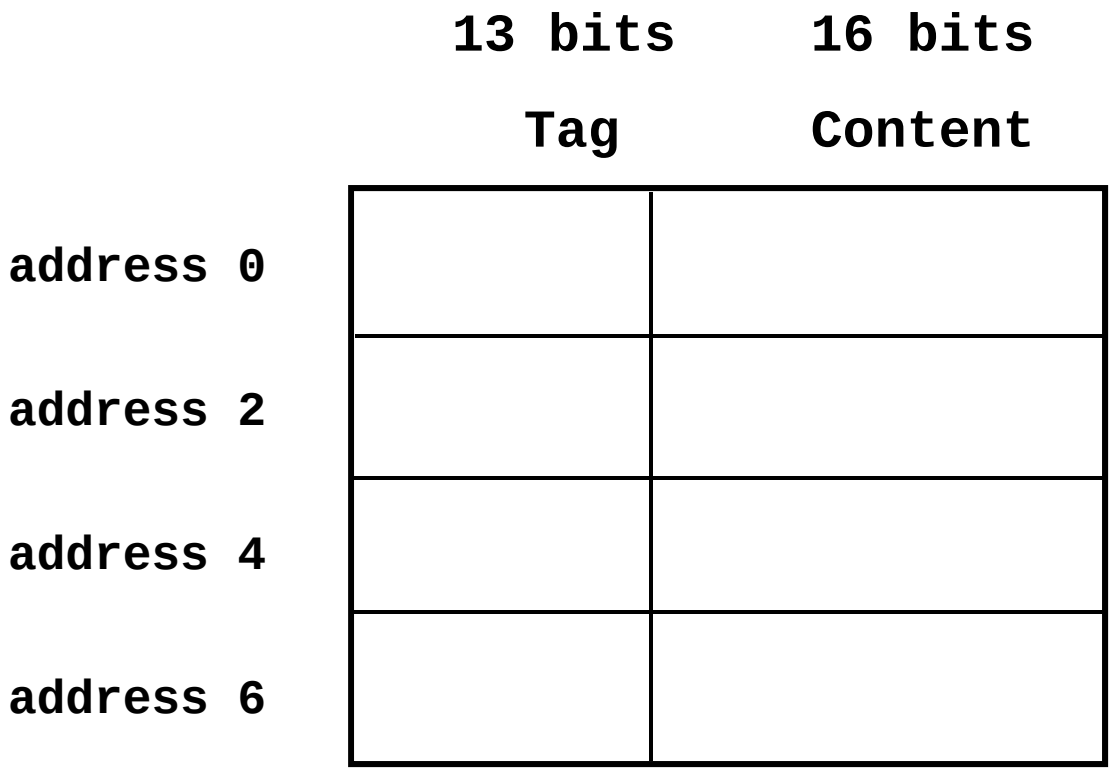
22a Cache Examples – Part 1
CSC 230

Department of Computer Science
University of Victoria

Problem 5.7 (from HVZ)

A computer uses a small direct-mapped cache between the main memory and the processor. The cache has **four 16-bit words**, and each word has an associated 13-bit tag, as shown in the figure below. When a miss occurs during a read operation, the requested word is read from the main memory and sent to the processor. At the same time, it is copied into the cache, and its block number is stored in the associated tag.



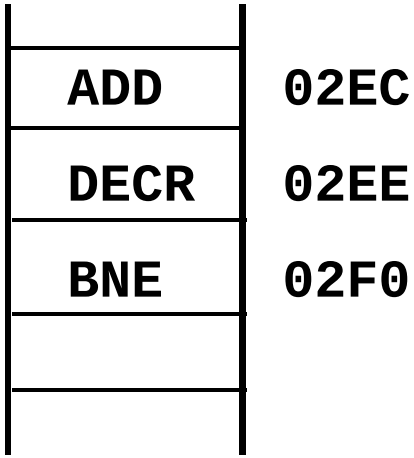
Consider the following loop in a program where all instructions and operands are 16 bits long and the code starts at address 02EC:

```
LOOP:  ADD    (R1)+,R0        02EC
       DECR   R2              02EE
       BNE    LOOP            02F0
```

