시스템프로그래밍

Homework #2 (1)

Implement a Sorting Program using SIC / SIC/XE

< Insertion Sort >

이름 : 정유진

학번 : 201120895

제출일 : 2016년 11월 23일

Insertion Sort

1. Pseudocode

```
for i ← 1 to length(A)-1
    j ← i
    while j > 0 and A[j-1] > A[j]
        swap A[j] and A[j-1]
        j ← j - 1
    end while
end for
```

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          .CHOSOT밑으로 몇칸씩 내려가야하는지
            CLEAR       A
CHOICE      TD          #0
            JEQ         CHOICE
            RD          #0
            STA         TMPNUM
            COMP        #50         .5까지/?
            JGT         CHOICE
            COMP        #49
            JLT         CHOICE
            JEQ         @TMPIND
            COMP        #50
            JEQ         EXIT

            ....

CHOSOT      JSUB        INSMSG
            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
..Input을 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 JUNK처리
            JLT     JUNK
            COMP    #57         .'9' 보다 크면 JUNK처리
            JGT     JUNK
            SUB     #48         .아스키 값을 10진수로 변경
            CLEAR   S
            ADDR    A, S        .S에 A값(읽어온 값)을 옮겨놓음
            CLEAR   A           .A 초기화

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

.JUNK
..JUNK는 Junk Check 변수에 값을 1로 해주고 junk임을 표시해 준다.
...그리고 해당하는 곳의 값을 다 제거해줘야되니까 띄어쓰기가 나올 때까지 Loop를 돌린다.
JUNK        LDA     #1
            STA     JUKCHK
```

```
                J               INLOOP

.ENDFIL
..공백을 만나게 되면 ENDFIL로 이동된다.
...Junk일 경우 Clear Ready로 이동하여 TempNum을 비워준다.
....1줄씩 옮겨가려면 3byte를 이동해야해서 3에 input 숫자를 곱해준다.(0부터 시작; 0 = 1개, 1 = 2개)
.....STArt ADDress에 첫번째 변수의 주소를 immediate addressing 을 통해 더해준다.
ENDFIL          LDA             JUKCHK          .Junk인지 check
                COMP            #1
                JEQ             CLERDY          .Junk면 다 폐기
                CLEAR           A
                LDA             #3              .시작주소 계산을 위해서
                MUL             INPNUM          .INPut NUMber 는 0부터 시작하며 Input의 개수를 나타낸다.
                STA             STAADD
.숫자 저장
                CLEAR           X

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

.EOFCHK
..EOF CHecK 는 byte단위로 READ를 진행 중 'E'가 발견되었을 시에 이동되어 EOF를 체크한다.
EOFCHK          RD              #0
                COMP            #79
                JEQ             EOFCHK
                COMP            #70
                CLEAR           A
                CLEAR           X
                JEQ             ENDINP

.CLERDY
..CLEar ReaDY는 Temp Number를 초기화하기 전 준비 단계이다.
CLERDY          CLEAR           A
                CLEAR           X
                CLEAR           S

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

.END INPut
..END INPut는 현재 숫자 받아오는 것이 끝난 상태이며 임시로 저장해놓은 이름이고 바뀔 수도 있다.
...RSUB 추가 예정 ver1.1
...RSUB 추가 ver1.3
...Print로 이름 바꿀 수도 ver1.4
ENDINP          LDA             #100
```

