COIS 2300H Assignment 4

This assignment is worth 10% of your final Grade

A. Textbook Question 5.3 – 6 marks

Based on textbook section 5.3

For a direct-mapped cache design with a 32-bit address, the following bits of the address are used to access the cache:

Tag 22 bits	Index 5 bits	Offset 5 bits
31–10	9–5	4–0

1) What is the cache block size (in words)?

32-bit address = 4 bytes
Offset (0-4) is of size = 5 bits
Therefore, cache block size = 2^5 bytes = 32 bytes
Therefore, cache block size in word
= $2^5/4$ = 8 words

2) How many entries does the cache have?

Index (9-5) = 5 bits
So number of entries = 2^5 = 32 entries

3) What is the ratio between total bits required for such a cache implementation over the data storage bits?

Data bits = 8words*32 entries = 256 bits
Tag bits = 22 bits
So total bits needed for cache implementation = 256+22 = 279 bits
Therefore,
Ratio = 279 / 256 = 1.086

4) The following table represents the byte-addressed cache references starting from power on. Time 0 is the first memory access, and time 15 is the most recent memory access. Fill in the table by converting the address to binary, deriving the index number, and recording if the entry in the cache was replaced.
Only the bolded values (added by Sri) are worth marks.

Time	Address(A)	Address in Binary	Index	Cache Entry Was Replaced (Yes or No)
0	0	00000000000000000000 00000 00000	00000	Miss – empty cache
1	4	00000000000000000000 00000 00100	00000	Hit
2	16	00000000000000000000 00000 10000	00000	Hit
3	132	00000000000000000000 00100 00100	00100	Miss
4	232	00000000000000000000 00111 01000	00111	Miss
5	160	00000000000000000000 00101 00000	00101	miss
6	1024	00000000000000000001 00000 00000	00000	Miss – replacement
7	30	00000000000000000000 00000 11110	00000	Miss - replacement
8	140	00000000000000000000 00100 01100	00100	Hit
9	3100	00000000000000000011 00000 11100	00000	Miss – replacement
10	180	00000000000000000000 00101 10100	00101	Hit
11	2180	00000000000000000010 00100 00100	00100	Miss- replacement

12	**3505**	00000000000000000011 01101 10001	01101	Miss – empty
13	**185**	00000000000000000000 00101 11001	00101	Hit
14	**869**	00000000000000000000 11011 00101	11011	Miss - empty
15	**7041**	00000000000000000110 11100 00001	11100	Miss - empty

5) What is the hit ratio?

Hit Ratio = 5/16 = 0.3125

6) List the final state of the cache, with each valid entry represented as a record of <index, tag, data>. There may be extra rows, or blank rows.

Index	Tag	Data
00000	00000000000000000011	0…..31
00100	00000000000000000010	128…159
00111	00000000000000000000	224…255
00101	00000000000000000000	160…191
01101	00000000000000000011	3488…3519
11011	00000000000000000000	864….895
11100	00000000000000000110	7040….7071

B. Textbook question 5.8 with adjustment by Sri – 4 marks.

Based on section 5.5 in the textbook.

Mean Time Between Failures (MTBF), Mean Time To Replacement (MTTR), and Mean Time To Failure (MTTF) are useful metrics for evaluating the reliability and availability of a storage resource. Explore these concepts by answering the questions about devices with the following metrics.

Item	**MTTF**	**MTTR**	**MTBF** **= MTTF + MTTR**	**Availability** **= MTTF / (MTTF + MTTR)**
Smart phone	3 years	1 day	3*365 + 1 = 1095 + 1 = 1096 =1096 * 24 hours = 26,304 hours	3*365/1096 = 1095/ 1096 = 0.999 = 99.9%
Any desktop hard drive	6 years	2 days	6*365 + 2 = 2190 + 2 = 2192 =2192 * 24 hours = 52,608 hours	6*365/2192 =2190/2192 =0.999 = 99.9%
Specific desktop hard drive (order through Amazon)	6 years	4 days	6*365 + 4 = 2190 + 4 = 2194 =2194 * 24 hours = 52,656 hours	6*365/2194 =2190/2194 =0.998 = 99.8%
Enterprise SSD with onsite replacement warranty	167 years	4 hours = 0.167 days	167*365 * 24 = 60,955 *24 = 1462920 + 4 = 1,462,924 hours	167*365/(60955+0.167) =60955/60955.167 =0.999 = 99.9%

1) Calculate the MTBF for each of the devices in the table by filling in the MTBF column in the chart above.

