Arquitectura de Computadores II
Colecção de Exercícios

Tiago M. Dias

Junho de 2008

Instituto Superior de Engenharia de Lisboa
Departamento de Engenharia de Electrónica e de Telecomunicações e de Computadores
Secção de Arquitecturas e Sistemas Operativos

Prefácio

Esta edição do texto "Arquitectura de Computadores II: Colecção de Exercícios" destina-se a apoiar o ensino da disciplina de *Arquitectura de Computadores II*, leccionada ao curso de Licenciatura em Engenharia Informática e de Computadores do Instituto Superior de Engenharia de Lisboa (ISEL).

A colecção inclui, essencialmente, exercícios não-resolvidos das matérias respeitantes a cada um dos capítulos em que se desdobra o programa da disciplina de *Arquitectura de Computadores II*. Os problemas apresentados resultam de uma compilação de exercícios já apresentados anteriormente em enunciados de testes da disciplina de *Arquitectura de Computadores II*, bem como da disciplina de *Microprocessaores II* do antigo curso de Bacharelato em Engenharia Informática e de Computadores do ISEL. A apresentação dos exercícios é feita por grupo temático e, dentro de cada grupo, por ordem cronológica inversa.

Futuramente, e sempre que oportuno, a colecção poderá vir a ser actualizada e/ou ampliada com a inclusão de novos problemas.

Tiago M. Dias
Abril de 2008

Versão 1.1, editada por Tiago M. Dias em Abril de 2008
Versão 1.0, editada por João Trindade em Janeiro de 2007

Conteúdo

Capítulo 1

Arquitectura do PC

1. [*2006/07v, Exame época especial*]
   Os computadores PC de todas as gerações têm internamente um *bus* com dados a 8 bits, mesmo que não disponham de fichas para inserção de placas, como é o caso das gerações actualmente no mercado.

   a) Para que serve esse *bus* interno e quais são as razões para que ele continue a existir?

   b) Como é suportado este *bus*, no caso de máquinas cujo processador tem dados a 32 ou 64 bits?

   c) Quais são as implicações, para o funcionamento deste *bus*, da enorme diferença de frequência de relógio entre os processadores dos PC mais antigos e dos actuais?

2. [*2006/07v, 1º Exame*]
   Na evolução do PC-AT com o processador 286 para o PC-AT com o processador 386, o barramento de dados passou de 16 para 32 bits, tendo no entanto sido garantida a compatibilidade de *software* no acesso aos periféricos. Enumere as vantagens e desvantagens desta solução.

3. [*2006/07v, 1º Teste*]
   Nos processadores intel com endereçamento e dados a 32 bit, há 30 linhas de endereço (`A2` a `A31`) e 4 sinais byte enable (`BE0#` a `BE3#`).

   a) Descreva o significado dos sinais `BE0#` a `BE3#`.

   b) O processador pode aceder a palavras com 1, 2 ou 4 bytes, em qualquer endereço, desdobrando automaticamente o acesso se necessário. Indique quais as configurações possíveis dos sinais `BE0#` a `BE3#`.

   c) Escreva a tabela de verdade para determinar os valores de A0 e A1 a partir dos sinais `BE0#` a `BE3#`.

   d) A arquitectura PC inclui circuitos de interface para ligação de periféricos e memórias a 8 bits. No caso de processadores com dados a 32 bits, indique o número de registos latch de 8 bits pertencentes a esta interface e descreva as circunstâncias em que estes registos são utilizados.

4. [*2006/07i, Exame época especial*]
   As versões da arquitectura PC posteriores ao modelo XT incluem circuitos de conversão para dispor de um bus de 8 bits a partir da largura de dados do processador.

   a) Explique a importância desta conversão.

   b) Indique o número de registos latch de 8 bits pertencentes a esta conversão, para os casos de processadores com dados a 16 e a 32 bits. Descreva as circunstâncias em que estes registos são utilizados.

5. [*2006/07i, Exame época especial*]
   Em relação ao Programmable Interrupt Controller (PIC) 8259, explique para que serve,

porque razão foi incluído no IBM-PC e por que motivo foi adicionado um segundo PIC no PC-AT.

6. [*2006/07i, 2º Exame*]
Comente a afirmação, relativa à evolução da arquitectura PC: "Enquanto nas gerações XT e AT-286 é viável expandir a DRAM do sistema usando placas de memória colocadas no bus de expansão, nas versões posteriores, com o processador 386 ou sucessores, só é interessante o uso de DRAM ligada à motherboard".

7. [*2006/07i, 1º Exame*]
Relativamente à evolução da arquitectura PC, explique como é assegurado o suporte de

   a) Placas de expansão desenhadas para PC-XT (8088) no PC-AT (286);

   b) Placas de expansão desenhadas para PC-AT (286) em sistemas com bus PCI.

8. [*2006/07i, 1º Teste*]
Relativamente à arquitectura PC, descreva as principais características das gerações PC-XT e PC-AT.

9. [*2006/07i, 1º Teste*]
Em todas as gerações de PC posteriores ao PC-XT, existe *hardware* para adaptar o barramento de 8 bit, de periféricos e memórias, à largura de dados do processador utilizado.

   a) Porquê?

   b) Que implicações resultariam se não existisse?

10. [*2005/06i, 1º Teste*]
Sabendo que os processadores do PC-XT (8088) e do PC-AT (80286) são arquitecturas a 16 bits,

    a) Indique as principais motivações que levaram ao aparecimento do PC-AT.

    b) Justifique as opções tomadas na ligação aos periféricos existentes na placa principal.

11. [*2005/06i, 1º Teste*]
Sabendo que a Intel disponibilizava, na mesma época, os processadores 8086, com *bus* de dados a 16 bits, e 8088, com *bus* de dados a 8 bits, que razão levou à opção pelo 8088 no projecto do PC-XT? Que inconvenientes resultaram dessa opção à época? Que inconvenientes surgiram, em consequência dessa opção, nas gerações seguintes?

12. Relativamente à evolução da arquitectura PC, descreva as vantagens e inconvenientes resultantes de utilizar, progressivamente, processadores com maior número de bits de dados.

13. Na evolução do PC-XT para o PC-AT, o *bus* de dados passou de 8 para 16 bits. Descreva as vantagens e inconvenientes desta modificação. Tenha em consideração que o software que realiza acessos a periféricos desenhado para o PC-XT deve continuar a ser executado.

14. Sabendo que a Intel disponibilizava, na mesma época, os processadores 8086, com *bus* de dados a 16 bits, e 8088, com *bus* de dados a 8 bits, que razão pode ter levado à opção pelo 8088 no projecto do PC-XT?

15. Indique as principais características das arquitecturas PC-XT e a PC-AT.

16. Relativamente à evolução do PC-XT, com processador 8088, para o PC-AT, com processador 286, descreva, justificando, os requisitos para o redesenho das fichas de expansão.

17. Na evolução do PC-XT para o PC-AT, o *bus* de dados passou de 8 para 16 bits. Descreva as vantagens e inconvenientes desta modificação. Tenha em conta a necessária compatibilidade de software no acesso a periféricos.

18. Enumere as principais diferenças entre a arquitectura PC-XT e a PC-AT.

19. Na maioria dos processadores, o *bus* de dados tem o mesmo número de bits que os registos internos e as operações aritméticas e lógicas.

    a) Quais são as razões desta regra?

b) Uma excepção é o 8088: os registos e a aritmética são a 16 bits mas o *bus* de dados é a 8 bits. Qual é o motivo? Que implicações tem?

c) Outra excepção é a gama de processadores Pentium: os registos e a aritmética são a 32 bits mas o *bus* de dados é a 64 bits. Qual é o motivo? O que permite a estes processadores tirar partido desta largura de *bus*?

20. Indique os principais requisitos a que teve de obedecer o desenho do PC-AT de forma a manter a compatibilidade com o PC-XT.

21. Pretende-se desenvolver um sistema para análise de dados. Os dados são sempre do mesmo tipo e analisados por um único programa. O sistema é baseado num processador Intel 486 e deve permitir um acesso à memória eficiente.

    a) Indique, justificando, qual a melhor solução para implementar a memória do sistema:

        i. uma memória cache para código e memória principal;

        ii. apenas memória principal.

    b) Mantinha a sua escolha se o processador a utilizar fosse um Intel 286? Justifique.

22. Comente a afirmação: "Numa arquitectura PC, dotada de uma *bridge* para ligar o *bus* do processador a um *bus* compatível com arquitecturas anteriores, a memória DRAM é ligada ao *bus* compatível."

23. Na evolução do PC-XT para o PC-AT, o *bus* de dados passou de 8 para 16 bits. Sabendo que o *hardware* resultaria mais simples se os periféricos fossem acedidos somente em endereços pares,

    a) Por que razão, no PC-AT, os periféricos são acedidos em endereços consecutivos (pares e ímpares)?

    b) Descreva o *hardware* existente no *bus* do PC-AT para suportar o acesso aos periféricos em endereços consecutivos (pares e ímpares).

24. Considerando diversos processadores, com diferentes larguras de *bus* de dados, é típico construir os sistemas de modo a aceder aos dispositivos de I/O a 8 bit em endereços: i) consecutivos, no caso de *buses* de 8 bits; ii) pares, no caso de *buses* a 16 bits; iii) múltiplos de 4, no caso de *buses* de 32 bits.

    a) Porquê?

    b) No caso de várias gerações da arquitectura PC usando processadores com estas larguras de *bus* de dados, não é seguida esta regra típica. Porquê? Que implicações daí resultam?

25. Na evolução da arquitectura PC, o *bus* de expansão em algumas variantes é directamente derivado do *bus* do processador; noutras tem uma concepção própria, independente do processador. Apresente, justificando, as vantagens e inconvenientes de cada uma dessas opções. Ilustre com exemplos.

26. Sabendo que os periféricos utilizados na arquitectura PC-AT são com interface a 8 bits.

    a) Indique os motivos que levaram à evolução para uma arquitectura com um processador de 16 bits.

    b) Explique como foi resolvido o problema de adaptação do *bus* do processador (16 bits) e dos periféricos (8 bits).

27. Nas arquitecturas PC das primeiras gerações as linhas de interrupção disponíveis no *bus* de expansão são sensíveis à transição ascendente com permanência a nível um. Indique, justificando, se é possível partilhar uma linha de interrupção, associando-lhe vários periféricos

    a) localizados na mesma placa de expansão;

    b) localizados em diversas placas de expansão.

28. Relativamente aos processadores Intel, de diversas gerações, com diferentes larguras de barramento de dados,

a) Por que motivo o barramento de endereço não contém A0 e A1 no caso de dados a 32 bits e não contém A0 a A3 no caso de dados a 64 bits?

b) Descreva a informação disponibilizada pelos sinais: A0 e BHE#, no caso de dados a 16 bits; BE0# a BE3#, no caso de dados a 32 bits; BE0# a BE7#, no caso de dados a 64 bits.

c) Descreva as implicações, internas e externas ao processador, de ser admitido o acesso a palavras em endereços não alinhados (não múltiplos da sua dimensão).

29. Nas arquitecturas PC podem-se destacar três gerações, PC-XT, PC-AT e PC-Pentium. Enumere as principais características de cada uma destas gerações.

Capítulo 2

Memória RAM

1. [*2006/07v, Exame época especial*]
   Relativamente à organização das memórias DRAM, descreva:

   a) A motivação para se usar uma estrutura em matriz;

   b) Os nomes, significado e funcionalidade dos pinos típicos de um circuito integrado, agrupando-os em endereço, controlo, dados e alimentação;

   c) A importância dos modos de acesso por página e a sua relação com a escolha dos bits de endereço, de maior ou menor peso, para a selecção de linha ou coluna.

2. [*2006/07v, Exame época especial*]
   Relativamente ao mecanismo de *precharge* da DRAM, explique:

   a) A razão da sua existência;

   b) O seu funcionamento;

   c) A sua relação com os tempos de acesso.

3. [*2006/07v, 2º Exame*]
   No contexto das optimizações realizadas no funcionamento das DRAMs, designadas por *page mode*, *hyper page mode* (EDO) e *interleaving*, indique para cada caso, justificando, se é aplicável a frase "... é mais rápido porque faz o acesso enquanto está a decorrer o acesso anterior".

