Numeric Representation

Lecture Objectives:

1) Define the terms least significant bit and most significant bit.
2) Explain how unsigned integer numbers are represented in memory
3) Understand the limitations of using sign and magnitude to represent signed integer numbers
4) Explain how signed integer numbers are represented in memory using twos complement notation.
5) Convert twos complement numbers into decimal.
6) Convert decimal numbers into twos compliment format.
7) Explain the concept of sign extension
8) Convert a binary number into hexadecimal

Homework

- Read sections 2.5, 2.6, and 2.7 for the next class.

Given a binary number, convert it to an unsigned decimal format

- $1\,1011_{two}$ ← Base / binary

11? 27?

$1x2^0 + 1x2^1 + 0x2^2 + 1x2^3 + 1x2^4$

1 + 2 + 0 + 8 + 16

27

MSOE

Given a decimal number, convert it to unsigned binary

- 51ten=?two

2|51
25 R 1
12 R 1
6 R 0
3 R 0
1 R 1
0 R 1

$1 \times 2^0 + 1 \times 2^1 + 0 \times 2^2 + 0 * 2^3 + 1 \times 2^4 + 1 \times 2^5$

$1 + 2 + 0 + 0 + 16 + 32$

51

Signed Numbers

- First approach
  - Add a sign bit to the number
  - 0 indicates positive
  - 1 indicates negative
- Example
  - $-11_{ten}=?_{two}$

Sign bit

1 1011

2 | 11
5 R 1
2 R 1
1 R 0
0 R 1

- 1 00000000 == 0 00000000

Problem with sign and magnitude

Two Zeros

Negative Zero

Positive Zero

Solution: 2's compliment numbers

- Leading 0's mean positive, leading 1's mean negative
- Range: -2^{n-1} to $+2^{n-1}-1$
- Example
  - $1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1100_2$
    $= -1\times2^{31} + 1\times2^{30} + \ldots + 1\times2^{2} + 0\times2^{1} + 0\times2^{0}$
    $= -2{,}147{,}483{,}648 + 2{,}147{,}483{,}644 = -4_{10}$

