编号66

南京航空航天大学

课设报告

题 目 五级流水线处理器设计

学生姓名 李应飞

学 号 161610338

学 院 计算机科学与技术学院

专 业 计算机科学与技术

班 级 1616103

指导教师 施慧彬 副教授

二〇一八年七月

单周期MIPS CPU的设计

摘 要

为了提升对单周期 MIPS CPU 的理解能力和动手能力，以个人为单位，进一步在9条指令扩展成36条指令，然后完成一个五级流水线处理器设计

研究的结果与主要结论：完成36条指令的单周期，以及五级流水线的初步开发。开发流水线需要对流水线具有深刻的认识，并且能够处理各种冒险。

关键词：流水线，处理器，汇编语言，模拟，冒险

The Design of a Single Cycle CPU in MIPS

Instruction Set

Abstract

To improve the understanding single-cycle MIPS CPU and ability of practice, complete a single cycle CPU in MIPS instruction set in personal. Finally，trying to design 36 instrction’s pipeline.

The results of the study and experiment: To expand to 36 instrction.make a five degree pipeline and know how it to work.Apart fromthese,we mustbe to handle the hazard.

Key Words：pipeline, processor, CPU, assembly language, simulation ，hazard

目 录

摘 要 .................................................................... ⅰ Abstract ................................................................... ⅰ 第一章 引 言 ............................................................. 1

1.1 单周期CPU的理论结构 ................................................ 1

1.1.1 PC模块定义.................................................... 1

1.1.2 NPC模块定义................................................... 2

1.1.3 ALU模块定义................................................... 3

1.1.4 MUXtip模块定义.................................................4

1.1.5 registerFile模块定义...........................................4

1.1.6InstructionMemory模块定义...................................... 5

1.1.7 dataMemory模块定义................................... ........ 5

1.1.8 contrl模块定义.................................................6

1.1.9 signext模块定义............................................... 7

1.2 各个指令与控制信号之间的关系 ........................................ 8

1.2.1 R-type指令与控制信号.......................................... 8

1.2.2 I-type指令与控制信号.......................................... 8

1.2.3 Load指令与控制信号............................................ 9

1.2.4 Store指令与控制信号........................................... 9

1.2.5 分支指令与控制信号 ........................................... 10

1.2.6 各指令控制信号取值 ........................................... 10 第二章 单周期MIPS CPU的具体设计与调试 ................................... 12

2.1 具体Verilog代码实现 ............................................... 12

2.2 ModelSim模拟及测试 ................................................. 23

2.2.1 综合测试用代码 ............................................... 23

第三章 总结与展望 ........................................................ 30 参考文献 .................................................................. 31 附录1和2 .............................................................. . 31

# 引 言

* 1. 指令实现

实现指令如下：

addu, addi, subu, slt, and, nor, or, xor, sll, srl, addiu, beq, bne, lw, sw, lui, j, sltu, jalr, jr, rs, sllv, sra, srav, srlv, slti, sltiu, bgez, bgtz, blez, bltz, lb, lbu, sb, andi, ori, xori, jal共36条指令

* 1. 完成情况

1. 写了36条指令，并且初步尝试了五级流水线设计。由于时间关系，并没有成功实现五级流水线设计。但有了一定的构想，已经写了三个阶段。
2. 数据通路：
   1. 流水线设计说明

