2021年北航计组P5实验报告

1 整体架构设计

1.1 CPU设计方案综述

本实验基于verilog实现了流水线cpu，支持指令集MIPS_lite {addu, addiu, ori, and, subu, lw, sw, lb, sb, beq, blez, lui,, sll, slt, j, jr, jal, nop}，包含IFU, F2D, NPC, CMP, RF, EXT, D2E, ALU, E2M, DM, M2W等模块.

1.2 关键模块

1. ALU - **arithmetic logic unit**
   module ALU(
   input [31:0] A
   input [31:0] B
   input [4:0] shamt
   input [3:0] ALU_op
   output [31:0] C
   );

操作名	功能
ALU_and	A与B
ALU_or	A或B
ALU_nor	A或非B
ALU_Xor	A异或B
ALU_slt	A补码比较小于B置为1
ALU_sltu	A原码比较小于B置为1
ALU_add	A加B
ALU_sub	A减B
ALU_sll	B逻辑左移A的后五位
ALU_srl	B逻辑右移A的后五位
ALU_sra	B算术右移A的后五位

2. CMP - **compare unit**
   module CMP(
   input [31:0] A
   input [31:0] B
   input [2:0] CMP_op
   output cmp
   );

操作名	功能
cmp_eq	A等于B置1
cmp_ne	A不等于B置1
cmp_gez	A大于等于0置1
cmp_gtz	A大于0置1
cmp_lez	A小于等于0
cmp_ltz	A小于0置1
cmp_rtez	B等于0置1
cmp__rtnez	B不等于置1

3. CTR - **control unit**

```
module CTR(
    input [31:0] Instr,
    input cmp,
    //decode
    output [25:0] imm26,
    output [15:0] imm16,
    output [4:0] rs,
    output [4:0] rt,
    output [4:0] rd,
    output [4:0] shamt,
    //forward
    output [4:0] RF_A3,
    //control
    output [4:0] ALU_op,
    output [4:0] CMP_op,
    output [2:0] NPC_op,
    output [2:0] EXT_op,
    output [2:0] DM_op,
    output [2:0] ALU_B_sel,
    output [2:0] RF_WD_sel,
    output [2:0] RF_A3_sel,
    output DM_wr,
    output RF_wr,
    //classify
    output lui,
    output jimm,
    output jreg,
    output jlink,
    output cali,
    output calr,
    output load,
    output store,
    output shifts,
    output branch,
    output branchl,
    output branchlr,
    output bmlr,
    output mov
);
```

4. D2E - **pipeline register (D&E's interstage)**

```
module D2E(
input [31:0] D_Instr
input [31:0] D_PC
input [31:0] D_PC8
input [31:0] D_RS
input [31:0] D_RT
input [31:0] D_EXT_OUT
input D_cmp
input D2E_en
input flush
input clk
input reset
output [31:0] E_Instr
output [31:0] E_PC
output [31:0] E_PC8
output [31:0] E_RS
output [31:0] E_RT
output [31:0] E_EXT_OUT
output E_cmp
);
```

5. DM - **data memory**

```
module DM(
input [31:0] A
input [31:0] WD
input [31:0] PC
input [31:0] Instr
input [2:0] DM_op
input clk
input reset
input DM_wr
output [31:0] DMout
);
```

操作名	功能
DM_lw	`mem[addr] <= WD`
DM_lb	`mem[addr][7+8*A[1:0] -:8] <= WD[7:0]`
DM_sw	`DMout <= mem[addr];`
DM_sb	`DMout <= signed_b(mem[addr][7+8*A[1:0] -:8]);`

6. - **pipeline register (E&M's interstage)**

```
module E2M(
input [31:0] E_Instr
input [31:0] E_PC
input [31:0] E_PC8
input [31:0] E_RS
input [31:0] E_RT
input [31:0] E_ALU_C
input [31:0] E_EXT_OUT
input E_cmp
input E2M_en
input clk
input reset
output [31:0] M_Instr
output [31:0] M_PC
output [31:0] M_PC8
output [31:0] M_RS
output [31:0] M_RT
output [31:0] M_ALU_C
output [31:0] M_EXT_OUT
output M_cmp
);
```

7. EXT - **bit extender**

```
module EXT(
input [15:0] imm16
input [2:0] EXT_op
output [31:0] EXTout
);
```

操作名	功能
EXT_unsigned	将 imm16 输入的 16 位数据做无符号扩展
EXT_signed	将 imm16 输入的 16 位数据无符号扩展
EXT_lui	将imm16 输入的 16 位数据加载到 32 位输出的高位

8. - **pipeline register (F&D's interstage)**

```
module F2D(
input [31:0] F_Instr
input [31:0] F_PC
input F2D_en
input clk
input reset
output [31:0] D_Instr
output [31:0] D_PC
);
```

9. IFU - **instruction fetch unit**

```
module IFU(
input clk
input reset
input IFU_en
input [31:0] NPC
output [31:0] PC
output [31:0] Instr
);
```

功能名称	功能
同步复位	当复位信号有效时，将PC值设置为 0x00003000
取指令	根据当前PC值从IM中取出指令，并输出

10. - **pipeline register (M&W's interstage)**

```
module M2W(
input [31:0] M_Instr
input [31:0] M_PC
input [31:0] M_PC8
input [31:0] M_RS
input [31:0] M_RT
input [31:0] M_ALU_C
input [31:0] M_EXT_OUT
input [31:0] M_DM_OUT
input M_cmp
input M2W_en
input clk
input reset
output [31:0] W_Instr
output [31:0] W_PC
output [31:0] W_PC8
output [31:0] W_RS
output [31:0] W_RT
output [31:0] W_ALU_C
output [31:0] W_EXT_OUT
output [31:0] W_DM_OUT
output W_cmp
);
```

11. NPC - **next pc**

```
module NPC(
input [31:0] F_PC
input [31:0] D_PC
input [31:0] JR
input [25:0] imm26
input [2:0] NPC_op
input cmp
output [31:0] NPC
output [31:0] PC8
);
```

功能名称	功能
输出PC4	NPC = PC + 4
跳转b类型	对于branch类型的指令，输出下一个PC的值
跳转j类型	对于j，jal这样的指令，输出下一个PC的值
跳转寄存器类型	对于jr这样的指令，输出下一个PC的值

