시스템프로그래밍

Homework #2 (2)

Implement a Sorting Program using SIC / SIC/XE

< Bubble Sort >

이름 : 정유진

학번 : 201120895

제출일 : 2016년 11월 23일

Bubble Sort

1. Pseudocode

```
procedure bubbleSort( A : list of sortable items )
   n = length(A)
   repeat
     swapped = false
     for i = 1 to n-1 inclusive do
       /* if this pair is out of order */
       if A[i-1] > A[i] then
         /* swap them and remember something changed */
         swap( A[i-1], A[i] )
         swapped = true
       end if
     end for
   until not swapped
end procedure
```

2. 구현 코드

```
INSBUB    START    0
CHOMSG    LDA      #95
          WD       OUTDEV
          TIX      #55
          JLT      CHOMSG
          LDA      #10
          WD       OUTDEV
          WD       OUTDEV
          WD       OUTDEV
          CLEAR    X
          LDA      #CHOLEN
          SUB      #CHOTXT
          STA      CHOLEN
CHOPRT    LDA      CHOTXT, X
          WD       OUTDEV
          TIX      CHOLEN
          JLT      CHOPRT
          LDA      #CHOSOT
          STA      TMPIND
          LDT      #8          .CHOSOT 밑으로  몇칸씩  내려가야하는지
          CLEAR    A
CHOICE    TD       #0
          JEQ      CHOICE
          RD       #0
          STA      TMPNUM
          COMP     #50
          JGT      CHOICE
          COMP     #49
          JLT      CHOICE
          JEQ      @TMPIND
          COMP     #50
          JEQ      EXIT


CHOSOT    JSUB     BUBMSG
          CLEAR    X
          J        CHOMSG

.....END.....
          END      INSBUB
EXIT      J        EXITTW
EXITTW    J        EXIT
```

```
.....

.....Input Processing.....
.Input Msg 출력
INSAMP      CLEAR       X
            LDT         #5
INPMSG      LDA         #10
            WD          OUTDEV
            TIX         #2
            JLT         INPMSG
            CLEAR       X
            LDA         #IMSLEN
            SUB         #IMSTXT
            STA         IMSLEN
IMSPRT      LDA         IMSTXT, X
            WD          OUTDEV
            TIX         IMSLEN
            JLT         IMSPRT
            CLEAR       X
.#ZERO      ~ #FIGURE CLEAR
            LDA         #ZERO
            STA         IMSLEN
CLENUM      CLEAR       A
            STA         @IMSLEN
            LDA         #3
            ADD         IMSLEN
            STA         IMSLEN
            COMP        #STRLEN
            JLT         CLENUM
            CLEAR       A

.INput LOOP
..Input 을 받아옵니다.
INLOOP      TD          #0
            JEQ         INLOOP
            RD          #0
            COMP        #69         .'E' EOF 체크를 위해서 비교
            JEQ         EOFCHK
            COMP        #32         .공백을 나타내는 아스키값
            JEQ         ENDFIL
            COMP        #0
            JEQ         INLOOP
            COMP        #48         .'0' 보다 작으면 JUNK 처리
            JLT         JUNK
            COMP        #57         .'9' 보다 크면 JUNK 처리
            JGT         JUNK
            SUB         #48         .아스키 값을 10 진수로 변경
            CLEAR       S
            ADDR        A, S        .S 에 A 값(읽어온 값)을 옮겨놓음
            CLEAR       A           .A 초기화

.STRTMP
..A 에 TMPNUM 예전꺼 집어넣고 10 곱해줘서 자릿수 맞추어줌
...6 자리 넘어갈 경우 JUNK 처리하고 버림
....그리고 S 에 있던 이번에 읽어온 값을 1 의 자리에 넣어주는 형식
STRTMP      LDA         TMPNUM
            MUL         #10
            ADDR        S, A
            TIXR        T           .6 자리 이상이면 JUNK 처리
            JGT         INLOOP
            STA         TMPNUM
