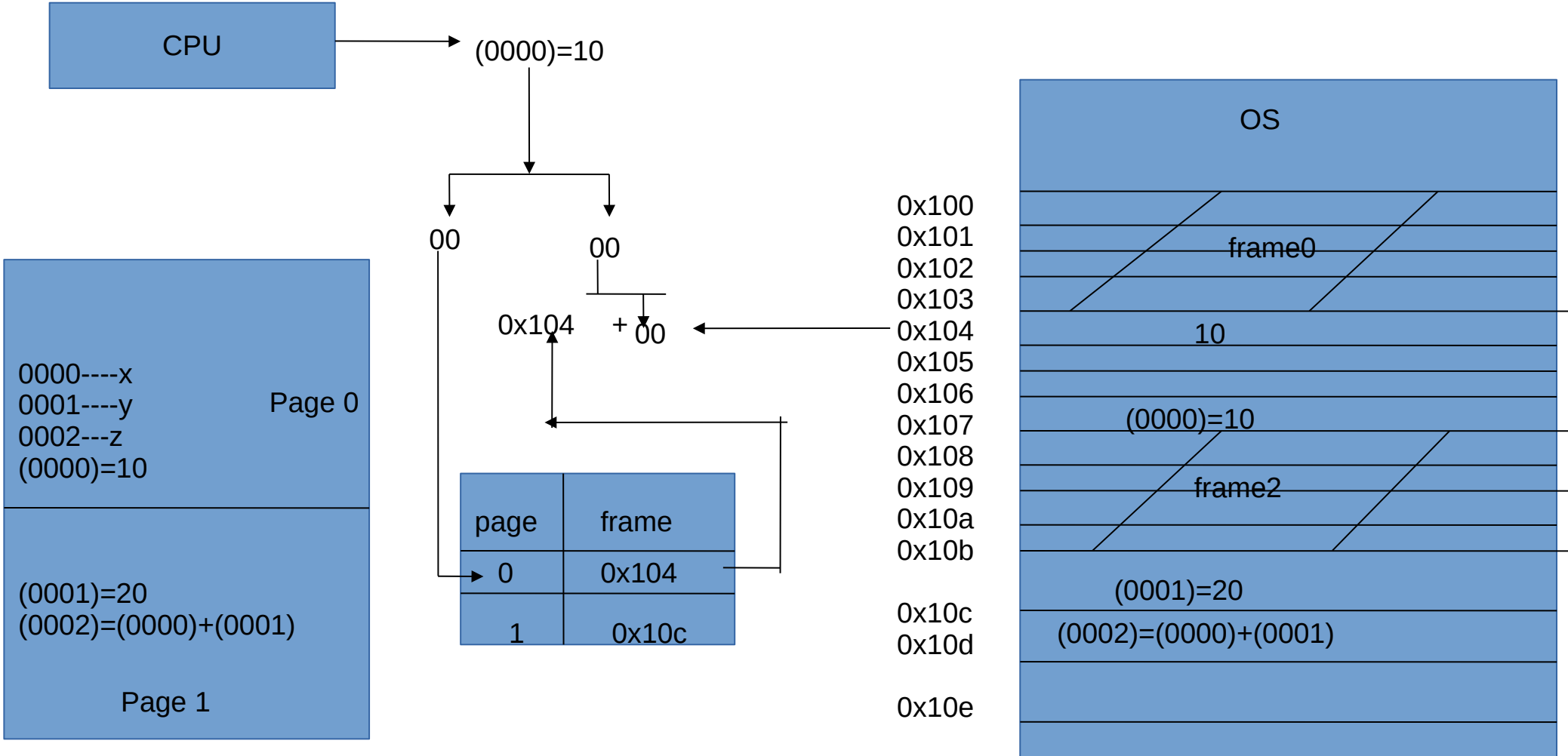

CPU
(0000)=10
00
00
0x104
+ 00
0000----x
0001----y
0002---z
(0000)=10
Page 0
(0001)=20
(0002)=(0000)+(0001)
Page 1
page
frame
0
0x104
1
0x10c
0x100
0x101
0x102
0x103
0x104
0x105
0x106
0x107
0x108
0x109
0x10a
0x10b
0x10c
0x10d
0x10e
OS
frame0
10
(0000)=10
frame2
(0001)=20
(0002)=(0000)+(0001)

Consider page size = frame size = 4 bytes

Suppose the logical address space for a process is 20 bytes.

How many pages are there in this process?

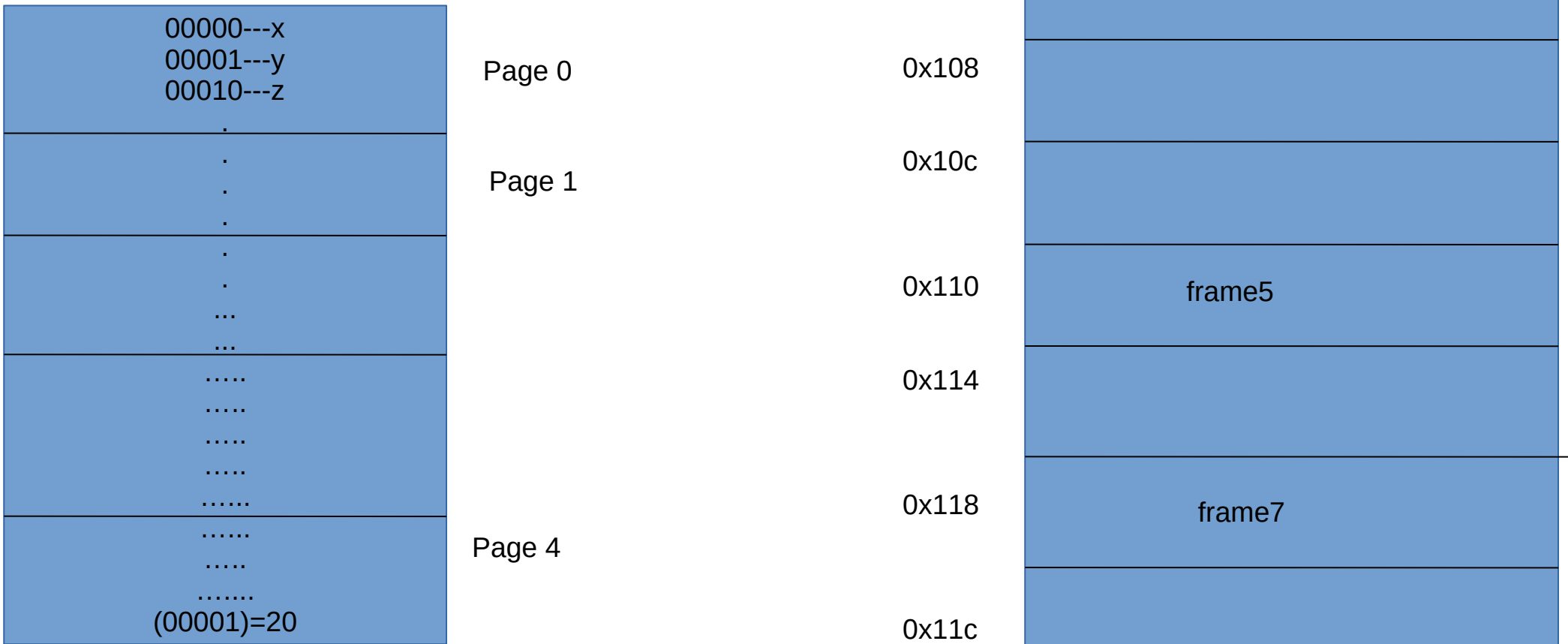
Draw the page table for this process.....

