Computing Science 300 Assignment 3 Solution

1. A particular computer system has a virtual memory with 4096 pages. The physical memory of this computer system has 16 page frames. The size of each page frame is 8Kb (1Kb=2^{10} bits). Each lower level page table entry uses 8 bytes. Each upper level entry uses 8 bytes.

   a) **[4 points]** Assume that each byte of memory in the virtual memory has an address. How many addresses would be needed to address all bytes in one page? Why? How many bits would be need in an address that provided a unique address for each byte in a page?
   The size of each page frame is equal to the size of each page.
   Therefore, each page hold $8*2^{10}=21^{13}$ bits=2^{10} bytes,
   One address is used to address one byte, therefore we would need 2^{10} = 1024 addresses to address one page of memory.
   We would need 10 bits to provide a unique address for each page

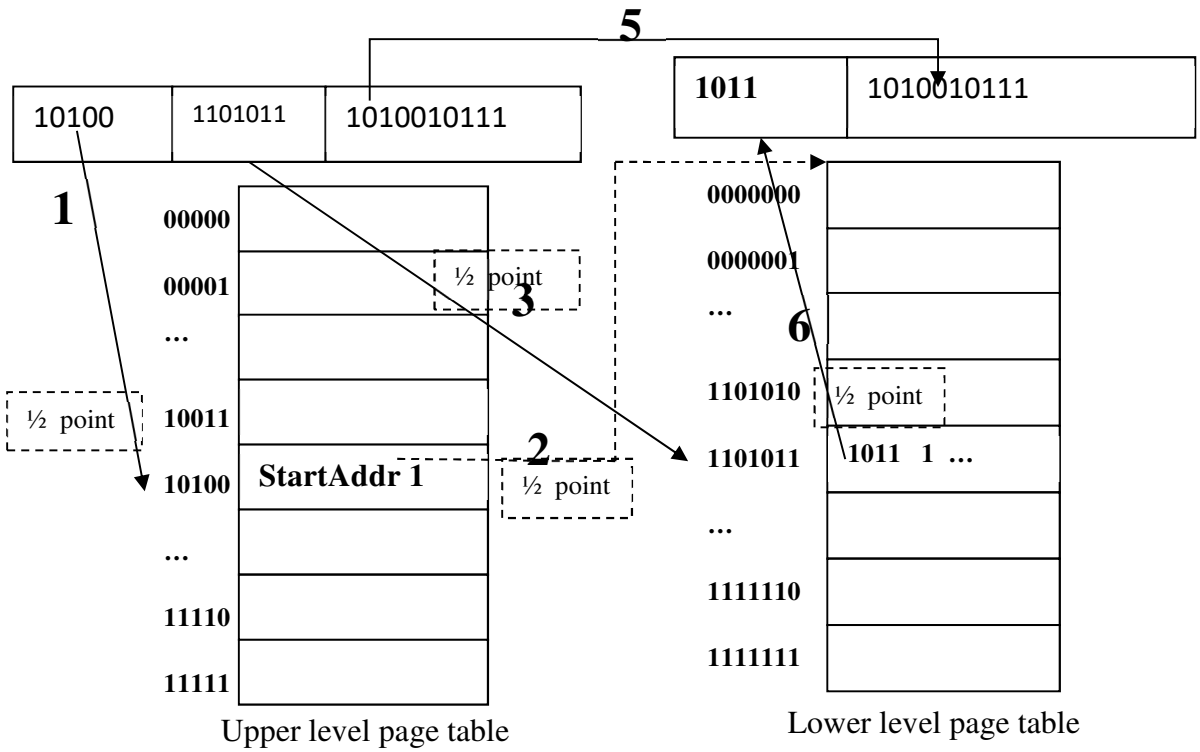
   b) **[4 points]** How many page table entries fit in one page of virtual memory? Why? What is the minimum number of bits would be needed in an address that provided a unique address for every page table entry in a particular page of page table entries?
   Each page is 8 Kb=1KB
   Each lower level page table entry is 8 bytes
   1024/8=128 page table entries per page
   To address 128 page table entries we need 128 addresses. To get 128 unique addresses we need 7 bits

   c) **[4 points]** What is the minimum number of pages of lower level page table needed to hold all the page table entries? Why? What is the minimum number of bits we could use to address the pages of lower level page tables?
   There are 4096 pages of virtual memory, so we need 4096 page table entries.
   We need 4096/128= 32 pages of page table
   To address 32 pages of page table we need 32 unique addresses, to get 32 unique addresses we need 5 bits.

