计算机组成原理 P3 单周期 CPU 设计实验报告

一、模块规格

1. IFU

（1） 端口说明

IFU 端口定义表

信号名	方向	描述
PCin[31:0]	I	PC 输入信号
clk	I	时钟信号
reset	I	复位信号
PCout[31:0]	O	PC 输出信号
Instr[31:0]	O	32 位指令信号
op[5:0]	O	6 位操作码
Rs[4:0]	O	第一个寄存器源操作数
Rt[4:0]	O	第二个寄存器源操作数
Rd[4:0]	O	寄存器目标操作数
shamt[4:0]	O	5 位位移码
Func[5:0]	O	6 位函数码
imm[15:0]	O	16 位立即数

（2） 功能定义

IFU 功能描述表

序号	功能定义	功能描述
1	取指令	根据 PC 的值取出相应指令
2	拆分指令	将指令拆分出所需的操作数
3	复位	复位信号有效时 PC 清零

2. GRF

（1） 端口说明

GRF 端口定义表

信号名	方向	描述

RegWrite	I	数据写入信号
clk	I	时钟信号
reset	I	复位信号
WD[31:0]	I	目标寄存器写入数据
Rs[4:0]	I	第一个寄存器源操作数
Rt[4:0]	I	第二个寄存器源操作数
Rd[4:0]	I	寄存器目标操作数
Readdata1[31:0]	O	第一个源寄存器读出数据
Readdata2[31:0]	O	第二个源寄存器读出数据

（2） 功能定义

GRF 功能描述表

序号	功能定义	功能描述
1	从寄存器堆读数据	读出 rs、rt 地址对应寄存器中的数据到 Readdata1 和 Readdata2
2	向寄存去堆写数据	RegWrite 信号有效时，将 WD 写入 rd 地址对应的寄存器
3	复位	Reset 信号有效时，所有寄存器数据清零。

3. ALU

（1） 端口说明

ALU 端口定义表

信号名	方向	描述
A[31:0]	I	参与 ALU 计算的第一个值
B[31:0]	I	参与 ALU 计算的第二个值
ALUop[3:0]	I	ALU 功能选择信号 0000：A&B 0001：A|B 0010：A+B 0110：A-B
result[31:0]	O	ALU 计算的结果
Zero	O	判断 A 于 B 是否相等，相等输出 1

（2） 功能定义

ALU 功能描述表

序号	功能定义	功能描述
1	与运算	reselt = A & B
2	或运算	reselt = A | B
3	加运算	reselt = A + B
4	减运算	reselt = A - B
5	比较运算	If（A==B）; Zero = 1;

4. DM

（1） 端口说明

DM 端口定义表

信号名	方向	描述
Address[31:0]	I	32 位内存地址
Writedata[31:0]	I	内存写入数据
clk	I	时钟信号
reset	I	复位信号
memwrite	I	信号有效时，向内存中写入数据
memread	I	信号有效时，从内存中读出数据
MemAddr[4:0]	O	5 位内存地址信号
Readdata[31:0]	O	从内存中读出的数据

（2） 功能定义

DM 功能描述表

序号	功能定义	功能描述
1	读数据	根据地址信号从内存中读取数据
2	写数据	根据地址信号向内存中吸入数据
3	复位	Reset 信号有效时，内存清零

5. EXT

（1） 端口说明

EXT 端口定义表

信号名	方向	描述
imm[15:0]	I	输入 16 位立即数
0-ext[31:0]	O	0 扩展到 32 位
sign-ext[31:0]	O	符号扩展到 32 位
Shift2 [31:0]	O	符号扩展后左移 2 位
Shift16[31:0]	O	左移 16 位

（2） 功能定义

EXT 功能描述表

序号	功能定义	功能描述
1	0 扩展	将 16 位立即数零扩展至 32 位
2	符号扩展	将 16 位立即数符号扩展至 32 位
3	左移 2 位	将符号扩展后的数左移 2 位
4	左移 16 位	将立即数左移 16 位（加载到高位）

6. Controller

（1） 端口说明

Cotroller 端口定义表

信号名	方向	描述
func[5:0]	I	6 位函数码
op[5:0]	I	6 位操作码
RegWrite	O	寄存器写入控制
MemRead	O	内存读取控制
MemWrite	O	内存写入控制
RegDst	O	写寄存器的目标寄存器来自 rt/rd 操作数
Branch	O	Beq 指令信号
MemtoReg	O	写入寄存器选择信号