Assume that, before this loop is entered, registers R0, R1 and R2 contain: R0 = 0, R1 = 054E, R2 = 3. Also assume that main memory contains the data as shown below:

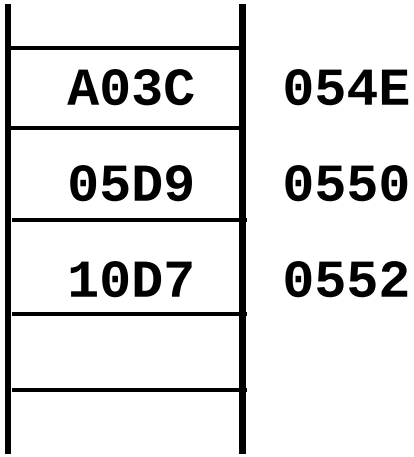
A03C	054E
05D9	0550
10D7	0552

Show the contents of the cache at the end of each pass through the loop.



02EC
02EE
02F0

. . . .



054E
0550
0552

```
LOOP:   ADD (R1)+,R0        02EC
        DECR    R2          02EE
        BNE     LOOP        02F0
```

translated to ARM code

```
LOOP:   LDR     R3,[R1],#2
        ADD     R0,R0,R3
        SUBS    R2,R2,#1
        BNE     LOOP
```

Example will remain in language as the book for consistency of the answers

```
LOOP:  ADD (R1)+,R0     02EC =   0000 0010 1110 1100    ➔ 4
       DECR   R2        02EE =   0000 0010 1110 1110    ➔ 6
       BNE LOOP         02F0 =   0000 0010 1111 0000    ➔ 0
```

tag

3 bits = which block in cache

A03C
05D9
10D7

```
054E =   0000 0101 0100 1110    ➔ 6
0550 =   0000 0101 0101 0000    ➔ 0
0552 =   0000 0101 0101 0010    ➔ 2
```

this tells us the cache block number where each data will be stored

this tells us the cache block number where each instruction will be stored

```
LOOP:  ADD (R1)+,R0    02EC =   0000 0010 1110 1100    ➔ 4
       DECR   R2       02EE =   0000 0010 1110 1110    ➔ 6
       BNE LOOP        02F0 =   0000 0010 1111 0000    ➔ 0
```

A03C	054E =	0000 0101 0100 1110	➔ 6
05D9	0550 =	0000 0101 0101 0000	➔ 0
10D7	0552 =	0000 0101 0101 0010	➔ 2

R0 = 0

R1 = 054E

R2 = 3

13 bits 16 bits

Tag Content

address 0		
address 2		
address 4		
address 6		

```
LOOP:  ADD (R1)+,R0    02EC =   0000 0010 1110 1100    ➔  4
       DECR   R2       02EE =   0000 0010 1110 1110    ➔  6
       BNE LOOP        02F0 =   0000 0010 1111 0000    ➔  0
```

A03C	054E =	0000 0101 0100 1110	➔ 6
05D9	0550 =	0000 0101 0101 0000	➔ 0
10D7	0552 =	0000 0101 0101 0010	➔ 2

R0 = 0

R1 = 054E

R2 = 3

13 bits 16 bits

Tag Content

Fetch ADD (1 M)

address 0		
address 2		
address 4		*ADD*
address 6		

```
LOOP:  ADD (R1)+,R0    02EC =   0000 0010 1110 1100    ➔ 4
       DECR   R2       02EE =   0000 0010 1110 1110    ➔ 6
       BNE LOOP        02F0 =   0000 0010 1111 0000    ➔ 0
```

A03C	054E =	0000 0101 0100 1110	➔ 6
05D9	0550 =	0000 0101 0101 0000	➔ 0
10D7	0552 =	0000 0101 0101 0010	➔ 2

R0 = 0

R1 = 054E

R2 = 3

13 bits 16 bits

Tag Content

Fetch (R1) (1 M)

address 0		
address 2		
address 4		*ADD*
address 6		*A03C*

```
LOOP:  ADD (R1)+,R0    02EC =   0000 0010 1110 1100    ➔ 4
       DECR   R2       02EE =   0000 0010 1110 1110    ➔ 6
       BNE LOOP        02F0 =   0000 0010 1111 0000    ➔ 0
```

