Computer Architecture I Mid-Term II

Chinese Name: ____________________

Pinyin Name: ____________________

Student ID: ____________________

E-Mail ... @shanghaitech.edu.cn: ____________________

Question:	1	2	3	4	5	6	7	8	9	Total
Points:	1	18	14	21	6	12	7	12	9	100
Score:										

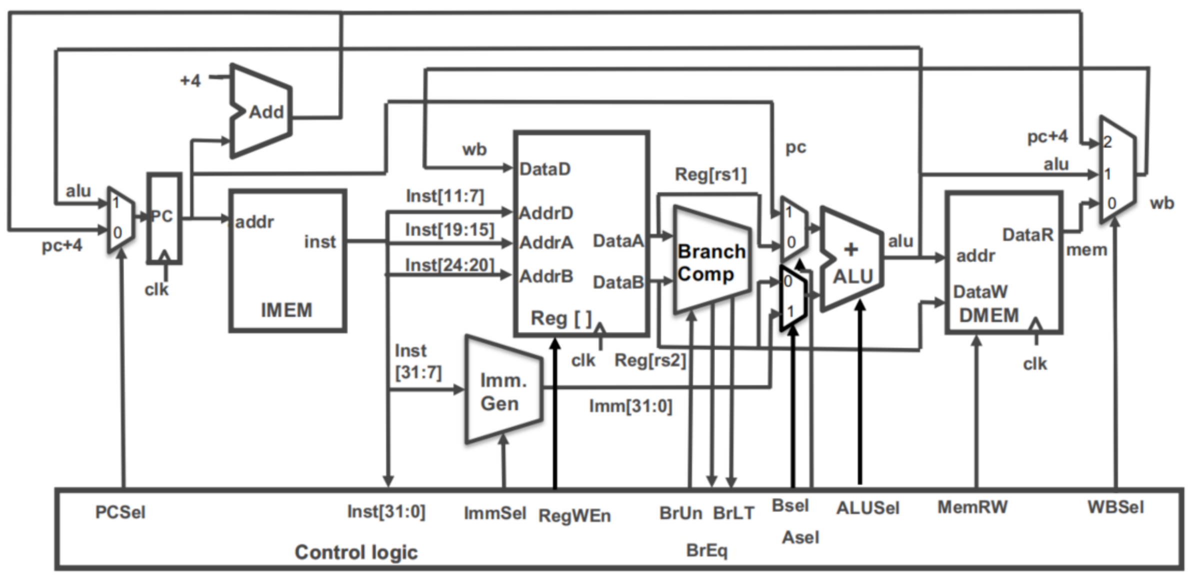
- This test contains 22 numbered pages, including the cover page, printed on both sides of the sheet.
- We will use Gradescope for grading, so only answers filled in at the obvious places will be used.
- Use the provided blank paper for calculations and then copy your answer here.
- Please turn **off** all cell phones, smartwatches, and other mobile devices. Remove all hats and headphones. Put everything in your backpack. Place your backpacks, laptops and jackets out of reach.
- Unless told otherwise always assume a 32bit machine.
- The total estimated time is 120 minutes.
- You have 120 minutes to complete this exam. The exam is closed book; no computers, phones, or calculators are allowed. You may use two A4 pages (front and back) of handwritten notes in addition to the provided green sheet.
- There may be partial credit for incomplete answers; write as much of the solution as you can. We will deduct points if your solution is far more complicated than necessary. When we provide a blank, please fit your answer within the space provided.
- Do **NOT** start reading the questions/ open the exam until we tell you so!

1 1. **First Task (worth one point): Fill in you name**

Fill in your name and email on the front page and your ShanghaiTech email on top of every page (without @shanghaitech.edu.cn) (so write your email in total 22 times).

2. **RISC-V Datapath**

The following diagram is the RISC-V single-cycle control datapath.



6 (a) In the RISC-V datapath above, what is used for the `jalr` instruction. Some questions may have more than one answer, please select (fill in table below) **all** that apply.

