“计算机设计与实践”处理器实验设计报告

姓名：杜晨鹏

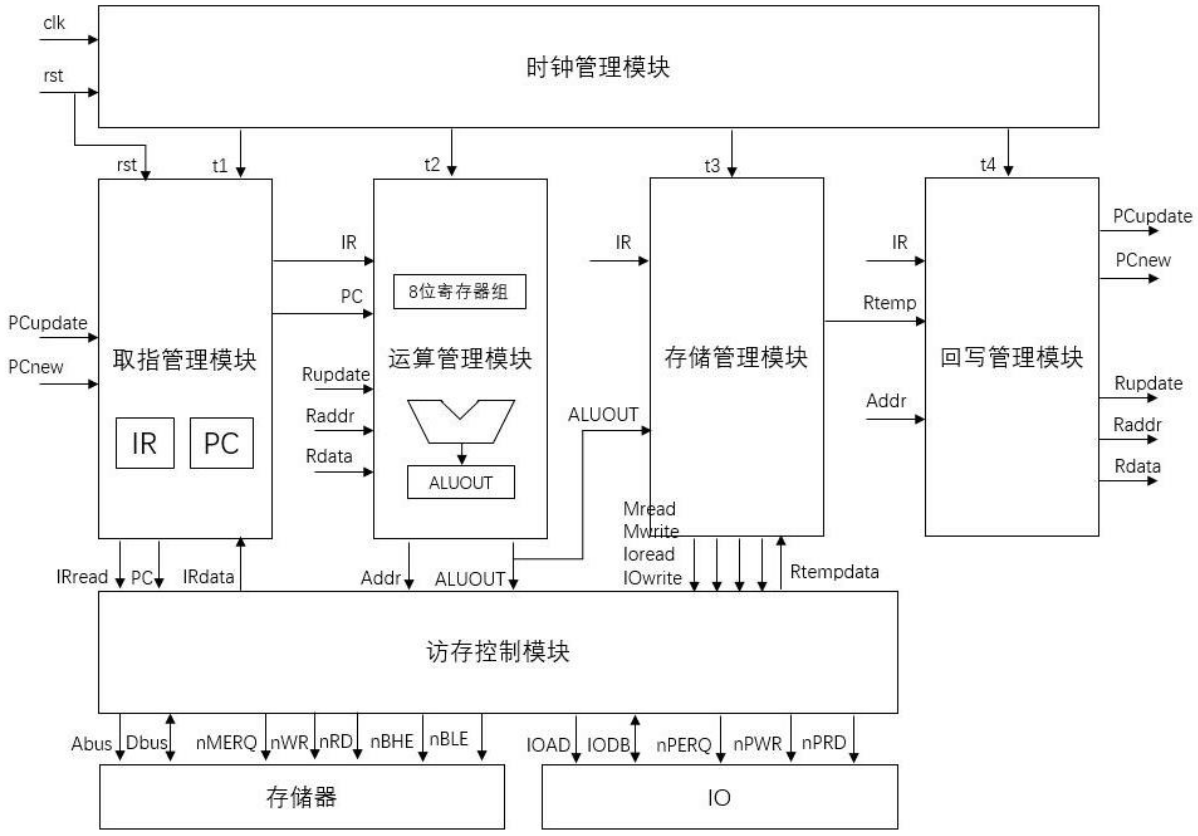
班级：1403101

学号：1141310310

哈尔滨工业大学计算机学院

2016 年 7 月

一、 详细设计整体框图；



二、 各模块接口的详细说明；

①时钟管理模块。

clk：时钟

rst：节拍重置

t1—t4：四个机器周期的节拍

②取指管理模块。

enable：取指周期使能

rst：PC 归 1

PC：欲取指令的地址

IRread：读指令信号

IRdata：读到的 16 位指令

IR：当前指令

PCupdate：回写周期中更新 PC 的信号

PCnew：更新 PC 时欲写入的地址

③运算管理模块

enable：运算周期使能

PC：当前指令的地址

IR：当前指令

ALUOUT：运算器的计算结果

Addr：将 8 位的[R7]和指令 IR 的低 8 位拼接，得到 16 位地址

Rupdate：回写周期中更新寄存器的信号

Raddr：更新寄存器时欲写入的地址

Rdata：更新寄存器的数据

④存储管理模块

enable：运算周期使能

IR：当前指令

ALUOUT：运算器的计算结果

Rtempdata：从存储器或 IO 中读到的 8 位数据

