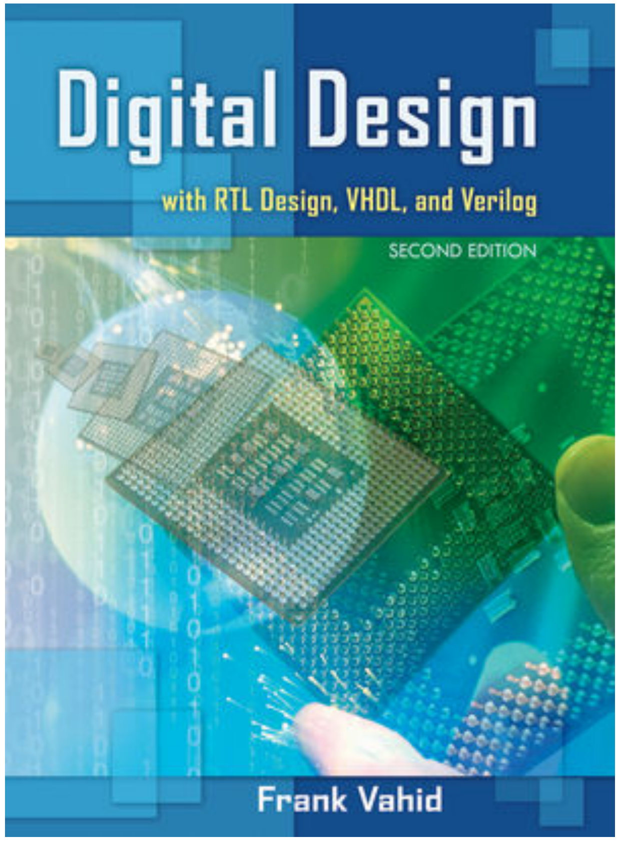
Introduction to Class (1/2)

2150686901
Digital Circuit Design

- Lecture Web site ; SmartCampus
  - 학교 강의웹사이트

- Lecture Objectives
  - 개요 ; 디지털 회로 동작 원리 이해 및 설계 방법 학습 및 실습
  - 교육목표 ;
    - 조합 및 순차 디지털 회로 원리 및 동작 이해
    - Verilog HDL 에 의한 디지털 회로 해석 이해 및 실습
    - RTL 디자인 방법 이해
    - Zedboard/VIVADO 에서의 실제 디지털 회로 설계 실습

Textbook 1



2nd ed, Wiley. 2011.

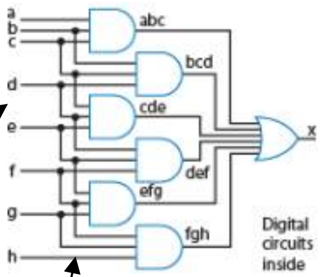
http://www.amazon.com/Digital-Design-RTL-VHDL-Verilog/dp/0470531088

Introduction to Digital Signal, Digitization and Digital Representation

- 교재 1장

Why Study Digital Design?

- Look “under the hood” of computers
  - Solid understanding → confidence, insight, even better programmer when aware of hardware resource issues
- Electronic devices becoming digital
  - Enabled by shrinking and more capable chips
  - Enables:
    - Better devices: Sound recorders, cameras, cars, cell phones, medical devices,...
    - New devices: Video games, PDAs, ...
  - Known as “embedded systems”
    - Thousands of new devices every year
    - Designers needed: Potential career direction



1995	1997	1999	2001	2003	2005	2007	
Portable music players	Satellites	Cell phones	DVD players	Cameras	Video recorders	TVs	Musical instruments ???

- Years shown above indicate when digital version began to *dominate*
  - (*Not* the first year that a digital version appeared)

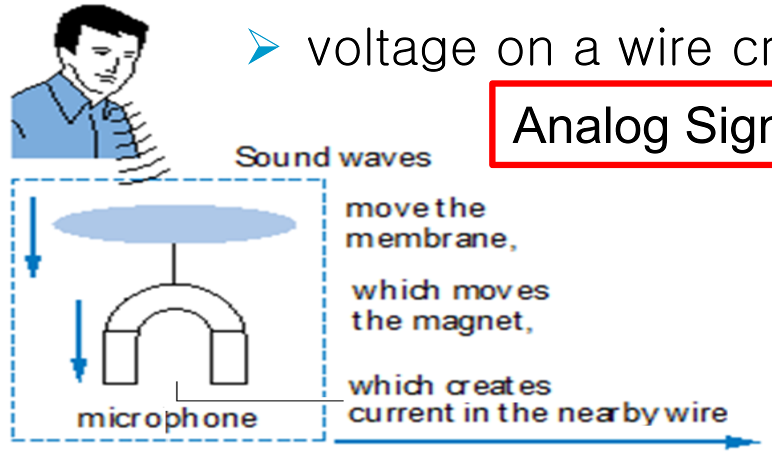
What Does “Digital” Mean?

