Enunciados dos trabalhos de Arquitetura de Computadores 2016-2.

Para cada equipe será determinado, mediante algum critério, um dos 6 enunciados abaixo. O programa correspondente deverá ser implementado, **incluindo comentários com explicações no código-fonte,** e entregue ao professor via *link* específico no Moodle até o dia **07/12/2016.**

Caso seja necessário, o professor pode chamar a equipe para explicar a implementação entregue.

Enunciado 1

Implemente um programa em *assembly* ARM7TDMI para o jogo de "Clique rápido" utilizando a interface do *plugin* EmbestBoard no ARMSim#. Este jogo tem as seguintes características:

a) Inicialmente o display de 8 segmentos está apagado e o display LCD mostra o número da rodada (1 a 6) e o tempo total acumulado (inicialmente 0 ms), no seguinte formato:

```
Rodada 1: 00.000 s
```

b) Quando o usuário clicar em qualquer um dos botões "Left" ou "Right", o display de 8 segmentos deve apresentar um dígito aleatório entre 0 e F (dica: executar a SWI 0x6d para obter o valor do "tick" do relógio interno do simulador e utilizar os 4 bits menos significativos como valor aleatório).
c) Após exibir o dígito aleatório, o display de LCD deve atualizar continuamente a contagem de tempo, atualizada a cada 50 ms, até que o usuário clique algum botão do teclado.
d) Caso o usuário clique o botão correspondente ao dígito aleatório exibido, a contagem de tempo deve ser interrompida. Esta contagem corresponde ao tempo acumulado de resposta do usuário até a rodada atual.
e) Caso o usuário clique um botão errado, o display de LCD deve mostrar "Perdeu!" indicando que o usuário perdeu.
f) Caso o usuário clique o botão correto, a rodada é incrementada e o algoritmo se repete a partir do item b) até a rodada de número 6.
g) Após a rodada de número 6, o jogo mostra no display o tempo total acumulado e é finalizado.

Enunciado 2

Implemente um programa em *assembly* ARM7TDMI que simule um relógio com alarme, utilizando a interface do *plugin* EmbestBoard no ARMSim#. O programa tem as seguintes características:

a) O relógio pode funcionar em 2 modos: alarme (caracter 'A' exibido no display de 8 segmentos) e relógio (caracter 'C' exibido no display de 8 segmentos).

b) Quando no modo relógio, o programa deve mostrar no display de LCD a hora e data atuais no formato DD-MM-AAAA HH:MM:SS e atualizar este valor a cada 1 s (usar SWI 0x6d para marcação do tempo).
c) Quando no modo alarme, o programa deve mostrar no display de LCD o horário configurado para o alarme no formato HH:MM.
d) A mudança entre os modos ocorre por meio de uma das teclas do teclado (pode ser a tecla do canto inferior direito, por exemplo).
e) Em ambos os modos, um dígito qualquer pode ser ajustado teclando-se um novo valor no teclado (0 a 9). O dígito específico a ser ajustado é marcado por meio de um cursor (caracter – ou _), exibido na linha do display LCD imediatamente abaixo do valor sendo exibido. Para navegar o cursor para a esquerda ou para a direita o usuário deve utilizar os botões “Left” ou “Right” respectivamente.

Por exemplo, caso o display LCD exiba o dado a seguir, qualquer dígito teclado ajustará o valor da dezena do minuto (marcado pelo cursor logo abaixo):

```
23-05-2014 15:10:45
                _
```

Quando o horário ajustado para o alarme e o horário atual coincidirem, os LEDs vermelhos deverão acender durante 10 segundos e depois apagar.

Obs: caso não seja possível obter data e hora atualizadas via funções do ARMSim#, pode-se utilizar uma data e hora padrão na inicialização do relógio (p. ex. 01/01/2000 12:00).