12. RF - **register file**

```
module RF(
input [4:0] A1
input [4:0] A2
input [4:0] A3
input [31:0] WD
input [31:0] PC
input [31:0] Instr
input reset
input RF_wr
input clk
output [31:0] RD1
output [31:0] RD2
);
```

功能名称	功能描述
同步复位	当同步复位信号有效时，将所有寄存器的值设置为 0x00000000
读数据	读出 A1 和 A2 地址对应寄存器中存储的数据到 RD1 和 RD2
写数据	当 WE 有效且时钟上升沿到来时，将 WD 的数据写入A3 对应的寄存器中

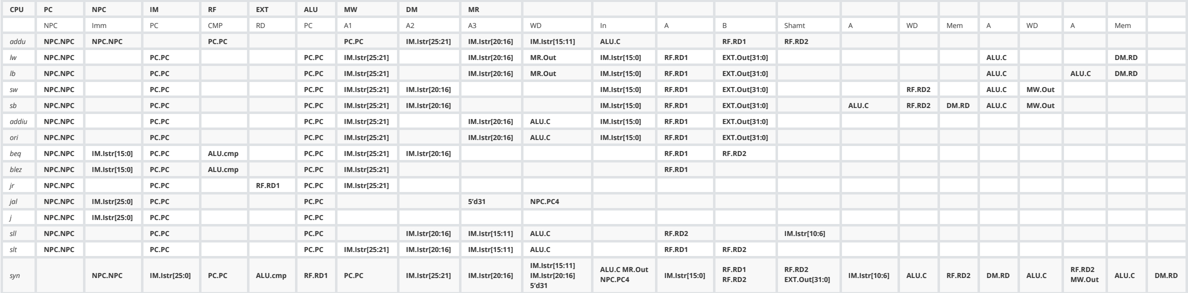
13. STL - **stall unit**

```
module STL(
input [31:0] D_Instr
input [31:0] E_Instr
input [31:0] M_Instr
input [31:0] W_Instr
output IFU_en
output F2D_en
output D2E_en
output D2E_flush
output E2M_en
output M2W_en
output stall
);
```

Control Signals Table

opcode	000000				100011	101011	000100	000110	001101	001001	000010	000011
func	100001	001000	000000	101010	x							
	addu	jr	sll	slt	lw	sw	beq	blez	ori	addiu	j	jal
NPC_op	000	011	000	000	000	000	001	001	000	000	010	010
EXT_op	x	x	x	x	1	1	x	x	0	1	x	x
RF_wr	1	0	1	1	1	0	0	0	1	1	0	1
RF_A3_sel	00	x	00	00	01	x	x	x	01	01	x	10
RF_WD_sel	00	x	00	00	01	x	x	x	00	00	x	10
ALU_B_sel	0	x	x	0	1	1	0	x	1	1	x	x
DM_wr	0	0	0	0	0	1	0	0	0	0	0	0
ALU_op	0000	x	0010	0100	0000	0000	0011	0101	0110	0000	x	x

1.3 DataPath Table



CPU	PC	NPC	IM	RF	EXT	ALU	MW	DM	MR													
	NPC	Imm	PC	CMP	RD	PC	A1	A2	A3	WD	In	A	B	Shamt	A	WD	Mem	A	WD	A	Mem	
addu	NPC.NPC	NPC.NPC		PC.PC			PC.PC	IM.Istr[25:21]	IM.Istr[20:16]	IM.Istr[15:11]	ALU.C		RF.RD1	RF.RD2								
lw	NPC.NPC		PC.PC			PC.PC	IM.Istr[25:21]		IM.Istr[20:16]	MR.Out	IM.Istr[15:0]	RF.RD1	EXT.Out[31:0]					ALU.C			DM.RD	
lb	NPC.NPC		PC.PC			PC.PC	IM.Istr[25:21]		IM.Istr[20:16]	MR.Out	IM.Istr[15:0]	RF.RD1	EXT.Out[31:0]					ALU.C		ALU.C	DM.RD	
sw	NPC.NPC		PC.PC			PC.PC	IM.Istr[25:21]	IM.Istr[20:16]			IM.Istr[15:0]	RF.RD1	EXT.Out[31:0]			RF.RD2		ALU.C	MW.Out			
sb	NPC.NPC		PC.PC			PC.PC	IM.Istr[25:21]	IM.Istr[20:16]			IM.Istr[15:0]	RF.RD1	EXT.Out[31:0]		ALU.C	RF.RD2	DM.RD	ALU.C	MW.Out			
addiu	NPC.NPC		PC.PC			PC.PC	IM.Istr[25:21]		IM.Istr[20:16]	ALU.C	IM.Istr[15:0]	RF.RD1	EXT.Out[31:0]									
ori	NPC.NPC		PC.PC			PC.PC	IM.Istr[25:21]		IM.Istr[20:16]	ALU.C	IM.Istr[15:0]	RF.RD1	EXT.Out[31:0]									
beq	NPC.NPC	IM.Istr[15:0]	PC.PC	ALU.cmp		PC.PC	IM.Istr[25:21]	IM.Istr[20:16]				RF.RD1	RF.RD2									
blez	NPC.NPC	IM.Istr[15:0]	PC.PC	ALU.cmp		PC.PC	IM.Istr[25:21]					RF.RD1										
jr	NPC.NPC		PC.PC		RF.RD1	PC.PC	IM.Istr[25:21]															
jal	NPC.NPC	IM.Istr[25:0]	PC.PC			PC.PC			5’d31	NPC.PC4												
j	NPC.NPC	IM.Istr[25:0]	PC.PC			PC.PC																
sll	NPC.NPC		PC.PC			PC.PC		IM.Istr[20:16]	IM.Istr[15:11]	ALU.C		RF.RD2		IM.Istr[10:6]								
slt	NPC.NPC		PC.PC			PC.PC	IM.Istr[25:21]	IM.Istr[20:16]	IM.Istr[15:11]	ALU.C		RF.RD1	RF.RD2									
syn		NPC.NPC	IM.Istr[25:0]	PC.PC	ALU.cmp	RF.RD1	PC.PC	IM.Istr[25:21]	IM.Istr[20:16]	IM.Istr[15:11] IM.Istr[20:16] 5’d31	ALU.C MR.Out NPC.PC4	IM.Istr[15:0]	RF.RD1 RF.RD2	RF.RD2 EXT.Out[31:0]	IM.Istr[10:6]	ALU.C	RF.RD2	DM.RD	ALU.C	RF.RD2 MW.Out	ALU.C	DM.RD

