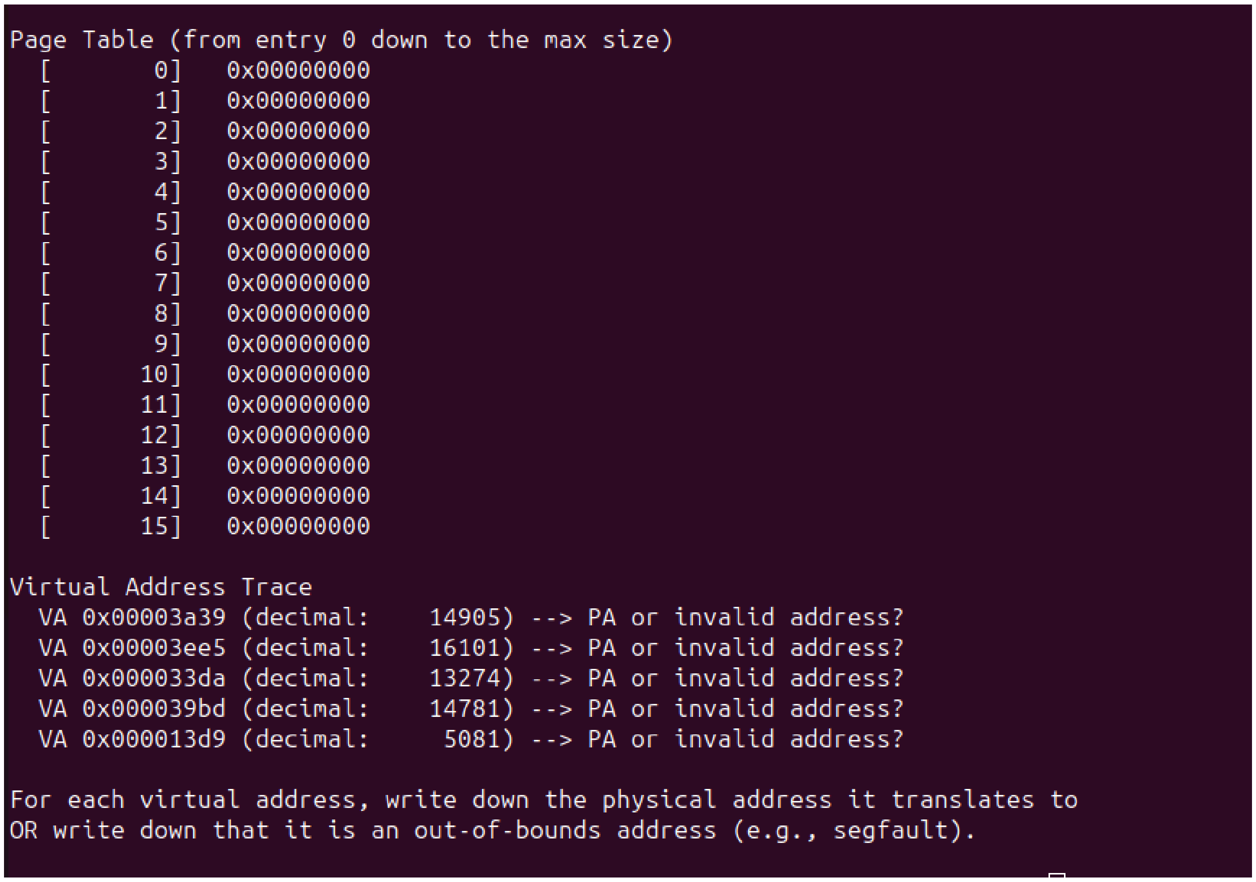
Ch.18 Simulation hw
Tyler Holmquist

1. As the address space grows page table size will grow as well since it requires more pages to store the memory needed. As page sizes grow our page tables will shrink because we are able to store more information on each page. But we don't want excessively large pages because that can lead to a lot of wasted memory when a large page isn't getting utilized fully.
2.

