

华中科技大学

2018

计算机组成原理

·实验报 告·

专 业： 计算机科学与技术

班 级： CS1603

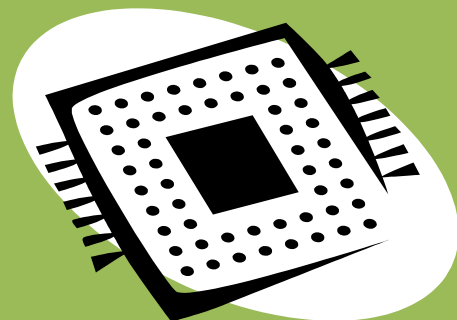
学 号： U201614577

姓 名： 龙际全

电 话： 15007172871

邮 件： boyjqlong@foxmail.com

完成日期： 2018/12/9



计算机科学与技术学院

华中科技大学课程实验报告

目 录

1	CPU 设计实验	2
1.1	设计要求	2
1.2	方案设计	2
1.3	实验步骤	4
1.4	故障与调试	15
1.5	测试与分析	17
2	总结与心得.....	21
2.1	实验总结	21
2.2	实验心得	21
	参考文献.....	23

1 CPU 设计实验

1.1 设计要求

- (1) 构建一个 32 位 MIPS CPU 处理器，包括单周期硬布线 CPU、多周期微程序 CPU 以及多周期硬布线 CPU，该处理器应支持核心指令集中列出的所有指令，见表 1.1。具体指令功能参见附件中的 MIPS 标准文档。最终设计完成的 CPU 应能运行标准测试程序。

表 1.1 核心指令集

#	指令	格式	备注
1	Add	add \$rd, \$rs, \$rt	指令功能及指令格式 参考 MIPS32 指令集
2	Add Immediate	addi \$rt, \$rs, immediate	
3	Load Word	lw \$rt, offset(\$rs)	
4	Store Word	sw \$rt, offset(\$rs)	
5	Branch on Equal	beq \$rs, \$rt, label	
6	Branch on Not Equal	bne \$rs, \$rt, label	
7	Set Less Than	slt \$rd, \$rs, \$rt	
8	syscall (display or exit)	syscall	系统调用，用于停机

1.2 方案设计

1.2.1 单周期 CPU 功能部件

- (1) 指令计数器 PC，用于存放当前指令执行位置的地址；
- (2) 指令存储器 IM，储存 CPU 要执行的指令；
- (3) 数据存储器 DM，根据设计需求，所有指令都必须在一个时钟周期之内完成，只采用一个存储器不可能在一个时钟周期内同时完成对指令和数据的操作；
- (4) 立即数扩展器 S-EXT，用于将 I 型指令中的 16 位立即数扩展为 32 位；

- (5) 控制器，产生控制信号，控制指令执行的数据通路；
- (6) 寄存器堆 RegiFile，提供 32 个 MIPS 通用寄存器；
- (7) 算术逻辑单元 ALU，产生运算结果。

1.2.2 多周期 CPU 功能部件

- (1) 指令计数器 PC，用于存放当前指令执行位置的地址；
- (2) 存储器 Mem，包括数据存储器以及指令存储器；
- (3) 指令寄存器 IR，存放当前指令；
- (4) 数据存储器 DR，存放要操作的数据；
- (5) 寄存器堆 RegiFile，提供 32 个 MIPS 通用寄存器；
- (6) 立即数扩展器 S-EXT，用于将 I 型指令中的 16 位立即数扩展为 32 位；
- (7) 控制器，产生控制信号，控制指令执行的数据通路；
- (8) 算术逻辑单元 ALU，产生运算结果；
- (9) 另外，增加三个寄存器 A、B、C 暂存 RegiFile 和 ALU 的数据输出。

1.2.3 地址转移逻辑 NPC

执行完一条指令后，下一步该执行哪条指令，根据实验要求的八条核心指令分析，有以下几种情况：

- (1) 顺序执行，继续执行相邻的下一条语句；
- (2) 遇到有条件分支，跳转到当前地址加上偏移地址。

两种可能的跳转，一次只能选择一个值，需要 1 个控制信号，由操作控制器生成。

1.2.4 立即数扩展器

立即数扩展有两种，都是扩展成 32 位。

- (1) 条件跳转指令中 16 位偏移地址的扩展，偏移量可正可负，需要用符号扩展；
- (2) 在 I 型指令中，操作数为 16 位的立即数，操作数可正可负，需要用符号

扩展。

1.2.5 操作控制器

操作控制器根据指令的操作码和 func 产生控制信号。使用工程化方法生成。

- (1) 列出所有指令下，各个数据通路的值；
- (2) 将每一项数据都进行合并，当一项数据有多种情况，表明该数据项需要用控制信号进行选择，否则，不需要选择，可以直连；
- (3) 控制信号综合，列出各指令下，每个控制信号的值为 1 或 0，从而可以得出控制信号的逻辑表达式，就可以生成电路了；
- (4) 多周期控制器可分为微程序控制器和硬布线控制器，由于篇幅较长，将在实验步骤中给出。

1.3 实验步骤

1.3.1 指令解析

将 32 位输入操作码用分线器接出，最高 6 位为操作码 op，21-25 位为 rs 寄存器编号，16-20 位为 rt 寄存器编号、11-15 位为 rd 寄存器编号，0-5 位为功能码 func；取低 16 位，作为 I 型指令中的立即数操作数。单周期和多周期指令译码方式相同，如下图所示：

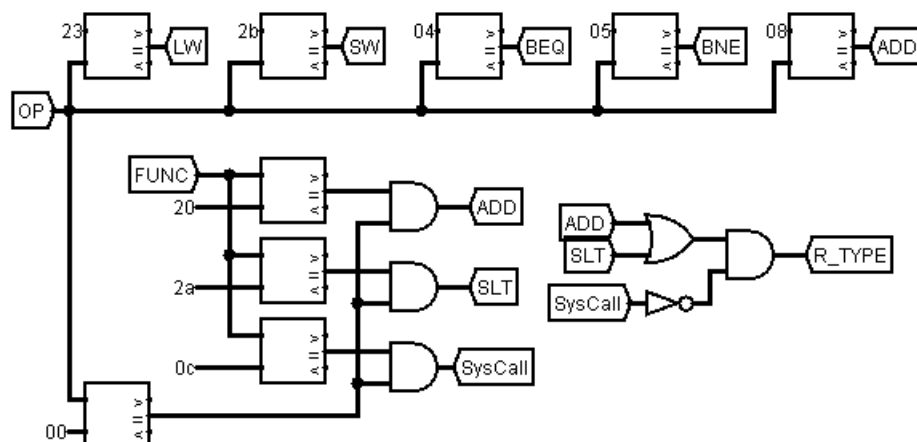


图 1.1 指令译码

华中科技大学课程实验报告

1.3.2 实现地址转移逻辑

PC+4 接到 MUX 的 0 输入端, 16 位立即数扩展结果左移 2 位, 加上 PC+4, 接到 MUX 的 1 输入端, 选择信号为 branch。实验要求的分支转移指令为 beq 和 bne, 两条指令均为 I 型指令, 由于两条指令的特殊性, 可利用 ALU 的 equal 信号, 最终 $branch = beq * equal + bne * /equal$ 形成分支信号。

