COMP1001
Computer Systems

20 CREDIT MODULE

ASSESSMENT: 100% Coursework **W1: 30% Set Exercises**
W2: 70% Report

MODULE LEADER: Dr. Vasilios Kelefouras

MODULE AIMS

This module provides students with an underpinning knowledge of how computers work. Topics include low-level systems and representation of data, operating systems, and an introduction to subjects such as virtualisation, parallelism, state and communications. Students will learn how operating systems manage processes and scheduling, how memory management works and how software interacts with hardware.

ASSESSED LEARNING OUTCOMES (ALO):

1. Identify the functionality provided by an operating system and describe how each part works.
2. Explain the way in which data and processes are represented at the machine level and interact with hardware.
3. Outline strategies for achieving parallelism, resource allocation and scheduling within an operating system.

Overview

This document contains all the necessary information pertaining to the assessment of *COMP1001 Computer Systems*. The module is assessed via **100% coursework**, across two elements: *30% Set of Exercises* and *70% Report*.

The sections that follow will detail the assessment tasks that are to be undertaken. The submission and expected feedback dates are presented in Table 1. All assessments are to be submitted electronically via the respective DLE module pages before the stated deadlines.

	Submission Deadline	Feedback
Report Outline (30%)	**3rd November 2025 (15:00)**	**1st December 2024 (15:00)**
Report (70%)	**9th January 2026 (15:00)**	**6th February 2025 (15:00)**

Table 1: Assessment Deadlines

All assessments will be introduced in class to provide further clarity over what is expected and how you can access support and formative feedback prior to submission. **Whilst the assessment information is provided at the start of the module, it is not necessarily expected you will start this immediately** – as you will often not have sufficient understanding of the topic. The module leader will provide guidance in this respect.

Moderation

This assignment brief has been moderated in line with the university policy.

Moderator Name	**Moderation Date**
Muhammad Asad	16th Sept. 2025

Useful Assignment Information

Please see below some useful information regarding submissions for your modules.

Good Coding Practices

Like all things, no code is perfect. Just because your code compiles and runs does not mean it is perfect, there is always room for improvement. No code will achieve 100% of the available marks.
Where code submission is required by a module, it will be assessed against the following criteria:

1) **Functionality**
    a. Does your code meet all the specified requirements of the assignment?
    b. Does your code behave correctly across typical and edge-case scenarios?
    c. Does you code have appropriate error handling that does not leading to it crashing or undefined behaviour?
2) **Efficiency**
    a. Is your code optimised in terms of performance and resource use?
    b. Does you code handle functions, variables, data calls efficiently?
    c. Is there any unnecessary repetition, complexity or processing reducing efficiency within your code?
3) **Readability**
    a. Is your code logically structured and easy to read?
    b. Have you maintained standard formatting conventions like indentations, spacing and naming, within your code?
    c. Are variables, functions, and classes clearly named and purposeful?
4) **Documentation**
    a. Are there helpful comments explaining non-obvious parts of your code?
    b. Do your comments document your process, development or rationale clearly?
    c. Could someone unfamiliar with the code understand the approach you have taken?

Use of Generative AI for Creating Code

Each assignment element for each module will have a clear indication of the permitted level of use of AI (solo, assisted or partnered) including the generation of code. Please ensure that you **read and**

understand the permitted use. If you are unsure of a particular use, please reach out to the Module Leader and ask.

We strongly encourage you to read and explore beyond the core content delivered within the modules. However, this must be done correctly following academic process. Wherever code is drawn from an outside source (e.g. you have not written it yourself), regardless of whether this is AI generated or from an online repository (GitHub, Stack Overflow etc) **you must reference the original source**. This can be done in your documentation as comments or part of your write ups. Students found to be utilising code without referencing could face an academic offence. **If in doubt speak to your Module Leader.**

Versioning

A critical part of a development (software or academic) is versioning. Keeping a number of iterations over the course of development ensures you always have backups to fall back on. It also provides clear demonstration of the development process you implemented.