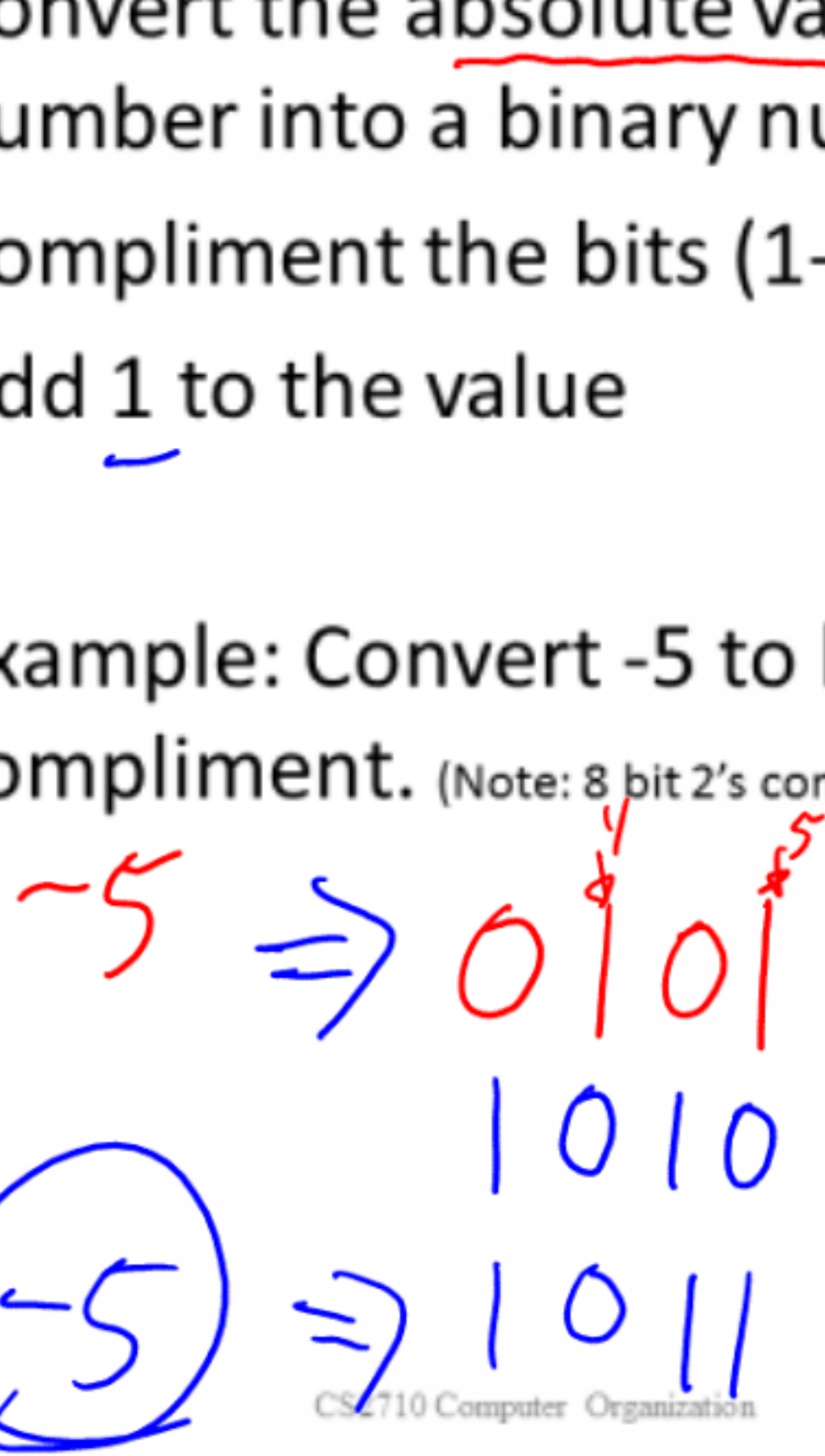
$$x = -x_{n-1}2^{n-1} + x_{n-2}2^{n-2} + \cdots + x_1 2^1 + x_0 2^0$$

- Using 32 bits
  - $-2{,}147{,}483{,}648$ to $+2{,}147{,}483{,}647$

Converting a number into 2's compliment

- Convert the absolute value of the number into a binary number
- Compliment the bits (1-> 0, 0->1)
- Add 1 to the value

- Example: Convert -5 to binary using 2's compliment. (Note: 8 bit 2's compliment number)