4. [*2006/07v, 2º Exame*]
   Relativamente às DRAMs, o aumento da dimensão em bits tem como contrapartida o aumento da frequência de refrescamento.

   a) Este aumento de frequência é motivado por quê? De que forma se relaciona com o aumento da dimensão?

   b) Com o objectivo de minimizar o aumento da frequência de refrescamento, aumenta-se a capacidade dos condensadores que retêm a informação nas células de memória dinâmica. Porquê? Que consequências (vantagens e desvantagens) resultam desta solução.

5. [*2006/07v, 1º Exame*]
   Considerando que existem no mercado circuitos integrados de RAM estática com dimensões relativamente grandes, é possível conceber um computador cuja memória principal seja RAM estática em vez de RAM dinâmica.

   a) Que alterações resultariam dessa opção no comportamento da memória principal?

   b) Quais são as implicações dessa opção em relação à utilidade da cache existente nos processadores actuais?

   c) Porque não se encontram no mercado computadores construídos dessa forma?

6. [*2006/07v, 1º Exame*]
   Os ciclos de leitura e de escrita da DRAM incluem sempre uma fase de refrescamento. No caso da escrita, tendo em conta que o valor refrescado na célula acedida é esmagado pelo valor escrito, por que motivo é realizado o refrescamento?

7. [*2006/07v, 1º Teste*]
   Relativamente à ligação de DRAMs em *interleaving*, descreva

   a) O seu princípio de funcionamento;

   b) A motivação para o uso deste tipo de organização;

   c) As implicações que tem na escolha dos módulos de DRAM a utilizar. Exemplifique para o caso de instalar 512 Mbyte de DRAM num sistema com bus de dados a 64 bits.

8. [*2006/07v, 1º Teste*]
   Na evolução das DRAMs, o aumento de dimensão do chip, em bits, implica o aumento do tempo de retenção da informação em cada célula, na ausência de refrescamento. Descreva os motivos desta regra.

9. [*2006/07i, Exame época especial*]
   Para o projecto de um sistema embebido baseado no processador 486, pretende-se construir o módulo de memória principal com 64 Mbytes.

   a) Desenhe o módulo de memória utilizando circuitos integrados DRAM de $16M \times 1$. Quantos circuitos integrados são necessários e quantos pinos têm? Quantos ciclos são necessários para o seu refrescamento completo?

   b) Desenhe o módulo de memória utilizando circuitos integrados DRAM com a mesma capacidade em bits mas com organização a 4 bits de dados. Quantos circuitos integrados são necessários e quantos pinos têm? Quantos ciclos são necessários para o seu refrescamento completo?

   c) Desenhe o módulo de memória utilizando circuitos integrados SRAM com organização a 8 bits de dados, admitindo que estes circuitos têm a mesma área de silício que os de DRAM anteriores e que uma célula de memória SRAM ocupa o quádruplo da área de uma célula de DRAM. Quantos circuitos integrados são necessários e quantos pinos têm?

   d) Indique, justificando, qual das opções anteriores é preferível.

10. [*2006/07i, 2º Exame*]
    Na geração de SIMM de 8 bits de dados, utilizavam-se DRAMs organizadas com dados a 1 bit. Na geração de DIMM de 64 bits de dados, utilizam-se vulgarmente DRAMs organizadas com dados a 8 bits. Apresente as razões que motivam estas opções, tendo em conta:

    a) Espaço físico ocupado;

    b) Complexidade da lógica de controlo;

    c) Eficiência dos acessos.

    Pode ilustrar as razões com exemplos de dimensões de módulos de memória.

11. [*2006/07i, 1º Exame*]
    As células de memória das RAMs estáticas também são usadas, com modificações, nas RAMs dinâmicas. Para quê? Tendo em conta que leituras e escritas são feitas sobre o mesmo tipo de célula, por que motivo é mais lento o acesso às RAMs dinâmicas?

12. [*2006/07i, 1º Exame*]
    Nas memórias DRAM, os sinais de endereço são multiplexados – através dos mesmos pinos é passado o endereço de linha, registado pelo sinal `RAS`, e o endereço de coluna, registado pelo sinal `CAS`.

    a) Qual é a importância de o endereçamento ser multiplexado?

    b) Em alguma circunstância ocorre activação de `RAS` ou de `CAS` sem registar endereço? Justifique.

c) Em alguma circunstância há bits de endereço de coluna registados pelo sinal `RAS` ou bits de endereço de linha registados pelo sinal `CAS`? Justifique.

13. [*2006/07i, 1º Teste*]
Um determinado sistema funciona com dois módulos SIMM de 30 pinos mas não funciona com apenas um. Que característica do processador justifica este comportamento?

14. [*2006/07i, 1º Teste*]
Relativamente às DRAMs, descreva:

a) Os diferentes tipo de *refresh*.

b) A importância do *refresh* `Cas-Before-Ras` para o desenho do controlador das DRAMs.

c) A característica que possibilita o refrescamento em simultâneo de vários bancos de memória.

15. [*2006/07i, 1º Teste*]
Comente a seguinte afirmação: "Na evolução da arquitectura PC, com o i80386 torna-se evidente a necessidade de cache.".

16. [*2006/07i, 1º Teste*]
Considere uma cache do tipo 8-way set associative com 1024k byte de RAM de dados em linhas de 32 byte, integrada num sistema com endereçamento a 32 bit.

a) Determine, em bits, a dimensão dos campos OFFSET, INDEX (SET) e TAG.

b) Descreva uma possível política de substituição que seja eficiente.

c) A taxa de ocupação de uma cache com 4-way set associative, com a mesma dimensão de memória de dados, é menor que a de uma 8-way set associative. Indique um cenário que confirme esta afirmação.

17. [*2005/06i, 1º Exame*]
Pretende-se desenvolver um sistema para análise de dados. Os dados são sempre do mesmo tipo e analisados por um único programa. O sistema é baseado num processador Intel 486 e deve dispor permitir um acesso à memória eficiente.

a) Indique, justificando, qual a melhor solução para implementar a memória do sistema:

i. uma memória cache para código e memória principal;

ii. apenas memória principal.

b) Mantinha a sua escolha se o processador a utilizar fosse um Intel 286? Justifique.

18. [*2005/06i, 2º Exame*]
Admita uma arquitectura com dois bancos de memória `DRAM`, com controlador na placa principal que implementa uma política de refrescamento *RAS-only*.

a) Indique as razões que possibilitam a acção de refrescamento em simultâneo nos dois bancos.

b) Quais as vantagens de se utilizar esta técnica?

19. [*2005/06i, 1º Exame*]
Relativamente ao modo de acesso *fast-page-mode* (`EDO`) das `DRAMs`:

a) Indique as suas características.

b) Indique a(s) vantagem(ns) em relação ao *page-mode*.

20. [*2005/06i, 2º Teste*]
Relativamente às tecnologias de memórias `RAM` estudadas nas aulas:

a) Caracterize uma célula de memória `RAM` estática (`SRAM`);

b) Caracterize uma célula de memória `RAM` dinâmica (`DRAM`);

c) Indique possíveis cenários, na arquitectura do PC, em que utilizaria `RAM` estática e `RAM` dinâmica. Justifique as opções.

21. Na organização da matriz de memória das DRAMs alguns fabricantes optam por:
    - Uma matriz quadrada;
    - Matriz rectangular com mais colunas do que linhas;
    - Duas ou mais matrizes, em bancos.

    Apresente as vantagens que podem motivar cada uma destas opções.
22. Admita uma arquitectura com dois bancos de DRAM em que a acção de refrescamento é realizada em simultâneo.
    a) Indique as razões que possibilitam este paralelismo.
    b) Quais vantagens de utilizar esta técnica?
23. Um sistema funciona com quatro módulos SIMM de 30 pinos mas não funciona com apenas um ou dois. Que característica do processador justifica este comportamento?
24. Relativamente ao tempo de refrescamento, comente a frase "Quanto maior for a dimensão de uma DRAM, maior deve ser o tempo que cada célula retém os dados na ausência de refrescamento."
25. Um sistema funciona com dois módulos SIMM de 72 pinos mas não funciona com apenas um. Que característica do processador justifica este comportamento?
26. Admitindo que um circuito de DRAM de 16 Mbits pode ter as células de memória organizadas em matriz de 4096×4096 ou de 2048×8192
    a) Que razões podem levar o fabricante a optar entre as duas possibilidades?
    b) Se for o segundo caso, 2048 é o número de linhas ou de colunas? Porquê?
    c) Considerando um banco de DRAM formado por 8 *chips* do segundo tipo, com os pinos de endereço e controlo ligados em paralelo, quantos ciclos são necessários para o seu refrescamento completo?
27. Relativamente às tecnologias de memórias RAM dinâmicas estudada nas aulas:
    a) Explique, sucintamente, os diferentes tipos de *refresh*.
    b) Justifique as principais razões por que alguns fabricantes optam por utilizar matrizes não quadradas para implementação das memórias.
    c) Comente a afirmação: "Quanto maior for a dimensão de uma DRAM maior deve ser o tempo que cada célula mantém os dados na ausência de refrescamento."
28. Comente justificando a seguinte afirmação: "Para implementar um espaço de memória de 256 Mbytes pode conseguir-se melhor desempenho nos acessos usando 2 *chips* de memória DRAM de 128 Mbytes do que um único *chip* de memória DRAM de 256 Mbytes".
29. Admitindo que um circuito de DRAM de $4M \times 4$ bits pode ter as células de memória organizadas em 4 matrizes de 2048×2048 ou 4 matrizes de 1024×4096.
    a) Que razões podem levar o fabricante a optar entre as duas possibilidades?
    b) Se for o segundo caso, 4096 é o número de linhas ou de colunas?
    c) Considerando um banco de DRAM formado por 2 *chips* do segundo tipo, com os pinos de endereço e controlo ligados em paralelo, quantos ciclos são necessários para o seu refrescamento completo?
30. Admitindo um circuito de DRAM com células de memória organizadas em matriz de 2048×4096.
    a) Indique a dimensão de memória do circuito de DRAM.
    b) Na matriz de memória 4096 é o número de linhas ou de colunas? Justifique a opção.
    c) Admita um banco de DRAM formado por 4 circuitos, com os pinos de endereço e controlo ligados em paralelo, quantos ciclos são necessários para o seu refrescamento completo? Justifique.

31. Comente, justificando, a afirmação: "A escrita de um bit numa DRAM implica o refrescamento das células da mesma linha."
32. Considere que se pretende construir um módulo de memória dinâmica com $128M \times 8$ bits. Admita que pode utilizar circuitos de $128M \times 1$ bit ou de $32M \times 4$ bits.
Enumere as vantagens e inconvenientes de cada uma destas opções.
33. Relativamente aos ciclos de acesso a memórias DRAM,
    a) Qual o significado do intervalo mínimo entre acessos consecutivos? De que resulta esta exigência?
    b) Admitindo que dispõe de circuitos de memória que suportam acessos com a duração de 15 ns, separados por um intervalo mínimo de 40 ns, apresente uma organização das memórias para optimizar os tempos de acesso.
    c) Comente se há possibilidade de ocorrerem acessos não optimizados na organização relativa à alínea anterior.
34. Relativamente à tecnologia de RAM dinâmica, descreva resumidamente:
    a) A organização das células de memória em matriz;
    b) O funcionamento multiplexado dos pinos de endereço, para os diversos ciclos de acesso;
    c) O mecanismo de *precharge*;
35. Sabendo que as memórias dinâmicas, em comparação com as estáticas, apresentam algumas desvantagens, indique, justificando:
    a) Quais são essas desvantagens?
    b) Por que razões, apesar disso, se usam como memória principal nos computadores?
36. Comente a seguinte afirmação: "Um controlador de RAM dinâmica que implemente *interleaving* melhora o desempenho do sistema."
37. Considere um circuito de RAM dinâmica com uma matriz de células de memória de 1024 linhas e 4096 colunas.
    a) Qual a capacidade de memória e quantos ciclos são necessários para assegurar o seu refrescamento completo?
    b) Quais as vantagens desta organização da matriz de células relativamente a uma matriz com 4096 linhas e 1024 colunas?
    c) Diga, justificando, quantos circuitos destes são necessários para construir um módulo SIMM de 30 pinos? Que capacidade em bytes tem esse módulo? Quantos ciclos são necessários para o seu refrescamento completo?
38. Explique, resumidamente, o funcionamento de uma memória dinâmica, relativamente aos sinais de interface, ciclos de *refresh* e *precharge*.
39. Comente, justificando, a afirmação: "Apesar de apresentarem desvantagens em relação às memórias estáticas, as memórias dinâmicas continuam a ser utilizadas como memória principal dos computadores."
40. Descreva as principais características dos módulos de memória DRAM de 72 pinos.
41. Comparando dois circuitos integrados de memória RAM, um de dinâmica e outro de estática, com a mesma dimensão em bits, produzidos com tecnologia da mesma geração e com encapsulamento semelhante,
    a) Qual deles tem maior área de silício? Justifique.
    b) Qual deles ocupa mais espaço no circuito impresso? Justifique.
42. Comente, justificando, a afirmação: "O desempenho de uma arquitectura PC moderna pode ser melhorado se for utilizado um controlador de DRAM que funcione em *interleaving*."

