What does a fully associative cache store?

1

How does the CPU locate an item in a fully associative cache?

2

What is temporal locality?

3

What is spatial locality?

4

What are the three common cache replacement algorithms?

5

Explain the write-through cache write strategy.

6

Explain the copy-back cache write strategy.

7

Why does a direct mapped cache usually use static RAM?

8

Addresses and their corresponding data

1

Hardware compares the input address with all stored addresses (in parallel)
If we get a match we have a hit
If no match we must go to main memory

2

The principle that if you use an address once, you may use it again soon e.g. loops

3

The principle that if you use an address once, you are also likely to use addresses nearby e.g. arrays

4

Least Recently Used (LRU)
Round Robin
Random

5

Whenever a write is done to the cache, the write is also done to main memory

6

When a cache line is replaced, if the dirty bit is set, the modified value is written to main memory

7

It is a lot faster than dynamic RAM

8

How many transistors do DRAM and SRAM use per bit?

9

Briefly explain what a set associative cache consists of.

10

What is the advantage of using a set associative cache?

11

What two control bits are usually used in cache entries?

12

Explain what a compulsory cache miss is?

13

Explain what a capacity cache miss is?

14

Explain what a conflict cache miss is?

15

What are the two types of virtualisation?

16

A number of directly mapped caches operating in parallel

10

1 and 6.

9

Valid bit and dirty bit

12

We have more flexible cache replacement strategies as we could choose any one of the caches to replace from

11

Since the cache is limited in size, we cant contain all of the pages for a program, so if an address is evicted from the cache to make more space, but is then requested after, then it will be a capacity miss.

14

When we first start the computer, the cache is empty, so until the cache is populated, we're going to have a lot of misses (or whenever the cache hasn't seen an address before).

13

System virtualisation (run whole OS inside software e.g. VMware) and process virtualisation (run a process under a control layer of software, e.g. JVM).

16

In a direct mapped or set associative cache, there is competition between memory locations for places in the cache. If the cache was fully associative, then misses due to this wouldnt occur.

15

What are the three main advantages of virtualisation?

17

A hypervisor runs in [redacted] *mode, and can run virtual machines in a* [redacted] *mode, having* [redacted] *for when the guest OS does something that requires* [redacted].

18

What happens when you start a VM?

19

When is it best to stop a VM?

20

What is retained when a VM is stopped/paused?

21

What operations can we do on a VM?

22

What are the two phases in live migration?

23

What are the three main categories of permanent storage media?

24

A hypervisor runs in privileged mode, and can run virtual machines in a unprivileged mode, having traps for when the guest OS does something that requires system privileges.

18

Translation (between instruction sets, system API's etc), abstraction (providing garbage collection, debugging etc), or multiplexing (e.g. RAID or emulating CD drives).

17

When the VM's IO is quiescent (i.e. not doing anything).

20

- *Save the current registers*
- *Load the VM registers*
- *Move the PC to the start address of the VM*

19

- *Move VM's between machines (live migration)*
- *Take a snapshot of a VM*
- *Restore a VM from a snapshot (quickly)*
- *Load balancing using live migration*

22

Memory, IO state, CPU registers, open files, network connections etc

21

Write Once Read Many, Write Many Read Many, Write (not too) Many Read Many

24

The warm up phase and the stop and copy phase.

23

What are the three terms used to quantify hard drive performance characteristics?

25

What is the seek time?

26

What is the search time?

27

What is the transfer rate?

28

Give the equation for disk access time.

29

How can we work out the search time from the RPM?

30

What happens if the OS wants to read a file that is split over multiple sectors on the hard drive?

31

What does RAID stand for?

32

The time it takes for the head to reach the target track on the platter.

26

Seek time, search time and transfer rate.

25

The amount of data that can be read/written per unit time.

28

The time it takes for the target sector to arrive under the head.

