8,1

Q1. 1,4/1,5
Q2. 1/1
Q3. 1,5/1,5
Q4. 1/1
Q5. 1,1/1,5
Q6. 0,85/1,5
Q7. 0,5/1
Q8. 0,75/1

EA 869 – Turma A – 1. Semestre 2007
Prova-1 – 03/maio/2007 – Prof. Léo Pini Magalhães
(com consulta a 1 folha A4 que não pode ser fotocópia – assine a sua folha)

Nome: Vitor Hugo Ramos Ficcini [illegible] Número: 065017

Q1. (1,5) Você está prestes a obter aquele emprego sonhado e para isto deve responder às duas questões abaixo envolvendo as três funções de complexidade de três algoritmos ligados a um processo industrial:

- a: $13 + 5 . n$
- b: $29 + 8 . \log_2 n$
- c: $1 + 2 . n^2$ (sendo n o tamanho da entrada)

(a) **(1,0)** qual dos algoritmos é o mais apropriado para cada região de operação (as regiões de operação são definidas pelo tamanho da entrada) ? Justifique.
(b) **(0,5)** qual das opções é a melhor para o comportamento assintótico do tamanho da entrada ? Justifique.

0,3+0,3+0,3+0

a) Para n pequeno, o melhor é a função ©, pois com ⓝ pequeno o termo quadrático não altera muito o resultado e, tendo um valor inicial menor que a anterior, faz com que seja o melhor, por exemplo, para n=2: a = 23, b = 37, c = 9, mas já para n=4, ela se iguala à função ⓐ: a = 33 b = 45 c = 33 e para cima dessa já não compensa usar ©. Para n médio, acho que o melhor é usar de ⓐ, pois [illegible] crescimento [illegible] e possui um valor inicial menor que b, contudo, quando n=8, assim como antes, o valor de b e a são iguais: a = 53, b = 53, então compensa usar a função b desde então, pois a função que menos cresce é a logarítmica. Ou seja: n≤4 → ©, 4 < n ≤ 8 → ⓐ, n ≥ 8 → ⓑ, lembrando que [illegible] do domínio da função $\log_2 n$ para os valores de n≤0 ? tamanho da entrada não pode ser negativo ! ⊖

cont. ⓑ atrás

Q2.(1,0) (0,5) Por quê ao iniciar a resolução de um problema você primeiramente trata a questão da computabilidade e posteriormente a da complexidade ? **(0,5)** Se você identificou o problema como NP o que você acha sensato fazer ?

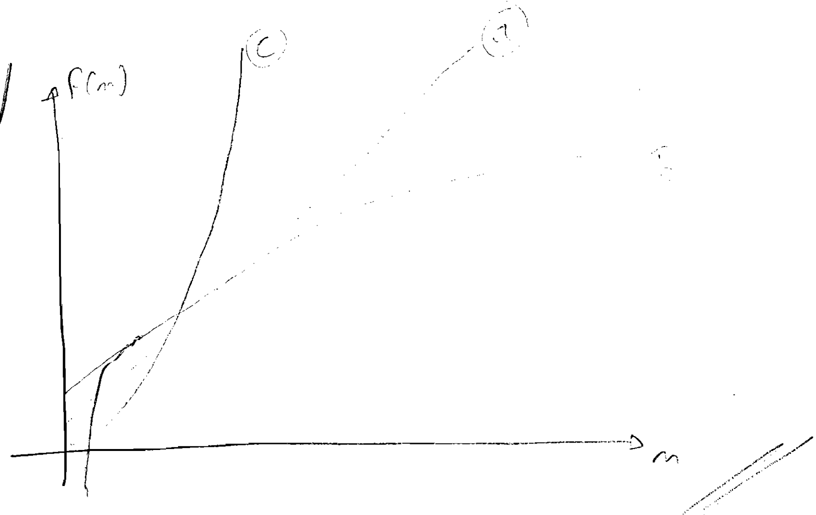
0,5 ✓

a) Antes de se iniciar a resolução do problema é preciso avaliar a computabilidade pois caso o problema não seja computável, significa que ele não pode ser resolvido em um número finito de passos, ou seja, será impossível achar a solução correta. Contudo, é preciso avaliar a complexidade pois é preciso verificar se o problema, sendo computável pode ser resolvido em tempo hábil, pois não podemos esperar a resposta do problema caso ele demande muito tempo, que é o caso dos problemas de complexidade exponencial.

