Your turn

Assume the contents of the T registers are as below:

```
T5   0x1122334455667788
T6   0x99aabbccddeeff11
T7   0xabcdeffedcba0cdc
```

The following directives have been used to reserve memory locations (memory has been reset beforehand).

```
data{
a:   byte 0x12;
b:   long 0x34567890;
c:   byte 0xab;
d:   word 0xcdef;
e:   long 0x87654321;
     }
```

Show the contents of the memory and the labels for various locations assuming that label **a** is stored at starting address 0x10…00.

label	address	content	label	address	content
	0x10…00			0x10…08	
	0x10…01			0x10…09	
	0x10…02			0x10…0a	
	0x10…03			0x10…0b	
	0x10…04			0x10…0c	
	0x10…05			0x10…0d	
	0x10…06			0x10…0e	
	0x10…07			0x10…0f	

Which of the instructions below cannot be executed?

Indicate how the contents of the memory and registers are changed by the executable instructions.

```
     ldiq $T8, a;
     ldw  $T0,0($T8);
     ldiq $T8, c;
     ldq  $T1, 0($T8);
     ldiq $T8, d;
     ldwu $T2, 0($T8);
     ldiq $T8, c;
     lda  $T3, 0($T8);
     ldw  $T4, 2($T3);
     stl  $T5, -8($T3);
     stb  $T6, 3($T3);
     ldiq $T8, e; stw  $T7, 0($T8);
     ldiq $T8, b;  ldq  $T0, ($T8);
     stw  $T1,-7($T3);
```

Solution

Memory is allocated as:

label	address	content	label	address	content
a	0x10…00	12	c	0x10…08	ab
	0x10…01	00		0x10…09	00
	0x10…02	00	d	0x10…0a	ef
	0x10…03	00		0x10…0b	cd
b	0x10…04	90	e	0x10…0c	21
	0x10…05	78		0x10…0d	43
	0x10…06	56		0x10…0e	65
	0x10…07	34		0x10…0f	78

After processing instructions, memory and register contents are modified as:

label	address	content	label	address	content
a	0x10…00	88	c	0x10…08	ab
	0x10…01	77		0x10…09	00
	0x10…02	66	d	0x10…0a	ef
	0x10…03	55		0x10…0b	11
b	0x10…04	90	e	0x10…0c	dc
	0x10…05	78		0x10…0d	0c
	0x10…06	56		0x10…0e	65
	0x10…07	34		0x10…0f	87

```
T0   0x0000000000000012
T1   0x87654321cdef00ab
T2   0x000000000000cdef
T3   0x0000000000000008
T4   0xffffffffffffcdef
```

The following two cannot be executed, since the addresses are not aligned.

```
ldq $T0, ($T8);      stw $T1, -7($T3)
```

Alignment statements can be used to round the current address up to a multiple of the size of a specified type. This is needed because data has to be aligned appropriately, for it to be accessed.
Generally, it is a good idea to align data labels to quadwords, no matter what the size of the data. If labels are not at least aligned to quadwords, then the memory display in the simulator will be confused.
Example:

```
data {
    align quad;
a:  byte 0;
    align quad;
b:  word 0;
    align quad;
c:  long 0;
    align quad;
d:  quad 0;
    }
```

If label a is supposed to be stored at the starting address 0x10…00, then label b will start at address 0x10…08, label c at address 0x10…10 and label d at address 0x10…18.

More ?

Suppose we have memory
0x1000000 0x123456789abcdef0
0x1000008 0x0000000000000000
0x1000010 0x0000000000000000

How will the registers and memory change after executing the instructions
ldiq $t0, 0x1000000;
ldq $t1, ($t0);
stb $t1, 8($t0);
ldbu $t2, 2($t0);
sll $t2, 56, $t3;
sra $t2, 56, $t4;
stq $t4, 16($t0);

4.2 Memory Integer Load/Store Instructions

The instructions in this section move data between the integer registers and memory.

They use the Memory instruction format. The instructions are summarized in Table 4–2.

Table 4–2: Memory Integer Load/Store Instructions

Mnemonic	Operation
LDA	Load Address
LDAH	Load Address High
LDBU	Load Zero-Extended Byte from Memory to Register
LDL	Load Sign-Extended Longword
LDL_L	Load Sign-Extended Longword Locked
LDQ	Load Quadword
LDQ_L	Load Quadword Locked
LDQ_U	Load Quadword Unaligned
LDWU	Load Zero-Extended Word from Memory to Register
STB	Store Byte
STL	Store Longword
STL_C	Store Longword Conditional
STQ	Store Quadword
STQ_C	Store Quadword Conditional
STQ_U	Store Quadword Unaligned
STW	Store Word