Question 1:

Galaga was a bit confusing, as the highest and lowest numbers were spurious. Note also that the TLB appears a bit bigger than 512 (it is), but 1024 is clearly incorrect because the access time at that point has almost doubled.

Note that some people got bad data, like this test run someone turned in. I believe this is due to other users on the machine; no penalty will be assessed.
Question 2: The page table is shown below. Note that physical pages could have been assigned differently, as long as the page numbers in the PTE were consistent.

Decimal translations:
0x020 = 32
0x024 = 36
0x2FF = 767
0x3BC = 956
0x3BD = 957
0x3BE = 958
0x3BF = 959

Physical addresses:
page 00000 index 0x020: 00000080
0x024: 00000090
0x2FF: 0000BFC
page 00001 index 0x048: 00001120
page 00002 index 0: 00002000
1: 00002004
2: 00002008
page 00003 index 0x3BC: 00002EF0
0x3BD: 00002EF4
0x3BE: 00002EF8
0x3BF: 00002EFC

08048 = 0000 1000 00 00 0100 1000
= 00 0010 0000 00 0100 1000
= 020 048

09000 = 0000 1001 00 00 0000 0000
= 00 0010 0100 00 0000 0000
= 024 000

BFFBC = 1011 1111 11 11 1011 1100
= 10 1111 1111 11 1011 1100
= 2FF 3BC
Question 3: Implement a 3-way synchronizer with (a) semaphores and (b) a monitor. Threads will be released three at a time, and each group of three threads will receive a unique sequence number as the return value of the sync3() function.

(a1) Semaphore S1 = 1 -- mutex
Semaphore S2 = 0 -- wait list
int xid = 0, count = 0

int sync3() {
    int tmp_xid, tmp_count
    down(S1)
    count++
    tmp_xid = xid
    tmp_count = count
    up(S1)
    if (count < 3) {
        down(S2)
    } else {
        xid++
        count = 0
        up(S2)
        up(S2)
    }
    return tmp_xid
}

(a2) Semaphore S1 = 1
Semaphore S2 = 0
int N = 0

int sync3() {
    int tmp_xid, tmp_count
    down(S1)
    N++
    if (N % 3 != 0) {
        up(S1)
        down(S2)
    } else {
        up(S1)
        up(S2)
        up(S2)
    }
    return N / 3
}

You can either use a separate count and sequence number, or combine them with modulo arithmetic.

Note, however, that there is a subtle bug in (a2) – a fourth thread can enter S1 and increment N before threads 1 and 2 get to the line “return N/3”. (A1) avoids this by making a local copy of the variables while holding the lock.

(b1) Monitor {
    Condition C
    int N = 0

    int sync3() {
        N++
        if (N % 3 != 0) {
            wait(C)
        } else {
            broadcast(C)
        }
        return N/3
    }
}

(b2) Monitor {
    Condition C
    int next_xid = 0, count = 0

    int sync3() {
        int tmp = next_xid
        count++
        if (count < 3) {
            while (next_xid <= tmp) {
                wait(C)
            }
        } else {
            count = 0
            next_xid++
            broadcast(C)
        }
        return tmp_xid
    }
}

(b1) has a similar bug; in addition, the fourth thread could be accidentally woken by the broadcast.

(b2) fixes this, again by keeping a local copy of one of the state variables.
Question 4: provide the list of resident pages and hit/miss indications for FIFO, LRU, and OPT on the following access stream: 1 2 3 4 1 2 5 2 1 6 2 3 7 6 1 2 6 2 1 3

<table>
<thead>
<tr>
<th>FIFO</th>
<th>H</th>
<th>H</th>
<th>H</th>
<th>H</th>
<th>H</th>
<th>H</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 3 4 4 4 4</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LRU</th>
<th>H</th>
<th>H</th>
<th>H</th>
<th>H</th>
<th>H</th>
<th>H</th>
<th>H</th>
<th>H</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 3 4 1 2 5</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OPT</th>
<th>H</th>
<th>H</th>
<th>H</th>
<th>H</th>
<th>H</th>
<th>H</th>
<th>H</th>
<th>H</th>
<th>H</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 3 4 4 4 4</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>
|               | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 2