Q4. 1,8/2 Q5. 1/1 7,0

Q6. 1/1

Q1. 1,75/2

Q2. 0,75/2

Q3. 1/2

EA 869 – Turma A – 1. Semestre 2007

Prova 2 – 28/06/2002 – Prof. Léo Pini Magalhães.

(com consulta a 1 folha A4 que não pode ser fotocópia – assine a sua folha)

Nome: Nilton Hugo Ramos Firmino RA: 065017

Q1. (2,0)

Considere o seguinte programa em linguagem assembly: RA=pqrstu

```
          ORG     stuq        ; stuq são 4 dígitos de seu RA como acima definido
inicio:   MOVE    cont, D1    ; D1 ← (cont): move (cont) para o registrador D1
          ADDI    #2, D1      ; D1 ← (D1) + 2: adiciona o valor 2 à (D1)
          JNE     fim         ; se D1≠ 0, PC ← endereço "fim"
          MOVE    D1, saida   ; saida ← (D1) , move (D1) para a variável saida
fim:      STOP
cont:     DW      4           ; define a var. "cont" com valor 4
saida:    DS      1           ; define espaço para a variável saída
          END
```

5017 0175 Leu-se? Ler a questão antes de resolvê-la.

TIM

Mnemônico	C.O. (hexadec.)	Compr.(bytes)
ADDI , D1	B1	3
MOVE , D1	D4	4
MOVE D1,	C4	4
JNE end	F3	4
STOP	FF	1

TPI

Mnemônico	Compr.(bytes)
ORG	- -
DW	4
DS	4 * i
END	- -

1,25

(a) (1,50) Usando a base hexadecimal, escreva o resultado dos passos (1) e (2) do Montador, respectivamente a Tabela de Símbolos e o Endereço/Código de cada instrução do Programa em Linguagem de Máquina. Comente seu raciocínio.

(b) (0,50) Quais modificações você teria de fazer no programa fonte para o mesmo poder ser processado em um montador de 1 passo ?

a)

5017	Início: move cont, D1	D4
5018		00
5019		50
501A		27
501B	addi #2, D1	B1
501C		00
501D		02
501E	JNE fim	F3
501F		00
5020		50
5021		26
5022	move D1, saida	C4
5023		00
5024		50
5025		2B
5026	fim: Stop	FF
5027	cont: DW 4	00
5028		00
5029		00
502A		04
502B	Saida: DS 1	00
502C		00
502D		00
502E		00

Preenche com zeros as posições de memória reservadas para saída

TS

Início	5017
fim	5026
cont	5027
saida	502B

move +4, addi +3, JNE +4, move +4; Stop +1; DW +4

0,4

Primeiro montamos a Tabela com endereços de memória, tendo início no endereço 5017H, depois, vamos pulando posições de memória conforme o tamanho das instruções, dados e pseudo instruções. Observando o endereço dos rótulos e montando a TS.

Por fim, voltamos ao início do programa e vamos substituindo os mnemônicos e rótulos por seu código em hexadecimal

b) Para que o programa pudesse ser processado em um passo, deveríamos colocar a declaração Todos os rótulos no início do programa. Dessa forma, não seria preciso fazer o passo 1, que é basicamente montar a Tabela de símbolos para que no passo 2 o montador pudesse ir somente substituindo os rótulos, pois estes já estariam definidos.

Q12

JMP fim ⇒ BRA FIM
end relativo

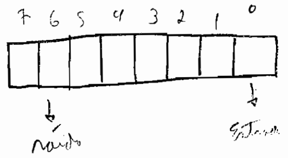
0,75

Q2. (2,0)
Esta questão trata uma interface serial em um processador de 8 bits que realiza E/S mapeada. A interface possui os registradores de dados (RD com 1 byte), de estado (RE com 1 byte) e de controle (RC com 1 byte). No RE os bits 0 e 6 sinalizam respectivamente entrada e saída prontas a operar. No RC devem ser definidos os seguintes valores para operação da interface: E/S, Loop de espera/condicional ou interrupção, paridade par/ímpar, stop e start bits, baud rate, simplex, half duplex ou full duplex. Caso necessário utilize registradores da CPU D1, D2, etc.

