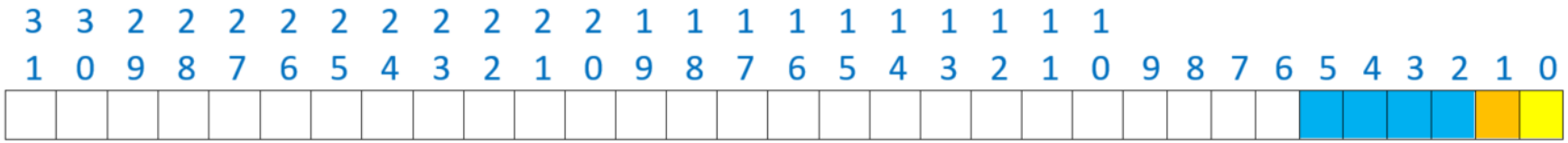
QUESTION 1(A)

Addresses are for bytes, and two bytes make up a word.
So any two addresses that share the upper 31 bits are of the same word.
A block is made up of two words. So any two addresses that share the upper 30 bits are in the same block.
And the 4 bits bits 2-5 forms the block index for the 16 block direct mapped cache.



4 = 0x004 = 0b00000000000000000000000000**0001**00

Memory address		Hit (H) or Miss (M)?
(in decimal)	(in hexadecimal)	
4	4	M (cold)

92 = 0x05c = 0b00000000000000000000000001**0111**00

Memory address		Hit (H) or Miss (M)?	
(in decimal)	(in hexadecimal)		
4	4	M (cold)	Block 1
92	5C	M (cold)	

```
7 = 0x007 = 0b00000000000000000000000000000111
```

Memory address		Hit (H) or Miss (M)?	
(in decimal)	(in hexadecimal)		
4	4	M (cold)	Block 1
92	5C	M (cold)	Block 7
7	7	H (same block as 0x4)	

146 = 0x092 = 0b00000000000000000000000010**0100**10

Memory address		Hit (H) or Miss (M)?	
(in decimal)	(in hexadecimal)		
4	4	M (cold)	Block 1
92	5C	M (cold)	Block 7
7	7	H (same block as 0x4)	Block 1
146	92	M (cold)	

```
30 = 0x01e = 0b00000000000000000000000000011110
```

Memory address		Hit (H) or Miss (M)?	
(in decimal)	(in hexadecimal)		
4	4	M (cold)	Block 1
92	5C	M (cold)	Block 7
7	7	H (same block as 0x4)	Block 1
146	92	M (cold)	Block 4
30	1E	M (cold)	

95 = 0x05f = 0b00000000000000000000000101**0111**11

Memory address		Hit (H) or Miss (M)?	
(in decimal)	(in hexadecimal)		
4	4	M (cold)	Block 1
92	5C	M (cold)	Block 7
7	7	H (same block as 0x4)	Block 1
146	92	M (cold)	Block 4
30	1E	M (cold)	Block 7
95	5F	M (conflict with 0x1e)	

176 = 0x0b0 = 0b00000000000000000000000010**1100**00

Memory address		Hit (H) or Miss (M)?	
(in decimal)	(in hexadecimal)		
4	4	M (cold)	Block 1
92	5C	M (cold)	Block 7
7	7	H (same block as 0x4)	Block 1
146	92	M (cold)	Block 4
30	1E	M (cold)	Block 7
95	5F	M (conflict with 0x1e)	Block 7
176	B0	M (cold)	

93 = 0x05d = 0b00000000000000000000000001**0111**01

Memory address		Hit (H) or Miss (M)?	
(in decimal)	(in hexadecimal)		
4	4	M (cold)	Block 1
92	5C	M (cold)	Block 7
7	7	H (same block as 0x4)	Block 1
146	92	M (cold)	Block 4
30	1E	M (cold)	Block 7
95	5F	M (conflict with 0x1e)	Block 7
176	B0	M (cold)	Block 12
93	5D	H (same word as 0x5c)	

145 = 0x091 = 0b00000000000000000000000010**0100**01

Memory address		Hit (H) or Miss (M)?	
(in decimal)	(in hexadecimal)		
4	4	M (cold)	Block 1
92	5C	M (cold)	Block 7
7	7	H (same block as 0x4)	Block 1
146	92	M (cold)	Block 4
30	1E	M (cold)	Block 7
95	5F	M (conflict with 0x1e)	Block 7
176	B0	M (cold)	Block 12
93	5D	H (same word as 0x5c)	Block 7
145	91	H (same block as 0x92)	

264 = 0x108 = 0b000000000000000000000100**0010**00

Memory address (in decimal)	Memory address (in hexadecimal)	Hit (H) or Miss (M)?	
4	4	M (cold)	Block 1
92	5C	M (cold)	Block 7
7	7	H (same block as 0x4)	Block 1
146	92	M (cold)	Block 4
30	1E	M (cold)	Block 7
95	5F	M (conflict with 0x1e)	Block 7
176	B0	M (cold)	Block 12
93	5D	H (same word as 0x5c)	Block 7
145	91	H (same block as 0x92)	Block 4
264	108	M (cold)	

```
6 = 0x006 = 0b00000000000000000000000000000110
```

Memory address (in decimal)	Memory address (in hexadecimal)	Hit (H) or Miss (M)?	
4	4	M (cold)	Block 1
92	5C	M (cold)	Block 7
7	7	H (same block as 0x4)	Block 1
146	92	M (cold)	Block 4
30	1E	M (cold)	Block 7
95	5F	M (conflict with 0x1e)	Block 7
176	B0	M (cold)	Block 12
93	5D	H (same word as 0x5c)	Block 7
145	91	H (same block as 0x92)	Block 4
264	108	M (cold)	Block 2
6	6	H (same word as 0x7)	

QUESTION 1(B)

General approach: work the trace backwards!

