Course Number and Title: ELC 275 Microprocessor Systems

Campus Location:
Georgetown, Dover, Stanton

Effective Date:
2020-53

Prerequisite:
ELC 265, ELC 266 or concurrent

Co-Requisites:
none

Course Credits and Hours:
4.00 credits
3.00 lecture hours/week
4.00 lab hours/week

Course Description:
This course introduces microprocessors as embedded devices. Emphasis is on Input/Output techniques, interrupts, real-time operation, high-level code debugging and interfacing to various types of sensors and actuators. Projects that address various embedded applications are a major part of the course.

Required Text(s):
Obtain current textbook information by viewing the campus bookstore - https://www.dtcc.edu/bookstores online or visit a campus bookstore. Check your course schedule for the course number and section.

Additional Materials:
None

Schedule Type:
Classroom Course

Disclaimer:
None

Core Course Performance Objectives (CCPOs):

1. Describe the characteristics and application of embedded systems. (CCC 1, 2, 3, 5, 6; PGC 1, 2, 3, 4, 5)
2. Explain the function of a state machine versus a datapath. (CCC 1, 2, 3, 6; PGC 1, 2, 3, 4)
3. Explain processor architecture. (CCC 1, 2, 3, 6; PGC 1, 2, 3, 4)
4. Describe memory, bus, and input/output (IO) systems. (CCC 1, 2, 3, 6; PGC 1, 2, 3, 4)
5. Formulate assembly language into usable instruction. (CCC 1, 2, 3, 6; PGC 1, 2, 3, 4)
6. Discuss interrupts and their application to a microprocessor. (CCC 1, 2, 3, 6; PGC 1, 2, 3, 4)
7. Explain embedded system design concepts. (CCC 1, 2, 3, 4, 5, 6; PGC 1, 2, 3, 4, 5, 6)
8. Develop embedded system applications with the C programming language. (CCC 1, 2, 3, 4, 6; PGC 1, 2, 3, 4, 6)

See Core Curriculum Competencies and Program Graduate Competencies at the end of the syllabus. CCPOs are linked to every competency they develop.

Measurable Performance Objectives (MPOs):
Upon completion of this course, the student will:

1. Describe the characteristics and application of embedded systems.
    1. Explain how the term “embedded” has multiple implications with descriptions like firmware, dedicated purpose, and invisible components.
    2. Explain the difference between the term “systems” when combined with the term embedded as opposed to the more common use of systems when referring to personal computers (PCs), minicomputers or mainframes.
    3. Explain how a general purpose computer is comprised of numerous embedded systems but as a whole is not an embedded system.
    4. Describe the characteristics of embedded systems in terms of their constituent hardware and software components, timeliness, interconnectedness, reliability, and marketplace pervasiveness.
    5. Describe a real-time operating system (RTOS), and explain the meanings of hard, firm (strict), and soft real-time constraints.
    6. Describe specific examples of applications of embedded systems.
2. Explain the function of a state machine versus a datapath.
    1. Diagram the input, control, and output of a clocked sequential logic circuit.

2. Diagram an algorithmic state machine (ASM) comprised of a finite state machine (FSM) and a datapath unit.
3. Describe important FSM characteristics and datapath characteristics.
4. Describe and flowchart the key design considerations of an ASM system.
5. Describe applications of ASM design, and identify the FSM and datapath components of the applications.
6. Explain the differences and similarities between the Mealy machine and the Moore machine.

3. Explain processor architecture.
    1. Describe and diagram the classic components of computer and microcontroller architectures.
    2. Describe the differences among von Neumann, Harvard, and modified Harvard architectures.
    3. Describe the difference between the reduced instruction set computer (RISC) architecture and the complex instruction set computer (CISC) architecture.
    4. Describe the dominant core architectures implemented by the majority of the embedded (also referred to as low-end) system market.
    5. Discuss the terms advanced RISC machines (ARM), microprocessor without interlocked pipeline stages (MIPS), and advanced virtual RISC (AVR).
    6. Discuss the microprocessor's core registers (also called the central processing unit's [CPU] programming model), and describe their functions.
    7. Identify the major elements of instruction set architecture (ISA) (registers, memory, word size, endianness, instructions, and addressing modes), and explain the purpose of the ISA.
4. Describe memory, bus, and input/output (IO) systems.
    1. Define bits, nibbles, bytes, words, cache, main memory, and secondary memory.
    2. Identify and explain read-only memory (ROM), static random access memory (SRAM), dynamic random access memory (DRAM), the various types of flash memory, and magnetic memory.
    3. Describe volatile memory and non-volatile memory, and categorize memory systems as volatile or non-volatile.
    4. Explain the processor - memory performance gap.
    5. Describe the approximate cost per gigabyte and access speed for SRAM, DRAM, and hard drives.
    6. Discuss where SRAM, DRAM, and hard drives fit into the memory hierarchy, and explain the principle of locality.
    7. Calculate memory performance in terms of average memory access time (AMAT).
    8. Explain logical memory configuration versus physical memory configuration.
    9. Identify transistor level diagrams, design specifications, and uses of historical and modern volatile memory types.
    10. Identify transistor level diagrams, design specifications, and uses of historical and modern non-volatile memory types.
    11. Describe byte addressing and word addressing, and explain the difference between them.
    12. Describe addressing of bytes comprising a digital word in terms of big endian and little endian.
    13. Define data bus, and describe different bus architectures.
    14. Discuss the following design considerations master versus slave, internal versus external, bridged versus flat, memory versus peripheral, synchronous versus asynchronous, high speed versus low speed, serial versus parallel, single master versus multi-master, single layer versus multi-layer, multiplexed versus de-multiplexed, and data lines versus control lines.
    15. Identify timing diagram conventions, and read and write data transfer diagrams with and without wait-states.
    16. Describe the high-speed CPU to memory bus and the low-speed peripheral bus.
    17. Describe memory-mapped IO and contrast it to port-mapped IO and direct memory access.
5. Formulate assembly language into usable instruction.
    1. Describe the levels of source or program code.
    2. Diagram the C code translation hierarchy comprised of static compilation and dynamic compilation.
    3. Describe how object modules are produced and then linked.
    4. Identify the instruction portion of a line of assembly, and explain its function using the microprocessor's instruction set.
    5. Identify the operand portion of a line of assembly, and explain which components represent values stored in registers or memory or are immediate values.
    6. Identify the constituent microcontroller assembly instructions comprising each high- level language instruction making up a well-designed high-level language program.
    7. Interpret how the assembly code generated by a compiler relates to the high level program code.
6. Discuss interrupts and their application to a microprocessor.
    1. Define interrupt, interrupt vector, interrupt pseudo-vector, interrupt service routine (or handler), and exception.
    2. Explain how the diversity of I/O devices, the ability to know when I/O is necessary, and how the I/O is implemented necessitates interrupts.
    3. Describe important aspects of polling and contrast to the advantages and disadvantages of interrupts.
    4. Describe key questions and corresponding answers that provide understanding to when, where, how, and how often an interrupt is serviced.
    5. Describe vectored interrupts and nested interrupts.
    6. Explain interrupt priority, preempt priority, and sub-priority.
    7. Describe the interrupt handling procedure from the occurrence of the interrupt (prologue) to resumption of the microcontroller's initial context (epilogue).
7. Explain embedded system design concepts.
    1. Identify processors best suited for various applications and their corresponding data requirements in context with the choice of embedded system design.
    2. Explain trends driving embedded system design metrics of cost, performance, power, and size.
    3. Describe the integrated chip (IC) manufacturing process from silicon (Si) ingot to tested package dies.
    4. Describe the design, verification, implementation, re-verification, prototyping, re-verification, and production processes to final embedded system application.

