ECE 331 Lesson Objectives Spring 2013

Midterm

Objectives: Students should be able to:

- Perform number base conversions for Dec, Hex, Bin
- Identify value range as function of number of bits; identify out of range overflow for signed and unsigned binary numbers
- Express numbers in signed 2's complement (S2C) form, perform 2's complement operation, and evaluate subtraction using S2C.
- Identify value range in S2C and determine 2C overflow
- Perform minimization of logic expressions using minimax terms, K-maps, and Boolean arithmetic
- Identity gate symbol and truth table for basic logic gates
- Describe operation of tri-state, mux, decoder
- Identity active low vs. active high logic
- Describe operation of flip flops
- Explain operation of sequential logic circuits including shift registers and counters
- Utilize DeMorgan's relations to implement logic circuits using only NAND (or NOR) gates
- Define important events/times in computer history
- Define and differentiate microprocessor, microcontroller, embedded system
- Describe and identify components in general computer architecture
- Draw and label general computer architecture
- Draw and label connections of CPU components
- Evaluate address and data bus size (# bits/signals) for a given memory size
- Identify architectural components of HC12/S12 block diagrams
- Identify components in programmers model
- Identify and describe main flags in the condition code register (CCR)
- Determine which CCR flag results from specific arithmetic operations
- Identify which CCR flags can change for each ASM instruction
- Identify instruction information from HC12 Instruction table
- Describe assembly (ASM) language instruction format
- Describe instruction execution cycle
- Identify the six main groupings of ASM instructions
- Describe operation/function of primary ASM instructions
- Prepare and use mask bytes in instruction like BCLR/BSET
- Identify and list address modes for HC12 instruction set
- Describe and write ASM code using inherent and immediate address modes
- Describe and write ASM code using direct and extended address modes
- Use ASM Instruction Chart to map results of ASM instructions
- List and identify ASM directives
- Use ASM simulator program to test and debug HC12 ASM code
- Write short ASM instruction blocks to achieve specific program tasks
- Explain simple ASM code and .lst output files
- Describe the steps in the programming process
- Prepare psuedocode and flowcharts to describe an algorithm
- Identify and code looping constructs
- Describe techniques of structured programming and their implementations in ASM code.
- Describe and write ASM code using indexed and relative address modes
- Describe the branch concept and branching instructions
- Implement (in ASM code) conditional operations using branch instructions
- Calculate relative offset (# bytes) for branch instructions
- Describe the steps in the assembly process
- Identify address, data, and program information within .LST and .S19 assembly output files
- Differentiate between data and program bytes stored in memory

- Calculate number of clock cycles and instruction time for blocks of ASM code.
- Write ASM loop constructs with specific delay times
- Define 'peripheral hardware' and identify key peripheral blocks on the case study microcontroller
- Explain how data memory, program memory, configuration registers and I/O devices are mapped to addresses in stored-program computer organization
- Describe I/O addressing modes for peripheral hardware (memory mapped vs. isolated)
- Read and write microcontroller bi-directional digital I/O ports using ASM code

Final

Objectives: Students should be able to:

- Identify and describe different types of memory (SRAM, DRAM, ROM, EEPROM)
- Describe memory array structures and interfacing requirements
- Describe microcontroller operating (addressing) modes
- Describe extended memory I/O bus signals and functions
- Design interface to an external memory array
- Explain the structure and operation of the stack (FILO) and stack pointer
- Utilize ASM instructions to control stack operations
- Describe subroutines and explain the difference between a branch loop and a subroutine
- Write ASM code using subroutines and track stack values through subroutine process
- Identify proper subroutine programming practices including parameter passing
- Track values in stack pointer through ASM subroutine calls and PSH/PUL instructions
- Describe exceptions in terms of microcontrollers
- Explain internal and external reset mechanisms in the HC12
- Describe the hardware and software interrupts of the Freescale HC12/S12 controller
- Identify the requirements for interrupt service routines
- Explain interrupt priority and describe priorities for internal/external resets and interrupts
- Calculate count time for timer hardware using prescale factors
- Describe and write functional control code for hardware timer peripheral blocks
- Describe the operation of the free running timer system and the function of timer registers on the HC12
- Write ASM code to create time delays using the HC12 free running timer
- Describe the input capture, output compare, and pulse accumulator functions of the timer hardware
- Construct hardware/software systems integrating timer, interrupt, and memory systems
- Explain basic sampling concepts including sampling rate, resolution, etc.
- Describe the operation of A/D converters
- Calculate A/D digital results from analog values and vice versa
- Describe and contrast different serial communication interfaces
- Describe operation of UART, SPI, and I^2C communication interfaces
- Describe the shared "open collector" (wired-OR) bus concept
- Describe how Cadence Virtuoso and Analog Development Environment are used in digital circuit/system design
- Describe similarities and differences between the HC12 and ARM microcontrollers