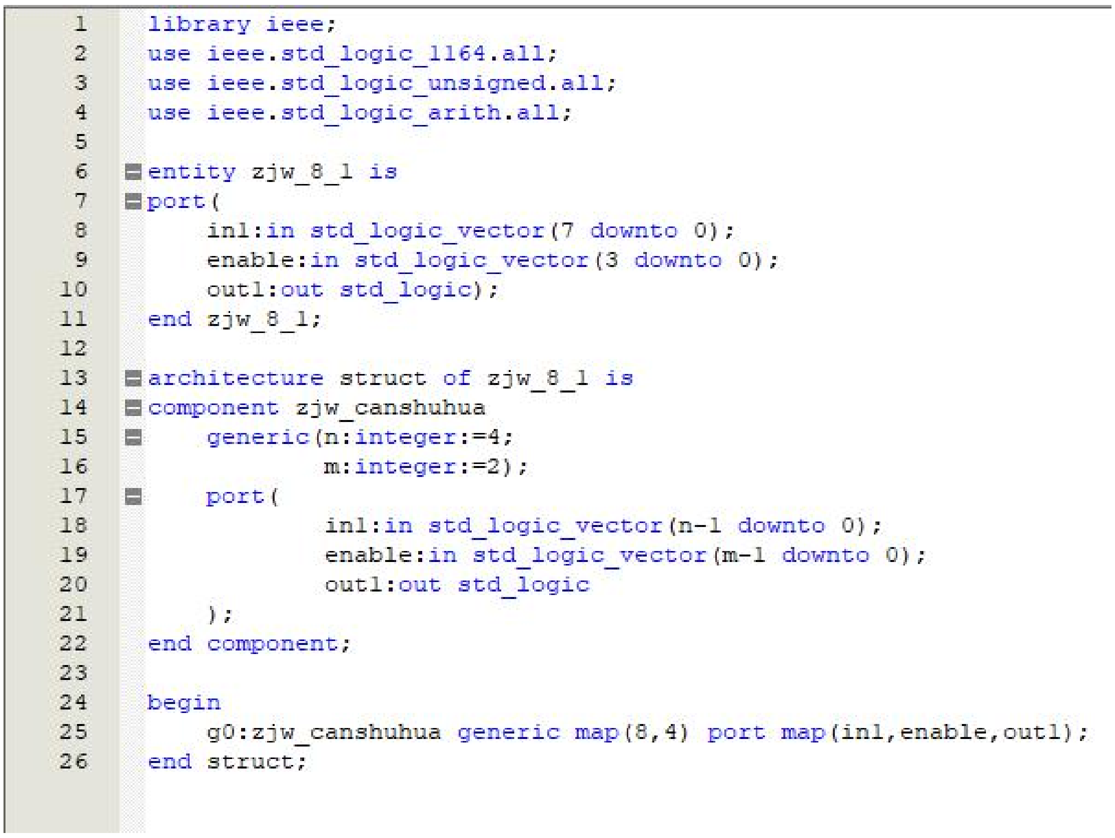
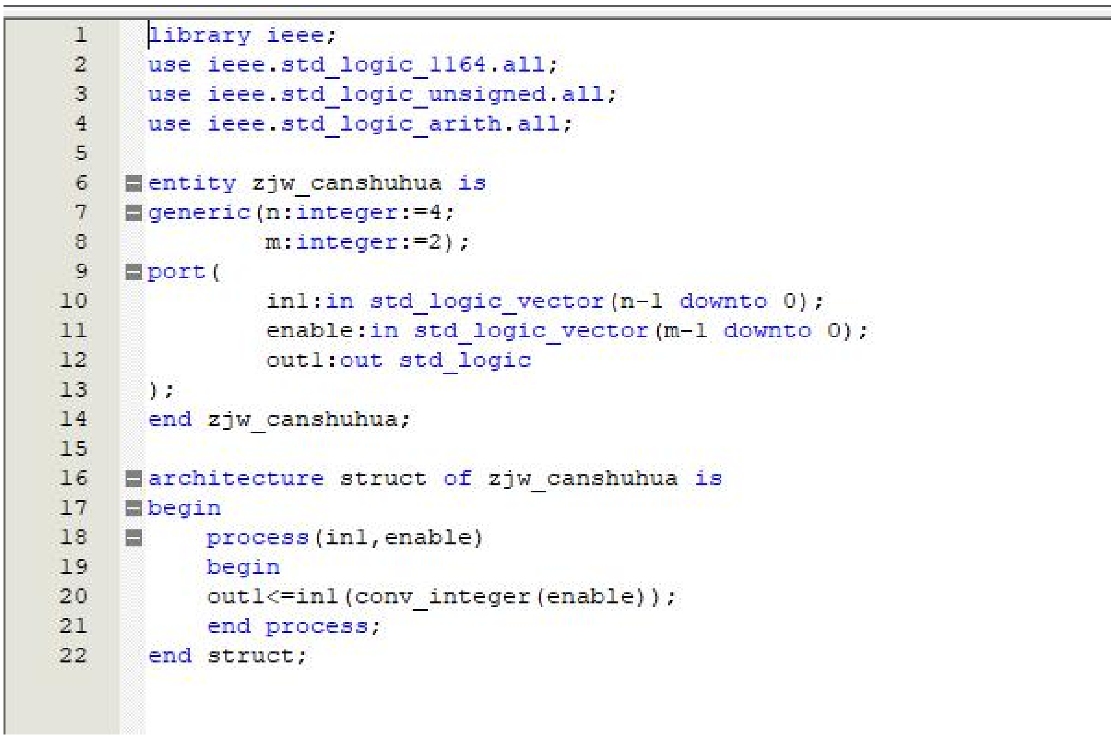
模型机的 8 重 3-1 多路复用器



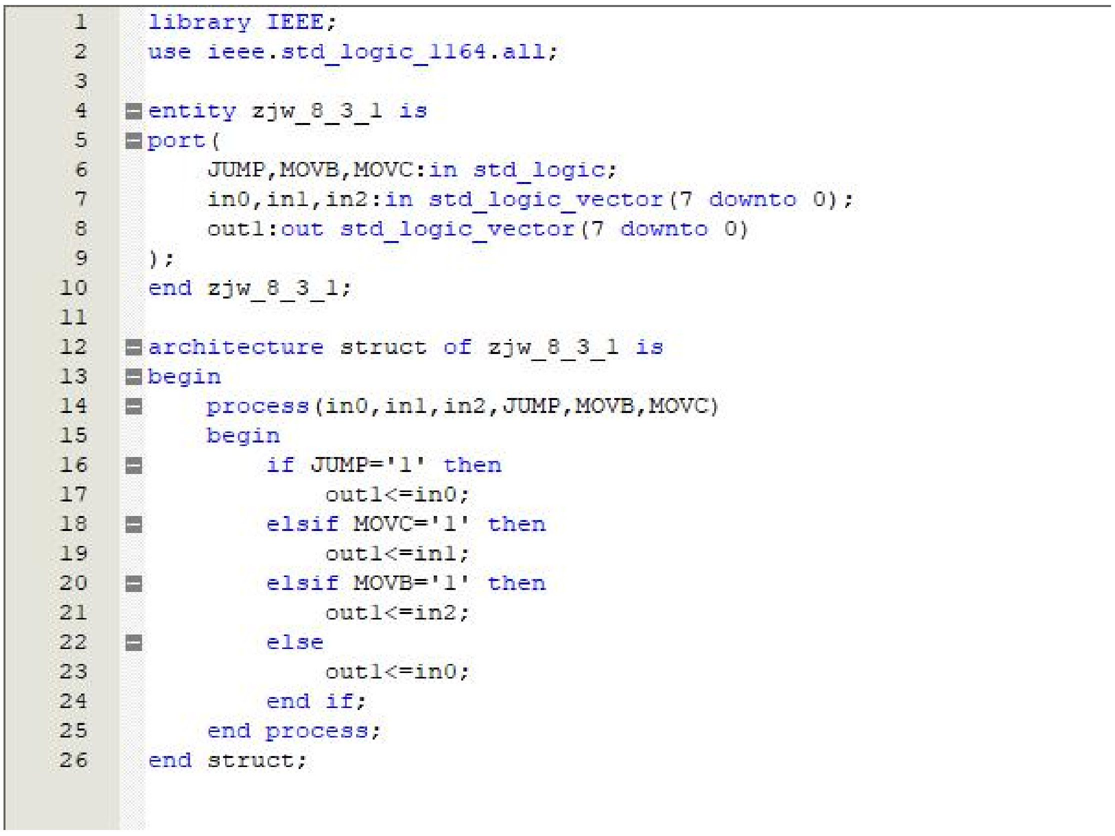
```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;

entity zjw_8_1 is
port(
    in1:in std_logic_vector(7 downto 0);
    enable:in std_logic_vector(3 downto 0);
    out1:out std_logic);
end zjw_8_1;

architecture struct of zjw_8_1 is
component zjw_canshuhua
    generic(n:integer:=4;
            m:integer:=2);
    port(
            in1:in std_logic_vector(n-1 downto 0);
            enable:in std_logic_vector(m-1 downto 0);
            out1:out std_logic
    );
end component;

begin
    g0:zjw_canshuhua generic map(8,4) port map(in1,enable,out1);
end struct;
```

参数化多路复用器



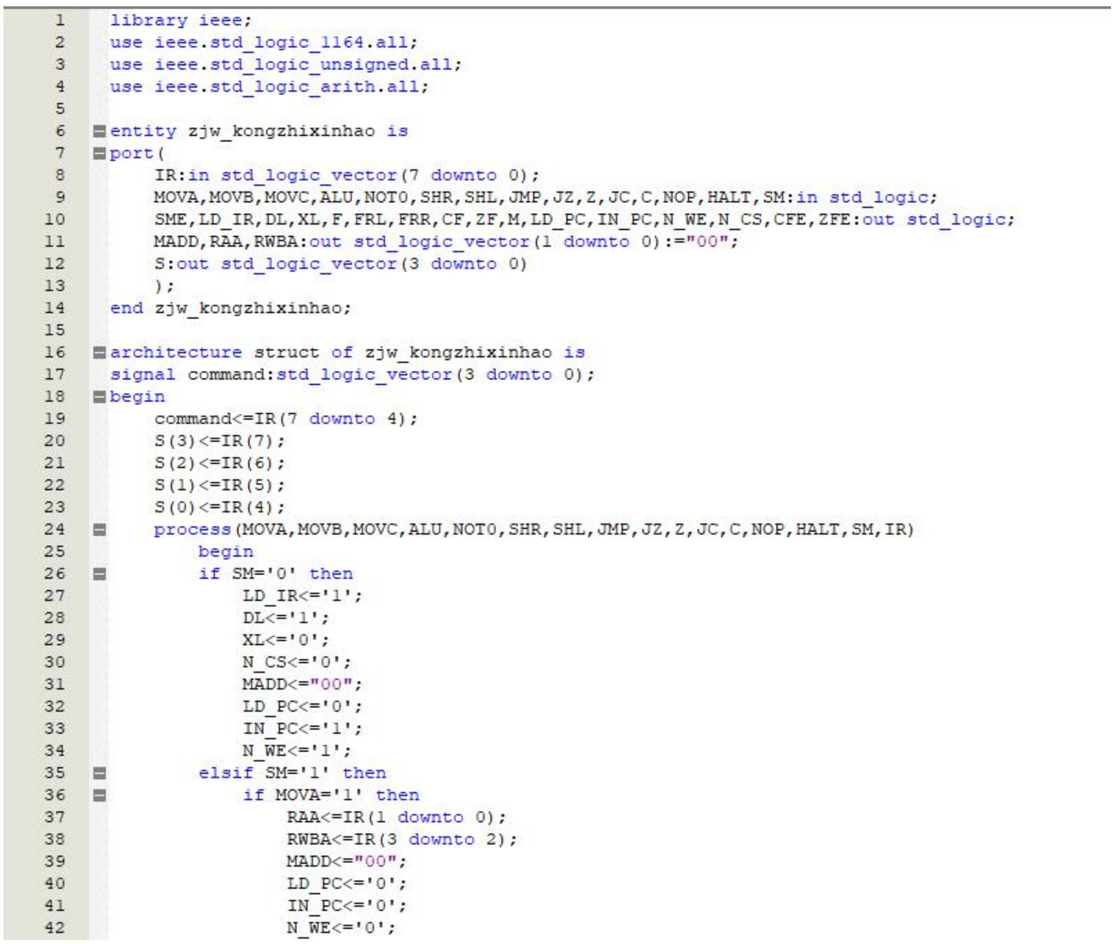
```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;

entity zjw_canshuhua is
generic(n:integer:=4;
        m:integer:=2);
port(
        in1:in std_logic_vector(n-1 downto 0);
        enable:in std_logic_vector(m-1 downto 0);
        out1:out std_logic
