TECH 3233

Number Systems in Microprocessors

1

Decimal – Base 10

- Decimal has digits 0 - 9.
- Number system we commonly use in our day to day lives.

Hundreds	Tens	Ones
10^2	10^1	10^0

2

Binary – Base 2

- Binary has digits 0 and 1.
- Commonly used in digital logic, computers and networking.

128	64	32	16	8	4	2	1
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

3

Example Decimal to Binary Conversion

- Convert 100 to binary using weighting factors.

128	64	32	16	8	4	2	1
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

4

Example Decimal to Binary Conversion

- Convert 200 to binary using weighting factors.

128	64	32	16	8	4	2	1
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

5

Convert 100 to binary
(division method)

Convert 100 to binary using division method.

Division	Quotient	Remainder

6

Example Binary to Decimal Conversion

Convert 1010110_2 to decimal

128	64	32	16	8	4	2	1
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

7

Example Binary to Decimal Conversion

Convert 11010010_2 to decimal

128	64	32	16	8	4	2	1
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

8

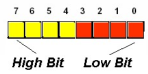
Labeling a Binary Number

- We need a way to tell a computer the difference between 1010_{10} and 1010_2
- In textbooks we use subscripts (like above)
- In this class a leading 0b sign will show a binary value.
- Some other methods include a leading % or a B at the end.

9

Binary Numbers (Definitions)

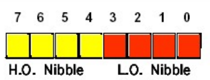
- The bit to the left is the High Bit
- The bit to the right is the Low Bit
- Note the lowest bit is called “Bit 0”

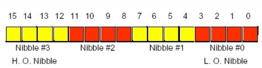


10

Binary Numbers (Definitions)

- Similarly the nibble to the left is the High Nibble
- The nibble to the right is the Low Nibble





11

Hex – Base 16

Base 10	Base2	Base 16		Base 10	Base2	Base 16
0	0000	0		8	1000	8
1	0001	1		9	1001	9
2	0010	2		10	1010	A
3	0011	3		11	1011	B
4	0100	4		12	1100	C
5	0101	5		13	1101	D
6	0110	6		14	1110	E
7	0111	7		15	1111	F

12

Hex – Base 16

- Most commonly used in computers and networking (error messages in windows and MAC addresses)
- Why base 16? Because 4 bits can be converted to decimal digits 0 -> 15.

8	4	2	1
2^3	2^2	2^1	2^0

13

Example Decimal to Hex Conversion

Convert 100 to Hex via Binary.

128	64	32	16	8	4	2	1
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
8	4	2	1	8	4	2	1

14

Example Decimal to Hex Conversion

Convert 200 to Hex via Binary.

128	64	32	16	8	4	2	1
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
8	4	2	1	8	4	2	1

15

Example Hex To Decimal Conversion

Convert A5 Hex To Decimal.

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

16

Example Hex To Decimal Conversion

Convert 7D Hex To Decimal.

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

17

Why use Hexadecimal?

- Hex is easier to read than binary and it is less likely to introduce errors (try copying down a list of 10 8 bit binary digits and do the same with the same 10 values represented in HEX)
- Each hex digits requires one nibble (four bits) to store in the computer's binary memory (easy to translate bin to/from hex)

18

Labeling a Binary Number

- We need a way to tell a computer the difference between 10_{10} and 10_{16}
- In textbooks we use subscripts (like above)
- In this class, Hex will be represented by a leading “0x” (0x10). Windows also uses this method.
- In some programs a leading $ sign will show a Hex value (0x10)

19

Other ways to store data in bits

20

Binary Coded Decimal

- Each Decimal Digit is represented by its 4 bit binary equivalent.
- EG

145

1 4 5

0001 0100 0101

- NOTE: This is not BINARY (145 dec to binary would be 10010001)

21

Time

- We can use a techniques like bit fields and packed data to store information like the date:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M	M	M	M	D	D	D	D	D	Y	Y	Y	Y	Y	Y	Y

•We know the month can be represented by values 1-12 we use 4 bits (which can represent values 0 – 15).
•The largest month has 31 days so we use 5 bits (which can represent values 0 – 31)
•Here we represent the year using the last two digits of the year so 7 bits are used (which can represent values 0 – 127)

22

Negative Numbers

Two's Complement

- used to represent both positive and negative numbers
- Give up MSB as a sign bit (1 -> negative)
- positive numbers are the same as they would be without the two's complement representation.

