Page Table Indexing

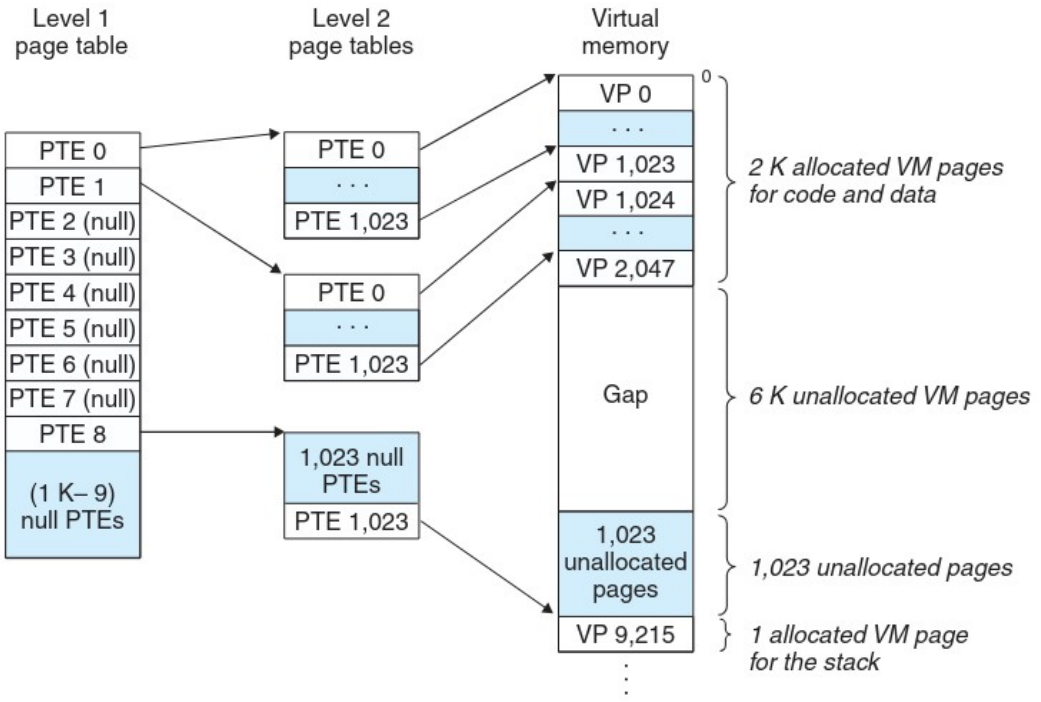
- n size of memory address
- p determines PAGE SIZE
- Address of memory
  - 32 bit
  - 64 bit

Symbol	Description
Basic parameters	
$N = 2^n$	Number of addresses in virtual address space
$M = 2^m$	Number of addresses in physical address space
$P = 2^p$	Page size (bytes)
Components of a virtual address (VA)	
VPO	Virtual page offset (bytes)
VPN	Virtual page number
TLBI	TLB index
TLBT	TLB tag
Components of a physical address (PA)	
PPO	Physical page offset (bytes)
PPN	Physical page number
CO	Byte offset within cache block
CI	Cache index
CT	Cache tag