Digital versus Analog

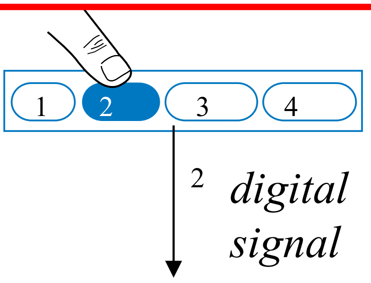
- Digital Signal : signal that at any time one of a finite of possible values
  - The number of fingers you hold up,
  - digital : Latin word, digit meaning finger
- Analog signal or Continuous Signal : One of an infinite of possible values :
  - voltage on a wire created by microphone

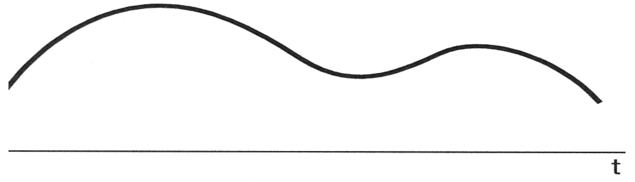
analog : 연속, 유사한, 상사형

Analog Signal→ Sampling, Quantization, Coding→ Digital Signal

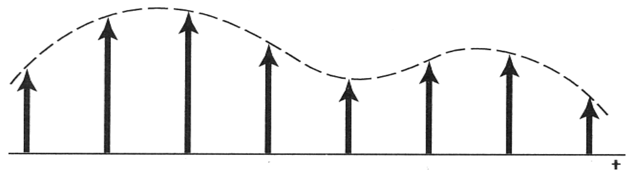


Analog Signal

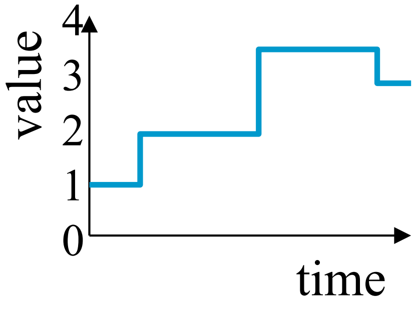




Analog signal or Continuous signal



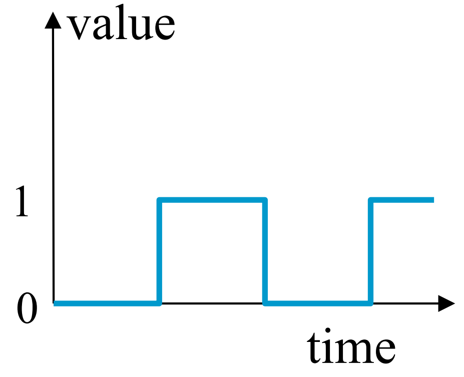
Discrete signal Or sampling signal



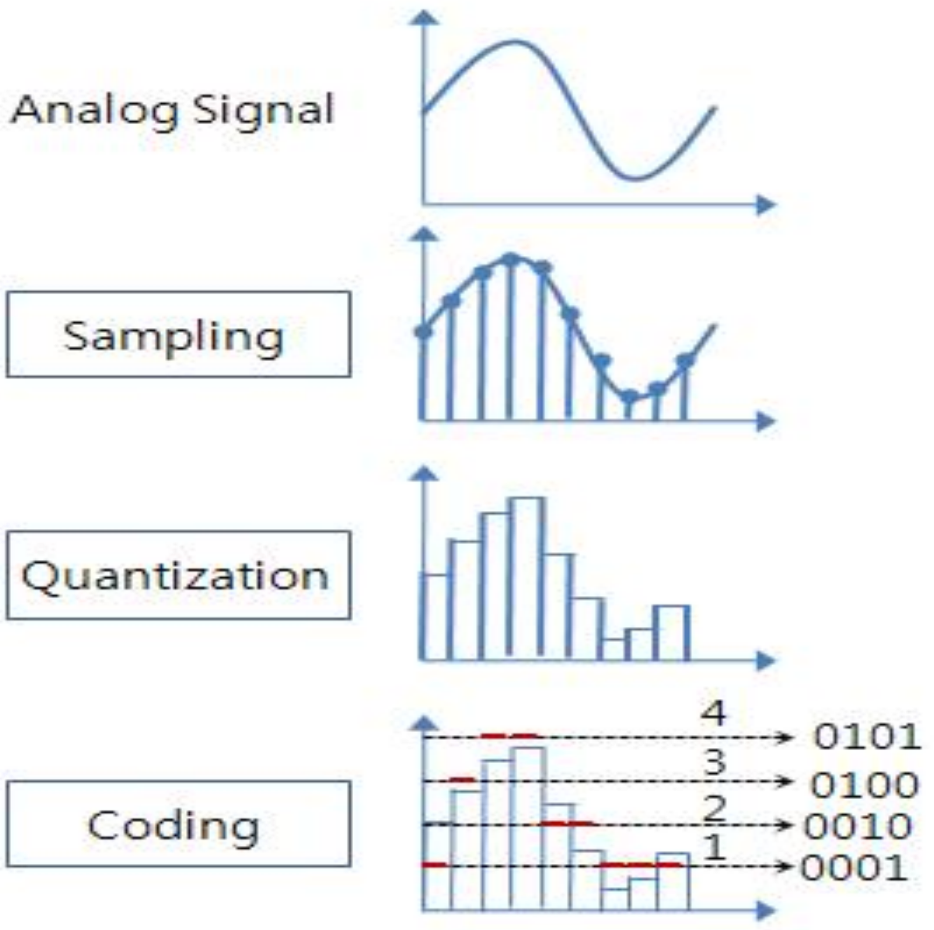
Possible values: 0, 1, 2, 3, or 4. That’s it.

