Number Systems & Encoding

Lecturer: Sri Parameswaran
Author: Hui Annie Guo
Modified: Sri Parameswaran

Lecture overview

- Basics of computing with digital systems
  - Binary numbers
  - Floating point numbers
  - Encoding
    - BCD
    - ASCII
    - Others

Number representation

- Any number can be represented in the form of

$$
\begin{aligned}
&(a_n a_{n-1} \dots a_1 a_0 . a_{-1} \dots a_{-m})_r \\
&= a_n \times r^n + a_{n-1} \times r^{n-1} + \dots + a_1 \times r + a_0 + a_{-1} \times r^{-1} + \dots + a_{-m} \times r^{-m} \\
\\
&r : \text{radix, base} \\
&0 \le a_i < r
\end{aligned}
$$

Decimal

- Example

$$
\boxed{\begin{aligned}&(3597)_{10}\\&= 3\times 10^3 + 5\times 10^2 + 9\times 10 + 7\end{aligned}}
$$

  - The place values, from right to left, are 1, 10, 100, 1000
  - The base or radix is 10
  - All digits must be less than the base, namely, 0~9

Binary

- Example

$$\boxed{\begin{aligned}&(1011)_2\\&= 1\times 2^3 + 0\times 2^2 + 1\times 2 + 1\end{aligned}}$$

  - The place values, from right to left, are 1, 2, 4, 8
  - The base or radix is 2
  - All digits must be less than the base, namely, 0~1

What are the first 16 binary integers?

Hexadecimal

- Example

$$
\begin{aligned}
&(F24B)_{16} \\
&= F \times 16^3 + 2 \times 16^2 + 4 \times 16 + B \\
&= 15 \times 16^3 + 2 \times 16^2 + 4 \times 16 + 11
\end{aligned}
$$

  - The place values, from right to left, are 1, 16, 16^2, 16^3
  - The base or radix is 16
  - All digits must be less than the base, namely, 0~9,A,B,C,D,E,F

Which numbers to use?

- Digital machines use ***binary numbers***
  - Because digital devices can easily produce high or low level voltages, which can represent 1 or 0.
- Hexadecimals or sometimes octal numbers are used
  - For neat binary representation
  - For easy number conversion between binary and decimal
- Humans are familiar with decimals

Number conversion

- From base r to base 10
  - Using

$$\boxed{\begin{aligned}&(a_n a_{n-1} \ldots a_1 a_0 . a_{-1} \ldots a_{-m})_r \\ &= a_n \times r^n + a_{n-1} \times r^{n-1} + \ldots + a_1 \times r + a_0 + a_{-1} \times r^{-1} + \ldots + a_{-m} \times r^{-m}\end{aligned}}$$

- Examples:

Examples

- From base 2

$$\boxed{(1011.1)_2 = 1\times 2^3 + 0\times 2^2 + 1\times 2 + 1 + 1\times 2^{-1} = 11.5}$$

- From base 8

$$\boxed{(1005.2)_8 = 1\times 8^3 + 0\times 8^2 + 0\times 8 + 5 + 2\times 8^{-1} = 517.25}$$

- From base 16

$$\boxed{(10A)_{16} = 1\times 16^2 + 0\times 16 + 10 = 266}$$

Number conversion

- From base 10 to base r

  Based on the formula

$$(a_n a_{n-1} \ldots a_1 a_0 . a_{-1} \ldots a_{-m})_r$$
$$= a_n \times r^n + a_{n-1} \times r^{n-1} + \ldots + a_1 \times r + a_0 + a_{-1} \times r^{-1} + \ldots + a_{-m} \times r^{-m}$$

  - For whole number
    - **Divide** the number/quotient repeatedly by **r** until the quotient is zero and the remainders are the digits of base r number, in reverse order
  - For fraction
    - **Multiply** the number/fraction repeatedly by **r**, the whole numbers of the products are the digits of the base r fraction number

Examples

- To base 2
  - To convert $(11.25)_{10}$ to binary
    - For whole number $(11)_{10}$ – division **(by 2)**

11	**1**
5	**1**
2	**0**
1	**1**
0	

    - For fraction $(0.25)_{10}$ – multiplication **(by 2)**

0.25	
0.5	**0**
0.0	**1**

$(11.25)_{10}=(1011.01)_2$

Examples

- To base 8
  - To convert $(99.25)_{10}$ to octal
    - For whole number $(99)_{10}$ – division **(by 8)**

