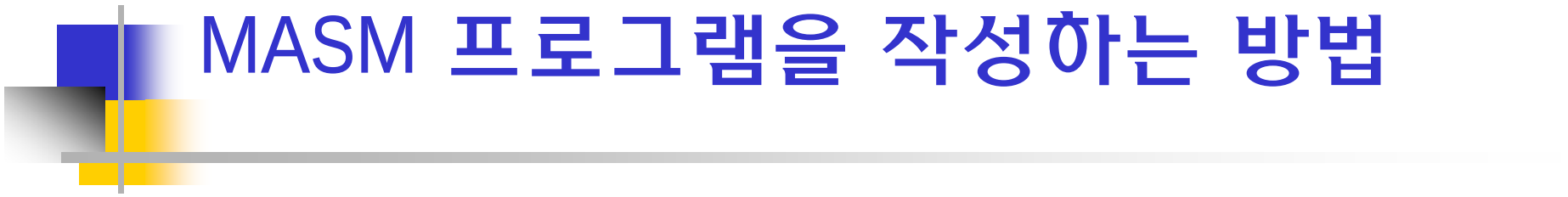
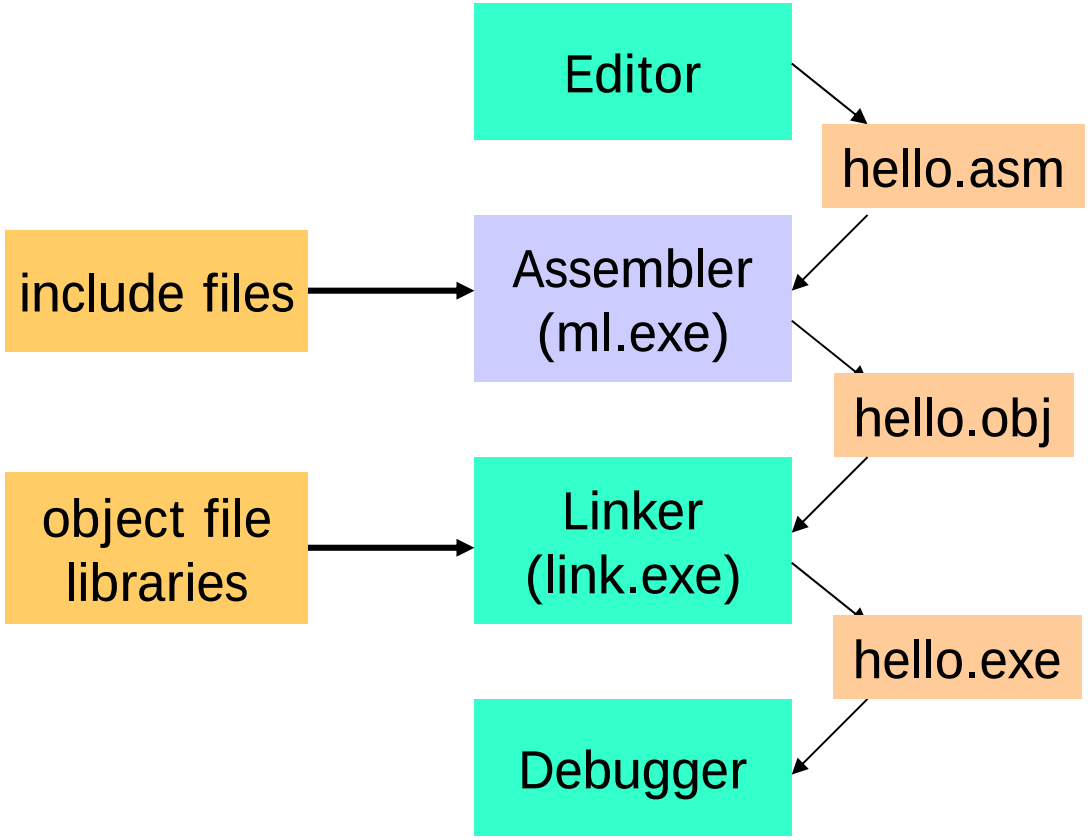


Visual Studio 2013을 사용하여 MASM 프로그램을 작성하는 방법

연세대학교
컴퓨터정보통신공학부
윤 상 균

필요한 소프트웨어


Editor
hello.asm
include files
Assembler
(ml.exe)
hello.obj
object file
libraries
Linker
(link.exe)
hello.exe
Debugger

Visual Studio 2013과 MASM

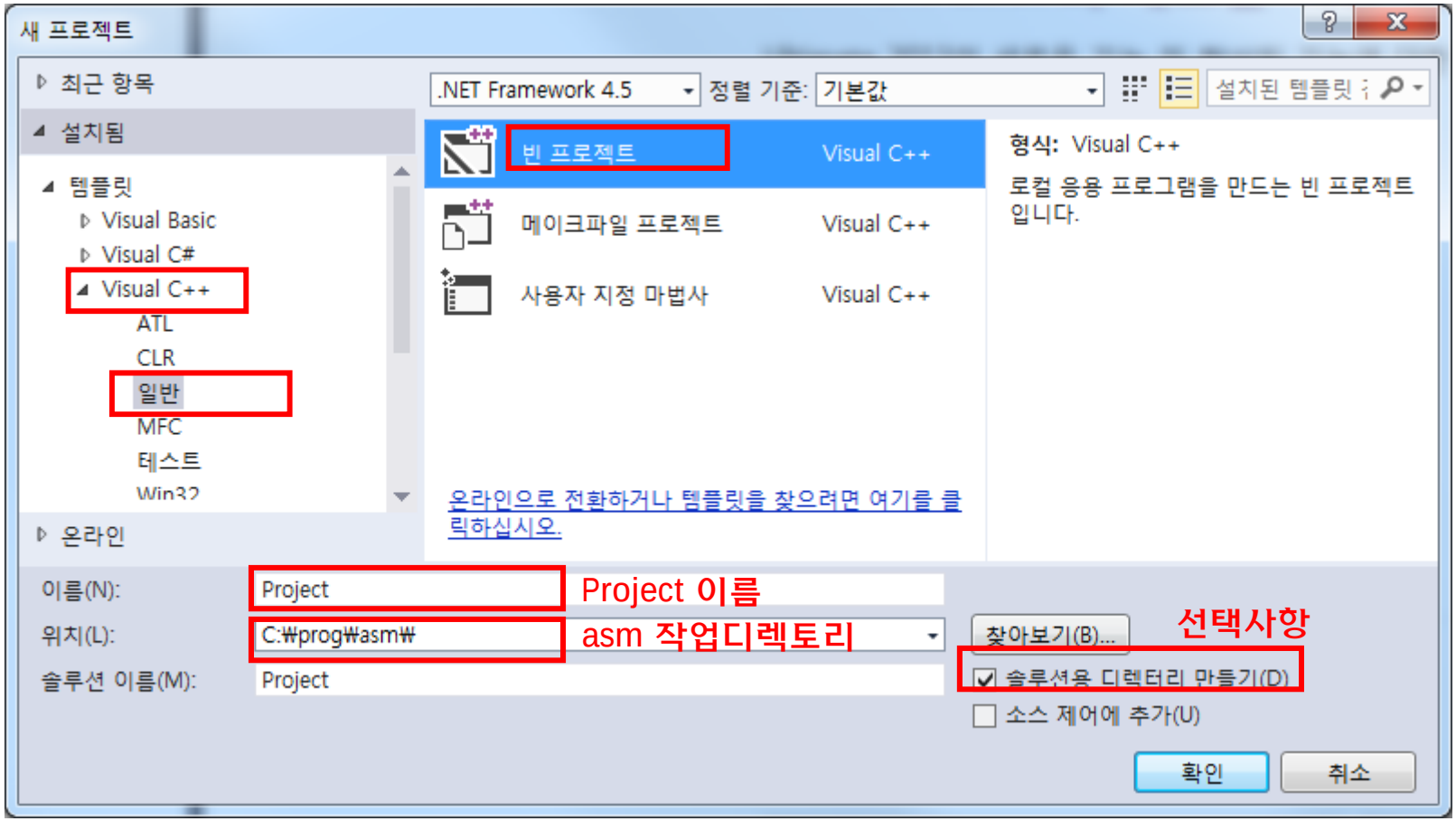
- Visual Studio 2008부터 Assembler가 포함됨
- Visual Studio 2010 이후
  - Editor, Linker, Debugger 와 Assembler를 모두 포함
- MASM 14.0, LINK 14.0
  - Visual Studio 2013에 포함됨
  - Custom Build rule에 MASM기능을 포함시켜야 함
- Visual Studio 2013에서 MASM 사용시 설정 사항
  - Custom build rules 설정 – MASM 선택
  - Assembly Language 프로그램 파일 추가 – 확장자: .asm
  - include path 설정 (assembler용)
  - library paths and library file 설정 (linker용)
  - subsystem을 “console”로 설정 (linker용)

