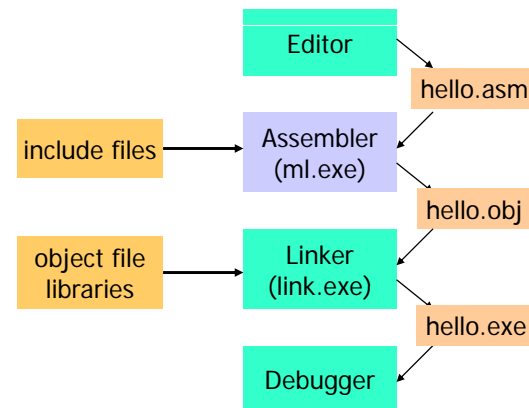


Visual Studio 2010을 사용하여 MASM 프로그램을 작성하는 방법

연세대학교
컴퓨터정보통신공학부
윤 상 균

필요한 소프트웨어



연세대학교

어셈블리어

Visual Studio 2010과 MASM

- Visual Studio 2008부터 Assembler가 포함됨
- Visual Studio 2010
 - Editor, Linker, Debugger 와 Assembler를 모두 포함
- MASM 10.0
 - Visual Studio 2010에 포함됨
 - Custom Build rule에 MASM기능을 포함시켜야 함
- Visual Studio 2010에서 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



