Exercise 4 Report/Write-Up

Jahnavi Pinnamaneni, Deeskshith Reddy Patil

April 14, 2022

ECEN 5623- Spring 2022

1 Problem 1

Therac - 25

BRIEF: The Therac-25 was a computer controlled radiation therapy machine produced by Atomic Energy of Canada Limited (AECL) in 1982. It was involved in at least six accidents between 1985 and 1987, in which patients were given massive overdoses of radiation. Because of concurrent programming errors (race conditions), it sometimes gave its patients radiation doses that were hundreds of times greater than normal, resulting in death or serious injury.

DESIGN: The machine has two modes of radiation therapy
1) Direct electron-beam therapy, in which a narrow, lowcurrent beam of high-energy (5 MeV to 25 MeV) electrons was scanned over the treatment area by magnets;
2) Megavolt X-ray (or photon) therapy, which delivered a fixed-width beam of X-rays, produced by colliding a narrow beam of 25 MeV electrons with a target, then passing the emitted X-rays through both a flattening filter and a collimator.
Previous models Therac-6 and Therac-20 had hardware interlocks to prevent such faults, but the Therac-25 had removed them, depending instead in software checks for safety.

CAUSES FOR FAILURE: 1) The source code used for the previous models was leveraged. The previous models had most of the software defects masked since they had hardware interlocks for safety.
2) The leveraged code was not thoroughly checked thus causing the bugs to be left unattended.
3) Furthermore, the AECL had not tested the Therac-25 with the combination of software and hardware until it was assembled at the hospital.
4) The errors merely displayed the word MALFUNCTION followed by a number in the range 1 to 64. The errors caused the system to be paused or suspended. Since the system was prone to frequent pauses, the pauses caused due to the serious errors were also overlooked.

EXACT ERROR: The problem was caused by faster keyboard input. During the initial years, the operators were not acquainted with the system and hence used to input the instructions via keyboard slowly. As the years passed, the technicians were able to input instructions at faster speeds. This caused the errors to surface.
When the keyboard input frequency increased it interfered with other tasks thus causing race conditions due to which some of the keyboard inputs were missed. This occurred due to lack of synchronization mechanisms between tasks.

PROPER DESIGN CONSIDERATIONS SUCH AS:
1) Synchronization methods (mutexes, semaphores etc);
2) Error handling
would have helped avoid the accidents and build a more robust system.

ARIANE 5

On June 1996, the maiden flight of the Ariane 5 launcher ended in a failure. Only about 40 seconds after initiation of the flight sequence at an altitude of about 3700m, the launcher veered off its flight path, broke up and exploded.

DESIGN: The Flight Control System of the Ariane 5 was of a standard design. The attitude of the launcher and its movements in space are measured by an Inertial Reference System(SRI). It has its own internal Computer in which angles and velocities are calculated on the basis of information from a strap-down inertial platform, with laser gyros and accelerometers. The data from the SRI are transmitted through the databus to the On-Board Computer, which executes the flight program and controls the nozzles of the solid boosters and the cryogenic engines. There was a standby SRI, and if the OBC detects that the active SRI has failed it immediately switches to the other one, provided the unit is functioning properly.

CAUSES FOR FAILURE: 1) Source code of the previous models was leveraged. But the problem occured since the Ariane 5 was designed to be much faster when compared to any of the previous models.
2) The SRI would do the calculations in 64 bit floating point numbers and then convert the result to a 16 bit integer while sending the data to the OBC.
3) This conversion lead to overflow error at very high speeds.
4) Proper Error-handling was not performed when the above fault occured.

EXACT CAUSE: The primary technical causes are the Operand Error when converting the horizontal bias variable BH, and the lack of protection of this conversion which caused the SRI computer to stop.
Not all the conversions where protected because a maximum workload target of 80 percent had been set for the SRI Computer. An analysis was performed to determine the vulnerability of unprotected code.The conversion of the floating point values to integers was analysed and operations involving seven variables were at risk of leading to an Operand Error. Three of these variables were left unprotected of which variable BH was one.
The decision to cease the processor operation proved fatal. Restart was not feasible since the attitude was too difficult to recalculate after a processor shutdown; therefore the Inertial Reference System became useless.

