

# OpenMP

# Lab 1

# 실습 결과물 제출 안내

- EL을 통해 과제 제출

- 제출 내용

- 작성한 소스코드

- 소스코드 원본 및 실행 파일 (압축하여 제출)
      - 헤더파일 및 소스코드(예, .c .cpp .h .hpp 등)와 실행 파일 만 포함할 것
      - 프로젝트 설정, DB, 오브젝트 파일 등 부수 파일 제출 금지
    - 소스코드를 모두 모은 한글 (or MS Word) 파일

\* 위 두가지 형태 모두 제출해야 함

- 보고서

- 실행 결과 캡처 및 결과에 대한 간단한 설명 및 소감

\* 소스코드 모음 한글 파일 및 보고서는 압축하지 않고 개별 파일로 제출 할 것

이재홍(2016136103)

컴퓨터공학부 (4학년)

정상제출

2019.03.14 오후 12:43:31

[자세히보기](#)

첨부파일 : Lab1-이재홍-2016136103.zip (0.97KB)

Lab1-source-이재홍-2016136103.docx (0.02MB)

Lab1-이재홍-2016136103.docx (0.1MB)

# 실습 결과물 제출 안내

- 제출 기한: **3/19(일) 23:59**
- **Copy 허용 (1.5점 인정)**
  - 과제 보고서에 Copy 했음을 반드시 명시 할 것
    - 명시 없이 Copy 및 적발 시, 최종 학점 한단계 하락 적용
  - 단, 직접 3번 typing 하는 영상 제출
    - Youtube 업로드 후, 링크 제출
- **감점 사항**
  - 지각 제출: -0.5점/일
  - 보고서 미제출: 0점 처리
    - 소감 미 포함 시, 보고서 불인정

