Midterm 2 Review

INFO/CSE 100, Spring 2006

Fluency in Information Technology

http://www.cs.washington.edu/100

Readings and References

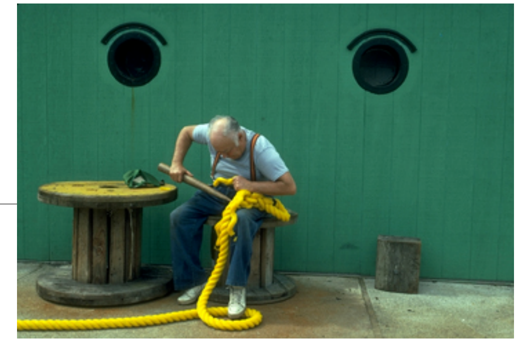
- Reading
  - » *Fluency with Information Technology*
    - Chapters 9, 11 18-21

Overview

- During this quarter, we're looking at the actual workings of computer systems
- Organized as "*layers of abstraction*"
  » application programs
  » higher level languages: Javascript, SQL, …
  » operating system concepts
  » bits, bytes, assembly language
  » transistors, electrons, photons

Layers of Abstraction

- At any level of abstraction, there are
  - » elements at that level
  - » the building blocks for those elements
- Abstraction
  - » isolates a layer from changes in the layer below
  - » improves developer productivity by reducing detail needed to accomplish a task
  - » helps define a single architecture that can be implemented with more than one organization




Architecture & Organization

- Architecture (the *logical definition*)
  - » defines elements and interfaces between layers
  - » Instruction Set Architecture
    - instructions, registers, addressing
- Organization (the *physical implementation*)
  - » components and connections
  - » how instructions are implemented in hardware
  - » many different organizations can implement a single architecture

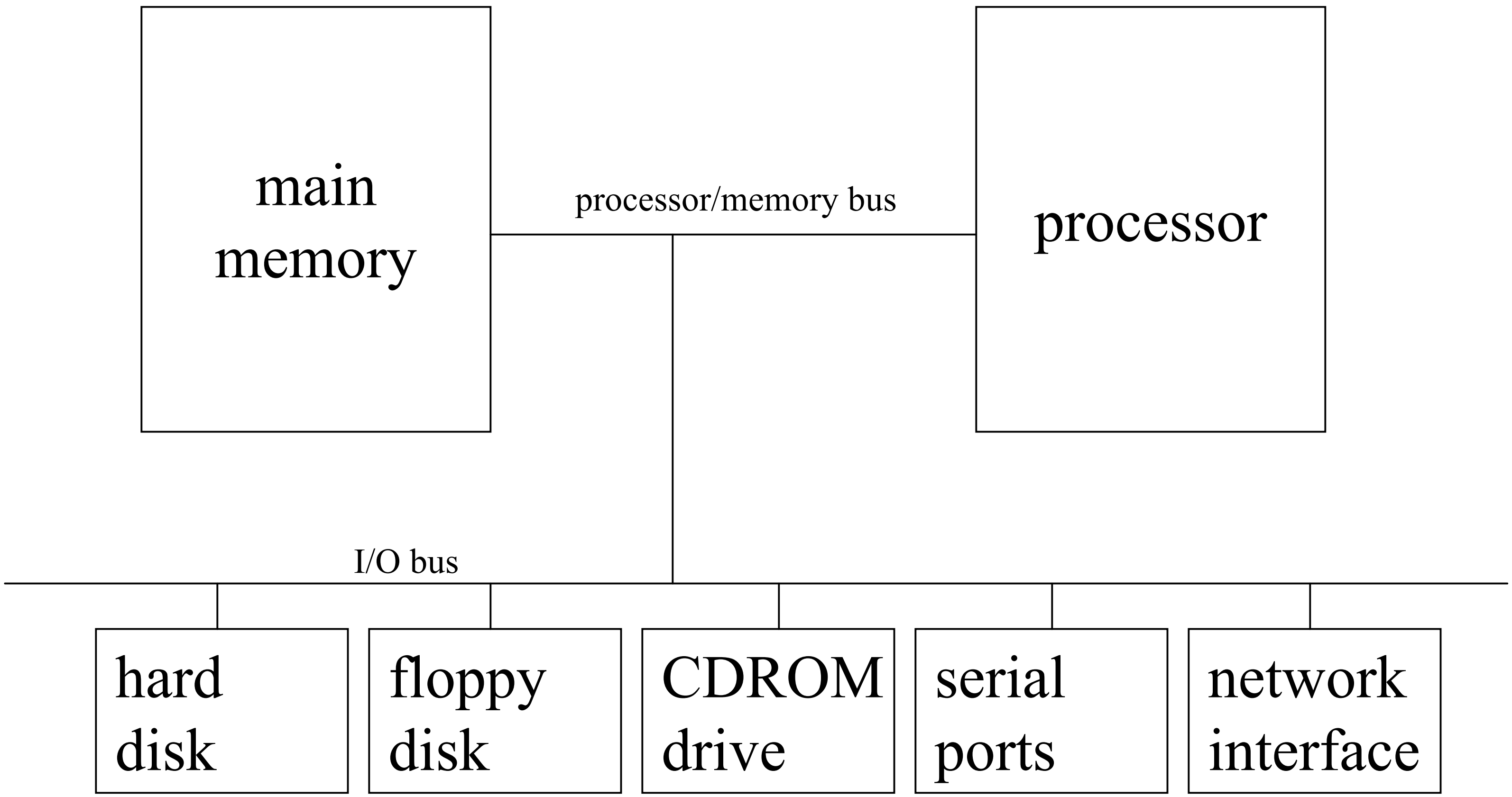
Computer Architecture

- Specification of how to program a specific computer family
  - » what instructions are available?
  - » how are the instructions formatted into bits?
  - » how many registers and what is their function?
  - » how is memory addressed?
- Some examples architectures
  - » IBM 360, 370, …
  - » PowerPC 601, 603, G5, …
  - » Intel x86 286, 386, 486, Pentium, …
  - » MIPS R2000, R3000, R4000, R5000, ...