Digital Signals with Only Two Values: Binary

- Most common digital signal
  : only two possible values
  - Typically represented as 0 and 1,   on or off,
  - A single binary signal : binary digit or bit
  - We'll only consider *binary* digital signals
  - Binary is popular because
    - Transistors, the basic digital electric component, operate using *two* voltages
    - Storing/transmitting one of *two* values is easier than three or more
  - Digital System : takes digital input, generates digital output
  - Digital Circuit : digital components that together comprise a digital system
  - Digital electronics became extremely popular : after invention of transistor
    electric switch: turn on or off



Process of Digitization



How to Encode Text: ASCII, Unicode

- ASCII (American Standard Code for Information Interchange) : 7- (or 8-) bit encoding of each letter, number, or symbol
- Unicode (Universal Code system) : Increasingly popular 16-bit encoding
  - Encodes characters from various world languages
  - 데이터 교환을 원할하게 하기 위하여 문자 1개에 부여되는 값을 16bit로 통일.

Sample ASCII encodings

Encoding	Symbol
010 0000	<space>
010 0001	!
010 0010	"
010 0011	#
010 0100	$
010 0101	%
010 0110	&
010 0111	'
010 1000	(
010 1001	)
010 1010	*
010 1011	+
010 1100	,
010 1101	-
010 1110	.
010 1111	/

Encoding	Symbol
100 0001	A
100 0010	B
100 0011	C
100 0100	D
100 0101	E
100 0110	F
100 0111	G
100 1000	H
100 1001	I
100 1010	J
100 1011	K
100 1100	L
100 1101	M

Encoding	Symbol
100 1110	N
100 1111	O
101 0000	P
101 0001	Q
101 0010	R
101 0011	S
101 0100	T
101 0101	U
101 0110	V
101 0111	W
101 1000	X
101 1001	Y
101 1010	Z

Encoding	Symbol
110 0001	a
110 0010	b
...	
111 1001	y
111 1010	z
011 0000	0
011 0001	1
011 0010	2
011 0011	3
011 0100	4
011 0101	5
011 0110	6
011 0111	7
011 1000	8
011 1001	9

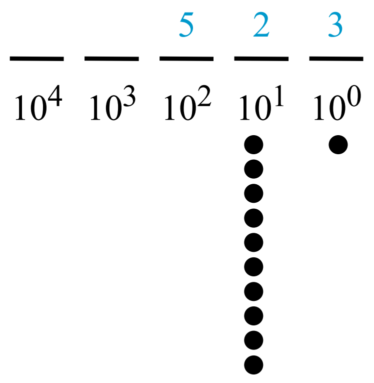
Question:
What does this ASCII bit sequence represent?

1010010 1000101 1010011 1010100

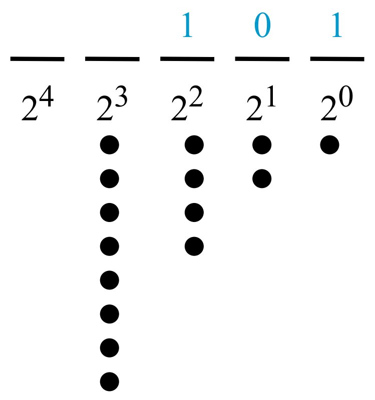
R E S T

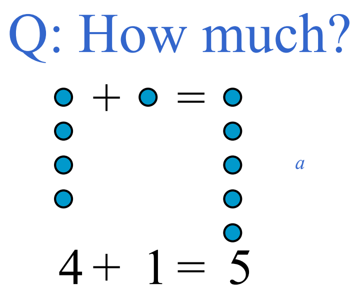
How to Encode Numbers: Binary Numbers

- Most important use of digital circuit : perform arithmetic computations
  To perform arithmetic computation ⇒ encode numbers as bits
- Each position represents a quantity
  ; symbol in position means how many of that quantity



- Base ten (*decimal*)
  - Ten symbols : 0, 1, 2, ..., 8, and 9
  - More than 9 -- next position
    - So each position power of 10
  - Nothing special about base 10
    -- used because we have 10 fingers
- Base two (*binary*)
  - Two symbols: 0 and 1
  - More than 1 -- next position
    - So each position power of 2

