



INSTITUTO SUPERIOR TÉCNICO

DEPARTAMENTO DE ENGENHARIA INFORMÁTICA

ORGANIZAÇÃO DE COMPUTADORES

LEIC

Conjunto de Exercícios I
Introdução à Arquitectura de Computadores

Versão 2.0

2023/2024

Representação em binário e hexadecimal

Operações aritméticas e lógicas

Em todas as bases de numeração segue-se a convenção de representação comum nos números em decimal: dígito mais significativo à esquerda, dígito menos significativo à direita. Nestes exercícios a base de numeração é identificada pelo prefixo do número (0x para números hexadecimais e 0b para números binários).

Exercício 1

Converta a representação dos seguintes números representados em binário (base 2) para hexadecimal (base 16), ou vice-versa.

- (a) 0b 1010 1110
- (b) 0b 0101 0000 0111
- (c) 0x 7FEA

Exercício 2

Realize as operações indicadas apresentando o resultado na mesma base dos operandos, ou na base indicada.

Considere que os números representam inteiros sem sinal.

$X \ll N$ desloca o número X para esquerda por N bits.

$X \gg N$ desloca o número X para direita por N bits.

$A \& B$ representa o *e bit a bit* (bitwise and) entre A e B . $A | B$ representa o *ou bit a bit* (bitwise or) entre A e B . $A \wedge B$ representa o *ou exclusivo* (bitwise xor) entre A e B .

- (a) 0x 707F + 0x D
- (b) 0x E003 – 0x 5
- (c) 0b 1010 1110 + 0b 0010
- (d) 0b 0111 \times 0x 2 = 0x _____
- (e) 0b 0111 \ll 1
- (f) 0x 75 \gg 2
- (g) 0x 75 & 0x 0F
- (h) 0x 75 \wedge 0x 55

Exercício 3

Uma palavra de 32 bits está estruturada nos seguintes campos (Bit 31 - maior peso, bit 0 - menor peso):

index 1 – bits 31 a 19,

index 2 – bits 18 a 8,

index 3 – bits 7 a 0.

- (a) index 1, index 2 e index 3 são usados para indexar tabelas. Quantas entradas tem cada uma destas tabelas?

- (b) Indique os valores em hexadecimal dos três campos correspondentes à palavra 0x 803FA5A5.
- (c) Usando operações aritméticas ou lógicas elementares - &, |, ^, << e >> - extraia da palavra o campo *index 2* de modo a aceder directamente à tabela correspondente.
- (Isto é, obter o valor *n* para aceder ao vector *tab2*: `var = tab2[n]`.)

Memória e Caches

Exercício 4

Considere um computador em que o BUS que liga o CPU à memória principal tem uma largura de 32 bits. Um acesso à memória principal tem uma latência de 90ns.

Neste computador executa-se um programa que gera 10.000 acessos sequenciais à memória.

- (a) Qual é o tempo total de execução deste programa? (Considere que o tempo de execução resulta exclusivamente dos acessos ao sistema de memória.)
- (b) Qual é a largura de banda da interface do processador com a memória (em MB/s, onde MB é 10^6 B*)?

Exercício 5

Uma nova versão do computador (do exercício 4) inclui uma cache entre o processador e a memória. A cache é uma memória com capacidade reduzida, mas rápida, que armazena as palavras mais recentemente acedidas pelo processador e verifica todos os acessos deste tentando satisfazer os pedidos de modo a evitar acessos, mais lentos, à memória.

- Em caso de *hit*, a cache realiza o acesso internamente e consegue satisfazer o pedido sem necessidade de aceder à memória;
- Em caso de *miss*, a cache lança o acesso à memória, carrega internamente a palavra em falta e conclui o acesso do processador.

Excluindo a diferença entre os tempos de acesso, as duas situações – *hit* ou *miss* – são transparentes para o processador e, portanto, para os programas que nele se executam:

- $t_{hit} = 10$ ns (tempo de acesso à cache);
- $t_{miss} = 100$ ns (tempo total de acesso à cache, incluindo o acesso desta à memória).

Dos 10.000 acessos gerados pelo processador ao executar o programa (do exercício anterior) 8.000 são satisfeitos imediatamente pela cache sem necessidade de aceder à memória.