6 = 0x006
= 0b00000000000000000000000000000110

Block index = 1
Tag = 0x0000000

In Word 1

Index	Tag value	Word 0	Word 1
0			
1	0x0000000	M[0x4]	M[0x6]
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			

264 = 0x108
= 0b00000000000000000000000100001000

Block index = 2
Tag = 0x0000004

In Word 0

Index	Tag value	Word 0	Word 1
0			
1	0x0000000	M[0x4]	M[0x6]
2	0x0000004	M[0x108]	M[0x10a]
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			

145 = 0x091
= 0b00000000000000000000000010010001

Block index = 4
Tag = 0x0000002

In Word 0

Index	Tag value	Word 0	Word 1
0			
1	0x0000000	M[0x4]	M[0x6]
2	0x0000004	M[0x108]	M[0x10a]
3			
4	0x0000002	M[0x90]	M[0x92]
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			

93 = 0x05d
= 0b00000000000000000000000001011101

Block index = 7
Tag = 0x0000001

In Word 0

Index	Tag value	Word 0	Word 1
0			
1	0x0000000	M[0x4]	M[0x6]
2	0x0000004	M[0x108]	M[0x10a]
3			
4	0x0000002	M[0x90]	M[0x92]
5			
6			
7	0x0000001	M[0x5c]	M[0x5e]
8			
9			
10			
11			
12			
13			
14			
15			

176 = 0x0b0
= 0b00000000000000000000000010110000

Block index = 12
Tag = 0x0000002

In Word 0

Index	Tag value	Word 0	Word 1
0			
1	0x0000000	M[0x4]	M[0x6]
2	0x0000004	M[0x108]	M[0x10a]
3			
4	0x0000002	M[0x90]	M[0x92]
5			
6			
7	0x0000001	M[0x5c]	M[0x5e]
8			
9			
10			
11			
12	0x0000002	M[0xb0]	M[0xb2]
13			
14			
15			

95 = 0x05f
= 0b00000000000000000000000001011111

Block index = 7
Tag = 0x0000001

In Word 1

Index	Tag value	Word 0	Word 1
0			
1	0x0000000	M[0x4]	M[0x6]
2	0x0000004	M[0x108]	M[0x10a]
3			
4	0x0000002	M[0x90]	M[0x92]
5			
6			
7	0x0000001	M[0x5c]	M[0x5e]
8			
9			
10			
11			
12	0x0000002	M[0xb0]	M[0xb2]
13			
14			
15			

30 = 0x01e
= 0b00000000000000000000000000011110

Block index = 7
Tag = 0x0000000

In Word 1

Index	Tag value	Word 0	Word 1
0			
1	0x0000000	M[0x4]	M[0x6]
2	0x0000004	M[0x108]	M[0x10a]
3			
4	0x0000002	M[0x90]	M[0x92]
5			
6			
7	0x0000001	M[0x5c]	M[0x5e]
8			
9			
10			
11			
12	0x0000002	M[0xb0]	M[0xb2]
13			
14			
15			

146 = 0x092
= 0b00000000000000000000000010010010

Block index = 4
Tag = 0x0000002

In Word 1

Index	Tag value	Word 0	Word 1
0			
1	0x0000000	M[0x4]	M[0x6]
2	0x0000004	M[0x108]	M[0x10a]
3			
4	0x0000002	M[0x90]	M[0x92]
5			
6			
7	0x0000001	M[0x5c]	M[0x5e]
8			
9			
10			
11			
12	0x0000002	M[0xb0]	M[0xb2]
13			
14			
15			

7 = 0x007
= 0b00000000000000000000000000000111

Block index = 1
Tag = 0x0000000

In Word 1

Index	Tag value	Word 0	Word 1
0			
1	0x0000000	M[0x4]	M[0x6]
2	0x0000004	M[0x108]	M[0x10a]
3			
4	0x0000002	M[0x90]	M[0x92]
5			
6			
7	0x0000001	M[0x5c]	M[0x5e]
8			
9			
10			
11			
12	0x0000002	M[0xb0]	M[0xb2]
13			
14			
15			

92 = 0x05c
= 0b00000000000000000000000001011100

Block index = 7
Tag = 0x0000001

In Word 0

Index	Tag value	Word 0	Word 1
0			
1	0x0000000	M[0x4]	M[0x6]
2	0x0000004	M[0x108]	M[0x10a]
3			
4	0x0000002	M[0x90]	M[0x92]
5			
6			
7	0x0000001	M[0x5c]	M[0x5e]
8			
9			
10			
11			
12	0x0000002	M[0xb0]	M[0xb2]
13			
14			
15			