Figure 9.11 **Summary of address translation symbols.**

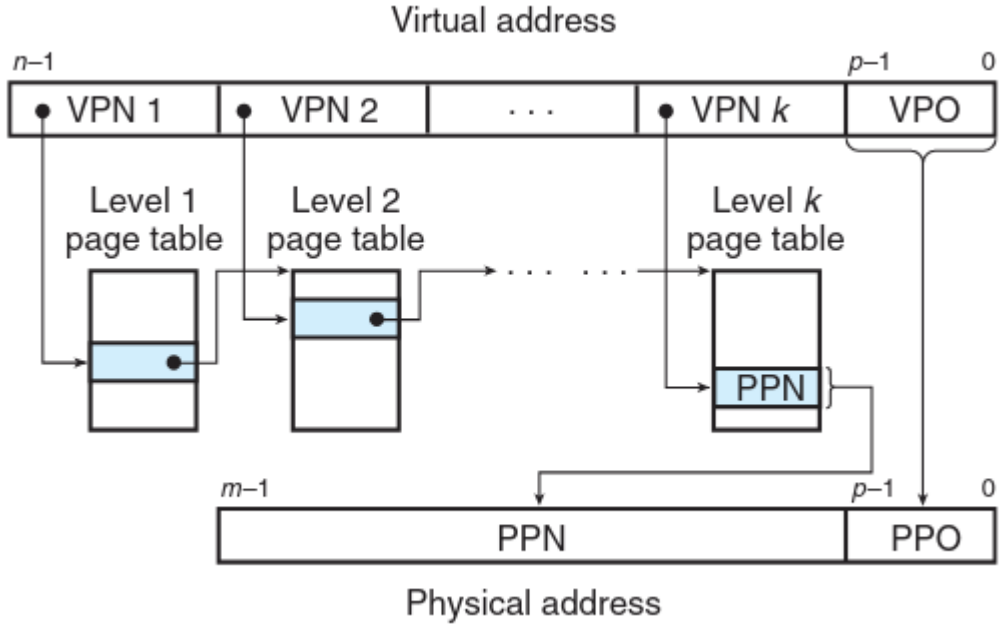
Levels of Indirection

- Using PTE locations
- Can get very involved



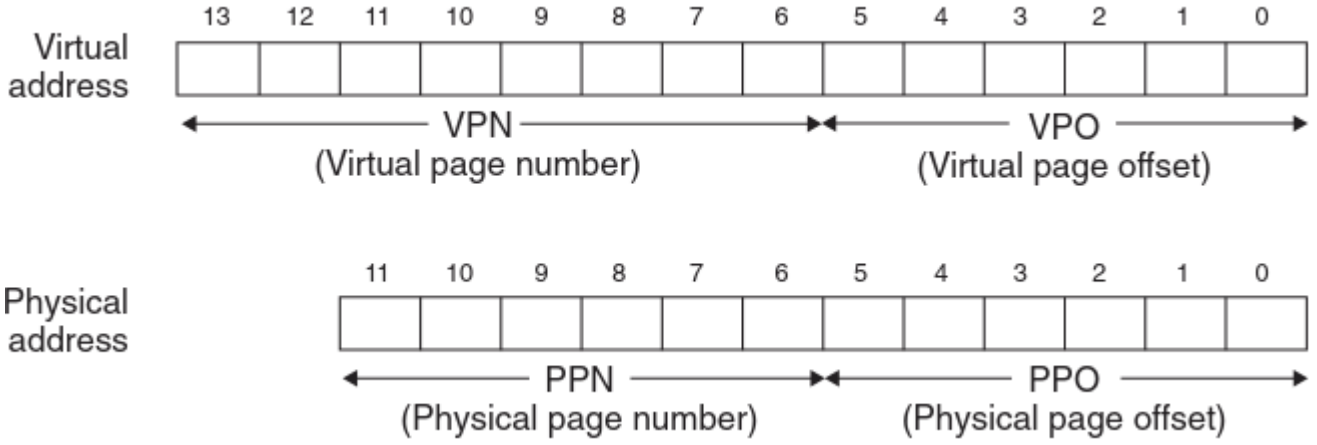
VPN Construction

- Solving problems with?
  - POINTERS
- Steal bits



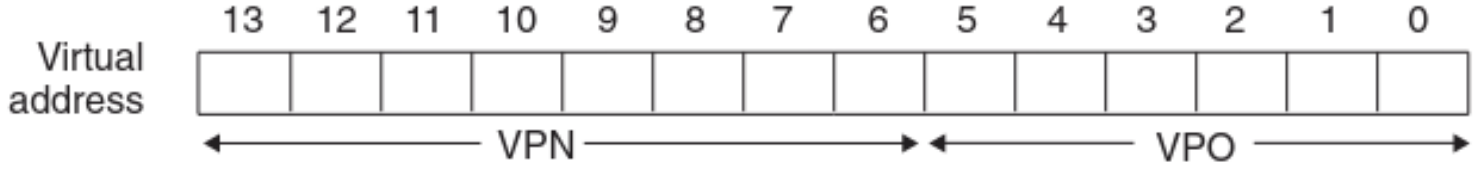
Organization of Memory

- 14 bits Virtual -- 12 bits Physical – 64 byte page



TLB First

- 4 sets – 16 entries total – 4-way associative



Set	Tag	PPN	Valid	Tag	PPN	Valid	Tag	PPN	Valid	Tag	PPN	Valid
0	03	–	0	09	0D	1	00	–	0	07	02	1
1	03	2D	1	02	–	0	04	–	0	0A	–	0
2	02	–	0	08	–	0	06	–	0	03	–	0
3	07	–	0	03	0D	1	0A	34	1	02	–	0

