

PTX 코드 추출

정보컴퓨터공학과 권혁동

대상 프로그램 작성

- PTX 코드를 추출하고자 하는 프로그램 작성
 - 예시 프로그램은 단순한 덧셈
 - GPU 상에서 연산
 - CPU, GPU간 자료 이동은 구현하지 않음
- 빌드하기에 앞서 몇 가지 설정을 진행

```
#include "cuda_runtime.h"
#include "device_launch_parameters.h"
#include <stdio.h>

__device__ static void add_function(int a, int b, int *c)
{
    *c = a + b;
}

__global__ void call(void)
{
    int x = 10, y = 20, z = 0;

    add_function(x, y, &z);

    printf("%d\n", z);
}

int main()
{
    call << 1, 1 >> > ();

    return 0;
}
```

프로젝트 설정

- 프로젝트에 우클릭 → 속성에 진입
- 또는 Alt + Enter 입력으로 속성에 진입



프로젝트 설정

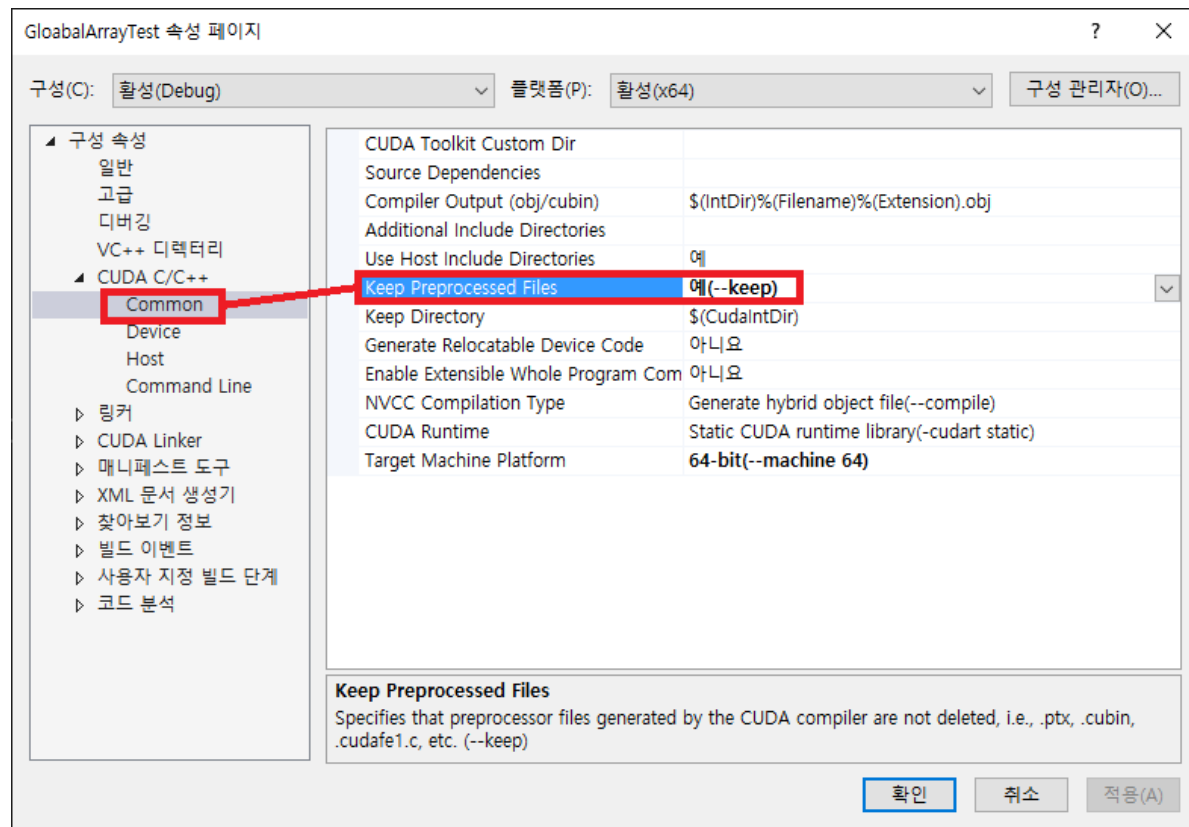
- CUDA C/C++

→ Common

→ Keep Preprocessed Files

→ Yes(--keep)

