The British University In Egypt الجامعة البريطانية في مصر Informatics and Computer Science		**Examination Paper Proofing & Printing Confirmation Sheet**	
Module Title	**Computer Architecture**	Module Code **19CSCI10I**	
Module Leader	**Prof. Samir Abou El-Seoud**	Semester **One**	
Proofed by	**Dr. Walid Hussein**		

I hereby confirm that:

- This examination paper has been proof-read (spelling and grammar) ☒
- This examination paper follows the approved template ☒
- All questions (and sub questions) have their marks specified ☒
- Appropriate model answers were provided with breakdown of marks ☒
- This examination paper assesses the ILOs for the module ☒

Signed (Proof Reader): Dr. Walid Hussain

Printing instructions & stationery requirements

Number of copies of examination paper to be printed		
Date of examination		
		Number required per student
Stationery Requirement(s)	8 page answer book	
	12 page answer book	
	Graph paper	
	Other	

Signed (Module Leader) **Prof.** Samir Abou El-Seoud

Informatics and Computer Science	**18CSCI10I** **Final Examination, 2018-2019**
Module Title **Computer Architecture**	
Module Leader **Prof. Samir Abou El-Seoud**	Semester **One**
Equipment allowed **Simple Calculator**	

Instructions to students:

- The exam paper is **3** pages long, and is in **2** sections **A** and **B**.
- You should select only **2** questions of section **A**
- You should answer **all 3 questions** of section **B**.
- The allocation of marks is shown in brackets by the questions.
- The total mark of the exam is 100 marks.

This examination is **Two** hours long.

[Turn Over *(only when instructed to do so)*]

Section A

Answer only 2 questions

Q 1

Consider a machine with three instruction classes and Clock cycle as follows:

Instruction class	Clock cycle of the instruction class
A	3
B	4
C	6

Suppose that we measured the code for a given program in two different compilers and obtained the following data:

Code sequence	Instruction Counts in Millions		
	A	B	C
Compiler 1	10	6	3
Compiler 2	12	5	1

Which code sequence will execute faster according to CPI?

a) What are that system’s cycle per instruction CPI and instruction per cycle IPC values for each code sequence? **[8 marks]**

b) Which code sequence will execute faster according to CPI? **[2 marks]**

c) Generally, compare between values of CPI and IPC for indicating the performance of the systems? **[5 marks]**

[Q1 Total: 15 marks]

Q 2

Suppose in a given architecture that does not have a hardware support for multiplication, so multiplications have to be done through repeated addition. If it takes 100 cycles to perform a multiplication in software and 5 cycles to perform a multiplication in hardware. Suppose a program runs on a machine, with multiply responsible for 40% of this time.

a) What is the overall speedup of the machine in this case? **[5 marks]**

b) If the execution time of a program before improvement is 200 seconds, what is the new execution time for the program? **[4 marks]**

c) How many cycles do we need for multiplication if we want the program to run 2 times faster? **[6 marks]**

[Q2 Total: 15 marks]

Q 3

a) The 8-bit registers R1, R2, R3, and DR initially have the following values:

R1 = 10010011 R2 = 10011111 R3 = 11101001 DR = 11101110

Determine the 8-bit values in each register after the execution of the following sequence of micro-operation:

$DR \leftarrow R1 \oplus R3, R2 \leftarrow R1 \vee R3$ **[6 marks]**

$R3 \leftarrow R1 + \overline{R2} + 1$ **[4 marks]**

$R1 \leftarrow \overline{DR}$ **[1 mark]**

b) Starting from an initial value of R = 11100101, determine the sequence of binary values in R after a circular shift-right, followed by arithmetic shift right, followed by an arithmetic shift left. State whether an overflow occurs or not. Show your work step by step. **[4 marks]**

[Q3 Total: 15 marks]

Section B

Answer all 3 questions

Q 1

Define each of the following items:

a) System Bus **[6 marks]**

b) Reduced Instruction Set Computers (RISC) **[6 marks]**

c) Addressing Mode **[6 marks]**

d) Parallel Processing **[6 marks]**

e) Pipelining **[6 marks]**

[Q1 Total: 30 marks]

Q 2

Answer the following questions:

a) What are the sequences of microoperations within any instruction cycle that are required to fetch an instruction from memory? **[10 marks]**

b) Explain the effect of each microinstruction **[10 marks]**

c) The following transfer statements specify memory operations. Explain the memory operation in each case:

```
R1 ← M[AR]
M[AR] ← R2
R1 ← M[R1]
```

[10 marks]

[Q2 Total: 30 marks]

Q 3

Suppose that we want to perform the arithmetic operation (A_i+B_i) (C_i+D_i) with a stream of numbers. Each suboperation is to be implemented in a segment within a pipeline. Each segment has one or two registers and a combinational circuit (adder and/or multiplier). The concerned registers receive new data with every clock pulse.

