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) What are the three categories of cache misses? Explain the cause in each case.
(3 marks)

*Compulsory misses occur when the cache is cold, either when the program starts or if the cache has been flushed or if this is the first access to a cache line. (1)*
*Capacity misses occur when the cache size is insufficient to hold the working set of a program (alt. when the program accesses more memory locations than can fit in the cache simultaneously). (1)*
*Conflict misses occur for direct-mapped or set-associative caches when a program accesses more memory locations mapping to a given set (alt. mapping to the same index in the cache) than the cache associativity level. (1)*

b) Explain the effect of either increasing or decreasing each of the following cache parameters on each of the three previous cache miss categories. You should only consider the effect of modifying one parameter at a time (NOT combined effects).
- cache size
- line size
- associativity

(5 marks)

*Increasing the cache size will reduce capacity misses (1) as well as conflict misses because there are fewer memory locations mapping to the same set. (1)*
*Increasing the line size will reduce the number of lines, therefore reducing the number of compulsory misses (1), but will increase conflict misses as the number of memory locations mapping to each set increases. (1)*
*Increasing the associativity of the cache increases the number of cache lines per set, therefore reducing conflict misses. (1)*

c) Describe the two forms of locality exploited by caches and give at least one example for each type of locality occurring in programs.
(4 marks)

*Temporal locality means that memory locations accessed recently are likely to be accessed again within a short time span. (1) Examples: instructions in a loop, instructions in frequently called functions, data in the stack frame (local variables) … (1 for a correct example, max 1)*
*Spatial locality means that memory locations located close to recently accessed addresses are likely to be accessed as well. (1) Examples: sequence of instructions, data in arrays,*

*structured data (fields in a struct or class), data in the stack … (1 for a correct example, max 1)*

d) Consider a CPU with a single level of cache such that the cache hit ratio is **h**, the cache access time (hit time) is $\mathbf{t_h}$ and the main memory access time (miss time) is $\mathbf{t_m}$.

1) What is the perceived average memory access time **T** for this CPU ? *[Note: You are expected to provide a symbolic expression and explain your reasoning.]*

(3 marks)

$$T = h*t_h + (1-h)*(t_h+t_m) = t_h + (1-h)*t_m$$

*When missing in the cache it is not only necessary to pay the memory access time $t_m$, but also the cache access time $t_h$ to determine whether the memory location is already cached. The resulting memory access time is therefore the hit time plus the miss ratio (1-h) multiplied by the main memory access time.*

2) What is the ***maximum speedup $S_{max}$*** of the memory system that can be achieved with a single level of cache if the hit ratio **h** is fixed but the hit time $\mathbf{t_h}$ can be reduced to 0 ?

(2 marks)

$$S = t_m / T = t_m / (t_h + (1-h)*t_m)$$

*So if we replace $t_h$ by 0, we get:*

$$S_{max} \approx (1-h)^{-1}$$

*The speedup is limited by 1/(miss ratio).*

3) Evaluate this maximum speedup $S_{max}$ when **h** is 90%, 95% and 98%. *[Note: you are expected to provide numerical answers.]*

(1 mark)

*If h = 90% then $S_{max}$ = 1/0.1 = 10.*
*If h = 95% then $S_{max}$ = 1/0.05 = 20.*
*If h = 98% then $S_{max}$ = 1/0.02 = 50.*

4) Consider that the main memory access time is $t_m$ = 200 CPU clock cycles. What is the average perceived memory access time $T_{min}$ in the three hit ratio scenarios above if $\mathbf{t_h}$ can be reduced to 0 ? (1 marks)

*$T_{min} = t_m / S_{max}$*
*If h = 90% then $T_{min}$ = 20 cycles.*
*If h = 95% then $T_{min}$ = 10 cycles.*
*If h = 98% then $T_{min}$ = 4 cycles.*

5) What could be done to improve the performance of this memory system? (1 mark)

*Open-ended question. Examples of sensible answers:*
*Adding additional cache levels between L1 and main memory.*
*Increase the size of L1.*
*Faster access to Main memory.*
*Improve hit rate through prefetching and/or replacement policies.*

2. **Storage**

a) A server hard disk drive (HDD) is specified as having a mean "seek time" of 4 milliseconds, a "rotation speed" of 10,000 revolutions per minute and a "transfer rate" of 200 megabytes per second.

