1 Consider the execution of a program which results in the execution of 2 million instructions on a 400-MHz processor. The program consists of four major types of instructions. The instruction mix and the CPI for each instruction type are given below based on the result of a program trace experiment:

Instruction Type	**CPI**	**Instruction Mix**
Arithmetic and logic	1	60%
Load/store with cache hit	2	18%
Branch	4	12%
Memory reference with cache miss	8	10%

Compute the average CPI when the program is executed on a uniprocessor with the above trace results. Compute the CPU time to execute the program.

The average CPI when the program is executed on a uniprocessor with the above trace results is:
$\text{CPI} = 0.6 + (2 \times 0.18) + (4 \times 0.12) + (8 \times 0.1) = 2.24$.

$$\text{CPU time} = \frac{\text{time}}{\text{program}} = \frac{\text{time}}{\text{cycle}} \times \frac{\text{cycles}}{\text{instruction}} \times \frac{\text{instructions}}{\text{program}}$$

Therefore, CPU time = $1/(400 \text{ x } 10^6) \times 2.24 \times 2\times10^6$ = 0.0112 sec = 11.2 milliseconds = $11.2 \text{ x } 10^{-3}$ sec

2 Suppose we have two implementations of the same instruction set architecture. Computer A has a clock cycle time of 250 ps and a CPI of 2.0 for some program, and computer B has a clock cycle time of 500 ps and a CPI of 1.2 for the same program. Which computer is faster for this program and by how much?

We know that each computer executes the same number of instructions for the program; let's call this number *I*.

First, find the number of processor clock cycles for each computer:

$$\text{CPU clock cycles}_\text{A} = I \times 2$$

$\text{CPU clock cycles}_B = I \times 1.2$
Now we can compute the CPU time for each computer:

$\text{CPU time}_A = \text{CPU clock cycles}_A \times$ Clock cycle time = $I \times 2 \times 250$ ps = $I \times 500$ ps

Similarly, CPU time of B can be calculated as:

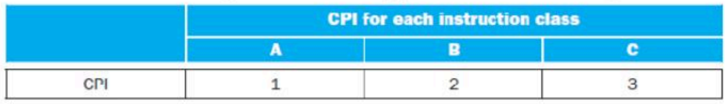
$\text{CPU time}_B = \text{CPU clock cycles}_B \times$ Clock cycle time = $I \times 1.2 \times 500$ ps = $I \times 600$ ps

Clearly, computer A is faster. The amount faster is given by the ratio of the execution times:

$$\frac{\text{CPU performance}_A}{\text{CPU performance}_B} = \frac{\text{Execution time}_B}{\text{Execution time}_A} = \frac{I \times 600 \text{ ps}}{I \times 500 \text{ ps}} = 1.2$$

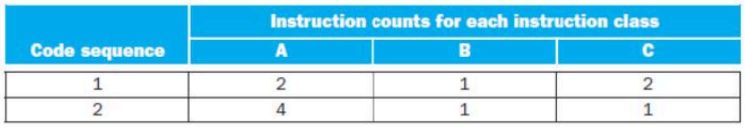
We can conclude that computer A is 1.2 times as fast as computer B for this program.

3 A compiler designer is trying to decide between two code sequences for a particular computer. The hardware designers have supplied the following facts:



	CPI for each instruction class		
	A	B	C
CPI	1	2	3

For a particular high-level language statement, the compiler writer is considering two code sequences that require the following instruction counts:



Code sequence	Instruction counts for each instruction class		
	A	B	C
1	2	1	2
2	4	1	1

Which code sequence executes the most instructions? Which will be faster? What is the CPI for each sequence?

