New Zealand

COMPSCI 210 – Mock Test

Discussion

THE UNIVERSITY OF AUCKLAND

NEW ZEALAND

Te Whare Wānanga o Tāmaki Makaurau

Mano Manoharan, Room 723.315, Tamaki Campus
mano@cs.auckland.ac.nz

The University of Auckland

Byte Address vs. Word Address

- Memory is a large single-dimension array of bytes.
- A memory *address* is an index into the array
- *Byte addressing* means that the index points to a byte of memory.
- *Words* are the smallest units of data we get out of/into memory.
  - A word could be 2 bytes, 4 bytes, or more, depending on the CPU.

0	8 bits of data
1	8 bits of data
2	8 bits of data
3	8 bits of data
…	8 bits of data

Byte Address vs. Word Address

- A word is 32 bits or 4 bytes in a 32-bit CPU.
  - 2^{32} bytes with byte addresses from 0 to $2^{32}-1$
  - 2^{30} words with byte addresses 0, 4, 8, ... $2^{32}-4$
- Similarly, in a 16-bit CPU (e.g. LC-3), a word is 16 bits

0	8 bits of data
1	8 bits of data
2	8 bits of data
3	8 bits of data
…	8 bits of data

0	32 bits of data
4	32 bits of data
8	32 bits of data
12	32 bits of data
…	32 bits of data

0	16 bits of data
2	16 bits of data
4	16 bits of data
6	16 bits of data
…	16 bits of data

Byte Address vs. Word Address

- Byte Address is the fundamental address in a memory system, and is the address of a byte.
- Word Address is the address of a word.
  - Not all byte addresses are valid word addresses.

0	8 bits of data
1	8 bits of data
2	8 bits of data
3	8 bits of data
…	8 bits of data

0	32 bits of data
4	32 bits of data
8	32 bits of data
12	32 bits of data
…	32 bits of data

0	16 bits of data
2	16 bits of data
4	16 bits of data
6	16 bits of data
…	16 bits of data

Executing Java Bytecode

Consider a Java method *GetMax* which takes two parameters: an integer array *a*, and an integer *sz* that denotes the size of the array *a*. It returns the value of the largest element of the array, assuming the elements are all positive values. The method is defined as follows.

```
static int GetMax(int a[], int sz)
{
   int max = 0;
   for ( int i = 0; i < sz; ++i ) {
      if ( a[i] > max ) {
         max = a[i];
      }
   }
   return max;
} // GetMax
```

Executing Java Bytecode

Consider now the Java bytecode sequence for the method *GetMax*.

```
0:    iconst_0
1:    istore_2
2:    iconst_0
3:    istore_3
4:    goto          21
7:    aload_0
8:    iload_3
9:    iaload
10:   iload_2
11:   icmple        18   // compare if less than or equal, if so go to label 18
14:   aload_0
15:   iload_3
16:   iaload
17:   istore_2
18:   iinc          3  1   // increment local variable #3 by 1
21:   iload_3
22:   iload_1
23:   icmplt        7    // compare if less than, if so go to label 7
26:   iload_2
27:   ireturn
```

Executing Java Bytecode

The local variables for the frame allocated for the method are as follows

Index	Variable
0	*a*
1	*sz*
2	*max*
3	*i*

Assume for an invocation of this method the size of the array is 50. The following six questions relate to this invocation. For these questions, the integer comparison operation at label 23 will return true, so that the branch to label 7 will take place. Also, assume that branch at label 11 always returns false so that branch to label 18 will not take place.

Executing Java Bytecode

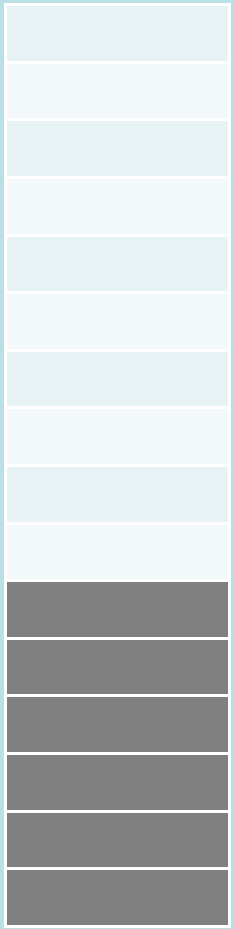
0	iconst_0		
1	istore_2		
2	iconst_0		
3	istore_3		
4	goto	21	
7	aload_0		
8	iload_3		
9	iaload		
10	iload_2		
11	icmple	18	
14	aload_0		
15	iload_3		
16	iaload		
17	istore_2		
18	iinc	3	1
21	iload_3		
22	iload_1		
23	icmplt	7	
26	iload_2		
27	ireturn		

Frame

Local variables

Index	Variable
0	a
1	sz
2	max
3	i

Operand stack



Executing Java Bytecode

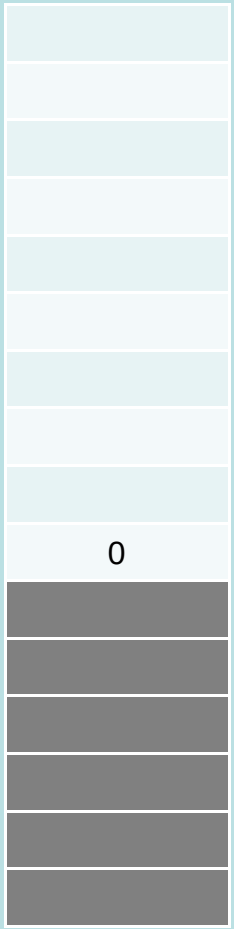
0	iconst_0	← PC
1	istore_2	
2	iconst_0	
3	istore_3	
4	goto 21	
7	aload_0	
8	iload_3	
9	iaload	
10	iload_2	
11	icmple 18	
14	aload_0	
15	iload_3	
16	iaload	
17	istore_2	
18	iinc 3 1	
21	iload_3	
22	iload_1	
23	icmplt 7	
26	iload_2	
27	ireturn	

