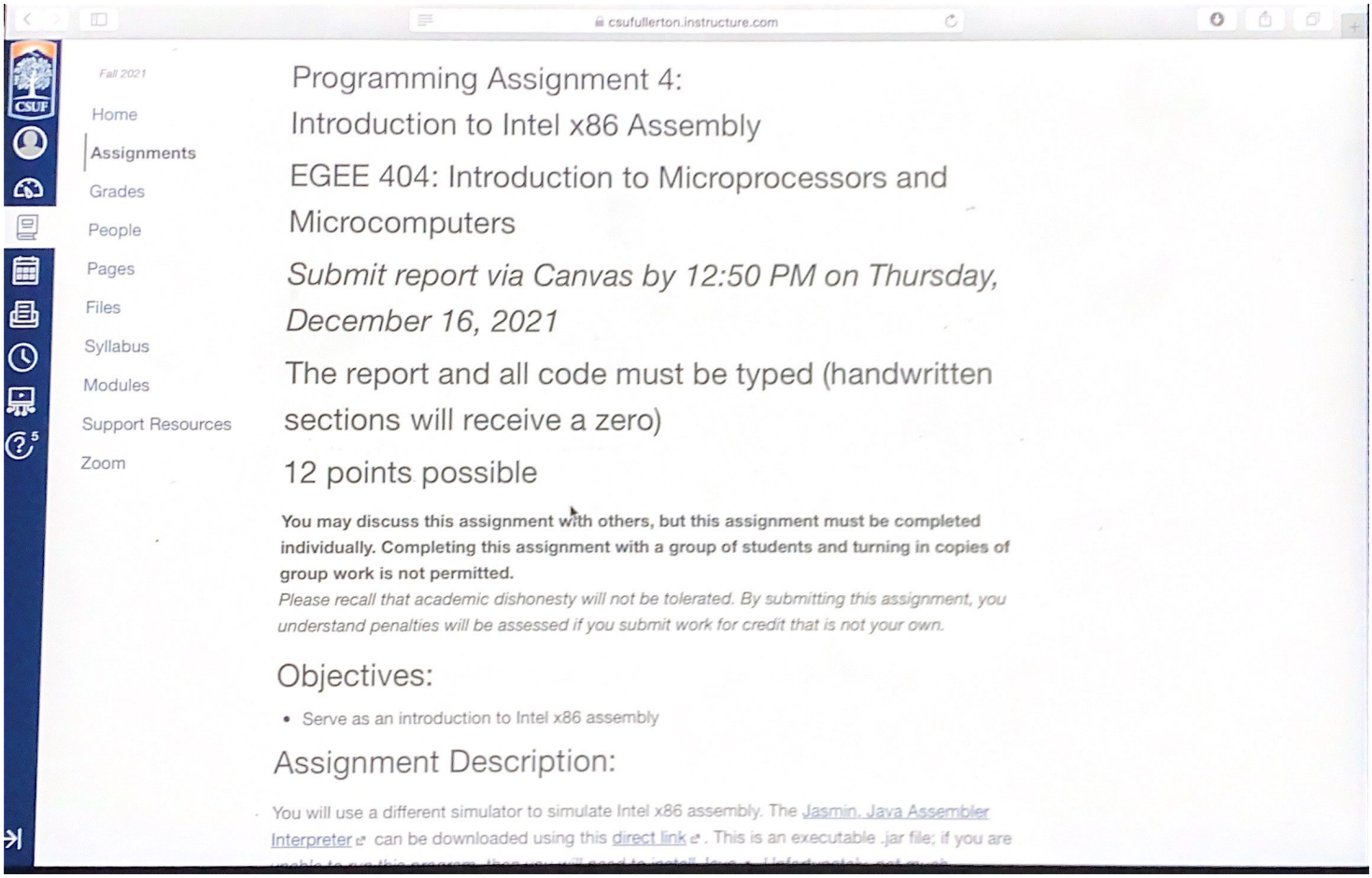

csufullerton.instructure.com
Fall 2021
Home
Assignments
Grades
People
Pages
Files
Syllabus
Modules
Support Resources
Zoom
Programming Assignment 4:
Introduction to Intel x86 Assembly
EGEE 404: Introduction to Microprocessors and Microcomputers
Submit report via Canvas by 12:50 PM on Thursday, December 16, 2021
The report and all code must be typed (handwritten sections will receive a zero)
12 points possible
You may discuss this assignment with others, but this assignment must be completed individually. Completing this assignment with a group of students and turning in copies of group work is not permitted.
Please recall that academic dishonesty will not be tolerated. By submitting this assignment, you understand penalties will be assessed if you submit work for credit that is not your own.