27

$$\textit{Search time (seconds)} = \frac{0.5 \textit{ rotations} \times 60}{RPM}$$

30

$$\textit{Disk access time} = \textit{Seek time} + \textit{Search time} + \textit{Transfer time}$$

29

Redundant Array of Independent Disks

32

The processor in the hard drive changes the read order to the most efficient order of sectors for the head to minimise wasted rotations.

31

RAID0 has a [redacted] transfer rate and a [redacted] redundancy because data is [redacted] over multiple disks.

33

RAID1 has a [redacted] transfer rate and a [redacted] redundancy since data is [redacted] over multiple disks.

34

We can use [redacted] such as [redacted] and [redacted] to provide redundancy using RAID.

35

Define RAID2,3,4,5,6.

36

What is the transistor used in SSD's?

37

What is wear-levelling?

38

RAID lets us break the concept of [redacted] by mapping them onto [redacted] using [redacted]. The actual details of storage are [redacted].

39

What is a SAN?

40

RAID1 has a low transfer rate and a high redundancy since data is mirrored over multiple disks.

34

RAID0 has a high transfer rate and a low redundancy because data is striped over multiple disks.

33

2 *Bit striping and hamming codes*
3 *Byte striping and parity bits*
4 *Block striping and parity*
5 *Block striping and distributed parity*
6 *Double distributed parity*

36

We can use error correction such as hamming codes and parity bits to provide redundancy using RAID.

35

When a SSD maps different logical addresses to different physical blocks so that all blocks are worn out at the same rate.

38

A Floating Gate Field Effect Transistor.

37

A Storage Area Network.

40

RAID lets us break the concept of file systems by mapping them onto multiple drives using striping and mirroring. The actual details of storage are abstracted away.

39

[redacted] is a volume aware filesystem. It protects against losing files, running out of space, corruption of data etc by being very flexible and having lots of [redacted] and implementing [redacted]-[redacted]-[redacted], simple [redacted], [redacted], self [redacted] and [redacted], sumchecking and more.

41

Explain each stage of a 5 stage pipeline

42

Briefly explain what pipelining is

43

What is a control hazard?

44

What are two ways of dealing with control hazards?

45

Briefly explain what branch prediction is

46

What is used to implement branch prediction and what does it do?

47

What is a data hazard?

48

IF - Fetch instruction from memory
ID - Decode instruction; select registers
EX - Perform an operation or calculate an address
MEM - Access an operand in memory
WB - Write to registers

42

ZFS is a volume aware filesystem. It protects against losing files, running out of space, corruption of data etc by being very flexible and having lots of ECC and implementing copy-on-write, simple rollback and recovery, wear leveling, self checking and healing, sumchecking and more.

41

If we have a branch at the ID stage, then the fetched instruction at the IF stage will have to be ignored all the way down the pipeline, wasting one full cycle and causing a bubble.

44

Where we get all the components of the CPU working at the same time, with buffers that are flushed every clock cycle inbetween each stage, so that we can overlap the execution of instructions to increase overall clock speed.

43

If we can remember what address a branch directed us to fetch next from what it did when we executed that branch previously, then we can pre-emptively load that instruction in the IF stage instead of fetching the instruction at the PC.

46

Pipeline bubbling (abort instructions that are incorrect) and branch prediction (guess what way to branch).

45

This is where we execute instructions that depend on each other in parallel or close together and the correct data might not be in the right place (e.g. registers).

48

A branch target buffer which maps the virtual address of one branch instruction onto the virtual address of the instruction that is branched to

47

What is forwarding?

49

How can we exploit instruction level parallelism?

50

What does VLIW stand for?

51

How can we implement an out of order processor?

52

Fetch multiple instructions per cycle
Have multiple ALU's to execute instructions in parallel (superscalar)
Have common registers and caches, since the instructions are operating on the same data

50

Where we add extra paths to the architecture to pass updated register values back to previous stages of the pipeline to avoid data hazards.

49

Have a buffer that instructions are fetched into
A scheduler to choose which instructions to execute at what times
A cache to store memory and register accesses until all instructions have finished so that the application can execute normally as though instructions were executed in parallel

52

Very Long Instruction Word

51