Coursework commentary 2018–2019

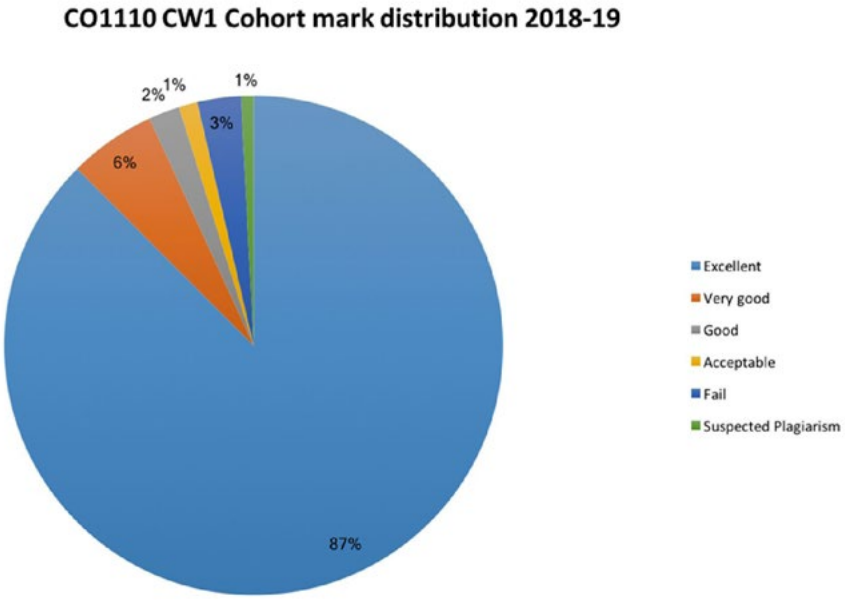
CO1110 Introduction to computing and the internet

Coursework assignment 1

General remarks

While most students did well with the first Coursework assignment, often students lost some credit, not because their answers were wrong, but because their answers were unclear, or because students did not show their calculations to demonstrate how they had arrived at a result.

See 2018-2019 cohort mark distribution below:



Comments on specific questions

Question 1

a. Students were asked to demonstrate how to transform the decimal number 1025.25 to IEEE 754 single precision representation of:

0 1000 1001 0000 0000 0101 0000 0000 000

In their answer students were asked to clearly identify the mantissa and the normalised mantissa, the exponent, the biased exponent, and the sign bit.

Solution

Converting 1025.25 (decimal) to unsigned binary gives:

$$1000\ 0000\ 001.01 \times 2^{0}$$

Shifting the binary point and incrementing the exponent gives:

$$1.000000000101 \times 2^{10}$$

Hence the **mantissa** is 1.000000000101

And the **normalised mantissa** is:

0000 0000 0101 0000 0000 000

The **exponent** is 10_{10}, so the **biased exponent** is 137_{10}. As a binary number 137 is 1000 1001.

Since this is a positive number the **sign bit** is 0, giving the answer:

0 1000 1001 0000 0000 0101 0000 0000 000

Comments

All students received some marks for this question. Most students who lost marks had correctly given the sign bit, mantissa and exponent values, but lost marks by not clearly identifying the mantissa and the normalised mantissa. To a lesser extent students also lost marks by not clearly distinguishing between the exponent and the biased exponent.

b. Students were asked to give the number 1.10111×2^{-130} in IEEE 754 32-bit denormalised form.

Solution

All that was necessary was to add one to the power of the exponent while at the same time moving the radix of the binary number one place to the left. This should be repeated until the exponent had the smallest possible value of 2^{-126}.

1.10111×2^{-130}	Starting value
0.110111×2^{-129}	First shift
$0.0110111 \times 2^{-128}$	Second shift
$0.00110111 \times 2^{-127}$	Third shift
$0.00010111 \times 2^{-126}$	Final shift

Answer: $0.00010111 \times 2^{-126}$

Comments

Some students gave their answer in complete IEEE denormalised form, which was good, but the above answer was accepted for full credit.

It was surprising that a common error was to give the number with an incorrect exponent, both 2^{-125} and 2^{-127} were seen. A more serious error was to transform the number given, 1.10111×2^{-130}, directly to IEEE 754 32-bit form, i.e. taking the mantissa as 1.10111, the normalised mantissa as 1011 1000 0000 0000 0000 000, and converting the exponent, 2^{-130}, to binary. Another common error was in rounding the mantissa, and losing some precision. It may be necessary, when giving a number in denormalised form, to lose some precision as perhaps, after shifting and adding some leading zeros, not all the digits of the mantissa will fit into the 23 places allowed. However, in this case it was not necessary to lose any precision, as there were less than 23 digits in the final answer.

Students can find a discussion, with an example, of a denormalised number in the subject guide in section **6.4.8 Range of IEEE 754 single precision**.

