|  |
| --- |
|  |
| Project2 MIPS多周期处理器-42指令核心模块说明 |
|  |
| 曾宇祥 |
|  |

## 模块定义

### 1.1. flopr（异步复位触发器）

#### 1.1.1. 基本描述

flopr主要异步复位触发器，可实例化为ALUOut、DR等无写使能信号的寄存器。当实例化flopr时，可使用#(XXX)，实例化位宽是XXX的触发器。

#### 模块接口

表格 1

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| rst | I | 复位信号 |
| d | I | 输入数据 |
| q | O | 输出数据 |

### 1.2. mux(多路选择器)

#### 1.2.1. 基本描述

mux主要功能是多路选择器。mux.v文件包含二选一、四选一、八选一、十六选一4中多路选择器。实例化多路选择器时，可使用#(XXX)，实例化位宽为XXX的多路选择器。

### 1.2.2. 模块接口

表格 2

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| d0、d1、d2... | I | 供选择数据（d0、d1） |
| s | I | 片选信号 |
| y | O | 片选后的数据 |

### 1.3. RF（寄存器文件）

#### 1.3.1.基本描述

RF主要功能是保存寄存器文件，并支持对通用寄存器的访问。

#### 1.3.2. 模块接口

表格3

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| A1 [4:0] | I | 需要读的寄存器1的地址 |
| A2 [4:0] | I | 需要读的寄存器2的地址 |
| A3 [4:0] | I | 需要写的寄存器的地址 |
| WD [31:0] | I | 需要写的寄存器的数据 |
| RFWr | I | 寄存器写使能端  0：寄存器不写  1：寄存器写 |
| clk | I | 时钟信号 |
| RD1 [31:0] | O | 需要读的寄存器1的数据 |
| RD2 [31:0] | O | 需要读的寄存器2的数据 |

#### 1.3.3. 功能定义

表格4

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 读取通用寄存器 | 根据输入的RS、RT域的值，输出相应通用寄存器所存储的数据。 |
| 2 | 写回通用寄存器 | 当RF写使能有效时，将待写的数据写入给定地址的通用寄存器中。 |

### 1.3. ALU（算术逻辑运算单元）

#### 1.3.1.基本描述

ALU主要功能是完成对输入数据的算数逻辑计算，支持42条指令涵盖的所有运算类型。

#### 1.3.2. 模块接口

表格5

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| A [31:0] | I | 操作数A |
| B [31:0] | I | 操作数B |
| ALUOp[4:0] | I | 需要进行的运算  编码详见ctrl\_encode\_def.v文件 |
| C [31:0] | O | 运算结果 |
| Compare | O | 即C[0]，用于分支类指令的比较结果。 |

#### 1.3.3. 功能定义

表格6

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 加法 | 执行加法运算 |
| 2 | 减法 | 执行减法运算 |
| 3 | 或运算 | 执行或运算 |
| 所支持的运算类型可参见ctrl\_encode\_def.v文件中的宏 | | |
| ...... | | |

### 1.4. EXT（扩展单元）

#### 1.4.1.基本描述

EXT主要功能是将16位的数据扩展为32位数据。

#### 1.4.2. 模块接口

表格7

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| Imm16 [15:0] | I | 需要进行扩展的数据 |
| EXTOP[1:0] | I | 扩展模式的控制信号  00：0扩展  01：符号扩展  10：将立即数扩展到高位 |
| Imm32 [31:0] | O | 扩展结果 |

#### 1.4.3. 功能定义

表格8

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 32位扩展 | 对16位立即数根据方式进行扩展 |

### 1.5. DM（数据存储器）

#### 1.5.1.基本描述

DM是数据存储器。相关参数可参考P2.docx文件。

#### 1.5.2. 模块接口

表格9

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| din [31:0] | I | 需要写回的数据 |
| DMWr | I | 读写操作的写使能端  0：禁止写  1：允许写 |
| addr[11:2] | I | 访问地址 |
| be[3:0] | I | 字节使能信号（针对sb/sh/sw指令） |
| dout[31:0] | O | 读出的数据 |

#### 1.5.3. 功能定义

表格10

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 读数据存储器 | 输出地址所对应的数据 |
| 2 | 写数据存储器 | 当写使能有效时，将待写数据写入对应地址 |

### 1.6. IM（指令存储器）

#### 1.6.1. 基本描述

IM是指令存储器。相关参数可参考P2.docx。

#### 1.6.2. 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| addr[11:2] | I | 访问地址 |
| dout[31:0] | O | 读出的指令 |

#### 1.6.3. 功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 读指令存储器 | 输出地址所对应的指令 |

### 1.7. becalc（计算DM.be）

#### 1.7.1. 基本描述

becalc主要功能是根据ALU计算的地址结果以及当前指令类型给出正确的字节使能be。

#### 1.7.2. 模块接口

表格11

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| ALUOut[1:0] | I | ALU计算结果的最后2位 |
| op[1:0] | I | 当前的指令类型，详见ctrl\_encode\_def.v文件 |
| be[3:0] | O | 计算出的be |

#### 1.7.3. 功能定义

表格12

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 计算be | 根据ALUOut和op计算出正确的be。 |

### 1.8. MemExtender(DM读出数据扩展)

#### 1.8.1. 基本描述

MemExtender主要功能是完成根据当前指令类型，将内存读出的数据挑出需要的字节，然后进行零扩展之后输出。（主要针对Load类指令）

#### 1.8.2. 模块接口

表格 13

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| ALUOut[1:0] | I | ALU计算结果的最后2位 |
| op[1:0] | I | 当前的指令类型，详见ctrl\_encode\_def.v文件 |
| din[32:0] | I | 内存读入的数据 |
| dout[32:0] | O | 挑出正确的部分之后输出的数据 |