A03C	054E =	0000 0101 0100 1110	➔ 6
05D9	0550 =	0000 0101 0101 0000	➔ 0
10D7	0552 =	0000 0101 0101 0010	➔ 2

R0 = 0 = A03C

R1 = 054E = 0550

R2 = 3

13 bits 16 bits

Tag Content

Increment R1 and ADD to R0

address 0		
address 2		
address 4		*ADD*
address 6		*A03C*

```
LOOP:  ADD (R1)+,R0    02EC =   0000 0010 1110 1100    ➔ 4
       DECR   R2       02EE =   0000 0010 1110 1110    ➔ 6
       BNE LOOP        02F0 =   0000 0010 1111 0000    ➔ 0
```

A03C	054E =	0000 0101 0100 1110	➔ 6
05D9	0550 =	0000 0101 0101 0000	➔ 0
10D7	0552 =	0000 0101 0101 0010	➔ 2

R0 = A03C

R1 = 0550

R2 = 3 = 2

13 bits 16 bits

Tag Content

Fetch DECR (1M)

1 collision

address 0		
address 2		
address 4		*ADD*
address 6		~~*A03C*~~ *DECR*

```
LOOP:  ADD (R1)+,R0    02EC =   0000 0010 1110 1100    ➔ 4
       DECR   R2       02EE =   0000 0010 1110 1110    ➔ 6
       BNE LOOP        02F0 =   0000 0010 1111 0000    ➔ 0
```

A03C	054E =	0000 0101 0100 1110	➔ 6
05D9	0550 =	0000 0101 0101 0000	➔ 0
10D7	0552 =	0000 0101 0101 0010	➔ 2

R0 = A03C

R1 = 0550

R2 = 2

13 bits 16 bits

Tag Content

Fetch BNE (1M)

address 0		*BNE*
address 2		
address 4		*ADD*
address 6		~~*A03C*~~ *DECR*

```
LOOP:  ADD (R1)+,R0    02EC =   0000 0010 1110 1100    ➔  4
       DECR   R2       02EE =   0000 0010 1110 1110    ➔  6
       BNE LOOP        02F0 =   0000 0010 1111 0000    ➔  0
```

A03C	054E =	0000 0101 0100 1110	➔ 6
05D9	0550 =	0000 0101 0101 0000	➔ 0
10D7	0552 =	0000 0101 0101 0010	➔ 2

R0 = A03C

R1 = 0550

R2 = 2

13 bits 16 bits

Tag Content

After 1st pass in loop

4 M

address 0		*BNE*
address 2		
address 4		*ADD*
address 6		*DECR*

2nd pass

```
LOOP:  ADD (R1)+,R0    02EC =   0000 0010 1110 1100    ➔ 4
       DECR   R2       02EE =   0000 0010 1110 1110    ➔ 6
       BNE LOOP        02F0 =   0000 0010 1111 0000    ➔ 0
```

A03C	054E =	0000 0101 0100 1110	➔ 6
05D9	0550 =	0000 0101 0101 0000	➔ 0
10D7	0552 =	0000 0101 0101 0010	➔ 2

R0 = 0

R1 = 0550

R2 = 2

13 bits 16 bits

Tag Content

Fetch ADD (1 C)

address 0		*BNE*
address 2		
address 4		*ADD*
address 6		*DECR*

```
LOOP:  ADD (R1)+,R0    02EC =   0000 0010 1110 1100    ➔  4
       DECR   R2       02EE =   0000 0010 1110 1110    ➔  6
       BNE LOOP        02F0 =   0000 0010 1111 0000    ➔  0
```

A03C	054E =	0000 0101 0100 1110	➔ 6
05D9	0550 =	0000 0101 0101 0000	➔ 0
10D7	0552 =	0000 0101 0101 0010	➔ 2

R0 = 0

R1 = 0550

R2 = 2

13 bits 16 bits

Tag Content

Fetch (R1) (1 M)

