Virtual Memory
Addressing Mechanism

Computer System Architecture (CS5202)
IIT Tirupati

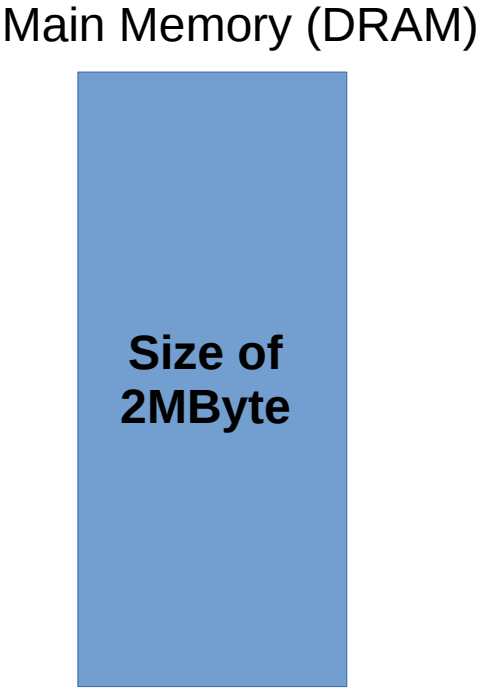
Jaynarayan T Tudu
jtt@iittp.ac.in

17th March, 2019

The problems

- How to execute a program of size larger than the available main memory?

```
int main(){

 int var1 = 0 ;
 float var2[1024][1024][8] = 5;
 float var3[1024][1024][8] = 7;

 if (var1 < 0) {
    function_call(var2, var3) ;
 }
else {

   function_dont_call(void);
}

return 0+0;
}
```



Requires minimum of 2MByte space

The problems

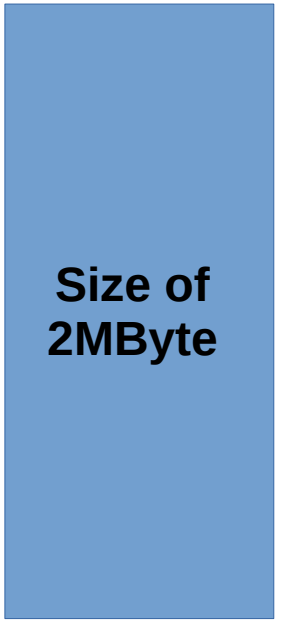
- How to execute multiple programs (or processes) in a given limited size memory?

```
int main(){
 int var1 = 0 ;
 float var2[1024][1024][8] = 5;
 float var3[1024][1024][8] = 7;

 //do something
 return 0;
}
```

```
int main(){
 int var4 = 0 ;
 float var5[1024][1024][8] = 5;
 float var6[1024][1024][8] = 7;

 //do something else
 return 0;
}
```

Two programs to be executed

Main Memory (DRAM)



The problems

- How to provide protection to each process?

Program 1

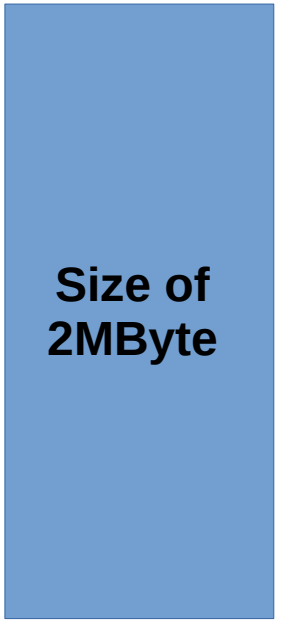
```
int main(){
 int var1 = 0 ;
 float var2[1024][1024][8] = 5;
 float var3[1024][1024][8] = 7;

 //do something
 return 0;
}
```

Program 2

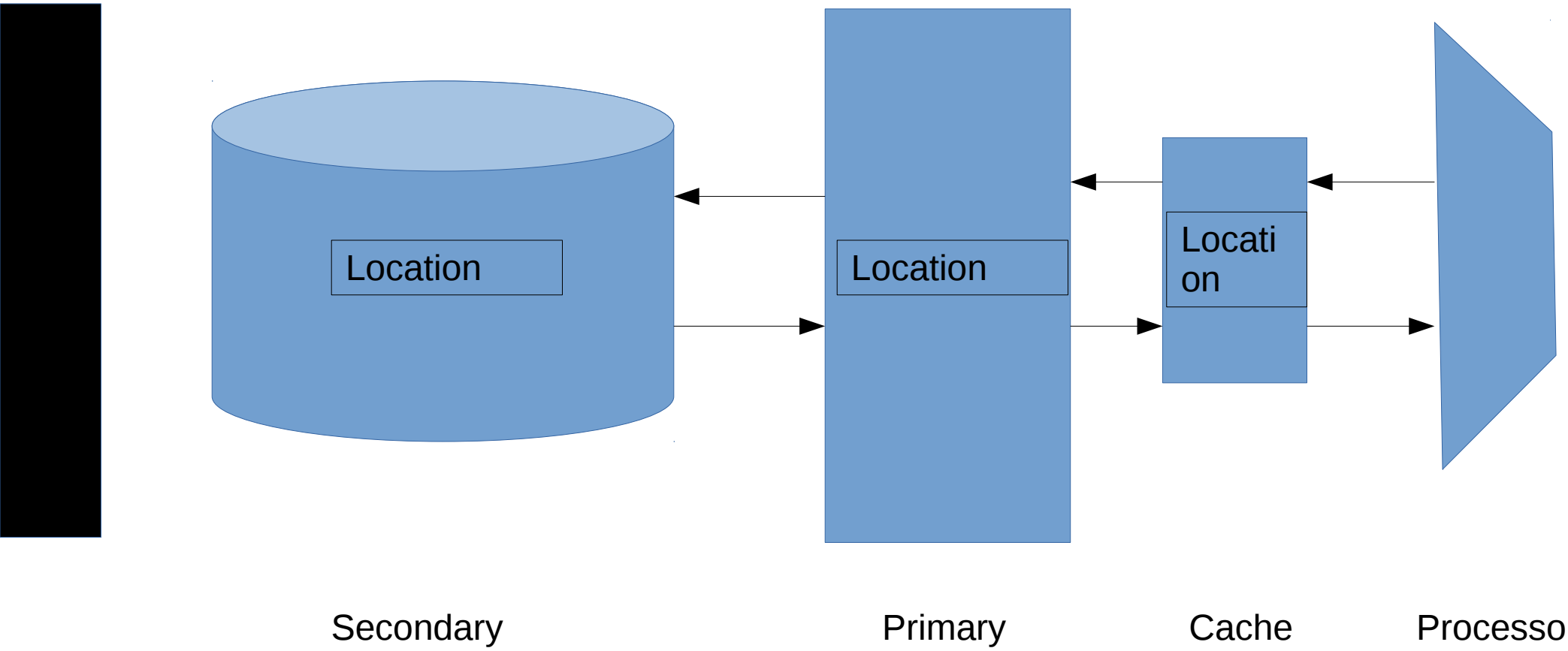
```
int main(){
 int var4 = 0 ;
 float var5[1024][1024][8] = 5;
 float var6[1024][1024][8] = 7;

 //do something else
 return 0;
}
```

Two programs to be executed

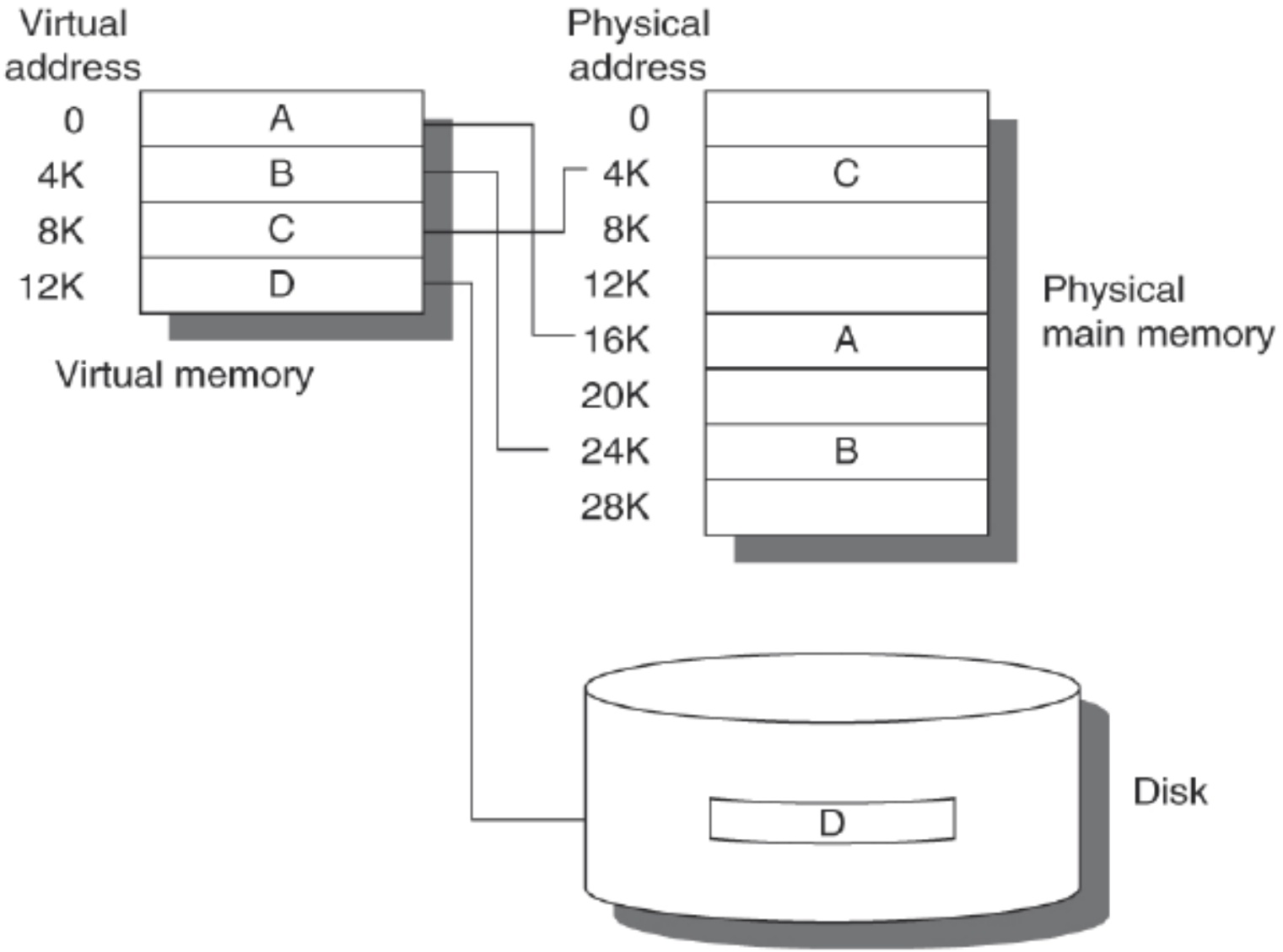
Main Memory (DRAM)