Sequence 1 executes 2 + 1 + 2 = 5 instructions. Sequence 2 executes 4 + 1 + 1 = 6 instructions. Therefore, sequence 1 executes fewer instructions.
We can use the equation for CPU clock cycles based on instruction count and CPI to find the total number of clock cycles for each sequence:

$$\text{CPU clock cycles} = \sum_{i=1}^{n} CPI_i \times IC_i$$

This yields
$\text{CPU clock cycles}_1 = (2 \times 1) + (1 \times 2) + (2 \times 3) = 2 + 2 + 6 = 10$ cycles

$\text{CPU clock cycles}_2 = (4 \times 1) + (1 \times 2) + (1 \times 3) = 4 + 2 + 3 = 9$ cycles

So code sequence 2 is faster, even though it executes one extra instruction. Since code sequence 2 takes fewer overall clock cycles but has more instructions, it must have a lower CPI. The CPI values can be computed by:

$$\text{CPI} = \frac{\text{CPU clock cycles}}{\text{Instruction count}} = \frac{\sum_{i=1}^{n} CPI_i \times IC_i}{\text{Instruction count}}$$

$$\text{CPI}_1 = \frac{\text{CPU clock cycles}_1}{\text{Instruction count}} = \frac{10}{5} = 2$$

$$\text{CPI}_2 = \frac{\text{CPU clock cycles}_2}{\text{Instruction count}} = \frac{9}{6} = 1.5$$

4 We wish to compare the performance of two different machines: M1 and M2. The following measurements have been made on these machines:

Programs	**Computer M1 (sec)**	**Computer M2 (sec)**
P1	10	5
P2	3	4

a) Which machine is faster for each program and how much?

b) If the following additional measurements were made, find the instruction execution rate (instruction per second) for each machine when running P1.

Program	**Instructions executed on M1**	**Instructions executed on M2**
P1	200×10^6	160×10^6

c) If the clock rates of the machines M1 and M2 are 200 MHz and 300 MHz, respectively, find the clock cycles per instruction (CPI) for M1 and M2.

(a) For program P1, M2 is (10/5) = 2.0 times faster than M1.

For program P2, M1 is (4/3) = 1.33 times faster than M2.

(b) Since we know the number of instructions executed and the time it took to execute the instructions; we can easily calculate the number of instructions per second while running program P1 as:

Instructions per second for M1: $(200\text{x}10^6)/10\text{s} = 20\text{x}10^6$
Instructions per second for M2: $(160\text{x}10^6)/5\text{s} = 32\text{x}10^6$

(c) CPI can be calculated as:

$$\text{Cylces per instruction} = \frac{\text{Cycles per second}}{\text{Instructions per second}}$$

For M1, $\text{CPI}_{\text{M1}} = \frac{200 \times 10^6}{20 \times 10^6} = 10$

For M2, $\text{CPI}_{\text{M2}} = \frac{300 \times 10^6}{32 \times 10^6} = 9.4$

5 Assume that multiply instructions take 12 cycles (CPI) and account for 10% of the instructions in a typical program and that the other 90% of the instructions require an average of 4 cycles for each instruction (CPI). What percentage of time does the CPU spend doing multiplication?

Percentage of time that CPU spend doing multiplication can be calculated as:

$$(12 \times 0.1) \Big/ (12 \times 0.1 + 4 \times 0.9) = 25\%$$

Another, alternative way to look into this:
Assume 100 instructions, then the number of cycles will be $90 \times 4 + 10 \times 12 =$ 480 cycles. Among these, 120 cycles are spent doing multiplication, and thus 25% of the time is spent doing multiplication.

6 Suppose a program (or a program task) takes 1 billion instructions to execute on a processor running at 2 GHz. Suppose also that 50% of the instructions execute in 3 clock cycles, 30% execute in 4 clock cycles, and 20% execute in 5 clock cycles. What is the execution time for the program or task?

We have the instruction count: 10^9 instructions.

The clock cycle time can be computed quickly from the clock rate as:
$1/(2 \times 10^9) = 0.5 \times 10^{-9}$seconds.

Execution time = Instruction Count × $\text{CPI}_{\text{overall}}$ × Clock cycle time =
$10^9 \times (0.5 \times 3 + 0.3 \times 4 + 0.2 \times 5) \times 0.5 \times 10^{-9}$ sec = 1.85 sec.

7 Suppose the processor in the previous question is redesigned so that all instructions that initially executed in 5 cycles now execute in 4 cycles. Due to changes in the circuitry, the clock rate has to be decreased from 2.0 GHz to 1.9 GHz. No changes are made to the instruction set. What is the overall percentage improvement?

