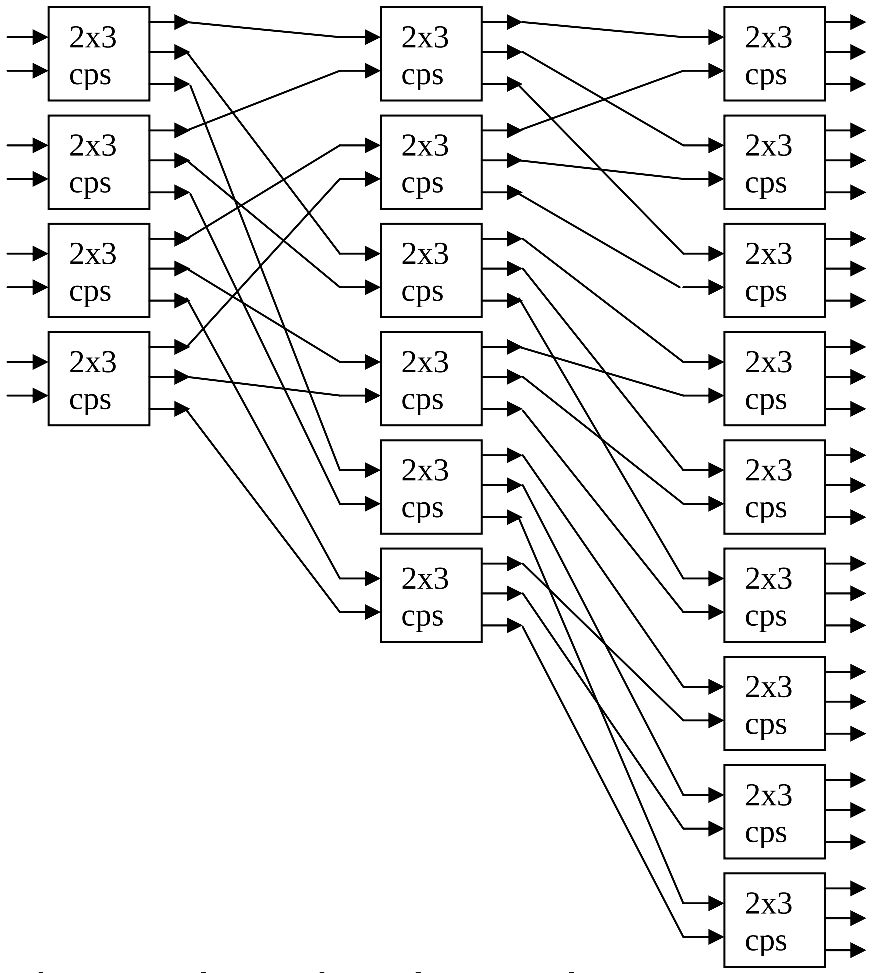
DEPARTMENT OF ELECTRONIC AND ELECTRICAL ENGINEERING

Autumn Semester 2006-2007 (2 hours)

Answers to Advanced Computer Architectures 411/6031, Questions 1…4

1. a. i. *Draw a basic schematic for a Banyan Network where s=2, f=3, and l=2.*



(4)

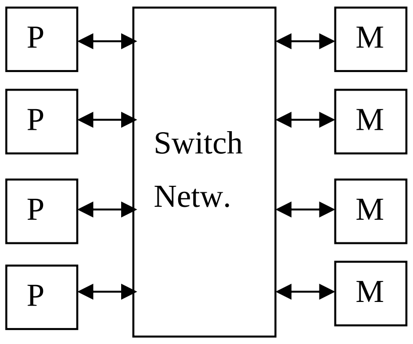
ii. *What approaches can be used to route data inputs to output and what advantages/disadvantages are associated with each approach?*

Answer should include centralised/distributed control - routing information can be carried in header of message as *l*+1, base *f* digits but constitutes an overhead in the message.

Answer should include circuit .v. packet switched. Circuit switched is fast, with well-defined latency but tends to hog links in a blocking switch that affects other communication. Packet switched splits messages up into shorter packets and uses *store-and-forward* to allow messages to be transmitted – need bigger header to allow destination and packet ID to be sent.