1. **PCSel Mux**:

   A. `pc + 4` branch
   B. `alu` branch
   C. Input dependent
   D. * (don't care)

2. **ASel Mux**:

   A. `pc` branch
   B. `Reg[rs1]` branch
   C. Input dependent
   D. * (don't care)

3. **BSel Mux**:

   A. `imm` branch
   B. `Reg[rs2]` branch
   C. `mem` branch
   D. * (don't care)

4. **WBSel Mux**:
   A. `pc+4` branch
   B. `alu` branch
   C. `mem` branch
   D. * (don't care)

5. **Datapath units**:
   A. `Branch Comp`
   B. `Imm. Gen`

6. **RegFile**:
   A. Value read from `Reg[rs1]`
   B. Value read from `Reg[rs2]`
   C. Writing to `Reg[rd]`

1	2	3	4	5	6

Solution: B; B; A; A; B; AC.

6 (b) In the RISC-V datapath above, what is used for the `beq` instruction. Some questions may have more than one answer, please select (in the table below) **all** that apply.

1. **PCSel Mux**:
   A. `pc + 4` branch
   B. `alu` branch
   C. Input dependent
   D. * (don't care)

2. **ASel Mux**:
   A. `pc` branch
   B. `Reg[rs1]` branch
   C. * (don't care)

3. **BSel Mux**:
   A. `imm` branch
   B. `Reg[rs2]` branch
   C. * (don't care)

4. **WBSel Mux**:
   A. `pc+4` branch
   B. `alu` branch
   C. `mem` branch
   D. * (don't care)

5. **Datapath units**:
   A. `Branch Comp`
   B. `Imm. Gen`

6. **RegFile**:
   A. Read `Reg[rs1]`
   B. Read `Reg[rs2]`
   C. Write `Reg[rd]`

1	2	3	4	5	6

Solution: C; A; A; D; AB; AB.

6 (c) In the RISC-V datapath above, what is used for a `mv` instruction. Some questions may have more than one answer, please select **all** that apply.

Please notice that `mv` is a pseudo instruction, you are required to find and use the corresponding base instruction as specified on the green card.

1. **PCSel Mux**:

   A. `pc + 4` branch
   B. `alu` branch
   C. Input dependent
   D. * (don't care)

2. **ASel Mux**:

   A. `pc` branch
   B. `Reg[rs1]` branch
   C. * (don't care)

3. **BSel Mux**:

   A. `imm` branch
   B. `Reg[rs2]` branch
   C. * (don't care)

4. **WBSel Mux**:

   A. `pc+4` branch
   B. `alu` branch
   C. `mem` branch
   D. * (don't care)

5. **Datapath units**:

   A. `Branch Comp`
   B. `Imm. Gen`

6. **RegFile**:

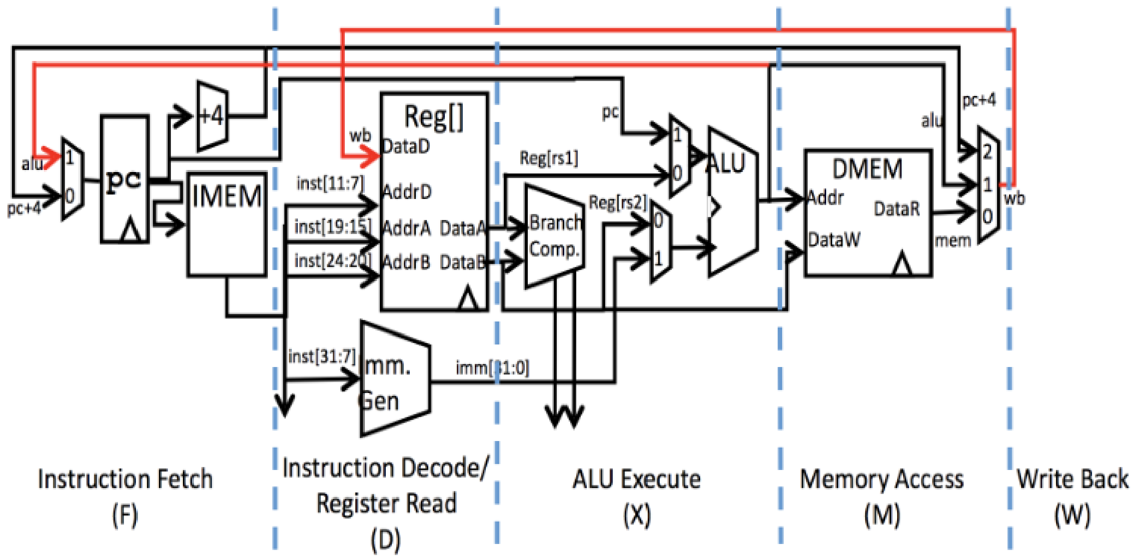
   A. Read `Reg[rs1]`
   B. Read `Reg[rs2]`
   C. Write `Reg[rd]`

1	2	3	4	5	6

Solution: A; B; A; B; B; AC.