- Specify a pipeline configuration to carry out this task and list the contents of all registers in the pipeline for i= 1 through 6.

[Q3 Total: 10 marks]

Model Answer

BUE The British University In Egypt الجامعة البريطانية في مصر Informatics and Computer Science	**18CSCI08** **Resit Examination, 2018-2019**
Module Title **Computer Architecture**	
`	Semester **One**
Equipment allowed **Simple Calculator**	

Instructions to students:

- The exam paper is **3** pages long, and is in **2** sections **A** and **B**.
- You should select only **2** questions of section **A**
- You should answer **all 3 questions** of section **B**.
- The allocation of marks is shown in brackets by the questions.
- The total mark of the exam is 100 marks.

This examination is **<u>Two</u>** hours long.

Question	**ILO assessed**
Section A	
Q1	1,2
Q2	1,2
Q3	1,2
Section B	
Q1	3,4,5
Q2	3,4,6
Q3	4,5,6

Section A

Answer only 2 questions

Q 1

Consider a machine with three instruction classes and Clock cycle as follows:

Instruction class	Clock cycle of the instruction
A	3
B	4
C	6

Suppose that we measured the code for a given program in two different compilers and obtained the following data:

Code sequence	Instruction Counts in Millions		
	A	B	C
Compiler 1	10	6	3
Compiler 2	12	5	1

Which code sequence will execute faster according to CPI?

a) What are that system's cycle per instruction CPI and instruction per cycle IPC values for each code sequence? **[8 marks]**

b) Which code sequence will execute faster according to CPI? **[2 marks]**

c) Generally, compare between values of CPI and IPC for indicating the performance of the systems? **[5 marks]**

[Q1 Total: 15 marks]

Solution:

a) CPI$_1$ = $\frac{(10\times3+6\times4+3\times6)\times10^6}{19\times10^6}$ = 3.78 **[2 marks]**

CPI$_2$ = $\frac{(12\times3+5\times4+1\times6)\times10^6}{18\times10^6}$ = 3.44 **[2 marks]**

IPC$_1$ = $\frac{19\times10^6}{(10\times3+6\times4+3\times6)\times10^6}$ = 0.26 **[2 marks]**

IPC$_2$ = $\frac{18\times10^6}{(12\times3+5\times4+1\times6)\times10^6}$ = 0.29 **[2 marks]**

b) Code Sequence 2 is faster. **[2 marks]**

c) When using IPC and CPI to compare systems, it is important to remember that high IPC values indicate that the reference program took fewer cycles to execute than low CPI values. Thus, a large IPC tends to indicate good performance; while a large CPI indicates poor performance. **[5 marks]**

[Q1 Total: 15 marks]

Q 2

Suppose in a given architecture that does not have a hardware support for multiplication, so multiplications have to be done through repeated addition. If it takes 100 cycles to perform a multiplication in software and 5 cycles to perform a multiplication in hardware. Suppose a program runs on a machine, with multiply responsible for 40% of this time.

a) What is the overall speedup of the machine in this case? **[5 marks]**

b) If the execution time of a program before improvement is 200 seconds, what is the new execution time for the program? **[4 marks]**

c) How many cycles do we need for multiplication if we want the program to run 2 times faster? **[6 marks]**

[Q2 Total: 15 marks]

Solution:

a) Frac$_{used}$ = 0.4, Frac$_{unused}$ =0.6, Seedup$_{used}$ = 100/5 = 20. **[1 mark]**

Plugging these values into Amdahl's Law, we get:

$$Speedup_{overall} = \frac{1}{Frac_{unused} + \dfrac{Frac_{used}}{Speedup_{used}}} = \frac{1}{0.6 + \dfrac{0.4}{20}} = 1.6$$ **[4 marks]**