Capítulo 3

Memória Cache

1. [*2006/07v, Exame época especial*]
   No contexto dos sistemas de cache n-*way set associative*, indique, justificando, se é verdadeira ou falsa a afirmação: "para carregar um novo conteúdo, pode ocorrer a necessidade de substituir uma linha preenchida, ainda que outras linhas estejam vazias".

2. [*2006/07v, Exame época especial*]
   Considerando uma cache de nível 2 com 256 kbyte de dados em linhas de 64 bytes, com organização 4-*way set-associative*, num computador com endereçamento a 32 bits, determine:

   a) Os bits de endereço que formam os campos `byte-offset`, `set` e `tag`;

   b) A dimensão total, em bits, da memória de gestão que inclui os campos de `Tag`, `Valid` e `Modified` (admitindo que a política de escrita é *write-back*);

   c) As alterações da memória de gestão se a organização fosse alterada para 8 vias, mantendo a dimensão total da memória de dados e o comprimento das linhas.

3. [*2006/07v, 2º Exame*]
   Admita a existência de um sistema de cache L2 2-*way set associative*, com as seguintes características:

   - `TAG`: 15 bits
   - `INDEX` (`SET`): 12 bits
   - `BYTE OFFSET`: 5 bits

   a) Determine a dimensão da memória de dados.

   b) Determine a capacidade (física) de endereçamento do sistema.

   c) Mantendo os campos `TAG`, `INDEX`, `BYTE OFFSET`, retirou-se uma *way* (via). Determine a nova dimensão da memória de dados.

   d) Mantendo a memória de dados com a mesma dimensão determinada na alínea a), retirou-se uma *way* (via). Determine, em bits, a dimensão dos campos `TAG`, `INDEX` e `BYTE OFFSET`.

   e) Indique, justificando, relativamente aos três sistemas de cache, o inicial, o determinado na alínea c) e o determinado na alínea d), por qual optava.

4. [*2006/07v, 1º Exame*]
   Admita a definição dos campos do endereço, para acesso a uma cache L2, como é mostrado na figura

A35 A18	A17 A6	A5 A0
`TAG`	`SET (INDEX)`	`BYTE OFFSET`

   a) Para a cache com estas definições, qual é a dimensão total mínima da memória de dados? Caracterize este sistema de cache, relativamente a número de vias, dimensão de cada via e dimensão da linha.

b) Mantendo os campos idênticos, duplicou-se o número de vias. Caracterize o novo sistema de cache.

c) Enumere as vantagens e desvantagens da solução apresentada na alínea anterior.

d) Indique a relação que existe entre: dimensão da `TAG` e a dimensão da memória de dados; dimensão do `SET` (`INDEX`) e a dimensão da memória de dados.

5. [*2006/07v, 1º Teste*]
Nos processadores Intel baseados na microarquitectura Core, há uma versão em que a cache de nível 2 é associativa de 8 vias com 2 Mbyte dados em linhas de 64 byte.

a) Considerando que o endereçamento destes processadores é a 36 bit, determine o número de linhas e os bits de endereço que definem os campos `BYTE OFFSET`, `SET` (`INDEX`) e `TAG`.

b) Noutra versão, a RAM de dados da cache tem 4 Mbyte. Indique as opções possíveis relativamente aos parâmetros da cache a alterar para ter esta dimensão.

c) Apresente as razões que no seu entender motivaram o fabricante a optar por duplicar o número de vias.

6. [*2006/07v, 1º Teste*]
Os processadores Intel baseados na microarquitectura Core, incluem no mesmo chip duas caches de primeiro nível (L1), de código e dados, e uma cache de segundo nível (L2), genérica. A comunicação entre a cache L1 de dados e a cache L2 é realizada através de um *bus* com 256 bit de dados.

a) Qual é a motivação?

b) Porque não se utiliza a mesma dimensão de *bus* entre a cache L1 de dados e o core de CPU nem entre a cache L2 e o exterior?

7. [*2006/07i, Exame época especial*]
Considere um processador com cache interna L1 e L2. Admita que a cache L2 é 4-*way set associative*, com 512 Kbyte de dados em linhas de 32 bytes.

a) Qual é a dimensão, em bits, da memória de `Tag`? Apresente os cálculos.

b) Que vantagens e inconvenientes teria a modificação para uma organização 8-*way*?

c) Que vantagens e inconvenientes teria a modificação da dimensão da linha para 64 bytes?

d) Admitindo a possibilidade de dispor de maior área de circuito integrado para memória de cache, explique os critérios a seguir para utilizar essa memória em `Data` ou em `Tag`.

8. [*2006/07i, 2º Exame*]
Para a concepção de uma cache, considerando uma determinada dimensão de RAM de dados, apresente as vantagens e inconvenientes de optar por mapeamento directo ou 2-*way set associative*.

9. [*2006/07i, 2º Exame*]
Numa cache do tipo 4-*way set associative*, cada valor de `SET` (`INDEX`) selecciona um campo de `TAG` ou quatro campos de `TAG` em simultâneo? Justifique.

10. [*2006/07i, 1º Exame*]
No "Intel 64 and IA-32 Architectures Optimization Refrrence Manual" (Nov. 06, pp. 3-61) afirma-se que: *O"n Intel Core 2 Duo, Intel Core Duo, Intel Core Solo, and Pentium M processors, there will be an excess of first-level cache misses for more than 8 simultaneous references to addresses that are apart by 4-KByte modulus.*" Tradução livre: Nos processadores Intel Core 2 Duo, Intel Core Duo, Intel Core Solo e Pentium M, a utilização simultânea de mais de 8 localizações afastadas entre si de múltiplos de 4 Kbytes resultará num excesso de ocorrências de cache-misses no primeiro nível de cache.
Sabendo que a dimensão de cada linha de cache é de 64 bytes:

a) Qual o tipo de organização da cache? Justifique.

b) Qual a capacidade (em dados) da cache? Apresente os cálculos.

c) Considerando um espaço de endereçamento a 32 bits, qual a capacidade da RAM de tags? Apresente os cálculos.

11. [*2005/06i, 2º Exame*]
Considere um sistema com endereçamento a 32 bits e dois níveis de cache de código. A cache L1 é do tipo mapeamento directo com 8 Kbytes de dados em linhas de 16 bytes. A cache L2 é do tipo associativo a 2-vias com 32 Kbytes.

    a) Determine para cada nível de cache, em bits, a capacidade da memória de `TAG`, bem como o número de bits no barramento de endereço usados para `INDEX` (`SET`) e `OFFSET`.

    b) Indique as vantagens de se utilizar uma cache do tipo associativa para implementação da cache L2.

    c) Indique a forma como a memória principal deve estar organizada para optimizar os acessos.

12. [*2005/06i, 1º Exame*]
Admitindo a existência de uma arquitectura PC com cache L2 de mapeamento directo com 32 Kbytes, indique, justificando, a melhor forma de aumentar a eficiência do sistema:

    a) substituir a cache por uma associativa de 2 vias com 32 Kbytes;

    b) substituir a cache por outra de mapeamento directo com 64 Kbytes.

13. [*2005/06i, 1º Teste*]
Comente a seguinte afirmação: "Na evolução da arquitectura PC, a vantagem de ter cache evidenciou-se a partir da geração baseada no 80386."

14. [*2005/06i, 1º Teste*]
Considere uma cache do tipo 2-*way set associative* com 256 kbytes de `RAM` de dados em linhas de 16 bytes, integrada num sistema com endereçamento a 32 bits.

    a) Explique o funcionamento da cache quando o processador faz um acesso à memória para leitura de dados.

    b) Determine, em bits, a capacidade da memória de `TAG`, bem como o número de bits no barramento de endereço usados para `INDEX` (`SET`) e `OFFSET`.

    c) De modo a reduzir o custo da cache, na sua implementação serão usados dois tipos de *chips* de memória com tempos de acesso diferentes. Diga, justificando, qual o tipo de *chips* que deve ser usado na implementação da `RAM` de `TAG` e para a implementação da `RAM` de dados.

15. Relativamente ao funcionamento da cache, descreva as políticas de actualização *write-through* e *write-back*, e indique as respectivas vantagens e desvantagens.

16. Considere, num sistema com endereçamento a 32 bits, uma cache com 512 kbytes de `RAM` de dados em linhas de 8 bytes. Admita duas opções de organização: mapeamento directo ou 4-*way set associative*.

    a) Determine, para cada organização, o número de bits utilizados para `OFFSET`, `SET` (`INDEX`) e `TAG`.

    b) Apresente as vantagens e desvantagens entre estas duas organizações.

    c) Descreva um cenário de utilização em que uma organização tenha preenchimento completo e a outra não.

17. Descreva as políticas de substituição das linhas de cache, para as caches de mapeamento directo e para as 4-*way set associative*.

18. Comente, justificando, a afirmação: "A existência de cache permite realizar menor número de acessos à DRAM e, além disso, melhora a eficiência dos acessos realizados".

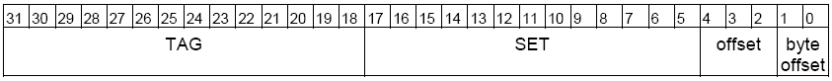
19. Qual a razão de, na implementação de cache, a `RAM` de `TAG` ser mais rápida do que a `RAM` de dados?

20. Que vantagens tem, nos processadores, a existência de caches L1 separadas, de código e dados, face à possibilidade de uma só cache L1?
21. A concepção de um sistema de cache pode optar por *direct map* (1 *way*) ou por n-*way set associative*, com n igual a 2, 4 ou 8, por exemplo. Discuta as vantagens e inconvenientes de optar por um valor de n menor ou maior.
22. Comente, justificando, a afirmação: "As DRAMs actuais estão concebidas para serem acedidas em *burst*, devido à existência de cache nos sistemas".
23. Considere a concepção de um sistema de cache, em que a memória de dados tem a dimensão de 1 Mbyte. Para calcular a dimensão da memória de TAG, discuta a influência que tem
    a) a dimensão das linhas de cache em que a memória de dados é dividida;
    b) o número de caminhos (n-*way set associative*) em que as linhas são agrupadas;
    c) a dimensão do espaço de endereçamento, do computador, suportado pela cache.
24. Explique o significado da memória de TAG nos sistemas de cache. Descreva a sua utilização, ilustrando com o caso de uma cache do tipo *direct map*.
25. Sabendo que, na geração actual de PCs, as caches de nível 1 e 2 residem ambas no circuito integrado do processador, indique porque existem os dois níveis e quais são as diferenças entre eles.
26. Considere uma arquitectura baseada num processador com endereçamento a 32 bits, dotada de cache 4-*way set associative*, com 512 Kbytes de RAM de dados em linhas de 32 bytes.
    a) Quantos são, no *bus* de endereço, os bits de SET?
    b) Quantos são, no *bus* de endereço, os bits de TAG?
    c) Qual é a dimensão total, em bits ou em bytes, da memória de TAG?
    d) Que vantagens e inconvenientes tem a opção de implementar a RAM de TAG com um número de bits de dados inferior ao determinado na alínea b)?
27. Comente, justificando, a afirmação: "A redução do número de bits das memórias de TAG da cache permite algum benefício económico mas limita a dimensão de DRAM que o sistema suporta".
28. Considere um sistema com endereçamento a 32 bits, com uma cache do tipo 4-*way set associative* com 2048 Kbytes de RAM de dados em linhas de 64 byte.
    a) Determine, em bits, a dimensão dos campos OFFSET, SET e TAG.
    b) Apresente as vantagens e desvantagens de realizar, em alternativa, a cache na forma 2-*way set associative* com linhas de 128 bytes, usando a mesma dimensão de RAM de dados.
    c) Indique, justificando, soluções que permitam reduzir os custos de implementação dos diferentes componentes da cache, considerando o dimensionamento da alínea a).
29. Relativamente aos sistemas de cache existentes nas arquitecturas de computadores descreva, justificando:
    a) Que benefícios introduziram?
    b) Quais as características dos acessos à memória que contribuem para o seu sucesso?
    c) Como funciona a associação entre os dados armazenados em cache e os dados correspondentes na DRAM geral da máquina?
30. Comente, justificando, a afirmação: "O número de bits das memórias de TAG da cache está relacionado com a dimensão da cache."
31. Comente a seguinte afirmação: "Se o custo da memória SRAM fosse idêntico ao da DRAM, o sistema de cache L2, podia ser eliminado."
32. Nos processadores actuais é comum encontrar a cache implementada no interior do processador, dividida em dois níveis, com as designações de L1 e L2.

a) Qual é a razão de existência deste dois níveis?

b) Quais são as características principais de cada um deles?

