GABARITO AD2 - Organização de Computadores 2012.1

Data de entrega: 22/05/2012

"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) (3,0) Crie um conjunto de instruções de um operando, definidas em Linguagem Assembly, necessárias para a realização de operações aritméticas e elabore um programa para o cálculo da seguinte equação:

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

Repita o problema para conjuntos de instruções de dois e três operandos, especificando o conjunto de instuções e o programa para cada um dos conjuntos.

I_ CONJUNTO DE INSTRUÇÕES PARA 1 OPERANDO:

LOAD X	=>	*ACC*	<- *(X)*
STORE X	=>	*(X)*	<- *ACC*
ADD X	=>	*ACC*	<- *ACC + (X)*
SUB X	=>	*ACC*	<- *ACC - (X)*
MUL X	=>	*ACC*	<- *ACC * (X)*
DIV X	=>	*ACC*	<- *ACC / (X)*

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 X , Y, Z      =>    (X) <- (Y) + (Z)
SUB X , Y, Z      =>    (X) <- (Y) - (Z)
MUL X , Y, Z      =>    (X) <- (Y) * (Z)
DIV X , Y, Z      =>    (X) <- (Y) / (Z)
```

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

PARA 1 OPERANDO:

```
LOAD B      =>  ACC  ← (B)
SUB F       =>     ACC  ← ACC - (F)
STORE T1    => (T1) ← ACC
LOAD E      =>  ACC ← (E)
```

```
DIV T1      =>     ACC ← ACC / (T1)
MUL D       =>     ACC ← ACC * (D)
STORE T1    => (T1) ← ACC
LOAD C      =>  ACC ← (C)
SUB T1      =>     ACC ← ACC - (T1)
ADD B       =>     ACC ← ACC + (B)
MUL B       =>     ACC ← ACC * (B)
MUL E       =>     ACC ← ACC * (E)
ADD A       =>     ACC ← ACC + (A)
STORE Y     => (Y) ← ACC
```

PARA 2 OPERANDOS:

MOV T1,B => (T1) ← (B)
SUB T1,F => (T1) ← (T1) – (F)
MOV T2,E => (T2) ← (E)
DIV T2,T1 => (T2) ← (T2) / (T1)
*MUL T2,D => (T2) ← (T2) * (D)*
MOV T1,C => (T1) ← (C)
SUB T1,T2 => (T1) ← (T1) – (T2)
ADD T1,B => (T1) ← (T1) + (B)
*MUL T1,B => (T1) ← (T1) * (B)*
*MUL T1,E => (T1) ← (T1) * (E)*
ADD T1,A => (T1) ← (T1) + (A)
MOV Y,T1 => (Y) ← (T1)

PARA 3 OPERANDOS:

```
SUB Y, B, F      =>    (Y) <- (B) - (F)
DIV Y, E, Y      =>    (Y) <- (E) / (Y)
MUL Y, D, Y      =>    (Y) <- (D) * (Y)
SUB Y, C, Y      =>    (Y) <- (C) - (Y)
ADD Y, Y, B      =>    (Y) <- (Y) + (B)
MUL Y, B, Y      =>    (Y) <- (B) * (Y)
MUL Y, Y, E      =>    (Y) <- (Y) * (E)
ADD Y, A, Y      =>    (Y) <- (Y) + (A)
```

2) (1,0) Descreva:

2.1) Os modos de endereçamentos, explicitando suas aplicações, vantagens e desvantagens.

Imediato: *Seu campo operando contém o dado, não requer acessos a memória principal sendo mais rápido que o modo direto. Possui como vantagem a rapidez na execução da instrução e como desvantagem a limitação do tamanho do dado, e é inadequado para o uso com dados de valor variável.*

Direto: *Seu campo operando contém o endereço do dado, requer apenas um acesso a memória principal, sendo mais rápido que o modo indireto. Possui como vantagem a flexibilidade no acesso a variáveis de valor diferente em cada execução do programa e como desvantagem a perda de tempo, se o dado for uma constante.*

Indireto: *O campo operando corresponde ao endereço que contém a posição onde está o conteúdo desejado, necessita de 2 acessos a memória principal, portanto mais lento que os 2 modos anteriores. Tem como vantagem o manuseio de vetores e utilização como ponteiro, e desvantagem como muitos acessos a memória principal.*

Base mais deslocamento: *o endereço é obtido da soma do campo de deslocamento com o conteúdo do registrador base. Este modo de endereçamento tem como principal objetivo permitir a modificação de endereço de programas ou módulos destes , bastando para isso alterar o registrador base.*

***Indexado**: o registrador é fixo e o deslocamento é variável. Este modo é freqüentemente utilizado no manuseio de estruturas de dados que são armazenadas em endereços contíguos de memória tais como vetores.*

2.2) Os modos compilação e interpretação, indicando em que circunstâncias um modo é mais vantajoso do que o outro.

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.

3. (1,0) Explique, comparando:

3.1 Computadores vetoriais e Computadores matriciais

O termo computadores vetoriais que correspondem a sistemas compostos por processadores vetoriais que freqüentemente são associados à organizações de ULAs com pipeline de operações.

E o termo computadores matriciais correspondem a sistemas compostos por processadores matriciais cuja organização é formada de ULAs paralelas.

3.2 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 podemos ter vários nós multiprocessadores, cada qual com seu próprio barramento, resultando em pequenas alterações na aplicação (software).

3.3 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.

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 reduzir 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.

4 (2,4) Considere um computador, cuja representação para ponto fixo e para ponto flutuante utilize 20 bits.

4.1 (0,8) Considere o seguinte conjunto de bits representado em hexadecimal C0C00. Indique o valor deste número em decimal, considerando-se que o conjunto representa:

$(C0C00)16 = (1100\ 0000\ 1100\ 0000\ 0000)_2$

4.1.1. um inteiro sem sinal

$2^{19} + 2^{18} + 2^{11} + 2^{10} = 789.504$

4.1.2. um inteiro em sinal magnitude

$-(2^{18} + 2^{11} + 2^{10}) = -265.216$

4.1.3. um inteiro em complemento a 2

$-2^{19} + (2^{18} + 2^{11} + 2^{10}) = -259.072$

4.2 (0,8) (0,8) Na representação em ponto flutuante, como na representação IEEE 754, utiliza-se o bit mais à esquerda para representar o sinal, os próximos 6 bits representam o expoente e os 13 bits seguintes representam os bits depois da vírgula. Quando todos os bits que representam o expoente são iguais a 0 ou iguais a 1 temos os casos especiais. Caso contrário, as combinações possíveis de bits representam números normalizados no formato +/- $(1, b_{-1}b_{-2}b_{-3}b_{-4}b_{-5}b_{-6}b_{-7}b_{-8}b_{-9}b_{-10}b_{-11}b_{-12}\ b_{-13}) \times 2^{expoente}$, onde o bit mais à esquerda representa o sinal (0 para números positivos e 1 para números negativos), os próximos 6 bits representam o expoente em excesso e os 13 bits seguintes representam os bits b_{-1} a b_{-13}, como mostrado na figura a seguir:

4.2.1. (0,2) Determine o valor do excesso utilizado, sabendo que os projetistas desta máquina utilizaram o mesmo critério utilizado pelo padrão IEEE754 para definir o valor do excesso.

O excesso será determinado pela fórmula: $2^{n-1} - 1$, sendo n o número de bits a ser utilizado. Como o expoente tem n=6 bits. Então, $2^{n-1} - 1 = 31$.

4.2.2. (0,2) Indique o valor do conjunto de bits do item anterior considerando que este conjunto está representando um número normalizado em ponto flutuante com a representação acima.

(1 100000 0110000000000)2
Sinal = 1 = negativo
Expoente = (1000000) = 32 - 31 = 1 (por excesso)
Mantissa = ,0110000000000
Normalizado: $-1,011000000000 \times 2^1 = -(10,11)_2 = -2,75$

4.2.3. (0,2) Qual será a representação em ponto flutuante dos seguintes valores decimais neste computador:

4.2.3.1. +50,625

+50,625 = 110010,101 = 1,10010101 x 2^6
Sinal = 0 = positivo
Expoente = 5 + 31 = 36 = 100100 (por excesso)
Mantissa = ,10010101
Resposta: 0 100100 1001010100000

4.2.3.2. –0,02

-0,02 = -0,00000101000111110101111 = -1,010001111010111 x 2^{-6}
Sinal = 1 = negativo
Expoente = -6 + 31 = 25 = 011001 (por excesso)
Mantissa = ,0100011110101
Resposta: 1 011001 0100011110101

4.2.4. (0,2) Indique o menor e o maior valor positivo normalizado na representação em ponto flutuante para este computador. Mostre os valores em decimal.

Maior número positivo: 0 111110 1111111111111 = +4294705152
Menor número positivo: 0 000001 0000000000000 = +9,31 x 10^{-1}

4.3 (0,8) Considere que este computador não detecta overflow.

4.3.1. (0,4) Indique o valor em decimal do resultado das seguintes operações aritméticas. Considere que cada conjunto de 20 bits representa um número em complemento a 2.

X = B0000 + 40F00

Em Hexadecimal	Em binário	Em decimal (complemento a 2)
+ B0000	+ 10110000000000000000	+ -327680
40F00	01000000111100000000	+265984
X= F0F00	X= 11110000111100000000	X= - 61696

Y = FC000 + FD000

Em Hexadecimal	Em binário	Em decimal (complemento a 2)
+ FC000	+ 11111100000000000000	+ - 16384
FD000	11111101000000000000	- 12288
Y= **1**F9000	Y= **1**11111001000000000000	Y= - 28672
o digito extra será descartado	bit de carry será descartado (sem overflow)	

4.3.2. (0,4) Indique o valor em decimal do resultado das seguintes operações aritméticas. Considere que cada conjunto de 20 bits representa um número representado em ponto flutuante igual à descrição do

enunciado.

X = B0000+40F00

Em Binário

B0000 = 1011000000000000000 = 1 011000 0000000000000
normalizado = - 1,00000000000000 x 2^{-7}

40F00 = 0100000111100000000 = 0 100000 0111100000000
normalizado = + 1,00111100000000 x 2^{+1}

Para realizar a soma temos que igualar os expoentes, utilizando como referência o maior expoente.

- 1,00000000000000 x 2^{-7} =
- 0,10000000000000 x 2^{-6} =
- 0,01000000000000 x 2^{-5} =
- 0,00100000000000 x 2^{-4} =
- 0,00010000000000 x 2^{-3} =
- 0,00001000000000 x 2^{-2} =
- 0,00000100000000 x 2^{-1} =
- 0,00000010000000 x 2^{0} =
- 0,00000001000000 x 2^{+1}

Teremos:

(- 0,00000001000000 x 2^{+1}) + (+ 1,00111100000000 x 2^{+1})

Com os expoentes iguais poderemos realizar a soma. Entretanto, são números com sinais diferentes. Realizaremos uma subtração entre as mantissas normalizadas (da mantissa de maior valor sem o sinal pela de menor valor). E adotaremos o sinal da mantisssa de maior valor sem o sinal (módulo).