- 프리프로세스 파일을 남기는 옵션



프로젝트 설정

- CUDA C/C++

- Device

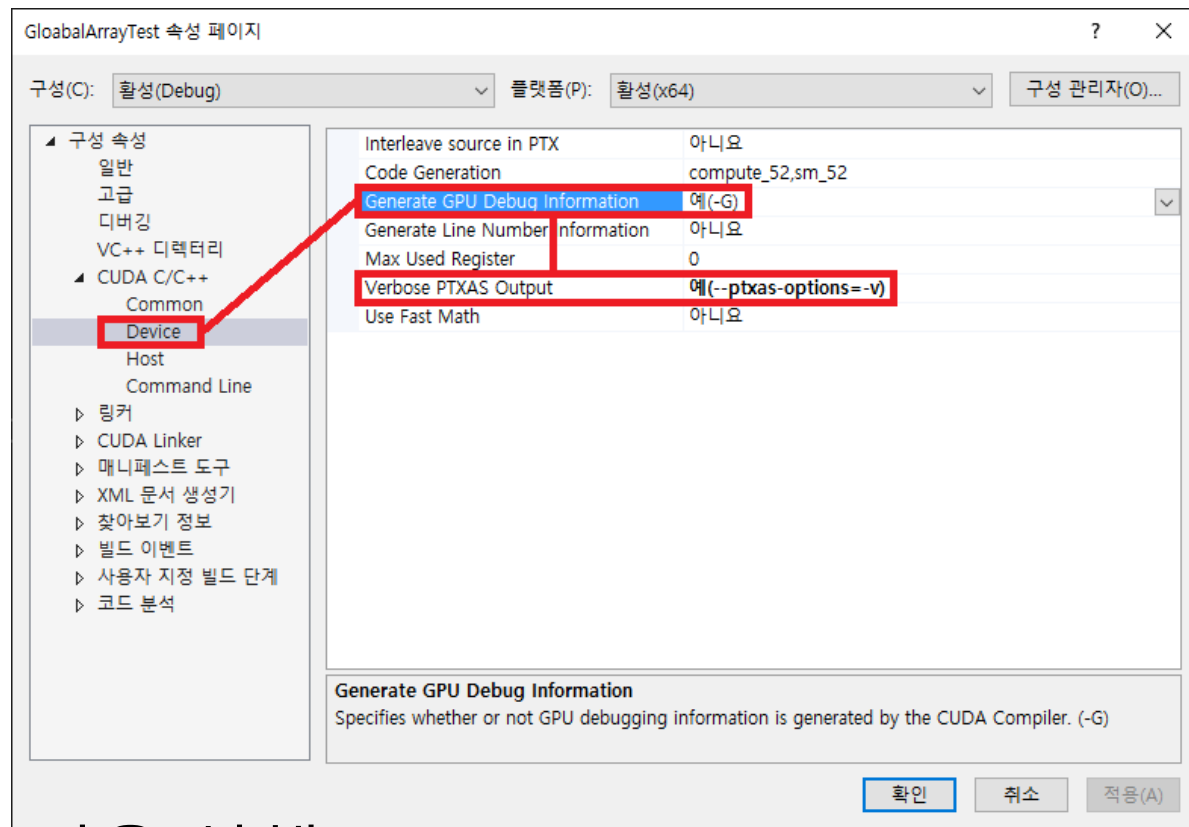
- Generate GPU Debug Information

- Yes(-G)

- Verbose PTXAS Output

- Yes(--ptxas-options=-v)

