National University of Computer and Emerging Sciences

FAST School of Computing **Spring-2024** **Islamabad Campus**

CS-2006: Operating Systems Sessional-II Exam

Date: 4rd April, 2024

Course Instructor(s)

Dr. M. Faisal Cheema, Dr. Adnan Tariq, Ms. Maryam Shahbaz, Ms. Rabail Zahid, Mr. M. Aadil-ur-Rehman

Total Time: 1 Hour

Total Marks: 45

Student Name	Roll No.	Course Section	Student Signature
_____	_____	_____	_____

Instructions:

1. Read the question carefully, understand the question, and then attempt your answers. **Attempt Question 1 on the Question Paper and Question 2,3,4 on the Answer Sheet.**
2. Verify that you have **Seven (7)** printed page of the question paper including this page. There are **Four (4)** questions.
3. Calculator sharing is strictly prohibited.

Question 1: "Note : Attempt it on Question Paper"

Below table contains information of different processes running on a uni-processor system**. [15 Marks]**

Process	Arrival Time	CPU burst time (ms)
A	1	12
B	2	8
C	3	6
D	4	4
E	8	5

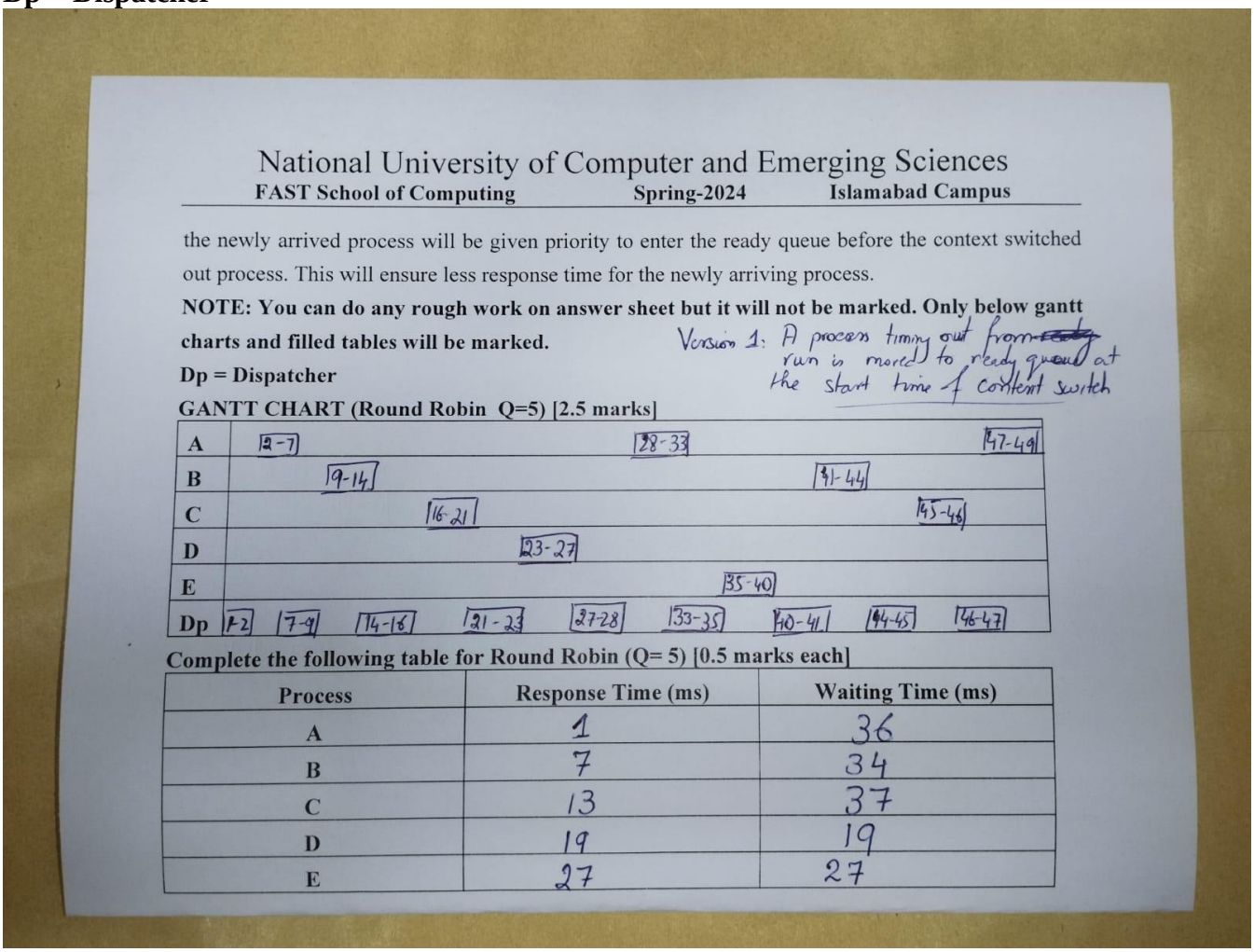
Schedule the above processes using **Round Robin (RR)** and **Shortest Remaining Time First (SRTF).** For Round robin the time quantum is **5 ms.** For Round Robin approach, you need to compute **response time and waiting time** for each process **and fill them in a table**. For Shortest Remaining time approach, you need to compute **waiting time and turnaround time** for each process **and fill them in a table**. You will be given marks for correct calculations of response time, turnaround time and waiting times. **Also, show your simulation by completing the Gantt chart**. **Without the Gantt chart you will be awarded 0 marks in the question, even if you write correct waiting time etc.**

