Instructions:

- Print your name in the space provided below.
- This examination is closed book and closed notes. No calculators or other computing devices may be used.
- Answer each question in the space provided. If you need to continue an answer onto the back of a page, clearly indicate that and label the continuation with the question number.
- If you want partial credit, justify your answers, even when justification is not explicitly required.
- There are 10 questions, priced as marked. The maximum score is 100.
- When you have completed the test, sign the pledge at the bottom of this page and turn in the test.
- Note that either failing to return this test, or discussing its content with a student who has not taken it is a violation of the Honor Code.

Do not start the test until instructed to do so!

Name Solution
printed

Pledge: On my honor, I have neither given nor received unauthorized aid on this examination.

signed

1. [6 points] Briefly describe the role the trap instruction plays, from the perspective of a user program.

   When a user program calls a system function, the user-accessible function is merely a stub that sets the system into kernel mode and then invokes the trap instruction in order to call the actual system function. The trap instruction searches the trap table for the address of the entry point of the matching actual system function and then issues an unconditional branch to that address.

2. [6 points] When the trap instruction was introduced, it was suggested that it might not be a good idea for a user program to be able to access the trap table (index) directly. Give an example of an undesirable possibility if a user program could modify the contents of the trap table.

   In general, if a user program could access the trap table directly then it could call protected system functions without first setting the system to kernel mode. In particular, if a user program could modify entries in the trap table it could "hijack" system calls by substituting addresses of user code for those of protected system functions.

3. [6 points] Assume that it is important to prevent user programs from executing certain instructions. Describe one particular way in which hardware support might be provided to implement such restrictions.

   The hardware can include a mode bit to distinguish whether privileged code or user code is executing. The mode bit would only be altered via a privileged instruction.

4. [12 points] List six items that should be included in or linked to the process control block (process descriptor) for a user process.

   process id | **id of process owner**

   process page map | **process priority**

   process status registers | **process data registers**

5. [10 points] The simple program shown below creates a collection of threads that are intended to cooperate in the initialization of the elements of an array. Will the program always produce correct results, sometimes produce correct results, or never produce correct results? If not, explain how to fix it. If yes, explain why. Assume appropriate includes and declarations necessary for the code to compile and link; that is not the point.

```
const unsigned int MAX_THREADS = 256;
const unsigned int OFFSET      = 100;
struct Package {
      int* L;
      int  sIdx;
};
. . .
int main() {

   pthread_t hThread[MAX_THREADS];
   int* List = new int[OFFSET * MAX_THREADS];
   Package P[MAX_THREADS];

   for (int p = 0; p < MAX_THREADS; p++) {
      P[p].L    = List;
      P[p].sIdx = p * OFFSET;
      pthread_create(&hThread[p], NULL, F, (void*) &P[p]);
   }

   Print(cout, List); // print contents of array
   pthread_exit(0);
}

void* F(void* D) {

   Package* Data = (Package*) D;

   unsigned int Base = Data->sIdx;

   for (unsigned int Idx = 0; Idx < OFFSET; Idx++) {
      unsigned int Curr = Base + Idx;
      Data->L[Curr] = Base;
   }

   pthread_exit(0);
}
```

There is no problem with the sharing of data because each thread accesses a different range of the array than the others.

However, there is nothing to make the main thread wait until all (or any) of the threads have finished before proceeding to call the Print() function. It is possible all the threads will terminate before that, but there is certainly no guarantee of it.

The problem is easily fixed by adding the following loop immediately before the call to Print():

```
for (int p = 0; p < MAX_THREADS; p++) {
   pthread_join(&hThread[p]);
}
```

6. [12 points] Consider the simple hold/ready/run/block state transition scheme for process scheduling. Draw the state diagram, showing all the possible transitions. Label each transition with a brief, precise description of an event that might cause a process to make that transition.

This is straight from the text and lectures.

7. [12 points] Consider a system that uses the SJN (shortest job next, no timesharing) strategy for scheduling jobs. Complete the table below to show the relevant statistics if the given set of jobs is executed on such a system.

Job	CPU Need	Wait time	Finish time
0	50	**20**	**70**
1	80	**70**	**150**
2	20	**0**	**20**

8. [12 points] Consider the use of virtual memory versus the use of only real (physical) memory. Briefly describe two different major advantages of using virtual memory from the perspective of the user of an operating system.

