# 8장 고급프로시저(2)



## 8.4 INVOKE, ADDR, PROC, PROTO*

- INVOKE directive
- ADDR operator
- PROC directive – parameter list 포함
- PROTO directive



## INVOKE Directive

- INVOKE procedureName [, argumentList]
  - CALL instruction이 여러 개의 argument를 사용할 때에 push와 call의 대신에서 사용할 수 있는 directive

```
push 6
push 5
call AddTwo
. . .
```

→

```
INVOKE AddTwo, 5, 6
. . .
```

- Argument의 유형
  - 상수, 상수식 (ex) 100, OFFSET myList, TYPE array
  - register (ex) eax, bl
  - variable (ex) myList, array
  - address 식 (ex) [myList+2], [ebx+esi]
  - ADDR name (ex) ADDR myList



## ADDR Operator

- ADDR variable
  - variable에 대한 주소를 return함
  - INVOKE의 argument로 변수의 주소를 전달할 때에 사용함
  - memory model에 따라서 주소의 크기가 다름
    - Small model: 16-bit offset (near pointer)
    - Large model: 32-bit segment/offset (far pointer)
    - Flat model: 32-bit offset
- Example:

```
.data
myWord WORD ?
.code
INVOKE mySub, ADDR myWord
```

## PROC Directive

- label PROC [, paramlist]

  . . .

  label ENDP

  - label: procedure 이름
  - paramlist: procedure의 parameter list (선택사항)

- paramList
  - 형식: paramName: type, paramName: type, . . .
  - type은 표준 MASM type (BYTE, SBYTE, WORD 등) 또는 이 type에 대한 pointer type (PTR BYTE, PTR WORD 등)을 사용함



## Example: AddTwo Procedure

- procedure AddTwo
  - parameters: var1, var2 (stack parameter)
  - result: eax ← var1 + var2

```
AddTwo PROC, val1:DWORD, val2:DWORD
   mov eax,val1
   add eax,val2
   ret
AddTwo ENDP
```

PROC directive를 사용하여 parameter list를 선언하면 parameter를 [EBP+12] 와 같은 표기 대신에 val1, val2와 같은 표기를 사용할 수 있어서 편리함



## PROC Examples

- procedure FillArray
  - parameter: BYTE 배열의 주소, BYTE배열을 채울 값, 배열크기

```
FillArray PROC,
    pArray:PTR BYTE,
    fillVal:BYTE,
    arraySize:DWORD

    mov ecx,arraySize
    mov esi,pArray
    mov al,fillVal
L1: mov [esi],al
    inc esi
    loop L1
    ret
FillArray ENDP
```

parameter



## PROC Examples

```
Swap PROC,
   pValX:PTR DWORD,
   pValY:PTR DWORD
   . . .
Swap ENDP
```

```
ReadFile PROC,
   pBuffer:PTR BYTE
   LOCAL fileHandle:DWORD
   . . .
ReadFile ENDP
```

parameter

local variable

## PROTO Directive

- label PROTO paramList
  - procedure의 prototype 선언
  - procedure가 정의되기 전에 사용되는 경우에 필요함
    (C/C++의 prototype 선언과 같은 용도)
  - INVOKE directive를 사용하여 호출되는 procedure는 반드시 prototype을 가져야 함

```
MySub PROTO ...      ; procedure prototype

.code
   INVOKE MySub      ; procedure call
   . . .

MySub PROC ...       ; procedure implementation
   . . .
MySub ENDP
```



## Example: ArraySum

- Prototype, Invocation

```
ArraySum PROTO,ptrArray:PTR DWORD,szArray:DWORD
.data
array DWORD 10000h, 20000h, 30000h, 40000h, 50000h
Sum   DWORD ?
.code
main PROC
   INVOKE ArraySum, ADDR array, LENGTHOF array
   mov Sum, eax
   . . .
main ENDP
```



## Example: ArraySum(계속)

- Implementation

```
ArraySum PROC,ptrArray:PTR DWORD,szArray:DWORD
   mov esi,ptrArray
   mov ecx,szArray
   cmp ecx,0
   je L2
   mov eax, 0
L1:add eax,[esi]
   add esi, 4
   loop L1
L2:ret
```



## Passing by Value or by Reference

- Passing by Value
  - 32-bit 값을 stack에 push (16-bit mode에서는 16-bit값 push 가능)

```
.data
myData DWORD 10000h
.code
main PROC
   INVOKE Sub1,myData
```