(2)

iii. *Draw a schematic showing how a switching network could be used in a tightly-coupled multiprocessor system to connect between the processors and memories.*



(1)

iv. *Estimate the complexity of the Banyan network in* **i.** *relative to the correspondingly sized cross-point switch.*

Complexity of an sxf cps is sf.

Number of inputs to $s{:}f{:}l$ Banyan switch is s^{l+1} and number of outputs is f^{l+1}. Consequently, number of sxf cps in 0th column is $s^{l+1}/s = s^l$, number of outputs from 0th column/inputs to 1st column is s^lf^1 .Number of sxf cps in 1st column is $s^lf^1/s= s^{l-1}f^1$. Outputs to 2nd column is $s^{l-1}f^2$. etc, to lth column. Consequently, total number of sxf cps is:

$\sum_{i=0}^{l} s^{l-i} f^{i}$. Hence complexity is $\sum_{i=0}^{l} s^{l+1-i} f^{i+1}$.

Complexity of corresponding cps $=s^{l+1}f^{l+1}$. Consequently, dividing these two yields:

$$\sum_{i=0}^{l} s^{-i} f^{i-l}$$

For the switch in part **i**. the switch is 8x27 and the cps complexity is 216. The corresponding complexity for the Banyan is (4+6+9)*2*3 = 114. Consequently, the ratio is 0.527 ($2^0 3^{-2}+2^{-1}3^{-1}+2^{-2}3^0 = 0.527$).

(4)

b. *A closely-coupled processor system consists of a number of processor modules connected by a cross-point switch to a number of memory modules. It can be shown that the probability, p_A, that a memory access generated by a processor proceeds without delay is:*

$$p_A = \frac{M}{NRT_{acc}}\left(1-\left(1-\frac{RT_{acc}}{M}\right)^N\right)$$

Identify the meaning of the terms in this equation.

M is the number of memory modules at the output of the cps

N is the number of processors

R is the rate at which each processor is generating independent memory accesses across the cps to the memory

T_{acc} is the time for each memory access.

(1)

A particular processing system is being designed with the following *characteristics:*

- *There are eight identical processors connected via a cross-point switch to four identical memory modules.*
- *Each core processor performs 4x10^8 memory accesses per second and have internal write-back caches put between the core processors and their connection to the cross-point switch. Studies show that for three different possible line lengths, the caches that could be used have the properties shown in* ***Table 1****, where 'Dirty Lines' is the proportion of the lines in the cache that get written to, on average.*

Line Length (words)	***Hit Rate (%)***	***Dirty Lines (%)***
16	*97*	*15*
32	*99*	*10*
64	*98*	*8*

Table 1

- *When a line is transferred to/from memory, it is done as a single block transfer and the time taken to perform a block transfer across the connection network is 30+2*n ns, where n is the number of words transferred in a single block.*

Identify the cache line length that should be chosen, for best performance – ensure that your answer is supported by an analysis.

There are a number of things that need to be looked at to determine the answer. The first is the average length of the memory access (assuming that there is no competition) and this will be basic access time (1/4x10^8) * hit rate + miss rate * transfer time +miss rate * write blocks * block transfer time. From the results in the table, it is clear that the average access time for the line of length 32 seems to be the best. This means that the rate at which the processor is actually issuing memory accesses, in the absence of contention, is not 4x10^8 but is the reciprocal of average access time. However, we have to consider the effect of contention for memory and to solve this we have to use the equation above but we need to determine what R and what T_{acc} are and this depends on the cache line length/characteristics. Only those accesses that miss in the cache go across the cps. So for the first case only 3% of the 2.19x10^8 accesses per second go across the cps = 6.9x10^6. However, 15% of these misses will need to replace a block because it has been changed. Consequently, there will by 1.15x this number of block transfers so R = 7.56x10^6. In this case, the length of block transfer is T_{acc} = 30+2*16=62ns. From this, knowing N=8, and M=4 we can calculate the probability that a processor's access will be honoured.