2 测试方案

通过手造数据，现成数据，自动生成数据相结合的方式生成数据，并利用python脚本与mars对拍实现自动化测试

2.1 测试程序生成器

```
#include <cstdio>
#include <algorithm>
#include <queue>
#include <map>
#include <cstring>
#include <cmath>
#include <cstdlib>
#include <set>
#include <unordered_map>
#include <vector>
#include <ctime>
#define maxn 1010
typedef long long ll;
using namespace std;
unsigned int grf[32];
int reg[] = {0, 1, 2, 3, 4, 5, 30, 31, 29};
int dm[1024];
#define R reg[rand() % 8]
#define I (rand() + rand())
#define B (rand() % 30)
void addu(int rs, int rt, int rd)
{
    printf("addu $%d,$%d,$%d\n", rd, rt, rs);
    if (rd)
        grf[rd] = grf[rs] + grf[rt];
}
void _and(int rs, int rt, int rd)
{
    printf("and $%d,$%d,$%d\n", rd, rt, rs);
    if (rd)
        grf[rd] = grf[rs] & grf[rt];
}
void subu(int rs, int rt, int rd)
{
    printf("subu $%d,$%d,$%d\n", rd, rt, rs);
    if (rd)
```

```
        grf[rd] = grf[rs] - grf[rt];
}
void sll(int rs, int rt, int rd)
{
    int s = rand()%31;
    printf("sll $%d,$%d,%d\n", rd, rt, s);
    if (rd)
        grf[rd] = grf[rt] << s;
}
void slt(int rs, int rt, int rd)
{
    printf("slt $%d,$%d,$%d\n", rd, rs, rt);
    if (rd)
        grf[rd] = (grf[rs] < grf[rt]);
}
void ori(int rs, int rt, int imm)
{
    printf("ori $%d,$%d,%d\n", rt, rs, imm);
    if (rt)
        grf[rt] = grf[rs] | imm;
}
void lui(int rs, int rt, int imm)
{
    printf("lui $%d,%d\n", rs, imm);
    if (rs)
        grf[rs] = 1u * imm << 16;
}
void addiu(int rs, int rt, int imm)
{
    imm = rand()%65535;
    printf("addiu $%d,$%d,%d\n", rs, rt, imm);
    if (rs)
        grf[rs] = grf[rt] + imm;
}
void lw(int rs, int rt)
{
    int imm = rand() % 31 * 4;
    printf("lw $%d,%d($0)\n", rt, imm);
    grf[rt] = dm[imm / 4];
}
void sw(int rs, int rt)
{
    int imm = rand() % 31 * 4;
    printf("sw $%d,%d($0)\n", rt, imm);
    dm[imm / 4] = grf[rt];
}
void lb(int rs, int rt)
{
    int imm = rand() % 127;
    printf("lb $%d,%d($0)\n", rt, imm);
    int byte = imm%4;
    int mask = 0;
    for(int i=8*byte;i<=7+8*byte;i++) mask |= (1<<i);
    grf[rt] = dm[imm / 4] & mask;
}
void sb(int rs, int rt)
{
    int imm = rand() % 127;
    printf("sb $%d,%d($0)\n", rt, imm);
}

int jump[1010];
void beq(int rs, int rt, int k)
{
    int jaddr = k + rand()%50+1;
    while (jump[jaddr]||jaddr>=maxn)
        jaddr = k + rand()%50+1;
    printf("beq $%d,$%d,label%d\n", rs, rt, jaddr);
}
void j(int k)
{
    int jaddr = k + rand()%50+1;
    while (jump[jaddr]||jaddr>=maxn)
        jaddr = k + rand()%50+1;
    printf("j label%d\n", jaddr);
}
void jal(int k)
{
    int jaddr = k + rand()%50+1;
    while (jump[jaddr]||jaddr>=maxn)
        jaddr = k + rand()%50+1;
    printf("jal label%d\n", jaddr);
}
void jalr(int rd, int rs)
{
    printf("jalr $%d, $%d\n", rd, rs);
}
int jr(int rs, int rt, int k)
{
    int i;
    vector<int> can;
    can.clear();
    for (i = 0; i < 10; i++)
        if (reg[i] > (0x3000+(k<<2)) && reg[i] < 0x3000+((k+50)<<2) && reg[i] <
0x3000+(maxn<<2))
            can.push_back(reg[i]);
    if (can.size() == 0)
    {
        beq(rs, rt, k);
        return 0;
    }
    rs = can[rand() % can.size()];
    printf("jr $%d\n", rs);
    return 1;
}
void nop()
{
    printf("nop\n");
}
int main()
{
    int i;
    srand(time(NULL));
    freopen("test.asm", "w", stdout);
    printf("subu $31,$31,$31\n"); //���$sp
    int last = -1;
    for (i = 0; i < maxn; i++)
    {
        printf("label%d: ", i);
        int instr = rand() % 16;
        while ((i < 90 || last == 1) && instr >= 6 && instr <= 9)
        { //j+j
            instr = rand() % 16;
        }
        int rs = R, rt = R, rd = R, imm = I;
        if (instr == 0)
            addu(rs, rt, rd);
        else if (instr == 1)
            subu(rs, rt, rd);
        else if (instr == 2)
            ori(rs, rt, imm);
        else if (instr == 3)
            lui(rs, 0, imm);
        else if (instr == 4)
            lw(rs, rt);
        else if (instr == 5)
            sw(rs, rt);
        else if (instr == 6)
            beq(rs, rt, i);
        else if (instr == 7)
            j(i);
        else if (instr == 8)
            jal(i);
        else if (instr == 14)
            _and(rs, rt, rd);
        else if (instr == 10)
            sll(rs, rt, rd);
        else if (instr == 11)
            slt(rs, rt, rd);
        else if (instr == 12)
            lb(rs, rt);
        else if (instr == 13)
            sb(rs, rt);
        //else if (instr == 15)
        //  jalr(rd, rs);
        else if (instr == 9)
        {
            int yes = jr(rs, rt, i);
            if (!yes)
                instr = 6; //beq
        }
        else
            nop();
        jump[i] = last = (instr >= 6 && instr <= 9);
    }
    //printf("label:\n beq $0,$0,label");
    return 0;
}
```