통합개발환경 사용 방법

- Setting
  - Project name: *Hello*
  - Solution name: *Hello*
  - Source code file: *hello.asm*
  - Additional library dependencies: Irvine32.lib user32.lib
  - Used headers: Irvine32.inc
  - Additional library paths: C:\Irvine
  - Additional include paths: C:\Irvine
- 프로그램 작성 순서
  - 프로젝트 생성
  - custom build rules 지정*
  - source code file 생성 (확장자 .asm)
  - project property 지정*
  - source code file 작성
  - 실행파일 build & run

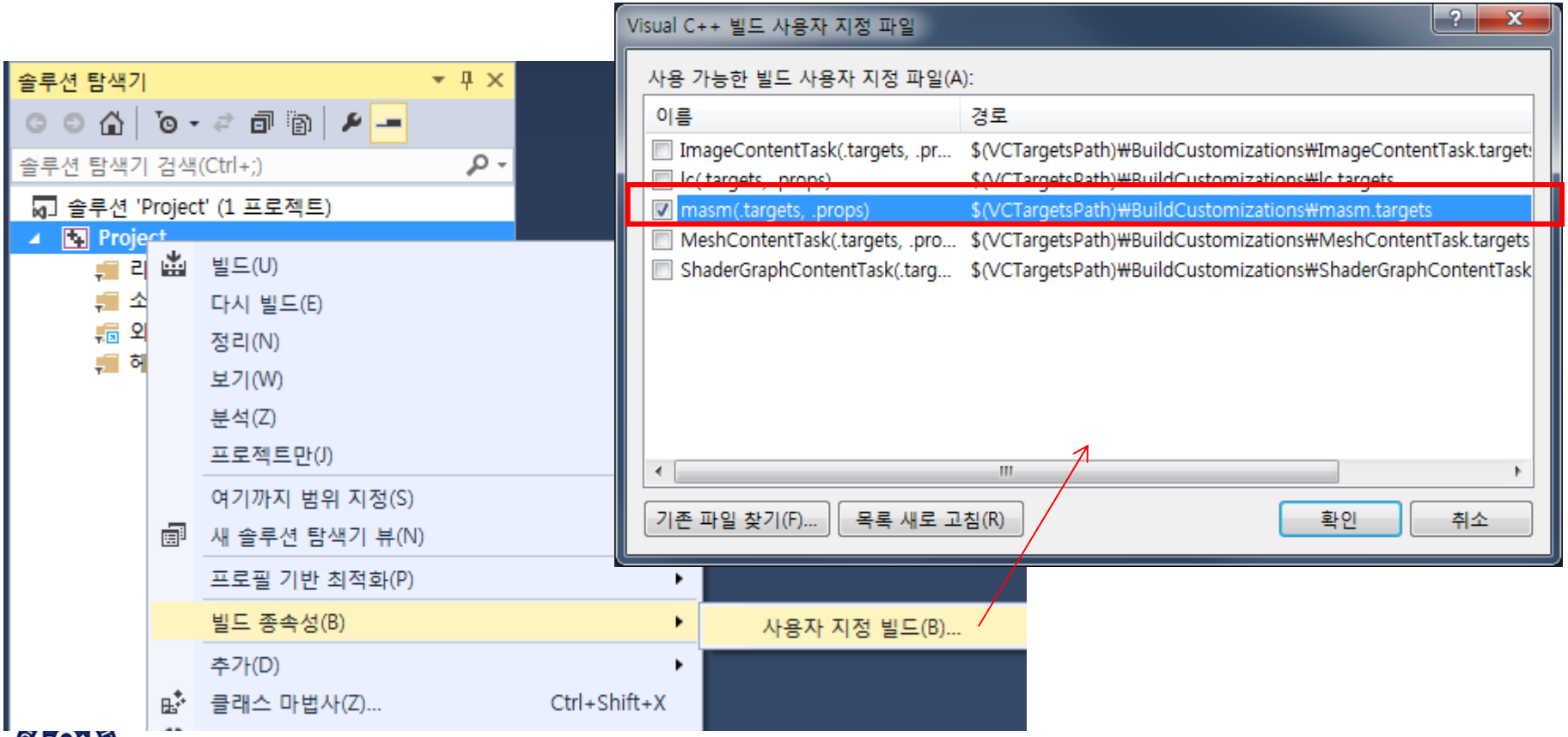
New project 생성

- [File > New > Project]
- New project **창에서** [Visual C++ > **빈 프로젝트**]



