Compilers can have a profound impact on the performance of an application. Assume that for a program, compiler A results in a dynamic instruction count of 1.0E9 and has an execution time of 1.1 s, while compiler B results in a dynamic instruction count of 1.2E9 and an execution time of 1.5 s.

a. Find the average CPI for each program given that the processor has a clock cycle time of 1 ns.

b. Assume the compiled programs run on two different processors. If the execution times on the two processors are the same, how much faster is the clock of the processor running compiler A's code versus the clock of the processor running compiler B's code?

c. A new compiler is developed that uses only 6.0E8 instructions and has an

average CPI of 1.1. What is the speedup of using this new compiler versus using compiler A or B on the original processor?

a. $CPI = T_{exec} \times f/\text{No. instr.}$

Compiler A CPI = 1.1

Compiler B CPI = 1.25

b. $f_B/f_A = (\text{No. instr.(B)} \times CPI(B))/(\text{No. instr.(A)} \times CPI(A)) = 1.37$

c. $T_A/T_{new} = 1.67$

$T_B/T_{new} = 2.27$

Problem 1.4 Consider the execution of an object code with 2×10^6 instructions on a 400-MHz processor. The program consists of four major types of instructions. The instruction mix and the number of cycles (CPI) needed 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%

(a) Calculate the average CPI when the program is executed on a uniprocessor with the above trace results.

(b) Calculate the corresponding MIPS rate based on the CPI obtained in part (a).

a) Average CPI = 2.24
b) MIPS rate= 178.6

Divide - 145 by 13 in binary twos complement notation, using 12-bit words.

Divisor = 13 = $(001101)_2$ is placed in M register.
Dividend = –145 = $(111101101111)_2$ is placed in A and Q registers

A	Q	M	
111101	101111	001101	Initial
111011	011110		Shift
001101			Add
001000			
111011	011110		Restore
110110	111100		Shift
001101			Add
000011			
110110	111100		Restore
101101	111000		Shift
001101			Add
111010	111001		$Q_0 \leftarrow 1$
110101	110010		Shift
001101			Add
000110			
110101	110010		Restore
101011	100100		Shift
001101			Add
111000	100101		$Q_0 \leftarrow 1$
110001	001010		Shift
001101			Add
111110	001011		$Q_0 \leftarrow 1$

Remainder = $(111110)_2$ = –2
Quotient = twos complement of 001011 = $(110101)_2$ = –11

Given x = 0101 and y = 1010 in twos complement notation (i.e., x = 5, y = - 6), compute the product p = x * y with Booth's algorithm.

A	Q	Q_{-1}	M	
0000	1010	0	0101	Initial
0000	0101	0	0101	Shift
1011	0101	0	0101	A ← A – M
1101	1010	1	0101	Shift
0010	1010	1	0101	A ← A + M
0001	0101	0	0101	Shift
1100	0101	0	0101	A ← A – M
1110	0010	1	0101	Shift

Use the Booth algorithm to multiply 23 (multiplicand) by 29 (multiplier), where each number is represented using 6 bits.

A	Q	Q_{-1}	M	
000000	010111	0	010011	Initial
101101	010111	0	010011	A ← A – M
110110	101011	1	010011	Shift
111011	010101	1	010011	Shift
111101	101010	1	010011	Shift
010000	101010	1	010011	A ← A + M
001000	010101	0	010011	Shift
110101	010101	0	010011	A ← A – M
111010	101010	1	010011	Shift
001101	101010	1	010011	A ← A + M
000110	110101	1	010011	Shift

Express the following numbers in IEEE 32-bit floating-point format:
a. - 5
b. -6
c. - 1.5
d. 384
e. 1/16
f. -1/32

a. 1 10000001 01000000000000000000000

b. 1 10000001 10000000000000000000000

c. 1 01111111 10000000000000000000000

d. 384 = 110000000 = 1.1 × 21000
Change binary exponent to biased exponent:
127 + 8 = 135 = 10000111
Format: 0 10000111 00000000000000000000000
e. 1/16 = 0.0001 = 1.0 × 2–100
127 – 4 = 123 = 01111011
Format: 0 01111011 00000000000000000000000
f. –1/32 = –0.00001 = –1.0 × 2–101
127 – 5 = 122 = 01111010
Format: 0 01111010 00000000000000000000000

