AD2 - Organização de Computadores 2008.2 - GABARITO

"Atenção: Como a avaliação a distância é individual, caso seja constatado que provas de alunos distintos sejam cópias umas das outras, independentemente de qualquer motivo, a todas será atribuída a nota ZERO. As soluções para as questões podem ser buscadas por grupos de alunos, mas a redação final de cada prova tem que ser individual."

1. (1,0) Crie 3 conjuntos de instruções de um, dois, e três operandos, definidas em Linguagem Assembly, necessárias para a realização de operações aritméticas e elabore programas para o cálculo das equações abaixo (no total de 3 programas para cada item abaixo, sendo o 1º programa utilizando o conjunto de 1 operando, o 2º utilizando o conjunto de 2 operandos e finalmente o 3º utilizando o conjunto de 3 operandos). No conjunto de 2 operandos considere o uso na equação com salvamento.

```
I_ CONJUNTO DE INSTRUÇÕES PARA 1 OPERANDO:
   LOAD M      =>    ACC   <-  M
   STORE M     =>    M     <-  ACC
   ADD M       =>    ACC   <-  ACC + M
   SUB M       =>    ACC   <-  ACC - M
   MUL M       =>    ACC   <-  ACC * M
   DIV M       =>    ACC   <-  ACC / M

II_ CONJUNTO DE INSTRUÇÕES PARA 2 OPERANDOS:
   ADD X,Y   =>   X • X + Y
   SUB X,Y   =>   X • X - Y
   MUL X,Y   =>   X • X * Y
   DIV X,Y   =>   X • X / Y
   MOV X,Y   =>   X • Y

III_ CONJUNTO DE INSTRUÇÕES PARA 3 OPERANDOS:
   ADD Y , Z, X      =>    X = Y + Z
   SUB Y , Z, X      =>    X = Y - Z
   MUL Y , Z, X      =>    X = Y * Z
   DIV Y , Z, X      =>    X = Y / Z
```

a) X = (A/D - C)+(B(D - E/B)* (C + A) + E)*

i _ Resolvendo através do conjunto de instruções com 1 operando:

```
LOAD A          =>    ACC <- A
DIV  D          =>    ACC <- ACC / D
SUB C           =>    ACC <- ACC - C
STORE T1        =>    T1 <- ACC
LOAD E          =>    ACC <- E
DIV B           =>    ACC <- ACC / B
STORE T2        =>    T2 <- ACC
LOAD D          =>    ACC <- D
SUB T2          =>    ACC <- ACC - T2
STORE T2        =>    T2 <- ACC
LOAD C          =>    ACC <- C
```

ADD A => *ACC <- ACC + A*
MUL T2 => *ACC <- ACC * T2*
MUL B => *ACC <- ACC * B*
ADD E => *ACC <- ACC + E*
ADD T1 => *ACC <- ACC + T1*
STORE X => *X <- ACC*

ii _ Resolvendo através do conjunto de instruções com 2 operandos:
MOV X, A => *X <- A*
DIV X, D => *X <- X / D*
SUB X, C => *X <- X – C*
MOV T1, E => *T1 <- E*
DIV T1, B => *T1 <- T1 / B*
MOV T2, D => *T2 <- D*
SUB T2, T1 => *T2 <- T2 – T1*
MOV T3, C => *T3 <- C*
ADD T3, A => *T3 <- T3 + A*
MUL T3, T2 => *T3 <- T3 * T2*
MUL T3, B => *T3 <- T3 * B*
ADD T3, E => *T3 <- T3 + E*
ADD X, T3 => *X <- X + T3*

iii _ Resolvendo através do conjunto de instruções com 3 operandos:
```
DIV  A, D, X   =>   X  <- A / D
SUB  X, C, X   =>   X  <- X - C
DIV  E, B, T1  =>   T1 <- E / B
SUB  D, T1, T1 =>   T1 <- D – T1
ADD  C, A, T2  =>   T2 <- C + A
MUL  B, T1, T1 =>   T1 <- B * T1
MUL T1, T2, T1 =>   T1 <- T1 * T2
ADD T1, E, T1  =>   T1 <- T1 + E
ADD X, T1, X   =>   X  <- X + T1
```

OBS: Sejam A,B,C,D,E,X, T1 e T2 endereços.

