Feedback on Quiz # 1 (Summative)

The **Quiz # 1** was composed of **10** questions, which were randomly selected from a Question Bank of 30 questions. Answers and feedback comments for all of the questions are given below:

Q1.1	Represent the decimal number -19.75 in binary using the **sign-and-magnitude** binary representation (negative = 1).
(a)	110011.110
(b)	111100.100
(c)	111111.001
(d)	101100.010
Feedback: $(19.75)_{10} = (10011.110)_2$ Sign bit (leftmost bit) is negative, so 1 Therefore, $(-19.75)_{10} = (110011.110)_2$	

Q1.2	Represent the decimal number -28.50 in binary using the **sign-and-magnitude** binary representation (negative = 1).
(a)	111100.100
(b)	110011.110
(c)	111111.001
(d)	100011.100
Feedback: $(28.50)_{10} = (11100.100)_2$ Sign bit (leftmost bit) is negative, so 1 Therefore, $(-28.50)_{10} = (111100.100)_2$	

Q1.3	Represent the decimal number -31.125 in binary using the **sign-and-magnitude** binary representation (negative = 1).
(a)	111111.001
(b)	110011.110
(c)	111100.100
(d)	100000.111
Feedback: $(31.125)_{10} = (11111.001)_2$ Sign bit (leftmost bit) is negative, so 1 Therefore, $(-31.125)_{10} = (111111.001)_2$	

Q2.1	Consider the unsigned binary integer $(1100101111)_2$ What would be its equivalent representation in the octal number system?
(a)	$(1457)_8$
(b)	$(0815)_8$
(c)	$(6274)_8$
(d)	$(3260)_8$

Feedback:

To obtain the octal equivalent, we take numbers in groups of 3, from right to left as (remember to add 0s to the left most position):

001 100 101 111

1 4 5 7 = $(1457)_8$

Q2.2	Consider the unsigned binary integer $(1011001010011)_2$ What would be its equivalent representation in the hexadecimal number system?
(a)	$(1653)_{16}$
(b)	$(5715)_{16}$
(c)	$(B298)_{16}$
(d)	$(45720)_{16}$

Feedback:

To obtain the hexadecimal equivalent, we take numbers in groups of 4, from right to left as (remember to add 0s to the left most position):

0001 0110 0101 0011

1 6 5 3 = $(1653)_{16}$

Q2.3	Consider the unsigned binary integer $(1000110010110)_2$ What would be its equivalent representation in the hexadecimal number system?
(a)	$(1196)_{16}$
(b)	$(4502)_{16}$
(c)	$(8CB0)_{16}$
(d)	$(36016)_{16}$

Feedback:

To obtain the hexadecimal equivalent, we take numbers in groups of 4, from right to left as (remember to add 0s to the left most position):

0001 0001 1001 0110

1 1 9 6 = $(1196)_{16}$

Q3.1	Which of these statements are true?
(a)	Registers can be accessed more quickly than main memory.
(b)	Accessing data from cache is faster than accessing main memory.
(c)	Registers can be addressed with fewer address bits than a main memory address.
(d)	Every instruction has an equal likelihood of being executed.
(e)	Using cache changes the result that a program will produce.

Feedback:

Registers are faster to access than cache which is faster to access than main memory.

There will be a small number of registers and so a register can be addressed using fewer bits. Main memory is larger and so we will need more bits to specify an address (e.g. 5 bits to specify 32 registers against, perhaps, 32 or 64 bits to specify a main memory address).

Empirically, some instructions will be executed more often than others e.g. instructions in the body of loops. Cache only speeds up the access to memory. It does not change the result of a program.

Q3.2	Which of these statements are false?
(a)	Main memory is larger than cache memory.
(b)	Accessing data from main memory is faster than accessing instructions.
(c)	Programs will run faster if the instructions are smaller.
(d)	Caching is effective because the likelihood of an address being accessed is greater if it has been accessed recently.
(e)	In a von Neumann machine there are separate memory pathways to data and instructions.

Feedback:

Main memory is much larger (and slower to access) than cache memory.

The data paths are the same for accessing data and instructions, so essentially we get the same access time for data and instructions.