*1)* Explain what each term in this specification means for the performance of disk operation. (3 marks)

*The seek time measures the time it takes the head assembly to move to the track of the disk that will be accessed. (1)*
*The rotation speed determines the time it takes on average for a given disk sector to arrive under the head, also called "search time". (1)*
*The transfer rate determines the amount of data per unit of time that can be accessed sequentially once the head assembly is in position. (1)*

2) How long on average would a transfer of 8 kilobytes take from a random position on the disk? And for 8 megabytes? (4 marks)

*The time required to transfer data is "seek time" + "search time" + "transfer time" (1)*
*At 10,000 RPM, the search time is ½ * 60/10000 = 3 milliseconds. (1 mark)*
*A transfer of 8 kilobytes will take 4ms + 3ms + 8/200000s = 7.04 ms ("about 7ms" is acceptable) (1 mark)*
*A transfer of 8 megabytes will take 4ms + 3ms + 8/200s = 47ms (1 mark)*

b) A solid state drive (SSD) is specified as having a streaming read transfer rate of 500 megabytes per second, a streaming write transfer rate of 250 megabytes per second, a random 4 kilobytes read latency of 12 microseconds and a random 4 kilobytes write latency of 24 microseconds.

1) Considering the HDD from the previous question, approximately how much faster would this SSD be to **read** 8 kilobytes from a random position? And 8 megabytes? (2 marks)

*For the 8 kilobytes read on the SSD: 12us + 4/500000 = 20us*
*Compared to the HDD: 7040/20 = 352 (accept around 350 times)*

*For the 8 megabytes read on the SSD: 8/500 = 16ms*
*Compared to the HDD: 47/16 = 2.9375 (accept around 3 times)*

2) Same question for **writing** 8 kilobytes and 8 megabytes. (2 marks)

*For the 8 kilobytes write on the SSD: 24us + 4/250000 = 40us*
*Compared to the HDD: 7040/40 = 176 (accept around 175 times)*

*For the 8 megabytes read on the SSD: 8/250 = 32ms*
*Compared to the HDD: 47/32 = 1.47 (accept around 1.5 times)*

3) Could multiple HDDs be used to bridge this performance gap with the SDD? Explain for each of the two scenarios above how this can be achieved or why it cannot.

(3 marks)

*Striping across three disks (RAID 0) would allow to achieve similar transfer rates for 8 megabyte reads and better performance for writes in this example. (1)*
*However, regardless of the type of RAID configuration, hard disk drives incur a static transfer initiation cost (search + seek time) that cannot be lowered due to physical constraints (mechanical operation). It is therefore not possible to bridge the gap in the case of random short accesses. (2)*

e) Explain the two main reasons, aside from increasing storage capacity, why more than one disk drive may be used in a single system and how these goals may be achieved. (6 marks)

*The main reasons are to enhance performance and reliability. (2)*
*Performance is improved by striping the file system across multiple disks (RAID 0) which increases the transfer rates. (1)*
*Reliability can be improved either by mirroring (RAID 1) (1) or by storing parity data that allows reconstructing lost data in case of disk failure (RAID 2-6) (1).*
*(1 additional mark for identifying the corresponding RAID versions in each case).*

## <u>Section B</u>

3. **<u>Pipelining</u>**

a) What is pipelining in the context of processor design? What benefits does it provide? What issues arise from this architecture? (6 marks)

*Solution: [Bookwork]*

**Definition (1 mark)**
*Pipelining is a technique in which the instructions are split into different stages in a way that multiple instructions can overlap their execution.*
*e.g.*

| | t=0 | t=1 | t=2 | t=3 | t=4 | t=5 | t=6 | t=7 | t=8 | t=9 |
|---|---|---|---|---|---|---|---|---|---|---|
| ldr r1 x | IF | ID | EX | MEM | WB | | | | | |
| ldr r2 y | | IF | ID | EX | MEM | WB | | | | |
| ldr r3 z | | | IF | ID | EX | MEM | WB | | | |
| add r4 r1 r2 | | | | IF | ID | EX | MEM | WB | | |
| sub r5 r4 r3 | | | | | IF | ID | EX | MEM | WB | |
| str r5 a | | | | | | IF | ID | EX | MEM | WB |

**Benefits (1 mark)**
*It provides a more efficient utilisation of the processor resources while at the same time allowing increasing clock frequency. Computational throughput (i.e. instructions executed per unit of time) is increased.*

