ECE 372 – Microcontroller Design

Data Formats

THE UNIVERSITY OF ARIZONA. TUCSON ARIZONA

ECE 372 – Microcontroller Design

Humor

- There are 10 types of people in the world: Those who get binary and those who don't.

ECE 372 – Microcontroller Design
Information vs. Data

- Information
  - An abstract description of facts, processes or perceptions
  - How can we represent information?
  - How can we represent changing information?
  - We need to associate different values with different events
- Data
  - Individual fact value or set of facts or values
  - Measurement or storage

September 19th
October 28, 2010
$250
1.8 Trillion

ECE 372 – Microcontroller Design
Information vs. Data

- Information vs. Data
  - Information is data with context or meaning

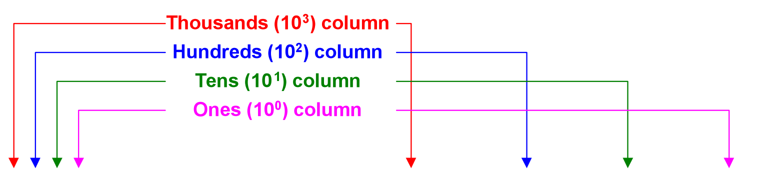
International Talk Like a Pirate Day:	September 19th
Date of your first ECE 372 midterm:	October 28, 2010
How much Bill Gates earns per second:	$250
Number of pennies to fill Empire State Building	1.8 Trillion

ECE 372 – Microcontroller Design
Data Representation

- Data Representation
  - The same data can be represented with:
    - Different symbols
      - English, Cyrillic, Arabic
    - Different numeric bases
      - Binary, Octal, Hexadecimal, Decimal
    - Different formats
      - Little Endian, Big Endian, Binary Coded Decimal

ECE 372 – Microcontroller Design
Data Structure (Encodings)

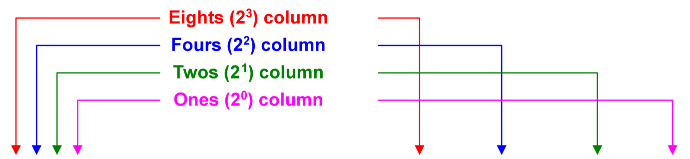
- Decimal Numbers
  - Uses the ***ten*** numbers from 0 to 9
  - Each column represents a power of ***10***



$1999_{10} = 1x10^3 + 9x10^2 + 9x10^1 + 9x10^0$

ECE 372 – Microcontroller Design
Data Structure (Encodings)

- Binary Numbers
  - Uses the ***two*** numbers from 0 to 1
  - Every column represents a power of ***2***



$1001_2 = 1x2^3 + 0x2^2 + 0x2^1 + 1x2^0$

0	0	0	0	1	0	0	1
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

ECE 372 – Microcontroller Design
Data Structure (Encodings)

- Binary Numbers

0	0	1	0	1	0	0	1
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

$= 0*2^7 + 0*2^6 + 1*2^5 + 0*2^4 + 1*2^3 + 0*2^2 + 0*2^1 + 1*2^0$

0	0	1	0	1	0	0	1
128	64	32	16	8	4	2	1

$= 0*128 + 0*64 + 1*32 + 0*16 + 1*8 + 0*4 + 0*2 + 1*1$

$= 41$

ECE 372 – Microcontroller Design
Data Structure (Encodings)

- Hexadecimal Numbers
  - Uses the ***ten*** numbers from 0 to 9 and ***six*** letter A to F
  - Each column represents a power of ***16***

0	0	0	0	A	0	C	5
16^7	16^6	16^5	16^4	16^3	16^2	16^1	16^0

Binary	Decimal	Hexadecimal
0	0	0
1	1	1
10	2	2
11	3	3
100	4	4
101	5	5
110	6	6
111	7	7
1000	8	8
1001	9	9
1010	10	A
1011	11	B
1100	12	C
1101	13	D
1110	14	E
1111	15	F
10000	16	?
10001	17	?