2.2 自动评测程序

```python
import os
import re
import random


machine=[]
hex_to_bi={"0":"0000","1":"0001","2":"0010","3":"0011",
           "4":"0100","5":"0101","6":"0110","7":"0111",
           "8":"1000","9":"1001","a":"1010","b":"1011",
           "c":"1100","d":"1101","e":"1110","f":"1111"}
reg={"0":"$0",  "1":"$at", "2":"$v0", "3":"$v1",
     "4":"$a0", "5":"$a1", "6":"$a2", "7":"$a3",
     "8":"$t0", "9":"$t1", "10":"$t2", "11":"$t3",
     "12":"$t4", "13":"$t5", "14":"$t6", "15":"$t7",
     "16":"$s0", "17":"$s1", "18":"$s2", "19":"$s3",
     "20":"$s4", "21":"$s5", "22":"$s6", "23":"$s7",
     "24":"$t8", "25":"$t9", "26":"$k0", "27":"$k1",
     "28":"$gp", "29":"$sp", "30":"$fp", "31":"$ra"}
def bi_to_hex(a):
    bcode = ""
    for char in a:
        if not(char==" "):
            bcode += char
    return hex(int(str(int(bcode,2))))


def dasm(hexcode):
    out=["" for i in range(200)]
    labelcount=1
    label={}
    mipscount=0

    bicode=""
    for char in hexcode:
        bicode += hex_to_bi[char]

    op=bicode[0:6]
    func=bicode[26:32]
    rs=reg[str(int(bicode[6:11],2))]
    rt=reg[str(int(bicode[11:16],2))]
    rd=reg[str(int(bicode[16:21],2))]
    shamt=bicode[21:26]
    imm=bi_to_hex(bicode[16:32])
    mips=""

    if op=='000000':
        itype="R"
    elif op=='000010' or op=='000011':
        itype="J"
    else:
        itype="I"

    if itype=="J":
        if op=='000010':
            mips="j "
        elif op=='000011':
            mips="jal "
        mips += imm

    elif itype=="R":
        if bicode=='00000000000000000000000000000000':
            mips="nop"
        elif func=='100000':
            mips="add "+rd+", "+rs+", "+rt
        elif func=='100001':
            mips="addu "+rd+", "+rs+", "+rt
        elif func=='100100':
            mips="and "+rd+", "+rs+", "+rt
        elif func=='001101':
            mips="break"
        elif func=='011010':
            mips="div "+rs+", "+rt
        elif func=='011011':
            mips="divu "+rs+", "+rt
        elif func=='001001':
            mips="jalr "+rd+", "+rs
        elif func=='001000':
            mips="jr "+rs
        elif func=='010000':
            mips="mfhi "+rd
        elif func=='010010':
            mips="mflo "+rd
        elif func=='010001':
            mips="mthi "+rd
        elif func=='010011':
            mips="mtlo "+rd
        elif func=='011000':
            mips="mult "+rs+", "+rt
        elif func=='011001':
            mips="multu "+rs+", "+rt
        elif func=='100111':
            mips="nor "+rd+", "+rs+", "+rt
        elif func=='100101':
            mips="or "+rd+", "+rs+", "+rt
        elif func=='000000':
            mips="sll "+rd+", "+rt+", "+shamt
        elif func=='000100':
            mips="sllv "+rd+", "+rt+", "+rs
        elif func=='101010':
            mips="slt "+rd+", "+rs+", "+rt
        elif func=='101011':
            mips="sltu "+rd+", "+rs+", "+rt
        elif func=='000011':
            mips="sra "+rd+", "+rt+", "+shamt
        elif func=='000111':
            mips="srav "+rd+", "+rt+", "+rs
        elif func=='000010':
            mips="srl "+rd+", "+rt+", "+shamt
        elif func=='000110':
            mips="srlv "+rd+", "+rt+", "+rs
        elif func=='100010':
            mips="sub "+rd+", "+rs+", "+rt
        elif func=='100011':
            mips="subu "+rd+", "+rs+", "+rt
        elif func=='001100':
            mips="syscall"
        elif func=='100110':
            mips="xor "+rd+", "+rs+", "+rt

    elif itype=="I":
        if op=='001000':
            mips="addi "+rt+", "+rs+", "+imm
        elif op=='001001':
            mips="addiu "+rt+", "+rs+", "+imm
        elif op=='001100':
            mips="andi "+rt+", "+rs+", "+imm
        elif op=='000100':
            mips="beq "+rs+", "+rt+", "+imm
        elif op=='000001' and bicode[11:16]=='00001':
            mips="bgez "+rs+", "+imm
        elif op=='000111':
            mips="bgtz "+rs+", "+imm
        elif op=='000110':
            mips="blez "+rs+", "+imm
        elif op=='000001' and bicode[11:16]=='00000':
            mips="bltz "+rs+", "+imm
        elif op=='000101':
            mips="bne "+rs+", "+rt+", "+imm
        elif op=='010000' and func=='011000':
            mips="eret"
        elif op=='100000':
            mips="lb "+rt+", "+imm+"("+rs+")"
        elif op=='100100':
            mips="lbu "+rt+", "+imm+"("+rs+")"
        elif op=='100001':
            mips="lh "+rt+", "+imm+"("+rs+")"
        elif op=='100101':
            mips="lhu "+rt+", "+imm+"("+rs+")"
        elif op=='001111':
            mips="lui "+rt+", "+imm
        elif op=='100011':
            mips="lw "+rt+", "+imm+"("+rs+")"
        elif op=='010000' and bicode[6:11]=='00000':
            mips="mfc0 "+rt+", "+rd
        elif op=='010000' and bicode[6:11]=='00100':
            mips="mtc0 "+rt+", "+rd
        elif op=='001101':
            mips="ori "+rt+", "+rs+", "+imm
        elif op=='101000':
            mips="sb "+rt+", "+imm+"("+rs+")"
        elif op=='101001':
            mips="sh "+rt+", "+imm +"("+rs+")"
        elif op=='001010':
            mips="slti "+rt+", "+rs+", "+imm
        elif op=='001011':
            mips="sltiu "+rt+", "+rs+", "+imm
        elif op=='101011':
            mips="sw "+rt+", "+imm+"("+rs+")"
        elif op=='001110':
            mips="xori "+rt+", "+rs+", "+imm
    out[mipscount] += mips
    mipscount += 1
    return out[0]

for testflie in range(900,1300):
    #asmfilename=("test1.asm")
    asmfilename=("testpoint%d.asm"%(testflie))
    xlinx="D:\\PROGRAM\\14.7\\ISE_DS\\ISE"
    time="10us"
    os.environ['XILINX']=xlinx
    path=os.path.dirname(os.path.realpath(__file__))
    os.chdir(path)
    filelist=os.walk(path)
    with open("mips.prj","w") as prj:
        for folder in filelist:
            for file in folder[2]:
                if(len(file.split("."))>1 and file.split(".")[1]=="v"):
                    prj.write("verilog work \""+folder[0]+"\\"+file+"\"\n")
    with open("mips.tcl","w") as tcl:
        tcl.write("run "+time+";\nexit;\n")

    print("start running point%d"%(testflie))
        #"java -jar Mars.jar test.asm nc mc CompactTextAtZero a dump .text
HexText "+rom_name
        # problem: can not exit mars
    os.system("java -jar Mars.jar db a nc mc CompactDataAtZero dump .text
HexText data0.txt 1000000 "+asmfilename)
    os.system("java -jar Mars.jar db nc mc CompactDataAtZero dump .text HexText
data0.txt >out_std.txt 1000000 "+asmfilename)
    #print("std done")
    os.system(xlinx+"\\bin\\nt64\\fuse "+"--nodebug  --prj mips.prj -o mips.exe
mips_tb >log.txt")
    os.system("mips.exe -nolog -tclbatch mips.tcl >out_source.txt")
    #print("source done")

    process=0
    with open("out_source.txt","r") as my:
        lines=my.readlines()
        if(len(lines)==0):
            print("fail to simulate")
            os._exit(1)
        if(lines[0][0]=='I'):
            process=1
    n=0
    while(1):
        if(lines[n][9]=="@"):
            break
        else:
            n=n+1
    if(process):
        with open("out_source.txt","w") as my:
            my.writelines(lines[n:])
    i=0
    biao=0
    instr = open("Instr.txt","r")
    with open("out_source.txt","r") as source:
        with open("out_std.txt","r") as std:
            while(1):
                i+=1
                l1=source.readline().strip()
                l2=std.readline().strip()
                Instr = l1[0:8]
                asm = dasm(Instr)
                l1 = l1[9:]
                if((l1== "" or l1==None) or (l2=="" or l2==None)):
                    break
                elif l1==l2:
                    continue
                    print("AC at line:%d "%(i)+"    source::"+l1+"  asm::"+asm)
                elif l1!=l2 and not "$ 0"in l2 and not "$ 0" in  l1:
                    biao=1
                    print("WA at line:%d "%(i)+"    source::"+l1+"  std::"+l2+"
asm::"+asm)
                    '''
                    if l2=="" or l2 == None:
                        print("Wrong answer occur in line %d of code: "%(i)+"we
got "+l1+" when we expected Nothing")
                    else:
                        print("Wrong answer occur in line %d of code: "%(i)+"we
got "+l1+" when we expected "+l2)
                    '''
    if biao==0:
        print("Accepted on the point")
    else:
        print("WA at testflie::%d"%(testflie))
        os._exit(1)
```

