For the following questions, assume the following:
- 32-bit virtual addresses
- 1 MiB pages
- 512 MiB of physical memory with LRU page replacement policy
- Fully associative TLB with 32 entries and an LRU replacement policy

(a) How many entries does a page table contain? __________

(b) How wide is the page table base register? __________

```c
int histogram[MAX_SCORE];
void update_hist(int *scores, int num_scores) {
    for (int i = 0; i < num_scores; i++)
        histogram[scores[i]] += 1;
}
```

Assume that only the code and the two arrays take up memory, ALL of code fits in 1 page, the arrays are page-aligned (start on page boundary), and this is the only process running.

(c) If `update_hist` were called with `num_scores` = 10,
   how many page faults can occur in the worst-case scenario? __________

(d) In the best-case scenario, how many iterations of the loop can occur before a TLB miss?
   You can leave your answer as a product of two numbers. __________

(e) For a particular data set, you know the scores are clustered around fifty different values, but you still observe a high number of TLB misses during `update_hist`. What pre-processing step could help reduce the number of TLB misses? __________
**Question 6:** It’s Virtual Insanity! (13 points, 26 minutes)

Our 32-bit uniprocessor machine has 1 GiB of RAM with 1 KiB pages, a fully-associative TLB that holds 8 entries and uses LRU, and a direct-mapped, write-back data cache with 32 B blocks and 32 slots. The instruction cache is 256 B and fully-associative with 32 B blocks.

a) What is the maximum number of valid entries in the page table for a single process? Answer in IEC.

b) What is the TLB Reach of our system? 
   \[ \text{TLB Reach} = \text{TLB entries} \times \text{page size} \] (i.e. amount of data "reached" by TLB)

Examine the following function. Assume the entire program’s code takes the entirety of one page and \( \text{sizeof(int)} = \text{sizeof(int *)} = 4 \). (This was taught in a 32-bit system)

\[
\text{void addConst(int *ptr, char c) }
\]
\[
\text{ for(int i = 0; i < 1; i+=4) }
\]
\[
\text{ ptr[i] += c; }
\]

c) If \( \text{ptr[]} \) lives in disk and \( \text{ptr[0]} \) is page-aligned, what is the TLB hit rate for data accesses only?

d) If \( \text{ptr[]} \) lives in disk and \( \text{ptr[0]} \) is page-aligned, what fraction of D$ misses are also TLB misses?

e) If \( \text{ptr[0]} \) is in physical memory, what is the minimum value of \( i \) that could cause a page fault?

f) If \( \text{ptr[0]} \) is in physical memory, what is the minimum value of \( i \) that could cause a protection fault?

g) If \( \text{ptr[0]} \) is in physical memory, what is the maximum value of \( i \) that causes the first cache miss in the loop?

h) If \( \text{ptr[0]} \) is in physical memory, what is the maximum value of \( i \) that causes the first TLB miss in the loop? You may leave your answer as a product.
Question 9: Virtual Memory (8 pts)

This election season, the US will computerize the voting system. There were approximately \(2^{27}\) voters in 2012. There are four candidates in the running and so each voter will submit letter A, B, C, or D. The votes are stored in the char votes[] array.

The following loop will count the votes to determine the winner. We are given a 1 MiB byte-addressed machine with 4 MiB of VM and 128 KiB pages. Assume that votes[] and candidates[] are page-aligned and i is stored in a register.

```c
#define NUM_VOTERS 134217728     // 2^27
int candidates[] = {0,0,0,0};    // initialize to 0s
for (int i = 0; i < NUM_VOTERS; i++) {   // Loop 1
    if (votes[i] == 'A') candidates[0]++;
    if (votes[i] == 'B') candidates[1]++;
    if (votes[i] == 'C') candidates[2]++;
    if (votes[i] == 'D') candidates[3]++;
}
```

a) How many bits wide are the following?

<table>
<thead>
<tr>
<th>VPN</th>
<th>Page Offset</th>
<th>PPN</th>
<th>Page Table Base Register</th>
</tr>
</thead>
</table>

b) We are given a fully-associative TLB with 4 entries and LRU replacement policy. One entry is reserved for the Code. In the best case scenario, how many votes will be counted before a TLB miss occurs?

We want to improve our machine by expanding the TLB to hold 8 entries instead of 4. We also revised our for loop, which replaces Loop 1. Assume i and vote are stored in registers.

```c
for (int i = 0; i < NUM_VOTERS; i++) {   // Loop 2
    char vote = votes[i];
    if (vote == 'A') candidates[0]++;
    if (vote == 'B') candidates[1]++;
    if (vote == 'C') candidates[2]++;
    if (vote == 'D') candidates[3]++;
}
```

c) Now how many votes can be counted before a TLB miss in the best case scenario?

d) In the worst case scenario, how many TLB misses would occur if this improved loop ran to completion? In other words, what is the highest number of TLB misses possible when running this loop?