CMSC 130

Lecture 4 – Arithmetic Operations, Addition, Subtraction and Overflow

Arithmetic

- Base *r* Addition
- BCD Addition
- Base *r* Subtraction
- BCD Subtraction
- Overflow

Arithmetic

- To clearly demonstrate arithmetic on number systems, keep in mind the following,
  - Rules of the arithmetic (addition, subtraction, multiplication, division) on decimal system apply to other number systems
  - other number systems use symbols **0 to (base-1)** (e.g. Decimal, 0 to (10-1) or 9)
  - **Operate on value**, then convert the value to the corresponding number using the symbols of the number system (e.g. $(7)_8 + (3)_8 =$ 10 in value = $(12)_8$)

ADDITION

Binary Addition

- Rules on Decimal system addition applies, only that binary has symbols 0 and 1 only

$$\begin{array}{r} 0 \\ +\,0 \\ \hline 0 \end{array} \qquad \begin{array}{r} 1 \\ +\,0 \\ \hline 1 \end{array} \qquad \begin{array}{r} 0 \\ +\,1 \\ \hline 1 \end{array} \qquad \begin{array}{r} 1 \\ +\,1 \\ \hline 10 \end{array}$$

Binary Addition

- Example,

		1	1	1		1	←	carry
	0	0	1	1	1	0	1	0
+	0	0	0	1	1	0	1	1
	0	1	0	1	0	1	0	1

BCD Addition

- Binary Addition works well in binary numbers with direct value equivalent
- For binary numbers that represent values in terms of code instead of direct conversion equivalent, this is where BCD Addition comes in
- Remember the difference between **coding** and **conversion** of a value into binary

BCD Addition

- Same rules apply,

$$\begin{array}{r} 0 \\ +\,0 \\ \hline 0 \end{array} \qquad \begin{array}{r} 1 \\ +\,0 \\ \hline 1 \end{array} \qquad \begin{array}{r} 0 \\ +\,1 \\ \hline 1 \end{array} \qquad \begin{array}{r} 1 \\ +\,1 \\ \hline 10 \end{array}$$

- In addition to the rules: if the resulting binary number does not correctly represent the resulting value***, an adjustment is to be done (just add $(6)_{10}$ or $(\mathbf{0110})_{\mathbf{2}}$ to the error)
- ****This can be caused by an end carry.*

BCD Addition

- Examples,

	17		0	0	0	1		0	1	1	1
+	18		0	0	0	1		1	0	0	0
	35		0	0	1	0		1	1	1	1
Adjustment of 6 (0110)								0	1	1	0
Final Answer			**0**	**0**	**1**	**1**		**0**	**1**	**0**	**1**

	17		0	0	0	1		0	1	1	1
+	11		0	0	0	1		0	0	0	1
	28		**0**	**0**	**1**	**0**		**1**	**0**	**0**	**0**
Final Answer. No need for adjustment.											

Addition on Other Bases

- Octal – base 8, symbols 0 to 7
  - Very similar to Decimal system. Use only the symbols 0 to 7
  - Example,

Decimal:		Base 8:	1	1	
	54		0	6	6
+	26		0	3	2
	80		1	2	0

Addition on Other Bases

- Hexadecimal – base 16, symbols 0 to 9 and A to F
  - **Add using values**, then **convert to symbols for the base**
  - Example,

			1	
	90		**5**	**A**
+	**150**		**9**	**6**
	240		**F**	**0**

carry

Addition on Other Bases

- For other bases, similar rules apply
- Just use the values and convert to symbols used for the base of the number system
- Keep in mind that a number system in base ***r*** only has the symbols **0 to *r*-1** for use in its numbers
  - Base 3: 0 to 2
  - Base 5: 0 to 4
  - Base 8: 0 to 7
  - Base 16: 0 to F (A to F in hex represents 10 to 15 in decimal)

