Universidade Federal de Minas Gerais – UFMG
Instituto de Ciências Exatas – ICEX
Departamento de Ciência da Computação – DCC
Trabalho Prático de OC2 – 2º Semestre Letivo de 2016
Professor: Omar Paranaiba Vilela Neto
Monitores: Mateus Tymburibá e Thiago Rodrigues

Especificação detalhada da 1ª entrega do Trabalho Prático

A 1ª etapa do trabalho prático consiste em implementar, na linguagem de descrição de hardware Verilog, os estágios *Fetch*, *Decode* e *Execute d*a versão multiciclos do microprocessador MIPS de 32 bits. Essa implementação deve ser gravável em uma placa de FPGA DE2-115 (família Cyclone IV) ou DE2 (família Cyclone II) e suportar, no mínimo, as seguintes instruções:

- add (soma)
- addi (soma com imediato/constante)
- sub (subtração)
- lw (load word)
- sw (store word)
- and (and bit a bit)
- andi (and bit a bit com imediato/constante)
- or (or bit a bit)
- ori (or bit a bit com imediato/constante)
- nor (nor bit a bit)
- xor (xor bit a bit)
- slt (set if less than)
- slti (set if less than imediato/constante)
- sll (shift left logical)
- srl (shift right logical)
- beq (branch on equal)
- bne (branch on not equal)
- j (jump)

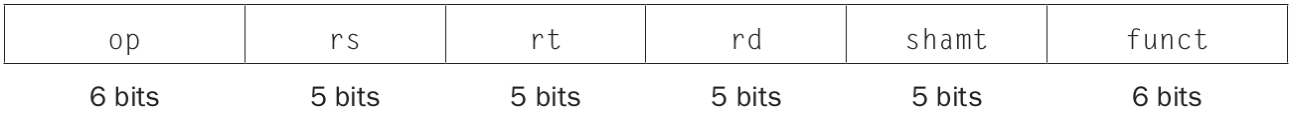
A fim de garantir a consistência das diversas implementações a serem elaboradas pelos grupos, abaixo estão reproduzidos os detalhes de projeto da versão multiciclos do MIPS32, conforme disponível em [1].

Formato das instruções

As seguintes expressões serão usadas para indicar a finalidade dos campos das instruções:

- op: código de operação (*opcode*)
- rs: 1º registrador-fonte
- rt: 2º registrador-fonte
- rd: registrador-destino
- shamt: quantidade de bits a serem deslocados (*shift amount*)
- funct: código de função (indica operação específica da ULA)

Formato R:



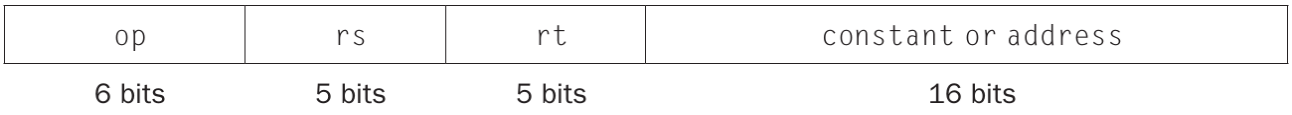
- Instruções add, sub, and, or, nor, xor e slt:

Executam a operação indicada pelos campos “op” e “funct” da instrução sobre os registradores indicados nos campos “rs” e “rt”. Guardam o resultado no registrador indicado pelo campo “rd” da instrução.

- Instruções sll e srl:

Nessas instruções, o campo correspondente ao 2º operando (rt) deve ser desprezado (sem efeito). O número de bits do operando (rs) a serem deslocados é indicado no campo “shamt” em formato de número inteiro sem sinal.

Formato I:



- Instruções lw e sw:

Calculam um endereço de memória somando o conteúdo do registrador indicado no campo “rs” ao valor imediato (constante/endereço) indicado na instrução (bits 15 a 0). Para isso, o valor imediato é expandido para 32 bits através da extensão de seu sinal (replicação do bit 15 nos bits mais significativos: bits 31 a 16).

lw: o valor lido da memória é gravado no registrador indicado pelo campo “rt” da instrução.

sw: o valor armazenado no registrador indicado pelo campo “rt” da instrução é gravado na memória.

