Digital Representation of Information
(unsigned whole numbers)

A major stumbling block many beginners face while learning assembly language → common use of the binary and hexadecimal number systems

- Learning to program in assembly language requires some familiarity especially in…
  … the ***binary*** (*base 2*) number system and…
  … the ***hexadecimal*** (*base 16*) number system, …
- And some proficiency in…
  … the *conversion* between these systems and…
  … the ***decimal*** (base ***10***) number system
- What's the more technical word for *base*? radix

Digital Representation of Information
(unsigned whole numbers)

- The binary number system as used by computers (*why?*) is a *positional* number system (*what's that?*)
  - Just like the *decimal* number system we are accustomed to
  - (similarly for the *hexadecimal* number system)
- Decimal (base 10) number example:
  - 95243 =
    $9\times10^4 + 5\times10^3 + 2\times10^2 + 4\times10^1 + 3\times10^0$
  - *Most significant digit* and *least significant digit*: which ones?
- Binary (base 2) number example:
  - 1011010b = b to indicate that it's in binary
    $1\times2^6 + 0\times2^5 + 1\times2^4 + 1\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0$
  - *Most significant bit* and *least significant bit*: which ones?

Digital Representation of Information
(unsigned whole numbers)

- ***Horner's method*** for efficient evaluation of *polynomials*
  - Uses least number of multiplications and additions
- Mathematical basis (d for **d**igit, r for **r**adix):
  - $d_n r^n + d_{n-1} r^{n-1} + d_{n-2} r^{n-2} + \ldots + d_2 r^2 + d_1 r^1 + d_0 =$
    $((\ldots((d_n r + d_{n-1})r + d_{n-2})r + \ldots + d_2)r + d_1)r + d_0$
- Example (n is 6, r is 2):
  - 1011010b =
    $1\times2^6 + 0\times2^5 + 1\times2^4 + 1\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0 =$
    $(((((1\times2 + 0)2 + 1)2 + 1)2 + 0)2 + 1)2 + 0 = 90$

Digital Representation of Information
(unsigned whole numbers)

- Binary to decimal conversion example:
  - Convert 1011010b to decimal (illustrate)
- Binary to decimal – hard way: apply definition as-is
  - Start at right end of binary number
  - Write 1 over the last digit
  - Work your way to the left and write numbers 2 times the one last written over each successive digit
  - Circle the numbers that are over 1's
  - Add the circled numbers

Digital Representation of Information
(unsigned whole numbers)

- Binary to decimal conversion example:
  - Convert 1011010b to decimal (illustrate)
- Binary to decimal – easy way: apply (definition + Horner's Method)
  - *sum* = current bit = leftmost bit
  - Scan number bit by bit from left to right:

double & add

```
if current bit is not rightmost bit (i.e., there's a next bit)
    sum = 2*sum + next bit
    continue scanning (i.e., make next bit the current bit)
else
    end scanning
```

  - *sum* now has answer

sometimes called the **double-&-add** method

Digital Representation of Information
(unsigned whole numbers)

- Decimal to binary conversion example:
  - Convert 90 to binary (illustrate)
- Decimal to binary conversion
  - One way: *repeated division* method
  - Another way: *repeated subtraction* method

Digital Representation of Information
(unsigned whole numbers)

- Basis for the repeated division method
  - Do you see why it works?
  - Case to observe: repeatedly divide a decimal number by 10
    - Each division moves a digit into the *remainder*
    - The first division moves the *least significant digit* (and the last division moves the *most significant digit*)
  - Same is observed: any *unsigned* number is divided by its *base*
  - What would be the corresponding result for *multiplication*?
- The above shows that the repeated division method can also be used to convert between *any two* bases
  - Division must be done using the *original* base!
  - Generally for *humans*: *decimal* to another base
  - Generally for *computers*: *binary* to another base

Digital Representation of Information
(unsigned whole numbers)

- Looking at repeated division method another way:

Decimal-to-Binary Conversion for 35 Using Repeated Division Method

Division by 2 = Outcome	Meaning of Outcome	Bit(s) Unraveled					
		2^5	2^4	2^3	2^2	2^1	2^0
35/2 = 17R1	$35 = 17(2^1) + 1(2^0)$						1
17/2 = 8R1	$17(2^1) = 8(2^2) + 1(2^1)$					1	
8/2 = 8R0	$8(2^2) = 4(2^3) + 0(2^2)$				0		
4/2 = 2R0	$4(2^3) = 2(2^4) + 0(2^3)$			0			
2/2 = 1R0	$2(2^4) = 1(2^5) + 0(2^4)$		0				
1/2 = 0R1	$1(2^5) = 0(2^6) + 1(2^5)$	1					