3. **Hazardous Bit Fiddling**

Consider a typical 5-stage (Fetch, Decode, EXecute, Memory, WriteBack) pipeline. Assume pipeline registers exist where the dotted lines are.



For this question, note the following considerations:

- We **can** read and write from the same registers or memory address in the same clock cycle.
- No other optimizations are implemented in this datapath (unless explicitly stated in the question).

```
1  mystery:
2      srai    t0, a0, 31
3      add     a0, a0, t0
4      xor     a0, a0, t0
5      ret
6
7  mystery_alternative:
8      bge     __, __, end
9      sub     __, __, __
10 end:
11     ret
```

2 (a) How many hazard(s) are there in `mystery` (lines 1 to 5)? What kind(s) of hazard(s) are they?

Solution:
3, instruction (2, 3), (3, 4), (2, 4); Data Hazard.

Data hazards involve pairs of instructions. An instruction in itself cannot lead to data hazards. Also, instructions concerned with the same hazard may not be adjacent to each other.

1 (b) How many stalls would need to be added for the program to be executed correctly on the pipelined machine? (Ignore `ret`)

Solution:

```
mystery:
    srai    t0, a0, 31
    nop
    nop
    add     a0, a0, t0
    nop
    nop
    xor     a0, a0, t0
    ret
```

4

A typical data hazard stalls the pipeline for 3 cycles if we don't allow write and read to the register file in the same cycle and 2 cycles otherwise. We do not need to stall extra cycles for (`srai`, `xor`) since `t0` is already written to the register file before add is executed.

1 (c) Assuming that **forwarding is implemented**, count the total number of cycles it takes to complete `mystery` (excluding `ret`)

Solution: 3

2 (d) Try to walk through `mystery` with a few inputs and see what it outputs. Suggest a C function signature for `mystery` that best conveys its semantics. (function name and type, e.g. `type function_name(type param);`)

Solution:

`int abs(int a);`

- Function name should imply absolute value. input -1 output 1, input 2 output 2.

- We are dealing with signed 2's complement values, so the type of the parameter and return value should both be `int`
- According to the RISC-V calling convention, the first integer parameter is passed in a0, and integer return values are stored in a0 too.

[2] (e) Fill in the register operands in `mystery_alternative`, such that it performs the same functionality as `mystery`.

`bge` ______________________________

`sub` ______________________________

Solution:

```
1 mystery_alternative:
2     bge     a0, x0, end
3     sub     a0, x0, a0
4 end:
5     ret
```

[2] (f) How many hazard(s) are present in `mystery_alternative`? What kind(s) of hazard(s) are they?

____

Solution: 1; Control Hazard

[2] (g) Suppose that you decided to implement a branch predictor with the following configuration. A predicted branch takes 1 cycle and a mispredicted branch takes 5 cycles. Assuming the prediction accuracy is p, calculate on average how many cycles it takes to execute `bge` (line 8) in function `mystery_alternative`? (Write your answer as a formula containing p)

____

____

Solution: $p + (1 - p) \times 5 = 5 - 4 \times p$

[2] (h) Compare your answer in (g) against (c), under what condition would you favor the first function against the second function? (For simplicity, ignore the time it takes to execute the `sub` instruction. Give a range of p)

____

Solution:

$5 - 4 \times p > 3 \Rightarrow p < \frac{1}{2}$

Modern CPUs usually have much deeper pipelines and thus heavier branch misprediction penalty. Therefore the compiler attempts to aggressively optimize out simple branches, like the absolute value function. In this case, the optimization may not worth the effort because of the shallow pipeline. A random predictor predicts correctly 50% of the time.

The motivation for crafting this question is as following

- While this question mainly focuses on pipeline hazards, it also synthesize points from number representation and RISC-V convention.
- The 2 functions presented in this question each represents a kind of pipeline hazard and may perform differently under different hardware configuration, whose trade off should be made aware of by the students when doing performance engineering.

