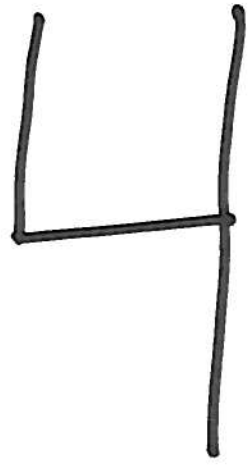
Integer Representation

CS520

Dept. of Computer Science
Univ. of New Hampshire



0100

$$123_{10} = \underbrace{1 \times 10^{2} + 2 \times 10^{1} + 3 \times 10^{0}}_{\text{evaluation formula}}$$

10 ← radix or base

binary - radix = 2 (0, 1)

octal - radix = 8 (0...7)

hexadecimal - radix = 16 (0...9, A...F)

$$11_{10} = 1011_2 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$$

$$8 + 0 + 2 + 1$$

$$11_{10}$$

$$11_{10} = B_{16} = 11_{10}$$

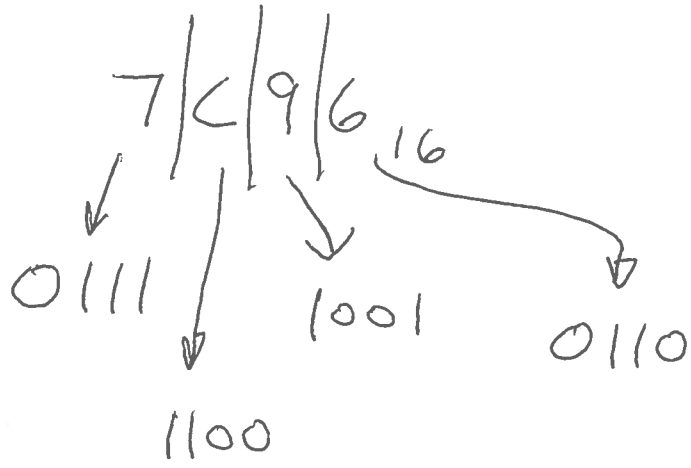
$$11_{10} = 13_8 = 1 \times 8^1 + 3 \times 8^0$$

$$8 + 3$$

$$11_{10}$$

1. hex ⟷ binary

take it one hex digit at a time!



Easy

0111 | 1100 | 1001 | 0110

7 C 9 6

binary	decimal
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7

binary	decimal	
1000	8	
1001	9	
1010	10	A
1011	11	B
1100	12	C
1101	13	D
1110	14	E
1111	15	F

2. hex → decimal

use the evaluation formula

$1A3_{16} = 1 \times 16^2 + 10 \times 16^1 + 3 \times 16^0 =$

$256 + 160 + 3 =$

419_{10}

256
160
3

419

3. decimal → hex

$$16 \overline{)\, 2113_{10}} \quad 132 \text{ r } 1$$

16
51
48
33
32
1

$$16 \overline{)\, 132} \quad 8 \text{ r } 4$$

128
4

$$16 \overline{)\, 8} \quad 0 \text{ r } 8$$

0
8

answer: 841_{16}

$8 \times 16^2 + 4 \times 16^1 + 1 \times 16^0 =$

$2048 + 64 + 1 =$

2113_{10}

256 × 8 = 2048

2048 + 64 + 1 = 2113

Intel architectures

8 bits: byte
16 bits: word
32 bits: longword
64 bits: quadword

Intel IA-32

32-bit registers
32-bit addresses

Intel 64

64-bit registers
64-bit addresses

So hex is convenient for Intel

byte is 2 hex digits
word is 4 hex digits
longword is 8 hex digits
quadword is 16 hex digits

Assembly Language Programming

for the

Control Data 6000 Series and the Cyber 70 Series

Ralph Grishman

Algorithmic

Control Data Corporation 6600 (CDC 6600)