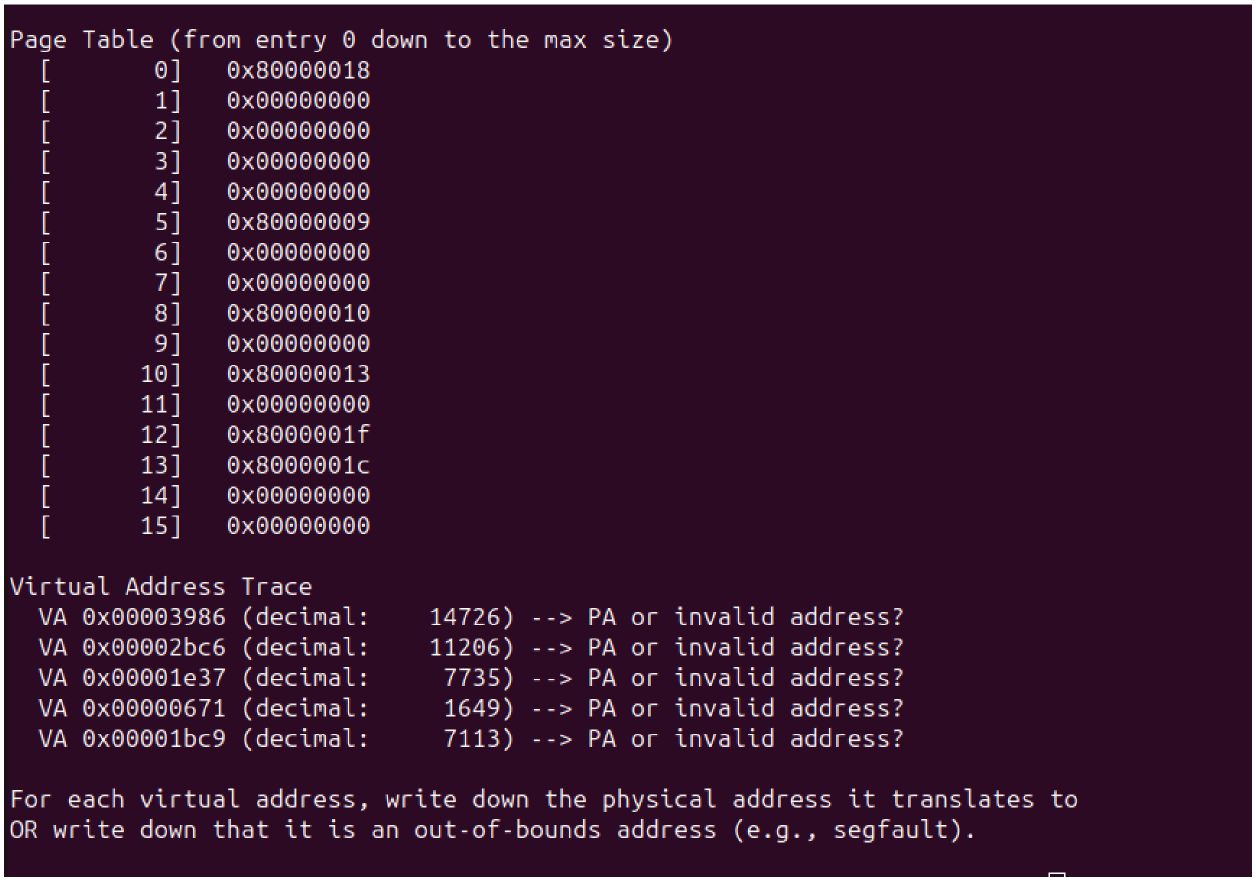
   python paging-linear-translate.py -P 1k -a 16k -p 32k -v -u 0 -c



```
Page Table (from entry 0 down to the max size)
  [       0]   0x00000000
  [       1]   0x00000000
  [       2]   0x00000000
  [       3]   0x00000000
  [       4]   0x00000000
  [       5]   0x00000000
  [       6]   0x00000000
  [       7]   0x00000000
  [       8]   0x00000000
  [       9]   0x00000000
  [      10]   0x00000000
  [      11]   0x00000000
  [      12]   0x00000000
  [      13]   0x00000000
  [      14]   0x00000000
  [      15]   0x00000000

Virtual Address Trace
  VA 0x00003a39 (decimal:    14905) --> PA or invalid address?
  VA 0x00003ee5 (decimal:    16101) --> PA or invalid address?
  VA 0x000033da (decimal:    13274) --> PA or invalid address?
  VA 0x000039bd (decimal:    14781) --> PA or invalid address?
  VA 0x000013d9 (decimal:     5081) --> PA or invalid address?

For each virtual address, write down the physical address it translates to
OR write down that it is an out-of-bounds address (e.g., segfault).
```

