CS M145: COMPUTER ARCHITECTURE AND ORGANIZATION

Originator
enikjeh

Co-Contributor(s)

Name(s)
Loay, Alnaji (lalnaji)

College
Moorpark College

Discipline (CB01A)
CS - Computer Science

Course Number (CB01B)
M145

Course Title (CB02)
Computer Architecture and Organization

Banner/Short Title
Computer Arch. & Organization

Credit Type
Credit

Start Term
Spring 2021

Catalog Course Description
Introduces the organization and behavior of real computer systems at the assembly language level. Studies the mapping of statements and constructs in a high-level language into sequences of machine instructions. Discusses the internal representation of simple data types and structures and examines numerical computation, data representation errors and procedural errors.

Taxonomy of Programs (TOP) Code (CB03)
0706.00 - Computer Science (transfer)

Course Credit Status (CB04)
D (Credit - Degree Applicable)

Course Transfer Status (CB05) (select one only)
A (Transferable to both UC and CSU)

Course Basic Skills Status (CB08)
N - The Course is Not a Basic Skills Course

SAM Priority Code (CB09)
E - Non-Occupational

Course Cooperative Work Experience Education Status (CB10)
N - Is Not Part of a Cooperative Work Experience Education Program

Course Classification Status (CB11)
Y - Credit Course

Educational Assistance Class Instruction (Approved Special Class) (CB13)
N - The Course is Not an Approved Special Class

Course Prior to Transfer Level (CB21)
Y - Not Applicable

Course Noncredit Category (CB22)
Y - Credit Course

Funding Agency Category (CB23)
Y - Not Applicable (Funding Not Used)

Course Program Status (CB24)
1 - Program Applicable

General Education Status (CB25)
Y - Not Applicable

Support Course Status (CB26)
N - Course is not a support course

Field trips
Will not be required

Grading method
Letter Graded

Alternate grading methods
Student Option- Letter/Pass
Pass/No Pass Grading

Does this course require an instructional materials fee?
No

Repeatable for Credit
No

Is this course part of a family?
No

Units and Hours

Carnegie Unit Override
No

In-Class

Lecture
Minimum Contact/In-Class Lecture Hours
43.75
Maximum Contact/In-Class Lecture Hours
43.75

Activity

Laboratory
Minimum Contact/In-Class Laboratory Hours
26.25
Maximum Contact/In-Class Laboratory Hours
26.25

Total in-Class

Total in-Class
Total Minimum Contact/In-Class Hours
70
Total Maximum Contact/In-Class Hours
70

Outside-of-Class

Internship/Cooperative Work Experience

Paid

Unpaid

Total Outside-of-Class

Total Outside-of-Class
Minimum Outside-of-Class Hours
87.5
Maximum Outside-of-Class Hours
87.5

Total Student Learning

Total Student Learning
Total Minimum Student Learning Hours
157.5
Total Maximum Student Learning Hours
157.5

Minimum Units (CB07)
3
Maximum Units (CB06)
3

Prerequisites
CS M10A OR CS M125

Entrance Skills

Entrance Skills
CS M10A OR CS M125

Prerequisite Course Objectives
CS M10A-describe the basic organization of a computer system.
CS M10A-describe the basic components, syntax, and semantics of the C++ programming language.
CS M10A-analyze programming problems and design algorithms to solve those problems.
CS M10A-describe and identify sequential, selection, and iteration control structures.
CS M10A-describe and apply the concepts of structured programming including function usage and parameter passing.
CS M10A-describe and apply composite data types such as arrays and structures.
CS M10A-describe and apply user defined data types such as enumerations and structured data.
CS M10A-describe, analyze, and use the C++ string class.
CS M10A-describe and apply dynamic memory allocation using pointers.
CS M10A-describe and apply file input and output.
CS M10A-describe and identify good programming practice and style.
CS M125-describe the basic organization of a computer system.
CS M125-describe the basic components, syntax, and semantics of the C++ programming language.
CS M125-analyze programming problems and design algorithms to solve those problems.
CS M125-identify sequential, selection, and iteration control structures.
CS M125-apply the concepts of structured programming including function usage and parameter passing.
CS M125-apply composite data types such as arrays and structures.
CS M125-demonstrate and understanding of user defined data types such as enumerations and structured data.
CS M125-describe and apply dynamic memory allocation using pointers.
CS M125-identify file input and output.
CS M125-identify good programming practice and style.

