



# University of Massachusetts Amherst

## Engin112 – Lecture 5

### Number Codes and Registers

Maciej Ciesielski  
Department of Electrical and Computer Engineering  
9/16/2011

## Recap From Last Lecture

- Arithmetic with binary numbers
- Signed numbers
  - Signed Magnitude
  - Complement representation (1's complement, 2's complement)
- Two algorithms for converting a number to 2's complement
  - (Combinational / standard)  
Compute 1's complement (flip bits) and add 1 to the result.
  - (Sequential): *Leave least significant 0's and first 1 unchanged, replace 1's with 0's and 0's with 1's in all higher significant digits."*
- **IMPORTANT**: converting unsigned (positive) numbers
  - Sign extension: add one (or more) bits (0) in the MSB position before converting an unsigned number to 2's complement.
  - Example:  $101 = +5 \rightarrow 0101 = +5 \rightarrow 1010+1 = 1011 = -5$
  - Do you see why extension is needed?
- Today: codes for non-numeric information

# Binary Codes

- Most data on computers are not numbers
  - Data is still encoded with 0's and 1's
- Need codes for
  - Text characters
  - Floating point numbers
  - Images
  - Programs
  - ...
- An n-bit binary code is a group of n bits that assume up to  $2^n$  distinct combinations of 1's and 0's, with each combination representing one element of the set that is being coded.

# Binary Codes

- Example: coding of compass directions: E, W, N, S
  - 2-bit code can assume  $2^2=4$  combinations
  - Code: 00  $\leftrightarrow$  E, 01  $\leftrightarrow$  W, 10  $\leftrightarrow$  N, 11  $\leftrightarrow$  S
- Is this the only possible code?
  - Other 2-bit codes (24 possible solutions)
  - Other n-bit codes ( $n>2$ , infinitely many solutions)
  - E.g.: 0001  $\leftrightarrow$  E, 0010  $\leftrightarrow$  W, 0100  $\leftrightarrow$  N, 1000  $\leftrightarrow$  S
- Observations
  - Each element must be assigned at least one unique binary bit combination
  - Not all binary bit combinations need to have an element assigned (some codes can be unused)

# BCD Code

- Some (old) computers code numbers in decimals

- “Binary Coded Digits”
- Each 4-bit binary sequence encodes one base-10 digit

Decimal symbol	BCD digit
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

- Sanity check:

- Does each element of the coded set have a unique bit sequence assigned?
- Are there any invalid/unassigned bit sequences

- Decimal numbers are coded as multiple BCD digits

- E.g.,  $(396)_{10} = (0011\ 1001\ 0110)_{\text{BCD}}$
- How many bits would it take in binary?

# BCD Addition

- Addition is done BCD digit by BCD digit

- 4 bits at the time

- Can we use normal binary addition?

- Problem: digits adding up to more than 9
  - » Binary addition will result in invalid BCD codes
  - » 1010 ... 1111 are not valid
- Solution: check if resulting value is greater than 9
  - » if so, add 6
  - » 6 will offset the invalid BCD codes and generate the carry

## BCD Addition

- Example: 184 + 576

BCD carry	1	1		
	0001	1000	0100	184
	+0101	0111	0110	+576
Binary sum	0111	10000	1010	
Add 6	_____	0110	0110	
BCD sum	0111	0110	0000	760

## Decimal Arithmetic

- Everything needs to be 4-bit aligned
  - '+' represented by 0 (= '0000')
  - '-' represented by 9 (= '1001')
- Signed magnitude representation or complements
  - Signed magnitude hardly used
  - 10's complement most common

- Example: 375 + (-240)

- Negative numbers represented by 10's complement
- 10's complement of 240 is  $10^4 - 240 = 9760$
- Addition of all digits and discard of end carry:

$$\begin{array}{r}
 0\ 375 \\
 +9\ 760 \\
 \hline
 0\ 135
 \end{array}$$

- Sign of result automatically correct

## Other Decimal Codes

- Decimals can be encoded in many ways in 4 bits
- Weighted codes
  - Each bit position is assigned a weighting factor
  - Value of code is sum of weights where bits are '1'
  - Is BCD a weighted code?
    - » Yes! It's a 8,4,2,1 code
  - Other weighted code:
    - » 2,4,2,1 code (yields non-unique coding)
    - » 8,4,-2,-1 code
- Self-complementing codes
  - 9's complement by exchanging 1's with 0's and 0's with 1's
  - Excess-3 and 2,4,2,1 codes are self-complementing