1.3.3 单周期硬布线控制器

(1) 数据通路综合

- 构建数据通路。绘制主要功能部件输入来源表, 该表主要用于描述控制类信号, 仅保留数据类信号, 具体如表 1.2, 最左侧为指令助记符, 第一行为数据项。
- 输入源合并。计算表 1.2 中每一个数据项的输入种类, 若有多个输入来源, 则需要控制信号配合多路选择器进行选择。

表 1.2 单周期功能部件输入来源表

指令	PC	IM	RegiFile (RF)				S-EXT	ALU		DM	
			R1#	R2#	W#	Din		A	B	Addr	Din
add	PC+4	PC	rs	rt	rd	ALU		RF.D1	RF.D2		
slt	PC+4	PC	rs	rt	rd	ALU		RF.D1	RF.D2		
addi	PC+4	PC	rs		rt	ALU	IMM[15:0]	RF.D1	S-EXT		
lw	PC+4	PC	rs		rt	DM.Dout	IMM[15:0]	RF.D1	S-EXT	ALU	
sw	PC+4	PC	rs	rt			IMM[15:0]	RF.D1	S-EXT	RF.D2	ALU
beq	PC+4+offset/PC	PC	rs	rt			IMM[15:0]	RF.D1	RF.D2		
bne	PC+4+offset/PC	PC	rs	rt			IMM[15:0]	RF.D1	RF.D2		
syscall	PC	PC									
合并	2输入	1输入	1输入	1输入	2输入	2输入	1输入	1输入	2输入	2输入	1输入

- 列出所有功能部件、多路选择器控制信号、运算操作选择的产生条件, 如表 1.3 所示, 横坐标给出的是不同指令的译码信号, 表中有 1 的位置表示当前指令会产生对应的信号, 利用译码电路生成各指令译码信号, 然后以行为单位将各个产生信号的条件相加 (逻辑或) 即可得到控制信号的逻辑表达式。

- RegDst 为 1 表示 RegiFile 写回地址由 R 型指令 rd 字段给出, 否则由 I 型指令 rt 字段给出;
- RegWrite 为 1 打开 RegiFile 写使能, 表示数据写回 RegiFile;

华中科技大学课程实验报告

- c) AluSrcB 为 1 表示 ALU 的第二个操作数将由立即数扩展器 S-EXT 给出，否则由 RegiFile 的第二个输出给出；
- d) bne 为 1 表示指令为 bne；
- e) beq 为 1 表示指令为 beq, beq 与 bne 信号与 ALU equal 信号结合可控制 PC 数据来源；
- f) MemWrite 为 1 打开数据存储器 DM 写使能，将 ALU 运算结果写入数据存储器 DM；
- g) MemToReg 为 1 表示从数据存储器 DM 中选数据送入 RegiFile；
- h) Halt 为停机信号，为 1 时系统停机；
- i) 加法信号，表示 ALU 应对两个操作数执行加法操作；
- j) 比较信号，表示 ALU 应对两个操作数执行比较操作。

表 1.3 单周期硬布线控制器信号表

控制信号	1	2	3	4	5	6	7	8
	add	slt	addi	lw	sw	beq	bne	syscall
RegDst	1	1						
RegWrite	1	1	1	1				
AluSrcB			1	1	1			
bne							1	
beq						1		
MemWrite					1			
MemToReg				1				
Halt				1				1
加法	1		1	1	1			
比较		1						

(3) 根据以上分析，最终形成的单周期硬布线控制器电路如下：

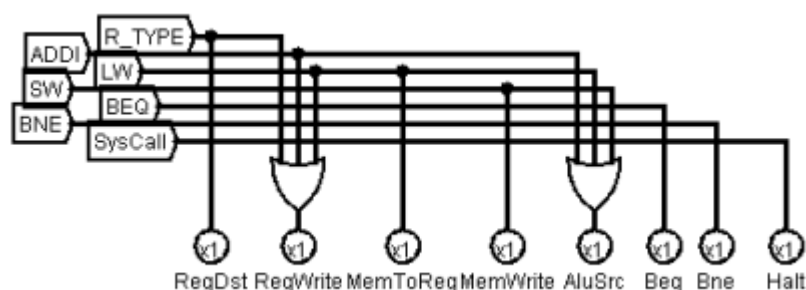


图 1.2 MIPS 单周期 CPU 控制器电路

1.3.4 单周期 CPU 总体结构图

通过上面的分析，我们已经得到单周期 CPU 的所有功能部件，我们也知道了所有指令的数据通路，以及如何产生控制信号控制指令数据的传递，因此把这些部件和控制信号连接起来就得到了单周期 CPU 总体结构图。