(2 points for the law and 2 points for the calculations)

b) From part "a" we have

$$Speedup_{overall} = \frac{Exceutiontime_{old}}{Executiontime_{new}} = \frac{1}{Frac_{unused} + \dfrac{Frac_{used}}{Speedup_{used}}} = 1.6$$ **[1 mark]**

$$Executiontime_{new} = \frac{Exceutiontime_{old}}{1.6} = \frac{200}{1.6} = 124 \text{ seconds}$$ **[3 marks]**

(1 point for the law and 2 points for the calculations)

c) Frac$_{used}$ = 0.4, and Frac$_{unused}$ =0.6. **[1 mark]**
Plugging these values into Amdahl's Law, we get:

$$Speedup_{overall} = \frac{1}{Frac_{unused} + \dfrac{Frac_{used}}{Speedup_{used}}}$$

$$2 = \frac{1}{0.6 + \dfrac{0.4}{Speedup_{used}}}$$ therefore,

One enhancement cannot reach a speed up 2 unless
the enhancement is used more than 50% of the program.

(1 point for the law and 5 points for the calculations) **[5 marks]**

[Q2 Total 15 marks]

Q 3

a) The 8-bit registers R1, R2, R3, and DR initially have the following values:

R1 = 10010011 R2 = 10011111 R3 = 11101001 DR = 11101110

Determine the 8-bit values in each register after the execution of the following sequence of micro-operation:

DR ← R1 ⊕ R3, R2 ← R1 V R3 **[6 marks]**

R3 ← R1+ $\overline{\mathrm{R2}}$ +1 **[4 marks]**

R1 ← $\overline{\mathrm{DR}}$ **[1 mark]**

b) Starting from an initial value of R = 11100101, determine the sequence of binary values in R after a circular shift-right, followed by arithmetic shift right, followed by an arithmetic shift left. State whether an overflow occurs or not. Show your work step by step. **[4 marks]**

[Q3 Total: 15 marks]

Solution:

a) DR ← R1 ⊕ R3 ≡ **01111010 (**10010011⊕ 11101001) **[3 marks]**

R2 ← R1 V R3 ≡ **11111011** (10010011 V 11101001) **[3 marks]**

R3 ← R1+ $\overline{\mathrm{R2}}$ +1 ≡ **10011000** (10010011 + 00000100 + 1) **[4 marks]**

R1 ← $\overline{\mathrm{DR}}$ ≡ **10000101** **[1 mark]**

b) R = 11100101

cir = 11110010 **[1 mark]**

asr = 11111001 **[1 mark]**

ashl = 11110010 **[1 mark]**

No overflow occurred **[1 mark]**

[Q3 Total: 15 marks]

Section B

Answer all 3 questions

Q1.

Define each of the following items:

a) System Bus **[6 marks]**

b) Reduced Instruction Set Computers (RISC) **[6 marks]**

c) Addressing Mode **[6 marks]**

d) Parallel Processing **[6 marks]**

e) Pipelining **[6 marks]**

[Total: 30 marks]

Solution

f) **System Bus:** A system bus is a single computer bus that connects the major components of a computer system (processor, memory, I/O). It is a collection of wires through which data is transmitted from one part of a computer to another. It is a communication pathway connecting all the internal computer components to the CPU and main memory. A key characteristic of a system bus is that it is a shared transmission medium. A system bus consists, typically, of from 50 to 100 separate lines. Each line is assigned a particular meaning or function. All system buses consist of three parts: **[2 marks]**

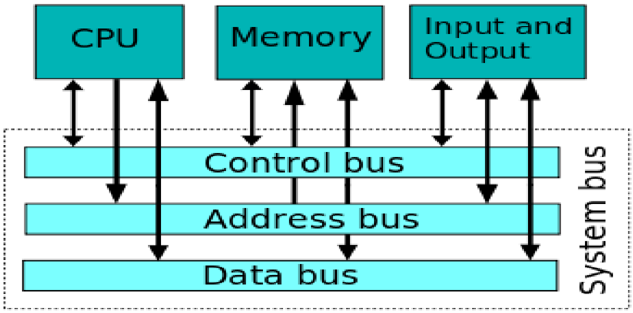
a)

i. a data bus,

ii. an address bus, and

iii. a control bus. **[2 marks]**

The data bus (data lines) transfers actual data whereas the address bus (address lines) transfers information about where the data should go. The control bus (control lines) is used to control the access to and the use of the data and address lines.



