一、RISC-V概要

1. 基本信息

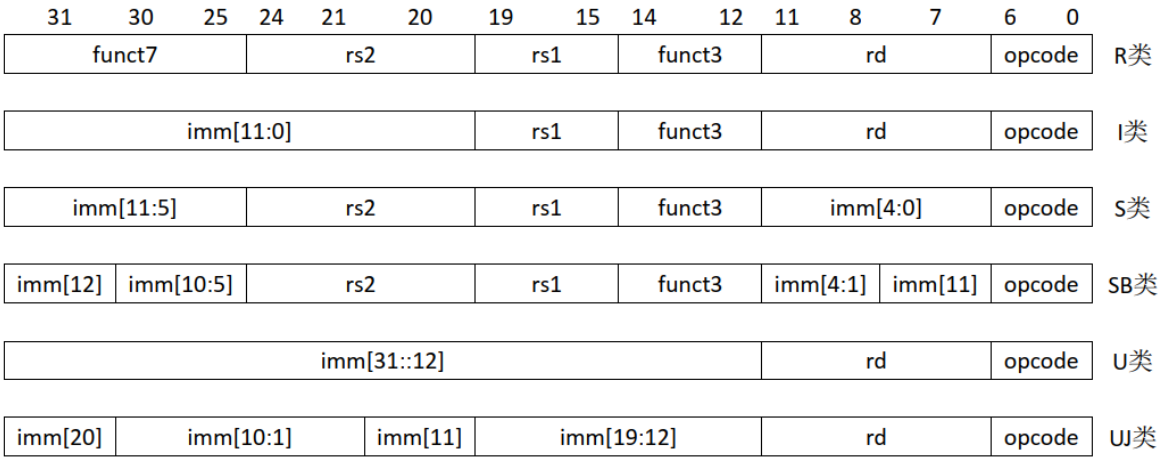
RISC-V是一个完全开放的ISA。其基础的32位整数指令集**RISC-V32I**仅有47条指令，若不考虑控制和状态寄存器指令则只有37条，相对于MIPS等**CISC**指令集来说十分精简。拓展指令集包括**RISC-V32M**、**RISC-V32A**、**RISC-V32F**、**RISC-V32D**等，分别拓展了乘除法操作、原子操作、单精度浮点操作和双精度浮点操作。本文档只讨论**RISC-V32I/M**指令集

2. 程序员模型

RISC-V32I有31个通用寄存器**x1~x31**，它们保存整数数值。寄存器**x0**是硬件连线的常数0。在一个过程调用中，标准软件调用约定使用寄存器**x1**来保存返回地址。 此外，还有一个额外的程序计数器**pc**用于保存当前指令地址。所有寄存器均为32位。