Requisite Justification

Requisite Type

Prerequisite

Requisite

CS M125 OR CS M10A

Requisite Description

Course in a sequence

Level of Scrutiny/Justification

Required by 4 year institution

Student Learning Outcomes (CSLOs)

	Upon satisfactory completion of the course, students will be able to:
1	understand register transfer language and how register transfer language is used to describe the design and organization of a basic computer system.
2	understand basic digital circuits (full adders, half-adders, decoders, multiplexers, registers and ALU's).
3	understand Boolean algebra, logic gates, and flip-flops.
4	understand 80x86 assembly language program(s) using arithmetic, logical, shift and rotate instructions.
5	understand 80x86 assembly language syntax including mnemonics, registers, different operand types and addressing modes.
6	understand data representation including number systems, signed and unsigned integers, and IEEE floating point representation of data.

Course Objectives

	Upon satisfactory completion of the course, students will be able to:
1	explain why everything is represented as data, including instructions, in computers.
2	explain the reasons for using alternative formats to represent numerical data.
3	describe how negative integers are stored in sign-magnitude and twos-complement representations.
4	explain how fixed-length number representations affect accuracy and precision.
5	describe the internal representation of non-numeric data, such as characters, strings, records, and arrays.
6	convert numerical data from one format to another.
7	write simple programs at the assembly/machine level.
8	explain the organization of the classical von Neumann machine and its major functional units.
9	describe how an instruction is executed in a classical von Neumann machine with extensions for threads, multiprocessor synchronization, and single instruction, multiple data (SIMD) execution.
10	describe instruction-level parallelism and hazards and how they are managed in typical processor pipelines.
11	summarize how instructions are represented at both the machine level and in the context of a symbolic assembler.
12	demonstrate how to map between high-level language patterns into assembly/machine language notations.
13	explain different instruction formats, such as addresses per instruction and variable length vs. fixed length formats.
14	explain how subroutine calls are handled at the assembly level.
15	demonstrate how fundamental high-level programming constructs are implemented at the machine language level.
16	explain how the concept of layers is used in computer design.
17	explain the basic components of memory (gates, flip-flops, latches, etc.) and design and simulate basic circuits.
18	explain and use of boolean algebra to design and implement (through simulation) logic circuits.
19	explain virtual memory (page and segmented models), the techniques used to implement it, and issues associated with it.

Course Content

Lecture/Course Content

(5%) Introduction to Computer Architecture and Organization:

• Concept of computer architecture
• Concept of layers and multilayer machines
• Concept of virtual (logical) machines
• Concept that hardware and software are logically equivalent
• Introduction to how computers process programs
• Historical view of computer architecture development

(15%) Operating System Level
• Page and segmented memory models
• Different memory types including cache
• Virtual I/O
• Processes including race conditions and synchronization

(15%) Gates, Boolean Algebra, and Microarchitecture
• Concept of gates
• Boolean algebra and the implementation of boolean functions
• Basic digital logic circuits including clocking
• Memory including flip-flops, latches, registers, and organization
• Busses
• I/O interfaces and interrupts

(40%) Assembly Level Machine Organization
• Basic organization of the von Neumann machine
• Control unit; instruction fetch, decode, and execution
• Instruction sets and types (data manipulation, control, Input/output (I/O))
• Assembly/machine language programming
• Instruction formats
• Addressing modes
• Subroutine call and return mechanisms

(25%) Machine Level Representation of Data
• Bits, bytes, and words
• Numeric data representation and number bases
• Fixed- and floating-point systems
• Signed and twos-complement representations
• Representation of non-numeric data (character codes, graphical data)
• Representation of records and arrays

Laboratory or Activity Content

(17%) Create an assembly language program from scratch demonstrating proper syntax and semantics for each of the following general programming concepts:
1.Mnemonics
2.Registers: general purpose, index, flag
3.Operand Types: register, immediate, memory
4.Direct Addressing Modes: direct and direct-offset
5.Indirect Addressing Modes: register-indirect, indexed, base-indexed, base-indexed with displacement, scaled-indexed
6.Overall Program Considerations:
• Good program design including correct use of functions
• Program must assemble and link and correctly execute
• Program must have good programming style including documentation and correct program formatting

(17%) Create a program in assembly language to convert a high-level programming language to a machine level program.