[2 marks]

b) **Reduced Instruction Set Computers (RISC)**

RISC processors only use simple instructions that can be executed within one clock cycle. Because RISC architectures are implemented using the **load-store** mode, a processor based upon this concept would use few instructions, which would require fewer transistors, and make them cheaper to manufacture. By reducing the number of transistors and instructions to only those most frequently used, the computer would get more done in a shorter amount of time. **[2 marks]**

Advantages:

- Reduced instructions only
- Produces faster and cheaper processors
- Simpler hardware

- Shorter design cycle **[2 marks]**

Disadvantages:

- Requires more lines of code that becomes increasingly complex
- Low cycle per second
- Despite the speed advantages of the **RISC** processor, it cannot compete with a **CISC** CPU that boasts twice the number of clock cycles **[2 marks]**

c) **Addressing mode**

Addressing modes form part of the instruction set architecture for some particular type of CPU (processors). The term ***addressing modes*** refers to the way in which the operand of an instruction is specified. Information contained in the instruction code is the value of the operand or the address of the result/operand. The job of a microprocessor is to execute a set of instructions stored in memory to perform a specific task. Operations require 1) the operator or opcode which determines what will be done, and 2) the operands which define the data to be used in the operation. Some machine instructions will need to refer to (addresses of) operands in memory.

Briefly, an addressing mode specifies how to calculate the effective memory address of an operand by using information held in registers and/or constants contained within a machine instruction. The addressing mode specifies a rule for interpreting or modifying the address field of the instruction before the operand is actually referenced. When a microprocessor accesses memory, to either read or write data, it must specify the memory address it needs to access. **[3 marks]**

It is difficult to agree on how many addressing modes are available on a particular computer architecture. The most types of common addressing modes are:

- Immediate addressing mode
- Direct addressing mode
- Register addressing mode
- Register indirect addressing mode
- Implicit addressing mode
- Relative mode
- Index mode and base address mode **[3 marks]**

A microprocessor’s instruction set may use some or all of these modes, depending on its design.

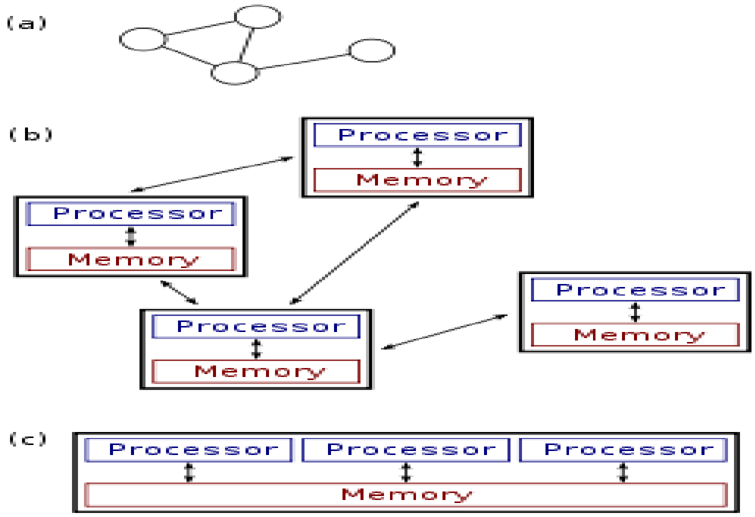
A microprocessor’s instruction set may use some or all of these modes, depending on its design. **[5 marks]**

d) **Parallel Processing**

Parallel processing involves a technique by which complex data sets are broken into individual threads and processed simultaneously across one or more cores. Both AMD and Intel processors have incorporated this technique (known as HTT) to greatly increase the speed at which they operate. Until recently, this did not always provide a significant increase in speed because the technology to properly split up data sets and then bring them back together was in its infancy.

In other words, parallel computing is a form of computation in which many calculations are carried out simultaneously, operating on the principle that large problems can often be divided into smaller ones, which are then solved concurrently ("in parallel"). **[2 marks]**

Parallel computers can be roughly classified according to the level at which the hardware supports parallelism, with multi-core and multi-processor computers having multiple processing elements within a single machine. In parallel computing, all processors may have access to a shared memory to exchange information between processors. The term **Parallelism** refers to the ability to do more than one thing at once. On the contrary, in distributed computing, each processor has its own private memory (distributed memory). Information is exchanged by passing messages between the processors. **[2 marks]**



(a)–(b) A distributed system.