99	**3**
12	**4**
1	**1**
0	

    - For fraction $(0.25)_{10}$ – multiplication **(by 8)**

0.25	
0.0	**2**

$(99.25)_{10}=(143.2)_8$

Examples

- To base 16
  - To convert $(99.25)_{10}$ to hexadecimal
    - For whole number $(99)_{10}$ – division **(by 16)**

99	**3**
6	**6**
0	

    - For fraction $(0.25)_{10}$ – multiplication **(by 16)**

0.25	
0.0	**4**

$(99.25)_{10}=(63.4)_{hex}$

Number conversion

- Between binary and octal
  - Direct mapping based on the observation:

$$
\begin{aligned}
&(\mathbf{abcdefgh.jklmn})_2 \\
&= (a\cdot 2+b)\cdot 2^6 + (c\cdot 2^2 + d\cdot 2 + e)\cdot 2^3 + \\
&\quad (f\cdot 2^2 + g\cdot 2 + h) + (j\cdot 2^2 + k\cdot 2 + l)\cdot 2^{-3} + \\
&\quad (m\cdot 2^2 + n\cdot 2 + 0)\cdot 2^{-6} \\
&= (0ab_2)\cdot 8^2 + (cde_2)\cdot 8^1 + (fgh_2)\cdot 8^0 + \\
&\quad (jkl_2)\cdot 8^{-1} + (mn0_2)\cdot 8^{-2}
\end{aligned}
$$

  - The expressions in parentheses, being less than 8, are the octal digits.

Number conversion

- Between binary and octal (cont.)
  - Binary to octal
    - The binary digits (“bits”) are grouped from the radix point, three digits a group. Each group corresponds to an octal digit.
  - Octal to binary
    - Each of octal digits is expanded to three binary digits

Examples

- Binary to octal
  - Convert $(10101100011010001000.10001)_2$ to octal :

    $010\ 101\ 100\ 011\ 010\ 001\ 000\ .\ 100\ 010_2$

    $= \quad 2 \quad 5 \quad 4 \quad 3 \quad 2 \quad 1 \quad 0 \quad . \quad 4 \quad 2_8$

    $= \ 2543210.42_8$.
- Note:
  - Whole number parts are grouped from right to left. The leading 0 is optional
  - Fractional parts are grouped from left to right and padded with 0s

Examples

- Octal to binary
  - Convert 37425.62_8 to binary :

$$\begin{array}{ccccccccc} 3 & 7 & 4 & 2 & 5 & . & 6 & 2_8 \end{array}$$

$$= 011\ 111\ 100\ 010\ 101\ .\ 110\ 010_2$$

$$= 11111100010101.11001_2$$

- Note:
  - For whole number parts, the leading 0s can be omitted.
  - For fractional parts, the trailing 0s can be omitted.

Number conversion

- Between binary and hexadecimal
  - Binary to hexadecimal
    - The binary digits ("bits") are grouped from the radix point, **four** binary digits a group. Each group corresponds to a hexadecimal digit.
  - Hexadecimal to binary
    - Each of hexadecimal digits is expanded to four binary digits

Examples

- Binary to hexadecimal
  - Convert $10101100011010001000.10001_2$ to hexadecimal :

    $1010\ 1100\ 0110\ 1000\ 1000\ .\ 1000\ 1000_2$

    $= \quad A \quad C \quad 6 \quad 8 \quad 8 \quad . \quad 8 \quad 8 \quad _{16}$

    $= \quad AC688.88_{16}$.
- Note:
  - Whole number parts are grouped from right to left. The leading 0 is optional
  - Fractional parts are grouped from left to right and padded with 0s

Examples

- Hexadecimal to binary
  - Convert $2F6A.78_{16}$ to binary :

$$
\begin{array}{ccccccc}
 & 2 \quad F \quad 6 \quad A \; . \; 7 \quad 8_{16} \\
= & 0010\ 1111\ 0110\ 1010\ .\ 0111\ 1000_2 \\
= & 10111101101010.01111_2
\end{array}
$$

- Note:
  - For whole number parts, the leading 0s can be omitted.
  - For fractional parts, the trailing 0s can be omitted.

Conversion to binary via octal

The direct conversion of 2001_{10} to binary looks like this ...

2001	
1000	1
500	0
250	0
125	0
62	1
31	0
15	1
7	1
3	1
1	1
0	1

... and gives 11111010001.

It may be quicker to convert to octal first ...

2001	
250	1
31	2
3	7
0	3

... yielding **3721_8**, which can be instantly converted to **$11\ 111\ 010\ 001_2$**.