DESIGN DECISIONS SUCH AS:
Better error handling by allowing the SRI computers to continue providing the best estimations.
Better protection for converted data
would have helped avoid this failure.

BOEING 737 MAX

The Boeing 737 MAX passenger airliner was grounded worldwide between March 2019 and December 2020 after 346 people died in two crashes, Lion Air Flight 610 on October 29, 2018 and Ethiopian Airlines Flight 302 on March 10, 2019.

DESIGN: In order to compete with the market, Boeing decided to replace their existing engines to a newer model and this lead to a change in the position of the engine. The change in position of the engine caused the airplane's angle of attack to increase hence a software was included in the Flight Control Computer to compensate for this increase in inclination. This software was called the Maneuvering Characteristics Augmentation System (MCAS).

CAUSES FOR FAILURE: Preliminary investigations revealed serious flight control problems and signs of angle-of-attack sensor and other instrument failures. Boeing had not provided any information abou the MCAS to the customers and did not suggest simulation training for pilots using this new flight.

EXACT CAUSE: While in flight the MCAS was inteded to mimic pitching behaviour of the previous generation, correcting the elevated angle of attack by electronically adjusting the horizontal stabilizer and trim tab to push the nose down. The accidents occured when MCAS acted on false indications from the exterior sensors. The pilots could not override this software and thus the MCAS pushed the plane in nose down direction at very high speeds thus leading to fatal crash

More time for development and testing would have helped avoid this accident from happening.

Ranking: 1) Boeing 737 Max: This is in my opinion the worst failure since it costed so many human lives and millions of dollars.
2) Ariane 5: This failure could have been avoided by proper testing after leveraging the previous source code
3) Therac 25: This failure, though could have been avoided by proper design, since it was in the later 80's, resources might have been scarce.

2 Problem 2

2a) According to Henry Giffin, commander of the Atlantic Fleet's Naval Surface, "The Yorktown lost control of its propulsion system because its computers were unable to divide by the number zero, the memo said. The Yorktowns Standard Monitoring Control System administrator entered zero into the data field for the Remote DataBase Manager program. That caused the database to overflow and crash all LAN consoles and miniature remote terminal units, the memo said."

But According to Anthony DiGiorgio, a civilian engineer, the NT operating system was the source of the problem. Ron Redman, deputy technical director of the Fleet Introduction Division of the AegisProgram Executive Office, also confirmed that there have been numerous software failures associated with NT aboard the Yorktown.

2b) Other papers that I have gone through confirm Anthony DiGiogia's statement that Windows NT was the source of the issue.

"Sunk by Windows NT" http://www.wired.com/news/technology/0,1282,13987,00.html Contains advocacy paragraph: "Why Windows NT Server 4.0 continues to exist in the enterprise would be a topic appropriate for an investigative report in the field of psychology or marketing, not an article on information technology," said John Kirch, a networking consultant and Microsoft certified professional, in his white paper, Microsoft Windows NT Server 4.0 versus Unix. "Technically, Windows NT Server 4.0 is no match for any Unix operating system."

"When Smart Ships Divide By Zer0 — Stranding the USS Yorktown" https://medium.com/dataseries/when
smart-ships-divide-by-zer0-uss-yorktown-4e53837f75b2 ""using Windows NT... on a war-

ship is similar to hoping that luck will be in our favor.""

2c) Based on my understanding, I am of the opinion that the root cause does not involve the operator interface.

I stand by the opinion of DiGiorgio that, "If you understand computers, you know that a computer normally is immune to the character of the data it processes, he wrote in the June U.S. Naval Institutes Proceedings Magazine. Your 2.95 dollars calculator, for example, gives you a zero when you try to divide a number by zero, and does not stop executing the next set of instructions. It seems that the computers on the Yorktown were not designed to tolerate such a simple Failure."

Therefore I re-iterate that, in my opinion, Windows NT cannot be real-time applications and is the root cause of near fatal accidents involving this machine.