Lab. 1-1

# Hello OpenMP!

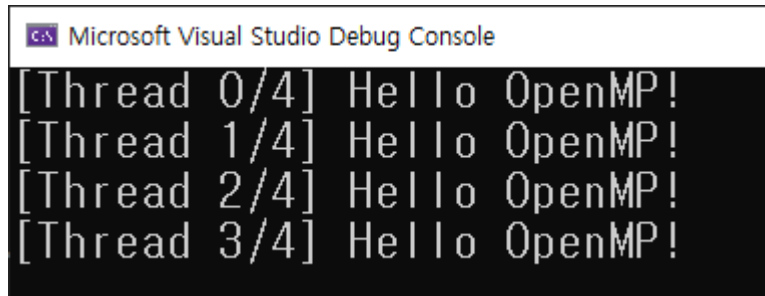
Your First OpenMP Programs

# Lab 1-1. Hello OpenMP

- 사용하는 스레드 수 만큼 “Hello OpenMP”를 출력하는 프로그램을 작성하기

- 과제 목표

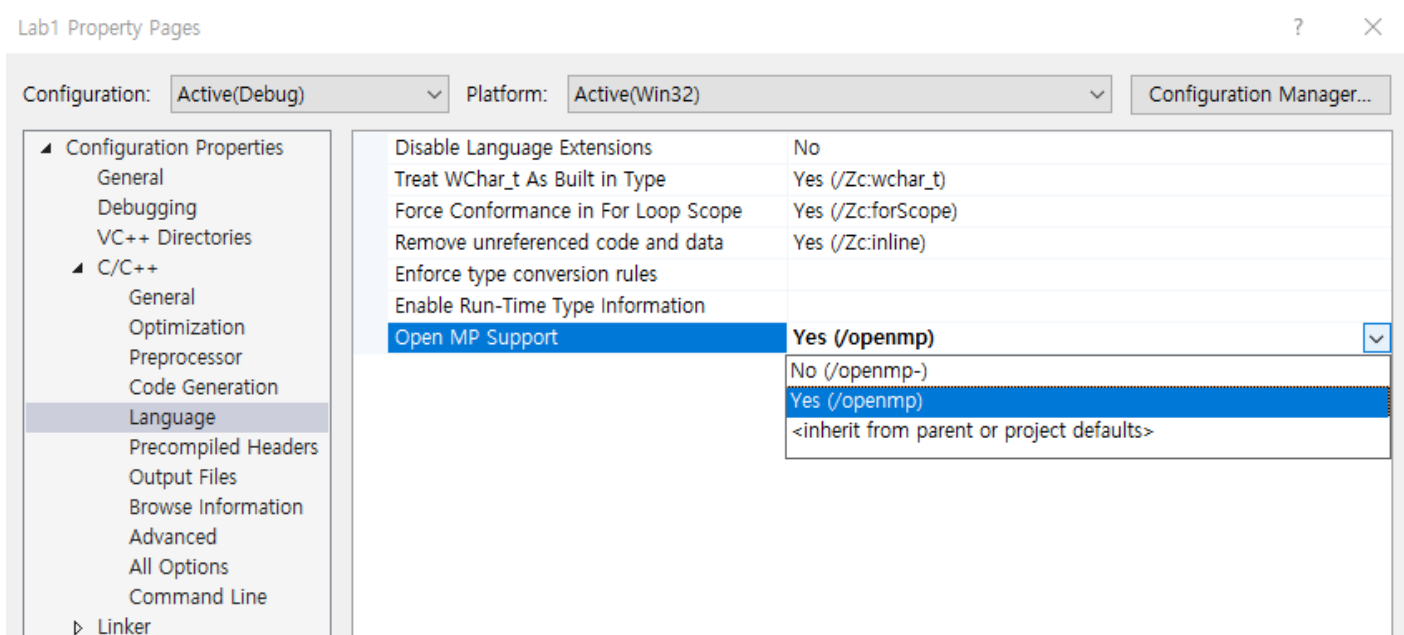
- OpenMP 컴파일 환경 구축 하기
  - OpenMP와 인사하기!



```
Microsoft Visual Studio Debug Console  
[Thread 0/4] Hello OpenMP!  
[Thread 1/4] Hello OpenMP!  
[Thread 2/4] Hello OpenMP!  
[Thread 3/4] Hello OpenMP!
```

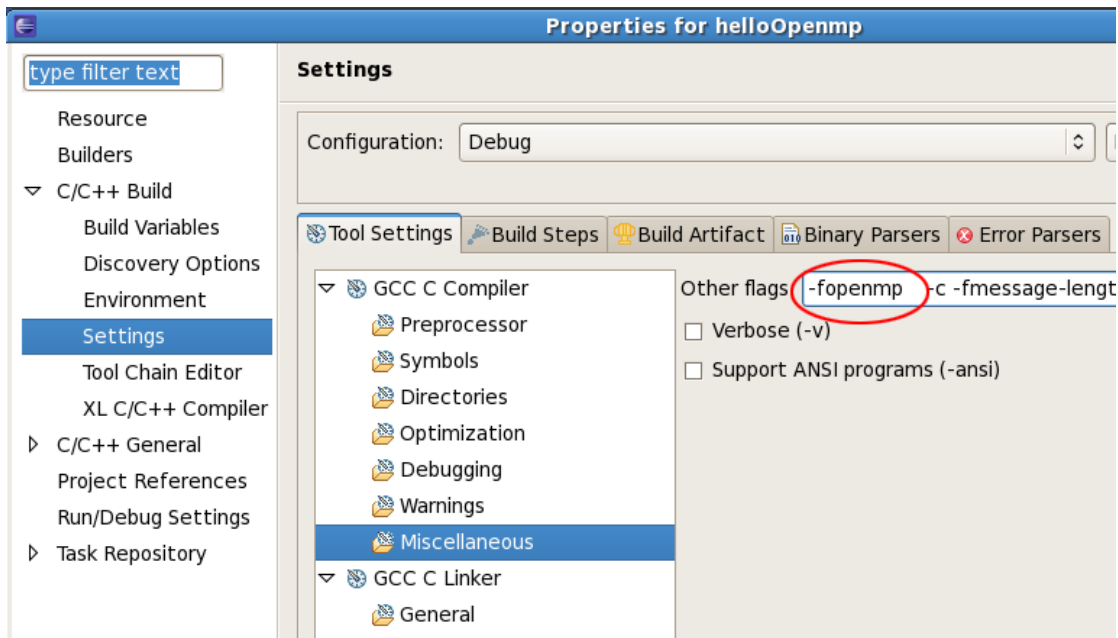
[목표 출력 결과]

# 개발/컴파일 환경 설정 - Visual Studio



# 개발/컴파일 환경 설정 - 기타 환경

- 본인이 사용하는 **Compiler**의 **OpenMP** 사용 옵션 확인
- Eclipse CDT [[Link](#)]
- GCC [[Link](#)]



