Computer Architecture I Final Exam

Chinese Name: ______________________________

Pinyin Name: ______________________________

Student ID: ______________________________

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

Question	Points	Score
1	1	
2	30	
3	10	
4	11	
5	8	
6	14	
7	22	
8	4	
Total:	100	

- This test contains 20 numbered pages, including the cover page, printed on both sides of the sheet.
- We will use blackboard 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 32-bit 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: Fill in you name** (1 point)
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 20 times).

2. **Misc.**

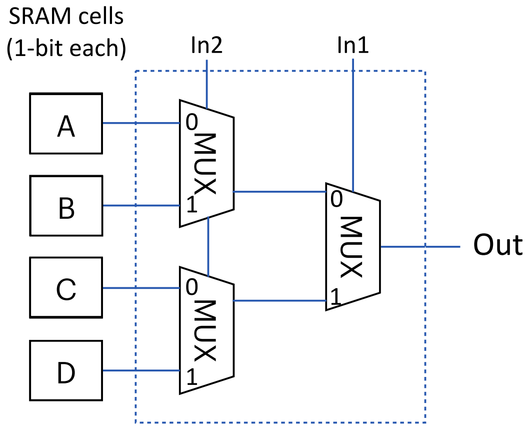
2 (a) (Single Choice) The following C code prints ______________. Note that `float` and `int` are 32 bits in size, and `float` follows the IEEE 754 standard.

```c
#include <stdio.h>
int main() {
    float a = -2.5;
    printf("%x\n", *(int*)(&a));
    return 0;
}
```

A. `c0200000`
B. `c0400000`
C. `c0a00000`
D. `c0c00000`

Solution: A
difficulty: medium
$2.5 = 2^1 \times (1 + 2^{-2})$
sign bit: 1
exponent bits: 128=10000000
mantissa bits: 010000......

2 (b) In FPGA, look-up table (LUT) is a basic reconfigurable element. Below shows a 2-input LUT structure in an FPGA. What should the SRAM cells store to implement a 2-input XOR gate?
A. ________ B. ________ C. ________ D. ________



XOR truth table

In1	In2	Out
0	0	0
0	1	1
1	0	1
1	1	0

Solution: A. 0; B. 1; C . 1; D. 0.

2 (c) Which statement(s) is (are) correct? ___________. Note that there may be more than one correct statement.

A. RAID 1 guarantees the same level of fault tolerance as RAID 0.

B. RAID 4 and RAID 5 use parity calculation to provide data redundancy and fault tolerance.

C. RAID 4 and RAID 5 have the same data striping characteristics.

D. RAID 3 can sustain the failure of multiple drives simultaneously without data loss.

E. None of the above is correct.

Solution:

Answer
B

2 (d) Read the following C code

```c
#define PI 3.14
const float pi = 3.14;
```

Which of the following statements is/are correct? ____________

A. PI and pi have the same type.

B. PI is assignable while pi is not.

C. Code runs approximately at the same speed with these two.

D. Variable pi takes more space than macro PI.

E. They behave the same in all situations.

Solution: C

2 (e) Assume that disks are quoted to have a 1,200,000-hour MTTF. Assume that a warehouse-scale computer (WSC) is composed of 50,000 servers, with two disks installed per server. Please answer the following questions accordingly.

(1 point) What does MTTF mean? _____________________________

(1 point) For this WSC, about ______ disks would expect to fail per year.

Solution:

Mean time to failure

730

2 (f) Suppose we have a processor with a base CPI of 1.0 if all references hit in a single-level cache and the processor's clock rate is 2GHz. Assume a main memory access time of 100 nanoseconds, including all the miss handling. Suppose the miss rate per instruction at the

single-level cache is 2%. The CPI consequently increases with the average memory stall cycle(s) per instruction. If we add an inclusive secondary cache that has 5 nanoseconds access time for either a hit or a miss and is large enough to reduce the miss rate to main memory to 0.4%, the new CPI would be _______.

Solution: 2

Calculation: $1 + 2\% \times (5 \times 2) + 0.4\% \times (100 \times 2) = 2.0$

18 (g) **(True or False)** Please check if the following statements are true (T) or false (F).

