THE HONG KONG POLYTECHNIC UNIVERSITY

DEPARTMENT OF COMPUTING

SEMESTER 2023 / 2024

PROGRAMME : 2011, 2012, 232, 241

YEAR : 2023/2024

SUBJECT CODE : COMP2432

SUBJECT TITLE : OPERATING SYSTEMS

TIME ALLOWED : 2 HOURS

THIS QUESTION PAPER HAS 6 PAGES (ATTACHMENTS INCLUDED)

INSTRUCTIONS TO CANDIDATES:

1. THIS MAKEUP MID-TERM TEST CONSTITUTES 20% OF THE GRADE.
   ANSWER ALL QUESTIONS. THE TIME ALLOWED IS 120 MINUTES.
2. ANSWER ALL QUESTIONS NEATLY AND CLEARLY IN THE ANSWER BOOKLET
3. OVERWRITTEN AND SCRIBBLED ANSWERS WILL NOT BE CONSIDERED FOR EVALUATION
4. ANY ELECTRONIC DEVICES OR CALCULATORS ARE NOT PERMITTED TO BE USED DURING THE EXAMINATION
5. ONLY 1 PAGE (2 SIDES) A4 SIZE CHEAT SHEET IS PERMITTED.
6. THE CHEAT SHEET SHOULD HAVE STUDENT LAST NAME, FIRST NAME AND STUDENT ID CLEARLY WRITTEN ON THE TOP OF THE CHEAT SHEET.

NOTES TO GENERAL OFFICE:
(E.G. STATIONERY AND NO. OF ANSWER BOOKS REQUIRED)

Question 1: **[10 Marks]**

(a) The time taken to switch between user and kernel modes of execution be t_1 while the time taken to switch between two processes be t_2. Explain in detail which of the following option is TRUE and why?

(i) $t_1 > t_2$ (ii) $t_1 = t_2$ (iii) $t_1 < t_2$
(iv) Nothing can be said about the relation between t_1 and t_2

(b) What does each of following commands does

```
$ grep -h -i '^comp2432' *.txt
(i)_____
$ grep -i comp2432 *.txt
(i)_____
$ grep -l -i '^comp2432' *.txt
(iii)_____
```

(c) Write the output of the following commands

```
$ result=`ps`
$ echo $result  # result is a list of words
PID TTY TIME CMD 8379 pts/1 00:00:00 bash 23142 pts/1 00:00:00 ps
$ result2=$(ps)
$ echo $result2
(i)_____
$ echo files are `ls`
files are first.sh lab1A.c lab1B.c
$ p="$PWD `whoami`- "
$ echo $p
/home/12345678d 12345678d-
$ p2="($PWD) $(whoami)- "
$ echo $p2
(ii)_____
$ p3="$(PWD) $(whoami)- "
(iii)_____
$ echo 'I need $500.00 now!!'
(iv)_____
$ echo "I need $5.00"
I need .00
$ echo "I need $500.00 now!!"
(v)_____
```

Question 2: **[10 Marks]**

Assume you are an OS guru, an industry seeks your advice as which kernel among Monolithic and Microkernal OS should be used for their new development product.

(a) Draw a clear diagram showing all the contents of user space and kernel space

(b) State your advisory reasons by differentiation between these two kernels

Question 3: **[10 Marks]**

Consider a system with two processes P and Q, running a Unix-like operating system as studied in class. Consider the following events that may happen when the OS is concurrently executing P and Q, while also handling interrupts.

(A) The CPU program counter moves from pointing to kernel code in the kernel mode of process P to kernel code in the kernel mode of process Q.

(B) The CPU stack pointer moves from the kernel stack of P to the kernel stack of Q.

(C) The CPU executing process P moves from user mode of P to kernel mode of P.

(D) The CPU executing process P moves from kernel mode of P to user mode of P.

(E) The CPU executing process Q moves from the kernel mode of Q to the user mode of Q.