```
            STA     FIGURE
            LDA     #0
            STA     TMPNUM
            STA     FIGURE      .FIGURE, TMPNUM 초기화
            LDA     #100        .자릿수 조절을 위해 FIGURE에 100 추가 최대 3자리.
            STA     FIGURE
            LDA     #STR1
            STA     NUMADD      .STR1 주소 불러와서 NUMADD에 저장
            ..X는 input 개수 chk해야되니까 indirect로
            LDA     #3
            MULR    X, A        .X는 몇번째 Sample인지 나타내고 3을 곱해주어서 해당 주소로 이동하게 한다.
            ADD     NUMADD
            STA     NUMADD
            LDA     @NUMADD     .해당 되는 주소의 값을 불러온다.
            STA     TMPNUM      .TMPNUM에 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      .TMPNUM을 불러와서 현재 자릿수(Figure)로 나누어준다.
            DIV     FIGURE
            ADDR    A, T        .T reg에 A reg 값을 할당한다.(현재 A reg값은 해당 자릿수의 값이다)
            LDA     INPLEN
            COMP    #0
            JGT     PRTNUM
            CLEAR   A
            ADDR    T, A
            COMP    #0          .만약 0이면 해당 자릿수에 값이 없는 것이니 JZERO로 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 reg에 할당한 뒤 TMPNUM값에서 빼주어 해당 자릿수를 없앤다.
            LDA     TMPNUM
            SUBR    S, A
            STA     TMPNUM
            CLEAR   A           .A reg를 초기화 한뒤 T reg에 할당해 놓은 자릿수의 계수 값을 받아온다.
            ADDR    T, A
            ADD     #48         .48을 더해주어 ASCII값으로 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 CALFIG로 Jump하고
..안남았으면 X를 1 증가 시키고 Input Number보다 작으면 ENDINP로 Jump하고 아니면 루틴을 빠져나온다.
JZERO       LDA     FIGURE
            DIV     #10
            STA     FIGURE
            COMP    #0
            JGT     CALFIG
            TIX     INPNUM
            CLEAR   A
```

```
                STA             INPLEN
                JLT             ENDINP
                RSUB

.....Insertion Sorting Ready.....

INSMSG          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             #INSLEN
                SUB             #INSTXT
                STA             INSLEN
INSPRT          LDA             INSTXT,         X
                WD              OUTDEV
                TIX             INSLEN
                JLT             INSPRT

.....Insertion Sort Processing.....
...ver1.4꺼 일단 삭제 좀
...다시 생각한 거로 해보고 복구를 하든
...ver1.4
....Ready에서는 A X TMPNUM 클리어.
....INSIND에 STR2주소값 저장. 3뺀거 TMPIND에 저장
....큰 LOOP
INSRDY          CLEAR           A
                CLEAR           X
                STA             TMPNUM
                LDA             #STR2
                STA             PIVIND
....큰 LOOP 시작과 동시에 작은 LOOP 준비?
....큰 LOOP는 밑에서 비교해서 값 추가하는 방식으로 하고
....작은 LOOP는 건너뛰는 방식으로
....작은 LOOP를 밑에서 빠져나와서 PIVIND가 증가되고 TMPIND의 초기 값을 변경해주는 작업
.INSERT         CLEAR           A
INSERT          CLEAR           S
                LDA             @PIVIND
                STA             PIVNUM
                LDA             PIVIND
                STA             TMPNXT
..작은 LOOP시작
...@TMPIND 값이랑 @(TMPIND+3)값이랑 비교
...비교 후 끝이면 S reg에 1 저장 ( s = 1이면 작은 loop빠져 나옴)
...크면 @(TMPIND+3) = @TMPIND
...., 옮기기 후 S reg chk 0 이면 TMPIND = TMPIND-3 후 작은 loop 돌림
...., 옮기기 후 1이면 작다로 이동
...작으면 @(TMPIND+3) = PIVNUM 후 PIVIND = PIVIND+3 후 TMPIND클리어 후 큰 LOOP(INSERT)로 이동
...|||      |      |||              |||||
.....TMPIND TMPNXT      PIVIND(PIVNUM)
...
INSTEP          CLEAR           S
                SUB             #3
                STA             TMPIND      .TMPNXT에서 3씩 빼서 TMPIND에 저장
                LDT             PIVNUM      .T reg에 PIVNUM 저장, A reg에 TMPIND저장
                LDA             TMPIND
                COMP            #STR1       .STR1이랑 비교해서 마지막까지 왔는 지 chk
                JEQ             NOEND
```