IMPORTANT: (1) For both approaches, **you must add dispatch latency when dispatcher brings a process to running state**, as per the given below policy. Dispatch latency is the period of time the dispatcher needs to stop one process and start another running. If no process is holding the CPU and the newly arriving process in the ready queue is handed over the CPU **OR** some process exists after completion + another process is handed over the CPU the **total** dispatch latency is **1 ms**. But, if a process is timed out and context switch is performed, the **total** dispatch latency is **2 ms** in this case. **(2)** If a process is context switched out and a new process also tries to enter the ready queue at the same time,

the newly arrived process will be given priority to enter the ready queue before the context switched out process. This will ensure less response time for the newly arriving process.

NOTE: You can do any rough work on answer sheet but it will not be marked. Only below gantt charts and filled tables will be marked.

Dp = Dispatcher



National University of Computer and Emerging Sciences
FAST School of Computing Spring-2024 Islamabad Campus

the newly arrived process will be given priority to enter the ready queue before the context switched out process. This will ensure less response time for the newly arriving process.

NOTE: You can do any rough work on answer sheet but it will not be marked. Only below gantt charts and filled tables will be marked.

Version 1: A process timing out from ~~ready~~ run is moved to ready queue at the start time of context switch

Dp = Dispatcher

GANTT CHART (Round Robin Q=5) [2.5 marks]

A	2-7			28-33					47-49
B		9-14					41-44		
C			16-21					45-46	
D				23-27					
E						35-40			
Dp	1-2	7-9	14-16	21-23	27-28	33-35	40-41	44-45	46-47

Complete the following table for Round Robin (Q= 5) [0.5 marks each]

Process	Response Time (ms)	Waiting Time (ms)
A	1	36
B	7	34
C	13	37
D	19	19
E	27	27

National University of Computer and Emerging Sciences
FAST School of Computing **Spring-2024** **Islamabad Campus**

the newly arrived process will be given priority to enter the ready queue before the context switched out process. This will ensure less response time for the newly arriving process.

NOTE: You can do any rough work on answer sheet but it will not be marked. Only below gantt charts and filled tables will be marked.

Version 2: If a student assumed that a new process is given priority in ready list during half of context switch. —

