UNIVERSIDADE FEDERAL DE RORAIMA
DISCIPLINA CONSTRUÇÃO DE COMPILADORES
PROF.: DR. LUCIANO FERREIRA
ALUNO: FELIPE DERKIAN DE SOUSA FREITAS

LISTA 1

BOA VISTA, 19 DE SETEMBRO DE 2020

Link códigos fontes: https://github.com/felipekian/compiladores_lista_1_implementacao

Lista 1 - Compiladores Felipe Kevian

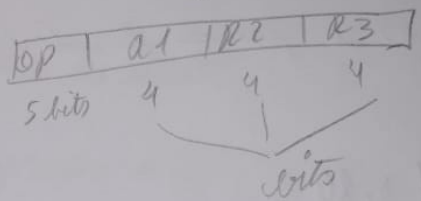
① a) Um processador com 38 instruções que podem ter referência a dois endereços de memória de 32 bits cada um.

- Para instruções serão necessários $2^6 = 64$ instruções, pois $2^5 = 32$ e não caberiam as 38 instruções.
- 32 bits para cada endereço de memória.

```
  1
 32 bits
+32 bits
  6 bits
-------
 70 bits ✗
```

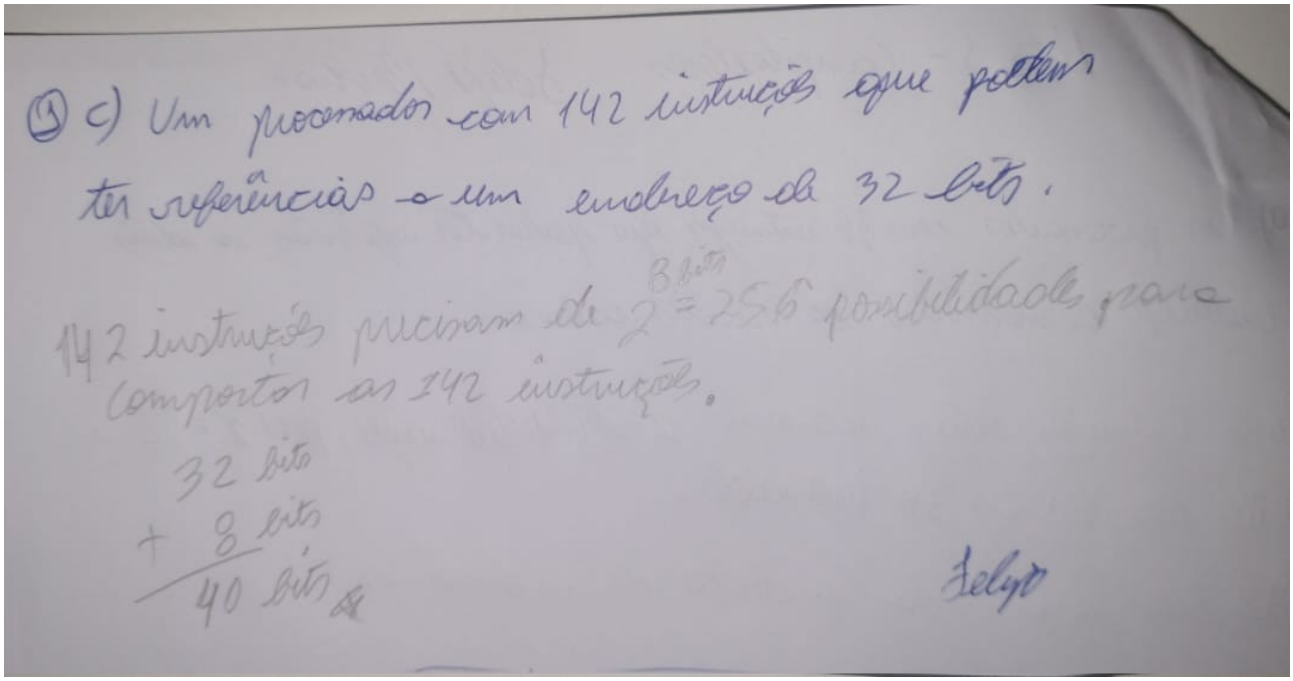
b) Um processador com 32 instruções que podem ter referências a três registradores, sendo que eles 16 registradores no processador.