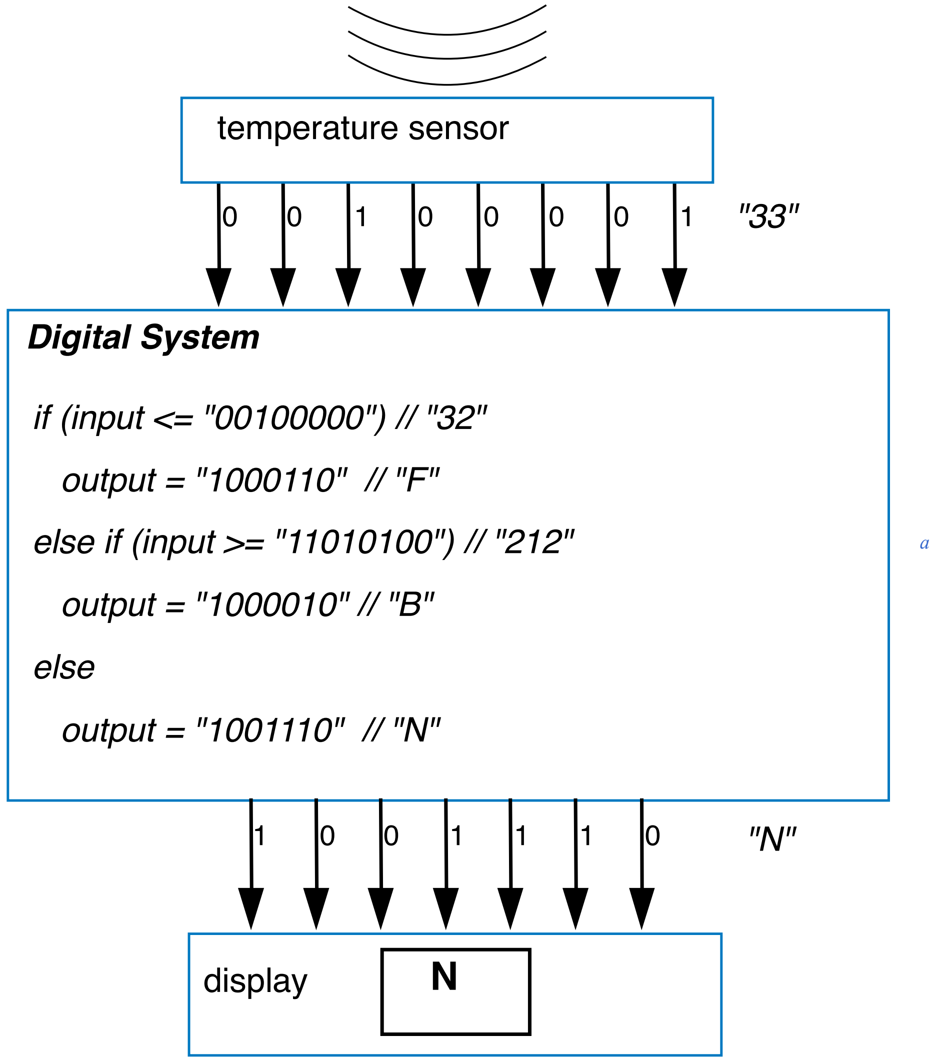




- 진법: 자리와 값의 관계에 관한 규칙

Using Digital Data in a Digital System

- A temperature sensor outputs temperature in binary
- The system reads the temperature, outputs ASCII code:
  - "F" for freezing (0–32)
  - "B" for boiling (212 or more)
  - "N" for normal
- A display converts its ASCII input to the corresponding letter



```
if (input <= "00100000") // "32"
   output = "1000110"  // "F"
else if (input >= "11010100") // "212"
   output = "1000010" // "B"
else
   output = "1001110"  // "N"
```

Converting from Binary to Decimal

- Just add weights
  - 1_2 is just $1*2^0$, or 1_{10}.
  - 110_2 is $1*2^2 + 1*2^1 + 0*2^0$, or 6_{10}. We might think of this using base ten weights: $1*4 + 1*2 + 0*1$, or 6.
  - 10000_2 is $1*16 + 0*8 + 0*4 + 0*2 + 0*1$, or 16_{10}.
  - 10000111_2 is $1*128 + 1*4 + 1*2 + 1*1 = 135_{10}$. Notice this time that we didn't bother to write the weights having a 0 bit.
  - 00110_2 is the same as 110_2 above — the leading 0's don't change the value.

Useful to know powers of 2:

2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
512	256	128	64	32	16	8	4	2	1

