**“计算机设计与实践”处理器实验设计报告**

姓名：肖华冬

班级：1403106

学号：1144420112

哈尔滨工业大学计算机学院

2016年7月

1. 详细设计整体框图

clk

**时钟模块**

reset

运算模块

T0

回写模块

T1 T2 T3

访存控制

存储管理

取指模块

ir ctrl ctrl

pc

aluout temp

pcaddr

pc rdata wr

Ir pcnew data

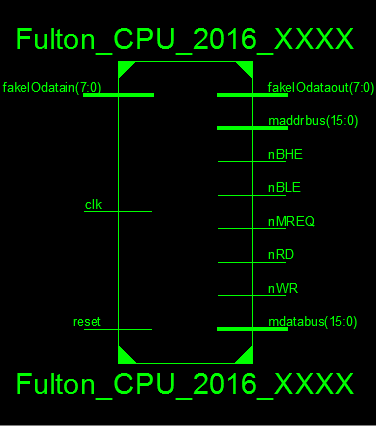
rd

IO

nWR nRD nBLE nBHE aBus nMREQ dBus

存储器

1. 各模块接口的详细说明
2. cpu与外部接口

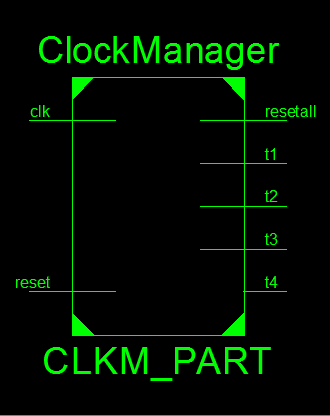


接口信号

|  |  |
| --- | --- |
| clk | 时钟信号 |
| reset | 复位信号 |
| nMREQ | 取指节拍信号 |
| nRD | 运算节拍信号 |
| nBHE | 存储管理节拍信号 |
| nBLE | 回写节拍信号 |
| maddrbus | 单向地址总线 |
| mdatabus | 双向数据总线 |
| fakeIOdatain | 从模拟io来的数据 |
| fakeIOdataout | 送往模拟io的数据 |

总模块与外部存储器和io设备连线，接受时钟信号和开关机复位信号。

1.时钟模块

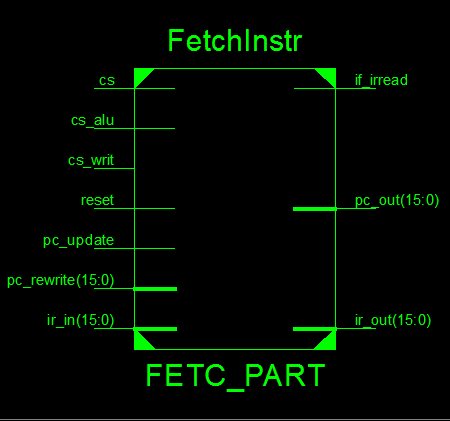


接口信号

|  |  |
| --- | --- |
| clk | 时钟信号 |
| reset | 复位信号 |
| T1 | 取指节拍信号 |
| T2 | 运算节拍信号 |
| T3 | 存储管理节拍信号 |
| T4 | 回写节拍信号 |
| resetall | 控制其他模块进行内部寄存器复位 |

时钟模块由clk产生四个连续节拍t1,t2,t3，t4，分别控制之后四个模块(取指，运算，存储管理，回写)运行。reset是复位信号，输出resetall信号，高有效时，各个模块进行复位操作。

