UNIVALI – Universidade do Vale do Itajaí Disciplina – Arquitetura e Organização de Processadores
Avaliação 02 – Programação em linguagem de montagem

Lei atentamente as instruções abaixo:

1. Esta atividade pode ser realizada em até 3 pessoas.

2. A atividade consiste na implementação de de um programa utilizando a linguagem de montagem do MIPS (conforme as instruções a seguir).

3. Deve ser postado um relatório, com uma capa identificando a Instituição, o curso, a disciplina, o professor, o nome da atividade, os autores do trabalho e a data em que o mesmo for entregue.

4. O corpo do relatório deverá conter a resolução do problema, incluindo: **código-fonte em linguagem de alto nível** (preferencialmente C ou C++), **código-fonte em linguagem de montagem do MIPS** e **capturas de tela** que ***demonstrem claramente a execução correta das entradas e saídas realizadas via console do simulador e os resultados da execução dos programas***. Para cada exercício desenvolvido, apresentar uma discussão dos resultados.

5. Cada código fonte deve conter um cabeçalho comentado que identifique a disciplina, a atividade, o programa e os nomes dos membros do grupo. Ex:

```
# Disciplina: Arquitetura e Organização de Processadores
# Atividade: Avaliação 01 – Programação em Linguagem de Montagem
# Atividade 02
# Alunos: Ringo Starr, John Lennon, Paul McCartney
```

6. O relatório deve ser em formato PDF e postado no ambiente Material Didático, conforme instruções fornecidas em aula. O prazo para entrega do relatório é o indicado no ambiente Material Didático. Não serão aceitos trabalhos entregues em atraso.

7. **A implementação deverá apresentar resultados corretos para qualquer conjunto de dados.** Uma solução que **não execute corretamente** terá, automaticamente, um **desconto de 50% na nota**, sendo que o professor também avaliará a correção de segmentos específicos do código (controle de execução, acesso a memória,...).

8. Se forem identificados **trabalhos** com grau de **similaridade** que caracterize cópia (autorizada ou não) ou adaptação, a nota dos grupos será a **nota de um trabalho dividida** pelo número de grupos que entregou esses trabalhos similares.

Enunciado:
Implemente um programa que leia dois vetores via console e, após a leitura dos vetores, produza um terceiro vetor em que cada elemento seja a soma dos elementos de mesmo índice dos outros dois vetores. Por fim, o programa deve imprimir esse novo vetor na tela.

Leia atentamente cada requisito listado abaixo, pois o atendimento deles será considerado para fins de avaliação (o não atendimento de algum requisito implica em desconto de nota).

Requisitos:

1. Na seção de declaração de variáveis (`.data`), cada vetor deve ser declarado com 8 elementos inicializados em 0. Eles devem ser claramente identificados com nomes como `Vetor_A`, `Vetor_B` e `Vetor_C`, por exemplo.

2. O programa deve solicitar o número de elementos dos vetores aceitando no máximo vetores com 8 elementos. Para leitura, deve ser apresentada uma mensagem solicitando a entrada desse valor, indicando o seu limite máximo. Ex: "`Entre com o tamanho dos vetores (máx. = 8).`".

3. O programa deve solicitar a entrada do número de elementos até que ele seja maior que 1 e menor ou igual a 8. Ou seja, deve implementar um mecanismo de filtragem que não aceite entrada diferente da especificada. No caso de entrada inválida, o programa deve imprimir uma mensagem de advertência antes de solicitar novamente a entrada. Ex: "`Valor inválido`".

4. O programa deve primeiramente ler todos os elementos do `Vetor_A` antes de iniciar a leitura dos elementos do `Vetor_B` . Para leitura, o programa deve solicitar ao usuário a entrada de cada elemento do vetor, um a um, com mensagens do tipo:

```
Vetor_A[0] =
Vetor_A[1] =
...

Vetor_B[0] =
Vetor_B[1] =
...
```

5. Após a entrada dos elementos dos dois vetores, o programa deve somar os elementos de mesmo índice dos dois vetores e armazenar o resultado de cada operação no elemento de posição correspondente do `Vetor_C`.

6. Ao final, o programa deve apresenta ao usuário os elementos do terceiro vetor com mensagens do tipo:

```
Vetor_C[0] =
Vetor_C[1] =
...
```

7. Os programas devem ser escritos respeitando o estilo de programação ASM, usando tabulação para organizar o código em colunas (rótulos, mnemônicos, operandos e comentários).

8. Procure comentar ao máximo o seu código. Isto é um hábito da programação *assembly*.