Lập trình song song với OpenMP

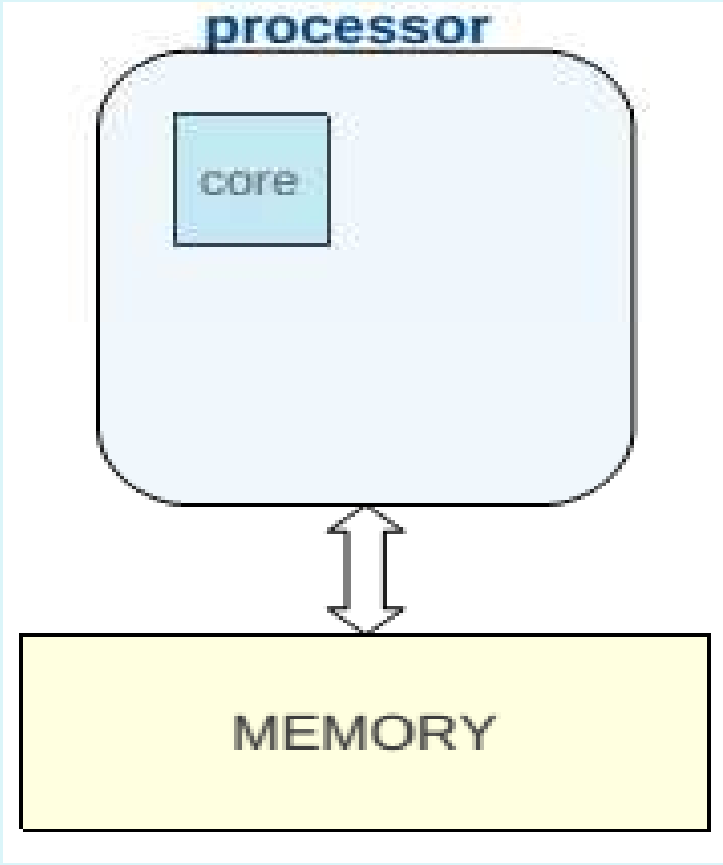
TS. Vũ Văn Thiệu

Nội dung

- Giới thiệu
- OpenMP API
- Biên dịch OpenMP
- OpenMP directives

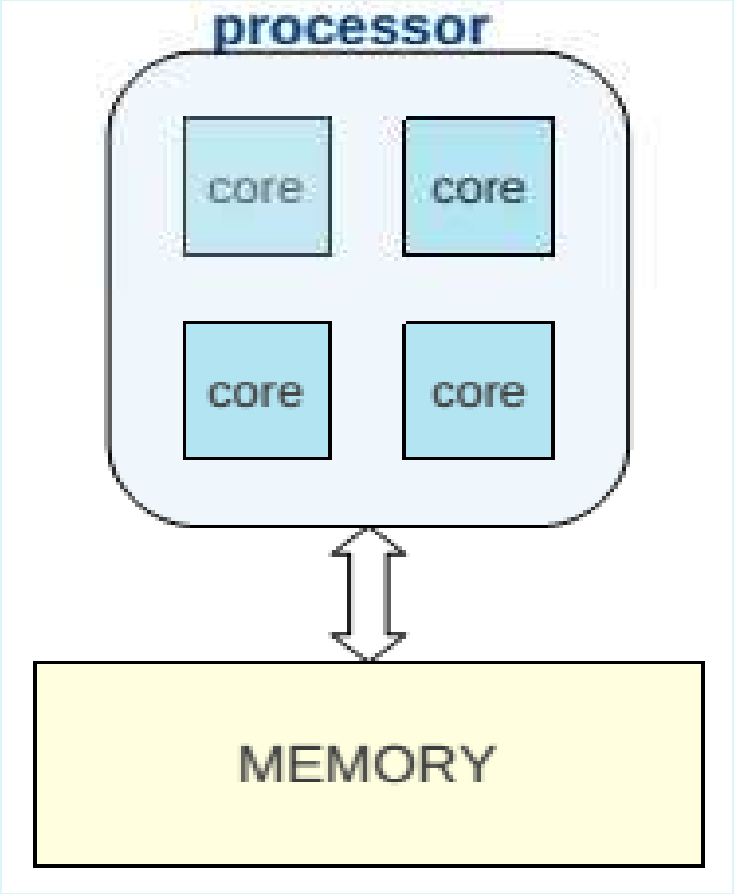
Cấu trúc CPU ban đầu

- Mỗi Processor có một core



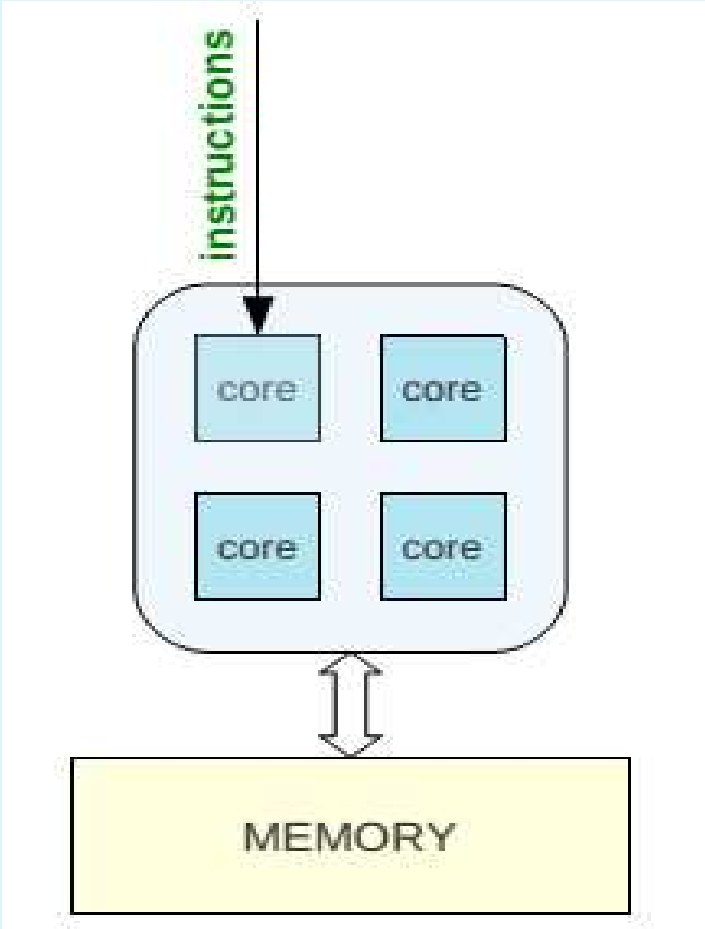
CPU ngày nay

- Mỗi Processor có thể có 2, 4, 8 core
- Nhiều core chia sẻ chung một bộ nhớ



Chương trình tuần tự

- Chương trình được thực hiện trên 1 core
- Các core khác Idle
- Lãng phí tài nguyên hệ thống



OpenMP

- OpenMP là một API cho phép viết ứng dụng song song trên các hệ thống có bộ nhớ chia sẻ bằng C, C++, Fortran
- OpenMP bao gồm:
  - Compiler directives
  - Runtime Library routines
  - Environnent variables

HelloWorld OpenMP

```
#include <omp.h>
#include <stdio.h>
int main() {
   #pragma omp parallel
   printf("Hello from thread %d, nthreads %d\n",
            omp_get_thread_num(),
              omp_get_num_threads());
}
```

HelloWorld OpenMP

- Biên dịch:
  - gcc –fopenmp HelloWorld.c
- Kết quả:

