

计算机组成原理课程设计 实验报告

单周期处理器设计与实现

学号 112015XXXX

姓名 XXX

班级 071115XX

二零一八年九月十八号

单周期处理器设计与实现

一、 设计目的

1. 了解现代计算机硬件设计的基本流程和方法。
2. 了解典型的 RISC 处理器 MIPS 的体系结构。
3. 了解汇编语言到机器语言到计算机执行软硬的逻辑关系。
4. 掌握处理器的设计原理和方法。
5. 培养寄存器级硬件故障的检错和排错能力。

二、 设计内容

计单周期控制的处理器，支持 MIPS 指令子集:Lui, Addiu, Add, Lw, Sw, Beq, j，以及一条随机抽取的指令。为了简化模型，指令可以不支持溢出。

随机抽取的指令为: bgtz.

三、 设备器材

操作系统: Windows 10

仿真软件: ModelSim

四、 设计原理及内容

单周期 CPU 指的是一条指令的执行在一个统一周期内完成，然后开始下一条指令的执行，即所有指令用一个周期完成。电平从低到高变化的瞬间称为时钟上升沿，两个相邻时钟上升沿之间的时间间隔称为一个时钟周期。时钟周期一般也称为振荡周期。

CPU 在处理指令时，一般需要经过以下几个步骤：

(1) 取指令(IF)：根据程序计数器 PC 中的指令地址，从存储器中取出一条指令，同时，PC 根据指令字长度自动递增产生下一条指令所需要的指令地址，但遇到“地址转移”指令时，则控制器把“转移地址”送入 PC，当然得到的“地址”需要做些变换才送入 PC。

(2) 指令译码(ID)：对取指令操作中得到的指令进行分析并译码，确定这条指令需要完 成的操作，从而产生相应的操作控制信号，用于驱动执行状态中的

各种操作。

(3) 指令执行(EXE)：根据指令译码得到的操作控制信号，具体地执行指令动作，然后 转移到结果写回状态。

(4) 存储器访问(MEM)：所有需要访问存储器的操作都将在这个步骤中执行，该步骤给 出存储器的数据地址，把数据写入到存储器中数据地址所指定的存储单元或者从存储器中得 到数据地址单元中的数据。(5) 结果写回(WB)：指令执行的结果或者访问存储器中得到的数据写回相应的目的寄存器中。单周期 CPU，是在一个时钟周期内完成这五个阶段的处理。

单周期的数据通路图：

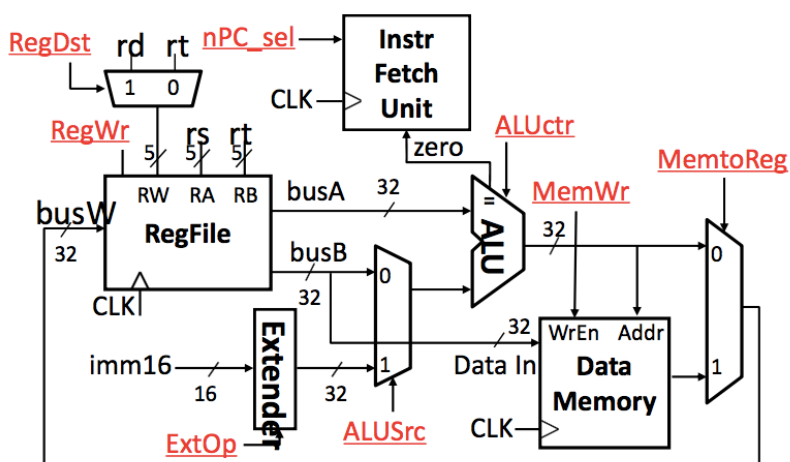


图 4-1 单周期处理器数据通路图

五、 设计步骤

整个实验的核心便是上面的数据通路图。用模块化的思想方法，不难将上面的数据通路图进行分解。

对于每一个模块，我们只关心如何从他的输入得到他的输出，并将其中的逻辑过程转换为代码。例如，对于立即数扩展模块，我们只关心如何从 16 位的立即数输入和 ExtSel 信号输入这两个条件，确定该模块的输出：32 位的扩展后的立即数。根据所学的知识，不难确定其中的逻辑。当 ExtSel 为 0 时，对 16 位的立即数进行零扩展；当 ExtSel 为 1 时，对 16 位的立即数进行符号扩展。于是，这个模块的内容，便是这个逻辑转换而成的代码。

分解后的模块相互独立，并不能驱动 CPU 运转。为此，还要设计一个 CPU 顶层模块 mips， 将这些模块通过线，有条不紊地连接起来，形成流水线，提高吞

吐量。同时，还可以暴露各种控制单元信号和线的值，便于调试和验证。

为进行测试，还需要设计一个测试模块 testbench。测试模块实例化 CPU，传入时钟信号和 Reset 信号，可以从这里观测各个信号的取值情况，绘制波形图。

因为老师已经给出了一个支持七条指令的处理器 demo，所以我就直接在老师给的 demo 的基础上添加了我抽取到的指令——bgtz。

5.1 获得指令行为

首先通过公众号查得 bgtz 的指令说明如下：

bgtz: 大于 0 时转移


编码	31	26	25	21	20	16	15	0
	bgtz 000111		rs		0 00000		offset	
	6		5		5		16	
格式	bgtz rs, offset							
描述	if (GPR[rs] > 0) then 转移							
操作	if (GPR[rs] > 0) PC ← PC + sign_extend(offset 0 ²) else PC ← PC + 4							
示例	bgtz \$s1, -2							
其他	 北理计算机系统能力实践教学							

