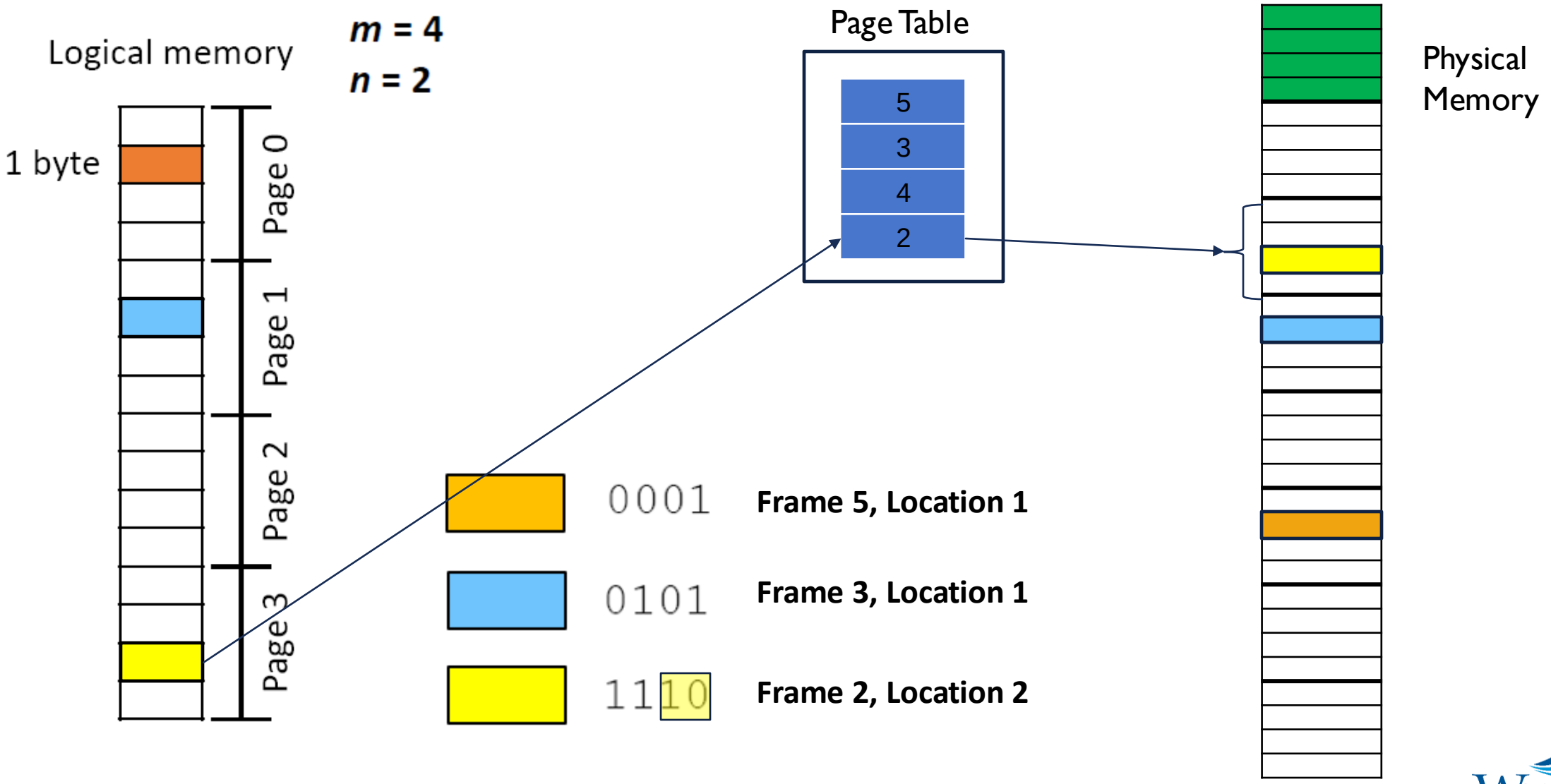
OPERATING SYSTEMS INTERNALS

CSCI 509

PAGING



FRAGMENTATION

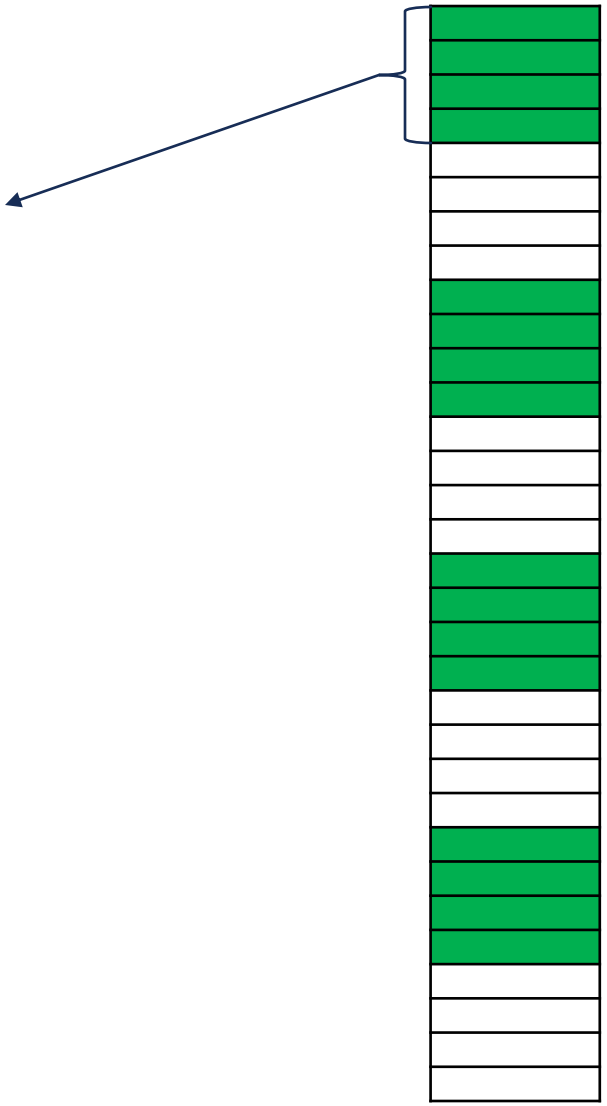
Q: Can we have external Fragmentation?

FRAGMENTATION

Q: Can we have external Fragmentation?

All memory holes are made up of a multiple of pages (1x , 2x , ..) since frame size = page size.

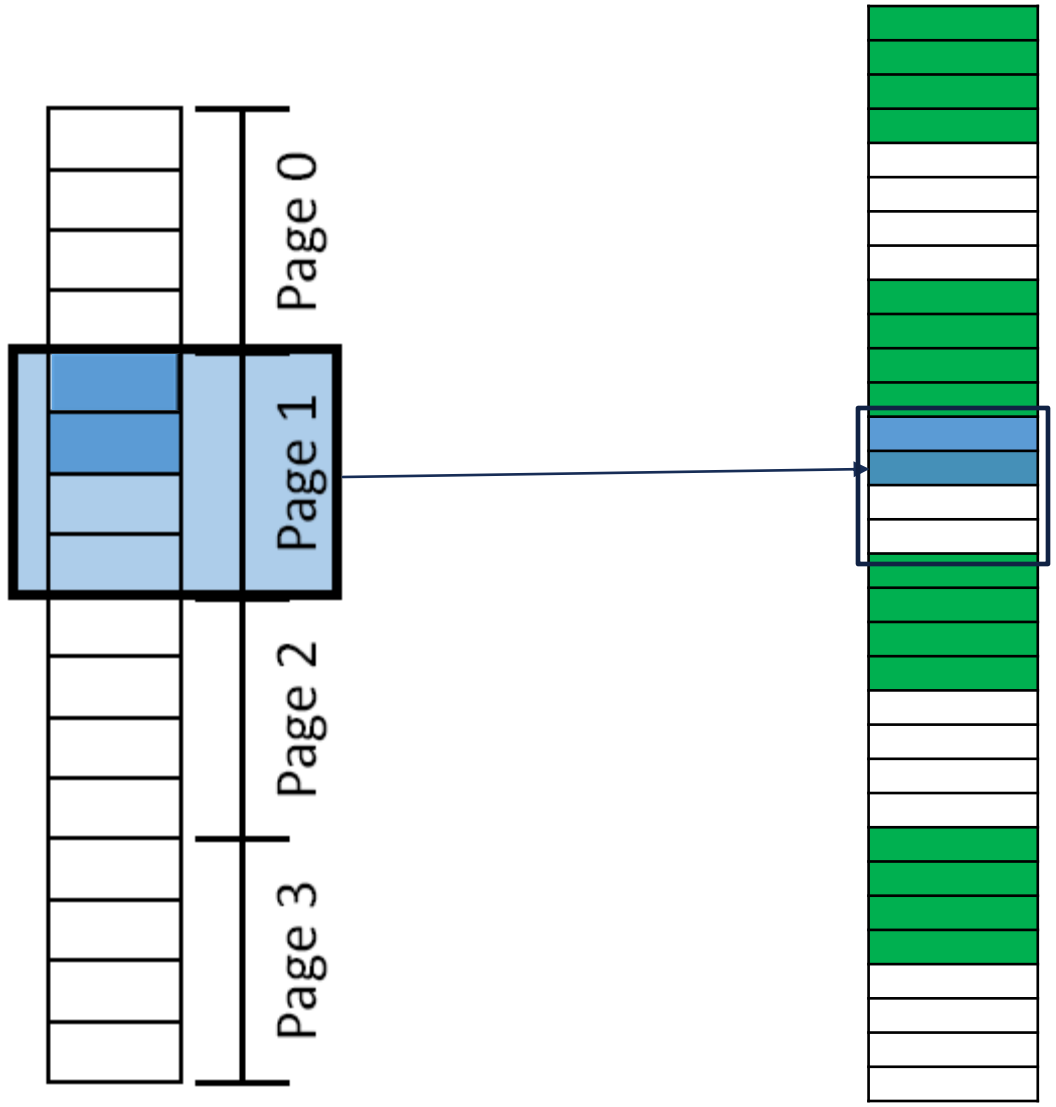
No External Fragmentation!



INTERNAL FRAGMENTATION

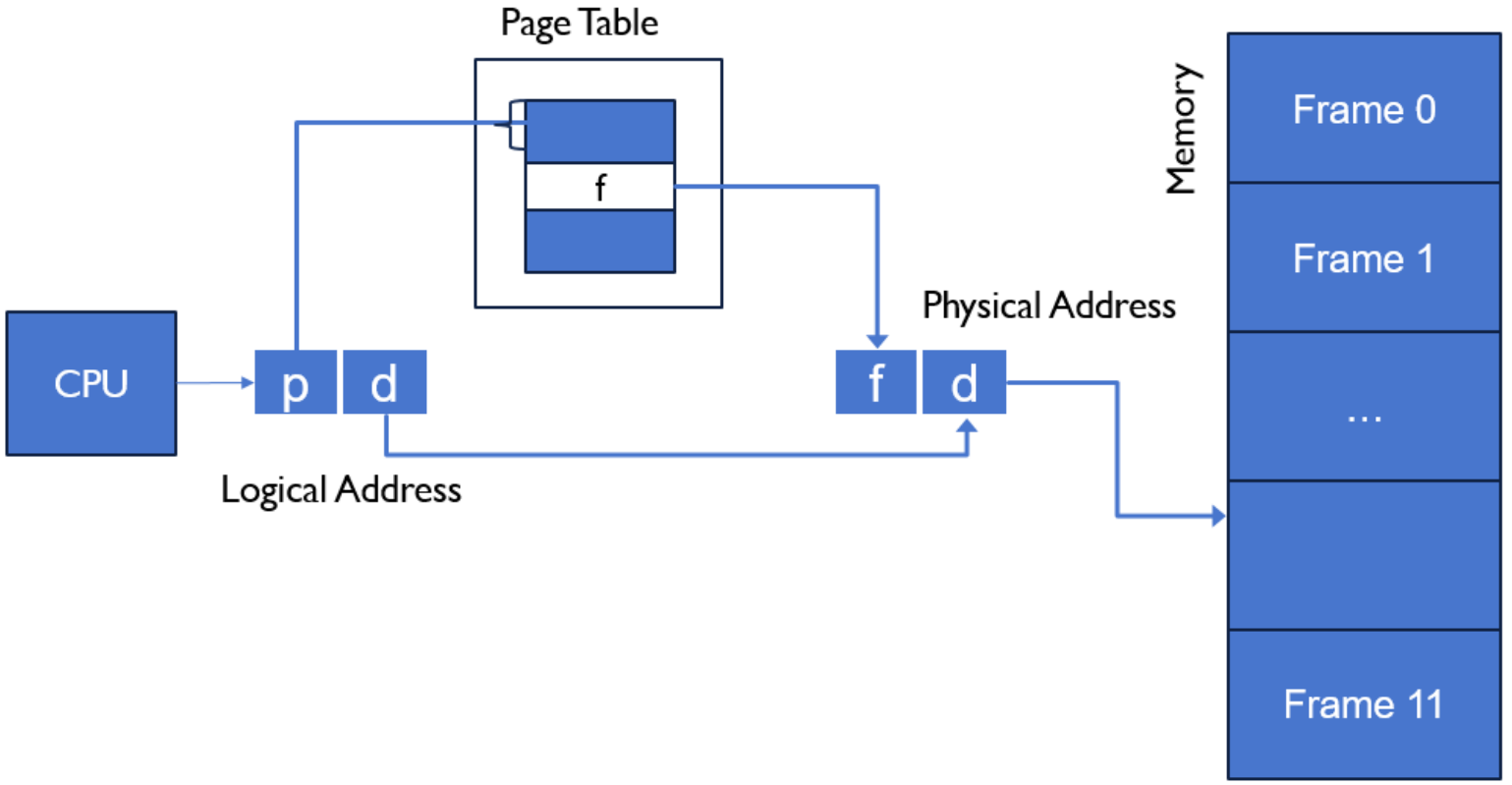
A process might not use its last page fully …

Internal fragmentation still exists.



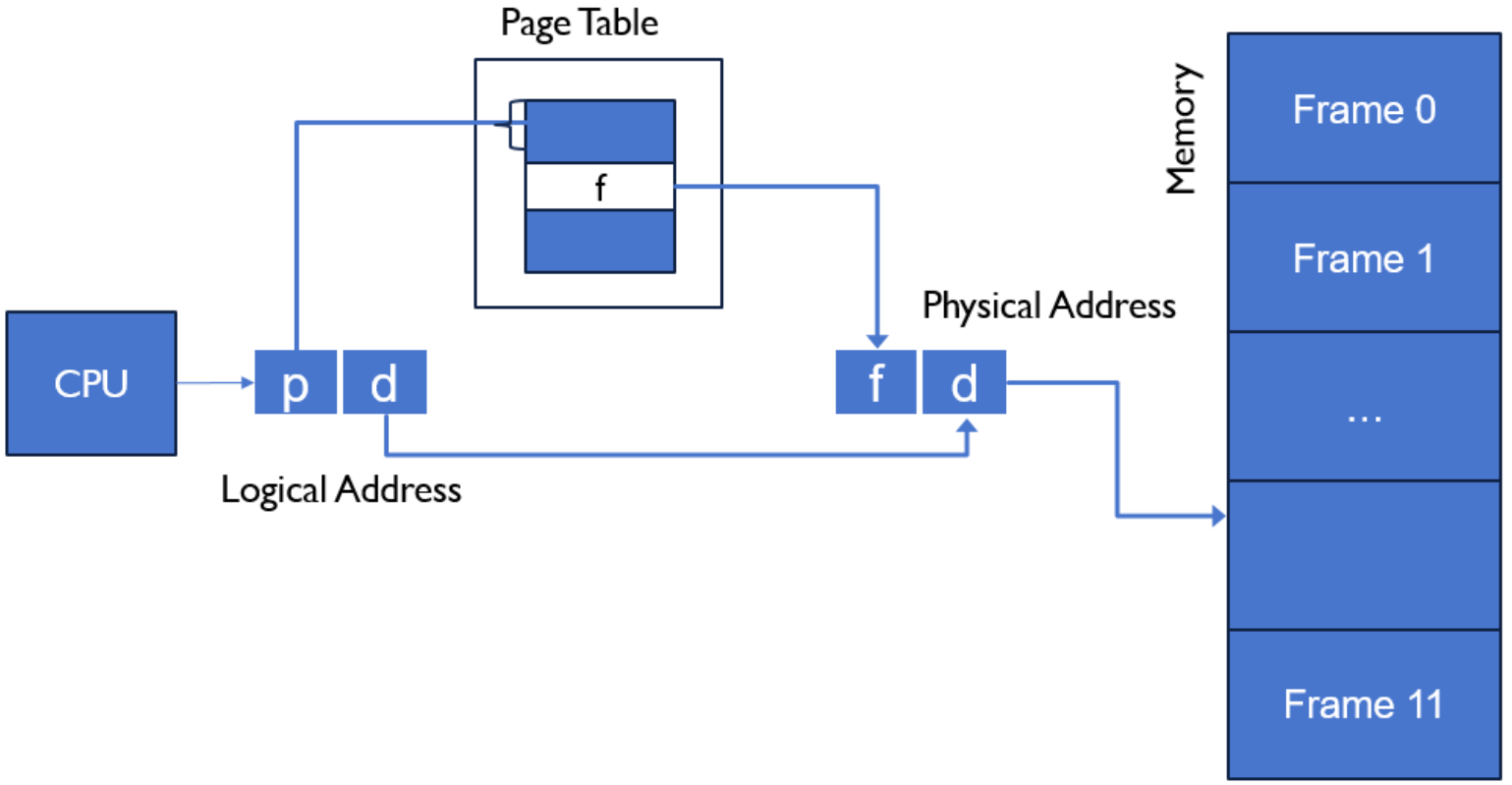
PAGE TABLE

- Every address resolution needs to go through the Page Table.
- How large is the page table?



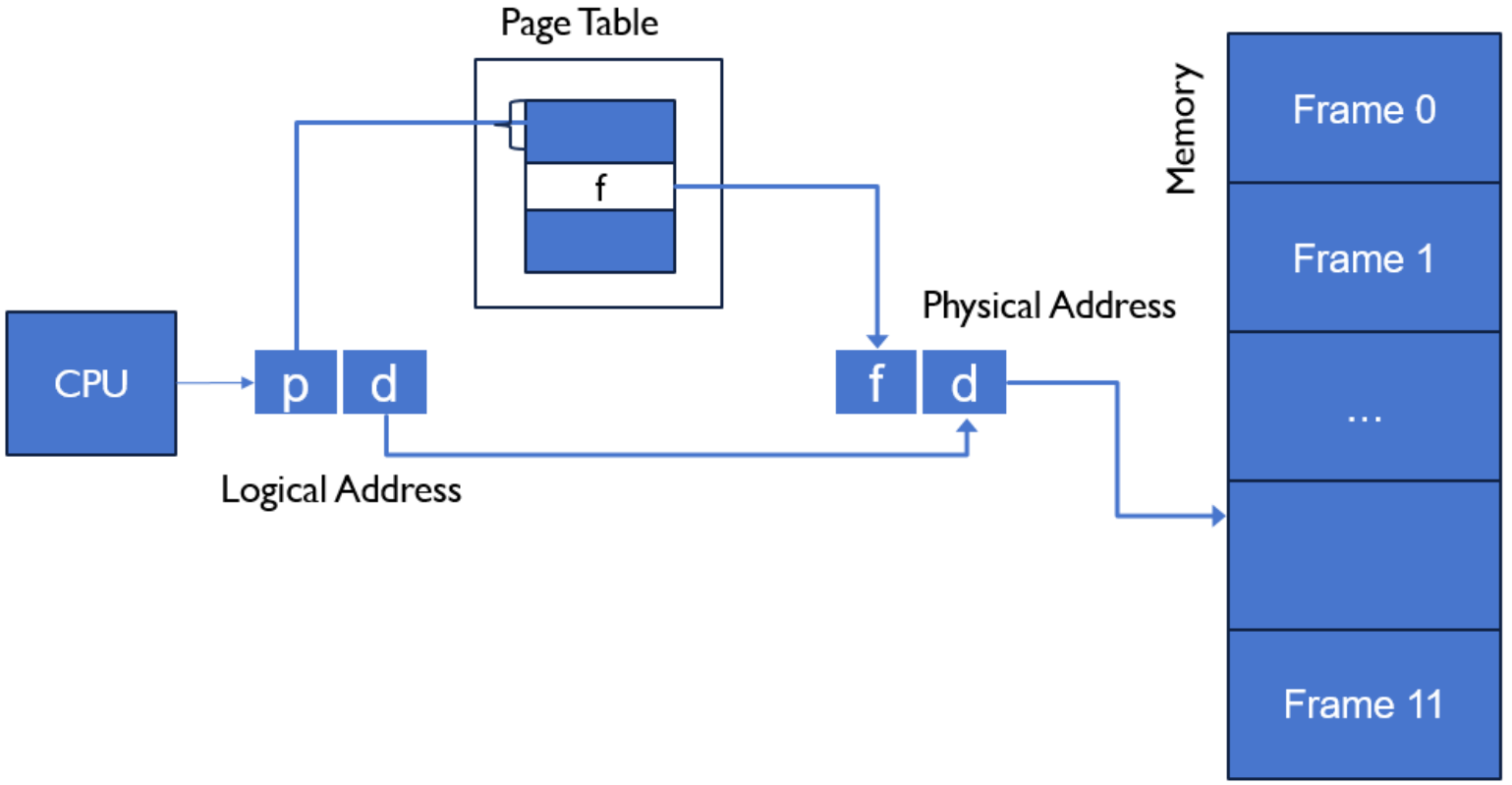
PAGE TABLE

- Assume 32-bit machine.
- If page size is 1 KB, how large would the page table be?



