计算机科学基础 C

北京大学

简单 CPU 设计

设计简单CPU

CPU位宽：16位

指令格式声明：

III XXX YYY，

III 表示指令，XXX 表示Rx寄存器，YYY 表示Ry寄存器，
Miv指令后跟随输入数据是16位立即数D。

指令编码表：

```
000:  mv    Rx,Ry     // Rx <- [Ry]
001:  mvi   Rx,#D     // Rx <- D
010:  add   Rx, Ry    // Rx <- [Rx] + [Ry]
011:  sub   Rx, Ry    // Rx <- [Rx] - [Ry]
```

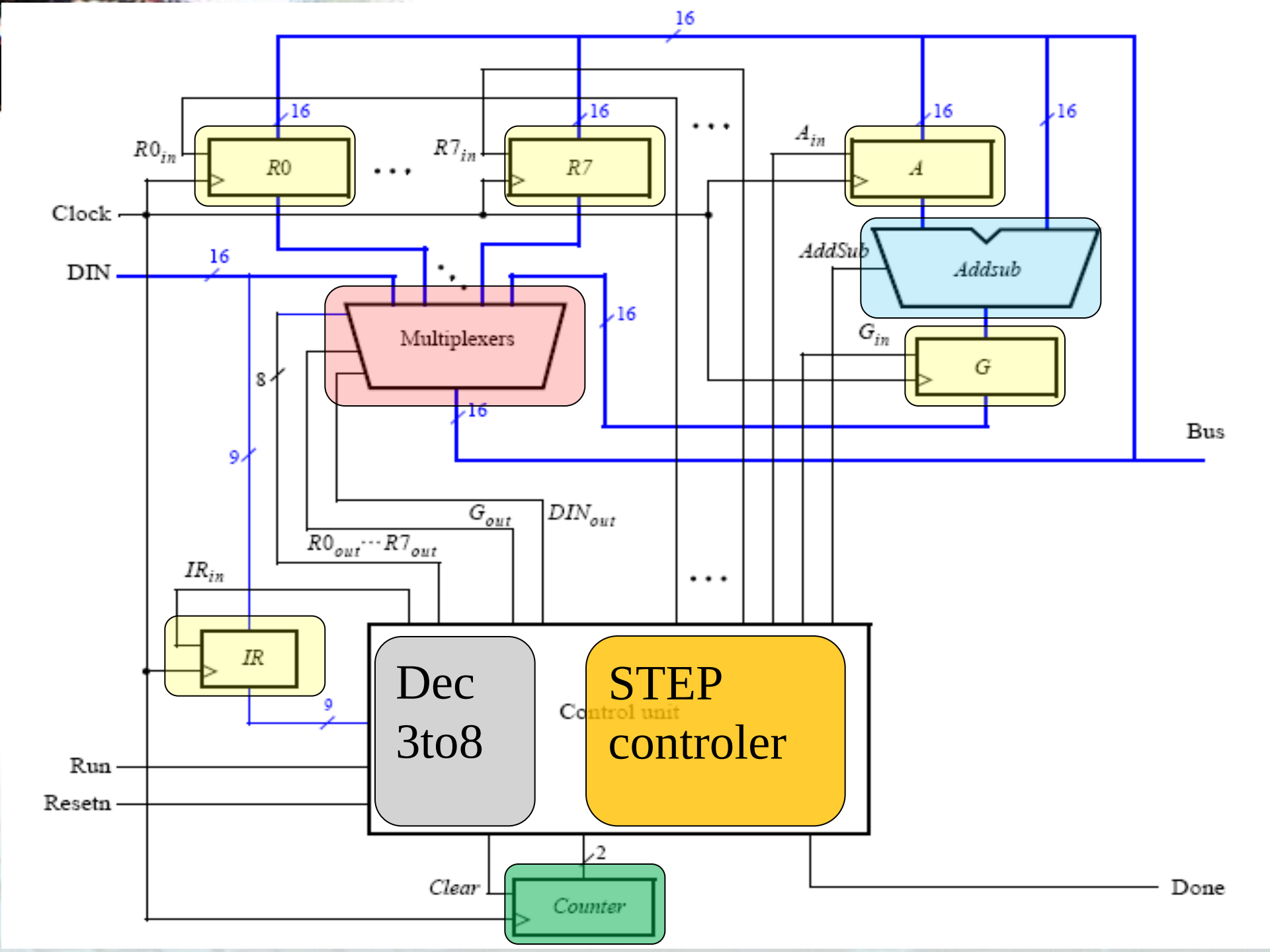
怎么做?