b) Y= (A - B(C + D/(E*(B-F))*C) + E)*

i _ Resolvendo através do conjunto de instruções com 1 operando:
LOAD B => *ACC <- B*
SUB F => *ACC <- ACC - F*
MUL E => *ACC <- ACC * E*
STORE T1 => *T1 <- ACC*
LOAD D => *ACC <- D*
DIV T1 => *ACC <- ACC / T1*
MUL C => *ACC <- ACC * C*
ADD C => *ACC <- ACC + C*
MUL B => *ACC <- ACC * B*
STORE T1 => *T1 <- ACC*
LOAD A => *ACC <- A*
SUB T1 => *ACC <- ACC - T1*
ADD E => *ACC <- ACC + E*
STORE X => *X <- ACC*

ii _ Resolvendo através do conjunto de instruções com 2 operandos:

MOV T1, B	=>	*T1 <- B*
SUB T1, F	=>	*T1 <- T1 – F*
MUL T1, E	=>	*T1 <- T1 * E*
MOV T2, D	=>	*T2 <- D*
DIV T2,T1	=>	*T2 <- T2 / T1*
MUL T2, C	=>	*T2 <- T2 * C*
ADD T2, C	=>	*T2 <- T2 + C*
MUL T2, B	=>	*T2 <- T2 * B*
MOV X, A	=>	*X <- A*
SUB X, T2	=>	*X <- X – T2*
ADD X, E	=>	*X <- X + E*

iii _ Resolvendo através do conjunto de instruções com 3 operandos:

SUB B, F, T1	=>	*T1 <- B – F*
MUL E, T1, T1	=>	*T1 <- E * T1*
DIV D, T1, T1	=>	*T1 <- D / T1*
MUL T1, C, T1	=>	*T1 <- T1 * C*
ADD C, T1, T1	=>	*T1 <- C + T1*
MUL B, T1, T1	=>	*T1 <- B * T1*
SUB A, T1, T1	=>	*T1<,- A – T1*
ADD T1, E, X	=>	*X <- T1 + E*

2. **(**1,0) Considere o programa assembly abaixo (baseado no assembly visto no capítulo 9 do livro texto) e o respectivo código de máquina :

	LDA Z	124
	ADD Y	325
ORG	SUB T	426
	STR X	227
	JZ FIM	523
	PRT X	B27
	JMP ORG	81E
FIM	HLT	000

Considere que as variáveis Y, Z, T e X estão armazenadas na memória nos seguintes endereços e com os conteúdos apresentados abaixo:

Variável	Endereço (hexadecimal)	Valor (hexadecimal)
Z	24	00B
Y	25	001
T	26	004
X	27	01A

Considere que o programa esteja armazenado na MP a partir do endereço 1C (hexadecimal), e o contador de instruções tem o valor 1C. Para a execução de cada instrução, mostre os valores do CI, RI , acumulador e das variáveis Y, Z, T e X

CI	*Conteúdo*		*RI *2*	*ACC*	*Z*	*Y*	*T*	*X*
1C			000	000	00B	001	004	01A
1D	124	LDA Z (ACC <- Z)	00B	00B	00B	001	004	01A
1E	325	ADD Y (ACC <- ACC + Y)	001	00C	00B	001	004	01A
1f	426	SUB T (ACC <- ACC – T)	004	008	00B	001	004	01A
20	227	STR X (X <- ACC)	004	008	00B	001	004	008
21	523	JZ 23 (Se ACC = 0, PC <- 23)	523	008	00B	001	004	008
22	B27	PRT X (exibe valor de X)	008	008	00B	001	004	008
1E	81E	JMP 1E (PC = 1E)	81E	008	00B	001	004	008
1F	426	SUB T (ACC <- ACC – T)	004	004	00B	001	004	008
20	227	STR X (X <- ACC)	004	004	00B	001	004	004
21	523	JZ 23 (Se ACC = 0, PC <- 23)	523	004	00B	001	004	004
22	B27	PRT X (exibe valor de X)	004	004	00B	001	004	004
1E	81E	JMP 1E (PC = 1E)	81E	004	00B	001	004	004
1F	426	SUB T (ACC <- ACC – T)	004	000	00B	001	004	004
20	227	STR X (X <- ACC)	000	000	00B	001	004	000
23	523	JZ 23 (Se ACC = 0, PC <- 23)	523	000	00B	001	004	000
*	000	HLT (END)	000	000	00B	001	004	000

*1 Endereço correspondente a da chamada da rotina, geralmente na pilha do sistema operacional

*2 O RI fica com o último valor recebido do barramento, por exemplo quando ele executa a instrução tipo LDA Z, ele recebe o conteúdo 124, depois na etapa de busca de operando, o RI recebe 00B (conteúdo do endereço Z)

3. (1,0) Faça uma busca na lista dos 500 sistemas de computadores com melhor desempenho do mundo em http://www.top500.org e descreva os três primeiros colocados (pesquise neste mesmo site e na internet). Descreva também o primeiro da lista que estiver no Brasil.