If an instruction is smaller then less data needs to be fetched which is often the bottleneck. Therefore, the program will run faster (e.g. 32 bit instructions require 4 bytes to be fetched compared to 8 bytes for 64-bit instructions.

Caching works because, empirically, programs will have both locality of instructions and data: i.e. loops execute the same instructions many times and these will access the same/nearby data.

In a von Neumann architecture, there is a single unified memory which holds both program and data.

Q3.3	Which of these statements are true?
(a)	We need 32 bits to address either a register or main memory location.
(b)	The purpose of a cache is to reduce the number of bits needed to address main memory.
(c)	The goal of using a cache is to reduce the time taken to access data and instructions.
(d)	Main memory can be accessed faster than registers
(e)	An advantage of the Harvard architecture is that instruction and data memories can be accessed in parallel.

Feedback:
There will be a small number of register locations and so these can be addressed using fewer bits (e.g. 5) whereas main memory is much larger and so a much larger address is needed to specify the memory locations.
Cache is transparent and does not affect the way that memory is addressed therefore the number of bits required is unaffected.
The point of cache is that it can be accessed quickly - it is faster to access than main memory and can be used for both data and instructions.
Registers are few but very fast to access - much quicker than main memory.
The Harvard architecture has separate memory pathways for data and instructions. This means that they can be accessed in parallel.

Q4.1	On a hypothetical computer, real numbers are stored in a two's complement fixed-point binary format: ● The first five bits represent the integer part of the number. ● The last three bits represent the fractional part of the number. Compute the binary representation of the answer to the arithmetic equation (01010.101) - (11100.100)
(a)	01110001
(b)	00111001
(c)	10111001
(d)	01111101

Feedback:

$(01010.101)_2$ - $(11100.100)_2$

= $(8+2+0.5+0.125)_{10}$ - $(-16+8+4+0.5)_{10}$

= $(10.625)_{10}$ - $(-3.5)_{10}$

= $(14.125)_{10}$

= $(01110.001)_2$

Alternatively, complement the second argument and add:

```
 01010.101
-11100.100
=
 01010.101
+00011.100  (flipped and +1 in the rightmost position)
—----------
 01110.001
```

Q4.2	On a hypothetical computer, real numbers are stored in a two's complement fixed-point binary format: ● The first five bits represent the integer part of the number. ● The last three bits represent the fractional part of the number. Compute the binary representation of the answer to the arithmetic equation (00010.111) - (10011.010)
(a)	01111101
(b)	10110001
(c)	00110001
(d)	01111011

Feedback:

$(00010.111)_2$ - $(10011.010)_2$

= $(2+0.5+0.25+0.125)_{10}$ - $(-16+2+1+0.25)_{10}$

= $(2.875)_{10}$ - $(-12.75)_{10}$

= $(15.625)_{10}$

= $(01111.101)_2$

Alternatively, complement the second argument and add:

```
 00010.111
-10011.010
=
 00010.111
+01100.110  (flipped and +1 in the rightmost position)
—----------
 01111.101
```

Q4.3	On a hypothetical computer, real numbers are stored in a two's complement fixed-point binary format: ● The first five bits represent the integer part of the number. ● The last three bits represent the fractional part of the number. Compute the binary representation of the answer to the arithmetic equation (01110.100) - (11111.001)
(a)	01111011
(b)	01101101
(c)	11101101
(d)	01110001

Feedback:

$(01110.100)_2$ - $(11111.001)_2$

= $(8+4+2+0.5)_{10}$ - $(-16+8+4+2+1+0.125)_{10}$

= $(14.5)_{10}$ - $(-0.875)_{10}$

= $(15.375)_{10}$

= $(01111.011)_2$

Alternatively, complement the second argument and add:

```
 01110.100
-11111.001
=
 01110.100
+00000.111  (flipped and +1 in the rightmost position)
—----------
 01111.011
```

Q5.1	Consider the binary integer $(10011010)_2$ If it is an unsigned integer, the decimal equivalent is _____. If it is a 2's complement integer, the decimal equivalent is _____.
(a)	154
(b)	-102
(c)	-154
(d)	-100
(e)	100
(f)	102

Feedback:

Unsigned:

$2^7 + 2^4 + 2^3 + 2^1$

= 128 + 16 + 8 + 2

= 154

2's complement:

As this binary number starts with "1", it is a negative number.

