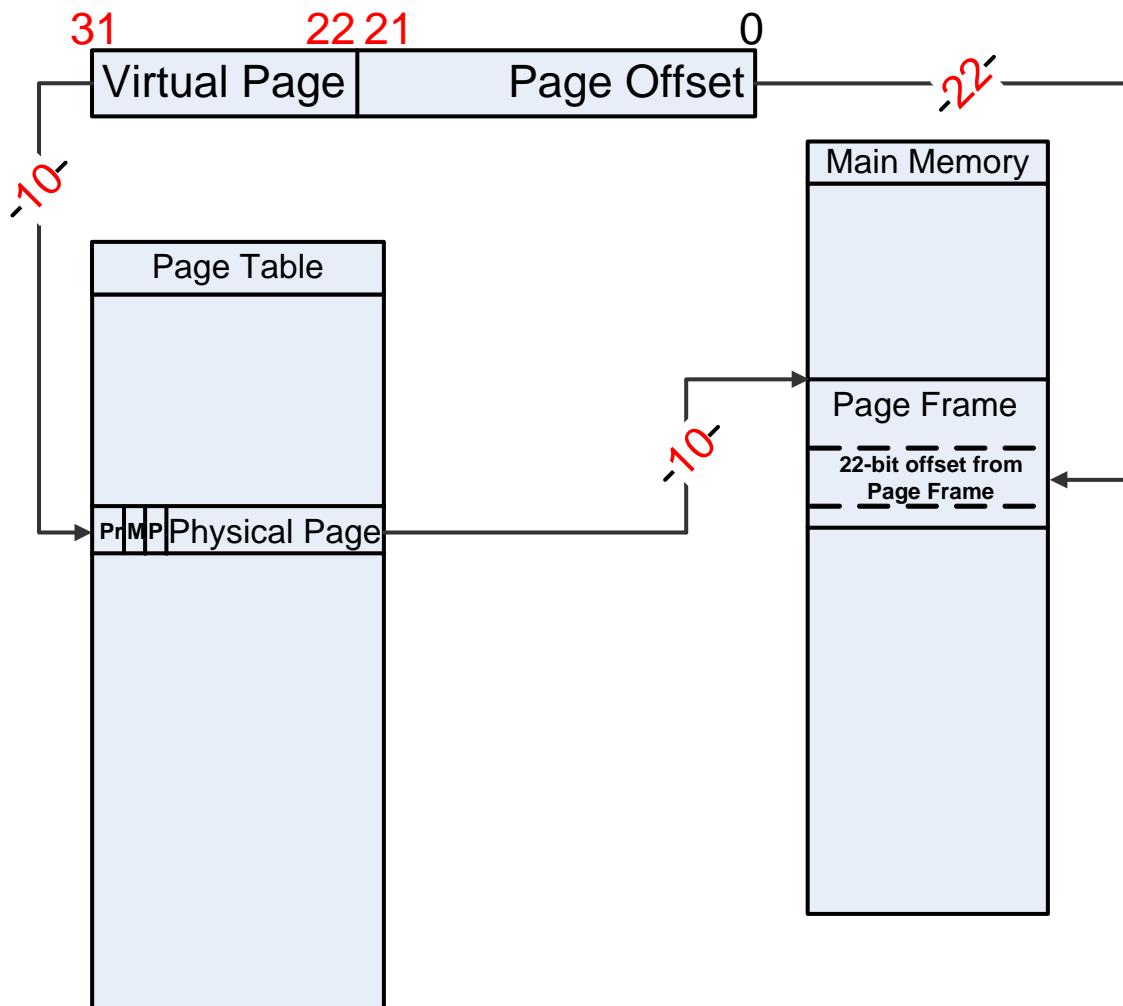


## Notes on the Solution for Week 11 Tutorial

### 1. Paging Scheme with 4MB pages

The Pentium 4 processor offers both a mode for using 4KB page sizes (used by default) and a newer mode for using a page size of 4MB. How would our basic paging diagram, reproduced here from the lecture slides, have to be modified for large page sizes?