块思维思想：运筹帷幄，布局规划，驾驭你的模块！

块思维方法:
1、分析CUP外有哪些port？
2、思考CPU内有哪些block？

port：
DIN，BusWires，Resetn，Clock，Run，Done

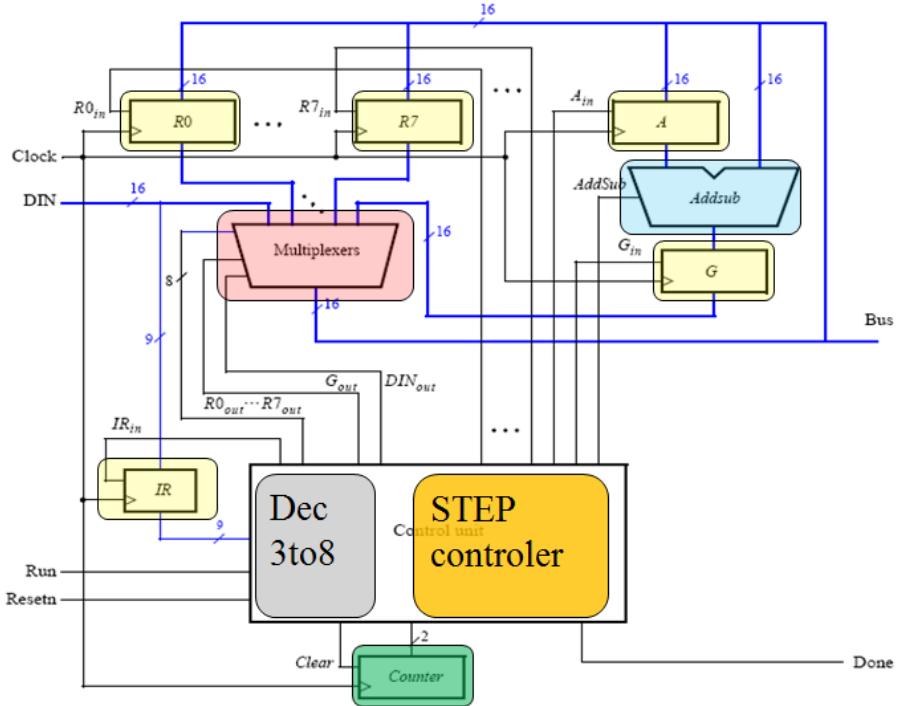
block：
alu(addsub)，controler，regn, mux, upcount, dec3to8


16
$R0_{in}$
R0
$R7_{in}$
R7
A_{in}
A
Clock
DIN
AddSub
Addsub
Multiplexers
G_{in}
G
8
9
Bus
G_{out}
DIN_{out}
$R0_{out} \cdots R7_{out}$
IR_{in}
IR
Dec
3to8
Control unit
STEP
controler
Run
Resetn
2
Clear
Counter
Done

控制时序

用组合电路准备 clk触发前瞬间更新 | 用组合电路准备 clk触发前瞬间更新 | 用组合电路准备 clk触发前瞬间更新 | 用组合电路准备 clk触发前瞬间更新

指令 \ 时间	T0 →	T1 →	T2 →	T3 →
(mv):I_0	IR_{in}	RY_{out}, RX_{in}, Done	—	—
(mvi):I_1	IR_{in}	DIN_{out}, RX_{in}, Done	—	—
(add):I_2	IR_{in}	RX_{out}, A_{in}	RY_{out}, G_{in}	G_{out}, RX_{in}, Done
(sub):I_3	IR_{in}	RX_{out}, A_{in}	RY_{out}, G_{in}, Addsub	G_{out}, RX_{in}, Done



Rout表示把某个寄存器中的数据放出到汇流台

Rin表示把汇流台中的数据放入指定的R寄存器

用组合电路准备 clk触发前瞬间更新 | 用组合电路准备 clk触发前瞬间更新 | 用组合电路准备 clk触发前瞬间更新 | 用组合电路准备 clk触发前瞬间更新

指令 \ 时间	T0 →	T1 →	T2 →	T3 →
(mv):I_0	IR_{in}	RY_{out}, RX_{in}, Done	—	—
(mvi):I_1	IR_{in}	DIN_{out}, RX_{in}, Done	—	—
(add):I_2	IR_{in}	RX_{out}, A_{in}	RY_{out}, G_{in}	G_{out}, RX_{in}, Done
(sub):I_3	IR_{in}	RX_{out}, A_{in}	RY_{out}, G_{in}, Addsub	G_{out}, RX_{in}, Done

