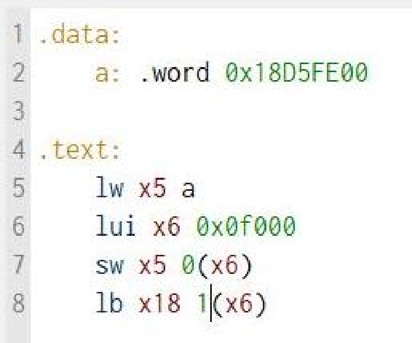
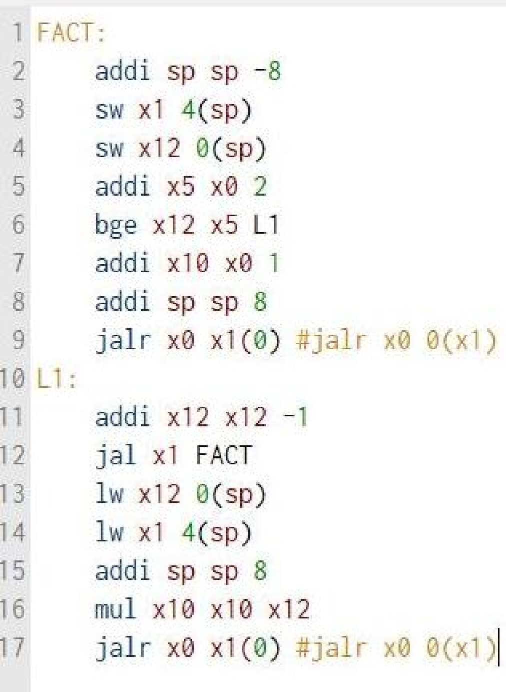


```
.data:
    a: .word 0x18D5FE00

.text:
    lw x5 a
    lui x6 0x0f000
    sw x5 0(x6)
    lb x18 1(x6)
```

x18	s2	0xfffffffe



```
FACT:
    addi sp sp -8
    sw x1 4(sp)
    sw x12 0(sp)
    addi x5 x0 2
    bge x12 x5 L1
    addi x10 x0 1
    addi sp sp 8
    jalr x0 x1(0) #jalr x0 0(x1)
L1:
    addi x12 x12 -1
    jal x1 FACT
    lw x12 0(sp)
    lw x1 4(sp)
    addi sp sp 8
    mul x10 x10 x12
    jalr x0 x1(0) #jalr x0 0(x1)
```

HW2 徐卫晴 520021910400 XuWeiqing

3. (1) ~~J-type~~ B-type

(2)
```
loop: blt x29 x0 Exit.
      beq x29 x0 Exit.
      addi x29 x29 -1
      jal x0 loop
Exit:
```

4. opcode: 1101111 jal J-type

rd: 10101 x21

Immediate number: $(11010000100111111111)_2 = (-)91489|_{10}$

(1) jal x21 PC-191489

(2) J-type

5. (1)

1111111	10101	00010	010	10000	0100011
Imm[11:5]	rs2	rs1	funct3	Imm[4:0]	opcode.

(2) S-type $Imm = (-16)_{10} = (111111110000)_2$

$rs2 = (21)_{10} = (10101)_2$

$rs1 = x2$

6. (1) R-type:

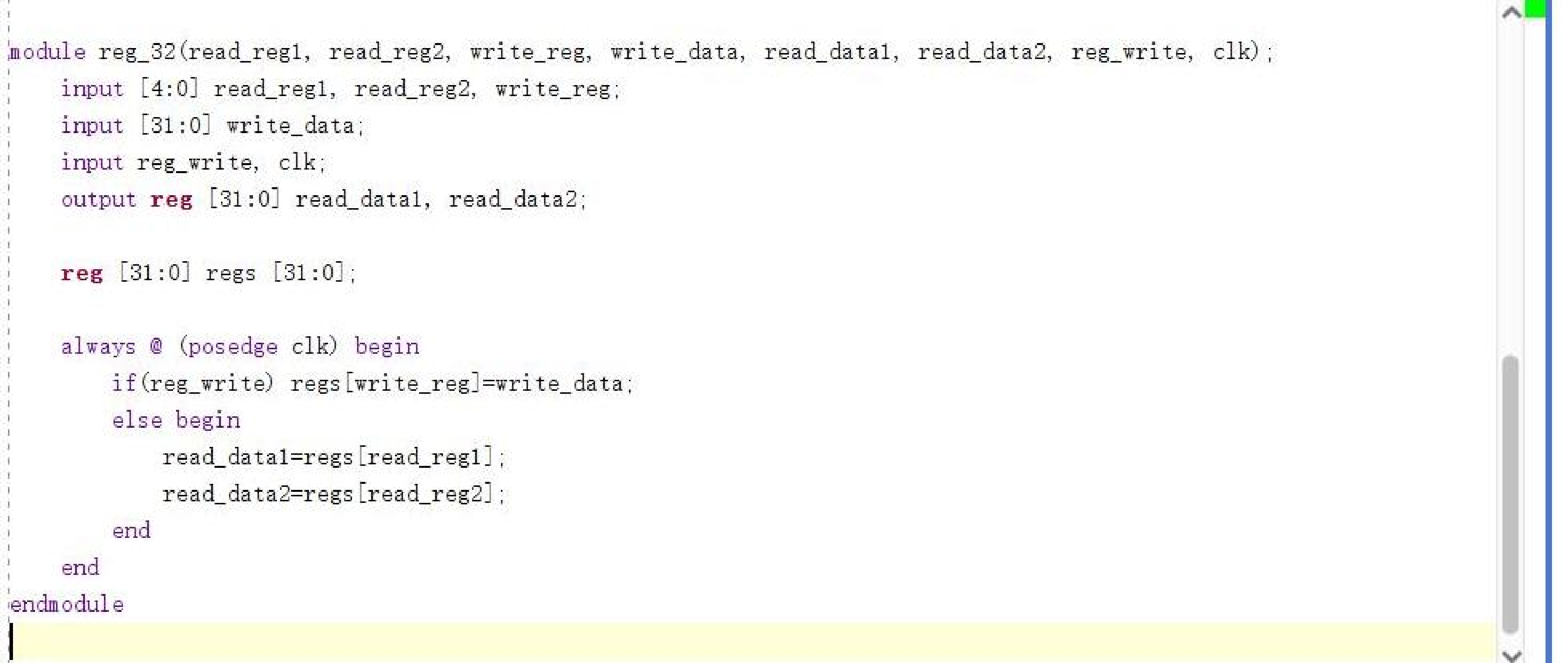
funct7	rs2	rs1	funct3	rd	Opcode
7bits	6 bits	6bits	3bits	6bits	7bits