(F) The interrupt handling code of the OS is invoked.

(G) The OS scheduler code is invoked.

For each of the two scenarios below, list out the chronological order in which the events above occur. Note that all events need not occur in each question.

(a) A timer interrupt occurs when P is executing. After processing the interrupt, the OS scheduler decides to return to process P.

(b) A timer interrupt occurs when P is executing. After processing the interrupt, the OS scheduler decides to context switch to process Q, and the system ends up in the user mode of Q.

Question 4: **[10 Marks]**

(a) What is the use of Translation Look-aside Buffer (TLB) in paging?

(b) Explain the concept of TLB hit and TLB miss with two neat diagrams and

(c) Derive the translation time.

Question 5: **[10 Marks]**

Base	Limit
219	600
2300	14
90	100
1327	580
1952	96

Consider the following Segment Table and answer the following:

(a) What are the physical addresses corresponding to the following two logical addresses: 1000, 2400

(b) What is the logical address for the physical address: 1375

Question 6: **[10 Marks]**

In this question, we try to understand the issues that surround the choice of page size in a system. In particular, we will discuss some new systems that have support for two pages sizes, one "small" sized (say 4 KB), and one that is "big" (say 1 MB)

(a) If we just have a single page size, and it is quite big, what are the possible negative consequences? In contrast, what negative consequences arise if our page size is too small

(b) Using big pages can improve performance. Which hardware resources of the system are better utilized with big pages?

(c) To support multiple page sizes, some aspects of the page table must change. Describe the most important changes needed, and how you would implement them. Assume a simple linear page table.

Question 7: **[10 Marks]**

Consider a system with a single CPU core and three processes A, B, C. Process A arrives at t = 0, and runs on the CPU for 10 time units before it finishes. Process B arrives at t = 6, and requires an initial CPU time of 3 units, after which it blocks to perform I/O for 3 time units. After returning from I/O wait, it executes for a further 5 units before terminating. Process C arrives at t = 8, and runs for 2 units of time on the CPU before terminating. For each of the scheduling policies below, calculate the time of completion of each of the three processes. Recall that only the size of the current CPU burst (excluding the time spent for waiting on I/O) is considered as the "job size" in these schedulers.

(i) First Come First Serve (non-preemptive).

(ii) Shortest Job First (non-preemptive)

(iii) Shortest Remaining Time First (preemptive)

Question 8: **[10 Marks]**

Refer the given named pipe program fragment and answer the following questions

(a) What programs does p1.c, p2.c and p3.c are referring to

(b) Write the command on Apollo to successfully execute this named pipe program fragment

(c) Write the output of (b)

```
//Program1: p1.c
#include<stdio.h>
#include<sys/types.h>
#include<sys/stat.h>
int main()
    {
        int res;
        res = mkfifo("fifo1",0777);
        printf("named pipe created\n");
    }
```

```
//Program2: p2.c
#include<unistd.h>
    #include<stdio.h>
    #include<fcntl.h>
    int main()
    {
            int res,n;
            res=open("fifo1",O_WRONLY);
            write(res,"Message",7);
            printf("Sender Process %d sent the data\n",getpid());
    }
```

```
//Program 3: p3.c
    #include<unistd.h>
    #include<stdio.h>
    #include<fcntl.h>
    int main()
    {
            int res,n;
            char buffer[100];
            res=open("fifo1",O_RDONLY);
            n=read(res,buffer,100);
            printf("Reader process %d started\n",getpid());
            printf("Data received by receiver %d is: %s\n",getpid(), buffer);
    }
```

Question 9: **[10 Marks]**

(a) Consider a process P1 that forks P2, P2 forks P3, and P3 forks P4. P1 and P2 continue to execute while P3 terminates. Now, when P4 terminates, which process will take P4?