(c) A parallel system. **[2 marks]**

e) **Pipelining:**

Pipelining is a technique for overlapping the execution of several instructions to reduce the overall execution time of a set of instructions.
[1 mark]

Pipelining refers to overlapping operations by moving data or instructions into a conceptual pipe with all stages of the pipe processing simultaneously. **Pipelining** is the process of overlapping parts of a large task to increase **throughput** without decreasing latency. A pipeline does not speed up an individual computation. **[1 marks]**

A pipeline processor executes instructions in an assembly line manner so multiple tasks can be performed simultaneously.

However, at no given time can two of the same tasks be performed, so each task is allotted the same time as the task that takes the longest amount of time. The net effect is that results are output more quickly than in a non-pipelined unit. This increases the **throughput**, the number of results generated per time unit. **[2 marks]**

Throughput is also defined to be the rate at which operations get executed (generally expressed as operations/second or operations/cycle). Think of an automated car wash line, where multiple cars are serviced, but only one vehicle at a time can be shampooed, conditioned or dried. "**Pipelining**", in the context of a processor, means that the CPU scheduler creates a specific list of **linked** instructions (actions) to be fed to the computation units to work. Generally speaking, this list is a series of actions which require the successful completion of the prior one - so, action A completes, then action B takes the output from A and does something, while action C then does something with the output of B, etc. **Pipelining** can bring significant performance benefits, as each successive action finds all its prerequisites already satisfied, so the action is ready to go immediately.

Pipelining does not reduce the time to complete an instruction, but increases the number of instructions that can be processed at once.

[2 marks]

Q2.

Answer the following questions:

a) What are the sequences of microoperations within any instruction cycle that are required to fetch an instruction from memory? **[10 marks]**

b) Explain the effect of each microinstruction **[10 marks]**

c) The following transfer statements specify memory operations. Explain the memory operation in each case:

$R1 \leftarrow M[AR]$

$M[AR] \leftarrow R2$

$R1 \leftarrow M[R1]$ **[10 marks]**

[Q2 Total: 30 marks]

Solution

a) Symbolically, we can write this sequence of events as follows:

t_1: MAR ←(PC) **[2.5 marks]**

t_2: MBR ←Memory **[2.5 marks]**

PC ←(PC)+I **[2.5 marks]**

t_3: IR ←(MBR) **[2.5 marks]**

where I is the instruction length. At the beginning of the fetch cycle, the address of the next instruction to be executed is in the program counter (PC).

b)

- The first step is to move the address in the PC to the memory address register (MAR), because this is the only register connected to the address lines of the system bus. **[3 marks]**

- The second step is to bring in the instruction. The desired address (in the MAR) is placed on the address bus, the control unit CU issues a READ command signal on the control bus, and the result appears on the data bus and is fetched instruction is copied into the memory buffer register (MBR). We also need to increment the PC by 1 to get ready for the next instruction. Because these two actions (read word from memory and add 1 to PC) do not interfere with each other, we can do them simultaneously to save time. **[4 marks]**
- The third step is to move the contents of the MBR to the instruction register (IR). Next step will be decoding the fetched instruction. This will take place by determining the opcode and the operand specifiers. Thereafter the indicated operation will be executed and the result will be stored in the memory. **[3 marks]**

c)

- First microoperation represents the movement of the contents of PC to MAR **[3 marks]**
- Second microoperation represents the movement of memory location specified by MAR to MBR and increment of the PC for the next cycle at the same time **[4 marks]**
- Third microoperation represents the movement of the contents of memory location specified by MBR to IR. **[3 marks]**

[Q3 Total 20 marks]

