1.

x – marks the chosen hole

a) First Fit

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Jobs**  **Holes** | **T1 (122k)** | **T2 (105k)** | **T3 (203k)**  **No possible holes** | **T4 (90k)** |
| **102k** | Too small | Too small | Too small | x |
| **205k** | x | Taken | Taken |  |
| **43k** |  | Too Small | Too small |  |
| **180k** |  | x | Too small |  |
| **70k** |  |  | Too small |  |
| **125k** |  |  | Too small |  |
| **91k** |  |  | Too small |  |
| **150k** |  |  | Too small |  |

b) Best Fit

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Jobs**  **Holes** | **T1 (122k)** | **T2 (105k)** | **T3 (203k)** | **T4 (90k)** |
| **102k** | Too small | Too small | Too small | Left Over = 12k |
| **205k** | Left Over = 83k | Left Over = 100k | Left Over = 2k (x) | Taken |
| **43k** | Too small | Too small | Too small | Too small |
| **180k** | Left Over = 58k | Left Over = 75k | Too small | Left Over = 90k |
| **70k** | Too small | Too small | Too small | Too small |
| **125k** | Left Over = 3k (x) | Taken | Taken | Left Over = 35k |
| **91k** | Too small | Too Small | Too small | Left Over = 1k (x) |
| **150k** | Left Over = 28k | Left Over = 45k (x) | Taken | Taken |

c) Worst Fit

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Jobs**  **Holes** | **T1 (122k)** | **T2 (105k)** | **T3 (203k)**  **No Possible Holes** | **T4 (90k)** |
| **102k** | Too small | Too small | Too small | Left Over = 12k |
| **205k** | Left Over = 83k (x) | Taken | Taken | Taken |
| **43k** | Too small | Too small | Too small | Too small |
| **180k** | Left Over = 58k | Left Over = 75k (x) | Taken | Taken |
| **70k** | Too small | Too small | Too small | Too small |
| **125k** | Left Over = 3k | Left Over = 20k | Too Small | Left Over = 35k |
| **91k** | Too small | Too Small | Too small | Left Over = 1k |
| **150k** | Left Over = 28k | Left Over = 45k | Too Small | Left Over = 60k (x) |

2.

i)

a)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Frames** | 201 | 404 | 205 | 201 | 207 | 302 | 302 |
| 302 | 302 | 206 | 203 | 203 | 203 | 203 |
| 203 | 201 | 302 | 302 | 206 | 201 | 206 |
| **Page Faults** | 3 | 5 | 8 | 10 | 12 | 14 | 15 |

b)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Frames** | 201 | 404 | 206 | 206 | 207 | 201 | 201 |
| 302 | 302 | 201 | 302 | 203 | 302 | 206 |
| 203 | 203 | 205 | 201 | 206 | 206 | 203 |
| **Page Faults** | 3 | 5 | 8 | 10 | 13 | 15 | 17 |

c)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Frames** | 201 | 201 | 201 | 201 | 203 | 203 | 203 | 203 |
| 302 | 302 | 302 | 302 | 302 | 207 | 302 | 302 |
| 203 | 404 | 205 | 206 | 206 | 206 | 201 | 206 |
| **Page Faults** | 3 | 4 | 5 | 6 | 7 | 8 | 10 | 11 |

ii)

a)

|  |  |  |  |
| --- | --- | --- | --- |
| **Frames** | 201 | 201 | 201 |
| 302 | 302 | 302 |
| 203 | 206 | 206 |
| 404 | 207 | 203 |
| 205 | 205 | 205 |
| **Page Faults** | 5 | 7 | 8 |

b)

|  |  |  |
| --- | --- | --- |
| **Frames** | 201 | 206 |
| 302 | 201 |
| 203 | 302 |
| 404 | 203 |
| 205 | 207 |
| **Page Faults** | 5 | 10 |

c)

|  |  |  |  |
| --- | --- | --- | --- |
| **Frames** | 201 | 201 | 201 |
| 302 | 302 | 302 |
| 203 | 203 | 203 |
| 404 | 404 | 207 |
| 205 | 206 | 206 |
| **Page Faults** | 5 | 6 | 7 |

