Data Representation

Muhammad Afzaal
m.afzaal@nu.edu.pk

Book Chapter

- "Assembly Language for x86 processors"
- Author "Kip R. Irvine"
- 6th Edition
- Chapter 1
  - Section 1.3

Data Representation

- Four basic data representation techniques
  - Binary (base 2)
  - Octal (base 8)
  - Decimal (base 10)
  - Hexadecimal (base 16)

System	Base	Possible Digits
Binary	2	0 1
Octal	8	0 1 2 3 4 5 6 7
Decimal	10	0 1 2 3 4 5 6 7 8 9
Hexadecimal	16	0 1 2 3 4 5 6 7 8 9 A B C D E F

Data Representation

- Binary Integers
  - Addition
- Hexadecimal Integers
- Base Conversions
  - Binary ←→ Decimal conversion
  - Hexadecimal ←→ Binary conversion
  - Hexadecimal ←→ Decimal conversion
- Integer Storage Sizes
- Signed Integers and 2's Complement Notation
- Character Storage

Binary Integers (1/2)

- Data is stored on transistors which have two states
- Digits 1 and 0 are used to represent
  - 1 → True
  - 0 → False
- Number stored as

MSB															LSB
1	0	1	1	0	0	0	1	1	1	1	0	1	1	0	0
15															0

- Leftmost bit is call Most Significant Bit (MSB)
- Rightmost bit is called Least Significant Bit (LSB)

Binary Integers (2/2)

- Each bit either 1 or 0
- Each bit is a power of 2

1	1	1	1	1	1	1	1
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

- Values at binary bit positions

2^n	Decimal Value	2^n	Decimal Value
2^0	1	2^8	256
2^1	2	2^9	512
2^2	4	2^{10}	1024
2^3	8	2^{11}	2048
2^4	16	2^{12}	4096
2^5	32	2^{13}	8192
2^6	64	2^{14}	16384
2^7	128	2^{15}	32768

Binary Addition

- Starting from LSB, add subsequent pair of bits
- 2 integers in binary system, so four possible outcomes of adding two binary digits
- Adding 1 to 1 generates carry to next higher bit position

0	0	1	1
+ 0	+ 1	+ 0	+ 1
0	1	1	1 0

Hexadecimal Integers

- Used to represent large binary numbers
- Digits 0 to 15 are used in hexadecimal notation
- Commonly used to represent memory addresses
- In Intel Assembly language, hex numbers are denoted by a suffix `h` or `H` e.g. '`14h`'

Base Conversions

- Unsigned binary integers to decimal
- Unsigned decimal integers to binary
- Hexadecimal to binary
- Binary to hexadecimal
- Hexadecimal to decimal
- Decimal to hexadecimal

Binary to Decimal (1/2)

- Weighted Positional Notation method

$$\text{Dec} = (D_{n\text{-}1} \times 2^{n\text{-}1}) + (D_{n\text{-}2} \times 2^{n\text{-}2}) + \ldots + (D_1 \times 2^1) + (D_0 \times 2^0)$$

  - D = binary digit
  - n = bit position number in binary number

Binary to Decimal (2/2)

$0\ 1\ 1\ 1_2$ → 4-bit number so $n = 4$

$D_0 = 1$

$D_1 = 1$

$D_2 = 1$

$D_3 = 0$

$$\text{Dec} = (D_{n-1} \times 2^{n-1}) + (D_{n-2} \times 2^{n-2}) + \ldots + (D_1 \times 2^1) + (D_0 \times 2^0)$$

$$= (D_{4-1} \times 2^{4-1}) + (D_{4-2} \times 2^{4-2}) + \ldots + (D_1 \times 2^1) + (D_0 \times 2^0)$$

$$= (D_3 \times 2^3) + (D_2 \times 2^2) + \ldots + (D_1 \times 2^1) + (D_0 \times 2^0)$$

$$= (0 \times 2^3) + (1 \times 2^2) + (1 \times 2^1) + (1 \times 2^0)$$

$$= ?\ 7$$

Decimal to Binary (1/2)

- Repeatedly divide the decimal integer by 2 until the quotient is 0
- The combination of remainders makes the binary number
- The first remainder goes at LSB position and last digit goes at MSB position

Decimal to Binary (2/2)