```
  -  1,00111100000000
     0,00000001000000
   ____________________

     1,00111011000000
```

Resposta: 1,00111011000000 x 2^{+1}

Em decimal

B0000 = 1011000000000000000 = 1 011000 0000000000000
normalizado = - 1,00000000000000 x 2^{-7} = -0,0078125

40F00 = 0100000111100000000 = 0 100000 0111100000000
normalizado = + 1,00111100000000 x 2^{+1} = +2,46875

```
 +   -0,0078125
     +2,46875
 X= -2,4609375

Conferindo com a resposta da operação em binário:

normalizado =  + 1,00111011000000 x  2^+1  =  +2,4609375
```

Y = FC000+FD000

Em Binário

```
FC000 =   1111110000000000000   =   1  111110  0000000000000
normalizado =  - 1,0000000000000 x 2^+31

FD000 =    1111110100000000000   =   1  111110  1000000000000
normalizado =  - 1,1000000000000 x 2^+31

Com os expoentes iguais poderemos realizar a soma das mantissas normalizadas

      1,0000000000000
      1,1000000000000
   ---------------------------
     10,1000000000000

Resposta: -10,1000000000000 x  2^+31   =
        -1,01000000000000 x  2^+32 = -5,37 x 10^9

        (1  111111 01000000000000)

Obs.:  O resultado alcançado está fora da faixa de representação de número
normalizado do modelo de ponto flutuante descrito no título da questão.