**Issues**
*The interleaving of instructions creates new issues (known as Hazards)*

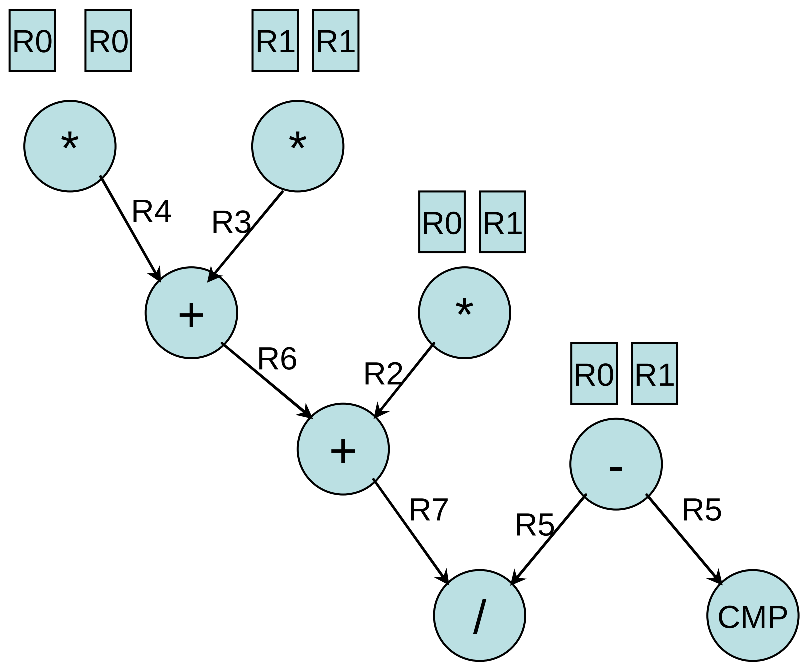
*Data Hazard* **(2 marks)***: An instruction needs to read from a register that is written by a previous instruction before it is stored in the register bank which will require to stall the pipeline until the data is available.*

*Control Hazard* **(2 marks)***: When executing a branch, the following instruction is not decided until the instruction is decoded if it is unconditional or until it's executed 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.*

b) Draw the data-dependency graph for the following application. (4 marks)

```
MUL R2, R0, R1
MUL R3, R1, R1
MUL R4, R0, R0
SUB R5, R1, R0
CMP R5, #0
ADD R6, R4, R3
ADD R7, R6, R2
DIV R8, R7, R5
```

*Solution: [practical question]*



c) Based on that dependency graph, discuss how suitable that code is for being executed in a superscalar processor (assume all arithmetic operations can be done in a single clock cycle). (2 marks)

*Solution: [bookwork + practical question]*

*The dependence graph above shows that the code could be executed perfectly well in a 2-way superscalar processor as 2 instructions will be ready for execution each cycle. However, it will not benefit of higher instruction level parallelism in the architecture, as it's not possible to run these 8 instructions in less than 4 cycles.*

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

*[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)*

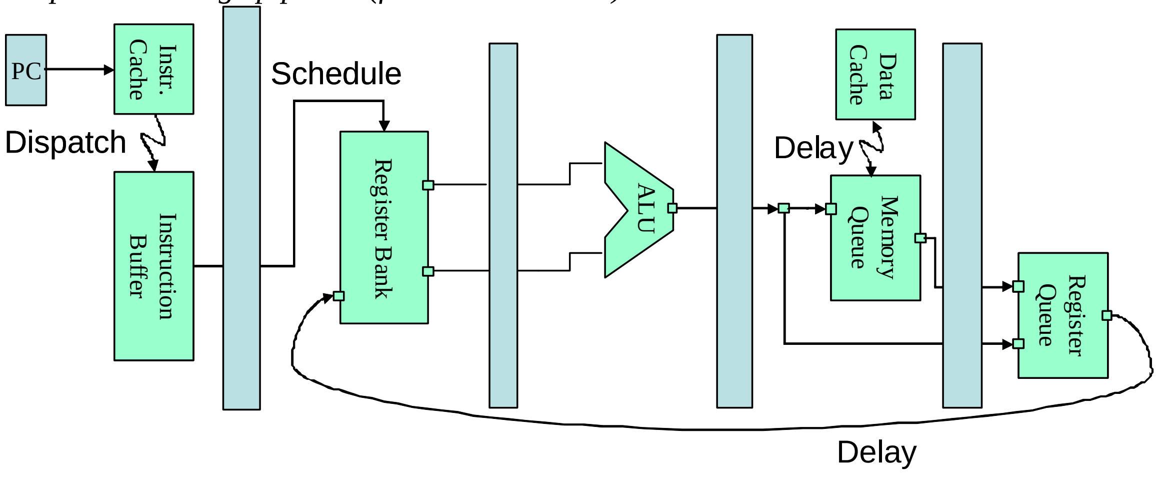
*Hardware*
*Pros: Legacy compatibility, Simple compiler, application independent*
*Cons: Hardware complexity affects freq. power, etc, small window of instructions, precise interruptions require complex hardware.*

e) What needs to be changed in an in-order processor to transform it into an out-of-order processor? (4 marks)

