Module	Computer Fundamentals / COMP1027 (CSF) / Semester 1
Module Convenor(s)	Tissa Chandesa

Assessment Name	Coursework 1	**Weight**	40%

Description and Deliverable(s)

This coursework has **THREE** parts, as detailed below:

Part 1 focuses on **Elementary Logic Gates**
Part 2 focuses on **Combinatorial & Sequential Circuits**
Part 3 focuses on **The Hack Computer**

Important Notes

1. For this coursework, **all tasks** are made as a **MINI GROUP WORK**. You are encouraged to divide the different tasks among yourselves, within the group. You should finish your tasks faster this way. However, you must maintain a healthy communication between yourselves to avoid conflict and delay in completing this coursework by the stipulated deadline. The effectiveness of your group will be assessed via the **peer assessment**.

2. You **MUST** maintain the provided folder structures. All files for Parts 1, 2 and 3 will be under THREE folders. Three additional sub-folders will be added for part 3 to reflect sections 3.1, 3.2 and 3.3. Please be observant to this and abide to the provided zip file as well as its folder and sub-folder structure and file names.

Files to Download

CW1-Files.zip provides all the skeleton .hdl and .asm (*as well as most of the related .tst and .cmp files*).

1. Download the zip file from the "Coursework 1" area on Moodle.
2. Extract the files and fill in the required HDL and Assembly codes, as per the detailed instructions given in this assessment sheet.

MAINTAIN the same folder structure, file name, and all test and compare files when you submit. You can add any additional files you may need to get your submissions working correctly.

Part 1 focuses on **Elementary Logic Gates**

Tessa is a talented baker who wishes to open her own café in 2025. In order to realise her dreams, she needs to secure a location whereby her café can be open, purchased relevant furniture, enlist several helpers, and prepare a set of delightful choices for the customers to choose from. For the short-term, she will be happy to successfully fulfil two to three of the four conditions, namely a location **and** furniture **or** menu.

1. Generate a **truth table** to represent the conditions that Tessa needs to fulfil to ensure her café is successfully operational by February 2025. Please represent location, furniture, helpers and menu as A, B, C, and D in your truth table. Output should be represented as F.

2. Based on the truth table in (1), write out the **pre-simplified** expression followed by the **simplified expression** using the Basic Identities of Boolean Algebra in Figure 1. **You are required to show your working details (Boolean Function, Boolean Expressions, Simplifications, …, etc.) in the** Truth Table and Simplification.docx**.**

Identity Name	AND Form	OR Form
Identity Law	$1x = x$	$0+x = x$
Null (or Dominance) Law	$0x = 0$	$1+x = 1$
Idempotent Law	$xx = x$	$x+x = x$
Inverse Law	$x\bar{x} = 0$	$x+\bar{x} = 1$
Commutative Law	$xy = yx$	$x+y = y+x$
Associative Law	$(xy)z = x(yz)$	$(x+y)+z = x+(y+z)$
Distributive Law	$x+yz = (x+y)(x+z)$	$x(y+z) = xy+xz$
Absorption Law	$x(x+y) = x$	$x+xy = x$
DeMorgan's Law	$\overline{(xy)} = \bar{x}+\bar{y}$	$\overline{(x+y)} = \bar{x}\bar{y}$
Double Complement Law	$\bar{\bar{x}} = x$	

Extracted from: L. Null, L. and J. Lobur, **The essentials of computer organisation and architecture**, Jones & Bartlett Publishers, 2003 (Fig. [illegible] P.26)

Figure 1

3. Based on your answers in (1) and (2), respectively, **implement** a **Tessa decision** chip that prioritise what she needs to address to ensure a timely opening of her café. **NOTE:** *to ensure that your test cases are aligned with our test cases, please only use 1-bit input.*

SUBMISSION: **SUBMIT *your* completed Tessa.hdl *file*.** (and all associated testing scripts & compare files) and **Truth Table and Simplication.pdf**. **NOTE**: we'll also use our own test versions during marking.

