Data Representation

COE 205

Computer Organization and Assembly Language

Dr. Aiman El-Maleh

College of Computer Sciences and Engineering

King Fahd University of Petroleum and Minerals

[Adapted from slides of Dr. Kip Irvine: Assembly Language for Intel-Based Computers]

Outline

- Introduction
- Numbering Systems
- Binary & Hexadecimal Numbers
- Base Conversions
- Integer Storage Sizes
- Binary and Hexadecimal Addition
- Signed Integers and 2's Complement Notation
- Binary and Hexadecimal subtraction
- Carry and Overflow
- Character Storage

Introduction

- Computers only deal with binary data (0s and 1s), hence all data manipulated by computers must be represented in binary format.
- Machine instructions manipulate many different forms of data:
  - Numbers:
    - Integers: 33, +128, -2827
    - Real numbers: 1.33, +9.55609, -6.76E12, +4.33E-03
  - Alphanumeric characters (letters, numbers, signs, control characters): examples: A, a, c, 1 ,3, ", +, Ctrl, Shift, etc.
  - Images (still or moving): Usually represented by numbers representing the Red, Green and Blue (RGB) colors of each pixel in an image,
  - Sounds: Numbers representing sound amplitudes sampled at a certain rate (usually 20kHz).
- So in general we have two major data types that need to be represented in computers; numbers and characters.

Numbering Systems

- Numbering systems are characterized by their base number.
- In general a numbering system with a base *r* will have *r* different digits (including the 0) in its number set. These digits will range from 0 to *r*-1
- The most widely used numbering systems are listed in the table below:

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

Binary Numbers

- Each digit (bit) is either 1 or 0
- Each bit represents a power of 2
- Every binary number is a sum of powers of 2

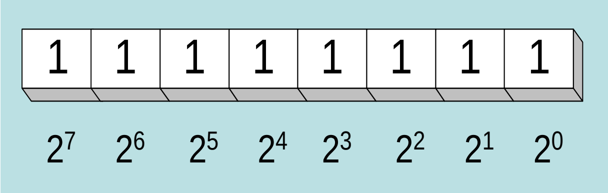


Table 1-3 Binary Bit Position Values.

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

Converting Binary to Decimal

- Weighted positional notation shows how to calculate the decimal value of each binary bit:

$Decimal = (d_{n-1} \times 2^{n-1}) + (d_{n-2} \times 2^{n-2}) + ... + (d_1 \times 2^1) + (d_0 \times 2^0)$

d = binary digit

- binary 10101001 = decimal 169:

$(1 \times 2^7) + (1 \times 2^5) + (1 \times 2^3) + (1 \times 2^0) = 128+32+8+1=169$

Convert Unsigned Decimal to Binary

❖ Repeatedly divide the decimal integer by 2. Each remainder is a binary digit in the translated value:

Division	Quotient	Remainder
37 / 2	18	1
18 / 2	9	0
9 / 2	4	1
4 / 2	2	0
2 / 2	1	0
1 / 2	0	1

least significant bit

most significant bit

stop when quotient is zero

37 = 100101

Another Procedure for Converting from Decimal to Binary

- Start with a binary representation of all 0’s
- Determine the highest possible power of two that is less or equal to the number.
- Put a 1 in the bit position corresponding to the highest power of two found above.
- Subtract the highest power of two found above from the number.
- Repeat the process for the remaining number

Another Procedure for Converting from Decimal to Binary

❖ Example: Converting 76d to Binary

- The highest power of 2 less or equal to 76 is 64, hence the seventh (MSB) bit is 1

1	.	.	.	.	.	.

- Subtracting 64 from 76 we get 12.
- The highest power of 2 less or equal to 12 is 8, hence the fourth bit position is 1

1	0	0	1	.	.	.

- We subtract 8 from 12 and get 4.
- The highest power of 2 less or equal to 4 is 4, hence the third bit position is 1

1	0	0	1	1	.	.

- Subtracting 4 from 4 yield a zero, hence all the left bits are set to 0 to yield the final answer

1	0	0	1	1	0	0

Hexadecimal Integers

❖ Binary values are represented in hexadecimal.