# OpenMP 기초

- Parallel region 만들기 `#pragma omp parallel`
- 사용할 스레드의 수 지정 `#pragma omp parallel num_threads(n)`
- 기초 함수
  - int `omp_get_num_threads()` / 전체 스레드의 수 반환
  - int `omp_get_thread_num()` / 자신의 스레드 ID 반환



# Hints

```
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>

int main(void) {

    #pragma omp parallel num_threads(4)
    {
        printf("Hello OpenMP\n");
    }
    return 0;
}
```



Microsoft Visual Studio Debug Console

```
[Thread 0/4] Hello OpenMP!
[Thread 1/4] Hello OpenMP!
[Thread 2/4] Hello OpenMP!
[Thread 3/4] Hello OpenMP!
```

Lab. 1-2

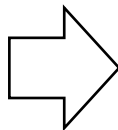
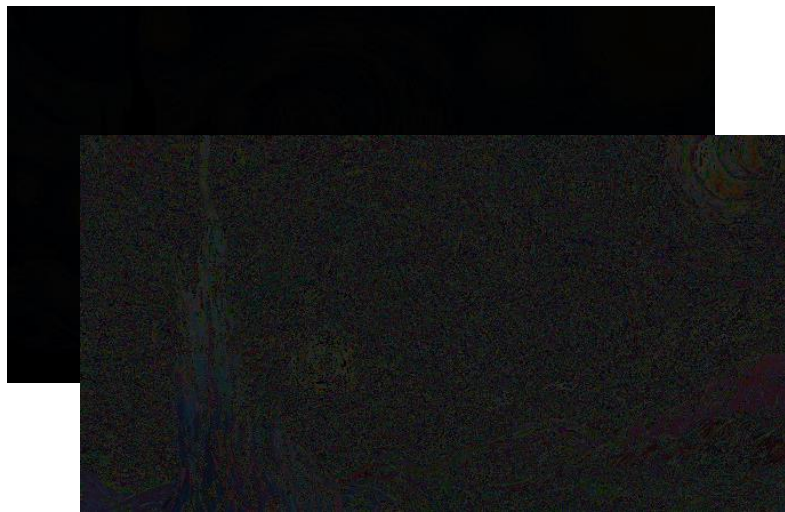
# Encrypted Image

Your First Parallel Computing Algorithm

# Lab 1-2. Encrypted Image



- 암호화 된 이미지 복원 하기



입력: 암호화 파일 A, B

출력: 복원된 그림 데이터

# Lab 1-2. Encrypted Image

- 암호화 된 이미지 복원 하기
- 과제 목표
  - 병렬처리의 개념 이해 하기
  - 일 분배 및 스레드의 개념 잡기
  - 병렬처리의 유용성 확인하기



Pixabay로부터 입수된 [Peggy und Marco Lachmann-Anke](#)님의 이미지입니다.

# Lab 1-2. Encrypted Image

- 과제 수행 방법

- 지금까지 배운 내용만 사용 해서 문제를 해결 할 것
  - 수업에서 다루지 않은, directive, clause, openMP 함수 사용 금지
  - 지금까지 배운 것
    - `#pragma omp parallel num_threads(n)`
    - `int omp_get_num_threads()`
    - `int omp_get_thread_num()`

# 과제 수행을 위해 필요한 파일들 (필수)

- EL의 과제 글을 통해서 다운로드
- 입력파일 : InputA.data, InputB.data

- 암호화된 이미지 파일
  - ImageSize.txt에 이미지 크기가 있음

- 복원된 이미지에는 메시지가 숨겨져 있음
- 보고서 제출시 메시지 내용을 적을 것

- 소스코드 템플릿

- 전체를 프로젝트에 포함해야 함

common.h

DS\_definitions.h

DS\_timer.cpp

DS\_timer.h

main.cpp

수정할 파일

# 소스코드 설명

## • 구현된 내용

- 데이터 읽기 (line 24~30)
- 직렬 처리 코드 (line 43~45)
- 병렬 처리 결과 검증 (line 63~77)
- 병렬 처리 결과 파일로 출력하기 (81~84)
- 직렬 및 병렬처리 시간 측정 및 출력

```
for (int i = 0; i < dataSize; i++) {  
    DECRYPT_ALGORITHM(A[i], B[i], serialC[i]);  
}
```

