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%)	**23rd of Nov. at 15.00**	By 15th of Dec.
Report (70%)	**20th of Jan. at 15.00**	By 1st of Febr.

Table 1: Assessment Deadlines

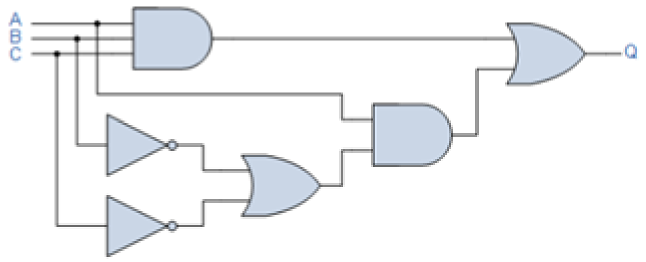
All assessments will be introduced in class to provide further clarity over what is expected and how you can access support and formative feedback prior to submission. Whilst the assessment information is provided at the start of the module, it is not necessarily expected you will start this immediately – 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. Write the Boolean expression of the following circuit diagram [1 marks].
   B. Set up the truth table [4 marks]



2. A. What decimal value does the 8-bit binary number 11011011 have if

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

B. Convert the positive number N=1011000001001 in single precision floating point format [2 marks]

3. Consider a main memory of 32 entries (thus, the memory addresses are from 0 to 31) and a cache memory of 8 entries. Also consider that the CPU loads the following memory addresses '0, 1, 2, 8, 9, 2'. How many cache hits occur when the cache is
   a) direct mapped [5 marks]
   b) 2-way associative [5 marks]

4. If main memory is of 64Mbytes and every word is of 2 bytes how many bits do we need to address any single word in memory? [5 marks]

5. Consider a 7-stage pipelined CPU where every stage is 50nsecs. How much time does it take to execute 1000 CPU instructions if no stall cycles occur? Provide the answer in nsecs [10 marks]

6. Consider that the CPU clock rate is 1 MHz and the Program takes 1 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. A. Convert the following C code into assembly code. The assembly code must be a) provided as a separate .asm file and b) included in the delivered .docx file [25 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). Alternatively, the division of a number which is a power of two can be implemented using a shift instruction, which is simpler, faster and easier to use.

```
void main(){
int i, A[10];
for (i=0; i< 10; i++){
  A[i]=(2*i+7)/2;
  }
}
```

Marks	0-5	6-15	16-25
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 bags.	The student has provided an outstanding implementation.

B. In Fig.1 the hardware architecture of Intel Skylake CPU is shown. A brief explanation is provided in https://en.wikichip.org/wiki/intel/microarchitectures/skylake_(client). Consider that the assembly code you wrote above will run on this CPU. According to what you have learned so far, briefly explain how your assembly program will run on this CPU. Which CPU parts will be involved for running your program? Ignore the hardware components that we have not studied in this module.

The marking scheme is as follows:

- How data are loaded from main memory to the CPU? Your answer must not exceed 6 lines (font 11). Longer answers will not be marked. [10 marks]

Marks	**0-5**	**6-10**
Marking Criteria	Little or no analysis is provided. Little understanding of the subject. The explanation provided is not accurate.	The student has appropriately explained how data are loaded/stored from/to main memory to the CPU. The student has appropriately explained how memory hierarchy works.

- Assuming that the assembly instructions are located in main memory, what are the main steps the CPU will apply to execute the program? Your answer must not exceed 8 lines (font 11). Longer answers will not be marked. [15 marks]

Marks	**0-6**	**7-15**
Marking Criteria	Little or no analysis is provided. Little understanding of the subject. The explanation provided is not accurate.	The student has briefly but appropriately explained a) how the code is loaded from main memory, b) the instruction decode process, c) the instruction execution process. The student has made clear whether the instructions are executed sequentially or in parallel, whether the execution is in-order or out-of-order etc.

