CCF 252 - Organização de Computadores I
Trabalho Prático 02 - Montador RISC-V

Gabriel Miranda(3857)[1], Felipe Dias(3888)[2], Mariana de Souza(3898)[3]

[1]Instituto de Ciências Exatas e Tecnológicas,
Universidade Federal de Viçosa, Florestal, MG, Brasil

1. Descrição

Neste trabalho, buscamos aplicar os conceitos do RISC-V apresentados nas aulas da disciplina de organização de computadores I, tendo como objetivo a implementação de um montador RISC-V simples, feito em linguagem de alto nível (C), lendo um arquivo de entrada com instrções assembly (.asm), e posteriormente compilando para uma saída binária.

2. Executar o Montador

2.1. Ubuntu

No processo de realização do código foi elaborado uma *Makefile*. Para a instalação do *make* no ubuntu, inserir o comando sudo apt-get install make no terminal, se for necessário, também insira posteriormente no terminal sudo apt-get update. Após a instalação insira make no terminal, insira ./tp, onde "./tp"é o nome do executável, como está sendo representado no terminal da **figura 1.**

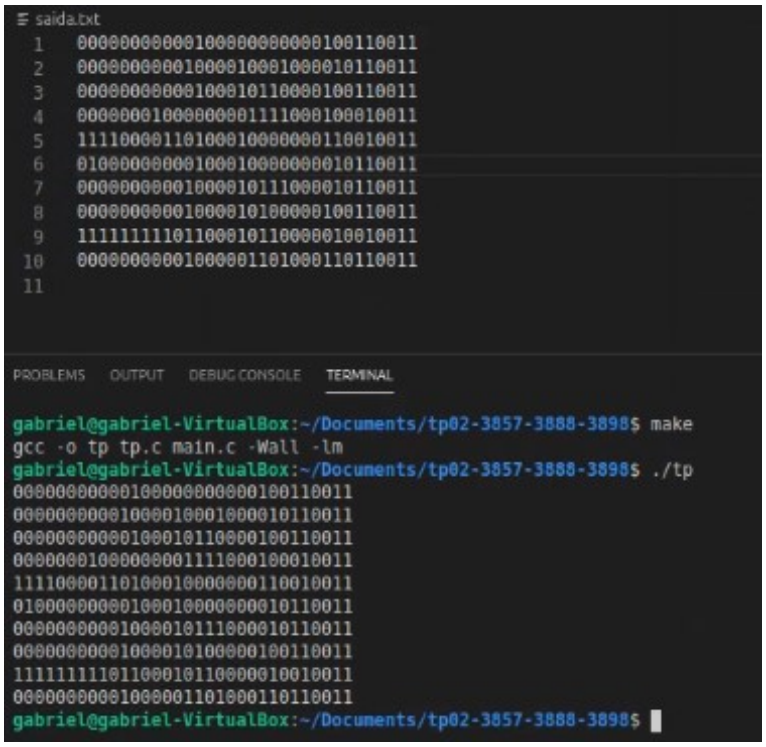

Figura 1. Apresentação do arquivo de saída e do print no terminal.

