P4实验报告

1. 模块定义
   1. PC
      1. 基本描述

PC主要功能是完成输出当前指令地址并保存下一条指令地址。

复位后，PC指向0x0000\_3000，此处为第一条指令的地址。

1. 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| NPC[31:2] | I | 下条指令的地址 |
| Clk | I | 时钟信号 |
| Reset | I | 复位信号。  1：复位  0：无效 |
| PC[31:2] | O | 30位指令存储器地址(最低2位省略) |

1. 实现功能

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 描述 |
| 1 | 复位 | 当复位信号有效时，PC被设置为0x0000\_3000. |
| 2 | 保存NPC并输出 | 在每个clock的上升沿保存NPC，并输出。 |
| 3 | 取指令 | 根据PC的值,从IM中得到指令的机器码 |

* 1. NPC
     1. 基本描述

NPC主要功能是根据PC和NPCOp得到下一条指令地址。

1. 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| PC[31:2] | I | 当前指令的地址 |
| Dout[31:0] | I | 当前指令 |
| Reset | I | 复位信号。  1：复位  0：无效 |
| NPC[31:2] | O | 下一条指令的地址(最低2位省略) |

1. 实现功能

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 描述 |
| 1 | 计算NPC并输出 | NPC主要功能是根据PC和NPCOp得到下一条指令地址。 |

* 1. IM
     1. 基本描述

根据addr取出相应指令。

1. 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| Addr[11:2] | I | 指令地址 |
| Dout[31:0] | O | 取出的指令 |

1. 实现功能

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 描述 |
| 1 | 取指令 | 根据addr取出相应指令 |

* 1. GPR
     + 1. 基本描述

GPR为通用寄存器,其中主要包括1024个用于存取数据的32位寄存

器

* + - 1. 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| Rs [4:0] | I | 第一个要读的寄存器 |
| Rt [4:0] | I | 第二个要读的寄存器 |
| Rd [4:0] | I | 要写入的寄存器 |
| WData[31:0] | I | 要写入的数据 |
| RegWrite | I | 写使能信号 |
| Clk | I | 时钟信号 |
| Reset | I | 复位信号 |
| RData1 [31:0] | O | 读出Rs存的数据 |
| RData2 [31:0] | O | 读出Rt存的数据 |

* + - 1. 实现功能

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 描述 |
| 1 | 取(读)数 | 根据Rs,Rt的值,读出其中所存的数据 |
| 2 | 存数 | 当写使能RegWrite为1时,将WData写入Rd寄存器中 |

* 1. ALU
     + 1. 基本描述

根据Op控制信号,进行相应的加减与或运算.

1. 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| ALU\_DA[31:0] | I | 第一个运算数 |
| ALU\_DB[31:0] | I | 第二个运算数 |
| ALUOp [2:0] | I | 运算方式控制数 |
| ALU\_DC[31:0] | O | 运算结果 |
| ALU\_Zero | O | 判断ALU\_DA, ALU\_DB相减是否为0 |

1. 实现功能

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 描述 |
| 1 | 实现加减与或(比较)运算 | 根据op的值选择相应运算,并把结果作为输出 |
| 2 | 判断ALU\_DA ALU\_DB是否相等 | 根据ALU\_DA-ALU\_DB的结果判断,zero为1相等,否则不等 |

* 1. EXT

1. 基本描述

EXT主要是实现各种立即数的扩展,以实现达到32位,从而能够

进入ALU进行运算或者实现相应功能.

1. 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| Din [15:0] | I | 要扩展的立即数 |
| EXTOp[1:0] | I | 扩展方式控制数 |
| Dout [31:0] | O | 扩展后的32位结果 |

1. 实现功能

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 描述 |
| 1 | 0扩展 | Op[1:0]=00时,0扩展,用于ori |
| 2 | 符号扩展 | Op[1:0]=01时,符号扩展,用于lw,sw, |
| 3 | 左移16位 | Op[1:0]=10时,用<<左移16位,用于lui |

* 1. DM
     1. 基本描述

实现数据的存储取出。

1. 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| Addr[11:2] | I | 要存入的地址 |
| Din[31:0] | I | 存入的数据 |
| we | I | 写使能信号 |
| clk | I | 时钟信号 |
| Dout[31:0] | O | 取出的数据 |

1. 实现功能

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 描述 |
| 1 | 存数 | 当存使能为1时,将Din存入addr指向的寄存器中 |
| 2 | 取数 | 将addr中的Data取出 |

* 1. Ctrl
     1. 基本描述

根据op码和Function码,产生各控制信号.

1. 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| Op[5:0] | I | 指令机器码的31:26位 |
| Function[5:0] | I | 指令机器码的5:0位 |
| NPCOp[1:0] | O | NPC控制信号 |
| ALUOp[2:0] | O | ALU控制信号 |
| RegWrite | O | 目的寄存器写使能 |
| EXTOp[1:0] | O | EXT控制信号 |
| MemWrite | O | DM写使能信号 |
| RegDst[1:0] | O | Mux\_Rd控制信号 |
| ALUSrc | O | Mux\_ALUSrc控制信号 |
| WDSrc[1:0] | O | Mux\_WDSrc控制信号 |

1. 实现功能

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 描述 |
| 1 | 产生控制信号 | 根据OP码和Function码,产生各部件的控制信号 |

* 1. mux\_ALUSrc
     1. 基本描述

根据ctrl产生的ALUSrc控制信号,产生ALU第二个操作数来源.

