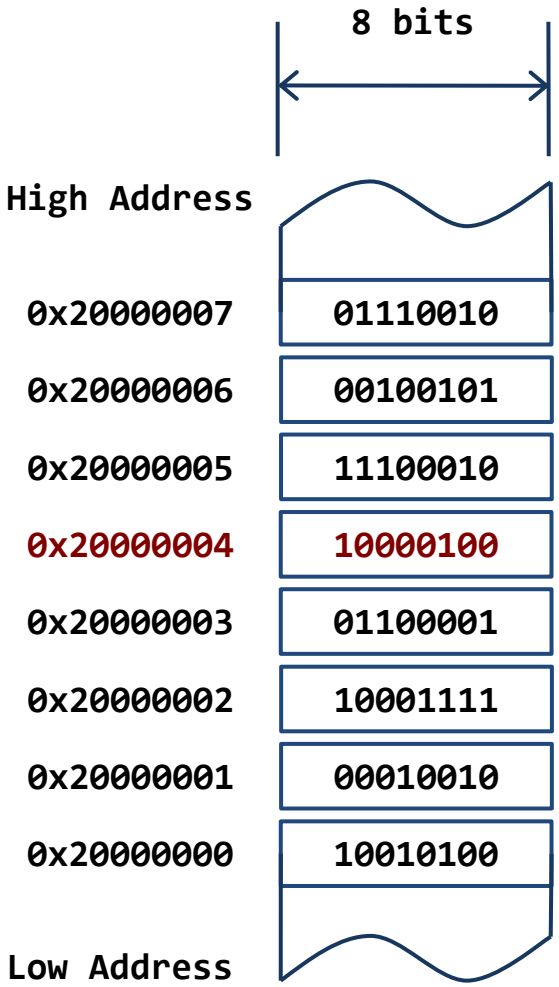
L2 (CHAPTER 5)

Programming in Assembly
Part 1: Computer Organization
Exercises ANS

Review

Logic View of Memory

- By grouping bits together we can store more values
  - 8 bits = 1 **byte**
  - 16 bits = 2 bytes = 1 **halfword**
  - 32 bits = 4 bytes = 1 **word**
- From software perspective, memory is an addressable array of **bytes**.
  - The byte stored at the memory address `0x20000004` is `0b10000100`
  - A word can only be stored at an address that's divisible by 4 (Word-address mod 4 = 0, binary address ends with 00)
  - Memory address of a word is the lowest address of all 4 bytes in that word.
  - A halfword can only be stored at an address that's divisible by 2 (Halfword-address mod 2 = 0, binary address ends with 0)
  - Memory address of a halfword is the lowest address of all 2 bytes in that word.