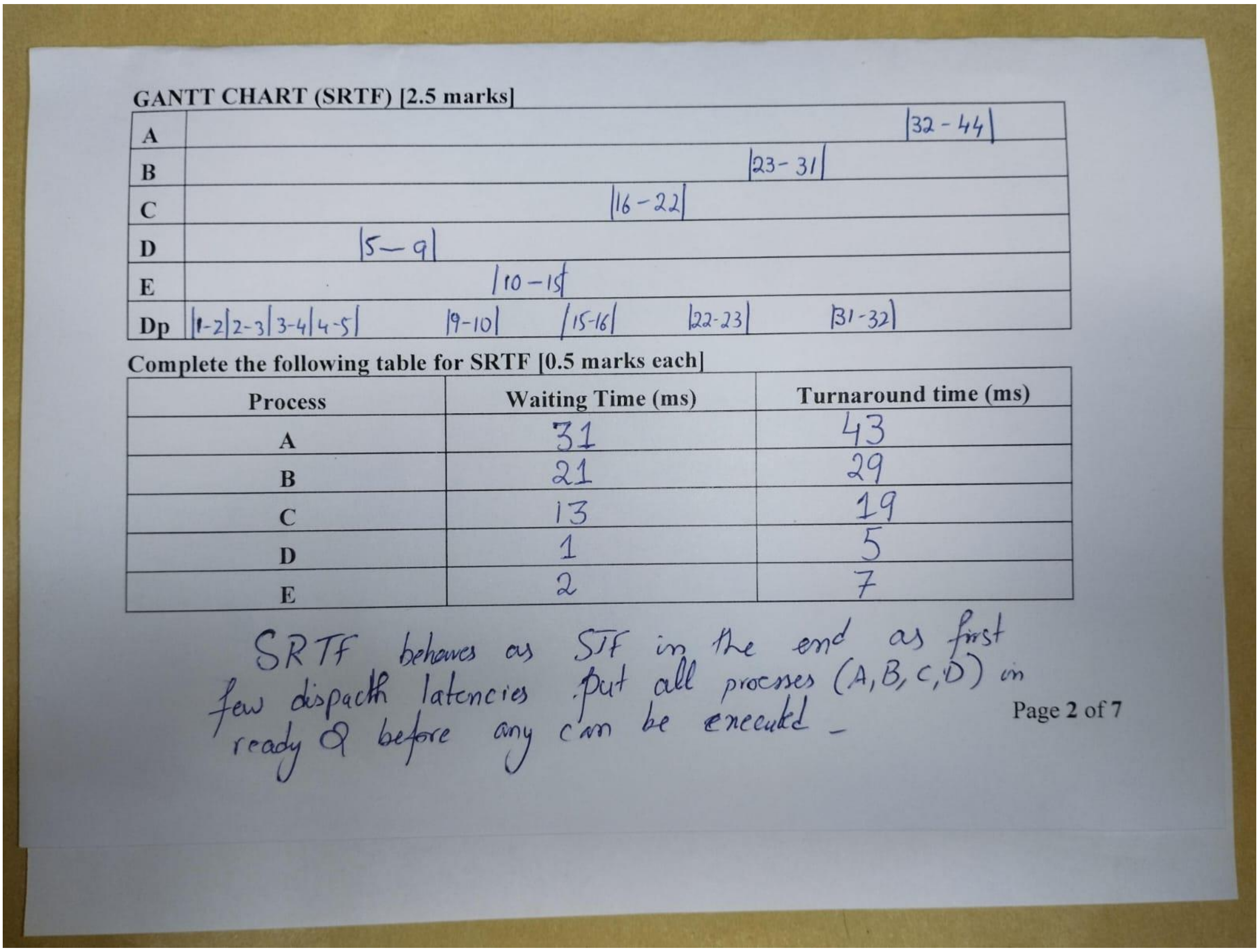
Dp = Dispatcher

GANTT CHART (Round Robin Q=5) [2.5 marks]

A	2-7	35-40	47-49						
B	9-14	41-44							
C	16-21	45-46							
D	23-27								
E	28-33								
Dp	1-2	7-9	14-16	21-23	27-28	33-35	40-41	44-45	46-47

Complete the following table for Round Robin (Q= 5) [0.5 marks each]

Process	Response Time (ms)	Waiting Time (ms)
A	1	36
B	7	34
C	13	37
D	19	19
E	20	20



GANTT CHART (SRTF) [2.5 marks]

A	|32 - 44|
B	|23 - 31|
C	|16 - 22|
D	|5 - 9|
E	|10 - 15|
Dp	|1-2|2-3|3-4|4-5| |9-10| |15-16| |22-23| |31-32|

Complete the following table for SRTF [0.5 marks each]

Process	Waiting Time (ms)	Turnaround time (ms)
A	31	43
B	21	29
C	13	19
D	1	5
E	2	7

SRTF behaves as SJF in the end as first few dispatch latencies put all processes (A, B, C, D) in ready Q before any can be executed.

Page 2 of 7

Question 2 : **Attempt on Answer Sheet. Please attempt the question parts together.**
We will write two programs for IPC using shared memory as an example. SERVER will create the shared segment, attach it, and then write some content in it. Then CLIENT will get itself to the shared segment and read the value written by SERVER. Make sure to use best coding practices, e.g. deleting shared memory after using it**.[8 Marks]**

```
//SERVER (Assume all libraries are included)
int main() {
   int shm_id;
      if((a.int shmid=shmget(2345 , 1024 ,  0666)< 0)
 {
      printf("Error in executing above statement");
      exit(1); }
   char *shm_ptr;
   if((b.  shm_ptr=shmat(shmid,NULL,0)< 0) {
      printf("Error in executing above statement");
```

```
      exit(1); }
    char buff[100];
 buff="HY ! CLIENT";
 c. strncpy(shm_ptr,buff,sizeof(buff));
  printf("Message Written.\n");
 d. shmdt(shm_ptr);
  return 0;}
```

```
//CLIENT (Assume all libraries are included)
int main() {
   int shm_id;
   if ((e.  shm_id=shmget(2345, 1024,  0666)< 0) {
      printf("Error in executing above statement");
      exit(1); }
   char *shm_ptr;
   if (f. shm_ptr=shmat(shmid,NULL,0)) < 0) {
     printf("Error in executing above statement");
     exit(1);   }
   printf("Reading Message %s\n", shm_ptr);
g. shmdt(shm_ptr);
 hshmctl(shmid, IPC_RMID, NULL);
 return 0;}
```