(15%) Create a simulation in a high level language of virtual memory with:
• Loading a program
• Paging
• Swap file
• Algorithm to determine which pages should be swapped

(17%) Demonstrate proper use of basic digital circuits by designing and building a functional digital circuit using basic digital circuits as building blocks:
1.Full-Adder/Half-Adder
2.Decoders
3.Multiplexers
4.Registers
5.Random Access Memory (RAM)/Read-Only Memory (ROM)
6.Arithmetic Logic Unit (ALU)

(17%) Digital design assignment demonstrating proper use of the following concepts:
1.Boolean Algebra
2.Karnaugh Maps
3.Flip-Flops
4.Circuits

5.Sequential Circuit Analysis and Design

(17%) Create an assembly language program from scratch demonstrating proper syntax and semantics for each of the following general programming concepts:
1.Arithmetic instructions: correct use of signed/unsigned arithmetic Instructions and operands
2.Logical Instructions (AND, OR, XOR, NOT)
3.Shift and Rotate Instructions including the use of the carry and overflow flags
4.Overall Program Considerations:
- Good program design including correct use of functions
- Program must assemble and link and correctly execute
- Program must have good programming style including documentation and correct program formatting

Methods of Evaluation

Which of these methods will students use to demonstrate proficiency in the subject matter of this course? (Check all that apply):

Problem solving exercises

Methods of Evaluation may include, but are not limited to, the following typical classroom assessment techniques/required assignments (check as many as are deemed appropriate):

Classroom Discussion
Objective exams
Other (specify)
Projects
Problem-solving exams
Participation
Reports/Papers/Journals

Other

Written homework exercises to demonstrate knowledge of concepts and techniques.
Programming assignments requiring students to analyze a problem, determine a solution, implement the solution using an assembly language, and test and verify the program

Instructional Methodology

Specify the methods of instruction that may be employed in this course

Distance Education
Laboratory activities
Lecture

Describe specific examples of the methods the instructor will use:

- Presentation of programming concepts with detailed coding examples
- Practice problems to develop proper programming skills and techniques
- Student/instructor interaction using questions and answers
- Projects and/or group work to enhance student understanding of the concepts, for example the will assign group work and allow students time in class to complete the work as a group and then present in front of class.

Representative Course Assignments

Writing Assignments

- Answer homework questions, using correct grammar, syntax, and terms, on concepts such as memory, circuit design and the components used in the design, virtual memory, layers (levels), and assembly language. These questions will be taken from the text and other sources.
- Comment on assembly programs, using correct grammar, syntax, and terminology, to document what a program is doing. These comments are in the programs themselves.

Critical Thinking Assignments

- Evaluate different algorithms for determining which pages should be swapped out of memory when virtual memory is used.
- Evaluate and correct a circuit design which does not function correctly.

Reading Assignments

- Go over our "Data Representation and Number system" chapter and read about the different numbering systems such as Binary and Hex

- Read and review the following published document explaining the different applications of Flip-FLops: https://www.lancasterschools.org/cms/lib/NY19000266/Centricity/Domain/1055/FlipFlopApplications.pdf

Skills Demonstrations

- Using the design tool listed below, design an SR Latch and print the truth table generated from your design. The tool can be found at: https://logic.ly/demo/samples
- Write an assembly program that can read an expression as an input and evaluate it and display the result. For example, the input could be: 5 + 2, the output should be: 5 + 2 = 7. You must implement the different skills learned in class including modularization (functions and procedures), conditional statements and proper memory management.

Outside Assignments

Representative Outside Assignments

- Design a counter using JK FF that counts in the following sequence:

5->4->3- > 2 - > 1 -> 0 then back to 5

Design it, test it, submit a 20 second video showing that it runs. Make sure you use a professional tool with a "time" display such as a virtual LED display.

- Create a circuit that adds 2 - 2 bit numbers. The circuit should give the following results:

  00 + 00 = 00

  00 + 01 = 01

  00 + 10 = 10

  00 + 11 = 11

  01 + 00 = 01

  01 + 01 = 10

  .11 + 11 = 10 with carry 1

  Make sure you identify the inputs in your circuits, identify each logic area by labeling the section explaining what it does, the Carry flag should be a light that turns on if it's 1, turns off if it's 0. Submit a word document with two screenshots. One screenshot showing your circuit, the other showing the truth table generated from your circuit

Articulation

C-ID Descriptor Number

COMP 142

Status

Approved

Equivalent Courses at 4 year institutions

University	Course ID	Course Title	Units
CSU Channel Islands	COMP 162	Computer Architecture and Assembly Language	3
CSU Chico	CSCI 221	Assembly Language Programming	3
CSU Dominguez Hills	CSC 221	Assembly Language and Introduction to Computer Organization	3
CSU Bakersfield	CMPS 2240	Computer Architecture I: Assembly Language Programming	4
CSU Northridge	COMP 122/122L	Computer Architecture and Assembly Language and Lab	1/1

District General Education

A. Natural Sciences

B. Social and Behavioral Sciences

C. Humanities

D. Language and Rationality

E. Health and Physical Education/Kinesiology

F. Ethnic Studies/Gender Studies

Course is CSU transferable
Yes

CSU Baccalaureate List effective term:
Fall 2016

CSU GE-Breadth

Area A: English Language Communication and Critical Thinking

Area B: Scientific Inquiry and Quantitative Reasoning

Area C: Arts and Humanities

Area D: Social Sciences

Area E: Lifelong Learning and Self-Development

CSU Graduation Requirement in U.S. History, Constitution and American Ideals:

UC TCA

UC TCA
Approved

IGETC

Area 1: English Communication

Area 2A: Mathematical Concepts & Quantitative Reasoning

Area 3: Arts and Humanities

Area 4: Social and Behavioral Sciences

Area 5: Physical and Biological Sciences

Area 6: Languages Other than English (LOTE)

Textbooks and Lab Manuals

Resource Type
Textbook

Description
Tanenbaum, Andrew S., and Todd Austin. *Structured Computer Organization*. 6th ed., Pearson, 2013.