ALUSrc[1:0]	O	参与 ALU 计算的数据的选择信号
ALUSrc[3:0]	O	ALU 功能选择信号

（2） 功能定义

Cotroller 功能描述表

序号	功能定义	功能描述
1	写寄存器的目标选择	0：写寄存器的目标寄存器号来自 rt 字段 1: 写寄存器的目标寄存器号来自 rd 字段
2	寄存器写入控制	寄存器堆写使能有效
3	内存读取控制	数据存储器读使能有效
4	内存写入控制	数据存储器写使能有效
5	Beq 指令信号	Branch 有效时指令为 beq
6	写入寄存器选择信号	0：向寄存器吸入 ALU 计算结果 1：向寄存器写入从 DM 读取的值
7	参与 ALU 计算的数据的选择信号	00：Readdata2 01：sign-ext 10：0-ext 11：加载立即数到高位
8	ALU 功能选择信号	0000：A&B 0001：A|B 0010：A+B 0110：A-B

7. PCadd

（1） 端口说明

PCadd 端口定义表

信号名	方向	描述
PC[31:0]	I	前 PC 的值
Shift2[31:0]	I	跳转指令需要跳转的位数
PCsrc	I	跳转指令控制信号

PC+ [31:0]	0	下一周期 PC 的值

（2） 功能定义

PCadd 功能描述表

序号	功能定义	功能描述
1	计算 PC	计算下一周期 PC 的值

二、控制器设计

1. 控制信号意义

控制信号	0	1
RegWrite	无	把数据写入寄存器堆中对应寄存器
MemRead	无	数据存储器 DM 读数据（输出）
MemWrite	无	数据存储器 DM 写数据（输入）
RegDst	寄存器堆写入端地址来选择 Rt 字段	寄存器堆写入端地址选择 Rd 字段
PCSrc	PC 输入源选择 PC+4	PC 输入选择 beq 指令的目的地址
MemtoReg	寄存器堆写入端数据来自 ALU 输出	寄存器堆写入端数据来自 DM 输出
控制信号	**功能定义**	**功能描述**
ALUSrc[1:0]	参与 ALU 计算的数据的选择信号	00：Readdata2 01：sign-ext 10：0-ext 11：加载立即数到高位

2. 控制信号真值表

	op	func	RegWrite	MemRead	MemWrite	RegDst	branch	MemtoReg	AlUSrc
addu	000000	100001	1	0	0	1	0	0	00
subu	000000	100011	1	0	0	1	0	0	00
ori	001101		1	0	0	0	0	0	10
lw	100011		1	1	0	0	0	1	01

sw	101011		0	0	1	0	0	0	01
lui	001111		1	0	0	0	0	0	11
beq	000100		0	0	0	0	1	0	00
nop	000000	000000	0	0	0	0	0	0	00

3. ALU 控制信号意义

控制信号	功能定义	功能描述
ALUSrc[3:0]	ALU 功能选择信号	0000：A&B 0001：A|B 0010：A+B 0110：A-B

4. ALU 控制信号真值表

	ALUop
addu	0010
subu	0110
ori	0001
lw	0010
sw	0010
lui	0010
beq	0110
nop	0010

三、测试程序

```
ori $t1,$zero,5
ori $t2,$zero,3
ori $t3,$t1,4

addu $t0,$t1,$t2
subu $t0,$t1,$t2
```

```
    lui $s0,1234
    beq $t1,$t3,label
    lui $s0,1
label:
    lw $t5,0($t0)
    sw $t5,0($t1)
    nop
```