Frame

Local variables

Index	Variable
0	a
1	sz
2	max
3	i

Operand stack



Executing Java Bytecode

0	iconst_0	
1	istore_2	← PC
2	iconst_0	
3	istore_3	
4	goto	21
7	aload_0	
8	iload_3	
9	iaload	
10	iload_2	
11	icmple	18
14	aload_0	
15	iload_3	
16	iaload	
17	istore_2	
18	iinc	3 1
21	iload_3	
22	iload_1	
23	icmplt	7
26	iload_2	
27	ireturn	

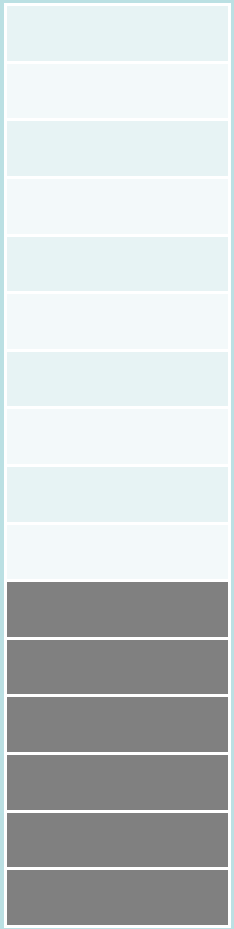
Frame

Local variables

Index	Variable
0	a
1	sz
2	max
3	i

max= 0

Operand stack



Executing Java Bytecode

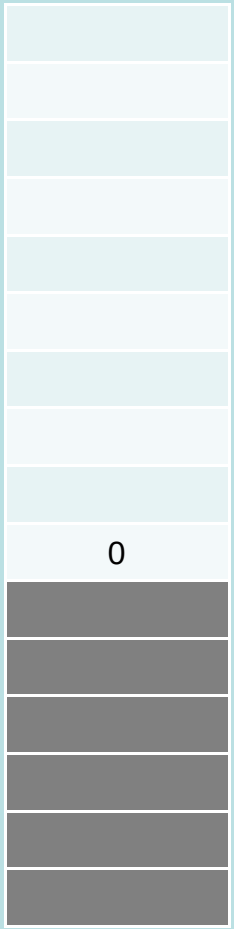
0	iconst_0	
1	istore_2	
2	iconst_0	← PC
3	istore_3	
4	goto	21
7	aload_0	
8	iload_3	
9	iaload	
10	iload_2	
11	icmple	18
14	aload_0	
15	iload_3	
16	iaload	
17	istore_2	
18	iinc	3 1
21	iload_3	
22	iload_1	
23	icmplt	7
26	iload_2	
27	ireturn	

Frame

Local variables

Index	Variable
0	a
1	sz
2	max
3	i

Operand stack



Executing Java Bytecode

0	iconst_0	
1	istore_2	
2	iconst_0	
3	istore_3	← PC
4	goto 21	
7	aload_0	
8	iload_3	
9	iaload	
10	iload_2	
11	icmple 18	
14	aload_0	
15	iload_3	
16	iaload	
17	istore_2	
18	iinc 3 1	
21	iload_3	
22	iload_1	
23	icmplt 7	
26	iload_2	
27	ireturn	

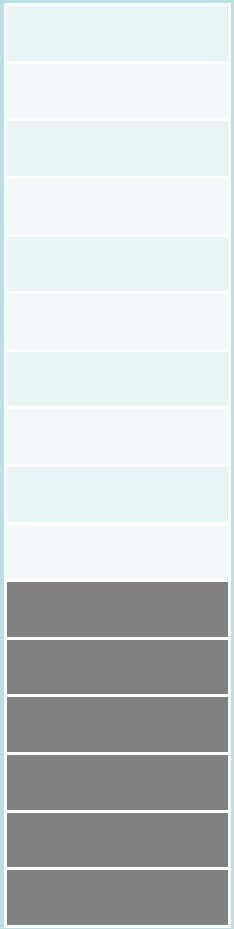
Frame

Local variables

Index	Variable
0	a
1	sz
2	max
3	i

i = 0

Operand stack



Executing Java Bytecode

0	iconst_0
1	istore_2
2	iconst_0
3	istore_3
4	goto 21
7	aload_0
8	iload_3
9	iaload
10	iload_2
11	icmple 18
14	aload_0
15	iload_3
16	iaload
17	istore_2
18	iinc 3 1
21	iload_3
22	iload_1
23	icmplt 7
26	iload_2
27	ireturn

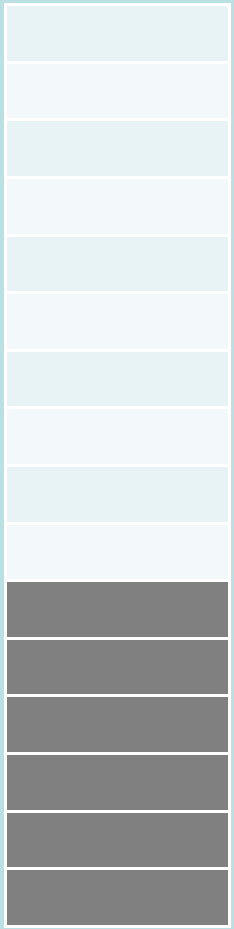
PC

Frame

Local variables

Index	Variable
0	a
1	sz
2	max
3	i

Operand stack



Executing Java Bytecode

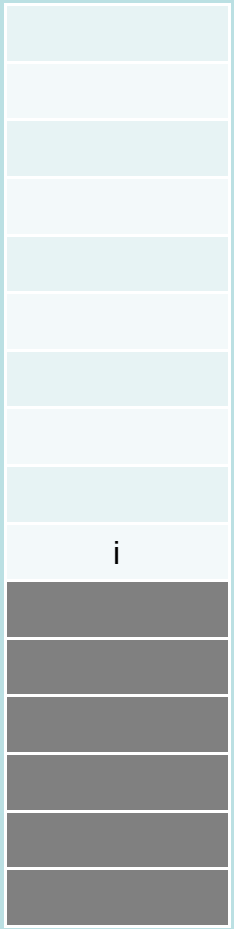
0	iconst_0	
1	istore_2	
2	iconst_0	
3	istore_3	
4	goto	21
7	aload_0	
8	iload_3	
9	iaload	
10	iload_2	
11	icmple	18
14	aload_0	
15	iload_3	
16	iaload	
17	istore_2	
18	iinc	3 1
21	iload_3	← PC
22	iload_1	
23	icmplt	7
26	iload_2	
27	ireturn	

