

Question1:

Describe in detail the classification of I/O interconnects based on the communication distance, bandwidth latency and reliability.

Answer:

The I/O interconnect is the glue that interfaces computer system components. I/O interconnects are facilitated using High speed hardware interfaces and logical protocols. Based on the desired communication distance, bandwidth, latency and reliability, interconnects are classified as used: Backplanes, channels, Networks

	Network	Channel	Backplane
Distance	>1000 m	10 - 100 m	1 m
Bandwidth	10 - 100 Mb/s	40 - 1000 Mb/s	320 - 1000+ MB/s
Latency	high (>ms)	medium	low (<μs)
Reliability	low Extensive CRC	medium Byte Parity	high Byte Parity
	message-based narrow pathways distributed	←————→	memory-mapped wide pathways centralized

Network interconnects are used where distance is very large where is more than 1 km, where are interconnects of type channel are used where the distance is 10-100m, and the backplane interconnect t are used very small distance. Backplane are those which use to connect the backplane of CPU or micro process based computing system with the external device.

Whereas based on the bandwidth with network basically are developed where the bandwidth are the rage of 10-100 MB/sec and the channel are used 40000 MB/sec and the backplanes have the bandwidth 320-1000+ MB.

The latency of the backplane is basically lowest where are the channel it is medium and for networks it is highest.

However the reliability of network is low and it is medium for the channels and it is very high for the backplane. Backplane usually is the byte parity matter to achieve the reliability in the system and it give us very high reliability

Interconnects are usually implements by using the bus system the bus is basically is shared community links in the sub system, it is collection of number of conductors. Where busses can be classified as the data bus the address bus and the control bus.

Question 2:

Suppose we have a processor with base CPI 1.0, assuming all reference hit in the primary (level-1) cache and the clock rate 500 mhz. assume a main memory access time of **20 ns** .including all miss handling. Suppose the miss rate per instruction at the primary cache is 5%. How much faster will the machine be if we add a second level cache that has a 20 ns access time for either a hit or miss and is large enough to reduce the miss rate to main memory to 2 %.

Answer:

$$\begin{aligned}\text{Miss penalty}_{\text{main memory}} &= 200 * 10^{-9} / (1 / (500 * 10^6)) \\ &= 100 \text{ Clock Cycles}\end{aligned}$$

$$\begin{aligned}\text{CPI} &= \text{Base CPI} + \text{Memory stalls per instruction} \\ &= 1.0 + (5/100) * 100 \\ &= 6\end{aligned}$$

$$\begin{aligned}\text{Miss penalty}_{\text{Level 2 cache}} &= 20 * 10^{-9} / (1 / (500 * 10^6)) \\ &= 10 \text{ Clock Cycles}\end{aligned}$$

$$\begin{aligned}\text{CPI}_{\text{with level 2 cache implemented}} &= \text{Base CPI} + \text{Level 1 memory stalls} + \text{Level 2 memory stalls} \\ &= 1 + (5 / 100) * 10 + (2 / 100) * 100 \\ &= 3.5\end{aligned}$$

$$\text{Comparison} = 6 / 3.5 = 1.7$$

Question 3:

Let us assume a computer has a 64 bytes cache block. an L2 cache that takes 7 clock cycle to get the critical 8 bytes, and then 1 clock cycle per 8 bytes +1 extra clock cycle to fetch the rest of the block without critical word first its 8 bytes clock for the first 8 bytes and the 1 clock per 8 bytes for the rest of the block.

Calculate the average penalty for critical word first assuming that there will be no other access to the rest of the block. Compare time with or without critical words first.

Answer:

1. With Critical word first:
Average miss penalty
= Miss Penalty of critical word + Miss penalty of the remaining words of the block
= $7 \times 1 + (8-1) \times 1 = 7 + 7 = 14$ clock cycles
2. Without critical word first (it requires block load)
= [Miss Penalty of first word + miss penalty of the remaining words of the block]
+ clock cycles to issue the load
= $7 \times 1 + (8-1) \times 1 + 8/4 = 14 + 4 = 18$ clock cycles
2 issues/cycle so 4cycles for 8 issues

Solution # 2:

1. With Critical word first:
Average miss penalty
= Miss Penalty of critical word + Miss penalty of the remaining words of the block

$$= 7 \times 1 + 1 \times (8-1) + 1 \times (8-1) = 7 + 7 + 7 = 21 \text{ clock cycles}$$

2. Without critical word first (it requires block load)

= [Miss Penalty of first word + miss penalty of the remaining words of the block]
+ clock cycles to issue the load

$$= [7 \times 1 + 1 \times (8-1) + 1 \times (8-1)] + 8/2 = 14 + 4 = 18 \text{ clock cycles}$$

Check this solution yourself (lecture 29)

Question 4:

The running program pattern is

0x0 0x8 0x10 0x18 0x20 0x28

- a) If you directed mapped cache size 1KB and block size of 8 bytes (2 words) how many set:
- b) With the same cache and block size what is miss rate of the directed mapped.
- c) On which would decrease miss rate the most
 - i. Increasing the degree of associative by 2.
 - ii. Increasing the block size to 16 bytes.