Objectives:
Serve as an introduction to Intel x86 assembly
Assignment Description:
You will use a different simulator to simulate Intel x86 assembly. The Jasmin, Java Assembler Interpreter can be downloaded using this direct link. This is an executable .jar file; if you are

Assignment Description:

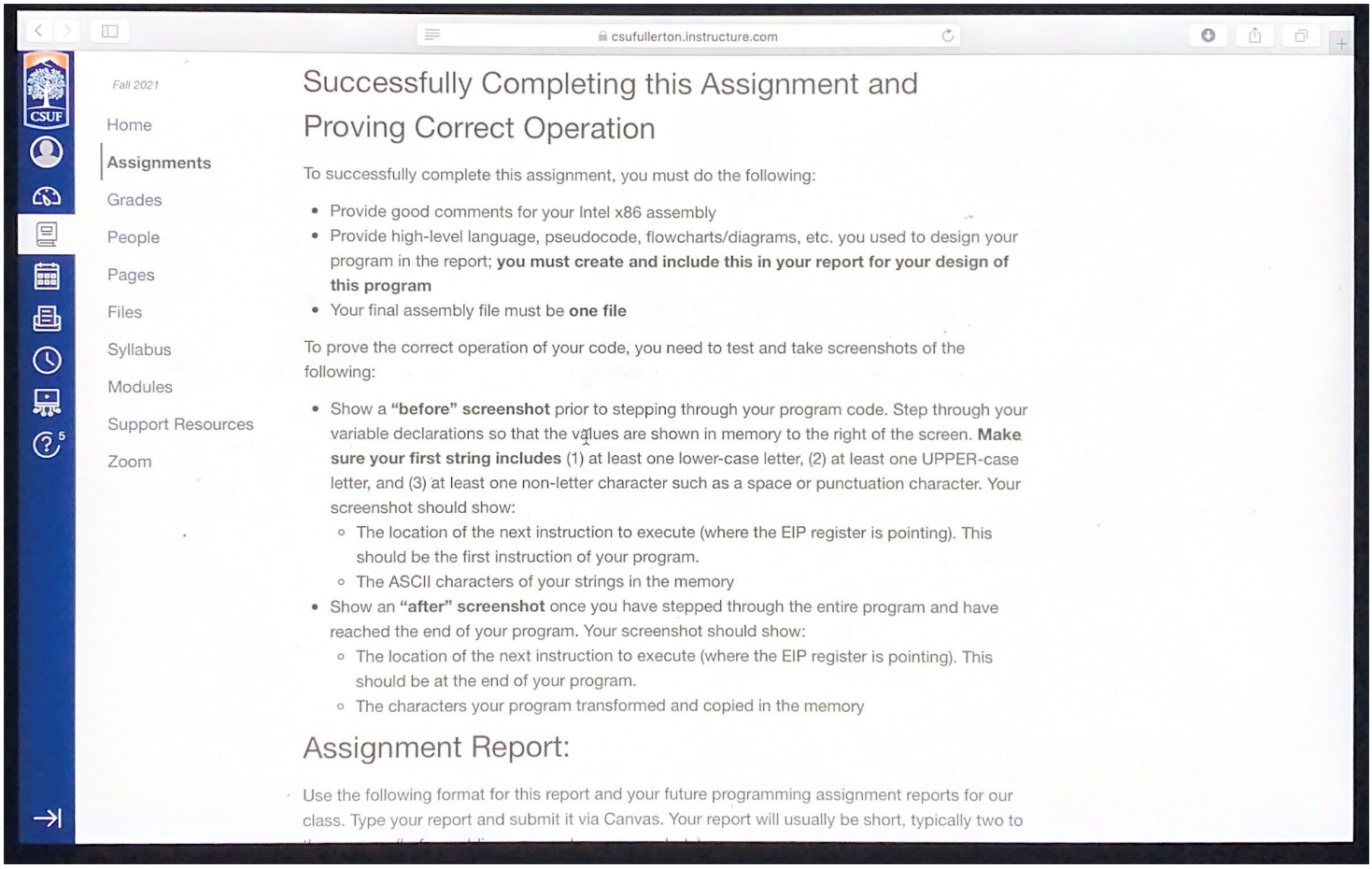
You will use a different simulator to simulate Intel x86 assembly. The Jasmin, Java Assembler Interpreter can be downloaded using this direct link. This is an executable .jar file; if you are unable to run this program, then you will need to install Java. Unfortunately, not much documentation is available for this simulator; please instead refer to the demonstration of Jasmin during our lecture.

