Show all your work to get any marks; no work = no marks!

Hand-write with pen and paper, then upload images or a PDF file of the exam before 9:30 PM.

Print and fill out the attached exam, with pen and paper. If you do not have a printer, just hand-write your answers with pen and paper instead; please keep the answers in order (starting at #1, etc). Upload a PDF file of your final exam submission; answer questions in order. Show ALL your work for every question. If you have questions, email Jason_Wilder@bcit.ca

Full Name: Monika Szucs

Signature: Monika Szucs

Student Number: A00878767

Score: ________ out of 21

1. a) Consider a multilevel computer in which all the levels are different. Each level has instructions that are 4 times as powerful as those of the level below it; that is, one level m instruction can do the work of 4 level m – 1 instructions. If a level-6 program requires 2 attoseconds to run, how long would equivalent programs take at levels 1 and 19, assuming 3 level m instructions are required to interpret a single m + 1 instruction? **(2 marks)**
   b) Also, what is wrong in that question? Explain. **(1 mark)** atto seconds 2×10^{-18} s

a) level 19 2 attosec $\left(\frac{3}{4}\right)^{13}$ = 0.0475 attosec

7 8 9 10 11 12 13 14 15 16 17 18 19

level 6 2 attosec

5 4 3 2

Level 1 2 attosec $\left(\frac{4}{3}\right)^{5}$ = 8.42798 attosec

Explain
b) This question is flawed because it thinks 3 level m instructions are required to interpret a m+1 instruction which contradicts the m+1 instruction is 4 times as powerful

Period = $\frac{1}{\text{Frequency}}$

2. Fill in the rows of this table: **(1 mark)**

	Frequency	Corresponding period
a	25kHz	0.000004 s
b	5×10^{7} Hz	20 nanoseconds
c	125Hz	0.008 s
d	3×10^{4} Hz	33.3333 microseconds

25 kHz $\times 10^3$ = 25,000 Hz

$\frac{1}{25000}$ = 0.000004 s

$\frac{1}{20\times10^{-9}\text{ s}} = \frac{1}{20}\times10^{9}$ Hz

$\frac{1}{125}$ = 0.008 s

$\frac{1}{33.3333\times10^{-6}\text{ s}} = \frac{1}{33.3333}\times10^{6}$ Hz

3. Consider a pipeline whose stages take 14 nanoseconds, 16000 femtoseconds, 0.00003 milliseconds, and 0.15 microseconds. What are a) its latency and b) its bandwidth? **(2 marks)**

→ $\boxed{14\text{ns}}$ $\boxed{16000 \text{ femtosec}}$ $\boxed{0.00003 \text{ millisec}}$ $\boxed{0.15 \text{ microsec}}$ →

14×10^{-3} µs 16000×10^{-9} µs 0.00003×10^{3} µs

→ $\boxed{0.014\,\mu s}$ $\boxed{0.016\,\mu s}$ $\boxed{0.03\,\mu s}$ $\boxed{0.15\,\mu s}$ →

latency 0.014 + 0.016 + 0.03 + 0.15 = 0.21 µs/data

bandwidth $\frac{1 \text{ data}}{0.15\,\mu s}$ = 6.67 data/µs

4. a) What is the Hamming Distance of a code whose words are 00000011, 10101010, 00001111, and 11110000?

00000011 and 0000 1111 is 2
00000011 and 11110000 is 6
00000011 and 10101010 is 4
00001111 and 1111 0000 is 8
0000 1111 and 1010 1010 is 4
11110000 and 10101010 is 4

b) How many errors can it correct? c) What are the properties of a good error-correcting code? Why are these good properties? Explain clearly. **(2 marks)**

b) # of correctable errors
min Hamming distance = 2

$\frac{2-1}{2} = \frac{1}{2} = 0$ errors

c) - the propries of good error correcting code is having a large minimum hamming distance making it easier for error detection and correction
- balance between clairity and space
- codewords are significantly far apart to avoid ambiguity
- ability to correct errors affecting fewer than half the bits
- No undetectable or uncorrectable errors

these properties are good because a larger hamming distance increases clairity (error detection/correction) while balancing space (redundant bits) ensuring errors can be detected and corrected effectively.