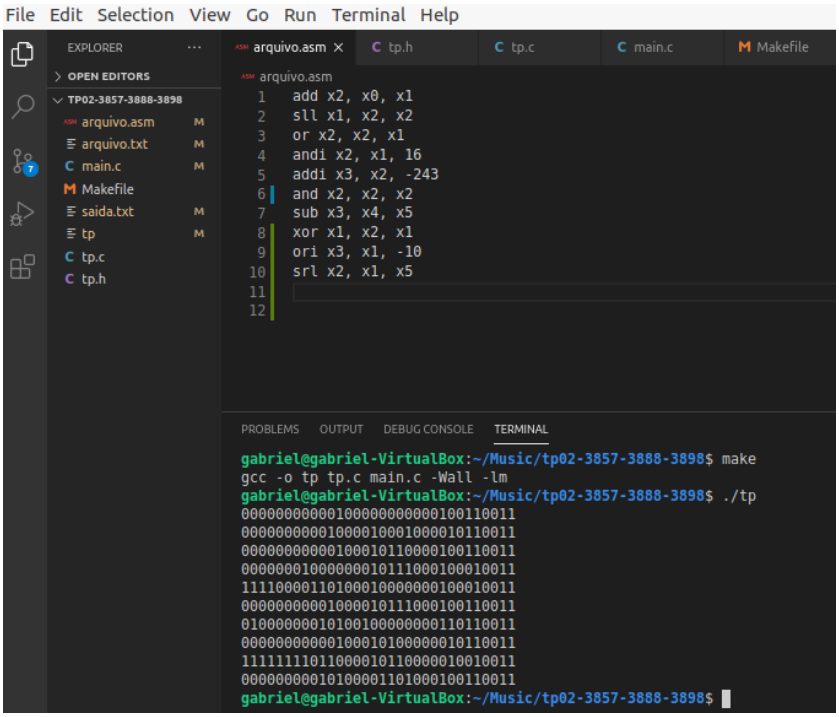

Figura 2. Apresenta o arquivo de entrada (*arquivo.asm*) com todas as instruções, também possui a organização dos arquivos, em *tp.h* que contém as chamadas das funções, *tp.c* declarando o corpo de todas as funções, *Makefile* para a compilação no terminal, *main* e o arquivo de saída (*saida.txt*).

3. Implementação do código em linguagem de alto nível(C)

3.1. Funções

```
int Elimina_X(char *rd, char *rs1, char *rs2, int n, char *instrucao){
      for (int i=0;i<n;i++){
                  if((!(strcmp(instrucao, "addi") == 0)) && (!(strcmp(instrucao, "andi") == 0)) && (!(strcmp(instrucao, "ori")
                      rd[i] = rd[i+1];
                      rs1[i] = rs1[i+1];
                      rs2[i] = rs2[i+1];
                  }
                  if((strcmp(instrucao,"addi") == 0) || (strcmp(instrucao,"andi") == 0) || (strcmp(instrucao,"ori") == 0)){
                      rd[i] = rd[i+1];
                      rs1[i] = rs1[i+1];
```

Figura 3. Implementação da função que separa string, caso a string seja ”x9,”a função separa o x da outra parte da string, como mostra o exemplo, após a aplicação a string se tornaria ”9,”.

```
int transforma_string_int(char *instrucao, int *rd_int, char *rd, int *rs1_int, char *rs1, int *rs2_int, char *rs2 ){
    char *string1;
  if((!(strcmp(instrucao, "addi") == 0)) && (!(strcmp(instrucao, "andi") == 0)) && (!(strcmp(instrucao, "ori") == 0))){
    *rd_int = strtoul(rd, &string1, 0);
    *rs1_int = strtoul(rs1, &string1, 0);
    *rs2_int = strtoul(rs2, &string1, 0);
  }
  if((strcmp(instrucao,"addi") == 0) || (strcmp(instrucao,"andi") == 0) || (strcmp(instrucao,"ori") == 0)){
    *rd_int = strtoul(rd, &string1, 0);
    *rs1_int = strtoul(rs1, &string1, 0);
    *rs2_int = atoi(rs2);
```

Figura 4. Essa função recebe uma string e transfomar em números inteiros. Diante disso, é importante ressaltar a função ”strtoul”, cujo o objetivo é de separar a parte numérica da parte de letras do vetor, entretanto , para isso acontecer o vetor de strings deve ser do tipo: números(inteiros) do lado esquerdo e caracteres do lado direito. Ex: (rd = ”98,uhg7”) , (string = ”43tp2”) . Com isso, a função ”strtoul”, transforma o lado esquerdo em inteiros e o lado e direito continua sendo do tipo char. Ex: rdint = strtoul(rd, string, 0);, no caso do ex: (rd = ”98,uhg7”) o rdint receberia o valor inteiro no caso do ”98”, ou seja, rdint = 98, e a variável string receberia ”,uhg7 ”(string = ”uhg7”).

```
int transformar_Binario(int tam, int rd_int, int rs1_int, int rs2_int, char *rd_B, char *rs1_B, char *rs2_B, char *instrucao){
    int i;
    int tam2 = 12;
    int j = 0;

    for( i = tam - 1; i>=0 ;i--){
        if(rd_int%2 == 0){
            rd_B[i] = '0';
            rd_int = rd_int/2;
        }
        else{
            rd_B[i] = '1';
            rd_int = rd_int/2;
        }
```