处理了一条指令的执行过程被分成五个阶段，每个阶段由相应的功能部件完成。如果将各阶段看成相应的流水段，则指令的执行过程就构成了一条指令流水线。

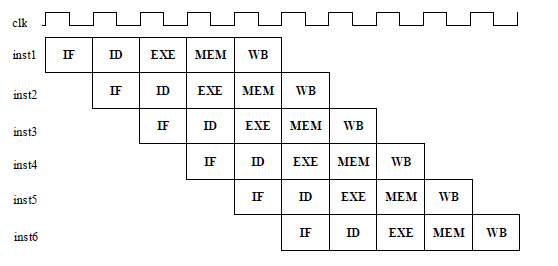
（1）Ifetch (取指) :取指令并计算PC+4 。

（2）Reg/Dec (取数和译码) : 从寄存器取数，同时指令在译码器进行译码 。

（3）Exec (执行) : 在ALU中对操作数进行计算，计算内存单元地址 。

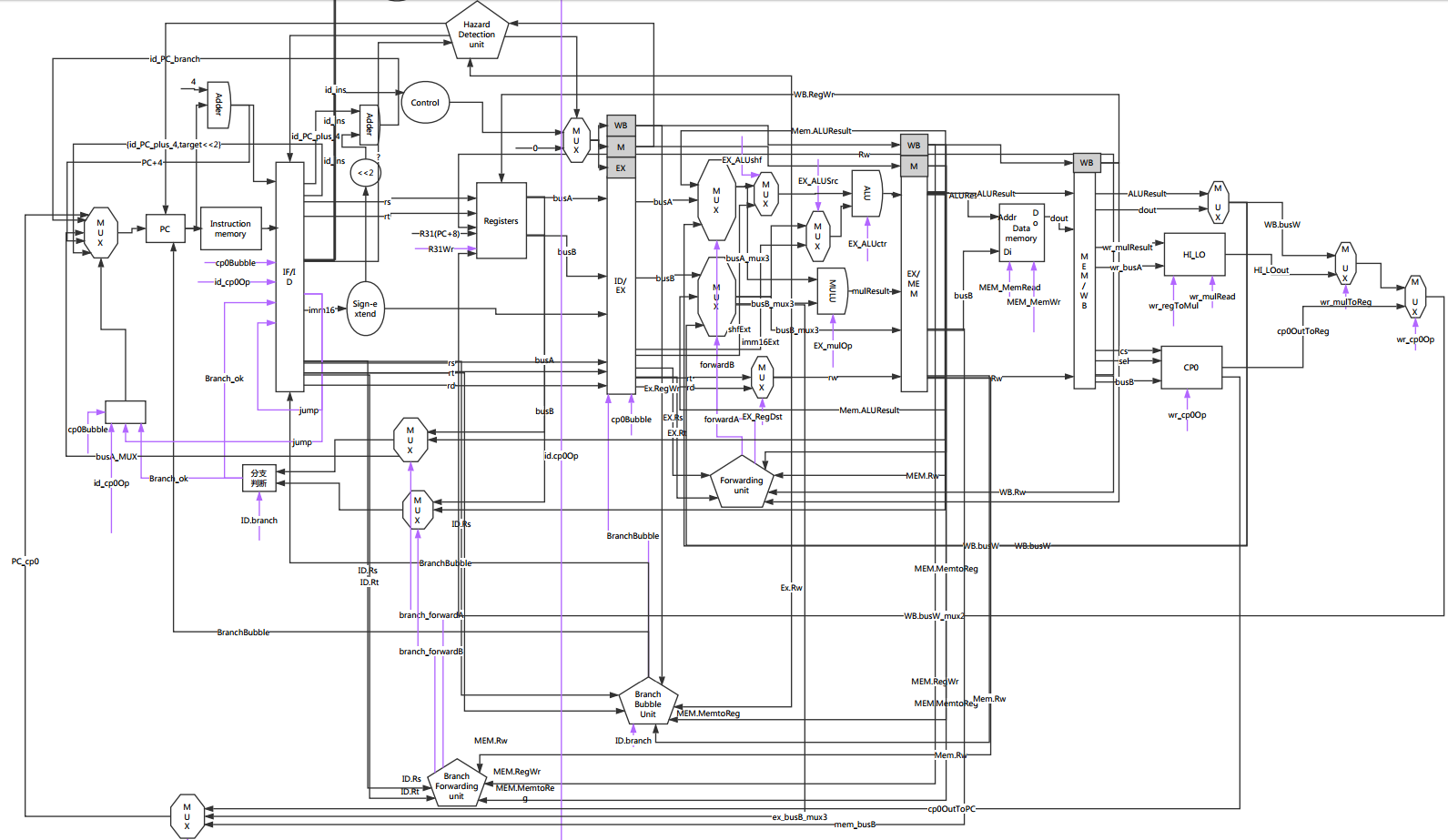
（4）Mem (读存储器) : 从数据存储器中读数据 。

（5）Wr(写寄存器): ALU计算的结果写到寄存器。



# 模块化和层次化设计说明

* 1. 总电路设计图



* 1. 模块化设计图

# 

## 1.1 单周期CPU的理论结构（36条指令）

单周期处理器的数据通路结构如图所示。

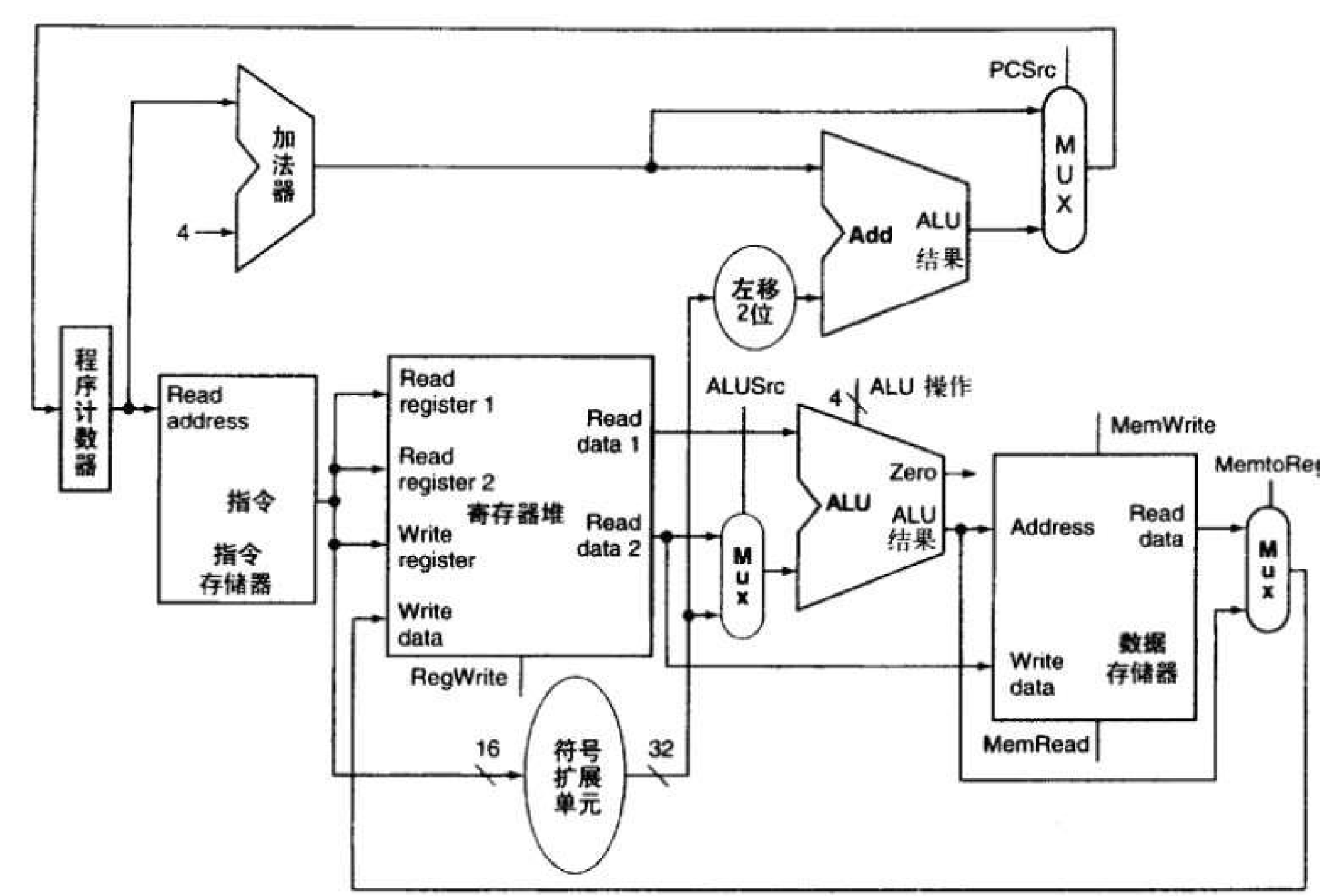


图1.1 单周期处理器的数据通路

### 1.1.1 PC模块定义

1. 基本描述

PC 主要功能是完成输出当前指令地址。复位后，PC指向0x0000\_3000，此处为第

一条指令的地址。

1. 模块接口

表1.1 PC接口的模块定义

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| [31:0] newAddress |  | 输入的指令地址 |
| CLK |  | 时钟信号 |
| Reset |  | 复位信号 |
| [31:0] currentAddress  pcW |  | 输出的指令地址  使其一直能够有pc产生（可无） |
| （3）功能定义 | 表1.2 PC接口的功能定义 |  |
| 序号 | 功能名称 | 功能描述 |
| 1 | 复位 | Reset=1时，将pc置为0X0000\_3000 |
| 2 | 输出指令地址 | 时钟信号到来时，将newAddress赋给currentAddress |

### 1.1.2 NPC模块定义

1. 基本描述

NPC主要功能是根据当前指令是否为beq指令，输出下一条指令的地址。该模块调用了MUX模块。

1. 模块接口

表1.3 NPC接口的模块定义

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| Zero | I | ALU计算结果：1表示当前两寄存  器(rs,rt)值相等;0表示不相等。 |
| Branch | I | 是否为beq指令，1为真，0为假。 |
| [31:0]npc | O | 输出下一条指令地址 |
| [15:0]immediate | I | 输入需要扩展的立即数 |
| [25:0]target | I | 输入需要jump到的地址 |
| Jump | I | Jump指令的控制信号 |
| [31:0]pc  Bgez  Bgtz  Blez  Bltz  Jal  Jalr  Jr | I  I  I  I  I  I  I  I | 输入当前指令地址  Bgez指令的控制信号  Bgtz指令的控制信号  Blez指令的控制信号  Bltz指令的控制信号  Jal指令的控制信号  Jalr指令的控制信号  Jr指令的控制信号 |

1. 功能定义

表1.4 NPC接口的功能定义

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | | 功能名称 | 功能描述 |
| 1 | | 输出指令地址 | 根据Zero与Branch的值输出下一条指令的地址 |
| 2 | | 输出指令地址 | 根据Jump,target, Jr,Bgez, Bgtz, Blez, Bltz,Jal,Jalr的值输出下一条指令的地址 |
| 1.1.3 ALU模块定义   1. 基本描述   实现加、减、与、或、小于   1. 模块接口   表1.5 | | 则赋1五种计算。  ALU接口的模块定义 |  |
| 信号名 |  | 方向 | 描述 |
| [2:0]ALUop |  | I | ALU控制信号 |
| [31:0]busA |  | I | rs寄存器的值 |
| [31:0]B  shf |  | I  I | rt寄存器或者立即数扩展后的值  移位值 |
| zero |  | O | A-B=0,则zero=1; 否则zero=0 |
| [31:0]result |  | O | ALU计算输出的结果 |
| （3）功能定义 | 表1.6 | ALU接口的功能定义 |  |
| 序号 |  | 功能名称 | 功能描述 |
| 1 |  | 输出计算结果 | 根据alu控制信号，输出A与B 的计算结果 |
| 2 |  | 输出zero | 若result=0则输出zero=1，否则  输出zero=0 |

### 1.1.4 MUXtip模块定义

1. 基本描述

实现 32 位和 5 位二选一数据选择器。

1. 模块接口

表1.7 MUX接口的模块定义

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| control | I | MUX控制信号 |
| [31:0]in1 | I | MUX输入 |
| [31:0]in2 | I | MUX输入 |
| [31:0]out | I | MUX输出 |
| （3）功能定义 | 表1.8 MUX接口的功能定义 |  |
| 序号 | 功能名称 | 功能描述 |
| 1 | 输出in1或in2的值 | control为1，result= in1 control为0，result= in2 |

