UECS1013 Introduction to Computer Organisation and Architecture

2014/May/Lecture 2

Outline: Number System

- **Conversion Between Number Systems**
  - Conversion from Decimal (Base 10) to other Bases
  - Sum of Weights Technique
  - Radix Divide and Multiply Technique
  - Special Conversion Cases: Related Number Bases
- **Manual Arithmetic in Different Number Bases**
  - Addition
  - Multiplication
  - Subtraction
  - Division
- **Computational Arithmetic**
  - Sign and Magnitude
  - 9's and 10's Decimal Complement
  - 1's and 2's Binary Complement
  - … To be continued

CONVERSION BETWEEN NUMBER SYSTEMS

3

Conversion from Decimal (Base 10) to Other Bases

There are two different methods to perform conversion from Decimal (Base 10) to other bases:

- Method 1: *Sum of Weights Technique*
- Method 2: *Radix Divide and Multiply Technique*
  - *Repeated division for integral part*
  - *Repeated multiplication for fractional part*

4

Sum of Weights

- [Recap] Converting a number (base 8) to base 10

  Number = 13754_8
  Value → 1 3 7 5 4_8
  Weight → 4096 512 64 8 1
  Number = 4096 + 1536 + 448 + 40 + 4
  = 6124_{10}

- Converting from base 10 to another base
  - Use the same method in reverse (not so simple)
  - Method: Find the values at each position each with different weights such that the total will add up to the base 10 number that we are trying to convert

Sum of Weights

Integral Part

Step1: Start from the first position from the left with a weight smaller than the decimal number.

Step2: Divide the remainder (initially the integral part of the original number) by the current weight to generate the remainder and quotient.

Step3: Go to next position and repeat step 2 until the first integral digit.

The value at the quotient is our result

Fractional Part

Repeat step1 to step 3 but for fractional part

28.75_{10} = 11100.11_2

Position	5	4	3	2	1	0	-1	-2
Weight (Base 2)	32	16	8	4	2	1	0.50	0.25
Division		**28**/16	12/8	4/4	0/2	0/1	**0.75** /0.5	0.25/0.25
Quotient		1	1	1	0	0	1	1
Remainder		12	4	0	0	0	0.25	0

Sum of Weights

- Example: Convert 6124_{10} to Base 5

$6124_{10} = 143,444_5$

Position	5	4	3	2	1	0	0
Weight (Base 5)	15625	3125	625	125	25	5	1
Division		6124/ 3125	2999/ 625	499/ 125	124/ 25	24/5	4/1
Quotient		1	4	3	4	4	4
Remainder		2999	499	124	24	4	0

7

Sum of Weights

- Example: Convert 3193_{10} to binary

Position	Weights	Division	Quotient	Remainder
12	4096			
11	2048	3193/2048	1	1145
10	1024	1145/1024	1	121
9	512		0	
8	256		0	
7	128		0	
6	64	121/64	1	57
5	32	57/32	1	25
4	16	25/16	1	9
3	8	9/8	1	1
2	4		0	1
1	2		0	1
0	1	1/1	1	0

Answer: $3193_{10} = 110001111001_2$

8

Radix Divide and Multiply

Division-by-Base (Base 2 example)

- For the ***integral part***, write down a sequence of bits, determined by the following procedure, to the left of the radix point:
  a) Divide the number by 2.
  b) If the result in non-integral, write down a 1. Otherwise write down a 0.
  c) Retain the integral part of the result.
  d) If it is non-zero, go back to step 1 and keep writing ***to the left*** of the previously written sequence.

9

Radix Divide and Multiply

Multiplication By Base (Base 2 example)

- For the ***fractional part***, write down a sequence of bits, determined by the following procedure, to the right of the radix point:
  a) Multiply the number by 2.
  b) If the integral part of the result is 1, write down a 1. Otherwise write down a 0.
  c) Retain the fractional part of the result.
  d) If it is non-zero, go back to step 1 and keep writing ***to the right*** of the previously written sequence.

10

Radix Divide and Multiply

- Example: Convert decimal number 28.625 into a binary number.

$28/2 = 14.0 \rightarrow$ 0.
$14/2 = 7.0 \rightarrow$ 00.
$7/2 = 3.5 \rightarrow$ 100.
$3/2 = 1.5 \rightarrow$ 1100.
$1/2 = 0.5 \rightarrow$ 11100.

Integral part: append to the left

$0.625 \times 2 = 1.25 \rightarrow$ .1
$0.25 \times 2 = 0.5 \rightarrow$ .10
$0.5 \times 2 = 1.0 \rightarrow$ .101

Fractional part: append to the right

Therefore, $28.75_{10} = 11100.101_2$.

- Will the procedure always stop? Try 0.2_{10}
- Example: Convert 13.705_{10} to binary.