Part 2 focuses on **Combinatorial & Sequential Circuits**

1. **Implement an ALU** chip (MyALU), of your own design, that computes 20 functions (*the 18 functions covered in the lecture, in addition to the X **XOR** Y and X **XNOR** Y*)

2. Table 1 shows the functions (Output), as per the required order and the 5 control bits (**C4, C3, C2, C1, C0**) with some samples of their values (*you are required to complete them according to the Decimal values in the left column*).

3. The **.hdl** skeleton is provided, but you will need to **create your .tst and .cmp files** for this chip.

Table 1: MyALU Functions (Output)

Decimal of Cs	C4	C3	C2	C1	C0	Output
0	0	0	0	0	0	0
1						1
2						-1
3	0	0	0	1	1	X
4						Y
5	0	0	1	0	1	X'
6						Y'
7						-X
8						-Y
9						X+1
10						Y+1
11						X-1
12						Y-1
13						X+Y
14						X-Y
15						Y-X
16						X AND Y
17						X OR Y
18						X XOR Y
19						X XNOR Y

SUBMISSION: SUBMIT your answers, i.e., completed **MyALU.hdl** files. (And all associated testing scripts & compare files).

Part 3 focuses on **The Hack Computer**

Additional Information

- In your lab sessions, we used *combinatorial* and *sequential* logic circuits to construct many of the various core logic circuits that form the basis of a CPU and Memory.

- This part concludes all the effort by completing the construction of the HACK Computer, i.e., *putting it all together and have the HACK Computer working & executing instructions*.
  (i). **Firstly**, the *MEMORY* will be built according to the HACK architecture.
  (ii). **Secondly**, the *CPU* will combine the *ALU* and other circuits. But you'll need to develop the "*Control Unit*" that manages the data flow and execution of instructions.
  (iii). **Finally**, all are connected as one chip (the "*Computer*").

- You should go through the Lecture slides, Chapter 5, and Appendix A of the primary text *carefully*.

- More lab reading resources are available at www.nan2tetris.org/05.php. In particular, "Chapter 5" and its relevant appendices.

SECTION 3.1: Assembly Code

In section 3.1, you are required to provide two Assembly programs (for the HACK computer) performing the following tasks:

a) An Assembly code that does **Power**:

- Implement an Assembly program to calculate the exponential power of a given number ***m***, P(*m, e*).
- For example: if *m* = 2 and *e* = 5 then P(m, e) would be 2*2*2*2*2 = 32.
- The user should enter the value of the number *m* into R0, i.e., RAM[0] and *e* into R1, e.g., RAM[1].
- The result P(*m, e*) should be saved in RAM[2].
- **SPECIAL CASE**: In ne – if e is ZERO, your program should store 1 in RAM[0] and end the program.

b) An Assembly code that does **Factorial**:

- Implement an Assembly program to calculate the factorial of a given number, **n**, F(n). A factorial of a number is given by:

  F(n) = n*(n-1)*(n-2)*…*2*1

- The user should enter the value of the number **n** into **R0**, i.e., **RAM[0]**.
- The factorial result **F(n)** should be saved in **RAM[1]**.

SUBMISSION: SUBMIT your answers, i.e., completed **Power.asm** & **Factorial.asm** files. (And associated tst & cmp files).

SECTION 3.2: The HACK Computer

Task A: Memory

- **Implement the Memory Chip**.
  Hint: The specification for the memory chip is described in the lecture and Chapter 5. Note that you would have to use RAM16K, Screen and Keyboard

"parts" (built-in, but you should refresh yourself with its interface specifications in Chapter 5, for the latter two).

- **If you make no progress.** You need to understand what makes a memory chip – there is this need for addressing/selecting "memory banks". How do you select 3 outputs (RAM16K, Screen, Keyboard)? (Mux4Way*? DMux4Way*?). You still need to interpret/decode a 15-bit address and pass them through to the right "memory banks" (RAM16K, Screen or Keyboard). The memory addresses are already given to you (e.g., lecture slides, and Chapter 5). The "logic" for interpretation/decoding would make more sense once you convert them to binary. You would note that only certain address bits are crucial for selection.