Binary arithmetic operations

- Similar to decimal calculations
- Examples of addition and multiplication are given in the next two slides.

Binary additions

- Example:
  - Addition of two 4-bit unsigned binary numbers. How many bits are required for holding the result?

    1001+0110 = (__________)

Binary multiplications

- Example:
  - Multiplication of two 4-bit unsigned binary numbers. How many bits are required for holding the result?

1001*0110 = (_______________________)

Negative numbers & subtraction

- Subtraction can be defined as addition of the additive inverse:

$$a - b = a + (-b)$$

- To elimination subtraction in binary arithmetic, we can represent –b by 2's complement of b.
- In n-digit binary arithmetic, 2's complement of b is

$$b^* = 2^n - b$$

  - $(b^*)^* = b$
  - The MSB (Most Significant Bit) of a 2's complement number is the sign bit
    - For example, for a 4-bit 2's complement system,
    - (1001) → -7,   (0111) → 7

Examples

-- 2's complement numbers

- Represent the following decimal numbers using 8-bit 2's complement format
  (a) 0
  (b) 7
  (c) 127
  (d) -12

- Can all the above numbers be represented by 4-bits?

Examples
4-bit 2's-complement additions/subtractions

(1) 0101 - 0010 (5 - 2):

0101
\+ 1110 (= 0010*)
= 10011

(2) 0010 - 0101 (2 - 5):

0010
\+ 1011 (= 0101*)
= 1101 (= 0011*).

Result means -3.

(3) -0101 - 0010 (-5 - 2):

1011 (= 0101*)
\+ 1110 (= 0010*)
= 11001

Result is 0111* *(how?)*

and means -7.

(4) 0101 + 0010 (5 + 2):

This is trivial, as no conversions are required. The result is 0111 (= 7).

Overflow in two’s-complement

- Assume a, b are positive numbers in an n-bit 2’s complement systems,
  - For a+b
    - If $a+b > 2^{n-1} - 1$, then a+b represents a negative number; this is **positive overflow**.
  - For -a-b
    - If $–a-b < 2^{n}-1$, then –a-b results in a positive number; this is **negative overflow**.

Positive overflow detection

Addition of 4-bit positive numbers without overflow looks like this:

```
  0xxx
+ 0xxx
= 0xxx .
```

The “carry in” to the MSB must have been 0, and the carry out is 0.

Positive overflow looks like this:

```
  0xxx
+ 0xxx
= 1xxx .
```

The “carry in” to the MSB must have been 1, but the carry out is 0.

Overflow occurs when $\boxed{\text{carry in} \neq \text{carry out.}}$

Negative overflow detection

Addition of negative twos-complement numbers without overflow:

```
   1xxx
+  1xxx
= 11xxx .
```

The carry in to the MSB must have been 1 (otherwise the sum bit would be 0), and the carry out is 1.

Negative overflow:

```
   1xxx
+  1xxx
= 10xxx .
```

The carry in to the MSB must have been 0, but the carry out is 1.

So negative overflow, like positive, occurs when $\boxed{\text{carry in} \neq \text{carry out.}}$

Overflow detection

- For n-bit 2's complement systems, condition of overflow for both addition and substraction:
  - The MSB has a carry-in different from the carry-out

Examples

1. Do the following calculations, where all numbers are 4-bit 2's complement numbers. Check whether there is any overflow.
   (a) 1000-0001
   (b) 1000+0101
   (c) 0101+0110

Floating point numbers

- Example

  Exponent

  1.01×2^{-12}

  Integer

  Binary point

  Radix (base)

- Normal Form

  $+(-)\ 1.x * 2^{y}$

  sign bit significand exponent

  - Things to be encoded:
    - sign bit
    - significant
    - exponent

IEEE 754 FP standard—single precision

- Bit 31 for sign
  - S=1 for negative numbers, 0 for positive numbers
- Bits 23-30 for biased exponent
  - The real exponent = E –127
  - 127 is called bias.
- Bits 0-22 for significand

IEEE 754 FP Standard—Single Precision (cont.)

The value, v, of the FP representation is determined as follows:

- If $0<E<255$ then $V=(-1)^S * 2^{E-127} * 1.F$
  - where "1.F" is intended to represent the binary number created by prefixing F with an implicit leading 1 and a binary point.
- If E = 255 and F is nonzero, then V=NaN ("Not a number")
- If E = 255 and F is zero and S is 1, then V= -Infinity
- If E = 255 and F is zero and S is 0, then V=Infinity
- If E = 0 and F is nonzero, then $V=(-1)^S * 2^{-126} * 0.F$. These are unnormalized numbers or subnormal numbers.
- If E = 0 and F is 0 and S is 1, then V=-0
- If E = 0 and F is 0 and S is 0, then V=0

IEEE 754 FP Standard—Single Precision (cont.)