);
end zjw_canshuhua;

architecture struct of zjw_canshuhua is
begin
    process(in1,enable)
    begin
    out1<=in1(conv_integer(enable));
    end process;
end struct;
```

参数化多路复用器定制为 8-1 多路复用器

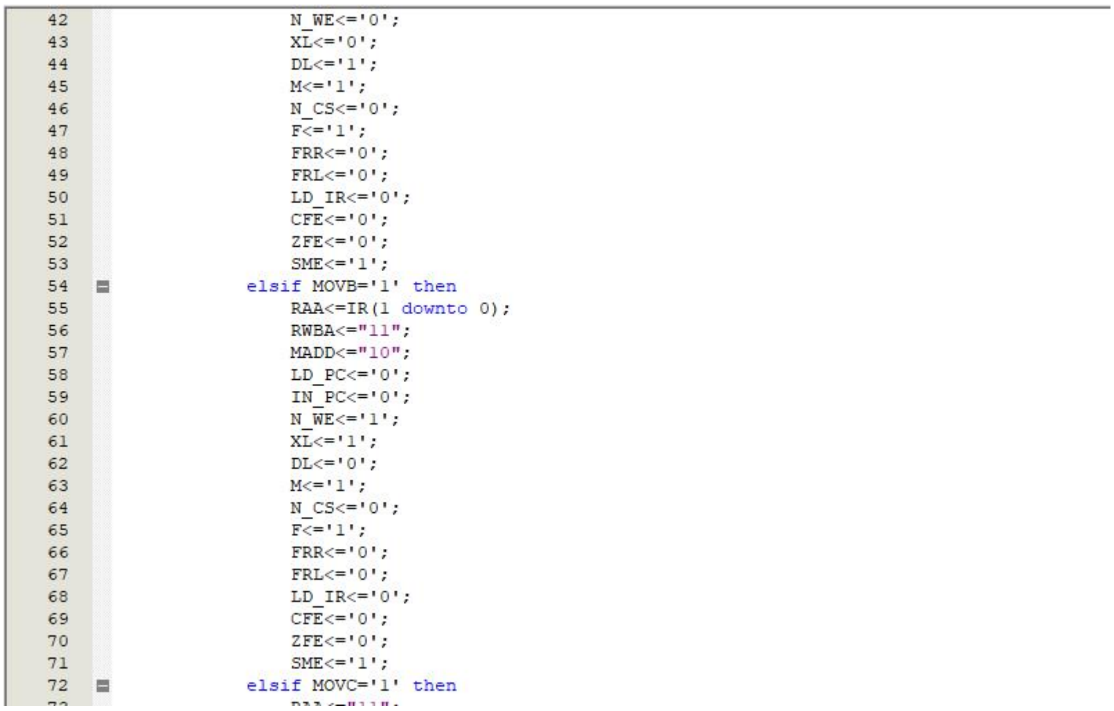


```
library IEEE;
use ieee.std_logic_1164.all;

entity zjw_8_3_1 is
port(
    JUMP,MOVB,MOVC:in std_logic;
    in0,in1,in2:in std_logic_vector(7 downto 0);
    out1:out std_logic_vector(7 downto 0)
);
end zjw_8_3_1;

architecture struct of zjw_8_3_1 is
begin
    process(in0,in1,in2,JUMP,MOVB,MOVC)
    begin
        if JUMP='1' then
            out1<=in0;
        elsif MOVC='1' then
            out1<=in1;
        elsif MOVB='1' then
            out1<=in2;
        else
            out1<=in0;
        end if;
    end process;
end struct;
```

模型机的控制信号产生逻辑

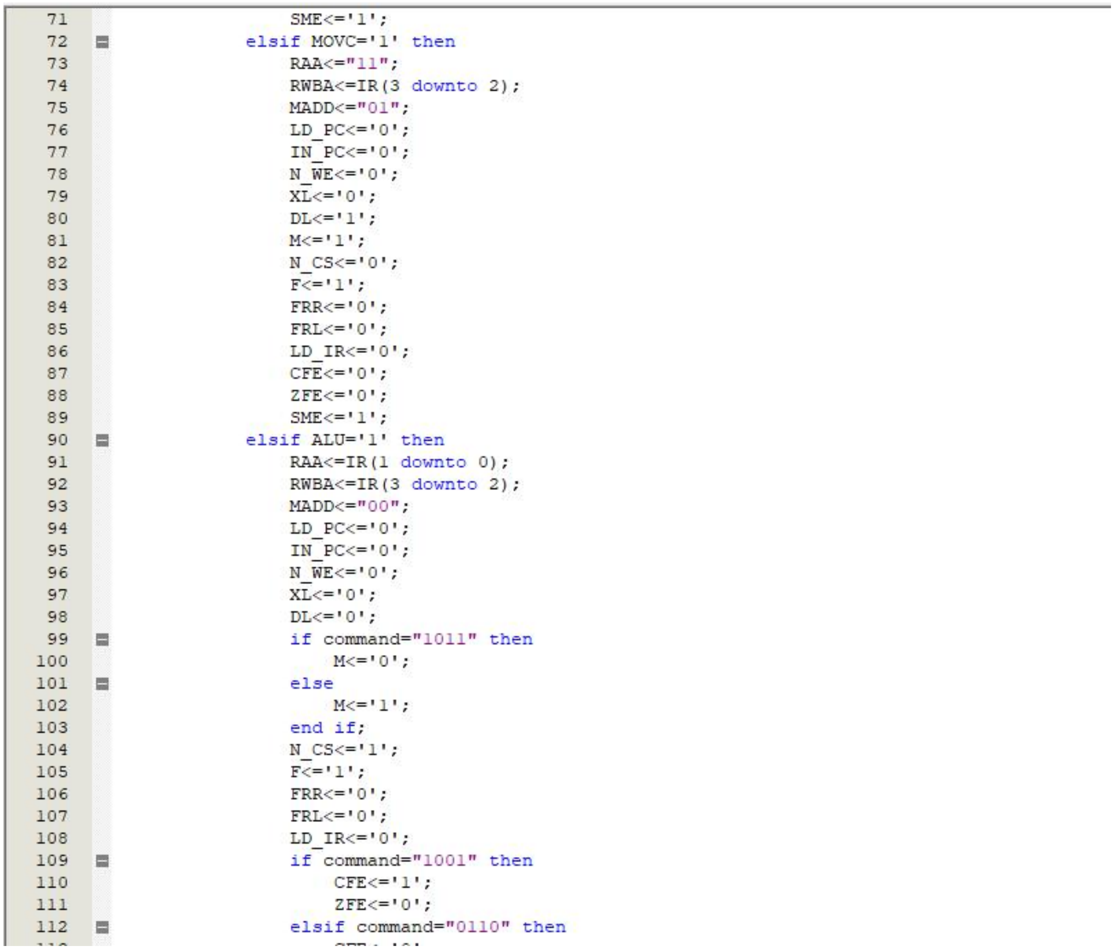


```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;

entity zjw_kongzhixinhao is
port(
    IR:in std_logic_vector(7 downto 0);
    MOVA,MOVB,MOVC,ALU,NOT0,SHR,SHL,JMP,JZ,Z,JC,C,NOP,HALT,SM:in std_logic;