Frame

Local variables

Index	Variable
0	a
1	sz
2	max
3	i

Operand stack



Executing Java Bytecode

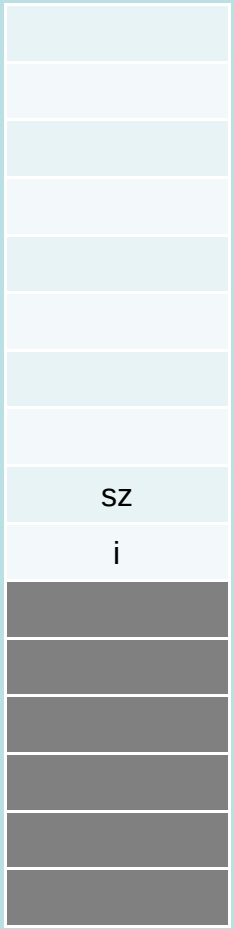
0	iconst_0		
1	istore_2		
2	iconst_0		
3	istore_3		
4	goto	21	
7	aload_0		
8	iload_3		
9	iaload		
10	iload_2		
11	icmple	18	
14	aload_0		
15	iload_3		
16	iaload		
17	istore_2		
18	iinc	3	1
21	iload_3		
22	iload_1		← PC
23	icmplt	7	
26	iload_2		
27	ireturn		

Frame

Local variables

Index	Variable
0	a
1	sz
2	max
3	i

Operand stack



Executing Java Bytecode

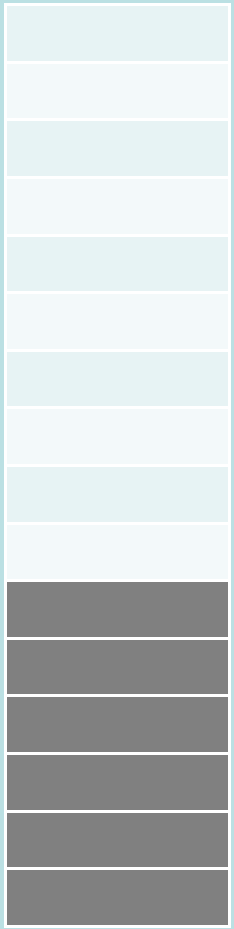
0	iconst_0		
1	istore_2		
2	iconst_0		
3	istore_3		
4	goto	21	
7	aload_0		
8	iload_3		
9	iaload		
10	iload_2		
11	icmple	18	
14	aload_0		
15	iload_3		
16	iaload		
17	istore_2		
18	iinc	3	1
21	iload_3		
22	iload_1		
23	icmplt	7	← PC
26	iload_2		
27	ireturn		

Frame

Local variables

Index	Variable
0	a
1	sz
2	max
3	i

Operand stack



if (i < sz) goto 7

Executing Java Bytecode

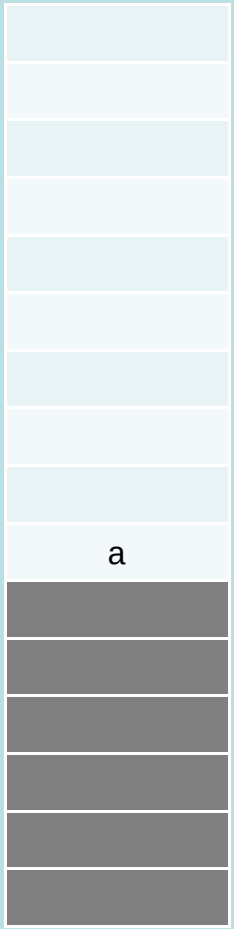
0	iconst_0	
1	istore_2	
2	iconst_0	
3	istore_3	
4	goto	21
7	aload_0	← PC
8	iload_3	
9	iaload	
10	iload_2	
11	icmple	18
14	aload_0	
15	iload_3	
16	iaload	
17	istore_2	
18	iinc	3 1
21	iload_3	
22	iload_1	
23	icmplt	7
26	iload_2	
27	ireturn	

Frame

Local variables

Index	Variable
0	a
1	sz
2	max
3	i

Operand stack



Executing Java Bytecode

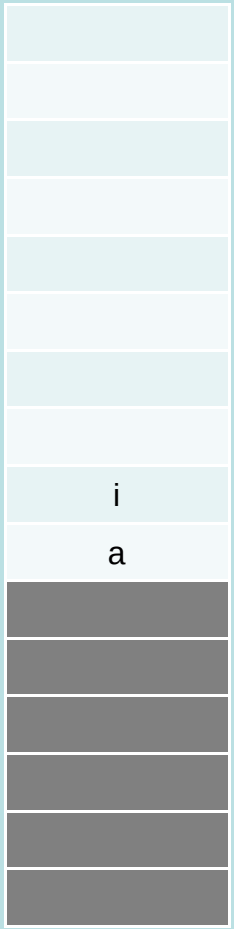
0	iconst_0	
1	istore_2	
2	iconst_0	
3	istore_3	
4	goto	21
7	aload_0	
8	iload_3	← PC
9	iaload	
10	iload_2	
11	icmple	18
14	aload_0	
15	iload_3	
16	iaload	
17	istore_2	
18	iinc	3 1
21	iload_3	
22	iload_1	
23	icmplt	7
26	iload_2	
27	ireturn	

Frame

Local variables

Index	Variable
0	a
1	sz
2	max
3	i

Operand stack



Executing Java Bytecode

0	iconst_0	
1	istore_2	
2	iconst_0	
3	istore_3	
4	goto	21
7	aload_0	
8	iload_3	
9	iaload	← PC
10	iload_2	
11	icmple	18
14	aload_0	
15	iload_3	
16	iaload	
17	istore_2	
18	iinc	3 1
21	iload_3	
22	iload_1	
23	icmplt	7
26	iload_2	
27	ireturn	

