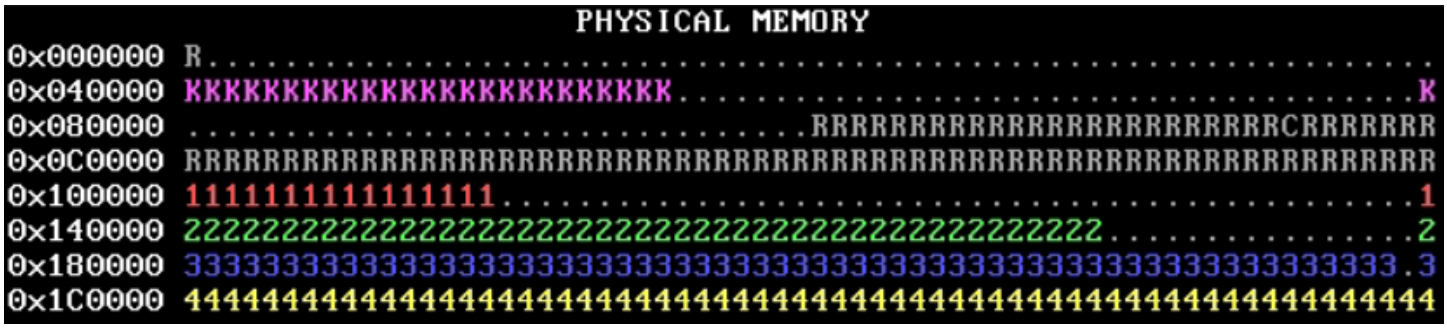
DISCUSSION A. Eve wants to force Alice to waste her processor time by executing an infinite loop. She wants her process to edit the code segment of Alice's process to only have the instruction `eb fe`. How will virtual memory stop Eve's plan?

DISCUSSION B. Consider this implementation of WeensyOS that doesn't use virtual memory. Process 4 has run out of contiguous address space, yet there are still empty physical pages.



How can using virtual memory mappings allow us to allocate additional memory for process 4?

QUESTION 1B. How many levels of pagetables are there in the x86-64 architecture?

QUESTION 1C. We say that page tables are a "sparse" data structure. What does "sparse" mean and why do we want "sparse" data structures?

QUESTION 1D. When translating addresses, how does the translation hardware know where to find the current process's L4 page table?

QUESTION 1E. Translate the following virtual address to its corresponding physical address, using the information below

```
0x 0000 7f46 9ae0 5445
```

```
0b 0000 0000 0000 0000 0111 1111 0100 0110 1001 1010 1110 0000 0101 0100 0100 0101
```

```
L4 page table

index | L3 pagetable address
------+---------------------
0     | 0
...   | ...
126   | 0
127   | 0x9020000
128   | 0x5000000
...   |
253   | 0x3900000
254   | 0x3000000
255   | 0
...   |
```

```
L3 page table at 0x3000000

index | L2 pagetable address
------+---------------------
0     | 0
...   | ...
140   | 0
141   | 0x2220000
142   | 0x9400000
...   |
280   | 0x8100000
281   | 0
282   | 0x9007000
...   |
```

0b 0000 0000 0000 0000 0111 1111 0100 0110 1001 1010 1110 0000 0101 0100 0100 0101

L2 page table at 0x2220000

index	L1 pagetable address
0	0
...	...
200	0x4444000
201	0x3400000
202	0x1200000
...	
215	0x9099000
216	0x1240000
217	0x1230000
...	

L2 page table at 0x9007000

index	L1 pagetable address
0	0
...	...
106	0x7800000
107	0x2210000
108	0x2208000
...	
215	0x4500000
216	0x7800000
217	0x7801000
...	

0b 0000 0000 0000 0000 0111 1111 0100 0110 1001 1010 1110 0000 0101 0100 0100 0101

L1 page table at 0x9099000

index	contents
0	0
...	...
3	0b 1000 0000 0000 0000 0000 1011 1101 1111 1111 0101 0111 1000 1101 0000 0000 0101
4	0b 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
5	0b 1000 0000 0000 0000 0000 1011 1101 1111 1111 0101 0111 1000 1101 0000 0000 0111
...	
129	0b 1000 0000 0000 0000 0000 1001 1100 1010 1011 0001 0101 1100 0001 0000 0000 0111
130	0b 1000 0000 0000 0000 0000 0010 1100 0110 1111 0001 0100 1000 0101 0000 0000 0011
131	0b 1000 0000 0000 0000 0000 0100 1111 0010 1011 0101 0001 1010 0001 0000 0000 0111
...	

L1 page table at 0x4500000

index	contents
0	0
...	...
3	0b 1000 0000 0000 0000 0000 1010 1100 1110 1011 0001 0100 1100 0101 0000 0000 0111
4	0b 1000 0000 0000 0000 0000 1010 1100 0010 1011 0001 0100 1111 0101 0000 0000 0111
5	0b 1000 0000 0000 0000 0000 1010 1100 0010 1011 0001 0000 1000 0101 0000 0000 0111
...	
129	0b 1000 0000 0000 0000 0000 0000 1100 1010 1011 0001 0100 1100 0101 0000 0000 0111
130	0b 1000 0000 0000 0000 0000 0110 1100 0010 1111 0001 0100 1100 0101 0000 0000 0111
131	0b 1000 0000 0000 0000 0000 1010 1100 0010 1011 0101 0000 1000 0001 0000 0000 0111
...	

0b 0000 0000 0000 0000 0111 1111 0100 0110 1001 1010 1110 0000 0101 0100 0100 0101

QUESTION 2A. How many unique 4 byte addresses can a 4 KB (4096 byte) page hold?

QUESTION 2B. How many bits would we need to index into each address for a page table that stores 4 byte addresses?

QUESTION 2C. Consider the following 4 byte virtual address:

0x 9ae0 5445

Its binary expansion is:

0b 1001 1010 1110 0000 0101 0100 0100 0101

Using two levels of pagetables, how are each of the bits in the above virtual address used in translating a 4 byte virtual address to the corresponding 4 byte physical address?