10801235
陳昭維
電機23

1. [section 5.3]
For a direct-mapped cache design with a 12-bit memory address, the following bit fields of the address are used to access the cache.

Tag	Index	Block offset	Byte offset
11- 9	8 - 5	4-2	1-0

1. (5 points) What is the cache block size (in B)?
2. (5 points) How many blocks does the cache have?
3. (5 points) What is the total cache size?
4. (10 points) What is the actual SRAM memory size used to implement the cache? (Please consider tag bits, dirty bit, valid bit).

①

Block size = $2^{(4-2+1)}$ words = 2^3 words = 32 B

②

The block number is $2^{(\text{Index bit})} = 2^{(8-5+1)} = 2^4 = 16$ blocks

③

Total cache size

$= 2^4 \times (2^3 \times 32) = 2^4 \times (2^8) = 4096$ bits

data size

$= 512$ B

④

If considering valid bit field, tag bit field, dirty bit field

$= 2^4 \times (2^3 \times 32 + (12-4-5) + 1 + 1)$

$= 2^4 \times (2^8 + 5)$

$= 4,176$ bits

$= 522$B

5. (33 points) Please mark the following memory access references as "hit" or "miss" in the above cache.

0xf48
0xf58
0xdc4
0x44c
0xdb4
0xdac
0xee4
0xf30
0xf40
0x84c
0x1cd

1. [section 5.3]
For a direct-mapped cache design with a 12-bit memory address,
the following bit fields of the address are used to access the cache.

Tag	Index	Block offset	Byte offset
11- 9	8 - 5	4-2	1-0

1. (5 points) What is the cache block size (in B)?
2. (5 points) How many blocks does the cache have?
3. (5 points) What is the total cache size?
4. (10 points) What is the actual SRAM memory size used to implement the cache? (Please consider tag bits, dirty bit, valid bit).

Byte address	Binary Address	tag	index	offset	hit/miss	Byte Replaced
0xf48	1111 0100 1000	0x7	0x0a	0x08	M	
0xf58	1111 0101 1000	0x7	0x0a	0x18	H	
0xdc4	1101 1100 0100	0x6	0x0e	0x04	M	
0x44c	0100 0100 1100	0x2	0x02	0x0c	M	
0xdb4	1101 1011 0100	0x6	0x0d	0x14	M	
0xdac	1101 1010 1100	0x6	0x0d	0x0c	H	
0xee4	1110 1110 0100	0x7	0x07	0x04	M	
0xf30	1111 0011 0000	0x7	0x09	0x10	M	
0xf40	1111 0100 0000	0x7	0x0a	0x00	H	
0x84c	1000 0100 1100	0x4	0x02	0x0c	M	replace
0x1cd	0001 1100 1101	0x0	0x0e	0x0d	M	replace

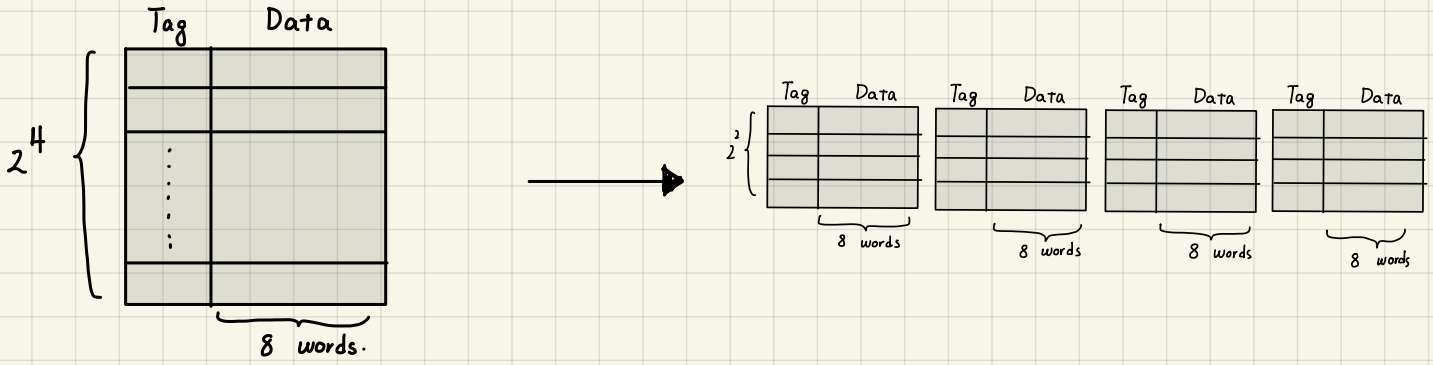
6. (22 points) Please also mark the above memory access references with "replace" in the above cache.

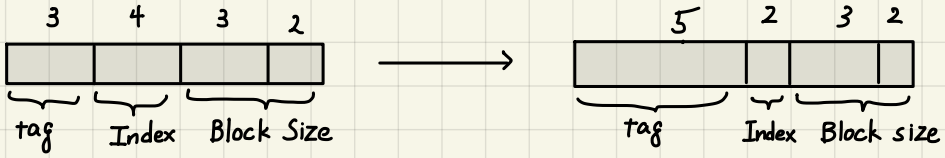
Now we would like to change the cache into a 4-way set associative design with the same total size.