- (a) Quais são o *hit rate* e o *miss rate* na cache?
- (b) Qual é a penalização da falta na cache (*miss penalty*)?

*Nota histórica: O valor exato de 1 MB é um tópico algo contencioso. No contexto das unidades SI toma o valor de 10^6 B, pois as unidades SI estão radicadas na base 10. Contudo, é muito mais natural do ponto de vista de um computador considerar um MB 2^{20} . É importante notar que $2^{20} > 10^6$ o que levou os fabricantes de meios de armazenamento a preferir utilizar as medidas de base 10 (podendo publicitar valores superiores). Para reduzir a ambiguidade, introduziu-se o *Mibibyte*, MiB, que tem o valor 2^{20} B. Contudo, como o nome soa pouco natural, por vezes utiliza-se MB com o intuito de se referir a 2^{20} B. Resultado: é preciso ter atenção e pensar no valor que parece natural para uma dada situação.

Referência: <https://xkcd.com/927/>

- (c) Qual é o tempo de acesso médio do processador à memória?
- (d) Qual é o tempo total de execução do programa neste computador? (Considere que o tempo de execução resulta exclusivamente dos acessos ao sistema de memória.)
- (e) Compara o tempo de execução do programa numa arquitectura com e sem cache. Comenta o tradeoff num caso genérico.

Exercício 6

Considere um sistema com um processador e memória de 8 bits. Os endereços e as palavras do processador e da memória têm 8 bits. O sistema possui uma cache, só para código, entre o processador e a memória com capacidade para armazenar 4 entradas, com as características seguintes:

- Cada entrada da cache pode armazenar uma qualquer palavra da memória;
 - A entrada da cache tem campos para armazenar
 - Estado (Válida / Inválida);
 - Tempo - regista o momento (o tempo) em que a entrada é acedida; é avaliado quando é necessário substituir uma entrada para carregar uma nova palavra;
 - Palavra - a palavra carregada da memória na sequência de um *miss*;
 - Endereço - endereço de memória da palavra carregada na entrada.
 - Inicialmente a cache está vazia. (Todas as entradas estão inválidas.)
 - Todos os acessos do processador à memória são verificados pela cache;
 - Os acessos para leitura ou escrita de dados (R – *read*, ou W – *write*) são ignorados pela cache e seguem directamente para a memória;
 - Os acessos para leitura de instruções (F – *opcode fetch*) são interceptados pela cache que verifica se consegue satisfazer o pedido internamente;
 - * se sim – cache *hit* – realiza o acesso com informação na cache, sem aceder à memória;
 - * se não – cache *miss* – lança, transparentemente ao processador, um acesso à memória para carregar internamente a palavra em falta, enviá-la ao processador e concluir o acesso;
 - O carregamento de uma nova entrada é feito na primeira posição livre da cache ou, se a cache estiver cheia, substitui a posição que não é acedida há mais tempo (*least recently used*, abreviadamente LRU).
- (a) Preenche a tabela seguinte que representa os primeiros 8 acessos do processador ao sistema de memória decorrentes da execução de um programa, com o estado da cache **antes** do acesso. (Os “endereço”s, as “palavras” e as “tags” são representados em hexadecimal.)

<i>t</i>	Acesso à memória			Cache					
	Tipo	Endereço [h]	Palavra [h]	<i>Hit / Miss</i>	Entrada [0 .. 3]	Estado [V, I]	Tempo [t]	Palavra [0x]	Tag [0x]
1	F	80	A5						
2	F	81	ED						
3	F	82	C7						
4	F	81	ED						
5	R	00	?						
6	F	83	A1						
7	W	00	3F						
8	F	8F	3F						

- (b) Qual é a percentagem de acessos à memória por tipo de acesso?
- (c) Qual é o *hit rate* no acesso à cache ao executar este programa?
- (d) Qual é o tempo de execução deste troço de programa? (Considere que o tempo de execução resulta exclusivamente dos acessos ao sistema de memória: $t_{hit} = 10 \text{ ns}$, $t_{miss} = 100 \text{ ns}$.)