2.3 测试数据及结果

手动构造数据

```
ori $a0,$0,1999
ori $a1,$a0,111
lui $a2,12345
lui $a3,0xffff
lui $t0,0xffff
beq $a3,$t0,eee
addu $s7,$0,$a0
nop
ori $a3,$a3,0xffff
addu $s0,$a0,$a1
addu $s1,$a3,$a3
addu $s2,$a3,$s0
beq $s2,$s3,eee
subu $s0,$a0,$s2
subu $s1,$a3,$a3
eee:
subu $s2,$a3,$a0
subu $s3,$s2,$s1
ori $t0,$0,0x0000
sw $a0,0($t0)
nop
sw $a1,4($t0)
sw $s0,8($t0)
sw $s1,12($t0)
sw $s2,16($t0)
sw $s5,20($t0)
lw $t1,20($t0)
lw $t7,0($t0)
lw $t6,20($t0)
sw $t6,24($t0)
lw $t5,12($t0)
jal end
ori $t0,$t0,1
ori $t1,$t1,1
ori $t2,$t2,2
beq $t0,$t2,eee
lui $t3,1111
jal out
end:
addu $t0,$t0,$t7
jr $ra
out:
addu $t0,$t0,$t3
ori $t2,$t0,0
beq $t0,$t2,qqq
lui $v0,10
qqq:
lui $v0,11
j www
nop
www:
lui $ra,100
```

```
@00003000: $31 <= 00000000
@00003004: $ 1 <= 00000001
@00003008: $ 2 <= 00000002
@0000300c: $ 3 <= 00000003
@00003010: $ 4 <= 00000004
@00003014: $ 5 <= 00000005
@00003018: $ 6 <= 00000006
@0000301c: $ 4 <= 00000000
@00003020: $ 6 <= 00000000
@00003024: $ 1 <= 00000000
@00003028: $31 <= 80000000
@0000302c: *00000008 <= 00000000
@00003030: *00000028 <= 00000000
@00003034: *0000000c <= 00000000
@00003038: $11 <= 00000001
@0000303c: $ 7 <= 00000000
@00003050: $20 <= 0000b343
@00003054: $31 <= 00003058
@0000305c: *00000040 <= 00003058
@00003060: *00000043 <= 00000000
@00003064: $22 <= 00003058
@00003068: $23 <= 00000001
@0000306c: $ 1 <= 00000000
@00003070: $ 1 <= 00000045
@00003074: $28 <= 00000045
@00003078: $24 <= 00000000
@00003058: $ 2 <= 11900000
@0000305c: *00000040 <= 00003058
```