4 = 0x004
= 0b00000000000000000000000000000100

Block index = 1
Tag = 0x0000000

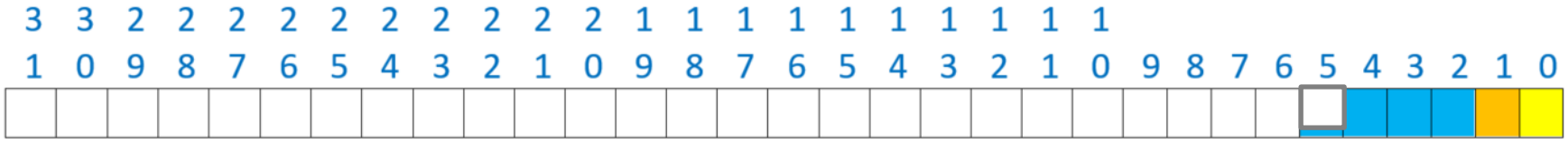
In Word 0

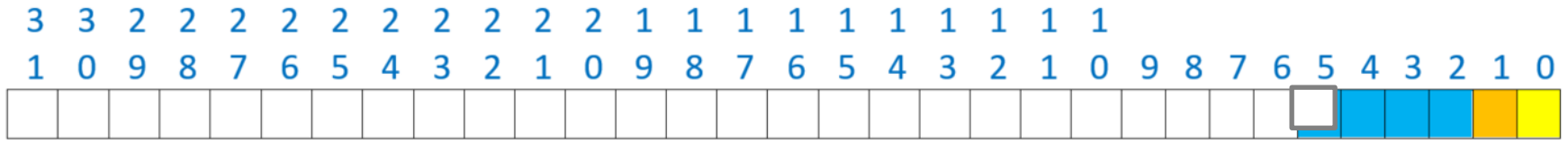
Index	Tag value	Word 0	Word 1
0			
1	0x0000000	M[0x4]	M[0x6]
2	0x0000004	M[0x108]	M[0x10a]
3			
4	0x0000002	M[0x90]	M[0x92]
5			
6			
7	0x0000001	M[0x5c]	M[0x5e]
8			
9			
10			
11			
12	0x0000002	M[0xb0]	M[0xb2]
13			
14			
15			

QUESTION 1(C)

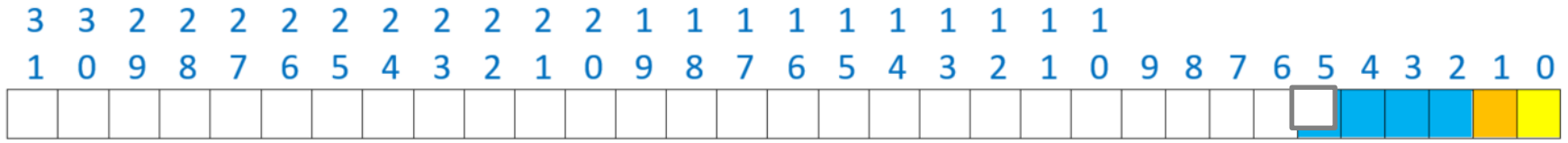
Addresses are for bytes, and two bytes make up a word.
So any two addresses that share the upper 31 bits are of the same word.
A block is made up of two words. So any two addresses that share the upper 30 bits are in the same block.

The 3 bits (bits 2-4) forms the set index for the 16-block two-way set associative cache.





Memory address (in decimal)	Memory address (in hexadecimal)	Hit (H) or Miss (M)?	
4	4	M (cold)	Block 1
92	5C	M (cold)	Block 7
7	7	H (same block as 0x4)	Block 1
146	92	M (cold)	Block 4
30	1E	M (cold)	Block 7
95	5F	M (conflict with 0x1e)	Block 7
176	B0	M (cold)	Block 12
93	5D	H (same word as 0x5c)	Block 7
145	91	H (same block as 0x92)	Block 4
264	108	M (cold)	Block 2
6	6	H (same word as 0x7)	



Memory address (in decimal)	Memory address (in hexadecimal)	Hit (H) or Miss (M)?	
4	4	M (cold)	Block 1
92	5C	M (cold)	Block 7
7	7	H (same block as 0x4)	Block 1
146	92	M (cold)	Block 4
30	1E	M (cold)	Block 7
95	5F	H (same word as 0x5c)	Block 7
176	B0	M (cold)	Block 4
93	5D	H (same word as 0x5c)	Block 7
145	91	H (same block as 0x92)	Block 4
264	108	M (cold)	Block 2
6	6	H (same word as 0x7)	

3 bits

```
4 = 0x004 = 0b00000000000000000000000000100
```

Memory address		Hit (H) or Miss (M)?
(in decimal)	(in hexadecimal)	
4	4	M (cold)

92 = 0x05c = 0b00000000000000000000000010111100

Memory address		Hit (H) or Miss (M)?
(in decimal)	(in hexadecimal)	
4	4	M (cold)
92	5C	M (cold)

Block 1

```
7 = 0x007 = 0b00000000000000000000000000001111
```

Memory address		Hit (H) or Miss (M)?	
(in decimal)	(in hexadecimal)		
4	4	M (cold)	Block 1
92	5C	M (cold)	Block 7
7	7	H (same block as 0x4)	

```
146 = 0x092 = 0b00000000000000000000000010010010
```

Memory address		Hit (H) or Miss (M)?
(in decimal)	(in hexadecimal)	
4	4	M (cold)
92	5C	M (cold)
7	7	H (same block as 0x4)
146	92	M (cold)

```
30 = 0x01e = 0b00000000000000000000000000011110
```

Memory address		Hit (H) or Miss (M)?
(in decimal)	(in hexadecimal)	
4	4	M (cold)
92	5C	M (cold)
7	7	H (same block as 0x4)
146	92	M (cold)
30	1E	M (cold)

95 = 0x05f = 0b00000000000000000000000010111**11**

Memory address		Hit (H) or Miss (M)?
(in decimal)	(in hexadecimal)	
4	4	M (cold)
92	5C	M (cold)
7	7	H (same block as 0x4)
146	92	M (cold)
30	1E	M (cold)
95	5F	H (same block as 0x5c)

```
176 = 0x0b0 = 0b00000000000000000000000101100000
```

Memory address		Hit (H) or Miss (M)?
(in decimal)	(in hexadecimal)	
4	4	M (cold)
92	5C	M (cold)
7	7	H (same block as 0x4)
146	92	M (cold)
30	1E	M (cold)
95	5F	H (same block as 0x5c)
176	B0	M (cold)

```
93 = 0x05d = 0b00000000000000000000000010111101
```

