



어셈블리 프로그래밍 (Assembly Programming)

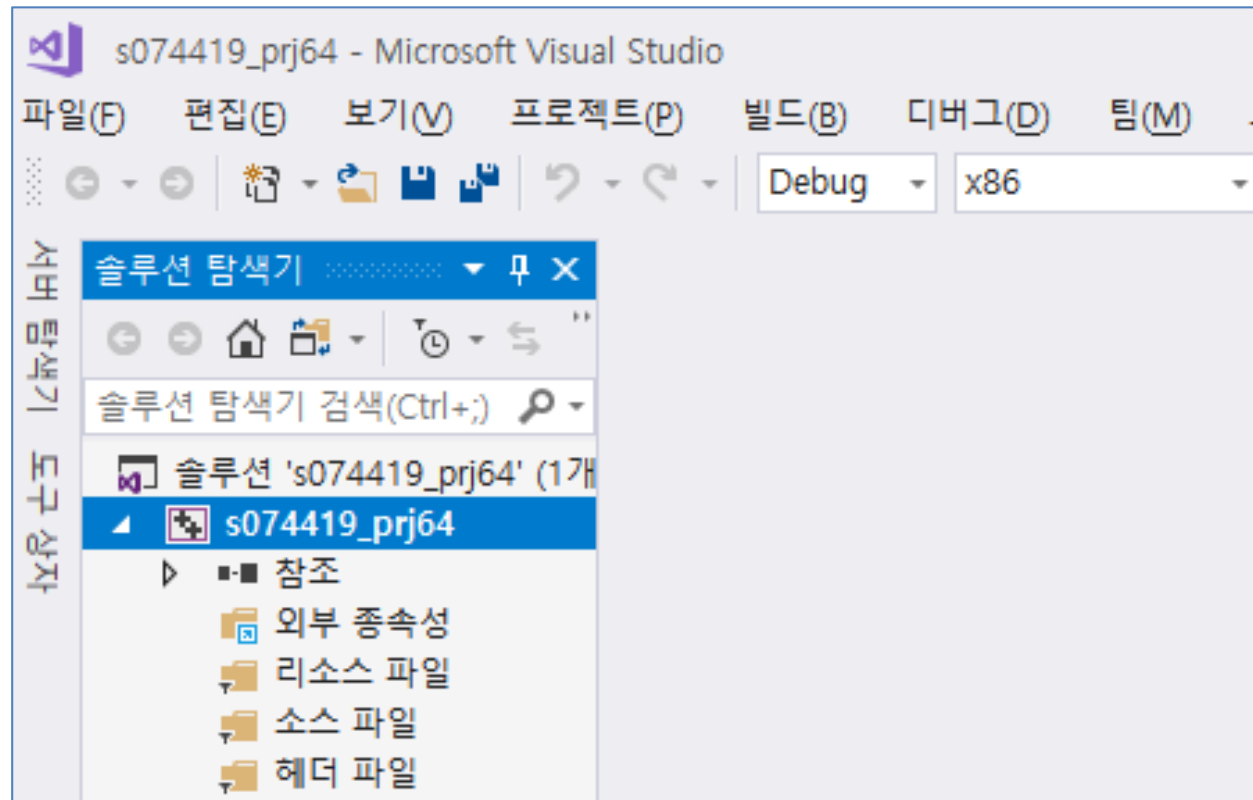
64bit Assembly Programming

CSE3030 Dept. of CS&E
Sogang University

Project Setting for 64 bit Assembly