32 instruções $= 2^5 = 32$
16 registradores $= 2^4 = 16$




```
 5 bits
+4 bits
 4 bits
 4 bits
-------
17 bits ✗
```

Link códigos fontes: https://github.com/felipekian/compiladores_lista_1_implementacao

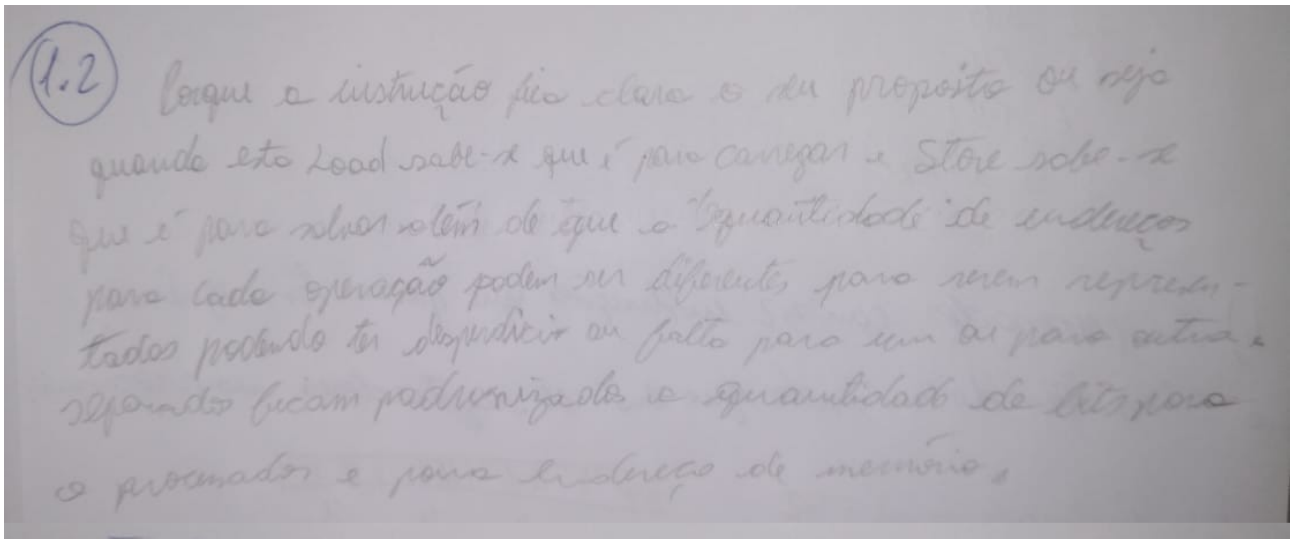


(1) c) Um processador com 142 instruções que podem ter referências a um endereço de 32 bits.

142 instruções precisam de $2^{8 \text{ bits}} = 256$ possibilidades para comportar as 142 instruções.