2.取指模块

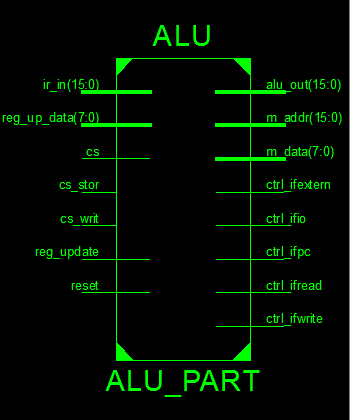


接口信号

|  |  |
| --- | --- |
| ir\_in | 接收访存控制模块得到的新指令 |
| pc\_rewrite | 接受由回写模块回写的pc内容 |
| pc\_update | 回写pc所需的控制信号 |
| cs | 时钟模块产生的节拍，控制取指进行 |
| cs\_alu | pc自增的信号 |
| cs\_writ | 控制回写的时钟信号 |
| reset | 复位信号 |
| ir\_out | 对外输出新的指令 |
| pc\_out | 对外输出pc |
| if\_irread | 读指令的控制信号 |

在t1节拍到来时，取指模块向访存控制模块发出取指令请求信号if\_irread和指令地址pc，并将得到的ir对外输出ir\_out；t2节拍到来时，pc+1。pc\_rewrite是回写模块返回的pc值，pcupdate为其控制信号，并且高有效。

3.运算模块

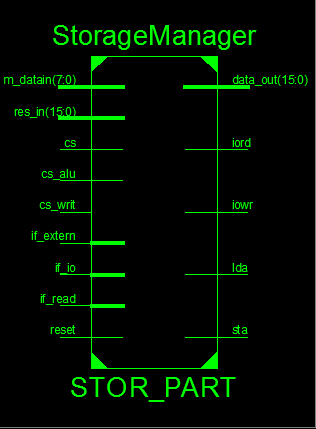


接口信号

|  |  |
| --- | --- |
| ir\_in | 接收由取指模块得到的指令 |
| reg\_up\_data | 接受回写模块回写的数据 |
| reg\_update | 回写数据的控制信号 |
| cs | 运算模块控制节拍 |
| cs\_stor | 接受t3阶段时刻输出数据和地址 |
| cs\_writ | 回写时钟信号 |
| reset | 复位信号 |
| addr | 输出访存地址或新的pc地址 |
| m\_addr | 输出地址 |
| m\_data | 输出数据 |
| ctrl\_ifextern | 是否访问外存 |
| ctrl\_ifio | 是否访问io |
| ctrl\_ifpc | 是否回写pc |
| ctrl\_ifread | 是否读 |
| ctrl\_ifwrite | 是否回写 |
| alu\_out | 输出运算数据 |

在t2节拍，由取指得到的ir进行分析，可得具体操作。若为add,sub,mov.mvi指令，结果由aluout输出到回写模块；若为lda,sta指令，访存时的地址由addr输出，数据由aluout输出；若为jmp,jz指令，新生成的pc由addr输出到回写模块；若为in/out指令，数据由aluout输出。Rdata为回写模块送回到寄存器的数据，rupdate为其控制信号。

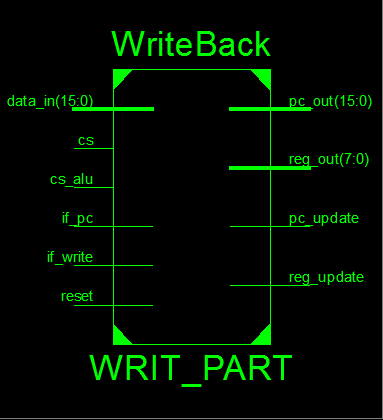
4.存储管理



|  |  |
| --- | --- |
| m\_datain | 接受存储器中读出的数据 |
| res\_in | 来自alu的数据 |
| cs\_alu | 接受alu的数据控制 |
| cs | 控制节拍 |
| cs\_writ | 输出回写模块的信号 |
| reset | 复位信号 |
| sta | 传到访存控制的控制信号 |
| lda | 传到访存控制的读控制信号 |
| iord | io读控制信号 |
| iowr | io写控制信号 |
| if\_extern | 来自alu的信号 |
| if\_io | 来自alu的信号 |
| if\_read | 来自alu的信号 |
| data\_out | 数据输出到回写模块或访存控制模块 |

在t3节拍，data接受数据，根据操作码可知是alu得到的数据还是控制模块的数据，并且接受的数据由rtemp输出。如果是访存指令,lda,sta某个有效，控制访问存储器，iowr,iord某个有效时，控制访问IO.