Frame

Local variables

Index	Variable
0	a
1	sz
2	max
3	i

Operand stack

Executing Java Bytecode

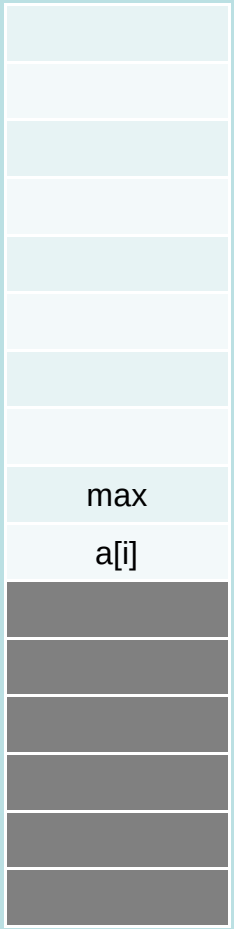
0	iconst_0	
1	istore_2	
2	iconst_0	
3	istore_3	
4	goto	21
7	aload_0	
8	iload_3	
9	iaload	
10	iload_2	← PC
11	icmple	18
14	aload_0	
15	iload_3	
16	iaload	
17	istore_2	
18	iinc	3 1
21	iload_3	
22	iload_1	
23	icmplt	7
26	iload_2	
27	ireturn	

Frame

Local variables

Index	Variable
0	a
1	sz
2	max
3	i

Operand stack



Executing Java Bytecode

0	iconst_0	
1	istore_2	
2	iconst_0	
3	istore_3	
4	goto	21
7	aload_0	
8	iload_3	
9	iaload	
10	iload_2	
11	icmple	18
14	aload_0	
15	iload_3	
16	iaload	
17	istore_2	
18	iinc	3 1
21	iload_3	
22	iload_1	
23	icmplt	7
26	iload_2	
27	ireturn	

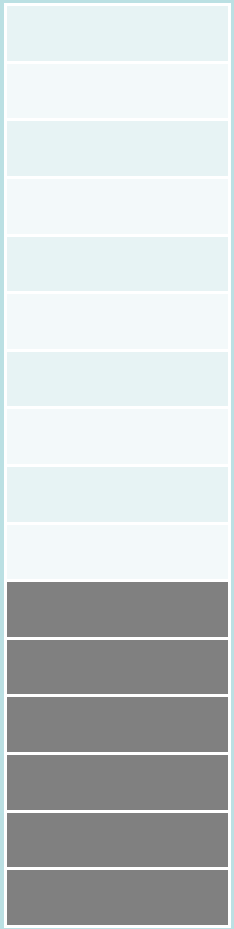
PC

Frame

Local variables

Index	Variable
0	a
1	sz
2	max
3	i

Operand stack



if (max <= a[i])
goto 18

Executing Java Bytecode

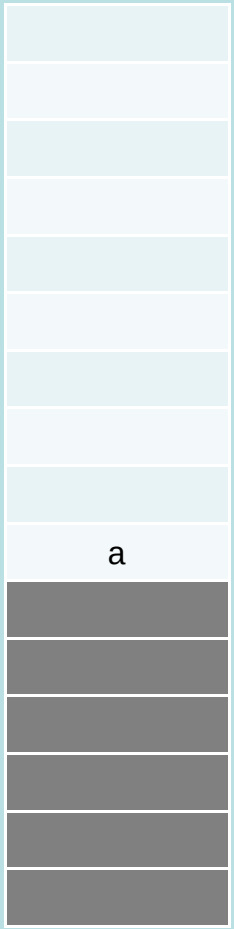
0	iconst_0	
1	istore_2	
2	iconst_0	
3	istore_3	
4	goto	21
7	aload_0	
8	iload_3	
9	iaload	
10	iload_2	
11	icmple	18
14	aload_0	← PC
15	iload_3	
16	iaload	
17	istore_2	
18	iinc	3 1
21	iload_3	
22	iload_1	
23	icmplt	7
26	iload_2	
27	ireturn	

Frame

Local variables

Index	Variable
0	a
1	sz
2	max
3	i

Operand stack



Executing Java Bytecode

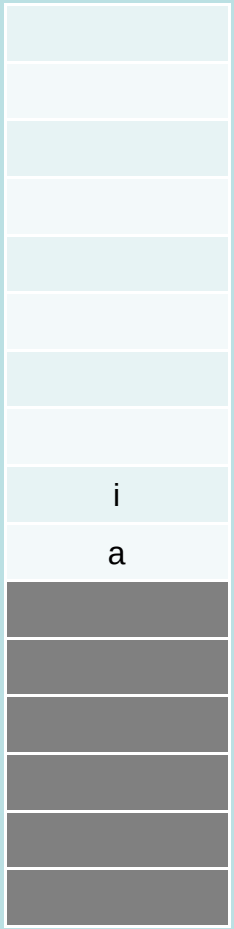
0	iconst_0		
1	istore_2		
2	iconst_0		
3	istore_3		
4	goto	21	
7	aload_0		
8	iload_3		
9	iaload		
10	iload_2		
11	icmple	18	
14	aload_0		
15	iload_3		← PC
16	iaload		
17	istore_2		
18	iinc	3	1
21	iload_3		
22	iload_1		
23	icmplt	7	
26	iload_2		
27	ireturn		

Frame

Local variables

Index	Variable
0	a
1	sz
2	max
3	i

Operand stack



Executing Java Bytecode

0	iconst_0		
1	istore_2		
2	iconst_0		
3	istore_3		
4	goto	21	
7	aload_0		
8	iload_3		
9	iaload		
10	iload_2		
11	icmple	18	
14	aload_0		
15	iload_3		
16	iaload		← PC
17	istore_2		
18	iinc	3	1
21	iload_3		
22	iload_1		
23	icmplt	7	
26	iload_2		
27	ireturn		