Practice counting up by powers of 2: 512 256 128 64 32 16 8 4 2 1

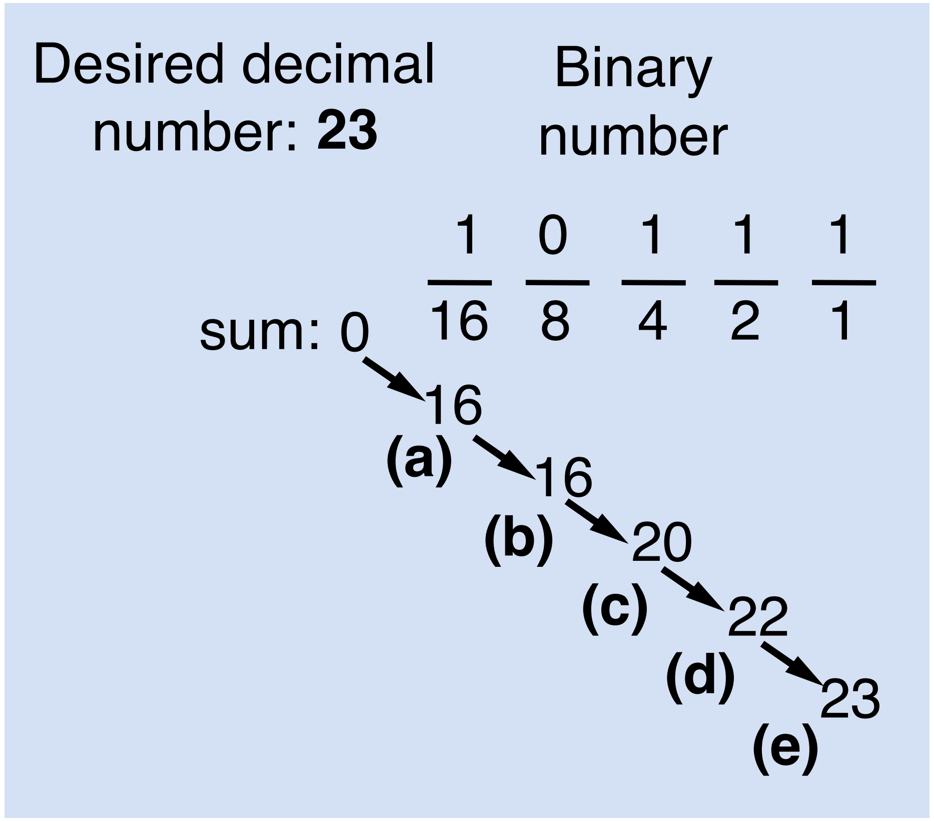
Converting from Decimal to Binary

- Put 1 in leftmost place without sum exceeding number
- Track sum

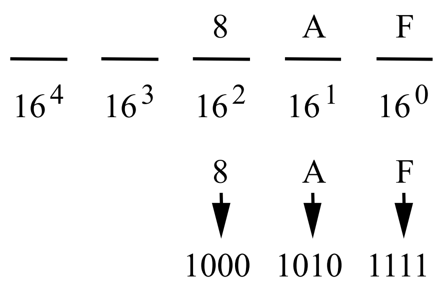
	Desired decimal number: **12**	Current sum	Binary number
(a)	16 > 12, too big; Put 0 in 16's place	0	$\underset{\boldsymbol{16}}{\underline{\mathbf{0}}}\ \underset{8}{\underline{\quad}}\ \underset{4}{\underline{\quad}}\ \underset{2}{\underline{\quad}}\ \underset{1}{\underline{\quad}}$
(b)	8 <= 12, so put 1 in 8's place, current sum is 8	8	$\underset{16}{\underline{0}}\ \underset{\boldsymbol{8}}{\underline{\mathbf{1}}}\ \underset{4}{\underline{\quad}}\ \underset{2}{\underline{\quad}}\ \underset{1}{\underline{\quad}}$
(c)	8+4=12 <= 12, so put 1 in 4's place, current sum is 12	12	$\underset{16}{\underline{0}}\ \underset{8}{\underline{1}}\ \underset{\boldsymbol{4}}{\underline{\mathbf{1}}}\ \underset{2}{\underline{\quad}}\ \underset{1}{\underline{\quad}}$
(d)	Reached desired 12, so put 0s in remaining places	*done*	$\underset{16}{\underline{0}}\ \underset{8}{\underline{1}}\ \underset{4}{\underline{1}}\ \underset{2}{\underline{0}}\ \underset{1}{\underline{0}}$

Converting from Decimal to Binary

- Example using a more compact notation



Base Sixteen: Another Base Used by Designers

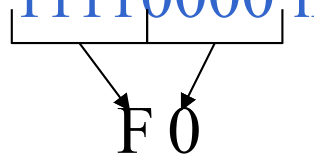


hex	binary
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111

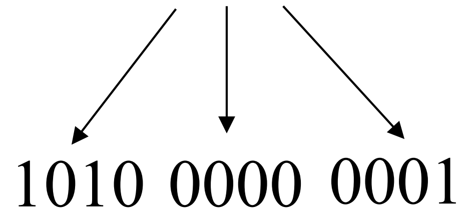
hex	binary
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

- Nice because each position represents four base-two positions
  - Compact way to write binary numbers
- Known as *hexadecimal*, or just *hex*