Page Table Next

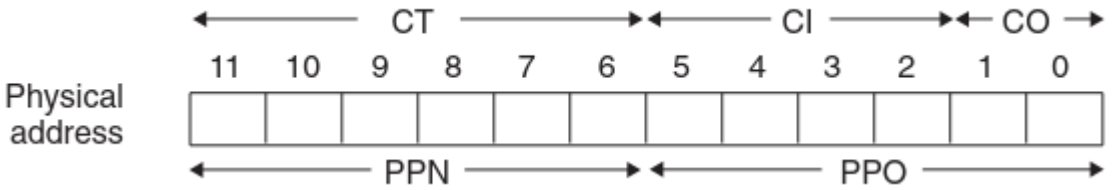
- How many total Pages are there?
- What does Valid mean?

VPN	PPN	Valid
00	28	1
01	—	0
02	33	1
03	02	1
04	—	0
05	16	1
06	—	0
07	—	0

VPN	PPN	Valid
08	13	1
09	17	1
0A	09	1
0B	—	0
0C	—	0
0D	2D	1
0E	11	1
0F	0D	1

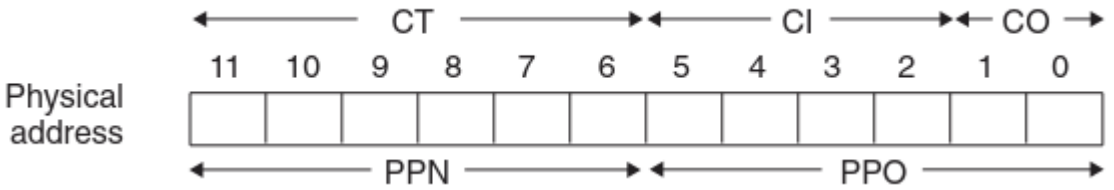
Cache (L1)

- Even the bits in the PPN are a mask.
- Cache Tag (CT)
- Cache Index (CI)
- Cache Offset (CO)



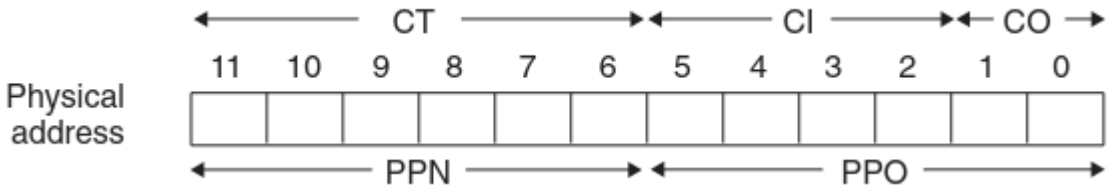
Cache (L1)

- Even the bits in the PPN are a mask.
- Cache Tag (CT)
- Cache Index (CI)
- Cache Offset (CO)



Cache (L1)

- Translation of Address
- This is where HIT or MISS



Idx	Tag	Valid	Blk 0	Blk 1	Blk 2	Blk 3
0	19	1	99	11	23	11
1	15	0	—	—	—	—
2	1B	1	00	02	04	08
3	36	0	—	—	—	—
4	32	1	43	6D	8F	09
5	0D	1	36	72	F0	1D
6	31	0	—	—	—	—
7	16	1	11	C2	DF	03
8	24	1	3A	00	51	89
9	2D	0	—	—	—	—
A	2D	1	93	15	DA	3B
B	0B	0	—	—	—	—
C	12	0	—	—	—	—
D	16	1	04	96	34	15
E	13	1	83	77	1B	D3
F	14	0	—	—	—	—

