组成原理实验课程第_六_次实报告

实验名称	单周期 CPU 实现			班级	张金老师班
学生姓名	蒋薇	学号	2110957	指导老师	董前琨
实验地点	A308		实验时间	2023.6.5	

1、 实验目的

1. 理解 MIPS 指令结构，理解 MIPS 指令集中常用指令的功能和编码，学会对这些
指令进行归纳分类。
2. 了解熟悉 MIPS 体系的处理器结构，如延迟槽，哈佛结构的概念。
3. 熟悉并掌握单周期 CPU 的原理和设计。
4. 进一步加强运用 verilog 语言进行电路设计的能力。

2、 实验内容说明

1. 学习 MIPS 指令集，深入理解常用指令的功能和编码，并进行归纳确定处理器各部件的控制码，比如使用何种 ALU 运算，是否写寄存器堆等。

2. 确定自己本次实验中的准备实现的 MIPS 指令，要求至少实现一条 load 指令、一条 store 指令、10 条基础运算指令、一条跳转指令。其中基础运算指令最好包含多 种类型的操作，必须包含一条加法和一条减法指令。不考虑指令可能产生异常的情 况。单周期 CPU 的实验重点是搭建出一个 CPU 架构，为避免被繁琐的指令所困惑，建议在单周期 CPU 实验中只实现十几条指令。

3. 对准备实现的指令进行分析，完成表的填写

4. 、原始代码实验验证使用实验箱验证，验证时在运行一系列指令之后，实验箱拍照，对比说明各个寄存器中的数据是否是执行正确的结果即可。

5. 改进要求，针对目前 CPU 可运行的 R 型和 I 型 MIPS 指令，各补充一条新的指令，需要修改的 ALU 模块可参照实验四当时的 ALU 改进。改进时注意以下几点：1）MIPS 指令格式要使用规范格式；2）指令执行验证需要修改 inst_rom 中预存储的 16 进制指令数据；3）注意代码中单周期 CPU 模块（single_cycle_cpu)中实现主要功能使用的都是组合逻辑，改进过程中避免使用 alway(clk)这样的时序逻辑。

