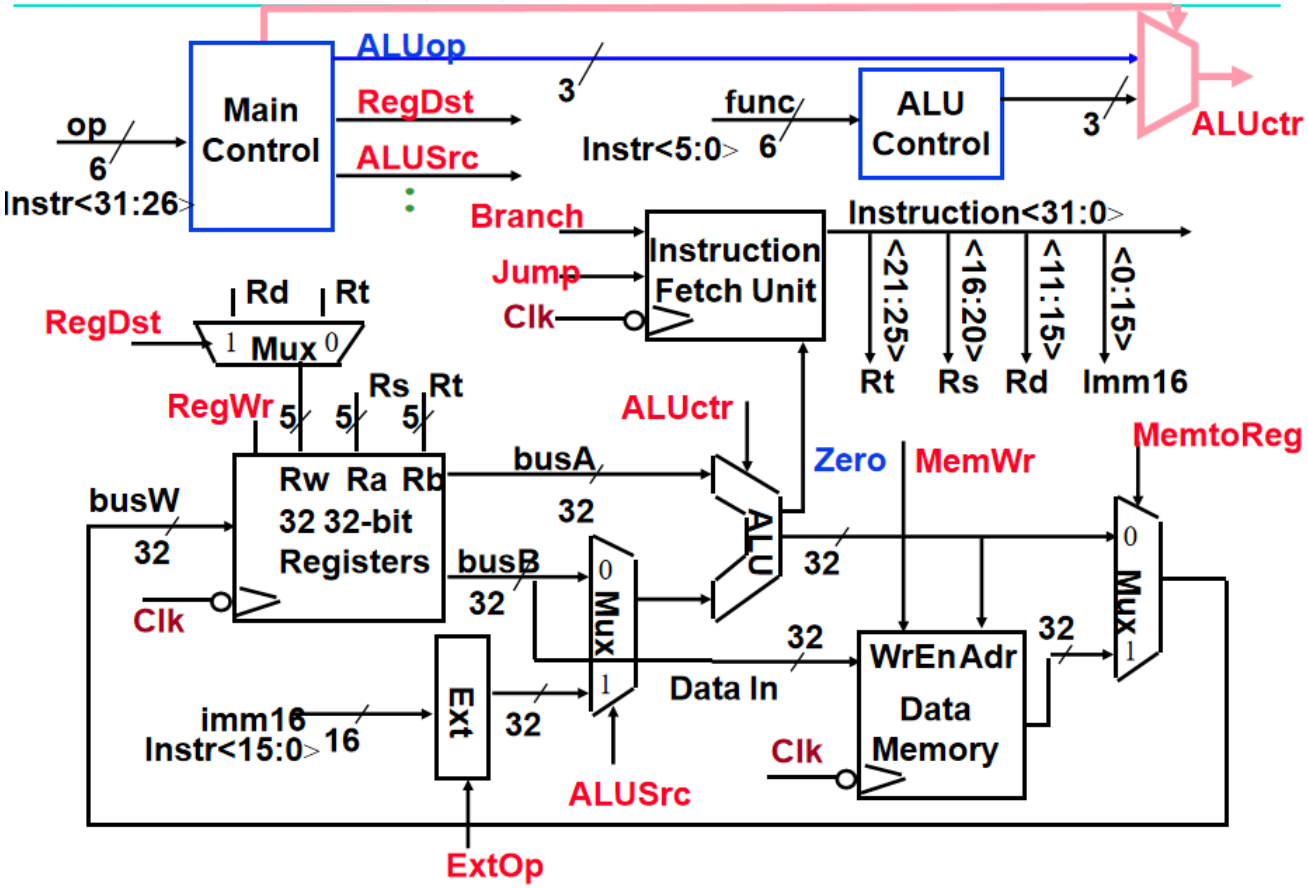
实验五：数据通路

设计思路



模块结构：

1. `fetchins` 模块：

- **初始化指令内存：** 在 `initial` 块中，初始化了一个包含几条MIPS指令的指令内存 `inst_mem` 。
- **指令获取：** 根据程序计数器(`pc`)的值，从指令内存中获取当前指令 `inst` 。
- **跳转和分支目标计算：** 根据指令中的跳转和分支条件，计算下一个地址 `next_addr` 。
- **程序计数器更新：** 在每个时钟的下降沿，将程序计数器 `pc` 更新为下一个地址 `next_addr` 。

