Subject:	Computer Science	**Course/Grade Level:**	Operating System Design / 11th-12th
Focus Statement:	Students will show how to build a computer from the ground up from hardware logic to the operating system. Students will show how to create the computer architecture, assembler, programming language, compiler, and operating system.		

Outcome 1:

CTE.OSD.1		**Students will show how to create more complex logic gates using only NAND gates.**	
Pacing:		**Local Code:**	**Components:**
Instruct	**Assess**		**Students will:**
NA	NA	CTE.OSD.1.1	Show how to represent boolean functions using truth tables.
NA	NA	CTE.OSD.1.2	Show how to design a composite logic gate using primitive logic gates.
NA	NA	CTE.OSD.1.3	Show how to build and test a composite logic gate using a hardware description language (HDL).
NA	NA	CTE.OSD.1.4	Show how to test a hardware design from an HDL in a hardware simulator.
NA	NA	CTE.OSD.1.5	Design an NAND gate in an HDL.
NA	NA	CTE.OSD.1.6	Design basic logic gates including And, Or, Xor, Multiplexor (Mux) and Demultiplexer (DMux) in an HDL.

NA	NA	CTE.OSD.1.7	Design multi-bit logic gates in an HDL.
NA	NA	CTE.OSD.1.8	Design multi-way logic gates in an HDL.

Outcome 2:

CTE.OSD.2		**Students will show how to use boolean arithmetic to create arithmetic chips including half-adders, full-adders, adders, incrementers, and ALUs.**	
Pacing:		**Local Code:**	**Components:**
Instruct	**Assess**		**Students will:**
NA	NA	CTE.OSD.2.1	Show how to add binary numbers, including dealing with overflow.
NA	NA	CTE.OSD.2.2	Show how to represent signed numbers using binary.
NA	NA	CTE.OSD.2.3	Design half-adders, full-adders, and adders in an HDL.
NA	NA	CTE.OSD.2.4	Design an incrementer in an HDL.
NA	NA	CTE.OSD.2.5	Design an arithmetic logic unit (ALU) in an HDL.

Outcome 3:

CTE.OSD.3		**Students will show how to build chips that can maintain state such as registers, memory, and counters using Data Flip-Flop (DFF) gates.**	
Pacing:		**Local Code:**	**Components:**

Instruct	**Assess**		**Students will:**
NA	NA	CTE.OSD.3.1	Explain how a computer keeps track of time.
NA	NA	CTE.OSD.3.2	Explain how a DFF works.
NA	NA	CTE.OSD.3.3	Design a 1-bit register using DFF gates in an HDL.
NA	NA	CTE.OSD.3.4	Design a Random Access Memory (RAM) unit using DFF gates in an HDL.
NA	NA	CTE.OSD.3.5	Design a counter using DFF gates in an HDL.

Outcome 4:

CTE.OSD.4		**Students will show how to program using machine language.**	
Pacing:		**Local Code:**	**Components:**
Instruct	**Assess**		**Students will:**
NA	NA	CTE.OSD.4.1	Explain how memory, the central processing unit (CPU), and registers work together to run a program.
NA	NA	CTE.OSD.4.2	Show how to use arithmetic and logic operations in a machine language.
NA	NA	CTE.OSD.4.3	Show how to access memory using direct addressing, immediate addressing, and indirect addressing.
NA	NA	CTE.OSD.4.4	Show how to use conditional jump and unconditional jump commands to control program flow.

Outcome 5:

CTE.OSD.5		**Students will show how to build a computer from logic gate designs.**	
Pacing:		**Local Code:**	**Components:**
Instruct	**Assess**		**Students will:**
NA	NA	CTE.OSD.5.1	Describe the components of a von Neumann machine.
NA	NA	CTE.OSD.5.2	Compare and contrast data memory and instruction memory.
NA	NA	CTE.OSD.5.3	Describe how a CPU consisting of an arithmetic logic unit (ALU), registers, and a control unit processes instructions.
NA	NA	CTE.OSD.5.4	Explain the differences between data registers, addressing registers, and a program counter register.
NA	NA	CTE.OSD.5.5	Show how memory-mapped I/O can be used to connect input and output devices to a computer.

Outcome 6:

CTE.OSD.6		**Students will develop an assembler that translates assembly language into binary code.**	
Pacing:		**Local Code:**	**Components:**
Instruct	**Assess**		**Students will:**
NA	NA	CTE.OSD.6.1	Describe the tasks necessary for an assembler to translate assembly language into binary instructions.
NA	NA	CTE.OSD.6.2	Write an assembler for programs with no symbols.
NA	NA	CTE.OSD.6.3	Write an assembler for programs with symbols.

Outcome 7:

CTE.OSD.7		**Students will develop a virtual machine that will run intermediate code.**	
Pacing:		**Local Code:**	**Components:**
Instruct	**Assess**		**Students will:**
NA	NA	CTE.OSD.7.1	Describe the benefits of a two-tiered translation model for compiling a high-level computer language.
NA	NA	CTE.OSD.7.2	Describe the stack machine model.
NA	NA	CTE.OSD.7.3	Implement stack arithmetic commands.
NA	NA	CTE.OSD.7.4	Create push and pop commands for a stack implementation.
NA	NA	CTE.OSD.7.5	Implement program flow commands for a virtual machine.
NA	NA	CTE.OSD.7.6	Implement function calling commands for a virtual machine.

Outcome 8:

CTE.OSD.8		**Students will build a compiler to translate computer programs from one language to another.**	
Pacing:		**Local Code:**	**Components:**
Instruct	**Assess**		**Students will:**
NA	NA	CTE.OSD.8.1	Describe what a compiler is and its role in the design of a computer programming language.
NA	NA	CTE.OSD.8.2	Create a tokenizer to categorize code into tokens.
NA	NA	CTE.OSD.8.3	Create a parser to handle lexical elements, program structure, and statements.
NA	NA	CTE.OSD.8.4	Create a parser to handle expressions.
NA	NA	CTE.OSD.8.5	Create a symbol table module as a part of a syntax analyzer.
NA	NA	CTE.OSD.8.6	Create a full compiler with code generation features.

Outcome 9:

CTE.OSD.9		**Students will build an operating system.**	
Pacing:		**Local Code:**	**Components:**
Instruct	**Assess**		**Students will:**
NA	NA	CTE.OSD.9.1	Implement a dynamic memory allocation algorithm for an operating system.
NA	NA	CTE.OSD.9.2	Develop a system for storing arrays and other variable-length entities in an operating system.
NA	NA	CTE.OSD.9.3	Implement simple math operations such as addition, subtraction, multiplication, and division in an operating system.
NA	NA	CTE.OSD.9.4	Implement mathematical functions such as absolute value, min, max, and square root in an operating system.
NA	NA	CTE.OSD.9.5	Develop a system for using strings in an operating system.
NA	NA	CTE.OSD.9.6	Develop a system for writing text to the screen in an operating system.
NA	NA	CTE.OSD.9.7	Develop a system for drawing graphics to the screen in an operating system including color, lines, rectangles, and circles.
NA	NA	CTE.OSD.9.8	Develop a system for accepting input from a keyboard in an operating system.