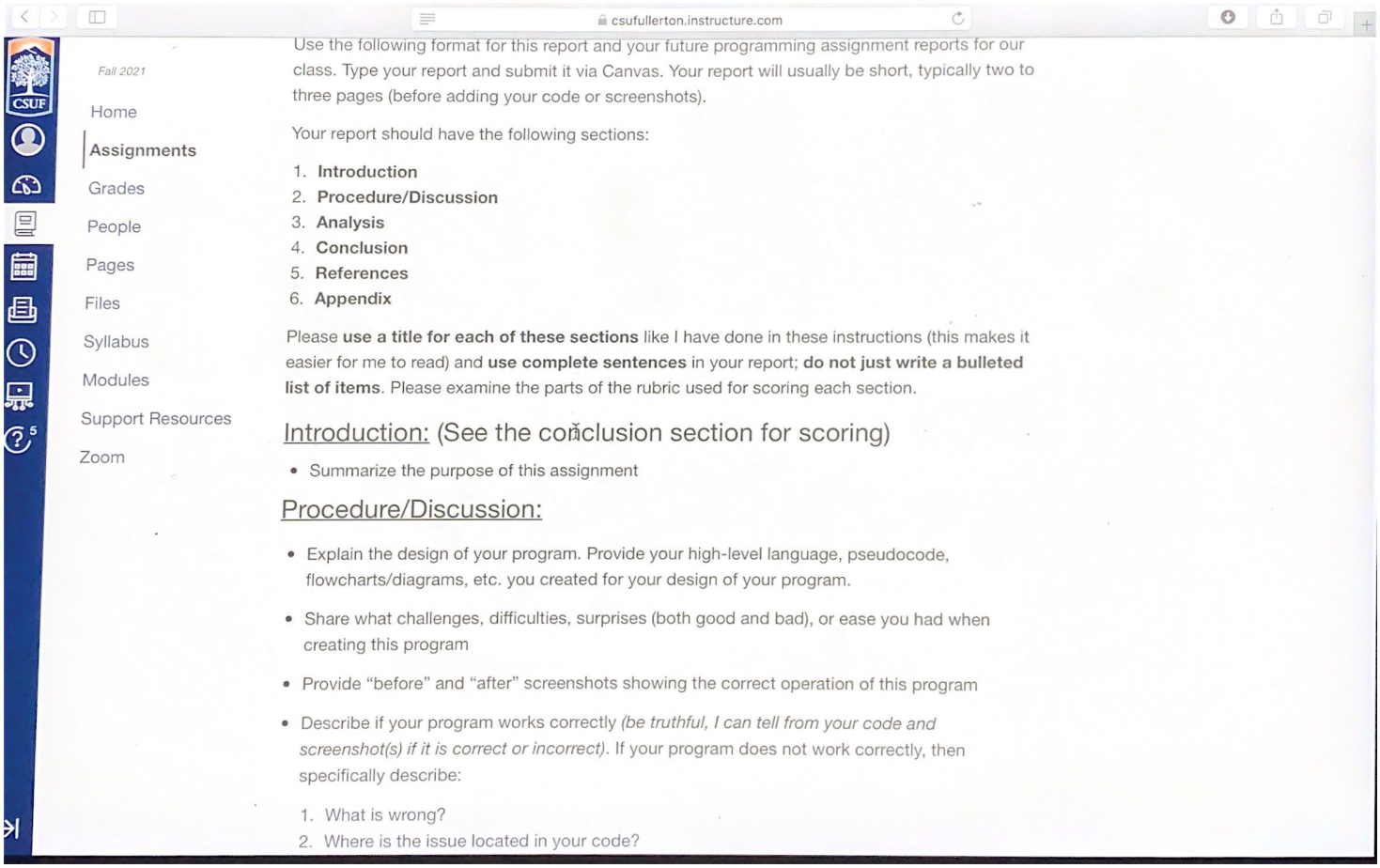
In this program, you will write a program similar to Programming Assignment 3. For this assignment you must:

1. Create two, null-terminated strings. The first string **must contain** (1) at least one lower-case letter, (2) at least one UPPER-case letter, and (3) at least one non-letter character such as a space or punctuation character.
2. Write Intel x86 assembly to copy, character by character, the first string to the second string (you will overwrite the second string). However, you must **switch the capitalization of any letters**. Non-letter characters should be echoed, but not modified. I recommend that you examine an ASCII table when determining how to code this.
3. Stop copying and transforming the string after the null-character (i.e., 0) is found; remember that the null-character is used to mark the end of a string.

Assignment Recommendations:

- You may hard-code or use global variables/constants for common values that you may use repeatedly your program; e.g., base addresses or mask values.
  - Note, for a global variable or constant, you could just use a register that you use nowhere else in your program to hold the value
- I highly suggest that your approach your program should be the following:
  - First, design an algorithm using high-level language, pseudocode, flowcharts/diagrams, etc.
  - Next, translate this algorithm into Intel x86 assembly then write instructions to test it


csufullerton.instructure.com
Fall 2021
Home
Assignments
Grades
People
Pages
Files
Syllabus
Modules
Support Resources
Zoom
Successfully Completing this Assignment and Proving Correct Operation
To successfully complete this assignment, you must do the following:
Provide good comments for your Intel x86 assembly
Provide high-level language, pseudocode, flowcharts/diagrams, etc. you used to design your program in the report; you must create and include this in your report for your design of this program
Your final assembly file must be one file
To prove the correct operation of your code, you need to test and take screenshots of the following:
Show a "before" screenshot prior to stepping through your program code. Step through your variable declarations so that the values are shown in memory to the right of the screen. Make sure your first string includes (1) at least one lower-case letter, (2) at least one UPPER-case letter, and (3) at least one non-letter character such as a space or punctuation character. Your screenshot should show:
The location of the next instruction to execute (where the EIP register is pointing). This should be the first instruction of your program.
The ASCII characters of your strings in the memory
Show an "after" screenshot once you have stepped through the entire program and have reached the end of your program. Your screenshot should show:
The location of the next instruction to execute (where the EIP register is pointing). This should be at the end of your program.
The characters your program transformed and copied in the memory
Assignment Report:
Use the following format for this report and your future programming assignment reports for our class. Type your report and submit it via Canvas. Your report will usually be short, typically two to


csufullerton.instructure.com
Fall 2021
Home
Assignments
Grades
People
Pages
Files
Syllabus
Modules
Support Resources
Zoom
Use the following format for this report and your future programming assignment reports for our class. Type your report and submit it via Canvas. Your report will usually be short, typically two to three pages (before adding your code or screenshots).
Your report should have the following sections:
1. Introduction
2. Procedure/Discussion
3. Analysis
4. Conclusion
5. References
6. Appendix
Please use a title for each of these sections like I have done in these instructions (this makes it easier for me to read) and use complete sentences in your report; do not just write a bulleted list of items. Please examine the parts of the rubric used for scoring each section.
Introduction: (See the conclusion section for scoring)
• Summarize the purpose of this assignment
Procedure/Discussion:
• Explain the design of your program. Provide your high-level language, pseudocode, flowcharts/diagrams, etc. you created for your design of your program.
• Share what challenges, difficulties, surprises (both good and bad), or ease you had when creating this program
• Provide "before" and "after" screenshots showing the correct operation of this program
• Describe if your program works correctly (be truthful, I can tell from your code and screenshot(s) if it is correct or incorrect). If your program does not work correctly, then specifically describe:
1. What is wrong?
2. Where is the issue located in your code?