Conjunto de Exercícios I

Introdução à Arquitectura de Computadores

Soluções

Exercício 1

A cada conjunto de 4 bits numa palavra em binário corresponde um dígito hexadecimal (0, 1, 2, ... , 9, A, B, ... , E, F). Os quartetos são formados da direita para a esquerda.

(a) $0b\ 1010\ 1110 = 0x\ AE$.

(b) $0b\ 0101\ 0000\ 0111 = 0x\ 507$.

(c) $0x\ 7FEA = 0b\ 0111\ 1111\ 1110\ 1010$.

Exercício 2

Por simplicidade realizam-se todas as operações em binário, convertendo, se necessário, os operandos.

(a) $0x\ 707F + 0x\ D$

$$\begin{array}{rcl} 0b\ 0111\ 0000\ 0111\ 1111 & = & 0x\ 707F \\ + \quad \quad \quad 0b\ 1101 & = & 0x\ D \\ \hline 0b\ 0111\ 0000\ 1000\ 1100 & = & 0x\ 708C \end{array}$$

(b) $0x\ E003 - 0x\ 5$

$$\begin{array}{rcl} 0b\ 1110\ 0000\ 0000\ 0011 & = & 0x\ E003 \\ - \quad \quad \quad 0b\ 0101 & = & 0x\ 5 \\ \hline 0b\ 1101\ 1111\ 1111\ 1110 & = & 0x\ DFFE \end{array}$$

(c) $0b\ 1010\ 1110 + 0b\ 0010$

$$\begin{array}{rcl} 0b\ 1010\ 1110 \\ + \quad \quad 0b\ 10 \\ \hline 0b\ 1011\ 0000 \end{array}$$

(d) $0b\ 0111 \times 0x\ 2$

$$\begin{array}{rcl} 0b\ 0111 & = & 0x\ 7 \\ \times \quad 0b\ 0010 & = & 0x\ 2 \\ \hline 0b\ 0000 \\ 0b\ 0\ 111 \\ \hline 0b\ 0\ 1110 & = & 0x\ E \end{array}$$

$$0x\ 7 = 7, 0x\ 2 = 2, 7 \times 2 = 14.$$

(e) $0b\ 0111 \ll 1$

$0b\ 0111 \ll 1 = 0\ 1110\ b = 0x\ E = 14.$

Cada posição deslocada para a esquerda é uma multiplicação por 2.

(f) $0x\ 75 \gg 2$

$0x\ 75 \gg 2 = 0b\ 0111\ 0101 \gg 2 = 0001\ 1101\ b = 0x\ 1D.$

Cada posição deslocada para a direita é uma divisão inteira por 2.

(g) $0x\ 75 \& 0x\ 0F$

$$\begin{array}{rcl} 0b\ 0111\ 0101 & = & 0x\ 75 \\ \& 0b\ 0000\ 1111 & = & 0x\ 0F \\ \hline 0b\ 0000\ 0101 & = & 0x\ 05 \end{array}$$

(h) $0x\ 75 \wedge 0x\ 55$

$$\begin{array}{rcl} 0b\ 0111\ 0101 & = & 0x\ 75 \\ \wedge 0b\ 0101\ 0101 & = & 0x\ 55 \\ \hline 0b\ 0010\ 0000 & = & 0x\ 20 \end{array}$$

Exercício 3

(a) Cada campo *index* tem

index 3 – 8 bits, indexa $2^8 = 256$ entradas,

index 2 – 11 bits, indexa $2^{11} = 2.048 = 2\ K$ entradas,

index 1 – 13 bits, indexa $2^{13} = 2^3\ K = 8\ K$ entradas.

($2^{10} = 1.024 = 1\ K.$)

(b) Representa-se a palavra de 32 bits em binário, extraem-se os bits correspondentes a cada campo e converte-se cada campo em hexadecimal.

$0x\ 803FA5A5 = 0b\ 1000\ 0000\ 0011\ 1111\ 1010\ 0101\ 1010\ 0101$

index 1 = $0b\ 0001\ 0000\ 0000\ 0111 = 0x\ 1007$

index 2 = $0b\ 0111\ 1010\ 0101 = 0x\ 7A5$

index 3 = $0b\ 1010\ 0101 = 0x\ A5$

(c) $n = (\text{palavra} \gg 8) \& 0x\ 0000\ 07FF$

Exercício 4

(a) tempo de execução = número de acessos \times tempo de acesso = $10000 \times 90\ ns = 900\ \mu s = 0,9\ ms.$

(b) Em cada 90 ns o processador transfere uma palavra de 32 bits com a memória.