ECE 372 – Microcontroller Design
Data Structure (Encodings)

0	0	0	0	A	0	C	5
16^7	16^6	16^5	16^4	16^3	16^2	16^1	16^0

$= A*16^3 + 0*16^2 + C*16^1 + 5*16^0$

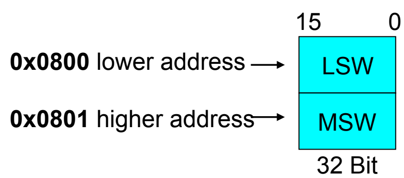
$= 10*4096 + 0*256 + 12*16 + 5*1$

$= 41{,}157_{10}$

ECE 372 – Microcontroller Design

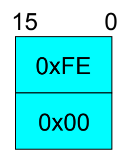
Big vs. Little Endian

- Big Endian
  - MSB (Most Significant Byte/Word) is at lowest address
  - MIPS, Sparc, HC12
- Little Endian
  - LSB (Least Significant Byte/Word) is at lowest address
  - PIC24F, x86




```
unsigned long int *a;

a = 0x0800;
*a = 254;
```



ECE 372 – Microcontroller Design

Positive vs. Negative Numbers

- Negative numbers are common
  - How can we represent negative numbers in binary?
- Signed-magnitude
  - Use leftmost bit for sign bit
    - So -5 would be: 1101
- One's Complement
  - Invert all bits for negative numbers
    - So -5 would be: 1010

ECE 372 – Microcontroller Design

Positive vs. Negative Numbers

- Two's Complement
  - Allows us to perform subtraction using addition
    - No need for dedicated subtractor within CPU's ALU
  - Two's complement of a number added to the number itself will equal zero
    - So -5 would be: 1011
    - $1011_2 + 0101_2 = 0000_2$ *(with carry of 1, ignored)*
  - Invert all bits and add 1 to get complement of a number
    - Fast conversion: find first 1 from right, invert after 1

```
  0011 1000 (56)
+ 1111 0110 (-10)
-----------------
  0010 1110 (46)
```

ECE 372 – Microcontroller Design

Positive vs. Negative Numbers

Decimal	Signed-Magnitude	One's Complement	Two's Complement
+4	-	-	-
+3	0 11	0 11	0 11
+2	0 10	0 10	0 10
+1	0 01	0 01	0 01
0	0 00	0 00	0 00
-1	1 01	1 10	1 11
-2	1 10	1 01	1 10
-3	1 11	1 00	1 01
-4	-	-	1 00

Are we missing a value?

ECE 372 – Microcontroller Design

Positive vs. Negative Numbers

- Data Ranges
  - Unsigned
    - 0 to 2^n-1
  - Signed-Magnitude
    - $-2^{n-1}-1$ to $2^{n-1}-1$
  - One's Complement
    - $-2^{n-1}-1$ to $2^{n-1}-1$
  - Two's Complement
    - -2^{n-1} to $2^{n-1}-1$

ECE 372 – Microcontroller Design

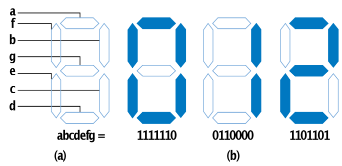
Positive vs. Negative Numbers

- Determine the two's complement representation for the following decimal numbers *(assume we are using 8-bit binary numbers)*:
  - -1
  - -11
  - -15
  - 22
  - -101

ECE 372 – Microcontroller Design
Binary Coded Decimal

- Binary Coded Decimal
  - Each digit of a decimal number is represented as a 4-bit binary number
  - Often used for 7-segment displays

Binary	BCD
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
Undefined	10
Undefined	11
Undefined	12
Undefined	13
Undefined	14
Undefined	15



