Computing Science 300

Sample Quiz 1 SOLUTION

1. Consider a computer system that includes a single 200 GB internal hard disk, 4 GB of RAM, 1 DMA and 1 output device that writes 1 word of data at a time. The DMA is used to transfer data and instructions between RAM and internal disk. It can transfer up to 200Mb at a rate of 12.5Mb/s. The DMA setup takes 4ms. The DMA sends an interrupt when it has completed the transfer. The output hardware setup takes 2ms and transfer of 1 word of data takes 1ms. The output device busy waits during the transfer of data. When an ISR, hardware setup or other process completes the OS scheduler is run to determine which process will run next. Assume that after the DMA transfer completes the OS scheduler will choose the process that started the DMA transfer as the next process to run. Time taken by context switches and ISRs can be ignored. State any further assumptions you make in your solution.
   a) **[12 points]** Explain briefly what happens during the DMA setup.
   b) **[18 points]** How long would it take to transfer a 500Mb block of data?
   .

a) During DMA setup the registers in the DMA hardware are initialized.
The CPU sends values to the DMA that will be placed in the control registers of the DMA.
These values tell the DMA which memory should be copied, how much memory should be copied, where the block of memory to be copied begins, and where the block of memory that the DMA should copy into begins. For a simple DMA this may be a simple as 3 values similar to those listed below. For a more sophisticated DMA the information transferred at setup time may be more extensive allowing a single DMA setup to copy multiple blocks of data from multiple different locations to multiple different locations.
For example, the values that the CPU may provide to initialize a DMA may be

- ***The memory address of the first word of the block of memory it is to copy from***
- ***The memory address of the first word in the block of memory to copy too***
- ***The number of bytes (words) to copy***

b)To calculate the time to transfer a block of 500Mb of data consider the following steps

- ***Observe the size of the block (500Mb) is larger than the size of the block the DMA can transfer, so it will be necessary to set up the DMA more than once***
- $.\frac{(size\ of\ block\ of\ data\ to\ be\ transferred)}{(\max size\ of\ block\ DMA\ can\ transfer} = \frac{500Mb}{200Mb} = 3$ ***so 3 transfers are needed***
- ***Each transfer of 200 Mb takes 16 s (16s * 12.5 Mb/s = 200 Mb)***
- ***The third transfer, transfers only 100MB and takes 8s (8s*12.5 Mb/s = 100MB)***
- ***The DMA must be set up for each of the 3 transfers 4ms for each transfer***
- ***For the first two transfers***
  2 transfers * (16s / transfer + 4ms / transfer) = 2 * 16 s + .004 = 32.008 s
- ***For the 3rd transfer***
  (8s / transfer + 4ms / transfer) = 8.004 s
- ***Total time for transfer 12ms + 40s = 40.012 s***

2. The C code snippet shown below is a small chunk of the code extracted from a large application. This code snippet shows a parent process creating a child process.

```
1. if ( (pid = fork()) == 0 ) {
2.       processitInner(); /* function called completes task 1 */
3.       exit (0);
4. }
5. processitOuter(); /* function called completes task 2 */
```

Answer each of the questions below. Explain your answers

- **[6 points]** Is task2 done by the child process?
- **[6 points]** What does the fork() function do?
- **[6 points]** What is the expected return value of the fork() function?
- **[6 points]** Are all illustrated lines of code executed by the parent process?
- **[6 points]** What were the differences between the child's process control block and the parents process control block that were discussed in class?

SOLUTION

— **Line 1 of the code begins by executing fork() which creates a copy of the present process.**

— **The copy has its own process control block.**

— **The newly created copy process is the child process; the original process is the parent process.**

— **In the parent process, the return value of the function fork() is the process id of the child process.**

— **in the child process The return value of fork( ) is 0**

Task 2 is not done by the child process.