Rtemp：欲写回寄存器的数据

Mwrite：写存储器数据

Mread：读存储器数据

IOwrite：写 IO 数据

IOread：读 IO 数据

⑤访存控制模块

IRread：读指令信号

PC：欲取指令的地址

IRdata：读出的 16 位指令

Addr：读写存储器数据的地址

ALUOUT：写入存储器的数据

Mwrite：写存储器数据

Mread：读存储器数据

IOwrite：写 IO 数据

IOread：读 IO 数据

Rtempdata：从存储器或 IO 中读到的 8 位数据

Abus：访存地址总线

Dbus：访存数据总线

nMREQ：存储器使能

nRD：读存储器

nWR：写存储器

nBHE：访存高 8 位有效

nBLE：访存低 8 位有效

nPREQ：IO 使能

IOAD：IO 地址总线

IODB：IO 数据总线

IODB_RD：从 4 个 8 位开关读入共 32 位数据

nPRD：读 IO 数据

nPWR：写 IO 数据

⑥回写管理模块

enable：运算周期使能

IR：当前指令

Addr：欲回写 PC 的地址

Rtemp：欲回写寄存器的数据

Rupdate：更新寄存器的信号

Rnew：更新寄存器时欲写入的地址

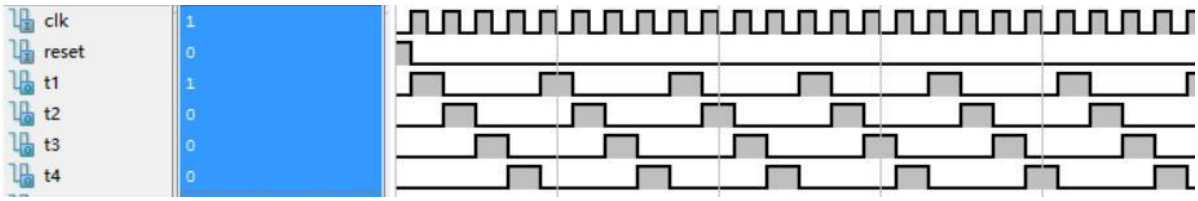