- 디버그 정보를 남기고 PTX 어셈블리 출력을 실행



PTX 코드 확인

- 실제 출력물은
(프로젝트명) / (프로젝트명) / x64 / Debug 디렉토리에 존재
- 파일명은 .cu 파일명과 동일
- 해당 파일을 텍스트 편집기로 열람
 - Atom
 - VS Code
 - Notepad++ 등등
- 본 문서에서는 Notepad++로 열람



| 이름 | 수정된 날짜 | 유형 | 크기 |
|--------------------------|---------------------|--------------------|---------|
| GlobalArrayTest.log | 2020-12-16 오전 11:25 | 텍스트 문서 | 1KB |
| kernel.cpp1.ii | 2020-12-16 오전 11:25 | II 파일 | 1,803KB |
| kernel.cpp1.ii.res | 2020-12-16 오전 11:25 | Compiled Resour... | 1KB |
| kernel.cpp4.ii | 2020-12-16 오전 11:25 | II 파일 | 1,478KB |
| kernel.cpp4.ii.res | 2020-12-16 오전 11:25 | Compiled Resour... | 1KB |
| kernel.cu.cache | 2020-11-19 오후 4:28 | CACHE 파일 | 2KB |
| kernel.cu.obj | 2020-12-16 오전 11:25 | 3D Object | 104KB |
| kernel.cu.obj.res | 2020-12-16 오전 11:25 | Compiled Resour... | 1KB |
| kernel.cu-354870077.deps | 2020-12-16 오전 11:25 | DEPS 파일 | 7KB |
| kernel.cudafe1.c | 2020-12-16 오전 11:25 | C Source File | 1KB |
| kernel.cudafe1.cpp | 2020-12-16 오전 11:25 | C++ Source File | 758KB |
| kernel.cudafe1.gpu | 2020-12-16 오전 11:25 | GPU 파일 | 13KB |
| kernel.cudafe1.stub.c | 2020-12-16 오전 11:25 | C Source File | 2KB |
| kernel.fatbin | 2020-12-16 오전 11:25 | FATBIN 파일 | 32KB |
| kernel.fatbin.c | 2020-12-16 오전 11:25 | C Source File | 91KB |
| kernel.module_id | 2020-12-16 오전 11:25 | MODULE_ID 파일 | 1KB |
| kernel.ptx | 2020-12-16 오전 11:25 | PTX 파일 | 98KB |
| kernel.sm_52.cubin | 2020-12-16 오전 11:25 | CUBIN 파일 | 58KB |
| vc142.pdb | 2020-12-16 오전 11:25 | Program Debug ... | 140KB |

PTX 코드 확인

- 전체 코드 줄은 약 1만줄
- 예시 프로젝트에서 생성한 내용을 확인 가능
 - 3개의 매개변수를 보내는 부분
 - 정수형 변수 A
 - 정수형 변수 B
 - 정수형 포인터 변수 C
 - 덧셈 과정
 - 덧셈 후 주소 값에 저장

```
func_begin6:
    .loc    1 0 0

    .loc 1 7 1
    ld.param.u32    %r1, [_ZN36_INTERNAL_14_kernel_cppl_ii_Z4callvl2add_functionEiiPi_param_0];
    ld.param.u32    %r2, [_ZN36_INTERNAL_14_kernel_cppl_ii_Z4callvl2add_functionEiiPi_param_1];
    ld.param.u64    %rd1, [_ZN36_INTERNAL_14_kernel_cppl_ii_Z4callvl2add_functionEiiPi_param_2];
func_exec_begin6:
    .loc    1 9 5
tmp12:
    add.s32    %r3, %r1, %r2;
    st.u32    [%rd1], %r3;
    .loc    1 10 1
    ret;
tmp13:
func_end6:
```

기타

- PTX 코드의 분량이 매우 방대하며 PTX 명령어의 종류도 매우 많음
- 따라서 **PTX 코드를 추출할 프로젝트는 단순하게 작성**하는 것이 중요
 - 특히 변수, 함수명은 특이하게 작성하는 것을 권장
 - PTX 명령어와 이름이 겹칠 경우 검색이 어려울 수 있기 때문
- 일부 **작성한 코드와 어셈블리 코드 상의 괴리가 존재**할 수 있음
 - 예시 프로젝트에서는 int형이지만 PTX 코드에서는 unsigned int로 작성됨
- 추출된 PTX 코드의 전체를 보는 것이 아닌 **일부만 보는 것이 필요**