11

Radix Divide and Multiply

Example:

a) Convert 250.25_{10} to hexadecimal
$250 / 16 = 15$ remainder of 10 $\rightarrow$ A
$15 / 16 = 0$ remainder of 15 $\rightarrow$ F
$0.25 \times 16 = 4.00$ $\rightarrow$ 4
The result = $FA.4_{16}$

a) Convert 90.0625_{10} to Octal
$90 / 8 = 11$ remainder of 2 $\rightarrow$ 2
$11 / 8 = 1$ remainder of 3 $\rightarrow$ 3
$1/8 = 0$ remainder of 1 $\rightarrow$ 1
$0.0625 \times 8 = 0.5$ $\rightarrow$ 0
$0.5 \times 8 = 4.0$ $\rightarrow$ 4
The result = 132.04_8

12

Special Conversion Cases: Related Number Bases

Binary (k=1)	Octal (k=3)	Decimal	Hexadecimal (k=4)
0 000	0	0	0
0 001	1	1	1
0 010	2	2	2
0 011	3	3	3
0 100	4	4	4
0 101	5	5	5
0 110	6	6	6
0 111	7	7	7
1 000	10	8	8
1 001	11	9	9
1 010	12	10	A
1 011	13	11	B
1 100	14	12	C
1 101	15	13	D
1 110	16	14	E
1 111	17	15	F

- The family of base 2^k number systems is basically compatible with each other.
- One complete permutation of 3 binary digits ≡ one complete round of octal set 0_8 to 7_8
- One complete permutation of 4 binary digits ≡ one complete round of hexadecimal set $0_{16}$to F_{16}
- Conclusion: conversion between base 2_k number systems can be easily performed by grouping the digits

13

Special Conversion Cases: Related Number Bases

1. **Hexadecimal to Binary Conversion**

   Convert $47.FE_{16}$ to binary

4	7	.	F	E
0100	0111	.	1111	1110

   $47.FE_{16}$ = 0100 0111 . 1111 1110_2

2. **Binary to Hexadecimal Conversion**

   Convert 10010.011011_2 to hexadecimal

0001	0010	.	0110	1100
1	2	.	6	C

   10010.011011_2 = $12.6C_{16}$

14

Special Conversion Cases: Related Number Bases

3. Octal to Binary Conversion

Convert 47.12_8 to binary

```
 4     7     .   1     2
100   111    .  001   010
```

47.12_8 = 10 0111.0010 1000_2

4. Binary to Octal Conversion

Convert 10010.011011_2 to octal

```
010 010    .    011 011
  2    2   .      3    3
```

$10010.011011_2 = 22.33_8$

15

Special Conversion Cases: Related Number Bases

6. Hexadecimal to Octal Conversion

Step 1: Convert Hex (or Octal) to binary

Step 2: Convert from binary to Octal (or Hex)

Example:

Convert 47.12_8 to hexadecimal

4 7 . 1 2_8		
100 111 . 001 010_2	(Convert to binary)	
10 0111. 0010 1000_2	(Regroup bits)	
2 7. 2 8_{16}	(Convert to hex)	

16

Review Questions

- **Convert the following number into Base 10.**

  $16AF_{16}$
- **Convert the following the following numbers into base 2**

  25.6_{10}
- **Convert the following the numbers into base 2**

  $6A_{16}$, 17_8
- **Convert the following the numbers into base 16**

  $1111\ 1011_2$, 67_8

17`

Review Questions

Base 10 Addition Table

	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9	10
2	2	3	4	5	6	7	8	9	10	11
3	3	4	5	6	7	8	9	10	11	12
4	4	5	6	7	8	9	10	11	12	13
5	5	6	7	8	9	10	11	12	13	14
6	6	7	8	9	10	11	12	13	14	15
7	7	8	9	10	11	12	13	14	15	16
8	8	9	10	11	12	13	14	15	16	17
9	9	10	11	12	13	14	15	16	17	18

- The table contains the answer when adding two ***single digits***
- Example: $3_{10} + 6_{10} = 9_{10}$ is given by the intersection of row 3 and column 6
- Addition of two single digits may generate a two digit result, e.g. ,

  $5_{10} + 8_{10} = 13_{10}$ where 1 is the ***carry digit*** and 3 is the ***sum digit***
- Addition of multiple digits will be handled later.

18

MANUAL ARITHMETIC IN DIFFERENT NUMBER BASES

Addition In Different Number Bases

Base 2 addition table

	0	1
0	0	1
1	1	10

Base 8 addition table

	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	10
2	2	3	4	5	6	7	10	11
3	3	4	5	6	7	10	11	12
4	4	5	6	7	10	11	12	13
5	5	6	7	10	11	12	13	14
6	6	7	10	11	12	13	14	15
7	7	10	11	12	13	14	15	16

$1_2 + 1_2 = 10_2$
equivalent to
$1_{10} + 1_{10} = 2_{10}$

$3_8 + 6_8 = 11_8$
equivalent to
$3_{10} + 6_{10} = 9_{10}$

Addition In Different Number Bases

Base 16 addition table

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10
2	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11
3	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12
4	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13
5	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14
6	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15
7	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16
8	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17
9	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18
A	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19
B	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A
C	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B
D	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C
E	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D
F	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E