Frame

Local variables

Index	Variable
0	a
1	sz
2	max
3	i

Operand stack

Executing Java Bytecode

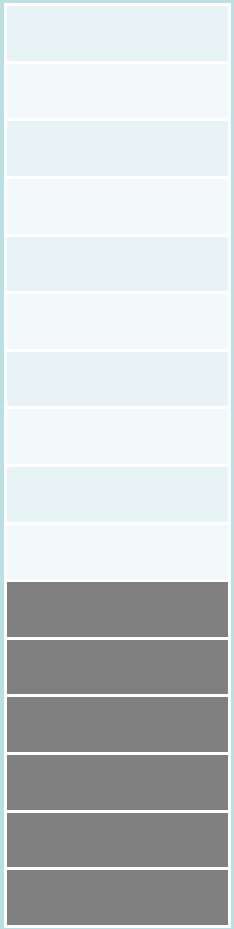
0	iconst_0		
1	istore_2		
2	iconst_0		
3	istore_3		
4	goto	21	
7	aload_0		
8	iload_3		
9	iaload		
10	iload_2		
11	icmple	18	
14	aload_0		
15	iload_3		
16	iaload		
17	istore_2		← PC
18	iinc	3	1
21	iload_3		
22	iload_1		
23	icmplt	7	
26	iload_2		
27	ireturn		

Frame

Local variables

Index	Variable
0	a
1	sz
2	max
3	i

Operand stack



max = a[i]

Executing Java Bytecode

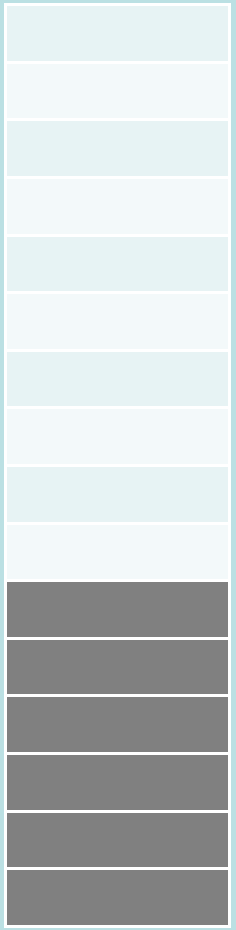

```
0   iconst_0
1   istore_2
2   iconst_0
3   istore_3
4   goto    21
7   aload_0
8   iload_3
9   iaload
10  iload_2
11  icmple  18
14  aload_0
15  iload_3
16  iaload
17  istore_2
18  iinc    3   1   <- PC
21  iload_3
22  iload_1
23  icmplt  7
26  iload_2
27  ireturn
```

Frame

Local variables

Index	Variable
0	a
1	sz
2	max
3	i

Operand stack



i = i + 1

Executing Java Bytecode

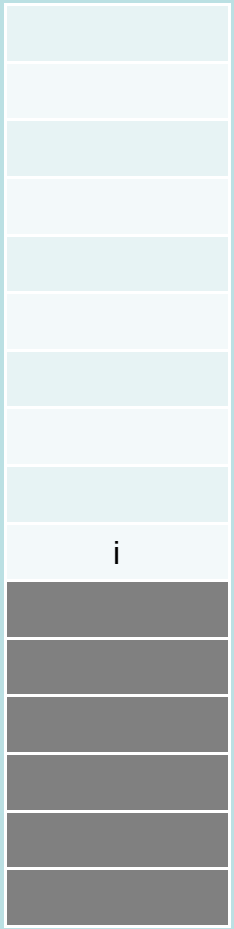
0	iconst_0	
1	istore_2	
2	iconst_0	
3	istore_3	
4	goto	21
7	aload_0	
8	iload_3	
9	iaload	
10	iload_2	
11	icmple	18
14	aload_0	
15	iload_3	
16	iaload	
17	istore_2	
18	iinc	3 1
21	iload_3	← PC
22	iload_1	
23	icmplt	7
26	iload_2	
27	ireturn	

Frame

Local variables

Index	Variable
0	a
1	sz
2	max
3	i

Operand stack



Executing Java Bytecode

0	iconst_0		
1	istore_2		
2	iconst_0		
3	istore_3		
4	goto	21	
7	aload_0		
8	iload_3		
9	iaload		
10	iload_2		
11	icmple	18	
14	aload_0		
15	iload_3		
16	iaload		
17	istore_2		
18	iinc	3	1
21	iload_3		
22	iload_1		
23	icmplt	7	
26	iload_2		
27	ireturn		

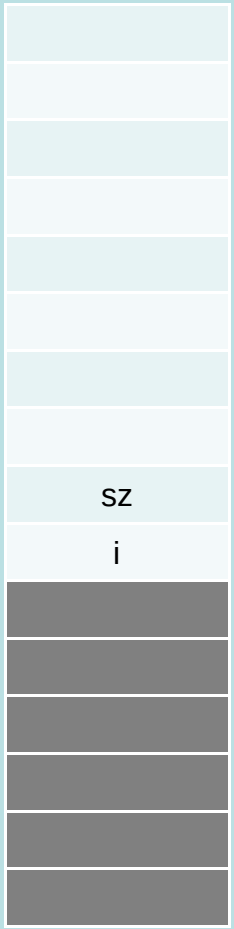
PC

Frame

Local variables

Index	Variable
0	a
1	sz
2	max
3	i

Operand stack



Executing Java Bytecode

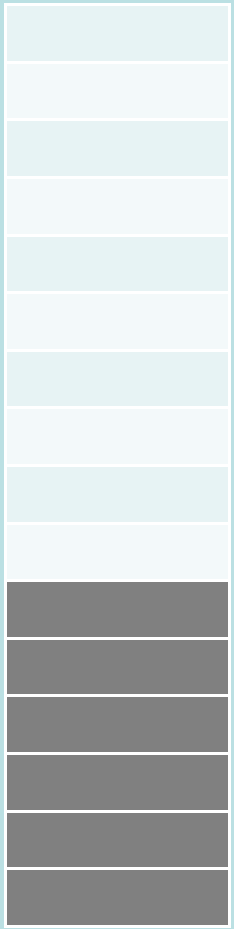
0	iconst_0		
1	istore_2		
2	iconst_0		
3	istore_3		
4	goto	21	
7	aload_0		
8	iload_3		
9	iaload		
10	iload_2		
11	icmple	18	
14	aload_0		
15	iload_3		
16	iaload		
17	istore_2		
18	iinc	3	1
21	iload_3		
22	iload_1		
23	icmplt	7	← PC
26	iload_2		
27	ireturn		

