University of Hertfordshire UH

School of Computer of Science

ASSIGNMENT BRIEFING SHEET (2016/17 Academic Year)

Assignment Title	Coco de Mere 8 (CdM8): Circuits and Programming with Logisim	**Submission Date**	9th January 2017
Module Title	Computer Architecture	**Module Code**	5COM1057
Tutor	**M L Walters**	**GROUP or INDIVIDUAL Assignment**	***Groups of <= 3***

GROUP ASSIGNMENT - *STUDENTS TO COMPLETE*

Group Name/Number (if allocated by module team)

(Student comments on this assignment can be made on the back of the assignment briefing sheet)

By completing **BOX B** below, we certify that the submission is entirely ours and that any material derived or quoted from the published or unpublished work of other persons has been duly acknowledged. **[ref. UPR AS/C/6.1, section 7 and UPR AS/C/5 (Appendix III)].)]**. We also certify, that any work with human participants has been carried out under an approved ethics protocol in accordance with UPR RE01

*Please print your forenames and surnames in capitals, provide your; - ID numbers, actual time spent on the assignment and your signatures. By signing the submission you certify that this work represents equal contributions from all team members. If this is not the case, the module leader **must** be informed before submission.*

BOX B

Student Forename *(in CAPS please)*	**Student Surname** *(in CAPS please)*	**Student ID Number (SRN)**	**Actual Time Spent by each Student (hours)**	**Signature of Student**

This sheet must be submitted with the assignment, signed and either BOX A or B filled in.

LATE SUBMISSION WILL ATTRACT A STANDARD LATENESS PENALTY.

1. For undergraduate modules, a score of 40% or above represents a pass mark.
2. For postgraduate modules, a score of 50% or above represents a pass mark.
3. For work submitted up to 5 working days late marked is capped to a bare pass (40% for undergraduate and

Regulations governing assessment offences including Plagiarism and Collusion are available from: http://sitem.herts.ac.uk/secreg/upr/pdf/AS14-Apx3-Assessment%20Offences-v06.0.pdf

School of Computer of Science

ASSIGNMENT BRIEFING SHEET (2016/17 Academic Year)

THE ASSIGNMENT TASK:

Using circuit simulation software (Logisim) produce a complete simulated digital computing system based on the Coco de Mere 8 bit (CDM8) microprocessor core.

MODULE LEARNING OUTCOMES ASSESSED BY THIS ASSIGNMENT:

- To be able to articulate and develop critical responses to the purpose of a range of components of modern computers, architectural features, and describe how they interact to form a functioning whole.
- To be able to apply their understanding of the principles and theories of processor and memory architecture to improve high program design and effciency

SUBMISSION REQUIREMENTS:

Logisim circuits and Assembly code listings to be submitted electronically via Studynet by the due date. Hard copy printouts of the Assignment Cover Sheet, (Logisim) circuit designs and program assembly code listings also submitted through Computer Science Reception by 3.00pm on the submission deadline date.

FEEDBACK FROM THIS ASSIGNMENT

Feedback and advice will be provided to groups on an ongoing basis during tutorial sessions, and there will be ample opportunity for student groups to discuss and receive feedback as they develop their circuit designs and associated software. Written specific and generic feedback will be provided after marking, and students can also arrange to meet with the tutor outside timetabled sessions for individual feedback and support.

MARKS AWARDED FROM 20 TOTAL FOR :

Circuit Design (10 marks), broken down as follows:

- 3 marks maximum – Two working (simulated) basic circuits (Tasks 1. and 2.) as described in the assignment brief.
- 2 marks maximum – Circuit design drawing clear to read and understand
- 5 marks maximum – Additional functionality, optimisation and error trapping etc. incorporated in the circuit design, including the advanced task (Task 4.) described in the assignment brief

Software Development (10 marks), broken down as follows:

- 3 marks maximum – Two working basic programs (Tasks 1. and 2.) are produced which satisfy the main requirements of the basic functionality described in the assignment brief.
- 2 marks maximum – Assembly program listings are well structured and clear to read.
- 5 marks maximum – Additional software functionality, optimisation techniques and/or error catching and handling, including the advanced task (Task 3.) described in the assignment brief

See the included **Generic Grading Criteria Sheet,** and the columns headed *Ideas/Concept Development*, *Application of Technology* and *Ideas/Concept Development* for more details on how these marks are allocated.