◆ Project 생성

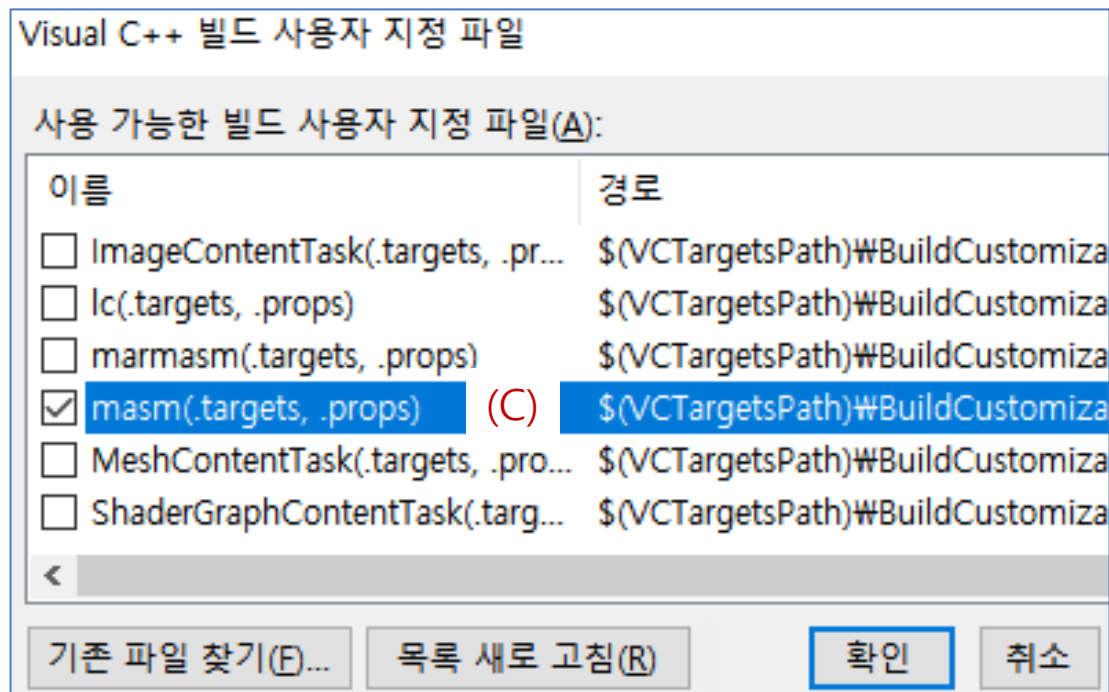
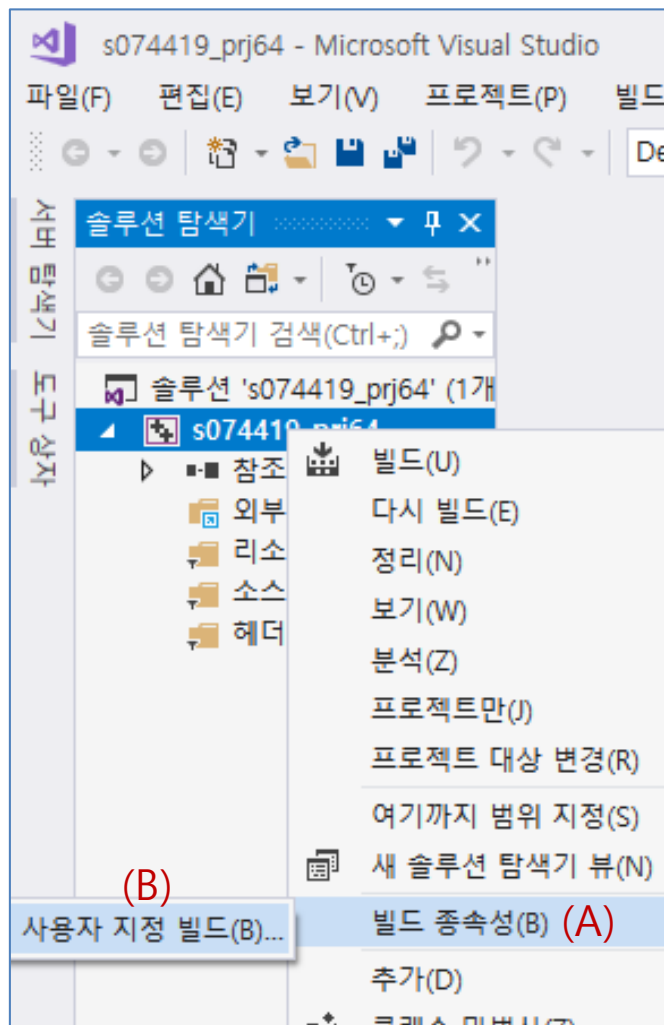
- ◆ 32-bit 어셈블러에서 처럼 빈 프로젝트 하나를 생성한다(프로젝트 이름은 **snnnnnn_prj64**로 하자⁽¹⁾).