   All are invalid since no entries in the table are valid.

python paging-linear-translate.py -P 1k -a 16k -p 32k -v -u 25

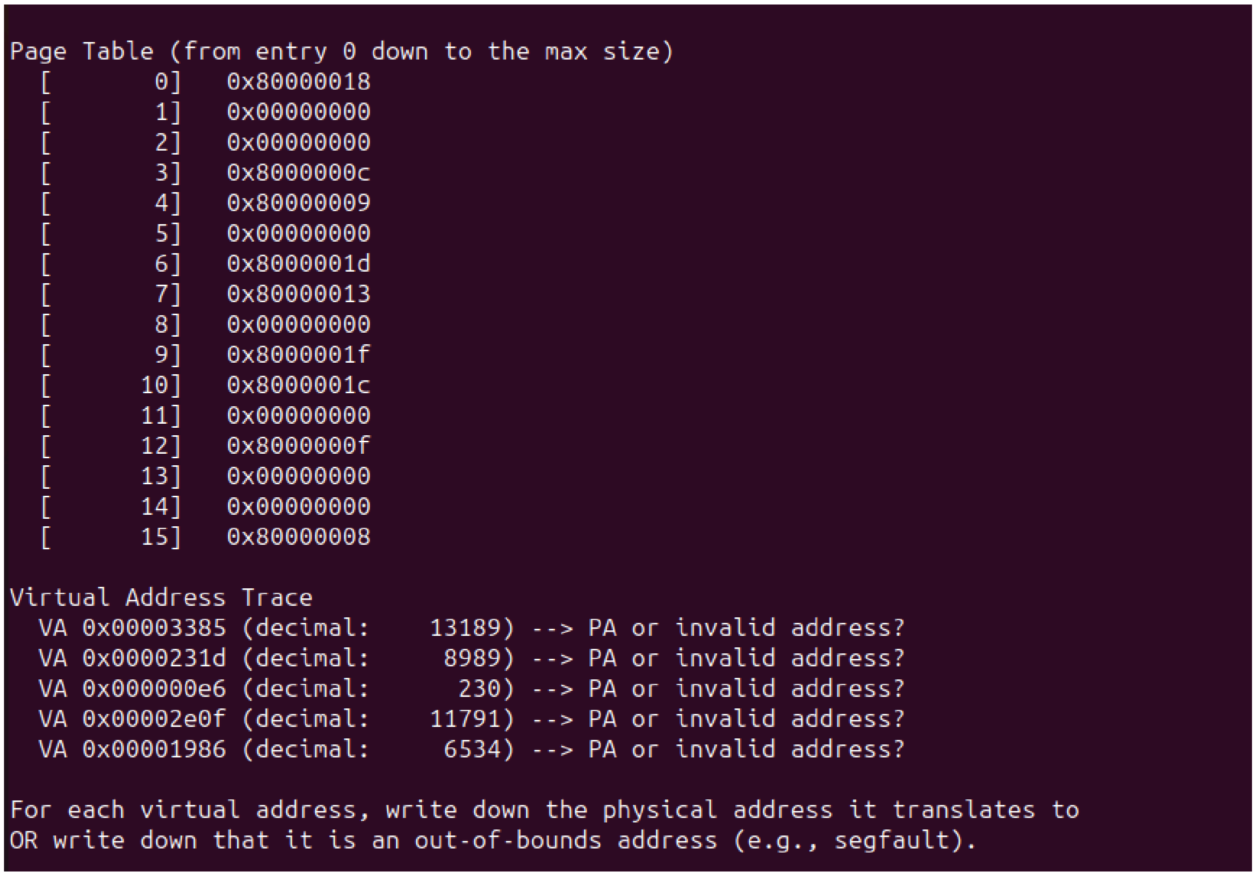


```
Page Table (from entry 0 down to the max size)
  [       0]   0x80000018
  [       1]   0x00000000
  [       2]   0x00000000
  [       3]   0x00000000
  [       4]   0x00000000
  [       5]   0x80000009
  [       6]   0x00000000
  [       7]   0x00000000
  [       8]   0x80000010
  [       9]   0x00000000
  [      10]   0x80000013
  [      11]   0x00000000
  [      12]   0x8000001f
  [      13]   0x8000001c
  [      14]   0x00000000
  [      15]   0x00000000

Virtual Address Trace
  VA 0x00003986 (decimal:    14726) --> PA or invalid address?
  VA 0x00002bc6 (decimal:    11206) --> PA or invalid address?
  VA 0x00001e37 (decimal:     7735) --> PA or invalid address?
  VA 0x00000671 (decimal:     1649) --> PA or invalid address?
  VA 0x00001bc9 (decimal:     7113) --> PA or invalid address?

For each virtual address, write down the physical address it translates to
OR write down that it is an out-of-bounds address (e.g., segfault).
```

