Department of Computer Science
PhD Qualifying Examination
Computer Architecture
Fall, 1997

1. (15 points)

Describe the relative advantages and disadvantages of the following network topologies: crossbar, bus, ring, grid, and hypercube.

Type	*Bandwidth*	*Latency*	*Cost*
`Crossbar`	*O(n)*	*O(1)*	*O*(n^2)
`Bus`	*O(1)*	*O(1)*	*O(n)*
`Ring`	*O(n)*	*O(n)*	*O(n)*
`Grid`	*O(n)*	*O*($\sqrt{n}$)	*O(n)*
`Hypercube`	*O(n)*	*O(log n)*	*O(n log n)*

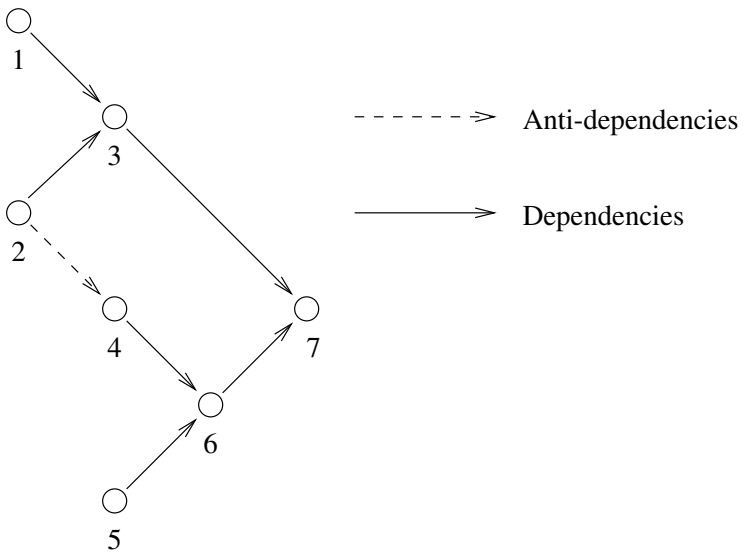
So, the crossbar has the best possible bandwidth and latency, but is very expensive. The bus has the best possible latency and cost, but very poor bandwidth. The ring exchanges latency for bandwidth; it has the best possible bandwidth, but very poor latency. The grid is theoretically superior to either the ring or bus (it has the good bandwidth of a ring, and while its latency is not as good as a bus, it is still much better than a ring). The hypercube is another compromise; its latency is better than that of a grid, but its cost is higher.

2. (40 points)

This question is based on considering the execution of the following fragment of code. Assume the first operand in each instruction is the result (so the first instruction shown adds the contents of register `r1` to the contents of `r2` and puts the result in `r3`).

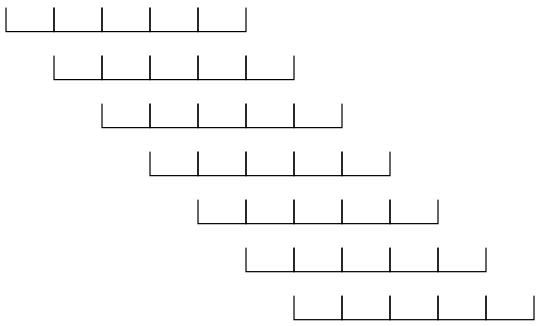
```
1.  add   r3,   r1,   r2
2.  add   r6,   r4,   r5
3.  add   r7,   r3,   r6
4.  add   r10,  r8,   r9
5.  add   r13,  r11,  r12
6.  add   r14,  r10,  r13
7.  add   r15,  r7,   r14
```

A. Draw a graph showing the data dependencies and antidependencies in the code.



B. Suppose the code were executed on an implementation with a five-stage pipeline (fetch, decode/read registers, alu, memory operation, write register) and as much forwarding as possible between stages. How many cycles would it take to execute, from the fetch of the first instruction to writing the result of the last instruction into its register?

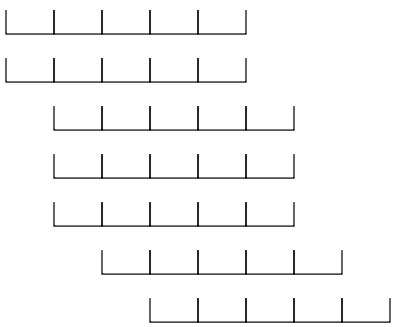
```
add   r3, r1, r2
add   r6, r4, r5
add   r7, r3, r6
add   r10,r8, r9
add   f13,r11,r12
add   r14,r10,r13
add   r15,r7, r14
```



The total is 11 cycles.

C. Now suppose it were executed on a superscalar implementation able to issue up to 8 instructions on a cycle (subject to dependencies, of course), but otherwise similar to that in Part B. Assume that if an instruction has to be delayed a cycle due to dependencies, all following instructions must also be delayed a cycle. Now how many cycles will it take?

```
add   r3, r1, r2
add   r6, r4, r5
add   r7, r3, r6
add   r10,r8, r9
add   f13,r11,r12
add   r14,r10,r13
add   r15,r7, r14
```



This time, the total is eight cycles.

D. If it were executed on an implementation like that in Part C, but with instruction reordering, how many cycles would it take?

```
add   r3, r1, r2     |__|__|__|__|__|
add   r6, r4, r5     |__|__|__|__|__|
add   r7, r3, r6        |__|__|__|__|__|
add   r10,r8, r9     |__|__|__|__|__|
add   f13,r11,r12    |__|__|__|__|__|
add   r14,r10,r13       |__|__|__|__|__|
add   r15,r7, r14          |__|__|__|__|__|
```

The total is now seven cycles.

3. (20 points)

Suppose you had a bus-based multiple-processor computer using a snoopy invalidation-based cache coherence protocol. Initially, memory block M is not valid in any cache.

Describe any changes to the state of the memory block in the different caches after each of the following transactions:

A. Processor P1 reads M
M is now clean in P1

B. Processor P2 reads M
M is now clean in P1 and P2

C. Processor P1 writes M
M is now dirty in P1, invalid in P2, and invalid in memory

D. Processor P3 writes M
M is now dirty in P3, invalid in P1

E. A DMA device transfers some data to M.
M is now valid in memory only — a return to the original situation.

5. (25 points)

In a word-addressable computer system, the latency of memory accesses is L and a word can be transferred between memory and cache in time T (note: the latency L includes the time T required to transfer the first word of fa block). A block of memory consists of B words. In order to reduce the penalty due to misses, we can adopt one of two strategies:

Early restart — The block is fetched in order and as soon as the required word arrives, it is sent to CPU.

Critical word first — The required word is fetched first and sent to CPU and the rest of the block is fetched after that.

Assume that data accesses are random and ignore branching effects in programs (i.e. assume that programs do not have branch instructions). How much better can the Critical word first scheme perform when compared to the Early restart scheme for

A. data misses?
*With both schemes, the first word is fetched in time L. With the critical word first strategy, the required word is the first fetched, so it is available after time L. With the early restart strategy the needed word is equally likely to be any of the B words of the block, so the expected time to transfer the word is L + T*B/2. So the ratio of access times is* $\frac{L}{L + T * B/2}$.

B. instruction misses?

If we are assuming no branches, the miss is always on the first word of the block. There is no difference.

Your answers should be expressed in terms of a ratio of expected access times. Assume that there is no interference between consecutive memory accesses (*ie* memory is always ready to go when a request comes in — you never have to wait for a previous request to finish before you can start).