**CPU实验报告**

1. **CPU接口定义**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 信号名 | 位数 | 方向 | 来源/去向 | 意义 |
| RST | 1 | I | 处理器板 | 高电平复位 |
| CLK | 1 | I | 处理器板 | 系统时钟 |
| ABUS | 16 | O | 主存储器 | 地址总线 |
| DBUS | 16 | I/O | 主存储器 | 数据总线 |
| MREQ | 1 | O | 主存储器 | 存储器片选信号 |
| RD | 1 | O | 主存储器 | 存储器读信号 |
| WR | 1 | O | 主存储器 | 存储器写信号 |
| BHE | 1 | O | 主存储器 | 高位字节允许访问 |
| BLE | 1 | O | 主存储器 | 低位字节允许访问 |

1. **各模块接口定义**

**ALU模块：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 信号名 | 位数 | 方向 | 来源/去向 | 意义 |
| A | 16 | I | 内部数据总线 | 第一操作数 |
| B | 16 | I | OpReg | 第二操作数 |
| Sel | 4 | I | Control | 操作码 |
| Z | 1 | I | Flag寄存器 | 零标志 |
| Cy | 1 | I | Flag寄存器 | 进位标志 |
| Flag | 8 | O | Flag寄存器 | 8个标志位 |
| C | 16 | O | OutReg | 运算结果 |

**Control模块：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 信号名 | 位数 | 方向 | 来源/去向 | 意义 |
| CLk | 1 | I | 处理器板 | 系统时钟 |
| RST | 1 | I | 处理器板 | 复位信号 |
| Instrreg | 16 | I | 指令寄存器 | 16位指令 |
| Progcntrwr | 1 | O | PC寄存器 | PC寄存器写入信号 |
| Progcntrrd | 1 | O | PC寄存器 | PC寄存器读信号 |
| Addrregwr | 1 | O | 地址寄存器 | 地址寄存器写入信号 |
| Outregwr | 1 | O | OutReg | OutReg写入信号 |
| Outregrd | 1 | O | OutReg | OutReg读信号 |
| Alusel | 4 | O | ALU | ALU操作码 |
| Opregwr | 1 | O | OpReg | OpReg写入信号 |
| Instrwr | 1 | O | 指令寄存器 | 指令寄存器写入信号 |
| Regsel | 3 | O | 通用寄存器组 | 通用寄存器组选择信号 |
| Regwr | 1 | O | 通用寄存器组 | 通用寄存器组写入信号 |
| Regrd | 1 | O | 通用寄存器组 | 通用寄存器组读信号 |
| Mreq | 1 | O | 主存储器 | 存储器片选信号 |
| Rd | 1 | O | 主存储器 | 存储器读信号 |
| Memwr | 1 | O | 主存储器 | 存储器写信号 |
| Bhe | 1 | O | 主存储器 | 高位字节允许访问 |
| Ble | 1 | O | 主存储器 | 低位字节允许访问 |
| Progcntrinc | 1 | O | PC寄存器 | PC寄存器自加1信号 |
| Flagwr | 1 | O | Flag寄存器 | Flag寄存器写入信号 |

**opreg、addrreg、instr模块（Reg模块）：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 信号名 | 位数 | 方向 | 来源/去向 | 意义 |
| A | 16 | I | 数据总线 | 寄存器输入数据 |
| CLK | 1 | I | 处理器板 | 系统时钟 |
| Wr | 1 | I | Control | 寄存器写入信号 |
| Q | 16 | O | 数据总线 | 输出数据 |

**outreg模块：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 信号名 | 位数 | 方向 | 来源/去向 | 意义 |
| A | 16 | I | ALU | 寄存器输入数据 |
| CLK | 1 | I | 处理器板 | 系统时钟 |
| Wr | 1 | I | Control | 写入信号 |
| Rd | 1 | I | Control | 读信号 |
| Q | 16 | O | 数据总线 | 输出数据 |

**progcntr模块：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 信号名 | 位数 | 方向 | 来源/去向 | 意义 |
| A | 16 | I | 数据总线 | 输入数据 |
| CLK | 1 | I | 处理器板 | 系统时钟 |
| Wr | 1 | I | Control | 写入信号 |
| Rd | 1 | I | Control | 读信号 |
| Inc | 1 | I | Control | PC加1信号 |
| Q | 16 | O | 数据总线 | 输出数据 |

**regarray模块：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 信号名 | 位数 | 方向 | 来源/去向 | 意义 |
| Data | 16 | I | 数据总线 | 输入数据 |
| CLK | 1 | I | 处理器板 | 系统时钟 |
| Sel | 3 | I | Control | 寄存器组选择信号 |
| Wr | 1 | I | Control | 写入信号 |
| Rd | 1 | I | Control | 读信号 |
| Q | 16 | O | 系统总线 | 输出数据 |

**Flag模块：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 信号名 | 位数 | 方向 | 来源/去向 | 意义 |
| A | 8 | I | ALU | 8个标志位 |
| CLK | 1 | I | 处理器板 | 系统时钟 |
| Wr | 1 | I | Control | 写入信号 |
| Cy | 1 | O | ALU | 进位标志 |
| Z | 1 | O | ALU | 零标志 |