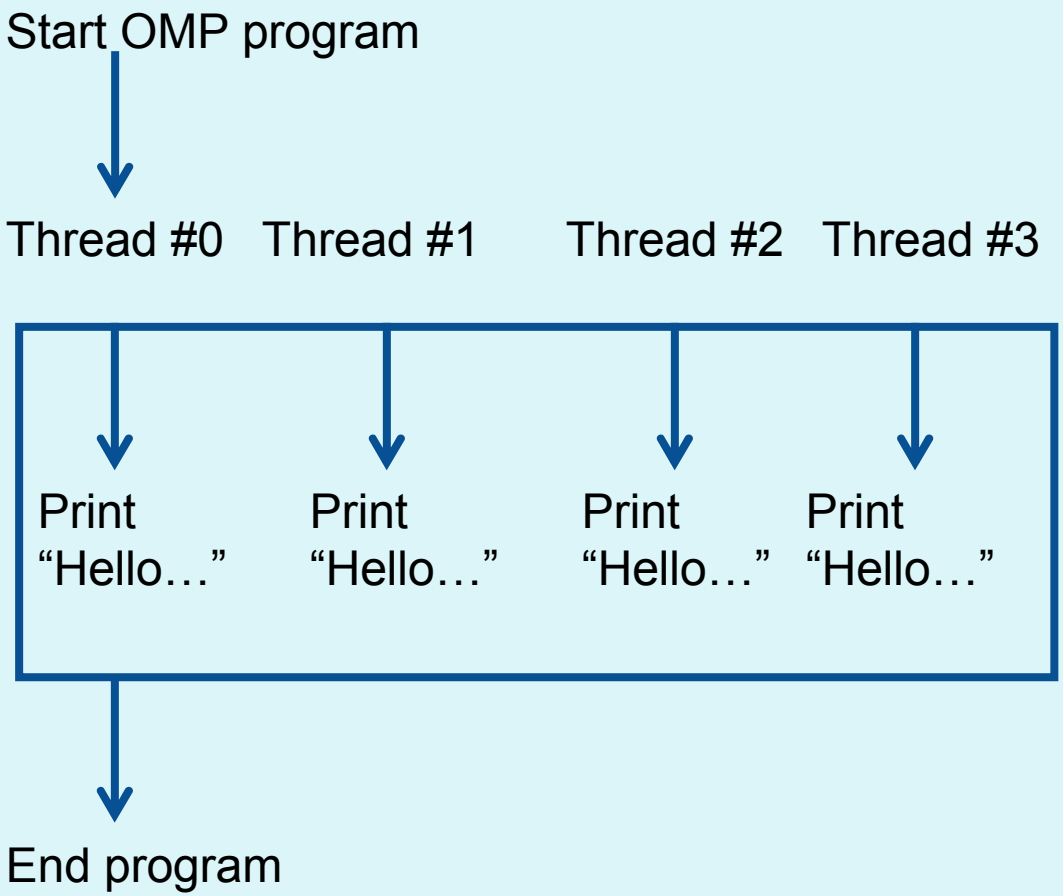
  Hello from thread 0, nthreads 4

  Hello from thread 1, nthreads 4

  Hello from thread 2, nthreads 4

  Hello from thread 3, nthreads 4

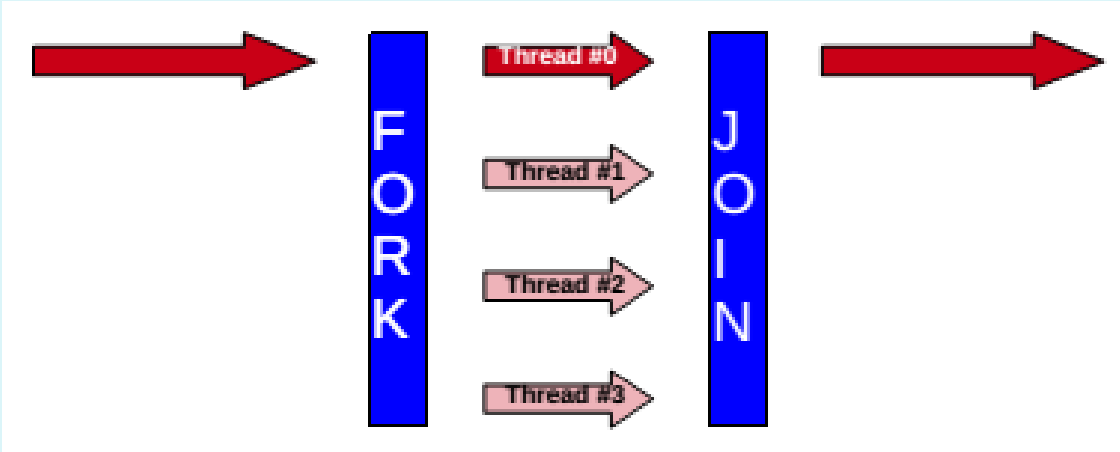
Khi thực hiện chương trình



Trong OMP block, tất cả các thread thực hiện tất cả các instructions

Mô hình Fork/Join

- OpenMP programs start with a single thread; the master thread (Thread #0)
- At start of parallel region master creates team of parallel ”worker” threads (FORK)
- Statements in parallel block are executed in parallel by every thread
- At end of parallel region, all threads synchronize, and join master thread (JOIN)



OpenMP Threads vs Cores

- Thread is independent sequence of execution of program code
  - Block of code with one entry and one exit
- Unrelated to Cores/CPUs
- OpenMP threads are mapped onto physical cores
- Possible to map more than 1 thread on a core
- In practice best to have one-to-one mapping.