33. Considere, num sistema com endereçamento a 32 bits, uma cache com 1024 kbytes de `RAM` de dados, que interpreta os bits de endereço da seguinte forma:



a) Caracterize a cache, relativamente à dimensão de cada palavra, à dimensão das linhas, ao número de linhas por via e ao número de vias.

b) Apresente as vantagens e desvantagens de, usando a mesma dimensão de `RAM` de dados, aumentar o número de bits do campo `SET`.

34. Considere um sistema com endereçamento a 32 bits, com uma cache do tipo 2-*way set associative* com 1024 kbytes de `RAM` de dados em linhas de 32 bytes.

a) Determine, em bits, a dimensão dos campos `OFFSET`, `SET` e `TAG`.

b) Indique, justificando, soluções de implementação da cache que permitam reduzir o seu custo.

Capítulo 4

Assembly IA-32

1. [*2006/07i, Exame época especial*]
   Considere as seguintes definições:

   ```
   typedef struct {
           void * context;
           int (*func)(void * ctx, int n, ...);
   } Function;

   int TestAll(Function * f[], int nfuncs, int nwords, ...);
   ```

   Implemente, em *assembly* IA-32, a função `TestAll`, que, para cada um dos `nfuncs` elementos de `f`, chama a função indicada pelo respectivo campo `func`, passando como argumentos o valor do campo `context`, o valor de `nwords` e as `nwords` *words* recebidas por `TestAll` na lista de argumentos variável. A função `TestAll` retorna a soma dos valores retornados pelas várias chamadas.

2. [*2006/07i, 2º Exame*]
   Escreva, em *assembly* IA-32, o código da função

   ```
   char * getStringAddress(int count, const char * match, ...);
   ```

   que recebe, na lista de parâmetros variável, um conjunto de ponteiros para *string*s, com tipo `char *`, e compara cada uma das *strings* com a apontada por `match`, terminando quando tiver encontrado `count` *strings* idênticas a `match` e retornando o endereço da última delas. Se não houver *string*s idênticas em número suficiente, retorna `NULL`. A lista de parâmetros variável é terminada por um ponteiro com o valor `NULL`. Para fazer as comparações, deve utilizar a função de biblioteca

   ```
   int strcmp( const char * str1, const char * str2 );
   ```

   que retorna zero se as *strings* apontadas por `str1` e `str2` forem idênticas.

3. [*2006/07i, 2º Teste*]

   a) Implemente, em *assembly* IA-32, a seguinte função:

   ```
   char * struprcnt(char * str, unsigned int * pcnt);
   ```

   que converte para maiúsculas os caracteres da *string* `str`, deixando na localização apontada por `pcnt` o número de caracteres convertidos. A função retorna o endereço da sequência convertida.

   Para a conversão de cada carácter, utilize a função `toupper`, que recebe o valor de um carácter e, se este for uma letra minúscula, retorna o valor da maiúscula correspondente; nas outras situações retorna o valor recebido, sem modificação.

   ```
   int toupper(int c);
   ```

Se precisar de verificar se um carácter corresponde a uma letra minúscula, utilize a função `islower`, que recebe o valor de um carácter, retornando um valor diferente de zero se este corresponder a uma minúscula.

```
int islower(int c);
```

b) Considere as seguintes definições:

```
struct StrToConvert { unsigned int nconverted; char * str; };
unsigned int ConvertAllStrs(struct StrToConvert toConv[],
    unsigned int nconvs, char * (*strconvf)(char *, unsigned int *));
```

Implemente, em *assembly* IA-32, a função `ConvertAllStrs`, que aplica a função de conversão apontada por `strconvf` às *string*s apontadas pelos campos `str` de cada um dos nconvs elementos do *array* `toConv`, retornando o número total de caracteres convertidos.

A função de conversão apontada por `strconvf` recebe, como primeiro argumento o endereço da *string* a converter e, como segundo argumento, o ponteiro para a localização onde deixar o número de caracteres convertidos. Retorna o endereço da sequência convertida.

4. [*2006/07i, 2º Teste*]
Escreva, em *assembly* IA-32, o código da função

```
unsigned int strspn(const char * str, const char * strCharSet);
```

que retorna o índice do primeiro carácter de `str` que não ocorre em `strCharSet`.

5. [*2006/07i, 2º Teste*]
Escreva, em *assembly* IA-32, o código da função

```
const char * longest_match(const char * vstr[], unsigned int nstrs,
                           const char * strCharSet);
```

que recebe, em `vstr`, um *array* de `nstrs` *string*s e retorna a que começar com o maior número de caracteres presentes em `strCharSet`. Tire partido da função desenvolvida no exercício anterior.

6. [*2005/06i, 2º Exame*]
Escreva, em *assembly* IA-32, o código da função

```
int copyNotIf(const char *orig[], char *dest[], const char *match);
```

que copia para o *array* `dest` as *strings* do *array* `orig` que sejam diferentes de `match`. A função retorna o número de cópias realizadas. Deve utilizar a função de biblioteca,

```
int strcmp(const char *s1, const char *s2);
```

que retorna 0 se as *strings* `s1` e `s2` forem iguais.

7. [*2005/06i, 1º Exame*]
Admita a seguinte declaração em C:

```
struct analysis{ int average; int max; int min; };
```

Escreva, em *assembly* IA-32, o código da função

```
struct analysis stat(const int data[], const int weights[], const int n);
```

que retorna uma estrutura do tipo `analysis` com a média ponderada, o valor máximo e mínimo presentes no *array* `data`. Os valores dos pesos a usar no cálculo da média são passados no parâmetro `weights` e o número de elementos que compõem os dois *arrays* é passado no parâmetro `n`.

8. [*2005/06i, 1º Teste*]
Escreva, em *assembly* IA-32, o código da função

```
void PrintFmt(const char *fmt, ...);
```

que afixa na consola a mensagem `fmt`, substituindo as ocorrências do carácter de controlo '%' pelos valores dos parâmetros seguintes (do tipo `int`). Utilize as funções auxiliares:

```
void PutChar(char c);       /* afixa na consola o caracter c */
void PutInteger(int i);     /* afixa na consola o inteiro i  */
```

9. Escreva, em *assembly* IA-32, o código da função

   ```
   int DeleteWord(char * str, const char * word);
   ```

   que elimina da *string* passada em `str` todas as sequências de caracteres idênticas à passada em `word`. A função devolve o número de eliminações realizadas. Utilize as funções de biblioteca

   ```
   char * strcpy(char * dest, const char * src);
   ```

   que copia o conteúdo apontado por `src` para `dest`. Termina após copiar o carácter terminador, `0x00`.

   ```
   char * strstr(const char * str1, const char * str2);
   ```

   que devolve a primeira ocorrência da `str2` na `str1` ou `NULL` se não existir.

10. Considere a seguinte declaração

    ```
    typedef struct { void * retAddress; int stackFrameSize; } StackInfo;
    ```

    Escreva, em *assembly* IA-32, o código da função

    ```
    StackInfo getStackInfo();
    ```

    que retorna o endereço de retorno e a dimensão da *stack frame* da função chamadora.

11. Escreva, em *assembly* IA-32, o código da função

    ```
    void PrintFmt(const char *fmt, ...);
    ```

    que afixa na consola a mensagem `fmt`, substituindo as ocorrências do carácter de controlo '%' pelos valores dos parâmetros seguintes (do tipo `int`). Utilize as funções auxiliares:

    ```
    void PutChar(char c);    // afixa na consola o caracter c;
    void PutInteger(int i); // afixa na consola o inteiro i;
    ```

12. Relativamente à convenção de passagem de parâmetros *Variable Parameter List*, responda, justificando:

    a) O ajuste do *stack* deve ser feito pela função chamada, pela função chamadora, ou pode ser de ambas as formas?

    b) De acordo com a ordem de empilhamento, os parâmetros, referidos do primeiro para o último, ficam colocados em endereços de *stack* crescentes, decrescentes, ou pode ser de ambas as formas?

13. Considere a seguinte definição

    ```
    struct FuncStruct {
       void (* function)(void *, char);
       void  * param1;
       char    param2;
    };
    ```

    Escreva, em *assembly* IA-32, o código da função

    ```
    void executeList(FuncStruct list[], unsigned int nelem);
    ```

    que, para cada elemento de `list`, chama a função `function`, passando-lhe os parâmetros `param1` e `param2`. O número de elementos é definido pelo parâmetro `nelem`.

14. Escreva, em *assembly* IA-32, o código da função

    ```
    int strCapitalize(char * str);
    ```

que muda para maiúscula a primeira letra de cada palavra contida na *string* `str`. Cada palavra é separada por um ou mais caracteres de espaço (`0x20`). A função deve devolver o número de palavras que foram modificadas. Deve utilizar a função de biblioteca,

```
char * strchr(const char * s, int c);
```

que retorna um apontador para a primeira ocorrência do carácter c na *string* `s`.

15. Escreva, em *assembly* IA-32, o código da função

    ```
    char * getStringAddress(char * \textit{string}s[], char * sample);
    ```

    que devolve o endereço da primeira *string*, referenciada por um elemento de *strings*, com conteúdo igual ao de `sample` ou `NULL`, se não existir. O fim do *array strings* é indicado por um elemento (*pointer*) com o valor `NULL`. Pode tirar partido das funções de biblioteca (por exemplo, `strcmp`).

16. Admita a seguinte declaração em C++:

    ```
    struct St { int a; int b; int c; };
    ```

    Escreva, em *assembly* IA-32, o código da função

    ```
    int searchStruct(St structs[], unsigned int count, int val);
    ```

    que devolve o índice da primeira estrutura contida em `structs` com o campo a igual a `val` ou -1, se não existir. A dimensão do *array* `structs` é indicada pelo parâmetro `count`.

17. Admita a seguinte declaração em C++:

    ```
    struct St { int field_a; int field_b; int field_c; };
    ```

    Escreva, em *assembly* IA-32, o código da função

    ```
    void getField(St structs[], unsigned int count, int result[]);
    ```

    que preenche cada elemento do *array* `result` com o valor do campo `field_a` do elemento correspondente no *array* `structs`. O parâmetro `count` indica o número de elementos dos *arrays* `structs` e `result`.

18. Considere a função

    ```
    void conv(void * d, int des, void * s, int ses,
              int ec, void (* cef)(void * d, void * s));
    ```

    que copia dados de um *array* para outro, aplicando uma conversão.
    O *array* destinatário tem o endereço `d` (*destination*) e os seus elementos têm dimensão `des` (*destination element size*). O *array* originário tem o endereço `s` (*source*) e os seus elementos têm dimensão `ses` (*source element size*). O número de elementos dos *arrays* é `ec` (*element count*). O parâmetro `cef` (*convert element function*) representa o endereço de uma função que deve ser chamada para realizar a conversão e cópia de cada elemento. Os seus parâmetros `d` e `s` representam os endereços de destino e de origem do elemento manipulado. Para cada utilização deve ser criada uma função de conversão de elemento específica, que conhece os tipos de dados e, portanto, a sua dimensão.

    a) Escreva, em linguagem C, a função de conversão

       ```
       void charToInt(void *d, void *s);
       ```

       que, recebendo em `d` e `s` os endereços de um `int` e de um `char`, copia o valor convertendo de `char` para `int`.

    b) Escreva, em linguagem C, a definição de dois *arrays*, `ca` e `ia`, com elementos dos tipos `char` e `int`, respectivamente. Escreva também uma linha de código que utilize as funções `conv` e `charToInt` para copiar o conteúdo de `ca` para `ia`, convertendo adequadamente.

    c) Escreva, em *assembly* IA-32, a função `conv`.