(b) Given the following program fragment: How many different copies of the variable c are there? What are their values? State clearly which process (parent or child or grandchild) execute first and prints what value?

```
main(int argc, char ** argv)
{
    int child = fork();
    int c = 5;

    if (child == 0)
    {
        c += 5;
    }
    else
    {
        child = fork();
        c += 10;

        if (child)
            c += 5;
    }
}
```

Question 10: **[10 Marks]**

(a) Consider a paging hardware with a TLB. Assume that the entire page table and all the pages are in the physical memory. It takes 10 milliseconds to search the TLB and 80 milliseconds to access the physical memory. If the TLB hit ratio is 0.6, calculate the effective memory access time (in milliseconds).

(b) The percentage of times a page number is found in the TLB is known as _____

(c) The number of pages that TLB contains in is equal to the page table. Justify this statement.

*** End ***

Question 1: **[10 Marks]**

(a) The time taken to switch between user and kernel modes of execution be t_1 while the time taken to switch between two processes be t_2. Explain in detail which of the following option is TRUE and why?

(i) $t_1 > t_2$ (ii) $t_1 = t_2$ (iii) $t_1 < t_2$
(iv) Nothing can be said about the relation between t_1 and t_2

Ans – (iii): Process switches or Context switches can occur in only kernel mode. So, for process switches first we have to move from user to kernel mode. Then we have to save the PCB of the process from which we are taking off CPU and then we have to load PCB of the required process. At switching from kernel to user mode is done. But switching from user to kernel mode is a very fast operation (OS has to just change single bit at hardware level). Thus T1< T2

(b) What does each of following commands does

```
$ grep -h -i '^comp2432' *.txt
```
(i) Print only the lines beginning with “comp2432”.
```
$ grep -i comp2432 *.txt
```
(ii) Find text files and lines containing comp2432 (ignore upper/lower case).

```
$ grep -l -i '^comp2432' *.txt
```
(iii) Print only the names of files with lines beginning with “comp243”.

(c) Write the output of the following commands

```
$ result=`ps`
$ echo $result  # result is a list of words
PID TTY TIME CMD 8379 pts/1 00:00:00 bash 23142 pts/1 00:00:00 ps
$ result2=$(ps)
$ echo $result2
```
(i) PID TTY TIME CMD 8384 pts/1 00:00:00 bash 23142 pts/1 00:00:00 ps
```
$ echo files are `ls`
files are first.sh lab1A.c lab1B.c
$ p="$PWD `whoami`- "
$ echo $p
/home/12345678d 12345678d-
$ p2="($PWD) $(whoami)- "
$ echo $p2
```
(ii) (/home/12345678d) 12345678d-
```
$ p3="$(PWD) $(whoami)- "
```
(iii) bash: PWD: command not found... Similar command is ’pwd’
```
$ echo 'I need $500.00 now!!'
```
(iv)I need $500.00 now!!
```
$ echo "I need $5.00"
I need .00
$ echo "I need $500.00 now!!"
```
(v) echo "I need $500.00 nowecho "I need $5.00"" I need 00.00 nowecho I need .00