Q: Write 11110000 in hex



Q: Convert hex A01 to binary



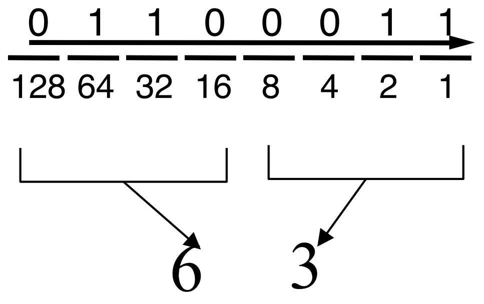
Ex. 1.6 : Decimal to Hex

- Easy method: convert to binary first, then binary to hex

Convert 99 base 10 to hex

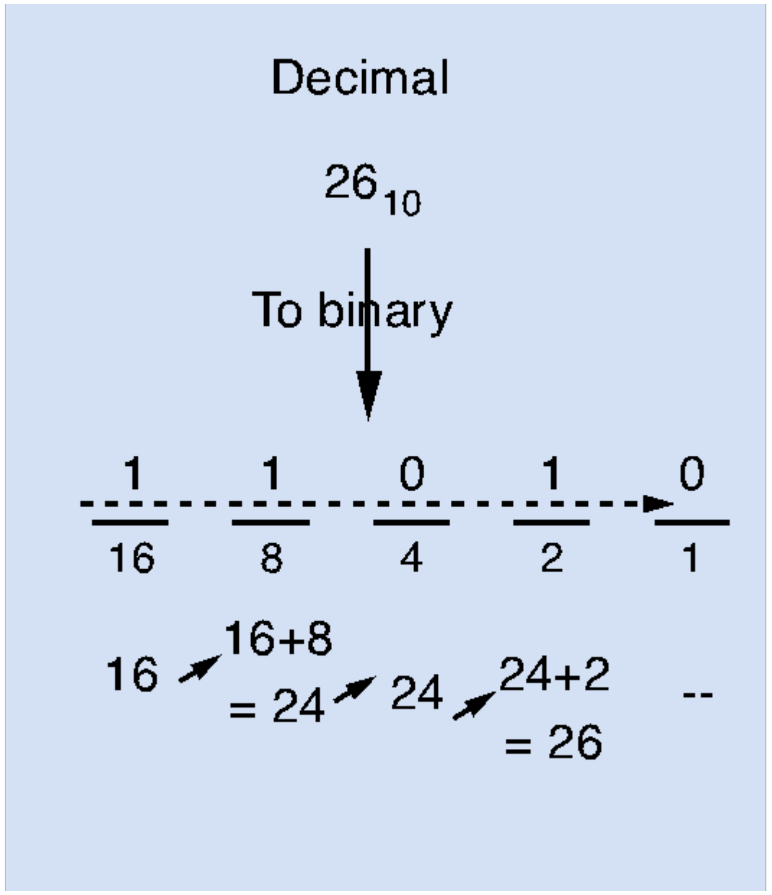
First convert to binary:

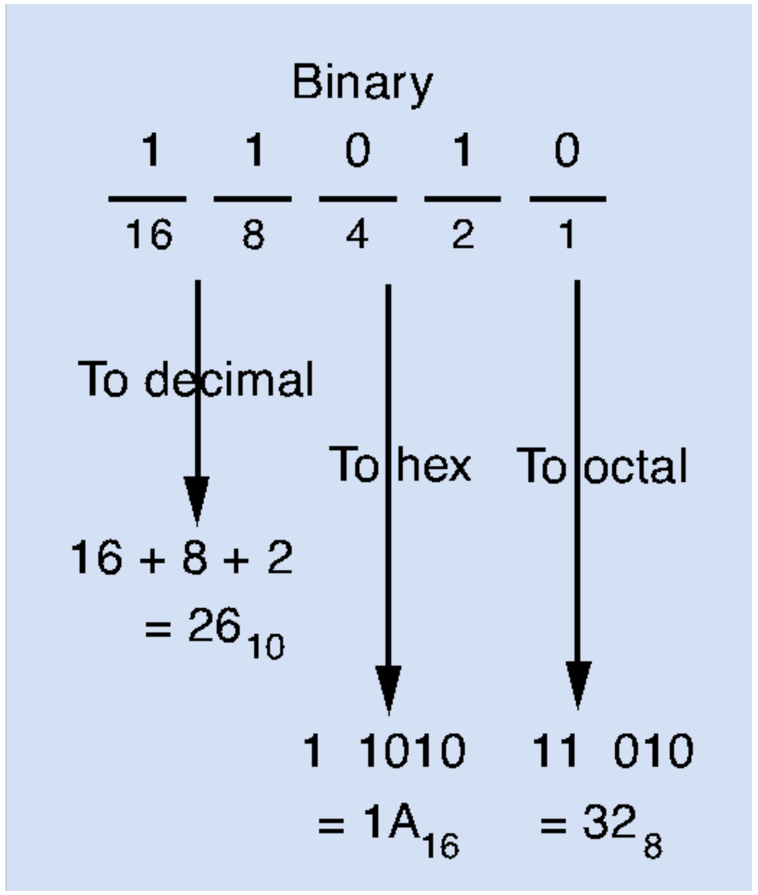
Then binary to hex:



(Quick check: 6*16 + 3*1 = 96+3 = 99)

Converting To/From Binary by Hand: Summary





Divide-By-2 Method Common in Automatic Conversion

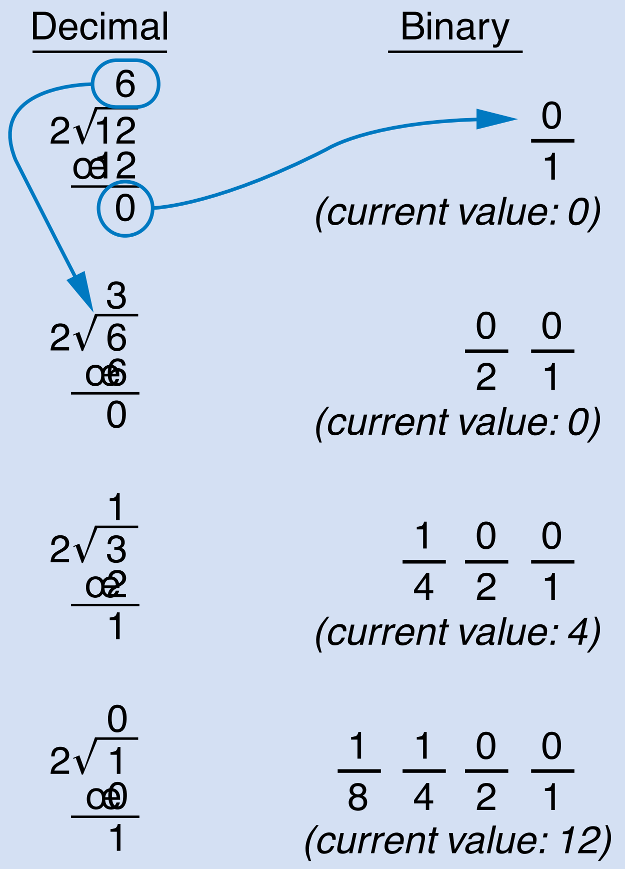
- Repeatedly divide decimal number by 2, place remainder in current binary digit (starting from 1s column)

