Quiz

Quiz1（Feb. 27）

Q：Write an essay describing a problem that would benefit from the use of parallel computing. Provide a rough outline of how parallelism would be used. Would you use task- or data-parallelism?

数据并行	任务并行
在相同数据的不同子集上执行相同的操作	对相同或不同的数据执行不同的操作
同步计算	异步计算
加速更多，因为只有一个执行线程在所有数据集上运行	由于每个处理器将在相同或不同的数据集上执行不同的线程或进程，因此加速比较少
并行化的数量与输入数据大小成正比	并行化的数量与要执行的独立任务的数量成正比
为多处理器的负载均衡设计	负载平衡取决于硬件的可用性和调度算法

Quize

Suppose we have two threads, one with id 0 and the other with id 1. Suppose also that each is storing a private variable *my_x*, thread 0's value for *my_x* is 5, and thread 1's is 9. Further, suppose both threads execute the following code:

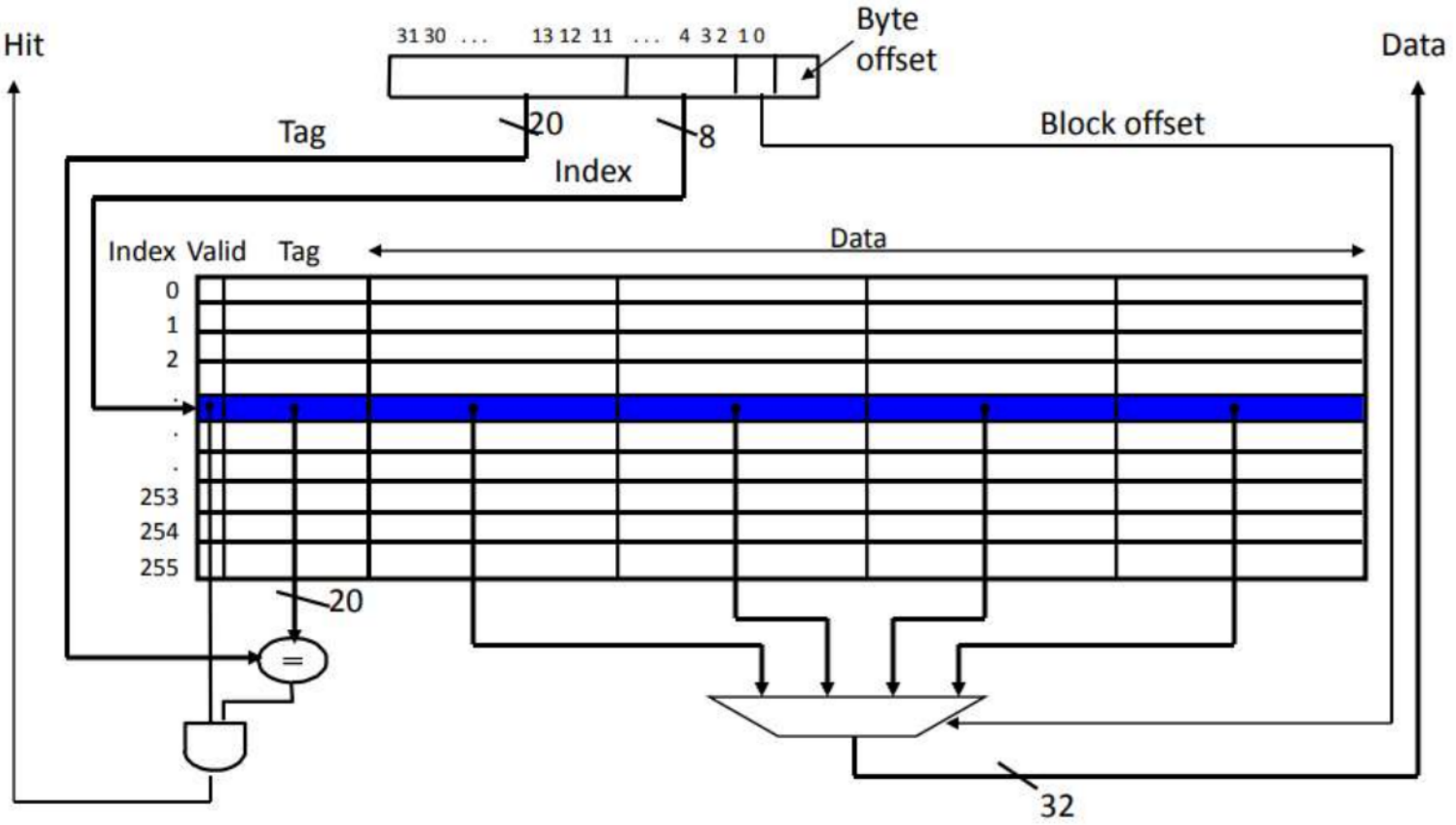
```
printf("Thread %d > my_x = %d\n", id, my_x);
```