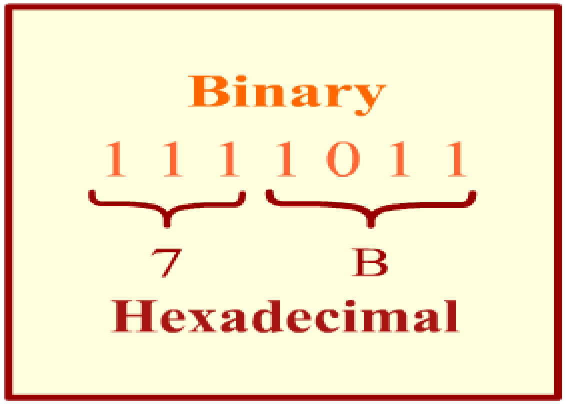
Table 1-5 Binary, Decimal, and Hexadecimal Equivalents.

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

Converting Binary to Hexadecimal

- Each hexadecimal digit corresponds to 4 binary bits.
- Example: Translate the binary integer 000101101010011110010100 to hexadecimal

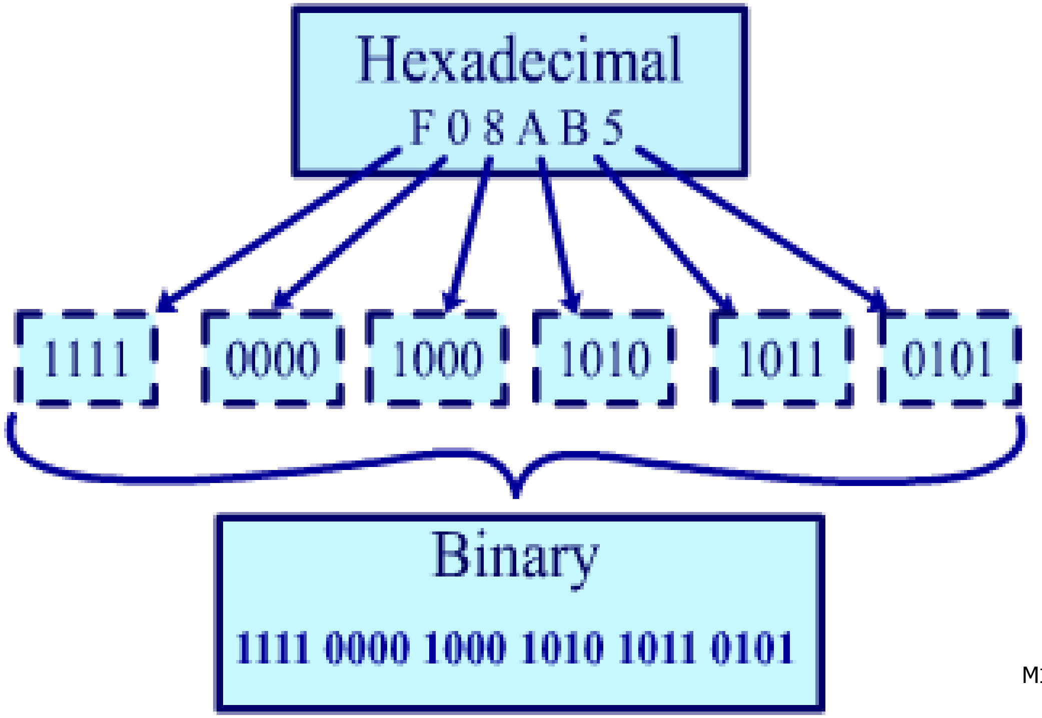
1	6	A	7	9	4
0001	0110	1010	0111	1001	0100



M1023.swf

Converting Hexadecimal to Binary

❖ Each Hexadecimal digit can be replaced by its 4-bit binary number to form the binary equivalent.