Enunciado 3

Implemente um programa em *assembly* ARM7TDMI que simule uma calculadora de números inteiros positivos que opera no modo RPN (notação polonesa inversa, ou posfixa), utilizando a interface do *plugin* EmbestBoard no ARMSim#. Este programa tem as seguintes características:

a) O teclado é utilizado para entrada de valores e de operações, conforme o arranjo abaixo:

1	2	3	+
4	5	6	-
7	8	9	*
ENTER	0	%	/

b) Ao terminar de digitar um valor numérico, o usuário deve digitar ENTER. Quando isso ocorrer, o programa deve armazenar o valor em uma pilha de operandos (dica: utilizar uma área de memória para isso, reservada antecipadamente por meio da diretiva .space).
c) Caso a pilha de operandos já esteja cheia (máx. 6), o programa não deve aceitar a entrada de valores numéricos.
d) Quando o usuário digitar uma das cinco operações definidas (+, -, *, /, %) o programa deve efetuar o cálculo com os dois valores no topo da pilha e empilhar o resultado.

e) Caso o usuário tente efetuar uma divisão ou operação de módulo por zero, o LED vermelho da direita deve acender e a operação não deve ser efetuada.
f) Caso o usuário clique sobre o botão da esquerda (“Left”), a pilha atual de operandos deve ser reiniciada (vazia).
g) O display LCD deve mostrar o estado atual da pilha, um valor por linha.

Enunciado 4

Implemente um programa em *assembly* ARM7TDMI que simule um controle de acesso baseado em senha utilizando a interface do *plugin* EmbestBoard no ARMSim#. Este controle de acesso possui três modos: “Memorização”, no qual o usuário configura uma nova senha; “Acesso”, no qual o sistema aguarda a senha previamente configurada para autorizar o acesso ou não; e “Bloqueado aguardando serviço”, que é atingido quando o usuário fornece uma senha incorreta em 3 tentativas consecutivas.

a) Inicialmente, o sistema entra no modo “Memorização”. Durante este modo, o botão da esquerda (“Left”) é utilizado para iniciar o processo de entrada da senha. Ao ser pressionado, é exibida a mensagem “Digite a nova senha para memorização” no display LCD.
b) O teclado é utilizado para entrada da senha de 8 dígitos. A cada novo dígito fornecido o display de LCD deve imprimir um novo asterisco (*). Caso o usuário queira corrigir um dígito fornecido erroneamente, deve pressionar o botão da direita (“Right”), fazendo com que o asterisco correspondente àquele dígito também seja apagado do display LCD.
c) Após digitar a senha, o usuário deve pressionar novamente o botão da esquerda (“Left”). Caso o sistema esteja no modo “Memorização”, este deve memorizar internamente a senha fornecida, o display LCD deve mostrar a mensagem “Senha memorizada” durante 5 segundos, o display de 8 segmentos deve mostrar o dígito 0 e o LED vermelho da esquerda deve acender, indicando que o sistema entrou no modo “Acesso”.
d) Durante o modo “Acesso”, o teclado é utilizado para a entrada da senha seguindo a mesma lógica apresentada no item b).
e) Caso o sistema esteja no modo “Acesso” e o botão da esquerda (“Left”) seja pressionado, a senha que foi fornecida deve ser comparada com a senha memorizada. Caso sejam iguais, o display LCD deve mostrar a mensagem “Acesso autorizado”, o LED vermelho da esquerda deve ser apagado e o sistema deve retornar ao modo “Memorização”.
f) Caso as senhas não sejam iguais, o display de LCD deve mostrar a mensagem “Senha incorreta”. O dígito mostrado no display de 8 segmentos deve ser incrementado, indicando que uma nova tentativa de senha será realizada.
g) Quando o dígito mostrado no display de 8 segmentos chegar ao valor 3, o display de LCD deve mostrar a mensagem “Número de tentativas máximo atingido!” e os dois LEDs vermelhos devem acender, indicando que o sistema entrou no modo “Bloqueado aguardando serviço”.