$E_{16} + D_{16} = 1B_{16}$ or equivalently $14_{10}+13_{10} = 27_{10}$

21

Addition In Different Number Bases

Method 1: By using the addition table

		1	2	3
+		D	E	F
Carry	0	1	1	
Sum +		E	0	2
Carry	0	0	0	
Sum		F	1	2

From addition table:

$3_{16} + F_{16} = 12_{16}$ → 1: carry, 2: sum

$2_{16} + E_{16} = 10_{16}$ → 1: carry, 0: sum

$1_{16} + D_{16} = 0E_{16}$ → 0: carry, E: sum

2_{16} → 0: carry, 2: sum

$1_{16} + 0_{16} = 1_{16}$ → 0: carry, 1: sum

$1_{16} + E_{16} = F_{16}$ → 0: carry, F: sum

$$123_{16} + DEF_{16} = F12_{16}$$

22

Addition In Different Number Bases

Method 2: By using an extended addition table

Extending the power table to 3 variables:
Recommended for adding two binary numbers.

	0	0	0	0	1
Addend	0	0	1	1	1
Augend	+0	+1	+0	+1	+1
Sum	0	1	1	0	1
Carry	0	0	0	1	1

The addition table in binary.

Find 1 0010 0011_2 + 1101 1110 1111_2

Augend		**1**	**0 0 1 0**	**0 0 1 1**
Addend		**1 1 0 1**	**1 1 1 0**	**1 1 1 1**
Carry	**+**	**1 1**	**1 1 0 1**	**1 1 1**
Sum		**1 1 1 1**	**0 0 0 1**	**0 0 1 0**

23

Multiplication In Different Number Bases

Base 10 Multiplication Table

	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9
2	0	2	4	6	8	10	12	14	16	18
3	0	3	6	9	12	15	18	21	24	27
4	0	4	8	12	16	20	24	28	32	36
5	0	5	10	15	20	25	30	35	40	45
6	0	6	12	18	24	30	36	42	48	54
7	0	7	14	21	28	35	42	49	56	63
8	0	8	16	24	32	40	48	56	64	72
9	0	9	18	27	36	45	54	63	72	81

- Provides the result of multiplying two single digit numbers
- Example: 3_{10} x 6_{10} = 18_{10}
- Product of multiple digits will be handled later.

24

Multiplication In Different Number Bases

Base 2 Multiplication Table

	0	1
0	0	0
1	0	1

$1_2 \times 1_2 = 1_2$

Base 8 Multiplication Table

	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	10	12	14	16
3	0	3	6	11	14	17	22	25
4	0	4	10	14	20	24	30	34
5	0	5	12	17	24	31	36	43
6	0	6	14	22	30	36	44	52
7	0	7	16	25	34	43	52	61

(no digit 8 or 9, of course)

$3_8 \times 6_8 = 22_8$ equivalent to $3_{10} \times 6_{10} = 18_{10}$

Multiplication In Different Number Bases

Base 16 Multiplication table

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2	0	2	4	6	8	A	C	E	10	12	14	16	18	1A	1C	1E
3	0	3	6	9	C	F	12	15	18	1B	1E	21	24	27	2A	2D
4	0	4	8	C	10	14	18	1C	20	24	28	2C	30	34	38	3C
5	0	5	A	F	14	19	1E	23	28	2D	32	37	3C	41	46	4B
6	0	6	C	12	18	1E	24	2A	30	36	3C	42	48	4E	54	5A
7	0	7	E	15	1C	23	2A	31	38	3F	46	4D	54	5B	62	69
8	0	8	10	18	20	28	30	38	40	48	50	58	60	68	70	78
9	0	9	12	1B	24	2D	36	3F	48	51	5A	63	6C	75	7E	87
A	0	A	14	1E	28	32	3C	46	50	5A	64	6E	78	82	8C	96
B	0	B	16	21	2C	37	42	4D	58	63	6E	79	84	8F	9A	A5
C	0	C	18	24	30	3C	48	54	60	6C	78	84	90	9C	A8	B4
D	0	D	1A	27	34	41	4E	5B	68	75	82	8F	9C	A9	B6	C3
E	0	E	1C	2A	38	46	54	62	70	7E	8C	9A	A8	B6	C4	D2
F	0	F	1E	2D	3C	4B	5A	69	78	87	96	A5	B4	C3	D2	E1

$E_{16} \times D_{16} = B6_{16}$ $(14_{10} \times 13_{10} = 182_{10})$

Multiplication In Different Number Bases

Find the product of two hex integers 123 and DEF

Step 1: Break down the second multiplier into single digits

123 x DEF = 123 x (D00 + E0 + F)