Secondary Memory:
The Larger Storage



Secondary Memory:
The Larger Storage



Secondary Memory:
The concept of virtual Memory

The four questions

Q1: Where can a block be placed in Main Memory?

Q2: How is a block found if it is Main Memory?

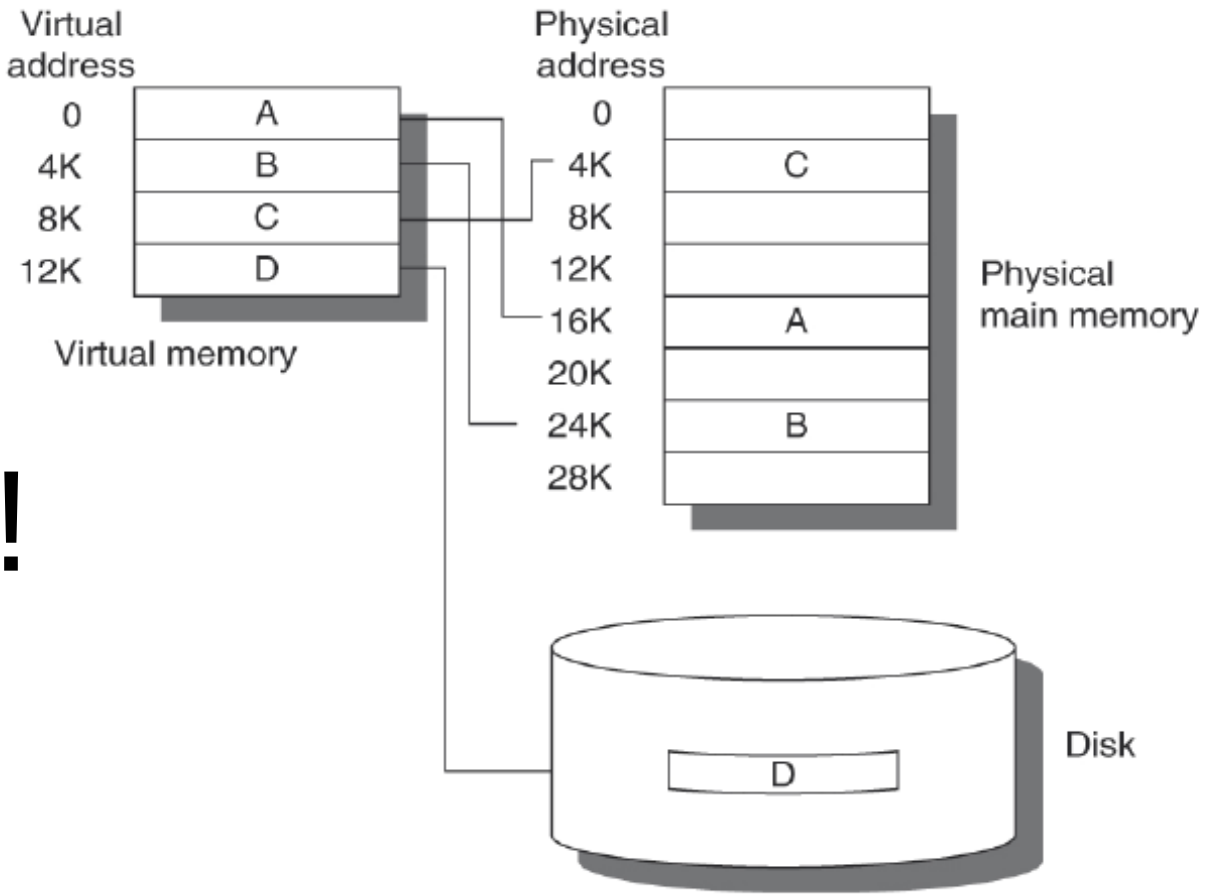
Q3: Which block should be replaced on Virtual Memory Miss?

Q4: What happens on Write?

Secondary Memory:
The concept of virtual Memory

Q1: Where can a block be placed in Main Memory?

Will all those ideas of cache memory can be applied in this scenario?

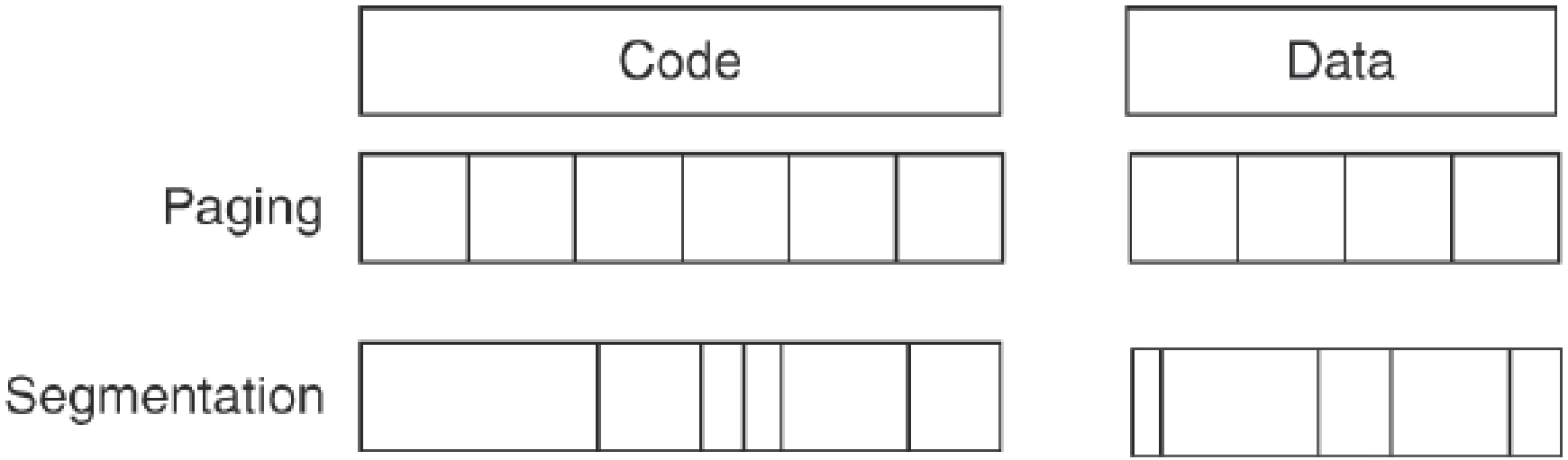


Any where!