1ª POSIÇÃO
BladeCenter QS22/LS21 Cluster, PowerXCell 8i 3.2 Ghz / Opteron DC 1.8 GHz , Voltaire Infiniband
Fabricante: IBM
País: EUA
Ano: 2008
Processadores: 122400
RMax: 1026,000 Tflops
RPeak: 1375,780 Tflops
Processador: PowerXCell-8i
Freqüência do Processador: 3200MHz
Família: IBM Cluster
Sistema Operacional: Linux

2ª POSIÇÃO
eServer Blue Gene Solution
Fabricante: IBM
País: EUA
Ano: 2007

Processadores: 212.992
RMax: 478,000 Tflops
RPeak: 596,378 Tflops
Processador: PowerPC 440
Freqüência Processador: 700MHz
Família: BlueGene/L
Sistema Operacional: CNK/SLES 9

3ª POSIÇÃO
Blue Gene/P Solution
Fabricante: IBM
País: EUA
Ano: 2007
Processadores: 163.840
RMax: 450,300 Tflops
RPeak: 557.056 Tflops
Processador PowerPC 450
Freqüência do Processador: 850MHz
Família: BlueGene/P
Sistema Operacional: CNK/SLES 9

138° POSIÇÃO:
Maior Computador Brasileiro
Local: Universidade Federal do Rio de Janeiro – NCE
PowerEdge 1950, 2.66 GHz, Infiniband
Fabricante: Dell
Ano: 2008
Processadores: 2048
RMax: 16240
RPeak: 21790
Processador Intel EM64T Xeon E54xx (Harpertown)
Freqüência do Processador: 2666
Família: Dell PowerEdge Cluster
Sistema Operacional: CentOS

4. (1,0) Responda:

a) Dados os valores de memória abaixo e uma máquina de 1 endereço com um acumulador:
palavra 20 contém 40
palavra 30 contém 50
palavra 40 contém 60
palavra 50 contém 70

Quais valores as seguintes instruções carregam no acumulador?
-Load imediato 30
-Load direto 30
-Load indireto 30

- LOAD IMEDIATO 30
Nesta instrução o valor a ser colocado no acumulador corresponde ao valor fornecido como operador, portanto: ACC <- 30 (o valor a ser colocado no acumulador é 30)

- LOAD DIRETO 30

Nesta instrução o valor a ser colocado no acumulador corresponde ao valor contido no endereço de memória fornecido como operador, portanto:

ACC <- (30) (o valor a ser colocado no acumulador é 50)

- LOAD INDIRETO 30

Nesta instrução o valor a ser colocado no acumulador corresponde ao valor contido no endereço que consta como valor no endereço de memória fornecido como operador, portanto

ACC <- ((30)) (o valor a ser colocado no acumulador é 70)

b) Analise os modos de endereçamento direto, indireto e imediato, estabelecendo diferenças de desempenho, vantagens e desvantagens de cada um.

MODO DE ENDEREÇAMENTO	*DEFINIÇÃO*	*VANTAGENS*	*DESVANTAGENS*	*DESEMPENHO*
Imediato	*O campo operando contém o dado*	*Rapidez na execução da instrução*	*Limitação do tamanho do dado. Inadequado para o uso com dados de valor variável*	*Não requer acesso a memória principal. Mais rápido que o modo direto*
Direto	*O campo operando contém o endereço do dado*	*Flexibilidade no acesso a variáveis de valor diferente em cada execução do programa*	*Perda de tempo, se o dado é uma constante*	*Requer apenas um acesso a memória principal. Mais rápido que o modo indireto*
Indireto	*O campo operando corresponde ao endereço que contém a posição onde está o conteúdo desejado,*	*Manuseio de vetores (quando o modo indexado não está disponível). Usar como "ponteiro"*	*Muitos acessos à MP para execução*	*Requer 2 acessos a memória principal*

c) Qual é o objetivo do emprego do modo de endereçamento base mais deslocamento? Qual é a diferença de implementação e utilização entre esse modo e o modo indexado?