Using online/cloud-based storage solutions and repositories provide additional peace of mind, alongside being industry practice. The university provide OneDrive to all students which offers inbuilt version history for Microsoft products. GitHub is the university recommended repository for versioning code, which includes great integration with a number of IDEs.

Submitting Code

It is vital that you confirm that all code that you submit is in the **correct format** and **compiles correctly** or **runs without error**. If you clean up your code before submission, please ensure that all dependencies (libraries, functions and variables) are included so that the marker can compile your code.

We can only mark what has been submitted. If the code does not run correctly, it is not our responsibility to spend time error handling to award you marks.

It is also important to **show your working**. Tidying up your code is a critical part of coding practices, but if you remove the workings that provides you with the required output, we cannot see how you got there.

In a worst-case scenario, if you remove a key part of your working, and on compiling your code it gives an error or different result, we have no way of confirming what went wrong, or indeed if you fabricated those outputs.

Please note, just because a piece of code has given the wrong output, it does not mean it should be deleted. Seeing these attempts with comments documenting your attempt allows us to understand where you may have made an error and provide feedback that addresses this. In some cases, you could also be awarded marks for the attempt.

Acceptable level of generative AI tool use in this assessment element
The acceptable level of GenAI use for this element is detailed with the allowed uses listed below. This is split into three categories (**Solo Work** – work must be your own with no AI support, **Assisted Work** – some uses of AI tools allowed, **Partnered Work** – AI tools integral part of the work). If you have any questions, please contact the module leader.

Solo Work	You must not use generative AI tools.	X
Assisted Work	You are permitted to use generative AI tools in an assistive role.	☐
Partnered Work	Generative AI tool use is required as an integral part of the assessment, but transparency is required.	☐

Table 2: Acceptable Level of generative AI use

You **must include a signed Generative AI Declaration as an appendix to your submission**. The declaration form can be found on DLE (or Here on the Programme Page). This form will not be included in any word count associated with this assignment.

Assessment 1: Set Exercises (30%)

Description

This set of exercises consists of a) an online questionnaire on the DLE page of the module (40 Marks), b) two coding questions (60 Marks).

1. Online questionnaire [40 Marks]. The questions will be comparable to those in the weekly module worksheets. They will be randomly selected from a database of questions. Some questions will be multiple choice, some require calculations, and some require textual input. It is recommended to have paper and pencil at hand when attempting the questions. However, students will not need to upload workings. A calculator may also be useful. The Windows calculator in scientific mode would be sufficient. To prepare, it is recommended that students do the weekly worksheets and study the solutions provided on the DLE. **The questionnaire will open 29/10/2025 at 15:00 and close 31/10/2025 at 15:00**. **Students have 2 hours to finish the questionnaire after they started it.** They must finish before the deadline (31/10/2025 at 15:00).

   Extenuating Circumstances for the online Questionnaire: You can only apply for extenuating circumstances before taking the questionnaire. If you start the questionnaire this will count as an attempt, and you will receive marks according to your performance. Only one attempt is allowed. You will not be able to claim for ECs after you have started.

2. Create an assembly program that does the following [15 Marks].
   a. It initializes an array of four unsigned integer values with values : 10, 20, 30, 40.
   b. It loads the last element of the array
   c. It multiplies the element by 3 using mul command and then it divides the result by 6 using div command.
   d. It stores the result into the last element of the array.

The marking scheme is as follows:

Marks	0-1	2-6	7-12	13-15

Marking Criteria	The code gives errors.	The student has provided an implementation that does not store the right value in memory. The student has not used mul/div instructions but other variants.	The student has provided an implementation that stores the right value in memory but contains bad practice, issues or bugs.	The student has provided an excellent implementation.