What a mess

- Each lookup costs time
- More complexity

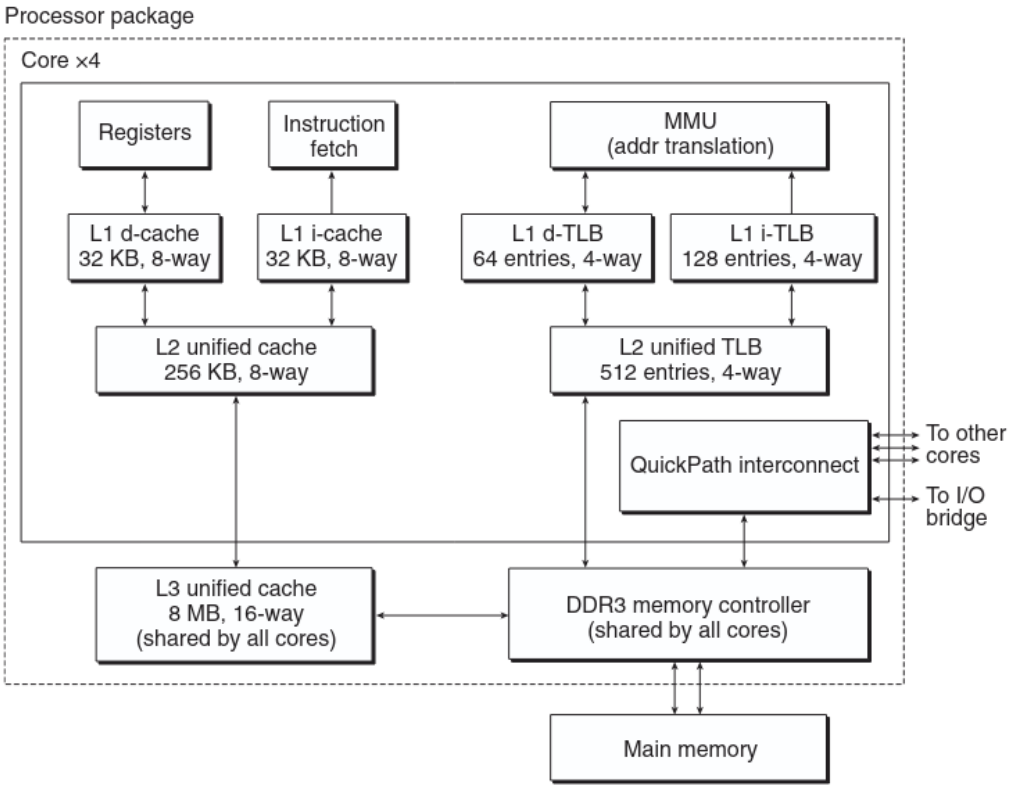
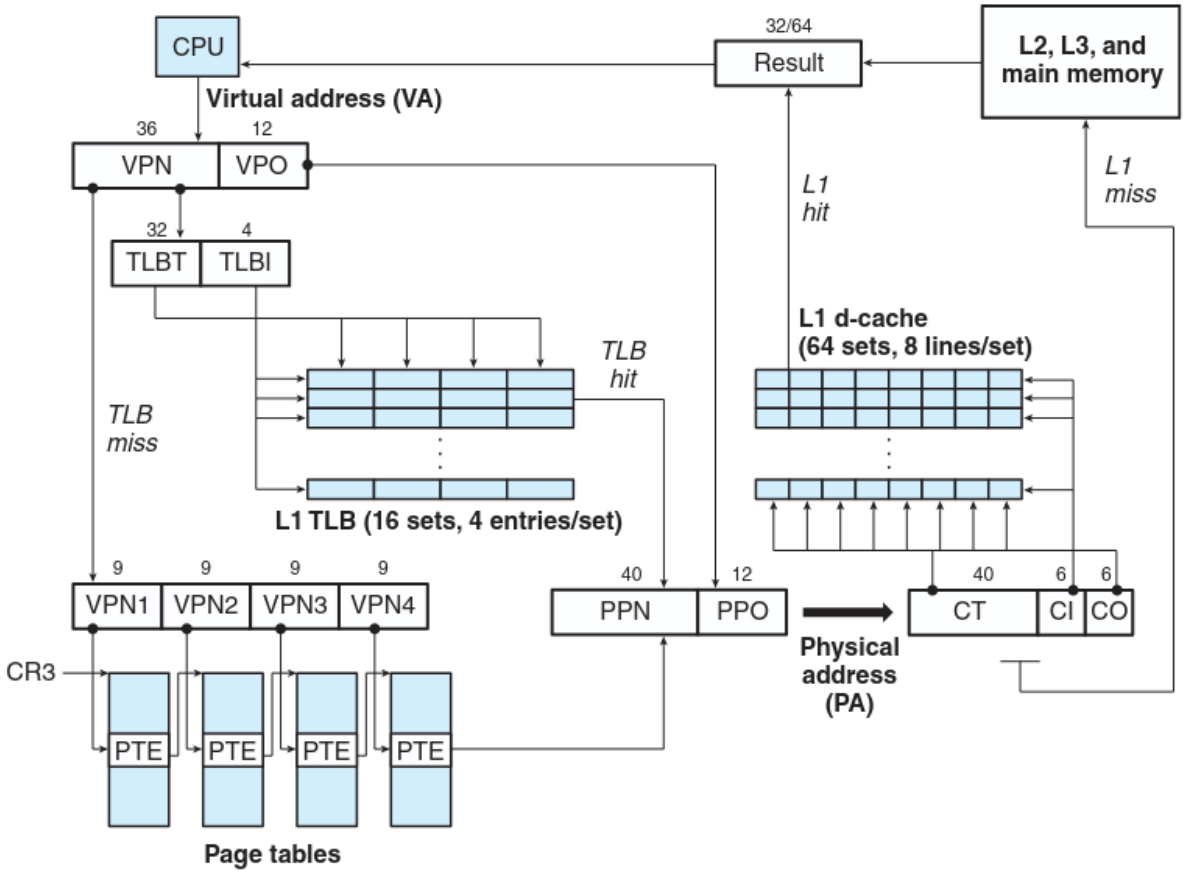
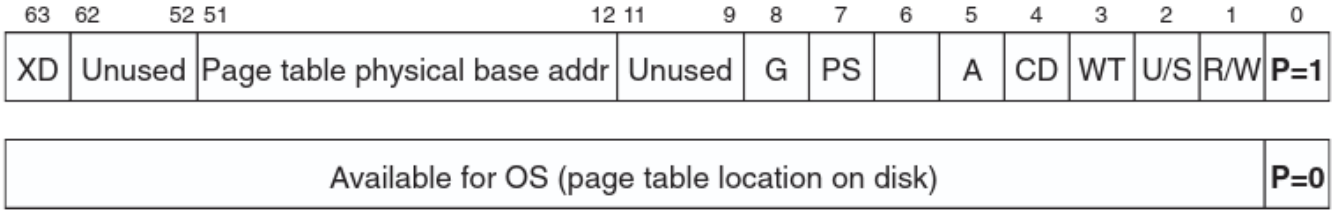


