CI1212 Arquitetura de Computadores - 2025/1 - UFPR

Relatório do Trabalho 2

Alejandro David Nava Nava (GRR: 20242778)

Diagrama do projeto do Bloco Iterativo do REDUX-V:

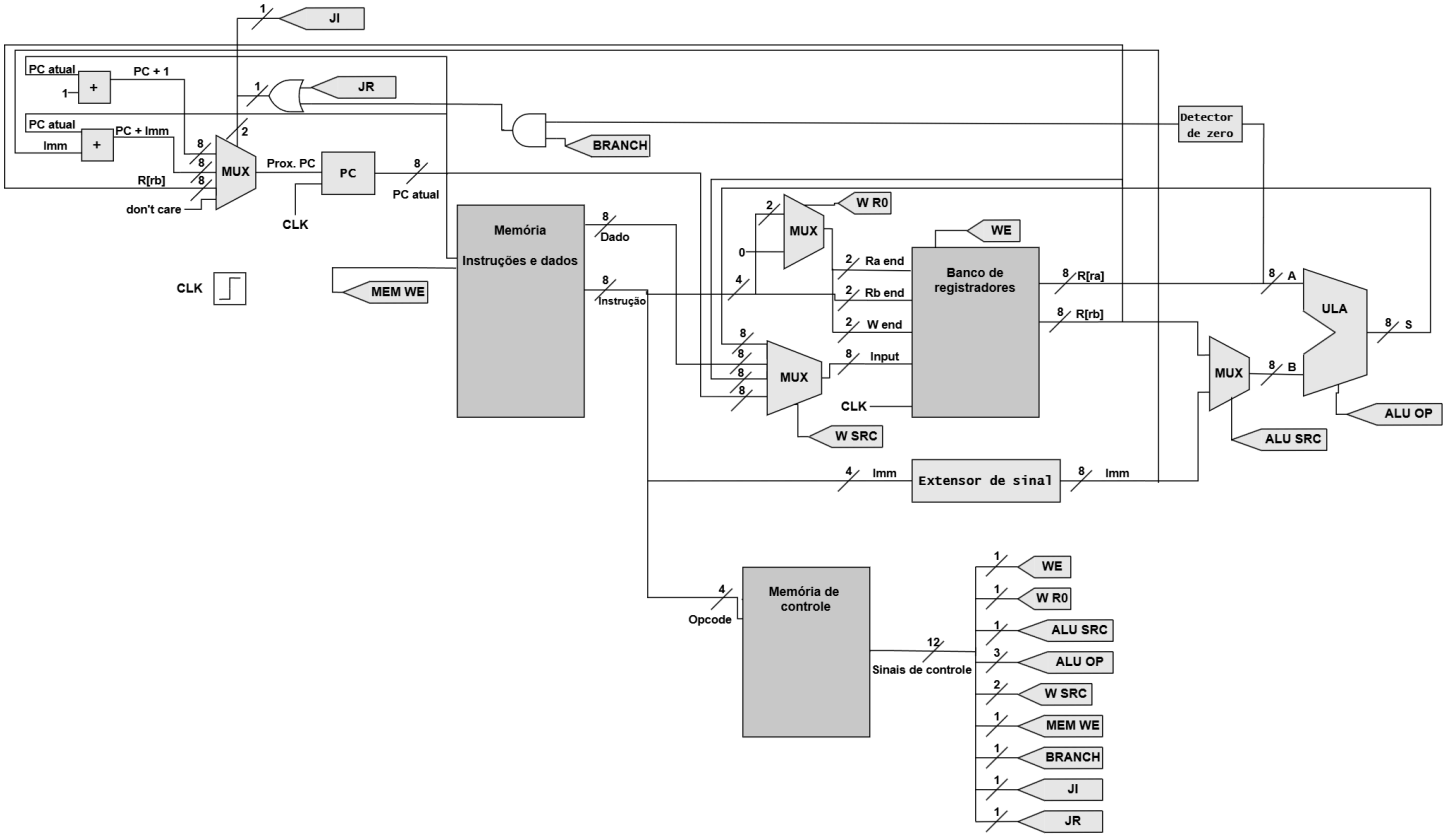
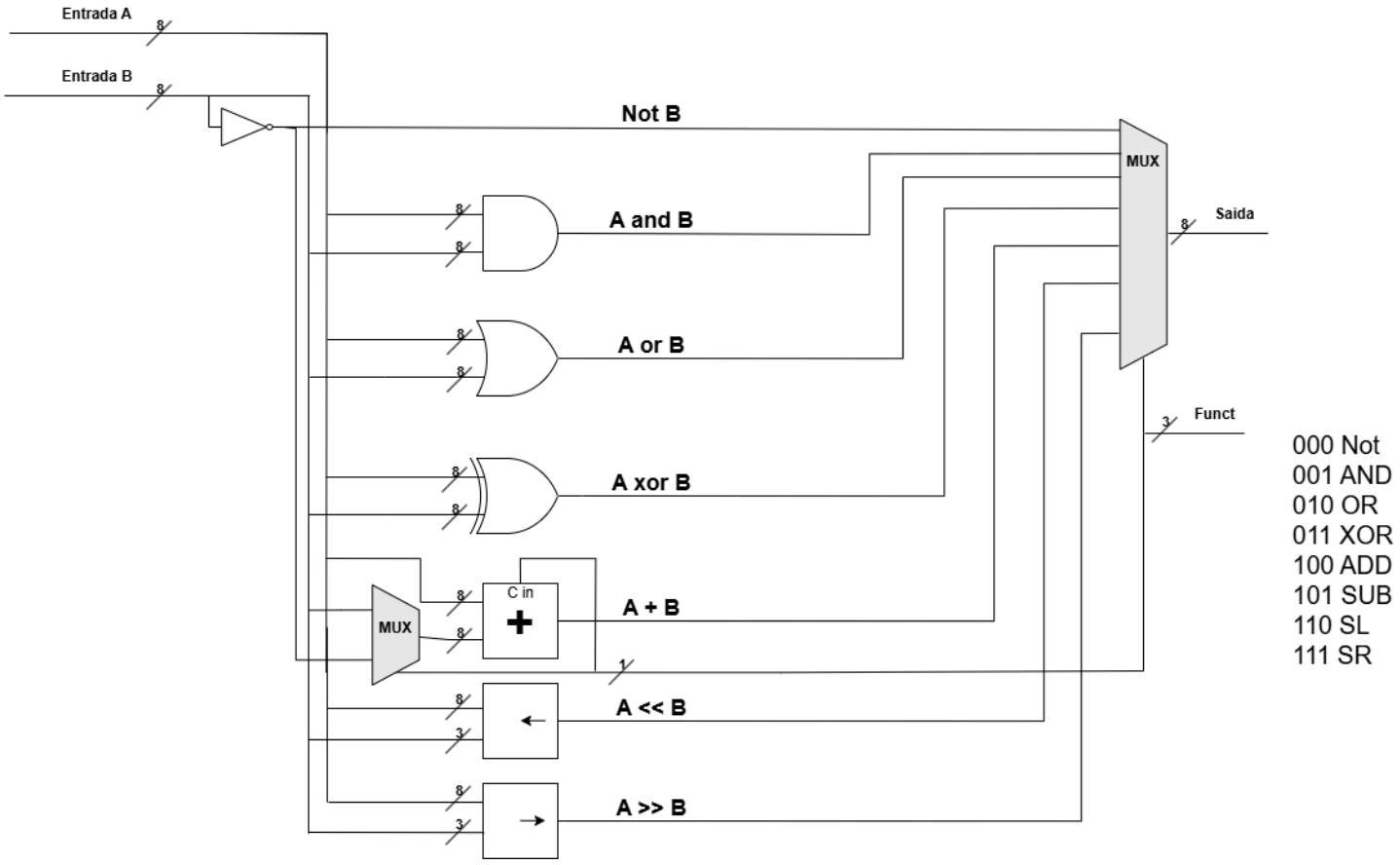