Secondary Memory:
The concept of virtual Memory

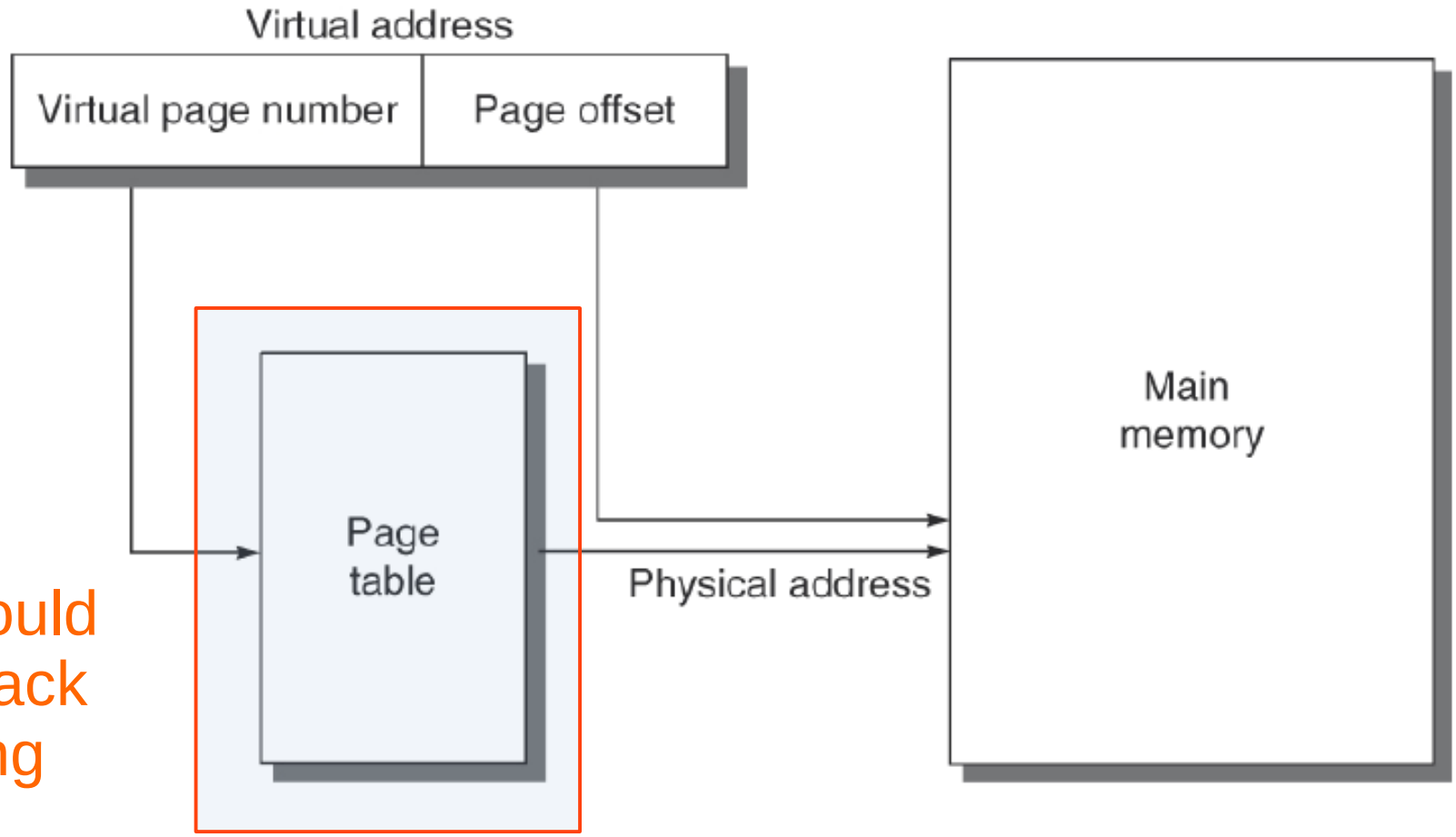
There can be two ways the blocks can be sized.

Fixed size block: **Page**
Variable size block: **Segment**



Secondary Memory:
The concept of virtual Memory

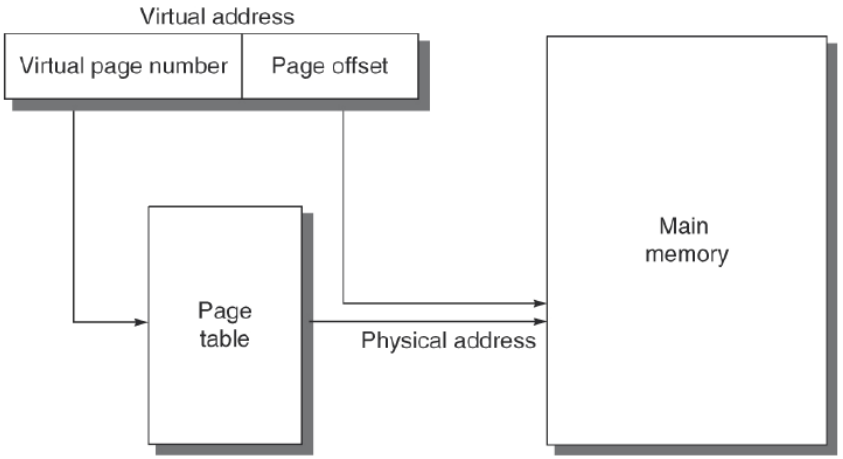
Q2: How is a block found if it is Main Memory?



Someone should be keeping track of who is going where

Secondary Memory:
The concept of virtual Memory

Q2: How is a block found if it is in Main Memory?



Size of Page table?

Example:

Let the virtual address is of 32 bit, Page size of 4 KB, and 4 byte per page table entry

Then,

Size of the page table would be = total pages X 4 bytes
= (2^32 / 2^12) X 4 bytes = 4 MB

So larger the page sizes lesser the page table size!

What are the other ways to reduce the page table size?

Secondary Memory:
The concept of virtual Memory

Q2: How is a block found if it is in Main Memory?

How to reduce the page table size?

How about this idea: instead of having an entry for each page in virtual address space, if can have entry for each frame of physical address space?

Inverted Page Table:

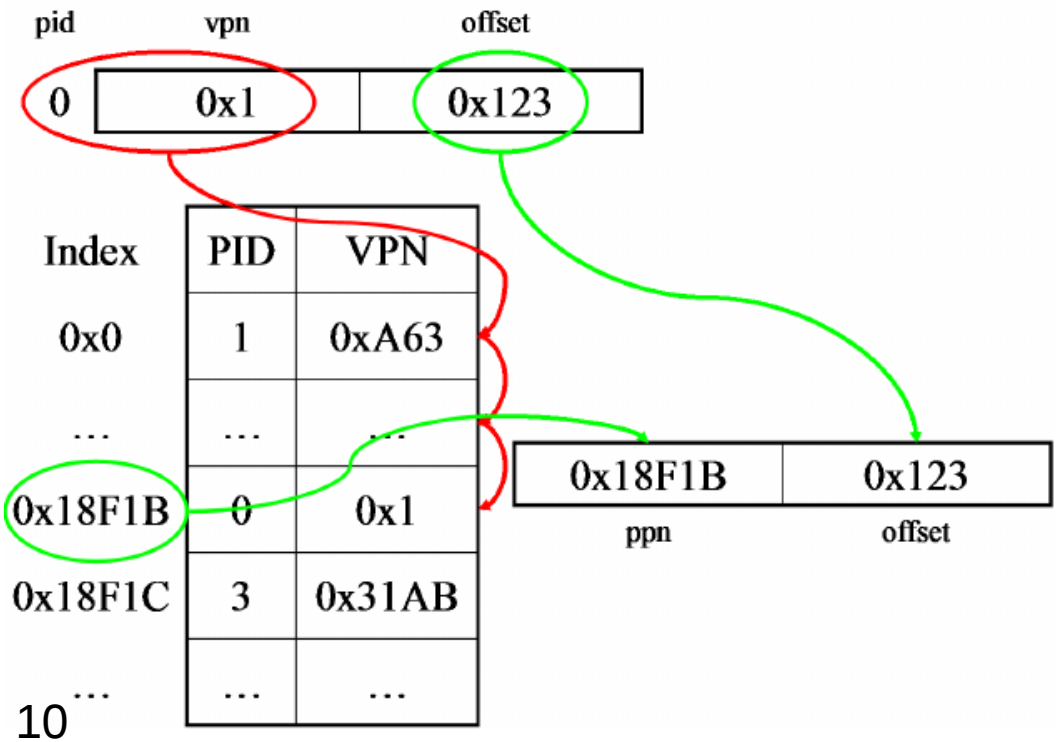


Figure ack:
Amir Kamil, CS162, UC Berkely Discussion 10

Secondary Memory:
The concept of virtual Memory

The four questions

Q1: Where can a block be placed in Main Memory?

Q2: How is a block found if it is Main Memory?

Q3: Which block should be replaced on Virtual Memory Miss? (Page Fault)

Q4: What happens on Write?

Secondary Memory:
The concept of virtual Memory

Q3: Which block should be replaced on Virtual Memory Miss? (Page Fault)

The primary objective is to minimize the page fault!

Page replacement policy:

- Optimal (The god's policy)
- **LRU: Least recent used (past is equivalent with future)**
- FIFO: First in first out (you have stayed enough)
- Random: (Fine with 50% accuracy)
- LFU: Least frequently used (no place for lazy guys)
- Least Recently Used K (LUR-K) (LRU + LFU)
- Adaptive Replacement Cache (track the dynamic behavior)

A good read theoretical analyis:
Alfred V Aho et al, Principles of Optimal Page Replacement, Journal of the ACM (JACM), 1971

Secondary Memory:
The concept of virtual Memory

Q3: Which block should be replaced on Virtual Memory Miss? (Page Fault)

The primary objective is to minimize the page fault!

Operating system is the decision maker, architect/designer provides facility such as *use bit* or *reference bit*.

Secondary Memory:
The concept of virtual Memory

Q3: What happen on Write?

Write through?

Or

Write back?

Must consider the cost of write.
Avoid unnecessary write.
Hardware support like dirty bit is needed.

Secondary Memory:
The concept of virtual Memory

How to improve the address translation?

Virtual to physical

Processor generates the virtual address, but the things are kept in physical location!

