0917432 Computer Architecture and Organization 2 (Fall 2023)

Midterm Exam

الاسم: **KEY** رقم التسجيل: رقم التسلسل:

==========

Instructions: Time 75 minutes. Open book and notes exam. No electronics. Please answer all problems in the space provided and limit your answer to the space provided. There are six problems.

<Good Luck>

P1. An IC manufacturing foundry uses 300-mm wafers to produce 31.4-mm^2 chips. Assuming that, on average, 1,000 chips fail the wafer test, and 125 chips fail the part test. Give an estimation of the manufacturing **yield** of this chip.

<5 marks>

Solution:

Chips/wafer ≈ wafer area / chip area

$= \pi \times r^2 / 31.4 \approx 3.14 \times (300/2)^2 / 31.4 = 150^2 / 10$

$= 2{,}250$

Yield = pass chips/wafer / total chips/wafer

$= (2{,}250 - 1{,}000 - 125) / 2{,}250$

$= 1{,}125 / 2{,}250$

$= 50\%$

P2. A single-cycle implementation of RISC-V ISA runs on 1-GHz processor clock. Assume that the stage times of a 5-stage pipeline implementation of this ISA are in the table below. What is the expected **peak speedup** of the pipeline implementation relative to the single-cycle implementation?

	F	D	E	M	W
Stage time	200 ps	100 ps	250 ps	200 ps	100 ps

<4 marks>

Solution:

Time between instructions $_{\text{unpipelined}}$ = 1 / f = 1 / 1 GHz = 1 ns

Time between instructions $_{\text{pipelined}}$ = time of longest stage = 250 ps

Peak speedup = Time between instructions $_{\text{unpipelined}}$ / Time between instructions $_{\text{pipelined}}$

