

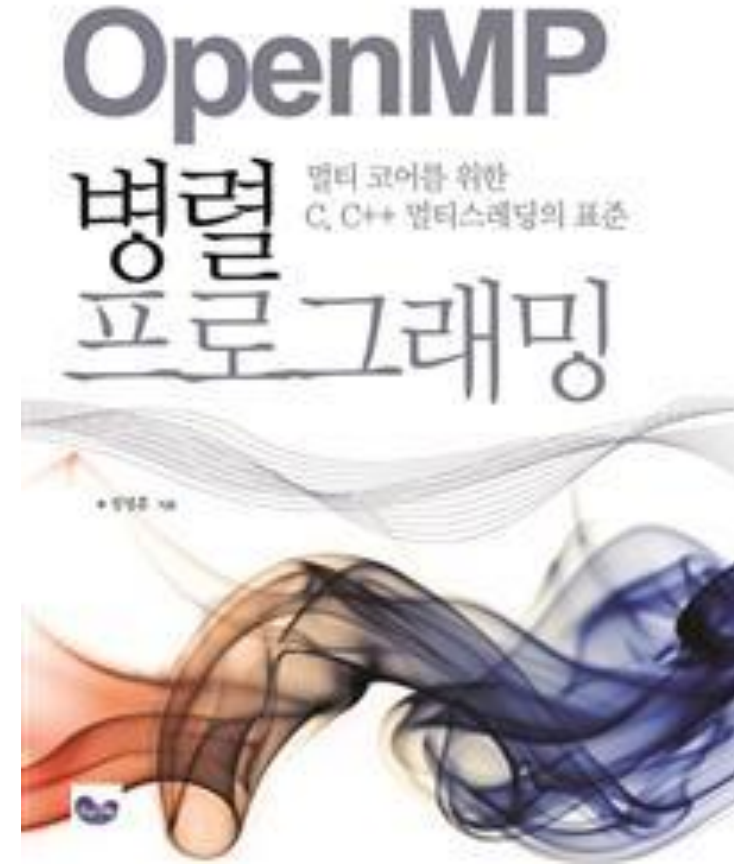
OpenMPTM : Tutorial

OpenMP 병렬프로그래밍 – 정영훈 지음.

안재원

목차

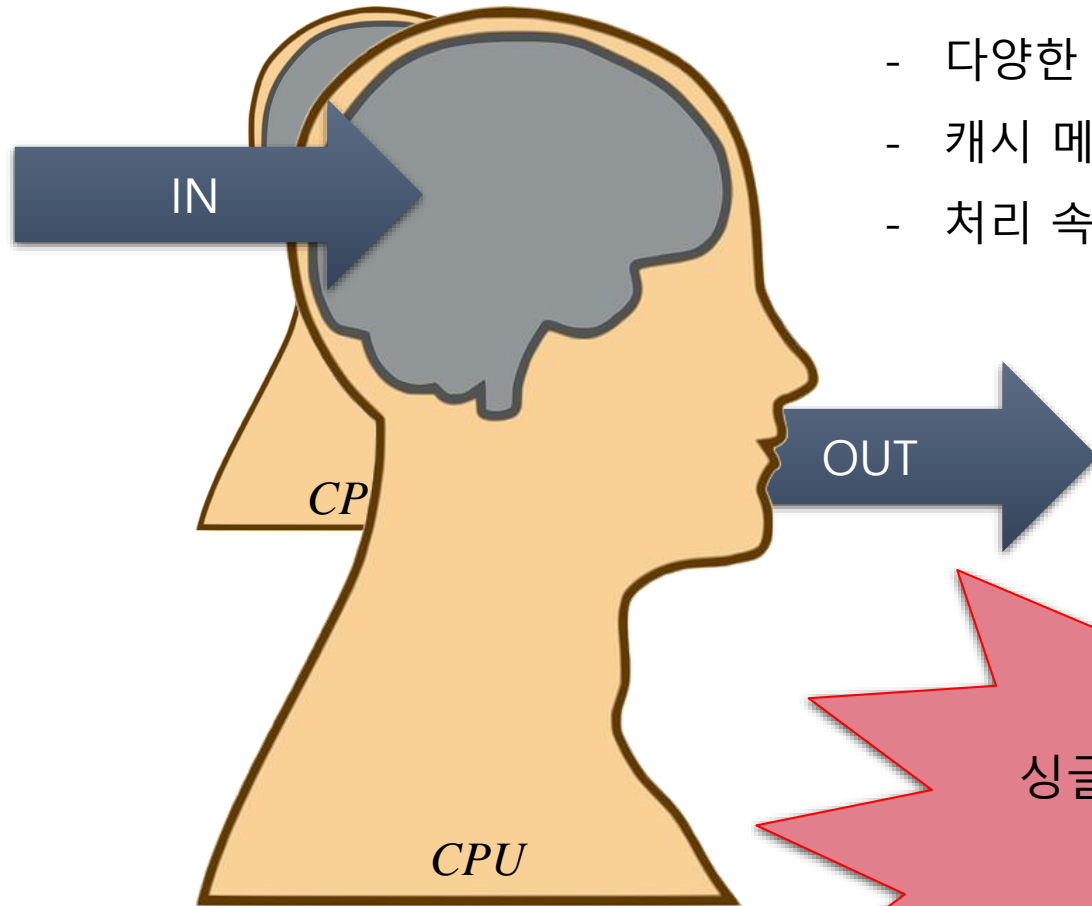
- Parallel Processing
- OpenMP
- Intel Parallel Studio
- Example & Result