cont. atrás

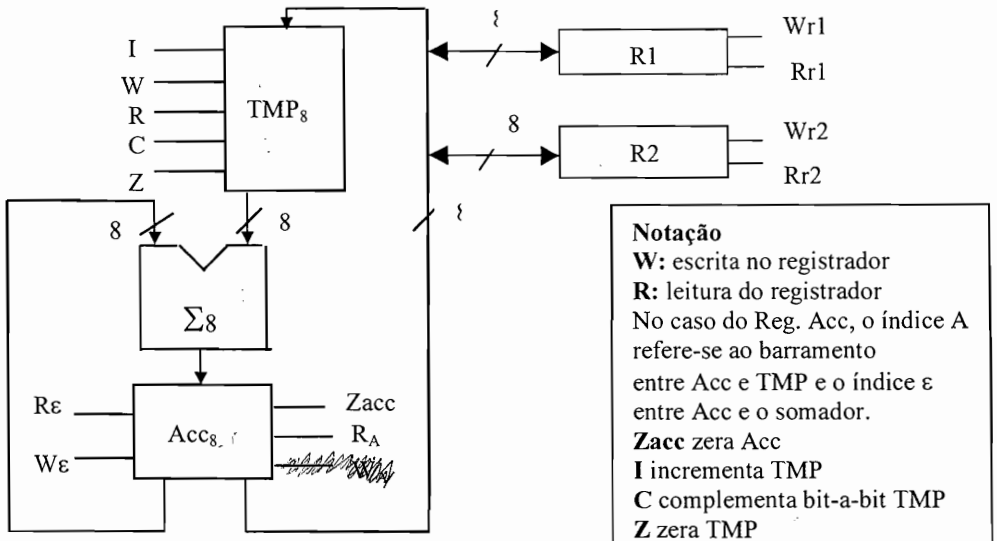
1) b) Para este caso, a melhor função será **b**, devido ao seu crescimento muito mais baixo comparado às funções anteriores em função de n, sendo algo do tipo:

0,5



2) Caso seja identificado um problema NP, sabemos que a melhor solução encontrada até agora tem complexidade exponencial. Logo, o que pode ser feito é: Analisá-lo para n pequenos; Ou verificar a solução dada (o que é feito em tempo polinomial); Ou tentar transformá-lo em um problema P.

Q3. (1,5) Para o esquema a seguir, 1,5



Notação
W: escrita no registrador
R: leitura do registrador
No caso do Reg. Acc, o índice A refere-se ao barramento entre Acc e TMP e o índice ε entre Acc e o somador.
Zacc zera Acc
I incrementa TMP
C complementa bit-a-bit TMP
Z zera TMP
Σ_8 opera em complemento de 2

preencha a tabela abaixo (complete com tantas linhas quanto o necessário) para a operação:

R1 ← (TMP) + (R2).

A solução esperada é aquela que apresentar o menor número de linhas (pulsos de relógio).

Pulso do relógio	Microoperação	Microcomandos	Observação (se for o caso)
1	Σ ← (TMP)	R	Smo pois o somador tem atraso
2	Acc ← (TMP)+0	Wε	
3	TMP ← (R2)	RR2, W	
4	Σ ← (R1)+(TMP)	Rε, R	
5	Acc ← (Acc)+(TMP)	Wε	
6	R1 ← (Acc)	RA, WR1	

ou considerando a hipótese de *

Q4. (1,0) Na arquitetura acima caso não existisse o sinal Z_{acc} , como você procederia ? Forneça os passos a realizar, não é necessário fornecer microoperações ou sinais de controle. Use o verso da folha se necessário.

1

Um modo pelo qual poderia-se zerar Acc é mandar ao seu conteúdo o complemento do mesmo, assim zerando-o. Smo pode ser implementado transferindo o conteúdo de Acc para TMP, complementando, processando e depois mandando também o conteúdo de Acc quando o de TMP para o somador, o que vai zerá-lo, e depois enviar para o Acc novamente o 0.

2

Q5. (1,5) Na folha anexa você encontra o computador microprogramado estudado em classe. Defina **somente a parte de execução** de um microprograma que realize:

. se Acc ≥ 0 , então Acc ← (Acc) + (R1)

. se Acc < 0 , então Acc ← (Acc) + (R2).