123_{10} → 0001, 0010, 0011 $\Rightarrow 123_{10} = 0001\ 0010\ 0011_{(BCD)}$

ECE 372 – Microcontroller Design
ASCII Characters

	b_4	b_3	b_2	b_1	0	1	2	3	4	5	6	7
					0	0	0	0	1	1	1	1 ← b_7
					0	0	1	1	0	0	1	1 ← b_6
					0	1	0	1	0	1	0	1 ← b_5
0	0	0	0	0	NUL	DLE	SP	0	@	P	\	p
1	0	0	0	1	SOH	DC1	!	1	A	Q	a	q
2	0	0	1	0	STX	DC2	"	2	B	R	b	r
3	0	0	1	1	ETX	DC3	#	3	C	S	c	s
4	0	1	0	0	EOT	DC4	$	4	D	T	d	t
5	0	1	0	1	ENQ	NAK	%	5	E	U	e	u
6	0	1	1	0	ACK	SYN	&	6	F	V	f	v
7	0	1	1	1	BEL	ETB	'	7	G	W	g	w
8	1	0	0	0	BS	CAN	(	8	H	X	h	x
9	1	0	0	1	HT	EM	)	9	I	Y	i	y
A	1	0	1	0	LF	SUB	*	:	J	Z	j	z
B	1	0	1	1	VT	ESC	+	;	K	[	k	{
C	1	1	0	0	FF	FS	,	⟨	L	\	l	|
D	1	1	0	1	CR	GS	–	=	M	]	m	}
E	1	1	1	0	SO	RS	.	⟩	N	∧	n	~
F	1	1	1	1	SI	US	/	?	O	___	o	DEL

Control characters   Printable characters (96)

ECE 372 – Microcontroller Design

Real Numbers

- Real Numbers
  - How can we represent a real number (i.e. numbers that contain a fractional part)?
    - Fixed Point Numbers
    - Floating Point Numbers
  - *Note*: our C compiler already has built-in routines to deal with real numbers
    - However the computational needs will be ***SIGNIFICANT***

ECE 372 – Microcontroller Design

Fixed Point Numbers

- Fixed Point Numbers
  - Real number with fixed number of digits before and after radix point
    - N-bits used to represent integer part
    - M-bits used to represent fractional part
    - Unsigned range: 0 to $2^M + {}^1/_{2^N}$

0	1	0	0	.	0	1	0	0
2^3	2^2	2^1	2^0		2^{-1}	2^{-2}	2^{-3}	2^{-4}

ECE 372 – Microcontroller Design

Fixed Point Numbers

0	1	0	0	.	0	1	0	0	
2^3	2^2	2^1	2^0		2^{-1}	2^{-2}	2^{-3}	2^{-4}	$= 0*2^3 + 1*2^2 + 0*2^1 + 0*2^0 + 0*2^{-1} + 1*2^{-2} + 0*2^{-3} + 0*2^{-4}$

0	1	0	0	.	0	1	0	0	
8	4	2	1		$^1/_2$	$^1/_4$	$^1/_8$	$^1/_{16}$	= 0*8 + 1*4 + 0*2 + 0*1 + 0*.5 + 1*.25 + 0*.125 + 0*.0625

= 4.25

ECE 372 – Microcontroller Design

Fixed Point Numbers

1	0	1	1	.	1	0	0	1	
2^3	2^2	2^1	2^0		2^{-1}	2^{-2}	2^{-3}	2^{-4}	$= 1*2^3 + 0*2^2 + 1*2^1 + 1*2^0 + 1*2^{-1} + 0*2^{-2} + 0*2^{-3} + 1*2^{-4}$

1	0	1	1	.	1	0	0	1	
8	4	2	1		$^1/_2$	$^1/_4$	$^1/_8$	$^1/_{16}$	= 1*8 + 0*4 + 1*2 + 1*1 + 1*.5 + 0*.25 + 0*.125 + 1*.0625