四、思考题

1. 若 PC（程序计数器）位数为 30 位，试分析其与 32 位 PC 的优劣。

   指令按照字边界对齐，即 PC 只用能够标识出不同的指令即可，故 PC 只需要 有 30 位即可。而 32 位 PC 不需改变扩展后指令位数。

2. 现在我们的模块中 IM 使用 ROM， DM 使用 RAM， GRF 使用寄存器，这种做法合理吗？ 请给出分析，若有改进意见也请一并给出。

   合理。

   IM 只需取指令，故使用只读存储器；DM 同时需要存取数据，故使用随机存取存 储器；GRF 存取数据但数量有限，故使用寄存器堆。

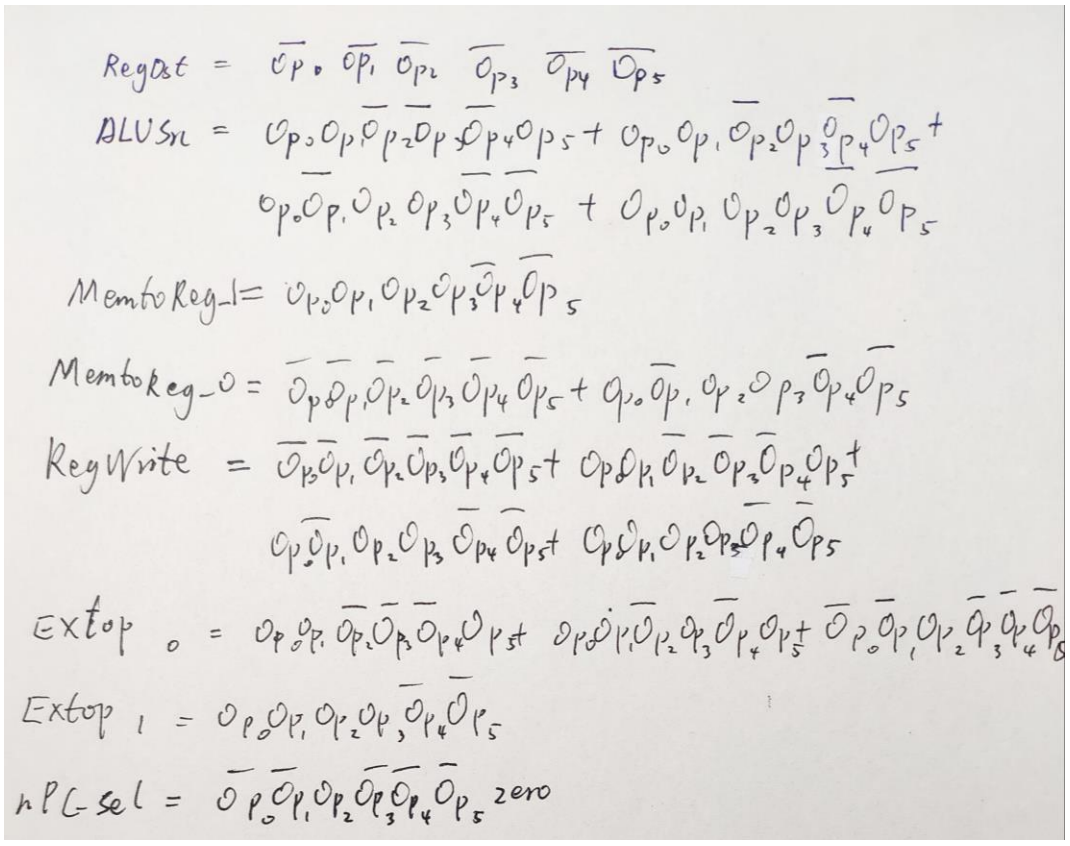
   改进建议：可以在 IM 中使用 RAM，增加指令的改写和增补功能。

3. 结合上文给出的样例真值表，给出 RegDst， ALUSrc， MemtoReg，RegWrite，nPC_Sel，ExtOp 与 op 和 func 有关的布尔表达式(表达式中只能使用“与、或、非”3 种基本逻辑运算。)