3、 实验原理图

单周期 CPU 的实现框图如下：

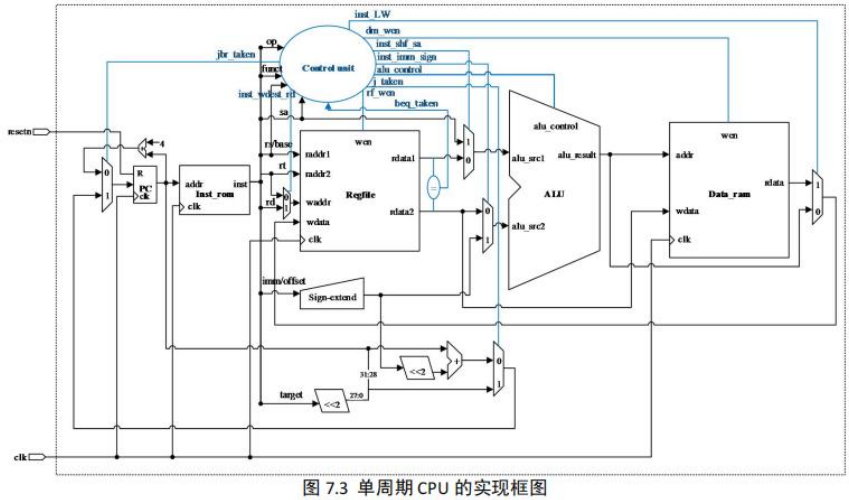


图 7.3 单周期 CPU 的实现框图

单周期 CPU 是指一条指令的所有操作在一个时钟周期内执行完。设计中所有寄存器和存储器都是异步读同步写的，即读出数据不需要时钟控制，但写入数据需时钟控制。

4、 实验步骤

1. 完成下表的填写

表 7.4 单周期 CPU 实现的 mips 指令特性归纳

指令类型	汇编指令	指令码	源操作数 1	源操作数 2	源操作数 3	目的寄存器	功能描述
R 型指令	addu rd,rs,rt	000000|rs|rt|rd|00000|100001	[rs]	[rt]		rd	GPR[rd]=GPR[rs]+GPR[rt]
	subu rd,rs,rt	000000|rs|rt|rd|00000|100011	[rs]	[rt]		rd	GPR[rd]=GPR[rs]-GPR[rt]
	slt rd,rs,rt	000000|rs|rt|rd|00000|101010	[rs]	[rt]		rd	GPR[rd]=(sign(GPR[rs])<sign(GPR[rt]))
	and rd,rs,rt	000000|rs|rt|rd|00000|100100	[rs]	[rt]		rd	GPR[rd]=GPR[rs]&GPR[rt]
	nor rd,rs,rt	000000|rs|rt|rd|00000|100111	[rs]	[rt]		rd	GPR[rd]=~(GPR[rs]|GPR[rt])
	or rd,rs,rt	000000|rs|rt|rd|00000|100101	[rs]	[rt]		rd	GPR[rd]=GPR[rs]|GPR[rt]
	xor rd,rs,rt	000000|rs|rt|rd|00000|100110	[rs]	[rt]		rd	GPR[rd]=GPR[rs]^GPR[rt]
	sll rd,rt,shf	000000|00000|rt|rd|shf|000000		[rt]		rd	GPR[rd]=zero(GPR[rt])<<shf
	srl rd,rt,shf	000000|00000|rt|rd|shf|000010		[rt]		rd	GPR[rd]=zero(GPR[rt])>>shf
I 型指令	addiu rt,rs,imm	001001|rs|rt|imm	[rs]	sign_ext (imm)		rt	GPR[rt]=GPR[rs]+ sign_ext (imm)
	beq rs,rt,offset	000100|rs|rt|offset	[rs]	[rt]			if GPR[rs]=GPR[rt] then PC=PC+ sign_ext (offset)<<2
	bne rs,rt,offset	000101|rs|rt|offset	[rs]	[rt]			if GPR[rs]≠GPR[rt] then PC=PC+ sign_ext (offset)<<2

指令类型	汇编指令	指令码	源操作数 1	源操作数 2	源操作数 3	目的寄存器	功能描述
I 型指令	lw rt,offset(b)	100011|b|rt|offset	[b]	sign_ext (offset)		rt	GPR[rt]=Mem[GPR[b]+sign_ext(offset)]
	sw rt,offset(b)	101011|b|rt|offset	[b]	sign_ext (offset)	[rt]		Mem[GPR[b]+sign_ext(offset)]=GPR[rt]
I 型指令	lui rt,imm	001111|00000|rt|imm		{imm, 16'd0}		rt	GPR[rt]= {imm, 16'd0}
J 型指令	j target	000010|target					PC={PC[31:28],target<<2}

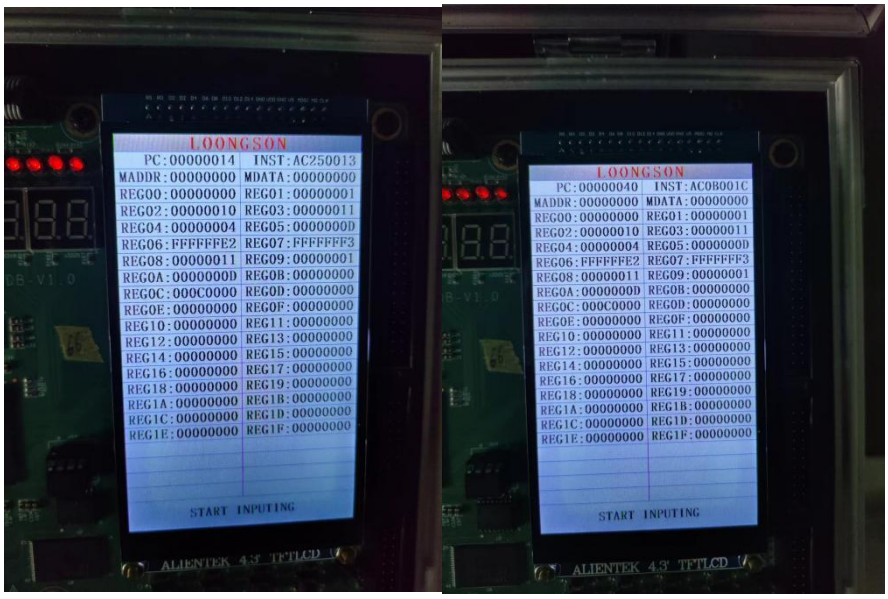
结果描述：

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	subu $5,$3,$4	[$5] = 0000_000DH	00642823	0000_0000_0110_0100_0010_1000_0010_0011
14H	sw $5,#19($1)	Mem[0000_0014H] = 0000_000DH	AC250013	1010_1100_0010_0101_0000_0000_0001_0011
18H	nor $6,$5,$2	[$6] = FFFF_FFE2H	00A23027	0000_0000_1010_0010_0011_0000_0010_0111
1CH	or $7,$6,$3	[$7] = FFFF_FFF3H	00C33825	0000_0000_1100_0011_0011_1000_0010_0101

