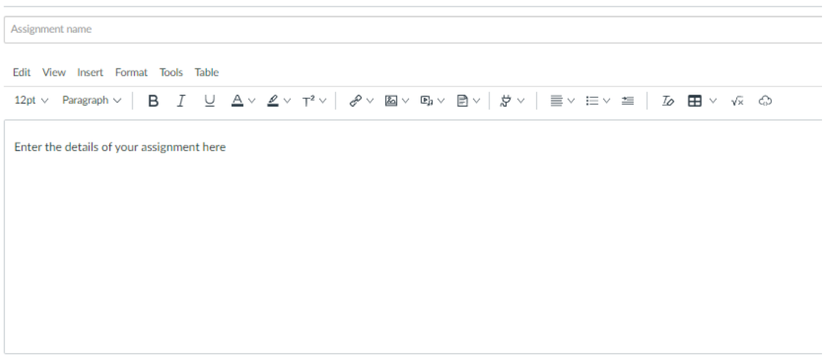
University of Hertfordshire UH
School of Physics, Engineering and Computer Science

Assignment briefings (2023/24)

All assignment briefings and marking scheme/rubrics must be published to students via the Assignments section on Canvas, using the Assignments description area.

What must be included in the Assignments description area.

As before, all assignments must have a name (title).Additionally, the following information must be included in the Assignments 'box':


Assignment name
Edit View Insert Format Tools Table
12pt Paragraph
Enter the details of your assignment here

Weighting %:	40%	**Submission deadline (for students):**	
Authorship:	Group	**Target date for returning marked coursework:**	
Tutor setting the work:	Miss Nur Diana	**Number of hours you are expected to work on this assignment:**	70 hours

This Assignment assesses the following module Learning Outcomes (from the Definitive Module Document):
1. Formulate assembly language programs that makes use of various data structures.

Assignment Tasks: Assembly Language Project for a Smart Home Automation System
In this project, students will design and implement a smart sensor system for a Smart Home Automation System prototype using Assembly language. The project focuses on programming the embedded system controller responsible for managing the interaction between sensors (for lighting, security, and temperature control) and a central control unit. This project will serve as a foundation for transitioning to higher-level programming languages, such as C, for the development of the complete embedded system.
The project is divided into the following stages:
A. Machine Instruction and Data Allocation (20% Weightage):
1. Introduction to Assembler:
    a. Provide a brief overview of the assembler and its role in converting assembly code to machine code.
    b. Discuss the importance of writing efficient assembly code for embedded systems in Smart Home Automation.
2. Machine Instructions:
    a. Introduce 0-, 1-, and 2-register instructions in the context of managing lighting, security, and temperature sensors.
    b. b. Allocate memory for sensor data storage and declare data segments for sensor readings (e.g., light levels, motion detection, and temperature).
3. Arithmetic and Logic Instructions:
    a. Implement arithmetic and logic operations to process sensor inputs (e.g., calculating average temperature or detecting motion)
    b. Demonstrate the use of flags and comparisons for decision-making (e.g., turning on lights when motion is detected).

4. Memory Operations and Loading Constants:
    a. Perform memory operations to store and retrieve sensor data dynamically.
    b. Load constants for calibration and threshold values for various sensors (e.g., brightness levels or temperature limits).

B. Logic Structures Using Macros (10% Weightage):

1. Comparisons and Flags:
    a. Write macros for comparing sensor readings to predefined thresholds (e.g., light intensity for dimming or brightening).
    b. Use flags to trigger automated actions, such as activating the security alarm.
2. If-Then-Else and Loop Structures:
    a. Implement If-Then-Else structures to control device behavior based on sensor conditions.
    b. Develop loop structures for continuous monitoring of the sensors in real time.

C. Introduction to Subroutines (10% Weightage):

1. Concept of Subroutines:
    a. Explain the concept of subroutines in the context of modular programming for Smart Home systems.
    b. Discuss how subroutines improve code organization, reusability, and system scalability.
2. Subroutines Calling and Interaction:
    a. Write subroutines for handling specific sensor actions, such as adjusting temperature or enabling security cameras.
    b. Demonstrate interactions between subroutines for seamless operation of the smart sensor system.

D. Data Structures (10% Weightage):

1. Array and Stack Operations:
    a. Use arrays to store multiple readings from temperature and light sensors.
    b. Apply stack operations for efficient management of sensor data during processing
2. Stack Memory and Parameter Passing:
    a. Leverage stack memory to manage subroutine calls and pass parameters (e.g., current temperature or motion status).
    b. Demonstrate effective parameter passing for subroutine communication.
3. Addressing Memory via the Stack and Heap Memory:
    a. Use stack memory for temporary storage of sensor data.
    b. Introduce heap memory for handling dynamic memory allocation in the Smart Home system.

