## CS2100 Midterm Test

AY2024/25 Semester 2

This is the report for the CS2100 midterm test held on 12 March 2025.

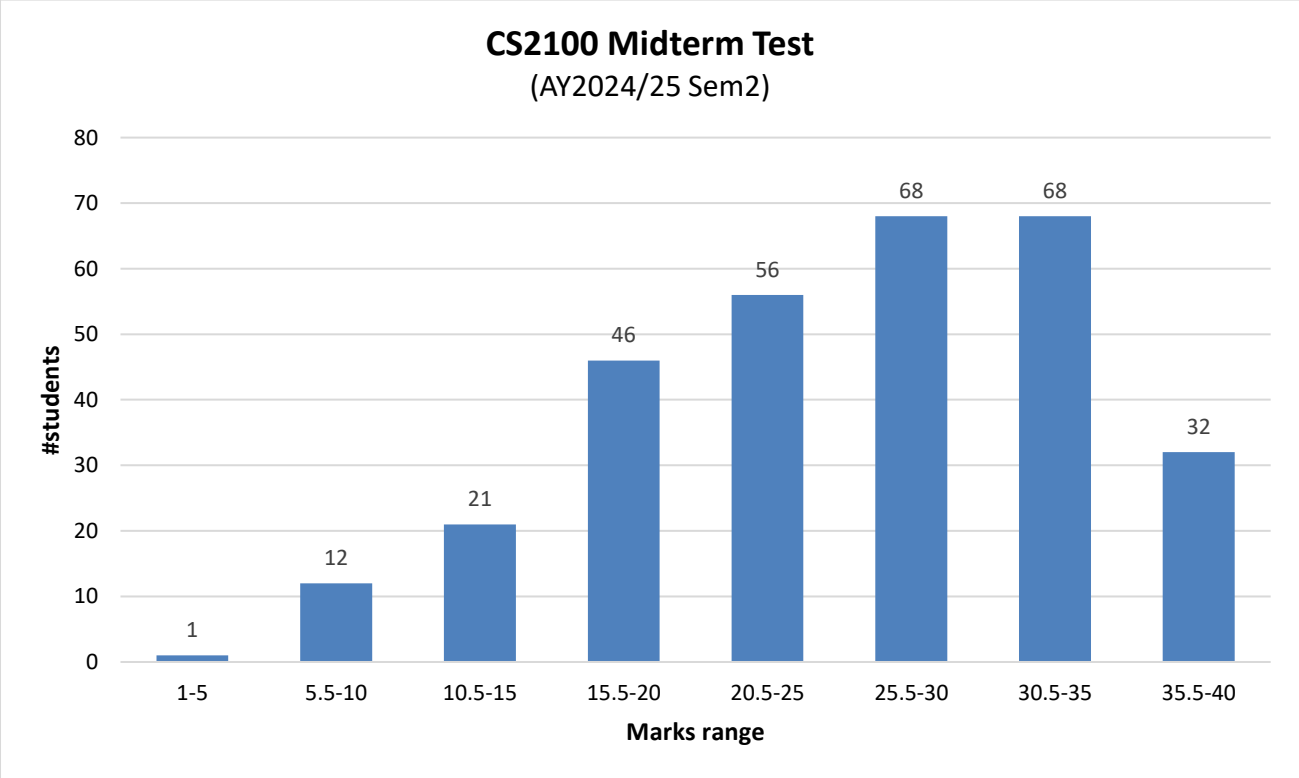
304 (out of 313) students sat for the midterm test.

### 1. General statistics

| | Part A (Q1-20) (20 marks) | Q21 (4 marks) | Q22 (4 marks) | Q23 (6 marks) | Q24 (6 marks) | **Total (40 marks)** |
|---|---|---|---|---|---|---|
| Average mark | 13.33 (66.6%) | 3.08 (76.9%) | 3.04 (76.1%) | 3.25 (54.1%) | 2.88 (48.1%) | **25.58** (63.9%) |
| Median mark | 13 | 3.5 | 3.5 | 3.5 | 3 | 26.25 |
| Standard deviation | 3.43 | 1.09 | 1.19 | 1.98 | 2.19 | 7.84 |

The average is 63.9% as compared against the average of 61.6% (AY2024/25 semester 1), 59.4% (AY2023/24 semester 2), and 54.4% (AY2023/24 semester 1).