2) Calculate the availability for each of the devices in the table by filling in the Availability column in the chart above.

3) a) What happens to availability as the MTTR approaches 0?

As MTTR approaches 0, the availability becomes 100%.
Because when MTTR is 0, the MTBF would be only equal to MTTF, thus making availability = MTTF/MTTF.

b) Is this a realistic situation?

If the mean time to replacement is 0, that means we can easily replace any hardware, and thus making it feasible, as there are cheap drives coming. However, replacing file systems and data takes significant time. Thus it is not a realistic situation, as it does take time to replace a disk because of the data.

4) a) What happens to availability as the MTTR gets very high, i.e., a device is difficult to repair?

As the MTTR becomes very high, the availability should decrease, according to the MATH. However, it depends on the MTTF as well. Because if MTTF is very much greater than MTTR, than even if MTTR is high, the availability would still be high since MTTR won't have much effect.

b) Does this imply the device has low availability? (Sri note: This is a real problem with NAS/SAN's where a 'drive' is really a box with many drives in it, and fixing a SAN could require a specialist electronics technician to diagnose the board and solder replacement components on, this has happened at Trent before).

No, the availability is not low. As discussed before,
Let's say MTTF is at least 50 times more than MTTR, then even very high MTTR wouldn't decrease availability by a lot.

C. Textbook Question 6.10 – 4 marks.

Based on second 6.4 in the textbook.

Answers for the two questions combined should fit on one page, 1 or 2 sources that are just official webpages for the listed products are enough, as is a review comparing them.

Virtualization software is being aggressively deployed to reduce the costs of managing today's high-performance servers. Companies like VMWare, Microsoft and IBM have all developed a range of virtualization products. The general concept, described in Chapter 5, is that a hypervisor layer can be introduced between the
hardware and the operating system to allow multiple operating systems to share the same physical hardware.

The hypervisor layer is then responsible for allocating CPU and memory resources, as well as handling services typically handled by the operating system (e.g., I/O). Virtualization provides an abstract view of the underlying hardware to the hosted operating system and application soft ware. This will require us to rethink how
multi-core and multiprocessor systems will be designed in the future to support the sharing of CPUs and memories by a number of operating systems concurrently.

1) Select two hypervisors on the market today, and compare and contrast how they virtualize and manage the underlying hardware (CPUs and memory).

Virtualization is change from physical to logical. Using https://vapour-apps.com/what-is-hypervisor/ as a reference, there are two types of hypervisors : Type 1 and Type 2. I am going to compare these two hypervisors.

Type 1 hypervisors : *VMware ESX and ESXi*.
These hypervisors run directly on the system hardware. It is called a bare metal hypervisor, since, it's directly embedded into hardware. One type 1 hypervisor is ***VMware ESX and ESXi***.
Talking about this hypervisor, it has advanced features and scalability, (well what tech doesn't). However, it needs licensing for the user sand thus it's higher price. VMware is like the leader in the type 1 hypervisors. There are some ways to reduce the cost of this hypervisor technology. This

