VM paging

./paging-linear-translate.py -v

Page Table

[0] 0x8000000c

[1] 0x00000000

[2] " 0

[3] 0x80000006

VA trace

VA 0x00003229 (decimal: 12841) → PA or invalid address

VPN and offset

asize = 16KB

psize = 4KB

14 bits VA → 12 bits offset + 2 bits VPN

0x3229 → 0b11001000101001

VPN

$= 3 \longrightarrow 6$ PFN (from page table)

0x6000 OR 0x0229 (offset) $=$ 0x6229 valid

VA 0x1369

VPN = b01 = 1 $\rightarrow$ 0x0 invalid

VA 0x1e80

VPN = b01 = 1 $\Rightarrow$ 0x0 invalid

VA 0x2556

VPN = b10 = 2 $\rightarrow$ 0x0 invalid

VA 0x3a1e

VPN = b11 = 3 $\Rightarrow$ 0x6000 valid

0X6000 OR 0xa1e = 0x6a1e ✓

① asize ↑ page-table size ↑ since page size is same

asize=1m

```
[      1012]   0x8001d1ab
[      1013]   0x8007df94
[      1014]   0x800052d0
[      1015]   0x00000000
[      1016]   0x00000000
[      1017]   0x00000000
[      1018]   0x00000000
[      1019]   0x8002e9c9
[      1020]   0x00000000
[      1021]   0x00000000
[      1022]   0x00000000
[      1023]   0x00000000
```

asize=2m

```
[      2042]   0x00000000
[      2043]   0x00000000
[      2044]   0x00000000
[      2045]   0x00000000
[      2046]   0x8000eedd
[      2047]   0x00000000
```

asize=4m

```
[      4091]   0x800160f8
[      4092]   0x80015abc
[      4093]   0x8001483a
[      4094]   0x00000000
[      4095]   0x8002e298
```

Page size increase ↑, page table size ↓

p = 1k

```
[      1020]   0x00000000
[      1021]   0x00000000
[      1022]   0x00000000
[      1023]   0x00000000
```

p = 2k

```
[       507]   0x00000000
[       508]   0x8001a7f2
[       509]   0x8001c337
[       510]   0x00000000
[       511]   0x00000000
```

p = 4k

```
[       250]   0x00000000
[       251]   0x8001efec
[       252]   0x8001cd5b
[       253]   0x800125d2
[       254]   0x80019c37
[       255]   0x8001fb27
```

Using big pages we don't use in general because

① Need more time to load from disk → memory

② Takes more space in memory, leading to more quickly memory fill up potentially leading to quicker eviction of pages.

2 asize = 16k = $2^4 \times 2^{10}$ = 2^{14}
psize = 1k

14 bits ⇒ $\frac{\text{asize}}{\text{psize}} = 16 = 2^4$ pages

4 bits + 10 bits
VPN offset

2+4+8

VA 0x3986 ⇒ 0b$\underbrace{1110}$01110000110

VPN = 14 ⇒ 0x00 (page table)
invalid

VA 0x2bc6 ⇒ 0b$\underbrace{1010}$1111000110

VPN = 2+8 = 10 ⇒ 0x1300
Offset = 0x3c6
valid

PA = PFN × Page size + offset
= 0x4fc6

VA 0x1e37 ⇒ 0b$\underbrace{0111}$1000110111

VPN = 1+2+4 = 7 ⇒ 0x00

invalid x

VA 0x671 ⇒ 0b $\underbrace{0001}$1001110001

↓

VPN = 1 ⇒ 0x00

invalid

VA 0x1bc9 ⇒ 0b $\underbrace{0110}$1111001001

↓

VPN = 2+4 = 6 ⇒ 0x00

invalid x

with u = 50

VA = 0x3385 ⇒ 0b $\boxed{1100}$1110000101

↓

VPN = 4+8 = 12

⇓

0xf

valid

offset = 0x385

$PA = \text{psize} \times PFN + \text{offset}$
$= 0x4 \times 0xf + 0x385$
$= 0x3f85$