Secondary Memory:
The concept of virtual Memory

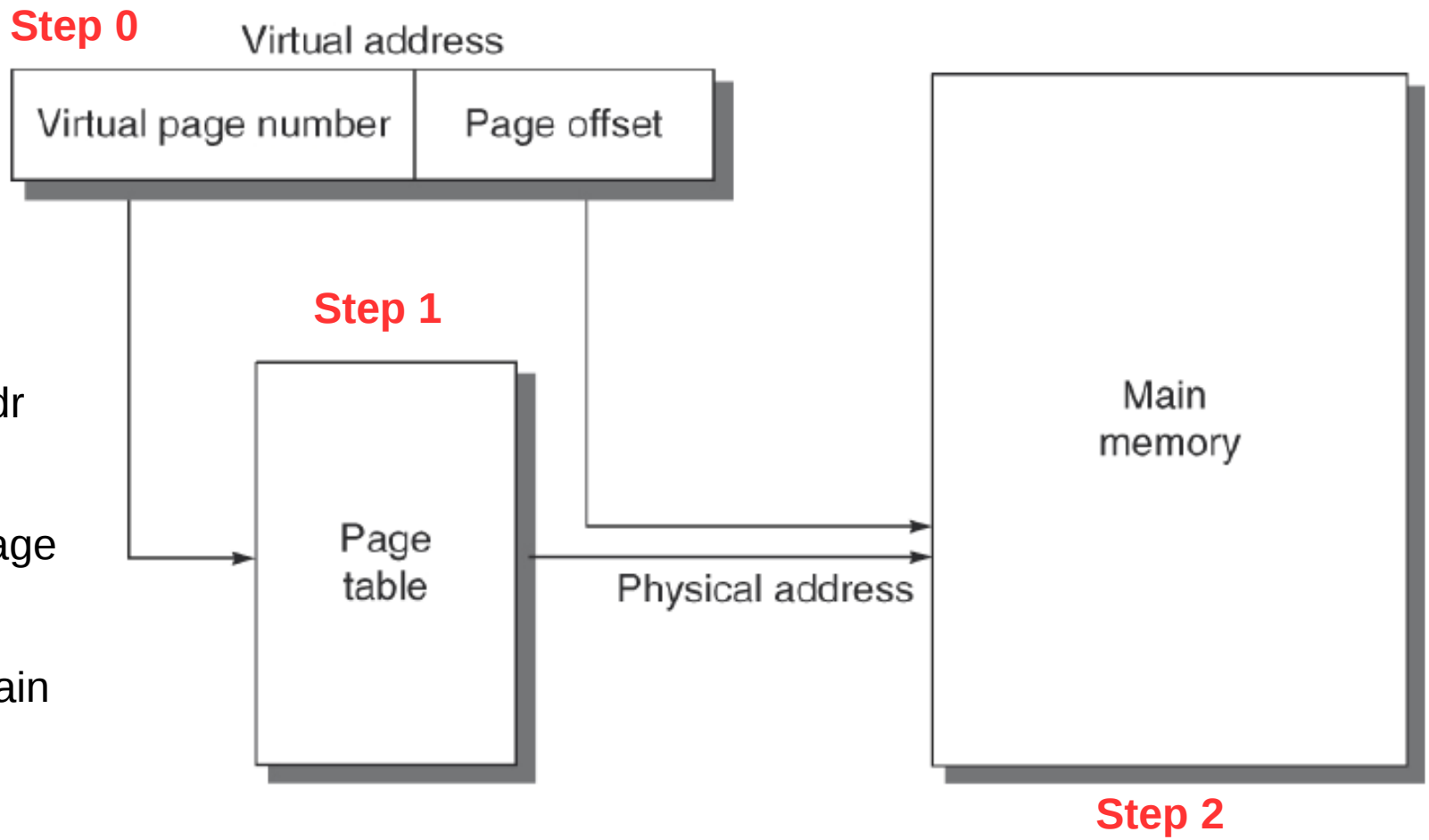
How to improve the address translation? Virtual to physical

What are the steps involved to get the desired data from virtual addr?

Step 0: Virtual addr from processor

Step 1: Access Page table

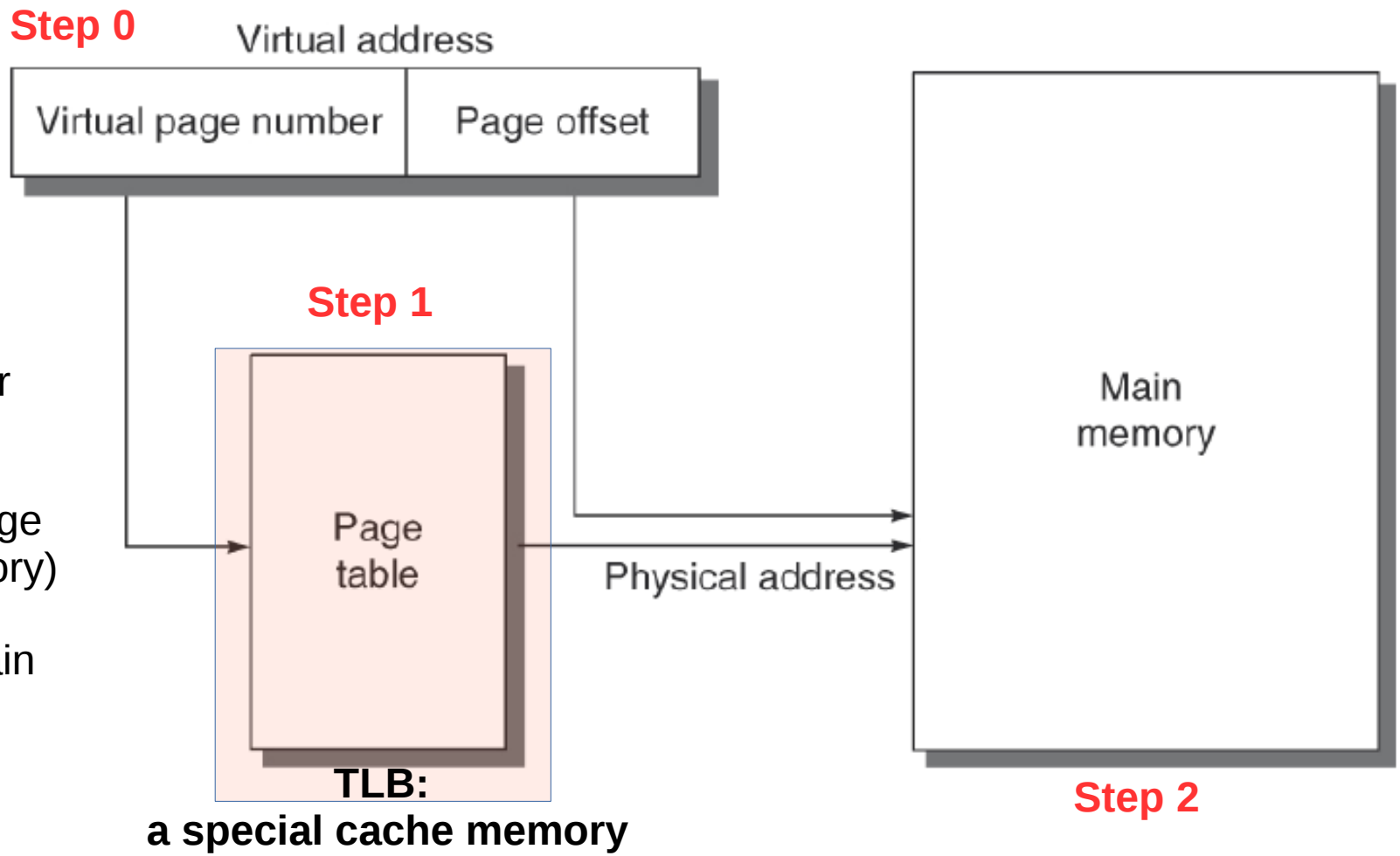
Step 2: Access main memory



Secondary Memory:
The concept of virtual Memory

How to improve the address translation?

