aluCtr,B("0000"));
        return;
    }
    else if(in[funct]==B("00101"))  {
        setOutput(aluCtr,B("0001"));
        return;
    }
    else if(in[funct]==B("01010"))  {
        setOutput(aluCtr,B("0111"));
        return;
    }
}
```

- **ALU:**

```
LineData result = 0;
LineData re_zero = 0;

switch (in[aluCtr])
{

case 0://and
    result = in[input1] & in[input2];
    break;

case 1://or
    result = in[input1] | in[input2];
    break;

case 2://add
    result = in[input2] + in[input1];
    break;

case 6://subtract
    result = in[input1] - in[input2];
    if(result == 0)  re_zero = 1;
    break;

case 7://set on less than
    result = (in[input1] < in[input2]) ? 1 : 0;
    break;

case 12://nor
    result = ~(in[input1] | in[input2]);
    break;
}

setOutput(zero, re_zero);
setOutput(aluRes, result);
```

- **Register:**

onChange():

```
setOutput(readData1, memory[in[readReg1]]);
setOutput(readData2, memory[in[readReg2]]);
```

onClock():

```
if(in[regWrite] && in[writeReg] && in[writeData])  {
    memory[in[writeReg]] = in[writeData];
}
onChange();
```

- **PC:**

```
if((in[branch]==1) && (in[zero]==1) && (in[immData]))  {
    m_pc = m_pc + 4 + in[immData]*4;
    setOutput(newPC, m_pc);
    return;
}
else  {
    m_pc = m_pc + 4;
    setOutput(newPC, m_pc);
    return;
}
```

- **CPU:**

```
//pc
    BIND(pc, newPC, instMem, address);

//instMem
    instMem.input(memRead, 1);
    instMem.bind(readData, partialListener(26, 31, ctr, opCode));
    instMem.bind(readData, partialListener(21, 25, reg, readReg1));
    instMem.bind(readData, partialListener(16, 20, reg, readReg2));
    instMem.bind(readData, partialListener(16, 20, muxRegDes, input1));
    instMem.bind(readData, partialListener(11, 15, muxRegDes, input2));
    instMem.bind(readData, partialListener(0, 15, signExtend, immInput));
    instMem.bind(readData, partialListener(0, 3, aluControl, funct));

//ctr
    BIND(ctr, regDst, muxRegDes, muxSel);
    BIND(ctr, branch, pc, branch);
    BIND(ctr, memRead, dataMem, memRead);
    BIND(ctr, memToReg, muxMem2Reg, muxSel);
    BIND(ctr, aluOp, aluControl, aluOp);
    BIND(ctr, memWrite, dataMem, memWrite);
    BIND(ctr, ALUSrc, muxAlu, muxSel);
    BIND(ctr, regWrite, reg, regWrite);
```

```
//muxAlu
    BIND(muxAlu, muxOut, alu, input2);

//reg
    BIND(reg, readData1, alu, input1);
    BIND(reg, readData2, muxAlu, input1);
    BIND(reg, readData2, dataMem, writeData);

//signExtend
    BIND(signExtend, immData, muxAlu, input2);
    BIND(signExtend, immData, pc, immData);

//muxRegDes
    BIND(muxRegDes, muxOut, reg, writeReg);

//aluControl
    BIND(aluControl, aluCtr, alu, aluCtr);

//alu
    BIND(alu, aluRes, dataMem, address);
    BIND(alu, aluRes, muxMem2Reg, input1);
    BIND(alu, zero, pc, zero);

//dataMem
    BIND(dataMem, readData, muxMem2Reg, input2);

//muxMem2Reg
    BIND(muxMem2Reg, muxOut, reg, writeData);
```

3 Gains

I've got a fruitful insight on how the different parts of a processor coordinates and finally make the processor work, some details are as follows:

(1). Minimize signals of uncertain state

We should ensure all the control signals that might affect some other part stable at 0 or 1, especially when they seems irrelevent to the current instruction.

As the problem I once met in Project2, I only set the zero signal of the ALU to 1 when the ALU does subtract action and forget to set it to 0 under the other circumstance. It caused the failure of "cpu.feb" test.

(2). WriteReg only at the positive edge of the CLK signal, while readReg can happen almost simultaneously

Nowadays, according to common standard, registers can only deal with write action at the postive edge of the CLK signal, while we can almost instantly get the data just after the stable input signals are in.

I suppose the onChange func is used to deal with reading action, while onClock func with writing action (top of P12 of the guide).

4 Thoughts and Thankfulness

This experiment provides us students with a hands-on opportunity enhancing our understanding on how a single-cycle processor (RISC) works with the well-organized C++ project.

In general, I have really gained a lot thoughout the somehow tough progress. Thanks a lot for prof. Haojin Zhu and dear TA's hardwork!