Logic Design

lecture 04

- Bit _ nibble _ Byte
  word
- Overflow
- Carry

Ex

$$(53)_{10} \to (110101)_2 \to (35)_{16}$$

Ex

$$(110100)_2 \to (34)_{16} \to (52)_{10}$$

Ex

$$(111000)_2 \xrightarrow{\text{1's complement}} 000111$$

$$(111000)_2 \xrightarrow{\text{2's complement}} 001000$$

Ex

$$\begin{array}{r} 1100 \\ -\ 0111 \\ \hline \end{array} \Rightarrow \begin{array}{rr} 1100 & \\ 1000 & + \\ \hline \boxed{1}\ 0100 & \\ + & \\ \hline 0101 & \end{array}$$

$\nearrow +5$

Ex

$$\begin{array}{r} 0110 \\ 1001\ - \\ \hline \end{array} \Rightarrow \begin{array}{rr} 0110 & \\ 0110 & + \\ \hline \boxed{x}\ 1100 & \\ - \quad 0011 & \end{array}$$

$-3 \swarrow$

Ex Represent $+3$ in two' complement signed number in 3_bit, also Do the same for $+1$, then add the two number

-4	2	1
0	1	1

+

-4	2	1
0	0	1

1 0 0 ← this is Number -4

How it comes

in 3-bit ⇒ maximum no that could be represented is

$$2^{n-1}-1 \Rightarrow 3$$

After addition you will have 4

So How it could be represented

Overflow

Represent in 4-bit

-8	4	2	1
0	0	1	1
0	0	0	1
0	1	0	0

+

$0 \; \underbrace{1 \; 0 \; 0}_{4}$

Positive

How processor knows that overflow occur

- Signed number
- Same sign
- and carry in the last two bit are the same

Ex

$$\begin{array}{r} \boxed{0\,1}\,1 \\ 011 \\ 001 \\ \hline 100 \end{array}$$

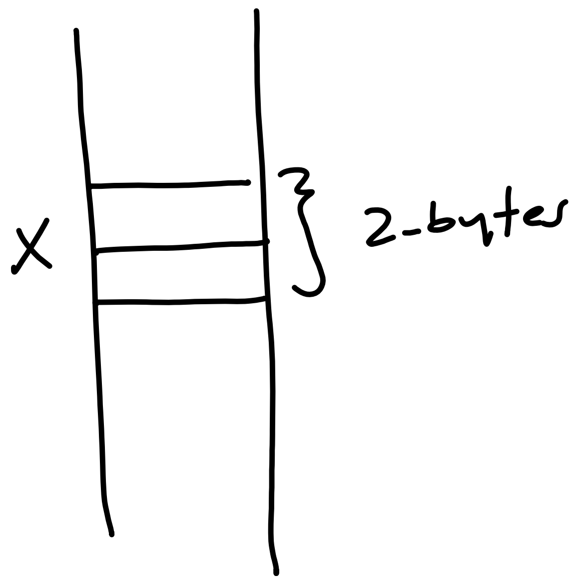
overflow

if you have any idea about programming

int x

⇓

x takes integer values



that cover all 16-bit range

if unsigned ⇒ 0 ⇒ ≃ 64000

if signed ⇒ -32,000 ⇒ 32,000

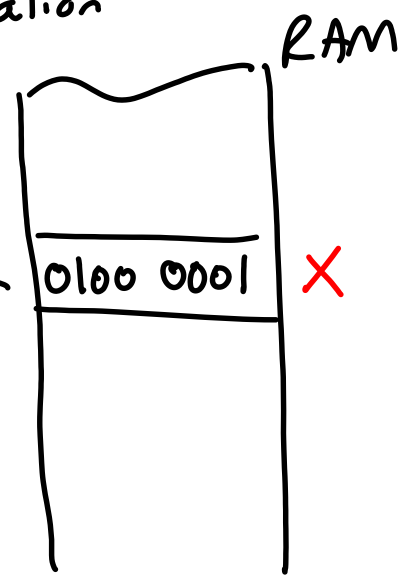
Ex

char x

↳ define x as character variable

x : is 1-byte location

x = 'A'



No. of combination that can represented by 1-byte is $2^8 \Rightarrow \underline{\underline{256}}$

enough for

- character
  - small
  - upper
- Number
- Special characters
- foreign language

one form of this is ASCII

Note

$2^{10} = 1024 \simeq 1\ K$

$2^{20} \simeq 1\ M$

$2^{30} \simeq 1\ G$

$1K = 2^{10}$ $1M = 2^{20}$ $1G = 2^{30}$

storing $\Rightarrow$ Register

No. of flip flops

8-bit Register = 8 flip flop

16-bit Register

32-bit Register

Binary codes

- Electronic digital systems use signals that have two distinct values $\Rightarrow$ two states