   One advantage is that most processes that are eligible to run will be allocated less physical memory than they would if only real memory were used. That means that more processes can be in the ready-run cycle at once, increasing the potential level of multi-tasking.

   Another advantage is that virtual memory makes it possible to execute processes whose address space is much larger than the amount of real memory available in the system.

---

9. Consider the execution of a process on a system that uses 32-bit addresses and demand paged virtual memory with 4096-byte pages.

   a) [4 points] What is the maximum number of frames of physical memory that the system could support? Express your answer as a power of 2.

   Since the page size equals 2^12, the page offset must be represented using 12 bits. That leaves 20 bits to represent the frame number, and so there cannot be more than 2^20 frames.

   b) [8 points] Suppose a process running on this system generated a relative address of `0x12345678`. Describe how that would be translated into the proper physical address.

   The address is first decomposed into a page number and offset; given the information above, the page number would be 0x12345 and the offset would be 0x678.

   The page number must be mapped to a frame number in which the page is actually loaded. (This may require servicing a page fault, but that is not really part of the address translation.)

   The frame number is then concatenated with the offset to obtain the proper physical address.

c) [4 points] What specialized hardware support might be provided to make the process you described in the previous part more efficient?

The only time-consuming phase is the resolution of the page number to a frame number, which requires a lookup in the page map table. In order to make this efficient, it is desirable to store key portions of the PMT for rapid access.

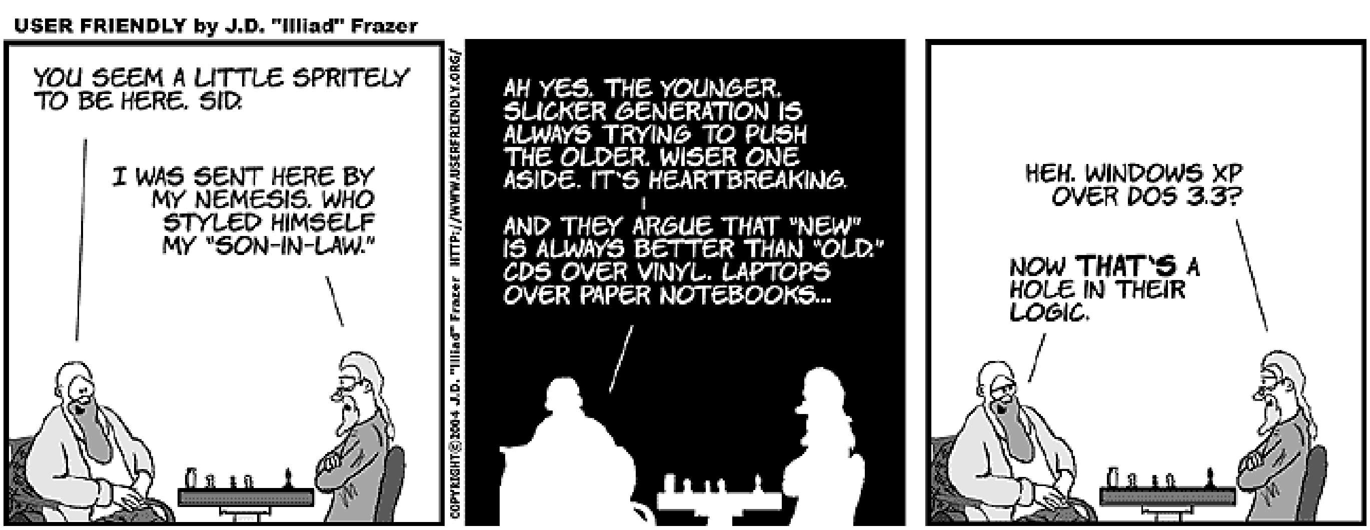
This can be achieved by providing hardware to support a parallel (associative) lookup. This is called a translation look-aside buffer. If it it managed properly, many of the page lookups will not require searching beyond the TLB.

---

10. [8 points] Suppose that the WSClock algorithm is used to manage the allocation of page frames in a virtual memory system. Assuming that the fixed window size for a process is 100, what precise statements can be made regarding the number of page frames that might actually be allocated to the process?

When the WSClock algorithm is used, it is possible that a process will own more frames than the window size would indicate, because "expired" pages are not removed from their frames immediately. Rather, they simply become eligible for replacement.

So, it is likely that at some times the process will own fewer than 100 frames, and that at some times it may own more than 100 frames.



Seemed appropriate somehow….