DEADLINES AND ASSIGNMENT WEIGHTINGS

1 This assignment is worth	20%	of the **overall assessment** for this module
2 You are expected to spend about	*10 Hrs (per gp. member)*	Hours to complete this assignment to a satisfactory standard

3 Date assignment set	22nd November 2016	Assignment submission date	9th January 2017
4 Target date for return of marked assignment	30th January 2017		

INTERNAL MODERATION

This assignment has been internally moderated. I confirm: • That the assignment set, meets the requirements of the module and that the brief provides appropriate content for students to successfully complete the assignment. • That the assessment is at an appropriate level and matches QAA level descriptors and is an appropriate form of assessment within the total range of assessments for this module. • That the marking scheme is attached and that students can determine how marks are allocated. • That this assessment can be completed **and** marked within University timeframes, and provides detailed feedback (more than just a grade) that supports learning. .	***Moderator name, signature and date***

Regulations governing assessment offences including Plagiarism and Collusion are available from: http://sitem.herts.ac.uk/secreg/upr/pdf/AS14-Apx3-Assessment%20Offences-v06.0.pdf

University of Hertfordshire UH

Assignment Brief

1 *Using a Slow Bulk Data Storage Device with the CDM8 Microproccessor System*

Your task for this assignment is to interface and utilise a slow bulk Flash RAM device with a Coco de Mere 8 (CDM8) bit simulated microprocessor system. To complete this assignment you will need the following Logisim library files:

1. CdM8-FullCore-CW.circ – this is the CDM8 processor core . In order to use this you will have to load this as a Library circuit into your main Logisim design. To make a working system, you will need to design your own memory and Input/Output circuits.
2. SlowRAM-16x8bit.circ – this is the Cheapo Corp. Ltd. Flash RAM chip which is to be used for the external bulk data storage for your system. This chip should also be loaded as a library part into your main Logisim design and interfaced to suitable Input and Output circuits so that it can utilised as a bulk data storage peripheral for the CDM8 system.

For this assignment, students should work in groups of preferably two persons (but three in a group is acceptable). Typically (at least) one person should concentrate on the circuit designs, and the other(s) on the software development. Direct Enrtry students to the module should work with a fellow group member who has completed the Platforms for Computing Module (4COM1042) last year as they will already be familiar with the basics of CDM8 Assembly language programming.

2 *Assignment Work Tasks*

The work for this assignment is divided into two parts: Basic Tasks and Advanced Tasks, each set of tasks worth up to 10 marks from an overall possible total of 20 marks. To obtain a mark of 10 (50%), the BasicTasks must all be successfully completed. To obtain up to a further 10 marks (> 50%) , then one or more of the Advanced Tasks must be attempted or successfully completed. The work tasks are ordered by degree of increasing difficulty and please note, it is likely that not all groups will successfully complete ALL tasks!

2.1 Basic Work Tasks: 50% (12 marks) maximum.

From a new Logisim file, load the supplied CMM8_core Logisim library and design the circuits to connect the CDM8 core to suitable RAM memory. Use a single 256x8bit memory chip if you plan to use Von Neuman Architecture (i.e. same memory chip for mixed program and data memory). The Basic Work for this assignment consists of the following tasks:

1. Design suitable Input/Output circuits to interface the supplied Flash RAM chip. The Flash RAM chip is supplied as a a Logisim Library file, so you will need to load this into your main design file. The supplied Flash RAM is an asynchronous design, and is an order of magnitude slower to acces than standard static RAM (as available as a standard Logisim library part). Note, the Flash RAM chip has an internal clock circuit, so when testing/using the chip, you will need to “Enable Clock Ticks” from the Logisim Simulation Menu. Write a suitable test program that demonstrates the CDM8 system writing (storing) data to the slow Flash RAM, and also reading (loading) previously stored data back from the Flash RAM into the CDM8 system fast memory. Save your test circuit as CW1.circ and your associated Assembly Code listing as CW1.asm.
2. Add additional IO circuitry using standard Logisim Keyboard and Terminal components and extend your program functionality so that a user can enter the command “s”, then type in a short sentence, which is then saved to the Flash RAM. The display should then be cleared. Your program should then wait until you enter a command “r”, then read back the sentence data from the Flash Memory and display it on the Terminal. Save your revised circuit as CW2.circ and your extended Assembly Code listing of your program as CW2.asm.

University of Hertfordshire UH

2.2 Advanced Work Tasks: 50% (10 marks) maximum.

In order to complete these advanced tasks below, you may want change your system configuration to a Harvard type architecture. This will provide 256 bytes of ROM for your program Code memory only, and another 256 bytes for RAM Data memory. Note the dat/ins' signal from the CDM8 core which is used to select ROM (program instructions) or RAM (data) memory. Also note the additional opcode "ldc" allows programs to indirectly load values from data (RAM) memory.

3. Extend your software functionality from 1. and 2. so that multiple user supplied sentences can be saved to the slow Flash RAM device. You will need to create your own data structures and "filename" (which maybe just numerical index) system for saving each sentence to the Flash RAM in such a way that it can later be located and read back into memory by your CDM8 system. The user should be able to provide the desired "filename" (which may be just a number) for each sentence stored to Flash RAM. The user should be able to re-load and display a previously stored sentence from the Flash RAM back into the CDM8 system main memory, by supplying the "filename" which was used when storing a particular sentence. The "filename" should be used as an index to locate a particular previously stored sentence in the Flash RAM. Save your revised Assembly code listing as CW3.asm
4. Extend your circuit design to include a cache memory between the CDM8 IO circuit and the Flash RAM. The cache memory operation should be totally transparent to the CDM8 and the program that is running. However, when data is read from the Flash memory, the data should be cached in a fast RAM chip, and whenever an address which is cached is detected ("hit"), the CDM8 should be able to read back the fast RAM data cache data full speed. It is up to you to decide the cache block size and number of cached blocks. This will be related to the maximum size of sentence data you wish to store, as well as minimising the time penalty incurred by caching larger block sizes. Save your revised circuit design as CW4.circ

3 Notes and Hints

The CDM8 memory in its standard Von Neuman configuration is limited to a maximum of 256 bytes unless more advanced techniques, such as adopting Harvard Architecture and/or memory Page switching (for which the CDM8 core does provide support!). In any case, you will want to limit the length of allowed sentences to leave enough memory for both your program and data. For displaying messages on the Terminal, rather than filling valuable CDM8 program and/or data memory with long ASCII strings, you may want to use a dedicated ROM (Read Only Memory) preloaded with standard messages, which can be simply selected via an IO port from the CDM8 system, then the characters clocked to the Terminal display by an additional dedicated circuit. When designing a computer system, such trade offs between hardware/software complexity and/or execution speed are common. You will need to think about how you can fit all the required functionality into your resource constrained CDM8 system. Often you can compensate for limited software resources (memory, execution speed etc.), by including additional dedicated digital circuits and vice-versa.

Hints: Save on program memory (i.e. empty wait/delay loops) by using an AND gate to stop ("throttle") the CDM8 system clock in order to pause program execution whenever the slow Flash RAM must be read. Release the clock to run at full speed whenever the cache memory is hit. You will need to decide on the required number and size of cache blocks and also how your system will overwrite ptreviously stored cache blocks with new data when the cache is full.