a. Creates shared memory segment with key 2345, having size 1024 bytes
b. Attach shared memory segment
c. Write a message to shared memory
d. Detach shared memory segment
e. Get the shared memory segment
f. Attach shared memory segment
g. Detach shared memory segment
h. Delete shared memory segment

Question 3: Attempt it on Answer Sheet. Please attempt the question parts together.

Consider a following command line statement:

a.out < numbers.txt | sort -n > result.txt

You have to implement the above shell statement in C language using **fork(), pipe(), dup2(), exec(), fopen()/open()** only. You are **NOT** allowed to directly read/write from/to the files numbers.txt and result.txt. Hint: Use STDIN and STDOUT for file reading/writing. Consider "sort" as a binary program. Make sure to use best coding practices, such as no zombie processes and no unnecessary open file descriptors etc. [12 Marks]

```
// Assume all libraries are included
int main() {
A [2 Marks]
int pipefd[2];
pid_t pid; // Create pipe
pipe(pipefd); // Fork process
pid = fork();

if (pid == 0) {// Child process
B [5 Marks]
close(pipefd[1]); // Close write end of pipe
dup2(pipefd[0], STDIN_FILENO);
close(pipefd[0]); // Close original read end of pipe
int fd = open("result.txt", O_WRONLY | O_CREAT | O_TRUNC, 0666);
dup2(fd, STDOUT_FILENO);
execlp("sort", "sort", "-n", NULL);
perror("execlp");
exit(EXIT_FAILURE);

}
else {
// Parent process
C [5 Marks]
close(pipefd[0]); // Close read end of pipe
dup2(pipefd[1], STDOUT_FILENO);
close(pipefd[1]); // Close original write end of pipe
int fd = open("numbers.txt", O_WRONLY | O_CREAT | O_TRUNC, 0666);
dup2(fd, STDOUT_FILENO); execlp("./a.out", "./a.out",  NULL);
} return 0;}
```

Question 4 : **Attempt on Answer Sheet. Please attempt the question parts together.**

This exam question is specifically tailored to build upon the knowledge and skills you've developed through Assignment 1, focusing on Linux shell scripting and Docker. It's crafted to evaluate your practical understanding and application of these essential tools in real-world scenarios.

Part A: Using the snapshots of **/proc/cpuinfo** provided below, write a shell script that extracts the following information:

1. Total Number of Processors,
2. Total Number of Physical Cores,
3. whether Hyperthreading is Enabled (Yes/No),
4. Total Number of Cores with Hyperthreading,
5. Processor Model.
6. Store information generated in part **1 and 2 (# of Processors, # of Physical Cores)** in a file named CPU_info.txt.

The code of this script is provided below with some missing parts. Your task is to provide the missing code statements. Comments are provided with each missing part for your guidance [5 marks]

```
processor       : 0         # the cpu identifier used for cpusets
vendor_id       : GenuineIntel
cpu family      : 6
model           : 79
model name      : Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz
stepping        : 1
microcode       : 0xb00002a
cpu MHz         : 2109.843
cache size      : 20480 KB
physical id     : 0         # the physical socket number
siblings        : 16
core id         : 0         # the cpu core number
cpu cores       : 8
apicid          : 0
initial apicid  : 0
fpu             : yes
fpu_exception   : yes
cpuid level     : 20
wp              : yes
flags           : ht        # lots of flags, ht means hyperthreading
bugs            :
bogomips        : 4190.23
clflush size    : 64
cache_alignment : 64
address sizes   : 46 bits physical, 48 bits virtual
```

```
root@ubuntu:~# cat /proc/cpuinfo
processor       : 0
vendor_id       : AuthenticAMD
cpu family      : 20
model           : 2
model name      : AMD E-450 APU with Radeon(tm) HD Graphics
stepping        : 0
microcode       : 0x5000119
cpu MHz         : 825.000
cache size      : 512 KB
physical id     : 0
siblings        : 2
core id         : 0
cpu cores       : 2
apicid          : 0
initial apicid  : 0
fpu             : yes
fpu_exception   : yes
cpuid level     : 6
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic se
t fxsr_opt pdpe1gb rdtscp lm constant_tsc rep_good nopl nons
gacy svm extapic cr8_legacy abm sse4a misalignsse 3dnowprefe
filter
bugs            : fxsave_leak sysret_ss_attrs null_seg spect
bogomips        : 3291.84
TLB size        : 1024 4K pages
clflush size    : 64
cache_alignment : 64
address sizes   : 36 bits physical, 48 bits virtual
power management: ts ttp tm stc 100mhzsteps hwpstate

processor       : 1
vendor_id       : AuthenticAMD
cpu family      : 20
model           : 2
model name      : AMD E-450 APU with Radeon(tm) HD Graphics
stepping        : 0
```

