**计算机组成原理**

**课内大作业报告**

**学 号\_\_\_\_ \_\_20074214\_\_\_\_\_\_\_\_\_\_\_\_**

**姓 名\_ \_周子渔\_\_\_\_\_\_\_\_\_\_\_\_\_**

**指导教师\_ \_\_\_魏坚华\_\_\_\_\_\_\_\_\_\_\_\_**

**提交日期\_ 2022.5.4\_\_\_\_\_\_ \_**

**成绩评价表**

|  |  |  |
| --- | --- | --- |
| **报告内容** | **报告结构** | **报告最终成绩** |
| **□丰富正确**  **□基本正确**  **□有一些问题**  **□问题很大** | **□完全符合要求**  **□基本符合要求**  **□有比较多的缺陷**  **□完全不符合要求** |  |
| **报告与Project功能一致性** | **报告图表** | **总体评价** |
| **□完全一致**  **□基本一致**  **□基本不一致** | **□符合规范**  **□基本符合规范**  **□有一些错误**  **□完全不正确** |  |

**教师签字:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

目录

**一、顶层模块3**

**二、模块描述4**

2.1 ALU模块4

2.1.1 add32模块5

2.1.2 add16模块6

2.1.3 add4模块8

2.1.4 sub32模块10

2.1.5 or32模块11

2.2 Extender模块12

2.3 DM模块13

2.4 GPR模块14

2.5 Controller模块16

2.6 IFU模块18

**三、测试文件与机器指令描述20**

3.1测试文件及MARS运行结果20

3.2机器指令描述20

3.3测试指令描述21

**四、寄存器结果截图及文字解释23**

4.1指令123

4.2指令223

4.3指令323

4.4指令424

4.5指令524

4.6指令624

4.7指令725

4.8指令825

4.9指令925

4.10指令1026

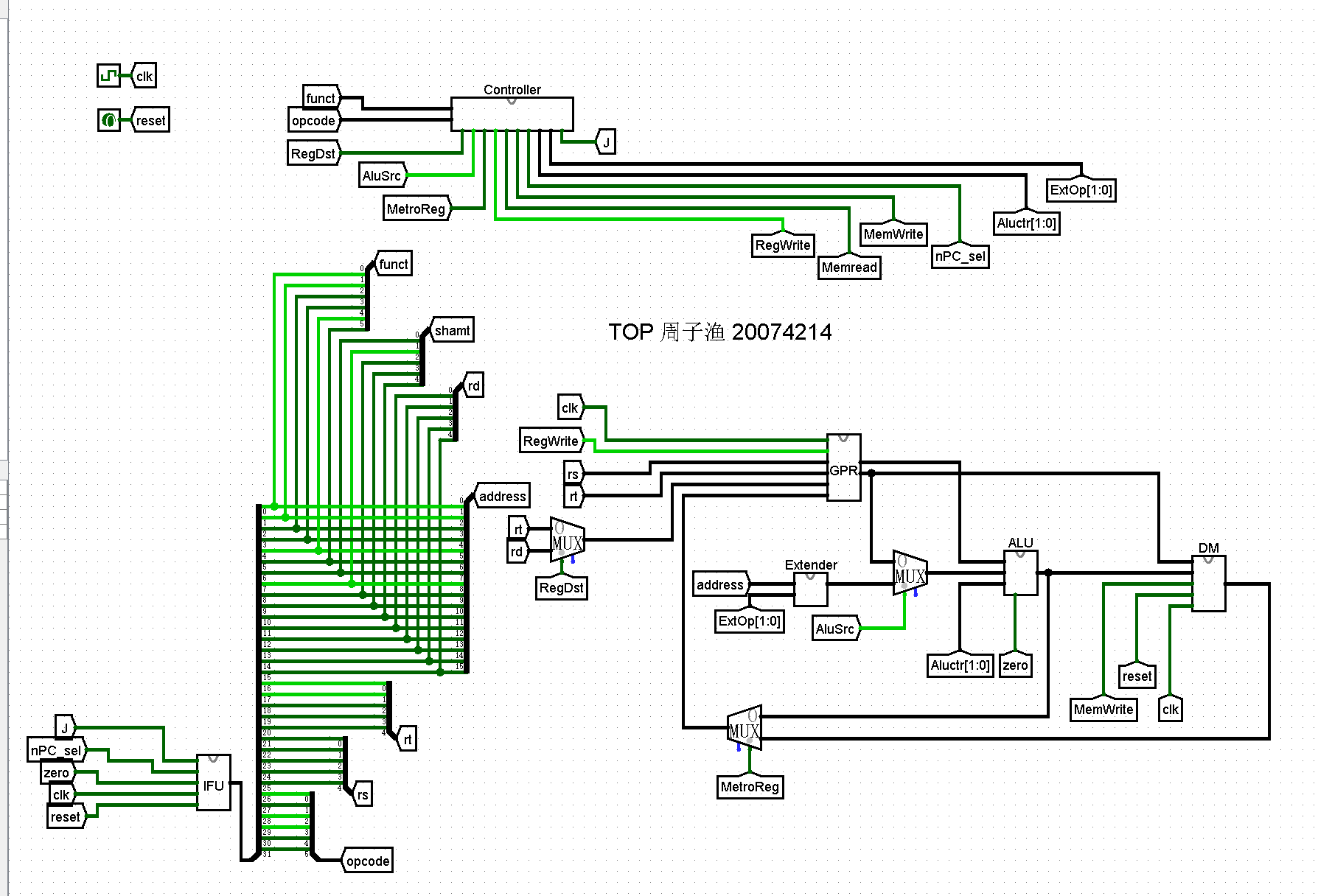
4.11指令1126

4.12指令1226

4.13指令1327

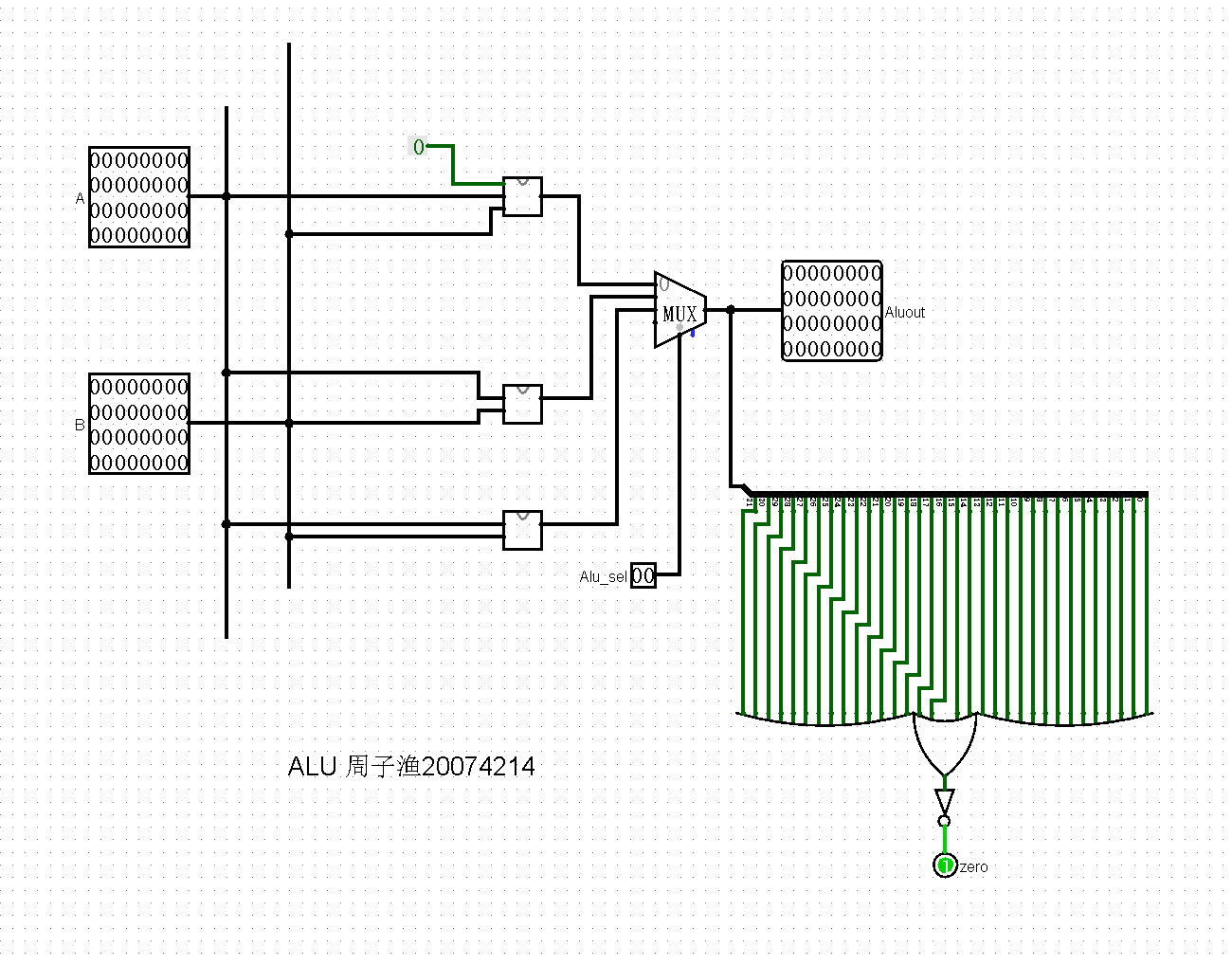