2d) Yes, I believe that upgrading the Aegis systems to RedHawk Linux would help reduce the operational anomalies and defects. The reasons are as follows,

RedHawk Linux comes with Real-Time Optimized Linux Kernel. "RedHawk is a complete Linux distribution designed to fully support time-critical applications. RedHawk provides a true single-kernel programming environment that directly controls all system operation. Complex time-critical applications often require that high-speed file I/O, networking and graphics be performed deterministically together with real-time task scheduling"

Frequency-Based Scheduler. "RedHawk's Frequency-Based Scheduler (FBS) is a high-resolution task scheduler that enables the user to run processes in cyclical execution patterns. FBS controls the periodic execution of multiple, coordinated processes utilizing major and minor cycles with overrun detection. A performance monitor is also provided to view CPU utilization during each scheduled execution frame."

Multithreading and Preemption
"To provide deterministic response, many critical sections of the kernel have been tuned and optimized to dramatically shorten non-preemptable conditions. These changes are key to allowing a high-priority process to respond immediately to an external event, even when the CPU is currently in use. Semaphores internal to RedHawk Linux also support priority inheritance to prevent priority inversion when multiple threads of an application are competing for operating system resources."

Due to the above mentioned reasons, I believe that RedHawk Linux is more suitable for Real-Time applications over Windows or Cyclic Executive.

3 Problem 3

3a) Team Members: Jahnavi Pinnamaneni and Deekshith Reddy Patil

The aim of the project is to perform a comparison between Linux kernel and Linux kernel with Preempt Real-Time patch on a Raspberry Pi 3B. By means of this project we want to understand the performance gain that a preempt RT patch provides as opposed to normal Linux OS in terms Worst Case Execution Time, Jitter, Interrupt Response Latency etc. Furthermore we intend to understand proper techniques for testing and profiling of code on different kernels.

The tasks that are used to test the kernels involve the following,
An image capture thread
A thread to perform Hough Line Transform on the captured image
A thread to perform Hough Circle Transform on the captured image
After executing the program on the above-mentioned kernels, we intend to contrast the performance differences in terms of execution time and deadline misses.

The Reference links are as follows:
i) https://www.mdpi.com/2079-9292/10/11/1331/pdf
ii)https://mdpi-res.com/d_attachment/computers/computers-10-00064/article_deploy/computers-10-0006
pdf
iii)https://www.icterra.com/real-time-linux-comparison/

3b)
Jahnavi Pinnamaneni
My responsibilities in this project are as follows:
Add preempt RT patch to Linux on Raspberry pi 3B and check if this works fine by executing a sample code.
Code the tasks Image Capture and Hough Elliptical transform and integrate them with the rest of the code.
Profile the code on Raspberry Pi with Preempt RT patch.
Combine all the results and draw final conclusions.

Deekshith Reddy Patil
My role in this project is to port OpenCV4 to Raspberrypi's Linux Kernel.
Complete the sequencer thread and Hough line transform thread in the program.
Test and obtain results on the raspberry pi running regular linux kernel.

4 Problem 4

Major functional capability requirements of the project are as follows:
a) Addition of preempt RT kernel to Raspberry Pi 3b.
b) Capturing a continuous stream from camera using opencv.
c) Performing Hough Line and Hough Circle transformation on the obtained image.
d) Profiling the above program on regular linux kernel and Preempt RT patch kernel.
e) Analyze the obtained profiling data to contrast between the two kernels and draw conclusions about the performance gain obtained.

