VE370 RC (Week 12)

Li Shi

2020.11.27

Before we start …
Recitation Class On-site Evaluation

- Nov 19 – Practical Teaching Workshop

- **Nov 27 – RC On-site Evaluation (today)**

Content

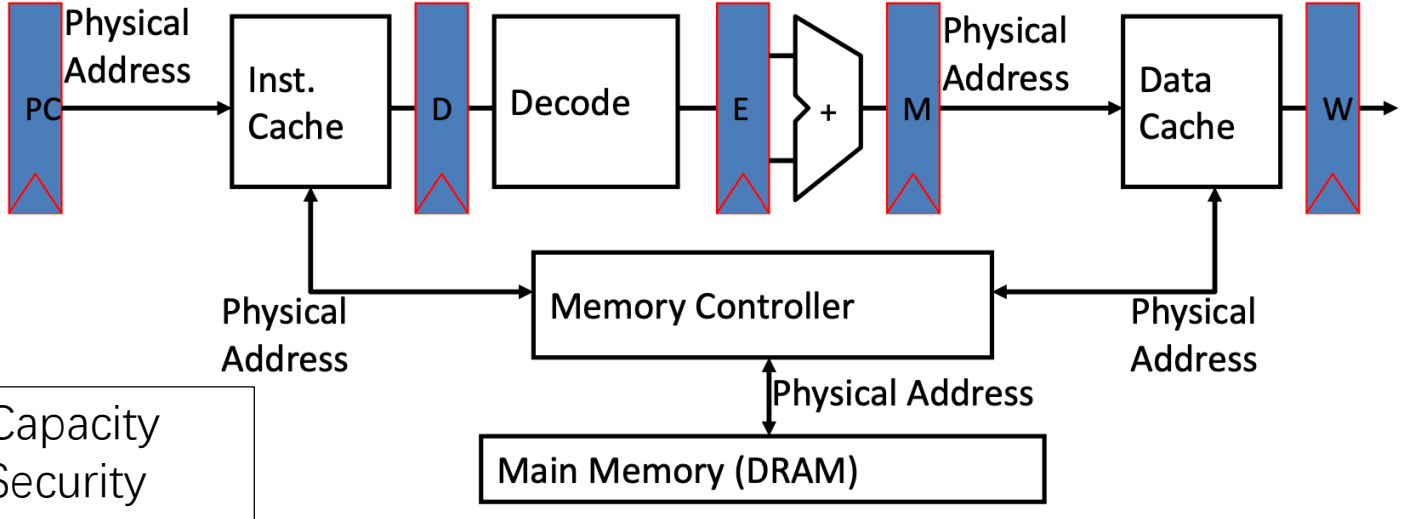
- Key concept 1: Motivation of virtual memory (VM)
- Key concept 2: Page table (PT)
- Key concept 3: Translation look-aside buffer (TLB)

Key concept 1: Motivation of VM
Issues with memory

- Issue 1: Capacity
  - Large program on hard drive vs. small capacity of memory
- Issue 2: Security
  - Program A may access to memory possessed by program B
- Issue 3: Complexity
  - We need to manage memory quickly and efficiently, but CPU already has many other issues