3.

a)

The page memory reference will take at least 500ns. This is due the fact that first, the page number will be recovered from memory, which takes 250ns, then the memory location will be calculated using the page number and finally the memory address will be referenced which will take another 250ns.

b)

The formula to calculate this effective memory access time is the following:

Effective Access Time = Hit Rate \* (Memory Access Time + TLB Overhead) + Fail Rate \* (2 \* Memory Access Time + TLB Overhead)

= 0.8 \* (250 + 30) + 0.2 \* (2 \* 250 + 30)

= 0.8 \* (280) + 0.2 \* (530)

= 330ns

The reason the EAT is calculated as such, is because it is simply a weighted average between the hit chances and the failure chances with their respective times to access memory.

c)

Adding another layer such as the TLB can greatly improve performance time due to a few reasons. First, since the hit ratio is very favourable (.8 hit, .2 fail) it will be hitting many more times than failing. Excluding the TLB, the time it takes is 500, but in the best case scenario WITH TLB, its 330 ns. Second of all, since the fail case is effectively only 30 ns longer than normal (530 – 500 = 30ns), this is negligible as there are far more hits than failures.

4.

a) The processor’s logical address is a combination of the page number and the page offset. The page offset is calculated by taking the size of the page and then finding the corresponding power of 2 which in this case, would be 11. The closest power of 2 to 2Kbytes is 2^11 as it gets 2048 bytes, meaning the page offset will be 11 bytes. The page number is calculated by taking the size of a page (2^11) and subtracting that from the total size of the logical address space. The logical address space is 32 pages of the 2kbytes meaning that the full logical address space is 2000bytes \* 32 which is 64 000 bytes. The closest power of 2 is 16, therefore the page number in the logical memory address is 16 – 11 which is 5 bytes.

b) The length of the page table associated with this logical memory space is going to be 32 pages long. The width of the page table will only be 5 bytes as the control bit is disregarded meaning that only the page number is required.

c) If the physical memory size is cut in half, there will be no effect on the page table other than the control bits. Those will be changed to allow for the proper paging into the new memory space but otherwise in terms of length and width only the logical memory space can affect it.

5.

a)

A race condition is when two processes try to access the same resource, but the outcome differs depending on which process access the resource first. An example of this is when two processes try to access shared memory and writes an integer value to it. Both are trying to access at the same time but slightly differ in their access speed and then causes a race. The outcome will depend on which process arrives first and writes the integer value.

b)

Disabling interrupts is not a good mechanics to prevent race conditions because take for example a time sharing system. If we were to prevent interrupts in one computer, all computers would have to be interrupted. This would cause a huge overhead. In a uniprocessor system this is fine because only one process may be active at a time.

6.

A semaphore is a mutex which allows multiple processes to access the same shared memory. Depending on the value of the semaphore, the process will either be blocked from shared memory or accepted.

7.

The open operation is a system call that the user can invoke, meaning that a request is sent to the kernel. The user can specify what operation they want to perform on the file and then the kernel has to check if the users permission is allowed on the user’s requested file. The kernel will then move the file on to the “Open-File Table” if the user is allowed to access the file with their specified mode. The user can then access their file as needed.

8.

a)

Pointer block size = Disk Block Size / pointer size

= 8192 Bytes / 4 Bytes

= 2048 bytes

Total file size =

(12 \* 8KB) + (2048 \* 8KB) + ((2048 ^ 2) \* 8KB) + ((2048 ^ 3) \* 8KB) = 64 TB

b) In order to store a file that is larger than the maximum size computer, the size of the disk blocks need to be increased. An example of this is if the disk block size is increased to 16 KB from 8KB. If the pointers are assumed to keep a size of 4 Bytes, then the size of the pointer block now becomes 16KB/4B = 4096B. The total file system size can now be calculated as follows:

(12 \* 16KB) + (4096 \* 16KB) + ((4096 ^ 2) \* 16) + ((4096 ^ 3) \* 16) = 1126 TB