Provided Code (Fill in the Missing Lines) :

```
# Shell Script Starting Line
1. _#!/bin/bash__________                                   # [0.5 marks]

# Total Number of Processors
2. total_processors=$(__grep -c_____________________ "^processor"
/proc/cpuinfo)
OR
total_processors=$(awk '/^processor/ {count++} END {print count}'
/proc/cpuinfo)                                              # [0.5 marks]

# Total Number of Physical Cores
3. physical_cores=$(__grep "^cpu cores"_/proc/cpuinfo | uniq | _awk
{print $4}_________)
OR
physical_cores=$(awk '/^cpu cores/ {print $4; exit}'
/proc/cpuinfo)                                              # [0.5 marks]
```

```
# Is Hyperthreading Enabled
4. siblings=$( grep "^siblings" /proc/cpuinfo | uniq | awk '{print
$3}')
OR
siblings=$(awk '/^siblings/ {print $4; exit}' /proc/cpuinfo)

 # [0.5 marks]
5. if [ "$siblings" -gt "$physical cores" ]; then # [0.5 marks]
6.   hyperthreading="Yes"
7. else
8.   hyperthreading="No"
9. fi

# Processor Model
10. model=$( grep -m 1 "model name" /proc/cpuinfo | cut -d ':' -f2 | sed
's/^ //')
OR
model=$(awk -F ': ' '/model name/ {print $2; exit}' /proc/cpuinfo)
OR
model=$(grep -m 1 "model name" /proc/cpuinfo | head -n 1 | awk
'{sub(/^model name[[:space:]]*: /,""); print}')



                                    # [0.5 marks]

# Concatenate all information into a single string
11. output="Total Number of Processors: $total_processors\n"
12. output+="Total Number of Physical Cores:
$physical_cores\n"                                    # [0.5 marks]

# Output the string to CPU_info.txt
13. echo -e "$output" > CPU_info.txt              # [0.5 marks]

# Append random content to the file
14. echo "Random Content:" >> CPU_info.txt
15. cat /dev/urandom | head -c <any number of lines> >>
CPU_info.txt           # [1 marks]
```

Note: Please write the missing statements on the provided answer sheet in the same sequence, by mentioning the line number.

Part B: You are required to create a Dockerfile for running the shell scripts developed for the assignment. The Dockerfile should set up an environment with all necessary dependencies and execute the bash as default command to open a channel for communication with the container.

Code of the Docker file is provided below with some missing parts. Your task is to provide the missing statements. Comments are provided with each missing part for your guidance [3 marks]

Provided Dockerfile (Complete the Missing Parts):

```
# Load latest ubuntu as base image
```

```
1. FROM ubuntu:latest                                          # [0.5 marks]

# Install necessary packages
2. RUN apt-get update && apt-get install _-y gcc_              # [1 marks]

# Set working directory
3. WORKDIR /app

# Copy all scripts into the working directory with a single line
4. COPY _*.sh /app/__                                          # [0.5 marks]

# Make all scripts executable
5. RUN _chmod +x /app/*.sh                                     # [0.5 marks]

# Set the default command to execute bash
6. _CMD ["/bin/bash"]_____________                             # [0.5 marks]
```

Note: Please write the missing statements on the provided answer sheet in the same sequence, by mentioning the line number.

Part C: Remember the submission guidelines of the assignment: [2 Marks]

- Create your account on DockerHub using the official university email account and the username must be your registration number (i.e., 22i1234) number.
- Create your repository with name “Assignment_01_Section_X_SP2024_OS”
- Publish the image into the repository on DockerHub.

Write the necessary commands to perform the following operations:

1. Build the Docker image locally with a custom tag.
   _docker build -t <any custom tag> <location O>_____ [1 marks]

2. Publish the image to DockerHub with a custom tag.
   _docker push <reg #>/Assignment_01_Section_X_SP2024_OS:<tag>___ [1 marks]