O base mais deslocamento tem como seu principal objetivo permitir a modificação de endereço de programas ou módulos destes (que é a relocação de programa), bastando para isso uma única alteração no registrador base.

O base mais deslocamento tem como característica o endereço ser obtido da soma do deslocamento com o registrador base, diferindo do modo indexado onde o do registrador base é fixo e variar no deslocamento, ao contrário deste onde o deslocamento é fixo e com a alteração do registrador base permite-se a mudança do endereço.

5. (1,0) Explique, comparando:

a) Compilação e Interpretação

A compilação consiste na análise de um programa escrito em linguagem de alto nível (programa fonte) e sua tradução em um programa em linguagem de máquina (programa objeto).

Na interpretação cada comando do código fonte é lido pelo interpretador, convertido em código executável e imediatamente executado antes do próximo comando.

A interpretação tem como vantagem sobre a compilação a capacidade de identificação e indicação de um erro no programa-fonte (incluindo erro da lógica do algoritmo) durante o processo de conversão do fonte para o executável.

A interpretação tem como desvantagem o consumo de memória devido ao fato de o interpretador permanecer na memória durante todo o processo de execução do programa. Na compilação o compilador somente é mantido na memória no processo de compilação e não utilizado durante a execução. Outra desvantagem da interpretação está na necessidade de tradução de partes que sejam executadas diversas vezes, como os loops que são traduzidos em cada passagem. No processo de compilação isto só ocorre uma única vez. Da mesma forma pode ocorrer para o programa inteiro, em caso de diversas execuções, ou seja, a cada execução uma nova interpretação.

Exemplos de linguagem interpretadas: ASP, BASIC, Java, PHP, Python, Lisp entre outras.

Exemplos de linguagem compilada: C, Pascal, Delphi, Visual Basic, entre outras.

b) Sistemas SMP e Sistemas NUMA

Sistemas SMP (ou UMA) têm como característica o acesso a todas as partes da memória principal com tempo de acesso uniforme. Em sistemas NUMA, todos os processadores possuem também acesso a todas as partes da memória principal podendo diferir o tempo de acesso em relação às posições da memória e processador.

Nos sistemas SMP o aumento no número de processadores tem como conseqüência problemas de tráfego no barramento comum degradando o desempenho. Uma solução para isto é a utilização de clusters, que tem, usualmente, como conseqüência alterações significativas na aplicação (software). Nos sistemas NUMA podem-se ter vários nós multiprocessadores, cada qual com seu próprio barramento, resultando em pequenas alterações na aplicação (software).

c) Arquiteturas RISC e Arquiteturas CISC

RISC: Reduced Instruction Set Computer – Computador com um conjunto reduzido de instruções
CISC - Complex Instruction Set Computer: Computador com um conjunto complexo de instruções

CISC: Principais características:
Possui microprogramação para aumento da quantidade de instruções incluindo novos modos de endereçamento, de forma a diminuir a complexidade dos compiladores e em conseqüência permitir linguagens de alto nível com comandos poderosos para facilitar a vida dos programadores. Em contrapartida, muitas instruções significam muitos bits em cada código de operação, instrução com maior comprimento e maior tempo de interpretação

Exemplos: IBM /370-168 (exemplo antigo) , Intel 80486 , Intel Pentium, Intel Xeon

RISC: Principais características:

Menor quantidade de instruções e tamanho fixo. Não há microprogramação. Permite uma execução otimizada, mesmo considerando que uma menor quantidade de instruções vá conduzir a programas mais longos. Uma maior quantidade de registradores e suas utilizações para passagem de parâmetros e recuperação dos dados, permitindo uma execução mais otimizada de chamada de funções. Menor quantidade de modos de endereçamento com o objetivo da redução de ciclos de relógio para execução das instruções. Instruções de formatos simples e únicos tiram maior proveito de execução com pipeline cujos estágios consomem o mesmo tempo.

Exemplos: Power PC, Série Sparc, Alpha 21064

6. (1,5) Considere um computador, cuja representação para ponto fixo e para ponto flutuante utilize 24 bits.
Na representação em ponto flutuante, as combinações possíveis de bits representam números normalizados do tipo $+/-(1,b_{-1}b_{-2}b_{-3}b_{-4}b_{-5}b_{-6}b_{7}b_{8}b_{-9}b_{-10}b_{-11}b_{-12}b_{-13}b_{-14} \times 2^{Expoente})$, onde o bit mais à esquerda representa o sinal (0 para números positivos e 1 para números negativos), os próximos 9 bits representam o expoente em excesso de 256 e os 14 bits seguintes representam os bits b_{-1} a b_{-14}, como mostrado na figura a seguir:

a) (0,2) Mostre a representação do inteiro -7471103 em complemento a 2 neste computador.