5.回写模块

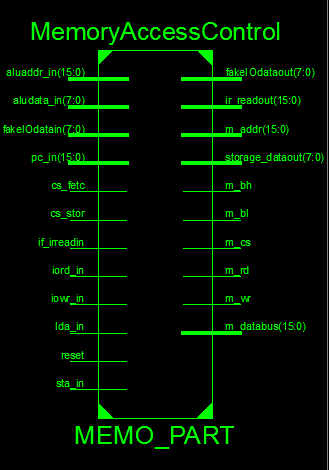


接口信号

|  |  |
| --- | --- |
| data\_in | 接收运算模块非访存指令产生的数据 |
| if\_write/if\_pc | 接收是否回写以及是否回写pc的控制信号 |
| cs | 节拍信号 |
| cs\_alu | 接受控制信号的节拍 |
| reset | 复位信号 |
| pc\_out | 输出回写的pc |
| reg\_out | 输出回写的数据 |
| pc\_update | Pc回写控制信号 |
| reg\_update | 数据回写控制信号 |

在t4节拍，回写模块根据操作码，回写pc或者寄存器。

6.访存控制

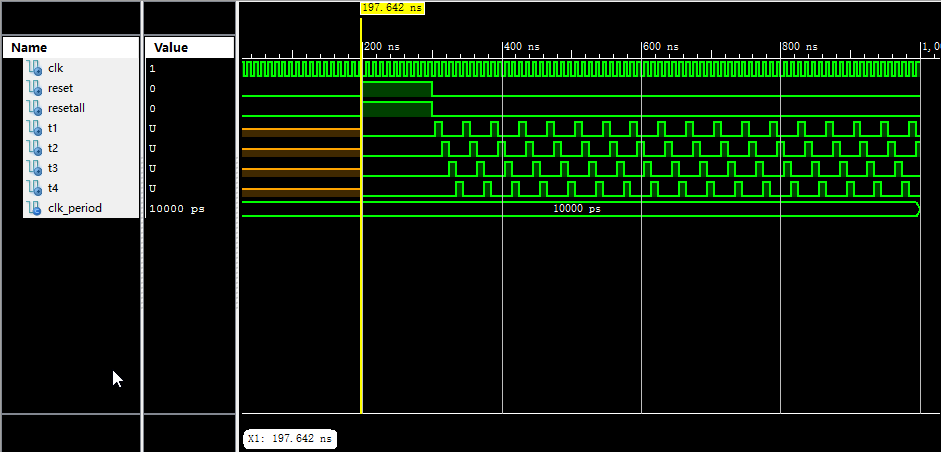


接口信号

|  |  |
| --- | --- |
| aluaddr\_in | 接收运算模块传来的访存地址 |
| aludata\_in | 接收运算模块传来的要写的数据 |
| pc\_in | 接收取指模块传来的pc地址 |
| cs\_fetc/cs\_stor | 分别与取指模块，存储管理模块同步工作信号 |
| if\_irreadin | 取指模块传来的读取指令信号 |
| lda\_in/sta\_in/iowr\_in/iord\_in | 来自存储管理模块的访存或io读写信号 |
| reset | 复位信号 |
| ir\_readout | 读取的指令 |
| m\_addr | 地址总线 |
| storage\_dataout | 输出给存储管理的数据 |
| fakeIOdataout | 写模拟io的数据线 |
| m\_bh | 存储器高位有效 |
| m\_bl | 存储器低位有效 |
| m\_cs | 访存控制信号 |
| m\_rd | 输出访存读信号 |
| m\_wr | 输出访存写信号 |
| m\_databus | 数据总线 |

