

Discussion questions: memory management 2

 web.eecs.umich.edu/~harshavm/eecs482/homework/vm2.html

Please work out these problems before your next discussion section. The GSIs and IAs will go over these problems during the discussion section.

1. Multi-level paging

A machine uses 32-bit addresses with a 3-level page table. The virtual address is divided into 4 parts as follows: 10-bits, 8-bits, 6-bits, and 8-bits, i.e., the first 10-bit are used to index into the first level, etc., and the last 8-bits as offset into the page.

Assume the valid virtual address range is the lowest 256 KB.

a. What is the page size in such a system?

b. How many page table entries are there in the first level (6-bit)?

c. How many page table entries are there in the second level (8-bits)?

d. How many page table entries are there in the third level (10-bits)?

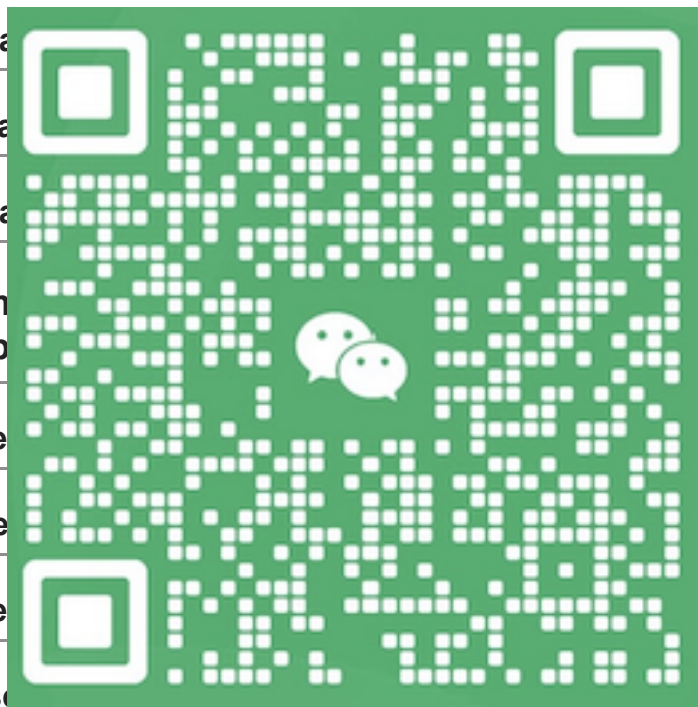
2. Now, assume the virtual address 0x60a9bb is used. What is the virtual address offset?

a. Calculate the index into the first level.

b. Calculate the index into the second level.

c. Calculate the index into the third level.

d. Calculate the offset into the page. What is the physical address translation.



3. Instruction time

If an instruction takes 10nsec and a page fault takes an additional 10 msec, then

a. Give a formula for the effective instruction time if page faults occur once every K instructions.

b. What should K be if you wanted the effective instruction time to be less than 20nsecs.

4. Page faults and program structure

Consider the following system and think about how the program structure affects performance. Your assignment is to initialize an array that has 1024 rows and 1024 columns. Page size in your system is 1024 bytes and fits 1 row of the array. Consider the following initialization routine.

Option 1:

```
for (row = 0; row < 1024; row++) {  
    for(column = 0; column < 1024; column++) {  
        array[row][column] = 0;  
    }  
}
```

Option 2:

```
for (column = 0; column < 1024; column++) {  
    for(row = 0; row < 1024; row++) {  
        array[row][column] = 0;  
    }  
}
```

You system has less than 1024 pages and you use LRU page replacement.

a. How many page faults will occur during the initialization of the array?

b. Do you think that the two loops for typical disk access patterns are different?

5. System call checks

`mprotect()` is a function that can be used to change the permissions of memory. See the `mprotect` manual page for more details.

```
int mprotect(void *addr, rlen_t len, int prot)
```

To check whether the calling procedure invoked `mprotect()` with `addr` and `len` in the arena, one might write:

```
if (addr < beginArena || addr+len > endArena) {  
    return -1;  
}
```

Are the above checks sufficient (assuming the arena is contiguous)? If not, how would you remove such a security hole?

