System Programming and Practice Assignment2

Student ID : 201721083
Name : 김태석

```
MAIN    START   1000
.스택 및 스택의 사이즈 크기 설정
FIRST   LDA     #STACK
        STA     TOP
        LDA     ZERO
        STA     SIZE
.A B C D 값을 반복하여 입력받기
LOOP    JSUB    INLOOP
        LDX     ZERO
        LDA     LENGTH
        COMP    ZERO
        JEQ     LOOP
        LDCH    TYPE,X
        COMP    ALPHA_D
        JSUB    WHICH3
        J       LOOP

INLOOP  LDX     ZERO
        LDA     ZERO
        STA     TEMP
INER    LDA     TEMP
        COMP    #2
        JEQ     GOBACK
        ADD     #1
        STA     TEMP
        TD      STDIN
        JEQ     INER
        RD      STDIN
        STCH    TYPE,X
        TIX     MAXLEN
        JLT     INER

GOBACK  STX     LENGTH
        RSUB

.입력받은 알파벳이 무엇인지 판별하기
WHICH3  JEQ     D
        JLT     WHICH2

WHICH2  COMP    ALPHA_C
        JEQ     C
        JLT     WHICH1

WHICH1  COMP    ALPHA_B
        JEQ     B
        JLT     A
.A기능 실행
A       LDA     SIZE
        COMP    ZERO
        JEQ     EMPTY
        JGT     A_GO    .스택이 없다면 EMPT출력
A_GO    STL     TEMPL
        JSUB    POP
        LDA     @TOP
        STA     TEMP
        LDA     ZERO
        STA     @TOP
        LDA     TEMP
        COMP    ASEN
        JEQ     A_TEN
        JGT     A_TEN
        JLT     A_ONE

A_ONE   ADD     #48
        WD      STDOUT
        LDA     ASEN
        WD      STDOUT
        LDL     TEMPL
        RSUB            .스택의 수가 1의자리 수일때

A_TEN   STA     TEMP1
        DIV     #10
        ADD     #48
        WD      STDOUT
        SUB     #48
        MUL     #10
        STA     TEMP10
        LDA     TEMP1
        SUB     TEMP10
        ADD     #48
        WD      STDOUT
        LDA     ASEN
        WD      STDOUT
        LDL     TEMPL
        RSUB            .스택의 수가 10의자리 수일때
```

-코드 설명-

코드는 총 A,B,C,D 기능을 수행하는 부분, 알파벳을 반복적으로 입력 받는 부분, 필요한 상수들을 선언해 놓은 부분, STACK의 POP과 PUSH를 구현한 부분, 문구를 출력해주는 부분으로 나뉜다. 시작은 알파벳을 입력 받는 부분에서 시작하며 A에선 스택에 최상단에 쌓여있는 값을 출력함과 동시에 스택에서 POP을 해주는 기능을, B에선 값을 입력 받고 입력 받은 값이 0~99의 값인지를 판별하고 그를 스택에 저장하는 기능을, C에선 스택 최상단의 값을 단순 출력해주는 기능을, D에선 스택의 크기를 출력해주는 기능을 구현하였다. PUSH와 POP에선 직접 값을 넣고 빼는 것이 아니라 스택의 포인터가 어디를 가르키는지 TOP변수에 저장해주는 기능을 구현하였다.

-알고리즘-

처음은 A,B,C,D의 알파벳을 판별하여 해당 부분으로 J하는 것으로 시작한다. 이를 위해 입력받은 값의 아스키 코드값을 비교하여 해당 값을 비교했을 때 COMP,JEQ,JLT,JGT 를 통하여 원하는 기능을 실행 시킬 수 있다.