= **(123 x D)** x 100 + **(123 x E)** x 10+ **(123 x F)** x 1

Step 2: Find the product in parenthesis individually

	1	2	3
X			D
		2	7
	1	A	
	D		
	E	C	7

	1	2	3
X			E
		2	A
	1	C	
	E		
	F	E	A

	1	2	3
X			F
		2	D
	1	E	
	F		
1	1	0	D

Step 3: Sum up the individual products

123xDEF = EC7 x 100 + FEA x 10 + 110D

= EC700 + FEA0 + 110D = FD6AD

Multiplication In Different Number Bases

- **Example**: Find the product of two hex integers 123 and DEF

The three steps can be simplified into a single step as follows

		1	2	3	Multiplicand
X		D	E	F	Multiplier
	1	1	0	D	(123xF)
	F	E	A		(123xE)
E	C	7			(123xD)
F	D	6	A	D	

Multiplication In Different Number Bases

- **Example**: Find the product of two binary 1101 x 101

			1	1	0	1	
				1	0	1	
			1	1	0	1	Position 0 (1st digit)
			0	0	0		Position 1 (2nd digit)
	1	1	0	1			Position 2 (3rd digit, first operand *shifted* by 2 bits)
1	0	0	0	0	0	1	Add

redundant

29

Subtraction In Different Number Bases

- Similar to Base 10. However a borrow is required from its immediate left digit, the base value is borrowed.
- **Example**: $1100\ 1011_2 - 0110\ 1110_2$

Equals to 2 in Base 10

Borrow			10	01	01	10			
		0	~~0~~	~~10~~	~~10~~	~~0~~	2		
Minuend		~~1~~	~~1~~	~~0~~	~~0~~	~~1~~	0	1	1
Subtrahend	-	0	1	1	0	1	1	1	0
Result		0	1	0	1	1	1	0	1

Notes: Borrow, Minuend, Subtrahend and Result are expressed in the corresponding base (Base 2 in this example)

30

Subtraction In Different Number Bases

- **Example**: 10 1101_2 – 01 1010_2

Minuend		1	0	1	1	0	1
Subtrahend	-	0	1	1	0	1	0
		0	1	0	0	1	1

- **Example**: $5475_8 - 3764_8$

Equals to 12 in Base 10

Borrow		4	14		
Minuend		~~5~~	~~4~~	7	5
Subtrahend	-	3	7	6	4
		1	5	1	1

31

Subtraction In Different Number Bases

- **Example**: $245D_{16} - 15FC_{16}$

Equals to 19 in Base 10

Equals to 21 in Base 10

Borrow		1	13 ~~3~~	15	
Minuend		~~2~~	~~4~~	5	D
Subtrahend	-	1	5	F	C
			E	6	1

32

Division In Different Number Bases

- Division of Z / Y where Z is the dividend and Y is the divisor
- Steps are similar to Base 10:
  1. Align the divisor (Y) with the most significant end of Z. Let the portion of the dividend to its bit aligned with the LSB of the divisor be denoted as X.
  2. Compare X and Y.
     - If X ≥ Y, the quotient digit is $\lfloor X/Y \rfloor$ and perform X – Y.
     - If X < Y, the quotient digit is 0
  3. Shift Y by ONE bit to the right and go to step 2.

33

Division In Different Number Bases

- **Example**: $1100\ 1011_2$ / $0110\ 1110_2$

```
                0  1  1  1          Quotient
          ------------------------------------
 1  1  1 | 1  1  0  1  0  1
           0  0  0
          ------------------------------------
           1  1  0  1
              1  1  1
          ------------------------------------
              1  1  0  0
                 1  1  1
                 ---------------------------
                 1  0  1  1
                    1  1  1
                 ---------------------------
                    1  0  0          remainder
```

34

Division In Different Number Bases

- **Example**: 543_8 / 7_8

	0	6	2	Quotient
7	5	4	3	
	0			
	5	4		
	5	2		
		2	3	
		1	6	
			5	Remainder

Division In Different Number Bases

- **Example**: $1AF3_{16}$ / E_{16}

	0	1	E	C	Quotient
E	1	A	F	3	
	0				
	1	A			
	0	E			
		C	F		
		C	4		
			B	3	
			A	8	
				B	Remainder

Review Question

- Perform the following operation

  $1476_8 + 3554_8$ (answer = 5252_8)

  $127_8 + 29_{16}$

- Find the following operation

  $1001101_2 \times 100110_2$

  $1E4A_{16} \times FA2_{16}$ (answer = $1D980D4_{16}$)

- Find the product of two hexadecimal numbers 22_{16} and 67_{16}

37

Review Question

- Perform the following operation

  $540045_8 - 325654_8$ (answer = 212171_8)

  $540045_{16} - 325654_{16}$ (answer = $21A9F1_{16}$)

38

COMPUTATIONAL ARITHMETIC

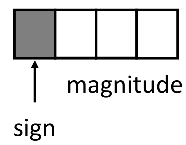
Representation for Signed Integers

- How do computer represent negative numbers (signed numbers) and real numbers?
- Numbers can be represented as a combination of
  - Value or magnitude
  - Sign (plus or minus)
- To handle **negative numbers**, the following systems are used for **signed** numbers:
  - 2's-complement (most commonly used)
  - 1's-complement
  - Sign-and-magnitude
- To handle **real numbers**
  - Floating Points (float, double)