Frame

Local variables

Index	Variable
0	a
1	sz
2	max
3	i

Operand stack



if (i < sz) goto 7

Executing Java Bytecode

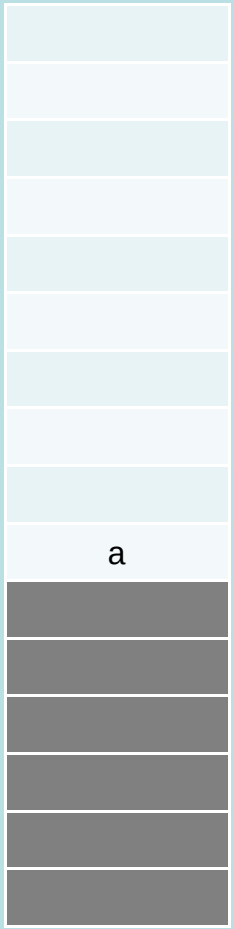
0	iconst_0	
1	istore_2	
2	iconst_0	
3	istore_3	
4	goto	21
7	aload_0	← PC
8	iload_3	
9	iaload	
10	iload_2	
11	icmple	18
14	aload_0	
15	iload_3	
16	iaload	
17	istore_2	
18	iinc	3 1
21	iload_3	
22	iload_1	
23	icmplt	7
26	iload_2	
27	ireturn	

Frame

Local variables

Index	Variable
0	a
1	sz
2	max
3	i

Operand stack



Decimal to 32-bit Floating Point

- Consider the decimal value 8.125_{10}.
  - This in binary is 1000.001_2.
  - Normalizing this gives us 1.000001×2^3.

		Integral part
0.125 x 2	0.250	0
0.250 x 2	0.5	0
0.5 x 2	0	1

- The sign is positive, so S = 0
- Exponent is 3 + bias (127) = $130_{10} = 10000010_2$.
- Mantissa is 000001 (leaving out the hidden 1 and the point).

- So we have 0 10000010 00000100000…0

32-bit Floating Point to Decimal

- Consider the floating point representation $4123ABCD_{16}$.
  - This is 0 10000010 $01000111010101111001101_2$.
- Sign is 0 (positive)
- Exponent is $10000010_2 = 130_{10}$; 130 − 127 = 3.
- Thus, we have 1. $\underline{01000111010101111001101_2} \times 2^3$.
- Denormalizing, we have 1 $\underline{010.01110101011110011012}$.
- From this we get $1x\ 2^3 + 0x\ 2^2 + 1x\ 2^1 + 0x\ 2^0 + 0x\ 2^{-1} + 0x\ 2^{-2} + 1x\ 2^{-3} + 1x\ 2^{-4} + 1x\ 2^{-5} + \ldots = 10.2294$

Direct-Mapped Caches

Consider a direct-mapped cache of size 4. Each slot in the cache can have just one item (i.e. the line size is 1 item). The cache is empty to start with.

Work out if the following accesses to the given addresses are hits or misses. Each access is numbered with a sequence id for your convenience.

Sequence Id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Address	3	6	3	1	9	1	9	1	9	3	4	5	6	9	3	4	5	6	9	3
Hit/Miss																				

Direct-Mapped Caches

Slot #	Address	Item
0		
1		
2		
3		

Sequence Id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Address	3	6	3	1	9	1	9	1	9	3	4	5	6	9	3	4	5	6	9	3
Hit/Miss																				

Direct-Mapped Caches

Slot #	Address	Item
0		
1		
2		
3	3	

Sequence Id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Address	3	6	3	1	9	1	9	1	9	3	4	5	6	9	3	4	5	6	9	3
Hit/Miss	M																			

Direct-Mapped Caches

Slot #	Address	Item
0		
1		
2	6	
3	3	

Sequence Id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Address	3	6	3	1	9	1	9	1	9	3	4	5	6	9	3	4	5	6	9	3
Hit/Miss	M	M																		

Direct-Mapped Caches

Slot #	Address	Item
0		
1		
2	6	
3	3	

Sequence Id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Address	3	6	3	1	9	1	9	1	9	3	4	5	6	9	3	4	5	6	9	3
Hit/Miss	M	M	H																	

Direct-Mapped Caches

Slot #	Address	Item
0		
1	1	
2	6	
3	3	

Sequence Id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Address	3	6	3	1	9	1	9	1	9	3	4	5	6	9	3	4	5	6	9	3
Hit/Miss	M	M	H	M																

Direct-Mapped Caches

Slot #	Address	Item
0		
1	~~1~~ 9	
2	6	
3	3	

Sequence Id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Address	3	6	3	1	9	1	9	1	9	3	4	5	6	9	3	4	5	6	9	3
Hit/Miss	M	M	H	M	M															

Direct-Mapped Caches

Slot #	Address	Item
0		
1	~~9~~ 1	
2	6	
3	3	

Sequence Id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Address	3	6	3	1	9	1	9	1	9	3	4	5	6	9	3	4	5	6	9	3
Hit/Miss	M	M	H	M	M	M														

Direct-Mapped Caches

Slot #	Address	Item
0		
1	~~1~~ 9	
2	6	
3	3	

Sequence Id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Address	3	6	3	1	9	1	9	1	9	3	4	5	6	9	3	4	5	6	9	3
Hit/Miss	M	M	H	M	M	M	M													

Direct-Mapped Caches

Slot #	Address	Item
0		
1	~~9~~ 1	
2	6	
3	3	

Sequence Id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Address	3	6	3	1	9	1	9	1	9	3	4	5	6	9	3	4	5	6	9	3
Hit/Miss	M	M	H	M	M	M	M	M												

Direct-Mapped Caches

Slot #	Address	Item
0		
1	~~1~~ 9	
2	6	
3	3	

Sequence Id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Address	3	6	3	1	9	1	9	1	9	3	4	5	6	9	3	4	5	6	9	3
Hit/Miss	M	M	H	M	M	M	M	M	M											