01

Parallel Processing

- Intro



- 다양한 명령어
- 캐시 메모리 증가
- 처리 속도 향상 {
 - 신뢰도
 - 전력 & 발열

$$P = C \times V^2 \times F$$

P : 칩의 전력 소모량

C : 클럭 당 스위칭 된 전기 용량

V : 전압

F : 주파수

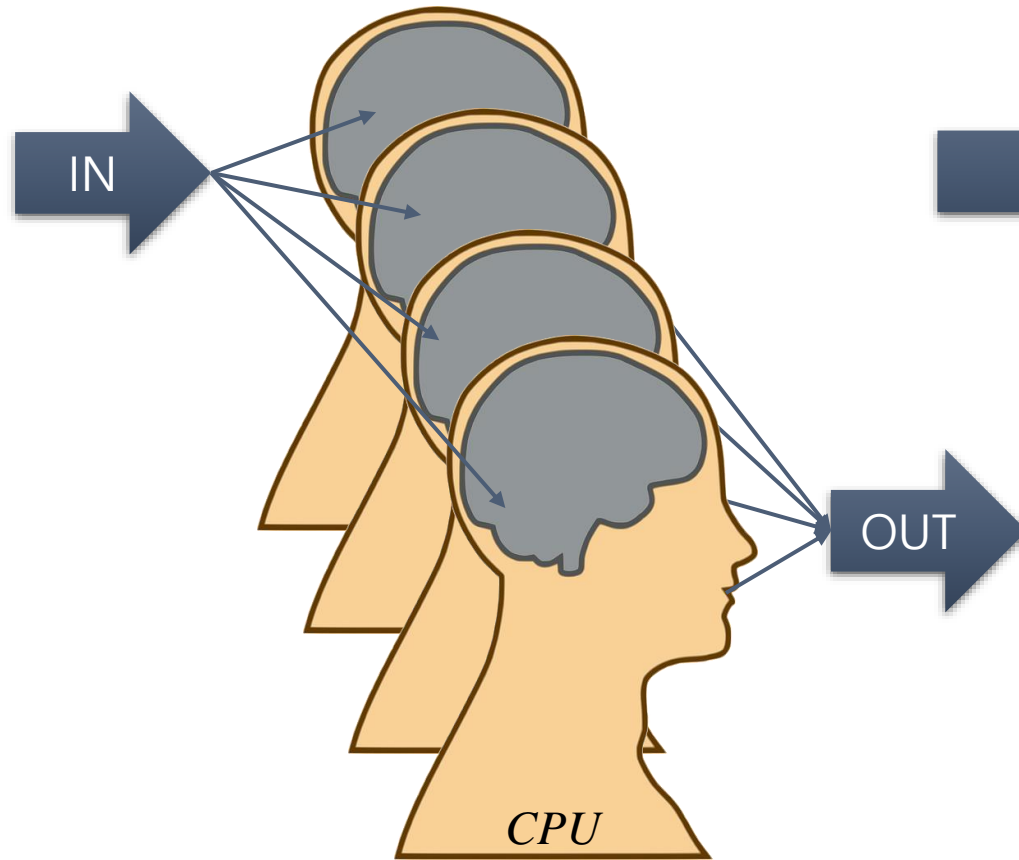
싱글 코어의 한계



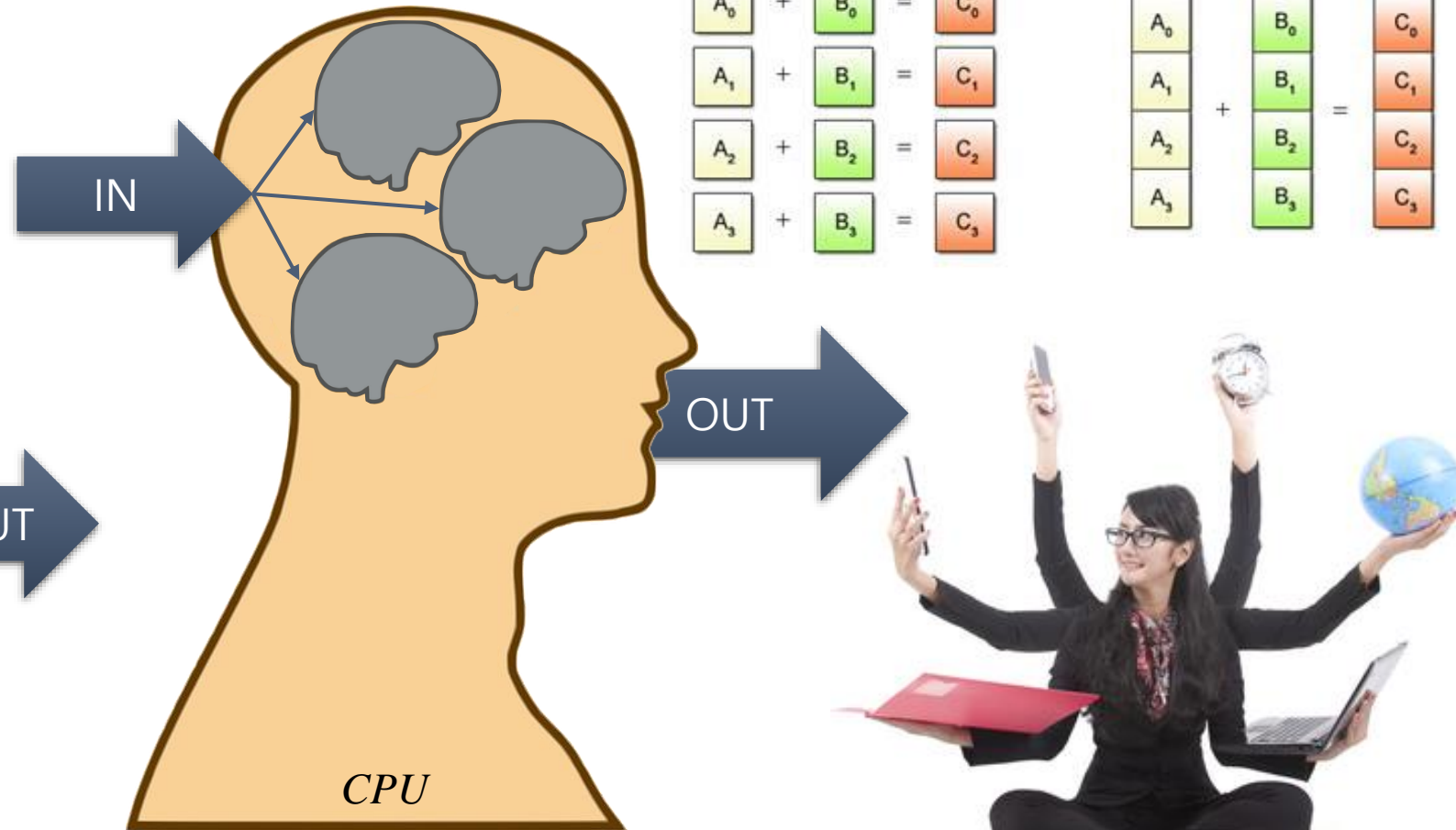
01

Parallel Processing

- Intro



- 멀티 코어



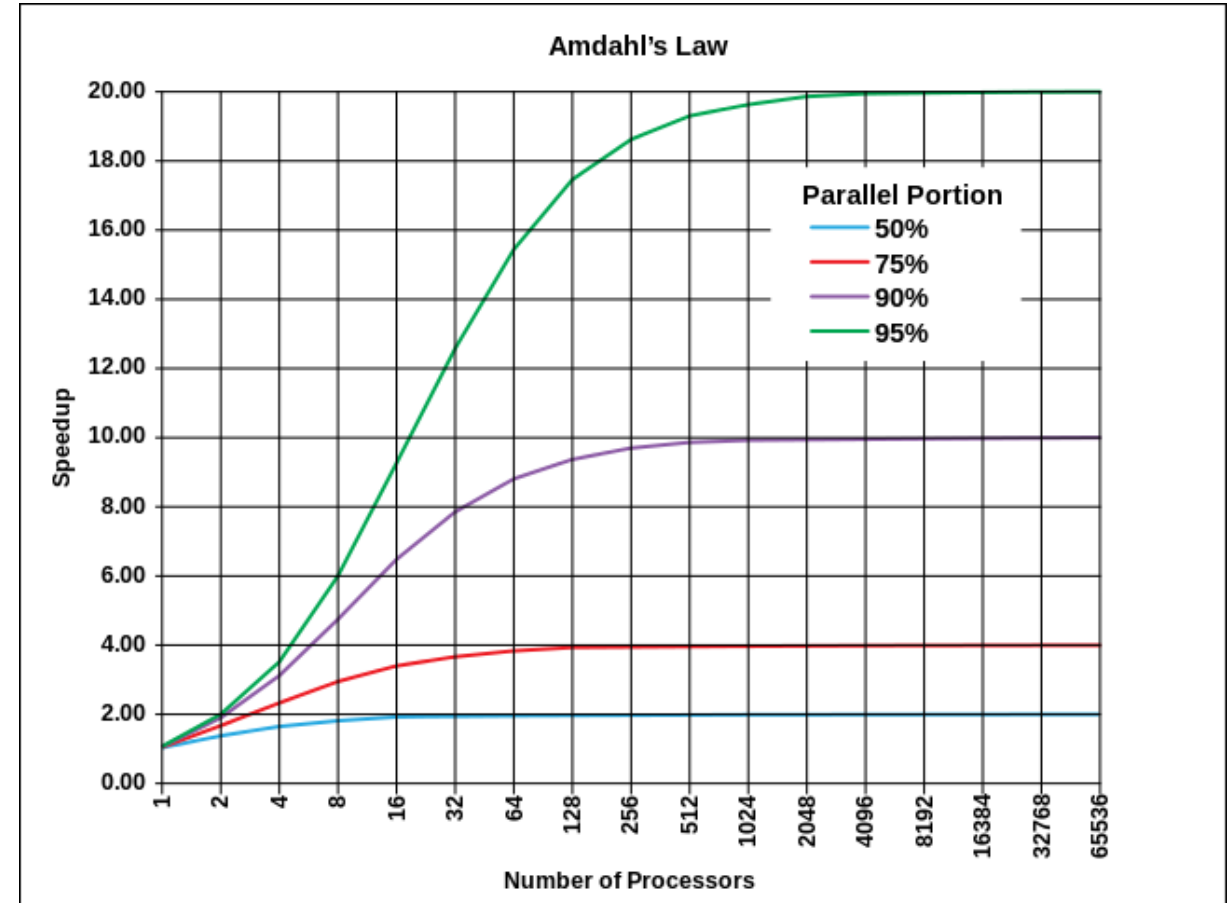
- SIMD

01

Parallel Processing

- Amdahl's Law

$$\frac{1}{(1-P) + \frac{P}{S}}$$

 P : 개선 비율 S : 개선 정도

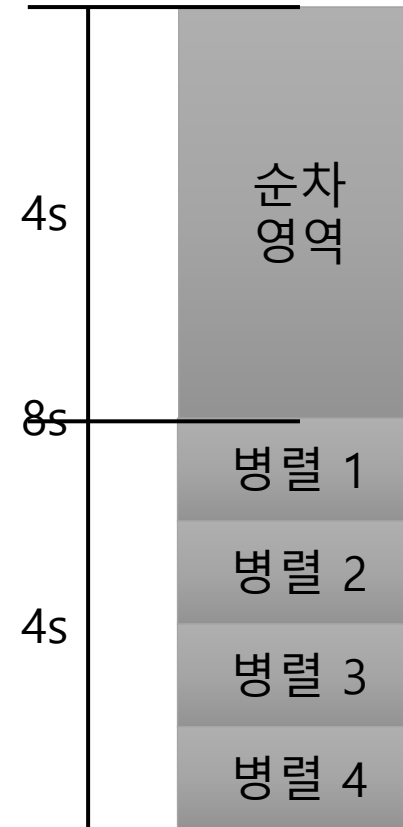
01

Parallel Processing

- Amdahl's Law

$$\frac{1}{(1-P) + \frac{P}{S}}$$

P : 50% (0.5)
 S : 4

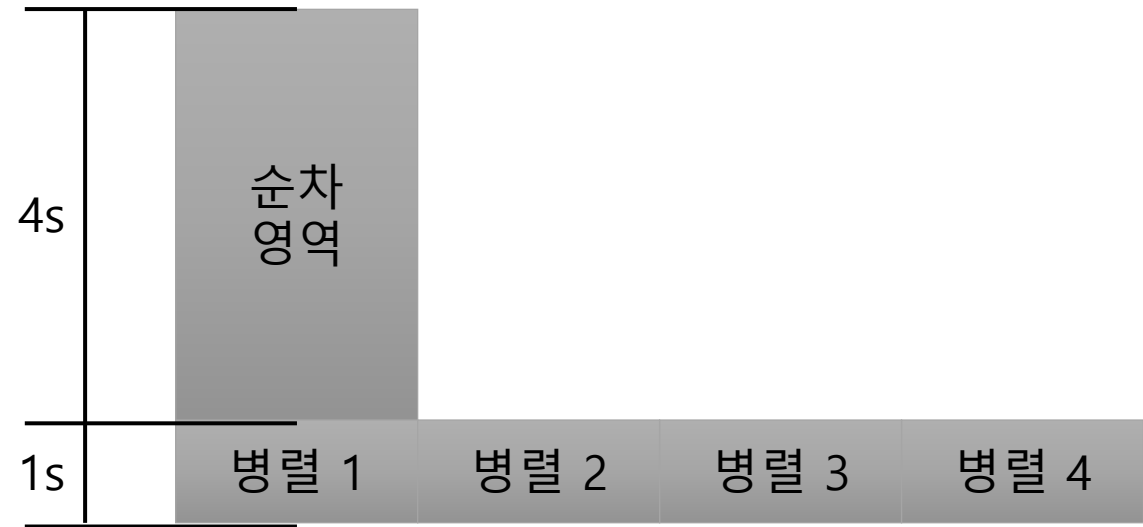


01

Parallel Processing

- Amdahl's Law

$$\frac{1}{(1-P) + \frac{P}{S}} \quad \begin{array}{l} P : 50\% (0.5) \\ S : 4 \end{array}$$

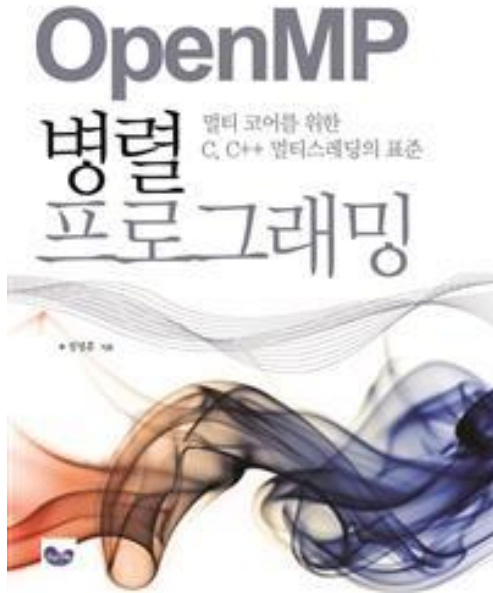


성능 개선 효과 : $\frac{1}{(1-0.5) + \frac{0.5}{4}} = 1.6 = \frac{8}{5}$

02

OpenMP

- OpenMP