late 1960's

6-bit char

60-bit word

18-bit address

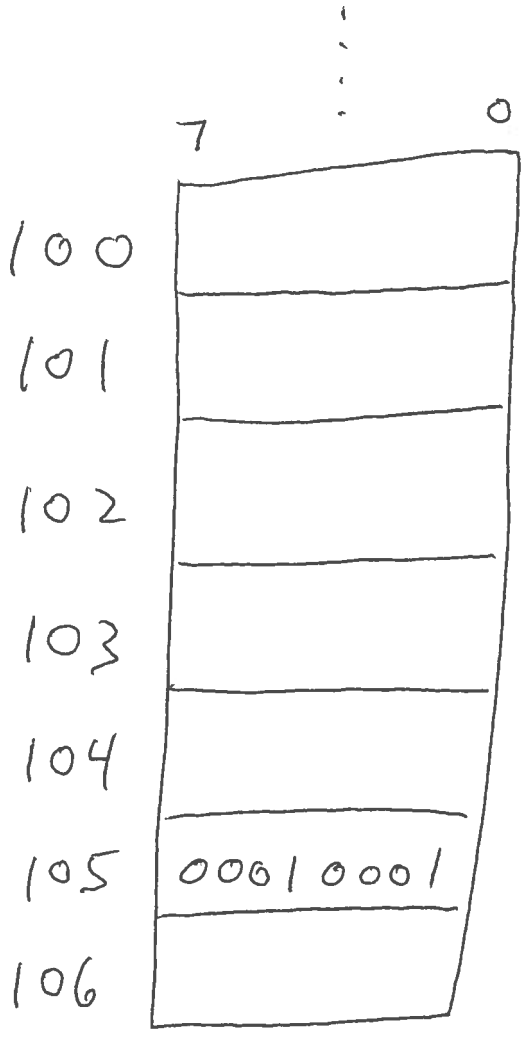
18-bit and 60-bit registers

Octal was convenient for this machine:

char is 2 octal digits

word is 20 octal digits

address is 6 octal digits



$17_{10} = 10001_2$

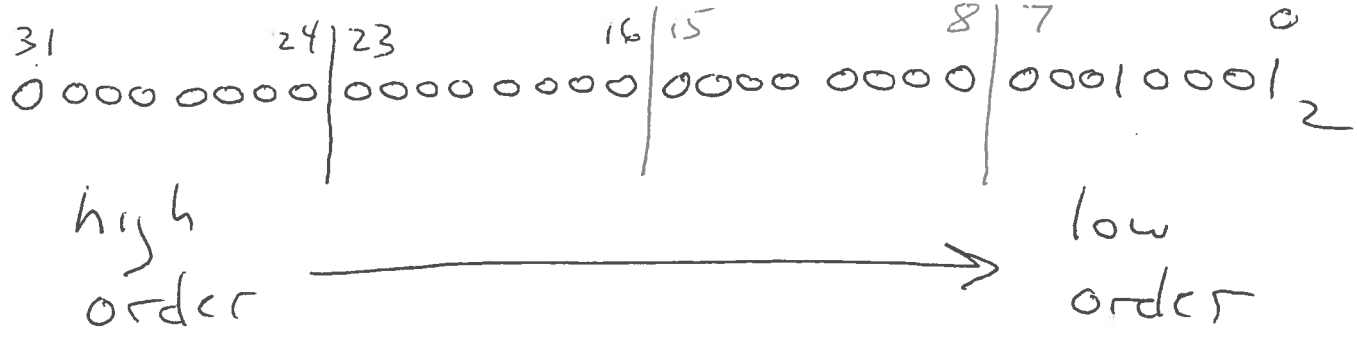
00010001_2

Big Endian – high-order byte goes first

or

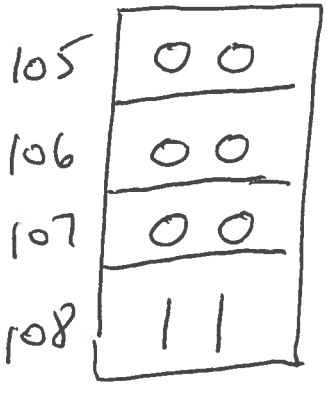
Little Endian – low-order byte goes first

17_{10} = 0000 0000 0000 0000 0000 0000 0001 0001$_2$

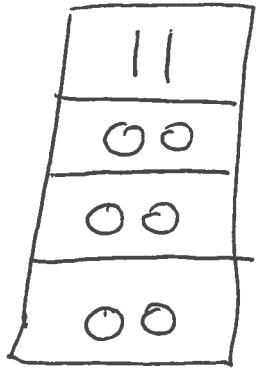


17_{10}

Big Endian



Little Endian



bits shown as hex

next time:

representing negative integers