Chapter 10: Structures and Macros

Structure

- Structure
  - 관련된 변수들의 그룹으로 이루어진 자료구조 – template, pattern
  - field – structure를 구성하는 변수

  (cf) C언어의 struct
- 프로그램의 structure 접근
  - entire structure 또는
  - individual fields
- Structure는 procedure의 관련이 있는 인수들을 한꺼번에 전달할 때에 편리함
  - example: file directory information
- Structure 사용하기
  - structure 정의
  - structure 변수 선언
  - structure 변수 참조

Structure 정의

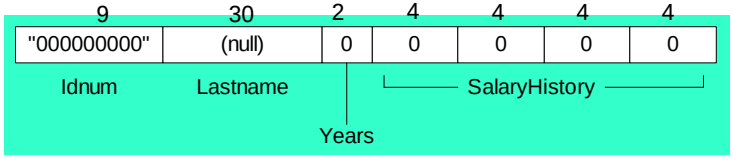
- Structure 정의

default 초기값

```
Employee STRUCT
   IdNum          BYTE "000000000"
   LastName       BYTE 30 DUP(0)
   Years          WORD 0
   SalaryHistory  DWORD 0,0,0,0
Employee ENDS
```

structure이름

field 선언은 변수선언과 같은 방식



Structure 변수 선언

- Structure 변수 선언
  - structure를 정의 한 다음에 structure 변수를 선언

```
COORD STRUCT
  X WORD ?
  Y WORD ?
COORD ENDS
```

structure 정의

structure이름을 type으로 사용

```
.data
point1 COORD <5,10>
point2 COORD <>
worker Employee <>
```

<>안에 초기값 지정

비어있으면 default초기값 지정

structure변수 이름

Structure 변수 선언 – 초기값, 배열

- Structure 초기값 지정

```
.data
emp Employee < ,,,2 DUP(20000)>
```

3개의 field를 건너뜀
(default 초기값 사용)

배열 field의 원소들의 일부 또는
전부를 초기화 하기 위해 DUP를 사용

- Array of Structure

```
NumPoints = 3
AllPoints  COORD NumPoints DUP(<0,0>)
RD_Dept    Employee 20 DUP(<>)
accounting Employee 10 DUP(<,,,4 DUP(20000) >)
```


Structure 변수 참조

- 변수의 field 참조

(형식 예) worker.SalaryHistory ; dword배열 field

```
mov dx,worker.Years
mov worker.SalaryHistory,20000        ; first salary
mov worker.SalaryHistory+4,30000      ; second salary

mov edx,OFFSET worker.LastName

mov esi,OFFSET worker
mov ax,[esi].Years                    ; invalid (ambiguous)
mov ax,(Employee PTR [esi]).Years
```


Structure 변수 참조

- structure 변수에 대한 크기 참조
  - TYPE Employee ; TYPE: 자료형의 메모리 크기 (57)
  - SIZEOF Employee ; SIZEOF=TYPE*LENGTHOF (57*1=57)
  - SIZEOF worker ; 변수 worker의 메모리 크기 (57)
  - TYPE Employee.SalaryHistory ; 4 (dword 자료형)
  - SIZEOF Employee.SalaryHistory ; 4*4=16 (dword형 배열, 4 원소)

```
Employee STRUCT
   IdNum          BYTE "000000000"     ;  9
   LastName       BYTE 30 DUP(0)       ; 30
   Years          WORD 0               ;  2
   SalaryHistory DWORD 0,0,0,0         ; 16
Employee ENDS                          ; (57)
```


예: 구조체 배열 참조하는 Loop

- 모든 점을 x, y 좌표를 함께 증가시키면서 저장
  - (1,1) (2,2) (3,3) ...

```
.data
NumPoints = 3
AllPoints COORD NumPoints DUP(<0,0>)

.code
   mov edi,0                  ; array index
   mov ecx,NumPoints          ; loop counter
   mov ax,1                   ; starting X, Y values
L1:
   mov (COORD PTR AllPoints[edi]).X,ax
   mov (COORD PTR AllPoints[edi]).Y,ax
   add edi,TYPE COORD
   inc ax
   Loop L1
```