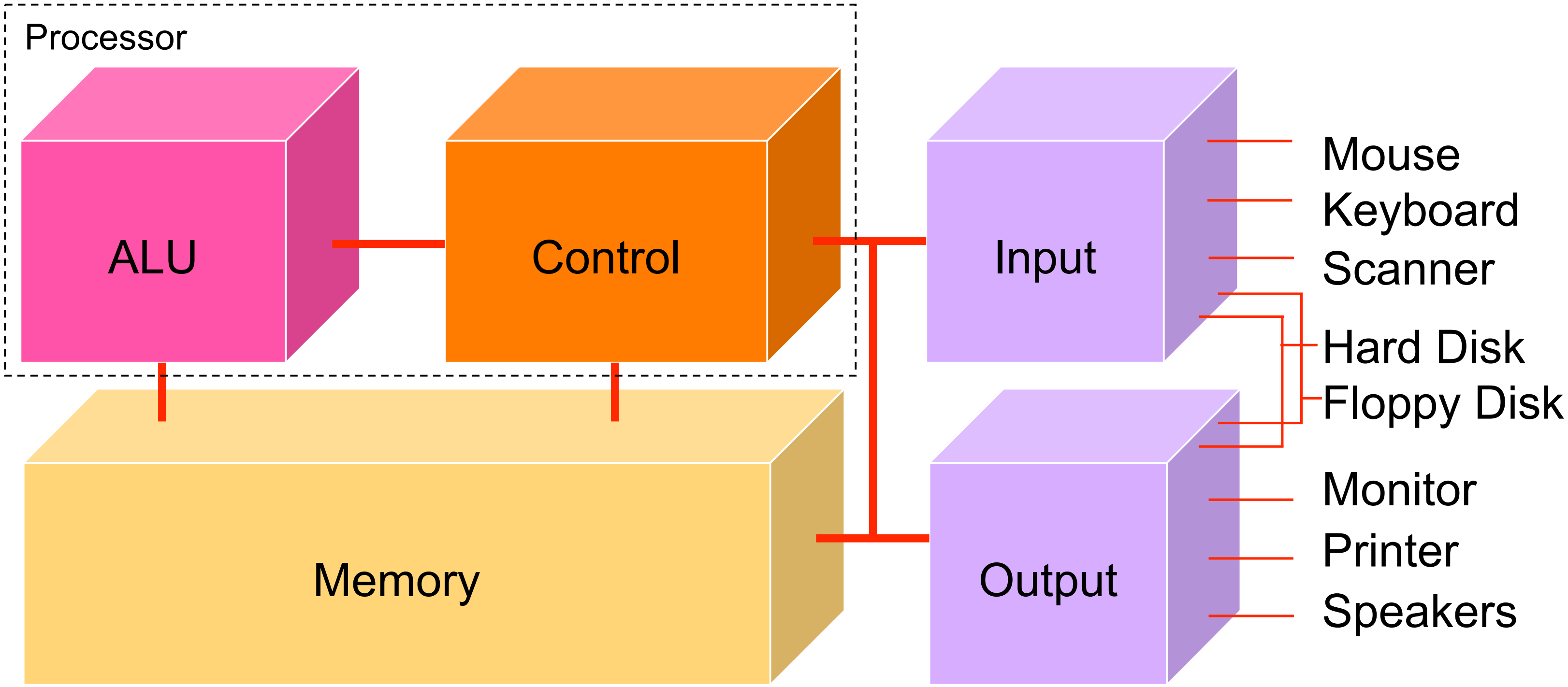
Computer Organization

- Processor
  - » Data path (ALU) manipulate the bits
  - » The control controls the manipulation
- Memory
  - » cache memory - smaller, higher speed
  - » main memory - larger, slower speed
- Input / Output
  - » interface to the rest of the world

A Typical Organization



Anatomy of a Computer



Fetch/Execute Cycle

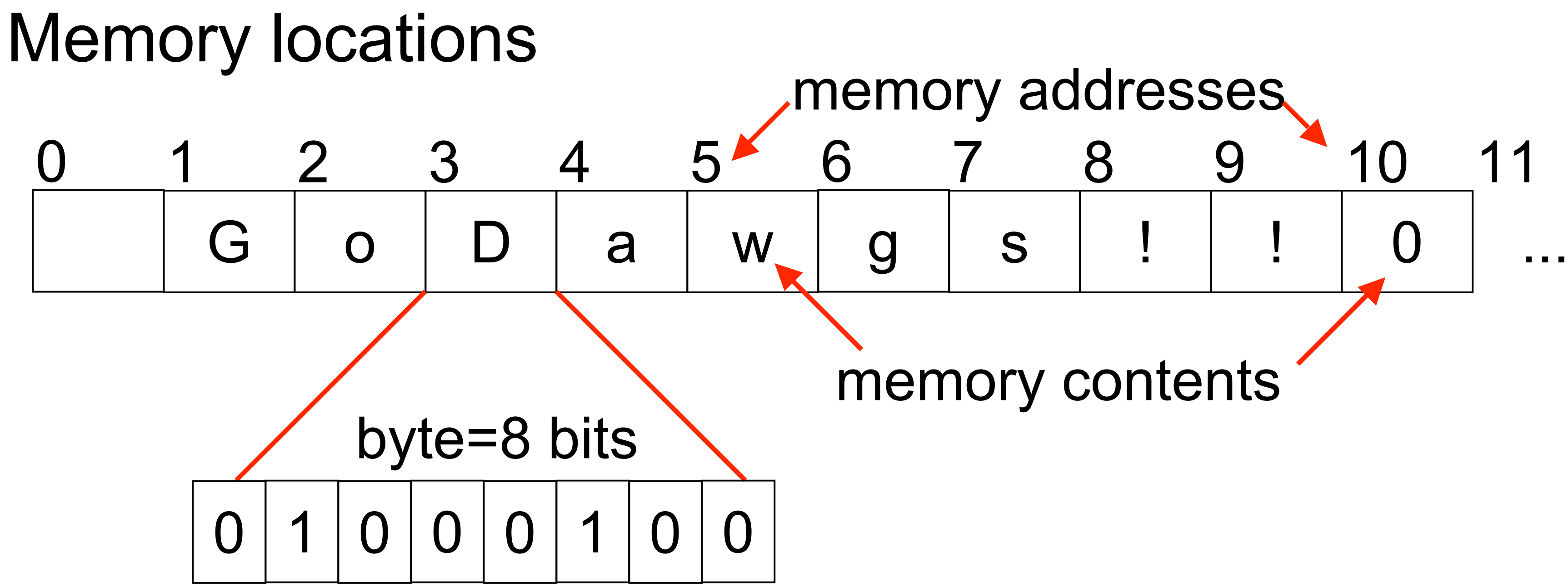
Computer = instruction execution engine

» The fetch/execute cycle is the process that executes instructions

Instruction Fetch (IF)
Instruction Decode (ID)
Data Fetch (DF)
Instruction Execution (EX)
Result Return (RR)

Memory ...

Programs and the data they operate on must be in the memory while they are running



Control

- The Fetch/Execute cycle is hardwired into the computer's control, i.e. it is the actual "engine"
- Depending on the Instruction Set Architecture, the instructions say things like
  - » Put in memory location 20 the contents of memory location 10 + contents of memory location 16
  - » The instructions executed have the form ADDB 10, 16, 20
    - Add the bytes from memory address 10 and memory address 16 and store the result in memory address 20

10	11	12	13	14	15	16	17	18	19	20	21
6						12				18	…

ALU

The Arithmetic/Logic Unit does the actual computation

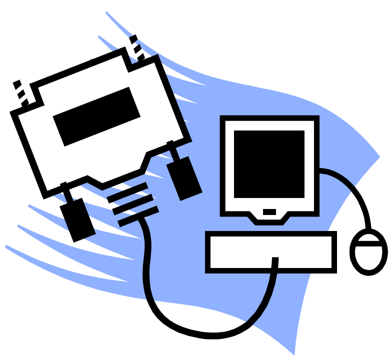
Depending on the Instruction Set Architecture, each type of data has its own separate instructions

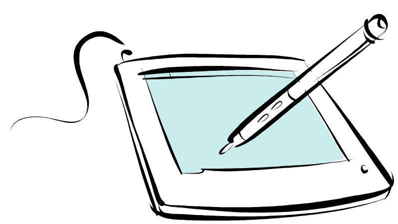
ADDB	: add bytes	ADDBU	: add bytes unsigned
ADDH	: add half words	ADDHU	: add halves unsigned
ADD	: add words	ADDU	: add words unsigned
ADDS	: add short decimal numbers		
ADDD	: add long decimal numbers		

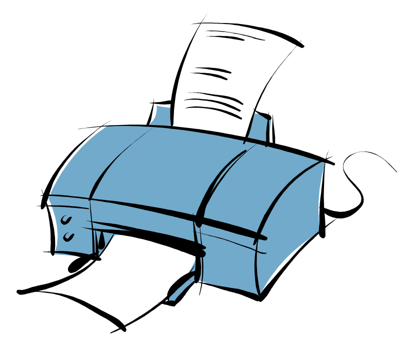
Most computers have only about a 100-150 instructions hard wired

Input/Output

- Input units bring data to memory from outside world; output units send data to outside world from memory
  - » Most peripheral devices are "dumb", meaning that the processor assists in their operation

