Problem Set 2
Due Thursday, March 7, 2002

Problems 1, 3, and 4 will be graded. There are 70 points on these problems. Note: You must do all the problems, even the non-graded ones. If you do not do some of them, half as many points as they are worth will be subtracted from your score on the graded problems.

**Problem 1. (25 points)** As described at the end of Lecture 12, there are three ways of organizing the addresses in interleaved memory:

- **Fine interleaving** or low-order interleaving, which distributes the addresses so that consecutive addresses are located within consecutive modules.
- **Coarse interleaving** or high-order interleaving, which distributes the addresses so that each module contains consecutive addresses.
- A combination in which both low- and high-order interleaving are used.

(a) Suppose that a 16-megaword memory is built from 1M-bit chips, so that there are at least $2^{20}$ addresses per module. How many different interleaved organizations can be constructed? For each organization, give the format of an address. This will require you to specify how many bits there are in (up to) 3 fields: the group number, the module number, and the address within the module.

(b) Choosing one of the above organization involves a tradeoff between bandwidth and reliability. Which of your organizations optimizes accesses to consecutive words of memory?

(c) Some of the organizations are less reliable than others, because the failure of a single memory module scatters “holes” of unusable words throughout the address space. Which scheme does not suffer from this problem?

(d) Assume that a program is referencing every third word. For the S-access and C-access strategies, what are the throughputs of each organization of part (a)? Express the throughput in terms of words/access. For example, if a memory is 16-way low-order interleaved, and 16 consecutive words are referenced, the throughput is 16 words/access. If only the even-numbered words are referenced, however, the throughput is only 8 words/access, because only 8 of the 16 words delivered by the memory modules are used.

(e) Repeat part (d), but assume that the program is referencing every second word.

**Problem 2. (30 points)** This problem examines the LRU cache-management implementation using status flip-flops, as covered in Lecture 11. The diagram is reproduced below.
The example design suffers from several faults:

- **Redundant logic.** Note that $C$ appears—
  - twice as a factor in and 1,
  - three times as a factor in and 2, and
  - four times as a factor in and 3.

- **Susceptibility to hazards (races).** It is generally considered bad practice to create a shift register in which all elements are not clocked from the same logic signal. Several different faults can occur:
  - **Value flushed through two stages on one clock.** Consider the case where the block that has just been accessed ($I$) is not $X$. Then the $I$ value should move to the $X$-register, and the value in the $X$-register to the $Y$-register. But if the flip-flops are fast, and and 1 is slow, then the $I$ value will be loaded into the $X$-register, and when and 1 switches, also clocked into the $Y$-register.
  - **Clock chopped off by change of register value used to gate it.** Also consider that and 1 includes the term $(I \neq X)$. In the case above, at the start of the cycle, $I$ does not equal $X$, and and 1 will produce a 1. As soon as the $X$-register loads, $I$ will equal $X$, and and 1 will turn off. This cuts the clock pulse from and 1 short, so it may not reliably load the $Y$-register when it should.
  - **Clock enabled by change of register value used to gate it.** Consider the case where the block that has just been accessed ($I$) is $Y$. Then the $I$ value should move to the $X$-register, and the value in the $X$-register to the $Y$-register. The $Z$- and $W$-registers should not change. At the start of the cycle, $(I \neq Y) = 0$, preventing the $Z$-register from changing. But if the $Y$-register loads from the $X$-register before the end of the cycle, then $(I \neq Y)$ will turn to 1, and the $Z$-register will take on a new, undesired value.

The second hazard, clock-chopping, is avoided if the circuit is implemented with master-slave flip-flops, but the other two problems are only alleviated if the clock period is long.

In this problem, you will derive a hazard-free design that uses a common clock on all stages from the example design.

(a) Write the Boolean equations for the clocks to the $X$, $Y$, $Z$, and $W$-registers for the current design.

(b) The following table shows the correct value in each latch at the end of a cycle. Complete the table.

