Computer Architecture I Final

Chinese Name: ______________________________

Pinyin Name: ______________________________

Student ID: ______________________________

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

Question	Points	Score
1	1	
2	14	
3	7	
4	5	
5	11	
6	8	
7	6	
8	7	
9	7	
10	20	
11	12	
12	2	
Total:	100	

- This test contains 14 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.
- 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 RISC-V 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!
- Unless otherwise stated, always assume a 32-bit machine for this exam.

1 1. First Task (worth one point): Fill in you name
Fill in your name, email and student ID on the front page and your ShanghaiTech email on top of every following page (without @shanghaitech.edu.cn) (so write your email in total 14 times).

2. General questions

[2] (a) Using IEEE 754 representation, what decimal number is encoded?

0x41860000 → ______________________________

Convert the following decimal number to single precision IEEE 754 floating point format (to hexadecimal form):

-21.75 → 0x______________________________

[2] (b) Convert 2333_{ten}

To Binary ______________________________

To Hexadecimal ______________________________

[2] (c) How does the CPU communicate with these I/O devices (as introduced in the lecture)?

Mouses: ____________________ Disks: ____________________

Unicode is a powerful coding scheme that can represent almost every character, e.g. the codepoint for emoji "Ear of Maize" (🌽) is U+1F33D. However, it takes too much space for characters like ASCII characters. Thus, variable-length encoding schemes like UTF-8 were developed. In the following 3 questions, suppose address space grows from left to right, i.e. low address is at left and high address is at right.

[2] (d) The corn emoji is represented by a two-byte surrogate pair: 0xd83c and 0xdf3d in UTF-16. How are they stored in hexadecimal in a big-endian machine and a little-endian machine? Choose the correct answer.

A. `0xd8 0x3c 0xdf 0x3d` B. `0xdf 0x3d 0xd8 0x3c`
C. `0x3d 0xdf 0x3c 0xd8` D. `0x3c 0xd8 0x3d 0xdf`

Big-endian: ____________________ Little-endian: ____________________

Consider the following C program:

```c
#include <stdio.h>
#include <stdint.h>
#define SLL(x,k) (x << k)
#define XOR(x,y) (x ^ y)

void f (char *s, float *x){
      while (*s) {
            printf("%d.", (*s++)-'a');
      }
      printf("%f\n", *x);
}

```

```
13 int main (){
14         int i = SLL (2,2) - XOR (1,2);
15         char *p = "it_is_weird";
16         float x = 6.25e-1;
17         f(p, &x);
18         return 0;
19 }
```

2 (e) Which part of memory do the following variables belong to? Choose the correct answer from ”stack”, ”heap” or ”data”.

`"it_is_weird"`: ____________________ `p`: ____________________

2 (f) What is the value of `i`?

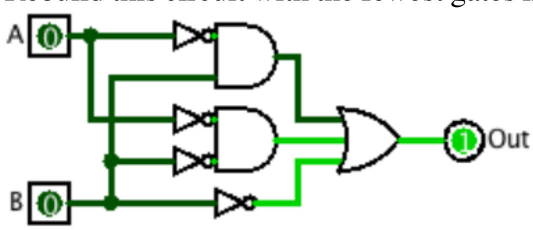
____

2 (g) What is the type of `&x` in line 17?

____

3. **SDS**

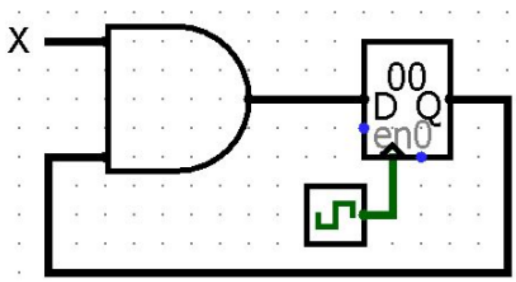
3 (a) Rebuild this circuit with the fewest gates in the box, using **only** AND, OR and NOT gates:



4 (b) For the circuit below, assume:

- setup time is 15ns
- hold time is 30ns
- AND gate delay is 10ns

If the clock rate is 10 MHz and x updates 25ns after the rising edge of the clock, what are the minimum and maximum values for the clk-to-Q delay to ensure proper functionality?



