计算机组成原理实验报告

单周期处理器开发

班级：1616401班

学号：161640111

姓名：杜云

学院：计算机科学与技术学院

2018.06.01

**文档目录：**

1. 功能设计说明
2. 模块化和层次化设计说明
3. 具体模块定义
4. 测试代码及结果
5. 实验完成时间安排
6. 心得体会

**1. 功能设计说明**

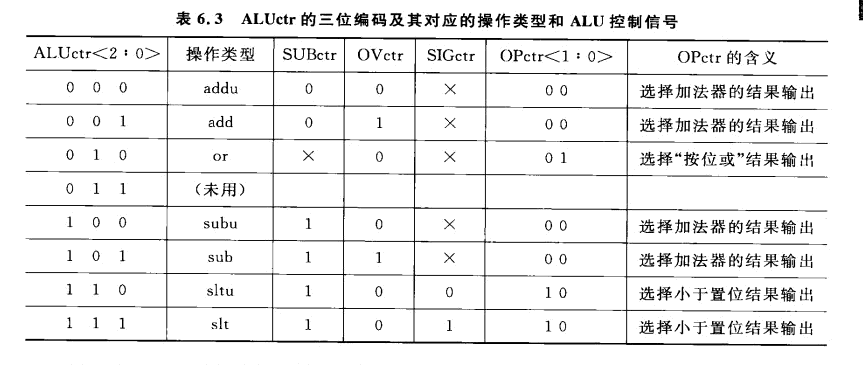
1.完成的指令集: add, sub, ori, lw, sw, beq, j

2.处理器为单周期设计。

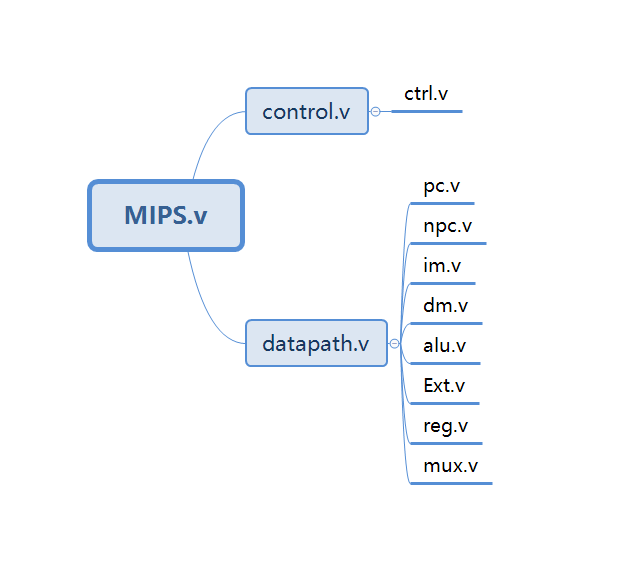
3.功能模块根据书上162页的图5-22设计，信号控制采用书上166页表5.4。