Answer:

(a)

- Cache size = 1KB=1024 Bytes
- Block size = 8 Bytes
- No. of blocks = $1024/8 = 128$
- if we makes sets of 2 blocks then the number of sets = $128/2 = 64$

(b)

- When we access 0x0 it is miss
- Block of 8 bytes (0-7) comes into cache
- When we access 0x8 it is miss because till now it is not in cache
- Block of 8 bytes (8-15) comes into cache
- When we access 0x10(16 binary) it is miss because till now it is not in cache
- Block of 8 bytes (16-23) comes into cache
- When we access 0x18(24 binary) it is miss because till now it is not in cache
- Block of 8 bytes (24-31) comes into cache
- When we access 0x20(32 binary) it is miss because till now it is not in cache
- Block of 8 bytes (32-39) comes into cache
- When we access 0x28(40 binary) it is miss because till now it is not in cache
- Block of 8 bytes (40-47) comes into cache
- Number of access = 6
- Number of misses = 6
- Miss rate = 100 %

(c)

- Increasing the block size to 16 bytes.

Check your own self

Question 5:

The virtual memory system you are designing uses a single-level page table built from dedicated hardware (SRAM and associated logic). It supports 25-bit virtual addresses, 22-bit physical addresses, and 2^{16} -byte (64 KB) pages. Each page table entry contains a physical page number, a valid bit (*V*) and a dirty bit (*D*).

- What is the total size of the page table, in bits?
- The operating system team proposes reducing the page size from 64 to 16 KB, but the hardware engineers on your team object on the grounds of added hardware cost. Explain their objection.
- The page table is to be integrated on the processor chip, along with the on-chip cache. The on-chip cache deals only with physical (not virtual) addresses. Is it possible to access the appropriate set of the on-chip cache concurrently with the page table access for a given memory access? Explain briefly the relationship that is necessary for concurrent access to the cache set and page table entry.
- Is it possible to perform the tag comparison in the on-chip cache concurrently with the page table access for a given memory access? Explain briefly.

Answer:

- Each entry in the page table has 2 status bits (*V* and *D*), and a physical page number ($22 - 16 = 6$ bits). The page table has $2^{25 - 16} = 2^9$ entries. Thus, the total page table size is 29×8 bits = 4096 bits.
- This would increase the virtual page number to $25 - 14 = 11$ bits, and the physical page number to $22 - 14 = 8$ bits. This would increase the page table size to: $2^{11} \times 10$ bits = 20480 bits. This increases the page table by 5 times, wasted valuable hardware to store the extra page table bits.
- Yes, this is possible. In order for concurrent access to take place, the number of set + block offset + byte offset bits must be less than the page offset bits.
- It is impossible to perform the tag comparison in the on-chip cache concurrently with the page table access because the upper (most significant) bits of the physical address are unknown until after the page table lookup (address translation) completes

Question 6:

What is Write Back strategy and Write Buffer Saturation?

Answer:

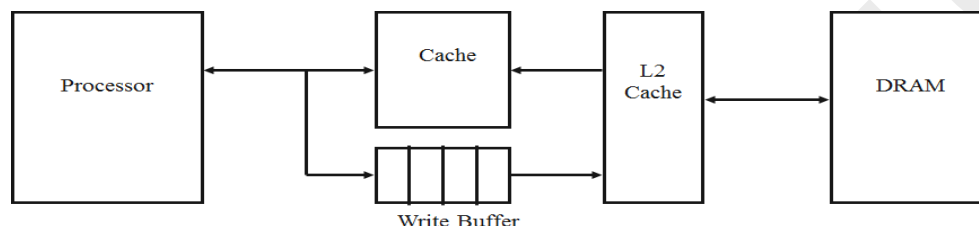
Write back: The information is written only to the block in the cache. The modified cache block is written to main memory only when it is replaced

Pros and Cons

- No write to the lower level for repeated writes to cache
- A dirty bit is commonly used to indicate the status as the cache block is modified (dirty) or not modified (clean)
- Reduce memory-bandwidth requirements, hence the reduces the memory power requirements

Write Buffer Saturation:

- In that case, it does NOT matter how big you make the write buffer, the write buffer will still overflow because you are simply feeding things in it faster than you can empty it
- There are two solutions to this problem:
- The first solution is to get rid of this write buffer and replace this write through cache with a write back cache



Question 7:

Let us consider a fully associative write-back cache with cache entries that start empty. Consider the following sequence of five memory operations and find, which address is not in the cache for no-write allocate.

Write Mem [100]
 Write Mem [100]
 Read Mem [200]
 Write Mem [200]
 Write Mem [100]

Answer:

- For no-write allocate, the address [100] is not in the cache (i.e., its tag is not in the cache)
- So the first two writes will result in MISSES
- Address [200] is also not in the cache, the read is also miss
- The subsequent write [200] is a hit
- The last write [100] is still a miss
- The result is 4 MISSES and 1 HIT

Question 8:

Write down the I/O Performance Parameters.

Answer:

- Diversity: Which I/O device can connect to the CPU
- Capacity: How many I/O devices can connect to the CPU

- Latency: Overall response time to complete a task
- Bandwidth: Number of task completed in specified time - throughput
- The parameters diversity that refers to which I/O device and capacity means how many I/O devices can connect to the CPU are the I/O performance measures having no counterpart in CPU performance metrics.
- In addition, the latency (response time) and bandwidth (throughput) also apply to the I/O system.
- An I/O system is said to be in equilibrium state when the rate at which the I/O requests from CPU arriving, at the input of I/O queue (buffer) equals the rate at which the requests departs the queue after being fulfilled by the I/O device.