M1021.swf

Converting Hexadecimal to Decimal

- Multiply each digit by its corresponding power of 16:

  $\text{Decimal} = (d3 \times 16^3) + (d2 \times 16^2) + (d1 \times 16^1) + (d0 \times 16^0)$

  d = hexadecimal digit

- Examples:
  - Hex 1234 = $(1 \times 16^3) + (2 \times 16^2) + (3 \times 16^1) + (4 \times 16^0)$ = Decimal 4,660
  - Hex 3BA4 = $(3 \times 16^3) + (11 * 16^2) + (10 \times 16^1) + (4 \times 16^0)$ = Decimal 15,268

Converting Decimal to Hexadecimal

❖ Repeatedly divide the decimal integer by 16. Each remainder is a hex digit in the translated value:

Division	Quotient	Remainder
422 / 16	26	6
26 / 16	1	A
1 / 16	0	1

least significant digit

most significant digit

stop when quotient is zero

Decimal 422 = 1A6 hexadecimal

Integer Storage Sizes

Standard sizes:

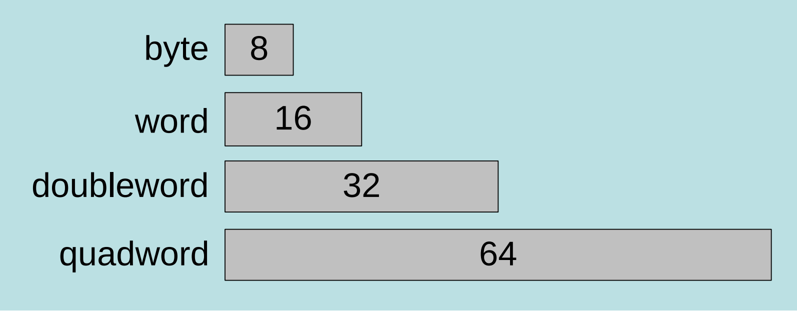


Table 1-4 Ranges of Unsigned Integers.

Storage Type	Range (low–high)	Powers of 2
Unsigned byte	0 to 255	0 to $(2^8 - 1)$
Unsigned word	0 to 65,535	0 to $(2^{16} - 1)$
Unsigned doubleword	0 to 4,294,967,295	0 to $(2^{32} - 1)$
Unsigned quadword	0 to 18,446,744,073,709,551,615	0 to $(2^{64} - 1)$

What is the largest unsigned integer that may be stored in 20 bits?

Binary Addition

- Start with the least significant bit (rightmost bit)
- Add each pair of bits
- Include the carry in the addition, if present

carry:						1			
	0	0	0	0	0	1	0	0	(4)
+	0	0	0	0	0	1	1	1	(7)
	0	0	0	0	1	0	1	1	(11)
bit position:	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	

Hexadecimal Addition

❖ Divide the sum of two digits by the number base (16). The quotient becomes the carry value, and the remainder is the sum digit.

		1	1
36	28	28	6A
42	45	58	4B
78	6D	80	B5

21 / 16 = 1, remainder 5

Important skill: Programmers frequently add and subtract the addresses of variables and instructions.

Signed Integers

- Several ways to represent a signed number
  - Sign-Magnitude
  - 1's complement
  - 2's complement
- Divide the range of values into 2 equal parts
  - First part corresponds to the positive numbers (≥ 0)
  - Second part correspond to the negative numbers (< 0)
- Focus will be on the 2's complement representation
  - Has many advantages over other representations
  - Used widely in processors to represent signed integers

Two's Complement Representation

- ❖ Positive numbers
  - ✧ Signed value = Unsigned value
- ❖ Negative numbers
  - ✧ Signed value = Unsigned value - 2^n
  - ✧ n = number of bits
- ❖ Negative weight for MSB
  - ✧ Another way to obtain the signed value is to assign a negative weight to most-significant bit

1	0	1	1	0	1	0	0
-128	64	32	16	8	4	2	1

= -128 + 32 + 16 + 4 = -76

8-bit Binary value	Unsigned value	Signed value
00000000	0	0
00000001	1	+1
00000010	2	+2
. . .	. . .	. . .
01111110	126	+126
01111111	127	+127
10000000	128	-128
10000001	129	-127
. . .	. . .	. . .
11111110	254	-2
11111111	255	-1

Forming the Two's Complement

starting value	`00100100 = +36`
step1: reverse the bits (1's complement)	`11011011`
step 2: add 1 to the value from step 1	`+        1`
sum = 2's complement representation	`11011100 = -36`