- 병렬화 변환이 쉬움
- 순차 방식에서 자연스럽게 전환
- 팀원간의 공유 원할
- 많은 부분 병렬화 가능.

<병렬 프로그래밍 구현 순서>

- 병렬화할 부분을 찾아낸다.

- 동기화가 필요하면 동기화 기능을 구현한다.

- 프로그램의 오류가 발생하면 디버깅을 한다.
- 순차 프로그램과 병렬프로그램의 성능을 평가한다.

02

OpenMP

- OpenMP

```

#include "stdafx.h"
#include "windows.h"

const int THREAD_COUNT = 4;

DWORD WINAPI PrintHelloWorld(LPVOID arg)
{
    printf("Hello world\n");
    return 0;
}

int main()
{
    HANDLE hThreadArray[THREAD_COUNT];
    int i = 0;

    //4개의 Thread 생성.
    for(i = 0; i < THREAD_COUNT; i++)
    {
        hThreadArray[i] = CreateThread(NULL, 0, PrintHelloWorld, NULL, 0, NULL);
    }

    //4개의 작업이 끝나기를 기다림.
    WaitForMultipleObjects(THREAD_COUNT, hThreadArray, TRUE, INFINITE);

    return 0;
}

```

<병렬 프로그래밍 구현 순서>

- 병렬화할 부분을 찾아낸다.
- 병렬화할 부분을 별도 함수로 분리시킨다.
- 별도의 스레드로 생성하여 실행할 함수를 연결한다.
- 동기화가 필요하면 동기화 기능을 구현한다.
- 메인 스레드는 별도 생성된 작업 스레드의 작업완료를 기다린다.
- 프로그램의 오류가 발생하면 디버깅을 한다.
- 순차 프로그램과 병렬프로그램의 성능을 평가한다.