Rupdate：更新寄存器的信号

Raddr：更新寄存器时欲写入的地址

Rdata：更新寄存器的数据

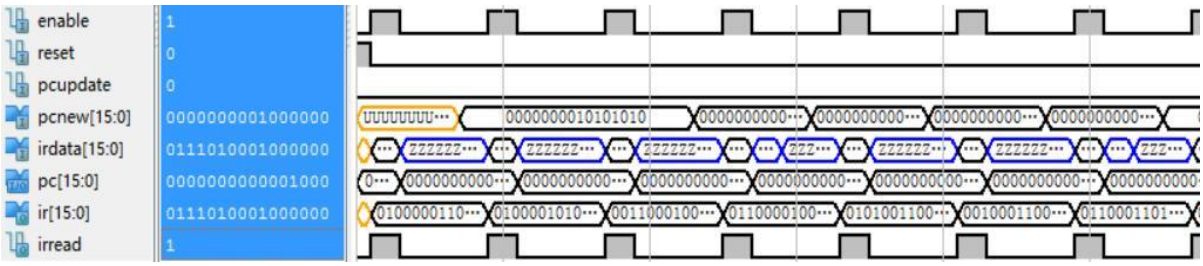
三、 系统仿真波形；

①时钟管理模块



给入 reset 信号后，节拍重置。之后根据时钟信号，循环产生 4 个节拍。

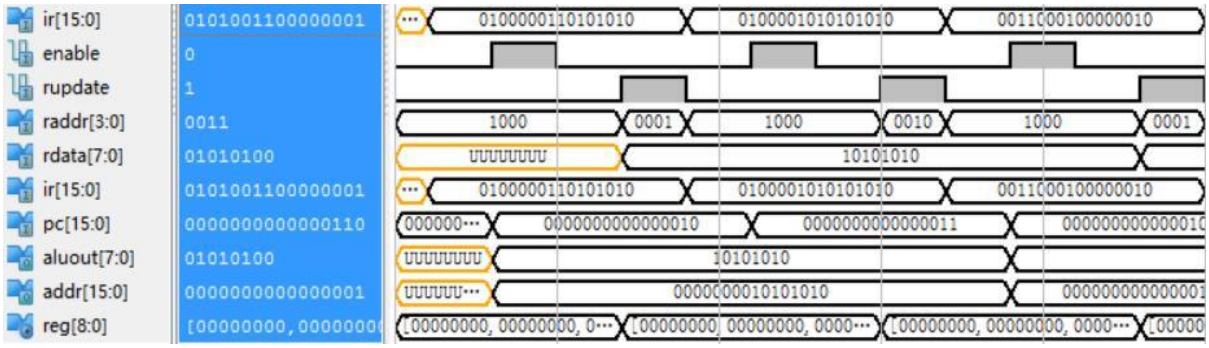
②取指管理模块



给入 reset 信号后，PC 归 1。之后每当进入取指周期时，IRread 置 1 进行取指操作，取完的指令存入 IR。另外，在取指周期结束时，PC 加 1。

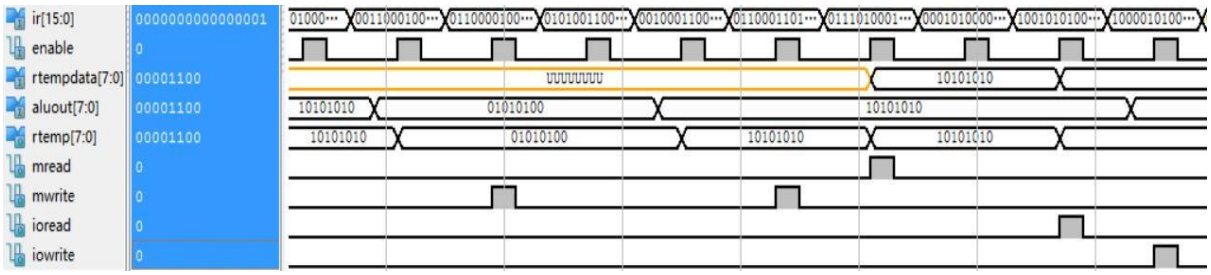
当回写模块中 PCupdate 为 1 时，PC 置为 PCnew 的地址。

③运算管理模块



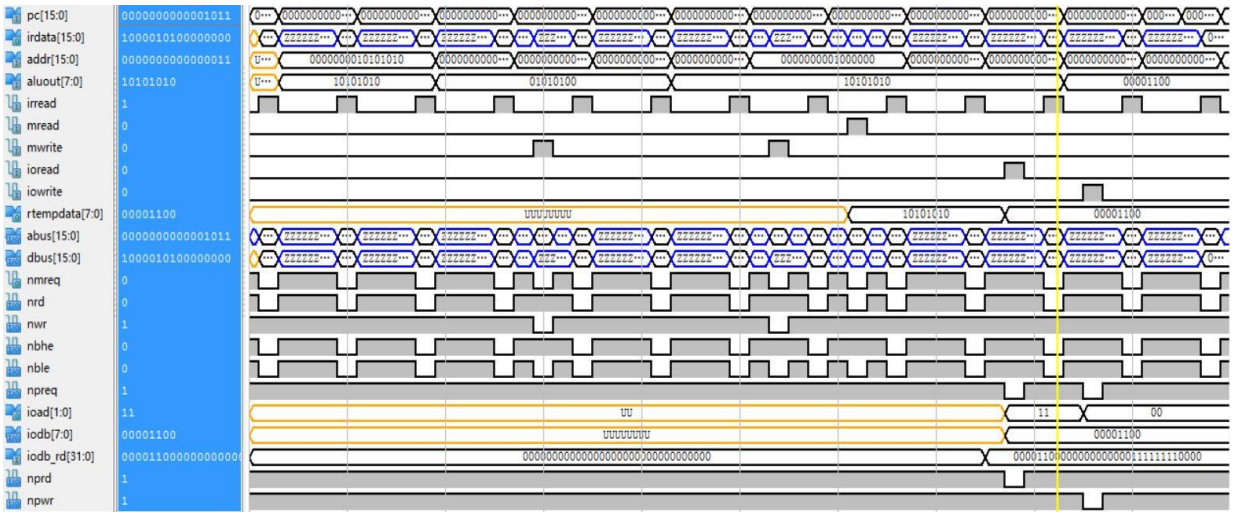
enable 为 1 时表示进入运算周期，进行运算。ALUOUT 为运算器计算结果或要回写的数据，Addr 为将 8 位的[R7]和指令 IR 的低 8 位拼接得到的 16 位地址。之后在回写周期中 Rupdate 置 1，根据 Raddr 的地址将 Rdata 写入相应的寄存器。8 个 8 位寄存器由二维数组信号组成。

