Esta ficha foi elaborada para acompanhar o estudo da disciplina de Arquitetura de Computadores Avançada (a4s1) ou para complementar a preparação para os momentos de avaliação finais da mesma. Num segundo ficheiro poderás encontrar um conjunto de propostas de solução aos exercícios que estão nesta ficha. É conveniente relembrar que algum conteúdo destes documentos pode conter erros, aos quais se pede que sejam notificados pelas vias indicadas na página web, e que serão prontamente corrigidos, com indicações de novas versões.

1. Um determinado processador apresenta uma memória endereçável de 40 bits. Qual é o número máximo de bytes e de palavras duplas de 64 bits, exprimido aproximadamente numa potência de base 10?

Uma memória com um espaço endereçável de 40 bits, significa que temos 2^{40} endereços diferentes disponíveis para serem endereçados. Numa aproximação feita por defeito, podemos considerar que $2^{10} \approx 10^3$, isto é, $1024 \approx 1000$. Assim, como ter $2^{40} = 2^{10}. 2^{10}. 2^{10}. 2^{10} = (2^{10})^4$, podemos fazer a aproximação $(10^3)^4$, aplicando a suposição anteriormente feita, ficando com um número máximo de 10^{12} bytes em memória.

Por outro lado, para verificarmos o número de palavras duplas de 64 bits precisamos de ter consciência quantos bytes são 64 bits (8 bytes = 2^3 bytes). É importante passar a nossa unidade para bytes porque geralmente afirmamos que as memórias têm o seu tamanho por unidade de byte, não bits (dizemos que uma memória tem 1 Gb ou 8 Gb, nunca bits). Assim, temos de contar o número de blocos de 8 bytes que existem no espaço completo endereçável. Logo, aos 2^{40} dividimos por blocos de 2^3, obtendo 2^{37} blocos. Estes 2^{37}, assumindo a mesma aproximação que anteriormente, pode ser traduzido em $2^7 \times (2^{10})^3 = 128 \times (10^3)^3 = 128 \times 10^9 = 1.28 \times 10^{11}$ palavras duplas de 64 bits.

2. Qual é a diferença entre estrutura e funcionalidade e em que dois conceitos é que cada um se contextualiza, no âmbito da área de estudo da arquitetura de computadores?

Quando referimos a estrutura de um componente (quer seja lógico ou físico), geralmente estamos a referir a forma como este se organiza (como se apresenta) - conceito de organização -, numa abordagem que procura encontrar o ponto de maior rendimento através da obtenção da eficiência. Por outro lado, quando referimos a funcionalidade estamos a querer indicar o que é que um determinado componente produz, isto é, para que conjunto que entradas $X_1, X_2, \ldots, X_i$ é que um componente produz uma saída $Y_1, Y_2, \ldots, Y_j$, e sobre que forma é que o faz. Aqui, sobre o conceito de arquitetura, procura-se obter a maior eficácia.

3. Explica porque é que "tornar o caso comum mais rápido" é, frequentemente, indicado como um corolário à Lei de Amdahl.

A Lei de Amdahl mostra que o $speedup_{overall}$ é conseguido aumetando o rendimento de uma parte qualquer Y por uma constante, chamemos-lhe K. Se um programa inteiro for X+Y, então o speedup será de 1/(X+Y / K). Note-se que quão mais frequente uma porção de código é, a sua parte correspondente Y (ou X) tornar-se-á maior, isto é, assumindo que "o caso comum" é Y, fazendo com que o caso comum mais rápido torne o programa melhor.

4. Se com a Lei de Amdahl obtemos um speedup de 2.5 para um programa A e de 3.14159 para um programa B, o que é que se pode concluir acerca dos tempos de execução de ambos os programas depois da melhoria?

A primeira coisa que se pode concluir é que ambos os programas são executados de forma mais rápida. Assumindo que estamos perante um $speedup_{overall}$, então o programa A corre 2.5 vezes mais rápido e o programa B 3.14159 vezes mais rápido. Isto significa que os tempos de execução são menores. No entanto, não conseguimos inferir qualquer relação entre os desempenhos de ambos os programas.

5. Suponhamos que instalamos um compilador novo (o HyperOptimizing Compiler), abaixo abreviado como HO, que aumenta a rapidez de execução das operações de vírgula flutuante por um fator K (quaisquer outras operações não são abrangidas por este aumento). Quando criamos os programas para compilar, compilamos uma versão normal (sem a otimização) e uma versão otimizada para compararmos os desempenhos por tempo de execução. Sendo T o tempo original de execução e T_{HO} o tempo de execução da versão otimizada pelo compilador:

a) Se metade do tempo original de execução for despendido em instruções de vírgula flutuante, qual deverá ser o valor de K para que possamos atingir um speedup (efetivo ou overall) de 4?

Não interessa qual poderá ser o valor de K, porque não podemos obter um speedup de 4. Ao fazer apenas uma melhoria de metade das instruções, no melhor dos casos, o speedup seria de 2.

b) Escreve uma expressão para o novo tempo de execução T_{HO}, em termos de K, X e Y, onde X é a quantidade de tempo despendido em execução de instruções de vírgula flutuante antes da aplicação do compilador HO e Y é a quantidade de tempo despendido em execução de outras instruções (sem ser as de vírgula flutuante) antes da aplicação do compilador HO.