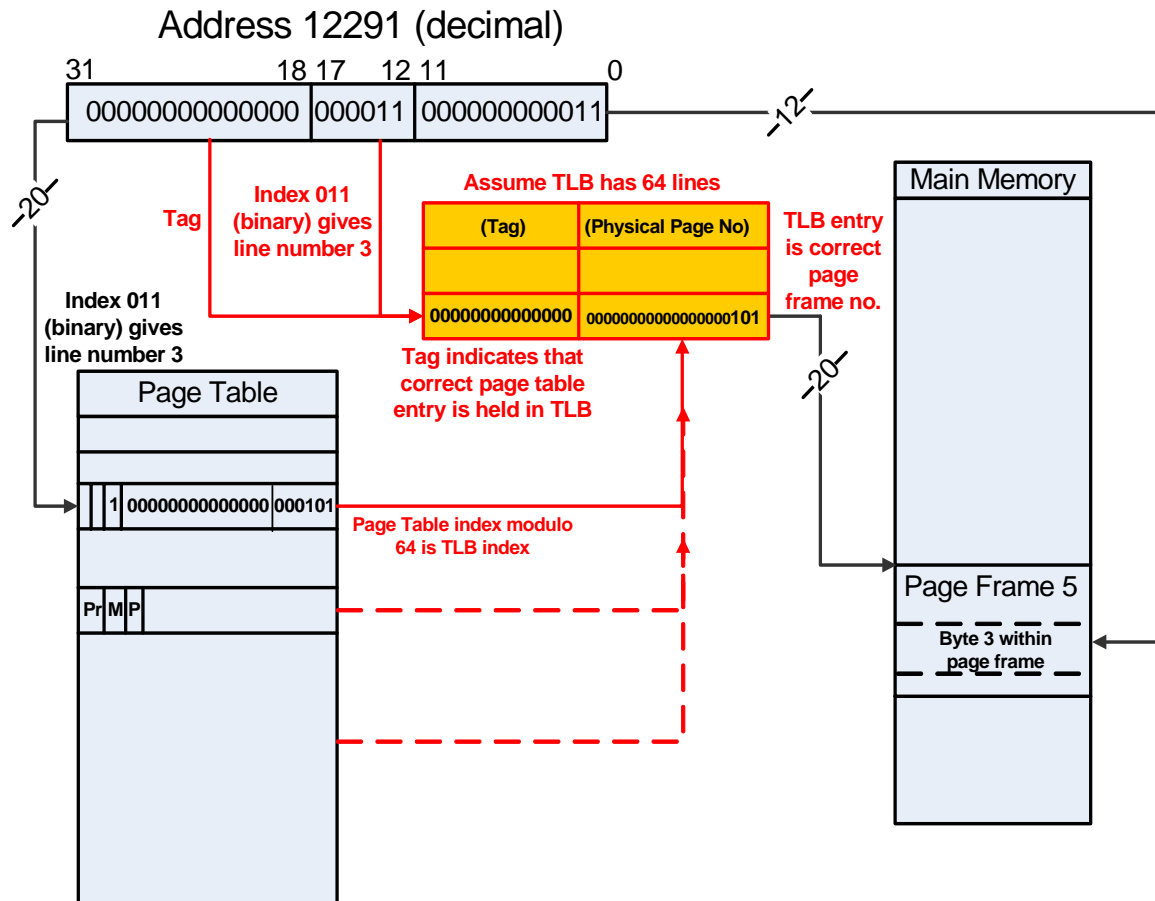


What would the size of the page table be?

