Arquitectura de Computadores

2011/2012 2º Semestre

1º Teste - Repescagem - 27/06/2012

Número: _____ Nome: _____

INSTRUÇÕES:

- A duração da prova é de 2 horas.
- Preencha imediatamente o seu número e nome de forma bem legível, em todas as folhas.
- Responda apenas no presente enunciado; não serão aceites folhas adicionais. Seja sucinto.
- A cotação das perguntas está indicada entre parênteses à direita do texto.
- Nas perguntas Verdade/Falso ou de escolha múltipla as respostas erradas descontam.

I

1. Considere os seguintes valores para os registos do processador P3:

R1	R2	R3	R4	R5	R6	R7	PC	SP	RE
0011h	0002h	0370h	41F0h	805Ah	FF01h	0000h	0009h	FADEh	0018h

Nas alíneas a), b) e c), indique qual o novo valor, em hexadecimal, para todos (**e apenas**) os registos que são escritos na execução de cada instrução. Use **?** caso não tenha informação suficiente para determinar o novo valor de um registo.

As alíneas são independentes, isto é, assuma como valores iniciais para cada alínea os indicados na tabela acima.

a) DIV R1, R2 **(1)**

R1	R2	R3	R4	R5	R6	R7	PC	SP	RE
0008h	0001h						000Ah		0010h

b) POP M[R6+1] **(1)**

R1	R2	R3	R4	R5	R6	R7	PC	SP	RE
							000Bh	FADFh	

c) CALL 100h **(1)**

R1	R2	R3	R4	R5	R6	R7	PC	SP	RE
							0100h	FADDh	

d) Para o ciclo completo de execução da instrução SHRA M[SP+11h], 2 indique na tabela seguinte a sequência de acessos à memória, especificando o endereço, dados e tipo de acesso (**L**eitura/**E**scrita). **(2)**

Nota 1: A tabela tem 5 linhas, utilize apenas as que considerar necessárias.
Nota 2: Utilize os valores iniciais dos registos indicados na tabela no cimo desta página.
Nota 3: Use **?** para indicar que não tem informação suficiente para determinar um dado valor.

	Endereço	Dados	L / E
1	0009h	68BEh	L
2	000Ah	0011h	L
3	FAEFh	?	L
4	FAEFh	?	E
5			

(página propositadamente em branco; pode usar para rascunho)

II

2.1 Considere o seguinte troço de código escrito na linguagem de programação C, o qual se pretende implementar em Assembly do P3.

```
#define SIZE 32

short int resultado = 0;
short int vectorA[SIZE];
short int vectorB[SIZE];
short int vectorSaida[SIZE];

register short int indice;
register short int soma;

soma = 0;
indice = 0;
while(indice < SIZE){
    vectorSaida[indice] = vectorA[indice] + vectorB[indice];
    soma += vectorSaida[indice];
    indice++;
}
resultado = soma;
```

a) Comece por definir as constantes e declarar as variáveis do programa. A zona de dados deverá começar em 8000h. **(1,5)**

```
SIZE        EQU     32

            ORIG    8000h
resultado   WORD    0
vectorA     TAB     SIZE
vectorB     TAB     SIZE
vectorSaida TAB     SIZE
```

b) O programa deverá começar por inicializar os registos necessários para a implementação do algoritmo. Apresente o código correspondente e indique, na forma de comentário, a função de cada registo. **(0,5)**

```
            MOV     R1, R0  ; R1 = indice
            MOV     R2, R0  ; R2 = soma
```

c) Implemente o ciclo de acordo com as respostas das alíneas anteriores. Assuma que nenhuma operação aritmética gera *overflow* ou transporte. Procure optimizar o seu código em termos de eficiência e utilização de memória. **(3)**

```
Loop:       CMP     R1, SIZE
            BR.NN   Fim
            MOV     R3, M[R1+vectorA]
            ADD     R3, M[R1+vectorB]
            MOV     M[R1+vectorSaida], R3
            ADD     R2, R3
            INC     R1
            BR      Loop
Fim:        MOV     M[resultado], R2
```

(página propositadamente em branco; pode usar para rascunho)