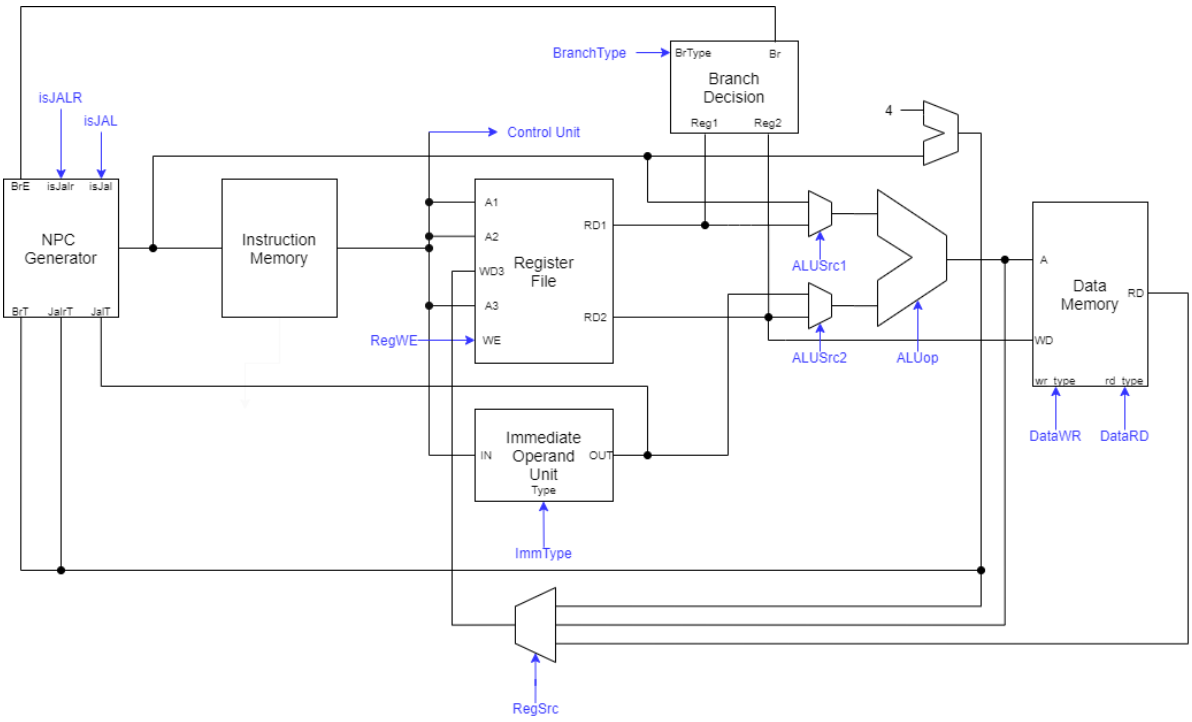
3. 指令格式

RISC-V32I有4种基本指令格式，分别为R、I、S、U。根据立即数编码方式不同，S、U类分别有变种SB、UJ。4种基本指令格式及2种变种格式的具体格式如下图所示：



4.数据通路

在这里给出**RISC-V32I**单周期CPU的数据通路如下。其中蓝色文字与连线表示译码器模块或控制信号。



二、部件说明

1. ALU

逻辑运算单元（Arithmetic and Logic Unit, ALU）是CPU的核心组件之一，用于完成对两个操作数的一元或二元运算。在**RISC-V32I**指令集中，ALU需支持加、减等10种二元运算操作；在**RISC-V32M**指令集中，需要另外拓展支持乘法（输出结果低32位）、乘法（输出结果高32位）、除法、取余4种操作。

输入端口

端口名	位数	内容
SrcA	32	第一个操作数
SrcB	32	第二个操作数
func	4	运算类型

- `SrcA`和`SrcB`为32位操作数，通过多路选择器选择来源。
- `func`为4位的操作码，最多可支持16种运算。在拓展ALU操作前，需要考虑拓展`func`位数。为方便设计译码器，给定`func`的值与运算类型对应如下：

func	运算	指令	输出
0000	加法	ADD	SrcA + SrcB
1000	减法	SUB	SrcA - SrcB
0001	逻辑左移	SLL	SrcA << SrcB[0:4]
0010	有符号小于	SLT	SrcA < SrcB ? 1 : 0
0011	无符号小于	SLTU	SrcA < SrcB ? 1 : 0
0100	异或	XOR	SrcA ⊕ SrcB
0101	逻辑右移	SRL	SrcA >> SrcB[0:4]
1101	算术右移	SRA	SrcA >>> SrcB[0:4]
0110	或	OR	SrcA || SrcB
0111	与	AND	SrcA & SrcB
1001	未确定	-	0
1010	未确定	-	0
1011	未确定	-	0
1100	未确定	-	0
1110	无运算	-	SrcB
1111	无运算	-	0

输出端口

端口名	位数	内容
Result	32	运算结果

- `Result`为运算结果，通过多路选择器选择来自不同运算部件的结果。`Func`为多路选择器的选择信号。

注意事项

RISC-V32I不提供整数运算的溢出检测，因此ALU中**不需要**输出OF溢出标志。为了检测溢出，需要自行增加额外的指令，其中OF为溢出处理代码：

```
ADD x5，x6，x7
SLTI x28，x7，0
SLT x29，x5，x6
BNE x28，x29，OF
```

其原理为原理：$x6+x7=x5+0$，$x7$和0的大小关系应与$x6$和$x5$的大小关系相反。

存在两个较简单的特殊情况：

- 无符号数相加：

```
ADD x5，x6，x7
BLTU x5，x6，OF
```

- 有符号数的立即数加，且立即数为正数：

```
DDI x5，x6，+imm
BLT x5，x6，OF
```

2. 寄存器

寄存器文件（Register Files, REG）用于存储多个32位的二进制数。在**RISC-V32I**中，共有32个寄存器。其中，**x0**为零寄存器，只会输出0，屏蔽所有写操作；**x1~x31**为通用寄存器，可以进行正常的读写操作。