2. `Reg` 模块：

- **寄存器读写控制：** 根据输入的控制信号，判断是否进行寄存器的读取和写入操作。
- **寄存器读取和写入：** 根据输入的寄存器地址 `Ra` 和 `Rb` ，从寄存器中读取数据。根据写入信号，将数据写入寄存器。

3. `MUX2to1` 模块：

- **2选1多路复用器：** 根据输入的选择信号 `ctr`，选择其中一个输入作为输出。

4. `alu` 模块：

- **ALU运算：** 根据输入的操作数 `A`、`B` 和 ALU 控制信号 `ALUctr` 进行 ALU 运算。
- **运算结果：** 输出运算结果 `Result`、零标志位 `Zero`、溢出标志位 `Overflow`。

5. `DataMemory` 模块：

- **数据存储器：** 根据输入的写入使能信号 `WrEn`，在时钟沿上升或写入使能信号上升沿时，将数据写入或读取数据存储器。

6. 主模块：

- **指令获取和解析：** 使用 `fetchins` 模块获取当前指令，并解析指令的操作码和操作数。
- **寄存器操作：** 使用 `Reg` 模块进行寄存器的读取和写入。
- **ALU运算：** 使用 `alu` 模块进行ALU运算，计算ALU的输出。
- **数据存储器操作：** 使用 `DataMemory` 模块进行数据存储器的读取和写入。
- **多路复用器操作：** 使用 `MUX2to1` 进行数据选择。

这样，整个模块实现了一个基本的MIPS指令集的数据通路控制，根据不同的指令和控制信号，执行相应的操作。

代码展示

```
module
dataroad(clk,RegWr,Branch,Jump,ExtOp,AluSrc,Aluctr,MemWr,MemtoReg,RegDst,instr
uction
,busA,busB,AluB,AluF,busW,Datain,Dataout);//,run,busA
parameter n=32;
input clk,RegWr,Branch,Jump,ExtOp,AluSrc,MemWr,MemtoReg,RegDst;//,run
input[2:0] Aluctr;
output [n-1:0] instruction;
output [n-1:0] busA,busB,AluB,AluF,busW,Datain,Dataout;

wire[5:0] op,func;
wire[4:0] Rs,Rt,Rd;
wire[15:0] imm16;
wire[4:0] Rw,Ra,Rb;
wire[n-1:0] imm32;
wire  Zero,Overflow,Ve;

//instruction
fetchins fetch(clk,Zero,Branch,Jump,instruction);//,run
//module fetchins( Clk,Zero,Branch,Jump,inst);
assign op=instruction[31:26];
assign Rs=instruction[25:21];
assign Rt=instruction[20:16];
```

```
assign Rd=instruction[15:11];
assign func=instruction[5:0];
assign imm16=instruction[15:0];

//registers
MUX2to1 mux6(Rt,Rd,RegDst,Rw);//module MUX2to1(X,Y,ctr,Z);RegDst
xuanzejichuqixieru 1 rd 0 rt
assign Ra=Rs;
assign Rb=Rt;
assign Ve=RegWr & ~Overflow;//buyichu qie kexieru
Reg regs(clk,busW,Ve,Rw,Ra,Rb,busA,busB);//,run
//module Reg(Clk, busW, RegWr, Rw, Ra, Rb, busA, busB,run);

//ALU
assign imm32={{16{imm16[15]&ExtOp}},imm16[15:0]};//imm sign extension
MUX2to1 mux7(busB,imm32,AluSrc,AluB);//0 busB 1 imm sign extension  B
alu alu1(busA,AluB,Aluctr,Zero,Overflow,AluF);
//module alu(A,B,ALUctr,Zero,Overflow,Result);

//data memory
assign Datain=busB;
DataMemory datamem(Datain,clk,MemWr,AluF,Dataout);
//module DataMemory(DataIn,Clk,WrEn,Adr,DataOut);
MUX2to1 mux8(AluF,Dataout,MemtoReg,busW);//dizhi huozhe cunchuqilideneirong

endmodule
```

波形分析

波形分析将在下一个实验和译码器组成单周期cpu后验证。