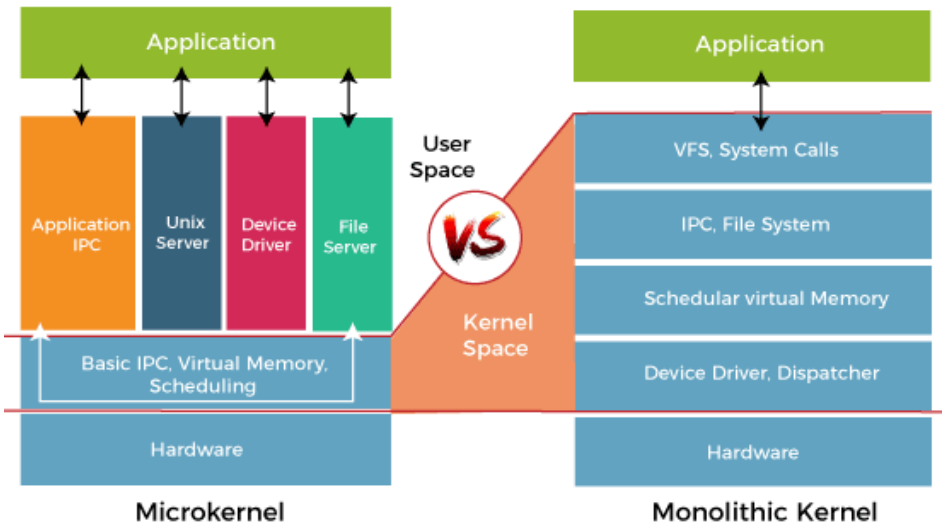
cont’d…

Question 2: **[10 Marks]**

Assume you are an OS guru, an industry seeks your advice as which kernel among Monolithic and Microkernal OS should be used for their new development product.

(a) Draw a clear diagram showing all the contents of user space and kernel space
(b) State your advisory reasons by differentiation between these two kernels

MICROKERNEL KERNEL	MONOLITHIC KERNEL
A kernel type that provides mechanisms such as low-level address space management, thread management and interprocess communication to implement an operating system	A type of kernel in operating systems where the entire operating system works in the kernel space
OS services and kernel are separated	Kernel contains the OS services
Slow	Fast
Failure in one component will not affect the other components	Failure in one component will affect the entire system
Easier to add new functionalities	Difficult to add new functionalities
Smaller in size	Larger in size



Question 3: **[10 Marks]**

Consider a system with two processes P and Q, running a Unix-like operating system as studied in class. Consider the following events that may happen when the OS is concurrently executing P and Q, while also handling interrupts.

(A) The CPU program counter moves from pointing to kernel code in the kernel mode of process P to kernel code in the kernel mode of process Q.

(B) The CPU stack pointer moves from the kernel stack of P to the kernel stack of Q.

(C) The CPU executing process P moves from user mode of P to kernel mode of P.

(D) The CPU executing process P moves from kernel mode of P to user mode of P.

(E) The CPU executing process Q moves from the kernel mode of Q to the user mode of Q.

(F) The interrupt handling code of the OS is invoked.

(G) The OS scheduler code is invoked.

For each of the two scenarios below, list out the chronological order in which the events above occur. Note that all events need not occur in each question.

(a) A timer interrupt occurs when P is executing. After processing the interrupt, the OS scheduler decides to return to process P.

(b) A timer interrupt occurs when P is executing. After processing the interrupt, the OS scheduler decides to context switch to process Q, and the system ends up in the user mode of Q.

Ans: (a) C F G D (b) C F G B A E

cont'd…

Question 4: **[10 Marks]**

(a) What is the use of Translation Look-aside Buffer (TLB) in paging? Explain the concept of TLB hit and TLB miss with two neat diagrams and state the translation time.

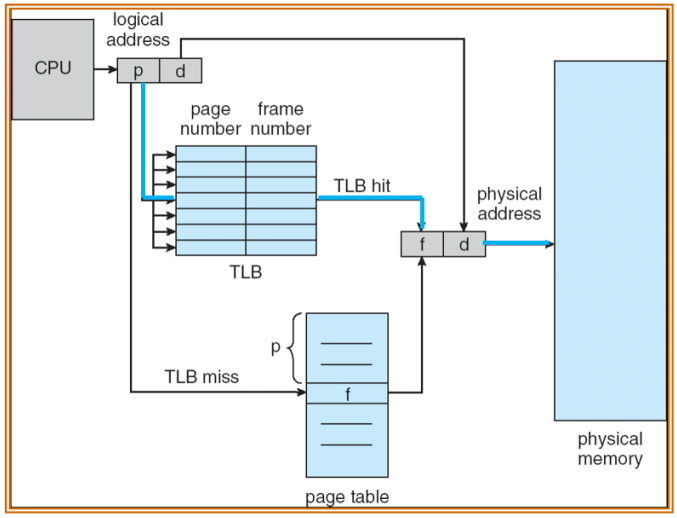
(b) Consider a paging hardware with a TLB. Assume that the entire page table and all the pages are in the physical memory. It takes 10 milliseconds to search the TLB and 80 milliseconds to access the physical memory. If the TLB hit ratio is 0.6, calculate the effective memory access time (in milliseconds).

