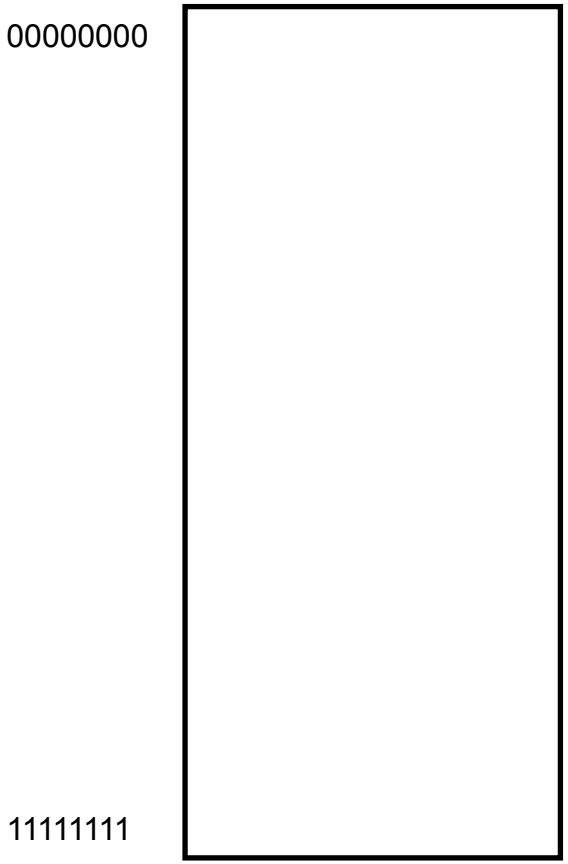
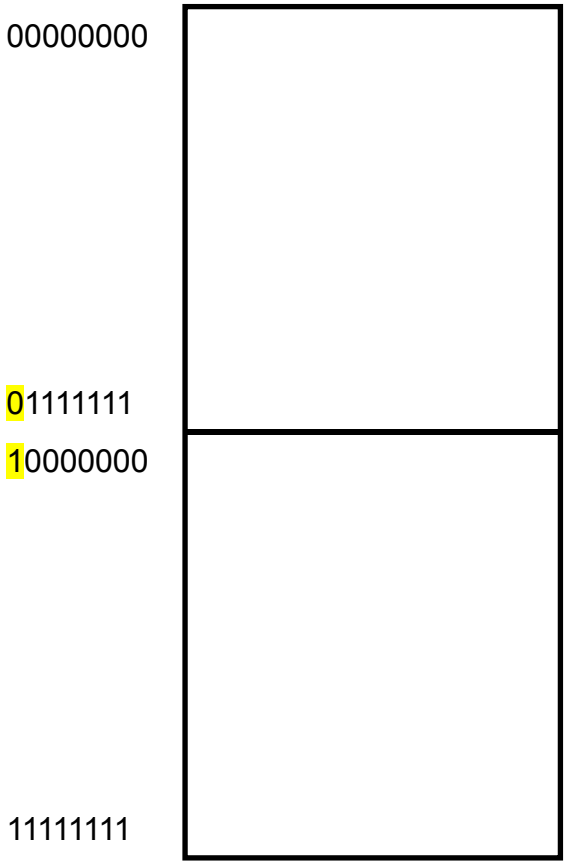
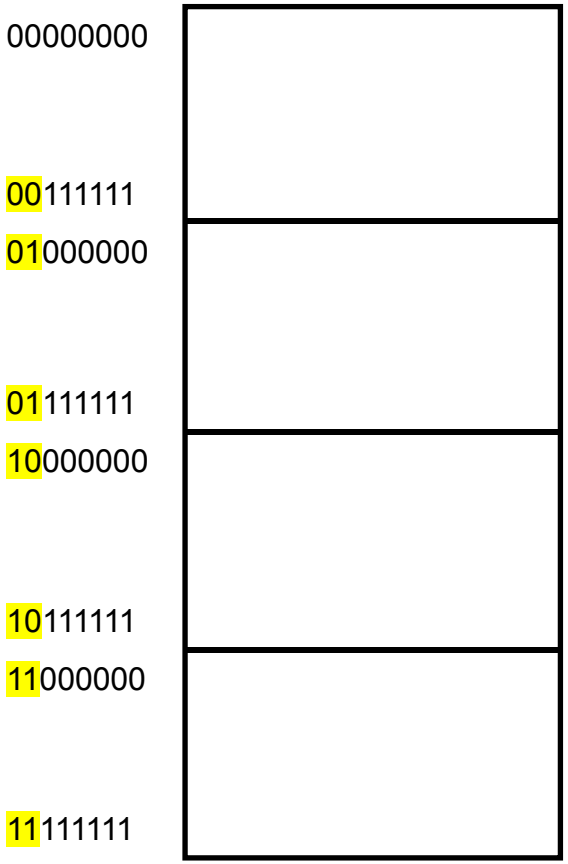
Physical Memory
256 bytes – requires 8 bits to address