Line Length (words)	***Ave Acc. (ns)***	***Access Rate (M)***	***R***	***T_{acc}***	***p_A***
16	4.56	219.3	7.56x10^6	62x10^{-9}	0.67
32	3.5	285.7	3.13x10^6	94x10^{-9}	0.78
64	5.86	170.6	3.68x10^6	158x10^{-9}	0.61

Clearly, the line length of 32 has the greatest probability of its memory transfers being honoured and this, in conjunction with the better access time, would lead us to choose the line length of 32. **(8)**

In fact, and this is beyond the scope of what is being asked, to first order, we might assume that those accesses that fail due to contention result in one further transfer (an extra (1-P_A)x transfers). This, in turn, increases the effective transfer time for blocks by a corresponding amount and this increases the average access time further, reducing R and P_A until equilibrium is reached. Based on the values use and as confirmation, this yields net access rates of 192, 268, and 143 million accesses per second for the three cases.

2. *A pipelined system, as shown in* ***Figure 2****, consists of four, interconnected processing blocks which are all of approximately equal complexity and cost.*

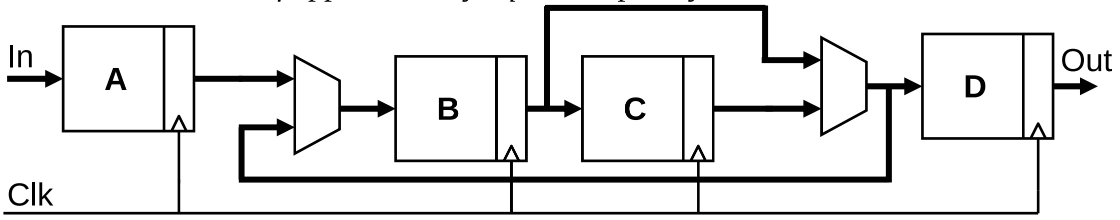


Figure 2

The algorithm applied to the sequence of input data is:

$$In \rightarrow A \rightarrow B \rightarrow C \rightarrow B \rightarrow B \rightarrow C \rightarrow B \rightarrow D \rightarrow Out$$

a. *For this set of operations:*

i. *Draw the reservation table;*

Time	A	B	C	D
0	*			
1		*		
2			*	
3		*		
4		*		
5			*	
6		*		
7				*

(4)

ii. *Determine the value of the collision vector;*

10111

iii. *Calculate the throughput of the pipeline in datum/clock cycle;*

(2)

Time	A	B	C	D		A	B	C	D
0	D_1					D_1			
1		D_1					D_1		
2			D_1					D_1	
3		D_1					D_1		
4	D_2	D_1					D_1		
5		D_2	D_1					D_1	
6		D_1	D_2			D_2	D_1		
7		D_2		D_1			D_2		D_1
8	D_3	D_2						D_2	
9		D_3	D_2				D_2		
10		D_2	D_3				D_2		
11		D_3		D_2				D_2	
12		D_3				D_3	D_2		
13			D_3				D_3		D_2

Based on the left hand side, one datum every 4 clock cycles = 0.25 datum / clock. However, at time=6 D1 is in B and D2 is in C. Unfortunately, at time = 7, B goes to D whilst C goes to B and the data collides in the multiplexer. However, I will accept this as an answer. I would also accept the answer on the right hand side which does not let the data collide. In this case, one datum every 6 clock cycles = 0.167 datum/clock.

iv. *Calculate the utilisation of each of the processing blocks* **(2)**

A=25%, B=100%, C=50%, D=25% or

A=16.6%, B=66.6%, C=33.3%, D=16.6%

(2)

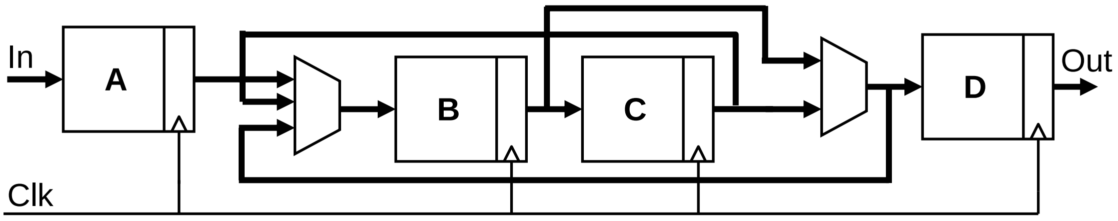
b. *How do you recognise that there are no simple cosmetic changes (e.g. only adding a register) that might be made to the design to improve performance*