1. **各模块测试波形**

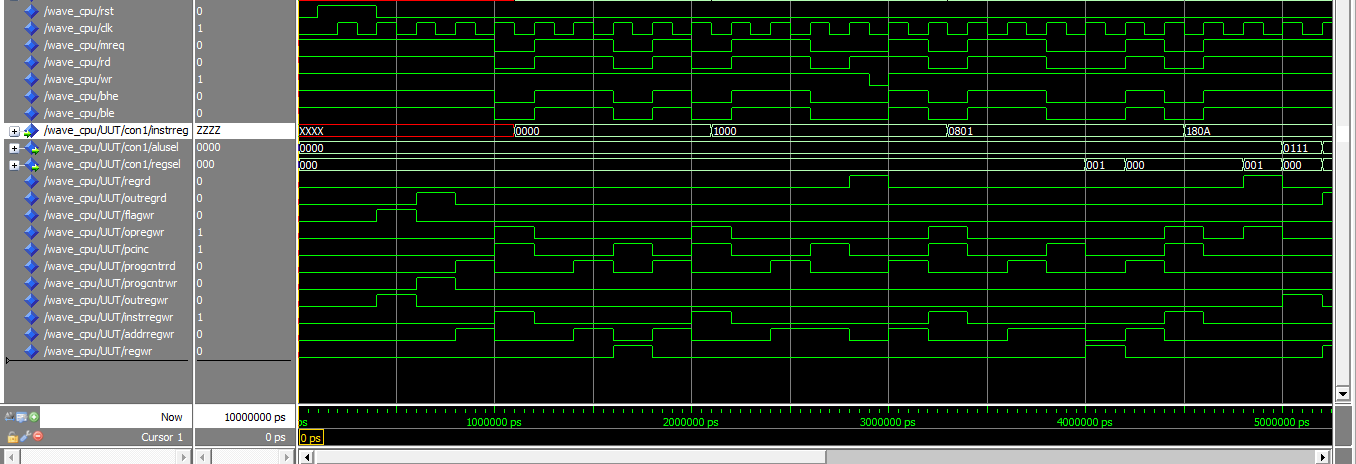
**ALU模块：**

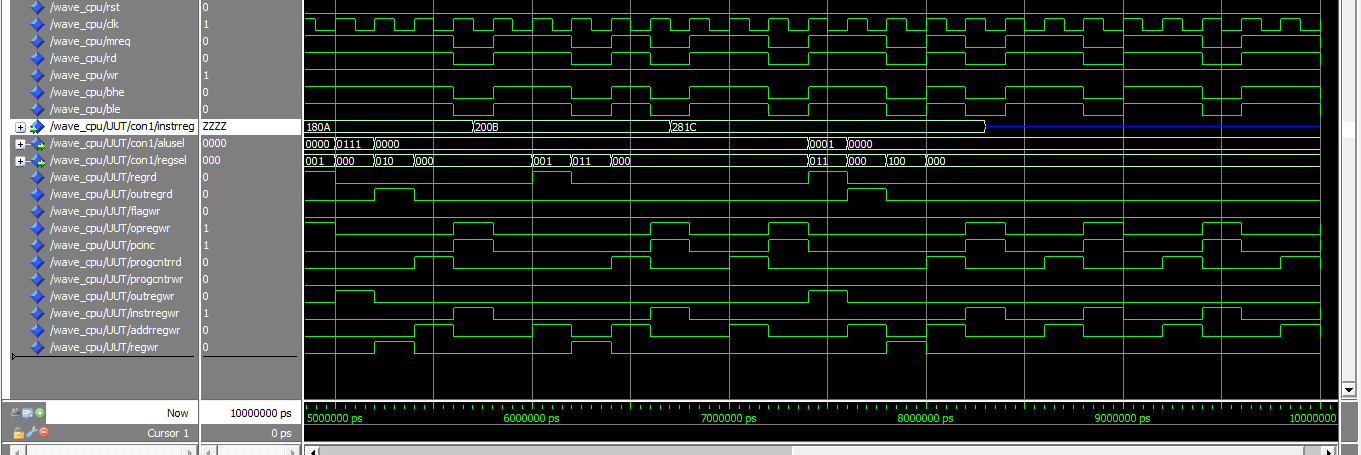


波形含义

Sel是ALU的操作码。当Sel=0000时，ALU输出结果为0000，用于初始化PC，同时由于结果为零所以标志寄存器为02（00000010，零标志Z=1）。当Sel为0001，执行a+b。当Sel=1001，执行算数加法，a+b+Cy，从波形可以看出当Cy由0变成1时，结果由01E0变成01E1，标志寄存器为00（Cy=0，Z=0）。当Sel=1010时，执行a-b-Cy，从波形可以看出Cy=0时，结果为0000，标志寄存器为02，当Cy=1时，结果变为FFFF，同时标志寄存器变为01（Z=0，Cy=1）。当Sel=0011时，执行a and b。当Sel=0100时，执行a or b。当Sel=0101时，用于JZ的跳转运算。当Sel=0110时，用于JC的跳转运算。当Sel=0111时，把b直接作为结果输出（c <= b）。当Sel=1011时，将Cy置零。当Sel=1100时，将Cy置一。

**Control模块：**





波形含义：

CPU初始化阶段，ALU操作码为0000，产生0000数据，outregwr和flagwr有效数据写入OutReg同时修改标志寄存器。然后outregrd和progcntwr同时有效，数据从outreg写入PC寄存器中。之后进入取指周期，progcntrd和addrregwr同时有效，数据从PC寄存器写入addrreg地址寄存器中，然后在下一个周期进行访存操作，这个周期里instrregwr和pcinc有效，指令写入IR指令寄存器中，同时PC完成自动加1。第一条指令操作码为0000，指令为MOV R0，---（立即数），译码后进入一个取指周期，取出指令的后半部分，这时regwr有效并且regsel为000（选中R0），数据写入R0中。

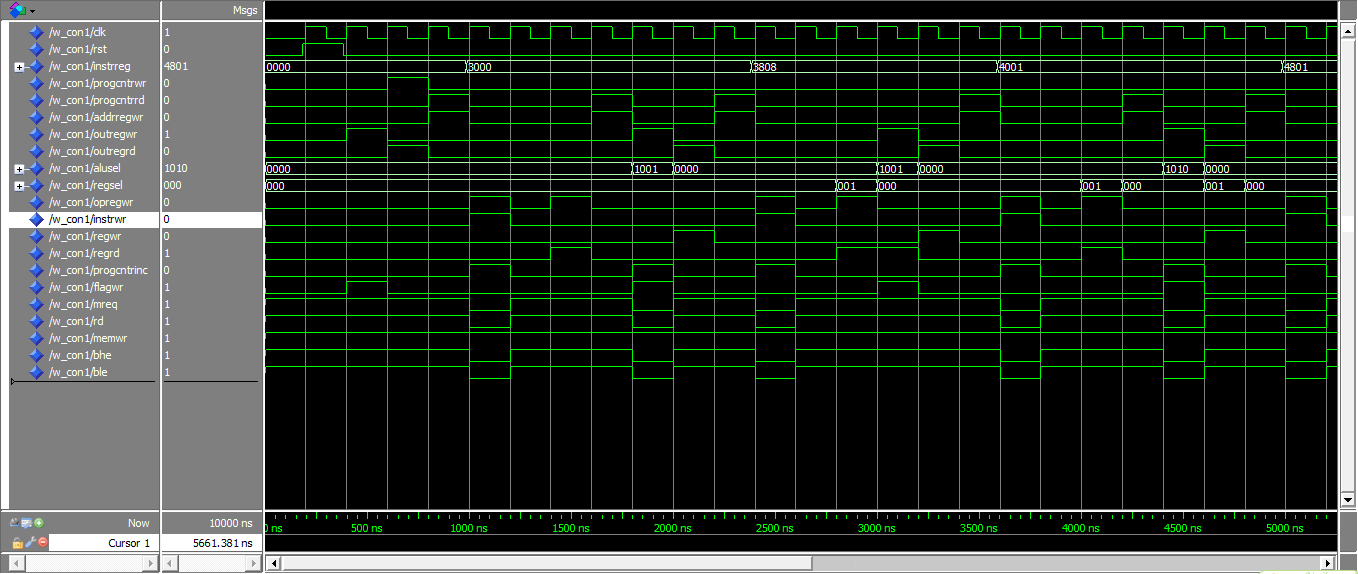
第二条指令操作码为1000，指令为MOV ---（地址），R0。译码后进入一个取指周期，在取指周期的第二个周期里，addrregwr有效，指令的后半部分由内存传输给地址寄存器。下一周期regrd有效，同时regsel为000，R0的数据被放到数据总线上，同时内存写入信号有效，数据由R0写入内存。

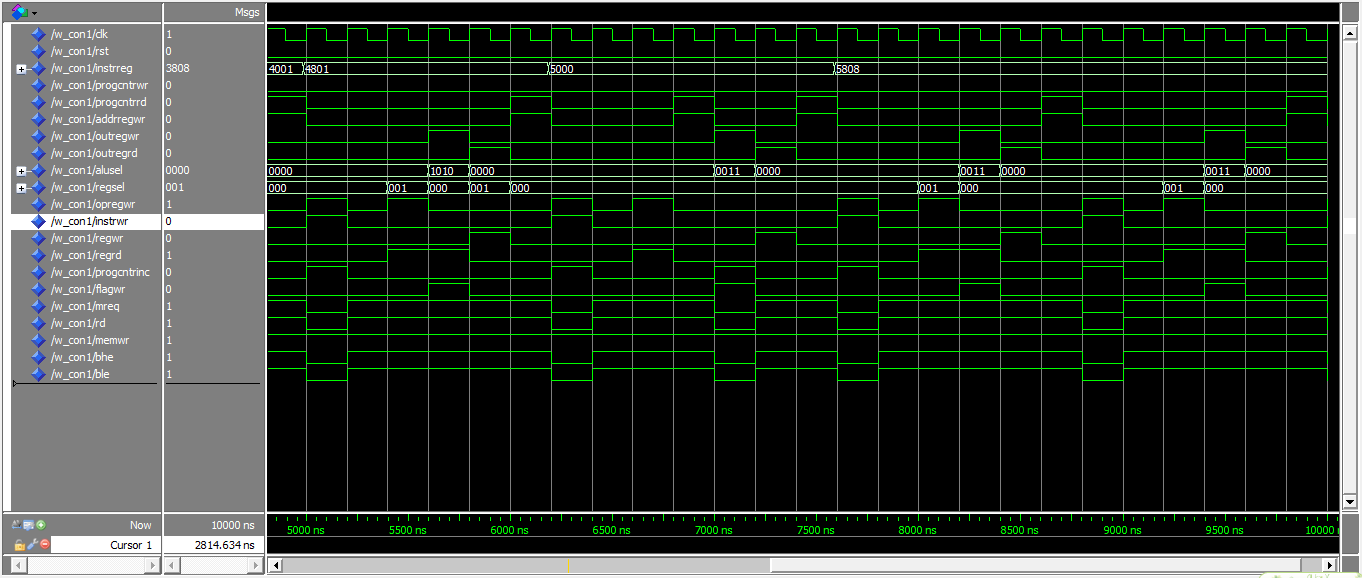
第三条指令操作码为0801，指令为MOV R1，---（地址）。译码后进入一个取指周期，在取指周期的第二个周期里，addrregwr有效，指令的后半部分由内存传输给地址寄存器。下一周期内存读取信号有效，内存的数据被放到数据总线上，同时regwr有效，同时regsel为001，，数据由内存写入R1。