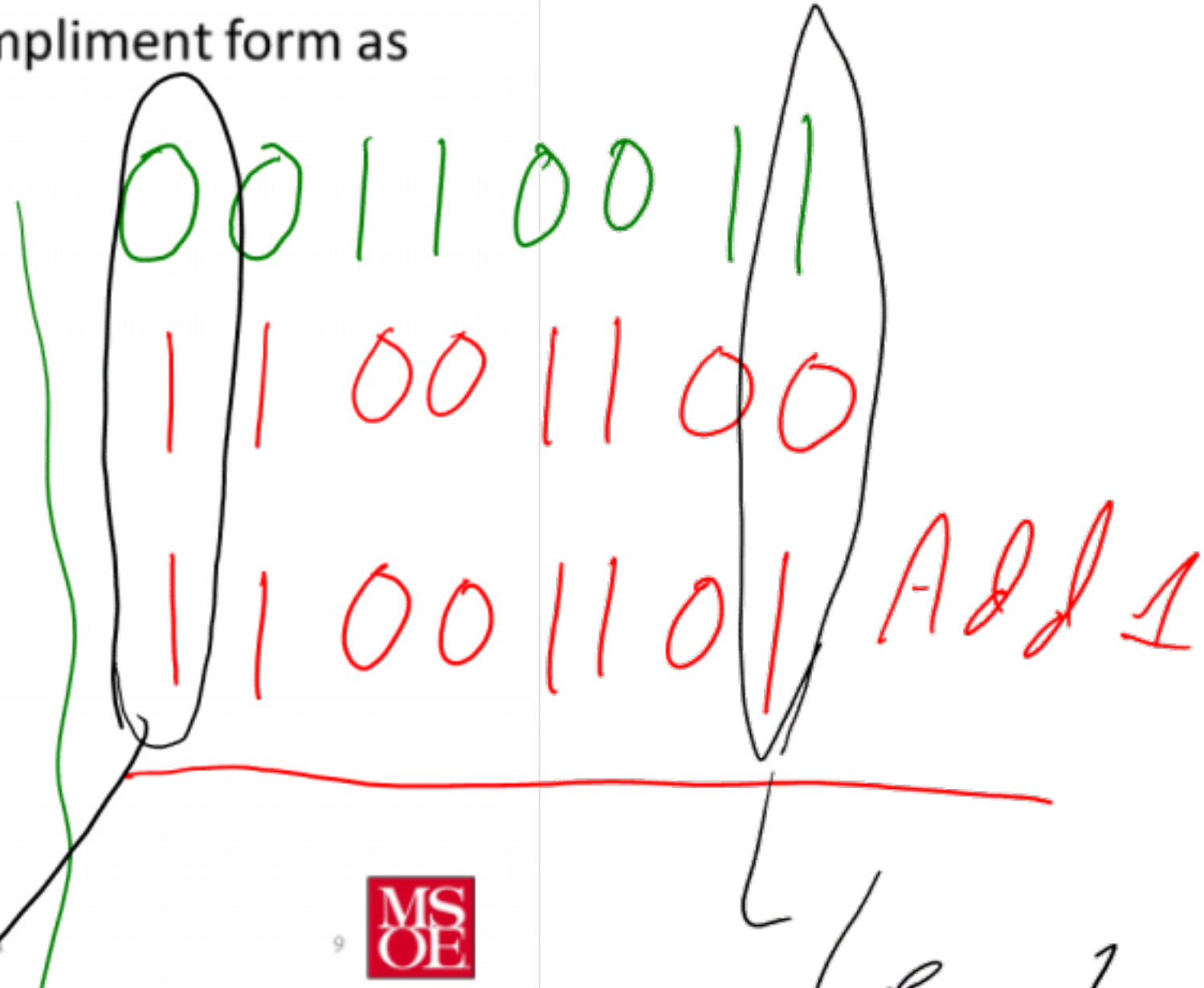


Example: Represent

- Convert -51 to 2's compliment form as an 8 bit number

|51| => 2|51

25 R 1
12 R 1
6 R 0
3 R 0
1 R 1
0 R 1



Most significant bit

Least significant bit

-51 as an 8 bit #

Sign extension

- Representing a number using more bits
  - Preserve the numeric value
- Replicate the sign bit to the left
  - c.f. unsigned values: extend with 0s
- Examples: 8-bit to 16-bit
  - +2: 0000 0010 => 0000 0000 0000 0010
  - –2: 1111 1110 => 1111 1111 1111 1110

Blue is extended bits

All the same.

- Convert -51 to a 16 bit signed binary representation based on the 2's complement value from before

Example

00110011 → 51

11001100

11001101 ← -51

Copy 7, make 16 bit #

1111111111001101

Converting binary to hex

- Hexadecimal – Base 16
  - Group binary digits into sets of 4
  - Convert each group into a hex digit

0000	0	1000	8
0001	1	1001	9
0010	2	1010	A
0011	3	1011	B
0100	4	1100	C
0101	5	1101	D
0110	6	1110	E
0111	7	1111	F

Binary Values

Hex Values

- Convert the number 10100101 to hex

0x A 5

Example

- Convert the number 1100100 to hex
  - What decimal number is this?