The only valid VA here is 0x2bc6 because in binary that is 1010 | 1111000110 the first four bits give a VPN of 10 and ten is valid with a value of 0x13. 0x13 in binary is 10011 and left shifted 10 times for the VPN is 0000001111000110. Then OR'd with the offset from 0x2bc6 becomes 0100111111000110 or 0x4fc6 which would be the physical address.

The rest of the VA's are invalid since the VPN's they are attached to are invalid.

python paging-linear-translate.py -P 1k -a 16k -p 32k -v -u 50



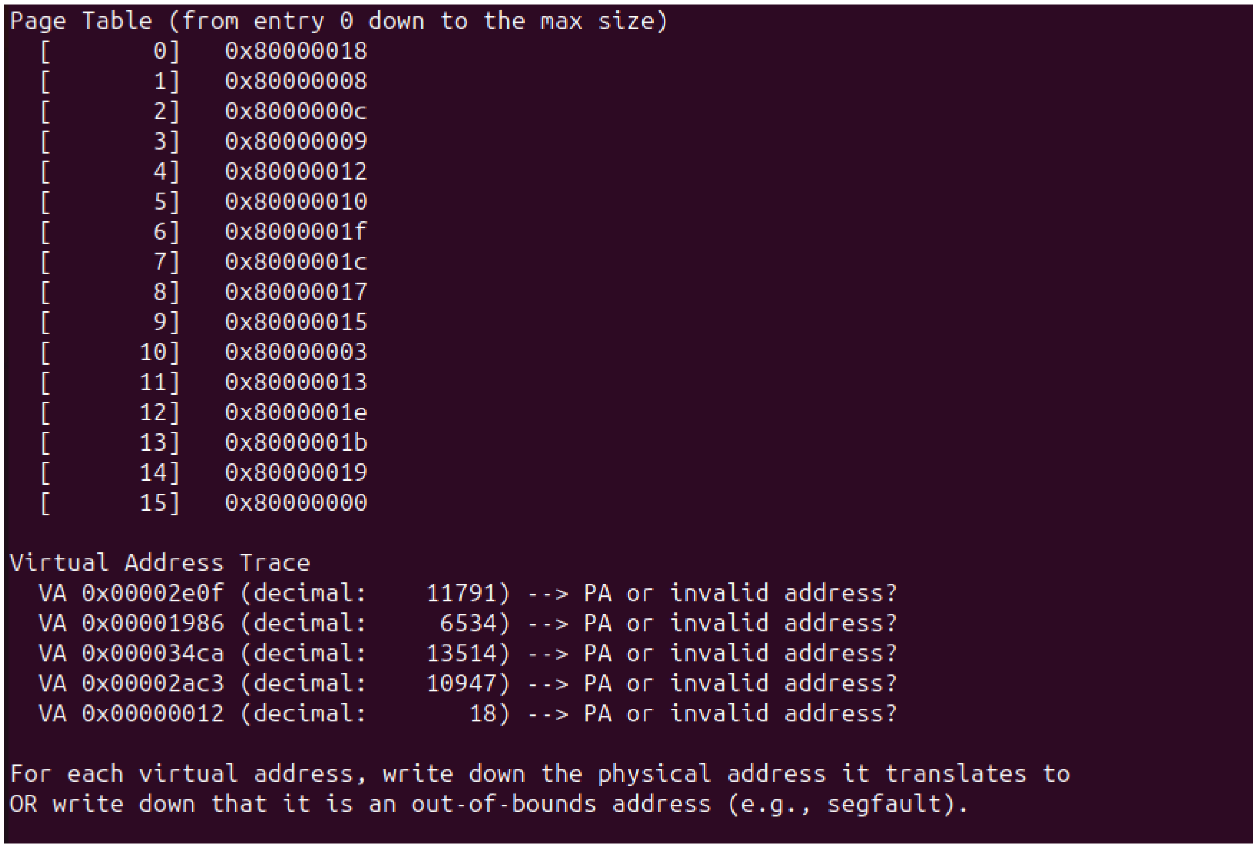
```
Page Table (from entry 0 down to the max size)
  [       0]   0x80000018
  [       1]   0x00000000
  [       2]   0x00000000
  [       3]   0x8000000c
  [       4]   0x80000009
  [       5]   0x00000000
  [       6]   0x8000001d
  [       7]   0x80000013
  [       8]   0x00000000
  [       9]   0x8000001f
  [      10]   0x8000001c
  [      11]   0x00000000
  [      12]   0x8000000f
  [      13]   0x00000000
  [      14]   0x00000000
  [      15]   0x80000008

Virtual Address Trace
  VA 0x00003385 (decimal:    13189) --> PA or invalid address?
  VA 0x0000231d (decimal:     8989) --> PA or invalid address?
  VA 0x000000e6 (decimal:      230) --> PA or invalid address?
  VA 0x00002e0f (decimal:    11791) --> PA or invalid address?
  VA 0x00001986 (decimal:     6534) --> PA or invalid address?

For each virtual address, write down the physical address it translates to