Question 9:

Consider a disk subsystem comprising the following component 15 Marks Lecture # 40

- 12 disks, each with MTTF = 1,000,000 Hrs
- 1 SCSI controller with MTTF = 500,000 Hrs
- 1 SCSI cable with MTTF = 1,000,000 Hrs
- 1 power supply with MTTF = 200,000 Hrs
- 1 fan with MTTF = 200,000 Hrs

Find the System Failure Rate and System MTTF.

Answer:

$$\begin{aligned}\text{System Failure Rate} &= 12 (1/1,000,000) + 1/500,000 + 1/1,000,000 + 1/200,000 + 1/200,000 \\ &= 25 / 1,000,000 \text{ Hrs}\end{aligned}$$

$$\text{System MTTF} = 1/\text{Failure Rate} = 1,000,000/25 = 40,000 \text{ Hrs} = 4.56 \text{ years approximately.}$$

Question 10:

Assume a computer has CPI=1.0 when all memory accesses are hit. The only data accesses are load/store access and these are 50% of the total instructions. If the miss rate is 2% and miss penalty is 25 clock cycles, how much faster the computer will be if all instructions are HIT.

Answer:

$$\begin{aligned}\text{CPUtime} &= (\text{CPUTime} + \text{MemoryStalls}) \times \text{ClockCycleTime} \\ &= (IC \times CPI + \text{MemoryStalls}) \times \text{ClockCycleTime}\end{aligned}$$

When all instructions are hit

$$\begin{aligned}\text{CPUtime}_{\text{Ideal}} &= (IC \times CPI + \text{MemoryStalls}) \times \text{ClockCycleTime} \\ &= (IC \times 1.0 + 0) \times \text{ClockCycleTime} \\ &= IC \times \text{ClockCycleTime}\end{aligned}$$

In reality:

$$\begin{aligned}
 \text{MemoryStallCycles} &= IC \times \frac{\text{MemAccess}}{\text{Inst}} \times \text{MissRate} \times \text{MissPenalty} \\
 &= IC \times (1 + 0.5) \times 0.02 \times 25 = IC \times 0.75
 \end{aligned}$$

$$\begin{aligned}
 \text{CPUtime}_{\text{Cache}} &= (IC \times \text{CPI} + \text{MemoryStalls}) \times \text{ClockCycleTime} \\
 &= (IC \times 1.0 + IC \times 0.75) \times \text{ClockCycleTime} \\
 &= 1.75 \times IC \times \text{ClockCycleTime}
 \end{aligned}$$

Question 11:

Let us consider 32KB unified cache with misses per 1000 instruction equals 43.3 and instruction/data split caches each of 16KB with instruction cache misses per 1000 as 3.82 and data cache as 40.9.

Answer:

Execution Time for all Hit = $IC \times 1.0 \times \text{cycle time}$

CPU Execution time with real cache = CPU Execution time + Memory Stall time

$$\begin{aligned}
 \text{Memory Stall Cycles} &= \\
 &= IC \times (\text{Instruction access} + \text{data access}) \text{ per instruction} \times \text{miss rate} \times \text{miss penalty} \\
 &= IC (1 + 0.5) \times 0.02 \times 25 \\
 &= IC \times 0.75
 \end{aligned}$$

$$\begin{aligned}
 \text{CPU Execution time (with cache)} &= \\
 &= (IC \times 1.0 + IC \times 0.75) \times \text{clock time} \\
 &= 1.75 \times IC \times \text{Cycle time}
 \end{aligned}$$

Computer with no cache misses is 1.75 times faster

Question15: (Similar Question in Exam)

Statement: Let us consider 32KB unified cache with misses per 1000 instruction equals 43.3 and instruction/data split caches each of 16KB with instruction cache misses per 1000 as 3.82 and data cache as 40.9; Assume that

- 36% of the instructions are data transfer instructions;
 - 74% of memory references are instruction references; and
 - hit takes 1 clock cycle where the miss penalty is 100 cycles and
 - a load or store takes one extra cycle on unified cache
-
- Assuming write-through caches with write-buffer and ignore stalls due to write buffer – Find the average memory access time in each case
 - Note to solve this problem we first find the miss rate and then average memory access time

Solution:

1. Miss Rate = (Misses/1000) / (Accesses/ inst.)
 - ✓ Miss Rate 16KB Inst = $(3.82/1000) / 1.0 = 0.0038$
 - ✓ Miss Rate 16KB data = $(40.9/1000) / 0.36 = 0.114$
 - ✓ As about 74% of the memory access are instructions therefore overall miss rate for split caches = $(74\% \times 0.0038) + (26\% \times 0.114) = 0.0324$
 - ✓ Miss Rate 32KB unified = $(43.3/1000) / (1+0.36) = 0.0318$
 - ✓ i.e., the unified cache has slightly lower miss rate