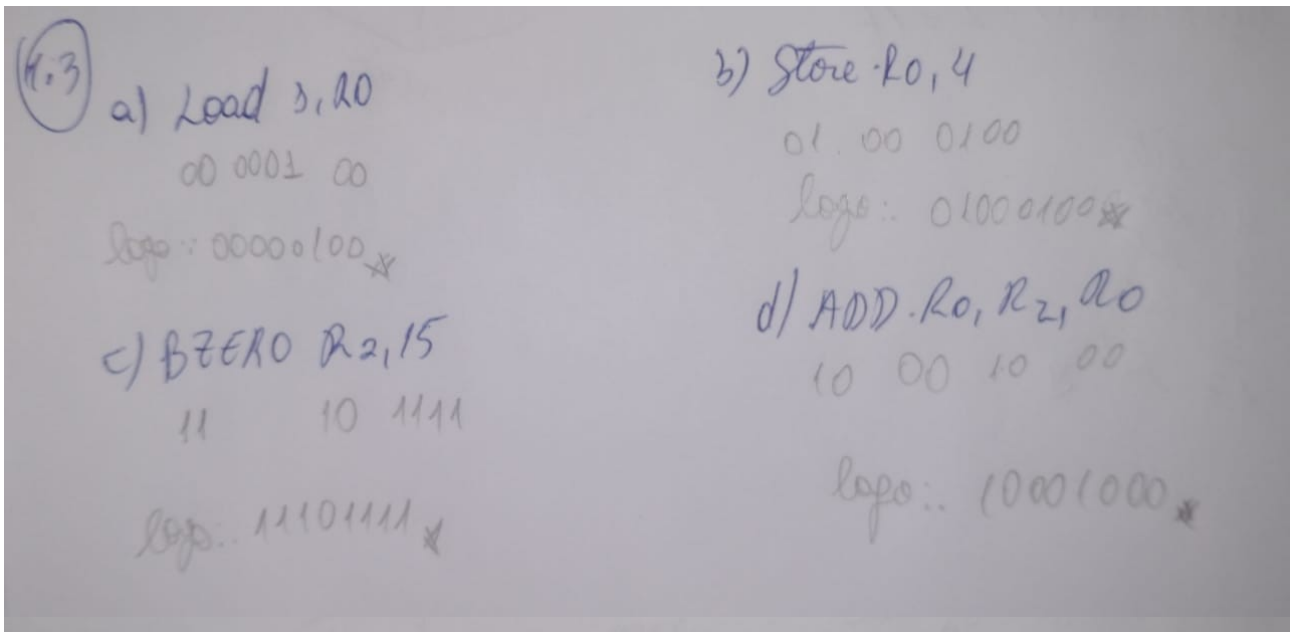
32 bits
+ 8 bits
40 bits

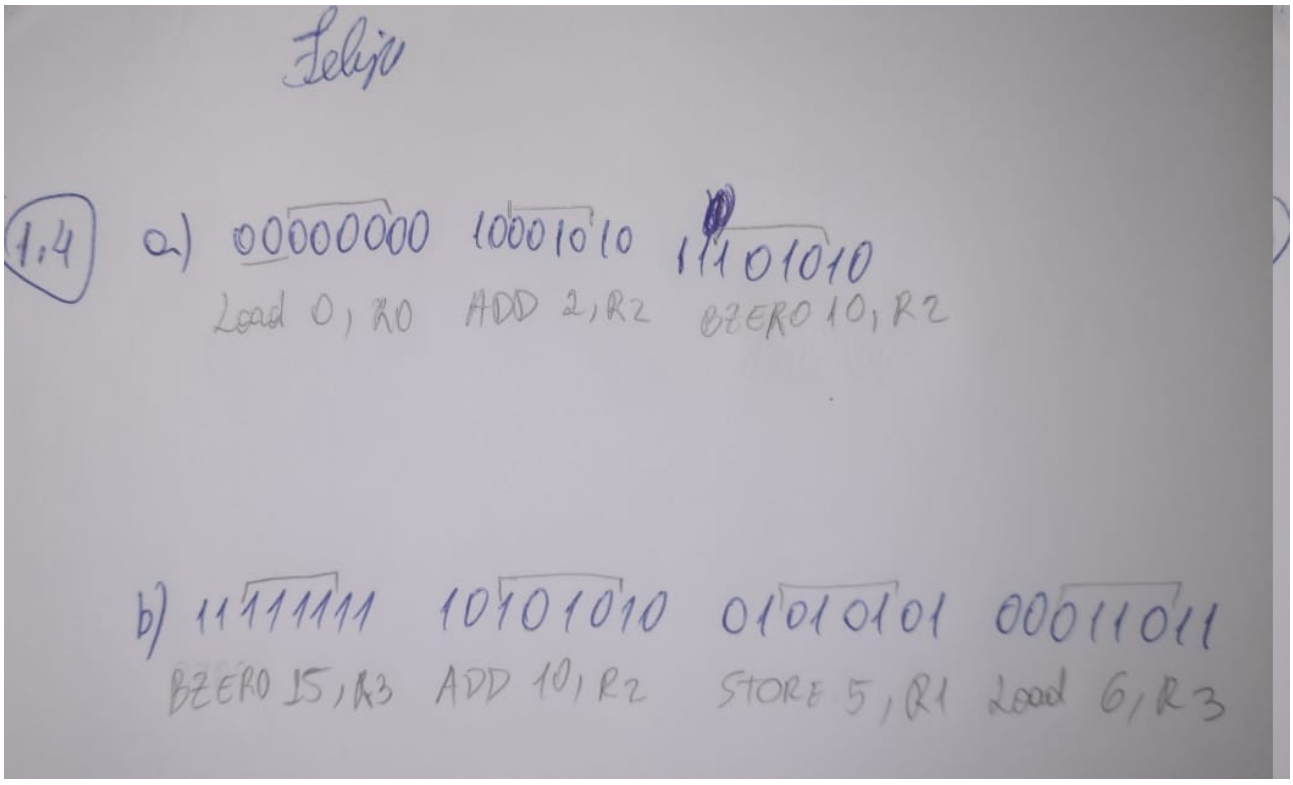
Felipe



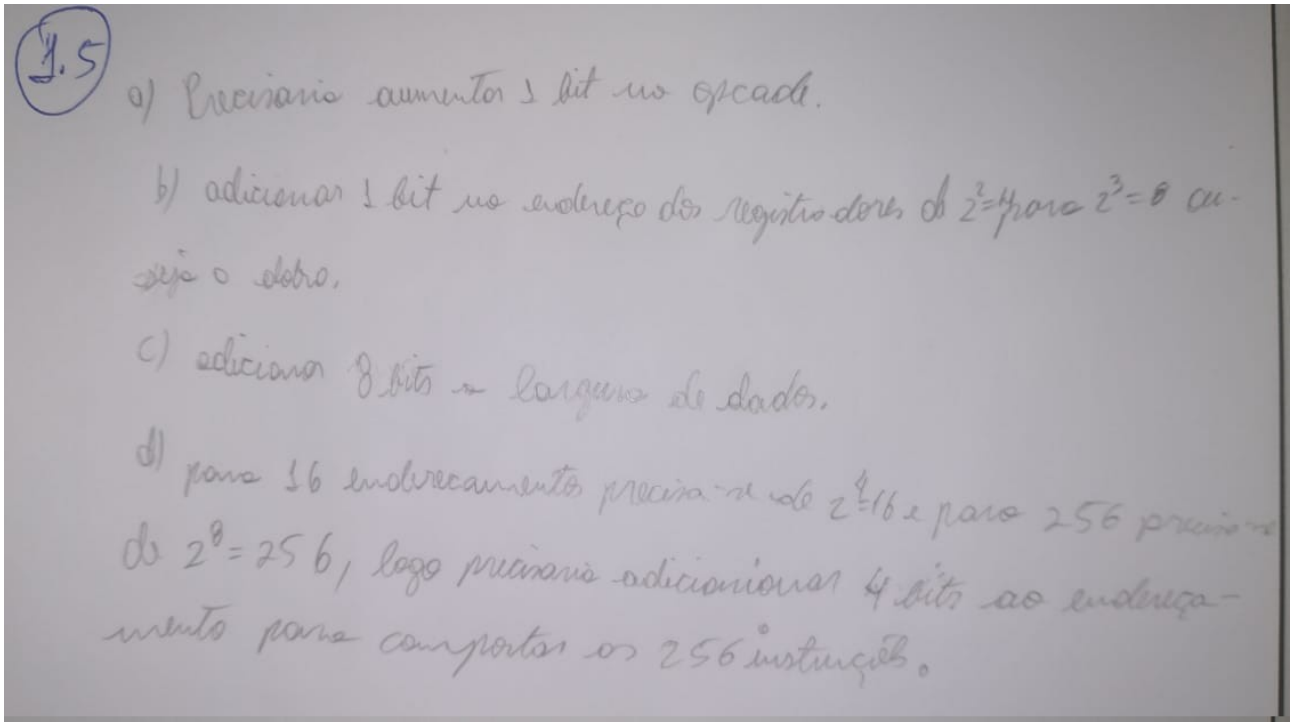
(1.2) Porque a instrução fica clara o seu propósito ou seja quando esta Load sabe-se que é para carregar e Store sabe-se que é para salvar além de que a quantidade de endereços para cada operação podem ser diferentes para serem representados podendo ter desperdício ou falta para um ou para outro. Separados ficam padronizados a quantidade de bits para o processador e para endereço de memória.

