**Laboratório IV – Projeto CPU Monociclo**

Nome: Gabriel Gatti da Silva Matrícula: 2021100336

Nome: Thiago F. N. Lahass Matrícula: 2021100178

Data: 24/11/22

**1. OBJETIVOS**

Entender os passos necessários para estender uma CPU para que ela dê suporte a execução de novas instruções na arquitetura (Instruction Set Architecture ou ISA) que não haviam sido implementadas no projeto inicial da CPU monociclo.

**2. ATIVIDADES**

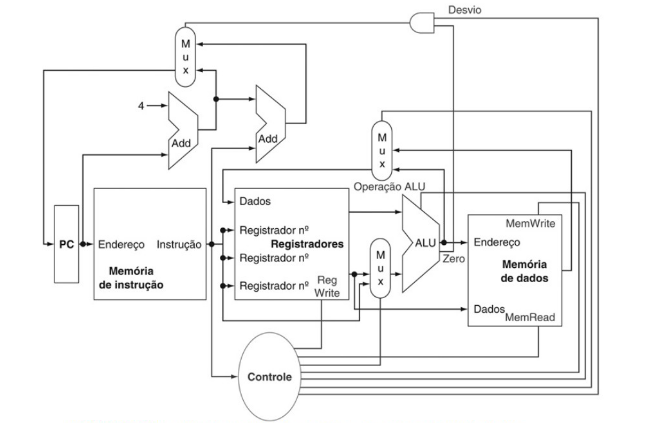
****

Figura 1: Circuito com implementação básica do subconjunto MIPS.

**2.1. Instrução 1:**

*lwr Rt, Rd(Rs)*

*– Reg[Rt] = Mem[Reg[Rd] + Reg[Rs]]*

a) Quais blocos existentes (se existirem) podem ser reaproveitados para esta nova instrução?

Todos os blocos podem ser reaproveitados, exceto a porta AND e o segundo ADD que não são necessários para essa instrução.

b) Quais novos blocos devem ser adicionados para esta nova instrução?

Somente um extensor de sinal para que os 16 bits do deslocamento passem a ter o formato correto na entrada da ALU (32 bits) e uma unidade de controle para a ALU (por simplicidade).

c) Quais sinais são necessários para se executar esta nova instrução?

1. Uma instrução é buscada da memória de instruções e o PC é incrementado.

2. Um valor de registrador é lido do banco de registradores.

3. A ALU calcula a soma do valor lido do banco de registradores com os 16 bits menos significativos com sinal estendido da instrução (offset).

4. A soma da ALU é usada como o endereço para a memória de dados.

5. Os dados da unidade de memória são escritos no banco de registradores; o registrador de destino é fornecido pelos bits 20:16 da instrução (rt).

d) Redesenhe o projeto com o suporte para a instrução.

Por questões de simplicidade foi utilizado o drMips para geração dos circuitos, como não há como editar os blocos no mesmo (adicionar/remover), utilizar como critério que os blocos em vermelho devem ser ignorados e que a sua entrada está ligada na sua saída.

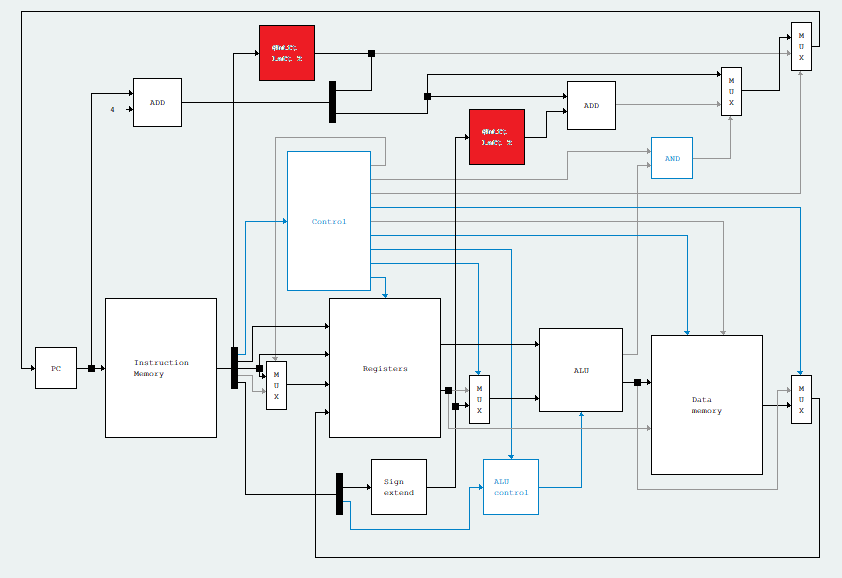


Figura 2: Circuito com suporte a instrução *load* ( linhas em cinza e blocos controlados/ que se utilizam delas devem ser desconsideradas pois são parâmetros não utilizados pela instrução).

**2.2. Instrução 2:**

*addi Rt, Rd, imed.*

*– Reg[Rt] = Reg[Rd] + imed*

a) Quais blocos existentes (se existirem) podem ser reaproveitados para esta nova instrução?

Todos os blocos podem ser reaproveitados, exceto a porta AND.

b) Quais novos blocos devem ser adicionados para esta nova instrução?

Extensor de sinal, deslocador de 2 bits à esquerda, e uma unidade de controle para a ALU.

c) Quais sinais são necessários para se executar esta nova instrução?

