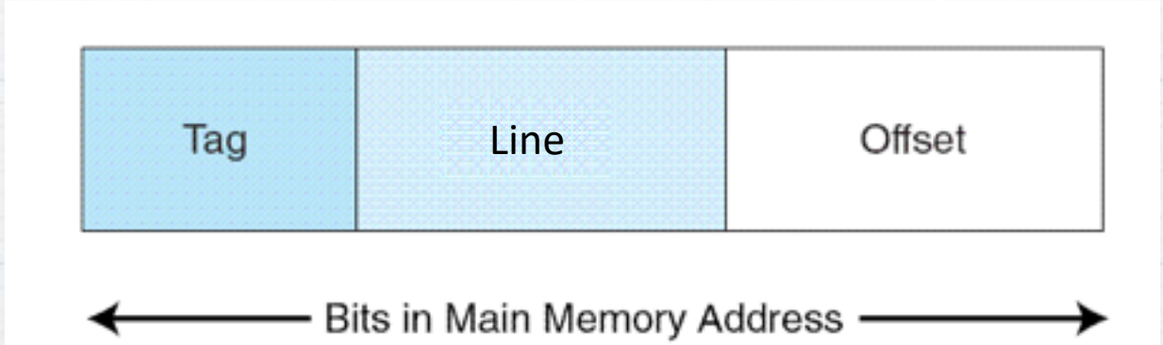
Direct Mapped

- Fully Associative is one extreme
  - A separate comparator is required for every cache line
- Direct mapped is the other extreme
  - Only one comparator is needed for the entire cache
- With a direct mapped cache with N lines:
  - Block B of memory maps to line L = B mod N
    - B is the block number, not the address
- Example: direct mapped cache with 10 lines, Line 6 may hold block 6 or 16 or 26 or 36 ....

Direct Mapped

- Address is split into 3 fields for direct mapping
  - The *offset* field identifies location within line
  - The *line* field selects a unique line of cache
  - The *tag* field is whatever is left over.



- Offset width OW = $\log_2$ line size
- Line# width LW = $\log_2$ number of lines
- Tag width = (Bits in address) - LW - OW

cache

line	tag	data
00		
01		
10		
11		

main memory

address
0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111

A system has 2^4 bytes of main memory and a direct mapped cache with 8 bytes.

A line is 2 bytes.

Address = 4 bits

Offset field = log(2) = 1 bit

Number of lines in cache =
(8 bytes/cache) / (2 bytes/line) = 4 lines/cache

Line field = log(4) = 2 bits

Tag field = 4 - 1 - 2 = 1 bit

For address 1010 what are the offset, line, and tag fields?

1010 = 1 01 0
tag = 1, line = 01, offset = 0

cache

line	tag	data
00		
01		
10		
11		

address
0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111

A system has 2^4 bytes of main memory and a direct mapped cache with 8 bytes.

A line is 2 bytes.

Address = 4 bits

Offset field = log(2) = 1 bit

Number of lines in cache =
(8 bytes/cache) / (2 bytes/line) = 4 lines/cache

Line field = log(4) = 2 bits

Tag field = 4 - 1 - 2 = 1 bit

For address 1010 what are the offset, line, and tag fields?

1010 = 1 01 0
tag = 1, line = 01, offset = 0

cache

line	tag	data
00	1	0xACD7
01	0	0x1243
10	0	0xA398
11	1	0x72AC

Say the state of the cache described in the last slide is as shown.

The following memory addresses are requested. For each, is it a hit? If it is a hit, what is the data requested?

0xA
0xF
0x3

A system has a main memory of 2^{32} bytes with a block size of 16 bytes. It has a direct mapped cache of 2^{10} bytes.

How many bits are in an address?

What is the size of the offset field?

How many lines are in the cache?

What is the size of the line field?

What is the size of the tag field?

A system has a main memory of 2^{32} bytes with a block size of 16 bytes. It has a direct mapped cache of 2^{10} bytes.