Direct-Mapped Caches

Slot #	Address	Item
0		
1	9	
2	6	
3	3	

Sequence Id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Address	3	6	3	1	9	1	9	1	9	3	4	5	6	9	3	4	5	6	9	3
Hit/Miss	M	M	H	M	M	M	M	M	M	H										

Direct-Mapped Caches

Slot #	Address	Item
0	4	
1	9	
2	6	
3	3	

Sequence Id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Address	3	6	3	1	9	1	9	1	9	3	4	5	6	9	3	4	5	6	9	3
Hit/Miss	M	M	H	M	M	M	M	M	M	H	M									

Direct-Mapped Caches

Slot #	Address	Item
0	4	
1	~~9~~ 5	
2	6	
3	3	

Sequence Id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Address	3	6	3	1	9	1	9	1	9	3	4	5	6	9	3	4	5	6	9	3
Hit/Miss	M	M	H	M	M	M	M	M	M	H	M	M								

Direct-Mapped Caches

Slot #	Address	Item
0	4	
1	5	
2	6	
3	3	

Sequence Id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Address	3	6	3	1	9	1	9	1	9	3	4	5	6	9	3	4	5	6	9	3
Hit/Miss	M	M	H	M	M	M	M	M	M	H	M	M	H							

Direct-Mapped Caches

Slot #	Address	Item
0	4	
1	~~5~~ 9	
2	6	
3	3	

Sequence Id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Address	3	6	3	1	9	1	9	1	9	3	4	5	6	9	3	4	5	6	9	3
Hit/Miss	M	M	H	M	M	M	M	M	M	H	M	M	H	M						

Direct-Mapped Caches

Slot #	Address	Item
0	4	
1	9	
2	6	
3	3	

Sequence Id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Address	3	6	3	1	9	1	9	1	9	3	4	5	6	9	3	4	5	6	9	3
Hit/Miss	M	M	H	M	M	M	M	M	M	H	M	M	H	M	H					

Direct-Mapped Caches

Slot #	Address	Item
0	4	
1	9	
2	6	
3	3	

Sequence Id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Address	3	6	3	1	9	1	9	1	9	3	4	5	6	9	3	4	5	6	9	3
Hit/Miss	M	M	H	M	M	M	M	M	M	H	M	M	H	M	H	H				

Direct-Mapped Caches

Slot #	Address	Item
0	4	
1	~~9~~ 5	
2	6	
3	3	

Sequence Id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Address	3	6	3	1	9	1	9	1	9	3	4	5	6	9	3	4	5	6	9	3
Hit/Miss	M	M	H	M	M	M	M	M	M	H	M	M	H	M	H	H	M			

Direct-Mapped Caches

Slot #	Address	Item
0	4	
1	5	
2	6	
3	3	

Sequence Id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Address	3	6	3	1	9	1	9	1	9	3	4	5	6	9	3	4	5	6	9	3
Hit/Miss	M	M	H	M	M	M	M	M	M	H	M	M	H	M	H	H	M	H		

Direct-Mapped Caches

Slot #	Address	Item
0	4	
1	~~5~~ 9	
2	6	
3	3	

Sequence Id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Address	3	6	3	1	9	1	9	1	9	3	4	5	6	9	3	4	5	6	9	3
Hit/Miss	M	M	H	M	M	M	M	M	M	H	M	M	H	M	H	H	M	H	M	

Direct-Mapped Caches

Slot #	Address	Item
0	4	
1	9	
2	6	
3	3	

Sequence Id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Address	3	6	3	1	9	1	9	1	9	3	4	5	6	9	3	4	5	6	9	3
Hit/Miss	M	M	H	M	M	M	M	M	M	H	M	M	H	M	H	H	M	H	M	H

Fully-Associative Caches

Consider a fully-associative cache of size 4. Each slot in the cache can have just one item (i.e. the line size is 1 item). The cache is empty to start with.

The cache uses an LRU replacement policy: every slot has a counter; every time a slot is accessed, a global counter is incremented and the value stored in the slot counter; the slot with the lowest counter value is chosen for replacement.

Work out if the following accesses to the given addresses are hits or misses. Each access is numbered with a sequence id for your convenience.

Sequence Id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Address	3	6	3	1	9	1	9	1	9	3	4	5	6	9	3	4	5	6	9	3
Hit/Miss																				

Fully-Associative Caches

Slot #	Address	Item	Local Counter
0			
1			
2			
3			

Global Counter
Value

Sequence Id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Address	3	6	3	1	9	1	9	1	9	3	4	5	6	9	3	4	5	6	9	3
Hit/Miss																				

Fully-Associative Caches

Slot #	Address	Item	Local Counter
0			
1			
2			
3			

Global Counter
Value

Sequence Id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Address	3	6	3	1	9	1	9	1	9	3	4	5	6	9	3	4	5	6	9	3
Hit/Miss																				

Fully-Associative Caches

Slot #	Address	Item	Local Counter
0	3		0
1			
2			
3			

Global Counter	
Value	0

Sequence Id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Address	3	6	3	1	9	1	9	1	9	3	4	5	6	9	3	4	5	6	9	3
Hit/Miss	M																			

Fully-Associative Caches

Slot #	Address	Item	Local Counter
0	3		0
1	6		1
2			
3			

Global Counter	
Value	1

Sequence Id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Address	3	6	3	1	9	1	9	1	9	3	4	5	6	9	3	4	5	6	9	3
Hit/Miss	M	M																		

Fully-Associative Caches

Slot #	Address	Item	Local Counter
0	3		2
1	6		1
2			
3			

Global Counter	
Value	2

Sequence Id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Address	3	6	3	1	9	1	9	1	9	3	4	5	6	9	3	4	5	6	9	3
Hit/Miss	M	M	H																	

Fully-Associative Caches

Slot #	Address	Item	Local Counter
0	3		2
1	6		1
2	1		3
3			

Global Counter	
Value	3

Sequence Id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Address	3	6	3	1	9	1	9	1	9	3	4	5	6	9	3	4	5	6	9	3
Hit/Miss	M	M	H	M																