Key concept 1: Motivation of VM
Issues with memory from a historical view

- Solution 1: Bare machine

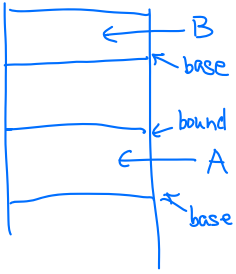


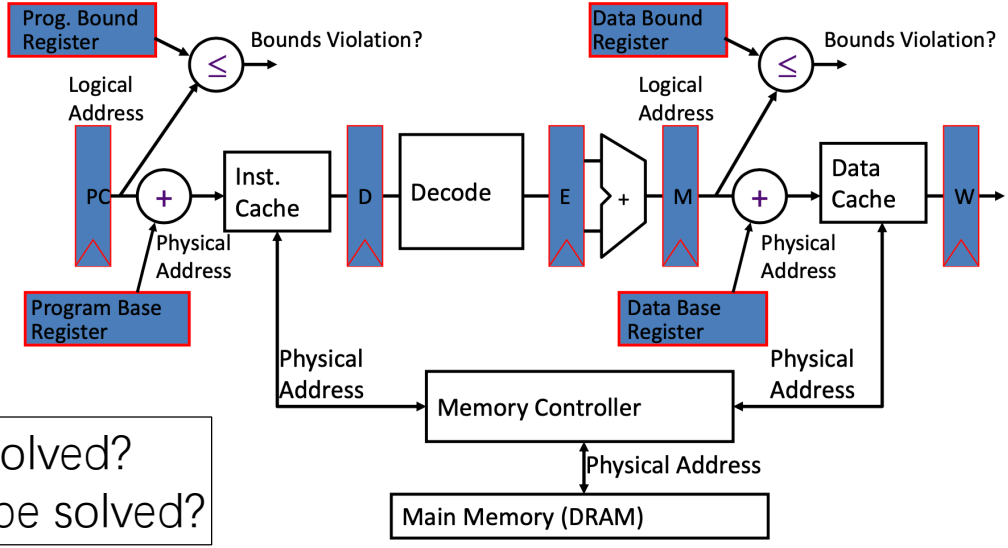
1) Capacity
2) Security
3) Complexity

This slide will not be covered in the exam.

Key concept 1: Motivation of VM
Issues with memory from a historical view

- Solution 2: Base and bound machine

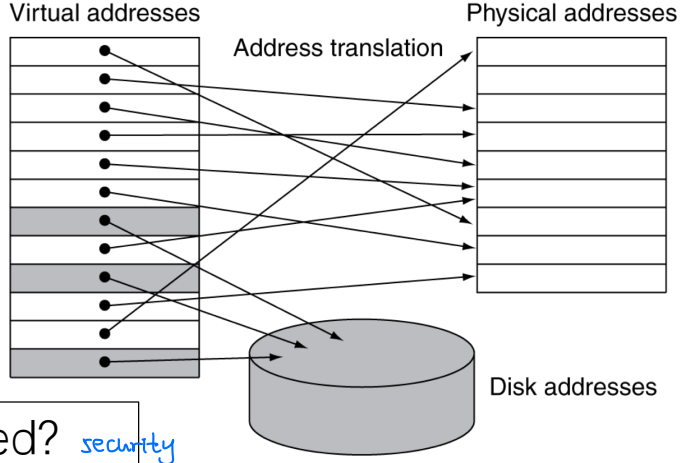




What issues **can** be solved?
What issues **cannot** be solved?

Key concept 1: Motivation of VM
Issues with memory from a historical view

- Ultimate solution: Virtual Memory with Page Table



What issues **can** be solved? security
What issues **cannot** be solved?

Key concept 2: Page Table
Game: quick ask and quick answer

- Q1: Where is page table located at?
  A. Register file B. Cache C. Main memory D. Disk
- Q2: Who possesses a page table?
  A. A program B. A process 进程 C. The cache D. The CPU
- Q3: What is the typical size of one page?
  A. 16 Byte B. 4 KB C. 512 KB D. 4 MB
- Q4: Usually what kind of associativity is used?
  A. Directed map B. 2-way C. 4-way D. Full
- Q5: When a page fault occurs, where is the possible destination to find the page?
  A. Swap space B. OS kernel C. Main memory D. TLB

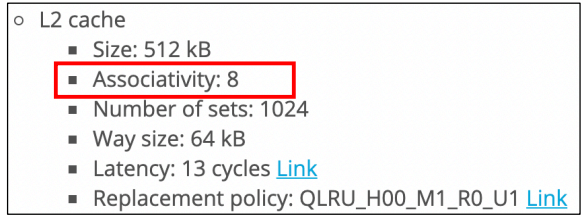
disk.

Key concept 2: Page Table
Game: quick ask and quick answer

- Q1: Where is page table located at?
  A. Register file B. Cache C. Main memory D. Disk
- Q2: Who possesses a page table?
  A. A program B. A process C. The cache D. The CPU
- Q3: What is the typical size of one page?
  A. 16 Byte B. 4 KB C. 512 KB D. 4 MB
- Q4: Usually what kind of associativity is used?
  A. Directed map B. 2-way C. 4-way D. Full
- Q5: When a page fault occurs, where is the possible destination to find the page?
  A. Swap space B. OS kernel C. Main memory D. TLB

Key concept 2: Page Table
Beyond these "simple" questions …

- Q4: Usually what kind of associativity is used? D. Full (associativity)
- Compare with CPU cache (e.g., L2 cache of Intel Core i5-1035G1, shown in the following figure).



use full asso in page table.

- Why not use full associativity in cache?
  - Reason: Must search all entries to find a hit – Reduce performance
- Why do we use full associativity in page table?
  - Goal: Reduce page fault rate × penalty time (very large)

(miss rate).