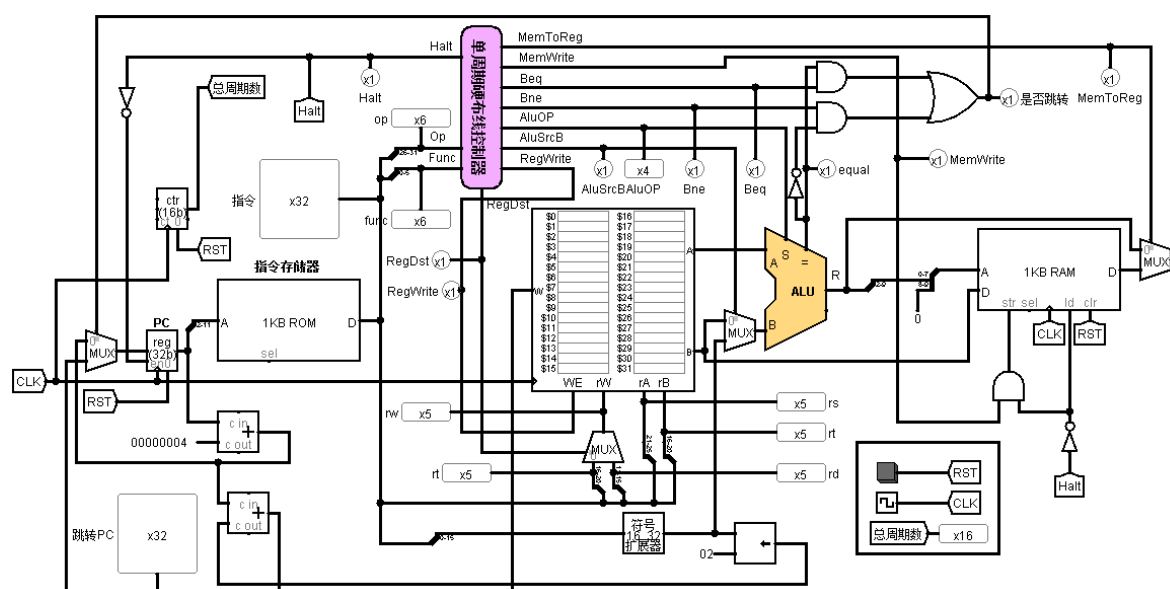


图 1.3 MIPS 单周期 CPU 总体结构图

1.3.5 从单周期到多周期

多周期处理器中，指令的执行需要占用多个时钟周期，不同的指令所需的时钟周期数不一定相同，相同指令在不同时钟节拍下产生的控制信号也不同。正如 1.2.2 所述，多周期 CPU 的整体架构也与单周期 CPU 有些许不同，如下：

- (1) 不再区分指令存储器与数据存储器，指令和数据保存在同一个存储器中；
- (2) 部分功能单元，如 ALU、RegiFile 可以在一条指令执行过程的不同周期中多次使用；
- (3) 主要功能单元输出端都增加了寄存器，在后续时钟周期中要用到的所有数据必须存储在相应的寄存器中，增加数据寄存器 DR，用于存放从存储器读取的数据，增加指令寄存器 IR，用于存放从存储器读出的指令，增加三个寄存器 A、B、C 保存 RegiFile 和 ALU 的输出；
- (4) 更改后 ALU 输出结果有三种情况：
 - a) 分支目标地址，由 beq、bne 指令给出，该地址将被写到 PC 中；

华中科技大学课程实验报告

- b) 指令的运算结果, 将被写入到寄存器文件中;
- c) 为存储器访问指令 lw 和 sw 提供存储器地址;
- (5) PC 作为指令计数器, 由于不同指令时钟周期数不同, 因此 PC 不再仅由时钟周期控制, 而是增加了专门的写操作控制信号;

1.3.6 多周期控制信号

多周期 CPU 控制下, 不同指令对应不同的时钟周期数, 因此不能像单周期控制器一般一次性给出一条指令对应的所有控制信号, 而每一条指令的执行又可以拆解为三部分“取指->译码->执行”三个阶段, 其中取指和译码两个阶段所有指令对应的数据通路相同, 因此现在的任务退化为分析每条指令执行阶段需要的时钟周期并给出每个时钟周期下所有的控制信号, 指令的执行控制应遵循以下几个原则:

- (1) 有序进行;
- (2) 不会破坏系统中保存的结果, 仅仅对指令本身功能要求对系统状态做出改变;
- (3) 保证不同周期下指令的每一步执行所用到的数据都来自正确来源, 保证每一步的运算结果保存到正确输出。

基于此原则, 给出取指、译码阶段以及每条指令的执行流程表, 如下各表所示:

表 1.4 取指阶段操作流程

指令阶段	操作流程
取指令	$IR \leftarrow (MEM[PC])$ $PC \leftarrow (PC) + 4$
译码及取操作数	$A \leftarrow (R[IR[25:21]])$ $B \leftarrow (R[IR[20:16]])$ $C \leftarrow (PC) + (S-EXT(IR[15:0]) \ll 2)$

表 1.5 add 指令执行操作流程

指令阶段	操作流程
加运算	$C \leftarrow (A) + (B)$
写回	$R[IR[15:11]] \leftarrow (C)$

华中科技大学课程实验报告

表 1.6 slt 指令执行操作流程

指令阶段	操作流程
比较运算	$C \leftarrow ((A) < (B))$
写回	$R[IR[15:11]] \leftarrow (C)$

表 1.7 lw 指令执行操作流程

指令阶段	操作流程
计算地址	$C \leftarrow (A) + S-EXT(IR[15:0])$
访存	$DR \leftarrow (MEM[PC])$
写回	$R[IR[20:16]] \leftarrow (DR)$

表 1.8 sw 指令执行操作流程

指令阶段	操作流程
计算地址	$C \leftarrow (A) + S-EXT(IR[15:0])$
访存	$DR \leftarrow (MEM[PC])$

表 1.9 beq 指令执行操作流程

指令阶段	操作流程
送目标地址	$If(A == B) PC \leftarrow (C)$

表 1.10 bne 指令执行操作流程

指令阶段	操作流程
送目标地址	$If(A \neq B) PC \leftarrow (C)$

表 1.11 addi 指令执行操作流程

指令阶段	操作流程
加运算	$C \leftarrow (A) + S-EXT(IR[15:0])$
写回	$R[IR[20:16]] \leftarrow (C)$

表 1.12 syscall 指令执行操作流程

指令阶段	操作流程
空操作，停机	锁住 PC

综合以上八条操作流程，可以给出所有指令不同阶段的控制信号表，如表 1.13 所示，控制信号说明如下：

- (1) PCWrite: PC 写使能控制，取指令周期，分支指令执行；

华中科技大学课程实验报告

- (2) IorD: 指令还是数据, 0 表示指令, 1 表示数据;
- (3) Irwrite: 指令寄存器写使能;
- (4) MemWrite: 写内存控制信号;
- (5) MemRead: 读内存控制信号;
- (6) Beq: beq 指令译码信号;
- (7) Bne: bne 指令译码信号;
- (8) PcSrc: PC 输入源, 0 表示顺序寻址, 1 表示跳跃寻址;
- (9) AluControl: ALU 控制信号, 即加法或比较运算;
- (10) AluSrcA: ALU 第一输入选择, PC 还是寄存器输出;
- (11) AluSrcB: ALU 第二输入选择, R 型指令输入为寄存器输出, 取指阶段为 4, sw、lw、addi 指令为立即数, 跳转指令 bne、beq 相应的偏移地址;
- (12) RegWrite: 寄存器 RegiFile 写使能;
- (13) RegDst: RegiFile 的第二个寄存器编号由 R 型指令 rd 给出;
- (14) MemToReg: lw 指令, 写入寄存器的数据来自存储器。