4.说明：不支持溢出。





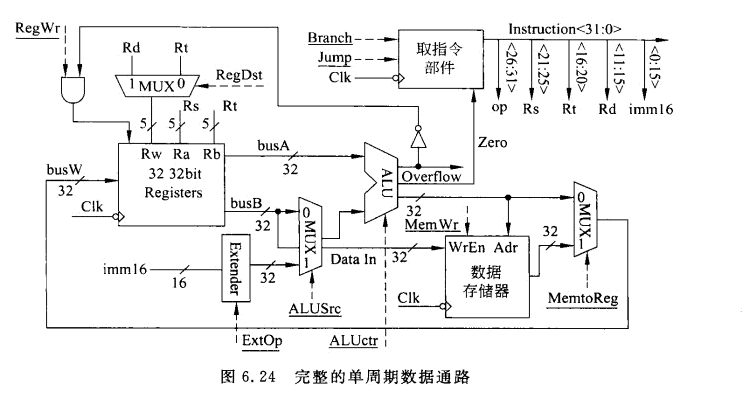
1. **模块化和层次化设计说明**



此图为模块化的说明，其中

1. Mips为信号文件，用于设置时钟以及控制信号以启动Cpu
2. Control是根据所选信号来控制所选指令
3. Datapath是数据通路，用于组织各底层文件
4. **具体模块定义**

3.1 所使用的数据通路：



* 1. **PC模块**

**3.2.1基本描述:**在时钟上升沿输出指令地址。当rst为1时复位为0x0000\_3000

**3.2.2模块接口**

|  |  |  |
| --- | --- | --- |
| **信号名** | **方向** | **描述** |
| **[31:0]NPC** | **I** | **输入的指令地址** |
| **clk** | **I** | **时钟信号** |
| **rst** | **I** | **复位信号** |
| **[31:0]PC** | **O** | **输出的指令地址** |

**3.2.3功能定义**

|  |  |  |
| --- | --- | --- |
| **序号** | **功能名称** | **功能描述** |
| **1** | **复位** | **rst=1时，将PC置为0X0000\_3000** |
| **2** | **输出指令地址** | **时钟信号到来时，将NPC赋给PC** |

* 1. **NPC模块**

**3.3.1基本描述:根据Branch,Jump信号输出下一条地址**

**3.3.2模块接口**

|  |  |  |
| --- | --- | --- |
| **信号名** | **方向** | **描述** |
| **[31:0]PC** | **I** | **输入当前指令地址** |
| **[15:0]immi16** | **I** | **指令的低16位** |
| **[25:0]target** | **I** | **J指令的跳转地址** |
| **Branch** | **I** | **是否为beq指令。1是0否** |
| **Jump** | **I** | **是否为j指令。1是0否** |
| **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。** |
| **[31:0]NPC** | **O** | **输出下一条指令地址** |

**3.3.3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和Jump的值输出下一条指令的地址。** |

* 1. **REG模块**

**3.4.1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。

**3.4.2模块接口**

|  |  |  |
| --- | --- | --- |
| **信号名** | **方向** | **描述** |
| **[4:0]Rs** | **I** | **Rs寄存器地址** |
| **[4:0]Rt** | **I** | **Rt寄存器地址** |
| **[4:0]Rw** | **I** | **Rw寄存器地址** |
| **RegWr** | **I** | **写寄存器使能信号** |
| **clk** | **I** | **时钟信号** |
| **[31:0]busA** | **O** | **Rs寄存器值** |
| **[31:0]busB** | **O** | **Rt寄存器值** |
| **[31:0]busW** | **I** | **写入寄存器的信号** |

**3.4.3功能定义**

|  |  |  |
| --- | --- | --- |
| **序号** | **功能名称** | **功能描述** |
| **1** | **读寄存器数据** | **读rs、rt寄存器的数据** |
| **2** | **向寄存器写入数据** | **根据写信号向寄存器选择写入数据** |

* 1. **ALU模块**

**3.5.1基本描述:** 数据快速存储模块，由输入的寄存器编号将相应的数据输出，或将数据存入相应的寄存器内

**3.5.2模块接口**

|  |  |  |
| --- | --- | --- |
| **信号名** | **方向** | **描述** |
| **[2:0]ALUctr** | **I** | **ALU控制信号** |
| **[31:0]busA** | **I** | **busA的值** |
| **[31:0]busB** | **I** | **busB的值** |
| **Zero** | **O** | 1. **B=0,则zero=1;**   **否则zero=0.** |
| **[31:0]ALUresult** | **O** | **Alu计算输出的结果。** |

**3.5.3功能定义**

|  |  |  |
| --- | --- | --- |
| **序号** | **功能名称** | **功能描述** |
| **1** | **输出计算结果** | **根据ALUctr，输出busA与busB的计算结果** |
| **2** | **输出Zero** | **若ALUresult=0**  **则输出Zero=1，否则输出Zero=0。** |