第四条指令操作码为180A，指令为MOV R2，R1。译码后的第一个周期里，regrd有效并且regsel为001，R1的数据放到数据总线上，这时opregwr有效，数据从总线上写入opreg中。下一个周期里，ALU操作码alusel为0111（结果输出b的值，即保持不变），同时outregwr有效，数据写入outreg中。再下一个周期，outregrd有效，数据放入总线上，regwr有效并且regsel为010，数据从总线上写入R2中。

第五条指令操作码为200B，指令为MOV R3，[R1]。regrd有效并且regsel为001，R1的数据放到数据总线上，这时addrregwr有效，数据从总线上写入addrreg中。然后访存信号有效，同时regwr有效并且regsel为011，数据从总线上写入R3中。

第六条指令操作码为2B1C，指令为MOV R4，[R3+---]。译码后进入一个取指周期，在取指周期的第二个周期里，opregwr有效，立即数从内存写入opreg中。然后regrd有效并且regsel为011，R1的数据放到数据总线上，ALU操作码为0001，这时outregwr有效运算结果写入outreg中。下一周期里，outregrd有效，数据放入总线上，addrregwr有效，数据写入地址寄存器。再下一个周期访存，regwr有效并且regsel为100，数据从总线上写入R4中。





波形含义：

第一条指令操作码为3000，指令为ADC R0，---。译码后的第一周期里，regrd有效并且regsel为000，R0的数据放入总线上，同时opregrd有效，总线上的数据写入opreg中。然后是一个取指周期，在取指周期的第二个周期里，访存信号有效，同时ALU操作码alusel为1001（执行A+B+Cy），outregwr和flagwr有效，运算结果写入outreg中8个标志位写入标志寄存器中。再在下一个周期里，outregrd有效，数据放到总线上，同时regwr有效并且regsel为000，数据写入R0中。

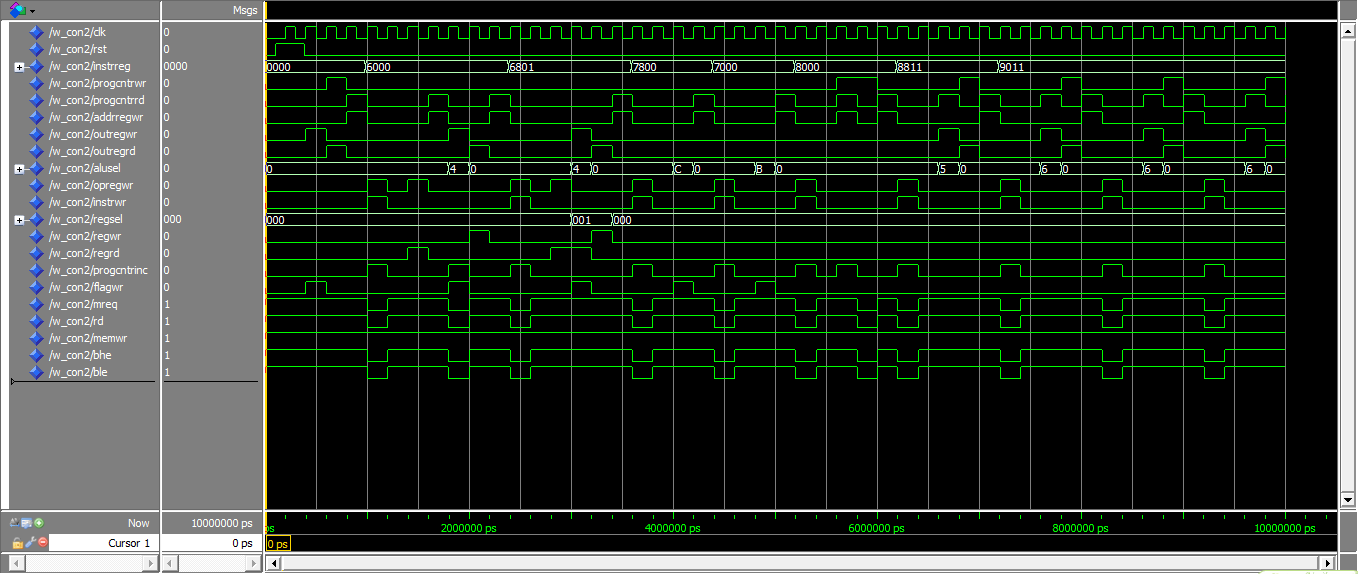
第二条指令操作码为3808，指令为ADC R0，R1。译码后的第一周期里，regrd有效并且regsel为001，R1的数据放入总线上，同时opregrd有效，总线上的数据写入opreg中。下一个周期里，regrd有效并且regsel为000，R0的数据放入总线上，同时ALU操作码alusel为1001（执行A+B+Cy），outregwr和flagwr有效，运算结果写入outreg中8个标志位写入标志寄存器中。再在下一个周期里，outregrd有效，数据放到总线上，同时regwr有效并且regsel为000，数据写入R0中。

第三条指令操作码为4001，指令为SBB R1，---。译码后的第一周期里，regrd有效并且regsel为001，R1的数据放入总线上，同时opregrd有效，总线上的数据写入opreg中。然后是一个取指周期，在取指周期的第二个周期里，访存信号有效，同时ALU操作码alusel为1010（执行A-B-Cy），outregwr和flagwr有效，运算结果写入outreg中8个标志位写入标志寄存器中。再在下一个周期里，outregrd有效，数据放到总线上，同时regwr有效并且regsel为001，数据写入R1中。

第四条指令操作码为4801，指令为SBB R1，R0。译码后的第一周期里，regrd有效并且regsel为001，R1的数据放入总线上，同时opregrd有效，总线上的数据写入opreg中。下一个周期里，regrd有效并且regsel为000，R0的数据放入总线上，同时ALU操作码alusel为1010（执行A-B-Cy），outregwr和flagwr有效，运算结果写入outreg中8个标志位写入标志寄存器中。再在下一个周期里，outregrd有效，数据放到总线上，同时regwr有效并且regsel为001，数据写入R1中。

第五条指令操作码为5000，指令为AND R0，---。译码后的第一周期里，regrd有效并且regsel为000，R0的数据放入总线上，同时opregrd有效，总线上的数据写入opreg中。然后是一个取指周期，在取指周期的第二个周期里，访存信号有效，同时ALU操作码alusel为0011（执行A AND B），outregwr和flagwr有效，运算结果写入outreg中8个标志位写入标志寄存器中。再在下一个周期里，outregrd有效，数据放到总线上，同时regwr有效并且regsel为000，数据写入R0中。

第六条指令操作码为5808，指令为AND R0，R1。译码后的第一周期里，regrd有效并且regsel为001，R1的数据放入总线上，同时opregrd有效，总线上的数据写入opreg中。下一个周期里，regrd有效并且regsel为000，R0的数据放入总线上，同时ALU操作码alusel为0011（执行A AND B），outregwr和flagwr有效，运算结果写入outreg中8个标志位写入标志寄存器中。再在下一个周期里，outregrd有效，数据放到总线上，同时regwr有效并且regsel为000，数据写入R0中。



波形含义：

第一条指令操作码为6000，指令为OR R0，---。译码后的第一周期里，regrd有效并且regsel为000，R0的数据放入总线上，同时opregrd有效，总线上的数据写入opreg中。然后是一个取指周期，在取指周期的第二个周期里，访存信号有效，同时ALU操作码alusel为0100（执行A OR B），outregwr和flagwr有效，运算结果写入outreg中8个标志位写入标志寄存器中。再在下一个周期里，outregrd有效，数据放到总线上，同时regwr有效并且regsel为000，数据写入R0中。

第二条指令操作码为6801，指令为OR R1，R0。译码后的第一周期里，regrd有效并且regsel为000，R0的数据放入总线上，同时opregrd有效，总线上的数据写入opreg中。下一个周期里，regrd有效并且regsel为001，R1的数据放入总线上，同时ALU操作码alusel为0100（执行A OR B），outregwr和flagwr有效，运算结果写入outreg中8个标志位写入标志寄存器中。再在下一个周期里，outregrd有效，数据放到总线上，同时regwr有效并且regsel为001，数据写入R1中。

