5.11 Memory alignment and endianness

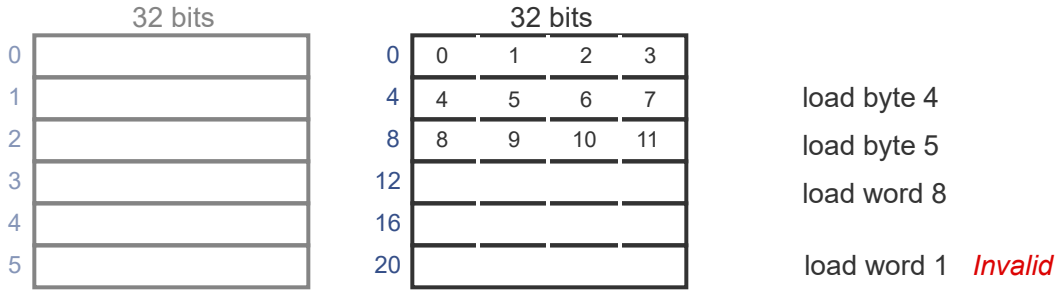
Memory alignment

A particular memory may store a sequence of 32-bit wide words (instructions or data). One might assume addresses for ea increment by 1, as in: 0, 1, 2, 3, 4, 5, etc. However, each byte in a word can be addressed individually. Thus, addresses of eac increment by 4: 0, 4, 8, 12, 16, etc. ***Memory alignment*** is the restriction of word addresses to multiples of 4 (or other multipl processors).

Instructions that load or store words must use addresses that are multiples of four. Instructions that load or store bytes ma address.

PARTICIPATION ACTIVITY 5.11.1: Memory alignment: Because each byte is addressable, word addresses are multiples of 4.

Start 2x speed



PARTICIPATION ACTIVITY 5.11.2: Memory alignment.

Consider the above animation on memory alignment.

1) How many bytes exist per word?

Check Show answer

2) How many byte addresses exist for one word?

Check Show answer

3) What are the byte addresses for the bytes in the word starting at address 12? Type as: 0, 1, 2, 3

Check Show answer

4) Is storing a byte into address 15 allowed? Type yes or no.

Check Show answer

5) Is storing a word into address 15

allowed? Type yes or no.

Check Show answer

6) A programmer wishes to load the last word of the memory shown in the animation. What address should be used in the load word instruction?

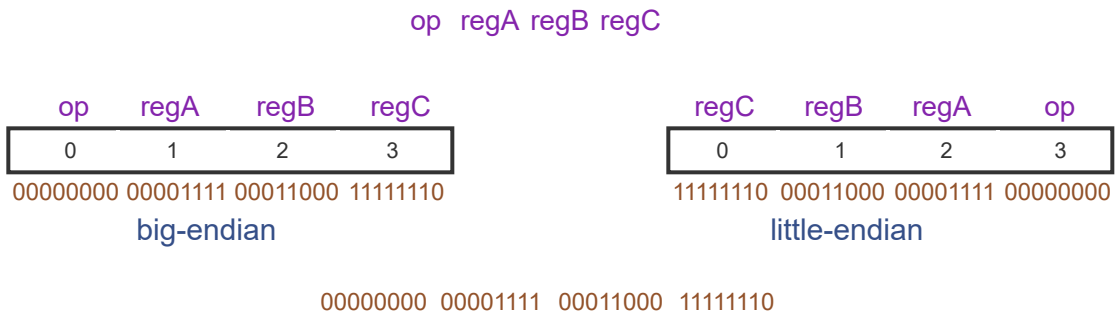
Check Show answer

Endianness

Endianness refers to whether bytes in a word are ordered starting with the most-significant byte first (***big-endian***) or the lea byte first (***little-endian***). Some processors use big-endian format, others use little-endian.

PARTICIPATION ACTIVITY 5.11.3: Big-endian vs. little-endian.

Start 2x speed



Endianness only impacts the ordering of bytes; the bits within the byte remain in the same order. Ex: 00001111 remains 00 big or little endian formats, and does not become 11110000.

Programmers usually need not be concerned with endianness, unless doing byte-level operations within a word (which is ra

PARTICIPATION ACTIVITY 5.11.4: Big-endian.

The binary number 00000000 00001111 11111111 11000000 (1048512 in decimal) is to be stored in word 20 in big-endian format. Indicate the byte address of each byte.

00000000	00001111	11000000	11111111
20			
21			
22			
23			

Reset

PARTICIPATION ACTIVITY 5.11.5: Little-endian.

The binary number 00000000 00001111 11111111 11000000 (1048512 in decimal) is to be stored in word 20 in little-endian format. Indicate the byte address of each byte.

11111111	00001111	00000000	11000000
20			
21			
22			
23			

Reset

PARTICIPATION ACTIVITY 5.11.6: Endianness.

1) Little-endian processors are faster than big-endian.
   - ○ True
   - ○ False

2) Programmers spend much time and effort focusing on endianness.
   - ○ True
   - ○ False

3)

In little-endian format, 10000000 would become 00000001.

- ○ True
- ○ False

Provide feedback on this section