   d) **[9 points]** The diagram below illustrates a two level page table. Not all entries in the upper and lower page tables are shown as boxes in the diagram (there are more entries than there are illustrated boxes). Consider the virtual address 10100110101110100101 11. Assume the page of the lower level page table illustrated below contains the page frame entry for the page containing the virtual address above. Assume the address of the start of the illustrated lower level page table page is StartAddr. On the diagram below add the following information.
   - The virtual address and the corresponding physical address.
   - Offsets into the tables, for the entries in the upper and lower level page tables. Place the offset from the start of the page table page to the left of each box in the diagram. Note that **…** may be placed to the left of any box to indicate that a series of consecutive addresses has been omitted.
   - The contents of the entry in the upper level page table used to translate the given virtual address to a physical address.

- The contents of the entry in the lower level page table used to translate the given virtual address to a physical address.

Be sure to provide the correct number of binary digits including leading zeros in the boxes for each part of the virtual and physical addresses, in the entries in the page tables. You should indicate what each item in the lower level page table entry you provided represents.



e) **[9 points]** Explain step by step, and illustrate on the diagram above, each step in the process of translating a virtual address to a physical address. You should number the steps in your explanation and label your additions to the diagram. The labels on your additions to the diagram should be the number of the step in the explanation that is being illustrated

1. ***Entries in the upper level page table reference a particular page of the page table. Take the first 5 digits (digits 1-5) of the virtual address and use them as an index into the upper level page table.***
2. ***This will give you the address of the desired page of the lower level page table. Go to the correct page of the lower level page table***
3. ***Each page of the lower level page table is full of page table entries. Use the next 7 digits (digits 6-12) to index into the lower level page table and find the desired page table entry***
4. ***Inside the page table entry use the flag that indicates if the page of memory is in a page frame. If the page is not in memory page fault and load it into memory, then update the page table entry***
5. ***Take the offset into the page (the remaining 10 unused digits in the address) and make them the rightmost digits of the physical address***
6. ***Take the page frame number from the page table entry and prepend it before the offset to give the physical address***

2. **[25 points]** (NOTE: this is an example of the type of problem that might appear on a written exam for this topic) Consider a system used to control entry into a ride at an amusement park. The ride is extremely popular and will only be operated when it is full. A full load for this ride is two adults and four children. People (adults or children) are added to the system at random intervals. Before we begin to load the ride we must wait until two adults and four children are present. Then the six people (two adults and four children) can be loaded into the ride. Each person in the load must execute function load() to be loaded onto the ride. After an adult or a child has been successfully loaded (the load() function is complete) that person must signal the ride that they are loaded (signal a semaphore "whoIsLoaded"), then wait (wait a semaphore "loadingComplete") until all the other passengers in the load are loaded. When the ride has received signals from all the people in the load the ride signals the "loadingComplete" semaphore once for each of the people in the load. When an adult or child is released from the "loadingComplete" semaphore's queue it will terminate. Write pseudo code, using processes, mutexes, and semaphores, for the functions representing the child and adult. ***DO NOT*** write the pseudo code, for generating people (adults and children) at random intervals, or for the function load().