Reversing 2's complement:

10011010 - 1 = 10011001

Flip all bits: 10011001 -> 01100110

Decimal representation of 01100110 is

$2^6 + 2^5 + 2^2 + 2^1$

= 64 + 32 + 4 + 2

= 102

Therefore, the answer for 2's complement is -102.

Q5.2	Consider the binary integer $(10101101)_2$ If it is an unsigned integer, the decimal equivalent is _____. If it is a 2's complement integer, the decimal equivalent is _____.
(a)	173
(b)	-83
(c)	-173
(d)	-81
(e)	81
(f)	83

Feedback:

Unsigned:

$2^7 + 2^5 + 2^3 + 2^2 + 2^0$

= 128 + 32 + 8 + 4 + 1

= 173

2's complement:
As this binary number starts with "1", it is a negative number.
Reversing 2's complement:
10101101 - 1 = 10101100
Flip all bits: 10101100 -> 01010011

Decimal representation of 01010011 is
$2^6 + 2^4 + 2^1 + 2^0$
= 64 + 16 + 2 + 1
= 83

Therefore, the answer for 2's complement is -83.

Q5.3	Consider the binary integer $(11010010)_2$ If it is an unsigned integer, the decimal equivalent is _____. If it is a 2's complement integer, the decimal equivalent is _____.
(a)	210
(b)	-46
(c)	-210
(d)	-44
(e)	44
(f)	46

Feedback:
Unsigned:
$2^7 + 2^6 + 2^4 + 2^1$
= 128 + 64 + 16 + 2
= 210

2's complement:
As this binary number starts with "1", it is a negative number.
Reversing 2's complement:
11010010 - 1 = 11010001
Flip all bits: 11010001 -> 00101110

Decimal representation of 00101110 is
$2^5 + 2^3 + 2^2 + 2^1$
= 32 + 8 + 4 + 2
= 46
Therefore, the answer for 2's complement is -46.

Q6.1	Which of these statements are true?
(a)	A pipeline allows successive instructions to be at different stages of the execution cycle.
(b)	Instructions which transfer control (e.g. jump instructions) will reduce the effectiveness of the pipeline.
(c)	A 5 stage pipeline will mean that one instruction is completed on every clock tick.
(d)	If the processor clock speed is doubled then programs will run at twice the speed.
(e)	Using a pipeline will increase the throughput of the processor.

Feedback:

The idea of a pipeline is that we can have multiple instructions at different stages of instruction execution (once an instruction has completed the instruction fetch phase and moved to the instruction decode phase then the next instruction can enter the instruction fetch phase).
Ideally, this will mean that we can have an instruction complete execution on every clock tick.
However, there are many reasons why this will not happen: we have to fill the pipeline, control transfer may mean that the instructions in the pipeline need to be discarded, memory waits and so on. It will still improve throughput, though.
If we increase clock speed we would expect to see improved throughput - but not in proportion because of factors like memory bandwidth.

Q6.2	Which of these statements are true?
(a)	Data is transferred between processor and memory using the control bus.
(b)	Doubling the length of a pipeline will double the throughput of the processor.
(c)	The Data Fetch stage of instruction execution loads the next instruction into the instruction register.
(d)	Conditional jump instructions will affect the benefit gained from pipelining.
(e)	The use of cache is transparent to the programmer

Feedback:

Data is transferred between processor and memory using the data bus.
Breaking instruction execution into smaller steps allows a longer pipeline which will improve throughput - but not in proportion to the length of the pipeline.
The Instruction Fetch stage loads the next instruction into the Instruction Register.
Pipelining relies on ‘knowing’ the order of instruction execution. When there are conditional jumps then the next instruction is unpredictable and so throughput will be affected.
Cache will speed up the program’s execution, because it will speed memory access, but it will not change the the way the program behaves (except, perhaps, when there are real-time issues)

Q6.3	Which of the following are true?
(a)	An instruction normally specifies each of a) the operation to be performed b) the two operands c) the location to store the result d) the address of the next instruction.
(b)	The value of an ‘immediate’ operand is stored in a register.
(c)	The number of instructions executed per second is equal to the clock speed.
(d)	An instruction set architecture specifies the instruction set precisely.
(e)	The throughput of a pipeline is affected by the number of ‘jump’ instructions.