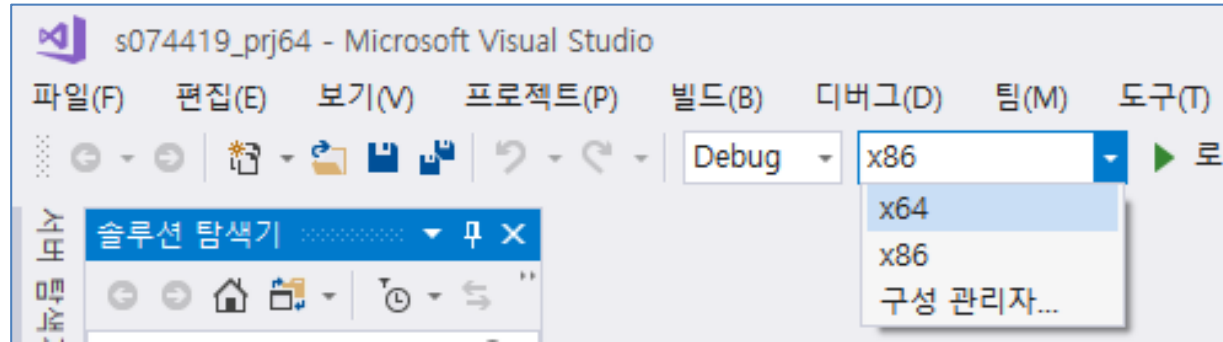
(1) nnnnnn은 자신의 학번 뒤 6자리).

◆ Project Setting for 64 bit Assembly Programming

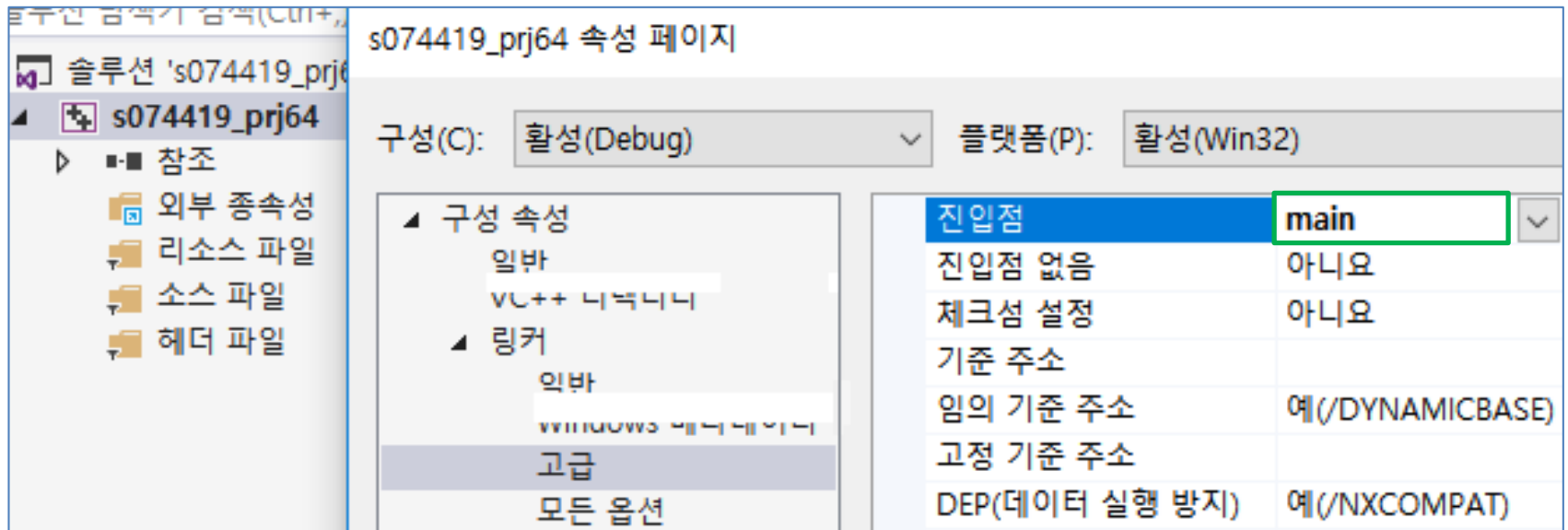
◆ 프로젝트 ..._prj 우클릭 → 빌드 종속성(A) → 사용자 지정 빌드(B) →
masm(.targets,.props) 체크(C) → 확인