$\text{CPU}_{\text{time (original)}} = 1.85$ sec

$$\text{CPU}_{\text{time (new)}} = 10^9 \times (0.5 \times 3 + 0.3 \times 4 + 0.2 \times 4) \times \frac{1}{1.9 \times 10^9}$$
$$= 1.8421052632$$

$$\text{Performance improvement} = \frac{1.85 - 1.8421052632}{1.85} = 0.43\%$$

You can also refer to the following link for an alternative solution:

https://www.d.umn.edu/~gshute/arch/performance-equation.xhtml#improvements-solution

8 Suppose that we are considering an enhancement that runs 10 times faster than the original machine but is only usable for 40% of the time. What is the overall speedup gained by incorporating the enhancement?

$\text{Fraction}_{\text{enhanced}}$, $f = 0.4$

$\text{Speedup}_{\text{enhanced}}$, $N = 10$

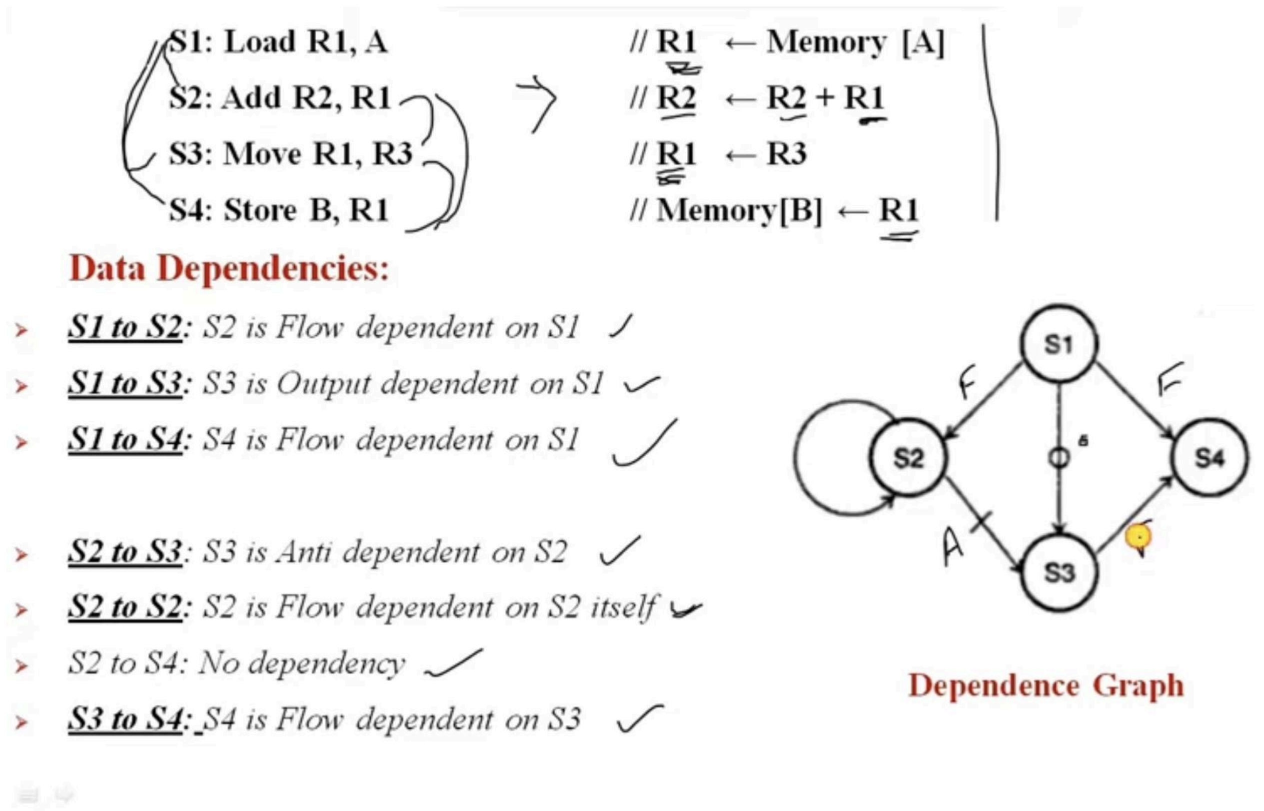
$$\text{Speedup}_{\text{overall}} = \frac{1}{(1-f)+\frac{f}{N}} = \frac{1}{(1-0.4)+\frac{0.4}{10}} = 1.5625$$

