시스템프로그래밍 HW3
20101432 조윤지

- 자신의 학번과 이름을 화면에 출력한 뒤 프로그램을 시작
- Unsigned 5 Byte Integer들의 평균을 구하여 16진수로 화면에 출력
• 32개의 5 Byte 정수 사용: 난수를 생성하여 32개의 5 Byte 정수를 메모리상에 생성
• 32개 정수들의 평균 값(소수점 이하 절삭) 을 5 Byte 정수로 계산하여 화면에 출력
- MASM과 Irvine32 라이브러리를 이용하여 프로그램 작성 및 어셈블링 수행

코드

초기 설정 및 데이터 영역

```
INCLUDE C:₩Irvine32₩Irvine32.inc    ;사용할 라이브러리 추가

.386    ; this as a 32-bit program
.model flat, stdcall    ; program's memory model (flat), calling convention (named stdcall) for procedures
.stack 4096     ; 4096 bytes of storage for the runtime stack

.data
MyInfo BYTE "20101432 조윤지", 0    ; 학번과 이름을 저장한 null-terminated string
PrintInt BYTE "32개 unsigned 5byte integer 난수 생성", 0
hexa BYTE "0x",0
PrintSum BYTE "32개 unsigned 5byte integer 합: ", 0
PrintQuotient BYTE "평균: ", 0
val BYTE 160 DUP(?) ; 32개의 5byte 정수, 즉 160 byte 난수를 저장하는 배열
sum BYTE 6 DUP(0) ; sum
Quotient BYTE 6 DUP(0) ; sum을 32로 나눈 몫이면서 평균 값
```

코드 영역의 메인 프로시저 앞부분
- 학번과 이름을 출력한다.
- 난수생성 함수를 사용하여 eax에 난수값을 저장하고, 이를 1바이트씩 분리하여 val에 저장한다.
- 32 * 5 = 160 byte가 필요하고 eax는 4바이트이므로 L1을 40번 반복한다.

```
56  main PROC   ; 메인 프로시저 시작
57
58  call Clrscr ; 라이브러리에서 지원하는 프로시저. 콘솔창을 clear
59
60  mov edx, OFFSET MyInfo
61  call WriteString    ; "20101432 조윤지"를 출력
62
63  ; val 초기화 부분
64  call Randomize
65  mov ecx, 40
66  mov esi, 0
67  mov ebx, OFFSET val ; val 배열의 시작 주소를 ebx 레지스터에 복사합니다.
68  L1 : call Random32
69       mov BYTE PTR [ebx], al ; eax의 하위 1바이트 값을 val[0]에 저장합니다.
70       shr eax, 8             ; eax를 8비트만큼 오른쪽으로 이동합니다.
71       mov BYTE PTR [ebx+1], al ; eax의 하위 1바이트 값을 val[1]에 저장합니다.
72       shr eax, 8             ; eax를 8비트만큼 오른쪽으로 이동합니다.
73       mov BYTE PTR [ebx+2], al ; eax의 하위 1바이트 값을 val[2]에 저장합니다.
74       shr eax, 8             ; eax를 8비트만큼 오른쪽으로 이동합니다.
75       mov BYTE PTR [ebx+3], al ; eax의 하위 1바이트 값을 val[3]에 저장합니다.
76       inc esi
77       add ebx, 4
78
79       loop L1
```

구한 값들은 5바이트씩 끊어서 화면에 출력한다.
Display 프로시저를 이용하는데 자세한 내용은 sum을 출력하는 부분에서 설명한다.

32개 5byte unsigned int 합을 구한다.

- L2 루프를 32번 반복하므로 ecx에 32를 넣는다.
- 값들이 val에 연속적으로 저장되어 있으므로 edi에 val offset을 넣는다.
- L2 내부에서 Extended_Add에 다른 ecx값이 필요하므로 ecx를 push, pop 한다.