### 1.1.5 registerFile模块定义

1. 基本描述

根据输入的两个寄存器地址，输出相应寄存器的值，根据寄存器写信号和寄存器地址，将输入的数据选择写入寄存器。

1. 模块接口

表1.9 registerFile接口的模块定义

信号名

方向

描述

CLK I 时钟信号

[4:0]rt I Rd寄存器地址

[4:0]rs I Rs寄存器地址

[31:0]busA O Rs寄存器值

[31:0]busB O Rt寄存器值

|  |  |  |
| --- | --- | --- |
| WriteReg | I | 写寄存器地址 |
| WriteData | I | 写入寄存器值 |
| RegWr  [31:0] pc  Jalr  Jal | I  I  I  I | 写入寄存器的信号  Pc  跳转信号  跳转信号 |
| （3）功能定义 | 表1.10 REGFILE接口的功能定义 |  |
| 序号 | 功能名称 | 功能描述 |
| 1 | 读寄存器数据 | 读rs、rt，rd寄存器的数据 |
| 2  3 | 向寄存器写入数据  向寄存器存pc | 根据写信号向寄存器选择写数据  根据Jal、Jalr向寄存器存pc |

### 1.1.6 InstructionMemory模块定义

1. 基本描述

初始化从code.txt载入指令。根据输入的指令地址，输出当前位置存储的指令。

1. 模块接口

表1.11 InstructionMemory接口的模块定义

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| [31:0]IAddr  [5:0] op  [4:0] rs  [4:0] rt  [4:0] rd  [4:0] shf  [5:0] func  [15:0] immediate  [25:0] targe | I  I  I  I  I  I  I  I  I | 指令地址  op域  rs段  rt段  rd段  shf段  func段  立即数  Target  段 |
| （3）功能定义 | 表1.12 InstructionMemory接口的功能定义 |  |
| 序号 | 功能名称 | 功能描述 |
| 1 | 载入指令 | 初始化载入code.txt中的指令 |
| 2  3 | 输出指令  指令的划分 | 根据输入指令地址，输出当前指令及指令的划分  观察指令的正确性 |

### 1.1.7 datamemory模块定义

1. 基本描述

根据输入的地址读出数据内存中的数据，并根据数据写信号，将输入的数据选择写入数据内存中。

1. 模块接口

表1.13 dataMemory接口的模块定义

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| CLK | I | 时钟信号 |
| [31:0]Adr | I | 数据地址 |
| [31:0]DataIn | I | 写入的数据 |
| MenWr  \_B | I  I | 数据内存写信号  SB LB LBU信号 |
| [31:0]DataOut | O | 读出的数据 |
| （3）功能定义 | 表1.14 dataMemroy接口的功能定义 |  |
| 序号 | 功能名称 | 功能描述 |
| 1 | 读数据内存数据 | 根据输入的数据地址，读出数据内存的数据，读出的数据不一定被使用，只有lw指令才会使用 |
| 2 | 向数据内存写入数据 | 在时钟信号到来时，根据写数据信号，将输入的数据选择写入数据内存中 |

### 1.1.8 contrl模块定义

1. 基本描述根据输入的指令高 6 位(op 字段)和指令最低6位（func字段），利用真值表化简，输出 Branch, Jump,RegDst, ALUsrc,funcop，aluop, MemtoReg, RegWr, MemWr,ExtOp，pcW，Rtype控制信号。真值表采用书上166页

图4-22，再加上Jump指令的输入输出。

1. 模块接口

表1.15 contrl和control1接口的模块定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 信号名 | | 方向 | | 描述 | |
| [5:0]op  [5:0]func  [2:0] funcop  [2:0] alucop  pcW  Rtype  ExtOp  ALUop | | I  I  I  I  I  I  I | | 指令高6位  指令低6位  由func控制的信号  由op控制的信号  控制一直取指令  R型指令用于调试和选择ALUop由funcop和 alucop决定  符号扩展信号  ALU单元扩展信号 | |
| RegDst | O | | Rd寄存器控制信号 | |
| ALUsrc | O | | ALU数据来源控制信号 | |
| MemtoReg | O | | 数据内存数据写入寄存器控制信号 | |
| RegWr | O | | 寄存器写控制信号 | |
| MemWr | O | | 数据内存写控制信号 | |
| Branch\_be  Branch\_bn | O  O | | beq指令控制信号  bne指令控制信号 | |
| Jump  Bgez  Bgtz  Blez  Bltz  \_B  Jr  Jal  Jalr | O  O  O  O  O  O  O  O  O | | J指令控制信号  Bgez指令控制信号  Bgtz指令控制信号  Blez指令控制信号  Bltz指令控制信号  \_B(SB LB LBU)指令控制信号  Jr指令控制信号  Jal指令控制信号  Jalr指令控制信号 | |
| （3）功能定义 | 表1.16 contrl和contrl1接口的功能定义 | |  | |
| 序号 | 功能名称 | | 功能描述 | |
| 1 | 输出各种控制信号 | | 根据输入的OP，利用真值表化简，输出各种控制信号。 | |

### 1.1.9 signext模块定义

1. 基本描述

将输入的16位地址按符号位扩展为32位。

1. 模块接口

表1.17 dm\_4k接口的模块定义

|  |  |  |  |
| --- | --- | --- | --- |
| 信号名 |  | 方向 | 描述 |
| [15:0]immediate |  | I | 输入的16位地址 |
| [31:0]imm32 |  | O | 输出的32位地址 |
| ExtOp |  | I | 拓展的信号 |
| （3）功能定义 | 表1.16 | MUX1接口的功能定义 |  |
| 序号 |  | 功能名称 | 功能描述 |
| 1 | 输出扩展的32位地址. | | 根据信号extop将immediate扩展为 32位的imm32. |

## 1.2 各个指令与控制信号之间的关系

### 1.2.1 R-type指令与控制信号

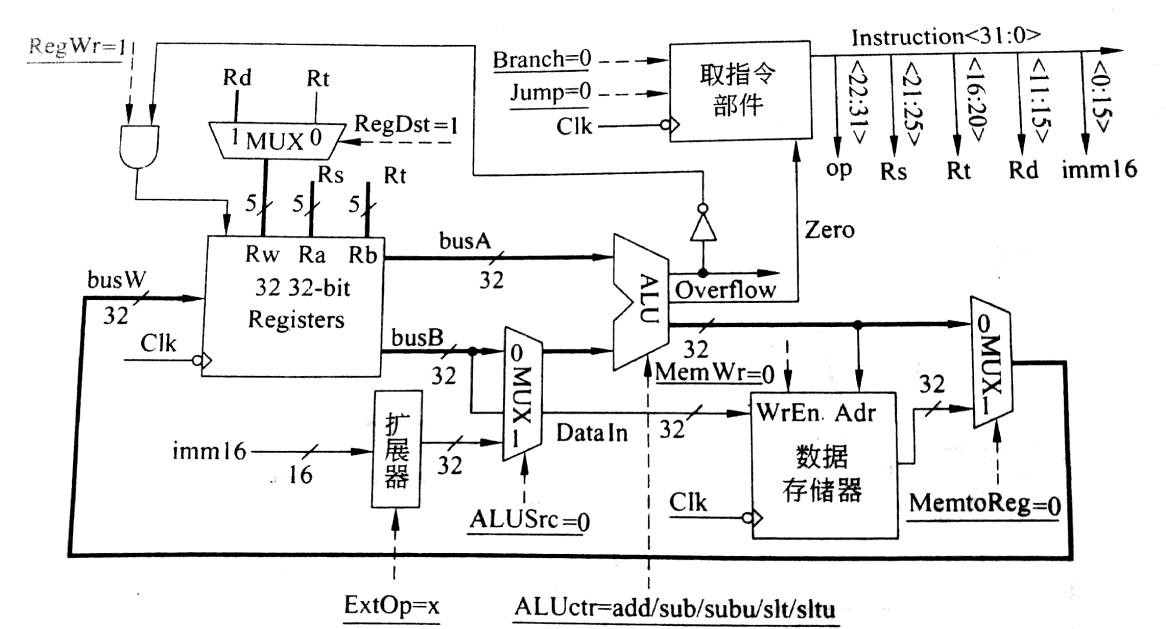


图1.2 R-type指令与控制信号

路径：RegFile(Rs, Rt)->busA, busB->ALU->RegFile(Rd)