1. Divide decimal number by 2
   Insert remainder into the binary number
   Continue since quotient (6) is greater than 0

2. Divide quotient by 2
   Insert remainder into the binary number
   Continue since quotient (3) is greater than 0

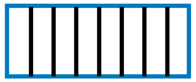
3. Divide quotient by 2
   Insert remainder into the binary number
   Continue since quotient (1) is greater than 0

4. Divide quotient by 2
   Insert remainder into the binary number
   Quotient is 0, done



Note: Works for any base N—just divide by N instead

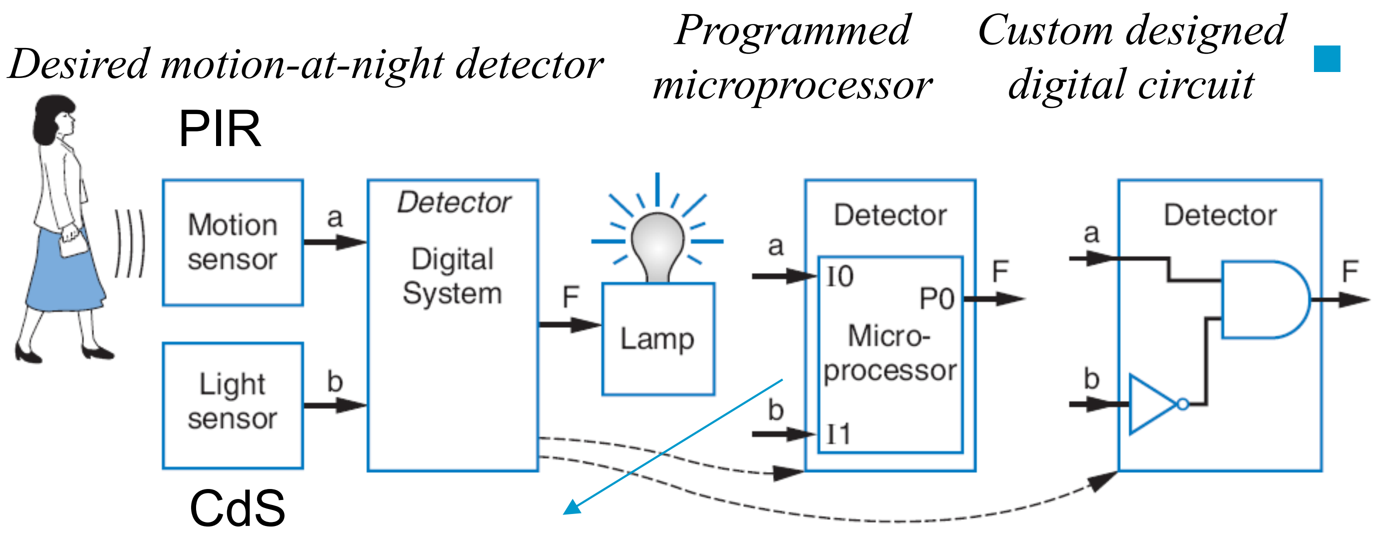
Bytes, Kilobytes, Megabytes, and More

- Byte: 8 bits 
- Common metric prefixes:
  - kilo (thousand, or 10^3), mega (million, or 10^6), giga (billion, or 10^9),
    tera (trillion, or 10^{12}), peta (10^{15}), exa (10^{18}), zettta (10^{21}), yotta (10^{24}),
    e.g., kilobyte, or KByte
- BUT, metric prefixes also commonly used inaccurately
  - 2^{16} = 6,5536 commonly written as “64 Kbyte”

2^{10} = 1,024	“1KB”	2^{11} = 2,048	“2KB”
2^{12} = 4,096	“4KB”	2^{20} = 1,048,576	“1MB”
2^{30} = 1,073,741,824	“1GB”		

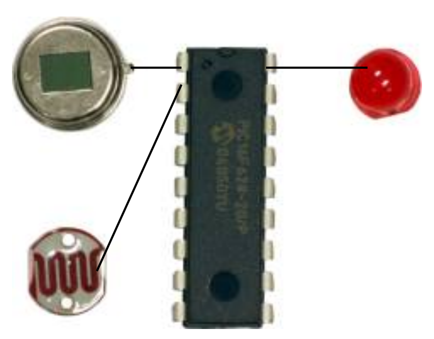
  - Typical when describing memory sizes : often powers of 2
- Also watch out for “KB” for kilobyte vs. “Kb” for kilobit