How many bits are in an address? 32 bits

What is the size of the offset field? log(16) = 4 bits

How many lines are in the cache? 2^{10} bytes/cache / 2^4 bytes/line = 2^6 lines/cache

What is the size of the line field? log(2^6) = 6 bits

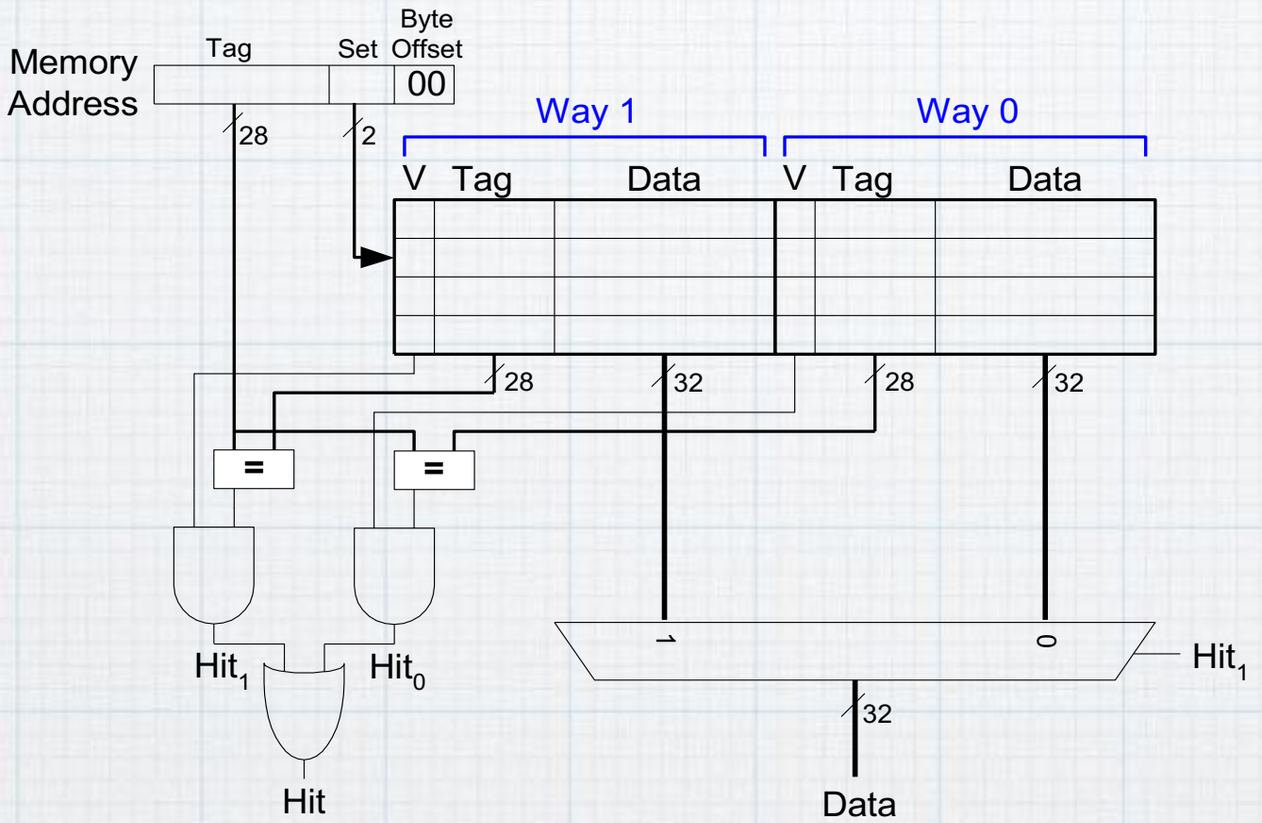
What is the size of the tag field? 32-6-4 = 22 bits

Set Associative

- Lines are put into sets
- A number of blocks can be mapped to and put in the same set
- Reduces conflicts
- Addresses are divided into three fields
  - tag, set, offset
- Set field determines set (group of lines) to which the memory block maps
- Tag field identifies which line within the set contains the block (if it is there)
- Offset field chooses the location within the line

Set Associative

Example: 32-bit address, 4 sets, 2 lines per set, 4 bytes per line



Address	Data
00000	
00010	
00011	
00101	
00110	
01000	
01001	
01011	
01100	
01110	
01111	
10001	
10010	
10100	
10101	
10111	
11000	
	0xB2
11100	0xB9
11101	0x9A
11110	0x77

A system has 32 bytes of main memory. A block is 4 bytes. It has a 2-way set associative cache with 16 bytes.

Number of sets = (16 bytes/cache) / ((4 bytes/line) * (2 lines/set))

= 2 sets

Offset field = log(4) = 2 bits
Line field = log(2) = 1 bit
Tag = 5 - 1 - 2 = 2 bits

Set	Way0 Tag	Way0 Data	Way1 Tag	Way1 Data
0				
1				

Address	
00001	
00010	
00100	
00101	
00111	
01000	
01010	
01011	
01101	
01110	
10000	
10001	
10011	0x1D
10100	0x55
10101	0x67
10110	0x7A
10111	0xD4
11000	0x4B
11001	0xBB
11010	0xCB
11011	0xB2
11100	0xB9
11101	0x9A
11110	0x77

A system has 32 bytes of main memory. A block is 4 bytes. It has a 2-way set associative cache with 16 bytes.