Physical Memory
256 bytes – requires 8 bits to address
Break into 2 parts- note 1st block 0 in 1st bit, 2nd block has a 1

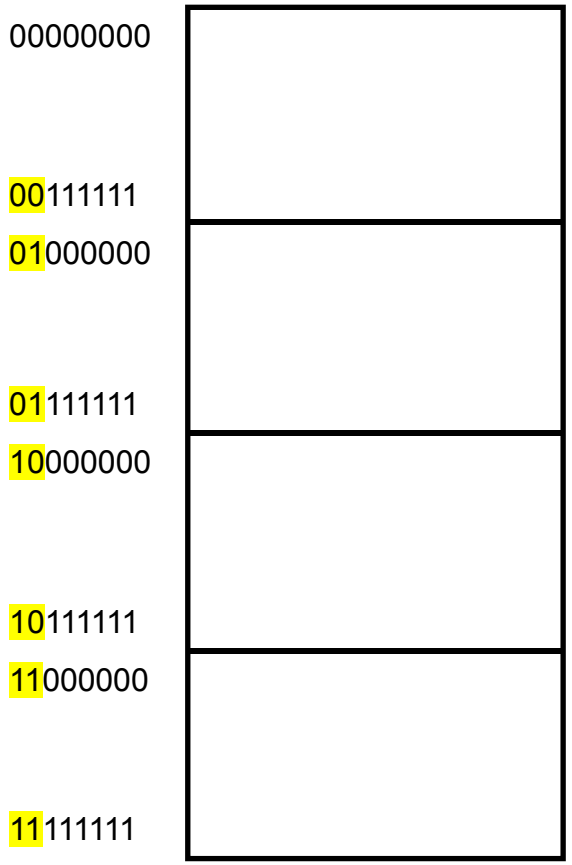


Physical Memory
256 bytes – requires 8 bits to address
Break into 4 parts- note first 2 bits in each block



What if we have 2 processes
A and B that need 2 blocks and
1 block of memory to run?

Physical Memory
256 bytes – requires 8 bits to address
Break into 4 parts- note first 2 bits in each block



What if we have 2 processes A and B that need 2 blocks and 1 block of memory to run?

The OS does this nifty lookup trick (virtual to physical page)

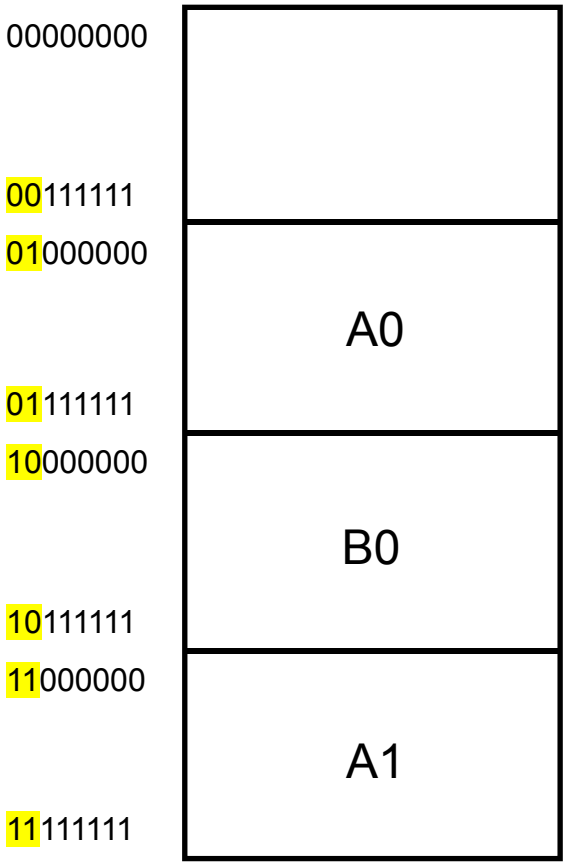
Process A

00	01
01	11
10	-
11	-

Process B

00	10
01	-
10	-
11	-

Physical Memory
256 bytes – requires 8 bits to address
Break into 4 parts- note first 2 bits in each block



What if we have 2 processes
A and B that need 2 blocks and
1 block of memory to run?