Total: 35 bits

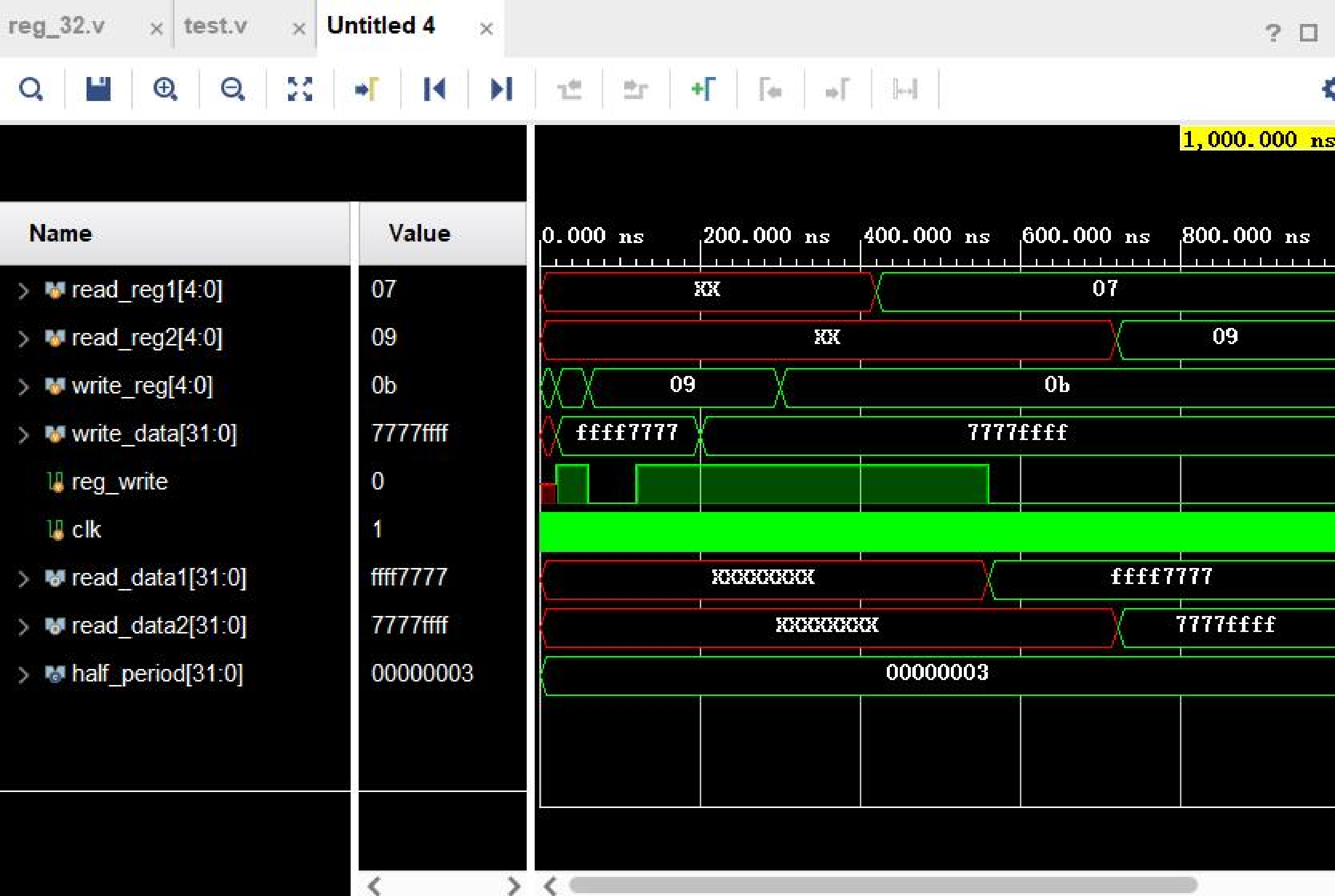
(2) The number of bits of Immediate will come to [illegible] bits instead of 12 bits