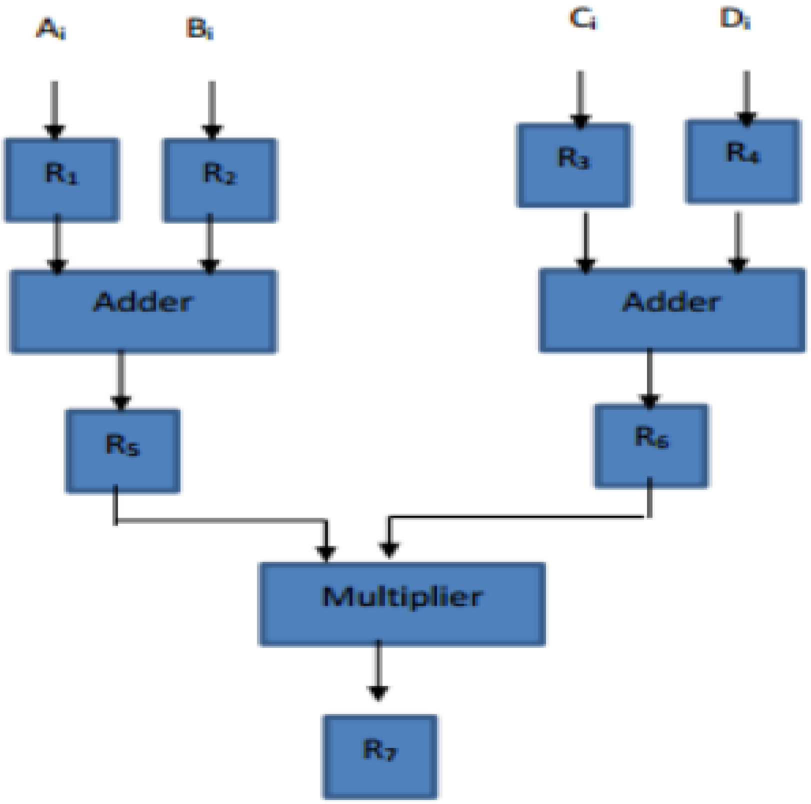
Q3.

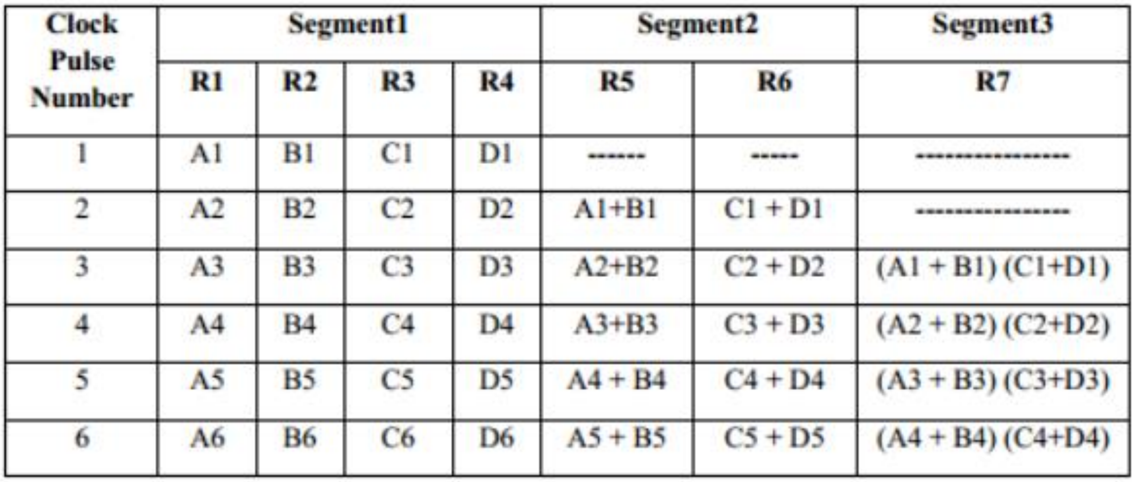
Suppose that we want to perform the arithmetic operation (A_i+B_i) (C_i+D_i) with a stream of numbers. Each suboperation is to be implemented in a segment within a pipeline. Each segment has one or two registers and a combinational circuit (adder and/or multiplier). The concerned registers receive new data with every clock pulse.

- Specify a pipeline configuration to carry out this task and list the contents of all registers in the pipeline for i= 1 through 6.

[Q2 Total: 10 marks]

Solution





Clock Pulse Number	Segment1				Segment2		Segment3
	R1	R2	R3	R4	R5	R6	R7
1	A1	B1	C1	D1	------	-----	----------------
2	A2	B2	C2	D2	A1+B1	C1 + D1	----------------
3	A3	B3	C3	D3	A2+B2	C2 + D2	(A1 + B1) (C1+D1)
4	A4	B4	C4	D4	A3+B3	C3 + D3	(A2 + B2) (C2+D2)
5	A5	B5	C5	D5	A4 + B4	C4 + D4	(A3 + B3) (C3+D3)
6	A6	B6	C6	D6	A5 + B5	C5 + D5	(A4 + B4) (C4+D4)

[10 marks]

Module Specification 2017/18

Module Code: 18CSCI10I	**Title:** Computer Architecture	
Level: I	**Modular weight:** 10	**Faculty/Dept:** ICS/CS,SE,CN,IS
Pre-requisite modules: CSCI08C		
Reassessment: No restriction		
Module Leader: Prof. Samir Abou El-Seoud		
Semester taught: One		
Date of latest revision: April 2017		

Aims

The module aims to impart fundamental knowledge of modern computer architectures in terms of instruction set architecture, organisation and hardware. It develops an understanding of the architectural features and the principles of operation of modern microprocessors and peripheral devices. The module further seeks to provide a sound understanding in the following:

- The main families of microprocessors and their differences;
- How computers execute their programs at machine instruction level; and
- Principles of the practical design of processor architectures and how design features influence machine coding and performance features.