100011100000000000000001

b) (0,5) Para conjunto de bits obtido acima, indique o valor que está sendo representado em decimal, considerando-se que o conjunto representa:

(b.1) (0,1) um inteiro sem sinal

$2^{23} + 2^{19} + 2^{18} + 2^{17} + 2^{0} = 9.306.113$

(b.2) (0,2) um inteiro em sinal magnitude

$-(2^{19} + 2^{18} + 2^{17} + 2^{0}) = -917.505$

(b.3) (0,2) um número normalizado em ponto flutuante conforme descrição do enunciado. S Expoente representado em excesso de 256 b_{-1} b_{-2} b_{-3} b_{-4} b_{-5} b_{-6} b_{-7} b_{-8} b_{-9} b_{-10} b_{-11} b_{-12} b_{-13} b_{-14}

Sinal => 1 = negativo
Expoente => 000111000 = 56 – 256 = -200
Mantissa => 00000000000001 =
Resultado = 1,00000000000001 x 2^{-200} *= -6,2234 x* 10^{-61}

c) (0,2) Qual será a representação em ponto flutuante dos seguintes valores decimais neste computador:

c.1) +201,625

201,625 = 11001001, 101 = 1,1001001101 x 2^{7}
Sinal => 0 = positivo
Expoente => 7 + 256 = 100000111
Mantissa => ,10010011010000
Resposta: 0 100000111 10010011010000

c.2) –2,3

-2,3 = 10, 01001100110011001100 11... = 1,00100110011001100110011... x 2^{1}
Sinal => 1 = negativo
Expoente => 1 + 256 = 100000001
Mantissa => , 00100110011001
Resposta: 1 100000001 00100110011001

d) (0,4) Na representação para números em ponto flutuante, quando todos os bits do expoente forem iguais a 1 ou todos iguais a 0, o conjunto de bits não representa números normalizados e sim casos especiais, como no padrão IEEE 754. Indique o menor e o maior valor positivo e o menor e maior valor negativo de números normalizados que podem ser representados na representação em ponto flutuante para este computador. Mostre os valores em decimal.

Resposta correta para a questão

Menor valor positivo: 0 000000001 00000000000000 => 1,00000000000000 x 2^{-255} *= +1,72723 x* 10^{-77}
Maior valor positivo: 0 111111110 11111111111111 => 1,11111111111111 x 2^{+254} *= +5,78943 x* 10^{+76}

Maior valor negativo: 1 000000001 00000000000000 => -1,00000000000000 x 2^{-255} *= -1,72723 x* 10^{-77}
Menor valor negativo: 1 111111110 11111111111111 => -1,11111111111111 x 2^{+254} *= -5,78943 x* 10^{+76}

Resposta que poderá ser considerada para efeitos de correção da AD porque alguns tutores indicaram para não levar em consideração os casos especiais.

Menor valor positivo: 0 000000000 00000000000000 => 1,00000000000000 x 2^{-256} *= + 8,636168 x* 10^{-78}
Maior valor positivo: 0 111111111 11111111111111 => 1,11111111111111 x 2^{+255} *= +1,578855 x*10^{+77}

Maior valor negativo: 1 000000000 00000000000000 => -1,00000000000000 x 2^{-256} *= - 8,636168 x* 10^{-78}

Menor valor negativo: 1 111111111 11111111111111 => -1,11111111111111 x 2^{+255} = -1,578855 x10^{+77}

e) (0,2) Caso se utilize a representação em complemento a 2 para representar o expoente, indique como será a representação dos números dos itens c.1 e c.2.