(3) Immediate number: 10 bits range: $\pm 2^{10}$ bytes

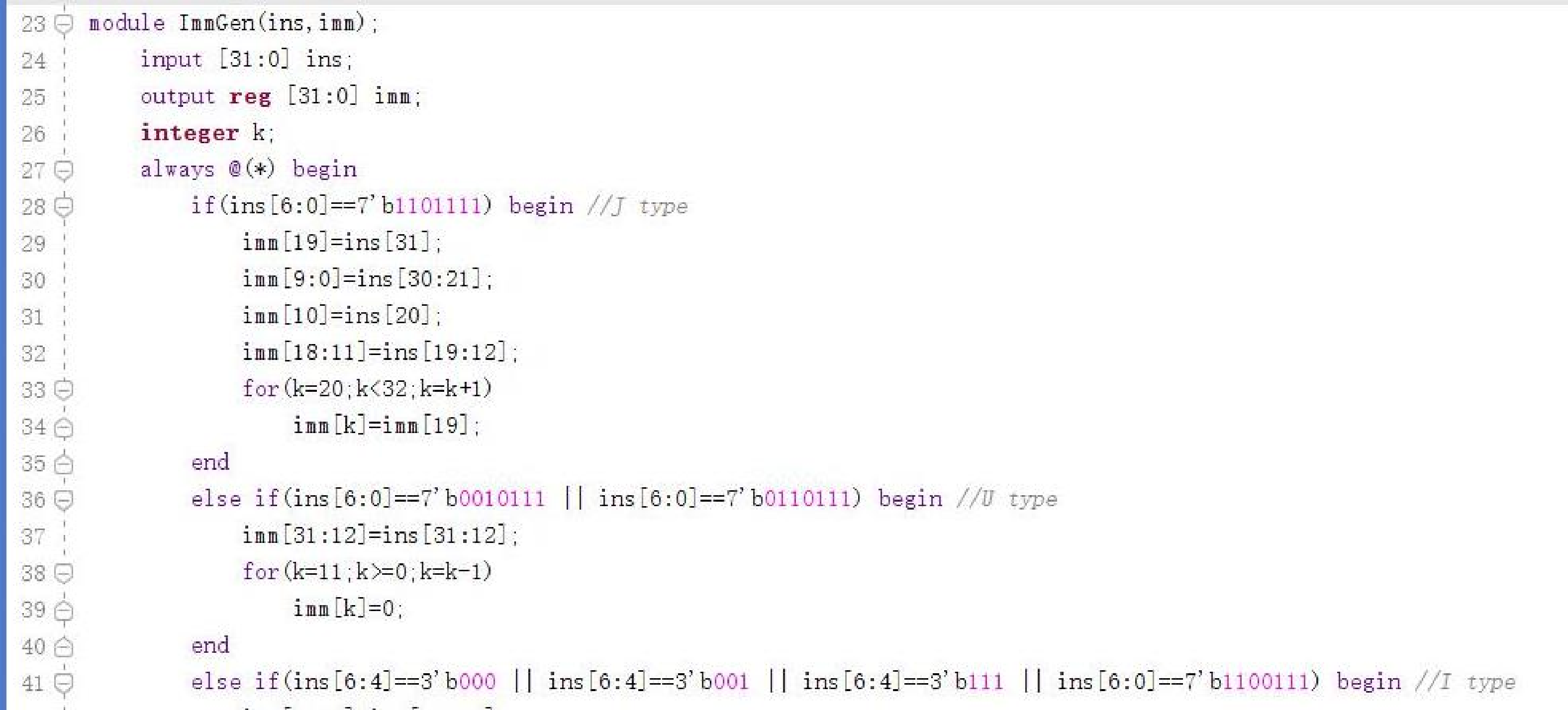
7.
```
0000000 00101 00000 101 01000 1100011
00000001000000000000 00000 1101111
111111111111 00101 000 00101 0010011
0000000 00101 11001 000 11001 0110011
11111111000111111111 00000 1101111
```



