COMP25212

Two hours

No special instructions.

UNIVERSITY OF MANCHESTER
SCHOOL OF COMPUTER SCIENCE

System Architecture

Date

Time

Please answer any THREE Questions from the FOUR questions provided

Use a SEPARATE answerbook for each SECTION

For full marks your answers should be concise as well as accurate.
Marks will be awarded for reasoning and method as well as being correct

---

The use of electronic calculators is permitted provided they
are not programmable and do not store text.

[PTO]

Section A

1. **Caches**

a) You are currently working as a member of the design team for a new processor and have a choice between using either a split Level 1 (L1) cache design (i.e., Harvard architecture) or unified L1 cache. After testing the two designs on a simulator, your team has obtained the following average miss rates:

i) Option 1 (split cache design): 1% L1 instruction cache misses and 10% L1 data cache misses.
ii) Option 2 (unified cache design): 2% L1 cache misses.

In each case, the penalty of missing in the L1 cache was 15 clock cycles and half of all instructions also access data memory. Accessing L1 takes 1 clock cycle in all cases.
What is the average memory access time in each case? Which of the two designs achieves the lowest average memory access time? How much better is the chosen solution than the other?

(5 marks)

Solution: [bookwork + problem-solving]

For every instruction we need to load the instruction and, on average, pay ½ the cost of loading a piece of data.
The average access times are:

Option 1: $T = 1 * (1 + 1/100 * 15) + ½ * (1 + 10/100 * 15)$
$= 1.15 + 2.5/2$
$= 2.4$ cycles *(2 marks)*

Option 2: $T = (1 + ½) * (1 + 2/100 * 15)$
$= 1.5 * 1.3$
$= 1.95$ cycles *(2 marks)*

Option 2 is better: in this scenario, a unified L1 cache is better than a split cache.
T1/T2 = 1.23 so 23% better. Any answer around 20% and 25% is acceptable. *(1 mark)*

Other well-founded answers based on different interpretations of what a better solution is can get accepted as well.

b) Let us consider a system with a byte-addressable memory within a 16-bit address-space and with a direct-mapped cache of 32 lines, where each cache line holds 32 bytes of data.

How many bits of each address will be allocated respectively to the offset, to the index and to the tag? (2 marks)

Solution: [bookwork + practical question]

Each cache line holds 32 bytes = 2^5 bytes, therefore 5 offset bits.
The cache holds 32 lines and is direct-mapped, so 2^5 entries and 5 index bits.
Out of a 16 bit address, the tag contains 16 – 5 – 5 = 6 bits.

c) Within the system proposed in question (1.b) above, let us consider the following sequence of memory addresses used in this exact order to reference memory. Assume that the cache is initially empty. For each memory reference, give the tag and index bits and whether the access is a hit or a miss in the cache.

i) 0101 1111 1011 0101
ii) 0101 0110 1100 1010
iii) 1100 1001 1010 0000
iv) 0101 0110 1101 1101
v) 1110 1110 1100 1001

(4 marks)

Solution: [practical question]

Address	**Tag (6 MSB)**	**Index (next 5 bits)**	**Cache access**
0101 1111 1011 0101	*0101 11*	*11 101*	*miss*
0101 0110 1100 1010	*0101 01*	*10 110*	*miss*
1100 1001 1010 0000	*1100 10*	*01 101*	*miss*
0101 0110 1101 1101	*0101 01*	*10 110*	*hit*
1110 1110 1100 1001	*1110 11*	*10 110*	*miss*

d) What type of cache misses are encountered in question (1.c) above? In which memory reference was each cache miss type encountered?

(2 marks)

Solution: [bookwork]

The first three cache misses are compulsory misses. Indeed, we assumed that the cache was initially empty and these indexes had not been used previously. (1 mark)

The last cache miss is a conflict miss because the index is shared with the second and fourth addresses, but the tag differs. (1 mark)

e) A microprocessor with a single level of cache has an average memory access time of 3 clock cycles. The access time is 1 clock cycle when data is found in the cache and 100 clock cycles are needed when the data is not found in the cache. It is proposed to add a second level of cache to improve the average memory access time.