- Subnormal numbers reduce the chance of underflow.
  - Without subnormal numbers, the smallest positive number is 2^{-127}
  - With subnormal numbers, the smallest positive number is $0.00000000000000000000001 * 2^{-126} = 2^{-(126+23)} = 2^{-149}$

Floating point additions

Given two decimal values

a = 12.025

b = 9.5

(a) What are their IEEE format representations?

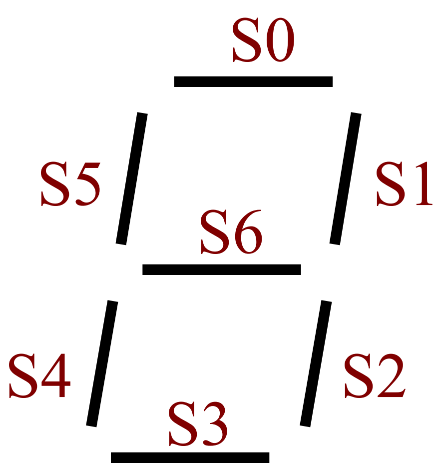
(b) How to calculate a+b in the IEEE format? And what is the result?

Encoding

- Beside numbers, a computer machine needs to represent all types of information it is to process.
- Examples:

Example 1

- To encode a decimal digit with 7 digits for 7-segment display



A	B	C	D	S_0	S_1	S_2	S_3	S_4	S_5	S_6
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1
1	0	1	0	x	x	x	x	x	x	x
1	0	1	1	x	x	x	x	x	x	x
1	1	0	0	x	x	x	x	x	x	x
1	1	0	1	x	x	x	x	x	x	x
1	1	1	0	x	x	x	x	x	x	x
1	1	1	1	x	x	x	x	x	x	x

Example 2

- To encode the locations in a memory. Assume the memory size is 2KB with (2 Bytes/location).
  - 1000 locations
  - Binary encoding
    - 10-bit

Binary codes for decimal digits

- Can be coded with 4-bit binary numbers
- Common ones:

BCD

Decimal	8,4,2,1	Excess3	8,4,-2,-1	Gray
0	0000	0011	0000	0000
1	0001	0100	0111	0100
2	0010	0101	0110	0101
3	0011	0110	0101	0111
4	0100	0111	0100	0110
5	0101	1000	1011	0010
6	0110	1001	1010	0011
7	0111	1010	1001	0001
8	1000	1011	1000	1001
9	1001	1100	1111	1000

ASCII

- American Standard Code for Information Interchange.
- Enable computers and computer programs to exchange information
- Provide 256 codes
  - Standard
  - Extended
- Nearly every computer uses American Standard Code for Information Interchange (ASCII)

ASCII

No.	char	No.	char	No.	char	No.	char	No.	char	No.	char
32		48	0	64	@	80	P	96	`	112	p
33	!	49	1	65	A	81	Q	97	a	113	q
34	"	50	2	66	B	82	R	98	b	114	r
35	#	51	3	67	C	83	S	99	c	115	s
...		...		...		...		...		...	
47	/	63	?	79	O	95	_	111	o	127	DEL

- Uppercase + 32 = Lowercase (e.g, B+32=b)
- tab=9, carriage return=13, backspace=8, Null=0

Strings

- Characters normally combined into strings, which have variable length
  - e.g., “student@unsw.edu.au”
- How to represent a variable length string?
  - 1st position of string reserved for length of string (Pascal)
  - an accompanying variable has the length of string (as in a structure)
  - last position of string is indicated by a character used to mark end of string
- C uses 0 (Null in ASCII) to mark the end of a string
  - How to represent “PASS”?

Reading Material

- Appendix A in Microcontrollers and Microcomputers.

