Student Name Chinese: ____________________

Student Name Pinyin: ____________________

Student Email: ____________________ **(Shanghaitech email w/o "@shanghaitech.edu.cn")**

Student Id: ____________________

School: ____________________

Year of Entrance: ____________________

ShanghaiTech University Midterm II Examination Cover Sheet

May 11 2021

Academic Year : ___2020___to___2021__ Term: __Spring______

Course-offering School: ____SIST___________

Instructor: Sören Schwertfeger & Chundong Wang

Course Name: Computer Architecture I

Course Number: CS110

Exam Instructions for Students:

1. All examination rules must be strictly observed throughout the entire test, and any form of cheating is prohibited.

2. Other than allowable materials, students taking closed-book tests must place their books, notes, tablets and any other electronic devices in places designated by the examiners.

3. Students taking open-book tests may use allowable materials authorized by the examiners. They must complete the exam independently without discussion with each other or exchange of materials.

4. Unless told otherwise always assume a 32bit machine.

For Marker's Use:

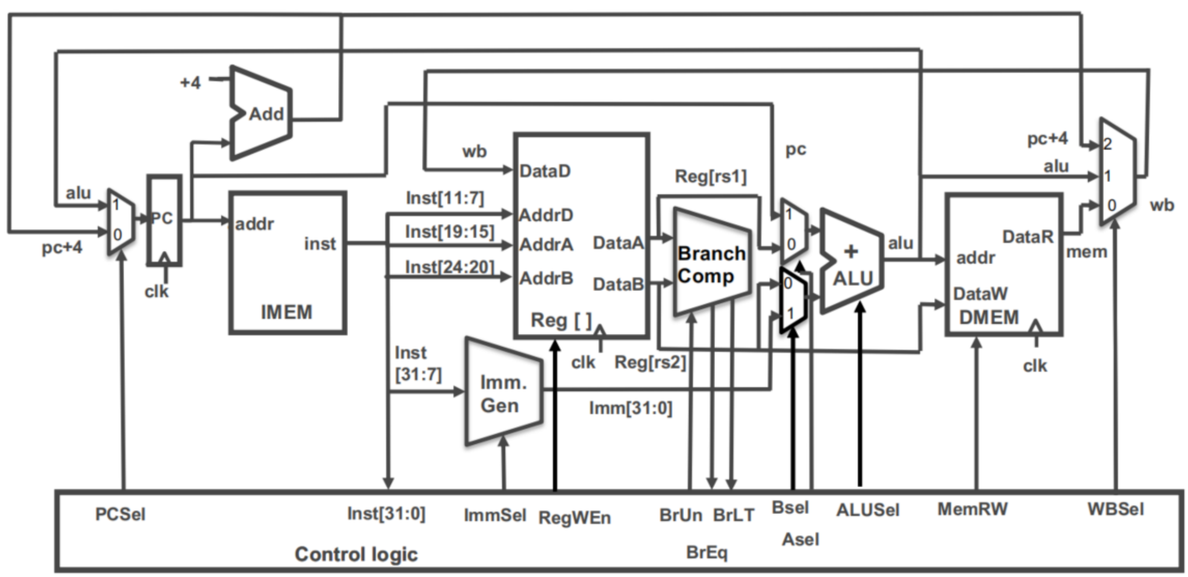
Section	1	2	3	4	5	6	7	8	9	Total
Max	1	18	14	21	6	12	7	12	9	100
Marks										
Recheck										

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 13 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

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

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.