- **In the child process the return value of fork() is 0, so pid = 0**
- **Because pid=0 the condition in the if statement is true**
- **Therefore, execution continues with lines 2 and 3 of the code**
- **After line 3 has executed the child process has terminated, it will never reach line 5 where task 2 is completed.**

What does the fork() function do

- ***It creates an exact copy of the present process in the RAM memory of the system***
- ***It creates a new process control block for the new (copied) process***
- ***The new process control block is placed in the queue of process control blocks***
- ***Both the parent process and the child process continue execution from the return to the fork command to the next operation in the code (the test to determine which line is executed next in line 1)***

The expected return value of the fork function is

- **In the child process the return value of fork() is 0, so the variable pid = 0**
- **In the parent process the return value of fork() is the process id of the child process (a positive integer)**

Are all illustrated lines of code executed by the parent process?
NO
Since the return value for the parent process is the pid of the child process, the test in line 1 of the code above is false.
Lines 2, 3 and 4 inside the if are not executed
The next line executed is line 4

The differences between the child's process control block and the parents process control block that were discussed in class

- ***The Parent and child processes have different process ids***
- ***The child has the parents pid as its own pid of parent, while the parent's pid of parent is the pid of the process that created the parent process***
- ***The parent adds the pid of the child to its list of process ids of its children***

3. Consider a system that uses a 64-bit word. This system uses a 512 byte cache memory on the CPU that has a 64 byte cache slot size. The system has of the 1 Mbyte of RAM memory. Remember 1 byte = 8 bits and 1Mbyte=1024*1024 bytes and 1Kbyte = 1024 bytes. When a cache line is placed in the cache, the following rules are used.

   i. There are M cache lines in this system. The first cache line in the cache is cache line 0, the next is cache line 1, and so on. The last cache line is cache line M-1. All cache lines are the same size.)
   ii. There are N cache slots in this system. The first cache slot is cache slot 0, the next is cache slot 1, and so on. The last cache slot is cache slot N-1. All cache slots are the same size.
   iii. Cache line K can be placed in cache slot K%(N/2) and cache slot K%(N/2)+N/2
   iv. If both cache slot K%(N/2) A and cache slot K%(N/2)+N/2 are empty the cache line will be placed in cache slot K%(N/2).
   v. If cache slot K%(N/2) is full and cache slot K%(N/2)+N/2 is empty the cache line will be placed in the empty slot.
   vi. If both cache slot K%(N/2) and cache slot K%(N/2)+N/2 are full the cache line will be placed in the cache slot with the earliest last accessed time.

   a) **[8 points]** What is a cache slot? How many cache slots would there be in this computer system? Where would the cache slots be located?
   A cache slot is a section of the cache memory. The cache memory is located on the CPU, the cache slot is located in the cache memory. Each cache slot is 64 bytes. The size of the entire cache memory is 512 bytes. Therefore, if we divide the size of the cache memory by the size of the cache slot we will find the number of cache slots 512/64 = 8. N=8
   b) **[8 points]** What is a cache line? How many cache lines would there be in this computer system? Where would the cache lines be located?
   A cache line is a section of the RAM memory. The cache line is located in the RAM memory. The cache line is the same size as the cache slot so the size of each cache line is 64 bytes. The size of the entire RAM memory is 1 Mbyte = 1024*1024 bytes. Therefore, if we divide the size of the RAM memory by the size of the cache line we will find the number of cache lines 1024*1024/64 = 16,384 M=16,384
   c) **[8 points]** What is the purpose of a mapping function? For the system described above what would the mapping function be?
   The purpose of the mapping function is to define the rules that determine which cache lines may be placed into which cache slots. The mapping function for this system is: Cache line K can be placed in cache slot K%(N/2) and cache slot K%(N/2)+N/2
   d) **[10 points]** What is the purpose of a replacement policy? For the system described above what would the replacement policy be?
   The purpose of the replacement policy is to define the rules that determine which of the MAPPED cache slots to place the cache line into. The mapping function for this system is:
      iv. If both cache slot K%(N/2) A and cache slot K%(N/2)+N/2 are empty the cache line will be placed in cache slot K%(N/2).

v. ***If cache slot K%(N/2) is empty and cache slot K%(N/2)+N/2 is full, or cache slot K%(N/2) is full and cache slot K%(N/2)+N/2 is empty the cache line will be placed in the empty slot.***

vi. ***If both cache slot K%(N/2) A and cache slot K%(N/2)+N/2 are full the cache line will be placed in the cache slot with the earliest last accessed time.***

e) **[6 points]** Which cache slots could hold cache line 1234.
1234%(8/2) = 2   1234%(8/2)+(8/2) = 6   slots 2 or 6 could hold cache line 1234