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

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

Addresses and their corresponding data

1

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

4

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

3

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

6

Least Recently Used (LRU)
Round Robin
Random

5

It is a lot faster than dynamic RAM

8

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

7

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 mode, and can run virtual machines in a mode, having for when the guest OS does something that requires .

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

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

30

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

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 transfer rate and a redundancy because data is over multiple disks.

33

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

34

We can use such as and 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 by mapping them onto using . The actual details of storage are .

39

What is a SAN?

40

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

33

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

34

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

35

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

A Floating Gate Field Effect Transistor.

37

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

38

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

A Storage Area Network.

40

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 and implementing - - , simple , , self and , 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

A data dependency between instructions. 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

How can we mitigate Data Hazards effects?

49

What is forwarding?

50

How can we exploit instruction level parallelism?

51

What does VLIW stand for?

52

How can we implement an out of order processor?

53

For pipelines with multiple execution flows, each flow is known as a .

54

: long wait before finishing execution.

55

: the required resource (i.e., Functional Unit) is not available.

56

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

50

Extra lines in the data path (Forwarding). Adding NOPs. Reordering instructions.

49

Very Long Instruction Word

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

51

For pipelines with multiple execution flows, each flow is known as a Functional Unit.

54

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

53

Structural Hazard: the required resource (i.e., Functional Unit) is not available.

56

Cache misses: long wait before finishing execution.

55

What is scoreboarding?

57

[redacted] *(read-after-write): Instruction A depends on the output of a previous instruction B.*

58

[redacted] *(write-after-read): Instruction A writes in the input of a previous instruction B. We need to ensure B reads the correct value instead of that generated by A.*

59

[redacted] *(write-after-write): Instruction A and B write in the same register. We need to ensure that the register keeps the value of the later instruction.*

60

[redacted]*: Decode instructions, check for structural hazards. Checks for WAW.*
[redacted]*: Wait until no data hazards, then read operands. Checks for RAW*
[redacted]*: Operate on operands.*
[redacted]*: Finish execution and write results. Checks for WAR.*

The four stages of the Scoreboard pipeline

61

[redacted]*: implicit register renaming by buffering source operands.*

62

In the context of Tomasulo, what is register renaming?

63

What's a ***Common Data Bus*** *and how does it differ to a normal one?*

64

True dependency (read-after-write): Instruction A depends on the output of a previous instruction B.

58

Scoreboarding is a method for dynamically scheduling a pipeline so that the instructions can execute out of order when there are no conflicts and the hardware is available.
In a scoreboard, the data dependencies of every instruction are logged. Instructions are released only when the scoreboard determines that there are no conflicts with previously issued and incomplete instructions.

57

Output Dependency (write-after-write): Instruction A and B write in the same register. We need to ensure that the register keeps the value of the later instruction.

60

Anti-dependency (write-after-read): Instruction A writes in the input of a previous instruction B. We need to ensure B reads the correct value instead of that generated by A.

59

Reservation stations: implicit register renaming by buffering source operands.

62

Issue: Decode instructions, check for structural hazards. Checks for WAW.
Read operands: Wait until no data hazards, then read operands. Checks for RAW
Execution: Operate on operands.
Write Result: Finish execution and write results. Checks for WAR.

The four stages of the Scoreboard pipeline

61

64 bits of data + 4 bits of Functional Unit address.
Functional units broadcast their result.
Reservation stations take the operand if it matches any input Functional Unit.
Register bank takes the operand if it matches the Functional Unit writing the result.

64

The concept of renaming a register to eliminate WAR & WAW hazards.
*Can be done by a compiler, but Tomasulo does it transparently in hardware (by the **Reservation Stations**).*

63

The first stage of the Tomasulo algorithm is .
It gets the instruction from the FP Op Queue.
If the reservation station is free (i.e. no), issue instruction and read operands.
Otherwise, .

65

The second stage of the Tomasulo algorithm is .
When both source operands are ready then execute; if not ready, watch for results.

66

The third (and final) stage of the Tomasulo algorithm is .
Write on Common Data Bus to all ; free .

67

What are the benefits of Out of Order Processors?

68

What are the limitations of Out of Order Processors?

69

Name some stuff that the operating system must load and store on a context switch.

70

How is a multi-threaded processor most commonly presented to the operating system?

71

What are the three types of hardware multithreading?

72

The first stage of the Tomasulo algorithm is Issue.
It gets the instruction from the FP Op Queue.
If the reservation station is free (i.e. no structural hazard), issue instruction and read operands.
Otherwise, stall the pipeline.

65

The second stage of the Tomasulo algorithm is Execute.
When both source operands are ready then execute; if not ready, watch Common Data bus for results.