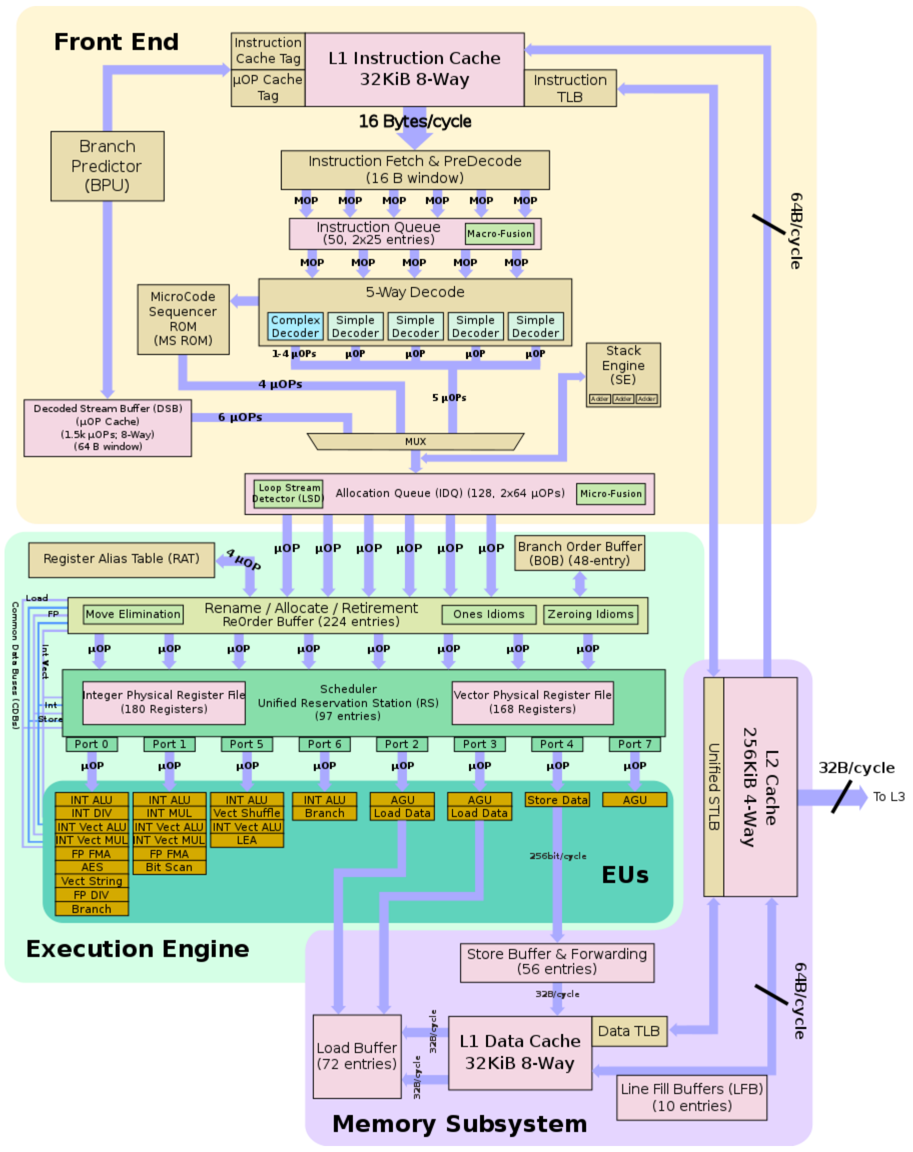


Fig.1 Intel Skylake hardware architecture

Submission Details

You should submit **two files**:

1. A **.docx file** containing the answers of the questions above (including the assembly code for question 8A).
2. An **.asm file** containing the assembly code of question 8A.

Assessment 2: Report (70%)

Description

This element of assessment consists of the following questions

1. Consider the following C routine and two different single core CPUs. Both CPUs support two levels of cache; a unified L2 cache, an L1 data cache and an L1 instruction cache. The first CPU has an L1 data cache of size 32kbytes, while the 2nd has an L1 data cache of size 16kbytes. Which CPU will run this routine faster and why? Assume that only this process runs on the CPU. You answer must be no longer than 3 lines. Longer answers will not be marked **[15 marks]**.

```
int X[8000];

void routine() {
int i,j;

for (i=1; i<1000; i++)
  for (j=0; j<8000; j++){
   X[j]+=i;
   }
}
```

Marks	0-7	8-15
Marking Criteria	Little or no analysis is provided. Little understanding of the subject. The explanation provided is not accurate.	The student has briefly but appropriately explained which CPU is the most appropriate for this code. The text provided is well-written.