### 1.2.2 I-type指令与控制信号

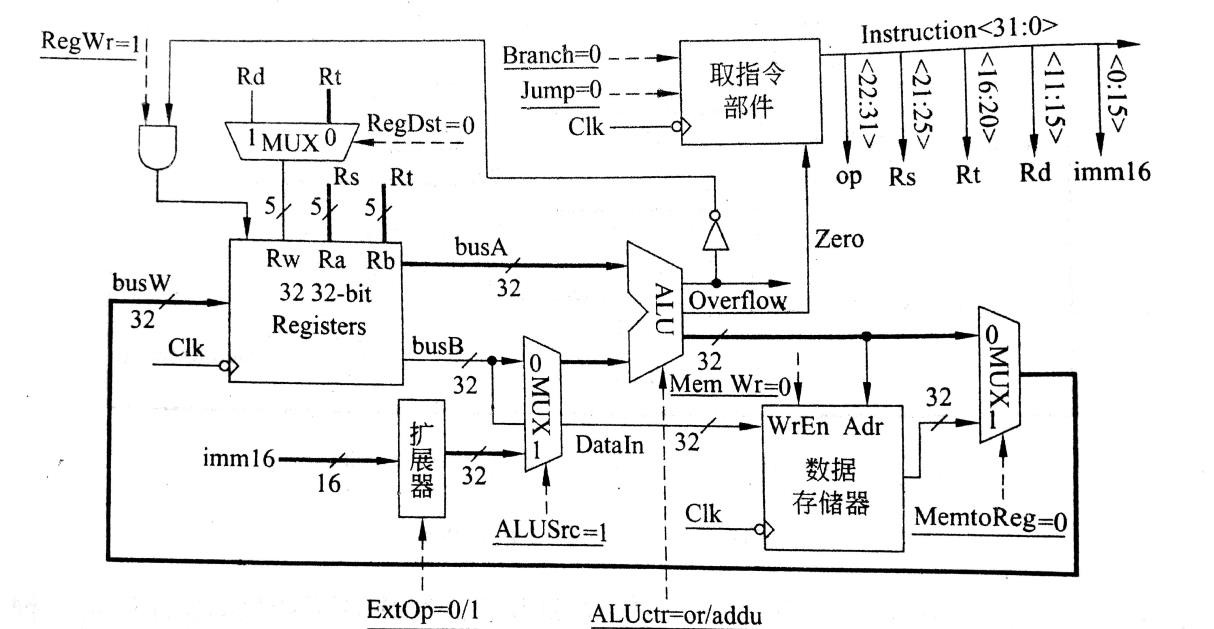


图1.3 I-type指令与控制信号

路径：RegFile(Rs)->busA, 扩展器(imm16)->ALU->RegFile(Rt)

### 1.2.3 Load指令与控制信号

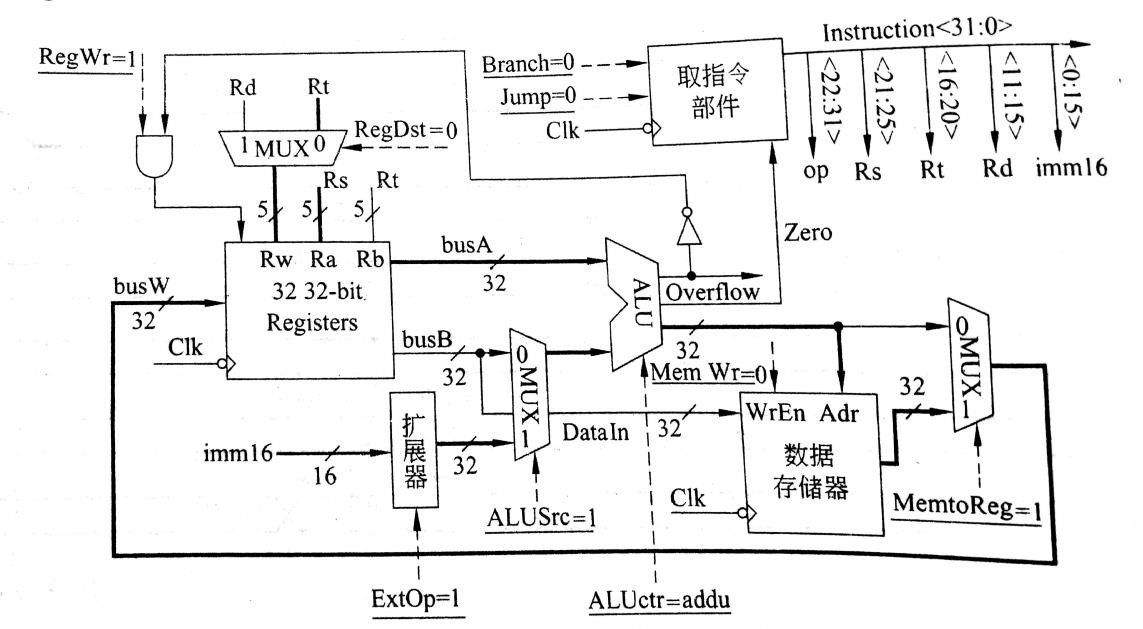


图1.4 Load指令与控制信号

路径：RegFile(Rs)->busA, 扩展器(imm16)->ALU(addu)->数据存储器

->RegFile(Rt)

### 1.2.4 Store指令与控制信号

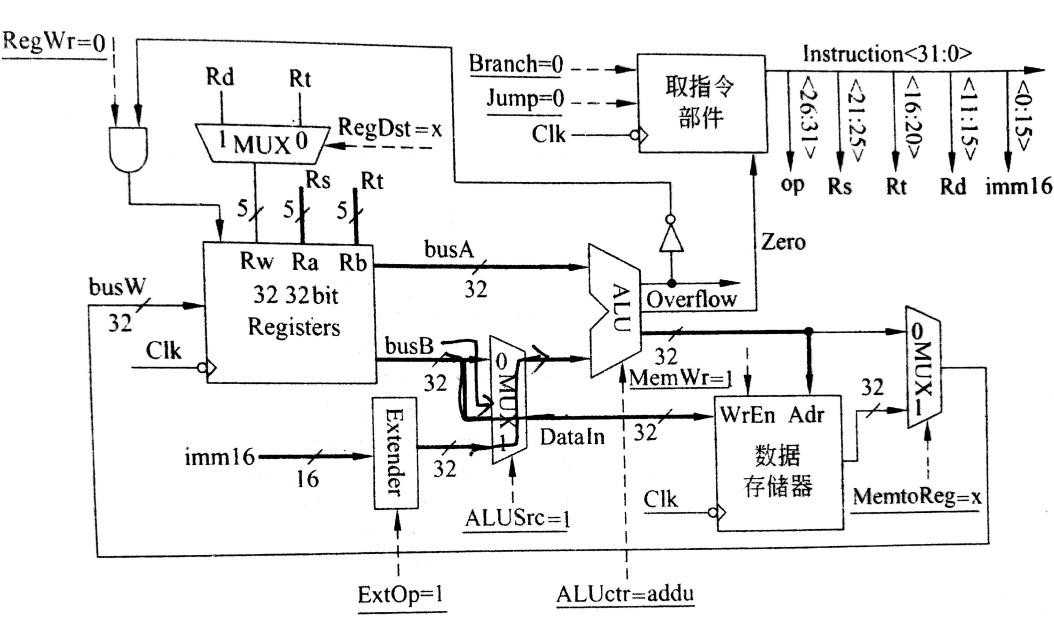


图1.5 Store指令与控制信号

路径：RegFile(Rs, Rt)->busA, 扩展器(imm16), busB->ALU(addu), busB->数

据存储器

### 1.2.5 分支指令与控制信号

路径：RegFile(Rs, Rt)->busA, busB->ALU(subu)->Zero->取指令部件

### 1.2.6 各指令控制信号取值

各个控制信号的逻辑表达式：

Rtype=(op==6'b000000);

pcW=1;

Branch\_bn=(op==6'b000101);//bne

Branch\_be=(op==6'b000100);//beq

Jump=(op==6'b000010);//jump

RegDst=(op==6'b000000);