   Remember that

   semWait

   - Decrements the semaphore value
   - If the value is >=0, the process is allowed to run its critical region
   - Otherwise the process is blocked (put to sleep) and placed in the blocked queue

   semSignal

   - Increments the semaphore value
   - If the semaphore value <=0 the first process in the blocked queue is woken up and placed in the ready queue

   mutexWait

   - If the value is 1, the value is changed to 0 and the process is allowed to run its critical region
   - If the value is 0 then the process is blocked and placed in the blocked queue

   mutexSignal

   - If there are processes in the blocked queue, unblock the first process in the queue
   - If there are no processes in the blocked queue set the semaphore value to 1

   Keep in mind this is only one possible solution

```
int child;    /* counter for number of children present */
int adult; /* counter for number of adults present */
mutex   protectChild = 1;
mutex   protectAdult = 1;
semaphore adults = 0;
semaphore children = 0;
semaphore whoIsLoaded = 0;
semaphore checkLoaded = 0;
childLoad() {

        mutex_wait(&protectChild)
        mutex_wait(&protectAdult);
```

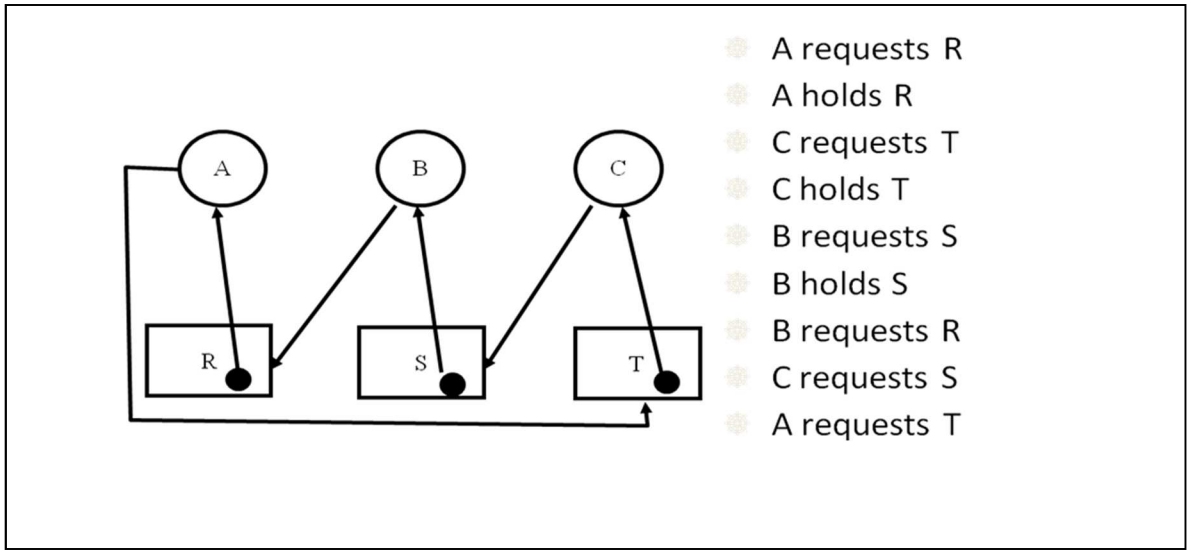
```
        child += 1
        if (adult >= 2 && child >= 4) {          both parent and child must be tested
                semaphore_signal(&adult);        correct number of signals
                semaphore_signal(&adult);
                adult -= 2;
                semaphore_signal(&child);            correct number of signals
                semaphore_signal(&child);
                semaphore_signal(&child);
                child -= 4
                mutex_signal(&protectAdult)      must be opposite order from waits
                mutex_signal(&protectChild)
        }
        else {
                mutex_signal(&protectAdult)      must be opposite order from waits
                mutex_signal(&protectChild)      must happen for adult about to wait
                semaphore_wait(&child);
        }
        load()
        semaphore_signal(&whoIsLoaded);
        semaphore_wait(&checkLoaded);
        exit(0);
}

adultLoad() {
        mutex_wait(&protectChild)                    protection same order as child
        mutex_wait(&protectAdult);
        adult += 1
        if (adult >= 2 && child >=4 )            both child and adult tested
                semaphore_signal(&adult);            correct decrement
                adult  -=  2                         correct number signals
                semaphore_signal(&child);
                semaphore_signal(&child);
                semaphore_signal(&child);            correct decrement
                semaphore_signal(&child);            correct number signals
                child  -=  4
                mutex_signal(&protectAdult)
                mutex_signal(&protectChild)
        }
        else {
                mutex_signal(&protectAdult)
                 mutex_signal(&protectChild)
                semaphore_wait(&adult);
        }
        load()
        semaphore_signal(&whoIsLoaded);
        semaphore_wait(&checkLoaded);
        exit(0);
}
```