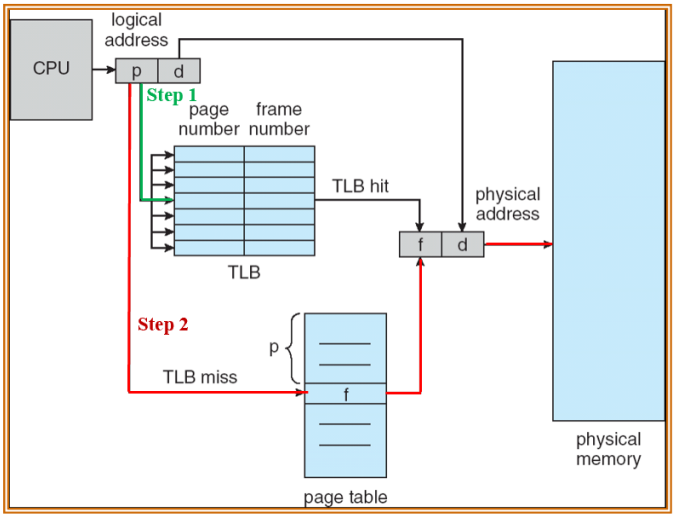
Ans: (a) Every data/instruction access requires two memory accesses: one for page table and one for data / instruction. This extra memory access to page table is expensive. This is solved by a hardware cache called Translation Look-aside Buffer (TLB) to store recently translated logical page numbers.

The processor examines the TLB if a page table entry is present (TLB hit), the frame number is retrieved and the real address is formed. If a page table entry is not found in the TLB (TLB miss), the page number is used as an index while processing the page table.

*Translation time = h (hit ratio) * cache access time + (1 – h) * (cache access time + memory access time) = cache access time + (1 – h) x memory access time.*

Effective access time = translation time + memory access time = cache access time + (2 – h) x memory access time.





*Answer: (b) EAT := TLB_miss_time * (1- hit_ratio) + TLB_hit_time * hit_ratio.*

*EAT := (TLB_search_time + 2*memory_access_time) *
(1- hit_ratio) + (TLB_search_time + memory_access_time)* hit_ratio.*

As both page table and page are in physical memory
*T(eff) = hit ratio * (TLB access time + Main memory access time) +*
*(1 – hit ratio) * (TLB access time + 2 * main memory time)*
= 0.6(10+80) + (1-0.6)*(10+2*80)*
*= 0.6 * (90) + 0.4 * (170)*
= 122 milliseconds

cont'd…

Question 5: **[10 Marks]**

Base	Limit
219	600
2300	14
90	100
1327	580
1952	96

Consider the following Segment Table and answer the following:

(a) What are the physical addresses corresponding to the following two logical addresses: 1000, 2400

(b) What is the logical address for the physical address: 1375

The logical address space of the process is drawn below according to the segment sizes of the 5 segments given in the segment table.