表 1.13 指令控制信号表

功能	IorD	PcSrc	AluSrcA	AluSrcB	MemToReg	RegDst	IrWrite	PcWrite	RegWrite	MemWrite	MemRead	BEQ	BNE	AluControl
取指令	0	0	0	01	0	0	1	1	0	0	1	0	0	00
译码	0	0	0	11	0	0	0	0	0	0	0	0	0	00
LW1	0	0	1	10	0	0	0	0	0	0	0	0	0	00
LW2	1	0	0	00	0	0	0	0	0	0	1	0	0	00
LW3	0	0	0	00	1	0	0	0	1	0	0	0	0	00
SW1	0	0	1	10	0	0	0	0	0	0	0	0	0	00
SW2	1	0	0	00	0	0	0	0	0	1	0	0	0	00
R型运算1	0	0	1	00	0	0	0	0	0	0	0	0	0	10
R型运算2	0	0	0	00	0	1	0	0	1	0	0	0	0	00
Beq	0	1	1	00	0	0	0	0	0	0	0	1	0	01
Bne	0	1	1	00	0	0	0	0	0	0	0	0	1	01
ADDI1	0	0	1	10	0	0	0	0	0	0	0	0	0	00
ADDI2	0	0	0	00	0	0	0	0	1	0	0	0	0	00
SYSCALL	0	0	0	00	0	0	0	0	0	0	0	0	0	00

1.3.7 多周期微程序控制器

- (1) 状态编码, 8 条指令共 13 个执行阶段, 采用 4 位二进制编码表示微指令地址, 如下表 1.14:

华中科技大学课程实验报告

表 1.14 状态编码表

微指令	状态	地址
取指令	S0	0
译码	S1	1
LW1	S2	2
LW2	S3	3
LW3	S4	4
SW1	S5	5
SW2	S6	6
R型运算	S7	7
R型运算	S8	8
Beq	S9	9
Bne	S10	10
ADDI1	S11	11
ADDI2	S12	12
SYSCALL	S13	13

- (2) 微指令地址转移逻辑，采用下址字段法，配合判断状态 P，取指阶段 P 为 1，表明控制存储器的地址由指令的第一个阶段的微程序地址给出，执行阶段 P 为 0，表明下一条微指令地址由下址字段给出，得出取指阶段指令的地址转移逻辑如下表 1.15：

表 1.15 微指令地址转移逻辑

R_Type	ADDI	LW	SW	BEQ	BNE	SYSCALL	微程序入口地址 (10进制)	S3	S2	S1	S0
1	X	X	X	X	X	X	7	0	1	1	1
X	1	X	X	X	X	X	11	1	0	1	1
X	X	1	X	X	X	X	2	0	0	1	0
X	X	X	1	X	X	X	5	0	1	0	1
X	X	X	X	1	X	X	9	1	0	0	1
X	X	X	X	X	1	X	10	1	0	1	0
X	X	X	X	X	X	1	13	1	1	0	1

- (3) 生成的地址转移逻辑表达式为：

$$S3 = \text{ADDI} + \text{BEQ} + \text{BNE} + \text{SYSCALL};$$

$$S2 = \text{R_Type} + \text{SW} + \text{SYSCALL};$$

$$S1 = \text{R_Type} + \text{ADDI} + \text{LW} + \text{BNE};$$

$$S0 = \text{R_Type} + \text{ADDI} + \text{SW} + \text{BEQ} + \text{SYSCALL}。$$

- (4) 控制存储器 16 进制微指令生成，根据 1.3.6 的分析，8 条指令执行需要的控制信号需要 16 位编码，而采用下址字段法，下址所需编码位数和微指令地址位数相同，即 4 位，一位判断标志位 P，P 为 0 表示取指阶段，

华中科技大学课程实验报告

控制存储器地址由指令的微指令转移逻辑产生，P 为 1 表示控制存储器地址由下址字段产生，即表示指令的不同执行阶段（即每条指令对应多条微指令），因此微指令编码 21 位，地址位编码 4 位，按 1.3.6 的控制信号编码，产生的微指令如下表 1.16:

表 1.16 微指令编码

微指令	十六进制
000010011001000000001	13201
000110000000000010000	30010
001100000000000000011	60003
100000000001000000100	100204
000001000100000000000	8800
0011000000000000000110	60006
100000000010000000000	100400
001000000000001001000	40048
000000100100000000000	4800
011000000000100100000	C0120
011000000000010100000	C00A0
0011000000000000001100	6000C
000000000100000000000	800
0000000000000000001101	D

1.3.8 多周期硬布线控制器

- (1) 状态编码，和微程序控制器编码相同；
- (2) 状态转移，，增加有限状态机 FSM 实现现态到次态的转换，次态即下一条微指令在控制存储器中的地址，而 FSM 采用纯组合逻辑电路实现，中间是具体的连线，这也是“硬布线”的由来，FSM 状态转换如表 1.17；