- Instruções andi, ori, slti:

Executam a operação indicada pelos campos “op” e “funct” da instrução sobre o registrador indicado no campo “rs” e o valor imediato. Para isso, o valor imediato é expandido para 32 bits através da extensão de seu sinal (replicação do bit 15 nos bits mais significativos: bits 31 a 16). Essas instruções guardam o resultado no registrador indicado pelo campo “rt”.

- Instruções beq e bne:

Comparam o conteúdo dos registradores indicados nos campos “rs” e “rt” da instrução. Para isso, efetuam uma subtração entre o conteúdo desses registradores. Se o resultado for igual a zero (registradores com conteúdo igual), o sinal de saída “Zero” da ULA é ativado (valor 1). Caso contrário, o sinal de saída “Zero” da ULA é desativado (valor 0). Esse sinal de saída é usado para definir se o desvio condicional deve ser efetuado ou

não. Abaixo, na seção Caminho de Dados, é possível observar o uso de uma porta “and” para avaliar a condição de instruções “beq”. Os alunos deverão modificar esse caminho de dados para incluir suporte à instrução “bne”.

O endereço de destino do eventual desvio é calculado somando o valor imediato da instrução ao valor de PC + 1 (PC é o registrador Contador de Programa, *Program Counter*, que guarda o endereço da instrução a ser executada). Para isso, o valor imediato é expandido para 32 bits através da extensão de seu sinal (replicação do bit 15 nos bits mais significativos: bits 31 a 16).

OBS: como a memória especificada para este trabalho endereça apenas palavras de 4 bytes, ao contrário do MIPS que endereça cada byte da memória, o endereço de PC sempre deverá ser acrescido de 1 unidade em todas as instruções (ao invés de somar 4 unidades, como é mostrado no livro).

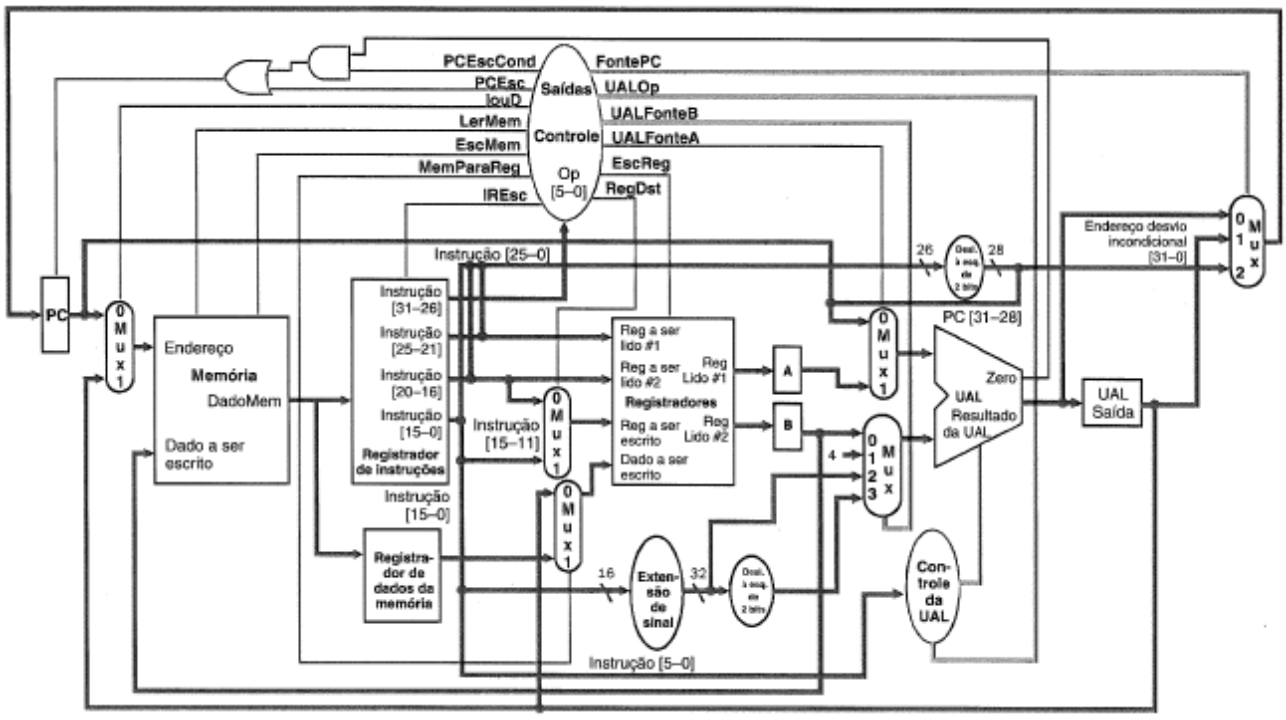
Formato J:

op	constante
6 bits	26 bits

- Instrução j:

Executa um desvio incondicional concatenando a constante (valor imediato indicado na instrução), deslocada 2 bits à esquerda, com os 4 bits mais significativos de PC + 1.

Caminho de dados original do MIPS (*data path*)



OBS: algumas instruções, como “sll”, “slr” e “bne”, poderão exigir ajustes no *data path* apresentado. A constante 4 deve ser substituída pela constante 1.

Sinais de Controle

Os sinais de controle de uma máquina multiciclos podem ser representados através de uma máquina de estados finitos. Abaixo está representada a máquina de estados finitos que representa os sinais de controle para o MIPS32.

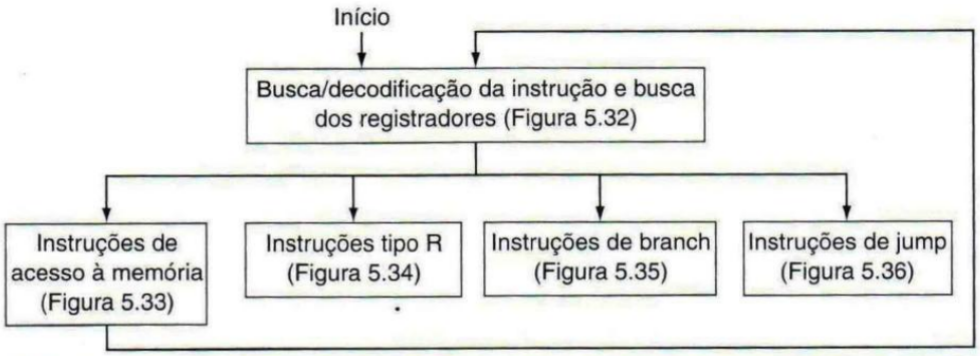


FIGURA 5.31 A visão de alto nível do controle da máquina de estados finitos. As primeiras etapas são independentes da classe de instrução; depois, uma série de seqüências que dependem do opcode da instrução é usada para completar cada classe de instrução. Após completar as ações necessárias para essa classe de instrução, o controle retorna para buscar uma nova instrução. Cada retângulo nesta figura pode representar um ou vários estados. O arco rotulado como *Início* indica o estado onde começar quando a primeira instrução está para ser buscada.