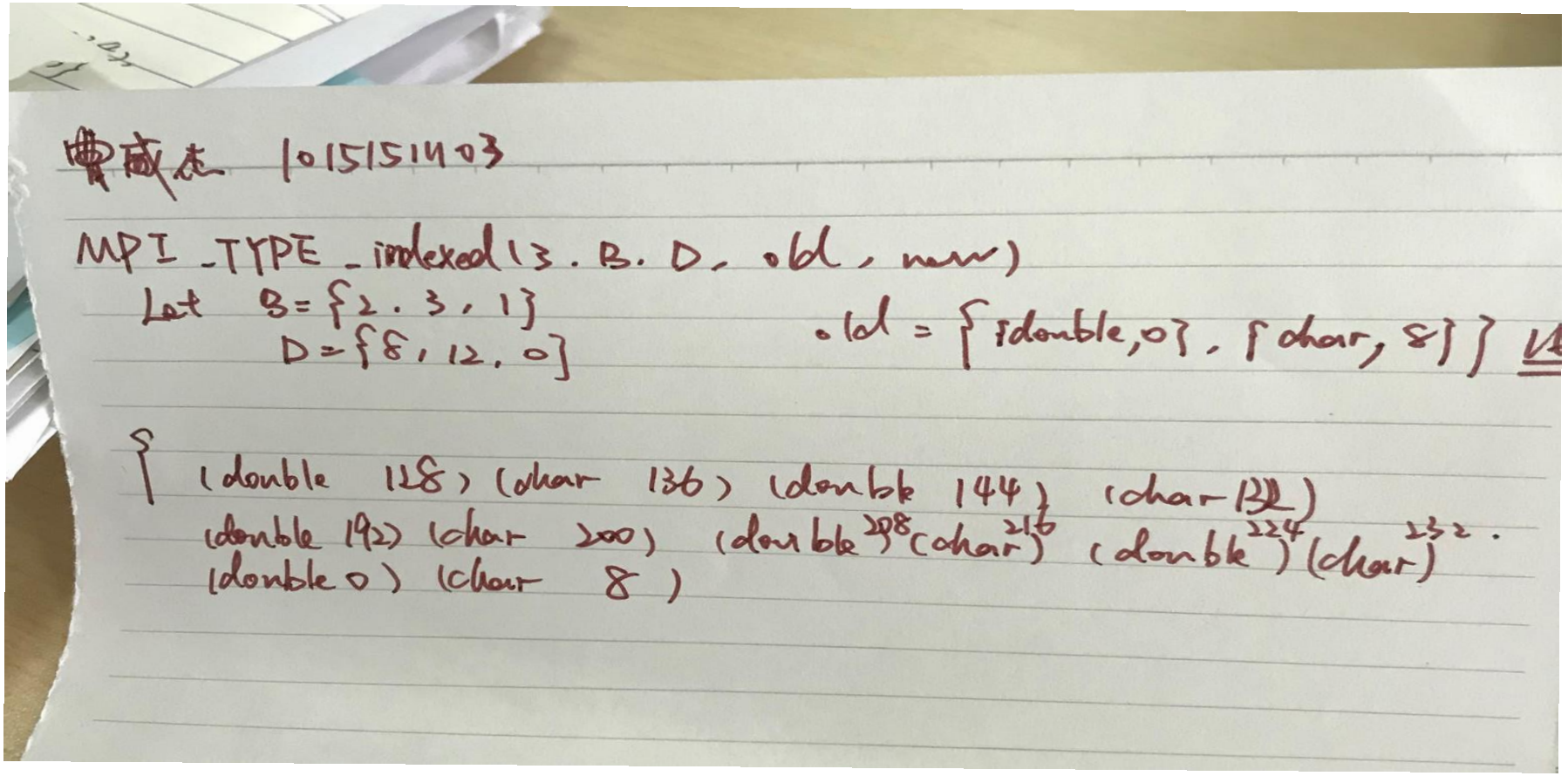
Please list all possible output.