The utilisation of any processor is 100%. This means that without a substantial change to the functional behaviour of the system, performance cannot improve because the processor is a bottleneck: if it is fully utilised then there is no possibility of improving throughput. **(2)**

c. *You recognise that there is a simple functional change to the architecture of the pipeline that you can make to improve performance.*

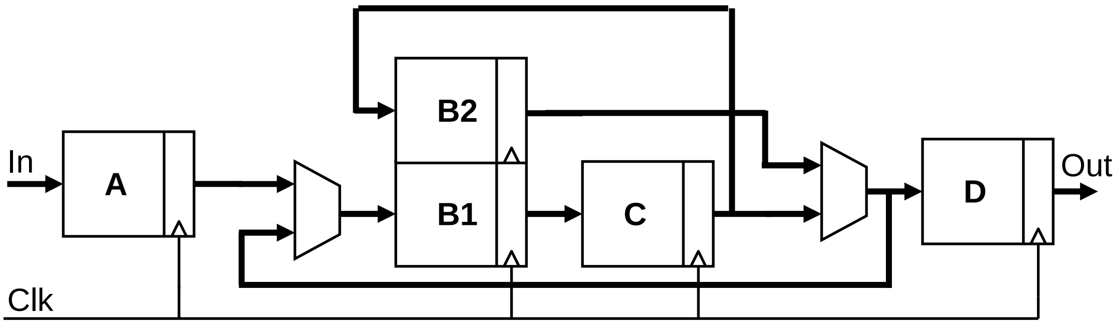
i. *What is this change?*

Depending on the answer in part **a**, there are various things that might be done. Firstly, the multiplexer can be augmented to allow more overlap.



In this case, C can feedback to B whilst B can output to D. This would improve the performance from 0.16datum/clock to 0.25datum/clock at which point, B would be saturated and performance could be improved no further without a more substantive change, as asked for.

A more substantive change, with significant improvement is most easily accomplished by separating out the B tasks that bypass/ do not bypass C. This has the added advantage that the multiplexer at the input side can be simplified to have only two inputs because now there is never any conflict because C is connected directly to B2.



(4)

ii. *How does this change affect throughput?*

Now the reservation table would be:

Time	**A**	**B1**	**B2**	**C**	**D**
0	D_1				
1	D_2	D_1			
2	D_3	D_2		D_1	
3		D_3	D_1	D_2	
4		D_1	D_2	D_3	
5		D_2	D_3	D_1	
6	D_4	D_3	D_1	D_2	
7	D_5	D_4	D_2	D_3	D_1
8	D_6	D_5	D_3	D_4	D_2
9		D_6	D_4	D_5	D_3
10		D_4	D_5	D_6	
11		D_5	D_6	D_4	

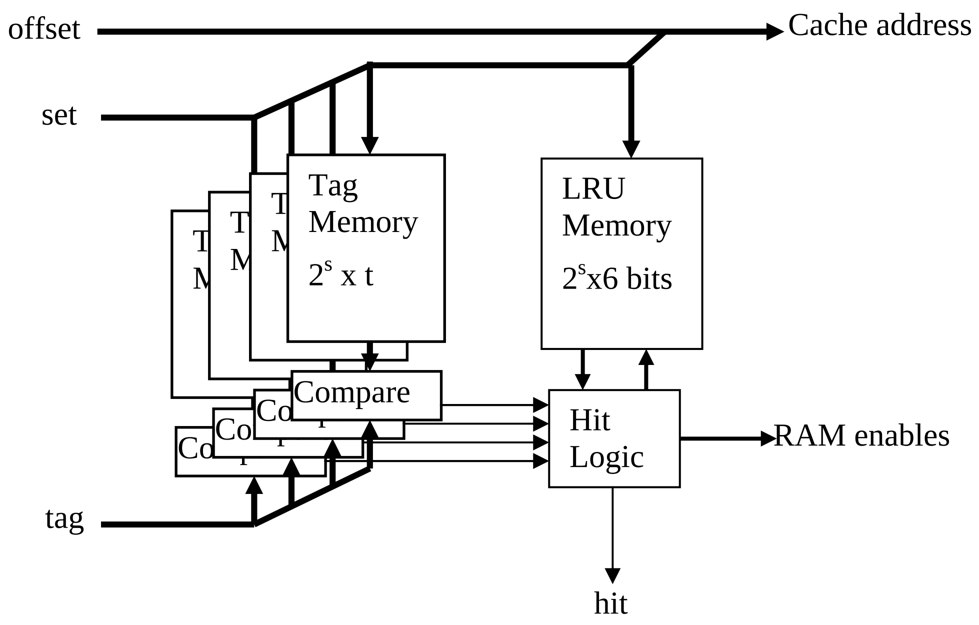
And the throughput is 3 datum every 6 clock cycles = 0.5 datum/clock. That is the performance has been doubled.