02

OpenMP

- OpenMP

```
#include "stdafx.h"

int main()
{
    printf("Hello world\n");
    return 0;
}
```



```
#include "stdafx.h"
#include <omp.h>

int main()
{
    //4개의 thread생성
    #pragma omp parallel num_threads(4)
    {
        printf("Hello world\n");
    }
    return 0;
}
```

<병렬 프로그래밍 구현 순서>

- 병렬화할 부분을 찾아낸다.

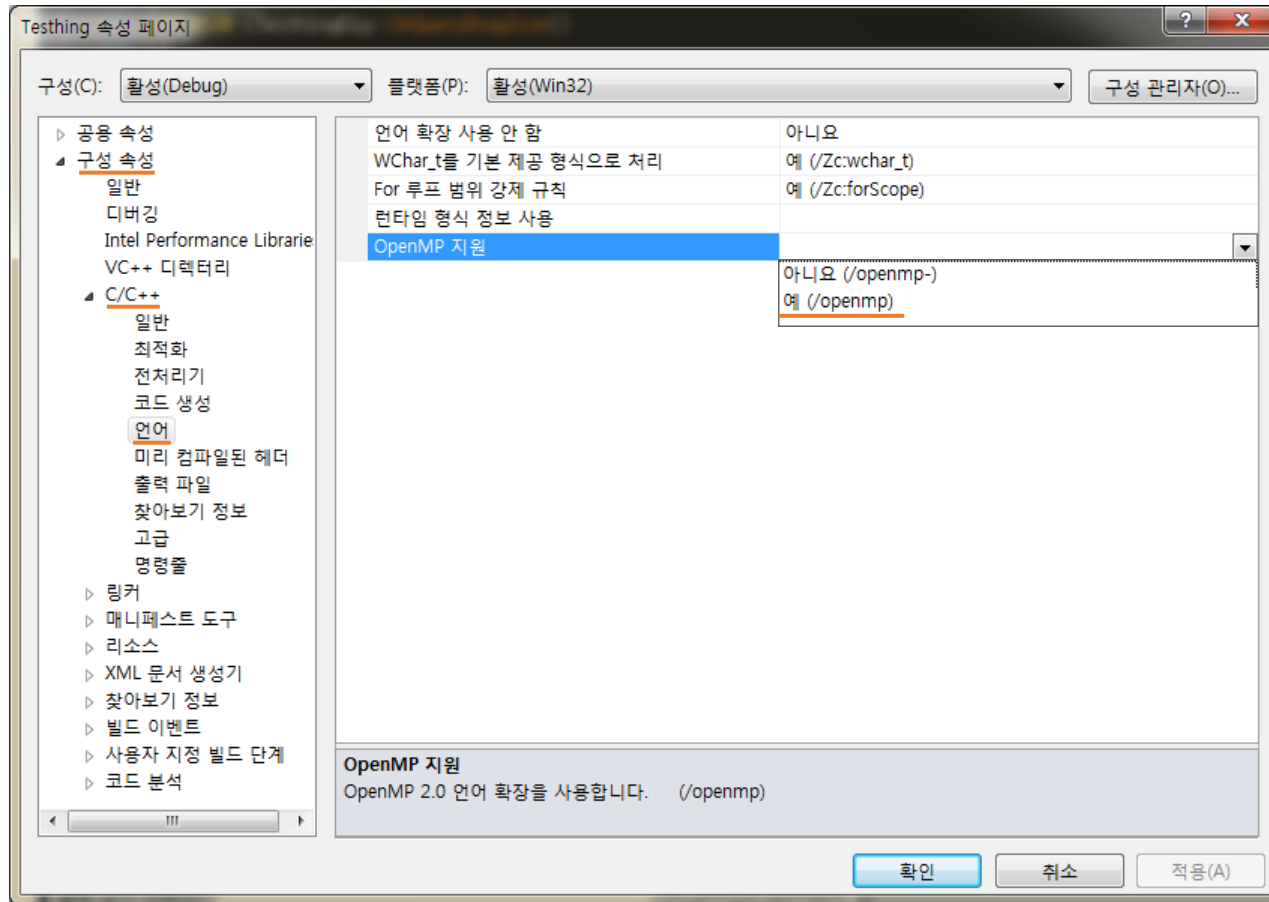
- 동기화가 필요하면 동기화 기능을 구현한다.

- 프로그램의 오류가 발생하면 디버깅을 한다.
- 순차 프로그램과 병렬프로그램의 성능을 평가한다.

02

OpenMP

- Active OpenMP



```
#include <omp.h>

int main()
{
    #pragma omp parallel
    {
        printf("Hello OpenMP world\n");
    }
}
```



```
C:\Windows\system32\cmd.exe

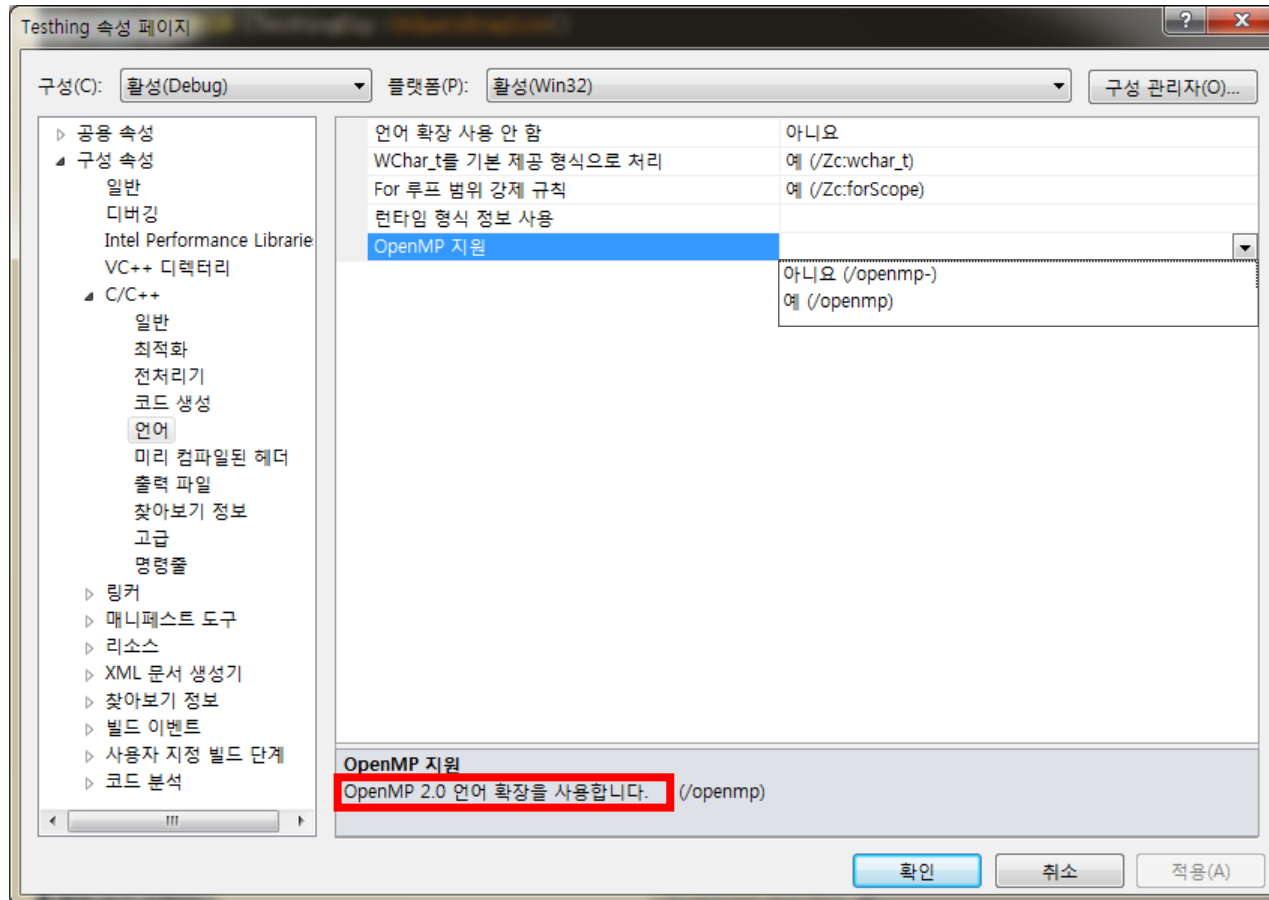
Hello OpenMP world
Hello OpenMP world
Hello OpenMP world
Hello OpenMP world
계속하려면 아무 키나 누르십시오 . . .
```

03

Intel Parallel Studio

Intel Parallel Studio XE 2015

- OpenMP 2.0?