9 Complete the following table. Show your working towards the final answer.

What is the overall speedup if you make 10% of a program 90 times faster?	$\text{Speedup}_{\text{overall}} = \frac{1}{(1-0.1)+\frac{0.1}{90}} = 1.11$
What is the overall speedup if you make 90% of a program 10 times faster?	$\text{Speedup}_{\text{overall}} = \frac{1}{(1-0.9)+\frac{0.9}{10}} = 5.26$
The new CPU is 20 times faster on search queries than the old processor. The old processor is busy with search queries 70% of the time, what is the speedup gained by integrating the enhanced CPU?	$\text{Speedup}_{\text{overall}} = \frac{1}{(1-0.7)+\frac{0.7}{20}} = 2.985$
We are considering an enhancement to the processor of a server. The new CPU 10X faster. It is an I/O bound server, so 60% time waiting for I/O.	$\text{Speedup}_{\text{overall}} = \frac{1}{(1-0.4)+\frac{0.4}{10}} = 1.56$

10. Analyze the data dependencies in following statements and draw the dependence graph.

S1: Load R1, A
S2: Add R2, R1
S3: Move R1, R3
S4: Store B, R1



S1: Load R1, A	// R1 ← Memory [A]
S2: Add R2, R1	// R2 ← R2 + R1
S3: Move R1, R3	// R1 ← R3
S4: Store B, R1	// Memory[B] ← R1

Data Dependencies:

- ***S1 to S2***: *S2 is Flow dependent on S1* ✓
- ***S1 to S3***: *S3 is Output dependent on S1* ✓
- ***S1 to S4***: *S4 is Flow dependent on S1* ✓
- ***S2 to S3***: *S3 is Anti dependent on S2* ✓
- ***S2 to S2***: *S2 is Flow dependent on S2 itself* ✓
- *S2 to S4: No dependency* ✓
- ***S3 to S4***: *S4 is Flow dependent on S3* ✓

Dependence Graph

11. Analyze the data dependencies in following statements:
  S1: LOAD R1, M(100)
  S2: MOV R2, R1
  S3: INC R1
  S4: ADD R2, R1
  S5: STORE M(100), R1

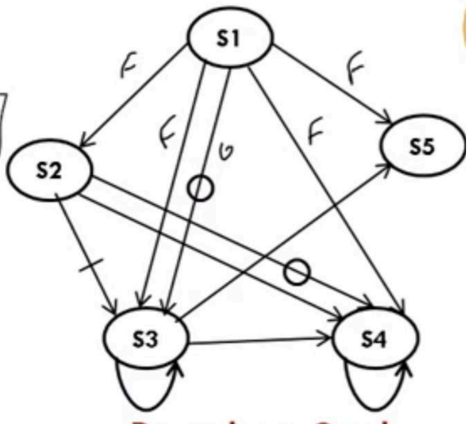
a) Draw dependency graph to show all the dependencies
b) Are there any resource dependencies if only one copy of each functional unit is available

Solution:

```
S1: LOAD R1, M(100)        // R1 ← MEM(100)
S2: MOV R2, R1             // R2 ← R1
S3: INC R1                 // R1 ← R1 + 1
S4: ADD R2, R1             // R2 ← R2 + R1
S5: STORE M(100), R1       // MEM(100) ← R1
```

Data Dependencies:

- ***S1 to S2***: *S2 is flow dependent on S1* ✓
- ***S1 to S3***: *S3 is flow and output dependent on S1* ✓
- ***S1 to S4***: *S4 is flow dependent on S1*
- ***S1 to S5***: *S5 is flow dependent on S1* ✓
- ***S2 to S3***: *S3 is anti dependent on S2* ✓
- ***S2 to S4***: *S4 is flow and output dependent on S2* ✓
- *S2 to S5: No dependency* ✓
- ***S3 to S4***: *S4 is flow dependent on S3*
- ***S3 to S3***: *S3 is flow dependent on S3 itself* ✓
- ***S4 to S4***: *S4 is flow dependent on S4 itself* ✓
- ***S3 to S5***: *S5 is flow dependent on S3*
- *S4 to S5: No dependency*



Dependence Graph

Resource Dependency:

- ***S3 and S4:*** *Both are using* ***adders (+***