- Convert 25_{10} into binary

Division	Quotient	Remainder
25 **/** 2	12	**1**
12 **/** 2	6	**0**
6 **/** 2	3	**0**
3 **/** 2	1	**1**
1 **/** 2	0	**1**

First remainder goes to LSB position

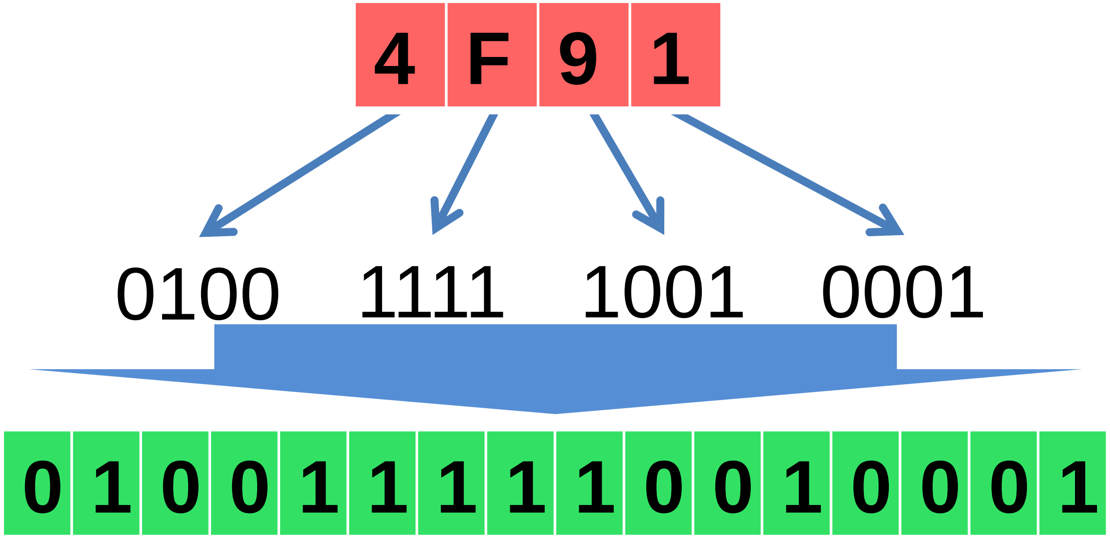
$$1\ 1\ 0\ 0\ 1_2$$

When quotient is 0, remainder goes at MSB position

- Final result is **0001 1001**

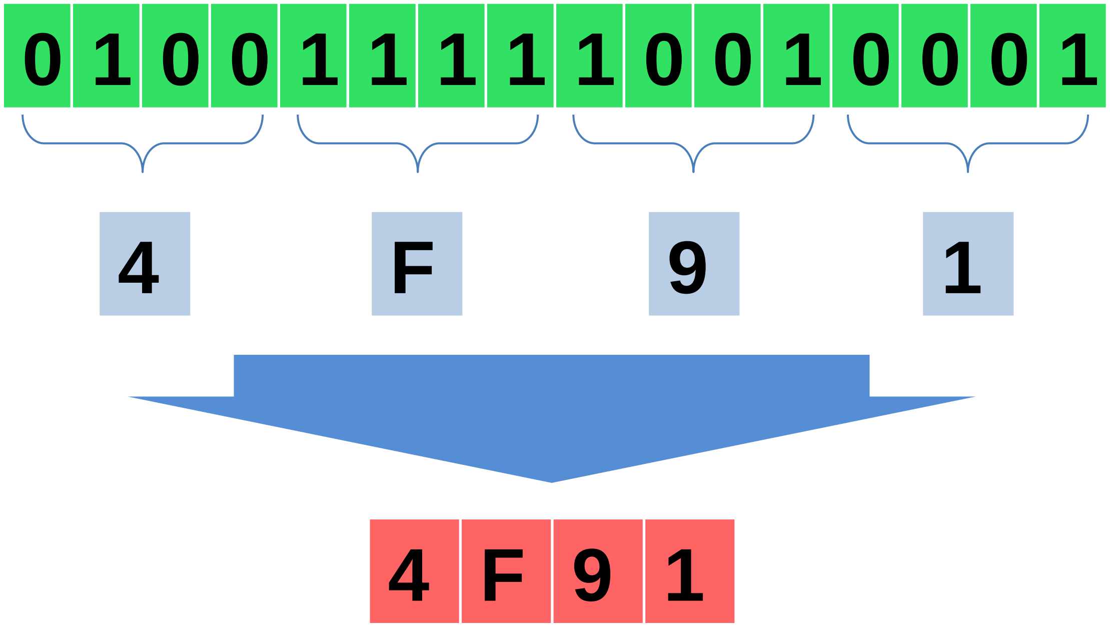
Hexadecimal to Binary

- Each hexadecimal integer corresponds to 4 binary bits
- Convert each hexadecimal number to corresponding binary number