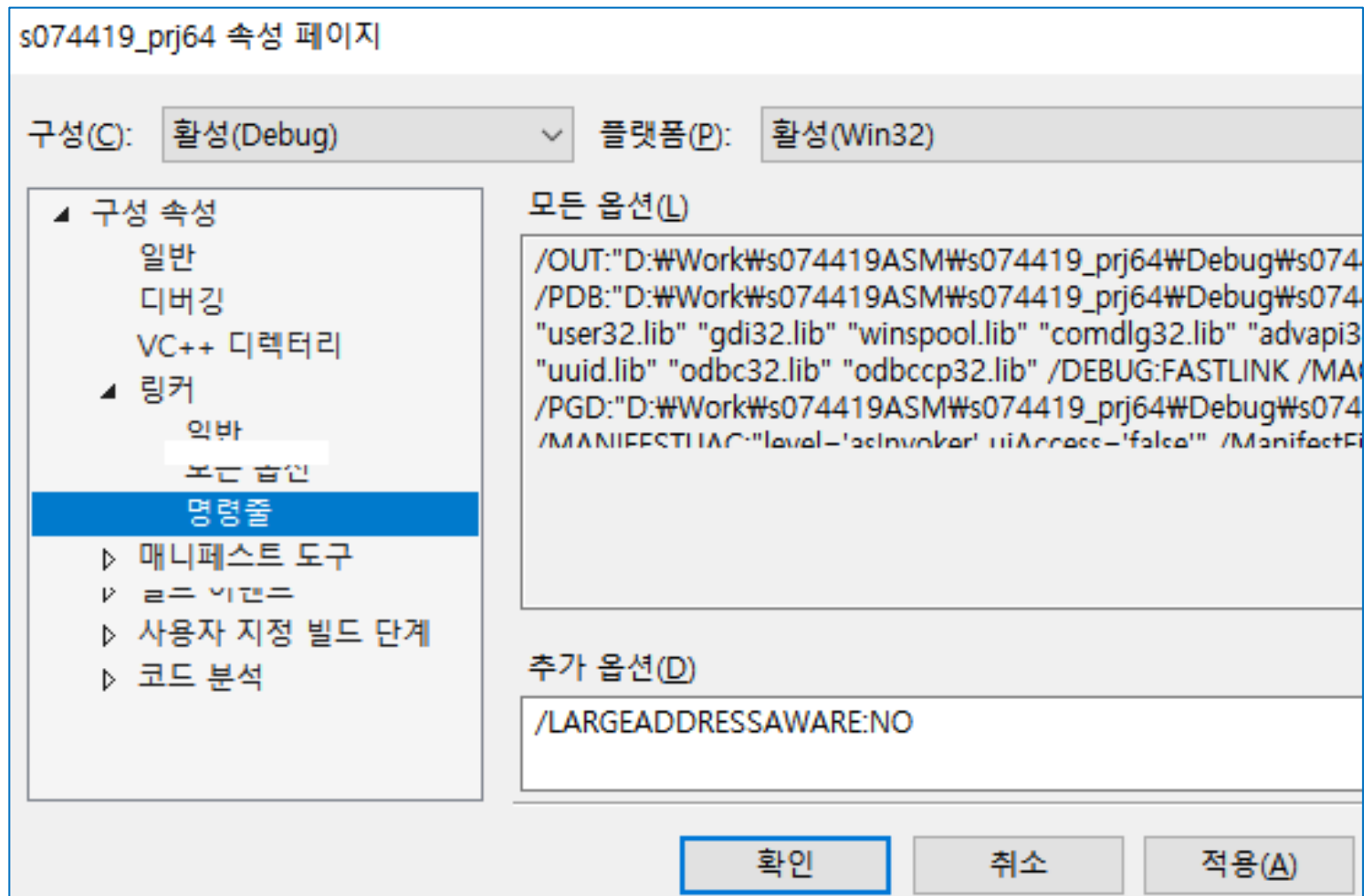
- ◆ 아래와 같이 플랫폼을 x64로 바꾼다.



- ◆ 진입점 설정 : 프로젝트속성 → 링커 → 고급 클릭 → 속성 창의 첫 줄 진입점 항목에 main 입력 → 확인.