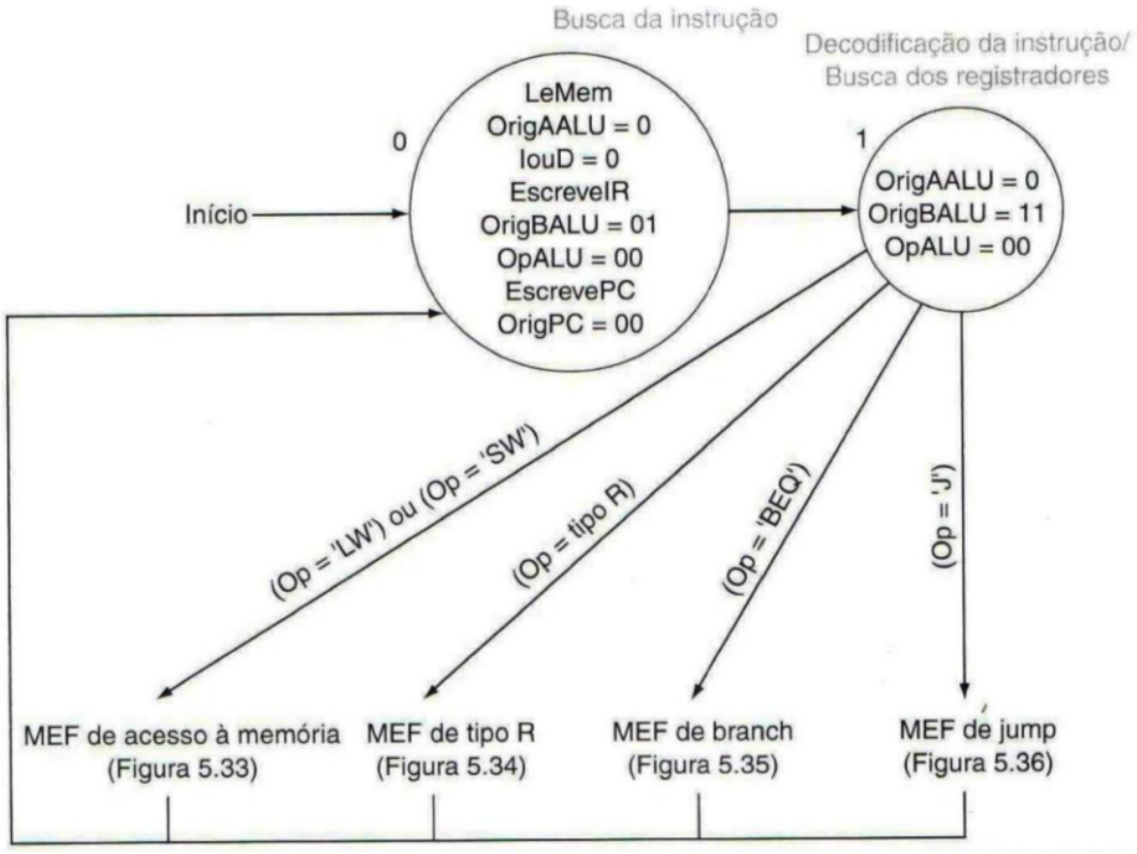


Figura 5.32 – Sinais de controle para Busca e Decodificação de instrução em MIPS multiciclos.
OBS: Sinal que não aparece é "don't care". Sinal que só aparece o nome significa que está ativo.