Questions

1. Do the following calculations by changing the given decimal numbers to 8-bit 2's complement numbers and then performing the indicated operation on the 2's complement numbers. Were there any 2's complement overflows?
   (a) (+127)+(-127)
   (b) (-50)-(-100)
   (c) (+75)+(126)

Questions

2. How to represent number (-13)

- (a) in 8-bit 2's complement format (what is the minimum number bits required for such number?)
- (b) IEEE 32-bit PF format

Questions

3. Find the equivalent numbers.

(a) $(11111111)_{2\text{'s complement}} = (\_\_\_\_\_\_\_\_)_{Hex}$

(b) $25_{hex} = (\_\_\_\_\_\_\_\_\_\_\_\_)_2$

(c) $(01110011)_{BCD} = (\_\_\_\_\_\_\_\_\_\_\_\_\_\_)_2$

(d) $(11000011)_2 = (\_\_\_\_\_\_\_\_\_\_\_\_\_)_8$

Questions

4. How many bits do you need to represent a~z 26 letters and 0~9 ten digits? Why?
5. A 32-bit address is given in hexadecimal format 0X2468BAFF, what is the address in binary form?
7. Convert the 8-bit two’s complement numbers 0100110 and 11001110 to the equivalent 16-bit two’s complement numbers.

Some Programming Examples

```
/* Example 1: Reading a value from a memory location
           and write that back into another location
*/


/* The header file to include */

.include "m64def.inc"


ldi r16,10        //Loading a value of 10 into register r16

sts 0x000106,r16   //Storing the value in r16 into the memory location 000106

lds r17,0x000106   //Loading the value in 000106 into register r17

sts 0x00010C,r17   //Storing the value in r17 into the memory location 00010C

loop: rjmp loop    //An infinite loop to end the program for AVR....


/* The values can be observed in the Memory window....View ->Memory ->data*/
```

```
/* Example2: Copy an Array into another array..
          The array is considered as a sequence of memory blocks..*/

/* The header file to include */
.include "m64def.inc"
/****  Storing some values into the array memory locations *****/
ldi r16,10
sts 0x000100,r16
ldi r16,11
sts 0x000101,r16
ldi r16,12
sts 0x000102,r16
ldi r16,13
sts 0x000103,r16
ldi r16,14
sts 0x000104,r16
/*****Copying the array from memory into another array *********/
lds r16,0x000100
sts 0x000105,r16
lds r16,0x000101
sts 0x000106,r16
lds r16,0x000102
sts 0x000107,r16
lds r16,0x000103
sts 0x000108,r16
lds r16,0x000104
sts 0x000109,r16
loop: rjmp loop    //infinte loop...
```

```
/* Example3: Add two arrays and store it a new array..
        The array is considered as a sequence of memory blocks....
*/
/* The header file to include */
.include "m64def.inc"
/****  Storing some values into the first array....*****/
ldi r16,10
sts 0x000100,r16
ldi r16,11
sts 0x000101,r16
ldi r16,12
sts 0x000102,r16
ldi r16,13
sts 0x000103,r16
ldi r16,14
sts 0x000104,r16
/****  Storing some values into the second array....*****/
ldi r16,15
sts 0x000105,r16
ldi r16,16
sts 0x000106,r16
ldi r16,17
sts 0x000107,r16
ldi r16,18
sts 0x000108,r16
ldi r16,19
sts 0x000109,r16
```

Example3 – cont'd

```
/* Retrieving the values from the array add the values together
   and write them back into another array
*/
lds r16,0x000100    //value taken from the first array
lds r17,0x000105    //value taken from the second array
add r16,r17         //adding the values together
sts 0x00010A,r16    //storing the value into another array

lds r16,0x000101
lds r17,0x000106
add r16,r17
sts 0x00010B,r16

lds r16,0x000102
lds r17,0x000107
add r16,r17
sts 0x00010C,r16

lds r16,0x000103
lds r17,0x000108
add r16,r17
sts 0x00010D,r16

lds r16,0x000104
lds r17,0x000109
add r16,r17
sts 0x00010E,r16

loop: rjmp loop
```

Example 4 –
First part same as for Example 3.

```
/* Retrieving the values from the array add the values together
   and write them back into another array
*/
lds r16,0x000100    //value taken from the first array
lds r17,0x000105    //value taken from the second array
add r16,r17         //adding the values together
sts 0x00010A,r16    //storing the value into another array
adc r18,r0          //carry is added to register r18
lds r16,0x000101
lds r17,0x000106
add r16,r17
sts 0x00010B,r16
adc r18,r0
lds r16,0x000102
lds r17,0x000107
add r16,r17
sts 0x00010C,r16
adc r18,r0
lds r16,0x000103
lds r17,0x000108
add r16,r17
sts 0x00010D,r16
adc r18,r0
lds r16,0x000104
lds r17,0x000109
add r16,r17
sts 0x00010E,r16
adc r18,r0

loop: rjmp loop    //infinite loop...
```