      Menor número negativo:   1 111110  1111111111111   = -4,29 x 10+9
      Maior número negativo:   1 000001  0000000000000   = -9,31 x 10-10
```

5) (0,8) Explique como funciona o RAID (Redundant Array of Independent Disks) utilizado para conectar discos rígidos em um computador (sugestão de fonte de consulta: o site http://www.infowester.com/raid.php/. Na sua resposta indique as suas fontes de consulta).

Texto baseado no site: ***http://www.infowester.com/raid.php/***

RAID (Redundant Array of Independent Disks) significa em português Matriz Redundante de Discos Independentes. Trata-se, basicamente, de uma solução computacional que combina vários discos rígidos (HDs)

para formar uma única unidade lógica de armazenamento de dados. Uma unidade lógica trata de fazer com que o sistema operacional enxergue o conjunto de HDs como uma única unidade de armazenamento, independente da quantidade de dispositivos que estiver em uso.

Fazer com que várias unidades de armazenamento trabalhem em conjunto resulta em muitas possibilidades:

- Se um HD sofrer danos, os dados existentes nele não serão perdidos, pois podem ser replicados em outra unidade (redundância);

- É possível aumentar a capacidade de armazenamento a qualquer momento com a adição de mais HDs;

- O acesso à informação pode se tornar mais rápido, pois os dados são distribuídos em todos os discos;

- Dependendo do caso, há maior tolerância a falhas, pois o sistema não é paralisado se uma unidade parar de funcionar;

- Um sistema RAID pode ser mais barato que um dispositivo de armazenamento mais sofisticado e, ao mesmo tempo, oferecer praticamente os mesmos resultados.

Para que um sistema RAID seja criado, é necessário utilizar pelo menos dois HDs (ou SSDs). Mas não é só isso: é necessário também definir o nível de RAID do sistema. Cada nível possui características distintas justamente para atender às mais variadas necessidades. Os níveis de RAID podem ser:

*R**AID 0 :** Também conhecido como stripping (fracionamento), é aquele onde os dados são divididos em pequenos segmentos e distribuídos entre os discos. Trata-se de um nível que não oferece proteção contra falhas, já que nele não existe redundância. Isso significa que uma falha em qualquer um dos discos pode ocasionar perda de informações para o sistema todo, especialmente porque "pedaços" do mesmo arquivo podem ficar armazenados em discos diferentes. O foco do RAID 0 acaba sendo o desempenho. Por ter estas características, o RAID 0 é muito utilizado em aplicações que lidam com grandes volumes de dados e não podem apresentar lentidão, como tratamento de imagens e edição de vídeos***.**

***RAID 1:** Neste uma unidade "duplica" a outra, razão pela qual o nível também é conhecido como mirroring (espelhamento). Com isso, se o disco principal falhar, os dados podem ser recuperados imediatamente porque existem cópias no outro. O RAID 1 tem foco na proteção dos dados, ou seja, não torna o acesso mais rápido. Na verdade, pode até ocorrer uma ligeira perda de desempenho, uma vez que o processo de gravação acaba tendo que acontecer duas vezes, uma em cada unidade.*

***RAID 0+1** : É um sistema "híbrido" (hybrid RAID), ou seja, que combina RAID 0 com RAID 1. Para isso, o sistema precisa ter pelo menos quatro unidades de armazenamento, duas para cada nível. Assim, tem-se uma solução RAID que considera tanto o aspecto do desempenho quanto o da redundância.*

***RAID 2:** é um tipo de solução de armazenamento que conta com um mecanismo de detecção de falhas do tipo ECC (Error Correcting Code). Hoje, este nível quase não é mais utilizado, uma vez que praticamente todos os HDs contam com o referido recurso.*

***RAID 3:** Este é um nível parecido com o RAID 5 por utilizar paridade. A principal diferença é que o RAID 3 reserva uma unidade de armazenamento apenas para guardar as informações de paridade, razão pela qual são necessários pelo menos três discos para montar o sistema. Este nível também pode apresentar maior complexidade de implementação pelo fato de as operações de escrita e leitura de dados considerarem todos os discos em vez de tratá-los individualmente.*

***RAID 4:** Também utiliza o esquema de paridade, tendo funcionamento similar ao RAID 3, com o diferencial de dividir os dados em blocos maiores e de oferecer acesso individual a cada disco do sistema. Este nível pode apresentar algum comprometimento de desempenho, pois toda e qualquer operação de gravação exige atualização na unidade de paridade.