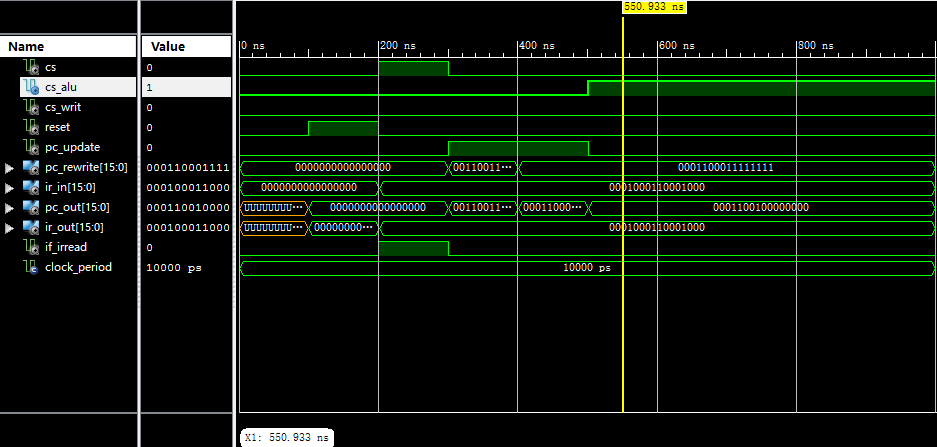
该模块分别与取指模块，存储管理模块同步工作，以便协调整个系统的节拍。输出控制信号到存储器或者io实现外部存储访问。