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
20H	xor $8,$7,$6	[$8] = 0000_0011H	00E64026	0000_0000_1110_0110_0100_0000_0010_0110
24H	sw $8,#28($0)	Mem[0000_001CH] = 0000_0011H	AC08001C	1010_1100_0000_1000_0000_0000_0001_1100
28H	slt $9,$6,$7	[$9] = 0000_0001H	00C7482A	0000_0000_1100_0111_0100_1000_0010_1010
2CH	beq $9,$1,#2	跳转到指令 34H	11210002	0001_0001_0010_0001_0000_0000_0000_0010
30H	addiu $1,$0,#4	不执行	24010004	0010_0100_0000_0001_0000_0000_0000_0100
34H	lw $10,#19($1)	[$10] = 0000_000DH	8C2A0013	1000_1100_0010_1010_0000_0000_0001_0011
38H	bne $10,$5,#3	不跳转	15450003	0001_0101_0100_0101_0000_0000_0000_0011
3CH	and $11,$2,$1	[$11] = 0000_0000H	00415824	0000_0000_0100_0001_0101_1000_0010_0100
40H	sw $11,#28($0)	Men[0000_001CH] = 0000_0000H	AC0B001C	1010_1100_0000_1011_0000_0000_0001_1100
44H	sw $4,#16($0)	Mem[0000_0010H] = 0000_0004H	AC040010	1010_1100_0000_0100_0000_0000_0001_0000
48H	lui $12,#12	[$12] = 000C_0000H	3C0C000C	0011_1100_0000_1100_0000_0000_0000_1100
4CH	j 00H	跳转指令 00H	08000000	0000_1000_0000_0000_0000_0000_0000_0000

5、 实验结果分析

原始代码实验验证使用实验箱验证如图：



截图 1：PC:00000014 时，INST:AC250013,寄存器 0-5 结果都符合运算结果，且实现 Mem[0000_0014H] = 0000_000DH，内存地址 00000014 处存储 0000_000DH;
截图 2：PC:00000040 时，INST:AC0B001C,寄存器 0-11 均符合预期结果。
第三格显示要观察的内存地址，第四格显示该内存地址对应的数据

2.改进要求，针对目前 CPU 可运行的 R 型和 I 型 MIPS 指令，各补充一条新的指令

添加 R 型指令：按位与非，nand rd,rs,rt
添加 I 型指令：立即数无符号减法：subiu, rt,rs,imm

指令名称	汇编指令	功能描述
按位与非	nand rd,rs,rt	GPR[rd] = ~(GPR[rs]&GPR[rt])
立即数无符号减法	subiu rt,rs,imm	GPR[rt] = GPR[rs]-sign_ext(imm)

特性：

指令类型	汇编指令	指令码	源操作数 1	源操作数 2	源操作数 3	目的寄存器	功能描述
R 型指令	nand rd,rs,rt	000000|rs|rt|rd|00000|100111	[rs]	[rt]		rd	GPR[rd] = ~(GPR[rs]&GPR[rt])
I 型指令	subiu rt,rs,imm	001001|rs|rt|imm	[rs]	sign_ext(imm)		rt	GPR[rt]=GPR[rs]-sign_ext (imm)

测试程序详述，在原有代码上，将

按位或非	nor rd,rs,rt	GPR[rd] = ~(GPR[rs] | GPR[rt])
立即数无符号加法	addiu rt,rs,imm	GPR[rt] = GPR[rs] + sign_ext(imm)

改为

按位与非	nand rd,rs,rt	GPR[rd] = ~(GPR[rs]&GPR[rt])
立即数无符号减法	subiu rt,rs,imm	GPR[rt] = GPR[rs]-sign_ext(imm)

```
wire inst_NAND,
wire inst_SUBIU,
```

```
// 实现指令列表
wire inst_ADDU, inst_SUBU , inst_SLT, inst_AND;
wire inst_NOR , inst_OR   , inst_XOR, inst_SLL;
wire inst_SRL , inst_ADDIU, inst_BEQ, inst_BNE;
wire inst_LW  , inst_SW   , inst_LUI, inst_J;
```

```
assign inst_add = inst_ADDU | inst_ADDIU | inst_LW | inst_SW;// 做加法运算
assign inst_sub = inst_SUBU; // 减法
assign inst_slt = inst_SLT;  // 小于置位
assign inst_sltu= 1'b0;      // 暂未实现
assign inst_and = inst_AND;  // 逻辑与
assign inst_nor = inst_NOR;  // 逻辑或非
assign inst_or  = inst_OR;   // 逻辑或
assign inst_xor = inst_XOR;  // 逻辑或非
```

```
assign inst_nand =
 inst_NAND;//逻辑
与非
```

```
inst_slt,
inst_sltu,
inst_and,
inst_nor,
inst_or,
```

inst_or --> inst_nand,

```
est_data <= DM[27];
est_data <= DM[28];
est_data <= DM[29];
est_data <= DM[30];
est_data <= DM[31];
```

alu模块修改为与非操作每一位有

~ (operationl & operation2)