Binary to Hexadecimal

- Convert each 4 bits of binary into its corresponding hexadecimal



Hexadecimal to Decimal (1/2)

- Multiply eat hexadecimal digit with its corresponding power of 16

$$\text{Dec} = (D_{n\text{-}1} \text{ x } 16^{n\text{-}1}) + (D_{n\text{-}2} \text{ x } 16^{n\text{-}2}) + \ldots + (D_1 \text{ x } 16^1) + (D_0 \text{ x } 16^0)$$

  - D = hexadecimal digit
  - n = digit position number in hexadecimal number

Hexadecimal to Decimal (2/2)

3 B A 4 → 4-digit number so n = 4

$D_0 = 4$

$D_1 = A$

$D_2 = B$

$D_3 = 3$

$= (D_{4\text{-}1} \times 16^{4\text{-}1}) + (D_{4\text{-}2} \times 16^{4\text{-}2}) + (D_1 \times 16^1) + (D_0 \times 16^0)$

$= (D_3 \times 16^3) + (D_2 \times 16^2) + (D_1 \times 16^1) + (D_0 \times 16^0)$

$= (3 \times 4096) + (11 \times 256) + (10 \times 16) + (4 \times 1)$

$= (12288 + 2816 + 160 + 4)$ = **15268**

Decimal to Hexadecimal (1/2)

- Repeatedly divide the decimal integer by 16 until last quotient is 0
- Each remainder is a hex digit
- First remainder goes at least significant position and last remainder goes at most significant position

Decimal to Hexadecimal (2/2)

- Convert 2895_{10} into hexadecimal

First remainder goes to LS position

Division	Quotient	Remainder
2895 **/** 16	180	**F**
180 **/** 16	11	**4**
11 **/** 16	0	**B**

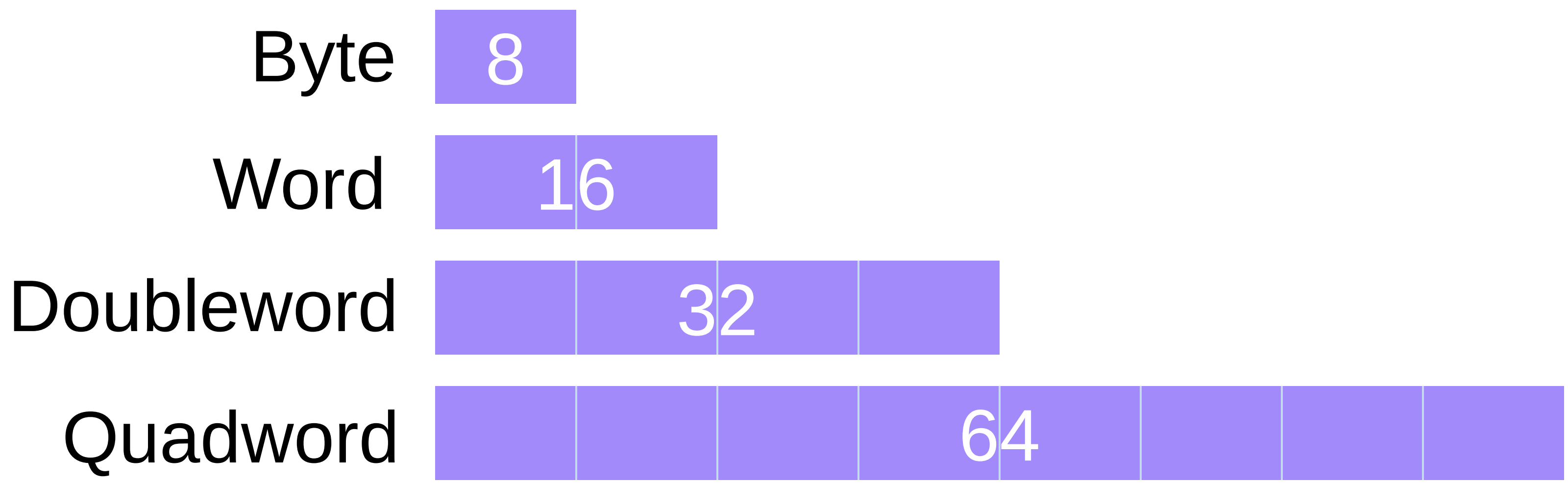
When quotient is 0, remainder goes at MS position