表 1.17 FSM 状态转换

现态 10进制	S3	S2	S1	S0	R_Type	LW	SW	BEQ	BNE	SYSCALL	ADDI	次态 10进制	N3	N2	N1	N0
0	0	0	0	0	X	X	X	X	X	X	X	1	0	0	0	1
1	0	0	0	1	1	X	X	X	X	X	X	7	0	1	1	1
1	0	0	0	1	X	1	X	X	X	X	X	2	0	0	1	0
1	0	0	0	1	X	X	1	X	X	X	X	5	0	1	0	1
1	0	0	0	1	X	X	X	1	X	X	X	9	1	0	0	1
1	0	0	0	1	X	X	X	X	1	X	X	10	1	0	1	0
1	0	0	0	1	X	X	X	X	X	1	X	13	1	1	0	1
1	0	0	0	1	X	X	X	X	X	X	1	11	1	0	1	1
2	0	0	1	0	X	X	X	X	X	X	X	3	0	0	1	1
3	0	0	1	1	X	X	X	X	X	X	X	4	0	1	0	0
4	0	1	0	0	X	X	X	X	X	X	X	0	0	0	0	0
5	0	1	0	1	X	X	X	X	X	X	X	6	0	1	1	0
6	0	1	1	0	X	X	X	X	X	X	X	0	0	0	0	0
7	0	1	1	1	X	X	X	X	X	X	X	8	1	0	0	0
8	1	0	0	0	X	X	X	X	X	X	X	0	0	0	0	0
9	1	0	0	1	X	X	X	X	X	X	X	0	0	0	0	0
10	1	0	1	0	X	X	X	X	X	X	X	0	0	0	0	0
11	1	0	1	1	X	X	X	X	X	X	X	12	1	1	0	0
12	1	1	0	0	X	X	X	X	X	X	X	0	0	0	0	0
13	1	1	0	1	X	X	X	X	X	X	X	13	1	1	0	1

华中科技大学课程实验报告

(3) 硬布线控制存储器，地址编码 4 位，由于状态转移已经由 FSM 给出，因此不再需要下址字段和判断字段 P，可减少为 16 位二进制编码，当然本实验中为减少工作量，可同样采用 21 位二进制编码，弃用下址字段和 P 即可，从而硬布线控存和微程序控存相同，在这里不再重复贴出。

1.3.9 多周期 CPU 总体结构图

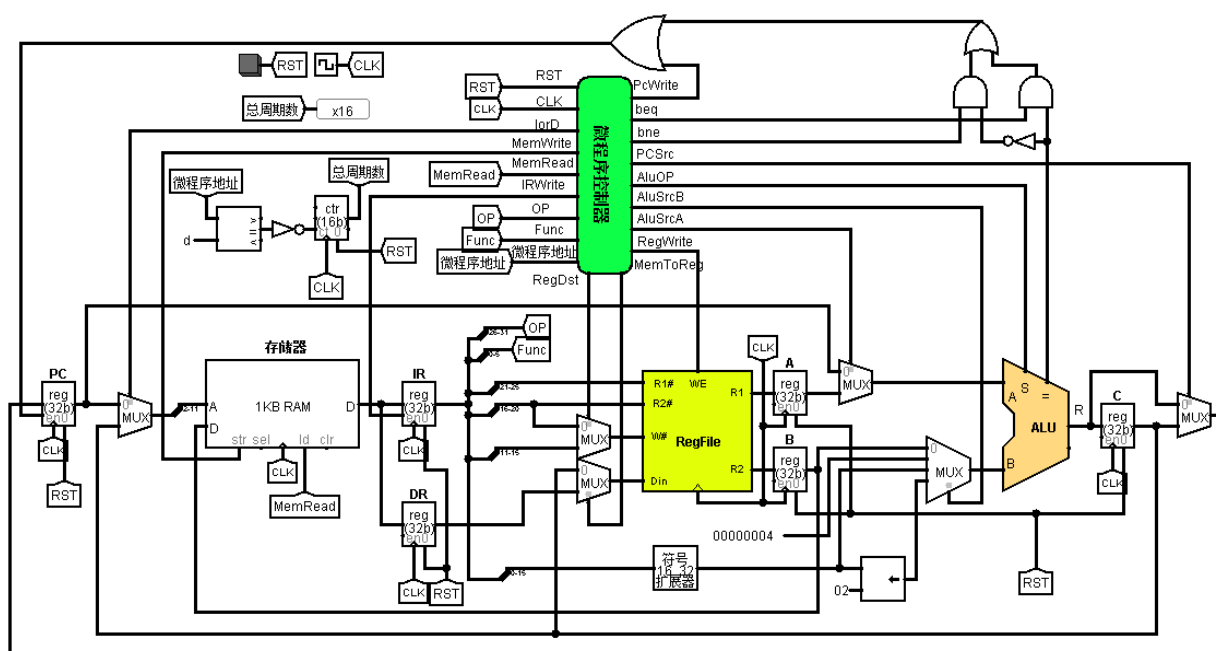


图 1.4 MIPS 多周期 CPU 总体结构图

1.3.10 一些细节

(1) 单周期 AluOP 产生：

上面加法，下面slt

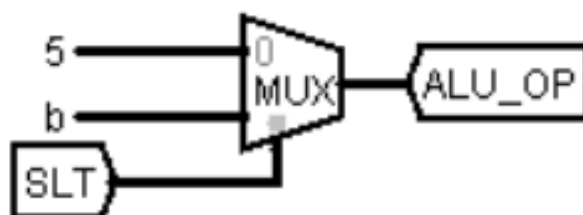


图 1.5 MIPS 单周期 CPU ALUOp 产生电路

(2) 单周期 PCBranch 信号产生;

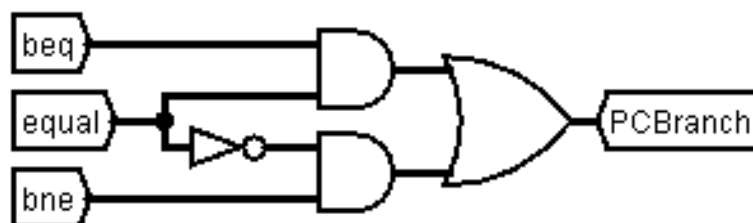


图 1.6 MIPS 单周期 CPU 跳跃寻址信号 PCBranch 产生电路

(3) 单周期周期计数模块;

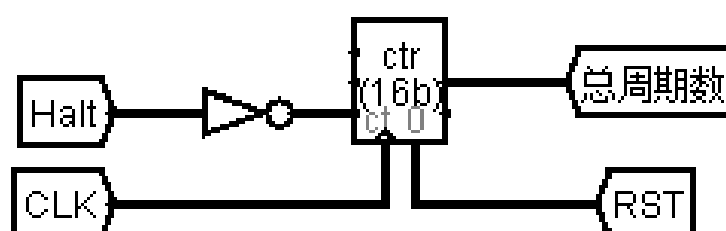
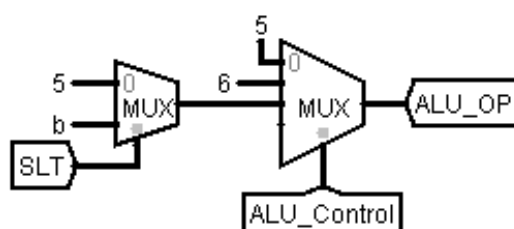