2. Average Memory Access Time =

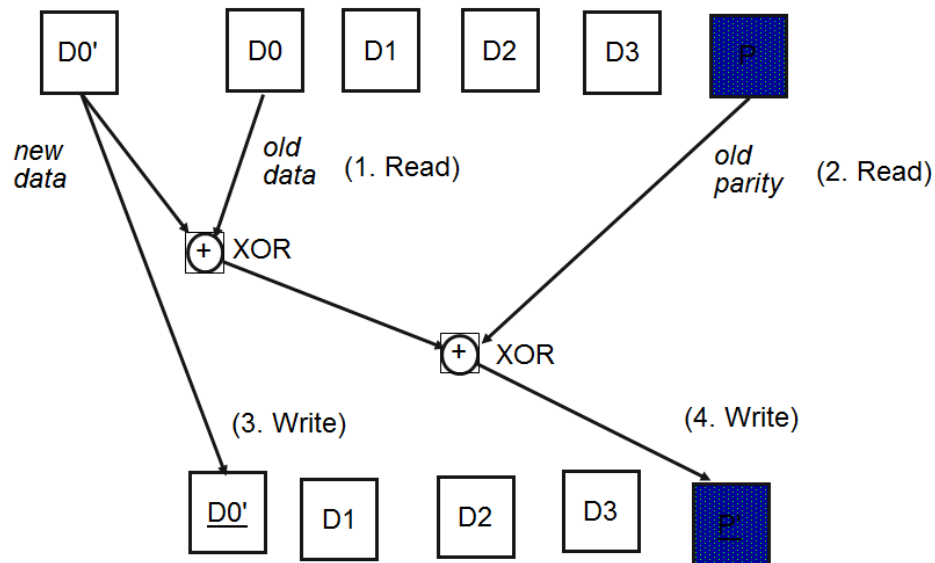
$$\begin{aligned} & \%inst \times (\text{Hit time} + \text{Inst. Miss rate} \times \text{miss penalty}) \\ & + \\ & \%data \times (\text{Hit time} + \text{data Miss rate} \times \text{miss penalty}) \end{aligned}$$
 - ✓ Average Memory Access Time split
 $= 74\% \times (1 + 0.0038 \times 100) + 26\% \times (1 + 0.114 \times 100) = 4.24$
 - ✓ Average Memory Access Time unified
 $= 74\% \times (1 + 0.0318 \times 100) + 26\% \times (1 + 1 + 0.0318 \times 100) = 4.44$
 - ✓ i.e., the split caches have slightly better average access time and also avoids Structural Hazards

Question 16:

Small writes in RAID3 & RAID 4/5

Answer:

- The parity calculation reads blocks D1, D2, and D3 before adding Block D0' to calculate the new parity P'
- Note that here the new data D0 comes directly from CPU, so disk are not involved in reading it
- The small writes in case of RAID 4/5 are as shown here
- Here, the old value of D0 is read (1: Read) and compared with new value D0' to see Which bit will change



- Once it has been checked, then the old parity P is read and corresponding bits are changed to form P'
- This is accomplished by the logical EX-ORs
- In this example, the 3 disk reads (D1, D2, D3) and 2 disk writes (D0' and P') involving all the disks, are replaced with the 2 disk reads (D0, P) and 2 disk writes (D0', P'), each involving just 2 disks
- Hence we can say that one (1) Logical Write in RAID 4 and RAID 5 is equivalent to 2 Physical Reads and 2 Physical Writes

Question 17:

Define Reliability, availability and dependability .

Answer:

- The performance of storage I/Os is measured in terms of its reliability, availability and dependability

Dependability

- Laprie defined dependability as the quality of delivered service such that reliance can justifiably be placed on this service;
- Where the service delivered by a system is its observed actual behavior and the system failure occurs when actual behavior deviates from the specified behavior
- Note that a user perceives a system alternating between two states of delivered service; these states are:
 - ✓ Service Accomplishment – service is delivered as specified and
 - ✓ Service Interruption – delivered service is different from the specified service
- Quantifying the transitions between service accomplishment and service interruption is the measure of the dependability

- The dependability is measured in terms of the measure of:
 - ✓ module reliability, which is the measure of the continuous service accomplishment;
 - ✓ and, module availability, which is the measure of the swinging between the accomplishment and interruption states of delivered service

Measuring Reliability

- Now before we discuss the reliable and dependable designs of the storage I/O let us understand the terminologies used to measure reliability, availability and dependability
- The reliability of a module is the measure of the time to failure from a reference initial instant
- In other words we can say the Mean Time To Failure (MTTF) of a storage module, a disk, is the measure of reliability; and
- The reciprocal of the MTTF is the rate of failure; and
- The service interruption is measured as the Mean Time To Repair (MTTR)
- Now let us understand, with the help of an example, how can we use these terminologies to measure the availability of a disk subsystem