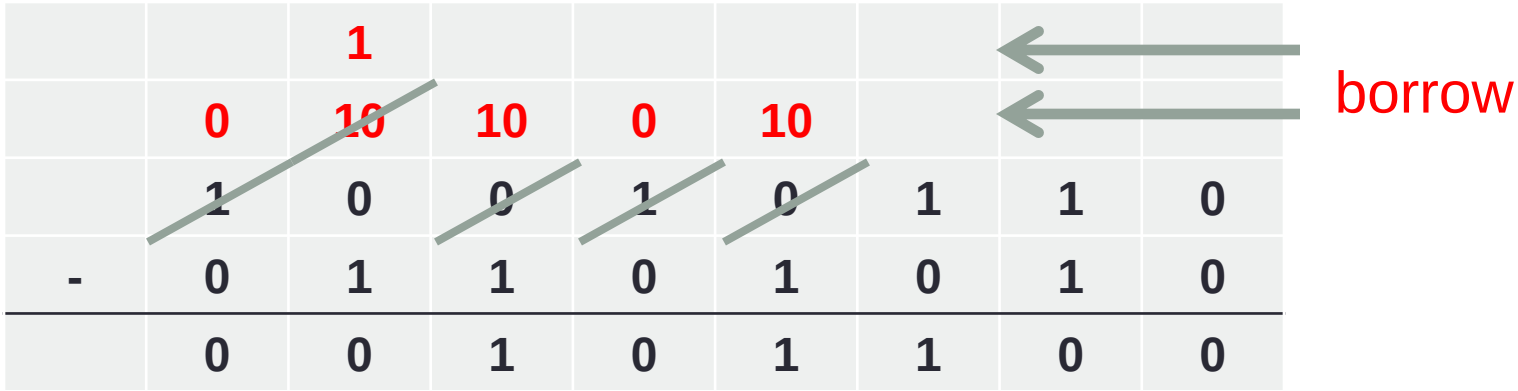
SUBTRACTION

Binary Subtraction

- Rules on Decimal system subtraction apply only that binary uses the symbols 0 and 1
  - **0 – 0 = 0**
  - **1 – 1 = 0**
  - **1 – 0 = 1**
  - **0 – 1 = 1 (borrow 1 from next more significant bit for 0)**
- On cases where the resulting value will be negative, use rules on subtraction with complement

Binary Subtraction

- Example demonstrating the basic rules,



BCD Subtraction

- Binary Subtraction works well in binary numbers with direct value equivalent
- For binary numbers that represent values in terms of code instead of direct conversion equivalent, this is where BCD Subtraction comes in
- Remember the difference between **coding** and **conversion** of a value into binary

BCD Subtraction

- Same rules apply
- In addition to the rules: if the resulting binary number does not correctly represent the resulting value, an adjustment is to be done (just subtract $(6)_{10}$ or $(\mathbf{0110})_{\mathbf{2}}$ from the result)

OVERFLOW

Overflow

- Occurs when two numbers of ***n*** digits each are added and the result occupies ***n*+1** digits
- This is in cases where fixed number of digits are allowed for the representation of the numbers (operands and result)
- This is an effect of having a limited storage width

Overflow

- Examples,

	Value				
	7	0	1	1	1
+	6	0	1	1	0
	13	1	1	0	1

	Value	Using 2's complement			
	-7	1	0	0	1
+	-6	1	0	1	0
	-13	0	0	1	1

Quiz (1/4) Show the necessary solutions.

- Convert the following decimal numbers to binary numbers before computing their sum/difference. Use 2's complement for negative numbers.
  1. 19+6
  2. 5+(-14)
  3. (-5)+14
  4. Bonus: (-12)+(-9)

- Perform BCD addition or subtraction to the ff.:
  1. 36+39, Bonus: 36-39
  2. 31+36, Bonus: 31-36

ADDITIONAL RULES

Addition of two negative numbers

- Express the negative numbers in two's complement
- Include their sign bits (e.g. -3 = 1111 1101)
- If there's an end carry, ignore it. Then, express your final answer in two's complement and place a '-' sign.

More Examples

- BCD Subtraction:
- 1. 39 – 36
- 2. 36 – 31

First Long Exam

- December 11, 2013
- Wednesday
- 1:00-3:00PM
- Rm. 229 or 224
- Bring YELLOW PAPERS