연세대학교

어셈블리어

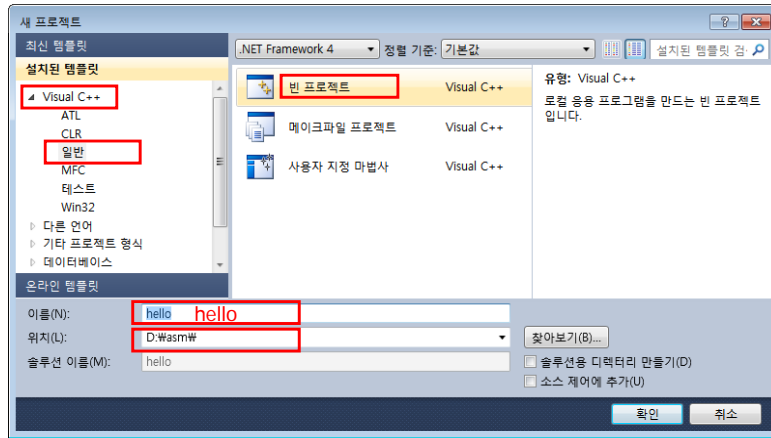


연세대학교

어셈블리어

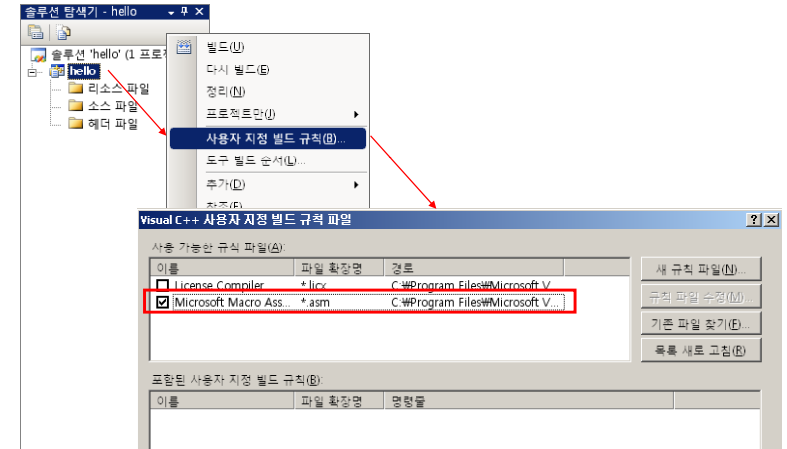
New project 생성

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



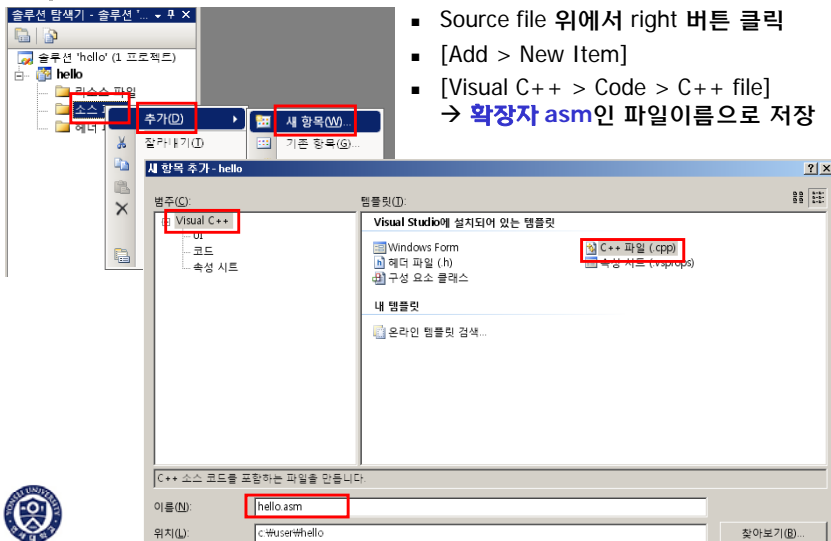
Custom Build Rules(사용자 지정 빌드)

- Solution Explorer에서 project 이름에서 right 버튼 클릭
- [Custom Build Rule] → Microsoft Macro Assembler 선택



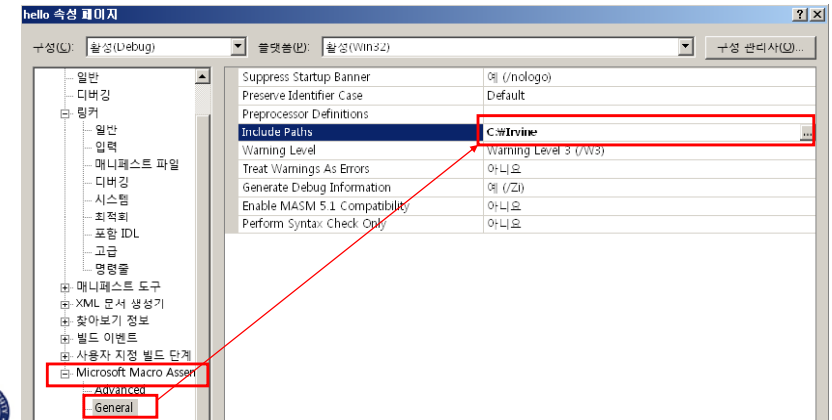
New source file 추가

- Source file 위에서 right 버튼 클릭
- [Add > New Item]
- [Visual C++ > Code > C++ file] → 확장자 asm인 파일이름으로 저장



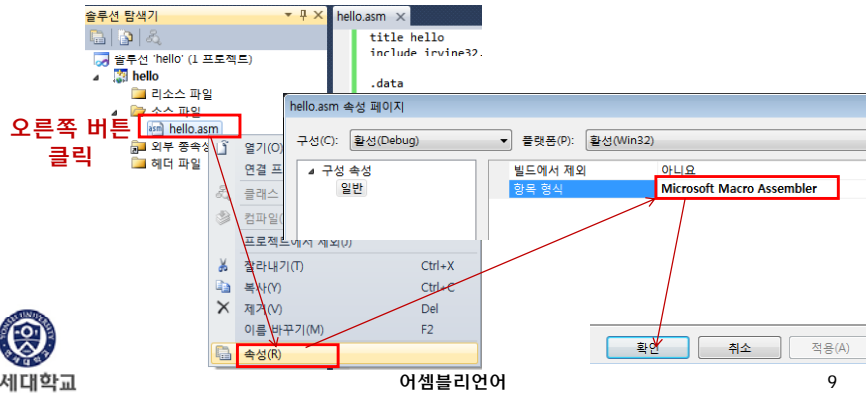
Property – MASM include path

- [Configuration Properties > Microsoft Macro Assembler > General > Include Paths]
- include file 경로 입력 : C:\ Irvine