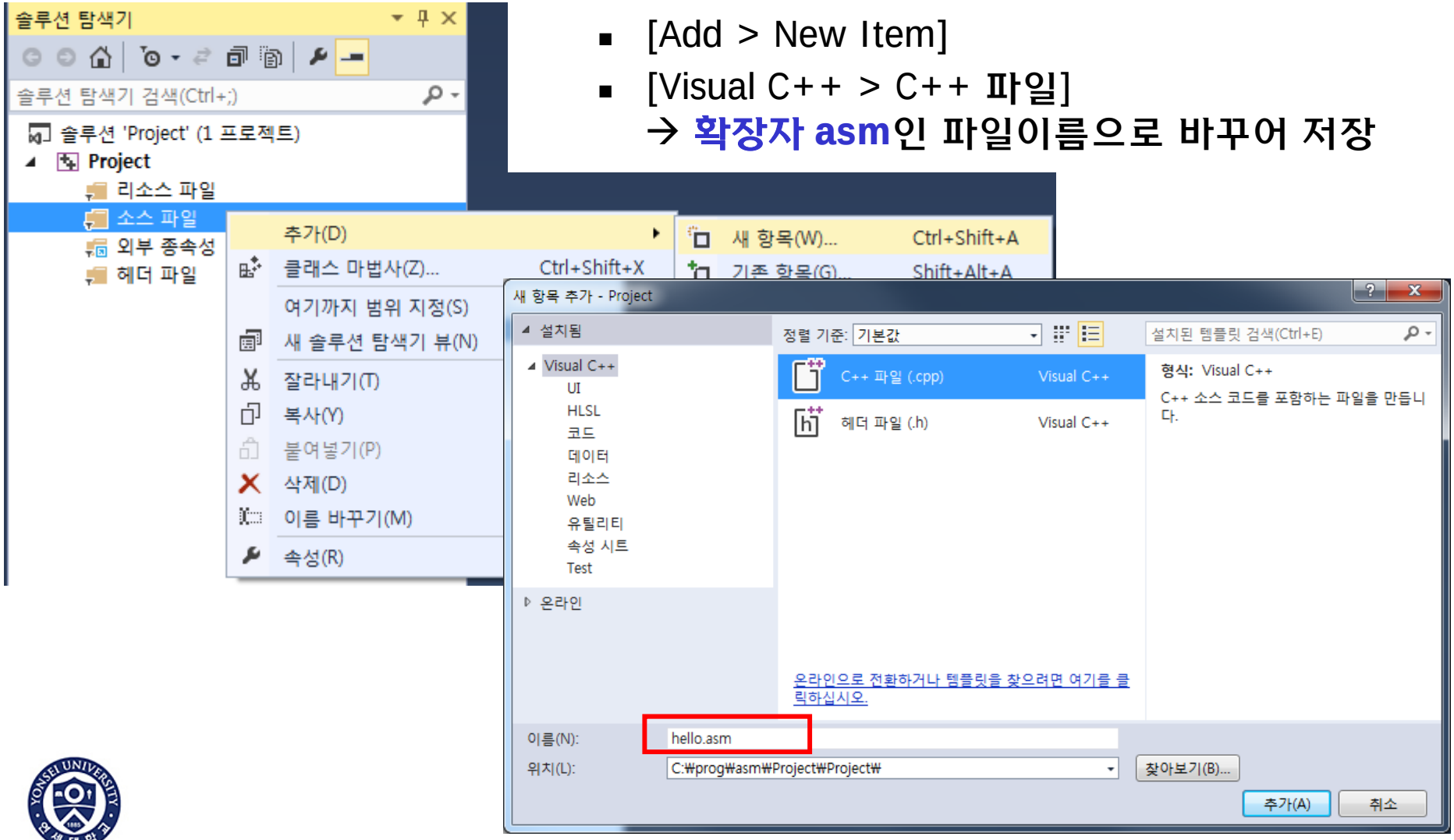
Custom Build Rules(사용자 지정 빌드)

- Solution 탐색기에서 project 이름에서 right 버튼 클릭
- [빌드종속성] → [사용자 지정 빌드] → masm 선택



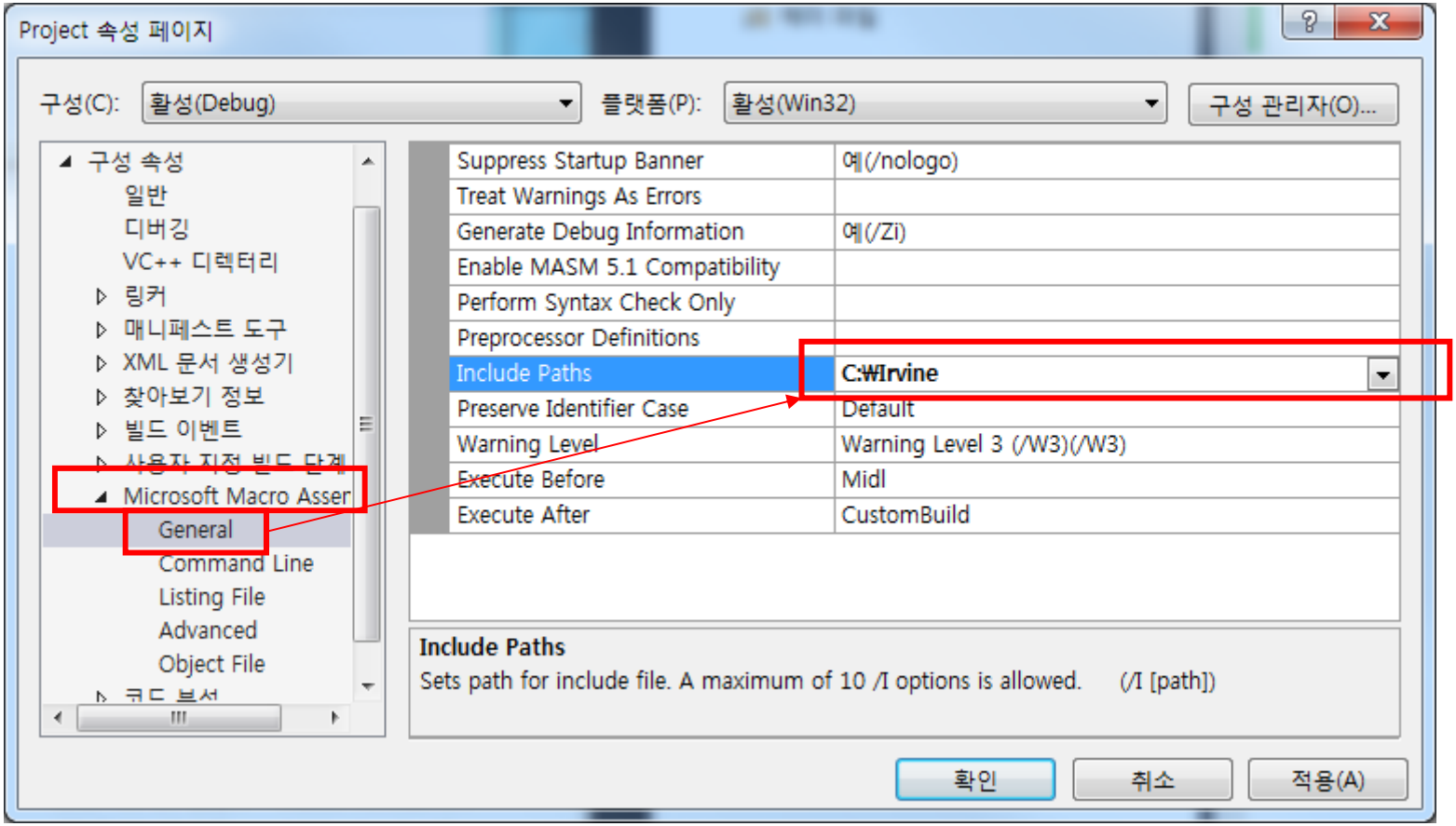
New source file 추가

- Source file 위에서 right 버튼 클릭
- [Add > New Item]
- [Visual C++ > C++ 파일]
  → **확장자 asm**인 파일이름으로 바꾸어 저장