Digital Representation of Information
(unsigned whole numbers)

- Main problem with the binary number system for humans is that decimal numbers of reasonable size are long and cumbersome in binary
  - 1000 = 1111101000b
  - 1 million is 20 binary digits long
- Digits of the *hexadecimal* number system often considered as convenient shorthand for writing numbers in binary
  - Hex digits: **0**, **1**, **2**, **3**, **4**, **5**, **6**, **7**, **8**, **9**, **A**, **B**, **C**, **D**, **E**, **F**
  - Will see a lot of them in assembly language
  - Need to be comfortable reading hex numbers

Digital Representation of Information
(unsigned whole numbers)

- Convention commonly used to identify hex numbers:
  - **`0x`** always precedes a hex number
  - Used by C/C++ and MIPS assembler
- There are other conventions
  - For a good list, see http://en.wikipedia.org/wiki/Hexadecimal

Digital Representation of Information
(unsigned whole numbers)

- Correspondence chart between hex and binary digits
  - Will be used often to convert back and forth between hex and binary
- Good to be able to quickly generate it on the fly
  - How? (illustrate systematic way)
- Even better if memorize it
  - What *memory aids*? (illustrate some of the tricks)

Digital Representation of Information
(unsigned whole numbers)

- Hex to binary conversion:
  - Simply replace *each hex digit* with its *4-bit group* equivalent
- Hex to binary conversion example:
  - Convert EACh to binary (illustrate)

Digital Representation of Information
(unsigned whole numbers)

- Binary to hex conversion:
  - Simply revert the process of converting from hex to binary
  - First divide the binary digits into 4-bit groups, then replace each group with its hex digit equivalent
  - *Caveat*: must start dividing beginning from the ***right*** end
- Binary to hex conversion example:
  - Convert 1100111001011b to hex (illustrate)

Digital Representation of Information
(unsigned whole numbers)

- Decimal to hex conversion:
  - Can first convert to binary and then to hex
  - (conversions involved → already discussed/illustrated)
  - But we can use either the *repeated subtraction* or *repeated division* method directly
  - (using *powers of 16* and *16 as divisor*, respectively)
- Decimal to hex conversion example:
  - Convert 6841 to hex (illustrate)

Digital Representation of Information
(unsigned whole numbers)

- Hex to decimal conversion:
  - Either of the methods suggested for binary to decimal conversion can be used
  - (with **2** replaced by **16** in *appropriate* places)
- Hex to decimal conversion example:
  - Convert 1AB9h to decimal

Digital Representation of Information
(unsigned whole numbers)

- Adding numbers in binary and hex
  - Straightforward adaptation of the usual decimal addition method
  - Are you aware of (and can you list) the rules followed?
- Rules for binary addition:

```
Carry-in bit:     0   0   0   0  |  1   1   1   1
Bit of 1st #:     0   1   0   1  |  0   1   0   1
Bit of 2nd #:     0   0   1   1  |  0   0   1   1
                 --- --- --- --- | --- --- --- ---
Sum bit:          0   1   1   0  |  1   0   0   1
Carry-out bit:    0   0   0   1  |  0   1   1   1
```

- What would the rules be for hex addition?

Digital Representation of Information
(unsigned whole numbers)

- Adding numbers in binary example:
  - Add 111101011b and 1010010001b (illustrate)
- Adding numbers in hex examples:
  - Add 1EBh and A6h (illustrate)
  - Add 8A3Fh and 38CDh (illustrate)
  - Add 9B34h and 5AE6h (illustrate)

Digital Representation of Information
(prelude to signed whole numbers)

- What about ***subtraction***?
  - Computer treats subtracting a number as adding a *negative* number
  - Must first know how *negative* numbers are represented
- Can you think of a straightforward way to represent negative numbers?

(odd or even & multiplication by powers of 2)

- 2 features to note for *binary unsigned whole numbers*:
  - Number is even if LSB is 0, otherwise (LSB is 1) number is odd
    Quick quiz: What's the underlying *general truth*? (TIP: *Multiple* of *2*, *4*, …)
  - Shifting bits *left* by x positions (and filling rightmost x bits with 0's) → multiplying number by 2^x