<table>
<thead>
<tr>
<th>$X_0$:</th>
<th>$I_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_1$:</td>
<td></td>
</tr>
<tr>
<td>$Y_0$:</td>
<td></td>
</tr>
<tr>
<td>$Y_1$:</td>
<td>$(I \neq X) \cdot X_1 \lor (I = X) \cdot Y_1$</td>
</tr>
<tr>
<td>$Z_0$:</td>
<td></td>
</tr>
<tr>
<td>$Z_1$:</td>
<td></td>
</tr>
<tr>
<td>$W_0$:</td>
<td></td>
</tr>
<tr>
<td>$W_1$:</td>
<td></td>
</tr>
</tbody>
</table>
(c) One safe design technique is to clock all the shift-register elements from the same clock, without gating it. To implement this design in that way, extra logic is required between the register elements. In terms of your answer to (b), what are the boolean equations for the data input \( D \) to each element? *Hint:* In the cases where a latch wouldn’t have clocked in the original design, you must assure that it keeps its old value.

(d) Draw the logic diagram.

**Problem 3. (25 points)** Assume you purchase a processor that has a single-level direct-mapped cache. The cache has the properties listed on the left, below. Using your new processor, you would like to run the sort algorithm on the right to sort an array of \( N = 20 \) integers.

- 8 lines
- 2 words/line
- 8-bit addressing
- LRU replacement

```plaintext
for (i=0; i < N-1; i++) {
    for (j = i+1; j < N; j++) {
        if (a[i] > a[j]) {
            c = a[i];
            a[i] = a[j];
            a[j] = c;
        } // if
    } // for j
} // for i
```

(You may assume that \( c \) is a local register that does not need to be loaded or stored in memory.)

(a) Assume that the array resides in contiguous cells in main memory beginning at address “00000000”. The array is initially in reverse order (\( a[0] = 20, a[1] = 29, \ldots, a[19] = 1 \)). How many cache misses will occur before the array is completely sorted?

(b) What is the miss rate of this cache?

(c) To reduce the miss rate, you decide to add an L2 cache to this processor. The L2 cache has the following properties:

- 16 lines
- 2 words/line
- 8-bit addressing
- 2-way set associativity
- LRU replacement

Making the same assumptions as in part (a) (contiguous memory, reverse order), how many L2 misses occur before the array is completely sorted? What is the new miss rate for the multi-level cache?

(d) Compute the total access time necessary to sort the array given the following parameters:

- L1 hit = 1 time unit
- L2 hit = 4 time units
- L2 miss = 10 time units
**Problem 4.** (20 points) In the shared memory diagram at the right, there are 3 distinct processors: P1, P2 and P3. Each processor’s cache reads from shared memory locations X, Y and Z respectively.

Assume the following operations are performed in order using the specific caching strategies listed below, for initial main-memory values \(X = 4\), \(Y = 8\), and \(Z = 3\).

After these operations are performed, for each write strategy (a)–(c), what values for \(X\), \(Y\), and \(Z\) will be in each processor’s cache, and which values will be in shared main memory? If there is no value, just write “–”.

1. P1 reads \(X\), \(Y\), \(Z\)
2. P3 reads \(X\), \(Y\), \(Z\)
3. P1 writes 5 to \(Z\)
4. P1 increments \(Y\)
5. P3 writes 2 to \(Z\)
6. P2 reads \(X\), \(Y\), \(Z\)
7. P1 replaces its \(X\), \(Y\), \(Z\) blocks
8. P2 writes 7 to \(Y\)
9. P3 reads \(X\), \(Y\), \(Z\)
10. P3 writes 1 to \(X\)
11. P2 reads \(X\), \(Y\), \(Z\)
12. P1 reads \(X\), \(Y\), \(Z\)

(a) for a write-back cache

(b) for a write-through cache

(c) for a write-through cache with an invalidation-based snooping protocol.

(d) State whether there would be a difference from the result of part (c) if a write-through cache with an update-based snooping protocol was used. Explain why there is or isn’t a difference between the two