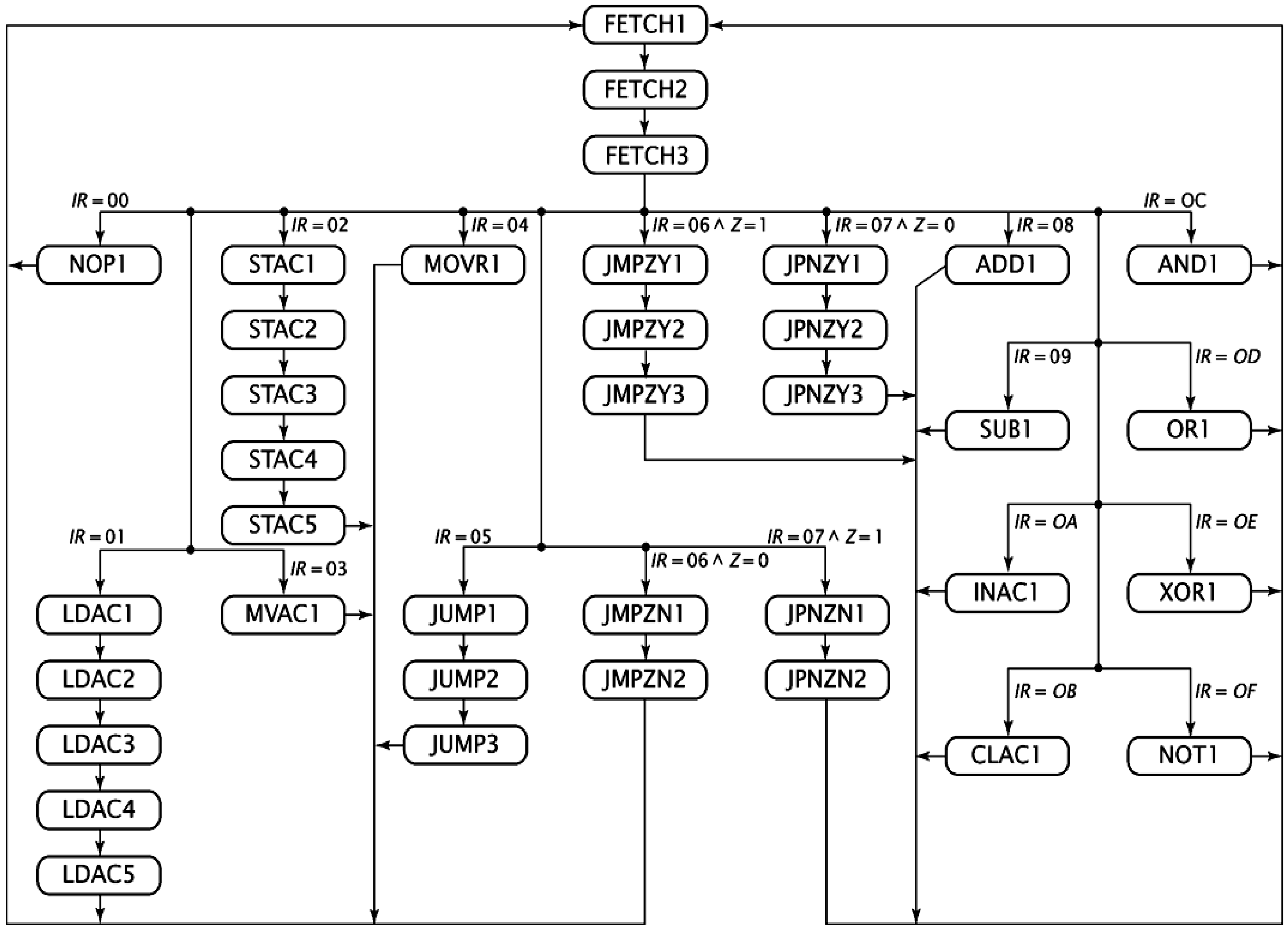
相对简单的CPU设计

班级：智能1602

姓名：马琛迎

学号：201607030227

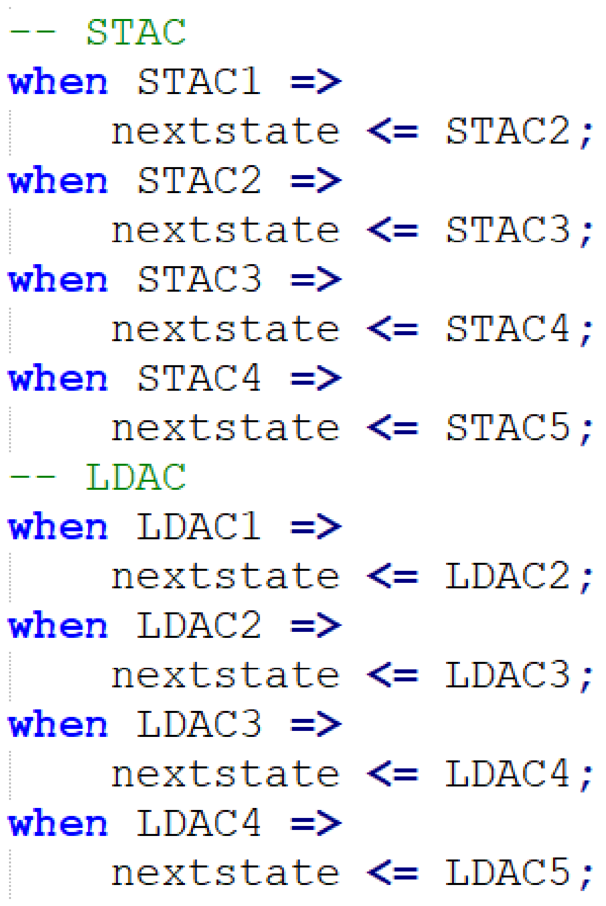
相对简单的CPU设计


FETCH1
FETCH2
FETCH3
IR = 00
IR = 02
IR = 04
IR = 06 ∧ Z = 1
IR = 07 ∧ Z = 0
IR = 08
IR = 0C
NOP1
STAC1
MOVR1
JMPZY1
JPNZY1
ADD1
AND1
STAC2
JMPZY2
JPNZY2
IR = 09
IR = OD
STAC3
JMPZY3
JPNZY3
SUB1
OR1
STAC4
STAC5
IR = 01
IR = 05
IR = 07 ∧ Z = 1
IR = 06 ∧ Z = 0
IR = 03
IR = OA
IR = OE
LDAC1
MVAC1
JUMP1
JMPZN1
JPNZN1
INAC1
XOR1
LDAC2
JUMP2
JMPZN2
JPNZN2
IR = OB
IR = OF
LDAC3
JUMP3
CLAC1
NOT1
LDAC4
LDAC5

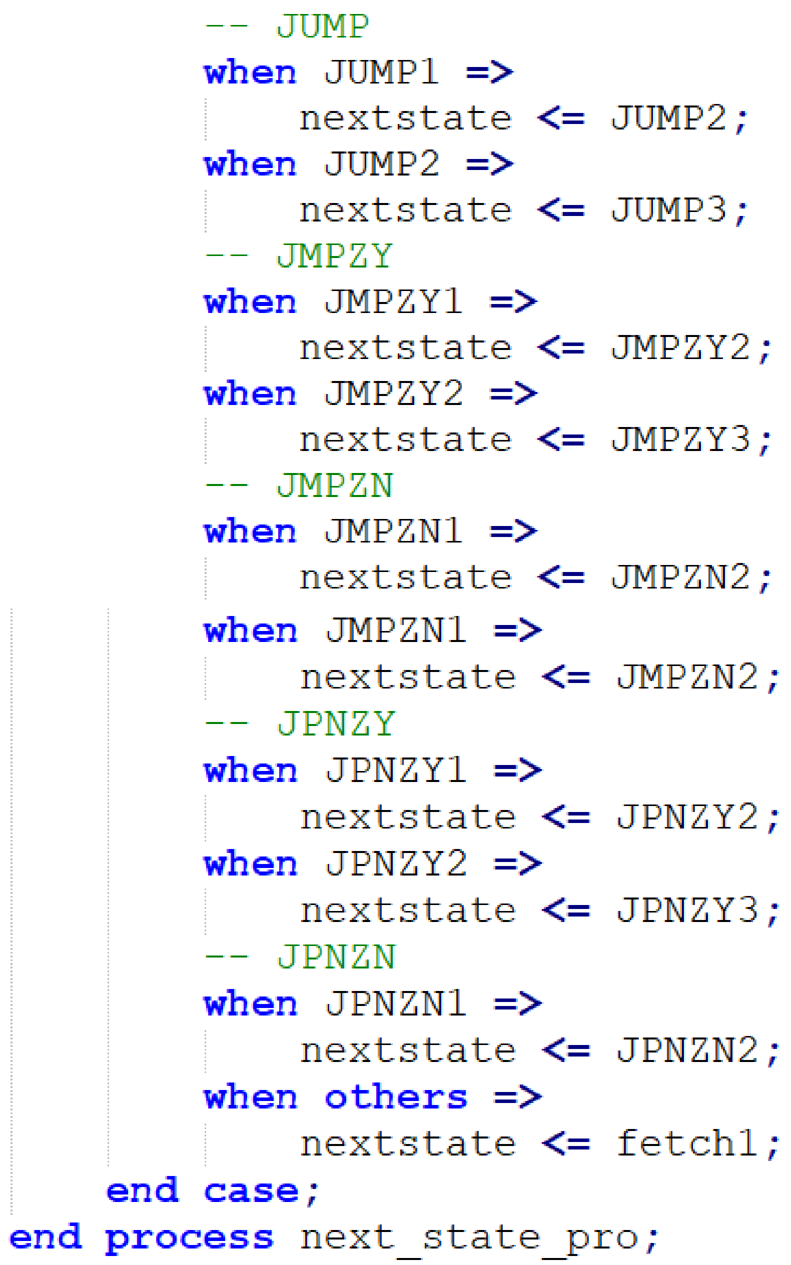
相对简单的CPU设计

```vhdl
-- State transition
next_state_pro: process(state,ir,z)
begin
    case state is
        when fetch1 =>
            nextstate <= fetch2;
        when fetch2 =>
            nextstate <= fetch3;
        when fetch3 =>
            case ir is
                -- More than one state
                when RSNOP =>
                    nextstate <= NOP1;
                when RSLDAC =>
                    nextstate <= LDAC1;
                when RSSTAC =>
                    nextstate <= STAC1;
                when RSMVAC =>
                    nextstate <= MVAC1;
                when RSMOVR =>
                    nextstate <= MOVR1;
                when RSJUMP =>
                    nextstate <= JUMP1;
                when RSJMPZ =>
                    --if(z='0') then
                    --  nextstate <= JMPZY1;
                    --else if(z='1') then
                    --  nextstate <= JMPZN1;
                    --end if;
                    case z is
                        when '0' =>
                            nextstate <= JMPZY1;
                        when '1' =>
                            nextstate <= JMPZN1;
                    end case;
                when RSJPNZ =>
                    --if(z='1') then
                    --  nextstate <= JPNZY1;
                    --else if(z='0') then
                    --  nextstate <= JPNZN1;
                    --end if;
                    case z is
                        when '0' =>
                            nextstate <= JPNZN1;
                        when '1' =>
                            nextstate <= JPNZY1;
                    end case;
                -- one state
                when RSADD =>
                    nextstate <= ADD1;
                when RSSUB =>
                    nextstate <= SUB1;
                when RSINAC =>
                    nextstate <= INAC1;
                when RSCLAC =>
                    nextstate <= CLAC1;
                when RSAND =>
                    nextstate <= AND1;
                when RSOR =>
                    nextstate <= OR1;
                when RSXOR =>
                    nextstate <= XOR1;
                when RSNOT =>
                    nextstate <= NOT1;
                when others =>
                    nextstate <= fetch1;
            end case;
```

