Cpu\_PC\_IF\_ID\_Rtype

# 作用：

完成对R类型指令的取指令阶段、指令译码阶段、指令执行阶段和指令回写阶段，其中R类型指令如下表所示。

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Mnemonic Symbol** | **Format** | | | | | | **Sample** |
| **Bit #** | **31..26** | **25..21** | **20..16** | **15..11** | **10..6** | **5..0** |  |
| **R-type** | **op** | **rs** | **rt** | **rd** | **shamt** | **func** |  |
| add | 000000 | rs | rt | rd | 0 | 100000 | add $1,$2,$3 |
| addu | 000000 | rs | rt | rd | 0 | 100001 | addu $1,$2,$3 |
| sub | 000000 | rs | rt | rd | 0 | 100010 | sub $1,$2,$3 |
| subu | 000000 | rs | rt | rd | 0 | 100011 | subu $1,$2,$3 |
| and | 000000 | rs | rt | rd | 0 | 100100 | and $1,$2,$3 |
| or | 000000 | rs | rt | rd | 0 | 100101 | or $1,$2,$3 |
| xor | 000000 | rs | rt | rd | 0 | 100110 | xor $1,$2,$3 |
| nor | 000000 | rs | rt | rd | 0 | 100111 | nor $1,$2,$3 |
| slt | 000000 | rs | rt | rd | 0 | 101010 | slt $1,$2,$3 |
| sltu | 000000 | rs | rt | rd | 0 | 101011 | sltu $1,$2,$3 |
| sll | 000000 | 0 | rt | rd | shamt | 000000 | sll $1,$2,10 |
| srl | 000000 | 0 | rt | rd | shamt | 000010 | srl $1,$2,10 |
| sra | 000000 | 0 | rt | rd | shamt | 000011 | sra $1,$2,10 |
| sllv | 000000 | rs | rt | rd | 0 | 000100 | sllv $1,$2,$3 |
| srlv | 000000 | rs | rt | rd | 0 | 000110 | srlv $1,$2,$3 |
| srav | 000000 | rs | rt | rd | 0 | 000111 | srav $1,$2,$3 |
| jr | 000000 | rs | 0 | 0 | 0 | 001000 | jr $31 |

# 参数说明

module pipe\_id(

input clk,

input rst,

input [31:0] instr, //从取指阶段得到的当前指令

input [31:0] wrf\_data, //回写阶段写寄存器的值

input rf\_wena, //回写阶段写寄存器的写信号

input rf\_waddr, //回写阶段写寄存器的地址

output [31:0] rd1, //译码阶段，从寄存器组得到的源操作数a

output [31:0] rd2, //译码阶段，从寄存器组得到的源操作数b

output [31:0] shamt32, //译码阶段，指令shamt经过扩展得到的32位立即数

output [4:0] rd, //译码阶段，得到要写寄存器组的地址

//控制单元信号

output [3:0] aluc, //译码阶段，从控制单元得到的alu的控制信号

output wrf, //译码阶段，从控制单元得到的回写寄存器的写信号

output shift,

output [1:0] pcsource

);

# 设计流程

译码阶段主要包括以下两个模块：

1. 寄存器组

采用位宽为32的，深度为32的寄存器组，其中0号单元为常数0，不得用户修改。为了满足在单周期内完成一条指令的执行，在clk的下降沿，对寄存器进行最后回写阶段的写操作。

1. 控制单元

控制单元主要完成对数据通路中的各个模块的信号的控制。通过对op和funct的译码，得到相应指令对不同模块的不同信号的控制，其中对于R类型指令，除了jr指令都需要在执行阶段进行运算，需要给出alu的控制信号aluc，和回写阶段对寄存器组的写信号rf\_wena。

其中对alu的控制信号译码由执行阶段的ALU决定，可以参考ALU模块的aluc控制信号。

具体设计过程根据将Rtype指令的数据通路的形态分成不同的类型，一一实现，具体实现步骤如下图数据通路







# 设计要点

Pcsource[1]要给初始值0