**五、收获体会27**

**一、顶层模块**

****

**二、模块描述**

**2.1 ALU模块**

****

1. 基本描述

ALU模块的主要功能是根据选择信号Alu\_ sel [1:0]来进行相应的加法、减法、或运算，分别产生对应的结果。同时设计零信号输出为zero，当输出结果为0的时候，该信号输出1。此模块由32位全加、全减和or模块构成，可以对于两个指定的32位数字进行指定的运算，输出结果为Alu\_out。

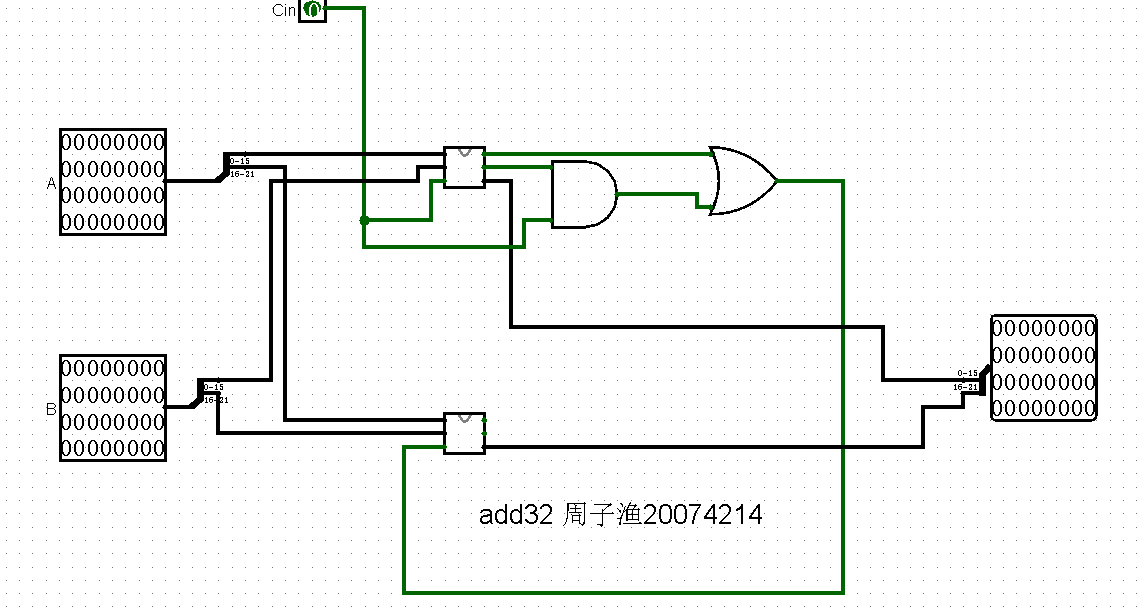
1. 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| Alu\_sel[1:0] | I | ALU做何种运算的选择信号。  00：无符号加法运算。  01：无符号减法运算。  10：或运算。 |
| A | I | 32位运算数A。 |
| B | I | 32位运算数B。 |
| Zero | O | 计算结果为0的标志信号。  1：结果为0。  0：结果非0。 |
| Alu\_out | O | ALU的计算结果。 |

1. 功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 计算 | 根据Alu\_sel[1:0]值的不同对运算数A、B进行计算。  00：Alu\_out=A+B。  01：Alu\_out=A-B。  10：Alu\_out=A or B。 |
| 2 | 判断结果是否为0 | 根据Alu\_out的值判断是否为0。 |

2.1.1 add32模块



1. 基本描述

add32模块是一个组内组间并行的32位全加器。是由两个16位的并行加法器并行构成，可以实现对输入的两个32位二进制数A和B进行加法运算。将A、B的低位放放入16位并行加法器，产生超前进位和16本地和S0-S15作为结果的低15位，再用16位加法器计算出结果的高位。将两个加法器输出的结果拼接，得到A和B相加的结果S。

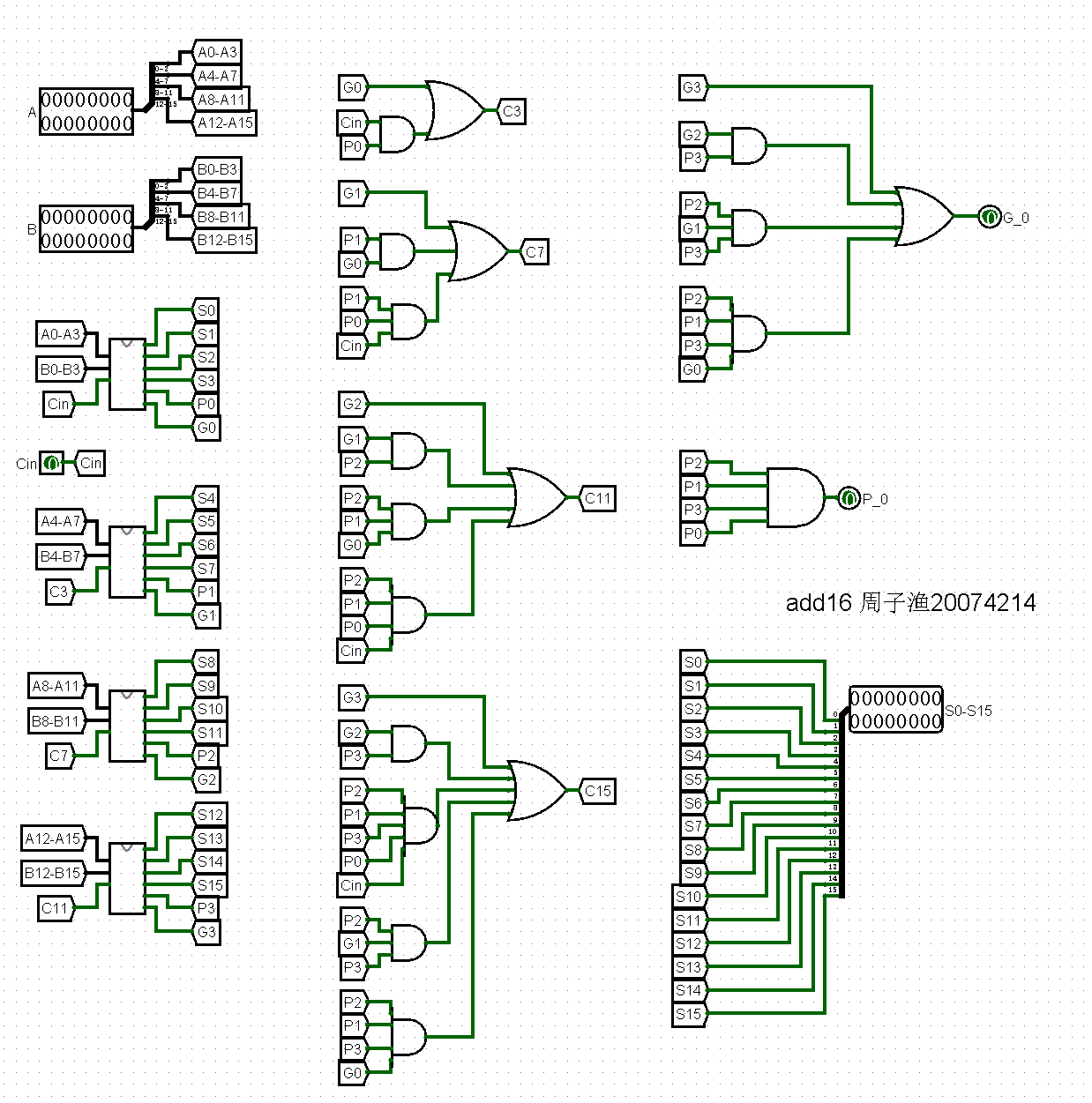
1. 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| Cin | I | 低位进位。 |
| A | I | 32位二进制数A。 |
| B | I | 32位二进制数B。 |
| S | O | 32位计算结果A+B。 |