(a) So, logical address1000 is in Seg. No. 3
Offset of the address in its segment = 1000 – 714 (base logical addr. Of Seg. 3) = 286

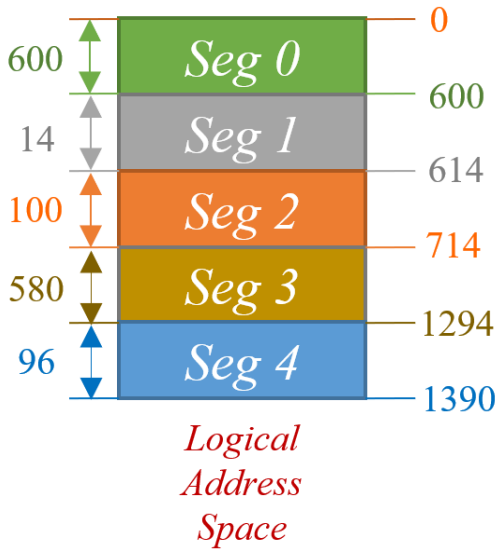
Hence, the corresponding physical address = 1327 (base physical addr. Of Seg. 3) + 286 = 1613

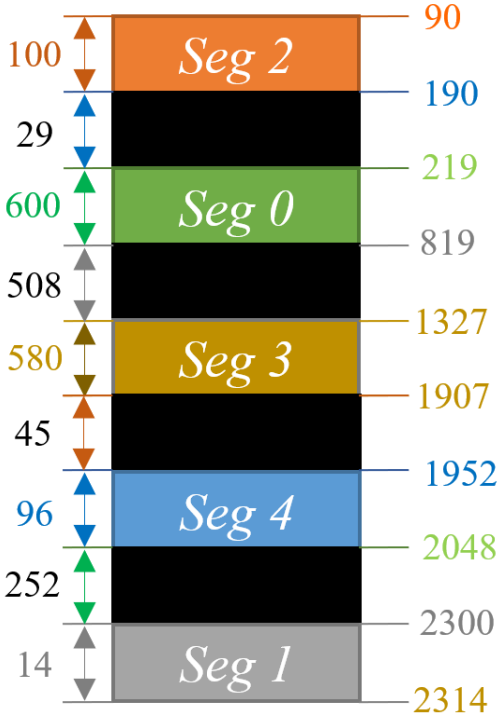
2400 is beyond the logical address space of the process. Hence, it will generate a TRAP: Addressing Error.

(b) Physical address 1375 belongs to the space allocated to Seg. 3 in main memory.
[Because physical address space for Seg. 3 is 1327 to (1327+580) = 1907.
// See the Segment table give in the question paper.]

So, offset of this physical address = 1375 – 1327 (base physical addr. Of Seg. 3) = 48

Hence, the corresponding logical address = 714 (base logical addr. Of Seg. 3) + 48 = 762





cont'd…

Question 6: **[10 Marks]**

In this question, we try to understand the issues that surround the choice of page size in a system. In particular, we will discuss some new systems that have support for two pages sizes, one "small" sized (say 4 KB), and one that is "big" (say 1 MB)

(a) If we just have a single page size, and it is quite big, what are the possible negative consequences? In contrast, what negative consequences arise if our page size is too small

(b) Using big pages can improve performance. Which hardware resources of the system are better utilized with big pages?

(c) To support multiple page sizes, some aspects of the page table must change. Describe the most important changes needed, and how you would implement them. Assume a simple linear page table.

Answer – (a)
Bigger pages have the problem of increased waste due to internal fragmentation.
Smaller pages have the problem of increasingly large page tables.

Answer – (b)
Big pages are primarily used to reduce pressure on the TLB – with big pages, a much bigger portion of your address space can be actively mapped by the TLB, and hence your TLB hit rates improve. If you didn't say "TLB" in your answer, it was hard for you to get full credit.

Big pages may also reduce memory pressure because you have smaller page tables and hence more other stuff can fit into memory. Finally, big pages may improve disk access times, as it is generally more efficient to move large pages in and out of disk.

Answer – (c)
To support multiple page sizes, some aspects of the page table must change. Describe the most important changes needed, and how you would implement them. Assume a simple linear page table.