ExtOp=(op==6'b100011)//lw

||(op==6'b101011)//sw

||(op==6'b001000)//addi

||(op==6'b000001) //bgez & bltz

||(op==6'b000111) //bgtz

||(op==6'b000110) //blez

||(op==6'b100000) //lb

||(op==6'b100100) //lbu

||(op==6'b101000) //sb

||(op==6'b001010) //slti

||(op==6'b001011) //sltiu

||(op==6'b001001);//addiu

ALUsrc=(op==6'b100011)//lw

||(op==6'b101011)//sw

||(op==6'b001000)//addi

||(op==6'b001001)//addiu

||(op==6'b100000) //lb

||(op==6'b100100) //lbu

||(op==6'b101000) //sb

||(op==6'b001010) //slti

||(op==6'b001011) //sltiu

||(op==6'b001101);//ori

MemtoReg=(op==6'b100000) //lb

||(op==6'b100100) //lbu

||(op==6'b100011);//lw

RegWr=(op==6'b000000)//R

||(op==6'b001000)//addi

||(op==6'b100011)//lw

||(op==6'b001001)//addiu

||(op==6'b001100)//andi

||(op==6'b001101)//ori

||(op==6'b001110)//xori

||(op==6'b100000) //lb

||(op==6'b100100) //lbu

||(op==6'b001010) //slti

||(op==6'b001011) //sltiu

||(op==6'b001111);//lui

MemWr=(op==6'b101000)//sb

||(op==6'b101011);//sw

Rtype=op?0:1;

Jr=(func==6'b001000&&op==6'b000000);

Bgez = (op==6'b000001 && rt==5'b00001);

Bgtz = (op==6'b000111 && rt==5'b00000);

Blez = (op==6'b000110 && rt==5'b00000);

Bltz = (op==6'b000001 && rt==5'b00000);

\_B = (op==6'b100000) || (op==6'b100100) || (op==6'b101000); //sb && lb && lbu

Jal = (op==6'b000011);

Jalr=(func==6'b001001&&op==6'b000000);

aluop[3]=(op==6'b001010) //slti

||(op==6'b001111);//lui

aluop[2]=(op==6'b001011) //sltiu

||(op==6'b001010) //slti

||(op==6'b001100 );//andi

aluop[1]=(op==6'b001110)//xori

||(op==6'b001101);//ori

aluop[0]=(op==6'b000100)//beq

||(op==6'b001101)//ori

||(op==6'b000101)//bne

||(op==6'b001010) //slti

||(op==6'b001011) //sltiu

||(op==6'b001111);//lui

// 000110

funcop[3]=(func==6'b000010)//srl

||(func==6'b000011)//sra

||(func==6'b000111)//srav

||(func==6'b000110)//srlv

||(func==6'b000100);//sllv;

funcop[2]=(func==6'b100100)//and

||(func==6'b101010)//slt

||(func==6'b101011)//sltu

||(func==6'b000000)//sll

||(func==6'b000111)//srav

||(func==6'b000110)//srlv

||(func==6'b100111);//nor

funcop[1]=(func==6'b100101)//or

||(func==6'b100110)//xor

||(func==6'b100111)//nor

||(func==6'b000000)//sll

||(func==6'b000011)//sra

||(func==6'b000110)//srlv

||(func==6'b000100);//sllv

funcop[0]=(func==6'b100101)//or

||(func==6'b100010)//sub

||(func==6'b101010)//slt

||(func==6'b100011)//subu

||(func==6'b000000)//sll

||(func==6'b000011)//sra

||(func==6'b000110)//srlv

||(func==6'b101011);//sltu

-

# 第二章 单周期36条指令MIPS CPU的具体设计与调试

## 2.1 具体Verilog代码实现

#### 2.2 ModelSim模拟及测试

##### 2.2.1 综合测试用代码

1. 汇编代码

loop0:addiu $1, $0,1

sll $2, $1,4

addu $3, $2,$1

srl $4, $2,2

slti $25,$4,5

bgez $1,loop1

subu $5,$3,$4

sw $5, 20($0)

nor $6, $5,$2

or $7, $6,$3

xor $8, $7,$6

sw $8, 28($0)

beq $8, $3,loop2

slt $9, $6,$7

loop2: addiu $1, $0,8

lw $10,20($1)

bne $10,$5,loop3

and $11,$2,$1

sw $11,28($1)

sw $4, 16($1)

loop3: jal loop4

loop1: lui $12,12

srav $26,$12,$2

sllv $27,$26,$1

jalr $27

loop4: sb $26,5($3)

sltu $13,$3,$3

bgtz $13,loop4

sllv $14,$6,$4

sra $15,$14,2

srlv $16,$15,$1

blez $16,loop1

srav $16,$15,$1

addiu $11,$0,140

bltz $16, loop6

lw $28,3($10)

bne $28,$29,loop7

loop8:sb $15,8($5)

lb $18,8($5)

lbu $19,8($5)