$32\ bits = 4\ bytes \equiv 4\ B$

Em cada segundo ocorrem $(1\ s)/(90\ ns) = 1,11 \times 10^7$ transferências.

A largura de banda da ligação é $1,11 \times 10^7 \times 4\ B = 4,44 \times 10^7\ B/s = 44,4\ MB/s.$

Exercício 5

(a) *hit rate* = acessos satisfeitos pela cache / total de acessos = $8.000 / 10.000 = 80\%.$

miss rate = $1 - \text{hit rate} = 20\%.$

- (b) d) A penalização da falta é o agravamento no tempo de acesso ao falhar a cache
 $miss\ penalty = t_{miss} - t_{hit} = 100\ ns - 10\ ns = 90\ ns$.
- (c) tempo de acesso (médio) = $hit\ rate \times t_{hit} + miss\ rate \times t_{miss} =$
 $= 80\% \times 10\ ns + 20\% \times 100\ ns = 8\ ns + 20\ ns = 28\ ns$.
- (d) tempo de execução = número de acessos \times tempo de acesso =
 $10.000\ acessos \times 28\ ns = 280.000\ ns = 280\ \mu s = 0,28\ ms$.
- (e) tempo de execução sem cache = $900\ \mu s$,
tempo de execução com cache = $280\ \mu s$.

O sistema com cache é mais rápido que o sistema sem cache $900 / 280 = 3,2$ vezes. Mas esta relação varia com o padrão de acessos à memória que, entre outros factores, depende da aplicação que está em execução. Esta relação costuma designar-se *speedup*: $S = 3,2$. Notar que este speedup depende do programa a ser executado e da localidade dos acessos. Em particular, notar que a introdução de uma cache torna o acesso à memória principal mais lento. Contudo na *esmagadora* maioria dos casos vale a pena.

Exercício 6

- (a) Considera-se como marcador de tempo o número do acesso na sequência - t - que é actualizado sempre que houver um acesso à entrada da cache. Cada linha mostra o conteúdo da entrada da cache após o acesso, isto é, devidamente actualizada.

t	Acesso à memória			Cache					
	Tipo	Endereço [h]	Palavra [h]	Hit / Miss	Entrada [0 .. 3]	Estado [V, I]	Tempo [t]	Palavra [h]	Tag [h]
1	F	80	A5	M		I			
2	F	81	ED	M		I			
3	F	82	C7	M		I			
4	F	81	ED	H	1	V	2	ED	81
5	R	00	?	-					
6	F	83	A1	M		I			
7	W	00	3F	-					
8	F	8F	3F	M	0	V	1	A5	80

- (b) Em 8 acessos do processador há 6 *fetches*, uma leitura e uma escrita de dados:
 $6/8 = 75\%$ de *fetches* de instruções,
 $1/8 = 12,5\%$ de leituras de dados e
 $12,5\%$ de escritas de dados.
Há $75\% + 12,5\% = 87,5\%$ de acessos de leitura.
- (c) Em 6 acessos a código há 1 *hit*: $hit\ rate = 1/6 \approx 17\%$, $miss\ rate = 5/6 \approx 83\%$.
- (d) $T = hits \times t_{hit} + misses \times t_{miss} + acessos\ a\ dados \times t_{mem}$
 $T = 1 \times 10\ ns + 5 \times 100\ ns + 2 \times t_{mem}$
Nos acessos a dados o processador vai directamente à memória. Este acesso será semelhante ao acesso que a cache realiza após a detecção de um miss, logo

$$t_{\text{mem}} = t_{\text{miss}} - t_{\text{hit}} = 100 \text{ ns} - 10 \text{ ns} = 90 \text{ ns}.$$

$$T = 1 \times 10 \text{ ns} + 5 \times 100 \text{ ns} + 2 \times 90 \text{ ns} = 690 \text{ ns}.$$