自动生成数据(节选)

```
subu $31,$31,$31
label0: lw $0,12($0)
label1: sb $4,1($0)
label2: sb $5,82($0)
label3: and $3,$3,$30
label4: lui $31,50940
label5: slt $30,$1,$5
label6: lb $1,71($0)
label7: addu $31,$0,$4
label8: lw $0,52($0)
label9: lw $2,56($0)
label10: lw $31,68($0)
label11: sw $0,24($0)
label12: lw $4,48($0)
label13: lw $31,16($0)
label14: lb $0,33($0)
label15: and $2,$5,$3
label16: slt $30,$1,$5
label17: lw $5,4($0)
label18: addu $30,$4,$5
label19: and $31,$30,$30
label20: lb $2,120($0)
label21: sll $1,$31,15
label22: nop
label23: ori $31,$2,28285
label24: sw $0,0($0)
label25: lui $4,21443
label26: subu $3,$4,$31
label27: sll $2,$0,8
label28: and $4,$4,$0
label29: sb $31,10($0)
label30: lb $31,79($0)
label31: sb $0,49($0)
label32: sb $5,123($0)
label33: lw $5,60($0)
label34: ori $5,$0,16741
label35: addu $4,$31,$2
label36: sll $4,$31,10
label37: sw $31,96($0)
label38: nop
label39: sb $1,24($0)
label40: lw $31,104($0)
label41: sw $1,32($0)
label42: addu $30,$0,$31
label43: subu $2,$3,$4
label44: lw $2,20($0)
label45: sw $31,48($0)
label46: sb $3,3($0)
label47: lui $3,33341
label48: nop
label49: lui $1,36188
label50: and $0,$31,$5
label51: sb $30,87($0)
label52: addu $0,$30,$31
label53: subu $1,$5,$4
```

```
label54: slt $0,$30,$30
label55: sll $5,$1,9
label56: and $0,$3,$0
label57: lw $5,116($0)
label58: and $1,$2,$30
label59: nop
label60: lui $2,54044
label61: slt $1,$31,$1
label62: lui $30,31679
label63: slt $5,$0,$3
label64: slt $2,$5,$2
label65: slt $4,$4,$30
label66: addu $3,$0,$2
label67: sw $30,72($0)
label68: lw $3,92($0)
label69: sb $30,42($0)
label70: subu $31,$5,$31
label71: lui $1,33153
label72: addu $30,$5,$3
label73: sw $2,76($0)
label74: lui $3,19816
label75: addu $5,$30,$3
label76: sw $30,84($0)
label77: ori $5,$2,16786
label78: addu $3,$0,$1
label79: slt $1,$30,$31
label80: lb $31,40($0)
label81: slt $2,$0,$2
label82: sll $3,$4,8
label83: slt $2,$5,$5
label84: addu $0,$2,$4
label85: addu $4,$5,$4
label86: lw $0,8($0)
label87: nop
label88: lui $3,9217
label89: ori $31,$4,48523
label90: jal label116
label91: sw $2,32($0)
label92: and $4,$4,$2
label93: addu $5,$3,$4
label94: ori $0,$2,10897
label95: sw $31,36($0)
label96: beq $30,$30,label98
label97: addu $5,$2,$2
label98: sll $1,$30,17
label99: and $0,$1,$2
label100: subu $30,$2,$31
label101: nop
label102: sw $31,48($0)
label103: beq $5,$31,label109
label104: ori $2,$2,43182
label105: sw $3,76($0)
label106: lb $5,25($0)
label107: subu $2,$31,$5
label108: slt $5,$30,$31
label109: and $5,$5,$2
label110: sw $31,100($0)
label111: addu $30,$3,$31
```

```
label112: sb $3,84($0)
label113: ori $1,$2,14981
label114: slt $3,$1,$4
label115: beq $2,$5,label164
label116: ori $0,$3,10078
label117: sb $1,93($0)
label118: subu $3,$3,$4
label119: nop
label120: jal label151
label121: slt $2,$5,$3
label122: subu $1,$31,$31
label123: jal label170
label124: sw $1,28($0)
label125: ori $4,$31,22347
label126: subu $2,$30,$4
label127: subu $1,$30,$31
label128: beq $3,$0,label148
label129: ori $31,$31,42068
label130: slt $2,$30,$4
label131: and $30,$1,$5
label132: subu $4,$4,$4
label133: sw $5,104($0)
label134: subu $1,$2,$0
label135: sb $5,71($0)
label136: lb $4,29($0)
label137: lui $30,48512
label138: subu $2,$2,$0
label139: j label163
label140: lb $5,1($0)
label141: jal label191
label142: sb $1,95($0)
label143: jal label158
label144: lw $31,60($0)
label145: ori $4,$2,6859
label146: nop
label147: nop
label148: lui $5,57412
label149: and $30,$31,$30
label150: jal label153
label151: sll $2,$1,0
label152: nop
label153: ori $1,$31,33784
label154: slt $3,$1,$5
label155: nop
label156: lui $4,10869
label157: lui $3,46482
label158: lb $5,7($0)
label159: lui $2,22265
label160: beq $2,$4,label192
label161: slt $0,$4,$5
label162: addu $5,$5,$2
label163: ori $1,$5,45108
label164: sll $5,$5,21
label165: lb $2,68($0)
label166: beq $2,$0,label185
label167: lb $1,59($0)
label168: j label185
label169: lw $5,108($0)
```

```
label170: lb $3,30($0)
label171: nop
label172: addu $30,$3,$0
label173: sb $2,88($0)
label174: beq $30,$0,label206
label175: lui $30,42464
label176: slt $30,$2,$30
label177: beq $3,$4,label224
label178: nop
label179: slt $4,$3,$1
label180: nop
label181: sll $31,$1,7
label182: sll $4,$30,18
label183: lui $4,41133
label184: subu $2,$30,$4
label185: j label188
label186: lui $31,55006
label187: jal label218
label188: sb $3,53($0)
label189: jal label235
label190: and $5,$31,$5
label191: slt $0,$3,$3
label192: sb $30,119($0)
label193: sw $4,4($0)
label194: beq $0,$2,label224
label195: slt $1,$4,$3
label196: sll $5,$0,21
label197: lb $4,81($0)
label198: sll $30,$3,0
label199: and $0,$0,$31
label200: sb $0,52($0)
label201: beq $2,$30,label231
label202: sw $3,80($0)
label203: subu $3,$1,$31
label204: sw $31,76($0)
label205: sw $1,64($0)
label206: addu $31,$2,$5
label207: addu $1,$31,$5
label208: lui $3,6798
label209: sb $4,38($0)
label210: subu $31,$2,$5
label211: addu $5,$4,$30
label212: slt $3,$4,$0
label213: addu $30,$3,$1
label214: nop
label215: ori $4,$3,31888
label216: sb $0,38($0)
label217: beq $1,$3,label263
label218: subu $4,$30,$3
label219: and $1,$5,$1
label220: lb $30,104($0)
label221: addu $5,$2,$31
label222: sw $4,92($0)
label223: slt $30,$2,$30
label224: ori $31,$5,59111