3. Consider three processes, Process A, Process B and Process C. All three of these processes are currently loaded into the memory of our computer system and are sharing the CPU and other resources of our computer system. Each of these processes may use a shared variable (resource S), a printer (resource T), and a DMA (resource R).

   a) **[10 points]** Draw a resource graph illustrating a circular wait involving processes A, B and C and resources R, S and T. Give a step by step description of how the group of processes and resources reached the state illustrated in your resource diagram. The step by step description should include each request for a resource made by a process, each time a requested resource is allocated to a process, and each time a process blocks waiting for a resource. Label your diagram to clearly identify what each symbol in the diagram represents.

   Ovals represent processes, rectangles represent resources. Each time a process requests a resource a directed line from the process to the resource is added to the resource graph. Each time a resource is acquired and held by a process the direction of the directed line between the process and the resource is reversed (now the line points from the resource to the process.



b) **[10 points]** Consider a series of processes that are sharing a variety of resources within a computer system. To avoid or recover from deadlocks it is necessary to be able to determine if the system is likely to deadlock. It is possible to decide if the system may deadlock based on the 'state' of the system determined using the banker's algorithm. A particular multiprocessing computer system is running 3 processes and is described by the matrices A, C, E and R below. Is the system described by the matrices A, C, E and R in a safe state? Use the banker's algorithm to answer this question (show your work step by step).

$$\mathbf{R} = \begin{bmatrix} 3 & 8 & 0 & 1 & 2 \\ 1 & 1 & 5 & 3 & 3 \\ 4 & 6 & 1 & 6 & 4 \end{bmatrix} \qquad C = \begin{bmatrix} 1 & 3 & 0 & 1 & 2 \\ 0 & 0 & 3 & 3 & 3 \\ 2 & 4 & 0 & 3 & 2 \end{bmatrix}$$

$$A = [1 \quad 3 \quad 2 \quad 0 \quad 0] \qquad E = 4 \quad 10 \quad 5 \quad 7 \quad 7$$

$$R-C=\begin{bmatrix}2 & 5 & 0 & 0 & 0\\ 1 & 1 & 2 & 0 & 0\\ 2 & 2 & 1 & 3 & 2\end{bmatrix} \qquad A=\begin{bmatrix}1 & 3 & 2 & 0 & 0\end{bmatrix}$$

Numbers shown in read indicate resources that may be oversubscribed

[2 5 0 0 0] > A cannot run (1 point)

[1 1 2 0 0] <= A runs to completion(1 point) (1 point)

When this second process runs to completion A becomes

$A=\begin{bmatrix}1 & 3 & 5 & 3 & 3\end{bmatrix}$ after adding the third row of C ***(1 point)***

[2 5 0 0 0] > A cannot run (1 point)

[2 2 1 3 2] > A cannot run (1 point)

Processes 1 and 3 may be deadlocked (1 point) the state is unsafe. (1 point)

4. [25 points] You will make a model of Smaug's world in which each visitor, sheep, and cow is a process, and Smaug is also a process. These processes interact in the same way as the characters in Smaug's world interact. Each sheep, cow, hunter, or theif process interacts with Smaug only once, after interacting the process terminates. As a first step in creating this simulation you will build state diagrams describing each kind of process (Smaug, sheep, cow, treasure hunter, thief).

SMAUG"S WORLD

The fierce dragon Smaug is guarding his hoard of treasure. Smaug is a large and perpetually hungry dragon with a very sensitive nose. Smaug likes to sleep on his bed of treasure. He only wakes up to eat, to battle any foolhardy treasure hunter that decides to try and kill him to win all of his treasure, or to play with any thief that tries to sneak in and steal some of his treasure.

Smaug is a magical dragon who lives in a cave. The only entrance to Smaug's cave is in a secluded valley. When a sheep or a cow wanders into that secluded valley it is enchanted and cannot leave the valley. When anyone wishes to visit Smaug's cave they have to travel along a magical path that seems to go on forever in order to find the valley.