Sign and Magnitude (SM)

- In daily usage, negative numbers are usually written by writing a minus sign in front.
  - Example: $-(12)_{10}$, $-(1100)_2$
- In sign-and-magnitude representation, this sign is usually represented by a bit:

  0 represents + and **1** represents -
- For example, a 4-bit number can have 1-bit sign and 3-bit magnitude.
  - **0**010$_{SM}$ represents 2 for S&M
  - **1**010$_{SM}$ represents -2 for S&M



- Note that the same bit pattern (1010) have different value in SM and other representations
  - 1010$_{SM}$ represents -2 for **S&M** but 6 for **unsigned binary**

Sign and Magnitude (SM)

- The range of an *n*-bit number ranges from

  $-(2^{(n-1)}-1)$ to $+(2^{(n-1)}-1)$
- For example, the range of numbers represented by a 4-bit number in SM is from

  Largest Positive Number:

  0 111 = $+(7)_{10}$

  Largest Negative Number:

  1 111 = $-(7)_{10}$
- Two zeroes:

  0 000 = $+(0)_{10}$

  1 000 = $-(0)_{10}$

bit pattern	number represented
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	-0
1001	-1
1010	-2
1011	-3
1100	-4
1101	-5
1110	-6
1111	-7

All the numbers that can be represented by 4-bit binary number in SM

Sign and Magnitude (SM)

- To negate a number, just invert the sign bit. For examples:

  $-0100001_2 = 1\ 0100001_{SM}$

  $-0000101_2 = 0\ 0000101_{SM}$
- Difficulty in implementation: the actual methods of arithmetic used depends on the signs of the operands and the *relative* magnitudes of the inputs

  4 + 2 vs 4 – 2 vs 12 - 4
- Directly mimicking how human perform calculations does not makes a good machine due to the following reasons:
  - Computer requires absolute definition of every possible condition. So, the algorithm must include every possibility
  - Calculation algorithms are complex and difficult to implement in hardware

43

The Methods of Complement

- The method of complements is a technique used to perform subtraction using only addition.
- Commonly used in modern computer and calculators
- In a complement method,
  - The first digit of the number represents a positive/ negative number
  - The positive numbers remain untouched.
  - The negative numbers are converted into its complement version.
  - **To subtract/add two numbers, simply add the two numbers**.
- Examples:
  - 9’s Complement in Decimal and equivalently the 1’s Complement in Binary
  - 10’s Complement in Decimal and equivalently the 2’s Complement in Binary

44

9's Decimal Complement

- Assume a **three digit** decimal (Base 10) number. Split the decimal into half.
- Positive numbers:
  - ❑ 0 to 499 are considered positive and they will represent itself
- Negative numbers:
  - ❑ To assign a value to the negative numbers, allow each digit to be subtracted from the largest numeral in the radix. In the case of decimal, the largest value numeral is the digit 9.

    -1_{10} is represented as (9-0=9, 9-0=9, 9-1=8 → 998_{9s})

    -499_{10} is represented as (9-4=5, 9-9=0, 9-9=0 → 500_{9s})
  - ❑ Subtracting a value from some standard base value is known as **taking the complement** of the number.

45

9's Decimal Complement

Numbers	Negative		Positive	
Range of **Decimal numbers**	***-499***	***-000***	***+0***	***499***
Representation method	Complement		Number itself	
Calculation	999 minus number		none	
9's Representation	***500***	***999***	***0***	***499***

– Increasing value +

- How to determine if a number is positive/negative?
  By looking at the first digit
  - 0 through 4 → positive number
  - 5 through 9 → negative number
- There are two ways to represent the value 0 in 9's complement:

  999_{9s} and 0_{9s}

46

9's Decimal Complement

- To convert a **positive** decimal number N into 9's representation,

  N_{10} is represented as N_{9s}
- To convert a **negative** decimal number N into 9's representation, we *take its complement* by subtracting a value from a standard basis value. Given a number *N* with *n digits*, its negation or -N when represented as 9's complement is defined as:

  $-N_{10}$ is represented as $(10^n-1-N)_{9s}$
- To convert a **positive** 9's representation number M (most significant digit <5) back to its decimal equivalent

  M_{9s} is represented as M_{10}
- To convert a **negative** 9's representation number M (most significant digit >=5) M back to its decimal equivalent

  M_{9s} is represented as $-(10^n-1-M_{SM})_{10}$ or

  M_{9s} is represented as $(M_{SM} - (10^{n-1}))_{10}$

47

9's Decimal Complement

- Example: Convert the following decimal numbers into their 3 digits 9's complement representation:

  (a) $459_{10} = 459_{9s}$ (Positive number remains unchanged)

  (b) $-459_{10} = (10^3-1) - 459 = 999 - 459 = 540_{9s}$

- Example: Convert -36_{10} into its 3 digits and 5 digits 9's complement representation:

  (a) $-36_{10} = (10^2-1) - 36 = 99 - 36 = 63_{9s}$

  (b) $-36_{10} = (10^5-1) - 36 = 99999 - 36 = 99963_{9s}$

- Example: Convert the following 9's representation into their decimal value:

  (a) $459_{9s} = 459_{10}$ (Positive number remains the same)

  (b) $540_{9s} = 540 - (10^3-1) = 540 - 999 = -459_{10}$

48

9's Decimal Complement

Steps to perform Addition/subtraction in 9's complement

1. Convert all the decimal operands into its 9's complement equivalent
   - If the #digits is specified, then extend the number of digits to the targeted number of digits

     For example, perform 119 – 20 in 4 digit, 9's complement

     $119_{10} \rightarrow 0119_{9s}$

     $-20_{10} \rightarrow 9979_{9s}$

   - If the #digits in the operation is not specified, then the number of digits follows the larger digits.

     For example, perform 119 – 20 in 9's complement

     $119_{10} \rightarrow 119_{9s}$

     $-20_{10} \rightarrow 979_{9s}$

9's Decimal Complement

Steps to perform Addition/subtraction in 9's complement (cont…)

- If the #digits in the operation is outside of the supported range, then the operation cannot be performed

  For example, perform 919 – 20 in 3 digit, 9's complement

  919_{10} is out of range of a 3 digit, 9's complement representation

2. Perform the addition with end-around carry.

   For example, 273 – 18

Base 10

$$\begin{array}{r} 273 \\ -\ \ 18 \\ \hline 255_{10} \end{array}$$

9's complement

$$\begin{array}{r} 273 \\ +\ 981 \\ \hline 1\ \ 254 \\ 1 \\ \hline 255_{9s} \end{array}$$

9's Decimal Complement

- **Addition without an end-around carry**
  - Happens when the addition does not result in the changes in the sign.
  - The addition in 9's Complement does not cross the modulus (999 for a three digit 9's complement representation)

	NEGATIVE NUMBERS				POSITIVE NUMBERS			
		+250 →				+250 →		
Base 10	-499	***-350***	***-100***	-0	0	***170***	***420***	499
9's Complement	500	***649***	***899***	999	0	***170***	***420***	499
		+250 →				+250 →		

In Base 10

-350
+250
-100

In 9's C

649
+250
899

Both in Base 10 and 9'sC

170
+250
420

51

9's Decimal Complement

- **Addition with end-around carry:**
  - The addition in 9's C crosses the modulus
  - Happens when the addition of a positive and a negative numbers generates a positive number.
  - Add the carry digit to the result (ignore the carry digit)

		+300 →			
Base 10	-499	***-200***	-000	0	***100***	499
9's Complement	500	***799***	999	0	***100***	499
		+300 →			

In Base 10

-200
+ 300
100

In 9's C

799
+ 300
1 099
+1
100

52

9's Decimal Complement

- **Subtraction**
  - 9 – 10 is implemented in terms of addition of a positive and a negative number 9 + (-10)
  - Count to the right (number of steps = 9's complement of the number, e.g. 699 steps for -300) to subtract a number
  - Subtraction will always generate a wraparound

-300

Base 10	-499	-000	0	***200***	499	-499	*-100*	-000
9's Complement	500	999	0	***200***	499	500	*899*	999

+699

In Base 10	**In 9's C**
200	200
-300	+ 699
100	899

53

9's Decimal Complement

- **Subtraction**

-2

Base 10	-499	**-199**	-000	0	499	-499	*-201*	-000
9's Complement	500	**800**	999	0	499	500	*897*	999

+997

In Base 10	**In 9's C**
-199	800
-2	+ 997
-201	1 797
	+1
	798

54

Review Questions

- Convert the following numbers into 9's complement
  - 362
  - -367
  - -177 (5 digits)
  - -479 (8 digits)

- Add the 2 numbers below using the following method.
  - 41 – 25 (9's complement)
  - 497 + 136 (9's complement)
  - 998 – 13 (3 digits, 9's complement)
  - 999 – 28 (5 digits, 9's complement)

10's Decimal Complement

- Problem with 9's complement:
  - Two representations for a single value 0
  - Extra operations when end-around carry is required
- 10's complement:
  - One single representation for a single value 0
  - No end-around carry required. The carry digit is simply ignored.