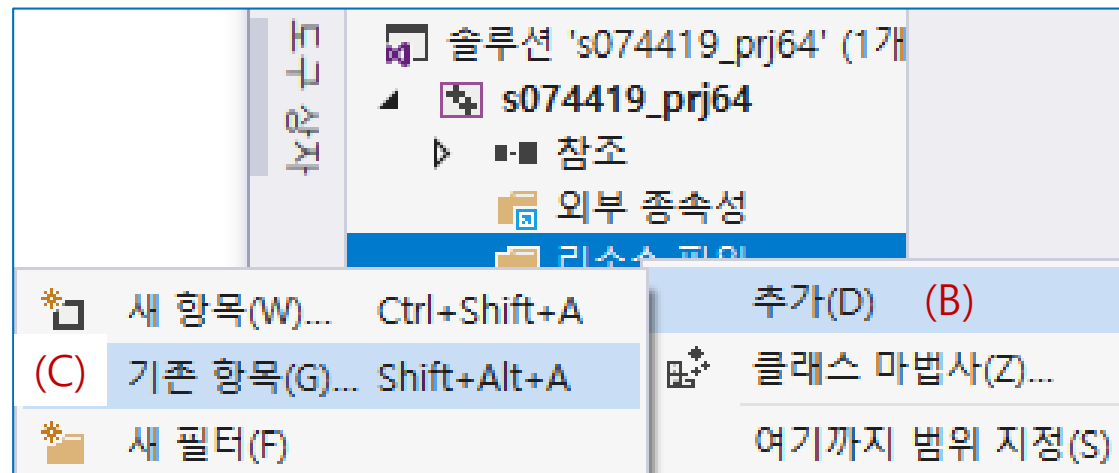
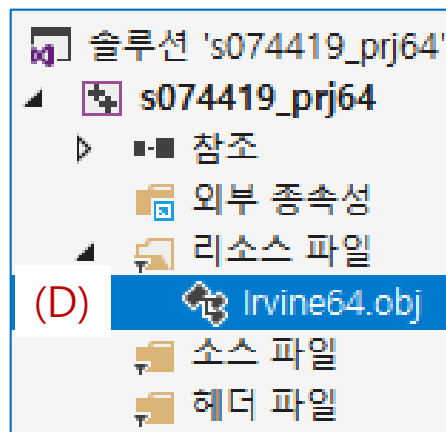
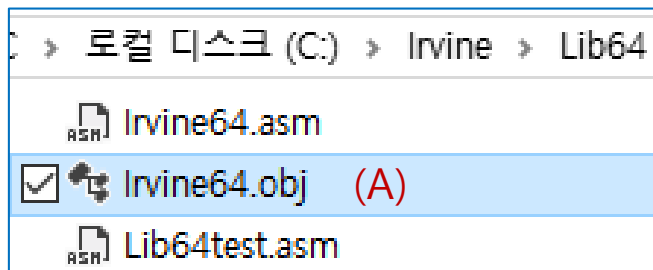
- ◆ 추가 옵션 세팅: 프로젝트속성 → 링커 → 클릭 명령줄 → 추가 옵션 칸에 **/LARGEADDRESSAWARE:NO** 를 입력 → 클릭 확인
- ◆ 이 설정은 2G보다 큰 주소 처리를 안하겠다는 의미인데, 교과서에서 제공하는 라이브러리를 사용하려면 어쩔 수 없다.