④存储管理模块



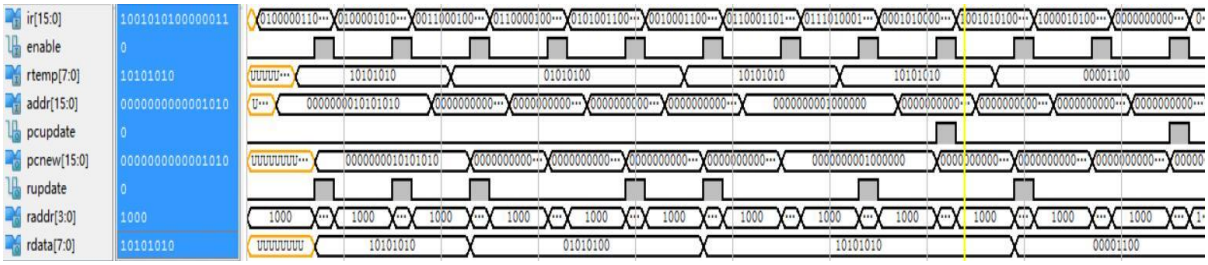
根据不同的指令，在访存周期判断进行读写存储器或 IO 的操作，进而给出相应的操作命令。另外，根据不同的指令，还决定将 ALUOUT 还是 Rtempdata 作为 Rtemp 输出，等待进入回写周期进行回写。

⑤访存控制模块



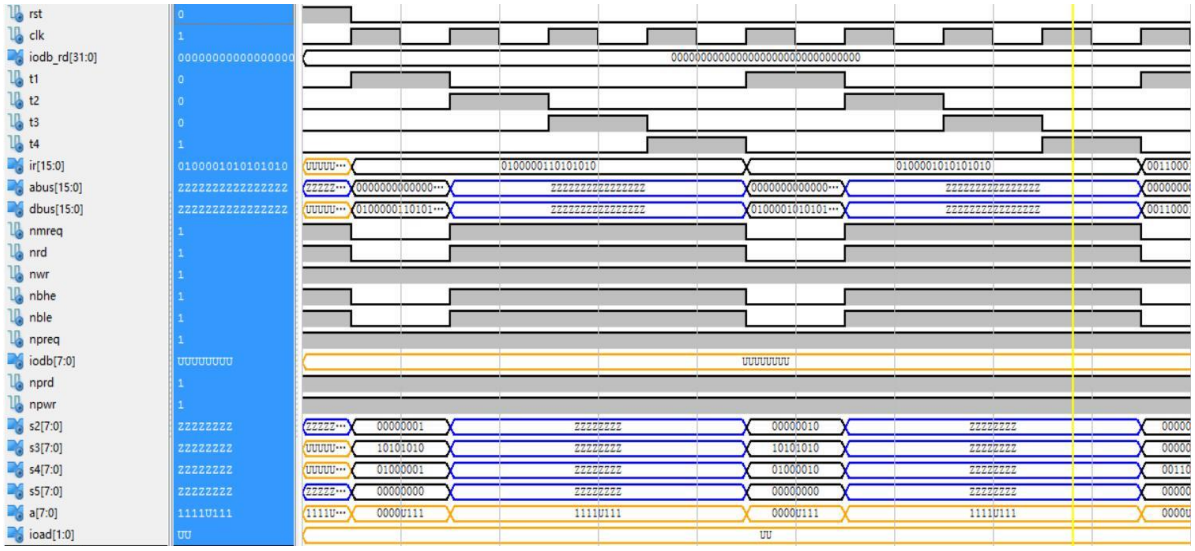
每个取指周期，IRread 置 1，进行读指令，地址由 PC 给出。根据指令的不同，在访存周期进行对存储器和 IO 的读写操作，根据存储管理模块给出的信号，对访存和 IO 信号进行控制。写数据时，地址由 ALUOUT 给出，地址则由 Addr 给出。读数据时，读到的数据存到 Rtempdata，送入回写管理模块。4 个 8 位开关数据由 IODB_RD 读入，然后根据 IOAD 取有效的 8 位送入 IODB。