Memory address		Hit (H) or Miss (M)?
(in decimal)	(in hexadecimal)	
4	4	M (cold)
92	5C	M (cold)
7	7	H (same block as 0x4)
146	92	M (cold)
30	1E	M (cold)
95	5F	H (same block as 0x5c)
176	B0	M (cold)
93	5D	H (same word as 0x5c)

145 = 0x091 = 0b00000000000000000000000010010001

Memory address		Hit (H) or Miss (M)?
(in decimal)	(in hexadecimal)	
4	4	M (cold)
92	5C	M (cold)
7	7	H (same block as 0x4)
146	92	M (cold)
30	1E	M (cold)
95	5F	H (same block as 0x5c)
176	B0	M (cold)
93	5D	H (same word as 0x5c)
145	91	H (same block as 0x92)

264 = 0x108 = 0b00000000000000000000000100001000

Memory address		Hit (H) or Miss (M)?
(in decimal)	(in hexadecimal)	
4	4	M (cold)
92	5C	M (cold)
7	7	H (same block as 0x4)
146	92	M (cold)
30	1E	M (cold)
95	5F	H (same block as 0x5c)
176	B0	M (cold)
93	5D	H (same word as 0x5c)
145	91	H (same block as 0x92)
264	108	M (cold)

```
6 = 0x006 = 0b00000000000000000000000000001110
```

Memory address		Hit (H) or Miss (M)?
(in decimal)	(in hexadecimal)	
4	4	M (cold)
92	5C	M (cold)
7	7	H (same block as 0x4)
146	92	M (cold)
30	1E	M (cold)
95	5F	H (same block as 0x5c)
176	B0	M (cold)
93	5D	H (same word as 0x5c)
145	91	H (same block as 0x92)
264	108	M (cold)
6	6	H (same word as 0x7)

QUESTION 1(C) – SECOND PART

General approach: work the trace backwards!

6 = 0x006
= 0b00000000000000000000000000000110

Block index = 1
Tag = 0x0000000

In Word 1

Index	Set 0			Set 1		
	Tag	Word 0	Word 1	Tag	Word 0	Word 1
0						
1	**0x0000000**	**M[0x4]**	**M[0x6]**			
2						
3						
4						
5						
6						
7						

264 = 0x108
= 0b00000000000000000000000100001000

Block index = 2
Tag = 0x0000008

In Word 0

Index	Set 0			Set 1		
	Tag	Word 0	Word 1	Tag	Word 0	Word 1
0						
1	**0x0000000**	**M[0x4]**	**M[0x6]**			
2	**0x0000008**	**M[0x108]**	**M[0x10a]**			
3						
4						
5						
6						
7						

145 = 0x091
= 0b00000000000000000000000100100001

Block index = 4
Tag = 0x0000004

In Word 0

Index	Set 0			Set 1		
	Tag	Word 0	Word 1	Tag	Word 0	Word 1
0						
1	**0x0000000**	**M[0x4]**	**M[0x6]**			
2	**0x0000008**	**M[0x108]**	**M[0x10a]**			
3						
4	**0x0000004**	**M[0x90]**	**M[0x92]**			
5						
6						
7						

93 = 0x05d
= 0b00000000000000000000000010111101

Block index = 7
Tag = 0x0000002

In Word 0

Index	Set 0			Set 1		
	Tag	Word 0	Word 1	Tag	Word 0	Word 1
0						
1	**0x0000000**	**M[0x4]**	**M[0x6]**			
2	**0x0000008**	**M[0x108]**	**M[0x10a]**			
3						
4	**0x0000004**	**M[0x90]**	**M[0x92]**			
5						
6						
7	**0x0000002**	**M[0x5c]**	**M[0x5e]**			

176 = 0x0b0
= 0b00000000000000000000000101100000

Block index = 4
Tag = 0x0000005

In Word 0

Index	Set 0			Set 1		
	Tag	Word 0	Word 1	Tag	Word 0	Word 1
0						
1	**0x0000000**	**M[0x4]**	**M[0x6]**			
2	**0x0000008**	**M[0x108]**	**M[0x10a]**			
3						
4	**0x0000004**	**M[0x90]**	**M[0x92]**	**0x0000005**	**M[0xb0]**	**M[0xb2]**
5						
6						
7	**0x0000002**	**M[0x5c]**	**M[0x5e]**			

95 = 0x05f
= 0b00000000000000000000000010111 11

Block index = 7
Tag = 0x0000002

In Word 1

Index	Set 0			Set 1		
	Tag	Word 0	Word 1	Tag	Word 0	Word 1
0						
1	0x0000000	M[0x4]	M[0x6]			
2	0x0000008	M[0x108]	M[0x10a]			
3						
4	0x0000004	M[0x90]	M[0x92]	0x0000005	M[0xb0]	M[0xb2]
5						
6						
7	0x0000002	M[0x5c]	M[0x5e]			

30 = 0x01e
= 0b00000000000000000000000000011110

Block index = 7
Tag = 0x0000000

In Word 1

Index	Set 0			Set 1		
	Tag	Word 0	Word 1	Tag	Word 0	Word 1
0						
1	0x0000000	M[0x4]	M[0x6]			
2	0x0000008	M[0x108]	M[0x10a]			
3						
4	0x0000004	M[0x90]	M[0x92]	0x0000005	M[0xb0]	M[0xb2]
5						
6						
7	0x0000002	M[0x5c]	M[0x5e]	0x0000000	M[0x1c]	M[0x1e]