Figure 9.21 The Core i7 memory system.

What a mess

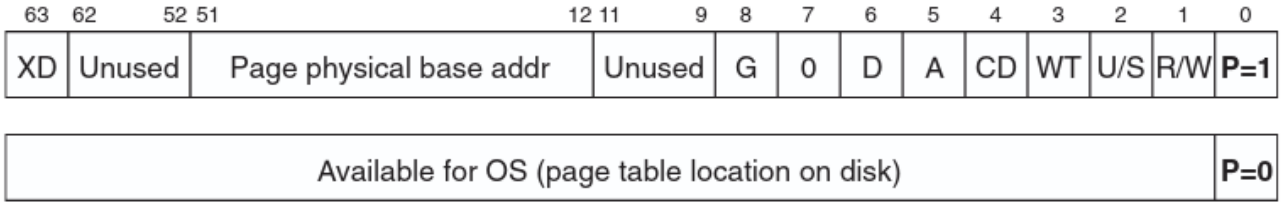
- 
CPU
32/64
Result
L2, L3, and main memory
Virtual address (VA)
36
12
VPN
VPO
L1 hit
L1 miss
32
4
TLBT
TLBI
L1 d-cache (64 sets, 8 lines/set)
TLB hit
TLB miss
L1 TLB (16 sets, 4 entries/set)
9
9
9
9
VPN1
VPN2
VPN3
VPN4
40
12
PPN
PPO
40
6
6
CT
CI
CO
Physical address (PA)
CR3
PTE
PTE
PTE
PTE
Page tables