OR write down that it is an out-of-bounds address (e.g., segfault).
```

0x3385 would be valid in VPN 12 and with a physical address of 0x3f85 since 0000001110000101 OR'd with 0011110000000000 is 0011111110000101.

0xe6 would be valid in VPN 0 with a PA of 0x60e6 since 0000000011100110 OR'd with 0110000000000000 is 0110000011100110.

0x198 would be valid in VPN 6 with a PA of 0x7586 because 0000000110000110 OR'd with 0111010000000000 is 0111010110000110.

python paging-linear-translate.py -P 1k -a 16k -p 32k -v -u 75



```
Page Table (from entry 0 down to the max size)
  [       0]   0x80000018
  [       1]   0x80000008
  [       2]   0x8000000c
  [       3]   0x80000009
  [       4]   0x80000012
  [       5]   0x80000010
  [       6]   0x8000001f
  [       7]   0x8000001c
  [       8]   0x80000017
  [       9]   0x80000015
  [      10]   0x80000003
  [      11]   0x80000013
  [      12]   0x8000001e
  [      13]   0x8000001b
  [      14]   0x80000019
  [      15]   0x80000000

Virtual Address Trace
  VA 0x00002e0f (decimal:    11791) --> PA or invalid address?
  VA 0x00001986 (decimal:     6534) --> PA or invalid address?
  VA 0x000034ca (decimal:    13514) --> PA or invalid address?
  VA 0x00002ac3 (decimal:    10947) --> PA or invalid address?
  VA 0x00000012 (decimal:       18) --> PA or invalid address?

For each virtual address, write down the physical address it translates to
OR write down that it is an out-of-bounds address (e.g., segfault).
```