Question 2

a. Students were asked to comment on whether or not the result of the following two's complement addition demonstrated an overflow:

```
 1001
 1010
 ----
10011
 ====
```

Solution

Answer: The result of adding two negative numbers is a positive number. This change of sign indicates an overflow (all marks).

Alternative answer: this is the addition of –7 and –6, giving –13, which is outside of the range of 4-bit two's complement (–8 to +7) (half marks).

Comments

One common mistake in answer to part (a) was to explain that there was an overflow because the result of addition with 4 bits was a number with 5 bits. This is completely wrong, as the subject guide explains in section 6.3.8, the extra bit is discarded and has no bearing on whether or not there is an overflow. Hence the result of the addition is 0011, a positive number since the sign bit is 0.

In answering this question there was no need to convert the two's complement numbers to their decimal values. Students were expected to understand from the subject guide (section **6.3.9 Integer overflow**) that 'we can tell that an overflow has occurred in two's complement notation, simply by noting that the sign bit of the result is different from the sign bits of the operands'. In fact, as the introduction to two's complement numbers (**6.3.8 Two's complement notation**) notes, this simple way to detect overflow is one of the advantages of two's complement arithmetic. For this reason, the alternative answer above, while correct, did not receive full credit.

b. Students were asked to sign extend the two's complement numbers given in part (a), and add them again, once again explaining whether the result of the addition did or did not demonstrate an overflow.

Solution

Sign extending (3 marks):

$$1001 \rightarrow 1\_\_001 \rightarrow 1\mathbf{11}001$$

$$1010 \rightarrow 1\_\_010 \rightarrow 1\mathbf{11}010$$

Addition (3 marks for showing working, 3 for correct result):

```
 111001
 111010
1110011
```

Note the extra carry bit is discarded leaving 110011 as the answer.

Comment on overflow (6 marks):

- The result has no overflow as the addition of 2 negative numbers is a negative number, as expected (all marks).
- The result has no overflow as –7 + –6 is –13 which is in the range of 6-bit two's complement (-32 to 31) (half marks).

Comments

By far the most common error in this entire assignment was with the sign extending.

To sign extend means to change an x-bit two's complement number to an (x+y)-bit two's complement number where *x* and *y* are positive integers, using a simple method as follows: Firstly, move the sign bit to the left, separated from the rest of the number by *y* spaces. Secondly, fill in those spaces with copies of the sign bit. This procedure is explained in the subject guide in section **6.3.13 Sign extension**, and will give the correct two's complement representation for the new number of bits.

Many students worked out that the original numbers were -6 and -7 in decimal, and directly converted those numbers to 6-bit two's complement numbers. Other students showed no working out for the sign extension; both of these groups received no marks for sign extension. Some students gave the wrong number, often by first correctly moving the sign bit to the left but then filling in the gaps with zeros, rather than with copies of the sign. These students received partial credit.

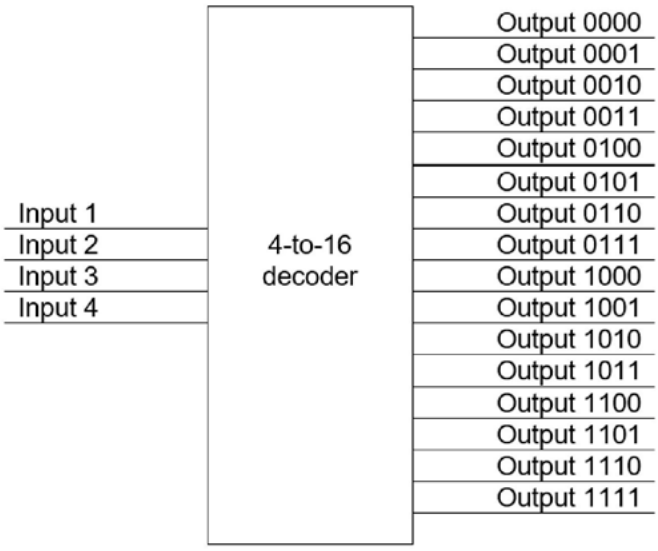
In answering this question some students made the same mistake they had made in part (a), claiming that the extra carried bit meant an overflow had happened, writing such statements as: *6-bit addition resulting in 7 bits indicates an overflow*. Again, in two's complement addition an extra carried bit is discarded, hence the answer, 110011, shows no change of sign from the operands and it is safe to conclude there is no overflow.

Note that students received 3 marks for showing the working of their 6-bit addition, even when the numbers used were incorrect. This means that those students who showed no working lost 3 marks, even when their 6-bit numbers were correct.

Question 3

a. Students were asked to draw a simplified diagram of a decoder with 4 inputs.

Solution



Comments

A few students submitted much more complex diagrams, which were very good and appreciated by the examiners. While these students received full credit, so did those submitting the simplified diagram above.