⑥回写管理模块

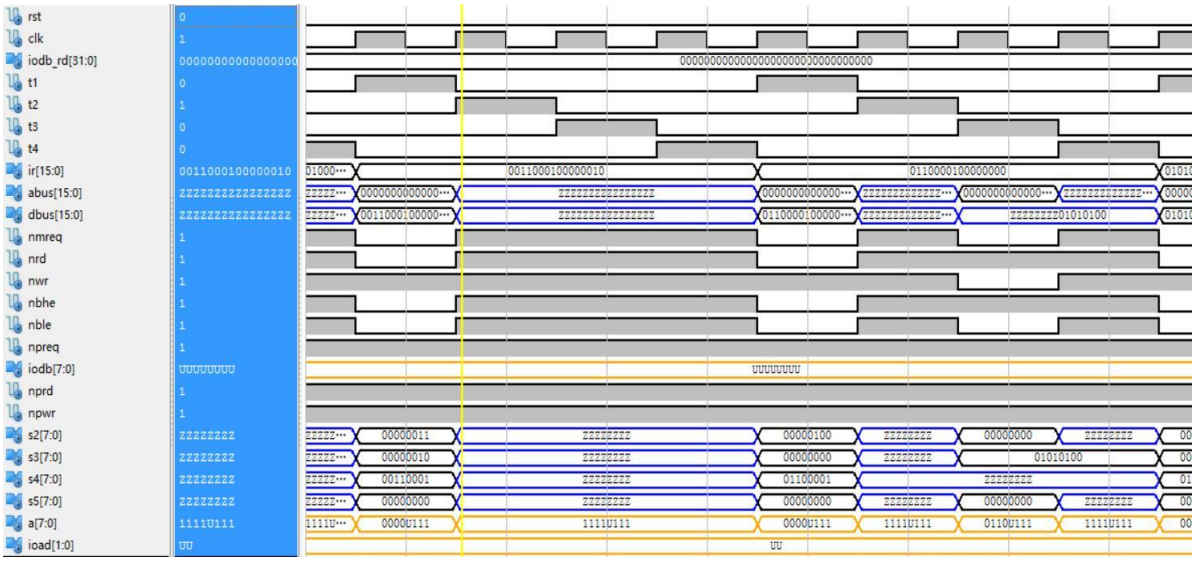


根据指令，对 PC 或寄存器进行更新。PC 地址来自 Addr，寄存器数据来自 Rtemp，寄存器地址来自指令第 10 到第 8 位。

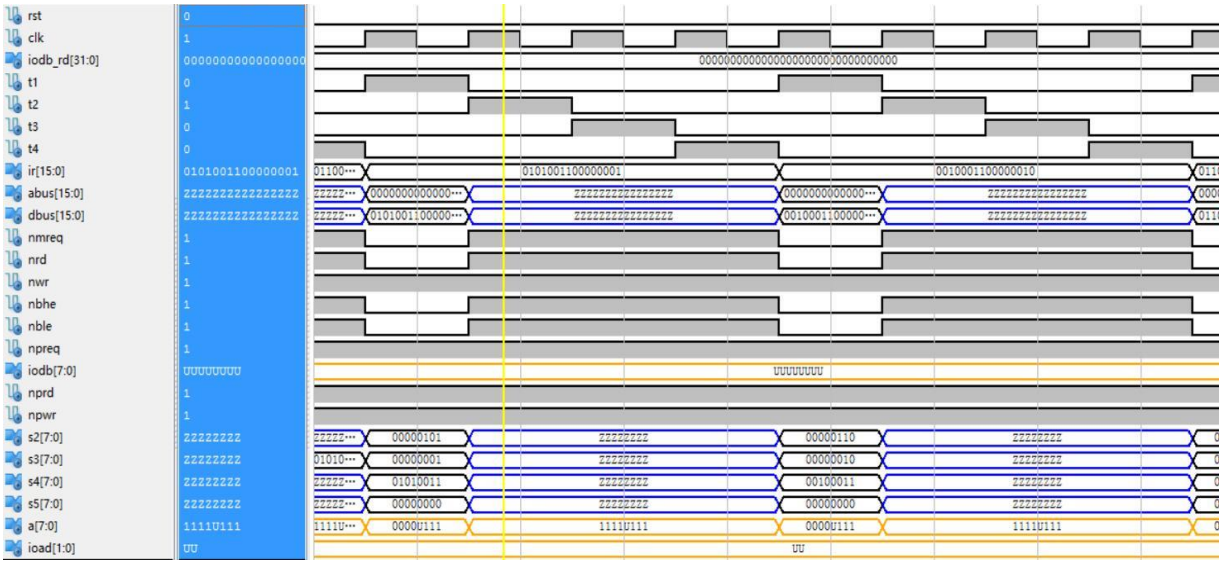
⑦CPU



这段波形测试 MVI R1,AA 和 MVI R2,AA 两条指令。S2-S5 对应数码管显示的内容，A 位 LED 灯，他们分别根据实验要求进行显示。数据总线和地址总线内容正确。PC 在取指周期结束后加 1.