5 Problem 5

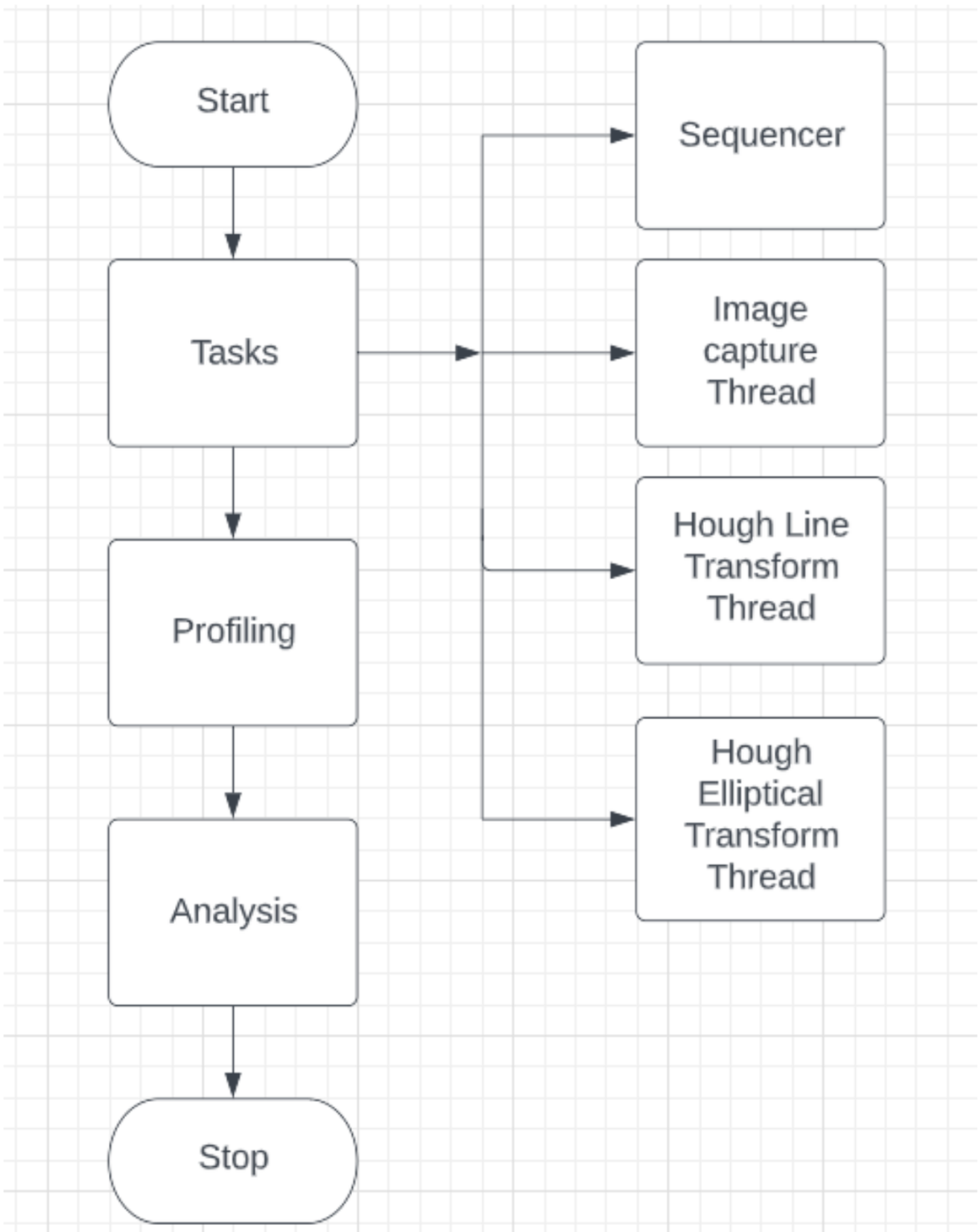

Figure 1: Block Diagram

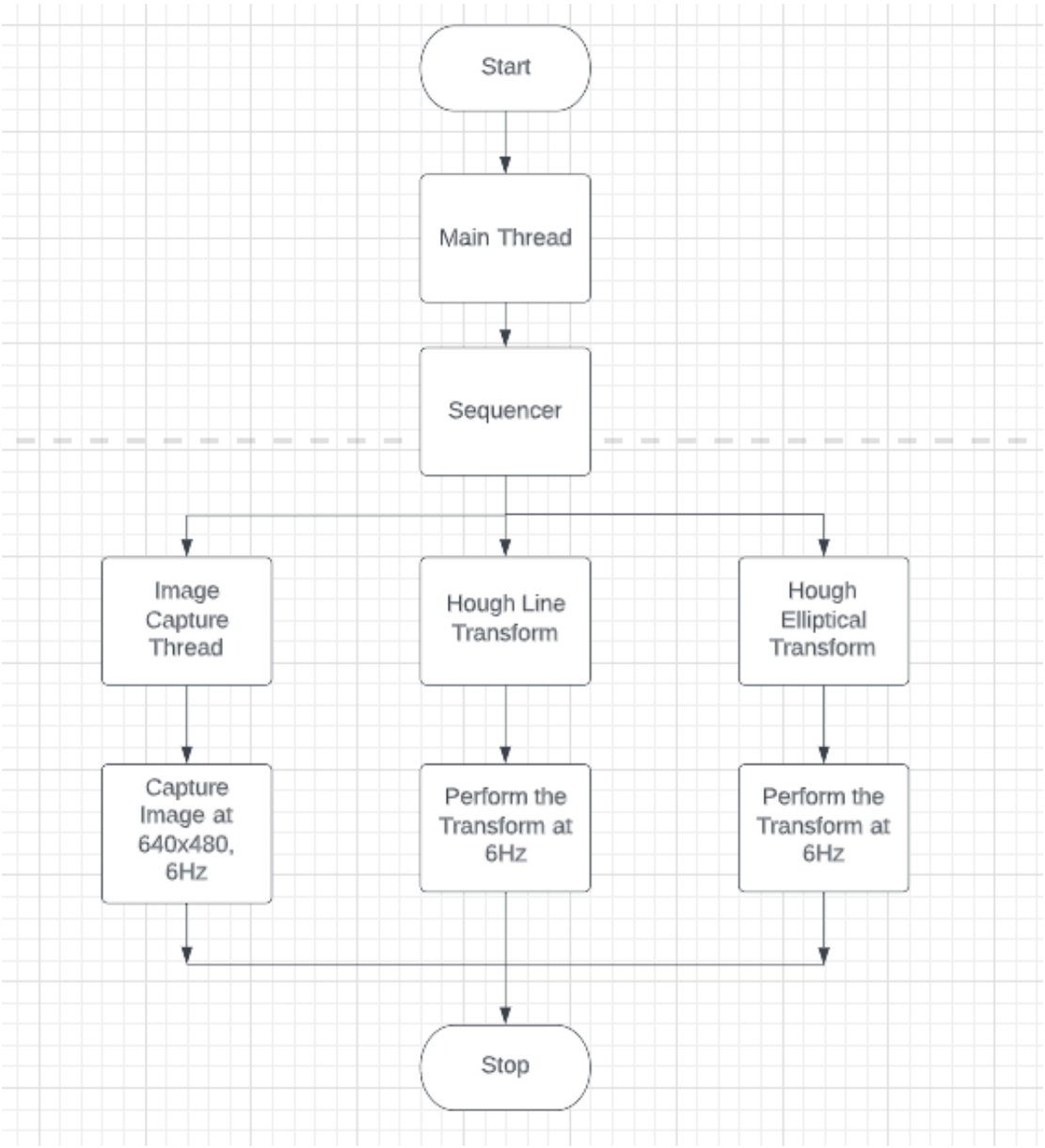

Figure 2: Working of the Main loop

Pseudo code:

Structure for sequencer:

```
while(1)
{
        • Increment counter
        • Check if the desired frequency has been reached
        • Give_semaphores to the tasks
}
```

Structure of each task:

```
while(1)
{
        • take_semaphore()
        • Perform the required function
}
```

Figure 3: Psuedo Code

6 Problem 6

Service	Ti (ms)	Ci (ms)	Di (ms)
Sequencer thread	1	TBD	NA
Image Acquisition thread	166	TBD	60
Hough Line Transform Thread	166	TBD	35
Hough Elliptical Transform Thread	166	TBD	35

Table 1: Ci, Ti, Di

***TBD: WCET is To Be Determined ***