4. **Superscalar**

8 (a) This section involves T / F questions. Please fill your answer (**T** or **F**) in the table below.

1. A superscalar CPU can execute more than one process or thread at a given time.
2. The number of clock cycles a floating point multiplier needs depends on the values of the operands.
3. Bypassing can not prevent increased write back latency from slowing down single cycle integer operations.
4. Out-of-order superscalar processors exploit instruction-level parallelism and adds more complexity to the compiler.
5. Superscalar processors use multiple execution units for additional instruction level parallelism.
6. A superscalar processor can execute more than one instructions per clock cycle, it allows performance gain in latency at a given clock rate.
7. According to Flynn's Law, a single-core superscalar processor is classified as an SIMD processor.
8. All but simplest machines have out-of-order completion, due to different latencies of functional units and desire to bypass values as soon as possible.

1	2	3	4	5	6	7	8

Solution: F; F; F; F; T; F; F; T.

2 (b) Assume the execution latency of the longest-latency instruction in a 4-wide superscalar, out-of-order machine implementing one algorithm is 500 cycles.

How large should the instruction window be such that the decode of instructions does not stall in the presence of this longest-latency instruction?

Solution:

2000 instructions.

500 cycles/stall $\times$ 4 instructions to buffer every cycle $\Rightarrow$ Need a 2000-instruction-entry window.

2 (c) Assume your friend at a processor design company designed a 1-instruction-wide processor with out-of-order execution. Every instruction in this machine takes a single cycle.

What would you suggest to your friend to simplify the design of the above processor? Please explain yourself briefly.

Solution:

Execute in-order. Out-of-order execution will not improve performance because instructions have fixed latency and never stall.

2 (d) What is the definition of CPI? Please use an equation to show it.

Solution:

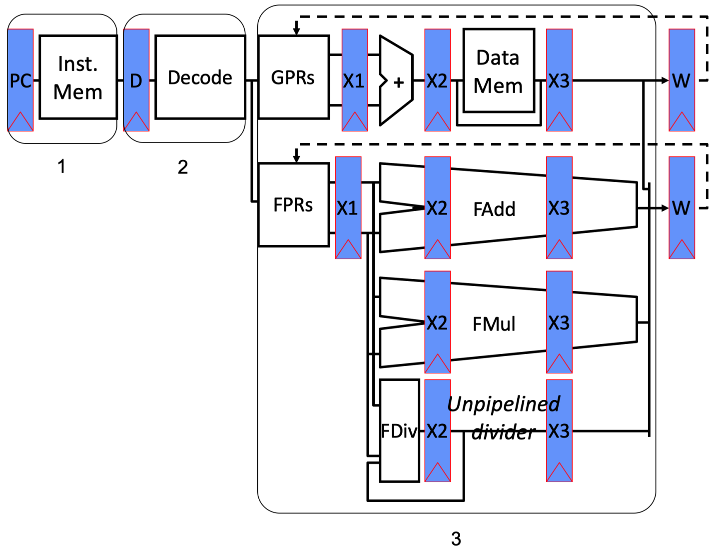
$$\begin{aligned}\text{CPI} &= \frac{\text{Cycles}}{\text{Instruction}} \quad \text{(1pt)} \\ &= \frac{\text{Time}}{\text{Program}} \div \left(\frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Time}}{\text{Cycle}}\right) \quad \text{(2pts)}\end{aligned}$$

2 (e) Calculate the CPI (cycle per instruction) of a program with following parameters.

Operation	Freq_i	CPI_i
ALU	45%	2
Load	30%	5
Store	15%	4
Branch	25%	3

Solution: $45\% \times 2 + 30\% \times 5 + 15\% \times 4 + 25\% \times 3 = 3.75$ (cycles).

5 (f) Here is a simplified datapath schematic diagram of a superscalar processor. Fill in the following blanks.



1. Issue buffer sits between stage ______________ and stage ______________.
2. Using this processor and fetching two instructions per cycle, it issues both simultaneously if one is ______________ and other is ______________.

Solution:

1. 2; 3.
2. integer/memory; floating point.

5. **Performance**

2 (a) A given program written in C runs 15 seconds on machine A. Suppose an optimized C compiler is released which compiles that program into 60% as much instructions as the old compiler. However, half of the instructions require 120% CPI than before. How long would the program complied by the newer compiler run on machine A? Give your calculation steps.