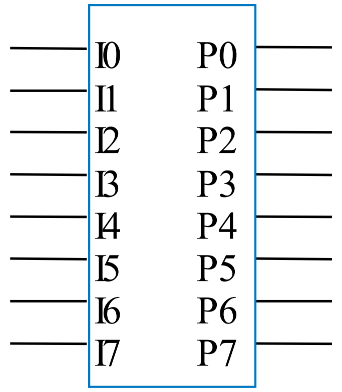
Implementing Digital Systems: Programming Microprocessors Vs. Designing Digital Circuits



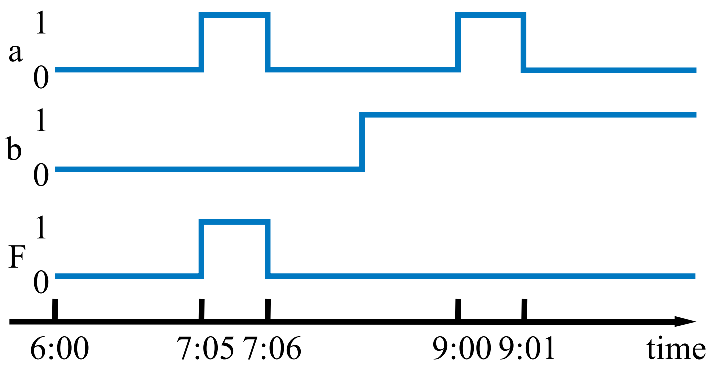
- Microprocessors a common choice to implement a digital system
  - Easy to program
  - Cheap (as low as $1)
  - Readily available

Turn on lamp (F=1) when motion sensed (a=1) and no light (b=0)





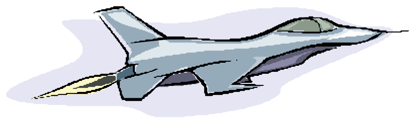
```
void main()
{
   while (1) {
      P0 = I0 && !I1;
      // F = a and !b,
   }
}
```



Timing diagram of motion-in-the-dark detector system

Digital Design: When Microprocessors Aren't Good Enough

- With microprocessors so easy, cheap, and available, why design a digital circuit?
  - Microprocessor may be too slow
  - Or too big, power hungry, or costly



Wing controller computation task:

• 50 ms on microprocessor

• 5 ms as custom digital circuit

If must execute 100 times per second:

• 100 * 50 ms = 5000 ms = 5 seconds

• 100 * 5 ms = 500 ms = 0.5 seconds

Microprocessor too slow, circuit OK.

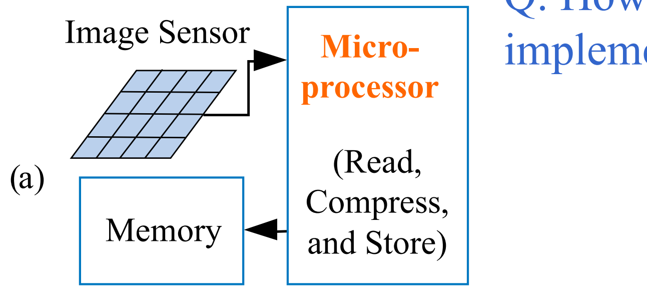
Digital Design: When Microprocessors Aren' t Good Enough

- Commonly, designers partition a system among a microprocessor and custom digital circuits

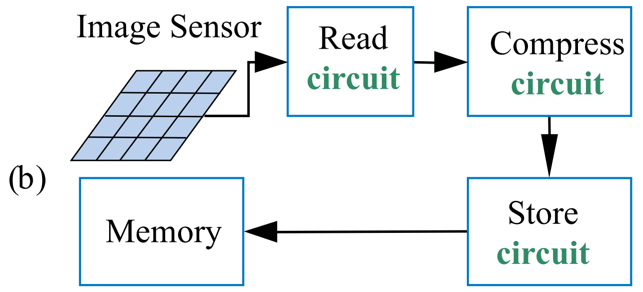
Sample digital camera task execution times (in seconds) on a microprocessor versus a digital circuit:

Task	Microprocessor	Custom Digital Circuit
Read	5	0.1
Compress	8	0.5
Store	1	0.8

Q: How long for each implementation option?

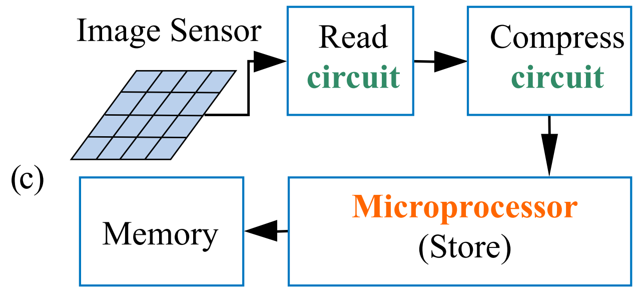


5+8+1
=14 sec



.1+.5+.8
=1.4 sec

a



.1+.5+1
=1.6 sec

Good compromise

Chapter Summary

- Digital systems surround us
  - Inside computers
  - Inside many other electronic devices (embedded systems)
- Digital systems use 0s and 1s
  - Encoding analog signals to digital can provide many benefits
    - e.g., audio—higher-quality storage/transmission, compression, etc.
  - Encoding integers as 0s and 1s: Binary numbers
- Microprocessors (themselves digital) can implement many digital systems easily and inexpensively
  - But often not good enough—need custom digital circuits