III

3.1 Considere o seguinte programa em Assembly do P3. Considere que a rotina EscCont escreve na janela de texto o número que se encontra na posição de memória Contador.

```
    SP_INICIAL       EQU     FDFFh

    INT_MASK_ADDR    EQU     FFFAh
    INT_MASK         EQU     8001h

    IO_WRITE         EQU     FFFEh

    TIMER_UNITS      EQU     FFF6h
    TIMER_CTRL       EQU     FFF7h
    BASETIME         EQU     ______

                     ...


                     ORIG    8000h
    Contador         WORD    0000h
    Event            WORD    0000h


                     ORIG    0000h
                     JMP     Inicio

    Timer:           CALL    ResetTimer
                     INC     M[Event]
                     RTI

    TrataI0:         MOV     M[Contador], R0
                     RTI

    ResetTimer:      MOV     R1, BASETIME
                     MOV     M[TIMER_UNITS], R1
                     MOV     R1, 1h
                     MOV     M[TIMER_CTRL], R1
                     RET

                     ...


 1:  Inicio:         MOV     R7, SP_INICIAL
 2:                  MOV     SP, R7
 3:                  MOV     R7, INT_MASK
 4:                  MOV     M[INT_MASK_ADDR], R7
 5:                  CALL    ResetTimer
 6:                  ENI

 7:  CicloCont:      CALL    EscCont
 8:  Espera:         CMP     M[Event], R0
 9:                  BR.___  Espera
10:                  ____________________
11:                  INC     M[Contador]
12:                  BR      CicloCont
```

a) Indique como deve ser definida a tabela de vectores de interrupção de modo a que a rotina "Timer" fique associada ao temporizador do P3 e a rotina "TrataIO" fique associada à interrupção 0. **(1)**

```
        ORIG   FE00h
INT0    WORD   TrataIO
        ORIG   FE0Fh
INT15   WORD   Timer
```

b) Indique o valor da constante BASETIME de modo a que o contador seja incrementado de 5 em 5 segundos. **(0,5)**

50

c) Complete o salto condicional da linha 9. **(0,5)**

BR.Z

d) Indique que instrução deverá estar na linha 10. **(1)**

MOV M[Event], R0

e) Pretende-se que, ao carregar no botão I0 (na "Janela Placa" do simulador do P3), o programa mostre o valor zero e passado 5 segundos mostre o valor 1 e assim sucessivamente. É este o comportamento actual do programa? Se não é, reescreva no espaço abaixo uma nova rotina TrataIO que resolva o problema. **(2)**

```
TrataIO:    MOV   M[Contador], R0
            CALL  ResetTimer
            CALL  EscCont
            RTI
```

IV

4.1 Considere o seguinte troço de código em Assembly do P3:

```
                ORIG    0100h

Conv:           MOV     R3, 10
Next:           MOV     R2, M[R1+15]
                CMP     R2, R0
                BR.NN   Skip
                NEG     M[R1+15]
Skip:           INC     R1
                DEC     R3
                JMP.NZ  Next
Avg:            MOV     R1, R0
                . . .
```

Analise o troço de código acima e preencha a seguinte tabela com o valor, em hexadecimal, correspondente a cada etiqueta. **(2)**

Etiqueta	Valor
Conv	0100h
Next	0102h
Skip	0108h
Avg	010Ch

4.2 Considere um programa escrito em Assembly do P3 em que estão a ser usados números positivos e negativos (representados em complemento para dois) e em que se convencionou trabalhar em vírgula fixa com 4 casas decimais. Indique, na base decimal, qual o valor representado por **FFB4** h. **(1)**

-4,75 _____

4.3 Indique se as seguintes afirmações são verdadeiras ou falsas. **(2)**

Nota: Cada pergunta certa vale 0,5 valores; respostas erradas descontam 0,25 valores.
A classificação mínima desta pergunta é 0.

a) As instruções NOP e BR 0 são em tudo equivalentes no contexto da lógica de um programa.

X☐ Verdadeiro ☐ Falso

b) A instrução com o código máquina 1881h corresponde a um RET.

X☐ Verdadeiro ☐ Falso

c) A instrução DIV R1, 2 é uma instrução válida.

☐ Verdadeiro X☐ Falso

d) A instrução SHR R1, 2 tem como efeito prático dividir o conteúdo de R1 por 2, independentemente do valor de R1.

☐ Verdadeiro X☐ Falso

(página propositadamente em branco; pode usar para rascunho)