图 5-1 bgtz 指令说明

结合上面的数据通路图，可以得知 bgtz 指令影响的控制信号有：op（识别 bgtz 指令），nPC_sel（关系到 nPC 和 PC 的取值），nPCop（接收 nPC_sel），RS1_0（判断 rs 的内容是否大于 0），nPC（下一条指令地址），PC（当前指令地址）。

5.2 实现

1. 新建项目：

Project - D:/altera/13.0/SingleCPU/SingleCPU					
Name	Status	Type	Order	Modified	
code.txt		Text	-	01/12/14 11:47:48 AM	
controller		Folder			
ctrl.v	✓	Verilog	10	08/22/17 05:47:22 PM	
datapath		Folder			
alu.v	✓	Verilog	0	08/22/17 10:31:08 AM	
dm.v	✓	Verilog	1	01/11/14 02:09:20 PM	
im.v	✓	Verilog	2	01/08/14 07:26:26 PM	
instruction_he...	✓	Verilog	3	09/02/18 03:01:13 PM	
Mux.v	✓	Verilog	4	08/22/17 05:09:46 PM	
nPC.v	✓	Verilog	5	08/22/17 06:01:28 PM	
PC.v	✓	Verilog	6	01/10/14 08:38:00 PM	
processor_he...	✓	Verilog	7	01/09/14 08:49:28 PM	
regfile.v	✓	Verilog	8	01/12/14 11:35:20 AM	
sign_Ext.v	✓	Verilog	9	01/11/14 04:26:34 PM	
mips.v	✓	Verilog	12	01/12/14 11:16:36 AM	
testbench.v	✓	Verilog	11	08/22/17 06:11:04 PM	

按照老师提示的结构导入各个.v 文件，并且同时导入待测用例 code.txt。

2. 顶层实体模块：

```
nPC U_nPC(imm_16,imm_26,PCout,npc_sel,PCin,PC_add_4,RS1out);

//module PC (rst,clk,nPC,PC);
PC U_PC (rst,clk,PCin,PCout);

//module RegFile( CLK_I, RS1_I, RS2_I, RD_I, RegWr_I,WData_I,RS1_O, RS2_O);
RegFile U_RF (clk,RS,RT,mux4_5out,RegWrt,mux4_32out,RS1out,RS2out);
```

bgtz 是跳转指令，其跳转执行的位置与 RS 寄存器的内容有关，处理下一条指令地址的模块是 nPC 模块. 所以我们需要将 RegFile 读取的 RS 寄存器内容 RS1out 传递给 nPC 模块, 对于指令 bgtz, 该值将影响到下一条 PC。

3. 控制单元模块：

```
wire Rtype,add,addiu,lw,sw,j,beq,bgtz; //Rtype,addu,subu,ori,lw,sw,beq,lui,addi,addiu,slt,j,jal,jr,lb,lbu,lh,lhu,sb,sh,
//wire add,sub,sll,srl,sra,sllv,srlv,srav,AND,OR,XOR,NOR,andi,xori,sltiu,bne,blez,bgtz,bltz,bgez,jalr; //added
//???
assign Rtype = (op==6'b000000)?1:0;
assign add = (Rtype&&funct==6'b100000)?1:0; //&& ???it's better to be included in "head.v"
assign sub = (Rtype&&funct==6'b100010)?1:0;
assign lw = (op==6'b100011)?1:0;
assign sw = (op==6'b101011)?1:0;
assign beq = (op==6'b000100)?1:0;
assign lui = (op==6'b001111)?1:0;
assign addiu = (op==6'b001001)?1:0;
assign j = (op==6'b000010)?1:0;
assign bgtz = (op==6'b000111)?1:0;

assign DMWrite = (sw)?'b1:'b0;
assign npc_sel = (j)?'b01:(beq&&beqout)?'b11:(bgtz)?'b10:2'b00; //singular valuebeq?11
assign RegWrt = (add||addiu||lw||lui)?'b1:'b0; //JAL????????????
assign ExtOp = (lui)?'b00:(lw||sw)?'b10:2'b10;
assign mux4_5sel = (addiu||lui||lw)?'b00:2'b01; //add
assign mux2sel = (lw||sw||addiu)?'b1:'b0; //beq
assign mux4_32sel = (lui)?'b11:(add||addiu)?'b00:(lw)?'b01:2'b10; //?????pc+4????JAL

assign ALUctr = (add||addiu||lw||sw)?'b001:(beq)?'b010:3'b000;
```

在此添加 bgtz 识别机制：当 op 为 000111 时，此时指令为 bgtz。bgtz 影响到下一条指令地址的选择，故将 10 编码添加到 npc_sel 变量中。