The OS does this nifty lookup
Trick (virtual to physical page)

Process A

00	01
01	11
10	-
11	-

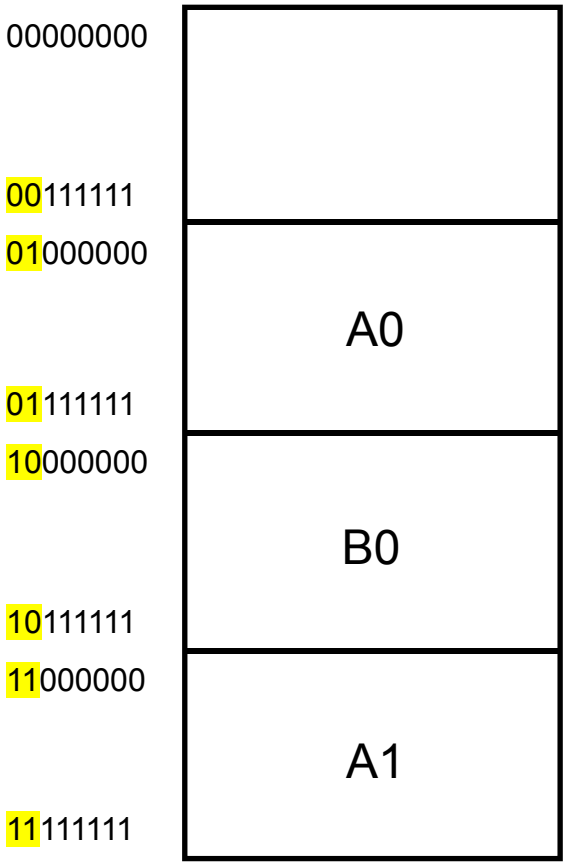
Process B

00	10
01	-
10	-
11	-

So if want to get to 00101010 in B
Sub 10 for 00 to get 10101010

Neither process can access others memory

Physical Memory
256 bytes – requires 8 bits to address
Break into 4 parts- note first 2 bits in each block



What if we have 2 processes
A and B that need 2 blocks and
1 block of memory to run?

The OS does this nifty lookup
Trick (virtual to physical page)

