Courtesy of khaled D

The format of the page table is simple:

The high-order (left-most) bit is the VALID bit.

If the bit is 1, the rest of the entry is the PFN.

If the bit is 0, the page is not valid.

Page Table (from entry 0 down to the max size)

[0] 0x80000016

[1] 0x8000003c

[2] 0x00000000

[3] 0x80000015

[4] 0x00000000

[5] 0x00000000

[6] 0x80000009

[7] 0x80000016

Virtual Address Trace

VA 0x00000568 (decimal: 1384) --> RA or invalid address? _____

VA 0x00000dc3 (decimal: 3523) --> RA or invalid address? _____

VA 0x00000c5d (decimal: 3165) --> RA or invalid address? _____

VA 0x00000ebb (decimal: 3771) --> RA or invalid address? _____

VA 0x00001c32 (decimal: 7218) --> RA or invalid address? _____

pagetrans:

1) from address space get VA bits and from page size get displacement (d) bits.

Address space =8k -> VA bits = log2(8k) = 13

Page size = 1k -> D bits = log2(1k) = 10

VA bits is 13 and D bits is 10

2) VA bits - d bits = VPN bits

VPN= 13 – 10 = 3 bits.

Solving 0x00000568

3) VPN = page number, look at page table in the code --> see the number and take leftmost hexa --> make it binary and see leftmost bit. if 1= valid. if 0=invalid.

0x00000568 = 0000 0101 0110 1000 VA

VA = 0 0101 0110 1000 VPN Displacement

VPN: 001 = 1 = page 1 -> 0x8000003c

0x8 -> 1000 -> valid

4) if valid, replace VPN with PFN (the number in the table from step 3).

VA =0 0101 0110 1000 VPN Displacement

VA = 0x8000003c 01 0110 1000 PFN Displacement

5) from memory size get real address --> take these number of bits and make it hexa.

Memory size = 64k = log2(64k) = 16 bits.

VA = 0x8000003c 01 0110 1000 PFN Displacement (take up to 16 bits)

0011 1100 01 0110 1000 (take up to 16 bits)

1111 0001 0110 1000 (rearrange from the right)

1111 0001 0110 1000 -> real address -> 0xF168

Solving 0x00000dc3

3) VPN = page number, look at page table in the code --> see the number and take leftmost hexa --> make it binary and see leftmost bit. if 1= valid. if 0=invalid.

0x00000dc3 = 0000 1101 1100 0011 VA

VA = 0 1101 1100 0011 VPN Displacement

VPN: 011 = 3 = page 3 -> 0x80000015

0x8 -> 1000 -> valid

4) if valid, replace VPN with PFN (the number in the table from step 3).

VA =0 1101 1100 0011 VPN Displacement

VA = 0x80000015 01 1100 0011 PFN Displacement

5) from memory size get real address --> take these number of bits and make it hexa.

Memory size = 64k = log2(64k) = 16 bits.

VA = 0x8000001501 1100 0011 PFN Displacement (take up to 16 bits)

0001 0101 01 1100 0011 (take up to 16 bits)

0101 0101 1100 0011 -> real address -> 0x55C3

Solving 0x00000c5d

3) VPN = page number, look at page table in the code --> see the number and take leftmost hexa --> make it binary and see leftmost bit. if 1= valid. if 0=invalid.

0x00000c5d = 0000 1100 0101 1101 VA

VA = 0 1100 0101 1101 VPN Displacement

VPN: 011 = 3 = page 3 -> 0x80000015

0x8 -> 1000 -> valid

4) if valid, replace VPN with PFN (the number in the table from step 3).

VA =0 1100 0101 1101 VPN Displacement

VA = 0x80000015 00 0101 1101 PFN Displacement

5) from memory size get real address --> take these number of bits and make it hexa.

Memory size = 64k = log2(64k) = 16 bits.

VA = 0x8000001500 0101 1101 PFN Displacement (take up to 16 bits)

0001 0101 00 0101 1101 (take up to 16 bits)

0101 0100 0101 1101 -> real address -> 0x545D

Solving 0x00000ebb

3) VPN = page number, look at page table in the code --> see the number and take leftmost hexa --> make it binary and see leftmost bit. if 1= valid. if 0=invalid.

0x00000ebb = 0000 1110 1011 1011 VA

VA = 0 1110 1011 1011 VPN Displacement

VPN: 011 = 3 = page 3 -> 0x80000015

0x8 -> 1000 -> valid

4) if valid, replace VPN with PFN (the number in the table from step 3).

VA =0 1110 1011 1011 VPN Displacement

VA = 0x80000015 10 1011 1011 PFN Displacement

5) from memory size get real address --> take these number of bits and make it hexa.

Memory size = 64k = log2(64k) = 16 bits.

VA = 0x8000001510 1011 1011 PFN Displacement (take up to 16 bits)

0001 0101 10 1011 1011 (take up to 16 bits)

0101 0110 1011 1011 -> real address -> 0x56BB

Solving 0x00001c32

3) VPN = page number, look at page table in the code --> see the number and take leftmost hexa --> make it binary and see leftmost bit. if 1= valid. if 0=invalid.

0x00001c32 = 0001 1100 0011 0010 VA

VA = 1 1100 0011 0010 VPN Displacement

VPN: 111 = 7 = page 7 -> 0x80000016

0x8 -> 1000 -> valid

4) if valid, replace VPN with PFN (the number in the table from step 3).

VA =1 1100 0011 0010 VPN Displacement

VA = 0x80000016 00 0011 0010 PFN Displacement

5) from memory size get real address --> take these number of bits and make it hexa.

Memory size = 64k = log2(64k) = 16 bits.

VA = 0x8000016 00 0011 0010 PFN Displacement (take up to 16 bits)

0001 0110 00 0011 0010 (take up to 16 bits)

0101 1000 0011 0010 -> real address -> 0x5832

Courtesy of khaled D

pagetablesize:

ARG bits in virtual address 38

ARG page size 1m = 1 mega

ARG pte size 16

1) from page number get d bits

Page number= 1M = Log2(1M) = 20

2) VA bits - d bits = VPN bits

38 – 20= 18

3) from VPN get number of bits --> 2^VPN and multiply it with pte size

$2^{18} * 16 = 4194$ k-byte = 4 MiB

Use google if you want to write result in mebibyte