Feedback:

Instructions typically specify two operands and what to do with the result. They might specify just 1 or 2 of those. They do not specify operands, result and address of the next location.

An ‘immediate’ operand is usually specified as part of the instruction.

The number of instructions executed per second is usually lower than the clock speed.

The ISA is a specification of how the architecture works. It is precise and unambiguous. The implementation of the architecture can vary but its semantics should be constant.

A pipeline relies on having successive instructions in the pipeline at different stages of execution. Instructions that change the flow of control can mean that the instructions in the pipeline need to be abandoned. Therefore the throughput is affected.

Q7.1	A hypothetical computer stores real numbers in floating point format in 7-bit words: ● The first bit is used for the sign of the number (1 is negative). ● The second bit used for the sign of the exponent (1 is negative). ● The next two bits are used for the magnitude of the exponent. (We do not add an offset to the exponent). ● The final three bits are used for the magnitude of the mantissa. Convert the value $(1011101)_2$ in this representation into its decimal equivalent.
(a)	-13
(b)	-0.203125
(c)	-5
(d)	-0.078125

Feedback:

Given value: 1011101

Sign number bit = 1, so number is negative

Sign exponent bit = 0, so exponent is positive

Exponent = $(11)_2$ = 3

Mantissa = 101

So it can be written in standard form as = 1.101 * 2^(3)

Which is equivalent to = 1101.0

Number = - $(1101.0)_2$

= - $(2^3+2^2+2^0)$

= - (8+4+1)

= - 13

Q7.2	A hypothetical computer stores real numbers in floating point format in 7-bit words: ● The first bit is used for the sign of the number (1 is negative). ● The second bit used for the sign of the exponent (1 is negative). ● The next two bits are used for the magnitude of the exponent. (We do not add an offset to the exponent). ● The final three bits are used for the magnitude of the mantissa. Convert the value $(1010110)_2$ in this representation into its decimal equivalent.
(a)	-7
(b)	-0.4375
(c)	-3
(d)	-0.1875

Feedback:

Given value: 1010110

Sign number bit = 1, so number is negative

Sign exponent bit = 0, so exponent is positive

Exponent = $(10)_2$ = 2

Mantissa = 110
So it can be written in standard form as = 1.110 * 2^(2)
Which is equivalent to = 111.0
Number = - $(111.0)_2$
= - $(2^2+2^1+2^0)$
= - (4+2+1)
= - 7

Q7.3	A hypothetical computer stores real numbers in floating point format in 7-bit words: ● The first bit is used for the sign of the number (1 is negative). ● The second bit used for the sign of the exponent (1 is negative). ● The next two bits are used for the magnitude of the exponent. (We do not add an offset to the exponent). ● The final three bits are used for the magnitude of the mantissa. Convert the value $(1001100)_2$ in this representation into its decimal equivalent.
(a)	-3
(b)	-0.75
(c)	-1
(d)	-0.25

Feedback:
Given value: 1001100
Sign number bit = 1, so number is negative
Sign exponent bit = 0, so exponent is positive
Exponent = $(01)_2$ = 1
Mantissa = 100
So it can be written in standard form as = 1.100 * 2^(1)
Which is equivalent to = 11.0
Number = - $(11.0)_2$
= - (2^1+2^0)
= - (2+1)
= - 3

Q8.1	Which of the following decimal integers can be stored in a 7-bit register?
(a)	-64
(b)	64
(c)	63
(d)	128
(e)	-128
(f)	255

Feedback:

<u>If the register uses 2's complement:</u>

The most negative value that we can represent in 7-bit is 1000000, where the most significant bit indicates the -ve sign. To find out its equivalent value in decimal, we can take 2's complement.