1 collision

address 0		~~BNE~~ 05D9
address 2		
address 4		ADD
address 6		DECR

```
LOOP:  ADD (R1)+,R0    02EC =   0000 0010 1110 1100    ➔  4
       DECR   R2       02EE =   0000 0010 1110 1110    ➔  6
       BNE LOOP        02F0 =   0000 0010 1111 0000    ➔  0
```

A03C	054E =	0000 0101 0100 1110	➔ 6
05D9	0550 =	0000 0101 0101 0000	➔ 0
10D7	0552 =	0000 0101 0101 0010	➔ 2

R0 = 0 = A615

R1 = 0550 = 0552

R2 = 2

13 bits 16 bits

Tag Content

Increment R1 and ADD to R0

address 0		*05D9*
address 2		
address 4		*ADD*
address 6		*DECR*

```
LOOP:  ADD (R1)+,R0    02EC =   0000 0010 1110 1100    ➔  4
       DECR   R2       02EE =   0000 0010 1110 1110    ➔  6
       BNE LOOP        02F0 =   0000 0010 1111 0000    ➔  0
```

A03C	054E =	0000 0101 0100 1110	➔ 6
05D9	0550 =	0000 0101 0101 0000	➔ 0
10D7	0552 =	0000 0101 0101 0010	➔ 2

R0 = A615

R1 = 0552

R2 = 2 = 1

13 bits 16 bits

Tag Content

Fetch DECR (1C)

address 0		*05D9*
address 2		
address 4		*ADD*
address 6		*DECR*

```
LOOP:  ADD (R1)+,R0    02EC =   0000 0010 1110 1100    ➔ 4
       DECR   R2       02EE =   0000 0010 1110 1110    ➔ 6
       BNE LOOP        02F0 =   0000 0010 1111 0000    ➔ 0
```

A03C	054E =	0000 0101 0100 1110	➔ 6
05D9	0550 =	0000 0101 0101 0000	➔ 0
10D7	0552 =	0000 0101 0101 0010	➔ 2

R0 = A615

R1 = 0552

R2 = 1

13 bits 16 bits

Tag Content

Fetch BNE (1M)

1 collision

address 0		~~05D9~~ *BNE*
address 2		
address 4		*ADD*
address 6		*DECR*

```
LOOP:  ADD (R1)+,R0    02EC =   0000 0010 1110 1100    ➔  4
       DECR   R2       02EE =   0000 0010 1110 1110    ➔  6
       BNE LOOP        02F0 =   0000 0010 1111 0000    ➔  0
```

A03C	054E =	0000 0101 0100 1110	➔ 6
05D9	0550 =	0000 0101 0101 0000	➔ 0
10D7	0552 =	0000 0101 0101 0010	➔ 2

R0 = A615

R1 = 0552

R2 = 1

13 bits 16 bits

Tag Content

After 2nd pass in loop 2 M + 2C

address 0		*BNE*
address 2		
address 4		*ADD*
address 6		*DECR*

3rd pass

```
LOOP:  ADD (R1)+,R0    02EC =   0000 0010 1110 1100    ➔ 4
       DECR   R2       02EE =   0000 0010 1110 1110    ➔ 6
       BNE LOOP        02F0 =   0000 0010 1111 0000    ➔ 0
```

A03C	054E =	0000 0101 0100 1110	➔ 6
05D9	0550 =	0000 0101 0101 0000	➔ 0
10D7	0552 =	0000 0101 0101 0010	➔ 2

R0 = A615

R1 = 0552

R2 = 1

13 bits 16 bits

Tag Content

Fetch ADD (1 C)

address 0		*BNE*
address 2		
address 4		*ADD*
address 6		*DECR*

```
LOOP:  ADD (R1)+,R0    02EC =   0000 0010 1110 1100    ➔ 4
       DECR   R2       02EE =   0000 0010 1110 1110    ➔ 6
       BNE LOOP        02F0 =   0000 0010 1111 0000    ➔ 0
```

A03C	054E =	0000 0101 0100 1110	➔ 6
05D9	0550 =	0000 0101 0101 0000	➔ 0
10D7	0552 =	0000 0101 0101 0010	➔ 2

R0 = A615

R1 = 0552

R2 = 1

13 bits 16 bits

Fetch (R1) (1 M)