第三条指令操作码为7800，指令为STC。译码后的第一个周期里，ALU操作码alusel为1100（将Cy置一），同时flagwr有效，标志写入标志寄存器。

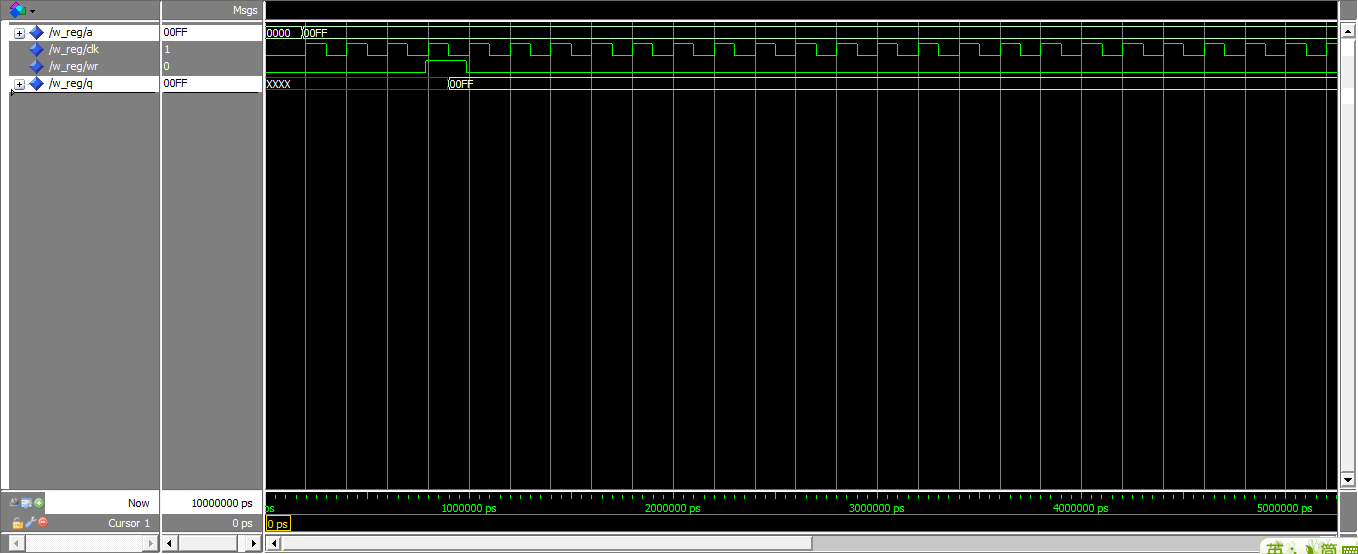
第四条指令操作码为7000，指令为CLC。译码后的第一个周期里，ALU操作码alusel为1011（将Cy置零），同时flagwr有效，标志写入标志寄存器。

第五条指令操作码为8000，指令为JMP ---。译码后进入一个取指周期，取出指令的后半部分，在取指周期的第二个周期里，访存信号有效，同时progcntwr有效，数写入到PC寄存器中。

第六条指令操作码为8811，指令为JZ 0011。译码后的第一个周期里，progcntrd有效，PC寄存器里的内容读入到总线上，同时ALU操作码alusel为0101（用于JZ指令对PC的值进行加减运算），outregwr有效，运算结果写入outreg中。再在下一个周期里，outregrd有效，数据放到总线上，progcntw有效，总线上的数据写入PC寄存器。

第七条指令操作码为9011，指令为JC 0011。译码后的第一个周期里，progcntrd有效，PC寄存器里的内容读入到总线上，同时ALU操作码alusel为0110（用于JC指令对PC的值进行加减运算），outregwr有效，运算结果写入outreg中。再在下一个周期里，outregrd有效，数据放到总线上，progcntw有效，总线上的数据写入PC寄存器。