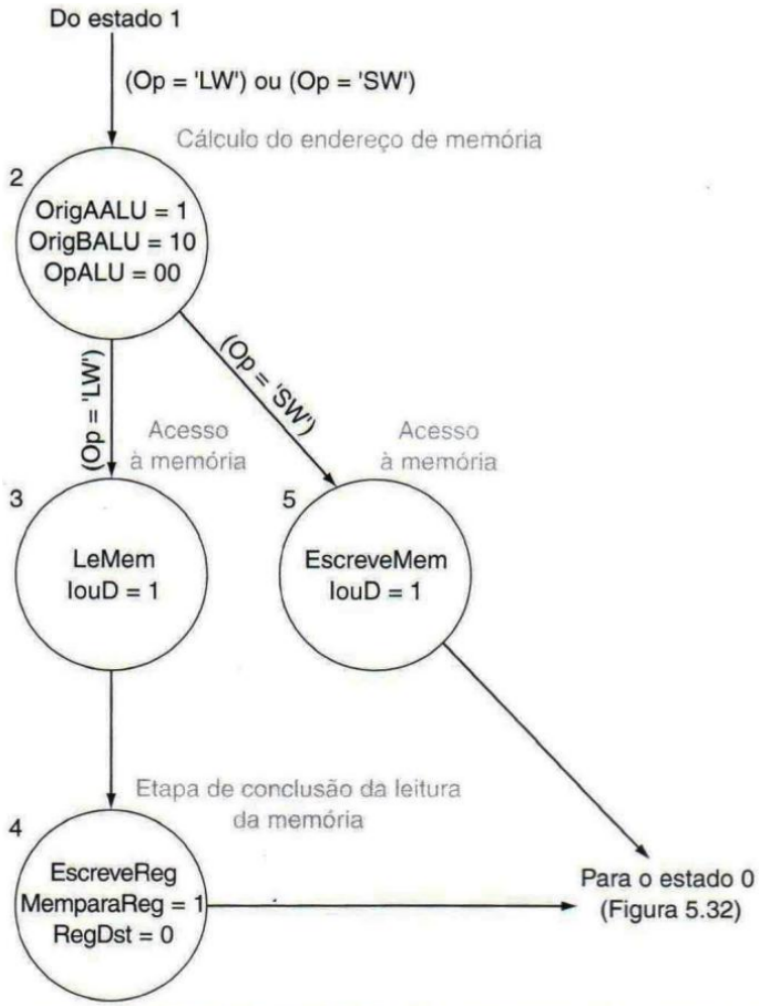


Figura 5.33

Figura 5.34

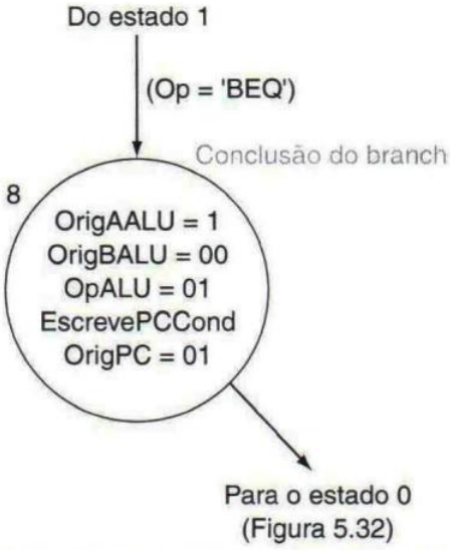
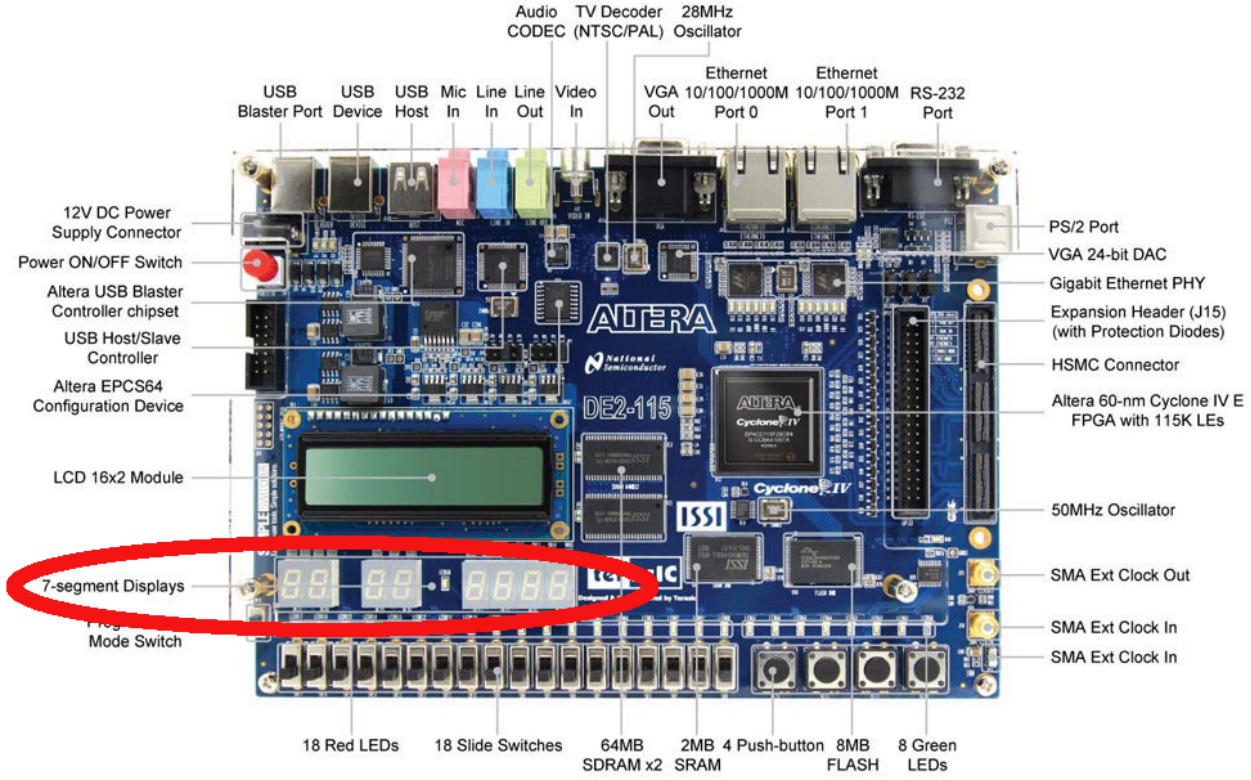