Key concept 2: Page Table
Beyond these "simple" questions …

- Q3: What is the typical size of one page? B. 4 KB
- But … why 4 KB? How about a larger size? (Homework 8.3.3)

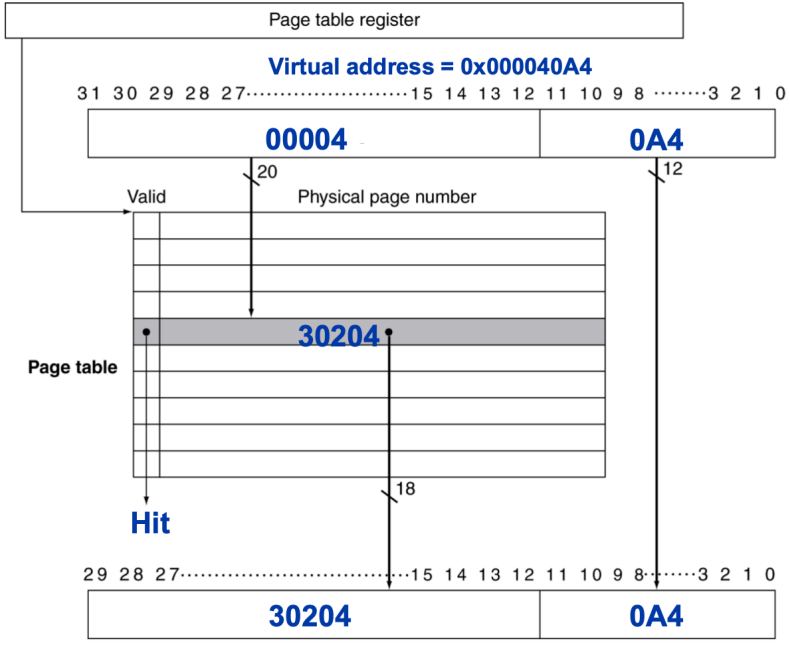
- Why we need to make it large?
  - T14 – Page 18

Most of the time is for getting the first word in the page – access time – very long
- Should have **large page size**, so one access fetches more data, also reduces page fault rate

- Why not too large?

?
- Internal fragmentation (Not all memory in page is used)
- Larger page fault penalty (more time to read from disk)

Key concept 2: Page Table Address translation



Key concept 2: Page Table
Class exercise 1 – Simulate a page table

- Page table: 4KB page size, LRU replacement, stored in the first page in physical memory.
- 16KB physical memory (14-bit physical address)
- 1MB virtual memory (20-bit virtual address)

- How many entries are there in this page table? 1MB / 4kB = 256.
- Suppose initially we have the page table on the next slide, simulate the page table when we access the following virtual addresses.

```
0x00F0C, 0x01F0C (w), 0x02F0C,
0x00100, 0xFF000 (w), 0x01FFF
```

Key concept 2: Page Table
Class exercise 1 – Simulate a page table

```
0x00F0C, 0x01F0C (w), 0x02F0C,
0x00100, 0xFF000 (w), 0x01FFF
```

Page Table

# (*)	V	D	Physical page number
0	1	0	1
1	1	0	2
2	0	–	Disk
...	0	–	Disk
255	0	–	Disk

Physical Memory

# (*)	Content
0	Page table
1	...
2	...
3	...

Note (*): Index field does not exist in the real page table and physical memory.

Key concept 2: Page Table
Class exercise 1 – Simulate a page table

```
0x00F0C, 0x01F0C (w), 0x02F0C,
0x00100, 0xFF000 (w), 0x01FFF
```

Page Table

#	V	D	Physical page number
0	1	0	1
1	1	0	2
2	0	–	Disk
...	0	–	Disk
255	0	–	Disk

Physical Memory

#	Content
0	Page table
1	...
2	...
3	...

Physical address: **0x1F0C**

Key concept 2: Page Table
Class exercise 1 – Simulate a page table

```
0x00F0C, 0x01F0C (w), 0x02F0C,
0x00100, 0xFF000 (w), 0x01FFF
```

Page Table

#	V	D	Physical page number
0	1	0	1
1	1	1	2
2	0	–	Disk
...	0	–	Disk
255	0	–	Disk

Physical Memory

#	Content
0	Page table
1	...
2	...
3	...

Physical address: **0x2F0C**

Key concept 2: Page Table
Class exercise 1 – Simulate a page table

```
0x00F0C, 0x01F0C (w), 0x02F0C,
0x00100, 0xFF000 (w), 0x01FFF
```