64 bit 어셈블리 프로그래밍

◆ Irvine 라이브러리 등록

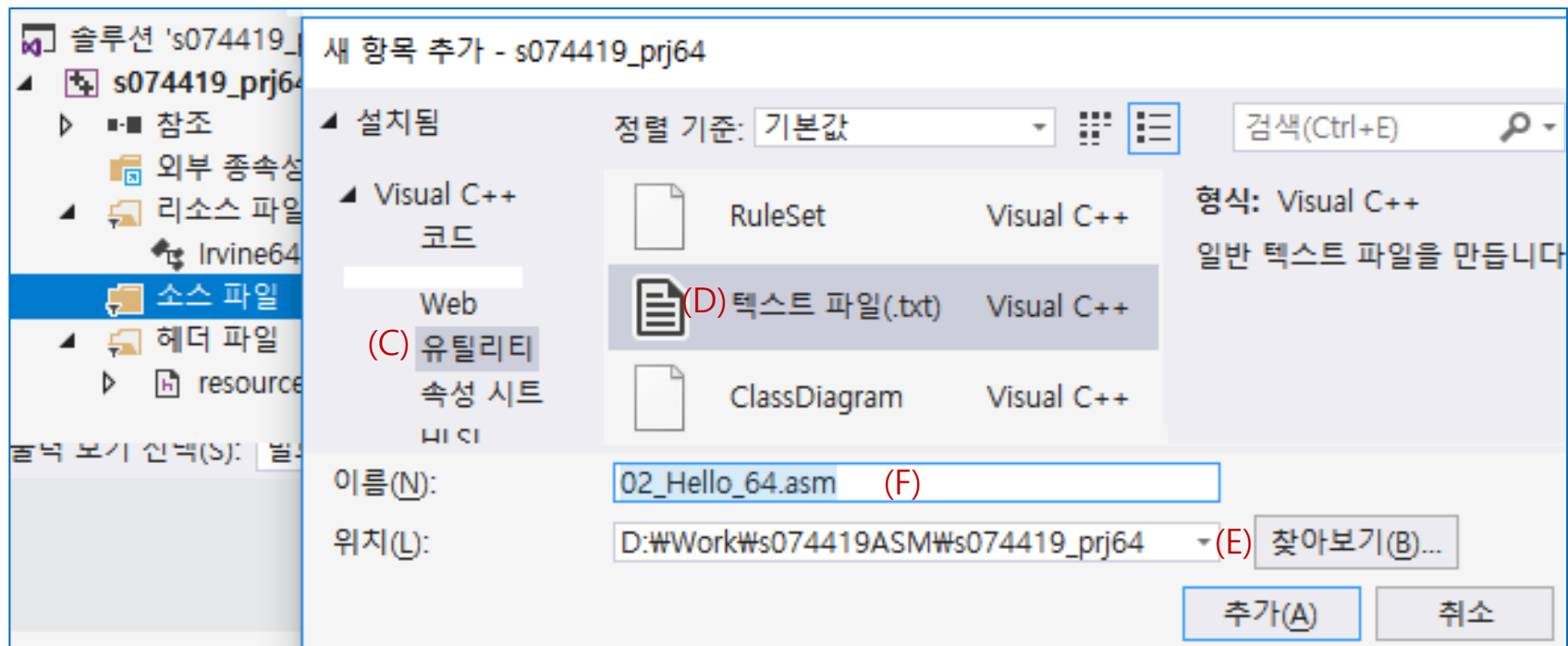
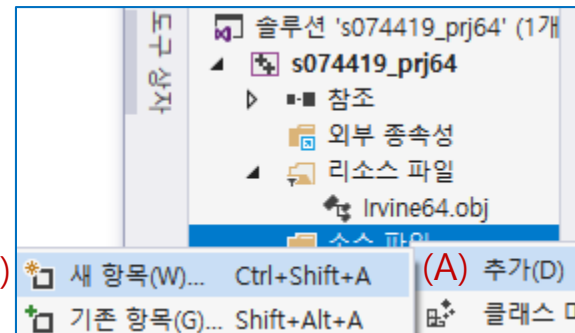
- ◆ 교과서 5.5항 (210 쪽)의 함수 라이브러리를 사용하려면, Irvine 폴더의 **Irvine64.obj**를 프로젝트에 등록하여야 한다(A).
- ◆ 또한, 사용하고자 하는 함수 이름을 **proto**로 선언해야 한다.
- ◆ 프로젝트 ..._prj → 리소스 파일 → 추가(B) → 기존 항목(C) → **Irvine64.obj** 찾아서 추가(D,1).



(1) 굳이 리소스 파일에 등록하지 않아도 된다. 소스 파일에 등록해도 된다.

◆ 새 파일 추가 및 실행

◆ 프로젝트 ..._prj → 소스 파일 → 추가(A) → 새 항목(B) → 새 항목 추가 창에서 → Visual C++ → 유틸리티(C) → 텍스트 파일(D) → 폴더 결정(E) → 파일명(02_Hello_64.asm) 입력(F).



◆ 새 파일에 다음과 같이 입력한다(1)

```
ExitProcess proto
WriteString proto
```

```
.data
```

```
sum qword 0
Hi byte "Hello!", 0
```

```
.code
```

```
main proc
```

