CSE3030 어셈블리 언어 프로그래밍 숙제 #5 List Sum

머리말

1. 본 과제에서 작성한 프로그램 일부는 기말 실기 시험에 필요할 수 있습니다. 따라서, 본 숙제를 반드시 완성하여 제출하는 것이 필요합니다.
2. C와 같은 고수준 언어로는 아주 쉬우나, 어셈블리 언어로는 상당히 까다로울 수 있으므로 가능한 일찍, 사전에 충분한 계획을 세워 차분하게 프로그램을 작성하도록 합시다(논리 오류가 있을 경우 디버거를 충분히 활용하세요).
3. 자신이 작성한 코드를 다른 학생에게 주지 마세요. 만일 실기 시험에서 과제로 작성한 프로그램의 일부가 두 명 이상의 학생이 사용한 것으로 체크되면 그 문제에 대해 해당 학생 모두는 0점을 받을 것입니다. 물론, 시험 중에 뭐든 공유한 것으로 체크되면 쌍방 0 점입니다.
4. Instruction 사용에 제한이 없지만 .IF, .WHILE 등 run time expression 사용을 금지합니다(이를 위반하면 역시 점수가 없습니다).

List Sum 문제

일련의 정수 값들을 입력 받아 이들의 합을 출력하는 프로그램을 작성하는 문제이다. 이 문제는 정수 값들을 하나씩 입력 받으면 Irvine 함수 ReadInt를 호출하여 쉽게 해결 할 수 있다.

그러나, 만일 정수 값들을 하나의 문자열로 입력 받으면 어찌해야 할까? 이 경우 우리는 문자열에서 각 정수 값들을 직접 추출하여야 한다(이는 C 언어의 함수 sscanf( )를 사용하여 문자열에서 정수 값들을 추출하는 것과 유사하다). 예를 들어 Irvine 함수 ReadString을 호출하여 문자열 " 20 -10 +34 "을 입력 받아 입력 버퍼에 저장했다고 하자. 그러면, 우리는 이 문자열로부터 정수 값 20, -10, 34를 추출하여 어떤 배열에 저장하고 이들을 모두 더한 값 44를 출력해야 한다. 이 과정은 C와 같은 고수준 언어로는 쉽게 해결할 수 있지만, 우리의 경우 이를 대행해 주는 함수가 제공되지 않아, 이 기능을 직접 프로그래밍해야 한다.

함수 작성

다음과 같은 기능을 수행하는 함수를 먼저 작성하면 나머지 프로그램은 쉽게 작성할 수 있다. 입력 받은 문자열을 저장한 입력 버퍼와 이로부터 추출한 정수들을 저장할 정수 배열이 예를 들어 다음과 같이 선언되어 있다고 가정하자:

```
BUF_SIZE EQU 256
inBuffer  BYTE BUF_SIZE DUP(?)      ; input string buffer(EOS is 0)
inBufferN DWORD ?                   ; string size(excluding EOS)
intArray  SDWORD BUF_SIZE/2 DUP(?)  ; integer array
intArrayN DWORD ?                   ; number of integers stored in intArray
```

작성해야 할 함수는 inBuffer의 문자열에서 정수 값들을 추출하여 intArray에 차례로 저장하고, 추출한 정수 값의 개수를 intArrayN에 저장하는 것이다. 앞에서 보인 예를 반복하여 기술하면, 문자열 " 20 -10 +34 "로부터 20, -10, 34를 차례로 추출하여 intArray에 저장하고, 3을 intArrayN에 저장하면 된다.

함수는 재사용이 가능하도록 작성해야 한다. 따라서, INVOKE를 사용하여 호출할 수 있도록 작성하거나 또는 인수를 레지스터를 통하여 전달하면 재사용이 쉬울 것이다. 이는 추후 숙제 또는 실기 시험에서 이 함수가 필요 할 수 있기 때문에 더욱 중요하다. 따라서, 예를 들어 inBuffer의 offset과 문자열의 크기를 각각 edx, ecx에 그리고 정수 배열의 offset을 edi에 저장한 후 함수를 호출하여 실행하면 추출한 정수 값들의 개수를 ecx에, 그리고 추출한 정수 값들을 edi가 가리키는 정수 배열에 차례로 저장 하도록 함수를 작성할 수 있겠다. 물론 다른 형태의 레지스터 할당도 얼마든지 가능하다.