Page Fault

Page Table

#	V	D	Physical page number
0	1	0	1
1	1	1	2
2	1	0	3
...	0	-	Disk
255	0	-	Disk

Physical Memory

#	Content
0	Page table
1	...
2	...
3	... VP 2.

Physical address: **0x3F0C**

Key concept 2: Page Table
Class exercise 1 – Simulate a page table

```
0x00F0C, 0x01F0C (w), 0x02F0C,
0x00100, 0xFF000 (w), 0x01FFF
```

Page Table

#	V	D	Physical page number
0	1	0	1
1	1	1	2
2	1	0	3
...	0	-	Disk
255	0	-	Disk

Physical Memory

#	Content
0	Page table
1	...
2	...
3	...

Physical address: **0x1100**

Key concept 2: Page Table
Class exercise 1 – Simulate a page table

```
0x00F0C, 0x01F0C (w), 0x02F0C,
0x00100, 0xFF000 (w), 0x01FFF
```

Page Fault

Page Table

#	V	D	Physical page number
0	1	0	1
1	0	0	Disk
2	1	0	3
...	0	0	Disk
255	1	1	2

Physical Memory

#	Content
0	Page table
1	...
2	... VP 1 -> VPFF
3	...

Physical address: 0x2000

Key concept 2: Page Table
Class exercise 1 – Simulate a page table

```
0x00F0C, 0x01F0C (w), 0x02F0C,
0x00100, 0xFF000 (w), 0x01FFF
```

Page Fault

Page Table

#	V	D	Physical page number
0	1	0	1
1	1	0	3
2	0	0	Disk
...	0	0	Disk
255	0	1	2

Physical Memory

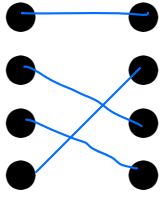
#	Content
0	Page table
1	...
2	...
3	... VP2 → VP1.

Physical address: **0x3FFF**

Key concept 3: TLB
Game: quick ask and quick answer

- Full name of TLB? Translation look-aside buffer

- Matching:
  - L1 cache
  - L2 cache
  - Main memory
  - TLB

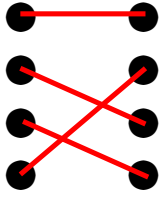


Cache of cache
Cache of page table
Cache of main memory
Cache of disk

locate in CPU (part of cache).

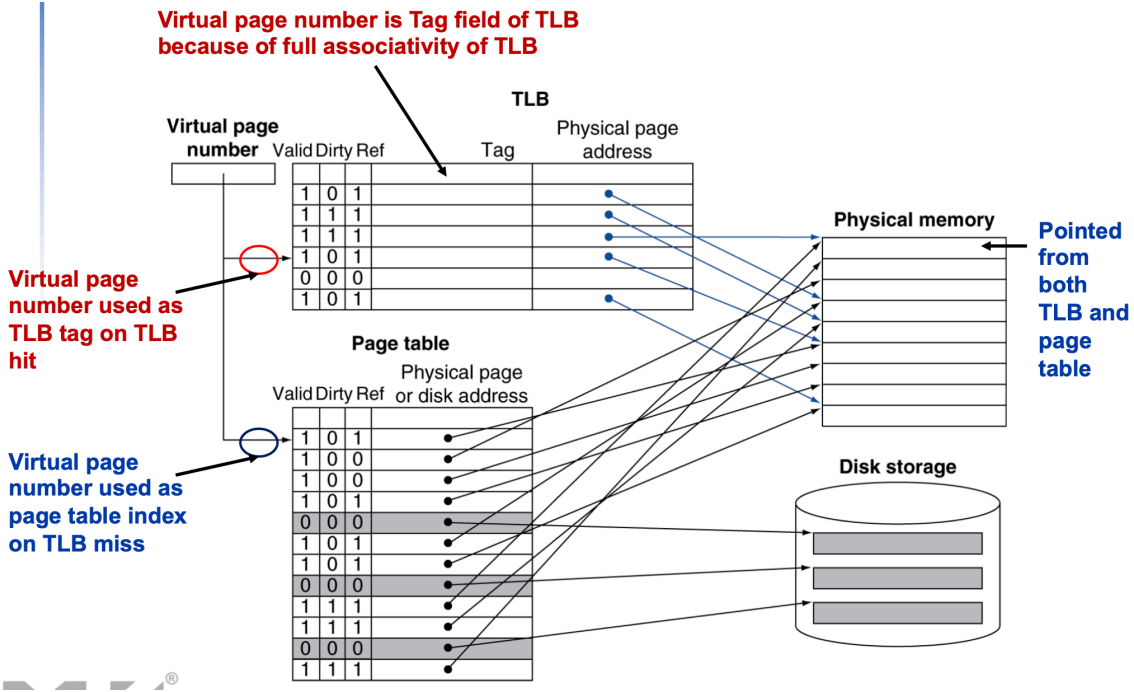
Key concept 3: TLB
Game: quick ask and quick answer