Level 1/2/3 PTE (Intel)



Field	Description
P	Child page table present in physical memory (1) or not (0).
R/W	Read-only or read-write access permission for all reachable pages.
U/S	User or supervisor (kernel) mode access permission for all reachable pages.
WT	Write-through or write-back cache policy for the child page table.
CD	Caching disabled or enabled for the child page table.
A	Reference bit (set by MMU on reads and writes, cleared by software).
PS	Page size either 4 KB or 4 MB (defined for level 1 PTEs only).
Base addr	40 most significant bits of physical base address of child page table.
XD	Disable or enable instruction fetches from all pages reachable from this PTE.

Level 4(4KB) PTE (Intel)

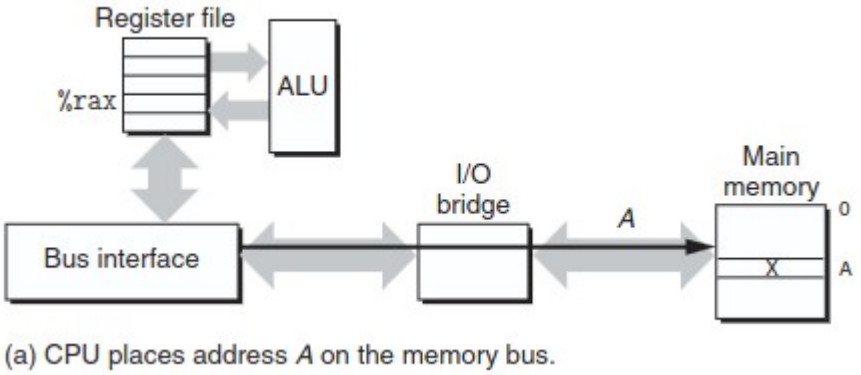


Field	Description
P	Child page present in physical memory (1) or not (0).
R/W	Read-only or read/write access permission for child page.
U/S	User or supervisor mode (kernel mode) access permission for child page.
WT	Write-through or write-back cache policy for the child page.
CD	Cache disabled or enabled.
A	Reference bit (set by MMU on reads and writes, cleared by software).
D	Dirty bit (set by MMU on writes, cleared by software).
G	Global page (don't evict from TLB on task switch).
Base addr	40 most significant bits of physical base address of child page.
XD	Disable or enable instruction fetches from the child page.

Reading From Memory pt1

Basic Memory I/O

Main Memory Miss

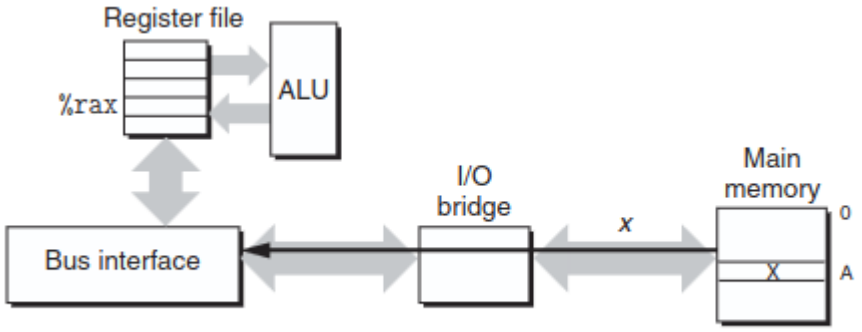


(a) CPU places address *A* on the memory bus.

Reading From Memory pt2

MMU and friends

Still not in %RAX

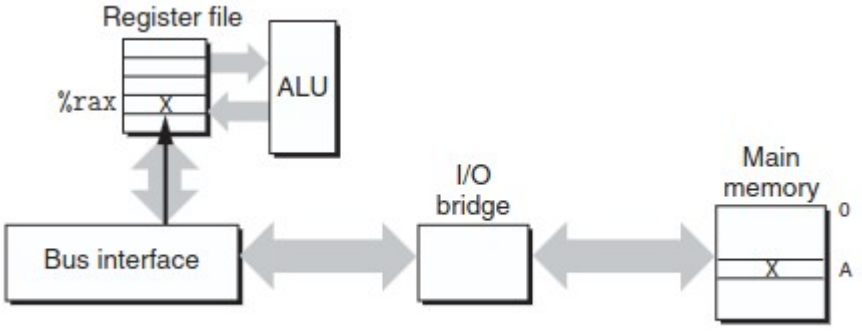


(b) Main memory reads *A* from the bus, retrieves word *x*, and places it on the bus.