3. How are you able to determine this?

Scoring for Procedure/Discussion (3 points possible)

- **Full credit (3/3) if:**
  - Explaining the design of the program:
    - Student thoroughly explains the process of writing the assembly program
    - Student includes high-level language, pseudocode, flowcharts/diagrams, etc. that were used to design the program
  - Proving correct operation of the program:
    - Student proves the code works properly or has a minor issue (but able to explain this thoroughly) with choice of test(s) and screenshot(s) and correctly identifies that the code works properly or has a specific, minor issue
- **1-point deduction for each of the following:**
  - Explaining the design of the program:
    - Student provides a good explanation of the process of writing the assembly program, *but did not provide information about an important part of the design process*
    - Student *forgets to include high-level language, pseudocode, flowcharts/diagrams, etc. that were used to design the program*
  - Proving correct operation of the program:
    - Student proves the code works properly or has a minor issue (but able to explain this thoroughly) with choice of test(s) and screenshot(s), *but does not identify that the code works properly or has a specific, minor issue*
    - Student *has some difficulty proving the code works properly (or has a minor issue) with choice of test(s) and screenshot(s)*, but correctly identifies that the code works properly or has a specific, minor issue
    - Student *proves the code has a major issue (but able to explain this thoroughly) with choice of test(s) and screenshot(s) and correctly identifies the specific, major issue with the code*
- **2-point deduction for each of the following:**
  - Explaining the design of the program:
    - Student provides a fair explanation of the process of writing the assembly program,

- Student provides a fair explanation of the process of writing the assembly program, *but did not provide information about multiple important parts of the design process*
- Proving correct operation of the program:
  - Student proves the code has a major issue (but able to explain this thoroughly) with choice of test(s) and screenshot(s), *but does not identify the specific, major issue with the code*
  - Student *has significant difficulty proving the code works properly (or has a minor issue) with choice of test(s) and screenshot(s)*, but correctly identifies that the code works properly or has a specific, minor issue
  - Student *proves code is incomplete or has multiple major issues (but able to explain this thoroughly) with test(s) and screenshot(s) and correctly identifies the missing code and/or specific, major issues*

- **No credit (0/3) if:**
  - Explaining the design of the program:
    - Student explains the process of writing the assembly program, *but it is off topic or hard to follow*
  - Proving correct operation of the program:
    - Student *cannot prove the functionality of the code*
    - Student's code is incomplete or has multiple major issues, *but does not identify the missing code and/or specific, major issues*
    - Student's *code not applicable to this assignment*
  - Student *commits academic dishonesty*

Analysis:

Carefully look at the loop you created in your program to process the characters, compare this to your loop from programming assignment 3, and discuss the following questions in your report:

- Compare your ARM code for the loop to process the characters in programming assignment 3 to your Intel x86 code for the loop to process the characters in this assignment. **What are the primary similarities and differences between your ARM code and your Intel x86 code?**
  - How do these similarities and differences relate to the general similarities/differences between RISC and CISC architectures?

between RISC and CISC architectures?
  - Which assembly code was easier to write, your ARM code or your Intel x86 code? Please explain.
  - Which assembly code do you think is more efficient, your ARM code or your Intel x86 code? Please explain.
- Assume that you execute your code on the following 5-stage pipeline: **IF-ID-MEM-EXE-WB**; note, when compared to our 5-stage pipeline from lecture, that **this pipeline has switched the order of the memory and execution stages**. Closely study your Intel x86 code for the loop to process the characters. Please complete the following:
  - Indicate all **data hazards** present in your loop (if any)
  - Indicate all instances of **data forwarding** (if any)
  - Based on this quick analysis, **how many clock cycles** are required to complete one iteration of your loop? **Remember to count the branch delay slot**; assume that this pipeline will evaluate the branch in the ID stage. **Also remember** to count the number of clock cycles from the IF stage of your first instruction of the first iteration of your loop up to, but not including, the IF stage of the second iteration of your loop.