1. 功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 计算 | 采用拼接两个16位并行加法器的方法计算出两个32位二进制数A+B的结果，输出为S。 |

2.1.2 add16模块



1. 基本描述

add16模块是一个组内组间并行的16位全加器。是由四个4位的并行加法器并行构成，可以实现对输入的两个16位二进制数A和B进行加法运算。将A、B的15-12、11-8、7-4、3-0位分别放入四个4位并行加法器中，产生超前进位C3、C7、C11，从而产生本地和S15-S12、S11-S8、S7-S4、S3-S0。并在最后根据四个4位并行加法器的G、P计算超前进位G\_0、P\_0。

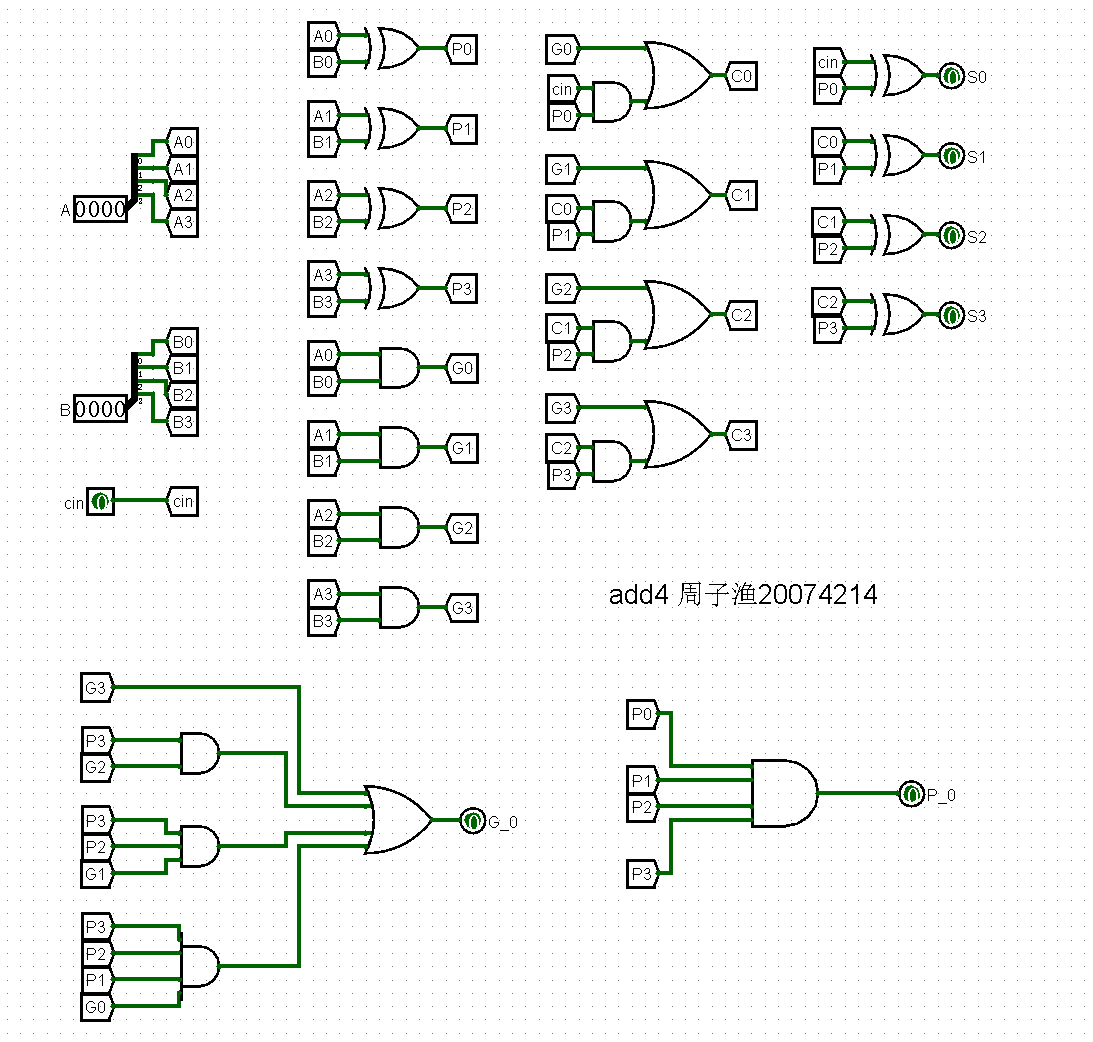
1. 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| Cin | I | 低位进位。 |
| A | I | 16位二进制数A。 |
| B | I | 16位二进制数B。 |
| S | O | 16位计算结果A+B。 |
| G\_0 | O | 用于计算32位加法器的超前进位。 |
| P\_0 | O | 用于计算32位加法器的超前进位。 |

1. 功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 计算 | 计算两个16位二进制数A+B的结果，输出为S。 |
| 2 | 产生计算超前进位的G\_0、P\_0。 | 对输入的两个16位二进制数A和B进行加法运算。将A、B的15-12、11-8、7-4、3-0位分别放入四个4位并行加法器中，产生超前进位C3、C7、C11，从而产生本地和S15-S12、S11-S8、S7-S4、S3-S0。并在最后根据四个4位并行加法器的G、P计算超前进位G\_0、P\_0。 |

2.1.3 add4模块



1. 基本描述

add4是四位二进制数并行加法器，以两个四位二进制数A、B作为输入，通过每位产生超前进位本地和进行计算，最后输出A+B的结果以及用于计算超前进位的G\_0、P\_0。