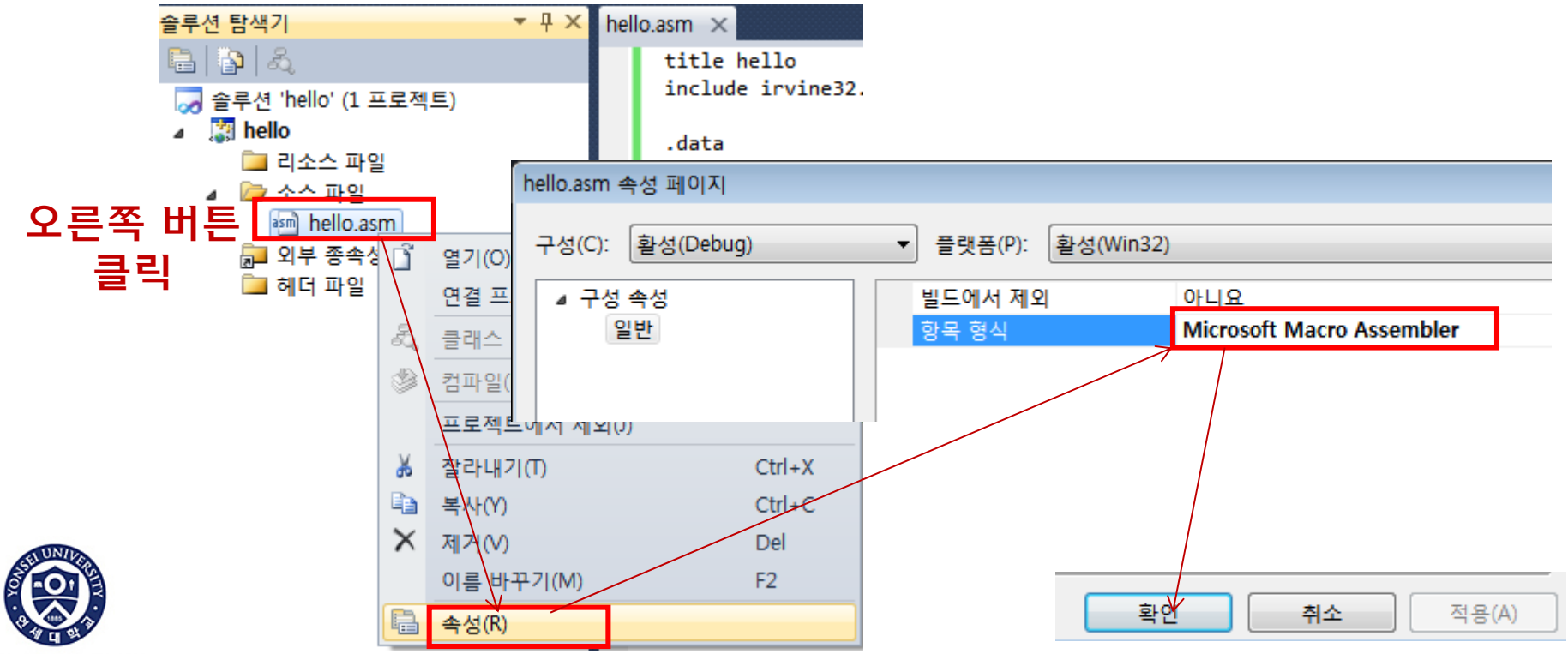
Property – MASM include path

- **[프로젝트 > 속성 > 구성속성 > Microsoft Macro Assembler >** General > Include Paths **]**
- include file **경로 입력** : **C:\irvine**



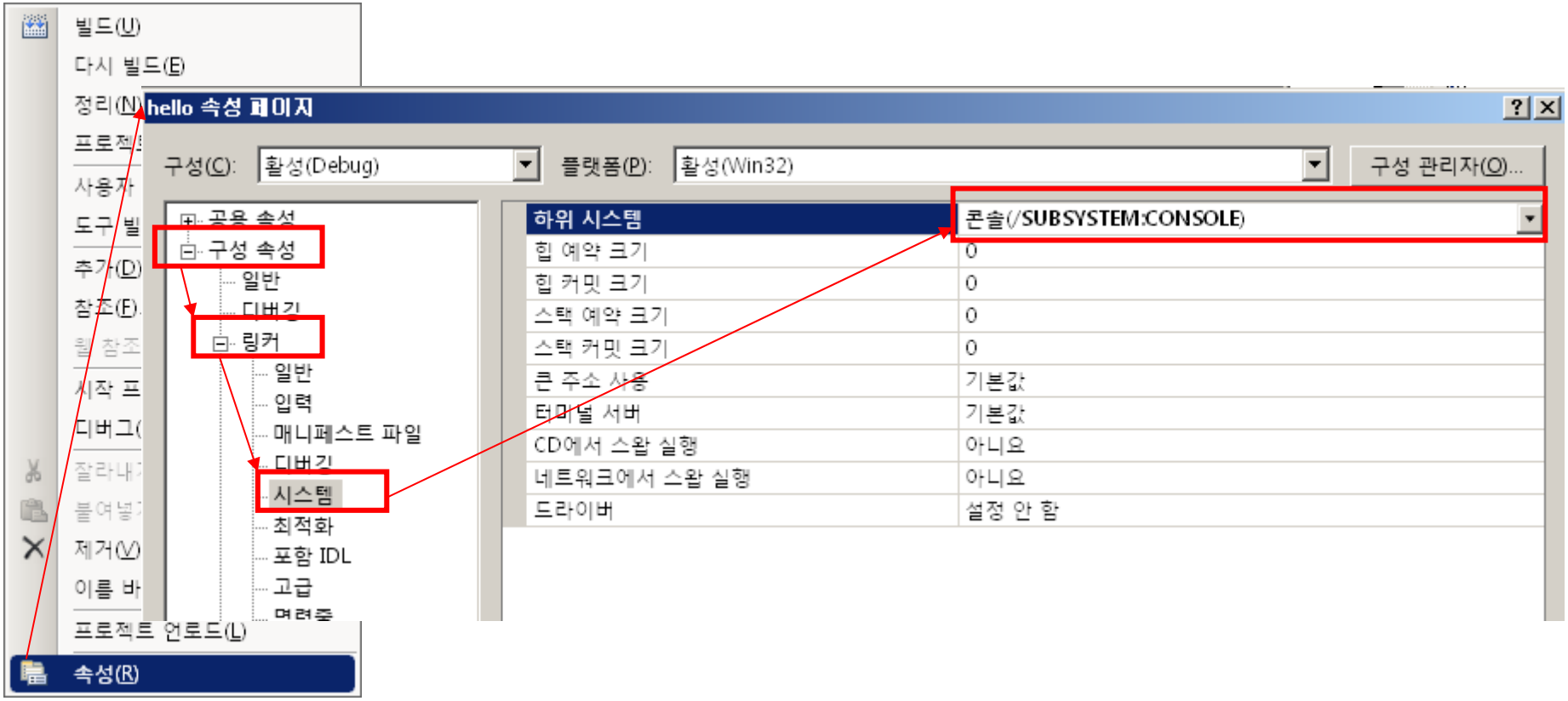
ASM 파일 속성 변경 (masm을 인식 못할 때)