시작 시 스택의 사이즈 값을 0으로 설정 하여 스택에 아무것도 넣지 않았을 경우 A,C기능을 실행 할 경우 EMPTY문구를 출력하게 되며 이는 COMP를 통해 ZERO와 스택 사이즈 값을 비교하여서 같을 경우 문구를 출력하는 방식을 사용했다. 기능 A에 대한 알고리즘을 먼저 설명하자면 출력은 바이트단위로 실시 되기 때문에 스택 TOP이 가르키는 값이 1의자리인지 10의 자리인지를 판별 한 후 1의 자리인경우 아스키코드값에 해당하는 문자를 출력하기 위하여 해당 정수값에 ADD 48을 한 후 WD를 통한 출력을 시켜주고 POP을 실시하여 스택 TOP값을 바꿔 준 후 해당 정수가 들어있던 스택에 0을 저장하여 스택을 비워주었다. 기능 B의 경우 입력 받은 정수를 버퍼에 저장 후 버퍼 속 값을 확인하여 해당 값이 0~99의 값이 아니라면 에러로 간주한 후 문구 출력과 동시에 프로그램을 끝내주었고 0~99의 값이라면 값이 1의자리 수인지 10의자리 수 인지 판별 후 1의 자리수라면 아스키코드 값에 SUB 48을 해주어 정수로 만들어준 후 스택에 저장하고 10의자리 수라면 DIV 10, MUL 10을 같이 활용하여 정수로 만들어 준 후 스택에 저장 해 주었다. 또한 스택에 PUSH할 때 마다 스택사이즈가 1씩 증가하게 되는데 만약 스택 사이즈가 5가 된다면 PUSH 직후 스택 5칸의 수 모두 POP해 주며 출력을 해 주었다. 이 때 A의 기능을 활용

```
.B기능 실행
B       LDA     SIZE
        ADD     #1
        STA     SIZE
        LDA     ZERO
        STA     TEMP
        STA     TEMP1
        STA     TEMP10
        STA     BUFFER
        LDX     #1
        STCH    BUFFER,X
        LDX     ZERO
        STCH    BUFFER,X

RLOOP   TD      STDIN
        JEQ     RLOOP
        RD      STDIN
        COMP    ASEN
        JEQ     CHECK
        STCH    BUFFER,X
        TIX     #3
        JLT     RLOOP
        JEQ     CHECK
        JGT     CHECK

CHECK   LDX     #2
        LDCH    BUFFER,X
        COMP    ASEN
        JEQ     STORE   .엔터값이 있다면 10의자리수
        JLT     STORE   .엔터값보다 작다면 1의자리수
        JGT     CANT    .엔터값보다 크다면 오류

STORE   LDX     #1      .1 혹은 10의자리수일때 저장
        LDCH    BUFFER,X
        COMP    ASEN
        JLT     ONE
        JGT     TEN

ONE     LDX     #0
        LDCH    BUFFER,X
        SUB     #48
        STA     TEMP1
        STA     TEMP
        STL     TEMPL
        JSUB    B_SEMI
        LDL     TEMPL
        RSUB            .1의자리 수 일때의 저장
```

```
TEN     LDX     #0
        LDCH    BUFFER,X
        SUB     #48
        MUL     #10
        STA     TEMP10
        LDX     #1
        LDCH    BUFFER,X
        SUB     #48
        STA     TEMP1
        ADD     TEMP10
        STA     TEMP
        STL     TEMPL
        JSUB    B_SEMI
        LDL     TEMPL
        RSUB            .10의자리 수 일때의 저장

B_SEMI  STL     TEMPL2
        LDA     TEMP
        STA     @TOP
        JSUB    PUSH
        LDA     SIZE
        COMP    #5
        JEQ     B_FINAL .스택이 다 찼다면 모두POP
        LDL     TEMPL2
        RSUB            .스택에 값 저장해주기

B_FINAL JSUB    A_GO
        JSUB    A_GO
        JSUB    A_GO
        JSUB    A_GO
        JSUB    A_GO    .POP 5회 실행 후 프로그램 종료
EOF     J       EOF
```

```
.C기능 실행
C       LDA     SIZE
        COMP    ZERO
        JEQ     EMPTY
        JGT     C_GO
C_GO    STL     TEMPL
        LDA     TOP
        SUB     #3
        STA     T_TOP
        LDA     @T_TOP
        STA     TEMP
        COMP    #10
        JEQ     C_TEN
        JGT     C_TEN
        JLT     C_ONE

C_ONE   ADD     #48
        WD      STDOUT
        LDA     ASEN
        WD      STDOUT
        LDL     TEMPL
        RSUB

C_TEN   STA     TEMP1
        DIV     #10
        ADD     #48
        WD      STDOUT
        SUB     #48
        MUL     #10
        STA     TEMP10
        LDA     TEMP1
        SUB     TEMP10
        ADD     #48
        WD      STDOUT
        LDA     ASEN
        WD      STDOUT
        LDL     TEMPL
        RSUB

.D기능 실행
D       LDA     SIZE    .사이즈값을 출력해주기
        ADD     #48
        WD      STDOUT
        LDA     ASEN
        WD      STDOUT
        RSUB
```

```
PUSH    LDA     TOP
        ADD     #3
        STA     TOP
        RSUB            .스택에 PUSH후 TOP의 주소값변경

POP     LDA     TOP
        SUB     #3
        STA     TOP
        LDA     SIZE
        SUB     #1
        STA     SIZE
        RSUB            .스택에서 POP후 TOP의 주소값변경

.사용할 각 변수 및 상수 선언
STACK   RESW    5
TOP     RESW    1
T_TOP   RESW    1
SIZE    RESW    1

STDIN   BYTE    0
STDOUT  BYTE    1
ALPHA   BYTE    0
TYPE    RESB    3

MAXLEN  WORD    4096
ASEN    WORD    10
ALPHA_A WORD    65
ALPHA_B WORD    66
ALPHA_C WORD    67
ALPHA_D WORD    68
ZERO    WORD    0

BUFFER  RESB    4096

TEMP    RESW    1
TEMP1   RESW    1
TEMP10  RESW    1
TEMPL   RESW    1
TEMPL2  RESW    1
LENGTH  RESW    1
```