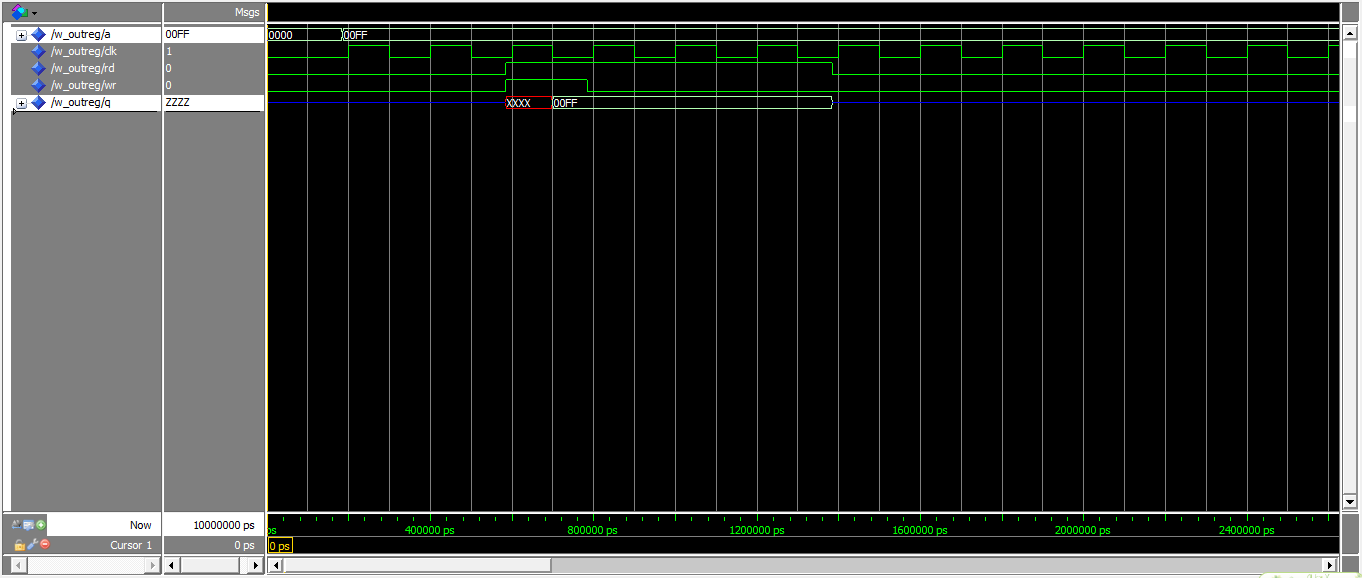
**opreg、addrreg、instr模块（Reg模块）：**



波形含义：

当Wr=1时，数据在时钟下降沿写入寄存器中。由于寄存器的初始状态未知，所以会有一段红线。

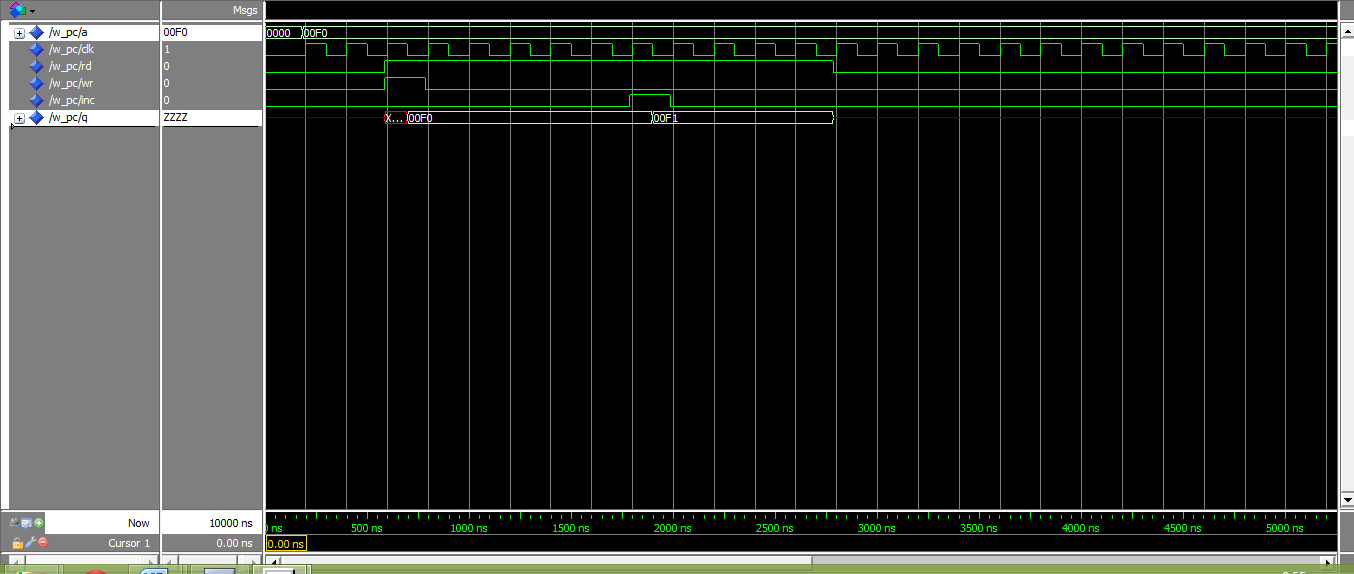
**Outreg模块：**



波形含义：

当Wr=1，数据在时钟下降沿写入。当Rd=0时，输出保持高阻态，当Rd=1时，将寄存器里的值输出。由于寄存器的初始状态未知，所以会有一段红线。

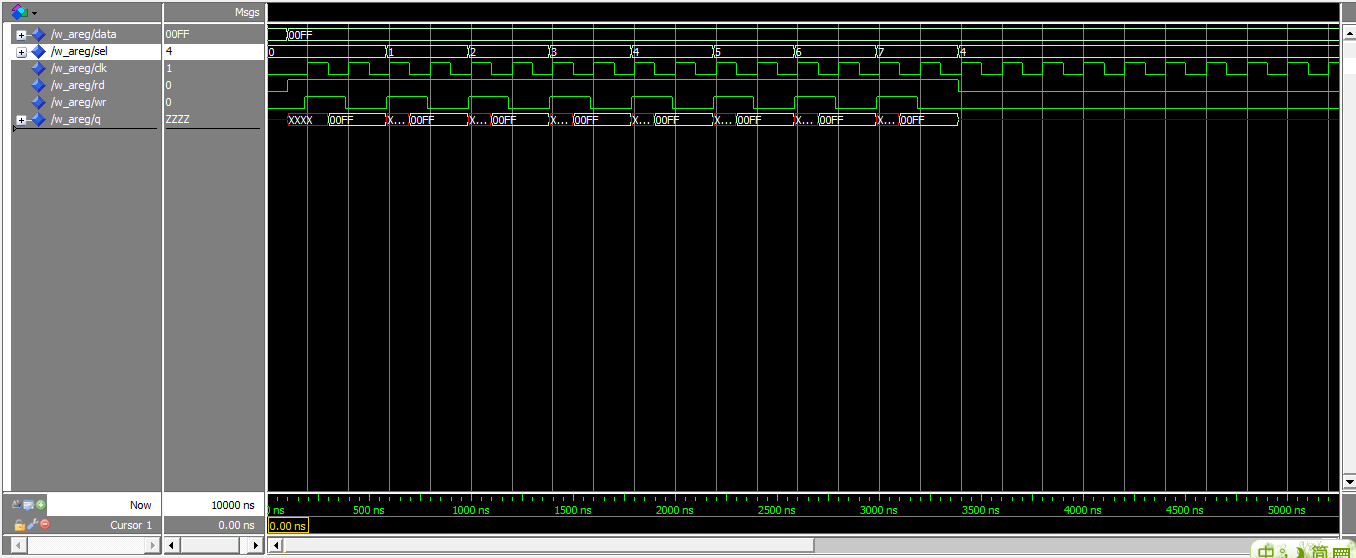
**Progcntr模块：**



波形含义：

当Wr=1，数据在时钟下降沿写入。当Rd=0时，输出保持高阻态，当Rd=1时，将寄存器里的值输出。当Inc=1时，寄存器的值在时钟下降沿时自动加1。由于寄存器的初始状态未知，所以会有一段红线。

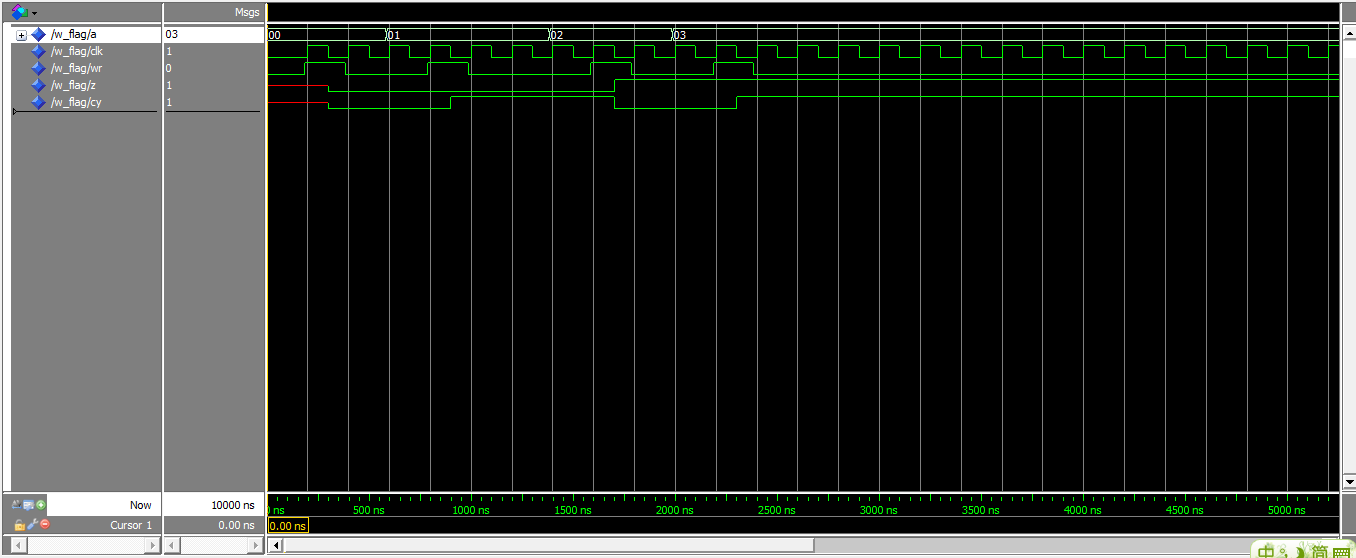
**Regarray模块：**



波形含义：

Sel的值标明当前使用第几号通用寄存器。当Wr=1，数据在时钟下降沿写入。当Rd=0时，输出保持高阻态，当Rd=1时，将寄存器里的值输出。由于寄存器的初始状态未知，所以会有一段红线。