iii. *What is the cost/benefit associated with the change?*

Assuming that the original cost was 4 and the new cost is 5 (all the blocks are of equal complexity/cost) and the old performance was 0.25 and the new is 0.5, the *cost/benefit* = 5/4 * 0.25/0.5 = 0.625. This is a considerable improvement, relative to the cost. **(2)**

(2)

3. **a.** *Draw a schematic diagram for a 4-way set associative cache system that uses LRU as the replacement policy, identifying how it operates.*



There are 4 entries in each set and so there is a parallel/4 way lookup of the tags for the accessed set against the tag in the address. The 4 results are used to control the hit an enables for the RAM banks (where the cached data is actually stored). The address to the cache RAMs is merely set:offset. The hit logic also updates the LRU entry for the accessed set. With 4 entries per set, there is a 2 bit ID for the LRU entry and a queue of 3 entries is needed to maintain the LRU item. Consequently, 6 bits per entry are needed. **(6)**

b. *The cache in part* ***a.*** *is 16Kwords in size and is organised as lines that are 16 words long. The memory address (for accessing words) issued by the processor is 32 bits long. Estimate the number of bits of memory used in the controller.*

The value of offset is 4 bits, with 16 words per line and there are 1Klines in the cache (16Kwords in the cache and 16 words per line). If there are 4 entries per set then there are 256 sets. Set is, therefore, 8 bits long and tag must be 20 bits long. Consequently, the tag memory must be 4 x 256 x 20 bits = 20480 bits. The LRU memory must be 256 x 6 = 1536 bits. Consequently, the total is 22016 bits. **(2)**

c. *A set-associative cache is controlled synchronously using a 1GHz clock and the basic access time for the memory used to construct the cache is one clock cycle. On a miss the memory access is posted to a reorder buffer (connected to the external memory) with read-around-write capability. The probability of a memory cycle being a memory read is 0.7, the hit rate of the cache is 0.85 and the penalty on a cache miss is 30ns (this figure subsumes all of the cost of replacing a block within the cache on a miss).*

i. *Estimate, simply, the effective access time of the cache.*

For a set-associative cache a read costs 1 clock cycle and a write costs 2 clock cycles. Consequently, the average access time on a hit is 10^{-9}*0.7+2*10^{-9}*0.3 = 1.3x10^{-9}. However, there is a hit only 85% of the time. On a miss, the memory accesses are posted to the buffer BUT the processor only has to wait for the reads, that is, the processor posts the writes and continues. Consequently, the true cost of a memory access should be $0.85*1.3x10^{-9}+0.15*0.7*3x10^{-8}=$ $4.255x10^{-9}$. This is not the whole story, however. Writes do not affect the performance as long as the total memory bandwidth between the reorder buffer and the memory is not exceeded. If this is the situation then the average access time is 4.255ns and 15% of these accesses result in a data transfer between the reorder buffer and memory – each costing 30ns. That is the time available for each transfer is 4.255/0.15 = 28.37ns. Consequently, the performance will be limited by the throughput that the reorder buffer/memory interface is capable of running at. That is $30x10^{-9}*0.15 = 4.5$ns. **(10)**

ii. *Does it matter how big the reorder buffer is?*

No, the issue identified here is that the average capacity of the reorder buffer to transfer the data required is the limit. Increasing the size does nothing to improve its long-term, average characteristics, it merely provides the ability to deal with short-term fluctuations. **(2)**