Below is the chart for the overall results.



## 2. Who to contact?

If you have any issues, please contact the respective grader. You may attach a screenshot of your script in your email, or give us your Student Number so that we can retrieve your script on SoftMark.

- MCQs, Q23, Q24: Aaron Tan (tantc@comp.nus.edu.sg)
- Q21, Q22: Prabhu (prabhu.n@nus.edu.sg)

## 3. Part A: MCQs 1 – 20

The table below shows the percentage of students who chose the correct answers, and of those who chose the most popular wrong answers:

| | Q1 | Q2 | Q3 | Q4 | Q5 | Q6 | Q7 | Q8 | Q9 | Q10 |
|---|---|---|---|---|---|---|---|---|---|---|
| | 3rd hardest | 2nd hardest | Hardest | Easiest | | | 2nd easiest | | | |
| %students who chose the correct answer | D (39.5%) | E (36.8%) | A (31.3%) | E (96.7%) | C (71.4%) | B (81.9%) | C (84.2%) | D (47.7%) | B (62.5%) | B (80.9%) |
| %students who chose the most popular wrong answer | E (22.0%) | C (25.0%) | D (23.7%) | B (2.0%) | B (12.2%) | E (9.5%) | E (6.3%) | C (40.1%) | C (21.1%) | A (7.6%) |

| | Q11 | Q12 | Q13 | Q14 | Q15 | Q16 | Q17 | Q18 | Q19 | Q20 |
|---|---|---|---|---|---|---|---|---|---|---|
| | | 3rd easiest | | | | | | | | |
| %students who chose the correct answer | D (79.6%) | C (83.6%) | E (64.1%) | D (74.7%) | A (80.3%) | D (55.3%) | A (48.0%) | D (64.1%) | B (69.1%) | C (80.6%) |
| %students who chose the most popular wrong answer | A/C (7.9%) | E (8.2%) | C (18.8%) | B (8.6%) | B (10.2%) | E (25.7%) | B (38.1%) | C (15.1%) | A (12.8%) | E (7.2%) |

We put the 3 hardest questions in front to test if students are applying the correct strategy – to read through all questions and attempt the easier ones first.

The explanation for some of the MCQs are as follows:

Q1. Option A: $0011_{1s} - 1010_{1s} = 0011_{1s} + 0101_{1s} = 1000_{1s}$, which is an overflow in 1's complement.

Option B: $0010_{2s} - 0101_{2s} = 0010_{2s} + 1011_{2s} = 1101_{2s}$. No overflow.

Option C: $1101_{2s} + 1011_{2s} = 1000_{2s}$. No overflow.

Option D: $1110_{1s} + 1001_{1s} = 1000_{1s}$. No overflow in 1's complement.

$1110_{2s} + 1001_{2s} = 0111_{2s}$. Overflow in 2's complement.

Q2. Each pair of characters before and after transfer maintains the same parity, preventing the single-bit parity scheme from detecting any errors.

Q3. First, we consider all binary representations with MSB 0, each representation has the same decimal value in Sign-and-Magnitude and 2's Complement. There are $2^{n-1}$ of them.

Next, consider binary representations with MSB 1.

Given an n-bit representation X with MSB 1, we find the decimal values it represents in SM, 2's Complement, and Excess-$(2^{n-1} + 2^{n-2})$.

In SM, it represents $-(X - 2^{n-1}) = 2^{n-1} - X$

- We obtain the expression by first removing MSB from X, then negating the remaining values.

In 2's Complement, it represents $X - 2^n$

- Given a positive number Y, we know $-Y = (2^n - Y)_{2s}$. Let X be the representation $2^n - Y$, then the decimal value $-Y = X - 2^n$

In Excess $(2^{n-1} + 2^{n-2})$, it represents $X - 2^{n-1} - 2^{n-2}$

Let $2^{n-1} - X = X - 2^n$, we have $2X = 2^n + 2^{n-1}$, $X = 2^{n-1} + 2^{n-2}$.
Let $2^{n-1} - X = X - 2^{n-1} - 2^{n-2}$, we have $2X = 2^n + 2^{n-2}$, $X = 2^{n-1} + 2^{n-3}$.
Let $X - 2^n = X - 2^{n-1} - 2^{n-2}$, we have $2^n = 2^{n-1} + 2^{n-2}$, which is reduced to = 3.