- For Parallel processing
- Parallel processing Debug
- OpenMP 4.0
- 4 component
 - 1) Intel Parallel Composer
 - 2) Intel Parallel Inspector
 - 3) Intel Parallel Amplifier
 - 4) Intel Parallel Advisor

03

Intel Parallel Studio

Intel Parallel Studio XE 2015

- <https://software.intel.com/en-us>

Intel Developer Zone

Development > Tools > Resources >

Join Today > Log in

Search our content library...

Free Software Tools

Supporting qualified students, educators, academic researchers and open source contributors

Free Tools for Students

Intel supports the next generation scientists and engineers through ensuring students have access to the latest Intel® Software Development Tools. The student program provides free access to Intel® Parallel Studio XE Professional Edition for C++ for Windows* and Linux*, and Intel® Parallel Studio XE Composer Edition for C++ OS X*. All products include user forum support. For more information, please visit [FAQ](#).

Qualifications:

- Matriculated students of a higher education institution defined as a public or private vocational school, correspondence school, junior college, college, or university.
- Students cannot be paid and/or compensated for software development. The products below are currently free for your use under the terms of the [Non-Commercial License](#).

Offer:
Get Intel® Parallel Studio XE Professional Edition:

[for C++ Windows*](#)

Academic Researcher >

Student

Educator >

Open Source Contributor >

Quick Links

[User Support Forums >](#)

[Product Information >](#)

[Software End-User License >](#)

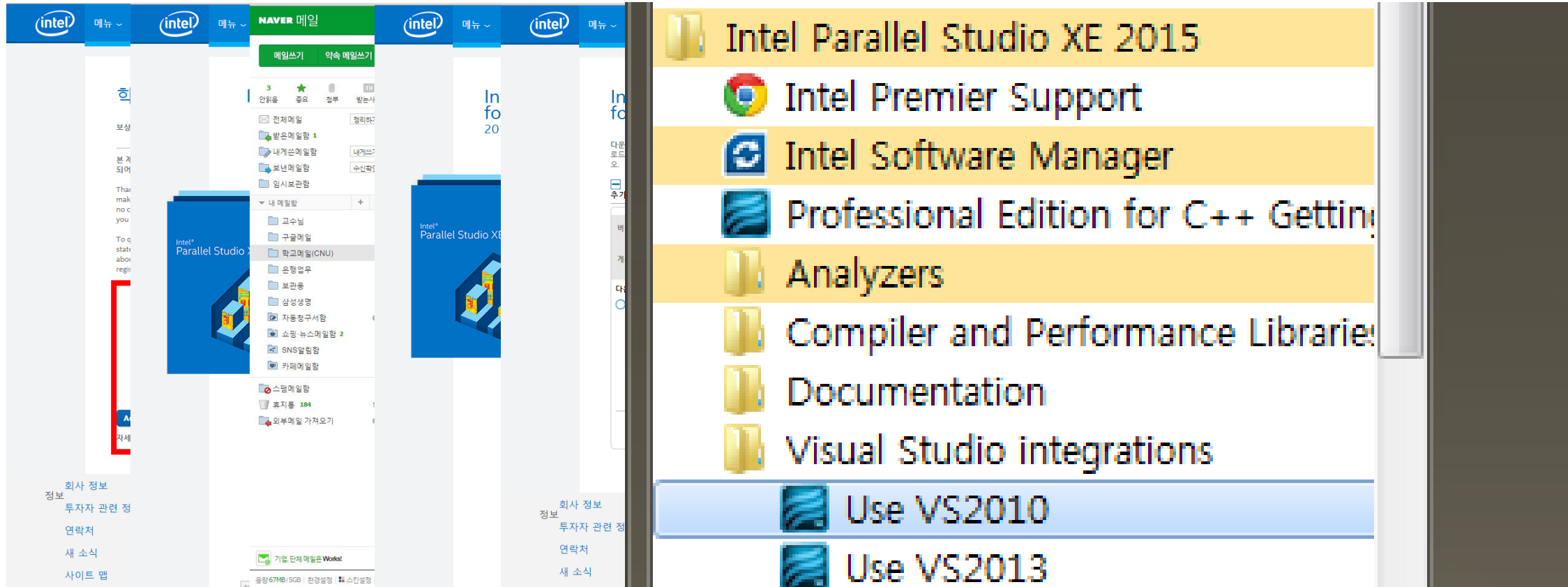
[Rate Us](#) ★★ ★

Look for us on: [f](#) [t](#) [g+](#) [in](#) [yt](#) [English >](#)

03

Intel Parallel Studio

Intel Parallel Studio XE 2015

- <https://software.intel.com/en-us>


Intel Parallel Studio XE 2015

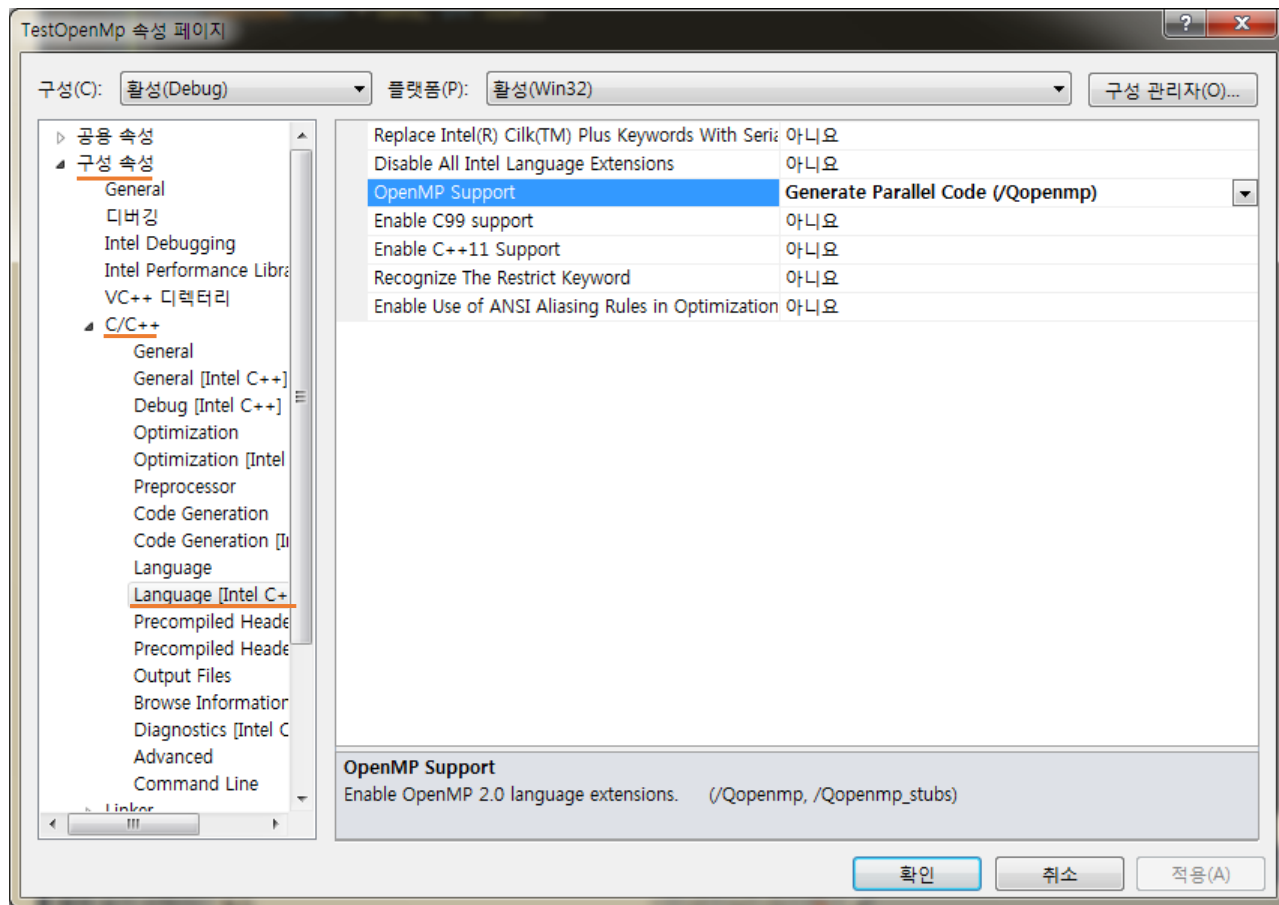
- Intel Premier Support
- Intel Software Manager
- Professional Edition for C++ Getting Started
- Analyzers
- Compiler and Performance Libraries
- Documentation
- Visual Studio integrations
- Use VS2010
- Use VS2013