label225: slt $31,$1,$30
label226: and $0,$4,$3
label227: lw $3,52($0)
```

```
label228: j label241
label229: lui $3,49052
label230: beq $5,$5,label233
label231: lui $1,35736
label232: sw $2,120($0)
label233: jal label258
label234: ori $31,$31,7088
label235: j label257
label236: lb $31,45($0)
label237: beq $2,$5,label273
label238: addu $3,$3,$2
label239: lw $1,0($0)
label240: lui $2,44168
label241: beq $31,$0,label251
label242: subu $3,$0,$0
label243: ori $2,$0,44962
label244: nop
label245: lb $5,0($0)
label246: addu $4,$30,$30
label247: sb $1,61($0)
label248: and $2,$30,$0
label249: jal label282
label250: ori $3,$1,33898
label251: jal label283
label252: lui $3,45278
label253: addu $30,$0,$31
label254: ori $30,$5,7157
label255: ori $2,$4,32300
label256: beq $3,$0,label270
label257: lw $0,16($0)
label258: sw $31,112($0)
label259: lb $3,82($0)
label260: j label286
label261: sw $3,56($0)
label262: beq $1,$5,label297
label263: sw $3,60($0)
label264: sll $30,$1,23
label265: and $0,$1,$2
label266: jal label282
label267: ori $30,$2,58136
label268: beq $5,$5,label300
label269: lw $0,40($0)
label270: lui $2,37604
label271: addu $31,$4,$3
label272: and $3,$2,$5
label273: addu $5,$0,$2
label274: sll $2,$31,14
label275: nop
label276: addu $0,$4,$3
label277: lb $3,65($0)
label278: lui $30,17196
label279: slt $4,$3,$5
label280: addu $0,$0,$5
label281: slt $2,$2,$2
label282: ori $3,$2,39430
label283: addu $0,$0,$3
label284: sll $3,$5,28
label285: sll $2,$4,2
```

```
label286: subu $3,$5,$5
label287: ori $3,$5,31211
label288: lui $1,18630
label289: ori $31,$30,36322
label290: addu $5,$31,$3
label291: slt $1,$5,$31
label292: nop
label293: jal label320
label294: and $3,$0,$2
label295: addu $2,$4,$1
label296: addu $5,$4,$31
label297: addu $5,$1,$30
label298: sb $0,46($0)
label299: sll $2,$3,5
label300: and $30,$4,$1
label301: addu $2,$5,$3
label302: sb $31,66($0)
label303: slt $31,$1,$2
label304: sll $3,$3,27
label305: lw $3,120($0)
label306: sw $31,44($0)
label307: j label349
label308: nop
label309: lui $4,46891
label310: beq $2,$30,label355
label311: addu $2,$0,$3
label312: subu $30,$2,$1
label313: lb $3,85($0)
label314: lw $4,52($0)
label315: jal label361
label316: nop
label317: jal label338
label318: lw $30,72($0)
label319: sll $5,$30,15
label320: ori $5,$2,35032
label321: j label344
label322: slt $30,$31,$2
label323: ori $5,$31,47740
label324: beq $4,$1,label337
label325: addu $4,$31,$1
label326: sb $5,28($0)
label327: jal label353
label328: nop
label329: beq $1,$1,label334
label330: sll $31,$2,11
label331: jal label363
label332: nop
label333: jal label351
label334: sll $30,$4,11
label335: lw $5,108($0)
label336: and $30,$0,$1
label337: sll $31,$1,2
label338: addu $3,$5,$1
label339: sb $2,32($0)
label340: sll $31,$0,11
label341: sll $31,$0,0
label342: lui $0,29255
label343: sw $5,48($0)
```

```
label344: sw $5,24($0)
label345: slt $4,$4,$4
label346: sb $2,111($0)
label347: beq $0,$1,label364
label348: sb $1,61($0)
label349: sw $4,4($0)
label350: subu $0,$30,$4
label351: sw $5,112($0)
label352: lw $2,108($0)
label353: addu $3,$31,$2
label354: lb $30,66($0)
label355: j label377
label356: lw $3,84($0)
label357: subu $30,$30,$5
label358: and $30,$30,$31
label359: sb $1,40($0)
label360: slt $3,$31,$5
label361: nop
label362: and $5,$5,$2
label363: lw $30,48($0)
label364: lui $30,30454
label365: j label405
label366: sb $5,37($0)
label367: lui $2,32833
label368: jal label370
label369: lui $4,34700
label370: sw $31,48($0)
label371: and $31,$0,$4
label372: sb $2,68($0)
label373: nop
label374: lui $31,23199
label375: sb $3,11($0)
label376: addu $0,$5,$2
label377: and $1,$0,$30
label378: sw $1,64($0)
label379: beq $3,$1,label403
label380: lui $3,19353
label381: beq $1,$2,label390
label382: lui $31,17992
label383: beq $31,$31,label413
label384: addu $31,$31,$4
label385: beq $0,$5,label406
label386: nop
label387: beq $1,$2,label426
label388: lb $2,19($0)
label389: lb $1,14($0)
label390: beq $5,$2,label415
label391: addu $0,$4,$5
label392: beq $5,$4,label427
label393: and $4,$3,$31
label394: lb $30,59($0)
label395: lb $31,76($0)
label396: lw $31,4($0)
label397: slt $5,$4,$2
label398: jal label441
label399: addu $1,$2,$4
label400: sw $31,96($0)
label401: subu $2,$5,$4
```

```
label402: sll $1,$30,3
label403: lw $31,64($0)
label404: nop
label405: and $4,$30,$1
label406: jal label440
label407: sll $5,$5,30
label408: j label432
label409: subu $2,$0,$2
label410: slt $5,$3,$5
label411: sb $0,110($0)
label412: nop
label413: sll $31,$2,9
label414: beq $1,$0,label427
label415: addu $2,$0,$30
label416: sw $1,44($0)
label417: sll $1,$2,25
label418: nop
label419: lw $5,92($0)
label420: addu $4,$2,$2
label421: nop
label422: sw $31,116($0)
label423: addu $4,$30,$30
label424: slt $30,$1,$5
label425: sll $0,$1,4
label426: beq $31,$0,label447
label427: sll $30,$5,9
label428: lw $2,68($0)
label429: lb $5,118($0)
label430: sll $0,$2,30
label431: lui $5,55194
label432: addu $30,$5,$4
label433: slt $4,$30,$2
label434: beq $3,$3,label471
label435: subu $4,$30,$30
label436: beq $4,$30,label473
label437: and $1,$31,$4
label438: beq $5,$2,label453
label439: and $30,$4,$31
label440: and $5,$4,$0
label441: beq $5,$1,label444
label442: sw $30,52($0)
label443: nop
```