0x2e0f is valid in VPN 11 with a PA of 0x4e0f because 0100110000000000 OR’d with 0000001000001111 is 0100111000001111.

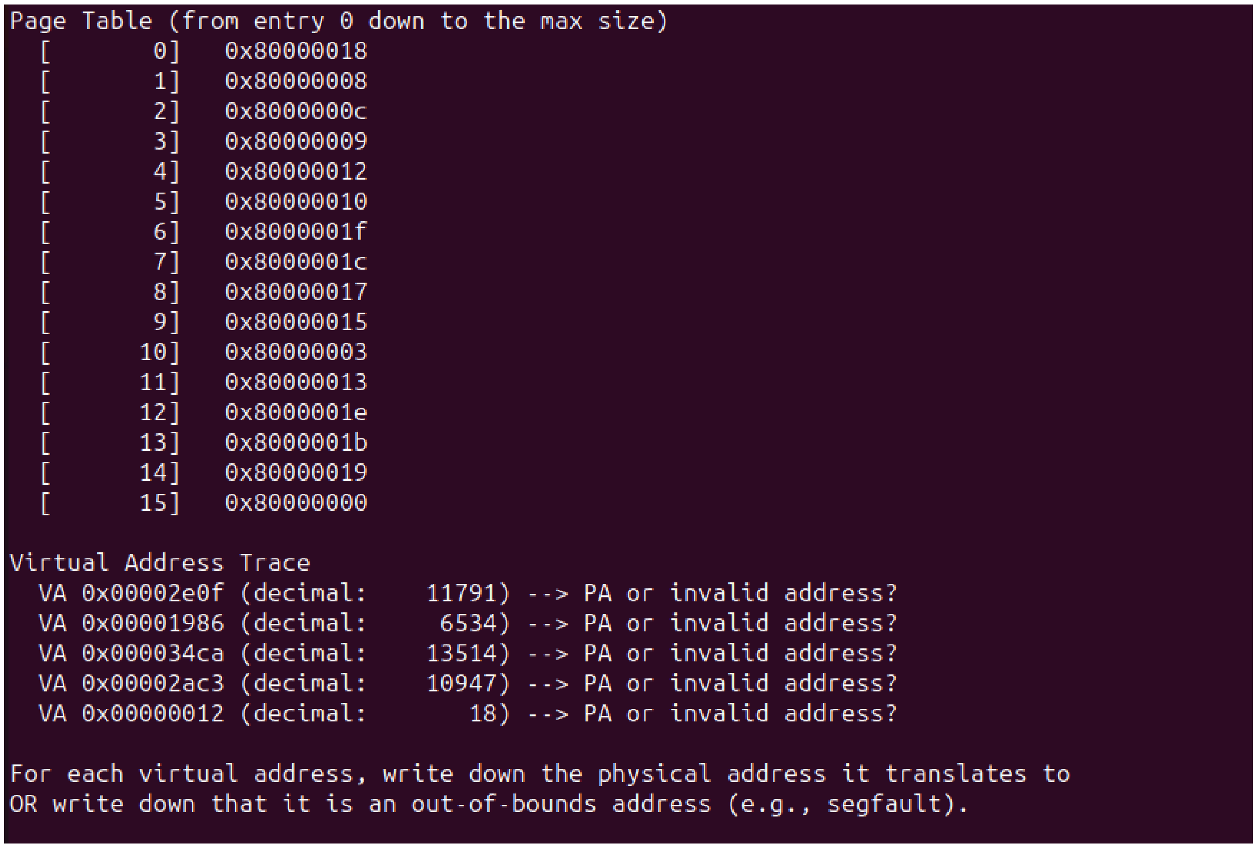
0x1986 is valid in VPN 6 with a PA of 0x7d86 because 0111110000000000 OR’d with 0000000110000110 is 0111110110000110.