Reading From Memory pt3

CPU still picks it up

Shoves it into its %RAX

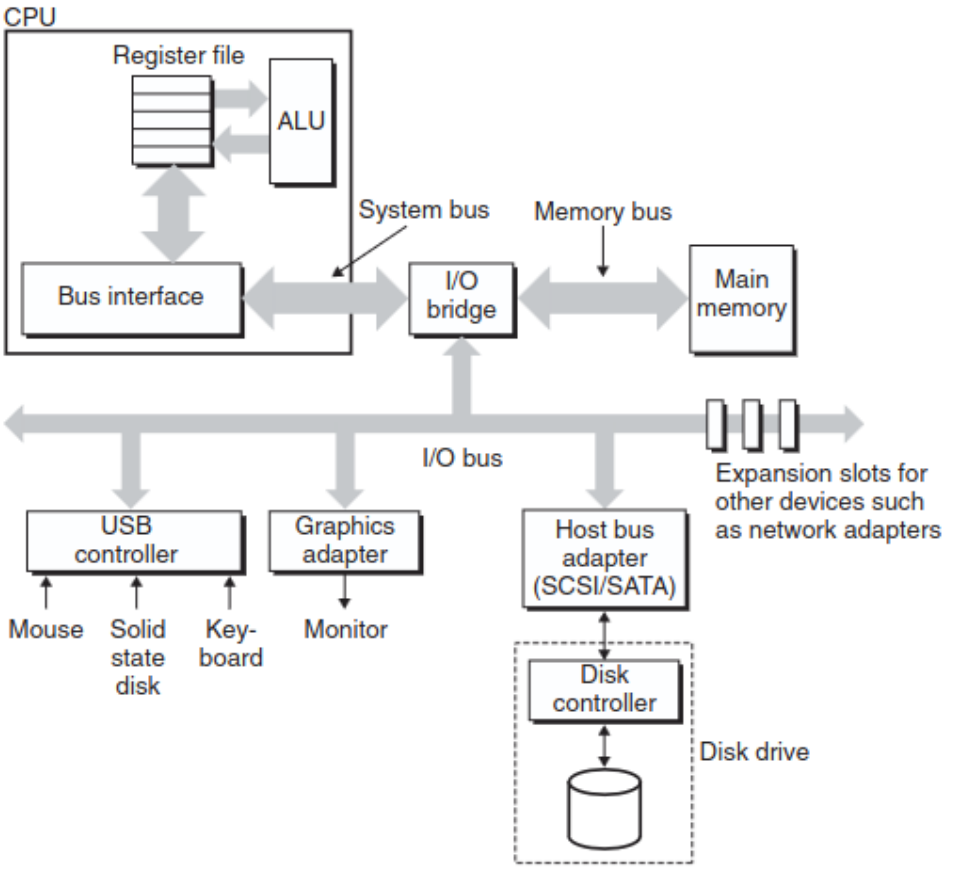


(c) CPU reads word *x* from the bus, and copies it into register %rax.

