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%)	**1st Dec. 2022 3pm**	Within 20 working days
Report (70%)	**24th Jan. 2023 3pm**	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. None of the questions needs a long answer.

1. A. Draw the circuit diagram of the following expression F=ABC + AB' (A' C')' [3 marks].
   B. Set up the truth table [8 marks]

2. A. What decimal value does the 8-bit binary number 11011111 have if

i) it is interpreted as an unsigned number? [1 mark]
ii) it is on a computer using signed-magnitude representation? [2 mark]
iii) it is on a computer using one's complement representation? [3 marks]
iv) it is on a computer using two's complement representation? [3 marks]

   B. Convert the positive number N=1010000001011 in single precision floating point format [3 marks]

3. If main memory is of 64 Mbytes and every word is of 2 bytes how many bits do we need to address any single word in memory? [6 marks]

4. How much RAM memory can a 16-bit, 32bit and 64-bit CPU can use? Provide your answer in bytes. [6 marks]

5. Consider a 6-stage pipelined CPU where every stage is 40nsecs. How much time does it take to execute 200 CPU instructions if no stall cycles occur? Provide the answer in nsecs. [5 marks]

6. Consider that the CPU clock rate is 2.2 MHz and the Program takes 1.2 million cycles to execute. What's the CPU time (provide the answer in seconds)? [5 marks]

7. Consider that a CPU supports 130 different instructions. How many bits are needed for the instruction's opcode? [5 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, A[10]={3,2,3,1,7,5,7,8,9,2};
for (i=0; i<10; i++){
  A[i] += 3 * (i + ( (2*i+1) / 3) );
  }
}
```

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 **.docx file** containing the answers of the questions above (including the assembly code for question 8).
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/COURSEWORK).

1. Download the q1.cpp file from the module's GitHub page.
   A. Amend the above program so as to print the FLOPs (Floating Point Operations per Second) value achieved by the q1() routine. The FLOPs value is given by the following formula $FLOPS = (2xN^2) / Ex.Time$, where N is the input size and Ex.Time is the execution time of the q1() routine **[5 Marks]**.
   B. Make a graph of *Ex.Time vs N* and a graph of *FLOPs vs N*, where N=[100, 400, 800, 1600, 3000, 6000, 8000, 10000, 12000]. The FLOPs and Ex.Time values must be shown in the y-axis while the N value must be shown in the x-axis **[5 Marks]**.
   C. Explain the two graphs in task B (in no more than 3 lines). For example, why the FLOPs value is decreased as N increases? Why the Ex.Time does not increase in a linear way? **[10 Marks]**
   D. Amend this program to allocate all the arrays dynamically, by using malloc() function. This should be implemented in a separate routine **[10 Marks]**
   E. Write a script in Linux (.sh file) which automates the experimental procedure of step B **[10 Marks]**. The script should do the following:
      - *Print a message such as : This is a script that …*
      - *Compile the q1.cpp program and generate the binary file (executable)*
      - *Run the executable by passing the 'N' and 'Iteration' values as input to the executable, e.g., ./exec 100 20.*
      - *Repeat the step above for all 'N' and 'Iteration' values (do not use a loop)*

      Note that question 1.E requires that q1.cpp should be compiled and run in Linux. To do so, you need to comment the lines #15 and #48 in q1.cpp file.

2. Below part of a C code is provided. Provide the answers to the following questions.
   - How many processes does this program include? **[2 marks]**
   - How many processes execute the printf() commands? **[5 marks]**
   - What messages will be printed in the screen when this program run? **[13 marks]**

Marks	0-3	4-7	8-13
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() {
 for (int i=0; i<2; i++){
  fork();
printf("\nFunct2 executes i=%d\n",i);

   }
}
```

3. Download the q3.cpp file (see module's GitHub repository). Your task is to convert the *'slow_routine()'* C routine to assembly, by using ***x86-64 SSE/SSE2/SSE4/AVX/AVX2*** C/C++ intrinsics. You will re-write this function using the above technologies and save it into 'q3_vectorized*()*' routine. 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/ .

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

Question 3. marks	0-2 marks	3-8 marks	9-20 marks	21-30 marks	31-40 marks
Marking criteria	The student has not provided an appropriate vectorised code. The routine does not generate the right output.	Just one loop kernel is vectorised. The implementation does not contain bad practice and works properly for any input size. The appropriate instructions have been used.	Two loop kernels are vectorised. The implementation does not contain bad practice and works properly for any input size. The appropriate instructions have been used.	Three loop kernels are vectorised. The implementation does not contain bad practice and works properly for any input size. The appropriate instructions have been used.	All the four loop kernels are vectorised. The implementation does not contain bad practice and works properly for any input size. The appropriate instructions have been used.

Hints: 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_mul_ps, _mm_storeu_ps, _mm_set1_ps, _mm_store_ss, _mm_hadd_ps, _mm_set_ps.*

Submission Details

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

- **q1.cpp** file containing the source code of question 1A, 1D and 1E
- **q1.sh** file containing the answer of question 1E
- A **.docx file** containing the answers of questions 1B, 1C and 2.
- **q3.cpp** file containing the source code of question 3.

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/