Number of sets = (16 bytes/cache) / ((4 bytes/line) * (2 lines/set))

= 2 sets

Offset field = log(4) = 2 bits
Set field = log(2) = 1 bit
Tag = 5 - 1 - 2 = 2 bits

Where does address 10101 map to?

10 1 01

Tag = 10, set = 1

Set	Way0 Tag	Way0 Data	Way1 Tag	Way1 Data
0				
1				

Address	
00000	
00001	
00011	
00100	
00110	
00111	
01001	
01010	
01100	
01101	
01111	
10000	
10011	0x1D
10100	0x55
10101	0x67
10110	0x7A
10111	0xD4
11000	0x4B
11001	0xBB
11010	0xCB
11011	0xB2
11100	0xB9
11101	0x9A
11110	

A system has 32 bytes of main memory. A block is 4 bytes. It has a 2-way set associative cache with 16 bytes.

Number of sets = (16 bytes/cache) / ((4 bytes/line) * (2 lines/set))

= 2 sets

Offset field = log(4) = 2 bits
Set field = log(2) = 1 bit
Tag = 5 - 1 - 2 = 2 bits

Where does address 10101 map to?

10 1 01

Tag = 10, set = 1

Set	Way0 Tag	Way0 Data	Way1 Tag	Way1 Data
0				
1	10	0xD47A6755		

Address	Data
00000	0x43
00001	0xB1
00010	0x3D
00011	0x13
00100	0x95
00101	0x94
00110	
01000	
01001	
01011	
01100	
01110	
01111	
10001	
10010	
10100	
10101	
10111	
11000	
	0xBB
11010	
11011	0xB2
11100	0xB9
11101	0x9A
11110	

A system has 32 bytes of main memory. A block is 4 bytes. It has a 2-way set associative cache with 16 bytes.

Number of sets = (16 bytes/cache) / ((4 bytes/line) * (2 lines/set))

= 2 sets

Offset field = log(4) = 2 bits
Set field = log(2) = 1 bit
Tag = 5 - 1 - 2 = 2 bits

Where does address 00000 map to?

00 0 00

Tag = 00, set = 0

Set	Way0 Tag	Way0 Data	Way1 Tag	Way1 Data
0	00	0x133DB143		

0xD47A6755

Address	Data
00000	0x43
00001	0xB1
00010	0x3D
00011	0x13
00100	0x95
00101	0x94
00110	0x65
00111	0xC4
01000	0xDB
01001	0x1A
01010	0x1C
01011	0x5D
01100	0x76
01101	0x69
01110	0x7A
01111	0x8D
10000	0xDE
10001	0xE4
10010	0xDF
10011	0x1D
10100	0x55
10101	0x67
10110	0x7A
10111	0xD4
11000	0x4B
11001	0xBB
11010	0xCB
11011	0xB2
11100	0xB9
11101	0x9A
11110	0x77
11111	0x89

A system has 32 bytes of main memory. A block is 4 bytes. It has a 2-way set associative cache with 16 bytes.

Number of sets = (16 bytes/cache) / ((4 bytes/line) * (2 lines/set))

= 2 sets

Offset field = log(4) = 2 bits
Set field = log(2) = 1 bit
Tag = 5 - 1 - 2 = 2 bits

Where does address 11010 map to?

11 0 10

Tag = 11, set = 0

Set	Way0 Tag	Way0 Data	Way1 Tag	Way1 Data
0	00	0x133DB143	11	0xB2CBBB4B
1	10	0xD47A6755		

A system has a main memory of 2^{32} bytes with a 2^5 byte block size. It has a 4-way set associative cache with 2^{10} bytes.

How many bits are in the address?

What is the size of the offset field?

How many sets are there?

How many bits in the set field?

How many bits in the tag field?

A system has a main memory of 2^{32} bytes with a 2^5 byte block size. It has a 4-way set associative cache with 2^{10} bytes.