`0b10000100` → `0x84` → `132`

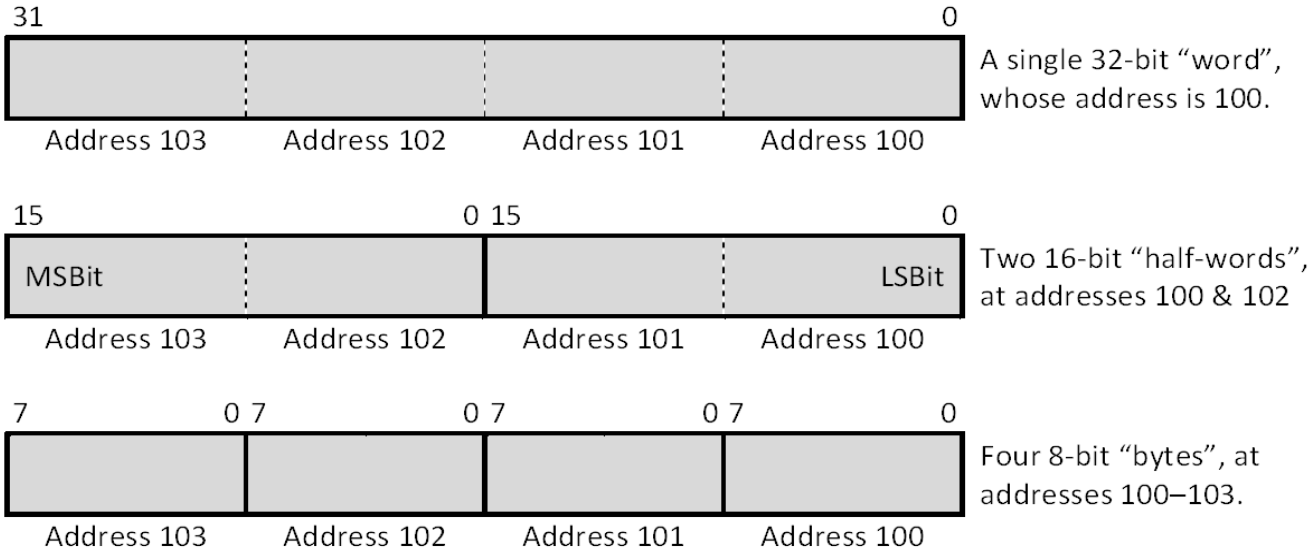
Binary Hexadecimal Decimal

2

Review

Memory Byte Ordering

- Two possible byte orderings: “little endian” and “big endian”
  - Little-endian: the LSB (Least Significant Byte) is stored at the lowest address.
  - Big-endian: the LSB (Least Significant Byte) is stored at the highest address.
  - Intel processors use Little-Endian; ARM processors can be configured as either Little- or Big-endian.
- Below are examples of Little Endian ordering (A 32-bit entity can contain one 32-bit word, or two 16-bit half-words, or four 8-bit Bytes)



Question: Endianness

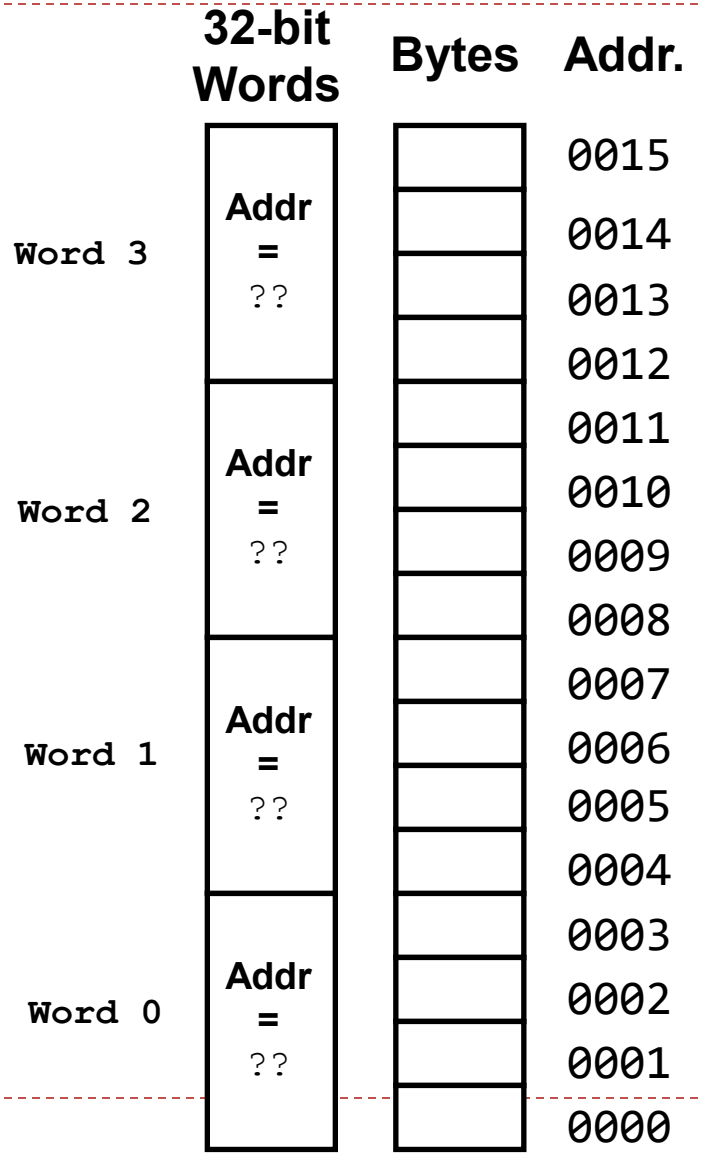
- Q: Assume Little Endian ordering. If a 32-bit word resides at memory address N, what is the address of:
  - (a) The MSB (Most Significant Byte)
  - (b) The 16-bit half-word corresponding to the most significant half of the word
- Q: Redo the question assuming Big Endian ordering.