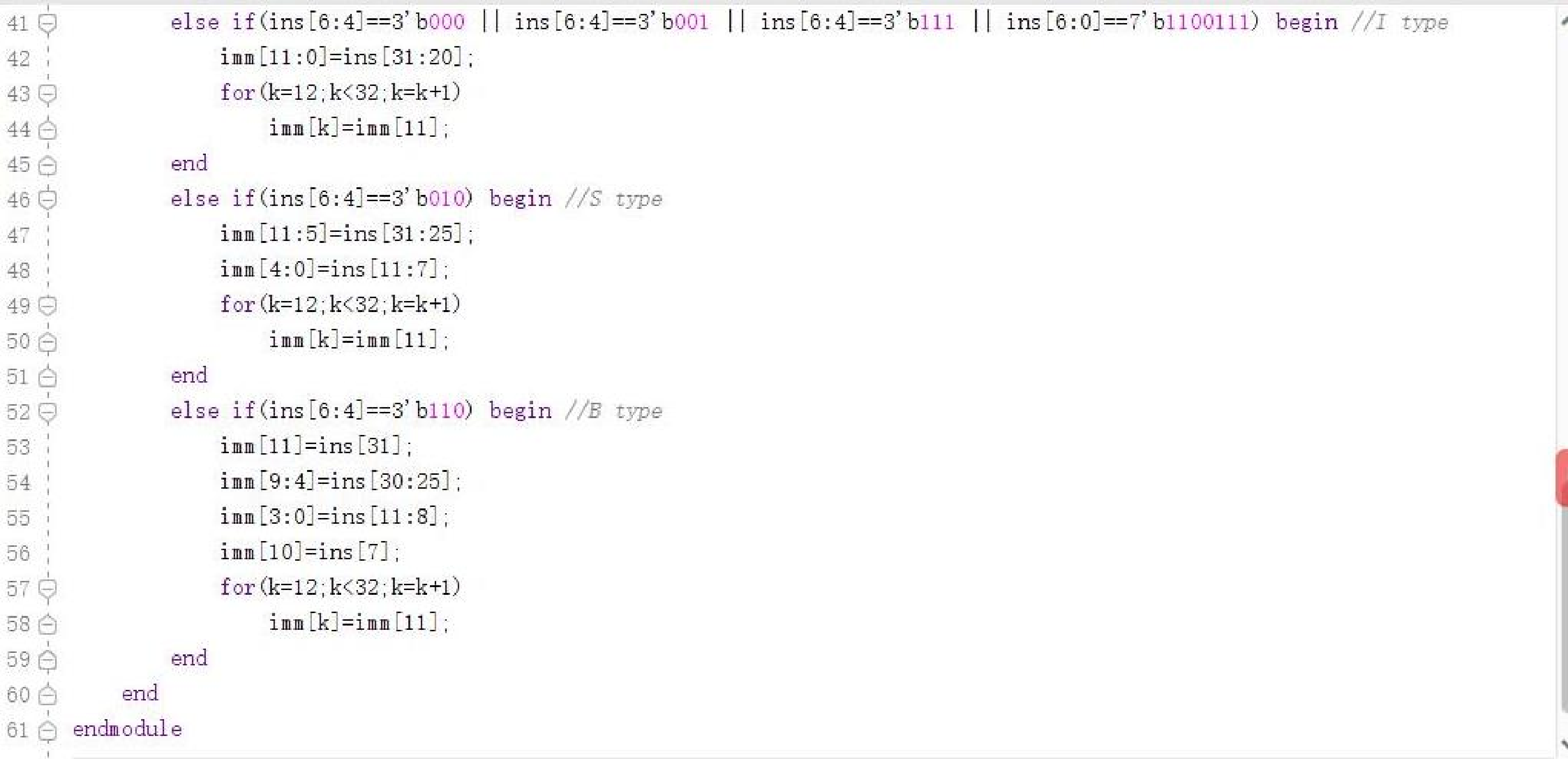
```
module reg_32(read_reg1, read_reg2, write_reg, write_data, read_data1, read_data2, reg_write, clk);
    input [4:0] read_reg1, read_reg2, write_reg;
    input [31:0] write_data;
    input reg_write, clk;
    output reg [31:0] read_data1, read_data2;

    reg [31:0] regs [31:0];

    always @ (posedge clk) begin
        if(reg_write) regs[write_reg]=write_data;
        else begin
            read_data1=regs[read_reg1];
            read_data2=regs[read_reg2];
        end
    end
endmodule
```