图 1.7 MIPS 单周期 CPU 周期计数

(4) 多周期 AluOp 产生;



ALU_Control= 00 运算器做加法

ALU_Control= 01 运算器做减法

ALU_Control= 10 运算方式由Func决定

图 1.8 MIPS 多周期 CPU ALUOp 产生电路

(5) 多周期 PC 写使能信号 PCEn 产生;

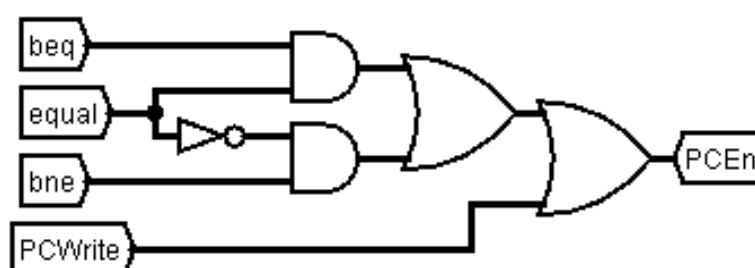


图 1.9 MIPS 多周期 CPU PC 使能信号 PCEn 产生电路

(6) 多周期微程序周期计数模块，微指令为 13 (0xd) 时，系统停机；

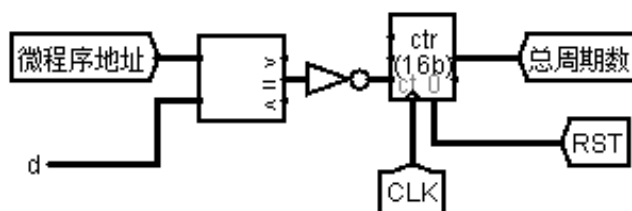


图 1.10 MIPS 多周期 CPU 微程序周期计数

(7) 多周期硬布线周期计数模块，状态为 13 (0xd) 时，系统停机；

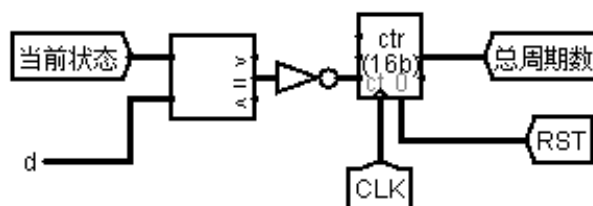


图 1.11 MIPS 多周期 CPU 硬布线周期计数

1.4 故障与调试

1.4.1 单周期数据存储器地址错误

故障现象：冒泡排序 sort.hex 程序执行完毕后，没有在指定的 80 号存储单元形成降序排列的数据，而是在 200 号存储单元形成降序排列数据，而且两个数据地址之差为 4。

200	00000006	00000000	00000000	00000000
204	00000005	00000000	00000000	00000000
208	00000004	00000000	00000000	00000000
20c	00000003	00000000	00000000	00000000
210	00000002	00000000	00000000	00000000
214	00000001	00000000	00000000	00000000
218	00000000	00000000	00000000	00000000
21c	ffffffff	00000000	00000000	00000000

图 1.12 存储地址错误

原因分析：Logisim 中 RAM 只支持一种访问模式，一次访问读出 32 位数据，直接给出字地址使得内存布局错误，如下图：

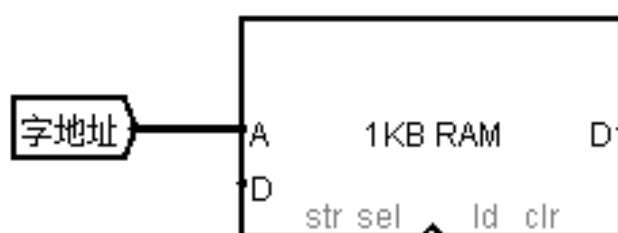


图 1.13 直接字地址访问

解决方案：字地址除以 4 即可得到正确的内存布局，即高两位补零作为字节地址送入存储器地址即可，如下图：

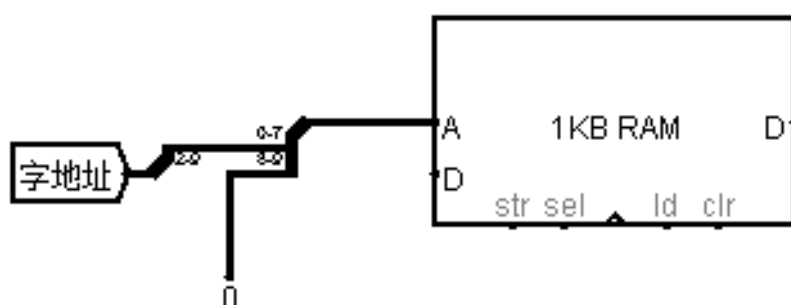


图 1.14 字节地址访问

1.4.2 多周期 CPU 循环震荡

故障现象：多周期 CPU 运行 sort.hex 程序时，能正确得到相应的内存布局，但是遇到停机指令无法停机，PCEn 使能信号产生振荡，PC 循环变化；

原因分析：指令译码错误，将 syscall 指令也当成了 R 型指令，导致控制器中没有产生正确的 PCWrite 信号，也没有进入 syscall 的死循环，从而使得 PCEn 信号出现震荡，系统无法停机；

