Kyle Kloberdanz

1)

As the address space grows, so too must the page table, because each location of the address space must be mapped in the page table

Larger page sizes means that the page table will be smaller.

Larger pages means that the size of the page table can be smaller, and would then appear at first to use memory more efficiently, but larger pages also have the potential to waste memory, because larger pages allow larger chunks of unused memory within each page.

2)

Command: paging-linear-translate.py -P 1k -a 16k -p 32k -v -u 0
16KB AS → 14 bit VA, 1KB Page Size → 10 bit offset

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?
→ Invalid address, valid bit = 0

VA 0x00003ee5 (decimal: 16101) --> PA or invalid address?
→ Invalid address, valid bit = 0

VA 0x000033da (decimal: 13274) --> PA or invalid address?
→ Invalid address, valid bit = 0

VA 0x000039bd (decimal: 14781) --> PA or invalid address?
→ Invalid address, valid bit = 0

VA 0x000013d9 (decimal: 5081) --> PA or invalid address?
→ Invalid address, valid bit = 0

Command: 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) -->  Invalid (VPN 14 not valid)
     VA  = 0011 1001 1000 0110
     VPN = 3
     PFN = Invalid!

  VA 0x00002bc6 (decimal:    11206) --> 00004fc6 (decimal    20422) [VPN 10]
     VA  = 0010 1011 1100 0110
     VPN = 0xa = 10
     PFN = 0x13
     PA  = 100 1111 1100 0110 = 0x4fc6

  VA 0x00001e37 (decimal:     7735) -->  Invalid (VPN 7 not valid)
     VA  = 001 1110 0011 0111
     VPN = 7
     PFN = Invalid!

  VA 0x00000671 (decimal:     1649) -->  Invalid (VPN 1 not valid)
     VA  = 0110 0111 0001
     VPN = 1
     PFN = Invalid!

  VA 0x00001bc9 (decimal:     7113) -->  Invalid (VPN 6 not valid)
     VA  = 0001 1011 1100 1001
     VPN = 6
     PFN = Invalid!
```

Command: 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]   0x80000009
  [       8]   0x80000013
  [       9]   0x00000000
  [      10]   0x8000001f
  [      11]   0x8000001c
  [      12]   0x00000000
  [      13]   0x8000001c
  [      14]   0x8000000f
  [      15]   0x00000000
```

Virtual Address Trace
```
  VA 0x00001bc9 (decimal:     7113) --> 000077c9 (decimal    30665) [VPN 6]
     VA  = 0110 1011 1100 1001
     VPN = 6
     PFN = 0x1d
     PA  = 0111 0111 1100 1001 = 0x77c9

  VA 0x00002718 (decimal:    10008) -->  Invalid (VPN 9 not valid)
     VA  = 0010 0111 0001 1000
     VPN = 9
     PFN = Invalid!

  VA 0x00003a6e (decimal:    14958) --> 00003e6e (decimal    15982) [VPN 14]
     VA  = 0011 1010 0110 1110
     VPN = 14
     PFN = 0xf
     PA  = 0011 1110 0110 1110 = 0x3e6e

  VA 0x00003ddc (decimal:    15836) -->  Invalid (VPN 15 not valid)
     VA  = 0011 1101 1101 1100
     VPN = 15
     PFN = Invalid!

  VA 0x00001e87 (decimal:     7815) --> 00002687 (decimal     9863) [VPN 7]
     VA  = 0001 1110 1000 0111
     VPN = 7
     PFN = 0x9
     PA  = 0010 0110 1000 0111 = 0x2687
```

Command: 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]   0x80000018
  [7]   0x80000008
  [8]   0x8000001f
  [9]   0x8000001c
  [10]   0x80000017
  [11]   0x80000015
  [12]   0x80000003
  [13]   0x80000013
  [14]   0x8000001e
  [15]   0x8000001b

Virtual Address Trace
  VA 0x000010ab (decimal: 4267) --> 000048ab (decimal   18603) [VPN 4]
     VA = 0001 0000 1010 1011
     VPN = 4
     PFN = 0x12
     PA = 0100 1000 1010 1011 = 0x48ab

  VA 0x00003385 (decimal: 13189) --> 00000f85 (decimal   3973) [VPN 12]
     VA = 0011 0011 1000 0101
     VPN = 12
     PFN = 0x3
     PA = 1111 1000 0101 = 0xf85

  VA 0x0000231d (decimal: 8989) --> 00007f1d (decimal   32541) [VPN 8]
     VA = 0010 0011 0001 1101
     VPN = 8
     PFN = 0x1f
     PA = 0x7f1d

  VA 0x000000e6 (decimal: 230) --> 000060e6 (decimal   24806) [VPN 0]
     VA = 1110 0110
     VPN = 0
     PFN = 0x18
     PA = 0110 0000 1110 0110 = 0x60e6

  VA 0x00002e0f (decimal: 11791) --> 0000560f (decimal   22031) [VPN 11]
     VA = 0010 1110 0000 1111
     VPN = 11
     PFN = 0x15
     PA = 0001 0101 10 0000 1111 = 0x560f

Command: 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]   0x80000018
  [7]   0x80000008
  [8]   0x8000001f
  [9]   0x8000001c
  [10]   0x80000017
  [11]   0x80000015
  [12]   0x80000003
  [13]   0x80000013
  [14]   0x8000001e
  [15]   0x8000001b

Virtual Address Trace
  VA 0x000010ab (decimal: 4267) --> 000048ab (decimal   18603) [VPN 4]
     VA = 0001 0000 1010 1011
     VPN = 4
     PFN = 0x12
     PA = 0100 1000 1010 1011 = 0x48ab