Scoring for ARM vs. Intel x86 Analysis (3 points possible)

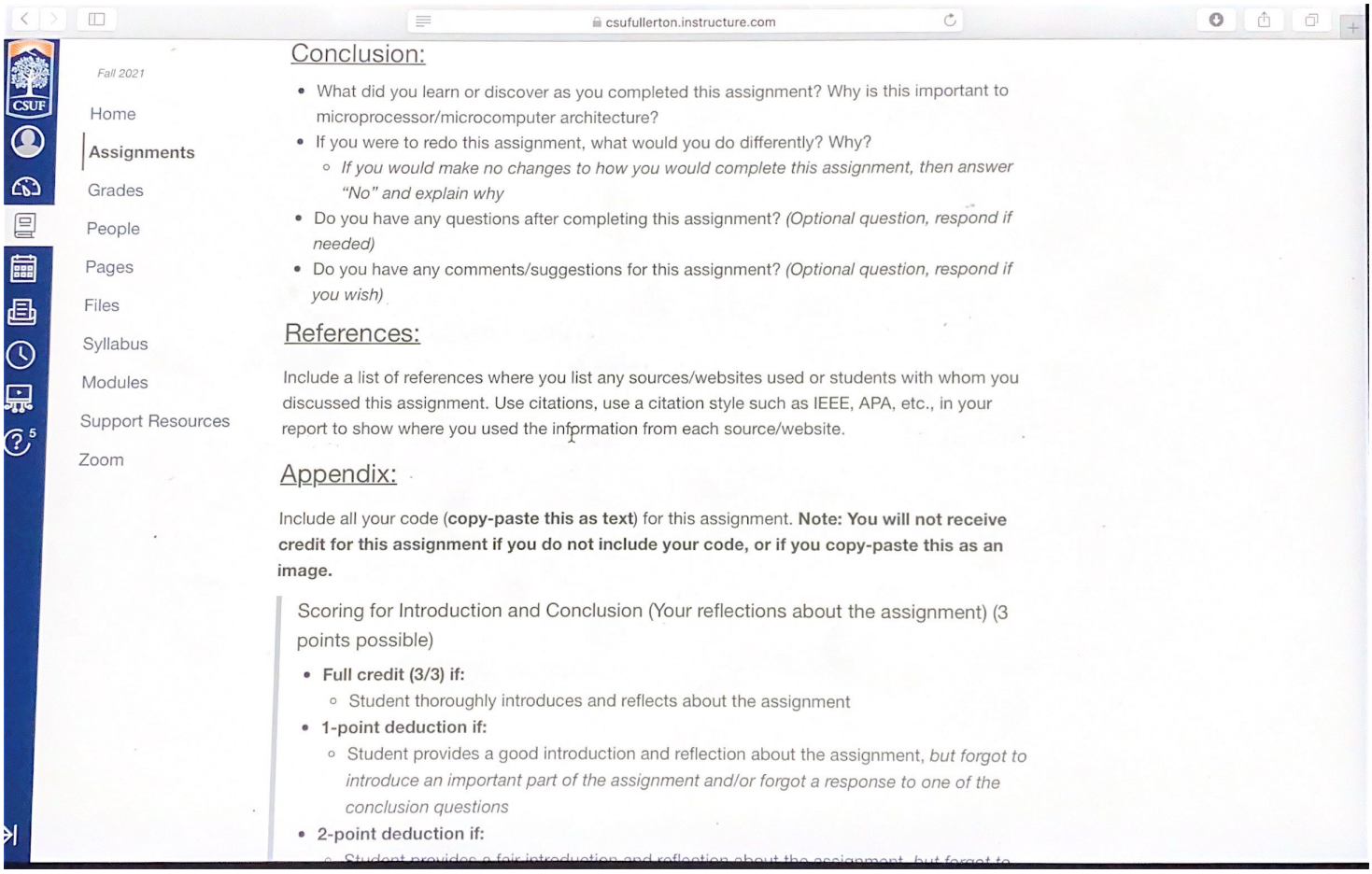
- **Full credit (3/3) if:**
  - Student thoroughly explains the primary similarities and differences of the code in the loop, properly relates these to differences between RISC and CISC, and thoroughly examines the ease of writing code and efficiency of the ARM vs. Intel x86 code
- **1-point deduction for each of the following:**
  - Student provides a good analysis of ARM vs. Intel x86, *but missed an important similarity/difference in the code in the loop, did not relate these to RISC vs. CISC, or did not comment on the ease of writing code or efficiency of ARM vs. Intel x86 code*
- **2-point deduction for each of the following:**
  - Student provides a fair analysis of ARM vs. Intel x86, *but missed multiple important similarities/differences in the code in the loop, did not relate these to RISC vs. CISC, and/or did not comment on the ease of writing code or efficiency of ARM vs. Intel x86 code*
- **No credit (0/3) if:**

