EE 109 Midterm Review
Number Systems

• Computer use base 2 (binary) – 0 and 1
• Humans use base 10 (decimal) – 0 to 9
• Humans using computers:
  – Base 16 (hexadecimal) – 0 to 15 (0 to 9,A,B,C,D,E,F)
  – Base 8 (octal) – 0 to 7
• Value of a number is calculated by the summing the coefficients (digits) times the radix raised to a power.
  
  \[ 326_{10} = 3 \times 10^2 + 2 \times 10^1 + 6 \times 10^0 \]
  
  \[ 11010_2 = 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 26_{10} \]
  
  \[ C7_{16} = 12 \times 16^1 + 7 \times 16^0 = 199_{10} \]

• Converting between hex and binary:
  – 2->16: Separate bits into groups of 4 and write equivalent hex digit
  – 16->2: Write out bits for each hex digit.
Number Systems

- Converting decimal to (unsigned) binary:
  - Start with largest power of 2 less than the decimal number
  - Coefficient for that power will be a one
  - Subtract that power of 2 from the decimal number
  - Repeat

- Example: $41_{10}$ to base 2 – Find powers of 2 that add up to 41 starting with larger ones first (like making change)
  - $41 - 32 = 9$
  - $9 - 8 = 1$
  - $1 - 1 = 0$
  - $41_{10} = 32 + 8 + 1$
  - $= 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$
  - $= 101001_2$
Unsigned Numbers

- Used to represent only positive numbers
  - In $n$ bits can represent numbers from 0 to $+2^n - 1$
  - Example: 8 bits can range from 0 to 255
Signed Numbers

• Have to represent negative numbers by using 1’s and 0’s (no ‘–’ sign)
• Several methods have been used
• Sign-magnitude:
  – Use the leading bit to represent the sign of the number (0=+, 1=–)
    • 0101 = +5
    • 1101 = –5
  – In ‘n’ bits can represent numbers from –(2^{n-1}-1) to +2^{n-1}-1
  – Problems:
    • Two versions of zero (0000 and 1000)
    • Difficult to use with hardware. Have to make tests to see how to do adds and subtracts.
Signed Numbers

• 2’s complement:
  – Negative numbers are the “2’s complement” of the positive number.
  – MSB has a value of \(-2^{n-1}\)
  – \(0101 = +5\)  \(1011 = -5\) \((-8 + 2 + 1 = -5)\)
  – In ‘n’ bits can represent numbers from \(-2^{n-1}\) to \(+2^{n-1}-1\)
  – Remember: A 2’s complement number is not necessarily a negative number.
  – 2’s complement is way of representing a range of positive AND negative numbers.

• Advantages
  – No problem with two zeros.
  – Easier to use with hardware.
Practice

• Show in hex the representation of:
  – char x = -92;
  
  – short int x = -2;

  – unsigned int x = 512;
Binary Codes

• Binary values can be defined to represent any number of different things.
  – ASCII character set
    • 128 characters (7-bits)
  – Unicode – lots of characters
  – Instructions (ADD, SUB, etc.)
  – Conditions (Error, ready, out of paper, etc.)
  – Defined by user.
Logical Operations

• All computers circuits are based on a small number of logical elements.
  – AND – output is true if ALL inputs are true
  – OR – output is true if ANY input is true
  – NOT – output is the logical complement of the input
• By combining many of these logical elements we can build any complex circuit we want.

• Combinational Logic:
  – The output of a circuit only depends on the CURRENT inputs
  – No memory of past inputs or outputs

• Sequential Logic:
  – The output of a circuit depends on the current inputs and past states and inputs to the circuit.
  – Circuit has memory.
Practice

- Find the output values of F and G for inputs:
  - ABC=011
  - ABC=001
  - ABC=110
Registers

• [Note: we did not spend too much time on this...but we'll just review it briefly]

• The fundamental building block of sequential circuits is the “flip-flop” which stores one bit.
  – When clocked, the flip-flop stores the current input bit.
  – The flip-flop always outputs whatever bit is stored in it.
  – Most common is the “Edge-triggered D flip-flop”
  – Bit is stored on the transition (edge) of the clock signal.