  VA 0x00003385 (decimal: 13189) --> 00000f85 (decimal   3973) [VPN 12]
     VA = 0011 0011 1000 0101
     VPN = 12
     PFN = 0x3
     PA = 1111 1000 0101 = 0xf85

  VA 0x0000231d (decimal: 8989) --> 00007f1d (decimal   32541) [VPN 8]
     VA = 0010 0011 0001 1101
     VPN = 8
     PFN = 0x1f
     PA = 0x7f1d

  VA 0x000000e6 (decimal: 230) --> 000060e6 (decimal   24806) [VPN 0]
     VA = 1110 0110
     VPN = 0
     PFN = 0x18
     PA = 0110 0000 1110 0110 = 0x60e6

  VA 0x00002e0f (decimal: 11791) --> 0000560f (decimal   22031) [VPN 11]
     VA = 0010 1110 0000 1111
     VPN = 11
     PFN = 0x15
     PA = 0001 0101 10 0000 1111 = 0x560f

As the percentage of pages allocated to each space increases, more valid pages exist. In this situation, the entries that existed for 75% and 100% were the same.

3)
Command: paging-linear-translate.py -P 8 -a 32 -p 1024 -v -s 1
32B AS -> 5 bit VA, 8B Page Size -> 3 bit offset
Page Table (from entry 0 down to the max size)
[0] 0x00000000
[1] 0x80000061
[2] 0x00000000
[3] 0x00000000

Virtual Address Trace
VA 0x0000000e (decimal: 14) --> 0000030e (decimal 782) [VPN 1]
VA = 1110
VPN = 1
PFN = 0x61
PA = 0110 0001 1110 = 0x30e

VA 0x00000014 (decimal: 20) --> Invalid (VPN 2 not valid)
VA = 1 0100
VPN = 2
PFN = Invalid!

VA 0x00000019 (decimal: 25) --> Invalid (VPN 3 not valid)
VA = 1 1001
VPN = 3
PFN = Invalid!

VA 0x00000003 (decimal: 3) --> Invalid (VPN 0 not valid)
VA = 1 0100
VPN = 0
PFN = Invalid!

VA 0x00000000 (decimal: 0) --> Invalid (VPN 0 not valid)
VA = 1 0100
VPN = 0
PFN = Invalid!

---

paging-linear-translate.py -P 8k -a 32k -p 1m -v -s 2
VA 0x000055b9 (decimal: 21945) --> Invalid (VPN 2 not valid)
VA = 0101 0101 1011 1001
VPN = 2
PFN = Invalid!

VA 0x00002771 (decimal: 10097) --> Invalid (VPN 1 not valid)
VA = 0010 0111 0111 0001
VPN = 1
PFN = Invalid!

VA 0x00004d8f (decimal: 19855) --> Invalid (VPN 2 not valid)
  VA = 0100 1101 1000 1111
  VPN = 2
  PFN = Invalid!

VA 0x00004dab (decimal: 19883) --> Invalid (VPN 2 not valid)
  VA = 0100 1101 1010 1011
  VPN = 2
  PFN = Invalid!

VA 0x00004a64 (decimal: 19044) --> Invalid (VPN 2 not valid)
  VA = 0100 1010 0110 0100
  VPN = 2
  PFN = Invalid!

Command: paging-linear-translate.py -P 1m -a 256m -p 512m -v -s 3
VA 0x0c63244f (decimal: 207823951) --> 1773244f (decimal 393421903) [VPN 198]
  VA = 1100 0110 0011 0010 0100 0100 1111
  VPN = 198
  PFN = 0x177
  PA = 1 0111 0111 0011 0010 0100 0100 1111 = 0x1773244f

VA 0x024c3a22 (decimal: 38550050) --> 06cc3a22 (decimal 114047522) [VPN 36]
  VA = 0010 0100 1100 0011 1010 0010 0010
  VPN = 36
  PFN = 0x6c
  PA = 0110 1100 1100 0011 1010 0010 0010 = 0x6cc3a22

VA 0x0fdc0755 (decimal: 266078037) --> Invalid (VPN 253 not valid)
  VA = 1111 1101 1100 0000 0111 0101 0101
  VPN = 253
  PFN = Invalid!

VA 0x07ac666b (decimal: 128738923) --> 1dac666b (decimal 497837675) [VPN 122]
  VA = 0111 1010 1100 0110 0110 0110 1011
  VPN = 122
  PFN = 0x1da
  PA = 1 1101 1010 1100 0110 0110 0110 1011 = 0x1da666b

VA 0x099581b2 (decimal: 160793010) --> Invalid (VPN 153 not valid)
  VA = 1001 1001 0101 1000 0001 1011 0010
  VPN = 153
  PFN = Invalid!

For the first command: paging-linear-translate.py -P 8 -a 32 -p 1024 -v -s 1
8 byte pages, 32 byte address space, and 1024 byte memory are extremely small, and not practical on modern machines. The string “Hello World” would not even fit in a single page. I couldn’t even imagine an embedded system having such a small address space

4)

If the address space is larger than physical memory, then the program does not work and throws an error. This makes sense, because the address space maps to physical memory, and you cannot map virtual memory to memory that does not exist.

In a similar situation, if the page size is the same size as the address space, then the program still works, but we only have one page for the entire physical memory, which defeats the purpose of paging.

The program will not work if the page size is larger than the address space. There is no page table as output in this situation.