- Visual Studio 2010에서는 project에 ASM 소스 파일이 추가되어야 Properties(속성)에 Macro Assembler 항목이 나타남.
- ASM 소스파일을 추가 후에도 MASM 항목이 나타나지 않으면 ASM 파일의 속성에서 항목형식을 “Microsoft Macro Assembler”로 변경



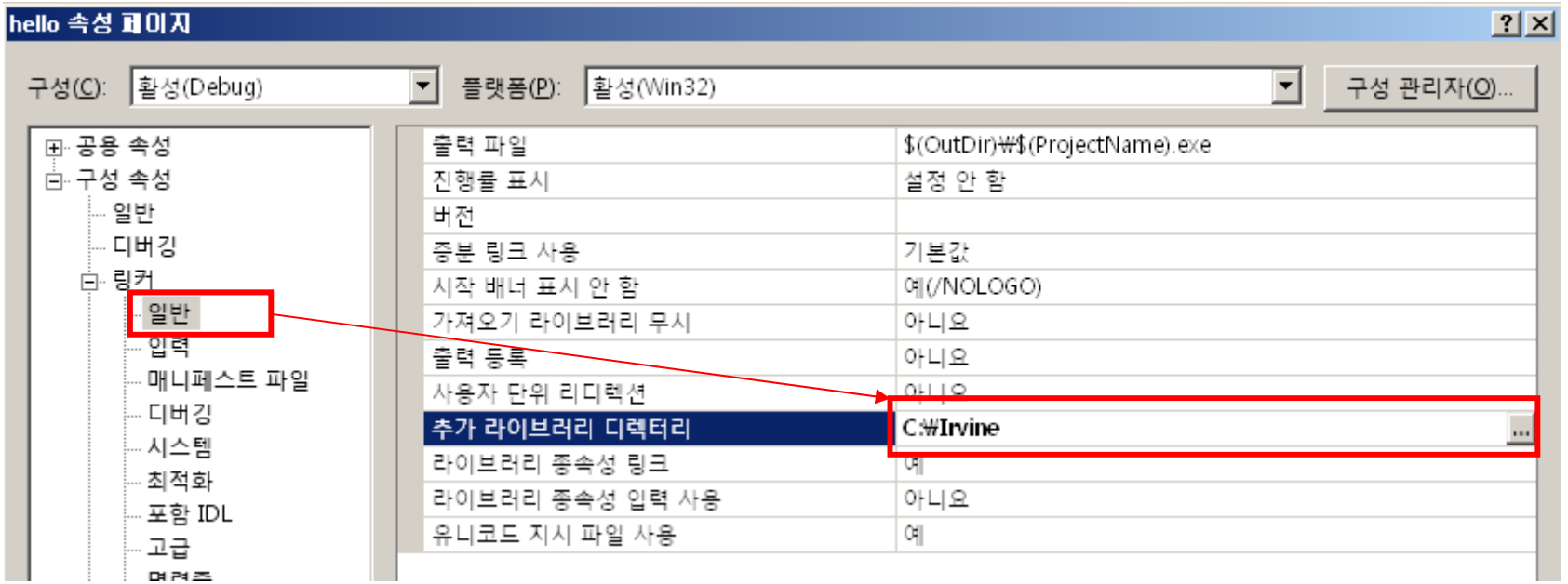
Property – “console” subsystem (Linker)

- [프로젝트 > 속성 > 구성속성 > 링커 >
  시스템 > 하위시스템 > 콘솔(/SUBSYSTEM:CONSOLE) 선택



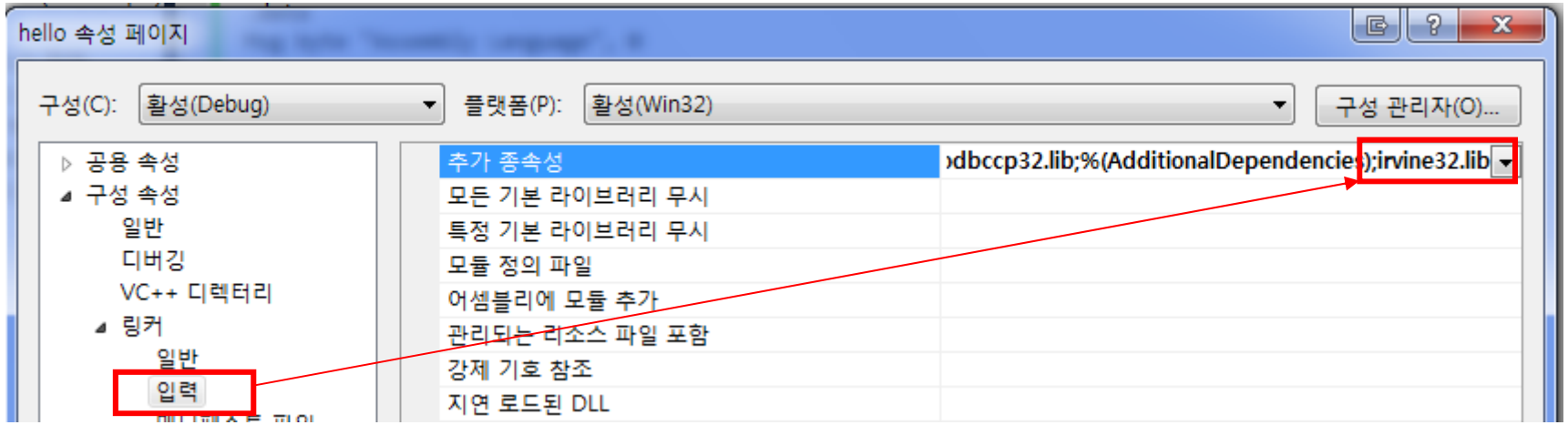
Property – library directories (Linker)