Min:__________ ns

Max:__________ ns

4. **CALL**

2 (a) Among all four types of addresses (1. PC-Relative Address; 2. Absolute Function Address; 3. External Function Reference; 4. Static Data Reference), which of the types need to be relocated in the linker?

____

3 (b) Please give the correct order of the loader.

1. Copies arguments passed to the program onto the stack
2. Jumps to start-up routine that copies programs arguments from stack to registers and sets the PC
3. Creates new address space for program large enough to hold text and data segments, along with a stack segment
4. Copies instructions and data from executable file into the new address space
5. Reads executable files header to determine size of text and data segments
6. Initializes machine registers

____

5. **Datapathology**

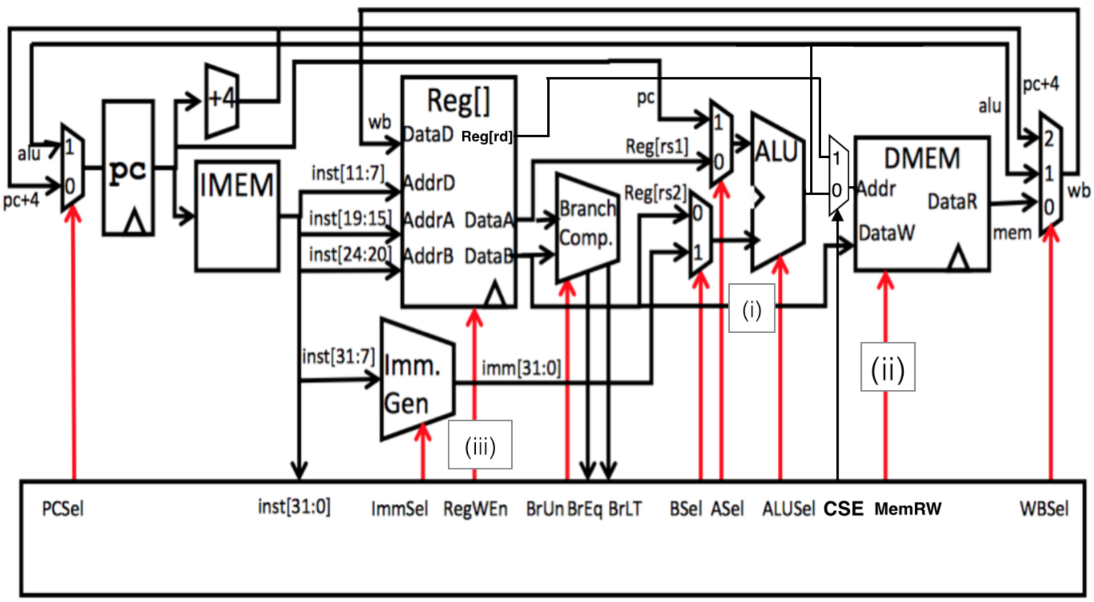
Consider adding the following instruction to RISC-V:

Instruction	Operation
cse rd, rs1, rs2	if (R[rs1]!= R[rs2]) R[rd] = R[rs1] - R[rs2]; else Mem[R[rd]] = 1;

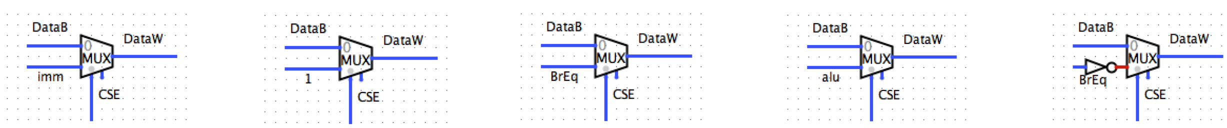
1 (a) What type of instruction will cse be? If multiple formats work, choose all that apply.
a. R-type b. I-type c. S-type d. U-type

(a) ____

6 (b) Implement cse in the datapath. Choose all correct implementations for (i), (ii), (iii).
Note 1: The control signal CSE is 1 if and only if the instruction is cse, 0 otherwise.
Note 2: The RegFile in the below datapath has one additional out-port Reg[rd], which outputs the value at AddrD from the RegFile.
Make sure the functions of the original RISC-V CPU are still preserved!