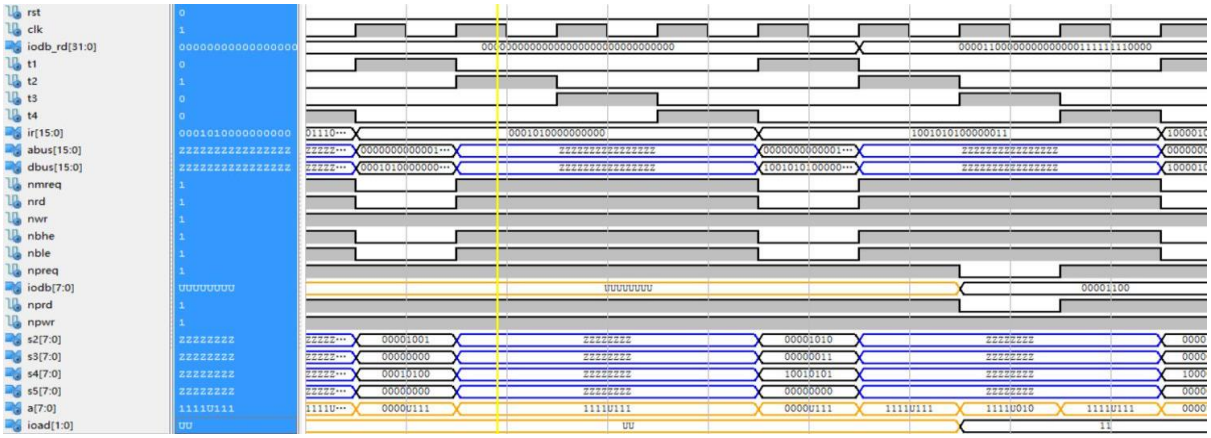


这段波形测试 ADD R1,R2 和 STA R1,00 两条指令。ADD 操作中，寄存器正确相加进行回写，得到结果 54。之后在 STA 操作中，可以在访存周期的数据总线低 8 位中看到计算结果“01010100”。

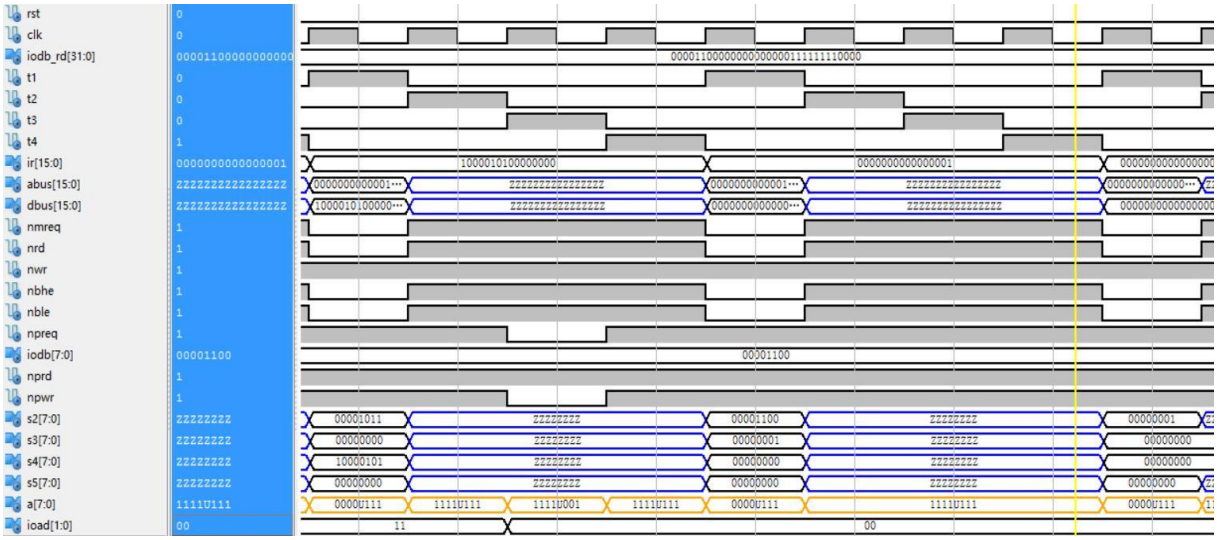


这段波形测试 MOV R3,R1 和 SUB R3,R2 两条指令。这两天指令没有访存过程，在回写周期中已经能对寄存器进行了回写。计算结果将在后续执行 STA 指令时看到。

这段波形测试 STA R3,40 和 LDA R4,40 两条指令。54-AA 的结果还是 AA，可以在执行 STA 的访存周期时的数据总线低 8 位中看到计算结果。之后再将刚才存入的数据 AA 送入 R4。