Mvi R3,5　001 011 .....　(SW15-SW0)　　clk1个T0

……101　　clk1个T1

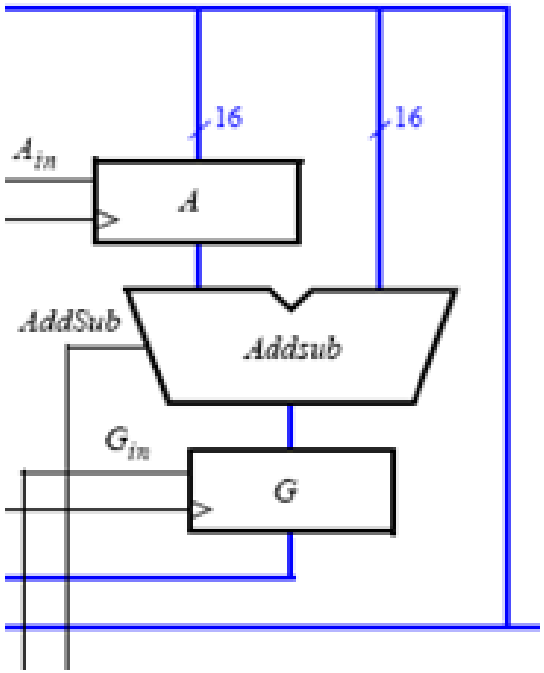
Mvi R7,9　001 111 ....　　clk1个T0

……1001　　clk1个T1

Add R3,R7　010 011 111…　　clk3个T0 T1 T2 出现到总线，1个clkT3进入R3

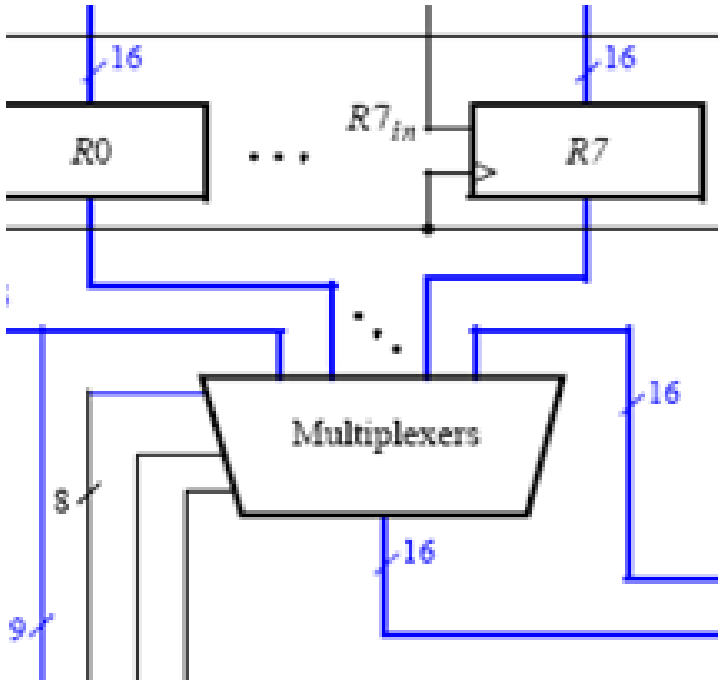
AddSub

```
1  //Name: AddSub
2  //Function:
3  //       AddSub = 0 Add
4  //       AddSub = 1 Sub
5  //Author: caojian
6
7  module addsub(AddSub, A, BusWires, Sum);
8  input AddSub;
9  input[15:0] A, BusWires;
10 output[15:0] Sum;
11 reg[15:0] Sum;
12
13 always@(AddSub or A or BusWires)
14 begin
15     if(!AddSub)
16     Sum = A + BusWires;
17     else
18     Sum = A - BusWires;
19 end
20
21 endmodule
22
```