0x34ca is valid in VPN 13 with a PA of 0c6cca because 0110110000000000 OR’d with 0000000011001010 is 0110110011001010.

0x2ac3 is valid in VPN 10 with a PA of 0xec3 because 110000000000 OR’d with 001011000011 is 111011000011.

0x12 is valid in VPN 0 with a PA of 0x6012 because 0110000000000000 OR’d with 0000000000010010 is 0110000000010010.

python paging-linear-translate.py -P 1k -a 16k -p 32k -v -u 100

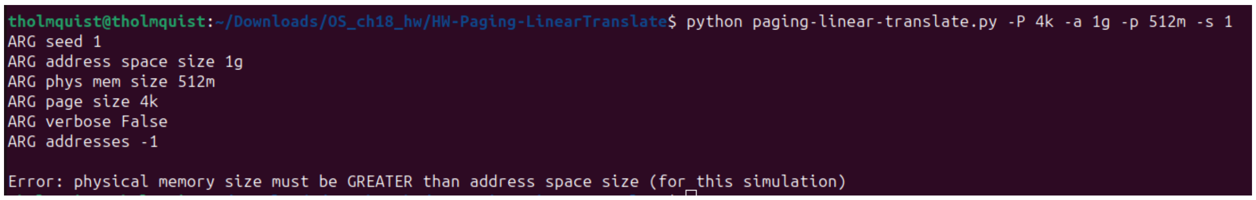


```
Page Table (from entry 0 down to the max size)
  [       0]   0x80000018
  [       1]   0x80000008
  [       2]   0x8000000c
  [       3]   0x80000009
  [       4]   0x80000012
  [       5]   0x80000010
  [       6]   0x8000001f
  [       7]   0x8000001c
  [       8]   0x80000017
  [       9]   0x80000015
  [      10]   0x80000003
  [      11]   0x80000013
  [      12]   0x8000001e
  [      13]   0x8000001b
  [      14]   0x80000019
  [      15]   0x80000000

Virtual Address Trace
  VA 0x00002e0f (decimal:    11791) --> PA or invalid address?
  VA 0x00001986 (decimal:     6534) --> PA or invalid address?
  VA 0x000034ca (decimal:    13514) --> PA or invalid address?
  VA 0x00002ac3 (decimal:    10947) --> PA or invalid address?
  VA 0x00000012 (decimal:       18) --> PA or invalid address?

For each virtual address, write down the physical address it translates to
OR write down that it is an out-of-bounds address (e.g., segfault).
```

With the number of pages set to 100 it is the same result as being set to 75.

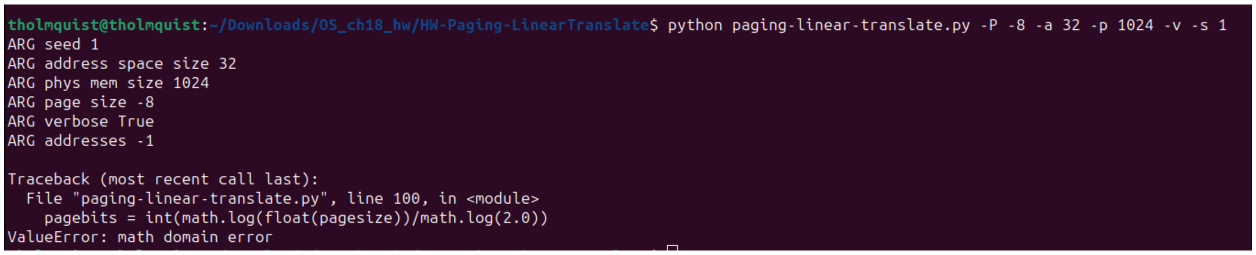
This shows that as the number of pages increases more virtual addresses become valid. But the usage of memory increases and once you hit a certain point for example 75 to 100 the benefits aren't seen anymore and memory is just being wasted with no gain.