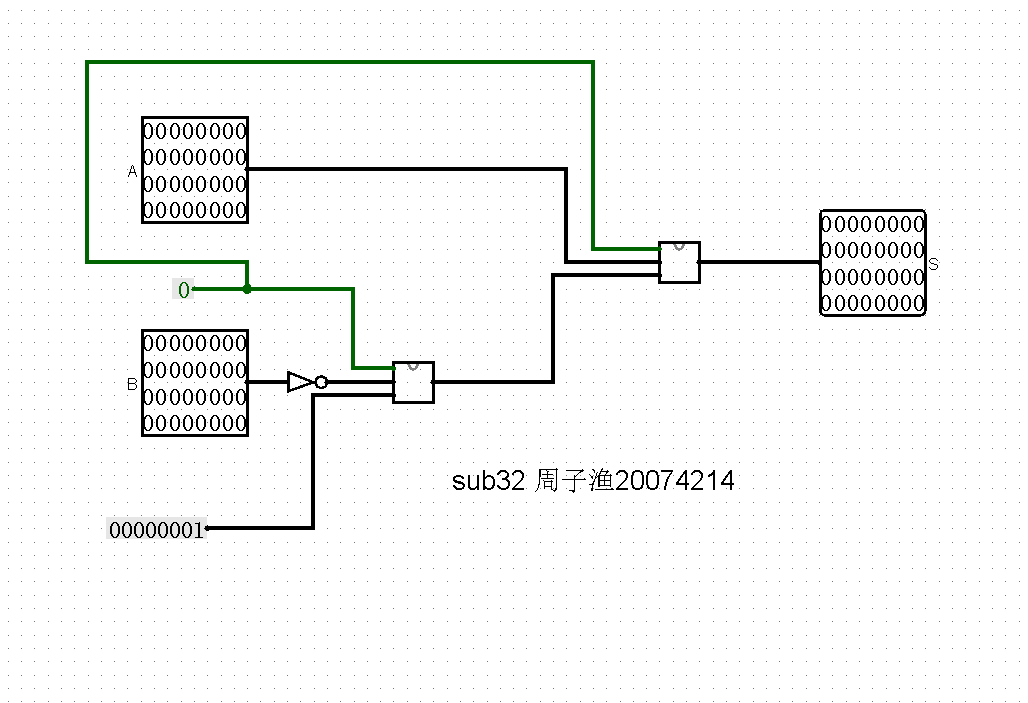
(2) 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| Cin | I | 低位进位。 |
| A | I | 4位二进制数A。 |
| B | I | 4位二进制数B。 |
| S | O | 4位计算结果A+B。 |
| G\_0 | O | 用于计算16位加法器的超前进位。 |
| P\_0 | O | 用于计算16为加法器的超前进位。 |

(3) 功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 计算 | 采用拼接两个16位并行加法器的方法计算出两个32位二进制数A+B的结果，输出为S。 |
| 2 | 产生计算超前进位的G\_0、P\_0 | 以两个四位二进制数A、B作为输入，通过每位产生超前进位本地和进行计算，最后输出A+B的结果以及用于计算超前进位的G\_0、P\_0。 |

2.1.4 sub32模块



1. 基本描述

sub32是实现两个32位二进制数减法运算的模块，输出结果为S。基于二进制数减法计算规则：两数相减等于被减数加减数的补码。实现两数相减的功能。

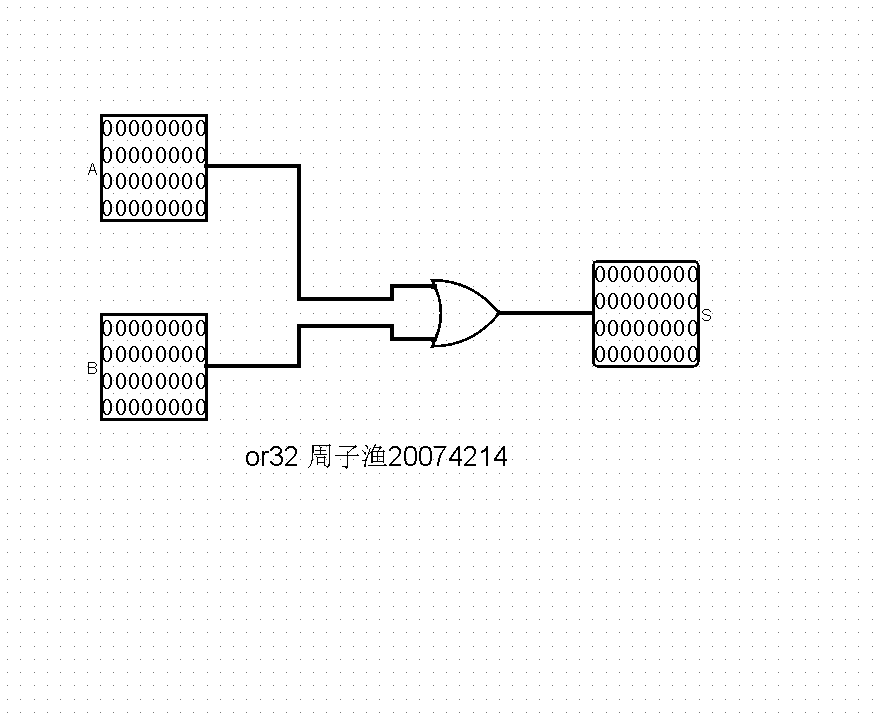
1. 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| A | I | 32位二进制数A。 |
| B | I | 32位二进制数B。 |
| S | O | 32位计算结果A-B。 |

1. 功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 计算 | 计算两个32位二进制数A-B的结果，输出为S。 |

2.1.5 or32模块



1. 基本描述

or32是实现两个32位二进制数按位或的结果。是由一个或门进行实现的。

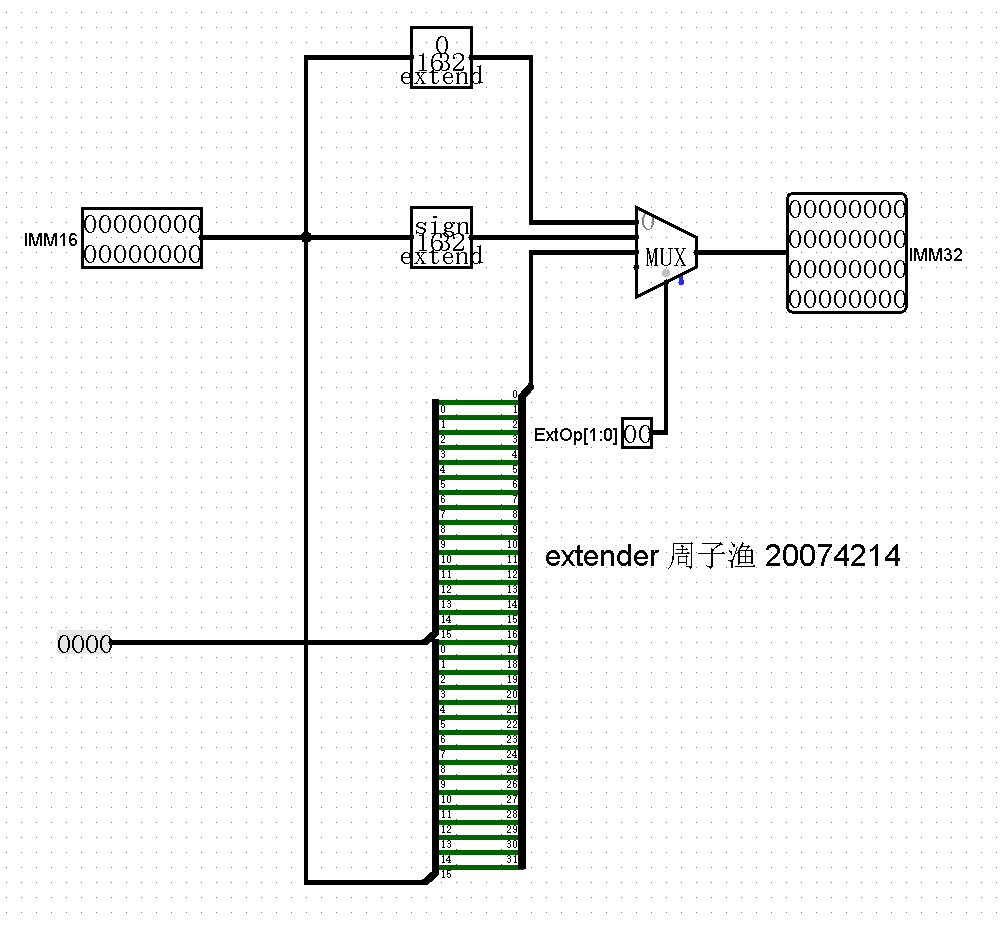
1. 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| A | I | 32位二进制数A。 |
| B | I | 32位二进制数B。 |
| S | O | 32位计算结果A or B。 |