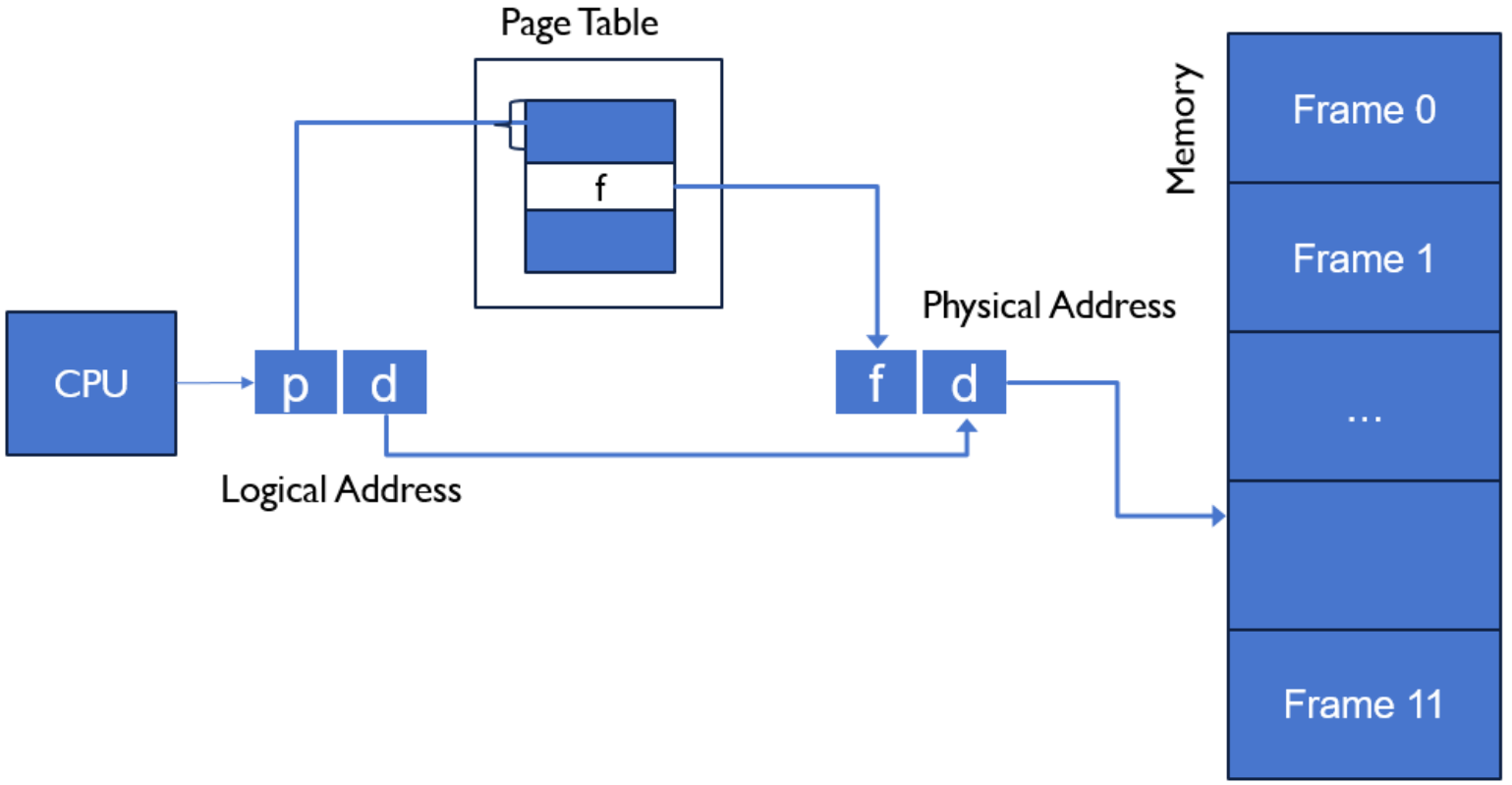
PAGE TABLE

- Assume 32-bit machine.
- If page size is 1 KB, how large would the page table be?
- Need to calculate m-n



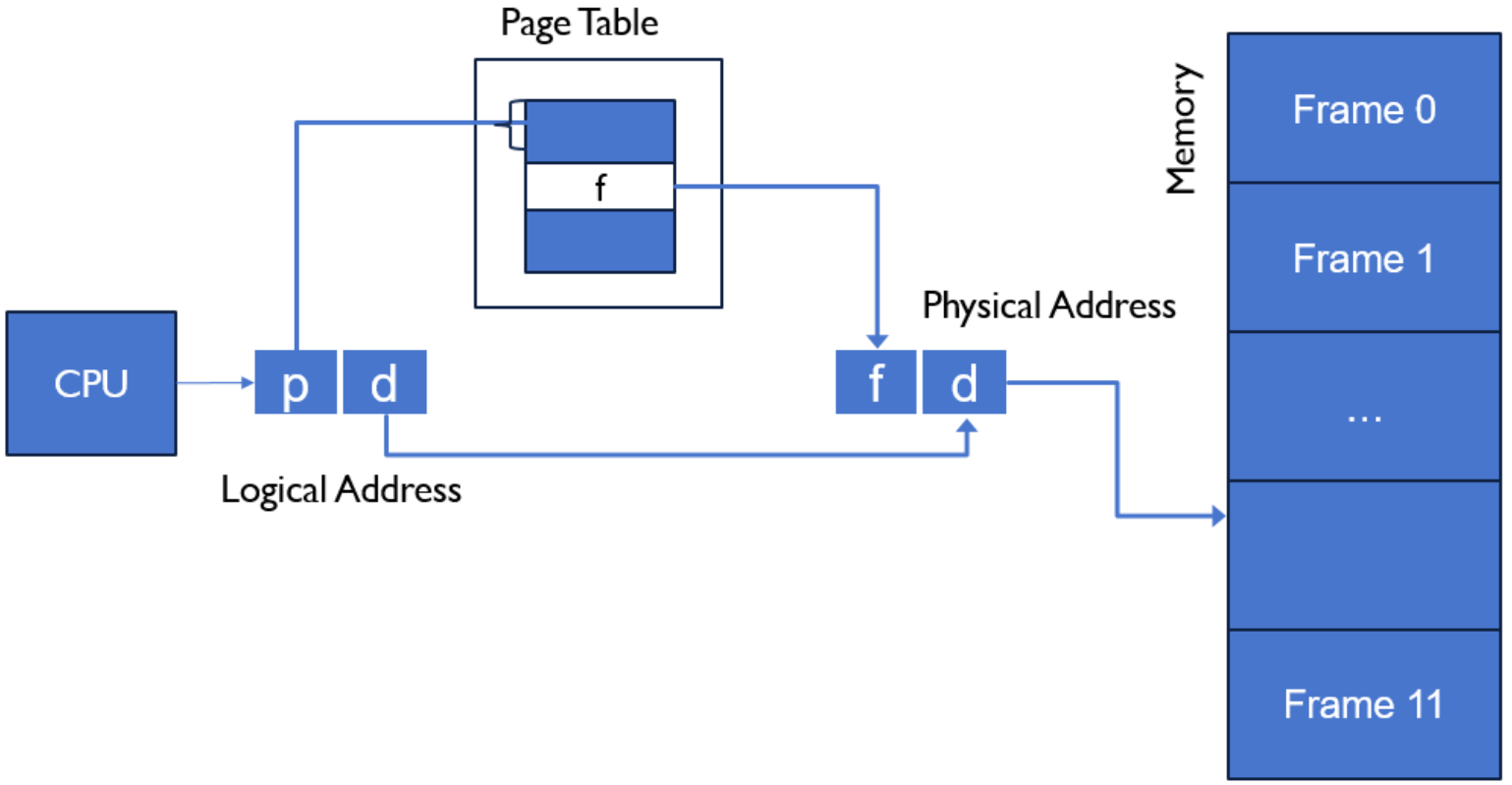
PAGE TABLE

- Assume 32-bit machine.
- If page size is 1 KB, how large would the page table be?
- Need to calculate m-n
- m = 32



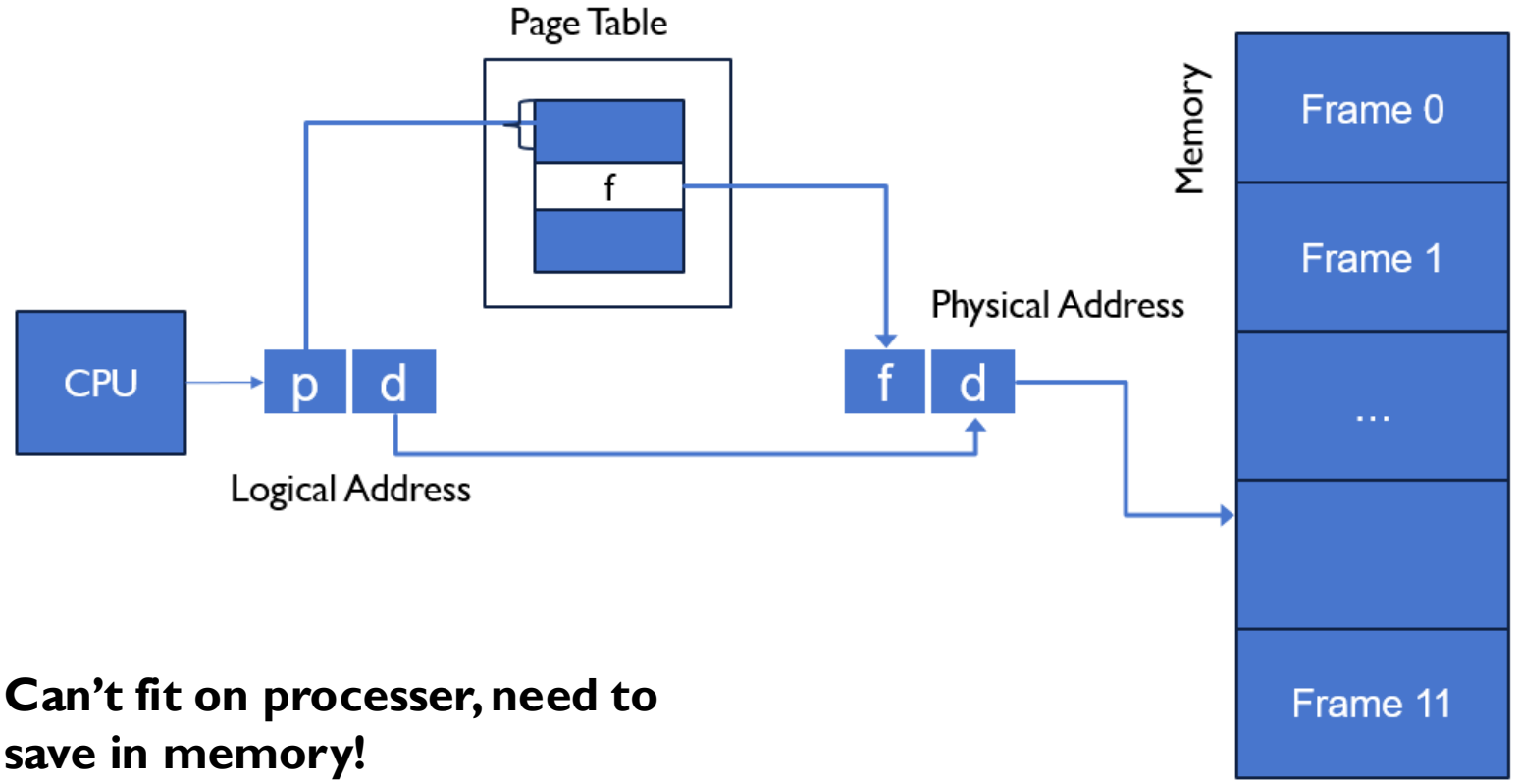
PAGE TABLE

- Assume 32-bit machine.
- If page size is 1 KB, how large would the page table be?
- Need to calculate m-n
- m = 32
- n = 10 = (2^10 = 1KB)



PAGE TABLE

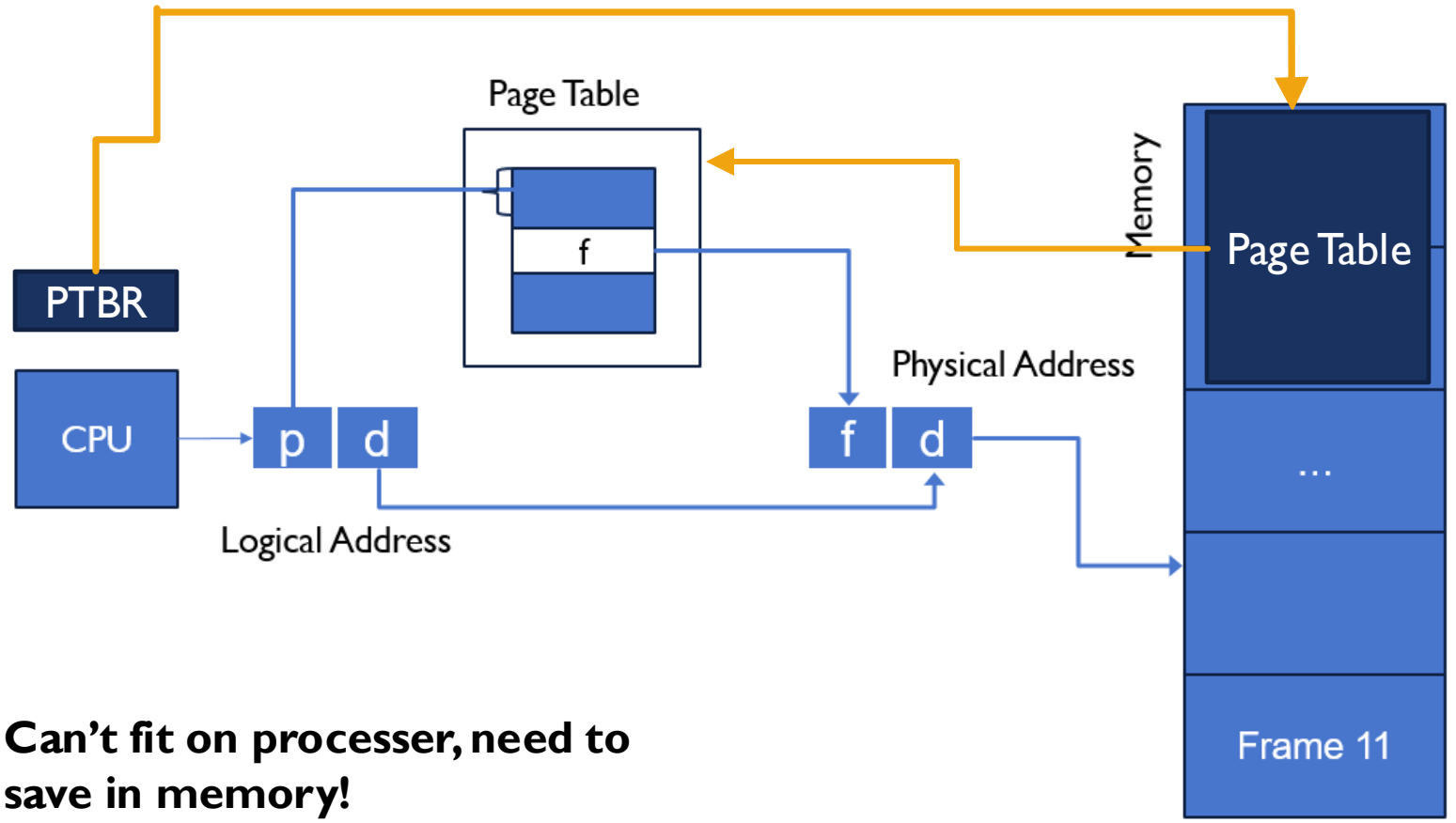
- Assume 32-bit machine.
- If page size is 1 KB, how large would the page table be?
- Need to calculate m-n
- m = 32
- n = 10 = (2^10 = 1KB)
- m-n = 22
- # entries in table: 2^{22}
- Size of table 4 bytes (32-bites) x 2^{22} MB = 16 MB.



Can't fit on processer, need to save in memory!

PAGE TABLE

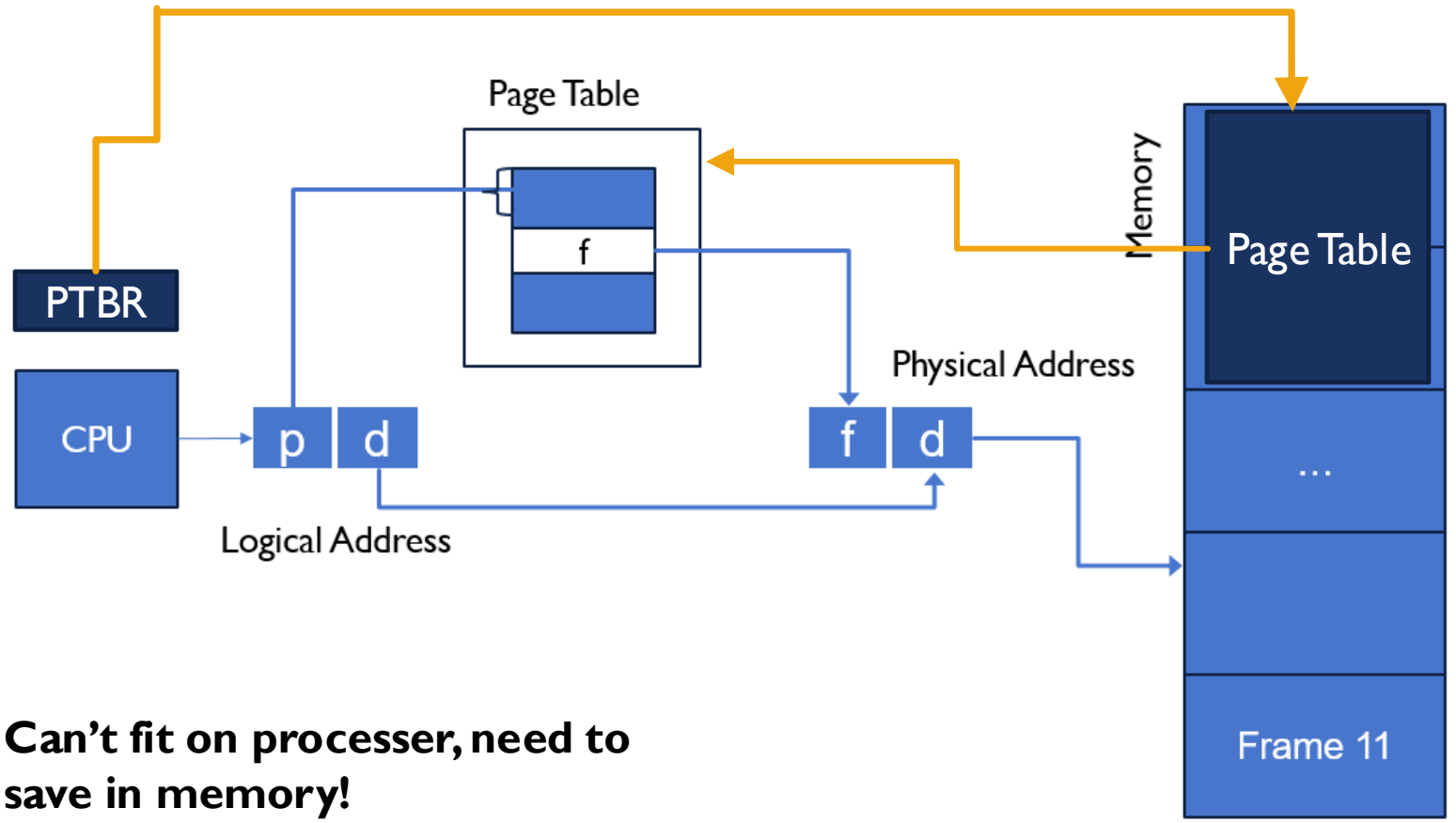
- Page table needs to be saved in memory.
- A page-table base register (PTBR) can be used to save the address of the page table.
- Entry is retrieved from memory.



Can't fit on processer, need to save in memory!

PAGE TABLE

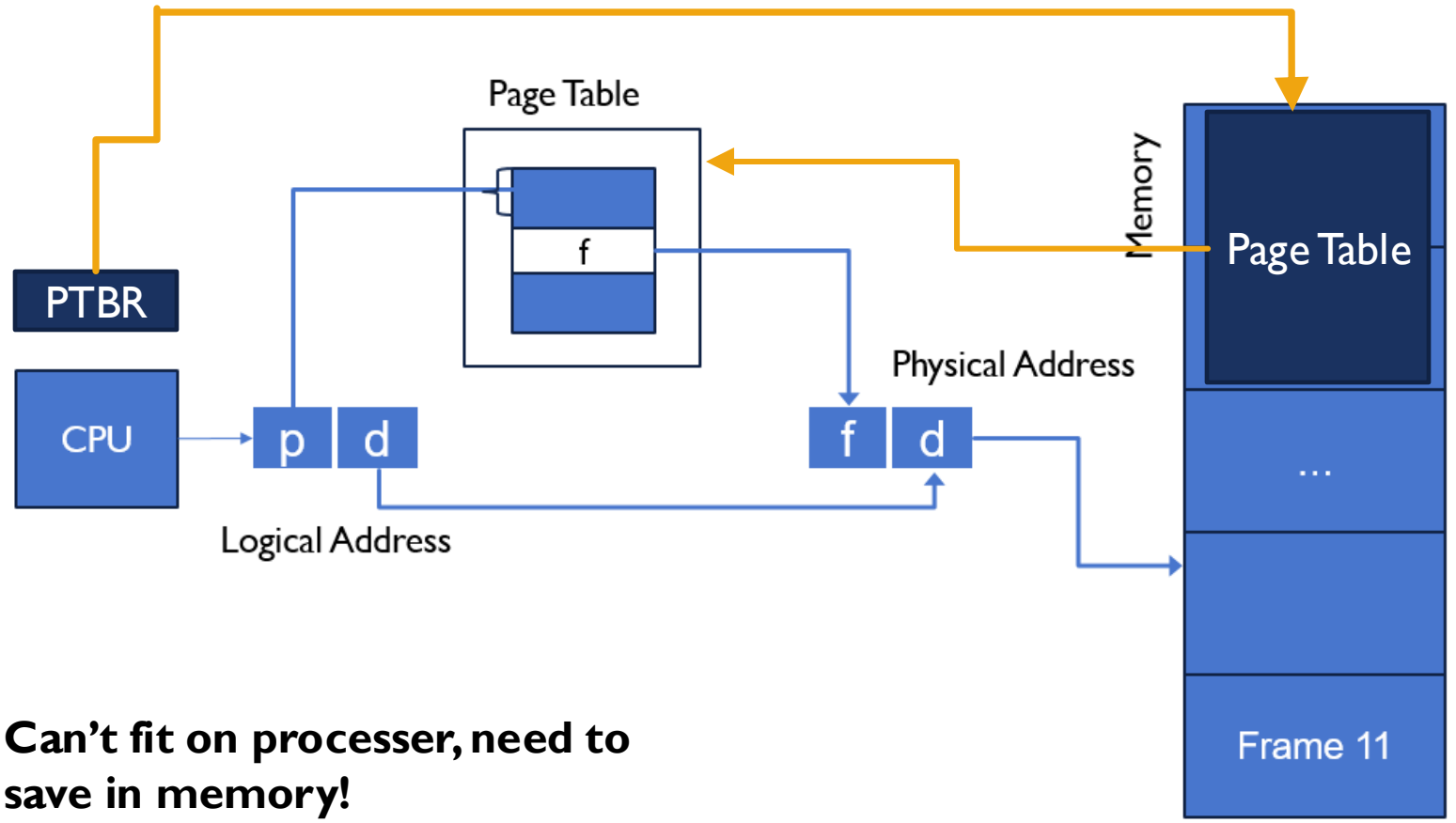
- Page table needs to be saved in memory.
- A page-table base register (PTBR) can be used to save the address of the page table.
- Entry is retrieved from memory.
- **Q: What are the disadvantages of this approach?**



Can't fit on processer, need to save in memory!