1) A heterogeneous computing system can contain different CPUs, GPUs and FPGAs. ___________

Solution: T

2) In a pipeline, any instruction is active in exactly one stage of the pipeline at a time. ___________

Solution: T

3) The address of variables is fixed by assemblers. ___________

Solution: F

4) Absolute function address, external function reference and static data reference will be relocated in the linker. ___________

Solution: T

5) Handling a page fault with NAND flash memory-based solid state drive (SSD) demands the SSD controller to allocate a free (clean) flash page for out-of-place update. ___________

Solution: F

6) Multiple exceptions may occur simultaneously in a single clock cycle for a program and the processor needs to firstly handle the one occurring at a pipeline stage that is the most distant to the write-back stage, so that it can quickly resume the execution for the program after handling exceptions. ___________

Solution: F

7) Given a program that is vulnerable to some side-channel attack due to cache hits/misses, programmers can mitigate the attack by running the program on a write-through multi-level cache hierarchy. ___________

Solution: F

8) Virtual memory management enables the operating system to load a program anywhere in the physical memory space. ___________

Solution: T

9) There is no need to maintain cache coherence for a single-core processor. ___________

Solution: T

10) For the load reserved/store conditional pair of instructions, if the contents of the memory location specified by the load-reserved are changed before the store-conditional to the same address occurs, the store-conditional fails and does not write the value to memory. ___________

Solution: T

11) In a non-inclusive multi-level cache hierarchy, a miss in the L1 instruction cache will not replace a cache block in the L1 data cache. ___________

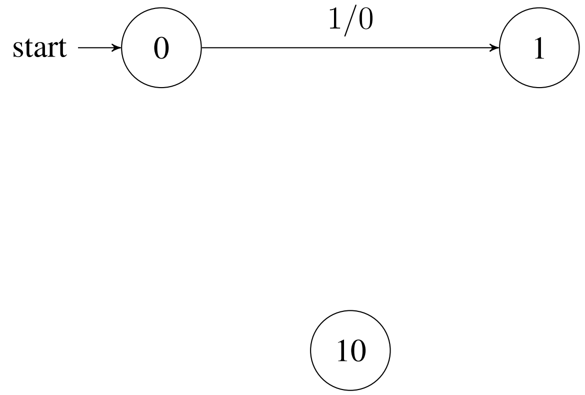
Solution: T

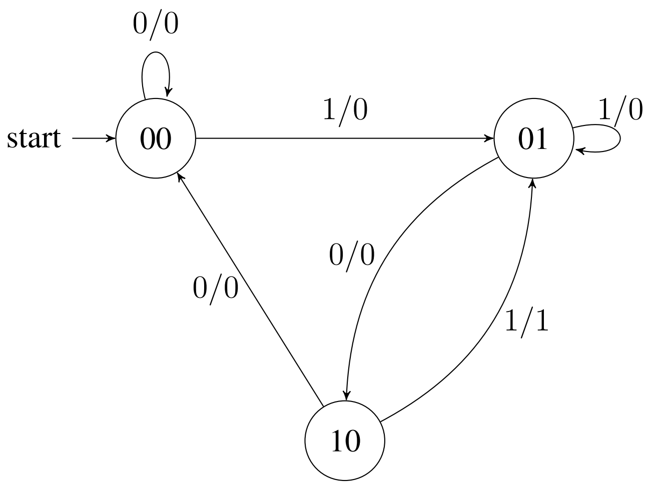
12) In the MapReduce programming model, the Map step splits input data into small pieces for parallel processing and then the Reduce step combines and processes intermediate results to obtain final result. The programmer handles all system faults during both steps. ___________

Solution: F

3. **FSM**

4 (a) Build a **Mealy** FSM model to detect ’101’ **with** overlapping, i.e., the tail 1 of ’101’ can be considered as the head 1 for the next detection. Complete the Mealy FSM state transition diagram by adding lines and necessary transition and output information. “0” state indicates a single 0 is detected; “1” state indicates a single 1 is detected; “10” state indicates a sequence of “10” is detected in the diagram below. (4 points)



