1. The page table and address translation
    a. Assume our processor generates 12-bit addresses
    b. However, we only have 2 KB of memory
        i. 2^{12} = 4 KB
        ii. Not enough space for multiple programs to reside in RAM at once
        iii. Might not even be enough room for *one* program
    c. Break main memory into multiple, smaller fixed-size chunks/frames
        i. Frames are physical, pages are virtual!
    d. Remap virtual addresses generated by the processor
        i. Translate these into the physical addresses that we need to access RAM
        ii. Page table tells us exactly how to make these translations
        iii. Parameters of memory above tell us how to lay out physical and virtual addresses
        iv. Virtual memory allows us to have a process with an address space larger than all of RAM!

2. Address layout example
    a. 12-bit virtual addresses
    b. Page size of 256 bytes
        i. Offset into page is just like cache offset
        ii. Offset = $\log_2 256$ = 8 bits
        iii. Frame size will be the same as page size
    c. Physical memory size = 2 KB = 2^{11} B
        i. Therefore, physical address size is $\log_2 2^{11}$ = 11 bits
    d. Layout of address below

Virtual Page Number (VPN) Whatever's left = 12 – 8 = 4 bits	**Offset into Virtual Page** $\log_2$ (page size) = 8 bits	=	**Virtual Address Size** Given, 12 bits

Physical Frame Number (PFN) Whatever's left = 11 – 8 = 3 bits	**Offset into Physical Frame** Same as page offset, 8 bits	=	**Physical Address Size** $\log_2$ (memory size) = 11 bits

    e. Offset is always same size for both addresses
        i. Same offset that we're accessing in the page is the same offset into the frame
    f. Translation is from 4-bit VPN to 3-bit PPN
        i. Page table tells us exactly what that translation should be

3. Page table access example – page fault
    a. Using same parameters as last problem, let's look at a memory and a page table
    b. Memory is 2 KB, and frame size = page size = 256 B
    c. Thus, divide memory into 8 frames of size 256 B each below
        i. PFN for each frame is equal to first three bits of each frame's address range

Memory

Physical Frame Number	Size	Address Range	Status
000	256 B	0x000 – 0x0FF	Busy
001	256 B	0x100 – 0x1FF	Busy
010	256 B	0x200 – 0x2FF	Empty
011	256 B	0x300 – 0x3FF	Busy
100	256 B	0x400 – 0x4FF	Empty
101	256 B	0x500 – 0x5FF	Process A
110	256 B	0x600 – 0x6FF	Busy
111	256 B	0x700 – 0x7FF	Empty

    d. Busy frames are already assigned to another process, empty frames are unallocated
        i. One other page from our current process A in memory right now
    e. Processor generates the following virtual address for process A
        i. 0x716 = 0111 0001 0110
    f. First, need to translate this virtual address into the physical address
        i. Look at page table below to do so
        ii. Page table is essentially a hash table
            1. Key is the VPN of the address, data is the PFN we want
            2. Page table contains $2^{\text{VPN bits}}$ number of entries = 2^4 = 16 entries here
            3. Each entry is (PFN bits) bits wide = 3 bits here

Page Table

Virtual Page Number	Status
0000	On Disk
0001	Frame 101
0010	On Disk
0011	On Disk
0100	On Disk
0101	On Disk
0110	On Disk
0111	On Disk
1000	On Disk
1001	On Disk
1010	On Disk
1011	On Disk
1100	On Disk
1101	On Disk
1110	On Disk
1111	On Disk

g. Use the first four bits of the address 0x716 above
    i. For entry 0111, the page is on the disk
    ii. We have a page fault, the page is not in RAM
h. On a page fault, OS gets called in
    i. Must go out and get the page from memory
    ii. Then needs to decide where to place it
    iii. Selects empty frame 010 since that was the first one it found that was free
    iv. Copy the data from the disk into that frame in RAM
    v. Update page table and list of frames

Memory

Physical Frame Number	Size	Address Range	Status
000	256 B	0x000 – 0x0FF	Busy
001	256 B	0x100 – 0x1FF	Busy
010	256 B	0x200 – 0x2FF	Process A
011	256 B	0x300 – 0x3FF	Busy
100	256 B	0x400 – 0x4FF	Empty
101	256 B	0x500 – 0x5FF	Process A
110	256 B	0x600 – 0x6FF	Busy
111	256 B	0x700 – 0x7FF	Empty

Page Table

Virtual Page Number	Status
0000	On Disk
0001	Frame 101
0010	On Disk
0011	On Disk
0100	On Disk
0101	On Disk
0110	On Disk
0111	Frame 010
1000	On Disk
1001	On Disk
1010	On Disk
1011	On Disk
1100	On Disk
1101	On Disk
1110	On Disk
1111	On Disk

i. Now our memory has the page we requested in frame 010
    i. Our page table has been updated to match
j. Use page table to convert address
    i. Take 4-bit VPN, replace with 3-bit PFN
    ii. Virtual address 0x716 = 0111 0001 0110
        1. Replace first 4 bits with 3-bit sequence 010
        2. Offset stays the same
    iii. Thus, corresponding physical address = 0x216 = 010 0001 0110
k. Now that we have the physical address, go out to cache/RAM to retrieve data at that location
    i. Completely possible to have a page fault, then a cache miss
    ii. This could even occur for the page table, since the page table needs to be stored in RAM too!