PAGE TABLE

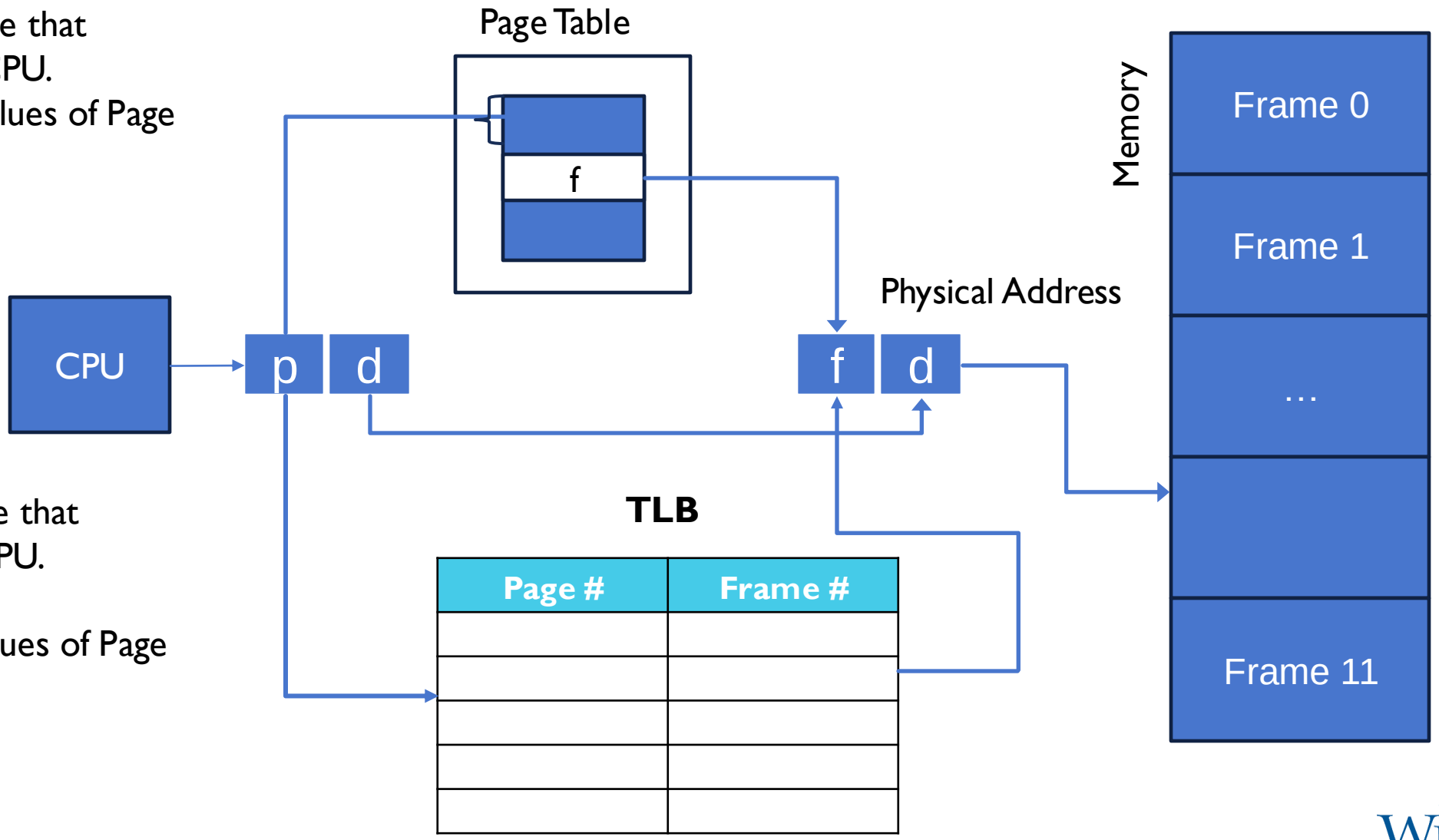
- Page table needs to be saved in memory.
- A page-table base register (PTBR) can be used to save the address of the page table.
- Entry is retrieved from memory.
- **Q: What are the disadvantages of this approach?**
- **Extremely slow!**



Can't fit on processer, need to save in memory!

TRANSLATION LOOK ASIDE BUFFER

- TLB: A small table that resides on the CPU.
- Caches in the values of Page table.

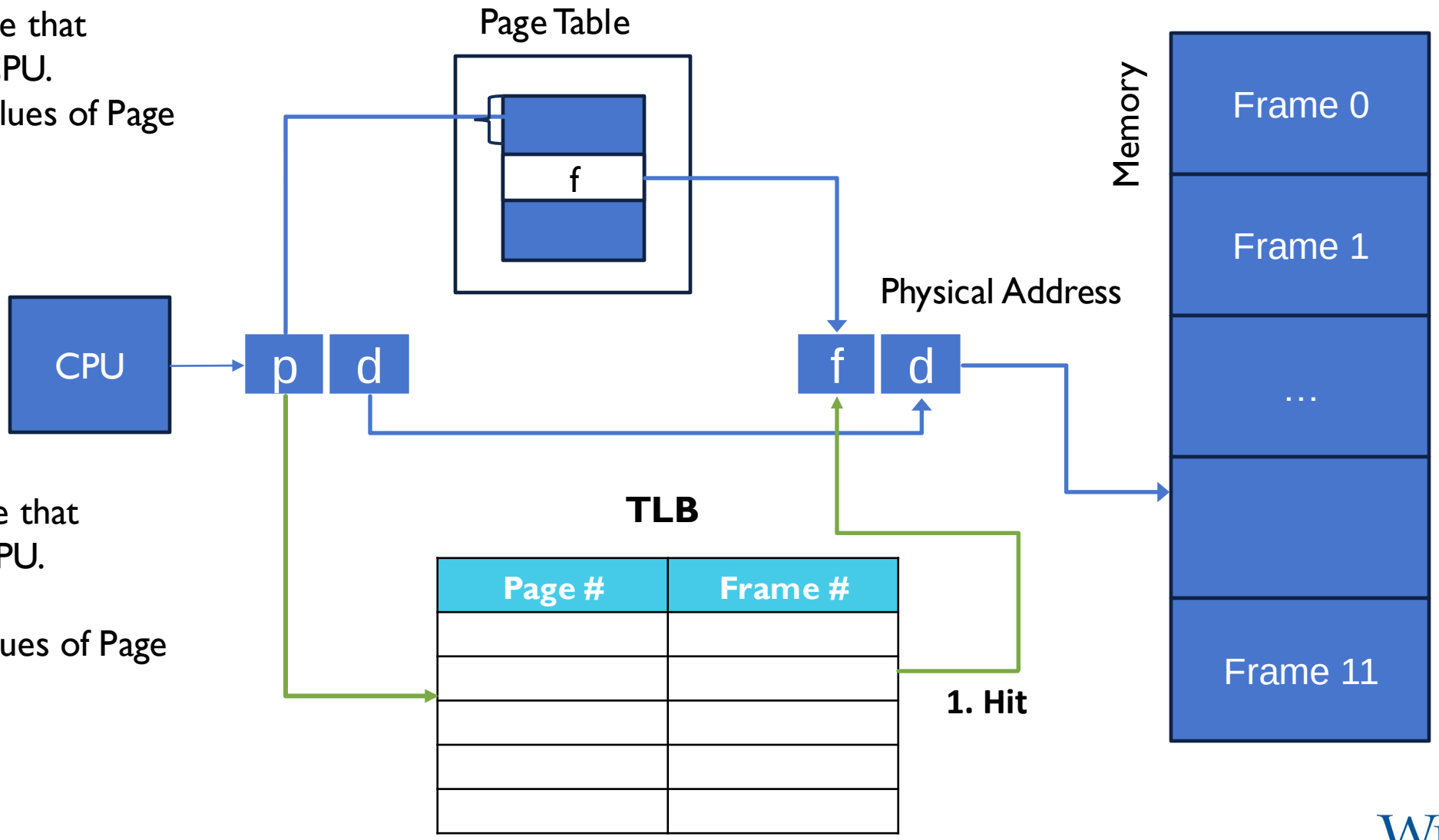


TLB: A small table that resides on the CPU.

Caches in the values of Page table.

TRANSLATION LOOK ASIDE BUFFER

- TLB: A small table that resides on the CPU.
- Caches in the values of Page table.

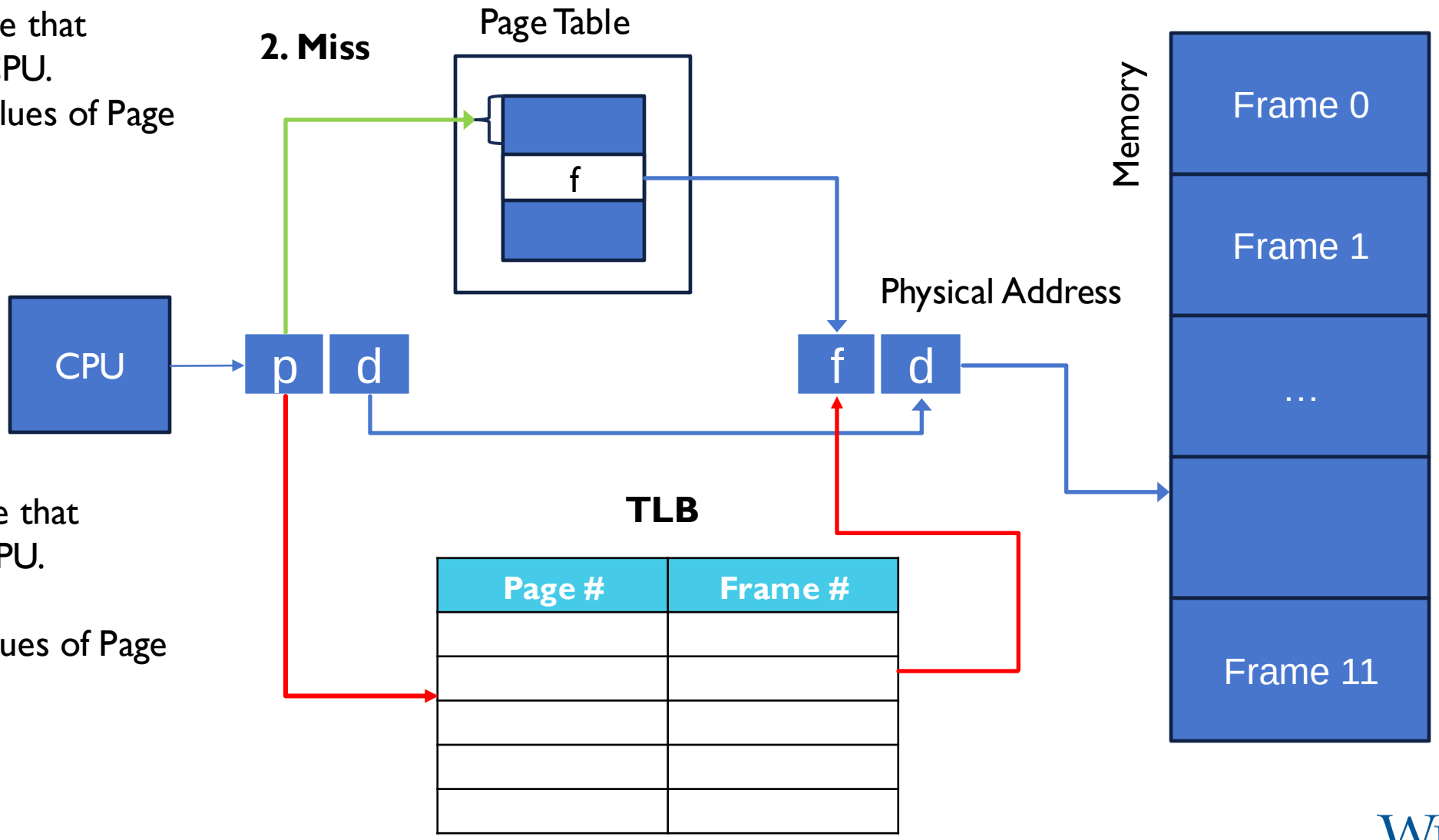


TLB: A small table that resides on the CPU.

Caches in the values of Page table.

TRANSLATION LOOK ASIDE BUFFER

- TLB: A small table that resides on the CPU.
- Caches in the values of Page table.

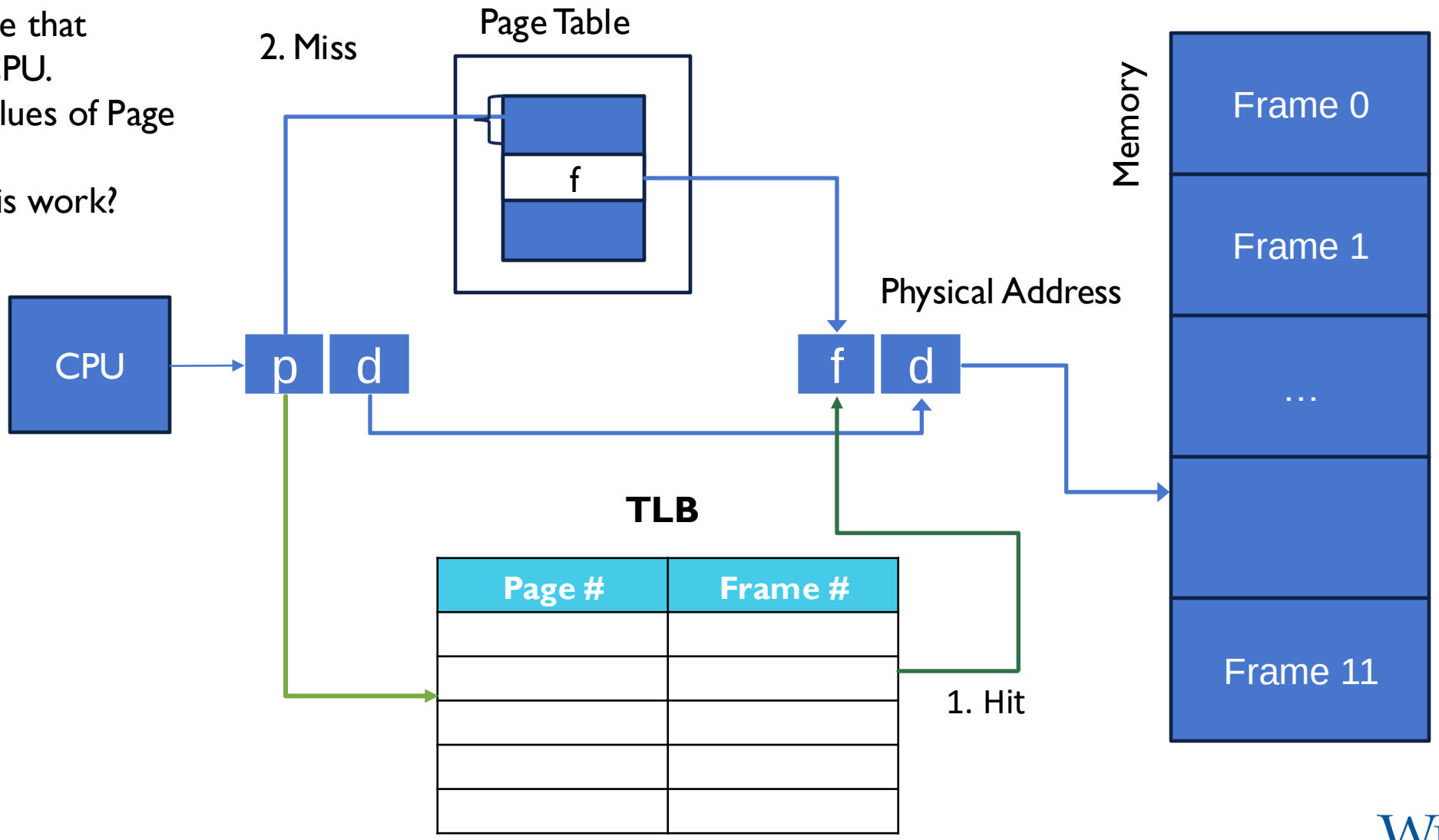


TLB: A small table that resides on the CPU.

Caches in the values of Page table.

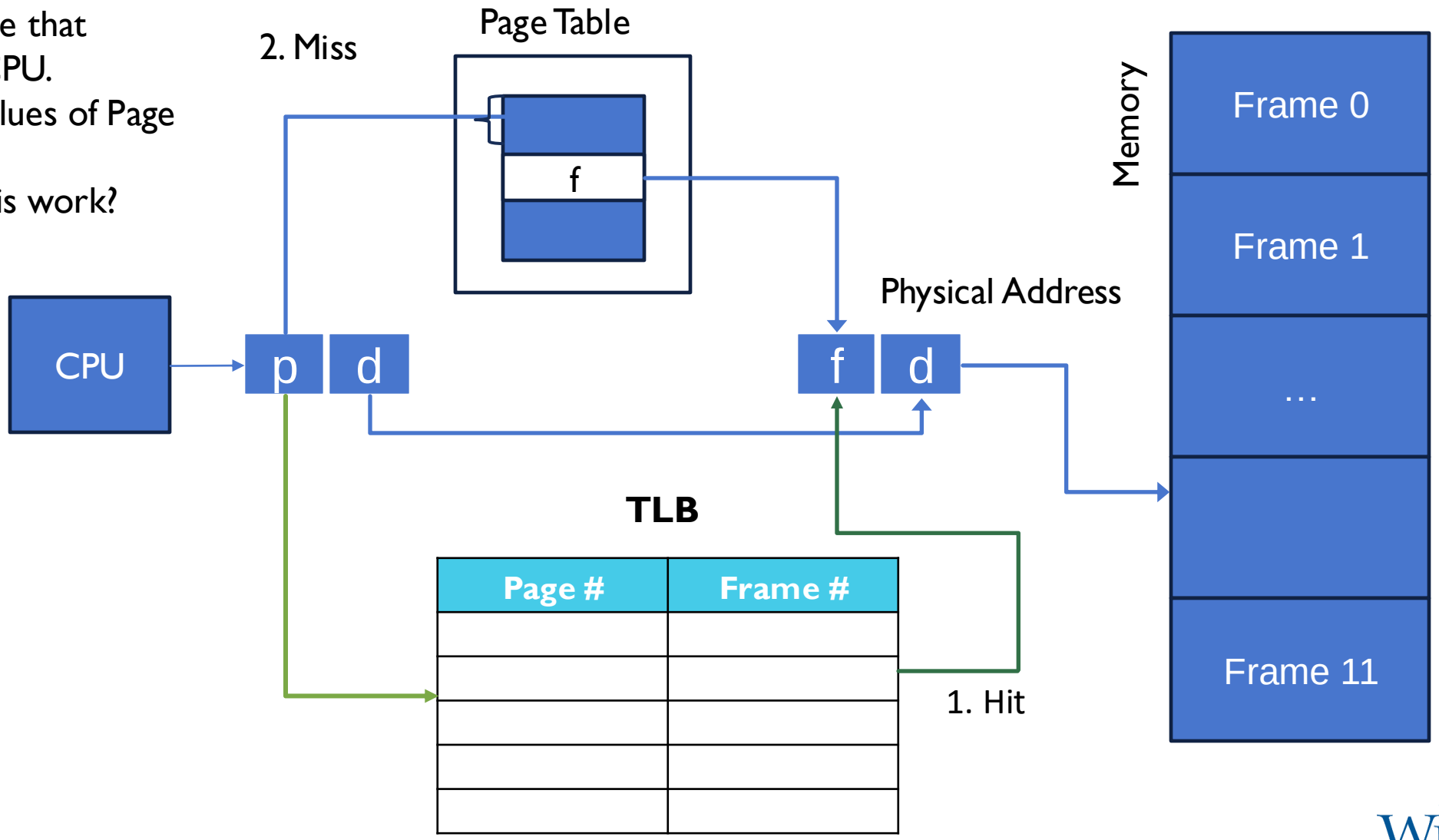
TRANSLATION LOOK ASIDE BUFFER

- TLB: A small table that resides on the CPU.
- Caches in the values of Page table.
- **Q:** Why does this work?