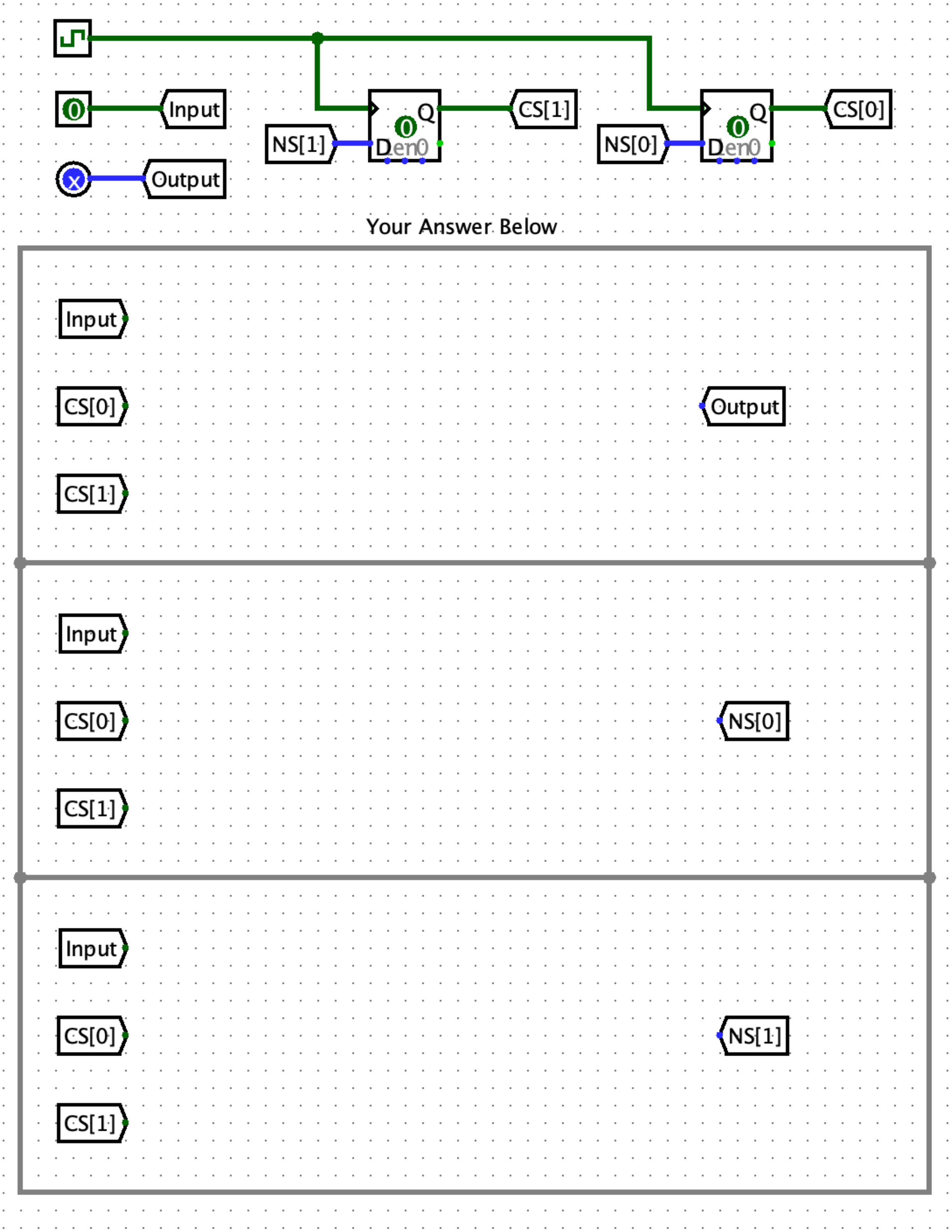


Solution:

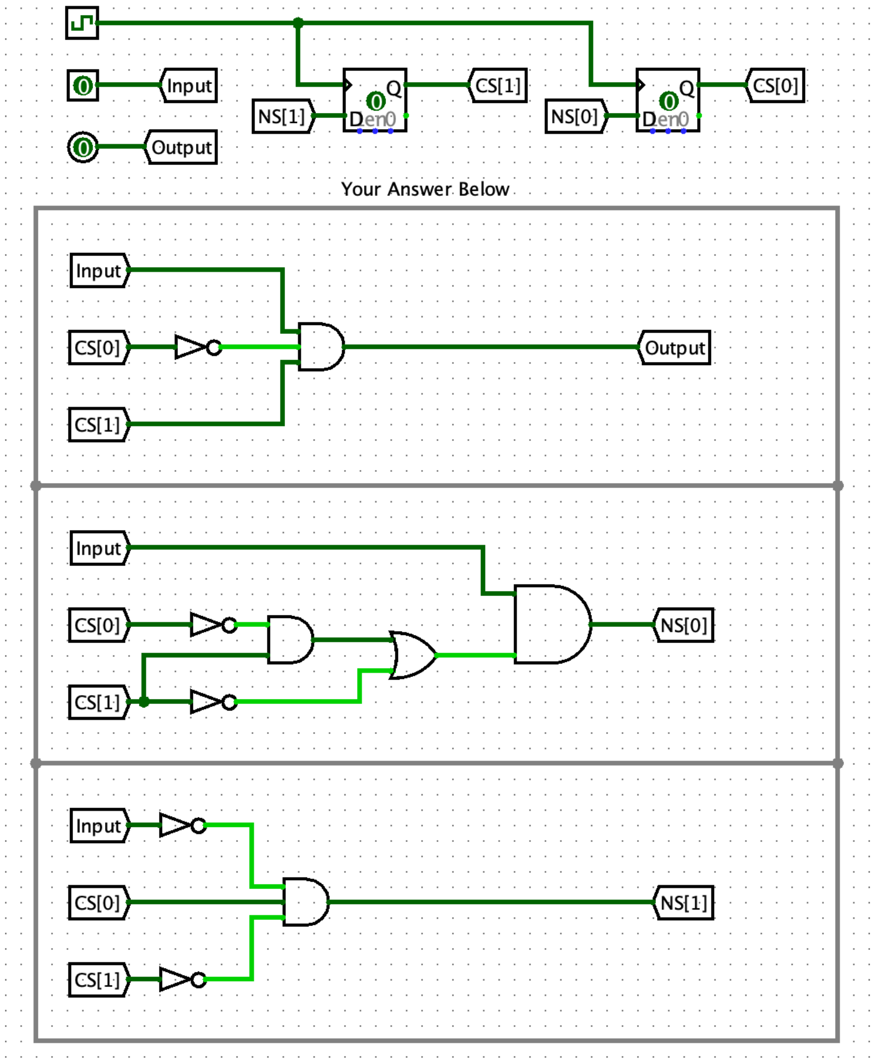
3 (b) Based on the Mealy FSM you drew. Write down the truth table for the next-state and output logic. Use 'CS' to represent current state and 'NS' for next state. We use "00" to represent state "0", "01" to represent state "1" and "10" to represent state "10". (3 points)

CS[1]	CS[0]	input	NS[1]	NS[0]	output
0	0	0	**0**	**0**	**0**
0	0	1	**0**	**1**	**0**
0	1	0	**1**	**0**	**0**
0	1	1	**0**	**1**	**0**
1	0	0	**0**	**0**	**0**
1	0	1	**0**	**1**	**1**

3 (c) Complete the following Logisim circuit using the truth table you wrote. (3 points)



Solution:



Any other reasonable solutions are accepted.

4. **Pipeline**

Consider the five-stage pipelined RISC-V processor we covered in the lectures with the following stages: Instruction fetch (IF), instruction decode and register read (ID), execute (EX), memory access (MEM), and write back (WB). Assume that the processor has no forwarding or hazard detection mechanisms implemented. The table below shows the delay for each stage.

Stage	IF	ID	EX	MEM	WB
Delay	200 ps	400 ps	100 ps	250 ps	200 ps

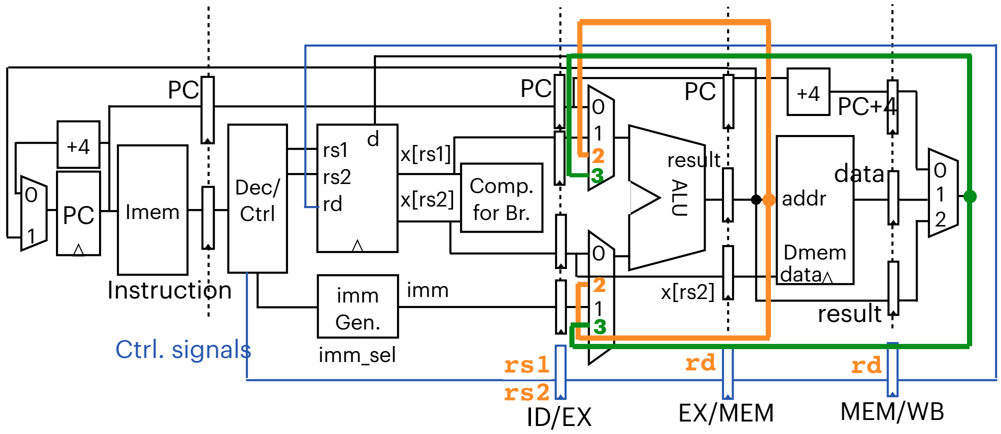
1 (a) (1 point) Calculate the maximum frequency of the processor. _________