201,625 = 11001001, 101 = 1,1001001101 x 2^{7}
Sinal => 0 = positivo
Expoente = +7 => 000000111 (por complemento a 2)
Mantissa => ,1001001101
Resposta: 0 000000111 10010011010000 (ou 00000001 11100100 11010000)

-2,3 = 10, 0100110011001100110011... = 1,00100110011001100110011... x 2^{1}
Sinal => 1 = negativo
Expoente = 1 => 000000001 (por complemento a 2)
Mantissa => , 00100110011001
Resposta: 1 000000001 00100110011001 (ou 10000000 01001001 10011001)

7. (0,8) O padrão mais utilizado para representar valores de variáveis do tipo float em C é o padrão IEEE 754 precisão simples. Para as variáveis do tipo int, utiliza-se a representação em complemento a 2 com 32 bits (Dica para resolver esta questão: Consultar http://www.lahey.com/float.htm).

a) (0,4) Três variáveis X, Y, Z, IY e W foram declaradas com o tipo float e as seguintes atribuições foram feitas a elas:

```
X=77777;
Y=7;
IY =1/Y;
Z=X/Y;
W=X * IY;
```

Verificou-se que ao se comparar as variáveis Z e W elas possuíam valores diferentes.

a.1) Mostre que matematicamente espera-se que o valor das duas variáveis seja igual .

W = Z ???
*Sabendo que W = X * IY e Z = X / Y*
*W = Z => X * IY = X / Y => como IY = 1 / Y*
*Então, X * (1 / Y) = X / Y => **X / Y = X / Y***

ou

W = Z ???
*W = W => W = X * IY , como IY = 1 / Y => W = X * (1 / Y) => W = X / Y , como Z = X / Y => W = Z !!!*

a.2) Descobriu-se que a razão dos valores serem diferentes é o fato de se estar utilizando representação IEEE 754, precisão simples. Explique porque utilizar esta representação pode fazer com que os valores sejam diferentes.

Os 24 bits da mantissa (incluindo o bit 1 implícito) em um número utilizando-se a representação IEEE 754, precisão simples, representam aproximadamente 7 dígitos decimais significativos. Neste sistema a representação de números reais não é contínua, ou seja, existem números que não podem ser representados. Um número que não pode ser representado tem então sua representação igualada à representação do valor representável mais próximo a ele, perdendo-se precisão.
No exemplo acima, o valor 1/7 não pode ser representado exatamente na representação IEEE 754, precisão simples, então ao se multiplicar 1/7 por 77777, não se terá o valor exato 11111. Já o valor 77777/7 pode ser

representado de forma exata na representação IEEE 754, precisão simples, fazendo com que os valores atribuídos a Z e W sejam considerados diferentes.

b) (0,4) Duas variáveis X e Y foram declaradas do tipo float e uma variável I do tipo int e as seguintes atribuições foram feitas a elas:

```
X=20.3;
Y=X*100.0;
I=Y;
```

Verificou-se que o valor da variável I é 2029, quando o esperado seria 2030. Explique o que pode estar ocorrendo.

20,30 armazenado como ponto flutuante = 0 10000011 01000100110011001100110 que corresponde a 20,299999237060546875
X = 20,3 => X armazena o valor 20,299999237060546875 (imprecisão na 6ª casa decimal)
*Y = X * 100,0 => Y passa a armazenar 2029,999....*
I = Y => Convertendo Y para I, há o truncamento não o arredondamento, desta forma I = 2029 (valor resultante)

8. (1,7) Considere um sistema com relógio de freqüência igual a 500 MHz. Este sistema pode realizar operações de entrada e saída por programação (*polling*) ou por interrupção. O número de ciclos de relógio para uma operação por programa é igual a 200 (inclui a chamada à rotina de "polling", acesso ao dispositivo e retorno da chamada). O número de ciclos de relógio para atender uma interrupção é igual a 500. Define-se overhead como o percentual de ciclos de relógio que são utilizados para operação de entrada e saída em relação ao número total de ciclos de relógio disponíveis em 1 segundo. Baseado nesta definição, responda as seguintes questões:

C = quantidade de ciclos por segundo = relógio do sistema = 500MHz = 500.000.000 ciclos / segundo

a) determine o overhead que ocorre quando se utiliza a interface por programa para os seguintes dispositivos:

a.1) (0,2) Um mouse que deve ser interrogado pelo sistema 50 vezes por segundo para garantir que nenhum movimento dele seja perdido.

Quantidade de vezes por segundo que o dispositivo deve ser "interrogado" por segundo = 50 vezes
Quantidade de ciclos por operação de E/S por programação = 200 ciclos
CM = Total de ciclos por segundo gastos "interrogando" o mouse = 50 x 200 = 10.000 ciclos/segundo
Overhead = CM / C = 10.000 / 500.000.000 = 0,00002 ou 0,002%

a.2) (0,3)Um disco rígido que gera dados a uma taxa de 12 MBytes/segundo para sua controladora que possui um buffer de 16 bytes, ou seja, os dados são transferidos para o sistema em unidades de 16 bytes.