SUBMISSION: SUBMIT your answers, i.e., completed **Memory.hdl** file (And associated tst & cmp files).

Task B: CPU

- **Implement the CPU chip.**
  Hint: This may be the most challenging task – so, give it ample attention. The CPU implementation (framework) is given in the lecture and Chapter 5, Section 5.3.1. In general, the CPU as a complex logical gate would fetch and execute instructions in their corresponding A- and C-Instruction codes (16-bits long).

- **If you make no progress.** Look at the Chip diagram of CPU implementation in the lecture and in Chapter 5. Adopt a divide-and-conquer approach, that is, try to solve the problem by parts. Use this skeleton and compare with the Chip diagram:

```
// Instruction decode
// Use a combination of elementary logical gates to decode the instructions
// You should first decode between A and C-Instructions,
// then the computation and destination
:
And (a=cInst, b=instruction[4], out=destD);
:
:

// A register and input mux
Mux??(…);
?? (…);
ARegister (…);

// D register
DRegister(in=aluOut, load=destD, out=dReg);

// ALU and input mux
Mux16 (…);
ALU (…);

// PC with jump test
// Use a combination of elementary logical gates to implement the truth table for
// jump functions, given in lectures.
// For example, try to figure out why one implementation would make use of:
// Or (a=jle, b=jgt, out=jump);
:
:
PC (in=aReg, reset=reset, inc=true, load=jump, out[0..14]=pc);
```

Note: the above pseudo-code is for illustration purposes only. It may **NOT** be complete/accurate for the HDL Simulator. You should only use it as a guide/hint.

SUBMISSION: SUBMIT your answers, i.e., completed **CPU.hdl** file. (And associated tst & cmp files). See **section 3.3** for details of additional tasks.

Task C: Computer

- **Implement the Computer chip.**
  Hint: This is easy and as a bonus only 3-lines of code:

  CPU (inM=??, instruction=??, reset=??, WriteM=??, outM=??, address=??, pc=??);
  Memory (in=??, load=??, address=??, out=??);
  ROM32K (address=??, out=??);

- The provided zip file contains a couple of test files (e.g**.,** *ComputerMax, ComputerAdd, ComputerRect*), to test your Computer chip. Make sure to utilise them and properly test your chip before submission.

SUBMISSION: SUBMIT your answers, i.e., completed **Computer.hdl** file. (And associated tst & cmp files).

SECTION 3.3: Circuitry Diagram and Justification

In **section 3.2** – **task B,** you would have designed a circuitry diagram prior to implementing the CPU chip. As such, you are required to submit your full **circuit diagram** of your CPU implementation, along with a 2-page summary justifying your circuitry design. **Focus** should be on part(s) of your CPU that you creatively designed and implemented, which are different to the default design we discussed in the lecture. Within folder PART 3 in the provided zip file contains a Word document (Justification.docx) for you to add your justification. Please make sure to fill your details (e.g., *name and studentID*) as well as your fellow group team members. Also, please indicate clearly which group are you from. **NOTE**: your justification should not exceed 2 pages - additional pages will incur a 5% marks penalty for each page. You may add **relevant** figures and tables to support your justification.

SUBMISSION: SUBMIT your answers, i.e., completed **Circuitry Diagram.pdf** and **Justification.pdf** files.

Final Submission Instructions

Each group will have **ONE** submission **ONLY** (performed by *one nominated member of the group*).

We reserve the right to ask all/some students to explain any/all of your submitted work at any time. Failure to explain it properly could affect YOUR mark.

Compress the whole folder structure/tree only (*see the provided zip file, for an example*). **Avoid** compressing each individual folder/file. Nested compression will prevent the marking process and could result in marking scripts failing. Should this occur to your submission, your entire coursework 1 mark **WILL** be awarded a 0%. So, please be careful!