Solution: 1/(400 ps) = 2.5 GHz

2 (b) (2 points) If the ID stage can be further pipelined into two stages as instruction decode (DEC) and register read (REG), with equal delays of 200 ps each. What is the maximum frequency of this processor now? _________
What is the latency or total time this processor fully process one load instruction from IF to WB stage? _________

Solution: 1/(250 ps) = 4 GHz. 250 ps × 6 = 1500 ps = 1.5 ns.

3 (c) (3 points) Forwarding datapaths are added to the five-stage pipelined RISC-V processor, circuit diagram as shown below.



The given code is executed on the processor:

```
1 ADD R1, R2, R3
2 SUB R4, R1, R5
3 AND R6, R1, R7
4 LW  R3, 0 (R6)
5 ADD R1, R2, R3
6 ADD R9, R10, R11
```

Please identify for which instruction(s) the forwarding datapaths are activated to reduce stall(s)? ______________ (use the **line number(s)** of the instruction(s))

Solution: 2, 3, 4, 5

5 (d) (5 points) With the forwarding datapaths, there is still one unavoidable stall left to correctly execute the code above. Please locate the instructions and registers that cause the stall and solve it by code scheduling (**use the blank below**).

The register _______ in instructions _______ and _______ (use the **line numbers** of the instructions) causes the unavoidable stall.
Write down the code without stalls after code scheduling.

1 ____________________
2 ____________________
3 ____________________
4 ____________________
5 ____________________
6 ____________________

Solution: The register `R3` in instructions 4 and 5 (use the line numbers of the instructions) causes the unavoidable stall.

```
1 ADD R1, R2, R3
2 SUB R4, R1, R5
3 AND R6, R1, R7
4 LW  R3, 0 (R6)
5 ADD R9, R10, R11
6 ADD R1, R2, R3
```

5. **Number Representation**

6 (a) (6 points) Consider two 8-bit hexadecimal bit patterns `0x7C` and `0xC0`, what are their values in decimal if they are unsigned integers?

`0x7C`____________ `0xC0`____________

What if in 2's complement representation?

`0x7C`____________ `0xC0`____________

Assume they are unsigned integers, select below what would happen if we perform 8-bit addition of them?_________

What if they are 2's complement numbers?________

A. Underflow.
B. Overflow.
C. Neither underflow nor overflow.
D. Both underflow and overflow.

Solution: 124; 192; 124; -64; B; C

2 (b) (2 points) Which of the following bit pattern(s) represent a 0 in IEEE-754 standard? (Hint: ±0 are both 0s.)________ Which of the following bit pattern(s) represent a NaN in IEEE-754 standard? ________

A. `0x00000000` B. `0x10000000`
C. `0x80000000` D. `0xff800000`
E. `0xff000000` F. `0xffabcdef`
G. `0x7ffedcba` H. `0xff800001`

Solution: AC; FGH

6. RISC-V

$\boxed{2}$ (a) (2 points) Multiply-and-accumulate (MAC), i.e., $c = c + a \times b$, is a basic operation in convolutional neural network (CNN) computations, where c is the value to be accumulated. To support efficient neural network computation, we extend the RV32I by adding a MAC instruction. First, we add **an extra special register** to store the value to be accumulated, **`Xmac`**. We assume `Xmac` register is large enough so that no overflow would occur. Then, we define that the extended instruction `mac rs1, rs2` performs

`Xmac = Xmac + X[rs1] * X[rs2],`

where `X[rs1]` and `X[rs2]` are ordinary general-purpose registers. Now we need to decide the format of this instruction by specifying its type. Which of the following could be the suitable type(s) for this instruction to be compatible with RV32I? __________.

A. R-type B. I-type C. U-type D. S-type

Solution: AD for full mark. A or D for 1 point.