19. Escreva, em *assembly* IA-32, o código da função

```
int arrayCopyInvert(void * dst, void * src,
                    unsigned int nel, unsigned int elsz);
```

que copia o conteúdo do *array* apontado por `src` para o *array* apontado por `dst`, invertendo a ordem dos elementos. Os parâmetros `nel` e `elsz` representam o número de elementos e a dimensão, em bytes, de cada elemento. A função devolve o número de bytes transferidos.

Para realizar a cópia de cada elemento deve utilizar a função de biblioteca

```
void * memcpy(char * d, const char * s, unsigned int n);
```

que copia `n` caracteres de `s` para `d` e devolve `d`.

20. Admita a seguinte declaração em C:

```
struct analysis { int average; int max; int min; };
```

Escreva, em *assembly* IA-32, o código da função

```
struct analysis stat(const int data[],
                     const int weights[], const int n);
```

que retorna uma estrutura do tipo `analysis` com a média ponderada e os valores máximo e mínimo presentes no *array* `data`. Os valores dos pesos a usar no cálculo da média são passados no parâmetro `weights` e o número de elementos que compõem os dois *arrays* é passado no parâmetro `n`.

21. Escreva, em *assembly* IA-32, o código da função

```
int arrayCount(void * base, int nel, int elsz,
               void * matchBase, int nelMatch);
```

que conta os elementos do *array* apontado por `base` que existem no *array* apontado por `matchBase`. Os parâmetros `nel` e `elsz` representam o número de elementos e a dimensão, em bytes, de cada elemento do *array* `base`. O parâmetro `nelMatch` representa o número de elementos do *array* `matchBase`.

Para realizar a comparação dos elementos, utilize a função de biblioteca

```
int memcmp(void * src1, void * src2, unsigned int size);
```

esta função devolve 0 se os elementos forem iguais.

22. Considere as seguintes declarações em C++

```
struct StA { char a[10]; int b; };
struct StB { char a[20]; int b; };
```

Escreva, em *assembly* IA-32, o código da função

```
void copyStructs(StA * x, StB * y, int n);
```

que, recebendo em `x` e `y` os endereços de dois *arrays* de estruturas do primeiro e segundo tipo, respectivamente, e em `n` o número de elementos destes *arrays*, copia o conteúdo de cada uma das estruturas de `x` para a posição correspondente em `y`.

Para copiar o campo a, utilize a função disponível na biblioteca

```
char * strcpy(char * dst, const char * src);
```

que copia a *string* apontada por `src` para a posição apontada por `dst`.

23. Escreva, em *assembly* IA-32, o código da função

```
unsigned int replace(const void * oldKey, const void * newKey,
                     void * base, unsigned int nel, unsigned int size);
```

que substitui, no *array* de elementos com endereço inicial `base`, os elementos idênticos ao elemento `oldKey` pelo novo elemento `newKey`. A dimensão dos elementos é dada pelo parâmetro `size` e o número de elementos do *array* é dado pelo parâmetro `nel`. A função devolve o número de elementos substituídos.

Para comparar os elementos utilize a função disponível na biblioteca:

```
int memcmp(const void * s1, const void * s2, unsigned int n);
```

que compara os `n` primeiros bytes dos blocos de memória apontados por `s1` e `s2`, devolvendo: um valor menor do que zero se `s1` for menor do que `s2`; zero se `s1` for igual a `s2` e; um valor maior do que zero se `s1` for maior do que `s2`.

Para copiar os elementos utilize a função disponível na biblioteca:

```
void * memcpy(void * dst, const void * src, unsigned int n);
```

que copia os `n` primeiros bytes do bloco de memória apontado por `src` para o bloco de memória apontado por `dst`, retornando dst tal como foi passado.

24. Escreva, em *assembly* IA-32, o código da função

```
void RepeatCall(int rep, void (*func)(int p), ...);
```

que executa `rep` vezes a chamada à função passada no parâmetro `func`.

Os argumentos para as sucessivas chamadas à função `func` são passados, por ordem de chamada, nos argumentos seguintes ao apontador para a função.

25. Considere a função

```
void translateIfDiff(const void * src, void * dst,
                     const void * key, unsigned int nelem,
                     unsigned int ses, unsigned int des,
                     void (* conv)(void * d, const void * s));
```

que copia, de um *array* para outro, os elementos diferentes da chave, aplicando uma conversão. O *array* de origem tem o endereço `src` e os seus elementos têm dimensão `ses`. O *array* de destino tem o endereço `dst` e os seus elementos têm dimensão `des`. O número de elementos de ambos os arrays é `nelem`. O parâmetro `key` é a chave para comparação de cada elemento do *array* `src`. O parâmetro `conv` representa o endereço de uma função que deve ser chamada para realizar a conversão. Os seus parâmetros `d` e `s` representam, respectivamente, os endereços de destino e de origem do elemento manipulado. Para cada caso deve ser criada uma função de conversão de elemento específica, que conhece os tipos de dados e, portanto, a sua dimensão.

Admita a declaração das seguintes estruturas

```
struct A { short a; short b; };
struct B { long a; long b; };
```

a) Escreva, em linguagem C, a definição de dois *arrays*, `aa` e `ab`, com elementos dos tipos `struct A` e `struct B`, respectivamente.

b) Escreva, em linguagem C, a função de conversão

```
void stAToStB(void * d, void * s);
```

que, recebendo em `d` o endereço de uma estrutura do tipo `struct B` e em `s` o endereço de uma estrutura do tipo `struct A`, copia os valores dos campos respectivos, convertendo-os de `short` para `long`.

c) Escreva a chamada à função `transalateIfDiff`, para copiar o conteúdo dos *arrays* declarados na alínea a). Utilize a função `stAToStB`, para realizar a conversão.

d) Escreva, em *assembly* IA-32, a função `transalateIfDiff`. Para a comparação com o parâmetro `key` utilize a função de biblioteca,

```
int memcmp(const char * p1, const char * p2, unsigned int n);
```

que retorna 0 se os blocos de memória apontados por `p1` e `p2` forem iguais.

26. Escreva, em *assembly* IA-32, uma função para realizar o histograma dos caracteres alfabéticos de uma *string*, com o seguinte assinatura

    ```
    void \textit{string}Histogram(char *str, int hist[26]);
    ```

    Cada elemento do *array* `hist` é preenchido com o número de ocorrências, na *string* `str`, do carácter alfabético correspondente. A contagem é realizada sem distinguir maiúsculas de minúsculas (`hist[0]` corresponde a 'A' ou 'a', `hist[1]` a 'B' ou 'b', etc.).
    Admita a existência de uma função, em linguagem C, que converte um carácter para maiúscula, com a seguinte assinatura:

    ```
    char toupper(char ch);
    ```

27. Escreva, em *assembly* IA-32, o código da função

    ```
    char * strSubstr(char * str1, char * str2);
    ```

    que devolve o apontador para a primeira ocorrência da *string* `str2` na *string* `str1`. Para comparar as *strings* utilize a função disponível na biblioteca

    ```
    int strncmp(const char * s1, const char * s1, unsigned int n);
    ```

    que compara até `n` caracters das *strings* `s1` e `s2`, devolvendo: um valor menor do que zero se `s1` for menor do que `s2`; zero se `s1` for igual a `s2` ou; um valor maior do que zero se `s1` for maior do que `s2`.
    Para determinar a dimensão das *strings* utilize a função disponivel na biblioteca

    ```
    int strlen(const char * s);
    ```

    que devolve a dimensão da *string* `s`.

28. Escreva, em *assembly* IA-32, o código da função

    ```
    unsigned int copyNotIf(const char * orig[], char * dest[],
                           const char * match);
    ```

    que copia para o *array* `dest` as *strings* do *array* `orig` que sejam diferentes de `match`. A função retorna o número de cópias realizadas.
    Deve utilizar a função de biblioteca,

    ```
    int strcmp(const char * s1, const char * s2);
    ```

    que retorna 0 se as *strings* `s1` e `s2` forem iguais.

29. a) Escreva, em *assembly* IA-32, a função

    ```
    int media(int a, int b);
    ```

    que devolve a média aritmética dos parâmetros.

    b) Escreva, em *assembly* IA-32, a função

    ```
    void interpol(int d[], int s[], int sz);
    ```

    que deposita em `d` os valores interpolados a partir de `s`, segundo as expressões

    ```
    d[i * 2] = s[i];
    d[i * 2 + 1] = media(s[i], s[i + 1]);
    ```

    O parâmetro `sz` representa o número de elementos do *array* `s`.
    Para calcular os elementos interpolados, deve utilizar a função `media` especificada na alínea a).

Capítulo 5

Barramento PCI

1. [*2006/07v, Exame época especial*]
   Considere um *bus* PCI a 32 bits com dois *masters*, designados por A e B, com um esquema de prioridade fixa, sendo o *master* A o mais prioritário.

   - O *master* A pretende realizar uma transacção de leitura de memória de oito bytes com os valores `0x12`, `0x34`, `0x56`, `0x78`, `0x11`, `0x22`, `0x33` e `0x44`, do endereço `0x00010002`. Admita que o *target* introduz dois estados de espera em cada fase de dados.
   - O *master* B pretende realizar uma transacção de escrita em memória, de um byte com o valor `0x5a` no endereço `0x0008f001`.

   Supondo que os dois masters activam `REQ#` ao mesmo tempo, desenhe os diagramas temporais para estas transacções. Represente a evolução de todos os sinais envolvidos (protocolo de arbitragem e de transferência). Os sinais `AD`[31:0] e `CBE#`[3:0] devem ser representados por valores em hexadecimal ou binário.

2. [*2006/07v, Exame época especial*]
   Relativamente ao sistema de interrupções do *bus* PCI,

   a) Descreva a motivação para as linhas de interrupção terem sensibilidade a nível e *active-low*;

   b) Considere um *bus* PCI e com quatro fichas que dispõe de quatro linhas de interrupção. Desenhe as ligações das linhas de interrupção às fichas, de modo que: se em todas as fichas forem inseridas placas e as placas usarem o mesmo número de linhas, as interrupções sejam uniformemente distribuídas pelas quatro linhas do sistema.

3. [*2006/07v, 2º Exame*]
   Considere um sistema com bus PCI a 32 bit com três *masters*, designados por A, B e C, com um esquema de prioridades fixas em que *master* A é o mais prioritário e o *master* C o menos prioritário.

   - O *master* A pretende fazer uma transacção: escrita de 2 bytes com os valores `0x33` e `0xef` da memória com o endereço inicial `0x0000ffff`; Admita que o *target* introduz um estado de espera em cada fase de dados.
   - O *master* B pretende fazer uma transacção:leitura de 4 bytes com o valores `0x46`, `0xa3`, `0xf7` e `0xcc` da memória com o endereço inicial `0x0900010c`; Admita que o *master* introduz um estado de espera.
   - O *master* C pretende fazer uma transacção: escrita de 3 bytes com o valor `0x1d`, `0x32` e `0x40`, na memória com o endereço inicial `0x01111111`.

   Supondo que os *masters* A e C pedem acesso ao *bus* após o *master* B tomar posse do mesmo para realizar a transacção, desenhe os diagramas temporais das transacções apresentadas. Represente a evolução de todos os sinais envolvidos (protocolo de arbitragem e de transferência). Os sinais `AD`[31:0] e `CBE#`[3:0] devem ser representados por valores em hexadecimal.

4. [*2006/07v, 1º Exame*]
   Na arbitragem entre dois *masters*, o árbitro pode desactivar um `GNT#` e activar o outro no mesmo ciclo de relógio ou em ciclos consecutivos.

   a) Em que situações ocorre cada um dos dois casos?

   b) Que motivo justifica a existência destas duas possibilidades?

5. [*2006/07v, 1º Exame*]
   O barramento PCI sendo um barramento *master/slave*, suporta o conceito de *multi-master*. Que importância tem este característica, quando a arquitectura que suporta o barramento inclui, tipicamente, um único processador?

6. [*2006/07v, 1º Exame*]
   Relativamente à situação designada por *turn-around* no *bus* PCI,

   a) Descreva o que significa, porque ocorre e que consequências tem na evolução temporal dos sinais.

   b) Enumere as situações em que ocorre. Justifique.

   c) A frequência de *turn-around* nos sinais `AD` é inferior, igual ou superior à dos sinais `CBE#`? Justifique.