Figura 5. Função recebe os valores inteiros da função apresentada anteriormente e retorna valores binários, que serão posteriormente os binários apresentados na saída do terminal.

```
if( (strcmp(instrucao, "addi") == 0 ) || (strcmp(instrucao, "andi") == 0) || (strcmp(instrucao, "ori") == 0)){

    if (rs2_int >= 0){
    for(i=(tam2-1); i>=0; i--){
        if (rs2_int% 2 == 0 ) {
            rs2_B [i] = '0' ;
        }
        else {
            rs2_B [i] = '1' ;
        }
        rs2_int = rs2_int / 2 ;

    }
}
 if (rs2_int < 0){
    for(i=(tam2-1); i>=0; i--){
        if (rs2_int% 2 == 0 ) {
            rs2_B [i] = '1' ;
        }
        else {
            rs2_B [i] = '0' ;
        }
        rs2_int = rs2_int / 2 ;
```

Figura 6. Essa parte da função *transformar Binario* analisa e diferencia registradores de imediatos dependendo do tipo da instrução na linha lida. No caso de imediatos, reconhece e distingue positivos e negativos, aplicando para estes a conversão em binário por complemento de 2.

```c
int main(){

    int n=5;
    char rd_B[n], rs1_B[n], rs2_B[n];
    int rd_int, rs1_int, rs2_int;
    char rd[n], rs1[n], rs2[n], instrucao[n];
    char immediat[5], immediat_B[12];
    int immediat_int;
    setlocale(LC_ALL,"");

    FILE *arquivo;
    arquivo = fopen("arquivo.asm","r");
    if(arquivo == NULL){
        printf("Não se pode abrir o arquivo...");
    }
    FILE *saida;
    saida = fopen("saida.txt","w+");

   //"While" está percorrendo cada linha do arquivo de entrada.
    while (fscanf(arquivo,"%s %s %s %[^\n]s", instrucao, rd, rs1, rs2) != EOF){

        //Condição caso seja tipo R.
        if((strcmp(instrucao,"add") == 0) || (strcmp(instrucao,"sub") == 0) || (strcmp(instrucao,"and")

            Elimina_X(rd,rs1,rs2,n, instrucao);
            transforma_string_int(instrucao,&rd_int, rd, &rs1_int, rs1, &rs2_int, rs2);
            transformar_Binario(n, rd_int, rs1_int, rs2_int, rd_B, rs1_B, rs2_B, instrucao);
            imprime_comando_binario(saida, instrucao, rd_B, rs1_B, rs2_B, immediat_B);

        }

        //Condição caso seja tipo I.
        if((strcmp(instrucao, "addi") == 0) || (strcmp(instrucao, "andi") == 0) || (strcmp(instrucao, "

            strcpy(immediat, rs2);
            Elimina_X(rd,rs1,immediat,n, instrucao);
            transforma_string_int(instrucao, &rd_int, rd, &rs1_int, rs1, &immediat_int, immediat);
            transformar_Binario(n, rd_int, rs1_int, immediat_int, rd_B, rs1_B, immediat_B, instrucao);
            imprime_comando_binario(saida, instrucao, rd_B, rs1_B, rs2_B, immediat_B);
```

Figura 7. Na *main* foram elaboradas condições para efetuar a apresentação dos binários no terminal, e no arquivo saída, onde, cada condição contém instruções do tipo I ou do tipo R, incluindo as chamadas das funções contidas no *tp.c* e *tp.h* .