$\boxed{5}$ (b) (5 points) A convolution can be implemented by a dot-product, which can be described as

$$c = \sum_{i=1}^{N} a_i b_i. \tag{1}$$

To perform the convolution, we add another instruction to RV32I, `setmac rs1`, to initialize the value of register `Xmac` and it performs `Xmac = X[rs1]`. Assume we have two 32-bit integer arrays `a` and `b`. Each of them has 9 elements. Please complete the following assembly code to accomplish the dot-product of these two arrays and **keep the result in `Xmac` register**. You may use more or less blanks than ones that are provided below.

```
        li t1 9            # set the loop number
        li a0 0x80000000   # the starting address of array a
        li a1 0x80000024   # the starting address of array b
        setmac x0          # set the initial value of Xmac to 0
loop:
        lw t2, 0(a0)        # load one MAC operand to register t2
        lw t3, 0(a1)        # load another to register t3
       _______________
       _______________
       _______________
       _______________
       _______________
       _______________
       _______________
       _______________
       _______________
```

Solution:

```
       li t1 9           # set the loop number
       li a0 0x80000000  # the starting address of array a
       li a1 0x80000024  # the starting address of array b
       setmac x0         # set the initial value of Xmac to 0
loop:
       lw t2, 0(a0)
       lw t3, 0(a1)
       mac t2, t3
       addi t1, -1
       addi a0, 4
       addi a1, 4
       bne t1, x0, loop
```

7

(c) (7 points) Assume Xmac is a 32-bit **callee-saved** register. Besides setmac and mac, assume we can perform logic and arithmetic, load and store instructions using Xmac register just as a regular general-purpose register. E.g. we can copy the value of Xmac to a0 by add a0, x0, Xmac; we can use lw Xmac, some address and sw Xmac, some address to interact with the memory. Also, we ignore the non-ideal effects such as overflow. Define a C function

```c
int mul(int a, int b){
        return a*b;
}
```

Translate the C function into RV32I assembly code along with the extended mac and setmac instructions. You may use more or less blanks than ones that are provided below. Please complete the assembly code below following the calling conventions we covered in the lectures. a and b are preloaded into registers a0 and a1, and the returned value should be put in register a0.

```
mul:
        ____________________
        ____________________
        ____________________
        ____________________
        ____________________
        ____________________
        ____________________
        ____________________
        ____________________
        jalr x0, 0(ra)          # back to the caller
```

Solution:

```
mul:
        addi sp, sp, -4
        sw Xmac, 0(sp)
        setmac x0 (addi Xmac, x0, x0/li Xmac, 0 is also fine)
        mac a0, a1
        add a0, x0, Xmac/addi a0, 0, Xmac
        lw Xmac, 0(sp)
        addi sp, sp, 4
        jalr x0, 0(ra)         # back to the caller
```

7. **Virtual Memory and Cache**

Table 1: A portion of memory used for page tables

	Address	**Contents (physical address)**
Memory	0x00	0x020
	0x08	0x040
	0x10	
	0x18	
	0x20	0x070
	0x28	0x090
	0x30	
	0x38	
	0x40	0x080
	0x48	
	0x50	
	0x58	
	0x60	
	0x68	
	0x70	0x110
	0x78	
	0x80	0x150
	0x88	0x170
	0x90	
	0x98	0x130

Table 2: Hit/miss and physical addresses for a sequence of virtual addresses

Virtual Address (8-bit)	TLB hit/page table hit/page fault	Physical Address (12-bit)
0x48	Page table hit	0x158
0x18		
0x50		
0x28		
0x58		

8

(a) (8 points) Table 1 shows the contents of a portion of physical memory used for page tables. Assume the system uses 64-bit words, 16-byte pages, three-level page tables, and a fully associative two-entry TLB with first-in-first-out (FIFO) eviction. Each stage of the page table uses a single bit index. At the beginning, the TLB is empty and the list of free pages contains page numbers 0x0A, 0x03, 0x18, 0x12, 0x19 in order from first-to-be-allocated to last-to-be-allocated.

For the virtual memory address trace shown by the leftmost column of Table 2, fill in Table 2 whether the access results in a TLB hit, a page table hit, or a page fault, and give the translated physical address. The first row of Table 2 has been filled as a hint for you.

Table 3: The TLB entries

	Entry 0	Entry 1
VPN		
PPN		

Fill out the memory table (Table 1) and the TLB (Table 3) with final states. Assume that the page table base register is set to 0 and TLB fills in from left to right. The entries in the page table are the full 12-bit physical addresses of the start of the page, not just the PPN. Note that you shall write down all numbers in the hexadecimal form.