*[Bookwork]*
***Instruction Issue: (2 marks)***
*Add a **dynamic scheduling unit** able to detect dependencies and dispatch independent instructions; e.g. Instruction buffer+schedule (as below), scoreboard, reservation stations.*

***Results management: (2 marks)***
*Add some logic to the writing of results to avoid output conflicts (WAW, WAR); e.g. Memory and Register queues (as below), stalling units, or register renaming.*

*Example in a 5-stage pipeline (from the handouts):*



4. **Multithreading / Multicore**

a) Explain cache thrashing and false sharing and give examples of when they can occur. (4 Marks)

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

*Cache trashing are pathological configurations largely affecting the performance of the cache hierarchy due to constant fetching and evicting of cache lines. It can happen:*

*Within a single program when memory addresses being accessed often have the same index. For set associative and direct mapped caches that means they will be replacing each other often, with the subsequent re-fetching. In a multithreaded processor (especially fine-grained and SMT) the same behaviour can happen across threads as they are sharing the processor's cache.*

*False sharing: another form of thrashing where two threads in different cores keep invalidating each other's cache line through different variables which are not meant to be shared but happen to be stored in the same cache line.*

b) Define what is meant by superscalar, multithreading and multicore, three techniques we can use to exploit parallelism in hardware. (6 Marks)

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

*A superscalar processor replicates execution logic to allow issuing (and executing) several instructions per cycle from a single instruction flow (thread). It exploits Instruction Level Parallelism.*

*With multithreading the control and status logic (including register bank and TLBs) are replicated so that instructions from several, typically 2, instruction flows (threads) can be issued but the execution logic is shared among the different threads. It exploits Thread Level Parallelism by sharing the resources of a single processing core. Simultaneous multithreading also exploits ILP by allowing instructions from different threads to be run in parallel in the same processor.*

*Finally, with multicore all the processing logic (control + execution) is replicated so that the different instruction flows are executed independently. It exploits Thread Level Parallelism by duplicating all the core logic.*

c) We are involved in the purchase of a high-performance server for a SME that works on data analytics. We have obtained the following specifications for 3 different configurations:

| | **Config 1** | **Config 2** | **Config 3** |
|---|---|---|---|
| **# of cores** | 8 | 6 | 8 |
| **Multithreading** | No | 4-way fine-grain | 2-way SMT |
| **Superscalar** | 4-way | 4-way | 2-way |
| **Clock freq.** | 1.5 GHz | 2.5 GHz | 3GHz |

Assuming that the 3 are priced at a similar range and that memory and cache systems are similar, which one you will choose (and why – please give numeric answers) if the main concern of the development team was increasing: (8 marks)

1) single-thread peak performance
2) peak IPC of the system (instructions per cycle)
3) peak computing throughput of the system (instructions per second)
4) the total number of hardware threads supported by the system

*Solution: [Problem-solving] - **2 Marks each***
*For single-thread performance we need to multiply the peak IPC (superscalar lanes) by the clock frequency*
*For peak IPC, we need to multiply the number of cores times the maximum number of instructions that can be executed per cycle.*
*For the computing throughput we need to multiply the number of cores, the IPC and the freq.*
*For the number of threads, we multiply the number of cores times the threads per core*

*Results are as follows (best in bold).*

| | *Config 1* | *Config 2* | *Config 3* |
|---|---|---|---|
| *Single-thread* | *4*1.5 = 6Ginstr/s* | ***4*2.5 = 10Ginstr/s*** | *2*3 = 6Ginstr/s* |
| *Peak IPC* | ***8*4 = 32 instr/cycle*** | *6*4 = 24 instr/cycle* | *8*2 = 16 instr/cycle* |
| *Peak Throughput* | *8*4*1.5 = 48Ginstr/s* | ***6*4*2.5 = 60Ginstr/s*** | *8*2*3 = 48 Ginstr/s* |
| *#threads* | *8*1 = 8 threads* | ***6*4 = 24 threads*** | *8*2 = 16 threads* |

d) What is cache coherence in the context of multicore processing systems? Why is it important? (2 marks)

*Solution: [Bookwork] - **1 Mark each***
*In multicore environments the main memory is shared by the different cores. However, if a cache hierarchy is implemented each core will have copies of main memory's data in their local caches. Keeping all these copies up-to-date is what is known as cache coherence.*

*Cache coherency is important because if no precaution is taken, two cores could see different values when accessing the same variable which is against the semantics/principles of shared memory systems which may lead to unstable or erroneous execution.*