Availability

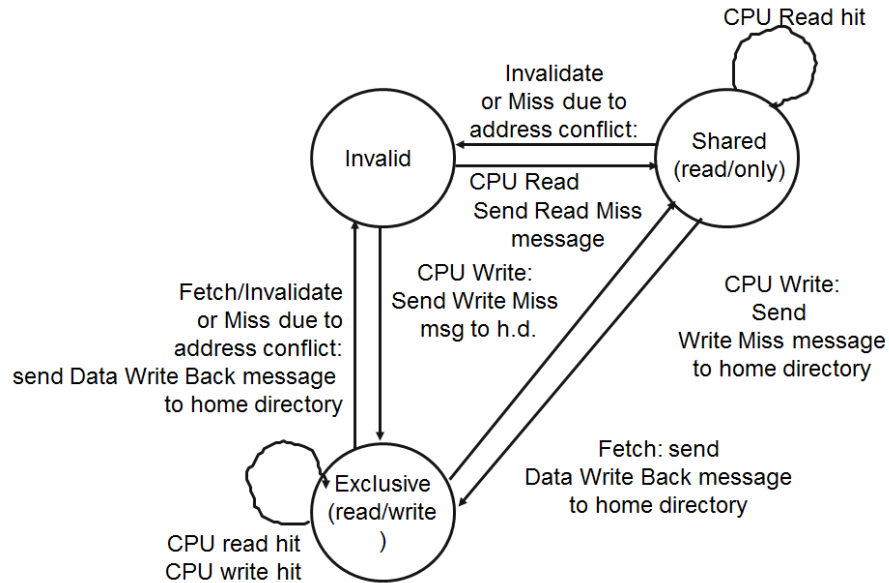
- The availability of a module is the measure of the service accomplishment with respect to the swinging between the two states of accomplishment and interruption
- The module availability, therefore can be quantified as the ratio of the MTTF and Mean Time Between Failure – MTBF (which is equal to the sum of MTTF and MTTR); i.e.,
- Availability = $\text{MTTF} / (\text{MTTF} + \text{MTTR})$
 $= \text{MTTF} / \text{MTBF}$

Question 18:

State transition diagram of directory based, and explain your diagram

Answer:

- State machine for CPU requests for each memory block
- Invalid state if in memory



- Here, the same states & structure is shown as the transition diagram for an individual cache
- Two actions performed are:
 1. update of directory state and
 2. send messages to satisfy requests

The controller tracks all copies of memory block; and also indicates an action that updates the sharing set, called Sharers, as well as sending a message

Question 19:

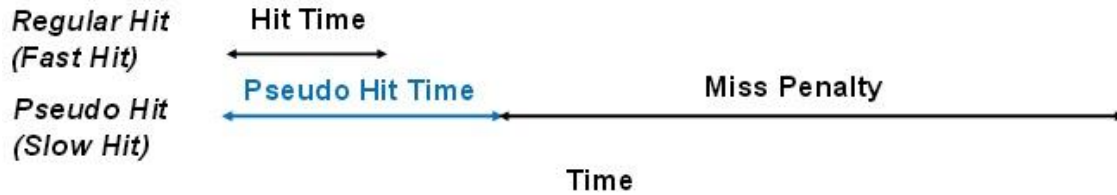
Define miss rate and explain how way prediction and pseudo associativity reduce the miss rate and improve performance 10marks

Answer:

- Miss Rate: is the fraction of memory accesses that are not found in the level-k memory or say the cache
 - ✓ Miss Rate = number of misses / total memory accesses
- As, Hit rate is defined as the fraction of memory access that are found in the level-k memory or say the cache, therefore Miss Rate = 1 – Hit Rate

Reducing Misses via “Pseudo-Associativity” and Way Prediction

- How to combine fast hit time of direct mapped cache and have the lower conflict misses of 2-way set associative cache?



Divide cache

- (if direct mapped): on a miss, check other half of cache to see if there, if so have a pseudo hit (slow hit) otherwise go to the lower level of hierarchy the lower level of hierarchy

Way prediction

- (if set associative) by extra bits to predict which of the two ways to try on the next cache access (predict the way to pre-set the mux)
 - ✓ If the way prediction is correct \Rightarrow Hit time
 - ✓ If not \Rightarrow Pseudo hit time and change the way predictor
 - ✓ Way prediction can also reduce power consumption
- Drawback: CPU pipeline is hard if hit takes 1 or 2 cycles
 - ✓ Better for caches not tied directly to processor (L2)
 - ✓ Used in MIPS R1000 L2 cache, similar in UltraSPARC
- The performance can degrade if many fast hit times become slow hit times
- For each set, it is important to indicate which block is the fast hit and which is the slow one

Question20:

Wider Main Memory: Example

Answer:

- 4 words (i.e. 32 byte) block
 - ✓ Time to send address = 4 clock cycles
 - ✓ Time to send the data word = 4 clock cycles
 - ✓ Access time per word = 56 clock cycles
 - ✓ Miss Penalty =
- $\text{No. of words} \times [\text{time to: send address} + \text{send data word} + \text{access word}]$

For 1 word organization

$$\text{Miss Penalty} = 4 \times (4 + 4 + 56) = 4 \times (64)$$

$$= 256 \text{ Clock Cycles;}$$

$$\text{The memory bandwidth} = \text{bytes/clock cycle} = 32/256 = 1/8 \text{ byte /cycle}$$

For 4-word organization

$$\text{Miss Penalty} = 1 \times (4 + 4 + 56) = 64 \text{ Clock Cycles; and}$$

$$\text{Memory bandwidth} = 32/64 = 1/2 \text{ bytes/cycle;}$$

Question 21:

You are building a system around a processor with in-order execution that runs at 1.1GHz and has a CPI of 0.7 excluding memory accesses. The only instructions that read or write data from memory are loads (20% of all instructions) stores (5% of all instructions).

The memory system for this computer is composed of a split L1 cache that imposes no penalty on hits. Both the I-cache and D-cache are direct mapped and hold 32KB each. The I-cache has 2% miss rate and 32-byte blocks, and the D-cache is write through with a 5% miss rate and 16-byte blocks. There is a write buffer on the D-cache that eliminates stalls for 95% of all writes.

