

Grupo I

(a)

R3 = ?
PC = 7E94h
RE = ?

(b)

R6 = E503h
RE = 0006h
PC = 7E94h

(c)

PC = ?
SP = AC3Ch
RE = ?

(d)

Endereço	Dados	L/E
7E93h	7932h	L
7E94h	0020h	L
EFF9h	?	L
EFF9h	?	E

/*****/

Grupo II

(a)

MOV R1, M[DATA_SIZE] ou MOV R1, 16
MOV R2, IN_DATA
MOV R3, OUT_DATA

(b)

BR.NN NEXT

(c)

INC R3
DEC R1

(d)

OUT_DATA[0]=2
OUT_DATA[1]=7
OUT_DATA[2]=2
OUT_DATA[3]=0
OUT_DATA[4]=5

(e)

SOMA = 89 ou 59h

(f)

SHRA R4, 4 ou SHR R4, 4 (uma vez que são todos positivos...)

Outras alternativas possíveis para as alíneas c) e f):

(c)

INC R3
CMP R2, OUT_DATA ou CMP R2, 8011h

(f)

DIV R4, R1

/*****

Grupo III

(a)

Solução 1:

; Antes das variaveis

```
INT0          ORG     FE00h    ;endereço da rotina de tratamento da int. I0
              WORD    TrataI0
              ORG     FE0Fh
INT_TIMER     WORD    Timer    ;endereço da rotina de tratamento da int. Timer
```

; Codigo

```
InitInts:     MOV     R1, 8001h    ;mascara das interrupções
              MOV     M[MASK_ADDR], R1 ; guarda a mascara no respectivo
              endereço
              RET
```

Solução 2:

; Constantes

```
INT_VECTOR    EQU     FE00h
```

; Codigo

```
InitInts:     MOV     R1, 8001h
              MOV     M[MASK_ADDR], R1
              MOV     R1, INT_VECTOR
              MOV     M[R1], TrataI0
              ADD     R1, 000Fh
              MOV     M[R1], Timer
              RET
```

(b)

A inicialização do timer "CALL Timer" é feita de modo incorrecto dado que se trata de uma rotina para tratar uma interrupção (finalização com RTI). A sua utilização indevida pode danificar o registo de estado e/ou comprometer o stack (ex: suponha que a inicialização do timer é feita dentro da rotina X; com o RTI, o endereço de retorno da rotina X é perdido).

Assim, propoe-se a seguinte correcção:

; Codigo

```
TimerInit:    MOV     R1, BASETIME
              MOV     M[TIMER_UNITS], R1 ; Inicializa o contador
              MOV     R1, 0001h
              MOV     M[TIMER_CTRL], R1 ; Activa o contador
              RET
```

;na função main, em vez de CALL Timer colocar a instrução

```
CALL TimerInit
```

(c)

A rotina TrataI0, tem como funcionalidade fazer a paragem do timer. Para isso coloca o resultado da operação Xor R1, R1 (R1 = 0) no controlador do Timer.

(d)

Solução 1:

; Codigo

```
PiscaLed:     MOV     R1, 0001h
              XOR     M[Var1], R1
              MOV     R1, M[Var1]
              MOV     M[LEDS_CTRL], R1
              RET
```

Solução 2:

; Codigo

```
PiscaLed:     COM     M[Var1]
              MOV     R1, 0001h
              AND     R1, M[Var1]
              MOV     M[LEDS_CTRL], R1
```

RET

/*****

Grupo IV

(a)

4720h		011101 1100 11 0101	=	0111 0111 0011 0101	=	7735h
4721h		10 0000			=	0020h
4722h		110001 0100 10 0000	=	1100 0101 0010 0000	=	C520h
4723h					=	4720h

(b)

Vantagem: Torna-se possível manter mais valores temporários guardados em registos, podendo-se assim reduzir a utilização da pilha e de variáveis em memória, tornando desta forma a execução mais rápida.

Desvantagem: É necessário mais um bit para codificar o registo na codificação das instruções, podendo fazer com que os programas se tornem maiores e levem, por isso, mais tempo a executar.

(c.i) - D

(c.ii) - A

(d.i) - Falso

(d.ii) - Falso

(d.ii) - Falso