$T_{HO} = (X + Y/K)$

Sendo X é o tempo despendido na parte não melhorada, Y é a parte que pode ser melhorada e K é a melhoria em Y.

c) Exprime a Lei de Amdahl em termos das variáveis X, Y, K, T e T_{HO}. Atenção que podes não precisar de todas as variáveis!

A Lei de Amdahl é o quociente entre o tempo de execução para uma tarefa sem melhorias e o tempo de execução para uma tarefa com melhorias. Assim sendo temos que o $speedup_{overall} = T / T_{HO}$. Mas não podemos ficar por aqui, porque para o tempo de 1 T, temos que o tempo de execução para uma tarefa com melhorias deve ser algebricamente modificado pela diferença entre a fração de tempo não melhorado e o quociente da fração de tempo melhorado pela sua melhoria. Assim sendo, obtemos a seguinte equação:

$$speedup_{overall} = T/T_{HO} = X + Y / (X + Y/K)$$

Note-se que há uma pequena diferença em relação à Lei conforme a demos, dado que X+Y não pode ser normalizado ao tempo de 1 T. Seja $W = X / (X + Y)$ e $Z = Y / (X + Y)$, então agora já podemos escrever a seguinte equação:

$$speedup_{overall} = 1 / (W + Z/K)$$

d) Se um novo código com operações de vírgula flututante corre 10 vezes mais rápido que o antigo, o código original demora 50 segundos e o $speedup_{overall}$ é de 5, quanto tempo é que o código novo perde a executar as instruções de vírgula flutuante?

Consideremos que X é o tempo de execução da parte não melhorada e Y é o tempo de execução do código novo. O tempo de execução do código antigo é de 10Y. Daqui conseguimos dizer que o código antigo, que demora 50 segundos é o total de X com 10Y ($X + 10Y = 50$). Por outro lado, também sabemos que sendo que o speedup é 5, então ($X + Y = 10$). Subtraindo ambas as equações temos que $9Y = 40$, isto é, que $Y = 40/9$ segundos (aproximadamente 4.44 segundos).

6. Um novo sistema computacional está para ser construído com uns incríveis 144 micro-processadores idênticos entre si. Os 144 processadores podem ser usados num de três modos distintos: todos em simultâneo, 72 em simultâneo ou em série (um de cada vez).

a) Considera que se quer substituir o software barato (mas moroso) de manipulação de vírgula flutuante (FPS) por um conjunto de chips para o mesmo efeito (FPC). Antes da modificação, o FPS é usado 80% do tempo. Ao aceitar-se a solução do FPC, haverá um aumento de desempenho por um fator de 20 sobre o FPS. Escreve uma expressão para o speedup esperado quando o FPC está em uso.

Isto é uma aplicação direta da Lei de Amdahl. Usando a sua versão mais clara, temos que a percentagem de tempo de uso do FPS é 0.8, sendo a fração de tempo com melhoria 0.8 e o $speedup_{melhoria}$ para FPC é 20. Assim sendo, temos que $speedup_{overall} = 1 / ((1-0.8) + (0.8/20)) = 1 / (0.2 + (0.8/20)) = 0.008$ (0.8% melhor).

b) Há algum processo para o qual um speedup de 144 é praticável?

Esta é basicamente outra aplicação direta da Lei de Amdahl. Se quiséssemos usar todos os processadores para uma porção do tempo de execução, $frac_{enhanced}$, (fração de tempo em melhoria), então podemos considerar o $speedup_{melhoria}$ como 144. Temos assim $1 / ((1 - frac_{enhanced}) + (frac_{enhanced} / 144))$.
Para atingir um speedup efetivo de 144, então $frac_{enhanced}$ teria de ser 1, tornando agora uma outra questão mais relevante: é praticável esperar por um $frac_{enhanced}$ de 1? Por várias razões, é impraticável fazer uso, assim, de todos os processadores em simultâneo - algumas partes de processos, como a inicialização de dados ou a sincronização de operações do processador, são inerentemente feitas em série e não poderão ser melhoradas com a junção de outros processadores.

c) Um certo processo usa o modo em série em 30% do tempo. Dá uma expressão para o speedup do modo em série se metade dos processadores forem usados para 50% do tempo original de execução, e todos os 144 processadores forem usados nos 20% restantes do tempo original de execução.

Esta é uma aplicação da Lei de Amdahl com várias melhorias, cada uma adicionada separadamente. O speedup associado com o uso de metade dos processadores é de 144/2 = 72, tal como o speedup do uso de todos os processadores é de 144.
Em enunciado, diz-se que a fração de tempo em que se usam metade dos processadores é 0.5, fazendo $frac_{enh\_half} = 0.5$ e o $speedup_{melhoria\_half} = 72$. De forma semelhante, a fração de tempo em que se usam todos os processadores é 0.2, fazendo $frac_{enh\_all} = 0.2$ e o $speedup_{melhoria\_all} = 144$.
A fração que não pode ser melhorada é $1 - frac_{enh\_half} - frac_{enh\_all} = 1 - 0.5 - 0.2 = 0.3$. Então, podemos afirmar que o tempo de execução mais recente é $0.3 + (frac_{enh\_half} / speedup_{melhoria\_half}) + (frac_{enh\_all} / speedup_{melhoria\_all}) = 0.3 + + (0.5 / 72) + (0.2 / 144)$. Fazendo o speedup, temos que $1 / (0.3 + (0.5 / 72) + (0.2 / 144))$.