Quantidade de vezes por segundo que o dispositivo deve ser "interrogado" por segundo = 12MBytes / 16 Bytes = 12.000.000 / 16 = 750.000 vezes por segundo
Quantidade de ciclos por operação de E/S por programação = 200 ciclos
CD = Total de ciclos por segundo gastos "interrogando" o disco = 750.000 x 200 = 150.000.000 ciclos/segundo
Overhead = CD / C = 150.000.000 / 500.000.000 = 0,3 ou 30%

b) determine o overhead que ocorre quando se utiliza a interface por interrupção para o disco rígido nas seguintes situações:

b.1) (0,4) O disco está ativo (gera dados) 100 % do tempo.

Quantidade de vezes que o dispositivo solicita interrupção por segundo = 12MBytes / 16 Bytes = 12.000.000 / 16 = 750.00 vezes por segundo
Quantidade de ciclos por operação de E/S por interrupção = 500 ciclos

CD = Total de ciclos por segundo gastos acessando o disco
= 750.000 x 500 = 375.000.000 ciclos/segundo
Overhead = CD / C = 375.000.000 / 500.000.000 = 0,75 ou 75%

b.2) (0,2) O disco está ativo (gera dados) em média 5 % do tempo.

Considerando que em apenas 5% do tempo o disco estará ativo, a quantidade de vezes que o dispositivo solicita interrupção será 0,005 x (12MBytes / 16 Bytes) = 0,005 x 750.000 = 3750 vezes por segundo

Quantidade de ciclos por operação de E/S por interrupção = 500 ciclos
CD = Total de ciclos por segundo gastos acessando o disco
= 3.750 x 500 = 1.875.000 ciclos/segundo
Overhead = CD / C = 1.875.000 / 500.000.000 = 0,00375 ou 3,75%

b.3) (0,6) Para se tentar diminuir o overhead, aumentou-se o tamanho do buffer da controladora para 64 bytes, ou seja somente quando o buffer estiver com 64 bytes é que deverá ser gerada uma transferência. Mostre se esta medida irá diminuir o overhead quando o disco está 100% ativo.

Quantidade de vezes que o dispositivo solicita interrupção por segundo = 12MBytes / 64 bytes
= 12.000.000 / 64 = 187.500 vezes por segundo
Quantidade de ciclos por operação de E/S por interrupção = 500 ciclos
CD = Total de ciclos por segundo gastos acessando o disco
= 187.500 x 500 = 93.750.000 ciclos/segundo
Overhead = CD / C = 93.750.000 / 500.000.000 = 0,1875 ou 18,75%

Esta medida irá diminuir o overhead.

9. (1,0) Explique o que são e como funcionam os arrays redundantes de discos (RAID).

Texto retirado do site http://www.infowester.com

*RAID é a sigla para **R**edundant **A**rray of **I**ndependent **D**isks. Sua definição em português seria "Matriz Redundante de Discos Independentes". Trata-se de uma tecnologia que combina vários discos rígidos (HD) para formar uma única unidade lógica, onde os mesmos dados são armazenados em todos (redundância). Em outras palavras, é um conjunto de HDs que funcionam como se fossem um só. Isso permite ter uma tolerância alta contra falhas, pois se um disco tiver problemas, os demais continuam funcionando, disponibilizando os dados. O RAID é uma tecnologia consolidada, que surgiu através de trabalho de pesquisadores da Universidade de Berkeley, na California (EUA) no final da década de 1980.*

Para que o RAID seja formado, é preciso utilizar pelo menos 2 HDs. O sistema operacional, neste caso, enxergará os discos como uma unidade lógica única. Quando há gravação de dados, os mesmos se repartem entre os discos do RAID (dependendo do nível). Com isso, além de garantir a disponibilidade dos dados em caso de falha de um disco, é possível também equilibrar o acesso às informações, de forma que não haja "gargalos".