When Smaug has no visitors he eats and sleeps. Smaug is particularly fond of sheep as a tasty snack. His nose is so sensitive he only wakes up to eat if he smells three sheep and two cows in the valley. While he is eating those five beasts, he ignores any additional beast that wanders into the valley and becomes enchanted. When he is done eating the five beasts he takes a short nap (he does not go to sleep, just naps). When he wakes he takes one deep breath to see if he can smell another snack or a visitor to his valley. If he smells another snack of three sheep and two cows he will eat it before he takes another short nap. After the shot nap he will take another deep breath. He will continue to eat snacks and take naps as long as there are snacks available. If he does not smell a snack or a visitor to his valley he goes to sleep.

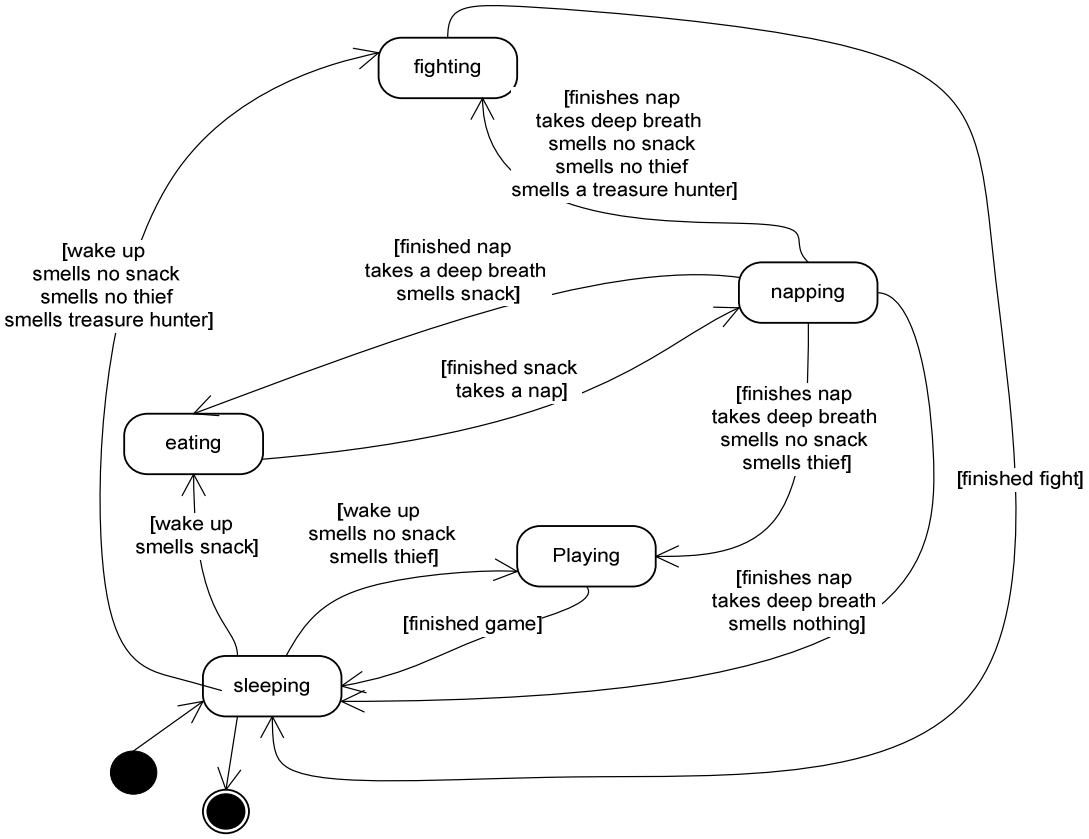
This quiet existence is occasionally interrupted by the arrival of visitors. Visitors to Smaug's cave are trying to obtain some of Smaug's treasure. Smaug's treasure is more important to Smaug that any tasty snack. Smaug's enchanted path will cause visitors to walk in circles, never finding his cave, until Smaug is ready to interact with them. Therefore, Smaug does not worry about interrupting his snacks to interact with visitors and protect his treasure. If Smaug is awake (eating a snack) or napping Smaug will detect any new visitors on the path when he takes a deep breath (after finishing his nap after his snack). If Smaug is asleep (not napping) his nose will wake him up when any visitor is walking along the magical path.
If there no snacks and there are visitors Smaug will interact with the next visitor.