Reading From Disk Pt1

Starts the same

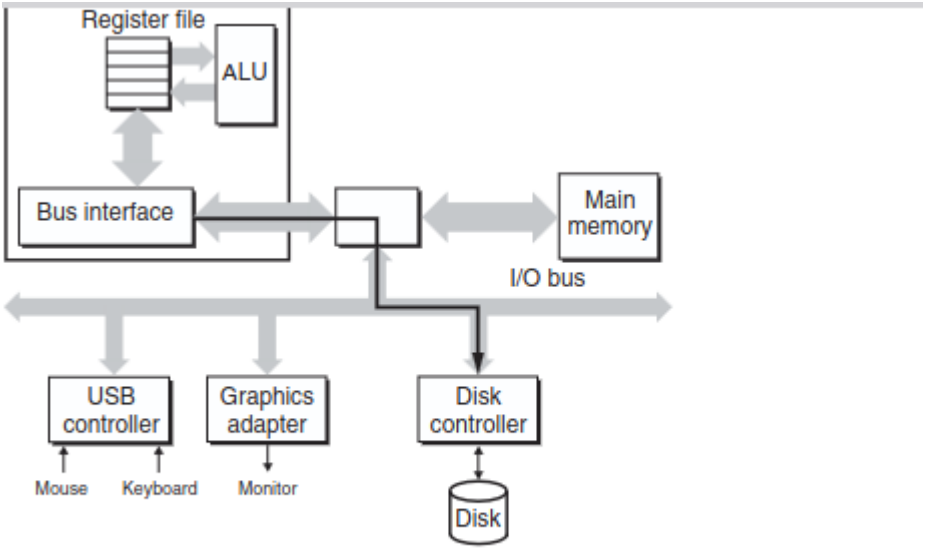
Extra wait step for device to put data on the I/O bus to main memory.



Reading From Disk Pt2

Send I/O command

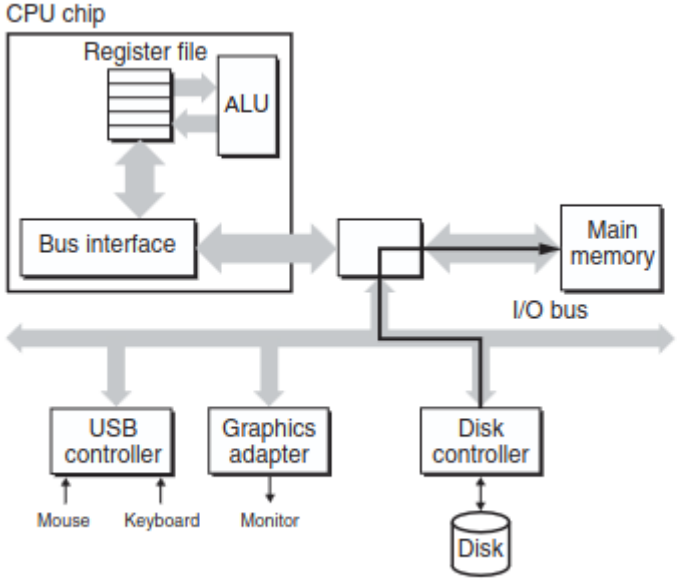
to bus and wait.



(a) The CPU initiates a disk read by writing a command, logical block number, and destination memory address to the memory-mapped address associated with the disk.

Reading From Disk Pt3

Disk controller itself puts the requested data into main memory (DMA)

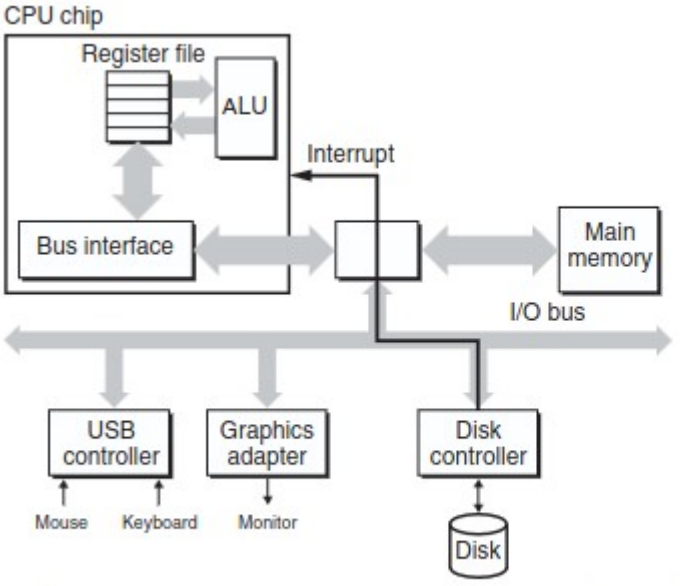


(b) The disk controller reads the sector and performs a DMA transfer into main memory.

Reading From Disk Pt4

Disk controller sends IRQ back to CPU to indicate “done”

CPU then treats the data as a normal mem request



(c) When the DMA transfer is complete, the disk controller notifies the CPU with an interrupt.

CUDA/Video Cards

All about Matrix Ops/Sec

All data is loaded

before execution