UNIOESTE - UNIVERSIDADE ESTADUAL DO OESTE DO PARANÁ
CENTRO DE ENGENHARIAS E CIÊNCIAS EXATAS
CURSO DE CIÊNCIA DA COMPUTAÇÃO

Relatório do trabalho 2 e 3

Bruna Luiza Stefanelli Eyng
Guilherme Silva Altarugio
Pedro Henrique Correa Kraus

FOZ DO IGUAÇU
2024

1.COMO FOI IMPLEMENTADO

```
memo *lemem(memo *m,FILE*txt){
    int i=0;

    while (fscanf(txt, "%[^\n]%*c", m->inst[i]) != EOF) {
        m->memory[i] = i * 4;
       // printf("%d %s\n", m->memory[i], m->inst[i]);
        i++;
    }
    return m;
}
```

A função “lemem” está lendo instruções de um arquivo de texto e armazenando-as as instruções em uma matriz e seus endereços de memória em um vetor separado, os quais pertencem a uma struct assim mantendo as instruções armazenadas até o encerramento do código.

```
void criaregistradores(int registradores[]){
```

Em seguida são criados os 32 registradores e preparado com o valor inicial em 0.

```
int instMem( FILE *txt, instruct *inst,memo *m)
```

Assim, está função, “instMem” é responsável por ler uma instrução da memória de instruções e decodificá-la para preencher a estrutura das instruções. Assim, a função decodifica primeiramente o opcode, pois ele indica de qual tipo de instrução se trata, após identificados, é passado para uma função auxiliar “achabinario” atribuir o devido valor dos outros campos da instrução. Para isso foi extraído uma sequência específica de caracteres da instrução armazenada na memória, sendo passado o início (posição onde o primeiro bit do campo se encontra na string da instrução) e o fim (último bit), e forma uma nova string com esses caracteres, em seguida converte essa nova string em um valor decimal, para o valor imediato da instrução é usada uma função a parte pois cada tipo de instrução tem os bit do imediato em uma posição diferente, dessa forma para o tipo I onde todos os bit estão juntos é utilizado a mesma função usada nos outros campos “achabinario”, mas para o tipo S que o imediato está separado em duas partes é o campo de rd como auxiliar que recebem suas respectivas partes do imediato, com o uso da função “achabinario”, já na instrução de tipo B onde os bits do imediato estão dispersos é criado uma string auxiliar onde ela acha o bit mais significativo na instrução e replica até completar os outros 32bits, se aloca bit a bit organizado na string auxiliar, que é

convertida para decimal através da instrução “toDecimal”. Por fim, a função retorna 1 se foi lida e decodificada corretamente. Se houve um problema na comparação, retorna -2.

```
long long int toDecimal(char *binario)
```

Converte uma representação binária de um número, que pode estar no formato de complemento de dois, em um valor decimal. E trabalha com os números negativos em complemento de dois, caso for necessário.

```
int achabinario(char string[], int inicio, int fim)
```

Assim, a próxima função precisa extrair partes de uma string que representa um número binário e convertê-las para um valor binário inteiro.

```
void Alucontrol(instruct *inst, Control *r,int resgistradores[],int memoria[], FILE * mem)
```

Em seguida foi necessário verificar o tipo de instrução de acordo com o opcode, para fazer a devida execução da instrução, passando os armazenado nos sinais de controle os respectivos comandos (0, 1, 2), significando se serão utilizados pela instrução ou não.

```
void alu(instruct *inst,int resgistradores[],int memoria[],FILE* mem)
```

E por último a função que realiza a verificação a o opcode, fun3 ou func7 de acordo com o formato da instrução, assim é feito as operações que cada instrução possui, como os cálculos com os registradores e endereços de memória.

Algumas funções também chamam outra como é o caso da função AluControl que chama a Alu para verificar quais os sinais de controle que pertencem a cada instrução que está sendo executada. (Figura1).