• Register = collection of flip-flops controlled by the same clock signal.
  – All the elements of a register store their input bits at the same time.
Electrical Circuits

• Current – amount of charge flowing through a point
  – Measured in Amps
  – Denoted by “I”

• Voltage – potential energy that can cause charge to move.
  – Measured in Volts between two points
  – Denoted by “V”

• Ground – a point of reference for measuring voltages
  – Usually connected to the negative side of the power source.
Electrical Circuits

• Kirchoff’s Current Law
  – Amount of current flowing into a point must equal the amount flowing out of the same point.

• Kirchoff’s Voltage Law
  – Sum of the voltages around a complete loop must equal zero.
Electrical Circuits

- **Resistance**
  - Makes it harder for current to flow, must use more voltage to cause charge to move.
  - Ohm’s Law: $V = IR$

- **Resistors in series**
  - Equivalent resistance is the sum of the individual resistances
    \[ R_{total} = R_1 + R_2 + \cdots + R_n \]

- **Resistors in parallel**
  - Equivalent resistance is reciprocal of the sum of the individual reciprocals of the resistances
    \[ \frac{1}{R_{total}} = \frac{1}{R_1} + \frac{1}{R_2} + \cdots + \frac{1}{R_n} \]

- A resistor of 0 ohms = a wire
- A resistor of inf. ohms = an open circuit (i.e. no connection)
Practice

• Find the current $i_1$ in terms of $V_s$ and the resistors
• Find the voltage $V_2$ in terms of $V_s$ and the other resistors
• If $R_3=0$ ohms what's the voltage $V_2$?
Arduino Uno

• Based on Atmel ATmega328P microcontroller
• 20 pins that can be used by various modules
• Three I/O ports are each controlled by three registers
  – PORT – output data, also controls pull-up resistor when an input
  – PIN – input data
  – DDR – data direction register
• Analog-to-Digital Converter
  – 6 input lines, only one can be used at a time
  – Converter clock rate controlled by a prescaler
  – Must have a voltage reference to operate
Arduino Modules

• Know generally how the Arduino modules and external devices we've used in lab work and how they need to be programmed at a high level (Digital I/O, ADC, Interrupts, LCD)
  – You don't need to memorize bit names or locations of each register controlling the module
  – You should know the ideas of what some of those bits mean (i.e. what a prescalar does, what an interrupt enable bit does, how to turn on a pull-up resistor, what a DDR register indicates)
Bit Masking

- Bit masking is useful when some of the bits of a variable (e.g., a register, or a port) need to remain unchanged while modifying the rest.


- If the current value of B is known, we may assign the updated value directly to B:
  - Assume: B = 0x75 = 0b01110101
  - We may then simply reassign with the updated value, i.e., B = 0b01111101

- What if we did not know the current value of the variable?

  - B = 0bXXXXXXXX Note: X means the bit value is unknown
  - In that case we can XOR B with a mask that exposes B[3] for a flip:
    - B = B ^ 0b00001000 = 0bXXXXXXXX ^ 0b00001000 = 0bXXXXX¯X XXX
      Note: ¯X denotes the flipped bit (still unknown, but flipped)
  - We may apply a shift operation to first calculate the mask:
    - 0b00000001 << 3 which produces: 0b00001000
  - Therefore: B = B ^ (0b00000001 << 3) or:
    - B = B ^ (1 << 3) or writing the short version:
      - B ^ = 1 << 3
Arduino Uno

• Registers can be read/written as a full byte
  – DDRD = 0x67;
  – y = ADCH;

• Reading a bit from an register
  x = PINB & (1 << PINB3);

• Writing a bit to an register;
  ADCSRA |= (1 << ADSC);
  PORTC &= ~(1 << PC2);

• Inserting multiple bits into a register
  Example: write low 4 bits of x into PORTB, don’t change high 4 bits in PORTB
  PORTB &= 0xf0; // zero out the lower 4-bits of PORTB
  PORTB |= (x & 0x0f); // clear the upper 4-bits of x
  // then OR into PORTB
Practice

• Write code to do the following tasks:
  – Turn on the top 4 bits of PORTB w/o affecting the lower 4 bits
  – Wait until the lower 2 bits of PIND are both 1 at the same time then continue in the program
  – Clear the LSB of DDRC
  – Copy all the bits of PORTB into PORTD
Practice

Binary Arithmetic

• Add column by column from right to left
• Subtraction converts to addition: A + ~B + 1
• Check if unsigned and signed overflow occurs

\[
\begin{align*}
11100111 & \quad + \quad 10001101 & \quad 01010011 \\
+ \quad 10001101 & \quad - \quad 11001001
\end{align*}
\]