Solution ?
Multi level paging

One way of getting around this problem is to use a two level paging scheme in which the page table itself is also paged.

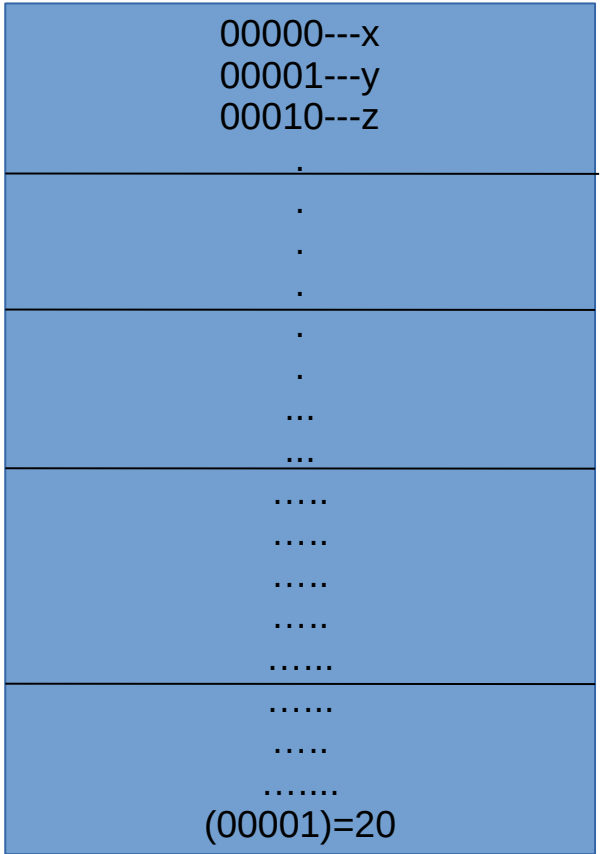
Page size= frame size=4

Total logical space 20 bytes



Page size= frame size=4

Total logical space 20 bytes

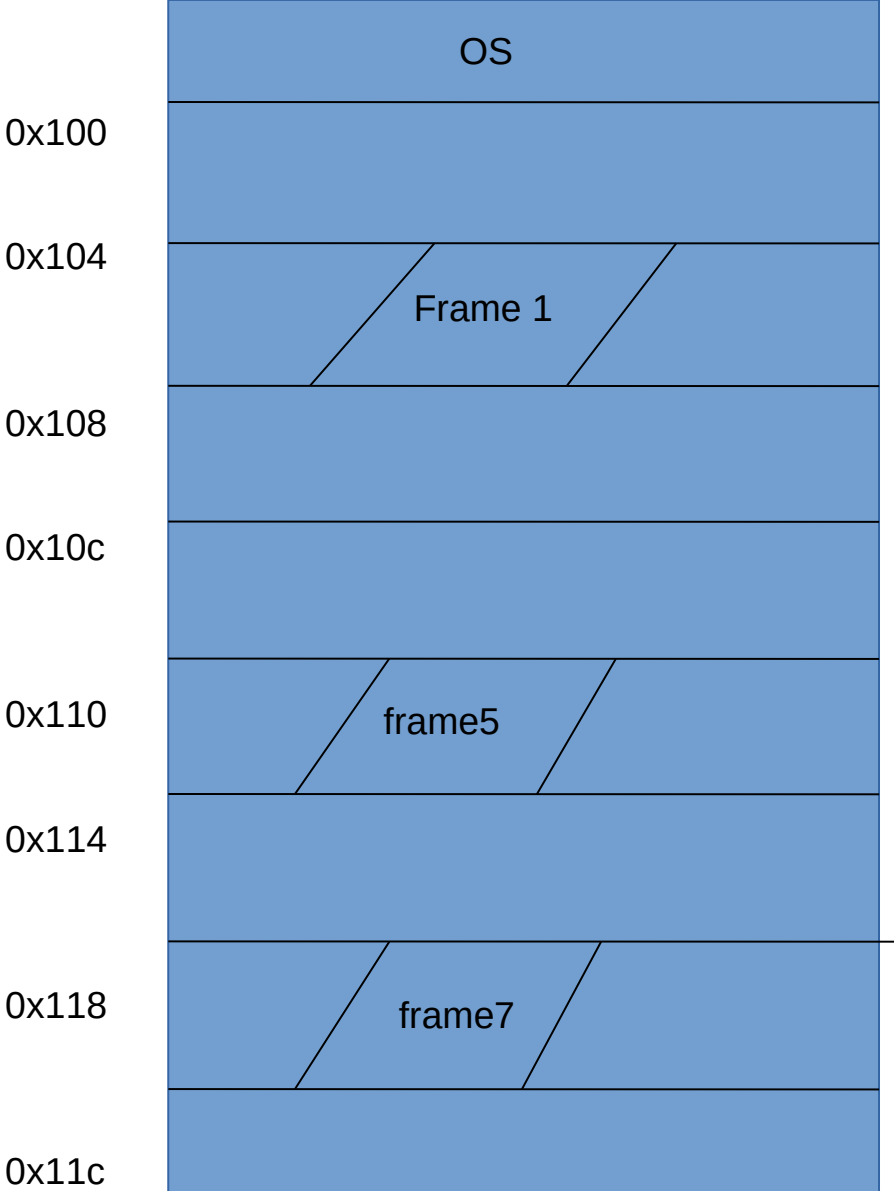


Page 0

Page 1

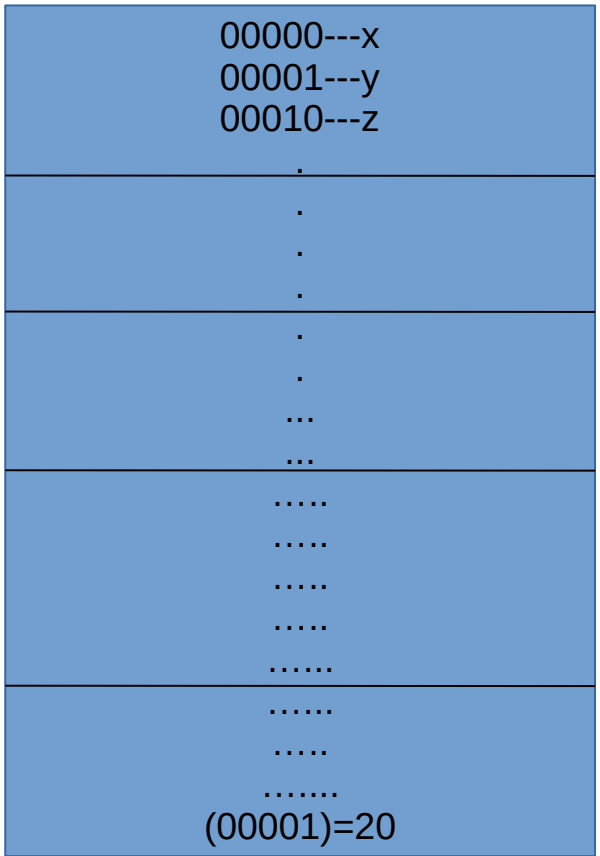
page	frame
0	0x100
1	0x108
2	0x10c
3	0x114
4	0x11c