loop6: sltiu $24,$15,-1

or $29,$12,$5

jr $11

loop7: andi $20,$15,-1

ori $21,$15,-1

xori $22,$15,-1

j loop0

（2）用mars将汇编指令生成机器码如图

24010001

00011100

00411821

00022082

28990005

07210010

00642823

ac050014

00a23027

00c33825

00e64026

ac08001c

11030001

00c7482a

24010008

8c2a0014

15450003

00415824

ac2b001c

ac240010

0c000019

3c0c000c

004cd007

003ad804

0360f809

a07a0005

0063682b

1da0fffd

00867004

000e7883

002f8006

1a00fff5

002f8007

240b008c

06000006

8d5c0003

179d0006

a0af0008

80b20008

90b30008

2df8ffff

0185e825

01600008

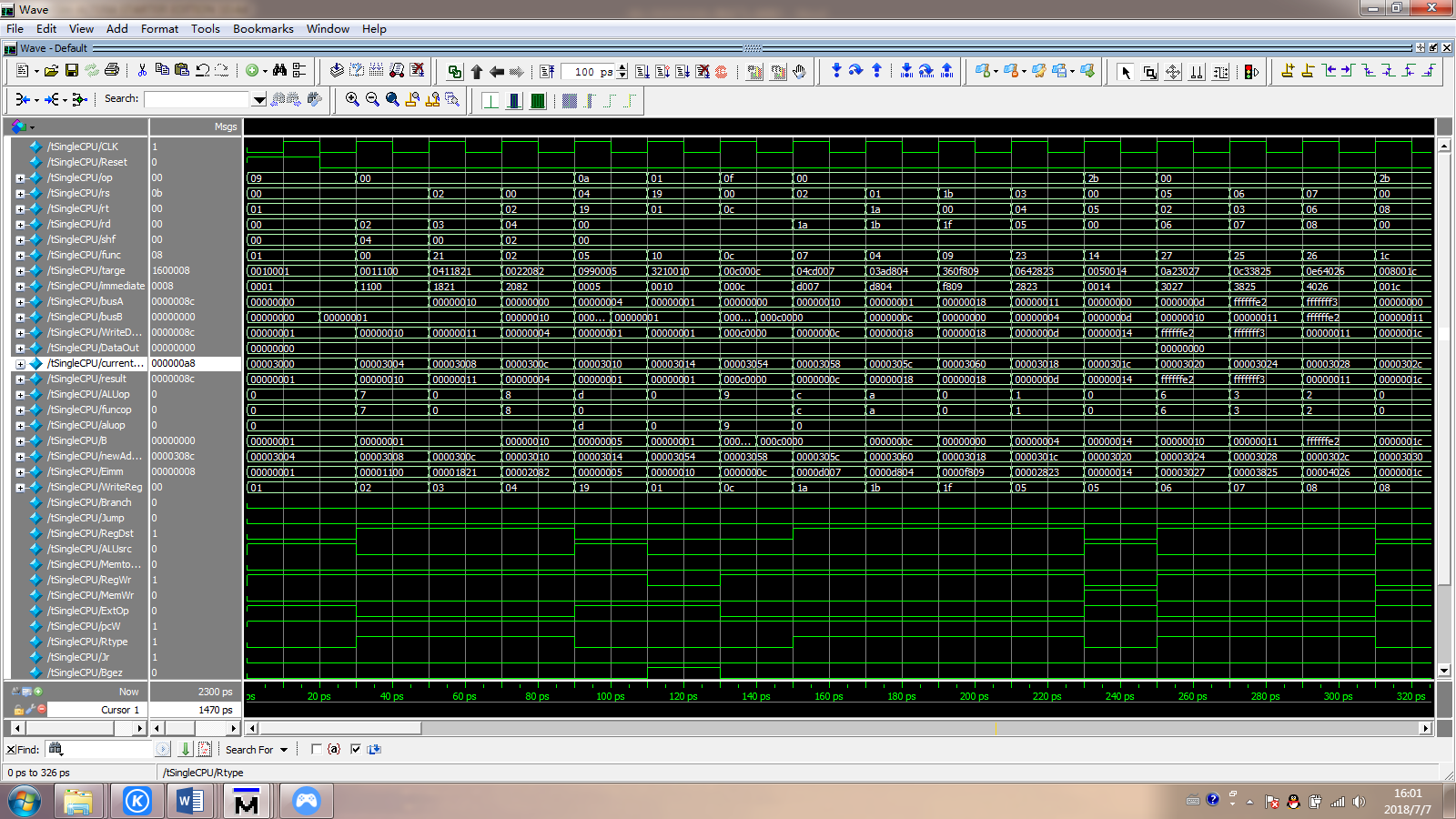
31f4ffff

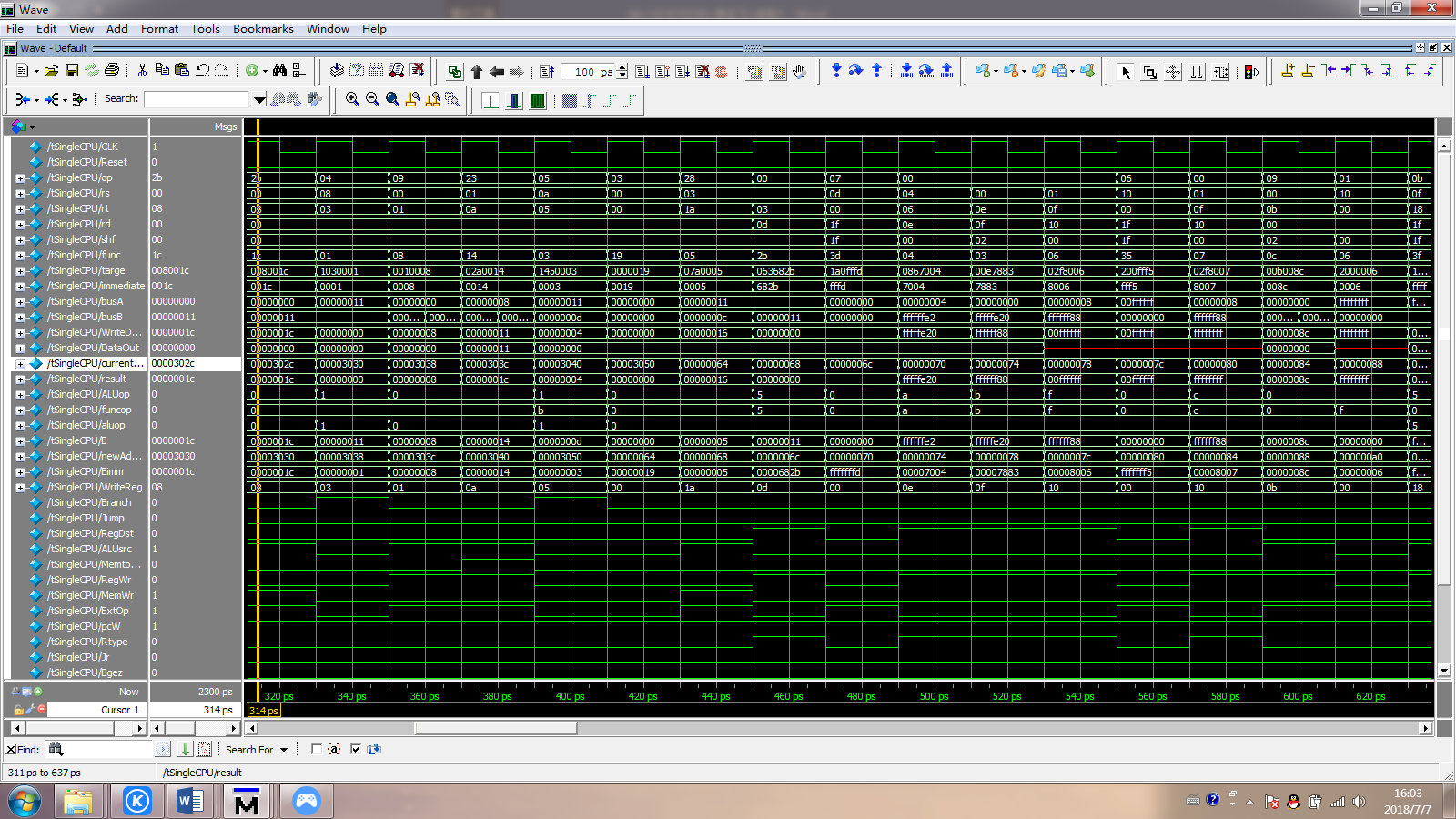
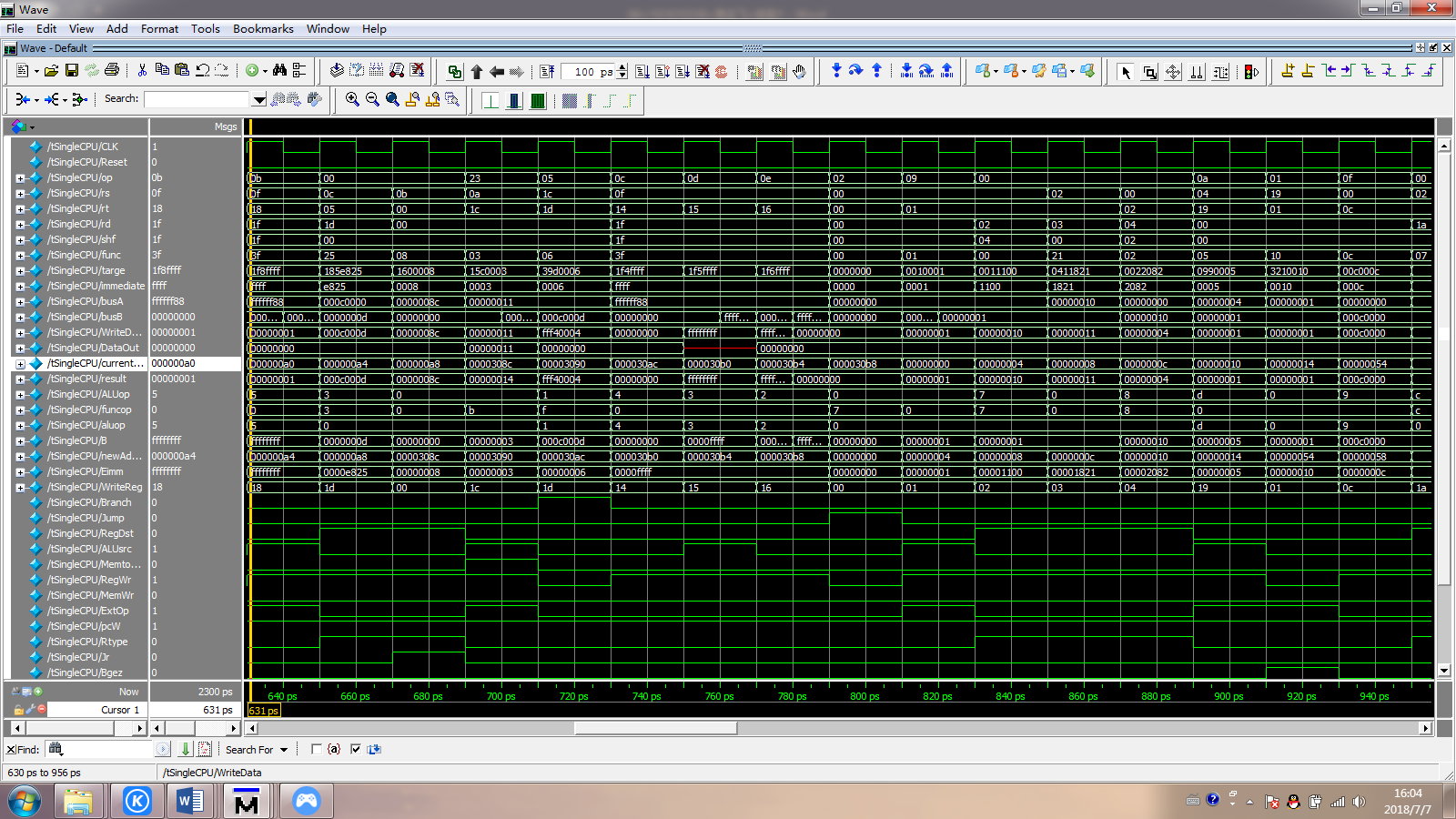
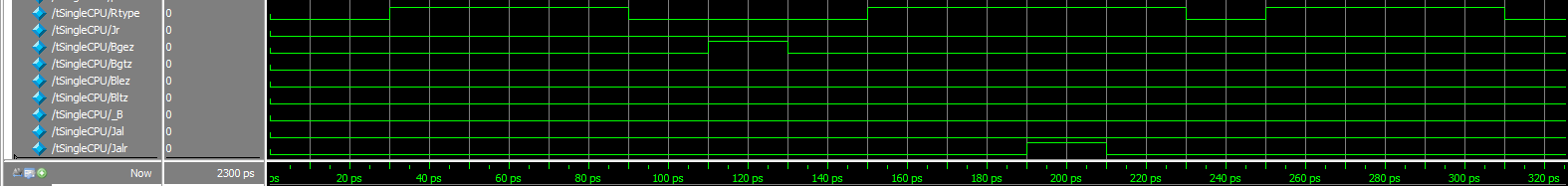
35f5ffff