The following numbers use the IEEE 32-bit floating-point format. What is the equiv-
alent decimal value?
a. 1 10000011 11000000000000000000000
b. 0 01111110 10100000000000000000000
c. 0 10000000 00000000000000000000000

a. –28

b. 13/16 = 0.8125

c. 2

Consider the following code:

```
for (i = 0; i <20; i++)
for ( j = 0; j < 10; j++)
a[i] = a[i]* j;
```

a. Give one example of the spatial locality in the code.
b. Give one example of the temporal locality in the code.

a. A reference to the first instruction is immediately followed by a reference to the second.
b. The ten accesses to a[i] within the inner for loop which occur within a short interval of time.

Consider a memory system that uses a 32-bit address to address at the byte level, plus a cache that uses a 64-byte line size.

a. Assume a direct mapped cache with a tag field in the address of 20 bits. Show the address format and determine the following parameters: number of addressable units, number of blocks in main memory, number of lines in cache, size of tag.

b. Assume an associative cache. Show the address format and determine the following parameters: number of addressable units, number of blocks in main memory, number of lines in cache, size of tag.

c. Assume a four-way set-associative cache with a tag field in the address of 9 bits. Show the address format and determine the following parameters: number of addressable units, number of blocks in main memory, number of lines in set, number of sets in cache, number of lines in cache, size of tag.

a. Address format: Tag = 20 bits; Line = 6 bits; Word = 6 bits
Number of addressable units = $2^{s+w} = 2^{32}$ bytes; number of blocks in main memory = $2^s = 2^{26}$; number of lines in cache $2^r = 2^6 = 64$; size of tag = 20 bits.
b. Address format: Tag = 26 bits; Word = 6 bits
Number of addressable units = $2^{s+w} = 2^{32}$ bytes; number of blocks in main memory = $2^s = 2^{26}$; number of lines in cache = undetermined; size of tag = 26 bits.
c. Address format: Tag = 9 bits; Set = 17 bits; Word = 6 bits
Number of addressable units = $2^{s+w} = 2^{32}$ bytes; Number of blocks in main memory = $2^s = 2^{26}$; Number of lines in set = k = 4; Number of sets in cache = 2^d
= 2^{17}; Number of lines in cache = $k \times 2^d = 2^{19}$; Size of tag = 9 bits.

A computer has a cache, main memory, and a disk used for virtual memory. If a referenced word is in the cache, 20 ns are required to access it. If it is in main memory but not in the cache, 60 ns are needed to load it into the cache, and then the reference is started again. If the word is not in main memory, 12 ms are required to fetch the word from disk, followed by 60 ns to copy it to the cache, and then the reference is started again. The cache hit ratio is 0.9 and the main memory hit ratio is 0.6. What is the average time in nanoseconds required to access a referenced word on this system?

Location of referenced word	Probability	Total time for access in ns
In cache	0.9	20
Not in cache, but in main memory	(0.1)(0.6) = 0.06	60 + 20 = 80
Not in cache or main memory	(0.1)(0.4) = 0.04	12ms + 60 + 20 = 12,000,080

For the 8-bit word 00111001, the check bits stored with it would be 0111. Suppose when the word is read from memory, the check bits are calculated to be 1101. What is the data word that was read from memory?

The Hamming Word initially calculated was:

bit number:	12	11	10	9	8	7	6	5	4	3	2	1
	0	0	1	1	0	1	0	0	1	1	1	1

Doing an exclusive-OR of 0111 and 1101 yields 1010 indicating an error in bit 10 of the Hamming Word. Thus, the data word read from memory was 00011001.

A distinction is made between physical records and logical records. A logical record is a collection of related data elements treated as a conceptual unit, independent of how or where the information is stored. A physical record is a contiguous area of storage space that is defined by the characteristics of the storage device and operating system.

Assume a disk system in which each physical record contains thirty 120-byte logical records. Calculate how much disk space (in sectors, tracks, and surfaces) will be required to store 300,000 logical records if the disk is fixed-sector with 512 bytes/sector, with 96 sectors/track, 110 tracks per surface, and 8 usable surfaces. Ignore any file header record(s) and track indexes, and assume that records cannot span two sectors.

Each sector can hold 4 logical records.

The required number of sectors is 300,000/4= 75,000 sectors.