Page 4



Total logical space 20 bytes m=5

Page size= frame size=4 n=2

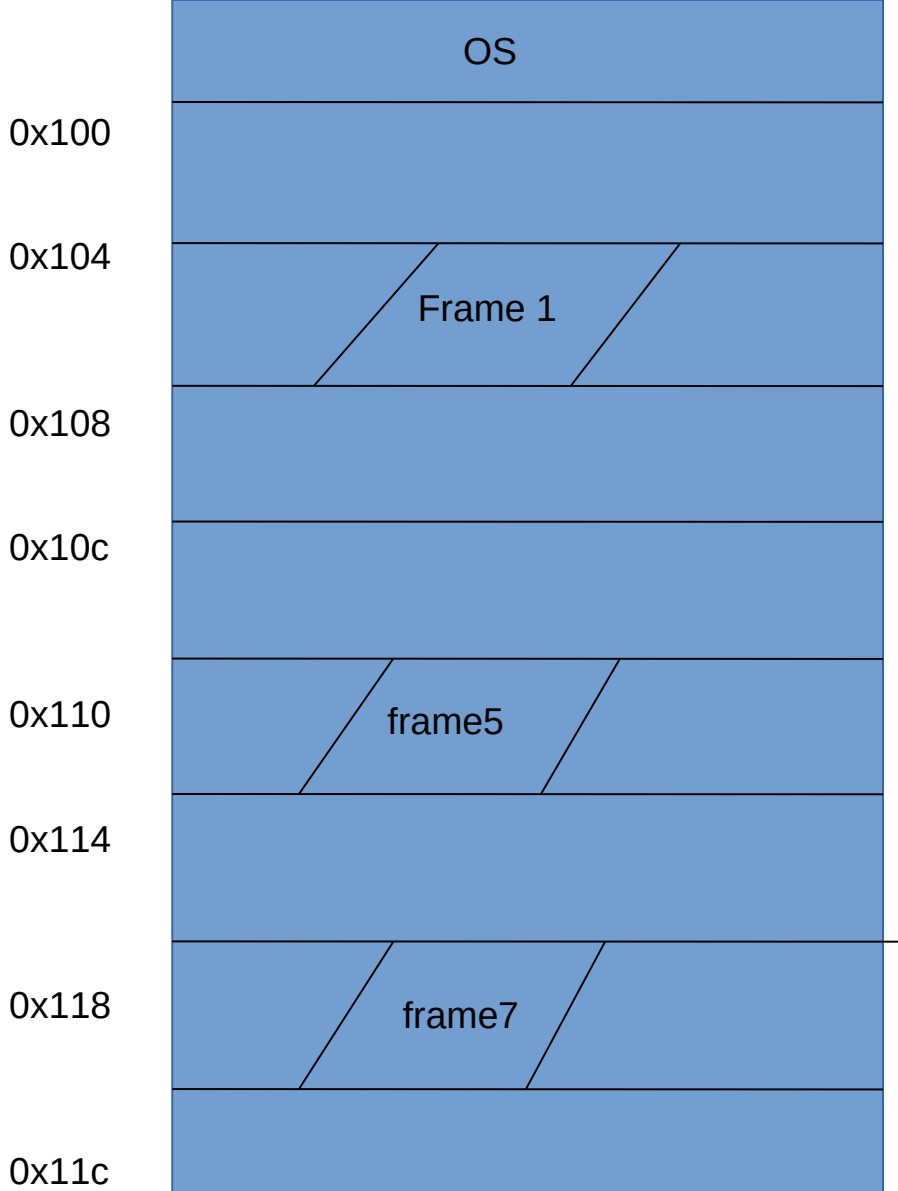


Page 0

Page 1

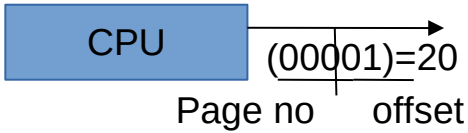
page	frame
0	0x100
1	0x108
2	0x10c
3	0x114
4	0x11c