하여 코드 줄 수를 줄일 수 있었다. 기능 C에서는 A와 상당히 유사하지만 POP기능을 제외하고 같은 방식으로 구현하였고 기능 D에서는 B를 통해 증가된 스택 사이즈를 ADD 48을 통해 아스키 코드값으로 변경 한 후 출력하도록 하였다. 각 기능들을 구현하면서 JSUB를 사용할 일이 잦았는데 JSUB 이후 주소를 원하는 곳으로 다시 되돌려야 하는 경우가 상당 수 존재했다. 이 때 factorial 실습에서 코드를 보며 이해했던 STL을 활용하여 JSUB를 반복 실행하기 전 이전 JSUB을 통해 L레지스터에 저장 되어 있던 주소값을 임시 변수 TEMPL에 저장해 준 후 JSUB실행 후 RSUB로 돌아 왔을 때 LDL 명령을 통해 다시 L 레지스터에 돌아가고자 했던 주소값을 올린 후 RSUB를 실행 시켜 주는 것을 통해 해당 문제를 해결할 수 있었다. (사진 첨부에 단순 문자 출력 부분은 게시하지 않았습니다.)

```
imtaeseok@MacBook-TaeSeok Sictools % java -jar out/make/sictools.jar
B
890
ONLY0~99
```

-0~99의 값이 아닐 경우 해당 문구 출력 후 프로그램 종료-

```
kimtaeseok@MacBook-TaeSeok Sictools % java -jar out/make/sictools.jar
A
EMPTY
C
EMPTY
D
0
B
78
```

Name	Address	Decimal	Hex	Char
Watch				
STACK	00697	78	00004E	N
STACK[1]	0069A	0	000000	
STACK[2]	0069D	0	000000	
STACK[3]	006A0	0	000000	
STACK[4]	006A3	0	000000	

-B로 입력한 값이 없어 A,C를 실시했을 때 EMPTY가 뜨게 된다. 또한 B를 통해 올바른 값을 입력했을 경우 해당 스택에 값이 쌓이게 됨을 확인 할 수 있다.-

```
 D
1
A
78
C
EMPTY
```

Watch			
STACK	00697	0	000000
STACK[1]	0069A	0	000000
STACK[2]	0069D	0	000000

-D를 통해 사이즈를 확인하면 1이 증가된 것을 확인 할 수 있고 A를 실행시켜 스택에서 값을 POP해주면 스택에 있던 78이 출력되고 해당 스택은 비어있는 상황이 됨을 알 수 있다. 또한 C를 통해 다시금 스택이 비어있는 것을 확인 할 수 있다.-

```
 B
20
B
2
B
48
B
12
B
7
7
12
48
2
20
```

-B를 통해 5번의 PUSH를 하게되면 스택이 꽉 차게 되므로 자동으로 5번 모두 POP을 하며 프로그램은 종료된다.-

에로사항 : JSUB를 여러번 사용할 경우 주소값이 꼬이게되는 문제가 있었지만 Factorial 실습 때 배운 예제 속 STL을 활용하여 이를 해결 할 수 있었다. 또한 STL로 저장한 주소값을 LDL 명령어를 통해 다시 L 레지스트리에 저장하는 방법을 새로 터득했다. 또 한가지 어려웠던 점은 바이트단위로 문자를 입력 받기 때문에 각 정수를 입력 받는 데에 자릿수 및 버퍼에 들어가는 문자 값을 확인하기에 어려웠고 아스키 코드값으로 출력을 해 주어야 하는 문제 때문에 문자 출력시 제대로된 문자가 출력되지 않는 경우가 많았다. 이는 sictool에서 step을 누르며 주소값이 원하는 흐름대로 흘러가는지 확인하고 A레지스터로 들어가 있는 값을 하나씩 확인해 가며 고쳐 갈 수 있었고, 아스키 코드표를 참고하며 적정값을 더하고 빼는 것을 통해 문제를 해결 할 수 있었다.