add. v加法修改为减法

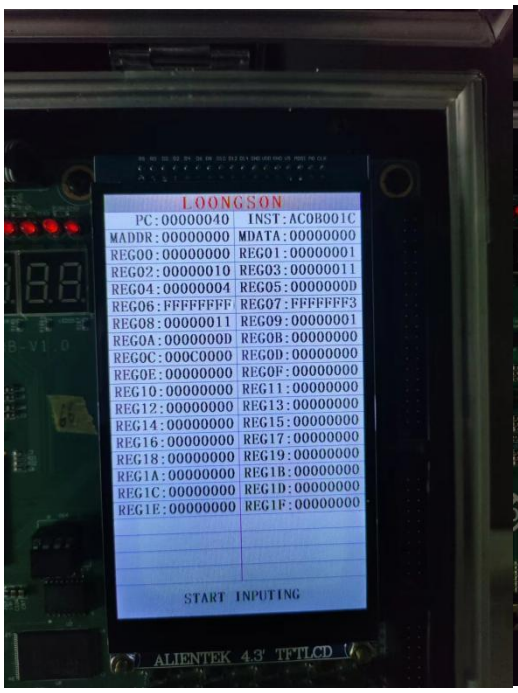
data_ram.v 均为自行搭建的异步存储器，也是在实验
码。单周期 CPU 的源代码还缺少 regfile.v 和 alu.v，
验四中给出，本实验中直接通过"Add Copy of Source"
调用了实验一中的 adder.v。

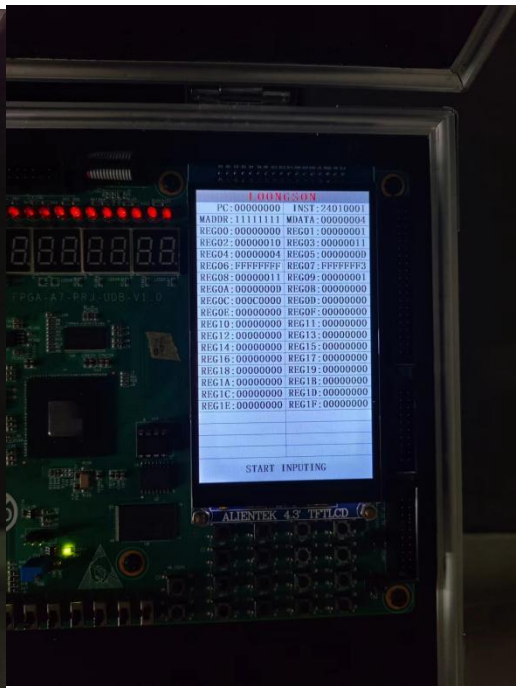
```
assign inst_rom[ 0] = 32'h24010001;//00H: addiu $1 ,$0,#1  | $1 = 0000_0001H
assign inst_rom[ 1] = 32'h00011100;//04H: sll   $2 ,$1,#4  | $2 = 0000_0010H
assign inst_rom[ 2] = 32'h00411821;//08H: addu  $3 ,$2,$1  | $3 = 0000_0011H
assign inst_rom[ 6] = 32'h00A23027;//18H: nor   $6 ,$5,$2  | $6 = FFFF_FFE2H
```

```
assign inst_rom[6] = 32'h00A23027;//18H:nand $6,$5,$2
|$6 = FFFF_FFFFH
```

```
assign inst_rom[ 8] = 32'h00E64026;//20H: xor   $8 ,$7,$6  | $8 = 0000_0011H
assign inst_rom[12] = 32'h24010004;//30H: addiu $1 ,$0,#4  | 不执行
```

```
assign inst_rom[12] = 32'h24010004;//30H,subiu
$1,$0,#4 |不执行
```





调用模块 alu_conctrol()或非操作更改为与非操作，
alu_module()加法更改为减法操作，

理论分析：
与非操作，
寄存器 2 的值为 0000_0010H
寄存器 5 的值为 0000_000DH
与非结果　　(0000_0000_0000_0000_0000_0000_0001_0000)2 与非（0000_0000_0000_0000_0000_0000_0000_1101)2=(1111_1111_11111_1111_1111_1111_11111_1111)2 = FFFF_FFFF
减法，
指令地址，30H，汇编指令， subiu $1,$0,#4 ,, 结果描述，沿用 addiu 加法时的不执行操作，此时机器指令的机器码 16 进制为 24010004，二进制 0010_0100_0000_0001_0000_0000_0000_0010 符合。

6.总结感想

单周期 CPU 是指一条指令的所有操作在一个时钟周期内执行完。设计中所有寄存 器和存储器都是异步读同步写的，即读出数据不需要时钟控制，但写入数据需时钟控制。单周期 CPU 的运作即：在一个时钟周期内，根据 PC 值从指令 ROM 中读出相应的指令，将指令译码后从寄存器堆中读出需要的操作数，送往 ALU 模块，ALU 模块运算得到结果。