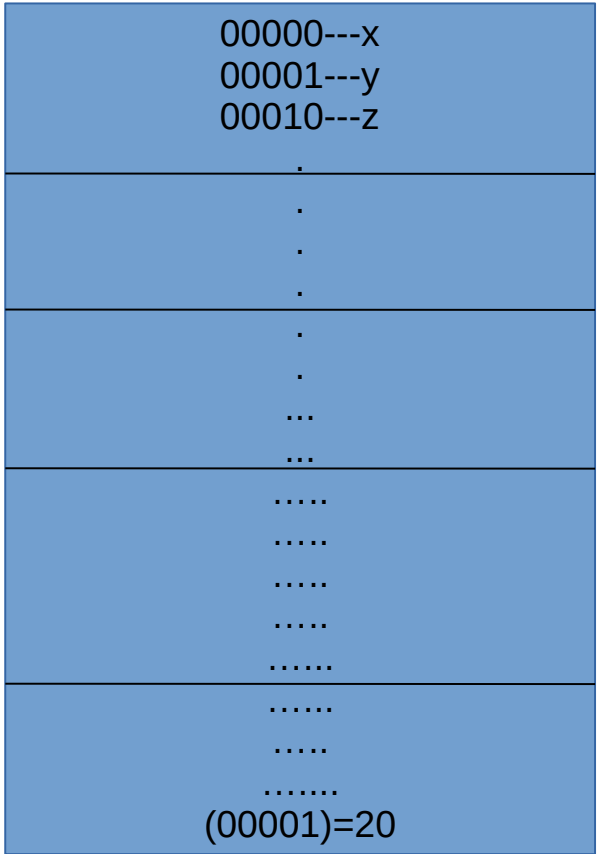
Page 4



Total logical space 20 bytes m=5

Page size= frame size=4 n=2



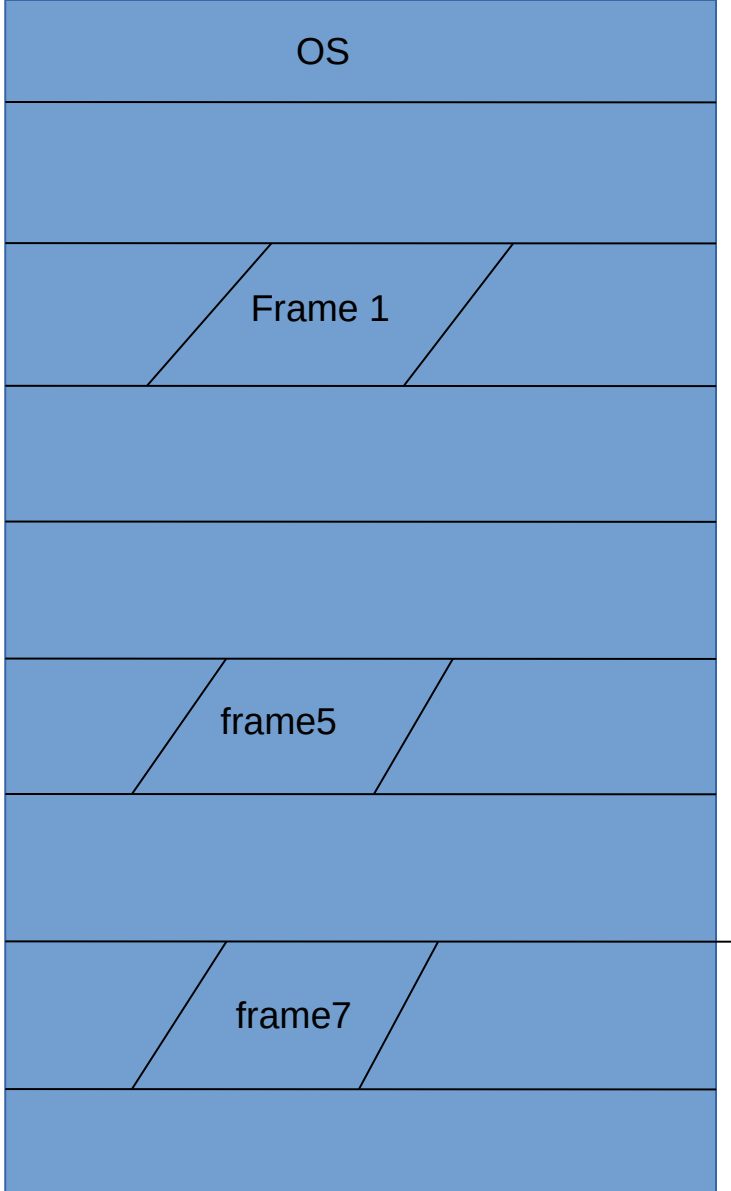


Page 0

Page 1

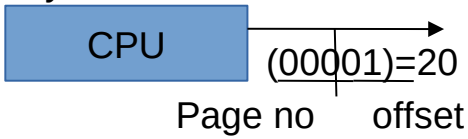
page	frame
0	0x100
1	0x108
2	0x10c
3	0x114
4	0x11c