* 1. **MUX模块**

**3.6.1基本描述:** 根据控制信号选择数据

**3.6.2模块接口**

|  |  |  |
| --- | --- | --- |
| **信号名** | **方向** | **描述** |
| **busB[31:0]** | **I** | **busB的值** |
| **Ext\_immi16[31:0]** | **I** | **扩展后的立即数** |
| **ALUSrc** | **I** | **Mux控制信号** |
| **DataIn[31:0]** | **O** | **Mux输出** |
| **Rd[4:0]** | **I** | **Rd输入值** |
| **Rt[4:0]** | **I** | **Rt输入值** |
| **RegDst** | **I** | **Mux控制信号** |
| **Rw[4:0]** | **O** | **Mux输出** |
| **ALUresult[31:0]** | **I** | **ALU计算结果** |
| **DataOut[31:0]** | **I** | **DM的输出结果** |
| **MemtoReg** | **I** | **Mux控制信号** |
| **busW[31:0]** | **O** | **Mux输出** |

**3.6.3功能定义**

|  |  |  |
| --- | --- | --- |
| **序号** | **功能名称** | **功能描述** |
| **1** | **输出busB或Ext\_immi16的值** | **ALUSrc为0，y=busB;**  **ALUSrc为1，y= Ext\_immi16.** |
| **2** | **输出Rd或Rt的值** | **RegDst为0，y=Rt;**  **RegDst为1，y=Rd;** |
| **3** | **输出ALUresult或DataOut的值** | **MemtoReg为0，输出ALUresult的值；**  **MemtoReg为1，输出DataOut的值；** |

* 1. **DM模块**

**3.7.1基本描述：**“数据内存”大小为4G，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。

**3.7.2模块接口**

|  |  |  |
| --- | --- | --- |
| **信号名** | **方向** | **描述** |
| **clk** | **I** | **时钟信号** |
| **[31:0]addr** | **I** | **数据地址** |
| **[31:0]DataIn** | **I** | **写入的数据** |
| **MemWr** | **I** | **数据内存写信号** |
| **[31:0]DataOut** | **O** | **读出的数据** |

**3.7.3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. **IM模块**

**3.8.1基本描述:**指令内存大小为4G，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。

**3.8.2模块接口**

|  |  |  |
| --- | --- | --- |
| **信号名** | **方向** | **描述** |
| **[31:0]PC** | **I** | **指令地址** |
| **[31:0]Ins** | **O** | **指令** |

**3.8.3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** | **输出指令** | **根据输入的指令地址，输出当前指令** |

* 1. **EXT模块**

**3.9.1基本描述:** 将输入的16位地址按符号位扩展为32位。

**3.9.2模块接口**

|  |  |  |
| --- | --- | --- |
| **信号名** | **方向** | **描述** |
| **[15:0]** **imm16** | **I** | **输入的16位地址** |
| **ExtOp** | **I** | **扩展控制信号** |
| **[31:0]** **Ext\_immi16** | **O** | **输出的32位地址** |

**3.9.3功能定义**

|  |  |  |
| --- | --- | --- |
| **序号** | **功能名称** | **功能描述** |
| **1** | **输出扩展的32位地址.** | **将imm16扩展为32位的Ext\_immi16** |

* 1. **ctrl模块**

**3.10.1基本描述：输入指令，输出各模块的对应信号**

**3.10.2模块接口**

|  |  |  |
| --- | --- | --- |
| **信号名** | **方向** | **描述** |
| **[31:0]Ins** | **I** | **输入的指令** |
| **Branch** | **O** | **分支信号** |
| **Jump** | **O** | **跳转信号** |
| **RegDst** | **O** | **Rw选择信号** |
| **ALUSrc** | **O** | **ALU计算选择信号** |
| **MemtoReg** | **O** | **寄存器写入选择信号** |
| **RegWr** | **O** | **寄存器写使能信号** |
| **MemWr** | **O** | **DM写使能信号** |
| **ExtOp** | **O** | **扩展信号** |
| **ALUctr** | **O** | **ALU的功能信号** |

**3.10.3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. **Datapath模块**

**3.11.1基本描述：将ctrl的各种控制信号分别传入各模块**

**3.11.2模块接口**

|  |  |  |
| --- | --- | --- |
| **信号名** | **方向** | **描述** |
| **clk** | **I** | **时钟信号** |
| **rst** | **I** | **复位信号** |

**3.11.3功能定义**

|  |  |  |
| --- | --- | --- |
| **序号** | **功能名称** | **功能描述** |
| **1** | **将控制信号传入各模块** | **根据输入的clk以及rst，读入code.txt的指令，根据crtl解析各信号，传入各模块。** |