这段波形测试 JZ R4,00 和 IN R5,3 两条指令。因为刚才将存储器中的 AA 读入了 R4，故 R4 不为 0，不进行跳转。另外，从 IO 的 4 个 8 位开关读入了 32 位数据，IOAD 位 11，即将 K3 的数据读入 IODB，得到“00001100”，该数据存入 R5。



这段波形测试 OUT R5,0 和 JMP 01 两条指令。在执行 OUT 的访存周期时，刚才读入 R5 的数据“00001100”送入 IODB，可在 LED 灯上进行显示。执行 JMP 指令后，下一个指令取指周期的 Abus 上可以看到，地址已经跳回“0000000000000001”。

四、 系统管脚定义的 UCF 文件；

```
NET "S2<0>" LOC = "P31";
NET "S2<1>" LOC = "P33";
NET "S2<2>" LOC = "P34";
NET "S2<3>" LOC = "P35";
NET "S2<4>" LOC = "P36";
NET "S2<5>" LOC = "P39";
NET "S2<6>" LOC = "P40";
NET "S2<7>" LOC = "P41";
NET "S5<0>" LOC = "P126";
NET "S5<1>" LOC = "P127";
NET "S5<2>" LOC = "P129";
NET "S5<3>" LOC = "P202";
NET "S5<4>" LOC = "P203";
NET "S5<5>" LOC = "P205";
NET "S5<6>" LOC = "P206";
NET "S5<7>" LOC = "P103";
NET "S3<0>" LOC = "P42";
NET "S3<1>" LOC = "P45";
NET "S3<2>" LOC = "P47";
NET "S3<3>" LOC = "P48";
NET "S3<4>" LOC = "P49";
```

```
NET "S3<5>" LOC = "P50";
NET "S3<6>" LOC = "P55";
NET "S3<7>" LOC = "P56";
NET "S4<0>" LOC = "P60";
NET "S4<1>" LOC = "P61";
NET "S4<2>" LOC = "P62";
NET "S4<3>" LOC = "P63";
NET "S4<4>" LOC = "P2";
NET "S4<5>" LOC = "P108";
NET "S4<6>" LOC = "P109";
NET "S4<7>" LOC = "P112";
NET "IR<0>" LOC = "P4";
NET "IR<1>" LOC = "P5";
NET "IR<2>" LOC = "P8";
NET "IR<3>" LOC = "P9";
NET "IR<4>" LOC = "P11";
NET "IR<5>" LOC = "P12";
NET "IR<6>" LOC = "P15";
NET "IR<7>" LOC = "P16";
NET "IR<8>" LOC = "P18";
NET "IR<9>" LOC = "P19";
NET "IR<10>" LOC = "P22";
NET "IR<11>" LOC = "P23";
NET "IR<12>" LOC = "P24";
NET "IR<13>" LOC = "P25";
NET "IR<14>" LOC = "P28";
NET "IR<15>" LOC = "P29";
NET "t1" LOC = "P196";
NET "t2" LOC = "P193";
NET "t3" LOC = "P192";
NET "t4" LOC = "P190";

NET "A<4>" LOC = "P185";
NET "A<5>" LOC = "P77";
NET "A<6>" LOC = "P82";
NET "A<7>" LOC = "P83";
NET "A<2>" LOC = "P99";
NET "A<1>" LOC = "P100";
NET "A<0>" LOC = "P102";

NET "IODB<0>" LOC = "P98";
NET "IODB<1>" LOC = "P3";
NET "IODB<2>" LOC = "P200";
NET "IODB<3>" LOC = "P199";
```

```
NET "IODB<4>" LOC = "P197";
NET "IODB<5>" LOC = "P186";
NET "IODB<6>" LOC = "P187";
NET "IODB<7>" LOC = "P189";

NET "clk" LOC = "P75";
NET "rst" LOC = "P51";
NET "Dbus<0>" LOC = "P167";
NET "Dbus<1>" LOC = "P165";
NET "Dbus<2>" LOC = "P164";
NET "Dbus<3>" LOC = "P163";
NET "Dbus<4>" LOC = "P162";
NET "Dbus<5>" LOC = "P161";
NET "Dbus<6>" LOC = "P160";
NET "Dbus<7>" LOC = "P153";
NET "Dbus<8>" LOC = "P120";
NET "Dbus<9>" LOC = "P122";
NET "Dbus<10>" LOC = "P123";
NET "Dbus<11>" LOC = "P128";
NET "Dbus<12>" LOC = "P132";
NET "Dbus<13>" LOC = "P133";
NET "Dbus<14>" LOC = "P134";
NET "Dbus<15>" LOC = "P135";
NET "Abus<0>" LOC = "P179";
NET "Abus<1>" LOC = "P178";
NET "Abus<2>" LOC = "P177";
NET "Abus<3>" LOC = "P172";
NET "Abus<4>" LOC = "P171";
NET "Abus<5>" LOC = "P151";
NET "Abus<6>" LOC = "P150";
NET "Abus<7>" LOC = "P147";
NET "Abus<8>" LOC = "P146";
NET "Abus<9>" LOC = "P113";
NET "Abus<10>" LOC = "P115";
NET "Abus<11>" LOC = "P116";
NET "Abus<12>" LOC = "P119";
NET "Abus<13>" LOC = "P140";
NET "Abus<14>" LOC = "P144";
NET "Abus<15>" LOC = "P145";
NET "nBLE" LOC = "P137";
NET "nBHE" LOC = "P138";
NET "nRD" LOC = "P139";
NET "nWR" LOC = "P152";
NET "nMREQ" LOC = "P168";
```

```
NET "IODB_RD<0>" LOC = "P154";
NET "IODB_RD<1>" LOC = "P148";
NET "IODB_RD<2>" LOC = "P142";
NET "IODB_RD<3>" LOC = "P136";
NET "IODB_RD<4>" LOC = "P130";
NET "IODB_RD<5>" LOC = "P124";
NET "IODB_RD<6>" LOC = "P118";
NET "IODB_RD<7>" LOC = "P110";
NET "IODB_RD<8>" LOC = "P174";
NET "IODB_RD<9>" LOC = "P204";
NET "IODB_RD<10>" LOC = "P194";
NET "IODB_RD<11>" LOC = "P175";
NET "IODB_RD<12>" LOC = "P169";
NET "IODB_RD<13>" LOC = "P101";
NET "IODB_RD<14>" LOC = "P97";
NET "IODB_RD<15>" LOC = "P96";
NET "IODB_RD<16>" LOC = "P94";
NET "IODB_RD<17>" LOC = "P93";
NET "IODB_RD<18>" LOC = "P91";
NET "IODB_RD<19>" LOC = "P90";
NET "IODB_RD<20>" LOC = "P89";
NET "IODB_RD<21>" LOC = "P87";
NET "IODB_RD<22>" LOC = "P80";
NET "IODB_RD<23>" LOC = "P78";
NET "IODB_RD<24>" LOC = "P72";
NET "IODB_RD<25>" LOC = "P71";
NET "IODB_RD<26>" LOC = "P69";
NET "IODB_RD<27>" LOC = "P68";
NET "IODB_RD<28>" LOC = "P65";
NET "IODB_RD<29>" LOC = "P64";
NET "IODB_RD<30>" LOC = "P58";
NET "IODB_RD<31>" LOC = "P57";
```