The group submission must be submitted via Moodle as a **ZIP** archive file. **Any other archive file formats WILL result in a penalty of a 5%-mark deduction of your group's overall mark.** Within your ZIP archive, it should contain all the requested files (same folder structure as per the downloaded files). Name that ZIP file as **CSF-CW1-XXX.zip**, where the "XXX" is your Group Number (*i.e.,* 001, 020, 043). **Incorrect naming of the ZIP file WILL result in a penalty of a 10%-mark deduction of your group's overall mark**. So, your attention to detail is pivotal here.

All the files should have the (**EXACT**) file names, and arranged under the (**EXACT**) subfolder name/structure as detailed below:

PART 1:

Tessa.hdl Tessa.tst Tessa.cmp
Truth Table and Simplication.PDF

	PART 2: MyALU.hdl (And all associated testing scripts & compare files) **PART 3:** **3.1_Assembly:** Factorial.asm, Factorial.cmp, Factorial.tst, Power.asm, Power.cmp, Power.tst (All associated tst & cmp files) **3.2_HACK:** Add.hack, Computer.hdl, ComputerAdd.cmp, ComputerAdd.tst, ComputerMax.cmp, ComputerMax.tst, ComputerRect.cmp, ComputerRect.tst, CPU.cmp, CPU.hdl, CPU.tst, Max.hack, Memory.cmp, Memory.hdl, Memory.tst, Rect.hack (All associated tst & cmp files) **3.3_Circuit Diagram and Justification:** Circuit Diagram.PDF Justification.PDF On Moodle, please click on the "**Coursework 1**" link within the "Coursework" section to perform **your group's** submission. Additionally, each of you will need to make an additional submission, **separately** for your **peer assessment** form. Please click on the "**Peer Assessment**" link on Moodle to perform this submission.
Release Date	Monday, 21st October 2024
Submission Date	Friday, 22nd November 2024, by 11:59pm
Late Policy (University of Nottingham default will apply, if blank)	Work submitted after the deadline will be subject to a penalty of 5 marks (the standard 5% absolute) for each late **working day** out of the total 100 marks.
Feedback Mechanism and Date	Marks and written individual feedback will be returned via Moodle within the week commencing 23rd December 2024.
Assessment Criteria	**IMPORTANT NOTE:** 1. You are advised to **add relevant comments** to the code (in the .hdl files) to indicate your understanding of the implementation. As the implementation become more and more complex, those comments become extremely useful for you (especially when we ask you). 2. You are **allowed to use Built-in chips**, for chips/gates where relevant (*except any of those that are required, to be developed in this coursework 1 itself*).

3. Marking & checks will be done through scripts, which expects specific file name (case sensitive). Any deviation from that will results in error, therefore, **resulting in a loss of marks**.
4. All chips will be tested through automated testing scripts.
5. Missing any required file(s), using different formats, naming, …, etc. will attract **penalty of 10% of your overall mark**.
6. Any submitted work that explicitly exhibit any cutting corners/plagiarism…, etc will result in the **entire coursework 1 being awarded 0%.**
7. **Chips that are unable to run will result in 0% being awarded for that task**.

CW1 Assessment Breakdown:

Part 1 (15%):

Correct Truth Table	**5**
Correct Simplification	**5**
Chip structure & executes correctly	**5**

Part 2 (10%):

MyALU - Correct chip structure & executes correctly	**10**

Part 3:

3.1 – Assembly Code (10%):

Power Program	**5**
Factorial Program	**5**

3.2 – HACK Computer (25%):

Task A – Memory.hdl	**5**
Task B – CPU.hdl	**5**
Task C – Computer.hdl	
Pass "Add" test	**5**
Pass "Max" test	**5**
Pass "Rect" test	**5**

3.3 – Circuitry Diagram & Justification (40%):

Circuitry Design	**15**
Justification of Circuitry Design	**25**