1. 功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 计算 | 计算两个32位二进制数A or B的结果，输出为S。 |

**2.2 extender模块**



1. 基本描述

extender32是一个多功能数字扩展器，可以根据选择信号ExpOp[1:0]信号选择扩展的类型，包括0扩展、符号位扩展和高位扩展。由扩展器和选择器构成。

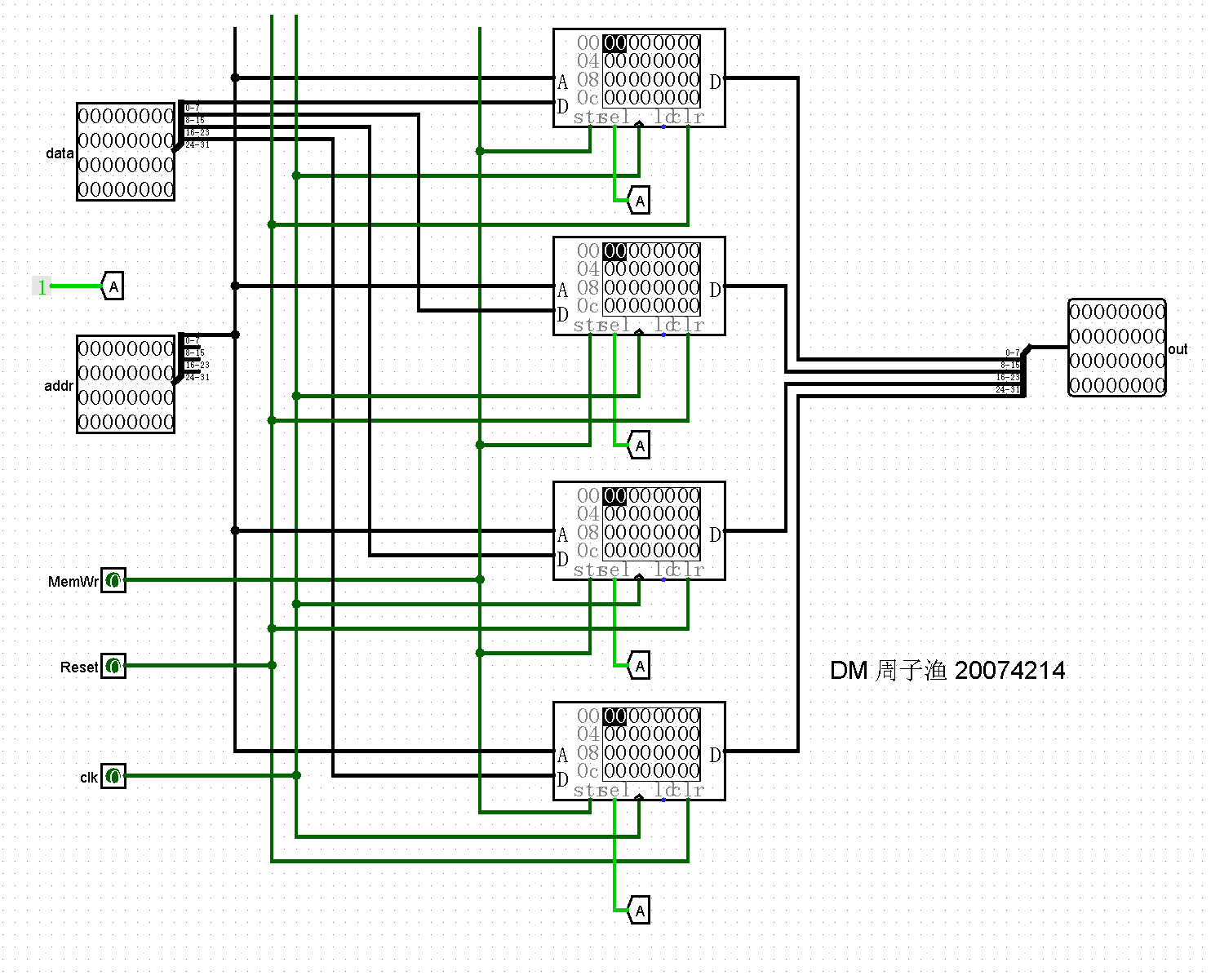
1. 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| IMM16 | I | 待扩展的16位二进制数A。 |
| Ext\_Op[1:0] | I | 控制扩展类型的信号。  00：立即数零扩展。  01：立即数符号位扩展。  10：立即数向高位扩展。 |
| IMM32 | O | 扩展后的32位二进制数。 |

1. 功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 立即数扩展 | 根据Exp\_Op[1:0]的取值选择扩展的类型。  00：立即数零扩展。  01：立即数符号位扩展。  10：立即数向高位扩展。 |

**2.3 DM模块**



1. 基本描述

DM是数据储存器件，同时能读取写入储存器的数据，设置有清零信号reset。当写使能信号MemWr=1，且时钟信号为上升沿的时候，可以将数据data写入addr对应的地址。使用4个存储体并行的小端序存储方式，每个可存256B，总共可存1KB。读取地址为addr对应的数据并将其送到out输出。

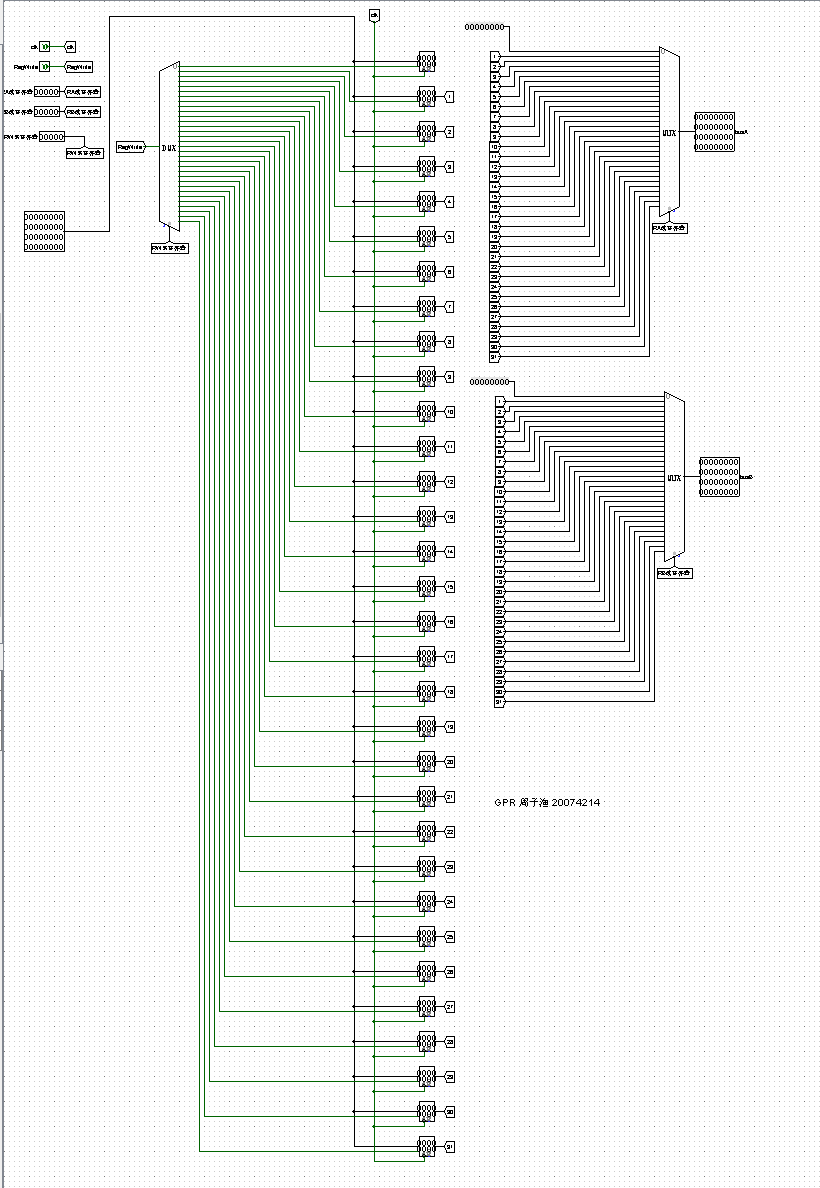
1. 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| data[31:0] | I | 写入存储器的数据。 |
| addr[9:2] | I | 要进行读、写的数据所在储存器中的地址 |
| MemWr | I | 写入数据的使能信号。 |
| clk | I | 时钟信号。 |
| reset | I | 复位信号。 |
| out | O | 从存储器中读取要输出的数据。 |