相对简单的CPU设计



```
-- STAC
when STAC1 =>
    nextstate <= STAC2;
when STAC2 =>
    nextstate <= STAC3;
when STAC3 =>
    nextstate <= STAC4;
when STAC4 =>
    nextstate <= STAC5;
-- LDAC
when LDAC1 =>
    nextstate <= LDAC2;
when LDAC2 =>
    nextstate <= LDAC3;
when LDAC3 =>
    nextstate <= LDAC4;
when LDAC4 =>
    nextstate <= LDAC5;
```



```
        -- JUMP
        when JUMP1 =>
            nextstate <= JUMP2;
        when JUMP2 =>
            nextstate <= JUMP3;
        -- JMPZY
        when JMPZY1 =>
            nextstate <= JMPZY2;
        when JMPZY2 =>
            nextstate <= JMPZY3;
        -- JMPZN
        when JMPZN1 =>
            nextstate <= JMPZN2;
        when JMPZN1 =>
            nextstate <= JMPZN2;
        -- JPNZY
        when JPNZY1 =>
            nextstate <= JPNZY2;
        when JPNZY2 =>
            nextstate <= JPNZY3;
        -- JPNZN
        when JPNZN1 =>
            nextstate <= JPNZN2;
        when others =>
            nextstate <= fetch1;
    end case;
end process next_state_pro;
```