解决方案：区分 syscall 指令和 R 型指令，使 syscall 指令与 R 型指令互斥，如下图：

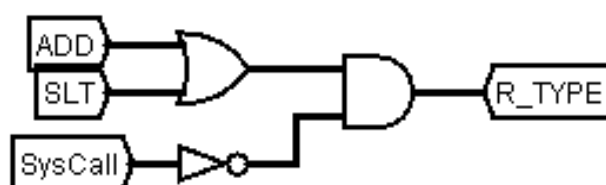


图 1.15 syscall 与 R 型互斥

1.5 测试与分析

1.5.1 单周期控制信号测试

- (1) 测试电路如下图，ROM 中依次存放了指令 add、slt、addi、lw、sw、beq、bne、syscall 的 op 和 func 字段；

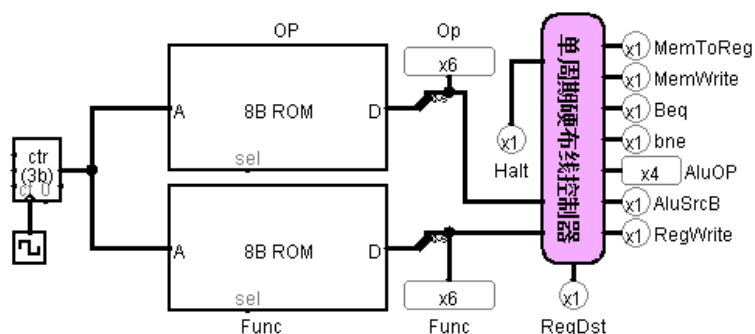


图 1.16 单周期控制器测试电路

- (2) 利用 Logisim 中记录器功能，具体为菜单->模拟->记录器，将探测器添加记录，设置基数为二进制即可，具体的 log 如下图，对比知结果正确。

单周期控制器.log - 记事本										
文件(F)	编辑(E)	格式(O)	查看(V)	帮助(H)						
Op	Func	RegDst	RegWrite	AluSrcB	AluOP	bne	Beq	MemWrite	MemToReg	Halt
000000	101010	1	1	0	1011	0	0	0	0	0
001000	000000	0	1	1	0101	0	0	0	0	0
100011	000000	0	1	1	0101	0	0	0	1	0
101011	000000	0	0	1	0101	0	0	1	0	0
000100	000000	0	0	0	0101	0	1	0	0	0
000101	000000	0	0	0	0101	1	0	0	0	0
000000	001100	0	0	0	0101	0	0	0	0	1

图 1.27 单周期控制器测试结果

1.5.2 单周期 CPU 执行 sort.hex

- (1) 内存布局，在 80 号单元开始处出现 6,5,4,3,2,1,ffff 的有符号降序数据，如图：

078	00000000	00000000	00000000	00000000
07c	00000000	00000000	00000000	00000000
080	00000006	00000005	00000004	00000003
084	00000002	00000001	00000000	ffffff
088	00000000	00000000	00000000	00000000
08c	00000000	00000000	00000000	00000000

图 1.18 单周期执行 sort.hex 后内存布局

华中科技大学课程实验报告

(2) 时钟周期，执行完毕后，系统停机，sort.hex 的时钟周期数为 224，如图：

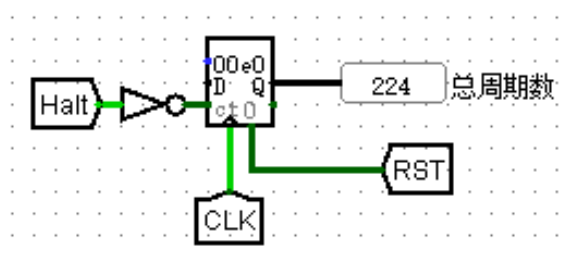


图 1.19 单周期执行 sort.hex 所需时钟周期数

1.5.3 多周期微程序控制信号测试

(1) 测试电路如下图，ROM 中依次存放了指令 add、slt、addi、lw、sw、beq、bne、syscall 的 op 和 func 字段；

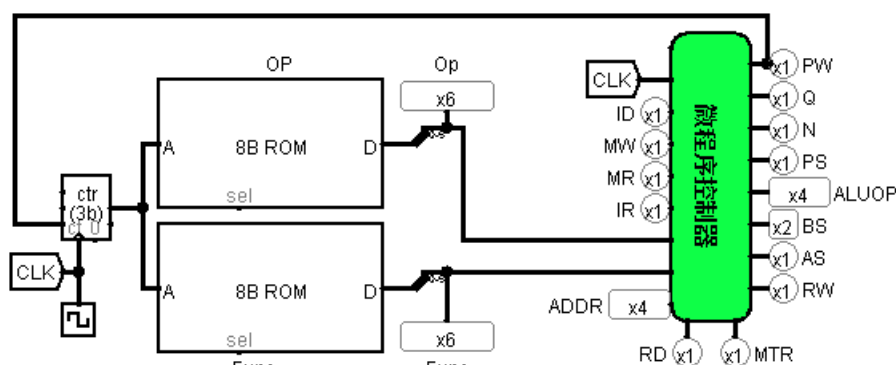


图 1.20 多周期微程序控制器测试电路

(2) 利用 Logisim 中记录器功能，具体为菜单->模拟->记录器，将探测器添加记录，设置基数为二进制即可，RD 指 RegDst，MTR 指 MemToReg，RW 指 RegWrite，AS 指 AluSrcA，BS 指 AluSrcB，PS 指 PCSrc，ID 指 IorD，MW 指 MemWrite，MR 指 MemRead，IR 指 IRWrite，ADDR 指微程序地址，具体的 log 如下图，对比知结果正确。

华中科技大学课程实验报告

多周期控制器.log - 记事本

文件(F)	编辑(E)	格式(O)	查看(V)	帮助(H)												
OP	FUNC	RD	MTR	RW	AS	BS	ALUOP	PS	N	Q	PW	ID	MW	MR	IR	ADDR
000000	100000	0	0	0	0	01	0101	0	0	0	1	0	0	1	1	0
000000	101010	0	0	0	0	11	0101	0	0	0	0	0	0	0	0	1
000000	101010	0	0	0	1	00	1011	0	0	0	0	0	0	0	0	7
000000	101010	1	0	1	0	00	0101	0	0	0	0	0	0	0	0	8
000000	101010	0	0	0	0	01	0101	0	0	0	1	0	0	1	1	0
001000	000000	0	0	0	0	11	0101	0	0	0	0	0	0	0	0	1
001000	000000	0	0	0	1	10	0101	0	0	0	0	0	0	0	0	11
001000	000000	0	0	1	0	00	0101	0	0	0	0	0	0	0	0	12
001000	000000	0	0	0	0	01	0101	0	0	0	1	0	0	1	1	0
100011	000000	0	0	0	0	11	0101	0	0	0	0	0	0	0	0	1
100011	000000	0	0	0	1	10	0101	0	0	0	0	0	0	0	0	2
100011	000000	0	0	0	0	00	0101	0	0	0	0	1	0	1	0	3
100011	000000	0	1	1	0	00	0101	0	0	0	0	0	0	0	0	4
100011	000000	0	0	0	0	01	0101	0	0	0	1	0	0	1	1	0
101011	000000	0	0	0	0	11	0101	0	0	0	0	0	0	0	0	1
101011	000000	0	0	0	1	10	0101	0	0	0	0	0	0	0	0	5
101011	000000	0	0	0	0	00	0101	0	0	0	0	1	1	0	0	6
101011	000000	0	0	0	0	01	0101	0	0	0	1	0	0	1	1	0
000100	000000	0	0	0	0	11	0101	0	0	0	0	0	0	0	0	1
000100	000000	0	0	0	1	00	0110	1	0	1	0	0	0	0	0	9
000100	000000	0	0	0	0	01	0101	0	0	0	1	0	0	1	1	0
000101	000000	0	0	0	0	11	0101	0	0	0	0	0	0	0	0	1
000101	000000	0	0	0	1	00	0110	1	1	0	0	0	0	0	0	10
000101	000000	0	0	0	0	01	0101	0	0	0	1	0	0	1	1	0
000000	001100	0	0	0	0	11	0101	0	0	0	0	0	0	0	0	1
000000	001100	0	0	0	0	00	0101	0	0	0	0	0	0	0	0	13