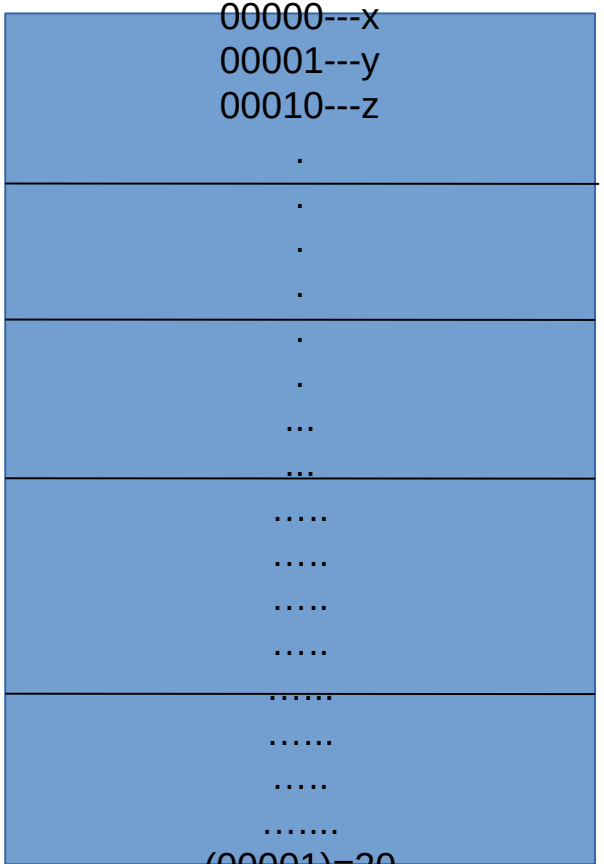
Page 4



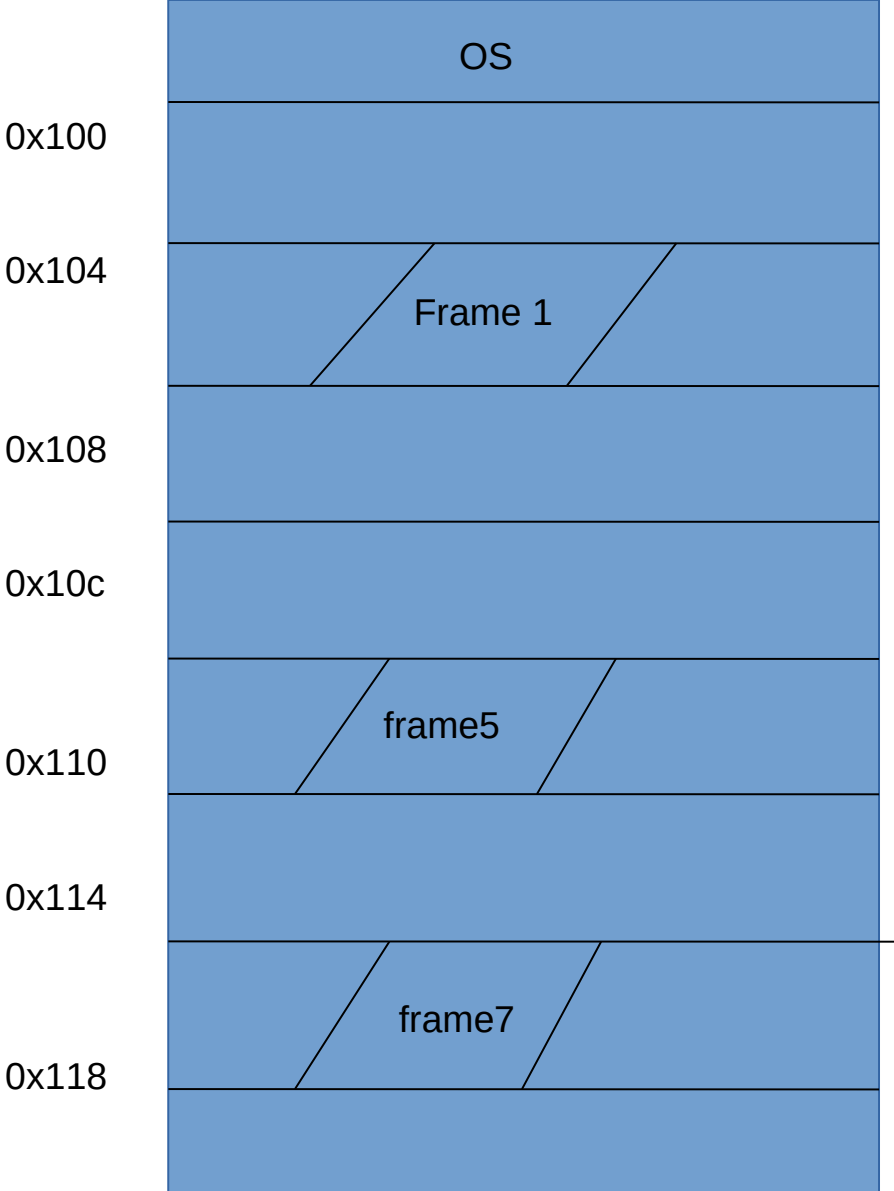
Total logical space 20 bytes m=5

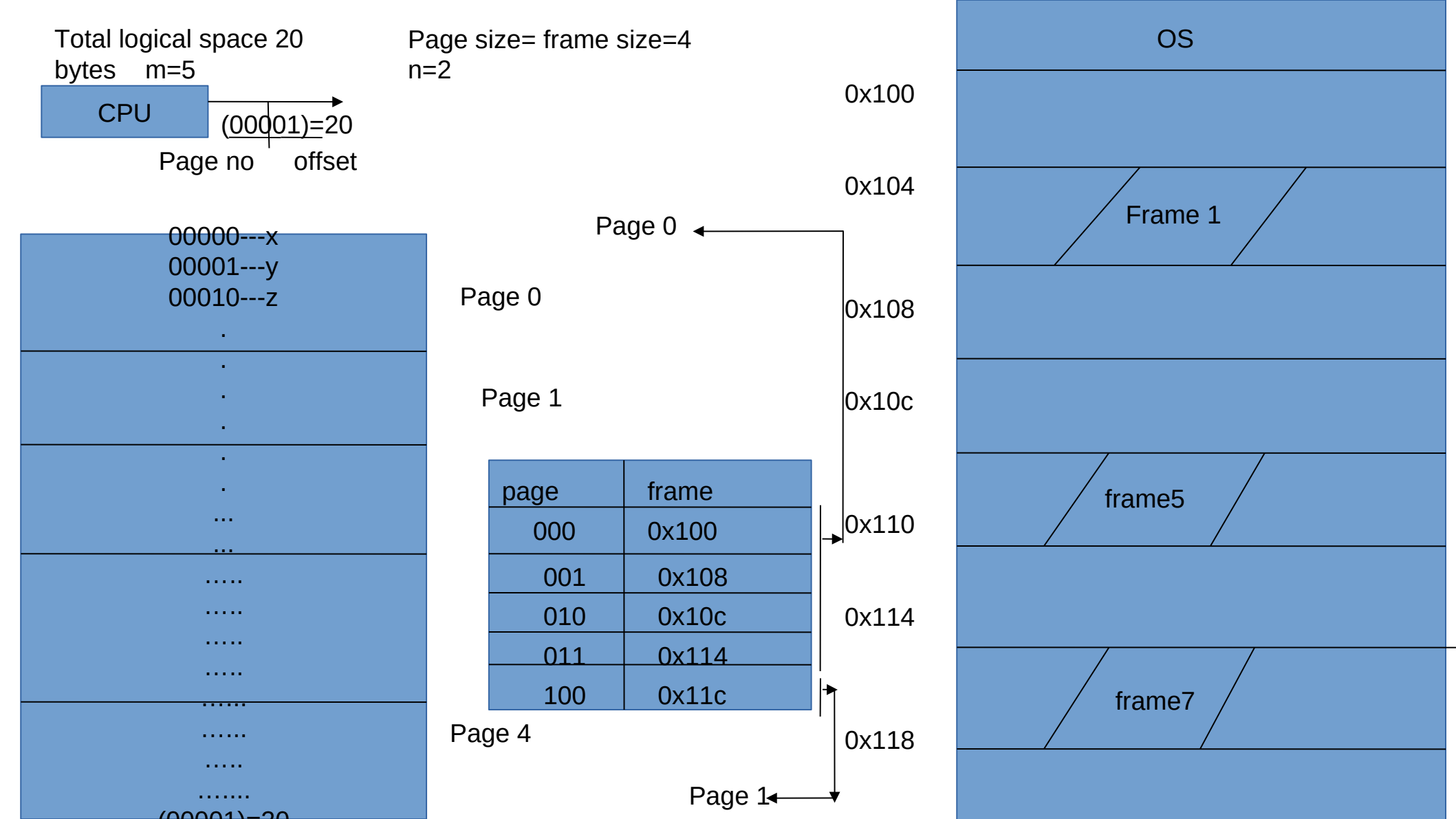
Page size= frame size=4
n=2

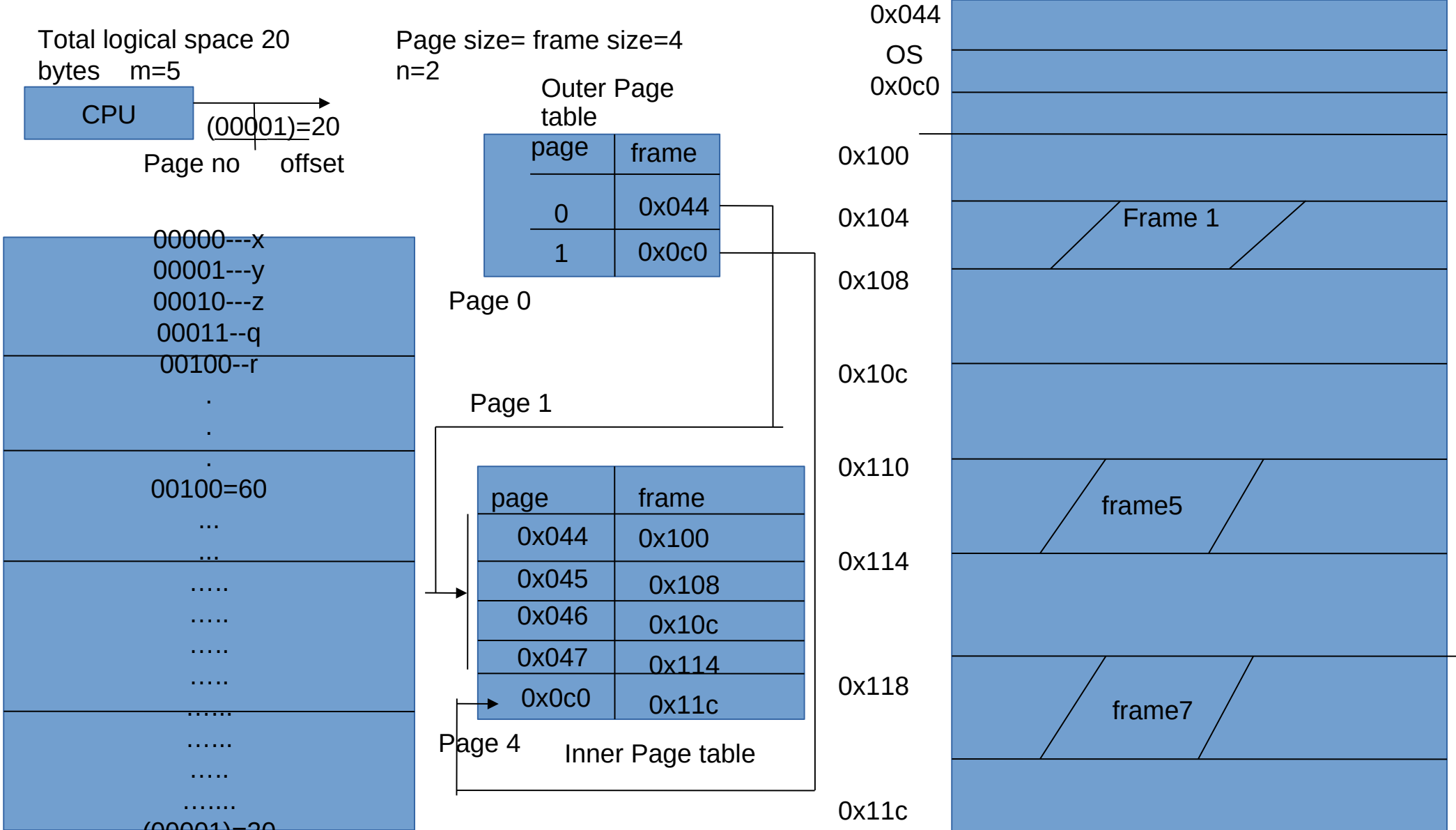




page	frame
00	0x100
01	0x108
10	0x10c
11	0x114
100	0x11c




Total logical space 20 bytes m=5
Page size= frame size=4 n=2
CPU
(00001)=20
Page no
offset
00000---x
00001---y
00010---z
0x100
0x104
0x108
0x10c
0x110
0x114
0x118
Page 0
Page 0
Page 1
page
frame
000
0x100
001
0x108
010
0x10c
011
0x114
100
0x11c
Page 4
Page 1
OS
Frame 1
frame5
frame7


Total logical space 20 bytes m=5
CPU
(00001)=20
Page no
offset
Page size= frame size=4
n=2
Outer Page table
page
frame
0
0x044
1
0x0c0
Page 0
00000---x
00001---y
00010---z
00011--q
00100--r
00100=60
Page 1
page
frame
0x044
0x100
0x045
0x108
0x046
0x10c
0x047
0x114
0x0c0
0x11c
Page 4
Inner Page table
0x044
OS
0x0c0
0x100
0x104
Frame 1
0x108
0x10c
0x110
frame5
0x114
0x118
frame7
0x11c

Disadvantage ?