146 = 0x092
= 0b00000000000000000000000100100010

Block index = 4
Tag = 0x0000004

In Word 1

Index	Set 0			Set 1		
	Tag	Word 0	Word 1	Tag	Word 0	Word 1
0						
1	0x0000000	M[0x4]	M[0x6]			
2	0x0000008	M[0x108]	M[0x10a]			
3						
4	0x0000004	M[0x90]	M[0x92]	0x0000005	M[0xb0]	M[0xb2]
5						
6						
7	0x0000002	M[0x5c]	M[0x5e]	0x0000000	M[0x1c]	M[0x1e]

7 = 0x007
= 0b00000000000000000000000000000111

Block index = 1
Tag = 0x0000000

In Word 1

Index	Set 0			Set 1		
	Tag	Word 0	Word 1	Tag	Word 0	Word 1
0						
1	0x0000000	M[0x4]	M[0x6]			
2	0x0000008	M[0x108]	M[0x10a]			
3						
4	0x0000004	M[0x90]	M[0x92]	0x0000005	M[0xb0]	M[0xb2]
5						
6						
7	0x0000002	M[0x5c]	M[0x5e]	0x0000000	M[0x1c]	M[0x1e]

92 = 0x05c
= 0b00000000000000000000000010111100

Block index = 7
Tag = 0x0000002

In Word 0

Index	Set 0			Set 1		
	Tag	Word 0	Word 1	Tag	Word 0	Word 1
0						
1	0x0000000	M[0x4]	M[0x6]			
2	0x0000008	M[0x108]	M[0x10a]			
3						
4	0x0000004	M[0x90]	M[0x92]	0x0000005	M[0xb0]	M[0xb2]
5						
6						
7	0x0000002	M[0x5c]	M[0x5e]	0x0000000	M[0x1c]	M[0x1e]

Note that if we traced forwards and fill set 0 before set 1, then for index 4 and index 7, set 0 and set 1 contents would be swapped. Both answers are acceptable for the exams. Just make sure the correct words are in the cache and at the correct index

Index	Set 0			Set 1		
	Tag	Word 0	Word 1	Tag	Word 0	Word 1
0						
1	0x0000000	M[0x4]	M[0x6]			
2	0x0000008	M[0x108]	M[0x10a]			
3						
4	0x0000004	M[0x90]	M[0x92]	0x0000005	M[0xb0]	M[0xb2]
5						
6						
7	0x0000002	M[0x5c]	M[0x5e]	0x0000000	M[0x1c]	M[0x1e]

4 = 0x004
= 0b00000000000000000000000000000100

Block index = 1
Tag = 0x0000000

In Word 0

QUESTION 1(D) – FULLY ASSOCIATIVE CACHE

Memory address		Hit (H) or Miss (M)?
(in decimal)	(in hexadecimal)	
4	4	M (cold)
92	5C	M (cold)
7	7	H (same block as 0x4)
146	92	M (cold)
30	1E	M (conflict with 0x5c)
95	5F	M (conflict with 0x1e)
176	B0	M (cold)
93	5D	H (same word as 0x5c)
145	91	H (same block as 0x92)
264	108	M (cold)
6	6	H (same word as 0x7)

For fully associative caches, there is no index. All bits up to the block offset will be part of the tag. The cache is of the same size, hence can accommodate 16 blocks.

Memory address		Hit (H) or Miss (M)?
(in decimal)	(in hexadecimal)	
4	4	M (cold)
92	5C	M (cold)
7	7	H (same block as 0x4)
146	92	M (cold)
30	1E	M (cold)
95	5F	H (same block as 0x5c)
176	B0	M (cold)
93	5D	H (same word as 0x5c)
145	91	H (same block as 0x92)
264	108	M (cold)
6	6	H (same word as 0x7)

Tag value	Word 0	Word 1
0x0000001	M[0x4]	M[0x6]
0x0000017	M[0x5c]	M[0x5e]
0x0000024	M[0x90]	M[0x92]
0x0000007	M[0x1c]	M[0x1e]
0x000002c	M[0xb0]	M[0xb2]
0x0000042	M[0x108]	M[0x10a]

QUESTION 2(A)

```
start:
      la   $s0, A                #PC=0x100
      la   $s1, B                #PC=0x104
      li   $t0, 1                #PC=0x108
loop:
      slt  $t1, $t0, 1000        #PC=0x10c
      beq  $t1, $zero, end_loop  #PC=0x110
      sll  $t2, $t0, 2           #PC=0x114
      add  $t3, $s0, $t2         #PC=0x118
      lw   $a0, 0($t3)           #PC=0x11c
      add  $t4, $s1, $t2         #PC=0x120
      lw   $a1, 0($t4)           #PC=0x124
      add  $v0, $a0, $a1         #PC=0x128
      sw   $v0, -4($t3)          #PC=0x12c
      addi $t0, $t0, 1           #PC=0x130
      j    loop                  #PC=0x134
end_loop:
      .    .    .
```

