L2 (CHAPTER 5)
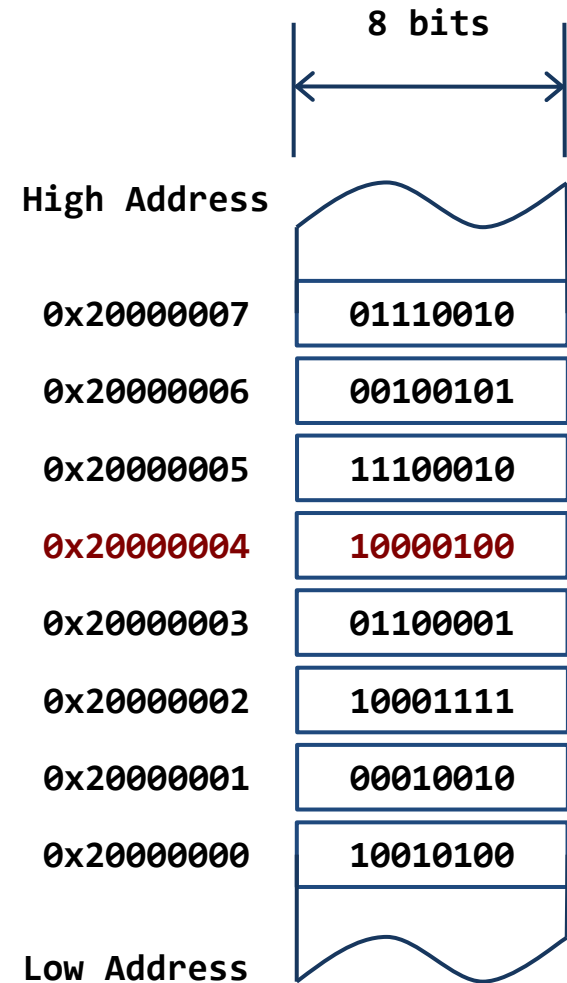
Programming in Assembly
Part 1: Computer Organization Exercises

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.

8 bits

High Address	
0x20000007	01110010
0x20000006	00100101
0x20000005	11100010
0x20000004	10000100
0x20000003	01100001
0x20000002	10001111
0x20000001	00010010
0x20000000	10010100
Low Address	

0b10000100 ⟶ 0x84 ⟶ 132

Binary   Hexadecimal   Decimal
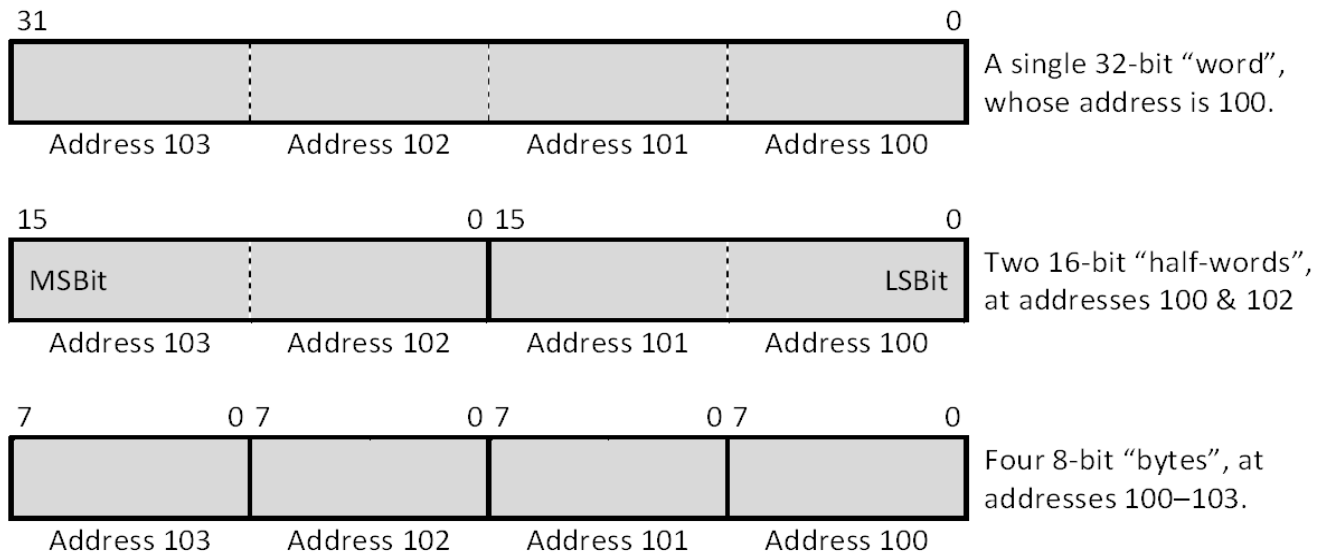
Memory Byte Ordering