Solution:

$$15 \times 60\% \times (50\% + 50\% \times 120\%) = 9.9 \text{ sec}$$

2 (b) Consider an ISA that instructions can be divided into four different classes (A, B, C, D) according to their CPI. P1 with a clock rate of 2.5 GHz and CPIs of 1, 2, 3 and 3; and P2 with a clock rate of 3 GHz and CPIs of 3, 2, 2 and 2. Given a program that contains 1×10^6 instructions with 10% A, 20% B, 50 % C and 20% D.

1. What is the average CPI of that program for P1 and P2? Give your calculation steps.

2. Which processor runs faster for that program? Justify your answer.

Solution:

1. P1: $10\% \times 1 + 20\% \times 2 + 70\% \times 3 = 2.6$;

   P2: $10\% \times 3 + 90\% \times 2 = 2.1$.

2. $\frac{2.6}{2.5} > \frac{2.1}{3}$; P2 runs faster.

2 (c) Assume for arithmetic, load/store and branch instructions, a processor has CPIs of 1, 12 and 5. Also assume that on a single core processor a program requires 2.56×10^9 arithmetic instructions, 1.28×10^9 load/store instructions and 2.56×10^8 instructions. Assume that each processor core runs on 2GHz clock.

Say that the program is parallelized to run over multiple cores. The number of arithmetic and load/store instructions per core is divided by $0.7 \times p$ (where p is the number of cores) but the number of branch instructions per core remains the same. To what should the CPI of load/store instructions be reduced in order for a single core processor to match the performance of four core processors? Give your calculation steps.

Solution:

$$\begin{aligned} & 2.56 \times 10^9 \times 1 + 1.28 \times 10^9 \times \text{CPI}_{\text{new}} + 2.56 \times 10^8 \times 5 \\ = & \frac{2.56 \times 10^9}{0.7 * 4} \times 1 + \frac{1.28 \times 10^9}{0.7 * 4} \times 12 + 2.56 \times 10^8 \times 5 \end{aligned}$$

Therefore

$$\text{CPI}_{\text{new}} = 3$$

6. **Cache**

3 (a) We have an 8-bit address space and a 2-way set associative cache with properties as follows:

1. Cache size is 32 Bytes;
2. Block size is 8 Bytes;

Calculate the bit width of tag, index, and offset bits.

TAG	Set Index	Block Offset

Solution: 4, 1, 3.

6 (b) We will access the data of addresses as follows. Fill in the blanks. It is about T/I/O (tag/index/offset, write down the value in decimal), classify the access as a Hit, Miss or Replace. (each line worth 1 pt.)

Address	T/I/O	Hit, Miss or Replace
0b00000100		
0b00000101		
0b01101000		
0b11001000		
0b01101000		
0b11011101		

Solution:

Address	T/I/O	Hit, Miss or Replace
0b00000100	0/0/4	Miss
0b00000101	0/0/5	Hit
0b01101000	6/1/0	Miss
0b11001000	12/1/0	Miss
0b01101000	6/1/0	Hit
0b11011101	13/1/5	Replace

3 (c) Assume we have a single-level, 1 KiB direct-mapped L1 cache, whose bit width of tag, index, and offset bits are 22, 6, 4 separately. An integer is 4 bytes. The array is block-aligned. Given the following C source code, what is the hit rate?

```c
#define LEN 512

int array[LEN];
int main() {
    for (int i = 0; i < LEN; i += 128) {
        array[i] = 0;
    }
    for (int i = LEN - 128; i >= 0; i -= 128) {
        array[i] = 0;
    }
    return 0;
}
```

____

Solution: 1/4

7. **Multilevel Cache**

2 (a) This section involves T / F questions. Please fill your answer (**T** or **F**) in the table below. Incorrect answers on T / F questions are penalized with negative credit (in total no less than 0 point). Notice: NO selection will be treated as a wrong choice.

1. Using multi-level cache will increase miss penalty.
2. Non-inclusive cache may yields higher performance.
3. Prefetching can eliminate compulsory cache misses.
4. A misprediction in prefetching will affect correctness.

1	2	3	4

Solution: F; T; T; F.

3 (b) Suppose you have the following system that consists of an:

- L1 cache with a local hot rate of 80% and a hit time of 2 cycles;
- L2 cache with a global miss rate of 8% and a hit time of 15 cycles.