- Full name of TLB?
  - Translation look-aside buffer
- Matching:
  - L1 cache
  - L2 cache
  - Main memory
  - TLB



Cache of cache
Cache of page table
Cache of main memory
Cache of disk

Key concept 3: TLB
How to use TLB? (T14 – Page 24)



Key concept 3: TLB
TLB Hit & Miss

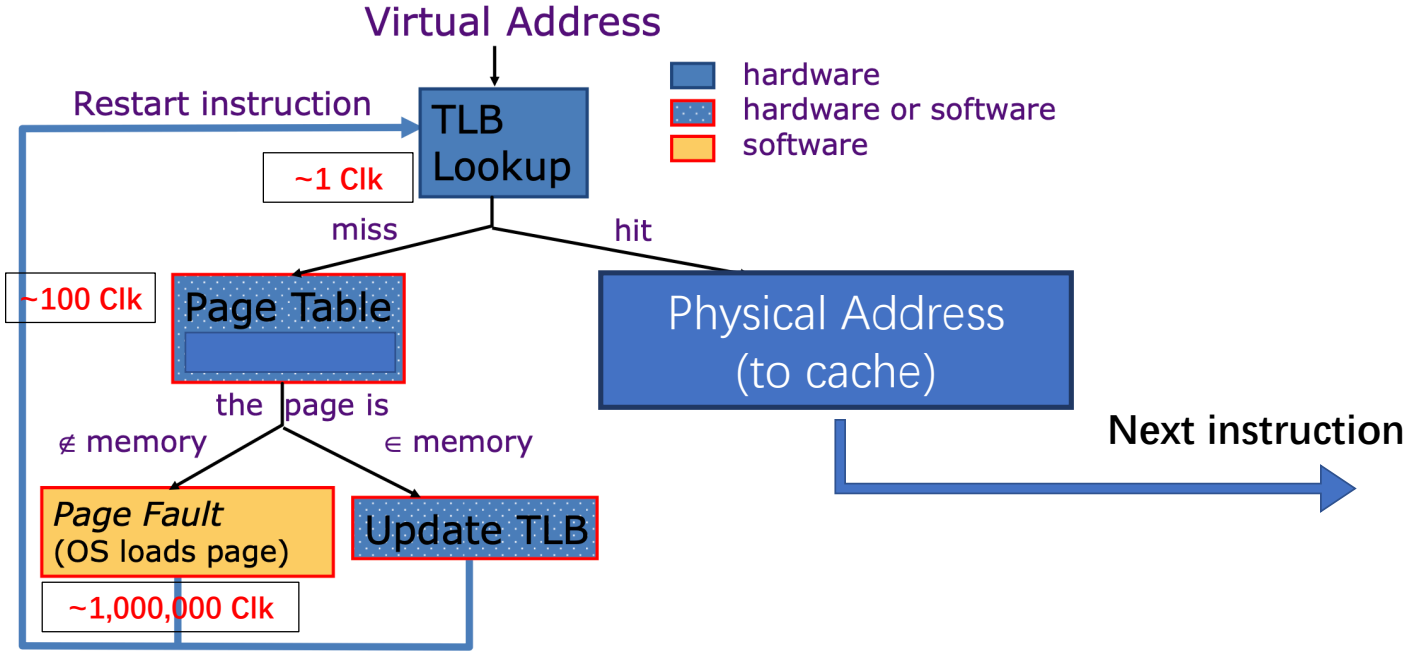
TLB Hit

- Provide physical address
- Reference bit 0/1→1
- If write:
  - dirty bit 0/1→1
- Go to next instruction

TLB Miss

- Page table hit:
  - Load page table entry to TLB
- Page table miss:
  - Page fault exception
  - OS loads the page from disk
  - Update page table & TLB
- Restart current instruction