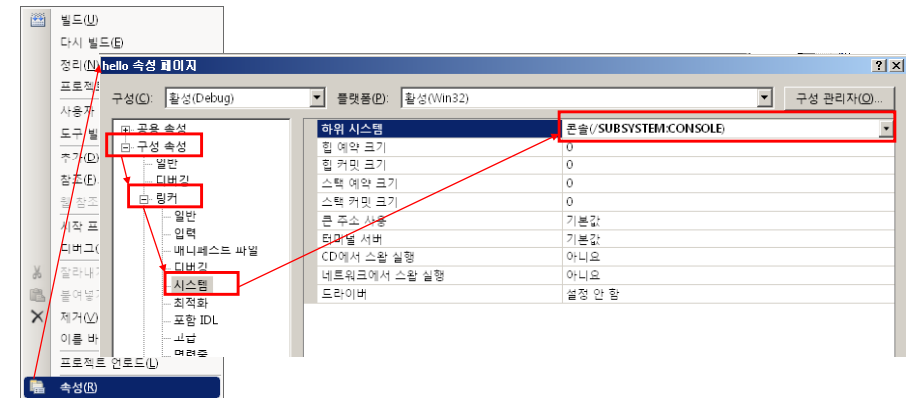
ASM 파일 속성 변경

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



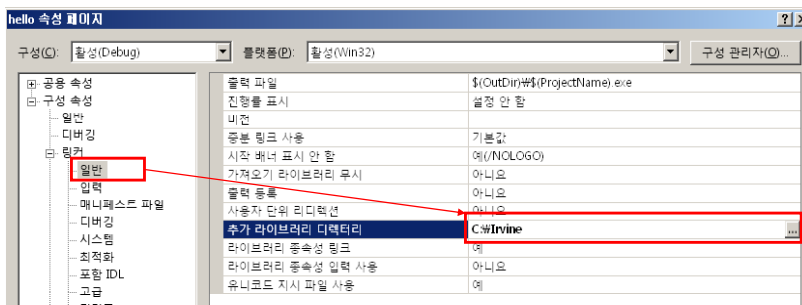
Property – "console" subsystem (Linker)

- [Configuration Properties > Linker > **System** > Subsystem]
- Select "**console**" as subsystem



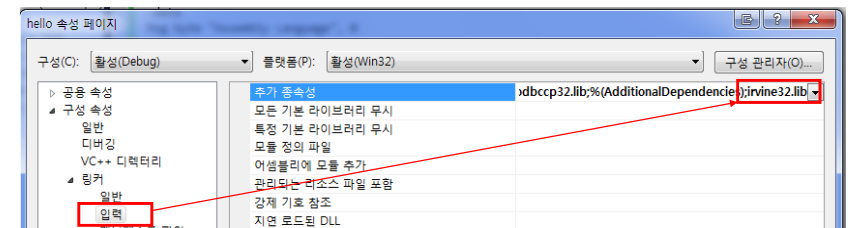
Property – library directories (Linker)

- [Configuration Properties > Linker > **General** > Additional Library Directories]
- library file 경로 입력 : **C:\ Irvine**



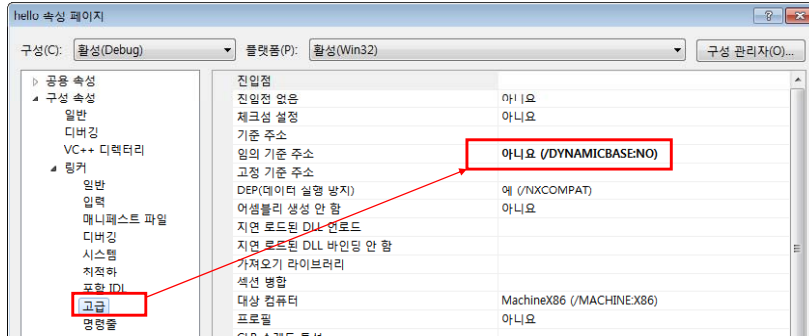
Property – library filename

- [Configuration Properties > Linker > Input > Additional Dependency]
- library file name 추가 입력 - ; Irvine32.lib



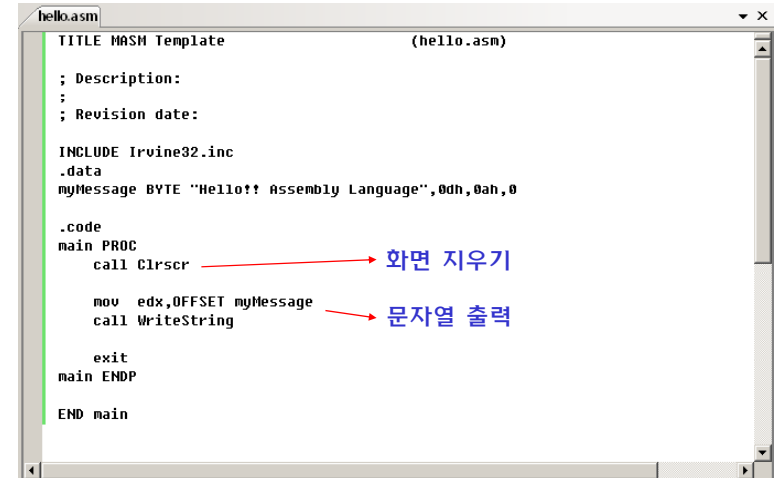
Property – Randomized Base Address

- [Configuration Properties > Linker > Advanced > Randomized Base Address]
- Select **No**