0 1 1 0 0 1 0 0

4 2 1 | 4 2 1

0x 6 4

Sign Extend w/ flex

⇒ Add either 0's

OR F's

0x81 ⇒ 8 bit signed

0xFF81

0x7F

↓

0x007F

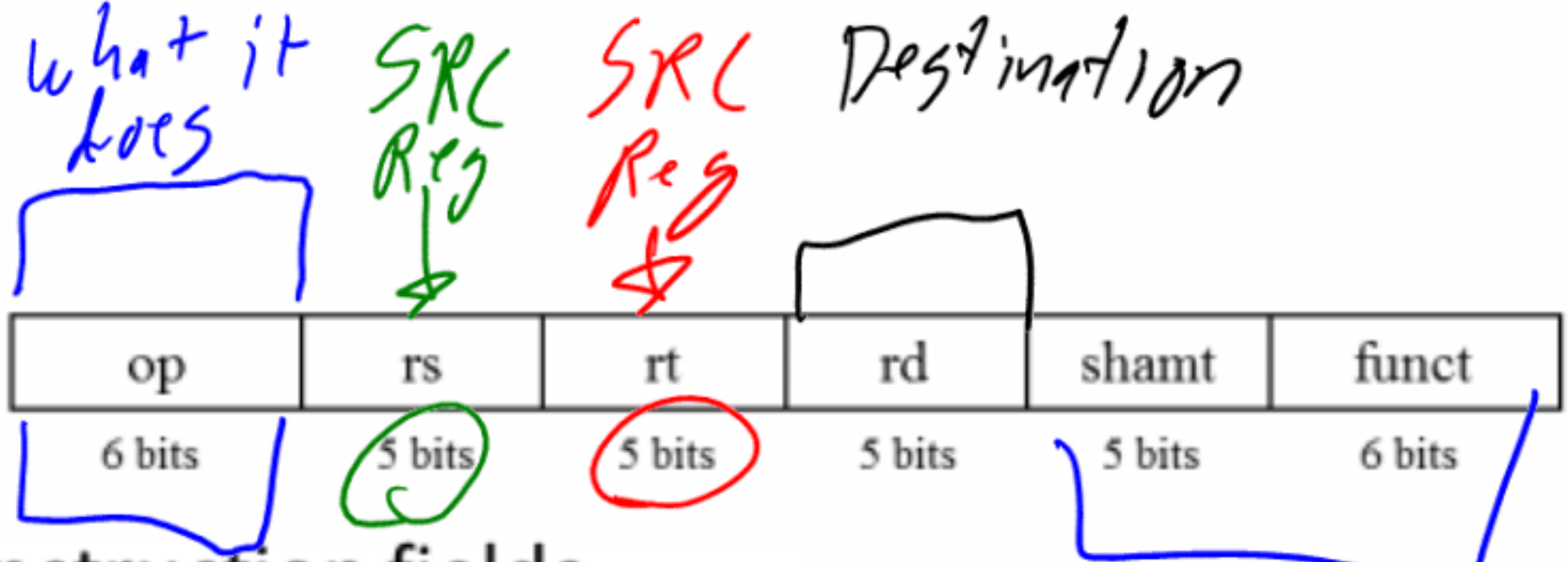
Sign Extension

- Representing a number using more bits
  - Preserve the numeric value
- In MIPS instruction set
  - `addi`: extend immediate value
  - `lb`, `lh`: extend loaded byte/halfword
  - `beq`, `bne`: extend the displacement
- Replicate the sign bit to the left
  - c.f. unsigned values: extend with 0s
- Examples: 8-bit to 16-bit
  - +2: 0000 0010 => 0000 0000 0000 0010
  - –2: 1111 1110 => 1111 1111 1111 1110

Representing Instructions

- Instructions are encoded in binary
  - Called machine code
- MIPS instructions
  - Encoded as 32-bit instruction words
  - Small number of formats encoding operation code (opcode), register numbers, …
  - Regularity!
- Register numbers
  - $t0 – $t7 are reg's 8 – 15
  - $t8 – $t9 are reg's 24 – 25
  - $s0 – $s7 are reg's 16 – 23

MIPS R-format Instructions



- Instruction fields
  - op: operation code (opcode)
  - rs: first source register number
  - rt: second source register number
  - rd: destination register number
  - shamt: shift amount (00000 for now)
  - funct: function code (extends opcode)

One type of mips instruction