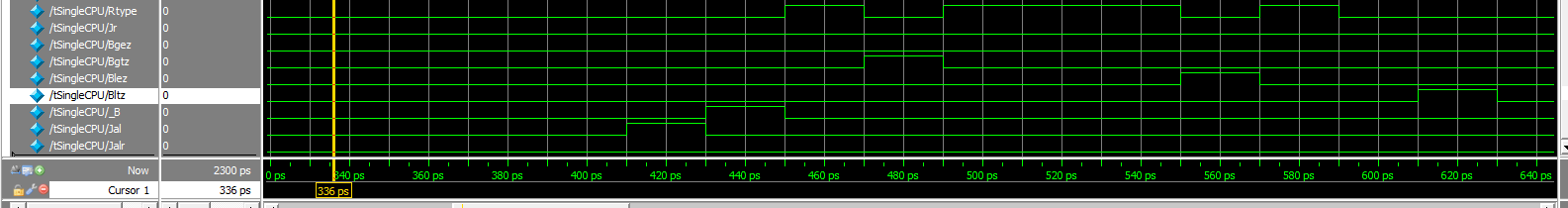
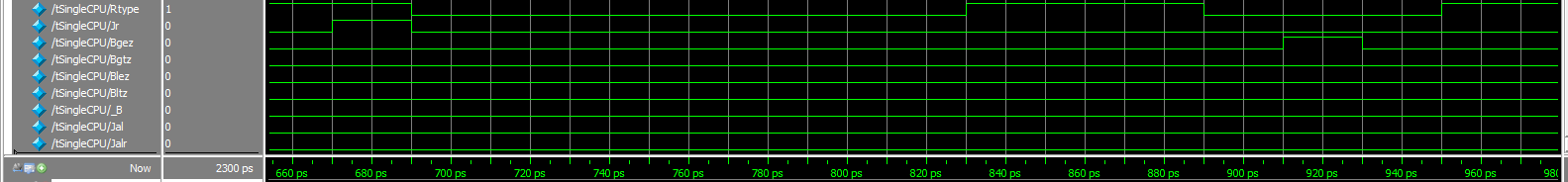
39f6ffff

08000000

运行结果如下







# 第三章 总结和展望

这次的课设对我来说是真的一次挑战。除了自己基础知识不扎实，还有每个阶段的细节不是很熟悉，导致自己的进度很慢。最后因为时间关系，自己放弃了做五级流水线的设计，只成功的做了36条指令的单周期。当然，虽然我放弃，我还是尝试写了几个阶段，也在一定程度上，了解了流水线的原理。

当然，我们的时间也是比较紧张的，每天都是从早到晚。还和我们班的学霸去了另外的实验室，这也和我出去做暑假工一样累。几乎每天早上起来眼睛红肿。不过在做出36条指令的时候还是有些小激动的。虽然还是有些许瑕疵。还有一些是老师规定的，因为没有及时看见消息，导致是自己的思路，以及从网上学来的，就像9条指令的时候，地址的改变。而我的就没有改变，一到验收，学长都是先怀疑是不是我自己写的，然后看到一些规定的，就严厉的批评。就在我验收36条的时候，Jalr指令地址出现错误，哪里如果把rs寄存器的值给pc， 就会导致地址不是从3000开始。我在写的时候，也是赶着时间写，就忘记加3000了。验证的时候，看地址看到后面都是看后面两位。那个时候，眼睛都是花的。我现在也觉得那个确实是个容易犯的错。等我再找学长来验收的时候，那个时候，我们班的同学和我交流了几句，这就很尴尬了，学长又一次怀疑我是抄人家的。还怀疑我不会。这个时候，我能说什么。虽然算我过了，可我心里不好受啊，每次都被怀疑。我承认我是比有些人笨，记性不怎么好，但也是我自己写的，那个错误也是我自己改正过来的。我检查一条指令，我都要打开那个专门有36条指令的看看，他的功能是什么。然后再去看看每个模块对不对。我可以保证都是我写的。

对我来说，只要指令意思理解了，写那些指令的控制信号这些，倒是不难。难就是自己对那些指令不怎么理解。这些也是我的薄弱地方吧，如同地址那些，不知道为什么，自己对其就是学不会或者是理解之后又忘记。导致自己笔试发挥失常。当然相比9条指令，写36条指令，我也学到了很多的地方，如寄存器的理解，如何传值和取值。还有对地址的进一步理解。特别是挑转指令的使用。更多理解了一条指令的流程。

有些遗憾的是，没有完成五级流水线的设计。虽然我也看了学长的课设以及和舍友讨论了一些。我现在觉得还在难的就是各个模块的传值，以及冒险的处理，这个阻塞信号如何传递。并且如何综合几个模块。当然通过看书，自己还是学到了很多，以及对流水线有了更深的理解。最后不用modelsim，也是一种比较好的事，这个软件真的难用，每次都是调试代码的时间最长。

最后，这次的课设也是一次心理挑战，我虽然对成绩没有什么追求，但还是希望公平公正。有的人拿着别人做的，去当课设，还带动所谓的‘节奏’。自己写的，容易错还被怀疑。还有写的慢的，心理也是不好受的。想开点，也是没什么的，每届都有这种情况。如果对成绩看得太重或者成绩对其比较重要，遭到了不平等对待，心理还是难受的。

参 考 文 献

1. MIPS32® Architecture For Programmers Volume II: The MIPS32® Instruction Set[M]. Revision 2.50, Mountain View, CA: MIPS Technologies Inc., July 1, 2005.
2. 袁春风，等. 计算机组成与系统结构[M]. 第 2 版，北京：清华大学出版社，2010.