7. (10 points) What is the Index range for the new cache (address bit positions)?

8. (10 points) What is the SRAM memory size used to implement the new cache?
In this cache, we also need to add bits for pseudo-LRU replacement policy for each set.

⑥





⑦ Index range changes from 0 ~ 15 to 0 ~ 3

⑧ Psuedo LRU uses 4 - 1 = 3 bits for 4-way associate.

$$2^2 \times (4 \times (\underbrace{2^3 \times 32}_{\text{data}} + \underbrace{5}_{\text{tag}} + \underbrace{1}_{\text{dirty \& valid bit}} + 1) + \underbrace{3}_{\text{psuedo LRU bit}})$$

$$= 2^2 \times (4 \times (263) + 3)$$

$$= 4{,}220 \text{ bits.}$$

9. (33 points) Please mark the following memory access references as "hit" or "miss" in the new cache. The access sequence is the same as above problem 5.

```
0xf48
0xf58
0xdc4
0x44c
0xdb4
0xdac
0xee4
0xf30
0xf40
0x84c
0x1cd
```

10. (22 points) Please also mark the above memory access references with "replace" in the 4-way cache. Assume we use true LRU in the replacement policy.

Byte address	Binary Address	tag	index	offset	hit/miss	Byte Replaced
0xf48	1111 0100 1000	0x1e	0x2	0x08	M	
0xf58	1111 0101 1000	0x1e	0x2	0x18	H	
0xdc4	1101 1100 0100	0x1b	0x2	0x04	M	
0x44c	0100 0100 1100	0x08	0x2	0x0c	M	
0xdb4	1101 1011 0100	0x1b	0x1	0x14	M	
0xdac	1101 1010 1100	0x1b	0x1	0x0c	H	
0xee4	1110 1110 0100	0x1d	0x3	0x04	M	
0xf30	1111 0011 0000	0x1e	0x1	0x10	M	
0xf40	1111 0100 0000	0x1e	0x2	0x00	H	
0x84c	1000 0100 1100	0x10	0x2	0x0c	M	
0x1cd	0001 1100 1101	0x03	0x2	0x0d	M	replace

2. (25 points) [section 5.4]

Consider the following processor with a few possible cache hierarchy choice (L2 or L3 designs):
The base CPI without memory stalls is 1. The main memory access delay is 100 cycles.

Level	access cycle and miss rate	
L1	L1 hit latency	1
L1	L1 miss rate	3%
L2 (option 1: direct mapped)	L2 access cycle	12
L2 (option 1: direct mapped)	L2 miss rate	2.5%
L2 (option 2: 8-way set associative)	L2 access cycle	14
L2 (option 2: 8-way set associative)	L2 miss rate	2.0%
L3	L3 access cycle	40
L3	L3 miss rate	1.5%

1. (5 points) Calculate the AMAT using only a 1-level cache.
2. (5 points) Calculate the AMAT using a 2-level (L1, L2 direct-mapped) cache.
3. (5 points) Calculate the AMAT using a 2-level (L1, L2 8-way set associative) cache.
4. (5 points) Calculate the AMAT using a 3-level (L1, L2 DM, L3) cache.
5. (5 points) Calculate the AMAT using a 3-level (L1, L2 8-way, L3) cache.

① AMAT = $1 + 0.03 \times 100 = 4$ clock cycle.

② AMAT = $1 + 0.03 \times (12 + 0.025 \times 100)$

$= 1.435$ clock cycle

③ AMAT = $1 + 0.03 \times (14 + 0.02 \times 100)$

$= 1.48$ clock cycle

④ AMAT = $1 + 0.03 \times (12 + 0.025 \times (40 + 0.015 \times 100))$

$= 1 + 0.03 \times (12 + 1.0375)$

$= 1.391125$ clock cycle

⑤ AMAT = $1 + 0.03 \times (14 + 0.02 \times (40 + 0.015 \times 100))$

$= 1 + 0.03 \times (14 + 0.83)$

$= 1.4449$ clock cycle.

3. (15 points) [section 5.7]

Consider the following page table parameters:

Virtual address size	Page size	Page table entry size
32 bits	32K bytes	4 bytes

1. (5 points) Please calculate the maximum page table size for a system running 5 processes (in MB).
2. (10 points) Assume we use the 2-level page table as specified by the following table.
   For a system running five applications that each utilize 100M Bytes of the virtual memory,
   please calculate the maximum page table size for a system running 5 processes (in MB).
   Please assume that P1 (first level) are all occupied and it also consume 4 bytes for each entry.

P1	P2	Offset
8 bits	9 bits	15 bits

1.

32 K bytes ⟶ 15 bit page offset

⇒ 32 − 15 = 17 bits for virtual page number

$2^{17} \times 4B = 0.5\ MB$ page size

one page for one process ⇒ 5 page for 5 process

$0.5 \times 5 = 2.5\ MB$

2.

$4 \times 2^8 + 2^8 \times 2^9 \times 4$

$= 2^{10} \times (1 + 2^9)$

$\approx 2^{19} B$

$= 0.5\ MB$

one page for one process ⇒ 5 page for 5 process

$0.5 \times 5 = 2.5MB$

To be accurate is 2.504 MB