            CLEAR       A
            STA         JUKCHK
            J           INLOOP
```

```
.JUNK
..JUNK 는 Junk Check 변수에 값을 1 로 해주고 junk 임을 표시해 준다.
...그리고 해당하는 곳의 값을 다 제거해줘야되니까 띄어쓰기가 나올 때까지 Loop 를 돌린다.
JUNK        LDA         #1
            STA         JUKCHK
            J           INLOOP

.ENDFIL
..공백을 만나게 되면 ENDFIL 로 이동된다.
...Junk 일 경우 Clear Ready 로 이동하여 TempNum 을 비워준다.
....1 줄씩 옮겨가려면 3byte 를 이동해야해서 3 에 input 숫자를 곱해준다.(0 부터 시작; 0 = 1 개, 1 = 2 개)
.....STArt ADDress 에 첫번째 변수의 주소를 immediate addressing 을 통해 더해준다.
ENDFIL      LDA         JUKCHK      .Junk 인지 check
            COMP        #1
            JEQ         CLERDY      .Junk 면 다 폐기
            CLEAR       A
            LDA         #3          .시작주소 계산을 위해서
            MUL         INPNUM      .INPut NUMber 는 0 부터 시작하며 Input 의 개수를 나타낸다.
            STA         STAADD
.숫자 저장
            CLEAR       X

.STRSAM
..SToRe SAMple 은 Temp Number 에 임시로 저장해 두었던 값을 배열?에 차례대로 정리해줍니다.
...끝나고 INPut NUMber 를 1 증가 시켜준다.
....최대로 받을 수 있는 숫자의 개수는 15 개로 한정하고 15 개가 넘어가면 값을 받아오는 것을 종료한다.
.....그렇지 않으면 CLEar ReaDY 로 이동해서 Temp Number 를 초기화한다.
STRSAM      LDA         TMPNUM
            ..그냥 Index 로 해도 될 듯
            ...처음에는 Indirect 로 하려고 했으나 그렇게까지 할 필요가..?
            LDX         STAADD
            STA         STR1, X
            LDA         INPNUM
            ADD         #1          .Input Number 1 증가
            STA         INPNUM
            COMP        #15         .Sample 의 숫자가 15 개가 넘어가면 종료한다.
            JGT         ENDINP
            CLEAR       S
            J           CLERDY

.EOFCHK
..EOF CHecK 는 byte 단위로 READ 를 진행 중 'E'가 발견되었을 시에 이동되어 EOF 를 체크한다.
EOFCHK      RD          #0
            COMP        #79
            JEQ         EOFCHK
            COMP        #70
            CLEAR       A
            CLEAR       X
            JEQ         ENDINP

.CLERDY
..CLEar ReaDY 는 Temp Number 를 초기화하기 전 준비 단계이다.
CLERDY      CLEAR       A
            CLEAR       X
            CLEAR       S

.CLETMP
..CLEar TeMP 는 Temp Number 를 초기화 해서 재사용하게 만든다.
CLETMP      STA         TMPNUM, X
            TIX         #3
            JLT         CLETMP
            CLEAR       X
            JEQ         INLOOP
```

```
.END INPut
..END INPut 는 현재 숫자 받아오는 것이 끝난 상태이며 임시로 저장해놓은 이름이고 바뀔 수도 있다.
...RSUB 추가 예정 ver1.1
...RSUB 추가 ver1.3
...Print 로 이름 바꿀 수도 ver1.4
ENDINP      LDA         #100
            STA         FIGURE
            LDA         #0
            STA         TMPNUM
            STA         FIGURE      .FIGURE, TMPNUM 초기화
            LDA         #100        .자릿수 조절을 위해 FIGURE 에 100 추가 최대 3 자리.
            STA         FIGURE
            LDA         #STR1
            STA         NUMADD      .STR1 주소 불러와서 NUMADD 에 저장
            ..X 는 input 개수 chk 해야되니까 indirect 로
            LDA         #3
            MULR        X, A        .X 는 몇번째 Sample 인지 나타내고 3 을 곱해주어서 해당 주소로 이동하게 한다.
            ADD         NUMADD
            STA         NUMADD
            LDA         @NUMADD     .해당 되는 주소의 값을 불러온다.
            STA         TMPNUM      .TMPNUM 에 해당 값을 저장한 뒤 S reg 를 초기화한다.
            CLEAR       S
            LDA         #32
            WD          OUTDEV      .쓰기
            LDA         NUMADD
            COMP        PIVIND
            JEQ         PIVMAK
            J           CALFIG
PIVMAK      LDA         #124
            WD          OUTDEV
            LDA         #32
            WD          OUTDEV
            LDA         #0
            STA         INPLEN