Intended Learning Outcomes

On completion of this module students should be able to:

Knowledge and Understanding

1. To develop and demonstrate knowledge and understanding, abilities and skills in functionality of computer architecture and organization. [A1, A6, A7]

Intellectual Skills

2. Analyse and evaluate the impact of architectural design choices on system performance; the importance of memory organisation and caching on machine performance. [B3, B5]

General and Transferable Skills

3. Experiment the basic principles behind the design of modern computer systems; Differentiate between the features of the main architectural families. [C1]
4. Develop basic skills for computer architecture design; produce code in assembler and a hardware description language. [C2]
5. Develop an analytical approach to evaluating computer systems and competing commercial architectures. [C8]

Transferable skills

6. Gain experience describing technical design using specialist vocabulary. [D4]
7. Plan, develop, evaluate and report on individual pieces of work.[D6]

Employability

This module will provide opportunities for students to:

1. Understand the importance of being self-motivated in order to progress the area of work. [A5]
2. Make decisions by determining the best course of action and evaluating different options based on logic and fact in order to present solutions. [C 2.5]
3. Demonstrate determination to get things done and to constantly looking for better ways of doing things. [C2.6]
4. Carry out a range of complex ICT activities related to their work that involve computer architecture design. [B.3.1]
5. To analyse situations by gathering information systematically to establish facts and principles and to use this to be able to evaluate computer systems. [C.2.3]
6. Gathering information systematically to understand the set of rules and methods that describe the functionality, organization, and implementation of computer systems. [C.2.3]

Indicative Content

Part 1 Introduction

- Introduction to basic CPU architecture. Approaches to CPU design. Amdahl's Law.

Part 2 Different Architectures and Introducing Assembler

- Example architectures (e.g. ARM, iA64). Assembler, calling conventions, register usage, data path and control.
- Pipelining: implementation of a pipeline, hazards, bypasses, exception, assessing the performance.
- Memory interface: alignment, endian, cache organisation, virtual and physical addressing, and coherence.

Part 3 Performance enhancements

- Performance enhancements: Pipelining, cache memory, RISC vs CISC architectures, superscalar architectures, multi-threaded and trace-based architectures.
- Input/output design and implementation

Methods of Learning, Teaching and Assessment

Total student effort for the module: 100 hours on average over one semester.

Type of session	ILOs Covered	Typical Student Effort		
		Typical number in the semester/s	Typical hours per week	Total hours
Lecture	1-6	12	2	24
Tutorial	-	-	-	-
Laboratory	4-7	12	2	24
Private study	1-7	-	-	52

Assessment

Assessment Type	Weight %	ILOs Assessed	Exam Semester	Exam/ Written Coursework Length
Individual Assignment 1: The assignment requires the submission of a fully functioning code of the given problems, and a written solution to a theoretical problem. The type of problems for the first assignment is based on phases 1 and 2 of the indicative content.	20	4-7	1	Assignment
Individual Assignment 2: The assignment requires the submission of a fully functioning code of the given problems, and a written solution to a theoretical problem. The type of problems for the second assignment is based on phases 2 and 3 of the indicative content.	20	4-7	1	Assignment
Final Examination: The final exam will cover all parts of the indicative content to be solved theoretically.	60	1-4	2	120 minutes

Methods of Feedback

In response to assessed work:

- Feedback will be provided for each assessed component through posting model answers with detailed grading schema, and through face to face discussion. If students require additional feedback, they are welcome to speak with the TA, and the module-leader.
- Generic exam feedback will be given on the e-learning system.

Developmental feedback generated through teaching activities:

- Dialogue between students and staff in workshops and Labs.

Indicative Reading List

- Stallings, W., "Computer Organisation and Architecture", 8th Edition, Prentice Hall, ISBN: 0130493074, (2010).
- M. Morris Mano, "Computer System Architecture", 3rd Edition, Prentice Hall International Editions, (1993)