4. a. i. *What is interleaving in the context of a memory system and why does it help?*

Interleaving is where memory banks are operated in parallel, being selected on the basis on sub-sets of the address word. In a multiprocessor system, it separates banks so that more than one access can proceed in parallel. **(2)**

ii. *Distinguish between high-, low-, and mixed-order interleaving, explaining where they might be used and their advantages/disadvantages.*

High-order interleaving uses the high-order address bits to distinguish between separate banks forming disjoint memory blocks – best used where multiprocessors tend to access generally separate areas and there is little data sharing.

Low-order interleaving uses the low-order address bits to distinguish between separate banks forming overlapped, interleaved blocks – best uses where multiprocessor share a lot of data although it does create some contention.

Mixed-order uses a combination of high- and low-order interleaving to provide the benefits of both – it can, however, result in increased numbers of parallel blocks and this increases system complexity **(2)**

iii. *Can interleaving help the performance of a uni-processor system and, if so, how?*

Low order interleaving allows a number of adjacent words to be accessed in parallel reducing the throughput delay during block transfers. **(2)**

iv. *How can home memory modules be used to reduce the cost and/or traffic on a switching network in a multiprocessor?*

A home memory module is a dual-ported memory module one port of which is accessed directly by a module and the other comes from the interconntection network. The assumption is that the home module tends to direct most of its accesses to its home memory module but can access other memory modules via the interconnection network at a greater cost. Generally allows the IN to be lower performance for the same overall system performance. **(2)**

v. *What problem might exist in a vector processor relating to interleaving (ensure that your answer describes stride)?*

Low-order interleaving can be used to accelerate vector (i.e. block transfers) from the innermost dimension of a matrix, where the data is stored contiguously. However, but for any other dimension where adjacent matrix elements along that dimension are stored at a distance *stride* away in linear memory addressing terms, the interleaving does not help to accelerate the transfers. **(2)**

vi. *What is the difference between static and dynamic coherence in a multiprocessor system and what problem do the methods attempt to overcome?*

Coherence relates to the issue of ensuring that shared data is up-to-date when it is read/written by processors in a multiprocessing system using caches. Static coherence forces shared data to be uncacheable and finesses the problem at a cost in performance (semaphores are still needed to enforce ownership of data).

Dynamic coherence uses bus snooping and global states applied to data (e.g. RO, EX, RW) to manage ownership and the right to alter data held in caches. **(2)**

vii. *What is indivisibility when applied to memory accessing.*

Indivisibility forces a sequence of operations applied to a memory location (e.g. read followed by write) to be performed atomically, that is, without any other access to the memory location splitting them. **(2)**

viii. *What basic function within an operating system does indivisible memory accessing support*

Semaphores, used to manage shared resources, for example, are dependent upon being able to force the data, forming the semaphore, to be accessed indivisibly. **(2)**

ix. *Identify how a processor implements indivisible memory accessing in a uni-processor system. What basic part of a 'normal' processor's behaviour leads to this requirement for indivisible memory accessing?*

The basic requirement in a uni-processor system is for a basic instruction that allows read-modify-write e.g. test-and-set, or exchange. This requirement stems from interrupts that can separate two contiguous instructions but not parts of an individual instruction (disabling interrupts can work but is much less efficient and may cause other problems). **(2)**

x. *Identify how this method must be enhanced if indivisible memory accessing is to be supported in a closely-coupled multi-processing system.*

In a multiprocessor system using shared memory, test-and-set in not sufficient because the read can be separated from the following write by a memory access from another processor. Consequently, a lock mechanism which allows a memory access to be accompanied by a request to lock out the memory, memory bank, page, block, or location to other processors is required. Typically, a subsequent instruction (not necessarily the next) would be accompanied by an unlock request. **(2)**

NLS / NA