7. [*2006/07v, 2º Teste*]
   Relativamente à arbitragem do bus PCI,

   a) Por que razão, para um master iniciar uma transacção, não basta ter o sinal `GNT#` mas é também necessário observar os sinais `FRAME#` e `IRDY#` inactivos?

   b) Se o sistema fosse concebido de modo que bastasse `GNT#` activo para o master iniciar a transacção, que implicações teria essa opção?

8. [*2006/07v, 2º Teste*]
   Um *master* PCI que usa os comandos `Memory Read Line`, `Memory Read Multiple` e `Memory Write and Invalidate` tem que disponibilizar, num registo de configuração, a dimensão da linha de cache. Porquê?

9. [*2006/07v, 2º Teste*]
   Explique a motivação para o PCI suportar a inserção de estados de espera, o significado dos sinais `IRDY#` e `TRDY#` e a condição para que se conclua uma fase de dados.

10. [*2006/07v, 2º Teste*]
    Considere um bus PCI a 32 bit com dois *masters*, designados por A e B, com um esquema de prioridades fixas, sendo o *master* A mais prioritário:

    a) O *master* A pretende realizar a escrita de um bloco de dados com 3 bytes com os valores `0x11`, `0x66`, `0xff` localizado no endereço de memória `0x002343a1`. Admita que o *target* introduz um estado de espera em cada fase de dados.

    b) O *master* B pretende realizar uma leitura de dois blocos de dados. Um com 1 byte com o valor `0x11h` localizado no endereço de memória `0xa0000bdc`; o outro com 6 bytes com os valores `0x31`, `0x32`, `0x33`, `0x34`, `0x35`, `0x36`, localizado no endereço de memória `0x08acc2dc`.

    Considerando que o *master* A manifesta a intenção de realizar a sua transacção quando o *master* B já está a realizar a primeira transacção, desenhe os diagramas temporais para estas transacções. Represente a evolução de todos os sinais envolvidos (protocolo de arbitragem e de transferência). Os sinais `AD`[31:0] e `CBE#`[3:0] devem ser representados por valores em hexadecimal ou binário.

11. [*2006/07i, Exame época especial*]
    Considere um bus PCI a 32 bits com dois *masters*, designados por A e B, com um esquema de prioridade rotativa, sendo o *master* A o primeiro a obter o *bus* no conjunto de transacções seguinte:

a) O *master* A pretende realizar duas transacções de leitura de memória de quatro bytes, a primeira com os valores 0x12, 0x34, 0x56 e 0x78, do endereço 0x00010002 e a segunda com os valores 0x11, 0x22, 0x33 e 0x44, do endereço 0x00018004.

b) O *master* B pretende realizar duas transacções de escrita em memória, a primeira de um byte com o valor 0x5a no endereço 0x0008f001 e a segunda de oito bytes com os valores 0x32, 0xf4, 0xff, 0xfd, 0x51, 0xde, 0x78 e 0xcc, no endereço 0x0008f002. Admita que o *target* introduz um estado de espera em cada fase de dados.

Desenhe os diagramas temporais para estas transacções. Represente a evolução de todos os sinais envolvidos (protocolo de arbitragem e de transferência). Os sinais `AD`[31:0] e `CBE#`[3:0] devem ser representados por valores em hexadecimal ou binário.

12. [*2006/07i, Exame época especial*]
Relativamente ao mecanismo de transferência directa de dados entre periféricos e a memória,

a) Apresente a motivação para a existência deste mecanismo.

b) Um periférico localizado numa placa de expansão PCI pode utilizar transferências deste tipo? Se não, porquê? Se sim, que características deve ter essa placa?

13. [*2006/07i, 2º Exame*]
Considere um sistema com bus PCI a 32 bit com dois *masters*, designados por A e B, com um esquema de prioridades fixas em que A é o mais prioritário. O *master* A pretende fazer uma transacção:

a) Escrita de 5 bytes com os valores 0x06, 0x43, 0x75, 0x5f e 0xff da memória com o endereço inicial 0x10da0c0a; Admita que o *target* introduz um estado de espera em cada fase de dados.

O *master* B pretende fazer duas transacções:

a) Leitura de 4 bytes com o valores 0xa6, 0xc3, 0x47 e 0x3c da memória com o endereço inicial 0x00800104; Admita que o *master* introduz um estado de espera;

b) Escrita de 2 bytes com o valor 0x2d e 0x04, na memória com o endereço inicial 0x0080100c.

Supondo que o *master* A pede acesso ao *bus* após o *master* B tomar posse para realizar a primeira transacção, desenhe os diagramas temporais destas transacções. Represente a evolução de todos os sinais envolvidos (protocolo de arbitragem e de transferência). Os sinais `AD`[31:0] e `CBE#`[3:0] devem ser representados por valores em hexadecimal.

14. [*2006/07i, 2º Teste*]
Relativamente ao bus PCI,

a) Descreva os significados dos sinais `C/BE#`[3:0].

b) Na leitura de diagramas temporais, é possível distinguir se uma transacção é de leitura ou escrita sem observar os sinais `C/BE#`[3:0]?

15. [*2006/07i, 2º Teste*]
A norma PCI especifica um bus *master*/*slave* que suporta a coexistência de vários *masters*. A norma USB especifica um bus *master*/*slave* com apenas um *master*.

a) No caso do PCI, como é gerido o acesso ao bus quando vários dispositivos têm dados para transferir?

b) No caso do USB, como é gerido o acesso ao bus quando vários dispositivos têm dados para transferir?

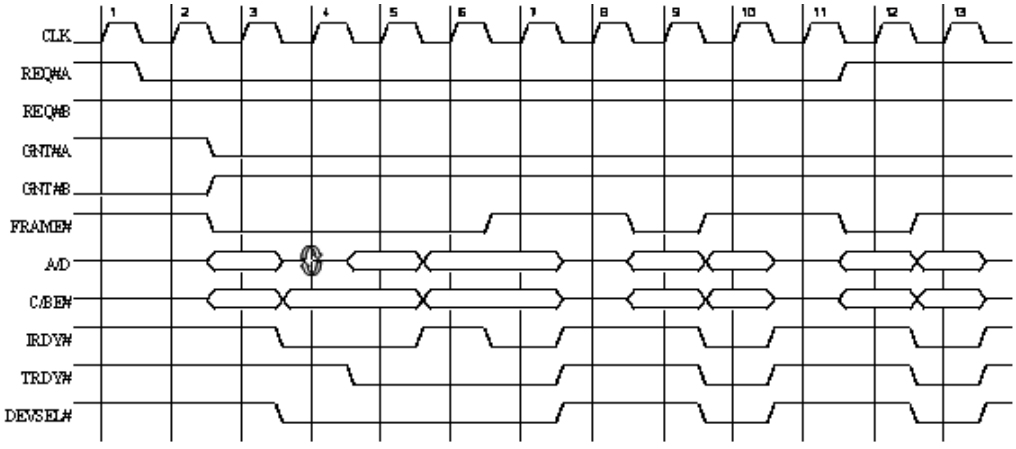
16. [*2006/07i, 2º Teste*]
No bus PCI, o final de uma transacção pode ser identificado pela ocorrência simultânea de `IRDY#`=1 e `TRDY#`=1?

17. [*2006/07i, 2º Teste*]
A associação da mesma linha de interrupção do PIC (Programmable Interrupt Controller) a vários dispositivos obriga à execução de múltiplos *handlers* (rotinas de tratamento de interrupção) sempre que um desses dispositivos gerar uma interrupção. Por outro lado, a norma PCI obriga a que todos os dispositivos que possam gerar interrupções utilizem (pelo menos) a linha `INTA#`.
Comente a afirmação: "O atendimento de uma interrupção `INTA#` obriga, tipicamente, à execução de uma cadeia de *handlers* muito mais longa do que o de uma interrupção `INTD#`."

18. [*2005/06i, 2º Exame*]
Comente as seguintes afirmações:
    a) "Num bus PCI multi-master, um master não pode ficar em posse do bus eternamente independentemente da sua prioridade."
    b) "O bus PCI pelo facto de suportar transferência em burst não permite a detecção e correcção de erros."

19. [*2005/06i, 2º Exame*]
Considere um sistema com bus PCI a 32-bit com dois *masters*, designados por A e B, com um esquema de prioridades rotativas em que o master A é o mais prioritário no início da seguinte sequência:
    a) i. o master A pretende fazer uma transacção para leitura de dois *bytes* com os valores `0x23` e `0x43` de um *target* C com o endereço inicial `0x800700c`;
    ii. o *target* não suporta modo *burst*;
    iii. o *target* introduz um estado de espera em cada fase de dados.
    b) Desenhe o diagrama temporal para esta transacção, representando a evolução de todos os sinais envolvidos (protocolo de arbitragem e de transferência). Os sinais `A/D`[31:0] e `C/BE#`[3:0] devem ser representados por valores em hexadecimal.
    c) Supondo que o *master* B obtém do árbitro `GNT#` para iniciar uma transacção de escrita de 38 *bytes* a partir do endereço `0xfedda008`, indique as alterações a introduzir ao diagrama desenhado na alínea anterior.
    d) Para melhorar o desempenho do sistema, e sabendo que sobre o *target* só são realizadas leituras com um máximo de 4 fases de dados, indique se o *target* deve ser substituído por outro que: *i)* não introduza estados de espera; ou *ii)* que permita o funcionamento em modo *burst*. Justifique.

20. [*2005/06i, 2º Teste*]
Relativamente à figura:



    a) Quantas transacções estão representadas, quais os seus tipos (leitura ou escrita) e que *master* realiza cada uma delas?

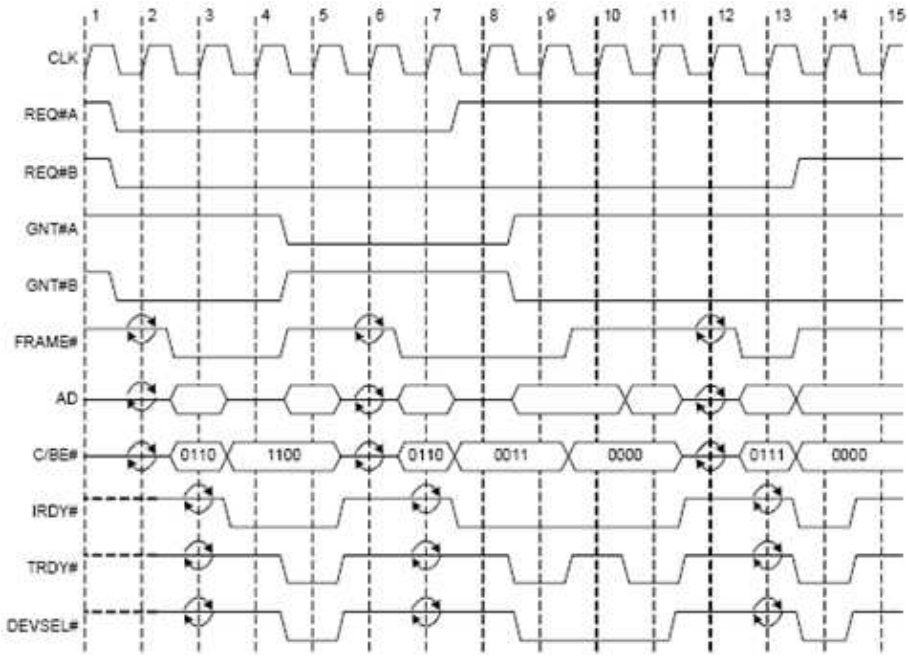
b) Indique o número de estados de espera representados, explicitando o master que os introduz.

c) Admitindo um algoritmo de arbitragem "justo" e prioridade rotativa, qual o *master* que realiza a próxima transacção após as representadas?

d) Explique as razões que levam a que o sinal de `FRAME#` fique inactivo apenas no ciclo de relógio 6 e não imediatamente no ciclo de relógio 5?

e) Justifique a existência do *turn-around* do barramento `A/D` no ciclo de relógio 4. Indique, sobre o barramento `AD`, se existem mais *turn-around*.

21. Explique o conceito de *parking* do *bus* PCI. Porque é necessário? Como é realizado?
22. Relativamente à figura:



a) Quantas transacções estão representadas e que *master* as realiza?

b) Indique o que significam os valores apresentados no barramento `CBE#`.

c) Os *turn-around* do barramento estão todos representados? Justifique.

d) Indique os ciclos nos quais existe transferência de dados. Atribua valores ao barramento `AD`.

e) Na segunda transacção, a que fase de dados pertence o estado de espera?

f) Justifique a existência do sinal de `GNT#B` a zero no ciclo de relógio 15.