B 4 F_{16}

- So 2895_{10} = **B 4 F**$_{16}$

Integer Storage System (1/2)

- **Byte** is the basic storage unit in x86 architecture
- Byte is composed of **8 bits**



Integer Storage System (2/2)

- Some larger measurements units
  - One kilobyte = 2^{10} bytes = 1024 bytes
  - One megabyte = 2^{20} bytes = 1,048,576 bytes
  - One gigabyte = 2^{30} bytes = 1,073,741,824 bytes
  - One terabyte = 2^{40} bytes = 1,099,511,627,776 bytes
  - One petabyte = 2^{50} bytes = 2^{40} kilobytes
  - One exabyte = 2^{60} bytes = 2^{10} petabytes
  - One zettabyte = 2^{70} bytes = 2^{30} terabytes
  - One yottabyte = 2^{80} bytes = 2^{20} exabytes

Signed Integers

- Signed integers are either positive or negative
- Not possible to stick negative sign to a number in binary numbers
- When explicitly mentioned as signed integer, then MSB decides the +ve and –ve sign
- In signed binary/octal/hex integers
  - MSB = 1 → integers is negative
  - MSB = 0 → integers is positive
- Negative integers are represented using 2's complement notation

Range of Signed Numbers

- A certain number of bits can store only a fixed number of signed integers

Bits	Range	Total Numbers
8	-128 to +127	256
16	-32768 to +32767	65,536
32	-2,147,483,648 to +2,147,483,647	4,294,967,296
64	-9,223,372,036,854,775,808 to +9,223,372,036,854,775,807	18,446,744,073,709,551,616

Range of Unsigned Numbers

- Total numbers in signed integers is exactly equal to the total numbers in unsigned integers in the same size of bits

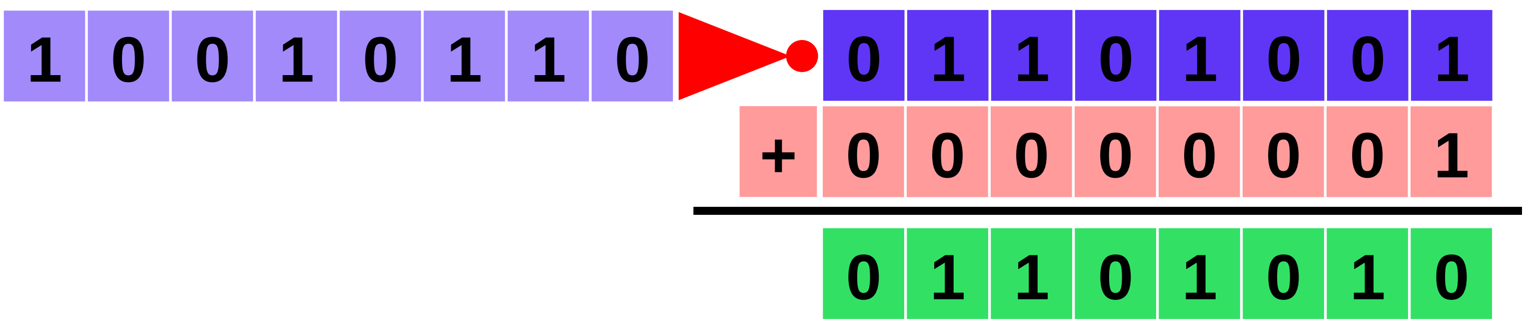
Bits	Range	Total Unsigned Numbers
8	0 to 255	256
16	0 to 65,535	65,536
32	0 to 4,294,967,295	4,294,967,296
64	0 to 18,446,744,073,709,551,615	18,446,744,073,709,551,616

