Wrapping up …

• **Terminology**
  - tera ($10^{12}$), giga ($10^9$), mega ($10^6$), kilo ($10^3$): Hz, Byte
  - 1 byte = 8 bits
  - For a 2.5 GHz CPU and 600 MHz bus, how much faster is the CPU than the bus?
    - 2.5 GHz = 2500 MHz, so ratio is 2500/600 about 4 times faster

• **Binary / Decimal**
  - conversion between them
  - 2’ complement

<table>
<thead>
<tr>
<th>-2^7</th>
<th>-2^6</th>
<th>-2^5</th>
<th>-2^4</th>
<th>-2^3</th>
<th>-2^2</th>
<th>-2^1</th>
<th>-2^0</th>
</tr>
</thead>
</table>

- To rep 1000 in 2’s complement needs how many bits?
  - range rep by N bits in 2’s complement is $-2^{(N-1)} \sim 2^{(N-1)} - 1$
  - 10 bits $\sim$ 511, 11 bits $\sim$ 1023, so 11 bits
- How to use 1 byte to rep -10 in 2’s complement?
  - 10 $\sim$ 1010 $\sim$ 0000 1010
  - 0000 1010 invert 1111 0101 +1 = 1111 0110

• **Digital logic**
  - gates (AND, OR, NOT, NAND, NOR, XOR), their uses and implementations
  - combinational circuits, sequential circuits: MUX, DEMUX, Encoder, Decoder, Adder, ALU, flipflops, parallel registers, shift registers
    - difference between combinational circuits and sequential circuits
      - The output of a combinational logic circuit depends solely on its inputs. It is memory less.
      - The output of a sequential logic circuit depends on inputs and the current state. It contains memory.
    - circuit design (k-map)
      - truth table from question specification, k-map (00, 01, 11, 10), sum of produce
      - practice from HW1 and Q1
  - circuit equivalent (notes)

• **Memory**
  - memory hierarchy (registers, cache, mm)
  - cache structure (mapping functions: direct, associative, set associative; addressable unit: word, half word, byte)
    - HW5, Q5
  - average access-time, hit-rate
    - hit rate: the ratio that the accessed word is found in the faster memory
    - The hit rate of L1 cache is 95% and its access time is 1 ns, the access time of main memory is 10 ns. What is the average access time.
      - $0.95 \times 1 + 0.05 \times (1 + 10) = 0.95 + 0.55 = 1.5$ ns
  - ISA
    - stack architectures
      - both operands of ALU popped from the top of the stack, result of ALU push into the top of the stack
- addressing modes (immediate, direct, indirect, register direct, register indirect, displacement)

- Assembly language
  - the benefits of assembly language
  - two-pass assembler
  - call stack (layout, how it works)
  - assembly program (Examples on the notes)

- Pipeline and superscalar
  - dependencies (true data, WAW, WAR), possible way to address dependencies (register renaming)
  - instruction issue policies (out-of-order/in-order issue/completion)