1. 系统仿真波形
2. 时钟管理模块



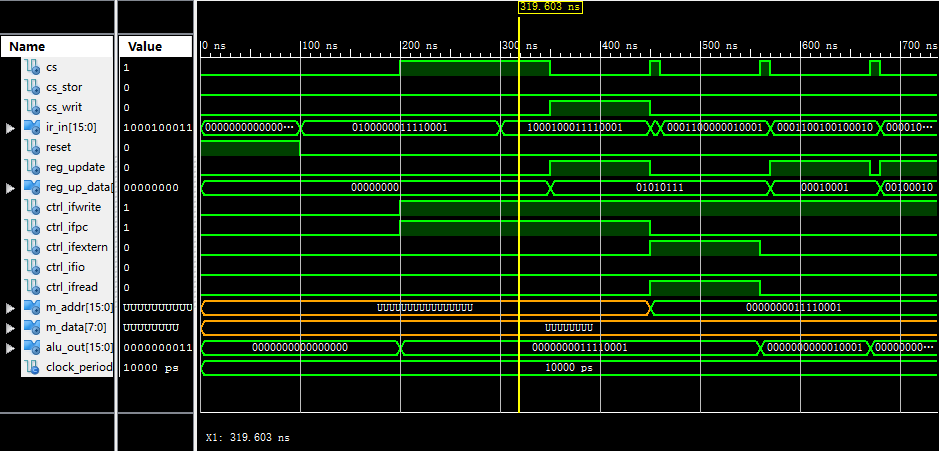
系统必须经过一次reset从而使得时钟正常工作。

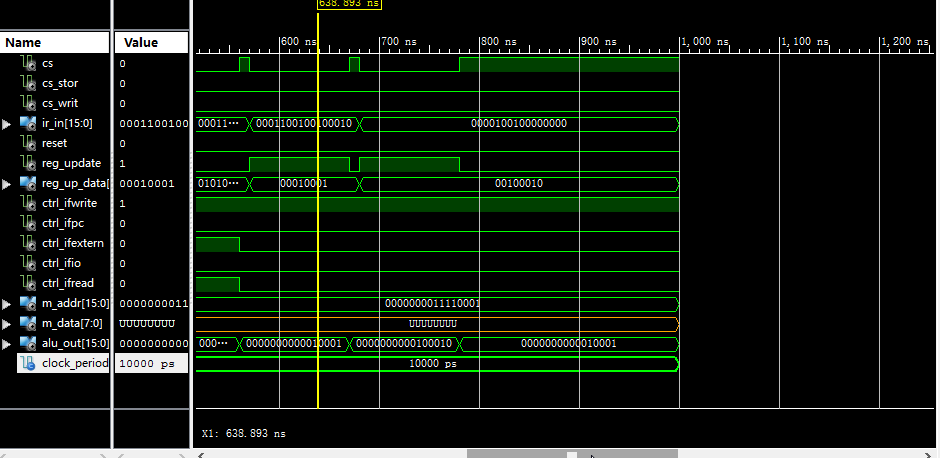
1. 取指令模块



在T1阶段（cs选中），读入ir\_in;T2阶段(cs\_alu选中)，pc完成自增;T4（cs\_writ选中），pc回写。

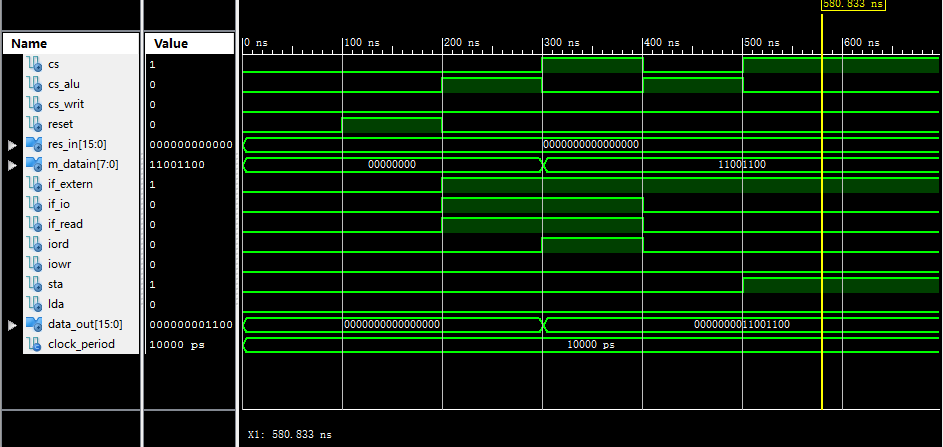
1. 运算模块





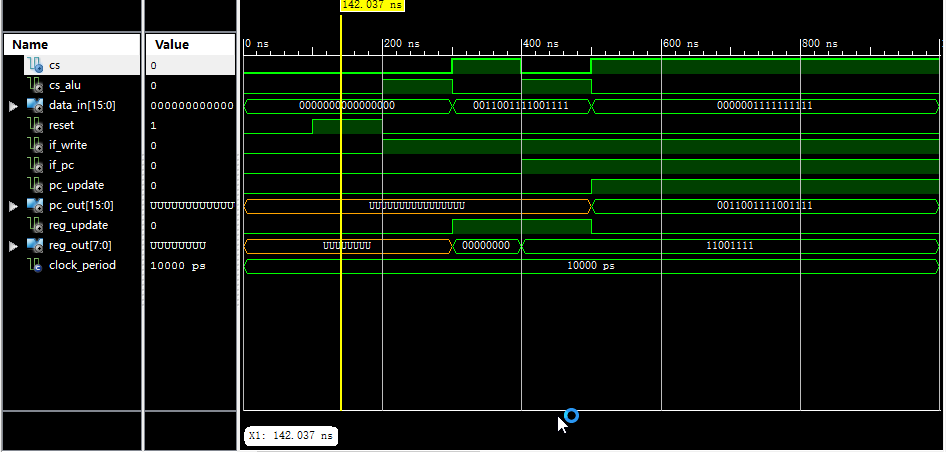
alu每处理一条指令都需要4个时钟周期，lda,mov,mvi,add,sub,in等指令需要回写寄存器，sta,out等需要读写外部存储,jmp,jz需要回写pc。

