COURSEWORK SPECIFICATION
ECM1413 - Computers and the Internet
Module Leader: Abdelkhalik Mosa
Academic Year: 2025/26

Title: Continuous Assessment
Submission deadline: 2026-02-19

This assessment contributes **30%** of the total module mark and assesses the following **intended learning outcomes**:

- Understand basics of Linux terminal
- Demonstrate understanding of computer architecture
- Apply basic understanding of pipelined execution

This is an **individual assessment**, and you are reminded of the University's regulations on collaboration and plagiarism. **You must avoid plagiarism, collusion, and any academic misconduct behaviours**. Further details about academic honesty and plagiarism can be found at `https://ele.exeter.ac.uk/course/view.php?id=1957`.

Use of GenAI Tools

The University of Exeter is committed to the ethical and responsible use of Generative AI (GenAI) tools in teaching and learning, in line with our academic integrity policies where the direct copying of AI-generated content is included under plagiarism, misrepresentation and contract cheating under definitions and offences in *TQA Manual Chapter 12.3*. This assessment falls under the **category of AI-minimal** in the University's Guidance on use of GenAI in Assessment.

Coursework Instructions

You need to submit a single compressed file (.zip) named `ECM1413_coursework.zip`. This compressed file must contain the following two files:

1. A PDF document named `ECM1413_coursework.pdf`, containing solutions to all tasks in Part 1 and Part 2 below.

2. `transcript.txt`, the transcript for your full terminal session completing the tasks in Part 1.

Part 1: The Linux Terminal [Marks: 50]

To record the transcript needed for the submission, follow the following steps:

Step 1: Start the transcript

Open a Linux terminal and type:

```
script transcript.txt
```

This will record all the commands you type and the terminal output.

Step 2: Complete the exercises

Complete the tasks below in the terminal. All commands and their outputs will be recorded in `transcript.txt`.

Step 3: Stop the transcript

When you have finished all tasks, type:

```
exit
```

This will finalize `transcript.txt`.

Step 4: Collect the transcript

If you record the transcript in the virtual machine, you can transfer it to your local machine by connecting via RDP. Once connected, you can either attach `transcript.txt` to an email sent to yourself or save it to a cloud storage service accessible from your local machine.

Note: For Tasks 1–4, you must clearly indicate any directory changes required to complete the task (e.g. by including the relevant `cd` commands). Do not assume the marker knows your current working directory.

Tasks

Task 1 [Marks: 4]

Create the following directory structure in your home directory (`/home/student` if you are using the VM):

```
projects/
  project1/
  project2/
```

Task 2 [Marks: 4]

Create an empty file `empty.txt` in `project1` and a separate empty file `empty.txt` in `project2`.

Task 3 [Marks: 4]

Rename `empty.txt` in `project2` to `empty2.txt` and move it to `projects/project1`. Confirm the move by listing all files in the destination folder.

Task 4 [Marks: 4]

In your home directory, create a file called `quotes.txt` containing four lines of text of your choice. Verify that the file has been created correctly by displaying its contents in the terminal.

Task 5 [Marks: 4]

Using a single command, create a copy of `quotes.txt` named `quotes_copy.txt` in the same directory. Verify that the new file contains the same contents by displaying it in the terminal.

Task 6 [Marks: 6]

Use `!!` to re-run the last command from the previous question, and combine it with output redirection to double the number of lines in `quotes.txt`. Use the `wc` command to verify that the number of lines in `quotes.txt` has doubled.

Task 7 [Marks: 6]

Save a listing of the directory contents in your home directory, including hidden files, into a hidden file named `.hidden.txt` using output redirection. Verify that the file exists and display its contents.

Task 8 [Marks: 5]

Set the permissions of `quotes.txt` so that only the owner can read and write, the group can read, and others have no permissions. Verify the permissions using an appropriate command.

Task 9 [Marks: 6]

Using a single command, create a new file `quotes_novowels.txt` containing the contents of `quotes.txt` with all upper-case and lower-case vowels removed.

Task 10 [Marks: 7]

Append an upper-case version of the contents of `quotes_copy.txt` to the end of the file `quotes.txt` using a single command. Verify that the file has been modified.

Part 2: Architecture [Marks: 50]

Prerequisites

Students are expected to have prior understanding of:

- Five-stage pipelined processor (IF, ID, EX, MEM, WB)
- Pipeline hazards: data, control, and structural
- Hazard resolution techniques: stalling, forwarding, flushing

Processor Assumptions

Assume a 5-stage RISC pipeline with the following characteristics:

- Separate instruction and data memory
- Forwarding (also called bypassing) is available
- Data hazards introduce one stall cycle
- Branch decision is made in the EX stage
- Taken branches cause pipeline flushing

Instruction Sequence

```
1.  LW   R1, 0(R2) // Load word from memory into R1, value = 0
2.  ADD  R3, R1, R4 // R3 = R1 + R4
3.  SUB  R5, R3, R6 // R5 = R3 - R6
4.  LW   R7, 4(R3) // Load word using R3 as base
5.  AND  R8, R7, R1 // R8 = R7 AND R1
6.  BEQ  R8, R0, LABEL // Branch if R8 == 0
7.  ADD  R9, R5, R3 // R9 = R5 + R3
8.  OR   R10, R1, R8 // R10 = R1 OR R8

LABEL:
9.  SUB  R11, R9, R4 // R11 = R9 - R4
```

Task 1: Pipeline Execution Diagram [Marks: 10]

Draw a cycle-by-cycle pipeline execution diagram for instructions 1 through 8.

- Show all pipeline stages [Marks: 6]
- Clearly indicate stall cycles by mentioning STALL in the corresponding cycle [Marks: 2]
- Clearly indicate flushed instructions (if any) by mentioning FLUSH in the corresponding cycle [Marks: 2]

Task 2: Hazard Identification [Marks: 10]

Identify and explain all pipeline hazards present in the instruction sequence.

- Data hazards (RAW, WAR, WAW where applicable) [Marks: 4]
- Control hazards [Marks: 4]
- Specify which instructions are involved [Marks: 2]

Task 3: Hazard Resolution [Marks: 10]

For each identified hazard:

- Explain how it can be resolved [Marks: 5]
- State whether forwarding, stalling, or flushing can be used [Marks: 3]
- Mention the number of stall cycles introduced by the hazard [Marks: 2]

Task 4: Performance Analysis [Marks: 10]

- Calculate the number of clock cycles required to execute the given code [Marks: 3]
- Compute the CPI (Cycles Per Instruction) [Marks: 3]
- Briefly discuss the impact of hazards on performance [Marks: 4]

Task 5: Instruction Reordering [Marks: 10]

Reorder the instructions (without changing correctness of the program) to reduce pipeline stalls.

- Present the reordered instruction sequence [Marks: 5]
- Draw a cycle-by-cycle pipeline execution diagram for the reordered instructions 1 through 8 [Marks: 3]
- Explain how the reordering improves performance [Marks: 2]