Possibly - different page sizes implied a differing number of bits needed in the Physical Page Number (PPN) in each page table. However, it's a little trickier than that, because you also have to index the page table with the VPN of the address, so a more complete answer would have to deal with that issue as well.

cont'd…

Question 7: **[10 Marks]**

Consider a system with a single CPU core and three processes A, B, C. Process A arrives at t = 0, and runs on the CPU for 10 time units before it finishes. Process B arrives at t = 6, and requires an initial CPU time of 3 units, after which it blocks to perform I/O for 3 time units. After returning from I/O wait, it executes for a further 5 units before terminating. Process C arrives at t = 8, and runs for 2 units of time on the CPU before terminating. For each of the scheduling policies below, calculate the time of completion of each of the three processes. Recall that only the size of the current CPU burst (excluding the time spent for waiting on I/O) is considered as the "job size" in these schedulers.

(i) First Come First Serve (non-preemptive).

(ii) Shortest Job First (non-preemptive)

(iii) Shortest Remaining Time First (preemptive)

Answer – (i) First Come First Serve (non-preemptive). Ans: A=10, B=21, C=15. A finishes at 10 units. First run of B finishes at 13. C completes at 15. B restarts at 16 and finishes at 21.

Answer – (ii) Shortest Job First (non-preemptive) Ans: A=10, B=23, C=12. A finishes at 10 units. Note that the arrival of shorter jobs B and C does not preempt A. Next, C finishes at 12. First task of B finishes at 15, B blocks from 15 to 18, and finally completes at 23 units.

Answer – (iii) Shortest Remaining Time First (preemptive) Ans: A=15, B=20, C=11. A runs until 6 units. Then the first task of B runs until 9 units. Note that the arrival of C does not preempt B because it has a shorter remaining time. C completes at 11. B is not ready yet, so A runs for another 4 units and completes at 15. Note that the completion of B's I/O does not preempt A because A's remaining time is shorter. B finally restarts at 15 and completes at 20.

cont'd…

Question 8: **[10 Marks]**

(a) Consider a process P1 that forks P2, P2 forks P3, and P3 forks P4. P1 and P2 continue to execute while P3 terminates. Now, when P4 terminates, which process will take P4?
Answer – init (orphan processes are reaped by init)

(b) Given the following program fragment: How many different copies of the variable c are there? What are their values? State clearly which process (parent or child or grandchild) execute first and prints what value?

```
main(int argc, char ** argv)
{
    int child = fork();
    int c = 5;

    if (child == 0)
    {
        c += 5;
    }
    else
    {
        child = fork();
        c += 10;

        if (child)
            c += 5;
    }
}
```

The piece of code shown creates two processes. Therefore, we have a total of three processes, the parent, the first and second child. Each of these has its own private copy of the variable c. For the parent, the variable c be 20 before the end of the program. For the first child (the one created in the first program statement), the variable c will contain the value 10 before the end of the program.

For the second child (the one created in the else clause), the variable c will contain the value 15 before the end of the program.

Reasoning: *There are two fork function invoked in the provided code. The first fork() function duplicates the original (main) process, thus the first child process is generated and within that child process, variable child = 0. However, the variable child in the main process is non-zero since fork return pid of child process to parent process.*

And the second fork() function is only invoked by the main process, within which, as previously mentioned, the variable child is non-zero, and thus the else part of conditioning statement is executed.

To summarize, the main process is duplicated twice while all the child processes are not copied. Therefore, there are three copies of variable c. Postcondition of variable c in each process:

c in main process: c = 5 + 10 = 15
c in first generated child process: c = 5 + 5 = 10
c in second generated child process: c = 5 + 10 + 5 = 20

cont'd…

Question 9: **[10 Marks]**

Refer the given named pipe program fragment and answer the following questions

(a) What programs does p1.c, p2.c and p3.c are referring to

(b) Write the command on Apollo to successfully execute this named pipe program fragment