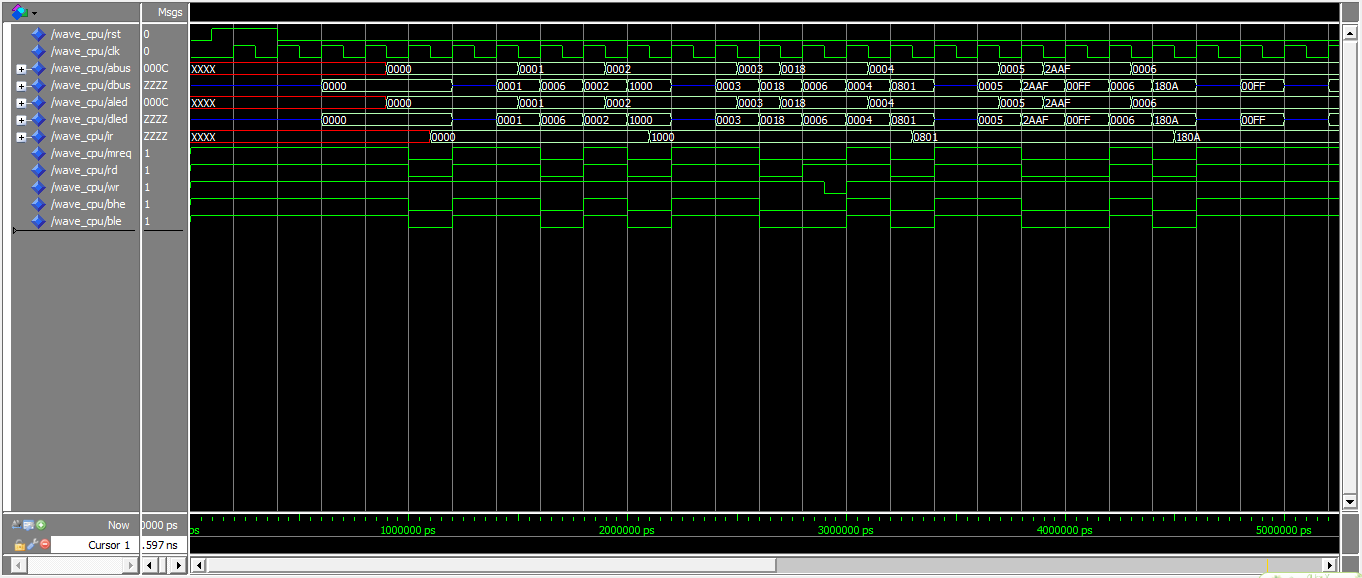
**Flag模块：**

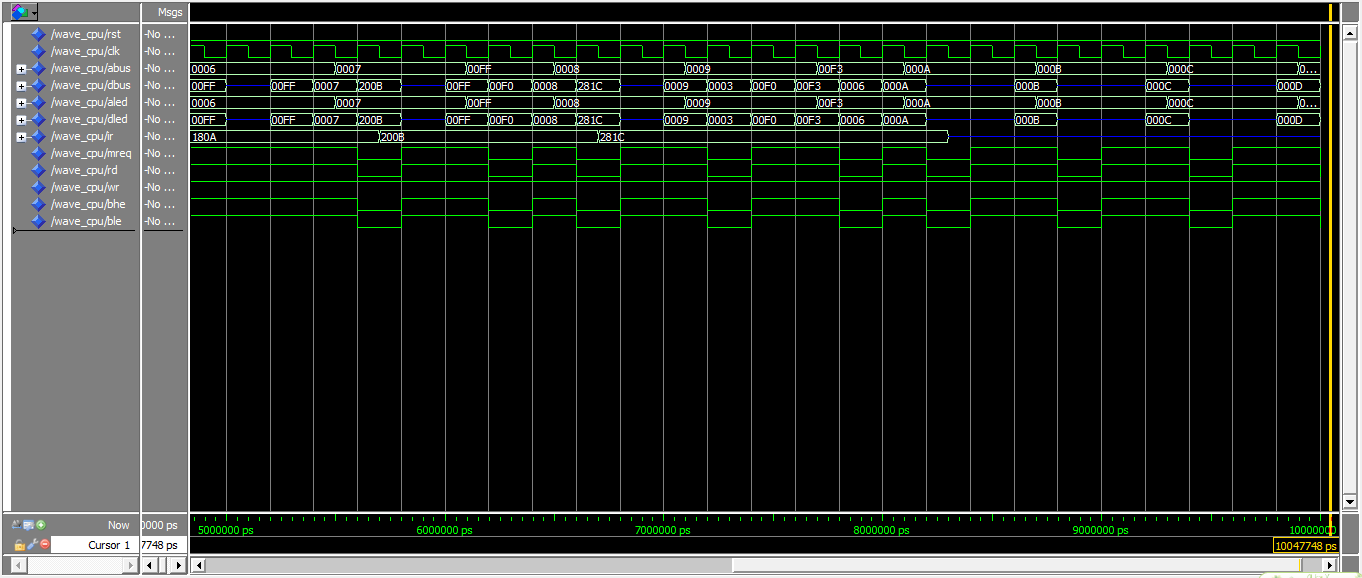


波形含义：

当Wr=1时，数据在时钟下降沿写入寄存器中。由于寄存器的初始状态未知，所以会有一段红线。寄存器内容为00时，Cy=0，Z=1；内容为01时，Z=0，Cy=1；内容为02时，Z=1，Cy=0；内容为03时，Z=1，Cy=1。

1. **总体测试波形**

第一部分波形：

覆盖的指令：

MOV R0,0006 立即数寻址

MOV [0018],R0 直接寻址，数据从寄存器到内存

MOV R1,[2AAF] 直接寻址，数据从内存到寄存器

MOV R2,R1 寄存器寻址

MOV R3,[R1] 间接寻址

MOV R4,[R3+0003] 变址寻址

波形含义：

当rst无效时，经过两个周期的初始化进入取指周期的第一个周期。第一个周期里，PC将值放到数据总线上，addrreg在时钟下降沿写入数据，所以abus的值在下降沿改变。取值第二周期里读内存信号有效，数据总线上出现第一条指令，ir信号在时钟下降沿改变，之后译码执行。由于该指令为双字，所有取指令的另一半，数据总线上出现0006，写入R0。

第二条指令可以检查0006是否已经写入到R0中。继续取指，译码执行。由于这是一条双字的指令，所以执行周期里仍有一次取指，0018出现在数据总线上，在时钟下降沿写入addrreg中。接着R0中的值0006出现在数据总线上，abus的值为0018，内存写入信号有效，0006数据写入到内存中地址为0018的存储单元。

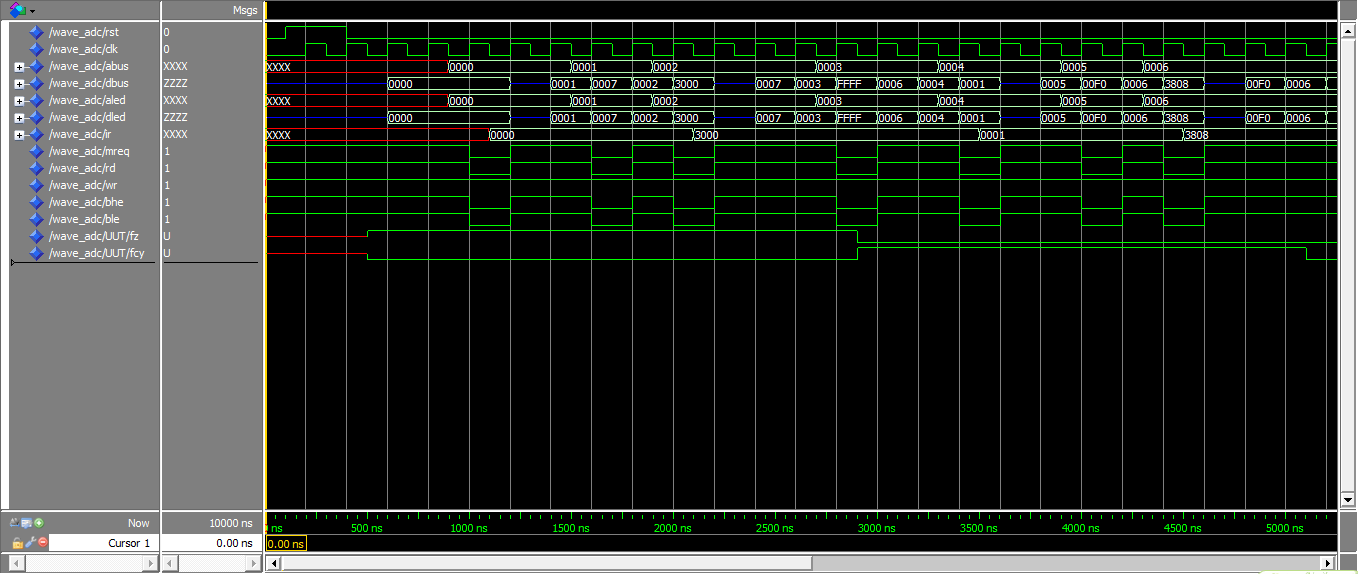
第三条指令是内存到寄存器。取指，译码，执行。这也是一条双字指令，译码后多一个取指周期，2AAF从内存中取出在时钟下降沿写入addrreg中。abus变成2AAF，这时，内存读信号有效，2AAF地址的数据00FF出现在数据总线上，写入R1中。

第四条指令是寄存器到寄存器。由于寄存器到寄存器的传输是通过ALU（ALU不进行任何运算）来中转的，所以可以看到R1的值00FF两次出现在数据总线上，一次是从R1到ALU，一次是从ALU的outreg到R2。

第五条指令是寄存器间接寻址。可以看到取指译码结束后R1中的数据00FF出现在数据总线上，并且在时钟下降沿写入到addrreg中，abus的值变成00FF。这时读信号有效，内存00FF地址处的值被读出，00F0写入R3中。

第六条指令是变址寻址。该条指令指示用R3作为变址寄存器，这是一条双字指令，执行周期里多一个取值周期。指令的后部分0003出现在数据总线上，写入opreg中，之后R3中的值00F0出现在数据总线上，相加后的结果00F3延后一个周期出现在数据总线上，abus在时钟下降沿变成00F3。00F3地址中的值0006写入R4中。