Tag Content

address 0		*BNE*
address 2		*10D7*
address 4		*ADD*
address 6		*DECR*

```
LOOP:  ADD (R1)+,R0    02EC =   0000 0010 1110 1100    ➔  4
       DECR   R2       02EE =   0000 0010 1110 1110    ➔  6
       BNE LOOP        02F0 =   0000 0010 1111 0000    ➔  0
```

A03C	054E =	0000 0101 0100 1110	➔ 6
05D9	0550 =	0000 0101 0101 0000	➔ 0
10D7	0552 =	0000 0101 0101 0010	➔ 2

R0 = B6EC

R1 = 0552 = 0554

R2 = 1

13 bits 16 bits

Tag Content

Increment R1 and ADD to R0

address 0		*BNE*
address 2		*10D7*
address 4		*ADD*
address 6		*DECR*

```
LOOP:  ADD (R1)+,R0    02EC =   0000 0010 1110 1100    ➔ 4
       DECR   R2       02EE =   0000 0010 1110 1110    ➔ 6
       BNE LOOP        02F0 =   0000 0010 1111 0000    ➔ 0
```

A03C	054E =	0000 0101 0100 1110	➔ 6
05D9	0550 =	0000 0101 0101 0000	➔ 0
10D7	0552 =	0000 0101 0101 0010	➔ 2

R0 = B6EC

R1 = 0554

R2 = 1 = 0

13 bits 16 bits

Tag Content

Fetch DECR (1C)

address 0		*BNE*
address 2		*10D7*
address 4		*ADD*
address 6		*DECR*

```
LOOP:  ADD (R1)+,R0    02EC =   0000 0010 1110 1100    ➔ 4
       DECR   R2       02EE =   0000 0010 1110 1110    ➔ 6
       BNE LOOP        02F0 =   0000 0010 1111 0000    ➔ 0
```

A03C	054E =	0000 0101 0100 1110	➔ 6
05D9	0550 =	0000 0101 0101 0000	➔ 0
10D7	0552 =	0000 0101 0101 0010	➔ 2

R0 = B6EC

R1 = 0554

R2 = 0

13 bits 16 bits

Tag Content

Fetch BNE (1C)

address 0		*BNE*
address 2		*10D7*
address 4		*ADD*
address 6		*DECR*

```
LOOP:  ADD (R1)+,R0    02EC =   0000 0010 1110 1100    ➔  4
       DECR   R2       02EE =   0000 0010 1110 1110    ➔  6
       BNE LOOP        02F0 =   0000 0010 1111 0000    ➔  0
```

A03C	054E =	0000 0101 0100 1110	➔ 6
05D9	0550 =	0000 0101 0101 0000	➔ 0
10D7	0552 =	0000 0101 0101 0010	➔ 2

R0 = B6EC

R1 = 0554

R2 = 0

13 bits 16 bits

Tag Content

After 3rd pass in loop 1 M + 3C

address 0		*BNE*
address 2		*10D7*
address 4		*ADD*
address 6		*DECR*

Assume that the access time of the main memory is 10 t and that of cache is 1 t. Calculate the execution time for each pass, ignoring the time taken by the processor between memory cycles.

1st pass	➔	**4 memory**	➔	**40t**
2nd pass	➔	**2 memory, 2 cache**	➔	**22 t**
3rd pass	➔	**1 memory, 3 cache**	➔	**13 t**

Repeat process assuming separate instruction and data cache. Compute speed of execution. Compare values to the ones obtained here with a single cache. Analyze and comment.

Another example (HVZ Exercise 5.6)

How are the total bits organized in a direct-mapped cache with:

Main memory = 64K words

Cache size = 1K words

Block size = 128 words

- ✓ **Memory = 64 K words = 2^6 x 2^{10} words = 2^{16} words**
- ✓ **Cache = 1 K words = 2^{10} words**

How is the cache organized exactly?