Byte address

Word size = 4bytes

Direct-mapped cache, 4 blocks, 2 words per block

Index	Tag value	Word 0	Word 1
0			
1			
2			
3			

$t0 starts at 1. **$t2** = **$t0** * 4. So the first **lw** is from address **$t3** = **$s0** + **$t2** = **0x1004**. Next **lw** is from **$t4** = **$s1** + **$t2** = **0x4014**. This is then followed by a **sw** to **-4($t3)** = **0x1000**. This completes one iteration and **$t0** is incremented by 1. So we have the following sequence:

```
start:
      la   $s0, A                 #PC=0x100
      la   $s1, B                 #PC=0x104
      li   $t0, 1                 #PC=0x108
loop:
      slt  $t1, $t0, 1000         #PC=0x10c
      beq  $t1, $zero, end_loop   #PC=0x110
      sll  $t2, $t0, 2            #PC=0x114
      add  $t3, $s0, $t2          #PC=0x118
      lw   $a0, 0($t3)            #PC=0x11c
      add  $t4, $s1, $t2          #PC=0x120
      lw   $a1, 0($t4)            #PC=0x124
      add  $v0, $a0, $a1          #PC=0x128
      sw   $v0, -4($t3)           #PC=0x12c
      addi $t0, $t0, 1            #PC=0x130
      j    loop                   #PC=0x134
end_loop:
      .   .   .
```

```
lw from 0x1004 - cold miss, index = 0, tag = 0x80
lw from 0x4014 - cold miss, index = 2, tag = 0x200
sw to   0x1000 - index = 0, tag = 0x80, HIT!
lw from 0x1008 - cold miss, index = 1, tag = 0x80
lw from 0x4018 - cold miss, index = 3, tag = 0x200
sw to   0x1004 - index = 0, tag = 0x80, HIT!
lw from 0x100c - index = 1, tag = 0x80, HIT!
lw from 0x401c - index = 3, tag = 0x200, HIT!
sw to   0x1008 - index = 1, tag = 0x80, HIT!
lw from 0x1010 - conflict miss, index = 2, tag = 0x80
```

QUESTION 2(B)

The trick is to work backwards. The loop goes from iteration 1 to 999 (inclusive). At iteration *i*, the 3 accesses are to:

`lw` from **`0x1000`** + 4*i*
`lw` from **`0x4010`** + 4*i*
`sw` to **`0x1000`** + 4(*i*-1)

In particular, for iteration 999, they are:

1. **`lw`** from **`0x1000`** + 4 * 999 = **`0x1f9c`** (index = 3, tag = **`0xfc`**)
2. **`lw`** from **`0x4010`** + 4 * 999 = **`0x4fac`** (index = 1, tag = **`0x27d`**)
3. **`sw`** to **`0x1000`** + 4 * 998 = **`0x1f98`** (index = 3, tag = **`0xfc`**)

We can fill these into the final content of the cache. Since this is the last iteration, we can be sure that it has displace whatever was in the cache index and remained there till the end.

Index	Tag value	Word 0	Word 1
0			
1	0x27d	M[0x4fa8]	M[0x4fac]
2			
3	0xfc	M[0x1f98]	M[0x1f9c]

For iteration 998, they are:

1. **`lw`** from **`0x1000`** + 4 * 998 = **`0x1f98`** (index = 3, tag = **`0xfc`**)
2. **`lw`** from **`0x4010`** + 4 * 998 = **`0x4fa8`** (index = 1, tag = **`0x27d`**)
3. **`sw`** to **`0x1000`** + 4 * 997 = **`0x1f94`** (index = 2, tag = **`0xfc`**)

Index	Tag value	Word 0	Word 1
0			
1	0x27d	M[0x4fa8]	M[0x4fac]
2	0xfc	M[0x1f90]	M[0x1f94]
3	0xfc	M[0x1f98]	M[0x1f9c]

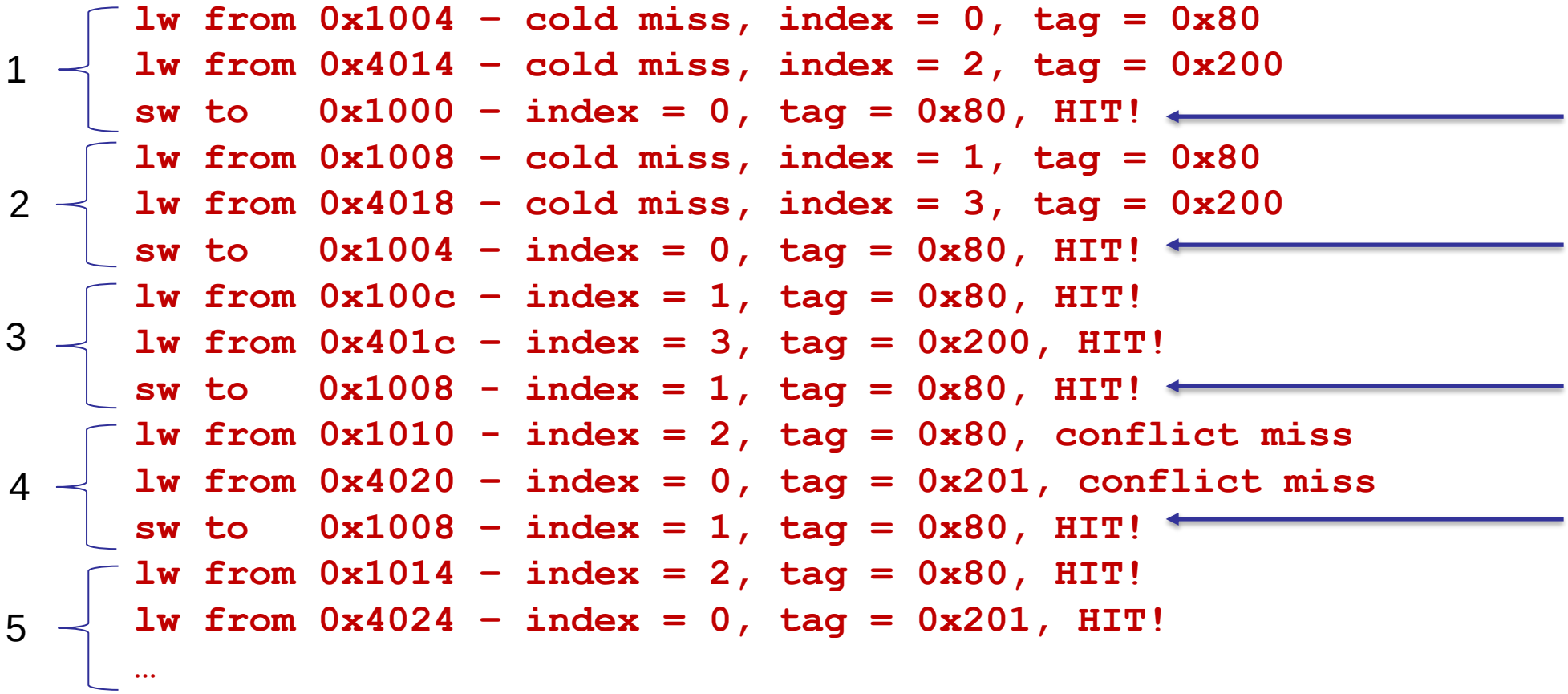
For iteration 997, they are:

1. **`lw`** from **`0x1000`** + 4 * 997 = **`0x1f94`** (index = 2, tag = **`0xfc`**)
2. **`lw`** from **`0x4010`** + 4 * 997 = **`0x4fa4`** (index = 0, tag = **`0x27d`**)
3. **`sw`** to **`0x1000`** + 4 * 996 = **`0x1f98`** (index = 3, tag = **`0xfc`**)

Index	Tag value	Word 0	Word 1
0	0x27d	M[0x4fa0]	M[0x4fa4]
1	0x27d	M[0x4fa8]	M[0x4fac]
2	0xfc	M[0x1f90]	M[0x1f94]
3	0xfc	M[0x1f98]	M[0x1f9c]

This would be the final state of the cache because we are working backwards through the iterations, these would be what remains, having displaced the earlier blocks in the same index.

QUESTION 2(C)



Iteration	1	2	3	4	5	6	7	8	9
Block for A	0	1	1	2	2	3	3	0	0
Block for B	2	3	3	0	0	1	1	2	2

Total number of data cache memory references = 3 * 999 = 2997.

After iteration 1, every other iteration hits on both lw. sw always hits.

Total number of hits = (998 / 2) * 2 + 999 = 1997.

After iteration 1, every other iteration misses on both lw. In iteration 1, both lw misses.

Total number of misses = (998 / 2 + 1) * 2 = 1000 = 2997 - 1997.

Miss rate = 1000/2997 = 33.37%.

QUESTION 3(A)

Working backwards, we get the following trace:

PC = 0x110 (index = 2, tag = 0x8)
PC = 0x10c (index = 1, tag = 0x8)
PC = 0x134 (index = 2, tag = 0x9)
PC = 0x130 (index = 2, tag = 0x9)
PC = 0x12c (index = 1, tag = 0x9)
PC = 0x128 (index = 1, tag = 0x9)
PC = 0x124 (index = 0, tag = 0x9)
PC = 0x120 (index = 0, tag = 0x9)
PC = 0x11c (index = 3, tag = 0x8)
PC = 0x118 (index = 3, tag = 0x8)
PC = 0x114 (index = 2, tag = 0x8)
PC = 0x110 (index = 2, tag = 0x8) – (repeat)

```
start:
      la   $s0, A                 #PC=0x100
      la   $s1, B                 #PC=0x104
      li   $t0, 1                 #PC=0x108
loop:
      slt  $t1, $t0, 1000         #PC=0x10c
      beq  $t1, $zero, end_loop   #PC=0x110
      sll  $t2, $t0, 2            #PC=0x114
      add  $t3, $s0, $t2          #PC=0x118
      lw   $a0, 0($t3)            #PC=0x11c
      add  $t4, $s1, $t2          #PC=0x120
      lw   $a1, 0($t4)            #PC=0x124
      add  $v0, $a0, $a1          #PC=0x128
      sw   $v0, -4($t3)           #PC=0x12c
      addi $t0, $t0, 1            #PC=0x130
      j    loop                   #PC=0x134
end_loop:
      .    .    .
```

Index	Tag value	Word 0	Word 1
0	0x9	PC=0x120	PC=0x124
1	0x8	PC=0x108	PC=0x10c
2	0x8	PC=0x110	PC=0x114
3	0x8	PC=0x118	PC=0x11c

QUESTION 3(B)

General idea: We work forward in the execution until we get to a steady state.

```
start:
        la   $s0, A                    #PC=0x100
        la   $s1, B                    #PC=0x104
        li   $t0, 1                    #PC=0x108
loop:
        slt  $t1, $t0, 1000            #PC=0x10c
        beq  $t1, $zero, end_loop      #PC=0x110
        sll  $t2, $t0, 2               #PC=0x114
        add  $t3, $s0, $t2             #PC=0x118
        lw   $a0, 0($t3)               #PC=0x11c
        add  $t4, $s1, $t2             #PC=0x120
        lw   $a1, 0($t4)               #PC=0x124
        add  $v0, $a0, $a1             #PC=0x128
        sw   $v0, -4($t3)              #PC=0x12c
        addi $t0, $t0, 1               #PC=0x130
        j    loop                      #PC=0x134
end_loop:
        .    .    .
```

Block 0

Block 1

Block 2

Block 3

Block 0

Block 1

Block 2

PC	Block	H/M
0x100	0	M
0x104	0	H
0x108	1	M

PC	Block	H/M
0x10c	1	H
0x110	2	M
0x114	2	H
0x118	3	M
0x11c	3	H
0x120	0	M
0x124	0	H
0x128	1	M
0x12c	1	H
0x130	2	M
0x134	2	H