.CALFIG
..CALculate FIGure 는 자릿수를 계산해서 저장한 뒤 출력해준다.
CALFIG      CLEAR       T
            LDA         TMPNUM      .TMPNUM 을 불러와서 현재 자릿수(Figure)로 나누어준다.
            DIV         FIGURE
            ADDR        A, T        .T reg 에 A reg 값을 할당한다.(현재 A reg 값은 해당 자릿수의 값이다)
            LDA         INPLEN
            COMP        #0
            JGT         PRTNUM
            CLEAR       A
            ADDR        T, A
            COMP        #0          .만약 0 이면 해당 자릿수에 값이 없는 것이니 JZERO 로 넘어간다.
            JEQ         JZERO
            STA         INPLEN
PRTNUM      CLEAR       S
            CLEAR       A
            ADDR        T, A
            MUL         FIGURE      .해당 자릿수의 값을 다시 곱해주어 200, 10, 40 이런 식으로 나타나게 한다.
            ADDR        A, S        .해당 값을 S reg 에 할당한 뒤 TMPNUM 값에서 빼주어 해당 자릿수를 없앤다.
            LDA         TMPNUM
            SUBR        S, A
            STA         TMPNUM
            CLEAR       A           .A reg 를 초기화 한뒤 T reg 에 할당해 놓은 자릿수의 계수 값을 받아온다.
            ADDR        T, A
            ADD         #48         .48 을 더해주어 ASCII 값으로 표현해 출력한다.
            WD          OUTDEV
```

```
JZERO
..Jump Zero? 는.. 자릿수가 없으면 점프해 오는 곳이고 본 목적은 자릿수를 한자리 감소시키는 역할을 한다.
..감소 시킨 뒤 자릿수가 남았으면 CALFIG 로 Jump 하고
..안남았으면 X 를 1 증가 시키고 Input Number 보다 작으면 ENDINP 로 Jump 하고 아니면 루틴을 빠져나온다.
JZERO   LDA     FIGURE
        DIV     #10
        STA     FIGURE
        COMP    #0
        JGT     CALFIG
        TIX     INPNUM
        CLEAR   A
        STA     INPLEN
        JLT     ENDINP
        RSUB

.....Bubble Sorting Ready.....
BUBMSG  CLEAR   A
        ADDR    L, A
        STA     RETADD
        LDA     #10
        WD      OUTDEV
        CLEAR   X
        JSUB    INSAMP
        LDL     RETADD
        LDA     #10
        WD      OUTDEV
        WD      OUTDEV
        CLEAR   X
        LDA     #BUBLEN
        SUB     #BUBTXT
        STA     BUBLEN
BUBPRT  LDA     BUBTXT,     X
        WD      OUTDEV
        TIX     BUBLEN
        JLT     BUBPRT

.....Bubble Sort Processing.....
....큰 LOOP
BUBRDY  CLEAR   A
        CLEAR   X
        LDA     INPNUM
        SUB     #1
        MUL     #3
        ADD     #STR1       .마지막으로 들어온 값 주소 계산
        STA     PIVIND      .마지막 주소값 PIVNUM 에 저장
.처음값 저장
BUBBLE  LDA     #STR2       .Lable 필요할까?
        STA     TMPNXT
BSTEP   LDA     TMPNXT
        SUB     #3
        STA     TMPIND
        LDA     @TMPIND
        COMP    @TMPNXT
        .Index(A reg)가 크면 교환
        ..3 증가 시켰을 때 TMPNXT 값이 PIVIND 보다 크거나 TMPIND 가 PIVIND 랑 같을 때 빠져나와서
        ..PIVIND 를 3 줄여주고 BUBBLE 로 이동해서 다시 시작
        ...PIVIND 를 3 줄였을 때 STR1 이랑 같으면 빠져나옴
        JLT     NOCHAN      .교환 안해도 됨
        LDA     @TMPIND
        STA     TMPNUM
        LDA     @TMPNXT
        STA     @TMPIND
        LDA     TMPNUM
        STA     @TMPNXT
        .교환 안해도 됨
```

```
NOCHAN  LDA     TMPNXT
        ADD     #3
        STA     TMPNXT
        COMP    PIVIND
        JEQ     BSTEP
        JLT     BSTEP

        CLEAR   A
        ADDR    L, A
        STA     RETADD
        LDA     #10
        WD      OUTDEV
        LDA     #91
        WD      OUTDEV
        CLEAR   X
        JSUB    ENDINP
        LDA     #32
        WD      OUTDEV
        LDA     #93
        WD      OUTDEV
        LDL     RETADD

        LDA     PIVIND
        SUB     #3
        STA     PIVIND
        COMP    #STR1
        JGT     BUBBLE
        LDA     #10
        WD      OUTDEV
        WD      OUTDEV
        J       RESMSG
```