How many bits are in the address? 32 bits

What is the size of the offset field? $\log(2^5) = 5$ bits

How many sets are there?
2^{10} bytes/cache / (2^5 bytes/line * 4 lines/set) = 2^3 sets/cache

How many bits in the set field? $\log(2^3) = 3$ bits

How many bits in the tag field? 32-3-5 = 24 bits

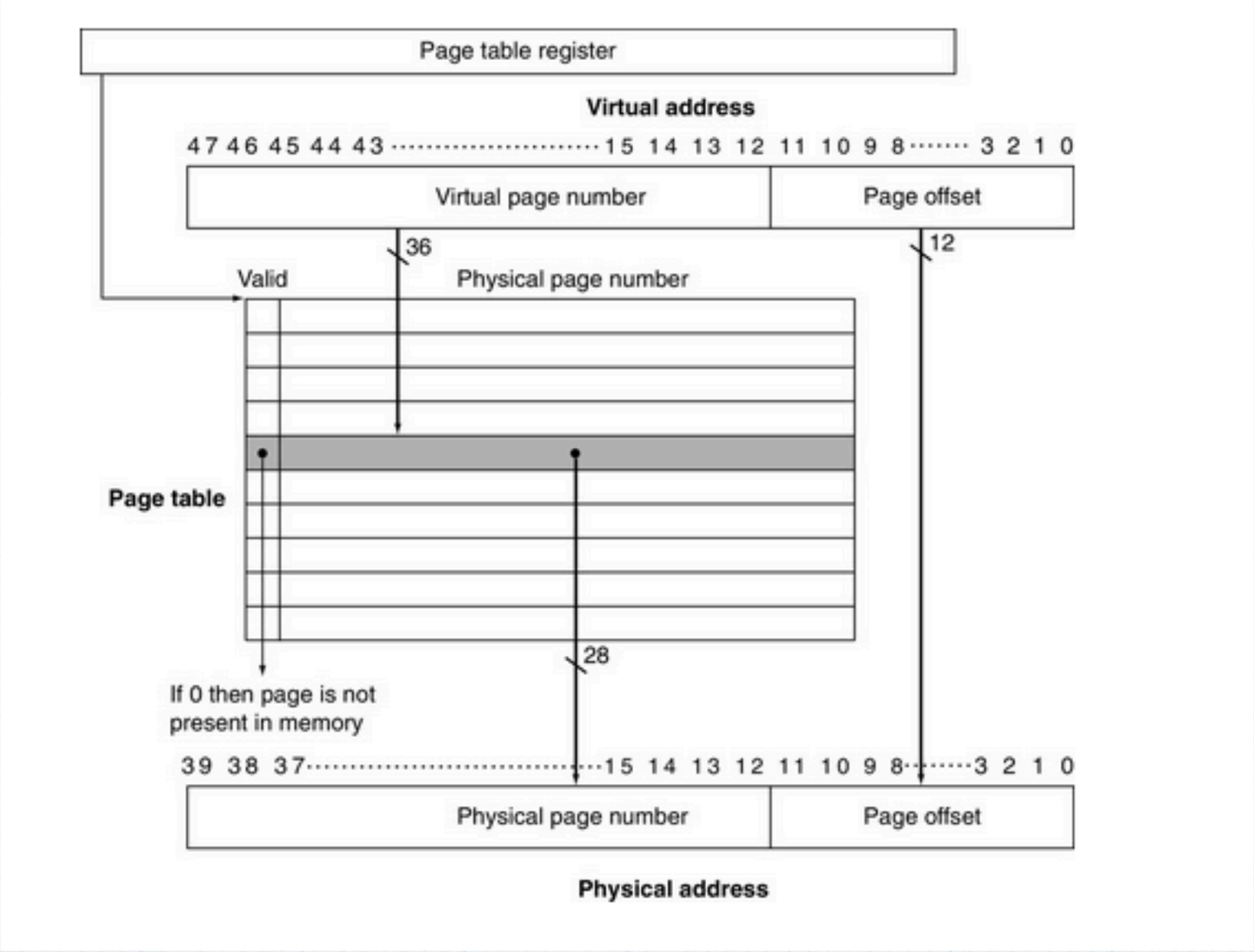
Virtual Memory

- Operating systems translate virtual addresses
- A virtual address is divided into two field
  - Page number
  - Offset
- Page number indexes into the Page Map Table to select a Page Table Entry
  - If valid bit = 0 there is page fault
  - If valid bit = 1, the frame number is read
    - Frame number concatenated with offset = physical address