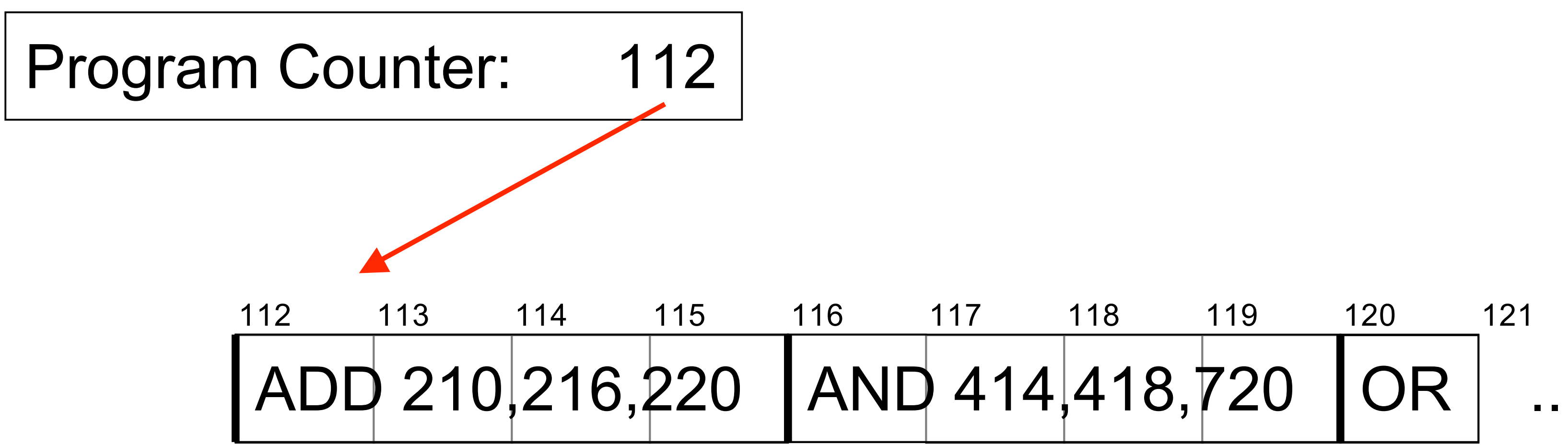






The PC's PC

- The program counter (PC) tells where the next instruction comes from
  - » In some architectures, instructions are always 4 bytes long, so add 4 to the PC to find the next instruction



Clocks Run The Engine

- The rate that a computer "spins around" the Fetch/Execute cycle is controlled by its clock
  - » Current clocks run 2-3 GHz
  - » The computer tries do at least one instruction per cycle, depending on the instruction and the availability of memory contents
  - » Modern processors often try to do more than one instruction per cycle

Clock rate is not a good indicator of speed anymore, because several things are happening every clock cycle

Algorithm

- Algorithm
  - » a precise, systematic method to produce a desired result

- For example, the placeholder technique for deleting a short string except where it occurs in longer strings is an algorithm with an easy specification:

longStringWithShortStringInIt ← placeholder
ShortString ← ε
placeholder ← longStringWithShortStringInIt

Programs vs Algorithms

- A program is an algorithm specialized to a particular situation
  - » an Algorithm

    longStringWithShortStringInIt ← placeholder

    ShortString ← e

    placeholder ← longStringWithShortStringInIt
  - » a Program that implements the Algorithm

    ↵↵ ← #   // replace double <newlines> with <#>

    ↵ ← e   // delete all single < newlines>

    # ← ↵↵   // restore all double <newlines>

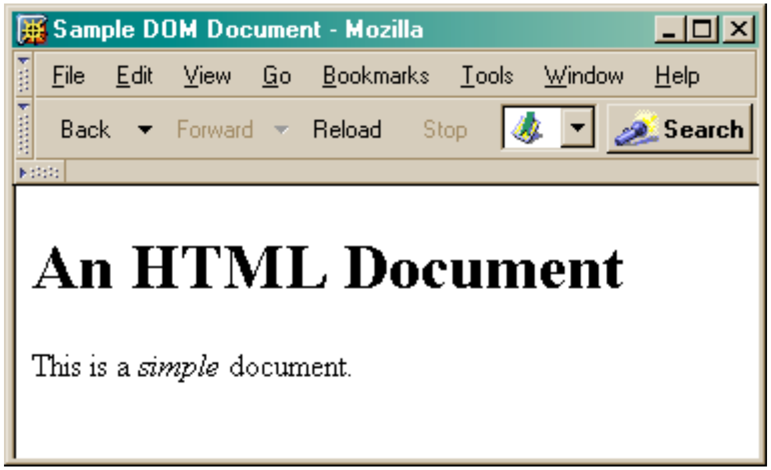
What the heck is the DOM?

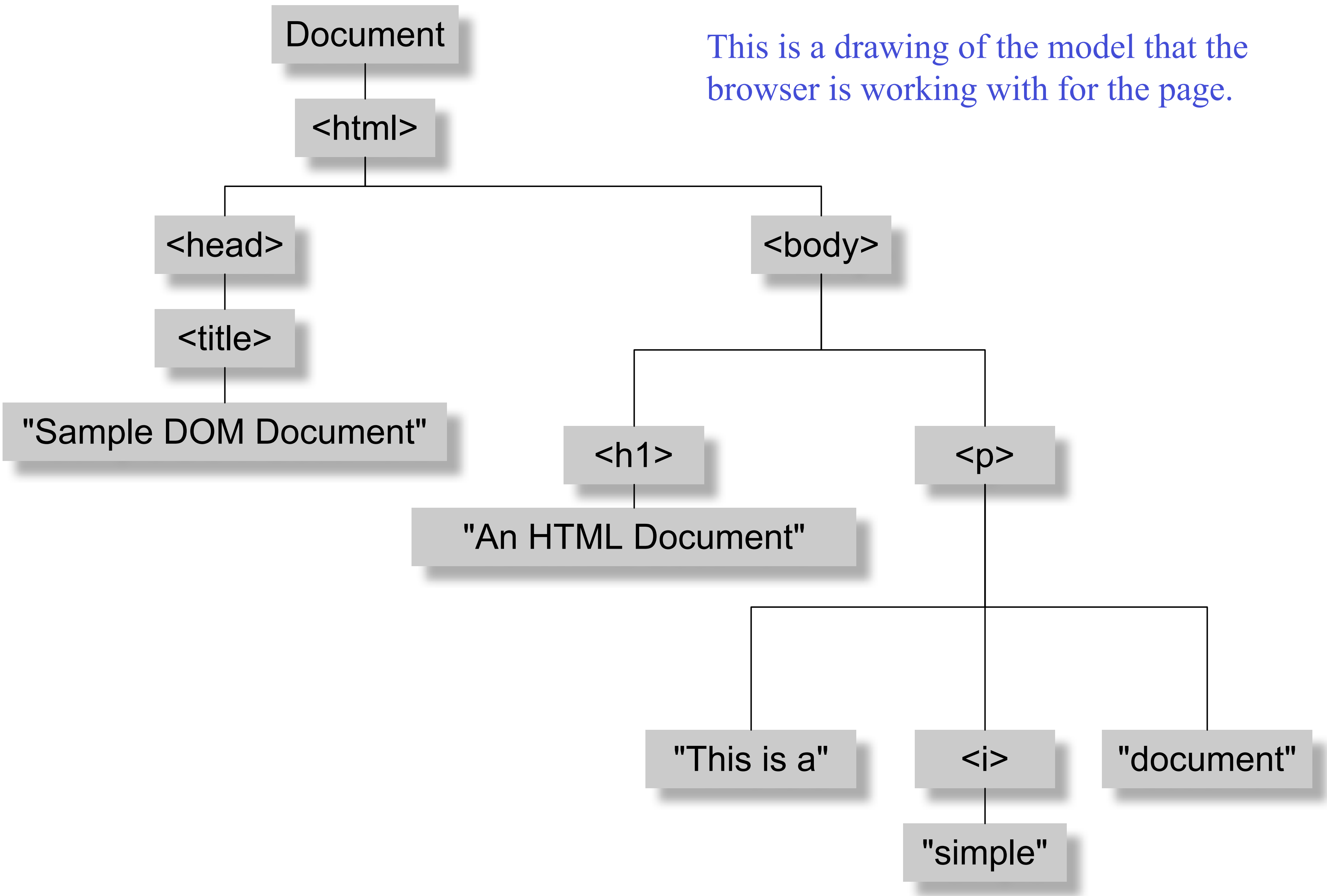
- Document Object Model
  - » Your web browser builds a *model* of the web page (the *document*) that includes all the *objects* in the page (tags, text, etc)
  - » All of the properties, methods, and events available to the web developer for manipulating and creating web pages are organized into objects
  - » Those objects are accessible via scripting languages in modern web browsers

This is what the browser reads (sampleDOM.html).

```
<html>
  <head>
    <title>Sample DOM Document</title>
  </head>
  <body>
    <h1>An HTML Document</h1>
    <p>This is a <i>simple</i> document.
  </body>
</html>
```

This is what the browser displays on screen.





This is a drawing of the model that the browser is working with for the page.