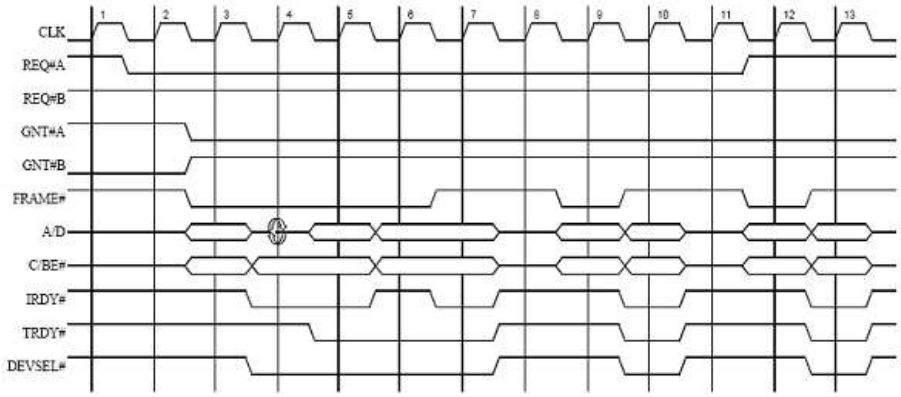
23. Relativamente ao *bus* PCI, comente a seguinte afirmação: "Um *master* com o sinal de `GNT#` a zero não tem garantia realizar a transação pedida."
24. Comente a seguinte afirmação: "Se o sinal de `FRAME#` só fosse retirado no final da última fase de dados, o desempenho do `bus` seria menor."
25. Considere um `bus` PCI a 32 bits. Indique as situações em que um *master* toma posse do `bus`, caracterizando os sinais de controlo (arbitragem e transacção) relevantes.
26. Considere um `bus` PCI a 32 bits com dois *masters*, designados por A e B, com um esquema de prioridades fixas, sendo A o mais prioritário:

    - O *master* A pretende realizar a leitura de 2 bytes de memória com os valores `0x46` e `0x56` com o endereço inicial `0x00afdacde`. Admita que o *target* introduz dois estados de espera em cada fase de dados;

- O *master* B pretende realizar duas transacções: a escrita de 4 bytes com os valores `0x46`, `0x56`, `0xfe` e `0x58` com o endereço inicial `0x00afdacde` e; a leitura de um bloco de dados de 2 bytes com os valores `0xf0` e `0x0f` no endereço `0x2f3fdccc`. Admita que o *master* introduz um estado de espera na primeira transacção.

Desenhe os diagramas temporais para estas transacções, representando a evolução de todos os sinais envolvidos (protocolo de arbitragem e de transferência). Os sinais `AD`[31:0] e `CBE#`[3:0] devem ser representados por valores em hexadecimal.

27. Relativamente à figura:



a) Quantas transacções estão representadas, quais os seus tipos (leitura ou escrita) e que *master* realiza cada uma delas?

b) Indique o número de estados de espera representados, explicitando o *master* que os introduz.

c) Admitindo um algoritmo de arbitragem "justo" e prioridade rotativa, qual o *master* que realiza a próxima transacção após as representadas?

d) Explique as razões que levam a que o sinal de `FRAME#` fique inactivo apenas no ciclo de relógio 6 e não imediatamente no ciclo de relógio 5?

e) Justifique a existência do turn-around do barramento `AD` no ciclo de relógio 4. Indique, sobre o barramento `AD`, se existem mais *turn-arounds*.

28. Relativamente à figura:

a) Quantas transacções estão representadas e quais os seus tipos (leitura ou escrita)?

b) Qual dos dois *masters*, A ou B, realiza cada uma delas?

c) Indique os números das transições de relógio em que ocorre *turn-around* no `bus AD`.

29. Se fosse possível aumentar a frequência do `bus` ISA para 33MHz, justifique a seguinte afirmação: "Uma transacção, no `bus` PCI, de um bloco de dados com dimensão de 8 bytes é realizada em menos de metade do tempo que no `bus` ISA."

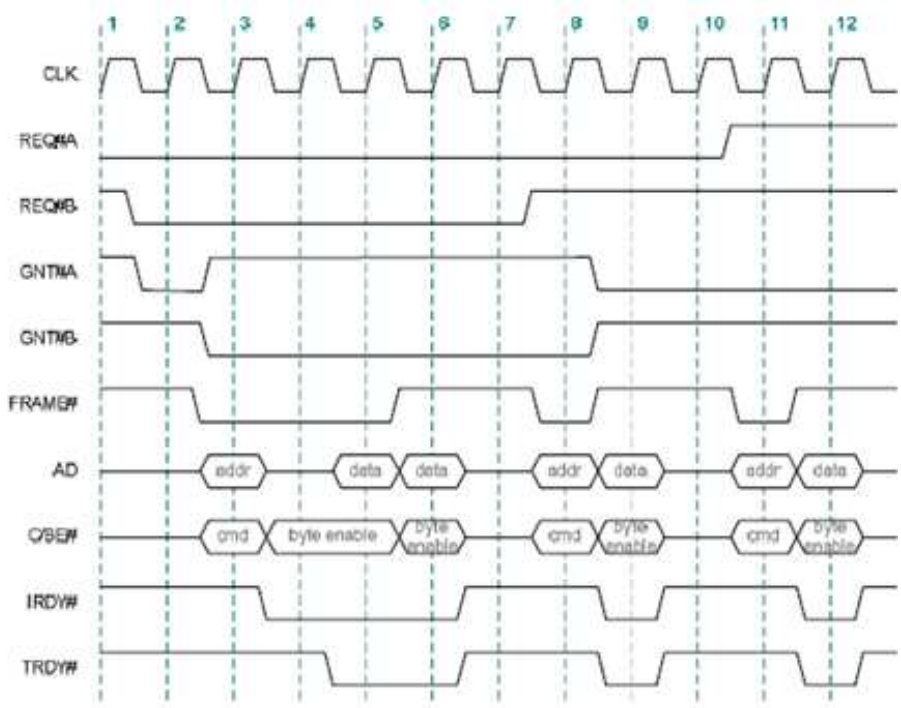
30. Considere um *target* que não suporta transacções em *burst* (não tem contador interno de endereços).

a) Explique o que acontece se um *master* tentar fazer um acesso em *burst* a este *target*.

b) Quais as consequências desta característica no desempenho das transacções?

31. Considere um `bus` PCI a 32 bits com dois *masters*, designados por A e B, com um esquema de prioridades rotativas:

- O *master* A pretende realizar a escrita de 3 bytes de memória, com os valores `0xa4`, `0xfd` e `0x00`, a partir do endereço inicial `0x00fdacee`. Admita que o *master* introduz dois estados de espera em cada fase de dados;



- O *master* B pretende realizar duas transacções: a leitura de 1 byte, com o valor `0x75`, a partir do endereço inicial `0x243fcab8`; a escrita de um bloco de dados de 2 bytes, com os valores `0xf3` e `0x3e`, no endereço `0x243fcaf0`. Admita que o *target* introduz um estado de espera na segunda transação.

Desenhe os diagramas temporais para estas transacções, representando a evolução de todos os sinais envolvidos (protocolo de arbitragem e de transferência). Os sinais `AD`[31:0] e `CBE#`[3:0] devem ser representados por valores em hexadecimal.

32. Admita um sistema com vários dispositivos *master* PCI. Indique em que condições um *master* pode iniciar uma transacção PCI.

33. Os débitos máximos (de pico) no `bus` PCI são

    - com dados a 32 bits e frequência de 33 MHz – 132 Mbytes/s;
    - com dados a 64 bits e frequência de 33 MHz – 264 Mbytes/s;
    - com dados a 32 bits e frequência de 66 MHz – 264 Mbytes/s;
    - com dados a 64 bits e frequência de 66 MHz – 528 Mbytes/s.

    a) Apresente a justificação para estes valores.

    b) Comente a afirmação "O débitos de pico do `bus` PCI nunca são atingidos".

34. Considere um `bus` PCI a 32 bits com dois *masters*, designados por A e B, com um esquema de prioridades fixas, sendo o *master* A mais prioritário:

    - O *master* A pretende realizar a leitura de um bloco de dados, com 4 bytes, com os valores `0x12`, `0x34`, `0x56` e `0x78`, localizado no endereço de memória `0x0023432a`. Admita que o *target* introduz um estado de espera em cada fase de dados.
    - O *master* B pretende realizar uma leitura de dois blocos de dados. Um, com 1 byte, com o valor `0x11`, localizado no endereço de memória 0x0a0c0bd0; o outro, com 8 bytes, com os valores `0x32`, `0xf4`, `0xff`, `0xfd`, `0x51`, `0xde`, `0x78` e `0xcc`, localizado no endereço de memória `0x08bfc20c`. Admita que o *master* introduz um estado de espera em cada fase de dados.

Considerando que o *master* B está a realizar a primeira fase de dados da primeira transacção quando o *master* A activa `REQ#`, desenhe os diagramas temporais para estas transacções.

Represente a evolução de todos os sinais envolvidos (protocolo de arbitragem e de transferência). Os sinais `AD`[31:0] e `CBE#`[3:0] devem ser representados por valores em hexadecimal ou binário.

35. Considere um `bus` PCI a 32 bits com dois *masters*, designados por A e B, com um esquema de prioridades fixas, sendo o *master* A mais prioritário:

    - O *master* A pretende realizar duas transacções. A primeira transacção é uma escrita de um bloco de dados, com 5 bytes, com os valores `0x34`, `0x12`, `0x34`, `0x56` e `0x78`, no endereço de memória `0x0023432a`. A segunda transacção é a leitura de um bloco de dados, com 3 bytes, com os valores `0x34`, `0x56` e `0xde`, localizado no endereço de memória `0x80068ff8`. Admita que o *target* seleccionado para a primeira transacção não suporta transacções em *burst*.
    - O *master* B pretende realizar a leitura de dois blocos de dados. Um, com 1 byte, com o valor `0x11`, localizado no endereço de memória `0x0a0c0bd0`; o outro, com 8 bytes, com os valores `0x32`, `0xf4`, `0xff`, `0xfd`, `0x51`, `0xde`, `0x78` e `0xcc`, localizado no endereço de memória `0x08bfc20c`. Admita que o *target* introduz um estado de espera em cada fase de dados.

    Considerando que o *master* B está a realizar a primeira fase de dados da primeira transacção quando o *master* A activa `REQ#`, desenhe os diagramas temporais para estas transacções.

    Represente a evolução de todos os sinais envolvidos (protocolo de arbitragem e de transferência). Os sinais `AD`[31:0] e `CBE#`[3:0] devem ser representados por valores em hexadecimal ou binário.

36. Considere um `bus` PCI a 32 bits com dois *masters*, designados por A e B, com um esquema de prioridades fixas, sendo o *master* A mais prioritário:

    - O *master* A pretende realizar a escrita de um bloco de 3 bytes, com os valores `0x23`, `0x35` e `0x37`, no endereço de memória `0x8023432d`. Admita que o *target* introduz um estado de espera em cada fase de dados.
    - O *master* B pretende realizar uma leitura de dois blocos de dados. Um, com 4 bytes, com os valores `0x11`, `0xff`, `0x34` e `0x33`, localizado no endereço de memória `0x0a0c0bd4`; o outro, com 6 bytes, com os valores `0x34`, `0xff`, `0xfd`, `0x5e`, `0x78` e `0xcc`, localizado no endereço de memória `0x0a0c0bd8`. Admita que o *master* introduz um estado de espera em cada fase de dados da primeira transacção.

    Represente a evolução de todos os sinais envolvidos (protocolo de arbitragem e de transferência). Os sinais `AD`[31:0] e `CBE#`[3:0] devem ser representados por valores em hexadecimal ou binário.

37. Comente a seguinte afirmação: "No `bus` PCI as linhas de interrupção `INTA#`, `INTB#`, `INTC#` e `INTD#`, limitam a existência de dispositivos que utilizam interrupções a um máximo de quatro."

38. Qual é o significado do sinal `GNT#` para um *master* PCI? Por que motivo não basta observar este sinal para iniciar uma transacção? Se o *master* observar `GNT#` activo durante um qualquer instante, isso significa que dispõe do `bus` para a transacção seguinte? Justifique.