Fully-Associative Caches

Slot #	Address	Item	Local Counter
0	3		2
1	6		1
2	1		3
3	9		4

Global Counter	
Value	4

Sequence Id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Address	3	6	3	1	9	1	9	1	9	3	4	5	6	9	3	4	5	6	9	3
Hit/Miss	M	M	H	M	M															

Fully-Associative Caches

Slot #	Address	Item	Local Counter
0	3		2
1	6		1
2	1		5
3	9		4

Global Counter	
Value	5

Sequence Id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Address	3	6	3	1	9	1	9	1	9	3	4	5	6	9	3	4	5	6	9	3
Hit/Miss	M	M	H	M	M	H														

Fully-Associative Caches

Slot #	Address	Item	Local Counter
0	3		2
1	6		1
2	1		5
3	9		6

Global Counter	
Value	6

Sequence Id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Address	3	6	3	1	9	1	9	1	9	3	4	5	6	9	3	4	5	6	9	3
Hit/Miss	M	M	H	M	M	H	H													

Fully-Associative Caches

Slot #	Address	Item	Local Counter
0	3		2
1	6		1
2	1		7
3	9		6

Global Counter	
Value	7

Sequence Id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Address	3	6	3	1	9	1	9	1	9	3	4	5	6	9	3	4	5	6	9	3
Hit/Miss	M	M	H	M	M	H	H	H												

Fully-Associative Caches

Slot #	Address	Item	Local Counter
0	3		2
1	6		1
2	1		7
3	9		8

Global Counter	
Value	8

Sequence Id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Address	3	6	3	1	9	1	9	1	9	3	4	5	6	9	3	4	5	6	9	3
Hit/Miss	M	M	H	M	M	H	H	H	H											

Fully-Associative Caches

Slot #	Address	Item	Local Counter
0	3		9
1	6		1
2	1		7
3	9		8

Global Counter	
Value	9

Sequence Id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Address	3	6	3	1	9	1	9	1	9	3	4	5	6	9	3	4	5	6	9	3
Hit/Miss	M	M	H	M	M	H	H	H	H	H										

Fully-Associative Caches

Slot #	Address	Item	Local Counter
0	3		9
1	~~6~~ 4		10
2	1		7
3	9		8

Global Counter	
Value	10

Sequence Id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Address	3	6	3	1	9	1	9	1	9	3	4	5	6	9	3	4	5	6	9	3
Hit/Miss	M	M	H	M	M	H	H	H	H	H	M									

Fully-Associative Caches

Slot #	Address	Item	Local Counter
0	3		9
1	4		10
2	~~1~~ 5		11
3	9		8

Global Counter	
Value	11

Sequence Id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Address	3	6	3	1	9	1	9	1	9	3	4	5	6	9	3	4	5	6	9	3
Hit/Miss	M	M	H	M	M	H	H	H	H	H	M	M								

Fully-Associative Caches

Slot #	Address	Item	Local Counter
0	3		9
1	4		10
2	5		11
3	~~9~~ 6		12

Global Counter	
Value	12

Sequence Id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Address	3	6	3	1	9	1	9	1	9	3	4	5	6	9	3	4	5	6	9	3
Hit/Miss	M	M	H	M	M	H	H	H	H	H	M	M	M							

Fully-Associative Caches

Slot #	Address	Item	Local Counter
0	~~3~~ 9		13
1	4		10
2	5		11
3	6		12

Global Counter	
Value	13

Sequence Id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Address	3	6	3	1	9	1	9	1	9	3	4	5	6	9	3	4	5	6	9	3
Hit/Miss	M	M	H	M	M	H	H	H	H	H	M	M	M	M						

Fully-Associative Caches

Slot #	Address	Item	Local Counter
0	9		13
1	4 3		14
2	5		11
3	6		12

Global Counter	
Value	14

Sequence Id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Address	3	6	3	1	9	1	9	1	9	3	4	5	6	9	3	4	5	6	9	3
Hit/Miss	M	M	H	M	M	H	H	H	H	H	M	M	M	M	M					

Fully-Associative Caches

Slot #	Address	Item	Local Counter
0	9		13
1	3		14
2	~~5~~ 4		15
3	6		12

Global Counter	
Value	15

Sequence Id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Address	3	6	3	1	9	1	9	1	9	3	4	5	6	9	3	4	5	6	9	3
Hit/Miss	M	M	H	M	M	H	H	H	H	H	M	M	M	M	M	M				

Fully-Associative Caches

Slot #	Address	Item	Local Counter
0	9		13
1	3		14
2	4		15
3	~~6~~ 5		16

Global Counter	
Value	16

Sequence Id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Address	3	6	3	1	9	1	9	1	9	3	4	5	6	9	3	4	5	6	9	3
Hit/Miss	M	M	H	M	M	H	H	H	H	H	M	M	M	M	M	M	M			

Fully-Associative Caches

Slot #	Address	Item	Local Counter
0	~~9~~ 6		17
1	3		14
2	4		15
3	5		16

Global Counter	
Value	17

Sequence Id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Address	3	6	3	1	9	1	9	1	9	3	4	5	6	9	3	4	5	6	9	3
Hit/Miss	M	M	H	M	M	H	H	H	H	H	M	M	M	M	M	M	M	M		

Fully-Associative Caches

Slot #	Address	Item	Local Counter
0	6		17
1	~~3~~ 9		18
2	4		15
3	5		16

Global Counter	
Value	18

Sequence Id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Address	3	6	3	1	9	1	9	1	9	3	4	5	6	9	3	4	5	6	9	3
Hit/Miss	M	M	H	M	M	H	H	H	H	H	M	M	M	M	M	M	M	M	M	

Fully-Associative Caches

Slot #	Address	Item	Local Counter
0	6		17
1	9		18
2	4 3		19
3	5		16

Global Counter	
Value	19

Sequence Id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Address	3	6	3	1	9	1	9	1	9	3	4	5	6	9	3	4	5	6	9	3
Hit/Miss	M	M	H	M	M	H	H	H	H	H	M	M	M	M	M	M	M	M	M	M