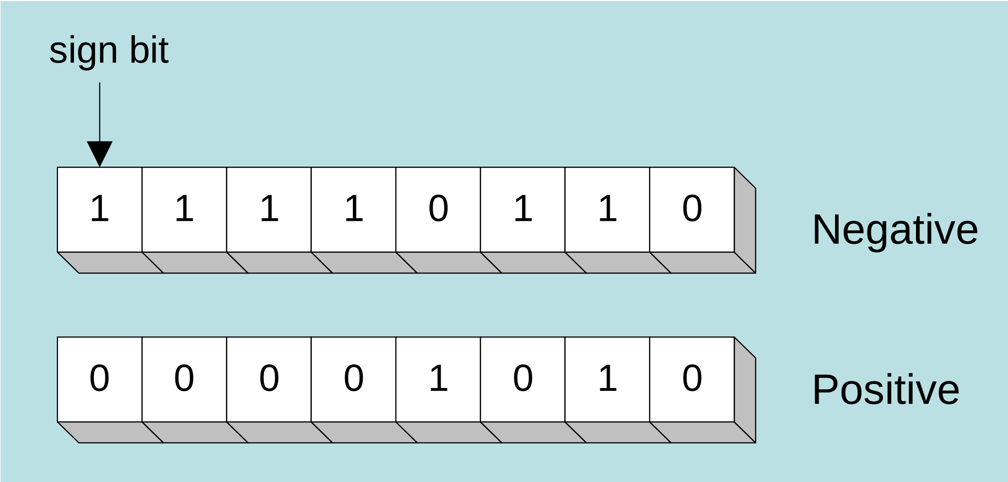
Sum of an integer and its 2's complement must be zero:

00100100 + 11011100 = 00000000 (8-bit sum) $\Rightarrow$ Ignore Carry

The easiest way to obtain the 2's complement of a binary number is by starting at the LSB, leaving all the 0s unchanged, look for the first occurrence of a 1. Leave this 1 unchanged and complement all the bits after it.

Sign Bit

Highest bit indicates the sign. 1 = negative, 0 = positive



If highest digit of a hexadecimal is > 7, the value is negative

Examples: 8A and C5 are negative bytes

A21F and 9D03 are negative words

B1C42A00 is a negative double-word

Sign Extension

Step 1: Move the number into the lower-significant bits

Step 2: Fill all the remaining higher bits with the sign bit

❖ This will ensure that both magnitude and sign are correct

❖ Examples

- ✧ Sign-Extend 10110011 to 16 bits

  10110011 = -77 ⇨ **11111111 10110011 = -77**

- ✧ Sign-Extend 01100010 to 16 bits

  01100010 = +98 ⇨ **00000000 01100010 = +98**

❖ Infinite 0s can be added to the left of a positive number

❖ Infinite 1s can be added to the left of a negative number

Two's Complement of a Hexadecimal

- To form the two's complement of a hexadecimal
  - Subtract each hexadecimal digit from 15
  - Add 1
- Examples:
  - 2's complement of 6A3D = 95C3
  - 2's complement of 92F0 = 6D10
  - 2's complement of FFFF = 0001
- No need to convert hexadecimal to binary

Two's Complement of a Hexadecimal

- Start at the least significant digit, leaving all the 0s unchanged, look for the first occurrence of a non-zero digit.
- Subtract this digit from 16.
- Then subtract all remaining digits from 15.
- Examples:
  - 2's complement of 6A3D = 95C3
  - 2's complement of 92F0 = 6D10
  - 2's complement of FFFF = 0001

```
   F F F 16
 - 6 A 3  D
 -----------
   9 5 C  3
```

```
   F F 16
 - 9 2  F 0
 -----------
   6 D  1 0
```

Binary Subtraction

- When subtracting A – B, convert B to its 2's complement
- Add A to (–B)

	0 0 0 0 1 1 0 0			0 0 0 0 1 1 0 0
–	0 0 0 0 0 0 1 0		+	1 1 1 1 1 1 1 0 (2's complement)
	0 0 0 0 1 0 1 0			0 0 0 0 1 0 1 0 (same result)

- Carry is ignored, because
  - Negative number is sign-extended with 1's
  - You can imagine infinite 1's to the left of a negative number
  - Adding the carry to the extended 1's produces extended zeros

Practice: Subtract 00100101 from 01101001.

Hexadecimal Subtraction

- When a borrow is required from the digit to the left, add 16 (decimal) to the current digit's value

```
  16 + 5 = 21
       -1 ↓                           1 1
     C675                  ⇒        C675
  -  A247                        +  5DB9   (2's complement)
     ----                           ----
     242E                           242E   (same result)
```

- Last Carry is ignored

Practice: The address of **var1** is 00400B20. The address of the next variable after var1 is 0040A06C. How many bytes are used by var1?