## Other Decimal Codes

Decimal digit	BCD 8421	2421	Excess-3	8 4 -2 -1
0	0000	0000	0011	0000
1	0001	0001	0100	0111
2	0010	0010	0101	0110
3	0011	0011	0110	0101
4	0100	0100	0111	0100
5	0101	1011	1000	1011
6	0110	1100	1001	1010
7	0111	1101	1010	1001
8	1000	1110	1011	1000
9	1001	1111	1100	1111

# Gray Code

- Imaging you code a 2-bit number with two light switches connected to light bulbs
  - Can you count binary without causing ambiguity?
- Probably not:
  - Switching from 01 to 10 cannot be done simultaneously on both bits
    - » Either 01->00->10 or 01->11->10
- This brief error or ambiguity can cause problems
- Gray code changes only one bit between consecutive numbers

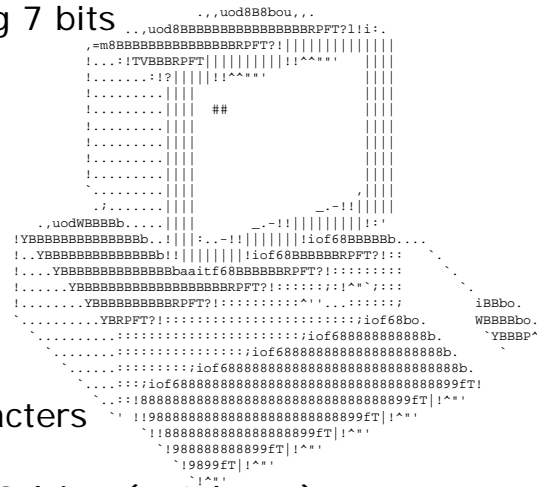
# Gray Code

Decimal digit	Gray code
0	0000
1	0001
2	0011
3	0010
4	0110
5	0111
6	0101
7	0100

Decimal digit	Gray code
8	1100
9	1101
10	1111
11	1110
12	1010
13	1011
14	1001
15	1000

# ASCII Character Code

- Most computer systems need to encode letters
- "American Standard Code for Information Interchange" (ASCII)
  - Specifies 128 characters using 7 bits
- Thanks to ASCII, text is interpreted the same on different computers
- ASCII also contains
  - Numbers
  - Control characters
  - Upper case / lower case characters
- Extended ASCII code uses 8 bits (=1 byte)



## ASCII Table

Least significant bits

Most significant bits

$b_4b_3b_2b_1$	$b_7b_6b_5$							
	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R		r
0011	ETX	DC3	#	3	C	S		s
0100	EOT	DC4	\$	4	D			
0101	ENQ	NAK	%	5	E			
0110	ACK	SYN	&	6	F			
0111	BEL	ETB	'	7	G			
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:		Z	j	z
	VT	ESC	+	;				{
	FF	FS	,	<				
	CR	GS	-	=				}
1110	SO	RS	.	>				~
1111	SI	US	/	?	O	-	o	DEL