(a) (0,9) inicialize (em Assembly) a entrada de dados por interrupção escolhendo ainda os outros valores para a operação da interface. Após descreva todo o processo de entrada que você definiu lembrando que CPU, interface e dispositivo atuam para o mesmo se concretizar.

(b) (0,9) inicialize (em Assembly) agora a saída de dados por loop de espera escolhendo ainda os outros valores para a operação da interface. Após descreva todo o processo de saída que você definiu lembrando que CPU, interface e dispositivo atuam para o mesmo se concretizar.

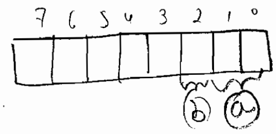
0,1 **(c) (0,2)** qual seria a diferença entre realizar a operação por half duplex ou por full duplex ?



{ 0 → não há dado
{ 1 → há dado

def RC × 0,4
inic 0,4 0,4
descr × 0,5
c/f × 0,25
loop 0,25 0,25

Definindo o RC:



(a) { M1 = 01 → loop espera
M2 = 10 → loop condicional
M3 = 11 → interrupção

(b) { 1 - entrada
0 - saída

c) A diferença entre os modos é que, em full duplex, podemos enviar e transmitir dados simultaneamente. Já em half duplex, podemos enviar e transmitir, mas não ao mesmo tempo.

b)
```
      Move #01110001₂, RC;
loop: Move RE, D1;
      Testbit 6;
      JZ loop;
      Move RD, D2;
```

a) Move #00110111₂, RC

Q3. (2,0) Um processador atua com máscara de interrupção e vetor com duas posições (psw e end) por linha.

(a) (0,5) ao ser aceita uma interrupção suponha que a aplicação necessite salvar o estado corrente do programa. Quais ações são tomadas pelo hardware/firmware e quais deverão ser realizadas pelo programador na rotina de serviço (RS) ?

(b) (1,5) o processador tem 4 linhas de interrupção vetorizadas a partir da posição hexadecimal 00xx (xx são seus 2 últimos dígitos de matrícula) de memória.

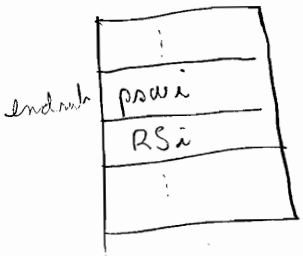
① • defina adequadamente todo o vetor de forma a que, ao início, nenhuma RS possa ser interrompida;

② • defina onde o programa principal inicia, via ORG, e inicialize o " estado corrente da PSW" do mesmo para um correto processamento das 4 interrupções;

③ • defina as RS esquematicamente.

a) No momento que a UCP aceita a interrupção, os seguintes comandos deverão ser executados:

```
Push PC;
Push PSW;
Move endrot, PSW;
Jump (endrot+1);
```



O hardware/firmware deverá executar esses comandos ao aceitar a interrupção.

Cabe ao programador definir a máscara de interrupção, o endereço (RSi) onde está a rotina e o endereço endrot para qual será feito o Jump. + salvar regs

Após a execução da rotina, os registradores e o pilha deverão estar como estavam no instante que deu o início da rotina, e então executa os comandos:

```
Pop PSW;
Pop PC;
```

b)

0017	0000
0018	RS1
0019	0000
001A	RS2
001B	0000
001C	RS3
001D	0000
001E	RS4

①

```
ORG   0017
DW    0
ADROW 1000
DW    0
ADROW 2000
DW    0
ADROW 3000
DW    0
ADROW 4000
```

②

```
ORG 1000
RS1: ... } corpo
RTI

ORG 2000
RS2: ... }
RTI

ORG 3000
RS3: ... }
RTI

ORG 4000
RS4 ... }
RTI
```

③

Considerando a definição do Vetor como ①;

O programa principal, no caso, o exemplo executado em Assembly, em ②

A definição de cada rotina em ③.