* 1. **Mips模块**

**3.12.1基本描述：设置时钟信号以及复位信号，启动datapath**

**3.12.2模块接口:无**

**3.12.3功能定义**

|  |  |  |
| --- | --- | --- |
| **序号** | **功能名称** | **功能描述** |
| **1** | **初始化** | **先将rst赋为1，初始化pc地址** |
| **2** | **设置时钟周期** | **每10ns，clk跳变一次** |

1. 测试代码及结果：

4.1 测试代码：

addiu $1 , $0 , 4 #$1寄存器赋值

addiu $2 , $0 , 5 #$2寄存器赋值

addiu $3 , $0, 8 #$3寄存器赋值

Start:add $4 , $1 , $1 #$4为$1加$3

add $5 , $2 , $4 #$5为$2加$4

sub $6 , $3 , $1 #$6为$3减$1

sub $7 , $5 , $3 #$7为$5减$3

ori $8 , $7 , 4 #$8为$7或4

or $9 , $8 , $1 #$9为$8或$1

beq $6 , $1 , Jump #若$6与$1相等，则跳转jump语句

slt $1 , $6 , $10

sw $1 , 0($3) #Mem[$3]=$1

Jump: slt $1 , $6 , $10

sw $1 , 0($3) #Mem[$3]=$1

lw $11 , 0($3) #$11=Mem[$3]