03

Intel Parallel Studio

Intel Parallel Studio XE 2015

- Intel Parallel Studio XE 2015



```
#include <omp.h>

int main()
{
    printf("OpenMP version is %d\n", _OPENMP);
    return 0;
}
```



```
1>MessageBuildingWithCompiler:
1> Building with Intel(R) C++ Compiler XE 15.0
1>ClCompile:
1> ***** ClCompile (Win32 - Intel C++)
```



```
C:\Windows\system32\cmd.exe

OpenMP version is 201307
계속하려면 아무 키나 누르십시오 . . .
```

04

Example & Result

- #Pragma omp for

```

68  const unsigned int MAX = 100000000;
69  int i = 0;
70  float * Data;
71  Data = new float[MAX];
72
73  InputData(Data,MAX);
74
75  CStopWatch watch;
76
77  watch.Start();
78  for(i = 0;i<MAX;i++)
79  {
80      Data[i] = sqrt(Data[i]);
81  }
82  watch.End();
83
84  PrintData(Data);
85  printf("순차 계산 시간 %.3fms\n\n",watch.GetDurationMilliSecond());
86
87
88  InputData(Data,MAX);
89  watch.Start();
90
91  #pragma omp parallel
92  {
93      #pragma omp for
94      for (i = 0;i<MAX;i++)
95      {
96          Data[i] = sqrt(Data[i]);
97      }
98  }
99  watch.End();
100
101  PrintData(Data);
102  printf("병렬 계산 시간 %.3fms\n\n",watch.GetDurationMilliSecond());
103
104
105  delete[] Data;

```

C:\Windows\system32\cmd.exe

Data : 0.000000 1.000000 1.414214 1.732051 2.000000
순차 계산 시간 2829.139msc

Data : 0.000000 1.000000 1.044274 1.071076 1.090508
병렬 계산 시간 2896.253msc

계속하려면 아무 키나 누르십시오 . . .

C:\Windows\system32\cmd.exe

Data : 0.000000 1.000000 1.414214 1.732051 2.000000
순차 계산 시간 2778.858msc

Data : 0.000000 1.000000 1.414214 1.732051 2.000000
병렬 계산 시간 942.788msc

계속하려면 아무 키나 누르십시오 . . .

04

Example & Result

- #Pragma omp sections

```

24 const unsigned int MAX = 100000000;
25 int i = 0;
26 float * Data1 = new float[MAX];
27 float * Data2 = new float[MAX];
28
29 //순차 계산.
30 InputData2(Data1,MAX);
31 InputData2(Data2,MAX);
32
33 CStopWatch watch;
34
35 watch.Start();
36 CalcSQRT(Data1,MAX);
37 CalcLOG(Data2,MAX);
38 watch.End();
39
40 PrintData(Data1);
41 PrintData(Data2);
42 printf("순차 계산 시간 %.3fsec\n\n",watch.GetDurationMilliSecond());
43
44 //병렬 계산.
45 InputData2(Data1,MAX);
46 InputData2(Data2,MAX);
47
48 watch.Start();
49 #pragma omp parallel
50 {
51     #pragma omp sections
52     {
53         #pragma omp section
54             CalcSQRT(Data1,MAX);
55         #pragma omp section
56             CalcLOG(Data2,MAX);
57     }
58 }
59 watch.End();
60
61 PrintData(Data1);
62 PrintData(Data2);
63 printf("병렬 계산 시간 %.3fsec\n\n",watch.GetDurationMilliSecond());
64
65 delete[] Data1;
66 delete[] Data2;

```

```

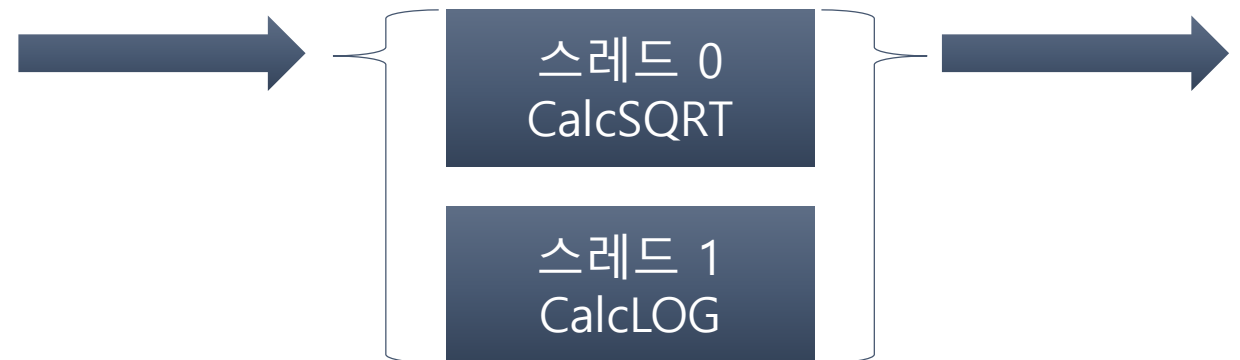
C:\Windows\system32\cmd.exe

Data : 0.000000 1.000000 1.414214 1.732051 2.000000
Data : -1.#INF00 0.000000 0.693147 1.098612 1.386294
순차 계산 시간 6503.475msec

Data : 0.000000 1.000000 1.414214 1.732051 2.000000
Data : -1.#INF00 0.000000 0.693147 1.098612 1.386294
병렬 계산 시간 3865.557msec

계속하려면 아무 키나 누르십시오 . . .