  - 0
  - 1

- Digital system represent and manipulate not only binary numbers but also many other <u>discrete</u> elements of information

  Binary code

  - Text characters
  - floating point
  - images
  - programs

- To represent 2^n distinct elements in a binary code requires a minimum of n-bits

Ex coding of compass direction
E, W, N, S

4 combinations $\Rightarrow$ need minimum two bits

00 $\Rightarrow$ E Code word

01 $\Rightarrow$ W

10 $\Rightarrow$ N

11 $\Rightarrow$ S

Note

- Each element must be assigned at least one unique binary bit combination
- Not all binary bit combinations need to have an element assigned

Decimal Code

Code for decimal numbers

$0 \rightarrow 9$: 10 combination

need minimum 4-bits

16 combinations

$6 \Rightarrow$ unused

Binary coded Decimal

BCD

- straight assignement of binary equivalent
- weighted 8 4 2 1

8	4	2	1	
0	0	0	0	⟹ 0
0	0	0	1	⟹ 1
0	0	1	0	⟹ 2
0	0	1	1	⟹ 3
0	1	0	0	⟹ 4
0	1	0	1	⟹ 5
0	1	1	0	⟹ 6
0	1	1	1	⟹ 7
1	0	0	0	⟹ 8
1	0	0	1	⟹ 9
1	0	1	0	
1	0	1	1	
1	1	0	0	⟹ Not used
1	1	0	1	
1	1	1	0	
1	1	1	1	

Ex

$(8)_{10} \xrightarrow{\text{binary}} 1000$

$(8)_{10} \xrightarrow{\text{BCD}} 1000$

$(23)_{10} \xrightarrow{\text{binary}} 10111$

$(23)_{10} \xrightarrow{\text{BCD}} 0010\ 0011$

BCD Addition

- Addition is done BCD digit by BCD digit
  - 4 bits at the time
- Can we use normal binary addition?
  - digits adding up to more than 9
    - Binary addition will result in invalid BCD codes

1010 . . .   1111

Solution

check if resulting value is greater than 9

>> if so, add 6

Ex

$$\begin{array}{r} 532 \\ 126 \; + \\ \hline 658 \end{array} \qquad \begin{array}{rrr} 0101 & 0011 & 0010 \\ 0001 & 0010 & 0110 \; + \\ \hline \underline{\underline{0110}} & \underline{\underline{0101}} & \underline{1000} \\ \Downarrow & \Downarrow & \Downarrow \\ 6 & 5 & 8 \end{array}$$

Ex

$$\begin{array}{r} 532 \\ + \; 268 \\ \hline 800 \end{array} \qquad \begin{array}{rrr} 0101 & 0011 & 0010 \\ 0010 & 0110 & 1000 \\ \hline 0111 & 1001 & \underline{1010} \\ 1000 & 1010 & 0110 \\ & 0110 & \overline{0000} \\ & \overline{0000} & \end{array}$$

Correction

Signed 10's complement

MSD ⇒ 0 indicates positive
9 indicates negative

0345 ⇒ 9655
↳ -345

Note Some computers have special hardware to perform arithmetic in BCD directly

Gray code
= Reflected code = cyclic code

a change in only one bit position as we move from one code word to the subsequent code word

0000 ⇒ 0
0001 ⇒ 1
0011 ⇒ 2
0010 ⇒ 3
0110 ⇒ 4
0111 ⇒ 5
0101 ⇒ 6
0100 ⇒ 7

Not weighted code

1 1 0 0 $\Rightarrow$ 8
1 1 0 1 $\Rightarrow$ 9
1 1 1 1
1 1 1 0
1 0 1 0
1 0 1 1
1 0 0 1
1 0 0 0

not used

$(49)_{10} \xrightarrow[\text{Code}]{\text{Gray}} (0110 \quad 1101)$

$(55)_{10} \xrightarrow[\text{Code}]{\text{Gray}} (0111 \quad 0111)$

<u>Note</u>

$4 \xRightarrow{\text{BCD}}$ 0100

Gray code 0110

$6 \xRightarrow{\text{BCD}}$ 0110

0 101

EXCESS-3 $\Rightarrow$ ADD 3 To BCD