7. Os discos rígidos magnéticos são os dispositivos se somam mais falhas em termos dos sistemas computacionais atuais. De forma a aumentar a fiabilidade dos discos rígidos, os servidores (que ainda usam discos magnéticos) tipicamente usam configurações RAID, por exemplo, com 5 discos, onde 4 são usados para guardar os dados e um

quinto disco é usado para bits de paridade (estes bits podem ser usados para reconstruir os dados eventualmente perdidos ou com perdas). Uma configuração RAID de 5 discos não permite que 2 discos falhem simultaneamente. Para os cálculos seguintes, assume que o MTTF de cada unidade de disco magnético é de 250.000 horas.

a) Para o primeiro passo do cálculo da fiabilidade, consideremos que são necessárias 50 horas para efetuar a substituição de um disco pela manutenção dos servidores. Assumindo assim que o MTTR é de 50h, quão grande pode ser o MTTF de uma configuração de 5 discos, onde o disco que contém os bits de paridade é antes considerado como um disco de cópias de segurança ('backup'), de qualquer outro dos 4 discos restantes?

O MTTF de um disco é de 250000 horas, isto é (para simplificação) 2.5×10^5 h e o MTTR é 50h. Como são 5 discos o número de falhas no tempo (FIT) é igual a $5/MTTF = 5/(2.5 \times 10^5) = 1 / 5 \times 10^4$, onde o $MTTF_{tudo}$ é de 5×10^4. Sem a quinta unidade, o $MTTF = MTTF_{tudo} / (MTTR/MTTF_{4\_unidades})$. Como o $FIT_{4\_unidades} = 4/(2.5 \times 10^5)$, então o $MTTF_{4\_unidades} = 6.25 \times 10^4$. Assim sendo, $MTTF = 5 \times 10^4 / (50 / (6.25 \times 10^4)) = 6.25 \times 10^7$ horas.

b) Caso uma unidade falhe, os dados dessa unidade têm de ser reconstruídos usando a informação das paridades bit-a-bit. Assim, o MTTR não só depende do tempo que demora aos dados a transitarem para o disco novo, mas também do tempo de reconstrução dos mesmos. Por esta razão, uma forma melhorada para o MTTR envolve os dois contributos e depende do tamanho, S, dos discos, expressos em Gb (gigabytes):

$$MTTR(S) = 50 \text{ horas} + 0.1 \text{ hora} \times S$$

Como exemplo de aplicação da expressão acima, podemos ter que demora 0.1 hora a reconstruir 1 Gb de dados. Usando esta nova e melhorada fórmula para o cálculo do MTTR, determina o MTTF como uma função do tamanho do disco S.

$MTTF(S) = MTTF_{tudo} / (MTTR(S) / MTTF_{4\_unidades}) = 5 \times 10^4 / ((50 + 0.1S) / (6.25 \times 10^4)) = (3.125 \times 10^9) / (50 + 0.1S)$ horas.

c) Qual é a fórmula para o MTTF(S) de um sistema computacional que possui duas configurações RAID de 5 discos, cada um com capacidade S? Considera as duas configurações como conjuntos perfeitamente independentes de discos rígidos.

Neste caso o FIT é igual a $((50 + 0.1S) / (3.125 \times 10^9)) \times 2$. Assim, temos que o $MTTF_{2RAID} = (3.125 \times 10^9) / (2 \times (50 + 0.1S))$.

d) Qual é a fórmula para o MTTF(S) de um sistema computacional que possui K configurações RAID de 5 discos, cada um com capacidade S?

$MTTF_{K\_RAID} = (3.125 \times 10^9) / (K \times (50 + 0.1S))$.

e) De forma a poder fornecer uma solução de sistema de 10 Tb (terabytes), uma empresa tem a opção de escolher discos de 250 Gb (o que requer 10 configurações RAID de 5 discos) ou discos de 500 Gb (o que requer 5 configurações RAID de 5 discos). Qual das duas opções é que promete um MTTF melhor de acordo com a fórmula obtida em d)?

Para a primeira opção, em d), basta substituir S por 250 e K por 10. Assim, temos o seguinte:

$MTTF_{10\_RAID} = (3.125 \times 10^9) / (10 \times (50 + 0.1 \times 250)) = 4.16666 \times 10^6$ horas.

Para a segunda opção, em d), basta substituir S por 500 e K por 5. Assim, temos o seguinte:

$MTTF_{5\_RAID} = (3.125 \times 10^9) / (5 \times (50 + 0.1 \times 500)) = 6.25 \times 10^6$ horas.

Como a segunda opção tem um MTTF maior torna-se preferível perante a primeira opção.