Figura 5.35

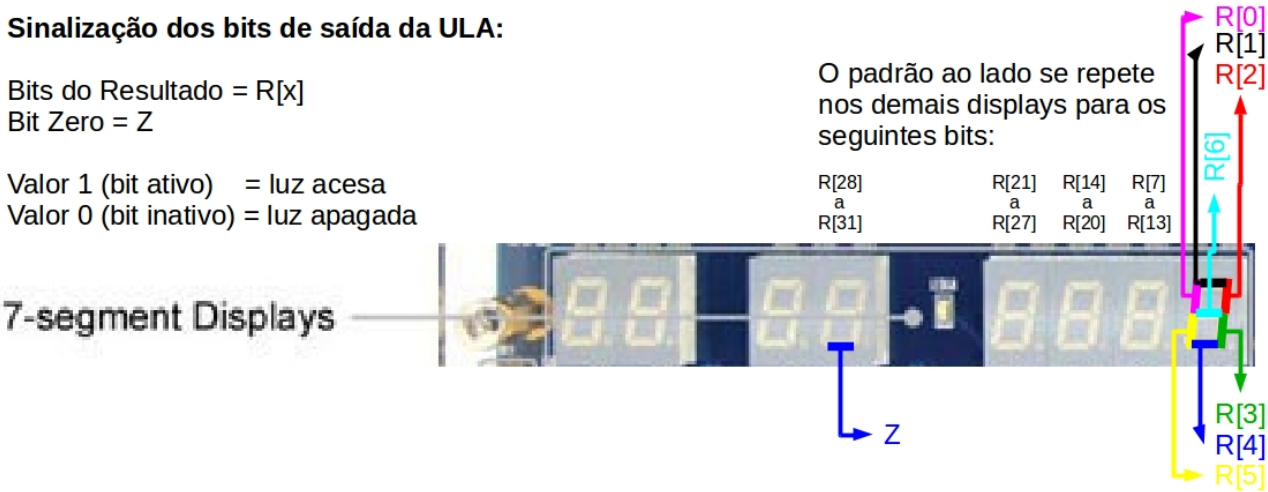
Análise dos resultados

Nesta etapa do trabalho, os valores de saída da ULA serão utilizados para avaliar a correção da implementação dos 3 estágios iniciais do MIPS multiciclos (IF, ID e EX). Portanto, as implementações em Verilog devem utilizar mecanismos de saída que informem ao avaliador os 32 bits do campo “resultado” da ULA, além do bit “zero” de saída da ULA.

