Designing a Cache

What is the basic operation we want our cache to do? (What is the input to the cache, and what is the output of the cache?)

What is the simplest way we could achieve this?

Assuming we have a 32-bit system, how memory efficient is this caching system? (For every bit of physical memory, how many bits are we caching?)

Could we make our cache system more memory efficient? How? (Can we exploit spatial locality?) How memory efficient is the new caching system you came up with? (Do you need to use all 32 bits to specify an address?)

Any address can be anywhere in our cache. As a result, looking through all of our starting addresses to see if there is any match can take a long time. Can we make this search time shorter, while keeping the same number of starting addresses? (Hint: What if we restricted the possible locations of each address?)

We'd like to keep our mechanism for the above question simple (why?), and what's simpler than using bits from the address? Which bits from the address would best fit our needs? (Hint: We want high variability!)

How many bits do we need to tag each of our starting addresses with now?

Oops, we've forgotten one small detail throughout this problem: How do we know if what's in the cache contains actual data? How can we fix this?

Congratulations, you've designed your very own cache! Now let's see what happens if we change some of the parameters.
Let C be the cache size, B the block size, S the set size, and E the associativity (the number of potential starting addresses per set).

How can we relate C, B, S, and E?

What happens if E=1? What happens if S=1? What is the value of E when S=1? What is the range of E?