Since block size = 128 words = 2^7 words, then:

2^{10} words / 2^7 words = 2^3 ➔ 8 blocks in cache

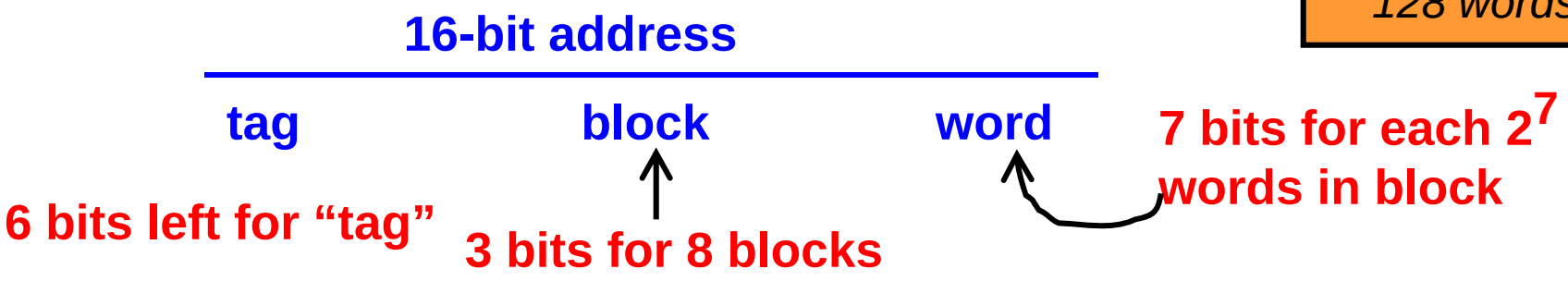
thus cache has 8 blocks, each containing 128 words

Exercise 5.6

Memory = 64 K words = 2^6 x 2^{10} words = 2^{16} words

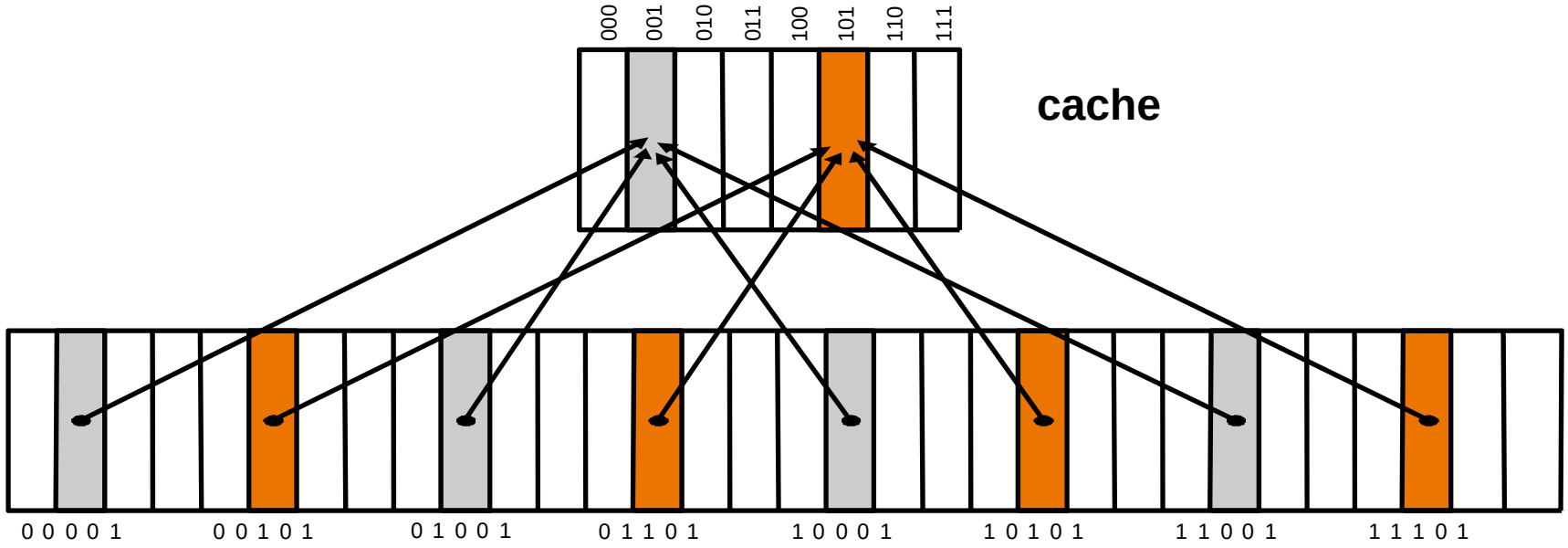
Cache = 1 K words = 2^{10} words

READ ONLY



Index	V	Tag	Data
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	N		
111	N		

Example of hit/misses in direct-mapped cache



Index	V	Tag	Data
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	N	10	Memory[10110]
111	N		