- [프로젝트 > 속성 > 구성속성 > 링커 >
  일반 > 추가 라이브러리 디렉터리 > **C:\Irvine** 입력



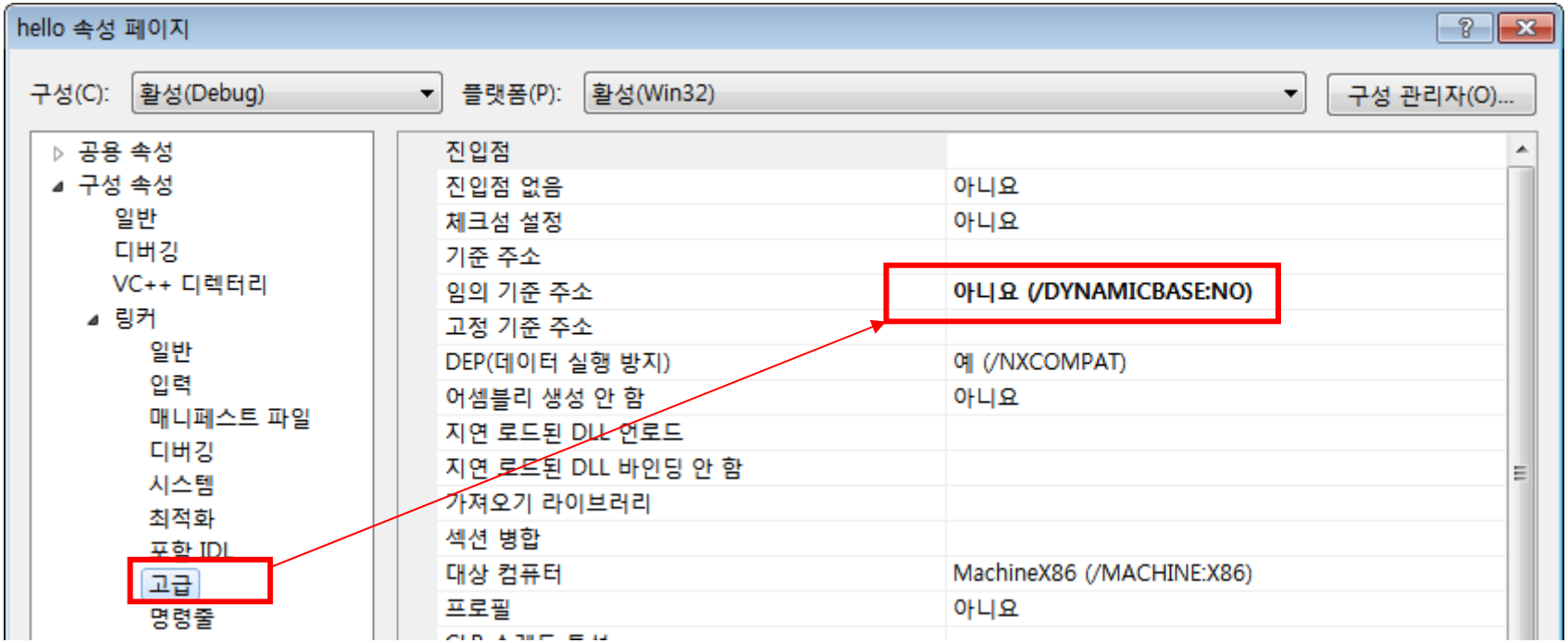
Property – library filename

- [프로젝트 > 속성 > 구성속성 > 링커 >
  입력> 추가 종속성 > **;irvine32.lib** 를 현재 입력 끝에 추가



Property – Randomized Base Address

- [프로젝트 > 속성 > 구성속성 > 링커 >
  고급 > 임의 기준 주소 > **아니오** 선택



Source code 작성

- assembly language source **프로그램 작성 및 저장**

hello.asm

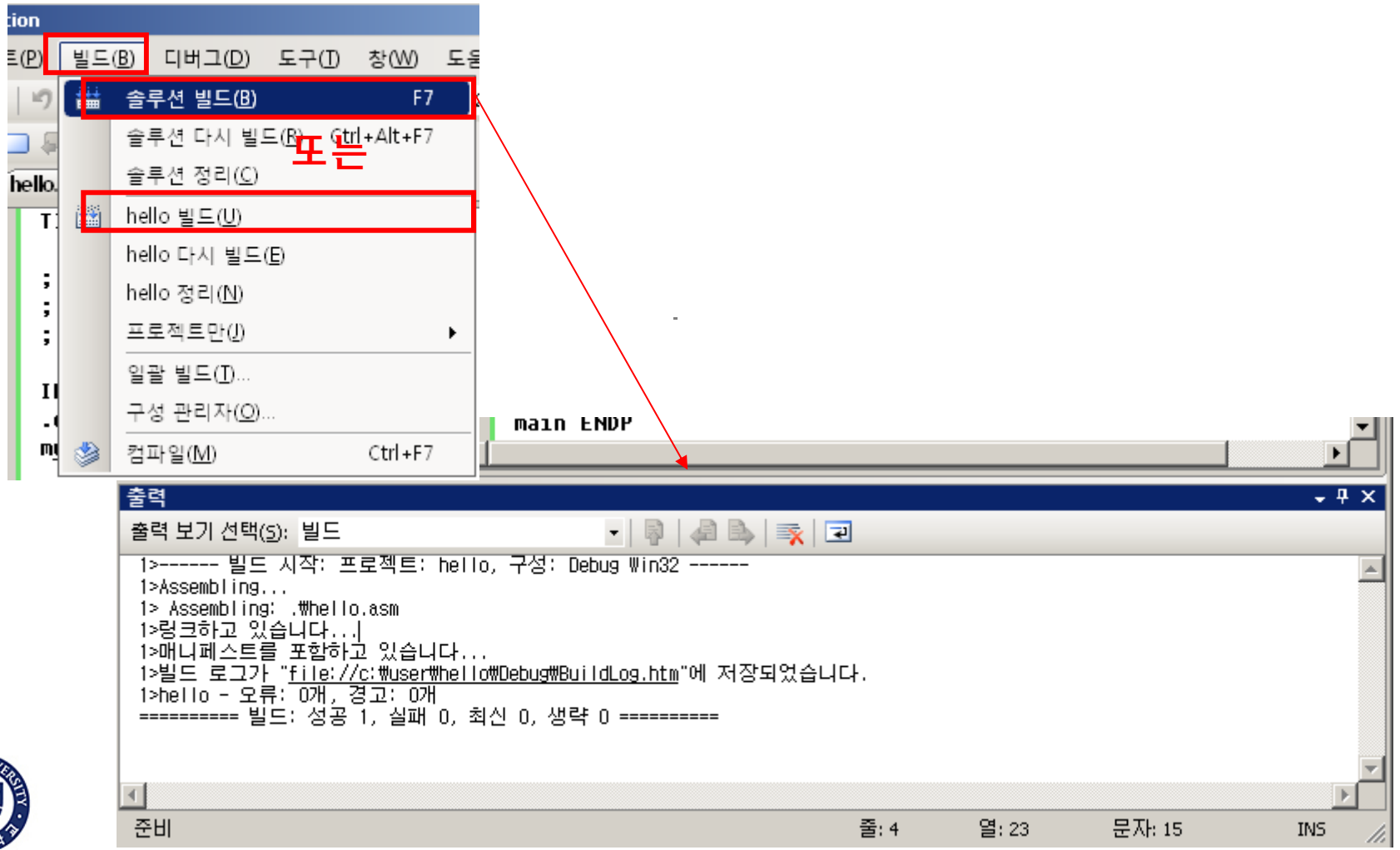
```
TITLE MASM Template                    (hello.asm)

; Description:
;
; Revision date:

INCLUDE Irvine32.inc
.data
myMessage BYTE "Hello!! Assembly Language",0dh,0ah,0

.code
main PROC
    call Clrscr

    mov  edx,OFFSET myMessage
    call WriteString

    exit
main ENDP

END main
```