TRANSLATION LOOK ASIDE BUFFER

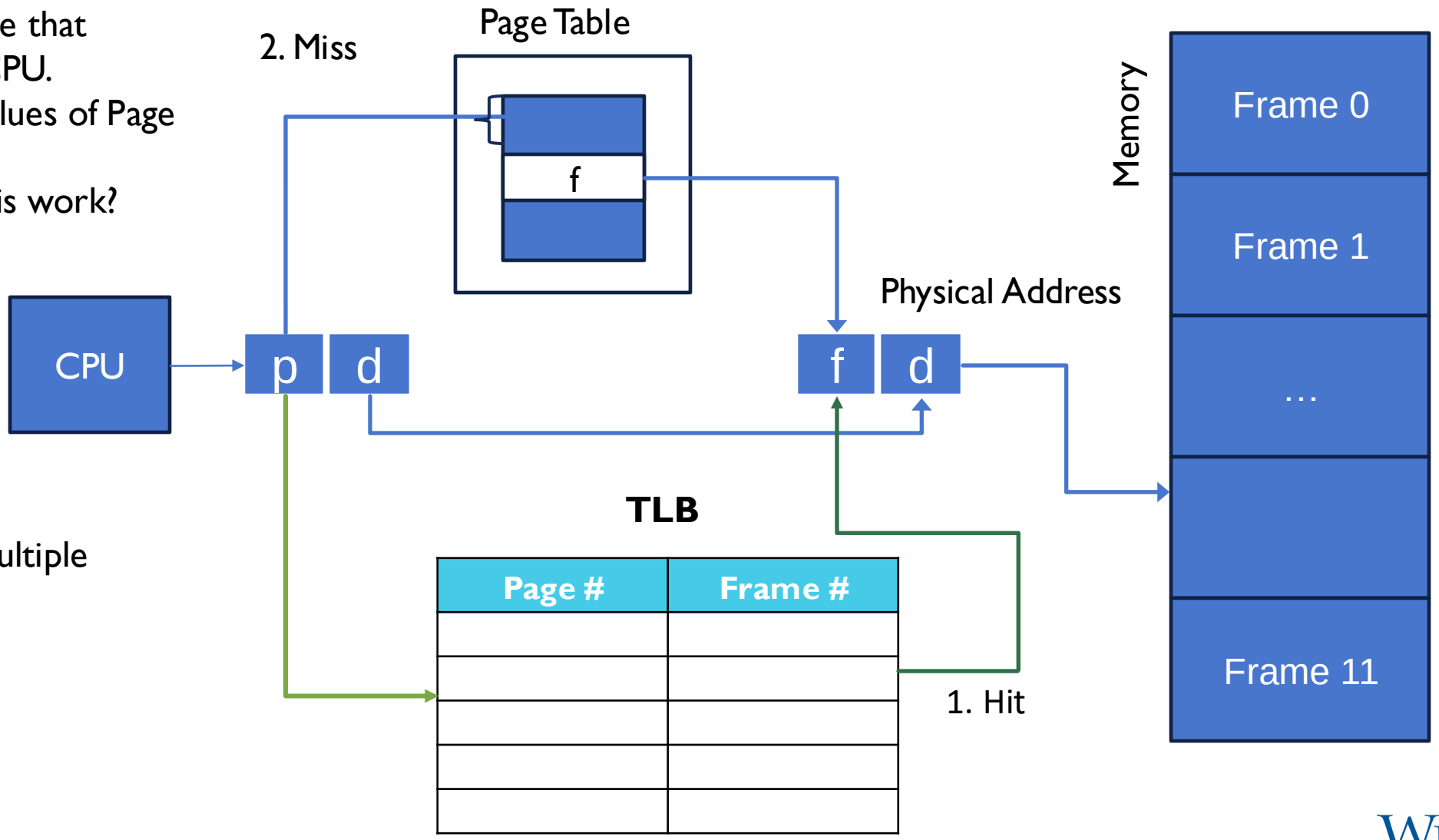
- TLB: A small table that resides on the CPU.
- Caches in the values of Page table.
- **Q:** Why does this work?



Data Locality:

TRANSLATION LOOK ASIDE BUFFER

- TLB: A small table that resides on the CPU.
- Caches in the values of Page table.
- **Q:** Why does this work?

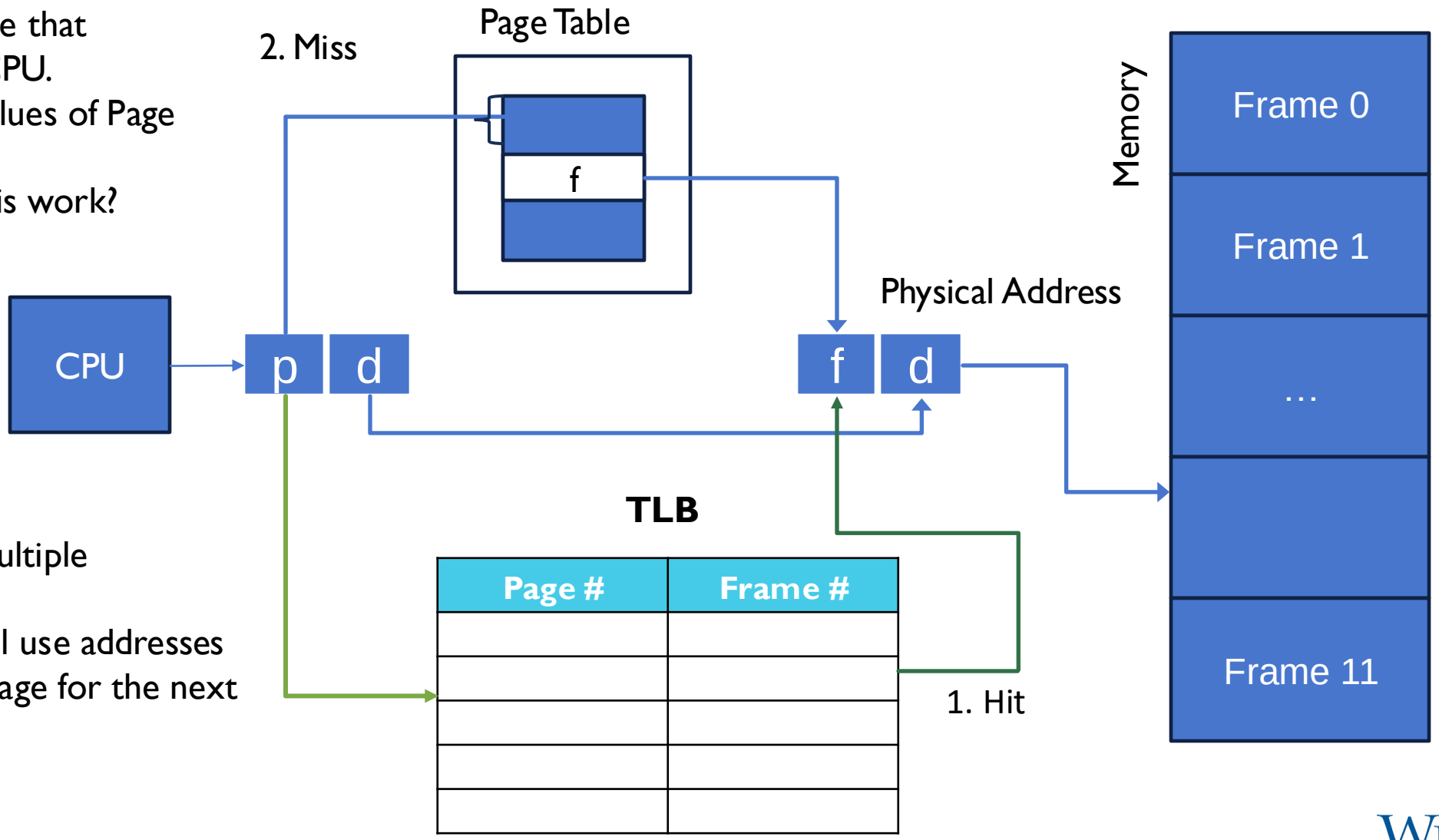


Data Locality:

- Each page has multiple addresses.

TRANSLATION LOOK ASIDE BUFFER

- TLB: A small table that resides on the CPU.
- Caches in the values of Page table.
- **Q:** Why does this work?

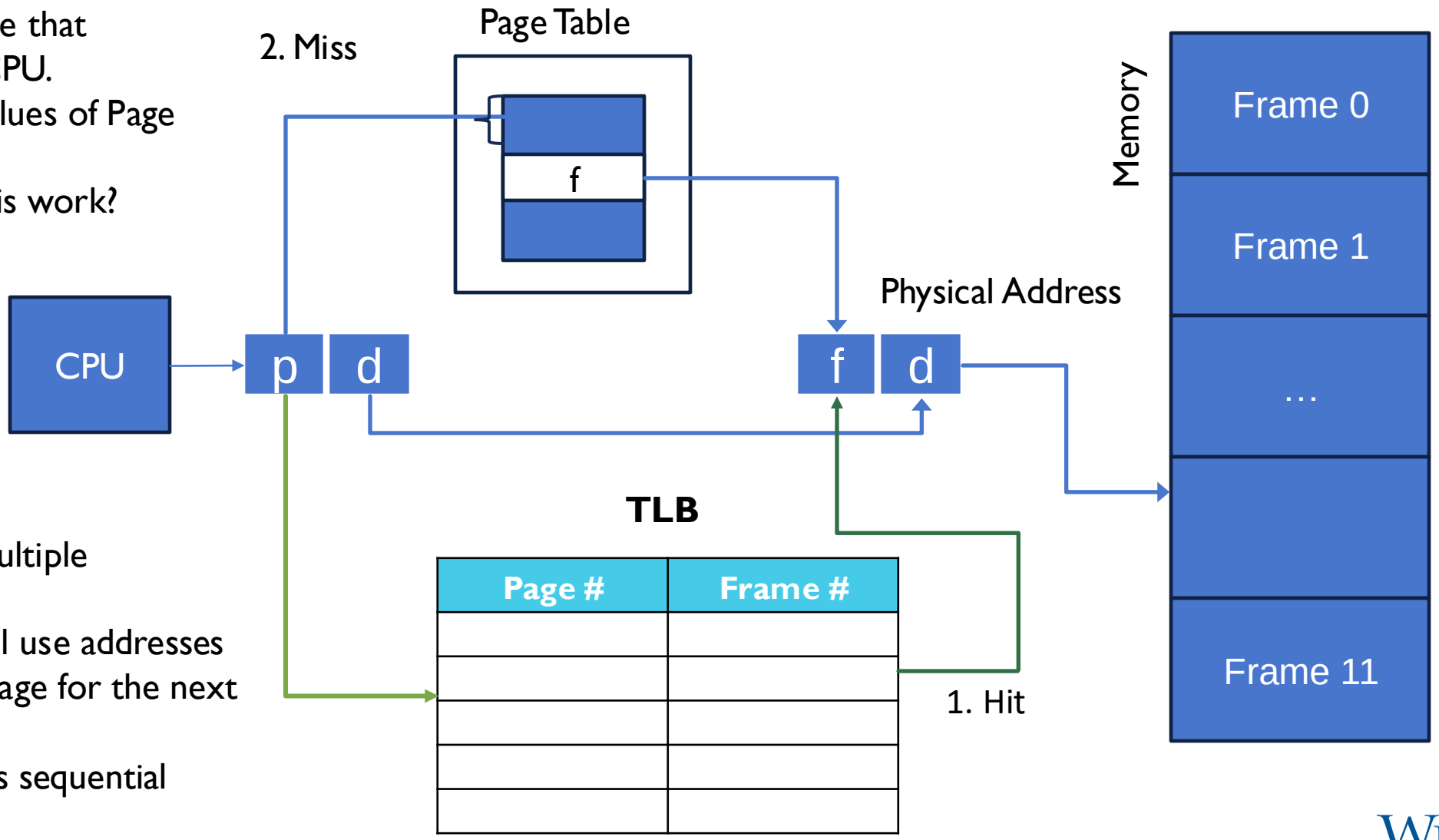


Data Locality:

- Each page has multiple addresses.
- You probably will use addresses from the same page for the next few cycles.

TRANSLATION LOOK ASIDE BUFFER

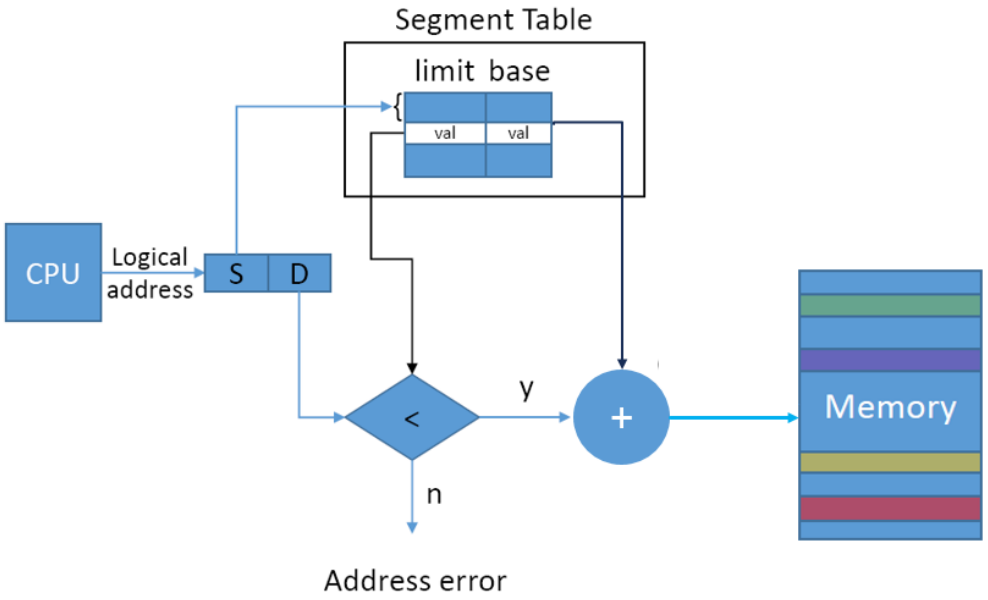
- TLB: A small table that resides on the CPU.
- Caches in the values of Page table.
- **Q:** Why does this work?

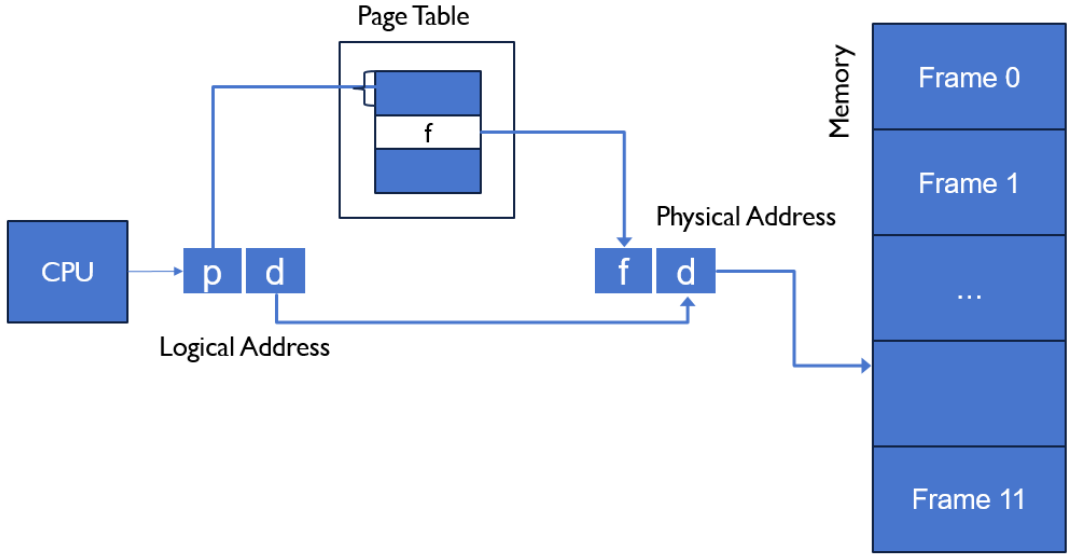


Data Locality:

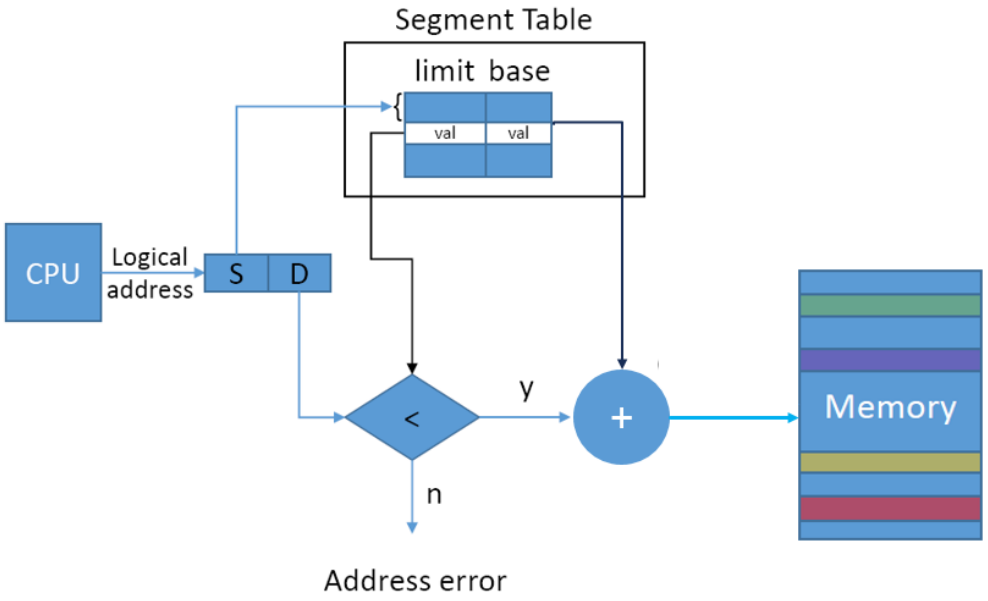
- Each page has multiple addresses.
- You probably will use addresses from the same page for the next few cycles.
- Example: Code is sequential

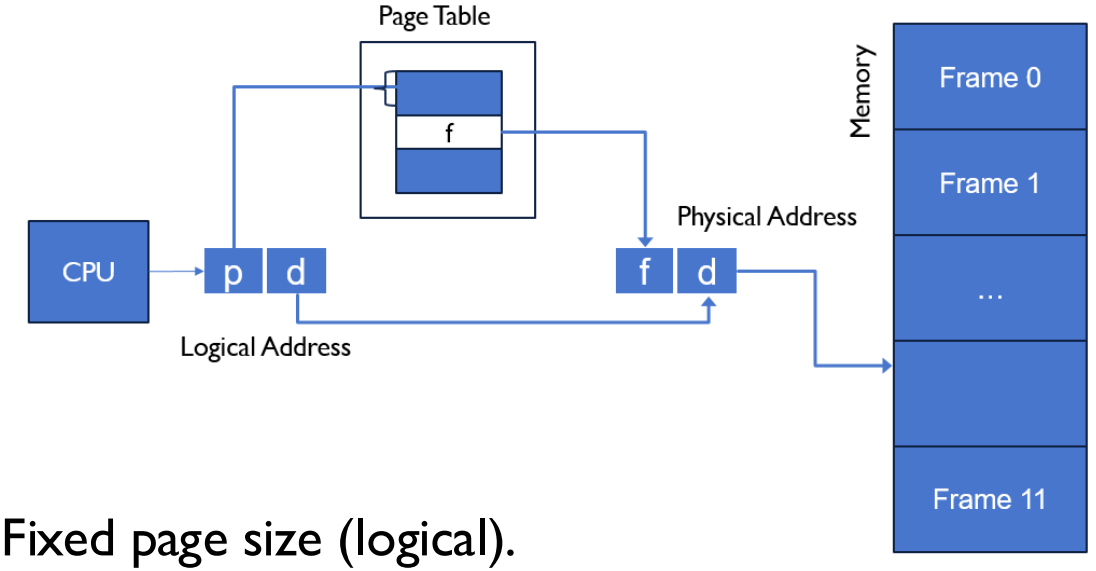
PAGING VS SEGMENTATION





PAGING VS SEGMENTATION





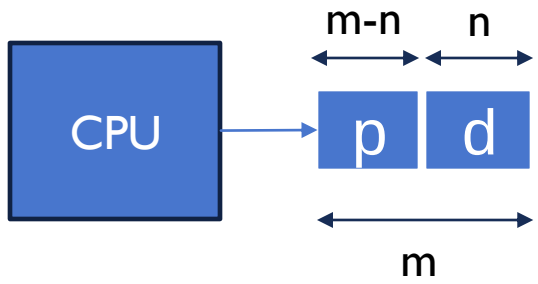
1. Fixed page size (logical).
2. Fixed frame size (physical) = page size.
3. Contiguous virtual (logical) address. Programmer doesn't see <p,d> but just an address.
4. Not all Pages need to be in memory!