Considering that the objective is to achieve a 50% speedup (i.e., improvement) in the average memory access time and engineers have determined that the L2 cache will have a 10 cycle access time, what is the minimum hit rate required for this second level of cache in order to achieve the targeted improvement? Show your reasoning. (7 marks)

Solution: [problem solving]

The objective is to reach a 50% speedup:

Speedup = (old average access time) / (new average access time)

(new average access time) = 3 cycles / 1.5 = 2 cycles

So the design objective is to achieve an average memory access time of 2 clock cycles. *(2 mark)*

We need to compute the original miss rate in the case where there is a single level of cache:

$T_i = t_i + m_i * T_{i+1}$ *where* T_i *is the perceived access time at level "i" of the memory hierarchy, "*t_i*" is the technology imposed cost of accessing the i-th level of the hierarchy and "*m_i*" is the miss rate at level "i".*

So in this case for a single level of cache:

$T_{old} = t_1 + m_1 * T_{mem}$

*(old access time) = (L1 cache access time) + (L1 miss rate) * (RAM access time)*

(L1 miss rate) = (3 – 1) / 100 = 2% *(2 marks)*

We can apply the same formula again at the second level of cache:

$T_{new} = t_1 + m_1 * (t_2 + m_2 * T_{mem})$

We substitute the known values:

$2 = 1 + 2/100 * (10 + m_2 * 100)$

$m_2 = (50 - 10) / 100 = 40\ \%$

The miss rate in the new L2 cache should be below 40% if we are to meet the 50% speedup target. *(3 marks)*

*Answers based on a different interpretations of 50% speed-up (i.e. 3*50%=1.5 cycles) will get accepted as well.*

2. **Virtualization and Storage**

a) Let us consider a RAID 50 system built using 10 identical disks, partitioned in two groups of 5 disks, each group is organised as a RAID 5 system. Each disk is of size 2TB (=$2*10^{12}$ bytes) and has a sustained bandwidth of 200 MB/s. We further assume that disk failures are independent and uniformly distributed, that each disk has a MTTF (mean time to failure) of 1 million hours and that, in the event of a failure, the MTTR (mean time to repair) is 10 hours.

i) What is the minimum number of disks which must fail before data is lost in this array? (2 mark)

Solution: [bookwork + practical question]

Data is lost only once the redundancy fails. In this case this will happen if either group loses more than one disk.
Only 1 disk can safely fail before data is lost. Indeed if two disks fail within the same group, the entire array fails.

ii) Consider the case where one disk has failed and has been replaced with a new, empty disk. Assuming that the disk was full, what proportion of the available bandwidth of the remaining disks will be used to restore the disk array on average over the MTTR period? (4 marks)

Solution: [bookwork + problem-solving]

In a RAID 5, a failed disk can be reconstructed by using the distributed parity across the remaining disks. In this scenario, 4 disks are left in the same group, that contain the data and parity required to reconstruct the failed disk.(1 mark)
This means that each of the 4 remaining disks must be read in full. This would take a minimum of:

$$2*10^{12} / (200 * 10^{6}) = 10{,}000 \text{ seconds}$$ *(1 mark)*

As the MTTR is stated as 10 hours, this gives us an average of:

$$10\,000 / (10 * 60 * 60) = 10 / 36 = 28\%$$

Accepted answers rounded up to "around a third". *(2 marks)*

iii) What is the MTTF of the entire array (also called "mean time to data loss" (MTTDL))?

(3 marks)

Solution: [bookwork + problem-solving]

Data loss will occur only if, once a disk has failed, a second disk fails within the same group during the repair time (MTTR period). *(1 mark)*
Critically, this relies on the assumption of independence of disk failures. We then have:

$$MTTDL = (MTTF / 10) * (MTTF / (4 * MTTR))$$
$$= MTTF^2 / (10 * 4 * MTTR)$$
$$= 2.25 * 10^{12} / 400$$
$$= 5.625 * 10^9 \text{ hours}$$ *(2 marks)*

b) You are responsible for selecting a cloud storage solution for your company. The service level agreement of provider A boasts a system that is 100% available and 99% reliable, while provider B offers 99% available and 100% reliable. Which provider will you choose? Why? (2 marks)

Solution: [problem-solving]

Choose Provider B. (1 mark)
Provider A does not guarantee reliability, so data may be lost permanently. (1 mark)

c) How can system virtualization be used to ensure high availability for important applications? (4 marks)

Solution: [bookwork]

High availability can be achieved by running an application in a VM and maintaining a copy of this VM up-to-date on a separate, standby hardware system. (2 marks)
The active VM's image can be regularly copied to the standby system without activating it. (1 mark)
If the main VM stops responding (VM crashes or VMM crashes or hardware fails), activate the VM on the standby system. (1 mark)

d) One of the early problems of achieving transparent virtualization (i.e., running unmodified guest operating systems in a VM) was that older versions of the x86 architecture contain instructions that behave differently in privileged mode and unprivileged mode without trapping.
For example, in privileged mode, the "popf" instruction would change both system flags (e.g., the Interrupt Flag which determines whether the CPU will handle or ignore certain hardware interrupts) as well as ALU flags (e.g., Zero Flag). However, in unprivileged mode, "popf" would not change any system flags, silently ignoring them.

i) Explain why an instruction like "popf" in old x86 architectures prevented transparent virtualization.