相对简单的CPU设计

LDAC1:	DR←M，PC←PC＋1，AR←AR＋1
LDAC2:	TR←DR，DR←M，PC←PC＋1
LDAC3:	AR←DR，TR
LDAC4:	DR←M
LDAC5:	AC←DR

STAC1:	DR←M，PC←PC＋1，AR←AR＋1
STAC2:	TR←DR，DR←M，PC←PC＋1
STAC3:	AR←DR，TR
STAC4:	DR←AC
STAC5:	M←DR

MVAC1:	R←AC
MOVR1:	AC←R

JUMP1:	DR←M，AR←AR+1
JUMP2:	TR←DR，DR←M
JUMP3:	PC←DR，TR

JMPZY1:	DR←M，AR←AR＋1
JMPZY2:	TR←DR，DR←M
JMPZY3:	PC←DR，TR

JMPZN1:	PC←PC＋1
JMPZN2:	PC←PC＋1

相对简单的CPU设计

JPNZY1: DR←M，AR←AR＋1

JPNZY2: TR←DR，DR←M

JPNZY3: PC←DR，TR

JPNZN1: PC←PC＋1

JPNZN2: PC←PC＋1

ADD1: AC←AC＋R，IF（AC＋R＝0）THEN Z←1 ELSE Z←0

SUB1: AC←AC－R，IF（AC－R＝0）THEN Z←1 ELSE Z←0

INAC1: AC←AC＋1，IF（AC＋1＝0）THEN Z←1 ELSE Z←0

CLAC1: AC←0，Z←1

AND1: AC←AC∧R，IF（AC∧R＝0）THEN Z←1 ELSE Z←0

OR1: AC←AC∨R，IF（AC∨R＝0）THEN Z←1 ELSE Z←0