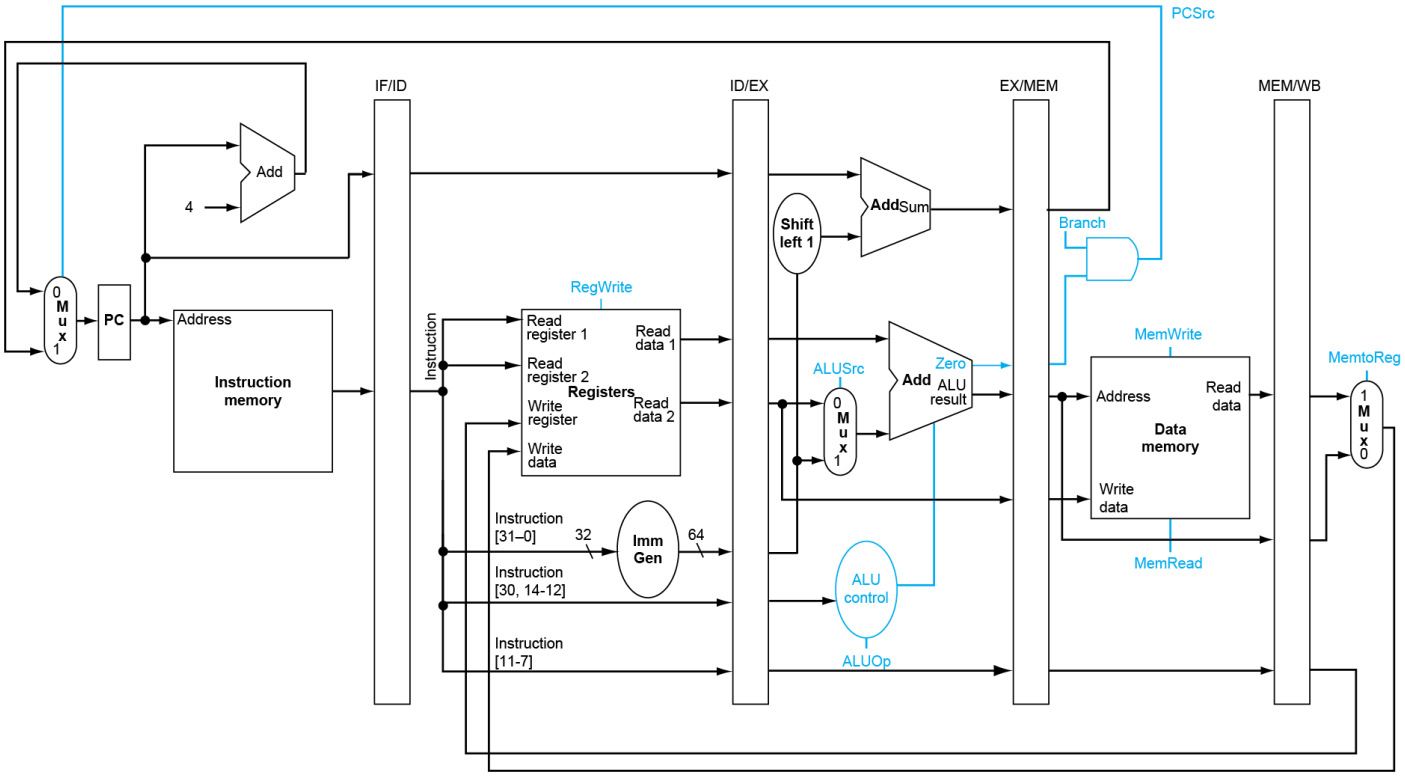
= 1 ns / 250 ps = 1000 ps / 250 ps

= 4

P3. Assume that the following five instructions are executed by the RISC-V pipeline shown below. Assume that this pipeline has the needed forwarding paths to solve data hazards, and assume that it uses the static branch prediction: Predict Not Taken. In the table below, specify the values of the shown six **fields/signals** when the first instruction has reached the Write-back stage. Note: All numbers shown are in decimal.

<6 marks>

```
Address   Instruction
100000    ld   x10, 40(x1)
100004    subi x11, x11, 16
100008    bne  x12, x13, 20
100012    add  x14, x3, x4
100016    ld   x15, 48(x1)
```



Field/Signal	Value
The output of the adder of the IF stage	**100020**
IF/ID.RegisterRs2	**4**
ID/EX.RegisterRs1	**12**
Lower input of the upper adder of the EX stage	**40**
EX/MEM.MemRead	**0**
MEM/WB.RegisterRd	**10**

P4. Assume that the 5-stage pipelined processor studied in the class solves data hazards through stalls and some forwarding; it only has the forwarding paths from the MEM stage to the EX stage and through the Register File where results written can be read in the same cycle. Use the **multi-cycle pipeline diagram** below to show how this processor executes the instruction sequence shown and indicate any forwarding using **arrow** between the involved pipeline stages.

<4 marks>

Instruction	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
`ld x2,0(x1)`	F	D	E	M	W										
`add x4,x2,x3`		F	D	D	D	E	M	W							
`sub x5,x4,x6`			F	F	F	D	E	M	W						
`sd x6,0(x5)`						F	D	E	M	W					

P5. Assume that you have a processor that supports SIMD operations on 512-bit registers. How many load instructions does this processor need to load the contents of 1,024-element vector. Assume that this vector holds single-precision floating-point numbers (32 bits each).

<5 marks>

Solution:

Elements/SIMD operation = 512 bits / 32 bits = 16 elements

Number of loads = 1,024 / 16

= 64 load operations

P6. Unroll the following loop **three times** and use the table below to **schedule** the unrolled loop efficiently for the static dual-issue processor described in the class. Remember that this processor has one pipeline for ALU and branch instructions and another for the memory instructions. Assume that this processor resolves branches in the Decode stage and solves data hazards through all necessary forwarding paths. Note: This loop finds the sum of a 300-element vector.

<6 marks>

```
# Assume x1 is initialized to 0
#        x2 is initialized to the starting addresses of a vector
#        x10 has the end-of-loop test value
loop: add     x3, x2, x1
      ld      x4, 0(x3)
      add     x9, x9, x4
      addi    x1, x1, 8
      blt     x1, x10, loop
```

	ALU/branch	Load/store	Cycle
loop:	add x3, x2, x1	nop	1
	nop	ld x4, 0(x3)	2
	addi x1, x1, 24	ld x5, 8(x3)	3
	add x9, x9, x4	ld x6, 16(x3)	4
	add x9, x9, x5	nop	5
	add x9, x9, x6	nop	6
	blt x1, x10, loop	nop	7
			8
			9