= 11.5625

ECE 372 – Microcontroller Design
Fixed Point Numbers

$$\begin{array}{lr} & 0100.0100 \ (4.25) \\ + & 1011.1001 \ (11.5625) \\ \hline & 1111.1101 \end{array}$$

$$\frac{1}{2^3}\ \frac{1}{2^2}\ \frac{1}{2^1}\ \frac{1}{2^0}\ .\ \frac{1}{2^{-1}}\ \frac{1}{2^{-2}}\ \frac{0}{2^{-3}}\ \frac{1}{2^{-4}} = 1*2^3 + 1*2^2 + 1*2^1 + 1*2^0 + 1*2^{-1} + 1*2^{-2} + 0*2^{-3} + 1*2^{-4}$$

$$\frac{1}{8}\ \frac{1}{4}\ \frac{1}{2}\ \frac{1}{1}\ .\ \frac{1}{1/2}\ \frac{1}{1/4}\ \frac{0}{1/8}\ \frac{1}{1/16} = 1*8 + 1*4 + 1*2 + 1*1 + 1*.5 + 1*.25 + 0*.125 + 1*.0625$$

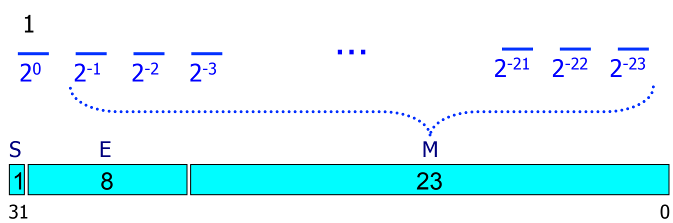
$$= 15.8125$$

ECE 372 – Microcontroller Design
Floating Point Numbers

- Floating Point Numbers
  - Real number representation similar to scientific notation
    - $x = M * B^E$
  - Base (B)
    - Base of the numbering systems considered
    - Binary (2) for computer based implementations
    - *We will assume base of 2 for remaining description*
  - Sign *(S)*
    - Indicating positive of negative number
    - 0 – Positive, 1 – Negative

ECE 372 – Microcontroller Design
Floating Point Numbers

- Floating Point Numbers
  - Real number representation similar to scientific notation
    - $x = M * 2^E$
  - Mantissa *(M)*
    - Digits corresponding to the magnitude
    - Stored in a normalized form, where the first bit is assumed to be 1
      - 1 is the only possible non-zero number in binary
    - Remaining bits correspond to fraction values (similar to fixed point)



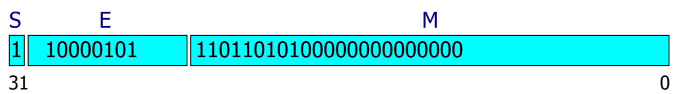
ECE 372 – Microcontroller Design
Floating Point Numbers

- Floating Point Numbers
  - Real number representation similar to scientific notation
    - $x = M * 2^E$
  - Exponent *(E)*
    - Needs to represent both positive and negative values
    - Stored exponent is adjust using the exponent bias
      - Exponent bias = $\boldsymbol{2^{N-1}-1}$, where N is the number of bits in the exponent
      - $E_{Actual} = E_{Stored} – E_{Bias}$
    - Example, 8-bit exponent:
      - $E_{Bias} = 2^{8-1}-1 = 127$
      - $E_{Stored} = 1000\ 0000_2 = 128$
      - $E_{Actual} = 128 - 127 = 1$

ECE 372 – Microcontroller Design

Floating Point Numbers

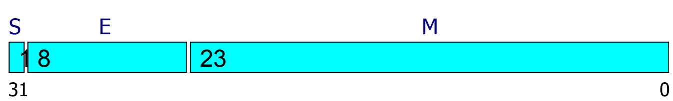
- Floating Point Numbers
  - Example: Convert the value −118.625 to floating point representation
  - 1. Determine sign bit:
    - -118.625 is negative, S = 1
  - 2. Convert to binary:
    - 118.625 = 1110110.101 (1110110 = 118, .101 = .625)
  - 3. Normalize number
    - $1110110.101 = 1.110110101 * 2^6$ (110110101 = Mantissa)
  - 4. Determine exponent
    - $E_{Stored} = E_{Actual} + E_{Bias}$
    - $E_{Stored} = 6 + 127 = 133_{10} = 10000101_2$