OpenMP Threads

- Number of openMP threads can be set using:
  - Environmental variable **OMP_NUM_THREADS**
  - Runtime function **omp_set_num_threads(n)**
- Other useful function to get information about threads:
  - Runtime function **omp_get_num_threads()**
    - Returns number of threads in parallel region
    - Returns 1 if called outside parallel region
  - Runtime function **omp_get_thread_num()**
    - Returns id of thread in team
    - Value between [0,n-1] // where n = #threads
    - Master thread always has id 0

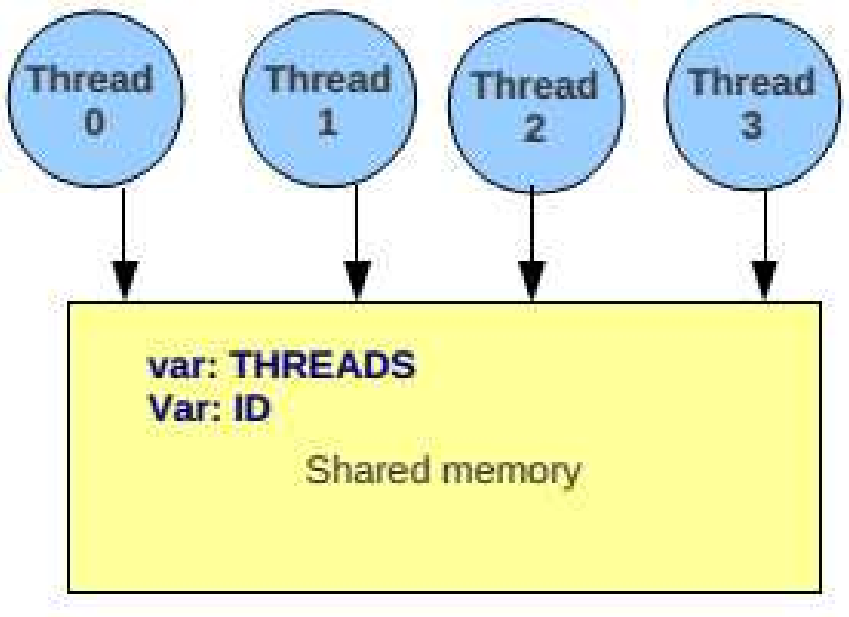
Bài tập 1

- Chuyển chương trình sau sang OpenMP sao cho dòng thông báo “Hello …” được in ra 10 lần

```
#include <stdio.h>
int main() {
    int threads, id;
    threads = 10;
    id = 10;
    printf("Hello from thread %d, nthreads %d\n",id,threads);
    return 0;
}
```

Mô hình Shared Memory

- The memory is (logically) shared by all the cpu's



“All threads try to access the same variable (possibly at the same time). This can lead to a race condition. Different runs of same program might give different results because of race conditions”