```



04

Example & Result

- #Pragma omp task

```

110 const int MAX = 41;
111 int FibNumber[MAX];
112
113 int i = 0;
114
115 CStopWatch watch;
116
117 //순차처리
118 watch.Start();
119 for(i = 1; i < MAX; i++)
120 {
121     FibNumber[i] = Fibonacci(i);
122 }
123 watch.End();
124
125 printf("순차처리 연산시간 : %.3fms\n피보나치수 : ", watch.GetDurationMilliSecond());
126 for(i = 1; i < 10; i++)
127 {
128     printf("%d ", FibNumber[i]);
129 }
130 printf("\n\n");
131
132 //병렬처리 1
133 watch.Start();
134 #pragma omp parallel
135 {
136     #pragma omp for
137     for(i = 1; i < MAX; i++)
138     {
139         FibNumber[i] = Fibonacci(i);
140     }
141 }
142 watch.End();
143
144 printf("병렬처리1 연산시간 : %.3fms\n피보나치수 : ", watch.GetDurationMilliSecond());
145 for(i = 1; i < 10; i++)
146 {
147     printf("%d ", FibNumber[i]);
148 }
149 printf("\n\n");

```

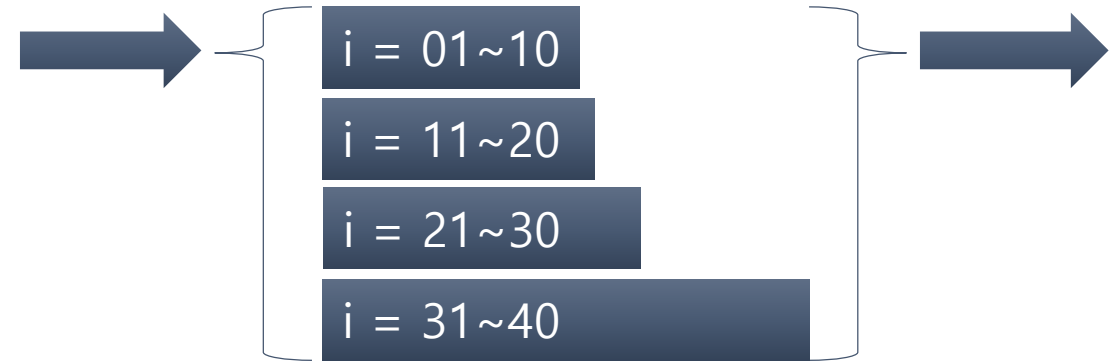
```

C:\Windows\system32\cmd.exe
순차처리 연산시간 : 9722.934ms
피보나치수 : 1 1 2 3 5 8 13 21 34

병렬처리1 연산시간 : 9663.698ms
피보나치수 : 1 1 2 3 5 8 13 21 34

```

$$F_n = F_{n-1} + F_{n-2}$$



04

Example & Result

- #Pragma omp task

```

151 //병렬처리 2
152 watch.Start();
153 #pragma omp parallel
154 {
155     #pragma omp single private(i)
156     for(i = 1; i < MAX; i++)
157     {
158         #pragma omp task
159         FibNumber[i] = Fibonacci(i);
160     }
161 }
162 watch.End();
163
164 printf("병렬처리2 연산시간 : %.3f msec\n피보나치수 : ", watch.GetDurationMilliSecond());
165 for(i = 1; i < 10; i++)
166 {
167     printf("%d ", FibNumber[i]);
168 }
169 printf("\n\n");

```

```

C:\Windows\system32\cmd.exe

순차처리 연산시간 : 9722.934msec
피보나치수 : 1 1 2 3 5 8 13 21 34

병렬처리1 연산시간 : 9663.698msec
피보나치수 : 1 1 2 3 5 8 13 21 34

병렬처리2 연산시간 : 3833.245msec
피보나치수 : 1 1 2 3 5 8 13 21 34

```



04

Example & Result

- #Pragma omp task

```

171 //병렬처리 3
172 watch.Start();
173 #pragma omp parallel
174 {
175     #pragma omp single private(i)
176     for(i = 1; i < MAX; i++)
177     {
178         FibNumber[i] = FibonacciTask(i);
179     }
180 }
181 watch.End();
182 printf("병렬처리3 연산시간 : %.3fms\n피보나치수 : ", watch.GetDurationMilliSecond());
183 for(i = 1; i < 10; i++)
184 {
185     printf("%d ", FibNumber[i]);
186 }
187 printf("\n\n");
188

```

```

256 int FibonacciTask(int n)
257 {
258     int x, y;
259     if(n < 2)
260     {
261         return n;
262     }
263     else
264     {
265         #pragma omp task shared(x)
266         x = Fibonacci(n-1);
267         #pragma omp task shared(y)
268         y = Fibonacci(n-2);
269         #pragma omp taskwait
270         return (x+y);
271     }
272 }

```

```

C:\Windows\system32\cmd.exe

순차처리 연산시간 : 9722.934msec
피보나치수 : 1 1 2 3 5 8 13 21 34

병렬처리1 연산시간 : 9663.698msec
피보나치수 : 1 1 2 3 5 8 13 21 34

병렬처리2 연산시간 : 3833.245msec
피보나치수 : 1 1 2 3 5 8 13 21 34

병렬처리3 연산시간 : 6522.077msec
피보나치수 : 1 1 2 3 5 8 13 21 34

계속하려면 아무 키나 누르십시오 . . .

```



04

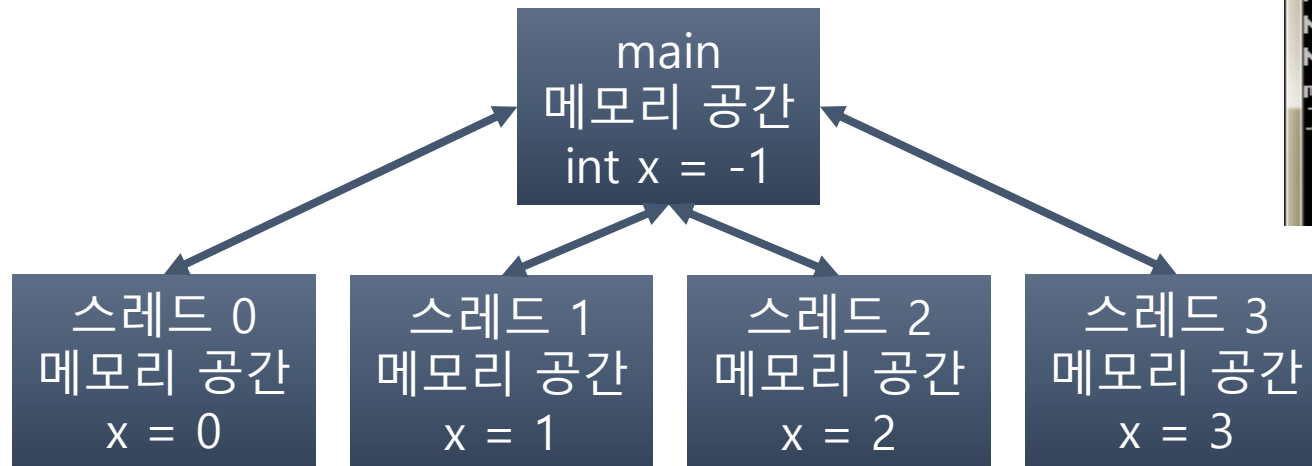
Example & Result

- omp Variable

```

191 int x = -1;
192
193 printf("main area x : %d\n",x);
194
195 #pragma omp parallel
196 {
197
198     printf("Num %d Thread area x(before) : %d\t",omp_get_thread_num(),x);
199
200     if (omp_get_thread_num()==0)x = 0;
201     else if (omp_get_thread_num()==1)x = 1;
202     else if (omp_get_thread_num()==2)x = 2;
203     else x = 3;
204
205     printf("Num %d Thread area x(after) : %d\n",omp_get_thread_num(),x);
206 }
207
208 printf("main area x : %d\n",x);

```



```

C:\Windows\system32\cmd.exe

main area x : -1
Num 1 Thread area x(before) : -1      Num 1 Thread area x(after) : 1
Num 2 Thread area x(before) : -1      Num 2 Thread area x(after) : 2
Num 0 Thread area x(before) : -1      Num 0 Thread area x(after) : 0
Num 3 Thread area x(before) : -1      Num 3 Thread area x(after) : 3
main area x : 3
계속하려면 아무 키나 누르십시오 . . .