- 执行结果与mars一致

3 思考题

流水线冒险

1. 在采用本节所述的控制冒险处理方式下，PC的值应当如何被更新？请从数据通路和控制信号两方面进行说明。
   - PC值更新有四种情况，一是指令为beq的时候，根据branch与equal信号控制更新，PC=PC+4+sign_ext(imm16)；二是j或jal指令时，根据jump信号控制更新,PC=sign_ext(imm26)；三是jr 指令时，根据pcSrc和jump信号一起控制更新PC=gpr[rs]；四，其余指令，则PC=PC+4。

2. 对于jal等需要将指令地址写入寄存器的指令，为什么需要回写PC+8?
    - PC + 4 的位置是延迟槽，跳回时应该跳到延迟槽下一条指令，即 PC + 8。

数据冒险的分析

1. 为什么所有的供给者都是存储了上一级传来的各种数据的**流水级寄存器**，而不是由ALU或者DM等部件来提供数据?
    - 此次CPU采用的是流水线方法构造，每一级流水线寄存器存储对应流水级的指令以及相关数据值，在时 钟上升沿到来时，将相关数据送入对应流水级供该级使用，流水线寄存器起到隔断作用，让每个流水级 中的数据不会受到上一个流水级的数据通路的影响。

AT法处理流水线数据冒险

1. “转发（旁路）机制的构造”中的Thinking 1-4；
    1. 如果不采用已经转发过的数据，而采用上一级中的原始数据，会出现怎样的问题？试列举指令序列说明这个问题。
        - 将会出现已经修改过的寄存器数值，在之后的指令中使用了未被更改过的原始数值，导致运行结果错误 的情况。例如：

          ```
          ori $1, $0, 1
          nop
          nop
          nop
          nop
          lw $1, 0($0)
          sw $1, 4($0)
          ```

          这样 `sw` 指令就会把 1 存到 DM 中。
    2. 我们为什么要对GPR采用内部转发机制？如果不采用内部转发机制，我们要怎样才能解决这种情况下的转发需求呢?
        - 为了防止W级还未写入GRF的数据在之后的指令中使用导致运行结果出现错误。
        - 如果不采用内部转发机 制，可以采用将即将从W级写入寄存器文件的数据端口连到cmp输入端口的转发端来解决。
    3. 为什么0号寄存器需要特殊处理?
        - 因为指令可以对 0 号寄存器赋值，只是不会造成实际作用，但是转发过程中如果不特判就默认 0 号寄存器的值被更改了，从而造成错误。
    4. 什么是“最新产生的数据”?
        - 数据的“新旧”是通过看转发处距D级/E级/M级（所需要的地方）的距离远近来判断，距离所需要的地方 越近，则越新。
2. 在AT方法讨论转发条件的时候，只提到了“供给者需求者的A相同，且不为0”，但在CPU写入GRF的时候，是有一个we信号来控制是否要写入的。为何在AT方法中不需要特判we呢？为了**用且仅用**A和T完成转发，在翻译出A的时候，要结合we做什么操作呢?
    - AT法使建立了指令和对应流水级到相应的A和T信号的映射，之后比较需要的GRF地址和待转发的数据值 是否相等且需要的地址不为0，we是控制是否向GRF写值，AT法中，如果采用当we为0，将A（供给者） 设为0，那么转发条件将不会满足，在此情况下则不需要特判we。

在线测试相关说明

1. 在本实验中你遇到了哪些不同指令类型组合产生的冲突？你又是如何解决的？相应的测试样例是什么样的？

   如果你是手动构造的样例，请说明构造策略，说明你的测试程序如何保证**覆盖**了所有需要测试的情况；如果你是**完全随机**生成的测试样例，请思考完全随机的测试程序有何不足之处；如果你在生成测试样例时采用了**特殊的策略，**比如构造连续数据冒险序列，请你描述一下你使用的策略如何**结合了随机性**达到强测的效果。

   此思考题请同学们结合自己测试CPU使用的具体手段，按照自己的实际情况进行回答。

   - 主要是数据冒险和控制冒险，分别通过暂停转发以及比较前移+延迟槽解决。
   - 数据生成器采用了特殊策略：单组数据中除了 0 和 31 号寄存器外，至多涉及 3 个寄存器。一方面，这样产生的代码中，邻近的指令几乎全部都存在数据冒险，可以充分测试转发和暂停；另一方面，当测试数据的组数一定多，几乎涉及了每个寄存器，避免了只测试部分寄存器。此外，所有跳转指令都是特殊构造的，不会进入死循环的同时如果跳转出错可以输出中体现。