- 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)

31			0	A single 32-bit "word", whose address is 100.
Address 103	Address 102	Address 101	Address 100	

15	0 15		0	Two 16-bit "half-words", at addresses 100 & 102
MSBit			LSBit	
Address 103	Address 102	Address 101	Address 100	

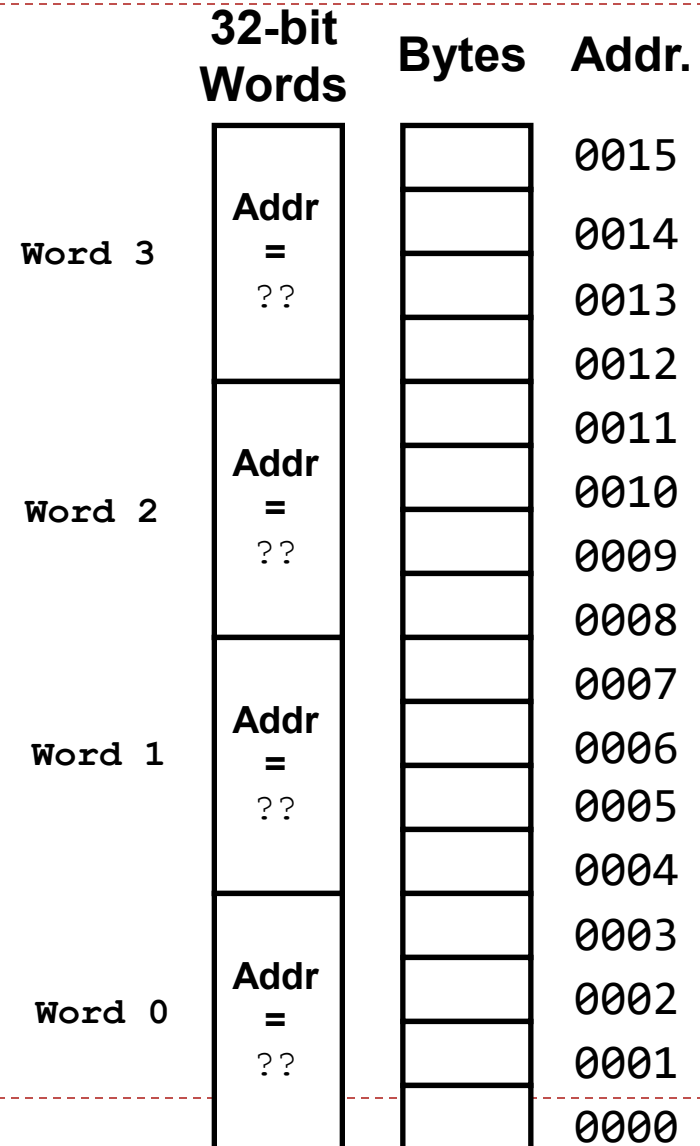| 7	0 7	0 7	0 7	0	Four 8-bit "bytes", at addresses 100–103.
Address 103	Address 102	Address 101	Address 100		

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.

Question: Endianness

32-bit Words **Bytes** **Addr.**

What are the memory address of these four words?

Word	Addr	Bytes	Addr.
			0015
Word 3	Addr = ??		0014
			0013
			0012
			0011
Word 2	Addr = ??		0010
			0009
			0008
			0007
Word 1	Addr = ??		0006
			0005
			0004
			0003
Word 0	Addr = ??		0002
			0001
			0000

5

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

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

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

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

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:

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

		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?

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

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

Question: Memory Cycles

- 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

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