P3实验报告

14061075 修闽珂

1. 模块定义
2. IFU模块定义
   1. 基本描述

IFU主要功能是完成取指令功能。IFU内部包括PC、IM(指令存储器)以及其他相关逻辑。IFU除了能执行顺序取值令外，还能根据BEQ指令的执行情况决定顺序取值令还是转移取值令。

* 1. 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| IfBeq | I | 当前指令是否为 beq 指令标志。  1：当前指令为 beq  0：当前指令非 beq |
| Zero | I | ALU 计算结果为 0 标志。  1：计算结果为 0  0：计算结果非 0 |
| clk | I | 时钟信号 |
| Reset | I | 复位信号。  1：复位  0：无效 |
| Instr[31:0] | O | 32 位 MIPS 指令 |

* 1. 功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 复位 | 当复位信号有效时，PC 被设置为 0x00000000。 |
| 2 | 取指令 | 根据 PC 从 IM 中取出指令。 |
| 3 | 计算下一条指令地址 | 如果当前指令不是 beq 指令，则 PCPC+1  如果当前指令是 beq 指令，并且 zero 为 0，则 PCPC+1  如果当前指令是 beq 指令，并且 zero 为 1 ，则  PCPC+sign\_ext(当前指令 15..0)  [注]PC 取地址为 4 字节，固低 2 位地址可以去除。 |

1. GPR模块定义
   1. 基本描述

GPR是一个32bit \* 32寄存器堆，寄存器带有写使能功能。能够根据读入的地址寻找相应寄存器的位置以及读出内部的值。当写使能有效时，能够在写入地址写入需要被储存的值。

* 1. 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| RS1 | I | 读取的第一个数据的寄存器地址 |
| RS2 | I | 读取的第二个数据的寄存器地址 |
| RegWrite | I | 写使能 |
| WA | I | 写入地址 |
| WData | I | 写入数据 |
| ResetReg | I | 寄存器重置 |

* 1. 功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 读数据 | 地址信号到来，寄存器输出这个地址储存的值 |
| 2 | 写数据 | 将数据写入到写入地址所示的寄存器中 |
| 3 | 清零 | 当这个信号不为零的时候，将所有寄存器中的值清零 |

1. ALU模块定义
   1. 基本描述

ALU模块是算数逻辑单元。在这个单元里面，能够进行加、减、按位或以及lui指令。数据输入到ALU后，同步进行这四种运算，由ALUCtrl来控制哪一个结果被输出。

* 1. 模块接口

|  |  |  |
| --- | --- | --- |
| c | 方向 | 描述 |
| A | I | 第一个运算数 |
| B | I | 第二个运算数 |
|  | I |  |
| ALUCtrl[3:0] | I | ALU控制器 |
| Result | O | 输出运算结果 |
| Zero | O | 运算结果是否为零的标志 |

1. 功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 加法 | 将第一个运算数和第二个运算数相加  ALUCtrl：0001 |
| 2 | 减法 | 第一个和第二个运算数相减  ALUCtrl：0010 |
| 3 | 按位或 | 将第一个数和第二个数按位或  ALUCtrl：0100 |
| 4 | Lui | 将指令的低16位作为输入，放置到高位，作为输出数据  ALUCtrl：1000 |

1. EXT模块定义
   1. 基本描述

通过EXTOp来决定信号的扩展是在高位还是在低位执行。

* 1. 模块接口

|  |  |  |
| --- | --- | --- |
| c | 方向 | 描述 |
| 16bits | I | 指令的低十六位 |
| EXTOp | I | 来自控制单元的信号  1：低位加零  0：高位加零 |
| 16Bits | I | 指令的低十六位数据 |
| OutPut | O | 输出的信号扩展结果 |

* 1. 功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 信号位宽扩增 | 在高位加零（lui） |
| 2 | 信号位宽扩增 | 在低位加零 |

1. DM模块定义
   1. 基本描述

用作数据存储，以及从内存中取数的来源。

* 1. 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| A | I | 访问存储器的地址 |
| D | I | 写入的数据 |
| MemToReg | I | 从内存中取数到寄存器中的使能  1：能  0：否 |
| Str | I | 存入内存相应地址中  1：是  0：否 |
| Clk | I | 时钟信号 |
| clr | I | 存储器重置  1：是  2：否 |
| Result | I | ALU运算的结果 |
| WData | O | 写入寄存器中的数据 |