$VA = 0x231d \Rightarrow 0b\underbrace{1000}_{\downarrow}1100\,0\,11101$

$VPN = 8 \Rightarrow 0x0$

invalid

$VA = 0xe6 \Rightarrow 0b\underbrace{0000}_{0}0011100110$

$VPN = 0 \Rightarrow 0x18$

$\text{offset} = 0xe6$

$PA = \text{psize} \times PFN + \text{offset}$

~~left~~size by number of offset bits to multiply

just index

$= \underbrace{PFN \ll 10}_{\text{base addr}} + \text{offset}$

$= 0x60e6$

VA 0x2e0f $\Rightarrow$ 0b$\underbrace{1011}$1000001111

VPN = 1+2+8 = 11

$\Downarrow$

0x00

invalid ×

VA 0x1986 $\Rightarrow$ 0b$\underbrace{0110}$0110000110

VPN = 2+4 = 6

offset = 0x186

$\Downarrow$

0x1d

PA = PFN<<10 + offset

= 0x7586

```
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) --> 00004e0f (decimal    19983) [VPN 11]
  VA 0x00001986 (decimal:     6534) --> 00007d86 (decimal    32134) [VPN 6]
  VA 0x000034ca (decimal:    13514) --> 00006cca (decimal    27850) [VPN 13]
  VA 0x00002ac3 (decimal:    10947) --> 00000ec3 (decimal     3779) [VPN 10]
  VA 0x00000012 (decimal:       18) --> 00006012 (decimal    24594) [VPN 0]
```

```
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) --> Invalid (VPN 14 not valid)
  VA 0x00002bc6 (decimal:    11206) --> 00004fc6 (decimal    20422) [VPN 10]
  VA 0x00001e37 (decimal:     7735) --> Invalid (VPN 7 not valid)
  VA 0x00000671 (decimal:     1649) --> Invalid (VPN 1 not valid)
  VA 0x00001bc9 (decimal:     7113) --> Invalid (VPN 6 not valid)
```

u=25

```
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) --> Invalid (VPN 14 not valid)
  VA 0x00003ee5 (decimal:    16101) --> Invalid (VPN 15 not valid)
  VA 0x000033da (decimal:    13274) --> Invalid (VPN 12 not valid)
  VA 0x000039bd (decimal:    14781) --> Invalid (VPN 14 not valid)
  VA 0x000013d9 (decimal:     5081) --> Invalid (VPN 4 not valid)
```

u=0

```
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) --> 00004e0f (decimal    19983) [VPN 11]
  VA 0x00001986 (decimal:     6534) --> 00007d86 (decimal    32134) [VPN 6]
  VA 0x000034ca (decimal:    13514) --> 00006cca (decimal    27850) [VPN 13]
  VA 0x00002ac3 (decimal:    10947) --> 00000ec3 (decimal     3779) [VPN 10]
  VA 0x00000012 (decimal:       18) --> 00006012 (decimal    24594) [VPN 0]
```

u=100

With increase of allocated space used,

seg voilation reduce.

3 -P 8 -a 32 -p1024

psize = 8 physical memory = 1024

asize = 32 = 2^5

VA → 5 bits

$$\frac{asize}{psize} = \frac{32}{8} = 4 = 2^2$$

pages

2 bits + 3 bits

VPN offset

-P 8k -a 32k -p 1m

psize = 8k , asize = 32k = $2^5 \times 2^{10}$

physical memory = 1m

2 bits + 13 bits

VPN offset

-P 1m -a 256m -p 512m

$\frac{256}{1} = 2^8 = 8$ bits

$256m = 2^8 \times 2^{10} \times 2^{10}$

8 bits + 20 bits
VPN offset

The most unrealistic seems to be the last one with page size = 1mb, mostly smaller page size are more used and standard. 4Kb is that value.
Similary page size = 8 B is very small leading to more pages needed to fetch even little info.
PTE ⇒ Page table entry that tracks the page, itself is 4-8B in size. Memory wasted per page:

with page size that is good that
same PTE size can track the page
more efficiently memory usage wise.

④ -u=used ⇒ percent of virtual
address space that is used
basically how many virtual addresses
have valid physical memory mapping

u = 75%. I was expecting 25%
VPN to be invalid but none were
according to the simulator.
- Then realised that it is possible that
within VPNs the unused percent
across is 25% on aggregate, leading
to pages overall being valid.

memory alignment, leaving unused space

- Also looked into the code it uses
random generate along with the "used" param
to mark page as used or unused overall.

Hence with smaller page size = 512
I got whole invalid pages

```
./paging-linear-translate.py -P 512 -a 16k -p 32k -v -u 75
```

```
[       15]  0x80000000
[       16]  0x8000002a
[       17]  0x00000000
[       18]  0x00000000
[       19]  0x00000000
[       20]  0x8000000f
[       21]  0x80000005
[       22]  0x8000002d
[       23]  0x80000034
[       24]  0x80000026
[       25]  0x80000024
[       26]  0x8000000c
[       27]  0x00000000
```

Similarly with slightly lower u = 73
I got couple of pages

```
./paging-linear-translate.py -P 1k -a 16k -p 32k -v -u 73
```

```
[         6]  0x8000001f
[         7]  0x8000001c
[         8]  0x80000017
[         9]  0x80000015
[        10]  0x80000003
[        11]  0x80000013
[        12]  0x8000001e
[        13]  0x8000001b
[        14]  0x00000000
[        15]  0x80000011
```

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

```
Error: must use smaller sizes (less than 1 GB) for this simulation.
```