Figure 17-1. The tree representation of an HTML document
Copied from JavaScript by Flanagan.

`document.getElementById("radioLC").checked`

- Reference to several nodes in the model of the page that the browser constructed
- **`document`**
  - » The root of the tree is an object of type `HTMLDocument`
  - » Using the global variable `document`, we can access all the nodes in the tree, as well as useful functions and other global information
    - title, referrer, domain, URL, body, images, links, forms, ...
    - open, write, close, getElementById, ...

`document.getElementById("radioLC").checked`

- **`getElementById("radioLC")`**
  - » This is a predefined function that makes use of the `id` that can be defined for any element in the page
  - » An `id` must be unique in the page, so only one element is ever returned by this function
  - » The argument to `getElementById` specifies which element is being requested

`document.getElementById("radioLC").checked`

- `checked`
  - » This is a particular property of the node we are looking at, in this case, a radio button
  - » Each type of node has its own set of properties
    - for radio button: `checked, name, ...`
    - refer to the HTML DOM for specifics for each element type
  - » Some properties can be both read and set

Representing Data as Symbols

- 24 Greek Letters
- And we decide to use 2 symbols, binary, to represent the data.
- How many bits do we need?!?
  - » 24 total possibilities
  - » $2x2x2x2x2 = 2^5 = 32$
    - We get 6 extra!

Info Representation

- Adult humans have 32 teeth
  - » sometimes a tooth or two is missing!
- How can we represent a **set** of teeth?
  - » How many different items of information?
    - 2 items - *tooth* or *no tooth*
  - » How many "digits" or positions to use?
    - 32 positions - one per tooth socket
  - » Choose a set of symbols

no tooth: 0 *tooth*: 1

What's your tooth number?

incisors		canines	pre-molars		molars		
0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0

no teeth ↔ 0000 0000 0000 0000 0000 0000 0000 0000

1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1	0 0 0 0	0 0 0 0	0 0 0 0

no molars ↔ 1111 1111 1111 1111 1111 0000 0000 0000

How many possible **combinations**? $2\times2\times2\times2\times\ldots\times2 = 2^{32} \approx$ 4 Billion

The Information School of the University of Washington

How many positions should we use?

It depends: how many numbers do we need?

one position

0
1

two numbers

two positions

0	0
0	1
1	0
1	1

four numbers

three positions

0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

eight numbers

Converting from binary to decimal

			$2\times2\times2\times2$	$2\times2\times2$	2×2	2	1	
$2^7 = 128$	$2^6 = 64$	$2^5 = 32$	$2^4 = 16$	$2^3 = 8$	$2^2 = 4$	$2^1 = 2$	$2^0 = 1$	base 10
1	0	0	0	1	0	1	0	base 2

$$1\cdot 128 + 0\cdot 64 + 0\cdot 32 + 1\cdot 8 + 0\cdot 4 + 1\cdot 2 + 0\cdot 1 = 138_{10}$$

$$1\cdot 128 + 1\cdot 8 + 1\cdot 2 = 138_{10}$$

Each position represents one more multiplication by the base value.

For binary numbers, the base value is 2, so each new column represents a multiplication by 2.

Base 16 Hexadecimal

- The base value can be **16** - *hexadecimal numbers*
  - » Sixteen symbols: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
  - » Each column represents a multiplication by sixteen
  - » Hex is easier to use than binary because the numbers are shorter even though *they represent the same value*

16×16×16	16×16	16	1	
$16^3 = 4096$	$16^2 = 256$	$16^1 = 16$	$16^0 = 1$	base 10
0	0	8	A	base 16

$$8 \cdot 16 + 10 \cdot 1 = 138_{10}$$

Four binary bits ⇔ One hex digit

binary base 2				⇔	hexdecimal base 16	⇔	decimal base 10
0	0	0	0		0		0
0	0	0	1		1		1
0	0	1	0		2		2
0	0	1	1		3		3
0	1	0	0		4		4
0	1	0	1		5		5
0	1	1	0		6		6
0	1	1	1		7		7

binary base 2				⇔	hexdecimal base 16	⇔	decimal base 10
1	0	0	0		8		8
1	0	0	1		9		9
1	0	1	0		A		10
1	0	1	1		B		11
1	1	0	0		C		12
1	1	0	1		D		13
1	1	1	0		E		14
1	1	1	1		F		15

Binary to Hex examples

base 2	1000	0010	0000	0111	1010	0001	0000	1111
base 16	8	2	0	7	A	1	0	F

$$10000010000001111010000100001111_2 = 8207A10F_{16}$$

---

base 2	1000	0011	0100	0101	0110	1001	1011	1110

$$10000011010001010110100110111110_2 = \underline{\hspace{6cm}}_{16}$$

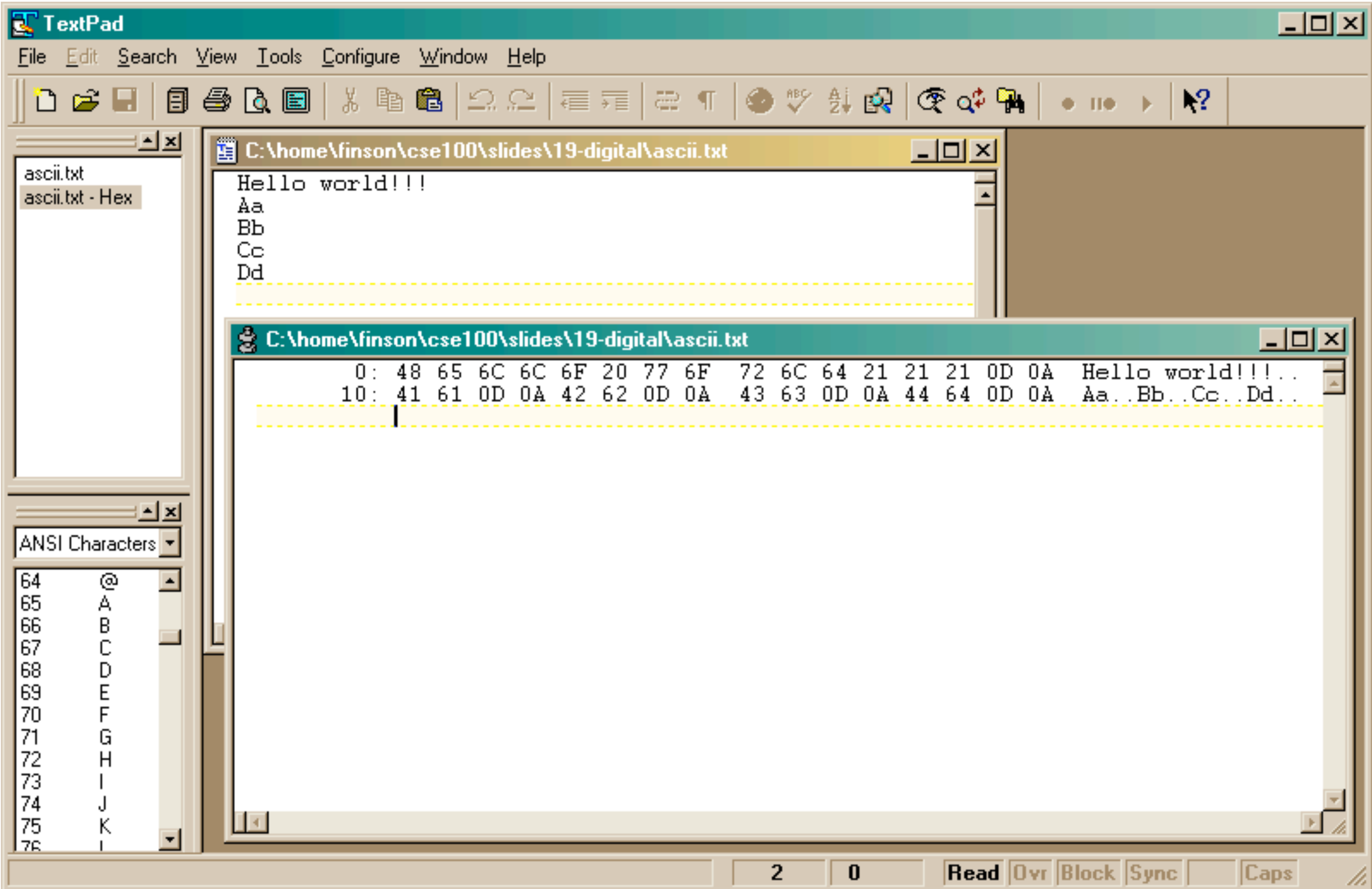
Represent Text - ASCII

- Assign a unique number to each character
  - » 7-bit ASCII
    - Range is 0 to 127 giving 128 possible values
    - There are 95 printable characters
    - There are 33 control codes like tab and carriage return

```
 !"#$%&'()*+,-./
0123456789:;<=>?
@ABCDEFGHIJKLMNO
PQRSTUVWXYZ[\]^_
`abcdefghijklmno
pqrstuvwxyz{|}~
```

