COMP1411 (Spring 2022) Introduction to Computer Systems

Individual Assignment 3 Duration: 00:00, 09-Apr-2022 ~ 23:59, 10-Apr-2022

Name	**JIANG Guanlin**
Student number	**21093962D**

Question 1. [4 marks]

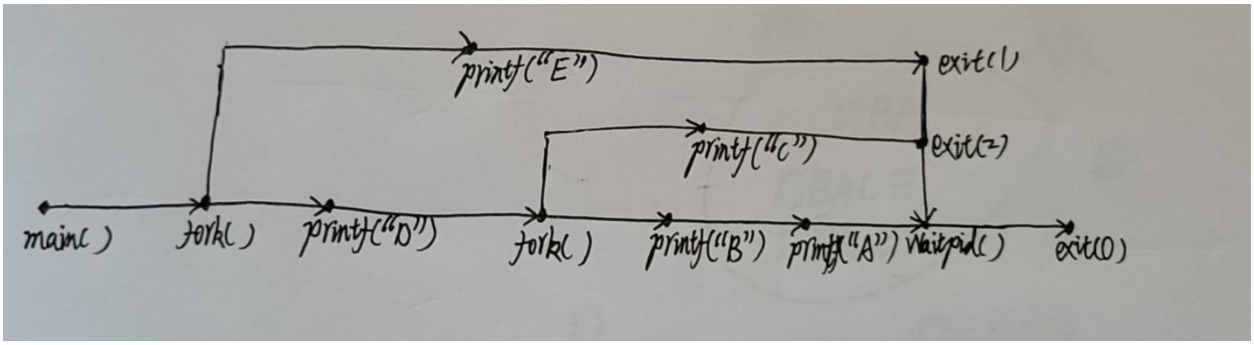
Consider the execution of the following function (written in the C language).

```
// all the needed headers are included
int main()
{
    int pid;

    if (fork() == 0){
        printf("E"); fflush(stdout);
        exit(1);
    }

    printf("D"); fflush(stdout);

    if ((pid = fork()) == 0){
        printf("C"); fflush(stdout);
        exit(2);
    }

    printf("B"); fflush(stdout);

    printf("A"); fflush(stdout);

    waitpid(pid, NULL, 0);

    exit(0);
}
```

1(a) Draw the process graph for the concurrent program.

In a process graph, each function, including main(), fork(), printf(), waitpid(), and exit(), is represented by a vertex. You can omit the fflush() function in the process graph. For each vertex, please write the function name below the vertex, and for printf() write the output character above the vertex. Each edge must be directed, with the direction representing the happen before relationship.

Answer:



1(b) List all the feasible outputs of the program.

For example, if there are two feasible outputs, please give the answer as follows (listing only one feasible output in each line and number them):

(1) A B C D E
(2) E D C B A

Note that the above two outputs are only used to demonstrate the format to give your answers for question 1(b), they do not have any indication of the correct answers.

Answer:

1. DEBAC
2. EDBAC
3. DECBA
4. EDCBA
5. DEBCA
6. EDBCA
7. DCEBA
8. DBACE
9. DBAEC
10. DCBEA
11. DBECA
12. DBCEA
13. DBCAE
14. DCBAE
15. DBEAC

Question 2 [3 marks]

Assume the system has a cache between the CPU and the main memory. Each time the CPU uses some data item, the CPU will always go to the cache to fetch the data item; if the data item is found in the cache, then there is a cache hit, and the data item will be loaded to the CPU; otherwise, if the data item is not found in the cache, then the item will be loaded from the main memory and at the same time put into the cache (we assume that all the accessed data are guaranteed to be stored in the main memory).

Each **cache block** has the size of **8B** (B = bytes), and **the cache has 3 blocks**, which means the **total size** of the cache is **24B**. If one main memory address is accessed, the whole block containing the accessed address will be loaded into the cache.

We assume that the mapping from an address to a block number is given below:

Address range	Block number
0x00 – 0x07	0
0x08 – 0x0F	1
0x10 – 0x17	2
0x18 – 0x1F	3
0x20 – 0x27	4
0x28 – 0x2F	5
0x30 – 0x37	6
0x38 – 0x3F	7

The cache is managed with the **FIFO** replacement policy: when all the 3 cache blocks are used up and a new data block will be loaded into cache, one data block out of the 3 blocks will be replaced out of the cache. The data block that was **earliest installed into the cache** will be replaced. We assume that at the very beginning, the cache is **empty**.

12 main memory addresses are accessed sequentially, listed in Figure 2.

Please fill in Figure 2: input the corresponding data block number for each accessed address into the cache states represented by vertical boxes and input whether the access to the address is a cache hit or miss by filling the brackets below the address, with “M” for cache miss and “H” for cache hit.

For example, starting from an empty cache, access address 0x2A and 0x34, the red contents are those you are supposed to fill in, as shown in Figure 1.

