NATIONAL RESEARCH UNIVERSITY

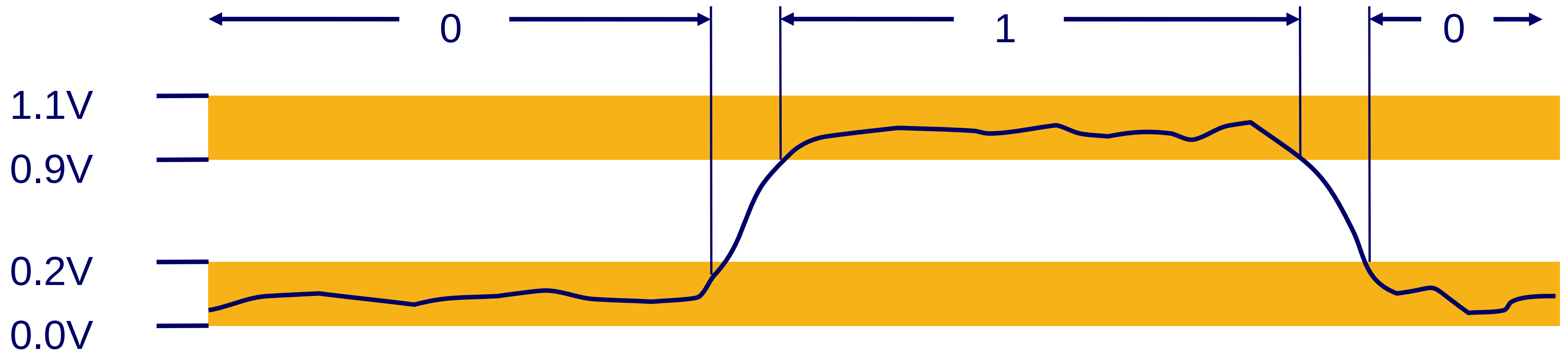
Computer Architecture and Operating Systems Lecture 2: Data Representation

Andrei Tatarnikov

atatarnikov@hse.ru
@andrewt0301

Everything is Bits

- Each bit is 0 or 1
- By encoding/interpreting sets of bits in various ways
  - Computers determine what to do (instructions)
  - … and represent and manipulate numbers, sets, strings, etc…
- Why bits? Electronic implementation
  - Easy to store with bistable elements
  - Reliably transmitted on noisy and inaccurate wires



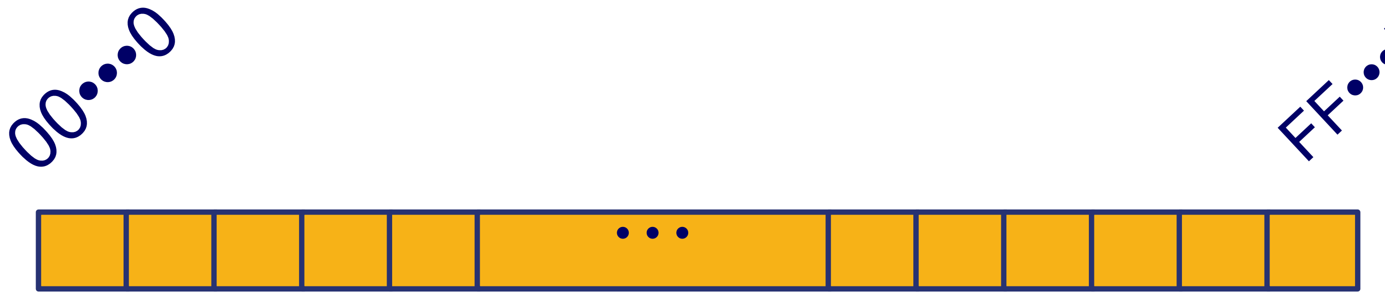
Encoding Byte Values

- Byte = 8 bits
  - Binary 00000000_2 to 11111111_2
  - Decimal: 0_{10} to 255_{10}
  - Hexadecimal 00_{16} to FF_{16}
    - Base 16 number representation
    - Use characters ‘0’ to ‘9’ and ‘A’ to ‘F’
    - Write $FA1D37B_{16}$ in C as
      - 0xFA1D37B
      - 0xfa1d37b