General idea: We work forward in the execution until we get to a steady state.

```
start:
        la   $s0, A                    #PC=0x100
        la   $s1, B                    #PC=0x104
        li   $t0, 1                    #PC=0x108
loop:
        slt  $t1, $t0, 1000            #PC=0x10c
        beq  $t1, $zero, end_loop      #PC=0x110
        sll  $t2, $t0, 2               #PC=0x114
        add  $t3, $s0, $t2             #PC=0x118
        lw   $a0, 0($t3)               #PC=0x11c
        add  $t4, $s1, $t2             #PC=0x120
        lw   $a1, 0($t4)               #PC=0x124
        add  $v0, $a0, $a1             #PC=0x128
        sw   $v0, -4($t3)              #PC=0x12c
        addi $t0, $t0, 1               #PC=0x130
        j    loop                      #PC=0x134
end_loop:
        .    .    .
```

Block 0

Block 1

Block 2

Block 3

Block 0

Block 1

Block 2

PC	Block	H/M
0x10c	1	M
0x110	2	M
0x114	2	H
0x118	3	H
0x11c	3	H
0x120	0	H
0x124	0	H
0x128	1	M
0x12c	1	H
0x130	2	M
0x134	2	H

PC	Block	H/M
0x100	0	M
0x104	0	H
0x108	1	M

PC	Block	H/M
0x10c	1	H
0x110	2	M
0x114	2	H
0x118	3	M
0x11c	3	H
0x120	0	M
0x124	0	H
0x128	1	M
0x12c	1	H
0x130	2	M
0x134	2	H

PC	Block	H/M
0x10c	1	M
0x110	2	M
0x114	2	H
0x118	3	H
0x11c	3	H
0x120	0	H
0x124	0	H
0x128	1	M
0x12c	1	H
0x130	2	M
0x134	2	H

Total number of accesses

= 3 + 11(999) + 2 = 10994

Total number of misses

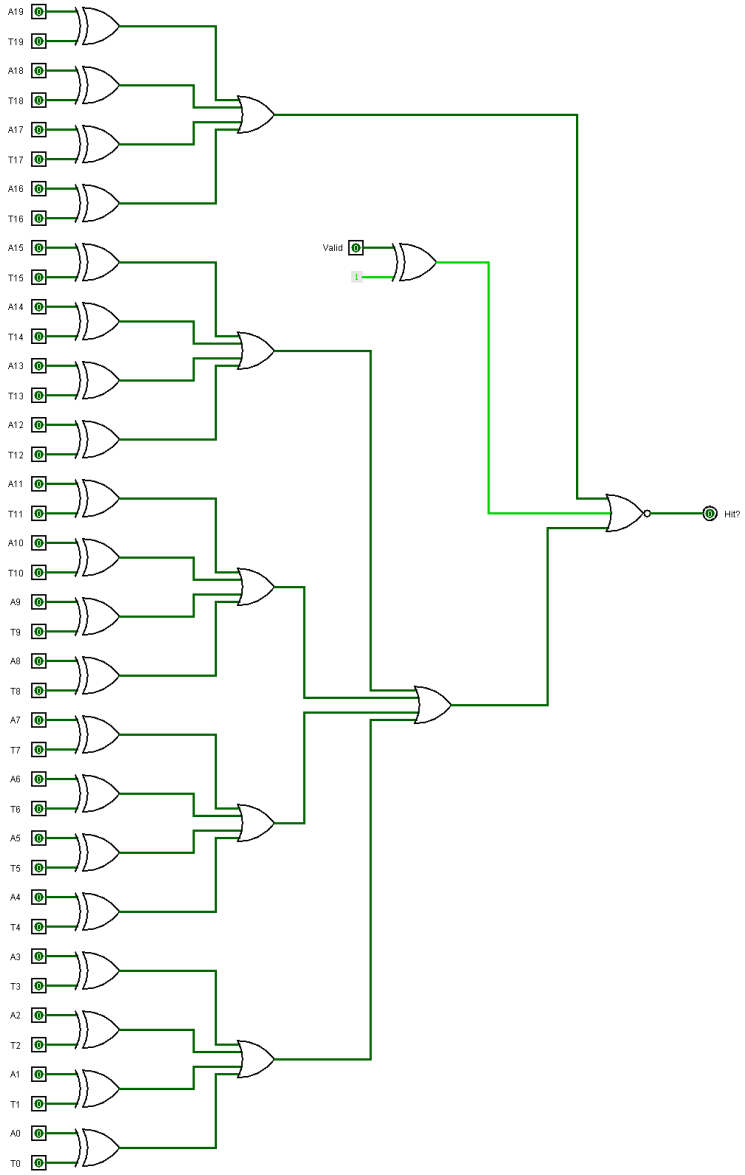
= 2 + 5 + 4(998) + 2 = 4001

Miss rate

= 4001/10994 = 36.39%

QUESTION 4

1. Use XOR for comparing address and tag bits.
2. Use XOR for comparing valid bit to 1.
3. If the output of all of the XOR gates are 0, then cache hit. Otherwise miss.
4. Use tree of OR gates to check if the output of any XOR gate is 1.
5. Invert the final output to give 1 if all of the XOR gates output 0.



Since this was not asked in the tutorial…

ADDITIONAL QUESTION

Using the information provided in Question 2c (same MIPS program, addresses, system and cache configurations), given the following information:

Hit time = 1ns

Miss penalty = 16ns

What is the average access time?

Using the information provided in Question 2c (same MIPS program, addresses, system and cache configurations), given the following information:

Hit time = 1ns

Miss penalty = 16ns

What is the average access time?

Miss rate = 1000/2997 = 33.37%

Hit rate = 1997/2997 = 66.63%
Miss rate = 0.6663 + 16(0.3337) = 6.01ns
Note that 16ns here is miss penalty and not DRAM access time (which was given in the lecture notes example).