Change from caps to lower case by changing one bit

Control characters

Numbers and characters are consecutive

## Error-Detecting Code

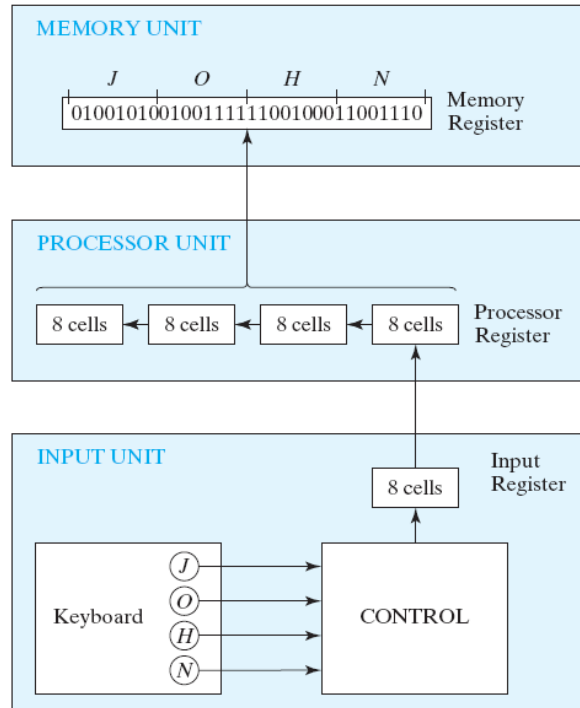
- Communication between systems can be “noisy”
  - Environmental conditions can cause bit flips
- Error-detecting codes
  - If one bit is flipped, the code becomes invalid
  - Communication system can detect that and retransmit
- “Parity bit” is added to binary data
  - “Even parity”: choose parity bit such that # of 1’s is even
  - “Odd parity”: choose parity bit such that # of 1’s is odd
- Example:
  - Data=1000001, even parity=0, odd parity=1
  - Data=1010100, even parity=1, odd parity=0
- Only one parity bit is used

## Binary Storage and Registers

- How is information stored on a digital system?
  - Bits are stored in “binary cells”
  - Binary cell can have two stable states: ‘0’ and ‘1’
- Binary cells are grouped into registers
  - n cells make up n-bit register
- Size of registers is typically predefined
  - Simple microcontroller: 8 bits = 1 byte
  - Pentium: 32 bits = 4 bytes
  - Mac G5: 64 bits = 8 bytes
- Digital system can usually process entire registers
  - “Register transfer” operation specify processing

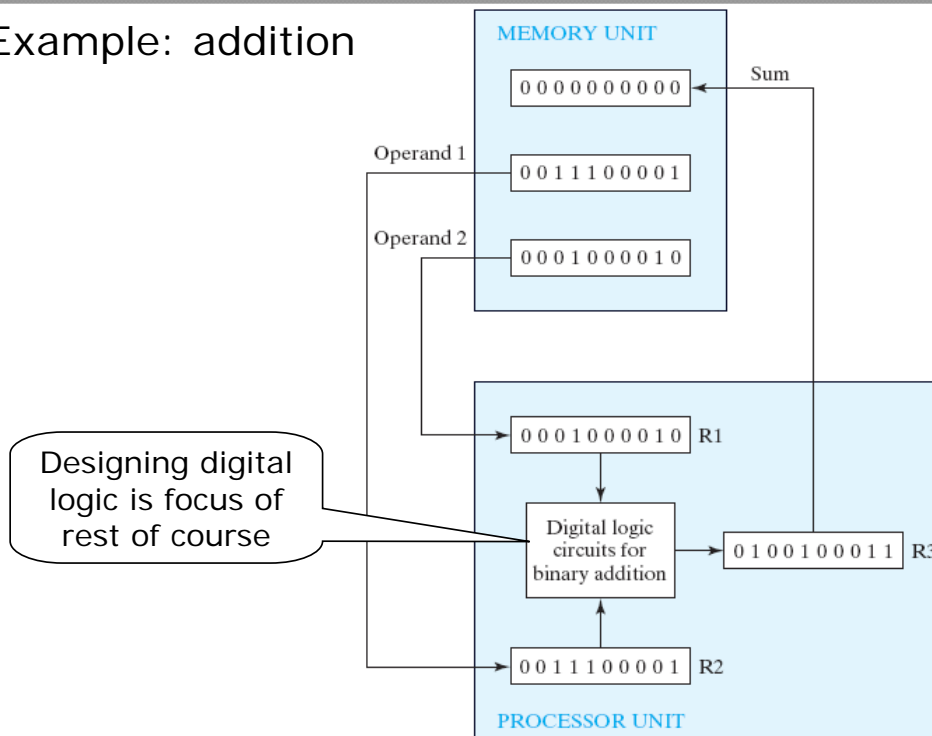
# Register Example

- Reading from keyboard:



# Register Transfer Operations

- Example: addition



Designing digital logic is focus of rest of course