Resource Type
Textbook

Description
Plantz, Robert G. *Introduction to Computer Organization with x86-64 Assembly Language and GNU/Linux*. No Starch Press, 2020.

Resource Type
Textbook

Description
Stallings, William. *Computer Organization and Architecture.* E-book, 11th ed., Pearson, 2019.

Resource Type
Textbook

Description
NASM Assembly. E-book, Tutorials Point, 2020, https://www.tutorialspoint.com/assembly_programming/index.htm. Accessed 3 September 2020.

Library Resources

Assignments requiring library resources
Locate articles in professional and academic publications.

Sufficient Library Resources exist
Yes

Example of Assignments Requiring Library Resources
Using the Library's print and online resources, research different assemblers and discuss how you can ensure the code you write for one CPU can run on a different type of CPU.

Distance Education Addendum

Definitions

Distance Education Modalities
Hybrid (51%–99% online)
Hybrid (1%–50% online)
100% online

Faculty Certifications

Faculty assigned to teach Hybrid or Fully Online sections of this course will receive training in how to satisfy the Federal and state regulations governing regular effective/substantive contact for distance education. The training will include common elements in the district-supported learning management system (LMS), online teaching methods, regular effective/substantive contact, and best practices.
Yes

Faculty assigned to teach Hybrid or Fully Online sections of this course will meet with the EAC Alternate Media Specialist to ensure that the course content meets the required Federal and state accessibility standards for access by students with disabilities. Common areas for discussion include accessibility of PDF files, images, captioning of videos, Power Point presentations, math and scientific notation, and ensuring the use of style mark-up in Word documents.
Yes

Regular Effective/Substantive Contact

Hybrid (1%–50% online) Modality:

Method of Instruction	Document typical activities or assignments for each method of instruction
E-mail	Instructor will email students with announcements about the course or an
Synchronous Dialog (e.g., online chat)	Instructor will schedule a zoom meeting or in-class meeting with students at least once a week
Asynchronous Dialog (e.g., discussion board)	Instructor will post weekly discussion questions in Canvas. Students need to respond to these DQs by posting substantive messages in order to participate in the class.
Other DE (e.g., recorded lectures)	Instructor may record the lectures and post them for students to view within a specified time frame to be ready for the accompanying assignment.

Hybrid (51%–99% online) Modality:

Method of Instruction	Document typical activities or assignments for each method of instruction
E-mail	Instructor will email students with announcements about the course or an upcoming event.
Asynchronous Dialog (e.g., discussion board)	Instructor will post weekly discussion questions in Canvas. Students need to respond to these DQs by posting substantive messages in order to participate in the class.
Other DE (e.g., recorded lectures)	Instructor may record the lectures and post them for students to view within a specified time frame to be ready for the accompanying assignment.
Synchronous Dialog (e.g., online chat)	Instructor will schedule a zoom meeting or in-class meeting with students at least once a week

100% online Modality:

Method of Instruction	Document typical activities or assignments for each method of instruction
E-mail	Instructor will email students with announcements about the course or an upcoming event.
Synchronous Dialog (e.g., online chat)	The instructor may be available on certain day(s) for a certain time frame to help students.
Other DE (e.g., recorded lectures)	Instructor may record the lectures and post them for students to view within a specified time frame to be ready for the accompanying assignment.
Asynchronous Dialog (e.g., discussion board)	Instructor will post a question, students will respond to the question.

Examinations

Hybrid (1%–50% online) Modality

Online
On campus

Hybrid (51%–99% online) Modality

Online
On campus

Primary Minimum Qualification

COMPUTER SCIENCE

Review and Approval Dates

Department Chair

07/29/2020

Dean
07/29/2020

Technical Review
09/03/2020

Curriculum Committee
9/15/2020

DTRW-I
MM/DD/YYYY

Curriculum Committee
MM/DD/YYYY

Board
MM/DD/YYYY

CCCCO
11/05/2020

Control Number
CCC000572320

DOE/accreditation approval date
MM/DD/YYYY