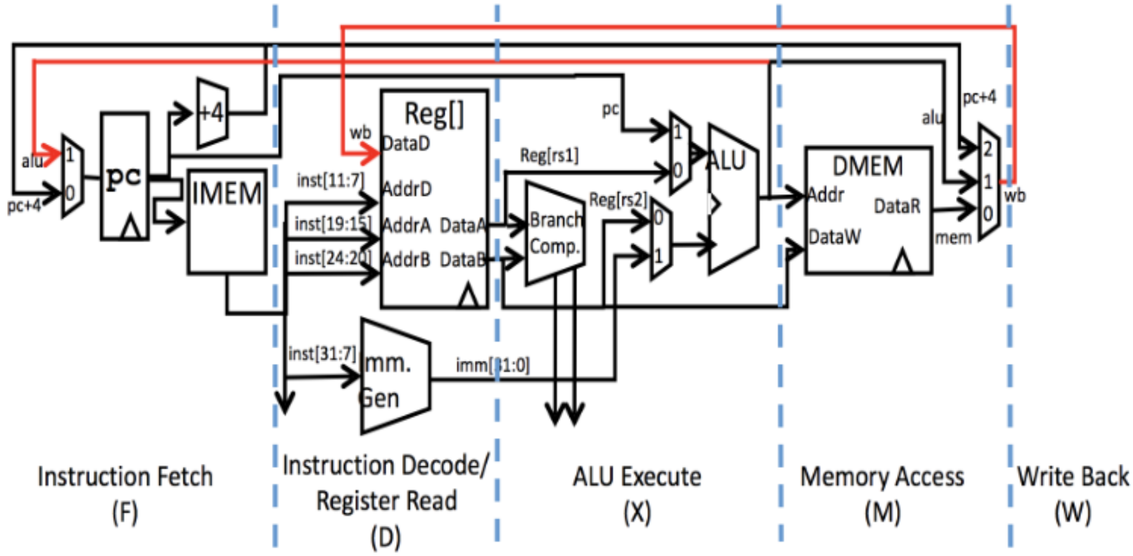
Please notice that `mv` is a pseudo instruction, you are required to find and use the corresponding base instruction as specified on the green card.

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

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?

____

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

____

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

___

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);`)

___

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

`bge` ___

`sub` ___

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

___

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)

___

___

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)

___

___

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

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?

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.

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

____

____

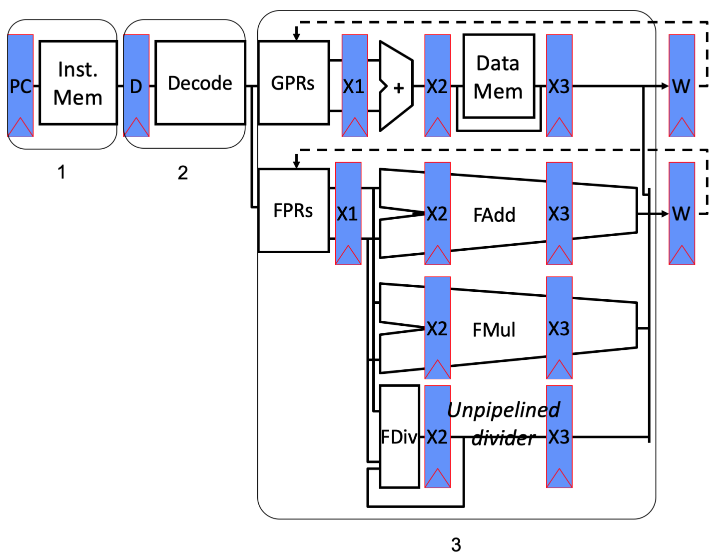
2 (e) Calculate the CPI (cycle per instruction) of a program with following parameters.

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

____

____

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 ______________.

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.

___

___

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.

   ___

   ___

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.

___

___

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

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		

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;
}
```

7. **Multilevel Cache**

[2] (a) This section involves T / F questions. Incorrect answers on T / F questions are penalized with negative credit (in total no less than 0 point). Circle the correct answer. 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

[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: ______________________________

[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:

____

____

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**.

____

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

____

[2] (c) Name one SIMD instruction set.

____

[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];
}
```

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` ____________________

[2] (b) What is wrong with the code?

____

____

[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)

____

____

[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)

____

____

No question here!