3. Convert the following C code into assembly code [45 Marks]. Do not simplify the code. The assembly code must be a) provided as a separate .asm file and b) included in the delivered .docx file [45 marks]
Tip. You have been taught how to use integer division only. So, implement the division using 'div' instruction; assume that the reminder is always zero (we are interested in the quotient only).

```
unsigned int i, B[10], A[10]={3,2,3,1,7,5,0,8,9,2}, C[10]={1,3,2,5,4,6,0,4,5,8};

for (i=0; i<10; i++){
  B[i] = (A[i]*2+1) + (C[i]*3+1) + (A[i]+C[i])/3;
 }
```

Marks	**0-9**	**10-25**	**26-35**	**36-45**
Marking Criteria	The student has provided an implementation that does not generate the right output. The code gives errors.	The student has provided an implementation that generates the right output but contains bad practice or bugs. The student has not used mul/div instructions but other variants.	The student has provided an excellent implementation using DWORD data type for the arrays.	The student has provided an outstanding implementation. The program uses the minimum amount of memory and the minimum number of assembly instructions. The arrays contain 8-bit elements.

Submission Details

You should submit **two files**:

1. **q1.asm file** containing the assembly code of question 1.

2. **q2.asm file** containing the assembly code of question 2.

Assessment 2: Report (70%)

Description

This element of assessment consists of three questions. The source code needed is provided in the 'coursework' directory of the module's Github repository https://github.com/kelefouras/COMP1001/tree/newversion/COMP1001-master/25_26_coursework/Report .

Question 1 – Develop an Image Processing Application [40 Marks]

In the question1 folder, you will find the C code for an image processing application. This code reads a specific image from the *input_images* folder (which contains multiple images) and generates one output image stored in the *output_images* folder. The output image is the result of applying edge detection. Edge detection is important in image processing because it helps in identifying and highlighting the boundaries or edges of objects within an image. See the output image to better understand the effect of edge detection. If you are unable to open the output images, you can use GIMP, a free image editing software.

In the provided C code, the image paths are fixed and stored in the IN, and OUT macros. Your tasks are as follows.

A. Amend the edge_detection() function to address the following issue. The border pixels of the output image are not currently processed. This happens because the current implementation skips the outer rows and columns, as it applies a 3×3 convolution that requires access to neighbouring pixels. To correctly compute the output values at the image borders, you must account for pixels where the 3×3 filter would extend beyond the image boundaries. Since actual image data are not available outside the bounds, you should assume that these out-of-bounds values are zero; this approach is known as zero padding. Your task is to update the convolution logic so that it processes every pixel in the image, including borders. For each pixel, if a neighbour accessed by the filter falls outside the image bounds, substitute a value of zero. You must use if statements to check whether the current neighbourhood indices are within the valid image range. [10 Marks]

B. Modify the provided program to process all the images in the input_images folder. Given that there are 31 input images, 31 output images should be generated and stored in the output folder. You should take into account the fact that all the input images have the same name followed by an index. Since the input images vary in size, you should handle multiple images by either dynamically allocating arrays (recommended) or statically defining separate arrays for each image (not recommended). To process multiple images, you can either use a loop to iterate through all the images (recommended) or create multiple copies of the existing code (not recommended). All solutions are acceptable, but your grade will be based on the efficiency of your implementation. **[30 Marks]**

   You need to use some of the following functions '*sprintf*', '*strcat*', 'memset', 'malloc', 'calloc', 'realloc' and 'free'. No other libraries should be used; you need to use C language code, not C++ or any C++ libraries. **Solutions that do not meet these requirements will be marked with zero grade.**

   The marking criteria are as follows:

Marks	**0-1**	**2-5**	**6-17**	**18-20**	**21-30**

Marking criteria	The code does not process more than one image. The code is not functional. Code does not compile (gives errors). The solution provided does not include the suggested libraries.	The code successfully processes some images but not all. The approach used to process multiple images does not use a loop. The arrays are defined statically.	The arrays are allocated/deallocated dynamically by using malloc/free functions. The approach used to process multiple images does not use a loop.	An efficient solution is provided. The code processes all the images by using a loop. Dynamic allocation is used using *malloc.*	An excellent solution is provided. Dynamic allocation is used using *realloc*. No redundant code is provided (new routines have been created where necessary).