    SME,LD_IR,DL,XL,F,FRL,FRR,CF,ZF,M,LD_PC,IN_PC,N_WE,N_CS,CFE,ZFE:out std_logic;
    MADD,RAA,RWBA:out std_logic_vector(1 downto 0):="00";
    S:out std_logic_vector(3 downto 0)
    );
end zjw_kongzhixinhao;

architecture struct of zjw_kongzhixinhao is
signal command:std_logic_vector(3 downto 0);
begin
    command<=IR(7 downto 4);
    S(3)<=IR(7);
    S(2)<=IR(6);
    S(1)<=IR(5);
    S(0)<=IR(4);
    process(MOVA,MOVB,MOVC,ALU,NOT0,SHR,SHL,JMP,JZ,Z,JC,C,NOP,HALT,SM,IR)
        begin
        if SM='0' then
            LD_IR<='1';
            DL<='1';
            XL<='0';
            N_CS<='0';
            MADD<="00";
            LD_PC<='0';
            IN_PC<='1';
            N_WE<='1';
        elsif SM='1' then
            if MOVA='1' then
                RAA<=IR(1 downto 0);
                RWBA<=IR(3 downto 2);
                MADD<="00";
                LD_PC<='0';
                IN_PC<='0';
                N_WE<='0';
```

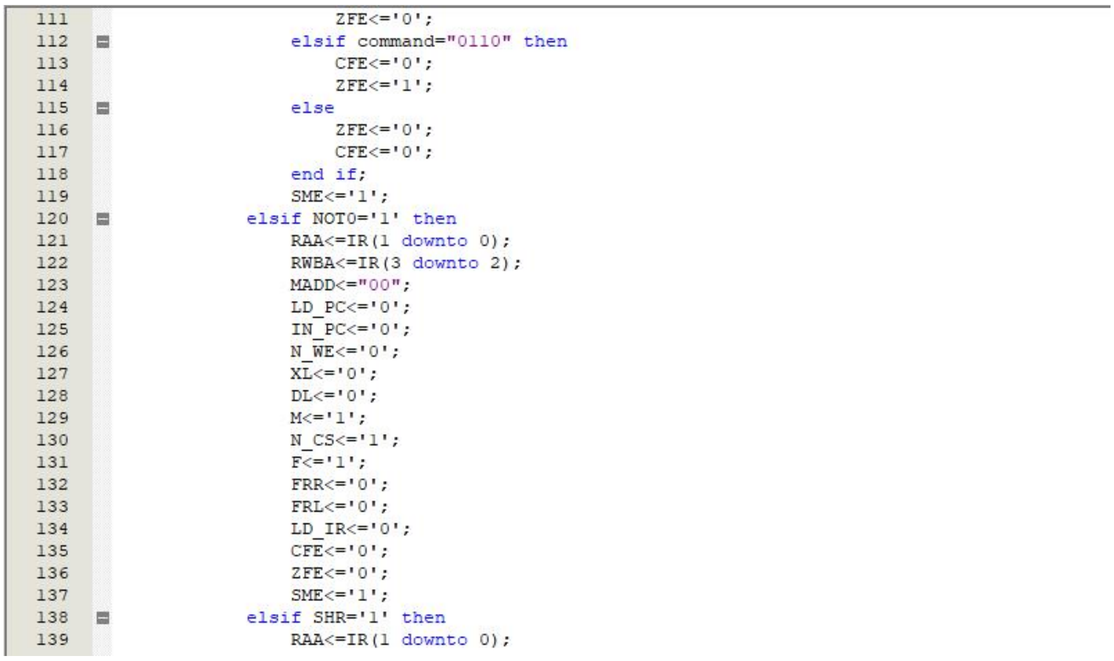


`N_WE<='0';`