#### 1.8.3 功能定义

表格 14

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 对内存读出的数据进行扩展 | 根据当前的指令类型，挑出32位输出数据中需要的几位，然后进行相应的扩展。 |

### 1.9. 宏定义文件

#### 1.9.1. global\_def.v

定义DEBUG宏，控制是否调试

#### 1.9.2. instruction\_def.v

定义42条指令的OPCODE/FUNCT等域的宏，其中BGEZ/BLTZ指令OP域相同，需要通过RT域进行判断。

#### 1.9.3. ctrl\_encode\_def.v

相关控制信号的宏。其中使用Verilog描述控制器的控制信号时，需与该宏保持一致。也可根据情况增加控制信号的宏。

## 测试

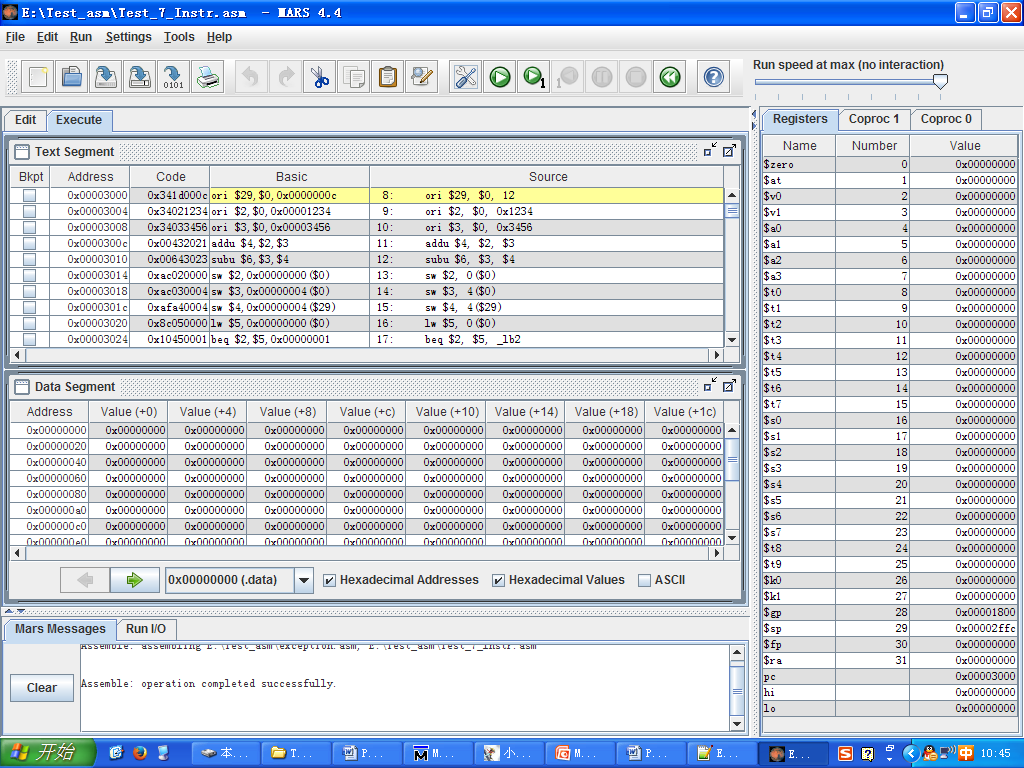
### 测试指令

见Test\_42\_Instr.asm文件

### 机器码文件生成

打开Mars4\_4.jar→ 打开Test\_42\_Instr.asm → 点击工具栏的如下图标可执行当前汇编指令  
2.png

点击后，界面如下：



其中，工具栏的按钮说明如下：下图1按钮可直接末；2按钮可单步执行；3按钮重新执行上一条指令；4按钮重新执行所有指令。



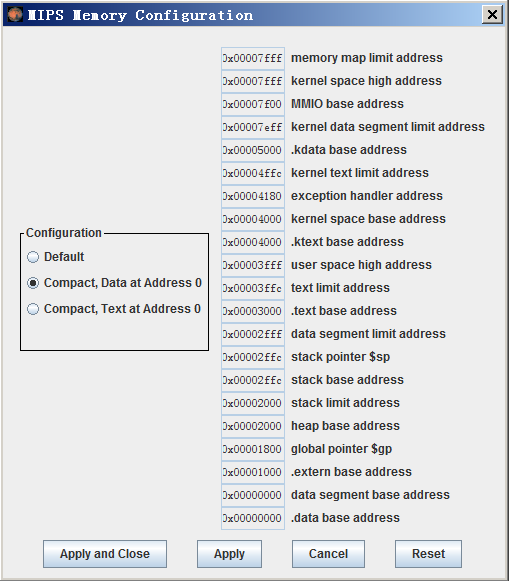
3

4

2

1

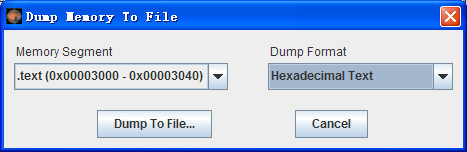
再运行指令前，需要保证Memory Configuration与处理器设计保持一致。可通过Settings → Memory Configuration设置。设置为如下模式后应用，即可执行汇编指令。



针对当前汇编指令，可单击工具栏如下按钮生成二进制文件：

4.png

单击后，显示为下图



选择Dump Format为Hex Text，生成16进制，单击Dump To File...按钮后选择路径，即可生成机器码作为多周期处理器测试文件。