五、 处理器功能测试程序，包括助记符和二进制代码。

```
MVI R1,AA    0100000110101010
MVI R2,AA    0100001010101010
ADD R1,R2    0011000100000010
STA R1,00    0110000100000000
MOV R3,R1    0101001100000001
SUB R3,R2    0010001100000010
STA R3,40    0110001101000000
LDA R4,40    0111010001000000
JZ R4,00     0001010000000000
IN R5,3      1001010100000011
```

OUT R5,0　1000010100000000

JMP 01　　0000000000000001

六、 设计、调试、波形、下载过程中遇到的问题及解决方法；

①设计问题。

实验箱上没有可进行 IO 寻址的部件，所以不能像访问存储器一样访问 IO。解决方法为，将 32 位开关全部读入，之后再根据 IOAD 取其中有效的 8 位送入 IODB。

②调试问题。

取指周期中，如果在给出 IRread 信号的同时将 IRdata 送入 IR 的话，IR 数据错误。原因是访存有微小的延时。故将 IRdata 作为敏感信号，单独写一个 process，即可保证 IR 数据正确。

③波形问题

在波形仿真中，IR 始终无法正确读入，原因是与真实的 FPGA 不同，仿真时 Dbus 信号由波形文件手动给出，故它不收节拍敏感信号的控制。在波形文件中确保 Dbus 在节拍发生后给出，即可解决。

④下载问题

在 FPGA 板上操作时，读取指令时高 8 位始终无法正确读出。解决方法是在代码中给 Dbus 赋为高阻。

七、 实验体会

在这个设计与实践的过程中，我对 CPU 的工作原理和流程有了更深刻的认识。在不断的代码调试过程中，我更加熟练的掌握了 VHDL 语言设计和 ISE 软件的使用，开发效率有效提高。在实验箱上调试时，我认识到了 inout 信号的操作规律，也对 FPGA 板的设计和工作有了更多的认识和了解。