ECE 372 – Microcontroller Design

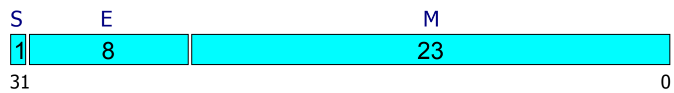
Floating Point Numbers

- Floating Point Numbers
  - Real number representation similar to scientific notation
    - $x = M * 2^E$
  - Zero
    - Due to assuming a leading 1 in the mantissa, we cannot directly represent the value 0 using floating point
    - Defined special case for value of 0
    - Define special case: Exponent and mantissa of all 0's corresponds to the value 0
  - Other special cases exist:
    - +/- Infinity
    - Denormalized value
    - Not a Number

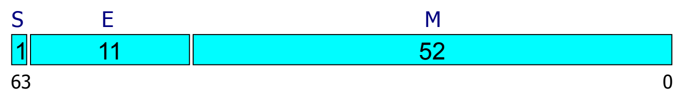


ECE 372 – Microcontroller Design
Floating Point Numbers

- Floating Point Numbers
  - IEEE Standard for Binary Floating-Point Arithmetic (IEEE 754)
  - Single Precision Floating Point



  - Double Precision Floating Point



ECE 372 – Microcontroller Design
Data vs. Information

- What does the bit pattern mean: 1011 1001
  - Unsigned: 185 decimal
  - Sign-Magnitude: -57 decimal
  - 1's complement: -70 decimal
  - 2's complement: -71 decimal
  - Fixed point 4bit.4bit: 11.5625 decimal
  - BCD:
    - 1011_2 =B (Undefined)
    - $1001_2 = 9_{10}$
  - ASCII: '9'
  - HC12 Opcode: $39 = ADCA *(Add with Carry to A, Ext. Addressing)*

ECE 372 – Microcontroller Design

Information Representation

- Text Page
  - 1 page letter
  - 8.5x11 inches, 60x68 12 point characters
  - ASCII 8bit: 60x68x8 = 32640 bits
  - *4 Kbytes*
- Fax Page Scan
  - Fax low resolution 204x98 dots/inch
  - 8.5x204x11x98 = 1869252 dots
  - *235 Kbytes (B/W)*
- DVD
  - 8Mbit/sec
  - 2 hours 2x60x60x8 = 57600 Mbit
  - *Approximately 7.2 Gbytes*

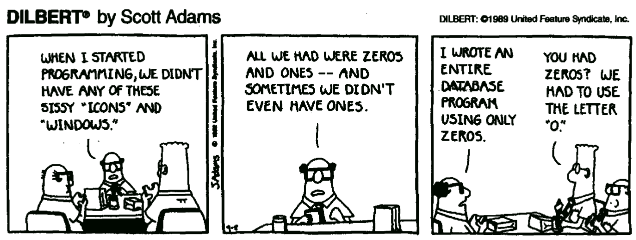
ECE 372 – Microcontroller Design

Information Representation

- Digital Audio
  - 44.2kHz sampling rate
  - 16bit/channel 1..65536=2^16
  - Stereo
  - How many seconds of sound fit in
    - 32kbyte EPROM mono?
    - 1.4 Mbyte floppy mono?
    - 660Mbyte CD Rom stereo?

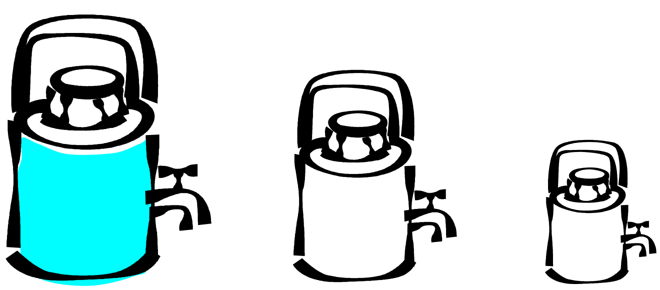
ECE 372 – Microcontroller Design

Humor



ECE 372 – Microcontroller Design

Time for Fun (or maybe not?)



10 Gallon 7 Gallon 3 Gallon