Ranges of Signed Integers

The unsigned range is divided into two signed ranges for positive and negative numbers

Storage Type	Range (low–high)	Powers of 2
Signed byte	−128 to +127	-2^7 to $(2^7 - 1)$
Signed word	−32,768 to +32,767	-2^{15} to $(2^{15} - 1)$
Signed doubleword	−2,147,483,648 to 2,147,483,647	-2^{31} to $(2^{31} - 1)$
Signed quadword	−9,223,372,036,854,775,808 to +9,223,372,036,854,775,807	-2^{63} to $(2^{63} - 1)$

Practice: What is the range of signed values that may be stored in 20 bits?

Carry and Overflow

- Carry is important when …
  - Adding or subtracting unsigned integers
  - Indicates that the unsigned sum is out of range
  - Either < 0 or > maximum unsigned *n*-bit value
- Overflow is important when …
  - Adding or subtracting signed integers
  - Indicates that the signed sum is out of range
- Overflow occurs when
  - Adding two positive numbers and the sum is negative
  - Adding two negative numbers and the sum is positive
  - Can happen because of the fixed number of sum bits

Carry and Overflow Examples

- We can have carry without overflow and vice-versa
- Four cases are possible

				1					
	0	0	0	0	1	1	1	1	15
+	0	0	0	0	1	0	0	0	8
	0	0	0	1	0	1	1	1	23

Carry = 0   Overflow = 0

1	1	1	1	1					
	0	0	0	0	1	1	1	1	15
+	1	1	1	1	1	0	0	0	245 (-8)
	0	0	0	0	0	1	1	1	7

Carry = 1   Overflow = 0

	1								
	0	1	0	0	1	1	1	1	79
+	0	1	0	0	0	0	0	0	64
	1	0	0	0	1	1	1	1	143 (-113)

Carry = 0   Overflow = 1

1			1	1					
	1	1	0	1	1	0	1	0	218 (-38)
+	1	0	0	1	1	1	0	1	157 (-99)
	0	1	1	1	0	1	1	1	119

Carry = 1   Overflow = 1

Character Storage

- Character sets
  - Standard ASCII: 7-bit character codes (0 – 127)
  - Extended ASCII: 8-bit character codes (0 – 255)
  - Unicode: 16-bit character codes (0 – 65,535)
  - Unicode standard represents a universal character set
    - Defines codes for characters used in all major languages
    - Used in Windows-XP: each character is encoded as 16 bits
  - UTF-8: variable-length encoding used in HTML
    - Encodes all Unicode characters
    - Uses 1 byte for ASCII, but multiple bytes for other characters
- Null-terminated String
  - Array of characters followed by a NULL character

ASCII Codes

The Charcter set of the ASCII Code

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

❖ Examples:

- ✧ ASCII code for space character = 20 (hex) = 32 (decimal)
- ✧ ASCII code for ‘A' = 41 (hex) = 65 (decimal)
- ✧ ASCII code for 'a' = 61 (hex) = 97 (decimal)

Control Characters

- The first 32 characters of ASCII table are used for control
- Control character codes = 00 to 1F (hex)
- Examples of Control Characters
  - Character 0 is the NULL character ⇒ used to terminate a string
  - Character 9 is the Horizontal Tab (HT) character
  - Character 0A (hex) = 10 (decimal) is the Line Feed (LF)
  - Character 0D (hex) = 13 (decimal) is the Carriage Return (CR)
  - The LF and CR characters are used together
    - They advance the cursor to the beginning of next line
- One control character appears at end of ASCII table
  - Character 7F (hex) is the Delete (DEL) character

Parity Bit

- Data errors can occur during data transmission or storage/retrieval.
- The 8th bit in the ASCII code is used for error checking.
- This bit is usually referred to as the parity bit.
- There are two ways for error checking:
  - Even Parity**:** Where the 8th bit is set such that the total number of 1s in the 8-bit code word is even.

P							
0	1	0	0	0	0	0	1

  - Odd Parity**:** The 8th bit is set such that the total number of 1s in the 8-bit code word is odd.

P							
1	1	0	0	0	0	0	1