2’s Complement Notation

- Useful for processors to perform subtraction with addition operation
- A fixed number of bits are used to represent the numbers
- The leftmost bit is called **sign bit**
- 2’s complement notation is used to represent both +ve and –ve numbers

How to calculate 2’s complement

- How to get 2’s complement of a binary number?
  - Take 1’s complement of that number(invert all its bits)
  - Add 1 into the inverted binary number
  - ... and the result is 2’s complement of that number



2's Complement of Hexadecimal

- Invert all bits of hex number
- All bits of hex numbers can be inverted simply by subtracting the number from F_{16}
- Add 1 into the inverted hex number and the result is the 2's complement
- Calculate 2's complement of $(B\ 4\ F)_{16}$

$$\begin{array}{r} F\ F\ F \\ -\ B\ 4\ F \\ \hline 4\ B\ 0 \end{array} \Rightarrow 4\ B\ 0 \Rightarrow \begin{array}{r} 4\ B\ 0 \\ +\quad\ \ 1 \\ \hline \mathbf{4\ B\ 1} \end{array}$$

Converting Signed Binary to Decimal

- If MSB is 0, then number is +ve and convert it into decimal in usual way
- If MSB is 1, then the number is in 2's complement notation and follow these steps
  - Calculate its 2's complement again
  - Convert this new number into decimal and add a –ve sign with it

1	**0**	**0**	**1**	**0**	**1**	**1**	**0**

$\xrightarrow{\text{2's comp.}}$

0	**1**	**1**	**0**	**1**	**0**	**1**	**0**

Decimal

1 0 6

- As the number was negative
  - So in decimal it is **– 1 0 6**

Converting Signed Decimal to Binary

- Convert absolute value of decimal into binary
- If original decimal number is –ve, calculate 2's complement of the binary number
- Convert -35 to binary

$$-35 \xrightarrow{\text{Absolute Value}} 35 \xrightarrow{\text{Binary}} 0010\ 0011$$

0	0	1	0	0	0	1	1	Invert	1	1	0	1	1	1	0	0
								+	0	0	0	0	0	0	0	1
									1	1	0	1	1	1	0	1

$$-35 = (1101\ 1101)_2$$

Convert Signed Decimal to Hexadecimal

- Convert absolute value of decimal to hex
- If decimal integer is –ve, create 2's complement of hexadecimal integer
- Convert -2895 to hexadecimal

$$-\,2895 \xrightarrow{\text{Absolute Value}} 2895 \xrightarrow{\text{Hexadecimal}} \text{B 4 F}$$

$$\text{B 4 F} \xrightarrow{\text{Bit Inversion}} \text{4 B 0} \xrightarrow{\text{Add 1}} \text{4 B 1}$$

2's Complement

Converting Signed Hex to Decimal (1/3)

- In signed hex number, if MSB=1, the number is –ve
- To convert it into decimal, follow these steps
  - Create its 2’s complement
  - Convert the 2’s complemented hex to decimal
  - Attach –ve sign to the decimal number

Converting Signed Hex to Decimal (2/3)

- Determine if Signed $8C_{16}$ is +ve or –ve
- By converting into binary
  - If MSB = 1, then number is –ve
  - $8C_{16} = (1000\ 1100)_2$
  - Since MSB = 1, so $8C_{16}$ is –ve
- Another method
  - If leftmost digit > 7, then number is –ve
  - Since leftmost digit i.e. 8 > 7
  - $8C_{16}$ is –ve

Converting Signed Hex to Decimal (3/3)

- Convert Signed $A3_{16}$ into decimal

A 3

A > 7 => A3 is –ve

2's complement of A3 = 5D

Binary Subtraction

- Big advantage of signed number is to use same circuit for addition and subtraction
- To perform A – B
  - Calculate –B by taking 2’s complement of B
  - Perform A+(-B)

	0	**1**	**1**	**1**	**0**	**1**	**1**	**0**
-	**0**	**0**	**1**	**0**	**0**	**0**	**1**	**1**

$\Rightarrow$

	0	**1**	**1**	**1**	**0**	**1**	**1**	**0**
+	**1**	**1**	**0**	**1**	**1**	**1**	**0**	**1**
	0	**1**	**0**	**1**	**0**	**0**	**1**	**1**

Next Week Lectures

- Basic Computer Organization
- IA-32 Architecture
- Instruction Execution Cycle
- Intel Microprocessors