5. The following Hamming codeword was made using even parity. 1101101. a) Was there an error? b) Where? c) What was the original dataword supposed to have been? d) Explain the limitations of Hamming code in this question. Under what circumstances might Hamming code have failed here? **(2 marks)**

even

×	✓		×			
1	1	0	1	1	0	1
1	2	2	4	4	4	4
		1		1	2	2
						1
P	P		P			

1+1+0+1 = 3 ×
1+0+0+1 = 2 ✓
1+0+1+1 = 3 ×

a) yes there was an error at either bits 1 or 4 or 5
b) most likely at 1+4 = bit 5
c) original data word
0001

d) The limitations of the hamming code here is hamming codewords can only correct single bit errors. If there are two or more errors the code might not correct it or not find it
Hamming codewords cant handle big amounts of errors together
Hamming code words require parity bits which increase the size of the codeword
If there are too many errors it might be undetectable or uncorrectable

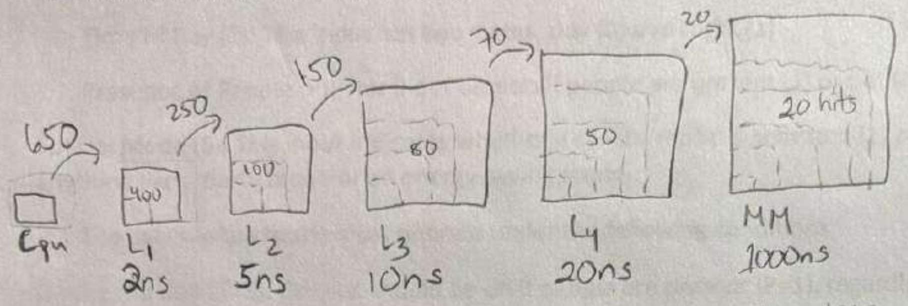
6. Create the odd-parity Hamming codeword for the dataword 010110111. Clearly identify parity bits. **(1 mark)**

0	1	0	1	1	0	1	1	1	0	1	1	1
1	2	2	4	4	4	4	8	8	8	8	8	8
		1		1	2	2		1	2	2	4	4
						1				1		1
P	P						P					

1+1+0+1+1+1 = 5
1+1+0+1+1+1 = 5
1+0+0+1+0+1 = 3
0+0+1+1+1+1+1

7. What is the mean memory access time for a system with four levels of cache and a main memory. The access times for these respectively are 2ns, 5ns, 10ns, 20ns, and 1 microsecond. 650 memory accesses were made. 400 hits were in level 1; 100 in level 2; 80 in level 3; 50 in level 4; the rest were in main memory. **(2 marks)**

1 micro x 1000
= 1000ns



650 * 2ns + 250 * 5ns + 150 * 10ns + 70 * 20ns + 20 * 1000ns = 25450ns total for 650 requests

25450ns / 650 req = 39.15384615 ns/req

8. How long does it take to read a disk with 4000 cylinders, each containing seven tracks of 256 sectors? First, all the sectors of track 0 are to be read starting at sector 0, then all the sectors of track 1 starting at sector 0, and so on. The rotation rate is 1200 RPM, and a seek takes 4 msec between adjacent cylinders and 40 msec for the worst case. Switching between tracks of a cylinder can be done in 13 msec. **(2 marks)**