E. Linked Data Structures and Recursion (10% Weightage):

1. **Linked Data Structures:**
    a. Implement linked data structures to manage interconnected sensor networks (e.g., hierarchical arrangement of sensors).
    b. Showcase the advantages of linked structures for scalability and complexity management.
2. **Recursion:**
    a. Use recursion to handle repetitive tasks, such as scanning sensors across multiple rooms.
    b. Discuss scenarios where recursion simplifies complex data processing tasks.

F. Final Demonstration (40% Weightage):

1. **Integrate and Demonstrate:**
    a. Integrate all project components into a functioning smart sensor system for the Smart Home Automation prototype.
    b. Present a demonstration highlighting the use of machine instructions, macros, subroutines, and data structures.
2. **Documentation and Reflection:**
    a. Submit comprehensive documentation of the code, including comments and explanations of functionality.
    b. Provide a reflective report on challenges faced, lessons learned, and how the system could be improved for real-world applications.

Submission Requirements:

A. Must be in a group

1. Refer to your localized college for grouping.
2. Max 3 person per group

B. Must include:

1. Initial Assemble Language Code
2. C Program
3. Report
    a. Introduction
    b. Table of Contents
    c. List of Figure (If Applicable)
    d. List of Table (If Applicable)
    e. All required components mentioned in Assignment Tasks
    f. References
4. Presentation Slides

Marks awarded for:
Please Refer to the next page for the details of marking schemes/rubrics

Type of Feedback to be given for this assignment:
Students demonstrated a clear understanding by successfully utilizing various data structures in assembly language programs; however, there is room for improvement in optimizing code efficiency and enhancing error handling.

Additional information:

- Regulations governing assessment offences including Plagiarism and Collusion are available from https://www.herts.ac.uk/__data/assets/pdf_file/0007/237625/AS14-Apx3-Academic-Misconduct.pdf (UPR AS14).
- Guidance on avoiding plagiarism can be found here: https://herts.instructure.com/courses/61421 (see the **Referencing** section) z
- For **undergraduate modules**:
    - a score of 40% or above represents a pass performance at honors level.
    - late submission of any item of coursework for each day or part thereof (or for hard copy submission only, working day or part thereof) for up to five days after the published deadline, coursework relating to modules at Levels 0, 4, 5, 6 submitted late (including deferred coursework, but except referred coursework), will have the numeric grade reduced by 10-grade points until or unless the numeric grade reaches or is 40. Where the numeric grade awarded for the assessment is less than 40, no lateness penalty will be applied.

University of Hertfordshire UH

School of Physics, Engineering and Computer Science

Group Name :			
Student's Name :	1.	Student's ID :	1.
	2.		2.
	3.		3.

PROJECT'S RUBRIC

Criteria	Excellent (5)	Good (4)	Satisfactory (3)	Needs Improvement (2)	Inadequate (1)	Marks Earned
Machine Instructions & Data Allocation (20%)	Successfully implements 0-, 1-, and 2-reg instructions for the sensor system.	Successfully implements 0-, 1-, and 2-reg instructions for the sensor system with minor issues.	Partially implements 0-, 1-, and 2-reg instructions for the sensor system with noticeable issues.	Incomplete implementation of 0-, 1-, and 2-reg instructions for the sensor system with significant issues	Fails to implement 0-, 1-, and 2-reg instructions for the sensor system.	
	Efficiently allocates space for data and declares content.	Adequately allocates space for data and declares content.	Allocates space for data and declares content adequately.	Limited allocation of space for data and declaration of content.	Inadequate allocation of space for data and declaration of content.	
	Demonstrates advanced arithmetic and logic instructions for processing sensor data	Demonstrates good arithmetic and logic instructions for processing sensor data.	Demonstrates basic arithmetic and logic instructions for processing sensor data.	Demonstrates basic arithmetic and logic instructions for processing sensor data.	Struggles with basic arithmetic and logic instructions for processing sensor data.	
	Effectively utilizes memory operations and loading constants.	Utilizes memory operations and loading constants effectively.	Utilizes memory operations and loading constants satisfactorily.	Utilizes memory operations and loading constants with minimal	Minimal utilization of memory operations and loading constants.	
Logic Structures Using Macros (10%)	Creates highly efficient macros for comparisons based on sensor thresholds.	Creates good macros for comparisons based on sensor thresholds.	Creates basic macros for comparisons based on sensor thresholds.	Limited development of macros for comparisons based on sensor thresholds.	Fails to create macros for comparisons based on sensor thresholds.	
	Implements If-Then-Else structures with precision.	Implements If-Then-Else structures effectively.	Implements If-Then-Else structures satisfactorily.	Struggles with If-Then-Else structure implementation.	Unable to implement If-Then-Else structures.	
	Develops loop structures for continuous sensor monitoring effectively.	Develops loop structures for continuous sensor monitoring adequately.	Attempts loop structures for continuous sensor monitoring.	Limited attempt at loop structures for continuous sensor monitoring.	No attempt at loop structures for continuous sensor monitoring.	
Introduction to Subroutines (10%)	Clearly explains the concept of subroutines in modular programming.	Adequately explains the concept of subroutines in modular programming.	Partially explains the concept of subroutines in modular programming.	Incomplete explanation of the concept of subroutines in modular programming.	Fails to explain the concept of subroutines in modular programming.	