Examples: $01101_2=13_{10}$
$11101_2=-3_{10}$

From: Chris Gregg - University of Va.

23

Converting to Two's Comp

Method 1

(example $00110_2=6_{10}$→ Convert to -6 in 2's complement)

- Flip all the bits
  - 11001 (this is the 1's complement)
- Add 1
  - 11010

24

Converting to Two's Comp

Method 2 – Shortcut

(example $00110_2 = 6_{10}$ → Convert to -6 in 2's complement)

- Start at least significant bit (the farthest to the right), and copy 0s until you get to a 1 (also copy the 1): 10
- Then flip the rest of the bits:11010

25

Why 2's Comp?

- Only one form of 0.
- *Easy* to subtract:
  - Just Add!

```
 0100 ( 4₁₀)
+1101 (-3₁₀)
-----
 0001 ( 1₁₀)
```

It works!

Two's Complement	Decimal
0111	7
0110	6
0101	5
0100	4
0011	3
0010	2
0001	1
0000	0
1111	-1
1110	-2
1101	-3
1100	-4
1011	-5
1010	-6
1001	-7
1000	-8

26

Why 2's Comp?

Just Add!

```
 1011 (-5₁₀)
+0010 ( 2₁₀)
-----
 1101 (-3₁₀)
```

It works!

Two's Complement	Decimal
0111	7
0110	6
0101	5
0100	4
0011	3
0010	2
0001	1
0000	0
1111	-1
1110	-2
1101	-3
1100	-4
1011	-5
1010	-6
1001	-7
1000	-8

27

Floating Point Numbers

- Computers use IEEE Standard 754 for storing Floating Point Numbers in Binary

  https://www.h-schmidt.net/FloatConverter/IEEE754.html

- Values are stored in 3 bit fields
  - Sign Bit
  - Exponent Field
  - Mantissa Field

From: http://steve.hollasch.net/cgindex/coding/ieeefloat.html

28

Floating Point Numbers

- Sign Bit
  - 0 denotes a positive number
  - 1 denotes a negative number

29

Floating Point Numbers

- Exponent
  - Can represent both positive and negative exponents.
  - A *bias* is added to the actual exponent in order to get the stored exponent.
  - For IEEE single-precision floats, this value is 127. Thus,
    - an exponent of zero means that 127 is stored in the exponent field.
    - A stored value of 200 indicates an exponent of (200-127), or 73.
  - Note: exponents of -127 (all 0s) and +128 (all 1s) are reserved for special numbers.

30

Floating Point Numbers

- The Mantissa (aka significand)
  - represents the precision bits of the number. It is composed of an implicit leading bit and the fraction bits.

31

Floating Point Numbers

- Single Precision (32 Bit - 127 Bias)

sign bit	biased exponent	fraction from normalized mantissa
1 bit	8 bits	23 bits
s	*e*	*f*

- Double Precision (64 bit – 1023 Bias)

sign bit	biased exponent	fraction from normalized mantissa
1 bit	11 bits	52 bits
s	*e*	*f*

- Diagrams from: http://www.math.byu.edu/~schow/work/IEEEFloatingPoint.htm

32

ASCII

- American Standard Code for Information Interchange (ASCII)
- Used to represent letters and symbols
- Standard was adopted in 1963

33

ASCII

- Broken into 4 groups of 32 characters
  - Group 1 (0x00 to 0x1F) – Non Printable control characters
  - Group 2 (0x20 to 0x3F) – punctuation, special characters and numeric digits
  - Group 3 (0x40 to 0x5F) – Letters ‘A..Z’ and some other characters
  - Group 4 (0x60 to 0x7F) – Letters ‘a..z’ and some other characters.
- http://asciitable.com

34

Op Codes

- Computers store instructions, called Operation Codes (or Op Codes) in memory as binary values.
- Each Microprocessor uses a different set of op codes.

- Example
  - 0x1B tells a 68HC11 to Add two values
  - The same value tells an 80x86 processor to subtract two numbers

35

Context!

- In a computer a value stored in memory cannot be interpreted unless it is put in context by a program!

36