Answer: Endianness

- Q: Assume Little Endian ordering. If a 32-bit word resides at memory address N, what is the address of:
  - (a) The MSB (Most Significant Byte)
  - (b) The 16-bit half-word corresponding to the most significant half of the word
- Q: Redo the question assuming Big Endian ordering.
- A: With Little Endian ordering:
- (a) N+3
- (b) N+2 (the half-word has address range of [N+2, N+3], so its address is N+2)
- With Big Endian ordering:
- (a) N
- (b) N (the half-word has address range of [N, N+1], so its address is N)

Question: Endianness

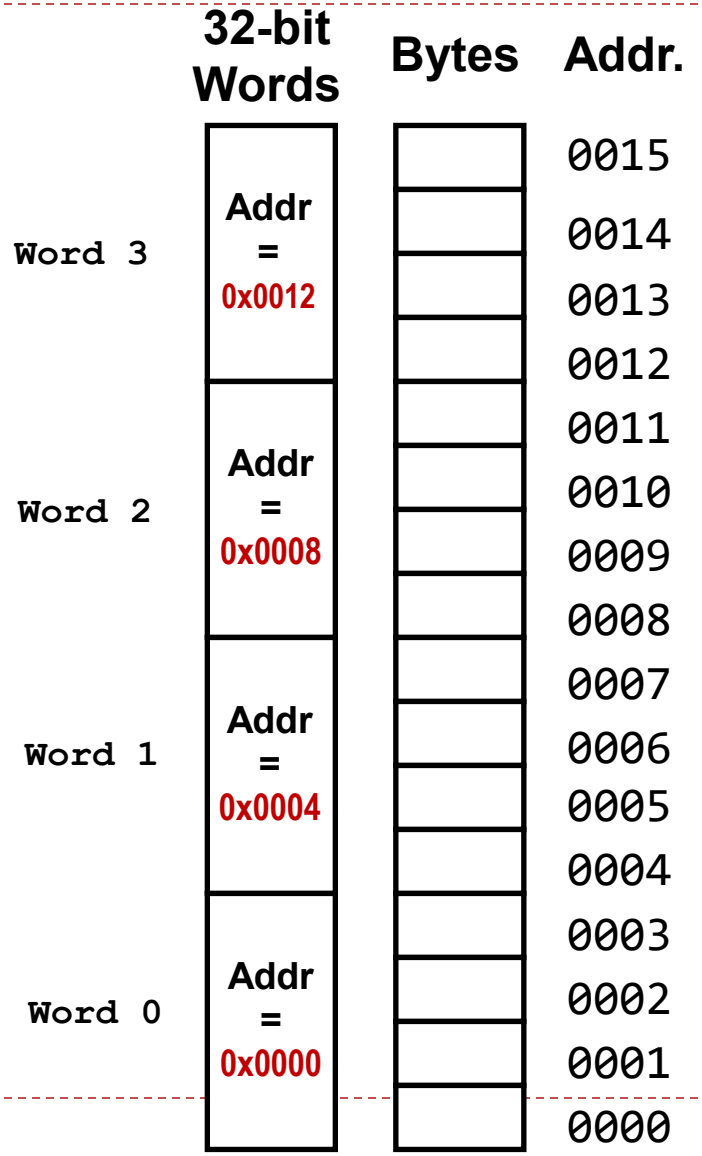
What are the memory address of these four words?