1. 存储管理模块



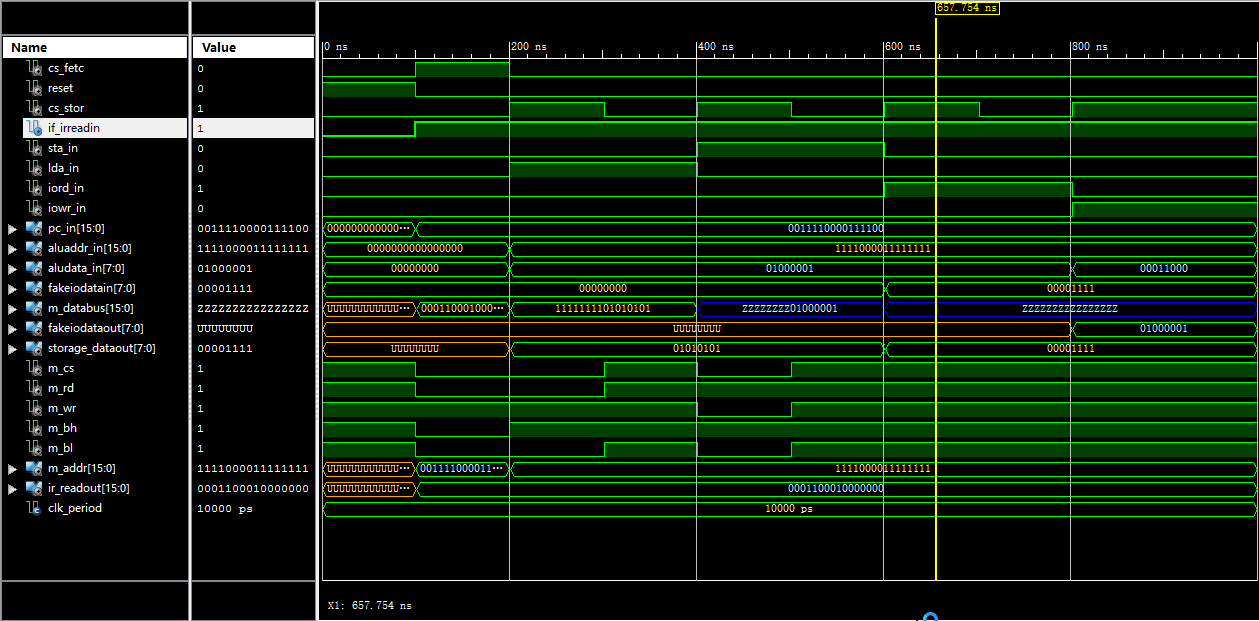
在T2阶段给控制信号，T3阶段依据控制信号控制访存信号。

1. 回写模块



T2阶段给控制信号，T4阶段控制是否回写。

1. 访存控制



访存模块需要读取指令，读取内存，写入内存，读取io，写入io

1. 系统管脚定义的UCF文件

NET "clk" LOC="P75";

NET "reset" LOC="P51"; #k47 为reset

NET "reset" CLOCK\_DEDICATED\_ROUTE = FALSE;

NET "maddrbus<15>" LOC="P145";

NET "maddrbus<14>" LOC="P144";

NET "maddrbus<13>" LOC="P140";

NET "maddrbus<12>" LOC="P119";

NET "maddrbus<11>" LOC="P116";

NET "maddrbus<10>" LOC="P115";

NET "maddrbus<9>" LOC="P113";

NET "maddrbus<8>" LOC="P146";

NET "maddrbus<7>" LOC="P147";

NET "maddrbus<6>" LOC="P150";

NET "maddrbus<5>" LOC="P151";

NET "maddrbus<4>" LOC="P171";

NET "maddrbus<3>" LOC="P172";

NET "maddrbus<2>" LOC="P177";

NET "maddrbus<1>" LOC="P178";

NET "maddrbus<0>" LOC="P179";