Nested Structures

- Nested structure
  - structure 필드를 포함한 structure
- Nested structure 변수의 초기화
  - nested braces (or brackets)을 사용

```
COORD STRUCT
    X WORD ?
    Y WORD ?
COORD ENDS
```

```
Rectangle STRUCT
    UpperLeft COORD <>
    LowerRight COORD <>
Rectangle ENDS

.code
rect1 Rectangle { {10,10}, {50,20} }
rect2 Rectangle < <10,10>, <50,20> >
```

- Nested structure 변수의 접근
  - (예) rect1.UpperLeft.X

Union

- union
  - 메모리를 공유하는 field들로 구성된 자료 구조
  - 메모리 크기는 가장 긴 field의 크기와 같음
- union 정의와 변수 선언 및 접근

```
Integer UNION
    D DWORD 0
    W WORD 0
    B BYTE 0
Integer ENDS
.data
val1 Integer <12345678h>
val2 Integer <100h>
val3 Integer <>
```

```
mov val3.B, al
mov ax,val3.W
add val3.D, eax
```


Macros

- Macro
  - 이름이 부여된 assembly lanauge 문장들의 block
    → macro procedure라고도 부름
  - macro를 사용(호출)할 때마다 정의된 문장의 복사본이 삽입됨
  - 필요하면 parameter를 사용할 수 있음

  (cf) C언어의 #define
- Macro 정의 및 호출

```
mNewLine MACRO              ; define the macro
    call Crlf
ENDM
.data

.code
mNewLine                    ; invoke the macro
```

call Crlf 로 대치됨

Parameter가 있는 Macro

- mPutchar macro
  - 인수의 문자를 화면에 출력

Definition:

```
mPutchar MACRO char
    push eax
    mov al,char
    call WriteChar
    pop eax
ENDM
```

Invocation:

```
.code
mPutchar 'A'
```

Expansion:

```
1   push eax
1   mov al,'A'
1   call WriteChar
1   pop eax
```

LST 파일

mWriteStr Macro

```
mWriteStr MACRO buffer
    push edx
    mov  edx,OFFSET buffer
    call WriteString
    pop  edx
ENDM
.data
str1 BYTE "Welcome!",0
.code
mWriteStr str1
```

```
1    push edx
1    mov  edx,OFFSET str1
1    call WriteString
1    pop  edx
```

잘못된 argument사용은 expand될 때에 확인됨

Macro Examples

- mReadStr - reads string from standard input
- mDumpMem - dumps a range of memory
- mWrite – write string into standard output

mReadStr Macro

```
mReadStr MACRO varName
    push ecx
    push edx
    mov edx,OFFSET varName
    mov ecx,(SIZEOF varName) - 1
    call ReadString
    pop edx
    pop ecx
ENDM

.data
firstName BYTE 30 DUP(?)

.code
mReadStr firstName
```


mDumpMem Macro

```
mDumpMem MACRO address, itemCount, componentSize
    push ebx
    push ecx
    push esi
    mov  esi,address
    mov  ecx,itemCount
    mov  ebx,componentSize
    call DumpMem
    pop  esi
    pop  ecx
    pop  ebx
ENDM
…
.code
    mDumpMem OFFSET array, 8, 4
```

mWrite Macro – data 포함

macro comment

```
mWrite MACRO text
   LOCAL string
   .data                        ;; data segment
   string BYTE text,0           ;; define local string
   .code                        ;; code segment
   push edx
   mov  edx,OFFSET string
   call Writestring
   pop  edx
ENDM
```

- LOCAL directive를 사용하여 macro 내에서 사용하는 변수 정의
- 다른 곳에서 같은 macro를 호출할 때에 같은 이름의 변수와 구별할 수 있음

Nested Macros

- Nested macro: 다른 macro를 포함하여 정의된 macro

```
mWriteLn MACRO text
   mWrite text
   call Crlf
ENDM
```

정의

```
mWriteLn "My Sample Macro Program"
```

호출

nesting level

```
2   .data
2   ??0002 BYTE "My Sample Macro Program",0
2   .code
2   push edx
2   mov  edx,OFFSET ??0002
2   call Writestring
2   pop  edx
1   call Crlf
```