DRAM accesses take 50 cycles.

What is AMAT?: ______________________________

L2 cache local miss rate: ______________________________

AMAT of L1 cache: ______________________________

Solution:
Average memory access time (considering both hits and misses in the cache);
40%;
9.

2 (c) We want to improve AMAT of L1 cache, make sure that it will not greater than 6 cycles, by improving L2 cache's hit rate.

The minimum local hit rate for L2 cache to meet our requirement is:

____

____

Solution: 90%.

8. Data-level Parallelism

2 (a) A program spends **3%** of its time traversing the network, and **7%** of its time transferring data.If the new hardware speeds up the first part by a factor of 1.5 and also speeds up transmission by a factor of 1.75, what is the speed up of the whole program? Write down the original formula **without simplification**.

Solution: $\frac{1}{1-0.1+\frac{0.03}{1.5}+\frac{0.07}{1.75}}$

2 (b) Explain why loop unrolling can improve performance.

Solution: less loop overhead; it can avoid data hazards; SIMD instructions can be used

2 (c) Name one SIMD instruction set.

Solution: SSE; AVX.

6 (d) Use SIMD to speed up the calculation of sum of squares. You can use function given below. Convert pointer type when needed.

1. `__m128i _mm_load_si128(const __m128i *mem_addr);`
   Load 128 bits from `mem_addr` to a `__m128i` variable.
2. `__m128i _mm_mullo_epi32(__m128i a, __m128i b);`
   Multiply corresponding 32-bit integers in `a` and `b` respectively, and return `__m128i` variable containing four 32-bits integers.
3. `__m128i _mm_add_epi32(__m128i a, __m128i b);`
   Add corresponding 32-bit integers in `a` and `b`, and return `__m128i` variable containing four 32-bits integers.

```
/*  a is an array pointer, n is number of element in the array.
    No tail case in this question. (n is multiple of 4) */
int sum_of_square(int *a, int n) {
    int ans[4];
    __m128i batch = _mm_setzero_si128();   /* set all bits to 0 */
    __m128i temp_square = _mm_setzero_si128();
    __m128i result = _mm_setzero_si128();
    for  (int i = 0; i < N; i += 4) {

```

```
            batch = ______________________________________________;

            temp_square = ________________________________________;

            result = _____________________________________________;
    }
    /* store the vectorization result to int array */
    _mm_storeu_si128((__m128i *) ans, result);

    return ans[0] + ans[1] + ans[2] + ans[3];
}
```

Solution:

```
_mm_load_si128((__m128i *) (a+i))
_mm_mullo_epi32 (batch, batch)
_mm_add_epi32(result, temp_square)
```

9. **OpenMP Intro**

We try to accelerate the calculation of Frobenius Norm of a matrix under the assistance of **OpenMP**. Read the following code.

```
1 #include <omp.h>
2 #include <math.h>
3 /*  Given a matrix ‘mat_a‘ of size m * n, calculate its Frobenius norm.
4     Hint: mat_a[i][j] := *((double *) mat_a + n * i + j)  */
5 double frobenius_norm(double **mat_a, int m, int n) {
6     omp_set_num_threads(4);
7     double norm = 0.0;
8     int i, j = 0;
9     #pragma omp parallel for private(j)
10    for (i = 0; i < m; i++) {
11        for (j = 0; j < n; j++) {
12            norm += pow(*((double *) mat_a + n * i + j), 2);
13        }
14    }
15    return sqrt(norm);
16 }
```

3 (a) Identify the data sharing attributes of the following variables with `shared` or `private`.

`norm` ____________

`i` ____________

`j` ____________

Solution: shared; private; private

2 (b) What is wrong with the code?

Solution:
The summation of shared variable *norm* happens inside parallel section; a *data race* may happen.

2 (c) Fix the bug using `reduction(operation: var)`. (You may want to modify a line of code or insert a new line of code. Clearly specify the line id, then write down the new line of code)

Solution:
Line 9. `#pragma omp parallel for private(j) reduction(+: norm)`

2 (d) Fix the bug using `#pragma omp critical`. (You may want to modify a line of code or insert a new line of code. Clearly specify the line id, then write down the new line of code)

Solution:
Between Line 11 and Line 12. `#pragma omp critical`

No question here!