39. Considere um `bus` PCI a 32 bits com três *masters*, designados por A e B e C, com um esquema de prioridades rotativas.

    - O *master* A pretende realizar a leitura de 6 bytes de memória, com os valores `0x4a`, `0xfe`, `0x10`, `0x32`, `0xed` e `0xa5`, a partir do endereço inicial `0x00fdaced`.
    - O *master* B pretende realizar duas transacções: a leitura de 2 bytes, com os valores `0x75` e `0x4e`, a partir do endereço inicial `0x243fcabd` e; a escrita de 2 bytes, com os valores `0x3d` e `0xd3`, a partir do endereço inicial `0x243fcdb0`. Admita que o *master* introduz um estado de espera nas fases de dados da segunda transacção.

- O *master* C pretende realizar uma leitura de 4 bytes, com os valores `0xed`, `0x45`, `0x65` e `0xaa`, a partir do endereço inicial `0x00ffff0c`. Admita que o *target* introduz um estado de espera.

Admita que a sequência de rotação da prioridade é A-B-C e que o *master* A é o primeiro a obter o `bus`.

Desenhe os diagramas temporais para estas transferências, representando a evolução de todos os sinais envolvidos (protocolo de arbitragem e de transferência). Os sinais `AD`[31:0] e `CBE#`[3:0] devem ser representados por valores em hexadecimal.

40. Considere um sistema com `bus` PCI a 32-bits com dois *masters*, designados por A e B, com um esquema de prioridades rotativas, em que o *master* A é o mais prioritário no início da seguinte sequência:

    - o *master* A pretende fazer uma transacção para leitura de dois bytes, com os valores `0x23` e `0x43` de um *target* C com o endereço inicial `0x800700c`;
    - o *target* não suporta modo *burst*;
    - o *target* introduz um estado de espera em cada fase de dados.

    a) Desenhe o diagrama temporal para esta transacção, representando a evolução de todos os sinais envolvidos (protocolo de arbitragem e de transferência). Os sinais `AD`[31:0] e `CBE#`[3:0] devem ser representados por valores em hexadecimal.

    b) Supondo que o *master* B obtém do árbitro `GNT#` para iniciar uma transacção de escrita de 38 bytes a partir do endereço `0xfedda008`, indique as alterações a introduzir ao diagrama desenhado na alínea anterior.

    c) Para melhorar o desempenho do sistema, e sabendo que sobre o *target* só são realizadas leituras com um máximo de 4 fases de dados, indique se o *target* deve ser substituído por outro que: *i)* não introduza estados de espera; ou *ii)* que permita o funcionamento em modo *burst*. Justifique.

41. Considere uma arquitectura PC com um `bus` PCI e um `bus` ISA ligado a partir dele. Se existirem dois periféricos com o mesmo endereço, um ligado ao `bus` PCI outro ao `bus` ISA, qual deles é acedido? Justifique.

42. Comente a seguinte afirmação: "O endereço colocado no barramento `turn-around`, numa transacção PCI, é múltiplo de 4 para as transacções de memória (leitura ou escrita) e pode ter qualquer valor as transacções de I/O."

43. Relativamente ao mecanismo de atribuição do `bus` PCI aos diversos masters, descreva:

    a) As condições para um *master* iniciar uma transacção;

    b) A necessidade de existência do árbitro de `bus`;

    c) O funcionamento dos sinais `REQ#` e `GNT#`.

44. Relativamente à figura:

    a) Quantas transacções estão representadas, quais os seus tipos (leitura ou escrita) e que *master* realiza cada uma delas?

    b) Porque é que o sinal de `FRAME#` sobe no ciclo de relógio 9 e não no cico de relógio 8?

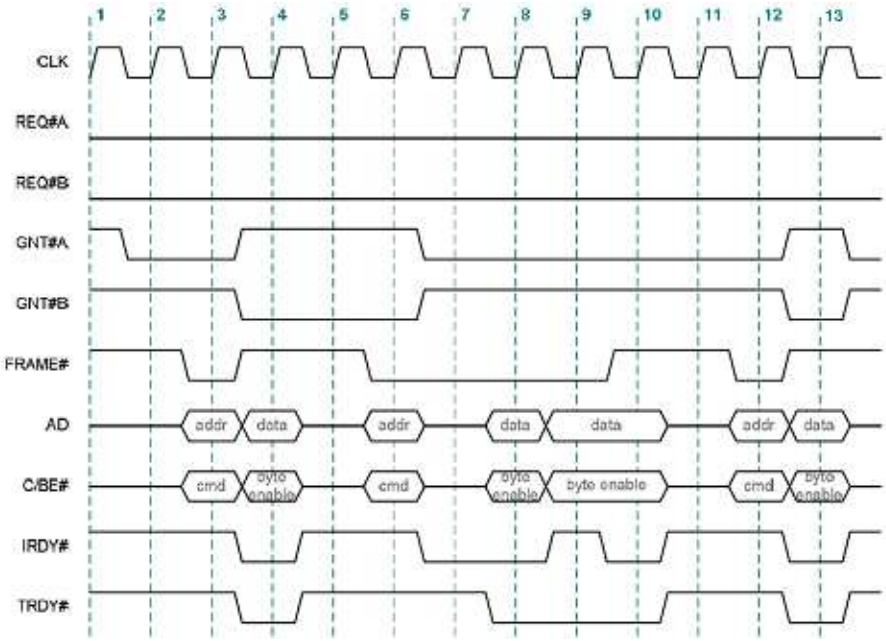
    c) Qual o *master* que realiza a próxima transacção, após as representadas?

    d) Indique os números das transições de relógio em que ocorre *turn-around* nos sinais de `FRAME#`, `IRDY#` e `TRDY#`.

45. Considere um sistema com `bus` PCI a 32 bits com dois *masters*, designados por A e B, com um esquema de prioridades fixas em que B é o mais prioritário.

    O *master* A pretende fazer duas transacções:

    - leitura de um byte, com o valor `0x7a`, do endereço `0x00800101`, ocorrendo um estado de espera provocado pelo *target*;



- escrita de um byte, com o valor `0xe2`, para o endereço `0x00800102`, sem ocorrência de estados de espera.

O *master* B pede acesso ao `bus` durante a primeira fase de dados da transacção para fazer uma transacção:

- leitura de 6 bytes, com os valores `0x75`, `0x86`, `0x97`, `0xa8`, `0xb9` e `0xca`, a partir do endereço inicial `0x1234dcba`, ocorrendo um estado de espera, introduzido pelo *target*, na segunda fase de dados.

Desenhe os diagramas temporais para estas transacções, representando a evolução de todos os sinais envolvidos (protocolo de arbitragem e de transferência). Os sinais `AD`[31:0] e `C/BE#`[3:0] devem ser representados por valores em hexadecimal.

46. Comente as seguintes afirmações:

    a) "Num `bus` PCI *multi-master* com um critério de prioridade fixa, o *master* de menor prioridade pode nunca conseguir realizar qualquer transacção".

    b) "No `bus` PCI é seleccionada automaticamente a largura do `bus` de dados de acordo com as características do *master*, *target* e do próprio *bus*".

47. Comente as seguintes afirmações:

    a) "Num `bus` PCI *multi-master*, um *master* não pode ficar em posse do `bus` eternamente, independentemente da sua prioridade."

    b) "O `bus` PCI, pelo facto de suportar transferência em *burst*, não permite a detecção e correcção de erros."

48. Por que motivo o sinal `FRAME#` é normalmente desactivado no início da última fase de dados da transacção? Em que circunstância não é desactivado neste instante? Porquê?

49. Comente a seguinte afirmação: "A concepção do `bus` PCI reflecte a existência de cache".

50. Considere um `bus` PCI a 32 bits com dois *masters*, designados por A e B, com um esquema de prioridades fixas, sendo o *master* A mais prioritário:

    - O *master* A pretende realizar a leitura de um bloco de dados de 8 bytes, com os valores `0x12`, `0x34`, `0xfd`, `0x34`, `0xea`, `0x65`, `0x56` e `0x78`, localizado a partir do endereço de memória `0x00101018`. Admita que o *target* introduz um estado de espera em cada fase de dados.

- O *master* B pretende realizar duas transacções. A primeira transacção é de leitura de um bloco de dados de 2 bytes com os valores `0x12` e `0x34`, localizados a partir do endereço de memória `0x00000bde`; a outra transacção é de escrita de um bloco de dados de 4 bytes, com os valores `0x32`, `0xff`, `0xfd` e `0xde`, localizado a partir do endereço de memória `0x0000bdc`. Admita que o *master* introduz um estado de espera nas fases de dados da primeira.

Considerando que o *master* A só activa `REQ#` quando o *master* B já está em posse do `bus`, desenhe os diagramas temporais para estas transacções. Represente a evolução de todos os sinais envolvidos (protocolo de arbitragem e de transferência). Os sinais `AD`[31:0] e `CBE#`[3:0] devem ser representados por valores em hexadecimal ou binário.

51. Considere um sistema com `bus` PCI a 32 bits com dois *masters*, designados por A e B, com um esquema de prioridades fixas, em que A é o mais prioritário.
    O *master* B pretende fazer duas transacções:
    - Leitura de três bytes, com os valores `0x7a`, `0xa7` e `0xfc`, a partir do endereço inicial `0x00800101`. Admita que o *target* não responde (assuma 3 impulsos de relógio de *timeout*).
    - Escrita de quatro bytes, com os valores `0xe2`, `0xfd`, `0xcb` e `0x00`, a partir do endereço inicial `0x0080100c`. Admita que o *master* introduz um estado de espera na última fase de dados.

    Após o *master* B ter tomado posse do `bus` para realizar a primeira transacção, o *master* A pede acesso para realizar um transacção:
    - Leitura de 5 bytes, com os valores `0x56`, `0x46`, `0xa7`, `0x8f` e `0xde`, a partir do endereço inicial `0x1000daaa`. Admita que o *target* introduz um estado de espera em cada fase de dados.

    Desenhe os diagramas temporais para estas transacções, representando a evolução de todos os sinais envolvidos (protocolo de arbitragem e de transferência). Os sinais `AD`[31:0] e `CBE#`[3:0] devem ser representados por valores em hexadecimal.
52. Considere um sistema com `bus` PCI a 32 bits com dois *masters*, designados por A e B, com um esquema de prioridades rotativas, em que o *master* A é o mais prioritário no início da sequência indicada.
    O *master* A pretende fazer duas transacções:
    - Leitura de dois bytes, com os valores `0xa9` e `0x9c`, a partir do endereço inicial `0x09f101cc`. Admita que o *master* introduz um estado de espera.
    - Escrita de três bytes, com os valores `0xad`, `0xdd` e `0xc4`, a partir do endereço inicial `0x031010dd`.

    O *master* B pretende fazer uma transacção:
    - Leitura de cinco bytes, com os valores `0x56`, `0x86`, `0xa2`, `0xc9` e `0xde`, a partir do endereço inicial `0xfdaff000`.

    Desenhe os diagramas temporais para estas transacções, representando a evolução de todos os sinais envolvidos (protocolo de arbitragem e de transferência). Os sinais `AD`[31:0] e `CBE#`[3:0] devem ser representados por valores em hexadecimal.
53. Considere um sistema com `bus` PCI a 32 bits com dois *masters*, designados por A e B, com um esquema de prioridades fixas, em que o *master* A é o mais prioritário. Os *masters*, para realizarem as suas transacções, pedem o `bus` em simultâneo.
    O *master* A pretende fazer duas transacções:
    - Leitura de quatro bytes, com os valores `0xda`, `0x93`, `0x57` e `0xfc`, a partir do endereço inicial `0x08fe010c`. Admita que o *target* introduz um estado de espera.
    - Escrita de dois bytes, com os valores `0xdd` e `0xff`, a partir do endereço inicial `0x08fe0110`. Admita que o *master* introduz um estado de espera na última fase de dados.

O *master* B pretende fazer uma transacção:

- Leitura de seis bytes, com os valores `0x48`, `0x63`, `0x73`, `0x7a`, `0xae` e `0xfe`, a partir do endereço inicial `0x8000a008`.

Desenhe os diagramas temporais para estas transacções, representando a evolução de todos os sinais envolvidos (protocolo de arbitragem e de transferência). Os sinais `AD`[31:0] e `CBE#`[3:0] devem ser representados por valores em hexadecimal.