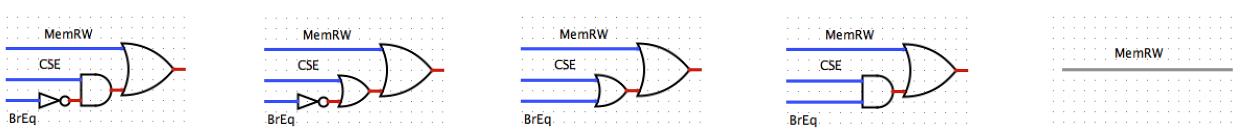


i. Select the correct component to fill in blank (i) in the Execution Stage.



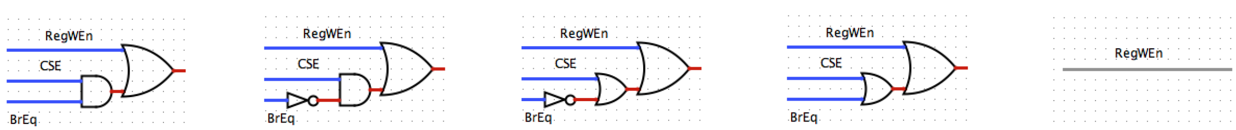
a b c d e

ii. Select the correct component to fill in blank (ii) in the Memory Stage.



a b c d e

iii. Select the correct component to fill in blank (iii) in the Write Back Stage.



a b c d e

4 (c) Fill in the correct control signals for cse. You may assume the immSel signal is correctly implemented. The possible values for each signal are given below. If the exact value of that signal doesn't matter, then select Don't Care (X). If it is possible for a signal to be "Don't Care", then select "Don't Care" instead of a more specific value (e.g. a value of (0) or (1) is correct when the signal could instead be (X)).
(0)Signal=0; (1)Signal=1; (2)Signal=2; (R)Write Disabled; (W)Write Enabled; (X)Don't Care; (A)AND; (B)OR; (C)ADD; (D)SUB

CSE	PCSel	RegWEn	BrUn	BSel	ASel	ALUSel	MemRW	WBSel
1								

6. **Pipelining**
Consider the standard 5-stage pipelined RISC-V CPU with instruction fetch, register read, ALU, memory, and register write stages. Register writes happen before register reads in the same clock cycle, branch comparison is done during the register read stage, there is a branch delay slot, and forwarding is implemented.
For the following stream of instructions, assume that t0 is not equal to 0, so the branch is not taken.

```
start: lw t0 0(a0)
       beq t0, 0, end
       addi t0, t0, 10
       sw t0 0(a0)
end:
```

Logic in each stage of the pipeline has the following timing:

Instruction Fetch	Register Read	ALU	Memory	Register Write
150ps	100ps	100ps	200ps	100ps

The pipelining registers in between stages have the following timing:

Clock-to-Q	Hold time	Setup
30ps	20ps	30ps

6 (a) For each pair of instructions, write down whether the CPU needs to be stalled for the execution of the second instruction, and if so, for how many cycles.
(i)

```
start: lw $t0 0($a0)
       beq $t0, 0, end
```

(ii)

```
beq $t0, 0, end
addi $t0, $t0, 10
```

(iii)

```
addi $t0, $t0, 10
sw $t0 0($a0)
```

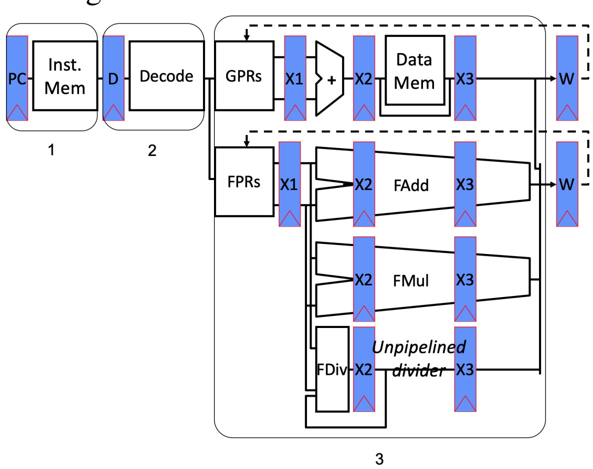
2 (b) What is the minimum clock period, in picoseconds, with which the processor can run?