## • 작성할 내용

- 병렬처리 코드 (line 49~60)
  - 결과는 parallelC[]에 저장 할 것

Hint: `DECRYPT_ALGORITHM(A[i], B[i],parallelC[i]);`

# 소스코드 설명

## • 인자 구성

- [암호파일A] [암호파일A] [이미지너비] [이미지 높이] [결과를 출력할 파일명]
- 예) Lab1-2.exe InputA.data InputB.data 4096 4096 output.data

## • 화면 출력내용

- 병렬처리 결과 검증 결과
- 직렬/병렬처리 시간 비교

**The results is correct - Good job!**

Yo **The result is not correct! :(**  
Your computer has 8 logical cores

```
*
*
*      DS_timer Report      *
** * The number of timer = 2, counter = 2
Se **** Timer report ****
Pa Serial Algorithm : 24.50570 ms (24.50570 ms)
** Parallel Algorithm : 12.45400 ms (12.45400 ms)
* **** Counter report ****
Th *      End of the report      *
The decryption result was written to output.data
```



# 유틸리티 프로그램

- **Viewer.exe (필수)**

- 입력 및 복원 이미지를 확인하는 프로그램
- 사용법 : Viewer.exe [데이터파일] [너비] [높이] [저장할파일(op)]
  - 4번째 인자 입력 시, 결과를 이미지 파일로 저장해 줌

- **Encryptor.exe (Optional)**

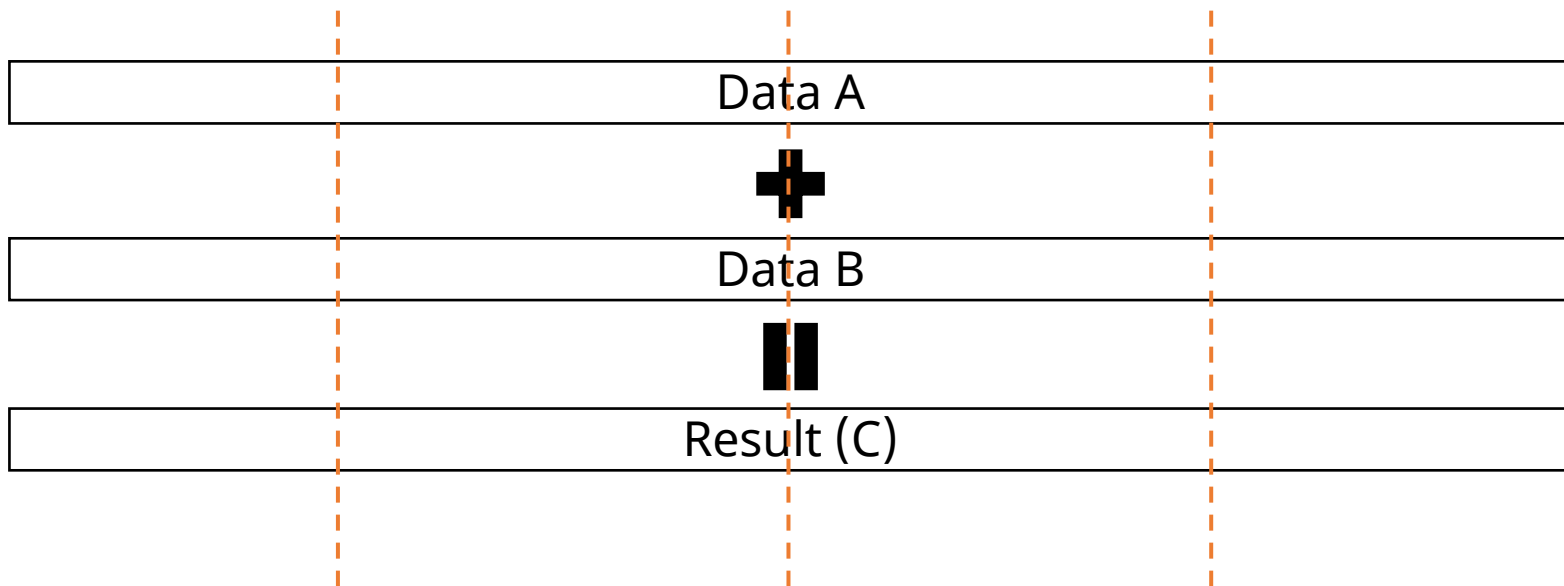
- 임의의 이미지를 두개의 파일로 암호화 하는 프로그램
- 사용법 : Encryptor.exe [입력이미지] [암호화파일이름Prefix]