5. Identify processes which involve the greatest cost for application specific integrated circuit (ASIC) and field programmable gate arrays (FPGA).
6. Describe embedded system performance in terms of peak versus sustained, average versus worst/best-case, and throughput versus latency.
7. Identify various design methodologies, and explain the flow of decision making for each.
8. Define *hardware design flow*, *platform design flow*, and *hardware/software co-design flow.*
9. Describe the different levels of integration of platform design flow.
10. Identify a hardware-centric view of a platform and a software-centric view of a platform, and explain advantages and disadvantages of each.
11. Describe co-design issues with emphasis on hardware/software partitioning and scheduling.
12. Describe partitioning styles, and compare hardware first approach to software first approach.
13. Describe partitioning costs in terms of software resources and hardware resources.
14. Describe how scheduling is managed with graphically representing inputs and outputs, and identifying scheduling constraints.
15. Determine the lowest cost schedule and the shortest time schedule using scheduling techniques described above.
16. Describe inter-processor communication cost.

8. Develop embedded system applications with the C programming language.
    1. Diagram program logic and flow through flowcharts of the embedded system applications.
    2. Identify variable types, operators, control flow structures, function prototypes, functions, and data structures.
    3. Develop well-designed high-level language programs.
    4. Develop well-documented high-level language programs.
    5. Perform programming tasks that involve sequence, decision, and repetition structures of programming.
    6. Create time delays.
    7. Interface I/O devices to the microcontroller.
    8. Write the code necessary to send/receive the data to/from these I/O devices.
    9. Create service routines to perform a function on the occurrence of a hardware or software interrupt.
    10. Predict, measure, and manipulate a program's execution time.

Evaluation Criteria/Policies:
Students must demonstrate proficiency on all CCPOs at a minimal 75 percent level to successfully complete the course. The grade will be determined using the Delaware Tech grading system:

92	**–**	**100**	**=**	**A**
83	**–**	**91**	**=**	**B**
75	**–**	**82**	**=**	**C**
0	**–**	**74**	**=**	**F**

Students should refer to the Student Handbook - https://www.dtcc.edu/handbook for information on the Academic Standing Policy, the Academic Integrity Policy, Student Rights and Responsibilities, and other policies relevant to their academic progress.

Core Curriculum Competencies (CCCs are the competencies every graduate will develop):

1. Apply clear and effective communication skills.
2. Use critical thinking to solve problems.
3. Collaborate to achieve a common goal.
4. Demonstrate professional and ethical conduct.
5. Use information literacy for effective vocational and/or academic research.
6. Apply quantitative reasoning and/or scientific inquiry to solve practical problems.

Program Graduate Competencies (PGCs are the competencies every graduate will develop specific to his or her major):

1. Integrate modern tools of the engineering discipline into the field of study.
2. Apply mathematics, science, engineering, and technology theory to solve electrical and computer engineering and electronics engineering technology problems.
3. Conduct, analyze, and interpret experiments using analysis tools and troubleshooting methods.
4. Identify, analyze, and solve electrical and computer engineering and electronics engineering technology problems.
5. Explain the importance of engaging in self-directed continuing professional development.
6. Demonstrate basic management, organizational, and leadership skills that commit to quality, timeliness, and continuous improvement.

Disabilities Support Statement:
The College is committed to providing reasonable accommodations for students with disabilities. Students are encouraged to schedule an appointment with the campus Disabilities Support Counselor to request an accommodation needed due to a disability. A listing of campus Disabilities Support Counselors and contact information can be found at the disabilities services - https://www.dtcc.edu/disabilitysupport web page or visit the campus Advising Center.