- Student provides an analysis of ARM vs. Intel x86, *but it is off topic or hard to follow*
- Student *commits academic dishonesty*

Scoring for Analysis of Data Hazards, Data Forwarding, and Clock Cycle Count for Loop (3 points possible)

- **Full credit (3/3) if:**
  - Student correctly determines the clock cycle count, or determines this with a very minor error, and indicates all data hazards and data forwarding used in the loop
- **1-point deduction for each of the following:**
  - Student determines the clock cycle count and indicates data hazards and data forwarding, *but there are noticeable errors with one or two stalls/instances of forwarding*
  - Student determines the clock cycle count and indicates data hazards and data forwarding, *but the clock cycle count does not seem to follow the correct data hazards and data forwarding*
- **2-point deduction for each of the following:**
  - Student determines the clock cycle count and indicates data hazards and data forwarding, *but there are noticeable errors with three or four stalls/instances of forwarding*
  - Student determines the clock cycle count and indicates data hazards and data forwarding, *but the clock cycle count does not seem to follow the data hazards and data forwarding (and there are one or two errors with the stalls/instances of forwarding)*
- **No credit (0/3) if:**
  - Student determines the clock cycle count and indicates data hazards and data forwarding, *but there are noticeable errors with five or more stalls/instances of forwarding*
  - Student determines the clock cycle count and indicates data hazards and data forwarding, *but the clock cycle count does not seem to follow the data hazards and data forwarding (and there are three or four errors with the stalls/instances of forwarding)*
  - Student's *work is incomplete*
  - Student *commits academic dishonesty*

Conclusion:


csufullerton.instructure.com
Fall 2021
Home
Assignments
Grades
People
Pages
Files
Syllabus
Modules
Support Resources
Zoom
Conclusion:
What did you learn or discover as you completed this assignment? Why is this important to microprocessor/microcomputer architecture?
If you were to redo this assignment, what would you do differently? Why?
If you would make no changes to how you would complete this assignment, then answer "No" and explain why
Do you have any questions after completing this assignment? (Optional question, respond if needed)
Do you have any comments/suggestions for this assignment? (Optional question, respond if you wish)
References:
Include a list of references where you list any sources/websites used or students with whom you discussed this assignment. Use citations, use a citation style such as IEEE, APA, etc., in your report to show where you used the information from each source/website.
Appendix:
Include all your code (copy-paste this as text) for this assignment. Note: You will not receive credit for this assignment if you do not include your code, or if you copy-paste this as an image.
Scoring for Introduction and Conclusion (Your reflections about the assignment) (3 points possible)
Full credit (3/3) if:
Student thoroughly introduces and reflects about the assignment
1-point deduction if:
Student provides a good introduction and reflection about the assignment, but forgot to introduce an important part of the assignment and/or forgot a response to one of the conclusion questions
2-point deduction if:

Include a list of references where you list any sources/websites used or students with whom you discussed this assignment. Use citations, use a citation style such as IEEE, APA, etc., in your report to show where you used the information from each source/website.

Appendix:

Include all your code (**copy-paste this as text**) for this assignment. **Note: You will not receive credit for this assignment if you do not include your code, or if you copy-paste this as an image.**

> Scoring for Introduction and Conclusion (Your reflections about the assignment) (3 points possible)
>
> - **Full credit (3/3) if:**
> - Student thoroughly introduces and reflects about the assignment
> - **1-point deduction if:**
> - Student provides a good introduction and reflection about the assignment, *but forgot to introduce an important part of the assignment and/or forgot a response to one of the conclusion questions*
> - **2-point deduction if:**
> - Student provides a fair introduction and reflection about the assignment, *but forgot to introduce multiple important parts of the assignment and/or forgot responses to multiple conclusion questions*
> - **No credit (0/3) if:**
> - Student provides an introduction and/or conclusion, *but it is off topic or does not substantially discuss the assignment*
> - Student *commits academic dishonesty*