Hence there are $2^{n-1} + 2$ such binary representations.

Q4. $1201_5 = 1 \times 5^3 + 2 \times 5^2 + 0 \times 5^1 + 1 \times 5^0 = 176$
$246_b = 2b^2 + 4b + 6.$
Solving $2b^2 + 4b + 6 = 176$ gives $b = 8.27$.

Q5. It suffices to compare only the fractional part:
$0.2425_{10} = 0.0201\ldots_3 \approx 0.020_3 = 0.222222_{10}$
$0.2425_{10} = 0.0332\ldots_4 \approx 0.100_4 = 0.25_{10}$
$0.2425_{10} = 0.1101\ldots_5 \approx 0.110_5 = 0.24_{10}$

Q6. $-12.375 = -1100.011_2 = -1.100011_2 \times 2^3$.
Sign bit = 1; exponent: $3 + 127 = 10000010_2$;
mantissa = 10001100000000000000000
1 10000010 10001100000000000000000 = 0xC1460000.
If students use 0 for sign, they will choose option (A).
If students use 3 for exponent, they will choose option (C).
If students use 1100011 for mantissa, they will choose option (D).

Q7. 0x41A800 = 0 10000011 010 1000 0000 0000 0000 0000;
Sign bit = 0; exponent = 131 – 127 = 4
Value = $1.0101_2 \times 2^4 = 10101_2 = 21_{10}$
If students use .0101 for mantissa, they will choose (A).
If students use 131 for exponents, they will chose (D).

Q8.

- $t0 and $t1 are both initialized to -1 (0xFFFFFFFF).
- Each iteration, add $t0, $t0, $t1 subtracts 1 from $t0. The value changes from -1 to -2,147,483,648. Further -1 causes the underflow that changes to 2,147,483,647. Again, further decrements reduced values from 2,147,483,647 to 0.
- Starting from 0xFFFFFFFF, subtracting 1 repeatedly takes 4294967295 iterations to reach 0, i.e., -2,147,483,647 → 0 → 2,147,483,647 = 4,294,967,295 iterations
- Thus, the loop executes 4294967295 times before exiting.

Q9.

- Jump instruction uses a 26-bit immediate shifted left by 2, and it combines that with the upper 4 bits of PC+4.
- Here, the jump instruction is at 0x4FFFFFFC, so the first 4 bits of PC+4 is 0101.
- Hence (iii) and (iv) are reachable but not (i) and (ii).

Q10. The C expression (v >> 4) | (v << 28) rotates the 32-bit value in v right by 4 bits by:

- Shifting v right by 4 bits, moving the upper 28 bits to the right.
- Shifting v left by 28 bits, moving the lower 4 bits to the left (to fill the vacant high-order bits).
- Combining these two results using the bitwise OR.

In MIPS:

- srl $t1, $t0, 4 shifts $t0 right by 4 bits logically, storing result in $t1.
- sll $t2, $t0, 28 shifts $t0 left by 28 bits, storing the result in $t2.
- Finally, or $t1, $t1, $t2 performs a bitwise OR to combine the shifted values.

Q12. arr[1].value is incremented from 20 to 25.

Q13. Maximum: we reserve one opcode for A, and B will then have $(2^5 - 1)\times 2 = 62$ opcodes. Hence the total is 63.
Minimum: we reserve $2^5 - 1 = 31$ opcodes for A. Then B will have 2 opcodes. Hence the total is 33.

Q15. Output:

0.142857
0.571429
1.857143

Q18. The code writes into *A*[9], *A*[8], …, *A*[1] then exits. We do not know what's in *A*[0].

Q19. *A*[5]=6, *A*[6]=7.

Q20. $2 + 9\times 5 + 2 = 49$.

## 4. Comments on Q21 (graded by Prabhu)

The average mark for Q21 is 3.08/4, or 76.9%. Percentage of students who got the individual questions correct as shown in the table:

| | (a) | (b) |
|---|---|---|
| %students who got it right | 69.4% | 41.8% |

For Q21(a) and (b), I have given 1 mark if the first half or second half of the answers is correct. I have given 1.5 marks if there is a mistake in 1 hexadecimal digit in the answer. Some students interchanged the rs and rt registers in the final answer.