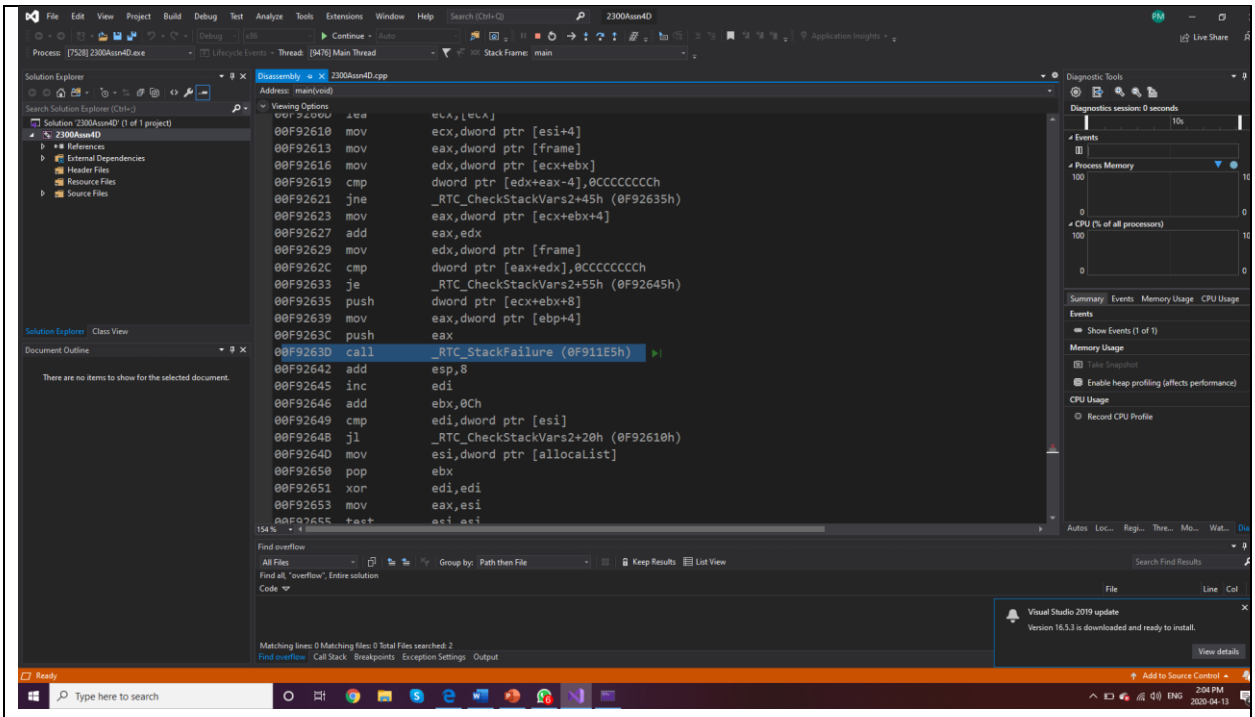
hypervisor doesn't have to load an underlying OS first. Since they are directly embedded into the hardware, they have direct access to the underlying hardware and thus more efficient, since there is no need for any other software for this purpose and also making it like a best performing hypervisor.

Type 2 hypervisors : *VMware Workstation/ Fusion/ Player*.
VMware Workstation/ Fusion/ Player is a type 2 hypervisor. Unlike the type 1 hypervisor it is installed over the OS and not into the hardware. It is a free virtualization hypervisor. It is intended to run only on one virtual machine thus having more advanced features like record/replay and VM snapshot support for that machine. Since it is over the OS, it is called a hosted hypervisor (it is like a parasite on the host). The OS or like the host, calls to CPU, memory, storage and network resources. This hypervisor helps in running multiple different OS on versions of one OS on one desktop.

2) Discuss what changes may be necessary in future multi-core CPU platforms in order to better match the resource demands placed on these systems. For instance, can multithreading play an effective role in alleviating the competition for computing resources?
Well multithreading is an advantage, we all know. While designing, people/designers need to take in consideration to use multiprocessors and multi-core architectures. All processes need to run fast which is why multi threading is helpful as it helps execution of multiple threads by the CPU. They enhance performance and efficiency which is what is needed by all applications. One other thing to keep in mind would be building other parts of software and hardware, required by these multi-core processors to run smoothly and efficiently across the resources.