6

Answer: Endianness

What are the memory address of these four words?
Same as the address of the lowest-address Byte
(this is true for either Little-Endian or Big-Endian ordering)



7

Question: Endianness

The word stored at address 0x20008000 with Big-Endian ordering is

?

The word stored at address 0x20008000 with Little-Endian ordering is

?

Memory Address	Memory Data
0x20008003	0xA7
0x20008002	0x90
0x20008001	0x8C
0x20008000	0xEE

8

Answer: Endianness

The word stored at address 0x20008000 with Big-Endian ordering is

0xEE8C90A7

The word stored at address 0x20008000 with Little-Endian ordering is

0xA7908CEE

Memory Address	Memory Data
0x20008003	0xA7
0x20008002	0x90
0x20008001	0x8C
0x20008000	0xEE

Endianness only specifies byte order, not bit order in a byte!

Review

Data Alignment

- Assume a byte-addressable memory with a data bus that is 32 bits (4 bytes) wide
- Consider 16 bytes of memory (addresses 0 to 15) arranged as four 32-bit words (4 bytes each)

Address 15	Address 14	Address 13	Address 12
Address 11	Address 10	Address 9	Address 8
Address 7 (MSbyte)	Address 6	Address 5	Address 4 (LSbyte)
Address 3	Address 2	Address 1	Address 0

Well-aligned: each word begins on a mod-4 address, which can be read in a single memory cycle

Address 15	Address 14	Address 13	Address 12
Address 11	Address 10	Address 9 (MSbyte)	Address 8
Address 7	Address 6 (LSbyte)	Address 5	Address 4
Address 3	Address 2	Address 1	Address 0

Ill-aligned: a word begins on address 6, not a mod-4 address, which can be read in 2 memory cycles

The first read cycle would retrieve 4 bytes from addresses 4 through 7; of these, the bytes from addresses 4 and 5 are discarded, and those from addresses 6 and 7 are moved to the far right;
The second read cycle retrieves 4 bytes from addresses 8 through 11; the bytes from addresses 10 and 11 are discarded, and those from addresses 8 and 9 are moved to the far left;
Finally, the two halves are combined to form the desired 32-bit operand:

		Address 7	Address 6 (LSbyte)

Address 9 (MSbyte)	Address 8		

Address 9 (MSbyte)	Address 8	Address 7	Address 6 (LSbyte)

Question: Data Alignment

Address 111	Address 110	Address 109	Address 108
Address 107	Address 106	Address 105	Address 104
Address 103	Address 102	Address 101	Address 100
Address 99	Address 98	Address 97	Address 96

- Q: Assume a byte-addressable memory with a data bus that is 32 bits (4 bytes) wide. Consider 16 bytes of memory (addresses 0 to 15) arranged as four 32-bit words (4 bytes each).
  - (a) What is the address of the most significant byte of the word at address 102, , assuming Little-Endian ordering?
  - (b) What is the address of the most significant byte of the word at address 102, , assuming Little-Endian ordering?
  - (b) How many memory cycles are required to read the word at address 102?
  - (c) How many memory cycles are required to read the half word at address 102?

Answer: Data Alignment

Address 111	Address 110	Address 109	Address 108
Address 107	Address 106	Address 105	Address 104
Address 103	Address 102	Address 101	Address 100
Address 99	Address 98	Address 97	Address 96

- Q: Assume a byte-addressable memory with a data bus that is 32 bits (4 bytes) wide. Consider 16 bytes of memory (addresses 0 to 15) arranged as four 32-bit words (4 bytes each).
  - (a) What is the address of the most significant byte of the word at address 102, , assuming Little-Endian ordering?
  - (b) What is the address of the most significant byte of the word at address 102, , assuming Little-Endian ordering?
  - (b) How many memory cycles are required to read the word at address 102?
  - (c) How many memory cycles are required to read the half word at address 102?
- A:
  - (a) 105
  - (b) 102
  - (c) 2
  - (d) 1