- \* 윈도우 외 다른 환경 사용자는 직접 build 해서 사용 or 윈도우 가상 머신 사용 권장
  - git repository [[link](#)]
  - OpenCV 필요

# Hints

- **Vector Sum**

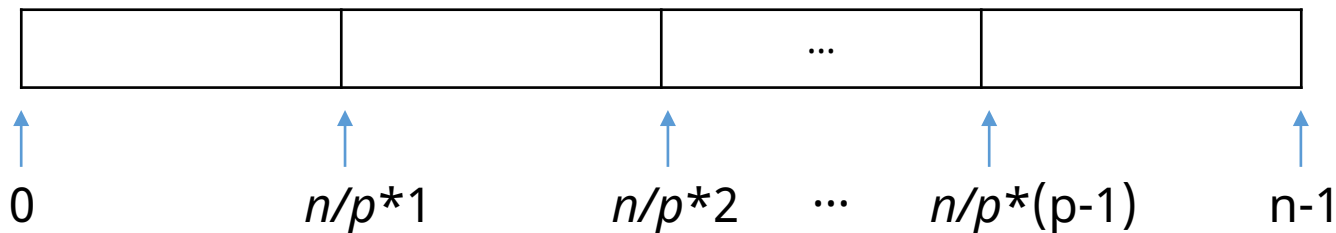
```
#define DECRYPT_ALGORITHM(a,b,c) (c += a*50 + b)
```



# Hints

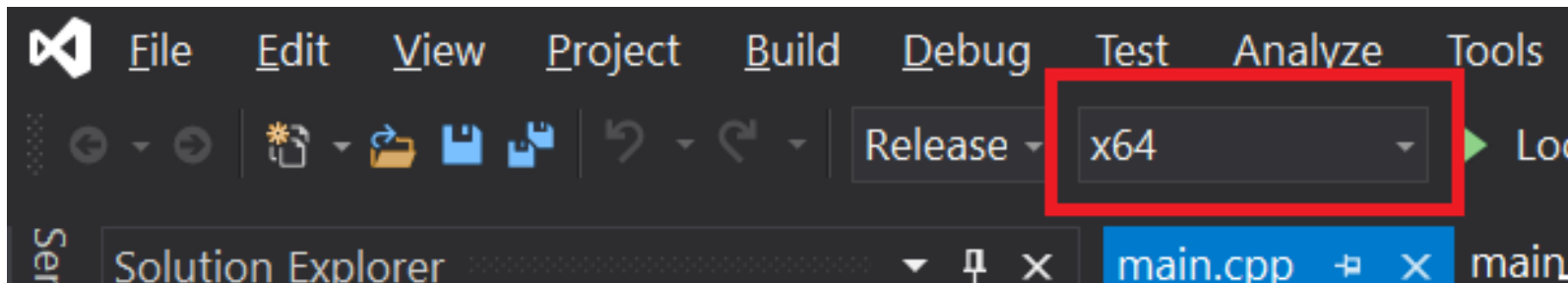
- **Data parallelism**

- $p$  스레드,  $n$  개의 데이터
- 각 스레드가 처리 할 데이터의 수  $m = n/p$
- $\text{start}[\text{tID}] = m * \text{tID}$ ,  $\text{end}[\text{tID}] = m * (\text{tID} + 1)$



# Note!

- 프로젝트를 x64로 만들어야 정상작동함



# Bonus Points!



- Lab 1-2를 해설하는 영상을 제작해서 제출 시,  
보너스 포인트(기타 점수 영역) **+2점**
  - 영상은 소스코드만으로 설명 하는 것이 아닌, 슬라이드를 만들어 설명해야 함
    - 영상 길이는 5분 이상
    - 제출된 영상은 온라인에 게시 및 다른 수강생들에게 공유될 예정
- 제출 방법
  - 영상 원본을 조교에게 이메일 제출
  - 기한: ~3/17(금) 23:59

# Q & A



[그림 출처: illustAC ([link](#))]

# 이미지 출처

- 본 슬라이드에 사용된 이미지들은,
  - 본 과목의 주재인 창의적 공학설계(김은경 지음, 한빛미디어) 및
    - [도서 링크](#)
  - 다음 출처로 부터 가져 왔으며, 상업적 사용 및 출처 표시 제한이 없는 이미지만 사용 했습니다
    - [Pixarbay](#)
    - [illustAC](#)



Pixabay로부터 입수된 Peggy und Marco Lachmann-Anke님의 이미지입니다.