(c) Write the output of (b)

```
//Program1: p1.c
#include<stdio.h>
#include<sys/types.h>
#include<sys/stat.h>
int main()
    {
        int res;
        res = mkfifo("fifo1",0777);
        printf("named pipe created\n");
    }

 //Program2: p2.c
 #include<unistd.h>
    #include<stdio.h>
    #include<fcntl.h>
    int main()
    {
            int res,n;
            res=open("fifo1",O_WRONLY);
            write(res,"Message",7);
            printf("Sender Process %d sent the data\n",getpid());
    }

//Program 3: p3.c
    #include<unistd.h>
    #include<stdio.h>
    #include<fcntl.h>
    int main()
    {
            int res,n;
            char buffer[100];
            res=open("fifo1",O_RDONLY);
            n=read(res,buffer,100);
            printf("Reader process %d started\n",getpid());
            printf("Data received by receiver %d is: %s\n",getpid(), buffer);
    }
```

cont'd…

Answer – (a) Creating fifo/named pipe, Writing to a fifo/named pipe, Reading from the named pipe

Answer – (b) ./p2 & ./p3

Answer – (c) Shown below

```
ammoham-apollo:/home/ammoham$ gcc -o p2 p2.c
ammoham-apollo:/home/ammoham$ gcc -o p3 p3.c
ammoham-apollo:/home/ammoham$ ./p2 & ./p3
[1] 10483
Sender Process 10483 sent the data
Reader process 10484 started
Data received by receiver 10484 is:
[1]+  Exit 35                 ./p2
ammoham-apollo:/home/ammoham$
```

Question 10: **[10 Marks]**

(a) Consider a paging hardware with a TLB. Assume that the entire page table and all the pages are in the physical memory. It takes 10 milliseconds to search the TLB and 80 milliseconds to access the physical memory. If the TLB hit ratio is 0.6, calculate the effective memory access time (in milliseconds).

(b) The percentage of times a page number is found in the TLB is known as _____

(c) The number of pages that TLB contains in is equal to the page table. Justify this statement.

Answer (b) – hit ratio

Answer (c) – This statement is not accurate. The Translation Lookaside Buffer (TLB) and the page table are both components of a computer's memory management system, but they do not necessarily contain the same number of pages.

The page table is a data structure used by the virtual memory system to store the mapping between virtual addresses and physical addresses. It contains an entry for every page of memory in the system.

On the other hand, the TLB is a small, fast cache that stores a subset of the page table's entries. It is used to speed up the translation of virtual addresses to physical addresses by storing the most recently used or frequently used translations. Because it is a cache, the TLB is typically much smaller than the page table.

Therefore, the number of pages that the TLB contains is usually much less than the number of pages in the page table. The exact number depends on the specific hardware and software configuration of the computer system.

**** The End ****

```
main(int argc, char ** argv)
{
   int child = fork();
   int c = 5;

   if (child == 0)
   {
      c += 5;
   }
   else
   {
      child = fork();
      c += 10;

      if (child)
         c += 5;
   }
}
```

```
//Program1: p1.c
#include<stdio.h>
#include<sys/types.h>
#include<sys/stat.h>
int main()
      {
            int res;
            res = mkfifo("fifo1",0777);
            printf("named pipe created\n");
      }

 //Program2: p2.c
 #include<unistd.h>
      #include<stdio.h>
      #include<fcntl.h>
      int main()
      {
               int res,n;
               res=open("fifo1",O_WRONLY);
               write(res,"Message",7);
               printf("Sender Process %d sent the data\n",getpid());
      }

//Program 3: p3.c
      #include<unistd.h>
      #include<stdio.h>
      #include<fcntl.h>
      int main()
      {
               int res,n;
               char buffer[100];
               res=open("fifo1",O_RDONLY);
               n=read(res,buffer,100);
               printf("Reader process %d started\n",getpid());
               printf("Data received by receiver %d is: %s\n",getpid(), buffer);
      }
```

**** The End ****