Review for Final Exam

- Numbers
  - Decimal to Hex (signed and unsigned)
  - Hex to Decimal (signed and unsigned)
  - Binary to Hex
  - Hex to Binary
  - Addition and subtraction of fixed-length hex numbers
  - Overflow, Carry, Zero, Negative bits of CCR
- Programming Model
  - Internal registers – A, B, (D = AB), X, Y, SP, PC, CCR
- Addressing Modes and Effective Addresses
  - INH, IMM, DIR, EXT, REL, IDX (Not Indexed Indirect)
  - How to determine effective address
- Instructions
  - What they do - Purple book
  - What machine code is generated
  - How many cycles to execute
  - Effect on CCR
  - Branch instructions – which to use with signed and which with unsigned
- Machine Code
  - Reverse Assembly
- Stack and Stack Pointer
  - What happens to stack and SP for instructions (e.g., PSHX, JSR)
  - What happens to stack and SP for interrupt
  - What happens to stack and SP when program leaves an interrupt service routine

- Assembly Language
  - Be able to read and write simple assembly language program
  - Know basic psuedo-ops – e.g., equ, dc.b, ds.w
  - Flow charts
- C Programming
  - Setting and clearing bits in registers
    - `PORTA = PORTA | 0x02;`
    - `PORTA = PORTA & ~0x0C;`
  - Using pointers to access specific memory location or port.
    - `* (unsigned char *) 0x0400 = 0xaa;`
    - ```
      #define PORTX (* (unsigned char *) 0x400)
      PORTX = 0xaa;
      ```
- Interrupts
  - Interrupt Vectors (and reset vector)
    - How to set interrupt vector in C
  - How do you enable interrupts (specific mask and general mask)
  - What happens to stack when you receive an enabled interrupt
  - What happens when you leave ISR with RTI instruction?
  - What setup do you need to do before enabling interrupts?
  - What do you need to do in interrupt service routine (clear source of interrupt, exit with RTI instruction)?
  - How long (approximately) does it take to service an interrupt?
      ```

- Timer/Counter Subsystem
  - Enable Timer
  - Timer Prescaler
    - How to set
    - How it affects frequency of timer clock
  - Timer Overflow Interrupt
  - Input Capture
  - Output Compare
  - How to enable interrupts in the timer subsystem
  - How to clear flags in the timer subsystem
  - Be able to look at timer registers and determine how timer is set up
    - Which channels are being used
    - Which are being used for Input Capture, which for Output Compare
  - How to time differences from Timer counts
- Real Time Interrupt
  - How to enable
  - How to change rate
  - How to enable interrupt
  - How to clear flag
- Pulse Width Modulation
  - How to get into 8-bit, left-aligned high-polarity mode
  - Calculate how many clock periods it takes to get desired PWM period (frequency)
  - How to set PWM period (frequency)
    - Using Clock Mode 0
    - Using Clock Mode 1
  - How to set PWM duty cycle
  - How to enable PWM channel
  - Be able to look at PWM registers and determine PWM frequency and duty cycle

- SPI
  - Pins used – SCLK, MOSI, MISO, SS
  - Make output pins output with DDRS
  - Difference of use in Master and Slave mode
  - SP0CR1 Register
    - Enable SPI
    - Master or Slave
    - Enable interrupts
    - Clock polarity
    - Clock phase
    - Automatically operate SS for single-byte transfers
    - LSB or MSB first
  - SP0CR2 Register — always 0 (normal mode)
  - SP0BR Register — Set speed (master only)
  - SP0SR Register — SPIF flag - clear by reading SP0SR, the access (read or write) SP0DR
  - SP0DR Register — shift register – master starts transfer by writing data to SP0DR

- A/D Converter
  - How to power-up A/D converter (ATDCTL2)
  - Write 0x01 to ATDCTL4 to set at fastest conversion speed and 8-bit conversions
  - Write 0x81 to ATDCTL4 to set at fastest conversion speed and 10-bit conversions
  - How to set modes of A/D converter (ATDCTL5)
    - 8-channel scan vs. 4-channel scan
    - Continuous Scan vs. Single Scan
    - Multichannel vs. Single Channel conversions
  - How to tell when conversion is complete - ATDSTAT register
  - How to read results of A/D conversions - $\mathrm{ADR}[0-7]\mathrm{H}$ (8-bit conversions)
  - How to read results of A/D conversions - $\mathrm{ADR}[0-7]$ (10-bit conversions)
    - Be able to convert from digital number to voltage, and from voltage to digital number (need to know V_{RH} and V_{RL}).

- The HC12 in Expanded Mode
  - Getting into expanded mode — MODA, MODB, BKGD pins or MODE Register
  - PEAR Register — enable ECLK, LSTRB, R/W on external pins
  - Ports A and B in expanded mode
    - Port A – AD 15-8 (Port A is for data for high byte, even addresses)
    - Port B – AD 7-0 (Port B is for data for low byte, odd addresses)
  - E clock
    - Address on AD 15-0 when E low, Data on AD 15-0 when E high
    - Need to latch address on rising edge of E clock
    - On write (output), external device latches data on signal initiated by falling edge of E
    - On read (input), HC12 latches data on falling edge of E
    - E-clock stretch - MISC register
  - R/W Line
  - LSTRB line
  - Single-byte and two-byte accesses
    - 16-bit access of even address – A0 low, LSTRB low – accesses even and odd bytes
    - 8-bit access of even address – A0 low, LSTRB high – accesses even byte only
    - 8-bit access of odd address – A0 high, LSTRB low – accesses odd byte only
  - Timing – Be sure to meet setup and hold times of device receiving data
    - For a write, meet setup and hold of external device
    - For a read, meet setup and hold of HC12
  - Timing — Be sure to meet address access time (length of time address needs to be on bus before external device is ready)