Page Page Table

Page	Frame #	Valid Bit
0	2	1
1	-	0
2	-	0
3	0	1
4	1	1
5	-	0
6	-	0
7	3	1

Virtual Address Translation



TLB Lookup Process

1. Extract page# and offset from the virtual address.

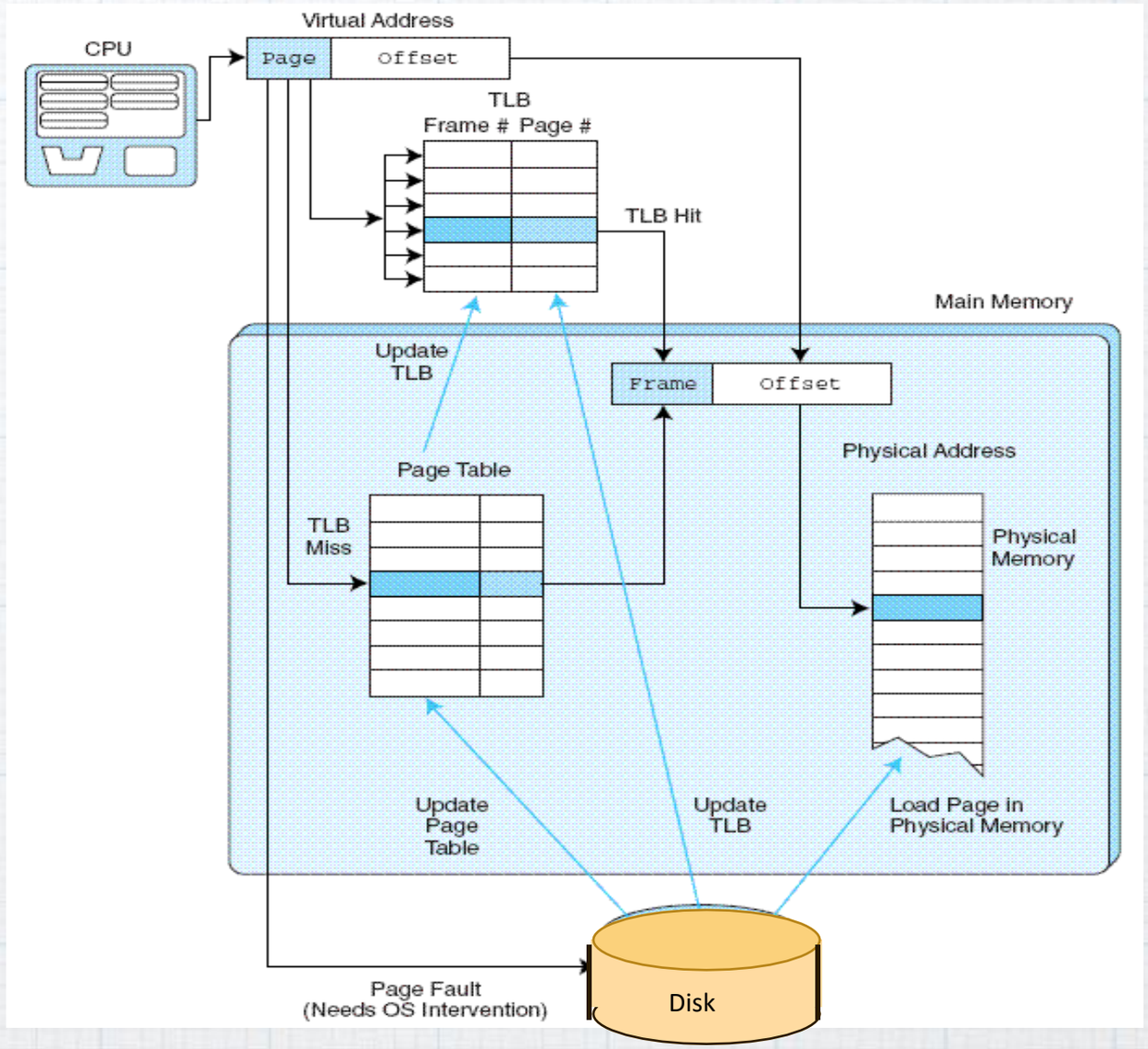
2. If TLB hit, combine frame# from TLB with offset to yield the physical address

3. Else use page# to check the page table entry (PTE) in the page map table (PMT).

If valid bit is set, combine frame# with offset to yield the physical address.