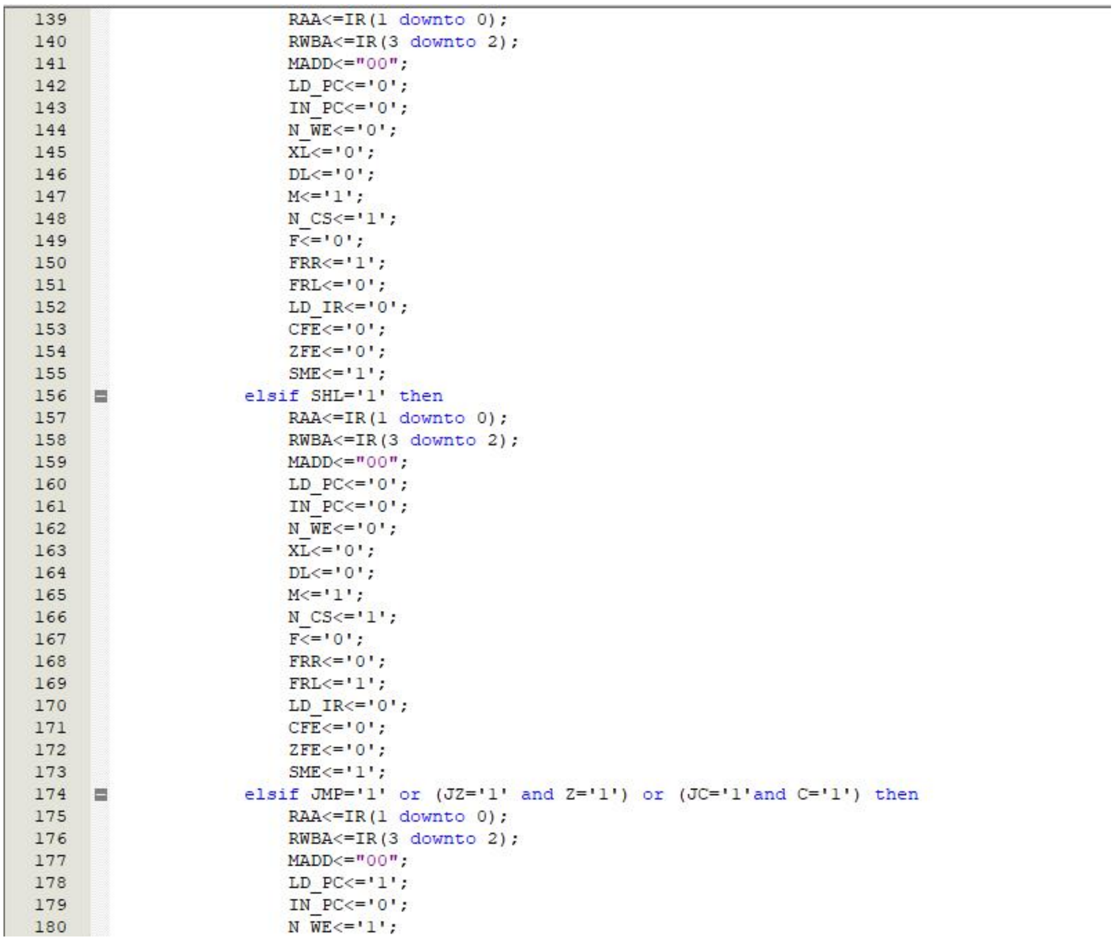
```vhdl
                XL<='0';
                DL<='1';
                M<='1';
                N_CS<='0';
                F<='1';
                FRR<='0';
                FRL<='0';
                LD_IR<='0';
                CFE<='0';
                ZFE<='0';
                SME<='1';
            elsif MOVB='1' then
                RAA<=IR(1 downto 0);
                RWBA<="11";
                MADD<="10";
                LD_PC<='0';
                IN_PC<='0';
                N_WE<='1';
                XL<='1';
                DL<='0';
                M<='1';
                N_CS<='0';
                F<='1';
                FRR<='0';
                FRL<='0';
                LD_IR<='0';
                CFE<='0';
                ZFE<='0';
                SME<='1';
            elsif MOVC='1' then
```



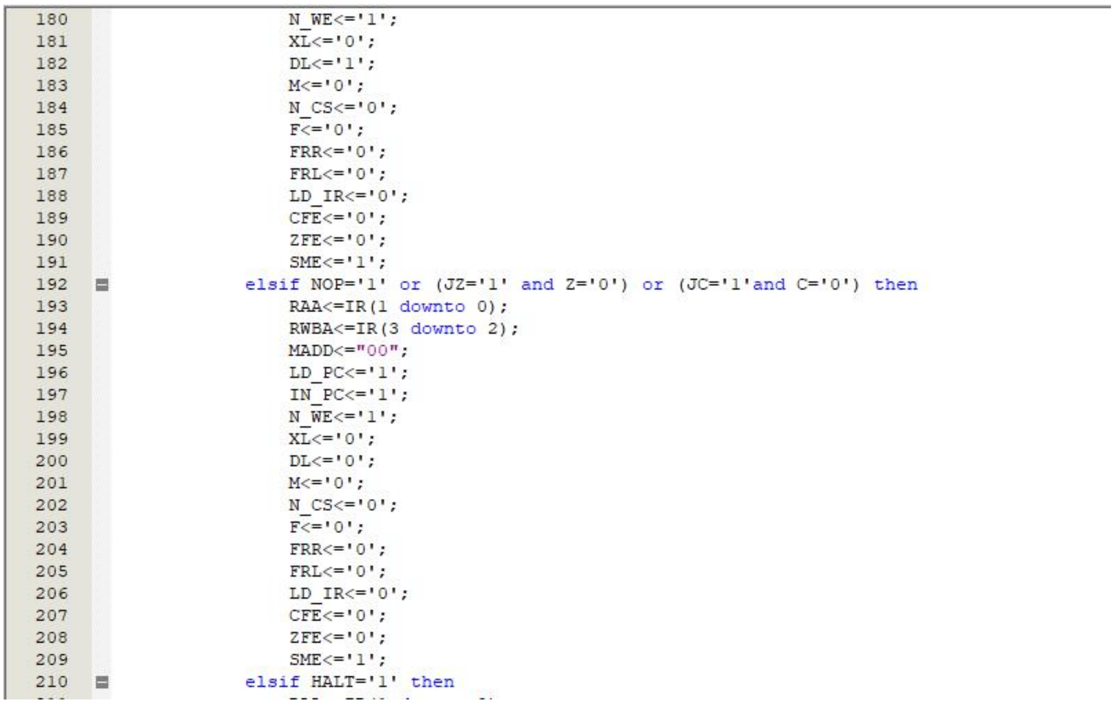
```
        SME<='1';
    elsif MOVC='1' then
        RAA<="11";
        RWBA<=IR(3 downto 2);
        MADD<="01";