66

The third (and final) stage of the Tomasulo algorithm is Write back.
Write on Common Data Bus to all awaiting units; free reservation station.

67

Accelerates the execution of programs.
More efficient design by increasing the utilisation of processor resources.

68

More complex design.
Expensive in terms of area power.
Non precise interrupts.

69

- *Process ID*
- *Program Counter*
- *Stack Pointer*
- *General registers*
- *Memory management information*
- *Open file list (and positions)*
- *Network connections*

70

As a processor with multiple cores (even though only one multithreaded core may be in the processor).

71

- *Coarse grain*
- *Fine grain*
- *Simultaneous MultiThreading (SMT)*

72

Briefly describe coarse grain multithreading.

73

What extras does coarse grain multithreading require from the processor?

74

What can a context switch do to the cache?

75

Briefly describe fine grained multithreading.

76

Briefly describe SMT.

77

What are the motivations that are driving us towards multi core systems?

78

What do different cores on a processor ***not*** *share?*

79

What is (my definition^{TM} of) consistency?

80

The processor switches threads (a context switch) whenever an expensive operation is started (such as a memory load).

73

You don't need to change much in the processor, just make it abort instructions after a cache miss and have it store (and later load) the state of the thread.

74

A context switch can trash the cache since the new thread may access different regions of memory and therefore all the memory reads will be misses. New values will be loaded into the cache which will destroy its previous data.

75

This involves interleaving the instructions of several threads. When memory is accessed, instructions from other threads will be executed to ensure stalls are brief. The aim is to reduce the cost of switching CPU threads to almost nothing.

76

We have instructions from multiple threads in the pipeline at the same time. This requires significant hardware overhead, but gives you more freedom for instruction scheduling (since instructions in different threads are rarely interdependent so you can interleave them).

77

- *So many transistors per unit area, cooling is a massive issue*
- *Small transistors have unpredictable characteristics*
- *Architecture of processors is becoming too complex to reason about*
- *Exponentially more complex hardware gives sublinear performance gains*
- *Have multiple but more simple cores instead*

78

An L1 cache (split into data and instruction caches) and sometimes an L2 cache. They also have their own registers obviously

79

The programmer's view of the system. For example, they expect that if a memory location is updated in one thread, then the change will be visible across all threads, not just the threads that are running on the core that has the new value in its L1D cache.

80

What are the three special instructions used to guarantee out of order processors maintain consistency?

81

What is transactional memory?

82

What are the two most simple snooping protocols?

83

Describe 'Write update' (the snooping protocol).

84

Describe 'Write invalidate' (the snooping protocol).

85

Why is write invalidate better than write update for things like loops or writes to different words of the same cache line?

86

What does MESI stand for?

87

What is a directory based protocol with reference to multi core systems?

88

- *A **fence** makes sure each memory access before the fence is complete before a new one is started.*
- *A **barrier** makes threads wait until they have all reached the barrier.*
- *A **lock** makes sure that only one thread enters a critical section of the program at a time (atomic access). Requires hardware support.*

81

Memory that supports transactions (yeah, duh). You can read and write to it however you like, but when you're finished, you have to commit, when your transaction is checked for conflicts and rolled back if it does conflict.

82

Write update and write invalidate.

83

When a core writes a value to memory, the value is updated in its L1 cache, the cache then broadcasts the address on the bus, and the snooping caches update their copy.

84

When a core writes a value to memory, the value is updated in its L1 cache, but sends a write invalidate message to the other caches which then invalidate the updated cache line in their copies.

85

If a value is being frequently updated, then write invalidate needs to happen once, but write update needs to happen on every update, which wastes power and can saturate the bus.

86

Modified, Exclusive, Shared, Invalid.

87

A protocol where there is a directory that holds information on what each L1 cache holds. This lets cores talk on a P2P basis rather than all using one bus.

88

What are the concerns about a NoC (Network on a Chip)?

89

Buses are at any one time and are controlled by a that divides its use into . You can in one and in a future one.

90

Name five NoC architectures.

91

Describe the three types of NoC routing.

92

What are the two types of packet switching in NoC's?

93

Buses are single usage at any one time and are controlled by a clock that divides its use into time slots. You can send in one slot and receive in a future one.

90

- *Bandwidth*
- *Latency*
- *Fault tolerance*
- *Area*
- *Power dissipation*

89

Minimal	*Always select the shortest path towards the destination*
Oblivious	*Take a fixed path every time (really simple!)*
Adaptive	*Take the least congested route, complex though and uses lots of power so its rarely used.*

92

- *Crossbar*
- *Ring*
- *Tree*
- *Fat Tree*
- *Mesh*

91

Store and forward (wait for all flits before sending) and wormhole (send flits as soon as the head flit arrives).

93