## 5. Comments on Q22 (graded by Prabhu)

The average mark for Q22 is 3.25/4, or 76.1%. Percentage of students who got the individual questions correct as shown in the table:

| | (a) $t0 | (a) $t1 | (b) $t0 | (b) $t1 |
|---|---|---|---|---|
| %students who got it right | 75.0% | 89.5% | 65.1% | 57.6% |

For Q22(a)(i) and Q22(a)(ii), I have reduced 0.5 marks if the first half or second half of the answers is correct. I have also given 0.5 in total to Q22(a)(ii), if the answers for (i) and (ii) are interchanged.

Same rubrics has been applied for Q22(b)(i) and Q22(b)(ii).

Some of the students failed to sign-extend the immediate operands to 32-bits. Some of them didn't do the 2's complement representation of the negative numbers.

**6. Comments on Q23** (graded by Aaron)

The average mark for Q23 is 3.25/6, or 54.1%. Percentage of students who got the individual questions correct as shown in the table:

| | ❶ | ❷ | ❸ | ❹ | ❺ | ❻ |
|---|---|---|---|---|---|---|
| %students who got it right | 58.2% | 20.1% | 66.4% | 64.5% | 38.8% | 72.4% |

The working are as follows:

❶ 24 = 0x0000 0018; -24 = 0xFFFF FFE8.

❷ PC+4 + (Immed x 4) = 0x0040 0054 + (0xFFFF FFE8 << 2)
= 0x0040 0054 + 0xFFFF FFA0 = 0x003F FFF4

❸ PC+4 = 0x0040 0054

❹ $s1 = 0x0000 8888

❺ ALU result = $t1 + immed = 0x1001 0014 + 0xFFFF FFE8 = 0x1000 FFFC

❻ 0b0010 (ALU control for add operation)

For ❶, some students zero-extended the value -24 instead of sign-extending it, writing 0x00000000E8 as the answer.

For ❷, some students did one or combination of mistakes: using PC instead of PC+4, wrongly calculated immed $\times$ 4, or added PC+4 with immed $\times$ 4 wrongly. This output has the lowest number of students getting it right.

For ❸, this is just PC + 4, so it's rather straight-forward, but only 2/3 of the class got it right.

For ❹, it is the value in $s1. Many students wrote the value of $t1, which is 0x10010014 instead. This means they mixed up the positions of rs and rt in the instruction format – rs is $t1 and rt is $s1 here.

For ❺, some students wrote 0x00008870, which is the sum of $s1 and -24. This is the mistake of using $s1 instead of $t1, the same mistake for ❹.

For ❻, this is very straight-forward, the ALU control for add operation. Some students wrote a 1-bit answer, some wrote an 8-bit answer, which I don't understand.

**7. Comments on Q24** (graded by Aaron)

The average mark for Q24 is 2.88/6, or48.1%. Percentage of students who got the individual questions correct as shown in the table:

| | $t0 | $t1 | $t2 | $t3 | $t4 |
|---|---|---|---|---|---|
| %students who got it right | 42.1% | 68.8% | 67.8% | 32.2% | 33.2% |

The working are as follows:

(a) **lw $t0, 0($s0)**
Loads a 32-bit word from address 0x10010000. The bytes at 0x10010000–0x10010003 are: FF, 12, 34, 56. Thus, $t0 = 0xFF123456.

(b) **lb $t1, 2($s0)**
Loads the byte at 0x10010002, which is 0x34. Since 0x34 is positive (MSB = 0), it is sign-extended to $t1 = 0x00000034.

(c) **lb $t2, 4($s0)**
Loads the byte at 0x10010004, which is 0x78. Sign-extension yields $t2 = 0x00000078.

(d) **lb $t3, 7($s0)**
Loads the byte at 0x10010007, which is 0xDE. As an 8-bit signed value, 0xDE is negative (since 0xDE ≥ 0x80), so it is sign-extended to 0xFFFFFFDE.

(e) **lb $t4, 8($s0)**
Loads the byte at 0x10010008, which is 0xF0. As an 8-bit signed value, 0xF0 is negative (since 0xF0 ≥ 0x80), so it is sign-extended to 0xFFFFFFF0.

Common mistakes include writing only two hexadecimal digits, when the question explicitly states “Fill in ALL the 8 hexadecimal digits”, and not sign-extending the value to 32 bits.

Note that as MIPS can be big-endian as well as little-endian, I have accepted answers for both, for example, 0x563412FF for $t0.

Prepared by Aaron Tan
Date: 17 March 2025