XOR1: AC←AC⊕R，IF（AC⊕R＝0）THEN Z←1 ELSE Z←0

NOT1: AC←AC'，IF（AC'＝0）THEN Z←1 ELSE Z←0

相对简单的CPU设计

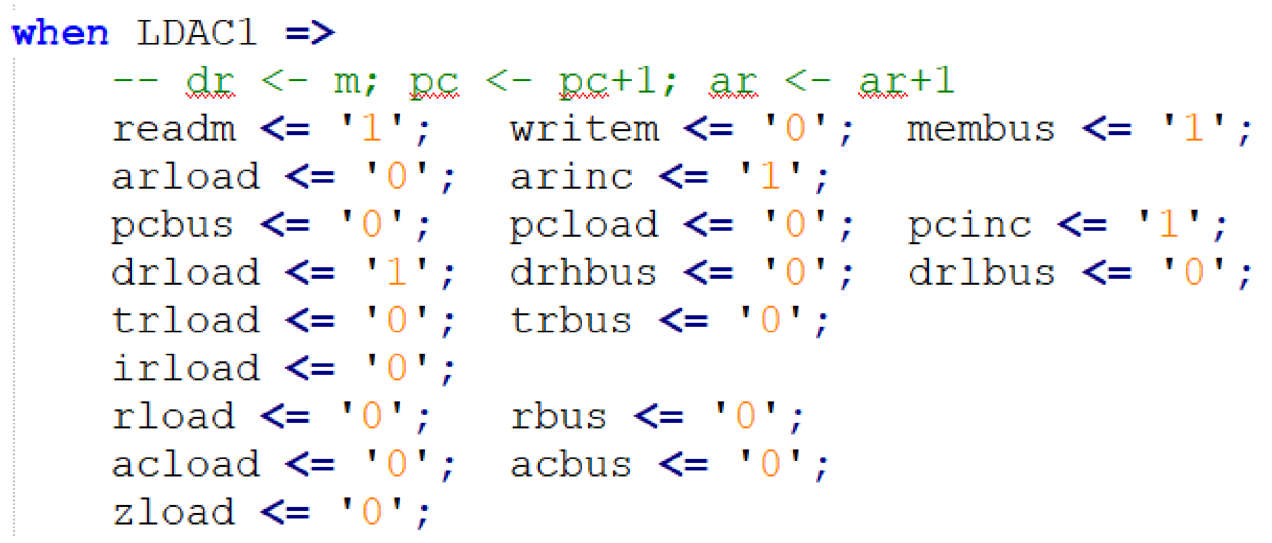
LDAC1: DR←M，PC←PC＋1，AR←AR＋1

LDAC2: TR←DR，DR←M，PC←PC＋1

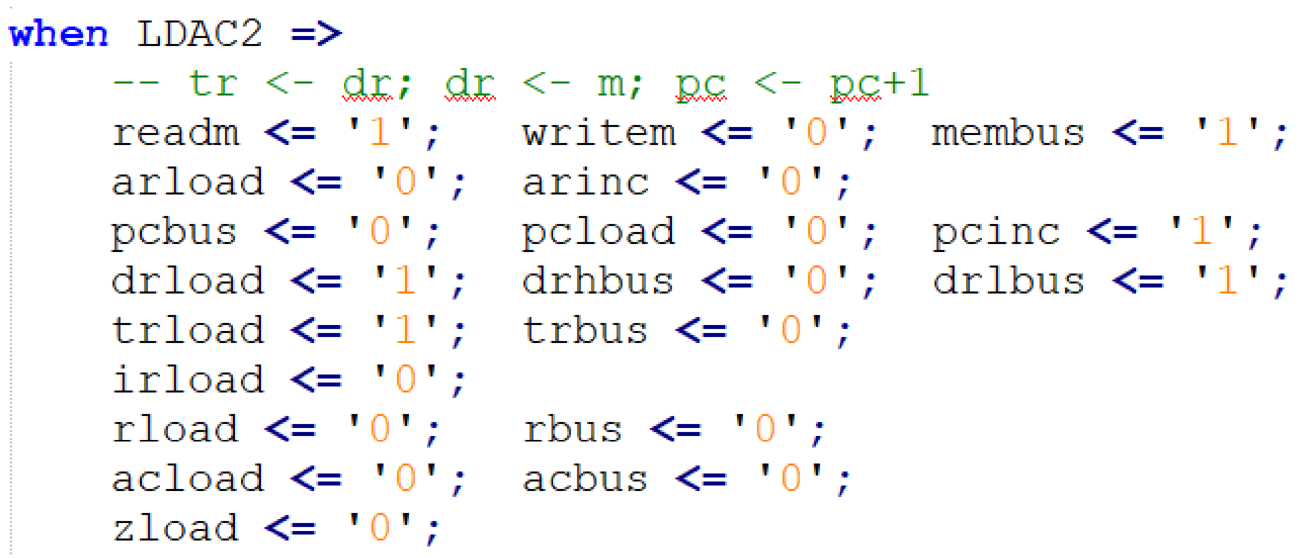
LDAC3: AR←DR，TR

LDAC4: DR←M

LDAC5: AC←DR

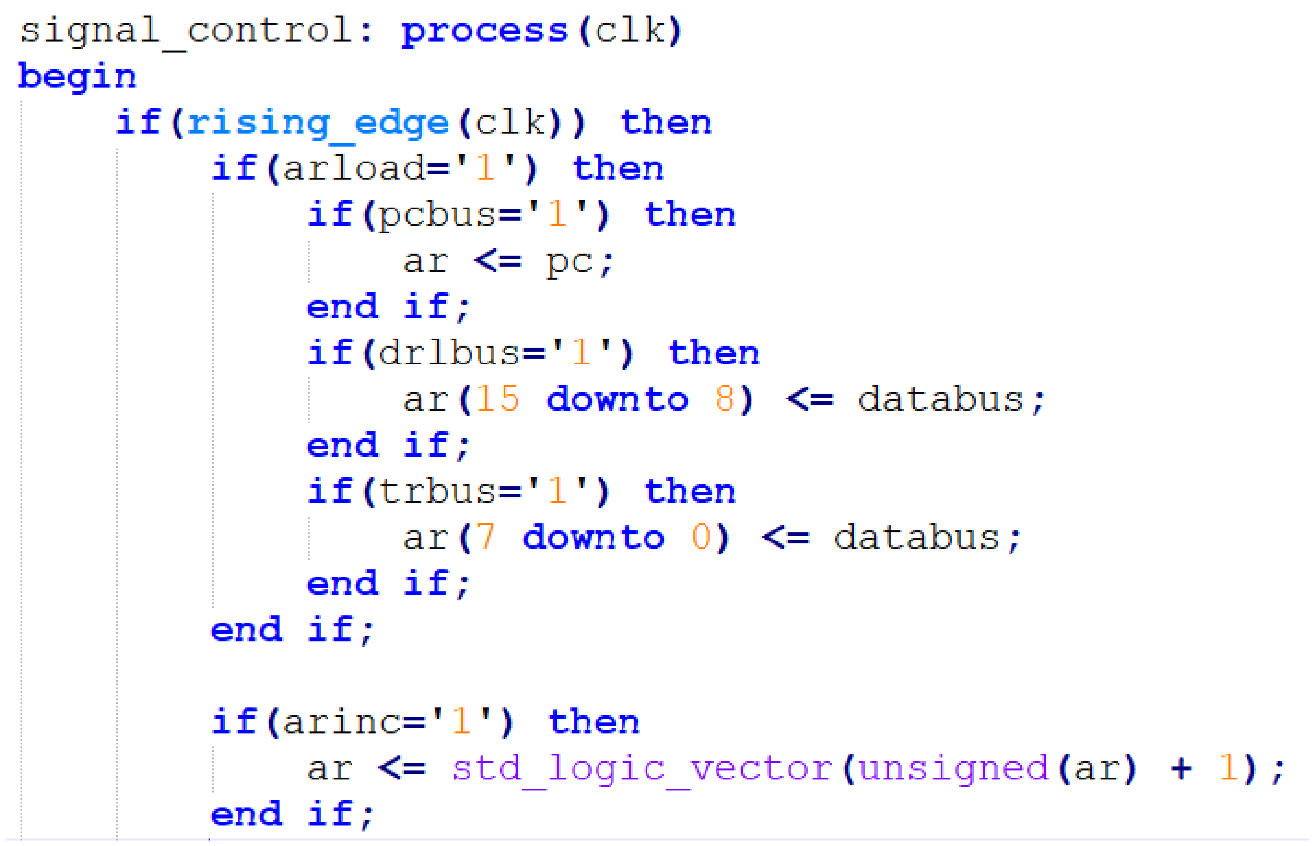


```
when LDAC1 =>
    -- dr <- m; pc <- pc+1; ar <- ar+1
    readm <= '1';   writem <= '0';  membus <= '1';
    arload <= '0';  arinc <= '1';
    pcbus <= '0';   pcload <= '0';  pcinc <= '1';
    drload <= '1';  drhbus <= '0';  drlbus <= '0';
    trload <= '0';  trbus <= '0';
    irload <= '0';
    rload <= '0';   rbus <= '0';
    acload <= '0';  acbus <= '0';
    zload <= '0';
```