Process A

00	01
01	11
10	-
11	-

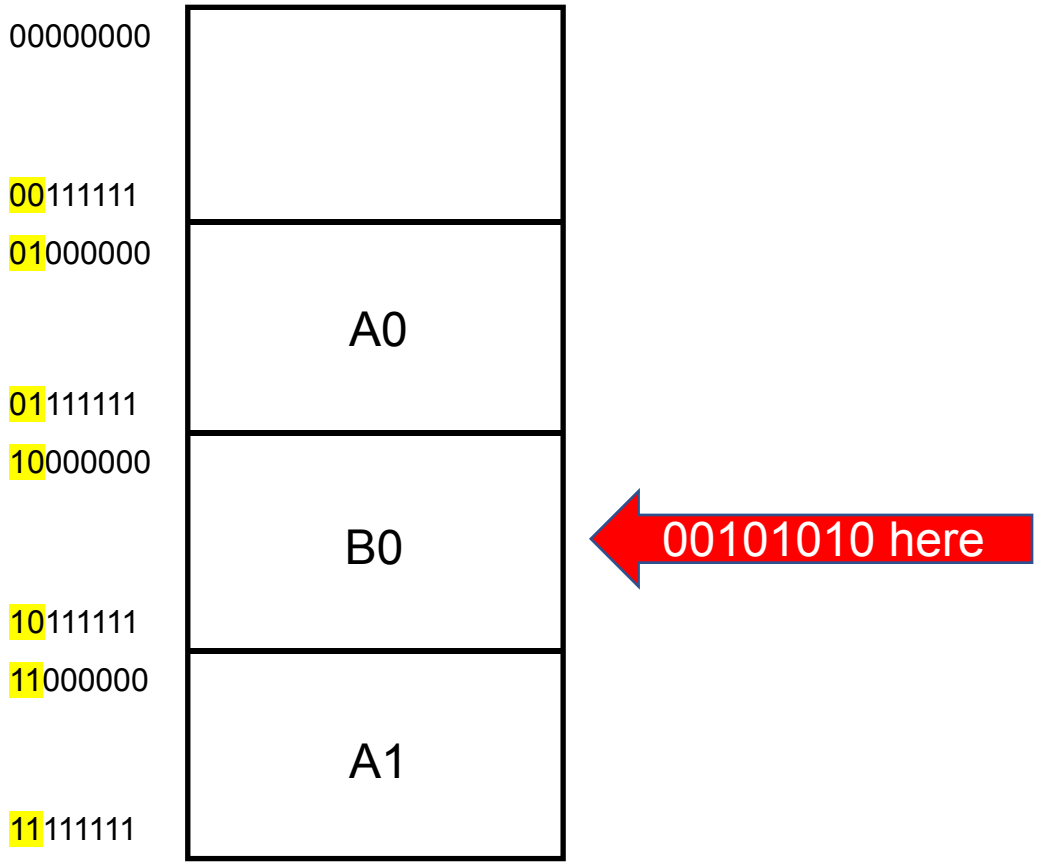
Process B

00	10
01	-
10	-
11	-

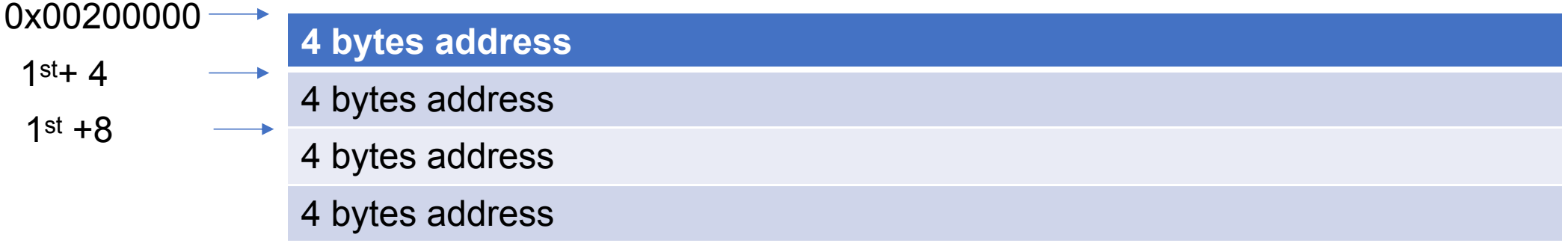
So if want to get to 00101010 in B
Sub 10 for 00 to get 10101010

Neither process can access others memory

Physical Memory
256 bytes – requires 8 bits to address
Break into 4 parts- note first 2 bits in each block



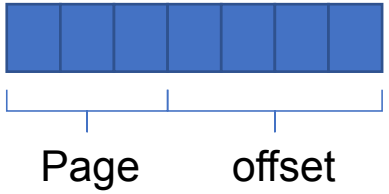
In Class Example
Page table- 32 bit system=> 32bits* 1 byte/8bits=4 bytes
Each row holds the physical address of the virtual page table, just add the offset to get the page of interest

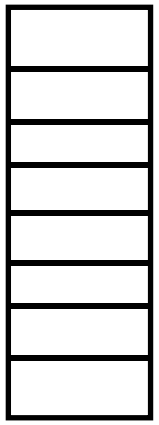


In Class Example
1st how big is your memory in bits?
2nd how large is your page(or frame) in bits?

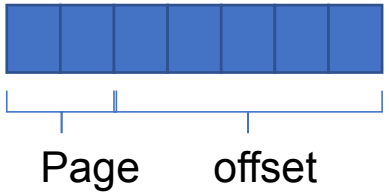
Ex.
Have 128 bytes of memory, #bits needed? 7

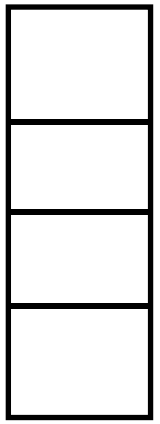
Page size?
If choose 16, then need 4 bits offset, 3 bits page
Will have 2**3 = 8 pages