Key concept 3: TLB
TLB Hit & Miss



Key concept 3: TLB
Class exercise 2 – Homework 8.3.1

- 4 KB pages / 4-entry fully associative TLB / LRU replacement.
- If pages must be brought in from disk, increment to the next largest page number.
- Access to the memory by the following virtual address (decimal): 12948, 49419, 46814, 13975, 40004, 12707, 52236
- Convert to hexadecimal
- `0x3294, 0xC10B, 0xB6DE, 0x3697,`
- `0x9C44, 0x31A3, 0xCC0C`

V	Tag	PPN
1	11	12
1	7	4
1	3	6
0	4	9

TLB

#	V	PPN
0	1	5
1	0	Disk
2	0	Disk
3	1	6
4	1	9
5	1	1
6	0	Disk
7	1	4
8	0	Disk
9	0	Disk
10	1	3
11	1	12

Page Table

Key concept 3: TLB
Class exercise 2 – Homework 8.3.1

- 4 KB pages / 4-entry fully associative TLB / LRU replacement.
- `0x3294, 0xC10B, 0xB6DE, 0x3697, 0x9C44, 0x31A3, 0xCC0C`

- Physical address: `0x6294`

TLB Hit

V	Tag	PPN
1	11	12
1	7	4
1	3	6
0	4	9

TLB

#	V	PPN
0	1	5
1	0	Disk
2	0	Disk
3	1	6
4	1	9
5	1	1
6	0	Disk
7	1	4
8	0	Disk
9	0	Disk
10	1	3
11	1	12

Page Table

Key concept 3: TLB
Class exercise 2 – Homework 8.3.1

- 4 KB pages / 4-entry fully associative TLB / LRU replacement.
- `0x3294, 0xC10B, 0xB6DE, 0x3697, 0x9C44, 0x31A3, 0xCC0C`

- Physical address: `0xD10B`

TLB miss

Page Fault

V	Tag	PPN
1	11	12
1	7	4
1	3	6
1	12	13

TLB

#	V	PPN
0	1	5
1	0	Disk
2	0	Disk
3	1	6
4	1	9
5	1	1
6	0	Disk
7	1	4
8	0	Disk
9	0	Disk
10	1	3
11	1	12
12	1	13

从disk 给入

Page Table

Key concept 3: TLB
Class exercise 2 – Homework 8.3.1

- 4 KB pages / 4-entry fully associative TLB / LRU replacement.
- `0x3294, 0xC10B, 0xB6DE, 0x3697, 0x9C44, 0x31A3, 0xCC0C`

- Physical address: `0xC6DE`

TLB Hit

V	Tag	PPN
1	11	12
1	7	4
1	3	6
1	12	13

TLB

#	V	PPN
0	1	5
1	0	Disk
2	0	Disk
3	1	6
4	1	9
5	1	1
6	0	Disk
7	1	4
8	0	Disk
9	0	Disk
10	1	3
11	1	12
12	1	13

Page Table

Key concept 3: TLB
Class exercise 2 – Homework 8.3.1

- 4 KB pages / 4-entry fully associative TLB / LRU replacement.
- `0x3294, 0xC10B, 0xB6DE, 0x3697, 0x9C44, 0x31A3, 0xCC0C`

- Physical address: `0x6697`

TLB Hit

V	Tag	PPN
1	11	12
1	7	4
1	3	6
1	12	13

TLB

#	V	PPN
0	1	5
1	0	Disk
2	0	Disk
3	1	6
4	1	9
5	1	1
6	0	Disk
7	1	4
8	0	Disk
9	0	Disk
10	1	3
11	1	12
12	1	13

Page Table

Key concept 3: TLB
Class exercise 2 – Homework 8.3.1

- 4 KB pages / 4-entry fully associative TLB / LRU replacement.
- `0x3294, 0xC10B, 0xB6DE, 0x3697, 0x9C44, 0x31A3, 0xCC0C`

- Physical address: **`0xEC44`**

0x31A3 → 0x61A3
0xCC0C → 0xDC0C.