화면 지우기

문자열 출력

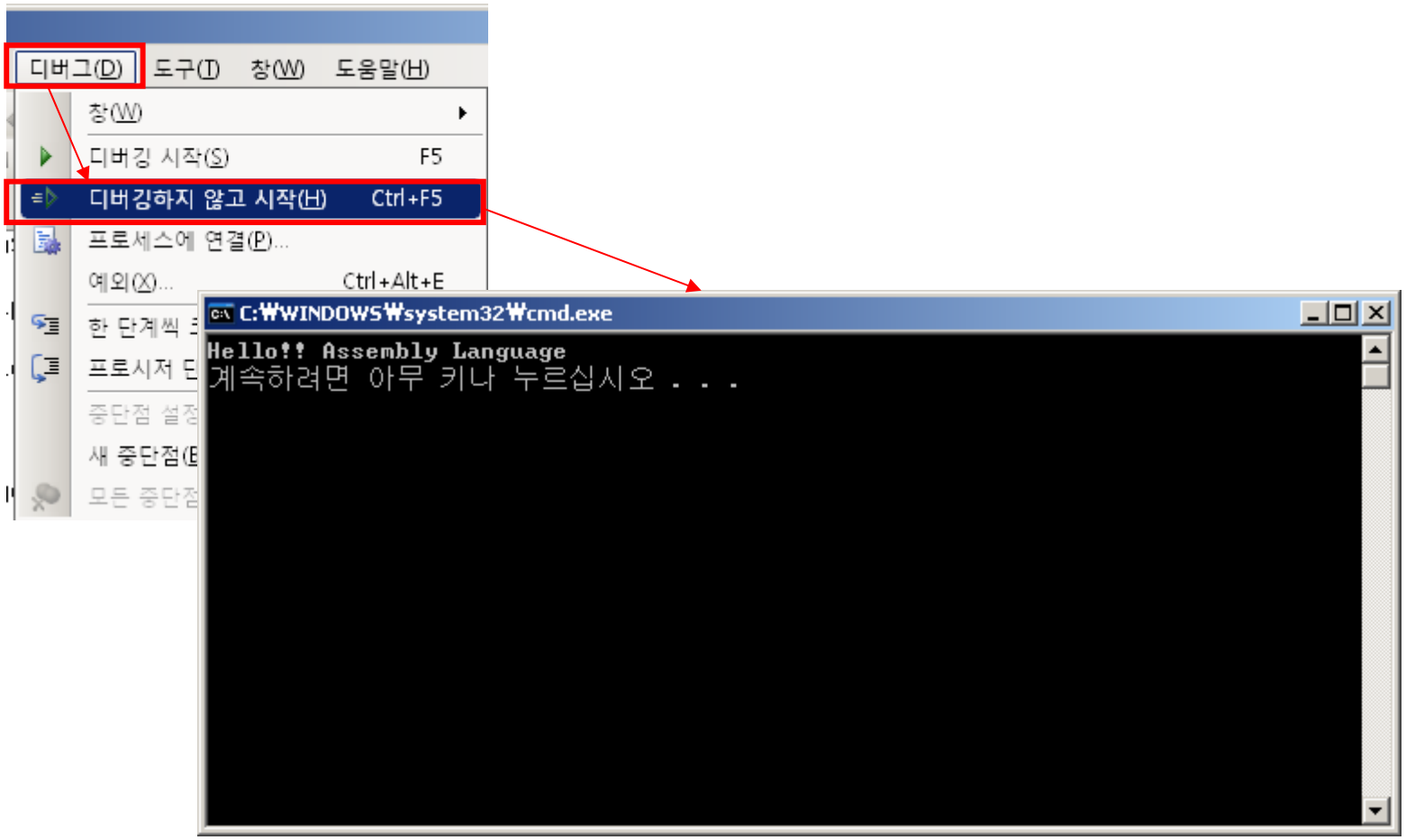
Build

- [빌드 > 솔루션 빌드] 또는 [빌드 > *Project* 빌드] → 실행파일 생성



Run

- [디버그 > 디버깅하지 않고 시작] (**Ctrl-F5**) 또는
  [디버그 > 디버깅 시작] (**F5**)



Visual Studio Debugger

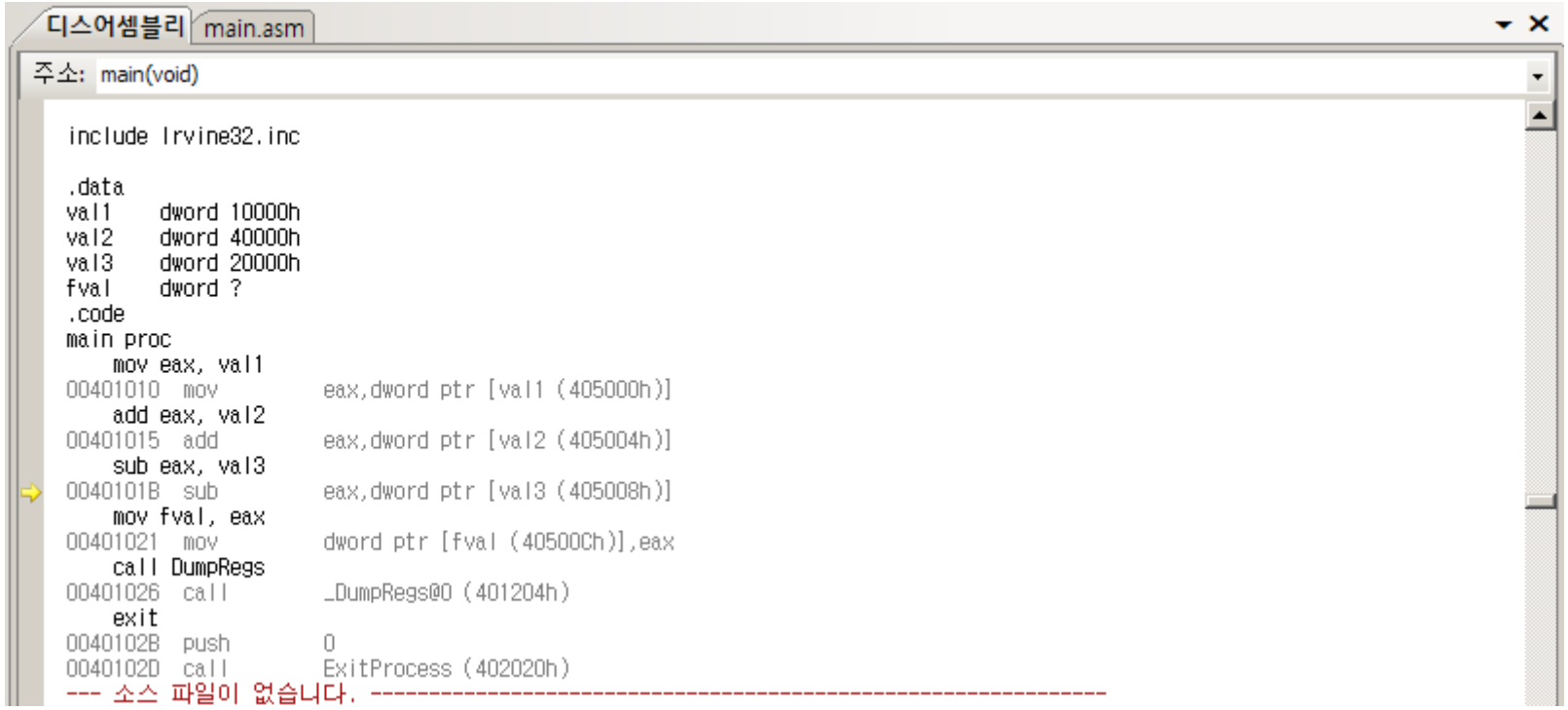
- 디버깅 방법
  - 프로시저단위 실행(step over): **F10** (디버그 >프로시저 단위 실행)
  - 한 단계 실행 (step into): **F11** (디버그 > 한 단계씩 코드 실행)
  - breakpoint 설정 후 디버그 실행
    1. breakpoint 설정: **F9** (디버그 > 중단점 설정/해제)
    2. 디버그 시작: **F5** (디버그 > 디버그시작/계속)