Sometimes Smaug wakes up and smells one or more visitors walking along the path AND a snack is waiting to be eaten. If he smells a snack, he will eat the snack first. When he wakes up from his nap after eating his last snack he will then eat any further available snacks then interact with any visitors. After Smaug finishes dealing with all visitors and snacks he is exhausted and goes to sleep.

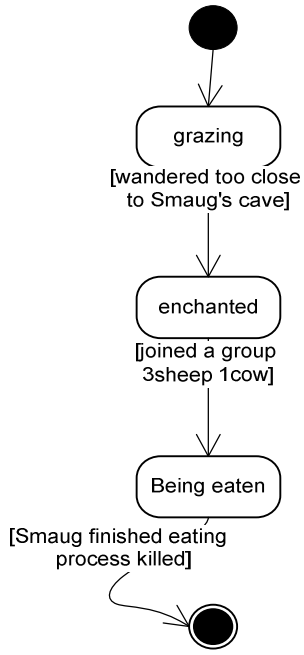
Smaug has two types of visitors, treasure hunters who plan to battle Smaug and kill him to obtain all his treasure and thieves who plan to sneak in and steal some of his treasure. When visitors arrive outside the valley that contains the entrance to Smaug's cave they must find the single magically concealed path into the valley. The spell on the path will cause then to wander through the valley not reaching Smaug's cave until Smaug is ready to meet them (after he smells them). When Smaug is ready, the spell allows the next visitor to see the entrance to the cave and enter the cave. When a treasure hunter enters Smaug's cave, Smaug fights the treasure hunter. Smaug is never killed by a treasure hunter but some fight well enough that Smaug gets tired and allows them to leave with a reward of 10 gems for providing him with exercise. If Smaug defeats a treasure hunter he will let the treasure hunter leave (instead of eating him) if the treasure hunter bribes him with 5 jewels. Since treasure hunters do not like to be eaten they always take five jewels with them just in case they are defeated (most are). When a thief enters Smaug's cave Smaug plays with the thief. If the thief wins the game he is sent home with 8 gems, if the thief loses he must pay 20 gems to leave the cave. No thief is stupid enough to come without 20 gems in his pocket. Smaug enjoys playing games with thieves, but finds fighting treasure hunters to be a boring task. Therefore, Smaug will always choose to let a thief come to play before he lets a treasure hunter come to fight.
After playing with a thief or fighting a treasure hunter Smaug goes to sleep.

Design your model using state diagrams. Develop one state diagram describing Smaug, one describing a sheep, one describing a cow, one describing a thief and one describing a treasure hunter. State diagrams for treasure hunter and for thief must include an initial state in which the treasure hunter or thief has not yet reached the path into the valley (is travelling to the path). State diagrams for sheep and cows should include a state when the sheep and cows are grazing, before they wander too close to the valley containing the dragons cave and are enchanted. Be sure to include the state diagrams with your submitted solution.

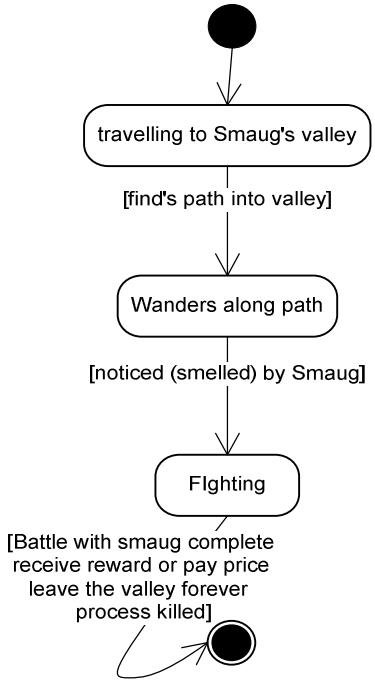
SMAUG



Cow or Sheep



treasure hunter



thief