Extended_Add 프로시저 설명

- 5바이트 unsigned int 합을 구하는 프로시저, 그러므로 ecx에 5를 넣고 Extend_Add에 들어온다.
- val이 실제로는 byte형이지만, 그 안에 저장된 숫자가 5바이트 단위로 의미를 가지도록 하기 위해 adc 함수를 사용하여 더한다.
- **carry**와 edi값을 offset으로 하는 **val** 값, ebx값을 offset으로 하는 **sum** 값을 더해서 sum에 저장한다.
- val을 5바이트 단위로 처리하므로 sum은 마지막 캐리까지 고려하여 6바이트로 설정한다.

```
;32개 합
mov ecx, 32
mov edi, OFFSET val
L2:
push ecx
mov ebx, OFFSET sum
mov ecx, 5
call Extended_Add
pop ecx
loop L2
```

```
.code
Extended_Add PROC
    clc

L1: mov al, [edi]
    adc [ebx], al
    inc edi
    inc ebx
    loop L1

    adc byte ptr [ebx], 0
    ret
Extended_Add ENDP
```

구한 sum 값을 Display 프로시저를 이용하여 출력한다.

- ppt에 있는 Display 프로시저에서 sub의 순서를 바꿔서 사용했다.
- sum 값은 메모리에 리틀엔디안으로 저장되어 있기 때문에 출력할 때는 메모리 주소값이 큰 쪽에 저장되어 있는 값부터 출력한다.
- add esi, ecx 를 실행하면 esi가 sum에 저장된 마지막 값 바로 밑을 가리키므로 1씩 sub해서 sum에 들어있는 바이트들을 하나씩 출력한다.

```
111    call Crlf
112    mov edx, OFFSET PrintSum
113    call WriteString
114    mov esi, OFFSET sum
115    mov ecx, LENGTHOF sum
116    call Display
117    call Crlf
```

```
31    Display PROC
32        add esi, ecx
33        mov ebx, TYPE BYTE
34
35    L1: sub esi, TYPE BYTE
36        mov al, [esi]
37        call WriteHexB
38        loop L1
39
40        ret
41    Display ENDP
```

평균을 구하는 프로시저를 호출하고, 구한 평균값을 출력한 뒤 프로그램 종료

```
119     call Divide
120
121
122     mov esi, OFFSET Quotient
123     mov ecx, LENGTHOF Quotient
124     call Crlf
125     mov edx, OFFSET PrintQuotient
126     call WriteString
127     call Display
128     call Crlf
129
130     exit    ; 라이브러리에서 지원하는 프로시저. invoke ExitProcess, 0 대신 사용
131     main ENDP   ; 메인 프로시저 종료
132     END main    ; 프로그램 종료
```

평균을 구하는 방법

-Divide 프로시저를 호출한다.