(2 marks)

Solution: [bookwork + problem-solving]

If the guest operating system is not aware of running within a VM, it will expect to run in privileged mode rather than in unprivileged mode. As the "popf" instruction silently ignores system flags (no trap) in unprivileged mode, the effect of this instruction will be different from what the guest OS expects.

ii) Could you change the x86 architecture to avoid this issue for virtualized systems while preserving the hardware's behaviour for non-virtualized systems? (2 marks)

Solution: [problem-solving]

An additional bit flag could be added that makes "popf" trap when it is set. The VMM could set this flag whenever a guest OS is running. This would also transparently preserve the existing behaviour of non-virtualized systems when the bit flag is cleared.

iii) Suggest an alternative to transparent virtualization which does not suffer from this issue. (1 mark)

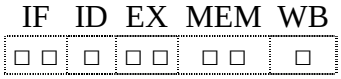
Solution: [bookwork]

Para-Virtualization: the guest OS could be made aware of virtualization and modified to account for the behaviour of such instructions (e.g., call the VMM to handle such cases).

Section B

3. **Processor architecture**

a) Consider an 8-stage fully pipelined scalar processor as the one below.



(IF, EX and MEM: 2 stages each; ID and WB: 1 stage each)

i) Explain what a Data hazard is and quantify the penalties incurred by Data hazards in such a processor.
ii) What techniques can we use to mitigate Data hazards and how they would affect these penalties?
iii) Explain what a Control hazard is and quantify the penalties incurred by Control hazards in this processor.
iv) What techniques can we use to mitigate Control hazards and how they would affect these penalties?

(10 marks)

Solution: [Problem-solving] & [Bookwork] 1 mark per each up to a maximum of 10 marks

i) ***Data Hazards:***
An instruction needs to read from a register that is written by a previous instruction before it is stored in the register bank which will require to stall the pipeline until the data is available. *(1 mark)*

Up to 4 cycle penalty (1EX+2 MEM+1WB) for 2 consecutive instructions with data dependencies if forwarding is not implemented. 1st instruction needs to WB before the 2nd advances to EX (see below). *(1 mark)*

	1	2	3	4	5	6	7	8	9	10	11	12	13
add r1,r0,r3	IF	IF	ID	EX	**EX**	MEM	MEM	WB					
sub r1,r0,r3		IF	IF	ID	stall	stall	stall	stall	EX	EX	MEM	MEM	WB

ii) ***Data Hazard Mitigation****:*
To reduce the impact of data hazards, forwarding/bypassing can be used. It consists in adding connections from each stage after EX (i.e. 5-8) to the input of the EX (4) and MEM (6) stages. *(1 mark)*
1 cycle penalty for EX → EX dependencies

	1	2	3	4	5	6	7	8	9	10
add r1,r0,r3	IF	IF	ID	EX	**EX**	MEM	MEM	WB		
sub r1,r0,r3		IF	IF	ID	stall	EX	EX	MEM	MEM	WB

3 cycle penalty for MEM → EX dependencies

	1	2	3	4	5	6	7	8	9	10	11	12
add r1,r0,r3	IF	IF	ID	EX	EX	MEM	**MEM**	WB				
sub r1,r0,r3		IF	IF	ID	stall	stall	stall	EX	EX	MEM	MEM	WB

(1 mark)

Reordering instructions either during compilation or in hardware can help to reduce the number of pipeline stalls by separating enough cycles those instructions that have data dependencies. Penalties will vary depending on the number of independent instructions.
(1 mark)

iii) ***Control Hazards:***
When executing a branch, the following instruction is not decided until ID if it is unconditional or EX if it is conditional, so the instruction fetched after a branch may not be the one that needs to be executed after the branch instruction. *(1 mark)*

Unconditional branches: 2 cycles (1 IF + 1 ID) *(1 marks)*

	1	2	3	4	5	6
b n	IF	IF	ID	EX	EX	MEM
inst 1		IF	IF	ID	EX	EX
inst 2			IF	IF	ID	EX
n				IF	IF	ID

Conditional branches: 4 cycles (1 IF + 1 ID + 2 EX) *(1 marks)*

	1	2	3	4	5	6
Beq n	IF	IF	ID	EX	EX	MEM
inst 1		IF	IF	ID	EX	EX
inst 2			IF	IF	ID	EX
inst 3				IF	IF	ID
inst 4					IF	IF
n						IF

iv) ***Control Hazards Mitigation****:*
The solution is either to stall the pipeline once a branch instruction is decoded, or to instrument the compiler to add NOP instructions after each branch instruction. *(1 mark)*

