

课程报告

课程名称： 计算机组成原理

报告题目： RISC处理器设计

所在院系： 计算学部

所在专业： 软件工程

学生姓名： 瞿久尧

学生学号： 120L022314

选课时间： 2022年秋季学期

评阅成绩：

目录

[1. 指令格式设计 3](#_Toc21552)

[1.1 单寄存器指令设计 3](#_Toc4623)

[1.1.1 LDA 3](#_Toc13473)

[1.1.2 STA 4](#_Toc17421)

[1.1.3 MVI 4](#_Toc19793)

[1.1.4 JZ 4](#_Toc29724)

[1.2 两寄存器指令格式设计 5](#_Toc15463)

[1.2.1 MOV 5](#_Toc24625)

[1.2.2 ADD 5](#_Toc9926)

[1.2.3 SUB 5](#_Toc1370)

[1.3 无寄存器指令格式设计 5](#_Toc14868)

[1.3.1 JMP 5](#_Toc13364)

[1.4 I/O指令格式设计 6](#_Toc8160)

[1.4.1 IN 6](#_Toc9704)

[1.4.2 OUT 6](#_Toc29467)

[1.5 指令设计表 6](#_Toc32334)

[2. 微操作的定义 7](#_Toc4675)

[2.1 非访存指令 7](#_Toc19429)

[2.1.1 ADD 7](#_Toc15341)

[2.1.2 SUB 7](#_Toc23739)

[2.1.3 MOV 8](#_Toc4239)

[2.1.4 MVI 8](#_Toc14213)

[2.2 访存指令 9](#_Toc11199)

[2.2.1 LDA 9](#_Toc17005)

[2.2.2 STA 9](#_Toc22592)

[2.3 转移指令 10](#_Toc11620)

[2.3.1 JZ 10](#_Toc15758)

[2.3.2 JMP 10](#_Toc9214)

[2.4 I/O指令 11](#_Toc6867)

[2.4.1 IN 11](#_Toc11034)

[2.4.2 OUT 11](#_Toc609)

[3. 节拍的划分 12](#_Toc28904)

[3.1 取指周期 12](#_Toc25131)

[3.2 译码周期 12](#_Toc18959)

[3.2.1 ADD指令 12](#_Toc18639)

[3.2.2 SUB指令 12](#_Toc11512)

[3.2.3 MOV指令 13](#_Toc5170)

[3.2.4 MVI指令 13](#_Toc17553)

[3.2.5 LDA指令 13](#_Toc24000)

[3.2.6 STA指令 13](#_Toc15872)

[3.2.7 JZ指令 13](#_Toc22979)

[3.2.8 JMP指令 13](#_Toc20771)

[3.2.9 IN指令 13](#_Toc19064)

[3.2.10 OUT指令 14](#_Toc16845)

[3.3 运算周期 14](#_Toc27848)

[3.3.1 ADD MOV MVI I/O指令 14](#_Toc18456)

[3.3.2 SUB指令 14](#_Toc13265)

[3.3.3 LDA STA JZ JMP指令 14](#_Toc88)

[3.4 访存周期 14](#_Toc4511)

[3.4.1 LDA指令 14](#_Toc18273)

[3.4.2 STA指令 14](#_Toc16071)

[3.4.3 IN指令 15](#_Toc20032)

[3.4.4 OUT指令 15](#_Toc16323)

[3.5 写回周期 15](#_Toc18596)

[3.5.1 ADD SUB MOV MVI指令 15](#_Toc7738)

[3.5.2 LDA指令 15](#_Toc29560)

[3.5.3 STA OUT指令 15](#_Toc15784)

[3.5.4 JZ指令 15](#_Toc23950)

[3.5.5 JMP指令 16](#_Toc11760)

[3.5.6 IN指令 16](#_Toc27197)

[4. 处理器结构设计框图及功能描述 16](#_Toc13286)

[4.1 PC 16](#_Toc1641)

[4.2 IR 16](#_Toc16090)

[4.3 Memory 17](#_Toc2636)

[4.4 CU 18](#_Toc31642)

[4.5 dataBUS 18](#_Toc22257)

[4.6 ALU 19](#_Toc30872)

[4.7 PORT 19](#_Toc29777)

[4.8 CPU 20](#_Toc11273)

[5. 操作时间表和控制信号逻辑图 20](#_Toc23907)

[5.1 各指令操作时间表 21](#_Toc29758)

[5.2 控制信号的逻辑框图 22](#_Toc20898)

[5.2.1 逻辑表达式表 22](#_Toc11143)

[5.2.2 逻辑图 22](#_Toc32728)

# 指令格式设计

## 单寄存器指令设计

### LDA

指令: LDA Ri X

OP字段: 0000

Ri: 四位

X: 八位

功能: 将主存单元的内容存于寄存器Ri中，X为形式地址，经扩充寻址生成真实地址，扩充寻址寄存器为R7。

具体操作: [R7//X] → Ri

### STA

指令: STA Ri X

OP字段: 0001

Ri: 四位

X: 八位

功能: 将寄存器Ri的内容存于主存单元中，X为形式地址，经扩充寻址生成真实地址，扩充寻址寄存器为R7。

具体操作: Ri → [R7//X]

### MVI

指令: MVI Ri X

OP字段: 0010

Ri: 四位

X: 八位

功能: 完成将指令中的立即数X向寄存器Ri中传送的操作。

具体操作: X → Ri

### JZ

指令: JZ Ri X

OP字段: 0011

Ri: 四位

X: 八位

功能: 指令根据寄存器Ri的内容决定下一条指令的地址，若为0，则八位形式地址扩充后的新地址送至PC，否则程序按原顺序执行，即PC = PC + 1

具体操作: if (Ri = 0) then [R7//X] → PC

## 两寄存器指令格式设计

### MOV

指令: MOV Ri Rj

OP字段: 0100

填充字段: 四位0000

Ri: 四位

Rj: 四位

功能: 将寄存器Rj的数据向Ri传送

具体操作: Rj → Ri

### ADD

指令: ADD Ri Rj

OP字段: 0101

填充字段: 四位0000

Ri: 四位

Rj: 四位

功能: 将两个寄存器内容相加，送回Ri寄存器

具体操作: Ri + Rj → Ri

### SUB

指令: SUB Ri Rj

OP字段: 0110

填充字段: 四位0000

Ri: 四位

Rj: 四位

功能: 将两个寄存器内容相减，送回Ri寄存器

具体操作: Ri - Rj → Ri

## 无寄存器指令格式设计

### JMP

指令: JMP X

OP字段: 0111

填充字段: 四位0000

X: 八位

功能: 无条件转移，将八位形式地址扩充后的新地址送至PC。

具体操作: [R7//X] → PC

## I/O指令格式设计

### IN

指令: IN Ri X

OP字段: 1000

Ri: 四位

X: 八位

功能: 从IO端口向CPU传送信息

具体操作: [PORT] → Ri

### OUT

指令: OUT Ri X

OP字段: 1001

Ri: 四位

X: 八位

功能: 从CPU向IO端口传送信息

具体操作: Ri → [PORT]

## 指令设计表



表1: 指令设计表



表2: 寄存器编号表

# 微操作的定义

## 非访存指令

### ADD

取指阶段:

PC → MAR

M(MAR) → MDR

MDR → IR

译码阶段:

OP(IR) → CU

R(Ad1(IR)) → valA

R(Ad2(IR)) → valB

运算阶段:

valA + valB → valR

访存阶段:

无

写回阶段:

valR → R(Ad1(IR))

PC + 1 → PC

### SUB

取指阶段:

PC → MAR

M(MAR) → MDR

MDR → IR

译码阶段:

OP(IR) → CU

R(Ad1(IR)) → valA

R(Ad2(IR)) → valB

运算阶段:

valA - valB → valR

访存阶段:

无

写回阶段:

valR → R(Ad1(IR))

PC + 1 → PC

### MOV

取指阶段:

PC → MAR

M(MAR) → MDR

MDR → IR

译码阶段:

OP(IR) → CU

R(Ad2(IR)) → valA

0 → valB

运算阶段:

valA + valB → valR

访存阶段:

无

写回阶段:

valR → R(Ad1(IR))

PC + 1 → PC

### MVI

取指阶段:

PC → MAR

M(MAR) → MDR

MDR → IR

译码阶段:

OP(IR) → CU

X → valA

0 → valB

运算阶段:

valA + valB → valR

访存阶段:

无

写回阶段:

valR → R(Ad(IR))

PC + 1 → PC

## 访存指令

### LDA

取指阶段:

PC → MAR

M(MAR) → MDR

MDR → IR

译码阶段:

OP(IR) → CU

X → valB

R7 → valA

运算阶段:

valA//valB → valR

访存阶段:

valR → MAR

M(MAR) → MDR

写回阶段:

MDR → R(Ad(IR))

PC + 1 → PC

### STA

取指阶段:

PC → MAR

M(MAR) → MDR

MDR → IR

译码阶段:

OP(IR) → CU

X → valB

R7 → valA

运算阶段:

valA//valB → valR

访存阶段:

valR → MAR

R(Ad(IR)) → MDR

MDR → M(MAR)

写回阶段:

PC + 1 → PC

## 转移指令

### JZ

取指阶段:

PC → MAR

M(MAR) → MDR

MDR → IR

译码阶段:

OP(IR) → CU

R7 → valA

X → valB

运算阶段:

valA//valB → valR

访存阶段: 无

写回阶段:

PC + 1 → PC

if : R(Ad(IR)) = 0

valR → PC

### JMP

取指阶段:

PC → MAR

M(MAR) → MDR

MDR → IR

译码阶段:

OP(IR) → CU

R7 → valA

X → valB

运算阶段:

valA//valB → valR

访存阶段: 无

写回阶段:

valR → PC

## I/O指令

### IN

取指阶段:

PC → MAR

M(MAR) → MDR

MDR → IR

译码阶段:

OP(IR) → CU

PORT → valA

0 → valB

运算阶段:

valA + valB → valR

访存阶段:

valR → PAB

M(PAB) → PDB

写回阶段:

PDB → R(Ad(IR))

PC + 1 → PC

### OUT

取指阶段:

PC → MAR

M(MAR) → MDR

MDR → IR

译码阶段:

OP(IR) → CU

PORT → valA

0 → valB

运算阶段:

valA + valB → valR

访存阶段:

valR → PAB

R(AD(IR)) → PDB

PDB → M(PAB)

写回阶段:

PC + 1 → PC

# 节拍的划分

对于上述的五个周期，即取指，译码，运算，访存，写回五个步骤，每个机器周期最多三个节拍 T0，T1，T2

## 取指周期

对于所有指令，取指周期都分为同样的三个节拍

T0: PC → MAR

T1: M(MAR) → MDR

T2: MDR → IR

## 译码周期

### ADD指令

T0: OP(IR) → CU

T1: 空

T2: R(Ad1(IR)) → valA; R(Ad2(IR)) → valB

### SUB指令

T0: OP(IR) → CU

T1: 空

T2: R(Ad1(IR)) → valA; R(Ad2(IR)) → valB

### MOV指令

T0: OP(IR) → CU

T1: 空

T2: R(Ad2(IR)) → valA; 0 → valB

### MVI指令

T0: OP(IR) → CU

T1: 空

T2: X → valA ; 0 → valB

### LDA指令

T0: OP(IR) → CU

T1: 无

T2: X → valB ; R7 → valA

### STA指令

T0: OP(IR) → CU

T1: 无

T2: X → valB ; R7 → valA

### JZ指令

T0: OP(IR) → CU

T1: 无

T2: X → valB ; R7 → valA

### JMP指令

T0: OP(IR) → CU

T1: 无

T2: X → valB ; R7 → valA

### IN指令

T0: OP(IR) → CU

T1: 无

T2: 0 → valB ; PORT → valA

### OUT指令

T0: OP(IR) → CU

T1: 无

T2: 0 → valB ; PORT → valA

## 运算周期

### ADD MOV MVI I/O指令

T0: 无

T1: 无

T2: valA + valB → valR

### SUB指令

T0: 无

T1: 无

T2: valA - valB → valR

### LDA STA JZ JMP指令

T0: 无

T1: 无

T2: valA//valB → valR

## 访存周期

除LDA,STA,I/O指令外，其它指令访存周期均无需进行任何操作

### LDA指令

T0: valR → MAR

T1: 无

T2: M(MAR) → MDR

### STA指令

T0: valR → MAR

T1: R(Ad(IR)) → MDR

T2: MDR → M(MAR)

### IN指令

T0: valR → PAB

T1: 无

T2: M(PAB) → PDB

### OUT指令

T0: valR → PAB

T1: R(Ad(IR)) → PDB

T2: PDB → M(PAB)

## 写回周期

### ADD SUB MOV MVI指令

T0: PC + 1 → PC

T1: valR → R(Ad1(IR))

T2: 无

### LDA指令

T0: PC + 1 → PC

T1: MDR → R(Ad(IR))

T2: 无

### STA OUT指令

T0: PC + 1 → PC

T1: 无

T2: 无

### JZ指令

T0: PC + 1 → PC

T1: 无

T2: if : R(Ad(IR)) = 0 ; valR → PC

### JMP指令

T0: PC + 1 → PC

T1: 无

T2: valR → PC

### IN指令

T0: PC + 1 → PC

T1: PDB → R(Ad(IR))

T2: 无

1. 处理器结构设计框图及功能描述

## PC

PC设计图如下:

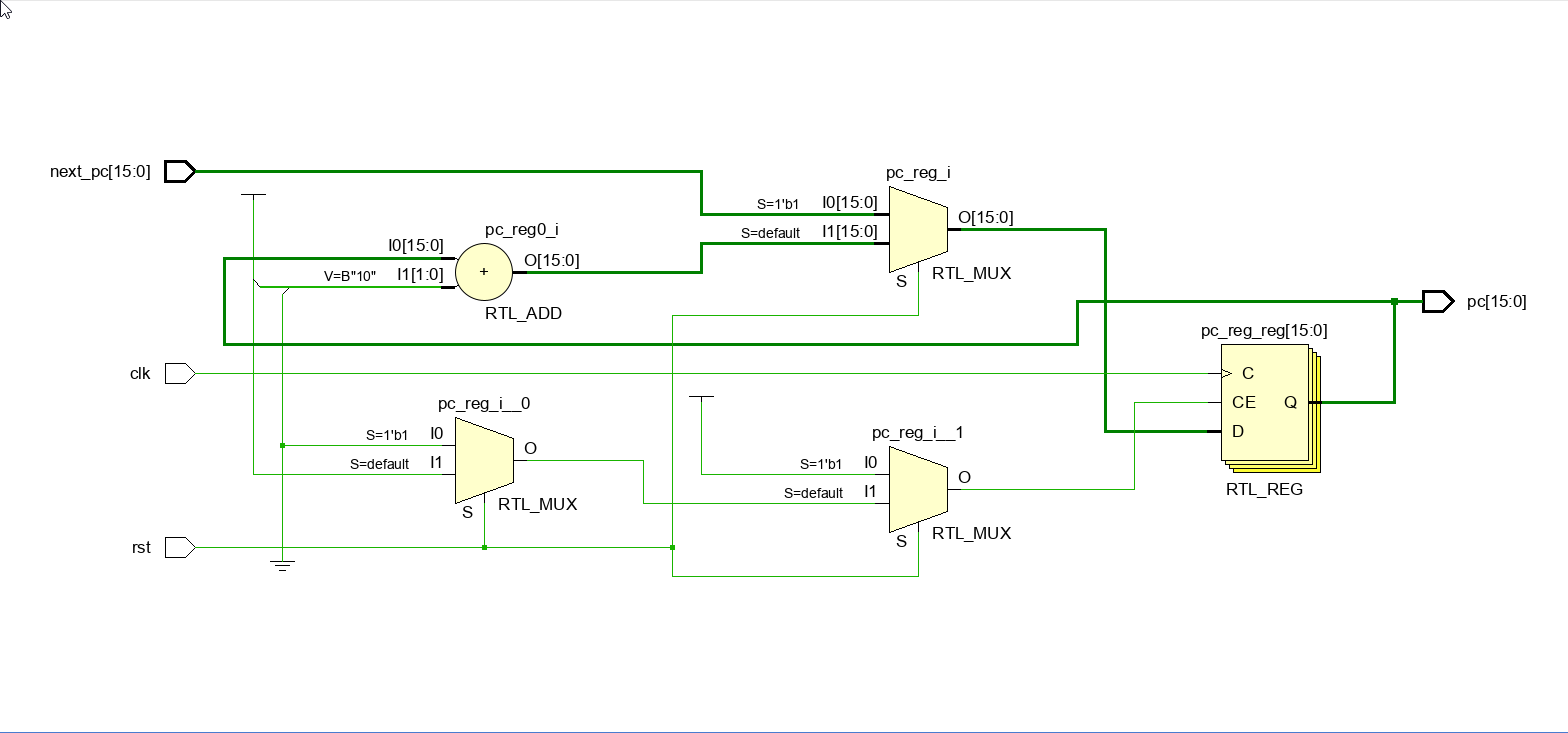


图4-1: PC设计图

功能设计:

PC的输出决定了接下来的几个周期中CPU执行什么任务，这里的PC负责在next\_pc，即如果发生跳转将执行的指令地址和原PC+2中按照rst的值筛选出一个应执行的PC值并输出。

## IR

IR设计图如下:

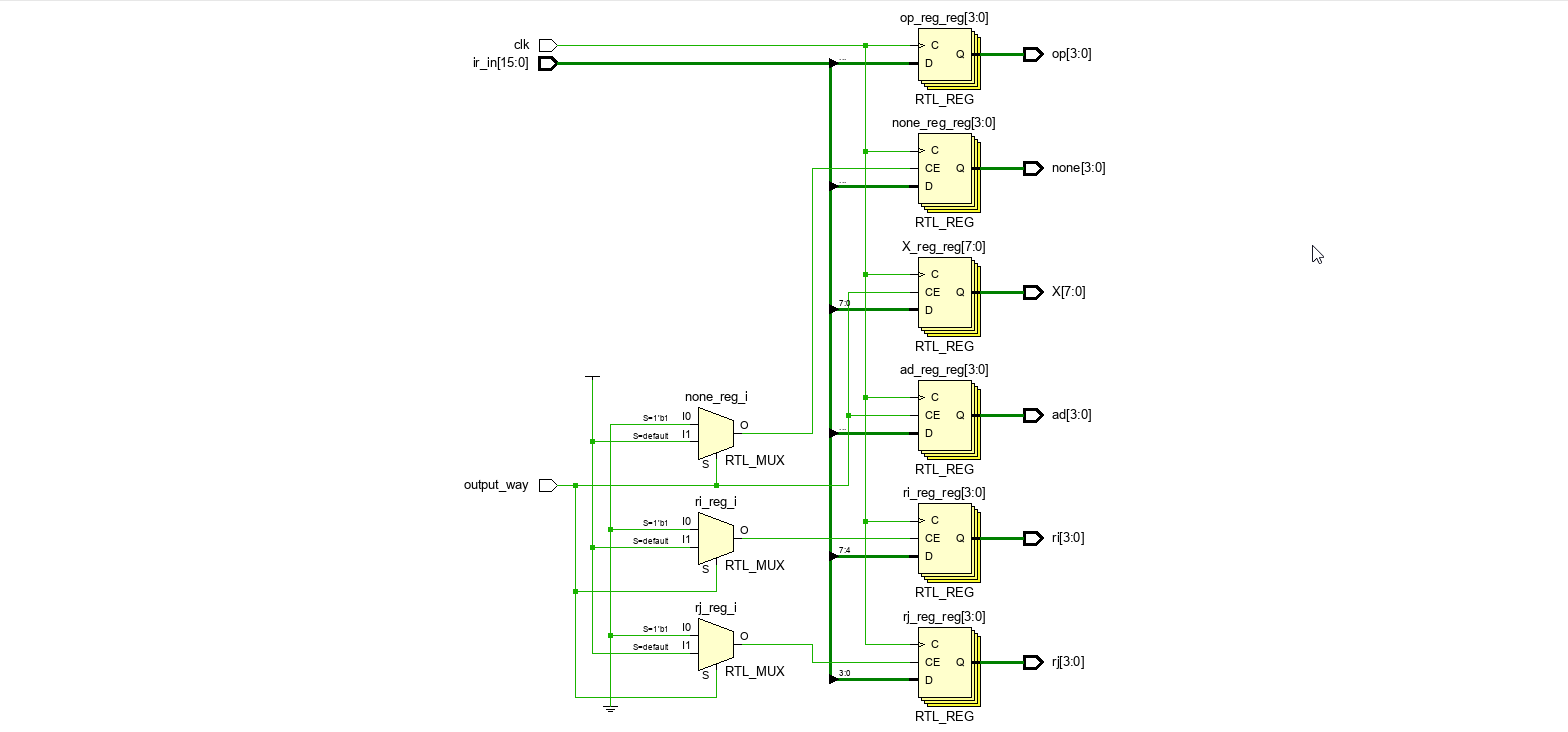


图4-2: IR设计图

功能设计:IR为指令寄存器，存入指令的同时根据译码器传来的信号将机器指令分割并输出，指令实际上可以分成两类，一类是op+R+X，另一类是OP+0000+ri+rj，所以只需要一位output\_way就可以完成输出类型的选择。

## Memory

Memory内部设计图如下:

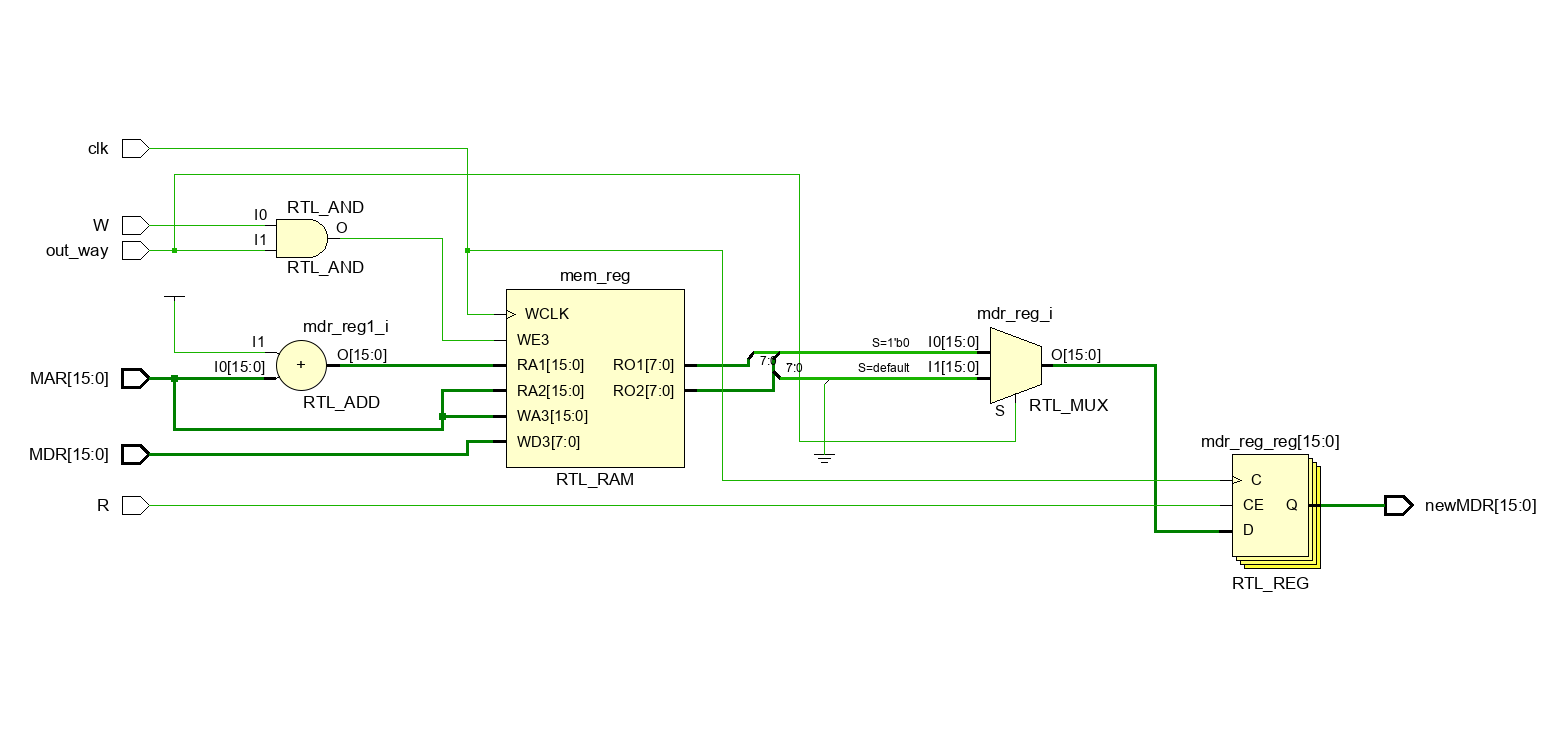


图4-3: Memory设计图

功能设计:

功能上MAR和MDR都看作它的一部分，利用两个使能点R和W来实现读写使能，在取指部分，R为1而W为0，out\_way为0，主存读取MAR传递进来的值，并两次访问存储单元，将得到的数据拼接起来，并输出给newMDR，在访存阶段，根据读写状态设置R，W的值，并存取数据。

## CU

CU内部设计图如下:

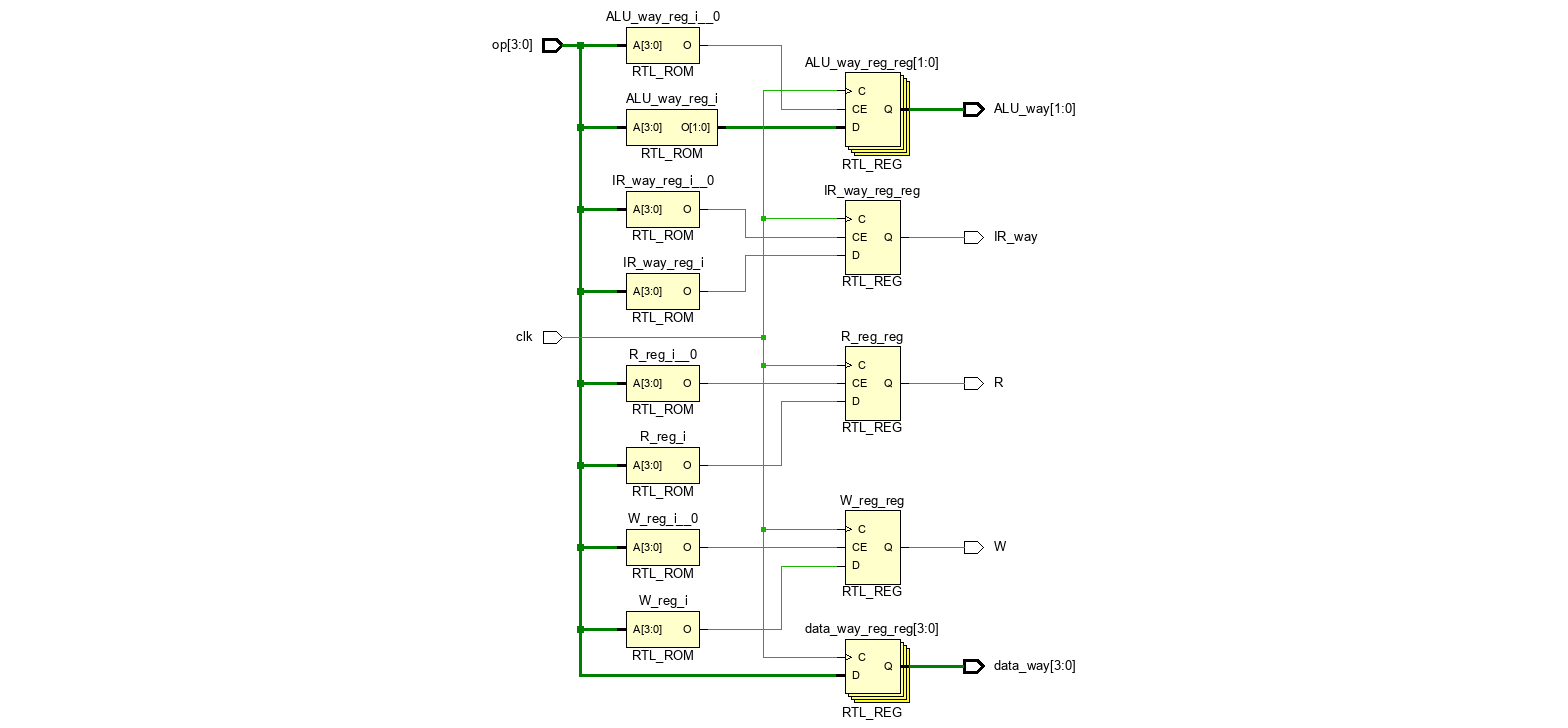


图4-4: CU设计图

功能: 通过分析传入的OP参数来生成一些必要的标志位，从而控制整个CPU内部的处理方式，并输出data\_way，用来控制接下来的数据流动方向。

## dataBUS

dataBUS设计图如下:

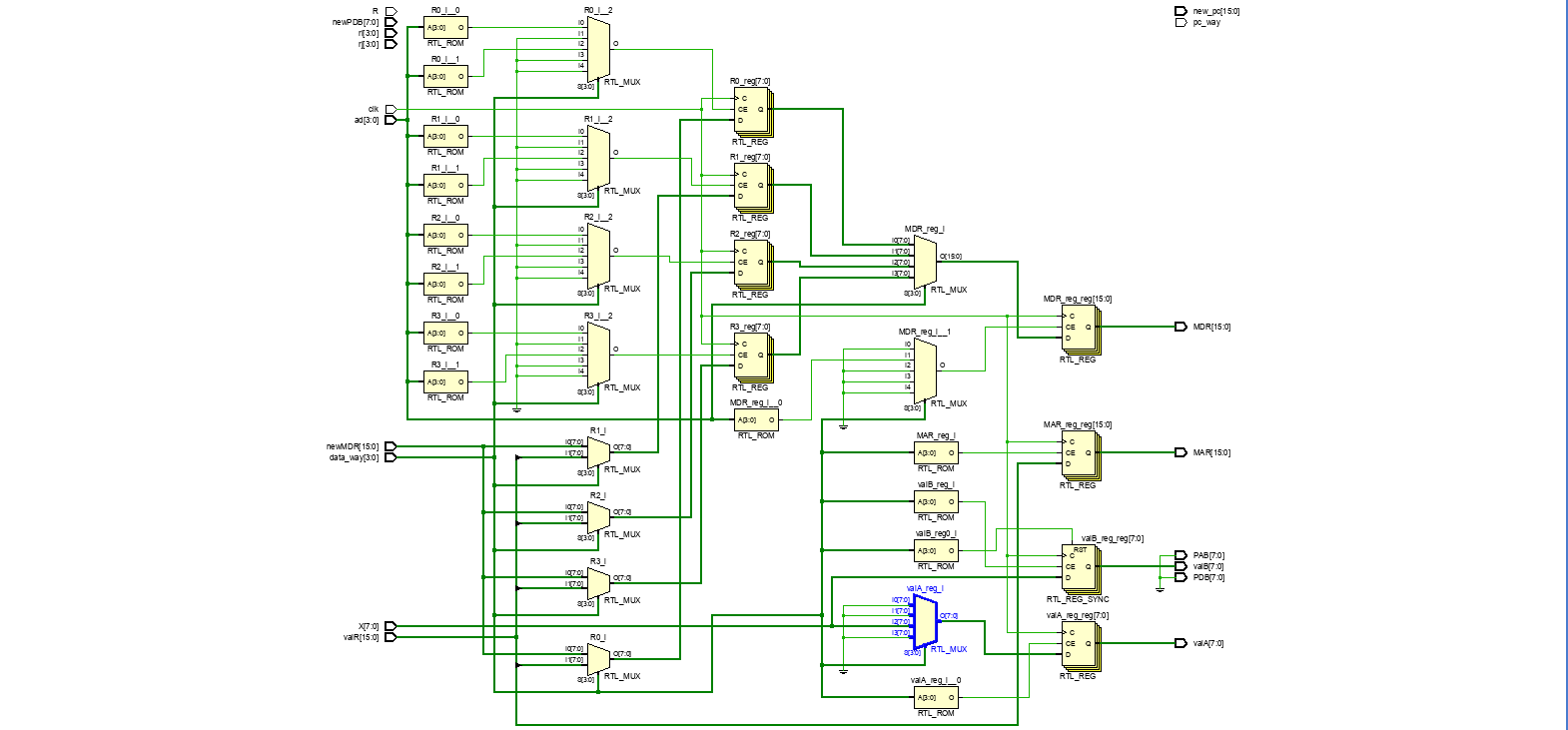


图4-5: dataBUS设计图

功能: 通过传递的dataway来实现选择数据流向，并配合时钟实现数组总线的功能，在数据读取，传递，写回阶段，流向以及数据的分配都由dataBUS所控制。

## ALU

ALU设计图如下:

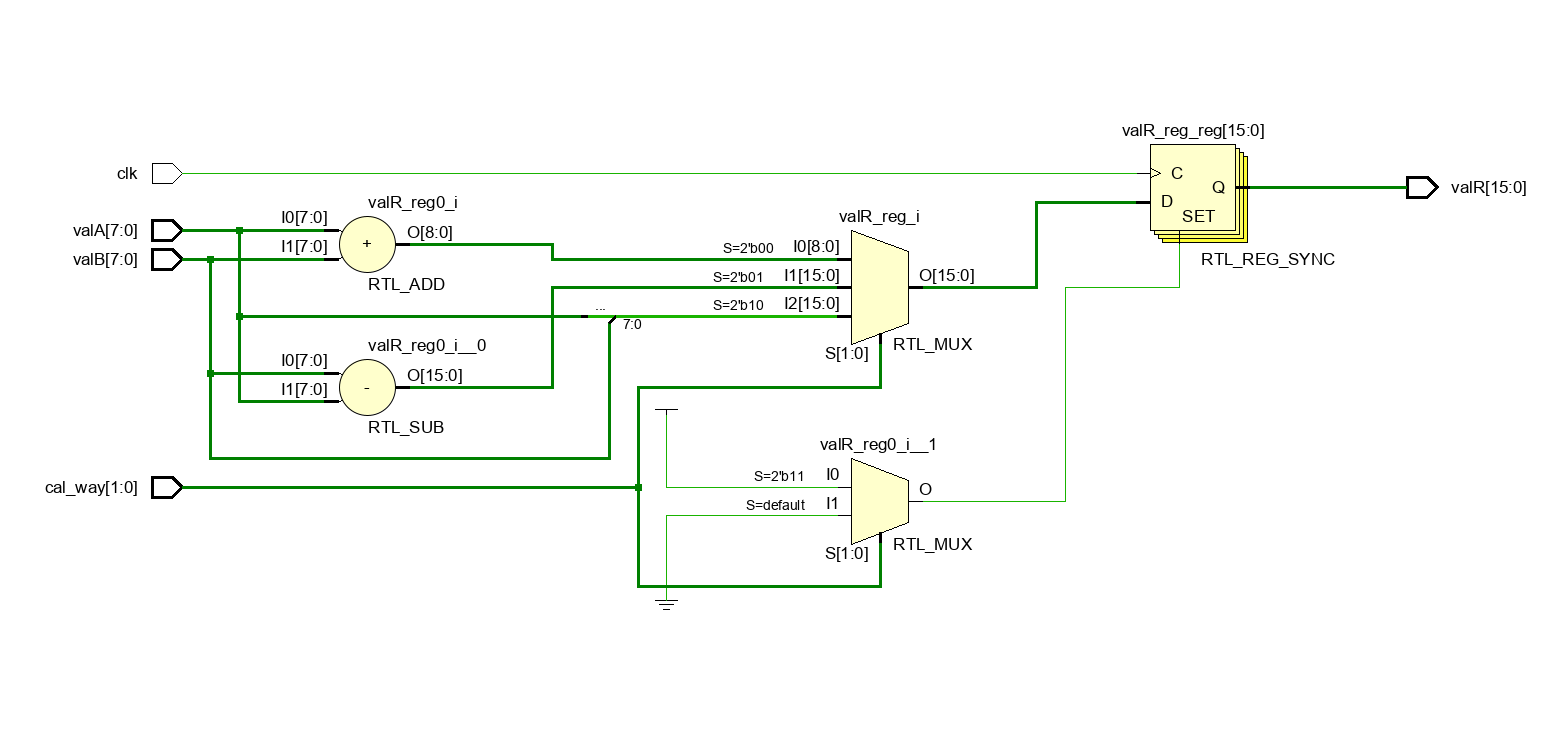


图4-6: ALU设计图

功能设计:

clk为时钟控制输入，上升沿触发将输出数据更新

valA和valB为两个输入的八位二进制数，ALU对他们分别执行加，减，拼接三种操作。

cal\_way是运算方式选择参数，输入00将执行加法操作，输入01将执行减法操作，输入10将执行拼接操作，输入11将输出全1的16位二进制数，实际上在ALU内部，三种方式的运算同步进行，通过选择输出的方式来实现选择运算。

valR为输出值，输出经过选择的结果。

## PORT

PORT设计图如下:

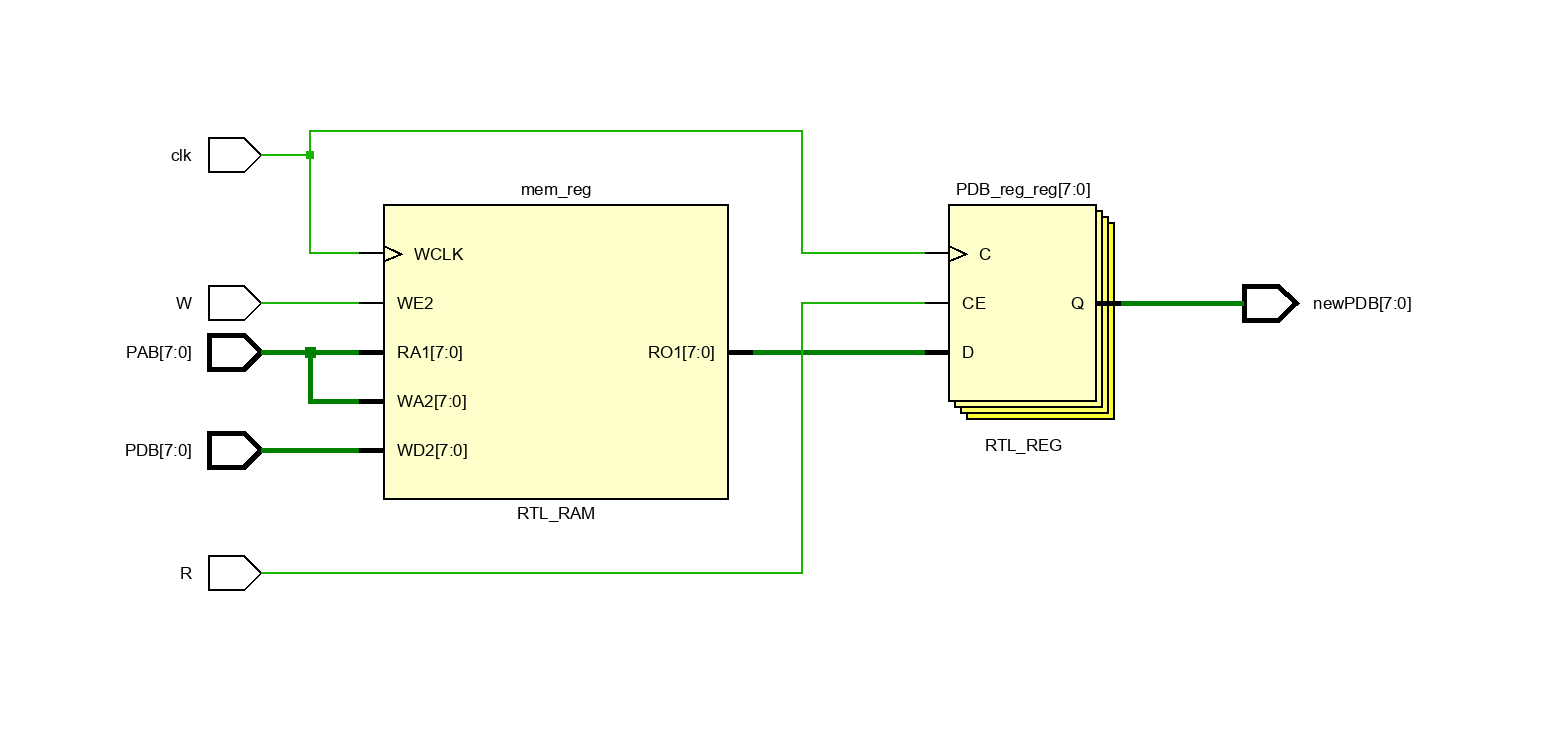


图4-7: PORT设计图

PORT功能: 为方便模拟，利用一个内存去模拟外部端口，根据W和R的使能去判断读写状态并更新数据。

## CPU

CPU设计图如下:

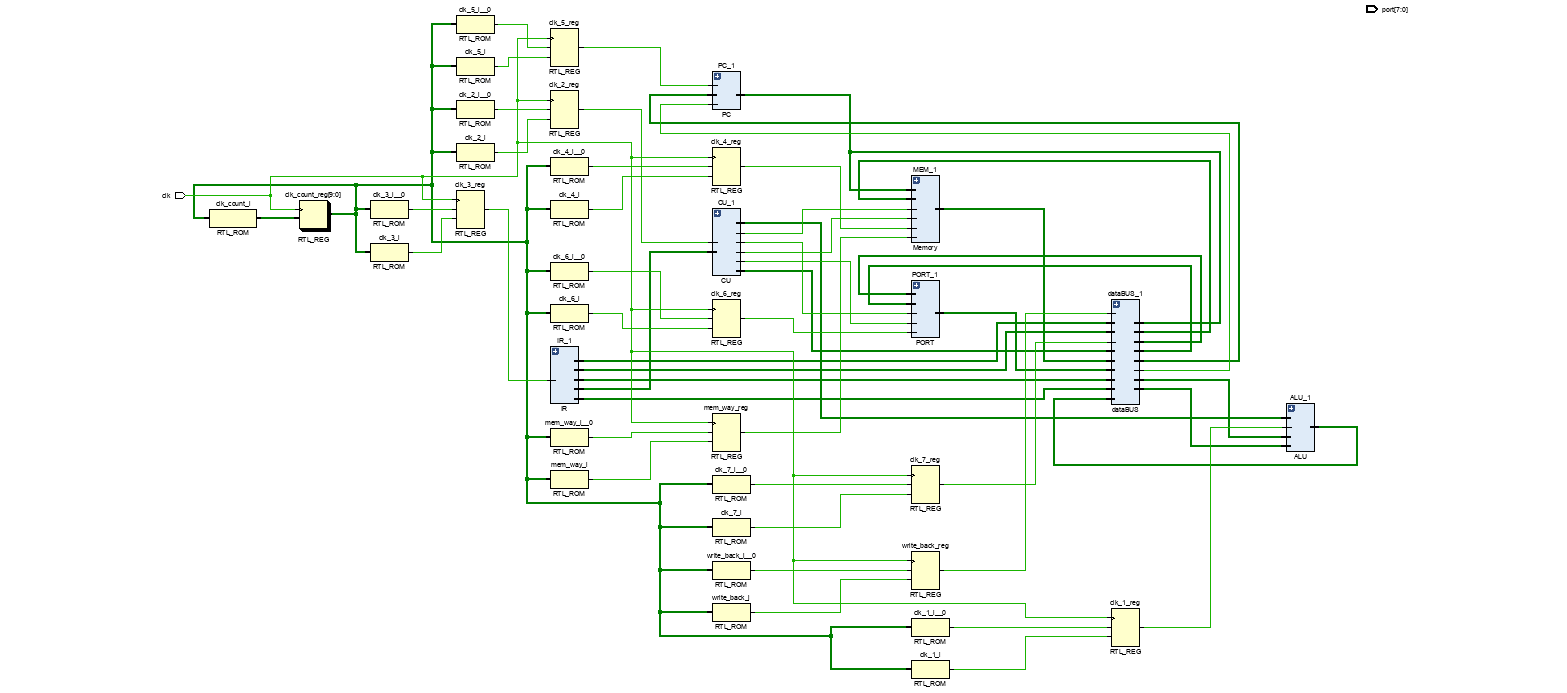


图4-8: CPU设计图

CPU功能:

在仿真过程中，CPU会输出port端口中0000\_0100端口的值，用来检验仿真程序是否正确执行。

1. 操作时间表和控制信号逻辑图

## 各指令操作时间表



表3: 操作时间表

## 控制信号的逻辑框图

### 逻辑表达式表



表4: 逻辑表达式表

### 逻辑图

PC → MAR:

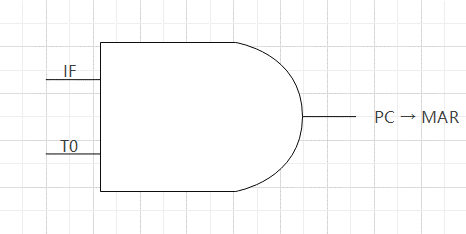


图5-1 PC → MAR逻辑图

M(MAR) → MDR:

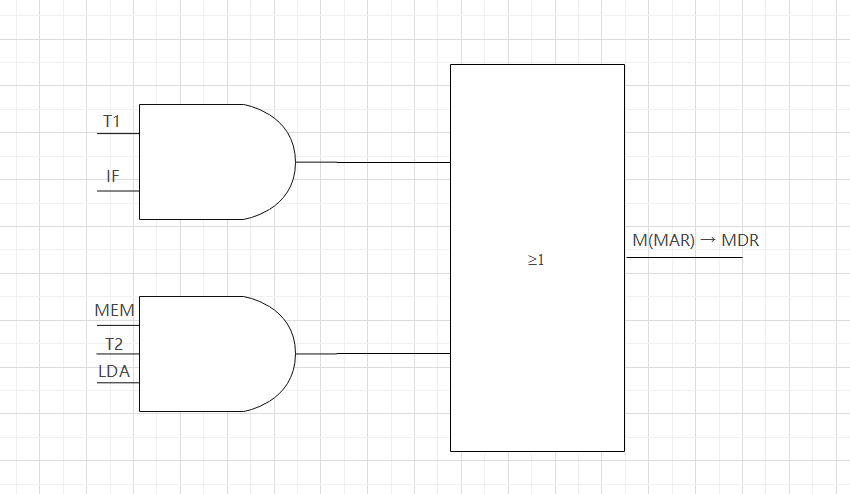


图5-2 M(MAR) → MDR逻辑图

MDR → IR:

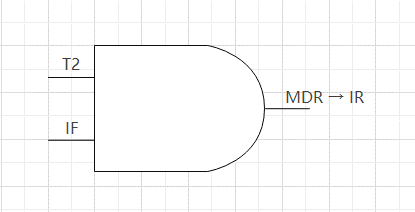


图5-3 MDR → IR逻辑图

OP(IR) → CU:

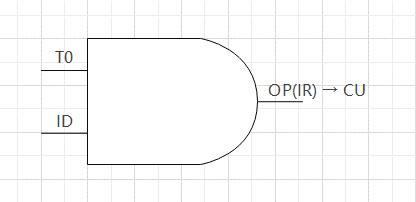


图5-4 OP(IR) → CU逻辑图

R(Ad1(IR)) → valA:

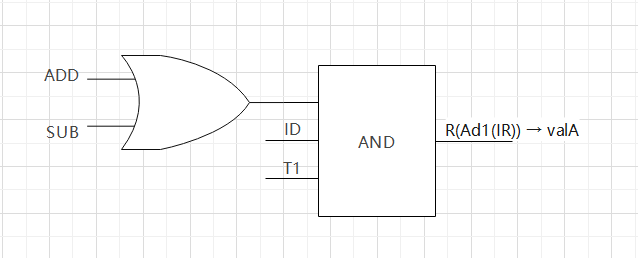


图5-5 R(Ad1(IR)) → valA逻辑图

R(Ad2(IR)) → valB:

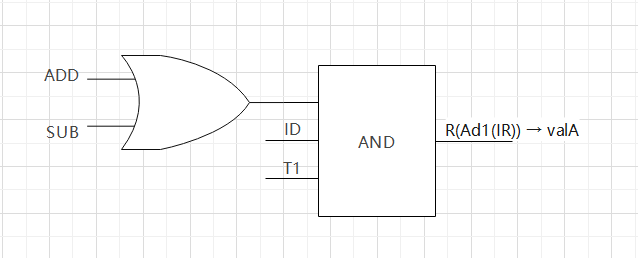


图5-6 R(Ad2(IR)) → valB逻辑图

R(Ad2(IR)) → valA:

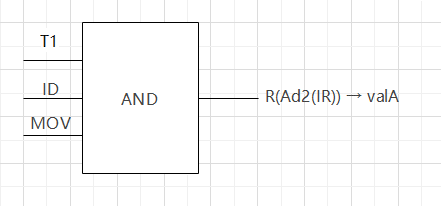


图5-7 R(Ad2(IR)) → valA逻辑图

0 → valB:

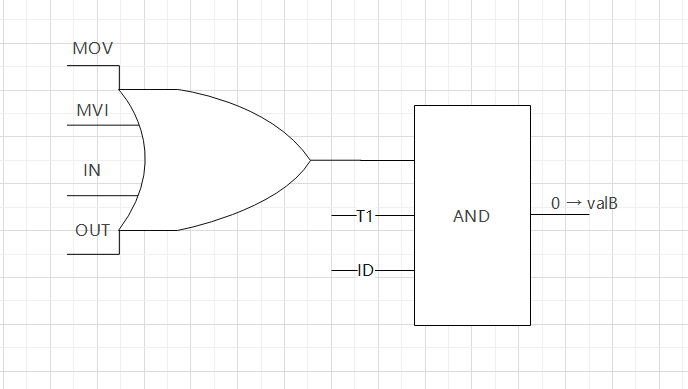


图5-8 0 → valB逻辑图

X → valA:

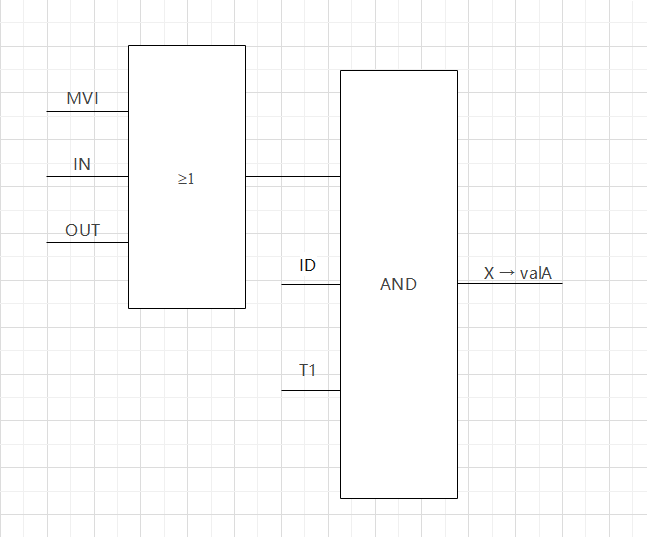


图5-9 X → valA逻辑图

X → valB:

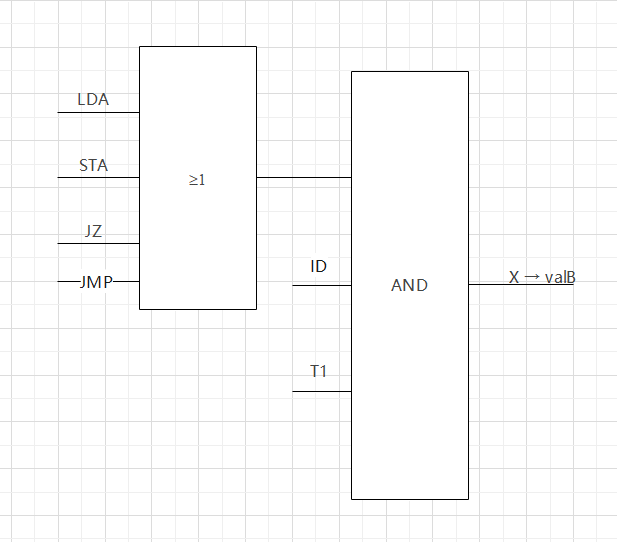


图5-10 X → valB逻辑图

R7 → valA:

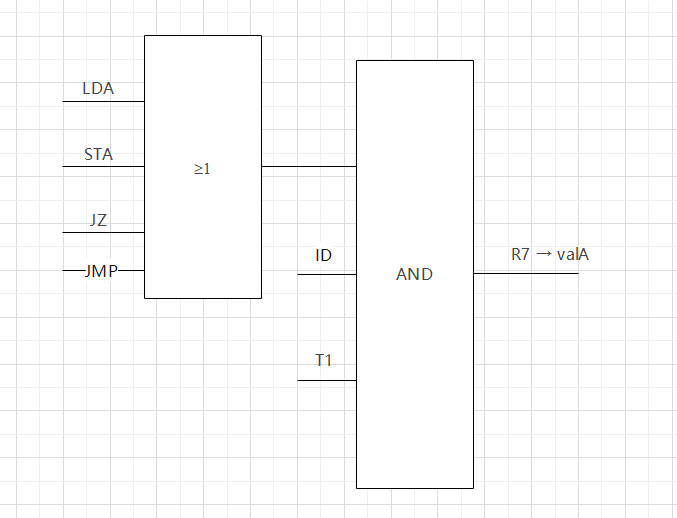


图5-11 R7 → valA逻辑图

valA + valB → valR:

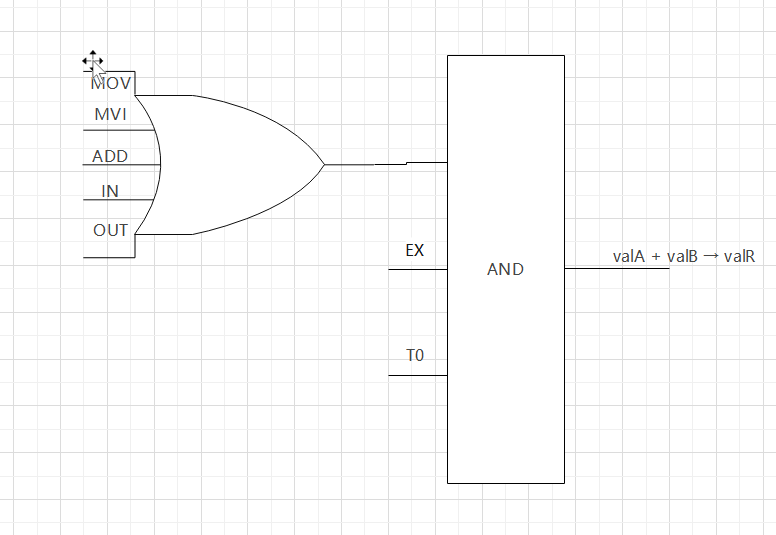


图5-12 valA + valB → valR逻辑图

valA - valB → valR:

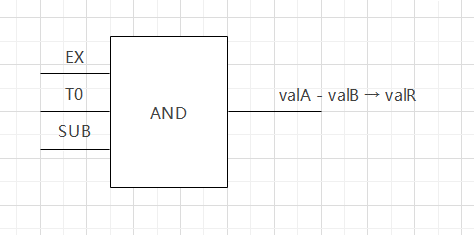


图5-13 valA - valB → valR逻辑图

valA//valB → valR:

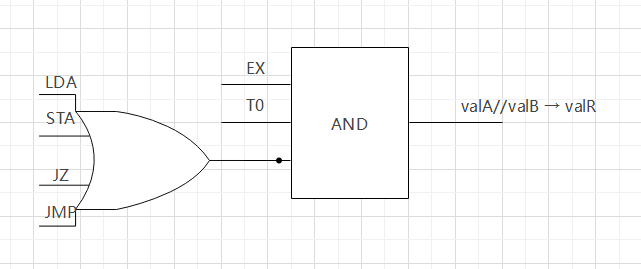


图5-14 valA // valB → valR逻辑图

valR → MAR:

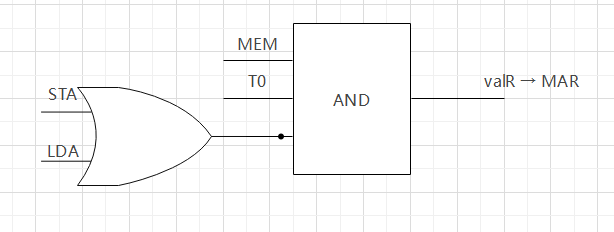


图5-15 valR → MAR逻辑图

valR → PAB:

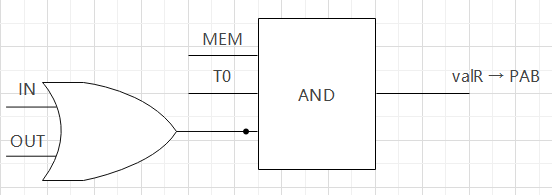


图5-16 valR → PAB逻辑图

R(Ad(IR)) → MDR:

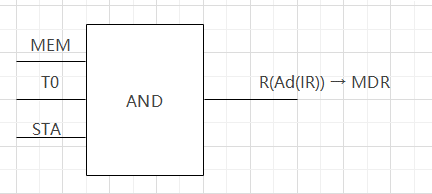


图5-17 R(Ad(IR)) → MDR逻辑图

R(Ad(IR)) → PDB:

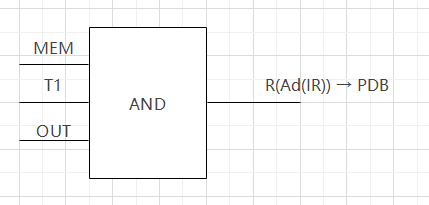


图5-18 R(Ad(IR)) → PDB逻辑图

M(PAB) → PDB:

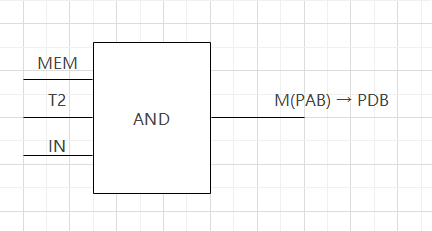


图5-19 M(PAB) → PDB逻辑图

MDR → M(MAR):

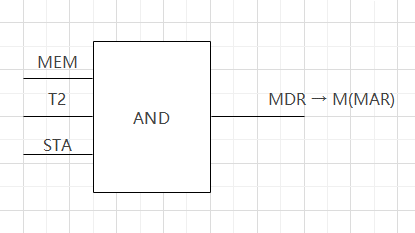


图5-20 MDR → M(MAR)逻辑图

PDB → M(PAB):

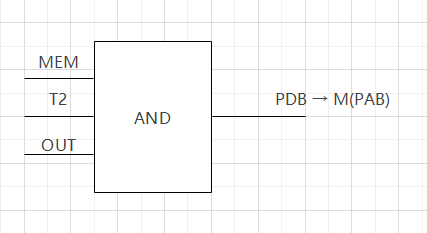


图5-21 PDB → M(PAB)逻辑图

PC + 1 → PC:

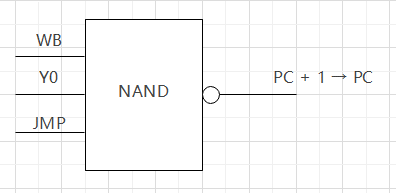


图5-22 PC + 1 → PC逻辑图

valR → R(Ad1(IR)):

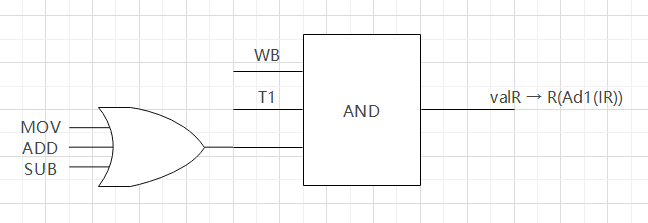


图5-23 valR → R(Ad1(IR))逻辑图

MDR → R(Ad(IR)):

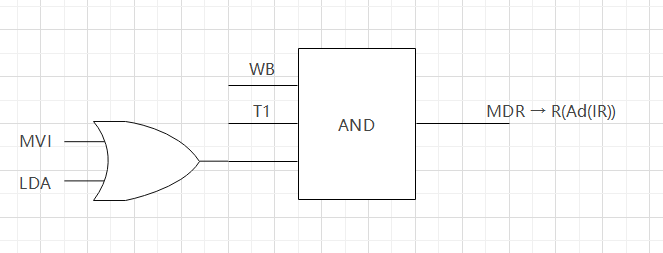


图5-24 MDR → R(Ad(IR))逻辑图

PDB → R(Ad(IR)):

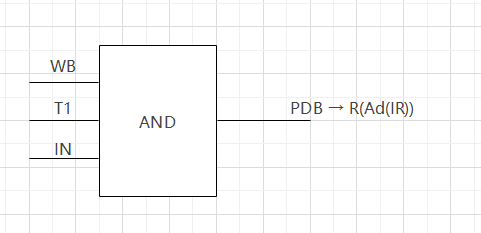


图5-25 PDB → R(Ad(IR))逻辑图

valR → PC:

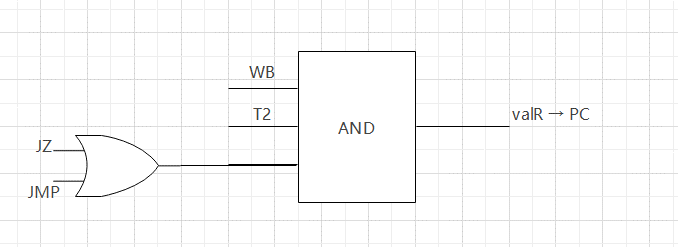


图5-26 valR → PC逻辑图