WORKSHEET

- **Given a 32-bit machine with a page size of 512 bytes. What would be the page table size?**

WORKSHEET

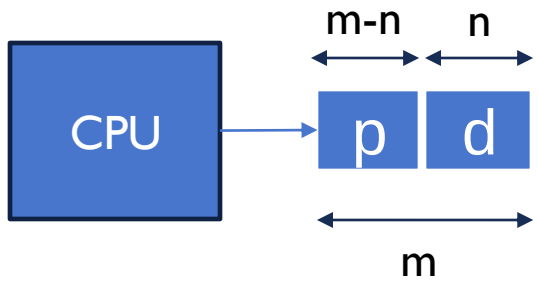
- **Given a 32-bit machine with a page size of 512 bytes. What would be the page table size?**



WORKSHEET

- **Given a 32-bit machine with a page size of 512 bytes. What would be the page table size?**

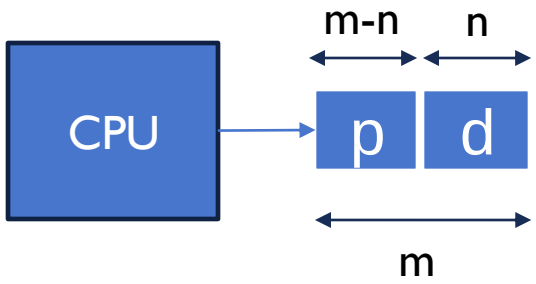
- m = 32



WORKSHEET

- **Given a 32-bit machine with a page size of 512 bytes. What would be the page table size?**

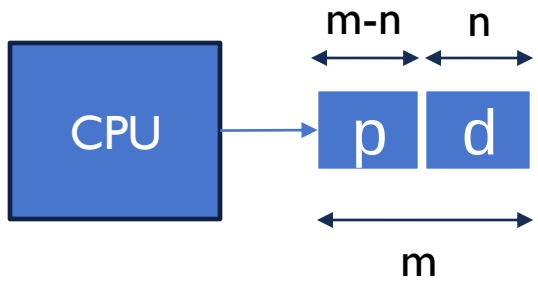
- m = 32
- Page size = 512 bytes



WORKSHEET

- **Given a 32-bit machine with a page size of 512 bytes. What would be the page table size?**

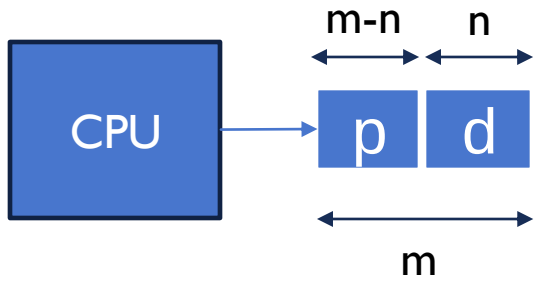
- m = 32
- Page size = 512 bytes → 2^9 → n= $\log_2(512)$ = 9.



WORKSHEET

- **Given a 32-bit machine with a page size of 512 bytes. What would be the page table size?**

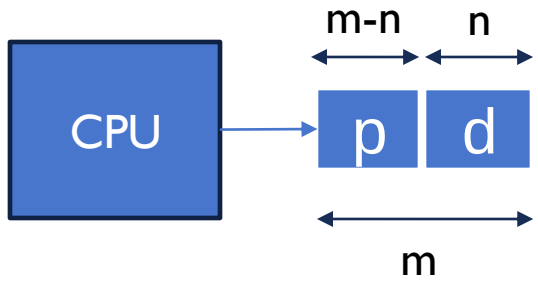
- m = 32
- Page size = 512 bytes → 2^9 → n= $\log_2(512) = 9$.
- m-n = 32-9 = 23 → 2^{23} possible pages ($2^{(m-n)}$)



WORKSHEET

- **Given a 32-bit machine with a page size of 512 bytes. What would be the page table size?**

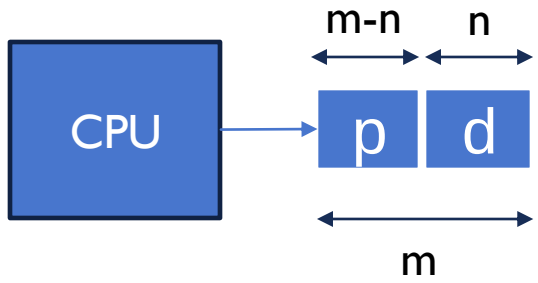
- m = 32
- Page size = 512 bytes → 2^9 → n= $\log_2(512) = 9$.
- m-n = 32-9 = 23 → 2^{23} possible pages ($2^{(m-n)}$)
- One entry for each page, each entry is 32-bit = 4 bytes.



WORKSHEET

- **Given a 32-bit machine with a page size of 512 bytes. What would be the page table size?**

- m = 32
- Page size = 512 bytes → 2^9 → n= $\log_2(512) = 9$.
- m-n = 32-9 = 23 → 2^{23} possible pages ($2^{(m-n)}$)
- One entry for each page, each entry is 32-bit = 4 bytes.
- Total size of page table: number of entries x size of entry:

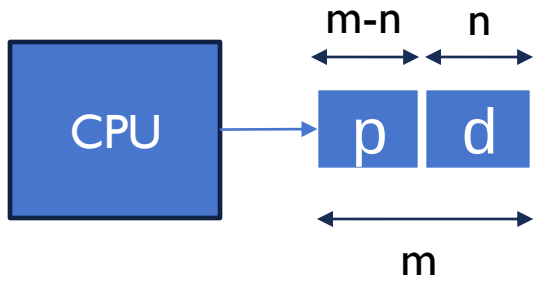


WORKSHEET

- **Given a 32-bit machine with a page size of 512 bytes. What would be the page table size?**

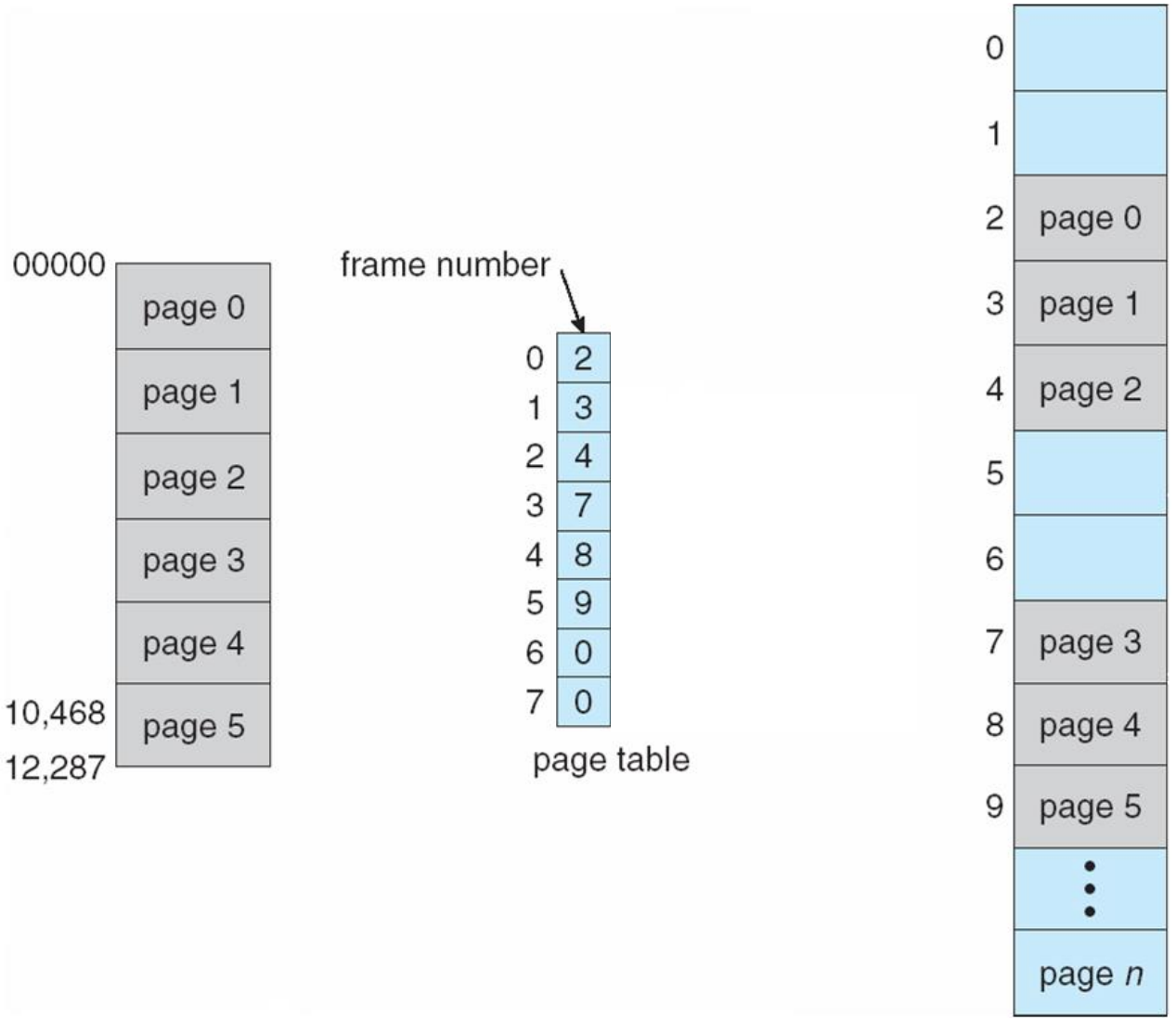
- m = 32
- Page size = 512 bytes → 2^9 → n= $\log_2(512)$ = 9.
- m-n = 32-9 = 23 → 2^{23} possible pages ($2^{(m-n)}$)
- One entry for each page, each entry is 32-bit = 4 bytes.
- Total size of page table: number of entries x size of entry:

  4 bytes x 2^{23} = 2^{25} bytes= 32 MB.



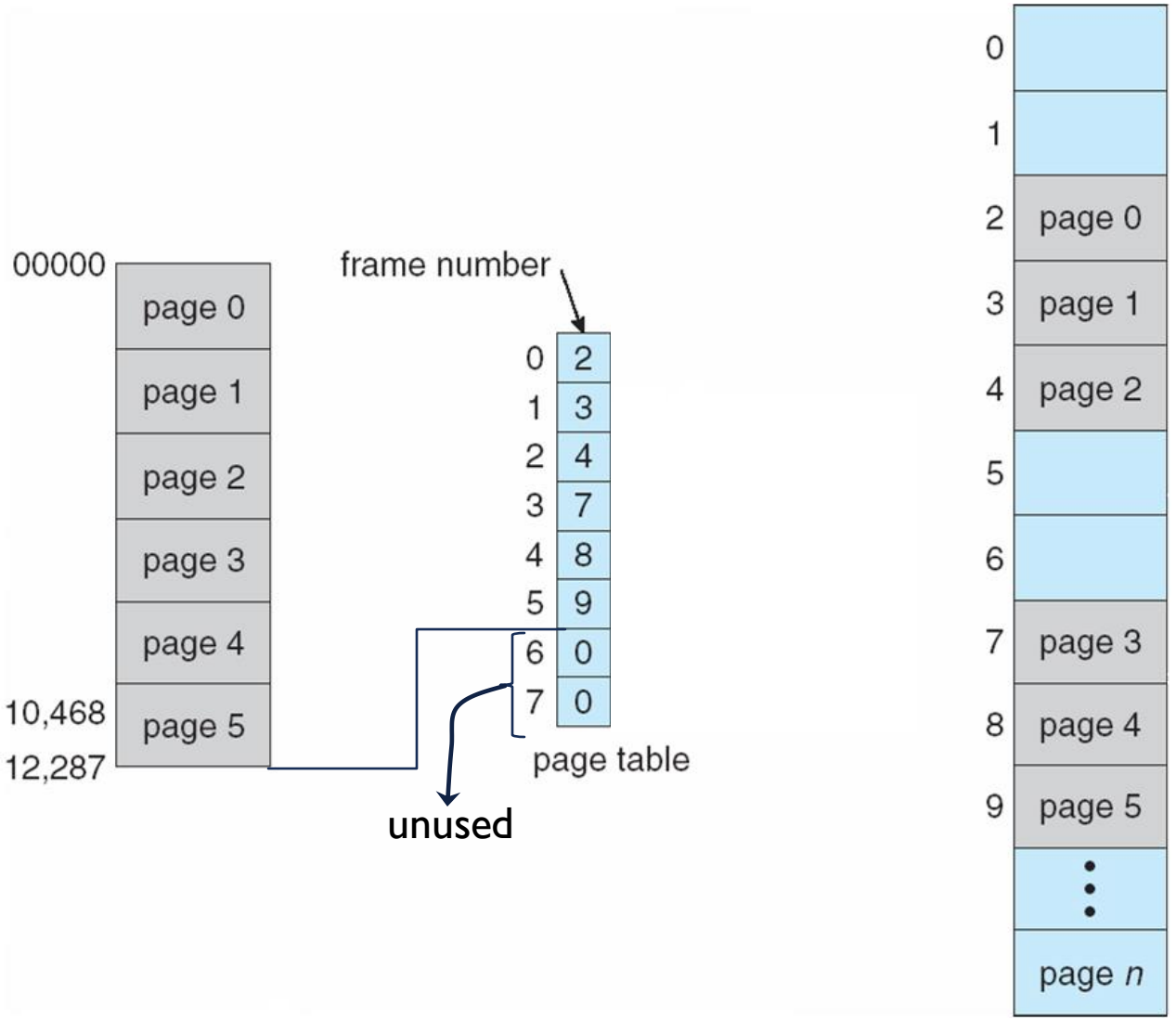
PAGE TABLE LENGTH

- Processes rarely used their entire virtual address space.
- Some page numbers in the page tables can thus be "empty".



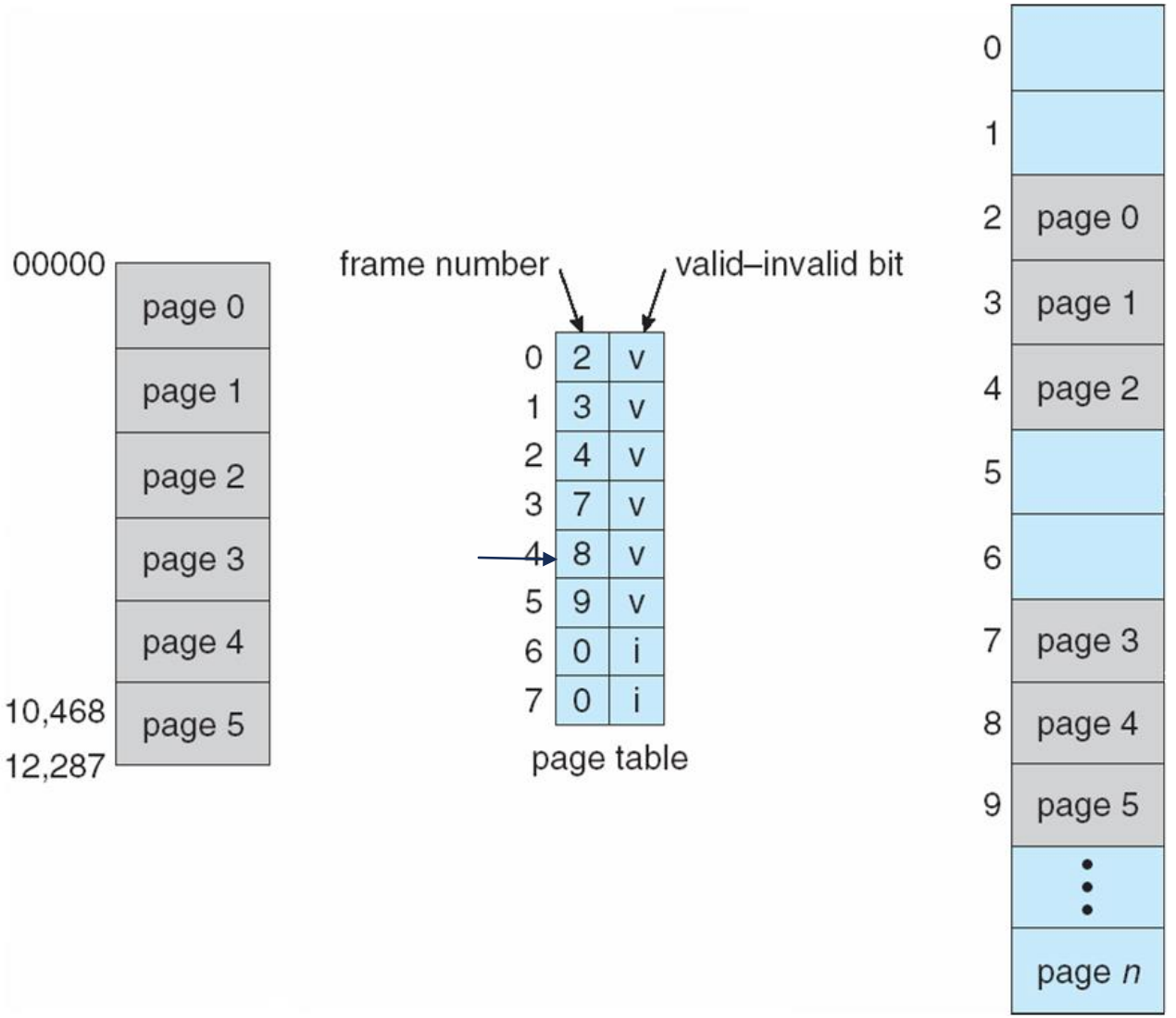
PAGE TABLE LENGTH

- Processes rarely used their entire virtual address space.
- Some page numbers in the page tables can thus be "empty".



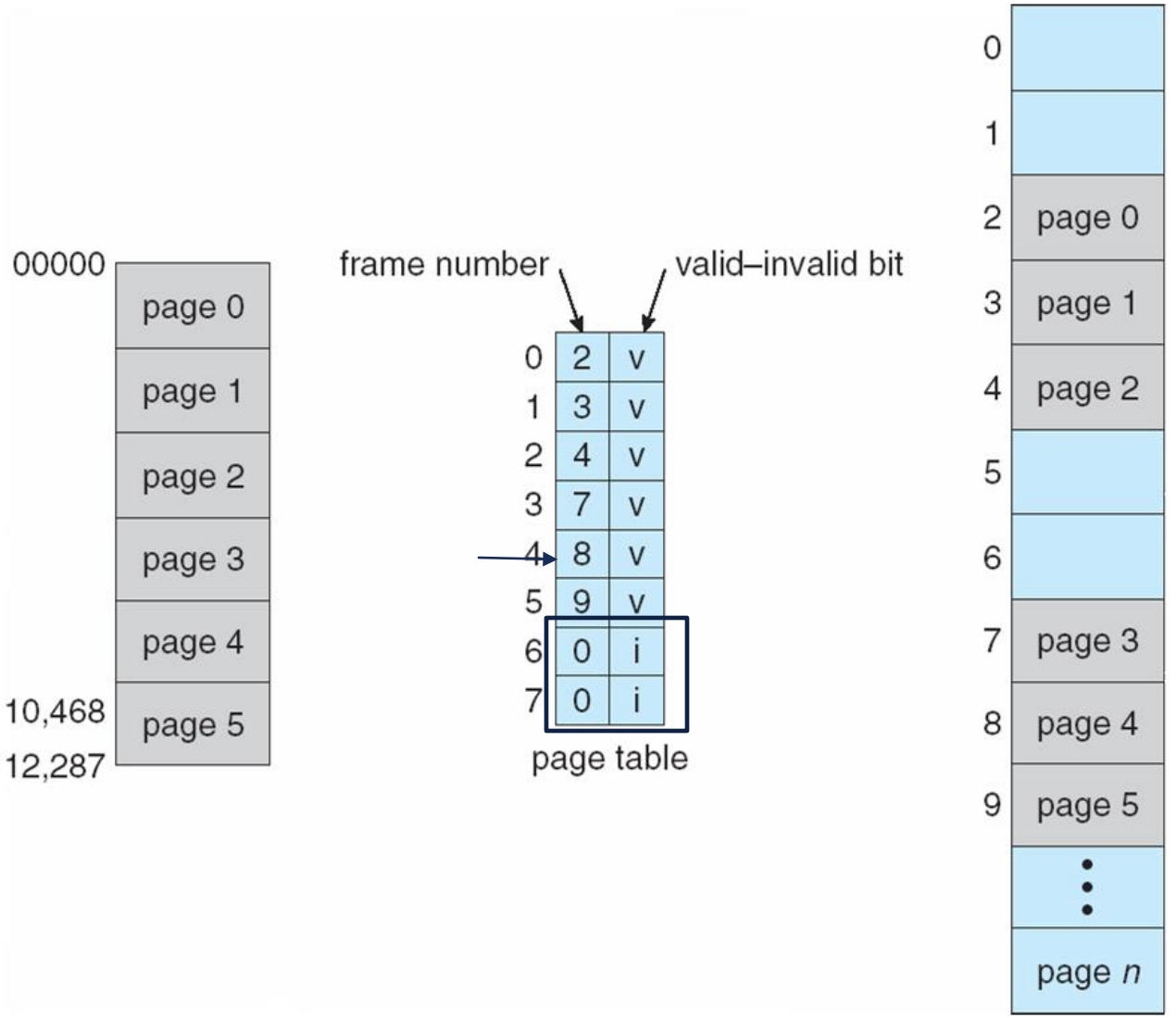
PAGE TABLE LENGTH

- Processes rarely used their entire virtual address space.
- Some page numbers in the page tables can thus be "empty".
- An valid-invalid bit can be added to indicate whether a logical address is valid or not.