This will not affect the penalties, but will ensure that no unnecessary instruction is executed.
(1 mark)

A more advanced solution is to use branch prediction and perform speculative execution, but a roll-back mechanism needs to be implemented to undo any changes made if prediction fails.
(1 mark)

No penalties when the branch prediction predicts correctly the branch. Same penalties when it misspredicts. *(1 mark)*

Other well-founded answers might get accepted as well. Diagrams are not needed as part of the answer, but are shown here for the sake of completeness.

b) Enumerate and explain the main factors limiting the scalability of processor design. (6 Marks)

Solution: [Bookwork] ***- 2 Marks each***

The Memory Wall*: Memory speed does not improve as fast as processor speed. Accessing data becomes the bottleneck for many applications.*

The Power Wall*: Power increases with frequency and voltage and leakage increases tremendously for shrinking transistor sizes. This power transforms into heat which affect the physical properties of the integrated circuits and may end up burning the chip. Dissipating this heat is only possible to a certain extent.*

The ILP Wall*: not all applications have enough inherent parallelism to be exploited by current processor technologies.*

c) Explain the benefits and limitations of reordering instructions in the compiler or in hardware. (4 marks)

Solution: [Bookwork]
1 mark for each correctly identified pro/con e.g.

Compiler
Pros: simpler hardware design (i.e. less area and power), static analysis can be more detailed (i.e. larger window of instruction)
Cons: binary optimised for a particular hardware (lack of portability), extra instructions (NOPS), fixed solution for each execution

Hardware
Pros: Legacy compatibility, Simple compiler, application independent, dynamically adaptable
Cons: Hardware complexity affects freq. power, etc, small window of instructions, precise interruptions/exceptions require complex hardware.

4. **Multithreading / Multicore**

a) We need to design a control system for a vision-controlled robot. The system will take a single stream of input data from its camera and execute 4 different, independent operations to take decisions in real-time. Assume that the 4 operations feature low ILP and that a frame from the camera fits within cache. Discuss the pros and cons (e.g., processor complexity, memory subsystem, performance) of running this control system over:

i) a non-pipelined processor
ii) a pipelined, superscalar processor
iii) a 4-way multithreaded processor
iv) a quad-core processor

Based on that, which of them will be the best solution and why? (10 marks)

*Solution: [Problem-solving] [not covered in the lectures] (**2 marks**)*

*i) A **non-pipelined processor** will have low complexity both in CPU and memory subsystem architecture, but will have very low performance with many wasted cycles and no opportunity to exploit operation-level parallelism (could be done in software, though) (**2 marks**)*

*ii) A pipelined**, superscalar processor** would improve the performance compared with the previous one, but will still not make use a good use of the computing resources as the operations have low ILP and operation-level parallelism is not exploited. It will increase processor complexity substantially. (**2 marks**)*

*iii) A **multithreading configuration** will add some complexity by replicating the control logic of the processor but, as there is data shared between the 4 operations, it will be able to exploit cache locality (once one of the threads bring the data, the rest will have it available) and will allow the low-ILP threads to increase the utilisation of the processing resources by reducing the number of stalls due to dependences. (**2 marks**)*

*iv) A **quad-core processor** will increase complexity greatly as it will require to replicate the whole of the processor 4 times. Moreover a cache coherency protocol will be needed which may reduce the performance due to a slower memory access due to the coherence overhead resulting of data sharing. (**2 marks**)*

*Overall, multithreading seems to be the best solution as it should get good resource utilization and performance with a (relatively) low-complexity architecture. (**2 marks**)*

Answers with well-founded reasoning will get some marks.

b) Explain the difference between memory coherence and consistency in the context of shared memory multicore systems. (4 Marks)

Solution: [Bookwork] - ***2 Marks each***

Memory coherency*: ensure all changes are seen everywhere, i.e. that no outdated copy remains in the system*

Memory consistency*: ensure correct ordering of data accesses, including atomic execution of operations when needed*

c) Enumerate and explain the different synchronization mechanisms that can be used in the context of shared memory multicore systems to ensure program consistency is maintained. (6 Marks)

Solution: [Bookwork] - ***2 Marks each***

Lock*: Ensure that only one thread can advance into an atomic section*

Fence*: Ensure all the instructions before a certain point have completed before starting issuing instructions after the fence (required in out-of-order architectures)*

Barrier*: Ensure all the threads have reached a certain point of the execution before any of the threads is able to advance.*