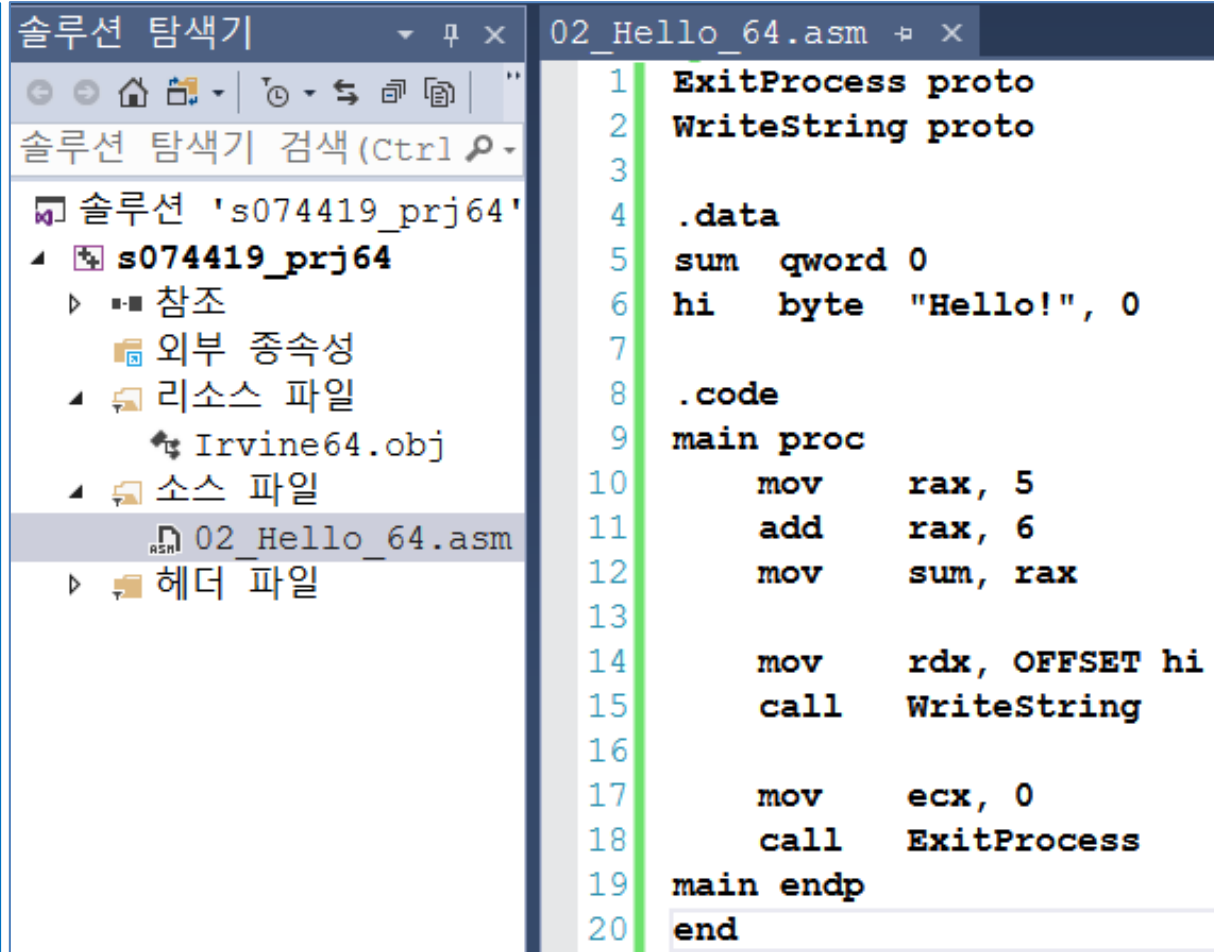
```
    mov rax, 5
    add rax, 6
    mov sum, rax
```

```
    mov rdx, OFFSET hi
    call WriteString
```

```
    mov ecx, 0
    call ExitProcess
```

```
main endp
```

```
end
```