```

```

C:\Windows\system32\cmd.exe

main area x : -1
Num 0 Thread area x(before) : -1      Num 0 Thread area x(after) : 0
Num 2 Thread area x(before) : -1      Num 2 Thread area x(after) : 2
Num 3 Thread area x(before) : -1      Num 3 Thread area x(after) : 3
Num 1 Thread area x(before) : -1      Num 1 Thread area x(after) : 1
main area x : 1
계속하려면 아무 키나 누르십시오 . . .

```

04

Example & Result

- omp Variable

```

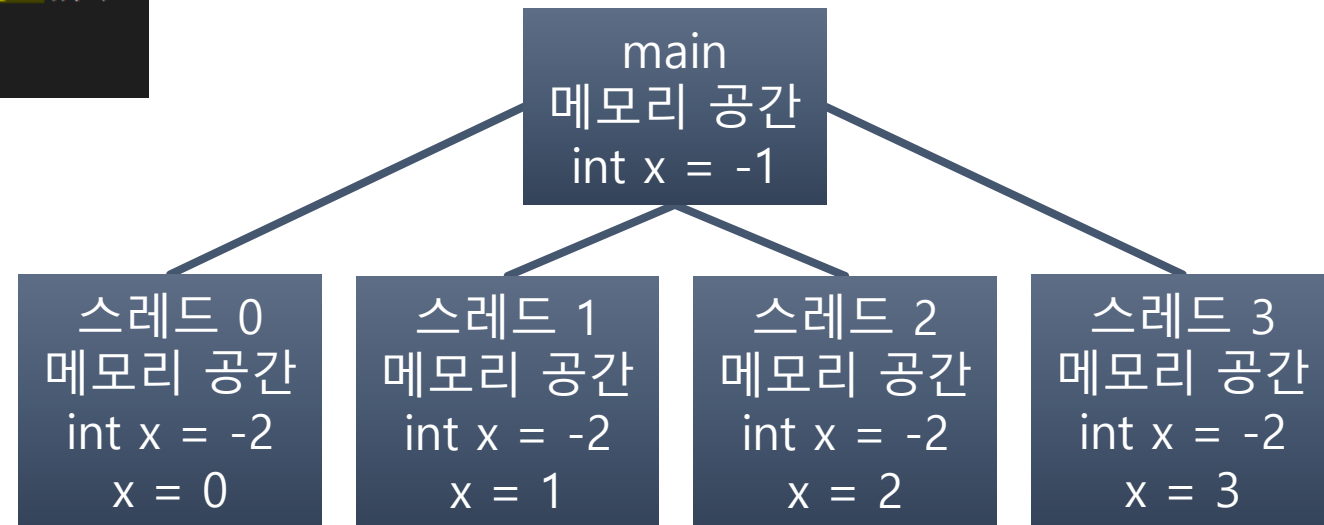
191 int x = -1;
192
193 printf("main area x : %d\n",x);
194
195 #pragma omp parallel
196 {
197     int x = -2;
198
199     printf("Num %d Thread area x(before) : %d\t",omp_get_thread_num(),x);
200
201     if (omp_get_thread_num()==0)x = 0;
202     else if (omp_get_thread_num()==1)x = 1;
203     else if (omp_get_thread_num()==2)x = 2;
204     else x = 3;
205
206     printf("Num %d Thread area x(after) : %d\n",omp_get_thread_num(),x);
207 }
208
209 printf("main area x : %d\n",x);

```

```

C:\Windows\system32\cmd.exe
main area x : -1
Num 0 Thread area x(before) : -2      Num 0 Thread area x(after) : 0
Num 1 Thread area x(before) : -2      Num 1 Thread area x(after) : 1
Num 3 Thread area x(before) : -2      Num 3 Thread area x(after) : 3
Num 2 Thread area x(before) : -2      Num 2 Thread area x(after) : 2
main area x : -1
계속하려면 아무 키나 누르십시오 . . .

```



04

Example & Result

- omp Variable

```

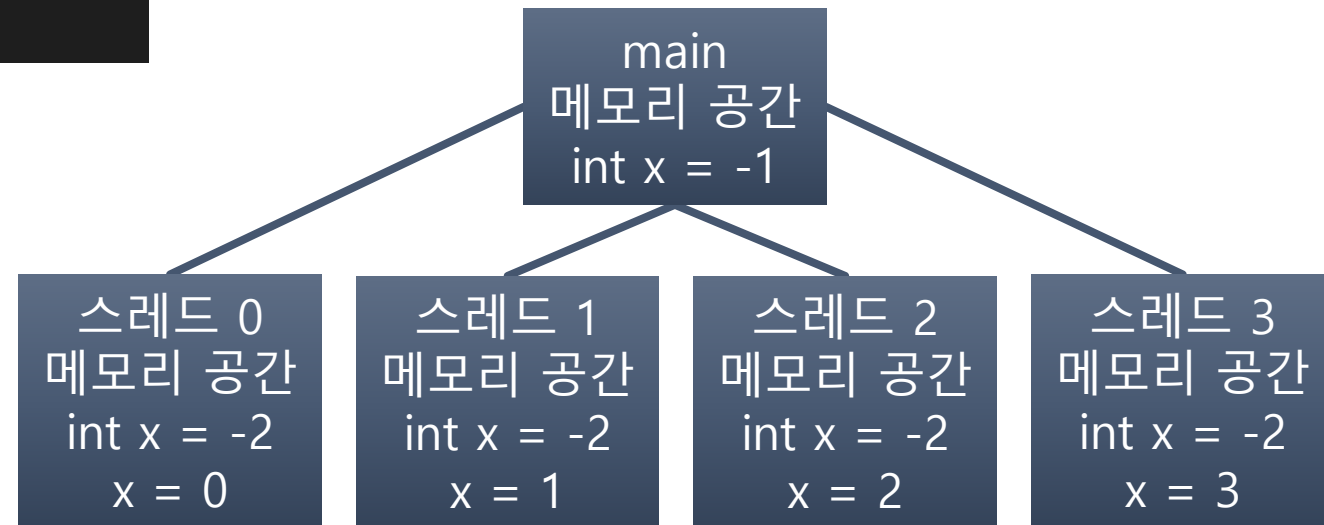
191 int x = -1;
192
193 printf("main area x : %d\n",x);
194
195 #pragma omp parallel private(x)
196 {
197     printf("Num %d Thread area x(before) : %d\t",omp_get_thread_num(),x);
198
199     if (omp_get_thread_num()==0)x = 0;
200     else if (omp_get_thread_num()==1)x = 1;
201     else if (omp_get_thread_num()==2)x = 2;
202     else x = 3;
203
204     printf("Num %d Thread area x(after) : %d\n",omp_get_thread_num(),x);
205 }
206
207 printf("main area x : %d\n",x);
208

```

```

C:\Windows\system32\cmd.exe
main area x : -1
Num 0 Thread area x(before) : -858993460      Num 0 Thread area x(after) : 0
Num 2 Thread area x(before) : -858993460      Num 2 Thread area x(after) : 2
Num 3 Thread area x(before) : -858993460      Num 3 Thread area x(after) : 3
Num 1 Thread area x(before) : -858993460      Num 1 Thread area x(after) : 1
main area x : -1
계속하려면 아무 키나 누르십시오 . . .

```



감사합니다

Q & A