        LD_PC<='0';
        IN_PC<='0';
        N_WE<='0';
        XL<='0';
        DL<='1';
        M<='1';
        N_CS<='0';
        F<='1';
        FRR<='0';
        FRL<='0';
        LD_IR<='0';
        CFE<='0';
        ZFE<='0';
        SME<='1';
    elsif ALU='1' then
        RAA<=IR(1 downto 0);
        RWBA<=IR(3 downto 2);
        MADD<="00";
        LD_PC<='0';
        IN_PC<='0';
        N_WE<='0';
        XL<='0';
        DL<='0';
        if command="1011" then
            M<='0';
        else
            M<='1';
        end if;
        N_CS<='1';
        F<='1';
        FRR<='0';
        FRL<='0';
        LD_IR<='0';
        if command="1001" then
            CFE<='1';
            ZFE<='0';
        elsif command="0110" then
```



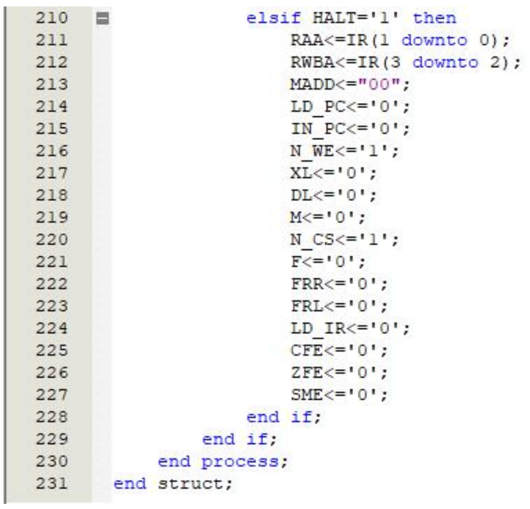
```
            ZFE<='0';
        elsif command="0110" then
            CFE<='0';
            ZFE<='1';
        else
            ZFE<='0';
            CFE<='0';
        end if;
        SME<='1';
    elsif NOT0='1' then
        RAA<=IR(1 downto 0);