TLB: Translation look-aside buffer or Translation buffer



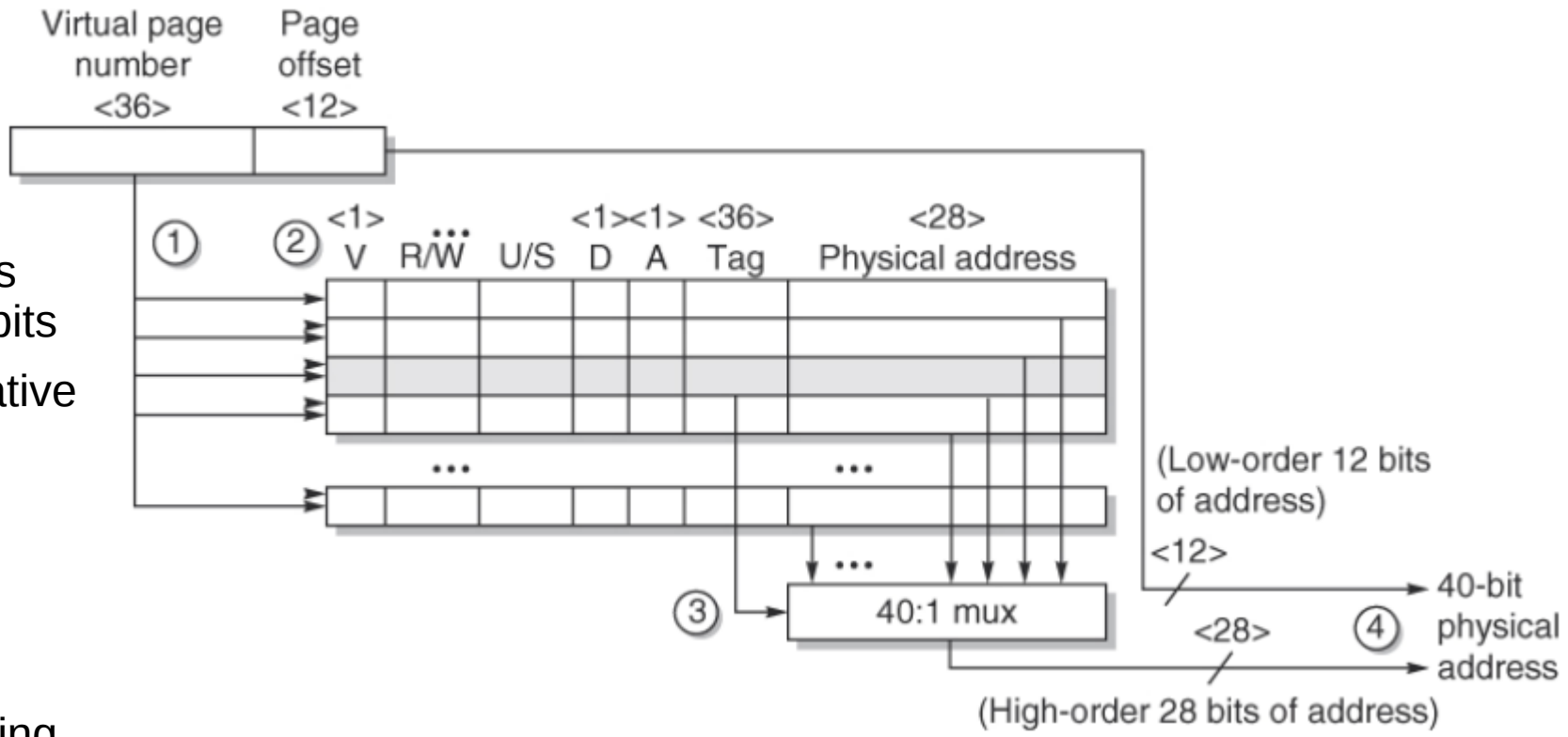
Step 0: Virtual addr from processor

Step 1: Access Page table (cache memory)

Step 2: Access main memory

Secondary Memory:
The concept of virtual Memory

Example: Opteron Data TLB



Virtual addr: 48 bits
Physical addr: 40 bits

TLB is fully associative
40 entries

Step1: tag matching
Step2: protection check
Step3: multiplexer selecting one line from 40
Step4: combining physical addr with offset

Secondary Memory:
The concept of virtual Memory

How to improve the address translation?

Page Size: Large, small, or average?

Good news for Large page size:

- The size of the page table is inversely proportional to the page size; therefore memory can be saved by making page size bigger.
- Larger page size can allow larger caches with fast hit times.
- Transferring larger page to or from secondary storage is more efficient than transferring smaller page.
- Larger page size helps in mapping more memory efficiently, thereby reduces TLB misses

Argument for smaller page size?

- Avoid internal fragmentation

Secondary Memory:
The concept of virtual Memory

Summary of the Address translation:

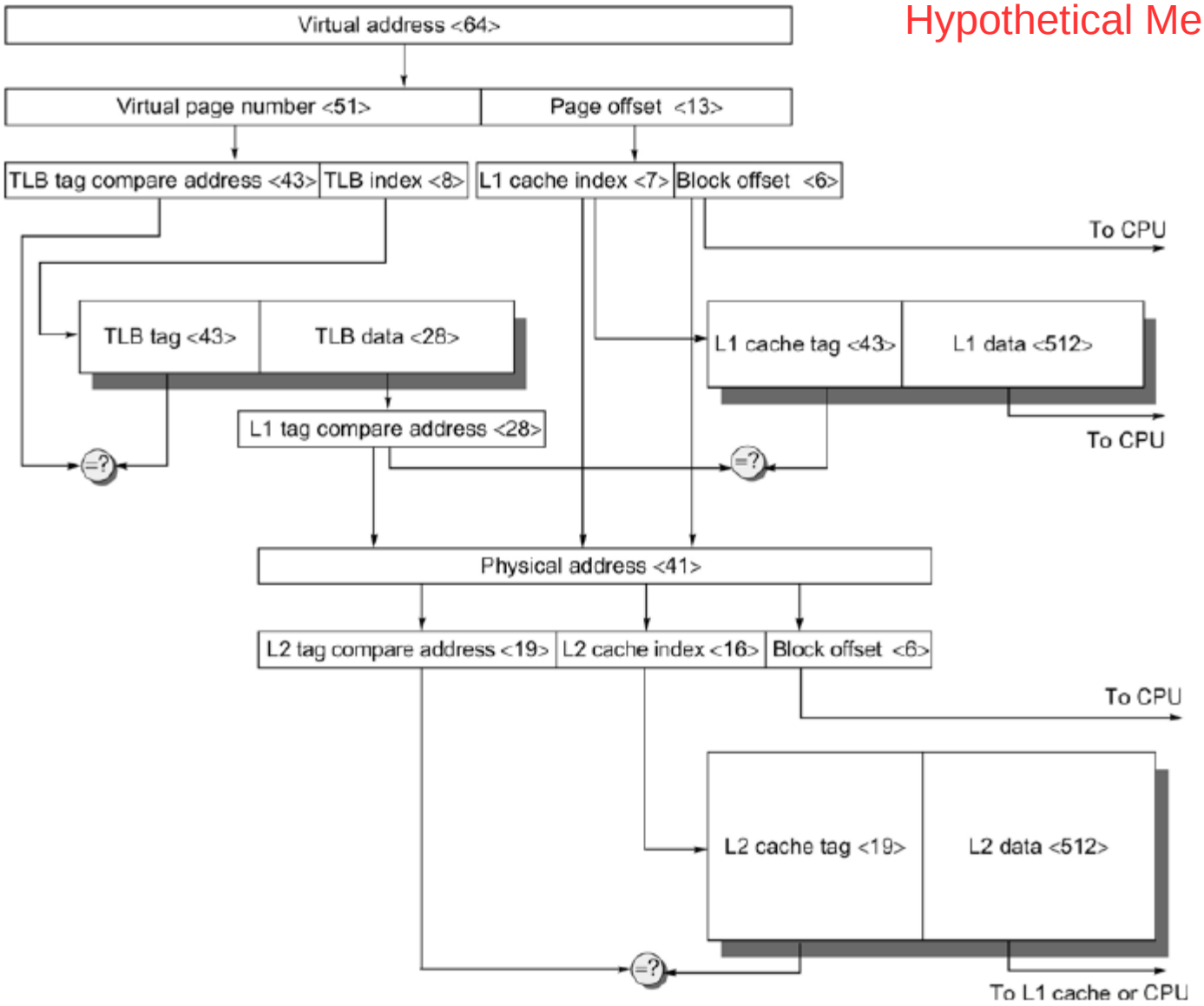
Virtual address
TLB
Pages
Main memory
Level 2 cache
Level 1 cache

Secondary Memory:
Virtual address to Cache Memory

Virtual Memory

Main Memory

Cache Memory



Secondary Memory:
Virtual address to Cache Memory

Page size: 8 KiB
So, page offset = 13 bits

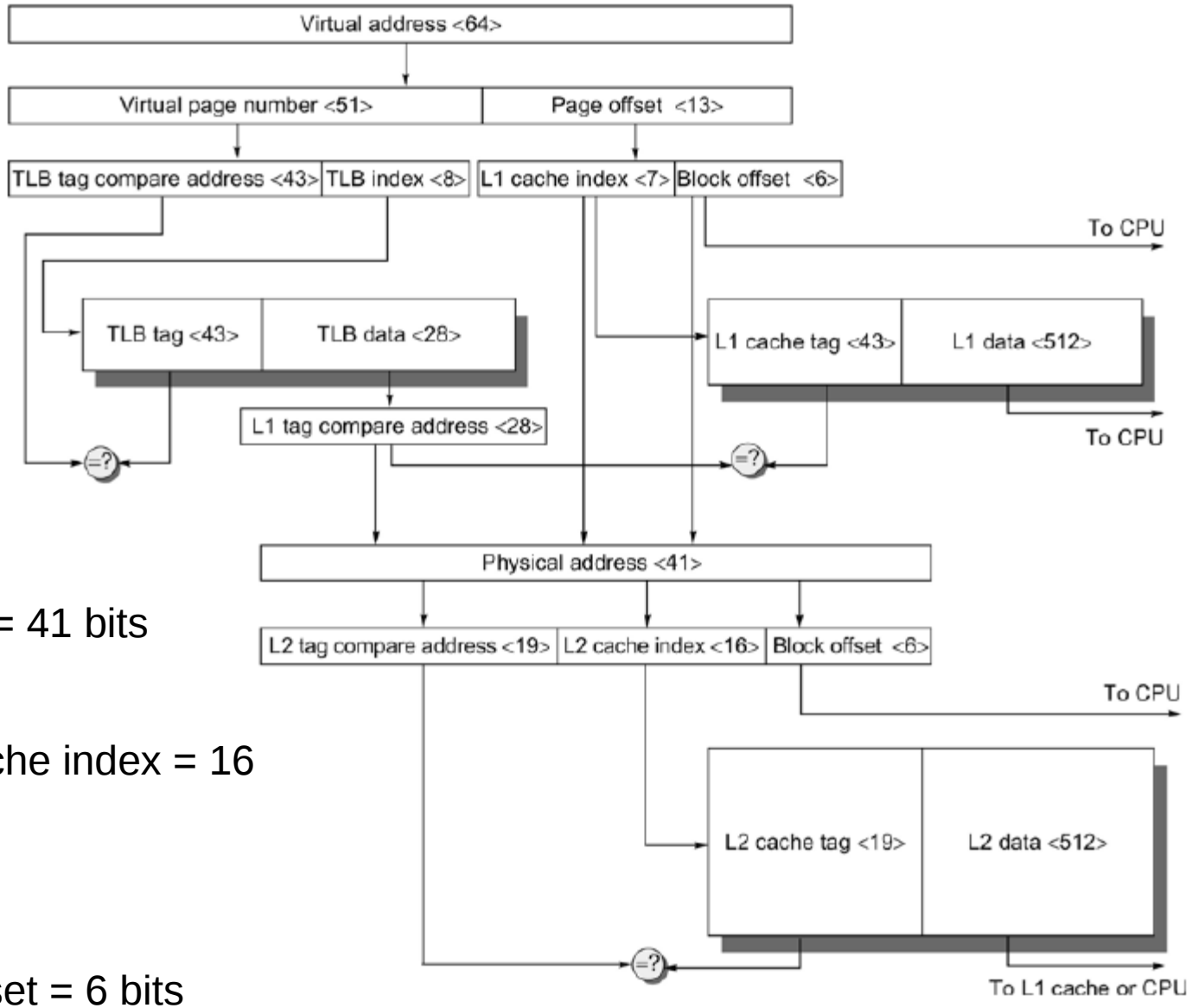
So, 64 – 13 = 51 should be
Page number

TLB is direct mapped cache
With 256 entries.
So, TLB index = 8 bits
Remaining 51 – 8 = 43 bits will
Be TLB tag.

