	IF	ID	EX	MEM	WB
a.	250ps	350ps	150ps	300ps	200ps
b.	200ps	170ps	220ps	210ps	150ps

Pipelining - How to optimize the execution of those instructions

i)

a) Pipelining to 5 stages reduces the cycle time to the length of the longest stage. Additionally, the cycle time needs to be slightly longer to accommodate the register at the end of the stage.
Pipelining CT= **350ps**

Cycle-time: there is no pipelining, so the cycle-time must allow an instruction to go through all the stages each cycle.
Non-Pipelining CT =250ps+350ps+150ps+300ps+200ps=**1250ps**

b) Pipelining CT=**220ps**
No Pipelining CT=200ps+170ps+220ps+210ps+150ps=**950ps**

ii) The LW (load word) instruction uses all 5 stages

a) For no pipelining the latency for an instruction is also the same, since each instruction takes 1 cycle to go from beginning fetch to the end of writeback. The throughput similarly is 1 cycle time instructions per second.
Latency =**1250ps**
Throughput=**1/1250 inst/ps**

For Pipelining the latency for both is 5 ∗ (cycle time), since an instruction needs to go through 5 pipeline stages, spending 1 cycle in each, before it commits. The throughput for both is still 1 instruction/cycle. Throughput increases because the cycle time reduces.
Latency=5*CT=5*350ps=**1750ps**
Throughput=1/CT=**1/220 inst/ps**

b) For no pipelining the latency for an instruction is also the same, since each instruction takes 1 cycle to go from beginning fetch to the end of writeback. The throughput similarly is 1 cycle time instructions per second
Latency =**950ps**
Throughput=**1/950 inst/ps**

For Pipelining the latency for both is 5 ∗ (cycle time), since an instruction needs to go through 5 pipeline stages, spending 1 cycle in each, before it commits. The throughput for both is still 1 instruction/cycle. Throughput increases because the cycle time reduces.
Latency=5*CT=5*220ps=**1100ps**
Throughput=1/CT=**1/220 inst/ps**

	ALU	BEQ	LW	SW
a.	45%	20%	20%	15%
b.	55%	15%	15%	15%

iii) Single Cycle each instruction takes one cycle to execute

- Clock period =
- Performance=Clock period/instruction

Multi-Cycle has the same clock cycle time as the pipelined organization. In pipelined, a long running program with no pipeline stalls completes one instruction in every cycle

- Load completes in 5 cycles
- Store completes in 4 cycles
- ALU completes in 4 cycles
- Branch completes in 4 cycle

a)

1) Multiple Cycle Execution is X times pipeline execution where X is:

Type	**CPI(*i*) for type**	**Frequency**	**CPI(*i*) *frequency**
ALU	4	45%	1.8
BEQ	4	20%	0.8
LOAD	5	20%	1.0
STORE	4	15%	0.6
Total Cycles	**17**		
Average CPI	**4.2 cycles**		

Execution time= 4.2*350ps*17= **25ns**

2) Single Cycle Execution is X times pipelined execution where X is:

$$\frac{\boldsymbol{Cycle\ time(Non-pipeline)}}{\boldsymbol{Cycle\ time\ (pipeline)}}$$

1250/350 =**3.57 cycles**

Execution Time=3.57*350ps*17=**21.25ns**

b)

1) Multiple Cycle Execution is X times pipeline execution where X is:

Type	**CPI(*i*) for type**	**Frequency**	**CPI(*i*) *frequency**
ALU	4	55%	2.20
BEQ	4	15%	0.6
LOAD	5	15%	0.75
STORE	4	15%	0.60
Total cycles	**17**		
Average CPI	**4.15 cycles**		

Execution time=17*220ps*4.15=**15.52ns**

2) Single Cycle Execution is X times pipelined execution where X is:

$$\frac{\boldsymbol{Cycle\ time(Non-pipeline)}}{\boldsymbol{Cycle\ time\ (pipeline)}}$$

950/220 =**4.32 cycles**

Execution time=4.32*220ps*17=**16.15ns**