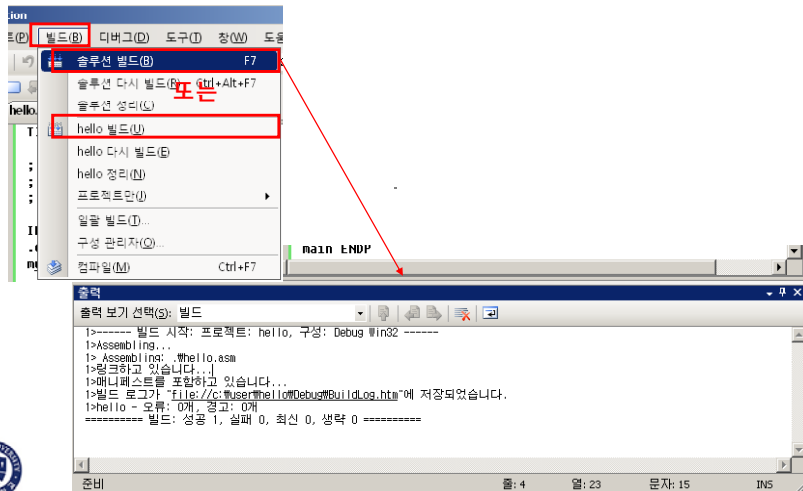
Source code 작성

- assembly language source 프로그램 작성 및 저장



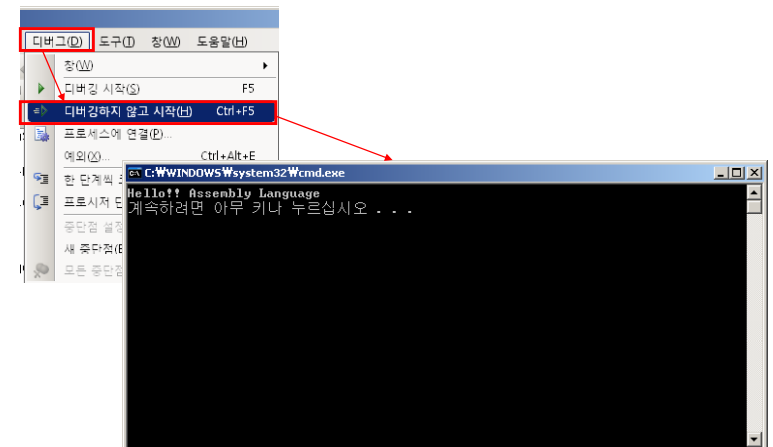
Build

- [Build > Build Solution] 또는 [Build > build hello] → 실행파일 생성



Run

- [Start without Debugging] (Ctrl-F5) 또는 [Start Debugging] (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
 - breakpoint 설정: F9 (디버그 > 중단점 설정/해제)
 - 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)

이름	값	형식
val1	0x00010000	unsigned long
val2	0x00040000	unsigned long
val3	0x00020000	unsigned long



Memory Window

[디버그 > 창 > 메모리]

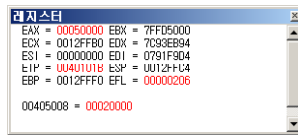
- 메모리 내용을 출력



Register Window

■ [디버그 > 창 > 레지스터]

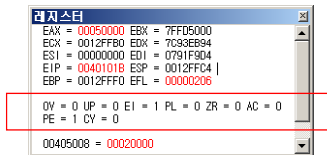
■ 레지스터 내용 출력



```
레지스터
EAX = 00050000 EBX = 7FFD5000
EDX = 0012FFB0 EDI = 7C93EB94
ESI = 00000000 EDI = 0791F9D4
EIP = 00401018 ESP = 0012FFC4
EBP = 0012FFFO EFL = 00000206

00405008 = 00020000
```

■ 플래그 출력: popup menu에서 플래그를 선택



```
레지스터
EAX = 00050000 EBX = 7FFD5000
ECX = 0012FFB0 EDI = 7C93EB94
ESI = 00000000 EDI = 0791F9D4
EIP = 00401018 ESP = 0012FFC4
EBP = 0012FFFO EFL = 00000206

OY = 0 UP = 0 EI = 1 PL = 0 ZR = 0 AC = 0
PE = 1 CY = 0

00405008 = 00020000
```