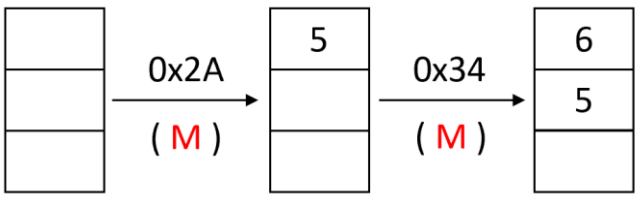
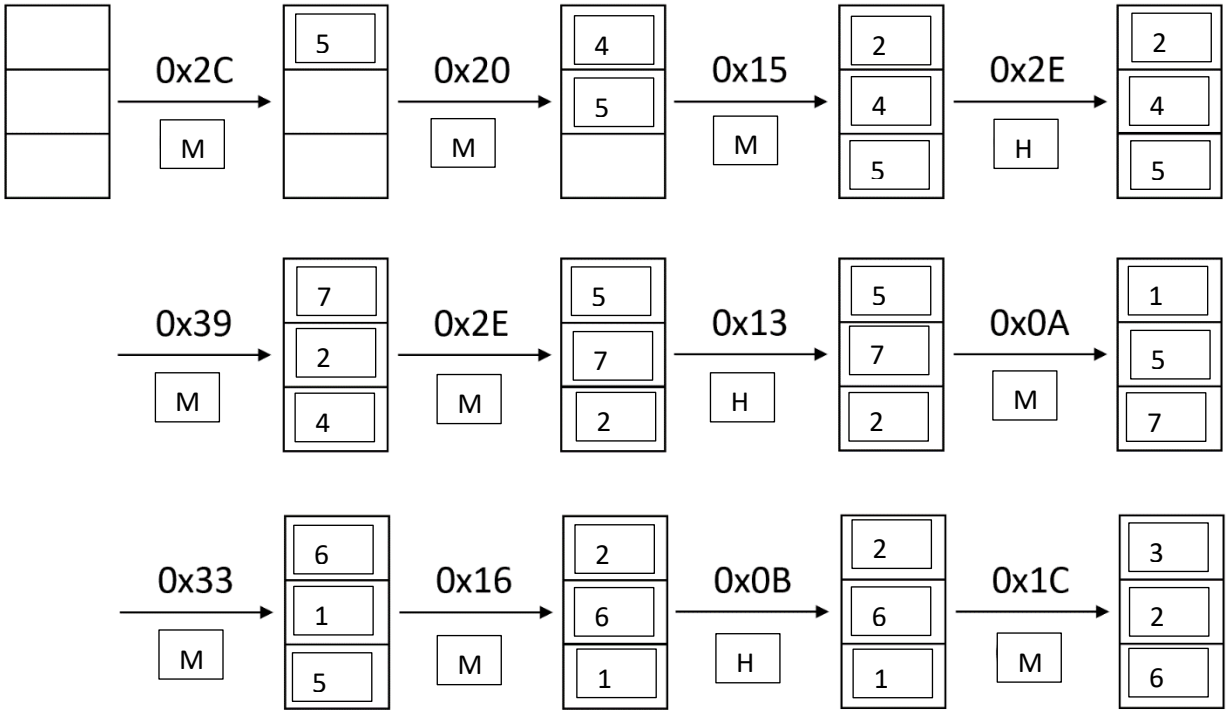


Figure 1

Answer:

Figure 2



Question 3. [3 marks]

Considering a virtual memory system with paging:

Assume that the **page size** is **4KB (B = bytes)**, and each **physical and virtual address** is represented by a **16-bit binary** or equivalently a **4-digit hexadecimal** number. So, the relationship between virtual memory address and virtual page number should be as follows:

Address Range	Virtual Page Number
0x0000 – 0x0FFF	0
0x1000 – 0x1FFF	1
0x2000 – 0x2FFF	2
0x3000 – 0x3FFF	3
0x4000 – 0x4FFF	4
0x5000 – 0x5FFF	5
……	……

A process was assigned **3 physical pages/frames** with frame numbers 7, 8, and 9. This means that no matter how many virtual pages the process has, all its virtual pages must be mapped into only the 3 given frames.

Now it is **time 100** (a larger time value indicates a later point in time). The current content of the page table is shown below with Table 1, in which the "**Access time**" column shows the **time of the most recent visit to the virtual page** in the corresponding row (e.g, 90 indicates that virtual page 2 is visited most recently at time 90).

Page numbers are given in **decimal format**. The "-" symbol in the virtual page column indicates that no virtual page has been mapped to the frame that is listed in the same row as the virtual page. The "-" symbol in the "Access time" column indicates that this physical page is not visited yet.

Assume that **LRU replacement** is used for page replacement. This means if all the 3 physical pages have been mapped with 3 virtual pages, and a 4th different virtual page is accessed, among the 3 mapped virtual pages, the one with the **oldest (smallest) "Access time"** will be replaced by the 4th new virtual page.

From now on, the following virtual addresses will be accessed sequentially:

0x136F, **0x549D**, **0x39C5**, **0x4556** at time **120**, **140**, **160** and **180** respectively.

Please complete the page table after the above four accesses, by filling in the blanks in Table 2. Note that you are required to input the page numbers in decimal format.

Table 1: The original page table

Virtual page number	Physical page number	Access time
2	7	90
5	8	70
-	9	-

Answer:

Table 2: The page table after the four accesses

Virtual page number	Physical page number	Access time
____3____	7	____160____
____5____	8	____140____
____4____	9	____180____