메모리를 절약하기 위하여 함수 내에서 로컬 메모리의 사용을 가능한 자제해야 하는데, 실제로 이 함수는 레지스터만으로도 프로그래밍이 가능하다. 프로그램 편집을 시작하기 전에 모든 레지스터의 활용 계획을 사전에 잘 기획해 보자. 그리고, 함수 내에서 global 변수(.data로 선언

된 변수들)를 사용하는 것은 범용성이 떨어지기 때문에 최대한 자제하여야 한다.

마지막으로, Irvine 함수나 또는 다른 함수를 작성하여 사용하면 보다 쉽게 이 함수를 작성할 수 있다. 예를 들어 Irvine 라이브러리의 ParseInteger32의 기능을 살펴보자.

프로그램 입출력

프로그램은 명령 프롬프트에서 실행할 수 있어야 하며 그 실행 예를 아래에 보인다:

```
D:₩Work>s074419HW05 <ent>
Enter numbers(<ent> to exit) :
 10   -20 -30   20 <ent>
-20
Enter numbers(<ent> to exit) :
  23 -3  128  35    <ent>
183
Enter numbers(<ent> to exit) :
      <ent>
Enter numbers(<ent> to exit) :
<ent>
Bye!
D:₩Work>
```

위 예에서 갈색 점선 상자로 둘러싼 부분은 user가 입력한 문자열이고, <ent>는 Enter 키를 누른 것을 의미한다. Irvine 함수 ReadString를 통하여 문자열을 입력 받아 메모리에 저장한다.

입력 문자열은 빈칸, 숫자, +, -로 구성되며(기타 다른 문자는 포함되지 않는다고 가정) 그 크기는 최대 255로 한정한다. 정수 값과 값 사이는 한 개 이상의 빈칸으로 분리하여 입력되며, 각 정수 값 앞에 기호 + 또는 - 를 붙일 수 있다(부호가 없으면 양수). 문자열의 초반부와 끝 부분에도 빈칸이 한 개 이상 있을 수 있다.

아무 입력 없이 enter만 누르면 프로그램을 종료한다. 반면에 빈칸만 입력한 후 enter를 누르면 다시 문자열을 입력하라는 프롬프트를 낸다(위 예의 세 번째 입력). 이 외에 다른 예외적인 입력은 없다고 가정한다.

하나의 입력된 문자열에 포함된 정수 값은 최대 125 개로 한정한다.

출력은 위 예에서 보인 것처럼 양수인 경우에는 값 앞에 + 기호를 붙이지 않는다. 음수인 경우에는 당연히 값 앞에 - 기호를 붙인다.

프로그램 제출

1. 파일 이름: snnnnnnHW05.asm (nnnnnn은 자신 학번 뒤 6 자리. s는 소문자)..
2. 제출: 사이버 캠퍼스의 해당 과제 제출함에 제출(기한은 사이버 캠퍼스에 지정되어 있음).

주의 사항

1. 문자열에서 정수 값들을 반드시 적절한 배열에 저장하도록 프로그래밍하여야 합니다. 그렇지 않으면 추후 이 함수를 재사용 할 수 없게 됩니다. 또한, 만일 순전히 문자열 처리 방법으로 정수 값들을 합산하면(가능한가?) 이 역시 본 과제를 수행하는 의미가 없으며, 발견될 경우 점수를 얻지 못할 것입니다.
2. 무분별한 전역 변수(.data 영역) 또는 로컬 변수(stack 영역)를 사용하는 경우 감점할 수 있습니다. 이들을 사용하지 않거나 최소로 사용하도록 planning을 잘하여 레지스터를 최대한 사용합시다. 특히, 함수 내에서 전역 변수를 사용할 경우 크게 감점할 것입니다.
3. 어셈블리 오류인 경우, 복사로 판정한 경우에는 이유 불문 점수가 없습니다.
4. 제출 파일 이름, 출력 형식 등이 요구한 사항과 다르면 감점합니다.
5. 작성 후 다시 검토하여 코드+데이터 크기 합이 가능한 작도록 프로그램을 개선 합시다.
6. 완성 후 주석, 들여쓰기, 빈 줄 등을 적절히 추가하여 프로그램을 보기 좋게 만드세요.