# 附录1

测试代码解析

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **指令地址** | **汇编指令** | **结果描述** | **机器指令的机器码** | |
| **16进制** | **二进制** |
| 00H | addiu $1, $0,#1 | [$1] = 0000\_0001H | 24010001 | 0010\_0100\_0000\_0001\_0000\_0000\_0000\_0001 |
| 04H | sll $2, $1,#4 | [$2] = 0000\_0010H | 00011100 | 0000\_0000\_0000\_0001\_0001\_0001\_0000\_0000 |
| 08H | addu $3, $2,$1 | [$3] = 0000\_0011H | 00411821 | 0000\_0000\_0100\_0001\_0001\_1000\_0010\_0001 |
| 0CH | srl $4, $2,#2 | [$4] = 0000\_0004H | 00022082 | 0000\_0000\_0000\_0010\_0010\_0000\_1000\_0010 |
| 10H | slti $25,$4,#5 | [$25] = 0000\_0001H | 28990005 | 0010\_1000\_1001\_1001\_0000\_0000\_0000\_0101 |
| 14H | bgez $25,#16 | 跳转到54H | 07210010 | 0000\_0111\_0010\_0001\_0000\_0000\_0001\_0000 |
| 18H | subu $5, $3,$4 | [$5] = 0000\_000DH | 00642823 | 0000\_0000\_0110\_0100\_0010\_1000\_0010\_0011 |
| 1CH | sw $5, #20($0) | Mem[0000\_0014H] = 0000\_000DH | AC050014 | 1010\_1100\_0000\_0101\_0000\_0000\_0001\_0100 |
| 20H | nor $6, $5,$2 | [$6] = FFFF\_FFE2H | 00A23027 | 0000\_0000\_1010\_0010\_0011\_0000\_0010\_0111 |
| 24H | or $7, $6,$3 | [$7] = FFFF\_FFF3H | 00C33825 | 0000\_0000\_1100\_0011\_0011\_1000\_0010\_0101 |
| 28H | xor $8, $7,$6 | [$8] = 0000\_0011H | 00E64026 | 0000\_0000\_1110\_0110\_0100\_0000\_0010\_0110 |
| 2CH | sw $8, #28($0) | Mem[0000\_001CH] = 0000\_0011H | AC08001C | 1010\_1100\_0000\_1000\_0000\_0000\_0001\_1100 |
| 30H | beq $8, $3,#2 | 跳转到38H | 11030002 | 0001\_0001\_0000\_0011\_0000\_0000\_0000\_0010 |
| 34H | slt $9, $6,$7 | 不执行 | 00C7482A | 0000\_0000\_1100\_0111\_0100\_1000\_0010\_1010 |
| 38H | addiu $1, $0,#8 | [$1] = 0000\_0008H | 24010008 | 0010\_0100\_0000\_0001\_0000\_0000\_0000\_1000 |
| 3CH | lw $10,#20($1) | [$10] = 0000\_0011H | 8C2A0014 | 1000\_1100\_0010\_1010\_0000\_0000\_0001\_0100 |
| 40H | bne $10,$5,#4 | 跳转到50H | 15450004 | 0001\_0101\_0100\_0101\_0000\_0000\_0000\_0100 |
| 44H | and $11,$2,$1 | 不执行 | 00415824 | 0000\_0000\_0100\_0001\_0101\_1000\_0010\_0100 |
| 48H | sw $11,#28($1) | 不执行 | AC2B001C | 1010\_1100\_0010\_1011\_0000\_0000\_0001\_1100 |
| 4CH | sw $4, #16($1) | 不执行 | AC240010 | 1010\_1100\_0010\_0100\_0000\_0000\_0001\_0000 |
| 50H | jal #25 | 跳转到64H,  [$31] = 0000\_0054H | 0C000019 | 0000\_1100\_0000\_0000\_0000\_0000\_0001\_1001 |
| 54H | lui $12,#12 | [$12] = 000C\_0000H | 3C0C000C | 0011\_1100\_0000\_1100\_0000\_0000\_0000\_1100 |
| 58H | srav $26,$12,$2 | [$26] = 0000\_000CH | 004CD007 | 0000\_0000\_0100\_1100\_1101\_0000\_0000\_0111 |
| 5CH | sllv $27,$26,$1 | [$27] = 0000\_0018H | 003AD804 | 0000\_0000\_0011\_1010\_1101\_1000\_0000\_0100 |
| 60H | jalr $27 | 跳转到18H ,  [$31] = 0000\_0064H | 0360F809 | 0000\_0011\_0110\_0000\_1111\_1000\_0000\_1001 |
| 64H | sb $26,#5($3) | MEM[0000\_0016H] = 000C\_000DH | A07A0005 | 1010\_0000\_0111\_1010\_0000\_0000\_0000\_0101 |
| 68H | sltu $13,$3,$3 | [$13] = 0000\_0000H | 0063682B | 0000\_0000\_0110\_0011\_0110\_1000\_0010\_1011 |
| 6CH | bgtz $13,#3 | 不跳转 | 1DA00003 | 0001\_1101\_1010\_0000\_0000\_0000\_0000\_0011 |
| 70H | sllv $14,$6,$4 | [$14] =FFFF\_FE20H | 00867004 | 0000\_0000\_1000\_0110\_0111\_0000\_0000\_0100 |
| 74H | sra $15,$14,#2 | [$15] =FFFF\_FF88H | 000E7883 | 0000\_0000\_0000\_1110\_0111\_1000\_1000\_0011 |
| 78H | srlv $16,$15,$1 | [$16] =00FF\_FFFFH | 002F8006 | 0000\_0000\_0010\_1111\_1000\_0000\_0000\_0110 |
| 7CH | blez $16,#8 | 不跳转 | 1A000008 | 0001\_1010\_0000\_0000\_0000\_0000\_0000\_1000 |
| 80H | srav $16,$15,$1 | [$16] =FFFF\_FFFFH | 002F8007 | 0000\_0000\_0010\_1111\_1000\_0000\_0000\_0111 |
|  |  |  |  |  |
| 84H | addiu $11,$0,#140 | [$11] = 0000\_008CH | 240B008C | 0010\_0100\_0000\_1011\_0000\_0000\_1000\_1100 |
| 88H | bltz $16, #6 | 跳转到A0H | 06000006 | 0000\_0110\_0000\_0000\_0000\_0000\_0000\_0110 |
| 8CH | lw $28,#3($10) | [$28] = 000C\_000DH /000C\_880DH | 8D5C0003 | 1000\_1101\_0101\_1100\_0000\_0000\_0000\_0011 |
| 90H | bne $28,$29,#7 | 不跳转/跳转ACH | 179D0007 | 0001\_0111\_1001\_1101\_0000\_0000\_0000\_0111 |
| 94H | sb $15,#8($5) | Mem[0000\_0015H] = 0000\_0088H | A0AF0008 | 1010\_0000\_1010\_1111\_0000\_0000\_0000\_1000 |
| 98H | lb $18,#8($5) | [$18] =FFFF\_FF88H | 80B20008 | 1000\_0000\_1011\_0010\_0000\_0000\_0000\_1000 |
| 9CH | lbu $19,#8($5) | [$19] = 0000\_0088H | 90B30008 | 1001\_0000\_1011\_0011\_0000\_0000\_0000\_1000 |
| A0H | sltiu $24,$15,#0xFFFF | [$24] = 0000\_0001H | 2DF8FFFF | 0010\_1101\_1111\_1000\_1111\_1111\_1111\_1111 |
| A4H | or $29,$12,$5 | [$29] = 000C000DH | 0185E825 | 0000\_0001\_1000\_0101\_1110\_1000\_0010\_0101 |
| A8H | jr $11 | 跳转指令8CH | 01600008 | 0000\_0001\_0110\_0000\_0000\_0000\_0000\_1000 |
| ACH | andi $20,$15,#0xFFFF | [$20] = 0000\_FF88H | 31F4FFFF | 0011\_0001\_1111\_0100\_1111\_1111\_1111\_1111 |
| B0H | ori $21,$15,#0xFFFF | [$21] =FFFF\_FFFFH | 35F5FFFF | 0011\_0101\_1111\_0101\_1111\_1111\_1111\_1111 |
| B4H | xori $22,$15,#0xFFFF | [$22] = FFFF\_0077H | 39F6FFFF | 0011\_1001\_1111\_0110\_1111\_1111\_1111\_1111 |
| B8H | j #00H | 跳转指令00H | 08000000 | 0000\_1000\_0000\_0000\_0000\_0000\_0000\_0000 |

# 附录2

实现指令详解