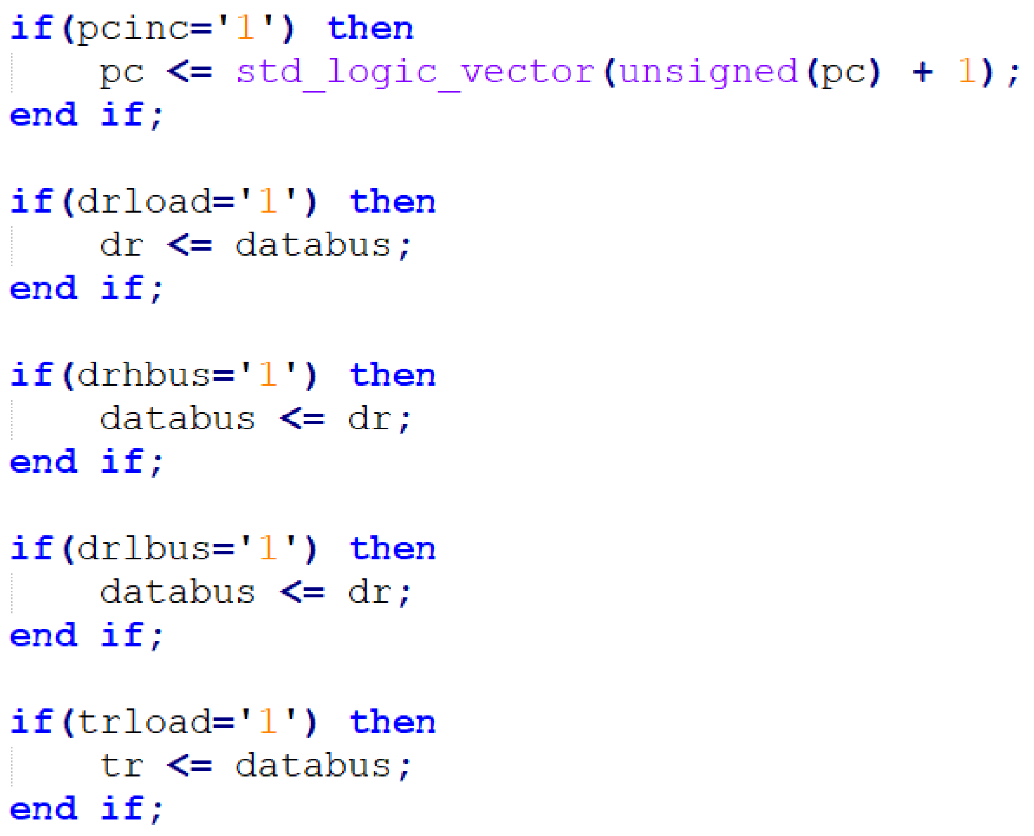


```
when LDAC2 =>
    -- tr <- dr; dr <- m; pc <- pc+1
    readm <= '1';   writem <= '0';  membus <= '1';
    arload <= '0';  arinc <= '0';
    pcbus <= '0';   pcload <= '0';  pcinc <= '1';
    drload <= '1';  drhbus <= '0';  drlbus <= '1';
    trload <= '1';  trbus <= '0';
    irload <= '0';
    rload <= '0';   rbus <= '0';
    acload <= '0';  acbus <= '0';
    zload <= '0';
```

相对简单的CPU设计



```
signal_control: process(clk)
begin
    if(rising_edge(clk)) then
        if(arload='1') then
            if(pcbus='1') then
                ar <= pc;
            end if;
            if(drlbus='1') then
                ar(15 downto 8) <= databus;
            end if;
            if(trbus='1') then
                ar(7 downto 0) <= databus;
            end if;
        end if;

        if(arinc='1') then
            ar <= std_logic_vector(unsigned(ar) + 1);
        end if;
```