```
            JGT     NOEND
            LDS     #3          .마지막 값인데 작다로 끝날 경우에 그냥 넣어야되니까
.NO END
..끝이 아니라면 Pivot 값 왼쪽으로 3씩 움직이면서 작은게 나오면 멈추고(왼쪽은 정렬되어있기때문에)
..해당 자리에 값을 넣어주고 다 한칸씩 밀어준다.
..주소값을 이동해야해서 Indirect Addressing을 사용한다.
NOEND       LDA     @TMPIND
            COMPR   A, T
            JLT     LOSTEP
            .A가 크면?
            .@(TMPIND+3)(TMPNXT) = @TMPIND
            LDA     @TMPIND
            STA     @TMPNXT
            LDA     PIVNUM
            STA     @TMPIND
            CLEAR   A
            COMPR   A, S        .S가 크다는 말은 끝이라는 말이니까 저장하는 LOSTEP으로 간다.
            JGT     LOSTEP
            LDA     TMPIND
            STA     TMPNXT
            J       INSTEP
..LOw STEP
..Pivot보다 작은 값이 나오면 멈추고 그 자리에 값을 넣어준다.
..작은 값을 발견하거나 마지막일 경우 해당 값이 그 순서이거나 제일 작기때문에 저장한다.
LOSTEP      LDA     TMPNXT
            STA     TMPNXT
            LDA     PIVNUM
            STA     @TMPNXT
            LDA     PIVIND
            ADD     #3
            STA     PIVIND
            CLEAR   A
            ADDR    L, A        .Return Address를 잠깐 RETADD에 저장해둔다.
            STA     RETADD
            LDA     #10         .줄바꿈
            WD      OUTDEV
            LDA     #91         .'[' ASCII CODE
            WD      OUTDEV
            CLEAR   X
            JSUB    ENDINP
            LDA     #32         .공백
            WD      OUTDEV
            LDA     #93         .']' ASCII CODE
            WD      OUTDEV
            LDL     RETADD      .Return Address를 돌려준다.

.Pivot이 마지막 값인지 체크
            LDA     NUMADD
            ADD     #3
            COMP    PIVIND
            JGT     INSERT
            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
```

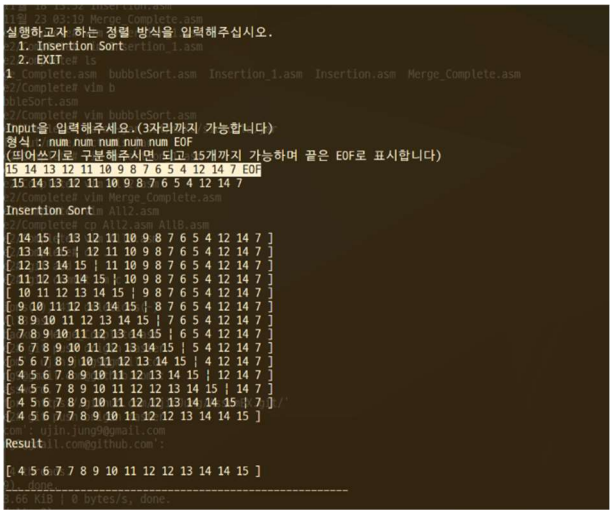
```
            WD      OUTDEV
            TIX     RESLEN
            JLT     RESPRT
            CLEAR   A
            ADDR    L, A
            STA     RETADD
            LDA     #10
            WD      OUTDEV
            LDA     #91
            WD      OUTDEV
            CLEAR   X
            JSUB 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


ZERO        WORD    0                   .ZERO 필요없지 않나?
STAADD      RESW    1                   .각 SAMPLE Input 시작 주소 Index값
NUMADD      RESW    1                   .각 SAMPLE Input 숫자 시작 주소 Index값
                                        .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                   .Junk인지 chk하기 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
            RESW    1
            RESW    1
STRLEN      RESW    1
FIGURE      WORD    100
INDEV       BYTE    X'00'
OUTDEV      BYTE    X'01'
RETADD      RESW    1               .Main으로 Return Address
PREADD      RESW    1               .Print하고 Sort로 Return Address

CHOTXT      BYTE    C'  실행하고자 하는 정렬 방식을 입력해주십시오.'
            BYTE    10
            BYTE    C'  1. Insertion Sort'
            BYTE    10
            BYTE    C'  2. EXIT'
            BYTE    10
CHOLEN      RESW    1
IMSTXT      BYTE    C'  Input을 입력해주세요.(3자리까지 가능합니다)'          .Input MaSsage Text
```