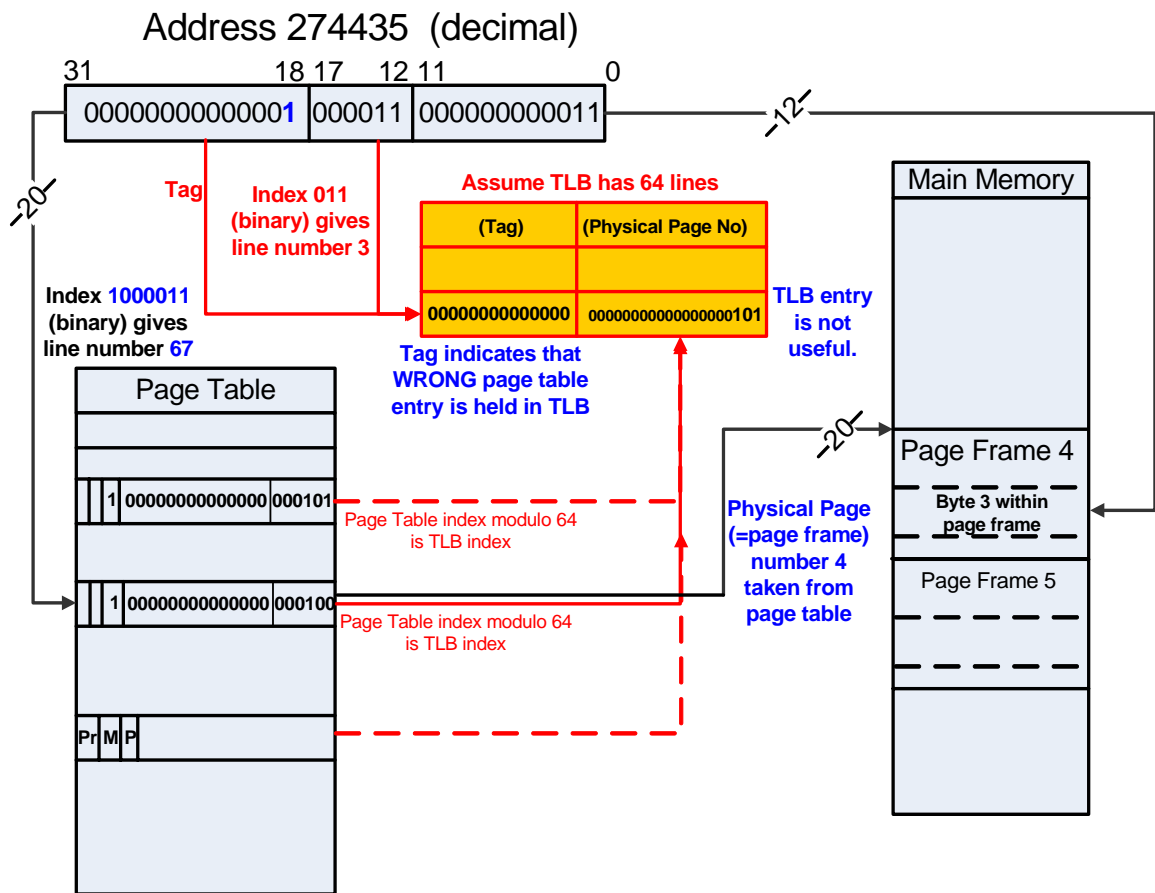
Each line in the page table will consist of a 10-bit physical page number, plus at least four protection bits – assume that we allow 2 bytes (16 bits). There are  $2^{10} = 1K$  page table entries, therefore the total size of the page table will be about 2 KB.

## 2. Worked Paging Example with TLB

Work through the same example as done in the lecture notes (4 KB pages, for decimal address 12291), but this time for paging with TLB, indicating the correct entries for address, page table, TLB and memory.

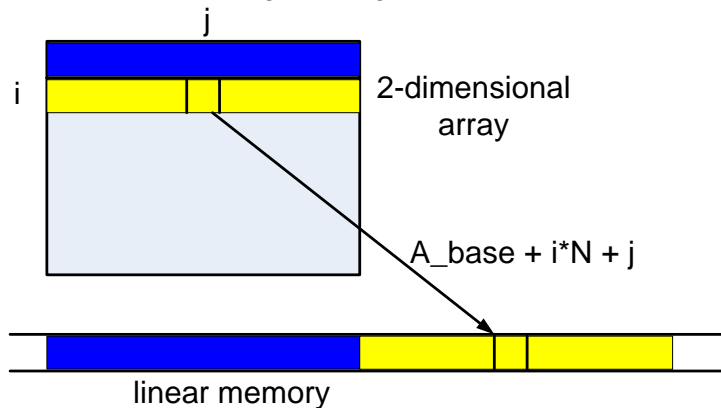


3. Suppose we now next access decimal address 274435 (= 12291 + 4096\*64)? What will happen?



#### 4. TLB Missrates

2-dimensional arrays in C are typically laid out in memory in row-major order. This means that for an  $N \times N$  array, element  $A[i][j]$  is stored at location  $A\_base + i*N + j$ . This is illustrated in the following drawing



What is the expected TLB miss rate on a Pentium 4 when traversing this array in the following two loops?

Loop 1

```
int I, J, a[1024][1024]
for( I = 0; I < 1024; ++I ) {
    for( J = 0; J < 1024; ++J ) {
        a[I][J] = 0;
    }
}
```

Loop 2

```
int I, J, a[1024][1024]
for( I = 0; I < 1024; ++I ) {
    for( J = 0; J < 1024; ++J ) {
        a[J][I] = 0;
    }
}
```

We will assume a 64 entry, direct-mapped TLB (direct mapped means using the scheme I have illustrated in the other examples above, i.e. TLB line number is page table line modulo 64).

Notice that the array  $a$  occupies exactly one page with each row.

**Loop 1** therefore touches the same page 1024 times before moving on to the next page, and does not touch that page again after that. Therefore the TLB miss rate is  $1/1024 =$  approx. 0.09%.

**Loop 2** touches a different page on each iteration of the inner (for-j) loop. The only question is whether there is any chance of those page translations still being in TLB by the time we get round the for-I loop again, when the same set of pages will be touched. The answer is no, since the for-j loop will touch 1024 different pages, but there are only 64 slots in the TLB. Therefore we get a TLB miss on every memory access, the miss rate is 100%.