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() command? **[5 marks]**
   - Draw a graph, where each node refers to a process and each edge refers to process creation (the graph of a single fork() command is given in Fig.1). **[13 marks]**

```
int main() {

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

 printf("\nThis is the End\n");
 exit(0);
}

void funct1() {
    execlp("echo", "echo", "Cheers", "from child !", (char*)0);
```

```
        perror("execlp");
        exit(1);
    }

     void funct2() {
     for (int i=0; i<2; i++){
      fork();
       }
    }
```

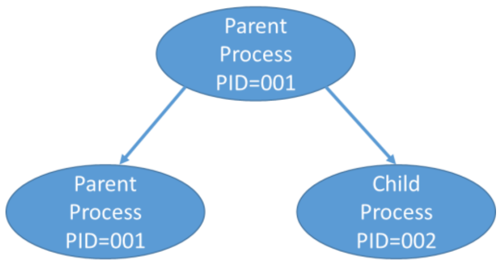


Fig.1 Fork graph.

Marks	**0-3**	**4-7**	**8-13**
Marking Criteria	The graph provided is wrong. Several nodes are missing.	The graph provided is wrong, but most of the processes are included. The answer provided is very close to the right one.	The student has provided the right graph.

3. Download the '*coursework_code_to_start.zip*' file. This file contains two .cpp and one .h file. Create a new C++ project in visual studio using these files.

   a. In this program, the 'V1', 'V2' and 'test' arrays are allocated statically. Amend the initialization() routine so as the 'V1', 'V2' and 'test' arrays are allocated dynamically, using malloc function. **[15 marks]**

Marks	**0-7**	**8-15**
Marking Criteria	The code does not run or contains bad practice.	Memory is appropriately allocated/deallocated. The student has addressed the case where memory cannot be allocated.

   b. Implement the *'default_routine()* routine using ***x86-64 SSE/SSE2/SSE4* C intrinsics**. You will re-write this function using SSE technology and save it into '*SSE()*' routine. Your program must work for any input size value (any 'M' value). You will use the following intrinsics . All the C x86-64 intrinsics are provided in the following link: https://software.intel.com/sites/landingpage/IntrinsicsGuide/ .
   The marking scheme is as follows **[25 marks]**:

Question 3c. marks	**0-5 marks**	**6-10 marks**	**11-15 marks**	**16-25 marks**
Question 3c. marking criteria	The student has not provided appropriate vectorised code. The routine is not fully vectorised. The routine does not generate the right output.	The routine is fully vectorised. However, the implementation contains bad practice.	The routine is fully vectorised. The implementation does not contain bad practice. However, the implementation does not work properly for any input size.	The routine is fully vectorised. The implementation does not contain bad practice and works properly for any input size.

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

4. Download the coursework_pthreads.c file. Using ***pthreads***, parallelize the '*gemver_default()*'routine into four threads. This task can be done **on Linux only**. If you do not have Linux installed on your PCs, you can use the school's PCs remotely. The marking scheme is as follows **[25 marks]**:

Question 3d. marks	**0-4 marks**	**5-9 marks**	**10-15 marks**	**16-25 marks**
Question 3d. marking criteria	The student has not provided appropriate parallel code. The routine does not generate the right output.	The routine is parallelized. The routine generates the right output. However, there is bad practice.	The routine is well parallelized. The implementation does not contain bad practice. However, the implementation does not work properly for any input size.	The routine is well parallelized. The implementation does not contain bad practice. The implementation works properly for any input size.

Submission Details

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

- A **.docx file** containing the answers of questions 1 and 2.
- The **main.cpp, coursework.cpp** and **coursework.h** files containing the source code of question 3.
- The **coursework_pthreads.c** file containing the source code of question 4.

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/