INSTITUTO SUPERIOR TÉCNICO

DEPARTAMENTO DE ENGENHARIA INFORMÁTICA

ORGANIZAÇÃO DE COMPUTADORES

LEIC

Conjunto de Exercícios I
Introdução à Arquitectura de Computadores

Versão 2.0

2019/2020

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 sufixo do número. Os números em hexadecimal também são frequentemente representados com o prefixo ‘”0x” (0x7FEA $\equiv$ 7FEAh).

Exercício 1

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

(a) 1010 1110b

(b) 101 0000 0111b

(c) 7FEAh

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:
shift_left (N, P) desloca P bits para a esquerda o número N, com P representado em decimal;
shift_right (N, P) desloca P bits para a direita o número N, com P representado em decimal;
and (N, P) faz o E LÓGICO de N com P, bit a bit;
xor (N, P) faz o OU EXCLUSIVO de N com P, bit a bit.

(a) 707Fh + Dh

(b) E003h – 5h

(c) 1010 1110b + 10b

(d) 0111b $\times$ 2 h = _____ h

(e) shift_left (0111b, 1)

(f) shift_right (75h, 2)

(g) and (75h, 0Fh)

(h) xor (75h, 55h)

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 803FA5A5h.

(c) Usando operações aritméticas ou lógicas elementares - and, or, xor, shift_left, shift_right - 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

Um computador é constituído por um processador de 32 bits ligado directamente a uma memória, também de 32 bits, com um tempo de acesso de 90 ns.
Neste computador executa-se um programa que gera 10.000 acessos 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)?

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*, após a verificação, 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*)?

(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) Estritamente com base nos tempos de execução do programa compare os sistemas sem e com cache.

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 proessador 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) Preencha a tabela seguinte que representa os primeiros 8 acessos do processador ao sistema de memória decorrentes da execução de um programa. (Os "endereço"s e as "palavras" são representados em hexadecimal.)

	Acesso à memória			**Cache**					
t	Tipo	Endereço [h]	Palavra [h]	*Hit* / *Miss*	Entrada [0 .. 3]	Estado [V, I]	Tempo [*t*]	Palavra [h]	Endereço [h]
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 ns, t_miss = 100 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) 1010 1110 b = AE h.

(b) 0101 0000 0111 b = 507 h.

(c) 7FEA h = 0111 1111 1110 1010 b.

Exercício 2

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

(a) 707Fh + Dh

	0111 0000 0111 1111 b	=	707F h
+	1101 b	=	D h
	0111 0000 1000 1100 b	=	708C h

(b) E003h – 5h

	1110 0000 0000 0011 b	=	E003 h
-	0101 b	=	5 h
	1101 1111 1111 1110 b	=	DFFE h

(c) 1010 1110b + 10b

	1010 1110 b
+	10 b
	1011 0000 b

(d) 0111b × 2 h

	0111 b	=	7 h
×	10 b	=	2 h
	0000		
	0 111		
	0 1110 b	=	E h

7 h = 7 d (d, decimal, base 10), 2 h = 2 d, 7 × 2 = 14.

(e) shift_left (0111b, 1)

shift_left(0111 b, 1) = 0 1110 b = E h = 14 d.
Cada posição deslocada para a esquerda é uma multiplicação por 2.

(f) shift_right (75h, 2)

shift_right(75 h, 2) = shift_right(0111 0101 b, 2) = 0001 1101~~01~~ b = 1D h.
Cada posição deslocada para a direita é uma divisão inteira por 2.

(g) and (75h, 0Fh)

	0111 0101 b	=	75 h
and	0000 1111 b	=	0F h
	0000 0101 b	=	05 h

(h) xor (75h, 55h)

	0111 0101 b	=	75 h
xor	0101 0101 b	=	55 h
	0010 0000 b	=	20 h

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.

803FA5A5 h = 1000 0000 0011 1111 1010 0101 1010 0101 b
index 1 = 1 0000 0000 0111 b = 1007 h
index 2 = 111 1010 0101 b = 7A5 h
index 3 = 1010 0101 b = A5 h

(c) n = and[shift_right(palavra, 8), 0000 07FF h]

Exercício 4

(a) tempo de execução = número de acessos × tempo de acesso = 10000 × 90 ns = 900 μs = 0,9 ms.

(b) Em cada 90 ns o processador transfere uma palavra de 32 bits com a memória.
32 bits = 4 bytes ≡ 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 - *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* × t_hit + miss rate × t_miss =
= 80% × 10 ns + 20% × 100 ns = 8 ns + 20 ns = 28 ns.

(d) tempo de execução = número de acessos × tempo de acesso =
10.000 acessos × 28 ns = 280.000 ns = 280 μs = 0,28 ms.

(e) tempo de execução sem cache = 900 μs ,
tempo de execução com cache = 280 μs .

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 .

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.

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

(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 ≈ 17% , *miss rate* = 5/6 ≈ 83% .

(d) T = *hits* × t_hit + *misses* × t_miss + acessos a dados × t_mem
T = 1 × 10 ns + 5 × 100 ns + 2 × 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_mem = t_miss – t_hit = 100 ns – 10 ns = 90 ns.
T = 1 × 10 ns + 5 × 100 ns + 2 × 90 ns = 690 ns.