Binary address of reference	Hit or miss in cache	cache block (where)
10110	miss	110
11010	miss	010
10110	hit	110
11010	hit	010
10000	miss	000
00011	miss	011
10000	hit	000
10010	miss	010

Index	V	Tag	Data
000	N		
001	N		
010	N	11	Memory[11010]
011	N		
100	N		
101	N		
110	N	10	Memory[10110]
111	N		

Binary address of reference	Hit or miss in cache	cache block (where)
10110	miss	110
11010	miss	010
10110	hit	110
11010	hit	010
10000	miss	000
00011	miss	011
10000	hit	000
10010	miss	010

Index	V	Tag	Data
000	N		
001	N		
010	N	11	Memory[11010]
011	N		
100	N		
101	N		
110	N	10	Memory[10110]
111	N		

Binary address of reference	Hit or miss in cache	cache block (where)
10110	miss	110
11010	miss	010
10110	hit	110
11010	hit	010
10000	miss	000
00011	miss	011
10000	hit	000
10010	miss	010

Index	V	Tag	Data
000	N		
001	N		
010	N	11	Memory[11010]
011	N		
100	N		
101	N		
110	N	10	Memory[10110]
111	N		

Binary address of reference	Hit or miss in cache	cache block (where)
10110	miss	110
11010	miss	010
10110	hit	110
11010	hit	010
10000	miss	000
00011	miss	011
10000	hit	000
10010	miss	010

Index	V	Tag	Data
000	N	10	Memory[10000]
001	N		
010	N	11	Memory[11010]
011	N		
100	N		
101	N		
110	N	10	Memory[10110]
111	N		

Binary address of reference	Hit or miss in cache	cache block (where)
10110	miss	110
11010	miss	010
10110	hit	110
11010	hit	010
10000	miss	000
00011	miss	011
10000	hit	000
10010	miss	010

Index	V	Tag	Data
000	N	10	Memory[10000]
001	N		
010	N	11	Memory[11010]
011	N	00	Memory[00011]
100	N		
101	N		
110	N	10	Memory[10110]
111	N		

Binary address of reference	Hit or miss in cache	cache block (where)
10110	miss	110
11010	miss	010
10110	hit	110
11010	hit	010
10000	miss	000
00011	miss	011
10000	hit	000
10010	miss	010

Index	V	Tag	Data
000	N	10	Memory[10000]
001	N		
010	N	10	Memory[10010]
011	N	00	Memory[00011]
100	N		
101	N		
110	N	10	Memory[10110]
111	N		

Binary address of reference	Hit or miss in cache	cache block (where)
10110	miss	110
11010	miss	010
10110	hit	110
11010	hit	010
10000	miss	000
00011	miss	011
10000	hit	000
10010	miss	010

Think it through

How many total bits are required for a direct-mapped cache with 16KB of data and 4-word blocks, assuming a 32-bit address and word size = 4 bytes?

Think it through: read only

How many total bits are required for a direct-mapped cache with 16KB of data and 4-word blocks, assuming a 32-bit address and word size = 4 bytes?

$16KB = 16 \times 1024 = 2^4 \times 2^{10} = 2^2 \times 2^2 \times 2^{10}$

2^2: each of 4 bytes

$2^2 \times 2^{10}$: 4K words

2^{12} words ➔ with block size of 4 words ➔ 2^{10} blocks

where each block has 4 x 32 bits = 128 bits, plus tag

tag = 32 – 10 – 4 = 18 (10 is # blocks, 4 is #words in block)

Cache size = $2^{10} \times (128 + (18) + 1) = 2^{10} \times 147 = 147$ K bits

or 18.4 KB for a 16 KB cache

➔ 1.15 times the size of data ➔ 15% increase