3. The unrealistic parameters are -P 8 -a 32 -p 1024 -v -s 1, because the page sizes and address space size are much too small.
4.



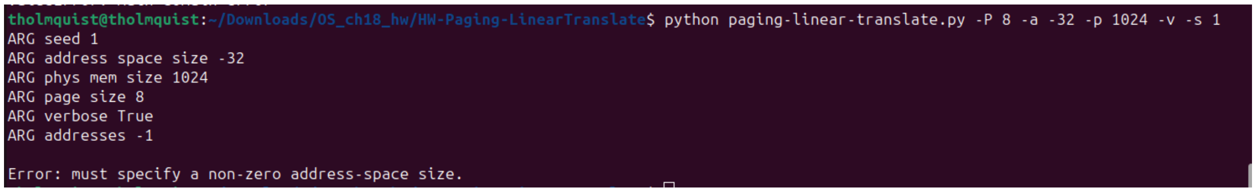
```
tholmquist@tholmquist:~/Downloads/OS_ch18_hw/HW-Paging-LinearTranslate$ python paging-linear-translate.py -P 4k -a 1g -p 512m -s 1
ARG seed 1
ARG address space size 1g
ARG phys mem size 512m
ARG page size 4k
ARG verbose False
ARG addresses -1

Error: physical memory size must be GREATER than address space size (for this simulation)
```

Address space size is greater than physical memory size, it throws an error.



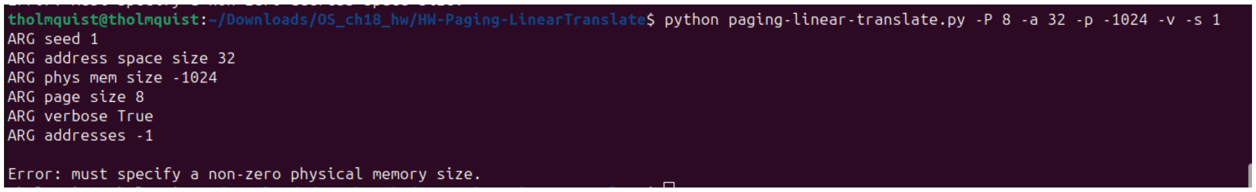
```
tholmquist@tholmquist:~/Downloads/OS_ch18_hw/HW-Paging-LinearTranslate$ python paging-linear-translate.py -P -8 -a 32 -p 1024 -v -s 1
ARG seed 1
ARG address space size 32
ARG phys mem size 1024
ARG page size -8
ARG verbose True
ARG addresses -1

Traceback (most recent call last):
  File "paging-linear-translate.py", line 100, in <module>
    pagebits = int(math.log(float(pagesize))/math.log(2.0))
ValueError: math domain error
```

Page size is negative



```
tholmquist@tholmquist:~/Downloads/OS_ch18_hw/HW-Paging-LinearTranslate$ python paging-linear-translate.py -P 8 -a -32 -p 1024 -v -s 1
ARG seed 1
ARG address space size -32
ARG phys mem size 1024
ARG page size 8
ARG verbose True
ARG addresses -1

Error: must specify a non-zero address-space size.
```

Address space size is negative



```
tholmquist@tholmquist:~/Downloads/OS_ch18_hw/HW-Paging-LinearTranslate$ python paging-linear-translate.py -P 8 -a 32 -p -1024 -v -s 1
ARG seed 1
ARG address space size 32
ARG phys mem size -1024
ARG page size 8
ARG verbose True
ARG addresses -1

Error: must specify a non-zero physical memory size.
```

Physical memory size is negative