Question #1. Instruction Design (25 points)

a) (16 points) Consider the sequence of four instructions shown below:

```
Loop :   lw     $6, 0($5)
         Add    $7, $7, $6
         Addi   $5, $5, -2
         Bne    $5, $7, loop
```

let lw = 2

let add function = 1

16

let bne = 3

In the space provided below, show the binary representation of each of the four instructions in this sequence. Clearly mark the instruction fields and show their decimal equivalent.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	0	0	1	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

5

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	1	1	1	0	0	1	1	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1

5

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	1	0	1	1	1	1	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1

3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	0	0	1	1	1	0	0	1	0	1																

3

b) (9 points) Consider the following information that helps you in designing an ISA:

- A 9-bit address bus, and a 9-bit data bus
- A register file with 8 registers
- All registers in the processor are 9-bits
- A two address/operand instruction format
- Eight different Opcodes

?

Draw the instruction formats (ISA) for register-register instructions, register-memory addressing instructions, and specify the size in bits for the opcode, register numbers and memory addresses.

b) (9 points) Consider the following information that helps you in designing an ISA:

- A 9-bit address bus, and a 9-bit data bus
- A register file with 8 registers
- All registers in the processor are 9-bits
- A two address/operand instruction format
- Eight different Opcodes

?

Draw the instruction formats (ISA) for register-register instructions, register-memory addressing instructions, and specify the size in bits for the opcode, register numbers and memory addresses.

Question #3. Pipelining and Performance (20 points)

MIPS (Microprocessor without Interlocked Pipeline Stages) is an instruction set architecture that is used in many applications and generalizes to many other architectures. MIPS instructions are classified into three formats the R-Format, I-Format, and the J-Format. These instruction executes in the well-known instruction fetch-decode-execute (F-D-E) life cycle.

a) (4 points) What stages of the F-D-E life cycle does each of the R, I and J format execute?

R-Format:

I-Format:

J-Fromat:

It is well-known that every stage of the F-D-E life cycle takes different execution time. Assume it takes 200 ps to access memory (IF & MEM stage), 50 ps to access Registers (instruction decode and write back stages), and 100 ps to perform ALU operations (Execute stage).

Now consider a program which is composed of an instruction mix of 70% from R-Format, 20% from I-Format and 10% from J-Format.

b) (4 points) For a **single cycle** implementation, an instruction executes in one clock cycle:

1. What is the clock cycle time?

2. How long does it take to execute 50 instructions?

(If needed, use the back of this page for solution)

c) (4 points) For a **multi-cycle implementation**, a stage of the F-D-E life cycle executes in one clock cycle:

1. What is the clock cycle time?

2. How long does it take to execute 50 instructions?

d) (4 points) For a **five-stage fully pipelined** implementation, assuming there are no hazards,

1. What is the clock cycle time?

2. How long does it take to execute 50 instructions? Be sure to count the exact number of clock cycles.

e) (4 points) Suppose we now have **6 stage-pipelining** by splitting a MEM stage into two parts: MEM1 and MEM2, and each takes 100 picoseconds. Then assuming there are no hazards and that we use the minimum clock cycle time. What would be the speedup of the new implementation over the original 5 stage pipelining?

(If needed, use the back of this page for solution)

Question #4. **MIPS Language (15 points)**

Execute the following MIPS code fragments, showing the changes that occur in the register file and in memory. The contents of the register file and the memory are shown in hexadecimals. You only need to show the changes.

```
        addi $1, $0, 1Bh
Loop:   lw   $3, 8($2)
        add  $4, $4, $3
        addi $2, $2, 2
        subi $1, $1, 1
        bne  $1, $2, loop
        sw   $4, 4($4)
```

Is the branch taken? (Circle one)

YES [✓ NO]

BEFORE

REGISTERS		MEMORY	
$0	0h	20	Ch
$1	14h	24	20h
$2	18h	28	30h
$3	Ch	2C	40h
$4	1Ch	30	50h

AFTER

REGISTERS		MEMORY	
$0		20	
$1		24	
$2		28	
$3		2C	
$4		30	

(If needed, use the back of this page for solution)

Question #5. **Single-cycle datapath (15 points)**

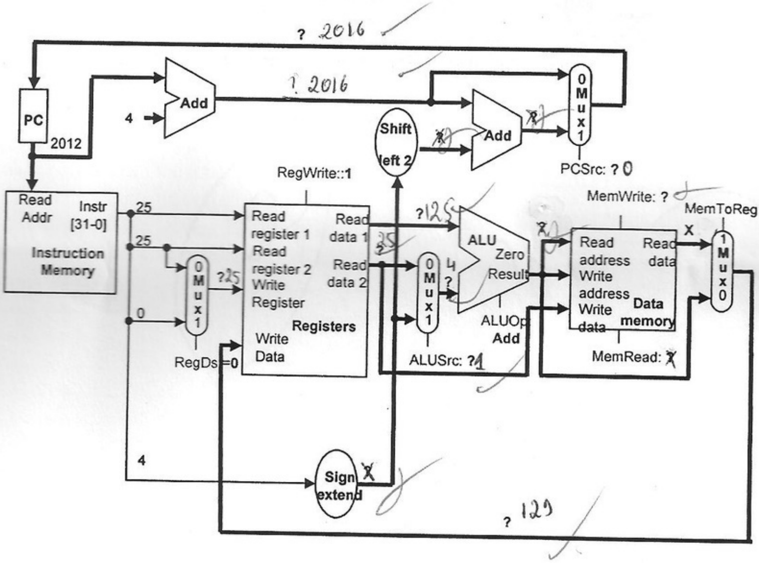
Let's say we want to execute the following immediate addition instruction in the single-cycle datapath:

addi $25, $25, 4

The single-cycle datapath diagram below shows the execution of this instruction. Several of the datapath values (in decimals) are filled in already. You are to provide values for the twelve remaining signals in the diagram, which are marked with a ? symbol. (2 points each)