1. 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| Data1[31:0] | I | 第一个来源（寄存器） |
| Data2 | I | 第二个来源（EXT） |
| ALUSrc | I | Ctrl产生的选择信号 |
| Data [31:0] | O | 选择产生的ALU第二个操作数 |

1. 实现功能

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 描述 |
| 1 | 产生ALU第二个源操作数 | 根据ctrl产生的ALUSrc,产生ALU第二个输入数据来源. |

* 1. mux\_Rd
     1. 基本描述

根据ctrl产生的RegDst控制信号,确定写入寄存器的地址.

1. 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| RD1 [4:0] | I | 第一个来源（RT） |
| RD2 [4:0] | I | 第二个来源（RD） |
| RegDst [1:0] | I | Ctrl产生的选择信号 |
| RD [4:0] | O | 选择产生的写入寄存器地址 |

1. 实现功能

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 描述 |
| 1 | 得到写入寄存器地址 | 根据ctrl产生的RegDst,选择产生最终要写入寄存器的地址 |

* 1. mux\_ WDSrc
     1. 基本描述

根据ctrl产生的WDSrc控制信号,确定写入寄存器的值.

1. 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| Data1 [31:0] | I | 第一个来源（ALU\_C） |
| Data2 [31:0] | I | 第二个来源（Data\_out） |
| Data3 [31:0] | I | 第三个来源(PCLink) |
| WDSrc [1:0] | I | Ctrl产生的选择信号 |
| Data [31:0] | O | 写入寄存器的内容 |

1. 实现功能

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 描述 |
| 1 | 得到写入寄存器的内容 | 根据ctrl产生的WDSrc,选择产生最终要写入寄存器的内容 |

1. 测试程序原理
   1. 测试程序代码

ori $t1,$zero,1 # t1 = 1

ori $t2,$zero,4 # t2 = 4

ori $t0,$zero,14 # t0 =5 times

ori $a0,$zero,1 # n

ori $a1,$zero,0 #address

while:

beq $a0,$t0,endwhile #n<=5

jal f #阶乘函数

sw $v0,0($a1) #结果保存到内存dm中

addu $a1,$a1,$t2 #地址加四

addu $a0,$a0,$t1 #n+1

beq $zero,$zero,while #继续循环

multi: #乘法函数

addu $s0,$zero,$a0 #将第一个乘数保存到$s0中

addu $s1,$zero,$v0 #将第二个乘数保存到$s1中

while1: #当第一个乘数大于一时,结果加第二个乘数

beq $s0,$t1,endwhile1

addu $v0,$v0,$s1

subu $s0,$s0,$t1

beq $zero,$zero,while1

endwhile1:

jr $ra

f: #阶乘递归函数

subu $sp,$sp,8 #$ra,$a0进栈

sw $ra,0($sp)

sw $a0,4($sp)

beq $a0,$t1,return #当n = 1,返回1

subu $a0,$a0,$t1 #否则,计算f(n-1)

jal f

lw $a0,4($sp) #$a0退栈,

jal multi #计算n\*f(n-1)

lw $ra,0($sp) #$ra退栈

addu $sp,$sp,8

jr $ra #返回

return: addu $v0,$zero,$t1 #结果赋1

lw $ra,0($sp) #$ra,$a0退栈

lw $a0,4($sp)

addu $sp,$sp,8

jr $ra #返回

endwhile:

* 1. 导出的机器码

34090001

340a0004

3408000b

34040001

34050000

10880022

0c000c12

aca20000

00aa2821

00892021

1000fffa

00048021

00028821

12090003

00511021

02098023

1000fffc

03e00008

3c010000

34210008

03a1e823

afbf0000

afa40004

10890009

00892023

0c000c12

8fa40004

0c000c0b

8fbf0000

3c010000

34210008

03a1e821

03e00008

00091021

8fbf0000

8fa40004

3c010000

34210008

03a1e821

03e00008

* 1. 测试原理

实现递归函数和乘法函数,

依次循环计算1~10的阶乘,保存到DM前十个位置

* 1. 测试期望

Dm的前十个寄存器保存1~10的阶乘

1. 问题
   1. 请说明为什么在忽略溢出的前提下，addi与addiu是等价的，add与addu是等价的。提示：阅读《MIPS32® Architecture For Programmers Volume II: The MIPS32® Instruction Set》中相关指令的Operation部分

答:根据指令描述分析,只有在两个加数都是负数的情况下才会产生溢出,溢出的一位1相当于符号位(代表负数),其他情况下两者的计算结果完全相同且正确.所以在不考虑溢出的情况下,溢出位即符号位被忽略,add,addi与addu,addiu的结果是一致的(虽然此时结果与理论结果不一致).