输入端口

端口名	位数	内容
CLK	1	时钟信号
A1	5	读寄存器地址
A2	5	读寄存器地址
A3	5	写寄存器地址
WD3	32	写寄存器数据
WE3	1	寄存器写使能信号

- `CLK`为时钟信号端口。在CPU中，为所有依赖时钟边沿进行操作的部件有一个CLK端口，CPU使用专门的时钟信号单元产生时序信号，从而使得各部件能够同步工作。
- `A1`、`A2`为读地址端口。寄存器文件以`A1`作为选路信号，使用多路选择器从32个寄存器选择一个作为输出`RD1`。`RD2`同理。
- `A3`为写地址端口，若`WE3`有效，则将`WD3`写入`A3`对应的寄存器。

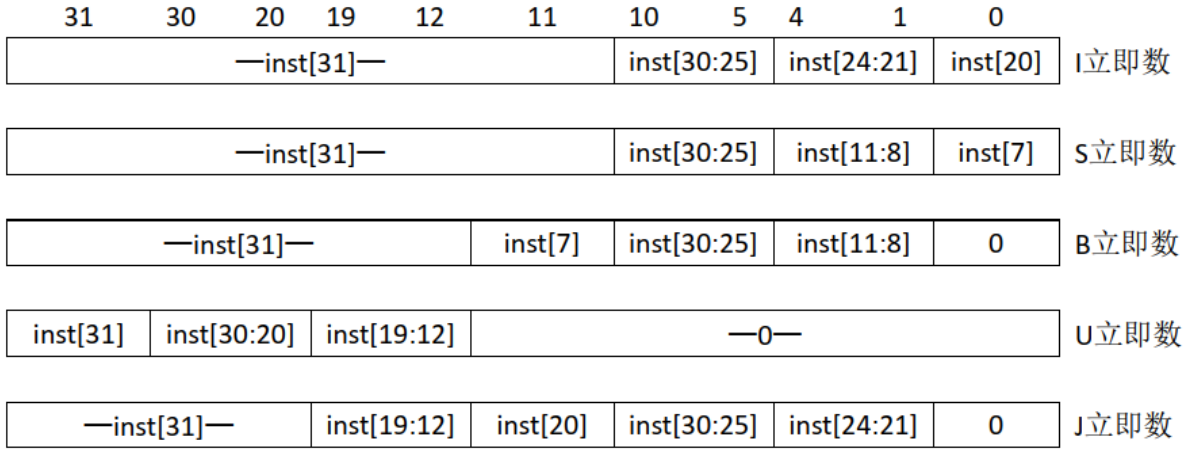
输出端口

端口名	位数	内容
RD1	32	读出的寄存器数据
RD2	32	读出的寄存器数据

- `RD1`和`RD2`是根据`A1`和`A2`从寄存器中读出的数据。

3. 立即数产生器

指令集存在5种立即数格式，因此需要一个机构根据不同格式生成立即数。立即数生成器接受指令本身和来自控制器的信号输入，根据控制信号提取出指令中的立即数并输出。5种立即数编码方式如下图所示：



输入端口

端口名	位数	内容
instr	32	指令
type	3	立即数种类

- `type`表示立即数种类。由于输入了32位的指令本身，`type`的信息可以从`instr`中得到，但为了保持控制相关信号的统一，我们依然在控制器中生成该信号。

输出端口

端口名	位数	内容
imm	32	立即数

4. 程序计数器（程序地址生成器）

程序计数器（Program Counter， PC）记录了当前指令的地址。每执行完一条指令，PC需要根据执行结果确定下一条指令的地址。

输入端口

端口名	位数	内容
CLK	1	时钟信号
BrT	32	分支跳转地址
JalR	32	无条件跳转地址
JalrT	32	无条件相对跳转地址
isBr	1	分支跳转使能
isJal	1	无条件跳转使能
isJalr	1	无条件相对使能

- `isBr`、`isJal`、`isJalr`三个信号最多只有一个有效。当这三个信号存在有效信号时，选择对应的跳转地址；否则，选择PC+4。

输出端口

端口名	位数	内容
PCout	32	当前指令地址

5. 译码器

译码器（Decoder），用于解析指令内容并输出控制信号。译码器产生控制信号的方式通常有硬布线和微指令两种，这里采用硬布线控制，即只使用组合逻辑电路产生控制信号。