7. **Superscalar Processors**

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

Operation	Freq_i	CPI_i
ALU	50%	2
Load	20%	6
Store	10 %	4
Branch	20 %	3

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



1. Fetch buffer sits between stage __________ and stage __________.

2. In each cycle, fetch two instructions; issue both simultaneously if one is __________ and other is __________.

8. Vectorization

[4] (a) You are required to compute the sum of all elements in a vector, that is sum $= \sum_{i=0}^{n} a_i$. The `int` vector is stored in an array whose address is aligned to a multiple of 16 bytes. Suppose `n` is a multiple of 4 and the sum does not cause overflow. Finish the following C code using the SSE Intrinsics. All variables have been declared for you and you **should not** declare more variables. You might find the following intrinsics useful:

- `__m128i _mm_add_epi32 (__m128i a, __m128i b)`: add packed 32-bit integers
- `__m128i _mm_srli_si128 (__m128i a, int imm8)`: shift `a` right by `imm8` bytes while shifting in zeros
- `int _mm_cvtsi128_si32 (__m128i a)`: Copy the lower 32-bit integer in `a` to `dst`.

```
int sum (int *a, int n) {
    __m128i result = _mm_loadu_si128 ((__m128i const *) a), tmp;

    for (int i = ______________________________ ) {

        ________________________________________

        ________________________________________
    }

    ____________________________________________

    ____________________________________________

    ____________________________________________

    ____________________________________________

    ____________________________________________
}
```

[1] (b) Which type of Flynn Taxonomy does SSE fit in ? Give its full name.

____

[2] (c) Can the previous program achieve $4.5\times$ speed up compared with the linear method (i.e. use a single thread to add one by one)? Provide the name of the corresponding law. What is the maximum speed up it can achieve in theory?

____

9. Virtual Memory/ TLB

4 (a) Consider an access pattern to those page tables: 3, 2, 1, 0, 3, 2, 4, 3, 2, 1, 0, 4. How many misses in the TLB will happen if the TLB can hold 3 entries? Which pages are in the TLB in the end? What if theTLB can hold 4 entries? The replacement policy is Least Recently Used (LRU) and the TLB is empty at start.

3 entries: Misses: Entries at end: __________

4 entries: Misses: Entries at end: __________

2 (b) Consider the same access pattern in (a) and the TLB size is 3. If we change the replacement policy to FIFO, how many misses will happen? What if the TLB size is 4?

3 entries: Misses: Entries at end: __________

4 entries: Misses: Entries at end: __________

1 (c) Does the increment in page table's size always lead to reduction in page faults? *This is also called Bélády's anomaly*.

____

10. RISC-V meets $!

Notice: We assume a 32-bit machine by default. No matter which replacement policy, if the block is not full, use FIFO to fill it.

4 (a) Read the given code, answer following questions. (You should show the process of your calculation. Only giving a solution will receive no points.)

```
uint64_t array[LENGTH] = {0};
for (int i = 0; i < REPEAT; i++) {
   for (int j = 0; j < LENGTH; j += STEP_SIZE) {
      array[j] = array[j] + j;
      printf("%ld\n", array[j]);
   }
}
```

If `REPEAT = 4`, `LENGTH = 32`, `STEP_SIZE = 1` and the four-way set-associative cache has 4 sets, the length of *tag* field is 26-bit. The replacement policy is FIFO, write policy is write back. Calculate the miss rate, then answer the dominat type of miss (3Cs).

