Marcelo Heredia e Pedro Castro

Instruções de uso T2

Chamada da função:

python3 MarceloHeredia_PedroCastro.py argumento arquivo.extensao

Onde:

argumento:

-d = Decodificar para assembly mips(hex → assembly mips)

-c = Codificar para hexadecimal (assembly mips →hex)

arquivo.extensao:

é aceita qualquer extensao de arquivo (asm, txt…). Deve-se colocar o nome do arquivo com a extensão para o programa entender

Arquivo:

Colocar o arquivo de entrada na pasta IN do projeto. O arquivo com a resposta estará na pasta OUT do projeto.

Exemplos:

Chamada de codificação:

python3 MarceloHeredia_PedroCastro.py -c teste2.asm

Esta chamada pegará o arquivo teste2.asm, executará codificação do assembly mips para hexadecimal e criará um arquivo out_teste2.asm

Chamada de decodificação:

python3 MarceloHeredia_PedroCastro.py -d testesem0x.asm

Esta chamada pegará o arquivo testesem0x.asm, executará decodificação para assembly mips e criará o arquivo out_testesem0x.asm

Funcionamento e particularidades:

-Codificação:

-As labels, incluindo main, devem estar na mesma linha da próxima instrução a ser executada, não na anterior.

-Nomenclatura de labels aceitas: main e L_n onde n é igual ao numero do label

-São aceitos espaços normalmente entre label e instrução, instrução e primeiro registrador, entre as virgulas que separam os registradores e no final da linha. Porém a instrução deve estar toda na mesma linha. Ex:

```
.text
.globl main
main: lui $1,0x00001001
ori   $8 ,   $1,    0x00000000
lw $9,0x00000000($8)
```

-O codificador aceita tanto os registradores em número, conforme na foto acima, quanto com as letras conforme o simulador mars. Ex “$s1” etc..

-Decodificação:

-Na decodificação, o código em hexadecimal é aceito tanto na forma completa: 0x…. quanto removendo o ‘0x’ da frente dos números.

-Na saida gerada pelo decodificador, pode haver uma troca de numeração nas labels, o motivo é que, como o leitor não tem as labels declaradas previamente, ele as cria conforme recebe os comandos (j, beq, etc) com o endereço para onde é o salto. Essa troca não influencia no funcionamento. Exemplo:

```
1  .text
2  .globl main
3  main:addu $0,$0,$0
4  L_1:and $10,$10,$10
5  L_2:and $9,$9,$9
6  j L_2
7  j L_1
8
```

→

```
1  .text
2  .globl main
3  main:addu $0,$0,$0
4  L_2:and $10,$10,$10
5  L_1:and $9,$9,$9
6  j L_1
7  j L_2
8
```