输入端口

端口名	位数	内容
instr	32	指令

输出端口

端口名	位数	内容
ALUSrc1	1	ALU源操作数`SrcA`选择信号
ALUSrc2	1	ALU源操作数`SrcB`选择信号
ALUop	4	ALU运算类型
ImmType	3	立即数拓展类型
BrType	3	分支类型
RegWR	1	寄存器写信号
RegSrc	2	寄存器写回数据`WD3`选择信号
DataWR	2	数据存储器写类型
DataRD	3	数据存储器读类型
isJAL	1	无条件跳转使能
isJALR	1	无条件相对跳转使能

6. 分支信号产生器

指令集中含有6种分支指令。由于ALU不能同时计算分支目标地址与判断是否分支，因此需要一个机构用于判断分支成功与否。

输入端口

端口名	位数	内容
REG1	32	寄存器读端口1
REG2	32	寄存器读端口2
Type	3	指令的12-14位

- `REG1`和`REG2`分别为寄存器读出的两个值。分支指令总是使用到这两个端口的输出作为输入。
- `Type`为当前指令的第12-14位。`Type`与分支类型对应如下：

Type	操作	指令	输出
000	相等判断	BEQ	REG1 == REG2 ? 1 : 0
001	不等判断	BNE	REG1 != REG2 ? 1 : 0
100	小于判断	BLT	REG1 < REG2 ? 1 : 0
101	大于等于判断	BGE	REG1 >= REG2 ? 1 : 0
110	无符号小于判断	BLTU	REG1 < REG2 ? 1 : 0
111	无符号大于等于判断	BGEU	REG1 >= REG2 ? 1 : 0
其他	无操作	无	0

输出端口

端口名	位数	内容
BrE	1	分支使能

7. 指令存储器

指令存储器用于存放指令。

输入端口

端口名	位数	内容
Addr	32	指令地址

- `Addr`为32位宽。根据指令存储器大小不同，实际使用到的位数也有一定差别。在存储器大小为8KB时，需使用到`Addr`的0~12位。由于存储器按字读取，实际接入存储器的部分为`Addr`的2~12位，第0、1位总为00。

输出端口

端口名	位数	内容
instr	32	指令

注意事项

RISC-V指定存储器采用小端字节顺序，但也允许使用大端字节顺序或双端顺序。在logisim中，存储器为大端字节存储且无法修改存放方式，为了方便设计和调试，我们采用大端存储。数据存储器也同理。

8. 数据存储器

数据存储器用于存储数据。通常，存储器是按字存取的。为了支持半字存取以及字节存取，需要增加一系列组合逻辑电路截取、拓展读写的整字。

输入端口

端口名	位数	内容
CLK	1	时钟信号
Addr	32	读/写地址
Din	32	写入数据
WR	2	写控制信号
RD	3	读控制信号

- `Addr`为32位地址。根据数据存储器大小不同，实际使用到的位数也有一定差别。在存储器大小为8KB时，需使用到`Addr`的0~12位。由于存储器按字读取，实际接入存储器的部分为`Addr`的2~12位，第0、1位在读写半字或字节时用于选择。
- `WR`为写控制信号，其值与功能对应如下：

WR	操作	指令
00	写入字节	SB
01	写入半字	SH
10	写入整字	SW
11	不写入	-

- `RD`为读控制信号。其最高位表示读出半字或字节时的拓展方式，低2位表示读取方式。其值与功能对应如下：

RD[0:1]	操作	指令
00	读字节	LB/LBU
01	读半字	LH/LHU
10	读整字	LW
11	无操作	-

RD[2]	操作
0	符号扩展
1	零扩展

输出端口

端口名	位数	内容
Dout	32	读出数据

注意事项

访存时的对齐，指的是地址须是访问长度的整数倍，即访问32位（4字节）时地址为4的整数倍，访问16位（2字节）时地址为2的整数倍。

RISC-V的基本ISA支持非对齐的访问，但这实现起来会相对复杂，并且不是很有必要，因此在这里只实现对齐访问。此外，在实际实现中，非对齐的访存可能会增加大量时间开销，并且可能要多次访存，这使得访存操作不是原子的，存在潜在的危险。