1. 功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 复位 | 复位信号reset高有效时，DM数据清零。 |
| 2 | 存储数据 | 当MemWr=1时，代表着写入功能有效，在下一个clk上升沿时将数据data存入此时值为addr的地址处。 |
| 3 | 读取数据 | 在下一个clk上升沿到来时，将存储区中addr地址所储存的数据传入out输出。 |

**2.4 GPR模块**



1. 基本描述

GPR实现的是通用寄存器组。由32个32位的寄存器堆、解复用器和多路选择器构成，通过tunnel连接输入输出端并构造寄存器组。能够实现读写的寄存器功能。模块暂时存储相应的数据（0号寄存器恒存0且无法改变），当写使能RegWrite为1时，表示此时需要向寄存器堆写入数据，在下一个clk上升沿到来时将数据WD写入RW写寄存器信号对应编号的寄存器。在每一个clk上升沿到来时将RA读寄存器和RB读寄存器对应编号的数据分别送到busA和busB中输出。

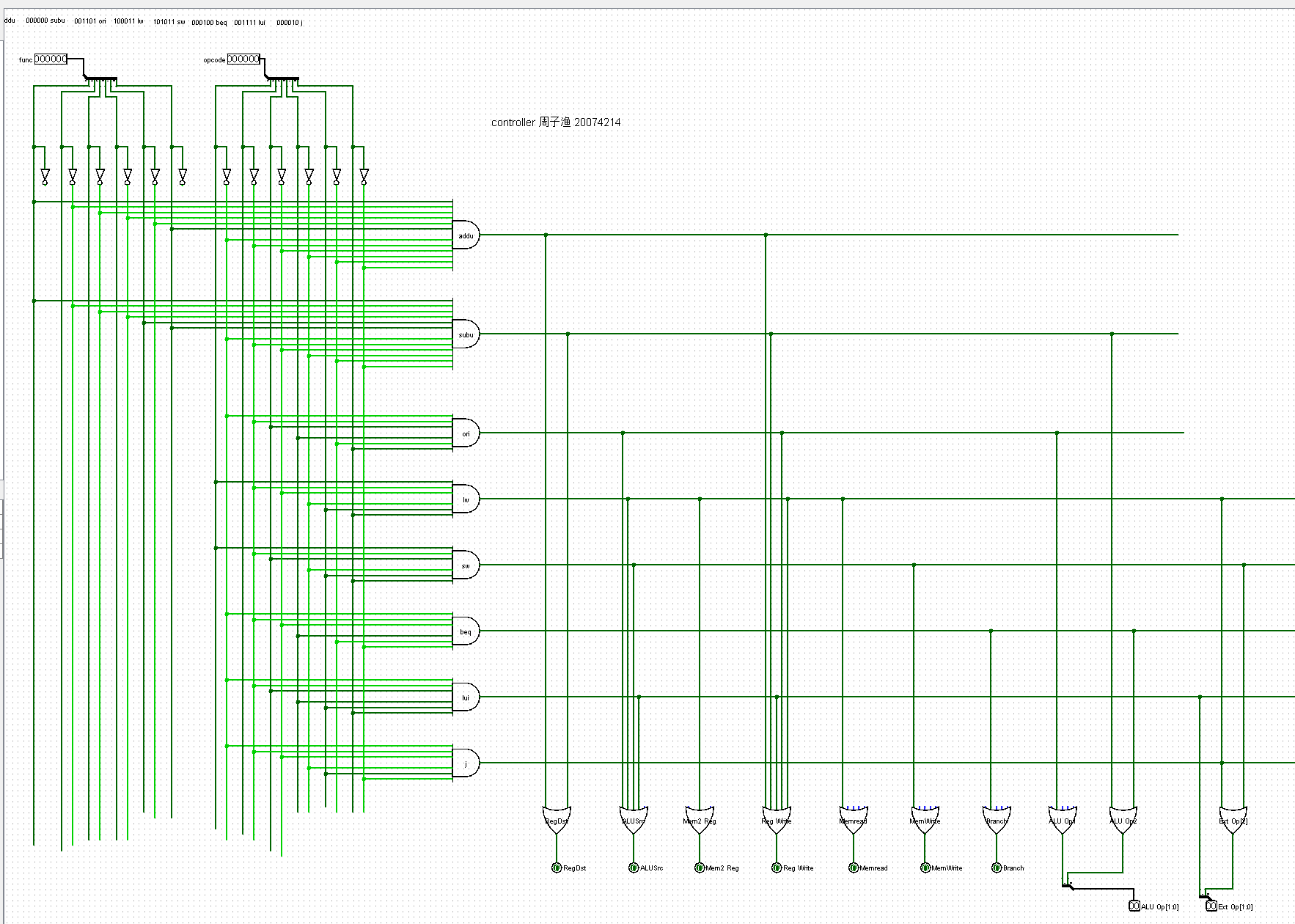
1. 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| WD | I | 要写入寄存器的32位二进制数。 |
| RegWrite | I | 写入数据的使能信号。 |
| clk | I | 时钟信号。 |
| RA读寄存器 | I | busA读取数据的寄存器的5位编号。 |
| RB读寄存器 | I | busB读取数据的寄存器的5位编号。 |
| RW写寄存器 | I | 要将WD数据写入的寄存器的5位编号。 |
| busA | O | 输出的第一个32位数据 |
| busB | O | 输出的第二个32位数据 |

1. 功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 存储数据 | 当写使能信号RegWrite有效时，代表着写入功能有效，在下一个clk上升沿到来时将数据WD写入RW写寄存器信号对应编号的寄存器。 |
| 2 | 读取数据 | 在每一个clk上升沿到来时将RA读寄存器和RB读寄存器对应编号的数据分别送到busA和busB中输出。 |

**2.5 Controller模块**



1. 基本描述

Controller模块是控制器，可以接受机器码的最高六位和最低六位。利用安排好的逻辑门输出对应的指令信号，根据指令信号与输出信号的逻辑关系给相应的输出信号赋值。主要的功能为根据opcode和funct的值解读该条机器码为哪条指令，并输出对应的信号加以控制。