j Start #跳转start语句

4.2 对应二进制文件：

00100100000000010000000000000100

00100100000000100000000000000101

00100100000000110000000000001000

00000000001000010010000000100000

00000000010001000010100000100000

00000000011000010011000000100010

00000000101000110011100000100010

00110100111010000000000000000100

00000001000000010100100000100101

00010000110000010000000000000010

00000000110010100000100000101010

10101100011000010000000000000000

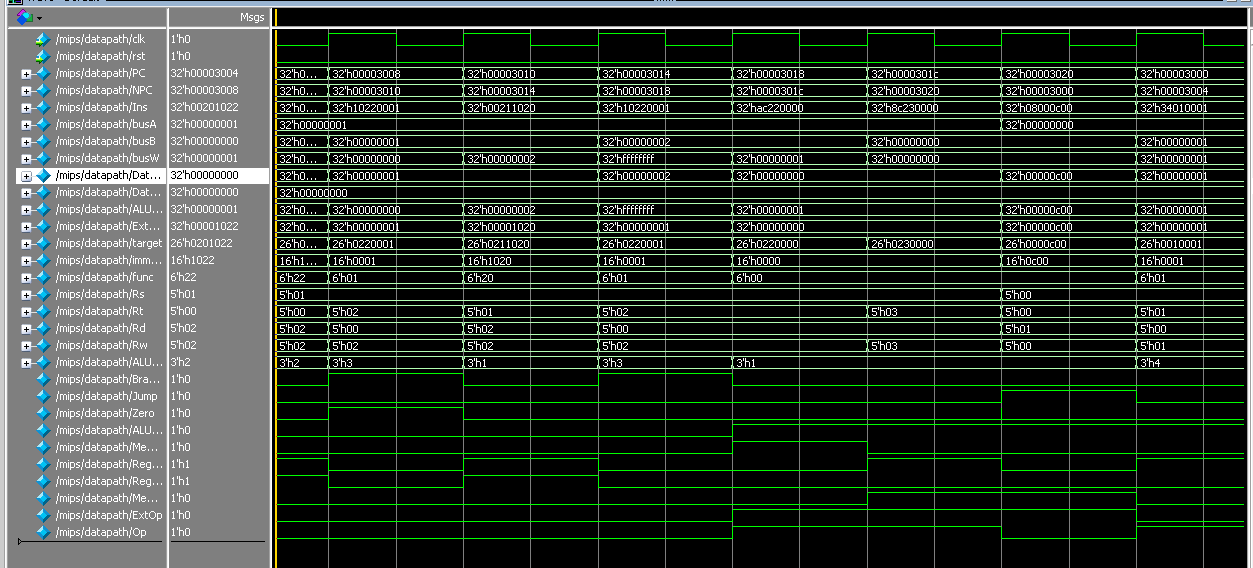
00000000110010100000100000101010

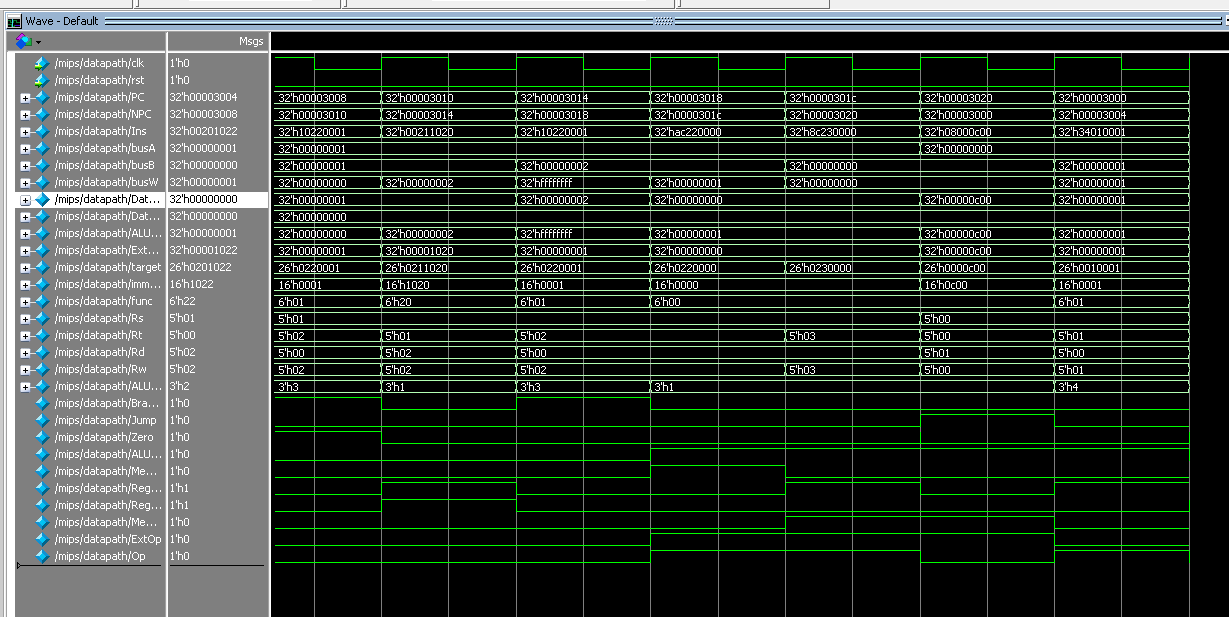
10101100011000010000000000000000

10001100011010110000000000000000

00001000000000000000110000000011

4.3波形展示：





4.4测试结果分析

通过观察每条通路控制指令的值以及最终输入到寄存器或是内存的值是否与预期相同来判断通路是否正确。例如执行R型指令时观察指令op以及相应的控制指令以及最终RegWD的数值来判断是否正确。

5 实验完成时间安排

1．试验前看书以及PPT，了解Datapath的整个流程，了解各个模块的连接。

2. 写代码时长：12小时

3. 调试时长：8小时

4. 总时长:20小时。

6、心得体会

我觉得这个实验最大的难处在于是否了解原理以及整体构造，Datapath的功能，各个模块是如何连接的以及如何使用控制信号完成模块的使用。当原理清晰之后，才能上手写代码。写完模块，对ctrl以及Datapath模块无从下手，本质就在于对原理不了解，对代码的整体分析还不够。之后凭借着和同学的交流以及学长的答疑，慢慢有种顿悟的感觉。Datapath里的那些实现的模块就相当于c语言里自己写的函数，利用ctrl模块对指令信号的分解产生控制信号传入其他模块。mips就是一个顶层的控制函数，初始化之后设置好clk,相当于启动整个Cpu。

经过单周期Cpu的设计，收获了很多，对单周期的Cpu有了新的也更深入的理解。对于代码的撰写也有一些感觉了，开始入门，在掌握了一些知识后，我有信心完成后面的流水线设计