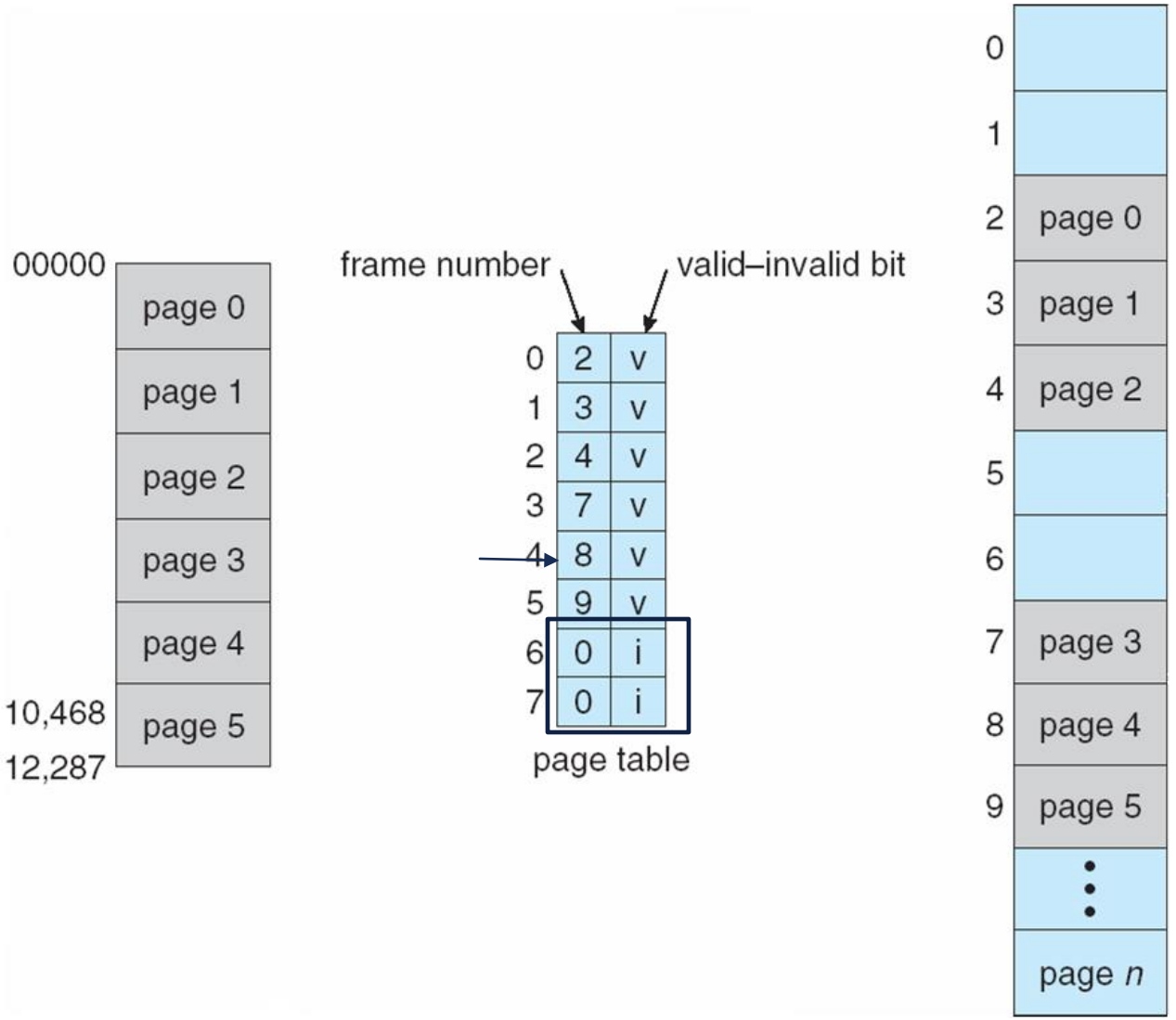
PAGE TABLE LENGTH

- Processes rarely used their entire virtual address space.
- Some page numbers in the page tables can thus be "empty".
- An valid-invalid bit can be added to indicate whether a logical address is valid or not.
- Q: Disadvantage?



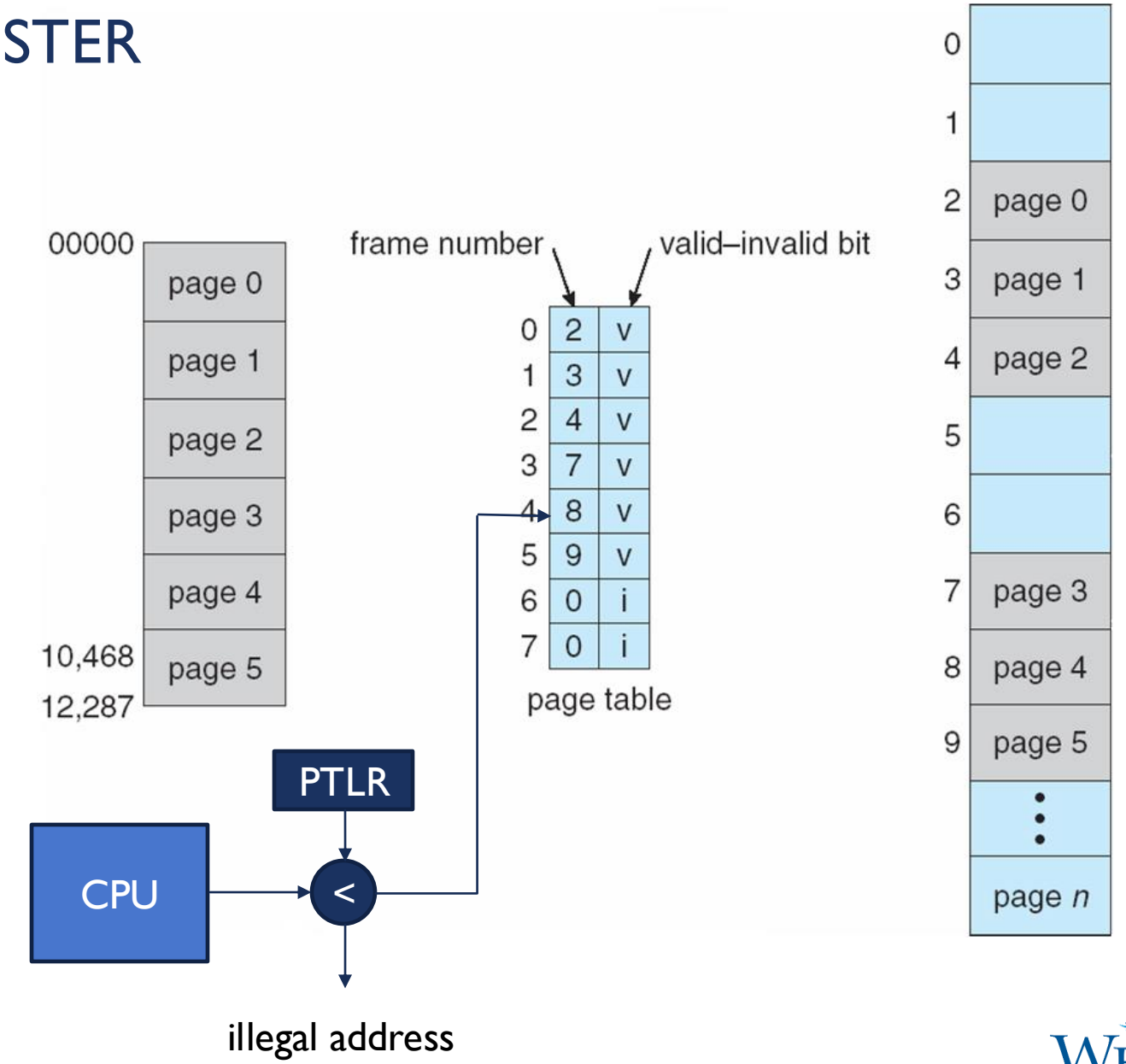
PAGE TABLE LENGTH

- Processes rarely used their entire virtual address space.
- Some page numbers in the page tables can thus be "empty".
- An valid-invalid bit can be added to indicate whether a logical address is valid or not.
- Q: Disadvantage? We still have to store a table containing all the entries .. Which can be huge and wasteful.



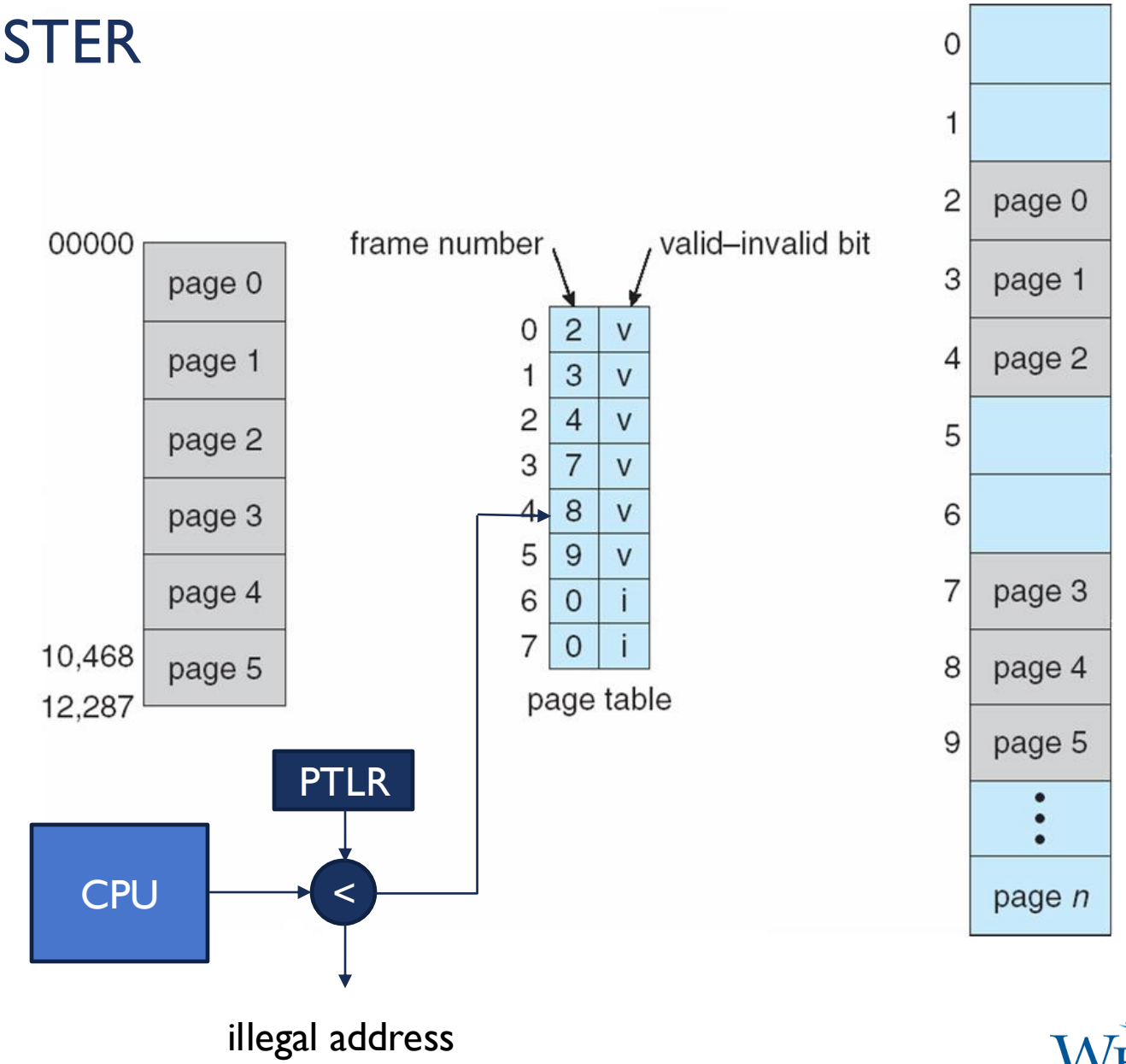
PAGE TABLE LENGTH REGISTER

- Alternative: A Page Table Length Register (PTLR) can be used to store the length of the table instead.
- Any address issued will be compared to this register value.



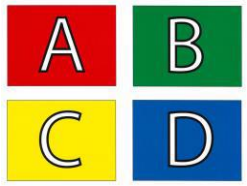
PAGE TABLE LENGTH REGISTER

- Alternative: A Page Table Length Register (PTLR) can be used to store the length of the table instead.
- Any address issued will be compared to this register value.
- Q: What should be the value of PTLR in this example?

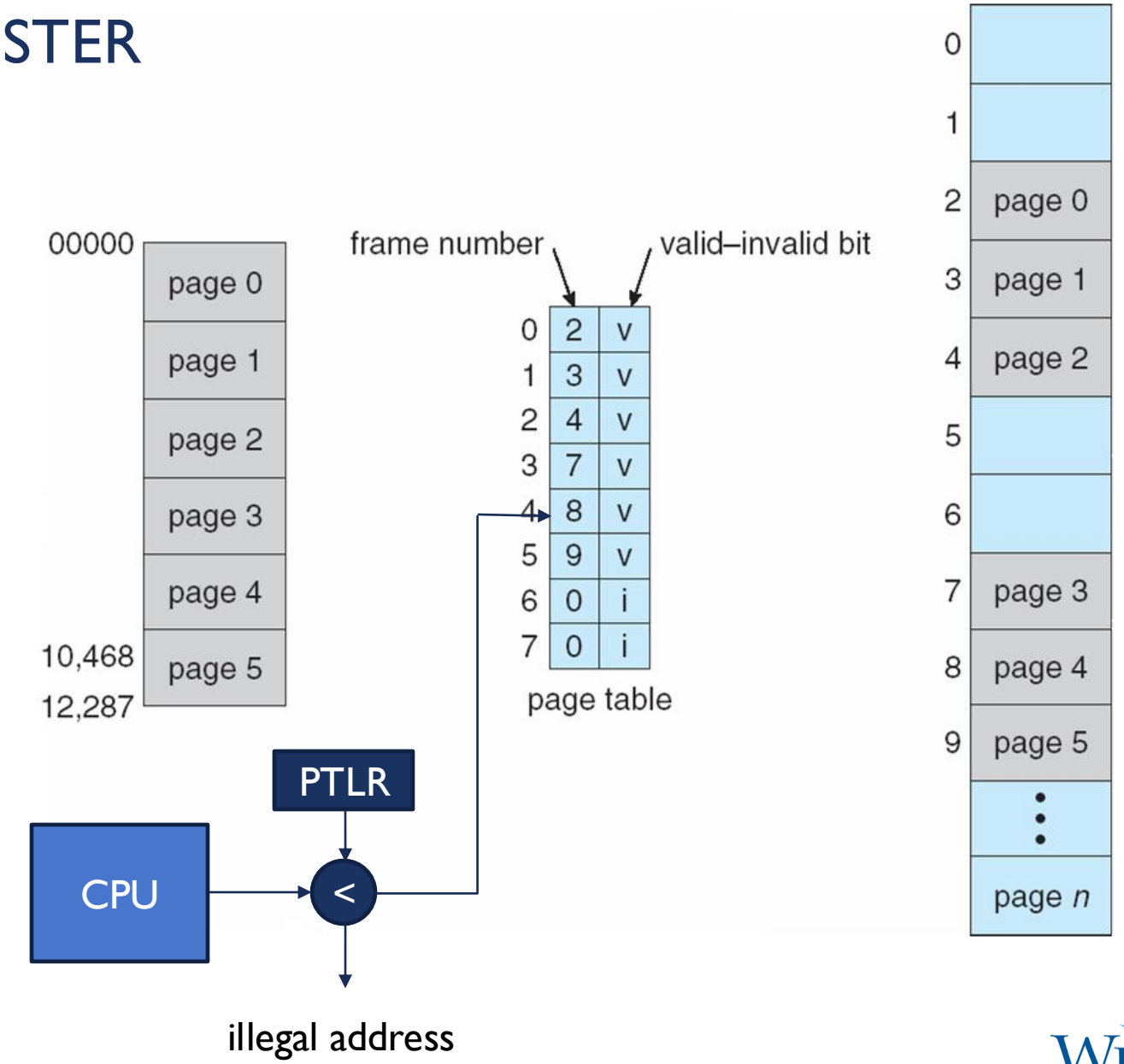


PAGE TABLE LENGTH REGISTER

- Alternative: A Page Table Length Register (PTLR) can be used to store the length of the table instead.
- Any address issued will be compared to this register value.
- Q: What should be the value of PTLR in this example?

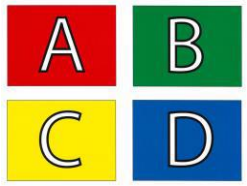


A: 5
B: 9
C: 7
D: 12,287

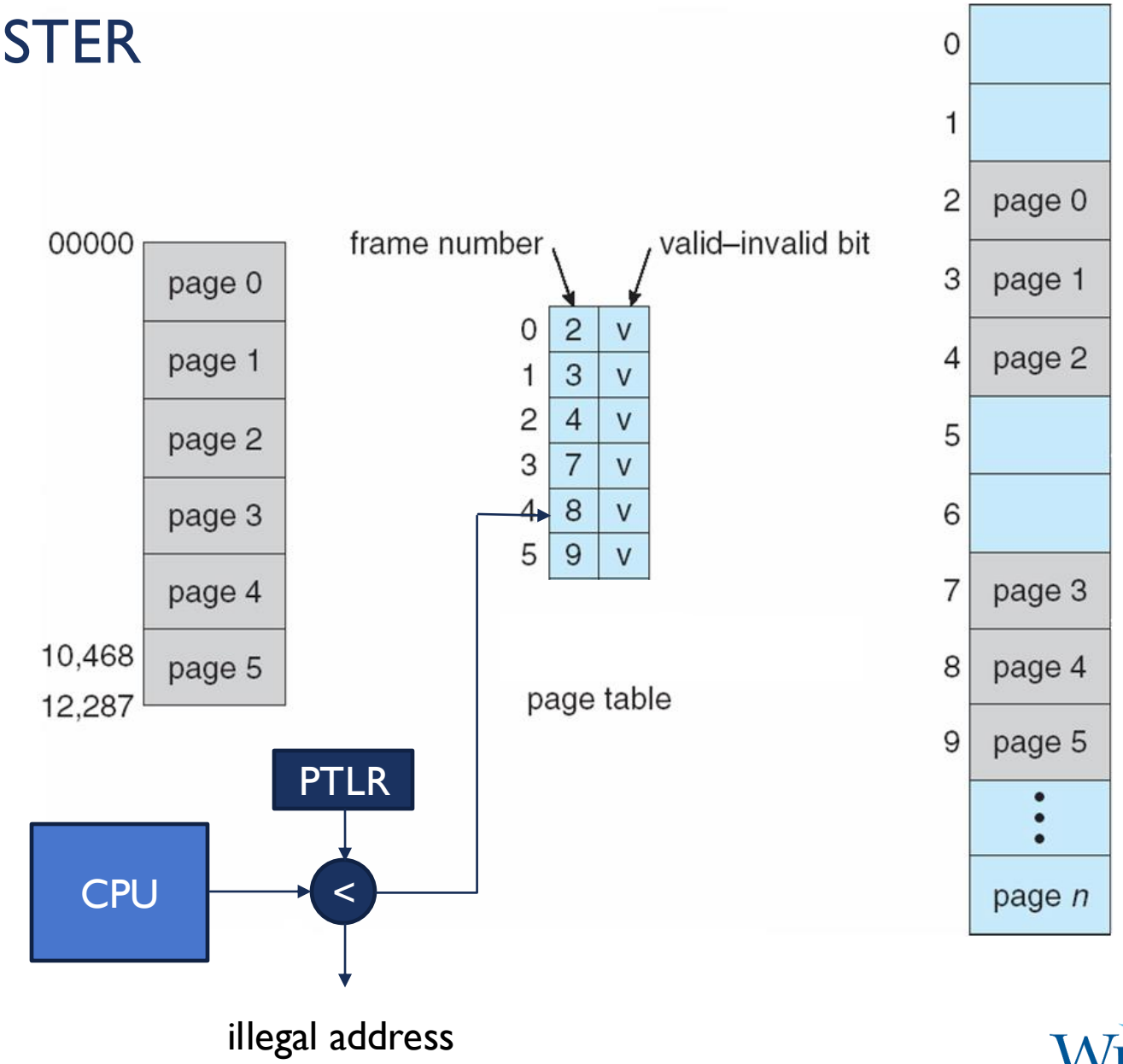


PAGE TABLE LENGTH REGISTER

- Alternative: A Page Table Length Register (PTLR) can be used to store the length of the table instead.
- Any address issued will be compared to this register value.
- Q: What should be the value of PTLR in this example?