Page table for each process consume lots of memory.....

Inverted page table

To overcome the disadvantages mention in the previous slides an inverted page table is used.

Inverted page table has one entry for each frame of memory.

Each entry consists of the logical(or virtual) address of the page stored in that memory location. With information about the process that owns it.

Therefore there is only one inverted page table in the system and it has only one entry for each frame of physical memory.

Each logical address in the system is consist of a triplet.

< process id, page number, offset>

each inverted page table entry is a pair<process id,page number>.

When a memory reference occurs part of the logical address consisting of <process id,page number> is presented to memory subsystem .

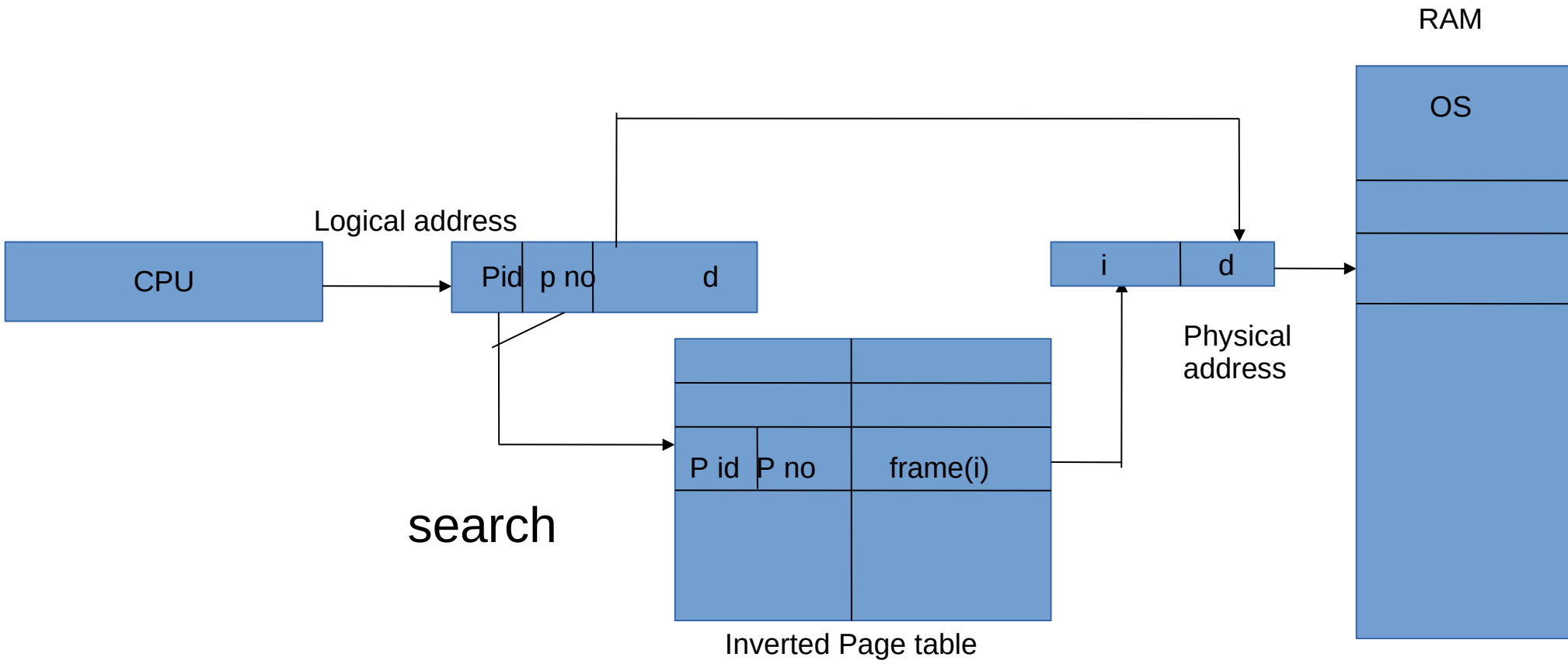
The inverted page table is then searched for this

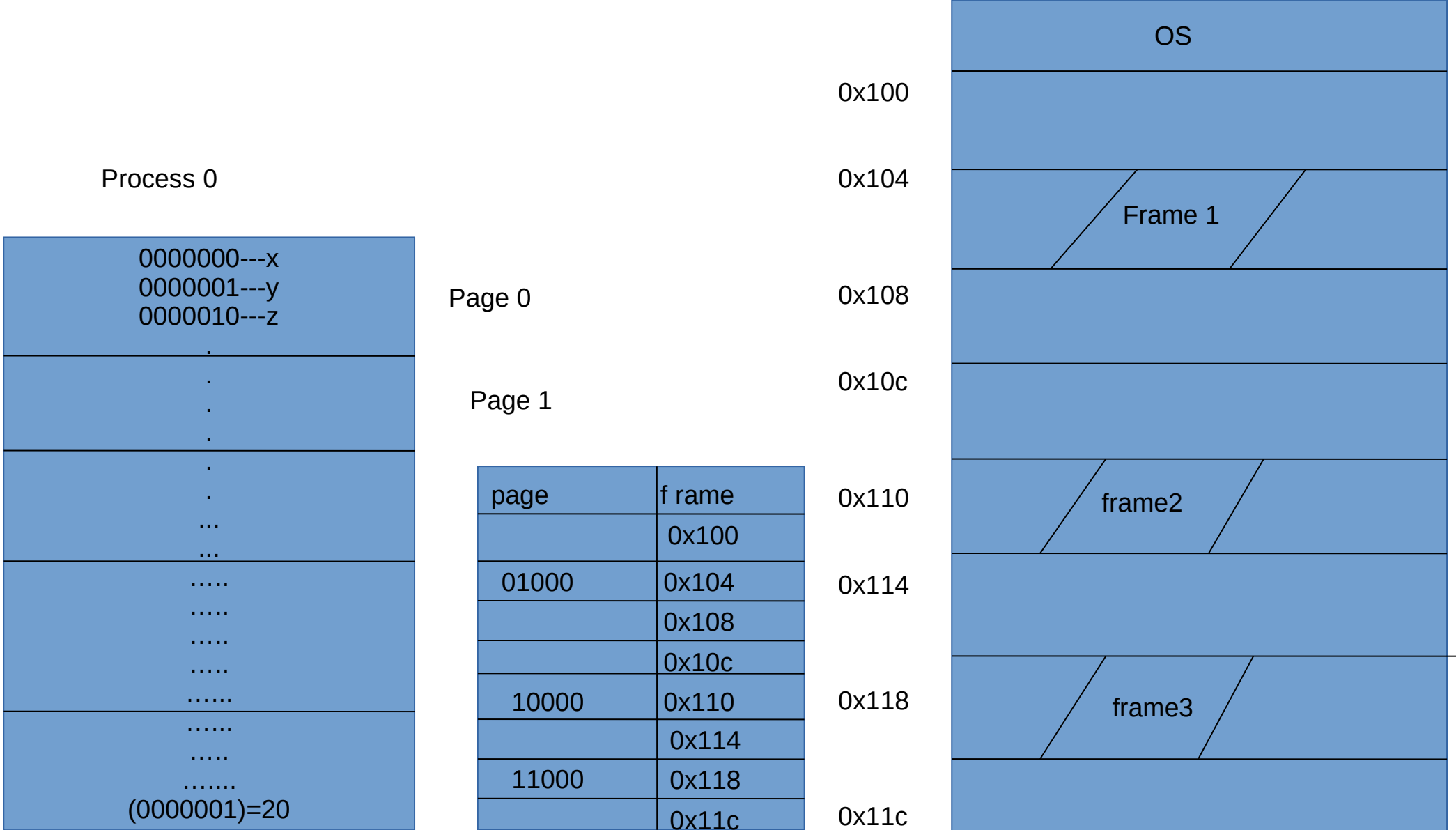
<process id,page number>.