Hex	Decimal	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

Byte-Oriented Memory Organization

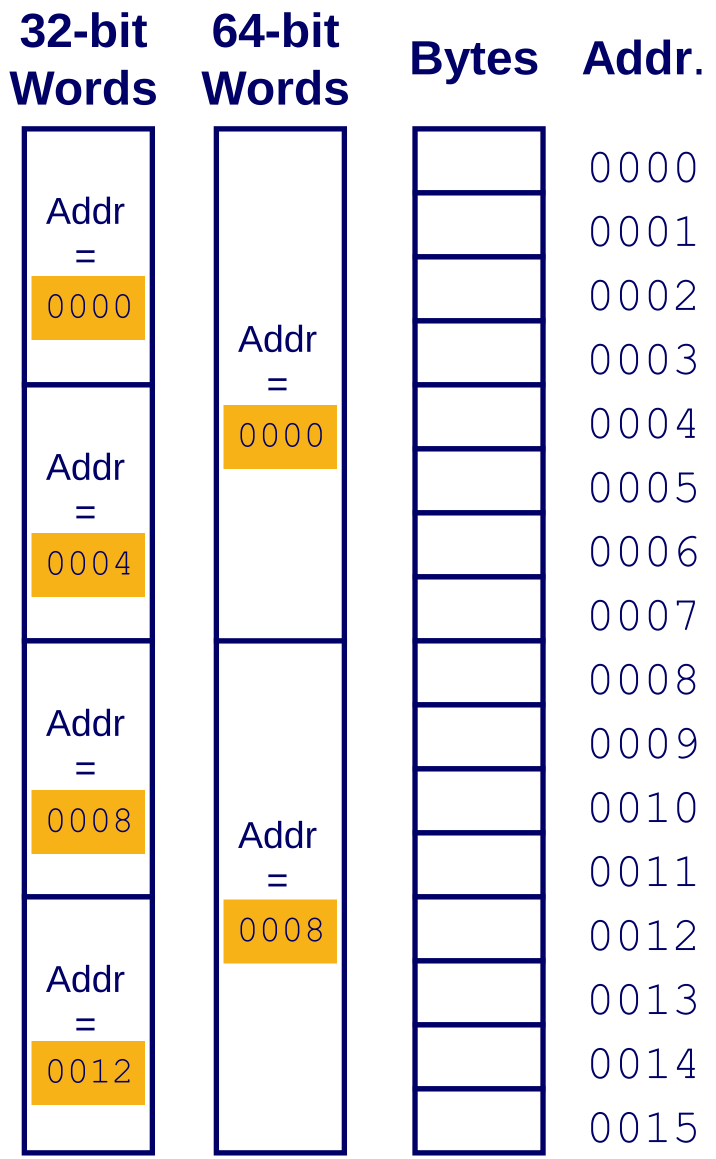
- Programs refer to data by address
  - Conceptually, envision it as a very large array of bytes
    - In reality, it's not, but can think of it that way
  - An address is like an index into that array
    - and, a pointer variable stores an address
- Note: system provides private address spaces to each "process"
  - Think of a process as a program being executed
  - So, a program can clobber its own data, but not that of others



Machine Words

- Word is a native unit of information handled by computer
- Any computer has a "Word Size"
  - Nominal size of integer-valued data
    - and of addresses
  - Until recently, most machines used 32 bits (4 bytes) as word size
    - Limits addresses to 4GB (2^{32} bytes)
  - Increasingly, machines have 64-bit word size
    - Potentially, could have 18 EB (exabytes) of addressable memory
    - That's 18.4 X 10^{18}
  - Machines still support multiple data formats
    - Fractions or multiples of word size
    - Always integral number of bytes

Word-Oriented Memory Organization



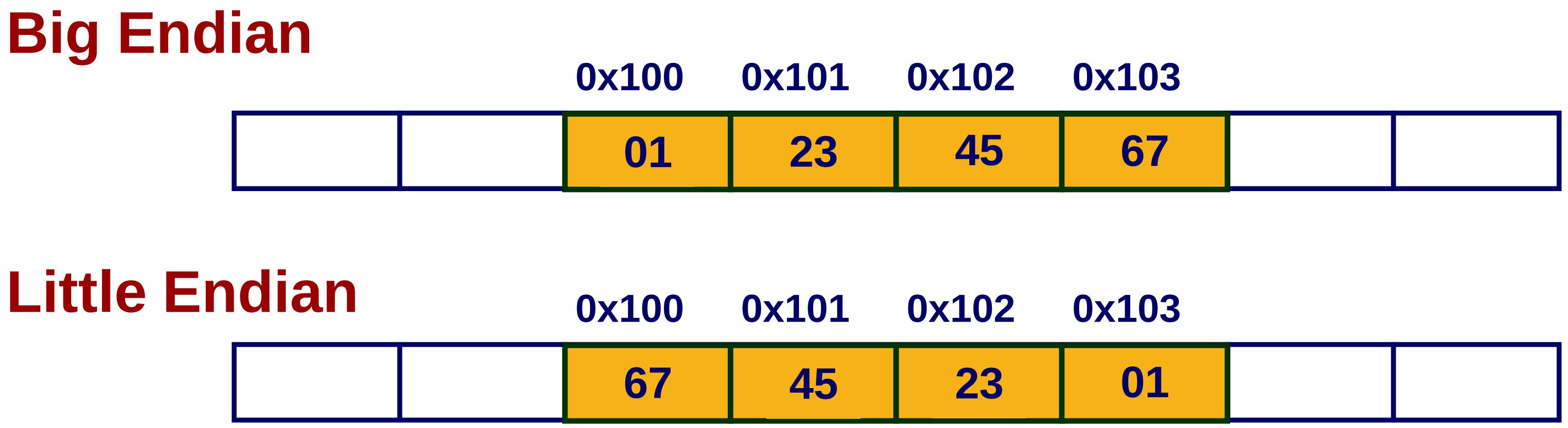
- Addresses Specify Byte Locations
  - Address of first byte in word
  - Addresses of successive words differ by 4 (32-bit) or 8 (64-bit)

Byte Ordering

- How are the bytes within a multi-byte word ordered in memory?
- Conventions
  - **Big Endian**: Sun, PPC Mac, Internet
    - Least significant byte has highest address
  - **Little Endian**: x86, ARM processors running Android, iOS, and Windows, RISC-V
    - Least significant byte has lowest address

Byte Ordering Example

- Example
  - Variable x has 4-byte value of 0x01234567
  - Address given by &x is 0x100



Any Questions?

```
                .text
__start:        addi t1, zero, 0x18
                addi t2, zero, 0x21
cycle:          beq t1, t2, done
                slt t0, t1, t2
                bne t0, zero, if_less
                nop
                sub t1, t1, t2
                j cycle
                nop
if_less:        sub t2, t2, t1
                j cycle
done:           add t3, t1, zero
```