A tecnologia RAID funciona de várias maneiras. Tais maneiras são conhecidas como "níveis de RAID". No total, existem 6 níveis básicos, os quais são mostrados a seguir:

***RAID nível 0 -** Este nível também é conhecido como "Striping" ou "Fracionamento". Nele, os dados são divididos em pequenos segmentos e distribuídos entre os discos. Este nível não oferece tolerância a falhas, pois não existe redundância. Isso significa que uma falha em qualquer um dos HDs pode ocasionar perda de informações. Por essa razão, o RAID 0 é usado para melhorar o desempenho do computador, uma vez que a distribuição dos dados entre os discos proporciona grande velocidade na gravação e leitura de informações. Quanto mais discos houver, mais velocidade é obtida. Isso porque, se os dados fossem gravados em um único disco, esse processo seria feito de forma seqüencial. Com o RAID, os dados cabíveis a cada disco são gravados ao mesmo tempo. O RAID 0, por ter estas características, é muito usado em aplicações de CAD e tratamento de imagens e vídeos.*

RAID nível 1 *- Também conhecido como "Mirroring" ou "Espelhamento", o RAID 1 funciona adicionando HDs paralelos aos HDs principais existentes no computador. Assim, se, por exemplo, um computador possui 2 discos, pode-se adicionar mais um HD para cada um, totalizando 4. Os discos que foram adicionados trabalham como uma cópia do primeiro. Assim, se o disco principal recebe dados, o disco adicionado também os recebe. Daí o nome de "espelhamento", pois um HD passa a ser uma cópia praticamente idêntica do outro. Dessa forma, se um dos HDs apresentar falha, o outro imediatamente pode assumir a operação e continuar a disponibilizar as informações. A conseqüência neste caso, é que a gravação de dados é mais lenta, pois é realizada duas vezes. No entanto, a leitura dessas informações é mais rápida, pois pode-se acessar duas fontes. Por esta razão, uma aplicação muito comum do RAID 1 é seu uso em servidores de arquivos.*

RAID nível 2 *- Este tipo também possibilita o fracionamento de dados entre os discos como o RAID nível 0 e armazena informações que possibilitam verificar e corrigir erros, utilizando técnicas de correção de erros do tipo códigos de Hamming.*

RAID nível 3 *- Neste nível, os dados são divididos entre os discos da matriz, exceto um, que armazena informações de paridade. Assim, todos os bytes dos dados tem sua paridade (acréscimo de 1 bit, que permite identificar erros) armazenada em um disco específico. Através da verificação desta informação, é possível assegurar a integridade dos dados, em casos de recuperação. Por isso e por permitir o uso de dados divididos entre vários discos, o RAID 3 consegue oferecer altas taxas de transferência e confiabilidade das informações. Para usar o RAID 3, pelo menos 3 discos são necessários.*

RAID nível 4 *- Este tipo de RAID, basicamente, divide os dados entre os discos, sendo que um é exclusivo para paridade. A diferença entre o nível 4 e o nível 3, é que o RAID 4 possibilita a utilização de segmentos maiores que o RAID 3. O RAID 4 é indicado para o armazenamento de arquivos grandes, onde é necessário assegurar a integridade das informações. Isso porque, neste nível, cada operação de gravação requer um novo cálculo de paridade, dando maior confiabilidade ao armazenamento (apesar de isso tornar as gravações de dados mais lentas).*

RAID nível 5 *- Este é muito semelhante ao nível 4, exceto o fato de que a paridade não fica destinada a um único disco, mas a toda a matriz de discos. Isso faz com que a gravação de dados seja mais rápida, pois não é necessário acessar um disco de paridade em cada gravação. Apesar disso, como a paridade é distribuída entre os discos, o nível 5 tende a ter um pouco menos de desempenho que o RAID 4. O RAID 5 é o nível mais utilizado e que oferece resultados satisfatórios em aplicações não muito pesadas. Este nível precisa de pelo menos 3 discos para funcionar.*

Existem 2 tipos de implementação de RAID, sendo uma baseada em hardware e a outra baseada em software. Cada uma possui vantagens e desvantagens. O primeiro tipo é a mais utilizada, pois não depende de sistema operacional (pois estes enxergam o RAID como um único disco grande) e fornece acessos rápidos, o que possibilita explorar integralmente seus recursos. Sua principal desvantagem é ser um tipo caro inicialmente. A foto ao lado mostra um poderoso sistema RAID baseado em hardware. Repare que na base da direita estão armazenados vários discos. O RAID baseado em hardware utiliza dispositivos denominados "controladores RAID", que podem ser, inclusive, conectados em slots PCI da placa-mãe do computador.

Já o RAID baseado em software não é muito utilizado, pois apesar de ser menos custoso, é mais lento, possui mais dificuldades de configuração e depende do sistema operacional para ter um desempenho satisfatório. Este tipo ainda fica dependente do poder de processamento do computador em que é utilizado.

Foto retirada do site www.infowest.com