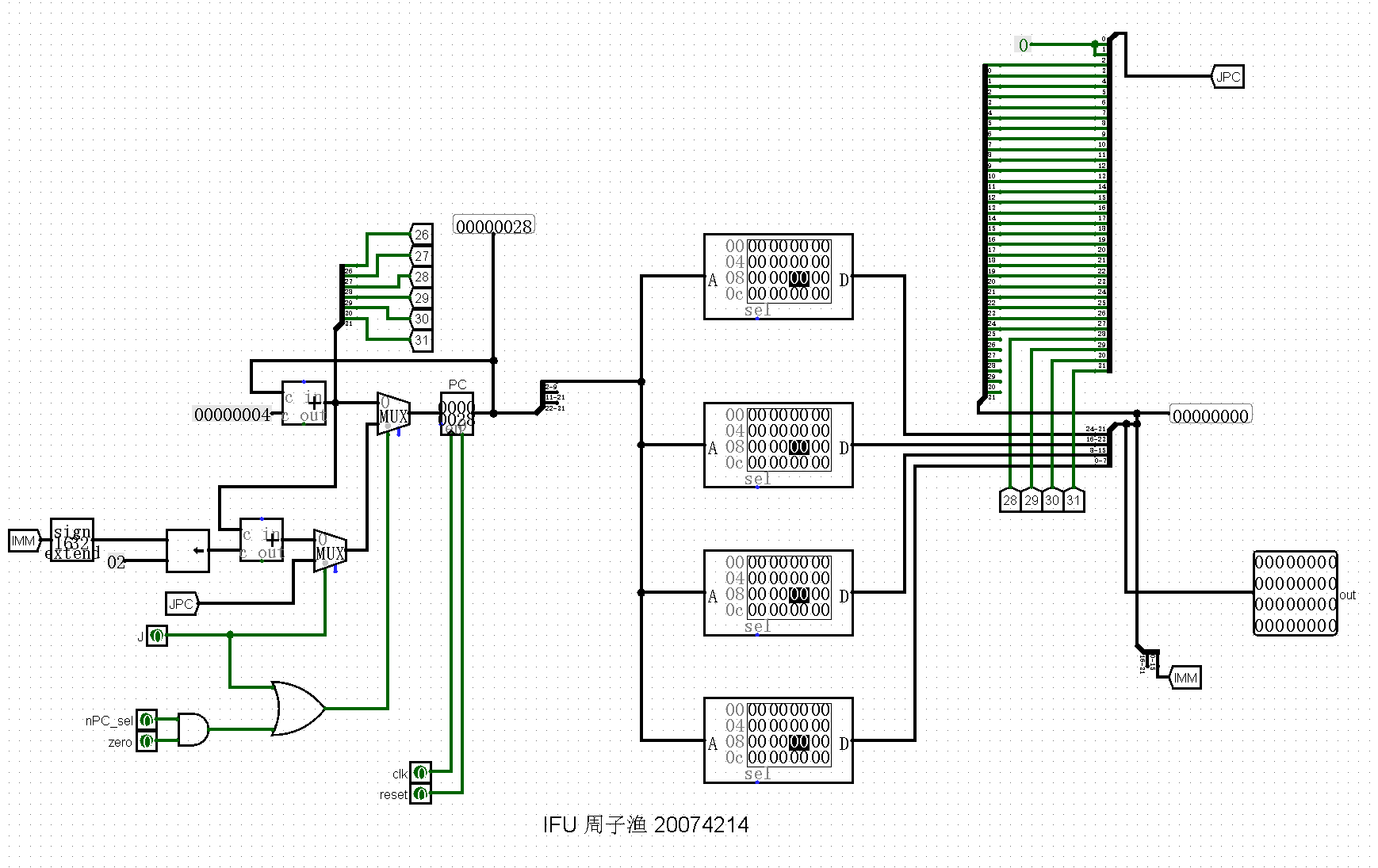
1. 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| funct[5:0] | I | 32位操作数的低6位。 |
| opcode[5:0] | I | 32位操作数的高6位。 |
| lui | O | 标志是否为lui信号。 |
| RegDst | O | 选择rt或者rd对应的寄存器。 |
| AluSrc | O | 寄存器取值或者扩展器取值。 |
| Mem2Reg | O | ALU模块到寄存器或者数据储存器到寄存器。  1：前者。  0：后者。 |
| RegWrite | O | GPR写使能信号。 |
| MemWrite | O | DM写使能信号。 |
| nPC\_sel | O | npc选择信号。 |
| j | O | J跳转信号。 |
| ExtOp[1:0] | O | 选择扩展类型信号。  00：0扩展。  01：符号位扩展。  10：lui扩展。 |
| ALUctr[1:0] | O | 选择对应的操作类型。  00：加法操作。  01：减法操作。  10：或运算。 |

1. 功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 指令译码 | 根据funct和opcode进行译码，将机器码转化为对应指令。 |
| 2 | 信号编码 | 根据译码结果给对应的输出信号赋值。 |

**2.6 IFU模块**



1. 基本描述

IFU模块可以实现取值功能。可以实现顺序取值（pc+4），根据zero、nPC\_sel信号实现beq跳转，以及根据J信号及输入的地址进行J跳转。包括32位寄存器、4个8位地址位宽、8个数值位宽的只读存储器（1kb）和加法器、选择器等组成。

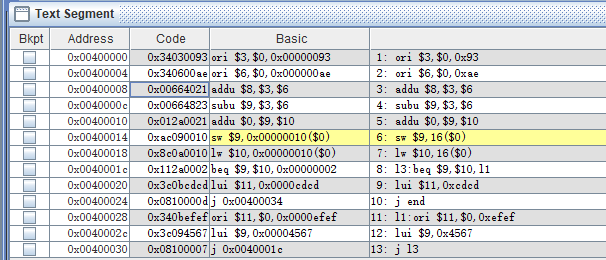
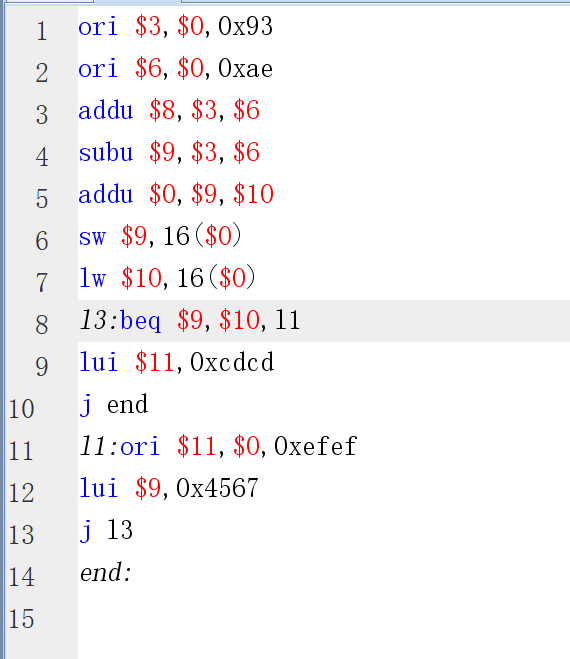
1. 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| nPC\_sel | I | 表示当前指令是否为beq。  1：当前指令为beq。  0：当前指令非beq。 |
| zero | I | ALU模块计算结果为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。 |
| out | O | 取出的32位指令。 |

1. 功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 复位 | 复位信号reset有效的时候，PC被设置成0×00000000 |
| 2 | 取指令 | 根据PC从IM中取出指令。 |
| 3 | 计算下一条指令 | 如果当前指令不是beq则PC+4。  如果当前指令是beq，并且zero为0或nPC\_sel为0，且j为0时，PC+4。  如果当前指令是beq，并且zero、nPC\_sel为1，且j为0时，则PC+4+（sign\_ext(out[15:0])<<2）。  如果当前指令是j指令，且j为1时，PC={（PC+4）[31:28],zero\_ext(out[25:0],00)} |