You should:

- Write your answers directly on the diagram, but write clearly.
- Show decimal values.
- Assume register $25 initially contains the number 125.
- If a value cannot be determined, mark it as 'X.'



(If needed, use the back of this page for solution)

Question #6. Pipelining (10 points)

Assume the following latencies for each stage in our single-cycle processor.

IF	ID	ALU	MEM	WB
200ps	100ps	200ps	200ps	100ps

a. If the time for an ALU operation can be shortened by 25% will it affect the speedup obtained from pipelining? If yes, by how much? If no, why not? (5 points)

b. What if the ALU operation now takes 25% more time? (5 points)

Performance

1. Formula for computing the CPU time of a program P running on a machine X:

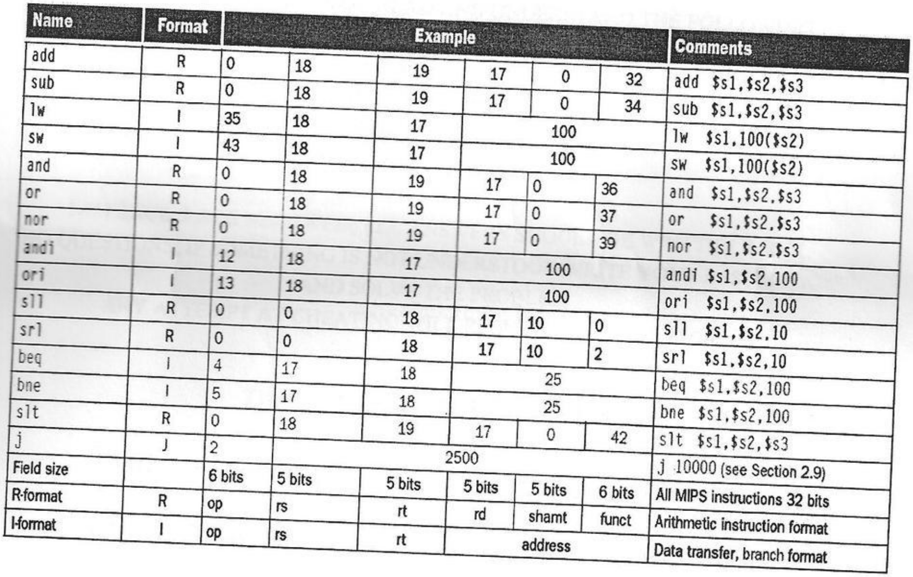
$$CPU\ time_{XP} = Number\ of\ instructions\ executed_P \times CPI_{XP} \times Clock\ cycle\ time_X$$

2. CPI is the average number of clock cycles per instruction:

$$CPI = Number\ of\ cycles\ needed\ /\ Number\ of\ instructions\ executed$$

3. Speedup is a metric for relative performance of 2 executions:

$$\begin{aligned} Speedup &= Performance\ after\ improvement\ /\ Performance\ before\ improvement \\ &= Execution\ time\ before\ improvement\ /\ Execution\ time\ after\ improvement \end{aligned}$$



Name	Format	Example						Comments
add	R	0	18	19	17	0	32	add $s1,$s2,$s3
sub	R	0	18	19	17	0	34	sub $s1,$s2,$s3
lw	I	35	18	17	100			lw $s1,100($s2)
sw	I	43	18	17	100			sw $s1,100($s2)
and	R	0	18	19	17	0	36	and $s1,$s2,$s3
or	R	0	18	19	17	0	37	or $s1,$s2,$s3
nor	R	0	18	19	17	0	39	nor $s1,$s2,$s3
andi	I	12	18	17	100			andi $s1,$s2,100
ori	I	13	18	17	100			ori $s1,$s2,100
sll	R	0	0	18	17	10	0	sll $s1,$s2,10
srl	R	0	0	18	17	10	2	srl $s1,$s2,10
beq	I	4	17	18	25			beq $s1,$s2,100
bne	I	5	17	18	25			bne $s1,$s2,100
slt	R	0	18	19	17	0	42	slt $s1,$s2,$s3
j	J	2	2500					j 10000 (see Section 2.9)
Field size		6 bits	5 bits	5 bits	5 bits	5 bits	6 bits	All MIPS instructions 32 bits
R-format	R	op	rs	rt	rd	shamt	funct	Arithmetic instruction format
I-format	I	op	rs	rt	address			Data transfer, branch format

(If needed, use the back of this page for solution)