Physical address=
TLB data.L1Index.BlockOffset = 41 bits

L2 is direct mapped 4 MiB
Block size is 64 byte, so L2 cache index = 16 bits
Remaining will be L2 tag bits.

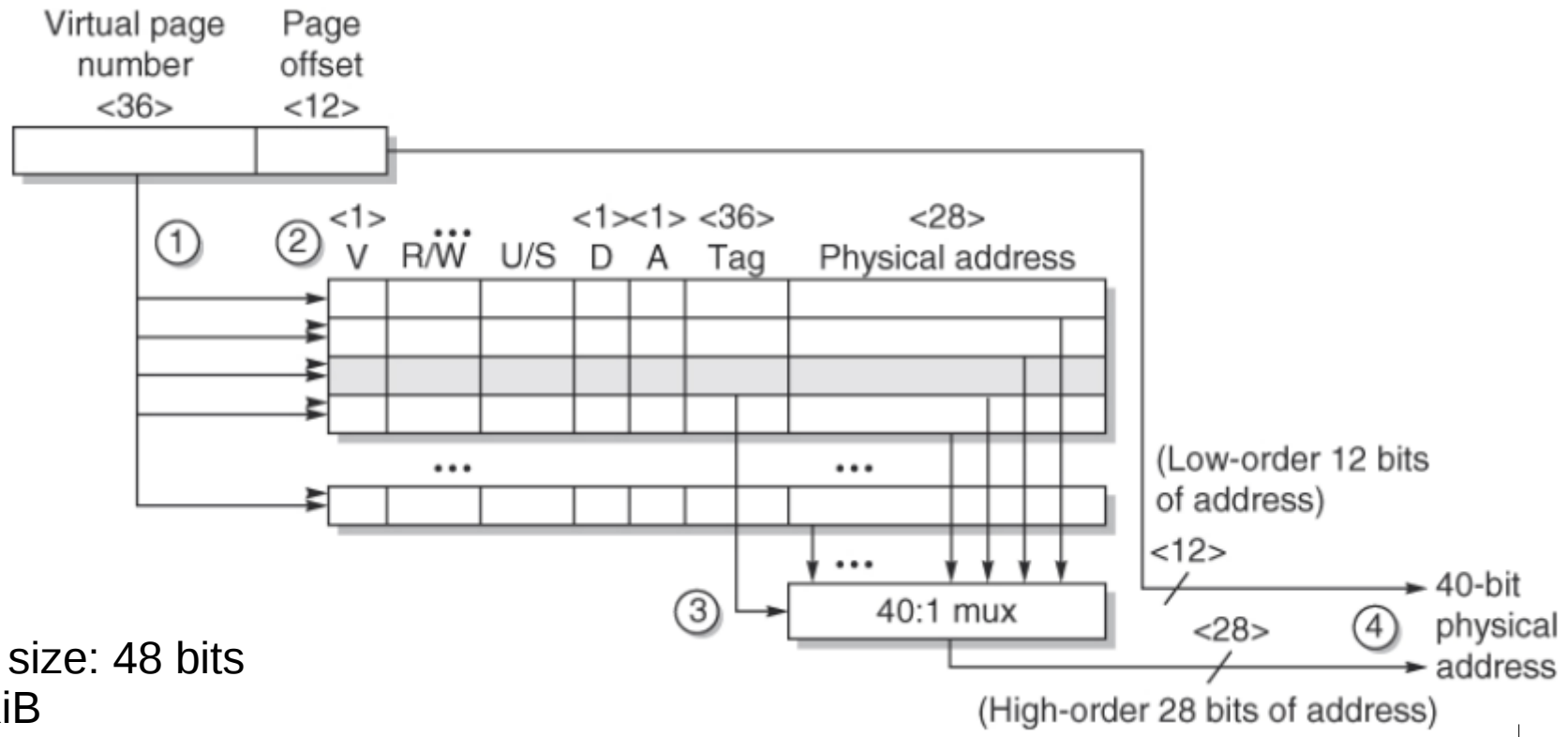
L1 is direct mapped 8 KiB
So index = 7 bits and block offset = 6 bits
L1 is virtually indexed physically tagged.



Secondary Memory:
TLB and Cache Address Together

Example: AMD Opteron

Address translation using TLB



Vitual address size: 48 bits
Page size: 4 KiB

TLB fully associative with 40 entries

Physical address = 40 bits

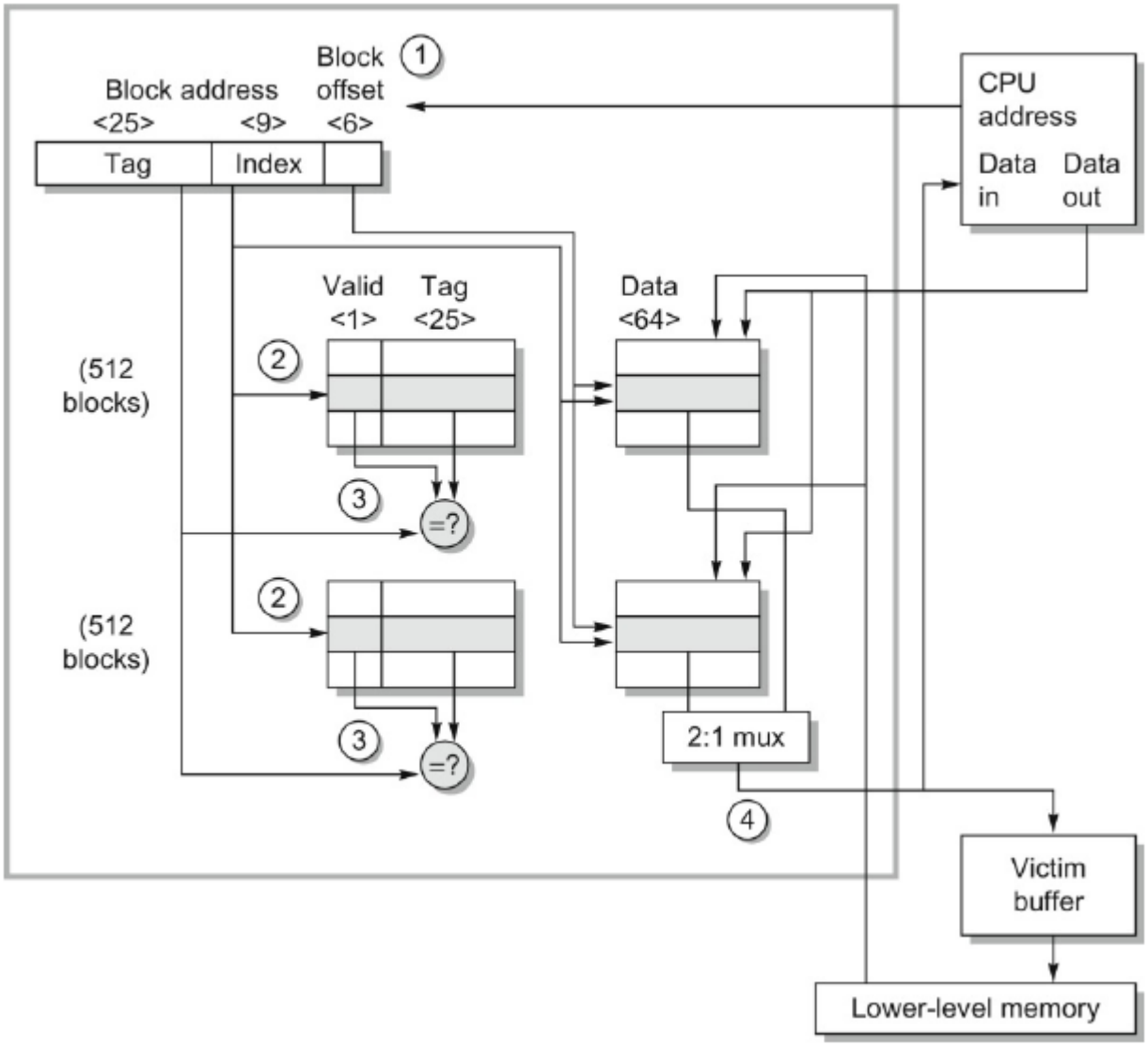
To Cache Memory

Secondary Memory:
TLB and Cache Address Together

Example: AMD Opteron Data Cache

From Physical Memory to Cache mapping

Data cache size = 64 KiB,
Two-way sett associative,
Block size = 64 byte
Offset = 6 bits
Total no of set = 512
Set index = 9 bits
Tag bits = 25 bits