Defina um valor para **N** e considere que o microprograma que realiza a busca inicia no endereço 100 (hexadecimal) da micromemória. **(Não esqueça isto ao terminar o seu microprograma !)**

MICROMEMÓRIA

Endereço (hexa)	Microoperação	Microcomandos	Observação (se for o caso)
N = 10	MPC ← (mPC) + ...	SC: 21,23	MPC recebe o novo dado ...
11	JMP 13	SC: 24,6,9,10	Se Acc ...
12	JMP 15	SC: ...	Se Acc ...
13	Acc ← (Acc) - (R2) MPC ← (MPC) + 1	SC: 2,4,11 SC: 23,13	Reduz o novo e incrementa MPC
14	JMP 100	SC: 24,2	Volta p/ o buscar
15	Acc ← (Acc) + (R1) MPC ← (MPC) + 1	SC: 2,3,11 SC: 23,13	... MPC
16	JMP 100	SC: 24,2	Volta p/ o buscar

Q6. (1,5) Considere a representação em ponto flutuante normalizada com 1 bit para sinal, 6 bits para mantissa e 4 bits para expoente (em complemento de 2 e base 2).

(a) (0,75) Dê a representação dos números +3,48 e +4,53 e apresente o resultado da soma dos mesmos em ponto flutuante. Qual o valor em decimal ? Mostre seu raciocínio e comente a questão da precisão.

Em binário com 6 dígitos, sendo o primeiro dígito o sinal:

$3,48_{10} = 11,0111_2 = ,110111 . 2^{0010}$

$4,53_{10} = 100,1001_2 = ,100101 . 2^{0011}$

$$3,48 : 0,011011 \cdot 2^{0011}$$

$$+\ 4,53 : 0,100101 \cdot 2^{0011}$$

$$(0)\,1\,00000 \cdot 2^{0100} = \times$$

Em decimal isso seria 8,01, contudo essa precisão não é atingida devido a perda de bits ao esquerda no expoente.

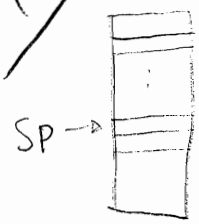
(b) (0,75) Dê a precisão da mantissa e a precisão da representação para expoente igual a 2^{-3}.

Como a mantissa possui 6 dígitos, podemos dizer que sua precisão é de [illegible]. Contudo, não podemos afirmar a precisão da representação para expoente 2^{-3} sem saber a quantidade de bits na mantissa, uma vez que o expoente não determina a precisão.

0,35

Q7. (1,0) PUSH (R5) Como poderia ser implementado em uma máquina que não possua endereçamento por pilha? Defina os registradores necessários e use instruções de 2 endereços típicas.

0,5



Seja R5 o registrador no qual queremos armazenar o dado e SP o registrador que aponta para o primeiro endereço vazio no topo da memória, um modo de implementar a operação seria:

(SP) ← ((R5)) : MOVE (R5), (SP) ✗

SP ← (SP)+1 : ADD #1, SP

Q8. (1,0) Para o trecho de programa dado a seguir e os conteúdos das posições de memória fornecidos, dê os valores de R1, R2 e das posições de memória. Lembrar que os operandos fonte estão à esquerda da vírgula.

0,75

	R1	R2	Preencha !
MOVE #200, R2	? ✓	200 ✓	R2 ← 200
MOVE (R2)+, R1	20 ✓	201 ✓	R1 ← ((R2)), R2 ← (R2)+1
ADD (R2)+, R1	50 ✓	202 ✓	R1 ← (R1)+((R2)), R2 ← (R2)+1
MULT (R2)+, R1	300 ✓	203 ✓	R1 ← (R1)*((R2)), R2 ← (R2)+1
ADD (R2)+, R1	505 ✓	204 ✓	R1 ← (R1)+((R2)), R2 ← (R2)+1
ADD (R2)+, R1	525 ✓	205 ✓	R1 ← (R1)+((R2)), R2 ← (R2)+1
DIV R1, (R2)	13,125 ✗	205 ✓	R1 ← (R1)/((R2))

Endereço	25	30	40	...	200	201	202	203	204	205	206	207	208	209
Conteúdo-início	26	25	25		20	30	10	5	20	40	25	42	40	30
Conteúdo-final				.										

→ Igual para todos, não muda

Computador microprogramado