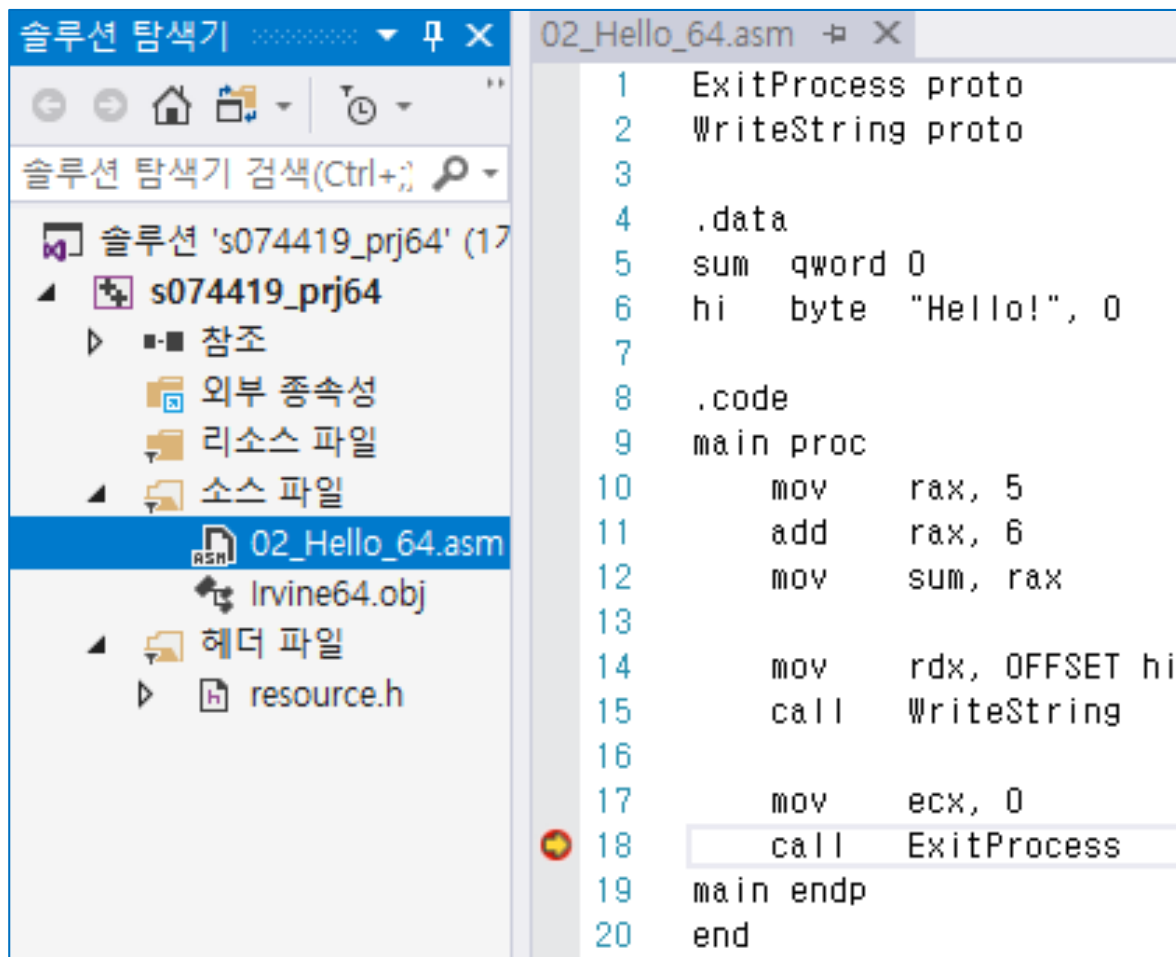


The screenshot shows an IDE with two panes. The left pane is the 'Solution Explorer' showing a project named 's074419_prj64'. It contains folders for '참조' (References), '외부 종속성' (External Dependencies), '리소스 파일' (Resource Files), '소스 파일' (Source Files), and '헤더 파일' (Header Files). The '소스 파일' folder is expanded, showing the file '02_Hello_64.asm'. The right pane is the 'Assembly Editor' showing the contents of '02_Hello_64.asm'. The code is as follows:

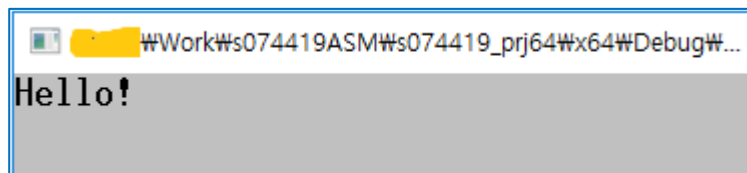
```
1  ExitProcess proto
2  WriteString proto
3
4  .data
5  sum qword 0
6  hi byte "Hello!", 0
7
8  .code
9  main proc
10     mov     rax, 5
11     add     rax, 6
12     mov     sum, rax
13
14     mov     rdx, OFFSET hi
15     call    WriteString
16
17     mov     ecx, 0
18     call    ExitProcess
19 main endp
20 end
```

(1) 좌측 내용을 복사/붙여넣기 할 경우 오류가 생길 수 있으니 이를 수정하거나 또는 직접 타이핑하세요.

- ◆ 아래 그림과 같이 breakpoint를 잡고 F5로 실행하면 **Hello!**가 출력된다.



```
1  ExitProcess proto
2  WriteString proto
3
4  .data
5  sum  qword 0
6  hi   byte  "Hello!", 0
7
8  .code
9  main proc
10     mov     rax, 5
11     add     rax, 6
12     mov     sum, rax
13
14     mov     rdx, OFFSET hi
15     call    WriteString
16
17     mov     ecx, 0
18     call    ExitProcess
19 main endp
20 end
```



#Work#s074419ASM#s074419_prj64#x64#Debug#...
Hello!



◆ 디버깅 및 실행 파일 백업

- ◆ 디버깅 방법은 32 bit 어셈블리와 동일하다.
- ◆ 방금 작성한 실행 파일은 `\s074419ASM\s074419_prj64\x64\Debug`에 생성된다.
- ◆ 디버그 폴더는 프로그램 개발을 마치면 불필요하기 때문에 삭제할 수 있다.
- ◆ 따라서, 실행 파일 `s074419_prj64.exe`를 `\s074419ASM\s074419_exe` 폴더에 `02_Hello_64.exe`로 이름을 바꾸어 보관하자.