```
        BYTE    10
        BYTE    C'형식 : num num num num num EOF'
        BYTE    10
        BYTE    C'(띄어쓰기로 구분해주시면 되고 15개까지 가능하며 끝은 EOF로 표시합니다)'
        BYTE    10
IMSLEN  RESW    1
RESTXT  BYTE    C'   Result'
        BYTE    10
RESLEN  RESW    1
INSTXT  BYTE    C'   Insertion Sort'
        BYTE    10
INSLEN  RESW    1
```

3. 실행

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

2. Input : 15 14 13 12 11 10 9 8 7 6 5 4 12 14 7 EOF



주요 기능

```
.....Insertion Sorting Ready.....

INSMSG    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      #INSLEN
          SUB      #INSTXT
          STA      INSLEN
INSPRT    LDA      INSTXT,  X
          WD       OUTDEV
          TIX      INSLEN
          JLT      INSPRT
```

Insertion 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.

Insert Text 를 Print 한다.

```
.....Insertion Sort Processing.....
....큰 LOOP
INSRDY    CLEAR    A
          CLEAR    X
          STA      TMPNUM
          LDA      #STR2
          STA      PIVIND

INSERT    CLEAR    S
          LDA      @PIVIND
          STA      PIVNUM
          LDA      PIVIND
          STA      TMPNXT

INSTEP    CLEAR    S
          SUB      #3
          STA      TMPIND      .TMPNXT에서 3씩 빼서 TMPIND에 저장
          LDT      PIVNUM      .T reg에 PIVNUM 저장, A reg에 TMPIND저장
          LDA      TMPIND
          COMP     #STR1       .STR1이랑 비교해서 마지막까지 왔는 지 chk
          JEQ      NOEND
          JGT      NOEND
          LDS      #3          .마지막 값인데 작다로 끝날 경우에 그냥 넣어야되니까


NOEND     LDA      @TMPIND
          COMPR    A, T
          JLT      LOSTEP
          .A가 크면?
          .@(TMPIND+3)(TMPNXT) = @TMPIND
          LDA      @TMPIND
          STA      @TMPNXT
          LDA      PIVNUM
          STA      @TMPIND
          CLEAR    A
          COMPR    A, S        .S가 크다는 말은 끝이라는 말이니까 저장하는 LOSTEP으로 간다.
          JGT      LOSTEP
          LDA      TMPIND
          STA      TMPNXT
          J        INSTEP
```

전체 Input 을 chk 하는 큰 Loop

pivot 을 기준으로 왼쪽의 값들과 비교하는 작은 Loop 로 나뉜다.

Instep 에서 Pivot 과 비교되어지는 값인 temp 값을 한칸씩 왼쪽으로 이동한다.

이동 후 마지막 값인지 chk 하고 마지막일 경우에는 비교 후 무조건 넣어야 하기 때문에 S register 에 3 을 넣어준다.

어려웠던 점

1. 처음 Input을 받는 과정이 어려웠다.

아직 정확한 이유는 모르지만 RD INDEV 로 하고 INDEV 레이블에서 INPUT 디바이스 아이디를 받아올 경우 frequency가 빠른 상태에서는 읽지 못하고 넘어가버리는 현상이 발생했다. 이런 현상때문에 immediate 방식을 이용해 RD #0 와 같은 방식으로 코드를 작성하였다.

2. Input을 받아 온 뒤 어떻게 처리할 것인가의 문제

처음으로 시도해 본 방법은 shift하는 방식을 이용해 4bit에 숫자 한자리 씩 들어가는 방식이었다. 하지만 값을 비교하는 과정에서 단순히 더해 진행하는 과정보다 오래 걸려 제외하였다.

두번째로 시도해 본 방법은 값을 받아와 10진수로 변환하는 방법이었다. Sictool 내부에서 보기도 편했고 자릿수 계산하기도 편해 이 방법을 써 코드를 작성하였다. 공백이 나올 때까지 숫자를 입력받도록 설계되어있고 숫자가 들어올 때마다 기존 수에 10을 곱한 뒤 들어온 값을 더해주어 10진수로 나타내준다.