1. Recall the execution time equation:

$$CPU\_time_{X,P} = Instructions\_executed_P * CPI_{X,P} * Clock\_cycle\_time_X \quad (1)$$

Consider a basic machine with the following characteristics:

OP Type	Freq (f_i)	Cycles	CPI_i
`Load`	20%	5	1.0
`Store`	10%	3	0.3
`Branch`	20%	2	0.4
`ALU`	50%	3	1.5

Calculate CPI for each instruction type.

`Execution time before improvement: 3.2 * I * CCT. Since I and CCT remain unchanged, we will only consider CPI.`

How much faster would the machine be if:

(a) we added a cache to reduce average load time to 3 cycles?

`Originally, load instructions added 1.0 to the CPI. After decreasing the number of cycles by 2, they add .6 CPI. CPI after improvement: 2.8. Improvement: 3.2 - 2.8 /3.2 = 12.5%`

(b) we added a branch predictor to reduce branch time by 1 cycle?

`Originally branch instructions added .4 to the CPI. After reducing the number of cycles in half, they add .2 CPI. CPI after improvement: 3.0. Improvement: 3.2 - 3.0 / 3.2 = 6.25%`

(c) we could do two ALU operations in parallel?

`Originally ALU instructions added 1.5 to the CPI. Running two ALU operations in parallel is the same as reducing the number of cycles in half, down to .75 CPI. CPI after improvement: 2.45. Improvement: 3.2 - 2.45 /3.2 = 23.4%`

2. Use the basic machine table from question 1, but assume that the frequency column indicates the percentage of execution time spent in the corresponding instruction type. Use Amdahl's Law to answer the following: if you could decrease the cycle time of one of the instruction types by 1 cycle, which instruction type would you optimize? How would the new execution time compare to the original one?

$$Execution\_time\_after\_improvement = \frac{Time\_affected\_by\_improvement}{Amount\_of\_improvement} + Time\_unaffected\_by\_improvement \quad (2)$$

OP Type	Exec time	Cycles	New exec time	Improvement
`Load`	20%	5	.2 / 5/4 + .8 = .16 + .8 = .96	4%
`Store`	10%	3	.1 / 3/2 + .9 = .067 + .9 = .967	3.3%
`Branch`	20%	2	.2 / 2/1 + .8 = .1 + .8 = .9	10%
`ALU`	50%	3	.5 / 3/2 + .5 = .33 + .5 = .83	16.7%

`It's best to optimize ALU instructions. The expected improvement is 16.7%.`

3. Intel's Itanium (IA-64) ISA is designed to facilitate executing multiple instructions per cycle. If an Itanium processor achieves an average CPI of .3 (3 instructions per cycle), how much faster is it than a Pentium4 (which uses the x86 ISA) with an average CPI of 1?

(a) Itanium is three times faster

(b) Itanium is one third as fast

(c) **Not enough information** - `We need cycle time and number of instruction to calculate the execution times of these machines. Comparing only CPIs, just like comparing only CCTs is misleading.`

4. Assume the following delays for the main functional units of the single-cycle datapath shown below:

Functional Unit	Time Delay
`Memory`	5ns
`ALU`	4ns
`Register File`	3ns

Given the following instructions: `lw`, `sw` and `add`, calculate:

(a) Minimum time to perform each instruction

(b) Time required on a single-cycle datapath

Write your answers in the table below. State any assumptions.

```
(a) lw (in that order): access memory (to read instruction), read operands from register
    file, ALU (effective address calculation), access memory (to read data) and write
    back to register file (the data read from memory).
(b) sw (in that order): access memory (to read instruction), read operands from register
    file, ALU (effective address calculation), access memory (to write data).
(c) add (in that order): access memory (to read instruction), read operands from register
    file, ALU (ALU computation based on func value) and write back to register file
    (value computed in ALU).
```

Instruction	instruction time	instruction in Single-Cycle datapath
`lw`	20ns	20ns
`sw`	17ns	20ns
`add`	15ns	20ns

5. Consider the following set of instructions:

```
add $sp, $sp, -4
sub $v0, $a0, $a1
lw  $t0, 4($sp)
or  $s0, $s1, $s2
lw  $t1, 8($sp)
```

Assuming the instructions are executed on a **single-cycle machine** with 10ns cycle time, compute:

(a) cycle time - `10 ns`

(b) instruction latency - `10ns`

(c) instruction throughput - $\frac{1 instruction}{10ns}$

6. Assume that the code above is executed on a 5-stage **pipelined** machine discussed in lecture. You might first draw the pipeline diagram in the space below.

If a pipeline stage takes 2ns, calculate:

(a) cycle time - `2ns`

(b) instruction latency - $5 * 2ns = 10ns$, `just like single-cycle`

(c) instruction throughput - $\frac{5 instructions}{18ns}$ `- much better than single-cycle`