第二部分波形：





覆盖指令：

MOV R0,0007（非测试指令）

ADC R0,FFFF

MOV R1,00F0（非测试指令）

ADC R0,R1

SBB R1,000F

STC

SBB R1,R0

CLC

波形含义：

CPU初始化时，是通过ALU产生0000然后通过数据总线赋值给PC，所以初始化后Z零标志为1而Cy进位标志为0。首先给R0赋值0007，便于后边测试加法运算。第一条测试指令为加法指令，这是一条双字指令，在执行周期里会有一个取指周期（两时钟周期）。译码后的第一个周期里将R0的值0007放到数据总线上，传输给OpReg。然后取指令的后半部分，FFFF出现在数据总线上，这时将ALU的运算值写入OutReg，同时修改标志寄存器（Z=0，Cy=1）。在下一个周期里，OutReg的内容放到数据总线上，写入R0里。

第二条测试指令是两个寄存器相加（在这之前先将R1赋值为00F0）。取指，译码，之后在一个周期内将R1的值00F0通过数据总线给OpReg，然后下一个周期将R0的值0006放到数据总线上，同时就可以将ALU的值00F7（由于Cy=1）在时钟下降沿时写入OutReg中，并修改标志寄存器（Z=0，Cy=0）。之后一个周期将OutReg中的值放到数据总线上，写入R1。

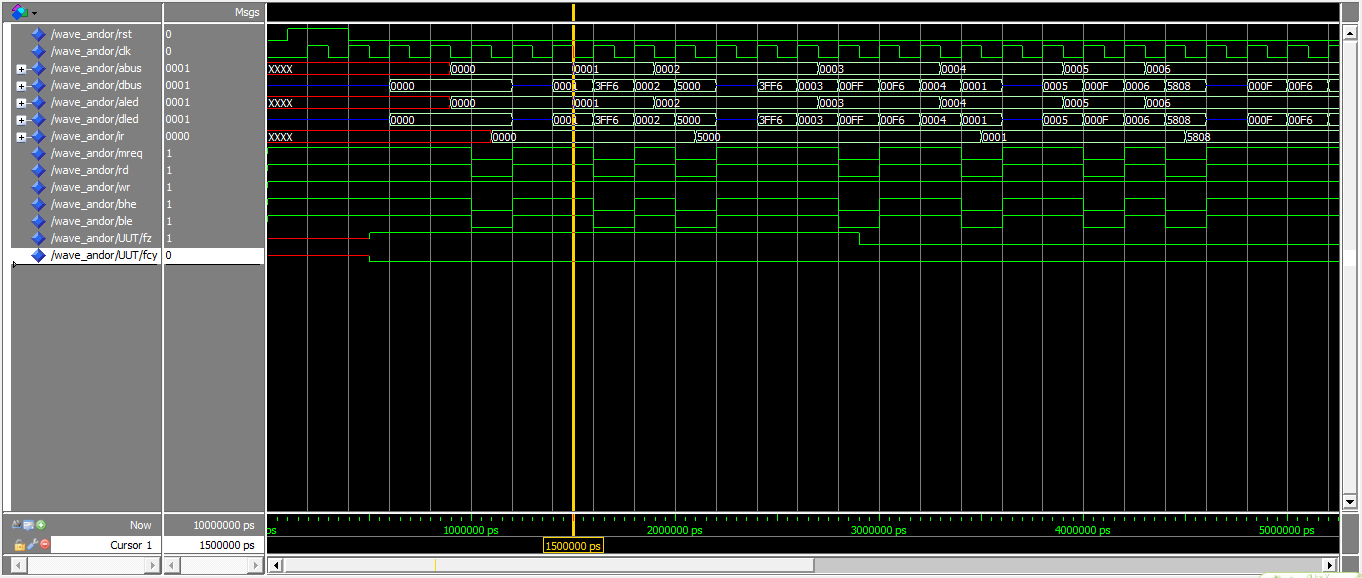
第三条测试指令为减法指令。取指，译码后的第一个周期里R1的值00F0出现在数据总线上，OpReg在时钟下降沿写入该值。然后是一个取指周期（两时钟周期），将指令的后半部分000F放到数据总线上，同时将结果00E1写入OutReg中并修改标志寄存器。下一个时钟周期里OutReg将值放到数据总线上，写入R1中。

第四条指令测试STC，将Cy进位标志直接置一。取指，译码后的一个周期里，Cy由零变成了一。

第五条指令测试寄存器间的减法。R1的值00E1首先通过数据总线传输给OpReg，下一个周期里R0的值00F7出现在数据总线上，同时将ALU的结果FFE9（由于Cy=1）在时钟下降沿写入OutReg并修改标志寄存器。之后一个周期将OutReg的值通过数据总线写入到R1中。

第六条指令测试CLC，直接将Cy标志置零。取值，译码后的一个周期里，Cy由一变成了零。

第三部分波形：





覆盖指令：

MOV R0,3FF6（非测试指令）

AND R0,00FF

MOV R1,000F（非测试指令）

AND R0,R1

OR R0,F080

OR R1,R0

JZ 0011

波形含义：

第一条测试指令为寄存器与立即数求与（在这条指令之前先将R0赋值为3FF6）。取指，译码后的第一个周期里，R0的值3FF6通过数据总线传输给OpReg。然后进入一个取指周期（两时钟周期），指令的后半部分00FF出现在数据总线上，同时将ALU运算结果00F6写入OutReg中并修改标志寄存器（Z=0）。下一个周期里将OutReg里的值通过数据总线写入到R0。

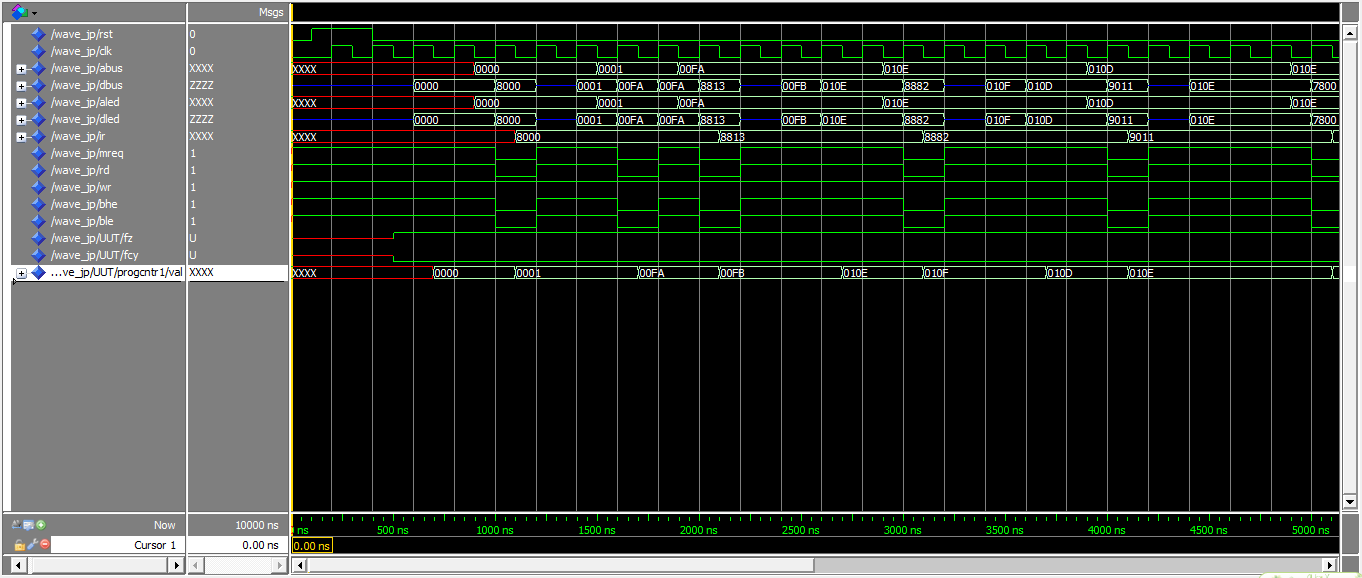
第二条测试指令为寄存器与寄存器做与（在该条指令之前先将R1赋值为000F）。取指，译码后的第一个周期里将R1的值000F通过数据总线传输给OpReg。然后下一个周期里R0的值00F6出现在数据总线上，同时将ALU的结果0006在时钟下降沿写入OutReg中并修改标志寄存器。再后一个周期里，OutReg的值通过数据总线写入R0。