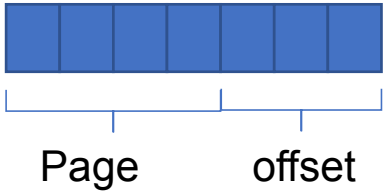


If choose 32, then need 5 bits offset, 2 bits page
Will have 2**2 = 4 pages





In Class Example
Page Table
If we have 128 bytes of memory and 3 bits offset
Then have 2**4 = 16 pages
Each page is 2**3 or 8 bytes long



Say we have 2 processes with the following page tables

P1

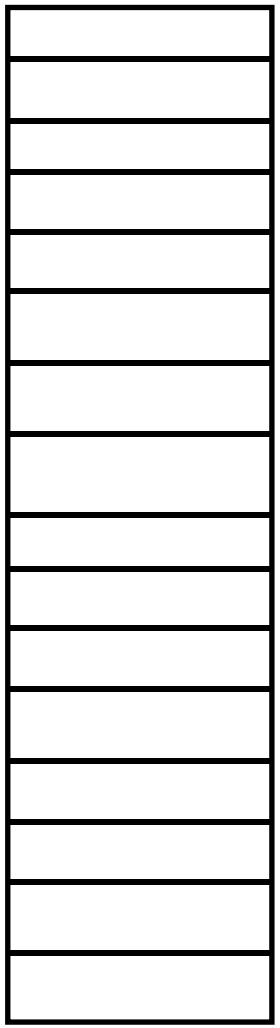
valid	
1	1011
1	1111
1	0111
0	1010
0	
0	
0	
0	
0	
0	
0	
0	
0	
0	
0	
0	

P2

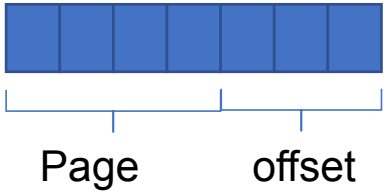
valid	
1	0011
1	0010
0	
0	
0	
0	
0	
0	
0	
0	
0	
0	
0	
0	
0	
0	

Where are the processes in Physical memory?

Physical memory



In Class Example
Page Table
If we have 128 bytes of memory and 3 bits offset
Then have 2**4 = 16 pages
Each page is 2**3 or 8 bytes long



Say we have 2 processes with the following page tables

P1

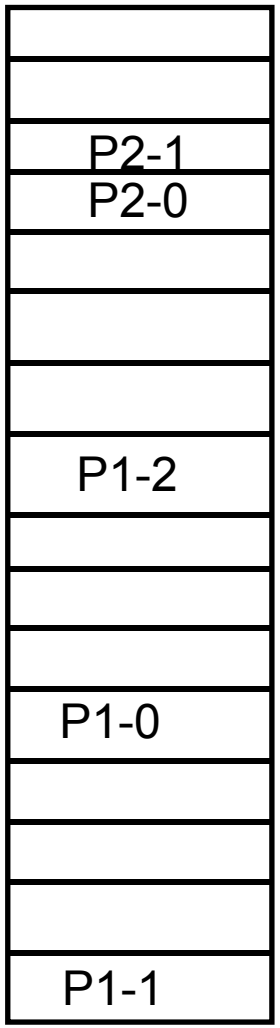
valid	
1	1011
1	1111
1	0111
0	1010
0	
0	
0	
0	
0	
0	
0	
0	
0	
0	
0	
0	

P2

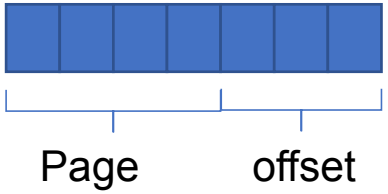
valid	
1	0011
1	0010
0	
0	
0	
0	
0	
0	
0	
0	
0	
0	
0	
0	
0	
0	

Where are the processes in Physical memory?

Physical memory



In Class Example
Page Table
If we have 128 bytes of memory and 3 bits offset
Then have 2**4 = 16 pages
Each page is 2**3 or 8 bytes long



Say we have 2 processes with the following page tables

P1

valid	
1	1011
1	1111
1	0111
0	1010
0	
0	
0	
0	
0	
0	
0	
0	
0	
0	
0	
0	

P2

valid	
1	0011
1	0010
0	
0	
0	
0	
0	
0	
0	
0	
0	
0	
0	
0	
0	
0	

Where are the processes in Physical memory?

Translate the following address
P1 0000001=>1011001

P1 0011000=>Fault, row 3 in P1's Page table does not contain a valid entry

Physical memory