Operating Systems: Homework #1

Due on January 18, 2016 at 11:59pm

Professor Qu
Monday & Wednesday 3:30pm — 5:17pm

Nicholas Land

Problem 1

A computer is built that uses 15 bits for integers and for addresses.

1. How many bytes of memory can be addressed?
2. What is the range of values for signed (two's complement) integers. We want the largest magnitude negative number, and the largest magnitude positive number.

Solution

1. $2^{15} = 32768$ bytes can be addressed.
2. $-2^{14} = -16384$ is the samllest negative integer & $2^{14} - 1 = 16383$ is the largest integer.

Problem 2

Describe two of the primary motivations for having Virtual Memory in the computer system.

Solution

1. To allow efficient and safe sharing of memory among multiple programs.
2. To allow a user to exceed the size of primary memory.

Problem 3

Explain why virtual memory in a system without Translation Lookaside Buffer will be much slower than physical memory.

Solution

If a system did not contain a Translation Lookaside Buffer it would be much slower than physical memory because there would be an extra step involved in looking a page address up. Because on every reference the virtual page number is looked up in TLB, and without a TLB, it would result in a miss. A TLB miss takes about 13 clock cycles.

Problem 4

Describe what is wrong with the following function and propose modifications to fix it, submit with a test program and test results.

```
1 char *string_duplicator(char *s) {
2   char *cpy;
3   for (i = 0; i <= strlen(s); i++) {
4     cpy[i] = s[i];
5   }
6   return(cpy);
7 }
```

Problem 4 continued on next page...

Solution

The problem with the previous function was two fold; the first issue was ‘strlen‘ was part of a library that was not imported. The second issue was that the variable ’i’ was not declared.

Proposed Solution

```c
#include <stdio.h>
#include <string.h>

// Prototype so that the main function can call
// string_duplicator
char *string_duplicator(char *s);

int main(int argc, char const *argv[]) {
  char testString[] = "hello";
  printf("%s\n", string_duplicator(testString));
}

char *string_duplicator(char *s) {
  char *cpy;
  for (int i = 0; i <= strlen(s); i++) {
    cpy[i] = s[i];
  }
  return(cpy);
}
```

This produces a result of **Hello**

Problem 5