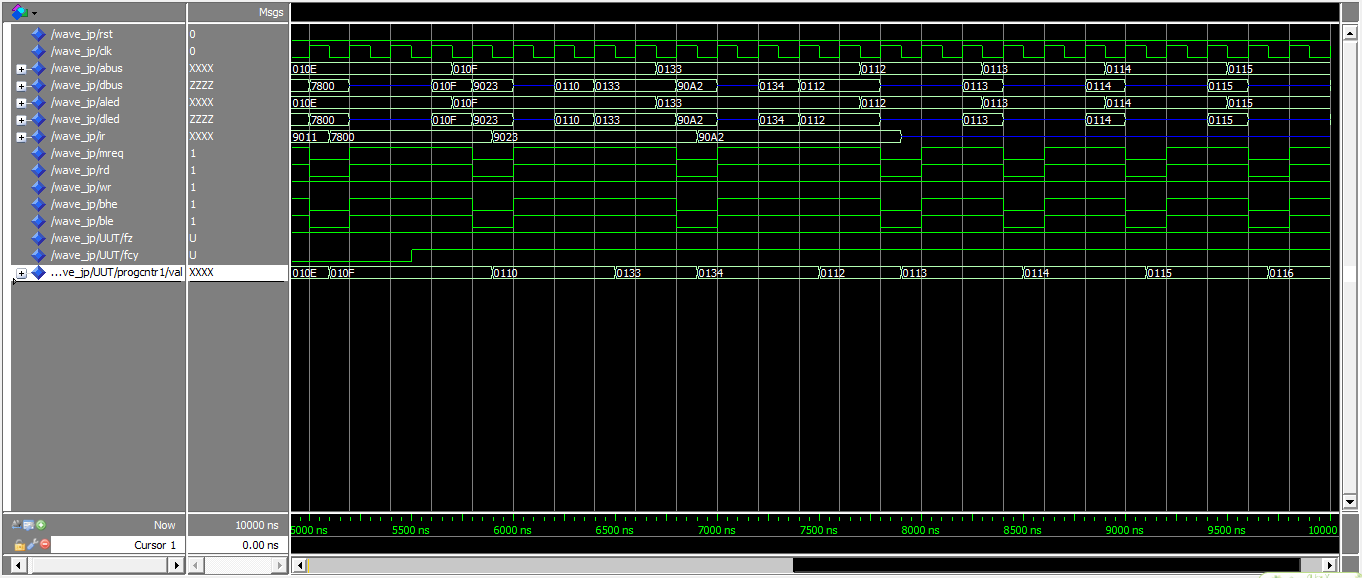
第三条测试指令为寄存器与立即数求或。取指，译码后的第一个周期里，R0的值0006通过数据总线传输给OpReg。然后进入一个取指周期（两时钟周期），指令的后半部分F080出现在数据总线上，同时将ALU运算结果F086写入OutReg中并修改标志寄存器。下一个周期里将OutReg里的值通过数据总线写入到R0。

第四条测试指令为寄存器与寄存器做或操作。取指，译码后的第一个周期里将R0的值F086通过数据总线传输给OpReg。然后下一个周期里R1的值000F出现在数据总线上，同时将ALU的结果F08F在时钟下降沿写入OutReg中并修改标志寄存器。再后一个周期里，OutReg的值通过数据总线写入R0。

第五条测试指令为JZ，即当Z=1是跳转。由于此时零标志Z=0，所以跳转无效，PC值不改变。

第四部分波形：





覆盖指令：

JMP 00FA

JZ 0013

JZ 0082

JC 0011

STC（非测试指令）

JC 0023

JC 00A2

波形含义：

第一条测试指令为无条件跳转JMP。该指令为双字长，取指，译码后进入一个取指周期（两个时钟周期），00FA出现在数据总线上，在时钟下降沿写入PC。之后进入下一个指令周期，可以看到在取指周期里abus的值已变为00FA。

第二条为有条件跳转指令JZ，跳转条件为零标志Z=1。这是一条单字长指令，并且指令的第八位是一个有符号数。从波形可以看出译码后的第一个周期里PC的值（PC已经完成自动加1操作）出现在数据总线上，同时ALU的运算结果会写入OutReg里。下一个周期里结果010E通过数据总线写入PC里。从下一个指令周期的取指周期可以看出abus的值的确已变为010E。

第三条为有条件跳转指令JZ，跳转条件为零标志Z=1。这是一条单字长指令，并且指令的第八位是一个有符号数。从波形可以看出译码后的第一个周期里PC的值（PC已经完成自动加1操作）出现在数据总线上，同时ALU的运算结果会写入OutReg里。下一个周期里结果010D通过数据总线写入PC里。从下一个指令周期的取指周期可以看出abus的值的确已变为010D。

第四条为有条件跳转指令JC，跳转条件为进位标志Cy=1。由于此时Cy=0，条件不满足，所以从波形可以看出PC的值没有发生改变。

第五条为有条件跳转指令JC，跳转条件为进位标志Cy=1。此时Cy=1，条件满足，这是一条单字长指令，并且指令的第八位是一个有符号数。从波形可以看出译码后的第一个周期里PC的值（PC已经完成自动加1操作）0110出现在数据总线上，同时ALU的运算结果会写入OutReg里。下一个周期里结果0133通过数据总线写入PC里。从下一个指令周期的取指周期可以看出abus的值的确已变为0133。

第六条为有条件跳转指令JC，跳转条件为进位标志Cy=1。此时Cy=1，条件满足，这是一条单字长指令，并且指令的第八位是一个有符号数。从波形可以看出译码后的第一个周期里PC的值（PC已经完成自动加1操作）0134出现在数据总线上，同时ALU的运算结果会写入OutReg里。下一个周期里结果0112通过数据总线写入PC里。从下一个指令周期的取指周期可以看出abus的值的确已变为0112。

1. **测试指令序列**

|  |  |  |
| --- | --- | --- |
| 地址 | 储存器内容 | 指令 |
| 0000 | 0000 | MOV R0,00F6 |
| 0001 | 00F6 |  |
| 0002 | 0801 | MOV R1,[00A0] |
| 0003 | 00A0 |  |
| 0004 | 1000 | MOV [00B0],R0 |
| 0005 | 00B0 |  |
| 0006 | 180A | MOV R2,R1 |
| 0007 | 2013 | MOV R3,[R2] |
| 0008 | 281C | MOV R4,[R3+0006] |
| 0009 | 0006 |  |
| 000A | 1818 | MOV R0,R3 |
| 000B | 3004 | ADC R4,FA00 |
| 000C | FA00 |  |
| 000D | 3004 | ADC R4,AF54 |
| 000E | AF54 |  |
| 000F | 3804 | ADC R4,R0 |
| 0010 | 4004 | SBB R4,0005 |
| 0011 | 0005 |  |
| 0012 | 4000 | SBB R0, 00FF |
| 0013 | 00FF |  |
| 0014 | 4819 | SBB R1,R3 |
| 0015 | 5001 | AND R1,00F0 |
| 0016 | 00F0 |  |
| 0017 | 5819 | AND R1,R3 |
| 0018 | 6003 | OR R3,0F06 |
| 0019 | 0F06 |  |
| 001A | 6819 | OR R1,R3 |
| 001B | 7000 | CLC |
| 001C | 3001 | ADC R1,0002 |
| 001E | 0002 |  |
| 001F | 7800 | STC |
| 0020 | 3001 | ADC R1,0001 |
| 0021 | 0001 |  |
| 0022 | 8000 | JMP 0030 |
| 0023 | 0030 |  |
| 0030 | 8811 | JZ 0011 |
| 0031 | 5000 | AND R0,0000 |
| 0032 | 0000 |  |
| 0033 | 881F | JZ IF |
| 0053 | 889A | JZ 9A |
| 003A | 7000 | CLC |
| 003B | 9011 | JC 11 |
| 003C | 7800 | STC |
| 003E | 901F | JC 1F |
| 005D | 9083 | JC 83 |
| 005B | 0000 |  |
| 00A0 | 0070 |  |
| 0070 | 0040 |  |
| 0046 | 56AC |  |