4. Implentação das funções do código em pseudocódigo

4.1. Pseudocódigo das funções

Abaixo segue a amostra completa em pseudocódigo das funções, referenciadas pela main do código implementado para o tp, para melhor análise das etapas de transformações do montador RISC-V deste trabalho.

```
//Função que elimina o x da string. Exemplo: x9, -> Aplicando a função "Elimina_x" ficaria 9,
Função Elimina_X(string endereço rd, string endereço rs1, string endereço rs2, inteiro n, string endereço instrucao)
Começo:
       Para i=0 e enquanto i<n faça:
                  Se (instrucao != "addi") e (instrucao != "andi") e (instrucao != "ori") faça:
                       rd[i] = rd[i+1]
                       rs1[i] = rs1[i+1]
                       rs2[i] = rs2[i+1]
                  Fim do se
                  Se (instrucao == "addi") ou (instrucao == "andi") ou (instrucao == "ori") faça:
                       rd[i] = rd[i+1]
                       rs1[i] = rs1[i+1]
                  Fim do se
              Fim do para

Fim da Função Elimina_X

//Função para tranformar todas strings em números inteiros.
Função transforma_string_int(string endereço instrucao, inteiro endereço rd_int, string endereço rd, inteiro endereço
rs1_int, string endereço rs1, inteiro endereço rs2_int, string endereço rs2 ) Começo:
    Declara string endereço string1
  Se (instrucao != "addi") e (instrucao != "andi") e (instrucao != "ori") faça:

    Endereço rd_int = parte inteira de rd
    string1 = parte string de rd
    Endereço rs1_int = parte inteira de rs1
    String1 = parte string de rs1
    Endereço rs2_int = parte inteira de rs2
    String1 = parte string de rs2
  Fim do se
  Se (instrucao == "addi") ou (instrucao == "andi") ou (instrucao == "ori") faça:

    Endereço rd_int = parte inteira de rd
    string1 = parte string de rd
    Endereço rs1_int = parte inteira de rs1
    String1 = parte string de rs1
    Endereço rs2_int = inteiro rs2

  Fim do se
```

Figura 8.

```
Fim da Função transforma_string_int

//Função que tranforma os decimais em binários.
Função transformar_Binario(inteiro tam, inteiro rd_int, inteiro rs1_int, inteiro rs2_int, string endereço
rd_B, string endereço rs1_B, string endereço rs2_B, string endereço instrucao) Começo:
    Declara inteiro i
    Declara inteiro tam2 = 12
    Declara inteiro aux = 0

        Para i=tam-1 e enquanto i>=0 faça:
          Se (rd_int%2 == 0) faça:
            rd_B[i] = '0'
            rd_int = rd_int/2
         Fim do se
         Se não faça:
            rd_B[i] = '1'
            rd_int = rd_int/2
        Fim do se não
      Fim do para

      Para i=tam-1 e enquanto i>=0 faça:
            Se (rs1_int%2 == 0) faça:
            rs1_B[i] = '0'
            rs1_int = rs1_int/2
        Fim do se
        Se não faça:
            rs1_B[i] = '1'
            rs1_int = rs1_int/2
        Fim do se não
      Fim do para
    Se (instrucao != "addi") e (instrucao != "andi") e (instrucao != "ori") e (instrucao != "ld") e (instrucao
!= "sd") faça:

      Para i=tam-1 e enquanto i>=0 faça:
            Se (rs2_int%2==0) faça:
                rs2_B[i] = '0'
                rs2_int = rs2_int/2
            Fim do se
            Se não faça:
```

Figura 9.

```
            rs2_B[i] = '1'
            rs2_int = rs2_int/2
        Fim do se não
   Fim do para

Fim do se


//Aplicando condição para o immediate positivo.
Se (instrucao == "addi") ou (instrucao == "andi") ou (instrucao == "ori") faça:

    Se (rs2_int >= 0) faça:
    Para i=tam2-1 e enquanto i>=0 faça:
        Se (rs2_int% 2 == 0 ) faça:
            rs2_B [i] = '0'
        Fim do se
        Se não faça:
            rs2_B [i] = '1'
        Fim do se não
        rs2_int = rs2_int / 2

    Fim do para
Fim do se
 // Aplicando condição caso o imediato seja negativo.
 Se (rs2_int < 0) faça:
    Para i=tam2-1 e enquanto i>=0 faça:
        Se (rs2_int% 2 == 0 ) faça:
            rs2_B [i] = '1'
        Fim do se
        Se não faça:
            rs2_B [i] = '0'
        Fim do se não
        rs2_int = rs2_int / 2

    Fim do para
    Para i=tam2-1 e enquanto i>=0 faça:
        Se (i == (tam2-1)) faça:
            Se (rs2_B [i] == '1') faça:
                rs2_B [i] = '0'
                aux = 1
            Fim do se
```