The inversion of 1000000 is 0111111.
Add 1 to 0111111 = 0111111+1 = $(1000000)_2 = (64)_{10}$
So we can represent $(-64)_{10}$.

The largest positive value that we can represent in 7-bits = 0111111 = $(63)_{10}$

In general, for n bits, the range is: -2^{n-1} --> $2^{n-1} - 1$

Therefore, the range is -64 to 63.

i.e., In this case, the register can store -64 and 63.

<u>If the register uses unsigned:</u>

Total possible number = $2^7 = 128$. Therefore, the range is 0 to 127.
In general, for n bits, the range is: 0 to $(2^n - 1)$.

i.e., In this case, the register can store 63 and 64.

Q8.2	Which of the following decimal integers can be stored in a 9-bit register?
(a)	-256
(b)	256
(c)	255
(d)	512
(e)	-512
(f)	*1023*

Feedback:

<u>If the register uses 2's complement:</u>

The most negative value that we can represent in 9-bit is 100000000, where the most significant bit indicates the -ve sign. To find out its equivalent value in decimal, we can take 2's complement.

The inversion of 100000000 is 011111111.
Add 1 to 011111111 = 011111111+1 = $(100000000)_2$ = $(256)_{10}$
So we can represent $(-256)_{10}$.

The largest positive value that we can represent in 9-bits = 011111111 = $(255)_{10}$

In general, for n bits, the range is: -2^{n-1} --> $2^{n-1} - 1$

Therefore, the range is -256 to 255.

i.e., In this case, the register can store -256 and 255.

<u>If the register uses unsigned:</u>
Total possible number = 2^9 = 512. Therefore, the range is 0 to 511.
In general, for n bits, the range is: 0 to (2^n - 1).

i.e., In this case, the register can store 255 and 256.

Q8.3	Which of the following decimal integers can be stored in a 11-bit register?
(a)	-1024
(b)	1024
(c)	1023
(d)	2048
(e)	-2048
(f)	4095

Feedback:
<u>If the register uses 2's complement:</u>
The most negative value that we can represent in 11-bit is 10000000000, where the most significant bit indicates the -ve sign. To find out its equivalent value in decimal, we can take 2's complement.

The inversion of 10000000000 is 01111111111.
Add 1 to 01111111111 = 01111111111+1 = $(10000000000)_2$ = $(1024)_{10}$
So we can represent $(-1024)_{10}$.

The largest positive value that we can represent in 11-bits = 01111111111 = $(1023)_{10}$

In general, for n bits, the range is: -2^{n-1} --> $2^{n-1} - 1$

Therefore, the range is -1024 to 1023.

i.e., In this case, the register can store -1024 and 1023.

<u>If the register uses unsigned:</u>
Total possible number = 2^{11} = 2048. Therefore, the range is 0 to 2047.
In general, for n bits, the range is: 0 to (2^n - 1).

i.e., In this case, the register can store 1023 and 1024.

Q9.1	Which of these statements are true?
(a)	A compiler must generate binary machine code that can be loaded and run directly.
(b)	An interpreter is a software implementation of an existing Instruction Set Architecture.
(c)	A just in time compiler generates executable code at runtime.
(d)	The JVM is only capable of executing programs written in Java.
(e)	A program compiled using a compiler will generate the same result as one executed through an interpreter.

Feedback:

A compiler can generate code in any language. It might be another high level language, assembly language or raw binary. It is unusual that even binary code be directly executed because we usually need to 'relocate' the code because of multiple modules and libraries.

An interpreter is usually an implementation of a virtual machine. It could be an existing concrete machine (this is usually referred to as an emulator) but not usually.

Just in time compilers generate code at runtime.

JVM has been used as the target of compilers for many languages, not just Java. Moreover, JVM executes bytecode, which is an intermediate representation and is different from Java.

The source program is a specification of its behaviour. How it is implemented (on what machine or using what techniques) should not change its behaviour - this is not strictly true of real time behaviour but that is dependent on many things.