Secondary Memory:
Paged Virtual Memory

Example: 64 bit AMD Opteron Memory Management

AMD64 Virtual Address space: 64 bits
AMD64 Physical address space: 52 bits

So, AMD64 architecture has provision for 64 bit to 52 bit physical addressing capability

The Opteron however uses only:

48 bit virtual address and
40 bit physical address

Variable **Page sizes:**
4 KiB, 2MiB, and 4MiB
(This is a sort of segment)

For 4 KiB page size the
Page offset = 12 bits

AMD Opteron considers
Page table size = Page size,
Each page table entry = 64 bits,
Total entry in a page table = 512,
So index bit = 9 bits

Secondary Memory:
Paged Virtual Memory

Example: 64 bit AMD Opteron Memory Management

AMD64 Virtual Address space: 64 bits
AMD64 Physical address space: 52 bits

So, AMD64 architecture has provision for 64 bit to 52 bit physical addressing capability

The Opteron however uses only:

48 bit virtual address and
40 bit physical address

Variable **Page sizes:**
4 KiB, 2MiB, and 4MiB
(This is a sort of segment)

For 4 KiB page size the
Page offset = 12 bits

AMD Opteron considers
Page table size = Page size,
Each page table entry = 64 bits,
Total entry in a page table = 512,
So index bit = 9 bits

Total No of Page Levels:

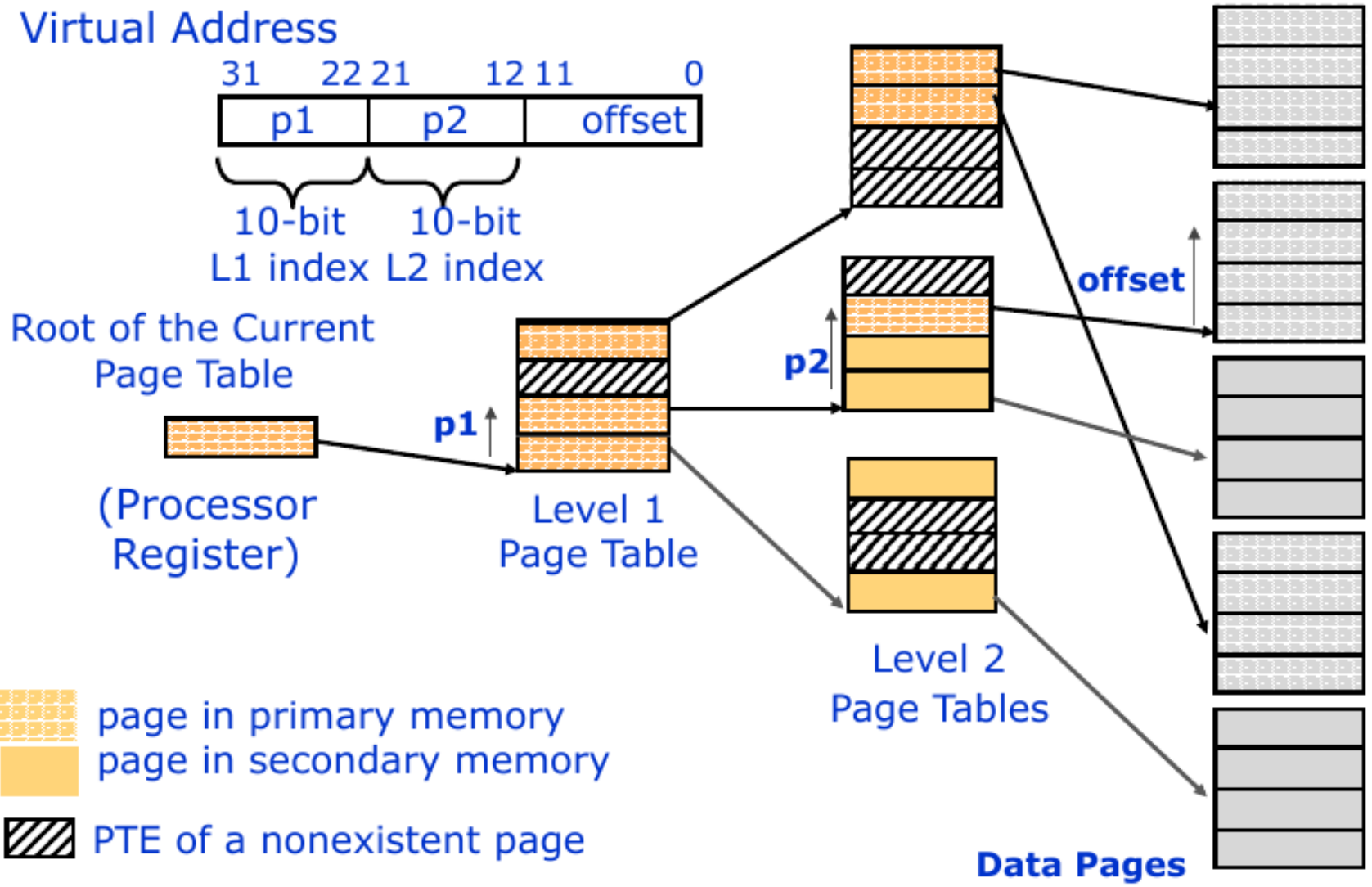
Total addressable pages:
2^36 (why? 48 -12)

How many page table entries a page can accommodate?
= 2^9

So, how many page levels are required:
$36 - 9x = 0$, where x is levels
So, $x = 4$

Secondary Memory:
Paged Virtual Memory

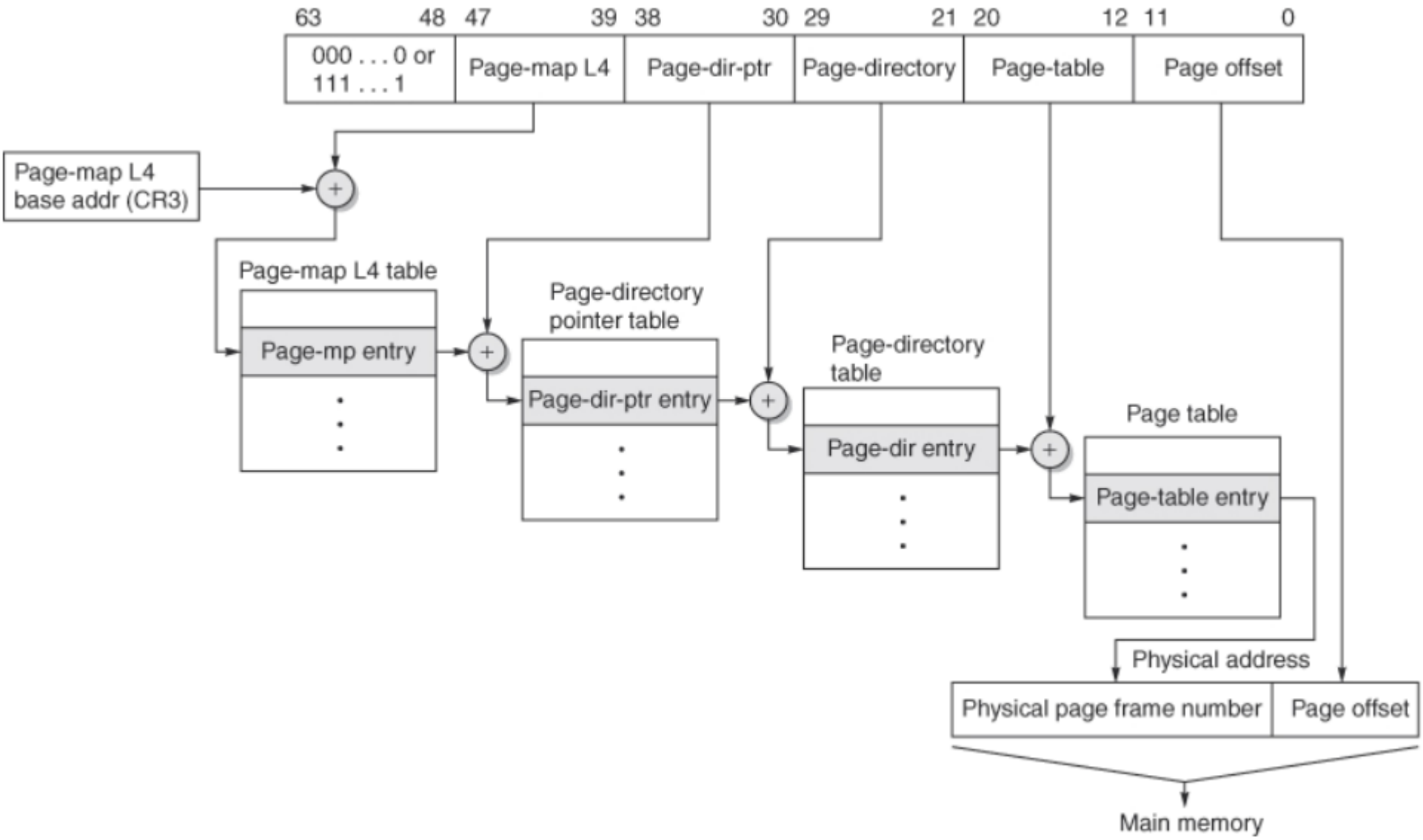
An hypothetical example of Paged Virtual Memory



