1. The following RISC-V program is to be run on a RISC-V pipeline processor

(i) identify all data dependencies beside each instruction

```
1. add x7, x6, x8     No dependencies b/c no previous instructions
2. slt x9, x10, x9    No dependencies with respect to 1.
3. beq x9, x0, next   RAW on x9 re: 2.
4. lw x6, 80(x10)     WAR on x6 re: 1.; RAR on x10 re: 2.
```

(ii) Work out and diagram the optimal pipeline schedule using forwarding from EX or MEM stages to any other stage, then compute the pipeline CPI

	cc1	cc2	cc3	cc4	cc5	cc6	cc7	cc8	cc9	cc10
add x7, x6, x8	IF	ID	EX	MEM	WB					
slt x9, x10, x9		IF	ID	EX	MEM	WB				
beq x9, x0, next			IF	ID	EX	MEM	WB			
lw x6, 80(x10)				IF	ID	EX	MEM	WB		

branch not taken

CPI = 8 cycles/4 instructions = 2 (assume branch not taken)

2. Assume that you have a computer with 1 clock cycle per instruction (CPI=1) when all accesses to memory are in cache. The only accesses to data come from load and store instructions. Those accesses account for 25 % of the total number of instructions. Miss penalty is 50 clock cycles and miss rate is 5 %. Determine the speedup obtained when there is no cache miss compared to the case when there are cache misses

Given: **cp**: Processor cycles. **cw**: Wait cycles. **T**: Clock period. **IC**: Number of instructions. **CPI**: Cycles per instruction. $\mathbf{a_i}$: Amount of accesses produced by instructions. $\mathbf{a_d}$: Amount of accesses produced by data. **mr**: Miss rate. **mp**: Miss penalty.

In case of no misses:

$$Tcpu = (cp + cw) \cdot T = (IC \cdot CPI + 0) \cdot T = IC \cdot 1 \cdot T = IC \cdot T$$

Including misses:

$$cw = IC \cdot (a_i + a_d) \cdot mr \cdot mp$$

$$= IC(1 + 1 \cdot 0.25) \cdot 0.05 \cdot 50 = IC \cdot 1.25 \cdot 0.05 \cdot 50 = 3.125$$

So,

$$Tcpu = (IC \cdot 1 + IC \cdot 3.125) \cdot T = 4.125 \cdot IC \cdot T$$

$$Speedup = S = 4.125 \cdot IC \cdot T / [IC \cdot T] = 4.125$$

3. When Program X is run, the user CPU time is 3 seconds, the total wall clock time is 4 seconds, and the system performance is 10 MFLOP/sec. Assume that there are no other processes taking any significant amount of time, and the computer is either doing calculations in the CPU, or doing I/O, but it can't do both at the same time. We now replace the processor with one that runs six times faster, but doesn't affect the I/O speed. What will the user CPU time, the wall clock time, and the MFLOP/sec performance be now?

$\text{CPU performance}_B / \text{CPU performance}_A = \text{CPU time}_A / \text{CPU time}_B$

$6 = 3/\text{CPU time}_B$

User CPU Time = .5 seconds

Since the I/O time is unaffected by the performance increase, it still takes 1 second to do I/O. Therefore, it takes 1 + .5 = 1.5 seconds to run Program A on the faster CPU

Wall clock Time = 1.5 seconds

System Performance in MFLOPS =

Number of Floating-Point Operations * 10^6 / Wall clock Time

Old System Performance (10 MFLOP/sec) = #FLOP * $10^6/4$

#FLOP = $40 * 10^6$

New System Performance = $40 * 10^6 / 1.5$

= 26.67 MFLOP/sec

4. A. This question considers the basic, RISC-V, 5-stage pipeline. (In this problem, you may assume that there is full forwarding.)

(i) Explain how pipelining can improve the performance of a given instruction mix

Pipelining provides "pseudo-parallelism" - instructions are still issued sequentially, but their overall execution (i.e., from fetch to write back) overlaps

- Performance is improved via higher throughput

- An instruction (ideally) finishes every short Clock Cycle

(ii) Show how these instructions will flow through the pipeline:

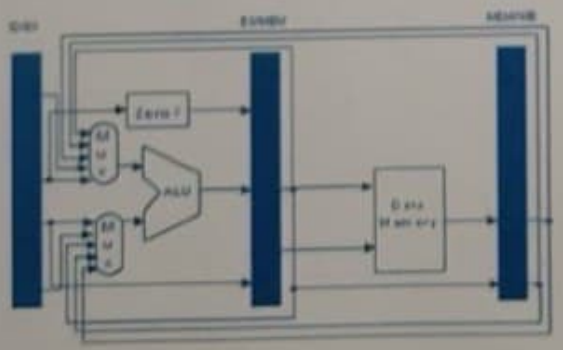
	1	2	3	4	5	6	7	8	9	10	11
lw x10, 0(x11)	F	D	E	M	W						
add x9, x11, x11		F	D	E	M	W					
sub x8, x10, x9			F	D	E	M	W				
lw x7, 0(x8)				F	D	E	M	W			
sw x7, 4(x8)					F	D	E	M	W		

(iii) Where might the sw instruction get its data from? Be very specific. (i.e. "from the lw instruction" is not a good answer!)

Data is available from the lw in the MEM / WB register

- There could be a feedback path from this register back to one of the inputs to data memory

- A control signal could then select the appropriate input to data memory to support this lw - sw input combination



8. This question considers the basic, RISC-V, 5-stage pipeline. (In this problem, you may assume that there is full forwarding.) Show how these instructions will flow through the pipeline below. For the instruction mix above, on what instruction results does the last add instruction depend on? predict that the beq instruction is not taken

	1	2	3	4	5	6	7	8	9	10	11	12
`beq x1, x2, X`	F	D	E									
`lw x10, 0(x11)`		F	D									
`sub x14, x10, x10`			F									
`X: add x4, x1, x2`				F	D	E	M	W				
`lw x1, 0(x4)`					F	D	E	M	W			
`sub x1, x1, x1`						F	D	D	E	M	W	
`add x1, x1, x1`							F	F	D	E	M	W