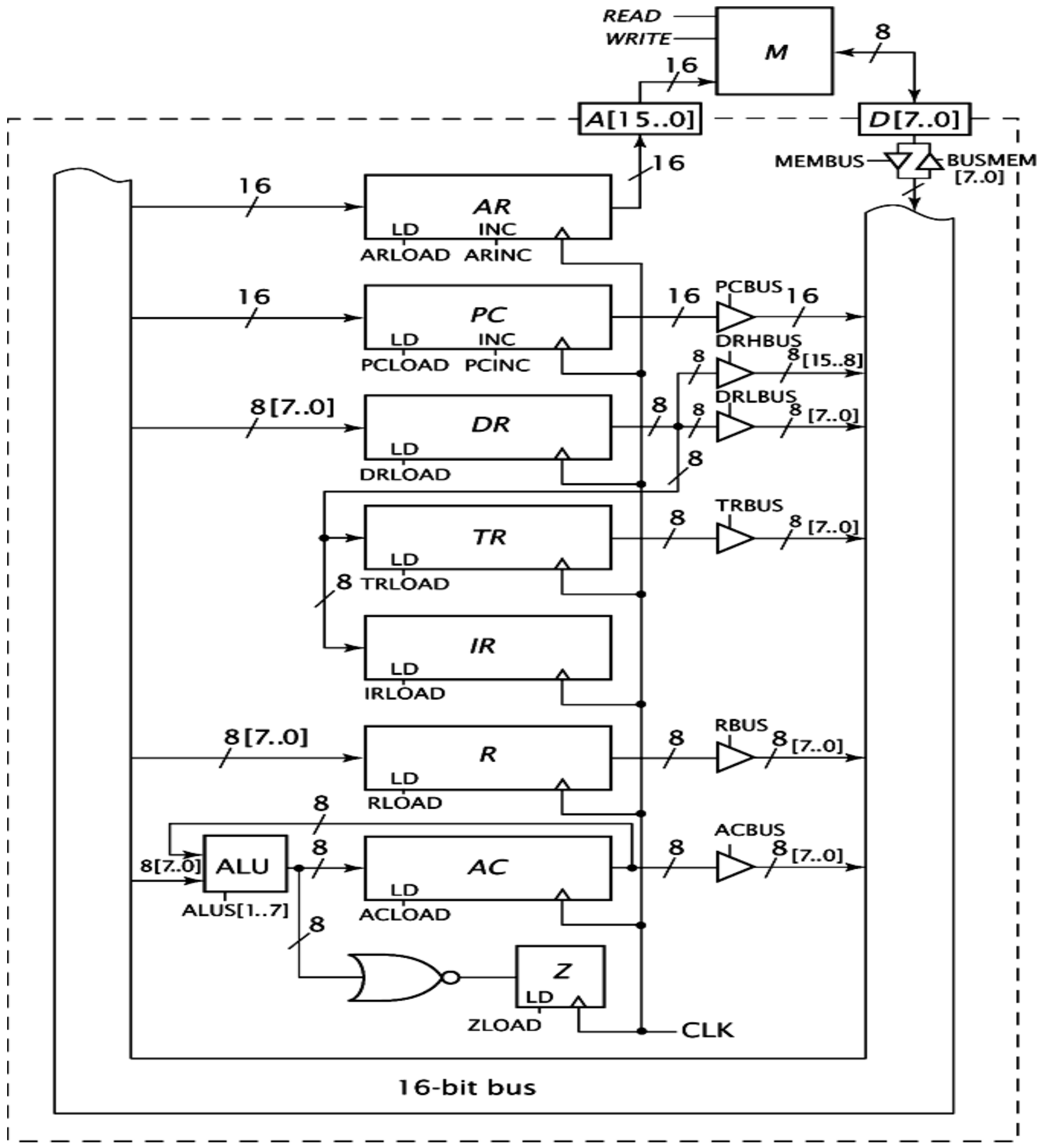


```
if(pcinc='1') then
    pc <= std_logic_vector(unsigned(pc) + 1);
end if;

if(drload='1') then
    dr <= databus;
end if;

if(drhbus='1') then
    databus <= dr;
end if;

if(drlbus='1') then
    databus <= dr;
end if;

if(trload='1') then
    tr <= databus;
end if;
```

相对简单的CPU设计

```
if(alusel="0001") then
    ac <= std_logic_vector(unsigned(ac) + unsigned(r));
    if(ac="00000000") then
        z <= '1';
    else
        z <= '0';
    end if;
elsif(alusel="0010") then
    ac <= std_logic_vector(unsigned(ac) - unsigned(r));
    if(ac="00000000") then
        z <= '1';
    else
        z <= '0';
    end if;
elsif(alusel="0011") then
    ac <= std_logic_vector(unsigned(ac) + 1);
    if(ac="00000000") then
        z <= '1';
    else
        z <= '0';
    end if;
```