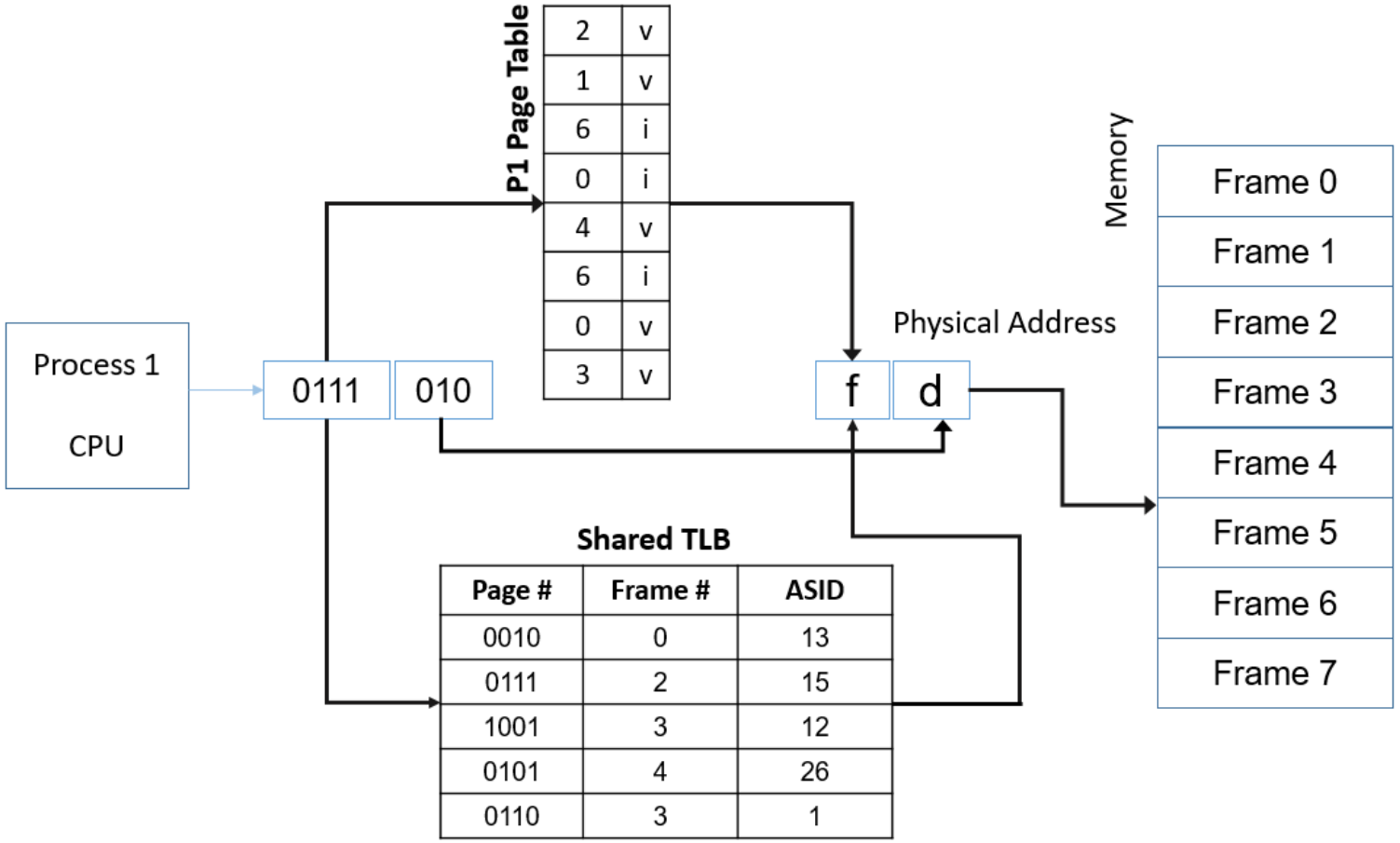


A: 5
B: 9
C: 7
D: 12,287



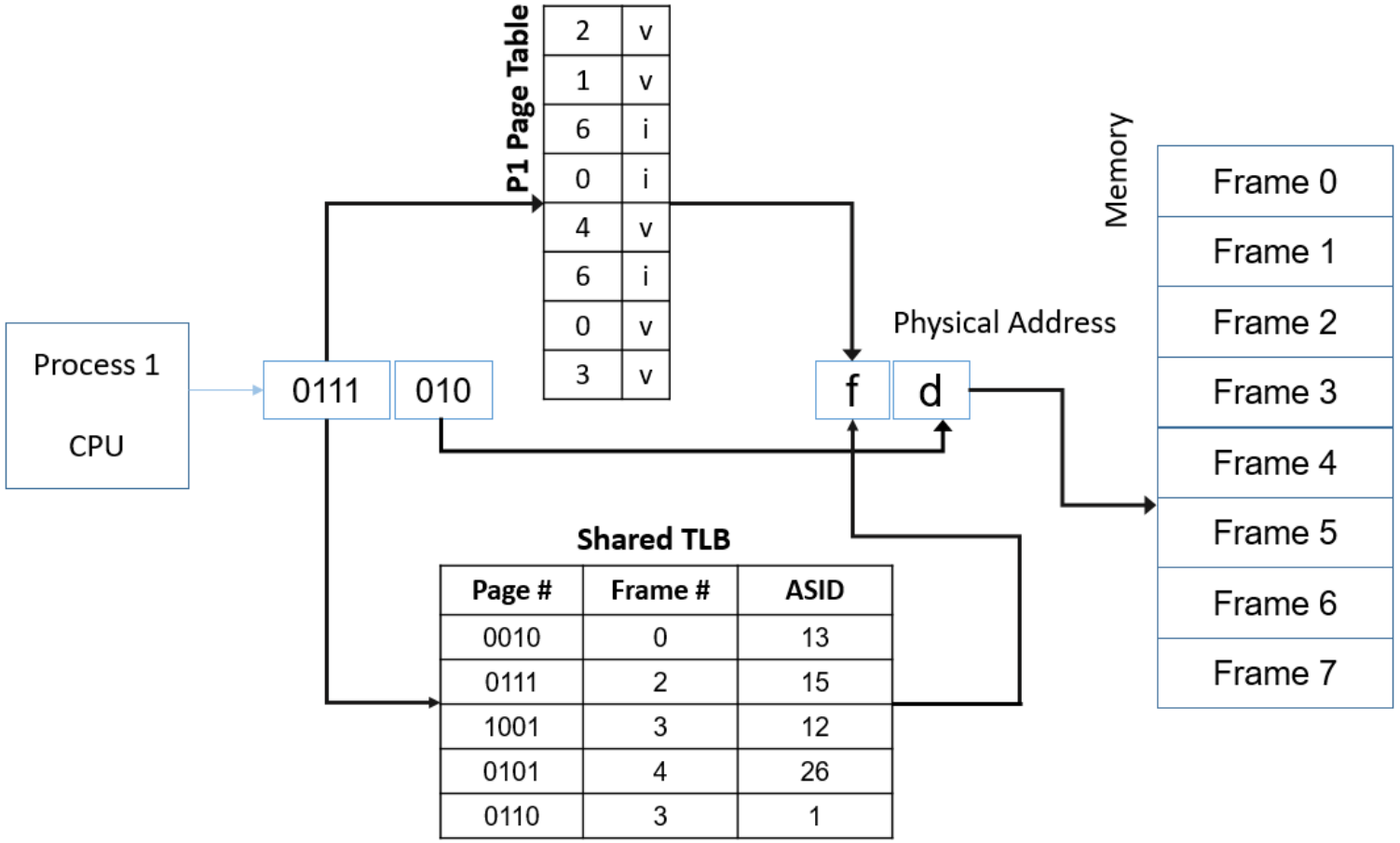
WORKSHEET Q2

- Will it cause a hit or miss at the TLB? Explain.
- Will this attempt cause a page fault? Explain.
- What is the frame number (physical frame address) retrieved? Explain.
- What byte address does the process attempt to access? Explain.



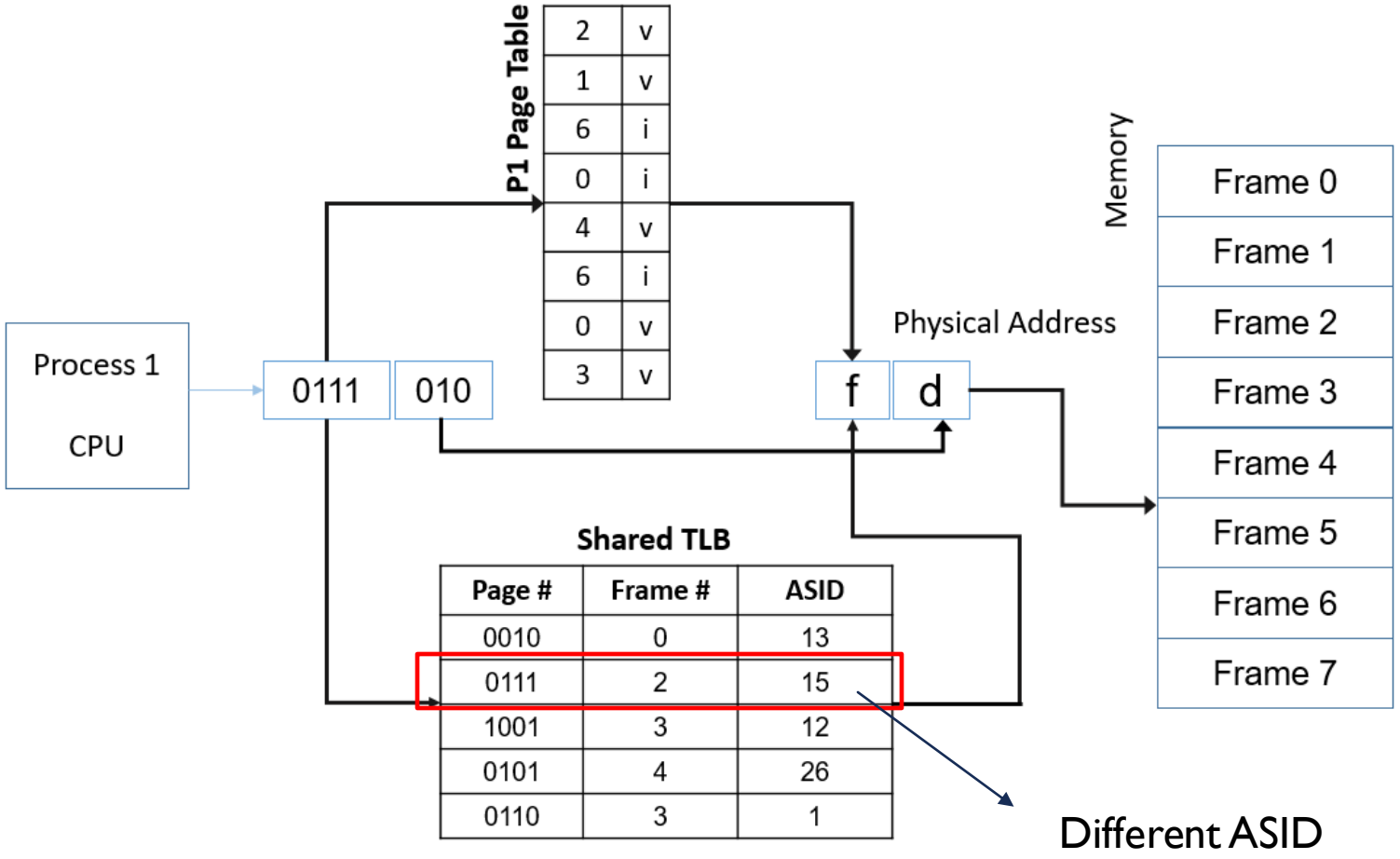
WORKSHEET Q2

- TLB Hit or Miss?



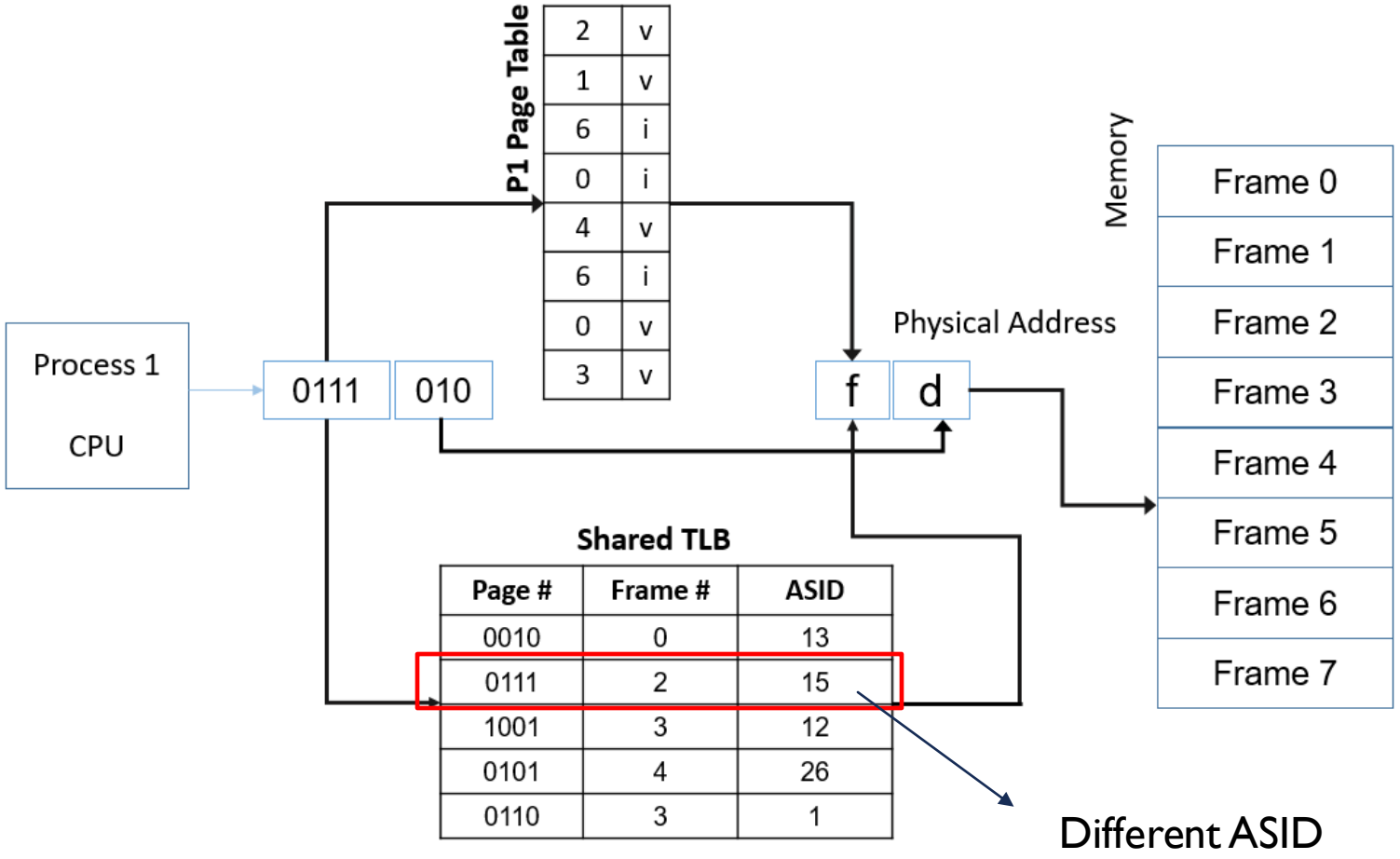
WORKSHEET Q2

- TLB Hit or Miss?



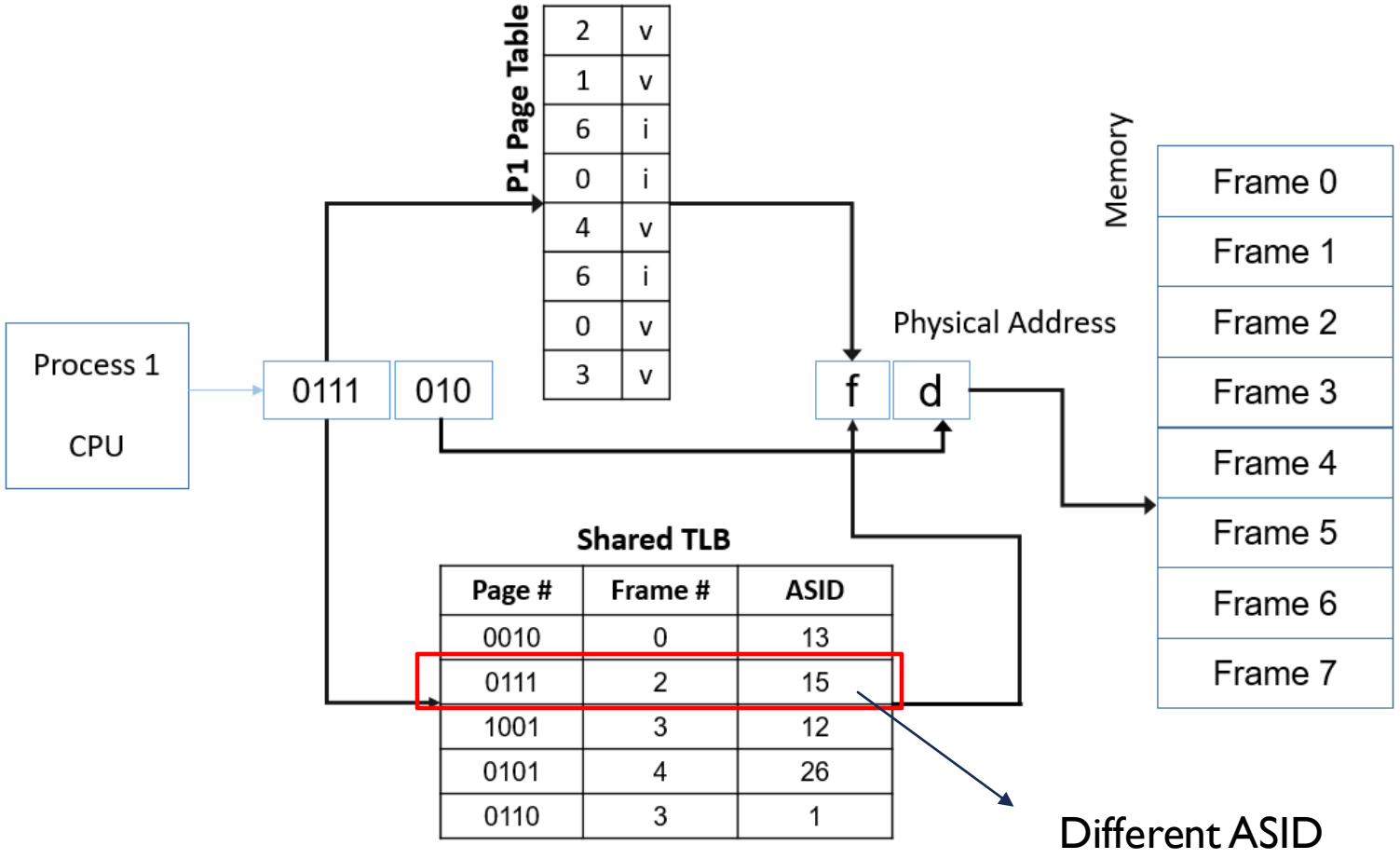
WORKSHEET Q2

- TLB Hit or Miss? Miss



WORKSHEET Q2

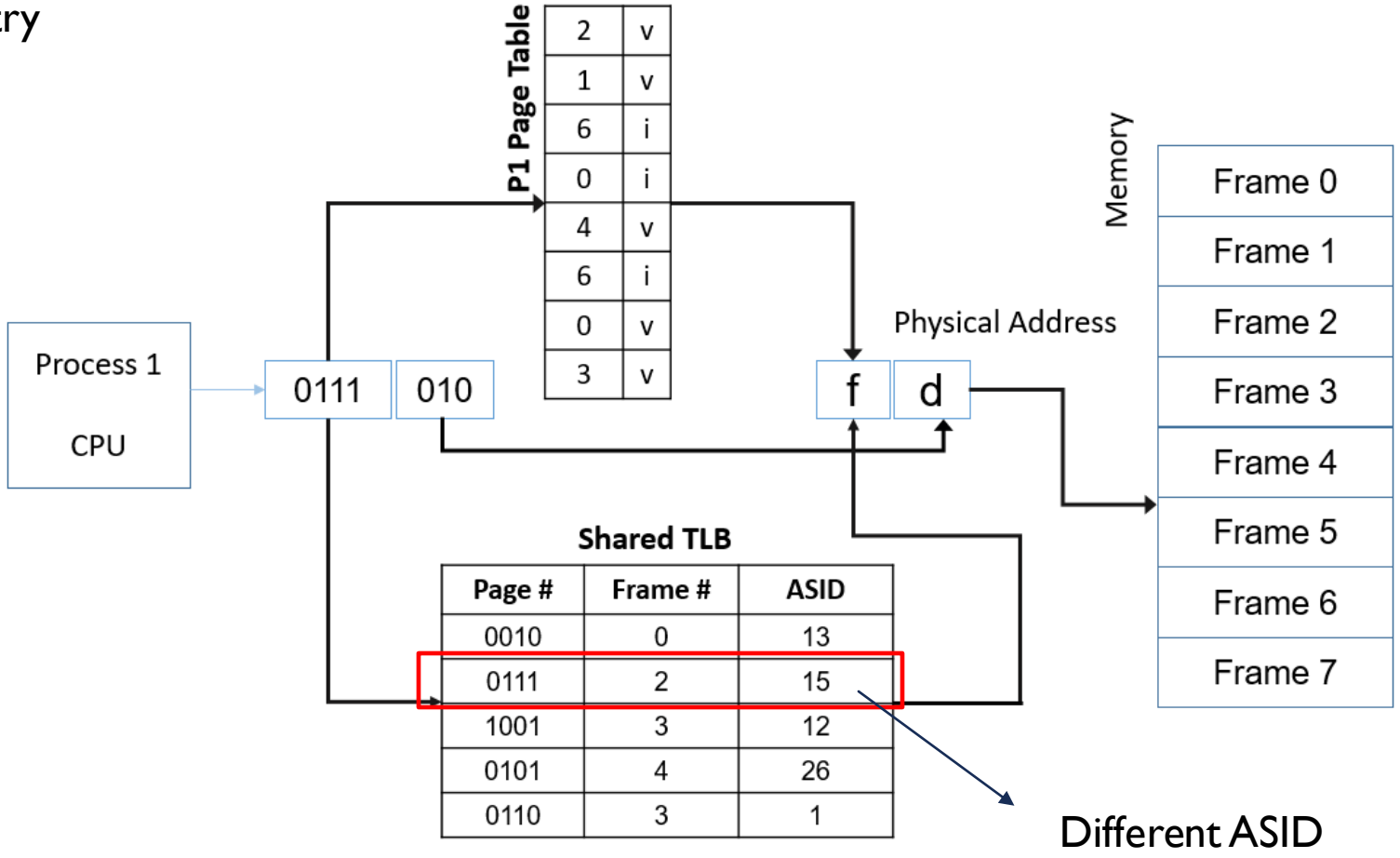
- Page Fault?



