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 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
Set Exercises (30%)	**22nd Nov. at 15.00**	Within 20 working days
Report (70%)	**17th Jan. at 15.00**	Within 20 working days

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.

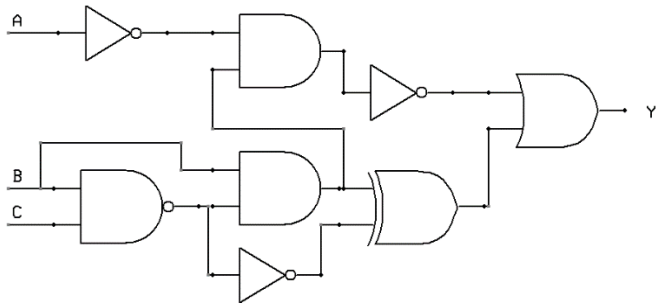
Assessment 1: Set Exercises (30%)

Description

This set of exercises consists of a number of questions. Please provide concise answers. **Answers without explanation (where needed) will be marked with zero.**

1. A. Write the Boolean expression of the following circuit diagram [2 marks].

   B. Set up the truth table [9 marks]



2. A. Given the two following decimal numbers : 37, -126

i) Represent the two numbers in One's Complement representation (using 8-bit binary in the result) [4 marks]
ii) Represent the two numbers in Two's Complement representation (using 8-bit binary in the result) [6 marks]

   B. Convert the positive number N=1010000001001 in single precision floating point format [3 marks]

3. If main memory is of 4 giga bytes and every word is of 2 bytes how many bits do we need to address any single word in memory? [4 marks]

4. How much RAM memory can a 16-bit CPU can use? Provide your answer in bytes. [4 marks]

5. Consider a 7-stage pipelined CPU where every stage is 30nsecs. How much time does it take to execute 1000 CPU instructions if no stall cycles occur? Provide the answer in nsecs. [5 marks]

6. Let a a CPU with CPU frequency 2 GHz. It executed 1.5 million instructions in 6 million CPU clock cycles.

   A. What is the average value of the CPI for this CPU? [4 marks]

   B. How long did it take to complete all the instructions? [3 marks]

7. A CPU has an average CPI of 2.5. It took 1.6 seconds to execute 9.6 million instructions. What is the speed of this CPU? [6 marks]

8. Convert the following C code into assembly code. 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 [50 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).

```
void main(){
unsigned int i, B[10], A[10]={3,2,3,1,7,5,0,8,9,2};
for (i=0; i<10; i++){
  B[i] = A[i] + 2 * (2*i + ( (3*i+1) / 5) );
  }
}
```

Marks	0-9	10-19	20-35	36-50
Marking Criteria	The student has provided an implementation that does not generate the right output.	The student has provided an implementation that generates the right output, but contains bad practice or bugs.	The student has provided an efficient implementation.	The student has provided an outstanding implementation. The program uses the minimum amount of memory.

Submission Details

You should submit **two files**:

1. A **.pdf file** containing the answers to all the questions above (including the assembly code for question 8). Do **not** submit word files.
2. An **.asm file** containing the assembly code of question 8.

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/23_24_coursework/Report .

Question 1 [40 Marks]

Download the q1.cpp file (see module's GitHub repository). Your task is to vectorize the *'routine1( )'* and *'routine2( )'* routines by using ***x86-64 SSE/SSE2/SSE4/AVX/AVX2*** C/C++ intrinsics. You will re-write these two functions using one of the above technologies and save them into *'routine1_vec( )'* and *'routine2_vec( )'* new routines, respectively. Your program must work for any input size value (any 'N, M' values). All the C/C++ x86-64 intrinsics are provided in the following link: https://software.intel.com/sites/landingpage/IntrinsicsGuide/ .

The marking scheme is as follows **[40 marks]**:

Question 1. marks	**0-2 marks**	**3-11 marks**	**12-15 marks**	**16-25 marks**	**26-40 marks**
Marking criteria	The student has not used the instrinsics appropriately. The code contains bad practice. The routine does not generate the right output.	Just one loop kernel is vectorised. The code does not works properly for any input size.	Just one loop kernel is vectorised. The code works properly for any input size. The student has not developed appropriate routines to check the correctness of his/her routines.	Just one loop kernel is vectorised. The student has developed routines to check the correctness of his/her routines.	Two loop kernels are vectorised. The code works properly for any input size. The student has developed routines to check the correctness of his/her routines.