图 1.21 多周期控制器测试结果

1.5.4 多周期 CPU 执行 sort.hex

- (1) 内存布局，在 80 号单元开始处出现 6,5,4,3,2,1,ffff 的有符号降序数据，如图：

000	2010ffff	20110000	ae300200	22100001	22310004	ae300200	22100001	22310004
008	ae300200	22100001	22310004	ae300200	22100001	22310004	ae300200	22100001
010	22310004	ae300200	22100001	22310004	ae300200	22100001	22310004	ae300200
018	00008020	2011001c	8e130200	8e340200	0274402a	11000002	ae330200	ae140200
020	2231ffff	1611ffff	22100004	2011001c	1611ffff	2002000a	0000000c	00000000
028	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
030	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
038	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
040	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
048	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
050	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
058	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
060	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
068	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
070	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
078	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
080	00000006	00000005	00000004	00000003	00000002	00000001	00000000	ffffff
088	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000

图 1.22 多周期 CPU 执行 sort. hex 后内存布局

华中科技大学课程实验报告

(2) 时钟周期，执行完毕后，系统停机，sort.hex 的时钟周期数为 891，如图：

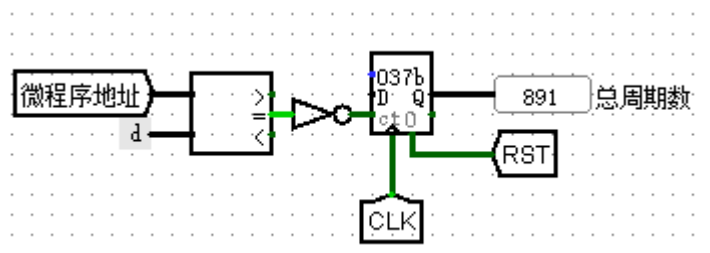


图 1.23 多周期 CPU 执行 sort. hex 所需时钟周期数

2 总结与心得

2.1 实验总结

本次实验主要完成了如下几点工作：

- 1) 实现了 32 位定长指令的解析；
- 2) 实现了立即数扩展；
- 3) 实现了程序执行指令数目的记录；
- 4) 实现了寄存器读写；
- 5) 实现了数据存储器的读写；
- 6) 实现了操作数的计算；
- 7) 实现了控制信号的生成；
- 8) 实现了各部件多输入来源的多选一；
- 9) 实现了 MIPS 多周期 CPU 控制存储器；
- 10) 实现了 MIPS 多周期 CPU 微程序地址转移逻辑；
- 11) 实现了 MIPS 多周期 CPU 硬布线控制器状态机；
- 12) 完成了 MIPS CPU 数据通路综合，最终的 CPU 能完整支持 MIPS 核心指令集。

2.2 实验心得

- 1) 学会了使用工程化方法生成电路，了解了 logisim 常用 debug 方法；
- 2) 宏观上掌握了 CPU 的运行机理和模块划分，而且还学会了从微观上准确地生成每一个控制信号；
- 3) 熟悉了 MIPS 核心指令，对 MIPS 的三类指令有了较为大概的实现思路；
- 4) 实验最大的感受是对于课程内容理解的加深。在上完课以后往往没有真的学懂知识，在做实验的过程中，自然就会主动地复习上课讲过的内容，而且必须真的弄懂才能完成实验；
- 5) 实验里有一点不好的体验就是文档更新不及时，常常不知道实验需求，建

华中科技大学课程实验报告

议老师在就实验过程中同学反馈的关键点在课程学习群中发布公告，或者以文档形式实时更新，供其他同学参考；

- 6) 有个建议是关于 QQ 群的，我们很理解老师将 QQ 群设为匿名的初衷，是希望更多的同学能够勇敢没有顾虑的问问题，但是群里总是有很多同学借助匿名的方便去讨论其他没有营养的东西，甚至还有同学匿名攻击别人，这应该不是老师想看到的吧，我的建议是在 QQ 群里多设置管理员（比如每个班的班长或者学习委员），管理员可以禁言言辞不当的同学，这样不光可以让同学们大胆问问题，也会让同学们自觉去遵守聊天纪律。

参考文献

- [1] DAVID A. PATTERSON(美). 计算机组成与设计硬件/软件接口(原书第 4 版). 北京: 机械工业出版社.
- [2] David Money Harris(美). 数字设计和计算机体系结构(第二版). 机械工业出版社
- [3] 秦磊华, 吴非, 莫正坤. 计算机组成原理. 北京: 清华大学出版社, 2011 年.
- [4] 袁春风编著. 计算机组成与系统结构. 北京: 清华大学出版社, 2011 年.
- [5] 张晨曦, 王志英. 计算机系统结构. 高等教育出版社, 2008 年.

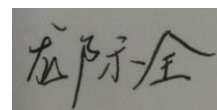
• 指导教师评定意见 •

一、原创性声明

本人郑重声明本报告内容，是由作者本人独立完成的。有关观点、方法、数据和文献等的引用已在文中指出。除文中已注明引用的内容外，本报告不包含任何其他个人或集体已经公开发表的作品成果，不存在剽窃、抄袭行为。

特此声明！

作者签字：



二、对课程实验的学术评语（教师填写）

三、对课程实验的评分（教师填写）

评分项目 (分值)	报告撰写 (30 分)	课设过程 (70 分)	最终评定 (100 分)
得分			

指导教师签字：_____

2019-05-04