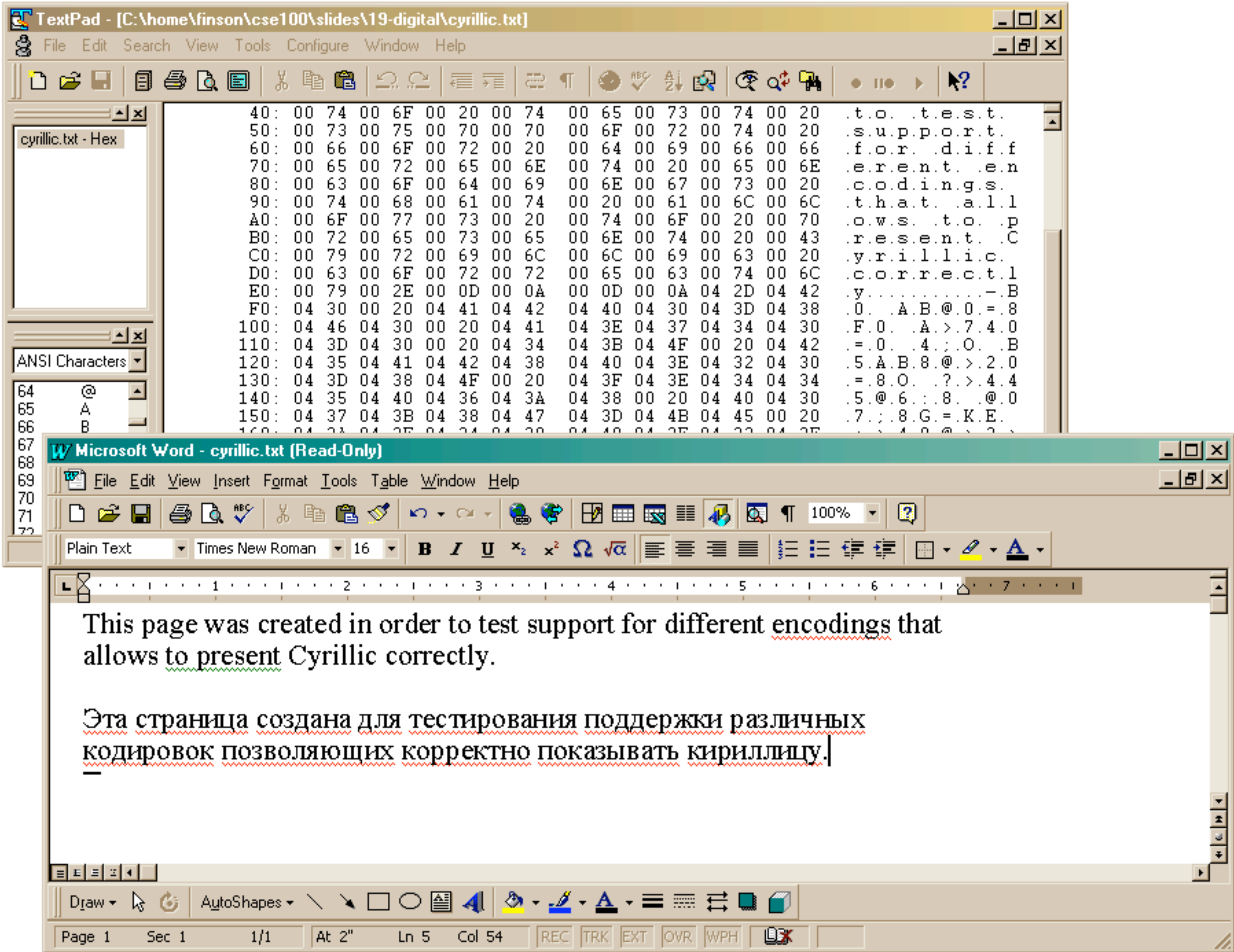
image is from Wikipedia

ASCII text


TextPad
File Edit Search View Tools Configure Window Help
ascii.txt
ascii.txt - Hex
C:\home\finson\cse100\slides\19-digital\ascii.txt
Hello world!!!
Aa
Bb
Cc
Dd
C:\home\finson\cse100\slides\19-digital\ascii.txt
0: 48 65 6C 6C 6F 20 77 6F 72 6C 64 21 21 21 0D 0A Hello world!!!..
10: 41 61 0D 0A 42 62 0D 0A 43 63 0D 0A 44 64 0D 0A Aa..Bb..Cc..Dd..
ANSI Characters
64 @
65 A
66 B
67 C
68 D
69 E
70 F
71 G
72 H
73 I
74 J
75 K
2
0
Read Ovr Block Sync Caps