Question: Data Alignment

- Q: Assume a byte-addressable memory with a data bus that is 32 bits (4 bytes) wide. Consider 16 bytes of memory (addresses 0 to 15) arranged as four 32-bit words (4 bytes each). How many memory cycles are required to read each of the following from memory?
  - (a) A 2-Byte operand read from decimal address 5
  - (b) A 2-Byte operand read from decimal address 15
  - (c) A 4-Byte operand read from decimal address 10
  - (d) A 4-Byte operand read from decimal address 20

Answer: Data Alignment

Address 15	Address 14	Address 13	Address 12
Address 11	Address 10	Address 9	Address 8
Address 7	Address 6	Address 5	Address 4
Address 3	Address 2	Address 1	Address 0

- Q: Assume a byte-addressable memory with a data bus that is 32 bits (4 bytes) wide. How many memory cycles are required to read each of the following from memory?
  - (a) A 2-Byte operand read from decimal address 5
  - (b) A 2-Byte operand read from decimal address 15
  - (c) A 4-Byte operand read from decimal address 10
  - (d) A 4-Byte operand read from decimal address 20
- A: (a) The operand contains memory content in address range [5,6]. It can be read in 1 memory cycle; the memory controller returns a word in address range [4,7]. The operand can be obtained via 1-Byte offset addressing into the word.
- (b) The operand contains memory content in address range [15,16]. It can be read in 2 memory cycles; the memory controller returns 2 words in address ranges [12,15] and [16, 19], which can be combined to return a word in address range [14,17]. The operand can be obtained via 1-Byte offset addressing into the word.
- (c) The operand contains memory content in address range [10,13]. It can be read in 2 memory cycles; the memory controller returns 2 words in address ranges [8,11] and [12, 15], which can be combined to return a word with address range [10,13].
- (d) The operand contains memory content in address range [20,23]. Since 20%4=0, it is well-aligned, and can be read in 1 memory cycle.

Question: Memory Cycles

Address 15	Address 14	Address 13	Address 12
Address 11	Address 10	Address 9	Address 8
Address 7	Address 6	Address 5	Address 4
Address 3	Address 2	Address 1	Address 0

- Q: Assume a byte-addressable memory with a data bus that is 32 bits (4 bytes) wide.
  - It takes ____ memory cycle(s) to read a Byte from memory
  - It takes ____ memory cycle(s) to read a half-word from memory
  - It takes ____ memory cycle(s) to read a word from memory
  - It takes ____ memory cycle(s) to read a double word from memory

Answer: Memory Cycles

Address 15	Address 14	Address 13	Address 12
Address 11	Address 10	Address 9	Address 8
Address 7	Address 6	Address 5	Address 4
Address 3	Address 2	Address 1	Address 0

- Q: Assume a byte-addressable memory with a data bus that is 32 bits (4 bytes) wide.
  - It takes ____ memory cycle(s) to read a Byte from memory
  - It takes ____ memory cycle(s) to read a half-word from memory
  - It takes ____ memory cycle(s) to read a word from memory
  - It takes ____ memory cycle(s) to read a double word from memory
- A:
  - It takes __1__ memory cycle(s) to read a Byte from memory
  - It takes __1 or 2__ memory cycle(s) to read a half-word from memory
  - It takes __1 or 2__ memory cycle(s) to read a word from memory
  - It takes __2 or 3__ memory cycle(s) to read a double word from memory (a double word may span at most 3 consecutive words in memory)

Question: Arrays

- Q: If the first element of a one-dimensional array x[] is located in memory at address 0x12345678, what will be the address of the second element if the array x[] contains
  - (a) chars
  - (b) shorts
  - (c) ints
  - (c) longs

Answer: Arrays

- Q: If the first element x[0] of a one-dimensional array x[] is located in memory at address 0x12345678, what will be the address of the second element x[1] if the array x[] contains
  - (a) chars
  - (b) shorts
  - (c) ints
  - (c) longs
- A: x[1]'s address is x's address plus the data type size in Bytes
  - (a) chars: 0x12345678+1= 0x12345679
  - (b) shorts: 0x12345678+2= 0x1234567A
  - (c) ints: 0x12345678+4= 0x1234567C
  - (c) longs: 0x12345678+8= 0x12345680