Link códigos fontes: https://github.com/felipekian/compiladores_lista_1_implementacao





Link códigos fontes: https://github.com/felipekian/compiladores_lista_1_implementacao



Link códigos fontes: https://github.com/felipekian/compiladores_lista_1_implementacao

1.6) Um programa em C ou C ++ permite a passagem de argumentos da linha de comando por meio de dois parâmetros da função main:

O primeiro parâmetro, que tipicamente recebe o nome argc (argument count), indica o número de palavras (separadas por espaços) presentes na linha de comando, incluindo o próprio nome do programa. O segundo parâmetro, cujo nome típico é argv (argument value), é um arranjo de ponteiros para caracteres, onde cada elemento do arranjo representa uma das palavras da linha de comando. Com o uso desses argumentos, desenvolva um programa em C ++ para apresentar na saída padrão o conteúdo de um arquivo cujo nome é fornecido na linha de comando.

```
#include <iostream>
#include <stdio.h>
#include <string.h>

using namespace std;

int main(int argc, char const *argv[])
{
   // valida a quantidade de parametros
   if (argc > 2)
   {
      cout << "Informe somente o nome de um arquivo." << endl;
      return 0;
   }

   // abre o arquivo
   FILE *arq = fopen(argv[argc-1], "r");

   // valida se arquivo existe
   if(!arq){
      cout << "Arquivo não encontrado." << endl;
   } else {
      cout << "Arquivo encontado... \n\n---------- LENDO ----------\n\n";
   }

   // ler o arquivo enquanto não for o final do arquivo.
   while(!feof(arq)){
      char v;
      fscanf(arq, "%c", &v);
      cout << v;
   }

   // fecha o arquivo
   fclose(arq);

   // mensagem de fim
   cout << "\n\n---------- FIM ----------" << endl;
```

Link códigos fontes: https://github.com/felipekian/compiladores_lista_1_implementacao

```
    return 0;
}
```

Link códigos fontes: https://github.com/felipekian/compiladores_lista_1_implementacao

1.7) Com o uso de argc e argv, definidos anteriormente, desenvolva um programa em C ++ para implementar a cópia do conteúdo de um arquivo, cujo nome é passado como o primeiro argumento para o programa na linha de comando, para outro arquivo, cujo nome é passado como o segundo argumento na linha de comando.

```
#include <iostream>
#include <stdio.h>
#include <string.h>

using namespace std;

int main(int argc, char const *argv[])
{
   // valida a quantidade de parametros
   if (argc != 3)
   {
      cout << "Informe somente o nome de um arquivo de origem e de destino." << endl;
      return 0;
   }

   // abre o arquivo
   FILE *arq = fopen(argv[argc-2], "r");
   FILE *arq_final = fopen(argv[argc-1], "w");

   // valida se arquivo existe
   if(!arq){
      cout << "Arquivo não encontrado." << endl;
   } else {
      cout << "Arquivo encontado... \n\n---------- LENDO ----------\n";
   }

   // ler o arquivo enquanto não for o final do arquivo.
   while(!feof(arq)){
      char v;
      // ler do arquivo 1
      fscanf(arq, "%c", &v);
      // salva no arquivo 2
      fprintf(arq_final, "%c", v);
   }

   // fecha o arquivo
   fclose(arq);
   fclose(arq_final);

   // mensagem de fim
   cout << "\n---------- COPIA FINALIZADA ----------" << endl;

   return 0;
}
```

Link códigos fontes: https://github.com/felipekian/compiladores_lista_1_implementacao