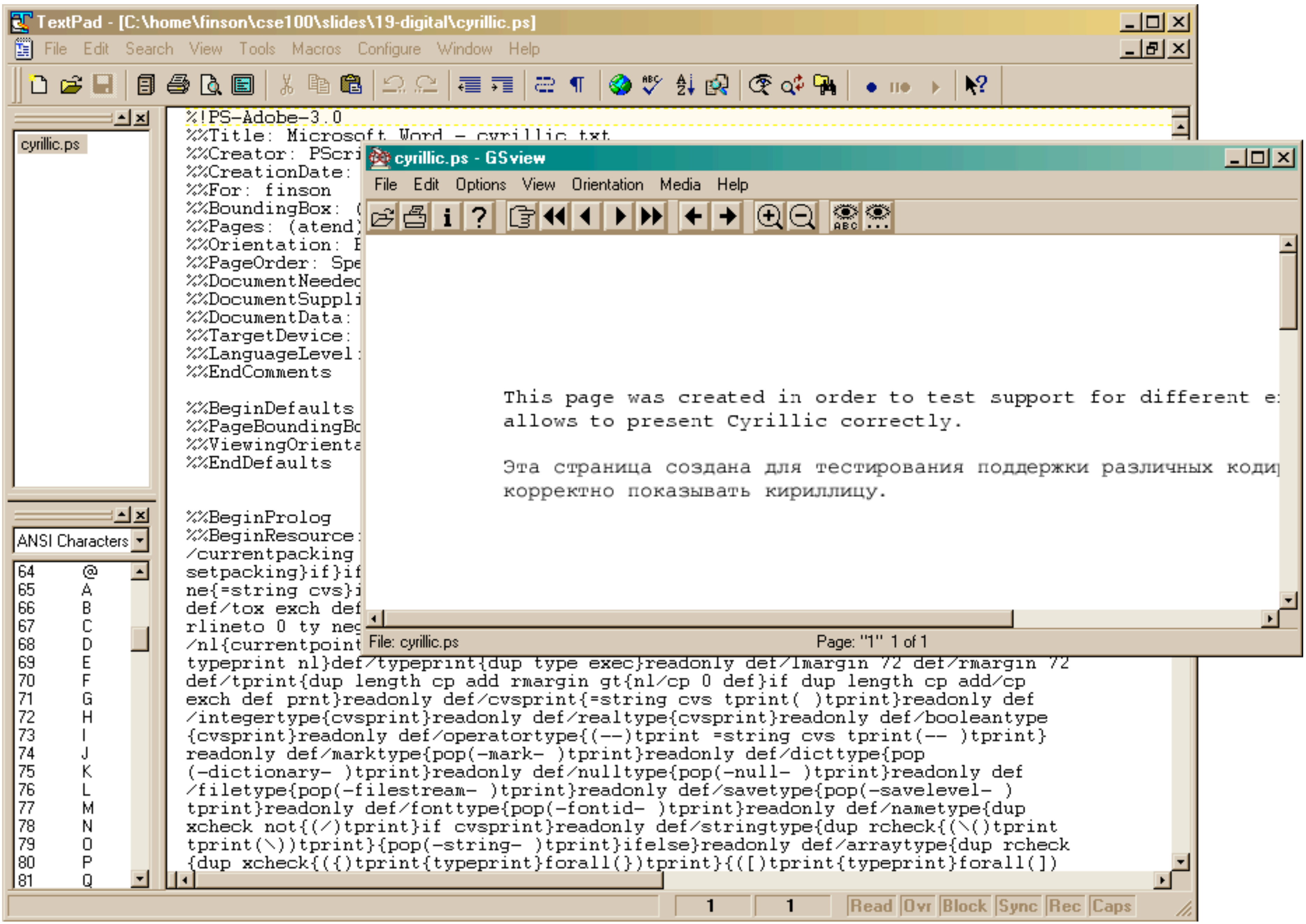
Represent Text - Unicode

- The goal of Unicode is to provide the means to encode the text of every document people want to store in computers
- Unicode aims to provide a unique number for each letter, without regard to typographic variations used by printers
- Unicode encodes each character in a number
  - » the number can be 7, 8, 16, or 32 bits long
  - » 16-bit encoding is common today


TextPad - [C:\home\finson\cse100\slides\19-digital\cyrillic.txt]
File Edit Search View Tools Configure Window Help
cyrillic.txt - Hex
ANSI Characters
40: 00 74 00 6F 00 20 00 74 00 65 00 73 00 74 00 20 .t.o. .t.e.s.t.
50: 00 73 00 75 00 70 00 70 00 6F 00 72 00 74 00 20 .s.u.p.p.o.r.t.
60: 00 66 00 6F 00 72 00 20 00 64 00 69 00 66 00 66 .f.o.r. .d.i.f.f
70: 00 65 00 72 00 65 00 6E 00 74 00 20 00 65 00 6E .e.r.e.n.t. .e.n
80: 00 63 00 6F 00 64 00 69 00 6E 00 67 00 73 00 20 .c.o.d.i.n.g.s.
90: 00 74 00 68 00 61 00 74 00 20 00 61 00 6C 00 6C .t.h.a.t. .a.l.l
A0: 00 6F 00 77 00 73 00 20 00 74 00 6F 00 20 00 70 .o.w.s. .t.o. .p
B0: 00 72 00 65 00 73 00 65 00 6E 00 74 00 20 00 43 .r.e.s.e.n.t. .C
C0: 00 79 00 72 00 69 00 6C 00 6C 00 69 00 63 00 20 .y.r.i.l.l.i.c.
D0: 00 63 00 6F 00 72 00 72 00 65 00 63 00 74 00 6C .c.o.r.r.e.c.t.l
E0: 00 79 00 2E 00 0D 00 0A 00 0D 00 0A 04 2D 04 42 .y.........-.B
F0: 04 30 00 20 04 41 04 42 04 40 04 30 04 3D 04 38 .0. .A.B.@.0.=.8
100: 04 46 04 30 00 20 04 41 04 3E 04 37 04 34 04 30 .F.0. .A.>.7.4.0
110: 04 3D 04 30 00 20 04 34 04 3B 04 4F 00 20 04 42 .=.0. .4.;.O. .B
120: 04 35 04 41 04 42 04 38 04 40 04 3E 04 32 04 30 .5.A.B.8.@.>.2.0
130: 04 3D 04 38 04 4F 00 20 04 3F 04 3E 04 34 04 34 .=.8.O. .?.>.4.4
140: 04 35 04 40 04 36 04 3A 04 38 00 20 04 40 04 30 .5.@.6.:.8. .@.0
150: 04 37 04 3B 04 38 04 47 04 3D 04 4B 04 45 00 20 .7.;.8.G.=.K.E.
Microsoft Word - cyrillic.txt (Read-Only)
File Edit View Insert Format Tools Table Window Help
Plain Text
Times New Roman
16
100%
This page was created in order to test support for different encodings that allows to present Cyrillic correctly.
Эта страница создана для тестирования поддержки различных кодировок позволяющих корректно показывать кириллицу.
Draw
AutoShapes
Page 1 Sec 1 1/1 At 2" Ln 5 Col 54 REC TRK EXT OVR WPH

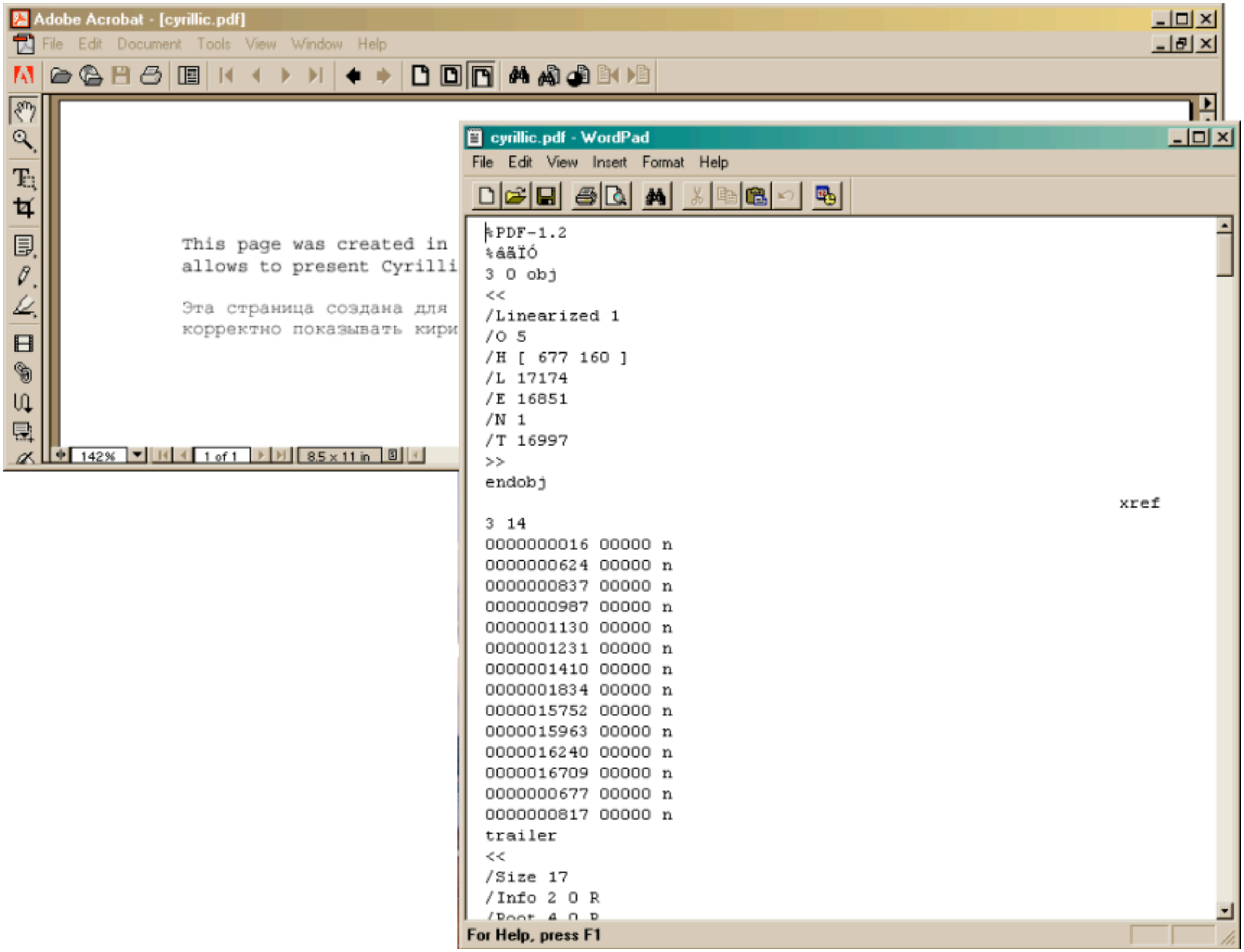
Represent Text - Postscript

- Postscript is a page description language somewhat like HTML
  - » The file is mostly text and can be looked at with a regular text editor
  - » programs that know what it is can interpret the embedded commands
  - » Programs *and printers* that understand Postscript format can display complex text and graphical images in a standard fashion