Most students received full credit for their answers to this question, with only a few mistakes seen with the output: (1) giving the wrong number of outputs (4 or 8 rather than 16); (2) not labelling each output, for example by giving the first 2 or 3 only and using dots to imply the rest; and (3) giving the output as decimal numbers.

b. Students were asked some questions about a byte addressable memory comprised of 128 chips. The chips were 256 x 8-bit RAM chips.

Solution

i. Each chip has 2^8 bytes on it. There are 128, or 2^7 chips. Since memory is byte addressable, the number of addressable memory locations is the number of bytes. There are $2^8 \times 2^7 = 2^{15}$ bytes, so the number of addressable memory locations is 2^{15}.

ii. The answer to part (b)(i) means that 15 bits are needed for memory addresses.

iii. There are 2^7 chips, so 7 address lines are needed for chip select.

iv. For byte selection, 15-7 address lines are needed, or 8.
Or, since there are 2^8 bytes on a chip, 8 bits are needed for byte addressing.

Comments

Most students gave correct answers to question (b). Incorrect answers to this question were divided between students who made a mistake with part (i), sometimes leading to further mistakes, and those who correctly answered part (i), and sometimes (ii) too, but could not give correct answers to (iii) and (iv). Incorrect answers varied and it was hard to determine common mistakes.

c. Students were asked some questions about a memory of the same composition as that given in part (b). However, the memory was now word addressable, with a word size of 64 bits.

Solution

i. Again, memory has 2^{15} bytes. A word is composed of 64 bits, or 8 bytes, which is 2^3 bytes. $2^{15}/2^3 = 2^{12}$. So there are now 2^{12} addressable memory locations.

ii. The 128 chips are grouped into sets of 8, which is 16 groups, or 2^4. Hence 4 bits are needed for chip select. Since there are 12 bits in addresses, this leaves 8 bits for word selection. Alternatively, 12 bits are needed for addresses. Since each chip has 256 bytes, or 2^8 bytes, and since chips are grouped with one byte of a word on each chip, 8 bits only are needed for word selection. This leaves 4 bits for chip selection. The most significant bits of the address are used for chip selection, hence **1111** is chip selection and **1010 1001** word selection.

Comments

Very few mistakes were seen with part (i), with nearly all students answering correctly. This was expected, since the 12-bit example address given in part (i) was a strong hint that the memory size was 2^{12}. With part (ii) many students lost credit. Sometimes it was because their answers were incorrect, but usually it was because students gave answers that were correct but not specific enough. Students were expected to state that four bits were used for chip selection, and to identify those bits as either 1111 in the address given, or as the four most significant bits of the address, or both. For example, the following answers would gain partial credit only:

- The chip select is 4 bits. The remaining 8 bits is for memory address.
- In this memory there are 4 bits for chip select.
- The first 4 bits are used for chip select, and the remaining 8 bits are used to identify the word.

Question 4

Students were asked to give an explanation of how the cache uses spatial locality to increase the cache hit rate in just one sentence. Students were asked to note that an answer to the question of:

> Cache memory uses spatial locality to increase the cache hit rate by storing frequently used addresses.

would gain no marks.

Solution

To exploit the principle of spatial locality, which states that memory locations accessed in any time period are likely to be clustered in one place, when the cache admits a new memory unit it also admits the surrounding memory units.

Comments

Temporal locality and spatial locality are jointly known as locality of reference. Temporal locality notes that a processor is likely, in a short time period, to access again memory locations that have been used recently because much program execution is spent in loops. Spatial locality notes that the processor often accesses locations that are clustered in one place, because instructions are normally accessed sequentially.

The example answer given implied that the cache monitors how often certain addresses are used, and would keep those addresses used repeatedly in the cache. This was incorrect for two reasons: first, the example answer given discussed **frequently used** addresses, which references the temporal locality principle; and second, both locality of reference principles are used to predict what memory locations are likely to be accessed next.

While some correct answers were seen, many answers were poor and received partial or no credit. The major mistake was that many students failed to clearly demonstrate an understanding that spatial locality is used to anticipate which memory locations are likely to be needed by the processor next. In other words, the cache uses locality of reference in order to guess what words will soon be needed. In addition to this, other mistakes were:

- Defining spatial locality without explaining how it is used to increase the cache hit rate.
- Writing inappropriately about 'frequently used addresses'. For example: *Cache memory uses spatial locality to increase the cache hit rate by storing addresses that are near to frequently used addresses*.
- Giving answers that lacked clarity for example: *Spatial locality means that information that is related is stored in memory locations that have a close proximity in order to speed up the cache*.
- Writing more than one sentence. Two sentences was acceptable, but more than that meant a loss of credit.

Often it seemed that students were guessing in their answers, even though locality of reference is explained in section **4.7.1 Locality of reference and cache memory** in volume 1 of the subject guide.