Quiz

Assume there are three small cache, each consisting of four one-word blocks. One cache is fully associative, a second is two-way set-associative, and the third is direct-mapped. Find the number of misses for each cache organization given the following sequence of block addresses: 0, 8, 0, 6, and 8.

- Four words/block, cache size = 1K words




曹威杰 10151514103
MPI_TYPE_indexed(3, B, D, old, new)
Let B={2, 3, 1}
D={8, 12, 0}
old={double, 0}, {char, 8}
{(double 128) (char 136) (double 144) (char 152)
(double 192) (char 200) (double 208) (char 216) (double 224) (char 232).
(double 0) (char 8)}

Homework

C Language

- Hard code

```
/*Get the sample_input.txt*/
file=argv[1];
fp=fopen(file,"r");
fp_result=fopen("output.txt","w");
```

- Clear statement

```
/* Global variables */
int     thread_count;
int     m, n;
double* A;
double* x;
double* y;

/* Serial functions */
void Usage(char* prog_name);
void Read_matrix(char* prompt, double A[], int m, int n);
void Read_vector(char* prompt, double x[], int n);
void Print_matrix(char* title, double A[], int m, int n);
void Print_vector(char* title, double y[], double m);

/* Parallel function */
void *Pth_mat_vect(void* rank);
```

C Language

- Careful with ‘While’

- use these to deal with errors:
- EOF, NULL, etc.

```
while (1)
{

    for (int i = 0; i < m; i++)
        ok[i] = 0;

    c = fgetc(fp);
    while (c != '\n')
    {
        if (c == EOF)
        {
            fclose(fp);
            fclose(fp2);
            return 0;
        }
        c = fgetc(fp);
    }

    fscanf(fp, "%d\n", &m);

    while ((c = fgetc(fp)) != '\n')
        ;
    fscanf(fp, "%d\n", &n);

    while ((c = fgetc(fp)) != '\n')
        ;
    fscanf(fp, "%d\n", &t);

    while ((c = fgetc(fp)) != '\n')
        ;
    for (int i = 0; i < m; i++)
        for (int j = 0; j < n; j++)
            fscanf(fp, "%d", &matrix[i][j]);

    while ((c = fgetc(fp)) != '\n')
        ;
    while ((c = fgetc(fp)) != '\n')
        ;
    for (int i = 0; i < n; i++)
        fscanf(fp, "%d", &v[i]);

    for (int i = 0; i < t; i++)
        pthread_create(&id[i], NULL, f, (void *)i);

    for (int i = 0; i < t; i++)
        pthread_join(id[i], NULL);

    for (int i = 0; i < m; i++)
    {
        while (ok[i] != 1)
            ;
```

C Language

- Better to allocate dynamically

```
double y[1000],x[1000];
double S[1000][1000];
```

```
A = malloc(m*n*sizeof(double));
x = malloc(n*sizeof(double));
y = malloc(m*sizeof(double));
```

- About Debug
- Conditional compiling
- example: #if defined(content)

#if !defined(content)

“-DDEBUG”

```
#  ifdef DEBUG
   Print_digraph();
#  endif
```

Pthreads matrix-vector multiplication

- Two components: Serial and Parallel

```
/* Serial functions */
void Usage(char* prog_name);
void Read_matrix(char* prompt, double A[], int m, int n);
void Read_vector(char* prompt, double x[], int n);
void Print_matrix(char* title, double A[], int m, int n);
void Print_vector(char* title, double y[], double m);

/* Parallel function */
void *Pth_mat_vect(void* rank);
```

MPI merge sort

1. 0: read the input.txt & scatter the data to other processes
2. Each: sort the local array
3. Merge sort
4. Output the result

```
int cmp(const  void *a,const  void  *b)
{

    return *(int *)a-*(int *)b;
}
```

MPI merge sort

- qsort()

```
int cmp(const  void *a,const  void  *b)
{

    return *(int *)a-*(int *)b;
}
```