TextPad - [C:\home\finson\cse100\slides\19-digital\cyrillic.ps]
File Edit Search View Tools Macros Configure Window Help
cyrillic.ps
ANSI Characters
%!PS-Adobe-3.0
%%Title: Microsoft Word - cyrillic.txt
%%Creator: PScri
%%CreationDate:
%%For: finson
%%BoundingBox:
%%Pages: (atend)
%%Orientation:
%%PageOrder: Spe
%%DocumentNeedec
%%DocumentSuppli
%%DocumentData:
%%TargetDevice:
%%LanguageLevel:
%%EndComments
%%BeginDefaults
%%PageBoundingBo
%%ViewingOrienta
%%EndDefaults
%%BeginProlog
%%BeginResource:
/currentpacking
setpacking}if}if
ne{=string cvs}i
def/tox exch def
rlineto 0 ty neg
/nl{currentpoint
typeprint nl}def/typeprint{dup type exec}readonly def/lmargin 72 def/rmargin 72
def/tprint{dup length cp add rmargin gt{nl/cp 0 def}if dup length cp add/cp
exch def prnt}readonly def/cvsprint{=string cvs tprint( )tprint}readonly def
/integertype{cvsprint}readonly def/realtype{cvsprint}readonly def/booleantype
{cvsprint}readonly def/operatortype{(--)tprint =string cvs tprint(--)tprint}
readonly def/marktype{pop(-mark-)tprint}readonly def/dicttype{pop
(-dictionary-)tprint}readonly def/nulltype{pop(-null-)tprint}readonly def
/filetype{pop(-filestream-)tprint}readonly def/savetype{pop(-savelevel-)
tprint}readonly def/fonttype{pop(-fontid-)tprint}readonly def/nametype{dup
xcheck not{(/)tprint}if cvsprint}readonly def/stringtype{dup rcheck{(\()tprint
tprint(\))tprint}{pop(-string-)tprint}ifelse}readonly def/arraytype{dup rcheck
{dup xcheck{({)tprint{typeprint}forall(})tprint}{([)tprint{typeprint}forall(])
64 @
65 A
66 B
67 C
68 D
69 E
70 F
71 G
72 H
73 I
74 J
75 K
76 L
77 M
78 N
79 O
80 P
81 Q
1 1 Read Ovr Block Sync Rec Caps
cyrillic.ps - GSview
File Edit Options View Orientation Media Help
This page was created in order to test support for different e
allows to present Cyrillic correctly.
Эта страница создана для тестирования поддержки различных коди
корректно показывать кириллицу.
File: cyrillic.ps
Page: "1" 1 of 1

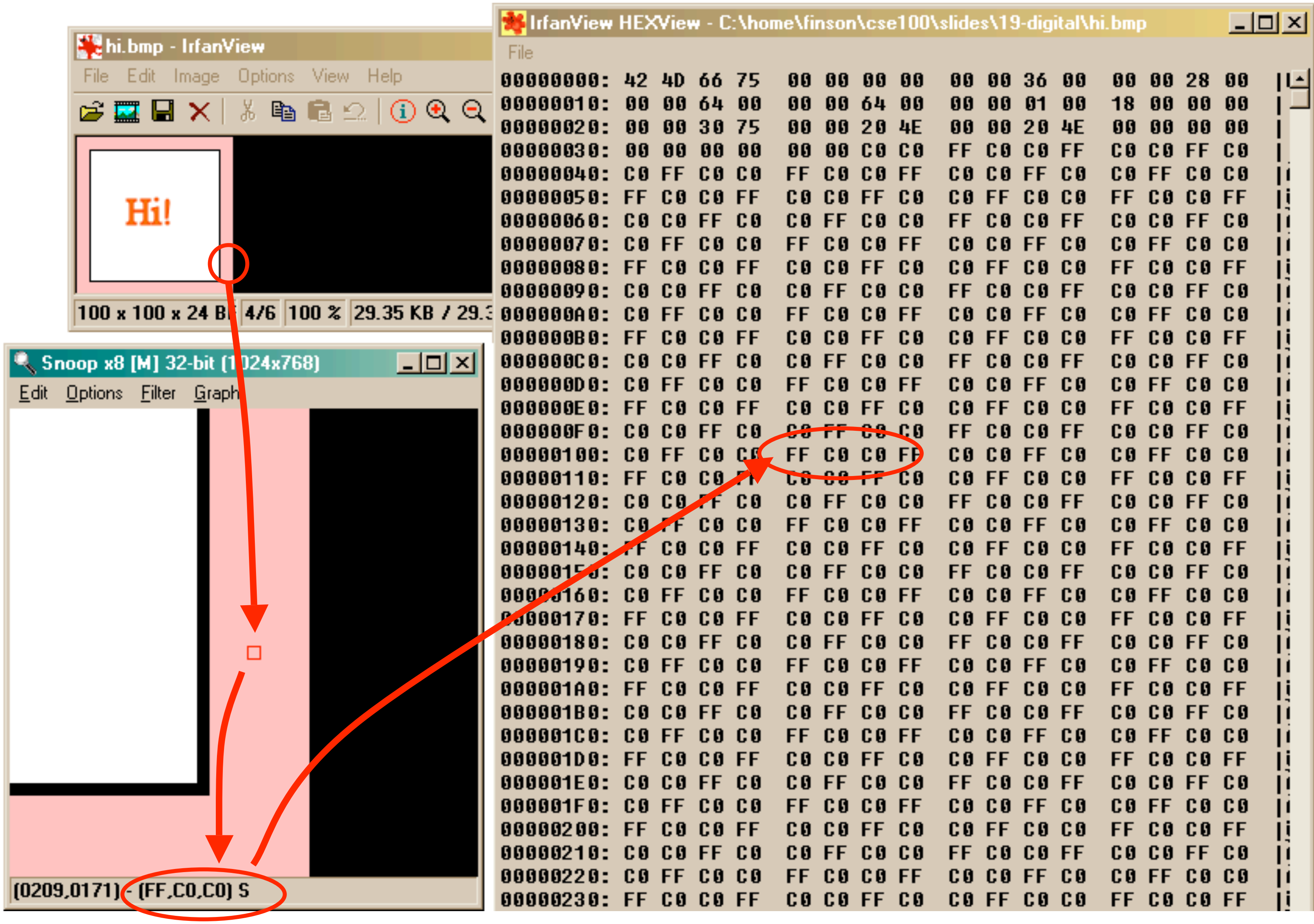
Represent Text - PDF

- PDF is another page description language based on Postscript
- The file is mostly text
  - » can be looked at with a regular text editor
  - » programs that know what it is can interpret the embedded commands
  - » just like Postscript and HTML in that respect


Adobe Acrobat - [cyrillic.pdf]
File Edit Document Tools View Window Help
This page was created in
allows to present Cyrilli
Эта страница создана для
корректно показывать кири
142%
1 of 1
8.5 x 11 in
cyrillic.pdf - WordPad
File Edit View Insert Format Help
%PDF-1.2
%âãÏÓ
3 0 obj
<<
/Linearized 1
/O 5
/H [677 160]
/L 17174
/E 16851
/N 1
/T 16997
>>
endobj
xref
3 14
0000000016 00000 n
0000000624 00000 n
0000000837 00000 n
0000000987 00000 n
0000001130 00000 n
0000001231 00000 n
0000001410 00000 n
0000001834 00000 n
0000015752 00000 n
0000015963 00000 n
0000016240 00000 n
0000016709 00000 n
0000000677 00000 n
0000000817 00000 n
trailer
<<
/Size 17
/Info 2 0 R
For Help, press F1