Diagrama do projeto da ULA:



Instruções novas implementadas:

Foram implementadas as seguintes instruções:

Opcode	Tipo	Mnemonic	Nome	Operação
0101	R	mov	Move	R[ra] = R[rb]
0110	R	jr	Jump register	PC = R[rb]
0111	I	gpci	Get PC + immediate	R[0] = PC + Imm

As novas instruções implementadas resultam de muita utilidade na nova versão do código, permitindo reduzir o número de linhas e facilitando certas operações. Por exemplo, no caso da instrução **mov**, ela consegue reduzir pela metade o seguinte código para passar o conteúdo do registrador rb para o registrador ra:

```
sub ra, ra
add ra, rb
```

Já no caso da instrução **gpci**, ela resulta útil para simplificar o cálculo dos endereços das instruções às quais é preciso saltar no programa a partir do PC atual e faz com que esse cálculo fique mais claro, facilitando o entendimento do código. De forma complementar, o instruçao **jr** permite realizar saltos incondicionais maiores, sem a necessidade de jumps auxiliares, deixando o código menor e mais estilizado.

Programa em Assembly:

```
        sub r0, r0  ; zera registradores
        mov r1, r0
        mov r2, r0
        mov r3, r0

        addi 1      ; salva 1 em r2
        mov r2, r0  ; auxilia em varios calculos

        addi 4
        slr r0, r0  ; calcula endereco base de memoria (160)
        mov r1, r0  ; salva em r1

        st r3, r1   ; salva iterador na memória no endereço base

        add r1, r2
        gpci 7      ; calcula endereço do começo do laço
        addi 5
        st r0, r1   ; salva no endereco base + 1

        add r1, r2

        addi 7      ; calcula endereço do fim do laço
        addi 7
        addi 7
        addi 6
        st r0, r1   ; salva em endereço base + 2

        sub r1, r2
        mov r0, r1  ; deixa endereco base + 1 em r0 para facilitar a iteracao no laço
        sub r1, r2  ; r1 volta a conter o endereço base

loop:
        addi 1      ; r0 guarda endereco base + 2
        ld r3, r0   ; recupera o endereço do fim do laço
        ld r0, r1   ; recupera o iterador
        addi -5
        addi -5     ; calcula iteração atual
        brzr r0, r3 ; se iterador == 10 vai para o fim do laço

        mov r0, r1  ; coloca o endereço base em r0
        ld r3, r0   ; resgata o iterador e coloca em r3
        addi 3      ; desloca até a primeira posição do vetor A
        add r0, r3  ; desloca até a posição do vetor com base na iteração atual

        slr r3, r2  ; r3 = iterador * 2

        st r3, r0   ; salva r3 na memoria em A[i]

        addi 5
        addi 5      ; desloca na memória até o vetor B

        add r3, r2  ; r3 = (iterador * 2) + 1
```

```
        st r3, r0   ; salva r3 em B[i]

        addi 5
        addi 5      ; desloca na memória até o vetor R

        sub r3, r3  ; zera r3

        st r3, r0   ; salva r3 em R[i]

        ld r0, r1   ; recupera iterador da memoria
        addi 1      ; incrementa iterador em 1
        st r0, r1   ; salva iterador atualizado na memória
        mov r0, r1
        addi 1      ; r0 recebe endereco base + 1
        ld r3, r0   ; recupera endereço do começo do laço e coloca em r3

        jr r3

fim_loop:

        add r1, r2
        gpci 7
        addi 7      ; calcula endereço do começo do laço da soma
        st r0, r1   ; salva na memória em endereço base + 1

        add r1, r2
        addi 7
        addi 7
        addi 7
        addi 4      ; calcula endereço do fim do laço da soma
        st r0, r1   ; salva na memória em endereço base + 2

        sub r1, r2
        mov r0, r1  ; deixa endereco base + 1 em r0
        sub r1, r2  ; r1 volta a conter o endereço base

        sub r3, r3
        st r3, r1   ; reinicia iterador

loopS:
        addi 1
        ld r3, r0   ; resgata endereço do fim do laço da memória
        ld r0, r1   ; resgata iterador
        addi -5
        addi -5     ; calcula interação atual
        brzr r0, r3 ;

        mov r0, r1  ; carrega endereço base de memória em r0
        addi 3      ; desloca até a primeira posição do vetor A

        ld r3, r1   ; carrega iterador em r3
        add r0, r3

        ld r2, r0   ; carrega A[i] em r2

        addi 5
        addi 5      ; desloca até o vetor B

        ld r3, r0   ; carrega B[i] em r3

        add r2, r3  ; A[i] + B[i]
```

```
        addi 5
        addi 5      ; desloca até o vetor R

        st r2, r0   ; R[i] = A[i] + B[i]

        ld r0, r1
        addi 1      ; incrementa iterador em 1
        st r0, r1   ; salva na memoria

        mov r0, r1
        addi 1
        ld r3, r0   ; recupera o endereço do começo do laço da memória

        jr r3

fim_loopS:
        ji 0        ; fim do programa
```

Obs.:

- Ao executar o projeto do processador no logisim é necessário carregar o arquivo ***program_code*** (presente no diretório de entregue) na memória RAM;
- Ambos os diagramas presentes no relatório estão no formato PDF no mesmo diretório;
- O arquivo com extensão .asm que contém o código em Assembly implementado também está presente no diretório.