	Successfully implements subroutines for specific sensor tasks.	Successfully implements subroutines for specific sensor tasks with minor issues.	Partially implements subroutines for specific sensor tasks with noticeable issues.	Struggles with implementing subroutines for specific sensor tasks.	Unable to implement subroutines for specific sensor tasks.	
	Demonstrates effective calling and interaction between subroutines.	Demonstrates good calling and interaction between subroutines.	Demonstrates basic calling and interaction between subroutines.	Limited calling and interaction between subroutines	No effective calling and interaction between subroutines.	
Data Structures (10%)	Successfully utilizes arrays for storing multiple sensor readings.	Utilizes arrays for storing multiple sensor readings effectively.	Partially utilizes arrays for storing multiple sensor readings. parameter passing	Limited use of arrays for storing multiple sensor readings.	Fails to utilize arrays for storing multiple sensor readings.	
	Implements stack operations effectively for data management.	Implements stack operations for data management adequately.	Attempts stack operations for data management with noticeable issues.	Struggles with stack operations for data management.	Unable to implement stack operations for data management.	
	Demonstrates efficient parameter passing via stack.	Demonstrates good parameter passing via stack.	Demonstrates basic parameter passing via stack.	Inadequate parameter passing via stack.	No effective parameter passing via stack.	
	Addresses memory via the stack for dynamic data management.	Addresses memory via the stack satisfactorily.	Limited addressing of memory via the stack.	Minimal addressing of memory via the stack	Fails to address memory via the stack.	
Linked Data Structures & Recursion (10%)	Successfully implements linked data structures for advanced sensor data organization.	Implements linked data structures for advanced sensor data organization with minor issues.	Partially implements linked data structures for sensor data organization.	Limited implementation of linked data structures for sensor data organization.	Fails to implement linked data structures for sensor data organization.	
	Demonstrates effective use of recursion in sensor system tasks.	Demonstrates good use of recursion in sensor system tasks.	Attempts recursion in sensor system tasks with noticeable issues.	Struggles with recursion in sensor system tasks.	Unable to utilize recursion in sensor system tasks.	
Final Demonstration & Documentation (40%)	Integrates all components for a fully functional smart sensor system.	Integrates all components for a functional smart sensor system with minor issues.	Partially integrates components for a smart sensor system with noticeable issues.	Limited integration of components for a smart sensor system with significant issues.	Fails to integrate components for a smart sensor system.	
	Demonstrates exceptional documentation with thorough comments and explanations.	Provides good documentation with clear comments and explanations.	Documentation is satisfactory with basic comments and explanations.	Documentation is minimal with limited comments and explanations.	No effective documentation provided.	

	Reflects on challenges faced, lessons learned, and improvements made during the project.	Reflects on challenges faced, lessons learned, and improvements made during the project adequately.	Provides basic reflection on challenges faced, lessons learned, and improvements made during the project.	Provides minimal reflection on challenges faced, lessons learned, and improvements made during the project.	Lacks reflection on challenges faced, lessons learned, and improvements made during the project.	
Comment					**Total Marks (100%):**	
					Final Marks (40%):	

Remarks:

1. Total Marks = (Machine Instructions & Data Allocation * 4) + (Logic Structures Using Macros * 2) + (Introduction to Subroutines * 2) + (Data Structures * 2) + (Linked Data Structures & Recursion * 2) + (Final Demonstration & Documentation *8)
2. Final Marks = (Total Marks / 100) * 40