Solution:

	Address	**Contents (physical address)**
Memory	0x00	0x020
	0x08	0x040
	0x10	
	0x18	
	0x20	0x070
	0x28	0x090
	0x30	
	0x38	
	0x40	0x080
	0x48	
	0x50	
	0x58	
	0x60	
	0x68	
	0x70	0x110
	0x78	**0x0A0**
	0x80	0x150
	0x88	0x170
	0x90	**0x030**
	0x98	0x130

Virtual Address (8-bit)	TLB hit/page table hit/page fault	Physical Address (12-bit)
0x48	Page table hit	0x158
0x18	**Page fault**	**0x0A8**
0x50	**Page table hit**	**0x170**
0x28	**page fault**	**0x038**
0x58	**TLB hit**	**0x178**

	Entry 0	Entry 1
VPN	**0x05**	**0x02**
PPN	**0x17**	**0x03**

9 (b) (9 points) After you obtain the trace of accessed physical addresses, your close friend Li Hua asks you to run the trace with her/his CPU cache managed in the way of PIPT. The cache is two-way set-associative. In all there are four sets, in which each cache line has 16 bytes. The replacement policy is FIFO. Assume that the cache is empty at start. Please fill Table 4 with the trace of physical addresses you have got. The first row of Table 4 has been filled as a hint. Note that you shall write down all numbers in the hexadecimal form.

What does PIPT mean? ____

Table 4: Cache access traces

Virtual Address	Cache			
	Tag	Set Index	Way Number	Hit or Miss
0x48	0x05	0x01	0x00	Miss
0x18				
0x50				
0x28				
0x58				

Solution: PIPT: Physically indexed, virtually tagged

Virtual Address	Cache			
	Tag	Set Index	Way Number	Hit or Miss
0x48	0x05	0x01	0x00	Miss
0x18	**0x02**	**0x02**	**0x00**	**Miss**
0x50	**0x05**	**0x03**	**0x00**	**Miss**
0x28	**0x00**	**0x03**	**0x01**	**Miss**
0x58	**0x05**	**0x03**	**0x00**	**Hit**

2 (c) (2 points) Li Hua decides to apply Hamming ECC (Error Correction Code) we have learned in class to the lower 8 bits of each physical address you just fill in Table 2. Please write down the code words in Table 5. Note that you shall write down all numbers in the hexadecimal form and do not add an extra parity bit at the MSB.

Table 5: Code word for each physical address (lower 8-bit only)

Physical address	0x158				
Code word					

Solution:

Physical address	0x158	**0x0A8**	**0x170**	**0x038**	**0x178**
Code word	**0xCB8**	**0x358**	**0x1E0**	**0x078**	**0x9F8**

3 (d) (3 points) What is the hit rate of running your trace with Li Hua's cache? Assume that a hit in Li Hua's cache takes 5 cycles while retrieving data from main memory takes 45 cycles. Calculate the overall access latency. In addition, assuming that you have another trace with 1,000 accesses but 1/10 of your current hit rate, calculate the overall access latency.

Solution: Hit rate: $\frac{1}{5} = 20\%$.
First overall latency: 5 + (45+5)*4 = 205 cycles.
Second overall latency: 1000 * 2% * 5 + 1000 * (1-2%) * (5 + 45) = 49100.

4 8. **Multithreading**

We have received a task of counting the number of positive and negative numbers in an array A that only holds integers other than zero. Using a single thread is two slow, so let us parallel it with the following code.

```
#include <stdio.h>
#include <omp.h>
void count_num (int *A, int size, int threads) {
      int result[2] = {0, 0};
      int i = 0;

      omp_set_num_threads(threads);

      #pragma omp parallel for
      for (i = 0; i < size; ++i)
            if (A[i] > 0) result[0] += 1;
            else result[1] += 1;

      printf("Positive: %d\n", result[0]);
      printf("Negative: %d\n", result[1]);
}
```

(2 point) As we increase the number of threads running this code, will it print the correct values for positive and negative integers? Explain your answer.

Solution: No. There may be a data race.

(1 point) Can there be false sharing if the cache block size is 8 bytes?

Solution: Yes.

(1 point) Can there be false sharing if we increase the cache block size to be 64 bytes?

Solution: Yes.