Question 2 – Vectorize simple loop kernels [40 Marks]

Download the q1.cpp file. Your task is to vectorize the *q1(), q2() and q3() r*outines by using ***x86-64 SSE/SSE2/SSE4/AVX/AVX2*** C/C++ intrinsics. This is a popular technique to improve the performance of software. You will re-write the above functions using one of the above technologies. Your program must work for any input size value (any 'N' value). All the C/C++ x86-64 intrinsics are provided in the following link: https://software.intel.com/sites/landingpage/IntrinsicsGuide/ .

A. Vectorize 'q1()' routine. The code should be written in a new routine 'q1_vec()'. The marking scheme is as follows **[10 marks]**:

Question 1B. marks	**0-2 marks**	**3-5 marks**	**6-8 marks**	**9-10 marks**
Marking criteria	The student has not used the instrinsics appropriately. The code contains bad practice. The routine does not generate the right output.	The code does not work properly for any input size (N).	An efficient implementation is provided. A separate routine is developed to verify the correctness of the results.	An excellent implementation is provided. fmadd instruction is used.

Hint: There are many different ways to implement this routine and each solution includes different intrinsics. However, a valid solution exists using the following instructions: *_mm_loadu_ps, _mm_set1_ps, _mm_add_ps, _mm_mul_ps, _mm_storeu_ps.*

B. Vectorize 'q2()' routine. The code should be written in a new routine 'q2_vec'. The marking scheme is as follows **[15 marks]**:

Question 1B. marks	**0-2 marks**	**3-6 marks**	**7-12 marks**	**13-15 marks**
Marking criteria	The student has not used the instrinsics appropriately. The code contains bad practice. The routine does not generate the right output.	The code does not work properly for any input size (N).	An efficient implementation is provided. A separate routine is developed to verify the correctness of the results.	An excellent implementation is provided. fmadd instruction is used.

Hint: A valid solution exists using the following instructions: *_mm_loadu_ps, _mm_load_ps1, _mm_add_ps, _mm_mul_ps, _mm_storeu_ps, _mm_set1_ps, _mm_store_ss,* _mm_fmadd_ps, *_mm_set_ps.*

C. Vectorize 'q3()' routine. The code should be written in a new routine 'q3_vec'. The marking scheme is as follows **[15 marks]**:

Question 1B. marks	**0-2 marks**	**3-6 marks**	**7-12 marks**	**13-15 marks**
Marking criteria	The student has not used the instrinsics appropriately. The code contains bad practice. The routine does not generate the right output.	The code does not work properly for any input size (N).	An efficient implementation is provided. A separate routine is developed to verify the correctness of the results.	An excellent implementation is provided. AVX technology is used, not SSE. fmadd instruction is used.

Hint: A valid solution exists using the following instructions: *_mm_loadu_ps, _mm_load_ps1, _mm_add_ps, _mm_mul_ps, _mm_storeu_ps, _mm_set1_ps, _mm_store_ss, _mm_hadd_ps,* _mm_fmadd_ps, *_mm_set_ps.*

Question 3 – Linux Bash Script and Performance Analysis [20 Marks]

You are provided with a C program that performs parallel Matrix-Matrix Multiplication (MMM). Your task is to conduct a performance analysis of this program in a Linux environment.

A. You are required to modify the provided C code so that the matrix size N and the number of threads Num_Threads are not hardcoded in the source file. Instead, these values should be passed to the program as command-line arguments. To achieve this, you should define the main() function as main(int argc, char *argv[]). You must also create a Linux bash script that compiles the C program and runs the resulting binary. The script should call the executable multiple times, once for each of the following values of N: 64, 128, 256, 512, 1024, and 2048. During each run, the bash script should pass the chosen N value and a value for Num_Threads as arguments to the executable. The script should also display a custom message at the start and end of the execution.

You are also required to modify the C program to measure the execution time of the MMM() function and calculate its performance in terms of FLOPs (Floating Point Operations per Second). The FLOPs should be calculated using the formula FLOPs = $(2 \times N^3)$ / Execution Time, where the execution time is measured in seconds. The program should print both the execution time and the calculated FLOP/s value to the terminal for each run.