4. Else generate a page fault and load page from disk.

5. Restart the access once page is in memory.



Page	Frame	Valid
00000		0
00001		0
00010	010	1
00011		0
00100		0
00101	000	1
00110		0
00111		0
01000	011	1
01001		0
01010		0
01011	111	1
01100		0
01101		0
01110	001	1
01111		0
10000		0
10001		0
10010		0
10011		0
10100	101	1
10101		0
10110		0
10111	100	1
11000		0
11001		0
11010		0
11011	110	1
11100		0
11101		0
11110		0
11111		0

A system has 2^{15} bytes of virtual address space. A frame is 2^{10} bytes. Physical address space is 2^{13} bytes.

How many bits are in the virtual address?

What is the size of the page field?

What is the size of the offset field?

How many bits are in the physical address?

What is the size of the frame field?

What is the size of the offset field?

Page	Frame	Valid
00000		0
00001		0
00010	010	1
00011		0
00100		0
00101	000	1
00110		0
00111		0
01000	011	1
01001		0
01010		0
01011	111	1
01100		0
01101		0
01110	001	1
01111		0
10000		0
10001		0
10010		0
10011		0
10100	101	1
10101		0
10110		0
10111	100	1
11000		0
11001		0
11010		0
11011	110	1
11100		0
11101		0
11110		0
11111		0

A system has 2^{15} bytes of virtual address space. A frame is 2^{10} bytes. Physical address space is 2^{13} bytes.

How many bits are in the virtual address? 15 bits
What is the size of the page field?
15-10 = 5 bits
What is the size of the offset field? 10 bits

How many bits are in the physical address? 13 bits

What is the size of the frame field? 13-10 = 3 bits

What is the size of the offset field? 10 bits

Page	Frame	Valid
00000		0
00001		0
00010	010	1
00011		0
00100		0
00101	000	1
00110		0
00111		0
01000	011	1
01001		0
01010		0
01011	111	1
01100		0
01101		0
01110	001	1
01111		0
10000		0
10001		0
10010		0
10011		0
10100	101	1
10101		0
10110		0
10111	100	1
11000		0
11001		0
11010		0
11011	110	1
11100		0
11101		0
11110		0
11111		0

A system has 2^{15} bytes of virtual address space. A frame is 2^{10} bytes. Physical address space is 2^{13} bytes.

A system wants to access the following virtual address:
0x5EDA

Is it in main memory?

If so translate the virtual address to a physical address.

Page	Frame	Valid
00000		0
00001		0
00010	010	1
00011		0
00100		0
00101	000	1
00110		0
00111		0
01000	011	1
01001		0
01010		0
01011	111	1
01100		0
01101		0
01110	001	1
01111		0
10000		0
10001		0
10010		0
10011		0
10100	101	1
10101		0
10110		0
10111	100	1
11000		0
11001		0
11010		0
11011	110	1
11100		0
11101		0
11110		0
11111		0

A system has 2^{15} bytes of virtual address space. A frame is 2^{10} bytes. Physical address space is 2^{13} bytes.

A system wants to access the following virtual address:
0x5EDA
0101 1110 1101 1010
10111 1011011010

Is it in main memory? Yes

If so translate the virtual address to a physical address.

100 1011011010
1 0010 1101 1010
0x12DA

Page	Frame	Valid
00000		0
00001		0
00010	010	1
00011		0
00100		0
00101	000	1
00110		0
00111		0
01000	011	1
01001		0
01010		0
01011	111	1
01100		0
01101		0
01110	001	1
01111		0
10000		0
10001		0
10010		0
10011		0
10100	101	1
10101		0
10110		0
10111	100	1
11000		0
11001		0
11010		0
11011	110	1
11100		0
11101		0
11110		0
11111		0

A system has 2^{15} bytes of virtual address space. A frame is 2^{10} bytes. Physical address space is 2^{13} bytes.

A system wants to access the following virtual address:
0x25D8

Is it in main memory?

If so translate the virtual address to a physical address.

Page	Frame	Valid
00000		0
00001		0
00010	010	1
00011		0
00100		0
00101	000	1
00110		0
00111		0
01000	011	1
01001		0
01010		0
01011	111	1
01100		0
01101		0
01110	001	1
01111		0
10000		0
10001		0
10010		0
10011		0
10100	101	1
10101		0
10110		0
10111	100	1
11000		0
11001		0
11010		0
11011	110	1
11100		0
11101		0
11110		0
11111		0

A system has 2^{15} bytes of virtual address space. A frame is 2^{10} bytes. Physical address space is 2^{13} bytes.

A system wants to access the following virtual address:
0x25D8
0010 0101 1101 1000
01001 0111011000

Is it in main memory? No

If so translate the virtual address to a physical address.