MASM →

```
push myData
call Sub1
```

- Passing by Reference
  - address를 stack에 push

```
.data
myData WORD 1000h
.code
main PROC
   INVOKE Sub1,ADDR myData
```

→

```
push OFFSET myData
call Sub1
```

## Parameter Classifications

| parameter 종류 | passing by value | passing by reference |
|---|---|---|
| input | O | O<br>(변수 값을 수정하지 않음) |
| output | X | O<br>(변수 값을 사용하지 않음) |
| input-output | X | O<br>(변수 값을 사용하며 수정도 함) |



## Example: Exchanging Two Integers

- Swap procedure
  - 두 32비트 정수변수의 값을 교환

```
Swap PROC USES eax esi edi,
   pValX:PTR DWORD,        ; pointer to first integer
   pValY:PTR DWORD         ; pointer to second integer

   mov esi,pValX           ; get pointers
   mov edi,pValY
   mov eax,[esi]           ; get first integer
   xchg eax,[edi]          ; exchange with second
   mov [esi],eax           ; replace first integer
   ret
Swap ENDP
```

pValX, pValY는 input-output parameter임



## Trouble-Shooting Tips

- Save and restore registers
  - procedure에서 값이 수정될 때에 (예외: return값 저장용 register)
- Wrong operand size
  - operand size에 따라서 배열 원소의 주소 계산에 주의해야 함
    [DArray + 1] (x)  [DArray + 4] (o)
- Wrong type of pointer
  - assembler는 PTR BYTE와 PTR DWORD등을 서로 구분하지 못하므로 다른 type에 대한 pointer를 인수로 전달하지 않아야 함
- Passing wrong immediate values
  - reference parameter로 상수를 전달하지 않아야 함



## 8.5 다중모듈 프로그램 작성

- 다중 모듈 프로그램
  - source코드가 여러 개의 파일(module)로 나누어서 작성된 프로그램
  - 각 module은 분리된 OBJ file로 어셈블 또는 컴파일되며, 모든 OBJ file들은 LINK utility를 사용하여 하나의 EXE file로 link됨 (static linking)
- 구성방법
  1. EXTERN 디렉티브 사용한 방법 – 전통적 방법
  2. INVOKE, PROTO 디렉티브 사용한 방법 – 저수준 내용 숨김

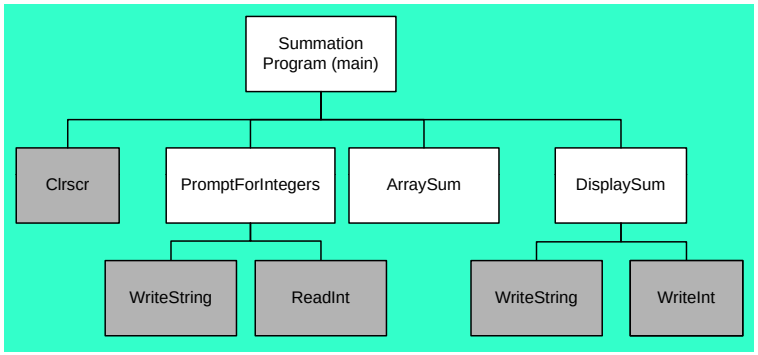
## Multimodule Program의 구성

- Multimodule program의 구성
  - main module 작성 – main procedure 포함
  - procedure 또는 related procedure의 집합을 별도의 source code module로 작성
  - PROTO directive를 사용한 procedure의 prototype 선언을 포함하는 include file 작성
  - include file을 INCLUDE directive를 사용하여 포함



## Example: ArraySum Program

- ArraySum program의 구조도






## INCLUDE File

- Include file "sum.inc"

```
INCLUDE Irvine32.inc

PromptForIntegers PROTO,
    ptrPrompt:PTR BYTE,        ; prompt string
    ptrArray:PTR DWORD,        ; points to the array
    arraySize:DWORD            ; size of the array

ArraySum PROTO,
    ptrArray:PTR DWORD,        ; points to the array
    count:DWORD                ; size of the array

DisplaySum PROTO,
    ptrPrompt:PTR BYTE,        ; prompt string
    theSum:DWORD               ; sum of the array
```



## Individual Modules

- Main
- PromptForIntegers
- ArraySum
- DisplaySum

(textbook의 source file 참조)

- Assemble and Link
  - Custom batch file 사용 (textbook참조) 또는 통합환경 사용
  - asm32는 single module program을 위한 batch file임