NET "mdatabus<15>" LOC="P135";

NET "mdatabus<14>" LOC="P134";

NET "mdatabus<13>" LOC="P133";

NET "mdatabus<12>" LOC="P132";

NET "mdatabus<11>" LOC="P128";

NET "mdatabus<10>" LOC="P123";

NET "mdatabus<9>" LOC="P122";

NET "mdatabus<8>" LOC="P120";

NET "mdatabus<7>" LOC="P153";

NET "mdatabus<6>" LOC="P160";

NET "mdatabus<5>" LOC="P161";

NET "mdatabus<4>" LOC="P162";

NET "mdatabus<3>" LOC="P163";

NET "mdatabus<2>" LOC="P164";

NET "mdatabus<1>" LOC="P165";

NET "mdatabus<0>" LOC="P167";

NET "nMREQ" LOC="P168";

NET "nRD" LOC="P139";

NET "nWR" LOC="P152";

NET "nBHE" LOC="P138";

NET "nBLE" LOC="P137";

NET "fakeIOdatain<7>" LOC="P110";

NET "fakeIOdatain<6>" LOC="P118";

NET "fakeIOdatain<5>" LOC="P124";

NET "fakeIOdatain<4>" LOC="P130";

NET "fakeIOdatain<3>" LOC="P136";

NET "fakeIOdatain<2>" LOC="P142";

NET "fakeIOdatain<1>" LOC="P148";

NET "fakeIOdatain<0>" LOC="P154";

NET "fakeIOdataout<7>" LOC="P185";

NET "fakeIOdataout<6>" LOC="P77";

NET "fakeIOdataout<5>" LOC="P82";

NET "fakeIOdataout<4>" LOC="P83";

NET "fakeIOdataout<3>" LOC="P98";

NET "fakeIOdataout<2>" LOC="P99";

NET "fakeIOdataout<1>" LOC="P100";

NET "fakeIOdataout<0>" LOC="P102";

1. 处理器功能测试程序，包括助记符和二进制代码。
2. 加法测试：分别把11h，22h赋值给r0，r1，然后r0，r1相加，输出r0

mvi r0,11h

mvi r1,22h

add r0,r1

out r0

二进制代码：1811 1922 0001 2000

测试的指令：mvi add out mov

1. 条件跳转,：设置r0=0，基址r7=00h，r1=1F；在目标地址处显示r1的内容

mvi r0,00h

mvi r7,00h

mvi r1,1F

jz r0,10h

（at 0x0010h）out r1

二进制代码: 1800 1F00 191F 4010 …（at 0x0010）2100

测试的指令：jz jmp（同理）

1. 复制数据：在0x0180h处有数据006F，通过程序把它复制到0x0181h处

mvi r7,01h

lda r0,80h

out r0

sta r0,81h

二进制代码：1F01 8880 2000 8081

at 0x0180h:006F

测试的指令:lda sta

1. in输入：in输入r0，复制到r1；再输入r0，相减，显示r0结果

in r0 ; --输入11h

mov r1,r0;

in r0;--输入22h

sub r0，r1

out r0

二进制代码：2800 1100 2800 0801 2000

测试的代码:in sub mov

总程序：(0x0000)1811 1922 0001 2000 1800 1F00 191F 4010 (0x0010)2100 1F01 8880 2000 8081 2800 1100 2800 0801 2000 (0x0180)006F

1. 设计、调试、波形、下载过程中遇到的问题及解决方法

遇到了一系列问题，比如ise 64 位版本不能在win10上正常工作，于是重装winxp系统，发现版本太高运行不了，只好去网吧运行。

下载过程中的主要问题是实验室没有开放足够的时间调试，或者没有通知，或者学生的消息灵通，总之没有太多时间调试。

1. 实验体会

实验很给力，全部都是自己做的，没有任何人的干预，当然也谈不上任何人的帮助。好评，5星，会推荐学弟来学。