.....Print Result.....

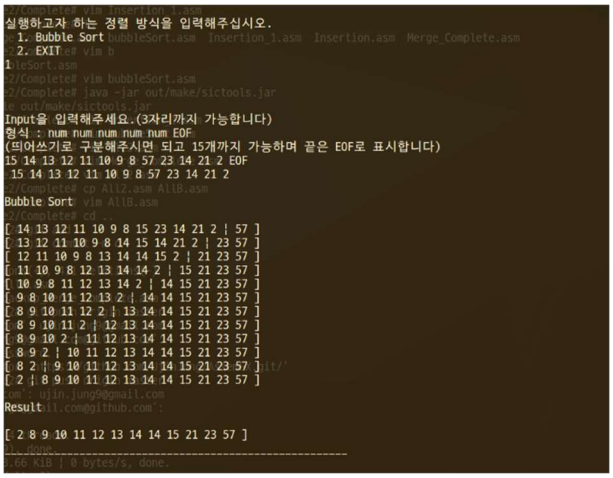
```
RESMSG  CLEAR   X
        CLEAR   A
        STA     PIVIND
        LDA     #RESLEN
        SUB     #RESTXT
        STA     RESLEN
RESPRT  LDA     RESTXT, X
        WD      OUTDEV
        TIX     RESLEN
        JLT     RESPRT
        CLEAR   A
        ADDR    L, A
        STA     RETADD
        LDA     #10
        WD      OUTDEV
        LDA     #91
        WD      OUTDEV
        CLEAR   X
        JSUB    ENDINP
        LDA     #32
        WD      OUTDEV
        LDA     #93
        WD      OUTDEV
        LDA     #10
        WD      OUTDEV
        LDL     RETADD
        RSUB
```

.교환 후에 TMPNXT 3 증가 후 PIVIND 와 비교
.PIVIND 보다 크지 않으면 BUBBLE 로 이동해서 시작
.TMPNXT > PIVIND 이면 PIVIND 3 감소
..PIVIND 3 감소 후 STR1 과 비교 후 같으면 끝