Shared and Private Variables

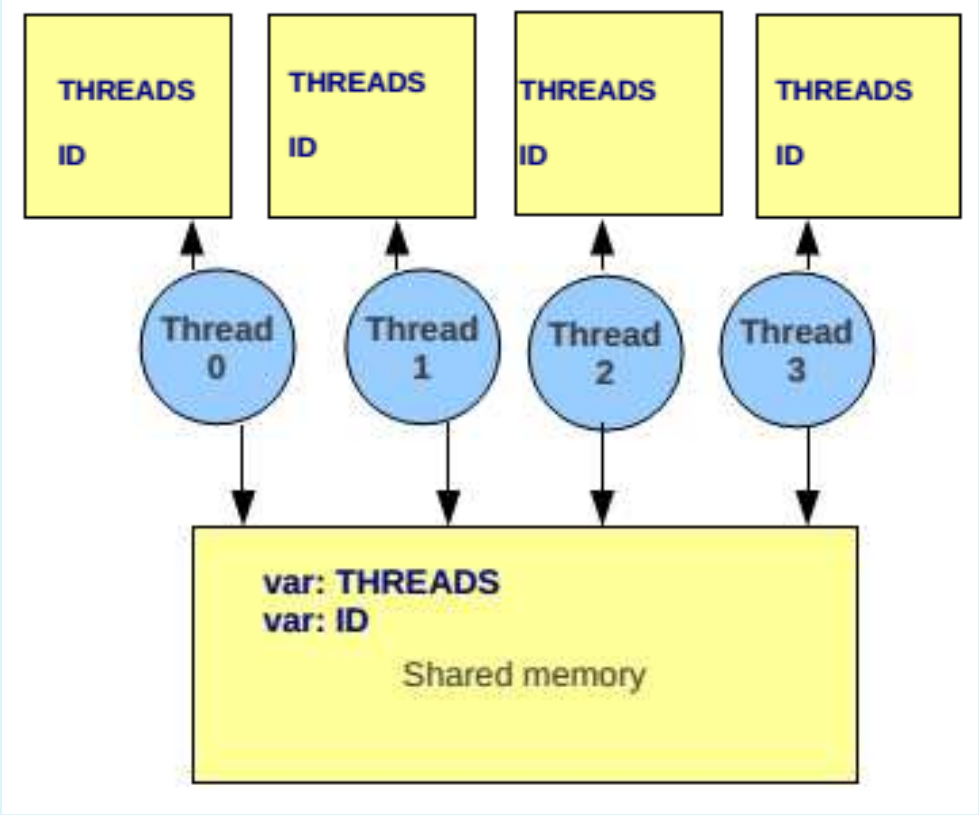
- OpenMP provides a way to declare variables private or shared within an OpenMP block. This is done using the following OpenMP clauses:
  - SHARED (list)
    - All variables in list will be considered shared.
    - Every openmp thread has access to all these variables
  - PRIVATE (list)
    - Every openmp thread will have it's own “private” copy of variables in list
    - No other openmp thread has access to this “private” copy
    - E.g.: #pragma omp parallel private(a,b,c)
- By default most variables are considered shared in OpenMP. Exceptions include index variables (Fortran, C/C++) and variables declared inside parallel region (C/C++)

Bài tập 2: Chạy thử chương trình sau, nhận xét KQ, khắc phục

```
#include <stdio.h>
#include <omp.h>
int main() {
   int id,x;
   omp_set_num_threads(100);
   #pragma omp parallel
   {
         id = omp_get_thread_num();
         x = 10*id;
         printf("\n");
         printf("Hello from thread %d,x =  %d", id, x);
         printf("\n");
   }
   return 0;
}
```

Mô hình Shared memory

- The memory is (logically) shared by all the cpu's
  - There is also private memory for every openmp thread



shared variables ”threads” and ”id” still exist, but every thread also has a private copy of variables “threads” and ”id”. There will not be any race condition for these private variables

Note: Private variable

- OpenMP creates separate data stack for every worker thread to store copies of private variables (master thread uses regular stack)
- Size of these stacks is not defined by OpenMP standards
  - Intel compiler: default stack is 4MB
  - gcc/gfortran: default stack is 2MB
- Behavior of program undefined when stack space exceeded
  - Although most compilers/RT will throw segmentation fault
- To increase stack size use environment var OMP_STACKSIZE, e.g.
  - export OMP_STACKSIZE=512M
  - export OMP_STACKSIZE=1GB
- To make sure master thread has large enough stack space use
  - ulimit -s command (unix/linux).

Bài tập 1

a) Viết CT tính tổng của hai mảng kích thước N:

A(N) + B(N) = C(N)

bằng ngôn ngữ C, sau đó chuyển thành CT OpenMP chạy song song trên M threads (N chia hết cho M).

- Xét bài toán 2 chiều?

b) Viết chương trình C để tính tổng tất cả các phần tử của một mảng A gồm N phần tử. Sau đó chuyển thành chương trình OpenMP

c) Viết chương trình C để tìm phần tử max trong một mảng A gồm N phần tử. Sau đó chuyển thành chương trình OpenMP

Bài tập 2

- Viết chương trình C để tính tích 2 ma trận:

  A[M][N] * B[N][P] = C[M][P]

- Chuyển chương trình trên thành chương trình OpenMP. Tính hiệu quả của chương trình OpenMP