* 1. 功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 写入到存储器中 | 写使能为是的时候将写入的数据放入相应地址中 |
| 2 | 取数到寄存器中 | 将数据从相应地址中去除放到寄存器中 |

1. 控制器模块定义
2. 控制信号真值表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Func | 100001 | 100011 | n/a | | | | |
| Op | 000000 | 000000 | 001101 | 100011 | 101011 | 000100 | 001111 |
|  | add | sub | ori | lw | sw | beq | lui |
| RegDst | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| ALUSrc | 0 | 0 | 1 | 1 | 1 | 0 | 1 |
| MemtoReg | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| RegWrite | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| MemWrite | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| nPC\_Sel | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| ExtOp | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| ALUctr[3:0] | 0001 | 0010 | 0100 | 0001 | 0001 | 0010 | 1000 |

1. 功能控制信号的布尔表达式












1. 测试数据
   1. mips代码

ori $1, $1, 20#$1 give 20 to this register

ori $2, $2, 21#give 21 to this register

ori $3, $3, 1#give 1 to $3

addu $4, $2, $3#addu $2+$3 = $4

lui $4, 8#the lui， to $4

lui $5, 9# the lui, to $5

addu $6, $5, $4

subu $7, $4, $3

addu $8, $1, $2

subu $9, $8, $7

sw $6,4($1)

sw $7,3($2)

lw $10, ($1)

lw $11, 4($1)

sw $4, ($1)

lw $5, -1($3)

ori $13, $13, 1

addu $12, $11, $13

hehe: subu $12, $12, $3

beq $11, $12, hehe

* 1. 代码

34210014#$1 give 20 to this register

34420015#give 21 to this register

34630001#give 1 to $3

00432021#addu $2+$3 = $4

3c040008#the lui， to $4

3c050009# the lui, to $5

00a43021#$5 + $4 = $6

00833823#$7 = $4 - $3

00224021#$1+$2 = $8

01074823#$7+$8 = $9

ac260004#store content in $6 to $1 shift 4

ac470003#store content in $2 to $7 shift 3

8c2a0000#load $1 contents to $10

8c2b0004# load $1 contents to $11 shift is 4

ac240000#save content in $4 to $1

8c65ffff#shift is minus, is -1, store content in $5 to $3

35ad0001#ori, let the content of $13 to be 1

016d6021# $12 = $13 + $11

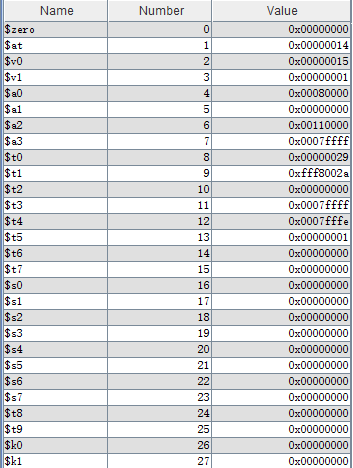
01836023# $12 subtract $3

116cfffe # if $11 and $12 is equal jump to previous instruction

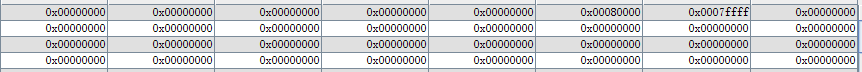
* 1. 预期结果

运行结束后应该得到如下的结果

* + 1. 寄存器部分



* + 1. 存储器部分



以上是程序运行的理想结果

四． 思考题

1. 



2 .

|  |  |  |  |
| --- | --- | --- | --- |
| 19 | 2输入与门 | 2输入或门 | 非门 |
| RegDst | 17 | 1 | 10 |
| RegWrite | 32 | 4 | 10 |
| ALUSrc | 20 | 3 | 5 |
| MemWrite | 5 | 0 | 5 |
| MemToReg | 5 | 0 | 3 |
| ExtOp | 15 | 2 | 3 |

1. 第一种的每个控制信号都需要对其分配单独的与或非门，比较浪费元件；第二种则先把opFunc变成相应的指令信号，再由指令信号生成控制信号，当一种指令对应的多种控制信号为1时，不必在为每个信号单独分配与门，公用指令的信号只要再添加或门就可以了。