1.8) Com o uso de argc e argv, definidos anteriormente, desenvolva um programa em C ++ para contar o número de caracteres, palavras e linhas no arquivo cujo nome foi especificado na linha de comando e apresentar esses totais na tela (saída padrão).

```
#include <iostream>
#include <stdio.h>
#include <string.h>

#define FALSE 0
#define TRUE 1

using namespace std;

int flag = FALSE;
int inicio = TRUE;

int cont_palavras = 0;
int cont_letras = 0;
int cont_linhas = 0;

// imprime o resultado
void imprimir(int linhas, int letras, int palavras)
{
   cout << "LINHAS  : " << linhas + 1 << endl;
   cout << "LETRAS  : " << letras << endl;
   cout << "PALAVRAS: " << palavras << endl;
   return;
}

// realiza a contagem das informações
void contabilizar(char v)
{
   if (v == '\n' && flag == TRUE) {
      cont_linhas++;
      cont_palavras++;
      flag = FALSE;
   } else if (v != ' ' && v != '\n' && v!='\r' && v!='\t'){
      cont_letras++;
      flag = TRUE;
   } else if ((v == ' ' || v == '\0') && flag == TRUE) {
      cont_palavras++;
      flag = FALSE;
   }
}
```

Link códigos fontes: https://github.com/felipekian/compiladores_lista_1_implementacao

```
int main(int argc, char const *argv[])
{

    // valida a quantidade de parametros
    if (argc > 2)
    {
        cout << "Informe somente o nome de um arquivo." << endl;
        return 0;
    }

    // abre o arquivo
    FILE *arq = fopen(argv[argc - 1], "r");

    // valida se arquivo existe
    if (!arq)
    {
        cout << "Arquivo não encontrado." << endl;
    }
    else
    {
        cout << "Arquivo encontado... \n\n---------- LENDO ----------\n\n";
    }

    // ler o arquivo enquanto não for o final do arquivo.
    while (!feof(arq))
    {
        if(inicio){
            inicio = FALSE;
            flag = TRUE;
        }

        char v;
        fscanf(arq, "%c", &v);
        contabilizar(v);
    }

    // fecha o arquivo
    fclose(arq);

    imprimir(cont_linhas, cont_letras, cont_palavras);

    // mensagem de fim
    cout << "\n\n---------- FIM ----------" << endl;

    return 0;
}
```

Link códigos fontes: https://github.com/felipekian/compiladores_lista_1_implementacao

1.9) Qual é o erro associado a cada uma das seguintes declarações de variáveis em um programa C ++ ? Com o auxílio de um compilador C ++ , interprete as mensagens associadas a esses erros.

(a) int do;

R: do palavra reservada da linguagem.

(b) int valor = 078;

R: valor inicializado com 0 a esquerda.

(c) char a.c = 0;

R: nome da variável com ponto (.) no meio da palavra.

(d) char b = 715.

tribuindo inteiro em uma variável do tipo char que deve receber um caractere entre aspas e não finalizando com ponto e virgula (;) e sim com ponto (.).

Link códigos fontes: https://github.com/felipekian/compiladores_lista_1_implementacao

1.10) A função atoi, da biblioteca padrão da linguagem C, permite a conversão de uma seqüência de caracteres (passada como argumento da função) para um valor inteiro (seu valor de retorno). Use essa função para implementar uma função C ++ que receba qualquer quantidade de inteiros na linha de comando e apresente na saída padrão a soma desses valores. Por exemplo, se o programa executável tem o nome de total, a execução total 1 20 100 deve apresentar na tela o valor 121.

```
#include <iostream>
#include <stdio.h>
#include <string.h>

using namespace std;

int main(int argc, char const *argv[])
{
   if( strcmp(argv[1], "total") != 0 ){
      cout << "parametros não reconhecidos" << endl;
      return EXIT_SUCCESS;
   }

   int total = argc;
   int contabilizar = 0;

   for (int i = 2; i < total; i++)
   {
      contabilizar += atoi(argv[i]);
   }

   cout << "\n" << contabilizar << endl;

   return EXIT_SUCCESS;
}
```

Link códigos fontes: https://github.com/felipekian/compiladores_lista_1_implementacao