This requires 75,000/96 = 782 tracks, which in turn requires

782/110 = 8 surfaces.

Consider a disk that rotates at 3600 rpm. The seek time to move the head between adjacent tracks is 2 ms. There are 32 sectors per track, which are stored in linear order from sector 0 through sector 31. The head sees the sectors in ascending order. Assume the read/write head is positioned at the start of sector 1 on track 8. There is a main memory buffer large enough to hold an entire track. Data is transferred between disk locations by reading from the source track into the main memory buffer and then writing the date from the buffer to the target track.

a. How long will it take to transfer sector 1 on track 8 to sector 1 on track 9?

b. How long will it take to transfer all the sectors of track 8 to the corresponding sectors of track 9?

a.

The time consists of the following components: sector read time; track access time; rotational delay; and sector write time. The time to read or write 1 sector is calculated as follows:

A single revolution to read or write an entire track takes 60,000/360 = 16.7 ms.

Time to read or write a single sector = 16.7/32 = 0.52 ms. Track access time = 2 ms, because the head moves between adjacent tracks. The rotational delay is the time required for the head to line up with sector 1 again. This is 16.7 × (31/32) = 16.2 ms. The head movement time of 2 ms overlaps the with the 16.2 ms of rotational delay, and so only the rotational delay time is counted. Total transfer time = 0.52 + 16.2 + 0.52 = 17.24 ms.

b.

The time to read or write an entire track is simply the time for a single revolution, which is 16.7 ms. Between the read and the write there is a head movement time of 2 ms to move from track 8 to track 9. During this time the head moves past 3 sectors and most of a fourth sector. However, because the entire track is buffered, sectors can be written back in a different sequence from the read sequence. Thus, the write can start with sector 5 of track 9. This sector is reached 0.52 × 4 = 2.08 ms after the completion of the read operation. Thus the total transfer time = 16.7 + 2.08 + 16.7 = 35.48 ms.

Consider a hard disk with:

- ***4 surfaces***
- ***64 tracks/surface***
- ***128 sectors/track***
- ***256 bytes/sector***

What is the capacity of the hard disk?

- ***Disk capacity = surfaces * tracks/surface * sectors/track * bytes/sector***
  *Disk capacity = 4 * 64 * 128 * 256*
  Disk capacity = 8 MB

The disk is rotating at 3600 RPM, what is the data transfer rate?

- *60 sec -> 3600 rotations*
  1 sec -> 60 rotations
  Data transfer rate = number of rotations per second * track capacity * number of surfaces (since 1 R-W head is used for each surface)
  *Data transfer rate = 60 * 128 * 256 * 4*
  Data transfer rate = 7.5 MB/sec

The disk is rotating at 3600 RPM, what is the average access time?

- *Since seek time, controller time and the amount of data to be transferred is not given, we consider all three terms as 0.*
  Therefore, ***Average Access time = Average rotational delay***
  Rotational latency => 60 sec -> 3600 rotations
  1 sec -> 60 rotations
  Rotational latency = (1/60) sec = 16.67 msec.
  Average Rotational latency = (16.67)/2
  = 8.33 msec.
  Average Access time = 8.33 msec.

A magnetic disk has 100 cylinders, each with 10 tracks of 10 sectors. If each sector contains 128 bytes, what is the maximum capacity of the disk in kilobytes?

Data:

number of cylinders = C= 100

number of tracks = T = 10

sectors per track = S = 10

number of byes in each sector = D= 128

Formula:

Disk capacity = C × T × S × D bytes

Calculation:

Disk capacity = 100 × 10 × 10 × 128 Bytes

$$\text{Disk capacity} = \frac{10^4 \times 128}{1024} \text{ kilobytes} = 1250 \text{ KB}$$

NOTE:

In official answer key, option B is the correct answer.

K is taken as 1000. Therefore,

$$\text{Disk capacity} = \frac{10^4 \times 128}{1000} \text{ kilobytes} = 1280 \text{ KB}$$

tB = tS + tL + tT

$$t_T = \frac{b}{rN}$$

where

b = number of bytes to be transferred
N = number of bytes on a track
r = rotation speed, in revolutions per second

Thus the total average block read or write time T_{total} can be expressed as

$$t_B = t_S + \frac{1}{2r} + \frac{b}{rN} \quad \textbf{(7.1)}$$