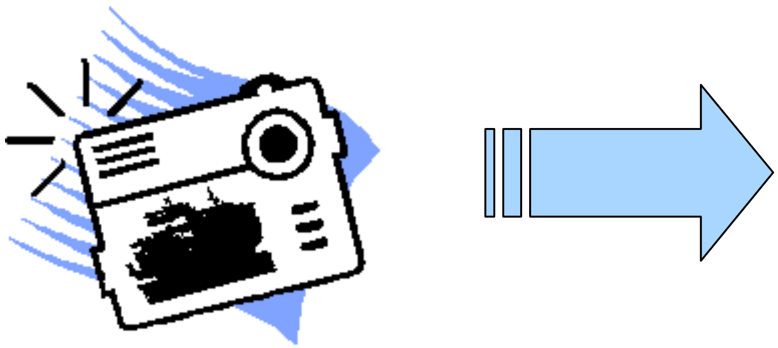
Represent Color - Bit Map

- Numbers can represent anything we want
- Recall that we can represent colors with three values
  - » Red, Green, Blue brightness values
- There are *numerous* formats for image files
  - » All of them store some sort of numeric representation of the brightness of each color at each pixel of the image
  - » commonly use 0 to 255 range (or 0 to FF_{16})

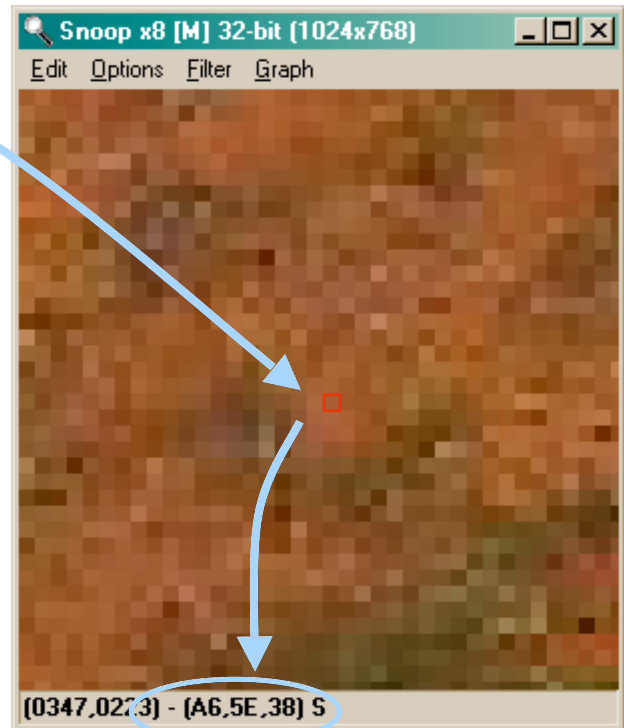

hi.bmp - IrfanView
File Edit Image Options View Help
Hi!
100 x 100 x 24 BP
4/6
100 %
29.35 KB / 29.3
Snoop x8 [M] 32-bit (1024x768)
Edit Options Filter Graph
(0209,0171) - (FF,C0,C0) S
IrfanView HEXView - C:\home\finson\cse100\slides\19-digital\hi.bmp
File
00000000: 42 4D 66 75 00 00 00 00 00 00 36 00 00 00 28 00
00000010: 00 00 64 00 00 00 64 00 00 00 01 00 18 00 00 00
00000020: 00 00 30 75 00 00 20 4E 00 00 20 4E 00 00 00 00
00000030: 00 00 00 00 00 00 C0 C0 FF C0 C0 FF C0 C0 FF C0
00000040: C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0
00000050: FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF
00000060: C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0
00000070: C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0
00000080: FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF
00000090: C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0
000000A0: C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0
000000B0: FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF
000000C0: C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0
000000D0: C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0
000000E0: FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF
000000F0: C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0
00000100: C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0
00000110: FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF
00000120: C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0
00000130: C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0
00000140: FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF
00000150: C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0
00000160: C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0
00000170: FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF
00000180: C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0
00000190: C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0
000001A0: FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF
000001B0: C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0
000001C0: C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0
000001D0: FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF
000001E0: C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0
000001F0: C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0
00000200: FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF
00000210: C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0
00000220: C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0
00000230: FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF C0 C0 FF

What about "continuous" signals?

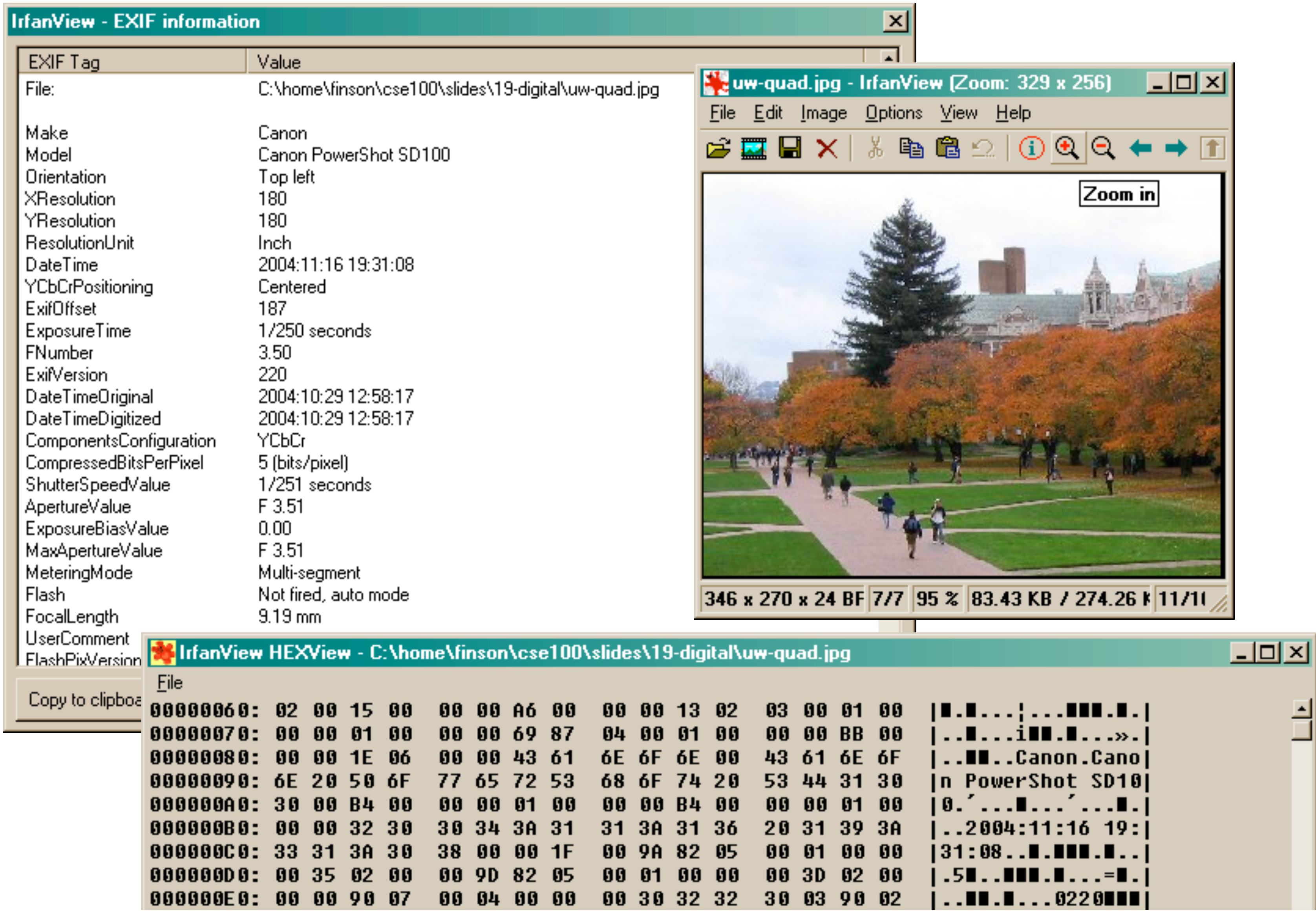
- Color and sound are natural quantities that don't come in nice discrete numeric quantities
- But we can "make it so!"



Digitized image contains color data



And much, much more!



Summary

- Bits can represent any information
  - » Discrete information is directly encoded using binary
  - » Continuous information is made discrete
- We can look at the bits in different ways
  - » The format guides us in how to interpret it
  - » Different interpretations let us work with the data in different ways