**三、测试文件与机器指令描述**

**3.1 测试文件及MARS运行结果**

根据老师提供的测试代码，导入MARS中获得机器码如上图所示。

**3.2 机器指令描述**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 助记符 | 指令格式 | | | | | | 功能解释 |
| R-type | Op | rs | rt | rd | shamt | funct | / |
| addu | 000000 | rs | rt | rd | 00000 | 100001 | Rd<-rs+rt |
| subu | 000000 | rs | rt | rd | 00000 | 100011 | Rd<-rs-rt |
| I-type | op | rs | rt | immediate | | | / |
| ori | 001101 | rs | rt | immediate | | | Rt<-rs |（immediate（0扩展）） |
| lui | 001111 | rs | rt | immediate | | | Rt<-immediate\*65536（移至高16位） |
| lw | 100011 | rs | rt | immediate | | | Rt<-memory[rs+immediate（符号位扩展）] |
| sw | 101011 | rs | rt | immediate | | | Rt->Memory[rs+immediate（符号位扩展）] |
| beq | 000100 | rs | rt | immediate | | | PC<-PC+4+4\*immediate |
| J-Type | op | address | | | | | / |
| J | 000010 | address | | | | | PC<-{(PC+4)[31:28],address,00} |

3.3 测试指令描述

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 指令序号 | Mips指令 | 执行结果 | 含义 | 16进制机器码 |
| 1 | ori $3,$0,0x93 | 3号寄存器中的数变为0x93 | 将0号寄存器中的数据与0x93相按位或并将结果存入3号寄存器 | 34030093 |
| 2 | ori $6,$0,0xae | 6号寄存器中的数变为0xae | 将0号寄存器中的数据与0xae相按位或并将结果存入6号寄存器 | 340600ae |
| 3 | addu $8,$3,$6 | 8号寄存器中的结果变为0x141 | 将3号、6号寄存器中的数相加并将结果放入8号寄存器 | 00664021 |
| 4 | subu $9,$3,$6 | 9号寄存器中的数据变为ffffffe5 | 将3号寄存器的数据减去6号寄存器的数据的结果存入9好寄存器 | 00664823 |
| 5 | addu $0,$9,$10 | 无效指令 | 将9号、10号寄存器的数字相加放入0号寄存器 | 012a0021 |
| 6 | sw $9,16($0) | 存储器中以0号寄存器位基址，偏移量为16的地址中的数据以小端序存储0xffffffe5 | 将9号寄存器的数据写入以0号寄存器位基址，偏移量为16的地址 | ac090010 |
| 7 | lw $10,16($0) | 10号寄存器的内容变为0xffffffe5 | 将存储器中0号寄存器位基址偏移量为16对应地址的内容写入10号寄存器 | 8c0a0010 |
| 8 | l3:beq $9,$10,l1 | 1. 跳转至L1指 2. 进行下一条指令 | 比较9、10号寄存器中的数据是否相等，是则跳转至L1指令 | 112a0002 |
| 9 | lui $11,0xcdcd | 11号寄存器的数据为0xcdcd0000 | 将立即数0xcdcd进行高位拓展并存入11号寄存器中 | 3c0bcdcd |
| 10 | j end | 结束程序 | 无条件跳转至end指令处 | 08000c0d |
| 11 | l1:ori $11,$0,0xefef | 11号寄存器的内容变为0xefefef | 将0号寄存器中的内容与0xefef进行按位或运算，将结果存储到11号寄存器 | 340befef |
| 12 | lui $9,0x4567 | 9号寄存器的数据为0x45670000 | 将立即数0x4567进行高位拓展并存入11号寄存器中 | 3c094567 |
| 13 | J l3 | 无条件跳转至l3指令 | 无条件跳转至l3指令 | 08000c07 |

**四、寄存器结果截图及文字解释**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 指令序号 | Mips指令 | 执行结果 | 含义 | 16进制机器码 | 结果截图： |
| 1 | ori $3,$0,0x93 | 3号寄存器中的数变为0x93 | 将0号寄存器中的数据与0x93相按位或并将结果存入3号寄存器。 | 34030093 |  |

**4.1 指令1**

**4.2指令2**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 指令序号 | Mips指令 | 执行结果 | 含义 | 16进制机器码 | 结果截图： |
| 2 | ori $6,$0,0xae | 6号寄存器中的数变为0xae | 将0号寄存器中的数据与0xae相按位或并将结果存入6号寄存器。 | 340600ae |  |

**4.3 指令3**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 指令序号 | Mips指令 | 执行结果 | 含义 | 16进制机器码 | 结果截图： |
| 3 | addu $8,$3,$6 | 8号寄存器中的结果变为0x141 | 将3号、6号寄存器中的数相加并将结果放入8号寄存器。 | 00664021 |  |

**4.4 指令4**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 指令序号 | Mips指令 | 执行结果 | 含义 | 16进制机器码 | 结果截图： |
| 4 | subu $9,$3,$6 | 9号寄存器中的数据变为ffffffe5 | 将3号寄存器的数据减去6号寄存器的数据的结果存入9号寄存器。 | 00664823 |  |

**4.5 指令5**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 指令序号 | Mips指令 | 执行结果 | 含义 | 16进制机器码 | 结果截图： |
| 5 | addu $0,$9,$10 | 0号寄存器不发生变化 | 将9号、10号寄存器的数字相加放入0号寄存器. | 012a0021 |  |

**4.6 指令6**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 指令序号 | Mips指令 | 执行结果 | 含义 | 16进制机器码 | 结果截图： |
| 6 | sw $9,16($0) | 存储器中以0号寄存器位基址，偏移量为16的地址中的数据以小端序存储0xffffffe5 | 将9号寄存器的数据写入以0号寄存器位基址，偏移量为16的地址。 | ac090010 |  |

**4.7 指令7**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 指令序号 | Mips指令 | 执行结果 | 含义 | 16进制机器码 | 结果截图： |
| 7 | lw $10,16($0) | 10号寄存器的内容变为0xffffffe5 | 将存储器中0号寄存器位基址偏移量为16对应地址的内容写入10号寄存器。 | 8c0a0010 |  |

**4.8 指令8**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 指令序号 | Mips指令 | 执行结果 | 含义 | 16进制机器码 | 结果截图： |
| 8 | l3:beq $9,$10,l1 | 跳转至L1指令 | 比较9、10号寄存器中的数据是否相等，是则跳转至L1指令. | 112a0002 |  |

**4.9 指令9**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 指令序号 | Mips指令 | 执行结果 | 含义 | 16进制机器码 | 结果截图： |
| 9 | lui $11,0xcdcd | 11号寄存器的数据为0xcdcd0000 | 将立即数0xcdcd进行高位拓展并存入11号寄存器中 | 3c0bcdcd |  |

**4.10 指令10**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 指令序号 | Mips指令 | 执行结果 | 含义 | 16进制机器码 | 结果截图： |
| 10 | j end | 结束程序 | 无条件跳转至end指令处 | 08000c0d | 无 |

**4.11 指令11**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 指令序号 | Mips指令 | 执行结果 | 含义 | 16进制机器码 | 结果截图： |
| 11 | l1:ori $11,$0,0xefef | 11号寄存器的内容变为0xefefef | 将0号寄存器中的内容与0xefef进行按位或运算，将结果存储到11号寄存器， | 340befef |  |

**4.12 指令12**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 指令序号 | Mips指令 | 执行结果 | 含义 | 16进制机器码 | 结果截图： |
| 12 | lui $9,0x4567 | 9号寄存器的数据为0x45670000 | 将立即数0x4567进行高位拓展并存入11号寄存器中。 | 3c094567 |  |

**4.13 指令13**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 指令序号 | Mips指令 | 执行结果 | 含义 | 16进制机器码 | 结果截图： |
| 13 | J l3 | 无条件跳转至l3指令 | 无条件跳转至l3指令 | 08000c07 |  |

**五、收获体会**

通过此次大作业，我深入了解了单周期处理器的逻辑组成，进而对计算机组成原理这门课的第四章有了更深刻的认识。在完成这次大作业的过程中，我不仅回顾了上个学期所学的全加器全减器、编码译码器的相关知识，更学习并实践了计算机硬件内部寻址、取址的详细过程，了解了一条指令如何转换为内存中实际的变化。通过与舍友的交流学习，以及学长的答疑解惑，我解决了不少计组这门课学习的困惑，有助于自己巩固知识点并深化认识的过程。再次向帮助过我的老师和同学们表达感谢。