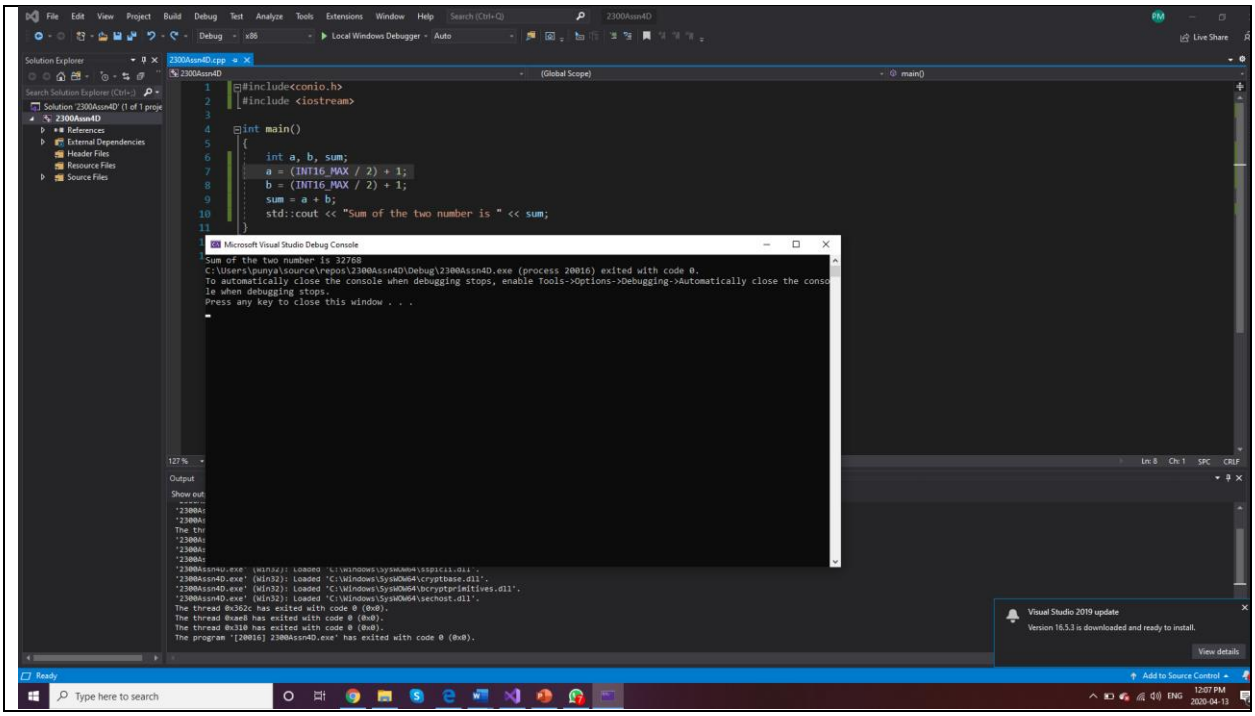
D. Programming (ish) Questions 6 marks total

1) Write a C program that causes overflow (just add two ints together which are (MAXINT/2) + 1 each, feel free to hard code this).
2) Output the assembly file and identify the chunk of code where the operation that causes overflow happens. Screenshot it or copy/paste it here.
(Realistically, you may not actually be able to find where the code is that does this, that's ok, but at least make an effort, explain your steps and show where you think this is happening. If you output assembly with the C++ code you should be able to search for that in the file at least).