If the match is found say at entry I then the physical address

<i, offset> is generated.

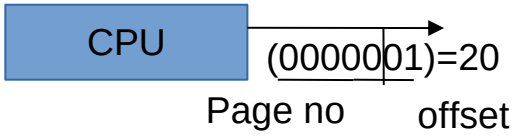
If no match is found an illegal address access has been attempted .


RAM
OS
Logical address
CPU
Pid
p no
d
i
d
Physical address
P id
P no
frame(i)
search
Inverted Page table


Process 0
0000000---x
0000001---y
0000010---z
.
.
.
.
.
.
...
...
…..
…..
…..
…..
……
……
…..
…….
(0000001)=20
Page 0
Page 1
page
f rame
0x100
01000
0x104
0x108
0x10c
10000
0x110
0x114
11000
0x118
0x11c
0x100
0x104
0x108
0x10c
0x110
0x114
0x118
0x11c
OS
Frame 1
frame2
frame3

Total logical space 20 bytes m=5+2

Page size= frame size=4 n=2

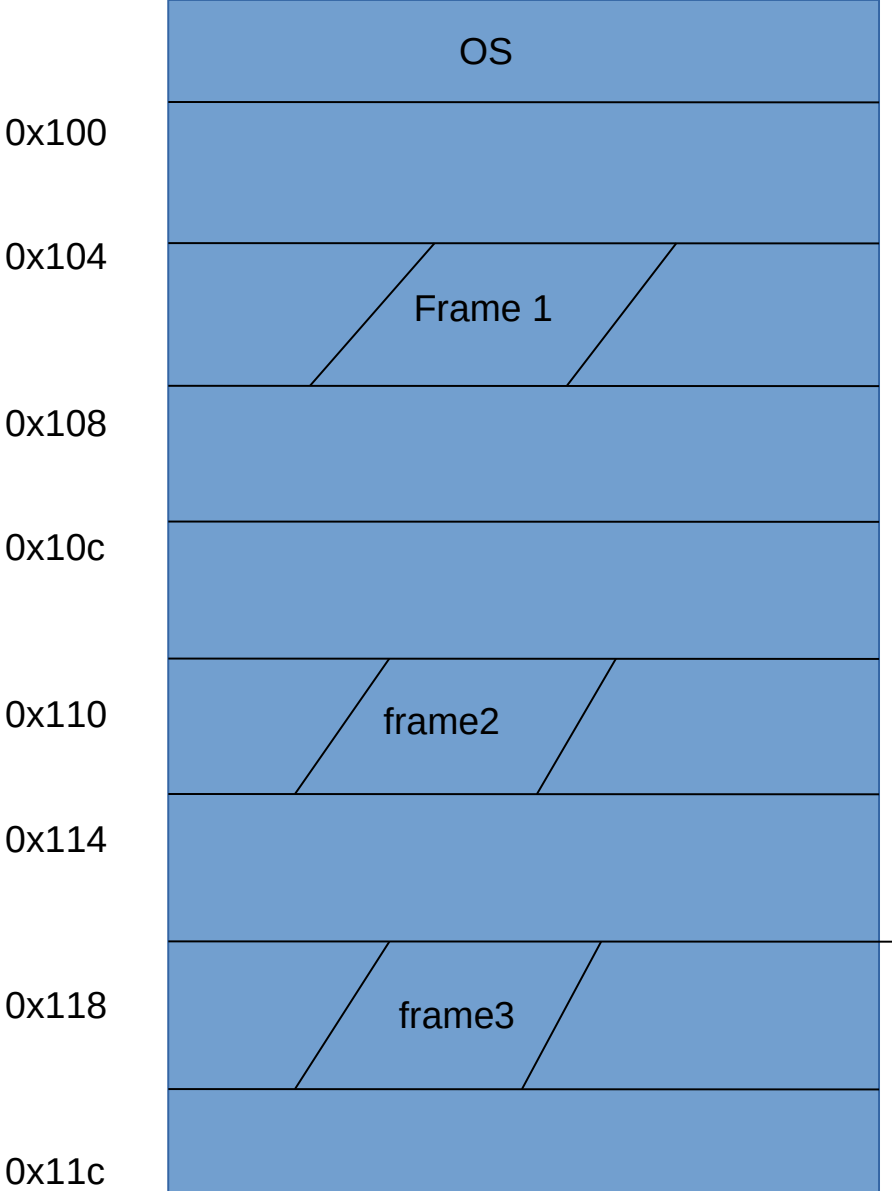


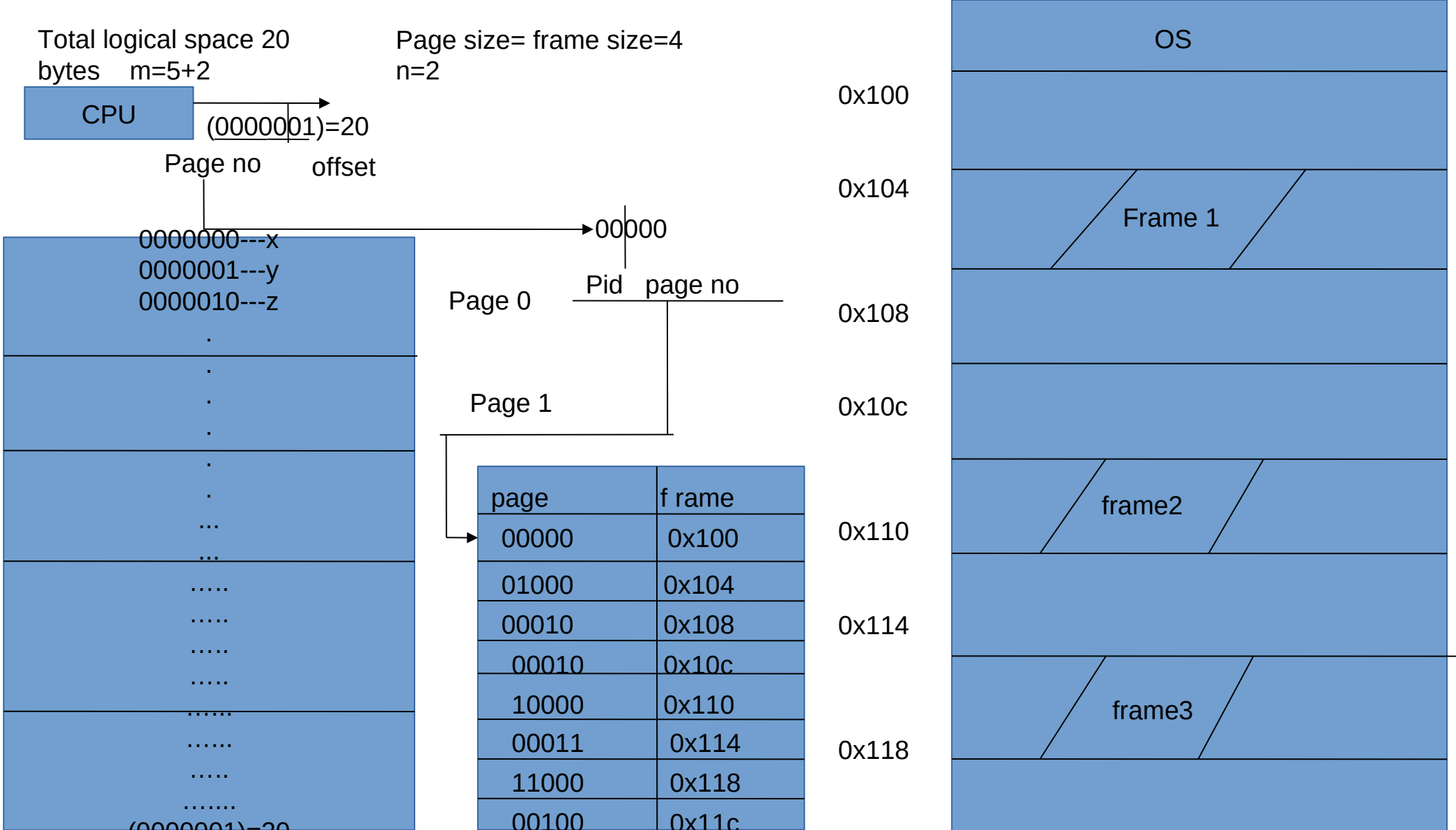
0000000---x
0000001---y
0000010---z
.
.
.
.
.
.
...
...
…..
…..
…..
…..
……
……
…..
…….
(0000001)=20

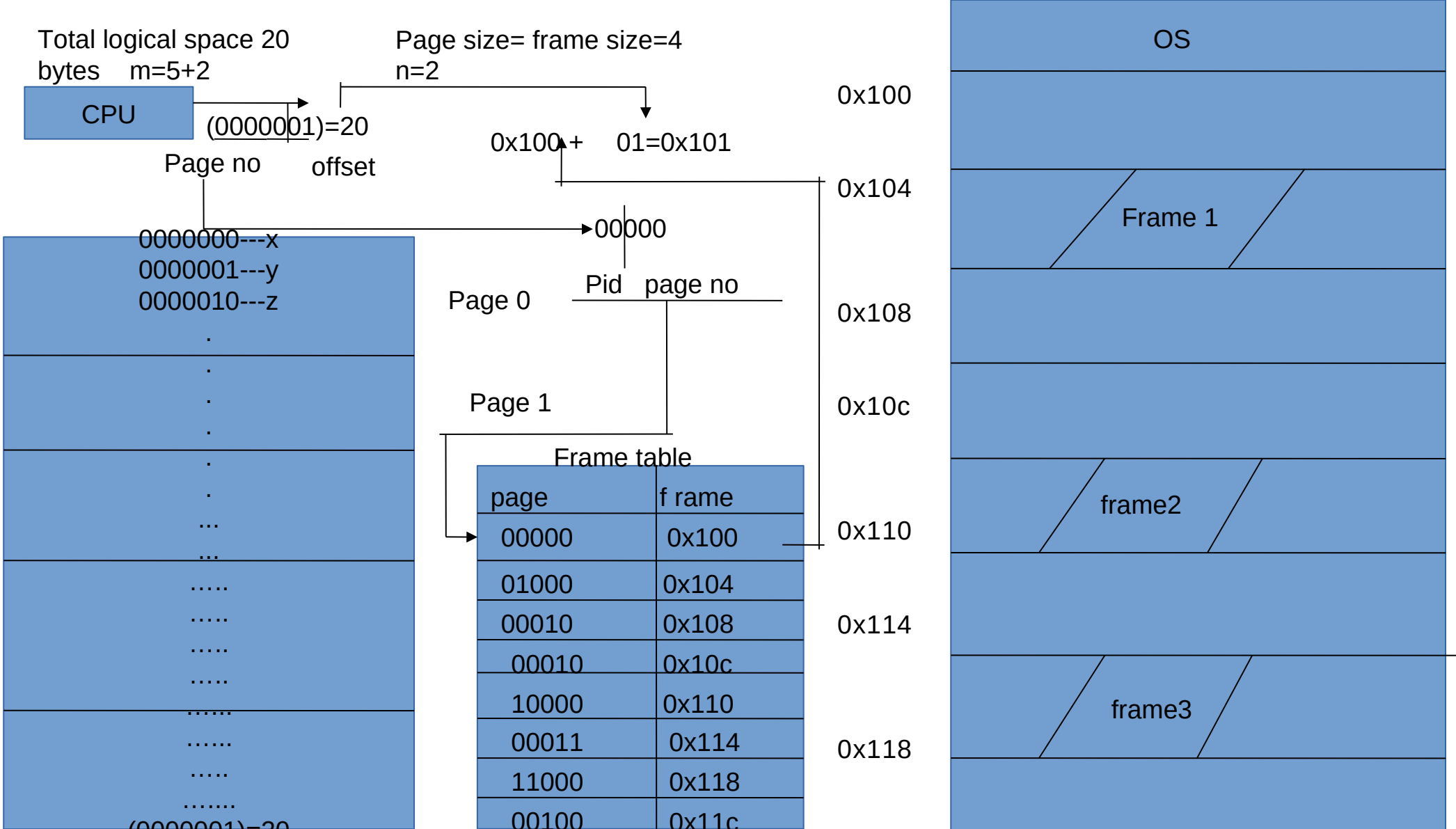
Page 0

Page 1

page	f rame
00000	0x100
01000	0x104
00001	0x108
00010	0x10c
10000	0x110
00011	0x114
11000	0x118
00100	0x11c




Total logical space 20 bytes m=5+2
Page size= frame size=4 n=2
CPU
(0000001)=20
Page no
offset
0000000---x
0000001---y
0000010---z
00000
Pid page no
Page 0
Page 1
page
f rame
00000
0x100
01000
0x104
00010
0x108
00010
0x10c
10000
0x110
00011
0x114
11000
0x118
00100
0x11c
0x100
0x104
0x108
0x10c
0x110
0x114
0x118
OS
Frame 1
frame2
frame3


Total logical space 20 bytes m=5+2
Page size= frame size=4 n=2
CPU
(0000001)=20
Page no
offset
0x100 + 01=0x101
0x100
0x104
0x108
0x10c
0x110
0x114
0x118
00000
Pid page no
Page 0
Page 1
Frame table
page
f rame
00000 0x100
01000 0x104
00010 0x108
00010 0x10c
10000 0x110
00011 0x114
11000 0x118
00100 0x11c
0000000---x
0000001---y
0000010---z
OS
Frame 1
frame2
frame3