principal ? X

1,8

Q4 (2,0)
Qual é o mecanismo de passagem de parâmetros, **na subrotina**, adotado em cada item abaixo? Identifique se é por valor ou por endereço e se é de entrada e/ou saída para cada parâmetro **da sub-rotina**, justificando a sua resposta. Nas instruções, o 2. operando é sempre o operando-destino.

a)

0,5

Programa

```
          MOVE #10, SP
          -------------------
          MOVE DADO1, DR1
          MOVE DADO2, DR2
          CALL ROT
          MOVE DR2, DADO2
          -------------------
          STOP
DADO1:         DW 1
DADO2:         DW 2
```

Sub-rotina

```
DR1:  DS 1
DR2:  DS 1
ROT:  -------------------
      -------------------
      AND DR1, DR2
      -------------------
      -------------------
      RTS
```

· Área de Dados associado à Sub-Rotina; pois no início da sub-rotina há um espaço de memória reservado para as variáveis.

Dado 1	valor	Entrada
Dado 2	valor	Entrada/Saída
DR 1	valor	Entrada
DR 2	valor	Entrada/Saída

·Pois trabalha com as variáveis, não com seu endereço

· DR1, Dado1 são somente entrada, estão no primeiro operando
DR2, Dado2 estão no 2° operando das operações, logo, também são escritos dados neles e assim, usados como saída.

b)

0,4

Programa

```
          MOVE #10, SP
          -------------------
          PUSH #DADO2
          PUSH #DADO1
          CALL ROT
          -------------------
          STOP
DADO1:         DW 1
DADO2:         DW 2
```

PC
Dado 1
Dado 2

Sub-rotina

```
ROT:  POP R1
      POP R2    dado 1
      POP R3    dado 2
      AND (R2), (R3)
      PUSH R1
      RTS
```

Por pilha; os endereços de Dado 1 e dado 2 são postos na pilha pelo programa e depois retirados pela sub-rotina.

Dado 1 (R2)	referência	entrada
Dado 2 (R3)	referência	entrada/saída

Pois trabalha com o endereço das variáveis

Dado 1 no 1° operando da operação, entrada
Dado 2 no 2° operando, logo, entrada e saída.

Q5. (1,0) Diferencie parâmetros reais e parâmetros formais. Exemplifique em uma situação fictícia através dos comandos de definição e chamada de uma subrotina:

SUBROUTINE SUB e CALL SUB.

Os parâmetros formais são os que estão definidos na criação (definição) da sub rotina, por exemplo:

Sub Routine sub A, B, C, D

corpo da Rotina:
```
ADD A,B;
ADD B,C;
MOVE C,D;
RTS
```

Os parâmetros A, B, C, D são formais, são apenas usados para realizar as operações dentro da subrotina.

1

Já os parâmetros reais são os valores (ou endereços) que passamos no lugar da chamada da subrotina:

call sub X, Y, W, Z

Eles são os dados (endereços) que serão passados para a subrotina e com os quais ela efetivamente trabalhará. No caso, quando, na definição da rotina, tínhamos A, ao ser chamada, trabalharemos com X, no lugar de B, Y, assim por diante.

Q6. (1,0)

Considere a funcionalidade a seguir que você deverá usar em diversas situações de seu programa:

- em algumas ocasiões você só necessitará somar os registradores D4 = D1 + D2 ; em outras
- D4 = D1 + D2 + D3, e em outras
- somente carregar D1 em D4.

1

Defina uma macro e forneça as chamadas (1 para cada caso) para exemplificar a realização das tarefas acima.

```
macro  nome D1, D2, D3, D4
       move D1, D4;
       Aif ('D2' = " ") .cond1 ;
       ADD D2, D4;
.cond1 Aif ('D3' = " ") .fim ;
       ADD D3, D4;
.fim   ENDMACRO
```

```
1º) nome A, B, , D
    { move A, D;
      Add B, D;

2º) nome A, B, C, D
    { move A, D;
      Add B, D;
      Add C, D;

3º) nome A, , , D
    { move A, D;
```