Numbers	Negative		Positive	
Range of decimal numbers	***-500***	***-001***	***0***	***499***
Representation method	Complement		Number itself	
Calculation	1000 minus number		none	
Representation example	***500***	***999***	***0***	***499***

10’s Decimal Complement

- Given a number *N* with *n digits*, its negation or -N when represented as 10’s complement is defined as:

  9’s complement + 1 = $(10^n-1) - N + 1 = 10^n - N$
- The positive number remains the same
- Example: *Find the 10’s complement of the following decimal numbers:*

  (a) 459 = 459_{10c} = 459_{9s} (Positive number remains the same)

  (b) -459 = $10^3 - 459$ = $1000 - 459$ = $(541)_{10c}$

  (c) -36 = $10^2 - 36$ = $100 - 36$ = $(64)_{10c}$

  (d) -36 (Use 5 digits) = $10^5 - 36$ = $100000 - 36$ = $(99964)_{10c}$
- Example: *Convert from 10’s complement back to decimal:*

  (a) $(459)_{10s}$ = $(459)_{10}$ (Positive number remains the same)

  (b) $(541)_{10s}$ = $541 - (10^3)$ = $541 - 1000$ = $-(459)_{10}$

10’s Decimal Complement

Addition/subtraction in 10’s Decimal Complement

Example: $273_{10} - 18_{10}$

1. Convert all the decimal operands into its 10’s complement equivalent. The same rules for conversion as 9’s complement applies.

   273 + (–018) → $273_{10} = (273)_{9s}$

   $018_{10} =$ $(10^3 - 018)_{10s} = (982)_{10s}$

2. Perform the addition and <u>ignore</u> the carry digit

<u>In Base 10</u>

$$\begin{array}{r} 273 \\ +\ 982 \\ \hline 1255_{10} \end{array}$$

<u>In 10’s</u>

$$\begin{array}{rl} 273 & \\ +\ 982 & \\ \hline 1255 & \text{(ignore carry)} \\ 255_{10s} & \end{array}$$

Review Questions

- Example 1: Get the 10's complement of
  247
  -328

- Example 2: Find 3 and 5 digits 10's complement representation of -28

- Example 3: Find the signed value of the 10's complement representation 777.

59

1's Binary Complement

- Without loss of generalization, in a 4-bit number system:

- A positive integer, X, is represented as $0b_2b_1b_0$

  Example: 3_{10} = 0011_{1s}

- A negative integer -X is represented as $\overline{0}\overline{b_2}\,\overline{b_1}\,\overline{b_0}$where $\overline{b}$ represents the **compliment** of b.

  Example: -3_{10} = -0011_2 = 1100_{1s}

- The method can be extended to any arbitrary number of bits.

  Example: Convert -6 into its 5 bit 1's Complement equivalent

  -6 = -00110_2 = -11001_{1s}

Figure: 1's Complement of a 4 bit number

bit pattern	number represented
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	-7
1001	-6
1010	-5
1011	-4
1100	-3
1101	-2
1110	-1
1111	-0

60

1's Binary Complement

- Note that the value of a 4 bit number in binary is not equivalent to actual value of the number in 1s Complement

  1000_2 is 8 in value but 1000_{1s} has a value of -7

- Formally, the number represented -*N* has the bit pattern of $(2^n-1)-N$ where n is the number of bit. Try compare with 9's C.

  Example: When N = 6, -N or -6 is represented by the bit pattern $(2^4-1)-6 = 9$ or 1001 in 1's complement

- The range that can be represented by n digits in 1's complement is

  $-(2^{(n-1)}-1)$ to $+(2^{(n-1)}-1)$

  Example: The range of a 4 bit number is $-(2^3-1)$ to $(2^3-1)-1 = -7$ to 7

61

1's Binary Complement

- Examples (assuming 8-bit binary numbers):

  $(14)_{10} = (00001110)_2 = (00001110)_{1s}$

  $-(14)_{10} = -(00001110)_2 = (11110001)_{1s}$

  $-(80)_{10} = -(\ ?\ )_2 = (\ ?\ )_{1s}$

62

1's Binary Complement

Algorithm for addition, A + B:

1. Perform binary addition on the two numbers.
2. Perform ***end-around carry***: if there is a carry out of the MSB, add 1 to the result.

Algorithm for subtraction, A – B:

A – B = A + (–B)

1. Take 1s complement of B by inverting all the bits.
2. Perform the addition operation as the steps per above

1's Binary Complement

- Examples: 4-bit binary system

```
  +3         0011
+ +4       + 0100
----       -------
  +7         0111
----       -------
```

```
  +5         0101
+ -5       + 1010
----       -------
  -0         1111
----       -------
```

```
  -2        1101
+ -5    +   1010
----      ------
  -7       10111
----     +     1
          ------
            1000
```

```
- 1         1110
- 2       + 1101
----      ------
 -3        11011
----     +     1
          ------
            1100
```

2's Binary Complement

Without loss of generalization, in a 4-bit number system:

- A positive integer, X, is represented as $0b_2b_1b_0$

  Example: $3_{10} = 0011_{2s}$
- 2's Complement and 1's Complement has the same bit pattern for positive values

  $3_{10} = 0011_{1s} = 0011_{2s}$
- A negative integer $-X$ is represented as $\overline{0b_2}\,\overline{b_1}\,\overline{b_0} + 1$

  Example: $-3_{10} = -0011_2 = 1101_{2s}$
- The method can be extended to any arbitrary number of bits.