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_load_ps1, _mm_add_ps, _mm_sub_ps, _mm_mul_ps, _mm_storeu_ps, _mm_set1_ps, _mm_store_ss, _mm_hadd_ps, _mm_set_ps.*

Question 2 [10 Marks]

Below part of a C code is provided. Provide the answers to the following questions.

- How many processes does this program include? **[3 marks]**
- How many processes print the message '*This is the End (The Doors)'* ? **[3 marks]**
- What messages will be printed in the screen when this program run? **[4 marks]**

Marks	0	1-3	4
Marking Criteria	The output messages provided are wrong.	Some of the output messages provided are wrong, but most of the messages are correct. The answer provided is very close to the right one.	The student has provided the right output messages in the right order.

```
int main() {

printf("\nMain started \n");

if (fork()==0)
  funct2();
else if (fork()==0)
  funct1();

printf("\nThis is the End (The Doors) \n");

exit(EXIT_SUCCESS);
}

void funct1() {

fork();

execlp("echo", "echo", "Cheers", "from our world !", (char*)0);
perror("execlp");
exit(EXIT_FAILURE);
}

void funct2() {

fork();
printf("\nHi from Funct2");
}
```

Question 3 [50 Marks]

Under the '*question3*' folder you will find the C code of an image processing application. This code reads a specific image from the '*input_images*' folder (this folder contains many images) and creates two new images which are stored into the '*output_images*' folder; the two output images are the result of the Gaussian Blur and Sobel popular algorithms, respectively. The first algorithm reduces the noise from the image while the second prints its edges (applies edge detection). In the C code provided, the image paths are fixed and they are stored into '*IN, OUT, OUT2*' macros.

A. In this task you will amend the software application provided so as the user can provide the image paths from a Linux bash script. To this end, you need to both write a Linux bash script

and also amend the C code provided (this task will be carried out in Linux). The script should compile and run the C program; the image paths should be passed as inputs to the C program's main function as command line arguments; this will be realized by defining the main function as '*int main(int argc, char *argv[ ])*'. **[7 Marks]**

B. Amend the given program to process **all** the images in the '*input_images*' folder. In this task you will **not** use a bash script and thus you can carry this task out in any programming environment, e.g., Visual Studio. Note that the input images are of different size and therefore to process multiple images you need either to statically define separate arrays for each image (this is not recommended), or to allocate the arrays dynamically (good practice). Furthermore, you can process multiple images by either creating multiple copies of the code being provided (not recommended) or by using a loop (good practice). All the solutions are acceptable but your mark will be calculated based on how efficient your implementation is. If you choose **not** to use a loop, the code will be very long and thus I would suggest to process just three images not all of them (your mark will be the same regardless of the number of the images processed in this case). **[31 Marks]**
The marking criteria are as follows:

Marks	**0-1**	**2-5**	**6-15**	**16-31**
Marking criteria	The code does not process more than one images. The code is not functional. Code that does not compile (code with errors) will be marked with zero.	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 excellent solution is provided. The code processes all the images by using a loop. Dynamic allocation is used.

Important: *Based on the marking scheme above, if the code does not generate the right output or does not compile you be marked with zero. Thus, it is preferable to deliver an inefficient implementation that works rather than an advanced implementation (e.g., using dynamic allocation and/or loops) that does not work properly.*

C. Rewrite the '*Sobel()*' routine using x86-64 SSE/SSE2/SSE4/AVX/AVX2 C/C++ intrinsics. **[12 Marks].**

The marking criteria are as follows:

Marks	0-1	2-6	7-12
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.	An excellent implementation is provided.

Tip. Fully unroll the two innermost loops and use the 'maddubs_epi16' and 'add_epi16' instructions.

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.pdf file** containing the answers of question 2. Do **not** submit word files.
- **q3a.c** and **q3a.sh** files containing the source code of question 3A.
- **q3bc.cpp or q3bc.c** file containing the answer of question 3B and 3C.

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/