Tracks 4000 per platter
draw 7 of ? disks
draw 256 of ? pie slices/sectors
1200 RPM → 1200/60 = 20s/rev → 1/20 = 0.05 s/rev x1000 = 50 ms/rev

step 1
move to track 0
best case 0
worst case 40msec
average case (0+40)/2 = 20 msec

step 2
wait for sector 0 to spin to the read/write head 50ms/rev / 2 = 25ms/rev

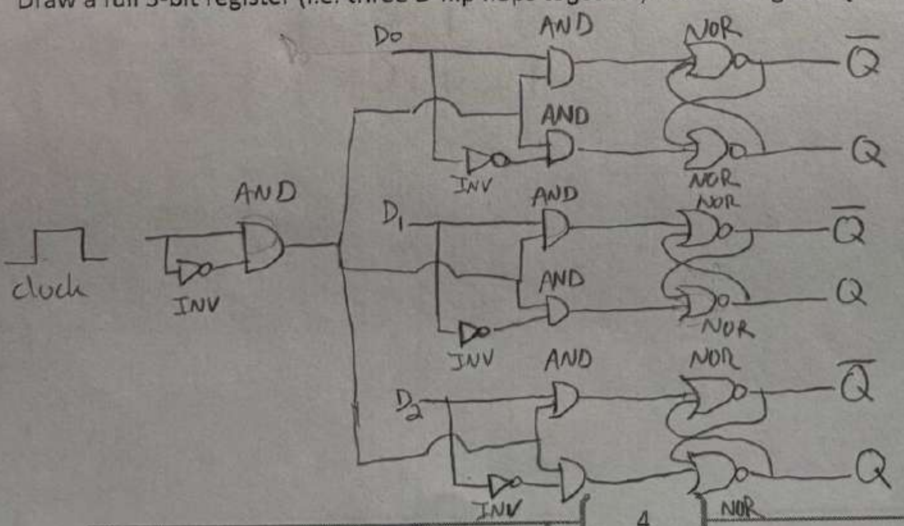
step 3
-read track of first platter 50ms/rev
-switch to next platter 13 msec
-wait for sector 0 50ms − 13ms = 37ms

step 4
do step 3 seven times

step 5 -seek adjacent cylinder 4ms
-repeat steps 3-4-5 4000 times

40ms + 25ms + 7 (50ms + 13ms + 37ms) * 4000 = 2800065 / 1000
= 2800.065s

9. Draw a full 3-bit register (i.e. three D-flip flops together). Label the gates. **(2 marks)**



10. Imagine you are tasked with designing a control system for the smart lighting in a home. The system has four lights (L1, L2, L3, L4) and is controlled by three inputs: Time of Day (T), Presence of People (P), and Special Mode (S).

Time of Day (T): This input has two states, Day (0) and Night (1).

Presence of People (P): This input detects if people are present (1) or not (0).

Special Mode (S): This input indicates whether a special mode is activated (1) or not (0). Special mode could be anything like a party mode or an energy-saving mode.

The lights in the home must operate under the following conditions:

During the Day (T=0): Only L2 should be on if people are present (P=1), regardless of the Special Mode (S).

During the Night (T=1) and No Special Mode (S=0): L1 and L3 should be on if people are present (P=1); otherwise, all lights should be off.

During the Night (T=1) and Special Mode (S=1): L4 should be on, and L1 should also be on if people are present (P=1), regardless of their presence for L4.

Your task is to:

a) Develop a truth table that for the logic to turn each light on or off based on the inputs T, P, and S. **(1 mark)**

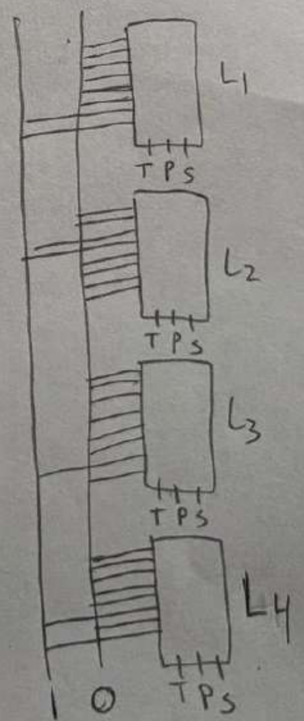
b) Design a set of multiplexers that implement this logic for controlling the lights. **(1 mark)**

a) Truth table:

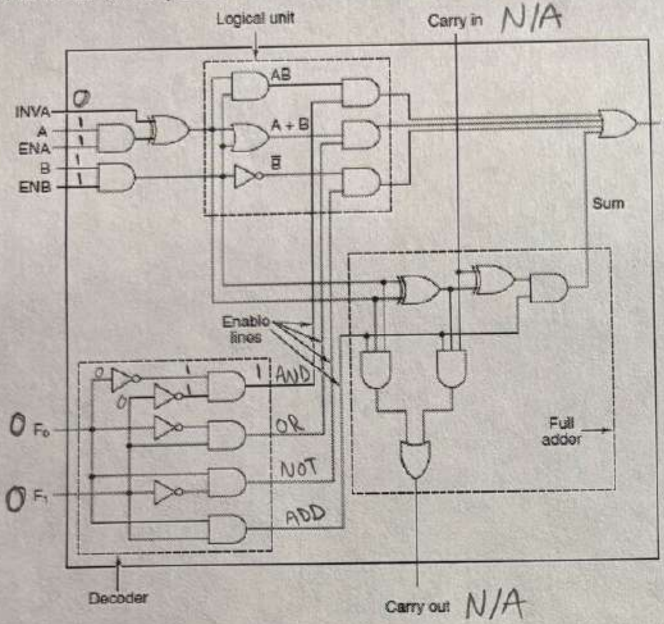
T	P	S	L1	L2	L3	L4
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	1	0	0
0	1	1	0	1	0	0
1	0	0	0	0	0	0
1	0	1	0	0	0	1
1	1	0	1	0	1	0
1	1	1	1	0	0	1

Multiplexer

b)



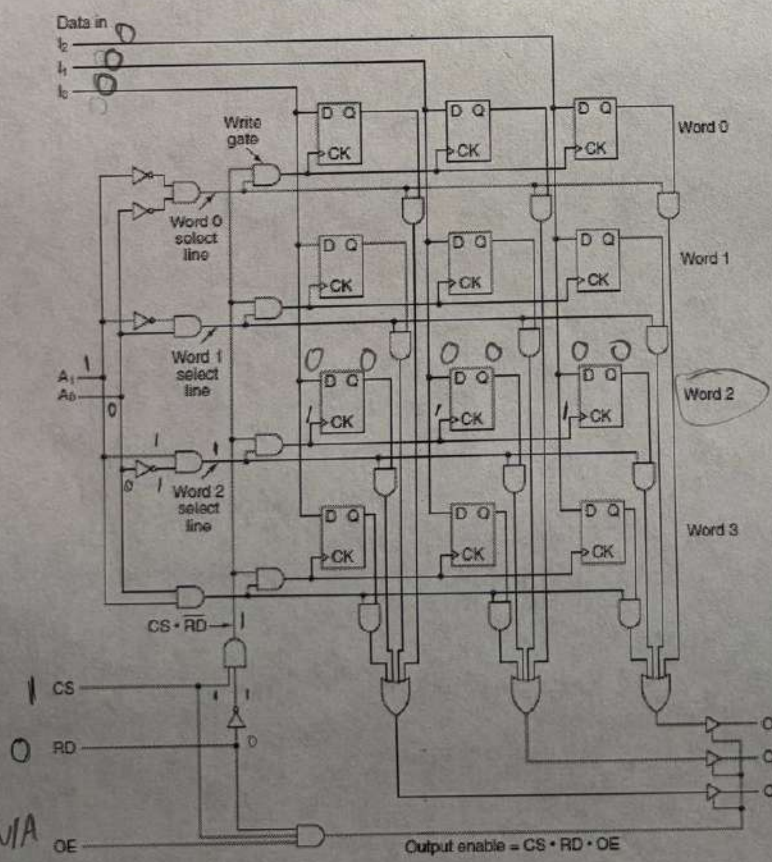
11. Fill in all the inputs and outputs for this diagram. The circuit must perform the operation "1 and 1": **(1 mark)**



AND 0 0

12. Fill in all the inputs and outputs for this diagram to perform the operation "write 000 to word 2": **(1 mark)**

1 0
2 1



NONE