Enunciado 5

Implemente um programa em *assembly* ARM7TDMI que simule um jogo no qual o jogador tenta adivinhar um número de 0 a 1023 sorteado pelo sistema, utilizando a interface do *plugin* EmbestBoard no ARMSim#.

a) Inicialmente o sistema exibe o dígito 0 no display de 8 segmentos e a mensagem "Forneça um palpite (0 a 1023)" no display de LCD. Além disso, o sistema sorteia um número aleatório entre 0 e 1023 e o armazena internamente (dica: executar a SWI 0x6d para obter o valor do "tick" do relógio interno do simulador e utilizar os 10 bits menos significativos como valor aleatório).
b) O usuário deve fornecer um palpite por meio do teclado, cujo arranjo é mostrado abaixo. Cada novo dígito fornecido do palpite é mostrado no display de LCD.

1	2	3	
4	5	6	
7	8	9	
	0		

c) Quando o botão da esquerda ("Left") for pressionado, o sistema deve incrementar o dígito no display de 8 segmentos, indicando o número de tentativas efetuadas, e deve comparar o palpite do usuário com o número aleatório sorteado. Caso sejam iguais, o display de LCD deve mostrar a mensagem "Parabéns! Você acertou!" e o sistema deve aguardar que o usuário pressione o botão da direita ("Right") para zerar o dígito no display de 8 segmentos e reiniciar o jogo. Além disso, o sistema deve mostrar no display de LCD o tempo decorrido desde o início do jogo (dica: executar a SWI 0x6d para obter o valor do "tick" do relógio interno do simulador, tanto no início do jogo quanto no final, e calcular a diferença de tempo).
d) Caso o palpite não seja igual ao valor sorteado e o número de tentativas mostrado no display de 8 segmentos seja menor do que 9, o display de LCD deve mostrar a mensagem "Palpite incorreto e muito grande", caso o valor do palpite seja maior do que o valor sorteado, ou a mensagem "Palpite incorreto e muito pequeno", caso o valor do palpite seja menor do que o valor sorteado. Em ambos os casos, a execução retorna para o item b) e um novo palpite é aguardado,
e) Caso o palpite não seja igual ao valor sorteado e o display de 8 segmentos esteja mostrando o valor 9, o display de LCD deve exibir a mensagem "Número de tentativas máximo atingido!" e o sistema deve aguardar que o usuário pressione o botão da direita ("Right") para zerar o dígito no display de 8 segmentos e reiniciar o jogo.

Enunciado 6

Implemente um programa em *assembly* ARM7TDMI que simule um editor gráfico, utilizando a interface do *plugin* EmbestBoard no ARMSim#.

a) O programa define um cursor na forma de um traço (-) , inicialmente posicionado no canto superior esquerdo do display de LCD. A área útil do display de LCD é utilizada como área de desenho, utilizando um conjunto determinado de caracteres.
b) O usuário pode utilizar o teclado para movimentar o cursor, conforme as teclas de seta do arranjo mostrado abaixo. As demais teclas servem para definir qual caractere será impresso no display na posição do cursor, à medida que ele é movimentado (o caractere "Espaço" imprime um espaço em branco, ou seja, apaga a posição onde se encontra o cursor).

	↑		Espaço
←		→	#
	↓		*
		O	-

c) O botão da esquerda ("Left") é utilizado para limpar completamente a área de desenho, preenchendo-a com espaços. O botão da direita ("Right") é utilizado para inverter o desenho, de tal maneira que os caracteres diferentes de espaço sejam preenchidos com um espaço e os caracteres de espaço sejam preenchidos com '#'.
d) O LED vermelho da esquerda deve ser aceso sempre que um caractere diferente de espaço estiver selecionado para ser preenchido na posição do cursor. O LED vermelho da direita, por sua vez, deve ser acesso sempre que o caractere de espaço estiver selecionado.
e) O display de 8 segmentos deve mostrar o percentual da área de desenho preenchida com caracteres diferentes de espaço, em múltiplos de 10%. Por exemplo, se 62% das posições da área de desenho estiverem preenchidas, o display de 8 segmentos deve mostrar o dígito 6. Se 100% estiverem preenchidos o display de 8 segmentos deve mostrar a letra 'A'.