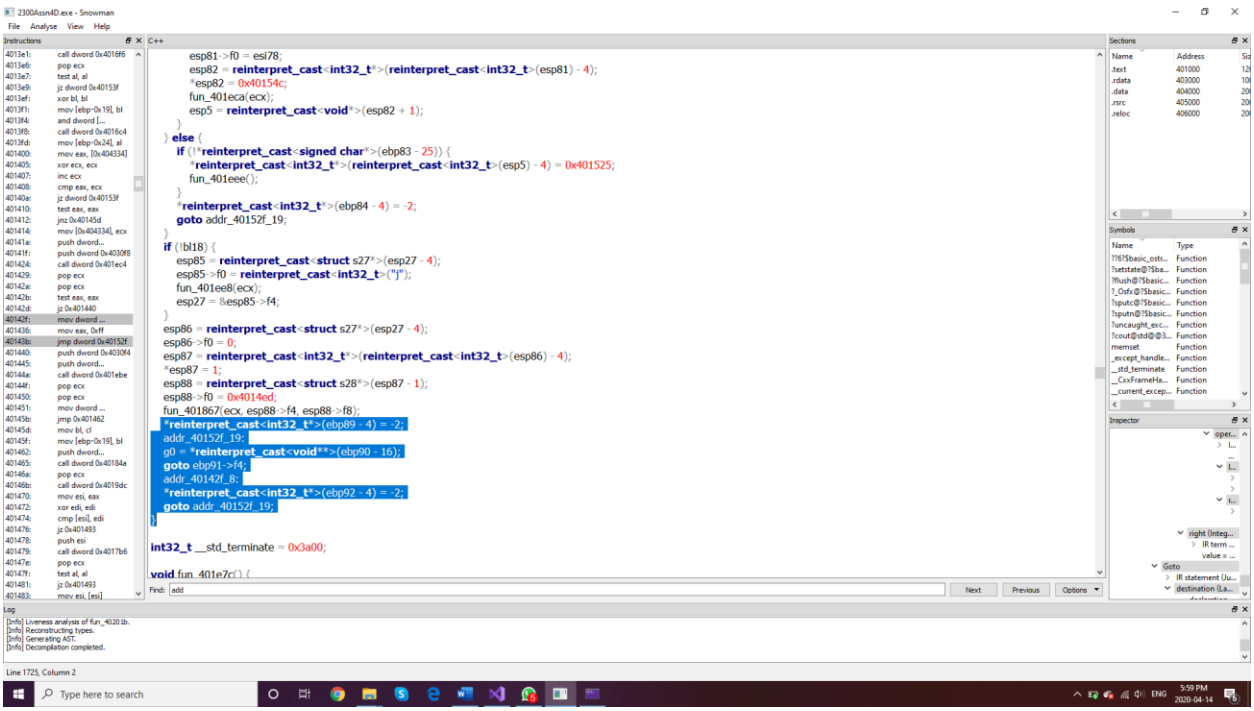
So on outputting the assembly file I was at the code line sum = a + b; however, the assembly file code at that point has nothing of the overflow, but only using of values of a and b and that sort of code. So, I assumed it will happen down, where it calls something maybe. And then on line 00F9263D it is calling sack failure. The highlighted line, I guess somewhere from that line is where the overflow code lines are. Probably because it says stack failure lol. Also however, on calling the stack failure there is a bracket with 0F911E5h line, maybe that is where the failure happened. But apparently there is no 0F911Eh, huh, so probably this line in the screenshot is where overflow occurs.

3) Run the program and screenshot the overflow output and paste it here.



4) Build that C program **in release mode**.
5) Then upload the executable here: https://retdec.com/decompilation/ as an input file (you don't need anything else).
6) Look at the assembly output and compare to the assembly output from your program. Can you even find where your program actually is? (if so list the line number if not tell me what you did to try and find it, realistically, you probably can't find it). My executable is 9Kb, and the output is about 5000 thousand lines long. If you do this in debug mode you'll get an output that's about 50 000 lines. **Submit the decompiled code as well (just copy/paste it into a file)**

   This is how you actually reverse engineer a program (such as a virus etc. Though to do that well you'd pay for tools that make some of it less terrible than it is with raw code, but those tools see what you just saw.).



So, it's been difficult finding the code lines, however, there is a function of add among these highlighted lines and I think this is where our code is probably.

7) Lastly, have a look at http://jd.benow.ca/ go here and download the jd-gui-1.4.0.jar, you'll get a security warning, but it's fine, run it anyway. Run the program and open the MARS jar file you've been using for other assignments. Show me a screenshot of the output here. (Yes, that reconstructed the entire source structure from the JAR file, including comments).