Figura 10.

```
            Se não faça:
                rs2_B [i] = '1'
                aux = 0
            Fim do se não
        Fim do se
        Se não faça:
            Se (aux == 1) faça:
                Se (rs2_B [i] == '1') faça:
                    rs2_B [i] = '0'
                    aux = 1
                Fim do se
                Se não faça:
                    rs2_B [i] = '1'
                    aux = 0
                Fim do se não
            Fim do se
            Se (aux == 0) faça:
                Se (rs2_B [i] == '1') faça:
                    rs2_B [i] = '1'
                    aux = 0
                Fim do se
                Se não faça:
                    rs2_B [i] = '0'
                    aux = 0
                Fim do se não
            Fim do se
        Fim do se não
    Fim do para
  Fim do se

Fim do se

Fim da Função transformar_Binario

//Print para comandos binários do tipo R e I
Função imprime_comando_binario(string endereço instrucao, string endereço rd, string endereço rs1, string endereço
rs2, string endereço immediate) Começo:

    Declara arquivo saida
```

Figura 11.

```
saida = abre ("arquivo.txt") para adicionar escrita
Declara strings funct7[10], funct3[10], opcode[10]

Se (instrucao == "add") faça:

   funct7 = "0000000"

   funct3 = "000"

   opcode = "0110011"

   Escreva (funct7,rs2,rs1,funct3,rd,opcode \corta a linha)
   Escreva no arquivo saída (funct7,rs2,rs1,funct3,rd,opcode \corta a linha)

Fim do se

Se (instrucao == "sub") faça:

   funct7 = "0100000"

   funct3 = "000"

   opcode = "0110011"

   Escreva (funct7,rs2,rs1,funct3,rd,opcode \corta a linha)
   Escreva no arquivo saída (funct7,rs2,rs1,funct3,rd,opcode \corta a linha)
Fim do se
Se (instrucao == "and") faça:

   funct7 = "0000000"

   funct3 = "111"

   opcode = "0110011"

   Escreva (funct7,rs2,rs1,funct3,rd,opcode \corta a linha)
   Escreva no arquivo saída (funct7,rs2,rs1,funct3,rd,opcode \corta a linha)
Fim do se
Se (instrucao == "or") faça:

   funct7 = "0000000"
```

Figura 12.

```
    funct3 = "110"

    opcode = "0110011"

    Escreva (funct7,rs2,rs1,funct3,rd,opcode \corta a linha)
    Escreva no arquivo saída (funct7,rs2,rs1,funct3,rd,opcode \corta a linha)
Fim do se
Se (instrucao == "xor") faça:

    funct7 = "0000000"

    funct3 = "100"

    opcode = "0110011"

    Escreva (funct7,rs2,rs1,funct3,rd,opcode \corta a linha)
    Escreva no arquivo saída (funct7,rs2,rs1,funct3,rd,opcode \corta a linha)
Fim do se
Se (instrucao == "sll") faça:

    funct7 = "0000000"

    funct3 = "001"

    opcode = "0110011"

    Escreva (funct7,rs2,rs1,funct3,rd,opcode \corta a linha)
    Escreva no arquivo saída (funct7,rs2,rs1,funct3,rd,opcode \corta a linha)
Fim do se
Se (instrucao == "srl") faça:

    funct7 = "0000000"

    funct3 = "101"

    opcode = "0110011"

    Escreva (funct7,rs2,rs1,funct3,rd,opcode \corta a linha)
```

Figura 13.

```
        Escreva no arquivo saída (funct7,rs2,rs1,funct3,rd,opcode \corta a linha)
    Fim do se

    Se (instrucao == "addi") faça:
        funct3 = "000"

        opcode = "0010011"

        Escreva (immediate,rs1,funct3,rd,opcode \corta a linha)
        Escreva no arquivo saída (immediate,rs1,funct3,rd,opcode \corta a linha)

    Fim do se

    Se (instrucao == "andi") faça:

        funct3 = "111"

        opcode = "0010011"

        Escreva (immediate,rs1,funct3,rd,opcode \corta a linha)
        Escreva no arquivo saída (immediate,rs1,funct3,rd,opcode \corta a linha)

    Fim do se
    Se (instrucao == "ori") faça:

        funct3 = "110"

        opcode = "0010011"

        Escreva (immediate,rs1,funct3,rd,opcode \corta a linha)
        Escreva no arquivo saída (immediate,rs1,funct3,rd,opcode \corta a linha)

    Fim do se
    Fecha (saída)
Fim da Função imprime comando binario
```

Figura 14.

5. Refência Bibliográfica

- Referência comando make para ubuntu
- Guia para comandos Github
- Vscode com Github

- Livro Computer Organization and Design RISC-V Edition: The Hardware Software Interface (The Morgan Kaufmann Series)