Versões do código voltadas para simulação no ModelSim podem utilizar funções de impressão na tela como $monitor e $display para facilitar a visualização dos resultados. Quando utilizadas pelo grupo, essas versões devem ser submetidas para os avaliadores. Contudo, é obrigatória a criação de uma versão do código para execução nas placas FPGA. Nesse caso, as saídas da ULA devem ser mapeadas para os displays numéricos de 7 segmentos existentes na placa, da seguinte maneira:



Indicação da localização na placa FPGA dos displays numéricos de 7 segmentos.



Padronização de exibição das saídas da ULA nos displays.

Os alunos deverão criar conjuntos de testes para verificar o funcionamento de cada uma das instruções em diversos cenários. Esse conjunto de testes elaborado pelos alunos também deve ser encaminho aos avaliadores. Contudo, os avaliadores utilizarão um conjunto de testes próprio para confirmar o correto funcionamento do código.

Além disso, cada grupo deverá entregar uma versão modificada do caminho de dados exibido nesta especificação. O objetivo é adaptá-lo para oferecer suporte a todas as instruções elencadas neste enunciado.

Todo o material a ser entregue deve ser submetido por e-mail, até o fim do dia 04/10/2016, para os seguintes endereços de e-mail:

- omar@dcc.ufmg.br
- mateustymbu@dcc.ufmg.br
- thiagorbss@gmail.com

O material a ser anexado ao e-mail deverá ser reunido em uma pasta com o seguinte nome (OBS: o caractere X abaixo deve ser substituído pelo número do grupo. Os números dos grupos estão indicados no Apêndice A deste enunciado):

TP-OC2_Entrega1_GrupoX

Em seguida, essa pasta deverá ser compactada, gerando um arquivo com o seguinte nome:

TP-OC2_Entrega1_GrupoX.zip

Certifique-se de que o arquivo não está corrompido.

O título do e-mail a ser enviado deverá ser:

TP-OC2_Entrega1_GrupoX

Siga rigorosamente o padrão de nomes descrito acima.

A fim de evitar problemas na abertura de arquivos, certifique-se de que nenhum arquivo incluído na pasta a ser compactada possui caracteres especiais em seus nomes (cedilha, acentos, espaços, etc).

Os integrantes de cada grupo serão avaliados individualmente. Para isso, deverão comparecer, juntos, a uma entrevista sobre o trabalho a ser agendada posteriormente. Nessa ocasião, todos os integrantes deverão responder perguntas sobre as decisões de projeto e de implementação adotadas pelo grupo, além de demonstrar o funcionamento do código.

Referência

[1] Patterson, D. Hennessy, J. “Organização e Projeto de computadores: a interface Hardware/Software”. 3ª edição. Seção 5.5.

Apêndice A – Listagem dos grupo

Nº do grupo	Integrantes
1	Sebastião Mendes
	Luis Pedraza
	Tiago Amador
	Gabriel Noreg
	Mateus Rezende
2	Gabriel Carvalho
	Juliana Ramos
	Lucas Machado
	Marcel Henrique
	Nélio Cézar
3	Adler Melgaço Ferreira
	João Paulo Bregunci
	Marina Monteiro Moreira
	Pedro Elias Valadares Castanheiras
	Ronald Davi Rodrigues Pereira
4	Giovanni Leite
	Rafael Rubiolli
	Fernanda Ramalho
	Danilo Viana
	Manoel Junior
5	Delisson Junio
	Gabriel Oliveira
	Lucas Peixoto
	Pedro Paulo
	Rafael Grandrire
6	Jota Vicente
	Pedro Dalla
	Luiz Otávio
	Ivan Soares
	Edson Roteia
7	David Alexandre
	Diogo Leite
	Matheus Filipe Sieiro Vargas
	Matheus de Paula
8	Jéssica Cristina Carneiro
	Luiz Carlos de Oliveira
	Nathalia Campos
	Fabio Lelis
9	Andrei dos Santos Silva
	Michael Lopes
	Maelon Dias
	Ivanei Souza
10	Dourival Pimentel
	Ana Luiza de Avelar
	Julio Leandro