The 512KB write-back, unified L2 cache has 64-byte blocks and an access time of 15ns. It is connected to the L1 cache by a 128-bit data bus that runs at 266 MHz and can transfer one 128-bit word per bus cycle. Of all memory references sent to the L2 cache in this system, 80% are satisfied without going to main memory. Also 50% of all blocks replaced are dirty.

The 128-bit-wide main memory has an access latency of 60ns, after which any number of bus words may be transferred at the rate of one per cycles on the 128-bit-wide 133 MHz main memory bus. What is the overall CPI, including memory accesses?

You are considering replacing the 1.1GHz CPU with one that runs at 2.1GHz, but is otherwise identical. How much faster does the system run with a faster processor? Assume the L1 cache still has no hit penalty, and that the speed of the L2 cache, main memory, and buses remains the same in absolute terms (e.g. the L2 cache still has a 15 ns access time and a 266MHz bus connecting it to the CPU and L1 cache).

Answer:

A useful tool for solving this type of problem is to extract all of the available information from the problem description. It is possible that not all of the information will be necessary to solve the problem,

but having it in summary form makes it easier to think about. Here is a summary:

- CPU: 1.1 GHz (0.909ns equivalent), CPI of 0.7 (excludes memory accesses)
- Instruction mix: 75% non-memory-access instructions, 20% loads, 5% stores
- Caches: Split L1 with no hit penalty, (i.e., the access time is the time it takes to execute the load/store instruction
- L1 I-cache: 2% miss rate, 32-byte blocks (requires 2 bus cycles to fill, miss penalty is 15ns + 2 cycles
- L1 D-cache: 5% miss rate, write-through (no write-allocate), 95% of all writes do not stall because of a write buffer, 16-byte blocks (requires 1 bus cycle to fill), miss penalty is 15ns + 1 cycle
- L1/L2 bus: 128-bit, 266 MHz bus between the L1 and L2 caches
- L2 (unified) cache, 512 KB, write-back (write-allocate), 80% hit rate, 50% of replaced blocks are dirty (must go to main memory), 64-byte blocks (requires 4 bus cycles to fill), miss penalty is 60ns + 7.52ns = 67.52ns
- Memory, 128 bits (16 bytes) wide, first access takes 60ns, subsequent accesses take 1 cycle on 133 MHz, 128-bit bus
- **A.** The average memory access time for instruction accesses:

- L1 (inst) miss time in L2: 15ns access time plus two L2 cycles (two = 32 bytes in inst. cache line/16 bytes width of L2 bus) = $15 + 2 \times 3.75 = 22.5\text{ns}$. (3.75 is equivalent to one 266 MHz L2 cache cycle)
- L2 miss time in memory: 60ns + plus four memory cycles (four = 64 bytes in L2 cache/16 bytes width of memory bus) = $60 + 4 \times 7.5 = 90\text{ns}$ (7.5 is equivalent to one 133 MHz bus cycle).
- Avg. memory access time for inst
 $= \text{avg. access time in L2 cache} + \text{avg. access time in memory} + \text{avg. access time for L2 write-back}$
 $= 0.02 \times 22.5 + 0.02 \times (1 - 0.8) \times 90 + 0.02 \times (1 - 0.8) \times 0.5 \times 90$
 $= 0.99\text{ns}$ (1.09 CPU cycles)
- **B.** The average memory access time for data reads: Similar to the above formula with one difference: the data cache width is 16 bytes which takes one L2 bus cycles transfer (versus two for the inst. cache), so
- L1 (read) miss time in L2: $15 + 3.75 = 18.75\text{ns}$
- L2 miss time in memory: 90ns
- Avg. memory access time for read
 $= 0.02 \times 18.75 + 0.02 \times (1 - 0.8) \times 90 + 0.02 \times (1 - 0.8) \times 0.5 \times 90$
 $= 0.92\text{ns}$ (1.01 CPU cycles)
- **C.** The average memory access time for data writes: Assume that writes misses are not allocated in L1, hence all writes use the write buffer. Also assume the write buffer is as wide as the L1 data cache.
- L1 (write) time to L2: $15 + 3.75 = 18.75\text{ns}$
- L2 miss time in memory: 90ns
- Avg. memory access time for data writes
 $= 0.05 \times 18.75 + 0.05 \times (1 - 0.8) \times 90 + 0.05 \times (1 - 0.8) \times 0.5 \times 90$
 $= 2.29\text{ns}$ (2.52 CPU cycles)
- **D.** What is the overall CPI, including memory accesses:
- Components: base CPI, Inst fetch CPI, read CPI or write CPI, inst fetch time is added to data read or write time (for load/store instructions).
- $\text{CPI} = 0.7 + 1.09 + 0.2 \times 1.01 + 0.05 \times 2.52 = 2.19 \text{ CPI}$