WORKSHEET Q2

0111 = 7 in decimal
Check 8th entry

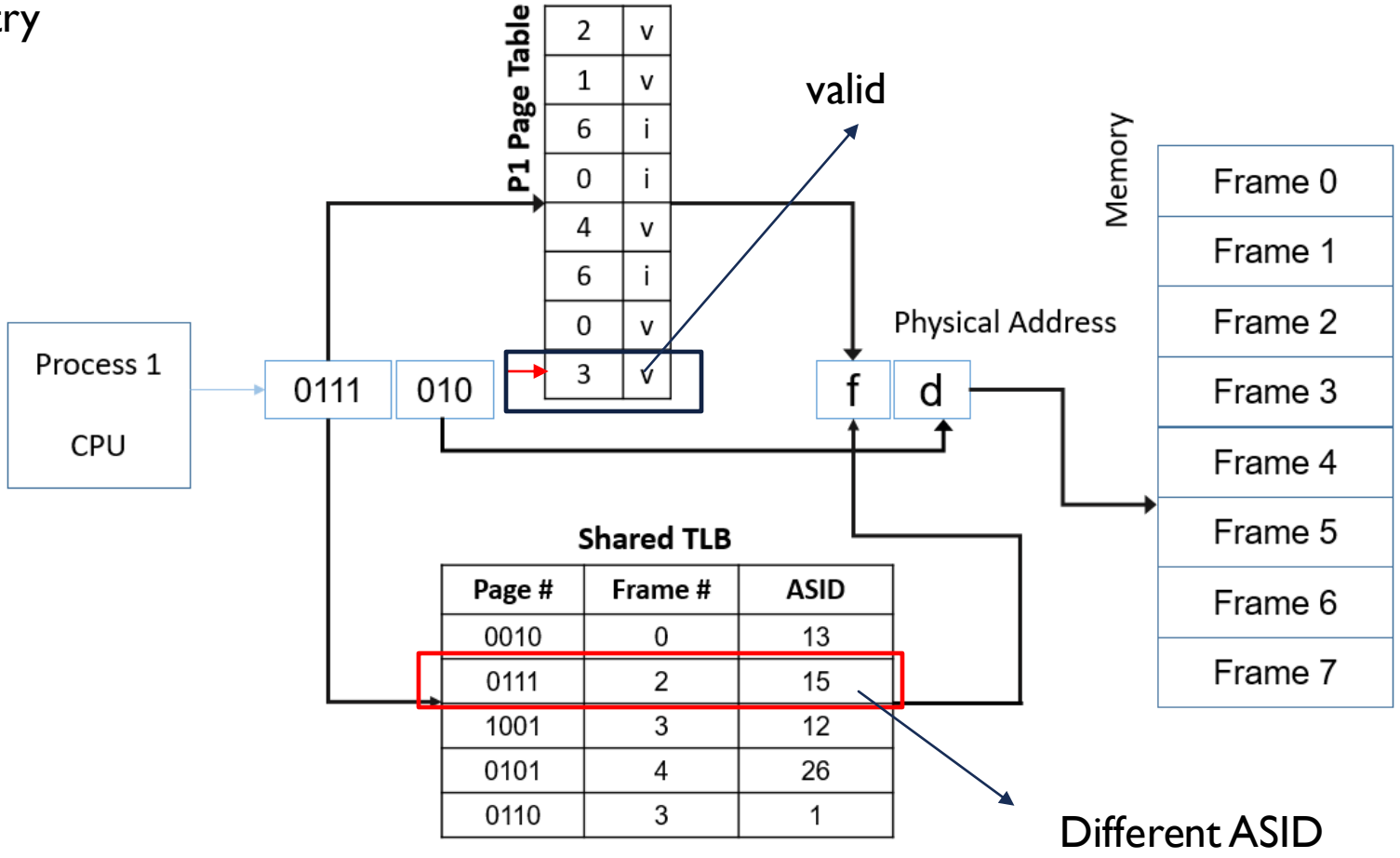
- Page Fault?



WORKSHEET Q2

0111 = 7 in decimal
Check 8th entry

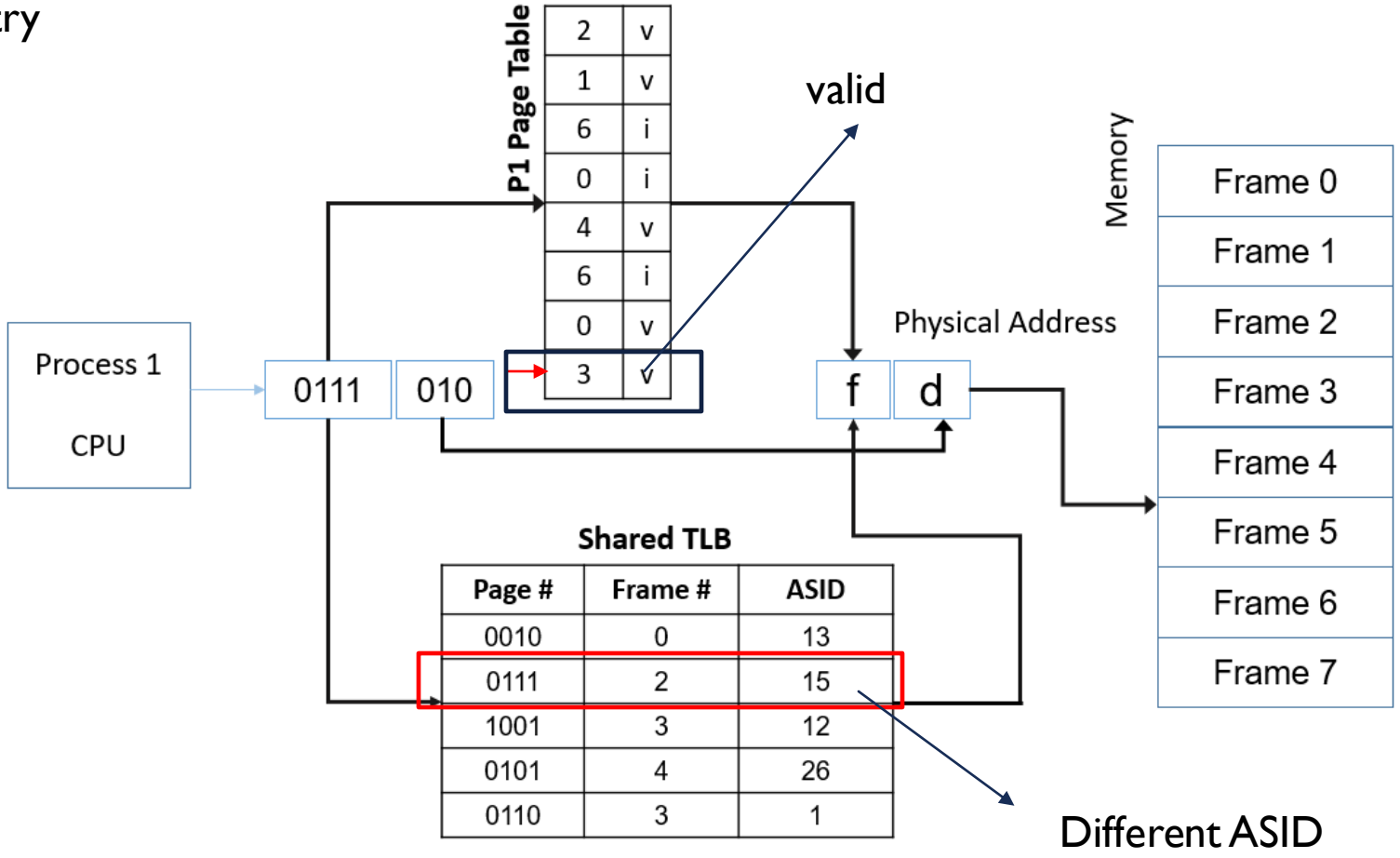
- Page Fault?



WORKSHEET Q2

0111 = 7 in decimal
Check 8th entry

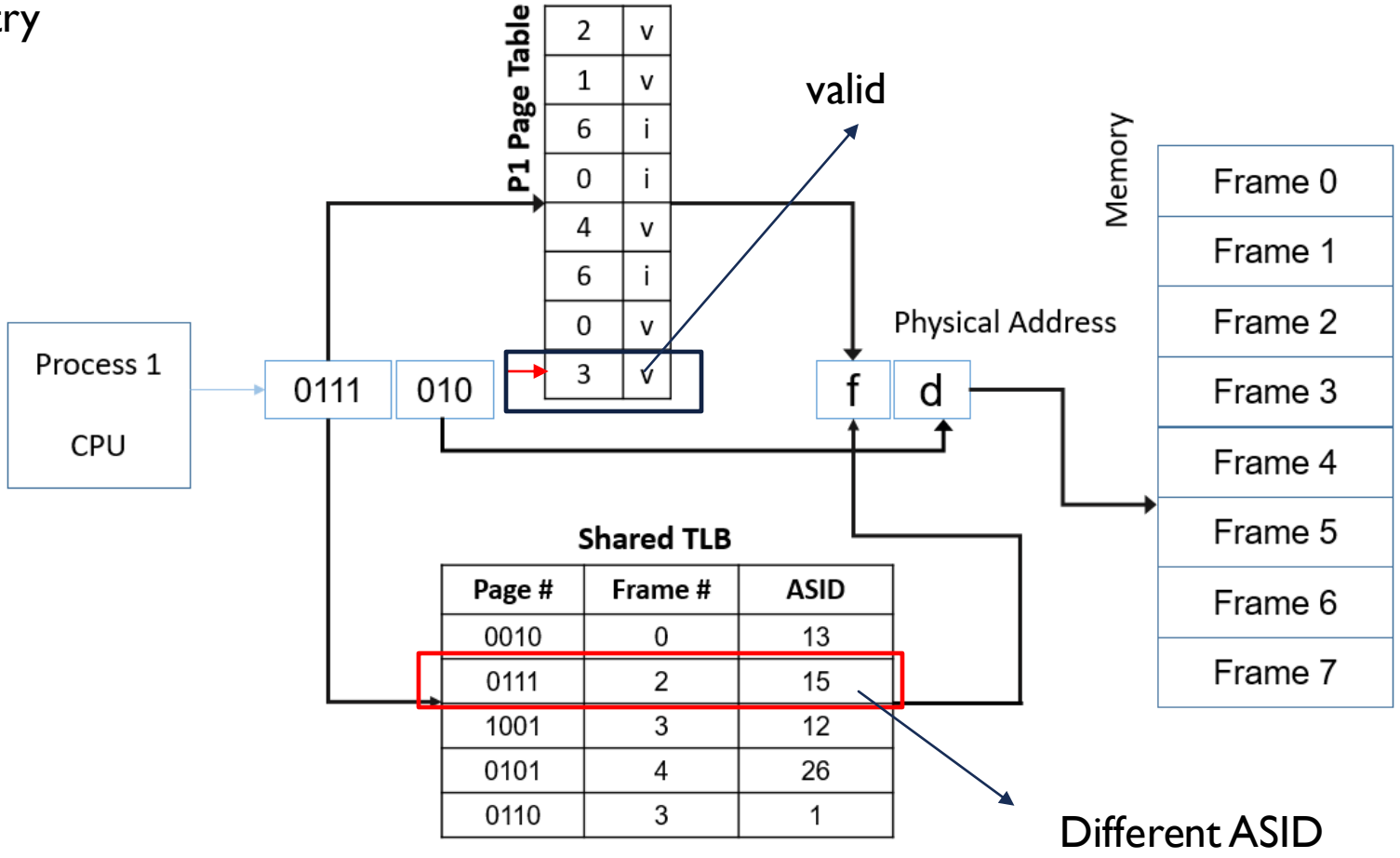
- Page Fault? No page fault!



WORKSHEET Q2

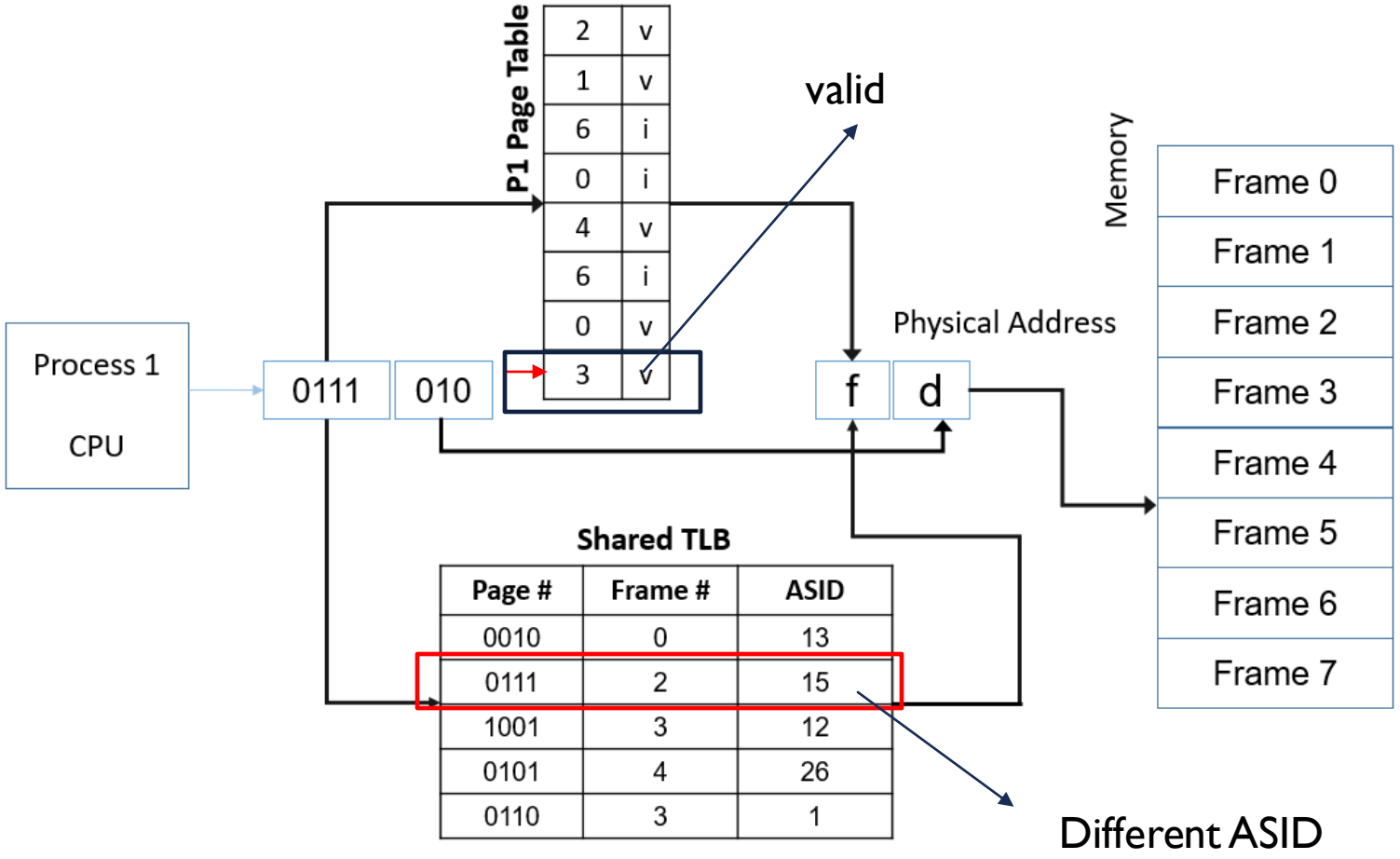
0111 = 7 in decimal
Check 8th entry

- Page Fault? No page fault!



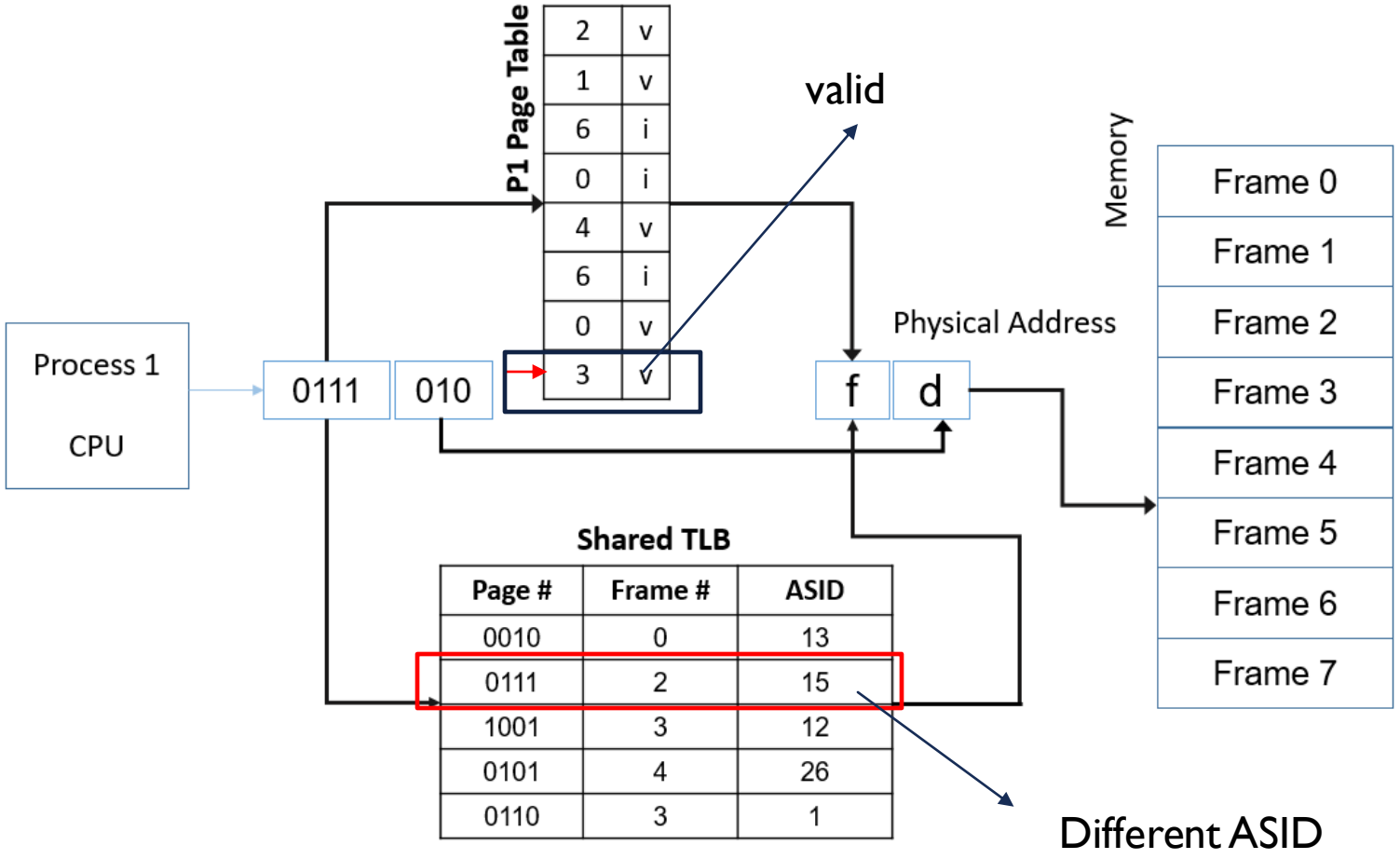
WORKSHEET Q2

- Frame retrieved?



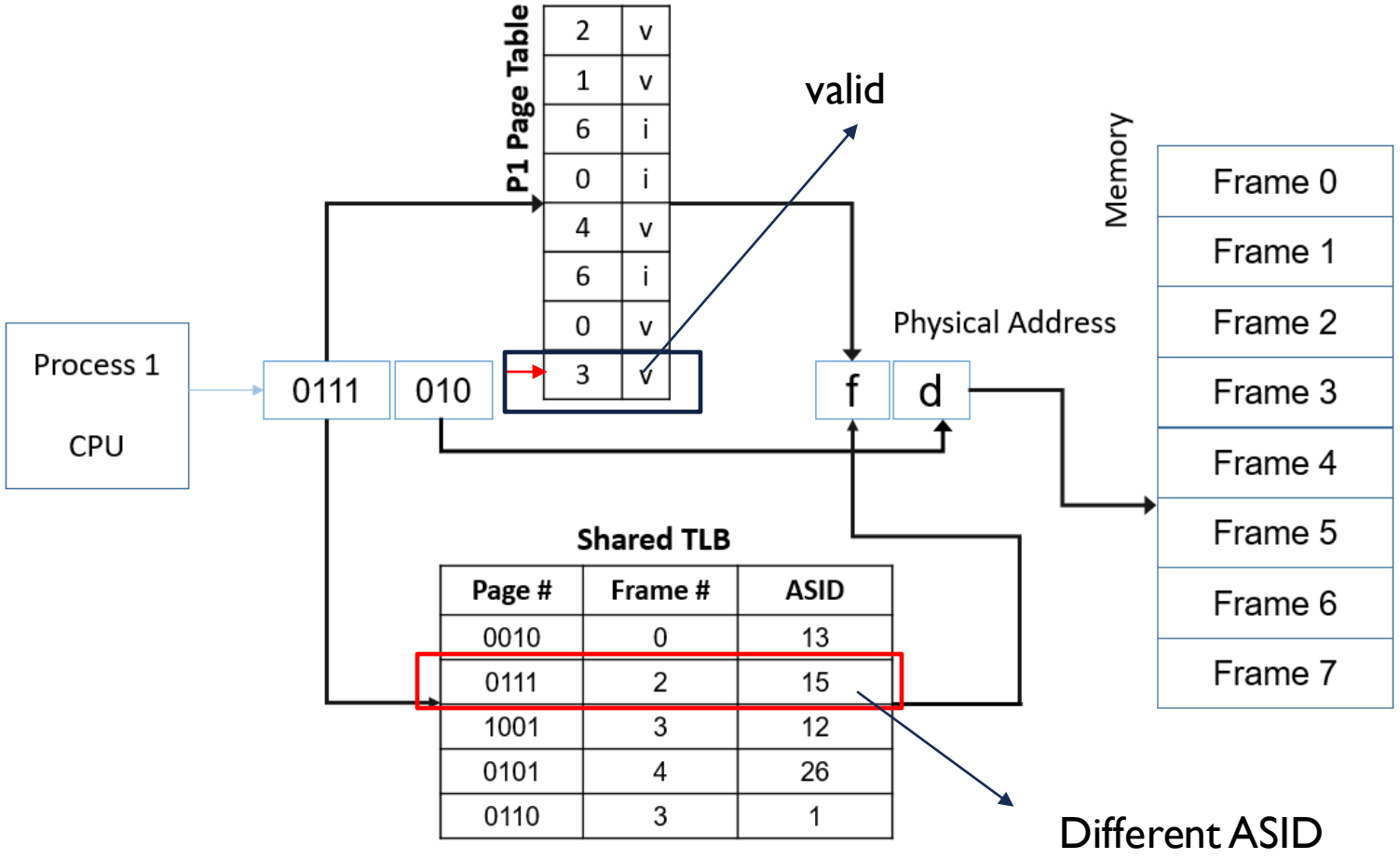
WORKSHEET Q2

- Frame retrieved? Frame #3



WORKSHEET Q2

- Byte offset in the physical address?



WORKSHEET Q2

- Byte offset in the physical address?
- Same as logical: 010