busmux

```
1  //Name: busmux
2  //Function:
3  //      select which reg push to bus
4  //Author: caojian
5  module busmux(Rout, Gout, DINout, R0, R1, R2, R3, R4, R5, R6, R7, G, DIN, BusWires);
6  input[7:0]  Rout;
7  input Gout, DINout;
8  input[15:0] R0, R1, R2, R3, R4, R5, R6, R7, G, DIN;
9  output[15:0] BusWires;
10 reg[15:0] BusWires;
11 wire[9:0] Sel;
12 assign Sel = {Rout, Gout, DINout};
13 always@(*)
14 begin
15     if(Sel == 10'b1000000000)
16         BusWires = R0;
17     else if(Sel == 10'b0100000000)
18         BusWires = R1;
19     else if(Sel == 10'b0010000000)
20         BusWires = R2;
21     else if(Sel == 10'b0001000000)
22         BusWires = R3;
23     else if(Sel == 10'b0000100000)
24         BusWires = R4;
25     else if(Sel == 10'b0000010000)
26         BusWires = R5;
27     else if(Sel == 10'b0000001000)
28         BusWires = R6;
29     else if(Sel == 10'b0000000100)
30         BusWires = R7;
31     else if(Sel == 10'b0000000010)
32         BusWires = G;
33     else BusWires = DIN;
34 end
```



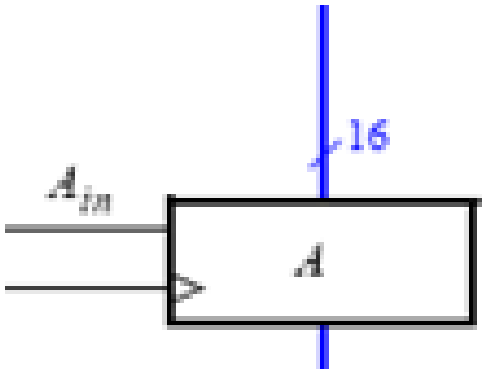
select which reg push to bus

dec3to8

```
1  //File name: dec3to8
2  //Function: dec 3 to 8
3  //Author: Zhang Lei
4
5  module dec3to8 (W, En, Y);
6
7  input [2:0] W;
8  input  En;
9  output[7:0] Y;
10 reg   [7:0] Y;
11
12
13 always @ (W or En)
14  begin
15   if(En == 1)
16      case(W)
17        3'b000:  Y = 8'b10000000;
18        3'b001:  Y = 8'b01000000;
19        3'b010:  Y = 8'b00100000;
20        3'b011:  Y = 8'b00010000;
21        3'b100:  Y = 8'b00001000;
22        3'b101:  Y = 8'b00000100;
23        3'b110:  Y = 8'b00000010;
24        3'b111:  Y = 8'b00000001;
25      endcase
26   else
27      Y = 8'b00000000;
28  end
29
30 endmodule
```