*

***RAID 5**: É outro nível bastante conhecido. Nele, o aspecto da redundância também é considerado, mas de maneira diferente: em vez de existir uma unidade de armazenamento inteira como réplica, os próprios discos servem de proteção. Neste método de proteção, os dados são divididos em pequenos blocos. Cada um deles recebe um bit adicional - o bit de paridade - Em uma tarefa de verificação o sistema constatar o erro de um bit e se o sistema conseguir identificá-lo, conseguirá substituí-lo imediatamente. A restauração dos dados poderá ser feita inclusive depois de o HD ter sido trocado. Durante a substituição, é possível manter o sistema em funcionamento, principalmente com o uso de equipamentos que suportam* hot-swaping*, ou seja, a troca de componentes sem necessidade de desligamento do computador. Isso é possível porque os dados são distribuídos entre todos os discos. Caso um falhe, o esquema de paridade permite recuperar os dados a partir das informações existentes nas demais unidades.*

***RAID 6:** Trata-se de uma especificação mais recente e parecida com o RAID 5, mas com uma importante*

diferença: trabalha com dois bits de paridade. Com isso, é possível oferecer redundância para até dois HDs no sistema, em vez de apenas um.

JBOD (Just a Bunch Of Disks): *Sigla para Just a Bunch Of Disks (algo como "Apenas um Conjunto de Discos"). Não se trata de um nível de RAID, mas sim de um método que simplesmente permite o uso em conjunto de dois ou mais HDs (independente de sua capacidade) de forma a fazer com que o sistema operacional enxergue o arranjo como uma única unidade lógica. O JBOD é semelhante ao RAID, mas não possui foco em desempenho ou redundância, considerando apenas o aumento da capacidade de armazenamento. Aqui, os dados são simplesmente gravados e, quando um disco fica lotado, a operação continua no outro. Desta forma, se um HD sofrer danos, os dados existentes nos demais não são prejudicados.*

6) (1,8) Descreva detalhadamente os três possíveis métodos de comunicação entre um controlador de entrada e saída e a unidade central de processamento e memória principal: por E/S programada, por interrupção e por acesso direto à memória. Indique uma vantagem e uma desvantagem de cada método.

E/S por programa: *O processador tem controle direto sobre a operação de E/S, incluindo a detecção do estado do dispositivo, o envio de comandos de leitura ou escrita e transferência de dados. Para realizar uma transferência de dados, o processador envia um comando para o módulo de E/S e fica monitorando o módulo para identificar o momento em que a transferência pode ser realizada. Após detectar que o módulo está pronto, a transferência de dados é realizada através do envio de comandos de leitura ou escrita pelo processador. Se o processador for mais rápido que o módulo de E/S., essa espera representa um desperdício de tempo de processamento. Uma vantagem deste método é o hardware simples. As desvantagens são: utilização do processador para interrogar as interfaces, o que acarreta perda de ciclos de processador que poderiam ser utilizados na execução de outras instruções, utilização do processador para realizar a transferência de dados, o que também acarreta perda de ciclos de processador.*

E/S por interrupção: *Neste caso, o processador envia um comando para o módulo de E/S e continua a executar outras instruções, sendo interrompido pelo módulo quando ele estiver pronto para realizar a transferência de dados, que é executada pelo processador através da obtenção dos dados da memória principal, em uma operação de saída, e por armazenar dados na memória principal, em uma operação de entrada. A vantagem deste método é que não ocorre perda de ciclos de processador para interrogar a interface, já que neste caso, não se precisa mais interrogar a interface, ela avisa quando pronta. As desvantagens são: necessidade de um hardware adicional (controlador de interrupções , por exemplo), gerenciamento de múltiplas interrupções e perda de ciclos de relógio para salvar e recuperar o contexto dos programas que são interrompidos.*

E/S por DMA: *Nesse caso a transferência de dados entre o módulo de E/S e a memória principal é feita diretamente sem envolver o processador. Existe outro módulo denominado controlador de DMA que realiza a transferência direta de dados entre a memória e o módulo de E/S. Quando o processador deseja efetuar a transferência de um bloco de dados com um módulo de E/ , ele envia um comando para o controlador de DMA indicando o tipo de operação a ser realizada (leitura ou escrita de dados), endereço do módulo de E/S envolvido, endereço de memória para início da operação de leitura ou escrita de dados e número de palavras a serem lidas ou escritas. Depois de enviar estas informações ao controlador de DMA, o processador pode continuar executando outras instruções. O controlador de DMA executa a transferência de todo o bloco de dados e ao final envia um sinal de interrupção ao processador, indicando que a transferência foi realizada. As vantagens deste método são: permite transferência rápida entre interface e memória porque existe um controlador dedicado a realizá-la e libera a UCP para executar outras instruções não relacioandas a entrada e saída. A desvantagem é que precisamos de hardware adicional.*