The marking scheme is as follows **[12 marks]**:

Question 3A. marks	**0-1 marks**	**2-9 marks**	**10-12**
Marking criteria	Neither the script nor the c code is properly developed.	Either the script or the c code is properly developed, but not both. FLOPs is well calcuated.	An excellent implementation is provided.

B. You are required to plot two graphs showing FLOPs versus N. One graph should correspond to the case where Num_Threads = 1, and the other to the case where Num_Threads = 4. You may use Microsoft Excel, or any similar tool to create the graphs. Along with the graphs, you must submit the modified C source code, and the bash script used to generate the data (Question 3A). To ensure the accuracy of your performance measurements, you must verify that the MMM() routine runs for at least 3 seconds when using a single thread, and at least 30 seconds when using four threads. If the routine completes too quickly, you should modify the program to run the MMM computation multiple times in a loop and report the average execution time. The timing should be implemented using the omp_get_wtime() function, as demonstrated in the lab sessions.

The marking scheme is as follows **[8 marks]**:

Question 3B. marks	**0-1 marks**	**2-6 marks**	**7-8**
Marking criteria	The graphs do not contain the right values. The experimental procedure is not carried out properly. The source code is not submitted.	The graphs still have some wrong values. The experimental procedure includes minor flaws.	An excellent graph is provided with clear labels and units of measurements.

Submission Details

The submission will be done via the submission link on the COMP1001 DLE page. You should submit **the following files (PLEASE DO NOT UPLOAD ANY ZIP FILES)**:

- **q1.cpp** or **q1.c** file containing the source code of question 1.
- **q2.cpp or q2.c** file containing the answer to question 2.
- **q3.c** and **q3.sh** files containing the source code of question 3.
- **q3.pdf** file containing the answer of question 3B.

General Guidance

Extenuating Circumstances

There may be a time during this module where you experience a serious situation which has a significant impact on your ability to complete the assessments. The definition of these can be found in the University Policy on Extenuating Circumstances here:
https://www.plymouth.ac.uk/uploads/production/document/path/15/15317/Extenuating_Circumstances_Policy_and_Procedures.pdf

Plagiarism

All of your work must be of your own words. You must use references for your sources, however you acquire them. Where you wish to use quotations, these must be a very minor part of your overall work.

To copy another person's work is viewed as plagiarism and is not allowed. Any issues of plagiarism and any form of academic dishonesty are treated very seriously. All your work must be your own and other sources must be identified as being theirs, not yours. The copying of another persons' work could result in a penalty being invoked.

Further information on plagiarism policy can be found here:

Plagiarism: https://www.plymouth.ac.uk/student-life/your-studies/essential-information/regulations/plagiarism

Examination Offences: https://www.plymouth.ac.uk/student-life/your-studies/essential-information/exams/exam-rules-and-regulations/examination-offences

Turnitin (http://www.turnitinuk.com/) is an Internet-based 'originality checking tool' which allows documents to be compared with content on the Internet, in journals and in an archive of previously submitted works. It can help to detect unintentional or deliberate plagiarism.

It is a formative tool that makes it easy for students to review their citations and referencing as an aid to learning good academic practice. Turnitin produces an 'originality report' to help guide you. To learn more about Turnitin go to:
https://guides.turnitin.com/01_Manuals_and_Guides/Student/Student_User_Manual

Referencing
The University of Plymouth Library has produced an online support referencing guide which is available here: http://plymouth.libguides.com/referencing.

Another recommended referencing resource is Cite Them Right Online; this is an online resource which provides you with specific guidance about how to reference lots of different types of materials.

The Learn Higher Network has also provided a number of documents to support students with referencing:
References and Bibliographies Booklet:
http://www.learnhigher.ac.uk/writing-for-university/referencing/references-and-bibliographies-booklet/
Checking your assignments' references:
http://www.learnhigher.ac.uk/writing-for-university/academic-writing/checking-your-assigments-references/