  Example: Convert -6 into its 5 bit 1's

2's Complement of a 4 bit number

bit pattern	number represented
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	-8
1001	-7
1010	-6
1011	-5
1100	-4
1101	-3
1110	-2
1111	-1

65`

2's Binary Complement

- As in previous method, the value of a 4 bit number in binary is not equivalent to actual value of the number in 1s Complement
  - 1000_2 is 8 in value but 1000_{2s} has a value of -8
- Formally, the number represented $-N$ has the bit pattern of 2^n-N where n is the number of bit.
  - Example: when N = 6, -N or -6 is represented by the bit pattern $2^4-6 = 10$ or 1010_{2s} in 2's complement
- The range that can be represented by n digits in 1's complement is

  $-2^{(n-1)}$ to $+(2^{(n-1)}-1)$
  - Example: The range of a 4 bit number is -2^3 to $(2^3-1)-1$ = -8 to 7

66`

2’s Binary Complement

- Sign-and-magnitude and 1st complement systems have -0 (redundant)
- The 2's complement system does not have -0.
- The 2’s complement system yields the most efficient logic circuit implementation (most often used in computers).
- The positive numbers are the same for all systems

bit pattern	S-&-M	1's compl.	2's compl.
0000	0	0	0
0001	1	1	1
0010	2	2	2
0011	3	3	3
0100	4	4	4
0101	5	5	5
0110	6	6	6
0111	7	7	7
1000	-0	-7	-8
1001	-1	-6	-7
1010	-2	-5	-6
1011	-3	-4	-5
1100	-4	-3	-4
1101	-5	-2	-3
1110	-6	-1	-2
1111	-7	-0	-1

2’s Binary Complement

Algorithm for addition, A + B:

1. Perform binary addition on the two numbers.
2. ***Ignore the carry out of the MSB (most significant bit).***

Algorithm for subtraction, A – B:

A – B = A + (–B)

1. Take 2s complement of B by inverting all the bits and adding 1.
2. Add the 2s complement of B to A.

2's Binary Complement

- Examples: 4-bit binary system

```
  +3        0011
+ +4      + 0100
----      -------
  +7        0111
----      -------
```

```
  -2        1110
+ -6      + 1010
----      -------
  -8       11000
----      -------
```

```
  +6        0110
+ -3      + 1101
----      -------
  +3       10011
----      -------
```

```
  +4        0100
+ -7      + 1001
----      -------
  -3        1101
----      -------
```


Carry and Overflow

- Signed binary numbers are of a fixed range.
- If the result of addition/subtraction goes beyond this range, overflow occurs.
- In practice, we check the consistencies between sign bit to determine for overflow
  - *positive + positive* gives negative → overflow
  - *negative + negative* gives positive → overflow
  - *positive* + *negative* will never cause overflow

Carry and Overflow

Examples: 4-bit numbers (in 2s complement)

- Range of number is $(1000)_{2s}$ to $(0111)_{2s}$ or (-8_{10} to 7_{10})
- $(\underline{0}101)_{2s} + (\underline{0}110)_{2s} = (\underline{1}011)_{2s}$
  $(5)_{10} + (6)_{10} = -(5)_{10}$?! (overflow!)
- $(\underline{1}001)_{2s} + (\underline{1}101)_{2s} = (10110)_{2s}$ discard end-carry
  $= (\underline{0}110)_{2s}$ (overflow!)

71

Carry and Overflow

Check if overflow happens for the following arithmetic computation

- **Example 1:**
  4 bit, 4 + 2
  –Correct result
  –No overflow, no carry

0100	=	(+ 4)
0010	=	+ (+ 2)
0110	=	(+ 6)

- **Example 2:**
  4 bit, 4 + 6
  –Incorrect result
  –Overflow, no carry

0100	=	(+ 4)
0110	=	+ (+ 6)
1010	=	(– 6)

72`

Carry and Overflow

- **Example 3:**

  4 bit, -4 + -2

  –Result correct ignoring the carry

  –Carry but no overflow

1100	=	(– 4)
1110	=	+ (– 2)
11010	=	(– 6)

- **Example 4:**

  4 bit, -4 + -6

  –Incorrect result

  –Overflow, carry ignored

1100	=	(– 4)
1010	=	+ (– 6)
10110	=	(+ 3)

73`

Carry and Overflow

- More examples: 4-bit binary system

```
   -3         1101
 + -6       + 1010
 ----       -------
   -9        10111
 ----       -------
```

```
   +5         0101
 + +6       + 0110
 ----       -------
  +11         1011
 ----       -------
```

Which of the above is/are overflow(s)?

Both (-9 and +11 is outside the range of a 4-bit binary 2nd complement system)

74

Review Question

- Perform the following arithmetic operations with binary numbers (7 bits) and 2's complement representation. Check if overflow happens

  (+35) + (+40)

  (-35) + (-40)

- Perform the following arithmetic operations in 4 bits, 2's complement. Check if thre is any overflows.

  -3 - 6

  5 + 6

75`