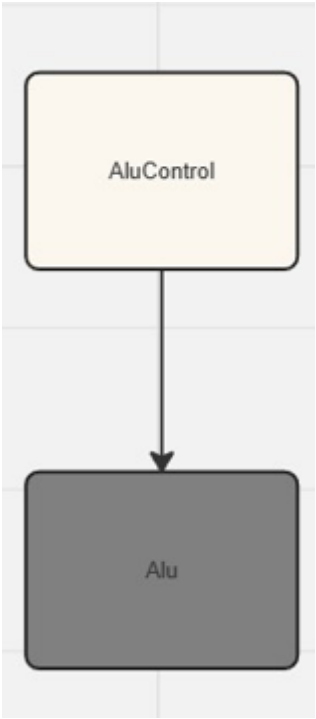

Figura 1

Outro caso é a função InstMem que chama a AchaBinário (Figura2)

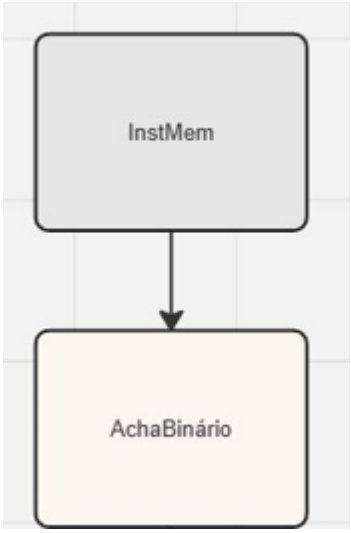

Figura 2

2. INSTRUÇÕES PARA USO DO PROGRAMA(README)

Para compilar o código fonte, o usuário precisa executar os seguintes comandos no prompt:

- gcc *.c
- .\a.exe

O software inicia com um menu simples, com apenas uma opção de entrada, que é qual o nome do arquivo texto que o usuário deseja entrar. Após a leitura do arquivo, o programa executa as instruções e mostra os respectivos caminhos de dados e os registradores que são utilizados e logo depois os sinais de controle que cada instrução utiliza e precisa para ser executada. Em seguida, após a leitura da instrução, pede-se ao usuário digitar 1 para processar a próxima linha ou digitar 0 para sair do programa. (Figura 3).

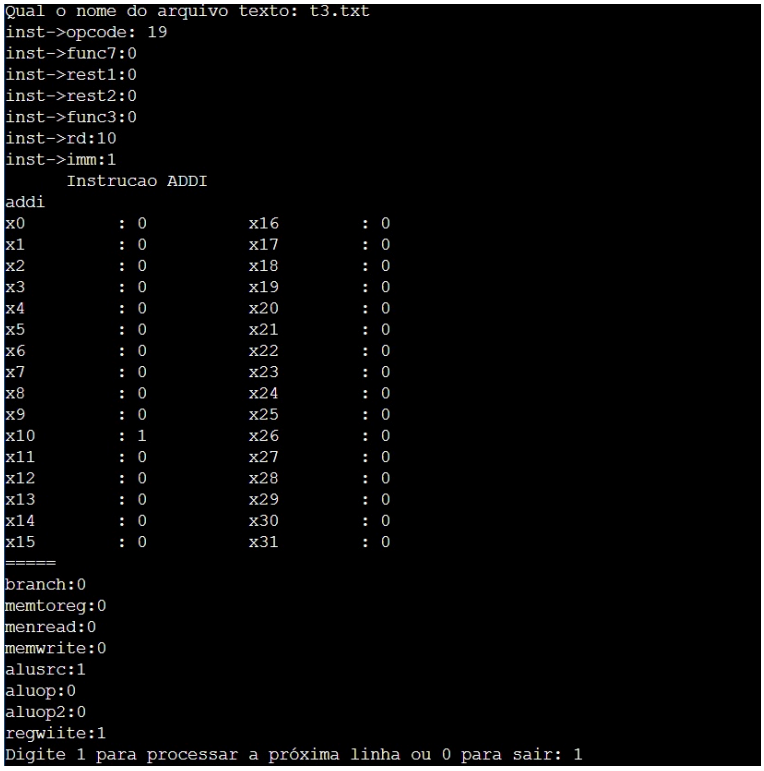

Figura 3

3.DESCRIÇÃO DA EXPERIÊNCIA: PONTOS POSITIVOS E NEGATIVOS

Foi possível aprofundar os conhecimentos sobre a arquitetura e o funcionamento do risc-v para a implementação das funções, sinais de controle, registradores e como funciona o caminho de dados para execução dos tipos de instruções. Analisar como os bits são distribuídos através de cada instrução, quais os campos que eram utilizados ou não e como o imediato seria implementado.

As dificuldades foram as implantações de funções que fazem a verificação se um número binário está em complemento de dois e converter para decimal e se o número for negativo, alterar o sinal. A lógica para as instruções que utilizam a memória como a load e a store, assim como as instruções de Branch devido a utilização do imediato para executar a instrução. E o erro do arquivo teste, influenciou na remodelagem do código, dessa forma foi preciso refazer, alterando o resultado anterior.