1. A instrução é buscada e o PC é incrementado.

2. Dois registradores, rs e rt, são lidos do banco de registradores, e a unidade de controle principal calcula a definição das linhas de controle também durante essa etapa.

3. A ALU opera nos dados lidos do banco de registradores, usando o código de função (bits 5:0, que é o campo funct, da instrução) para gerar a função da ALU.

4. O resultado da ALU é escrito no banco de registradores usando os bits 15:11 da instrução para selecionar o registrador de destino rd.

d) Redesenhe o projeto com o suporte para a instrução.

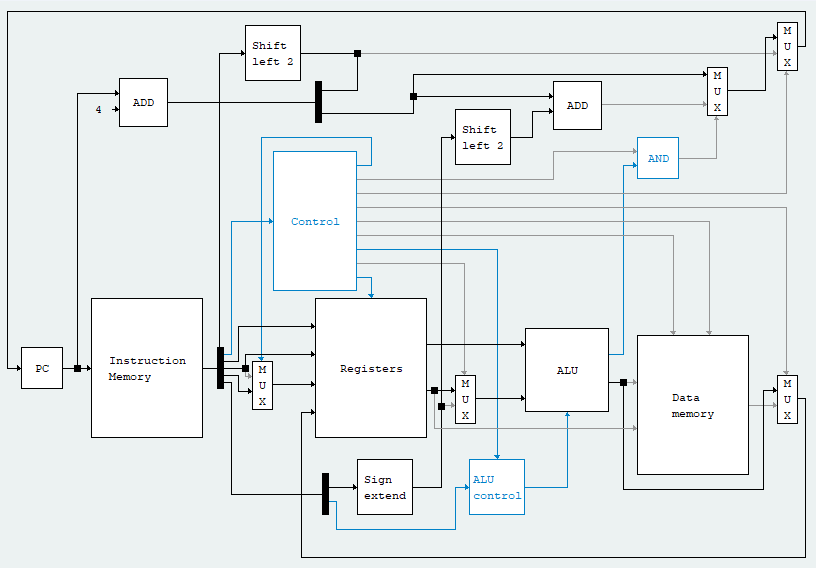


Figura 3: Circuito com suporte a instrução *addi*.

**2.3. Instrução 3:**

*bne Rt, Rd, desloc*

*– if (Reg[Rt] != Reg[Red]) {*

*PC = (PC + 4) + desloc; // salta se diferentes*

*}*

*else {*

*PC = PC + 4;*

*}*

a) Quais blocos existentes (se existirem) podem ser reaproveitados para esta nova instrução?

Todos os blocos podem ser reaproveitados.

b) Quais novos blocos devem ser adicionados para esta nova instrução?

Extensor de sinal, deslocador de 2 bits à esquerda, e uma unidade de controle para a ALU.

c) Quais sinais são necessários para se executar esta nova instrução?

1. A instrução é buscada e o PC é incrementado.

2. Dois registradores, rs e rt, são lidos do banco de registradores.

3. A ALU realiza uma subtração dos valores de dados lidos do banco de registradores. O valor de PC + 4 é somado aos 16 bits menos significativos com sinal estendido (offset) deslocados de dois para a esquerda; o resultado é o endereço de destino do desvio.

4. O resultado Zero da ALU é usado para decidir qual resultado do somador deve ser armazenado no PC. Se o sinal Zero for diferente de 0, então se deve saltar para o endereço de destino, para isso, podemos utilizar um sistema bem parecido com a instrução beq, porém no nosso caso, ao invés de uma porta AND que liga o sinal Zero com o sinal de controle Branch, fazemos uma porta AND que liga um sinal BrachNE definida pelo opcode quando a instrução for bne, a um sinal NOT Zero.

d) Redesenhe o projeto com o suporte para a instrução.

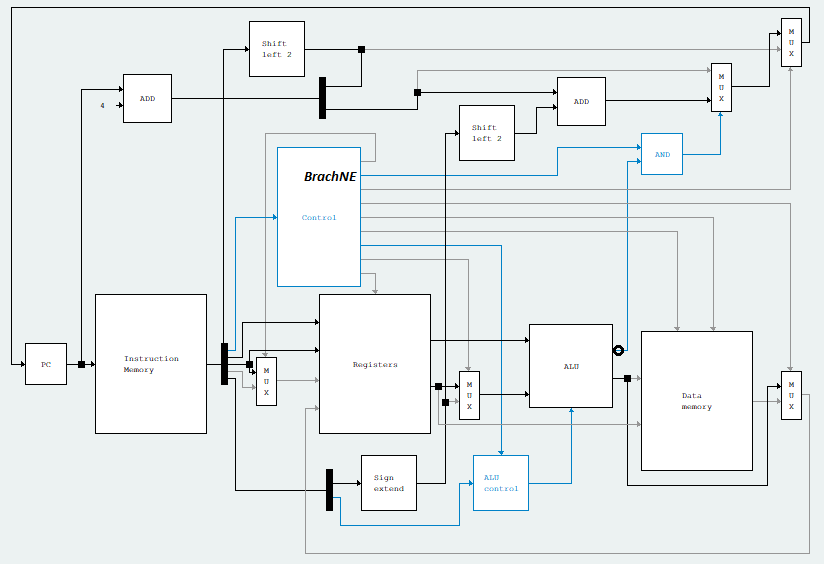


Figura 4: Circuito com suporte a instrução *bne*.