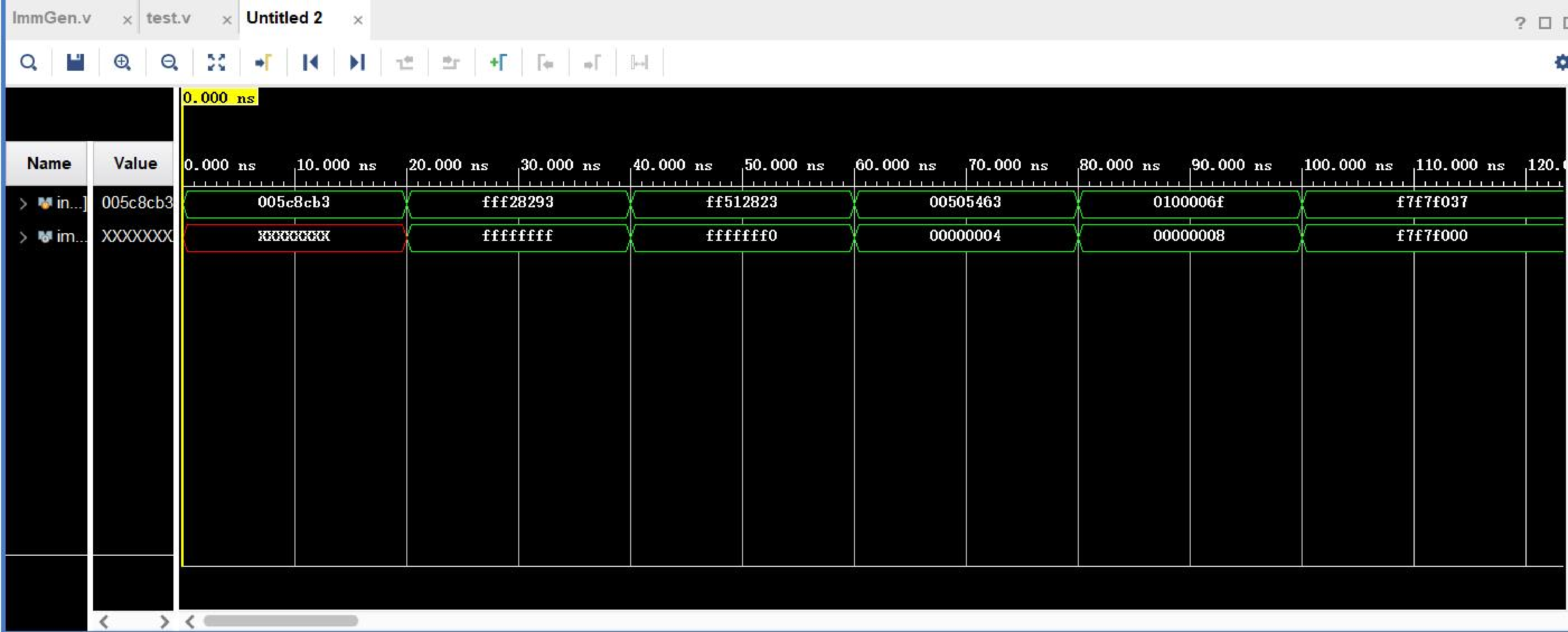
```
Name                Value      Waveform
read_reg1[4:0]      07         XX | 07
read_reg2[4:0]      09         XX | 09
write_reg[4:0]      0b         09 | 0b
write_data[31:0]    7777ffff   ffff7777 | 7777ffff
reg_write           0
clk                 1
read_data1[31:0]    ffff7777   XXXXXXXX | ffff7777
read_data2[31:0]    7777ffff   XXXXXXXX | 7777ffff
half_period[31:0]   00000003   00000003
```



```
module ImmGen(ins,imm);
    input [31:0] ins;
    output reg [31:0] imm;
    integer k;
    always @(*) begin
        if(ins[6:0]==7'b1101111) begin //J type
            imm[19]=ins[31];
            imm[9:0]=ins[30:21];
            imm[10]=ins[20];
            imm[18:11]=ins[19:12];
            for(k=20;k<32;k=k+1)
                imm[k]=imm[19];
        end
        else if(ins[6:0]==7'b0010111 || ins[6:0]==7'b0110111) begin //U type
            imm[31:12]=ins[31:12];
            for(k=11;k>=0;k=k-1)
                imm[k]=0;
        end
        else if(ins[6:4]==3'b000 || ins[6:4]==3'b001 || ins[6:4]==3'b111 || ins[6:0]==7'b1100111) begin //I type
```



```verilog
        else if(ins[6:4]==3'b000 || ins[6:4]==3'b001 || ins[6:4]==3'b111 || ins[6:0]==7'b1100111) begin //I type
            imm[11:0]=ins[31:20];
            for(k=12;k<32;k=k+1)
                imm[k]=imm[11];
        end
        else if(ins[6:4]==3'b010) begin //S type
            imm[11:5]=ins[31:25];
            imm[4:0]=ins[11:7];
            for(k=12;k<32;k=k+1)
                imm[k]=imm[11];
        end
        else if(ins[6:4]==3'b110) begin //B type
            imm[11]=ins[31];
            imm[9:4]=ins[30:25];
            imm[3:0]=ins[11:8];
            imm[10]=ins[7];
            for(k=12;k<32;k=k+1)
                imm[k]=imm[11];
        end
    end
endmodule
```



```
0.000 ns

       0.000 ns  10.000 ns  20.000 ns  30.000 ns  40.000 ns  50.000 ns  60.000 ns  70.000 ns  80.000 ns  90.000 ns  100.000 ns  110.000 ns  120.
in...]  005c8cb3   005c8cb3    fff28293    ff512823    00505463    0100006f    f7f7f037
im...   XXXXXXX    XXXXXXXX    ffffffff    fffffff0    00000004    00000008    f7f7f000
```