        RWBA<=IR(3 downto 2);
        MADD<="00";
        LD_PC<='0';
        IN_PC<='0';
        N_WE<='0';
        XL<='0';
        DL<='0';
        M<='1';
        N_CS<='1';
        F<='1';
        FRR<='0';
        FRL<='0';
        LD_IR<='0';
        CFE<='0';
        ZFE<='0';
        SME<='1';
    elsif SHR='1' then
        RAA<=IR(1 downto 0);
```



```
                RAA<=IR(1 downto 0);
                RWBA<=IR(3 downto 2);
                MADD<="00";
                LD_PC<='0';
                IN_PC<='0';
                N_WE<='0';
                XL<='0';
                DL<='0';
                M<='1';
                N_CS<='1';
                F<='0';
                FRR<='1';
                FRL<='0';
                LD_IR<='0';
                CFE<='0';
                ZFE<='0';
                SME<='1';
            elsif SHL='1' then
                RAA<=IR(1 downto 0);
                RWBA<=IR(3 downto 2);
                MADD<="00";
                LD_PC<='0';
                IN_PC<='0';
                N_WE<='0';
                XL<='0';
                DL<='0';
                M<='1';
                N_CS<='1';
                F<='0';
                FRR<='0';
                FRL<='1';
                LD_IR<='0';
                CFE<='0';
                ZFE<='0';
                SME<='1';
            elsif JMP='1' or (JZ='1' and Z='1') or (JC='1'and C='1') then
                RAA<=IR(1 downto 0);
                RWBA<=IR(3 downto 2);
                MADD<="00";
                LD_PC<='1';
                IN_PC<='0';
                N_WE<='1';
```



```
                N_WE<='1';
                XL<='0';
                DL<='1';
                M<='0';
                N_CS<='0';
                F<='0';
                FRR<='0';
                FRL<='0';
                LD_IR<='0';
                CFE<='0';
                ZFE<='0';
                SME<='1';
            elsif NOP='1' or (JZ='1' and Z='0') or (JC='1'and C='0') then
                RAA<=IR(1 downto 0);
                RWBA<=IR(3 downto 2);
                MADD<="00";
                LD_PC<='1';
                IN_PC<='1';
                N_WE<='1';
                XL<='0';
                DL<='0';
                M<='0';
                N_CS<='0';
                F<='0';
                FRR<='0';
                FRL<='0';
                LD_IR<='0';
                CFE<='0';
                ZFE<='0';
                SME<='1';
            elsif HALT='1' then
```



```vhdl
            elsif HALT='1' then
                RAA<=IR(1 downto 0);
                RWBA<=IR(3 downto 2);
                MADD<="00";
                LD_PC<='0';
                IN_PC<='0';
                N_WE<='1';
                XL<='0';
                DL<='0';
                M<='0';
                N_CS<='1';
                F<='0';
                FRR<='0';
                FRL<='0';
                LD_IR<='0';
                CFE<='0';
                ZFE<='0';
                SME<='0';
            end if;
        end if;
    end process;
end struct;
```