TLB miss

Page Fault

V	Tag	PPN
1	11	12
1	9	14
1	3	6
1	12	13

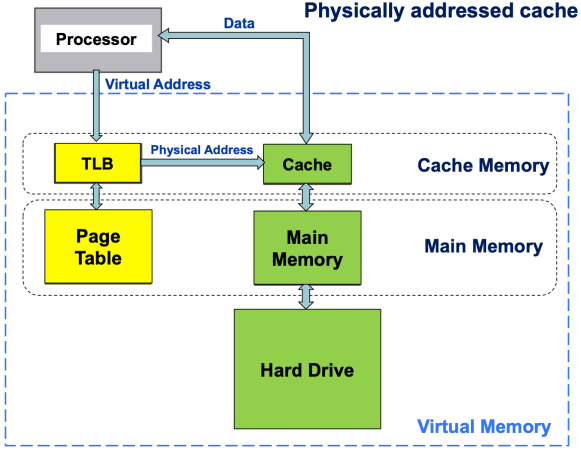
TLB

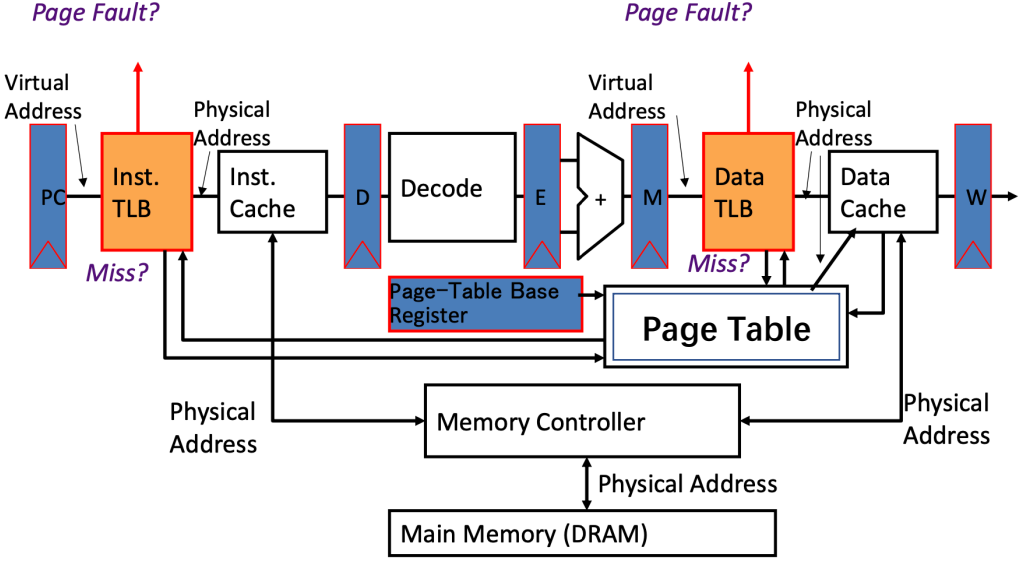
#	V	PPN
0	1	5
1	0	Disk
2	0	Disk
3	1	6
4	1	9
5	1	1
6	0	Disk
7	1	4
8	0	Disk
9	1	14
10	1	3
11	1	12
12	1	13

Page Table

The figure on the right will not be covered in the exam.

Key concept 3: TLB
The big picture (Design #1)



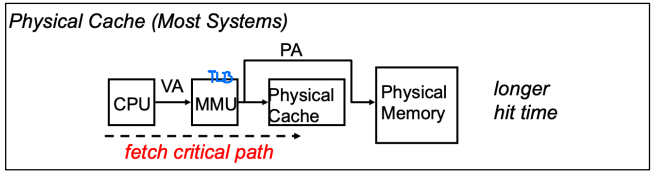


The figures on the left will not be covered in the exam.

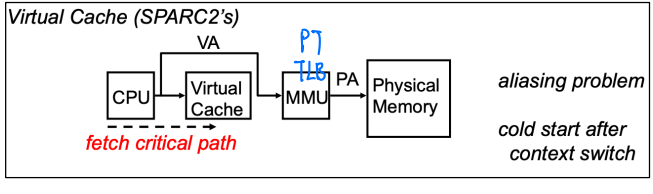
Key concept 3: TLB
The big picture (Design #2)

TLB in CPU.