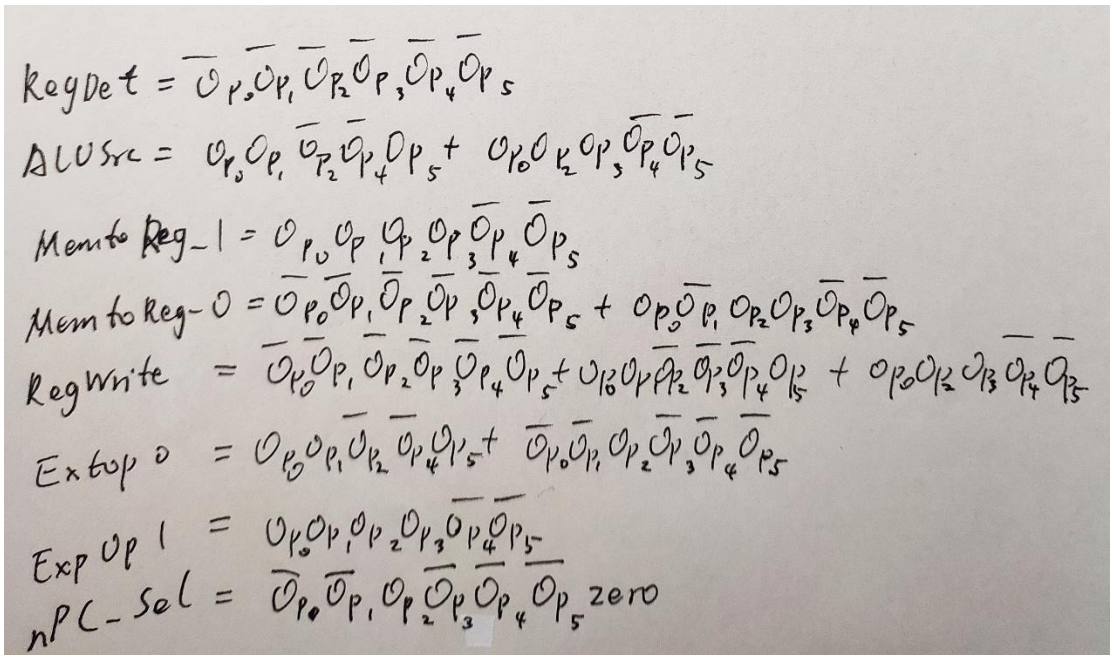
See MIPS Green Sheet

		add	sub	ori	lw	sw	beq
	func	10 0000	10 0010	n/a			
	op	00 0000	00 0000	00 1101	10 0011	10 1011	00 0100
Control Signals	RegDst	1	1	0	0	X	X
	ALUSrc	0	0	1	1	1	0
	MemtoReg	0	0	0	1	X	X
	RegWrite	1	1	1	1	0	0
	MemWrite	0	0	0	0	1	0
	nPC_sel	0	0	0	0	0	1
	ExtOp	X	X	0	1	1	X
	ALUctr<2:0>	Add	Subtract	Or	Add	Add	Subtract

All Supported Instructions



4. 充分利用真值表中的 X 可以将以上控制信号化简为最简单的表达式， 请给出化简后的形式。



5. 事实上，实现 nop 空指令，我们并不需要将它加入控制信号真值表，为什么？请给出你的理由。

   因为 nop 指令为 32 位 0，译码时 op = 000000，func = 000000，识别为 sll 移位指令，将$0 中数据左移零位 存入$0 中，故相当于没有进行操作。

6. 前文提到，“可能需要手工修改指令码中的数据偏移”，但实际上只需再增加一个 DM 片选信号,就可以解决这个问题。请阅读相关资料并设计一个 DM 改造方案使得无需手工修改数据偏移。

   把一个存储器分成多个分区，地址高位选择是哪一个存储器分区，低位选择是该存储器分区内的哪个地址。

   本题中 DM 地址只有 5 位，选取 ALU 结果中 6:2 位作为 DM 地址，无需修改数据偏移。

7. 除了编写程序进行测试外，还有一种验证 CPU 设计正确性的办法——形式验证。**形式验证**的含义是根据某个或某些形式规范或属性，使用数学的方法证明其正确性或非正确性。请搜索“形式验证（Formal Verification)”了解相关内容后，简要阐述相比与测试，形式验证的优劣。