Q9.2	Which of these statements are true?
(a)	An assembler just maps instruction mnemonics written in assembler code into the corresponding bit pattern.
(b)	A compiler must generate binary code for the machine on which it is running.
(c)	Interpreters are well suited to a development environment whilst compilers are more suited to production environments.
(d)	Code optimisation typically transforms the code into an equivalent program that will run faster.
(e)	A compiler must check the syntactic correctness of a program.

Feedback:

A compiler is a program which will analyse the source code (the program being compiled) and generate an equivalent program in the object code (the equivalent program in the language being generated). This language is often assembler which is then further processed or can be an intermediate language or other languages. Generating efficient code is an expensive process and so compiling to executable code is usually much slower than starting execution in an interpreter. So, an interpreter may be preferred for development - whilst compiled code will be preferred in a production environment.

The optimisation phase can be very complex. The aim is usually to produce an equivalent program that runs as quickly as possible - for instance by changing instruction or re-ordering instructions.

Whilst there is a much closer relationship between assembler and the binary machine code, the process of assembly involves more than just mapping 1:1. For instance, symbolic addresses (e.g. labels) need to be resolved. Some instructions in assembler are ‘pseudo-instructions’ which need to be mapped to 1 or more actual instructions.

Q9.3	Which of these statements are true?
(a)	Interpreters have to generate machine code at execution time.
(b)	Pure interpreters provide a software implementation of a machine.
(c)	Cross compilation means that we generate code for a different machine to that on which we compile it.
(d)	In general, an interpreted program will execute faster than a compiled one.
(e)	An interpreter written in a high level language will be portable across different computer architectures.

Feedback:

At its purest, an interpreter is a software implementation of a machine (often a ‘virtual’ machine but possibly a machine where there is a hardware implementation). They may generate executable code ‘on the fly’ but they may not. This means that if the implementation is written in a high level language (e.g. Java) then it can be executed on any machine that supports that language.

An interpreter is usually slower because the software execution will take longer than if the operation is performed by hardware.

Cross compilation is the process where a compiler generates code for a different machine. You might generate code on a PC or Mac that will be downloaded and executed on an embedded device (e.g. a washing machine), for instance.

Q10.1	Consider the following RPN expression: 7 3 5 4 * 9 - * + Evaluate this expression using a stack and select the correct answer from the given choices.
(a)	**40**
(b)	-44
(c)	-21
(d)	7
(e)	44
(f)	35

Feedback:

Let's evaluate the following expression with the help of a stack.

7 3 5 4 * 9 - * +

We can evaluate the RPN using the following steps:

Step #	Input	Stack
1	7	7
2	3	7, 3
3	5	7, 3, 5
4	4	7, 3, 5, 4
5	*	7, 3, 20
6	9	7, 3, 20, 9
7	-	7, 3, 11
8	*	7, 33
9	+	40

So, the correct answer is **40**, as shown in step#9 above.

Q10.2	Consider the following RPN expression: 7 3 * 5 4 9 + * - Evaluate this expression using a stack and select the correct answer from the given choices.
(a)	40
(b)	**-44**
(c)	-21
(d)	7
(e)	44
(f)	35

Feedback:

Let's evaluate the following expression with the help of a stack.

7 3 * 5 4 9 + * -

We can evaluate the RPN using the following steps:

Step #	Input	Stack
1	7	7
2	3	7, 3
3	*	21
4	5	21, 5
5	4	21, 5, 4
6	9	21, 5, 4, 9
7	+	21, 5, 13
8	*	21, 65
9	-	-44

So, the correct answer is **-44**, as shown in step#9 above.

Q10.3	Consider the following RPN expression: 7 3 5 + 4 9 * - + Evaluate this expression using a stack and select the correct answer from the given choices.
(a)	40
(b)	-44
(c)	**-21**
(d)	7
(e)	44
(f)	35

Feedback:

Let's evaluate the following expression with the help of a stack.

7 3 5 + 4 9 * - +

We can evaluate the RPN using the following steps:

Step #	Input	Stack
1	7	7
2	3	7, 3
3	5	7, 3, 5
4	+	7, 8
5	4	7, 8, 4
6	9	7, 8, 4, 9
7	*	7, 8, 36
8	-	7, -28
9	+	-21

So, the correct answer is **-21**, as shown in step#9 above.