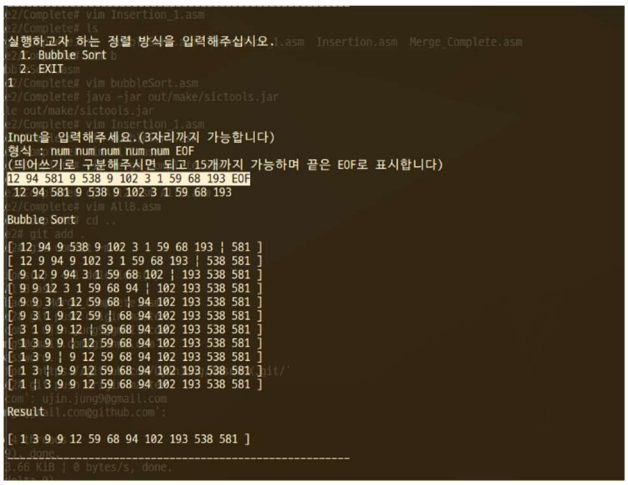
```
ZERO    WORD    0                       .ZERO 필요없지 않나?
STAADD  RESW    1                       .각 SAMPLE Input 시작 주소 Index 값
NUMADD  RESW    1                       .각 SAMPLE Input 숫자 시작 주소 Index 값
                                        .ver1.3 출력용 indirect 주소저장소로 사
ENDADD  RESW    1
INPLEN  RESW    1                       .각 SAMPLE 숫자 길이
INPNUM  WORD    0                       .각 SAMPLE Input 개수
TMPNUM  RESW    1                       .숫자를 임시로 저장해둔다.
PIVIND  RESW    1                       .PIVot INDex
PIVNUM  RESW    1
TMPIND  RESW    1
TMPNXT  RESW    1
JUKCHK  WORD    0                       .Junk 인지 chk 하기 위한 변수 (0 = not junk, 1 = junk)
STR1    RESW    1                       .배열 시작
STR2    RESW    1
        RESW    1
        RESW    1
        RESW    1
        RESW    1
        RESW    1
        RESW    1
        RESW    1
        RESW    1
        RESW    1
        RESW    1
        RESW    1
        RESW    1
        RESW    1
STRLEN  RESW    1
FIGURE  WORD    100
INDEV   BYTE    X'00'
OUTDEV  BYTE    X'01'
RETADD  RESW    1               .Main 으로 Return Address
PREADD  RESW    1               .Print 하고 Sort 로 Return Address

CHOTXT  BYTE    C'  실행하고자 하는 정렬 방식을 입력해주십시오.'
        BYTE    10
        BYTE    C'  1. Bubble Sort'
        BYTE    10
        BYTE    C'  2. EXIT'
        BYTE    10
CHOLEN  RESW    1
IMSTXT  BYTE    C'  Input 을 입력해주세요.(3 자리까지 가능합니다)'          .Input MaSsage Text
        BYTE    10
        BYTE    C'형식 : num num num num num EOF'
        BYTE    10
        BYTE    C'(띄어쓰기로 구분해주시면 되고 15 개까지 가능하며 끝은 EOF 로 표시합니다)'
        BYTE    10
IMSLEN  RESW    1
RESTXT  BYTE    C'  Result'
        BYTE    10
RESLEN  RESW    1
BUBTXT  BYTE    C'  Bubble Sort'
        BYTE    10
BUBLEN  RESW    1
```

3. 실행

1. Input : 15 14 13 12 11 10 9 8 57 23 14 21 2 EOF



2. Input : 12 94 581 9 538 9 102 3 1 59 68 193 EOF



주요 기능

```
.....Bubble Sorting Ready.....
BUBMSG    CLEAR    A
          ADDR     L, A
          STA      RETADD
          LDA      #10
          WD       OUTDEV
          CLEAR    X
          JSUB     INSAMP
          LDL      RETADD
          LDA      #10
          WD       OUTDEV
          WD       OUTDEV
          CLEAR    X
          LDA      #BUBLEN
          SUB      #BUBTXT
          STA      BUBLEN
BUBPRT    LDA      BUBTXT,     X
          WD       OUTDEV
          TIX      BUBLEN
          JLT      BUBPRT
```

Bubble Sorting 을 진행하기 이전에 Input 을 받기 위해서

Return Address 를 잠시 RETADD 에 저장해두고 Input 을 입력받은 뒤에 Return Address 를 L register 에 다시 Load 한다.

Bubble Text 를 Print 한다.

```
.....Bubble Sort Processing.....
....큰 LOOP
BUBRDY    CLEAR    A
          CLEAR    X
          LDA      INPNUM
          SUB      #1
          MUL      #3
          ADD      #STR1
          STA      PIVIND

BUBBLE    LDA      #STR2
          STA      TMPNXT
BSTEP     LDA      TMPNXT
          SUB      #3
          STA      TMPIND
          LDA      @TMPIND
          COMP     @TMPNXT
          JLT      NOCHAN
          LDA      @TMPIND
          STA      TMPNUM
          LDA      @TMPNXT
          STA      @TMPIND
          LDA      TMPNUM
          STA      @TMPNXT
          .교환 안해도 됨
          .
NOCHAN    LDA      TMPNXT
          ADD      #3
          STA      TMPNXT
          COMP     PIVIND
          JEQ      BSTEP
          JLT      BSTEP
```

크게 loop를 돌면서 가장 큰 값을 먼저 뒤에서부터 채운다.

현재 index된 값과 다음 값을 비교하여 뒤의 값이 작으면 교환한다.