# AOC-1: Trabalho Prático 7

## Instruções

- Use apenas instruções vistas em aula até agora (slides)
- No MARS, use a seguinte configuração:
  - No menu *Settings*, **habilite** a opção *Permit extended (pseudo) instructions and formats* e
  - **Habilite a opção *Delayed Branching***
- Seus exercícios serão corrigidos com o MARS configurado da forma descrita acima
- Utilize **EXATAMENTE** os registradores explicitados nos exercícios
  - Resultados armazenados em registradores diferentes serão considerados incorretos
- Comente seu código
- Todos os exercícios são individuais
  - Cópias detectadas resultarão em nota zero para ambos os alunos

# AOC-1: Trabalho Prático 7

## Instruções

- O material deve ser entregue pelo AVA (http://ava.ufpel.edu.br) e deverá obedecer às seguintes regras:
  - Será um arquivo compactado (**obrigatoriamente** no formato **.zip**) contendo os códigos fonte dos TPs
    - Ex: ***fulano_da_silva.zip***
  - Cada exercício deve ter o seguinte nome:
    - ***matricula_tp{n}_e{m}.asm***
    - Onde:
      - ***matricula*** é a matrícula do aluno e
      - ***{n}*** é o número do TP
      - ***{m}*** é o número do exercício
      - Ex: ***16100001_tp1_e1.asm***, ***16100001_tp1_e2.asm***, ...
- Trabalhos que não seguirem as regras a cima **NÃO SERÃO CORRIGIDOS**!
- O prazo de submissão do trabalho é até as **23:55** da próxima **terça-feira**. **NÃO** serão aceitos exercícios após a data/hora-limite.

# AOC-1: Trabalho Prático 7

## Instruções

1. Escreva um programa que remova os espaços de uma string. Por exemplo, a entrada

```
.data
                string: .asciiz "Eu amo muito meu professor de AOC-I."
```

deve produzir a string

```
"EuamomuitomeuprofessordeAOC-I."
```

Use apenas uma string (não use uma string de saída ou uma string auxiliar no seu programa). Não esqueça de terminar sua string com nulo (ver tabela ASCII para código do espaço e do \0 (null)).

A resposta deve ser a string de entrada modificada, e não uma nova string na memória, ou seja, iniciando no endereço de memória 0x10010000.

# AOC-1: Trabalho Prático 7

## Instruções

2. Escreva um programa que altere uma string para “capitalizar” a primeira letra de cada palavra. Por exemplo, a entrada

```
.data
   string: .asciiz "meu professor é o melhor"
```

deve produzir a string

```
"Meu Professor É O Melhor"
```

Assuma que a entrada possui apenas espaços e letras minúsculas. Pode haver mais de um espaço entre as palavras.

A resposta deve ser a string de entrada modificada, e não uma nova string na memória, ou seja, iniciando no endereço de memória 0x10010000.

# AOC-1: Trabalho Prático 7

## Instruções

3. Escreva um programa que leia um vetor de 10 posições (.word) da memória (começando na posição 0x10010000) e verifique se o vetor está ou não ordenado. Use o registrador $t0 como *flag*.
   Faça $t0 = 1 se o vetor estiver ordenado e $t0 = 0 caso contrário.

# AOC-1: Trabalho Prático 7

## Instruções

4. Escreva um programa que inverta a ordem dos elementos de um vetor (.word) com 5 posições. Por exemplo, a entrada: 1, 2, 3, 4, 5 deve produzir 5, 4, 3, 2, 1.

   A resposta deve ser o vetor de entrada modificado, e não um novo vetor na memória, ou seja, iniciando no endereço de memória 0x10010000.

# AOC-1: Trabalho Prático 7

## Instruções

5. Declare três vetores do mesmo tamanho:

```
.data
    tamanho:   .word 7
    vetor1:    .word -30, -23, 56, -43, 72, -18, 71
    vetor2:    .word 45, 23, 21, -23, -82, 0, 69
    soma:      .word 0, 0, 0, 0, 0, 0, 0
```

Inicialize um ponteiro para cada vetor (pseudo-instrução **la**) e faça a soma dos elementos dos vetores 2 a 2.

O vetor resultante deve ser armazenado depois dos elementos do segundo vetor.

Exemplo: `soma[i] = vetor1[i]+vetor2[i]`