相对简单的CPU设计

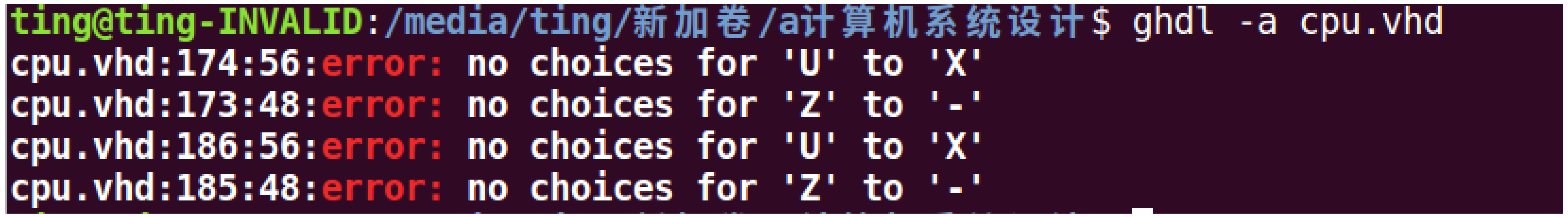

READ
WRITE
M
8
16
A[15..0]
D[7..0]
16
MEMBUS
BUSMEM [7..0]
16
AR
LD
INC
ARLOAD
ARINC
16
PC
LD
INC
PCLOAD
PCINC
16
PCBUS
16
DRHBUS
8
8 [15..8]
8[7..0]
DR
LD
DRLOAD
8
8
DRLBUS
8 [7..0]
8
TR
LD
8
TRLOAD
8
TRBUS
8 [7..0]
IR
LD
IRLOAD
8[7..0]
R
LD
RLOAD
8
RBUS
8 [7..0]
8
8[7..0]
ALU
ALUS[1..7]
8
AC
LD
ACLOAD
8
ACBUS
8 [7..0]
8
Z
LD
ZLOAD
CLK
16-bit bus

相对简单的CPU设计

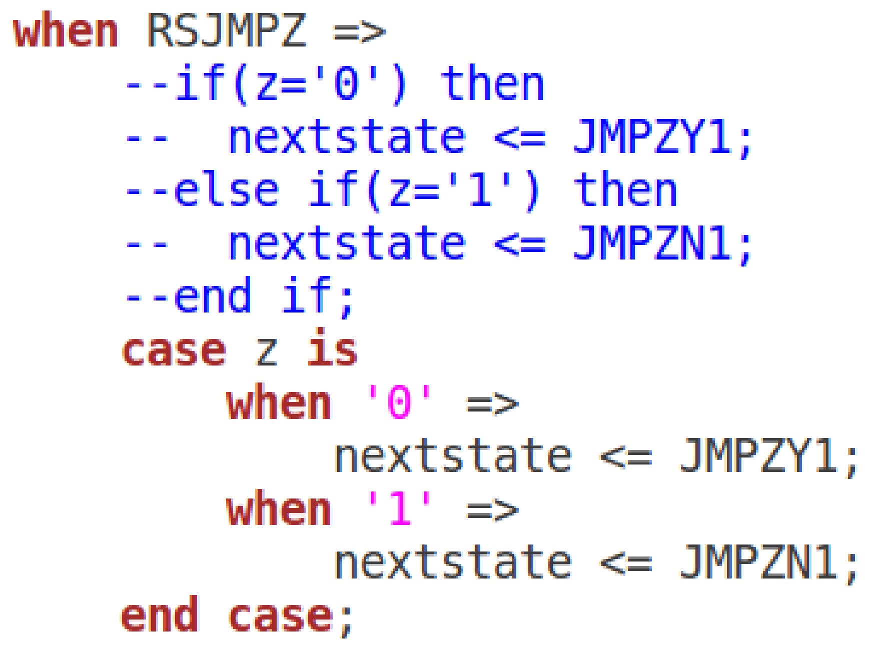
```
if(pcload='1') then
    if(drlbus='1') then
        pc(15 downto 8) <= databus;
    end if;
    if(trbus='1') then
        pc(7 downto 0) <= databus;
    end if;
end if;
```

```
if(arload='1') then
    if(pcbus='1') then
        ar <= pc;
    end if;
    if(drlbus='1') then
        ar(15 downto 8) <= databus;
    end if;
    if(trbus='1') then
        ar(7 downto 0) <= databus;
    end if;
end if;
```

相对简单的CPU设计



```
ting@ting-INVALID:/media/ting/新加卷/a计算机系统设计$ ghdl -a cpu.vhd
cpu.vhd:174:56:error: no choices for 'U' to 'X'
cpu.vhd:173:48:error: no choices for 'Z' to '-'
cpu.vhd:186:56:error: no choices for 'U' to 'X'
cpu.vhd:185:48:error: no choices for 'Z' to '-'
```



```
when RSJMPZ =>
    --if(z='0') then
    --   nextstate <= JMPZY1;
    --else if(z='1') then
    --   nextstate <= JMPZN1;
    --end if;
    case z is
        when '0' =>
            nextstate <= JMPZY1;
        when '1' =>
            nextstate <= JMPZN1;
    end case;
```

THANK YOU!