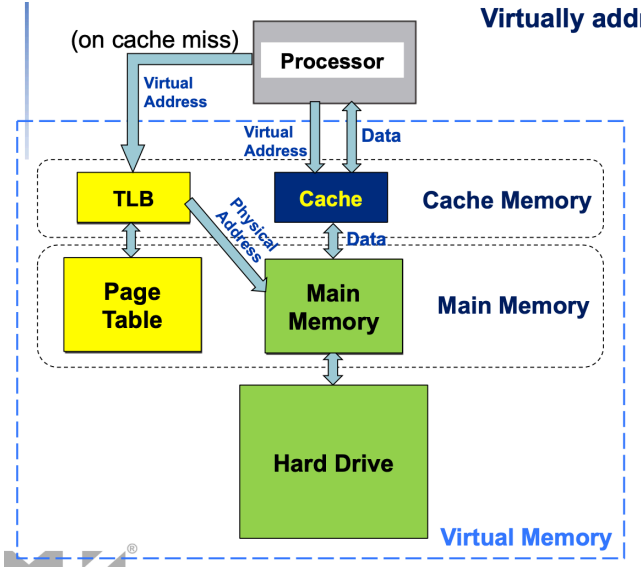
Get physical address in Design #1



Get physical address in Design #2



The big picture of design #2

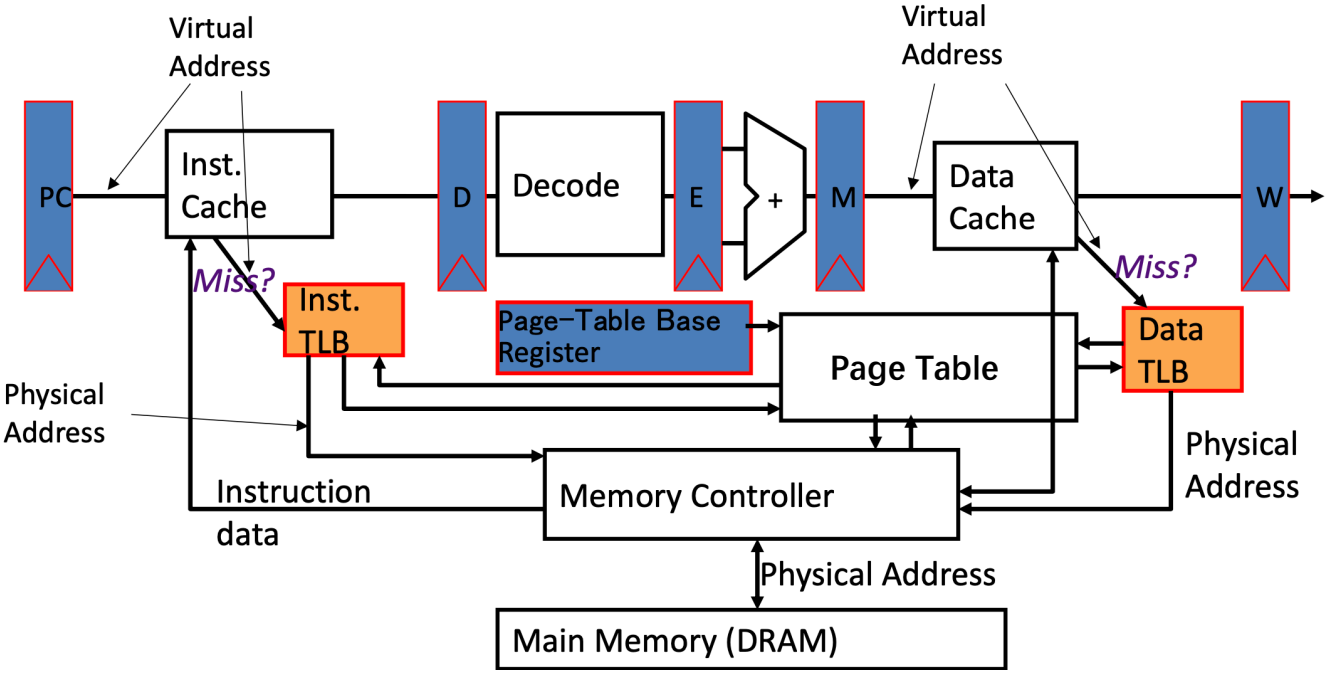


Virtually addressed cache

- No need for translation
- Translate when cache miss
- Complications due to aliasing
  - When different virtual spaces map to the same physical main memory location

This slide will not be covered in the exam.

Key concept 3: TLB
The big picture (Design #2)



Thank you!

Reference

- VE370 Lecture slides (T14 Virtual Memory).
- Princeton ELE475 Lecture slides (L10 Address Translation and Protection).
- UM EECS470 Lecture slides (L20 Virtual Memory).
- Caches. Accessed: 2020-11-12. https://uops.info/cache.html