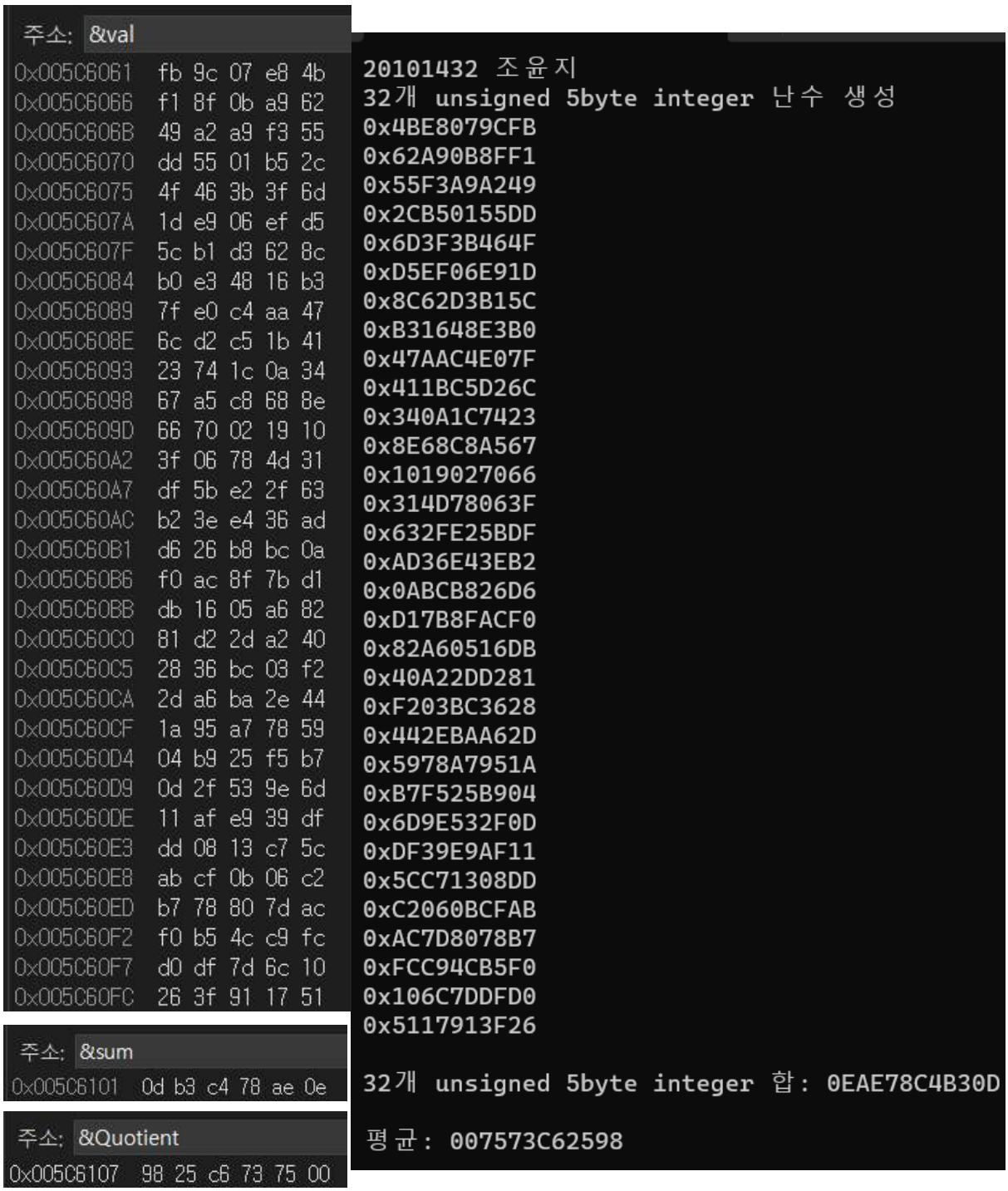
```
43      Divide PROC
44
45      mov eax, DWORD PTR sum  ; 하위 32비트
46      mov edx, DWORD PTR sum+4 ; 상위 32비트
47      shrd eax, edx, 5
48
49      shr edx, 5
50
51      mov WORD PTR Quotient+4, dx
52      mov DWORD PTR Quotient, eax
53      ret
54      Divide ENDP
```

sum을 32로 나눠야하므로
sum 값을 5비트 오른쪽으로 밀어야한다.
sum 값은 4바이트를 넘어가므로
edx와 eax에 나눠서 저장한다.

unsigned이므로 shrd와 shr을 이용한다.
shrd를 이용하여 나눗셈한 하위 32bit 값을 eax에 저장한다.
edx 값은 업데이트되지 않았으므로 shr를 이용하여 업데이트한다.

리틀엔디안을 고려하여 Quotient에 저장한다.

결과



- matlab에서 16진수를 사용하기 위해서 난수를 출력할 때 0x를 붙여서 출력하였다.

검증(matlab code)

```
x = [
0x4BE8079CFB
0x62A90B8FF1
0x55F3A9A249
0x2CB50155DD
0x6D3F3B464F
0xD5EF06E91D
0x8C62D3B15C
0xB31648E3B0
0x47AAC4E07F
0x411BC5D26C
0x340A1C7423
0x8E68C8A567
0x1019027066
0x314D78063F
0x632FE25BDF
0xAD36E43EB2
0x0ABCB826D6
0xD17B8FACF0
0x82A60516DB
0x40A22DD281
0xF203BC3628
0x442EBAA62D
0x5978A7951A
0xB7F525B904
0x6D9E532F0D
0xDF39E9AF11
0x5CC71308DD
0xC2060BCFAB
0xAC7D8078B7
0xFCC94CB5F0
0x106C7DDFD0
0x5117913F26
];

sumdec = sum(x); %배열 x에 있는 값을 더함. 10진수로 결과가 나옴
sumhex = dec2hex(sumdec) %10진수를 16진수로 바꿈

%10진수 합계 값을 32로 나눈 몫 (fix 함수)을 구하고 16진수로 바꿈
avg = dec2hex(fix(sumdec/32))
```

matlab 결과

```
sumhex =

    'EAE78C4B30D'


avg =

    '7573C62598'
```

어셈블리어로 작성한 파일의 실행 결과와 같다.

```
32개 unsigned 5byte integer 합: 0EAE78C4B30D
평균: 007573C62598
```