Read the given RISC-V code. (a0 is the begin address of an integer array with length 1024.)

```
        li  t0, 0
        li  t1, 1024
        li  t2, 2       # 1
exit:   blt t0, t1, loop # 2
        li  a0, 10
        ecall
loop:   rem t3, t0, t2
        beq t3, zero, if
else:   addi t3, t0, -1
        slli t3, t3, 2
        0x00AE0E33      # 3
        lw  t4, 0(t3)
        0x004E2F03      # 4
        add t4, t4, t5
        sw  t4, 4(t3)
        j   continue
if:     slli t3, t0, 2
        add t3, t3, a0
        lw  t4, 0(t3)
        addi t4, t4, 1
        sw  t4, 0(t3)
continue: addi t0, t0, 1
        j   exit
```

In the following questions, if the instruction is a pseudo instruction, you should convert it to its corresponding basic instruction. Every blank can only be filled with **ONE** instruction or hex number.

8 (b) Look at the following **Instruction** cache:

index	tag	block			
0b000	0x0	li t0, 0	li t1, 1024	li t2, 2	blt t0, t1, loop
0b001					
0b010					
0b011					
0b100					
0b101					
0b110					
0b111					

Because instructions are also stored in memory and PC is address, instruction cache calculation can be treated as normal cache. Answer the following questions (appendix: ecall is a 32-bit instruction):

1. Calculate the length of *tag* and *offset* field. (You should show the process of your calculation. Only giving a solution will receive no point)

2. If we run the previous RISC-V code, which sets will be filled? Please list them in their filling order.

3. In every row in the above table fill in the one instruction that is causing the miss. You must fill in the correct place, otherwise you will receive no point even the instruction is correct.

6 (c) Answer the following cache questions (You should show the process of your calculation. Only giving a solution will receive no point):

1. We have a 4-way associative 1-level data cache with LRU replacement policy. It has 64 16-Bytes blocks. Run the previous RISC-V code, calculate the cache hit rate.

2. Now assume a direct-mapped 3-level data cache. L1 cache has the same configuration as the previous sub-question. L1 hit time is 2 cycles. L2 cache has 90% local hit rate and 20 cycles hit time. L3 cache has 99% local hit rate and 400 cycles hit time. Direct access to thememory will take 1000 cycles. Run the previous RISC-V code (use the previous hit rate), calculate the AMAT.

[2] (d) Convert between instructions and hex numbers:

1. li t2, 2: ______________________________

2. 0x004E2F03: ______________________________

11. **OpenMP and Optimization**

[10] (a) Read this piece of code and answer the following questions.

```
#include <omp.h>

void matrixMul (int n, double *A, double *B, double *C) {
   int i, j, k;
   #pragma omp parallel for private(k)
   for (int i = 0; i < n; i++)
      for (int j = 0; j < n; j++)
         for (int k = 0; k < n; k++)
            C[i+j*n] += A[i+k*n] * B[k+j*n];
}
```

1. OpenMP is a parallel computing model based on shared memory. Identify the data sharing attributes of the following variables with 'shared' or 'private'.

   1) `i` ______________

   2) `j` ______________

   3) `k` ______________

2. Reorder the nested loops as well as the OpenMP directive to achieve performance at least no poorer than any of the serial versions. Note that you can only use exactly one line of OpenMP directive.

```
void newMatrixMul (int n, double *A, double *B, double *C) {
   int i, j, k;

   ______________________________

   ______________________________

   ______________________________

   ______________________________
               C[i+j*n] += A[i+k*n] * B[k+j*n];
}
```

2 (b) Time matters!! Use OpenMP directives to parallelize the dot production as fast as you can! P.S. you don't have to fill in both blanks.

```
#include <omp.h>

double dotp(double* x, double* y) {
    double sum = 0.0;

        ______________________________
        for(int i=0; i<ARRAY_SIZE; i++)

            ______________________________
            sum += x[i] * y[i];
    return sum;
}
```

2 12. **Meltdown**

Which of those concepts are essential to understanding how Meltdown works?
Circle the correct answer (T if the concept is essential, F otherwise).

- T / F : Virtual Memory
- T / F : Pipelining
- T / F : Abstraction
- T / F : Speculative Execution
- T / F : Caches
- T / F : Dependability via Redundancy
- T / F : Timing
- T / F : TLB
- T / F : Amdahl's Law
- T / F : Kernel Space

No question here - do not fill.