       breakpoint 지점에서 정지하며, F5를 다시 누를 경우
       다음 breakpoint 지점까지 계속 진행

- 디버그 창 (디버그 > 창 > ...)
  - 메모리
  - 레지스터
  - 조사식 (Watch)
  - 디스어셈블리 (Disassembly)
  - 중단점(Breakpoint)

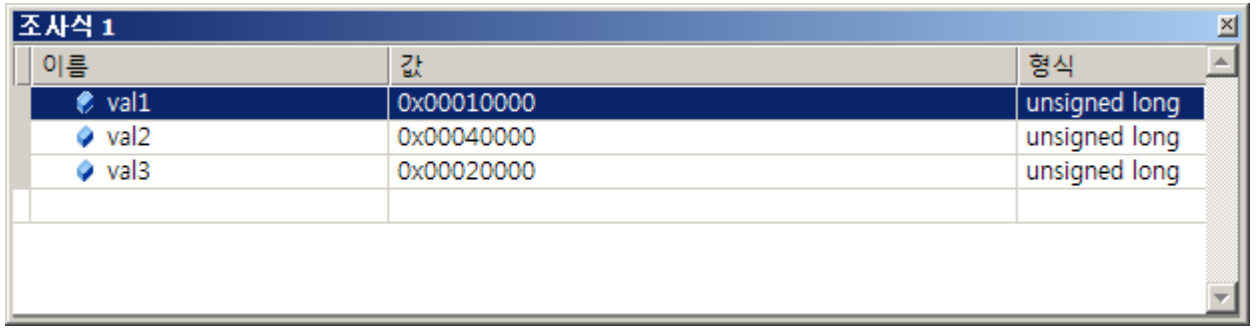
Disassembly Window

- [디버그 > 창 > 디스어셈블리]
  - popup menu에서 표시형식 변경 가능 (줄번호, 기계어코드 표시)



Watch Window

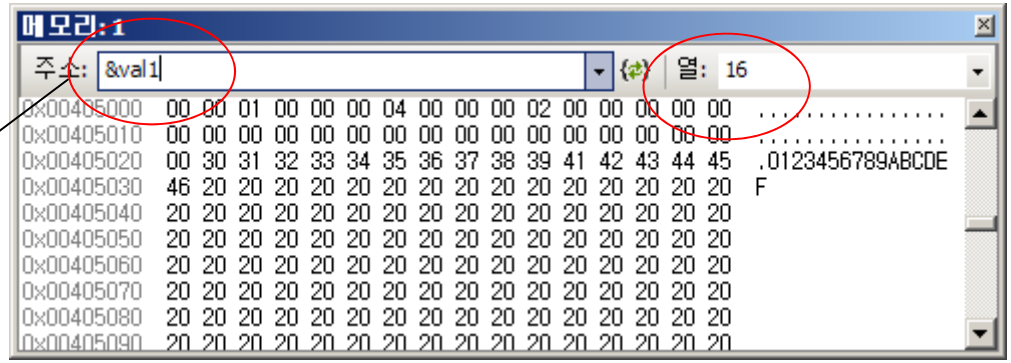
- [디버그 > 창 > 조사식]
  - 변수 값을 출력
  - 값의 표시 형식 변경 가능: 10진수, 16진수 (popup menu 사용)



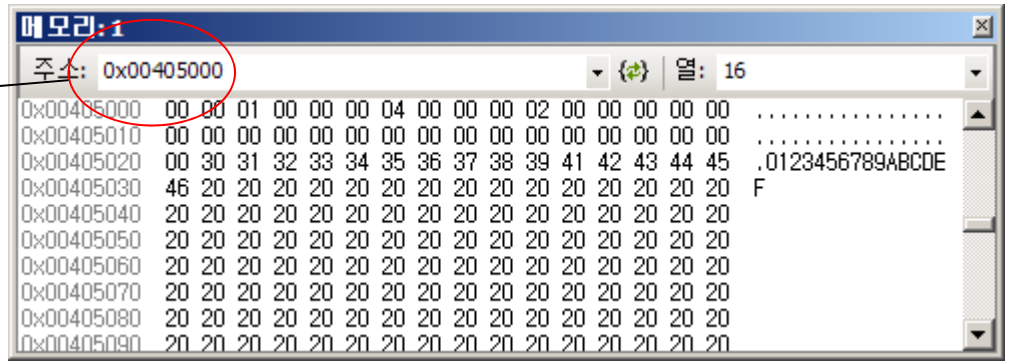
Memory Window

- [디버그 > 창 > 메모리]
  - 메모리 내용을 출력

변수주소
&val1

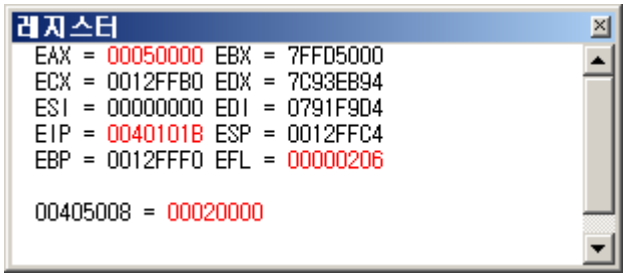


C언어
16진수 형식

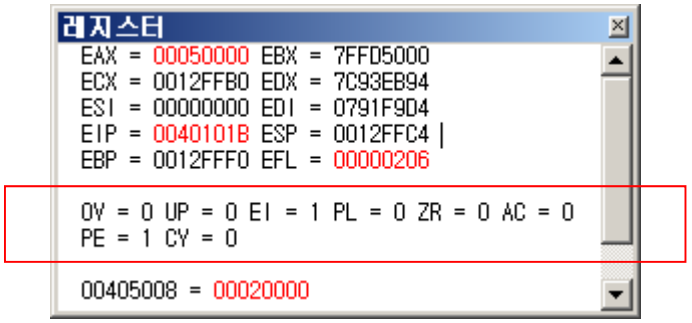


Register Window

- [디버그 > 창 > 레지스터]
  - 레지스터 내용 출력



  - 플래그 출력: popup menu에서 플래그를 선택



간편한 사용법

- Project 설정이 복잡하므로 생성한 Project 파일을 재활용하는 것이 편리함
  - 이전 프로젝트에서 사용했던 소스 파일을 제거
  - 새로운 소스 파일을 project에 추가