Solution:

a. AMAT of Instruction Access

AMAT_{instr} = HitTime_I-cache + MR_i-cache* (HT-L2 + Time to fetch miss block from L2 + Time to fetch from MM + Time to write back the dirty block when miss)

$$\begin{aligned}
 &= 0 + 2\% * (15 + (32 * 8 / 128) * TT_{L2}) + 20\% * ((60 + (64 * 8 / 128) * TT_{mm} + 20\% * 50\% * (60 + 64 * 8 / 128 * TT_{mm})) \\
 &= 2\% * (15 + 2 * 3.76 + 20\% * (60 + 4 * 7.52) + 20\% * 50\% * (60 + 4 * 7.52)) \\
 &= 2\% * (15 + 7.52 + 20\% * 90.08 + 20\% * 50\% * 90.08) = 0.99 \text{ (ns)}
 \end{aligned}$$

b. AMAT of Data Read

AMAT = HitTime_D-cache + MR_d-cache* (HT-L2 + Time to fetch miss block from L2 + Time to fetch from MM + Time to write back the dirty block when miss)

$$\begin{aligned}
 &= 0 + 5\% * (15 + (16 * 8 / 128) * TT_{L2}) + 20\% * ((60 + (64 * 8 / 128) * TT_{mm} + 20\% * 50\% * (60 + 64 * 8 / 128 * TT_{mm})) \\
 &= 5\% * (15 + 3.76 + 20\% * 1.5 * (60 + 4 * 7.52)) = 5\% * 45.78 = 2.29 \text{ (ns)}
 \end{aligned}$$

c. AMAT of Data Write (Every write will go to the write buffer. Each time for one word.)

AMAT = WriteTime when hit the write buffer Hit time of L1 + (1-95%)* (100% x (Hit time of L2 + Miss rate of L2 x Miss penalty of L2))

$$\begin{aligned}
 &= 0 + (1-95\%) * (15 + 1 * TT_{L2} + 20\% * (60 + 1 * TT_{mm})) \\
 &= 5\% * (15 + 3.76 + 20\% * (60 + 7.52 + 50\% * 90)) \text{ (NO WRITE ALLOCATE)} \\
 &= 1.61 \text{ (ns)}
 \end{aligned}$$

* ~~50%*90~~): No replacement will happen due to NO WRITE ALLOCATE.

How about write allocate?

$$\begin{aligned}
 &= 0 + (1-95\%) * (15 + 1 * TT_{L2} + 20\% * (90 + 50\% * 90)) \\
 &= 5\% * (15 + 3.75 + 20\% * 135) \\
 &= 2.29
 \end{aligned}$$

* 50%*90: time latency for write back the dirty block.

d. Over all of CPI

CPI = CPI of org + stalls for instruction reference + stalls of Data read per instruction + stalls of Data write per instruction

$$= 0.7 + 0.99/0.9 + 20\% * 2.29/0.9 + 5\% * 2.29/0.9 = 0.7 + 1.1 + 0.51 + 0.13 = 2.44$$

Question 22:

- a) Explain the concept of cache and locality in memory hierarchy
- b) How much fast is CPU if AVG CPI is 1.2 with instruction as hit ... as compared to one having miss penalty of 30 cycles having 40% instructions load...

Answer:**(a)****Concept of Caching**

- staging area or temporary-place to:
 - ✓ store frequently-used subset of the data or instructions from the relatively cheaper, larger and slower memory; and
 - ✓ To avoid having to go to the main memory every time this information is needed

Caching and Memory Hierarchy

- Memory devices of different type are used for each value k – the device level
- the faster, smaller device at level k , serves as a cache for the larger, slower device at level $k+1$
- The programs tend to access the data or instructions at level k more often than they access the data at level $k+1$
- Storage at level $k+1$ can be slower, but larger and cheaper per bit
- A large pool of memory that costs as much as the cheap storage at the highest level (near the bottom in hierarchy)
- serves data or instructions at the rate of the fast storage at the lowest level (near the top in hierarchy)

Principle of Locality

- Programs access a relatively small portion of the address space at any instant of time
- E.g.; we all have a lot of friends, but at any given time most of us can only keep in touch with a small group of them
- Spatial locality: Items with nearby addresses (i.e., nearby in space) be located at the same level, as they tend to be referenced close together in time
- Temporal locality: Recently referenced items (i.e., referenced close in time) be placed at the same memory level, as they are likely to be referenced in the near future
- Locality Example: Program


```
sum = 0;
for (i = 0; i < n; i++)
    sum += a[i];
return sum;
```

 - Spatial Locality: All the array-elements $a[i]$ or data, reference in succession at each loop iteration, so all the array elements be located at the same level. All the instructions of the loop are referenced repeatedly in sequence therefore be located at the same level
 - Temporal Locality: The data, sum is referred each iteration; i.e., recently referred data is referred in each iteration. The Instructions of a loop, $sum += a[i]$ Cycle through loop repeatedly

Question 23:

Find the local and global miss rates, the Average Memory Access Time and Average memory stall cycles per instruction, given that for 1000 reference with 40 misses in L1 cache and 20 in L2 Cache;

- Assume:
 - ✓ miss penalty for L2 cache-memory = 100 clock cycles
 - ✓ hit time for L2 cache is 10 clock cycle
 - ✓ Hit time for L1 cache is 1 Clock cycle
 - ✓ Memory Reference per instruction = 1.5

Answer:

- Miss rate for either local or global
- L1 cache is same = 4% $[(40/1000) \times 100]$
- Local Miss Rate for L2 = 50% $[20/40]$
- Global Miss Rate for L2 cache = 2% $[(20/1000) \times 100]$
- Average Memory Access time
 - = Hit Time L1 + Miss Rate L1 x Miss Penalty L1 ..(1)
- where, Miss Penalty L1
 - = Hit Time L2 + Miss Rate L2 x Miss Penalty L2 .. (2)
- Substituting 2 in 1 we get,
- Average Memory Access time
 - = Hit Time L1 + Miss Rate L1 x (Hit Time L2 + Miss Rate L2 x Miss Penalty L2)
 - = $1 + 4\% \times (10 + 50\% \times 100) = 3.4$ cycles
- Average Memory Stalls per instruction (i.e., miss penalty)
 - = Misses per instruction L1 x Hit Time L2
 - + Misses per instruction L2 x Miss Penalty L2
- For Memory references per instruction = 1.5
 - ✓ Misses per instruction for L1 = $40 \times 1.5 = 60$ per 1000 instructions
 - ✓ Misses per instruction for L2 = $20 \times 1.5 = 30$ per 1000 instructions
- Average Memory Stalls per instruction
 - = $(60/1000) \times 10 + 30/1000 \times 100$
 - = $0.6 + 3.0 = 3.6$ clock cycles
- i.e., the average miss penalty using multi level caches reduces by a factor of $100/3.6 = 28$ relative to the single level cache

Question 24:

here is a series of address of reference 2,3,11,16,21,13,64,48,19,11,3,22,4,27,11 assuming a direct mapping cache with four word blocks and a total size of 16 words that is initially empty ,

label each reference in hit the list as a hit or miss and show the final contents of cache.

Answer:

2–miss, 3–hit, 11–miss, 16–miss, 21–miss, 13–miss, 64–miss, 48–miss, 19–miss, 11–hit, 3–miss, 22–hit, 4–miss, 27–miss, 6–hit, 11–miss

Cache set	Address
00	[0, 1, 2, 3]
01	[4, 5, 6, 7]
10	[8, 9, 10, 11]
11	[12, 13, 14, 15]

Question 25:

We wish to compare the performance of two different computers: M1 and M2. The following measurements have been made on these computers:

Program	Time on M1	Time on M2
1	2.0 seconds	1.5 seconds
2	5.0 seconds	10.0 seconds

Program	Instructions executed on M1	Instructions executed on M2
1	5×10^9	6×10^9

- Which computer is faster for each program, and how many times as fast is it?
- Find the instruction execution rate (instructions per second) for each computer when running program 1.
- The clock rates for M1 and M2 are 3 GHz and 5 GHz respectively. Find the CPI for program 1 on both machines.
- Suppose that program 1 must be executed 1600 times each hour. Any remaining time should use to run program 2. Which computer is faster for this workload?
Performance is measured here by the throughput of program 2.

Answer:

a) For program 1, $M2$ is $2.0/1.5 = 1.33$ times as fast as $M1$.
For program 2, $M1$ is $10.0/5.0 = 2$ times as fast as $M2$.

b) For program 1:

Execution rate on $M1 = 5 \times 10^9 / 2.0 = 2.5 \times 10^9$ IPS (Instructions Per Second).

Execution rate on $M2 = 6 \times 10^9 / 1.5 = 4 \times 10^9$ IPS.

c) $CPI = \text{Execution time} \times \text{Clock rate} / \text{Instruction Count}$

For program 1:

$CPI \text{ on } M1 = 2.0 \times 3 \times 10^9 / (5 \times 10^9) = 1.2$ cycles per instruction

$CPI \text{ on } M2 = 1.5 \times 5 \times 10^9 / (6 \times 10^9) = 1.25$ cycles per instruction

d) Running program 1 1600 times each hour:

On M1, time required for program 1 = $1600 \times 2.0 = 3200$ seconds

On M2, time required for program 1 = $1600 \times 1.5 = 2400$ seconds

Time left to run program 2 each hour:

On M1, time left for program 2 = $3600 - 3200 = 400$ seconds

On M2, time left for program 2 = $3600 - 2400 = 1200$ seconds

In that time left, program 2 can run:

On M1, program 2 can run $400/5 = 80$ times

On M2, program 2 can run $1200/10 = 120$ times

Thus M2 performs better on this workload than M1.

Inverted memory... organization ???

Q#1: Define miss rate and explain how way prediction and pseudo associativity reduce the miss rate and improve performance (10 marks)

Q2: How snooping protocol cache coherence different from directory based protocol? Draw write invalidate technique for cache coherence. (10 marks)

Q3: What are the parameters of the reliability of I/O storage. (10 marks)

Q4: Draw transition diagram for individual cache block in directory based system explain state transition? (15 marks)

Q5: here is a series of address of reference 2,3,11,16,21,13,64,48,19,11,3,22,4,27,11 assuming a direct mapping cache with four word clocks and a total size of 16 words that is initially empty, label each reference in hit the list as a hit or miss and show the final contents of cache. (15 marks)

Q2. In multiprocessor programming what is coherence, if we have 3 processors, what are the different strategies for write.

Q4: You have a computer system, processor 1.1 GHz, L1 Cache Data and Instructions.

Instruction 64KB and data 16KB, 20% instruction are load/store.

20% instruction are mem access and 5% data references.

64KB Unified L2 Cache.

5% instruction miss in L1, and 2% data miss.

80% of L2 cache are Hit, 50% are dirty replaced.

processor 1.1GHz,

L1 to L2 bus 128bit at 266MHz, having 15ns access.

L2 is 128bit buss , at 133MHz after 60% load.

- a) what is the mem access time for data instructions?
- b) what is mem access time for data read?
- c) what is mem access time for data write?

Q5: Virtual Mem address, 25 bit, Physical address 22 bit, 2 power 16 (64kb) pages.

- a) page table size in bits.
- b) 64Kb to 16kb by OS , what are your suggestions.
- c),