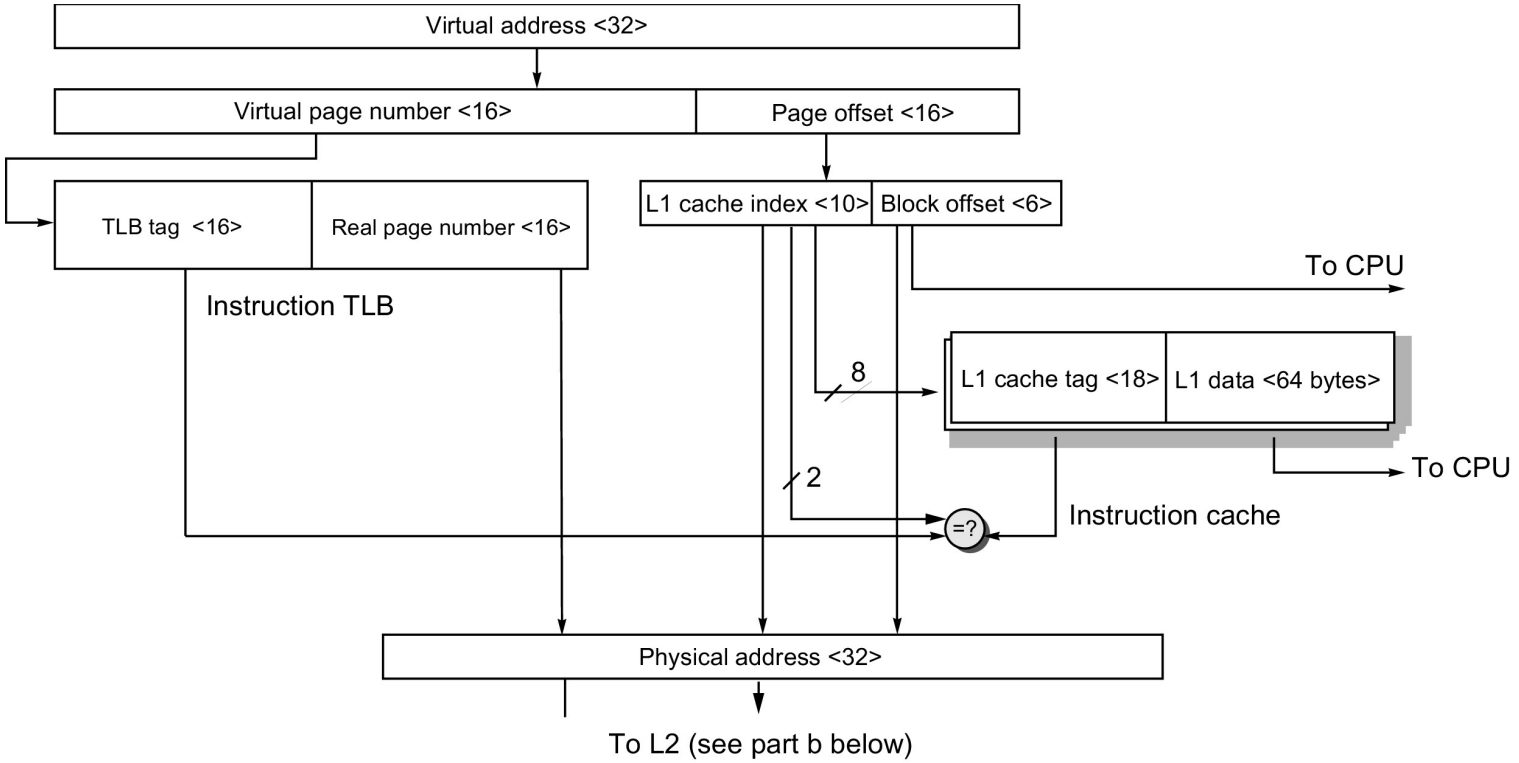
Acknowledgment: diagram taken from CS240 at UCSD

Secondary Memory:
Paged Virtual Memory

Example: 64 bit AMD Opteron Memory Management



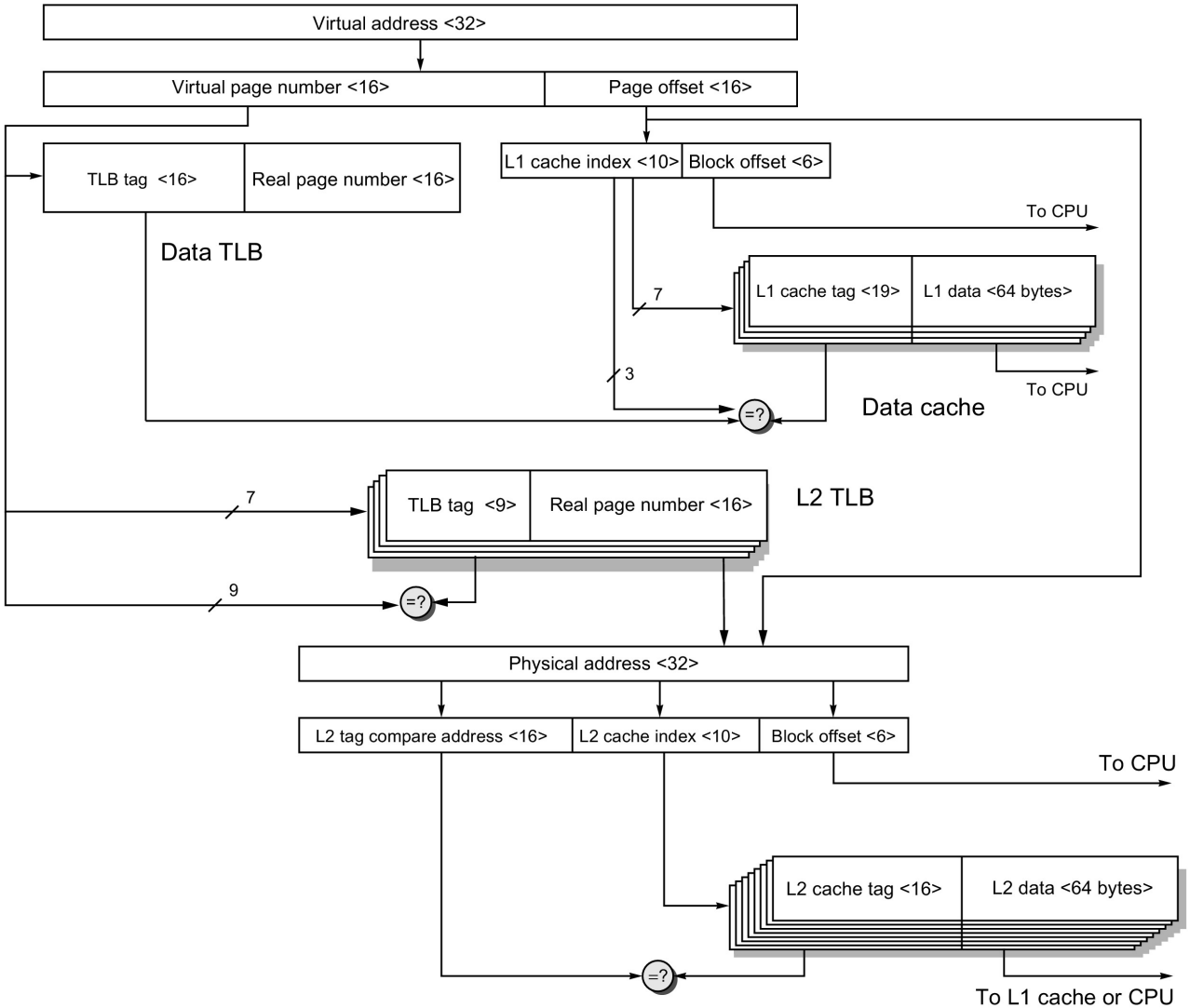
Secondary Memory:
Virtual to Cache in ARM Cortex-A53



(A) **The instruction access path**

Reference: Figure 2.20, 6th Edition, Henessy and Patterson

Secondary Memory:
Virtual to Cache in ARM Cortex-A53



(B) **The data access path**

Reference: Figure 2.20, 6th Edition, Henessy and Patterson

On-going Research in VM

- Arka Basu, **Revisiting** Virtual Memory, PhD Thesis, University of Wisconsin
  http://research.cs.wisc.edu/multifacet/theses/arka_basu_phd.pdf
  (Currently Arka Basu is with CSA, IISc Bangalore)

- Mohit Saxena and Michael M Swift, FlashVM: Virtual Memory Management **on Flash**, Proc of USENIX Annual Technical Conference, 2010 [https://www.usenix.org/legacy/event/atc10/tech/full_papers/Saxena.pdf]

- Sparch Mittal, A Survey of Architecting **TLB**, CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE, Wiley InterScience, 2016 [http://raiith.iith.ac.in/2863/1/2017_CPE_Mittal_SurveyTLB.pdf]

- Jee Ho Ryoo, Nagendra Gulur, Shuang Song, Lizy K. John ,
  Rethinking TLB Designs in **Virtualized Environments**: A Very Large Part-of-Memory TLB, Proc of ISCA 2017

Thanks

Reference:

- Patterson and Henessay;
  Computer Organization and Design: Hardware/Software Interface, 5th Edition
  (Refere Chapter 5, Section 5.7)

- Henessy and Patterson; Computer Architecture: Quantitative Approach,
  (Refer Chapter 2)

Next Lecture:

Instruction Level Parallelism
Super Scalar Architecture