4. nPC 模块：

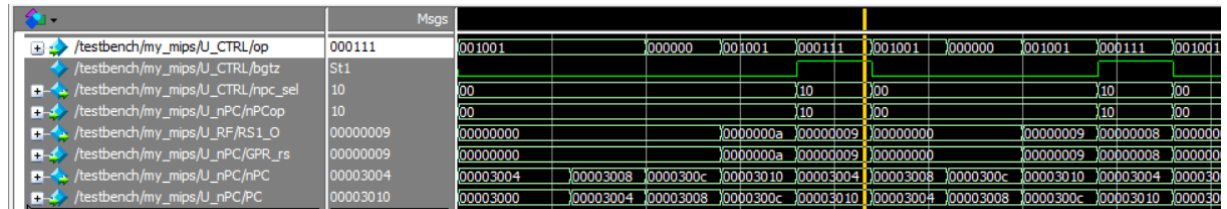
```
module nPC (imm_16,imm_26,PC,nPCop,nPC,PC_add_4,GPR_rs)
    input [25:0] imm_26 ;
    input [15:0] imm_16 ;
    input [31:0] PC;
    input [1:0] nPCop;
    input [31:0] GPR_rs;
    output [31:0] nPC;           //output to IM
    output [31:0] PC_add_4;     //output to $31

    assign nPC = (nPCop == 2'b00)?PC_add_4:
        (nPCop == 2'b01)?(PC[31:28],imm_26,2'b00)://j
        (nPCop == 2'b10&&GPR_rs>0)?(PC+{{14{imm_16[15]}},imm_16,2'b00}):`DEBUG_DEV_DATA
        (nPCop == 2'b10&&GPR_rs<=0)?(PC add 4):
        (nPCop == 2'b11)?(PC_add_4+{{14{imm_16[15]}},imm_16,2'b00}):`DEBUG_DEV_DATA ; //be
```

nPC 控制指令的执行顺序，处理顺讯执行和跳转行为。bgtz 属于跳转类型指令，其跳转位置和 rs 寄存器内容有关，在此我们读取其内容 GPR_rs，若 GPR_rs>0，nPC 等于 PC 加上带符号的 16 位扩展，后两位补零；否则，程序顺序往下执行，即加 PC 加 4。

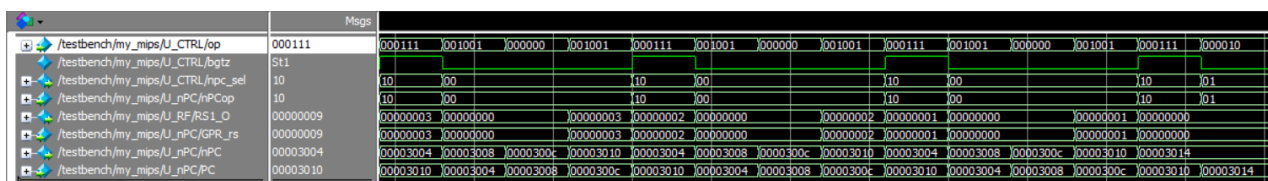
5.1 实验结果

使用老师提供的单周期测试程序测试结果如下：



如上图截取的部分波形，可以看出当执行到 op 为 000111 时，bgtz 变为 1，即该指令被正确识别；在 U_CTRL 中将 npc_sel 置为 10，并且将此值传递给 nPC 中的 nPCop，故两值均为 10；bgtz 指令需获取寄存器 rs 中的值，该值 U_RF 处理读出，即 RS1_O，传递给 nPC 时名为 GPR_rs，本次读出的数据为 9；因为 9>0，故执行 $PC \leq PC + \text{sign_extend}(\text{offset} || 0^2)$ ，故将 nPC 处理为 3004，即下一条指令的地址。

可以看出这是一个将 9 逐渐递减到 0 的循环，我们接下来看一下程序走到最后的波形：



从两张图可以看出程序不断在 3004/3008/300c/3010 之间循环，将 rs 中的内容 by 9 一直递减，当 $GPR_{rs} \leq 0$ 时，bgtz 应当执行 $PC \leq PC + 4$ ，即顺序往下执行，可以从波形图看出程序最后走出循环，到了 3014 的位置。

故验证了 bgtz 的单周期指令执行成功。

六、设计总结

本次实验是个人实验，完成一个单周期处理器的设计。因为老师已经给了设计手册以及支持七条指令的一个 demo，所以本次实验相对来说还是比较简单的，只要根据老师给出的单周期数据通路图，结合自己抽到的指令，修改每个和抽到的指令有关的控制信号即可，我抽到的指令是 bgtz，基本一步一步分解指令然后照着去修改受影响的相应模块就行了。

因为之前已经接触过 modelsim，仿真步骤也都得心应手，前期花费较多时间理解 CPU 和看懂各个模块上，入门之后一切都进展地比较顺利了。

经过本次实验，我充分理解到了处理器的工作流程，也算是对理论知识的一次巩固，希望今后还有机会能参与到完整的 CPU 的设计中去。