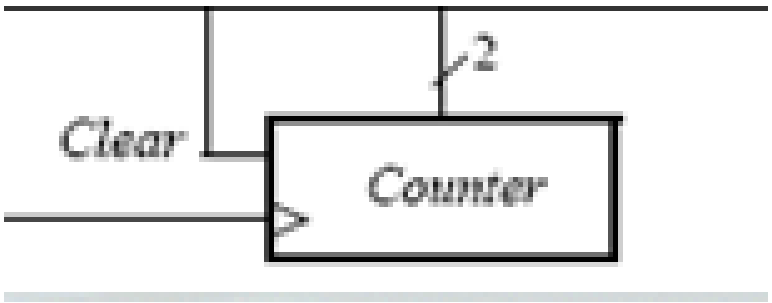
regn

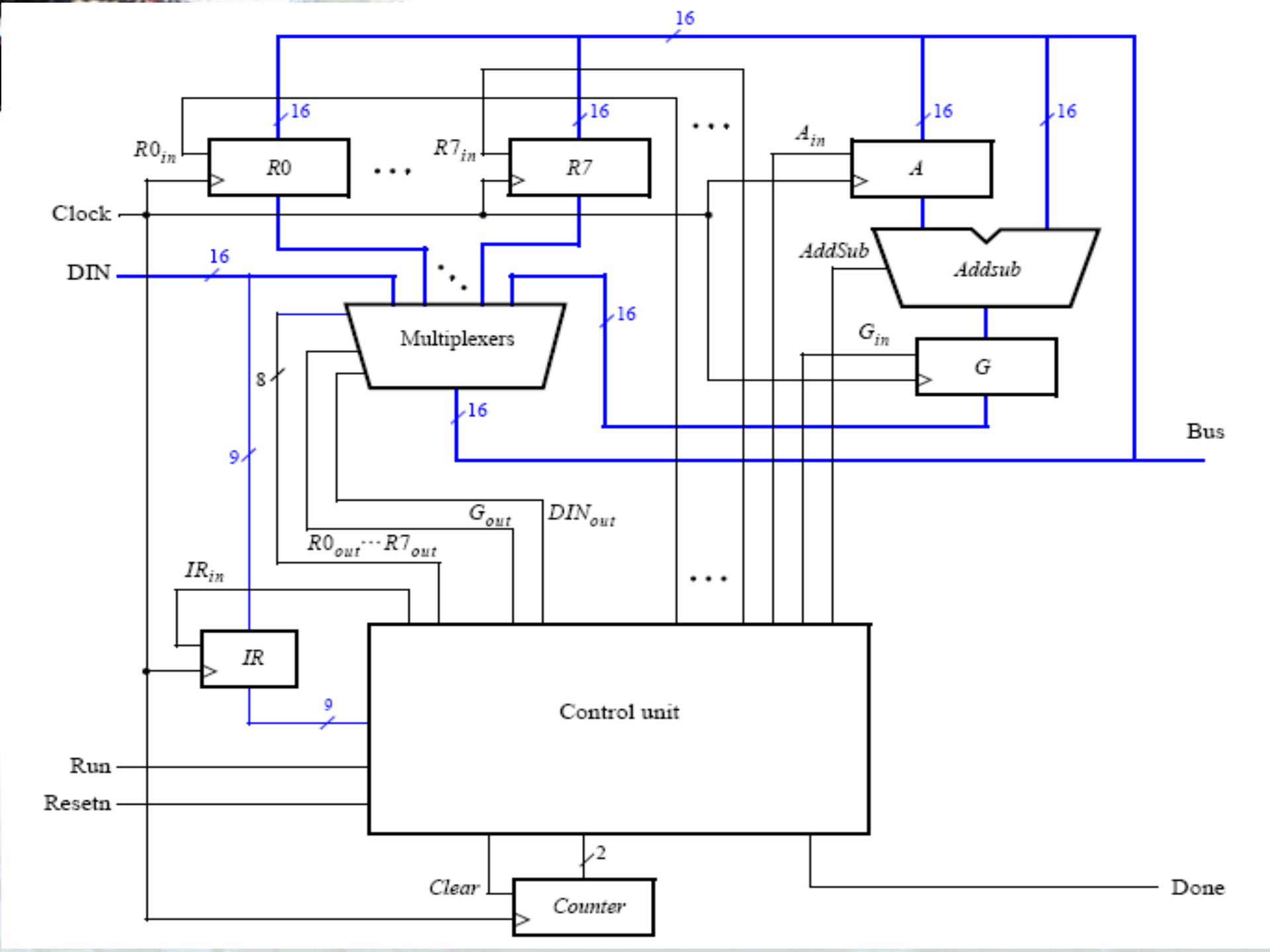
```
1   //File name:regn.v
2   //Function:Register,posedge clock active.
3   //Author:Eric Liu
4   module regn(R,Rin,Clock,Q);
5       parameter n = 16;
6       input[n-1:0] R;
7       input Rin,Clock;
8       output[n-1:0] Q;
9       reg [n-1:0] Q;
10
11  always @(posedge Clock)
12      if(Rin)
13    Q <= R;
14  endmodule
```



upcounte

```
1  //File:   upcount.v
2  //Function:   a counter for the instructions
3  //Author: Caojian
4  module upcount(Clear,Clock,Q);
5      input Clear,Clock;
6      output [1:0] Q;
7      reg [1:0] Q;
8
9      always @(posedge Clock)
10         if (Clear)
11             Q <= 2'b0;
12         else
13             Q <= Q+1'b1;
14 endmodule
```




16
16
16
16
16
$R0_{in}$
R0
$R7_{in}$
R7
A_{in}
A
Clock
DIN
16
AddSub
Addsub
Multiplexers
16
G_{in}
G
8
16
Bus
9
G_{out}
DIN_{out}
$R0_{out} \cdots R7_{out}$
IR_{in}
IR
9
Control unit
Run
Resetn
2
Clear
Counter
Done

CPU效果展示

用SW15-SW0驱动DIN；
用SW17作为Run；
KEY0作为Resetn；
KEY1作为Clock
总线接到LEDR15-LEDR0
Done接到LEDR17

完成prob.v的编写，上板调试。

作业：

完成简单的CPU设计
邮件标题：CPU1_学号_姓名
内容：附件：proc.v