优点：

(1)形式验证是对指定描述的所有可能的情况进行验证，覆盖率达到了 100%。

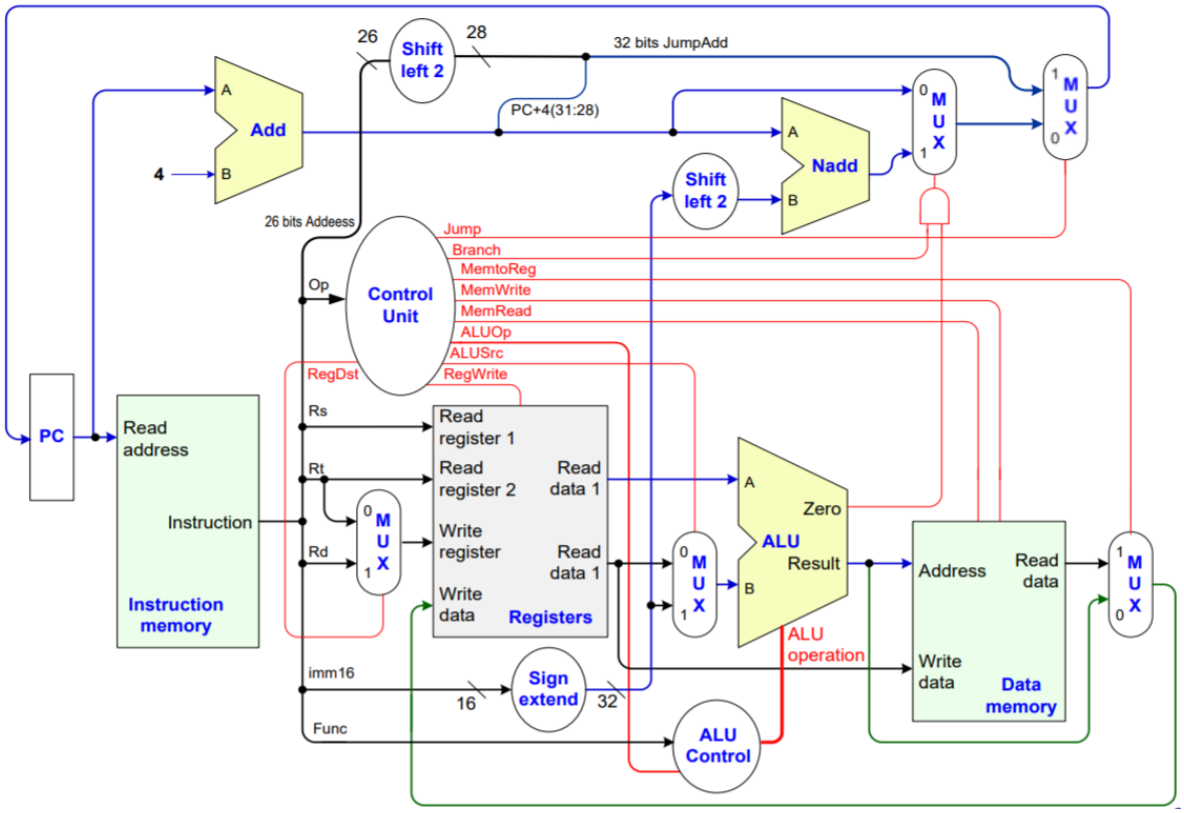
(2)形式验证技术是借用数学上的方法将待验证电路和功能描述或参考设计直接进行比较，不需要开发测试激励。

(3)形式验证的验证时间短，可以很快发现和改正电路设计中的错误，可以缩短设计周期。

缺点：

到目前为止不能有效地验证电路的性能，如时延和功耗等问题。

五、设计参考



指令	adder		PC	IM address	register				ALU		DM		0-ext	signext	shift16	shift2	Nadd	
	A	B			Reg1	Reg2	Wreg	Wdata	A	B	add.	Wdata						
addu	PC	4	adder	PC	Rs	Rt	Rd	ALU	Rdata1	Rdata2								
subu	PC	4	adder	PC	Rs	Rt	Rd	ALU	Rdata1	Rdata2								
ori	PC	4	adder	PC	Rs		Rt	ALU	Rdata1	0-ext			imm16					
lw	PC	4	adder	PC	Rs		Rt	DM	Rdata1	signext	ALU			imm16				
sw	PC	4	adder	PC	Rs	Rt			Rdata1	signext	ALU	Rdata2		imm16				
lui	PC	4	adder	PC	Rs		Rt	ALU	Rdata1	shift16					imm16			
beq	PC	4	adder|Nadd	PC	Rs	Rt			Rdata1	Rdata2				imm16		signext	Adder	shift2
nop	PC	4	adder	PC														
合并	PC	4	adder|Nadd	PC	Rs	Rt	Rt|Rd	ALU|DM	Rdata1	Rdata2|signext|0-ext|shift16	ALU	Rdata2	imm16	imm16	imm16	signext	Adder	shift2
			PCScr				RegDst	MemtoReg		ALUSrc								